UNIVERSIDAD DE GRANADA

DEPARTAMENTO DE
TEORÍA DE LA SEÑAL, TELEMÁTICA Y COMUNICACIONES

ATAQUES DE DENEGACIÓN DE SERVICIO A BAJA TASA CONTRA SERVIDORES

TESIS DOCTORAL

GABRIEL MACIÁ FERNÁNDEZ

Editor: Editorial de la Universidad de Granada
Autor: Gabriel Maciá Fernández
D.L.: Gr. 1308 - 2007
ISBN: 978-4-338-4368-5

UNIVERSIDAD DE GRANADA

DEPARTAMENTO DE
TEORÍA DE LA SEÑAL, TELEMÁTICA Y COMUNICACIONES

ATAQUES DE DENEGACIÓN DE SERVICIO A BAJA TASA CONTRA SERVIDORES

TESIS DOCTORAL

GABRIEL MACIÁ FERNÁNDEZ

Mayo, 2007

D. Pedro García Teodoro y **D. Jesús Esteban Díaz Verdejo**
Profesores Titulares de Ingeniería Telemática del
Departamento de Teoría de Señal, Telemática y Comunicaciones de la
Universidad de Granada

CERTIFICAN:

Que la presente memoria, titulada "Ataques de denegación de servicio a baja tasa contra servidores", ha sido realizada por D. Gabriel Maciá Fernández bajo nuestra dirección en el Departamento de Teoría de Señal, Telemática y Comunicaciones de la Universidad de Granada. Esta memoria constituye la Tesis que D. Gabriel Maciá Fernández presenta para optar al grado de Doctor Ingeniero en Telecomunicaciones.

Granada, a 18 de mayo de 2007

Fdo.: Dr. D. Pedro García Teodoro Director de la Tesis	Fdo.: Dr. D. Jesús Esteban Díaz Verdejo Director de la Tesis

Agradecimientos

Es difícil agradecer lo suficiente a todos los que, de alguna u otra manera, han contribuido a la realización de este trabajo.

Con certeza, la confianza y el apoyo recibido por parte de los directores de esta tesis, D. Jesús Esteban Díaz Verdejo y D. Pedro García Teodoro, han sido, en incontables momentos, decisivos para que se haya podido llevar a término. Ellos han sabido ser, a la vez, maestros y amigos durante todo este tiempo.

También quiero agradecer su apoyo constante a mis compañeros del grupo de investigación en Señales, Telemática y Comunicaciones de la Universidad de Granada, con los que he compartido tantos momentos durante la realización de esta tesis.

Al grupo de investigación en seguridad del departamento de informática y comunicaciones de la Università degli Studi di Milano y, en especial, al profesor Bruschi y a Lorenzo Cavallaro, Lorenzo Martignoni y Andrea Lanzi; a ellos quiero agradecer su cálida acogida, consejos y enseñanzas, sin las que el grado de doctor europeo para esta tesis no hubiera sido posible.

A mi familia y amigos, que han sabido soportar mis faltas y ausencias, y con los que he podido compartir los sinsabores diarios en la realización de este trabajo.

A Marta, mi eterna compañera, y a mis hijos, quienes tanto han compartido mi esfuerzo durante estos años que, en realidad, casi es injusto que no aparezcan como propios autores.

Y finalmente a Dios, sin el que nada sería igual…

Granada, mayo de 2007
El autor

A Marta, con mi eterna gratitud y devoción

... porque cuanto es el hombre ante Dios, tanto es y no más.

S. Francisco de Asís (Adm. 19,2)

Índice general

Prólogo **1**

1 Introducción a la seguridad y los ataques de denegación de servicio **3**

1.1 Introducción 3

1.2 La seguridad en las redes de comunicación 4

1.2.1 Políticas y modelos de seguridad 5

1.2.2 Servicios y mecanismos básicos de la seguridad 6

1.3 Ataques a la seguridad 9

1.3.1 Clasificación y tipos de ataque 10

1.4 Los ataques de denegación de servicio 12

1.4.1 Conceptos generales 12

1.4.2 Funcionamiento de los ataques DoS 13

1.4.3 Ataques de denegación de servicio a baja tasa 21

1.5 Métodos de defensa para los ataques de denegación de servicio 22

1.5.1 Prevención de ataques DoS 23

1.5.2 Estrategias de detección de ataques DoS 26

1.5.3 Mecanismos de respuesta ante los ataques DoS 29

1.6 Conclusiones del capítulo 31

2 Modelado del escenario de ataque **33**

2.1 Introducción 33

2.2 Escenario de estudio 34

2.3 Modelado del servidor 35

2.3.1 Naturaleza del servidor . . . 35
2.3.2 Modelo del servidor . . . 39
2.3.3 Aplicabilidad del modelo . . . 43
2.3.4 Tiempo entre salidas consecutivas en el servidor . . . 44
2.3.5 Distancia de superposición . . . 56
2.3.6 Sistemas servidores con superposición y sin superposición . . . 57
2.4 Modelado de los usuarios legítimos del sistema . . . 59
2.5 Conclusiones del capítulo . . . 61

3 El ataque DoS a baja tasa contra servidores 63
3.1 Introducción . . . 63
3.2 Especificación del ataque DoS a baja tasa contra servidores . . . 64
3.2.1 Estrategia general para la realización del ataque . . . 64
3.2.2 El periodo básico de ataque . . . 66
3.2.3 Características del ataque DoS de baja tasa contra servidores . . . 69
3.2.4 Comparación con otros ataques de baja tasa . . . 70
3.3 Indicadores de estimación del rendimiento del ataque . . . 72
3.4 Modelos matemáticos para el porcentaje de tiempo disponible . . . 73
3.4.1 Modelo matemático para el porcentaje de tiempo disponible en sistemas sin superposición . . . 74
3.4.2 Modelo matemático para el porcentaje de tiempo disponible en sistemas con superposición . . . 77
3.4.3 Conclusiones sobre los modelos para el porcentaje de tiempo disponible 96
3.5 Modelo matemático para la disponibilidad . . . 98
3.6 Modelo matemático para la sobrecarga . . . 101
3.7 Conclusiones del capítulo . . . 102
Apéndice 1. Cálculo del tiempo disponible en los diferentes tramos . . . 103

4 El ataque DoS a baja tasa contra servidores iterativos 115
4.1 Introducción . . . 115
4.2 Implementación del ataque contra servidores iterativos . . . 116
4.2.1 Fases del ataque . . . 119
4.3 Evaluación del rendimiento del ataque . . . 121

4.3.1 Variaciones con los parámetros de diseño . . . 121
4.3.2 Capacidades del ataque . . . 130
4.4 Validación en entornos reales . . . 132
4.5 Mejoras al ataque . . . 134
4.5.1 Algoritmo de media adaptativa . . . 135
4.6 Conclusiones de este capítulo . . . 137

5 El ataque DoS a baja tasa contra servidores concurrentes 139
5.1 Introducción . . . 139
5.2 Implementación del ataque contra servidores concurrentes . . . 140
5.2.1 Análisis de la vulnerabilidad en los sistemas concurrentes . . . 140
5.2.2 Procedimiento de ejecución del ataque . . . 147
5.3 Evaluación del rendimiento del ataque . . . 153
5.3.1 Variaciones con los parámetros de diseño . . . 154
5.3.2 Capacidades del ataque . . . 161
5.3.3 Validación en entornos reales . . . 165
5.4 Mejoras al ataque . . . 166
5.4.1 Estrategia de compartición de información entre hebras . . . 168
5.4.2 Estrategia del umbral de capturas . . . 178
5.5 Conclusiones del capítulo . . . 187

6 Conclusiones 189

A Thesis Summary 195
A.1 Introduction . . . 195
A.2 Scenario modeling . . . 197
A.3 Main strategy for the attack . . . 199
A.3.1 Vulnerability in iterative servers . . . 200
A.3.2 Vulnerability in concurrent servers . . . 204
A.4 Low-rate DoS attack specification . . . 205
A.4.1 Phases of the attack . . . 207
A.4.2 Execution against iterative servers . . . 208
A.4.3 Execution against concurrent servers . . . 209

A.5 Evaluation of the attack 210
A.5.1 Indicators for evaluating the attack 210
A.5.2 Mathematical models for the evaluation of the attack 211
A.5.3 Evaluation of the attack against iterative servers 211
A.5.4 Evaluation of the attack against concurrent servers 215
A.6 Conclusions and future work 217
References 221

Bibliografía **223**

Índice de figuras

1.1 Amenazas a la seguridad [Stallings, 2003b]. 11

1.2 Ataques de seguridad activos y pasivos [Kent, 1997]. 12

1.3 Diagrama de la arquitectura gestor/agente para la ocultación del ataque DDoS [Mirkovic et al., 2004]. 18

1.4 Ilustración de una máquina actuando como salto para un ataque DDoS. . . . 19

2.1 Escenario de estudio para los ataques DoS a baja tasa contra servidores. . . . 35

2.2 Clasificación de los servidores atendiendo al modo en que prestan el servicio. 36

2.3 Entorno multiservidor con balanceo de carga a la entrada. 38

2.4 Modelo genérico propuesto para el servidor. 40

2.5 Servidor monoproceso: diagrama de los tiempos entre salidas consecutivas y estado del procesamiento de varias peticiones cuando el tiempo de servicio es constante. 46

2.6 Simulación de funcionamiento de servidor monoproceso con la cola de servicio siempre ocupada: (a) valores de tiempo entre salidas consecutivas, τ_u^{mono}, y (b) histograma de las muestras. 48

2.7 Simulación del escenario en el que la cola de servicio del servidor monoproceso puede estar vacía: (a) valores de tiempo entre salidas, τ_u^{mono}, (curva negra) y ocupación de la cola de servicio (curva gris) y (b) histograma de las muestras obtenidas. 50

2.8 Evolución temporal de las salidas y sus funciones de probabilidad de ocurrencia en un sistema servidor monoproceso ($N_s = 1$), donde $f_j(t)$ se supone de distribución normal. 51

2.9 Evolución temporal de las salidas y sus correspondientes funciones de probabilidad de ocurrencia en un servidor multiproceso ($N_s = 2$). Representación de la evolución temporal independiente de los elementos de procesamiento 1 y 2, y de la conjunta de ambos elementos de procesamiento. La probabilidad de ocurrencia de una salida se ha supuesto con distribución normal. 52

2.10 Histograma de los tiempos entre salidas estimados en un sistema multiproceso con $T_s = \mathcal{N}(12\ s; 0, 3)$, para diferentes valores de N_s. 54

2.11 Histograma de los tiempos entre salidas estimados en un sistema multiproceso con $\overline{T_s} = 12$ y $N_s = 4$, para diferentes valores de la varianza, $var[T_s]$. 55

2.12 Diagrama de salidas y sus $f_j(t)$ asociadas correspondientes a dos elementos de procesamiento diferentes y representación de los valores instantáneos de s y τ_s. 56

2.13 Funciones de probabilidad de las variables aleatorias s y τ_s, para una configuración dada del servidor. 57

2.14 Diagrama con dos salidas consecutivas: (a) con superposición y (b) sin superposición. 59

2.15 Escenario con tráfico agregado procedente de N_u usuarios y su escenario equivalente con un único usuario. 60

3.1 Periodo básico de ataque: forma de onda y parámetros. 67

3.2 Esquema de la curva de probabilidad de ocurrencia de una salida con un periodo de ataque y sus puntos de cálculo asociados. 75

3.3 Esquema de dos salidas con superposición: funciones de probabilidad de ocurrencia de las salidas, puntos de cálculo, zonas de influencia y tramos de cálculo del tiempo disponible para una distancia de superposición s. 81

3.4 Escenario de llegada de mensajes de ataque separados por una distancia Δ, donde el tiempo total disponible generado es $T_1 + T_2 + T_3$. 89

3.5 Representación gráfica de la función $t_D^{\alpha}(s)\big|_{i\in\{A,G\}}$. 91

3.6 Representación gráfica de la función $t_D^{\alpha}(s)\big|_{B}$. 92

3.7 Escenario de aparición de un mensaje adicional de ataque en un intervalo debido a la generación de una salida en un intervalo anterior. 93

3.8 Escenario de aparición de mensajes de ataque adicionales. Determinación del instante de llegada del siguiente mensaje de ataque para una salida que se produce en t. 94

3.9 Tiempo disponible para todos los tramos para el ejemplo de ataque con la configuración mostrada en la Tabla 3.2. 98

3.10 Tiempo disponible para el tramo A, $T_D^A(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2. 104

3.11 Tiempo disponible para el tramo B, $T_D^B(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2. 105

3.12 Tiempo libre para el tramo C, $T_D^C(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2. 107

3.13 Tiempo disponible para el tramo D, $T_D^D(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2. 109

3.14 Tiempo disponible para el tramo E, $T_D^E(s)$, calculado para el ataque con la configuración indicada en la Tabla 3.2. 111

3.15 Tiempo disponible para el tramo F, $T_D^F(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2. 112

3.16 Tiempo disponible para el tramo G, $T_D^G(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2. 113

4.1 Periodo básico de ataque en el ataque DoS a baja tasa contra servidores iterativos: forma de onda y parámetros. 118

4.2 Diagrama con la ejecución de dos periodos básicos de ataque consecutivos en el tiempo. 121

4.3 Evolución de los indicadores D y T_D, ante la variación de los parámetros: a) t_{ontime}; b) intervalo, Δ; c) tiempo medio de ida y vuelta, $\overline{RTT}$; d) varianzas del tiempo de servicio y del tiempo de ida y vuelta, $var[T_s] + var[RTT]$; e) media del tiempo de servicio, $\overline{T_s}$; y f) tiempo medio entre llegadas de usuario, $\overline{T_a}$. 123

4.4 Evolución del indicador S ante la variación de los parámetros: a) t_{ontime}; b) intervalo, Δ; c) tiempo medio de ida y vuelta, $\overline{RTT}$; d) varianzas del tiempo de servicio y del tiempo de ida y vuelta, $var[T_s] + var[RTT]$; e) media del tiempo de servicio, $\overline{T_s}$; y f) tiempo medio entre llegadas de usuario, $\overline{T_a}$. . . . 124

4.5 Valores de T_D obtenidos mediante simulación y mediante el modelo matemático para 13 escenarios diferentes: (a) valores absolutos y (b) diferencias relativas con respecto al valor aportado por el modelo. 127

4.6 Valores de D obtenidos mediante simulación y mediante el modelo matemático para 13 escenarios diferentes: (a) valores absolutos y (b) diferencias relativas con respecto al valor aportado por el modelo. 128

4.7 Valores de S obtenidos mediante simulación y mediante el modelo matemático para 13 escenarios diferentes: (a) valores absolutos y (b) diferencias relativas con respecto al valor aportado por el modelo. 129

4.8 Valor de los indicadores de rendimiento para un ataque ejemplo. 130

4.9 Disponibilidad, D, *vs.* sobrecarga, S, para 25 configuraciones diferentes (valores de t_{ontime} y Δ) del ataque a baja tasa contra servidores iterativos. . . . 131

4.10 Comparación entre la estrategia inteligente y la *naïve* para tres configuraciones de ataque diferentes. . . . 131

4.11 Comparación entre la estrategia inteligente y *naïve* en un servidor saturado por el tráfico de los usuarios. . . . 132

4.12 Rendimiento del ataque para tres escenarios cuando se aplica el algoritmo de media adaptativa y cuando éste no se aplica: a) disponibilidad, D, y b) sobrecarga, S. . . . 136

5.1 Servidor HTTP persistente y temporizador de cierre de la conexión. . . . 143

5.2 Diagrama del procedimiento seguido para la estimación del instante de una salida en un servidor HTTP persistente. . . . 144

5.3 Diagrama con la superposición de dos periodos de ataque. . . . 148

5.4 Escenario de capturas en el que las hebras capturan posiciones que no son las que pretenden inicialmente. . . . 150

5.5 Estrategia de interrupción de *ontime* cuando se realiza una captura. . . . 152

5.6 Resultados del tiempo disponible obtenido para 11 escenarios diferentes para las dos posibles opciones de comportamiento en la estrategia básica cuando el atacante recibe una captura: (a) continuar con la fase de *ontime* cuando se captura una posición, y (b) finalizar la fase de *ontime* ante la captura de una posición. . . . 153

5.7 Comportamiento del ataque con respecto a variaciones en el intervalo de recuperación, Δ_r. . . . 155

5.8 Comportamiento del ataque básico con respecto a la variación del número de hebras de ataque, N_a. . . . 156

5.9 Comportamiento del ataque básico con respecto a variaciones en las desviaciones en la sincronización entre la llegada del periodo de ataque al servidor y el instante en que se produce la salida, $var[T_{out}] + var[RTT]$. . . . 157

5.10 Comportamiento del ataque básico con respecto a modificaciones en los valores del intervalo, Δ. . . . 158

5.11 Comportamiento del ataque básico con respecto a variaciones en el parámetro de ataque t_{ontime}. . . . 158

5.12 Comparativa de valores absolutos de tiempo disponible obtenidos mediante simulación y mediante el modelo matemático para 16 escenarios diferentes: a) valores obtenidos en ambos casos y b) diferencias absolutas entre ambos valores para cada escenario. . . . 160

5.13 Comparativa de valores absolutos de D obtenidos mediante simulación y mediante el modelo matemático para 12 escenarios diferentes: a) valores obtenidos en ambos casos y b) diferencias absolutas entre ambos valores para cada escenario. . . . 162

5.14 Comparativa de valores absolutos de S obtenidos mediante simulación y mediante el modelo matemático para 13 escenarios diferentes: a) valores obtenidos en ambos casos y b) diferencias absolutas entre ambos valores para cada escenario. 163

5.15 Rendimientos obtenidos para 18 configuraciones diferentes de ataque. 164

5.16 Comparación de eficiencia (D) obtenida con un ataque naïve y el aquí propuesto, para 4 escenarios diferentes. 165

5.17 Esquema de superposición entre dos hebras en el que una captura dos salidas y la otra ninguna. 169

5.18 Diagrama de flujo para el funcionamiento de la estrategia de compartición de información entre hebras (ITIS). 171

5.19 Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía el número de hebras de ataque, N_a, para cuatro valores diferentes de N_s. 173

5.20 Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía el intervalo de recuperación, Δ_r, para tres valores de N_s. . . 174

5.21 Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía $var[T_{out}] + var[RTT]$, para cuatro valores de N_s. 175

5.22 Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía la duración de t_{ontime}, para tres valores de N_s. 176

5.23 Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía el intervalo, Δ, para tres valores de N_s. 177

5.24 Diagrama de flujo para el funcionamiento de la estrategia del umbral de capturas. 180

5.25 Nivel de consecución del umbral para varios valores de P_0 en un servidor con un total de 40 posiciones: a) $P_0 = 20$, b) $P_0 = 30$ y c) $P_0 = 35$. 182

5.26 Nivel de consecución del umbral para $P_0 = 38$ en un servidor con un total de 40 posiciones. 183

5.27 Rendimiento del ataque contra un servidor de 40 posiciones con la estrategia de umbral de capturas para diferentes valores del umbral, P_0, y de tiempo entre llegadas de usuario, T_a: a) sobrecarga, S, y b) disponibilidad, D. 185

A.1 Scenario of study. 197

A.2 Model for the server. 199

A.3 Time diagram of the state of the requests in the service queue (bottom) and the associated timing of the generated outputs (top) for the considered example. In black, service time. 201

A.4 Simulation of observed inter-output times in a flooded service queue scenario: (a) Inter-output time values and (b) histogram of the samples. 203

A.5 Attack specification: waveform and parameters. 206

A.6 Attack against an iterative server dynamics. Example of a period of attack. 208

A.7 Figures for indicators of an attack against an iterative server in a simulated environment. 212

A.8 User perceived performance *vs.* effort for 25 different configurations (t_{ontime} and Δ values) of the low-rate DoS attack against an iterative server. 213

A.9 Comparison between intelligent and non-intelligent strategies for three different attacks against an iterative server. 213

A.10 Comparison between intelligent and non-intelligent strategies in an iterative server flooded by legitimate users. 214

A.11 Possible configurations for the attack: *UPP* vs *E*. 216

A.12 Efficiency and effort obtained with the intelligent attack against a concurrent server compared with a random flood of packets, depicted for 4 different scenarios. 217

Índice de tablas

3.1 Indicadores definidos para evaluar el rendimiento de un ataque DoS a baja tasa contra servidores. 73

3.2 Valores de configuración usados en el escenario de ejemplo mostrado en la Fig. 3.9 para la evaluación de $T_D(s)$. 98

4.1 Valores por defecto para los parámetros estudiados en los experimentos cuyos resultados se muestran en las Fig. 4.3 y 4.4. 122

4.2 Comparación del rendimiento de diferentes configuraciones de ataque en entornos reales y simulados. 134

5.1 Valores por defecto para los experimentos de evaluación del comportamiento del ataque. 155

5.2 Comparación entre los valores D y S obtenidos de ocho configuraciones de ataque diferentes evaluadas en entornos de simulación y real. 166

A.1 Defined indicators for efficiency and rate evaluation of a low-rate DoS attack. 211

A.2 Performance of real and simulated attacks. 215

A.3 Comparison between real and simulated environment results in the attack against a concurrent server. 218

Prólogo

Con el desarrollo de las tecnologías de la información y las comunicaciones, en las últimas décadas han ido surgiendo ciertos problemas que, actualmente, van adquiriendo cada vez mayor relevancia. Tal vez los más reseñables sean los relativos a la seguridad en las redes de comunicación. Ello ha provocado que numerosos investigadores y desarrolladores se hayan centrado en el estudio de mecanismos que garanticen unos mínimos requisitos de seguridad en las comunicaciones, de modo que, hasta el máximo nivel posible, se incremente la confianza del usuario en los servicios ofertados.

En este contexto se sitúa como un problema de creciente impacto el provocado por los ataques de denegación de servicio. Este tipo de ataques, cuando son ejecutados contra un cierto recurso, servicio o, en general, objetivo, son potencialmente capaces de provocar la indisponibilidad del mismo. Esta preocupante potencialidad ha provocado una implicación cada vez mayor por parte de un gran número de empresas y organizaciones que, habida cuenta de las cuantiosas pérdidas que este tipo de ataques puede generar, tratan de investigar e implementar medidas de defensa contra ellos.

Aunque un abundante trabajo ha sido desarrollado en este campo, siguen apareciendo nuevas técnicas de ataque para las que es necesaria la adopción de mecanismos de defensa oportunos. Como un buen jugador de ajedrez, todo aquel que busque establecer una buena defensa contra estos ataques debe estudiar previamente las capacidades, técnicas, posibles implementaciones y, cómo no, limitaciones del adversario. Este análisis es vital para comprender los fundamentos o bases que sustentan la ejecución de los ataques y, por consiguiente, partiendo de él, será más sencillo determinar la mejor estrategia para contrarrestarlos.

El trabajo desarrollado en esta tesis doctoral ilustra una novedosa metodología para la ejecución de ataques de denegación de servicio contra servidores de aplicación. Ésta consiste en la saturación de los recursos del objetivo del ataque, pero utilizando cierto conocimiento acerca del comportamiento del servidor, de modo que el tráfico dirigido hacia el mismo presente una baja tasa. Esta característica del ataque hace que, desde el punto de vista del atacante, se obtengan dos beneficios principales. En primer lugar, la cantidad de recursos necesarios para ejecutar el ataque se reduce considerablemente. En segundo lugar, el atacante podría evitar ser detectado por parte de ciertos mecanismos de defensa que se basan en la detección de altas tasas de tráfico.

El trabajo se centra en el estudio de dichos ataques desde el punto de vista del atacante, es decir, se exploran las posibilidades, capacidades, implementaciones alternativas

y también sus limitaciones. Aunque no se contempla el estudio de mecanismos de defensa, es evidente que la motivación fundamental es la de proporcionar una base para el desarrollo de posibles metodologías que, en esta dirección, consigan mitigar, cuando no anular, los efectos de este tipo de ataques.

Contenidos y estructura de la memoria

Esta memoria se compone de seis capítulos, cuyos contenidos y estructura se describen a continuación.

En el Capítulo 1 se realiza una descripción del problema de la seguridad en el contexto de las redes de comunicación, así como de los ataques de denegación de servicio. Se plantean los conceptos básicos para su comprensión y se explican los mecanismos fundamentales de defensa actualmente propuestos contra este tipo de ataques.

En el Capítulo 2 se realiza el modelado del escenario en el que se desarrollará el estudio de los ataques a baja tasa contra servidores. En él se describen las partes integrantes del mismo y se plantean las condiciones de estudio en las que se enmarcará el resto de contenidos. En este capítulo se puede encontrar un estudio sobre ciertas características de los servidores que será de utilidad para la comprensión de la estrategia seguida para ejecutar el ataque.

En el Capítulo 3 se presentan los mecanismos generales para ejecutar el ataque de baja tasa contra servidores. Su contribución principal es la descripción de la estrategia general seguida para la ejecución del mismo. Además, también se desarrollan unos modelos matemáticos de análisis del rendimiento del ataque que permiten evaluar su funcionamiento y capacidades.

En el Capítulo 4 se abordan las características del ataque de baja tasa cuando es ejecutado contra un tipo de servidor concreto: los servidores iterativos. En él se detallan los diferentes aspectos que, de forma particular, aparecen en la implementación de este tipo de ataques, y se realiza un análisis experimental del rendimiento conseguido.

En el Capítulo 5 se realiza el mismo desarrollo que en el anterior pero concretando los detalles de diseño que deben tenerse en cuenta cuando el ataque es ejecutado contra otro tipo de servidores: los concurrentes. También se efectúa el análisis experimental de su rendimiento y se discuten las diferentes posibilidades de implementación del mismo.

Finalmente, en el Capítulo 6 de la memoria se discuten, a modo de resumen, los diferentes resultados presentados a lo largo de la misma, recopilándose las principales conclusiones obtenidas, así como las más inmediatas líneas de trabajo futuro y algunos comentarios adicionales.

Capítulo 1

Introducción a la seguridad y los ataques de denegación de servicio

1.1 Introducción

En un mundo cada vez más globalizado e interconectado, en el que las relaciones, negocios, interacciones e intercambios de información precisan de la eliminación de las barreras impuestas por la distancia física, es fundamental la presencia de sistemas de comunicación eficientes en el transporte y difusión de la información. Se puede considerar notable, por tanto, el papel esencial que, en el desarrollo de la actualmente denominada sociedad de la información, juega el diseño e implementación de una estructura tecnológica que posibilite y facilite dicho proceso comunicativo.

Acorde con la importancia del desarrollo de estas bases tecnológicas, se ha realizado en las últimas décadas un importante esfuerzo económico, intelectual y social para hacerlo realidad. Fruto de estos esfuerzos fue la aparición de una red de datos mundial basada en la idea de la conmutación de paquetes [Baran, 1964] [Kleinrock, 1961], denominada ARPANET (*Advanced Research Project Agency Network*). Esta red nace en el departamento de defensa norteamericano (DARPA) como solución al problema de conseguir un modo que permitiera la comunicación entre equipos a través de una red reconfigurable. Si bien inicialmente el tamaño de esta red era más o menos reducido, posteriormente se fueron añadiendo a esta infraestructura de interconexión las redes de numerosas universidades, haciendo que adquiriese un tamaño considerable. Un proceso paulatino de incorporación de nuevas redes ha llevado a lo que actualmente conocemos como la red de redes, es decir, *Internet*. Esta gran y conocida red de comunicación de datos, junto con la red telefónica básica, constituyen actualmente los principales núcleos de comunicación a través de todo el mundo.

Este proceso histórico de desarrollo ha motivado que el diseño inicial de *Internet* haya estado orientado, casi exclusivamente, a resolver el problema de la comunicación entre nodos situados a cierta distancia. Ahora bien, conforme la tecnología se ha ido implantando y desarrollando y, por tanto, el número de usuarios ha ido creciendo, han surgido nuevos problemas de diverso calado que deben afrontarse. Uno de los más relevantes es la seguridad.

El objetivo de este capítulo es presentar el contexto y los conceptos básicos relacionados con la seguridad en las tecnologías de la información. Además, se profundizará en algunos aspectos adicionales que se consideran necesarios para el entendimiento del trabajo desarrollado en esta tesis doctoral.

El capítulo está estructurado de la siguiente forma. En primer lugar, el Apartado 1.2 expone en qué consiste el concepto de seguridad cuando es aplicado a los sistemas de comunicación. El Apartado 1.3 presenta una visión de los ataques a la seguridad. Posteriormente, el Apartado 1.4 ilustra uno de los tipos de ataque más destacados a la seguridad: los ataques de denegación de servicio (DoS[1]), en los que está centrado este trabajo. Finalmente, el Apartado 1.5 realiza una descripción de los métodos de defensa existentes para los ataques de denegación de servicio.

1.2 La seguridad en las redes de comunicación

La seguridad en el contexto de las tecnologías de la información y las comunicaciones, a la que nos referiremos en adelante simplemente como *seguridad*, es un campo de investigación y estudio que comprende numerosas disciplinas y aspectos. En los "Criterios Comunes para la Evaluación de la Seguridad en las Tecnologías de la Información" [CC, 2004], se precisan las relaciones que el concepto de seguridad incluye de la siguiente forma:

> "La seguridad está relacionada con la protección de los activos frente a amenazas, entendiendo por amenaza la capacidad potencial para hacer un mal uso de los activos protegidos. Todas las categorías de amenazas deben ser consideradas, aunque en el dominio de la seguridad se le presta más atención a aquellas relacionadas con actividades humanas de carácter malicioso o de cualquier otro tipo."

Aunque esta definición delimita un primer marco sobre el concepto de seguridad, la terminología en este ámbito no está completamente establecida, existiendo diferencias, aunque no significativas, entre los diferentes autores. Es preciso, por tanto, profundizar en el concepto de seguridad y clarificar los modelos y servicios básicos para su implementación.

[1] Se utilizará en adelante la notación DoS surgida de la denominación inglesa *Denial of Service* para referirse al concepto de denegación de servicio.

1.2.1 Políticas y modelos de seguridad

En general, es constatable que no se puede afirmar el hecho de que una determinada acción sea segura o insegura cuando no existe una base que permita juzgar la bondad o legalidad de dicha acción. Esta base está constituida por las denominadas *políticas de seguridad*. Una política de seguridad establece, en definitiva, los diferentes criterios que, directa o indirectamente, permiten discernir los eventos y acciones permitidos o prohibidos para un sistema, desde el punto de vista de su seguridad [Carracedo, 2004].

Las políticas de seguridad se pueden clasificar como *explícitas* cuando constituyen unas reglas bien documentadas, registradas y disponibles para su consulta por parte de un potencial ejecutor de la política. Además, también existirán las denominadas políticas *implícitas*, es decir, las que establecen criterios que no están documentados pero que se asumen bien por su obviedad, bien por costumbre [Lindqvist, 1999].

En general, el establecimiento de una política concreta de seguridad viene determinado por las propiedades de seguridad deseables para los activos a proteger. Es importante indicar que la implementación de las políticas suele repercutir directamente en la funcionalidad y la facilidad de uso de los sistemas, de modo que lo usual es alcanzar un compromiso entre diferentes factores [Fraser, 1997], tales como los servicios ofrecidos frente a la seguridad proporcionada, la facilidad de uso del sistema frente al nivel de seguridad requerido y el coste de la seguridad frente a los beneficios que aporta.

De esta forma, la política de seguridad concretará los servicios de seguridad que son necesarios, así como el nivel de profundidad preciso para la implementación de dichos servicios.

Para garantizar la implementación de los diferentes servicios de seguridad existen tres campos de trabajo que deben ser considerados:

- *Prevención*:

  Para proporcionar un primer nivel de seguridad, es necesario prevenir los ataques a la seguridad del sistema que debe ser protegido.

- *Detección*:

  Una vez que las medidas preventivas se han implementado, hay que considerar que un atacante puede evitar dichas medidas. Será necesario, por tanto, disponer de un método que permita detectar las violaciones de la política de seguridad que se produzcan en el sistema.

- *Respuesta*:

  Las organizaciones y compañías necesitan desarrollar un plan que establezca las respuestas concretas a ejecutar ante determinadas violaciones de la política de seguridad.

Por otro lado, para definir e implementar los servicios necesarios para garantizar que un sistema se considere seguro, se utilizan diferentes modelos [Canavan, 2001] que determinan la arquitectura de la red y la configuración de los sistemas. En general, la utilización de un modelo no excluye la incorporación de aspectos de otro, de modo que la mayoría de los

sistemas de seguridad incorporan características o variantes procedentes de alguno de estos modelos:

- *Seguridad por oscuridad*:

  Este modelo se basa en el concepto de que si una entidad no conoce la existencia de una red o sistema, no intentará atacarlo. Supone que la ocultación de un elemento en la red constituye un nivel de seguridad suficiente.

- *La defensa perimetral*:

  Este tipo de defensa es análoga a la ofrecida por una muralla que protege un castillo. Las empresas y corporaciones que utilizan este modelo de seguridad fortalecen sus sistemas perimetrales y encaminadores de salida. Un ejemplo de implementación de este modelo son los sistemas denominados *cortafuegos* [Goncalves, 1997], que separan los sistemas a proteger del exterior considerado no seguro.

- *La defensa en profundidad*:

  Esta es la aproximación más robusta. Persigue establecer un sistema seguro mediante la monitorización y fortalecimiento de todos los elementos que conforman la red y no solamente del perímetro como en la aproximación anterior. Aunque en dicho perímetro las medidas deben ser más robustas y con funcionalidades más avanzadas, la seguridad de los elementos interiores no dependerá exclusivamente de la aplicada en el perímetro, sino que tendrá que ser reforzada con nuevos elementos. Este paradigma es más difícil de implementar y requiere que todos los elementos interiores y sus administradores se involucren en la puesta en práctica de las políticas de seguridad.

1.2.2 Servicios y mecanismos básicos de la seguridad

Una condición previa para definir un sistema como seguro es la de establecer unos criterios que determinen cuándo lo es. Con este fin, hacia el año 1988, el subcomité SC 21 de ISO creó la norma ISO 7498 [ISO, 1988], en la que se estructura la arquitectura de seguridad requerida para un sistema en el ámbito de las comunicaciones.

En el contexto de las redes de comunicaciones se define el concepto de *servicio de seguridad* como aquél que protege las comunicaciones de los usuarios ante determinadas violaciones de la política de seguridad [Carracedo, 2004]. Un servicio de seguridad no difiere, desde un punto de vista conceptual, de cualquier otro servicio telemático, tal y como se definen en las arquitecturas de redes jerarquizadas en niveles o capas. Es importante señalar que los servicios de seguridad son un caso particular de servicios telemáticos y representan, la mayoría de las veces, un valor añadido. Por eso casi nunca aparecen de forma aislada, sino mejorando la funcionalidad de otros servicios.

En la norma ISO anteriormente citada [ISO, 1988] se definen cinco servicios básicos de seguridad: autenticación, confidencialidad, integridad, control de acceso y no repudio. También se han incluido con posterioridad, por parte de algunos autores [Carracedo, 2004], dos servicios básicos adicionales: disponibilidad y privacidad. A continuación se hará una breve descripción del contenido y fundamento de estos siete servicios de seguridad. En el

desarrollo de la misma también se mostrarán algunos mecanismos técnicos que posibilitan la implementación de los servicios.

Servicio de autenticación o autentificación

El servicio de autentificación o autenticación garantiza que una entidad comunicante (persona o máquina) sea realmente quien dice ser. La autenticación es crítica cuando es necesario que exista una relación de confianza entre distintas partes.

En el ámbito del acceso a un determinado sistema o del requerimiento de un servicio en la red, existen tres formas básicas de autenticación [Sandhu and Samarati, 1997] [Eric Guerrino and Kapito, 1997] basadas en: algo que la entidad conoce [Lamport, 1981] [Klein, 1990] [Feldmeirer and Karn, 1990], algo que la entidad tiene [Gobioff et al., 1996] [Guillou et al., 1992], o bien algo que la entidad es [McMordie, 1997] [Bouchier et al., 1996] [Daugman, 1998] [Huopio, 1998]. Muchas redes utilizan no solamente una de ellas sino una combinación de varias.

En el campo de la autenticación también hay que destacar el papel de la criptografía, en particular la de clave pública [Benantar, 2002], que ha resultado de gran utilidad en la construcción de mecanismos de autenticación a través del uso de *firmas digitales* [Scheneier, 1995] [Menezes et al., 1996].

Servicio de confidencialidad de los datos

Este servicio proporciona protección para evitar que los datos sean revelados, accidental o deliberadamente, a un usuario o sistema no autorizado. Es decir, garantiza que los datos tan solo van a ser entendibles por el destinatario o destinatarios de la comunicación. Para ello, el mensaje se alterará de tal manera que se garantice que aquellas entidades que no sean destinatarias autorizadas de la información, aunque dispongan de ella, no sean capaces de entender su significado.

Normalmente, el servicio es implementado mediante el uso del cifrado de la información [Scheneier, 1995], [Menezes et al., 1996], de modo que su lectura sea incoherente para cualquier entidad no autorizada que la capture.

Servicio de integridad de los datos

Este servicio garantiza al receptor de un mensaje que los datos recibidos coinciden exactamente con los enviados por el emisor de los mismos, de tal forma que puede tener garantías de que a la información original no se le ha añadido, modificado ni sustraído ninguna de sus partes. Es decir, el receptor del mensaje detectará si se ha producido o no una modificación del mensaje, lo que le permitirá aceptar o rechazar los datos recibidos.

Este servicio se implementa normalmente mediante el uso de códigos de resumen (*hash*) de un mensaje [Rivest, 1990] [Rivest, 1992]. El envío de estos códigos por parte del emisor permite a un destino comprobar si el mensaje recibido se puede resumir mediante

el mismo código que el que está cifrado. Es evidente que junto con la modificación de los datos se podría modificar también el código de resumen. Para evitar este problema se hace uso de la criptografía de clave pública y privada [Carracedo, 2004].

Servicio de no repudio

Este servicio tiene como finalidad evitar que alguno de los participantes en una comunicación niegue (repudie) haber intervenido en ella o en la autoría de un mensaje concreto [Carracedo, 2004]. Es un servicio esencial en aplicaciones como el comercio electrónico, donde una transacción económica debe disponer siempre de una prueba de su solicitud y/o realización.

Para que este servicio se pueda proporcionar será necesaria una infraestructura de seguridad que garantice y proporcione las evidencias exigidas. Esta es la razón por la que la implementación de este servicio resulta más compleja que la de los anteriormente comentados. No obstante, a pesar de la complejidad, este servicio se puede considerar como fundamental en el desarrollo de la sociedad de la información.

Algunas de las aplicaciones más conocidas en las que resulta esencial la implementación de estos servicios son aquellas relacionadas con la gestión de operaciones con las administraciones públicas (e-Administración).

Servicio de control de acceso

Este servicio tiene como objetivo evitar el uso no autorizado de los recursos de un sistema. La idea subyacente es la de construir una entidad, generalmente conocida como *monitor de referencia*, cuya función es la de gestionar las operaciones efectuadas por las entidades sobre los recursos [Sandhu, 1993] [Sandhu and Samarati, 1994] [Sandhu et al., 1996] [Sandhu and Samarati, 1997].

En la mayoría de los casos, este servicio se encuentra ligado al servicio de autentificación. Es en el momento en que un usuario ha demostrado quién es cuando se le aplican las restricciones y permisos necesarios.

Servicio de privacidad

El objetivo de este servicio es conseguir que la identidad del elemento que realiza una determinada operación permanezca oculta ante algunos de los sistemas, actores o servicios presentes en dicha operación [Carracedo, 2004]. Se trata de mantener el anonimato en determinadas ocasiones bien especificadas o definidas.

Con frecuencia tiende a confundirse este servicio con el de confidencialidad. Si bien es cierto que la existencia de un servicio de confidencialidad fiable y robusto es fundamental e imprescindible para la obtención de la privacidad, no deben mezclarse ambos conceptos ya que, para poder hablar de privacidad es necesario el establecimiento de una política de seguridad que decida los elementos de información en la comunicación que pueden ser

conocidos, y quiénes pueden ser observadores de dicha información. El uso coordinado de los servicios y las políticas de seguridad que se establezcan será lo que finalmente proporcione un nivel de privacidad determinado.

Servicio de disponibilidad

Este servicio se refiere a la capacidad de la red, sistema o servicio para estar disponible en cualquier momento y para recuperarse con prontitud a partir de la ocurrencia de un evento de interrupción del mismo [Carracedo, 2004].

Los mecanismos que pretenden afectar al funcionamiento de un entorno son los denominados *ataques de denegación de servicio*. Este tipo de ataques se explicarán con mayor detalle en el Apartado 1.4, dado que son la familia de ataques dentro de los cuales se enmarca el estudio que se desarrolla en este trabajo.

1.3 Ataques a la seguridad

Pese a la no existencia de una terminología ampliamente establecida y aceptada, algunos autores definen el concepto de *amenaza* a la seguridad como cualquier violación potencial de la política de seguridad establecida [Carracedo, 2004]. Esta definición concreta, para el contexto considerado, la aportada por [CC, 2004] y citada en el Apartado 1.2. Las amenazas pueden afectar al funcionamiento, operación, integridad o disponibilidad de una red o sistema.

Partiendo del concepto de amenaza, se define un *ataque* como la instanciación de una amenaza, es decir, como el hecho de que una amenaza se haya materializado. Mientras que la amenaza es una violación potencial de la política de seguridad, el ataque será una violación cierta de dicha política.

En la literatura de seguridad se emplea también con frecuencia el concepto de *riesgo*. Este término es similar al de amenaza. El riesgo al que se encuentra sometido un sistema se puede definir como la probabilidad de que no pueda forzarse el cumplimiento de la política de seguridad establecida, ante la ocurrencia de un evento que trata de violarla [Landwehr, 2001].

Se puede definir también el concepto de *vulnerabilidad*, entendiéndose por tal una debilidad inherente a un diseño, configuración o implementación de un sistema, que hace que sea susceptible a sufrir una amenaza [Carracedo, 2004]. Cuando los sistemas considerados son telemáticos, la mayoría de las vulnerabilidades son debidas a alguna de estas causas:

- *Diseños defectuosos*:

  Los sistemas hardware o software contienen frecuentemente fallos de diseño que pueden ser utilizados para realizar un ataque. Se dice entonces que los sistemas presentan *agujeros de seguridad* (*bugs*). Un ejemplo de este tipo de vulnerabilidad puede ser el agujero detectado en el cliente de correo *sendmail* en el año 1999 [Gregory, 1999].

- *Implementaciones defectuosas*:

  Esta situación se refiere a los sistemas que no están correctamente configurados y que, por esta razón, son vulnerables y susceptibles de sufrir ataques. Este tipo de vulnerabilidades está frecuentemente provocado por la inexperiencia de los administradores y del personal que configura una máquina o servicio. Un ejemplo de este tipo de vulnerabilidad puede ser cualquier sistema que no tiene acceso restringido a determinados ficheros ejecutables, permitiendo así su utilización a usuarios no legítimos.

- *Gestión deficiente*:

  Estas vulnerabilidades son causadas por la existencia de procedimientos de gestión inadecuados o determinados procesos de mantenimiento preventivo insuficientes para los sistemas. Las medidas de seguridad deben ser documentadas y supervisadas. Incluso algo tan simple como una copia de seguridad diaria debe ser verificada. Debe existir también una delimitación estricta de la responsabilidad para las tareas de gestión, de modo que se asegure que los procedimientos se están siguiendo adecuadamente. También es importante establecer un modelo de gestión de modo que no solamente una única persona sea la que esté a cargo de toda la gestión del sistema.

De forma general, las vulnerabilidades que un sistema presenta proceden normalmente de alguna de estas fuentes, aunque en numerosas ocasiones es la combinación de varias de ellas la que provoca la aparición de la vulnerabilidad.

1.3.1 Clasificación y tipos de ataque

Los ataques a la seguridad de un sistema se pueden entender bajo la perspectiva que considera como función primordial de dicho sistema el intercambio de información [Stallings, 2003a] [Stallings, 2003b]. Esta premisa supone la existencia de un flujo o comunicación desde un origen o emisor de la información a un destino o receptor, utilizando un canal intermedio o una red de comunicación, tal y como se muestra en la Fig. 1.1(a). En este contexto, es posible concebir cuatro categorías generales de ataques a la seguridad:

- *Interrupción*:

  En este caso, el objetivo del ataque es la fuente de información o el canal de comunicación —Fig. 1.1(b)—. Como consecuencia, un activo del sistema queda inutilizable. Este es un ataque, por tanto, efectuado contra el servicio de disponibilidad. El ataque impide o inhibe el uso normal o la gestión de recursos informáticos o de comunicaciones.

- *Intercepción*:

  Un elemento o parte no autorizada consigue el acceso a la información —Fig. 1.1(c)—. Se trata de un ataque realizado contra el servicio de confidencialidad.

- *Modificación*:

  En este caso, el atacante no sólo consigue el acceso a la información proporcionada por la fuente sino que, además, la modifica —Fig. 1.1(d)—. Es un ataque efectuado contra el servicio de integridad.

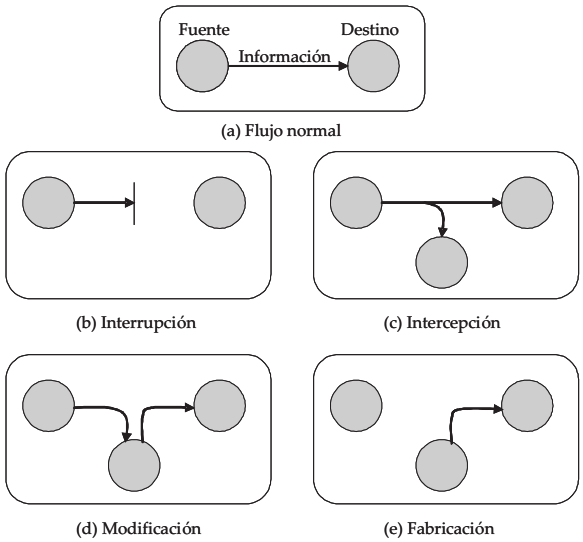


Fig. 1.1: Amenazas a la seguridad [Stallings, 2003b].

- *Fabricación*:

  Una parte no autorizada inserta cierta información en la comunicación, provocando así un ataque contra el servicio de autentificación –Fig. 1.1(e)–. Cuando el objeto o información introducida consiste en información previamente capturada, se suele hablar de ataques de repetición (*replay*). En estos, uno o varios mensajes legítimos son capturados y repetidos para producir un efecto no deseado, como por ejemplo ingresar dinero repetidas veces en una cuenta bancaria. Además, se usa el término enmascaramiento (*masquerading*) cuando el atacante trata de hacerse pasar por una fuente de información legítima.

Otro modo de clasificación de los ataques es el propuesto por S. Kent [Kent, 1997] (véase Fig. 1.2). Esta clasificación atiende al modo en que se llevan a cabo los ataques, de forma que se pueden encontrar *ataques pasivos*, es decir, aquellos en los que el atacante no altera la comunicación, sino que únicamente la escucha o monitoriza para obtener información que está siendo transmitida, y *ataques activos*, que implican algún tipo de modificación del flujo de datos transmitido o la creación de un falso flujo de datos, es decir, engloba a las categorías de la clasificación anterior denominadas como interrupción, modificación y fabricación.

Finalmente, también se podría establecer una clasificación atendiendo al lugar desde el que se efectúa el ataque. De este modo, se pueden encontrar los *ataques externos*, que se producen desde el exterior del entorno atacado, y los *ataques internos*, que se producen por parte de un atacante que se encuentra dentro de dicho entorno. Naturalmente, este último tipo de ataques es muy peligroso y difícil de corregir. La estructuración de la seguridad con modelos de defensa en profundidad hace más fácil el tratamiento de este tipo de ataques.

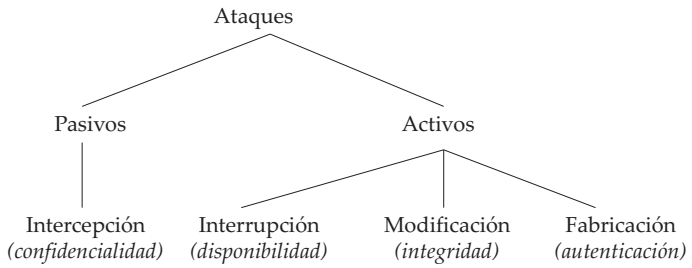


Fig. 1.2: Ataques de seguridad activos y pasivos [Kent, 1997].

1.4 Los ataques de denegación de servicio

En este apartado se realiza un estudio general acerca de una tipología específica de ataques, sobre la cual está centrado el desarrollo de este trabajo: los ataques de denegación de servicio. Para ello, se presentarán los conceptos generales y el funcionamiento detallado de la ejecución de dichos ataques. Se hará especial énfasis en los ataques de denegación de servicio a baja tasa, debido a su interés en el estudio abordado en este trabajo.

1.4.1 Conceptos generales

El ataque de denegación de servicio se puede clasificar (véase el Apartado 1.3.1), dentro de los ataques activos de interrupción. Tienen como objetivo reducir la disponibilidad de un determinado activo en el sistema mediante la realización de un ataque bien a la fuente de información, bien al canal de transmisión, o incluso a ambos [Mirkovic et al., 2004].

Los ataques de denegación de servicio son diferentes en su objetivo, forma y efecto a la mayoría de ataques que se efectúan contra redes de comunicaciones y sistemas informáticos. La mayoría de los ataques tratan de penetrar en un sistema, extraer secretos, robar números de cuentas de crédito, conseguir el control de una máquina para instalar sus programas o almacenar sus datos, modificar páginas web o alterar algún contenido importante en la máquina víctima. Frecuentemente, el atacante considera a las máquinas comprometidas o invadidas como recursos a utilizar en función de las necesidades que tenga en un determinado momento. Por el contrario, el objetivo de un ataque de denegación de servicio en una red de comunicación es evitar la ejecución de una actividad legítima, tal como la navegación por páginas web, escuchar la radio en *Internet*, transferir dinero desde la cuenta bancaria o incluso tareas críticas como la comunicación entre un avión y su torre de control. Este efecto de denegación de servicio se realiza enviando determinados mensajes hacia uno de los destinatarios o el propio canal de la comunicación de forma que se interfiera en su funcionamiento, impidiendo acceder total o parcialmente al servicio ofertado.

1.4.2 Funcionamiento de los ataques DoS

Existen dos métodos básicos para la realización de un ataque de denegación de servicio: la explotación de una vulnerabilidad descubierta en una máquina objetivo, o el envío de un amplio número de paquetes de apariencia legítima. El primer tipo se denomina usualmente *ataque de vulnerabilidad*, mientras que el segundo es conocido como *ataque de inundación* [Mirkovic et al., 2004].

Los ataques de vulnerabilidad funcionan enviando, a una aplicación que posee una determinada vulnerabilidad, uno o varios paquetes construidos de forma especial. La vulnerabilidad consiste, generalmente, en un fallo en la implementación del software de la aplicación o en una deficiencia en la configuración del recurso/utilidad. Los paquetes maliciosos procedentes del atacante pueden provocar en una determinada aplicación un estado que el programador no previó en el momento de su diseño. De esta forma, la llegada de dichos paquetes puede generar un bucle infinito, ralentizar gravemente la velocidad de ejecución de la aplicación, hacer que ésta deje de funcionar, provocar el reinicio de la máquina o consumir grandes cantidades de memoria, generando, en todos los casos, la denegación del servicio ofertado a los usuarios legítimos.

Por otro lado, los ataques de inundación funcionan enviando un número amplio de mensajes hacia un destino que se convierte en víctima del ataque, de forma que su procesamiento supone el agotamiento de determinados recursos críticos en dicha víctima. Por ejemplo, el procesamiento de peticiones complejas puede requerir un amplio tiempo de CPU, la transmisión de mensajes largos puede agotar el ancho de banda disponible para las comunicaciones y la recepción de mensajes que inician comunicaciones con nuevos clientes puede agotar la memoria disponible. Una vez que un recurso está agotado, los clientes legítimos no podrán hacer uso del servicio.

La principal característica de los ataques de inundación consiste en que su fortaleza reside más en el volumen de tráfico que en su contenido. Esto tiene dos implicaciones principales:

- Los atacantes pueden enviar una gran variedad de paquetes. El tráfico de ataque se puede hacer incluso similar al legítimo y adoptar, dentro de ciertos márgenes, una estructura y estadística arbitraria, lo cual facilita en gran medida la ocultación del ataque.
- El flujo de tráfico del ataque debe ser tan elevado como para consumir los recursos del destinatario.

Cuando un atacante pretende atacar destinos con recursos muy abundantes, es difícil conseguir una máquina que esté adecuadamente dimensionada para ejecutar el ataque. En este caso, el atacante puede adoptar una estrategia diferente, que consiste en tomar bajo control un número abundante de máquinas y conseguir que todas ellas envíen los mensajes de forma sincronizada a la máquina objetivo. A cada una de estas máquinas se las denomina *agentes*. De esta forma, se puede lograr que el tráfico agregado que llega a la máquina víctima del ataque supere al que ésta es capaz de procesar y, por tanto, sus recursos queden

saturados. Los ataques que siguen este tipo de estrategias se denominan ataques distribuidos de denegación de servicio (DDoS, del inglés *Distributed Denial of Service*).

En este punto de la discusión se puede, por tanto, precisar una diferenciación clara entre los ataques de denegación de servicio que se efectúan desde una localización única, es decir, una sola máquina atacante que envía tráfico malicioso a un destino, que denominaremos simplemente como DoS, y aquellos ataques en los que intervienen numerosas máquinas para atacar a una sola víctima de forma coordinada, para los que se utilizará el acrónimo DDoS.

Ambas estrategias de realización del ataque de denegación de servicio, DoS y DDoS, son una amenaza importante para el funcionamiento de los sistemas en *Internet*. Sin embargo, el peligro de los ataques DDoS es mayor, fundamentalmente porque utilizan un número elevado de máquinas para realizar el ataque. Esto supone que cualquier sistema objetivo del ataque, sin importar lo bien dimensionado que esté para prevenirlo, puede ser derrotado. Reclutar y tomar el control de una gran cantidad de máquinas en *Internet* se ha convertido en una tarea relativamente trivial, debido al elevado número de usuarios noveles que están conectados a la red y que no tienen en consideración los más mínimos requisitos de seguridad a cumplir. Por otra parte, se han desarrollado, y son fácilmente localizables, muchas herramientas automáticas que ejecutan ataques de tipo DDoS sin necesidad de conocimiento alguno sobre el funcionamiento de las mismas. Esto hace que cualquier usuario sin mucha experiencia pueda realizar este tipo de ataques. Además, incluso si la máquina objetivo identificara a todas las máquinas que la están atacando, sería difícil decidir qué acción tomar en contra de un número tan elevado de máquinas[2].

Otra característica adicional que incrementa la complejidad del tratamiento de los ataques DDoS es el hecho de que pueden utilizar tráfico similar o idéntico al legítimo. En este caso, los recursos de la víctima se consumen procesando un gran número de mensajes iguales que los legítimos, lo que hace que sea prácticamente inviable tomar acciones contra este tipo de ataques sin afectar al tráfico legítimo que llega a la máquina objetivo.

Ejecución de los ataques DoS

A continuación se realizará una clasificación de los ataques de denegación de servicio atendiendo al tipo de objetivo elegido. Para cada uno de ellos se detallarán los mecanismos que pueden ser efectuados por el atacante para llevar a cabo el ataque.

Así, cuando se considera el objetivo al que van a atacar, los ataques se pueden categorizar dentro de los siguientes tipos [Mirkovic et al., 2004]:

- *Ataques de vulnerabilidad*:

  Como ya se ha mencionado, el ataque a una vulnerabilidad consiste en el envío de un paquete, construído de forma especial, hacia un destino, de modo que aprovecha una determinada vulnerabilidad en el recurso objetivo del ataque y consigue afectar al servicio que se presta. Por ejemplo, algunas versiones de sistemas operativos se vuelven

[2] Actualmente, se puede estar hablando, para un ataque distribuido, de un número en torno a 100.000 máquinas implicadas.

inestables cuando se les envían paquetes fragmentados con marcas de fragmentación incoherentes. Esta vulnerabilidad se puede explotar enviando dos paquetes UDP con marcas de fragmentación mal construidas a la víctima. Para este *exploit*[3] existen numerosas variantes, conocidas como *bonk, boink, teardrop* y *newtear* [Center, 1998a].

- *Ataques de inundación pura*:

  La forma más obvia de ejecutar un ataque DoS de inundación pura es enviar una gran cantidad de mensajes, divididos en paquetes, a un servicio que resida en una máquina objetivo. A menos que algún elemento de red entre la víctima y las máquinas atacantes realice el filtrado de dichos paquetes, la víctima ocupará sus recursos en recibir y procesar los mensajes. Si se envían suficientes paquetes de ataque, todos los recursos de la máquina destino estarán ocupados en procesar peticiones que no poseen ningún valor.

  Otra opción para ejecutar este tipo de ataques consiste en atacar la interfaz de red de la víctima. El atacante solamente necesita generar flujos con velocidad mayor que el máximo admitido por la tarjeta de red de la víctima y enviarlos hacia ella. Asumiendo que no hay ninguna entidad intermedia que realice el filtrado de dicho tráfico, estos paquetes consumirán los recursos de red de la víctima. En tal caso, si coexistiese tráfico legítimo, habrá una alta probabilidad de que éste no sea atendido; ello se traducirá, por tanto, en el éxito del ataque DoS.

  El atacante también puede elegir como objetivo saturar la red local que une a la víctima con *Internet*. Para ello, puede enviar suficientes paquetes a la víctima o a otros nodos en dicho segmento de red para sobrecargarlos. En esta forma de ataque DoS, todos los nodos en el segmento de red sufren el ataque de forma similar. Este ejemplo ilustra una importante característica de los ataques DoS: el ataque no sólo afecta a la víctima, sino también a sus usuarios legítimos y a cualquiera que comparta el recurso crítico que está siendo atacado. Por ejemplo, el atacante puede elegir como objetivo la red de un ISP (Proveedor de servicios de *Internet*. Del inglés: *Internet Service Provider*). En este caso, todos los usuarios conectados a *Internet* a través de este ISP sufren los efectos del ataque.

  El ataque de denegación de servicio mediante inundación pura también se puede efectuar haciendo un uso fraudulento de servicios legítimos. Así, la técnica de *IP spoofing* [Scambray et al., 2000] permite conseguir que algunos servicios o sistemas que son perfectamente seguros y no comprometidos, participen en un ataque DoS. Para ello, el atacante elige un servicio o protocolo públicamente disponible, como por ejemplo el servicio de resolución de nombres de dominio (DNS), web o *ping*, y envía peticiones a dichos servidores, falsificando la dirección origen de las mismas y colocando en su lugar la dirección de la víctima. Los servidores responden a dichas peticiones enviando las respuestas hacia la víctima, provocando así la denegación de servicio. Un ejemplo de implementación de este tipo de ataques es el ataque contra servidores DNS [Center, 2000] o el ataque *Smurf* [Center, 1998b].

  Este último tipo de ataques DoS se denominan ataques de *reflexión* [Paxson, 2001], y a los servidores que participan en ellos se les llama *reflectores*. Un caso especial es aquél

[3] Se denomina *exploit* a la implementación de un ataque que aprovecha una determinada vulnerabilidad [Mirkovic et al., 2004].

que involucra a servicios que pueden generar respuestas muy largas o numerosas para pocas peticiones recibidas. En este caso, el ataque se denomina ataque de *amplificación* y permite crear una denegación de servicio con el envío de muy pocos paquetes. Un ejemplo de esta técnica consiste en enviar peticiones con direcciones orígenes falsas que se corresponden con la de la víctima, como en un ataque de reflexión normal, pero especificando un destino de difusión. Esto provoca que, ante un paquete de petición, todos los nodos presentes en el ámbito de difusión respondan enviando un paquete a la víctima, produciéndose de este modo la amplificación y, consecuentemente, la posterior denegación de servicio.

- *Ataques de protocolo*:

  Esta tipología está constituida por los ataques de inundación que aprovechan la debilidad o asimetría de un protocolo para conseguir su objetivo, evitando tener que enviar una cantidad muy elevada de mensajes para obtener ventaja, con respecto a la víctima, en el consumo de recursos respecto a la liberación de los mismos.

  Si la víctima tiene algún servicio activado que requiere más tiempo para procesar una petición que el que implica la generación de dicha petición, el atacante puede utilizar dicha asimetría. Esto también es aplicable a los casos en los que el servidor tiene que reservar un recurso limitado para atender a la petición, mientras que la generación de la misma no implica la reserva de ningún recurso. En este caso, incluso ráfagas cortas o poco frecuentes de tráfico de ataque pueden agotar el recurso crítico. Un ejemplo común de este tipo de ataques es el ataque a TCP de inundación SYN, descrito en detalle en [Center, 1996] [Schuba et al., 1997]. En éste, el atacante inunda a la víctima con paquetes TCP SYN, que se usan normalmente para establecer una conexión. La víctima reserva memoria en un registro limitado para gestionar dicha comunicación, mientras que el atacante puede enviar dichos paquetes sin ningún coste de memoria. Esta asimetría ayuda al atacante a deshabilitar el establecimiento de nuevas conexiones durante el ataque, simplemente mediante el envío de ráfagas de paquetes TCP SYN.

  Existe también una variante del ataque de inundación SYN consistente en establecer completamente la conexión pero sin desarrollar comunicación alguna a través de ella. El ataque mantiene la conexión abierta el máximo tiempo posible, respondiendo a los mensajes *keep-alive*, posibilitando de esta forma que se agoten los recursos de la víctima. Un ejemplo de este tipo de ataques es el *Naptha* [Corp., 2000].

- *Ataques a un recurso físico*:

  Se puede realizar un ataque contra un recurso físico específico de una máquina, como por ejemplo su CPU, memoria, capacidad de conmutación de un encaminador, etc. Los recursos que forman parte de la infraestructura de la red son particularmente atractivos para ser atacados, ya que estos ataques tienen impacto sobre gran parte de la población en *Internet*. Como ejemplo, el servicio de encaminamiento es un recurso de infraestructura crítico que puede ser atacado mediante DoS. Existen numerosas formas conocidas de ataque a este recurso [Greene, 2002] [Zhao et al., 2002] [Mittal and Vigna, 2002] [Jou, 2000].

- *Ataques de* middleware:

  En este caso, los ataques se efectúan contra un *middleware* como, por ejemplo, una función de generación de *hash* que ejecuta una operación en un tiempo determinado. Mediante el envío de una operación con valores de entrada selectivamente escogidos, se puede forzar al *middleware* a trabajar en su condición de peor caso, haciendo que el tiempo que tarde en ejecutar la función crezca exponencialmente [Menezes et al., 1996].

- *Ataques de aplicación*:

  El atacante puede tomar como objetivo una aplicación específica y enviarle un paquete que explota cierta vulnerabilidad de modo que dicha aplicación deje de funcionar total o parcialmente. También se le puede enviar tráfico hasta alcanzar el límite de peticiones de servicio que sea capaz de procesar. Por ejemplo, los servidores web tardan un cierto tiempo en procesar las peticiones de páginas web y enviarlas a su destinatario. Por tanto, existirá un número finito de peticiones por segundo que la aplicación servidora es capaz de procesar. El atacante enviará suficientes peticiones por segundo para saturar la capacidad de la aplicación.

La discusión previa ilustra la existencia de una delgada línea entre la clasificación de un ataque en una categoría u otra. En realidad, la mayoría de ataques se podrían clasificar en una o varias categorías, dado que en muchos casos no son excluyentes. De hecho, muchos de los ataques DoS son ataques combinados a varios objetivos, de modo que comparten las características de varias de las anteriores categorías.

Si bien se ha expuesto el modo básico de ejecución de un ataque de denegación de servicio, es preciso también ahondar en algunos detalles relacionados con la ejecución de ataques distribuidos, dada su mayor complejidad. Concretamente, es necesario estudiar cómo el atacante realiza el reclutamiento y toma el control de las máquinas atacantes y la forma en que trata de ocultar los vestigios del ataque para no ser descubierto. Los subapartados que se presentan a continuación tienen como objetivo presentar los detalles relacionados con los citados aspectos.

Ocultación del ataque

Uno de los problemas que debe resolver el atacante es el de ocultación de su identidad. El objetivo es evitar que se descubra quién ha llevado a cabo el ataque. Los mecanismos de análisis de tráfico y trazas en las máquinas pueden llevar a descubrir el origen de un tráfico de ataque y, consecuentemente, a la captura del atacante.

Para su ocultación, el atacante toma típicamente algunas medidas específicamente diseñadas para ello [Mirkovic et al., 2004]. En primer lugar, trata de eliminar u ocultar las trazas de la intrusión en las diferentes máquinas que invade. Además, intenta mantener su identidad oculta mediante el establecimiento de varias capas de indirección entre su máquina y los diferentes agentes del ataque. Esto consiste en que el atacante no será quien ordene directamente a los agentes la ejecución del ataque, sino que establecerá una capa intermedia de control, formada por diferentes máquinas denominadas *gestores*, cuya

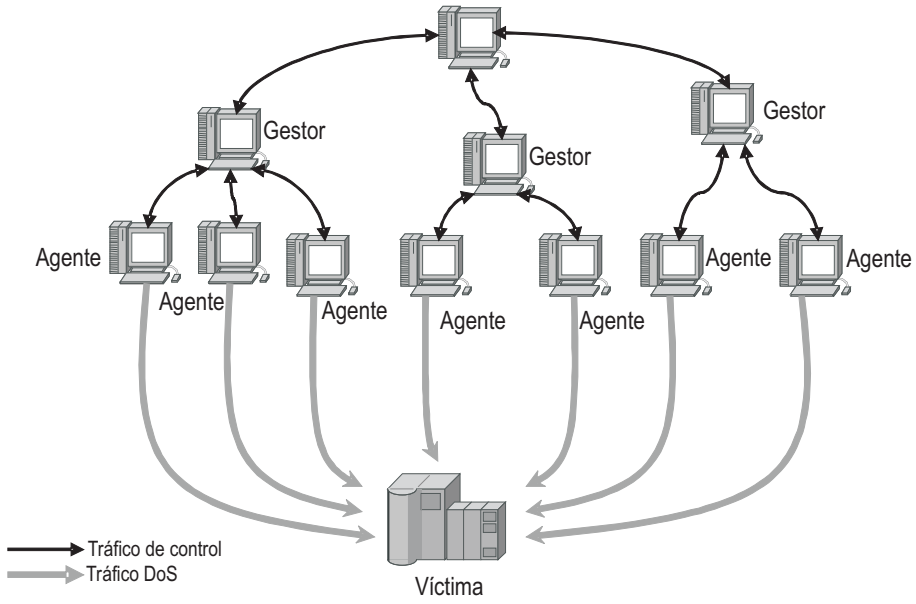


Fig. 1.3: Diagrama de la arquitectura gestor/agente para la ocultación del ataque DDoS [Mirkovic et al., 2004].

misión será la de controlar un grupo de agentes, enviar órdenes hacia ellos, recoger informes de comportamiento, etc. La Fig. 1.3 ilustra la arquitectura de múltiples capas que se implementa para la ocultación de la identidad.

Otro método utilizado por el atacante para ocultar su identidad consiste en realizar la comunicación con varias máquinas en secuencia antes de acceder a los gestores. Estos nodos intermedios entre la máquina del atacante y los gestores son llamados *saltos*[4]. La Fig. 1.4 muestra la comunicación entre el atacante y los gestores a través de un salto.

Tanto los gestores como los saltos son métodos efectivos para dificultar los intentos de investigación sobre la procedencia del ataque. Si las autoridades localizaran y examinaran una máquina con un agente instalado, sus comunicaciones apuntarían a uno de los gestores. Un examen más en detalle del gestor llevaría a un nodo de salto, de ahí a otro nodo de salto, y así sucesivamente hasta llegar al atacante. Si los nodos de salto se eligen en diferentes países y/o continentes, se convierte en una tarea muy difícil hacer el seguimiento del camino, debido a las trabas administrativas, legales y burocráticas implicadas por las diferentes administraciones.

Otro medio de ocultar la identidad de las máquinas atacantes es mediante la técnica, ya citada anteriormente, de *IP spoofing*[5] [CERT Coordination Center, 1996]. La cabecera del protocolo IP posee un campo que indica la dirección del emisor del paquete –campo dirección IP origen–. Esta información se rellena por parte de la máquina que envía el

[4] El término original inglés es *stepping stone*.
[5] Dada la amplia aceptación de este término, se optará por no realizar su traducción al español.

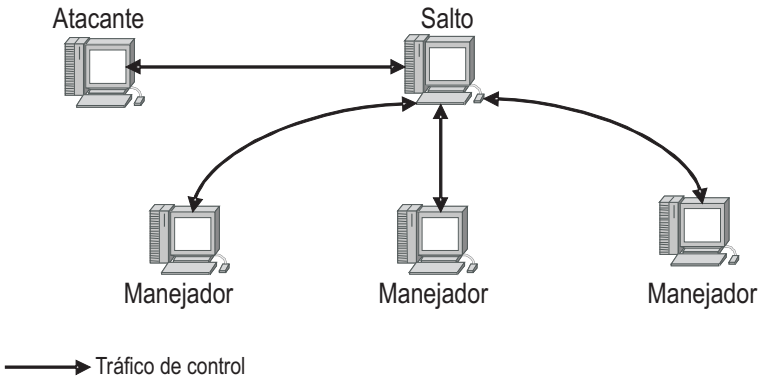


Fig. 1.4: Ilustración de una máquina actuando como salto para un ataque DDoS.

paquete y es usada por el destino, o por los encaminadores en el camino, hacia el destino para enviar respuestas hacia la fuente. Los atacantes falsifican usualmente este campo para ocultar su identidad en el ataque y que, de esa forma, sea más complicado detectar la ubicación de las máquinas que actúan como agentes. Este procedimiento de actuación también evita los mecanismos de defensa basados en la diferenciación entre los clientes legítimos y los atacantes mediante el análisis de las direcciones IP origen. Cuando se utiliza *IP spoofing* un atacante puede asumir fácilmente la identidad de un cliente legítimo, o incluso de varios de ellos.

Reclutamiento y control de máquinas atacantes

Un ataque distribuido de denegación de servicio requiere, tal y como ya se ha explicado anteriormente, el reclutamiento de múltiples máquinas que actuarán enviando tráfico de forma sincronizada a la víctima. Estas máquinas no son propiedad del atacante. Normalmente son sistemas poco seguros ubicados en universidades, empresas, instituciones del gobierno o simplemente sistemas domésticos. El atacante las invade, las controla y las utiliza para el ataque. Por ello, las máquinas atacantes son frecuentemente denominadas zombis, demonios, esclavos o agentes.

El primer paso para el reclutamiento de los agentes consiste en realizar una búsqueda de las posibles máquinas a invadir. Este proceso se denomina *escaneo*[6]. El atacante aprovecha determinados fallos de seguridad previamente conocidos para penetrar en las máquinas. Una vez explotada la vulnerabilidad, el atacante puede obtener acceso ilimitado al sistema, como si fuera un administrador. Las cuentas de usuario con contraseñas débiles, es decir, fácilmente adivinables, como combinaciones del nombre del usuario o palabras de diccionario, son otra puerta de entrada para un atacante a un sistema dado.

Una vez el atacante ha obtenido el control de la máquina, prosigue el proceso de ataque instalando en ella el agente de ataque DDoS, es decir, un software que se encargará

[6] Muchas de las técnicas de escaneo se pueden consultar en los textos de Ofir Arkin [Arkin, 1999] [Arkin, 2000].

de la ejecución posterior del ataque hacia la víctima final. Posteriormente se asegura de que todas las trazas de la intrusión están bien ocultas y que el código instalado funciona incluso cuando la máquina se reinicia.

Los ataques DDoS implican frecuentemente a cientos o miles de agentes. Para un atacante sería tedioso e implicaría mucho tiempo la tarea de ir reclutando agentes de modo manual. En lugar de seguir esta estrategia, existen herramientas que, de forma automática, descubren máquinas con vulnerabilidades mediante escaneo, explotan sus vulnerabilidades e instalan el código del agente, todo ello con un solo comando por parte del atacante. Además, le permiten recibir el resultado de la operación que se ha llevado a cabo. A las herramientas que realizan automáticamente los procesos de escaneo y propagación se las denomina *gusanos*[7] (*worms*) y existe una gran variedad de ellos [CERT Coordination Center, 2001a] [CERT Coordination Center, 2001b] [CERT Coordination Center, 2001d] [Moore, 2007] [Weaver, 2007] [CERT Coordination Center, 2001e] [CERT Coordination Center, 2001c]. Las diferentes técnicas de propagación de los programas atacantes se encuentran descritas con detalle en [Houle et al., 2001].

Además de los mecanismos de escaneo y propagación del software atacante[8], existen diferentes métodos para controlar la red de agentes. Este proceso de control se lleva a cabo mediante la realización de dos tareas: (a) envío de comandos para iniciar o terminar el ataque y (b) recopilación de datos, estadísticas e informes de funcionamiento del ataque. Para ello es necesario establecer un medio de comunicación entre el atacante y los agentes. Esta comunicación se puede realizar mediante el envío de:

- *Comandos directos*:

  Los comandos se envían de forma directa desde la máquina que controla al agente hasta el propio agente. Pueden ser comandos en texto plano, secuencias binarias o bien mensajes cifrados.

- *Comandos indirectos*:

  La técnica que utiliza comandos directos, anteriormente presentada, presenta un problema: Una vez que se captura un agente es posible rastrear la dirección del atacante debido a que se produce una comunicación directa con éste. Además, en una comunicación directa existe una limitación, impuesta por los sistemas operativos, en el número de conexiones abiertas que podrá tener una aplicación, en este caso la que controla los agentes. Si se quieren controlar muchos agentes sería necesario tener muchas conexiones abiertas por parte del atacante.

  Para evitar estos problemas, tanto el atacante como los agentes se conectan usualmente a un servidor de conversaciones IRC (acrónimo procedente del inglés: *Internet Relay Chat*) [Lo and Loon, 2004] de *Internet*, crean un canal de comunicación, usualmente protegido por contraseña, y se comunican a través de él. Los agentes tendrán codificado en sus programas el número de canal al que conectarse.

  Las ventajas de este esquema son muchas. En primer lugar, el servidor IRC está mantenido por terceras partes. También es difícil descubrir el canal entre los miles

[7] Hay que indicar que no todos los gusanos incluyen herramientas DDoS.
[8] También denominado *malware*.

que existen en un servidor típico. Aún cuando se descubra el canal, sólo se podrá eliminar con la cooperación de los administradores del servidor IRC, y ésta puede ser una difícil tarea, especialmente en el caso de servidores foráneos.

De igual modo que para los procesos de escaneo y propagación, también existe una gran disponibilidad de herramientas para el control de la red de agentes que realizan un seguimiento de los mismos y proporcionan mecanismos para el envío de comandos a todos ellos de una sola vez. Así, mediante un solo comando, el atacante puede ordenar a todos los agentes la ejecución de una inundación a una víctima elegida.

Una recopilación de las características de las herramientas más extendidas en la ejecución de ataques DDoS se puede encontrar en [Houle et al., 2001]. Entre ellas se pueden destacar las herramientas *Trinoo* [Dittrich, 1999a], *Tribe Flood Network* [Dittrich, 1999c], también denominada *TFN*, *Stacheldraht* [Dittrich, 1999b], *Shaft* [Dietrich et al., 2000], *Tribe Flood Network 2000 (TFN2K)* [CERT Coordination Center, 1999], y las utilidades *Mstream* [CERT Coordination Center, 2000] [Dittrich et al., 2000], *Trinity* [Hancock, 2000] y *Knight* [CERT Coordination Center, 2001f].

1.4.3 Ataques de denegación de servicio a baja tasa

De especial relevancia para el estudio realizado en este trabajo es la existencia de ataques de denegación de servicio que utilizan la técnica de inundación pero que, sin embargo, no envían una tasa alta de tráfico a la víctima del ataque. A estos ataques se les denomina ataques de denegación de servicio a baja tasa.

El concepto de baja tasa presentado puede ser objeto de discusión. Aunque no está completamente definido y acotado en este ámbito, se puede tomar como punto de partida el trabajo presentado en [Kuzmanovic and Knightly, 2003]. En él se muestra el concepto de alta tasa como aquella que genera una anomalía estadística en monitores de red (e.g. [Estan and Varghese, 2002] [R.Mahajan et al., 2001] [Snoeren et al., 2001]) de forma que el ataque pueda ser potencialmente detectado, el atacante identificado y los efectos de dicho ataque mitigados. Por tanto, para los efectos del estudio presentado en este trabajo, en adelante se entenderá como baja tasa aquella que permite al atacante ejercer su intrusión sin que pueda ser detectado por los sistemas de detección de intrusos que se basan en la observación de anomalías estadísticas debidas a tasas de tráfico más elevadas de lo usual.

Para poder llevar a cabo este tipo de ataques se debe utilizar necesariamente el conocimiento de alguna vulnerabilidad que, una vez explotada, permita denegar el servicio con baja tasa de tráfico de inundación. Por ello, se pueden clasificar en las categorías de ataques de inundación y también en la de vulnerabilidad, y su objetivo puede ser cualquiera de los presentados en la clasificación mostrada en el anterior apartado.

El ejemplo más significativo de los ataques DoS de baja tasa es el presentado por Kuzmanovic et al. en [Kuzmanovic and Knightly, 2003]. Este ataque, denominado ataque de baja tasa al protocolo TCP, estructura su funcionamiento en torno a una vulnerabilidad presente en el mecanismo de control de flujo del protocolo TCP, tal y como a continuación se describe.

El control de flujo en TCP opera en dos escalas temporales diferentes. Para escalas temporales pequeñas, aproximadamente de la magnitud del tiempo de ida y vuelta (RTT, del inglés *Round Trip Time*) que típicamente toman valores desde decenas a centenas de milisegundos, TCP realiza un control del tamaño de la ventana de transmisión variándolo mediante incrementos aditivos y decrementos multiplicativos (AIMD - *Additive-increase multiplicative-decrease*). Sin embargo, cuando se genera una congestión severa en la cual se producen múltiples pérdidas de paquetes, TCP opera en una escala temporal grande, del tamaño de un temporizador de retransmisión (RTO[9] del inglés *Retransmission Time-Out*). En estos casos, los flujos reducen su ventana de congestión a un segmento solamente y esperan durante un periodo de tiempo RTO tras el cual el segmento es reenviado en caso de no llegar la confirmación. Si se producen pérdidas subsecuentes, el valor de RTO se va duplicando. Cuando un segmento se recibe correctamente, TCP comienza de nuevo con el mecanismo AIMD.

El ataque de baja tasa al protocolo TCP consiste en una secuencia de ráfagas de ataque, maliciosamente separadas temporalmente, realizadas de tal forma que, durante una escala de tiempo equivalente a RTT, sean capaces de generar suficientes pérdidas en los flujos, de modo que TCP entre en el mecanismo de espera del temporizador RTO. Si el periodo de repetición de las ráfagas del ataque se aproxima al temporizador RTO, en el momento en que se produce la retransmisión por parte de TCP se vuelven a producir pérdidas, de modo que el flujo TCP duplicará el valor de RTO, obteniendo finalmente una tasa efectiva de transmisión cercana a cero.

El hecho de que los paquetes de ataque se envíen a la víctima en forma de una onda de tipo ON/OFF, en la cual solamente se envían paquetes a una tasa elevada durante el periodo ON, produce el efecto de que la tasa global de paquetes de ataque, muestreada durante un periodo de tiempo especificado, resulte en valores bajos. Esto puede hacer que los sistemas de detección basados en la medición de tasas altas de tráfico tengan serias dificultades para detectar el ataque y que, por tanto, el ataque pueda ser etiquetado como de baja tasa.

Además del ataque de baja tasa contra el protocolo TCP, la presente tesis doctoral muestra el estudio de un nuevo tipo de ataques de baja tasa: los ataques DoS a baja tasa contra servidores. La aparición de este nuevo tipo de ataques muestra que el ataque presentado en [Kuzmanovic and Knightly, 2003] no es más que la punta del iceberg de una familia nueva de potenciales ataques DoS.

1.5 Métodos de defensa para los ataques de denegación de servicio

En este apartado se realiza una revisión de los diferentes esquemas de defensa actualmente propuestos para los ataques de denegación de servicio conocidos, ofreciéndose una visión de la metodología seguida hasta la presente para luchar contra ellos. Así mismo, se hará especial énfasis en los métodos de defensa actualmente conocidos para ataques de baja tasa, con el fin de tener una visión del alcance de los efectos que dichos ataques pueden tener.

[9] Su valor es típicamente de 1 segundo.

A continuación se resumen las estrategias de defensa más importantes aplicadas en los campos de prevención, detección y respuesta. A lo largo de este apartado, las referencias a ataques DoS también implicarán a aquellos que son distribuidos, es decir, DDoS. Solamente cuando haga falta diferenciar entre ambos se realizará una referencia explícita.

1.5.1 Prevención de ataques DoS

Las estrategias de prevención tratan de eliminar la posibilidad de que un ataque se efectúe antes de que éste se produzca. Estas aproximaciones introducen cambios en los protocolos, aplicaciones y sistemas para fortalecerlos contra los intentos de ataque. La prevención, referida a los ataques DoS, tiene como objetivo aminorar el riesgo de sufrir algunos de los ataques de vulnerabilidad, dificultar al atacante la tarea de conseguir una cantidad de agentes elevada y reducir las probabilidades de éxito del ataque. Sin embargo, aún jugando un papel fundamental para la seguridad, la prevención no elimina la amenaza que suponen los ataques de denegación de servicio.

En el campo de la prevención de ataques DoS, se podrían clasificar las posibles medidas en cuatro grandes grupos [Mirkovic and Reiher, 2004]:

- *Mecanismos de seguridad del sistema*:

  Estos mecanismos tratan de incrementar la seguridad global del sistema, mediante la protección contra accesos ilegítimos, eliminando fallos (*bugs*) en las aplicaciones, actualizando las implementaciones de los protocolos para impedir intrusiones y la utilización del sistema con fines perversos, etc.

  La capacidad de los ataques de denegación proviene generalmente del gran número de máquinas comprometidas que generan flujos de ataque de forma coordinada. Si estas máquinas fueran seguras, los atacantes perderían su capacidad de reclutamiento y la amenaza que suponen los ataques DDoS desaparecería. Por otro lado, los sistemas vulnerables a intrusiones pueden convertirse por sí mismos en víctimas de ataques en los cuales el atacante, habiendo conseguido acceso ilimitado a la máquina, borra o altera sus contenidos. Por tanto, la implementación de mecanismos de seguridad del sistema evita la proliferación de máquinas vulnerables.

  Algunos ejemplos de mecanismos de seguridad del sistema incluyen la monitorización de accesos al mismo [Jajodia et al., 1995] [Tripwire, 2007], la instalación de cortafuegos [Strassberg et al., 2003], el mantenimiento de una configuración actualizada, el uso de antivirus [McAfee, 2007], el cierre de puertos no utilizados, la actualización automática del software y parches de seguridad [X.Geng and A.B.Whinston, 2000], la inhibición de la difusión IP en los elementos de la red, la prueba y chequeo de las aplicaciones en un entorno especializado [Provos, 2003], el establecimiento de listas de acceso para los recursos críticos [Cisco, 2007b], o la utilización de sistemas basados en legitimación de clientes [O'Brien, 2007].

- *Mecanismos de seguridad en protocolos*:

  Los mecanismos de seguridad en protocolos abordan el problema de un diseño defectuoso en los protocolos de comunicaciones. Como se ha comentado previamente (véase el Apartado 1.4.2), muchos protocolos contienen operaciones que son simples para el cliente pero costosas para el servidor. Estas asimetrías del protocolo se pueden explotar para agotar los recursos de la máquina servidora mediante el envío continuado de múltiples peticiones de servicio.

  Una primera aproximación para resolver estos problemas consiste en la validación de la fuente que origina los mensajes, de modo que se pueda eliminar el problema del *IP spoofing*. En esta línea, se han propuesto diferentes alternativas de filtrado de paquetes de información, tales como *ingress filtering*, propuesta por Ferguson y Senie [Ferguson and Senie, 2001], o también su equivalente para el tráfico saliente, denominada *egress filtering* [Center, 2007]. Además, han aparecido técnicas más avanzadas de filtrado, tales como el filtrado de paquetes basado en la ruta que siguen, propuesto por Park y Lee [Park and Lee, 2001], o también basándose en un histórico de las direcciones IP que han realizado comunicaciones previas, tal y como se propone en [Peng et al., 2003].

  Otros ejemplos de estas medidas incluyen guías para un diseño de protocolos seguros en los que los recursos se reservan en el servidor solamente después de que se haya realizado cierta autentificación [Leiwo et al., 2000] [Meadows, 1999], o de que el cliente haya realizado una operación también costosa (*puzzles*) [Aura et al., 2003], o bien el desarrollo de potentes servidores *proxy* que completen las conexiones TCP [Schuba et al., 1997]. Estos mecanismos evitan que el atacante utilice ciertos mecanismos de *spoofing* y permiten validar la fuente cuando los esquemas de filtrado fallan. La mayoría de ellos se basa en requerir un cierto trabajo por parte del cliente. Ahora bien, dicho trabajo debe ser igualmente costoso para todos los clientes, ya que en caso contrario, el atacante podría comprometer una máquina potente que resolviera rápido los trabajos. El problema que tienen estas aproximaciones es que ralentizan el servicio aun cuando no existe denegación de servicio. La investigación reciente [Laurie and Clayton, 2004] sugiere que la implementación de trabajos suficientemente costosos como para causar problemas a los atacantes inflige un daño excesivo a los clientes legítimos.

  Otra herramienta recientemente propuesta para la validación de la fuente es la de *Reverse Turing Tests* [Ahn et al., 2003]. El test más común de este tipo es el que presenta un dibujo borroso o con ruido y solicita al usuario que escriba los símbolos que aprecia.

- *Mecanismos de supervisión de recursos*:

  Los mecanismos de supervisión de recursos controlan el acceso de cada usuario a los recursos basándose en los privilegios que posee dicho usuario y en su comportamiento. Estos mecanismos garantizan un servicio adecuado a los usuarios legítimos, a la vez que deniegan el acceso a los que no tienen permiso. Obviamente, con el objetivo de evitar el robo de identidad, prácticamente todas estas medidas se utilizan conjuntamente con mecanismos para verificar la identidad de los usuarios, es decir, con métodos de autentificación. Algunas propuestas para realizar esta supervisión

de recursos se presentan en [Zheng and Leiwo, 1997] [Spatscheck and Peterson, 1999] [Garg and L. Narasimha Reddy, 2001] [Lau et al., 2000].

- *Mecanismos de multiplicación de recursos*:

  Estos mecanismos pretenden dotar de abundantes recursos a los sistemas para contrarrestar la amenaza que supone el agotamiento de los mismos por parte de un posible ataque DoS. La aproximación más común consiste en contratar un ancho de banda elevado y desplegar un número más o menos elevado de servidores detrás de un balanceador de carga. Los servidores pueden compartir la carga de igual modo a cualquier hora, o bien se pueden dividir en servidores principales y de respaldo, los cuales se activarán cuando las máquinas principales no pueden procesar toda la carga.

  Existen otras formas de implementar estos mecanismos más apropiadas para organizaciones que poseen numerosos servidores distribuidos a lo largo de *Internet*. En este caso, las peticiones de cliente, en caso de sobrecarga, se pueden redirigir a los servidores más cercanos o a los menos cargados.

  Estos mecanismos de multiplicación de recursos (también denominados de sobredimensionamiento) permiten no solamente tratar y defenderse de los ataques DoS, sino también cursar las ráfagas de tráfico legítimo debidas a determinadas situaciones en las que el servicio es muy requerido (por ejemplo una votación por *Internet*). A este tipo de tráficos se les denomina *flash crowds*. Un buen estudio de la similitud entre estos tráficos y el aportado por un ataque DoS se puede encontrar en [Jung et al., 2002].

Además de esta clasificación de las medidas preventivas, es importante señalar que existen algunas líneas de investigación adicionales en torno a la prevención de los ataques. Concretamente, es destacable la investigación respecto a los sistemas denominados *honeypots* [Weiler, 2002]. Un *honeypot* es un sistema similar al real, pero que se construye con una seguridad limitada de forma consciente y se utiliza para hacer de cebo para posibles atacantes, los cuales creen estar atacando un sistema real de producción. Los objetivos de estos sistemas son obtener datos acerca de los ataques que se realizan, aprender de los métodos que se utilizan y también poder detectar la posición de los atacantes; todo esto a la vez que los sistemas reales no son perjudicados.

Aparte de la multitud de medidas preventivas aquí presentadas para el tratamiento de los ataques DoS, la historia de la seguridad en redes de computadores muestra que, aunque con ellas se reduce de forma considerable la frecuencia e impacto de los ataques de denegación de servicio, estas medidas no son del todo efectivas. Es necesario, por tanto, el desarrollo de medidas de detección y también de respuesta a los ataques.

Medidas preventivas para ataques DoS de baja tasa

Ante la aparición del ataque DoS de baja tasa contra el protocolo TCP presentado en [Kuzmanovic and Knightly, 2003], en los últimos años se ha venido trabajando también en el campo de la prevención de este tipo de ataques. Ahora bien, hasta la presente solamente se ha entendido este ataque de forma aislada, de modo que las medidas preventivas propuestas han sido elaboradas *ad-hoc*.

Una medida preventiva para reducir la probabilidad de sufrir un ataque DoS de baja tasa contra el protocolo TCP es la propuesta por Yang en [Yang et al., 2004]. Consiste básicamente en aleatorizar el temporizador RTO de modo que sea difícil, para un atacante, la tarea de sincronizar la onda de ataque ON/OFF con el valor de dicho temporizador. Es evidente que las desventajas de esta medida consisten en una afectación al rendimiento del protocolo y también en la necesidad de modificar el funcionamiento de TCP. Esto hace que el despliegue de esta medida sea complejo.

1.5.2 Estrategias de detección de ataques DoS

El objetivo de la detección de ataques DoS es determinar la ocurrencia de cada intento de realización de un ataque de denegación de servicio tan pronto como sea posible, y con una máxima fiabilidad. Existen dos tipos de fallos que se pueden producir en este proceso: los *falsos positivos* y los *falsos negativos*. En el caso de los falsos positivos, el sistema de detección indica que se está produciendo un ataque cuando realmente no es así. El falso negativo se produce cuando un ataque se está llevando a cabo y el sistema de detección no es capaz de detectarlo. Para realizar la detección de ataques se utilizan típicamente unas herramientas denominadas *sistemas de detección de intrusiones* (IDS, del inglés: *Intrusion Detection System*) [Denning, 1987]. El objetivo de dichas herramientas consiste en realizar la detección de ataques reduciendo al máximo la tasa de falsos positivos y también la de falsos negativos.

Centrando la atención en la detección de ataques, se puede decir que existen dos modelos o paradigmas básicos: la detección basada en firmas y la detección basada en anomalías.

Detección de ataques DoS basada en firmas

Cuando se realiza el proceso de detección mediante el análisis del tráfico de red, una *firma* es un patrón de actividad conocido y caracterizado que permite identificar cuándo dicho tráfico se comporta de una determinada manera [Axelsson, 2000]. Una vez que se entiende el funcionamiento específico de un tipo de ataque, es sencillo extraer una firma o patrón de comportamiento de dicho ataque y especificarla, habitualmente, utilizando un lenguaje formal.

Los mecanismos de detección de ataques de denegación de servicio basados en firmas almacenan estos patrones de ataque en una base de datos. A continuación, monitorizan las comunicaciones y las comparan con las entradas en la base de datos para descubrir ocurrencias de ataques DoS. Ocasionalmente, es necesaria la actualización de la base de datos con nuevos patrones o firmas que se van descubriendo.

Este método de detección hace que los ataques conocidos cuyas firmas están identificadas sean fácilmente detectables, lo que permite que el nivel de falsos positivos se reduzca. Por el contrario, la principal desventaja de esta aproximación consiste en que solamente se pueden detectar ataques conocidos, no permitiendo la detección de nuevos tipos de ataque, ni siquiera para aquellos que supongan o impliquen variaciones mínimas de otros existentes.

Algunos ejemplos muy extendidos de IDS basados en firmas que analizan tráfico de red son los detectores de intrusiones SNORT [SourceFire, 2007], CISCO NetRanger [Cisco, 2007a], NID [Capability, 2007], SecureNet PRO [Mimestar.com, 2007], RealSecure [Systems, 2007] y NFR-NID [NFR Security, 2007].

Detección de ataques DoS basada en anomalías

Los mecanismos que utilizan detección de anomalías se basan en la existencia de una caracterización del comportamiento normal o anormal de un sistema, de modo que su funcionamiento en un instante determinado puede ser comparado con el modelo existente para determinar si se está produciendo o no una anomalía.

Una de las grandes dificultades de este método de detección reside en la definición del modelo. Para ello, se puede definir un *modelo de normalidad*, el cual especifica el comportamiento que se va a considerar normal en el sistema, y/o un *modelo de anormalidad*, que se corresponde precisamente con el comportamiento del sistema en situaciones anormales. En el primer caso, se detectará un ataque cuando se produzca un comportamiento que difiere del modelo y, en el segundo, será en el momento en que el comportamiento responda al modelo cuando se deducirá que se está produciendo un ataque.

Existen múltiples aproximaciones e implementaciones para la detección de ataques DoS basada en anomalías. Para ilustrar los métodos más comúnmente seguidos en este proceso, algunas de ellas se describen a continuación.

Talpade [Talpade et al., 1999] propuso un sistema escalable de monitorización denominado NOMAD. Este sistema detecta las anomalías realizando un análisis estadístico de la información contenida en las cabeceras IP. Se puede utilizar para detectar anomalías en una red local, pero no permite clasificar el tráfico agregado procedente de diferentes fuentes.

Otro método de detección de ataques DoS utiliza la información de gestión (MIB, del inglés: *Management Information Base*) de los encaminadores. Los datos contenidos en las MIB de un encaminador incluyen parámetros estadísticos sobre el tráfico y el encaminamiento. Cabrera [Cabrera et al., 2001] propuso la identificación de determinados patrones estadísticos en diferentes parámetros para detectar de forma temprana la aparición de ataques DoS. Esta aproximación obtiene resultados prometedores para determinados tipos de ataque pero necesita ser evaluada en entornos de redes reales.

Un mecanismo denominado *muestreo y filtrado de paquetes disparado por congestión* fue propuesto por Huang y Pullen [Huang and Pullen, 2001]. Éste consiste en que si se produce congestión y se descartan paquetes, se toma un subconjunto de ellos y se analizan estadísticamente. Si se detecta una anomalía en estos paquetes, ésta se identifica como un ataque.

Lee y Stolfo [Lee and Stolfo, 1998] utilizaron técnicas de minería de datos (*data mining*) para descubrir patrones en las características de un sistema que describieran el comportamiento de los programas y los usuarios. Con ello se puede crear un clasificador que reconozca anomalías e intrusiones. Este método utiliza como información variables o parámetros medidos en el propio sistema y no en los paquetes de información que viajan por la red.

Una mejora de esta técnica [Lee et al., 1999] utiliza los resultados provenientes de múltiples modelos para mejorar la detección.

Mirkovic [Mirkovic et al., 2002] propuso un sistema denominado D-WARD. Éste realiza la detección de DoS basándose en la hipótesis de que los ataques de denegación de servicio deben ser filtrados tan cerca de la fuente como sea posible. D-WARD se instala en los encaminadores frontera de una red y monitoriza el tráfico tanto entrante como saliente. Si se detectan asimetrías en las tasas de los paquetes generados por un sistema interno, se limita su tasa mediante un mecanismo de filtrado. El problema que presenta este sistema es que se pueden producir muchos falsos positivos debido a asimetrías en tasas de baja duración. Además, algunos flujos legítimos, como ciertos flujos UDP en tiempo real, son asimétricos.

En [Gil and Poleto, 2001], Gil y Poletto propusieron una estructura de datos heurística (MULTOPS) que recopila información basándose en las direcciones IP origen (modo orientado al ataque) o destino (modo orientado a la víctima). Cada elemento de red recopila información estadística en una estructura multinivel, de modo que solamente cuando una dirección o rango de direcciones supera un nivel de tasa de tráfico dado se comienzan a recopilar datos con mayor nivel de detalle. Este sistema permite detectar el origen del ataque (o rango de direcciones falsas desde las que se ejecuta) y también las máquinas víctimas de un ataque. Entre sus desventajas se puede anotar que requiere la reconfiguración de los encaminadores, mucha memoria y que no es capaz de detectar ataques con *spoofing* aleatorio generado por una sola fuente, o por un número lo suficientemente elevado de agentes.

Detección de ataques DoS de baja tasa

Como ya se ha comentado en el Apartado 1.4.3, los ataques de denegación de servicio a baja tasa consiguen inundar a la víctima del ataque mediante el envío de información a tasas no elevadas. Esto es posible mediante la explotación de una vulnerabilidad específica presente en el sistema a atacar o en sus mecanismos de comunicación.

Todas las aproximaciones de detección estadística de anomalías para ataques DoS anteriormente planteadas, y muchas otras no citadas, presentan una característica común: basan la detección de las anomalías en la localización de tasas elevadas de tráfico por parte de una fuente o dirigidas a un destino. Es por ello que la mayoría presentan problemas cuando el ataque se produce a baja tasa (ver [Siris and Papagalou, 2006]). Este hecho es importante ya que, en caso del diseño de ataques DoS a baja tasa, como el que se presenta en este trabajo, los mecanismos existentes no están preparados para efectuar la detección. Concretamente, dicha detección produce una elevada tasa de falsos negativos cuando el umbral de detección es poco sensible, o de falsos positivos en caso contrario, debido a que los sistemas de detección propuestos necesitan variaciones considerables de las magnitudes a medir; en este caso, la cantidad de tráfico con destino a la víctima.

Como alternativa, aparecen los mecanismos de detección basados en firmas como solución para abordar el problema de los ataques DoS de baja tasa. En esta línea de investigación han aparecido dos propuestas relacionadas con el ataque de baja tasa al protocolo TCP. El trabajo de Sun [Sun et al., 2004] propone la detección del patrón de ataque

ON/OFF mediante el estudio de una señal formada a partir del tráfico con destino a la víctima. La autocorrelación de dicha señal permite obtener unas características determinadas de la señal, que se comparan con el patrón de ataque mediante la técnica DTW (*Dynamic Time Warping*) [Keogh, 2002]. Por otro lado, Shevtekar y otros [Shevtekar et al., 2005] proponen la monitorización de los tiempos entre paquetes para cada flujo. A partir de esta información son capaces de detectar las frecuencias de los periodos de actividad y de inactividad de los flujos. Comparando dichas frecuencias con el tiempo de ida y vuelta correspondiente a cada flujo se determina si existe o no un ataque de baja tasa a TCP.

El problema de las soluciones propuestas es el inherente a los sistemas de detección basados en firmas. La aparición de nuevos métodos de ataque a baja tasa, tal y como se hace en este trabajo, hace necesario el desarrollo de nuevos mecanismos debido a que los existentes no resultan válidos para su detección.

1.5.3 Mecanismos de respuesta ante los ataques DoS

Para que la defensa contra un ataque de denegación de servicio sea lo más efectiva posible es recomendable conjugar el sistema de detección (IDS), con un sistema de respuesta o IRS (*Intrusion Response System*), de modo que, ante la detección de un ataque, el sistema de detección debe extraer la información sobre el paquete o flujo de tráfico que produce el ataque y notificarlo al sistema de respuesta para que éste actúe en consecuencia. Una arquitectura que, de forma automática, es capaz de generar una respuesta ante la detección de una intrusión se suele denominar IPS (*Intrusion Prevention System*) [Ierace et al., 2005].

Una vez que se notifica la detección de un ataque DoS, los mecanismos de respuesta tratarán de mitigar los efectos de dicho ataque y mejorar la situación para los usuarios legítimos. Existen tres tareas principales a realizar para conseguir este objetivo: control del tráfico, rastreo del ataque y diferenciación de servicios. Veámoslas más en detalle:

- *Control del tráfico*:

  La primera respuesta, y también la más deseable, a un ataque de denegación de servicio consiste en desechar todo el tráfico atacante. Esto provocaría que el ataque no tuviera efecto alguno sobre los usuarios legítimos. El problema reside en que, dado que los modelos de detección no son perfectos y no caracterizan completamente el tráfico de ataque, para realizar un control de tráfico eficiente hay que decidir qué tráfico desechar y cuál no.

  Los dos mecanismos básicos para realizar el control del tráfico son el *filtrado* y la *reducción de la tasa* del tráfico en cuestión. Mientras que el filtrado desecha todos los paquetes indicados como sospechosos por el sistema de detección, el mecanismo de reducción de la tasa fuerza al tráfico sospechoso a no superar una determinada tasa. En esta línea se puede citar el trabajo de Mahajan [Mahajan et al., 2002], en el que se justifica la necesidad de frenar el tráfico de ataque en un punto lo más cercano a la fuente como sea posible. Esto se puede conseguir con una acción tan simple como una mera desconexión de cable, o bien un filtrado en un encaminador. Otro tipo de mecanismos de control de tráfico son los denominados mecanismos de *control de congestión de agregados locales*, propuestos en [R.Mahajan et al., 2001] [Ioannidis and Bellovin, 2002], y

que consisten en determinar las propiedades del tráfico que se desecha en situación de congestión para caracterizar la firma del ataque y así poder implementar un filtrado en los encaminadores.

- *Rastreo del ataque*:

  El rastreo del ataque tiene tres propósitos fundamentales: identificar los agentes que están ejecutando el ataque DoS, tratar de conseguir la identificación del atacante y, finalmente, obtener información para poder hacer el control de tráfico tan cerca de la fuente como sea posible. El primero de ellos se puede conseguir, aunque puede ser complicado cuando cientos de agentes están atacando. El segundo es casi imposible hoy en día, debido a la utilización, por parte del atacante, de los saltos. En esta línea, se han hecho numerosos esfuerzos de investigación y han surgido muchas propuestas, como las diferentes técnicas de *IP traceback* [Song and Perrig, 2001] [Dean et al., 2001] [Bellovin, 2001] [Savage et al., 2000].

  Aunque tradicionalmente se ha considerado importante la realización del rastreo del ataque, algunos estudios [Ioannidis and Bellovin, 2002] argumentan que incluso aunque el camino que sigue el ataque haya sido identificado, no está claro cuáles son los siguientes pasos a seguir para mitigar los efectos del mismo.

- *Diferenciación de servicios*:

  Las técnicas de diferenciación de servicios tratan de dar prioridad a unos servicios frente a otros cuando los recursos disponibles son escasos. Es un modo de priorizar los tráficos que se cursan por parte de una red o sistema. En este campo, las arquitecturas de servicios integrados (*IntServ*) y las de servicios diferenciados (*DiffServ*) [Zhao et al., 2000] han surgido como las principales soluciones para distinguir los diferentes flujos de tráfico que atraviesan la red.

  Para combatir los ataques DoS se han utilizado técnicas de encolado. Existen muchas disciplinas de cola en los encaminadores, siendo las más ampliamente utilizadas las de encolado basado en clases (CBQ: *Class-based queuing*) y del conformado de tráfico. Estas técnicas han demostrado conseguir un cierto nivel de calidad de servicio ante ataques DDoS a servidores web [Kargl et al., 2001].

Aunque se han explorado numerosas técnicas de respuesta a ataques DoS, aún no se ha conseguido resolver el problema y amenaza que suponen estos ataques, debido fundamentalmente a que los mecanismos de detección tampoco están suficientemente desarrollados como para evitar la afectación al tráfico legítimo al ejecutar una determinada medida de respuesta.

Mecanismos de respuesta a los ataques DoS de baja tasa

La ya citada aparición del ataque DoS de baja tasa ejecutado contra el protocolo TCP [Kuzmanovic and Knightly, 2003] ha motivado también la investigación de métodos de respuesta adaptados especialmente para este tipo de ataques. En esta línea, se han proporcionado principalmente dos soluciones básicas.

La primera de ellas, planteada en el propio trabajo anteriormente mencionado de Kuzmanovic, y posteriormente desarrollada por Yang [Yang et al., 2004], consiste en la aleatorización de los temporizadores del protocolo TCP y, más concretamente, de los tiempos de retransmisión RTO. Esta medida, también planteada como preventiva en el Apartado 1.5.1, puede convertirse en una medida de respuesta si es aplicada solamente a partir de la detección del ataque. Como ya se ha comentado, si bien esta solución se muestra efectiva para este tipo de ataque, precisa la modificación del protocolo TCP, característica que hace complejo su despliegue.

Otro esquema diferente es el propuesto en [Sun et al., 2004], en el que los diferentes encaminadores de la red ejecutan un algoritmo de detección de flujos maliciosos. Cuando se detecta alguno, se notifica a los encaminadores de los que procede el flujo para que realicen la detección a su vez, y se aplica el algoritmo DRR (*Deficit Round Robin*) para la asignación de ancho de banda y protección de recursos.

1.6 Conclusiones del capítulo

El presente capítulo realiza una revisión del estado del arte de la seguridad en el contexto de las redes de comunicaciones, haciendo especial énfasis en los ataques de denegación de servicio. En la línea del estudio que se presenta en esta tesis doctoral, la revisión de estos conceptos ilustra ciertas conclusiones que sirven de punto de partida para el trabajo de investigación. Las principales conclusiones que se pueden citar son las siguientes:

- La seguridad es un aspecto que ha tomado una especial relevancia en el contexto de las redes de comunicación. Esto implica que todo diseño y arquitectura relacionados con las tecnologías y redes de comunicación deben estar impregnados de consideraciones sobre seguridad.
- Existen numerosas amenazas a la seguridad. Entre ellas, los ataques de denegación de servicio aparecen como un problema de gran importancia. Además, con las tecnologías actualmente desarrolladas no existe un método que resuelva este problema de seguridad. Aunque existen numerosos equipos de investigación trabajando sobre el tema, aún falta encontrar soluciones efectivas que determinen el modo de atajar el problema.
- Entre los ataques de denegación de servicio conocidos, han aparecido recientemente unos ataques que poseen una característica especial: los ataques de inundación realizados a baja tasa. Este tipo de ataques constituye un problema dentro de la defensa contra los ataques DoS, que complica más, si cabe, la adopción de una solución al problema general.

Capítulo 2

Modelado del escenario de ataque

2.1 Introducción

Para comprender el funcionamiento de los ataques DoS de baja tasa a servidores, antes de especificar los detalles relativos a su ejecución, es necesario ilustrar algunos aspectos adicionales. En primer lugar, hay que detallar los escenarios en los que pueden aparecer dichos ataques y modelar los diferentes componentes y partes integrantes de éstos. En este sentido, es fundamental realizar el modelado genérico de un servidor víctima potencial del ataque. Además, es necesario determinar las características del canal a través del cual el atacante llevará a cabo su ataque. Por último, es también conveniente concretar el comportamiento de los usuarios afectados por el ataque, es decir, aquellos que tratan de acceder de forma legítima al servicio atacado.

En este capítulo se presentan los fundamentos de los aspectos anteriormente mencionados para, en capítulos posteriores, poder profundizar en detalle sobre la forma en que se articulan los ataques de denegación de servicio a baja tasa contra servidores.

El resto del capítulo está estructurado de la siguiente forma. Primeramente, el Apartado 2.2 presenta el escenario donde pueden aparecer los ataques DoS de baja tasa contra servidores. En el Apartado 2.3 se desarrolla el modelo genérico para un servidor y, a lo largo del mismo, se analizan los aspectos de su comportamiento necesarios para ilustrar el funcionamiento del ataque. Posteriormente, en el Apartado 2.4 se modelan también los tráficos de los usuarios que acceden de forma legítima a dicho servidor. Finalmente, se presentan las principales conclusiones de este capítulo.

2.2 Escenario de estudio

El escenario general en el que se estudiará la ocurrencia de los ataques de baja tasa contra servidores de aplicaciones es el representado en la Fig. 2.1. En ella se pueden observar los siguientes elementos:

- Un servidor que ofrece un determinado servicio a los usuarios finales. Este servidor no presenta, en este escenario, ninguna restricción de diseño ni de arquitectura. Un modelado más extenso del servidor se realizará en el Apartado 2.3.

- Un número variable de usuarios que tratan de acceder legítimamente a los servicios ofrecidos por el servidor. La forma en que se legitiman los usuarios se establecerá bien mediante mecanismos de seguridad, tales como la autenticación, o bien se determinará por el modo o intención con que los usuarios acceden al servicio. Así, un usuario será legítimo si trata de acceder al servicio en la forma en que está prevista. Por ejemplo, si un servicio posee una determinada interfaz de acceso, los usuarios que traten de acceder mediante otro tipo de interfaz no prevista para ellos serán ilegítimos. También se considerará legítimo a todo usuario que trata de obtener del servicio aquellos productos que éste genera, es decir, si un usuario trata de que el servicio se comporte de un modo diferente al previsto, se considerará un atacante. En adelante se denominará simplemente *usuarios* a estos usuarios legítimos. Por contraposición, a aquellos usuarios que no cumplen las condiciones para ser legítimos se les denominará *atacantes*.

- Un número variable de máquinas desde las que uno o varios atacantes realizan el ataque al servidor. A estas máquinas se las denominará en adelante *máquinas atacantes*. Desde ellas es desde donde se envía el tráfico de ataque al servidor con el objetivo de provocar una denegación de servicio.

- Una red de transporte que permite, tanto a los usuarios como a los atacantes, el acceso al servidor. Se asume que esta red de transporte consiste en una red de conmutación de paquetes. La arquitectura típica en la que se reproduce este escenario es la de *Internet*. La red de transporte puede ser pública, privada o incluso estar dividida en varias partes públicas o privadas. Los tráficos procedentes tanto de los usuarios como de los atacantes se agregarán en un punto de entrada al servidor. Éste puede estar protegido por cualquiera de las estructuras de seguridad descritas en el Capítulo 1.

En una situación normal, los usuarios que acceden lo harán de forma legítima y controlada, pretendiendo únicamente la obtención del servicio ofrecido por el servidor. Sin embargo, el servidor podría recibir simultáneamente peticiones procedentes de diversos atacantes con la pretensión de causar una denegación de servicio. Sin pérdida de generalidad, el atacante puede tomar la decisión de realizar su ataque de forma distribuida o centralizada, dependiendo de las medidas preventivas implementadas en la red de acceso, o de sus propias capacidades y recursos. P.ej. si el servidor establece un límite en el número de conexiones realizadas desde un cliente específico, podría ser necesario para el atacante realizar un ataque distribuido.

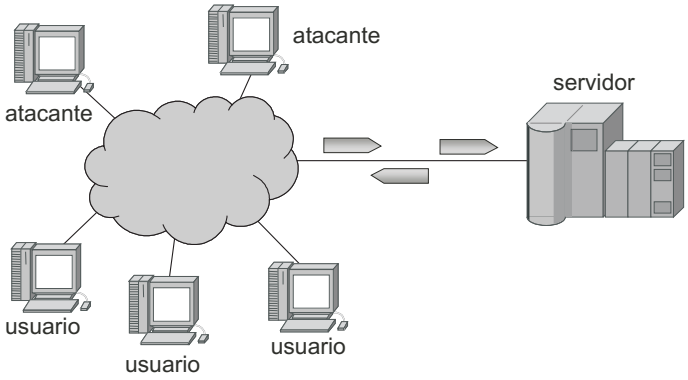


Fig. 2.1: Escenario de estudio para los ataques DoS a baja tasa contra servidores.

2.3 Modelado del servidor

En este apartado se clarifican, en primer lugar, las diferentes características que el servidor presente en el escenario de estudio puede presentar. Posteriormente se propone un modelo flexible para el servidor de modo que se adapte a las diferentes características anteriormente expuestas para los servidores. Finalmente se realiza un análisis detallado de varios aspectos presentes en el comportamiento del servidor que serán útiles para análisis posteriores.

2.3.1 Naturaleza del servidor

El servidor es un elemento insertado en una o varias máquinas con conectividad a la red de transporte, encargado de ofrecer uno o varios servicios a determinados clientes. Puede tener características muy diversas, dependiendo de su arquitectura software, modo de funcionamiento, ubicación, etc. Este hecho permite que se puedan establecer múltiples clasificaciones para los servidores [Stevens et al., 2004], según la característica observada. A continuación se presentarán algunas de estas clasificaciones y tipologías (ver Fig. 2.2).

En primer lugar, si se atiende al modo de funcionamiento del servidor, se puede realizar la siguiente clasificación:

- *Orientado a conexión:*

  En este caso, el acceso al servicio se realiza completando las fases de establecimiento de la conexión, transferencia de datos y cierre de la conexión. Durante la fase de transferencia de datos es posible realizar múltiples peticiones al servidor. Este tipo de servidores se utiliza típicamente para servicios en los que el servicio se desarrolla en varias etapas solicitud-respuesta. Un ejemplo de protocolo orientado a conexión es el que sustenta típicamente este tipo de servicios en *Internet*, denominado TCP [Postel, 1981].

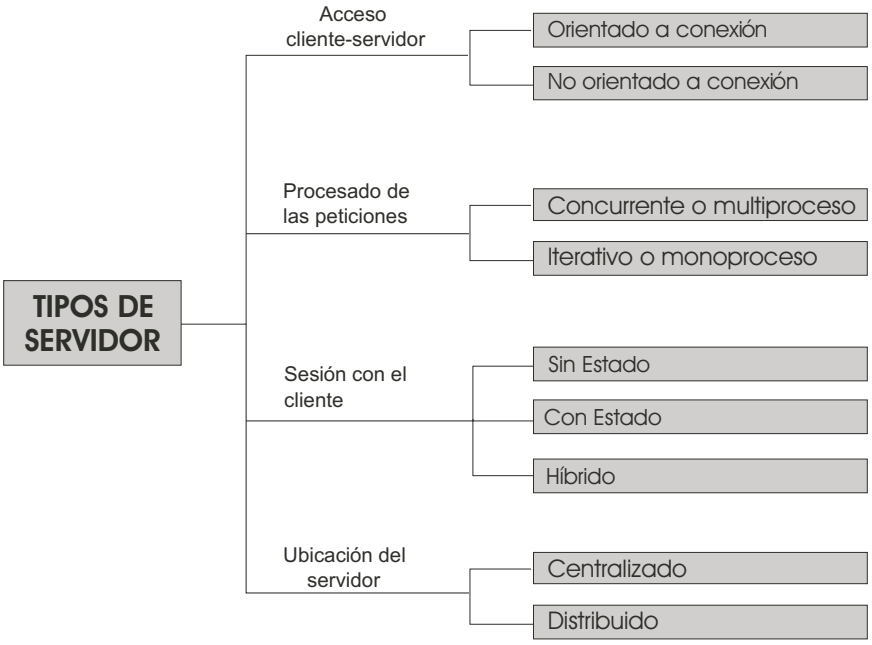


Fig. 2.2: Clasificación de los servidores atendiendo al modo en que prestan el servicio.

- *No orientado a conexión:*

  Los servidores no orientados a conexión aceptan peticiones, las sirven y emiten una respuesta. Ahora bien, no tiene que existir una conexión establecida previamente al envío de las diferentes peticiones, sino que éstas se envían independientemente unas de otras. Un ejemplo de protocolo sobre el que se sustentan este tipo de servicios en *Internet* es el protocolo UDP [Postel, 1980].

Otra clasificación de los servidores puede realizarse atendiendo al esquema temporal en que se procesan las peticiones recibidas. De este modo, un servidor se puede clasificar como:

- *Iterativo o monoproceso:*

  La arquitectura de este tipo de servidores está constituida por un único proceso con una sola hebra, encargado de procesar las peticiones que llegan al servidor. Dada la existencia de una única entidad procesadora, las peticiones se van atendiendo una a una, normalmente según el orden de llegada al servidor (FIFO), de modo que el procesado de una petición bloquea a las siguientes. Es evidente que, a costa de tener una mayor simplicidad en el diseño e implementación del servidor, el tiempo total que cada petición esperará en el servidor será mayor con respecto a otras arquitecturas. Por ello, los servidores iterativos implementan típicamente servicios que no requieren mucho tiempo de procesamiento; p.ej., un servicio que da la hora, o bien servicios

cuya ejecución es imposible para dos usuarios a la vez (p.ej., un sistema de control de movimiento de un brazo mecánico).

- *Concurrente o multiproceso:*

  Un servidor concurrente es aquel capaz de procesar en paralelo varias peticiones. Para ello, la arquitectura del servidor está formada por un proceso principal o padre, encargado de recibir las peticiones que van llegando al servidor, y de adjudicar la tarea del procesado a varias hebras o procesos hijos, que se encargan de ejecutar las instrucciones necesarias. De este modo, la existencia de varios procesos o hebras permite que el procesado de las peticiones se realice virtualmente de forma paralela en el tiempo, es decir, en cada instante se está procesando solamente una petición, pero el tiempo total se reparte para cada petición sin necesidad de que su procesado haya terminado, de modo que parece que se ejecutan en paralelo (concurrencia aparente). También se puede obtener una concurrencia real cuando se dispone de varios procesadores.

  El nivel de paralelismo requerido en el procesado de las peticiones determina el número de hebras y procesos disponibles. Con esta arquitectura de servidor se reduce, con respecto a los sistemas iterativos, el tiempo que las peticiones permanecen en el servidor. Ahora bien, el coste asociado consiste en un diseño e implementación del servicio más complejos.

  Las aplicaciones que utilizan típicamente este tipo de servidores son las que necesitan un tiempo considerable para el procesado de la información. Un ejemplo de servicio que se suele implementar de modo concurrente es el que proporciona un servidor web.

Considerando la forma en que se trata la sesión con el cliente, un servidor se puede clasificar como:

- *Sin estado:*

  Este tipo de servidor no almacena el estado de la "conversación" o sesión que mantiene con el cliente. Los servidores de este tipo son más sencillos de implementar, aunque sustentan solamente los servicios más elementales.

- *Con estado:*

  Estos servidores mantienen información del estado de la sesión con el cliente. Son servidores más complejos de implementar, porque precisan de mecanismos para la comunicación y almacenamiento de la información, aunque tienen la capacidad de implementar funcionalidades más complejas. Un ejemplo de servicio implementado en un servidor con estado es el denominado *la cesta de la compra*, en el cual se permite al cliente seleccionar los productos que desea comprar para, finalmente, dar la orden de ejecución de la compra; conforme el cliente va seleccionando productos, el servidor debe ir almacenando el estado de su selección.

- *Híbridos:*

  En este caso, la información de estado de la sesión se mantiene entre el cliente y el servidor, siendo ambos responsables de almacenar la información que les sea relevante. Ejemplos de implementación de este tipo de servidores son aquellos servidores web que utilizan *cookies*.

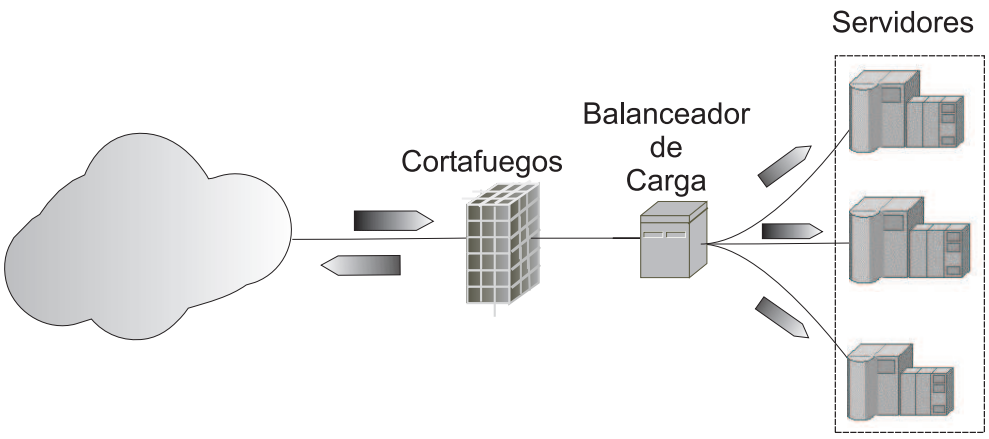


Fig. 2.3: Entorno multiservidor con balanceo de carga a la entrada.

Finalmente, si se atiende a la ubicación física del servidor, se pueden establecer las siguientes categorías:

- *Centralizados:*

  Son aquellos servidores que se ubican físicamente en una única máquina. Dicha máquina será la que tendrá conectividad física a la red de transporte que comunica con los usuarios legítimos y con los potenciales atacantes.

- *Distribuidos:*

  En este caso existen varias réplicas del servidor, las cuales se encuentran distribuidas en diferentes máquinas conectadas individualmente a la red. Una configuración típica es la mostrada en la Fig. 2.3. En ella se pueden observar las diferentes máquinas que albergan los servidores. Además de que puedan existir elementos de seguridad perimetral, como los cortafuegos, o algún elemento adicional, típicamente también se pueden encontrar los llamados equipos de balanceo de carga. Estos equipos reciben las peticiones procedentes de la red de transporte y tienen el cometido de elegir el servidor al que enviarlas. La elección se realiza según unos criterios configurables, tales como la propia carga del servidor, disponibilidad del mismo, distancias administrativas, etc. Un diálogo entre el servidor y estos equipos permite la correcta decisión sobre la distribución de las peticiones entrantes.

Un servidor concreto puede pertenecer a varias de las categorías expuestas. Por ello, a la hora de plantear un modelo para el servidor a considerar, un criterio de diseño será no restringir ninguna de las características de los diferentes tipos existentes.

2.3.2 Modelo del servidor

El modelo que se propone para el servidor, en el entorno en que se desarrollará el ataque DoS a baja tasa, consiste en un módulo que recibe mensajes o peticiones, los procesa y emite las respuestas que sean necesarias. Los mensajes recibidos corresponden a las peticiones de servicio procedentes bien de los usuarios legítimos, bien de los atacantes. El sistema reacciona a estas peticiones recibidas generando mensajes con la respuesta solicitada (que se denominará *mensaje OK*), o simplemente indicando que existe sobrecarga en las colas internas o memorias de la aplicación (que se denominará *evento o mensaje MO*). Es importante hacer notar que, en entornos reales y dependiendo de las características específicas del servidor, el evento MO podría ser observable a través de mensajes de protocolo, registros históricos, señales, o incluso no ser observable en absoluto.

Componentes del modelo

El modelo genérico del servidor, representado en la Fig. 2.4, está compuesto por M máquinas, en cada una de las cuales reside una réplica del servidor. Cada petición que llega al sistema entra a través de un balanceador de carga, que se encarga de distribuirla a una de las réplicas del servidor. Cada una de las M réplicas contiene los siguientes elementos:

- Una cola de servicio.
- Un módulo de servicio.

Estos dos elementos están conectados tal y como se muestra en la Fig. 2.4, donde las flechas muestran el camino del procesado seguido por las peticiones entrantes. El primer bloque funcional que una petición entrante encuentra es la cola de servicio, en la que se almacenan temporalmente las peticiones. Esta cola de servicio tiene una longitud finita de almacenamiento de peticiones. En el caso de que todas las posiciones de una cola estén ocupadas y llegue una nueva petición a dicha cola, se generará un evento MO. En caso contrario, dicha petición se almacenará en la cola.

La cola de servicio puede modelar, por ejemplo, a un registro de conexiones TCP (servidor orientado a conexión), o de peticiones UDP (servidor no orientado a conexión) o simplemente a determinados registros internos de la aplicación donde se guardan las peticiones antes de ser procesadas. En los sistemas Unix con el estándar POSIX, la cola de servicio puede representar al *socket* que almacena las peticiones entrantes para una determinada aplicación en un puerto dado.

Definimos el concepto de *saturación de una réplica del servidor* como el estado en el que la cola de servicio perteneciente a dicha réplica se encuentra completamente llena de peticiones y, por tanto, sin ninguna posición libre. De forma análoga, definimos la *saturación del sistema servidor* como el estado en el cual todas las réplicas se hallan en estado de saturación.

Una vez que una petición recibida en el servidor ha sido almacenada y ha permanecido durante un tiempo en la cola de servicio, el procesado de la misma se realiza en el módulo

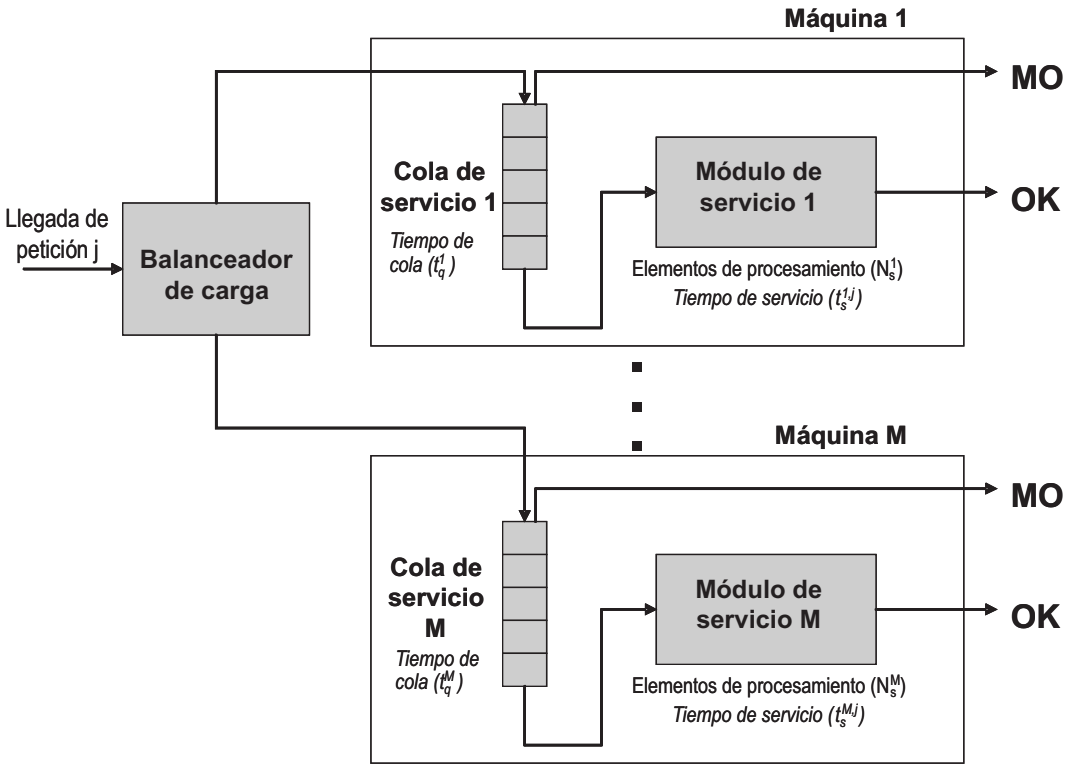


Fig. 2.4: Modelo genérico propuesto para el servidor.

de servicio. Este bloque funcional modela la existencia de un número variable de procesos o hebras, denominados de forma genérica *elementos de procesamiento*. Estos procesos o hebras son iguales entre sí y cada uno de ellos se ejecuta en el marco del sistema operativo y está encargado del procesado de los trabajos requeridos por la petición concreta que se está sirviendo.

El paso de una petición desde la cola de servicio al módulo de servicio representa el comienzo del procesamiento de la petición en la réplica del servidor correspondiente. Por ejemplo, en un servidor ubicado en un sistema Unix con el estándar POSIX, el paso de la cola de servicio (que modela al *socket*, tal y como se ha comentado) hacia el módulo de servicio representa a la llamada al sistema *accept*, cuyo efecto es tomar una conexión del *socket* asociado a un puerto y pasarla a la aplicación para su posterior manipulación [Stevens et al., 2004].

El número de elementos de procesamiento que contiene el módulo de servicio de la máquina i se denota por N_s^i, siendo el total de elementos de procesamiento existentes en el sistema servidor N_s:

$$N_s = \sum_{i=1}^{M} N_s^i \tag{2.1}$$

Los diferentes módulos de procesamiento componentes de un único módulo de servicio se pueden ejecutar bien en un solo procesador o bien en varios, como sucede en el caso de máquinas multiprocesador. El modelo para el módulo de servicio no impone restricción alguna sobre esta característica de implementación.

En resumen, el proceso que sigue una petición entrante a un servidor es el siguiente. En primer lugar, ésta es introducida en un balanceador de carga, el cual distribuye cada petición entrante hacia alguna de las réplicas existentes en el sistema servidor. Una restricción se aplica a este algoritmo de distribución: no se enviará una petición a una réplica del servidor que se encuentre en estado de saturación, excepto cuando el sistema servidor se encuentre completamente saturado, en cuyo caso no importa la réplica escogida, dado que el resultado siempre será la generación de un mensaje MO. Si la petición se encamina a una réplica no saturada, se almacenará en la cola de servicio correspondiente. Cuando a la petición le toca el turno de entrar en el módulo de servicio[1], ésta es procesada por algún elemento de procesamiento y, finalizada esta tarea, una respuesta es generada (mensaje OK) y enviada a quien originó la petición.

Tiempo de latencia de una petición

Definimos el *tiempo de latencia* de una petición, T_L, como la variable aleatoria que representa al tiempo transcurrido desde que se emite una petición al servidor, bien por parte de los usuarios legítimos o bien de los atacantes, hasta que se recibe la correspondiente respuesta.

El tiempo de latencia de una petición concreta j que se encamina para ser servida en la máquina i, $t_L^{i,j}$, que constituye una muestra de la variable aleatoria T_L, se puede evaluar haciendo un seguimiento de los diferentes tiempos que dicha petición va a invertir en seguir el proceso definido en el modelo. Las principales contribuciones a este tiempo son las siguientes:

- *Tiempo de ida y vuelta para la petición j* (rtt^j, del inglés: *round trip time*):

  Es el retardo desde un origen (el atacante o usuario) a un destino (el servidor), y desde este último hasta el primero. Este tiempo no es fijo y puede ser diferente según el mensaje observado, ya que depende de la dinámica y las condiciones existentes en la red de transporte. El muestreo de los diferentes valores de este tiempo para las distintas peticiones da lugar a una variable aleatoria que se notará como RTT.

  En [Elteto and Molnar, 1999] se propone como distribución estadística para la variable aleatoria que representa al tiempo de ida y vuelta la distribución normal truncada. En este trabajo se asumirá, por simplicidad, una distribución normal con media $\overline{RTT}$

[1] No se impone ninguna restricción a la disciplina de servicio de la cola. Por simplicidad, en adelante se considerará para todos los ejemplos y explicaciones que la disciplina de la cola de servicio es de tipo FIFO.

y varianza $var[RTT]$, para lo que se usará la notación $\mathcal{N}(\overline{RTT}; var[RTT])$, de modo que la probabilidad de obtener un valor t en la variable aleatoria RTT será:

$$P(RTT = t) = \mathcal{N}(\overline{RTT}; var[RTT]) = \frac{1}{\sqrt{2\pi \cdot var[RTT]}} \cdot e^{-\frac{(t-\overline{RTT})^2}{2var[RTT]}} \tag{2.2}$$

- *Tiempo de espera en la cola de servicio i* (t_q^i):

  Es el tiempo que la petición invierte en la cola de servicio de la máquina i esperando a ser atendida. Este tiempo, que no será fijo, dependerá principalmente del tiempo que tardarán en ser servidas las peticiones existentes en la cola y en el módulo de servicio en el instante en que la petición en cuestión entra a la cola de servicio. Es por esta razón por la que el tiempo de espera en la cola de servicio no depende de la petición j considerada.

  El muestreo de los tiempos de espera en cola para cada una de las peticiones que entran en las diferentes colas de servicio da lugar a una variable aleatoria que se notará como T_q.

- *Tiempo de servicio en el módulo de servicio i para la petición j* ($t_s^{i,j}$):

  Es el tiempo durante el que la petición j va a permanecer en el módulo de servicio i siendo servida por un elemento de procesamiento.

  El muestreo de los tiempos de servicio para las diferentes peticiones posibles que llegan al servidor y que entran en los distintos módulos de servicio da lugar a una variable aleatoria que se notará como T_s.

  De forma genérica, el modelo de distribución estadística a elegir para T_s dependerá del tipo de peticiones que se realizan al servidor, ya que esto influirá en el tiempo que se emplea en procesarlas. Algunos estudios han tratado de modelar este tiempo en determinados escenarios; por ejemplo, en el caso de un servidor web, se ha propuesto una distribución de tipo *heavy-tailed* para el tiempo de servicio [Liu et al., 2001]. De todos modos, la idoneidad del modelo propuesto dependerá del tipo de tráfico considerado, del perfil de los usuarios y de otros elementos adicionales que no se considerarán en este contexto.

  Para el propósito del estudio presentado en este trabajo será interesante analizar el tiempo de servicio cuando las peticiones que se realizan al servidor son idénticas entre sí. En principio, si las M máquinas que componen el servidor son idénticas, cabe esperar que dichas peticiones, dado que son iguales, involucren tiempos de servicio también iguales. Sin embargo, existen determinadas circunstancias que hacen que esto no sea cierto. En efecto, el tiempo de servicio empleado por dichas peticiones dependerá de factores tales como la máquina en la que se ejecute la petición (en caso de que sean diferentes), la capacidad del procesador en dicha máquina, la ocupación de memoria, el tráfico de red que se está procesando, el número de interrupciones que se están generando, la carga de la CPU, el número de procesos existentes en el instante de procesado de la petición, etc. De este modo, aunque dos peticiones sean idénticas, el tiempo de servicio involucrado en servir a cada una de ellas será diferente. En este caso, la distribución de la variable aleatoria que representa al tiempo de servicio

para las peticiones idénticas que se procesan en el servidor se puede modelar haciendo uso del teorema central del límite [Song, 2004], dado que el número de variables que afectan a dicho tiempo es muy elevado. De este modo, cuando se consideran peticiones idénticas, para la distribución del tiempo de servicio, T_s, se elige una variable normal con la siguiente función de probabilidad:

$$P(T_s = t) = \mathcal{N}(\overline{T_s}; var[T_s]) \tag{2.3}$$

donde $\overline{T_s}$ es la media del tiempo empleado en el procesamiento de las peticiones y $var[T_s]$ su varianza.

Considerando los distintos factores, la expresión que proporciona el valor de una muestra del tiempo de latencia para una petición concreta j que se encamina para su procesado hacia la máquina i,($t_L^{i,j}$), es:

$$t_L^{i,j} = t_q^i + t_s^{i,j} + rtt^j \tag{2.4}$$

De igual modo, para la variable aleatoria tiempo de latencia se obtiene la siguiente expresión:

$$T_L = T_q + T_s + RTT \tag{2.5}$$

2.3.3 Aplicabilidad del modelo

El modelo anteriormente propuesto es aplicable a cualquier tipo de servidor según las clasificaciones anteriormente propuestas en el Apartado 2.3.1.

El hecho de que el servidor sea orientado o no a conexión no viene determinado por el modelo aquí propuesto, es decir, no se impone ninguna restricción sobre el modo de funcionamiento del servidor relativo al establecimiento de conexiones. Esto es debido a que no se establece ninguna condición a la naturaleza de las peticiones, que podrán ser solicitudes de conexión, mensajes de datos, o cualquier tipo de intercambio de información, y tampoco sobre el recurso al que modela la cola de servicio, de manera que puede representar a una cola de conexiones, a un registro de almacenamiento de mensajes o a cualquier elemento de almacenamiento genérico.

Por otro lado, con respecto a la clasificación de los servidores en iterativos y concurrentes, en caso de que solamente exista un proceso o hebra para el procesamiento de las peticiones, estaremos hablando de sistemas *monoproceso* en los que se cumple que el número total de elementos de procesamiento en el sistema es uno ($N_s = 1$). Es evidente que esta condición implica que el número de replicas del servidor es también uno ($M = 1$). En caso contrario, es decir, cuando se trata de sistemas *multiproceso*, la condición que se cumple es $N_s > 1$. Esta diferenciación entre ambos tipos de sistemas es importante y será especialmente considerada en lo sucesivo. Este especial interés se debe a que la ejecución de

los ataques DoS de baja tasa, objeto de estudio de este trabajo, utilizará técnicas diferentes dependiendo de si el servidor es iterativo (monoproceso) o concurrente (multiproceso).

Finalmente, cuando se considera un servidor con estado, toda la información relativa al estado de la sesión con el cliente será mantenida por el módulo de servicio. Será transparente al usuario, desde el punto de vista del modelo, si un servidor es con o sin estado.

En cuanto a la ubicación, cualquier servidor que resida en una única máquina es representable mediante este modelo de forma inmediata mediante la condición $M = 1$. De igual modo, si el sistema servidor está distribuido en diferentes máquinas se cumplirá la condición $M > 1$.

2.3.4 Tiempo entre salidas consecutivas en el servidor

Definimos una *salida* en el servidor como la emisión, por parte de éste, de la respuesta correspondiente a cualquier petición genérica j que llegó a la cola de servicio i del servidor un tiempo $t_q^i + t_s^{i,j}$ antes. Definimos también el *tiempo entre salidas consecutivas*, τ, como la variable aleatoria que representa al tiempo transcurrido entre dos salidas cualesquiera que se producen consecutivamente en el servidor. El estudio del tiempo entre salidas consecutivas será determinante para comprender la vulnerabilidad que permite a un potencial atacante realizar el ataque a baja tasa contra un servidor.

El tiempo entre salidas se puede considerar desde dos puntos de vista diferentes. En primer lugar, se puede hacer una observación de τ desde el punto de vista del servidor, usándose entonces la notación τ_s. En este caso, para todas las salidas que se producen en el servidor, se muestrean los tiempos entre cada par de salidas consecutivas, constituyendo cada muestra una observación de la variable τ_s. En segundo lugar, la percepción de τ se puede realizar también desde el punto de vista de un usuario, el cual emite peticiones y recibe respuestas. En este caso la notación empleada será τ_u. El tiempo entre cada dos respuestas recibidas por el usuario constituye en este caso una muestra de la variable τ_u.

Tal como se ha definido τ_s, las muestras de esta variable se toman siempre a partir de salidas consecutivas en el servidor. Sin embargo, una observación de la variable τ_u no tiene por qué implicar necesariamente dos salidas consecutivas en el servidor, dado que entre las dos salidas correspondientes al usuario que está realizando la observación se pueden producir en el servidor otras destinadas a otros usuarios. Para eliminar esta componente aleatoria, cuando se analice la variable τ_u, se impondrá como condición que las salidas recibidas por el usuario se hayan producido también consecutivamente en el servidor.

Para el estudio del tiempo entre salidas consecutivas se hace también la suposición de que el servidor siempre tiene peticiones que atender. Esto no implica que las colas de servicio tengan todas sus posiciones ocupadas, es decir, que estén saturadas, sino solamente que las colas de servicio no estarán nunca vacías, evitando que los elementos de procesamiento del servidor estén desocupados en algún momento.

Además, se restringe el estudio al caso en el que todas las peticiones procesadas por los elementos de procesamiento son idénticas. Esta restricción implicará que se puede utilizar el modelado del tiempo de servicio dado por la Expresión (2.3).

Estas dos hipótesis bajo las que se hace el estudio del tiempo entre salidas se cumplirán cuando el servidor sea víctima del ataque DoS a baja tasa, tal y como se justificará en los siguientes capítulos, y es por esta razón por la que se restringe el estudio a este caso.

En el estudio del tiempo entre salidas consecutivas será necesario considerar dos escenarios diferentes:

a) Tiempo entre salidas consecutivas cuando el servidor es monoproceso.

b) Tiempo entre salidas consecutivas cuando el servidor es multiproceso.

A continuación se hará un análisis detallado y diferenciado para cada uno de estos dos casos.

Tiempo entre salidas consecutivas en servidores monoproceso

En un servidor monoproceso ($N_s = 1$), cuando se aplican las restricciones anteriormente descritas, el tiempo entre salidas consecutivas desde el punto de vista del servidor, τ_s^{mono}, vendrá determinado exclusivamente por el tiempo de servicio y será independiente del tiempo de cola. Ello es debido a que, al existir siempre en la cola alguna petición pendiente, el servidor está ocupado permanentemente procesando alguna otra petición y, dado que el tiempo que emplea en cada una es el tiempo de servicio, ésta será la única variable que afecta al tiempo entre salidas.

En efecto, dado que el número de elementos de procesamiento que se ejecutan en el servidor es uno, el servidor procesa una petición y, solamente cuando emite una respuesta, comienza a procesar la siguiente. Se concluye que, en caso de servidores monoproceso, el tiempo entre salidas consecutivas no solamente depende exclusivamente del tiempo de servicio, sino que además es igual a dicho tiempo.

Un esquema ilustrativo de este razonamiento se puede observar en la Fig. 2.5, en la que se muestra un diagrama de proceso de ocho peticiones encoladas consecutivamente en la cola de servicio de un servidor monoproceso. Para mayor claridad, se ha supuesto que el valor del tiempo de servicio es constante para todas las peticiones. Dado que existe solamente un elemento de procesamiento, las peticiones se van procesando secuencialmente una tras otra y sin pérdida de tiempo entre cada una de ellas. Se puede observar que los tiempos entre salidas obtenidos, τ_s^{mono}, también son constantes e iguales al tiempo de servicio T_s (en este caso, la distribución de T_s será una constante):

$$\tau_s^{mono} = T_s \tag{2.6}$$

Cuando la distribución del tiempo de servicio no es una constante como en el ejemplo anterior, el tiempo entre salidas será también una variable aleatoria que, consecuentemente, se corresponderá con T_s.

Por otro lado, cuando se consideran dos salidas consecutivas desde el punto de vista de un usuario, τ_u^{mono}, la correspondencia entre τ_u^{mono} y T_s se ve modificada por la variación

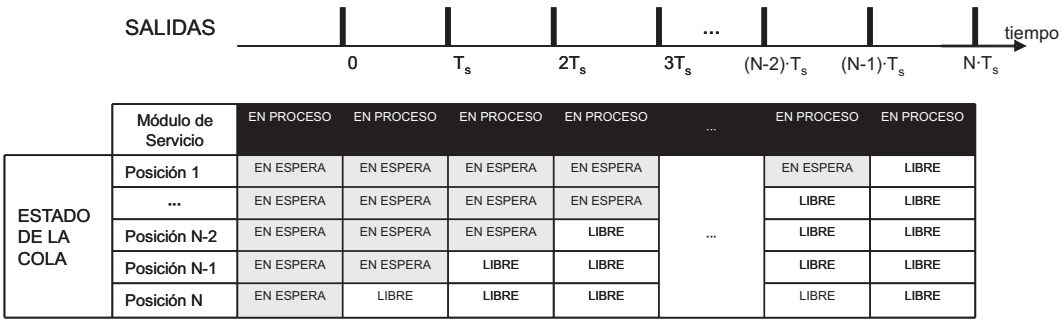


Fig. 2.5: Servidor monoproceso: diagrama de los tiempos entre salidas consecutivas y estado del procesamiento de varias peticiones cuando el tiempo de servicio es constante.

que experimentan las dos salidas consecutivas al viajar desde el servidor hasta el usuario. Esto quiere decir que dicha variación es la acumulada en los retardos correspondientes a ambas salidas.

Ahora bien, es conocido que, en una red de conmutación de paquetes de tipo datagrama como la existente en *Internet*, el camino de ida de una petición del usuario al servidor no tiene por qué ser igual que el de vuelta. Aun así, para una mayor simplicidad se considerará que ambos caminos tienen la misma variabilidad y, por tanto, se aproximará la variabilidad de ambos retardos por la correspondiente al tiempo de ida y vuelta entre el usuario y el servidor, $var[RTT]$. Por extensión, también se considerará que el valor medio del retardo entre usuario y atacante se puede aproximar por $\overline{RTT}/2$. Aunque esta aproximación no es del todo precisa, siempre se podrá diferenciar en las expresiones entre el tiempo de ida y de vuelta. Sin embargo, al aplicar esta aproximación se simplificarán considerablemente las expresiones que se propondrán a lo largo de este trabajo.

Al ser independientes el tiempo de servicio y el de ida y vuelta, y al estar ambos modelados mediante variables con distribución normal, la variable aleatoria resultante que modela los tiempos entre salidas consecutivas tendrá como varianza:

$$var[\tau_u^{mono}] = var[T_s] + var[RTT] \tag{2.7}$$

mientras que la media no se verá influenciada por el tiempo de ida y vuelta, dado que no afecta al espaciado entre salidas, de modo que el valor esperado del tiempo entre salidas desde el punto de vista del usuario será:

$$E[\tau_u^{mono}] = \overline{T_s} \tag{2.8}$$

Finalmente, dado que tanto T_s como RTT se han modelado como variables aleatorias de distribución normal, la variable aleatoria resultante, τ_u^{mono}, será también una variable con distribución normal cuyos parámetros se corresponden con las anteriores expresiones de media y varianza:

$$\tau_u^{mono} = \mathcal{N}\big(\overline{T_s}; var[T_s] + var[RTT]\big) \tag{2.9}$$

Validación experimental del tiempo entre salidas consecutivas en servidores monoproceso

Para tratar de validar estas expresiones en el escenario considerado, se ha implementado en el simulador de redes *Network Simulator 2* (NS2) [Fall and Varadhan, 2007] un servidor monoproceso con un funcionamiento acorde a las características especificadas en el modelo.

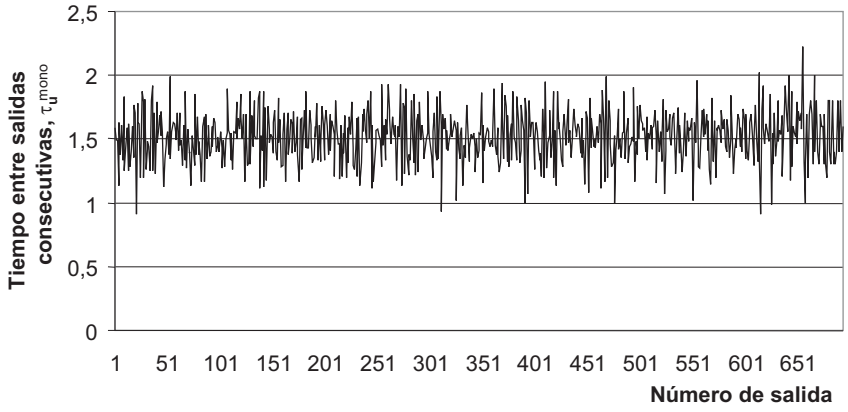
Se han realizado varios conjuntos de experimentos con diferentes valores de configuración de los parámetros del servidor y del tráfico generado por los usuarios para comprobar el comportamiento del tiempo entre salidas consecutivas. En todos estos experimentos se han ajustado inicialmente los parámetros del sistema de modo que el servidor se encuentre siempre con la cola de servicio ocupada.

La Fig. 2.6(a) muestra los tiempos entre salida obtenidos de una simulación en la que se han generado 694 salidas en un servidor monoproceso. El tamaño de la cola de servicio en el servidor, para este ejemplo, es de 40 posiciones y su tiempo de servicio es $T_s = \mathcal{N}(1,5\ s; 0,030)$. El valor del tiempo de ida y vuelta es $RTT = \mathcal{N}(1\ s; 0,010)$. En la simulación se ha hecho llegar tráfico de usuario al servidor a una tasa mayor que la del tiempo de servicio, de modo que se consigue tener siempre ocupada la cola de servicio. El valor medio obtenido de la simulación para el tiempo entre salidas es de 1,520 s, con una varianza de 0,041 s. Como era de esperar, estos resultados corroboran la hipótesis planteada en la Expresión (2.9).

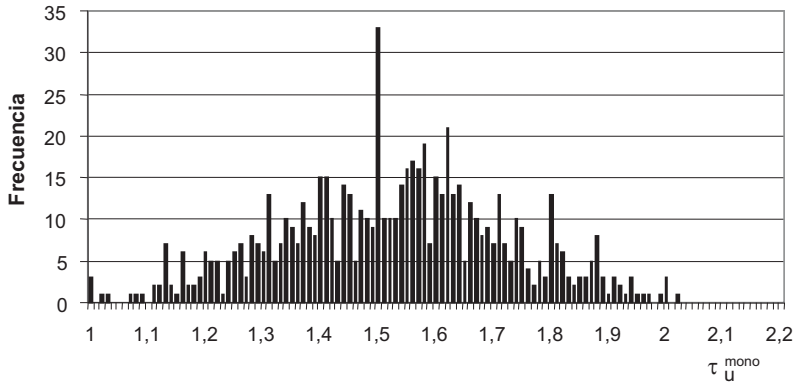
También se ha comprobado, a través del test de similaridad de Kolmogorov-Smirnov [Agostino and Stephens, 1986], que los histogramas obtenidos para los tiempos entre salidas se aproximan correctamente a una distribución normal –ver Fig. 2.6(b)–. En el ejemplo mostrado, el valor obtenido en el test para un nivel de significatividad del 20% (caso muy restrictivo) es de 0,034, lo que indica que la distribución normal es una aproximación adecuada para el histograma obtenido.

Además de los anteriores experimentos, en los que la cola de servicio nunca está vacía, se han realizado otros en los que no se cumple esta condición. El objetivo es comprobar la variación de los resultados cuando cambian las condiciones del experimento.

En este nuevo escenario se obtienen resultados muy similares a los previos. En efecto, la Fig. 2.7(a) muestra los valores de tiempo entre salidas consecutivas obtenidos de una simulación ejemplo que genera 923 salidas. En este caso, se ha ajustado el tráfico de entrada al servidor a la misma tasa media que el de salida. Los parámetros del servidor son los mismos que en el anterior experimento. Se obtiene de esta realización un tiempo medio entre salidas consecutivas de 1,536 s y una varianza de 0,114 s. Se comprueba que, aunque la media se aproxima bastante bien a la del tiempo de servicio, la varianza presenta una mayor desviación que en el caso en que la cola está siempre ocupada, debido a los picos que se producen en el tiempo entre salidas cuando la ocupación de la cola es nula. La curva en tono gris muestra el nivel de ocupación de la cola de servicio, representado porcentualmente, en el eje secundario. Es interesante comprobar cómo cuando el nivel de ocupación de la cola



(a)



(b)

Fig. 2.6: Simulación de funcionamiento de servidor monoproceso con la cola de servicio siempre ocupada: (a) valores de tiempo entre salidas consecutivas, τ_u^{mono}, y (b) histograma de las muestras.

de servicio toma un valor nulo, el tiempo entre salidas consecutivas adopta valores mayores de forma esporádica.

Como es de esperar, en este segundo escenario, los histogramas de los tiempos entre salidas obtenidos –véase el ejemplo de la Fig. 2.7(b)– no cumplen el test de Kolmogorov-Smirnov, incluso para valores de significatividad bajos (0,5%).

La principal conclusión a que lleva esta segunda experimentación es que, como era previsible, las expresiones propuestas para el tiempo entre salidas consecutivas solamente son válidas en situaciones en las que la cola de servicio siempre tiene peticiones esperando a ser servidas y, por tanto, el módulo de servicio nunca deja de procesar peticiones.

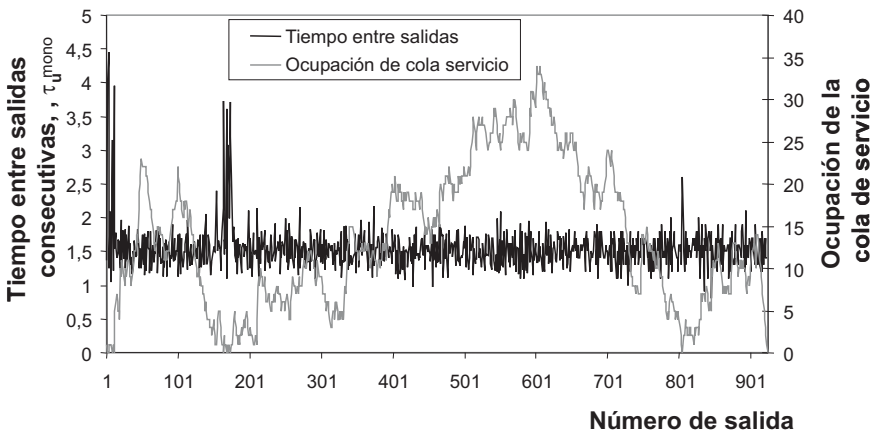
Tiempo entre salidas consecutivas en servidores multiproceso

Las conclusiones obtenidas sobre el comportamiento del tiempo entre salidas consecutivas en servidores monoproceso son directamente extrapolables al comportamiento que se obtiene en las salidas consecutivas procedentes de cada uno de los elementos de procesamiento de un servidor multiproceso. Es decir, aunque los N_s elementos de procesamiento del servidor tienen en común el hecho de extraer peticiones desde determinadas colas de servicio que están compartidas entre ellos, cuando se considera la hipótesis de que dichas colas siempre tienen peticiones, el comportamiento de cada elemento de procesamiento es independiente del resto. Esto es debido a que, bajo estas condiciones, cada elemento de procesamiento estará siempre procesando alguna petición y, por tanto, su comportamiento en la emisión de salidas no dependerá del de otros elementos de procesamiento, sino solamente de la naturaleza de las peticiones que sirva. Suponiendo siempre que todos los elementos de procesamiento en las M máquinas que conforman el servidor son iguales, todas las conclusiones extraídas para el servidor monoproceso se pueden considerar aplicables a cada uno de los elementos de procesamiento de forma independiente.

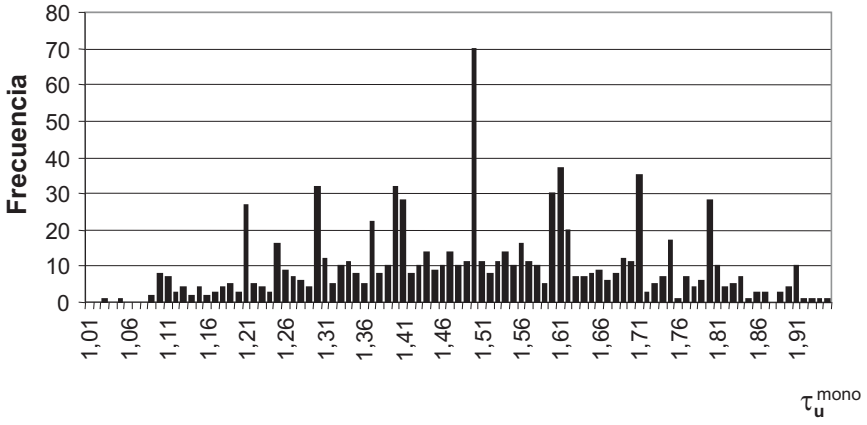
De forma genérica definimos entonces la *probabilidad de ocurrencia de una salida j*, $f_j(t)$, como la función que determina, a partir del instante t_0 en que sucede la salida *j-1* en un elemento de procesamiento, la probabilidad de que la siguiente salida procedente del mismo elemento de procesamiento se produzca en $t_0 + t$ segundos. Dado que el comportamiento en cada elemento de procesamiento es idéntico al correspondiente en el servidor monoproceso, cuando se suponen iguales todas las peticiones enviadas al servidor esta función toma el valor:

$$f_j(t) = \mathcal{N}(\overline{T_s}; var[T_s]) \tag{2.10}$$

El diagrama representado en la Fig. 2.8 muestra, para un sistema servidor con un solo elemento de procesamiento, las diferentes $f_j(t)$ correspondientes a tres salidas consecutivas en dicho elemento de procesamiento. Se puede observar que cada $f_j(t)$ está separada del instante en que se produce la salida *j-1* por un tiempo igual a la media del tiempo de servicio $\overline{T_s}$. En dicho elemento de procesamiento, cada salida se produce en torno a la distribución de $f_j(t)$, de modo que el tiempo entre salidas consecutivas será, en media, el valor correspondiente a la media de $f_j(t)$, es decir:



(a)



(b)

Fig. 2.7: Simulación del escenario en el que la cola de servicio del servidor monoproceso puede estar vacía: (a) valores de tiempo entre salidas, τ_u^{mono}, (curva negra) y ocupación de la cola de servicio (curva gris) y (b) histograma de las muestras obtenidas.

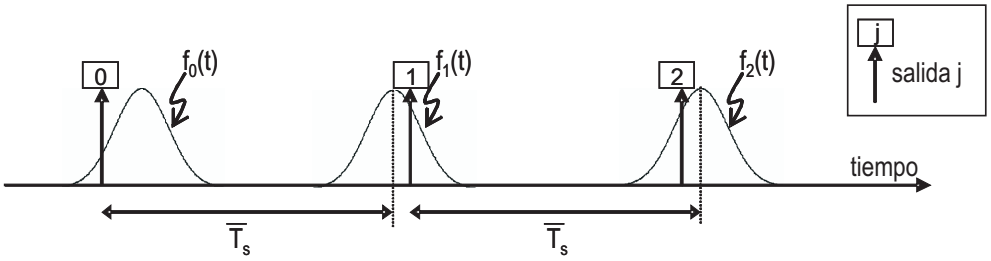


Fig. 2.8: Evolución temporal de las salidas y sus funciones de probabilidad de ocurrencia en un sistema servidor monoproceso ($N_s = 1$), donde $f_j(t)$ se supone de distribución normal.

$$\overline{\tau_s} = \overline{T_s} \tag{2.11}$$

Además, la variabilidad del instante en que se produce la salida en torno a dicha media corresponde al valor dado por la varianza del tiempo de servicio de las peticiones, $var[T_s]$. Se puede comprobar que este razonamiento es coherente con la Expresión (2.6). Por tanto, se obtiene como conclusión que, en el caso del servidor monoproceso, el tiempo entre salidas, τ_s, se corresponde con la función de probabilidad de ocurrencia de una salida, $f_j(t)$.

Por otro lado, cuando se considera un servidor multiproceso ($N_s > 1$), la correspondencia entre el tiempo entre salidas consecutivas y la función de probabilidad de ocurrencia de una salida no se cumplirá, tal y como se muestra a continuación.

En principio, es de esperar que la curva obtenida en la Fig. 2.6(b) para un servidor monoproceso quede modificada cuando se considera un servidor multiproceso, debido a que los elementos de procesamiento van generando salidas de forma independiente. Cuando existen N_s elementos de procesamiento en el servidor, el tiempo entre salidas consecutivas dependerá no solamente del tiempo de servicio aplicado a cada petición, sino también del número de elementos de procesamiento que están procesando peticiones en paralelo y de los instantes en que las diferentes peticiones entran a los módulos de servicio. Un ejemplo para ilustrar este razonamiento es el siguiente.

Si N_s peticiones idénticas entran en el módulo de servicio simultáneamente en $t = t_0$, es esperable conseguir que, en un intervalo de tiempo dado por la distribución de la variable aleatoria T_s, se produzcan N_s salidas. Dado que se asume una distribución normal para T_s, este intervalo de tiempo estará centrado en un valor $\overline{T_s}$ y se distribuye en torno a él con una varianza, $var[T_s]$. Cada vez que se produce una salida de los N_s diferentes elementos de procesamiento, se toma otra petición de la cola de servicio. Estas peticiones tardarán también en ser servidas un tiempo determinado por T_s. Ahora bien, los instantes en los que han entrado al módulo de servicio no son los mismos, por lo que la salida se producirá en un entorno de tiempo afectado por la variabilidad propia de T_s en este ciclo de procesamiento y también por la variabilidad de T_s en el anterior ciclo. Es de esperar que, pasados unos cuantos ciclos, los tiempos entre las diferentes salidas consecutivas no presenten una distribución como la correspondiente a T_s.

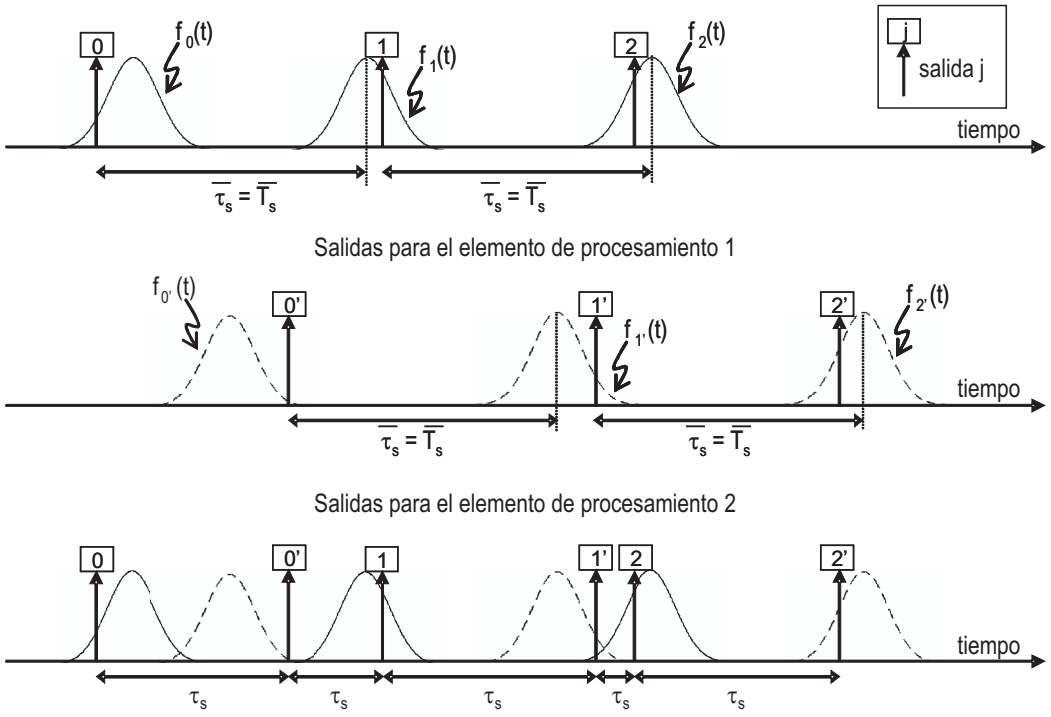


Fig. 2.9: Evolución temporal de las salidas y sus correspondientes funciones de probabilidad de ocurrencia en un servidor multiproceso ($N_s = 2$). Representación de la evolución temporal independiente de los elementos de procesamiento 1 y 2, y de la conjunta de ambos elementos de procesamiento. La probabilidad de ocurrencia de una salida se ha supuesto con distribución normal.

Un ejemplo más genérico está representado en la Figura 2.9. En ella se muestran por separado las salidas correspondientes a dos elementos de procesamiento diferentes[2] y sus probabilidades de ocurrencia. Además, se ilustra la distribución temporal conjunta de las salidas correspondientes a ambos elementos de procesamiento. Se puede observar que, mientras para cada uno de los elementos de procesamiento, considerados de forma independiente, se cumple que la distribución de τ_s coincide con la de T_s, no ocurre lo mismo cuando se considera la evolución conjunta de ambos elementos de procesamiento.

Aun así, aunque la distribución de los tiempos entre salidas consecutivas en un servidor concurrente sea diferente de T_s, dado que, en media, cada elemento de procesamiento emite una salida independientemente del resto cada $\overline{T_s}$, la media del tiempo entre salidas consecutivas cuando existen N_s elementos de procesamiento será equivalente al de un servi-

[2] En la Figura 2.9 se ha utilizado una notación numérica para las salidas. Las correspondientes al elemento de procesamiento 2 se numeran con prima (ej. $0'$), a diferencia de las correspondientes al elemento de procesamiento 1.

dor monoproceso en el que el tiempo de servicio se divide por el número de elementos de procesamiento:

$$\overline{\tau_s} = \frac{\overline{T_s}}{N_s} \tag{2.12}$$

Esta expresión indica que el estadístico de primer orden de la distribución del tiempo entre salidas consecutivas para un servidor multiproceso está relacionado con el tiempo de servicio y con el número de elementos de procesamiento. Nótese que el tiempo entre salidas se reduce conforme aumenta el número de elementos de procesamiento y/o disminuye $\overline{T_s}$. Además, es importante señalar que la Expresión (2.12) se puede también utilizar para el caso de servidores monoproceso, simplemente con asignar el valor $N_s = 1$.

Validación experimental del tiempo entre salidas consecutivas en servidores multiproceso

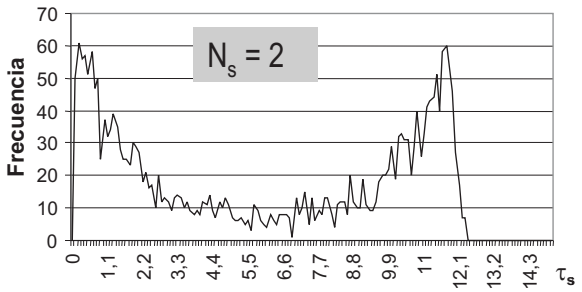
Para comprobar las hipótesis realizadas sobre el comportamiento del tiempo entre salidas consecutivas en un servidor concurrente, se ha implementado un servidor multiproceso en *Network Simulator 2* (NS2) [Fall and Varadhan, 2007] acorde con el modelo aquí estudiado.

Se han realizado varios conjuntos de experimentos con diferentes valores de configuración para el servidor y para el tráfico que se le ofrece por parte de los usuarios. La Fig. 2.10 muestra una comparativa extraída de uno de los experimentos realizados. En ella se ilustran cuatro escenarios en los que el servidor ha sido configurado con un tiempo de servicio $T_s = \mathcal{N}(12\ s; 0, 3)$. El número de elementos de procesamiento se ha variado para comprobar el comportamiento de los tiempos entre salidas consecutivas.

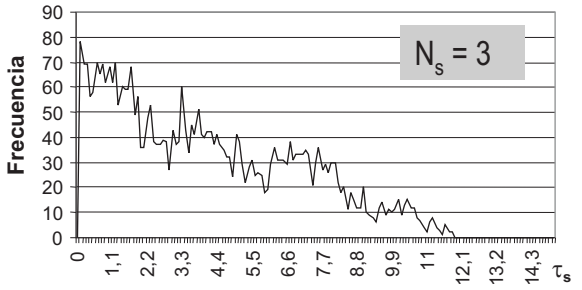
En los diferentes experimentos realizados se ha podido comprobar que se cumple la Expresión (2.12), en la que se establece la condición para el valor esperado de τ_s. Se puede corroborar, además, que la distribución de valores en la Fig. 2.10(d) está mucho más concentrada en torno a valores pequeños que en la Fig. 2.10(c). Ello es debido a que el número de elementos de procesamiento N_s de la primera es mayor.

Por otra parte, se observa que la distribución de valores con respecto a la media sí varía significativamente con la obtenida para los servidores monoproceso. En la Fig. 2.10 se comprueba que, incluso cuando las peticiones son idénticas, la función de probabilidad de los tiempos entre salidas consecutivas no se corresponderá con una distribución normal como en el caso de los servidores monoproceso. Además, cuando existe un número suficiente de elementos de procesamiento, la distribución tiende a seguir una curva exponencial decreciente, concentrando la mayor parte de sus puntos en valores pequeños de τ_s, tal y como se espera según la Expresión (2.12).

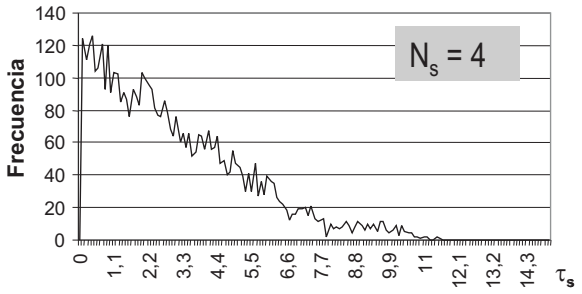
También es interesante comprobar el comportamiento de los tiempos entre salidas consecutivas cuando se produce una variación en la varianza del tiempo de servicio. Con esta finalidad, se han realizado varios experimentos en los cuales, para diferentes configuraciones del servidor, se ha modificado la varianza del tiempo de servicio. La Fig. 2.11 muestra los resultados de uno de los experimentos. Se pueden observar cuatro histogramas de los



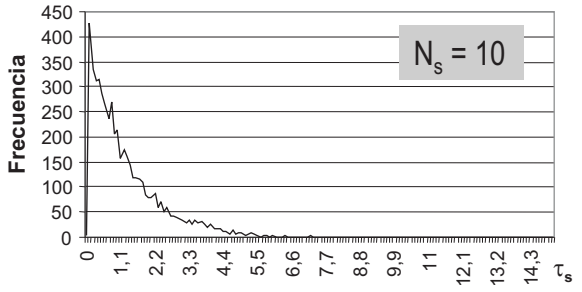
(a)



(b)

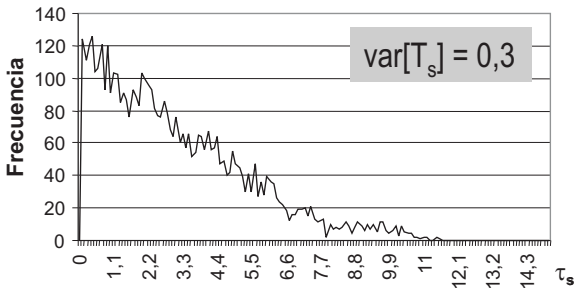


(c)

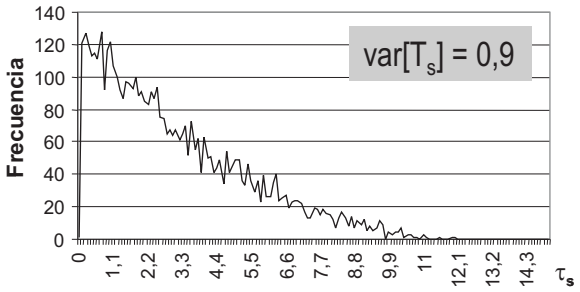


(d)

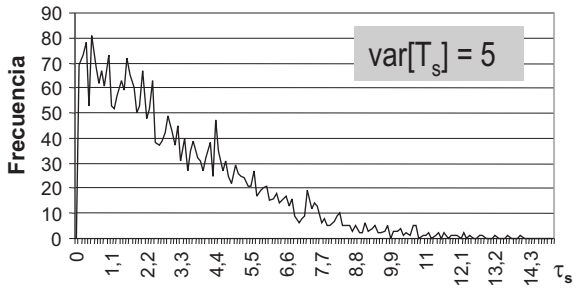
Fig. 2.10: Histograma de los tiempos entre salidas estimados en un sistema multiproceso con $T_s = \mathcal{N}(12\ s; 0, 3)$, para diferentes valores de N_s.



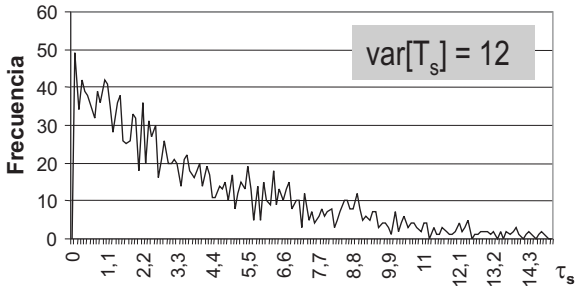
(a)



(b)



(c)



(d)

Fig. 2.11: Histograma de los tiempos entre salidas estimados en un sistema multiproceso con $\overline{T_s} = 12$ y $N_s = 4$, para diferentes valores de la varianza, $var[T_s]$.

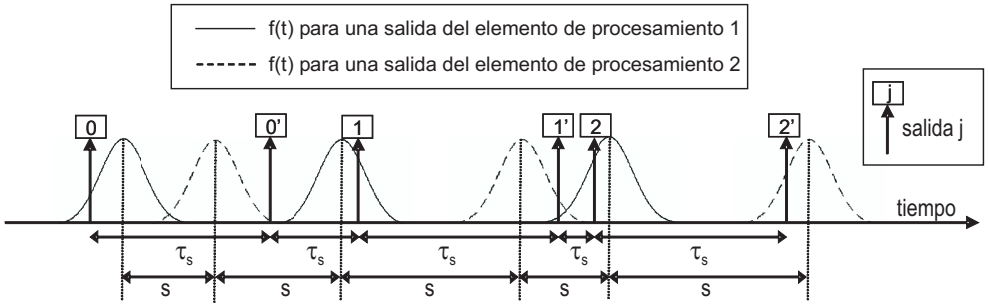


Fig. 2.12: Diagrama de salidas y sus $f_j(t)$ asociadas correspondientes a dos elementos de procesamiento diferentes y representación de los valores instantáneos de s y τ_s.

tiempos entre salidas consecutivas para diferentes valores de la varianza, $var[T_s]$. Se concluye de los experimentos que, comparando los valores obtenidos, las variaciones en la varianza del tiempo de servicio no producen cambios significativos en la distribución del tiempo entre salidas. El motivo puede ser que, cuando existe un número suficiente de elementos de procesamiento ejecutándose de forma independiente, las variaciones de los tiempos de servicio de unos elementos se ven compensadas por las de otros, haciendo la distribución del tiempo entre salidas no dependiente de la variación del tiempo de servicio.

2.3.5 Distancia de superposición

Dada una salida concreta en el servidor, definimos la *distancia de superposición*, s, como una variable aleatoria que representa el tiempo que separa los valores medios de las funciones de probabilidad de ocurrencia de dos salidas consecutivas. Para la función de probabilidad de dicha variable aleatoria se utilizará la notación $g(s)$.

Es necesario indicar que, aunque esta definición es muy parecida a la establecida para el tiempo entre salidas consecutivas, se trata de conceptos diferentes. El hecho significativo que los diferencia consiste en que el tiempo entre salidas consecutivas es una variable aleatoria cuya distribución se obtiene de muestrear observaciones del tiempo transcurrido entre las salidas que se producen consecutivamente, mientras que cada muestra que compone la distribución de s no es la distancia temporal entre las dos salidas, sino el tiempo entre el valor medio de $f_j(t)$ para una salida y dicho valor para la siguiente. En la Fig. 2.12 se pueden observar, para un sistema servidor con dos elementos de procesamiento, las diferencias entre las muestras de s y τ_s. Hay que notar que las muestras de s se extraen de la diferencia entre los valores medios de las funciones $f_j(t)$ consecutivas, mientras que las de τ_s se toman de la diferencia entre los instantes en que se producen realmente las salidas.

En el caso hipotético de que las salidas siempre se produjeran en el valor medio de la distribución $f_j(t)$, las observaciones de s y τ_s coincidirían. Sin embargo, cuando esto no se cumple, existirá una diferencia entre las observaciones de la distancia de superposición y el tiempo entre salidas. Esta diferencia es debida a una desviación en la estimación del instante en que se produce la salida siguiente. En efecto, la estimación de la distancia

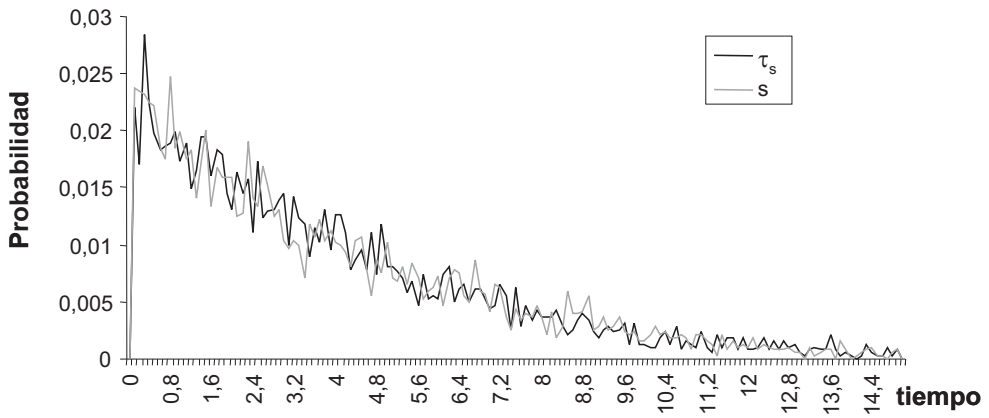


Fig. 2.13: Funciones de probabilidad de las variables aleatorias s y τ_s, para una configuración dada del servidor.

de superposición se realizará en base a la Expresión (2.12), por lo que la desviación en la estimación se puede deber a dos factores:

- Desviaciones en la estimación del valor esperado del tiempo de servicio $\overline{T_s}$.
- Desviaciones provocadas por la naturaleza aleatoria de T_s. Se producirán variaciones en torno al valor esperado de la variable T_s. Éstos vendrán determinados por la varianza de su distribución de probabilidad, $var[T_s]$. En el caso en que se considera la observación del tiempo entre salidas desde el punto de vista de un usuario, también hay que considerar las variaciones introducidas por la variabilidad del tiempo de ida y vuelta, $var[RTT]$, en los mismos términos en que se ha discutido para la Expresión (2.9).

Por otro lado, si bien existe una diferencia entre los valores instantáneos que dan lugar a ambas variables aleatorias, es interesante comprobar que, sin embargo, las distribuciones obtenidas para ambas variables aleatorias coinciden. Este hecho se ha constatado mediante un conjunto de experimentos de simulación realizados para diferentes configuraciones para el servidor, en los que se ha variado el valor medio y la varianza del tiempo de servicio y el número de elementos de procesamiento. En todas las configuraciones se ha obtenido una diferencia, en términos del error cuadrático medio, despreciable entre ambas distribuciones. En la Fig. 2.13 se representa el valor de la distribución de τ_s y de s para una de las configuraciones comprobadas, resultando en este caso el error cuadrático medio igual a $E = 3,5 \cdot 10^{-6}$.

2.3.6 Sistemas servidores con superposición y sin superposición

Basándonos en la definición de la función de probabilidad de ocurrencia de una salida $f_j(t)$, se puede inferir el concepto de superposición aplicado a dos salidas consecutivas en el servidor.

Se dice que *dos salidas consecutivas están superpuestas* cuando sus distribuciones $f_j(t)$ se superponen en el tiempo un valor ε.

La desigualdad de Tchebychev [Song, 2004] establece, para una variable aleatoria x, que la proporción de su distribución comprendida en el intervalo $\mu \pm k\sigma$, $k \in \Re$, es la siguiente:

$$P(\mu - k\sigma \leq x \leq \mu + k\sigma) \geq 1 - 1/k^2 \tag{2.13}$$

donde μ es la media de la distribución considerada y σ su desviación estándar.

Dado que la distancia entre las distribuciones $f_j(t)$ de dos salidas consecutivas vendrá dada por la distancia de superposición s, para dos salidas concretas cuya distancia de superposición sea s_0, la condición para que exista superposición será:

$$s_0 \leq 2 \cdot k \cdot \sigma \Longrightarrow s_0 \leq 2 \cdot k \cdot \sqrt{var[T_s]} \tag{2.14}$$

donde k vendrá dado por la Expresión (2.13) según el valor ε que se establezca como ocurrencia de superposición, de modo que:

$$\varepsilon \geq \frac{1}{k^2} \quad \Longrightarrow \quad k \geq \frac{1}{\sqrt{\varepsilon}}$$

Y la condición de superposición se convierte en:

$$s_0 \leq 2 \cdot \sqrt{\frac{var[T_s]}{\varepsilon}} \tag{2.15}$$

Aunque esta condición es general, cuando se consideran distribuciones normales, como sucede con $f_j(t)$ cuando las peticiones son idénticas, el 99,7% de la distribución normal está comprendido en el intervalo $\mu \pm 3\sigma$ [Peña, 2001]. En este caso, la acotación de Tchebychev aporta un intervalo mayor que el precisado y, por consiguiente, la distancia s_0 máxima para que exista superposición sería menor que la calculada con la Expresión (2.15).

En la Fig. 2.14(a) se representan las funciones de probabilidad de ocurrencia de dos salidas consecutivas con superposición para un umbral ε genérico. Además, la Fig. 2.14(b) representa también las funciones de probabilidad de ocurrencia de dos salidas consecutivas cuando entre ellas no se produce superposición.

Por otro lado, definiremos también el concepto de superposición aplicado a un sistema servidor. Se dirá que un *servidor presenta superposición* cuando al menos dos salidas consecutivas de dicho servidor presentan superposición. En caso contrario, se dirá que el sistema no presenta superposición.

Es necesario establecer la relación que existe entre el concepto de superposición en un servidor y su nivel de concurrencia, es decir, determinar los casos posibles en los que un servidor monoproceso o multiproceso pueden o no presentar superposicion. Un servidor multiproceso, aún teniendo un número de elementos de procesamiento muy bajo, siempre

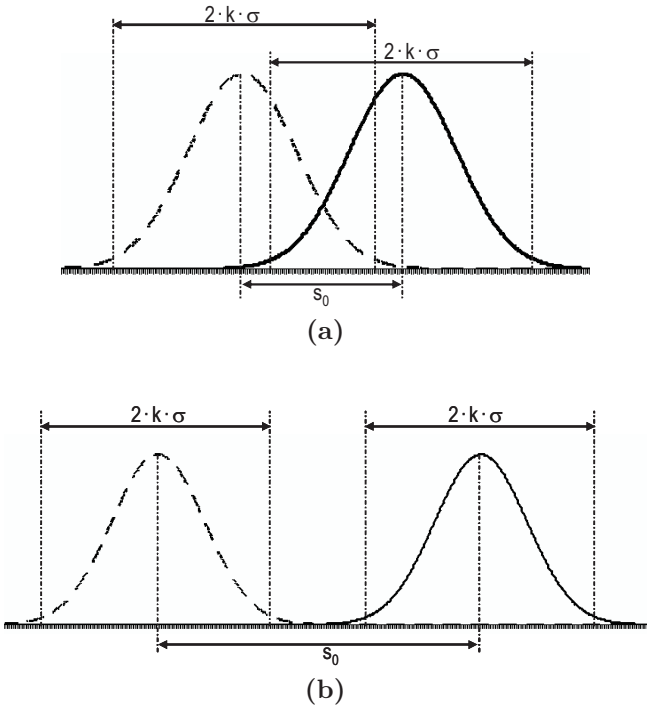


Fig. 2.14: Diagrama con dos salidas consecutivas: (a) con superposición y (b) sin superposición.

será un sistema con superposición, dado que su generación de salidas presenta siempre muestras con valores pequeños de distancia de superposición, tal y como se puede comprobar en la distribución de los valores de la variable aleatoria s (Fig. 2.10). Ahora bien, un servidor monoproceso podrá ser un sistema sin superposición siempre que se cumpla que la media del tiempo de servicio resulte suficientemente grande para que no haya solapamiento entre las $f_j(t)$ de salidas consecutivas. En caso contrario, el servidor monoproceso será un sistema con superposición. En resumen, un servidor monoproceso podrá presentar o no superposición, mientras que uno multiproceso siempre será un sistema con superposición.

2.4 Modelado de los usuarios legítimos del sistema

Los *usuarios legítimos* representan en el escenario de estudio a aquellos elementos, máquinas o entidades que precisan del servidor la ejecución de determinadas tareas o el envío de ciertos datos. También están autorizados a solicitar estos servicios y, además, envían las peticiones al servidor siguiendo un patrón de normalidad estadística, con corrección sintáctica y semántica.

Las peticiones de los usuarios entrantes en el sistema también se consideran un proceso aleatorio. Tal y como está recomendado en numerosos estudios de teletráfico [Martin, 1993],

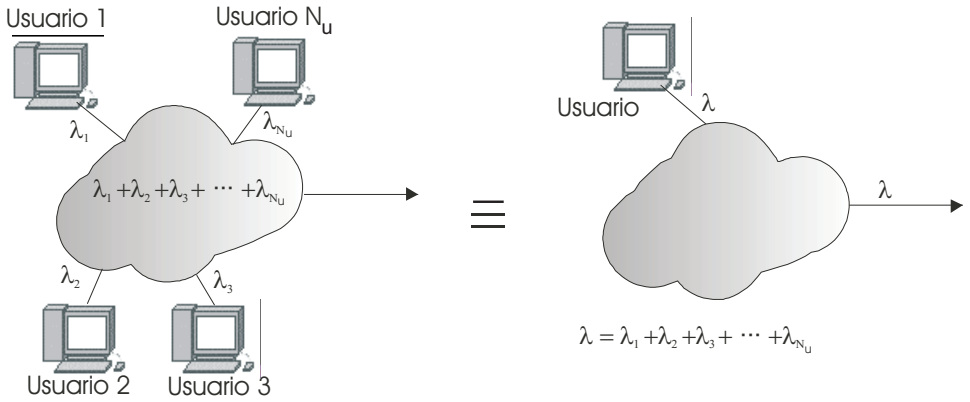


Fig. 2.15: Escenario con tráfico agregado procedente de N_u usuarios y su escenario equivalente con un único usuario.

dicho tráfico se modela como un proceso de Poisson. Para este tipo de procesos, la distribución del tiempo entre llegadas al servidor, T_a, corresponde con una función de probabilidad exponencial [Martin, 1993]:

$$P(T_a = t) = \lambda \cdot e^{-\lambda t} \tag{2.16}$$

donde λ es la tasa de llegada de paquetes procedentes de los usuarios.

La agregación de tráfico procedente de diferentes usuarios cuya distribución del tiempo entre llegadas es exponencial, con tasas λ_i, resulta en un tráfico cuya distribución del tiempo entre llegadas es también exponencial. Además, la tasa que corresponde a dicho tráfico agregado es la suma de las tasas de los tráficos que componen el total [Exp, 2006]:

$$\lambda = \sum_{i=1}^{N_u} \lambda_i \tag{2.17}$$

Como en el escenario presentado en la Fig. 2.1 los tráficos procedentes de los diferentes usuarios se agregan al llegar al servidor, se puede reducir este caso a uno en el cual exista solamente un usuario (ver Fig. 2.15), el cual generará todo el tráfico agregado (procesos de Poisson) procedente de N_u usuarios, con la condición de que la tasa de envío de mensajes procedentes de dicho usuario cumpla la condición mostrada en la Expresión (2.17).

En condiciones estacionarias es evidente que, para evitar la saturación del sistema, la condición que se debe cumplir con respecto a la tasa de llegadas de peticiones de usuario es que ésta sea menor que la tasa de servicio en el servidor:

$$\lambda \leq \frac{N_s}{\overline{T_s}} \tag{2.18}$$

2.5 Conclusiones del capítulo

En este capítulo se ha realizado el modelado del escenario de ataque planteado en este trabajo. Para ello, en primer lugar, se han definido los diferentes componentes que integran dicho escenario y sus interconexiones e interacciones mutuas.

Se ha propuesto un modelo flexible para el servidor que permite adaptarse a las diferentes características y arquitecturas que es posible implementar. De este modelo se han extraído determinadas características y conceptos que serán básicos para el desarrollo de este trabajo. Entre ellos destaca el tiempo entre salidas consecutivas, la distancia de superposición y el concepto de superposición.

Para cada una de las discusiones planteadas se ha llevado a cabo una evaluación experimental en entornos de simulación con el fin de comprobar la validez de las afirmaciones realizadas.

Finalmente, también se ha propuesto un modelo para el tráfico procedente de usuarios legítimos del sistema, basándose en estándares de tráfico ampliamente aceptados.

Este trabajo de modelado realizado en este capítulo constituye la base de estudio sobre la cual se desarrollarán los conceptos, estrategias y metodologías relacionados con la ejecución de los ataques DoS a baja tasa contra servidores, y que serán presentados en los capítulos siguientes.

Capítulo 3

El ataque DoS a baja tasa contra servidores

3.1 Introducción

Tal y como se ha planteado en el Capítulo 1, el objetivo de los ataques de denegación de servicio es limitar total o parcialmente el acceso de los usuarios legítimos de un servicio a la funcionalidad que éste ofrece. Esto se lleva a cabo, usualmente, mediante dos técnicas posibles: bien mediante la inundación o uso masivo de los recursos de la víctima, quedando ésta sin recursos suficientes para atender a los usuarios legítimos, o bien explotando una vulnerabilidad que afecte al comportamiento de la víctima de tal modo que el servicio que presta quede de algún modo inaccesible para los usuarios legítimos.

Una vez establecidos en el Capítulo 2 el escenario de trabajo en que se desarrolla el ataque, el modelo del servidor y el comportamiento de los usuarios del sistema, el objetivo de este tercer capítulo es mostrar los fundamentos acerca de cómo pueden sufrir los servidores un ataque de denegación de servicio en base a la recepción de una baja tasa de tráfico de mensajes. Para ello, se van a especificar las líneas maestras que articulan dicho tipo de ataques, profundizándose en los capítulos posteriores en los detalles de implementación de los mismos, diferenciando si el servidor objetivo es monoproceso o multiproceso.

También se abordará a lo largo de este capítulo una evaluación del rendimiento que este tipo de ataques puede alcanzar cuando se utiliza una configuración concreta tanto en el servidor como en los parámetros del ataque. Para ello, se desarrollarán unos modelos matemáticos que, al respecto del comportamiento a analizar, permitirán la mencionada evaluación.

La estructura del capítulo es la siguiente. En el Apartado 3.2 se especifica el modo en que se realizará el ataque de baja tasa, presentando la estrategia general seguida por el atacante y mostrando los detalles relativos a su ejecución, así como una comparación con otros ataques de baja tasa. En el Apartado 3.3 se propone una serie de indicadores que permitirán

la evaluación del rendimiento del ataque a baja tasa planteado. En los Apartados 3.4, 3.5 y 3.6 se presentan unos modelos matemáticos desarrollados para evaluar los indicadores del rendimiento del ataque previamente establecidos. Finalmente, el Apartado 3.7 presenta las conclusiones de este capítulo.

3.2 Especificación del ataque DoS a baja tasa contra servidores

En este apartado se realiza la especificación del método a seguir para realizar el ataque DoS a baja tasa contra servidores. Para ello, en primer lugar se aborda una descripción de la estrategia general seguida para, posteriormente, detallar el diseño del ataque. Finalmente, se relacionan las características que el ataque presenta y se hace una comparación de este tipo de ataques con otros ataques de baja tasa.

3.2.1 Estrategia general para la realización del ataque

La estrategia principal que el ataque DoS a baja tasa contra servidores seguirá es la de evitar la disponibilidad del servidor generando en el mismo un estado de saturación. Esto implica que el ataque debe conseguir que en el servidor no quede ninguna posición libre en las diferentes colas de servicio, de modo que, en esta situación, cualquier petición que llegue procedente de algún usuario legítimo del servicio sea descartado. Obviamente, la percepción del usuario en esta situación será que el servicio no se encuentra disponible.

Hay que tener en cuenta que, en la situación que se ha planteado, mientras que el usuario experimenta una denegación de servicio real, es decir, no es capaz de obtener el servicio deseado, el servidor propiamente dicho mantiene su ejecución de modo normal, es decir, no dejará de funcionar procesando peticiones tal y como hace normalmente. La única peculiaridad será que todas las peticiones que está sirviendo pertenecen a un único usuario: el atacante.

El atacante, por tanto, deberá tratar de conseguir que el servidor solamente sirva sus peticiones, lo que es equivalente a que las colas de servicio estén llenas (estado de saturación) con sus peticiones. Supongamos que el atacante consigue, mediante cierto método que no especificaremos en este punto, que el servidor llegue a un estado de saturación. Partiendo de esta situación, para que el atacante pueda mantener al servidor en este estado a lo largo del tiempo, y que además solamente sirva sus peticiones, deberá conseguir introducir una petición en el mismo antes de que ningún otro usuario lo haga, siempre que en alguna cola de servicio aparezca alguna posición libre. Al acto de introducir una petición en alguna cola del servidor cuando se genera una salida se le denominará *captura*. Por tanto, el objetivo del atacante será realizar el máximo número posible de capturas.

En las diferentes colas de servicio aparecerá una posición libre siempre que algún elemento de procesamiento del módulo de servicio termine de procesar una petición y, al quedar libre, extraiga una nueva petición de la correspondiente cola de servicio. En definitiva,

los instantes en que aparece una posición libre en el servidor corresponderán con aquellos en los que se produce una salida del mismo.

El atacante, a la hora de diseñar el ataque, podría inundar al servidor con una tasa suficientemente alta de mensajes de ataque de modo que siempre que aparezca una posición libre en el mismo la probabilidad de realizar la captura sea alta. Este modo de funcionamiento es el que tradicionalmente aplican los ataques DoS de inundación a alta tasa. Ahora bien, aunque es cierto que esta estrategia sería exitosa para el fin planteado, para el atacante es deseable poder utilizar una tasa baja de tráfico en lugar de una tasa elevada, con el fin de evitar la detección del ataque por parte de potenciales sistemas de seguridad presentes en el entorno de la víctima. Además, esto permitiría la ejecución del ataque utilizando muchos menos recursos (ancho de banda, número de máquinas comprometidas, ...) que en un ataque de inundación de tasa alta, lo que facilita la ejecución al potencial atacante.

Con el fin de reducir su tasa de tráfico, el atacante tratará de explotar alguna vulnerabilidad que le permita, de algún modo, conocer o predecir los instantes en los que se producirá una salida y, por consiguiente, se habilite una posición libre en alguna cola de servicio. Esta estrategia será la que se utilice para el diseño del ataque DoS a baja tasa.

La clave del ataque a baja tasa, por tanto, reside en la potencial capacidad del atacante para predecir los instantes en los que se generarán las salidas. Una vez realizada esta predicción, solamente haría falta enviar una petición de modo que llegue al servidor inmediatamente después de que la posición haya quedado libre en la cola de servicio correspondiente, y antes de cualquier otra procedente de un usuario legítimo. De esta forma, al tiempo que se evitaría el uso de una tasa elevada de tráfico contra el servidor, el atacante se aseguraría la captura de la mayoría de las posiciones disponibles en las colas del mismo, generando así la denegación de servicio.

Es necesario destacar un aspecto importante en todo este proceso. Los mensajes que el atacante envía al servidor son mensajes correctamente construidos a nivel de protocolo. Esto quiere decir que el servidor seguirá procesándolos correctamente durante el tiempo del ataque. Sin embargo, el atacante tratará, como es evidente, de que dichos mensajes utilicen el máximo tiempo de servicio posible en el servidor, con el fin de que éste se encuentre ocupado más tiempo en el procesado y la generación de salidas sea menos frecuente. Este tipo de estrategias ya ha sido empleado en algunos ataques de denegación de servicio, como el ataque *Naptha* [SANS Institute, 2001].

Evidentemente, en el proceso de implementación de esta estrategia de ataque surgen numerosas dificultades para el atacante. La problemática que aparece y las soluciones de diseño asociadas que el atacante podría adoptar serán analizadas a continuación.

Para los aspectos generales del ataque desarrollados en este capítulo se asumirán conocidas tanto la vulnerabilidad que permite realizar la predicción de los instantes de generación de las salidas, como las técnicas para realizar dicha predicción. Además, también se supondrá una situación inicial de saturación en el servidor. Los capítulos posteriores ilustrarán estos aspectos según el tipo de servidor elegido como víctima del ataque.

3.2.2 El periodo básico de ataque

Como ya se ha comentado, la estrategia general para la realización del ataque DoS a baja tasa se fundamenta en la predicción de los instantes en que se producen las salidas del servidor y en el envío de mensajes de ataque de modo que éstos lleguen al servidor justamente después de que se produzcan dichas salidas, con el fin de capturar las correspondientes nuevas posiciones habilitadas en las colas de servicio.

Inicialmente, para cada salida que se produzca sería necesario enviar un solo mensaje hacia el servidor, que es el que ocupará la posición liberada en la correspondiente cola de servicio. Ahora bien, no es sencillo para el atacante conseguir que dicho mensaje llegue al servidor justamente después del momento en que se produce la salida. Esto se debe fundamentalmente a que existen desviaciones entre el instante predicho para la salida y en el que realmente ésta sucede. Estas desviaciones, como ya se ha comentado en el Capítulo 2, son debidas a la propia estimación del instante de la salida y a la naturaleza aleatoria de RTT entre el servidor y el atacante.

De este modo, cuando el atacante decide enviar el mensaje de ataque hacia el servidor, debido a las desviaciones anteriormente mencionadas, pueden ocurrir tres situaciones diferentes:

(a) *El mensaje llega antes que la salida*:

En este caso, y suponiendo que el servidor se encuentra en estado de saturación antes de que se produzca la salida, el mensaje encontrará las colas de servicio llenas y, por tanto, será descartado. Posteriormente se liberará la posición correspondiente a la salida y ésta podrá ser ocupada por una petición de cualquier otro usuario. Consecuentemente, en esta situación se produce un fallo en la captura.

(b) *El mensaje de ataque llega después de la salida y antes que ningún otro mensaje procedente de un usuario legítimo*:

Este es el caso ideal para el atacante, ya que así capturará la posición libre de la cola de servicio y, por tanto, habrá conseguido el objetivo propuesto.

(c) *El mensaje de ataque llega después de la salida pero también después de la llegada de una petición procedente de algún otro usuario legítimo*:

En este caso, y también suponiendo el estado de saturación en el servidor antes de que ocurra la salida, el mensaje del usuario ocupa la posición en la cola de servicio y, por tanto, al llegar el mensaje de ataque al servidor, éste será descartado. Esto implica un fallo en la captura de la posición por parte del atacante.

El atacante debe tratar de maximizar la probabilidad de que se produzca la situación (b), de modo que el envío de su mensaje de ataque sea efectivo y pueda capturar la posición liberada en la correspondiente cola de servicio. Para ello, se utilizará más de una petición de ataque por cada estimación de salida. De este modo, para cada salida se ejecuta una onda de ataque, denominada *periodo básico de ataque*. Esta onda tendrá una forma de tipo ON/OFF, con periodos de actividad de emisión de mensajes de ataque (ON) y con periodos

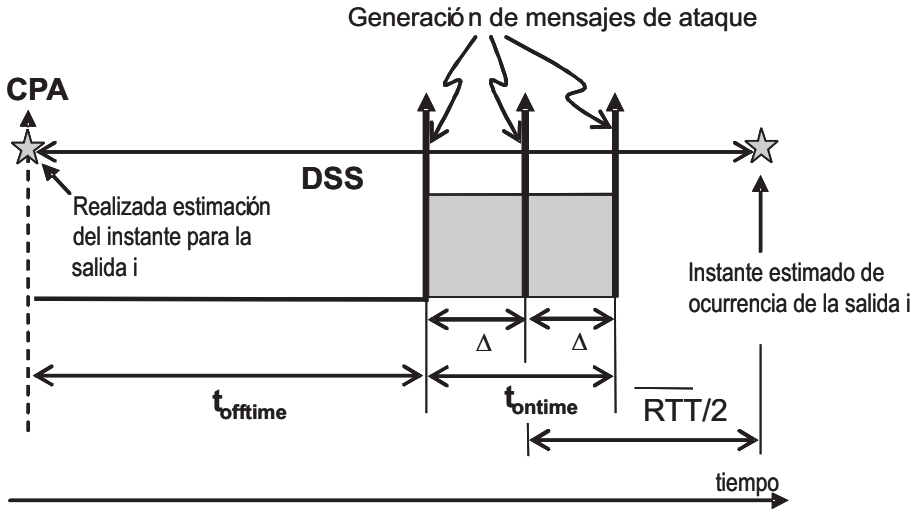


Fig. 3.1: Periodo básico de ataque: forma de onda y parámetros.

de inactividad (OFF) en los que no se emite nada. Las siguientes características definen al periodo básico de ataque, representado en la Fig. 3.1:

- *Comienzo del periodo de ataque* (*CPA*):

  Es el instante en el que se inicia el periodo básico de ataque. Cada periodo comenzará, por definición, una vez que se ha realizado la estimación del instante en que se va a producir la siguiente salida en el servidor.

- *Distancia a la siguiente salida* (*DSS*):

  Es el tiempo que transcurre desde que se ha iniciado el periodo de ataque (*CPA*) hasta el instante estimado en que se producirá la siguiente salida. El valor de esta estimación dependerá de la técnica de predicción usada, que se presentará en capítulos posteriores.

- *Tiempo* ontime (t_{ontime}):

  Es el intervalo de actividad durante el cual se envían una serie de mensajes de ataque con la intención de capturar la posición liberada en la cola de servicio. El objetivo del atacante será que dichas peticiones lleguen en torno al instante en que se prevé la salida. Por ello, dado que los mensajes emplean un tiempo $\overline{RTT}/2$ para viajar entre el atacante y el servidor[1], la emisión de los mensajes de ataque se realizará considerando dicho tiempo de modo que, en su llegada al servidor, t_{ontime} se encuentre centrado en torno al instante estimado de ocurrencia de la salida.

  El valor de t_{ontime} deberá considerar la magnitud de la desviación que el atacante prevé en la estimación de la salida y en RTT, para así reducir la influencia de éstas.

[1] En el Capítulo 2 ya se precisó que se haría uso de la aproximación consistente en considerar que el retardo entre atacante y servidor es similar a la mitad del tiempo de ida y vuelta entre ambos.

De este modo, cuanto mayor sea el valor previsto de las desviaciones, será necesario utilizar también un valor de t_{ontime} mayor.

- *Intervalo* (Δ):

  Es el tiempo transcurrido entre la emisión de dos mensajes de ataque consecutivos durante t_{ontime}. El tamaño del intervalo deberá cumplir la condición obvia de ser menor o igual que la duración de t_{ontime}:

$$\Delta \leq t_{ontime} \tag{3.1}$$

  El número de peticiones que se envían durante cada intervalo t_{ontime}, N_p, vendrá dado por la siguiente expresión:

$$N_p = floor\left[\frac{t_{ontime}}{\Delta}\right] + 1 \tag{3.2}$$

  donde la función *floor* redondea hacia abajo al entero más cercano.

- *Tiempo* offtime ($t_{offtime}$):

  Es el intervalo de inactividad previo a t_{ontime} en el periodo básico del ataque, durante el cual no se realiza emisión de mensajes de ningún tipo. Su inicio está situado en el comienzo del periodo de ataque. El valor de este tiempo viene determinado por la siguiente expresión (ver Fig. 3.1):

$$t_{offtime} = DSS - \frac{t_{ontime}}{2} - \frac{\overline{RTT}}{2} \tag{3.3}$$

Los parámetros relacionados anteriormente pueden ser modificados por el atacante para adaptar el ataque a las características del servidor objetivo. En realidad, los parámetros *CPA* y *DSS* vienen dados por la estimación de los instantes de las salidas, por lo que no son parámetros ajustables. Por otro lado, los valores elegidos para t_{ontime} y Δ determinarán la eficiencia obtenida por el ataque, debido a que son los parámetros que consideran, en mayor o menor medida, las desviaciones en la estimación del instante de la salida y el tiempo de ida y vuelta, RTT. Además, estos dos parámetros concretarán también el número de mensajes, N_p, enviados al servidor en cada periodo de ataque. Por último, el parámetro $t_{offtime}$ vendrá dado por el valor de los anteriores según la Expresión (3.3).

En resumen, en la ejecución del ataque DoS de baja tasa, el atacante, para cada salida que predice, debe programar la ejecución de un periodo básico de ataque tal y como ha sido definido.

Por otro lado, esta estrategia seguida por parte del atacante debe complementarse con otra medida, consistente en que siempre que el atacante reciba una respuesta procedente del servidor, se realizarán dos acciones:

1. El atacante envía un nuevo mensaje de ataque.

   Al realizar esta acción, cuando el periodo básico de ataque no tenga éxito en la captura de la posición libre en la cola, se reducirá la probabilidad de que una petición de otro usuario la ocupe. De este modo, el tiempo máximo durante el cual la posición estará disponible para la ocupación por parte de otro usuario será el tiempo de ida y vuelta, RTT. Obviamente, si la salida que se produce en el servidor no corresponde a una petición efectuada por el atacante, éste no recibirá respuesta alguna y, por consiguiente, no emitirá un mensaje de ataque. En este caso, el tiempo durante el cual la posición está disponible para su ocupación por parte de otro usuario podría ser mayor que RTT.

2. El periodo t_{ontime}, en caso de estar activo, se termina.

   Es decir, si llega la salida del servidor cuando aún no se han terminado de enviar los N_p mensajes de ataque, no se mandarán los pendientes. En este caso, los mensajes nuevos, que hipotéticamente se enviaran, no tendrían éxito en la captura de la posición en la cola de servicio, ya que, en caso de que la posición estuviera aún libre, quedaría ocupada por el mensaje de ataque de respuesta anteriormente citado.

3.2.3 Características del ataque DoS de baja tasa contra servidores

El ataque DoS a baja tasa contra servidores, tal y como se ha especificado, presenta las siguientes características (ver clasificación en Capítulo 1 –Apartado 1.4.2–):

- *Ataque de denegación de servicio:*

  Es un ataque de denegación de servicio y, por tanto, su objetivo fundamental es reducir total o parcialmente la capacidad que una víctima posee de ofrecer ciertos servicios a determinados usuarios.

- *Ataque de inundación:*

  El ataque DoS a baja tasa contra servidores es, en primera instancia, un ataque de "inundación" que trata de consumir los recursos de la víctima, de modo que ésta sea incapaz de asignar los recursos necesarios para dar servicio a los usuarios legítimos.

- *Ataque a baja tasa:*

  La característica que hace particular al ataque que se ha presentado reside en el hecho de que el modo en que el atacante satura a la víctima es utilizando una baja tasa de tráfico en lugar de un tráfico elevado. Esto hace que el ataque pueda evitar numerosos detectores de intrusiones que están basados, precisamente, en el hecho de que el potencial ataque utiliza una alta tasa de tráfico.

- *Ataque de vulnerabilidad:*

  El modo de conseguir que la "inundación" del atacante contra la víctima se pueda efectuar usando un tráfico de baja tasa es mediante la explotación de una determinada vulnerabilidad en la víctima. En efecto, aunque dicha vulnerabilidad se describirá detalladamente en los Capítulos 4 y 5, su existencia permitirá al atacante enviar el

tráfico hacia la víctima de forma inteligente, logrando de este modo reducir la tasa necesaria para conseguir los objetivos del ataque. Es por esto por lo que este ataque también se puede definir como de vulnerabilidad.

La vulnerabilidad a explotar dependerá de la naturaleza del servidor víctima del ataque: iterativo o concurrente.

- *Ataque contra servidores:*

  El ataque está diseñado para ser llevado a cabo contra aquellos servidores que responden al modelo presentado en el Capítulo 2, que se ubican en un escenario tal y como está definido en dicho capítulo y que presentan alguna característica adicional que se indicará en los siguientes capítulos.

- *Ubicación del atacante:*

  El atacante puede llevar a cabo el ataque DoS a baja tasa contra servidores tanto de forma centralizada (DoS), es decir, desde una sóla máquina, como de modo distribuido (DDoS). El diseño del ataque no impone ninguna restricción en cuanto al modelo de ubicación elegido, si bien hay que decir que la distribución del ataque conlleva ciertos detalles de implementación comunes a todos los ataques de tipo DDoS (ver Capítulo 1).

3.2.4 Comparación con otros ataques de baja tasa

Dada la importancia de la aparición de los ataques de baja tasa contra el protocolo TCP [Kuzmanovic and Knightly, 2003], ya discutida en el Capítulo 1, es importante presentar las similitudes y diferencias que existen entre éstos y los ataques a baja tasa contra servidores. Se pueden destacar entre ellos las siguientes similitudes:

- *Tipología del ataque*:

  Ambos son ataques de denegación de servicio con baja tasa de emisión de tráfico.

- *Vulnerabilidad*:

  Los dos tratan de tomar ventaja de una vulnerabilidad que les permite enviar su tráfico de forma controlada y optimizada de acuerdo al objetivo.

- *Técnica utilizada*:

  Utilizan una técnica similar para construir los periodos de ataque, es decir, ambos realizan la emisión de paquetes según un esquema ON/OFF.

Por el contrario, existen varias características fundamentales que diferencian ambos tipos de ataques:

- *Objetivo del ataque*:

  En el ataque de baja tasa a TCP, el objetivo del ataque es la capa de transporte, y la ubicación donde se inflige el daño es en algún enlace de la red (no conocido por el

atacante) entre el atacante y la víctima, en el que se provocan sobrecargas controladas para crear la denegación de servicio.

Por otro lado, en el ataque DoS a baja tasa contra servidores planteado en este trabajo, la ubicación donde se provoca el daño es en los sistemas finales, y su objetivo es un servicio concreto o grupo de servicios ubicados en dichos sistemas. Este ataque puede diferenciar los servicios atacados, haciendo que una técnica de detección de estos ataques deba discernir también entre servicios para poder detectarlos.

- *Vulnerabilidad*:

  El ataque de baja tasa a TCP aprovecha el conocimiento del temporizador de retransmisión RTO en el mecanismo de control de flujo del protocolo TCP[2]. Esta vulnerabilidad reside en el propio diseño del protocolo, de modo que para que el ataque pueda ser llevado a cabo se requiere que la implementación de TCP siga la recomendación. El ataque de baja tasa a servidores, sin embargo, estima dinámicamente el momento en que se produce cada salida o respuesta del servidor para provocar la denegación de servicio.

- *Parámetros del ataque*:

  En el periodo del ataque de baja tasa a TCP se establece un valor muy bajo para el parámetro Δ, es decir, durante t_{ontime} la tasa de emisión de paquetes es muy elevada para poder provocar las sobrecargas en los enlaces. Concretamente, dicha tasa debe ser equivalente a la capacidad del enlace en el que se produzca la sobrecarga. En los ataques a baja tasa contra servidores, el parámetro intervalo se puede elegir de modo que la tasa no sea muy elevada durante t_{ontime}.

- *Tasa global del ataque*:

  Mientras que en el ataque de baja tasa a TCP la tasa global medida en el enlace donde se producen las sobrecargas es menor que la tasa máxima del enlace, en el ataque a baja tasa contra servidores, si se quiere denegar el servicio completamente, la tasa del ataque siempre será al menos igual a la aceptada por el servidor. Aunque esto puede hacer parecer más detectable el ataque contra servidores, hay que tener en cuenta que, al crecer la tasa solamente en los flujos encaminados a un solo servidor dentro de una máquina, si existen varios servidores en ella, las tasas globales que recibe el sistema final se pueden mantener por debajo de los niveles de sobrecarga, eludiendo la mayoría de los posibles sistemas de defensa. Además, el ataque a baja tasa contra servidores puede elegir como objetivo reducir, en vez de anular, la capacidad de servicio del servidor, siendo necesaria, en este caso, una tasa de tráfico menor.

- *Consecuencia del ataque*:

  El ataque de baja tasa a TCP genera una pérdida completa de tráfico en el enlace en que se produzca la sobrecarga para todos los flujos que entran en retransmisión. Ello implica que, si un flujo no entrara en retransmisión, tendría disponible toda la capacidad del enlace[3]. Sin embargo, en el ataque a baja tasa contra servidores, los recursos

[2] Dado que está recomendado su uso con un valor estándar de *RTO* para todos los flujos [Allman and Paxson, 1999].

[3] En realidad, esta capacidad se iría utilizando conforme al algoritmo *exponential backoff* de TCP.

de la red tales como enlaces, encaminadores intermedios, etc., no sufren consecuencia alguna, e incluso el propio servidor seguirá ejecutando sus tareas normalmente. Únicamente los usuarios legítimos del servidor serán quienes experimenten la denegación de servicio.

Como primera conclusión de esta discusión se puede inferir que, aunque el trabajo presentado en [Kuzmanovic and Knightly, 2003] aparezca como un caso de ataque aislado con la característica de utilizar baja tasa global de emisión de paquetes, la propuesta del ataque DoS a baja tasa contra servidores hace pensar que los ataques a baja tasa forman en realidad una familia dentro de los ataques DoS, pudiéndose adoptar características y objetivos de ataque diversos.

3.3 Indicadores de estimación del rendimiento del ataque

Para llevar a cabo una evaluación del rendimiento que el ataque es capaz de alcanzar a la hora de infligir una denegación de servicio a un servidor, es necesario definir ciertos indicadores que permitan, de forma cuantitativa, establecer una métrica para realizar dicha evaluación.

Dos son las vertientes o aspectos a analizar en este tipo de ataques. En primer lugar, es interesante comprobar la eficiencia obtenida en cuanto al nivel de denegación de servicio que se está consiguiendo mediante el ataque. Por otro lado, también será necesario cuantificar la tasa o cantidad de tráfico que se está empleando para conseguir dicha eficiencia. Para ambas variables es necesaria la definición de tres indicadores:

- *Disponibilidad (D)*:

  Se define como la ratio, en porcentaje, entre el número de peticiones procedentes de usuarios legítimos que son efectivamente atendidas por el servidor y el número total de peticiones enviadas por éstos.

- *Porcentaje de tiempo disponible* (T_D):

  En un escenario en el que no existe tráfico de usuarios, se define el porcentaje de tiempo disponible, T_D, para un tiempo de observación fijado, como el porcentaje de dicho tiempo de observación durante el cual existe al menos una posición libre en el servidor de modo que una nueva petición entrante pueda ser aceptada.

- *Sobrecarga (S)*:

  Se define como la ratio, en porcentaje, entre la tasa de tráfico generada por el atacante y la máxima tasa de tráfico aceptada por el servidor.

La Tabla 3.1 resume los diferentes indicadores definidos y el aspecto del rendimiento del ataque que evalúan. Así, el indicador D, de acuerdo a su definición, mide el nivel de denegación de servicio experimentado por los usuarios, es decir, la eficiencia del ataque. Sin embargo, aunque D es un buen indicador, su valor no está determinado únicamente por la configuración de los parámetros de ataque, sino que también depende del comportamiento de

Evaluación de la tasa de tráfico	Evaluación de la eficiencia
Sobrecarga (S)	Disponibilidad (D)
	Porcentaje de tiempo disponible (T_D)

Tabla 3.1: Indicadores definidos para evaluar el rendimiento de un ataque DoS a baja tasa contra servidores.

los propios usuarios. Este hecho dificulta la evaluación del ataque de forma independiente del patrón de tráfico de los usuarios. El indicador T_D viene a solventar esta limitación. Nótese que éste es una medida de la eficiencia del ataque basándose sólo en los resultados obtenidos por el tráfico de ataque y, por consiguiente, independiente del patrón de tráfico de usuario.

Por otro lado, siempre que se reduzca el porcentaje de tiempo disponible, el valor de D será también menor puesto que existirá una menor probabilidad de que los usuarios ocupen posiciones en la cola. En efecto, dado que ambos indicadores, D y T_D, son medidas de la eficiencia, presentan una relación directa entre ellas. Así, aunque sus valores difieran, las tendencias serán iguales.

En cuanto a la cantidad de tráfico que el ataque genera, el indicador S aporta una medida normalizada con respecto a la capacidad del servidor, de modo que se pueden comparar diferentes configuraciones de ataque de forma independiente al servidor considerado.

Es necesario clarificar que un valor de sobrecarga de $S = 100\%$ no indica una congestión en la máquina o máquinas en las que reside el servidor. Si bien el servidor se encuentra inserto en una o varias máquinas, dichas máquinas pueden albergar otro tipo de aplicaciones que también hacen uso de sus recursos. De este modo, se puede dar la situación en que el ataque produzca la denegación de servicio en un servidor dentro de una máquina y se mantenga un funcionamiento normal del resto de aplicaciones en la misma máquina. Además, un valor $S < 100\%$ tampoco indica que el servidor no se encuentre en situación de saturación, ya que el indicador S solamente considera el tráfico de ataque. Puede suceder, así, que el tráfico de usuario restante sea el que provoca la saturación en el servidor.

Por último, hay que indicar que el objetivo del ataque será maximizar la eficiencia del mismo a la vez que se utiliza la mínima tasa de tráfico posible. En términos de los indicadores definidos, esta condición es equivalente a minimizar el valor de S, D y T_D.

3.4 Modelos matemáticos para el porcentaje de tiempo disponible

Dado que el atacante puede modificar el comportamiento de su ataque eligiendo el valor de los diferentes parámetros del periodo básico de ataque, es importante abordar la definición de un modelo matemático que relacione el porcentaje de tiempo disponible que genera una configuración específica del ataque como función de los parámetros del mismo. Esto

permitirá acotar los valores posibles a utilizar por el atacante y a entender las dificultades que éste encontrará para obtener el máximo rendimiento en el ataque.

Se puede abordar este estudio realizando una diferenciación entre sistemas servidores con superposición y sin superposición (ver Apartado 2.3.6). El modelado de un sistema con superposición es mucho más complejo que el correspondiente a un sistema que no presenta superposición. Por ello, inicialmente se abordará el problema para escenarios sin superposición, para, posteriormente, ampliar dicho estudio a escenarios con superposición.

3.4.1 Modelo matemático para el porcentaje de tiempo disponible en sistemas sin superposición

En este apartado se aborda la definición de un modelo matemático que relaciona la eficiencia del ataque, en términos de T_D, con los parámetros de configuración del mismo, para los sistemas sin superposición.

Se ha definido un sistema sin superposición como aquél en el que las curvas de probabilidad de ocurrencia $f_j(t)$ correspondientes a cualesquiera dos salidas consecutivas no presentan solapamiento entre sí. Para este tipo de sistemas, se puede abordar el cálculo del porcentaje de tiempo disponible estudiando su valor esperado en cada salida genérica j, $T_{D,j}$, y realizando el promedio durante un periodo de tiempo T durante el que se producen C salidas como:

$$T_D = 100 \cdot \frac{C \cdot T_{D,j}}{T} \tag{3.4}$$

Como paso previo al desarrollo del modelo matemático, se establecerá a continuación la notación necesaria para su uso en las diferentes expresiones. La Fig. 3.2 representa la función de probabilidad de ocurrencia de la salida j, $f_j(t)$, calculada por el atacante. El modelo que se presenta a continuación no asume, excepto cuando se indique explícitamente, ningún tipo de distribución para $f_j(t)$. Ahora bien, tal y como se justificará en los capítulos posteriores, el escenario de ataque será tal que todos los mensajes de ataque son idénticos entre sí, por lo que $f_j(t)$ se podrá modelar mediante la función de probabilidad de una variable normal –Expresión (2.10)–. Sin embargo, dado que la función de probabilidad de ocurrencia de la salida será la que estime el atacante, hay que tener en cuenta que la Expresión (2.10) deberá ser modificada para incorporar el efecto de la variabilidad del tiempo de ida y vuelta, de modo que:

$$f_j(t) = \mathcal{N}(\overline{T_s}; var[T_s] + var[RTT]) \tag{3.5}$$

El periodo básico de ataque, tal y como se recibe en el servidor, está también representado de forma superpuesta en la figura. Además, la llegada de los diferentes mensajes de ataque, representados por flechas verticales continuas, se produce en los instantes a_i ($i \geq 1$) dentro del periodo t_{ontime}. En este ejemplo, solamente tres mensajes de ataque aparecen durante el periodo t_{ontime}, debido al valor especificado para el intervalo Δ, aunque el ejem-

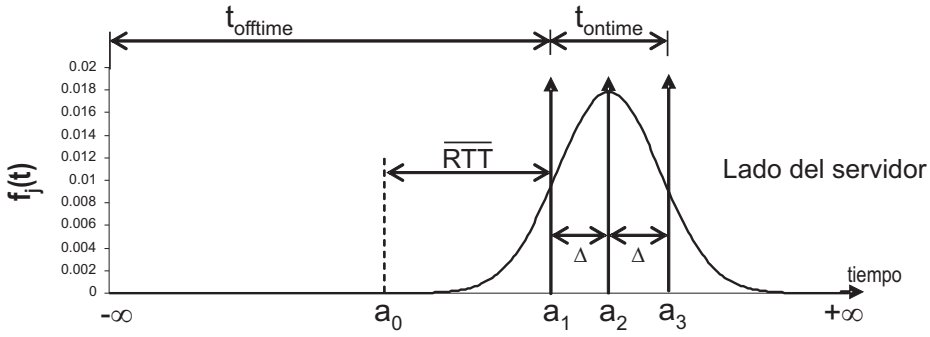


Fig. 3.2: Esquema de la curva de probabilidad de ocurrencia de una salida con un periodo de ataque y sus puntos de cálculo asociados.

plo se puede extender fácilmente a un escenario genérico con n mensajes. En adelante, a los instantes a_i de llegada de un mensaje de ataque se les denominará *puntos de cálculo*.

Además, aparece un punto de cálculo especial, denominado a_0, el cual no se corresponde con la llegada de ningún mensaje de ataque y está situado, por definición, un tiempo $\overline{RTT}$ antes de la llegada del primer paquete de ataque del periodo:

$$a_0 = a_1 - \overline{RTT} \tag{3.6}$$

En general, para cualquier periodo básico de ataque se define un conjunto de puntos de cálculo $\mathcal{A}$, compuesto por:

$$\mathcal{A} = \{a_0, a_1, \ldots, a_n\} \tag{3.7}$$

donde el índice del último mensaje de ataque, n, viene dado por la Expresión (3.2), esto es, $n = N_p$.

Los puntos de cálculo definidos delimitan una serie de intervalos en los que se debe especificar individualmente la forma en que hay que realizar el cálculo del tiempo disponible. De este modo, cuando la salida se produce en el intervalo $(-\infty, a_0)$, el valor del tiempo disponible generado, medido en segundos, será igual a $\overline{RTT}$, dado que el siguiente mensaje de ataque en llegar al servidor será el generado por el atacante como respuesta a la recepción de dicha salida, y este mensaje tardará en llegar un tiempo igual a $\overline{RTT}$ desde que se generó la salida.

En los intervalos que suceden a continuación, es decir, en el rango temporal que va desde a_0 hasta a_n, siempre que se cumpla la condición $a_i - a_{i-1} \leq \overline{RTT} \quad \forall \quad i \in [1, n]$, el tiempo disponible tomará el valor $a_i - t$, siendo t el instante en que se produce la salida, dado que el siguiente mensaje de ataque que llega al servidor lo hará en el instante a_i. Esta condición siempre se cumple para el caso $i = 1$, ya que por definición $a_1 - a_0 = \overline{RTT}$.

Dado que para $i > 1$ se cumple que $a_i - a_{i-1} = \Delta$, la condición se puede expresar de modo equivalente como $\Delta \leq \overline{RTT}$.

En caso de que la condición $\Delta \leq \overline{RTT}$ no se cumpla, el valor que toma el tiempo disponible en los intervalos desde a_0 hasta a_n se puede obtener considerando que cada uno de ellos se puede dividir a su vez en dos subintervalos: $(a_{i-1}, a_i - \overline{RTT})$ y $(a_i - \overline{RTT}, a_i)$. El valor del tiempo disponible en el primer subintervalo corresponderá a $\overline{RTT}$, mientras que en el segundo será $a_i - t$, tal y como se ha justificado anteriormente.

Finalmente, en el último intervalo, (a_n, ∞), el valor que tomará el tiempo disponible será, al igual que en el primero, $\overline{RTT}$.

Considerando los valores que toma el tiempo disponible en cada uno de los intervalos, la ecuación que evalúa el valor esperado de $T_{D,j}$ para la salida j en el servidor, cuando se cumple la condición $\Delta \leq \overline{RTT}$, es la siguiente:

$$\begin{aligned} T_{D,j}\big|_{(\Delta \leq \overline{RTT})} &= \int_{-\infty}^{a_0} \overline{RTT} \cdot f_j(t)dt + \sum_{i=1}^{n} \left[\int_{a_{i-1}}^{a_i} (a_i - t) \cdot f_j(t)dt \right] + \\ &+ \int_{a_n}^{\infty} \overline{RTT} \cdot f_j(t)dt \end{aligned} \tag{3.8}$$

De modo análogo, para el caso en que no se cumple la condición $\Delta \leq \overline{RTT}$, la expresión se convierte en:

$$\begin{aligned} T_{D,j}\big|_{(\Delta > \overline{RTT})} &= \sum_{i=1}^{n} \left[\int_{a_{i-1}}^{a_i - \overline{RTT}} \overline{RTT} \cdot f_j(t)dt + \int_{a_i - \overline{RTT}}^{a_i} (a_i - t) \cdot f_j(t)dt \right] + \\ &+ \int_{-\infty}^{a_0} \overline{RTT} \cdot f_j(t)dt + \int_{a_n}^{\infty} \overline{RTT} \cdot f_j(t)dt \end{aligned} \tag{3.9}$$

La resolución de estas dos ecuaciones proporciona el tiempo disponible sin necesidad de hacer consideración alguna sobre la distribución de $f_j(t)$. Sin embargo, un caso que resultará de interés es aquél en el que la función $f_j(t)$ adopta la función de probabilidad de una variable normal con media cero y varianza $var[T_s] = \sigma^2$, dado que es el que corresponde al modelado del tiempo de servicio para peticiones idénticas –Expresión (2.10)–:

$$f_j(t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{t^2}{2\sigma^2}} \tag{3.10}$$

En este caso se pueden resolver las Expresiones (3.8) y (3.9) analíticamente. Como ejemplo, se resolverá a continuación la Expresión (3.8). Para ello, en primer lugar se consideran los términos de la integral que corresponden a la función $f_j(t)$ multiplicada por una constante (a_i o $\overline{RTT}$), y por otro lado, los términos que aparecen afectados por la variable t.

Los términos correspondientes a la función $f_j(t)$ multiplicados por una constante resultan en:

$$\begin{aligned}&\overline{RTT} \cdot F_j(a_0) + a_1 \cdot \big[F_j(a_1) - F_j(a_0)\big] + a_2 \cdot \big[F_j(a_2) - F_j(a_1)\big] + \ldots + \\ &+ a_n \cdot \big[F_j(a_n) - F_j(a_{n-1})\big] + \overline{RTT} \cdot \big[1 - F_j(a_n)\big] = \\ &= \overline{RTT} \cdot \big[F_j(a_0) + 1 - F_j(a_n)\big] + \sum_{i=1}^{n} a_i \cdot \big[F_j(a_i) - F_j(a_{i-1})\big]\end{aligned}$$

donde $F_j(t)$ es la función de distribución correspondiente a $f_j(t)$.

Considerando ahora los términos que están afectados por la variable t en la Expresión (3.8), éstos resultarán en:

$$-\int_{a_0}^{a_n} t \cdot f_j(t)dt = -\int_{a_0}^{a_n} t \cdot \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{t^2}{2\sigma^2}} dt = \frac{\sigma}{\sqrt{2\pi}} \cdot (e^{-\frac{a_n^2}{2\sigma^2}} - e^{-\frac{a_0^2}{2\sigma^2}})$$

La suma de las contribuciones de todos los términos en la integral nos da finalmente el valor esperado del tiempo disponible producido en cada salida, $T_{D,j}\big|_{(\Delta \leq \overline{RTT})}$:

$$\begin{aligned}T_{D,j}\big|_{(\Delta \leq \overline{RTT})} \quad = \quad & \overline{RTT} \cdot \big[F_j(a_0) + 1 - F_j(a_n)\big] + \\ & + \sum_{i=1}^{n} a_i \cdot \big[F_j(a_i) - F_j(a_{i-1})\big] + \\ & + \frac{\sigma}{\sqrt{2\pi}} \cdot (e^{-\frac{a_n^2}{2\sigma^2}} - e^{-\frac{a_0^2}{2\sigma^2}})\end{aligned} \tag{3.11}$$

3.4.2 Modelo matemático para el porcentaje de tiempo disponible en sistemas con superposición

Tal y como se ha definido en el Apartado 2.3.6, se dice que dos salidas consecutivas presentan superposición cuando sus respectivas curvas de probabilidad de ocurrencia se superponen en el tiempo un valor ε. Para que un sistema se considere con superposición, esta condición se debe cumplir al menos para dos salidas consecutivas.

El cálculo del tiempo disponible usando las Expresiones (3.8) y (3.9) supone que la distancia de superposición, esto es, la distancia entre las curvas de probabilidad de ocurrencia, es suficientemente grande como para considerar independientes las influencias de unas salidas con respecto a las otras. Dicho de otro modo, cuando no hay superposición no existe solapamiento entre las diferentes curvas $f_j(t)$ y, por tanto, tampoco entre los ciclos de t_{ontime} asociados a ellas. Es por esto que el cálculo del porcentaje de tiempo disponible, T_D, se realiza para una salida genérica j, sin tener en cuenta la influencia que puedan tener las otras.

En el caso de los sistemas con superposición este método de cálculo para el tiempo disponible no es adecuado, debido precisamente al solapamiento de las curvas $f_j(t)$. A continuación se justificará esta afirmación y se propondrá la nueva estrategia para la evaluación del tiempo disponible.

Durante un periodo genérico de tiempo, T, en el que un servidor es víctima de un ataque DoS a baja tasa se irán sucediendo, una tras otra, las curvas $f_j(t)$ correspondientes a las diferentes salidas que el atacante va prediciendo. Nótese un detalle interesante al respecto. Estas curvas $f_j(t)$ siguen un orden secuencial. Ahora bien, debido a que representan solamente las probabilidades de que las correspondientes salidas ocurran, una vez dada una secuencia temporal de $f_j(t)$ no se puede deducir a ciencia cierta cuál será la secuencia temporal en que las salidas ocurrirán, ya que el orden de las $f_j(t)$ y el de las salidas no tiene por qué ser el mismo.

A la vez, durante dicho periodo de tiempo, la ejecución del ataque DoS a baja tasa consistirá en la generación de periodos básicos de ataque de forma que cada curva $f_j(t)$ tendrá asociado un ciclo *ontime* en forma de mensajes de ataque. De forma consecuente, estos ciclos *ontime* también sucederán de forma secuencial y en el mismo orden que las $f_j(t)$, de modo que, aunque pueda existir solapamiento entre dos o más de ellos, se puede establecer una secuencia temporal si se considera el instante de inicio de cada uno.

Además, durante T, cada una de las $f_j(t)$ contribuye a la generación de tiempo disponible según la probabilidad de que la salida asociada aparezca en una posición o en otra. En el modelo matemático para sistemas sin superposición presentado con anterioridad, dado que no se produce solapamiento entre las $f_j(t)$, para cualquier instante $t \in T$ considerado solamente una curva $f_j(t)$ contribuye a T_D. De este modo, en dicho modelo se puede evaluar el tiempo disponible medio generado por la influencia de cada $f_j(t)$ de forma independiente y, posteriormente, sumar las contribuciones de todas las $f_j(t)$, multiplicando por el número de capturas C –Expresión (3.4)–.

Sin embargo, en los sistemas con superposición, en un instante $t \in T$ dado pueden existir múltiples $f_j(t)$ contribuyendo simultáneamente al valor de T_D. Esto provoca que no se pueda considerar cada $f_j(t)$ por separado y, por tanto, haya que elegir una nueva estrategia para contabilizar las contribuciones al valor del tiempo disponible por parte de todas las $f_j(t)$.

En este contexto, la estrategia que se seguirá para considerar las contribuciones de todas las curvas $f_j(t)$ que aparecen durante el tiempo de observación T será la de dividir dicho tiempo en *secciones*. Cada una de las secciones durará desde el inicio de un ciclo *ontime* hasta el inicio del siguiente. Al tiempo disponible que se genera, en media, debido a la contribución de todas las $f_j(t)$ existentes en una sección genérica, nos referiremos como $\hat{T}_D$, medido en segundos. Dado que cada ciclo *ontime* va asociado a una $f_j(t)$ en el servidor, se puede decir que existirán tantos ciclos *ontime* como capturas se produzcan. Por tanto, cuando se considere un tiempo de observación T durante el cual se realizan C capturas, el porcentaje de tiempo disponible se podrá calcular como:

$$T_D = 100 \cdot \frac{C}{T} \cdot \hat{T}_D \tag{3.12}$$

Nótese que $\hat{T}_D$ representa un valor medio para todas las posibles secciones que aparezcan durante T. Supóngase ahora conocido el valor medio del tiempo disponible que se genera para todas las secciones cuya distancia, es decir, el tiempo entre el inicio de un *ontime* y el siguiente, sea la misma. Esta distancia, dada su definición, coincide con la distancia de superposición entre las curvas de probabilidad de ocurrencia asociadas a dichos ciclos de *ontime*, es decir, s. Así, debido a que este nuevo tiempo disponible medio depende de s, se utilizará para él la notación $T_D(s)$. Ya que el término $\hat{T}_D$ representará precisamente el valor medio de los diferentes $T_D(s)$, se podrá calcular utilizando la función de probabilidad de la distancia de superposición, $g(s)$, del siguiente modo:

$$\hat{T}_D = \int_0^\infty T_D(s) \cdot g(s) ds \tag{3.13}$$

de forma que, sustituyendo en (3.12), se obtiene que:

$$T_D = 100 \cdot \frac{C}{T} \cdot \int_0^\infty T_D(s) \cdot g(s) ds \tag{3.14}$$

En este punto, conocida la distribución $g(s)$ y los valores C y T, el problema de la obtención del porcentaje de tiempo disponible se ha reducido precisamente a la evaluación del término $T_D(s)$. Este término, como ya se ha comentado, evalúa la contribución de todas las $f_j(t)$ que aparezcan en una sección genérica (tiempo entre los inicios de dos *ontime* consecutivos) cuya distancia es s. Ahora bien, este cálculo presenta una gran complejidad debido, fundamentalmente, a dos factores:

1. La existencia de superposición hace que pueda existir más de una función de distribución de probabilidad $f_j(t)$ de valor no nulo en la sección considerada. En general, no solamente el número, sino también la posición de las $f_j(t)$ existentes en la sección considerada dependerá de las distancias de superposición existentes entre las diferentes $f_j(t)$ consecutivas anteriores y/o también posteriores a las correspondientes a la sección considerada. Por ejemplo, si se toma una sección entre los *ontime* relativos a $f_j(t)$ y $f_{j+1}(t)$, la existencia de un valor de probabilidad no nulo en dicha sección para $f_{j-1}(t)$ dependerá de la distancia de superposición entre ésta y $f_j(t)$. Así sucesivamente ocurre con todas las $f(t)$ anteriores y posteriores a $f_j(t)$ y $f_{j+1}(t)$.

   Si a esto se añade el hecho de que la distancia de superposición s presenta una alta variabilidad en sistemas con superposición (ver Fig. 2.12), se puede deducir que el número posible de escenarios que se podrán observar en las diferentes secciones cuya distancia es s será muy elevado. Habida cuenta de que el cálculo de $T_D(s)$ deberá considerar la contribución de las curvas $f_j(t)$ en todos los escenarios posibles, se puede intuir la complejidad en este cálculo.

2. Para el cálculo del tiempo disponible que se genera debido a la contribución de determinadas $f_j(t)$ hay que tener en cuenta no sólo el número y la posición de dichas curvas, sino también la influencia de los distintos mensajes procedentes del atacante. En cada sección considerada, tanto el número como la posición de los mensajes de ataque que se corresponden con *ontime* es conocido. Ahora bien, tal y como se ha definido la

estrategia de ataque, no solamente llegarán al servidor mensajes de ataque debidos a los ciclos de *ontime*, sino que cada salida que se produce en el servidor genera, al llegar al atacante, un nuevo mensaje de ataque como respuesta. Esto provoca que existan ciertos mensajes de ataque cuyo número y posición dependen de las salidas generadas con anterioridad a la sección considerada. En adelante llamaremos a éstos *mensajes de ataque adicionales*. Es evidente que, tal y como se ha planteado en el punto anterior, si es realmente complejo conocer el número y la posición de las $f_j(t)$ existentes en una sección, también lo será conocer los correspondientes a los mensajes de ataque adicionales.

Presentación del escenario genérico para el cálculo de $T_D(s)$

Para abordar la evaluación del término $T_D(s)$, se partirá de un escenario tipo para el que se propondrá un método de cálculo. El escenario tipo que se estudiará está representado en la Fig. 3.3. En ella se muestra el esquema correspondiente a dos $f(t)$ consecutivas, $f_j(t)$ y $f_{j+1}(t)$, y en él se pueden observar los dos periodos de ataque destinados a ocupar la posición liberada por cada una de sus salidas. Al igual que en el modelo para sistemas sin superposición, estas dos funciones son estimadas por el atacante en el proceso de ataque, de modo que se pueden caracterizar mediante la Expresión (3.5). Dado que s es la distancia de superposición entre las curvas de ambas salidas, se cumplirá que:

$$f_j(t) = f_{j+1}(t+s) \tag{3.15}$$

Se definen para este escenario ciertos puntos de cálculo correspondientes a las dos salidas, tal y como están representados en la Fig. 3.3. Cada vez que llega un mensaje de ataque al servidor se considera la aparición de un punto de cálculo. Por tanto, se considerarán puntos de cálculo correspondientes a los mensajes de ataque de ambos periodos *ontime*. Así, la notación $a_i,\ i \in [1, 2, ..., n]$, hace referencia a los puntos de cálculo asociados al periodo de ataque correspondiente a la salida con curva de probabilidad $f_j(t)$, mientras que los puntos $a'_i,\ i \in [1, 2, ..., n]$, se refieren a los puntos de cálculo generados por el periodo de ataque asociado a la curva $f_{j+1}(t)$. Además, los puntos a_0 y a'_0 se situan a una distancia $\overline{RTT}$ antes de a_1 y a'_1, respectivamente.

$$\begin{aligned} a_0 &= a_1 - \overline{RTT} \\ a'_0 &= a'_1 - \overline{RTT} \end{aligned} \tag{3.16}$$

En este escenario se definen también unos *tramos de cálculo*, que servirán posteriormente para descomponer el cálculo de $T_D(s)$ en varios factores. El conjunto de tramos de cálculo definidos para cada dos salidas, mostrados en la Fig. 3.3, se especifica como $\mathcal{I} = \{A, B, C, D, E, F, G\}$.

Es evidente que las características de los tramos A, B, F y G no dependerán del valor que se considere para la distancia de superposición, s. De este modo, los límites que abarcan dichos tramos son los siguientes:

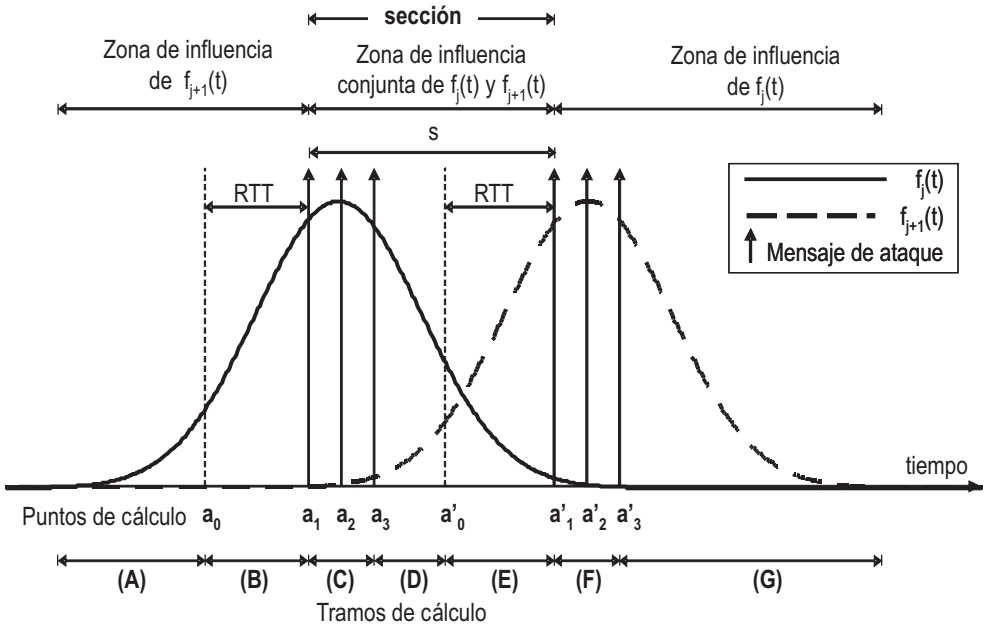


Fig. 3.3: Esquema de dos salidas con superposición: funciones de probabilidad de ocurrencia de las salidas, puntos de cálculo, zonas de influencia y tramos de cálculo del tiempo disponible para una distancia de superposición s.

$$\begin{aligned} \text{Tramo } A &\implies (-\infty, a_0) \\ \text{Tramo } B &\implies (a_0, a_1) \\ \text{Tramo } F &\implies (a'_1, a'_n) \\ \text{Tramo } G &\implies (a'_n, \infty) \end{aligned}$$

Por el contrario, nótese que cuando el valor de $s = 0$, los tramos C, D y E, que son precisamente los correspondientes a una sección, desaparecen. Por tanto, dependiendo del valor de s considerado para el escenario genérico, estos tramos quedarán afectados de la siguiente forma:

- *Tramo D*:

  Este tramo, que corresponde al intervalo (a_n, a'_0), existirá siempre que se cumpla que la diferencia entre dichos puntos sea mayor que cero, es decir, $a'_0 - a_n > 0$. En caso contrario, este tramo no existirá y, por tanto, no se realizará cálculo alguno sobre él. Esta condición, expresada en términos de la distancia de superposición, es:

$$\exists \text{ tramo } D \Leftrightarrow s > t_{ontime} + \overline{RTT} \tag{3.17}$$

- *Tramo E*:

  La existencia de este tramo, comprendido en el intervalo (a'_0, a'_1), dependerá de que se cumpla $a'_1 - a_n > 0$. Como antes, si el tramo E no existe, no se tomará en cuenta para el cálculo del tiempo disponible. La condición de existencia para dicho tramo, en función de la distancia de superposición, es:

  $$\exists \text{ tramo } E \Leftrightarrow s > t_{ontime} \tag{3.18}$$

  Ahora bien, hay que tener en cuenta que, cuando el valor de la distancia de superposición se sitúa en el rango $s \in (t_{ontime}, \overline{RTT} + t_{ontime}]$, el valor inicial del intervalo ya no será a'_0, sino a_n. De forma genérica, se dirá que el tramo E está comprendido en el intervalo (φ_E, a'_1) donde se cumple que:

  $$\varphi_E = \begin{cases} a'_0 & , \quad \text{si } s > \overline{RTT} + t_{ontime} \\ a_n & , \quad \text{si } t_{ontime} < s \leq \overline{RTT} + t_{ontime} \end{cases} \tag{3.19}$$

- *Tramo C*:

  Respecto a este tramo, que en la Fig. 3.3 comprende (a_1, a_n), puede decirse que el tamaño de dicho tramo variará dependiendo de la posición del periodo *ontime* asociado a la salida $j+1$. De este modo, cuando se cumple que $s \geq t_{ontime}$, el tramo C se situará entre (a_1, a_n). Sin embargo, para valores $s \in [0, t_{ontime})$, el límite del tramo vendrá dado por el inicio del periodo *ontime* de la salida $j+1$, es decir, por el punto de cálculo a'_1. De forma genérica, diremos que el tramo C está comprendido en el intervalo (a_1, φ_C), de forma que:

  $$\varphi_C = \begin{cases} a'_1 & , \quad \text{si } s < t_{ontime} \\ a_n & , \quad \text{si } s \geq t_{ontime} \end{cases} \tag{3.20}$$

Finalmente, también hay que señalar que cada uno de estos tramos puede estar formado, a su vez, por uno o varios *intervalos*. Los intervalos están delimitados por los puntos de cálculo existentes dentro de cada tramo. Así, un tramo puede estar compuesto por un solo intervalo, como es el caso de los tramos A, B, D, E y G, o bien por varios, como sucede en los tramos C y F.

Aproximaciones en el cálculo del tiempo disponible

Una vez presentado el escenario genérico de la Fig. 3.3, que será la base para el cálculo del término $T_D(s)$, se pretende a continuación explicar el método que se sigue para la obtención de las expresiones analíticas que evalúan dicho término.

De acuerdo a lo establecido en (3.14), $T_D(s)$ recoge la contribución media al tiempo libre generada por todas las $f_j(t)$ posibles cuyo valor es no nulo en una sección de tiempo de duración s y correspondiente a los inicios de los ciclos de *ontime* de dos $f_j(t)$ consecutivas. Ya se ha mencionado anteriormente la complejidad asociada a la evaluación de este término,

por lo que para realizar el cálculo de $T_D(s)$ será necesario realizar algunas aproximaciones que permitan su evaluación analítica. Estas aproximaciones se presentan a continuación.

- *Primera aproximación*:

  La zona ubicada entre los inicios de dos ciclos *ontime* consecutivos corresponde a los tramos C, D y E (sección) representados en la Fig. 3.3. Por tanto, será en estos tramos donde habrá que considerar la influencia de las múltiples $f_j(t)$ no nulas sobre el tiempo disponible. Nótese que, en dicha zona, la contribución al tiempo disponible por parte de una $f(t)$ genérica será tanto más elevada cuanto mayor sea su valor en dicha zona. Esto implica que, si se considera una sección entre las funciones $f_j(t)$ y $f_{j+1}(t)$, serán precisamente estas dos funciones las que, previsiblemente, mayor contribución al tiempo disponible aportarán. Por ello, en una primera aproximación, se considerará que en una sección solamente existirá la contribución de las dos funciones de probabilidad de ocurrencia que mayor valor tienen en dicha sección –genéricamente $f_j(t)$ y $f_{j+1}(t)$–, de modo que la contribución por parte del resto de funciones $f(t)$ se desprecia. Por esta razón, en el escenario de la Fig. 3.3, a la zona que comprende la sección entre los inicios de *ontime* se la denominará *zona de influencia conjunta de* $f_j(t)$ *y* $f_{j+1}(t)$.

  Adicionalmente, para facilitar el cálculo del término $T_D(s)$, se utilizan los diferentes tramos de cálculo, de modo que las contribuciones en cada uno de dichos tramos, $T_D^i(s),\ i \in \{C, D, E\}$, se evaluarán por separado. Finalmente se tendrá que:

$$T_D(s) = \sum_{i \in \{C,D,E\}} T_D^i(s) \tag{3.21}$$

- *Segunda aproximación*:

  Aunque el valor calculado para $T_D(s)$ mediante la primera aproximación recogerá la principal contribución al tiempo disponible mediante la consideración de, únicamente, $f_j(t)$ y $f_{j+1}(t)$, sería deseable incorporar también las contribuciones por parte de las restantes $f(t)$ –anteriores a $f_j(t)$ y posteriores a $f_{j+1}(t)$– en los tramos C, D y E.

  Ahora bien, debido a la ya mencionada dificultad de determinar el número y la posición de las diferentes $f(t)$ anteriores y posteriores, para poder considerar la aportación de éstas al tiempo disponible en la sección considerada se hará una segunda aproximación.

  En lugar de calcular la contribución de todas las posibles $f(t)$ existentes en una sección, para lo que haría falta conocer al menos su número, se evaluará la contribución media que cada $f(t)$ aporta al tiempo disponible cuando su valor no es nulo entre los inicios de *ontime* correspondientes a otras dos $f(t)$ cualesquiera y diferentes de ella. De esta forma, una vez incorporada esta contribución media aportada por una $f(t)$ al término $T_D(s)$, al multiplicar por el número de $f(t)$ (o, equivalentemente, por el número de capturas, C –ver Expresión (3.14)–), se obtendrá un valor de T_D que también considera la aportación de las múltiples $f(t)$ en la sección que se considere.

  Así, se puede comprobar que, en el escenario de la Fig. 3.3, se han representado las funciones $f_j(t)$ y $f_{j+1}(t)$ completas, y no solamente su parte correspondiente en los tramos C, D y E, correspondientes a la sección estudiada. El objetivo es, como

ya se ha comentado, poder incorporar al cálculo de T_D la influencia de cada $f(t)$ en las secciones que no son adyacentes a ella. Para ello, lo que se hará es calcular la contribución de la función $f_j(t)$ en los tramos F y G y, por otro lado, la de $f_{j+1}(t)$ en A y B. Nótese que no se debe considerar la contribución de la función $f_j(t)$ en los tramos A y B, ya que esta contribución estará ya considerada en la sección anterior a la estudiada, es decir, la existente entre $f_{j-1}(t)$ y $f_j(t)$. De igual modo sucede para la función $f_{j+1}(t)$, para la que no se incorpora la contribución en los tramos F y G, ya que ésta ya se considera en la sección entre $f_{j+1}(t)$ y $f_{j+2}(t)$. Por esta razón, a la zona comprendida por los tramos F y G se le denomina *zona de influencia de* $f_j(t)$, mientras que a la correspondiente a los tramos A y B se le llamará *zona de influencia de* $f_{j+1}(t)$ (ver estas zonas representadas en la Fig. 3.3).

De este modo, incorporar la influencia de todas las funciones $f(t)$ en la zona estudiada (segunda aproximación), es equivalente a modificar la Expresión (3.21) para incorporar las contribuciones que se han comentado previamente en todos los tramos $i \in \mathcal{I}$:

$$T_D(s) = \sum_{i \in \mathcal{I}} T_D^i(s) \tag{3.22}$$

- *Tercera aproximación*:

  La tercera aproximación que se realizará para evaluar el término $T_D(s)$ es la que permite la incorporación de la información sobre la posición correspondiente a los mensajes de ataque adicionales, es decir, aquellos que el atacante genera como respuesta a la recepción de una salida.

  Como se mostrará más adelante, esta aproximación se incorporará en el cálculo de $T_D(s)$ para cada uno de los tramos, de modo que las expresiones $T_D^i(s)$ serán las que quedarán modificadas mediante la introducción de la influencia de estos mensajes de ataque adicionales.

En adelante se desarrollarán los métodos y las estrategias de cálculo que permitirán evaluar $T_D(s)$ considerando cada una de las tres aproximaciones anteriormente propuestas, si bien las expresiones definitivas correspondientes a cada uno de los tramos de cálculo definidos, $T_D^i(s)$, se relacionarán posteriormente en el Apéndice 1 del presente capítulo.

1ª aproximación.
Cálculo de $T_D(s)$ en la zona de influencia conjunta de $f_j(t)$ y $f_{j+1}(t)$

En esta aproximación, tal y como se ha comentado, se evalúan los términos $T_D^i(s)$, $i \in \{C, D, E\}$, de forma que solamente se consideran las contribuciones al tiempo disponible aportadas por las curvas $f_j(t)$ y $f_{j+1}(t)$. Además, en este caso no se considerará tampoco la influencia que tiene en el cálculo para un tramo la llegada de mensajes de ataque adicionales que son respuesta a salidas generadas en tramos anteriores al considerado. Solamente cuando la salida se produzca en el tramo considerado se tendrá en cuenta que, tras un tiempo RTT, llegará un mensaje de ataque adicional al servidor.

Ya se ha comentado que el cálculo de $T_D(s)$ se fraccionará para cada uno de los tramos $i \in \mathcal{I}$. A su vez, cuando un tramo esté formado por varios intervalos, se dividirá también el cálculo del término $T_D^i(s)$ en la suma de las contribuciones al tiempo disponible en cada uno de los intervalos que lo forman. Por ello, en adelante, se estudiará cómo calcular dichas contribuciones en un intervalo genérico.

Dado que solamente se consideran dos funciones $f(t)$, los escenarios a tener en cuenta se reducirán a aquellos en los que ocurren cero, una o dos salidas en el intervalo considerado. El escenario en que aparecen cero salidas no se tendrá en cuenta porque, obviamente, si no se produce ninguna salida, el tiempo disponible generado será nulo.

El escenario más sencillo es aquel en el que se produce una única salida en el intervalo considerado, que denominaremos escenario U. En este caso, el tiempo disponible que se genera es el que transcurre desde el instante de aparición de la salida, t, hasta el instante en que llega al servidor el siguiente mensaje de ataque, que corresponde con el siguiente punto de cálculo, dígase $a_i,\ i > 0$.

De esta forma, el valor instantáneo que se obtendrá para el tiempo disponible cuando se produce una única salida, $t_D^U(t)|_{a_i}$, será $a_i - t$, si se cumple que $i > 0$ y $a_i - t < \overline{RTT}$. Ahora bien, ya es conocido que la generación de una salida en el servidor provoca que, al llegar al atacante, éste genere un nuevo mensaje de ataque como respuesta a dicha salida. Esto hace que el valor máximo que pueda tomar el tiempo disponible para cada salida sea $\overline{RTT}$. Por tanto, el valor correcto será:

$$t_D^U(t)\big|_{a_i} = min[a_i - t, \overline{RTT}], \quad i > 0 \tag{3.23}$$

Por otro lado, para el escenario en que se producen dos salidas simultáneas dentro del mismo intervalo genérico (a, b) y no separadas más de un tiempo $\overline{RTT}$ (escenario $\overline{U}$), el tiempo disponible instantáneo que se generará, $t_D^{\overline{U}}|_a^b$, transcurre desde el instante en que se produce la primera salida hasta que llegan dos mensajes de ataque. En este caso, dado que el instante de la primera salida se determina solamente de forma probabilística y que el segundo mensaje de ataque puede incluso pertenecer a otro tramo distinto del considerado, se realiza una aproximación para $t_D^{\overline{U}}|_a^b$. Nótese que este valor es independiente de t, ya que esta aproximación consiste en realizar la media uniforme entre el valor máximo y el mínimo que puede adoptar $t_D^{\overline{U}}|_a^b$ en esta situación. Por ejemplo, en el diagrama de la Fig. 3.3, si se produjeran las dos salidas en el intervalo (a_2, a_3), el valor mínimo de $t_D^{\overline{U}}|_{a_2}^{a_3}$ se alcanzará cuando ambas ocurran casi en a_3: $t_D^{\overline{U}}|_{a_2}^{a_3}(min) = \overline{RTT}$. Por otro lado, el valor máximo se alcanzará cuando una salida se produzca cerca de a_2 y la otra cerca de a_3: $t_D^{\overline{U}}|_{a_2}^{a_3}(max) = \Delta + \overline{RTT}$. Por tanto, para este ejemplo, el valor que se adoptaría sería la media uniforme de ambos:

$$t_D^{\overline{U}}\big|_{a_2}^{a_3} = \frac{t_D^{\overline{U}}\big|_{a_2}^{a_3}(max) + t_D^{\overline{U}}\big|_{a_2}^{a_3}(min)}{2} = \frac{\Delta}{2} + \overline{RTT}$$

Nótese que esta expresión no es más que un ejemplo que ilustra el método de cálculo del factor $t_D^{\overline{U}}|_a^b$. Este método habrá que particularizarlo para cada intervalo.

Si se considera ahora el escenario en el que se producen dos salidas en el mismo intervalo, pero de modo que los instantes de ocurrencia de ambas distan más de $\overline{RTT}$, realmente el tiempo que se obtendrá para cada una de ellas se derivará de considerar que se han producido de forma única en el tramo. Es decir, en este caso, el hecho de que se produzcan las dos salidas en el tramo no varía el valor del tiempo disponible instantáneo generado por cada una de las salidas con respecto al dado por la Expresión (3.23).

La probabilidad de que suceda el evento U será, por tanto, la que corresponde a que se produzca como única salida bien la j, bien la $j+1$, en el intervalo considerado, o bien ambas pero separadas por un tiempo $\overline{RTT}$. Veamos a continuación cómo se calcula esta probabilidad.

Dado que se están considerando solamente las funciones $f_j(t)$ y $f_{j+1}(t)$, dentro de un intervalo genérico (a,b) y para un instante $t \in (a,b)$ en el que se ha producido la salida j, la probabilidad de que no se produzca la salida $j+1$ en un entorno $t \pm \overline{RTT}$ dentro de dicho intervalo, $P_j^U(t)|_a^b$, se puede calcular como:

$$P_j^U(t)\Big|_a^b = 1 - F_{j+1}\left[t + min[b-t, \overline{RTT}]\right] + F_{j+1}\left[t - min[t-a, \overline{RTT}]\right] \tag{3.24}$$

donde $F_{j+1}(t)$ es la función de distribución correspondiente a la función de probabilidad $f_{j+1}(t)$.

De igual modo, se puede decir que, dentro de un intervalo genérico (a,b) y para un instante $t \in (a,b)$ en el que se ha producido esta vez la salida $j+1$, la probabilidad de que no se produzca la salida j en un entorno $t \pm \overline{RTT}$ dentro de dicho intervalo, $P_{j+1}^U(t)|_a^b$, se puede calcular como:

$$P_{j+1}^U(t)\Big|_a^b = 1 - F_j\left[t + min[b-t, \overline{RTT}]\right] + F_j\left[t - min[t-a, \overline{RTT}]\right] \tag{3.25}$$

siendo $F_j(t)$ es la función de distribución correspondiente a $f_j(t)$.

En cuanto a la probabilidad de que suceda el escenario $\overline{U}$, ésta será la correspondiente a que se generen, dentro de un intervalo genérico considerado (a,b), dos salidas en un entorno $t \pm \overline{RTT}$. Dado que se produce la salida j en el instante t del intervalo (a,b), la probabilidad de que se produzca la salida j en un entorno $t \pm \overline{RTT}$ dentro de dicho intervalo, $P_j^{\overline{U}}(t)|_a^b$, será:

$$P_j^{\overline{U}}(t)\Big|_a^b = F_{j+1}\left[t + min[b-t, \overline{RTT}]\right] - F_{j+1}\left[t - min[t-a, \overline{RTT}]\right] \tag{3.26}$$

De igual modo, supuesto que en el mismo intervalo se ha producido la salida $j+1$ en t, la probabilidad de que se produzca la salida $j+1$ en un entorno $t \pm \overline{RTT}$ dentro de dicho intervalo, $P_{j+1}^{\overline{U}}(t)|_a^b$, será:

$$P_{j+1}^{\overline{U}}(t)\Big|_a^b = F_j\left[t + min[b-t, \overline{RTT}]\right] - F_j\left[t - min[t-a, \overline{RTT}]\right] \tag{3.27}$$

Nótese la complementariedad de los términos definidos en las Expresiones (3.24) y (3.26), y también de los correspondientes a las Expresiones (3.25) y (3.27) de modo que se cumple:

$$\begin{aligned} P_j^U(t)\big|_a^b + P_j^{\overline{U}}(t)\big|_a^b &= 1 \\ P_{j+1}^U(t)\big|_a^b + P_{j+1}^{\overline{U}}(t)\big|_a^b &= 1 \end{aligned} \tag{3.28}$$

Una vez definidas estas probabilidades, el cálculo del tiempo disponible en un tramo genérico i formado por el intervalo (a, b), $T_D^i(s)$, se realizará considerando, en cada uno de los escenarios, la integración para los posibles valores de tiempo t dentro del intervalo en los que pueden ocurrir las salidas, teniendo en cuenta los valores instantáneos del tiempo disponible generado en cada escenario y las probabilidades de que dichos escenarios sucedan, de modo que se obtiene:

$$\begin{aligned} T_D^i(s) \;=\; & \int_a^b t_D^U(t)\big|_b \cdot P_j^U(t)\big|_a^b \cdot f_j(t)dt + \int_a^b t_D^U(t)\big|_b \cdot P_{j+1}^U(t)\big|_a^b \cdot f_{j+1}(t)dt + \\ & + \int_a^b t_D^{\overline{U}}\big|_a^b \cdot P_j^{\overline{U}}(t)\big|_a^b \cdot f_j(t)dt \end{aligned} \tag{3.29}$$

donde el tercer término de la ecuación, que considera la aportación debida al escenario $\overline{U}$, se puede calcular también con el factor $P_{j+1}^{\overline{U}}(t)|_a^b$, de modo que resulta la ecuación equivalente:

$$\begin{aligned} T_D^i(s) \;=\; & \int_a^b t_D^U(t)\big|_b \cdot P_j^U(t)\big|_a^b \cdot f_j(t)dt + \int_a^b t_D^U(t)\big|_b \cdot P_{j+1}^U(t)\big|_a^b \cdot f_{j+1}(t)dt + \\ & \int_a^b t_D^{\overline{U}}\big|_a^b \cdot P_{j+1}^{\overline{U}}(t)\big|_a^b \cdot f_{j+1}(t)dt \end{aligned} \tag{3.30}$$

Por último, es de destacar el caso especial en el que, para el tramo i considerado con límites (a, b), se cumple la condición $b - a \leq \overline{RTT}$. Este caso permite simplificar las expresiones (3.29) y (3.30), ya que los términos para las probabilidades se hacen independientes de la variable t:

$$\begin{aligned} P_j^U(t)\big|_a^b &= 1 - F_{j+1}[b] + F_{j+1}[a] \\ P_{j+1}^U(t)\big|_a^b &= 1 - F_j[b] + F_j[a] \\ P_j^{\overline{U}}(t)\big|_a^b &= F_{j+1}[b] - F_{j+1}[a] \\ P_{j+1}^{\overline{U}}(t)\big|_a^b &= F_j[b] - F_j[a] \end{aligned} \tag{3.31}$$

2ª Aproximación.
Influencia de las $f(t)$ anteriores y posteriores en el cálculo de la zona de influencia conjunta de $f_j(t)$ y $f_{j+1}(t)$

La segunda aproximación consiste en añadir al cálculo del tiempo disponible realizado mediante la primera aproximación, la contribución media al tiempo disponible de todas las $f(t)$ cuyo valor no es nulo en una sección (tiempo entre el inicio de un *ontime* y el inicio del siguiente).

Para ello, tal y como se ha justificado previamente, la estrategia a seguir será evaluar la contribución que, en media, una $f_j(t)$ aporta al tiempo disponible cuando tiene un valor no nulo en la zona de influencia conjunta (sección) de alguna otra pareja $f_k(t)$ y $f_{k+1}(t)$. Así, para el escenario genérico representado en la Fig. 3.3 se realizará, por un lado, el cálculo de la contribución al tiempo disponible generada por $f_j(t)$ en los tramos F y G, y por otro, se evaluará la contribución debida a $f_{j+1}(t)$ en los tramos A y B. Para la contribución media que una $f(t)$ genera en cada tramo $i \in \{A, B, F, G\}$, se utilizará la notación $t_D^{\alpha}|_i$.

Nótese que existe una correspondencia definida entre los tramos A, B, F y G de $f_j(t)$ y $f_{j+1}(t)$ y los tramos C, D y E asociados a la zona de influencia conjunta de $f_k(t)$ y $f_{k+1}(t)$. De este modo, es equivalente considerar que una salida j se produce en su tramo F o que se ha producido en el tramo C de las salidas cuyas $f(t)$ son posteriores, es decir, $f_{j+1}(t)$ y $f_{j+2}(t)$. Cuando una salida $j+1$ sucede en el tramo B, es equivalente a considerar que ha sucedido en el tramo E de dos $f(t)$ anteriores $f_k(t)$ y $f_{k+1}(t)$ (sin tener que ser las inmediatamente anteriores). Por lo que respecta al tramo A, cuando la salida $j+1$ sucede en dicho tramo, es equivalente a considerar que aparece bien en el tramo C o en el D de dos $f(t)$ anteriores $f_k(t)$ y $f_{k+1}(t)$. El tramo E queda descartado porque está incluido en el caso en que $j+1$ sucede en su tramo B. Por último, cuando la salida j sucede en el tramo G, este hecho es equivalente a considerar que dicha salida se produce en cualquiera de los tramos C, D o E de una pareja de $f(t)$ posteriores.

Por consiguiente, dado que el factor $t_D^{\alpha}|_i$ debe considerar la contribución que supone una salida en el cálculo que previamente se hubiera realizado en la zona de influencia conjunta de $f_k(t)$ y $f_{k+1}(t)$, para su valor se tomará el promedio de dicho incremento. Sin embargo, el cálculo de este valor promedio no es sencillo, ya que hay que tener en cuenta multitud de posibles escenarios. Esto motiva la necesidad de realizar ciertas aproximaciones que simplifiquen su cómputo. En primer lugar, el promedio para $t_D^{\alpha}|_i$ se realizará en función de la distancia de superposición s existente entre la pareja de funciones de probabilidad de ocurrencia $f_k(t)$ y $f_{k+1}(t)$ entre las que se considera que se va a generar la salida j o $j+1$, según corresponda. Para el valor instantáneo de $t_D^{\alpha}|_i$, cuando se hace dependiente de dicha distancia de superposición, se utilizará la notación $t_D^{\alpha}(s)|_i$. Por tanto, el valor promedio del tiempo disponible adicional que se genera en un tramo $i \in \{A, B, F, G\}$ de las funciones $f_j(t)$ y $f_{j+1}(t)$ se calculará como:

$$t_D^{\alpha}\big|_i = \int_0^{\infty} t_D^{\alpha}(s)\big|_i \cdot g(s)ds \tag{3.32}$$

donde $g(s)$ es la función de probabilidad de la distribución de la distancia de superposición s.

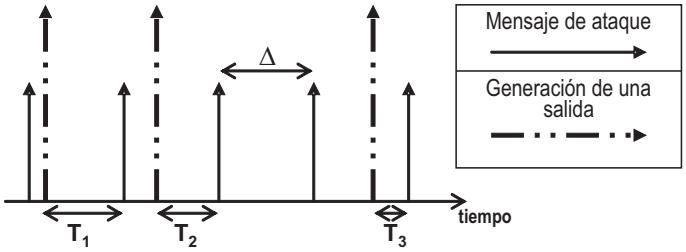


Fig. 3.4: Escenario de llegada de mensajes de ataque separados por una distancia Δ, donde el tiempo total disponible generado es $T_1 + T_2 + T_3$.

Para resolver la anterior expresión será necesario proponer el valor de $t_D^{\alpha}(s)|_i$ analizando los escenarios que aparecen en función de la distancia de superposición s de las dos funciones $f_k(t)$ y $f_{k+1}(t)$ genéricas entre las cuales aparece la salida considerada. Para ello, en primer lugar se realizará un análisis del tiempo disponible que generaría cualquier salida que se produjera en el escenario ejemplo de la Fig. 3.4. En éste se produce la llegada de numerosos mensajes de ataque (flechas continuas), todos ellos separados por un tiempo correspondiente al intervalo del periodo básico de ataque, Δ. Nótese que esto se produce cuando los periodos básicos de ataque estan separados precisamente un tiempo Δ. En la figura se pueden observar también varias salidas, representadas con flechas discontinuas, las cuales producirán un tiempo disponible igual a $T_1 + T_2 + T_3$.

En este escenario, el tiempo disponible adicional que generaría cualquier salida dependerá fundamentalmente de dos factores. En primer lugar, si la salida se produce en un intervalo donde no se produce ninguna otra, el tiempo disponible adicional será el tiempo que transcurre entre el instante en que ella se produce y la llegada del siguiente mensaje de ataque. Por otro lado, si la salida se produce de forma conjunta con otra en el mismo intervalo, el tiempo adicional que se generaría corresponde con el tiempo que tardaría en llegar otro mensaje de ataque, es decir, Δ. Nótese que, en este escenario, se puede decir que el tiempo disponible adicional que genera cada salida será, como máximo, el valor de Δ. Por otra parte, el mínimo tiempo disponible que se generará será aproximadamente cero, dado que la salida se puede producir justo antes de la llegada de un mensaje de ataque. Por ello, en este tipo de escenarios, un estimador del tiempo disponible promedio generado adicionalmente si se produjera una salida más (aparte de las consideradas en la primera aproximación) podría ser la media uniforme de los valores máximo y mínimo, es decir, $\Delta/2$.

Hay que señalar que el razonamiento anterior realizado para el escenario de la Fig. 3.4 se puede aplicar a todos aquellos escenarios en los que las distancias de superposición entre los periodos básicos de ataque hacen que éstos se encuentren próximos entre sí, al menos una distancia Δ. Por tanto, para todos los escenarios en los que, para las diferentes salidas consecutivas, se cumple la condición $s \leq t_{ontime} + \Delta$, se puede decir que cada salida que se produzca generará, en promedio, un tiempo disponible adicional de valor $\Delta/2$.

Por otro lado, considérese el escenario en el que una salida, dígase j, aparece en la zona de influencia conjunta de otras dos funciones de probabilidad de ocurrencia $f_k(t)$ y $f_{k+1}(t)$, anteriores o posteriores a j, y que están separadas por una distancia s cuyo valor

es suficientemente grande. En este caso, el tiempo adicional que se generará dependerá también de las circunstancias en las que la salida se produzca. Así, si la salida j ocurre en la fase *ontime* de la salida k, el tiempo disponible adicional se podría deducir como en el caso del escenario de ejemplo anterior, es decir, $\Delta/2$. Sin embargo, si se produjera en el tramo D de las funciones $f_k(t)$ y $f_{k+1}(t)$, y no coincidiera en una ventana de $\overline{RTT}$ con ninguna de las salidas k o $k+1$, el tiempo adicional sería precisamente $\overline{RTT}$. En caso de coincidir en el tiempo (en una ventana temporal $\overline{RTT}$) con alguna de estas dos salidas, el tiempo adicional sería inferior a $\overline{RTT}$, aunque difícil de cuantificar exactamente. Nótese que cuando la distancia s entre las curvas $f_k(t)$ y $f_{k+1}(t)$ es muy grande, la probabilidad de que coincidan en la ventana temporal la salida j y las k o $k+1$ es muy baja. Por tanto, en esta situación se podría aproximar el valor de $t_D^{\alpha}(s)|_i$ por $\overline{RTT}$. Sin embargo, conforme s disminuye, el tiempo disponible esperado disminuirá también, debido a que se pueden producir más solapamientos entre las salidas j y las k o $k+1$. Esta disminución será progresiva hasta llegar al escenario en que $s \leq t_{ontime} + \Delta$, en el que se podrá considerar de nuevo el valor $\Delta/2$.

Tomando en consideración los dos escenarios de ejemplo (s grande y s pequeña) se puede proponer una función para $t_D^{\alpha}(s)|_i$. Para ello, se hará un análisis diferenciado para los diferentes tramos que hay que considerar en ésta denominada "segunda aproximación":

- *Tramo F*:

  En el caso del tramo F, dado que se conoce positivamente que la salida j, si se produce, lo hará en el periodo *ontime* de la salida $j+1$, se aproximará tomando:

$$t_D^{\alpha}(s)\big|_F = \frac{\Delta}{2}, \quad \forall s \in \Re^+ \tag{3.33}$$

- *Tramos A y G*:

  Para los tramos A y G, dado que éstos considerarán la ocurrencia de las salidas j o $j+1$ en los tramos C, D y E de las funciones $f_k(t)$ y $f_{k+1}(t)$ (ver nota[4]), para $t_D^{\alpha}(s)|_{i\in\{A,G\}}$ se propone la función representada en la Fig. 3.5, en la que se aprecia que el límite a partir del cual se toma el valor $\overline{RTT}$ es $s \geq \gamma \cdot \overline{RTT}$ (con $\gamma \in \Re^+$ un valor a determinar experimentalmente contrastando los valores obtenidos a partir del modelo con los que se extraen en un entorno de simulación[5]). El valor para s pequeñas ($s \leq t_{ontime} + \Delta$) será $\Delta/2$ y, para los valores de $t_D^{\alpha}(s)$ en el intervalo $(t_{ontime} + \Delta, \gamma \cdot \overline{RTT})$, dado que la función debe ser creciente conforme s aumenta, se propone una interpolación lineal, resultando:

$$t_D^{\alpha}(s)\big|_{i\in\{A,G\}} = \begin{cases} \Delta/2 & , \text{ si } s \leq t_{ontime} + \Delta \\ \chi \cdot [s - (t_{ontime} + \Delta)] + \frac{\Delta}{2} & , \text{ si } s \in (t_{ontime} + \Delta, \gamma \cdot \overline{RTT}) \\ \overline{RTT} & , \text{ si } s \geq \gamma \cdot \overline{RTT} \end{cases} \tag{3.34}$$

[4] Para mayor exactitud, hay que indicar que el tramo G de $f_j(t)$ y $f_{j+1}(t)$ sólo considera la ocurrencia de la salida j en el tramo E de las funciones $f_k(t)$ y $f_{k+1}(t)$.

[5] Se ha comprobado experimentalmente que el valor $\gamma = 4$ supone una buena aproximación para este parámetro.

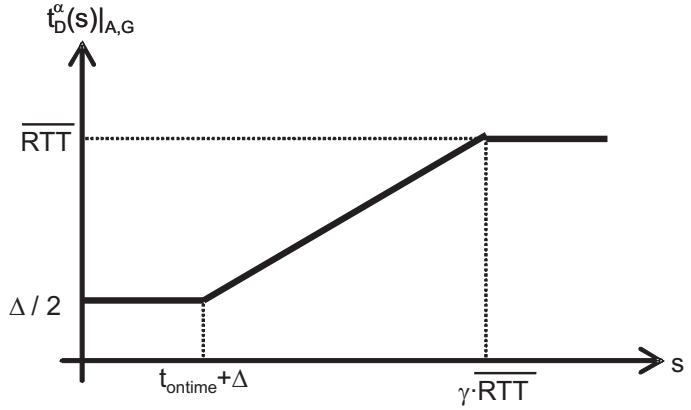


Fig. 3.5: Representación gráfica de la función $t_D^{\alpha}(s)\big|_{i\in\{A,G\}}$.

con

$$\chi = \frac{\overline{RTT} - \frac{\Delta}{2}}{\gamma \cdot \overline{RTT} - (t_{ontime} + \Delta)} \tag{3.35}$$

donde se ha asumido que se cumple la condición $t_{ontime} + \Delta \leq \gamma \cdot \overline{RTT}$. En caso contrario, la función siempre valdría $\Delta/2$. Además, también se supone que $\overline{RTT} > \Delta/2$. Esta suposición es realista, dado que el tiempo entre mensajes de ataque, Δ, debe ser pequeño. Aun así, en caso de que esto no se cumpla, el valor de la función sería $\overline{RTT}$ para todos los valores de s.

- *Tramo B*:

  El caso del tramo B es ligeramente diferente al de los tramos A y G, ya que se sabe que cuando se produce la salida j, independientemente de la distancia s entre las funciones $f_k(t)$ y $f_{k+1}(t)$, el tiempo transcurrido hasta la llegada de un mensaje de ataque (punto de cálculo a_1) será típicamente menor que $\overline{RTT}$. En este caso, para la función $t_D^{\alpha}(s)|_B$ se tomará como valor máximo el de $\overline{RTT}/2$ cuando se cumpla que $s \geq \overline{RTT}$. Este valor se ha tomado porque aproxima al valor medio que se obtendría para el tiempo disponible instantáneo en los distintos escenarios posibles cuando no existe ningún ciclo *ontime* en el tramo B (condición $s \geq \overline{RTT}$). Para el resto de valores de s se interpolará como en el caso de los tramos A y G. En la Fig. 3.6 se representa la función $t_D^{\alpha}(s)|_B$, cuya expresión es:

$$t_D^{\alpha}(s)\big|_B = \begin{cases} \Delta/2 & \text{si } s \leq t_{ontime} + \Delta \\ \frac{\frac{\overline{RTT}}{2} - \frac{\Delta}{2}}{\overline{RTT} - (t_{ontime} + \Delta)} \cdot [s - (t_{ontime} + \Delta)] + \frac{\Delta}{2} & \text{si } s \in (t_{ontime} + \Delta, \overline{RTT}) \\ \overline{RTT}/2 & \text{si } s \geq \overline{RTT} \end{cases} \tag{3.36}$$

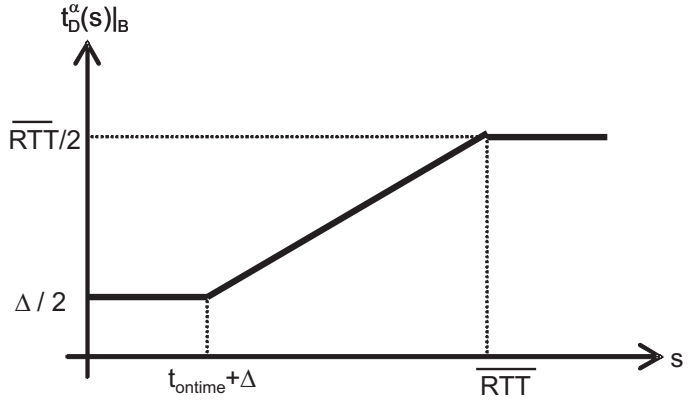


Fig. 3.6: Representación gráfica de la función $t_D^{\alpha}(s)\big|_B$.

donde también en este caso se ha asumido que se cumple la condición $t_{ontime} + \Delta \leq \overline{RTT}$. En caso contrario, la función siempre valdría $\Delta/2$. Además, también se supone que $\overline{RTT} > \Delta$. En caso de no ser así, el valor de la función sería $\overline{RTT}/2$ para todos los valores de s.

En resumen, para evaluar el tiempo disponible en los intervalos correspondientes, por un lado, a la zona de influencia de $f_j(t)$, y por otro, a la de $f_{j+1}(t)$ se debe, en primer lugar, calcular el factor $t_D^{\alpha}|_i$ para cada tramo según la Expresión (3.32), y posteriormente, considerar la probabilidad de que dicho valor tenga que ser aplicado. Esta probabilidad, para un intervalo genérico (a, b) situado en la zona de influencia de $f_j(t)$, corresponderá precisamente con $P_j^U(t)|_a^b$, por lo que:

$$T_D^i(s)\big|_{i\in\{F,G\}} = \int_a^b t_D^{\alpha}\big|_{i\in\{F,G\}} \cdot P_j^U(t)\big|_a^b \cdot f_j(t)dt \tag{3.37}$$

mientras que para los tramos A y B, situados en la zona de influencia de $f_{j+1}(t)$, la probabilidad a considerar será $P_{j+1}^U(t)|_a^b$, de modo que se obtiene:

$$T_D^i(s)\big|_{i\in\{A,B\}} = \int_a^b t_D^{\alpha}\big|_{i\in\{A,B\}} \cdot P_{j+1}^U(t)\big|_a^b \cdot f_{j+1}(t)dt \tag{3.38}$$

3ª Aproximación.
Inclusión de la aparición de mensajes de ataque adicionales en el cálculo de los términos $T_D^i(s)$.

Cada vez que se produce una salida y el mensaje de respuesta llega al atacante, según la especificación del ataque, el atacante envía un nuevo mensaje de ataque con destino el servidor. Este mensaje llegará, en media, $\overline{RTT}$ segundos más tarde del momento en que

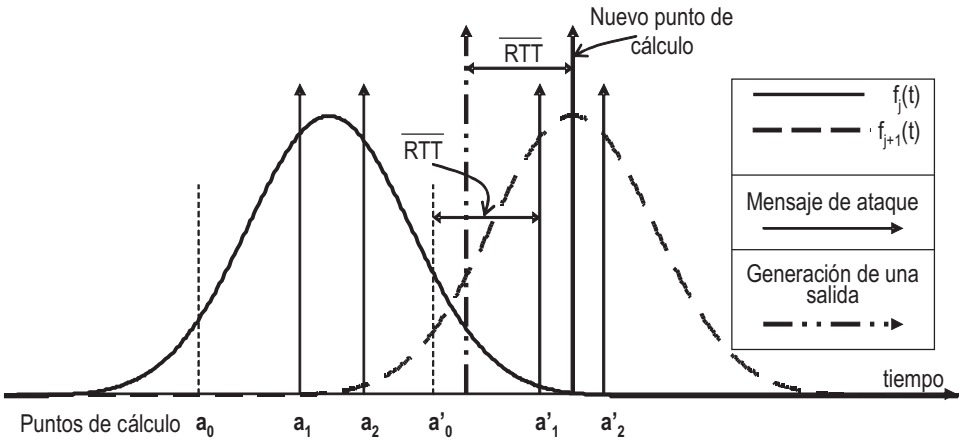


Fig. 3.7: Escenario de aparición de un mensaje adicional de ataque en un intervalo debido a la generación de una salida en un intervalo anterior.

se generó la salida. En las dos aproximaciones anteriormente presentadas para el cálculo de los términos $T_D^i(s)$ solamente se han tenido en cuenta estos mensajes de ataque cuando son respuesta a una salida que ha tenido lugar dentro del intervalo considerado para el cálculo. Ahora bien, puede suceder que, una vez fijado un intervalo para realizar el cálculo de $T_D^i(s)$, aparezcan en el mismo mensajes de ataque adicionales generados como respuesta a salidas que se produjeron en intervalos de tiempo anteriores al considerado. La Fig. 3.7 muestra gráficamente cómo tiene lugar este escenario. En ella se observa que la generación de una salida en un intervalo produce, $\overline{RTT}$ segundos más tarde y en el intervalo siguiente, la aparición de un mensaje de ataque adicional.

La aparición de estos mensajes de ataque adicionales afectará, según el escenario que se considere, al valor instantáneo del tiempo disponible correspondiente, $t_D^U(t)|_b$, $t_D^{\overline{U}}|_a^b$ o $t_D^{\alpha}|_i$. Por tanto, para incluir el efecto de los mensajes de ataque adicionales será necesario modificar estas expresiones.

Es evidente que la casuística con la que se produce este escenario es amplia. En general, el número de mensajes de ataque adicionales que aparecerán en un intervalo genérico (a, b) dependerá de los instantes en que se produzcan las salidas anteriores a dicho tramo y, más concretamente, del número de salidas que se generen en el intervalo $(a-\overline{RTT}, b-\overline{RTT})$.

Para tener en cuenta en el modelo la aparición de este tipo de escenarios se harán dos aproximaciones. La primera consiste en considerar que el número de salidas que se produce en un intervalo dado viene determinado por la generación media de salidas en el servidor para dicho intervalo. Dado que la media del tiempo entre salidas viene determinada por la Expresión (2.12), el número medio de éstas en el intervalo (a, b) considerado será:

$$\frac{b-a}{\overline{\tau_s}} = \frac{(b-a)\cdot N_s}{\overline{T_s}} \tag{3.39}$$

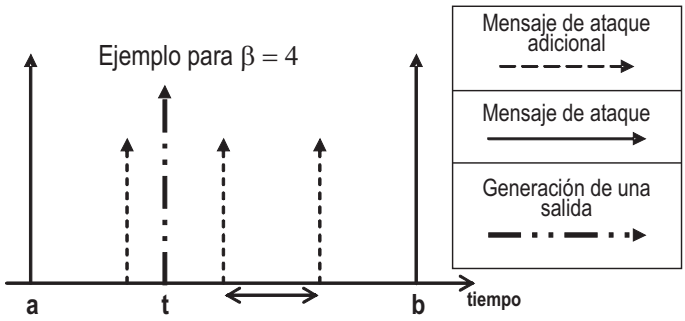


Fig. 3.8: Escenario de aparición de mensajes de ataque adicionales. Determinación del instante de llegada del siguiente mensaje de ataque para una salida que se produce en t.

La aparición de cada mensaje de ataque adicional generará, dentro del intervalo, un nuevo subintervalo. Dado que el número mensajes de ataque adicionales, obtenido de la Expresión (3.39), no es un número entero, el número de subintervalos en los que quedará dividido el intervalo genérico (a, b), β, será al menos uno más que el número menor posible de mensajes de ataque adicionales:

$$\beta = floor\left[\frac{(b-a)\cdot N_s}{\overline{T_s}}\right] + 1 \tag{3.40}$$

donde el operador *floor* indica el entero inferior más cercano.

La segunda aproximación a realizar pasa por considerar que las β divisiones del intervalo son iguales, es decir, que los $\beta - 1$ mensajes de ataque adicionales llegan equiespaciados. Esta suposición hace que sea conocida la estructura de los nuevos subintervalos dentro del intervalo que se considera, lo que permitirá modificar los valores instantáneos del tiempo disponible generado en los diferentes escenarios.

Consideraremos en primer lugar cómo quedaría modificado el valor de $t_D^U(t)|_b$, anteriormente propuesto en la Expresión (3.23). Cuando un intervalo genérico (a, b) queda dividido en β subintervalos iguales, el tiempo $t_D^U(t)|_b$, dependiente de t, dependerá del subintervalo en que se sitúe t. De esta forma, el mensaje de ataque adicional que llegará tras el instante t (ver Fig. 3.8) estará situado en:

$$a + \rho \cdot ceil\left[\frac{t-a}{\rho}\right]$$

donde $\rho = \frac{b-a}{\beta}$, y el operador *ceil* genera el entero superior del argumento dado como parámetro.

Por tanto, el valor de $t_D^U(t)|_b$ cuando se consideran β divisiones en el intervalo, lo que se notará como $t_D^U(t)|_b^\beta$, se debe calcular como:

$$t_D^U(t)\big|_b^\beta = min\left(a + \rho \cdot ceil\left[\frac{t-a}{\rho}\right] - t, \overline{RTT}\right) \tag{3.41}$$

Con respecto al valor de $t_D^{\overline{U}}|_a^b$, dado que se ha propuesto un cálculo de modo que este tiempo es independiente de t, el valor modificado cuando se considera la aparición de mensajes de ataque adicionales, $t_D^{\overline{U}}\big|_a^b(\beta)$, será el anterior dividido por el número de intervalos nuevos que aparecen:

$$t_D^{\overline{U}}\big|_a^b(\beta) = \frac{t_D^{\overline{U}}\big|_a^b}{\beta} \tag{3.42}$$

Todo este anterior razonamiento es también válido para los diferentes factores definidos en la "segunda aproximación", $t_D^\alpha|_i$, por lo que éstos quedarán modificados de la misma forma:

$$t_D^\alpha\big|_{i\in\{A,B,F,G\}}^\beta = \frac{t_D^\alpha\big|_{i\in\{A,B,F,G\}}}{\beta} \tag{3.43}$$

Por otro lado, dado que el número de mensajes de ataque adicionales, calculado con la Expresión (3.39), no es un valor entero, se puede deducir que el número de subintervalos podrá ser β o bien $\beta+1$, dependiendo de si llegan $\beta-1$ o β mensajes de ataque adicionales. De este modo, diremos que la probabilidad de que aparezcan β subintervalos dentro del intervalo considerado, $P(\beta)$, será:

$$P(\beta) = \beta - \frac{(b-a)\cdot N_s}{\overline{T_s}} \tag{3.44}$$

mientras que la probabilidad de que aparezcan $\beta+1$ subintervalos, $P(\beta+1)$, será su complementaria:

$$P(\beta+1) = 1 - P(\beta) = 1 - \beta + \frac{(b-a)\cdot N_s}{\overline{T_s}} \tag{3.45}$$

Finalmente, a la hora de introducir los efectos considerados en esta "tercera aproximación" en los cálculos propuestos para $T_D^i(s)$ según las anteriores aproximaciones, hay que tener en cuenta el hecho de que pueden aparecer β o $\beta+1$ nuevos subintervalos, por lo que hay que extender el cálculo propuesto considerando ambas probabilidades. Como ejemplo se muestra el modo en que habría que adaptar la Expresión (3.37), correspondiente a los tramos F y G:

$$T_D^i(s)\big|_{i\in\{F,G\}} = \int_a^b \frac{t_D^\alpha\big|_{i\in\{F,G\}}}{\beta}\cdot P_j^U(t)\big|_a^b \cdot P(\beta)\cdot f_j(t)dt + \\ + \int_a^b \frac{t_D^\alpha\big|_{i\in\{F,G\}}}{\beta+1}\cdot P_j^U(t)\big|_a^b \cdot P(\beta+1)\cdot f_j(t)dt \qquad (3.46)$$

3.4.3 Conclusiones sobre los modelos para el porcentaje de tiempo disponible

El ataque a baja tasa contra servidores es un proceso que dispone de ciertos parámetros sintonizables, como t_{ontime} y el *intervalo*, Δ, para poder adaptar el ataque a las características del servidor y de la red de acceso al mismo. Más concretamente, los parámetros del servidor que influyen en la eficiencia del ataque son la varianza y la media del tiempo de servicio, $var[T_s]$ y $\overline{T_s}$, y el número de elementos de procesamiento existentes, N_s. Estos parámetros determinarán la distribución probabilística de los tiempos entre salidas y, por consiguiente, también la distribución de la distancia de superposición, $g(s)$. Adicionalmente, como característica de la red de acceso, hay que señalar la representación del tiempo de ida y vuelta entre el atacante y el servidor víctima, RTT.

La aparición de tantos factores en el comportamiento final del ataque hace difícil evaluar cómo va a ser la dependencia entre ellos y su influencia en el resultado final. El establecimiento del modelo matemático permite efectuar una evaluación cuantitativa de la forma en la que los diferentes parámetros que modelan tanto el servidor como el ataque influyen en el rendimiento global del proceso.

Es necesario, por tanto, aclarar la influencia en el porcentaje del tiempo disponible, a través del modelo matemático, de los diferentes parámetros que caracterizan tanto al ataque como al servidor y a la red de transporte. De esta forma tenemos:

- *Parámetros de diseño del ataque*:

  Los parámetros que permiten diseñar la configuración del ataque son, fundamentalmente, el *intervalo* Δ y t_{ontime}. El parámetro $t_{offtime}$ se deduce de ellos dos. Estos parámetros de configuración determinan en los modelos propuestos la posición que ocupan los puntos de cálculo en cada caso.

- *Parámetros que caracterizan al servidor y a la red de transporte*:

  - Tiempo de ida y vuelta, RTT:
    En los modelos matemáticos, el valor medio del tiempo de ida y vuelta entre el atacante y el servidor víctima del ataque influye en tres aspectos diferentes. En primer lugar, en las expresiones que permiten calcular el valor instantáneo del tiempo disponible en los diferentes escenarios, $t_D^U(t)|_b$, $t_D^{\overline{U}}|_a^b$ y $t_D^\alpha|_i$. Además, también el valor de $\overline{RTT}$ influye en la posición de los puntos de cálculo a_0 y a_0'. Finalmente, el valor de las probabilidades para los escenarios U y $\overline{U}$, es decir, de los factores $P_j^U(t)|_a^b$ y $P_j^{\overline{U}}(t)|_a^b$ está influenciado por $\overline{RTT}$.

Por otro lado, la varianza del tiempo de ida y vuelta influye en la caracterización de las funciones de probabilidad de ocurrencia de las salidas, $f_j(t)$ y $f_{j+1}(t)$, según la Expresión (3.5).

– Tiempo de servicio, T_s:

Tanto la información contenida en la media del tiempo de servicio como en su varianza están reflejadas en los modelos matemáticos mediante las funciones de probabilidad de ocurrencia $f_j(t)$ y $f_{j+1}(t)$.

A lo largo de la discusión del modelo, aunque se haya considerado en todos los ejemplos una distribución normal, no se ha impuesto ninguna restricción al tipo de distribución de las funciones de probabilidad de ocurrencia de las salidas, $f_j(t)$ y $f_{j+1}(t)$. Sin embargo, como se justificará en capítulos posteriores, en la ejecución del ataque se utilizarán mensajes de ataque idénticos entre sí a efectos del tiempo de servicio que precisan del servidor, por lo que se podría aplicar la Expresión (2.10) (distribución normal) para dichas funciones de probabilidad.

Además de las distribuciones de probabilidad de ocurrencia de las salidas, el tiempo de servicio también afectará a la distribución de la distancia de superposición, $g(s)$, debido a su influencia en el valor medio de dicha distribución –Expresión (3.5)–.

– Número de elementos de procesamiento en el servidor, N_s:

El número de elementos de procesamiento que se ejecutan en el servidor influye también en la distribución de la distancia de superposición, $g(s)$, dado que afecta a su valor medio según se recoge en la Expresión (2.12).

El modelo matemático para el cálculo del porcentaje de tiempo disponible permite, además de la estimación del tiempo disponible generado tramo a tramo, analizar el valor que toma dicho tiempo en función de la distancia de superposición. La Fig. 3.9 muestra el valor obtenido para $T_D(s)$ en función de la distancia de superposición para un ataque ejemplo, cuyas características se indican en la Tabla 3.2. Para la evaluación de los valores representados en ella se han utilizado las expresiones deducidas para cada uno de los tramos aplicando las tres aproximaciones presentadas, las cuales vienen detalladas en el Apéndice 1 del presente capítulo. En dicha figura se puede apreciar que el valor del tiempo disponible generado es menor para valores pequeños de la distancia de superposición. Se puede constatar mediante la resolución de las expresiones del tiempo disponible que este comportamiento es generalizable a todas las configuraciones escogidas para el ataque, así como para las diferentes características del servidor y la red de transporte. Esto permite concluir que el fenómeno de la superposición mejora la eficiencia del ataque, dado que el tiempo disponible generado será menor. Esta afirmación era previsible, ya que se puede intuir que, cuando los periodos de ataque están próximos en el tiempo, éstos interaccionan de modo que unos capturan las salidas correspondientes a otros, reduciéndose el efecto negativo de la variabilidad sobre la eficiencia.

Además, se puede observar también en la Fig 3.9 que el valor del tiempo disponible generado tiene un valor asintótico cuando crece la distancia de superposición. Se puede constatar el hecho de que el valor obtenido a través del modelo para sistemas con superposición cuando s es suficientemente grande, coincide con el proporcionado por el correspondiente

Parámetros del servidor	Parámetros del ataque
$T_s = \mathcal{N}(12\ s; 0.03)$	$t_{offtime} = 11.9\ s$
$RTT = \mathcal{N}(0.5\ s; 0.06)$	$t_{ontime} = 0.2\ s$
$N_s = 4$	$\Delta = 0.1\ s$

Tabla 3.2: Valores de configuración usados en el escenario de ejemplo mostrado en la Fig. 3.9 para la evaluación de $T_D(s)$.

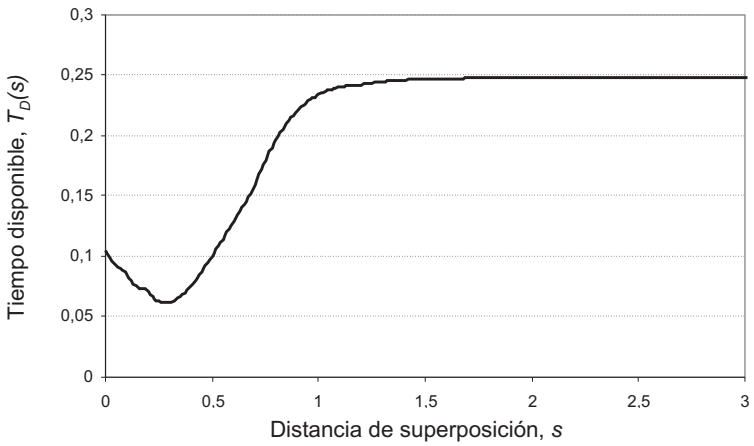


Fig. 3.9: Tiempo disponible para todos los tramos para el ejemplo de ataque con la configuración mostrada en la Tabla 3.2.

a sistemas sin superposición. Se puede comprobar analíticamente que las expresiones de ambos modelos coinciden para valores suficientemente elevados de s, ya que las influencias mutuas entre las funciones de probabilidad $f_j(t)$ y $f_{j+1}(t)$ serán nulas cuando éstas estén suficientemente alejadas entre sí. Esto implica que el modelo para sistemas sin superposición no es más que una particularización del modelo genérico presentado para sistemas con superposición.

3.5 Modelo matemático para la disponibilidad

Cuando se calcula el porcentaje de tiempo disponible, según el modelo matemático anteriormente propuesto, se considera que al servidor solamente le llega tráfico procedente del atacante. A la hora de proponer un modelo para la disponibilidad D, hay que considerar también la existencia de tráfico procedente de usuarios legítimos, el cual llegará al servidor agregado con el tráfico de ataque.

El modelo propuesto en este trabajo estará basado en los modelos de cálculo de tiempo disponible ya presentados. Dado que los modelos que estiman el tiempo disponible para los sistemas con superposición y sin superposición convergen cuando el valor de la distancia de superposición es suficientemente grande, el modelo que se propondrá para D será también función de s y, por tanto, aplicable a un sistema genérico, tanto monoproceso como concurrente. Por tanto, a lo largo del desarrollo del modelo, se considerará como base de estudio el escenario representado en el diagrama de la Fig. 3.3.

Tal y como se estableció en el modelo del tráfico de usuario propuesto en el Capítulo 2, la recepción de mensajes de usuario se representa mediante un proceso de Poisson. Por tanto, la probabilidad de recibir un mensaje de usuario durante un periodo de observación T viene dada por la función de distribución exponencial:

$$H(T) = 1 - e^{-\lambda T} \tag{3.47}$$

donde $\lambda = 1/T_a$ es la tasa de llegada de mensajes procedente de todos los posibles usuarios del sistema servidor.

El paso previo a calcular D es determinar la probabilidad de que, en un servidor víctima de un ataque DoS a baja tasa con una cierta configuración, un usuario realice la captura de una posición en alguna cola de servicio. Esta probabilidad, para la que se usará la notación P_u, puede ser calculada utilizando la Expresión (3.47) al considerar que el tiempo de observación está en función del tiempo disponible generado durante los periodos de ataque, es decir, haciendo $T = \hat{T}_D$ –ver Expresión (3.13)–. Ahora bien, dado que la Expresión (3.47) supone que el tiempo de observación T es continuo, es decir, no está formado por diferentes fragmentos separados, será más exacto realizar el cálculo de P_u considerando los diferentes tramos de cálculo del tiempo disponible. Es por esto que, en un primer paso, se calcula P_u cuando depende de la distancia de superposición s entre las funciones de probabilidad de ocurrencia genéricas $f_j(t)$ y $f_{j+1}(t)$, de modo que será dependiente de la variable s de esta forma:

$$P_u(s) = \sum_{i \in \mathcal{I}} \left(1 - e^{-\lambda T_D^i(s)}\right) \tag{3.48}$$

El modo de obtener la probabilidad de que se realice una captura por parte de un usuario es a través de la estimación del valor esperado de $P_u(s)$ según la distribución de la distancia de superposición, es decir, $g(s)$:

$$P_u = \int_0^\infty P_u(s) \cdot g(s) ds \tag{3.49}$$

Finalmente, para un periodo de observación genérico T de un ataque, en el que se producen C capturas, la disponibilidad D viene dada por el porcentaje entre el número de capturas realizadas por el usuario, $P_u \cdot C$, y el número de mensajes enviados por los usuarios, $\lambda \cdot T$:

$$D = 100 \cdot \frac{P_u \cdot C}{\lambda \cdot T} \tag{3.50}$$

Ahora bien, en el cálculo de la probabilidad P_u, el valor aportado por el término $T_D^i(s)$ y calculado según se ha propuesto en el Apartado 3.4.2, no considera la influencia del tráfico de los usuarios en el sistema. En efecto, dado que el tiempo disponible es una medida en ausencia de tráfico de usuario, en todo el desarrollo del modelo para T_D se ha supuesto que no existe este tipo de tráfico. Ahora bien, para el cálculo de la disponibilidad será necesario incluir el efecto que dicho tráfico tendrá sobre el modelo para el tiempo disponible.

Cuando existe tráfico de usuario, el cual llega a una tasa λ al servidor, se puede considerar que cada mensaje entrante tiene el mismo efecto que un mensaje de ataque, es decir, ocupará una posición en el servidor si existe alguna libre o será descartado en caso contrario. Por tanto, se puede considerar que la tasa de mensajes de ataque se incrementa según λ.

Por otro lado, durante un tiempo de observación, T, en el que se producen C capturas en el servidor, el hecho de que algunas de ellas, y más concretamente $P_u \cdot C$, sean realizadas por el usuario afecta también al modelo debido a que cada vez que se produzca una de estas salidas, a diferencia del comportamiento del atacante, el usuario no responderá con un nuevo mensaje ante su recepción. Esto supone que existirá una disminución en el número de mensajes de ataque que se reciban en el servidor. Si en ausencia de tráfico de usuario la tasa de mensajes de respuesta coincide con la tasa de generación de salidas del servidor ($N_s/\overline{T_s}$), dicha tasa quedará reducida debido a la inclusión de dicho tráfico en el modelo y tomará el valor:

$$\frac{N_s}{\overline{T_s}} \cdot (1 - P_u)$$

Se pueden incluir en el modelo las dos variaciones provocadas por la inclusión del tráfico de usuario anteriormente mencionadas, y ello a través del parámetro que controla el escenario de aparición de mensajes de ataque adicionales, β. En efecto, a la hora de incluir el efecto del tráfico de usuario se considerará que, en un intervalo genérico (a, b), el número de mensajes de ataque generados como respuesta a salidas (incluyendo los procedentes de los usuarios) que se reciben en el servidor será:

$$\left[\frac{N_s}{\overline{T_s}} \cdot (1 - P_u) + \lambda\right] \cdot (b - a)$$

por lo que el parámetro β –Expresión (3.40)–, es decir, el número de subintervalos en los que quedaría dividido el intervalo (a, b) quedaría modificado así:

$$\beta = floor\left\{\left[\frac{N_s}{\overline{T_s}} \cdot (1 - P_u) + \lambda\right] \cdot (b - a)\right\} + 1 \tag{3.51}$$

y su probabilidad asociada –Expresión (3.44)– también sería modificada:

$$P(\beta) = \beta - \left[\frac{N_s}{\overline{T_s}} \cdot (1 - P_u) + \lambda\right] \cdot (b - a) \tag{3.52}$$

En resumen, el modelo que permite el cálculo de la disponibilidad, D, utiliza las Expresiones (3.48), (3.49) y (3.50), pero ha de tenerse en cuenta que, a la hora de estimar el parámetro $T_D^i(s)$ hay que utilizar los valores de β y $P(\beta)$ aportados por las Expresiones (3.51) y (3.52).

Es importante destacar que, así realizado, el cálculo de los valores de P_u y β –y por ende $T_D^i(s)$–, se hace de forma recursiva, ya que existe una dependencia cruzada de ambos valores. Aunque esto podría resultar en una inestabilidad del modelo, en todos los cálculos realizados, el valor de ambas expresiones converge en un número pequeño de iteraciones, por lo que no se ha detectado problema alguno en este hecho.

3.6 Modelo matemático para la sobrecarga

La sobrecarga, S, está definida como la ratio, en porcentaje, entre la tasa de tráfico generada por el atacante y la máxima tasa de tráfico aceptada por el servidor.

El cálculo de la sobrecarga producida por un ataque se lleva a cabo tomando en cuenta el número de mensajes de ataque generados por el atacante durante un periodo de observación. Cuando el periodo de observación consiste en un periodo básico de ataque, el máximo número de mensajes aceptados por cada elemento de procesamiento en el servidor es de uno, ya que cada periodo de ataque está centrado en la captura de una salida. Por tanto, cuando se considera el periodo básico de ataque como periodo de observación, si todas las salidas están atacadas por un periodo básico de ataque, la sobrecarga vendrá dada por el número de mensajes de ataque emitidos.

Dos factores contribuyen en la generación de mensajes de ataque durante el periodo básico de ataque. En primer lugar, el periodo de actividad, t_{ontime}, durante el cual se generan N_p mensajes –Expresión (3.2)–, y en segundo término, el mensaje que es enviado como respuesta a la recepción de una salida procedente del servidor.

Para este modelo se asume que el atacante no recibe la salida antes o durante t_{ontime}. Esto es equivalente a decir que el periodo *ontime* no se interrumpe. Por ello, el número de paquetes generados durante esta fase corresponde a N_p.

Con respecto al mensaje generado como respuesta a la recepción de una salida, hay que tener en cuenta que no todas ellas son enviadas al atacante y que, por tanto, no siempre se generará este mensaje. El porcentaje de salidas no enviadas hacia el atacante se derivará del factor P_u.

Cuando se considera un tiempo de observación correspondiente a un periodo básico de ataque, tal y como se ha propuesto, el cálculo de la sobrecarga se simplifica. Esto es debido a que, por un lado, la tasa del servidor en dicho periodo es sólo de una salida, y por otro, la tasa de mensajes de ataque viene dada por N_p y por el número de mensajes de respuesta del atacante a las salidas, es decir, $1 - P_u$ en cada periodo básico de ataque. Por

ello, la sobrecarga que se obtiene para un periodo de observación igual a un periodo básico de ataque viene dada por la siguiente expresión:

$$S = 100 \cdot \left(floor \left[\frac{t_{ontime}}{\Delta} \right] + 1 + (1 - P_u) \right) \tag{3.53}$$

El caso en que se considere un periodo de observación genérico de duración T, hay que tener en cuenta que, de las C salidas que se producirán en dicho periodo, es posible que el atacante trate de capturar solamente un subconjunto de ellas, C'. Por consiguiente, la sobrecarga se calculará como:

$$S = 100 \cdot \frac{C'}{C} \cdot \left(floor \left[\frac{t_{ontime}}{\Delta} \right] + 1 + (1 - P_u) \right) \tag{3.54}$$

3.7 Conclusiones del capítulo

En este capítulo se han presentado los fundamentos del ataque DoS a baja tasa contra servidores. Para ello, se ha formulado la estrategia a seguir por parte del atacante para infligir el ataque, se ha detallado la técnica a utilizar y, además, se ha comparado con la considerada en otro tipo de ataques también de baja tasa.

Con el fin de poder evaluar el rendimiento del ataque, se ha propuesto un conjunto de indicadores que aportan una métrica del comportamiento obtenido para una configuración específica de ataque. Si bien la evaluación de estos indicadores se puede realizar de forma empírica, existe la necesidad de realizar un alto número de experimentos para la evaluación de las múltiples configuraciones posibles de modo que se puedan extraer resultados aceptables.

Por ello, se ha desarrollado un modelo matemático que permite extraer conclusiones sobre el comportamiento de los diferentes indicadores en función de los parámetros de configuración del ataque, del servidor y también de la red de transporte que los comunica.

En este capítulo se han aportado, por tanto, las herramientas para poder evaluar el rendimiento de este tipo de ataques, si bien se ha dejado para capítulos posteriores la realización de dicha evaluación, particularizando el ataque para un determinado tipo o característica del servidor víctima.

Apéndice 1.

Cálculo del tiempo disponible en los diferentes tramos

En este apéndice se calculará el tiempo disponible que aparece en cada tramo de cálculo de los especificados en la Fig. 3.3, aportando las expresiones para los diferentes términos $T_D^i(s)$ que recogen la contribución al tiempo disponible en los diferentes tramos.

Adicionalmente, para cada uno de los tramos $i \in \mathcal{I}$, se realizará un estudio detallado de un ataque ejemplo que permita estimar la aportación al tiempo disponible generada en el tramo concreto. En este estudio, el valor del tiempo disponible es calculado cuando se produce una variación en la distancia de superposición s. Se ha elegido un escenario común como ejemplo para ilustrar la evolución de $T_D^i(s)$ en cada tramo. Los valores de configuración de este escenario son los indicados en la Tabla 3.2, y los resultados obtenidos se irán mostrando conforme se vayan aportando las expresiones.

Tramo A

El tramo A es el correspondiente al intervalo $(-\infty, a_0)$. En este tramo, que considera la influencia de la función de probabilidad de ocurrencia $f_{j+1}(t)$ en las secciones anteriores a $f_j(t)$, la influencia de ésta última se considera nula, debido a que el tramo A se encuentra ubicado en la zona de influencia de $f_{j+1}(t)$.

Para el tramo A, el valor instantáneo del tiempo disponible a considerar es $t_D^\alpha|_A$, dado por la Expresión (3.34). Con las consideraciones dadas en todo el Apartado 3.4.2, la expresión final para la contribución al tiempo disponible en el tramo A, $T_D^A(s)$, queda de la siguiente forma:

$$T_D^A(s) = \int_{-\infty}^{a_0} t_D^\alpha\big|_A \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_{j+1}^U(t)\big|_{-\infty}^{a_0} \cdot f_{j+1}(t)dt \tag{3.55}$$

donde para el cálculo del valor de β y su probabilidad, $P(\beta)$, dados por las Expresiones (3.40) y (3.44), se considerará que la distancia del intervalo es la del máximo valor que puede obtener el tiempo disponible, es decir, $b - a = \overline{RTT}$, por lo que:

$$\beta = floor \cdot \left[\frac{\overline{RTT} \cdot N_s}{\overline{T_s}} \right] + 1 \tag{3.56}$$

$$P(\beta) = \beta - \frac{\overline{RTT} \cdot N_s}{\overline{T_s}} \tag{3.57}$$

La Fig. 3.10 muestra los valores obtenidos para $T_D^A(s)$ en el ataque ejemplo cuyos valores de configuración están indicados en la Tabla 3.2, cuando se varía la distancia de

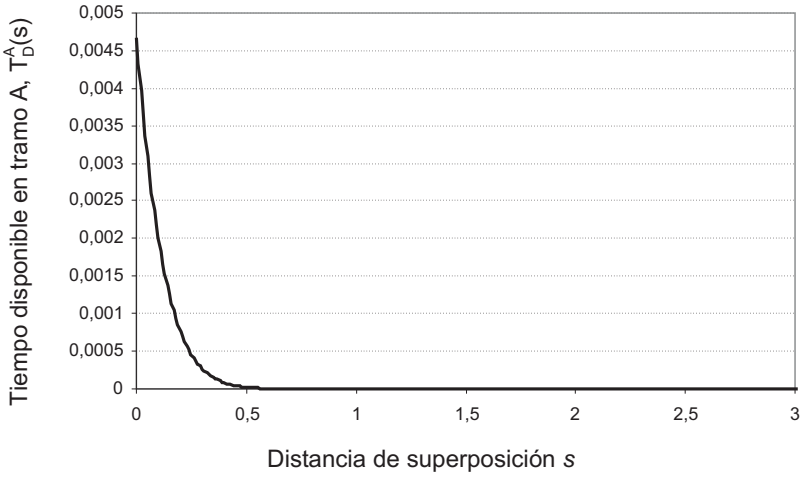


Fig. 3.10: Tiempo disponible para el tramo A, $T_D^A(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2.

superposición s. En la figura se puede apreciar que el máximo de $T_D^A(s)$ se alcanza cuando la distancia s es nula, ya que la aportación de la curva $f_{j+1}(t)$ en este tramo es máxima para este valor de s. Conforme la distancia crece, la influencia de $f_{j+1}(t)$ va haciéndose cada vez más pequeña, hasta ser nula con una distancia aproximada de $0,6\,s$.

Tramo B

El tramo B es el correspondiente al intervalo (a_0, a_1). Este tramo está dentro de la zona de influencia de $f_{j+1}(t)$, por lo que no se considera que se pueda producir una salida cuya función de probabilidad de ocurrencia es $f_j(t)$.

El tiempo disponible instantáneo que se generará en este intervalo corresponde con $t_D^\alpha|_B$, dado por la Expresión (3.36). Por tanto, la expresión final para el cálculo del tiempo disponible en este tramo es:

$$T_D^B(s) \;=\; \int_{a_0}^{a_1} t_D^\alpha\big|_B \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1}\right) \cdot P_{j+1}^U(t)\big|_{a_0}^{a_1} \cdot f_{j+1}(t)dt \qquad (3.58)$$

La Fig. 3.11 muestra los valores obtenidos de $T_D^B(s)$ en el ataque ejemplo, cuando se varía la distancia de superposición s. En ella se aprecia un comportamiento muy similar al experimentado en el tramo A, aunque la aportación de este tramo al tiempo disponible generado es bastante mayor que en B.

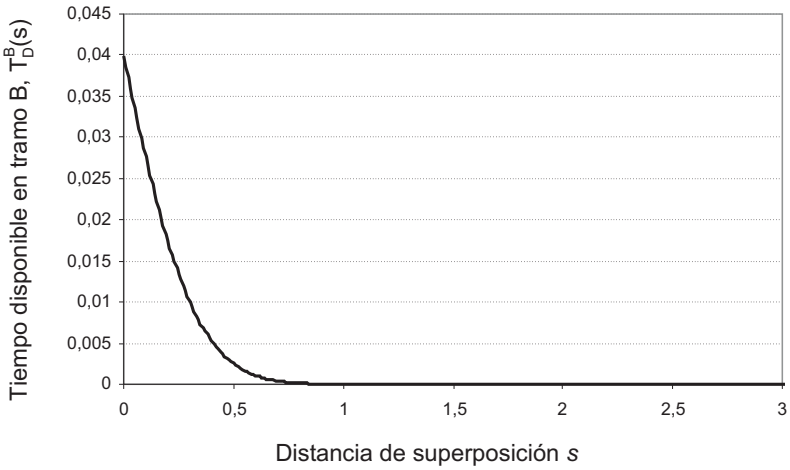


Fig. 3.11: Tiempo disponible para el tramo B, $T_D^B(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2.

Tramo C

El tramo C es el correspondiente al intervalo (a_1, φ_C). Al estar el tramo C en la zona de influencia conjunta de $f_j(t)$ y de $f_{j+1}(t)$, aparte de la aparición de mensajes de ataque adicionales, hay que considerar la ocurrencia tanto del escenario U como del $\overline{U}$. De este modo, para una mayor simplicidad al mostrar la expresión de cálculo del tiempo disponible para este tramo, $T_D^C(s)$, su cálculo se dividirá a su vez en la contribución realizada por los términos que consideran el escenario U, $T_D^C(s)|_U$, y aquellos que consideran el escenario $\overline{U}$, $T_D^C(s)|_{\overline{U}}$, de modo que:

$$T_D^C(s) = T_D^C(s)|_U + T_D^C(s)|_{\overline{U}} \tag{3.59}$$

Así pues, para la expresión que considera el escenario U hay que realizar una sumatoria que incluya a los diferentes intervalos dentro del tramo C, delimitados por la aparición de mensajes de ataque en el periodo *ontime*:

$$\begin{aligned} T_D^C(s)\big|_U \;=\; \sum_{i=2}^{\varphi_C} \Bigg[& \int_{a_{i-1}}^{a_i} \left[t_D^U(t)\big|_{a_i}^{\beta} \cdot P(\beta) + t_D^U(t)\big|_{a_i}^{\beta+1} \cdot P(\beta+1) \right] \cdot P_j^U(t)\big|_{a_{i-1}}^{a_i} \cdot f_j(t)dt + \\ & + \int_{a_{i-1}}^{a_i} \left[t_D^U(t)\big|_{a_i}^{\beta} \cdot P(\beta) + t_D^U(t)\big|_{a_i}^{\beta+1} \cdot P(\beta+1) \right] \cdot P_{j+1}^U(t)\big|_{a_{i-1}}^{a_i} \cdot f_{j+1}(t)dt \Bigg] \end{aligned} \tag{3.60}$$

Ahora, para el cálculo en el escenario $\overline{U}$ es necesario determinar previamente el valor de $t_D^{\overline{U}}|_a^b$ en los distintos intervalos del tramo. Para ello, como se ha justificado en el Apartado 3.4.2, se tomará el valor medio entre el máximo y el mínimo posibles. En este caso, hay que distinguir cuándo las dos salidas suceden en el último intervalo, al que nos referiremos como $(\varphi_{C-1}, \varphi_C)$, siendo φ_{C-1} el punto de cálculo anterior a φ_C, y cuándo ocurren en cualquier otro intervalo dentro del tramo C. En este último caso, el menor valor posible se genera cuando, en el intervalo (a_{i-1}, a_i), $a_i \neq \varphi_C$, las dos salidas ocurren muy próximas a a_i, pues el tiempo en llegar dos mensajes de ataque tendrá un valor $min[\Delta, \overline{RTT}]$. Por otro lado, si una salida se produce justo tras a_{i-1} y la otra justo antes de a_i, el valor del tiempo disponible será $2 \cdot min[\Delta, \overline{RTT}]$. Por tanto, en este caso, el valor que se adoptará será:

$$t_D^{\overline{U}}|_{a_{i-1}}^{a_i} = \frac{3 \cdot min[\Delta, \overline{RTT}]}{2}, \qquad i \in (2, ..., \varphi_{C-1}) \tag{3.61}$$

Cuando dentro del intervalo $(\varphi_{C-1}, \varphi_C)$ ocurren dos salidas, el menor valor posible del tiempo disponible corresponderá con la situación en la que ambas se producen justamente antes de φ_C, lo que hace que el valor del tiempo disponible sea el que transcurre desde φ_C hasta la llegada del siguiente mensaje de ataque. Curiosamente, el valor medio de este tiempo coincidirá con el valor $t_D^{\alpha}|_B$, calculado utilizando la función dada por la Expresión (3.36). Por otro lado, el valor máximo en este intervalo se obtendrá cuando una salida ocurra justo después de φ_{C-1} y la siguiente justo antes de φ_C. En este caso, el valor máximo será $min[\varphi_C - \varphi_{C-1}, \overline{RTT}] + t_D^{\alpha}|_B$. Haciendo la media de los valores máximo y mínimo se obtiene que:

$$t_D^{\overline{U}}|_{\varphi_{C-1}}^{\varphi_C} = \frac{min[\varphi_C - \varphi_{C-1}, \overline{RTT}]}{2} + t_D^{\alpha}|_B \tag{3.62}$$

La expresión que permite el cálculo del tiempo disponible en el tramo C cuando sucede el escenario $\overline{U}$ es la siguiente:

$$\begin{aligned} T_D^C(s)|_{\overline{U}} \;=\; & \sum_{i=2}^{\varphi_{C-1}} \left[\int_{a_{i-1}}^{a_i} t_D^{\overline{U}}|_{a_{i-1}}^{a_i} \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_j^{\overline{U}}(t)|_{a_{i-1}}^{a_i} \cdot f_j(t) dt \right] + \\ & \int_{\varphi_{C-1}}^{\varphi_C} t_D^{\overline{U}}|_{\varphi_{C-1}}^{\varphi_C} \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_j^{\overline{U}}(t)|_{\varphi_{C-1}}^{\varphi_C} \cdot f_j(t) dt \end{aligned} \tag{3.63}$$

La Fig. 3.12 muestra los valores obtenidos para $T_D^C(s)$ en el ataque con la configuración dada en la Tabla 3.2, cuando se varía la distancia de superposición s. En ella se puede apreciar, para distancias mayores a t_{ontime}, es decir, $s > 0.2$, que el comportamiento es decreciente asintótico, dado que las curvas de probabilidad de ocurrencia $f_j(t)$ y $f_{j+1}(t)$ se separan al aumentar la distancia s, disminuyendo así el valor del tiempo disponible generado.

También se observa que los valores del tiempo disponible desde que la distancia es nula hasta que llega a $s = t_{ontime} = 0.2\,s$ van tomando valores crecientes. Esto es debido a que el valor del límite superior del tramo, φ_C, va creciendo con la distancia de superposición

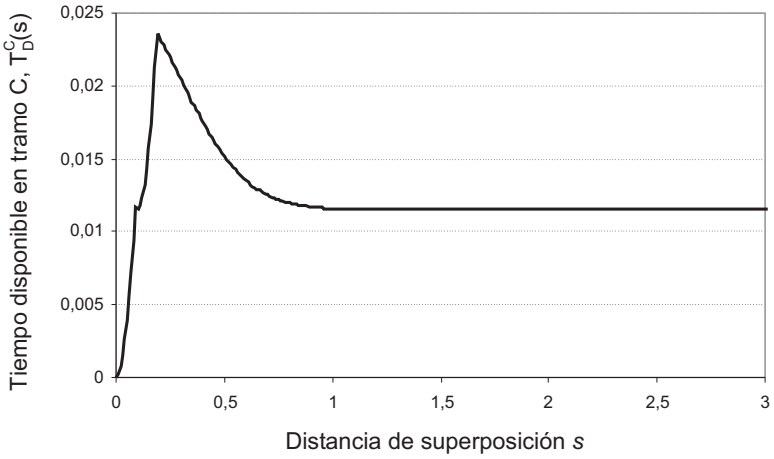


Fig. 3.12: Tiempo libre para el tramo C, $T_D^C(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2.

hasta llegar a su máximo en $s = t_{ontime}$, lo que se traduce en un aumento del tamaño del tramo. Esto hace que el tiempo disponible generado sea mayor conforme s aumenta. Nótese finalmente que, en el rango $s \in (0, t_{ontime})$, se produce una variación de la pendiente. Dicha variación es la que corresponde con el instante en el que existe un punto de cálculo, es decir, en $s = 0,1\ s$.

Tramo D

El tramo D, cuando existe, es el correspondiente al intervalo (a_n, a'_0). Para mostrar la expresión correspondiente a $T_D^D(s)$, al igual que para el tramo C, también se considerarán por separado los términos que calculan el tiempo disponible para los escenarios U y $\overline{U}$, de modo que:

$$T_D^D(s) = T_D^D(s)|_U + T_D^D(s)|_{\overline{U}} \tag{3.64}$$

Hay que destacar que, dado que el punto de cálculo a'_0, que es el límite del tramo, no se corresponde realmente con la llegada de un mensaje de ataque, sino con un tiempo $\overline{RTT}$ segundos antes de la llegada del siguiente mensaje de ataque. El tiempo disponible en caso de que se produzca el escenario U se generará hasta la llegada del mensaje de ataque en a'_1, por lo que su valor –Expresión (3.23)– será siempre $\overline{RTT}$, es decir:

$$t_D^U(t)\big|_{a'_1} = \overline{RTT} \tag{3.65}$$

Nótese que la anterior es la resolución de la Expresión (3.23), pero considerando que el siguiente punto de cálculo es a'_1 en lugar de a'_0, en cuyo caso el valor mínimo será siempre $\overline{RTT}$.

De este modo, el término de $T_D^D(s)$ que recoge el tiempo disponible generado por los términos relativos al escenario U será:

$$\begin{aligned} T_D^D(s)\big|_U &= \int_{a_n}^{a'_0} \overline{RTT} \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_j^U(t)\big|_{a_n}^{a'_0} \cdot f_j(t)dt + \\ &+ \int_{a_n}^{a'_0} \overline{RTT} \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_{j+1}^U(t)\big|_{a_n}^{a'_0} \cdot f_{j+1}(t)dt \end{aligned} \tag{3.66}$$

En la resolución de esta ecuación, para el cálculo del valor de β –Expresión (3.40)– y su probabilidad asociada –Expresión (3.44)– habrá que considerar, por la misma justificación que se hizo para el tramo A, un intervalo cuyo valor será:

$$b - a = min(a'_0 - a_n, \overline{RTT})$$

Respecto al valor del tiempo disponible generado en el escenario $\overline{U}$, hay que considerar que el máximo valor de tiempo disponible posible será $2 \cdot \overline{RTT}$, el cual se obtendrá cuando las dos salidas sucedan una justo $\overline{RTT}$ segundos tras la otra, mientras que el mínimo se consigue cuando ambas sucedan a la vez, generándose en este caso un valor de tiempo disponible igual a $\overline{RTT}$. Por esta razón, el valor medio se corresponderá con:

$$t_D^{\overline{U}}\big|_{a_n}^{a'_0} = \frac{3}{2} \cdot \overline{RTT} \tag{3.67}$$

Por último, la expresión para las contribuciones del escenario $\overline{U}$ al tiempo disponible en este tramo será:

$$T_D^D(s)\big|_{\overline{U}} = \int_{a_n}^{a'_0} \frac{3}{2} \cdot \overline{RTT} \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_j^{\overline{U}}(t)\big|_{a_n}^{a'_0} \cdot f_j(t)dt \tag{3.68}$$

La Fig. 3.13 muestra los valores obtenidos de $T_D^D(s)$ en el ataque con la configuración indicada en la Tabla 3.2, cuando se varía la distancia de superposición s. En ella se puede apreciar, en primer lugar, que los valores obtenidos son nulos cuando no se cumple la condición de existencia del tramo, es decir, cuando $s < t_{ontime} + \overline{RTT} = 0.7\ s$. Además, cuando la distancia de superposición va creciendo, el valor del tiempo disponible generado en el tramo también lo hace, aunque de modo asintótico. Esta asíntota se obtiene a partir del valor de la distancia de superposición que hace que no exista superposición entre las curvas de probabilidad de ocurrencia de las dos salidas.

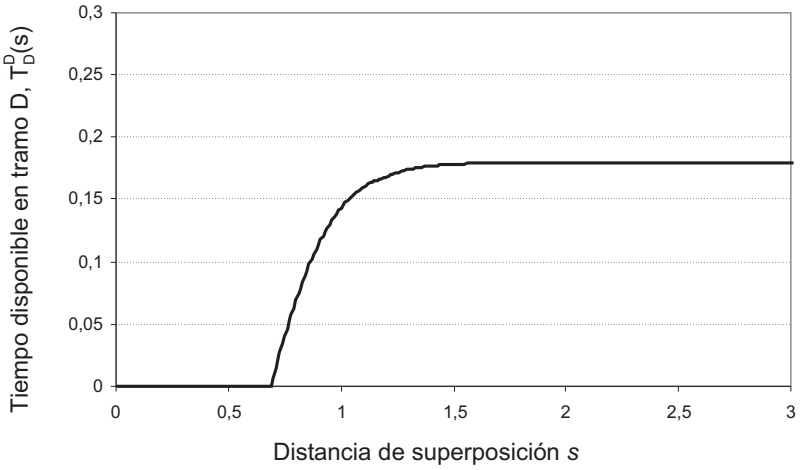


Fig. 3.13: Tiempo disponible para el tramo D, $T_D^D(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2.

Tramo E

El tramo E es el correspondiente al intervalo (φ_E, a'_1). Este tramo está situado en la zona de influencia conjunta de $f_j(t)$ y de $f_{j+1}(t)$, por lo que deberá considerar la influencia de ambas funciones, de modo que las salidas correspondientes podrán suceder de forma independiente (escenario U) o a la vez en una ventana temporal de tamaño $\overline{RTT}$ (escenario $\overline{U}$). Así, al igual que se ha realizado para los tramos C y D, se dividirá la expresión del tiempo disponible en este tramo, $T_D^E(s)$, en dos factores según consideren el escenario U o $\overline{U}$:

$$T_D^E(s) = T_D^E(s)|_U + T_D^E(s)|_{\overline{U}} \tag{3.69}$$

En este escenario, dado que la distancia $a'_1 - \varphi_E$ es siempre menor que $\overline{RTT}$, se cumplirá que el tiempo disponible instantáneo para el escenario U –Expresión (3.23)– siempre es:

$$t_D^U(t)\big|_{a'_1} = a'_1 - t \tag{3.70}$$

Y al aplicarle las consideraciones realizadas para la aparición de mensajes de ataque adicionales, este tiempo se convertirá, según la Expresión (3.41), en:

$$t_D^U(t)\big|_{a'_1}^{\beta} = \varphi_E + \rho \cdot ceil\left[\frac{t - \varphi_E}{\rho}\right] - t \tag{3.71}$$

donde:

$$\rho = \frac{a'_1 - \varphi_E}{\beta} \tag{3.72}$$

Por tanto, los factores correspondientes al cálculo del tiempo disponible en el tramo E cuando se considera el escenario U son:

$$\begin{aligned} T_D^E(s)\big|_U \quad = \quad & \int_{\varphi_E}^{a'_1} \left[t_D^U(t)\big|_{a'_1}^{\beta} \cdot P(\beta) + t_D^U(t)\big|_{a'_1}^{\beta+1} \cdot P(\beta+1) \right] \cdot P_j^U(t)\big|_{\varphi_E}^{a'_1} \cdot f_j(t)dt + \\ & + \int_{\varphi_E}^{a'_1} \left[t_D^U(t)\big|_{a'_1}^{\beta} \cdot P(\beta) + t_D^U(t)\big|_{a'_1}^{\beta+1} \cdot P(\beta+1) \right] \cdot P_{j+1}^U(t)\big|_{\varphi_E}^{a'_1} \cdot f_{j+1}(t)dt \end{aligned} \tag{3.73}$$

Con respecto al valor del tiempo disponible instantáneo cuando se considera el escenario $\overline{U}$ en este tramo hay que tener en cuenta que el mínimo valor de este tiempo se generará cuando las dos salidas se produzcan justamente antes de a'_1. En este caso, el tiempo que se generará será $min[\Delta, \overline{RTT}]$. En cuanto al valor máximo posible, éste se generará cuando una salida se produzca justamente después de φ_E y la otra justo antes de a'_1, por lo que el valor del tiempo en este caso será $min[\Delta, \overline{RTT}] + a'_1 - \varphi_E$. Realizando una media a dichos valores se obtiene que:

$$t_D^{\overline{U}}\big|_{\varphi_E}^{a'_1} = min[\Delta, \overline{RTT}] + \frac{a'_1 - \varphi_E}{2} \tag{3.74}$$

Finalmente, los factores que consideran el escenario $\overline{U}$ dentro de la expresión del tiempo disponible para el tramo E son:

$$T_D^E(s)\big|_{\overline{U}} = \int_{\varphi_E}^{a'_1} t_D^{\overline{U}}\big|_{\varphi_E}^{a'_1} \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_j^{\overline{U}}(t)\big|_{\varphi_E}^{a'_1} \cdot f_j(t)dt \tag{3.75}$$

La Fig. 3.14 muestra los valores obtenidos de $T_D^E(s)$ en el ataque con la configuración indicada en la Tabla 3.2, cuando se varía la distancia de superposición s. En ella se puede apreciar claramente que en el tramo se genera tiempo disponible a partir de los valores de la distancia de superposición que cumplen $s > t_{ontime} = 0.2\ s$, lo que corresponde con la condición de existencia del tramo. Además, a partir de este punto el tramo E va creciendo en extensión al incrementarse la distancia de superposición, lo que hace que el valor del tiempo disponible aumente. Esto se cumple hasta el valor de s en el que la extensión del tramo E es máxima, es decir, cuando $\varphi_E = a'_0$, lo que corresponde con el valor $s > \overline{RTT} + t_{ontime} = 0.7s$.

Para valores $s > 0.7\ s$, el comportamiento del tiempo disponible es decreciente, dado que el tamaño del tramo ya es fijo (el máximo) y las curvas de probabilidad se van separando progresivamente, haciendo que las probabilidades dentro del tramo sean menores. De hecho, el tiempo disponible tiene un valor asintótico cuando s crece, que se obtiene a partir del valor de s en el que no existe superposición entre las dos curvas de probabilidad de ocurrencia.

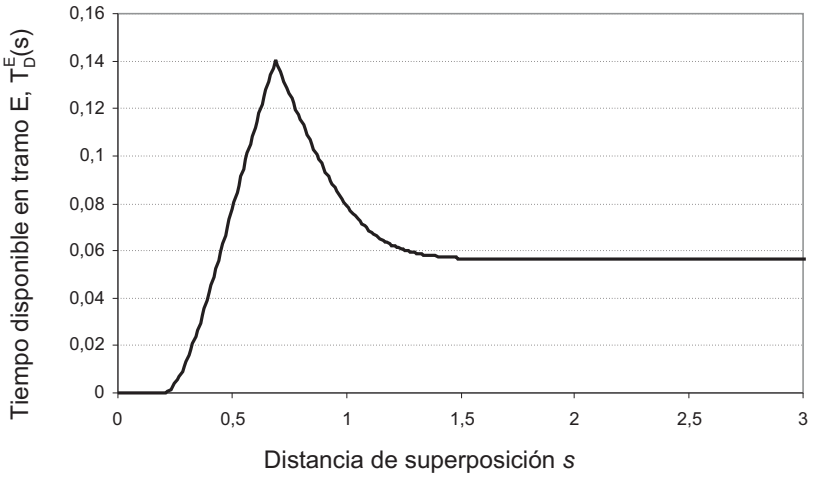


Fig. 3.14: Tiempo disponible para el tramo E, $T_D^E(s)$, calculado para el ataque con la configuración indicada en la Tabla 3.2.

Tramo F

El tramo F es el correspondiente al intervalo (a'_1, a'_n). Este tramo está dentro de la zona de influencia de $f_j(t)$, por lo que sólo se considera que se pueda producir una salida cuya función de probabilidad de ocurrencia es $f_j(t)$.

El tiempo disponible instantáneo que se generará en este tramo corresponde con $t_D^{\alpha}|_F$, dado por la Expresión (3.33). La expresión final para el cálculo del tiempo disponible en este tramo es:

$$T_D^F(s) = \int_{a'_1}^{a'_n} t_D^{\alpha}\big|_F \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_j^U(t)\Big|_{a'_1}^{a'_n} \cdot f_j(t)dt \tag{3.76}$$

La Fig. 3.15 muestra los valores obtenidos de $T_D^F(s)$ en el ataque con la configuración dada en la Tabla 3.2, cuando se varía la distancia de superposición s. En ella se aprecia que, dado que el tamaño del tramo es siempre fijo, el máximo valor se obtiene cuando la probabilidad aportada por la curva $f_j(t)$ es máxima, es decir, para $s = 0$. Cuando el valor de s se incrementa, dicha probabilidad va disminuyendo, hasta alcanzar una distancia de superposición en la que la curva $f_j(t)$ no tiene influencia alguna en este tramo. En este caso no existe una asíntota con valor no nulo debido a que, al estar el tramo F en la zona de influencia de $f_j(t)$, no se considera la influencia de $f_{j+1}(t)$.

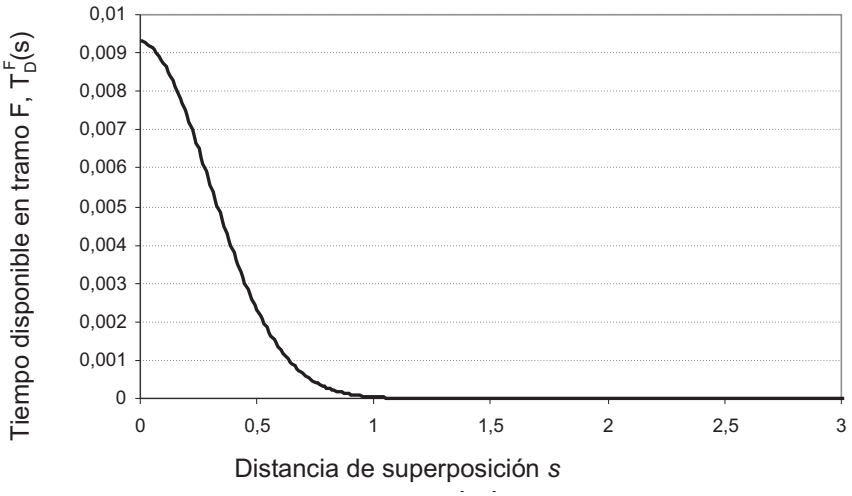


Fig. 3.15: Tiempo disponible para el tramo F, $T_D^F(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2.

Tramo G

El tramo G es el correspondiente al intervalo (a'_n, ∞). En este caso, que considera la influencia de la función de probabilidad de ocurrencia $f_j(t)$ en las secciones posteriores a $f_{j+1}(t)$, la influencia de $f_{j+1}(t)$ se considera nula, debido a que el tramo G se encuentra ubicado en la zona de influencia de $f_j(t)$.

Para el tramo G, el valor instantáneo del tiempo disponible a tener en cuenta es $t_D^{\alpha}|_G$, dado por la Expresión (3.34). Con las consideraciones dadas en todo el Apartado 3.4.2, la expresión final para la contribución al tiempo disponible en el tramo G, $T_D^A(s)$, queda de la siguiente forma:

$$T_D^G(s) \quad = \quad \int_{a'_n}^{\infty} t_D^{\alpha}\big|_G \cdot \left(\frac{P(\beta)}{\beta} + \frac{P(\beta+1)}{\beta+1} \right) \cdot P_j^U(t)\big|_{a'_n}^{\infty} \cdot f_j(t)dt \tag{3.77}$$

donde para el cálculo del valor de β y su probabilidad, $P(\beta)$, dados por las Expresiones (3.40) y (3.44), se considerará que la distancia del intervalo es la del máximo valor que puede obtener el tiempo disponible, es decir, $b - a = \overline{RTT}$, por lo que:

$$\beta = floor \cdot \left[\frac{\overline{RTT} \cdot N_s}{\overline{T_s}} \right] + 1 \tag{3.78}$$

$$P(\beta) = \beta - \frac{\overline{RTT} \cdot N_s}{\overline{T_s}} \tag{3.79}$$

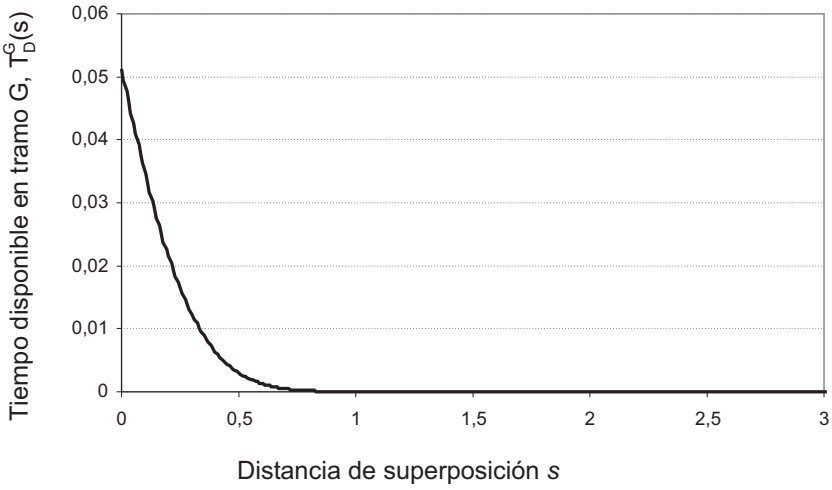


Fig. 3.16: Tiempo disponible para el tramo G, $T_D^G(s)$, calculado para el ataque con la configuración dada en la Tabla 3.2.

La Fig. 3.16 muestra los valores obtenidos para $T_D^G(s)$ en el ataque con la configuración dada en la Tabla 3.2, cuando se varía la distancia de superposición s. En ella se aprecia el valor decreciente del tiempo disponible cuando aumenta s dado que la influencia de $f_j(t)$ va siendo cada vez menor, hasta alcanzar un valor nulo. El comportamiento en este tramo es similar al obtenido para el tramo F; sin embargo, dada su mayor lejanía con respecto a la función $f_j(t)$, los valores de tiempo disponible obtenidos son menores y la asíntota en el valor cero se alcanza antes.

Capítulo 4

El ataque DoS a baja tasa contra servidores iterativos

4.1 Introducción

En este capítulo se abordará la aplicación de los mecanismos generales de ejecución del ataque DoS a baja tasa cuando el objetivo del ataque es un servidor iterativo.

Los servidores iterativos son aquellos en los que el procesamiento de las peticiones se realiza de modo secuencial, es decir, cada vez que una petición está siendo procesada, el resto debe esperar a que se termine el procesamiento que se está llevando a cabo para que llegue su turno. Así, para representar a un servidor iterativo se puede utilizar el modelo genérico para servidores, propuesto en el Capítulo 2, particularizándolo de modo que el valor del número de elementos de procesamiento del servidor, N_s, sea igual a la unidad.

Aunque el uso de este tipo de servidores no es común en las aplicaciones más extendidas presentes en *Internet*, como la navegación web, el correo electrónico o los servidores de archivos, aun así existen muchos campos en los que sí se utilizan, por dos motivos fundamentales. El primero consiste en que uno de los atractivos para implementar un servidor con una arquitectura iterativa es su simplicidad, tanto en su desarrollo como en su depuración. Los servidores iterativos son, además, idóneos para ejecutar tareas que necesitan poco tiempo de procesamiento, ya que en este caso, se podrá aprovechar la simplicidad de estos servidores sin que ello implique una latencia demasiado grande para procesar las peticiones. Típicamente, para las aplicaciones que utilizan protocolos no orientados a conexión, como por ejemplo UDP, puede estar recomendado el uso de un esquema iterativo de funcionamiento, más que con protocolos orientados a conexión en los que el tiempo en servir una petición será mayor debido a la necesidad de establecer previamente una conexión y posteriormente liberarla. Un ejemplo de aplicación iterativa es un servidor que proporciona la hora.

Hay una segunda motivación para el uso de los servidores iterativos: la existencia de casos en los que la propia operación que el servidor debe llevar a cabo exige este tipo de

diseño. Esto sucede en caso de que haya que acceder a determinados recursos en exclusividad, es decir, cuando no se puedan procesar dos o más peticiones al mismo tiempo debido a que éstas requieren el acceso a un recurso en modo exclusivo. Aplicaciones típicas en este sentido son aquellas que llevan a cabo el control de mecanismos. Por ejemplo, un servidor que controle un brazo mecánico debe ejecutar secuencialmente cada petición de movimiento que recibe.

En el Capítulo 3 se ha especificado el mecanismo general para llevar a cabo el ataque a baja tasa contra servidores, presentando la estrategia básica y el método general para ejecutarlo. La aplicación de estos principios generales cuando se ejecuta el ataque contra servidores iterativos implica concretar algunos detalles que permiten la adaptación de algunos mecanismos a las características propias de estas aplicaciones. Este capítulo tratará de profundizar en dichas circunstancias.

Por otro lado, también se realizará una evaluación del ataque en este tipo de escenarios, con el fin de comprobar los rendimientos obtenidos y los potenciales problemas a los que el atacante se enfrenta al ejecutarlo.

La estructura del capítulo es la siguiente. En primer lugar, en el Apartado 4.2 se muestran los detalles de implementación del ataque DoS a baja tasa cuando es ejecutado contra servidores iterativos. Posteriormente, en el Apartado 4.3 se realiza la evaluación del rendimiento del ataque en un entorno de simulación. En el Apartado 4.4 se amplía la evaluación previa a entornos reales de ejecución del ataque. Algunas mejoras del ataque se proponen en el Apartado 4.5. Finalmente, en el Apartado 4.6 se recogen las conclusiones de este capítulo.

4.2 Implementación del ataque contra servidores iterativos

La estrategia básica para ejecutar el ataque DoS a baja tasa contra servidores se basa en realizar, en primer lugar, la predicción de los instantes en los que se van a producir las salidas en el servidor para, posteriormente, tratar de ocupar las posiciones que las salidas dejan libres en las colas de servicio correspondientes mediante la ejecución de periodos básicos de ataque.

La adaptación de los mecanismos generales del ataque a un escenario en el que la víctima es un servidor iterativo implica dos tareas: descripción de la vulnerabilidad que permite realizar la predicción de los instantes de las salidas y del método para efectuarla, y adaptación del periodo básico de ataque a las características propias del servidor iterativo víctima del ataque.

Con respecto a la vulnerabilidad a explotar, el estudio del tiempo entre salidas presentado en el Capítulo 2 concluye que, en el caso de los servidores iterativos, cuando todas las peticiones que se procesan en el servidor son idénticas, el tiempo entre salidas que se observa desde el punto de vista de un usuario se obtiene utilizando la Expresión (2.9).

Esto implica que la media del tiempo entre salidas se corresponde con la media del tiempo de servicio requerido por todas y cada una de las peticiones, dado que éstas son iguales. Por tanto, para realizar la predicción de los instantes en los que se producen las salidas, el atacante puede realizar varias pruebas consistentes en el envío consecutivo en el tiempo de dos mensajes idénticos hacia el servidor y en la medición del tiempo entre las correspondientes respuestas. Nótese que para que la medición sea correcta es preciso que las dos peticiones enviadas se hayan introducido consecutivamente en la cola de servicio. Esta condición se cumplirá en la mayor parte de los casos siempre que el envío de las dos peticiones sea suficientemente próximo en el tiempo. Para los casos en los que no se cumple esta condición se podrá identificar el error identificando los tiempos que aportan alta varianza con respecto a la media, por lo que, tras varias pruebas, el atacante estará en condiciones de estimar el tiempo entre salidas mediante este simple mecanismo.

Una vez realizada la estimación del tiempo entre salidas, el atacante tratará de conseguir que la cola de servicio del servidor contenga peticiones idénticas procedentes del propio atacante. Más adelante se harán las consideraciones necesarias sobre el modo en que el atacante realiza esta tarea.

Cuando la cola de servicio está llena, el proceso de ataque consiste en la ejecución de la estrategia de ataque presentada en el Capítulo 3, es decir, en la predicción de los instantes en que sucederán las salidas y la programación consecutiva de periodos básicos de ataque en torno a dichos instantes.

Por tanto, para cada salida, el procedimiento de ejecución del ataque se realiza de la siguiente forma:

1. En primer lugar, cada vez que el atacante concluye que se ha producido una salida en el servidor, se realiza la predicción del instante en que sucederá la siguiente. Al instante en que el atacante realiza esta predicción se le llamará en adelante t_c. Se concluirá que se ha producido una salida en el servidor mediante la constatación de alguno de estos dos hechos posibles:

   - El atacante recibe una respuesta procedente del servidor. En este caso, se concluirá que el instante en que se produjo la salida anterior corresponde con $t_c - \overline{RTT}/2$. Además, para que el atacante reciba la respuesta, ésta debe corresponder necesariamente con una petición que, procedente del mismo, se introdujo previamente en la cola de servicio.
   - El atacante observa que, aunque no se haya recibido ninguna respuesta del servidor, el tiempo que ha pasado desde que se concluyó que se había producido la salida anterior es igual al valor medio del tiempo entre salidas, $\overline{\tau_u} = \overline{T_s}$. También entonces el atacante deduce que la salida anterior se produjo en el instante $t_c - \overline{RTT}/2$.

     Este caso se puede producir por dos motivos posibles. En primer lugar, puede suceder que la respuesta del servidor no corresponda con una petición procedente del atacante sino de otro usuario, por lo que la respuesta no se recibirá por parte del atacante. Nótese que concluir que se ha producido la salida cuando expira el tiempo entre salidas permite estimar los instantes de las salidas incluso cuando no se reciben las respuestas procedentes del servidor. Es evidente que, en este caso, la

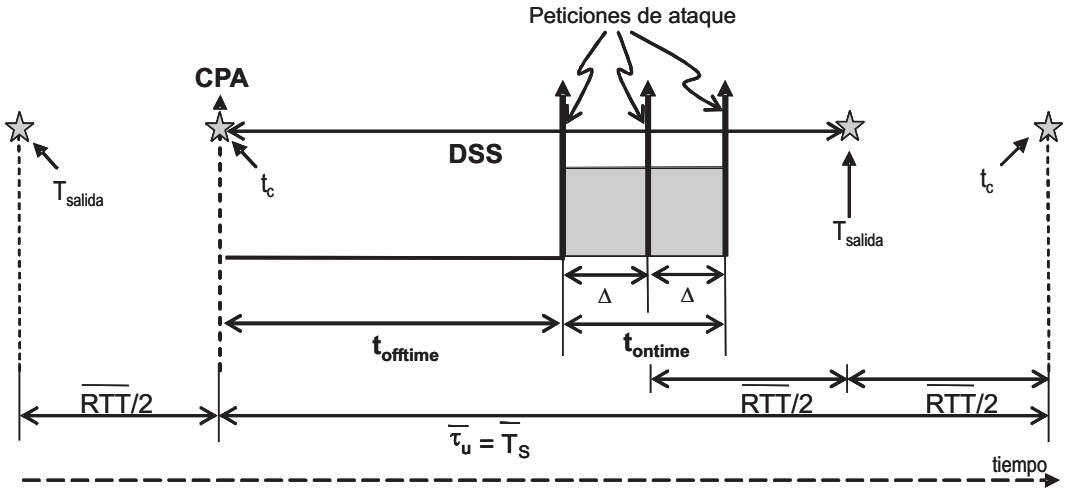


Fig. 4.1: Periodo básico de ataque en el ataque DoS a baja tasa contra servidores iterativos: forma de onda y parámetros.

estimación del momento en que se ha producido la salida tendrá mayor desviación que cuando se ha recibido la respuesta del servidor. En segundo lugar, también puede suceder que, por efecto de la variabilidad de RTT y T_s, la respuesta se haya retrasado. Si esto sucede, el atacante estaría concluyendo erróneamente que la anterior salida se ha producido $\overline{RTT}/2$ segundos antes. El método de ataque, que se detallará con posterioridad, tiene previsto un mecanismo para evitar que este error afecte al funcionamiento del ataque.

Por tanto, toda vez que el atacante concluye que se ha producido una salida en $t_c - \overline{RTT}/2$, el instante en que se producirá la siguiente salida en el servidor, T_{salida}, puede evaluarse de acuerdo a la siguiente función de probabilidad:

$$P(T_{salida} = t) = \mathcal{N}(t_c - \frac{\overline{RTT}}{2} + \overline{T_s}; var[T_s] + var[RTT]) \tag{4.1}$$

El valor medio de esta función de probabilidad, $\overline{T}_{salida}$, constituye, por tanto, la predicción del instante en el que se producirá la siguiente salida.

2. Se programa un periodo básico de ataque de modo que llegue al servidor en torno al instante previsto, T_{salida}, con el fin de capturar la posición en la cola de servicio que se liberará por la siguiente salida. Este periodo de ataque tiene las características definidas en el Capítulo 3, aunque los valores de sus parámetros se particularizan para el caso descrito (servidor iterativo) del modo en que se representa en la Fig. 4.1:

   - *Comienzo del periodo de ataque, CPA:*
     Por definición será el instante en el que se realice la predicción de la siguiente salida, es decir, se corresponde con t_c: $CPA = t_c$.

Ha de recordarse que el instante t_c corresponde con el instante en el que o bien se recibe una respuesta del servidor, o bien el tiempo transcurrido desde el último t_c es igual a $\overline{\tau_u}$.

- *Distancia a la siguiente salida, DSS*:

  Se calculará siempre que se inicia el periodo de ataque, *CPA*, según la Expresión (4.1), de modo que:

$$DSS = \overline{T}_{salida} - t_c = \overline{T_s} - \frac{\overline{RTT}}{2} \tag{4.2}$$

- Consecuentemente con el cálculo propuesto anteriormente para DSS, el valor del parámetro $t_{offtime}$ se puede calcular usando la Expresión (3.3) de este modo:

$$t_{offtime} = \overline{T_s} - \frac{t_{ontime}}{2} - \overline{RTT} \tag{4.3}$$

4.2.1 Fases del ataque

En la realización del ataque aparecen tres fases diferenciadas:

- *Fase de análisis*:

  Esta etapa es previa a la realización del ataque. En ella hay que realizar la estimación de los parámetros que, posteriormente, serán necesarios para la ejecución del mismo.

  Concretamente, el atacante deberá estimar, por un lado, el tiempo medio de ida y vuelta hasta el servidor, $\overline{RTT}$, y también el tiempo medio de servicio, $\overline{T_s}$, empleado en el procesamiento de las peticiones idénticas que serán utilizadas posteriormente para el ataque.

  Como se ha comentado previamente, para poder estimar el tiempo de servicio, el atacante deberá enviar parejas de peticiones con un mínimo espaciado en el tiempo entre ellas de modo que se maximice la probabilidad de que éstas se almacenen en la cola de servicio de forma consecutiva. El tiempo entre las respuestas a estas peticiones, de acuerdo a la Expresión (2.9), se corresponde con la media del tiempo de servicio, $\overline{T_s}$. Cuantas más veces se repita este procedimiento, más fiable será la estimación del tiempo medio de servicio.

- *Fase transitoria*:

  Esta fase se inicia en el momento en que el atacante comienza la ejecución del ataque. Su objetivo es llenar inicialmente la cola de servicio con peticiones propias. Ya se ha comentado que estas peticiones serán idénticas entre sí con el fin de preservar las características del tiempo entre salidas, lo que permite realizar la predicción de los instantes de las mismas.

  Aunque esta fase del ataque se podría realizar mediante una tasa elevada de tráfico, de esta forma el ataque sería más fácilmente detectable por parte de ciertos mecanismos de seguridad. Para evitar, por tanto, tasas de tráfico elevadas, será suficiente si se

utiliza una tasa ligeramente mayor que la tasa de salida del servidor. También se puede optar por ir capturando un cierto número de posiciones en la cola de servicio y mantenerlas mediante las técnicas utilizadas en la fase permanente del ataque que a continuación se detalla. En este caso, el precio a pagar consistirá en tener una fase transitoria de mayor duración.

Normalmente, el atacante puede detectar la saturación de la cola de servicio y, por tanto, el fin de la fase transitoria, mediante la observación del rechazo de sus mensajes. Por ejemplo, en caso de que se trate de un servidor orientado a conexión con protocolo TCP, la recepción de mensajes de tipo RST como respuesta a los mensajes de establecimiento de conexión SYN indicará que existe saturación en el servidor.

- *Fase permanente*:

  El objetivo de la fase permanente del ataque es mantener las posiciones capturadas en la cola de servicio el máximo tiempo posible. Esto se consigue mediante la estimación de los instantes de las salidas y la programación de periodos básicos de ataque consecutivos durante el tiempo que dure éste.

  Ya se ha comentado previamente que, aunque no se haya recibido una respuesta procedente del servidor por parte del atacante, en caso de que el tiempo transcurrido desde la última estimación realizada sea igual al tiempo entre salidas, se concluye que se produjo una salida y, por tanto, se inicia un periodo de ataque. Ahora bien, se ha señalado también que, en caso de que no se haya recibido la respuesta debido a un retraso motivado por la variabilidad de RTT y T_s, el ataque dispone de un mecanismo que prevé estas situaciones para corregir estos efectos.

  Para explicar este mecanismo es necesario un estudio detallado de la dinámica seguida cuando se ejecutan los periodos básicos de ataque de forma continuada. En la Fig. 4.2 se muestra un diagrama de los diferentes eventos y parámetros en dos periodos de ataque que se producen consecutivamente en el tiempo, desde la perspectiva del servidor y también desde la del atacante. Cuando se recibe una salida procedente del servidor (O1), se procede a responder con un mensaje de ataque (A1), comenzar un nuevo periodo de ataque (CPA_1) con el consiguiente cálculo de la distancia a la siguiente salida y, finalmente, programar el temporizador $t_{out}(1)$, que expirará cuando su valor sea $\overline{\tau_u}$. El valor del tiempo *offtime*, $t_{offtime}(1)$, se calcula utilizando la Expresión (4.3). Cuando finaliza $t_{offtime}(1)$, comienza $t_{ontime}(1)$ con la emisión de un nuevo mensaje de ataque (A2). Durante el periodo *ontime*, se manda un mensaje de ataque cada *intervalo*, Δ (mensajes A3 y A4). Al expirar el temporizador $t_{out}(1)$ se comienza otro nuevo periodo de ataque (CPA_2), con el consiguiente cálculo de $t_{offtime}(2)$ y la programación de otro temporizador $t_{out}(2)$, que también expirará cuando su valor sea $\overline{\tau_u}$.

  Si no se recibiera ninguna salida del servidor, el nuevo periodo de ataque se comportaría exactamente igual que el anterior. Sin embargo, si el atacante recibe una salida desde el servidor (O2), como se ilustra en la Fig. 4.2, el periodo de ataque se reinicia ($CPA_{2'}$) y se calculan de nuevo los parámetros $t_{offtime}(2')$ y $t_{out}(2')$. Este mecanismo es el que permite que el atacante se resincronice con cada salida recibida del servidor, evitando que las desviaciones en la estimación de las salidas se acumulen. Por último, al recibir la salida (O2) se genera un mensaje de ataque como respuesta (A5).

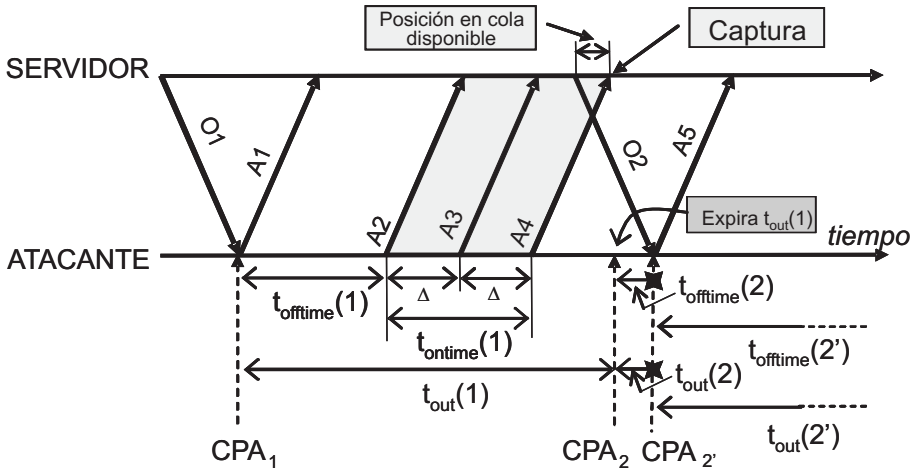


Fig. 4.2: Diagrama con la ejecución de dos periodos básicos de ataque consecutivos en el tiempo.

4.3 Evaluación del rendimiento del ataque

Una vez que se ha detallado el procedimiento para llevar a cabo el ataque DoS a baja tasa contra un servidor iterativo, es interesante evaluar el rendimiento que se puede obtener con dichos ataques. Con este fin se ha realizado la implementación, en el simulador *Network Simulator 2* (NS2) [Fall and Varadhan, 2007], de un servidor con comportamiento iterativo, de un grupo de usuarios que generan tráfico según el modelo especificado en el Capítulo 2 y, por último, de un módulo que ejecuta el ataque tal y como se ha especificado previamente. Por simplicidad, el módulo de ataque no se ha implementado de forma distribuida dado que esta característica no es necesaria para llevar a cabo el ataque.

En la implementación del simulador se ha habilitado la recogida de los datos que permiten evaluar las características que a continuación se van a presentar. Así, en este apartado se mostrará, en primer lugar, un estudio del comportamiento del rendimiento del ataque con la variación de los diferentes parámetros de configuración, tanto del ataque como del servidor. Posteriormente se procederá a validar los resultados con los modelos matemáticos propuestos en el Capítulo 3. Finalmente, se analizarán las capacidades del ataque en cuanto a sus prestaciones y la consecución de su objetivo.

4.3.1 Variaciones con los parámetros de diseño

Para obtener un conocimiento más detallado del comportamiento del ataque es necesario realizar un análisis de las variaciones que sufre su rendimiento cuando se modifican los diferentes parámetros de configuración, tanto del servidor como del ataque.

Parámetro	Valor
t_{ontime}	0,4 s
Intervalo, Δ	0,2 s
Tiempo de servicio, T_s	$\mathcal{N}$(5 s; 0,1)
Tiempo de ida y vuelta, RTT	$\mathcal{N}$(0,6 s; 0,1)
Tiempo entre llegadas de usuarios, T_a	6 s

Tabla 4.1: Valores por defecto para los parámetros estudiados en los experimentos cuyos resultados se muestran en las Fig. 4.3 y 4.4.

Los parámetros a estudiar serán:

- *Parámetros del ataque*:

  Los parámetros del ataque a considerar serán aquellos configurables por parte del atacante, es decir, t_{ontime} y el valor del *intervalo*, Δ.

- *Parámetros del servidor*:

  En este caso se analizará el único factor del servidor iterativo que influye en el ataque, es decir, el tiempo de servicio. Por ello, se estudian sólo dos parámetros: la media del tiempo de servicio, $\overline{T_s}$, y su varianza, $var[T_s]$.

- *Parámetros de la red de transporte*:

  Para la red de transporte se analizará el tiempo medio de ida y vuelta $\overline{RTT}$ entre el atacante y el servidor víctima del ataque, y también su varianza, $var[RTT]$. Dado que la varianza de RTT afecta conjuntamente con la varianza del tiempo de servicio –ver Expresión (2.9)–, realmente se estudiará la variación de $var[T_s] + var[RTT]$.

- *Parámetros del comportamiento de los usuarios*:

  El único parámetro a estudiar con respecto al usuario será el tiempo entre llegadas de mensajes de usuario, T_a.

En una primera aproximación, los ajustes de los diferentes parámetros se suponen independientes entre sí. Por tanto, para cada uno de ellos, se estudiará la evolución de los indicadores de disponibilidad, D, porcentaje de tiempo disponible, T_D, y sobrecarga, S, cuando los valores del parámetro considerado varían de forma independiente del resto.

Algunos resultados obtenidos a partir de los diferentes experimentos realizados se recopilan en las Fig. 4.3 y 4.4. En ellas se muestra, para cada uno de los parámetros analizados, la evolución de los indicadores de rendimiento del ataque. El valor por defecto de los parámetros que no se varían está indicado en la Tabla 4.1. Algunas conclusiones que se pueden extraer de estos resultados son:

- Las tendencias en el comportamiento de D y de T_D son similares. Es decir, un incremento en el valor de D supone también un mayor valor en T_D, tal y como es de esperar, ya que la existencia de mayor porcentaje de tiempo disponible aumenta la probabilidad de que el usuario capture posiciones en la cola de servicio.

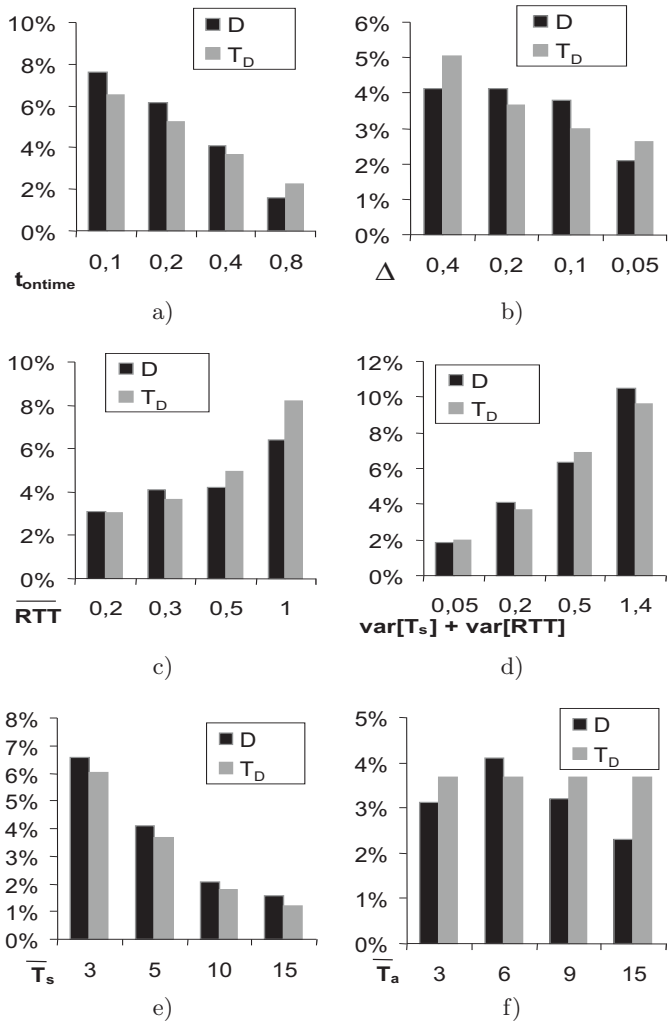


Fig. 4.3: Evolución de los indicadores D y T_D, ante la variación de los parámetros: a) t_{ontime}; b) intervalo, Δ; c) tiempo medio de ida y vuelta, $\overline{RTT}$; d) varianzas del tiempo de servicio y del tiempo de ida y vuelta, $var[T_s] + var[RTT]$; e) media del tiempo de servicio, $\overline{T_s}$; y f) tiempo medio entre llegadas de usuario, $\overline{T_a}$.

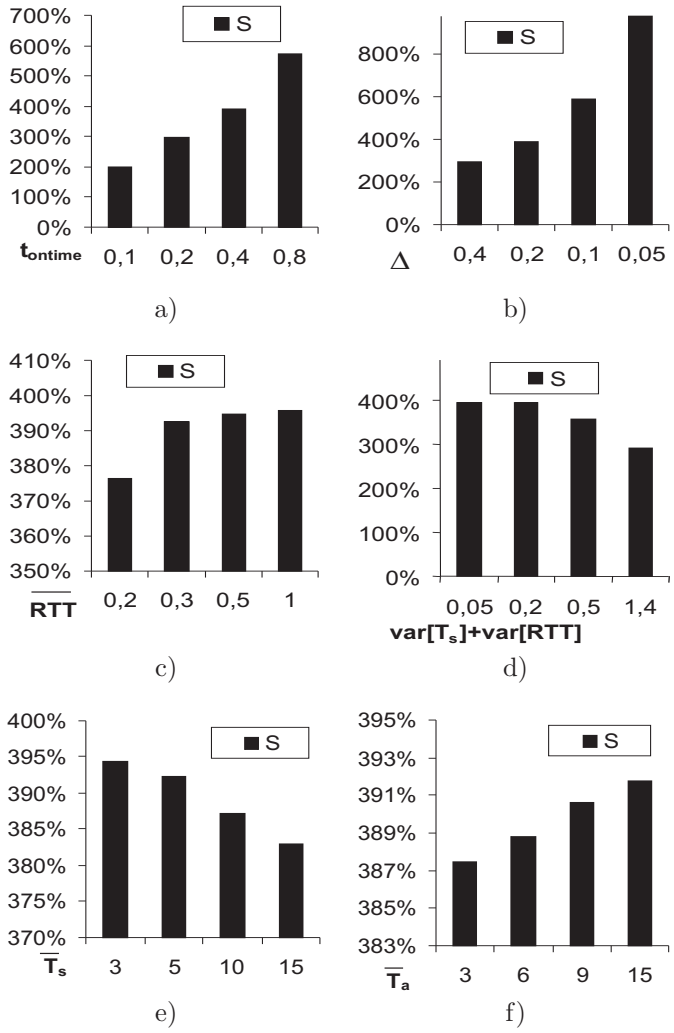


Fig. 4.4: Evolución del indicador S ante la variación de los parámetros: a) t_{ontime}; b) intervalo, Δ; c) tiempo medio de ida y vuelta, $\overline{RTT}$; d) varianzas del tiempo de servicio y del tiempo de ida y vuelta, $var[T_s] + var[RTT]$; e) media del tiempo de servicio, $\overline{T_s}$; y f) tiempo medio entre llegadas de usuario, $\overline{T_a}$.

- Cuando el tiempo *ontime* es mayor o cuando el intervalo es más corto, tanto D como T_D se reducen. Esto era esperable, ya que un mayor periodo de actividad en la onda de ataque reduce más los efectos de las posibles desviaciones producidas en la estimación de la salida o en el envío durante el periodo de ataque al servidor. Además, el hecho de que *ontime* sea mayor o el intervalo menor genera mayor sobrecarga, debido a que se envían más mensajes de ataque.

- Los incrementos en el valor de $\overline{RTT}$ en el rango de valores $\overline{RTT} > \Delta$ afectan solamente a la eficiencia (D y T_D) pero apenas modifican la sobrecarga. De hecho, conforme $\overline{RTT}$ aumenta se obtienen valores mayores para D y T_D, lo que implica una menor eficiencia.

- El incremento en el valor de $var[T_s] + var[RTT]$ implica la obtención de una menor eficiencia. Esto es debido a que las desviaciones que se producen en la estimación del instante de la salida pueden ser mayores y, por tanto, también lo será el número de salidas que se producirán fuera del periodo *ontime* de la onda de ataque.

- Los valores de D y T_D decrecen cuando el valor medio del tiempo de servicio $\overline{T_s}$ aumenta. En realidad, el tiempo disponible total generado en cada periodo de ataque será el mismo aunque se varíe $\overline{T_s}$. Ahora bien, como T_D se define como un porcentaje del tiempo de observación, al aumentar el valor medio del tiempo de servicio existirán menos periodos de ataque en dicho periodo de observación. Así, el porcentaje de tiempo disponible disminuye y, consecuentemente, también lo hace D.

- No se aprecia una dependencia lineal entre las variaciones de la tasa de llegada de mensajes de usuario y la eficiencia obtenida por el ataque. En cuanto a la sobrecarga, se aprecia un aumento no muy significativo cuando aumenta el tiempo entre llegadas de mensajes de usuario. Esto es debido a que cuando el usuario realiza un mayor número de capturas (al haber mayor número de llegadas), las salidas correspondientes no generan, por parte del atacante, un mensaje de respuesta adicional, rebajándose por tanto un poco la sobrecarga. Nótese que el hecho de que el usuario realice más capturas no implica que la disponibilidad suba, ya que este indicador se mide como el porcentaje de capturas realizadas por el usuario entre el número de mensajes enviados por el mismo, factor este último que también tendrá un valor mayor.

Validación de los resultados con el modelo matemático para sistemas sin superposición

Las conclusiones mostradas anteriormente se han obtenido a partir de la experimentación con diferentes escenarios y configuraciones de ataque y del servidor. Ahora bien, cabe preguntarse sobre la posible existencia de escenarios que no respondan al comportamiento obtenido en los experimentos realizados. En este punto es donde aportan su beneficio los modelos matemáticos para los indicadores de rendimiento desarrollados en el Capítulo 3. En efecto, la existencia de dicho modelos permite comprobar que las conclusiones aportadas anteriormente quedan refrendadas por los modelos matemáticos para los diferentes indicadores de rendimiento. Esto permite generalizar las conclusiones obtenidas en los escenarios probados para otros posibles.

Por otro lado, aunque las expresiones del modelo matemático muestran que la tendencia de los indicadores D, S y T_D corresponde con la obtenida en la experimentación, dado que en dichos modelos se han realizado ciertas aproximaciones, cabe preguntarse acerca de la exactitud de los valores que se obtienen a partir de los modelos con respecto a los que se miden en un entorno de simulación.

Con respecto al porcentaje de tiempo disponible, ya que el servidor iterativo se puede modelar como un sistema sin superposición, se analizará el modelo propuesto para los sistemas sin superposición. Dado que para este modelo se han realizado diversas aproximaciones y, también debido a que éste se basa en la aproximación estadística del comportamiento del ataque, es de esperar que exista una cierta desviación entre la estimación dada por el modelo y los valores reales obtenidos por un ataque.

Para comprobar la exactitud de las expresiones que permiten el cálculo del porcentaje de tiempo disponible en sistemas sin superposición se han simulado múltiples escenarios diferentes con configuraciones diversas de los parámetros de ataque, y con parámetros de configuración del servidor también distintos. Los resultados obtenidos se aproximan bastante bien a los conseguidos mediante simulación.

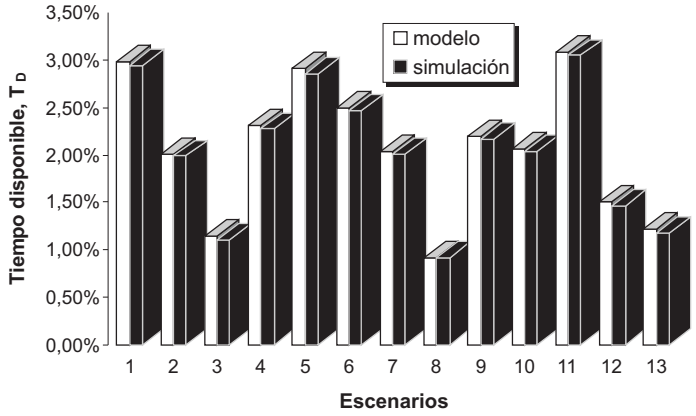
En la Fig. 4.5(a) se pueden observar los valores correspondientes a 13 de estas simulaciones realizadas. De entre todos los resultados obtenidos, se han mostrado los valores más significativos. La Fig. 4.5(b) muestra, para las mismas simulaciones, el valor absoluto de las diferencias entre el valor simulado y el teórico de T_D. Se puede comprobar que existe una diferencia máxima de variación del valor simulado respecto del obtenido teóricamente igual a 3,80%, con un valor promedio del 1,70%.

Para los modelos correspondientes a la disponibilidad, D, y la sobrecarga, S, se ha realizado el mismo estudio. En la Fig. 4.6(a) se representan los valores obtenidos para D en los 13 escenarios anteriormente elegidos, mientras que en la Fig. 4.6(b) se muestra el valor de las diferencias relativas entre los valores de simulación y del modelo, con respecto al valor del modelo. En este caso la máxima variación obtenida ha sido del $11,11\%$, mientras que el promedio de dichas diferencias es del $5,39\%$.

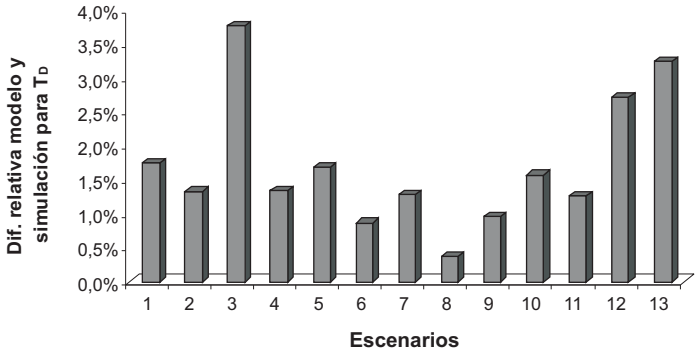
Los resultados correspondientes a los mismos 13 escenarios, en este caso para el indicador de sobrecarga S, se presentan en la Fig. 4.7(a). En ella se observan los valores obtenidos tanto con la simulación como a partir del modelo. La Fig. 4.7(b) muestra los valores porcentuales de las diferencias relativas con respecto al valor aportado por el modelo. El valor máximo obtenido para estas diferencias es del 4,02%, mientras que el promedio se sitúa en el 1,42%.

De los resultados obtenidos se puede comprobar que las desviaciones que aparecen en el modelo de D con respecto a la simulación son mayores que las obtenidas para T_D. Este hecho es coherente, ya que el modelo de D se apoya en el que estima el porcentaje de tiempo disponible, por lo que la aparición de mayores desviaciones se justifica por la realización de aproximaciones acumuladas.

Finalmente, de los experimentos realizados se puede concluir que, si bien los valores procedentes de los modelos y de la simulación presentan unas desviaciones que, en ciertos casos, pueden ser significativas, las tendencias del comportamiento de los indicadores son completamente análogos. Esto permite concluir la validez de los modelos matemáticos de-

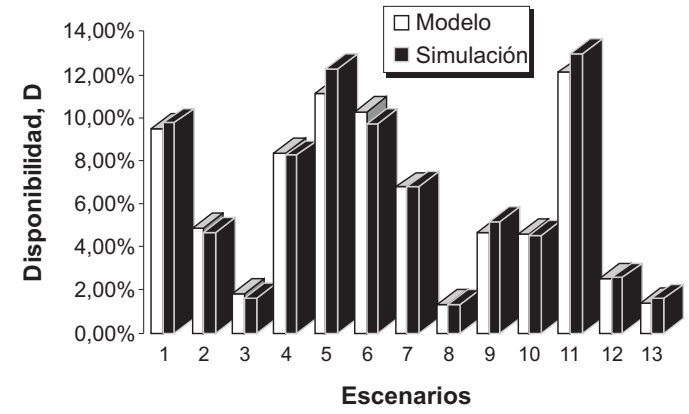


(a)

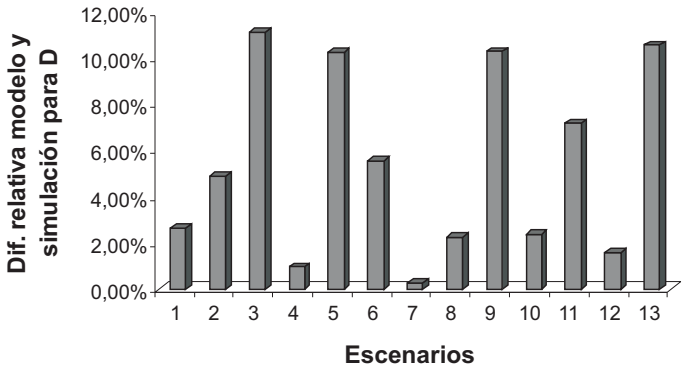


(b)

Fig. 4.5: Valores de T_D obtenidos mediante simulación y mediante el modelo matemático para 13 escenarios diferentes: (a) valores absolutos y (b) diferencias relativas con respecto al valor aportado por el modelo.

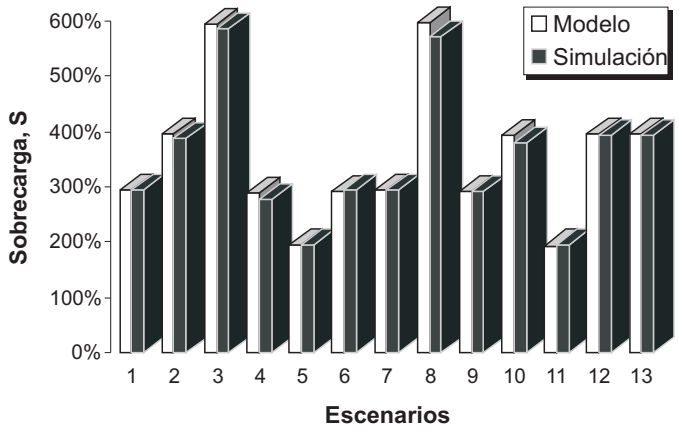


(a)

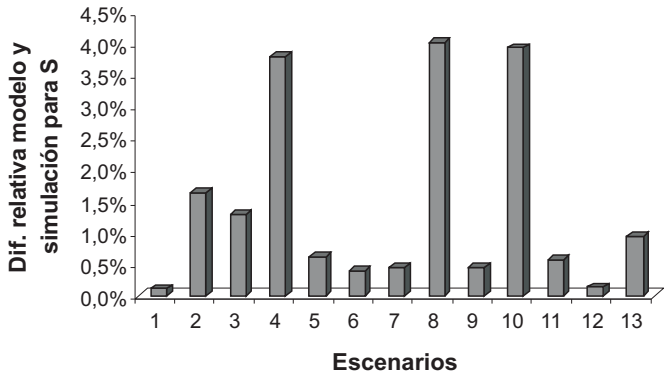


(b)

Fig. 4.6: Valores de D obtenidos mediante simulación y mediante el modelo matemático para 13 escenarios diferentes: (a) valores absolutos y (b) diferencias relativas con respecto al valor aportado por el modelo.



(a)



(b)

Fig. 4.7: Valores de S obtenidos mediante simulación y mediante el modelo matemático para 13 escenarios diferentes: (a) valores absolutos y (b) diferencias relativas con respecto al valor aportado por el modelo.

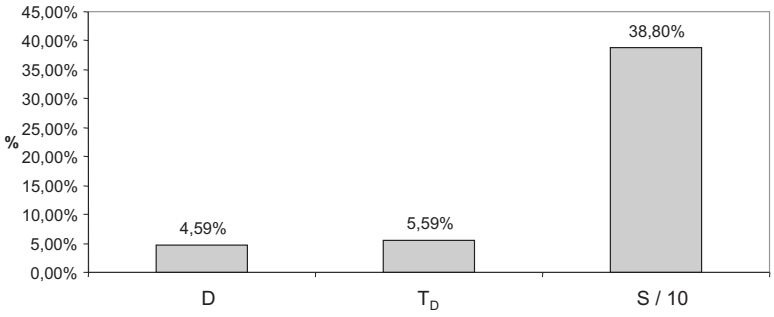


Fig. 4.8: Valor de los indicadores de rendimiento para un ataque ejemplo.

sarrollados para la estimación del rendimiento del ataque DoS a baja tasa contra servidores iterativos.

4.3.2 Capacidades del ataque

Con el fin de comprobar el rendimiento y las capacidades del ataque DoS a baja tasa contra servidores iterativos, se han llevado a cabo un conjunto de ataques con diversas configuraciones en el entorno de simulación. Los resultados obtenidos son reveladores, debido a la alta eficiencia demostrada por el mismo. Como ejemplo, en la Fig. 4.8 se muestran los resultados de una simulación de ataque en la que se han generado 1352 salidas. El ataque se ha ejecutado contra un servidor con un tiempo de servicio $T_s = \mathcal{N}(3\,s; 0.2)$. El tráfico generado por el usuario, con valor $\overline{T_a} = 4\,s$, está próximo al valor que saturaría al servidor. Los parámetros del ataque para este ejemplo son $t_{ontime} = 0.6\,s$ y $\Delta = 0.3\,s$. Por otro lado, se ha configurado un tiempo de ida y vuelta de $RTT = \mathcal{N}(0.6\,s; 0.2)$. Como se puede comprobar, en este ataque de ejemplo se consigue una eficiencia bastante alta, es decir, valores bajos de los indicadores: $D = 4.59\%$ y $T_D = 5.59\%$. Por otro lado, el valor de la sobrecarga para esta configuración de ataque es de $S = 388\%$, lo que indica que el tráfico ofrecido por el atacante al servidor es aproximadamente cuatro veces la capacidad de éste.

Los resultados obtenidos de los diferentes experimentos muestran que existen múltiples configuraciones de ataque posibles de forma que el atacante puede adaptar su ataque a las necesidades que tenga. De este modo, en la Fig. 4.9 se representa, para un subconjunto de los experimentos realizados, el valor de D con respecto a la sobrecarga S necesaria para llevar a cabo el ataque. Como se puede ver, es posible ajustar la sobrecarga del ataque al nivel que se desee para así eludir posibles sistemas de detección. Esto se consigue reduciendo el valor de t_{ontime}, o bien incrementando el valor del *intervalo*, Δ. El coste de reducir la sobrecarga para un ataque será obtener una menor eficiencia.

Por otro lado, aunque los resultados presentados previamente muestran el riesgo potencial de este tipo de ataques, es interesante realizar una comparación entre la estrategia de ataque propuesta, que presenta un cierto grado de sofisticación (estrategia con inteligencia), y otra estrategia consistente en el envío de mensajes de ataque de forma aleatoria en

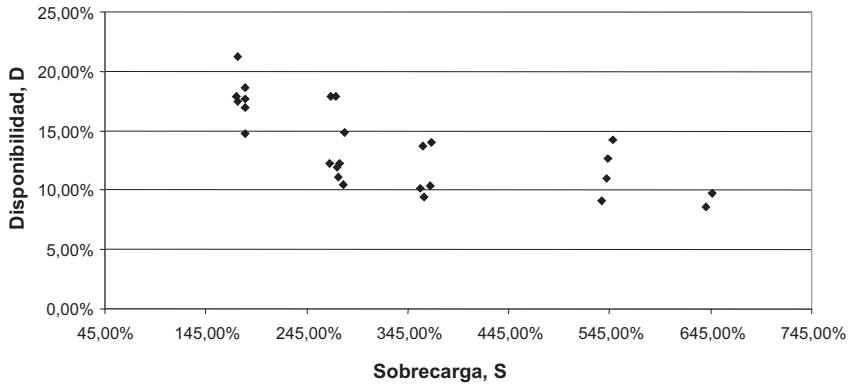


Fig. 4.9: Disponibilidad, D, *vs.* sobrecarga, S, para 25 configuraciones diferentes (valores de t_{ontime} y Δ) del ataque a baja tasa contra servidores iterativos.

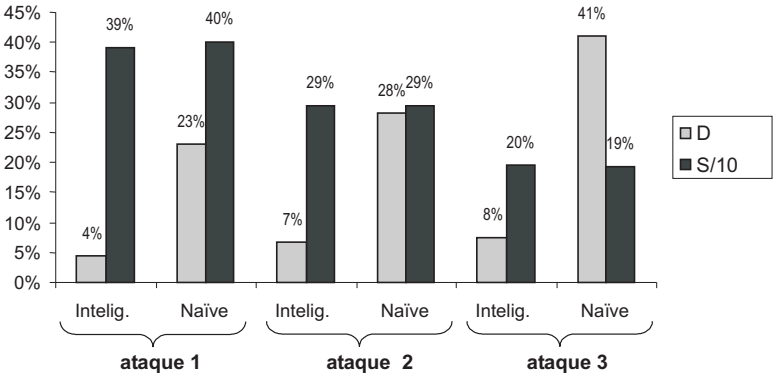


Fig. 4.10: Comparación entre la estrategia inteligente y la *naïve* para tres configuraciones de ataque diferentes.

el tiempo (estrategia *naïve*). Para que los resultados obtenidos mediante ambas estrategias sean comparables entre sí, la tasa media de mensajes enviados al servidor en los experimentos que se realizan debe ser similar, de forma que, para concluir qué estrategia es más efectiva, solamente se comparan las eficiencias respectivas obtenidas.

Las diferencias entre los resultados conseguidos mediante la aplicación de ambas estrategias se muestran en la Fig. 4.10, donde se ilustran los valores obtenidos para la disponibilidad, D, y la sobrecarga, S, correspondientes a tres configuraciones de ataque diferentes. El tiempo de servicio en el servidor se ha configurado como $T_s = \mathcal{N}(5.0\ s; 0.2)$ y se ha utilizado una tasa de tráfico de usuario de $\lambda = 1/T_a = 1/10$ mensajes por segundo. Para cada uno de los tres ataques se han utilizado la estrategia inteligente y la *naïve*. Como se puede observar, los resultados muestran que, aunque los valores de S son similares en ambas estrategias, los valores obtenidos para D indican que se alcanza una mayor eficiencia con la estrategia inteligente propuesta en este trabajo.

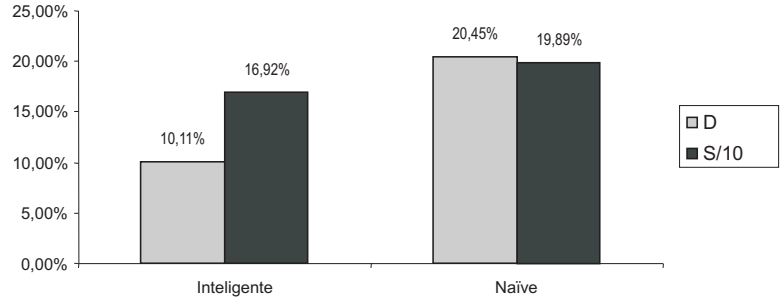


Fig. 4.11: Comparación entre la estrategia inteligente y *naïve* en un servidor saturado por el tráfico de los usuarios.

También se puede comprobar lo que sucedería en el caso de que los usuarios, al observar que el servicio no está accesible, incrementen su tasa de tráfico para poder conseguir así mayor número de capturas en el servidor. Se han realizado algunos experimentos que demuestran que, incluso bajo condiciones en las que el tráfico de usuario llegaría a saturar al servidor por sí mismo, el ataque con la estrategia inteligente obtiene un mayor nivel de denegación de servicio que la *naïve*. En la Fig. 4.11 se muestran los resultados obtenidos para un ataque llevado a cabo contra un servidor con $T_s = \mathcal{N}(5.0\,s; 0.2)$, y una tasa de tráfico de usuario de $\lambda = 1/1.66$ mensajes por segundo, es decir, una tasa que genera saturación en el servidor por sí misma. En ausencia del ataque, se realiza una medición experimental de la disponibilidad, obteniéndose $D = 33\%$. Como se puede comprobar en la figura, la estrategia inteligente reduce el valor de D en un 50% cuando se compara con el obtenido con la estrategia *naïve*, y en un 66% con respecto al nivel de denegación de servicio percibido por los usuarios cuando no existe ataque.

En resumen, el ataque propuesto consigue alcanzar altas cotas de eficiencia en cuanto al nivel de denegación de servicio que se inflige a los usuarios. El diseño que se ha propuesto permite al atacante adaptar su tasa de tráfico hasta alcanzar un umbral en el cual el ataque no es detectable por parte de determinados sistemas de seguridad. Además, la estrategia de ataque aporta unas ventajas evidentes para el atacante cuando se compara con una estrategia de inundación de tipo *brute-force* en la que no se aplica inteligencia alguna.

4.4 Validación en entornos reales

El ataque DoS a baja tasa contra servidores iterativos también se ha probado en un entorno real controlado para observar la viabilidad de su implementación y también para comprobar si los resultados obtenidos en un entorno de simulación son extrapolables a entornos reales.

El servidor seleccionado es un servidor web Apache 2.0, el cual se ha configurado de modo que sirva las peticiones siguiendo una estrategia iterativa. Esto es posible mediante el ajuste de la directiva `ThreadsPerChild = 1` en el fichero de configuración del servidor.

Aunque éste no es un ejemplo típico de un servidor iterativo, sí es útil para el propósito que se ha planteado.

Se asume que una petición de ataque consiste en la solicitud de una conexión con el servidor. El ataque tratará de establecer conexiones pero no envía mensajes en ellas, dejando al servidor que cierre la conexión abierta después de la expiración de un temporizador, cuyo valor está especificado por la directiva de Apache `Timeout`. Este temporizador está configurado, por tanto, con un valor constante y corresponde, en el modelo del servidor, con el tiempo medio de servicio $\overline{T_s}$ que conllevará el procesamiento de cada petición de ataque. Tal y como se discutió en el Capítulo 2, la varianza en el tiempo entre salidas observado por el atacante tendrá dos contribuciones: las variaciones en el tiempo de servicio causadas por la operación del servidor en una máquina con un sistema operativo que no es de tiempo real, y también la variabilidad del tiempo de ida y vuelta, RTT.

El escenario real es análogo al considerado en el análisis teórico. El tráfico de los usuarios se ha generado según un proceso de Poisson. El software de ataque ejecuta éste desde una sola máquina. Tanto las peticiones de usuarios legítimos como las de ataque deben atravesar una red WAN para alcanzar al servidor, con un tiempo de ida y vuelta que, tras su medición experimental, se ha modelado con la distribución $\mathcal{N}(17\,ms; 0.05)$.

Se han establecido trazas tanto en el lado de los usuarios como en el del atacante para obtener los datos necesarios para calcular los indicadores de rendimiento. En la Tabla 4.2 se muestran algunos resultados experimentales y una comparación entre los valores de los indicadores de rendimiento obtenidos en los entornos real y simulado. Los escenarios mostrados difieren en la configuración del tiempo medio de servicio $\overline{T_s}$ y el tiempo entre mensajes de usuario enviados al servidor, $\overline{T_a}$. Estos tiempos se han seleccionado de tal modo que no haya saturación en el servidor cuando no existe el ataque. Los parámetros para el ataque son $t_{ontime} = 0.4\,s$ y $\Delta = 0.4\,s$ para todos los experimentos mostrados.

Se puede observar que en algunos casos se obtiene incluso una mejor eficiencia (D menor) en el entorno real que en el simulado. Además, se puede apreciar que las diferencias en los valores de los indicadores de rendimiento entre ambos entornos son pequeñas y pueden ser debidas a las desviaciones en el modelado de las distribuciones del tiempo de servicio y del tiempo de ida y vuelta.

A modo de resumen, de los resultados presentados en entornos reales se pueden extraer dos conclusiones:

- Los experimentos realizados en entornos de simulación aportan resultados que se pueden considerar buenas aproximaciones al comportamiento obtenido en entornos reales.

- La implementación del ataque no presenta dificultades reseñables. Además, se verifica que el impacto real del ataque es bastante elevado y la vulnerabilidad mostrada en servidores iterativos resulta fácilmente explotable.

$\overline{T}_s$	$\overline{T}_a$	Entorno	*D*	*S*
3	3,5	**Simulación**	10,4%	260,8%
		Real	9,8%	239,7%
5	6	**Simulación**	5,7%	213,1%
		Real	7,8%	212,4%
10	12	**Simulación**	3,0%	198,3%
		Real	6,4%	201,6%
15	17	**Simulación**	3,0%	197,5%
		Real	2,5%	198,2%
20	22	**Simulación**	3,0%	197,5%
		Real	4,3%	196,2%
25	28	**Simulación**	1,8%	197,9%
		Real	1,8%	197,9%

Tabla 4.2: Comparación del rendimiento de diferentes configuraciones de ataque en entornos reales y simulados.

4.5 Mejoras al ataque

Aunque ya se ha comprobado la alta eficiencia alcanzada por los ataques DoS a baja tasa contra servidores iterativos, existen algunos aspectos susceptibles de mejora en el diseño del ataque que un potencial atacante podría abordar con el fin de depurar su modo de ejecución.

En primer lugar, el atacante puede intentar la ocultación del ataque mediante las técnicas conocidas de distribución del mismo. Es decir, convirtiendo el ataque en un DDoS se pueden utilizar las técnicas habituales para ocultar el ataque. Estas técnicas, presentadas en el Capítulo 1 de esta memoria, se pueden resumir en la técnica de saltos, la de establecimiento de múltiples capas de indirección y la falsificación de la dirección origen (*spoofing*).

Por lo que respecta a la técnica de *spoofing* hay que tener en cuenta que el atacante debe recibir las salidas procedentes del servidor para llevar a cabo la técnica de predicción de la siguiente salida. Es por esto que el atacante no podrá realizar un *spoofing* aleatorio de las direcciones origen. Sin embargo, esto no quiere decir que no se pueda utilizar esta técnica, ya que el modo de usarla es realizar la falsificación de la dirección origen cambiándola por otra dirección perteneciente a la misma red que la verdadera dirección origen. Obviamente, esto exige al atacante activar el modo promiscuo en la interfaz de red para recibir las salidas del servidor y actuar en consecuencia. En entornos con movilidad, las posibilidades que tiene el atacante son aún mayores, dado que puede ir cambiando de unas redes a otras y, por tanto, realizando el *spoofing* en el rango de direcciones de cada red que va visitando. En resumen, la técnica de *spoofing*, aunque de forma limitada, puede ser también utilizada aquí.

Por otro lado, en todo el desarrollo de la técnica para realizar el ataque se ha supuesto que las condiciones en las que el servidor trabaja se mantienen estables durante la ejecución del mismo, es decir, el ataque se ejecutará durante todo el tiempo con los parámetros estimados en la fase de análisis. En una máquina real, dichas condiciones pueden variar dinámicamente a lo largo del tiempo. Por ejemplo, la carga de un servidor web depende

fundamentalmente del periodo del día observado. Estos cambios en la carga implican la existencia de un número mayor o menor de elementos de procesamiento libres en el sistema servidor. De esta forma, aunque el atacante solicite un tiempo fijo de servicio, la CPU de la máquina que alberga al servidor compartirá la carga con más o menos procesos, lo que resultará en un tiempo de servicio obtenido mayor o menor, respectivamente. El ataque podría incluir mecanismos para adaptarse a estas condiciones que varían dinámicamente a lo largo del tiempo. Para ello se propone un algoritmo que hemos denominado *algoritmo de media adaptativa*, el cual se detalla a continuación.

4.5.1 Algoritmo de media adaptativa

El atacante estima, durante la fase de análisis, el tiempo de servicio y lo utiliza durante la fase permanente para calcular el parámetro *DSS* según la Expresión (4.2) correspondiente al periodo básico del ataque.

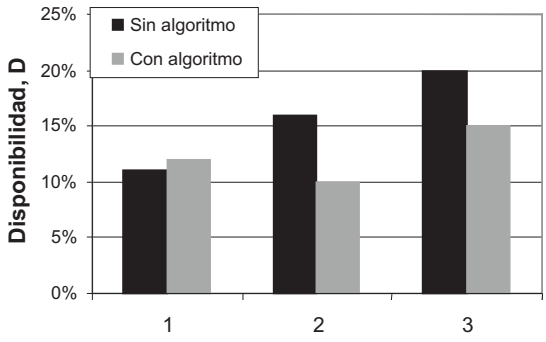
La mejora que el algoritmo de media adaptativa proporciona consiste en estimar dinámicamente el tiempo de servicio a lo largo de la ejecución del ataque. Esto se realiza utilizando un cálculo dinámico que actúa como un filtro paso baja. Cuando se recibe una salida en un periodo de ataque i, el tiempo de servicio en dicho periodo de ataque, $\overline{T_s^i}$, se calcula como el tiempo entre la salida recibida y la anterior (tiempo entre salidas). El tiempo de servicio empleado en la Expresión (4.2) para evaluar la distancia a la siguiente salida, *DSS*, en el periodo básico de ataque $i+1$, $\overline{T_s^{i+1}}$, se calculará del siguiente modo:

$$\overline{T_s^{i+1}} = \begin{cases} \overline{T_s^{i-1}} \cdot (1-K) + \overline{T_s^i} \cdot K & , \quad \text{si } \overline{T_s^i} < \nu \cdot \overline{T_s^{i-1}} \\ \overline{T_s^i} & , \quad \text{en otro caso} \end{cases} \tag{4.4}$$

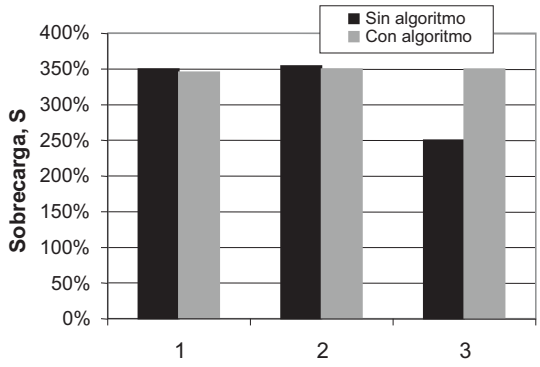
donde $K \in [0,1]$ es un parámetro constante del algoritmo que controla la sensibilidad con la que se modifica el tiempo de servicio estimado, y $\nu \in \Re^+$ es un parámetro que permite ajustar el rango temporal en el que se aplica el algoritmo. La introducción de este parámetro precisa de una explicación adicional.

El objetivo de que $\overline{T_s^i}$, en la Expresión (4.4), pueda tomar dos posibles valores es no sufrir falsas variaciones en la media estimada cuando se producen fallos en las capturas. En efecto, cuando un usuario realiza una captura, pasado el tiempo necesario, la salida correspondiente se enviará a dicho usuario. En este momento, el atacante no observará la salida, por lo que el tiempo entre las dos salidas que recibirá será aproximadamente el doble que el estimado. En este caso, el algoritmo no debe variar su estimación porque se trata de una muestra falsa. Esto implica que el valor con el que se debe ajustar el parámetro ν debería ser ligeramente inferior a 2. Evidentemente, si realmente se produce una variación de esta magnitud en el tiempo de servicio, el algoritmo no será capaz de detectarla y corregir su estimación. Por ello, este algoritmo está diseñado para corregir variaciones pequeñas en el tiempo de servicio. En caso de que se produzcan variaciones mayores, el atacante podría realizar de nuevo una fase de análisis y, tras ella, comenzar la fase permanente.

Para comprobar el funcionamiento del algoritmo presentado, se ha realizado una batería de experimentos. Entre todas las simulaciones realizadas se han extraído tres esce-



(a)



(b)

Fig. 4.12: Rendimiento del ataque para tres escenarios cuando se aplica el algoritmo de media adaptativa y cuando éste no se aplica: a) disponibilidad, D, y b) sobrecarga, S.

narios ilustrativos. Para todos ellos, los parámetros de configuración se han seleccionado de la siguiente forma: $K = 0,8$; $\nu = 1,8$; $T_s = \mathcal{N}(2\ s; 0.2)$; $t_{ontime} = 0.4\ s$; $\Delta = 0.4\ s$; $\overline{T_a} = 3\ s$; y $RTT = \mathcal{N}(0.6\ s; 0.2)$. Los tres escenarios están diferenciados según la forma en que varía el tiempo de servicio durante la ejecución del ataque, y son los siguientes:

Escenario 1. No se producen cambios en $\overline{T_s}$. Este escenario permitirá comprobar si el algoritmo degrada el funcionamiento normal en caso de que no se produzcan variaciones en $\overline{T_s}$.

Escenario 2. Se produce un cambio en la media del tiempo de servicio, de valor $\overline{T_s} = 2.0\ s$ a $\overline{T_s} = 2.2\ s$. En este escenario se comprobará si el algoritmo es capaz de seguir la fluctuación de la media cuando ésta se incrementa.

Escenario 3. Se produce un cambio en la media del tiempo de servicio, de valor $\overline{T_s} = 2.0\ s$ a $\overline{T_s} = 1.5\ s$. En este escenario se comprobará si el algoritmo es capaz de seguir la fluctuación de la media cuando ésta se decrementa.

En la Fig. 4.12 están representados los resultados obtenidos para los indicadores de eficiencia D y S en los tres escenarios. Los resultados correspondientes al primer escenario muestran que se obtienen valores similares cuando se aplica el algoritmo y cuando éste no se aplica. Esto significa que el algoritmo no degrada significativamente el comportamiento normal del ataque. En cuanto a los escenarios 2 y 3, los resultados muestran que se obtiene una mejor eficiencia (menor valor de D). Nótese que, para el escenario 3, la sobrecarga que se obtenía en el caso de no aplicación del algoritmo es menor que la obtenida con el algoritmo. Esto es debido a que, como el valor del tiempo de servicio ha disminuido, en muchas ocasiones la salida llega al atacante antes de que éste haya comenzado el periodo *ontime*, por lo que éste queda interrumpido evitando así la emisión de mensajes de ataque. Cuando se aplica el algoritmo, los periodos de ataque se ajustan mejor a las salidas, obteniendo mayor eficiencia y, por consiguiente, aumentando la sobrecarga, ya que se estará ejecutando un número mayor de periodos *ontime*.

4.6 Conclusiones de este capítulo

En el presente capítulo se han detallado los aspectos relativos a los ataques DoS a baja tasa cuando se realizan contra servidores iterativos.

En primer lugar se ha concretado el método para efectuar el ataque, con la definición de la estrategia seguida para explotar la vulnerabilidad existente en los servidores iterativos, las fases de ejecución del ataque y el modo en que los diferentes parámetros del ataque son configurados.

Además, se ha realizado una evaluación del rendimiento del ataque, en base a los indicadores de rendimiento definidos en el Capítulo 3. En esta evaluación se ha comprobado cómo se comporta el ataque ante la variación de sus parámetros de diseño y del comportamiento del servidor. Por otro lado, se han validado los resultados con los modelos matemáticos para los indicadores de rendimiento en sistemas sin superposición y se ha comprobado que el ataque no solamente alcanza altas cotas de eficiencia, sino que también es fácilmente implementable en un entorno real.

Por último, se han presentado algunas propuestas que mejoran el comportamiento del ataque en cuanto a su rendimiento y también se han hecho ciertas consideraciones respecto a la ocultación que el atacante puede realizar con el fin de sortear potenciales mecanismos de seguridad.

Capítulo 5

El ataque DoS a baja tasa contra servidores concurrentes

5.1 Introducción

Un servidor concurrente se caracteriza por servir las peticiones de forma paralela, ya sea real (concurrencia real) o virtualmente (concurrencia aparente), es decir, para el procesado de una petición no hay que esperar necesariamente a que se terminen de procesar otras previas. Este tipo de servidor se puede representar con el modelo propuesto en el Capítulo 2, simplemente haciendo cumplir la condición de que el número total de elementos de procesamiento que componen el servidor sea mayor que la unidad ($N_s > 1$).

Ya se ha concluido también en el Capítulo 2 que los sistemas concurrentes son sistemas que siempre presentan superposición, esto es, en los que existen al menos dos salidas consecutivas cuyas curvas de probabilidad de ocurrencia se solapan. Este hecho hace que, con respecto al diseño general del ataque presentado en el Capítulo 3, existan ciertos detalles por resolver cuando el atacante se enfrenta a la implementación del mismo.

Partiendo de los fundamentos para la ejecución del ataque DoS a baja tasa contra servidores, presentados en el Capítulo 3, en este capítulo se desarrolla el método con el que un atacante puede utilizar dichos mecanismos para efectuar la denegación de servicio a baja tasa contra un servidor de tipo multiproceso o concurrente. Las diferentes conclusiones que se presentarán a lo largo del capítulo se apoyan en resultados experimentales obtenidos tanto en entornos simulados como reales, describiéndose, en su caso, los experimentos de los que se derivan.

La estructura del capítulo es la siguiente. En primer lugar se exponen los detalles de implementación del ataque en el Apartado 5.2. Seguidamente, en el Apartado 5.3 se presenta una evaluación del rendimiento del mismo, utilizando para ello tanto entornos de simulación como reales. En el Apartado 5.4 se proponen algunas mejoras que optimizan el

funcionamiento del ataque presentado. Finalmente, las conclusiones del capítulo se resumen en el Apartado 5.5.

5.2 Implementación del ataque contra servidores concurrentes

La implementación del ataque DoS a baja tasa contra cualquier servidor multiproceso se apoya en la estrategia básica de ataque propuesta en el Capítulo 3. Así, el primer paso en su ejecución es predecir los instantes en los que se producen las salidas en el servidor para, en segundo lugar, utilizar una serie de ondas de ataque de tipo ON/OFF sincronizadas, en la llegada al servidor, con cada salida que éste genera. De esta forma, se pretende mantener la cola de servicio completamente ocupada con las peticiones procedentes del atacante, impidiendo la ocupación de sus posiciones por parte de usuarios legítimos.

Existen, por tanto, dos aspectos básicos a tener en cuenta en la implementación del ataque contra un servidor concurrente. Primero, hay que determinar la vulnerabilidad y el método que hace posible reducir la tasa del ataque mediante la predicción de los instantes en los que suceden las salidas y, segundo, se debe especificar el procedimiento para explotar dicha vulnerabilidad de forma que, a partir de ello, se pueda llevar a cabo el ataque.

5.2.1 Análisis de la vulnerabilidad en los sistemas concurrentes

Se puede afirmar que, al contrario que en el ataque DoS a baja tasa contra servidores iterativos, en el caso de los servidores multiproceso el conocimiento de los tiempos entre salidas no constituye una vulnerabilidad que permita predecir el instante en que se producirá una salida posterior. Una justificación de esta afirmación se presenta a continuación.

Tal y como se ha fundamentado en el Capítulo 2, cuando se considera un sistema con superposición, el tiempo entre salidas tiene una alta variabilidad con respecto a su valor medio (ver Fig. 2.10). Si en este tipo de sistemas, para la estimación del instante en que se va a producir una salida, se toma el correspondiente a la última salida más la media de la distribución del tiempo entre salidas –dada por la Expresión (2.12)–, la diferencia entre el instante estimado y el real podrá ser elevada, debido a que la varianza de la distribución del tiempo entre salidas es considerable. Así, si se quisiera mitigar la desviación entre los instantes estimado y real mediante el uso de un periodo de ataque de tipo ON/OFF, el periodo de actividad debería ser tan grande que se generaría una sobrecarga tal durante el ataque que haría que éste no se pudiera identificar como de baja tasa.

Este hecho hace necesario que el atacante, para llevar a cabo su ataque, deba buscar una vulnerabilidad diferente a la posible estimación del tiempo entre salidas en el servidor. A diferencia del caso del servidor iterativo, en el que el tiempo entre salidas es una característica común a todos ellos, para los servidores concurrentes no se ha encontrado una característica genérica que aporte información sobre los instantes en los que se producen las oportunas salidas.

Aunque la afirmación anterior induzca a concluir que, por tanto, no es posible ejecutar el ataque DoS a baja tasa contra los servidores concurrentes, esto no es así. En efecto, dicha afirmación implica únicamente que, para ejecutar el ataque contra un servidor concurrente concreto, es necesario encontrar una vulnerabilidad propia de dicho servidor que permita estimar los instantes de las salidas. El atacante deberá, por tanto, realizar un estudio del comportamiento específico del servidor a atacar con el fin de detectar los potenciales patrones de actividad temporal que posibiliten la predicción de dichos instantes.

Aunque a primera vista pueda parecer complicado encontrar vulnerabilidades de este tipo, la realidad es que en numerosos servidores de tipo concurrente es posible localizar dichos patrones de actividad. A continuación citamos algún ejemplo.

Supóngase un servidor concurrente cuyo servicio consiste en realizar el envío de información de un video publicitario a los usuarios que, bajo demanda, soliciten la reproducción de dicho video. El servidor está diseñado de modo que existe un número máximo de clientes que pueden solicitar a la vez la reproducción del video. En este escenario, el hecho de que la reproducción tarde un tiempo fijo constituye una vulnerabilidad que se puede explotar mediante las técnicas expuestas en este trabajo. En efecto, dado que el tiempo de reproducción es fijo, un potencial atacante puede predecir, conocido el instante inicial de ésta, cuándo terminará y, por tanto, cuándo un nuevo usuario podrá solicitar nuevamente la reproducción. Una vez predicho este instante, el atacante deberá utilizar la estrategia presentada en el Capítulo 3 para mantener ocupado al servidor en la reproducción continua de una instancia del video. Si esto se consigue para el número máximo de solicitudes de servicio permitidas, ningún otro usuario tendrá acceso, lográndose así la denegación de servicio.

Si el atacante consigue tener en reproducción continua el número máximo de videos ningún otro usuario tendrá acceso al servicio, por lo que se habrá conseguido denegar el servicio.

Si bien en el ejemplo del servidor anterior el servicio ofrecido suele ser no orientado a conexión, también en los servicios orientados a conexión es posible encontrar este tipo de vulnerabilidades. En efecto, toda vez que las características de las conexiones que el servicio ofrece muestren un patrón temporal fijo existe una vulnerabilidad explotable. Considérese, por ejemplo, un servidor de noticias que envía notificaciones a todo usuario que ha hecho una subscripción (mediante una conexión) al servicio. En caso de que exista un número máximo de suscripciones y de que cada subscripción expire en un tiempo fijo, la posibilidad de conocer dicho tiempo constituye una vulnerabilidad explotable por un potencial atacante, del mismo modo que en el anterior ejemplo.

En definitiva, el abanico de servidores atacables dependerá de las características de diseño de éstos. Como regla general, siempre que el servidor presente un esquema temporal fijo que pueda ser conocido por parte de un potencial atacante, este hecho constituye una vulnerabilidad que permitirá ejecutar un ataque DoS de baja tasa contra el mismo.

Dentro de los servicios orientados a conexión es importante destacar un servicio ampliamente utilizado en *Internet* que presenta una vulnerabilidad de este tipo: el ofrecido por los servidores HTTP [Fielding et al., 1997]. Además, si dichos servidores presentan la característica de conexiones persistentes [Mogul, 1995], el atacante podrá obtener mayores

beneficios para la ejecución del ataque. Más adelante, debido a su importancia, se describirán los detalles de este servicio y la vulnerabilidad correspondiente.

En todos los desarrollos que se harán a lo largo de este capítulo se detallarán los mecanismos generales necesarios para ejecutar el ataque DoS a baja tasa contra servidores concurrentes y, cuando por mayor claridad se requiera un mayor nivel de detalle, se concretará para el caso del servidor HTTP persistente debido a su relevancia y uso extendido.

Caso de estudio: vulnerabilidad en el servidor HTTP

Debido a la importancia que el servicio HTTP tiene en entornos de comunicaciones como *Internet*, se realiza a continuación un análisis detallado de la vulnerabilidad que permite ejecutar un ataque DoS de baja tasa contra este tipo de servidores. Hay que resaltar que, tal y como se ha comentado anteriormente, este estudio persigue solamente realizar una particularización del método general de ataque a los servidores concurrentes aplicado al caso del servidor HTTP, constituyendo éste simplemente un escenario de ejemplo.

El servidor HTTP, cuando no implementa la funcionalidad de persistencia en las conexiones (la cual se explicará más adelante), funciona del siguiente modo: cada conexión entrante es introducida en una cola, en espera de ser extraída por algún elemento de procesamiento del servidor. Para cada uno de ellos, siempre que se obtiene una conexión de la cola, se procesa la petición HTTP correspondiente y se cierra la conexión. Seguidamente, se pasará a extraer una nueva conexión de la cola. Nótese que el papel que juega una conexión en el modelo propuesto para el servidor es el de un mensaje de ataque y, por otro lado, una salida en el modelo consiste en este ejemplo en el cierre de una conexión.

En este contexto, siempre que el atacante conozca el tiempo invertido por los elementos de procesamiento en procesar una petición concreta, T_s (p.ej. porque la petición siempre sea la misma[1]), podrá predecir que, cada vez que se emite una salida en el servidor (la cual es recibida $\overline{RTT}/2$ s después por el atacante), la siguiente salida se producirá T_s segundos después. Por tanto, deberá utilizar los mecanismos presentados en el Capítulo 3 para ejecutar el ataque DoS a baja tasa.

De especial importancia, debido a que presenta unas características especiales que favorecen la ejecución del ataque DoS a baja tasa, es el servidor HTTP persistente. La característica de persistencia en un servidor HTTP permite que un usuario específico, realizando una sola conexión, pueda enviar sobre ella varias peticiones consecutivas. La gran diferencia con la versión no persistente del protocolo radica en que no hay necesidad de establecer una conexión para cada una de las peticiones que se envían. Dado que se ahorran muchos establecimientos y cierres de conexión, las principales ventajas de este modo de funcionamiento son el aumento de la tasa con la que se sirven las peticiones y la reducción del ancho de banda consumido en el desarrollo del servicio.

[1] Nótese que la presencia de una *cache* podría modificar el tiempo invertido por el servidor en dos peticiones idénticas y también haría que éstas no llegaran al propio servidor. Aunque esto es cierto, no supone un problema para ejecutar el ataque, pues el atacante podría sortear la *cache* solicitando páginas que no se almacenan en ésta (como las páginas de conexiones seguras HTTPS).

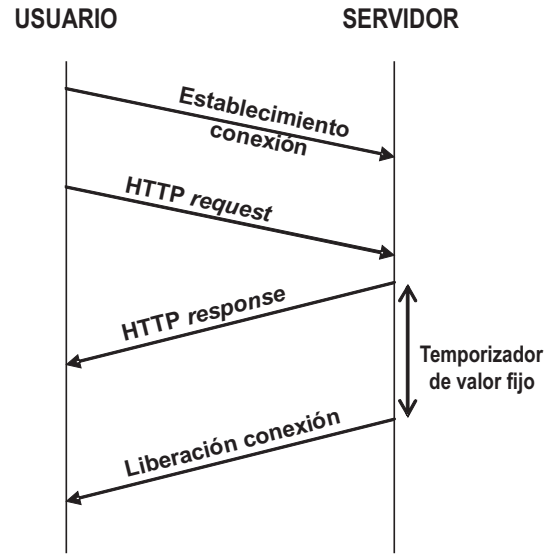


Fig. 5.1: Servidor HTTP persistente y temporizador de cierre de la conexión.

La evolución de las peticiones de un usuario que desea solicitar varios recursos a un servidor HTTP persistente es la siguiente (Fig. 5.1):

1. *Establecimiento de la conexión*:

   Dado que HTTP, cuando implementa la característica de persistencia, se convierte en un servicio orientado a conexión, en primer lugar el usuario establece una conexión con el servidor. En una implementación POSIX de UNIX, los datos de la conexión se almacenarán en un *socket*. La adecuación al modelo del servidor descrito en el Capítulo 2 es clara: la cola de servicio está contenida en el *socket*, de modo que cada conexión que se realiza entre usuario y servidor ocupa una posición en dicha cola, independientemente del número de peticiones HTTP que se emitan sobre ella. El número de peticiones solamente afectará al tiempo de servicio para dicha conexión.

2. *Envío de las peticiones HTTP*:

   Pasado un tiempo de espera en la cola de servicio, la conexión pasa al módulo de servicio. De nuevo en una implementación POSIX de UNIX, esto se produce cuando el nivel de aplicación extrae la conexión de la estructura *socket* mediante la llamada al sistema (*syscall*) *accept*. En este momento, un elemento de procesamiento consulta si en la conexión existe o no una petición HTTP. En caso de que no exista ninguna, pasado un temporizador fijo se procederá al cierre de la conexión. Si, por el contrario, existe alguna petición, ésta se procesa y se envía la respuesta a través de la conexión. Una vez que se ha respondido, la conexión no se cierra, sino que se mantiene abierta y el elemento de procesamiento espera otra posible petición sobre la misma conexión. El tiempo de espera tiene un valor fijo, cuyo valor se asigna en la configuración del servi-

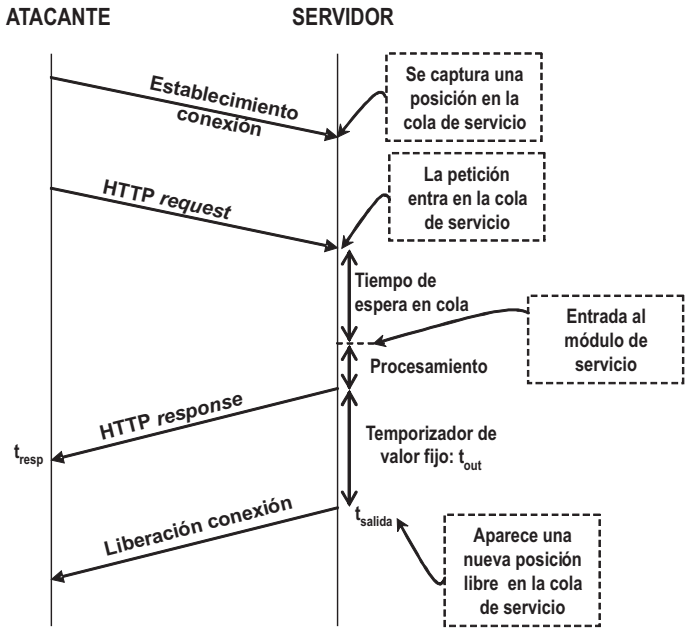


Fig. 5.2: Diagrama del procedimiento seguido para la estimación del instante de una salida en un servidor HTTP persistente.

dor[2]. Si alguna nueva petición llega antes de que expire este contador, es procesada y el temporizador se reinicia de nuevo.

3. *Cierre de la conexión*:

   Cuando el temporizador que se inicia tras el procesado de la última petición expira, la conexión se cierra y el elemento de procesamiento queda libre para extraer otra solicitud de la cola de servicio. En este momento es cuando, según el modelo propuesto del servidor, consideramos que se produce una salida, ya que este evento corresponde con la liberación de una posición en la cola.

Entendido el funcionamiento del servidor HTTP persistente, es sencillo percatarse de que cualquier atacante puede descubrir, con unas simples pruebas, el valor del temporizador de cierre de la conexión persistente. Lo preocupante es que, a partir de la existencia de este valor, se pueden estimar los instantes en los que se producirán las salidas para, consecuentemente, ejecutar el ataque a baja tasa contra el servidor. Un esquema del proceso que permite realizar esta estimación se encuentra representado en la Fig. 5.2 y consiste en los siguientes pasos:

Paso 1. El atacante establece una conexión con el servidor. Esta conexión ocupará una posición en la cola de servicio.

[2] En un servidor Apache 2.0, la directiva *KeepAliveTimeout* en el fichero de configuración *http.conf* especifica el valor de este temporizador.

Paso 2. El atacante envía una petición (*HTTP request*) al servidor sobre la conexión establecida.

Paso 3. La conexión establecida espera en la cola de servicio, durante un tiempo no fijo que dependerá del contexto en el que la conexión llega al servidor, su turno para entrar en el módulo de servicio.

Paso 4. Tras el tiempo de cola, un elemento de procesamiento extrae la conexión, procesa la petición enviada sobre ella y responde con el resultado oportuno (*HTTP response*), el cual llegará al atacante en el instante t_{resp}.

Paso 5. Una vez enviado el mensaje HTTP *response* se inicia un temporizador de valor fijo, t_{out}, en el elemento de procesamiento.

Paso 6. Cuando el temporizador expira, la conexión se cierra (instante t_{salida}). Dado que se está considerando que cada conexión ocupa una posición en la cola de servicio, su cierre corresponde en el modelo del servidor a la generación de una salida.

En este proceso, para predecir el instante en el que se produce la salida, t_{salida}, el atacante sólo tendrá que considerar el instante en que se recibe la respuesta *HTTP response* desde el servidor, t_{resp}, y el valor del temporizador de cierre de la conexión persistente, t_{out}. Tal y como se justificó en el Capítulo 2, aunque este tiempo esté configurado como un valor constante, tendrá cierta variabilidad debido a las condiciones de ejecución en el servidor, por lo que en adelante se representará como una variable aleatoria: T_{out}. Dado que este temporizador se aplica a todas las conexiones que se realizan con el servidor, dicha variable aleatoria tendrá una distribución que se puede modelar según la Expresión (2.3). Consistentemente, dado que el temporizador se comporta como una variable aleatoria, también el instante en que se produce una salida lo será. Así, la variable aleatoria que representa el instante en que se produce una salida, T_{salida}, presentará la siguiente distribución de probabilidad:

$$T_{salida} = \mathcal{N}\left(t_{resp} - \frac{\overline{RTT}}{2} + \overline{T_{out}}; var[T_{out}] + var\left[\frac{RTT}{2}\right]\right) \tag{5.1}$$

Nótese que la varianza del instante de la salida viene dada por la variabilidad que puede sufrir T_{out}, al ser considerado en un entorno determinado con unas condiciones cambiantes (concurrencia) y también por las variaciones que puede experimentar RTT. En este ejemplo de servidor, el papel que juega el temporizador, T_{out}, se asemeja al del tiempo de servicio, T_s, en el modelo del servidor. Dado que ambas magnitudes, RTT y T_{out}, se han modelado como variables aleatorias normales en el modelo –Expresiones (2.2) y (2.3)–, el resultado para el tiempo de salida es también una variable aleatoria distribuida según una normal.

En el caso del servidor HTTP persistente, es importante resaltar la necesidad de que, una vez establecida la conexión con el servidor, se envíe al menos una petición sobre ella antes de que se produzca su cierre. En efecto, el atacante no puede intentar simplemente una conexión al servidor y esperar a que ésta se cierre una vez expirado el temporizador. El motivo es que, una vez llega la solicitud de conexión al servidor, ésta se encolará en la

cola de servicio durante un tiempo que el atacante no podrá predecir. Solamente cuando se realiza el paso de la petición al módulo de servicio y se emite la respuesta se inicia el temporizador para el cierre de la conexión. Es por esto por lo que es necesario, por parte del atacante, monitorizar el instante t_{resp}, y esto sólo se podrá realizar mediante el envío previo de, al menos, una petición *HTTP request*.

Tanto en el caso del servidor HTTP cuando es persistente y cuando no lo es, el hecho que permite predecir los instantes de las salidas al atacante es la existencia de un temporizador de valor determinista. Ahora bien, el caso del servidor persistente es más beneficioso para el atacante por dos motivos: en primer lugar, para cada conexión realizada, en el servidor persistente se puede estimar cuándo se producirá su salida con un margen de desviación $var[T_s] + var[RTT/2]$. En el servidor no persistente, por el contrario, solamente se puede estimar que, para un elemento de procesamiento que ha emitido una salida, la siguiente que genere se producirá en un tiempo T_s con igual margen de desviación que en el anterior caso. Ahora bien, esto será cierto solamente si la siguiente petición que el elemento de procesamiento en cuestión procese pertenece al atacante y no a un usuario legítimo. En caso contrario, se producirá una mayor desviación en la predicción. Por consiguiente, en el caso del servidor HTTP no persistente se reducen los fallos de captura debido a que las desviaciones serán menores. En segundo lugar, el servidor HTTP persistente permite mantener la conexión activa durante más tiempo que en el no persistente. Para ello basta con enviar varias peticiones consecutivas sobre la misma conexión. Es notable el beneficio de esta medida: cuanto más tiempo se ocupe una posición en la cola de servicio del servidor, menor será el esfuerzo para realizar la denegación de servicio, ya que el número de capturas a realizar disminuirá.

Este estudio también permite concluir un hecho importante: la condición previa para poder realizar la predicción del instante de una salida es poseer una posición en la cola del servidor (en el caso HTTP una conexión con el mismo). Esta conclusión, aunque se ha ilustrado para el caso del servidor HTTP (persistente o no), es generalizable a cualquier servidor concurrente. Esto es evidente por el propio mecanismo de predicción del ataque. Es decir, el atacante es capaz de predecir el instante de una salida porque conoce el intervalo de tiempo (valor fijo) desde que sucede un determinado evento (realización de una conexión, comienzo de la reproducción de un video, etc.), hasta que se produce la correspondiente salida. Por tanto, la condición para que se pueda realizar la predicción es que dicho evento haya ocurrido con anterioridad, lo que típicamente equivaldrá a capturar previamente una posición en la cola del servidor para provocar la salida en cuestión.

Por último, hay que insistir de nuevo en el hecho de que la vulnerabilidad y el procedimiento para explotarla en el caso del servidor HTTP no son más que un ejemplo acerca del método que un atacante puede seguir para ejecutar el ataque contra un servidor concurrente. Para cualquier otro tipo de servidor, el atacante deberá, de igual forma, descubrir un esquema temporal determinista en el funcionamiento del servidor que permita la predicción de los instantes en los que se producirán las salidas.

5.2.2 Procedimiento de ejecución del ataque

Una vez detectada la vulnerabilidad que permite estimar los instantes de ocurrencia de las salidas en un servidor concurrente, el ataque contra el mismo se puede llevar a cabo siguiendo la estrategia definida en el Capítulo 3. En efecto, una vez que se ha realizado la predicción del instante en el que se va a producir una salida, hay que programar un periodo básico de ataque en torno a dicho instante de tal modo que los mensajes de ataque lleguen al servidor sincronizados con el momento en que se produce la salida. Esto implica ajustar los parámetros del periodo básico de ataque según los datos obtenidos en una fase previa de análisis de las carácterísticas del servidor.

Aunque, de forma general, este es el procedimiento de ataque, para mayor claridad se presentan a continuación, a modo de ejemplo y para el caso particular del servidor HTTP persistente, las consideraciones a tener en cuenta para realizar el ajuste del periodo básico de ataque:

- El comienzo del periodo básico de ataque, *CPA*, será (como en el caso de los servidores iterativos) el instante t_{resp}, en el que se recibe el mensaje *HTTP response*, dado que es en este momento en el que se predice el instante de la salida correspondiente.

$$CPA = t_{resp} \tag{5.2}$$

- Para el cálculo de la distancia a la siguiente salida, *DSS*, se considerará el valor medio de la variable aleatoria que determina el instante en que dicha salida ocurrirá, T_{salida} –Expresión (5.1)– de modo que su expresión será:

$$DSS = \overline{T}_{salida} - CPA = \overline{T_{out}} - \frac{\overline{RTT}}{2} \tag{5.3}$$

- El valor de $t_{offtime}$ se calculará, en cualquier caso, en base a la Expresión (3.3).
- Los parámetros t_{ontime} y Δ se ajustarán, también de forma general, para obtener un rendimiento específico en el ataque, en términos de la eficiencia y la sobrecarga que se pretenda obtener.

Durante el tiempo de ejecución del ataque se realizará la programación de un periodo básico de ataque de forma independiente para todas y cada una de las salidas cuyos instantes se van estimando a lo largo del tiempo. De este modo, el ataque conseguirá minimizar el tiempo disponible durante el que existen posiciones en las colas de servicio para que no se puedan ocupar por parte de cualquier usuario.

Por tanto, a la hora de implementar este procedimiento de ejecución, el atacante debe programar un periodo de ataque por cada instante estimado para la aparición de una salida. Ahora bien, la implementación de este mecanismo precisa resolver algún problema adicional que a continuación se presenta.

Debido a que las salidas se producen en un sistema con superposición, puede suceder que los instantes en que se generan estén bastante separados entre sí o, quizás al contrario,

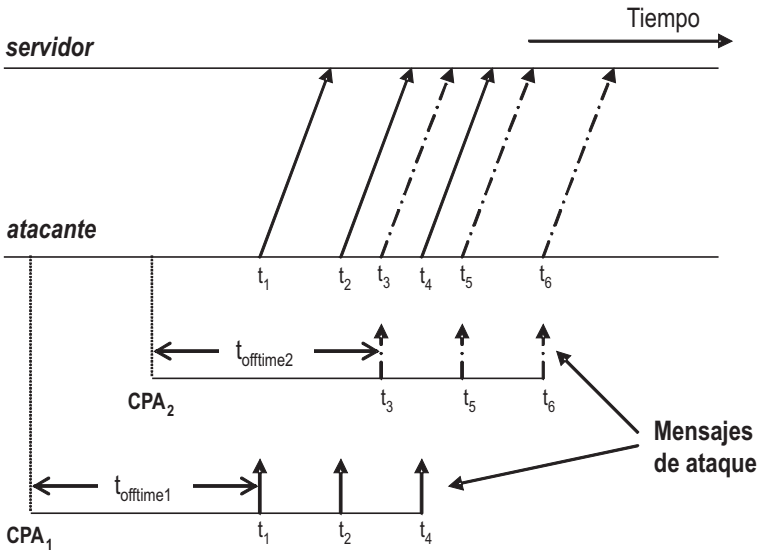


Fig. 5.3: Diagrama con la superposición de dos periodos de ataque.

que se produzcan en un rango de tiempo muy reducido. Cuando esto último sucede, la ejecución de los periodos básicos de ataque correspondientes a las parejas de salidas consecutivas que se producen con escasa separación entre sí también sufren solapamiento cuando dichos periodos llegan al servidor.

En la Figura 5.3 se muestra un diagrama de ejemplo en el que dos periodos de ataque se solapan entre sí. En ella se puede observar en primer lugar la posición del inicio de sus correspondientes periodos de ataque (CPA_1 y CPA_2). Ambos periodos tienen igual estructura y parámetros, de modo que cuando expiran sus temporizadores de inactividad, $t_{offtime1}$ y $t_{offtime2}$, se inicia la transmisión de mensajes de ataque (representados mediante flechas verticales, continuas para el periodo 1 y discontinuas para el periodo 2). Se puede comprobar que el hecho de que las salidas se produzcan en un periodo breve de tiempo provoca una superposición en los periodos *ontime* de las ondas de ataque, tanto a la salida del atacante como en la llegada al servidor.

El control de los diferentes periodos que se van generando, conforme las salidas se predicen por parte del atacante, se puede complicar si se pretende realizar su gestión mediante un solo proceso que envía todos los mensajes de ataque al servidor. Es por esto que se propone una arquitectura para el software atacante (*malware*) en la que existirá un número determinado de hebras activas de ataque, N_a. Cada una ellas está encargada de ejecutar periodos de ataque sobre instantes de salida estimados, de modo que una hebra lleva el control para solo un periodo de ataque en cada momento. De esta forma, la ejecución en paralelo del ataque evita problemas de ajuste que aparecerían por la superposición de periodos *ontime* cuando se utiliza un sistema no paralelo.

Por tanto, en una fase inicial de diseño[3], el software atacante está constituido por diferentes hebras que se ejecutan independientemente entre sí. El objetivo de cada una de ellas será mantener al menos una posición ocupada en la cola de servicio. Las hebras de ataque deben realizar, por tanto, dos tareas diferentes: a) estimación de los instantes de las salidas, y b) ejecución del periodo básico de ataque en torno a ellos.

Es importante reiterar que cada hebra se encargará de ejecutar el periodo de ataque para una sola salida a la vez. De esta forma, el comportamiento de todas las hebras de ataque será idéntico y se efectuará en paralelo durante la ejecución del ataque.

Seguimiento de capturas

Durante un periodo básico de ataque, la hebra de ataque que lo controla emitirá un número de paquetes N_p determinado por el tamaño del periodo de actividad t_{ontime} y por el intervalo Δ según la Expresión (3.2). Cuando se concluye la ejecución de cada periodo básico de ataque se pueden obtener dos posibles resultados:

- *El periodo básico de ataque ha tenido éxito*:

  Un periodo básico de ataque ha tenido éxito cuando, a su finalización, la hebra de ataque en cuestión ha conseguido introducir al menos una petición en alguna cola de servicio, ocupando el hueco liberado por una salida.

  En este proceso de captura de una posición puede producirse un fenómeno singular. Cuando una hebra captura una posición en la cola, dicha hebra puede no ser capaz de comprobar si la posición corresponde a la salida cuyo instante había estimado. Esto es debido a que, cuando existe superposición, se pueden capturar las posiciones que persiguen otras hebras en lugar de la esperada, sin que la hebra de ataque perciba la diferencia entre ambas.

  Un escenario que ilustra la ocurrencia de este fenómeno se muestra en la Fig. 5.4. En ella se pueden observar dos periodos de ataque ejecutados por dos hebras diferentes, e iniciados en los instantes CPA_1 y CPA_2, respectivamente. Se comprueba en la figura cómo la salida correspondiente a la hebra 1 es capturada por la 2 y viceversa, es decir, ambas hebras tienen éxito en la ejecución de sus periodos de ataque, pero no perciben que las posiciones capturadas no son las que pretendían inicialmente.

  Este fenómeno, aunque aparentemente inocuo, dado que al final cada hebra de ataque consigue su objetivo, que no es más que capturar una nueva posición, sí puede ser relevante en determinadas situaciones, como se verá a continuación.

- *El periodo básico de ataque no ha tenido éxito*:

  También es posible que se produzca un fallo en la captura de la posición en la cola de servicio. Esto se puede producir porque algún usuario legítimo haya conseguido introducir su petición en la posición liberada en la cola, o también porque, debido al fenómeno anteriormente descrito y representado en la Fig. 5.4, otra hebra, cuyo

[3] En el Apartado 5.4 se plantearán algunas modificaciones a este diseño que mejorarán el rendimiento del ataque.

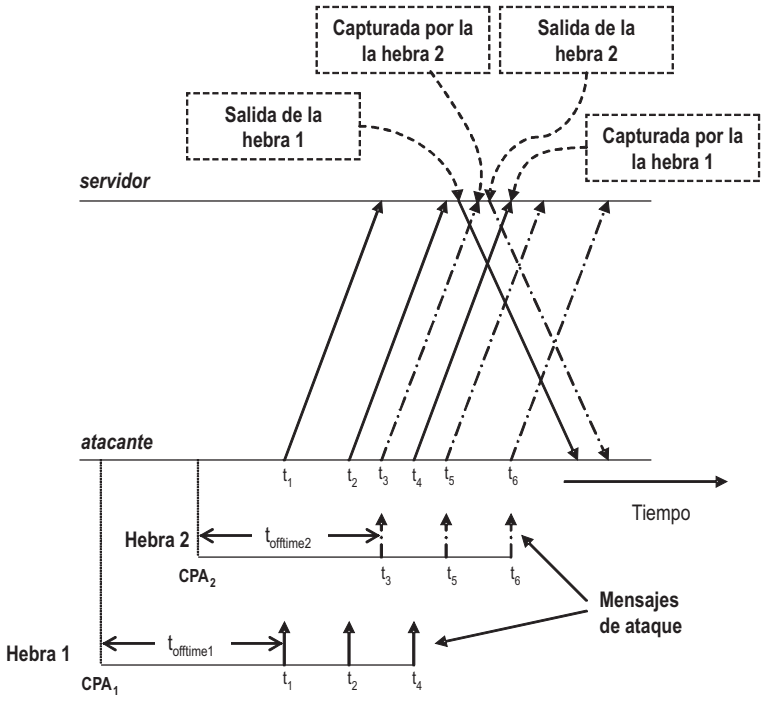


Fig. 5.4: Escenario de capturas en el que las hebras capturan posiciones que no son las que pretenden inicialmente.

periodo de ataque estaba superpuesto con el de la hebra en cuestión, haya capturado la posición liberada como consecuencia de la ejecución de su periodo de ataque.

Es importante señalar que, como se ha justificado anteriormente, dado que la condición previa para poder realizar la estimación del instante en que se produce una salida es capturar una posición en las colas del servidor, cuando se falla en la realización de la conexión, la hebra quedaría sin la posibilidad de estimar un nuevo instante de salida y, por tanto, de ejecutar un nuevo periodo de ataque en torno a dicho instante.

Nótese que la forma en que cada hebra distingue los dos posibles resultados, es decir, si ha tenido o no éxito en la captura, dependerá del tipo de servidor atacado, de su vulnerabilidad y de la técnica utilizada para explotarla. En el caso particular de un servidor HTTP, es la recepción del mensaje SYN/ACK lo que confirma que ha habido éxito en la ocupación de la cola, mientras que la recepción del mensaje RST indica lo contrario.

Denominaremos *estrategia de seguimiento de capturas* a aquella estrategia que determina el modo en que cada hebra de ataque se comporta ante los posibles resultados de la ejecución de un periodo básico de ataque. La estrategia de seguimiento de capturas debe especificar, de forma unívoca, el comportamiento que la hebra seguirá tanto si se produce éxito como si hay fallo en la captura de la posición deseada.

Estrategia básica de seguimiento de capturas

Se propone a continuación la *estrategia básica para el seguimiento de capturas*. Ésta se especifica del siguiente modo:

Caso a) *Fallo en la captura de una posición*:

Para el caso en el que la hebra detecta el fallo del periodo básico de ataque en una captura, la estrategia básica establece que la hebra debe realizar el envío de un mensaje de ataque cada intervalo de tiempo denominado *intervalo de recuperación*, Δ_r, con el fin de retomar una posición en las colas del servidor. Este comportamiento se mantiene hasta que la hebra tiene éxito y consigue su objetivo. En este punto, se recupera el modo normal de funcionamiento de ataque, es decir, se predice el instante de ocurrencia de la siguiente salida y se programa un nuevo periodo de ataque.

Caso b) *Éxito en la captura de una posición*:

Dado que el objetivo de cada hebra es conseguir durante el máximo tiempo posible una posición en la cola, una vez que se realiza la captura de una posición, en un primer estadío, el comportamiento más lógico sería detener la ejecución del periodo de actividad *ontime* y pasar a la estimación del instante de la siguiente salida, tal y como se hace en la versión del ataque contra servidores iterativos.

Ahora bien, ya se ha indicado que, debido al efecto de la superposición de los periodos de ataque, la hebra de ataque que ocupa una posición en la cola no puede saber si dicha posición era la que había estimado que iba a capturar o bien corresponde con la que intenta capturar otra hebra. Con esta consideración en mente, hay que indicar que el comportamiento propuesto en primera aproximación, para determinados escenarios, puede suponer un problema.

La Fig. 5.5 ilustra un escenario en el que se produce esta circunstancia. En ella se representan dos periodos de ataque superpuestos ejecutados por las hebras 1 y 2. Se puede observar que, en este caso, la salida correspondiente a la hebra 1 es capturada por la hebra 2. Debido a esto, la hebra 2 finaliza su periodo de actividad *ontime* y se dedica a la estimación de la siguiente salida a partir de la captura recientemente realizada. A su vez, la hebra 1 fallará en la captura de la posición pretendida, por lo que enviará un mensaje de ataque cada Δ_r (el siguiente será en $t = t_4 + \Delta_r$). El problema que aparece en este escenario es el siguiente. Al haber terminado la hebra 2 su periodo de ataque, la salida que estaba tratando de capturar deja de estar atendida. Esto provoca que el tiempo disponible que se genera tras dicha salida aumente, reduciéndose de este modo la eficiencia.

Para resolver este problema, para la estrategia básica de seguimiento se propone mantener el periodo de *ontime* hasta que éste termine, con independencia de si se recibe una notificación de captura de la salida producida en el servidor o no. Una vez que ha terminado este periodo de *ontime*, si se ha capturado alguna posición en la cola, se considera ésta para estimar su instante de salida. Esta estrategia hará que aumente (un poco) la eficiencia a costa de aumentar también un poco la sobrecarga. Nótese que el hecho de no terminar la fase de *ontime* cuando se recibe una salida en el atacante

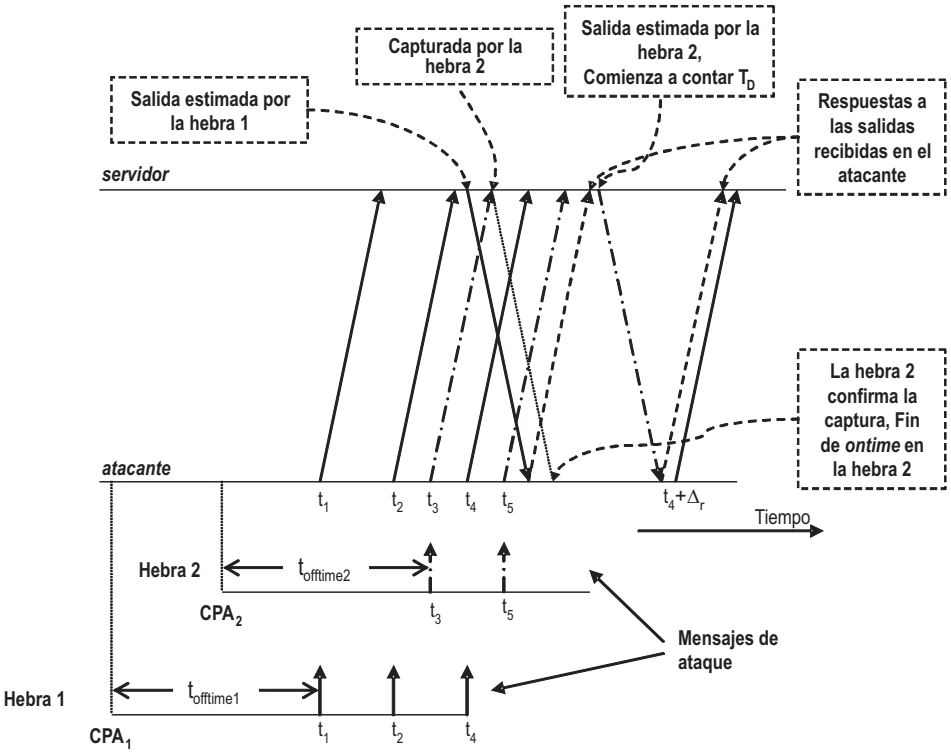


Fig. 5.5: Estrategia de interrupción de *ontime* cuando se realiza una captura.

supone una modificación con respecto a la estrategia general del ataque especificada en el Capítulo 3.

Por otro lado, hay que indicar que la magnitud de esta situación no es tan grande como pueda parecer a priori, ya que el número de escenarios en el que se produce es muy reducido. En la Fig. 5.6 se muestran los resultados de simulación correspondientes a 11 escenarios en los que se han configurado temporizadores de expiración de la conexión persistente T_{out} sin varianza, $var[T_{out}] = 0$, así como también $var[RTT] = 0$, con el fin de reproducir la problemática planteada y que el escenario de la Fig. 5.5 se produzca muchas veces a lo largo del tiempo. En este experimento, sólo los escenarios 8 y 10 cumplen los requisitos para que el problema se reproduzca. Se presentan los resultados obtenidos para el porcentaje de tiempo disponible generado durante la simulación para ambas opciones: a) continuar con la fase de *ontime* aunque se reciba la captura, y b) finalizar la fase de *ontime* cuando se recibe la captura. Se puede observar claramente la diferencia obtenida en el porcentaje de tiempo disponible entre las dos opciones para los escenarios 8 y 10, mientras que en el resto el comportamiento es prácticamente igual. Es decir, cuando ocurren determinados escenarios, que además son difíciles de reproducir, se obtienen estas diferencias, pero en el resto de los casos la eficiencia obtenida será similar.

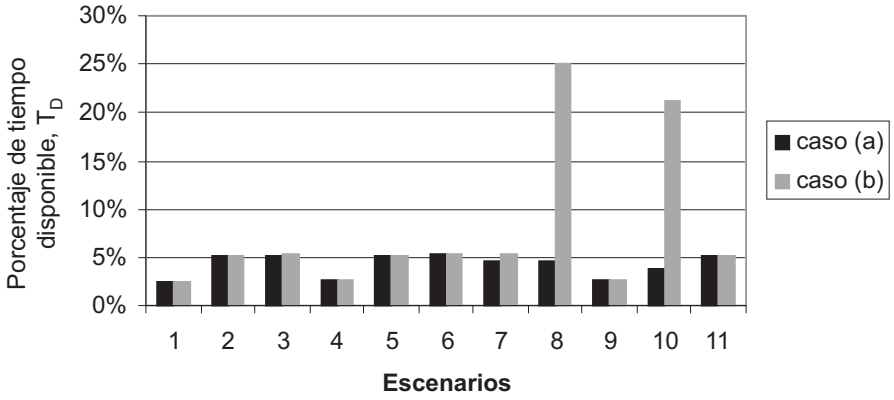


Fig. 5.6: Resultados del tiempo disponible obtenido para 11 escenarios diferentes para las dos posibles opciones de comportamiento en la estrategia básica cuando el atacante recibe una captura: (a) continuar con la fase de *ontime* cuando se captura una posición, y (b) finalizar la fase de *ontime* ante la captura de una posición.

Cuando se realiza el ataque contra un sistema real, en el que los tiempos son no deterministas — $var[T_{out}] \neq 0$, $var[RTT/2] \neq 0$ en la Expresión (5.1) —, los efectos de este problema se diluyen, ya que la variabilidad de los tiempos hace que los escenarios problemáticos, si suceden, no lo hagan un número elevado de veces, provocando una afección global poco considerable.

En resumen, el problema planteado no afectará considerablemente en un ataque real. Aun así, la consideración de esta situación ha permitido adoptar un criterio en la estrategia de seguimiento de capturas que define el modo de actuar cuando se realiza la captura de una posición. Este criterio, a modo de resumen, establece que ante la recepción de la correspondiente salida procedente del servidor, el atacante no terminará la ejecución de su fase *ontime*.

5.3 Evaluación del rendimiento del ataque

En este apartado se pretende realizar una evaluación del rendimiento del ataque DoS a baja tasa contra servidores concurrentes. Para ello, se utilizarán los indicadores de evaluación del ataque ya propuestos en el Capítulo 2: para estimar la cantidad de tráfico necesario para llevar a cabo el ataque se utilizará la sobrecarga, S; cuando se evalúe la eficiencia considerando la afectación a los usuarios legítimos se utilizará como indicador la disponibilidad, D, y cuando se estudie la eficiencia de forma independiente al comportamiento de dichos usuarios, el indicador idóneo será el porcentaje de tiempo disponible, T_D.

Para realizar la evaluación se ha implementado, en el simulador *Network Simulator 2* [Fall and Varadhan, 2007], un módulo de ataque con las características presentadas en el

Apartado 5.2, un servidor concurrente acorde con el modelo propuesto y también un módulo de generación de tráfico de usuario según el modelo presentado en el Capítulo 2.

Finalmente, hay que indicar también que la evaluación del ataque se ha particularizado para una implementación del escenario correspondiente al servidor HTTP persistente. La extensión a otros tipos de servidores dependerá de las características del ataque en dichos entornos.

5.3.1 Variaciones con los parámetros de diseño

Para obtener un conocimiento detallado del comportamiento del ataque en función del ajuste de los diferentes parámetros de configuración del servidor y del propio ataque, se ha realizado un estudio que determina las variaciones del rendimiento cuando los parámetros considerados varían. Los parámetros en cuestión para los que se ha realizado el estudio son los siguientes:

- Intervalo de recuperación: Δ_r.
- Número de hebras de ataque implementadas en el software atacante: N_a.
- Varianza de las desviaciones en la sincronización entre la llegada del periodo básico de ataque al servidor y el instante de generación de la salida. Nótese que dichas desviaciones vendrán dadas por las asociadas a la estimación del instante de la salida –Expresión (5.1)–, más las correspondientes al viaje del periodo de ataque entre atacante y servidor. Por ello, en total, dichas desviaciones son $var[T_{out}] + var[RTT]$.
- Intervalo entre la emisión de paquetes durante *ontime*: Δ.
- Duración de la fase de actividad dentro del periodo de ataque: t_{ontime}.

A continuación se presentan y discuten varios experimentos llevados a cabo en los que se ha realizado la variación de uno de los parámetros, dejando fijos los valores del resto. Los valores por defecto, cuando no se produce variación en ellos, se encuentran recopilados en la Tabla 5.1. Las Fig. 5.7 a 5.11 muestran los resultados de los diferentes experimentos realizados para comprobar evolución de los indicadores S y T_D con respecto a la variación de los anteriores parámetros. Para una mejor interpretación se ha representado en las figuras la magnitud $O = 100 - T_D$, que aportará, en lugar del porcentaje de tiempo durante el que el servidor está disponible para aceptar alguna nueva petición, el porcentaje de tiempo en el que el servidor se encuentra en estado de saturación. Esta magnitud representa, por tanto, la eficiencia que el ataque alcanza en términos del tiempo que se mantiene en saturación al servidor.

El primer experimento considera la variación con respecto al intervalo de recuperación, Δ_r. Algunos resultados se recogen en la Fig. 5.7 . En este caso, se puede observar claramente que la magnitud O mejora (mayor valor) cuanto menor es el valor de Δ_r. Del mismo modo, los valores pequeños para el intervalo de recuperación generan una mayor sobrecarga, es decir, una mayor carga en el sistema debido al ataque. Este comportamiento era esperable, ya que cuantos más mensajes de ataque se envíen, mejor será la eficiencia del ataque y mayor la sobrecarga.

Parámetro	Valor
Elementos de procesamiento en el servidor (N_s)	8
Hebras de ataque (N_a)	14
Número total de posiciones en las colas de servicio	14
Tiempo de ida y vuelta (RTT)	$\mathcal{N}(0.2\ s; 0, 1)$
Intervalo de recuperación (Δ_r)	1 s
Tiempo de expiración de la conexión persistente (T_{out})	$\mathcal{N}(12\ s; 0, 1)$
$t_{offtime}$	11,8 s
t_{ontime}	0,4 s
Intervalo (Δ)	0,2 s

Tabla 5.1: Valores por defecto para los experimentos de evaluación del comportamiento del ataque.

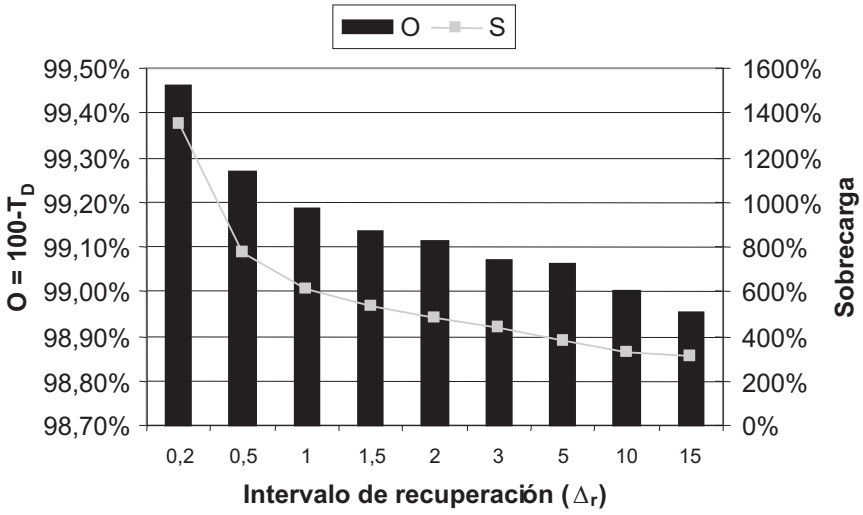


Fig. 5.7: Comportamiento del ataque con respecto a variaciones en el intervalo de recuperación, Δ_r.

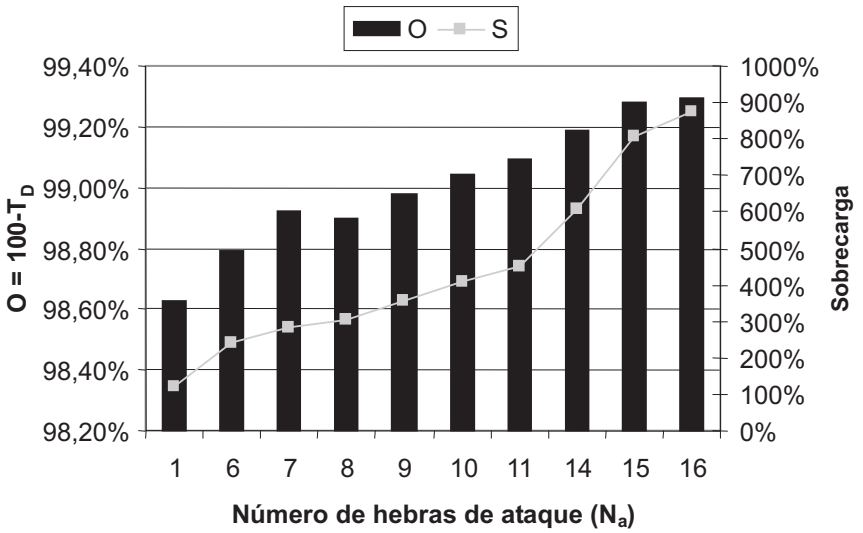


Fig. 5.8: Comportamiento del ataque básico con respecto a la variación del número de hebras de ataque, N_a.

La Fig. 5.8 recopila algunos resultados de los experimentos en los que se realiza una variación en el número de las hebras de ataque, N_a. En este caso se puede apreciar claramente cómo un aumento de N_a incrementa también tanto su eficiencia como la carga generada. Esto es debido a que la existencia de más hebras generará más mensajes y, por tanto, mayor eficiencia y sobrecarga. Nótese que, dado que el ataque se ha diseñado para que cada hebra capture una posición en la cola y mantenga dicha posición a lo largo del tiempo, el valor óptimo de N_a sería, en principio, coincidente con el total de posiciones en las colas de servicio, tal y como se ha configurado por defecto en el ataque ejemplo (Tabla 5.1). Ahora bien, el hecho de que las hebras de ataque puedan capturar más de una posición en las colas de servicio debido a que envían varios mensajes durante *ontime* hace que sea posible que el número óptimo para N_a sea distinto del número total de posiciones en las colas de servicio. En realidad, tal y como se muestra en la Fig. 5.8, a partir del número total de posiciones en la cola, en el ejemplo $N_a > 14$, el crecimiento de la sobrecarga parece más acentuado, debido a que habrá más hebras de ataque enviando mensajes de ataque cada Δ_r para recuperar posiciones en las colas de servicio. La magnitud de este crecimiento dependerá del valor establecido para el parámetro Δ_r.

La discusión anterior pone de manifiesto la dificultad, por parte del atacante, de elegir correctamente el valor para el parámetro N_a. Si bien en este punto del trabajo no se aporta más que una visión del problema, más adelante (Apartado 5.4.2) se propondrá un método para la determinación de este parámetro del ataque.

Para el experimento en que se modifican los valores de las varianzas que causan las desviaciones en la sincronización entre el instante de la salida y la llegada del periodo

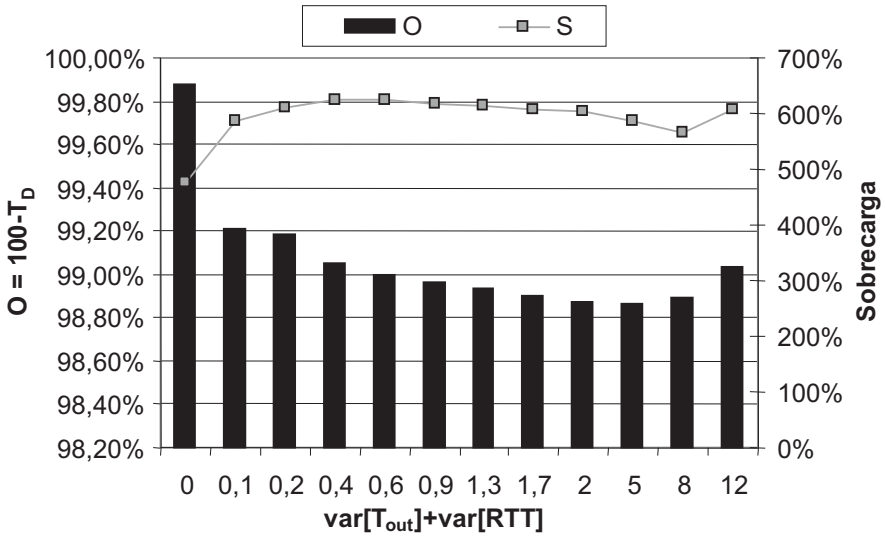


Fig. 5.9: Comportamiento del ataque básico con respecto a variaciones en las desviaciones en la sincronización entre la llegada del periodo de ataque al servidor y el instante en que se produce la salida, $var[T_{out}] + var[RTT]$.

básico de ataque al servidor, $var[T_{out}] + var[RTT]$, se muestran algunos resultados en la Fig. 5.9. En este caso es destacable que, cuando aumenta la varianza total, no se produce una disminución significativa del valor de la eficiencia. La justificación para este comportamiento es que, a diferencia del ataque contra servidores monoproceso, en los cuales podría no existir superposición, en este caso aparece superposición de las salidas. Esto hace que, cuando la varianza sea grande, la desviación en la sincronización de unos periodos de ataque se vea compensado por la llegada de otros periodos superpuestos, es decir, algunas posiciones serán capturadas por otras hebras que no están haciendo el seguimiento de la salida que las origina. En resumen, el comportamiento general que presenta el sistema cuando la varianza aumenta consiste en una disminución de la eficiencia, aunque ésta no es significativa debido a la interacción de unos periodos de ataque con otros. En cuanto a la sobrecarga, S, se puede observar que tampoco presenta una variación significativa con respecto a $var[T_{out}] + var[RTT]$, excepto para un valor 0 de varianza, en cuyo caso las hebras envían muchos menos mensajes en los periodos de ataque debido a su conocimiento exacto del instante de la salida.

En la Fig. 5.10 se puede comprobar el comportamiento de los indicadores del ataque ante variaciones del parámetro intervalo, Δ. Como era previsible, cuando el valor del parámetro Δ aumenta, la eficiencia disminuye, así como también la sobrecarga, dado que se envían menos mensajes durante el periodo *ontime* –según la Expresión (3.2)–. Por el contrario, una disminución del valor de Δ lleva aparejadas las consecuencias contrarias.

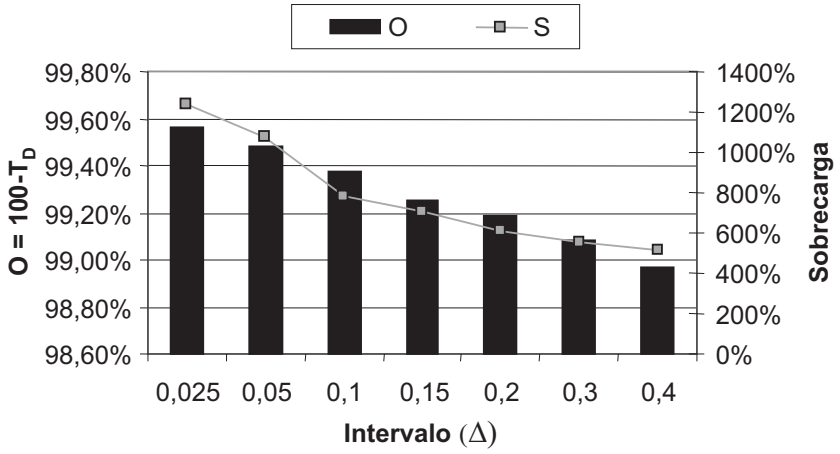


Fig. 5.10: Comportamiento del ataque básico con respecto a modificaciones en los valores del intervalo, Δ.

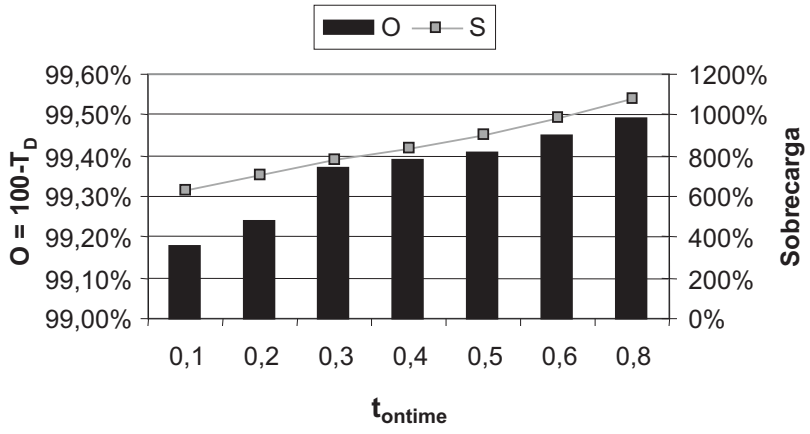


Fig. 5.11: Comportamiento del ataque básico con respecto a variaciones en el parámetro de ataque t_{ontime}.

Finalmente, la variación de la duración de la fase de actividad (t_{ontime}) dentro del periodo de ataque produce resultados como los que se muestran en la Fig. 5.11. Se observa que, al aumentar el tiempo de dicha fase, la eficiencia aumenta, obteniéndose mayores valores de ocupación a la vez que, consecuentemente, la sobrecarga es mayor. Esto es debido a que durante el periodo de actividad se emitirá un mayor número de mensajes de ataque. Por el contrario, cuando el periodo t_{ontime} disminuye, el comportamiento del sistema es justamente el inverso, como era de esperar.

Validación de los resultados del modelo matemático para sistemas con superposición

Los experimentos anteriormente presentados sobre la variación de parámetros del ataque y del servidor de forma independiente aportan una idea general sobre las tendencias en el comportamiento del ataque. Ahora bien, estos resultados, tal y como se han presentado, no se pueden generalizar debido a que, por un lado, el número de parámetros es considerable y, por consiguiente, no se han contemplado todos los escenarios posibles, ya que su número resulta excesivamente elevado. Por otro lado, las variaciones de los parámetros no se han realizado conjuntamente, sino sólo de forma independiente.

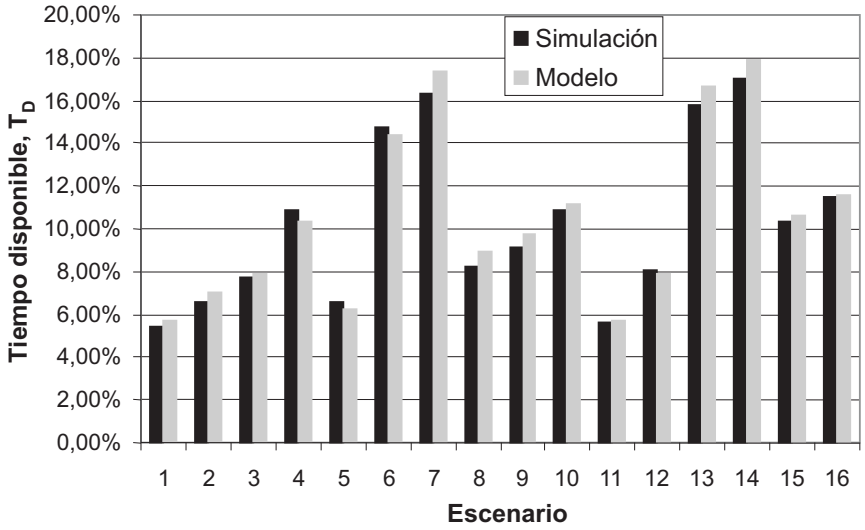
Así pues, para llevar a cabo la validación de los resultados experimentales obtenidos, la herramienta fundamental que permitirá extender estas primeras conclusiones a cualquier otro escenario es el modelo matemático para los indicadores de rendimiento del ataque en sistemas con superposición, presentado en el Capítulo 3. Este modelo presenta la relación analítica entre los diferentes indicadores de rendimiento con respecto a los parámetros del ataque y del sistema servidor.

Se puede comprobar, evaluando las expresiones del modelo, que los resultados aportados en los experimentos anteriores sobre la variación de los parámetros del ataque y del servidor son coherentes con el funcionamiento derivado del modelo.

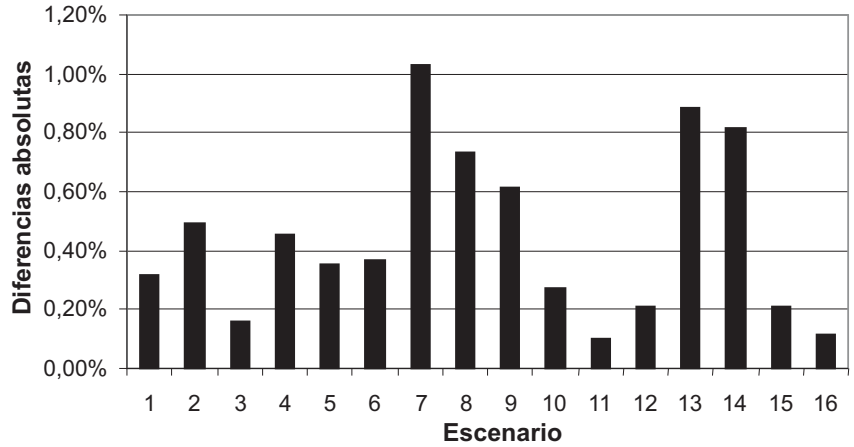
Por otro lado, para poder determinar la utilidad del modelo matemático, es necesario también comprobar si solamente las tendencias en los valores de los indicadores son reflejados por el mismo, o si también son estimados correctamente los valores absolutos de los indicadores del rendimiento. Con este fin se han realizado simulaciones con diferentes configuraciones de ataque, modificando los parámetros del ataque y del servidor.

En la Fig. 5.12 se representan los valores obtenidos para T_D en simulación y los derivados del modelo para 16 escenarios diferentes. En todos los escenarios representados, el número de elementos de procesamiento en el servidor es de cuatro ($N_s = 4$). Se ha comprobado que los valores aportados por el modelo, en los rangos de valores utilizados, se aproximan bastante bien a los obtenidos mediante simulación, existiendo una diferencia absoluta máxima entre ellos de 1,03%, con un valor promedio de 0,44%.

Además, también se ha efectuado la comprobación entre la diferencia de los valores obtenidos del modelo y los de un entorno de simulación para el indicador de rendimiento D. En la Fig. 5.13 se muestran los resultados obtenidos para 12 escenarios diferentes, con una comparativa de los valores obtenidos por ambos métodos. En los diferentes escenarios considerados en los experimentos se ha hallado una diferencia máxima absoluta de 3,86%



(a)



(b)

Fig. 5.12: Comparativa de valores absolutos de tiempo disponible obtenidos mediante simulación y mediante el modelo matemático para 16 escenarios diferentes: a) valores obtenidos en ambos casos y b) diferencias absolutas entre ambos valores para cada escenario.

y un promedio de 2,27%, lo que indica que los resultados del modelo se ajustan a los de simulación.

Finalmente, el mismo experimento se ha realizado para el indicador de sobrecarga, S, obteniéndose también como resultado que el modelo representa considerablemente bien el comportamiento en simulación. Los resultados correspondientes a 13 escenarios diferentes están representados en la Fig. 5.14. En este caso, se han obtenido una diferencia absoluta máxima de 4,18%, con un promedio de 1,92%.

En resumen, los resultados de la comparación del modelo matemático con entornos de simulación nos indican que el modelo matemático responde bastante bien al comportamiento simulado. Un ajuste fino en las aproximaciones realizadas podría llevar a obtener resultados más aproximados con el modelo.

5.3.2 Capacidades del ataque

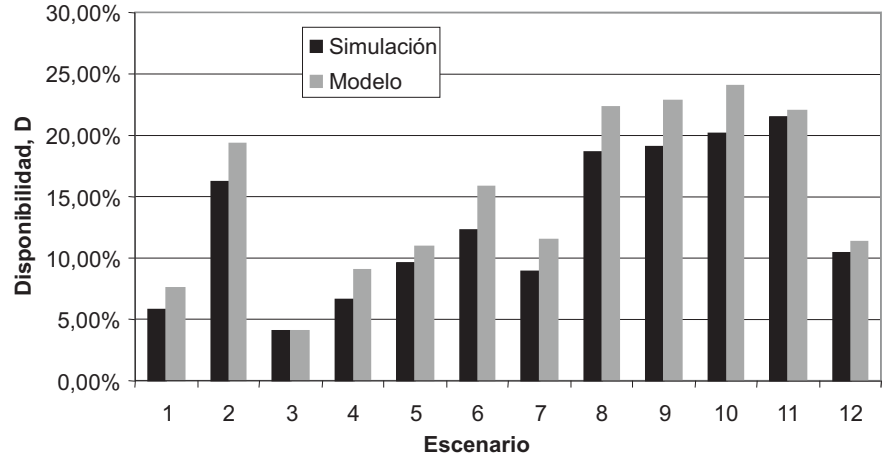
Para tener una percepción de la magnitud del impacto que podría suponer sufrir un ataque DoS a baja tasa contra un servidor concurrente es necesario evaluar las capacidades potenciales del mismo, es decir, qué rendimiento puede alcanzar.

Los resultados obtenidos y presentados en anteriores apartados, relativos a la ejecución del ataque en diferentes escenarios y configuraciones, muestran, en primer lugar, que es factible la realización del ataque. Por otro lado, también se puede comprobar que es posible conseguir diferentes tasas de ataque y distintos niveles de eficiencia dependiendo de la configuración de parámetros elegida para el ataque. La Fig. 5.15 muestra los rendimientos obtenidos, en función de D y S, para 18 configuraciones diferentes. Se puede observar que el atacante, mediante la configuración de los parámetros de ataque, puede elegir el punto de trabajo de éste para obtener la máxima eficiencia posible, a la vez que se mantiene la sobrecarga por debajo de un umbral con objeto de evitar ciertos mecanismos de seguridad que puedan proteger al servidor.

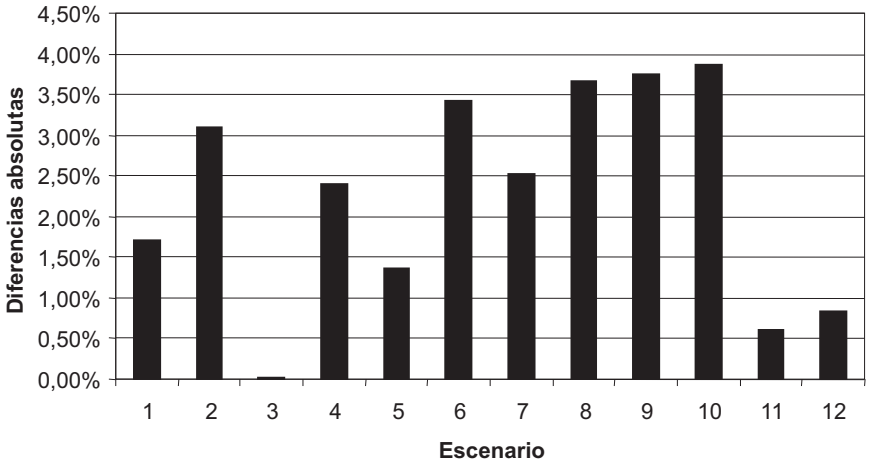
También se puede observar en la Fig. 5.15 que el atacante es capaz de obtener eficiencias muy buenas (valores de D bajos), con sobrecargas aceptables (en torno al 250% de la capacidad del servidor). Además, incluso cuando la sobrecarga es inferior al 100%, algunos valores de eficiencia obtenidos son aún preocupantes ($D = 50\%$).

Aunque los resultados ya presentados muestran que el ataque puede alcanzar una eficiencia muy elevada, también cabe preguntarse si el hecho de haber aplicado una estrategia "inteligente" al ataque produce una mejora considerable en cuanto a su eficiencia. Es decir, hay que comprobar cuál sería la eficiencia obtenida por este tipo de ataques en comparación con algún ataque que utilice la misma tasa pero que no aplique ningún tipo de inteligencia (ataque "naïve").

Para clarificar este punto se ha realizado un conjunto de experimentos sobre diferentes escenarios y configuraciones de ataque. En todos ellos se ha aplicado primeramente un ataque inteligente con las técnicas presentadas en este capítulo y se han obtenido los resultados correspondientes. Posteriormente se ha realizado un ataque *naïve* en los mismos escenarios. Este último se ha configurado de modo que utiliza una tasa de mensajes de

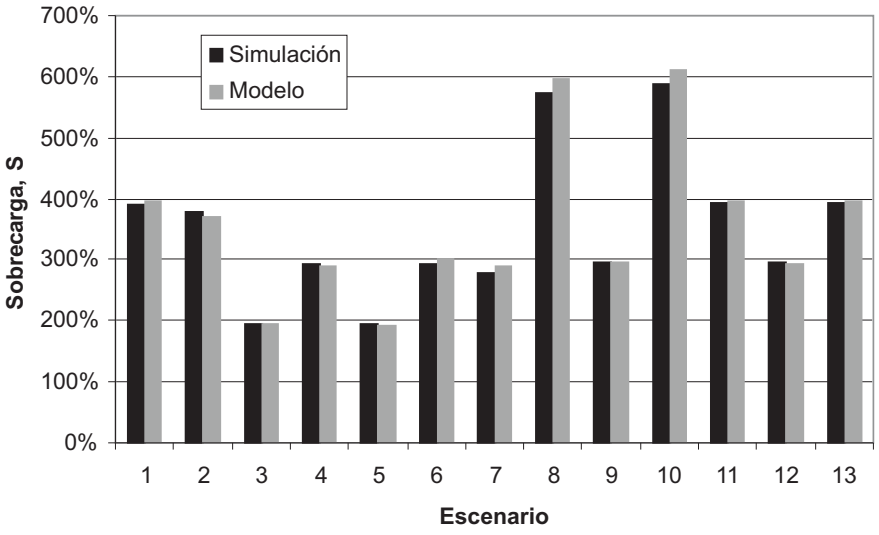


(a)

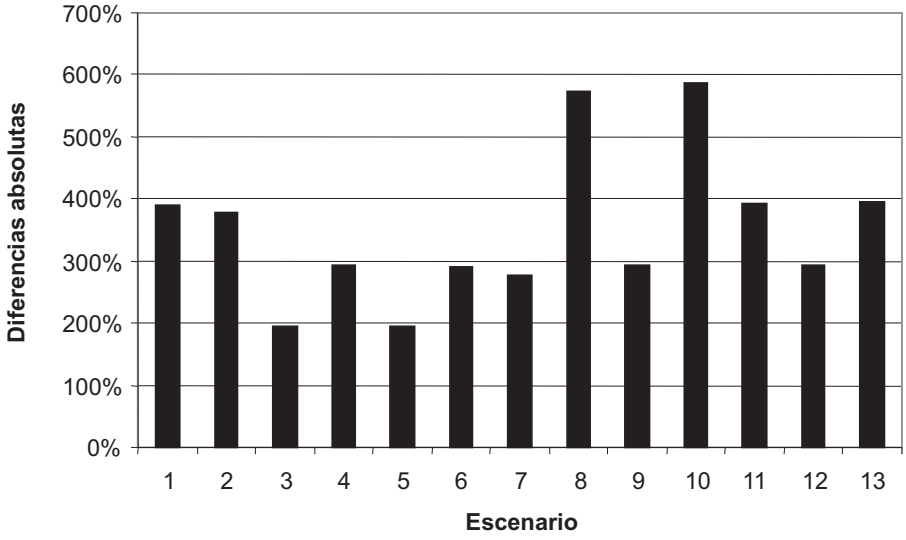


(b)

Fig. 5.13: Comparativa de valores absolutos de D obtenidos mediante simulación y mediante el modelo matemático para 12 escenarios diferentes: a) valores obtenidos en ambos casos y b) diferencias absolutas entre ambos valores para cada escenario.



(a)



(b)

Fig. 5.14: Comparativa de valores absolutos de S obtenidos mediante simulación y mediante el modelo matemático para 13 escenarios diferentes: a) valores obtenidos en ambos casos y b) diferencias absolutas entre ambos valores para cada escenario.

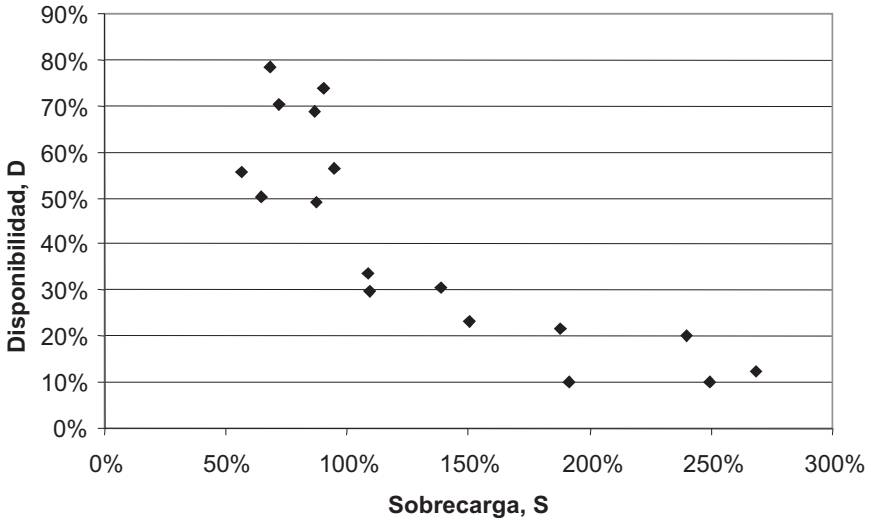


Fig. 5.15: Rendimientos obtenidos para 18 configuraciones diferentes de ataque.

ataque (sobrecarga) mayor a la del inteligente, pero realiza el envío de dichos mensajes de forma aleatoria según una distribución de *Poisson*. Aunque para realizar la comparación entre las eficiencias obtenidas por ambas estrategias hubiera sido deseable configurar los correspondientes parámetros de modo que utilicen idénticos valores de sobrecarga, este ajuste es complicado de obtener, y por este motivo es por el que se ha optado por configurar el ataque *naïve* con una sobrecarga siempre más elevada que la del ataque inteligente (peor caso para éste último).

Algunos resultados obtenidos en estos experimentos para los indicadores S y D se pueden observar en la Fig. 5.16. En ella se comprueba que, para los 4 escenarios representados, aunque la sobrecarga implicada por el ataque inteligente es menor que la del ataque *naïve*, los resultados en cuanto a eficiencia son mejores en el ataque inteligente, obteniéndose mejoras relativas de hasta un 35% (escenario 4). En el peor caso (escenario 3), la mejora obtenida para la eficiencia es aún considerable (12%).

En resumen, los resultados que arrojan estos experimentos ilustran la capacidad del ataque DoS a baja tasa contra servidores concurrentes:

- Con este ataque se pueden alcanzar valores de eficiencia muy elevados.
- Se pueden obtener múltiples valores de eficiencia y carga de tráfico en función de la configuración escogida para el ataque. Desde este punto de vista, se puede afirmar que el ataque es muy versátil.
- Comparado con el envío aleatorio de un tráfico de ataque al servidor, la estrategia seguida para la realización del ataque mejora sustancialmente la eficiencia conseguida.

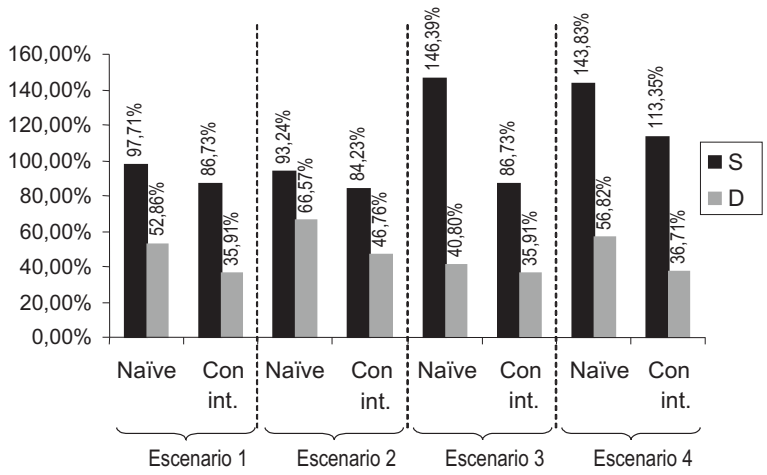


Fig. 5.16: Comparación de eficiencia (D) obtenida con un ataque naïve y el aquí propuesto, para 4 escenarios diferentes.

5.3.3 Validación en entornos reales

Se ha implementado un prototipo que ejecuta el ataque DoS a baja tasa contra servidores concurrentes en un entorno *Win32* para comprobar, por un lado, si es factible la implementación del ataque y, por otro, si los resultados obtenidos en las diferentes simulaciones sufren variaciones con respecto a los que se obtendrían en un entorno de explotación real.

El servidor elegido como víctima del ataque es un servidor Web Apache 2.0.52 alojado en una única máquina con sistema operativo Windows XP. El servidor se ha configurado con la directiva `KeepAliveTimeout=10` segundos, lo que corresponde al parámetro T_{out} en todo el desarrollo previo de este capítulo. Además, se ha configurado la directiva `ThreadsPerChild`, que representa al número de elementos de procesamiento del servidor, N_s, en un rango de valores entre 12 y 50, dependiendo del escenario evaluado.

El tráfico procedente de usuarios legítimos se ha generado sintéticamente de modo que cada mensaje o petición se ha emitido según una distribución de *Poisson* en un cierto rango de valores para el tiempo entre llegadas, T_a. Tanto en los usuarios como en el lado del atacante se ha habilitado la recolección de los datos necesarios para el cálculo de los indicadores de rendimiento del ataque.

La Tabla 5.2 muestra los resultados obtenidos para ocho configuraciones diferentes tomadas del conjunto de experimentos realizados. Para cada configuración se han presentado tanto los valores de D como de S obtenidos en el entorno real y en el de simulación.

Nótese que en los resultados se aprecia una ligera variación entre los entornos real y simulado, a veces con resultados incluso mejores en el caso del escenario real. Estas variaciones se deben, fundamentalmente, a las desviaciones en la estimación del tiempo de ida y vuelta y en el modelado de la varianza del temporizador T_{out}, y posiblemente también a las aproximaciones del modelo.

Entorno	S	D
simulación	86,73%	35,91%
real	81,74%	36,14%
simulación	84,23%	46,76%
real	74,57%	48,22%
simulación	113,35%	36,71%
real	109,00%	38,01%
simulación	268,47%	12,31%
real	269,82%	12,40%
simulación	68,04%	78,25%
real	76,00%	72,60%
simulación	249,49%	10,22%
real	256,41%	10,01%
simulación	89,88%	73,79%
real	92,41%	74,00%
simulación	191,56%	10,08%
real	183,80%	12,34%

Tabla 5.2: Comparación entre los valores D y S obtenidos de ocho configuraciones de ataque diferentes evaluadas en entornos de simulación y real.

Los resultados obtenidos en el entorno real confirman la preocupante conclusión extraída de las simulaciones, esto es, que el ataque DoS a baja tasa contra servidores concurrentes puede alcanzar una alta eficiencia. Además, la implementación del ataque no ha presentado dificultades técnicas dignas de resaltar, lo que demuestra su viabilidad y, por tanto, su riesgo.

5.4 Mejoras al ataque

En el desarrollo previo de este capítulo se ha demostrado no sólo que es posible ejecutar un ataque de baja tasa contra servidores concurrentes, sino también que éste puede alcanzar un alto rendimiento. Esto no implica que, como consecuencia de un proceso de depuración, el atacante no pueda mejorar el procedimiento de ejecución del ataque para obtener resultados mejores.

A continuación se proponen algunas mejoras que favorecen la ocultación del ataque:

- *Distribución del ataque*:

  Al igual que para el caso del servidor iterativo, el atacante puede decidir realizar el ataque de modo centralizado (DoS) o bien distribuido (DDoS), obteniendo en este último caso las ventajas que la distribución ofrece. Las desventajas al distribuir el ataque no son más que aquellas comunes a todos los ataques distribuidos, es decir, la complejidad que supone el reclutamiento de máquinas agentes y el establecimiento de mecanismos de comunicación y control entre ellas, los gestores y el atacante.

- *Utilización de la técnica de* spoofing:

También es posible mejorar la ocultación del ataque usando la técnica de *spoofing*. En el caso del servidor concurrente son de aplicación las mismas restricciones que las dadas para el caso del servidor iterativo, es decir, puesto que el atacante necesita recibir los mensajes procedentes del servidor, la dirección origen falsa se debe elegir en el rango de direcciones de la propia red donde está ubicada la máquina atacante que emite las peticiones.

- *Diversificación de los mensajes de ataque*:

  Con el objetivo de evitar sistemas de detección de intrusiones que analizan comportamientos repetitivos, el patrón para los mensajes de ataque debe ser tan variado como se pueda. Nótese que, en el caso ejemplo del servidor HTTP persistente, la petición que realiza el atacante puede ser tan variada como éste desee. Incluso se puede elegir entre peticiones de recursos no existentes en el servidor. El motivo es que, para todas ellas, el temporizador de cierre de la conexión será el mismo. De esta forma, el atacante puede utilizar este grado de libertad para evitar ser detectado por los sistemas de detección de intrusiones basados en el análisis de las peticiones realizadas al servidor.

También se proponen dos mejoras que favorecen la obtención de mayores rendimientos para el atacante:

- *Optimización del tiempo de ocupación de las posiciones en el servidor*:

  El atacante también tratará de que las peticiones que se envíen mantengan al servidor ocupado en su procesamiento el máximo tiempo posible, de modo que el esfuerzo de capturar una posición en la cola se vea rentabilizado. Así, en el caso del servidor HTTP persistente, existirá un máximo de peticiones posibles para enviar en una misma conexión[4]. El atacante puede obtener dicho valor máximo mediante alguna prueba y, posteriormente, en todas las conexiones realizadas, enviar dicho número de peticiones espaciándolas de modo que no se cierre la conexión. Nótese que esta técnica se ha utilizado previamente en algún otro tipo de ataque como *Naptha* [Corp., 2000].

- *Optimización de la estrategia de ataque*:

  El atacante puede también optimizar la propia estrategia de ataque utilizada contra el servidor para adaptarse mejor a las características del mismo y obtener mayor rendimiento. Esto dependerá, obviamente, de la estrategia utilizada en cada caso.

  Para el procedimiento de ataque presentado anteriormente caben algunas mejoras relativas a la estrategia de seguimiento de capturas. En efecto, la estrategia básica consiste en que las hebras de ataque que no poseen una posición capturada en la cola intentan conseguir una mediante el envío de mensajes de ataque con una tasa $1/\Delta_r$. Esta estrategia, aunque simple, no es la óptima, como se demostrará a continuación.

  En adelante se plantean dos posibles procedimientos que mejoran el comportamiento de la estrategia básica de seguimiento de capturas. Cada uno de ellos supone una mejora progresiva con respecto al anterior, al introducir nuevas características que permiten al atacante tener un mayor control sobre el ataque.

[4] En el servidor Apache 2.0, este máximo viene dado por la directiva *MaxKeepAliveRequests*. Cuando su valor es 0 no existe un máximo para el número de peticiones consecutivas en una única conexión.

5.4.1 Estrategia de compartición de información entre hebras

En la estrategia básica de seguimiento de capturas, una vez que se ha capturado una posición en alguna cola de servicio se estima el instante de ocurrencia de la salida correspondiente a dicha captura realizada. Entonces, el procedimiento para conseguir una nueva captura es programar un periodo básico de ataque sincronizado en torno al instante de ocurrencia estimado para la salida.

A una salida se la denominará en adelante *salida atendida* cuando una hebra programa un periodo de ataque en torno al instante de ocurrencia de dicha salida. Por el contrario, se habla de una *salida no atendida* cuando ninguna hebra programa un periodo básico de ataque en torno a su instante de ocurrencia previsto. Dado que una hebra solamente ejecuta un periodo de ataque a la vez, de todas las salidas correspondientes a estas capturas solamente una será la salida atendida, siendo el resto de ellas no atendidas. A continuación se clarifica cómo es posible que una hebra capture varias posiciones en el servidor.

En caso de que en el servidor exista una única posición libre mientras una hebra de ataque está ejecutando la fase de *ontime* de un periodo de ataque, solamente uno de los N_p paquetes enviados por el atacante podrá ocupar dicha posición. Ahora bien, debido al efecto de superposición que se produce en los sistemas multiproceso (Fig. 5.3), cuando los instantes de ocurrencia de varias salidas están próximos existirán varias hebras ejecutando sus periodos de ataque a la vez en torno a dichos instantes. Esto puede provocar que una hebra capture más de una posición en el servidor durante el periodo de actividad *ontime*, provocando que otras hebras fallen en la captura de una posición.

En la Fig. 5.17 se muestra un ejemplo en el que sucede este fenómeno. En ella están representados los instantes de ocurrencia de dos salidas, uno estimado por la hebra número 1 y la otra por la número 2. Ambas hebras tienen programados periodos básicos de ataque con $N_p = 3$. Para la hebra 1, la evolución de la transmisión de dichos paquetes en el tiempo está representada mediante flechas continuas, mientras que para la hebra 2 se utilizan flechas discontinuas. En la figura se puede apreciar que, una vez que se produce la salida correspondiente a la hebra 1, un mensaje de ataque procedente de dicha hebra captura la posición. Seguidamente, se produce la salida correspondiente a la posición previamente ocupada por la hebra 2. En este caso, el hecho de que la salida se produzca justo antes de la recepción del tercer mensaje de ataque de la hebra 1 hace que dicha hebra también capture esta posición. El resultado final de todo este proceso es que la hebra 1 ha capturado dos posiciones en el servidor. La salida estimada para una de ellas será elegida como salida atendida y la otra como no atendida. Por otro lado, la hebra 2 no ha capturado ninguna posición y, según la estrategia básica, debe entrar en la fase de recuperación de una posición en las colas consistente en enviar un mensaje de ataque cada Δ_r segundos.

En este punto, cabe plantearse que la estrategia básica no es óptima, es decir, que no tiene sentido que la hebra 2 envíe mensajes cada Δ_r ya que la hebra 1 ha capturado ya dicha posición en la cola. Solamente sería preciso enviar mensajes de ataque cada Δ_r segundos en el caso de que la captura haya sido realizada por un usuario legítimo. Se plantea entonces como solución permitir que las hebras de ataque puedan compartir información entre ellas relativa a las capturas que realizan, esquema que se denominará en adelante *estrategia de*

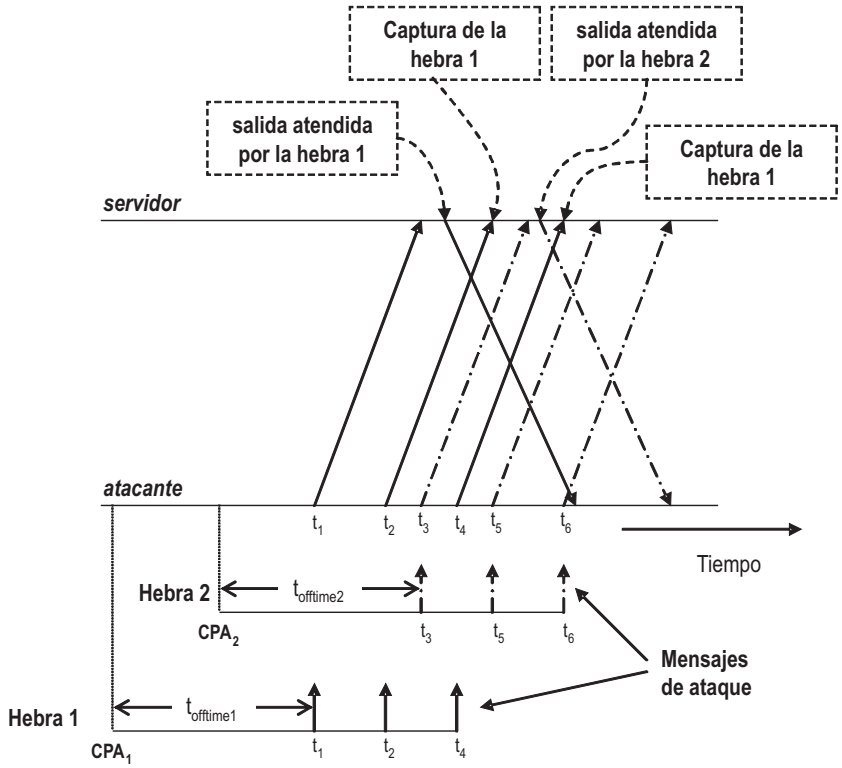


Fig. 5.17: Esquema de superposición entre dos hebras en el que una captura dos salidas y la otra ninguna.

compartición de información entre hebras, o de forma más reducida, *estrategia ITIS* (del inglés *Inter-Threads Information Sharing*).

El funcionamiento básico de la estrategia ITIS es el siguiente. Siempre que una hebra de ataque falle en la captura de una posición en el servidor tras la ejecución de un periodo de ataque, antes de entrar en la fase de recuperación de una posición como en la estrategia básica se especifica, solicitará a las otras hebras información sobre capturas realizadas y cuyas salidas no están siendo atendidas. En caso de que dichas capturas existan, la hebra tomará alguna de ellas y atenderá su salida, es decir, programará un periodo de ataque en torno al instante de ocurrencia de la salida correspondiente.

Cabe plantearse en este punto cuál es la información que una hebra de ataque debe solicitar a las otras cuando no tiene ninguna salida para atender. Dado que el ataque está diseñado de modo que cada una de las hebras pueda actuar de forma independiente y que, además, se pueda hacer en modo distribuido, no es conveniente pensar en el paso de una información que depende de la máquina desde la cual se ocupa la posición en la cola de conexiones (por ejemplo, si se considera que una captura equivale a una conexión, los datos correspondientes al *socket* que almacena la información sobre la conexión son una

información local a la máquina que corresponda). Con el fin de respetar estos principios de diseño, la información que se comparte entre las hebras consiste en los instantes de ocurrencia predichos para las salidas no atendidas.

Para poder hacer efectivo este paso de información, cada hebra deberá realizar la tarea de estimación de los instantes de ocurrencia tanto para las salidas que son atendidas como para las que no lo serán. Como ejemplo, para el caso del servidor HTTP persistente, la hebra atacante deberá enviar una petición *HTTP request* sobre todas las conexiones que consiga realizar, y esperar el instante de recepción de la respuesta *HTTP response*. Una vez realizada de este modo la estimación del instante de ocurrencia para una salida no atendida, dicha información podrá ser enviada a cualquier hebra que la solicite. De este modo, desde el momento en que se realiza la captura de la posición en la cola hasta que se produce la estimación del instante de la salida la información podría no estar disponible para otras hebras. Solamente después de la estimación del instante de la salida se podrá compartir información con otras hebras.

Cada hebra de ataque almacenará la información sobre los instantes estimados de ocurrencia de las salidas que no están siendo atendidas en la que denominaremos *cola de posiciones*. Cada vez que se produzca una captura y su correspondiente salida no vaya a ser atendida, se introducirá la información correspondiente en dicha cola. Del mismo modo, existirá un mantenimiento permanente de la cola de modo que las salidas cuyo instante de ocurrencia predicho haya caducado serán eliminadas de la cola. Cuando esto sucede, ningún periodo de ataque ha sido programado para dicha salida, por lo que la probabilidad de captura de dicha posición por parte de un usuario legítimo será mayor. Por ello, cada vez que alguna hebra solicite la información sobre una salida no atendida, se tomará la relativa a la más antigua existente en la cola de posiciones. De este modo se reduce el número de salidas que caducan y, por tanto, se mejorará la eficiencia.

También es posible que, tras fallar en la captura de una posición en el servidor, la propia hebra posea posiciones correspondientes a salidas no atendidas (*salida propia*). De este conjunto de salidas no atendidas, la hebra atenderá la que se vaya a producir antes. A este mecanismo le denominaremos *paso de información dentro de la propia hebra* y siempre será preferible al hecho de tomar una captura no atendida procedente de otra hebra, dado que así se simplifica y agiliza el proceso de compartición de información.

De este modo, el procedimiento general propuesto para el funcionamiento de una hebra de ataque se representa con el diagrama de flujo mostrado en la Fig. 5.18. Se puede observar que cada hebra de ataque, cuando no posee una salida atendida, consulta si existe alguna pendiente de ser atendida dentro de la propia hebra y, en caso negativo, realiza la misma consulta al resto de hebras. En caso de que no existan salidas no atendidas, se genera un mensaje de ataque y se espera un tiempo Δ_r. Si en este intervalo de recuperación no se ha capturado ninguna posición en el servidor, el proceso descrito anteriormente se repite. Cuando finalmente se obtiene una captura, se pasa a atender su correspondiente salida mediante la programación de un periodo de ataque.

Siempre que se está atendiendo una captura, se puede salir de esta tarea mediante la ocurrencia de dos eventos. En primer lugar, ante la captura de una nueva posición, en cuyo caso, sin interrumpir la ejecución del actual periodo básico de ataque, se procederá a estimar su instante de salida y a introducir la información sobre el instante estimado de ocurrencia

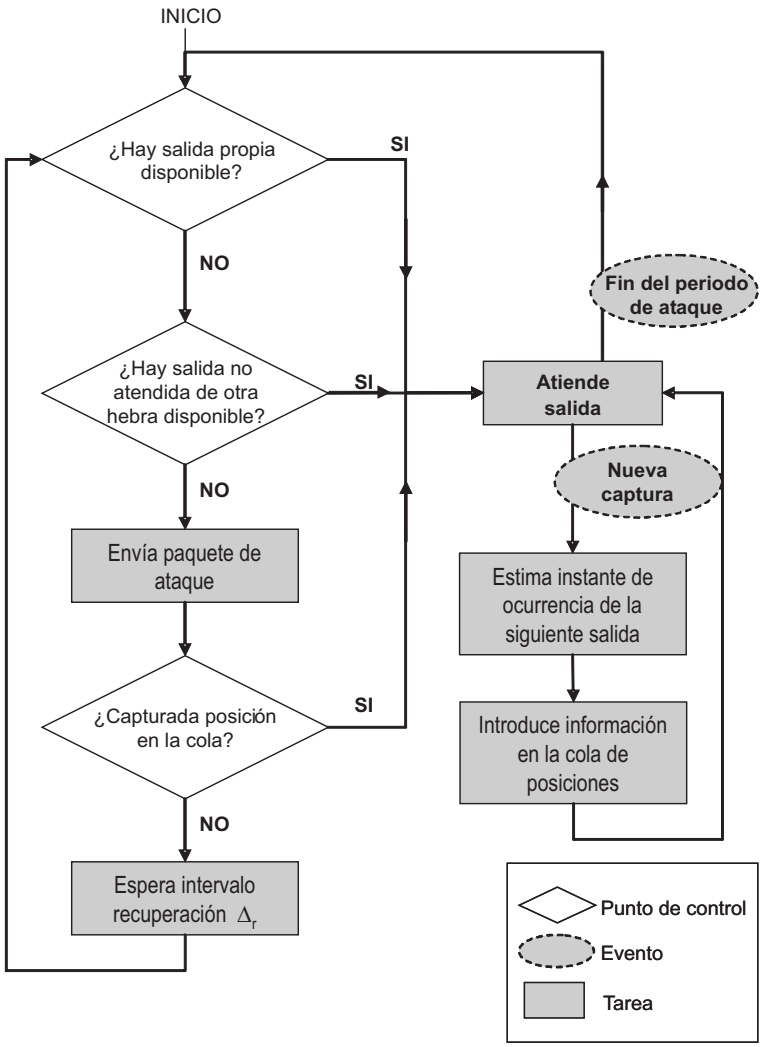


Fig. 5.18: Diagrama de flujo para el funcionamiento de la estrategia de compartición de información entre hebras (ITIS).

de la salida en la *cola de posiciones*. Nótese que, dado que no se interrumpe la ejecución del periodo básico de ataque, una vez realizadas las operaciones oportunas, el flujo vuelve a la tarea "atiende salida". El otro evento que provoca la salida del estado en que se está atendiendo a una salida es la finalización del periodo de ataque. En este caso, se realiza de nuevo el proceso de requerimiento de alguna salida en la propia hebra o bien en otras.

Comparación de las estrategias básica e ITIS

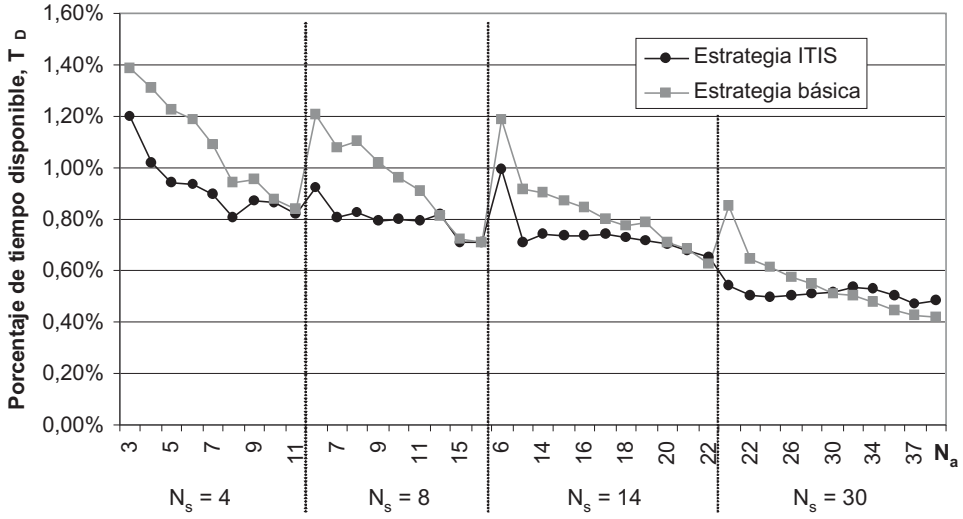
Para poder comprobar si la estrategia ITIS introduce una mejora con respecto a la estrategia básica de seguimiento de capturas, es necesario realizar un estudio experimental comparativo sobre el rendimiento de ambas estrategias. Para ello, se ha realizado la implementación en simulador de la estrategia ITIS. En esta implementación no se han tenido en cuenta detalles relativos a una distribución del ataque, de modo que éste se ejecuta desde un punto centralizado.

El método elegido para la comparación está basado en la observación de las parejas de indicadores que miden tanto la eficiencia (en términos del tiempo disponible, T_D), como la carga de mensajes de ataque necesaria por parte del atacante (sobrecarga, S). Dicha observación se hace para un número elevado de escenarios para los cuales se realiza la simulación en *Network Simulator 2* (NS2) [Fall and Varadhan, 2007]. Para cada escenario se estudiará la evolución de los indicadores de rendimiento cuando se varían, de forma independiente, los parámetros considerados en los experimentos presentados en la Sección 5.3.1, es decir, N_a, Δ_r, $var[T_{out}] + var[RTT]$, t_{ontime} y Δ.

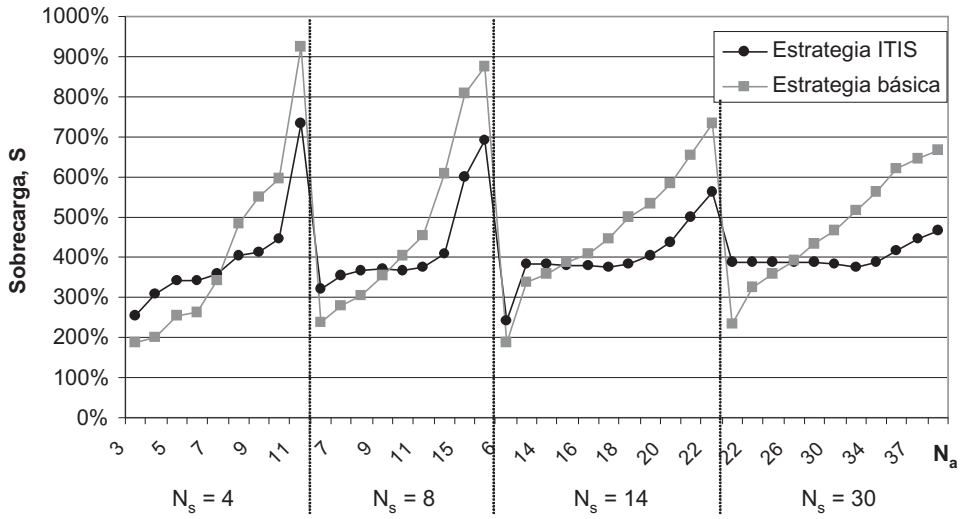
Las Fig. 5.19 a 5.23 presentan algunos resultados procedentes de los estudios realizados. En ellas se muestran los valores de T_D y S obtenidos tanto para la estrategia básica como para la estrategia ITIS en diferentes escenarios que a continuación se describirán, siendo los valores por defecto para los distintos parámetros involucrados los ya indicados en la Tabla 5.1.

En primer lugar, la Fig. 5.19 muestra el comportamiento de T_D y S con respecto al parámetro N_a. En ella se puede apreciar cómo la mayor parte de valores de N_a aportan una mejor eficiencia (menor T_D) con la estrategia ITIS que con la básica, incluso cuando en algunos escenarios la sobrecarga es menor para la estrategia ITIS. Nótese que, para escenarios en los que el valor de N_a es bajo, la sobrecarga con la estrategia ITIS es mayor que con la básica. La explicación de este fenómeno consiste en que, aunque se supone que con la primera estrategia se producirá una reducción de la carga de tráfico debido a los intervalos de recuperación, el hecho de que se traspase información entre las hebras produce también un mayor número de salidas atendidas con sus correspondientes periodos de ataque, lo que a su vez contribuye a una mayor sobrecarga. Sin embargo, cuando el número de hebras de ataque se incrementa, usando una estrategia coordinada entre ellas (ITIS) se reduce la sobrecarga obteniendo mayor eficiencia que con la estrategia básica.

En la Fig. 5.20 se muestran los resultados obtenidos en el mismo estudio, esta vez cuando se varía el parámetro intervalo de recuperación, Δ_r. En términos de eficiencia (T_D), solamente cuando Δ_r tiene un valor reducido la estrategia básica presenta mejor eficiencia, pero a costa de tener mayor sobrecarga. Sin embargo, para valores no pequeños de Δ_r

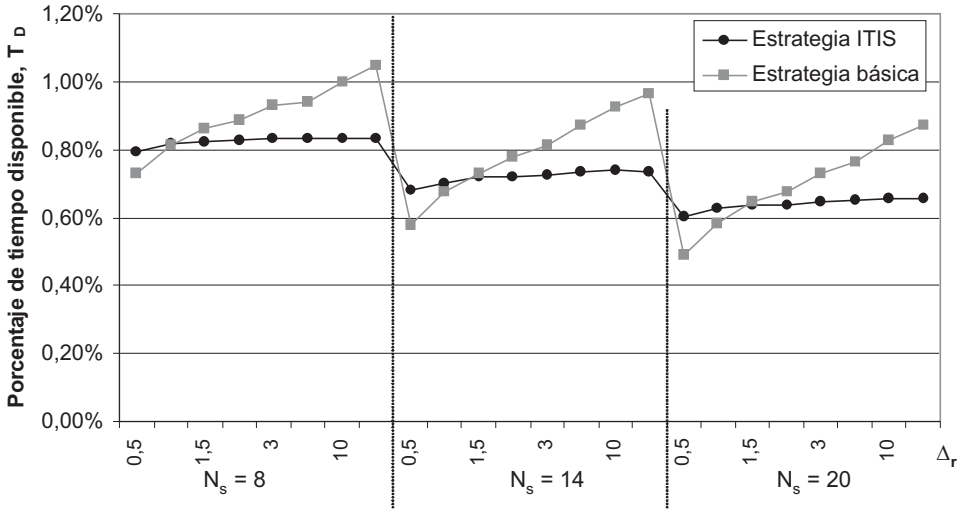


(a)

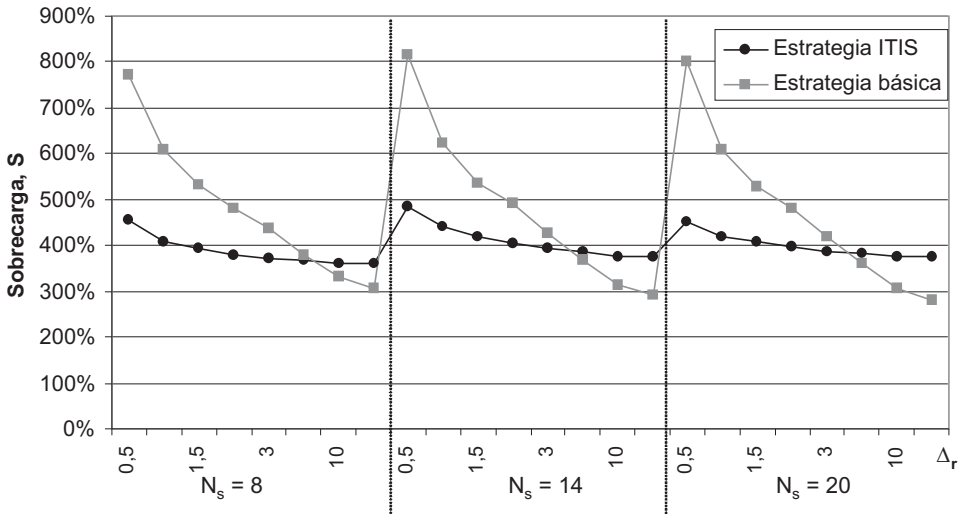


(b)

Fig. 5.19: Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía el número de hebras de ataque, N_a, para cuatro valores diferentes de N_s.

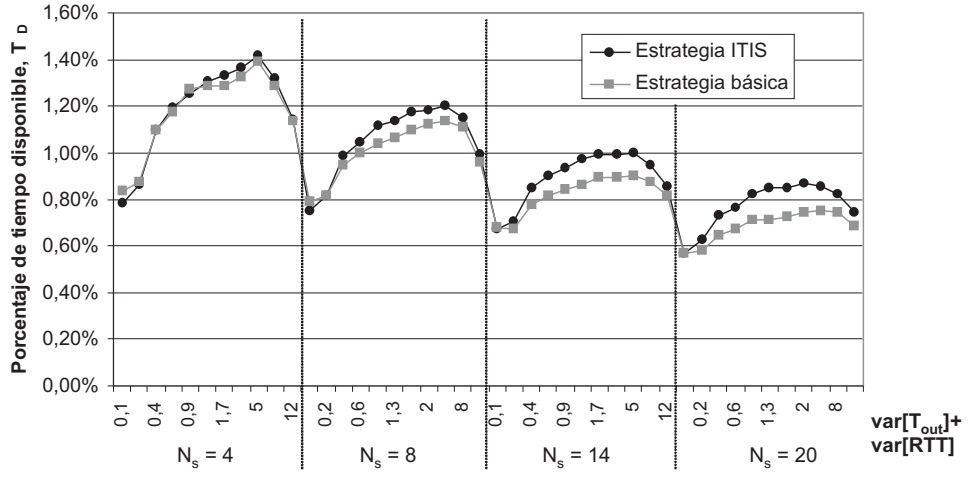


(a)

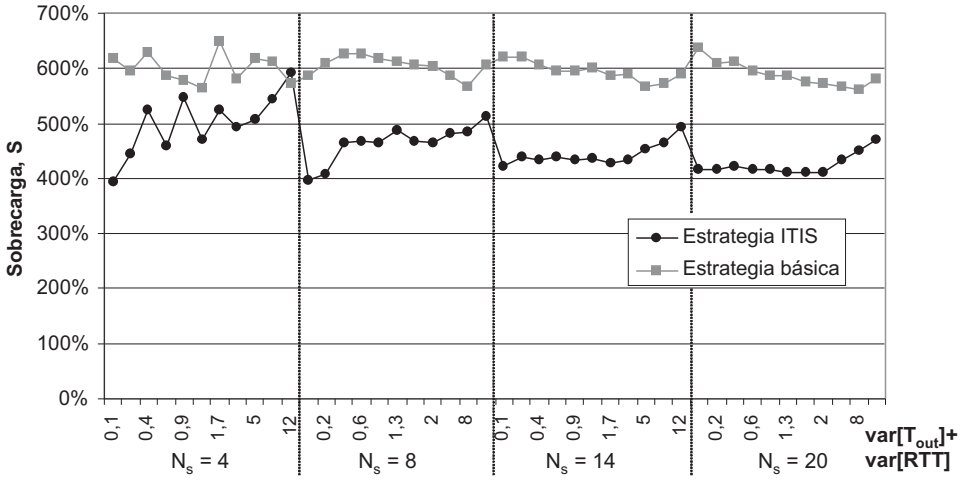


(b)

Fig. 5.20: Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía el intervalo de recuperación, Δ_r, para tres valores de N_s.

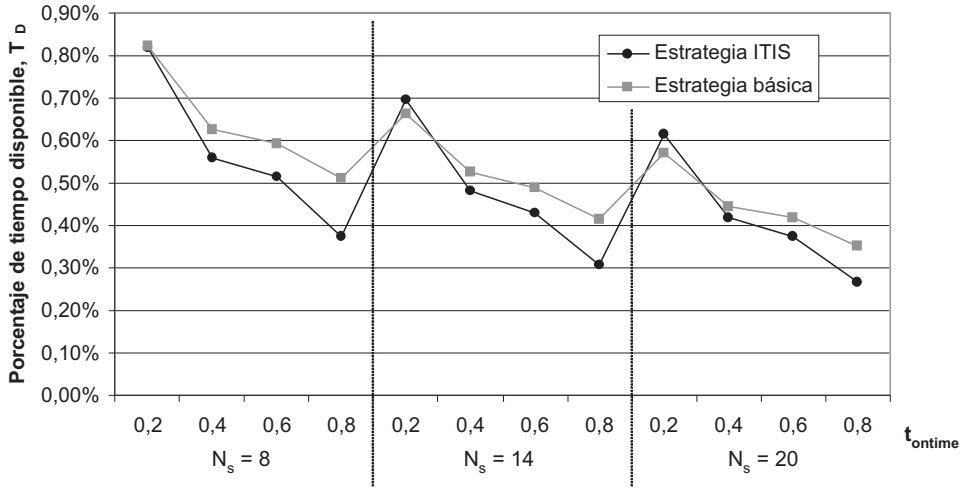


(a)

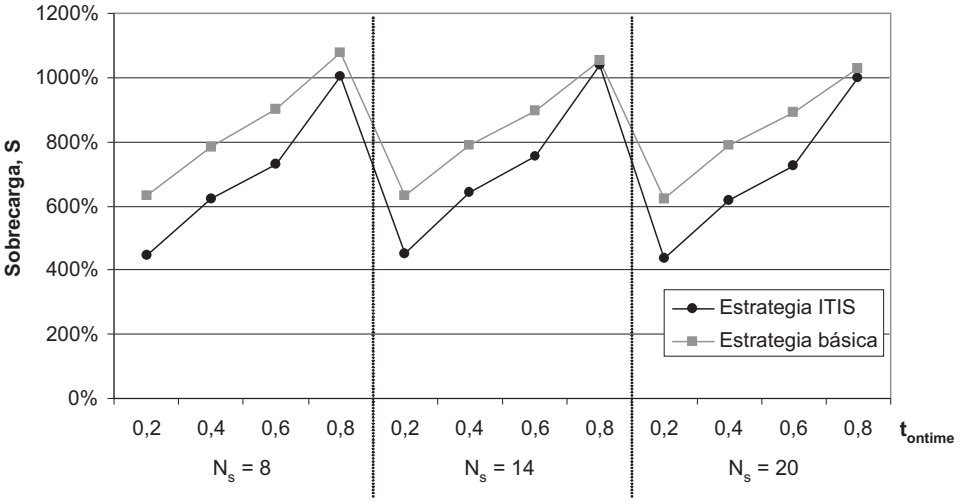


(b)

Fig. 5.21: Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía $var[T_{out}] + var[RTT]$, para cuatro valores de N_s.

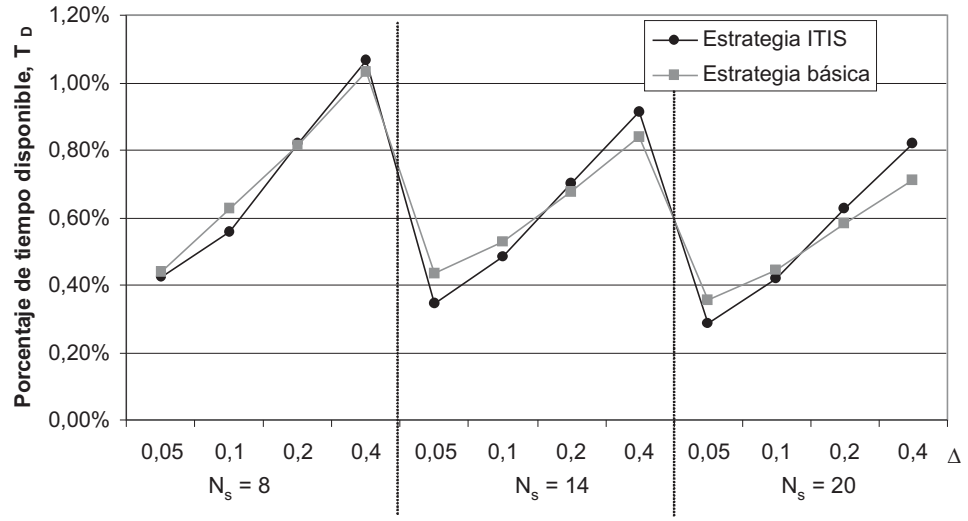


(a)

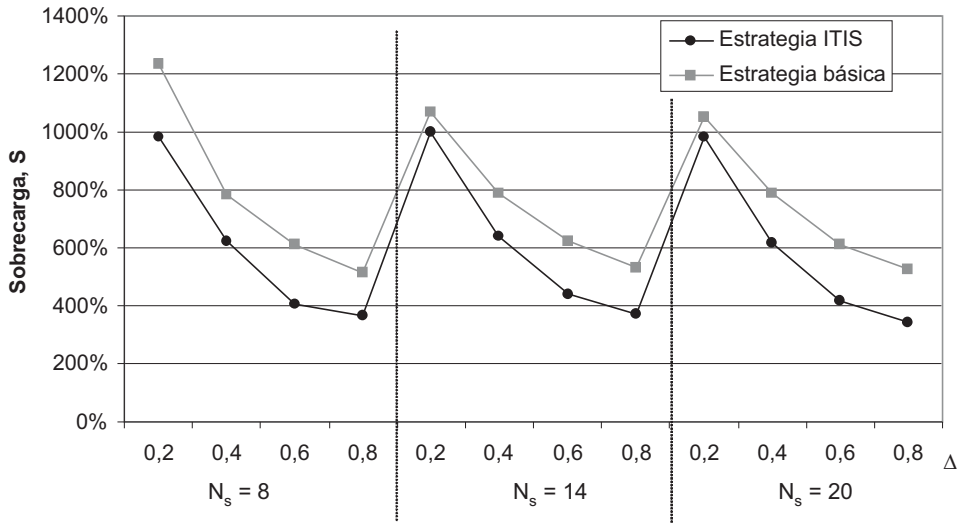


(b)

Fig. 5.22: Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía la duración de t_{ontime}, para tres valores de N_s.



(a)



(b)

Fig. 5.23: Comparación de los valores de T_D y S para las estrategias básica e ITIS cuando se varía el intervalo, Δ, para tres valores de N_s.

la estrategia ITIS es más eficiente, incluso utilizando una sobrecarga menor que con la estrategia básica.

Los resultados obtenidos cuando se varía la varianza de las desviaciones en la sincronización entre la llegada del periodo básico de ataque y el instante de ocurrencia de la salida, $var[T_{out}] + var[RTT]$, se muestran en la Fig. 5.21. En ella se aprecia cómo, aun con resultados similares para la eficiencia, la sobrecarga en la estrategia básica es mucho más elevada (el promedio del incremento absoluto es 142,83% para los escenarios considerados).

En la Fig. 5.22 se recopilan algunos resultados obtenidos con la variación del parámetro t_{ontime}. En ella se aprecia que, siendo la sobrecarga de la estrategia ITIS siempre menor, la eficiencia obtenida en la mayoría de los casos es también mejor (menor T_D). De los escenarios mostrados, solamente cuando $t_{ontime} = 0,2\ s$ se alcanza mejor eficiencia con la estrategia básica. Ahora bien, esta mejoría se consigue a costa de un incremento en sobrecarga de alrededor del 200%.

Finalmente, la Fig. 5.23 considera la variación del parámetro Δ. También en todos los casos se utiliza una menor sobrecarga en la estrategia ITIS, consiguiendo incluso en muchos de ellos una mejor eficiencia que con la básica.

Los anteriores experimentos muestran que la estrategia ITIS aporta un mejor rendimiento que la básica en cuanto a que se obtiene una mayor eficiencia para una sobrecarga dada. La conclusión obtenida de la comparación es la esperada: la estrategia ITIS mejora los resultados obtenidos con la estrategia básica considerablemente debido a que esta estrategia utiliza el conocimiento del comportamiento de unas hebras por parte de otras.

5.4.2 Estrategia del umbral de capturas

Los modelos matemáticos presentados en el Capítulo 3 permiten a un atacante seleccionar los parámetros del ataque una vez que se han estimado las características de comportamiento del servidor. Ahora bien, aunque dichos modelos proporcionan una guía útil para el ajuste del ataque, se fundamentan en la consideración de que todas las hebras están atendiendo salidas en todo instante. Esta premisa no es completamente cierta, dado que una hebra no podrá obtener una posición en la cola de servicio si dicha posición se ha ocupado previamente por parte de un usuario o por alguna otra hebra de ataque, con lo que, al no haber realizado captura alguna, puede no tener ninguna salida que atender.

En definitiva, esta aproximación realizada en los modelos matemáticos no considera el problema del seguimiento de capturas. En este sentido, la estrategia de compartición de información entre hebras, aunque mejora el porcentaje de tiempo en que las hebras están atendiendo a las salidas frente a la estrategia básica, aún no es óptima. Su principal deficiencia reside en que no se especifica el modo de calcular el número de hebras que el atacante debe establecer como activas a la hora de lanzar el ataque y, por tanto, tampoco define de forma clara cómo controlar la carga de tráfico que el atacante manda al servidor.

Dado que las especificaciones de las estrategias de seguimiento de capturas básica e ITIS concretan que cada hebra tratará de mantener la captura de una posición el mayor tiempo posible, es razonable pensar que el diseño óptimo se alcanzará cuando el número de

hebras de ataque se acerque al número de posiciones que existan en las colas del servidor. En caso de que el número de hebras sea menor, la ocupación de la cola de servicio no se podrá conseguir de manera completa. Por el contrario, si el número de hebras excede el tamaño de la cola, se generará una sobrecarga grande, debido principalmente a que dichas hebras pretenderán recuperar posiciones que realmente no existen mediante el envío de mensajes de ataque cada Δ_r segundos. Es evidente que la tasa de mensajes se elevará sin producir beneficio alguno para el atacante y provocando, sin embargo, que el ataque sea más susceptible de ser detectado.

Por tanto, el atacante que utilice tanto la estrategia básica como la ITIS se encontrará con el problema de estimar el número de posiciones de la cola de servicio y, por consiguiente, también el de determinar el número de hebras a activar para el ataque.

Para resolver este problema se plantea como propuesta la *estrategia del umbral de capturas* que a continuación se presenta. Mediante la aplicación de esta estrategia, el atacante será capaz de determinar de un modo heurístico la forma de llevar a cabo el ataque de baja tasa contra un servidor concurrente, eligiendo el volumen de tráfico a enviar de modo que se puedan eludir los posibles mecanismos de detección y respuesta que tratan de proteger al servidor.

La estrategia del umbral de capturas consiste en una mejora de la estrategia ITIS basada en una idea sencilla. El atacante establece, antes de lanzar el ataque, un umbral de capturas a conseguir, P_0, el cual se podrá modificar también durante la ejecución del ataque. Durante dicho tiempo monitoriza de manera continua el número de posiciones totales capturadas por todas las hebras de ataque en la cola de servicio del servidor, P. Mientras se cumple la condición $P < P_0$ el comportamiento de las hebras de ataque será idéntico al de la estrategia ITIS. Sin embargo, cuando la condición $P \geq P_0$ se alcanza, el comportamiento de las hebras se modificará de modo que ni se envían mensajes de ataque tras un intervalo de recuperación ni se extraen salidas, ni propias ni de la cola de posiciones, es decir, las hebras que no poseen ninguna salida atendida quedan inactivas (sin ejecutar periodos de ataque, en espera de que se cumpla $P < P_0$). De esta forma, el diagrama de flujo propuesto para la estrategia ITIS queda modificado cuando se cumple la condición $P \geq P_0$ tal y como se muestra en la Fig. 5.24.

Comportamiento de la estrategia del umbral de capturas

Cabe preguntarse, en primer lugar, si el ataque, siguiendo la estrategia del umbral de capturas, será capaz de conseguir el umbral de capturas P_0 y de mantenerlo a lo largo del tiempo. Para ello se han realizado una serie de experimentos consistentes en la ejecución del ataque contra distintos servidores con diversas configuraciones. En todas ellas, el servidor está recibiendo tráfico de usuario a la vez que de ataque. Para comprobar la eficiencia del ataque se ha elegido un escenario poco favorable al atacante. Así, la tasa de tráfico de usuario ha sido configurada con un valor elevado. De este modo, el usuario debería ser capaz de capturar más posiciones en las colas de servicio del servidor, dificultando en consecuencia al atacante la captura de las posiciones.

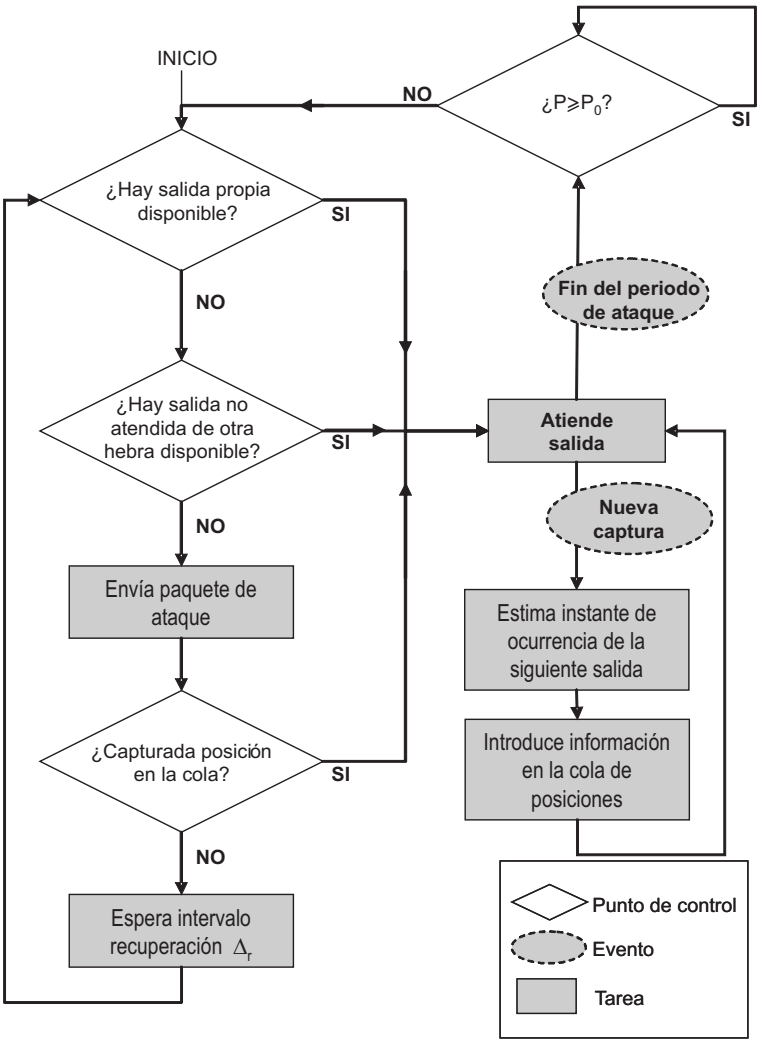


Fig. 5.24: Diagrama de flujo para el funcionamiento de la estrategia del umbral de capturas.

En estos experimentos se han constatado una serie de comportamientos que permiten establecer algunos criterios de diseño para el ataque. Dichos comportamientos son los siguientes:

- *Cuando se varía el umbral, el ataque es capaz de conseguir un número de posiciones que oscila en torno a dicho umbral*:

  La Fig. 5.25 muestra la evolución en el tiempo del número de posiciones capturadas en el servidor cuando se establecen tres umbrales diferentes. El servidor, en el escenario para el que se muestran los resultados, dispone de un total de 40 posiciones en las colas de servicio. Se observa claramente que, tras un transitorio corto, el valor de P oscila en torno al umbral establecido.

  Los valores $P > P_0$ se obtienen debido a que la entrada en actividad de todas las hebras cuando P se sitúa por debajo del umbral produce un excedente de capturas. Los valores $P < P_0$ se producen debido a la influencia del tráfico de usuario, que pugna por conseguir también posiciones en el servidor.

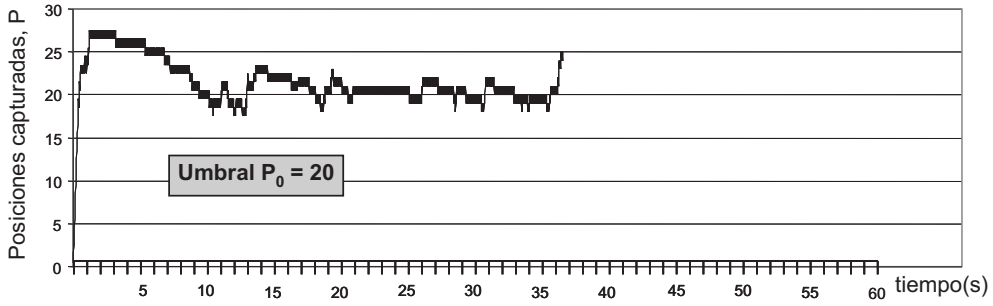
  Evidentemente, para que se produzca la consecución del número de capturas determinado por el umbral se debe cumplir que el número de hebras de ataque sea suficiente para capturar dichas posiciones. Concretamente, influirá no solamente el número de hebras sino también la capacidad de realizar capturas de cada hebra, es decir, el número de paquetes, N_p, enviados en cada fase *ontime* del periodo básico de ataque.

- *El número de hebras de ataque influye en la sobrecarga generada por éste*:

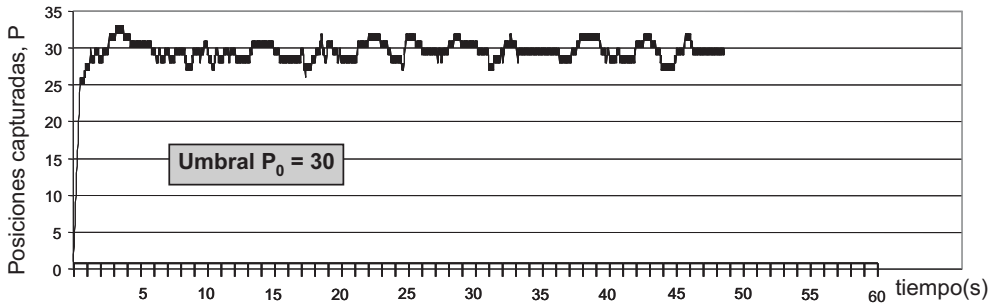
  Dado que la estrategia se ha diseñado de modo que, cuando se supera el umbral de capturas establecido, las hebras que no posean ninguna captura atendida pasan a un estado inactivo, el atacante podría pensar en elegir un número muy elevado de hebras de ataque con el fin de asegurarse que éste sea superior al de posiciones en el servidor (desconocido para el atacante).

  Sin embargo, se constata que el número de hebras sí influye en la sobrecarga del ataque, debido al efecto que éstas tienen en los instantes en que el número de posiciones capturadas, P, pasa de cumplir la condición $P \geq P_0$ a situarse por debajo del umbral, P_0. En estos momentos, todas las hebras que se encuentran inactivas se activan automáticamente y tratan de capturar posiciones, de modo que muchas de ellas enviarán mensajes de ataque cuando expire su temporizador Δ_r. Esto se realizará solamente una vez, dado que el número de posiciones capturadas subirá rápidamente por encima del umbral de nuevo. Sin embargo, en este transitorio se ha generado una sobrecarga que será tanto mayor cuantas más hebras de ataque inactivas se hayan activado.

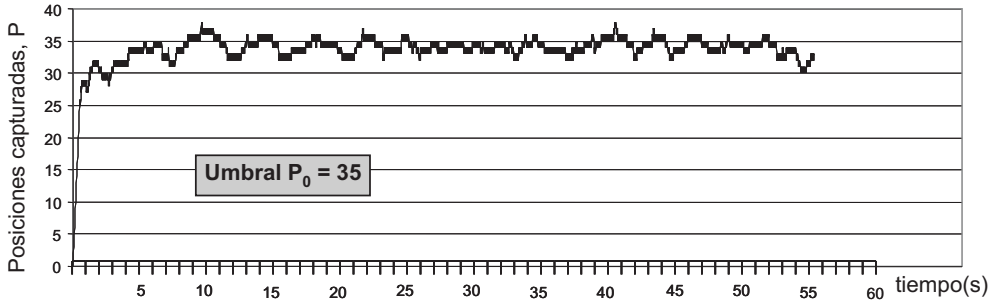
  Para evitar que se generen estas ráfagas de mensajes de ataque y conseguir que el número de hebras a las que les expira el temporizador Δ_r no sea elevado, se debe ajustar este tiempo de modo que el cociente entre el intervalo de recuperación y el número de hebras de ataque sea lo mayor posible. De este modo, el criterio de diseño para el número de hebras de ataque será elegir un número de ellas ligeramente mayor al umbral escogido, de modo que se garantice la consecución de éste. A su vez, este número determinará el valor del intervalo de recuperación, Δ_r. Posteriormente se propondrá un procedimiento para el ajuste de estos valores en base a estas conclusiones.



(a)



(b)



(c)

Fig. 5.25: Nivel de consecución del umbral para varios valores de P_0 en un servidor con un total de 40 posiciones: a) $P_0 = 20$, b) $P_0 = 30$ y c) $P_0 = 35$.

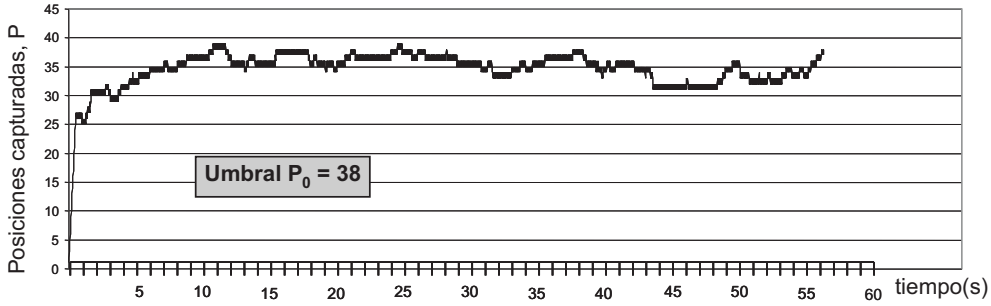


Fig. 5.26: Nivel de consecución del umbral para $P_0 = 38$ en un servidor con un total de 40 posiciones.

- *Existe un valor de umbral a partir del cual el ataque no es capaz de conseguir las posiciones requeridas*:

  En los diferentes experimentos se ha constatado que, a partir de un valor del umbral, que denominaremos *valor crítico de ocupación*, el atacante no es capaz de conseguir el número de posiciones especificado por dicho umbral sin elevar en exceso la carga de tráfico del ataque y, en ocasiones, ni tan siquiera utilizando una carga elevada de tráfico. Es fácil pensar que si el umbral P_0 supera al número total de posiciones del servidor, será imposible conseguir dichas posiciones, dado que no existen. Sin embargo, incluso si se considera esta cota para el umbral, la presencia de tráfico de usuario hace que, dependiendo de la tasa que dicho tráfico tenga, el valor crítico de ocupación pueda situarse por debajo del número total de posiciones del servidor.

  En el escenario de ejemplo cuyos resultados se han mostrado en la Fig. 5.25 –en este escenario el número de posiciones del servidor es 40– se aprecia claramente que, aunque para valores del umbral $P_0 \leq 35$ se consigue fácilmente dicho umbral, cuando éste se sitúa en el valor 38 (ver Fig. 5.26), aun siendo menor que el número total de posiciones en el servidor, 40, el ataque no es capaz de conseguirlo.

  En resumen, el valor crítico de ocupación se situará siempre por debajo del número total de posiciones en el servidor y, además, dependerá del trafico de usuario, de tal modo que cuanto mayor sea éste, menor será el valor crítico de ocupación.

  Nótese que, mediante la observación del nivel de consecución del umbral, el atacante será capaz de decidir si ha llegado al valor crítico de ocupación o no. Esta técnica permite al atacante, cuando el tráfico de los usuarios no es elevado, estimar una cota próxima al número de posiciones existente en las colas de servicio del sistema.

- *El establecimiento del umbral permite al atacante ajustar el punto de operación del ataque*:

  Cuando se define un umbral para el ataque, se está definiendo implícitamente un rendimiento determinado. La elección de umbrales altos implicará valores altos de sobrecarga, S, lo que redundará también a su vez en valores altos de eficiencia del ataque (disponibilidad, D, con valor bajo).

En la Fig. 5.27 se puede observar el rendimiento obtenido, en términos de la sobrecarga, S, y de disponibilidad, D, en el ataque a un servidor con un total de 40 posiciones. La figura muestra diferentes resultados según los valores del umbral elegidos para tres tráficos de usuario diferentes, cada uno de ellos caracterizado por su tiempo entre llegadas de mensajes, T_a. El valor máximo para el umbral es el número de posiciones del servidor, $P_0 = 40$, en el que se alcanza la mayor eficiencia para el ataque, aunque la carga de tráfico también será la mayor.

El atacante, por tanto, puede hacer uso del umbral para ir aumentando gradualmente el nivel de denegación de servicio obtenido por el ataque, de modo que éste comienza siendo un ligero incremento de ocupación y de tráfico en el servidor y acaba generando la denegación de servicio transcurrido el tiempo que el atacante desee. El incremento gradual del umbral, por tanto, permitirá eludir mecanismos de detección de intrusiones.

- *Cuando se establece un umbral igual o mayor al valor crítico de ocupación, la sobrecarga del ataque crece considerablemente*:

  Aunque ya se ha visto que la sobrecarga crece conforme el umbral aumenta, en los experimentos realizados se constata también que, al alcanzar el umbral el valor crítico de ocupación, la sobrecarga crece a mayor ritmo que con umbrales inferiores. Esto se debe a que, a partir de este valor, la carga de tráfico del ataque para conseguir posiciones es mucho mayor, lo que se traduce en una sobrecarga superior.

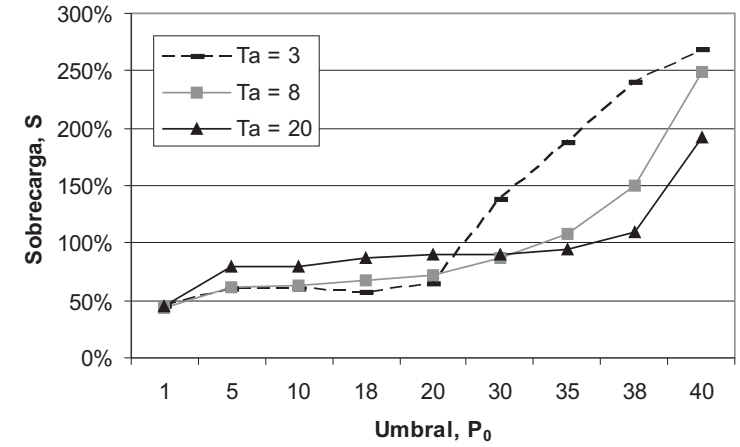
  En la Fig. 5.27(a) se puede observar cómo, para los diferentes tráficos de usuario, la sobrecarga comienza a crecer más rápidamente a partir de un valor del umbral. Dado que el valor crítico de ocupación depende del tráfico de usuario, el aumento se produce en diferentes valores de umbral para cada uno de los tres tráficos representados en la figura, siendo menor el valor del umbral cuanto mayor sea el tráfico de usuario (menor valor del tiempo entre llegadas, T_a).

  La principal conclusión a la que esta constatación lleva es que el atacante debe saber que, una vez alcanzado el valor crítico de ocupación, un incremento en la eficiencia del ataque lleva aparejado un aumento grande en la sobrecarga. En el caso de que el atacante quiera controlar la tasa de tráfico enviada al servidor, no deberá sobrepasar el valor crítico de ocupación.

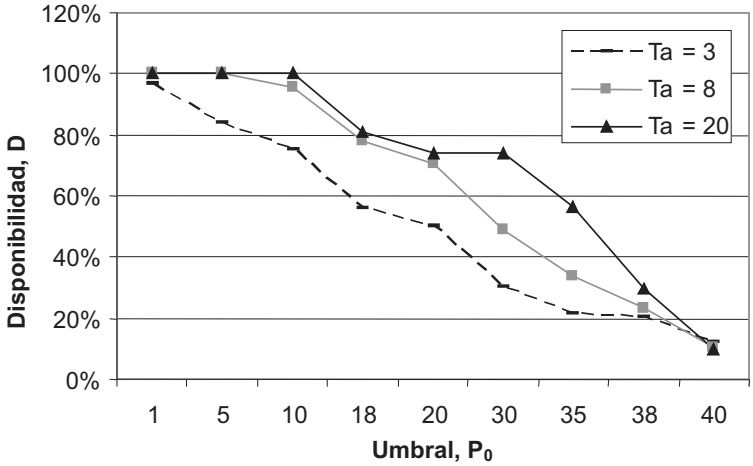
Procedimiento para la ejecución del ataque con estrategia del umbral de capturas

Recopilando las diferentes conclusiones obtenidas sobre el comportamiento del ataque cuando se aplica la estrategia del umbral de capturas, se puede elaborar un procedimiento de ataque que, de forma sencilla, permite al atacante establecer los pasos y parámetros fundamentales de cara a su ejecución.

Una vez que se ha realizado la fase preliminar de estudio de las vulnerabilidades del servidor y que se han elegido los valores adecuados para los parámetros del periodo básico de ataque, se deben llevar a cabo los siguientes pasos:



(a)



(b)

Fig. 5.27: Rendimiento del ataque contra un servidor de 40 posiciones con la estrategia de umbral de capturas para diferentes valores del umbral, P_0, y de tiempo entre llegadas de usuario, T_a: a) sobrecarga, S, y b) disponibilidad, D.

Paso 1:

- Inicializar los parámetros de configuración del ataque con los siguientes valores:

$$\begin{aligned} P_0 &= 1 \\ N_a &= 1 \\ \Delta_r &= N_a \end{aligned}$$

Paso 2:

- Monitorizar el valor del número de posiciones ocupadas, P, durante un tiempo de observación.
  - Si el valor del número de posiciones ocupadas, P, alcanza el umbral P_0 y se mantiene oscilando en torno a dicho umbral, esperar un tiempo de adaptación, T_{adapt}, cuyo valor estará predefinido, e ir al paso 3.
    Este tiempo se introduce siempre de forma previa a realizar un incremento del umbral, P_0. El objetivo que persigue es incrementar gradualmente el tráfico enviado hacia el servidor con el fin de eludir los posibles mecanismos de seguridad. El valor para este temporizador dependerá de los requisitos para que un tráfico se considere de ataque en el sistema de seguridad a eludir.
  - En caso contrario, si el número de posiciones ocupadas, P, no alcanza el umbral P_0, se ha llegado al valor crítico de ocupación. Ir al paso 4.

Paso 3:

- Actualizar los parámetros del ataque de la siguiente forma:

$$\begin{aligned} P_0 &= P_0 + 1 \\ N_a &= N_a + 1 \\ \Delta_r &= N_a \end{aligned}$$

- Ir al paso 2.

Paso 4:

- Actualizar los parámetros del ataque de la siguiente forma:

$$\begin{aligned} P_0 &= P_0 - 1 \\ N_a &= N_a - 1 \\ \Delta_r &= N_a \end{aligned}$$

- Ir al paso 2.

Por otro lado, el funcionamiento de algunos servidores permitirá que el atacante reciba un mensaje indicando que no existe ninguna posición libre en el servidor (mensaje MO) como respuesta a un mensaje de ataque previamente emitido. En el caso del servidor HTTP persistente, el atacante recibiría un mensaje TCP con el *flag* RST activado como respuesta a un mensaje de establecimiento de conexión. Nótese que este hecho permite que el atacante, conforme va aumentando el valor del umbral, puede determinar el instante en el que se está comenzando a producir una denegación de servicio en el servidor. En efecto, siempre que el servidor tenga un comportamiento normal, es decir, cuando acepte todas las peticiones hasta no tener más posiciones libres, si el atacante comienza a recibir mensajes MO, éstos pueden indicar que el servidor comienza a sufrir la denegación de servicio.

5.5 Conclusiones del capítulo

El presente capítulo ha discutido la aplicación de la estrategia general para la realización del ataque DoS a baja tasa contra servidores concurrentes. A través de los diferentes desarrollos y experimentos presentados se extraen y resumen las siguientes conclusiones:

- Siempre que se pueda encontrar una vulnerabilidad en un servidor concurrente consistente en la existencia de un patrón temporal fijo en su comportamiento que permita predecir los instantes en que se producen las salidas, se puede ejecutar un ataque DoS a baja tasa contra él.
- Se han propuesto algunos ejemplos de sistemas servidores concurrentes a los que es factible atacar mediante el procedimiento aquí expuesto. Entre ellos, por su importancia actual, destaca el servidor HTTP.
- La eficiencia obtenida por los ataques DoS a baja tasa contra servidores concurrentes es muy elevada.
- Este tipo de ataques es muy versátil. Permite al atacante elegir entre un número elevado de configuraciones dependiendo de las restricciones que existan entre el tráfico a enviar al servidor y la eficiencia a obtener en la denegación de servicio.
- El atacante dispone de herramientas que le permiten determinar los valores de los parámetros del ataque. Dichos mecanismos, como el modelo matemático presentado en el Capítulo 3, se ajustan al comportamiento real del ataque.
- El atacante dispone de mecanismos de control en la fase de ejecución del ataque, como la estrategia del umbral de capturas, que le permiten ir ejecutando de forma gradual el ataque con la finalidad de eludir la potencial existencia de sistemas de seguridad en el entorno de la víctima.
- Se ha realizado la implementación de este ataque en un entorno real y se ha concluido que dicha implementación no implica dificultades reseñables.

Capítulo 6

Conclusiones

En este capítulo se presentan las principales conclusiones y contribuciones que se han realizado a lo largo de este trabajo. Aunque éstas se han ido indicando al final de cada capítulo de esta memoria, se presentan a continuación de forma unificada:

- Se ha propuesto un modelo general y flexible para un servidor genérico, adaptable a las diferentes arquitecturas y características susceptibles de ser implementadas.
- Se ha realizado un análisis de las características generales de los servidores basándose en el modelo propuesto para éstos. En particular, se ha estudiado en profundidad el tiempo entre salidas.
- Se ha presentado la metodología general a seguir para la ejecución del ataque DoS a baja tasa contra servidores, ilustrando las implicaciones que ello conlleva, tanto en el lado del servidor atacado como en el del atacante.
- Con la aplicación de dicha metodología, se ha probado la alta capacidad del ataque DoS a baja tasa contra servidores tanto iterativos como concurrentes. La eficiencia obtenida por estos ataques es muy elevada, como se ha demostrado a través de la experimentación aportada en el trabajo. Además, comparado con un ataque *naïve* que no utilice las técnicas aquí propuestas, el rendimiento conseguido es considerablemente superior.
- Se ha particularizado la estrategia general de ejecución del ataque para dos casos diferentes: servidores iterativos y servidores concurrentes. Para cada uno de ellos se ha profundizado en las características diferenciadoras y se han propuesto las técnicas para adaptar el ataque a las mismas.
- Para los servidores de tipo iterativo se ha demostrado que el conocimiento del tiempo entre salidas del servidor constituye una vulnerabilidad que permite la ejecución del ataque DoS a baja tasa contra ellos. Esto hace que todos los servidores iterativos que permitan a un atacante predecir dicho tiempo pueden ser atacados de esta forma.

- En el caso de los servidores concurrentes, se ha mostrado cómo el conocimiento que es posible obtener del tiempo entre salidas no supone una vulnerabilidad que el atacante pueda aprovechar para ejecutar el ataque DoS a baja tasa. Sin embargo, se ha ilustrado cómo podrán ser atacados siempre que muestren patrones temporales deterministas de comportamiento que un atacante pueda predecir. De este modo, aunque no todos los servidores concurrentes son atacables mediante las técnicas expuestas en este trabajo, sí son numerosos los casos en los que el determinismo en el comportamiento del servidor puede suponer una vulnerabilidad aprovechable por un atacante para ejecutar un ataque DoS de baja tasa. En esta línea, se ha realizado un estudio detallado sobre el método a utilizar para llevar a cabo el ataque contra los servidores HTTP.

- La conclusión anterior permite recomendar, en la tarea de diseño del comportamiento de servidores, la eliminación de patrones temporales deterministas como medio de prevención de este tipo de ataques.

- Se ha mostrado que los ataques DoS a baja tasa son muy versátiles. Permiten al atacante elegir entre un número elevado de configuraciones, dependiendo de las restricciones que existan entre el tráfico a enviar al servidor y la eficiencia a obtener en la denegación de servicio. Además, el atacante puede utilizar ciertos mecanismos de control en la fase de ejecución del ataque, como la estrategia del umbral de capturas, que le permiten ir ejecutando el ataque de forma gradual, con la finalidad de eludir posibles sistemas de seguridad existentes en el entorno de la víctima.

- La ejecución de este tipo de ataques conlleva dos grandes ventajas para un atacante: en primer lugar, sería posible sortear los mecanismos de defensa basados en la detección de tasas altas de tráfico y, en segundo lugar, la cantidad de recursos necesarios para llevar a cabo el ataque se reduce drásticamente con respecto a los utilizados para llevar a cabo un ataque DoS de alta tasa.

- Se ha realizado una evaluación del rendimiento de los ataques DoS a baja tasa utilizando varias metodologías. Por una parte, se ha desarrollado un modelo matemático que relaciona el comportamiento de los indicadores de rendimiento con la variación de los parámetros de diseño del ataque. Adicionalmente, se ha realizado una amplia experimentación en entornos de simulación para comprobar la coherencia de los comportamientos y efectos que se deducen de la aplicación del ataque. Por último, también se ha llevado a cabo la implementación del ataque en un sistema real, con el fin de verificar su aplicabilidad e implementabilidad en entornos de producción.

- Se han explorado las diferentes posibilidades que el atacante podría utilizar en la ejecución del ataque. Para ello, una vez desarrollada la estrategia básica o general que posibilita dicha ejecución, se han propuesto mejoras a la misma y se ha comprobado el beneficio de sus efectos para el atacante.

Comentarios adicionales y líneas de trabajo futuro

Este trabajo viene a proponer que, para el desarrollo de estrategias de defensa contra los ataques a la seguridad, es fundamental realizar, como paso previo, un posicionamiento desde el punto de vista del propio atacante como agente en cuyo comportamiento no se puede confiar. Este tipo de estudios aporta una visión fundamental sobre las técnicas, métodos y posibilidades disponibles para la realización del ataque. Pero el aspecto quizá más importante es el conocimiento de las debilidades o limitaciones a las que se enfrenta un potencial atacante, ya que éstas serán las que constituyan la base para construir sistemas de defensa adecuados.

El trabajo realizado en esta tesis abre un campo de investigación en torno al estudio de los ataques de denegación de servicio de baja tasa. Si bien hasta ahora solamente se ha realizado este trabajo de investigación para el protocolo TCP [Kuzmanovic and Knightly, 2003] y para los servidores de aplicaciones (en este trabajo), será necesario explorar las capacidades que las técnicas aquí presentadas poseen para atacar otros tipos de sistemas y protocolos.

Por otro lado, una de las grandes contribuciones de la tesis es denunciar el peligro que el determinismo en el comportamiento de un sistema conlleva desde el punto de vista de la seguridad. En este campo, es necesario investigar los riesgos que los comportamientos deterministas puedan suponer en el contexto general de las redes de comunicación.

La presentación de este ataque DoS a baja tasa contra servidores, junto con la aparición previa de algún ataque DoS a baja tasa lleva a pensar que, más allá de ser simples ejemplos de modos de denegación de servicio, este tipo de ataques constituye una nueva familia que va adquiriendo cada vez mayor importancia.

El hecho de considerar que este tipo de ataques forma una familia o tipología hace deseable el desarrollo de medidas de defensa globales para todos ellos, aprovechando las características comunes que presentan.

Por ello, tras el trabajo de investigación de esta tesis doctoral, el siguiente objetivo debe ser la elaboración de métodos de defensa adecuados ante este tipo de ataques. El detallado conocimiento que sobre ellos aporta este trabajo supondrá una buena base para la elaboración de estas metodologías.

Conclusions

The main conclusions and contributions of this work are the following:

- A general and flexible model for a server has been proposed. It is adaptable to the different possible features and architectures that could be implemented.
- A study on the general characteristics of the servers has been made. It is based on the proposed model for them. Specifically, the inter-output time feature has been analized in depth.
- A general methodology for striking the low rate DoS attack against servers has also been presented. Both the server and the attacker side implications of the attack have been illustrated.
- Applying the mentioned methodology, the feasibility of the low rate DoS attack when launched against iterative or concurrent servers has been demonstrated. Moreover, from the different experiments shown in this work, the high levels of efficiency obtained from the execution of these attacks have been verified.
- The general strategy for the execution of the attack has been detailed for the two main classes of servers: iterative and concurrent ones. For each of them, the most differential features have been revealed and the general attack techniques have been adapted to these particular characteristics.
- For the iterative servers case, it has been demonstrated that the knowledge of the inter-output time represents a vulnerability exploitable by using a low rate DoS attack. This means that every server that allows an attacker to predict this time could be attacked in this way.
- For the concurrent servers case, it has been shown how, as a difference with the iterative case, the knowledge of the inter-output time is not enough to execute the low rate DoS attack. However, it has been illustrated how these servers could be attacked whenever they exhibit a temporal deterministic behavioural pattern that could be somehow predicted. This way, although not all the concurrent servers can be defeated by the attack techniques here presented, there exist a lot of cases in which the behavioural determinism of a server is exploitable by using a low rate DoS attack. In this direction, a detailed study on the method used for striking the attack against HTTP servers has been contributed.
- The former conclusion allows to strongly recommend, as a prevention measure against these attacks, to avoid the use of temporal deterministic patterns when designing server behaviours.
- It has been shown that the low rate DoS attacks are quite versatile. They allow an attacker to choose among a high possible number of different configurations, depending on the existing restrictions about the DoS efficiency needed and the maximum traffic rate that can be sent to the server. Besides, the attacker could use certain

control mechanisms during the execution phase of the attack, thus enabling its progressive execution aimed at bypassing possible security systems located in the victim surroundings.

- The execution of these attacks implies two main advantages for the attacker: firstly, it could be possible to bypass defense mechanisms that rely on the detection of high rate traffics and, secondly, the amount of resources needed for carrying out the attack is drastically reduced when compared with those needed for a high rate DoS attack.
- An evaluation of the low rate DoS attacks performance has been made by using several different methodologies. First, a mathematical model that relates the behaviour of the performance indicators to the parameters of the attack is contributed. Second, a wide variety of experiments for testing the behaviour and consequences of the attack has been carried out by using simulations. Finally, also a proof of concept has been implemented to check the attack in real environments. This has allowed to demonstrate the applicability and feasibility of these attacks in production environments.
- Several different strategies for the execution of the attack have been explored. For this, starting from the general strategy, some improvements have been proposed and their benefits for the attacker evaluated.

Apéndice A

Thesis Summary

Abstract

This appendix is a summary of the results presented in the PhD. Thesis written with the original Spanish title: "Ataques de denegación de servicio a baja tasa contra servidores".

The low-rate DoS attacks are presented as a novel approach for carrying out a denial of service (DoS) attack against servers. Although this kind of attacks are carried out by sending attack packets to the victim, it gets advantage of some vulnerabilities in the servers in order to reduce the amount of traffic needed for achieving its purposes. The final result is the use of a low-rate traffic against the victim, which potentially could allow the attack to bypass intrusion detection systems monitoring high rate traffics.

In this work, the fundamentals of the attack are presented, dealing with both iterative and concurrent servers. Besides, an analysis of its design and the evaluation of the performance of the attack is contributed. For this purpose, a mathematical framework is developed and besides, the different results are contrasted in both simulated and real environments.

A.1 Introduction

Nowadays, denial of service (DoS) attacks are one of the most serious security problems in Internet, due to the fact that they threaten not only technical aspects of trade but also give rise to financial expenses. In recent years, many companies have been affected by this kind of attacks, e.g. eBay, Amazon and Buy.com [1]. The increasing menace of these attacks parallels the difficulty in detecting, preventing and neutralising their effects. Moreover, modern DoS attacks use a distributed paradigm (DDoS) to achieve their purposes [2], which makes the process of detection and prevention even more difficult. Some examples of these attacks are presented in [3].

The aim of DoS attacks is to reduce or completely subvert the availability of the service provided to legitimate users. A common strategy used by an intruder to cause a DoS on a given target is to flood it with a continuous stream of packets that exhausts its availability. DoS attacks that use this kind of strategy are called brute-force or flooding attacks [2]. On the other hand, many others DoS attacks rely on the exploitation of a specific vulnerability in such a way that it results in a denial of the service.

Many efforts have also been made, in parallel with the evolution of DoS attacks, in the field of prevention and detection in networking security. In terms of prevention, some of the approaches that have been proposed include egress [4] or ingress filtering [5], disabling unused services [6], and honeypots [7]. However, although prevention measures offer increased security levels, they cannot completely eliminate the risk of an attack. It is advisable to establish an intrusion detection system (IDS) [8] aimed at detecting attacks that seek to bypass prevention techniques. Many proposals have been made for IDSs designed to detect DoS attacks [9, 10, 11], most of them being based on the statistical detection of high traffic rates coming from the intruder or intruders.

As a main contribution of this work, a new application level DoS attack [12] is described. The attack is able to succeed in defeating a server only by sending it low-rate traffic, in an intelligent way. Thus, it would be able to bypass detection mechanisms based on the monitoring of high-rate traffic. Kuzmanovic et al. presented in [13] a low-rate TCP attack that has similar characteristics to the one presented here; some methods for detecting it have since appeared [14, 15, 16, 17]. Fundamentally, both types of attack (TCP and application attack) try to take advantage of the vulnerability caused by the possibility of an intruder being aware of a specific time value concerning a protocol or application. Thus, both attacks inflict an ON/OFF waveform attack that results in overall low-rate traffic but with high efficiency in service denial. However, despite certain similarities, there are key differences between them. In the first place, the attack presented in [13] is TCP-targeted, whilst ours works at the application level, independently on the specific transport protocol needed. Secondly, Kuzmanovic's attack attempts to trigger the TCP congestion control mechanism by creating outages in a link, while ours simply seeks to overflow a service running in a machine, not creating network congestion at all. There are also differences in the vulnerability exploited in the two cases. In the TCP-targeted low-rate case, the knowledge of the RTO timer for congestion control handled in TCP is exploited, whilst in our case, inter-output times and fixed timeouts in the behaviour of the server are the key aspects to building the attack, as it will be shown. But the main difference between TCP and application attacks lies in the fact that although the former generates denial in some TCP flows, another flow with a special congestion control configuration could eventually bypass the attack and thus find the whole capacity of the link usable. In other words, the link capacity is free almost all the time. However, in our proposal the server is kept busy all the time, creating the perception that the server is not reachable among legitimate users. This latter feature is similar to the behaviour of the `Naptha` attack [18], although the main difference is that `Naptha` is carried out as a brute-force attack (high-rate traffic), while ours uses low-rate traffic to achieve its purpose. This discussion leads us to conclude that these attacks are not similar or comparable but complementary.

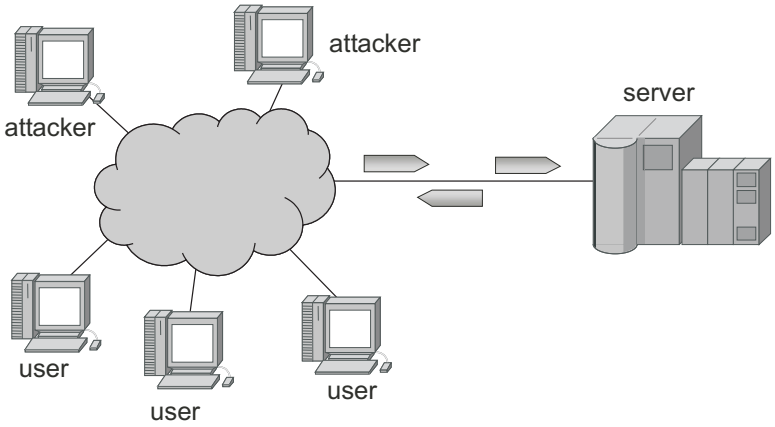


Figure A.1: Scenario of study.

The contributions of this work can be summarized as follows: firstly, a new type of DoS attack, mainly characterized by low-rate traffic, and which works at the application level, is described. Secondly, a method for the evaluation of its performance is contributed. Finally, the potential effects of the attack on both simulated and real applications are analyzed.

The structure of this document is as follows: the Section A.2 models the scenario of study. The main strategy for the attack and the vulnerabilities exploited are explained in Section A.3. Section A.4 deals with the specification of the details of the attack, and describes its execution against iterative and concurrent servers. In Section A.5 an evaluation of the attack in both simulated and real scenarios is made. Finally, the conclusions and some future work are summarized in Section A.6.

A.2 Scenario modeling

Before addressing the description and analysis of "our" low-rate DoS attack, let us briefly describe the scenario considered in this study. It consists of a generic client-server configuration in which a server will receive aggregated traffic coming from both legitimate users and intruders. The incoming traffic to the server corresponds to requests for a specific service offered by the server –see Fig. A.1–. The sources of traffic reach the server through a generic network; the details concerning the network topology and structure are irrelevant for the purposes of the present study.

In this scenario, a preliminary simplification can be made related to the number of legitimate users accessing the server. As recommended in many teletraffic studies [19], the requests arrivals from a specific user are modelled using a Poisson distribution. Consequently, the distribution of the user inter-arrival times, T_a, corresponds to an exponential probability (see [19]):

$$P(T_a = t) = \lambda \cdot e^{-\lambda t} \tag{A.1}$$

where λ represents the mean arrival rate of packets from the user. Moreover, it has been shown in [20] that the aggregate traffic from N_u users, whose inter-arrival times are exponentially distributed, with rates λ_i, results in a traffic pattern with inter-arrival times that also follow an exponential distribution, with the following inter-arrival time:

$$\lambda = \sum_{i=1}^{N_u} \lambda_i \tag{A.2}$$

This means that, for the purposes of our study, it is possible to consider the aggregate traffic from N_u users as a single flow from one user that generates exponentially distributed traffic at a mean rate λ packets/s that complies with the above expression.

Concerning the number of attackers in the scenario, this is exclusively determined by the manner in which the attack is launched by the intruder/s. If a distributed DoS technique is used, this number will be high, otherwise just one attacker should be considered. The conclusions drawn from this study are valid whatever attack technique is used. Therefore, for simplicity, we shall consider that the intruder launches the attack from a single point.

The proposed model for the server is depicted in Fig. A.2. It represents a farm of M machines with a common load balancer that sends each arriving requests to one of them. Once the server has been chosen, the incoming request is sent to be queued up in a finite length queue within that server (*service queue*). Of course, if no free positions are found, an overflow event, in the form of a log, a response, an event or even no response at all, is raised.

The requests allocated in a service queue remain in it during a *queue time*, t_q^i (being i the number of the considered service queue, $1 \leq i \leq M$), before passing to the module in charge of processing the petition, that is, the *service module*. Inside this module, a number of N_s^i processing elements can be present. Every processing element is able to serve only one request at a time. These elements can represent either threads or children processes of the parent process that implements the server. They could be running either in only one or in several processors within the machine. The total number of processing elements is $N_s = \sum_{i=1}^{M} N_s^i$ and they all are supposed to be identical.

Each processing element spends a time called *service time*, $t_s^{i,j}$, processing a request j in the service module i. Once the processing is finished, an answer is sent to the corresponding client. We will refer to these answers from the server as *outputs*.

Note that the service time $t_s^{i,j}$ is expected to be different for each request. It is mainly due to the usual different nature of the requests. This way, in a typical concurrent server, like a web server, the service time has been typically modelled as a heavy-tailed distribution, as it is dependent on the requested resource [21].

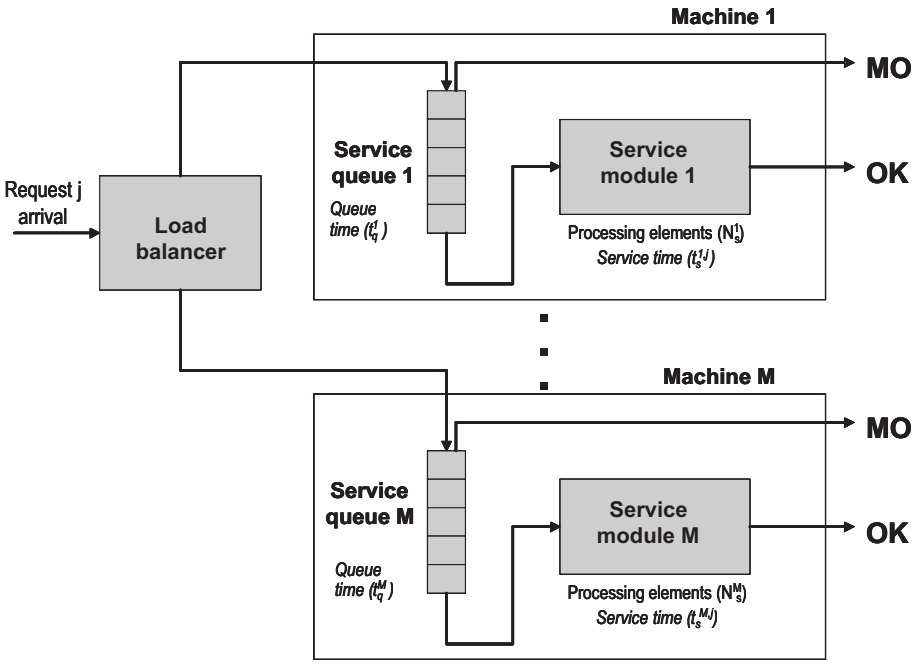


Figure A.2: Model for the server.

A.3 Main strategy for the attack

In the specified scenario, a DoS attack could be aimed at filling all the positions in the service queues with requests coming only from the intruder, preventing legitimate users from entering their requests in the queues. This aim is usually achieved by flooding the server with request packets, in the hope that most of the queued requests come from the attacker. This fact is observed by legitimate users as a denial of the offered service. The paradox is that, although the server is serving requests all the time, legitimate users cannot benefit from this.

Traditionally, the task of flooding the service queues has been carried out by means of a so-called *brute-force* attack, that is, the intruder (or intruders) sends as many requests to the server as possible. Although the aim of the attack presented in this work is the same, namely to "capture" the highest possible number of positions in the server queues, the methodology used by the attacker is slightly different. In our case, the intruder predicts the instants at which the outputs are raised in the server and, therefore, free positions in the service queues appear. This mechanism would allow the intruder to choose the key instants at which to send requests to the server, and thus to improve the efficiency of the attack in terms of the number of needed attack packets, in comparison with the case in which no knowledge of the behaviour of the server is possessed. Thus, whenever an intruder gets information about an instant at which a target server is going to raise an output,

this knowledge allows to strike an attack against the server in an intelligent way, by the generation of low-rate traffic of request packets.

Therefore, the key to success in denying the service relies on the ability to forecast the optimum instant at which the attack requests should be sent in order to acquire newly freed positions. In this way, the attack would eventually capture all the positions in the queue, thus causing the desired effect of denial of service to legitimate users. Obviously, the first question we must ask is: is it possible to forecast the instant when a position in the service queue is going to be freed? Our hypothesis is that, under certain circumstances, and by a simple inspection of the outputs generated by the server, this is not only possible but also relatively easy.

For predicting the instants at which the outputs are generated in the server, two possible strategies could be followed. The first one is designed for iterative servers, that is, servers that are only able to process one request at a time. And the other one is designed for concurrent servers, that is, servers that allow the processing in a (virtual or not) parallel mode. Next, the vulnerabilities exploited in both cases are described.

A.3.1 Vulnerability in iterative servers

Instead of addressing the task of forecasting the specific instant when an individual output will be generated by the server, we could tackle the problem by considering the time elapsed between two consecutive outputs from the server, that is, the *inter-output time*. In order to predict this time, it is necessary to analyze the complete process followed by a client request when it arrives to the server. This process is formalized as follows.

When a request enters the server, it is stored only if the service queue has free positions. Otherwise, an overflow message or event is raised and the request is discarded. A queued request waits in the queue during a queue time t_q. After t_q, the request passes through to the service module, where it is parsed, processed and served during a service time, t_s. Finally, the server sends the response to the client. After this, and if the service queue is not empty, the next request in the queue is fetched by the service module. At this point, a free position appears in the service queue. It is important to notice that, in the overall process, the instants when the free positions appear coincide with those at which the outputs are raised by the server. Because of this, the problem of forecasting the instants when a position is freed in the service queue is reduced to that of ascertaining the moments at which an output is going to be raised.

In short, we can conclude that the knowledge of the inter-output time τ, defined as the time elapsed between two consecutive outputs in a server, could constitute a vulnerability to be exploited by an attacker. For clarity, let us examine a simple scenario to discover how the intruder could guess the inter-output time. In this scenario, a fixed service time is considered. Although, at a first glance, this might seem a very restrictive case, it will be shown that the results are also valid for more complex cases in which this restriction is not made.

The considered scenario consists in a server with N positions initially occupied in the service queue. We study the period of time for which all the requests are served, assuming

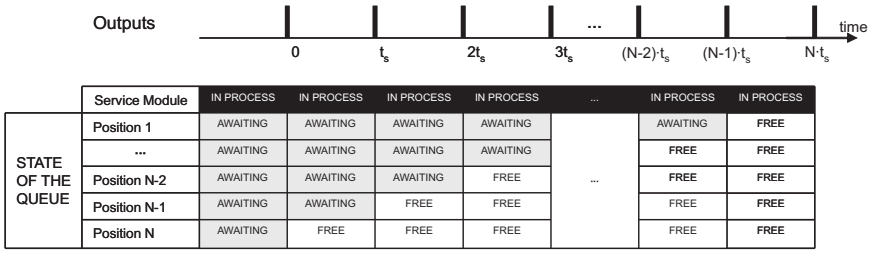


Figure A.3: Time diagram of the state of the requests in the service queue (bottom) and the associated timing of the generated outputs (top) for the considered example. In black, service time.

that no new requests enter the system. It is clear that, during the observation period, the service module is always engaged in the processing of a request. Both the state of the service queue, with a per-position detail, and the timing of the outputs generated by the server are shown in Fig. A.3. It can be observed that when a given request is in the "in process" state, the others are either waiting or free, due to the fact that the server is iterative. Thus, a request enters the service module at $t = 0$, reaching the "in process" state. After t_s seconds, the position is freed and the next request fetched. Again, after t_s seconds, the other position is liberated and a new request comes into the service module. This mechanism continues until the service queue is empty. It is noticeable that in the complete process the inter-output time does indeed correspond to the service time, and is not influenced by the queue time.

The service time for the incoming requests normally varies depending on the specific resource or operation solicited. Moreover, due to the different service times required, the inter-output time also varies from one output to the next. However, if most of the incoming requests to the server come from an attacker, the duration of the inter-output time can be set to a fixed value by always asking the server for the same resource. In this latter scenario, the service time could be considered fixed and deterministic, as in the example shown in Fig. A.3. However, even in this case, the main hypothesis in our study is that the service time behaves like a random process, T_s, due to the fact that the server processor is usually running other processes at the same time, which introduces variations into the value of the service time. Moreover, we hypothesize that the variations, in a stable load machine, will be low. Thus, we model the random process that represents the service time, T_s, by a normal distribution $\mathcal{N}(\overline{T_s}, var[T_s])$. The choice of the normal distribution is derived from the central limit theorem [22], because of the high number of random variables contributing to T_s (e.g. number of processes or threads, memory occupation, disk utilization, etc.).

Under this hypothesis, the expression for the inter-output time τ_s, when all the requests are identical, corresponds to:

$$\tau_s = \mathcal{N}(\overline{T_s}, var[T_s]) \tag{A.3}$$

Once we have established the relation between the service time and the inter-output time, a mechanism could allow a potential intruder to acquire knowledge about the service

time, based on the observation of the inter-output times. The estimation process can be performed by simply sending a set of probes, each one consisting of two consecutive identical requests, very close in time to ensure that they enter the target service queue consecutively, and, after that, observing the time elapsed between the reception of the corresponding outputs generated by the server. Note that the value of the inter-output time perceived by the intruder will differ from that directly observed in the server. This fact is due to the influence of the round trip time (RTT) experienced by the packets that traverse the portion of the network between the intruder and the server. Nevertheless, if all packets are assumed to be affected by this time, we can conclude that the round trip time will not modify the mean value of the observed inter-output time. However, this does introduce an additional variance into the general random process.

Some studies have shown that it is possible to represent the different values obtained for RTT as a random process modelled by a truncated normal variable [23]. This fact allows us to suppose that the estimation of the inter-output time made by a potential intruder located in the network, τ_u, also has a normal distribution, with the following characteristics:

$$\tau_u = \mathcal{N}(\overline{T_s}, var[T_s] + var[RTT]) \tag{A.4}$$

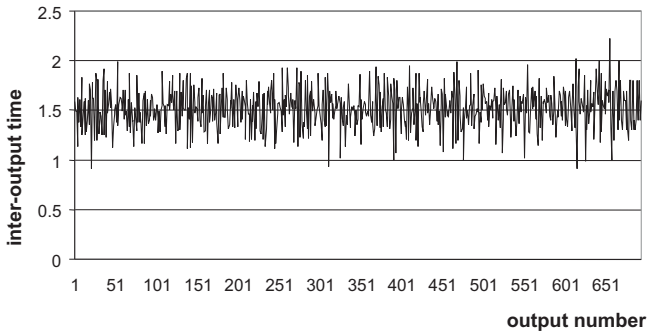
From this, it can be noted that the round trip time does not affect the mean value of the observed inter-output time, but only its variance.

Finally, it is important to notice that the service module must always be engaged serving requests as a necessary condition to justify the above assumptions and conclusions. In other words, the service queue must always have at least one pending request. Besides, it is remarkable that, in order to make possible the forecasting of the inter-output time, all the requests sent by the intruder against the server should be identical. Only in this way Eq. (A.4) is applicable.

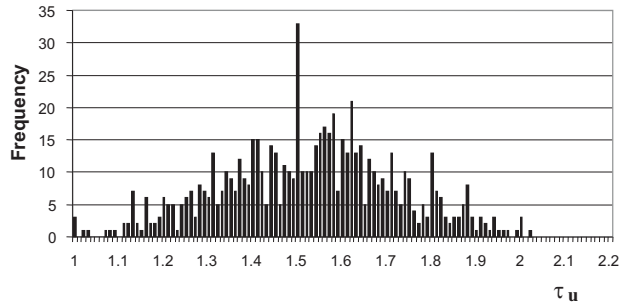
The scenario presented in Section A.2 has been implemented in a simulated environment. Thus, Network Simulator (NS2) [24] has been used to check whether Eq. (A.4) is a good approximation for the observed inter-output time, τ_u. For this purpose, we have implemented a new class derived from the application class that behaves like a server (*server class*), which makes traces and logs to track all the events and statistics necessary for our study.

Several experiments have been carried out to confirm the expected values for the observed inter-output time, as presented above. The case where the service time and the round trip time are modelled with a normal distribution has been considered. As shown below, the results obtained are very close to those predicted.

Fig. A.4 shows some simulation results. In this case, the service module has been kept busy processing identical requests. To achieve this, a traffic with an inter-arrivals mean time of $\overline{T_a} = 1.0\ s$ has been offered to the server. Other values used in the simulation are $\overline{T_s} = 1.5\ s$, $var[T_s] = 0.02$ and a round trip time $RTT = \mathcal{N}(0.6\ s, 0.02)$. Forty positions are considered for the service queue. It should be noted that $\overline{T_a}$ has been adjusted so that there is always at least one request in the service queue, that is, the arrival rate is greater than the service rate.



(a)



(b)

Figure A.4: Simulation of observed inter-output times in a flooded service queue scenario: (a) Inter-output time values and (b) histogram of the samples.

The simulation for a time period of 1000 s provided 694 outputs. The expected observed inter-output time distribution is $\mathcal{N}(1.5\,s, 0.04)$, as derived from Eq. (A.4). The mean value obtained from the simulation for the inter-output time is $\overline{\tau_u} = 1.520\,s$, with a variance of $var[\tau_u] = 0.041$, which accurately approximates $var[T_s] + var[RTT]$, as shown in Eq. (A.4).

We have also checked, through the Kolmogorov-Smirnov test of similarity [25], that the histogram obtained for the inter-output times approximates the normal probability function –see Fig. A.4(b)–. The value obtained for a significance level equal to 20% (very restrictive case) is 0.034, which indicates that a normal distribution is a good approximation for the histogram obtained.

The results obtained from this set of experiments show that the assumptions made for the functioning when the service module is always engaged are good enough. They also show that, even in simulated scenarios in which the service time varies as a normal distribution, the inter-output time may still be predictable for an observer. This makes it possible to build up an attack on the basis of this vulnerability.

A.3.2 Vulnerability in concurrent servers

In the case of concurrent servers, the instants at which the outputs are going to be generated in the server cannot be forecasted by inspecting the inter-output time as in the iterative server case. The main reason is that the behaviour of the different processing elements will make the inter-output time not to always follow the same pattern. In the thesis, a lot of results taken from different experiments that support this previous statement are contributed.

However, although the vulnerability in the iterative server case is not exploitable in the same way for concurrent servers, regretfully, it is still possible to attack a concurrent server with a low-rate traffic. Whenever the behaviour of a server is designed in such a way that it is possible to forecast the output instants, a low-rate attack could be launched against it.

Some examples for servers that keep the previously given condition can be given. First, consider a server that sends a publicity video information to those clients that request it. The server is designed with a capacity for sending the video stream to a limited number of users. Once that a user request the video stream and it begins, a position in the server is occupied. In this situation, if the video lasts always the same time, it is relatively easy to forecast the instant at which the video stream will finish (this is an output in our model) and, therefore, a new position could be seized by another client.

Another example is a concurrent server commonly used that can be attacked by this method is a web server that implements the HTTP 1.1 persistent connections feature. Due to its wide use in Internet, which makes it an important case study, its vulnerability is discussed in the following.

Case study: the HTTP server with persistent connections feature.

The persistent connection feature, that appears in the HTTP 1.1 specification [26], allows a web server to maintain a connection alive during a specified interval of time after a HTTP request has been served. This is used for reducing the traffic load in the case that several requests are going to be sent to the server by the same user, in a reduced time interval. Thus, for the first request, a connection is previously established with the server, the request is sent and, after that, the server waits a fixed amount of time before closing the connection[1]. If a new request arrives on this connection before the expiration of the mentioned time, the timer is reset again. This mechanism is repeated a fixed number of times, after which the connection is closed. In this scenario, the attacker could follow the next strategy in order to predict the instant at which an output is going to be raised:

- The attacker establishes a connection with the server. Making an analogy to the server model, this connection will occupy a position in the service queue.
- The attacker sends a HTTP request to the server on the established connection.

[1] In an Apache 2.0 server, the directive *KeepAliveTimeout* controls this timeout.

- The connection will be waiting in a service queue during a queue time for its turn to enter the service module i.
- After t_q^i seconds, a processing element extracts the connection from the service queue i, processes the request and answers (HTTP response) to the attacker. This response will reach the attacker at the instant t_{resp}.
- A *timeout* with a fixed value t_{out} is scheduled in the processing element before closing the connection. In the model for the server, t_{out} will play the role of T_s, because all the requests will wait this fixed time.
- When t_{out} expires, the connection is closed and, consequently, a new connection is extracted from the service queue, leaving a free position in the queue. This event implies, therefore, the raise of an output.

Note that as t_{out} plays the role of the service time in our model, it should also be modelled as a random variable, T_{out}. Making the same assumptions as in the iterative server case, its probabilistic distribution corresponds to a normal variable:

$$T_{out} = \mathcal{N}(\overline{T_{out}}, var[T_{out}]) \tag{A.5}$$

Thus, considering the instant t_{resp} as the origin for the time, the random variable that represents the instant at which the output is going to be raised from the server, T_{salida}, could be calculated as:

$$T_{salida} = \mathcal{N}\left(\overline{T_{out}} - \frac{\overline{RTT}}{2}, var[T_{out}] + var\left[\frac{RTT}{2}\right]\right) \tag{A.6}$$

Consequently, the attacker should sent the attack packets to the victim server at $t = \overline{T_{salida}} - \overline{RTT}/2$, for synchronizing their arrival around the instant of the raising of the output.

Note that, unlike in the iterative server case, in the HTTP persistent server case there is no need for always sending the same request to the server, because the timeout is independent on that. This fact makes the attack more dangerous, mainly because it could remain hidden for detectors that analyze the pattern of requests received by the server in order to detect an abnormal amount of identical requests.

This example could be extended to any concurrent server that exhibits a behavior in which a timing scheme could be known by a potential attacker. Of course, the way of predicting the instants of the outputs should be adapted for each particular case.

A.4 Low-rate DoS attack specification

As explained above, the intruder seeks to forecast the instant when the next output will happen, and thus focuses the attack only during a short period around the predicted instant,

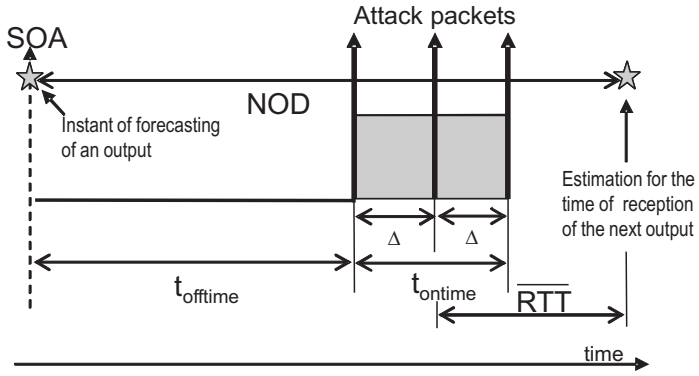


Figure A.5: Attack specification: waveform and parameters.

in order to "capture" the newly freed position in the queue. Only in this way the low-rate traffic can efficiently afflict a DoS attack.

However, as can be seen in Eq. (A.4) and (A.6), the instant at which the output is generated is forecasted as a probability function, so that the intruder only knows that the output is going to be generated with some probability within a given time interval.

In consequence, the proposed attack is designed to follow an ON/OFF pattern, that is, it comprises a succession of consecutive periods composed of an interval of inactivity, called *offtime*, followed by an interval of activity, called *ontime*, that will be centered around the predicted interval of the occurrence of the output. As depicted in Fig. A.5, the attack waveform is characterized by the following parameters:

- *Start of the attack period* [2] (*SOA*): each attack period starts with the forecasting by the intruder of the instant of occurrence of an output in the server.

  In the iterative server case, the reception of a response from the server delimits a new period of attack. However, it is important to remark that the beginning of the attack period is not always determined in this way. In effect, the intruder will receive a packet containing an answer from the server only when this corresponds to a request previously sent by the intruder. Otherwise, the response would be sent to the corresponding origin, in such a way that the intruder would not notice this fact. In this case, the attack period will start at the estimated instant at which the output should arrive at the intruder.

  On the other hand, in the concurrent server case, the beginning of the attack period depends on the considered victim server. Concerning the HTTP persistent case, the reception of a HTTP response will initiate the attack period.

[2] The original acronym for this parameter in the thesis is CPA.

- *Next output distance* [3] (NOD): this is the time elapsed from the start of an attack period until the predicted instant value for the reception of the next output. This time will correspond to:

$$NOD = \overline{T_s} \tag{A.7}$$

- *Ontime interval* for the output k ($t_{ontime}(k)$): the interval during which an attempt to seize a freed position in the service queue due to the output k is made by emitting request packets. These packets should arrive at the server around $\overline{T_{salida}}$.
- *Offtime interval* for the output k ($t_{offtime}(k)$): the interval before *ontime* in the period of attack corresponding to the output k during which there is no transmission of attack packets.
- *Interval* for the output k ($\Delta(k)$): the period of time comprised between the sending of two consecutive packets during the *ontime* interval.

An attack period starts whenever the attacker receives a response from the server. At this time, the value of $t_{offtime}(k)$ is obtained as:

$$t_{offtime}(k) = \overline{T_s} - \overline{RTT} - \frac{t_{ontime}(k)}{2} \tag{A.8}$$

where the mean value for the service time, $\overline{T_s}$ should be substituted by $\overline{T_{out}}$ in the HTTP persistent server case, and by $\overline{\tau_u}$ in the iterative server case.

For simplicity, the parameters for the design of the attack period, $t_{ontime}(k)$ and $\Delta(k)$, are made independent of the output k, and their values are consequently made fixed for all the outputs: t_{ontime} and Δ.

A.4.1 Phases of the attack

Having described the philosophy of the attack and its waveform, let us now describe in detail how it is carried out along the time. Starting from an initial prediction about the inter-output time of the target server, the attack takes place in two phases. Firstly, there is a transitory phase in which the intruder attempts to fill all the positions in the service queue. This process could be achieved through a brute-force strategy, although this is not the best solution if the intruder wants to bypass potential security detection mechanisms adopted at the destination. Instead, the intruder could use the attack dynamics proposed below, which are still effective, although requiring more time to complete the first phase.

After occupying the service queue, the second phase of the attack starts. The objective now is to maintain as many positions in the service queue as possible. This is carried out by sending a new request in such a way that it arrives at the server in the minimum possible time after a position is freed. To do so, the attack waveform presented in Fig. A.5, synchronized with the outputs from the server, is used.

[3] The original acronym for this parameter in the thesis is DSS.

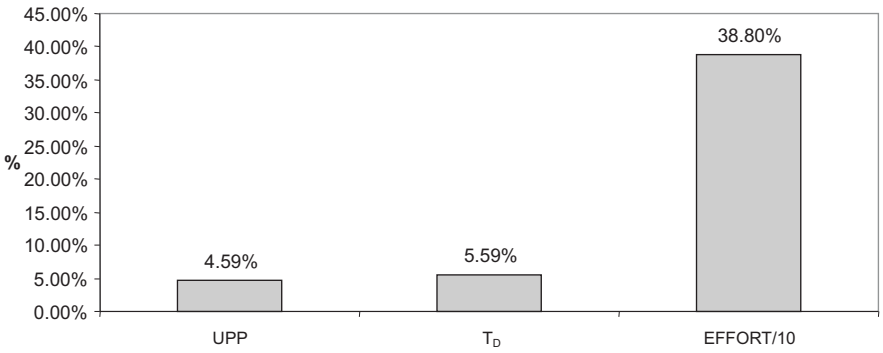


Figure A.6: Attack against an iterative server dynamics. Example of a period of attack.

A.4.2 Execution against iterative servers

A detailed study of the dynamics for an attack period against iterative servers is made in the following. Fig. A.6 shows a representation of the different events and parameters in an attack period along the time, from the intruders perspective as well as that from the server. Just after the reception of an output from the server (O1), an attack period starts (SOA_1) and the next output distance (NOD_1) is obtained, giving, along with the estimation of $\overline{RTT}$, the value of $t_{offtime}(1)$ for this period. At the end of this inactivity period, $t_{ontime}(1)$ is started by emitting an attack packet (A2). Within the *ontime* period, an attack packet is sent every *interval*, Δ (A3 and A4). In case the timer associated to NOD_1 expires, a new attack period starts (SOA_2), again with a calculation for the corresponding parameters NOD_2 and $t_{offtime}(2)$.

If no response is received from the server during this *offtime* time, the new attack period behaves in the same way as the previous one. This situation occurs whenever the response received from the server corresponds to a previously queued request coming from a legitimate user instead of from the intruder. However, when a response is received from the server, as illustrated in Fig. A.6 (O2), the attack period restarts ($SOA_{2'}$) and new parameters $NOD_{2'}$ and $t_{offtime}(2')$ are calculated. This reset mechanism allows to resynchronize the attack each time it fails in the prediction of the timing of the outputs.

In accordance with the explained procedure, whenever an output is received by the attacker, the attack period is restarted, independently of the running interval (*offtime* or *ontime*) at the reception instant. Moreover, upon this output arrival (O1 or O2), an attack packet is sent as a response to it (A1 and A5). The reason for the sending of this packet is to minimize the time during which the new freed position in the queue is available, in case that the packets sent during the *ontime* period did not succeed in the seizure.

In summary, two possible events may trigger the start of a new attack period. First, the reception of a new output packet by the intruder. This fact starts a new period even if the current one has not finished. Second, the expiration of the NOD timer. In this latter case, the period is restarted but the intruder will not send an attack packet at the

beginning of the attack period because it is not clear that a queue position has been freed. This behaviour reduces the traffic rate of the attack.

A.4.3 Execution against concurrent servers

One possible strategy to be followed by the attack software when attacking concurrent servers is a sequential scheduling of the different attack periods, as the outputs are predicted. The main problem of this design is that, when two or more outputs are given very close in time, the corresponding attack periods overlap, making the control of the attack software complex and not scalable. For this reason, the attacker could design the malware as a multithreaded process, in which every thread is in charge of seizing only one position and maintaining it as long as possible.

With this multithreaded design, a new parameter appears for the attack, the *number of attack threads*, N_a. It could be adjusted in order to reach a specific level of denial of service depending on the maximum traffic level allowed to be sent against the server.

On the other hand, we already know that whenever an attack thread seizes a new position in the queue, a new attack period is scheduled. Conversely, if a new connection has not been made, possibly because another user has seized the freed position, something should be made by the considered attack thread. The strategy followed by the attack thread in the latter case is called a *tracking strategy*. Next, the fundamentals of the strategies proposed in the thesis are summarized:

- *Basic tracking strategy*: whenever an attack thread fails in seizing a position in the queue, it tries to recover that position blindly by means of sending an attack packet every *recovery interval*, Δ_r. Obviously, the setting of this parameter will affect the final amount of traffic sent to the victim server.

- *Inter-thread information sharing strategy*: during the execution of the attack, each attack thread will seize at least one position in the service queues of the server. For these seized positions, the attack thread has to forecast the instants at which the outputs are going to be raised. The main purpose of this strategy is to allow any attack thread to share this information with others, in such a way that the amount of time during which the different attack threads have positions in the service queues is maximized.

  This strategy has been compared with the first one obtaining results that proof (from the intruder's perspective) its benefits. All of the experiments and results are detailed in the thesis.

- *Seizures threshold strategy*: in this strategy, the number of positions kept in the different service queues of the server is limited by a threshold. Whenever this threshold is achieved, the attack software only maintains the captured positions without trying to get more. This strategy, also detailed in the thesis, presents some advantages:

  – It allows the intruder to select the optimum number of attack threads, N_a, to be spawned.

– It also gives some guidance to the setting of the parameter Δ_r in the design of the attack.

– The attack is controlled at any time by the intruder, making it possible a slow (and succesive, if needed) increase of the traffic sent to the server.

A.5 Evaluation of the attack

As previously indicated, the evaluation of the attack must respond to two main questions. On the one hand, which degree of denial of service could an intruder achieve? And on the other hand, what rate of traffic is necessary to succeed in this purpose? The next subsections deals with that.

A.5.1 Indicators for evaluating the attack

In order to measure the traffic rate needed to carry out the DoS attack, we define the *effort of the attack* [4] *(E)*, as the ratio, in percentage, between the traffic rate generated by the intruder and the maximum traffic rate accepted by the server. It is important to note that although E measures the relative traffic rate involved in the attack, this parameter does not give an idea of the congestion level in the server, due to the fact that the latter depends not only on the traffic coming from the intruder but also on the legitimate users' requests.

To measure the level of DoS suffered by legal users, let us examine the simplification made in Section A.2, which allows us to consider a single user, who generates all the aggregated traffic coming from all the legitimate users. With this consideration in mind, we define the *user perceived performance* [5] *(UPP)* as the ratio, in percentage, between the number of user requests served by the server, and the total number of requests sent by this user. This parameter gives an idea about the service level experienced by legitimate users.

Although UPP is a good indicator of the DoS attack effectiveness, it depends on the traffic pattern of the legitimate users. Alternatively, we could consider a measure of the effectiveness of an attack independently of the user traffic pattern. This hypothetical indicator would enable a comparison between different design strategies and would make it easier to take a decision about the values to adopt for the attack parameters. With this fact in mind, we define, in a scenario free of user traffic, the *available time*[6] (T_{AV}), as the average time, in percentage, during which a free position in the service queue is available within an attack period. T_{AV} is an objective measure about the efficiency of the attack, because from it, the probability of a legitimate user seizing a queue position could be deduced.

The aim of the attack, in terms of the indicators defined (see Table A.1), should be to minimize the user perception on the availability of the service (UPP). This task can be achieved by minimizing the *available time* in the server, which reduces the probability of a legitimate user seizing a position in the queue. Additionally, the attack should minimize

[4] In the thesis, this indicator is called *sobrecarga*, S.
[5] In the thesis, this indicator is called *disponibilidad*, D.
[6] In the thesis, this indicator is called *tiempo disponible*, T_D

Rate Evaluation	Efficiency Evaluation
Effort (E)	User perceived performance (UPP)
	Available time (T_{AV})

Table A.1: Defined indicators for efficiency and rate evaluation of a low-rate DoS attack.

its *effort (E)*, thus making the attack less detectable by intrusion detection systems. As an expense, it is expected that the reduction in the *available time* will necessarily increase the *effort* of the attack, and vice versa.

A.5.2 Mathematical models for the evaluation of the attack

In the complete version of the thesis some mathematical models for the evaluation of the attack are contributed. These models establish a relation between the attack configuration parameters and the performance obtained with this settings, in terms of the previously presented indicators.

The development of these analytical models heavily rely in the concept of *available time*. First, a model for the evaluation of the *available time* is contributed. After that, the values for the *user perceived performance* are based on the previously obtained expressions for the *available time*. Finally, an expression for the effort is also contributed.

The analytical expressions of the mathematical model have been contrasted through a lot of experiments in which the different indicators are obtained in both simulated environment and from the model. These experiments verify the accuracy of our model in evaluating the behaviour of the attack.

In summary, these models are quite useful for both deciding the settings of an attack, and also getting a comprehensive knowledge about the attack behaviour, which is essential for building possible defense techniques.

A.5.3 Evaluation of the attack against iterative servers

We now address the task of evaluating all the hypotheses presented for the attack against iterative servers. For this purpose, both simulated and real scenarios are used. Firstly, we evaluate how efficient the attack would be in a simulated scenario. Then, the attack is also tested in a real environment.

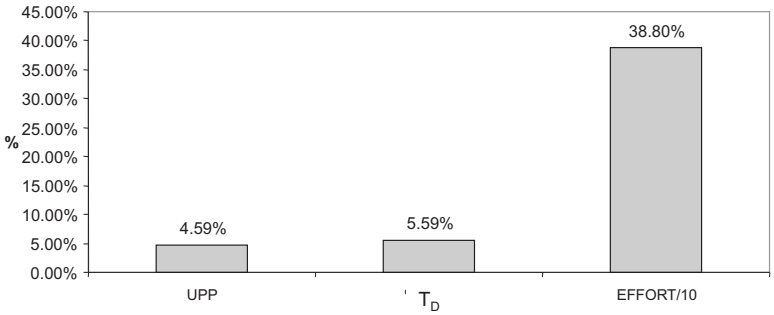


Figure A.7: Figures for indicators of an attack against an iterative server in a simulated environment.

Simulation results

To evaluate the efficiency of the attack against iterative servers, its behaviour was tested by using Network Simulator (NS2) [24]. An implementation of the attack, as well as the model for the iterative server, are made within NS2.

A set of variations of the attack has been tested within this environment, and worrying results derived, because of the high effectiveness demonstrated. For example, Fig. A.7 shows the results obtained from an attack simulation comprised of 1352 outputs. The attack was launched against a server with $\overline{T_s} = 3.0$ seconds and $var[T_s] = 0.2$. The traffic generated by the user, with $\overline{T_a} = 4.0\,s$, is close to keeping the server busy all the time by itself. The parameters of the attack for this example are: $t_{ontime} = 0.6s$, $t_{offtime} = 2.7s$, and $\Delta = 0.3s$. On the other hand, RTT was set to $\mathcal{N}(0.6\,s, 0.2)$. As previously stated, a very high level of efficiency is obtained, $UPP = 4.59\%$, which means that only 4.59 percent of legitimate requests are attended to by the server, and the percentage of *available time* during a period of attack is equal to 5.59%. On the other hand, an effort value of $E = 388\%$ was necessary to launch this attack, which indicates that the traffic offered to the server by the intruder approximates to four times the capacity of the server.

The results obtained from the experiments carried out show that there are multiple attack configurations available for the intruder to be able to adapt the attack to the desired aim. Thus, Fig. A.8 represents the *user perceived performance* versus the *effort* needed for a subset of the experiments. As shown, it is possible to adjust the *effort* of the attack and, therefore, the ability to bypass an IDS system capable of detecting attacks at a given rate, by reducing the value of *ontime* time, t_{ontime}, and/or by increasing the *interval*, Δ, at the cost of a reduction in the effectiveness of the attack.

Although the above results show the potential risks of the kind of attacks reported in this work, it is still interesting to compare the proposed attack, which presents a certain degree of intelligence, to one that uses the same effort but without any intelligence, that is, that sends the packets at instants taken from a poisson random distribution. The differences between these two strategies are revealed in the results shown in Fig. A.9, where the values obtained for the efficiency and rate indicators, *UPP* and *effort*, corresponding to three different attacks are illustrated. The service time in the server, T_s, has been adjusted

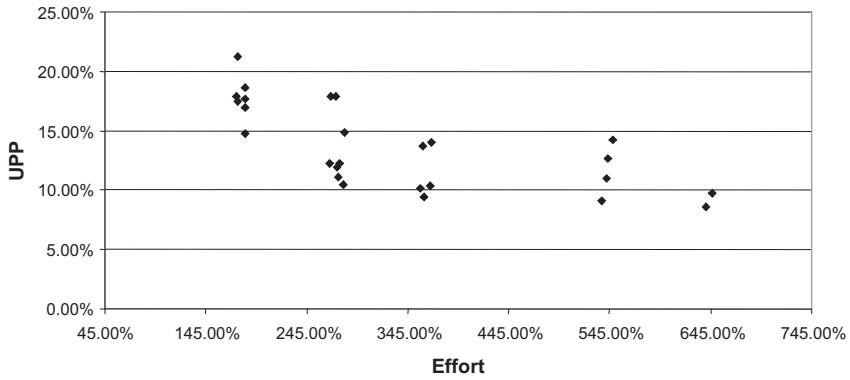


Figure A.8: User perceived performance *vs.* effort for 25 different configurations (t_{ontime} and Δ values) of the low-rate DoS attack against an iterative server.

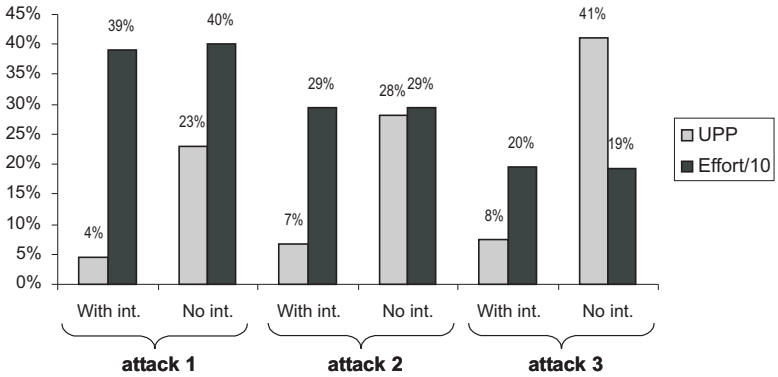


Figure A.9: Comparison between intelligent and non-intelligent strategies for three different attacks against an iterative server.

to $\mathcal{N}(5.0\,s, 0.2)$, and a traffic rate from legitimate users with $\lambda = 1/10$ packets/second has been generated. For each of the three attacks, both intelligent ("With int." series) and non intelligent ("No int." series) strategies have been used. As observed, the results show that, although the value of E is similar in both strategies, the derived values for *UPP* indicate that a considerably higher efficiency is achieved when running an intelligent strategy.

Finally, let us consider what would happen if the legitimate users increased their traffic rates so that the aggregate traffic becomes greater than that involved in the attack. We have made some experiments that demonstrate that even under these conditions, an intelligent attack attains a higher degree of DoS than does a non-intelligent strategy. Fig. A.10 shows the results obtained for an attack carried out on a server with a value for $T_s = \mathcal{N}(5.0\,s, 0.2)$, as in the previous experiments, but with legitimate users trying to reach the server at a rate of $\lambda = 1/1.66$ packets/second. In the absence of attack, a $UPP = 33\%$ is expected with this type of traffic, due to the fact that the rate of the users is higher than the service rate of the

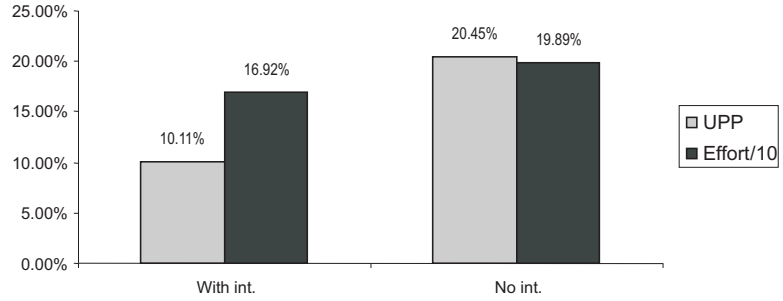


Figure A.10: Comparison between intelligent and non-intelligent strategies in an iterative server flooded by legitimate users.

server. As can be seen in Fig. A.10, the intelligent attack ("With int." series) improves the value of *UPP* by up to 50% compared with that obtained in the non-intelligent case ("No int." series), which are worrying results, even under these conditions.

In summary, the proposed attack seems to be very effective in terms of the denial of service degree afflicted to legitimate users. Moreover, its design allows the attacker to tune the traffic rate sent to the server in order to select a threshold, usually determined by the sensibility of intrusion detection mechanisms, in a way that allows the intruder to bypass them.

Real environment results

The proposed attack has also been tested in a controlled real environment to check its validity. The selected server is an Apache 2.0 web server that observes the condition of serving requests in an iterative way (`ThreadsPerChild = 1`). Although this is not a typical example of an iterative server, it is still useful for our purposes.

We assumed a client's petition to consist of a connection request to the server. The attack establishes connections and sends no messages on them, letting the web server close the connection after a timeout period specified by the Apache directive `Timeout`. This timeout is a deterministic and fixed value which corresponds, in our model, to the mean service time, $\overline{T_s}$. As discussed in previous sections, the variance in the inter-output time observed by the intruder, τ_u will be contributed to by variations in the service time caused by the operation of the server itself in a multiprocess environment within a non-real time operating system, and also by the variability of the round trip time.

The real scenario is analogous to that considered for the theoretical analysis. The user traffic was generated by following a Poisson process. Intruder software launches the attack from a single source. Both legitimate and intruder traffic traverse a WAN network to reach the server, with a round trip time modelled with the distribution $\mathcal{N}(17\ ms, 0.05)$. Traces on the users' and the intruder's side were established to obtain the data necessary to calculate the attack indicators.

$\overline{T_s}$	$\overline{T_a}$	*Environment*	*UPP*	*Effort*
3	3.5	Simulated	10.4%	260.8%
		Real	9.8%	239.7%
5	6	Simulated	5.7%	213.1%
		Real	7.8%	212.4%
10	12	Simulated	3.0%	198.3%
		Real	6.4%	201.6%
15	17	Simulated	3.0%	197.5%
		Real	2.5%	198.2%
20	22	Simulated	3.0%	197.5%
		Real	4.3%	196.2%
25	28	Simulated	1.8%	197.9%
		Real	1.8%	197.9%

Table A.2: Performance of real and simulated attacks.

Table A.2 shows some experimental results and a comparison between the real and the predicted values for different mean service times ($\overline{T_s}$) and user traffic inter-arrival times ($\overline{T_a}$). These times were selected in such a way that there was no congestion on the server when no attack was under way. The parameters of the attack were tuned to $t_{ontime} = 0.4$ s and $\Delta = 0.4$ s for all the experiments.

It can be seen that in some cases even better results in efficiency (lower *UPP*) can be obtained for the attack in a real environment. Moreover, some differences can be appreciated between the simulated and the real values. Such variations are due to the difficulty of modelling the service time and the round trip time distributions.

Two conclusions can be drawn from these results: (a) all the experiments made under simulation seem to provide results that are good approximations of the behaviour in real environments, and (b) the real impact of the attack is very high, showing that these vulnerabilities could be easily exploited in iterative servers.

A.5.4 Evaluation of the attack against concurrent servers

The evaluation of the attack against concurrent servers is made in the following. For this purpose, the performance of the attack, in terms of the defined performance indicators, E and *UPP*, is now assessed in both simulated and real environments. Although a comprehensive set of experiments could be found in the thesis, the most significative ones are presented here.

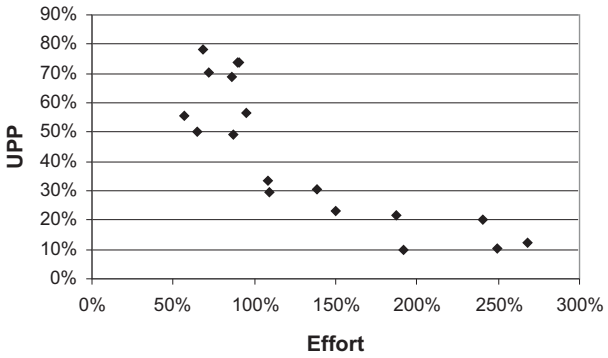


Figure A.11: Possible configurations for the attack: *UPP* vs *E*.

Simulation results

The efficiency achieved by the attack has been evaluated in a simulated environment where a low-rate DoS attack module as well as the legitimate users and the concurrent server have been implemented using Network Simulator 2.

Several experiments have been carried out to test both the efficiency and the flexibility of the attack. Regarding the flexibility, the results allow us to conclude that the attacker has a lot of possibilities for adjusting the attack parameters to obtain many possible combinations of efficiency and effort. Regretfully, this will allow to tune the attack parameters in order to bypass possible security mechanisms while a DoS is being made. The values for E and *UPP* obtained for 18 possible configurations of the attack in the same scenario are shown in Fig. A.11. The considered scenario is characterized by the values $N_s = 4$, $\overline{RTT} = 0.1\ s$, and $T_s = \mathcal{N}(12\ s, 0.1)$. Note that, for the attacker, a lot of configurations could be eligible and, what is more worrying, high denial of service levels (around $UPP = 10\%$) are easily reachable.

Additionally, the different experiments have shown that our attack strategy implies a big improvement for the DoS attack when compared with a similar rate of packets randomly sent (with no intelligence) to the server. Fig. A.12 shows the comparison for four different scenarios. For each one, both the values of *UPP* and E are represented for the "intelligent" attack as well as for a random flood of packets. Note that, as expected, in all the scenarios the efficiency obtained in the intelligent attack is considerably higher, even when the involved effort is lower.

Real environment results

A prototype that executes the low-rate DoS attack has been implemented in a Win32 environment in order to check its feasibility. The attack has been carried out against an Apache 2.0.52 web server hosted in a machine with Windows XP operating system. The server has

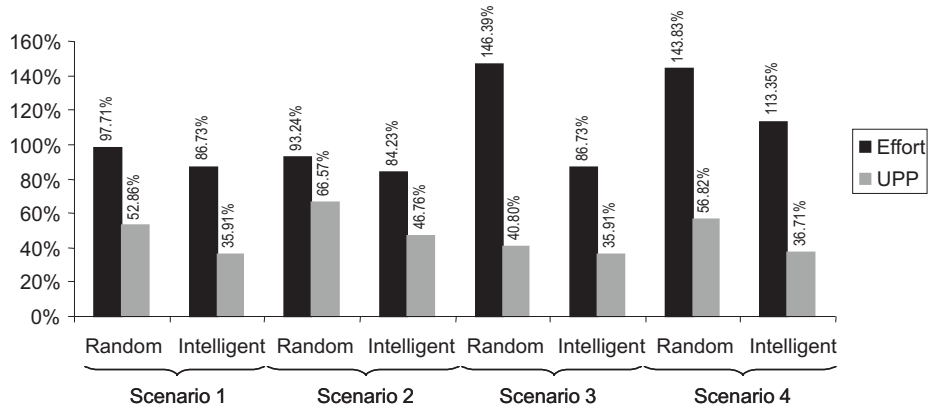


Figure A.12: Efficiency and effort obtained with the intelligent attack against a concurrent server compared with a random flood of packets, depicted for 4 different scenarios.

been configured with the directive $KeepAliveTimeout = 10$ seconds[7], which corresponds to the parameter $\overline{T_{out}}$ in our model. Besides, the directive $ThreadsPerChild$, which represents the number of threads for the processing of requests in the server, N_s, has been set in a range from 12 to 50.

The scenarios of the different experiments are analogous to that presented in Section A.2. The traffic from the legitimate users has been synthetically generated following a Poisson distribution within a range of inter-arrival time values, T_a. Traces for the legitimate users as well as for the intruder sides have been issued for collecting the necessary data to calculate the attack indicators.

Table A.3 shows the results obtained from eight different experiments taken from the set of experiments. For each different attack configuration, both simulation and real environment values for UPP and E have been obtained. Note that in the results there is a slight variation between the simulation and the real values, with even better results in some cases for the real environment than for the simulated one. These variations are mainly due to deviations in the estimation of the parameters for the statistical distributions of RTT and the service time. Nevertheless, the obtained results in the real environment confirm the worrying conclusions extracted from simulation, that is, our attack can achieve very high efficiency levels with a very feasible implementation.

A.6 Conclusions and future work

This work presents the fundamentals of a new kind of DoS attack aimed to defeating servers by sending a low-rate traffic against them. The attack exploits a vulnerability in servers regarding the possibility of forecasting, through statistical metrics about the behaviour of the server, the instants at which the server sends responses.

[7] The value for this parameter is 15 seconds by default in the Apache configuration file.

	E	UPP
simulated	86.73%	35.91%
real	81.74%	36.14%
simulated	84.23%	46.76%
real	74.57%	48.22%
simulated	113.35%	36.71%
real	109.00%	38.01%
simulated	268.47%	12.31%
real	269.82%	12.40%
simulated	68.04%	78.25%
real	76.00%	72.60%
simulated	249.49%	10.22%
real	256.41%	10.01%
simulated	89.88%	73.79%
real	92.41%	74.00%
simulated	191.56%	10.08%
real	183.80%	12.34%

Table A.3: Comparison between real and simulated environment results in the attack against a concurrent server.

Although this kind of attacks could be achieved by a brute-force technique, its main characteristic is that it can make use of low-rate traffic to subvert the provided service. Thus, it is possible to tune the parameters of the attack in order to select appropriate values for the efficiency and the amount of load generated in the target server. This characteristic makes it possible, under certain circumstances, to bypass existing IDS systems intended to protect the server, thus creating what is a non-detectable threat.

The fundamentals and design of a possible exploit have been explained. We have confirmed that such an attack could be very efficient and that it could be easily carried out. Moreover, its behaviour is neither detectable by parsing the packets that arrive to the server, due to the fact that they are similar to the users' requests, nor it is noticeable by high-rate detection.

Some different strategies for carrying out the attack has been developed depending on the nature of the server. Thus, two different possible strategies have been pointed out: the attack against iterative servers and the attack against concurrent ones. In the second group, an example of the attack in which a persistent HTTP 1.1 web server is the victim has been ilustrated.

The efficiency of the attack, in terms of the denial of service level and the amount of traffic directed to the server, has been evaluated in simulated and real scenarios, obtaining worrying results from both of them. Finally, a review of the behaviour of the attack is given when the different parameters of both the attack and the server are changed. The contributions of this study could be the starting point for the development of future defense techniques against the attack. This is the main research task open for future work on this field.

Finally, although there are similarities with the kind of attacks presented in [13], there are also key differences, namely that the attack presented here works at the application level, and takes advantage of the knowledge of timing schemes in the server to keep it constantly engaged serving intrusive requests. This fact points out the emergence of a new family of DoS attacks, characterized by a low-rate traffic that is shaped by the estimation of a specific timer or behaviour of the server.

Summary of the thesis: References

[1] M. Williams. *Ebay, Amazon, Buy.com hit by attacks*, 02/09/00. IDG News Service, 02/09/2000. http://www.nwfusion.com/news/2000/0209attack.html

[2] J. Mirkovic, P. Reiher. *A taxonomy of DDoS attack and DDoS defense mechanisms*, ACM SIGCOMM Computer Communication Review 34(2)(2004) 312-321.

[3] CERT coordination Center. *Denial of Service Attacks*, available from <http://www.cert.org/tech_tips/denial_of_service.html>

[4] Global Incident Analysis Center. - *Special Notice - Egress filtering*, available from <http://www.sans.org/y2k/egress.htm>.

[5] P. Ferguson, D. Senie. *Network ingress filtering: defeating Denial of Service attacks which employ IP source address spoofing*, RFC 2827, 2001.

[6] X.Geng, A.B.Whinston. *Defeating Distributed Denial of Service attacks*, IEEE IT Professional 2(4)(2000) 36-42.

[7] N.Weiler. *Honeypots for Distributed Denial of Service*, in: Proceedings of the Eleventh IEEE International Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises 2002, Pitsburgh, PA, USA, June 2002, 109-114.

[8] Axelsson S. *Intrusion detection systems: a survey and taxonomy*, Department of Computer Engineering, Chalmers University, Goteborg, Sweden. Technical Report 99-15; March 2000.

[9] J.B.D. Cabrera, L.Lewis, X.Qin, W.Lee, R.K.Prasanth, B.Ravichandran, R.K.Mehra. *Proactive detection of Distributed Denial of Service Attacks using MIB traffic variables - a feasibility study*, in Proceedings of the 7th IFIP/IEEE Internation Symposium on Integrated Network Management, Seattle, WA, May 14-18, 2001.

[10] J. Mirkovic, G.Prier, P.Reiher. *Attacking DDoS at the source*, in: Proceedings of ICNP 2002, Paris, France, 2002, 312-321.

[11] R.R.Talpade, G.Kim, S.Khurana. *NOMAD: Traffic-based network monitoring framework for anomaly detection*, in Proceedings of the Fourth IEEE Symposium on Computers and Communications, 1999, 442-452.

[12] C. Douligeris, A. Mitrokotsa. *DDoS attacks and defense mechanisms: classification and state-of-the-art*, Computer Networks 44 (2004) 643-646.

[13] A. Kuzmanovic and E. Knightly. *Low Rate TCP-targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants)*, in Proc. ACM SIGCOMM 2003, Aug. 2003, 75-86.

[14] H.Sun, J.C.S. Lui, and D.K.Y.Yau. *Defending Against Low-Rate TCP Attacks: Dynamic Detection and Protection*, in Proc. IEEE Conference on Network Protocols (ICNP2004), Oct. 2004, 196-205.

[15] G. Yang, M. Gerla, and M. Y. Sanadidi. *Randomization: Defense Against Low-rate TCP-targeted Denial-of-Service Attacks*, in Proc. IEEE Symposium on Computers and Communications, July 2004, 345-350.

[16] A. Shevtekar, K. Anantharam and N. Ansari. *Low Rate TCP Denial-of-Service Attack Detection at Edge Routers*, in IEEE Communications Letters 9(4) 363-365, April 2005.

[17] Haibin Sun, John C.S. Lui and David K.Y. Yau. *Distributed mechanism in detecting and defending against the low-rate TCP attack*, Computer Networks 50(13) 2312-2330, September 2006.

[18] SANS Institute. *NAPTHA: A new type of Denial of Service Attack*, December 2000. <http://rr.sans.org/threats/naptha2.php>

[19] R. R. Martin. *Basic Traffic Analysis*, Prentice-Hall Inc, September 1993, ISBN: 0133354075.

[20] <http://mathworld.wolfram.com/ExponentialDistribution.html>

[21] Z. Liu, , N. Niclausse, C. Jalpa-Villanueva. *Traffic model and performance evaluation of Web servers*, Performance Evaluation 46 (2-3) (2001) 77-100.

[22] R.E. Walpole, R.H. Myers, and S. L. Myers. *Probability and Statistics for Engineers and Scientists*, Sixth Edition, Prentice Hall College Div, 1997. ISBN: 0138402086

[23] T. Elteto and S. Molnar. *On the Distribution of Round-Trip Delays in TCP/IP Networks*, Proceedings of the 24th Annual IEEE Conference on Local Computer Networks, p.172, October 17-20, 1999

[24] Network Simulator 2. Available at: <http://www.isi.edu/nsnam/ns/>

[25] R. D'Agostino, M. Stephens. *Goodness-of-Fit Techniques*. Marcel Dekker, Inc. (1986).

[26] R. Fielding, U.C. Irvine, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, *RFC2068, Hypertext Transfer Protocol - HTTP/1.1*, Network Working Group, January 1997.

Bibliografía

[Agostino and Stephens, 1986] Agostino, R. D. and Stephens, M. (1986). *Goodness-of-Fit Techniques*. Marcel Dekker, Inc.

[Ahn et al., 2003] Ahn, L. V., Blum, M., Hopper, N., and Langford, J. (2003). CAPTCHA: Using hard AI problems for security. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2003)*, pages 294 – 311.

[Allman and Paxson, 1999] Allman, M. and Paxson, V. (1999). On estimating end-to-end network path properties. In *Proceedings of ACM SIGCOMM 99*, Vancouver, British Columbia.

[Arkin, 1999] Arkin, O. (1999). Network scanning techniques: Understanding how it is doing. Sys-Security Group. http://www.sys-security.com.

[Arkin, 2000] Arkin, O. (2000). ICMP usage in scanning, or understanding some of the ICMP protocol's hazard. Sys-Security Group. http://www.sys-security.com.

[Aura et al., 2003] Aura, T., P. Nikander, and Leiwo, J. (2003). DOS-resistant authentication with client puzzles. *Lecture Notes in Computer Science*, 2133/2001:170.

[Axelsson, 2000] Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy. Technical report, Department of Computer Engineering, Chalmers Univ., Goteborg, Sweden.

[Baran, 1964] Baran, P. (1964). On distributed communications. Memoranda RM-3420-PR, RM-3103-PR, RM-3578-PR, RM-3638-PR, RM-3097-PR, RM-3762-PR, RM-3763-PR, RM-3764-PR, RM-3765-PR, RM-3766-PR, RM-3767-PR.

[Bellovin, 1989] Bellovin, S. M. (1989). Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 19(2):32–48.

[Bellovin, 2001] Bellovin, S. M. (2001). ICMP traceback messages. Internet draft, http://search.ietf.org/internet-drafts/draft-ietf-itrace-01.txt.

[Benantar, 2002] Benantar, M. (2002). *Introduction to the Public Key Infrastructure for the Internet.* Prentice-Hall Inc.

[Bouchier et al., 1996] Bouchier, F., Ahrens, J., and Wells, G. (1996). Laboratory evaluation of the iriscan prototype biometric identifier. Technical Report SAND96-1033, Sandia National Laboratories.

[Cabrera et al., 2001] Cabrera, J. et al. (2001). Proactive detection of distributed denial of service attacks using MIB traffic variables - a feasibility study. In *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management*, pages 14–18, Seattle, WA.

[Canavan, 2001] Canavan, J. E. (2001). *Fundamentals of Network Security.* Artech House, Inc. ISBN: 1-58053-176-8.

[Capability, 2007] Capability, C. I. A. (2007). Network intrusion detector overview. Available from http://ciac.llnl.gov/cstc/nid/intro.html.

[Capitani et al., 2002] Capitani, S., Paraboschi, S., and Samarati, P. (2002). Access control: Principles and solutions. *Software Practice and Experience.* John Wiley & Sons.

[Carracedo, 2004] Carracedo, J. (2004). *Seguridad en redes telemáticas.* Mc Graw Hill. ISBN: 84-481-4157-1.

[Carracedo and J.D. Carracedo, 2002] Carracedo, J. and J.D. Carracedo (2002). Tarjetas de crédito anónimas para garantizar la privacidad de los ciudadanos. Morelia (México). I Congreso Iberoamericano de Seguridad Informática.

[CC, 2004] CC (2004). *Common criteria for information technology security evaluation. Part 1: Introduction and general model.* Version 2.2 Revision 256.

[Center, 1996] Center, C. C. (1996). CERT advisory CA 1996-21, TCP SYN flooding and IP spoofing attacks. September 1996; revised November 2000, http://www.cert.org/advisories/CA-1996-21.html.

[Center, 1998a] Center, C. C. (1998a). CERT summary CS 1998-02, SPECIAL EDITION – denial of service attacks targeting windows 95/NT machines. http://www.cert.org/summaries/CS-98.02.html.

[Center, 1998b] Center, C. C. (1998b). Smurf attack. http://www.cert.org/advisories/CA-1998-01.html.

[Center, 2000] Center, C. C. (2000). DoS using nameservers. http://www.cert.org/incident_notes/IN-2000-04.html.

[Center, 2007] Center, G. I. A. (2007). Special notice - egress filtering. http://www.sans.org/y2k/egress.htm.

[CERT Coordination Center, 1996] CERT Coordination Center (1996). TCP SYN flooding and IP spoofing attacks. http://www.cert.org/advisories/CA-1996-21.html.

[CERT Coordination Center, 1999] CERT Coordination Center (1999). CERT advisory CA 1999-17, denial of service tools. http://www.cert.org/advisories/CA-1999-17.html.

[CERT Coordination Center, 2000] CERT Coordination Center (2000). CERT incident note IN 2000-05, Mstream distributed denial of service tool. http://www.cert.org/incident_notes/IN-2000-05.html.

[CERT Coordination Center, 2001a] CERT Coordination Center (2001a). Code red. http://www.cert.org/incident_notes/IN-2001-08.html.

[CERT Coordination Center, 2001b] CERT Coordination Center (2001b). Code red II. http://www.cert.org/incident_notes/IN-2001-09.html.

[CERT Coordination Center, 2001c] CERT Coordination Center (2001c). erkms and li0n worms. http://www.cert.org/incident_notes/IN-2001-03.html.

[CERT Coordination Center, 2001d] CERT Coordination Center (2001d). Nimda worm. http://www.cert.org/advisories/CA-2001-26.html.

[CERT Coordination Center, 2001e] CERT Coordination Center (2001e). Ramen worm. http://www.cert.org/incident_notes/IN-2001-01.html.

[CERT Coordination Center, 2001f] CERT Coordination Center, C. M. S. E. I. (2001f). CERT advisory CA-2001-20 continuing threats to home users. http://www.cert.org/advisories/CA-2001-20.html.

[Cheswick et al., 2003] Cheswick, W., Bellovin, S., and Rubin, A. (2003). *Firewalls and Internet Security: Repelling the Willy Hacker*. Addison Wesley. ISBN: 020163466X.

[Cisco, 2007a] Cisco (2007a). NetRanger overview. Available at http://www.cisco.com/univercd/cc/td/doc/product/iaabu/csids/csids1/csidsug/overview.htm.

[Cisco, 2007b] Cisco (2007b). *Strategies to protect against distributed denial of service attacks*. http://www.cisco.com/warp/public/707/newsflash.html.

[Corp., 2000] Corp., B. (2000). The naptha DoS vulnerability. http://www.bindview.com/Support/RAZOR/Advisories/2000/adv_ NAPTHA.cfm.

[Daugman, 1998] Daugman, J. (1998). Recognizing persons by their iris patterns. In *Biometrics: Personal Identification in Networked Society*, pages 103 – 121, Kluwer.

[Dean et al., 2001] Dean, D., Franklin, M., and Stubblefield, A. (2001). An algebraic approach to IP traceback. In *Proceedings of the 2001 Network and Distributed System Security Symposium*.

[Denning, 1987] Denning, E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232.

[Dietrich et al., 2000] Dietrich, S., Long, N., and Dittrich, D. (2000). Analyzing distributed denial of service tools: The shaft case. In *Proceedings of 14th USENIX Systems Administration Conference (LISA 2000)*, pages 329–339.

[Dittrich, 1999a] Dittrich, D. (1999a). The DoS project's trinoo distributed denial of service attack tool. http://staff.washington.edu/dittrich/misc/trinoo.analysis.txt.

[Dittrich, 1999b] Dittrich, D. (1999b). The stacheldraht distributed denial of service attack tool. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt.

[Dittrich, 1999c] Dittrich, D. (1999c). The tribe flood network distributed denial of service attack tool. http://staff.washington.edu/dittrich/misc/tfn.analysis.txt.

[Dittrich et al., 2000] Dittrich, D., Weaver, G., Dietrich, S., and Long, N. (2000). The mstream distributed denial of service attack tool. http://staff.washington.edu/dittrich/mis/mstream.analysis.txt.

[Eichin and Rochlis, 1989] Eichin, M. W. and Rochlis, J. A. (1989). With microscope and tweezers: An analysis of the internet virus of november 1988. In *IEEE Symposium on Research in Security and Privacy*.

[Elteto and Molnar, 1999] Elteto, T. and Molnar, S. (1999). On the distribution of round-trip delays in TCP/IP networks. In *Proceedings of the 24th Annual IEEE Conference on Local Computer Networks*, page 172.

[Eric Guerrino and Kapito, 1997] Eric Guerrino, M. K. and Kapito, E. (1997). *User authentication and encryption overview*.

[Estan and Varghese, 2002] Estan, C. and Varghese, G. (2002). New directions in traffic measurement and accounting. In *Proceedings of ACM SIGCOMM'02*, Pittsburgh, PA.

[Ethereal, 2007] Ethereal (2007). Ethereal, the network protocol analyzer. Available from: http://www.ethereal.com.

[Exp, 2006] (2006). *The exponential distribution*. http://mathworld.wolfram.com/ExponentialDistribution.html. Revisited on May 2006.

[Fall and Varadhan, 2007] Fall, K. and Varadhan, K. (2007). *The ns manual*. Available at http://www.isi.edu/nsnam/ns/.

[Feldmeirer and Karn, 1990] Feldmeirer, D. C. and Karn, P. R. (1990). Unix password security - ten years later. In *G. Brassard, editor, CRYPTO89*, Springer-Verlag, pages 44 – 63. Lecture Notes in Computer Science.

[Ferguson and Senie, 2001] Ferguson, P. and Senie, D. (2001). Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing. RFC 2827.

[Fielding et al., 1997] Fielding, R., Irvine, U., Gettys, J., Mogul, J., Frystyk, H., and Berners-Lee, T. (1997). Hypertext transfer protocol - HTTP/1.1. RFC2068. Network Working Group.

[Fraser, 1997] Fraser, B. E. (1997). Site security handbook. RFC 2196.

[Garg and L. Narasimha Reddy, 2001] Garg, A. and L. Narasimha Reddy, A. (2001). Mitigating denial of service attacks using QoS regulation. Technical Report TAMU-ECE-2001-06, Texas A & M University.

[Gil and Poleto, 2001] Gil, T. M. and Poleto, M. (2001). MULTOPS: a data-structure for bandwidth attack detection. In *Proceedings of 10th USENIX Security Symposium*.

[Gobioff et al., 1996] Gobioff, H., Smith, S., Tygar, J., and Yee, B. (1996). Smart cards in hostile environments. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, The USENIX Association.

[Goncalves, 1997] Goncalves, M. (1997). *Firewalls complete*. Mc GrawHill.

[Greene, 2002] Greene, B. (2002). BGPv4 security risk assessment. CISCO white paper. http://www.cymru.com/Documents/barry2.pdf.

[Gregory, 1999] Gregory, P. H. (1999). *Solaris Security*. Prentice Hall and Sun Microsystems Press, 1st edition.

[Guillou et al., 1992] Guillou, L. C., Ugon, M., and Quisquater, J.-J. (1992). The smart card — a standardized security device dedicated to public cryptology. *Contemporary Cryptology — The Science of Information Integrity*, pages 561 – 614. IEEE Press.

[Hancock, 2000] Hancock, B. (2000). Trinity v3, a DDoS tool, hits the streets. *Computers Security*, 19(7):574.

[Houle et al., 2001] Houle, K., Weaver, G., Long, N., and Thomas, R. (2001). Trends in denial of service attack technology. Technical report, CERT Coordination Center. http://www.cert.org/archive/pdf/DoS_trends.pdf.

[Huang and Pullen, 2001] Huang, Y. and Pullen, J. (2001). Countering denial of service attacks using congestion triggered packet sampling and filtering. In *Proceedings of the 10th International Conference on Computer Communications and Networks*.

[Huopio, 1998] Huopio, S. (1998). Biometric identification. In *Seminar on Network Security: Authorization and Access Control in Open Network Environment*.

[Ierace et al., 2005] Ierace, N., Urrutia, C., and Bassett, R. (2005). Intrusion prevention systems. *Ubiquity*, 6(19):2–2.

[Ioannidis and Bellovin, 2002] Ioannidis, J. and Bellovin, S. (2002). Implementing push-back: router defense against DDoS attacks. In *Network and Distributed System Security Symposium NDSS02*, pages 6 – 8, San Diego, CA.

[ISO, 1988] ISO (1988). ISO 7498, OSI reference model, part 2, security architecture.

[Jajodia et al., 1995] Jajodia, S., Gadia, S., and Bhargava, G. (1995). Logical design of audit information in relational databases. *Information Security: An Integrated Collection of Essays*, pages 585 – 595. IEEE Computer Society Press.

[Jou, 2000] Jou, F. (2000). Design and implementation of a scalable intrusion detection system for the protection of network infrastructure. In *Proceedings of the 1st DARPA Information Survability Conference and Exposition (DISCEX 2000)*, volume 2, pages 69 – 83. http://projects.anr.mcnc.org/JiNao/jouy_reviewed.ps.

[Jung et al., 2002] Jung, J., Krishnamurthy, B., and Rabinovich, M. (2002). Flash crowds and denial of service attacks: Characterization and implications for CDNS and web sites. In *Proceedings of the 11th International World Wide Web Conference*, pages 293 – 304.

[Kargl et al., 2001] Kargl, F., Maier, J., and Weber, M. (2001). Protecting web servers from distributed denial of service attacks. In *Proceedings of the Tenth International Conference on World Wide Web*, pages 514 – 524, Hong Kong.

[Kent, 1997] Kent, S. (1997). Encryption-based protection for interactive user/computer communication. In *Proceedings of the 5th Data Communications Symposium.*

[Keogh, 2002] Keogh, E. (2002). Exact indexing of dynamic time warping. In *Proc. of the 28th VLDB Conference*, China.

[Klein, 1990] Klein, D. V. (1990). Foiling the cracker: A survey of, and improvements to, password security. In *Unix Security Workshop*, pages 5 – 14, The USENIX Association.

[Kleinrock, 1961] Kleinrock, L. (1961). Information flow in large communications nets. RLE quarterly progress reports.

[Kuzmanovic and Knightly, 2003] Kuzmanovic, A. and Knightly, E. (2003). Low-rate TCP-targeted denial of service attacks (The shrew vs. the mice and elephants). In *Proc. ACM SIGCOMM'03*, pages 75–86.

[Lamport, 1981] Lamport, L. (1981). Password authentication with insecure communication. *Communications of the ACM*, 24(11):770 – 772.

[Landwehr, 2001] Landwehr, C. (2001). Computer security. *International Journal on Information Security*, 1(1):3–13.

[Lau et al., 2000] Lau, F., Rubin, S. H., Smith, M. H., and Trajkovic, L. (2000). Distributed denial of service attacks. In *Proceedings of 2000 IEEE International Conference on Systems, Man, and Cybernetics.*

[Laurie and Clayton, 2004] Laurie, B. and Clayton, R. (2004). Proof-of-work proves not to work. In *Proceedings of the 3rd Annual Workshop on Economics and Information Security (WEIS 2004).* http://www.cl.cam.ac.uk/˜ rnc1/proofwork.pdf.

[Lee and Stolfo, 1998] Lee, W. and Stolfo, S. (1998). Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, pages 79 – 93, San Antonio, TX.

[Lee et al., 1999] Lee, W., Stolfo, S., and Mok, K. (1999). A data mining framework for building intrusion detection models. In *Proceedings of the1999 IEEE Symposium on Security and Privacy*, pages 120 – 132, Oakland, CA.

[Leiwo et al., 2000] Leiwo, J., P. Nikander, and Aura, T. (2000). Towards network denial of service resistant protocols. In *Proceedings of the 15th International Information Security Conference (IFIP/SEC 2000).*

[Lindqvist, 1999] Lindqvist, U. (1999). On the fundamentals of analysis and detection of computer misuse. PhD dissertation, Department of Computer Engineering, Chalmers University of Technology, Göteborg, Sweden.

[Liu et al., 2001] Liu, Z., Niclausse, N., and Jalpa-Villanueva, C. (2001). Traffic model and performance evaluation of web servers. *Performance Evaluation*, 46(2-3):77–100.

[Lo and Loon, 2004] Lo, J. and Loon, R. V. (2004). An IRC tutorial. http://www.irchelp.org/irchelp/irctutorial.html.

[Mahajan et al., 2002] Mahajan, R., Bellovin, S., Floyd, S., Ioannidis, J., Paxson, V., and Shenker, S. (2002). Controlling high bandwidth aggregates in the network. In *ACM SIGCOMM Computer Communications Review*, volume 32 of *3*, pages 62 – 73.

[Martin, 1993] Martin, R. R. (1993). *Basic Traffic Analysis*. Prentice-Hall Inc.

[McAfee, 2007] McAfee (2007). *VirusScan Online*. http://www.mcafee.com/myapps/vso/default.asp.

[McMordie, 1997] McMordie, D. (1997). Texture analysis of the human iris for high security authentication. Technical Report Image Processing 304 - 529, Department of Electrical Engineering, McGill University.

[Meadows, 1999] Meadows, C. (1999). A formal framework and evaluation method for network denial of service. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*.

[Menezes et al., 1996] Menezes, A., Oorschot, P. V., and Vanstone, S. (1996). *Handbook of Applied Cryptography*. CRC Press. ISBN: 0849385237.

[Mimestar.com, 2007] Mimestar.com (2007). Securenet pro feature list. Available from http://www.mimestar.com/products.

[Mirkovic et al., 2004] Mirkovic, J., Dietrich, S., Dittrich, D., and Reiher, P. (2004). *Internet Denial of Service. Attack and Defense Mechanisms*. Prentice Hall. ISBN: 0-13-147573-8.

[Mirkovic et al., 2002] Mirkovic, J., Prier, G., and Reiher, P. (2002). Attacking DDoS at the source. In *Proc. of ICNP 2002*, pages 312–321.

[Mirkovic and Reiher, 2004] Mirkovic, J. and Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53.

[Mittal and Vigna, 2002] Mittal, V. and Vigna, G. (2002). Sensor-based intrusion detection for intra-domain distance-vector routing. In ACM Press, editor, *Proceedings of the 9th ACM Conference on Computer and Communication Security*, pages 127 – 137. http://www.cs.ucsb.edu/vigna/pub/2002_mittal_vigna_ccs02.pdf.

[Mogul, 1995] Mogul, J. C. (1995). The case for persistent-connection HTTP. *SIGCOMM Comput. Commun. Rev.*, 25(4):299–313.

[Moore, 2007] Moore, D. (2007). The spread of the code red worm (crv2). http://www.caida.org/analysis/security/code-red/coderedv2_analysis.xml.

[Moore et al., 2001] Moore, D., Voelker, G., and Savage, S. (2001). Inferring internet denial-of-service activity. In *Proceedings of the 10th USENIX Security Symposium*, pages 9–22.

[NFR Security, 2007] NFR Security (2007). NFR network intrusion detection. Available from http://www.nfr.com.

[O'Brien, 2007] O'Brien, E. (2007). *NetBouncer: A practical cliente-legitimacy-based DDoS defense via ingress filtering.* http://www.nai/com/research/nailabs/development-solutions/netbouncer.asp.

[Park and Lee, 2001] Park, K. and Lee, H. (2001). On the effectiveness of route-based packet filtering for distributed DoS attack prevention in powerlaw internets. In ACM Press, editor, *Proceedings of the ACM SIGCOMM01 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications*, pages 15 – 26, New York.

[Paxson, 2001] Paxson, V. (2001). An analysis of using reflectors for distributed denial of service attacks. *ACM Computer Communication*, 31(3):38–47.

[Peña, 2001] Peña, D. (2001). *Fundamentos de Estadística.* Alianza. Pag. 183, ISBN: 84-206-8696-4.

[Peng et al., 2003] Peng, T., Leckie, C., and Ramamohanarao, K. (2003). Protection from distributed denial of service attack using history-based IP filtering. In *Proceedings of IEEE International Conference on Communications (ICC 2003)*, Anchorage, AL, USA.

[Postel, 1980] Postel, J. (1980). User datagram protocol. RFC 768. http://www.ietf.org/rfc/rfc0768.txt.

[Postel, 1981] Postel, J. (1981). Transmission control protocol. http://www.ietf.org/rfc/rfc0793.txt.

[Provos, 2003] Provos, N. (2003). Improving host security with system call policies. In *Proceedings of the 12th USENIX Security Symposium*, pages 257 – 272.

[Ptacek and Newsham, 1998] Ptacek, T. and Newsham, T. (1998). Insertion, evasion and denial of service: Eluding network intrusion detection. Technical report, Secure Networks, Inc.

[Rivest, 1990] Rivest, R. (1990). The MD4 message digest algorithm. *Crypto'90 Abstracts*, pages 281 – 291.

[Rivest, 1992] Rivest, R. (1992). The MD5 message digest algorithm. Internet Request for Comments RFC 1321.

[R.Mahajan et al., 2001] R.Mahajan, S.Floyd, and D.Wetherall (2001). Controlling high-bandwidth flows at the congested router. In *Proceedings of IEEE ICNP'01*, Riverside, CA.

[Sandhu, 1993] Sandhu, R. (1993). Lattice-based access control models. *IEEE Computer*, 26(11).

[Sandhu et al., 1996] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C. (1996). Role-based access control models. *IEEE Computer*, 29(2).

[Sandhu and Samarati, 1994] Sandhu, R. and Samarati, P. (1994). Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40 – 48.

[Sandhu and Samarati, 1997] Sandhu, R. and Samarati, P. (1997). Authentication, access control and intrusion detection. In *The Computer Science and Engineering Handbook*. CRC Press.

[SANS Institute, 2001] SANS Institute (2001). *NAPTHA: A new type of Denial of Service Attack*.

[Savage et al., 2000] Savage, S., Wetherall, D., Karlin, A., and Anderson, T. (2000). Practical network support for IP traceback. In *Proceedings of 2000 ACM SIGCOMM Conference*.

[Scambray et al., 2000] Scambray, J., McClure, S., and Kurtz, G. (2000). *Hacking Exposed: Network Security Secrets & Solutions*. McGraw-Hill/Osborne Media, 2nd edition. ISBN: 0072127481.

[Scheneier, 1995] Scheneier, B. (1995). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 2nd edition. ISBN: 0471117099.

[Schuba et al., 1997] Schuba, C., Krsul, I., Kuhn, M., Spafford, G., Sundaram, A., and Zamboni, D. (1997). Analysis of a denial of service attack on TCP. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 208 – 223.

[Shevtekar et al., 2005] Shevtekar, A., Anantharam, K., and Ansari, N. (2005). Low rate TCP denial-of-service attack detection at edge routers. *IEEE Communications Letters*, 9:363–365.

[Siris and Papagalou, 2006] Siris, V. A. and Papagalou, F. (2006). Application of anomaly detection algorithms for detecting SYN flooding attacks. *Computer Communications*, 29(9):1433 – 1442.

[Snoeren et al., 2001] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T., and Strayer, W. T. (2001). Hash-based IP traceback. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA.

[Song and Perrig, 2001] Song, D. X. and Perrig, A. (2001). Advanced and authenticated marking schemes for IP traceback. In *IEEE Infocom*.

[Song, 2004] Song, T. (2004). *Fundamentals of Probability and Statistics for Engineers*. John Wiley & Sons. ISBN: 0-470-86813-9.

[SourceFire, 2007] SourceFire (2007). SNORT: Sistema de detección de intrusiones basado en frmas. http://www.snort.org.

[Spafford, 1988] Spafford, E. H. (1988). The internet worm program: An analysis. Purdue Technical Report CSD-TR-823, Department of Computer Sciences, Purdue University, West Lafayette, IN, USA.

[Spafford, 1991] Spafford, E. H. (1991). The internet worm incident. Purdue Technical Report CSD-TR-933, Department of Computer Sciences, Purdue University, West Lafayette, IN, USA.

[Spatscheck and Peterson, 1999] Spatscheck, O. and Peterson, L. (1999). Defending against denial-of service requests in scout. In *Proceedings of the 1999 USENIX/ACM Symposium on Operating System Design and Implementation*.

[Stallings, 2003a] Stallings, W. (2003a). *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 3rd edition. ISBN: 0-13-091429-0.

[Stallings, 2003b] Stallings, W. (2003b). *Network Security Essentials*. Prentice Hall, 2nd edition. ISBN: 0-13-035128-8.

[Stevens et al., 2004] Stevens, W. R., Fenner, B., and Rudoff, A. M. (2004). *UNIX Network Programming, the sockets networking API*, volume 1. Addison-Wesley Proffessional, 3 edition. ISBN: 0-13-141155-1.

[Strassberg et al., 2003] Strassberg, K., Gondek, R., and Rollie, G. (2003). *Firewalls : manual de referencia*. McGraw Hill, Madrid. ISBN: 844813995X.

[Sun et al., 2004] Sun, H., Lui, J., and Yau, D. (2004). Defending against low-rate TCP attacks: Dynamic detection and protection. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP04)*, pages 196 – 205.

[Systems, 2007] Systems, I. S. (2007). Intrusion detection security products. Available from http://www.iss.net/securing_e-business/security_products/intrusion_detection/index.php.

[Talpade et al., 1999] Talpade, R. R., Kim, G., and Khurana, S. (1999). NOMAD: Traffic-based network monitoring framework for anomaly detection. In *Proc. of IEEE Symposium on Computers and Communications*, pages 442–451.

[Tripwire, 2007] Tripwire (2007). *Tripwire for servers*. Available at: http://www.tripwire.com/products/servers.

[Weaver, 2007] Weaver, N. (2007). Warhol worm. http://www.cs.berkeley.edu/˜ nweaver/warhol.html.

[Weiler, 2002] Weiler, N. (2002). Honeypots for distributed denial of service. In *Proceedings of the Eleventh IEEE International Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises 2002*, pages 109–114, Pitsburgh, PA, USA.

[X.Geng and A.B.Whinston, 2000] X.Geng and A.B.Whinston (2000). Defeating distributed denial of service attacks. *IEEE IT Professional*, 2(4):36–42.

[Yang et al., 2004] Yang, G., Gerla, M., and Sanadidi, M. Y. (2004). Defense against low-rate TCP-targeted denial-of-service attacks. In *Proc. IEEE Symposium on Computers and Communications (ISCC'04)*, pages 345 – 350, Alexandria, Egypt.

[Zhao et al., 2000] Zhao, W., Olshefski, D., and Schulzrinne, H. (2000). Internet quality of service: an overview. Columbia Technical Report CUCS-003-00.

[Zhao et al., 2002] Zhao, X., Pei, D., Wang, L., Massey, D., Mankin, A., Wu, S., and Zhang, L. (2002). Detection of invalid routing announcement in the internet. In *Proceedings of International Conference on Dependable Systems and Networks (DSN 2002)*, pages 59 – 68.

[Zheng and Leiwo, 1997] Zheng, Y. L. and Leiwo, J. (1997). A method to implement a denial of service protection base. In *Information Security and Privacy*, volume 1270, pages 90–101. Lecture Notes in Computer Science.