Addressing Multi-Stage Attacks Using Expert Knowledge and Contextual Information

Francisco J. Aparicio-Navarro*, Timothy A. Chadza†, Konstantinos G. Kyriakopoulos†,‡, Ibrahim Ghafir†, Sangarapillai Lambotharan†, and Basil AsSadhan§
*Faculty of Technology, De Montfort University, Leicester, LE1 9BH, UK
†Wolfson School of Engineering, Loughborough University, Loughborough, LE11 3TU, UK
‡Institute for Digital Technologies, Loughborough University London, London, E15 2GZ, UK
§Department of Electrical Engineering, King Saud University, Riyadh, 11421, Saudi Arabia
e-mail: fnavarro@dmu.ac.uk, {t.a.chadza, elkk, i.ghafir, s.lambotharan}@lboro.ac.uk, bsadhan@ksu.edu.sa

***Abstract*— New challenges in the cyber-threat domain are driven by tactical and meticulously designed Multi-Stage Attacks (MSAs). Current state-of-the-art (SOTA) Intrusion Detection Systems (IDSs) are developed to detect individual attacks through the use of signatures or identifying manifested anomalies in the network environment. However, an MSA differs from traditional one-off network attacks as it requires a set of sequential stages, whereby each stage may not be malicious when manifested individually, therefore, potentially be underestimated by current IDSs. This work proposes a new approach towards addressing this challenging type of cyber-attacks by employing external sources of information, beyond the conventional use of signatures and monitored network data. In particular, both expert knowledge and contextual information in the form of Pattern-of-Life (PoL) of the network are shown to be influential in giving an advantage against SOTA techniques. We compare our proposed anomaly-based IDS, based on decision making powered by the Dempster-Shafer (D-S) Theory and Fuzzy Cognitive Maps (FCMs), against Snort, one of the most widely deployed IDS in the world. Our results verify that the use of contextual information improves the efficiency of our IDS by enhancing the Detection Rate (DR) of MSAs by almost 50%.**

***Keywords*—Contextual Information, Fuzzy Cognitive Maps, Intrusion Detection System, Multi-Stage Attack, Network Security, Pattern-of-Life, Snort**

I. Introduction

Cyber-security has increasing importance to Internet users. Providing strong and reliable security mechanisms has become vital in all areas of society. Snort [1] is one of the most widely deployed Intrusion Detection Systems (IDS) worldwide, which has become the actual standard for the industry [2]. Since it is a publicly available open-source IDS, Snort counts millions of downloads to date. Many organisations base the security of their network infrastructure on the efficiency of Snort. Nonetheless, Snort is a signature-based IDS. Hence, the frequent update of the signatures database, as well as the manual creation of new signatures, are essential to maintain the efficiency of Snort against new cyber-attacks. However, this is time-consuming and requires intensive human involvement.

The appearance of new forms of cyber-threats, such as Multi-Stage Attacks (MSAs), has created new challenges, which current IDSs, such as Snort, may fail to address. An MSA differs from traditional one-off network attack as it is launched in multiple stages and steps [3], and aims to maintain long-term access to a target machine. Each of the stages that composes an MSA comprises of different steps, which may not be malicious when implemented individually, but all are necessary for its successful completion. Only when executed sequentially, can the attacker succeed in the completion of the MSAs. Also, the time between separate attack stages can span hours, days or months, making the detection of MSAs extremely challenging for most current IDSs.

Due to the challenges that this new type of cyber-threat presents, Snort might not be an efficient solution to detect MSAs. Although Snort could detect the individual malicious steps that compose an MSA, Snort may not be able to correlate the relationship between consecutive steps and fail to detect the completion of the MSA. Therefore, to overcome these new challenges, novel and more intelligent detection approaches that exploit new sources of information need to be proposed.

Current IDSs use measurable network traffic information from the protected system or signatures of known cyber-attacks during the intrusion detection process. However, these systems do not generally take into account available high-level information (i.e. above the network operation) regarding the protected system [4]. The next generation of IDSs should be able to adapt their detection characteristics based not only on the measurable network traffic information, but also on the context in which these systems operate, and the information provided by the users or administrators.

In [5], we described an unsupervised anomaly-based IDS designed to detect MSAs, which exploits contextual information in the form of a Pattern-of-Life (PoL) model, and information related to expert judgment on the network behaviour. In particular, this IDS focuses on detecting a 5-steps MSA, in real-time, without previous training process. The main goal of this MSA is to create a Point of Entry (PoE) to a targeted machine, which could be used for the completion of an Advanced Persistent Threat (APT) like attack [6]. As we demonstrated in [7], a Fuzzy Cognitive Map (FCM) [8] can be used to incorporate the PoL into the detection process.

In this paper, we have evaluated the efficiency of our anomaly-based IDS with FCM against Snort, using both the standard configuration file and adapted rules to detect an MSA. Specifically, the novel set of rules have been carefully customised to detect the different steps of the 5-steps MSA. Although, the MSA implemented for this work replicates the same steps as in [5], it is worth noting that the background network traffic, the duration of the individual steps, and the time between steps varies. Therefore, the presented detection results are completely new and unpublished. Additionally, the efficiency of our anomaly-based IDS has been evaluated in a live operational manner, analysing real network traffic in real-time.

The rest of the paper is organised as follows. In Section II, the most relevant previous work is reviewed. The signature-based IDS Snort is described in Section III. In Section IV, the proposed approach for the use of an FCM within our anomaly-based IDS is described. The network testbed, the analysed network traffic dataset, and the implemented MSA are described in Section V.

Section VI describes the experiment results. Finally, conclusions and suggestions for future work are given in Section VII.

II. Related Work

Snort has been described extensively in the literature. For instance, the authors of [2] present a good description of the Snort architecture. This work aims to identify factors that influence and impact the performance of Snort, by the use of analytical queuing models. In [9], the authors use Snort in a campus network with more than 40 thousand hosts. An IDS alert correlator, build upon Snort, has been deployed in this network aiming to reduce the number of false positive alarms. The authors of [10] use Snort to create a training dataset to detect malware. This dataset is utilised to train a proposed lightweight IDS. Specifically, Snort is used to scan the malware dataset to separate packets identified as malicious. The authors highlight the fact that Snort may not be able to detect the malware if that malware is a novel variant for which there is no matching signature. In [11], Snort is used as a mitigation measure to protect networked control systems. A Modbus control system is deployed to simulate cyber-attacks. In this work, Snort is customised into the supervisor hardware of the defensive architecture for Modbus TCP/IP traffic monitoring. In [12], Snort has also been employed to protect virtual environments such as Software-Defined Networks (SDN).

The authors of [13] survey current research on context-based information fusion systems, and highlight the importance gained by these systems in the last few years. One technique that provides the capability of integrating contextual information to the detection process is the FCM. The authors of [8] provide a detailed description of the FCM technique and its mathematical foundation. The authors of [14] develop, using FCMs, an actionable model of situation awareness for army infantry platoon leaders that could replicate human cognition. Their FCM design structures the goals and subgoals of the platoon, and the relationships between these goals. A similar approach is introduced in [15], in which situation awareness is represented using an FCM.

Many works on MSAs have used attack graphs on alert messages for network security assessment, attack countermeasure selection and mitigation deployment [16]. The attack graph technique has also been used in collaboration with alert correlation clustering to decrease the false positive alerts [17]. However, attack graph techniques do not scale effectively to represent rich and complex scenarios and for this reason are impractical to use in real case environments. The authors of [18] propose a flow-based IDS to detect Brute Force SSH (BFS) attacks in real-time. This work considers an BFS attack as a 3-phase MSA. The IDS uses the metrics packets-per-flow and minimum number of flow records to identify the different phases of the attack. The detection is based on a number of thresholds defined based on these metrics.

III. Snort - Intrusion Detection System

Snort is a signature-based IDS that captures and inspects the header and payload in network packets to identify malicious traffic. It can be configured as a sniffer to simply read network packets or as a packet logger to analyse and store packets. Snort uses a signature database of known indicators of attacks. The signatures are represented as a set of rules. The network packets are analysed against the signatures, and an alert is raised if a match is identified.

A. Snort Architecture

As represented in Fig. 1, the architecture of Snort comprises five main components: packet decoder, preprocessor, detection engine, alert and log system, and output module. Initially, Snort uses the packet decoder to capture raw network traffic, utilising Data Acquisition Library (DAQ), to extract and forward network packets to the preprocessor. Next, the preprocessor is used to arrange or modify corrupted network packets, before these are sent to the detection engine. Among other functionalities, the preprocessor performs early packet dropping, classification, layer three Internet Protocol (IP) fragment reassembly, and layer four Transmission Control Protocol (TCP) session reconstruction [2]. Then, the detection engine, the most important component of Snort, analyses the network packets against the rules in the attack signature database. If no match is found, the frame is dropped, otherwise the alert and log system generates the corresponding output.

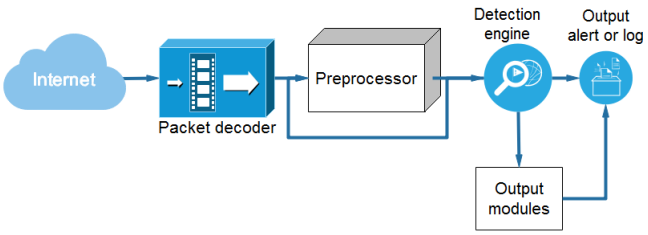

Fig. 1. Snort architecture schematic representation comprising of five components: packet decoder, preprocessor, detection engine, alert and log system, and output module.

By default Snort generates two outputs: the *alert* and *snort.log* files. The former is human readable and stores essential information about the rule, such as signature IDs, protocol type, classification, timestamps, alert description and packet header. In some cases, the original dump packet is also included. The *snort.log* file records the complete packet that triggered the alert, including the information present in *alert* file. The *snort.log* is machine readable whose format corresponds to the specified representation, for example, in unified2 *snort.u2* format. Finally, the output module controls the type of output of the alert and log system, which allows users to get a customised output, such as csv and unified2.

B. Snort Rules and Configuration

The Snort rules play a critical role in the detection of malicious activity in a network. Snort uses configuration files to specify all the rules used during the detection process. The standard Snort configuration file (i.e. the *snort.conf* file) can downloaded from the Snort website [1]. Other customised set of rules can be defined by the IDS administrator, as well as other rule configuration files can be downloaded from the Internet. For instance, the one managed by the Pulledpork [19], a perl script that aggregates all the Snort rules into a single file.The latest version of Snort rules (2.9.11.1), as well as a limited number of previous versions, can be also downloaded from the Snort website. The paths to all these rules have to be specified in the configuration file *snort.conf*. Additionally, the pulledpork can be configured to update Snort automatically with new rules.

Snort uses the file *snort.conf* to specify network variables, and to configure the packet decoder, the detection engine, the dynamic loaded libraries, the preprocessors, and the output plugins. The configuration file *snort.conf* is also used to customise the rule sets (i.e. administrator, preprocessor, decoded and shared object rules). Redundant rules and rules that generate false positives are deprecated. Customised rules can also be created by specifying essential parameters. For simple rules, Snorpy [20], a web based Snort rule creator can be used. Optionally, rules can be downloaded from other sources for example Emerging threats website [21]. As described in Section V, the configuration file *snort.conf* has been modified to customise the set of rules, as part of our experiments.

IV. Contextual Intrusion Detection Methodology

The detection methodology that we present to detect MSAs builds upon the unsupervised anomaly-based IDS that we previously presented in [7]. It is based on a cross-layer multi-metric architecture to carry out the detection. This IDS uses the Dempster-Shafer (D-S) Theory [22] as the data fusion technique to create an

overall belief on whether the currently analysed network traffic is normal or malicious. Moreover, an FCM is integrated within the IDS to add the contextual information and expert knowledge into the detection process. The outcome of the FCM is used to adjust the values to be fused by the D-S Theory, as described in [7].

A. Modelling Contextual Information

The graphical design of an FCM is characterised by a set of nodes interconnected by causal links. The nodes represent time-varying events or actions that describe the behaviour of the system. Each node C carries a weight $A(t)$ in the fuzzy range [0, 1], which indicates the importance that the concept has in the system, at time t. The links between nodes represent the causal relationship between events. Each link is assigned a weight value $w_{ij}(t)$ in the fuzzy interval [-1, 1], which indicates the relationship and degree of influence from the nodes C_i to C_j. An FCM can be represented by an $[m \times m]$ adjacency matrix M, where $[M(t)]_{ij} = |w_{ij}(t)|$, and m is the number of nodes in the FCM. The matrix M describes the relationship between the nodes and the weight $w_{ij}(t)$ associated with each link. A comprehensive description of the design of FCM models can be found in [8].

An FCM evolves via an iterative process in which, at each future time step, the weight value of each concept $A(t)$ is computed using an activation function f. The value of $A_i(t)$ changes at each iteration as described in Eq. (1):

$$A_i(t+1) = f(K) = f\left(A_i(t) + \sum_{j=1 j\neq i}^{m} w_{ij}(t) \cdot A_j(t)\right) \qquad (1)$$

where $A_i(t+1)$ is the weight value of node C_i at time $t+1$, $A_j(t)$ is the weight value of node C_j at time t, and $w_{ij}(t)$ is the degree of influence of node C_i on node C_j. In this work, we have employed the hyperbolic tangent activation function, defined in Eq. (2):

$$f(K) = \frac{e^K - e^{-K}}{e^K + e^{-K}} \qquad (2)$$

The FCM process continues for a number of iterations until the output of the activation function converges to a final fixed model (i.e. when the weight values $A(t)$ in all the nodes do not change in successive iterations). In our experiments, the output of the activation function always converges after a number of iterations.

B. FCM Design Based on Contextual Information

In this work, we have made use of contextual information, in the form of the PoL of the network usage, and expert knowledge related to judgment on the network behaviour during the different stages of the MSA. The network administrator defines the relationships between concepts and provides the different weight values $w_{ij}(t)$ associated to each link, based on previous knowledge and judgment.

In order to model the PoL, four nodes have been defined. These represent the different steps that compose the MSA implemented in this work: fping, nmap, OpenVAS, and BFS. Also, two additional concepts are defined as the two possible outcomes of the FCM (i.e. $C_5 = Normal$ and $C_6 = Attack$). The weights $A(t)$ associated with these two concepts are used to adjust the belief values assigned prior to the data fusion process. This process would allow incorporating the contextual information into the detection process of our IDS.

The FCM model used in our experiments is represented in Fig. 2. This is similar to the one that we presented in [5]. To avoid cluttering the figure, only four weights w_{ij} have been included in the FCM model representation. All the weights w_{ij} used in our experiments have been tabulated in the $[6 \times 6]$ matrix M, as shown in Fig. 3. As an example, the weight $w_{15} = 0.7$, which represents the level of influence of node C_1 on C_5, corresponds with the $(1,5)^{th}$ element of the adjacency matrix. In this example, since fping can be benign, a high weight value is assigned to w_{15}. Also, since fping may also be part of an MSA, a very low weight value is assigned to the concept *Attack* (i.e. $w_{16} = 0.1$). Similarly, a low weight value is assigned to w_{12} because fping can be followed by nmap as part of an MSA, although it may not be always the case.

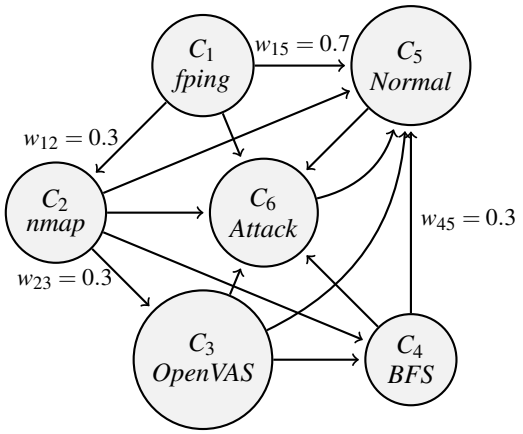

Fig. 2. FCM model of the MSA attack used in the presented experiments, in which nodes represent steps of the MSA and connections denote the relationships between concepts, previously shown in [5].

$$M = \left(\begin{array}{c|cccccc} & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \\ \hline C_1 & 0 & 0.3 & 0 & 0 & 0.7 & 0.1 \\ C_2 & 0 & 0 & 0.3 & 0.3 & 0.5 & 0.5 \\ C_3 & 0 & 0 & 0 & 0.3 & 0.3 & 0.7 \\ C_4 & 0 & 0 & 0 & 0 & 0.3 & 0.9 \\ C_5 & 0 & 0 & 0 & 0 & 0 & -0.1 \\ C_6 & 0 & 0 & 0 & 0 & -0.3 & 0 \end{array}\right)$$

Fig. 3. $[6 \times 6]$ adjacency matrix including the weight values w_{ij} used in our experiments. It maps the FCM model shown in Fig. 2.

We have defined a number of thresholds for each metrics, in order to allow the detection system inferring which of the MSA steps is being measured. These thresholds correspond with the metric value expected at a given time and day, based on the PoL of the network. This is based on the assumption that the network usage would show unexpected measurable changes from the normal behaviour during the implementation of the various steps of the MSA. These abnormal changes, which could occur at any time of the day, would manifest themselves differently in each analysed metric when different MSA steps are conducted.

In [5], we introduced an approach to address the challenge created by the temporal relation between the different steps of the MSA. We have defined a fixed time frame within which the weight value of $A_i(t)$ remains active after the attack in $C_{i,\ i=[1,\ 4]}$ has been detected. Since the MSA design for this work lasted for 7 minutes, we have empirically set the time frame to 2 minutes. As an example, consider the situation in which the IDS detects $C_2 = nmap$, the initial vector state would be $A(0) = [0,\ 1,\ 0,\ 0,\ 0,\ 0]$. Within the following 2 minutes, if the IDS detects $C_3 = OpenVAS$, the initial vector state would be $A(0) = [0,\ 1,\ 1,\ 0,\ 0,\ 0]$. However, if the IDS detects $C_3 = OpenVAS$ after the 2 minutes time frame has expired, the initial vector state would then be $A(0) = [0,\ 0,\ 1,\ 0,\ 0,\ 0]$. This approach brings the temporal relation between the different MSA steps into the initial vector state employed in Eq. (1).

V. Experimental Setup

A. Network Traffic Measurements

The network traffic used in the experiments has been gathered in the Local Area Network (LAN) presented in Fig. 4. This testbed LAN includes an attacker PC running Kali Linux, an IDS PC running Snort 2.9.11.1 on Ubuntu 16.04, and a victim PC

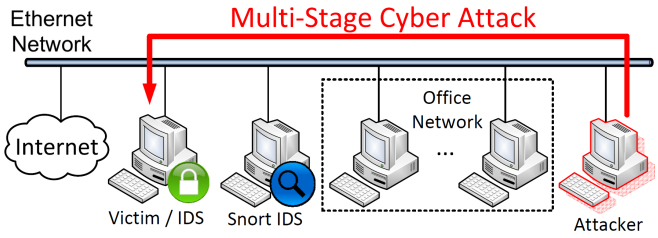

Fig. 4. Testbed LAN topology; PC on the right initiates an MSA against the victim PC in the left. The PCs in the Office Network generate background traffic, and the Snort IDS performs the signature-based detection.

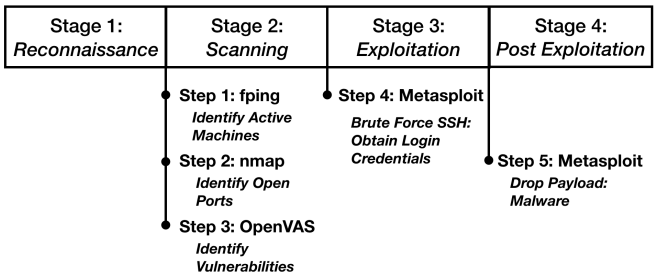

Fig. 5. Diagram comprising the different stages and steps that compose the 5-steps MSA employed in this work.

running our IDS. The attacker initiates an MSA against the victim. Additionally, the LAN comprises a portion of the University's network traffic, used as background traffic to construct the PoL.

The victim PC, which runs the IDS on Ubuntu 16.04, gathers the network traffic in pcap format using the network packets analyser tcpdump [23]. Next, the pcap file is processed using tshark [24] to filter the relevant set of metrics employed during the MSA detection process. The Snort IDS PC also gathered network traffic in pcap format using tcpdump. Then, the pcap file is used for offline analysis by Snort. It is worth noting that the network traffic from the office network, used as background traffic, was collected beforehand and retransmitted using tcpreplay [25]. A detailed description of the office network traffic can be found in [7].

In total, the gathered dataset comprises 300730 network traffic frames. In the case of the presented anomaly-based IDS, five different metrics have been extracted from the dataset, which are used to carry out the intrusion detection analysis. These metrics are the number of frames transmitted per second; the number of unique destination ports that receive traffic per second; Throughput, i.e. the number of transmitted bytes per second; the number of Address Resolution Protocol (ARP) frames transmitted per second; and the number of SSH requests transmitted per second.

B. Evaluated Multi-Stage Attack

The attacker implements a five-steps MSA against the victim. All the tools used to implement the MSA can be found as part of the Penetration Testing Linux distribution Kali Linux [26]. A bash script was used to automate and reduce the delay of implementing the different MSA steps. The attack lasted for 7 minutes approximately, and the different steps were launched stochastically. It is worth nothing that, although the script used to launch the MSA is similar to the one used in [5], this script allows the attacker to pause and resume each step at any time. Therefore, the duration of the individual steps and the time between steps changes stochastically. Furthermore, since the launch of the MSA is stochastic, the background network traffic used to construct the PoL would be different every time the MSA is conducted.

The implemented MSA is composed of the following steps:
1) Scanning for active machines in the network, using fping
2) Scanning for open network ports, using nmap
3) Scanning for vulnerabilities, using OpenVAS
4) Dictionary brute force SSH, using Metasploit
5) Drop malicious payload, also using Metasploit

Fig. 5 represents the different stages and steps that compose the implemented MSA. The steps that comprise the MSA have been specifically designed for this work. However, the order at which the different stages are implemented follows the Zero Entry Hacking (ZEH) methodology described in [3].

Initially, we assume that a passive reconnaissance stage has been conducted in which the attacker obtains the username and the network IP address where the targeted machine is connected. Since the reconnaissance is implemented passively and does not actively interact with the targeted victim, this step has been excluded from the MSA detection process.

The first step in the scanning stage uses fping to obtain a list of IP addresses for all the live machines in the network. This step provides the attacker with the information required to identify the specific IP address of the target. The next step is implemented using the network mapper nmap [27]. This is a popular open source tool that provides a variety of probing techniques for network exploitation and security auditing. This step would provide the attacker with a list of open ports in the target machine. The last step during the scanning stage aims to identify possible vulnerabilities. The vulnerability scan is implemented using OpenVAS [28], a software framework that offers vulnerability scanning and vulnerability management capabilities.

During the exploitation stage, the attacker makes use of the information obtained during the scanning stage. In particular, the fourth step attempts to guess the login credentials of the target victim using a dictionary brute force. The attacker uses a list of frequently used passwords to establish an SSH connection to the target. Finally, the attacker uses the connection established with the target to install a malware payload that will leave a back-door open in the victim machine. In other words, it will create a PoE to the target. The penetration testing framework Metasploit [29] has been used to implement the last two steps.

C. Snort Rules Customisation

For the Snort experiments, we have made use of the default set of rules, as well as a new set of rules, customised to detect the MSA. Regarding the default rules, the rules of Snort version 2.9.11.1 were used, as well as, the community rules, which are aggregated by pulledpork 0.7.4. The default set of rules contained 10444 rules, out of which 10026 were for the detection engine, 150 for the packet decoder, and 268 for the preprocessor. Regarding the customised rules, these were created in the directory */etc/snort/rules*, and the paths to these rules were specified in the *snort.conf* configuration file. All the default Snort preprocessor rules were used, along with 13 customised detection rules. Hence, the customised configuration file consisted of 281 rules in total (i.e. 13 rules for the detection engine, and 268 rules for the preprocessor). Next, a description of the Snort rules customization for the four MSA steps is provided.

During the fping attack, ARP packets were sent to probe for active machines. ARP detection rules could not be used for this step, because Snort supports only four protocols: TCP, UDP, IP and Internet Control Message Protocol (ICMP). Alternatively, the ARPSpoof preprocessor was used, which detects ARP spoofing attacks, unicast ARP requests, and Ethernet to IP mapping inconsistencies [1]. In particular, the preprocessor set of rules with Generator ID (GID) 112 was used, which raises an alert for any of the following four events: (i) unicast ARP request, (ii) Ethernet frame ARP mismatch at source, (iii) Ethernet frame ARP mismatch at destination, and (iv) ARP cache overwrite attack. However, the use of the preprocessor rules was ineffective because the fping attack was of an ARP

scanning nature, rather than ARP spoofing, which the preprocessor could not detect.

To detect nmap, a set of representative Snort network scanning rules pertaining to port scanning, IP mapping, and various application scanners, have been used. These are rules for the detection engine and can be found in [30]. In addition, one sfPortscan preprocessor was also used to trigger alerts for three types of port scanning threats: nmap, decoy and distributed, as well as, for portsweep events. Regarding the detection of OpenVAS , one single rule was manually created to raise an alert for any established TCP flow destined to the victim's range of HTTP ports. Finally, the BFS attack step was detected using both the Snort SSH preprocessor and a detection engine rule. The SSH preprocessor detects several SSH related exploits, including challenge response buffer overflow and protocol mismatch exploits. The detector rule was created to alert for any manifestation of the string *"SSH"* in the content of TCP packets, with a destination port number equal to 22.

The default output plugins have been used, and a script was created to extract essential fields (e.g. timestamp, signature ID, alert description, alert category, source and destination IPs, and source and destination port numbers) from the alert file for processing.

VI. Results and Analysis

This section describes the detection results of the presented IDS in an online and live mode, with and without an FCM. Additionally, these results are compared against those generated by the signature-based IDS Snort. The analysis of Snort has been conducted using the standard configuration and community rule files, and the customised set of rules previously defined. The efficiency of the IDS has been evaluated using the following performance metrics, which provide evidence of how effective the IDSs are at making correct detections:

- Detection Rate (DR) - Proportion of malicious data correctly classified as anomalous among all the malicious data:

$$DR = \frac{TP}{TP+FN} \tag{3}$$

- False Positive Rate (FPr) - Proportion of normal data misclassified as malicious among all the normal data:

$$FPr = \frac{FP}{TN+FP} \tag{4}$$

- Overall Success Rate (OSR) - Proportion of frames correctly classified among all the data:

$$OSR = \frac{TP+TN}{TP+FP+TN+FN} \tag{5}$$

- F-score - Tradeoff between Precision and DR, used to compare two distinctive classification methodologies:

$$F\text{-}score = \frac{2 \cdot Precision \cdot DR}{Precision + DR} \tag{6}$$

where True Positive (TP) represents anomalies classified as malicious; True Negative (TN) represents normal instances classified as normal; False Positive (FP) represents normal instances misclassified as attack; False Negative (FN) represents anomalies misclassified as normal; and $Precision = TP/(TP+FP)$.

A. Signature-based IDS Snort Results

As expected from a signature-based IDS, Snort produced remarkably low number of FP alarms, both when the default and the customised set of rules are used. Regarding the default set of rules, Snort triggered 16879 alerts, 16512 TP alerts (i.e. 97.83% of all the alerts) and only 367 FP alerts (i.e. 2.17% of all the alerts). In terms of FPr, Snort produced 0.15% FPr using the default set of rules. In terms of OSR, Snort also produced encouraging results, reaching 87.38% OSR. However, there were a high number of FNs. The most plausible reason to this is the misdetection of fping attack. As elaborated in Section V-C, Snort currently detects ARP spoofing and not ARP scanning, hence it missed the entire fping attack. In total, 37580 malicious packets were undetected. Therefore, Snort only produced 30.53% DR. It is worth highlighting that these results have been generated using the default set of rules, which require minimal time-consuming to configure by the IDS administrator.

Regarding the customised set of rules, Snort triggered 21180 alerts, 20728 TP alerts (i.e. 97.86% of all the alerts) and only 462 FP alerts (i.e. 2.14% of all the alerts). We can already appreciate a small improvement in comparison with the default set of rules. Although, Snort produced 0.19% FPr using the customised set of rules (i.e. 0.04% increase), Snort improved the OSR results, reaching 88.75% OSR. The most encouraging improvement is generated in terms of DR. Snort produced 38.3% DR, which represents a 7.77% improvement (i.e. 4216 more malicious packets correctly detected). All these results have been tabulated in Table I.

B. Anomaly-based IDS Results

The comparison evaluation of the experimental results with and without the use of an FCM are presented in Figs. 6-8. The Y-axis of the figures represents the results in percentage, while the X-axis of the graphs represents time in seconds. The graphs in blue and red correspond to the results with and without the use of contextual information, respectively. All the figures include extra annotations to help identify the different steps of the MSA.

Regarding the DR, there is an evident improvement in the detection results when contextual information is included in the detection process. The graphs in Fig. 6 display a step change like improvement, which relates to the different stages of the MSA that have been detected. When the contextual information is considered, the IDS generates 80% DR at the end of the experiment, reaching 82% peak DR. The change improvement is almost consistent in both graphs. When the MSA reaches the step BFS, the IDS without an FCM starts misclassifying a large portion of malicious traffic (i.e. produces FN alarms). At the end of the MSA, the DR reaches 31% produced by the IDS without contextual information. Additionally, the detection results when FCM is considered are better for most of the experiments due to a more accurate detection of most of the MSA steps.

The FPr results of our IDS with and without the use of an FCM are compared in Fig. 7. We can see that the use of contextual information produces a relatively low number of false positive alarms, reaching 12% FPr at the end of the MSA. This represents a 10% worsening in the FPr results in comparison with the IDS alone, which reaches 2% FPr. Nonetheless, it is worth repeating that these results are generated by an unsupervised IDS detecting MSA, without any prior training process. Therefore, producing 12% FPr at the end of the MSA can be considered as good results, considering the improvement provided in terms of DR.

Fig. 8 presents the OSR comparison results. There is no evident difference between the two approaches. When the contextual information is considered as part of the detection process, 87% OSR is reached at the end of the attack. Both the IDS with and without FCM produce significant results in terms of OSR. In particular, the presented results evidence that the use of the

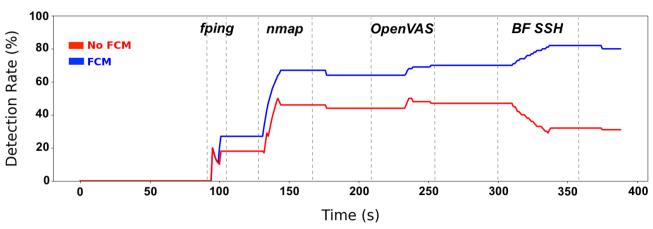

Fig. 6. Detection Rate results comparison between the methodologies: IDS with and without FCM.

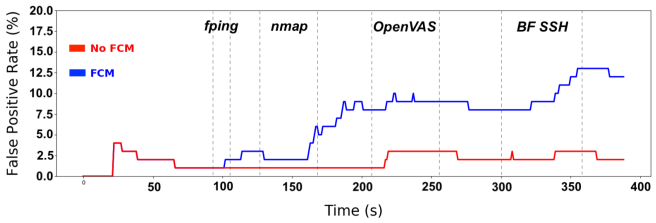

Fig. 7. False Positive Rate results comparison between two methodologies: IDS with and without FCM.

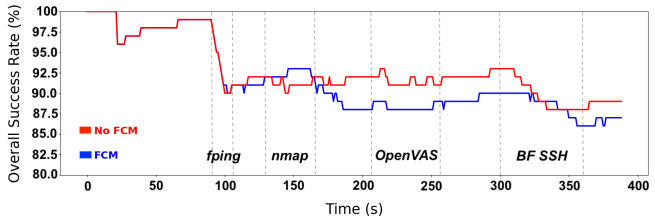

Fig. 8. Overall Success Rate results comparison between two methodologies: IDS with and without FCM

FCM provides improvement to the effectiveness of the IDS, without greatly affecting the correct classification of normal network traffic. All these results have been also tabulated in Table I.

VII. Conclusions

In this paper we have presented and evaluated a live operational IDS able to efficiently detect the presence of an MSA in real-time, without prior training process. This novel IDS exploits contextual information in the form of PoL model, and expert knowledge by the use of an FCM in conjunction with our IDS. The performance of our anomaly-based IDS has been compared against the signature-based IDS Snort. We have used both the default Snort rules and a customised set of rules adapted to detect the implemented 5-steps MSA.

From the presented results, the use of our anomaly-based IDS with an FCM outperforms the overall detection result generated by the rest of approaches evaluated in this work. The use of Snort with the customised set of rules generates better detection results than those generated by our IDS without an FCM. It also produces remarkably low number of false alarms using both, the default and the customised set of rules are used. However, Snort only produced 38.3% DR for the customised rules. When Snort uses the default rules, only 30.53% DR is reached. Furthermore, the time and the level of involvement required from the IDS administrator to customise the rules does not justify the small 7.77% improvement. Additionally, the use of signature-based IDS cannot adapt the frequent appearance of new forms of cyber-threats in a timely manner. On the other hand, the use of contextual information clearly improves the efficiency of our IDS detecting an MSA, as well as outperforming the efficiency of Snort. In terms of DR, the proposed live operational detection system, including an FCM, improves by $41-49\%$ DR when compared against the IDS without an FCM and Snort. Despite the clear increase in FP alarms, reaching 12% FPr, the gain in terms of DR well justifies the use of an FCM, and does not significantly affect the results in terms of OSR.

It is important to highlight that the design of an FCM is very context-specific, and may not be easily generalised. In our experiments, the design of the FCM has been adapted to an MSA specifically designed for this work. Furthermore, the approach that we have implemented to address the temporal relation between MSA steps does not efficiently adapt to the implementation of other MSAs. Hence, addressing this challenge remains an open issue.

TABLE I
MSA Detection Results Comparison

Detection Approach	DR(%)	FPr(%)	OSR(%)	F-score
IDS with FCM	80	12	87	0.62
IDS w/o FCM	31	2	89	0.43
Default Snort Rules	30.53	0.15	87.38	0.46
Custom Snort Rules	38.3	0.19	88.75	0.55

Finally, it is worth noting that the results generated by Snort have not been plotted in graphs, similar to those used to represent the results if our IDS. This is because Snort works with individual network packets, instead of aggregated traffic, making the graphical representation of the results more complex. Therefore, we decided to present the absolute result values only, once the analysis finished.

Acknowledgment

This work has been supported by the Gulf Science, Innovation and Knowledge Economy Programme of the UK Government under UK-Gulf Institutional Link grant IL 279339985.

References

[1] Cisco, "Snort - Network Intrusion Detection & Prevention System," [online] Available: https://www.snort.org (Access Date: 6 Jun, 2018).
[2] K. Salah, and A. Kahtani, "Improving snort performance under linux," in *IET communications*, vol. 3, no. 12, 2009, pp. 1883-1895.
[3] P. Engebretson, *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy*, Elsevier, 2013.
[4] A. Sadighian, S. T. Zargar, J. M. Fernández, and A. Lemay, "Semantic-based context-aware alert fusion for distributed Intrusion Detection Systems," in *Proc. of the International Conference on Risks and Security of Internet and Systems (CRiSIS)*, 2013, pp. 1-6.
[5] F. J. Aparicio-Navarro, K. G. Kyriakopoulos, I. Ghafir, S. Lambotharan, and J. A. Chambers, "Multi-stage attack detection using contextual information," in *Proc. of the IEEE Military Communications Conference (MILCOM)*, 2018, *in press*.
[6] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F. J. Aparicio-Navarro, "Detection of advanced persistent threat using machine-learning correlation analysis," in *Elsevier Future Generation Computer Systems*, vol. 89, 2018, pp. 349-359.
[7] F. J. Aparicio-Navarro, K. G. Kyriakopoulos, Y. Gong, D. J. Parish, and J. A. Chambers, "Using pattern-of-life as contextual information for anomaly-based intrusion detection systems," in *IEEE Access*, vol. 5, no. 1, 2017, pp. 22177-22193.
[8] C. D. Stylios, and P. P. Groumpos, "Modeling complex systems using fuzzy cognitive maps," in *IEEE Transactions on Systems, Man and Cybernetics: Systems and Humans*, vol. 34, no. 1, 2004, pp. 155-162.
[9] E. Raftopoulos, and X. Dimitropoulos, "IDS alert correlation in the wild with EDGe," in *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 10, 2014, pp. 1933-1946.
[10] K. F. Yu, and R. E. Harang, "Machine learning in malware traffic classifications," in *Proc. of the IEEE Military Communications Conference (MILCOM)*, 2017, pp. 6-10.
[11] C. Kim, and D. Robinson, "Modbus monitoring for networked control systems of cyber-defensive architecture," in *Proc. of the Annual IEEE International Systems Conference (SysCon)*, 2017, pp. 1-6.
[12] A. Abubakar, and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in *Proc. of the International Conference on Emerging Security Technologies (EST)*, 2017, pp. 138-143.
[13] L. Snidaro, J. García, and J. Llinas, "Context-based information fusion: A survey and discussion," in *Information Fusion*, 25, 2015, pp. 16-31.
[14] R. E. T. Jones, E. S. Connors, M. E. Mossey, J. R. Hyatt, N. J. Hansen, and M. R. Endsley, "Modeling situation awareness for Army infantry platoon leaders using fuzzy cognitive mapping techniques," in *Proc. of the Behavior Representation in Modeling and Simulation Conference (BRiMS)*, 2010, pp. 216-223.
[15] M. M. Kokar, and M. R. Endsley, "Situation awareness and cognitive modeling," in *IEEE Intelligent Systems*, vol. 3, 2012, pp. 91-96.

[16] S. Luo, J. Wu, J. Li, and L. Guo, "A multi-stage attack mitigation mechanism for software-defined home networks," in *IEEE Transactions on Consumer Electronics*, vol. 62, no. 2, 2016, pp. 200-207.
[17] N. El Moussaid, A. Toumanari, and M. El Azhari, "Security analysis as software-defined security for SDN environment", in *Proc. of the 4th International Conference on Software Defined Systems (SDS)*, 2017, pp. 87-92.
[18] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: a flow-based SSH intrusion detection system," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, 2012, pp. 86-97.
[19] J. J. Cummings, M. Shirk, and PulledPork Team, "PulledPork," [online] Available: https://github.com/shirkdog/pulledpork (Access Date: 20 Jun, 2018).
[20] C. Davis, "Snorpy 2.0 - Web Based Snort Rule Creator," [online] Available: http://snorpy.com/ (Access Date: 22 June, 2018).
[21] Emerging Threats, "Emerging Threats Rule Server," [online] Available: https://rules.emergingthreats.net/ (Access Date: 22 June, 2018).
[22] G. Shafer, *A mathematical theory of evidence*, Princeton University Press, 1976.
[23] V. Jacobson, C. Leres, and S. McCanne, "Tcpdump," 1987. [online] Available: http://www.tcpdump.org (Access date: 23 Jun, 2016).
[24] G. Combs, "TShark - The wireshark network analyser 2.4.6," [online] Available: https://www.wireshark.org/docs/man-pages/tshark.html (Access Date: 5 Apr, 2018).
[25] A. Turner and M. Bing, "Tcpreplay: Pcap editing and replay tools for *NIX and Windows," [online] Available: http://tcpreplay.sourceforge.net (Access Date: 05 Mar, 2018).
[26] M. Aharoni, D. Kearns, and R. Hertzog, "Kali Linux: Penetration Testing and Ethical hacking Linux Distribution," [online] Available: https://www.kali.org (Access Date: 05 Mar, 2018).
[27] G. Lyon, "Nmap: The network mapper Free security scanner," [online] Available: http://nmap.org/ (Access Date: 21 Jun, 2016).
[28] Greenbone, "Open Vulnerability Assessment System," [online] Available: http://www.openvas.org/index.html (Access Date: 05 Mar, 2018).
[29] Metasploit, L.L.C., "The metasploit framework," [online] Available: http://www.metasploit.com (Access Date: 05 Mar, 2018).
[30] M. Roesch, and B. Caswell, "Scan rules," [online] Available: https://github.com/eldondev/Snort/blob/master/rules/scan.rules (Access Date: 05 Sep, 2018).
[31] N. Dietrich, "Snort 2.9.9.x on Ubuntu 14 and 16 with Barnyard2, PulledPork, and BASE," [online] Available: https://www.snort.org/documents/snort-2-9-9-x-on-ubuntu-14-16 (Access Date: 17 Jul, 18).