


Multivariate Statistical Network Monitoring–Sensor: An effective tool for real-time monitoring and anomaly detection in complex networks and systems

International Journal of Distributed
Sensor Networks
2020, Vol. 16(5)
© The Author(s) 2020
DOI: 10.1177/1550147720921309
journals.sagepub.com/home/dsn


Roberto Magán-Carrión^{1,2} , José Camacho²,
Gabriel Maciá-Fernández² and Ángel Ruíz-Zafra¹

Abstract

Technology evolves quickly. Low-cost and ready-to-connect devices are designed to provide new services and applications. Smart grids or smart health care systems are some examples of these applications. In this totally connected scenario, some security issues arise due to the large number of devices and communications. In this way, new solutions for monitoring and detecting security events are needed to address new challenges brought about by this scenario, among others, the real-time requirement allowing quick security event detection and, consequently, quick response to attacks. In this sense, Intrusion Detection Systems are widely used though their evaluation often relies on the use of predefined network datasets that limit their application in real environments. In this work, a real-time and ready-to-use tool for monitoring and detecting security events is introduced. The Multivariate Statistical Network Monitoring–Sensor is based on the Multivariate Statistical Network Monitoring methodology and provides an alternative way for evaluating Multivariate Statistical Network Monitoring–based Intrusion Detection System solutions. Experimental results based on the detection of well-known attacks in hierarchical network systems prove the suitability of this tool for complex scenarios, such as those found in smart cities or Internet of Things ecosystems.

Keywords

Multivariate Statistical Network Monitoring, sensor, monitoring, anomaly detection, Intrusion Detection System, security, communication networks, Internet of Things, smart cities

Date received: 10 October 2019; accepted: 2 April 2020

Handling Editor: Deze Zeng

Introduction

Several technical reports forecast 30 billion IoT (Internet of Things) devices around the world by 2021 and more than 14 billion M2M (Machine to Machine) connections by 2023.^{1,2} This scenario enables new services and applications for improving people's daily life as well as new business opportunities. However, many challenges arise too, with security being one of the most important.

¹Department of Computer Engineering, School of Engineering, University of Cádiz, Cádiz, Spain

²Network Engineering & Security Group, University of Granada, Granada, Spain

Corresponding author:

Roberto Magán-Carrión, Department of Computer Engineering, School of Engineering, University of Cádiz, 10, Puerto Real, 11519 Cádiz, Spain.
Email: roberto.magan@uca.es; rmagan@ugr.es



Underlying systems and communications networks are continually being threatened by attackers, especially in these hiper-connected environments. For instance, it is worth mentioning two recent attacks, Dyn^{3,4} and VPNFilter,⁵ where thousands of IoT devices were compromised causing, on one hand, a high economic impact and, on the other hand, and even worse, personal costs.

Monitoring and controlling what is happening in these kinds of scenarios is a great challenge since the attack exposure surface grows with the number of devices interconnected. A more challenging issue is to manage the generated data gathered from different information sources (ISs) such as applications, networking devices, and communications. In this way, key aspects such as managing the volume, veracity, or velocity of the data are crucial for achieving quick and efficient detection and reaction against security attacks.⁶ Furthermore, these aspects may limit the practical application of the solutions, especially in the described scenario.

In this sense, IDS (Intrusion Detection System) tools are widely used by the specialized community on ICT (Information and Communication Technologies) security. IDS systems are commonly categorized as one of two types: (1) Network-based IDSs (NIDSs) and (2) Host-based IDSs (HIDSs). NIDSs monitor network events such as traffic flows or firewall logs, while HIDSs behave similarly but consider host (endpoint)-related events, for example, syslog, CPU load, and so on.

To evaluate the performance of IDSs, predefined datasets are widely used. Consequently, choosing one or another is a very relevant decision with a relevant impact, not only on the detection performance but also on the confidence on the conclusions the authors claim. In this way, Maciá-Fernández et al.⁷ built a recent and real network dataset that copes with the main drawbacks found in the most commonly used network datasets.

Nevertheless, most of the IDS solutions tested through network datasets are not valid for its use in real application scenarios, because there are differences between the environment where they were built and the one where the IDS solution will be deployed. Apart from that, some other functional requirements must be accomplished by the IDS solutions to be afterward deployed in real network environment. They are, among others, to be efficient in managing heterogeneous and large amount of data, real-time requirements, and adaptability. In this manner, ready-to-use solutions for real-time monitoring and anomaly detection are recommended. These types of approaches could eliminate the need to use previously gathered datasets, which are, on the contrary, very difficult to build and time-consuming.

In this work, the Multivariate Statistical Network Monitoring (MSNM)–Sensor tool is introduced to address the previous issues. MSNM-Sensor is an efficient tool for real-time network monitoring and anomaly detection based on the MSNM approach coined by Camacho et al.⁸ The MSNM-Sensor is ready-to-use and validates the real application of MSNM-based IDS solutions: the MSNM ecosystem.^{8–13} Among others characteristics, the MSNM-Sensor is able to manage heterogeneous data sources in real-time; reduces the monitoring network traffic without significant impact on the detection performance; is lightweight, scalable, versatile, and dynamically adaptable to changes in the network environments; keeps privacy on communications; provides a friendly interactive dashboard; and is an open source project.

In order to demonstrate its suitability to be used in complex network environments for real-time monitoring and anomaly detection, the MSNM-Sensor has been successfully tested in hierarchical networks and systems for detecting well-known attacks like *DoS*, *port scanning*, and *data exfiltration*. After that, these results are compared versus those obtained in the work Maciá-Fernández et al.¹⁰ where the authors test the detection capability of the MSNM approach in the same scenario but in offline mode. The obtained results validate the suitability of the tool to be deployed in real network scenarios with real-time monitoring and detection requirements.

The article has been organized as follows. The section “Fundamentals of MSNM” describes the fundamentals of the MSNM methodology, which supports the core functionality of the sensor. In section “MSNM-Sensor,” the components and operation modes of the MSNM-Sensor are described. The tool is validated in realistic network architectures through the deployment of several MSNM-Sensors for real-time monitoring and anomaly detection in the section “Application of MSNM-Sensor for monitoring and attack detection in complex network environments.” Attacks such as DoS, data exfiltration, or port scanning are successfully detected in the proposed network scenario. Moreover, the tool is also compared with a state-of-the-art solution. Finally, conclusions and further work are described in section “Conclusions and future work.”

Fundamentals of MSNM

In this section, we briefly introduce MSNM, which is the basis of the MSNM-Sensor tool. MSNM relies on Principal Components Analysis (PCA), a main tool for multivariate analysis. PCA has been established as a promising technology to perform network anomaly detection.^{14–17} The unsupervised nature of PCA allows

to unveil anomalous behavior from unknown attacks, which is a desired characteristic in those solutions working in real environments. Apart from its unsupervised nature, PCA is a white-box model. In comparison to the bulk of machine learning solutions, PCA models are explainable since trends or relationships in the features and observations of the data can be easily connected.

MSNM has been demonstrated to be a promising methodology in network anomaly detection through several works⁸⁻¹³ offering a high detection performance. Four stages comprise the MSNM methodology. They are (1) *parsing*, (2) *fusion*, (3) *detection*, and (4) *diagnosis*. All of them are introduced in the following and integrated in the MSNM-Sensor tool as can be seen in next section.

Parsing

Information from communication networks usually comes in the shape of huge logs and traffic-based files containing heterogeneous information. This makes it impossible to directly use this information as input of detection systems and learning models to identify different kinds of attacks. However, to overcome this issue, data sources are processed in order to build a more suitable input for automatic detectors or classifiers.

In this sense, the application of some feature engineering technique is proposed to build a well-structured input, suitable for monitoring and detection systems in general. Thus, the Feature as a Counter (FaaC)⁶ technique is used as a functional solution to the problem of learning from large heterogeneous datasets. It consists in transforming different data sources of information (structured or not) events in the new variables. The new ones are just counters of the original ones computed in a given interval of time. For instance, it could be interesting to count the number of accesses to port 22 in a given time window interval, because a high number might mean a brute force secure shell (SSH) attack. The practical implementation tool for the FaaC approach, named FCParse, is available online for downloading at Pérez-Villegas et al.¹⁸

Fusion

In communication networks and systems, it is expected to find more than one information data source to be monitored. Apart from the *parsing* functionality of FaaC, the FCParse tool is also able to *fuse* different data sources in a single set of features. For each different source of data, a set of features (counters) is defined. All the sources under monitoring are appended into a simple data stream to build the calibration matrix for the PCA model. For that, each source is

sampled with equal sampling rate, then parsed, and finally fused into a unique data stream. This procedure is periodically repeated at each sample time.

The combination of the *parsing* and the *fusion* procedures is specially suited for the subsequent multivariate analysis, resulting in high-dimensional feature vectors that need to be analyzed with dimension reduction techniques, like PCA. Moreover, the diagnosis procedure benefits from the definition of a large number of features for a better description of the anomaly taking place. Finally, counters and their correlation are easy to interpret.

Detection

PCA is the core of MSNM. PCA identifies a number of linear combinations of the original variables in a data set \mathbf{X} (contains the data matrix obtained in the previous two stages), the so-called PCs (Principal Components), containing most of its relevant information (variability). This process involves a change from the original variables in the \mathbf{X} space to those in the PC subspace. If \mathbf{X} contains M variables and N observations of each variable, PCA reduces its dimensions from M variables to A PCs by finding the A -dimensional latent subspace of the most variability captured.

PCA is described through the following equation

$$\mathbf{X} = \mathbf{T}_A \cdot \mathbf{P}_A^t + \mathbf{E}_A \quad (1)$$

where \mathbf{P}_A is the $M \times A$ loading matrix, \mathbf{T}_A is the $N \times A$ score matrix, and \mathbf{E}_A is the $N \times M$ residual matrix. The maximum variance directions are obtained from the eigenvectors of $\mathbf{X}^t \cdot \mathbf{X}$, and they are ordered as the columns of \mathbf{P}_A by the explained variance. The rows of \mathbf{T}_A are the projections of the original N observations in the new latent subspace. \mathbf{E}_A is the matrix that contains the residual error, and it plays a key role in the anomaly detection, as shown later. The projection (score) onto the PCA subspace of a new observation is obtained as follows

$$\mathbf{t}_{new} = \mathbf{x}_{new} \cdot \mathbf{P}_A \quad (2)$$

where \mathbf{x}_{new} is a $1 \times M$ vector that represents a new object and \mathbf{t}_{new} is a $1 \times A$ vector that represents the projection of the latent subspace, while

$$\mathbf{e}_{new} = \mathbf{x}_{new} - \mathbf{t}_{new} \cdot \mathbf{P}_A^t \quad (3)$$

corresponds to the residuals.

In order to detect anomalies in the monitored system, Q -st¹⁹ and D -st (also known as T^2)²⁰ statistics are commonly used. Q -st compresses the residuals in each observation, and D -st is computed from the scores.

Their values for a specific observation can be computed through the following equations

$$D_n = \sum_{a=1}^A \left(\frac{\tau_{an} - \mu_a}{\sigma_a} \right)^2 \quad (4)$$

$$Q_n = \sum_{m=1}^M (e_{nm})^2 \quad (5)$$

where τ_{an} represents the score of the n th observation of the a th PC; μ_a and σ_a stand for the mean and standard deviation for the scores of that PC in the calibration data, respectively; and e_{nm} represents the residual value corresponding to the n th observation of the m th variable.

With both statistics (4) and (5) computed from the calibration data \mathbf{X} , Upper Control Limits (UCLs), that is, detection thresholds, can be established with a certain confidence level.^{8,21} Subsequently, new data are monitored using these limits, and an anomaly is identified when UCL limits are exceeded by new incoming observations.

Diagnosis

After detecting something anomalous in the system, it is necessary to investigate its root causes. Most often, the diagnosis procedure is manually carried out by a security analyst alerted by the system.

The diagnosis procedure could be a tricky and tedious task, due to the large amount of information to analyze. Feature contributions to a given anomaly can be a useful tool to identify where the anomalous behavior comes from. Contribution plots or other diagnosis methods like oMEDA (observation-based Missing-data method for Exploratory Data Analysis)²² or Univariate Squared (US)²³ can be used to identify the feature contributions. Thus, anomalies are detected in the D -st and/or Q -st statistics, and then the diagnosis is performed with, for example, oMEDA. For instance, the output of oMEDA is a $1 \times M$ vector where each element contains the contribution of the corresponding feature to the anomaly under study. Those contributions with large magnitude, either positive or negative, are considered to be relevant.

MSNM-Sensor

In the following section, MSNM-Sensor modules and operations are introduced and thoroughly explained. Before that, an illustrative example of use in a hypothetical network scenario is introduced for a better understanding of how the tool works.

Example of use: an hierarchical approach

The MSNM-Sensor itself is able to monitor and detect anomalous behaviors from a wide range of heterogeneous data sources. However, the really novel idea behind the use of Q -st and D -st statistics is their capability of maintaining the monitoring and anomaly detection performance when they are included in the hierarchy of complex network environments. This useful characteristic has been demonstrated by Maciá-Fernández et al.¹⁰

Although the tool will be tested in the section “Application of MSNM-Sensor for monitoring and attack detection in complex network environments,” in a realistic network scenario, an example of several MSNM-Sensors cooperating within a hypothetical and common network deployment is described as follows. Figure 1 shows a simple network scenario where several MSNM-Sensors (orange boxes) are deployed at hosts and network devices. We can discern in the figure two involved information flows: the monitoring and diagnosis flows. The former (black dashed lines) transports pairs of monitoring statistics ($[Q_{(n,m)}, D_{(n,m)}]$), where n is the sensor’s ID and m is the network level in the hierarchy) coming from lower to higher levels in the hierarchy. In this synthetic example, sensors $S_{1,3}, \dots, S_{n,3}$ are deployed at hosts in the deepest architecture level, sending the generated statistics $[Q_{(1,3)}, D_{(1,3)}], \dots, [Q_{(n,3)}, D_{(n,3)}]$ to the next closest sensor in the hierarchy. Indeed, they act as remote sources of $S_{1,2}$. Now, this sensor aggregates and processes it, giving the $[Q_{(1,2)}, D_{(1,2)}]$ values. Finally, the root sensor ($S_{1,1}$) gathers the statistical information from its immediately lower levels, processes it, and generates the last statistics to be observed for anomaly detection. This final monitoring task is commonly carried out under the supervision of a security analyst, who determines the presence of an anomaly when certain control limits are exceeded by the statistics.⁸

Once the anomaly is detected, a deeper inspection should be done to determine, for example, where the anomaly comes from and its root causes. This is the diagnosis procedure, which is represented in Figure 1 with solid green and dashed brown lines. In this example, the anomaly comes from $S_{1,3}$, which is in charge of monitoring firewall traffic logs. First, a command message $[C_m]$ to find the anomaly origin is sent. The Diagnosis Routing Table (DRT) defines how this message should be routed across the multilevel scenario. To determine which data source motivates the anomaly at a certain timestamp, a diagnosis algorithm is invoked. In this work, the oMEDA algorithm is currently used, though some other methods could easily be integrated. This procedure is repeated until the data source origin is found. Shortly thereafter, the involved piece of

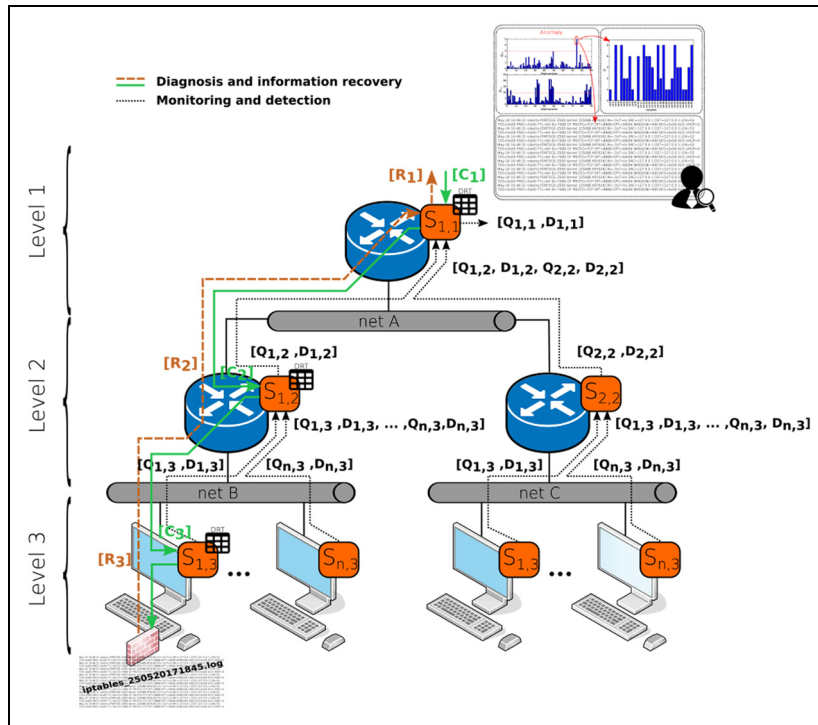


Figure 1. A hypothetical deployment of several MSNM-Sensors (orange boxes) throughout an interconnected system for hierarchical monitoring, anomaly detection (dashed black lines), and diagnosis (solid green and dashed brown lines).

information (raw) falling into the observation period of time is returned to the root sensor to be analyzed in detail. A new message called response $[R_m]$ allows this operation.

MSNM-Sensor modules

The MSNM-Sensor functional modules, depicted in Figure 2, are described as follows:

IS. The IS module handles the data coming from the ISs. Two types of data sources, according to their location, are considered:

- Local (LIS). The information gathered from these data sources is generated by the device where the MSNM-Sensor is deployed. For instance, local ISs can be obtained from firewall log files, NetFlow traffic flows, and host-based information (e.g. syslog in Linux-based systems), among others. LIS data are processed by the *parsing* and *fusion* modules.
- Remote (RIS). The incoming data from other MSNM-Sensors are handled as a remote IS. Most of the data from other MSNM-Sensors

will be the computed values of the monitoring statistics $Q_{n,m}$ -st and $D_{n,m}$ -st.

Parsing and fusion. As mentioned in section “Fundamentals of MSNM,” *parsing* and *fusion* modules do, on one hand, a transformation of LIS data sources (structures or not) into a new structured form where new features are counters of the original. On the other hand, in case of considering several LIS data sources, all of them are fused by appending one after another. As a result, we have a homogeneous data stream of quantitative values to be subsequently used in the *detection* module for anomaly detection. Both processes are carried out by the FCParse¹⁸ which implements the FaaC approach.

Detection. This module represents the sensor functional core for anomaly detection. It provides multivariate statistical-based methods and algorithms to compute the monitoring statistics $Q_{n,m}$ -st and $D_{n,m}$ -st in real time. This module enables the detection of anomalies in the monitored system when statistics computed from a new observation exceed certain control limits. Two control limits are calculated: UCLq and UCLd for $Q_{n,m}$ -st and $D_{n,m}$ -st, respectively. Detailed information

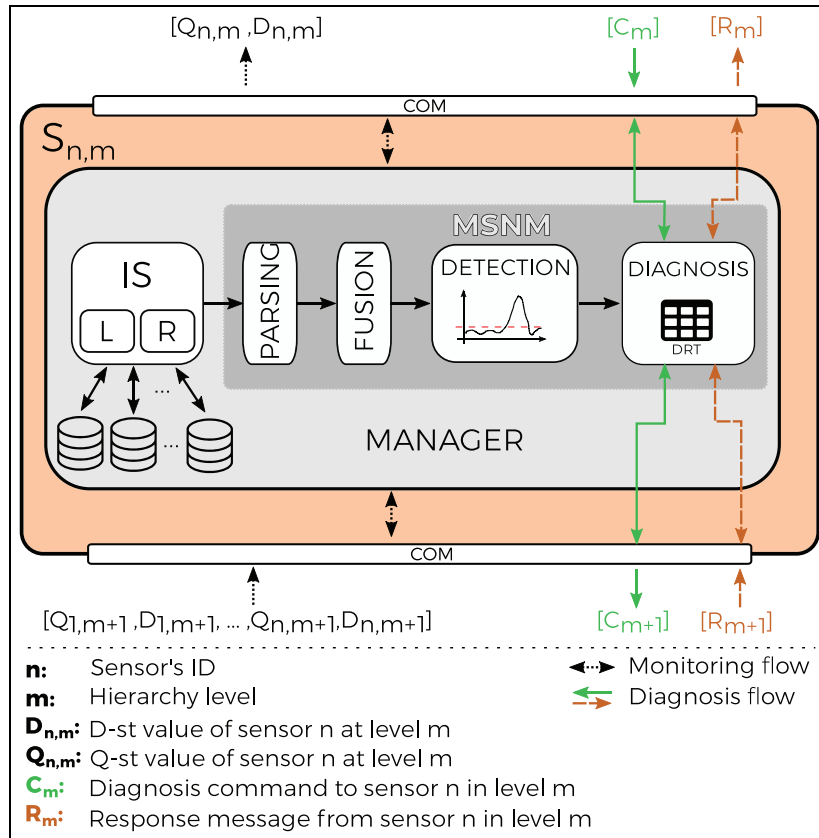


Figure 2. Functional modules of a generic sensor $S_{n,m}$. n corresponds to the sensor's ID, and m is the hierarchical level ID where the sensor n is deployed.

on monitoring statistics and UCL limits construction can be found in Camacho et al.⁸

The *detection* module supports three different functions:

- **Preprocessing.** The *detection* module performs data preprocessing for new observations and learning models. The standard normalization is used by default, but additional methods are also available and ready to use.
- **Modeling.** This operation is in charge of the sensor calibration from a set of observations that are, ideally, under Normal Operation Conditions (NOCs), that is, without known anomalies. Currently, the PCA model is used, but other machine-learning-based techniques can be easily integrated.^{24,25}
- **Monitoring.** This operation computes the above-mentioned statistics for each incoming observation. In addition to the precomputed control limits, the monitoring operation is able to detect anomalous behavior when these limits are exceeded.

Because of the dynamic nature of information coming from networks and systems, learning models should periodically be re-calibrated or re-trained in order to adapt them to normal changes in the environment and avoid high false positive (FP) rates. Usually, such changes are due to the normal cyclo-stationary nature of the information from network communications and systems,⁷ which make variables vary depending on the hour, day, week, and so on. These behavioral patterns are periodically repeated. The Exponentially Weighted Moving Average (EWMA) approach is used to dynamically calibrate the sensors every 60 min.²⁶ Each sample rate, new UCL limits are also computed.

Diagnosis. The diagnosis procedure takes place when an anomaly is detected on the monitored system. It is necessary to find out where the anomalous event came from and its root causes, to afterward deploy the adequate response measures to counteract the attack. How to manage this problem is the duty of the *diagnosis* module.

The *diagnosis* module relies on the use of statistical multivariate techniques to determine the source of the anomaly. Currently, the oMEDA²² method is

implemented, but other methods can be included, for instance, contribution plots or diagnosis methods like the one proposed in Fuentes-García et al.²³

Although the *diagnosis* module solves the anomaly diagnostic by itself, it is done locally, that is, in the device where the sensor was deployed. In addition, defining the device and data source(s) involved in an anomalous behavior in the whole system under monitoring is not a trivial task. For this reason, we create the DRT, which acts similarly to the well-known routing IP tables but adds together the information of local and remote sources. The diagnosis flow, diagnosis routing procedure, and the role of $[C_m]$ and $[R_m]$ messages were already introduced in the section “Example of use: a hierarchical approach” and will be explained in detail in section “Operations.”

COMmunications. The COMmunications (COM) module allows each MSNM-Sensor to exchange information. It handles the exchange of specific messages. The system supports (but is not limited to) two types of messages: data and command. The messages mainly differ in the payload and type. For instance, a data message can include any information required in sensor operations, for example, the monitoring statistics. On the contrary, command messages are devised to control these processes.

Depicted in Figure 2, two information flows are differentiated: monitoring and diagnosis. It is worth mentioning that only the first one (monitoring) is currently implemented, while the second one is an ongoing work (see the project in Magán-Carrión et al.²⁷). However, we decided to mention and describe both of them because they are complementary. In this way, monitoring information flow exchanges data messages containing the computed statistics $Q_{n,m}$ and $D_{n,m}$, while the diagnosis flow controls the entire diagnostic procedure.

In this early stage, there is no specific routing algorithm between sensors. Instead, each sensor must be manually configured to send and receive data to and from others. A self-deployed sensor process will be added in future releases.

Manager. As depicted in Figure 2, all modules work together following an execution pipeline and sharing the necessary information. The module in charge of orchestrating and managing the others is the *manager*.

As mentioned above, there are two different information flows: monitoring and diagnosis. The first one (monitoring) involves four main modules (IS, *parsing*, *fusion*, and *detection*) that should be invoked sequentially, because the output of each module is the input of the next one. However, the second flow (diagnosis) involves the *diagnosis* module, which is invoked if a specific message is received. Finally, *manager* handles the

orchestration of which MSNM-Sensor module should run and when.

MSNM-Sensor operations

Thus far, the main functional MSNM-Sensor modules have been described. However, high-level operations, including several modules, are devised in accordance with the principal MSNM-Sensor functionalities: monitoring and diagnosis. The diagnosis process is still an ongoing work; however, we decide to briefly describe it for completeness. Both operations are managed through the MSNM-Sensor dashboard which is described at the end of this subsection.

Monitoring operation. To be aware of what is happening in systems or networks, it is important to detect anomalous behaviors. However, this is not a trivial task, since a previous work must be done to select which element and information should be supervised. In this manner, the monitoring flow and the involved modules offer a versatile and scalable tool allowing the user to freely select data sources and variables to be monitored.

Figure 3 shows a detailed view of module interactions and the exchanged information. In the figure, we show a hypothetical local data source LIS_v with M_v variables to monitor a total of N_v gathered observations, the latter being split into k batches ($b_{v1}, b_{v2}, \dots, b_{vk}$). Each batch has $j = M_v$ variables and i different numbers of observations each. As a result, we are able to (1) process less information, reducing the computation time, which is a key point for real-time applications, and (2) adapt the monitoring time step granularity, sometimes hardly limited by the monitored data source or the anomaly to be detected. As explained in section “Application of MSNM-Sensor for monitoring and attack detection in complex network environments,” 60 s is enough to monitor *NetFlow*-based data sources in the detection of DoS attacks, for example.

For each b_{vk} batch of observations, a new one is generated by parsing and fusing the original (raw) information. This task is the duty of the *parsing* and *fusion* modules (see section “Parsing and fusion”). At the time of writing this article, the implemented module was the FCParse¹⁸ which implements the FaaC approach, a new feature engineering methodology that transforms the original information variable space into a new one where the new variables (z) are counters (C_v) of the original ones. This smart transformation makes the system highly versatile and scalable, allowing the user to define a large number of different new variables from a limited original set. For instance, counting the number of different destination ports in a certain period of time could be relevant for port scanning or port knocking attacks.

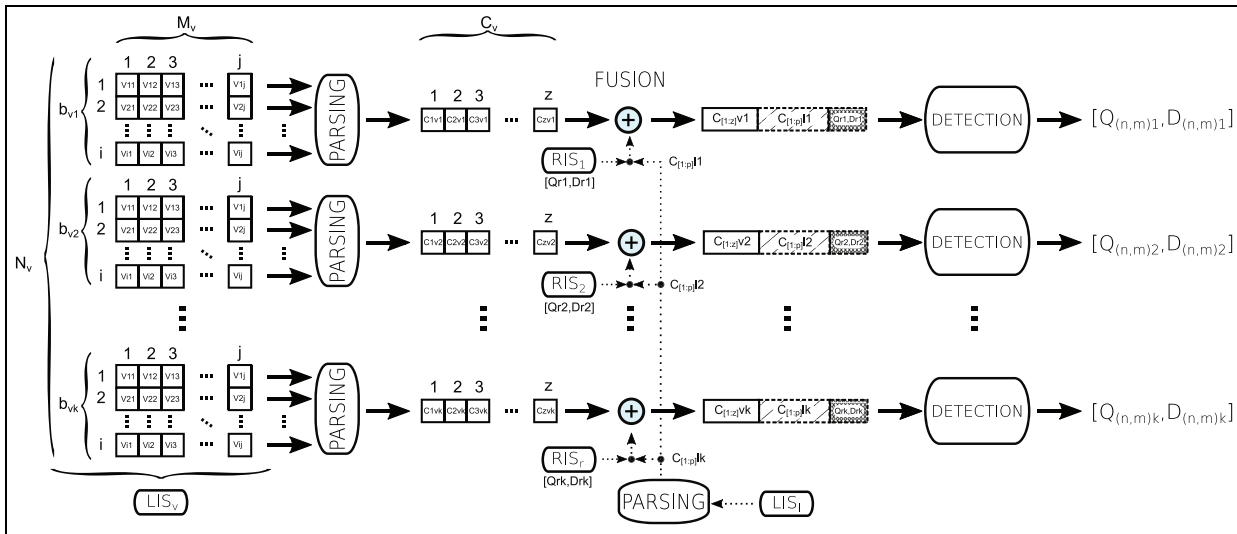


Figure 3. Involved modules and the information exchanged among them for a $S_{n,m}$ monitoring flow. The information comes from different data sources, namely, two local (LIS_v and LIS_i) and one remote (RIS_r), to be divided into k batches ($b_{v1}, b_{v2}, \dots, b_{vk}$). LIS sources (raw) are parsed for each batch. The new variables, counters of the original ones, are fused together with the remote ones. All variables together conform the observation to be monitored. After that, the $Q_{n,m}$ and $D_{n,m}$ statistics are computed by the MSNM module.

Although just one data source has been considered so far, additional local (LIS) or remote (RIS) data sources can also be included if needed. Figure 3 shows the procedure to add new data sources. In this case (but not limited to it), there are three data sources involved: two local (LIS_v and LIS_i) and one remote (RIS_r). At each monitoring step, a new fused observation is created. In this way, the extended space will have e variables, with $e = z + p + 2$, where z is the number of variables (counters) from a batch k of source LIS_v ; p is the number of variables (counters) from the same batch of source LIS_i ; and RIS_r has two variables as the number of statistics generated by the remote sensor. This observation is the input of the *detection* module which is in charge of computing the monitoring statistics ($Q_{(n,m)k}, D_{(n,m)k}$). At this point, the system can detect anomalous behaviors when the control limits are exceeded. In addition, if this sensor is not the root in the hierarchy, the generated statistics will also be sent to the corresponding remote sensor for hierarchical monitoring and anomaly detection.

Diagnosis operation. As aforementioned, after detecting an anomaly in the system, the diagnosis procedure should be launched to determine its origin. Figure 4 depicts the operation steps launched to determine which IS is involved in the anomaly. For instance, we suppose that a security event arose from the LIS_v local data source. This hypothetical data source represents an *iptables* firewall.

Once the corresponding observation to be inspected has been retrieved from the sensor, the oMEDA

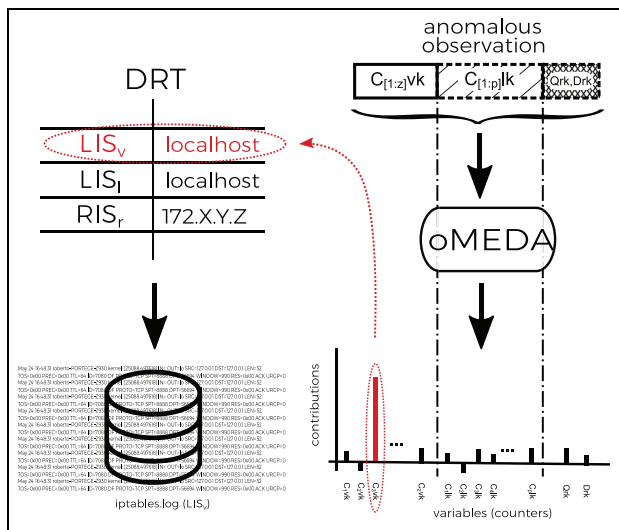


Figure 4. Diagnosis operation steps to determine the origin of an anomaly in the system. In this case, the anomaly comes from a variable ($C_{3}vk$) with a high contribution in the observation under inspection. This variable corresponds with a local data source LIS_v , which is a firewall log.

algorithm is launched. This algorithm outputs the contribution of each variable, where the higher the contribution, either positive or negative, the more relevant the variable was in the anomaly. In this case, the variable with a high positive value is $C_{3}vk$ (the one in red in Figure 4), which belongs to the LIS_v data source according to the DRT. The DRT structure and role in this operation are described in the next section.



Figure 5. Snapshot of the monitoring view of the MSNM-Sensor dashboard. The logical connections of the sensors (orange boxes) are shown in the upper section of the figure, while the monitoring graphs are in the bottom part.

Together, the timestamp and the diagnosis under investigation allow the sensor to extract the corresponding piece of raw information to be afterward analyzed by the security analyst Camacho et al.¹³

Management and control. The MSNM-Sensor application comes with an interactive dashboard. Thanks to this dashboard, the security analyst or the application operator can, in real-time, access the monitoring and diagnosis information and manage the sensor deployment. A snapshot of the monitoring main section of the dashboard is shown in Figure 5. The upper part shows the logical connections created among the sensors. The operator can see the direction of the monitoring flow as indicated by arrows. In this specific scenario, it is clearly shown how the sensors in routers R1, R2, and R3 are configured to send their monitoring information to the Border Router (BR) router. In addition, the monitoring graphs will appear in the bottom part by clicking on a specific sensor. Among other actions, the operator can interact with these graphs to pause/play the updating procedure for a better inspection. The dashboard is needed for whatever monitoring system

that allows the operator to reduce the response/reaction time when an attack takes place.

Application of MSNM-Sensor for monitoring and attack detection in complex network environments

Experimental environment

To validate the monitoring and anomaly detection performance of the MSNM-Sensor in complex systems, we deployed several sensors over a controlled scenario with virtual machines running in a cluster. This scenario has been previously devised in work of Maciá-Fernández et al.¹⁰ to theoretically prove the hypothesis of the application of MSNM for hierarchical systems. A performance comparison between the previous work and the MSNM-Sensor's deployment throughout the same network scenario, is provided in subsection "Comparative study."

The complete scenario with the different machines is depicted in Figure 6. This environment simulates a typical network architecture of a corporation so we can observe several subnetworks, network devices, and end

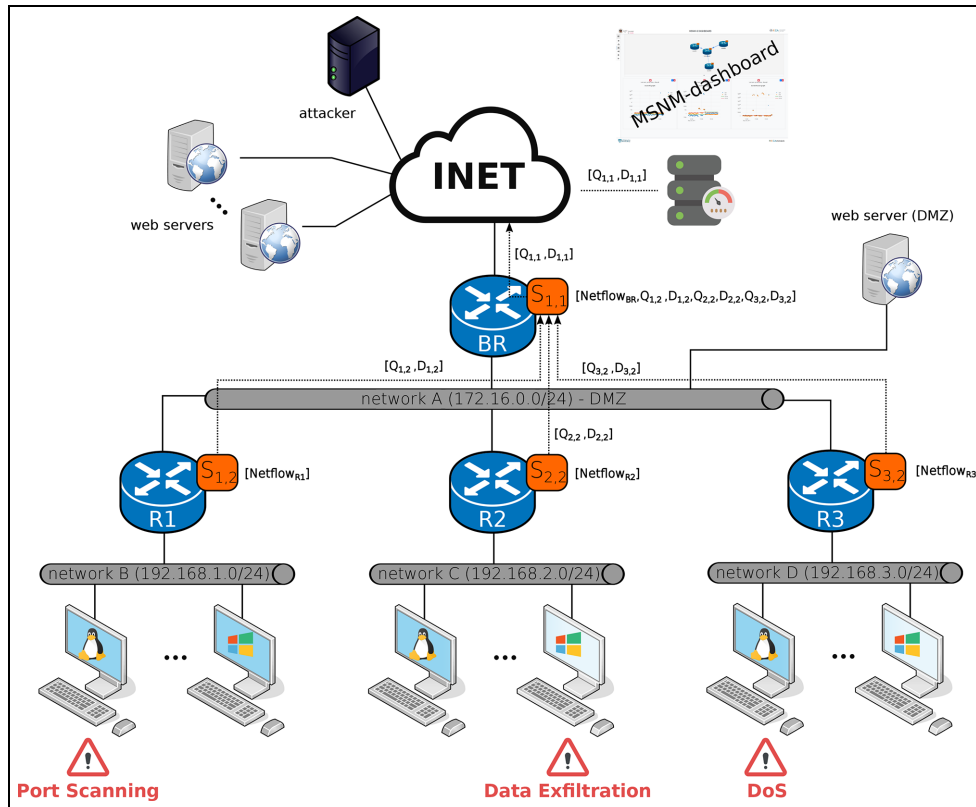


Figure 6. Experimental scenario. Four MSNM-Sensors ($S_{1,2}$, $S_{2,2}$, $S_{2,3}$, $S_{1,1}$) are deployed at each router. Each sensor monitors a NetFlow local data source generated from each router, while $S_{1,1}$ also aggregates all the statistics generated by the lower-level routers.

devices. A demilitarized zone (DMZ) is located in the inner network, separated from the outside world (Internet) with a BR and departmental networks in turn delimited by the corresponding routers (R1, R2, and R3).

In this scenario, we generated two types of network traffic: normal and malicious. Normal traffic comprises all HTTP communications from all departmental hosts requesting HTTP resources allocated at the several web servers placed in the Internet and DMZ. As shown in Figure 6, the incoming traffic from and outgoing traffic to the Internet goes through the BR. On the contrary, Rx routers, with $x = 1, 2, 3$, relay the corresponding portion of the previous HTTP traffic, which is generated by the hosts in their networks. In addition, departmental hosts request HTTP resources to the web server in the DMZ.

However, the malicious traffic is generated from different locations in the predefined architecture, simulating very well-known and state-of-the-art attacks. These are (1) DoS (high and low rate); (2) port scanning, a relevant step in the recognition phase in a penetration testing procedure; and (3) data exfiltration for privacy violation purposes.

We run our scenario during a period of 25 h. In the first 23.5 h, only normal traffic is generated. During

the last hour and a half, the attacks previously described are generated sequentially in 5-min intervals: high-rate DoS, low-rate DoS, scanning attack, and data exfiltration.

The different routers in the network (R1, R2, R3, and BR) are equipped with NetFlow inspectors that generate NetFlow v5 information. These data are afterward consumed in real time by the corresponding MSNM-Sensors, allocated in the mentioned network devices. These sensors are named $S_{1,2}$, $S_{2,2}$, $S_{2,3}$, and $S_{1,1}$, and represented by orange-colored boxes in Figure 6. Sensors in the inner routers R1 to R3 consider only a local data source: the generated information of the corresponding NetFlow inspectors. Sensor $S_{1,1}$ BR is in charge of aggregating the monitoring information in the form of statistics coming from the sensors in the network lower level. Every minute, a new observation is gathered by each sensor, meaning that two statistics are generated every minute.

Experimental results

As we stated in previous sections, a key characteristic of the MSNM-Sensor is its applicability for real-time monitoring and detection in complex network

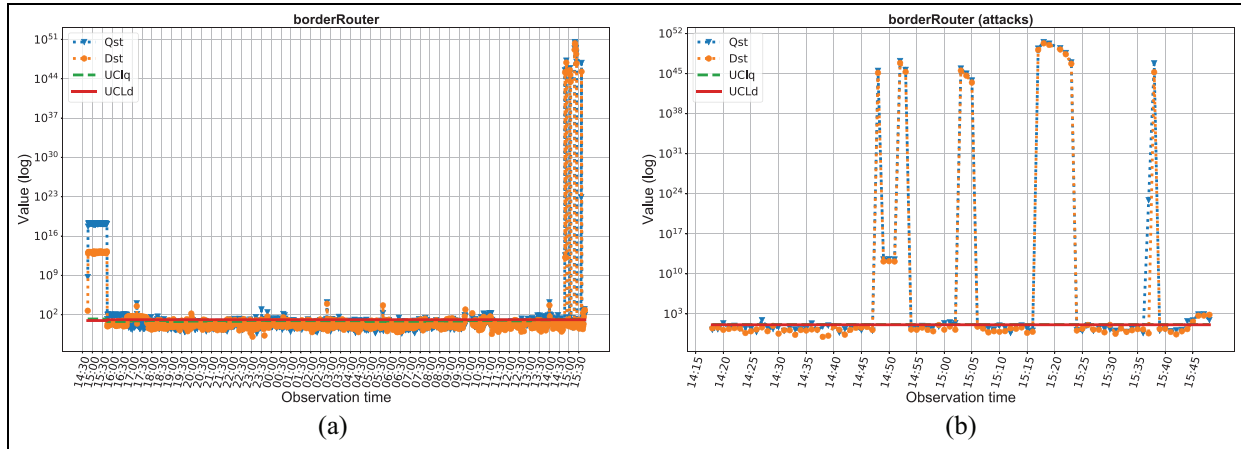


Figure 7. Q -st and D -st statistics evolution for the entire experiment (25 h) (a) and for the attack period (last 1.5 h) (b).

environments with a variety of devices and communications (to reproduce the obtained results, related data and scripts are available at the official GitHub repository).

There are several ways to evaluate the performance of the MSNM-Sensor as a detection system. In this work, the so-called monitoring plots²¹ and the widely used AUC (Area Under the Curve) and ROC (Receiver Operating Characteristic)^{24,28} are used. The first ones allow the security analyst to be alerted when anomalies are taking place in the system, that is, when the monitored statistics exceed the predefined control limits. In regard to the second ones, the AUC is a quantitative measurement of the area under the ROC curve. The ROC curve compares the evolution of the TP (True Positive) rate versus the FP rate for different values of the classifying threshold. The AUC is a performance indicator such that classifiers with $AUC = 1$ behave perfectly, that is, it is able to correctly classify all the observations, while a random classifier would get an AUC value around 0.5.

Anomaly detection. Figure 7 shows the Q -st (blue inverted triangles) and D -st (orange filled circles) statistics evolution with time obtained from the BR sensor. In addition, UCLs are also shown for the Q -st (UCLq), represented by a green dashed line, and for the D -st (UCLd), represented by a red continuous line. Figure 7(a) shows the statistics values for the total duration of the experiment, while Figure 7(b) shows the last 1.5 h, when the attacks were launched. In Figure 7(a), we can discern between three different intervals. In the first one, we experience a high FP rate, which is caused by the initial random calibration of the sensor. Due to the dynamism nature of the network traffic, each sensor uses the EWMA approach to dynamically calibrate the sensors every 60 min.²⁶ The effect of the EWMA-based

calibration can be seen in the second period that covers almost all the experimental time. We can see the effectiveness of the dynamic adaptation in comparison to the first period: most of the statistics values are below the control limits. Finally, the third one shows a clear deviation in the values of the statistics.

It is worth paying special attention to the attack period. In this way, Figure 7(b) clearly shows deviations in the statistics values when the attacks take place. First deviation corresponds to the high-rate DoS attack being the low-rate DoS attack in second place. Five minutes after the port scanning attack is taking place and the data exfiltration attack is the last one. In addition, the MSNM-Sensor deployment allows us to distinguish where the anomaly is coming from by inspecting similar monitoring plots from each of the inner routers R1 to R3. For instance, the port scanning attack is originated somewhere in the R1 network as it can be seen in Figure 8. Similarly, Figures 9 and 10 show that the data exfiltration and DoS-related attacks have been originated in R2 and R3 networks, respectively.

Detection performance. Figure 11 shows the ROC curves and AUC values obtained at BR. In this figure, on one hand, we consider the detection performance of all involved attacks (the black continuous line in the figure). On the other hand, we can see the corresponding detection performance per attack. In this case, only the attack's positive samples are considered, leaving the rest out. As expected, data exfiltration (or data leakage) is one of the most difficult attack to detect mainly due to its nature: it periodically sends little and randomly chosen portions of files to the attacker. On the contrary, the port scanning attack was almost completely detected. This is motivated by the high rate among scan attack occurrences. Finally, high-rate DoS attacks are

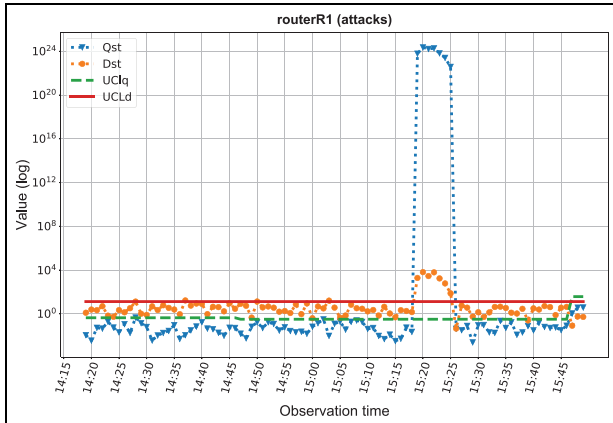


Figure 8. Q -st and D -st statistics evolution for the attack time interval in R1. It is clearly shown when the port scanning attack is taking place.

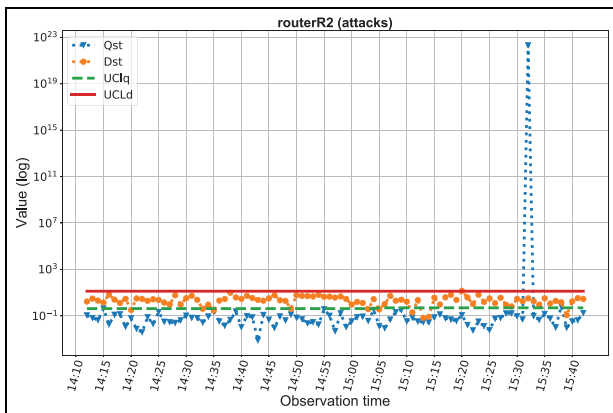


Figure 9. Q -st and D -st statistics evolution for the attack time interval in R2. It is clearly shown when the data exfiltration attack is taking place.

easy to detect since their network traffic pattern highly differs from the normal network traffic in comparison to low-rate DoS attacks.

Comparative study

Aiming to validate the real-time monitoring and detection capabilities of the MSNM-Sensor, we compare the previous results with the ones provided by Maciá-Fernández et al.,¹⁰ hereafter, the WIFS'16 results. This work confirms the hypothesis of that MSNM approach is able to monitoring hierarchical communication networks by aggregating statistics at the different network segments or levels. The authors use the gathered network traffic once the experiment has been finished to prove the MSNM hierarchical approach. Instead, the MSNM-Sensor deployment monitors and detects anomalies in real-time.

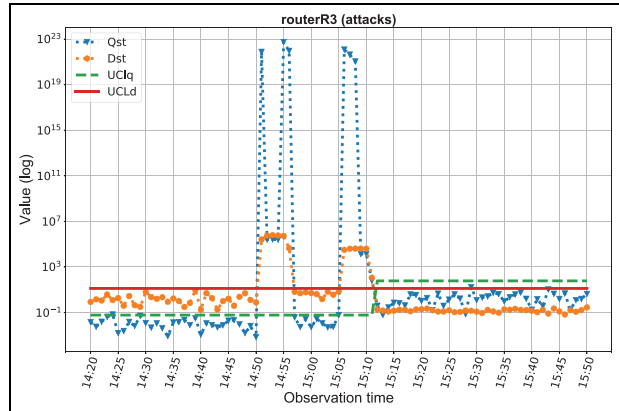


Figure 10. Q -st and D -st statistics evolution for the attack time interval in R3. It is clearly shown when the DoS attacks are taking place.

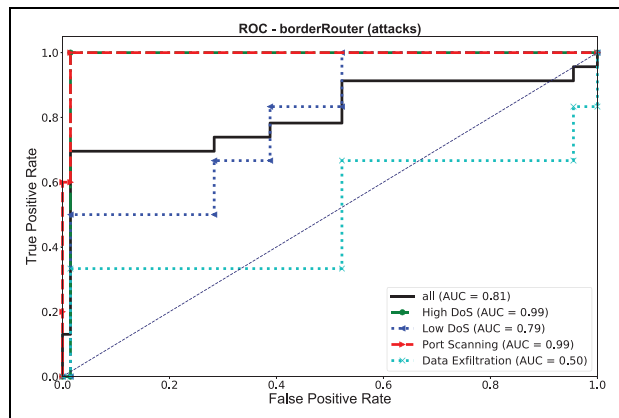


Figure 11. ROC curve and AUC values at BR for all involved attacks (black continuous line) and per attack.

Figures 12 and 13 show the contribution plots and ROC curves, respectively, from the above mentioned work. In comparison to those obtained by the MSNM-Sensor deployment experiment (see Figures 7(b) and 11), we can visually observe a similar pattern in regard to the Q -st and D -st evolution for every considered attack. In regard to the detection performance, the WIFS'16 results are slightly better than those provided here, mainly due to temporal synchronization issues in communications among sensors. They can add certain detection delay in comparison to the experiment ground-truth. Moreover, it is worth noting the influence of the dynamic calibration used here that could lead the system to reduce their detection performance. Different EWMA temporal windows of observations has been empirically considered, being 60 min the one getting better dynamical adaptation. However, extra work should be done in this direction, for example, to optimize this interval to reduce the calibration error.

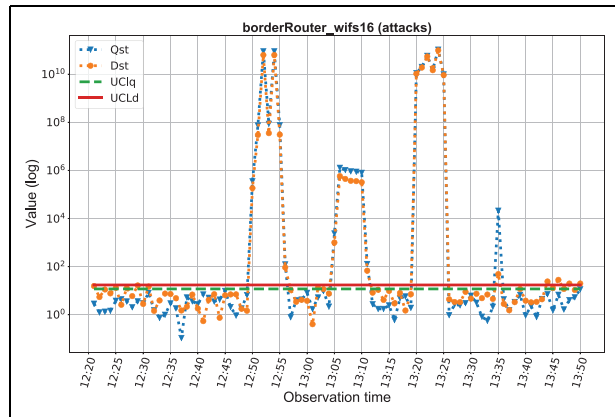


Figure 12. Q-st and D-st statistics evolution at BR router in WIFS'16 experiment for the attack time interval.

In general terms, the MSNM-Sensor deployment performs similar to the compared work. It proves its usefulness and suitability to be used in real network environments for real-time monitoring and anomaly detection.

Conclusions and future work

This work introduces the MSNM-Sensor, a ready-to-use tool for real-time monitoring and detection of security events in complex systems and network environments. The MSNM-Sensor offers an alternative way for testing MSNM-based IDS solutions in comparison to conventional approaches based on the use of predefined network datasets. Last approaches could limit the application of IDS solutions in real network environments.

Inherited from the MSNM methodology, MSNM-Sensor reduces and efficiently handles the monitoring information, maintaining its detection performance. On the contrary, existing IDS or security information and event management (SIEM) solutions consider raw data, thus adding extra monitoring traffic overhead. Moreover, MSNM-Sensor inherently adds privacy in monitoring communications since no sensible or raw data are sent to the central station just the statistics instead.

The solution relies on the use of lightweight algebraic and statistics operations that allow it to be embedded in less powerful devices, for example, wearable devices, environmental sensors, and IoT devices in general.

The MSNM-Sensor detection performance has been successfully tested for MSNM-based IDSs in hierarchical networks though some other IDS approaches can be also evaluated. The MSNM-Sensor should be tested in real scenarios, for example, IoT ecosystems or those found in smart cities in order to test its scalability and

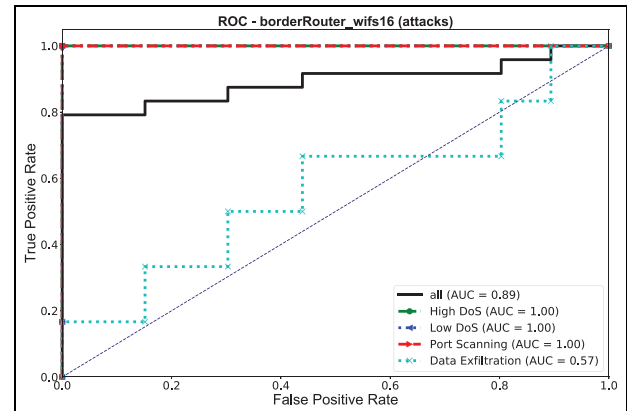


Figure 13. ROC curve and AUC values in WIFS'16 experiment at BR router for all involved attacks (black continuous line) and per attack.

detection performance. Finally, the anomaly diagnosis operation is still in development; thus, we will focus our future work on it too.

Supplemental material

The current version of the MSNM-S is released under a GPL license, and we encourage readers to be an active part of the project, which is available at Magán-Carrión et al.²⁷

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has been partially supported by Spanish MINECO (Ministerio de Economía y Competitividad) through projects TIN2014-60346-R, TIN2017-83494-R, and FEDER funds.

ORCID iD

Roberto Magán-Carrión  <https://orcid.org/0000-0002-7744-7308>

References

1. Nordrum A. Popular internet of things forecast of 50 billion devices by 2020 is outdated, <https://bit.ly/2kVkk9A> (accessed 20 March 2020).
2. Cisco Systems. Cisco visual networking index: global mobile data traffic forecast update, 2018-2023 white paper, <https://bit.ly/2TYO1DG> (accessed 20 March 2020).

3. ENISA. ENISA threat landscape report 2017, <https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2017> (accessed 22 September 2019).
4. Chaabouni N, Mosbah M, Zemhari A, et al. Network intrusion detection for IoT security based on learning techniques. *IEEE Commun Surv Tutor* 2019; 21(3): 2671–2701.
5. ENISA. ENISA threat landscape report 2018, <https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018> (accessed 22 September 2019).
6. Camacho J, Maciá-Fernández G, Díaz-Verdejo J, et al. Tackling the Big Data 4 vs for anomaly detection. In: *2014 IEEE conference on computer communications workshops (INFOCOM WKSHP)*, Toronto, ON, Canada, 27 April–2 May 2014, pp.500–505. New York: IEEE.
7. Maciá-Fernández G, Camacho J, Magán-Carrión R, et al. UGR'16: a new dataset for the evaluation of cyclostationarity-based network IDSs. *Comput Secur* 2018; 73: 411–424.
8. Camacho J, Pérez-Villegas A, García-Teodoro P, et al. PCA-based multivariate statistical network monitoring for anomaly detection. *Comput Secur* 2016; 59: 118–137.
9. Magán-Carrión R, Camacho J and García-Teodoro P. Multivariate statistical approach for anomaly detection and lost data recovery in wireless sensor networks. *Int J Distrib Sensor Netw* 2015; 11(6): 1–20.
10. Maciá-Fernández G, Camacho J, García-Teodoro P, et al. Hierarchical PCA-based multivariate statistical network monitoring for anomaly detection. In: *2016 IEEE international workshop on information forensics and security (WIFS)*, Abu Dhabi, United Arab Emirates, 4–7 December 2016, pp.1–6. New York: IEEE.
11. Camacho J, García-Teodoro P and Maciá-Fernández G. Traffic monitoring and diagnosis with multivariate statistical network monitoring: a case study. In: *2017 IEEE security and privacy workshops (SPW)*, San Jose, CA, 25 May 2017, pp.241–246. New York: IEEE.
12. Camacho J, Maciá-Fernández G, Fuentes-García NM, et al. Semi-supervised multivariate statistical network monitoring for learning security threats. *IEEE Trans Inf Forensics Secur* 2019; 14: 2179–2189.
13. Camacho J, García-Giménez JM, Fuentes-García NM, et al. Multivariate Big Data analysis for intrusion detection: 5 steps from the haystack to the needle. *Comput Secur* 2019; 87: 1–11.
14. Kanaoka A and Okamoto E. Multivariate statistical analysis of network traffic for intrusion detection. In: *Proceedings of 14th international workshop on database and expert systems applications*, Prague, Czech Republic, 1–5 September 2003, pp.472–476. New York: IEEE.
15. Lakhina A, Crovella M and Diot C. Diagnosing network-wide traffic anomalies. In: *Proceedings of the 2004 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'04)*, Portland, OR, 30 August–3 September 2004, pp.219–230. New York: Association for Computing Machinery.
16. Callegari C, Gazzarrini L, Giordano S, et al. Improving PCA-based anomaly detection by using multiple time scale analysis and Kullback–Leibler divergence. *Int J Commun Syst* 2014; 27(10): 1731–1751.
17. Delimargas A, Skevakis E, Halabian H, et al. Evaluating a modified PCA approach on network anomaly detection. In: *2014 international conference on next generation networks and services (NGNS)*, Casablanca, Morocco, 28–30 May 2014, pp.124–131. New York: IEEE.
18. Pérez-Villegas A, García-Jiménez J and Camacho J. *FaaC (feature-as-a-counter) parser*. Github, <https://github.com/josecamachop/FCParser> (accessed 30 September 2019).
19. Jackson JE and Mudholkar GS. Control procedures for residuals associated with principal component analysis. *Technometrics* 1979; 21(3): 341–349.
20. Hotelling H. *Multivariate quality control: techniques of statistical analysis*. New York: McGraw-Hill, 1947.
21. Nomikos P and MacGregor JF. Multivariate SPC charts for monitoring batch processes. *Technometrics* 1995; 37(1): 41–59.
22. Camacho J. Observation-based missing data methods for exploratory data analysis to unveil the connection between observations and variables in latent subspace models. *J Chemom* 2011; 25(11): 592–600.
23. Fuentes-García M, Maciá-Fernández G and Camacho J. Evaluation of diagnosis methods in PCA-based Multivariate Statistical Process Control. *Chemom Intell Lab Syst* 2018; 172: 194–210.
24. Bhuyan MH, Bhattacharyya DK and Kalita JK. Network anomaly detection: methods, systems and tools. *IEEE Commun Surv Tutor* 2014; 16(1): 303–336.
25. Li Z, Rios ALG, Xu G, et al. Machine learning techniques for classifying network anomalies and intrusions. In: *2019 IEEE international symposium on circuits and systems (ISCAS)*, Sapporo, Japan, 26–29 May 2019, pp.1–5. New York: IEEE.
26. Camacho J. Visualizing Big Data with compressed score plots: approach and research challenges. *Chemom Intell Lab Syst* 2014; 135: 110–125.
27. Magán-Carrión R, Camacho J and Maciá-Fernández G. *MSNM-S: multivariate statistical network monitoring–sensor*. Github, <https://github.com/nesg-ugr/msnm-sensor> (accessed 30 September 2019).
28. Salo F, Injadat M, Nassif AB, et al. Data mining techniques in intrusion detection systems: a systematic literature review. *IEEE Access* 2018; 6: 56046–56058.