Discovering Attackers Past Behavior to Generate Online Hyper-Alerts

Cláudio Toshio Kawakani [1], Sylvio Barbon Junior [1], Rodrigo Sanches Miani [2], Michel Cukier [3], Bruno Bogaz Zarpelão [1]

[1]Computer Science Department – State University of Londrina (UEL)
Londrina – PR – Brazil

[2]School of Computer Science (FACOM) – Federal University of Uberlândia
Uberlândia – MG – Brazil

[3]A. James Clark School of Engineering – University of Maryland
College Park – MD – USA

claudio.tk93@gmail.com, {barbon,brunozarpelao}@uel.br, miani@ufu.br

Abstract. *To support information security, organizations deploy Intrusion Detection Systems (IDS) that monitor information systems and networks, generating alerts for every suspicious behavior. However, the huge amount of alerts that an IDS triggers and their low-level representation make the alerts analysis a challenging task. In this paper, we propose a new approach based on hierarchical clustering that supports intrusion alert analysis in two main steps. First, it correlates historical alerts to identify the most common strategies attackers have used. Then, it associates upcoming alerts in real time according to the strategies discovered in the first step. The experiments were performed using a real dataset from the University of Maryland. The results showed that the proposed approach could properly identify the attack strategy patterns from historical alerts, and organize the upcoming alerts into a smaller amount of meaningful hyper-alerts.*

1. Introduction

Information and Communication Technology (ICT) plays a significant role in modern organizations, bringing many benefits concerning facility and productivity. Information is a valuable asset to every organization and its value dramatically increased in recent years [Shameli-Sendi et al. 2016]. Therefore, a security incident may cause loss of productivity, resources and reputation [Ahmad et al. 2012]. Incident management performs an important task for the information security of an organization, which defines processes to detect, analyze and respond to an incident [Ruefle et al. 2014].

Network or system intrusions can cause security incidents. Intrusion Detection Systems (IDS) are devices used to monitor information systems and networks for these intrusions. When the IDS detects a sign of security violation, it generates an alert and saves them into a log file. A better understanding of intrusions is possible by the analysis of alerts that the IDS triggers. Thereby, a security administrator can have a better situational awareness of the intrusions and efficiently respond to it. However, IDS tends to generate a huge amount of alerts. Moreover, each IDS alert presents low-level information, such as

KAWAKANI C. T.; BARBON JR, S.; MIANI, R. S.; CUKIER, M.; ZARPELÃO, B. B.
Discovering Attackers Past Behavior to Generate Online Hyper-Alerts
iSys | Revista Brasileira de Sistemas de Informação, Rio de Janeiro, vol. 10, No. 1, pp. 122-147, 2017

timestamp, source and destination IP address, ports, and signature. Therefore, the manual analysis of the IDS alerts becomes a time consuming and error prone task [Julisch 2003].

Two main problems in the intrusion alert analysis field have been approached in the literature. The first problem is the huge amount of alerts generated by IDS. To solve this problem, approaches to reduce the quantity of alerts using filtering and aggregation methods have been proposed [Julisch 2003, Spathoulas and Katsikas 2013]. The second problem is related to the low-level information of a single alert. A complete attack is composed of a sequence of attack steps (e. g., reconnaissance, vulnerability identification, and vulnerability exploit [Liu et al. 2008]). Therefore, instead of a single alert, a group of correlated alerts is required to represent a complete attack. Approaches to extract meaningful information of the correlated alerts, such as the attack strategies (sequence of alert signatures), have also been proposed [Treinen and Thurimella 2006, Liu et al. 2010].

In this work, we propose a new approach to assist the security analyst in the intrusion alert analysis, addressing the aforementioned problems. The proposed approach is composed of two elements: the offline correlator and the online correlator. The offline correlator aggregates historical alerts into attack scenarios using the connected components method, extracts attack strategy graphs and uses hierarchical clustering to group similar attack strategy graphs. For this, a new method to compare the similarity between two attack strategy graphs is also proposed. The attack characteristics of each generated cluster are then identified.

The online correlator generates hyper-alerts which contain useful attributes that assist the security analyst. A hyper-alert is composed by different low-level alerts and it is updated in real time as the upcoming low-level alerts are triggered. Considering the hypothesis that a previous attack strategy can be performed again in the future, each hyper-alert is associated with a cluster that the offline correlator defined previously. By analyzing the cluster information, the security analyst can better understand the characteristics of an attack when only the first alerts are available. The experiments were performed using a real dataset gathered by an IDS device deployed at the University of Maryland with about 40,000 computers.

The main contributions of this paper are: (a) a new correlation method that studies the past alerts and generates an information structure known as hyper-alerts, which describe the upcoming attack scenarios; (b) a new technique based on Jaccard Index to compare attack strategy graphs.

In the contribution (a), the proposed correlator could organize the huge amount of past alerts and identify the attack strategy patterns used previously. With the knowledge of the previous attack strategy patterns, the online correlator was able to summarize the upcoming alerts into a smaller amount of hyper-alerts that describe the upcoming attack scenarios. The attack scenario of each hyper-alert was associated to one of the attack strategy patterns, anticipating the nature of the attack. The results showed that the most of the attack strategies that the online correlator identified were similar to the attack strategies that the offline correlator discovered. This result strengthens the hypothesis that a previous attack strategy has a significant probability of being adopted again in the future. The contribution (b) is a new alternative to address the comparison of attack strategy graphs. This alternative could assist the hierarchical clustering to group the most similar attack

strategies and identify the attack strategy patterns.

This paper is an extended version of a previous work [Kawakani et al. 2016] and includes a new section about Intrusion Detection Systems, the study of additional related work, an improved explanation of the proposed approach, and new experiments and results focusing on the variation of the alerts sample size used in the offline correlator. The rest of this paper is organized as follows: Section 2 makes a brief summary of Intrusion Detection Systems, Section 3 presents the related work, Section 4 explains the offline and online correlators, Section 5 discusses the experiments and analyzes the results. Section 6 concludes this paper.

2. Intrusion Detection Systems

Security intrusion is a security event (or a combination of multiple security events) in which an intruder, with a set of actions, attempts to gain access to a system or disrupt the normal operations of a system [Stallings and Brown 2007]. Security intrusions threaten the integrity, confidentiality or availability of systems. Integrity ensures unauthorized users will not modify the data. Confidentiality guarantees the data will not be available for unauthorized users. Availability assures the data will always be accessible to authorized users.

To protect against intrusions, many layers of security can be used, such as user authentication, access control, and firewalls. Another layer of security may be the intrusion detection system (IDS). IDS is a software or a hardware device placed in computers or networks to detect suspicious activities, for example, unauthorized entry or file modification. When the IDS detects suspicious activity, it can display an alert or write the information about the event in a log file. Thereby, a security analyst can use this information to assist the system defense. An IDS can act proactively, attempting to stop possible intrusions. This kind of IDS is referred to as Intrusion Prevention System (IPS) [Patel et al. 2010, Scarfone and Mell 2007, Stallings and Brown 2007, Stavroulakis and Stamp 2010].

However, an IDS alert shows low-level information, e.g., source IP address, destination IP address, timestamp and signature of the attack. A single alert does not provide meaningful information to a security analyst. Furthermore, a massive amount of alerts tends to be generated when an IDS monitors a large network. For example, the dataset used in this work contains about 82 million alerts. This dataset was generated by an IDS deployed at the University of Maryland from April 2012 to March 2013. Therefore, the analysis of correlated alerts becomes more interesting than the analysis of single alerts.

2.1. Types of IDS

IDSs can be designed to operate in the host (host-based IDS) or in the network (network-based IDS), each one presenting different characteristics [Patel et al. 2010, Scarfone and Mell 2007, Stallings and Brown 2007, Stavroulakis and Stamp 2010, Debar 2002, Vacca 2013].

Host-based IDS was the first approach explored in intrusion detection. It aims to monitor the events of a single component, such as a server, a client host or an application service. These events can be detected in system logs, applications activity, file changes, system setting changes and host traffic. When a host-based IDS detects suspicious activity, it generates an alert with different attributes, such as timestamp, alert type, and rating

(e.g., priority, severity). Besides, depending on the alert type, some specific attributes are also informed (when available), such as user ID, application information or IP address and port information [Scarfone and Mell 2007].

With the increasing usage of the Internet, network attacks have become more frequent. The purpose of network-based IDS is to detect suspicious activity regarding network, transport and application protocols. Network-based IDSs are often placed at the boundaries between different networks, near to border firewalls or routers. When a network-based IDS detects suspicious activity, it generates an alert with some attributes, such as timestamp, alert type, rating, source and destination IP addresses, source and destination ports, and protocols of network, transport and application layers [Scarfone and Mell 2007].

The alert logs that IDSs generate can be preprocessed, correlated and analyzed to extract actionable information. Then, it may provide a better situational awareness of the monitored system, helping the response to an incident and the improvement of the system security.

2.2. IDS Detection Methods

IDSs detect security violations using different methods, such as signature-based, anomaly-based and hybrid.

A signature is the description of a known threat pattern [Scarfone and Mell 2007]. The signature-based detection method attempts to detect possible threats by comparing signatures with observed events [Scarfone and Mell 2007, Mitchell and Chen 2014]. If the signature matches with the observed event, an alert is raised [Debar 2002]. For instance, Snort[1], one of the most popular IDS, has a signature in its database named *ICMP Ping Nmap*. A Snort instance generates an alert with this signature when it detects an ICMP ping with zero data size. This signature indicates a possible attempt of network reconnaissance. The signature-based method is straightforward and efficient to detect known threats [Scarfone and Mell 2007, Liao et al. 2013]. However, it is ineffective to detect new attacks or variants of attacks and must often be updated to include new threats, which takes time to be discovered [Debar 2002, Vacca 2013, Liao et al. 2013].

By presuming malicious activities have a different behavior from the normal system activities, in anomaly-based detection, the IDS learns what behavior is normal for a given system. With this knowledge, it separates the normal behavior from the abnormal behavior. Therefore, whether a deviation from the normal behavior of the system is detected, an alert is generated. This method can detect new threats. However, learning the normal behavior of a system is not a simple task, since the normal behavior can change over time, and anomalies may be only unusual events, not necessarily an intrusion. Hence, anomaly-based IDSs may present high false positive rates [Scarfone and Mell 2007, Debar 2002, Vacca 2013, Mitchell and Chen 2014].

The hybrid detection method integrates the signature-based and anomaly-based detection into a single system, attempting to keep the advantages and overcome the drawbacks of each method. Hybrid IDSs employ the signature matching due to its speed and flexibility and the anomaly-based detection to identify unknown behaviors for further

[1] https://www.snort.org/

analysis [Vacca 2013, Liao et al. 2013].

3. Related Work

Researchers have proposed different methods to assist intrusion alert analysis in recent years. The proposed methods aim to reduce the number of alerts, discover relationships between them and provide a visual representation for the correlated alerts.

Because of the enormous amount of alerts that IDSs trigger, Julisch (2003) proposes an approach to handle the IDS alerts more efficiently. The proposed approach is based on the root-cause detection. A root-cause is the reason for the occurrence of an alert. Julisch claims that few root causes are responsible for 90% of the alerts. A clustering technique is proposed to group the alerts, supporting the root-cause analysis. The author evaluated the proposed approach with real data that was collected from a commercially used network.

Treinen and Thurimella (2006) proposed a framework to discover new attack strategies. Their objective was to feed an Enterprise Security Manager (ESM) with these attack scenarios, allowing the automatic construction of alert correlation rules. First, IDS alerts were grouped according to their related IP addresses using the connected components method. Then, the researchers applied the association rule mining in each group to find non-obvious associations between attack signatures and IP addresses. The results showed the framework could find new association rules with high confidence. Importantly, each experiment was conducted with real data composed of alerts that IDS sensors generated during 24 hours for 135 distinct production networks.

Liu et al. (2010) proposed a model that correlated the alerts into attack scenarios and provided a visualization for them. First, the alerts were converted into a common format and the false alerts were filtered. Then, using the alerts IP addresses, they were correlated into three attack scenario types: (a) Process-critical scenario: represents the intrusion steps that one attacker performed against one victim; (b) Attacker-critical scenario: represents the relationship between one attacker and many victims, and (c) Victim-critical scenario: represents the relationship between many attackers and one victim. Finally, the uncovered attack scenarios were represented as graphs. The experiments were performed using experimental data generated from two attacks in an isolated and controlled environment. The results showed that the proposed model could provide a high-level visualization of the alerts.

Lagzian et al. (2012) developed a method to correlate alerts and find the most frequent attack scenarios. The method consisted of three main steps: alerts preprocessing, alerts aggregation and attack scenario extraction. The preprocessing normalized and converted alert data into a universal format. After that, the alerts with related IP addresses were grouped, forming the attack scenarios. Then, the frequent pattern mining technique Bit-AssocRule was applied to discover the most frequent attack scenarios. DARPA 2000 dataset was used to test the proposed method. The results showed it had a good performance and could find the attack scenarios properly.

To improve the quality of intrusion alerts information, Spathoulas and Katsikas (2013) propose a system to process the intrusion alerts that belong to multiple IDSs and provide a high-level presentation of the security events. For this, the alerts that have the

same source and destination IP addresses, same attack identifier and are close in time are grouped. Then, the similar groups are clustered, and for each cluster, a visual representation is provided. The proposed system was evaluated using the DARPA 2000 dataset and an academic dataset generated by attack simulation.

Ghasemigol and Ghaemi-Bafghi (2015) proposed a system that correlates intrusion alerts using their entropy. The goal of the proposed system is to find causal relationships between the intrusion alerts, providing a high-level representation of the attack scenario. First, the number of alerts is reduced using data aggregation. Data aggregation is applied to group the alerts in clusters and the graphs of the clusters are generated for visualization. The authors reached a rate of 99.98% of data reduction using the DARPA 2000 dataset to evaluate the proposed system.

Works of Liu et al. (2010), Lagzian et al. (2012), Spathoulas and Katsikas (2013), and Ghasemigol and Ghaemi-Bafghi (2015) performed their tests using experimental datasets. However, experimental datasets (e.g. DARPA 2000 dataset) may show a low capacity of representing a real environment, and this fact has been widely criticized [McHugh 2000, Brown et al. 2009, Shiravi et al. 2012, Zuech et al. 2015]. Unlike many approaches in the alert correlation field, this work is evaluated with a real world dataset, generated by an IDS deployed in a large scale network at the University of Maryland.

Considering that one attacker performs multiple attack steps to reach its goal, Xuewei et al. (2014) proposed an approach to automatically identify the multistage attacks in intrusion alerts. First, alerts are clustered by related IP addresses using the method of connected components. Then, for each cluster, a directed graph that represents the attacker steps is generated. Each node in the graph represents an attack type, and each edge contains the probability of an attack type follows another attack type. The connected component method used by Xuewei et al. (2014) does not deal with the alerts timestamp. Therefore, alerts that are very distant in time can be grouped into the same attack scenario. Moreover, the algorithm proposed by the authors does not consider to merge two connected components if the source and destination IP addresses of an alert under analysis are already contained in these connected components. In this work, we apply the connected component method taking into consideration these two issues.

Alvarenga et al. (2015) proposed an offline approach which applies process mining techniques to analyze past IDS alerts and discover attack strategies. The main goal is to provide an intuitive visual representation of the discovered attack strategies. The proposed approach was evaluated using a real dataset from the University of Maryland. The results showed that the proposed approach could combine visual features with quantitative measures to assist the analysis of the past IDS alerts. As well as the work of Alvarenga et al. (2015), this work has an offline phase to extract the attack strategies from the past IDS alerts. Additionally, we also present an online phase which uses the information observed in the offline phase to assist the analysis of the upcoming IDS alerts.

Shittu et al. (2015) proposed a framework for intrusion alert analysis. The proposed framework reduces the quantity of false positive alerts and uses correlation methods to generate hyper-alerts, referred to as correlated alert graphs. A metric for prioritizing hyper-alerts and a method to cluster the similar hyper-alerts using DBSCAN were presented. Graph Edit Distance was used to compute the similarity between the correlated

alert graphs. The dataset used to test the framework belongs to the 2012 cyber range experiment that industrial partners of the British Telecom Security Practice Team performed. As seen in Section 4, the Graph Edit Distance technique presented some performance issues when applied to our attack strategy graphs. Therefore, in this work we propose a new method based on the Jaccard index to measure the similarity between two attack strategy graphs.

This work proposes a new offline correlation method that organizes the past alerts and identifies the attack strategies that the attackers adopted. Then, the similar attack strategies are grouped using agglomerative hierarchical clustering with Ward's method. A new approach based on Jaccard index was developed to measure the similarity between two attack strategies. Each cluster of similar attack strategies can represent an attack strategy pattern.

Furthermore, a new online correlation method is proposed. The online correlation method generates or updates hyper-alerts as the IDS triggers new alerts. Each hyper-alert comprises new attributes that help the security analyst to understand the attacks. Following the hypothesis that an attack strategy used in the past may repeat in the future, each hyper-alert is associated with one attack strategy pattern found by the offline correlation method. It is important to emphasize that a real dataset was used to evaluate the correlation method.

4. Proposed Approach

To address the main challenges related to intrusion alert analysis, i.e., the huge amount of alerts generated by IDSs, and the low-level information provided, this work proposes a new approach to generate online hyper-alerts based on the attackers past behavior. The proposed approach is composed of two correlators: the offline correlator and the online correlator. The offline correlator receives as input a set of past IDS alerts and identifies the attack strategy patterns to be used in the online correlation. The online correlator analyzes the incoming alerts in real time and generates hyper-alerts that gather information about groups of correlated alerts. Furthermore, considering the hypothesis that an attack strategy adopted in the past may be employed again in the future, each hyper-alert is associated with an attack strategy pattern that the offline correlator identified. These two correlators are described in the next subsections.

4.1. Offline Correlator

The offline correlator aims to organize a set of historical IDS alerts into clusters, giving a better understanding of the attacks. For this, the first step consists of separating the alerts using the connected component method, identifying the attack scenarios (aggregation step). Then, the attack strategy of each scenario is extracted and represented by attack strategy graphs. Considering the possibility of many attack strategy graphs be similar, they are grouped into clusters using hierarchical clustering techniques (clustering step). Each cluster represents one attack strategy pattern. Figure 1 presents an overview of the offline correlator. This process is detailed next.

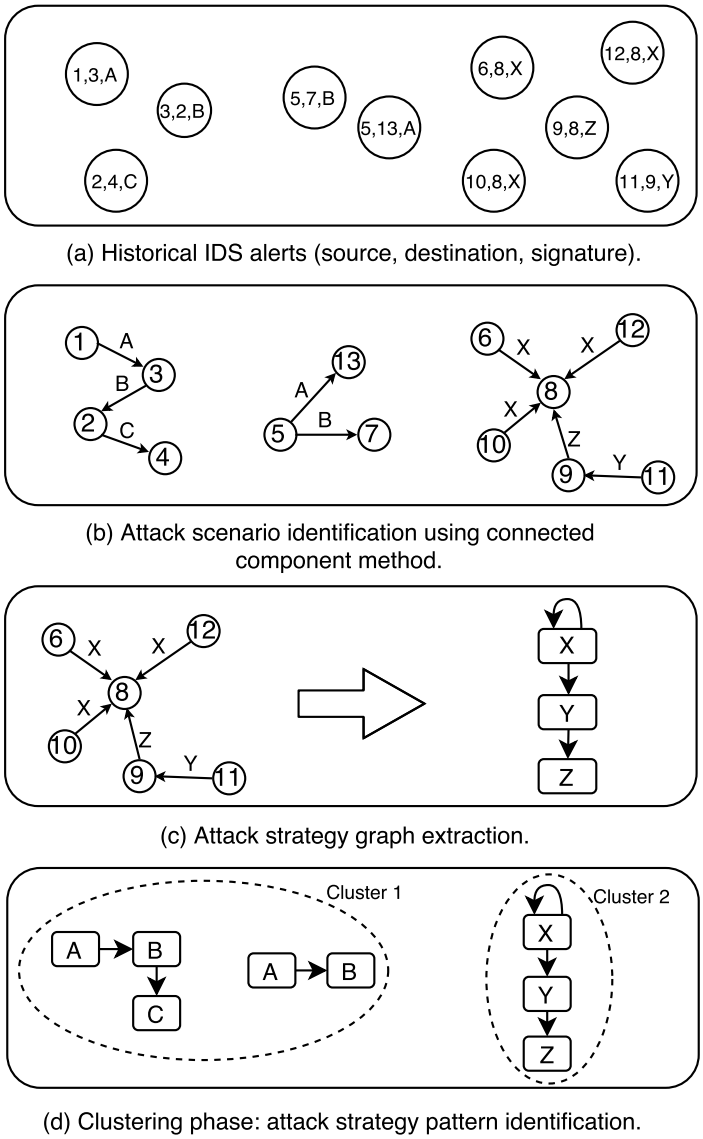

Figure 1. Offline correlator overview.

Let s, d, t, k be the source IP address, the destination IP address, the timestamp and the attack signature of an alert, respectively. Then, an alert A is defined as the 4-tuple $A = \langle s, d, t, k \rangle$. The offline correlator uses the IP address and timestamp information to organize the alerts. This organization is based on the method that Treinen and Thurimella (2006) used in their work, referred to as the connected component method. In the aggregation step, the connected component organization separates the alerts by their IP addresses. If the alerts have related IP addresses, i.e. $A_1.s = A_2.s$ or $A_1.d = A_2.d$ or $A_1.s = A_2.d$ or $A_1.d = A_2.s$, then they are grouped. In this work, another condition must be satisfied to group the alerts: the alerts must have a difference of timestamps shorter than x minutes, i.e. $A_2.t - A_1.t < x$ minutes. Each group built by the connected component method represents one attack scenario, in which all alerts have related IP addresses and are close in time.

A connected component is a visual representation of an attack scenario and is

defined as a directed graph $CC = (V, E)$, where the set of vertex V represents the IP addresses, and the directed edges E represents the direction of the attack (from source to destination).

As a first example, consider the set of alerts presented in Table 1. Two attack scenarios can be identified using the connected component separation method. Figure 2 shows the visual representation of each attack scenario.

Table 1. First example of IDS alerts.

Time	Source	Destination	Signature
01/01/2016 00:30:00	10.0.0.0	10.0.0.1	A
01/01/2016 00:35:00	10.0.0.1	10.0.0.2	B
01/01/2016 00:40:00	10.0.0.1	10.0.0.3	C
01/01/2016 00:45:00	10.0.0.64	10.0.0.65	D
01/01/2016 00:50:00	10.0.0.64	10.0.0.65	D
01/01/2016 00:55:00	10.0.0.64	10.0.0.65	E
01/01/2016 01:00:00	10.0.0.64	10.0.0.65	F

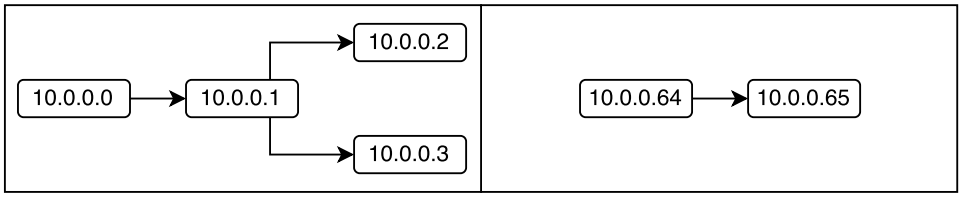

Figure 2. Connected components for the first example of IDS alerts.

Another example, from Table 2, highlights the importance of the condition related to the difference of timestamps of alerts with related IP addresses. All of the alerts in Table 2 have related IP addresses, but three of them were recorded hours after the first ones. Considering a maximum time of 60 minutes for the difference of timestamps of alerts with related IP addresses, the first three alerts compose one attack scenario, and the last three alerts compose another attack scenario, illustrated in Figure 3.

Table 2. Second example of IDS alerts.

Time	Source	Destination	Signature
01/01/2016 00:00:00	10.0.0.0	10.0.0.1	A
01/01/2016 00:40:00	10.0.0.1	10.0.0.2	B
01/01/2016 01:20:00	10.0.0.1	10.0.0.3	B
01/01/2016 10:00:00	10.0.0.0	10.0.0.4	A
01/01/2016 10:40:00	10.0.0.4	10.0.0.5	B
01/01/2016 11:20:00	10.0.0.4	10.0.0.6	B

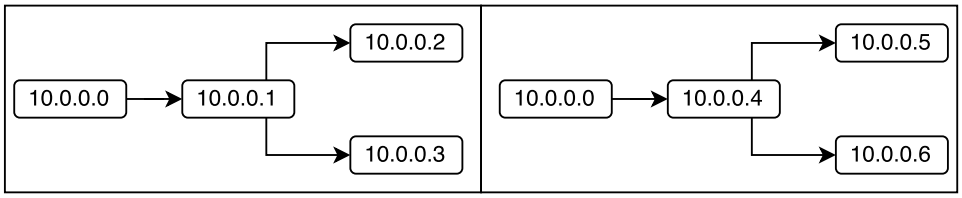

Figure 3. Connected components for the second example of IDS alerts.

Using the alerts signature and timestamp makes it possible to determine the sequence of signatures triggered in each attack scenario. The sequence of signatures is represented in a directed graph format, where each node represents one signature, and each edge represents the sequential relationship between the signatures. This graph is named attack strategy graph. One attack strategy graph is generated for each attack scenario. Figure 4 shows the attack strategy graphs created for each attack scenario of the alerts from Table 1. Figure 5 shows the attack strategy graphs created for each attack scenario of the alerts from Table 2.

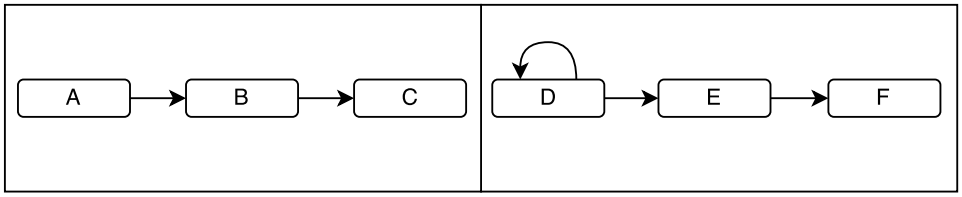

Figure 4. Attack strategy graphs of the alerts in Table 1.

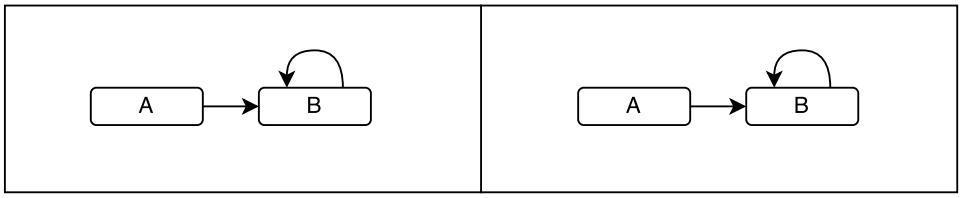

Figure 5. Attack strategy graphs of the alerts in Table 2.

The next step of the offline correlator is to group similar attack strategy graphs. For this, a method to verify if two graphs are similar (or dissimilar) should be defined. Ning and Xu (2003) calculated the similarity between two attack strategy graphs using the Graph Edit Distance technique. The Graph Edit Distance technique measures how many edit operations (node insertion, deletion or substitution, and edge insertion, deletion or substitution) are required to transform one graph into another. The more edit operations are needed, more dissimilar are two graphs.

However, this method showed to be inadequate to our attack strategy graphs due to its performance when applied to graphs with many nodes or edges. Also, for smaller graphs, the Graph Edit Distance is low even if the graphs are completely different, because few edit operations are necessary. Figure 6 demonstrates this problem: the cost for both situations will be one node substitution (X for Y), even taking into account that the situation (a) shows similar nodes (A and B). Therefore, if two graphs have a lot of

similar edges and nodes and only one different node, the Graph Edit Distance technique will consider only the different node.

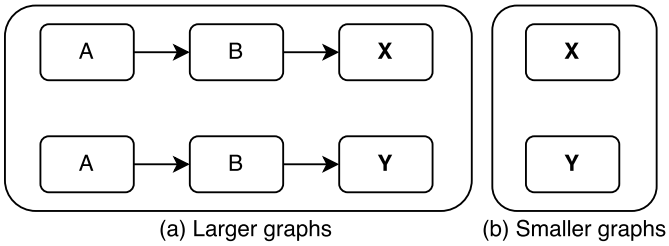

Figure 6. Example of attack strategy graphs for Graph Edit Distance and for the new method based on Jaccard index.

Following this reasoning, we propose a new method to measure attack strategy graph similarity. Attack strategy graphs are represented by a set of signatures (nodes), and a set of relationship between signatures (directed edges). Therefore, the proposed method is based on Jaccard index, which measures the similarity between two finite sets. Considering M and N two finite sets, the Jaccard index between M and N is defined as the size of the intersection of M and N, divided by the size of the union of M and N (Equation 1) [Niwattanakul et al. 2013]. To measure the similarity between two attack strategy graphs, two Jaccard indexes were calculated: first one Jaccard index J_1 for all nodes, then another Jaccard index J_2 for all pair of nodes (representing the edges). To give the same similarity influence for the nodes and edges, the mean of the two results was considered the similarity S between two attack strategy graphs (Equation 2).

For the attack strategy graphs of the situation (a) in Figure 6, the similarity is calculated as follows:

1. First, it is calculated the similarity J_1 of nodes, which is the size of the intersection of the nodes, i.e. $\{A, B\}$, divided by the size of the union of nodes, i.e. $\{A, B, X, Y\}$. Therefore, $J_1 = 2/4 = 0.5$.
2. Then, it is calculated the similarity J_2 of the edges, which is the size of the intersection of the edges, i.e. $\{A \rightarrow B\}$, divided by the size of the union of edges, i.e. $\{A \rightarrow B, B \rightarrow X, B \rightarrow Y\}$. Therefore, $J_2 = 1/3 = 0.3$.
3. The final similarity S is the mean between J_1 and J_2, which is 0.4.

$$J(M, N) = \frac{|M \cap N|}{|M \cup N|} \tag{1}$$

$$S = \frac{J_1 + J_2}{2} \tag{2}$$

It is possible to have an empty set of edges (a graph with only one node). If both compared graphs present empty edges, the final similarity S is simply defined by J_1 (see Equation 3). If both graphs are composed of only one node, the similarity between them must be 1 if the nodes are equal, and 0 if the nodes are different.

$$S = J_1 \tag{3}$$

With this new metric, the problem presented in Figure 6 is solved, because all nodes and their edges are taken into consideration. Also, this metric does not present performance problems, as observed with the Graph Edit Distance.

Using this new metric, all attack strategy graphs can be compared to generate a similarity matrix. The similarity matrix is a square matrix with size $n \times n$, where n is the quantity of attack strategy graphs and the content of the matrix is the similarity (from 0 to 1) between two graphs.

Figure 7 shows an example of a similarity matrix, which indicates the similarity between the graphs $G1$, $G2$, $G3$ and $G4$. Analyzing the content of this similarity matrix, we can realize that $G1$ and $G2$ are similar to each other and different from $G3$ and $G4$.

	G1	**G2**	**G3**	**G4**
G1	1	0.83	0	0
G2	0.83	1	0	0
G3	0	0	1	0.32
G4	0	0	0.32	1

Figure 7. Example of similarity matrix.

The similarity matrix is used as input to the agglomerative hierarchical clustering using Ward's method [Ward Jr 1963, Macfarlane 1996]. Considering z_{ik} the set of elements of a cluster k (which have n_k elements), and considering $\bar{z}_k$ the mean of the elements, the sum of squared errors for cluster k is given by the Equation 4.

$$E_k = \sum_{i=1}^{n_k} \parallel z_{ik} - \bar{z}_k \parallel^2 \tag{4}$$

Considering E_k defined by the Equation 4, the Ward's method calculates the total error sum of squares E for all g clusters using the Equation 5.

$$E = \sum_{k=1}^{g} E_k \tag{5}$$

The agglomerative hierarchical clustering method begins with n clusters that contain a single element (where n is the number of elements). Then it performs $n-1$ merging steps. At each step, the two clusters that give the smallest increase in the total error of the clusters are merged. The method ends with one cluster that contains all n elements. As a

result, this method outputs a dendrogram, which is a tree that represents the entire clustering process (Figure 8). The cutting height of the dendrogram that defines the number of clusters is problem dependent [Jain et al. 1999, Xu and Wunsch 2005]. In this work, the cutting height is defined as the value of the third quartile of the sorted set of heights, which separates the 75% lower heights from the 25% higher heights [Kerns 2011].

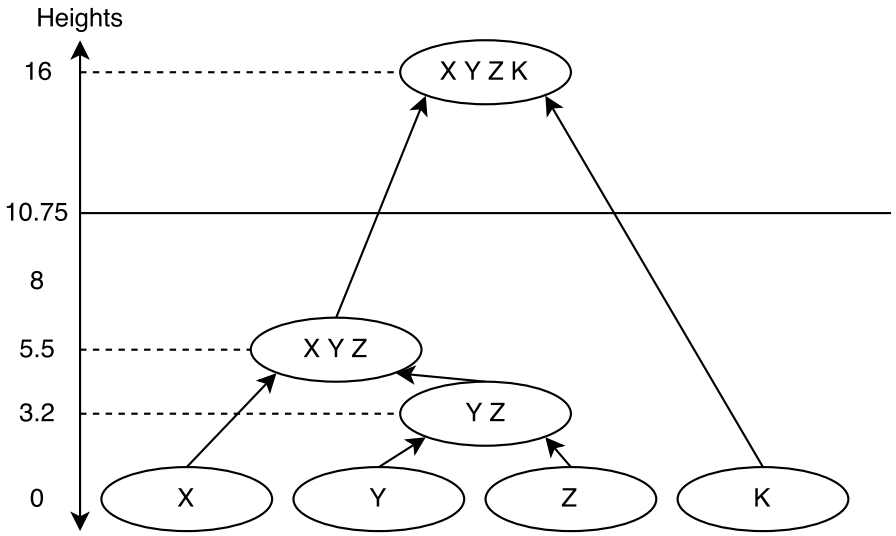

Figure 8. Example of dendrogram for agglomerative hierarchical clustering using Ward's method.

Figure 8 illustrates an example of the agglomerative hierarchical clustering technique, where each height represents the square root of the total error of the clusters after a merge between two clusters (calculated with Ward's method) [Macfarlane 1996]. In this example, the number of elements is four. Therefore, three $(n - 1)$ merging steps were required. The first merging step groups the clusters Y and Z, creating the new cluster YZ. The second merging step groups the clusters X and YZ, creating the new cluster XYZ. The last merging step groups the clusters XYZ and K, creating the new cluster $XYZK$. The dendrogram has been cut in the height 10.75 (third quartile of the heights 0, 3.2, 5.5 and 16). To compute the third quartile, at first, we applied the median to divide the ordered heights into two halves. Then, we applied the median in the upper half to find the third quartile value. Thus, two clusters were generated: the first cluster is composed of the elements X, Y and Z and the second cluster is composed of the element K. In our work, each element is an attack strategy.

After the offline correlation, we have all historical alerts separated into clusters according to the relationships between attack steps. Each cluster groups similar attack strategies and provides a summarized overview of them. The clusters information can be used to support the analysis of the upcoming alerts by the online correlator.

4.2. Online Correlator

Using the connected component separation method, the online correlator separates the alerts into attack scenarios as the IDS triggers them. For each attack scenario, one hyper-alert is generated. A hyper-alert contains many attributes to describe its attack scenario.

Figure 9 shows an hyper-alert A describing the attack scenario A. Table 3 lists and explains all hyper-alert attributes. The first column represents the category of the attributes, the second column represents the attributes of each category, and the third column describes each category. These attributes assist the security management by giving a better insight of each current attack scenario. As this correlator operates online, an attack scenario can receive new alerts constantly. Therefore, the attributes of an hyper-alert are updated whenever a new alert is added to the attack scenario it represents.

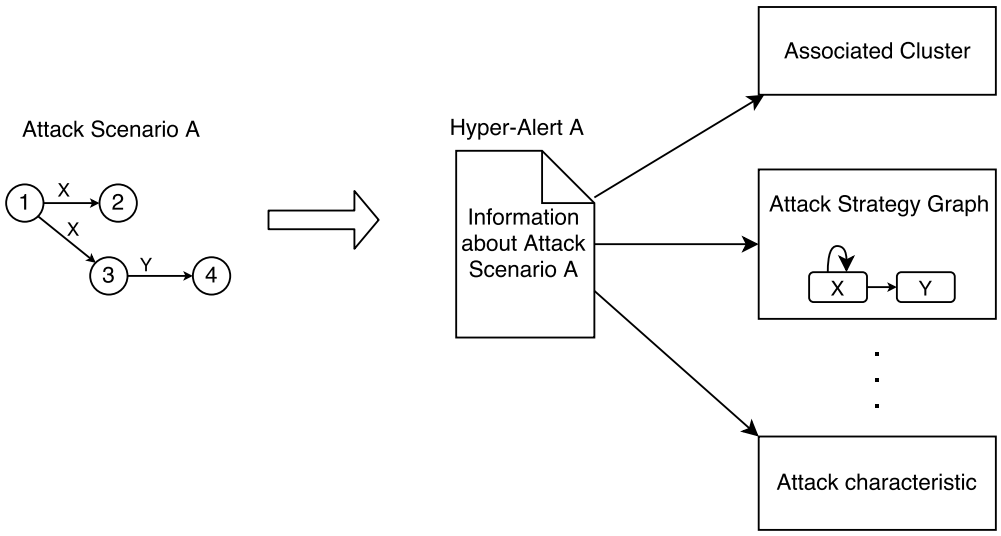

Figure 9. Hyper-alert A describing the attack scenario A.

One difference between the online and the offline correlator is that the online correlator process the alerts as they are generated in real time. Therefore, for each new alert A, it is verified if A belongs to an existing attack scenario comparing the IP addresses and timestamp (otherwise, A will belong to a new attack scenario). After determining to which attack scenario the new alert belongs, the attack strategy graph of this scenario is updated with the new alert A. Using Ward's method (Equation 5), the online correlator associates the updated attack strategy graph G with one of the attack strategy clusters that the offline correlator found.

For example, Figure 10a illustrates a new alert A with source IP represented by the number 21, destination IP represented by the number 15 and signature Y. Figure 10b shows the inclusion of this alert in the proper attack scenario. Figure 10c demonstrates the attack strategy graph being updated with the new signature Y. Finally, Figure 10d shows the association of the updated attack strategy graph with the Cluster 2 (found by the offline correlator).

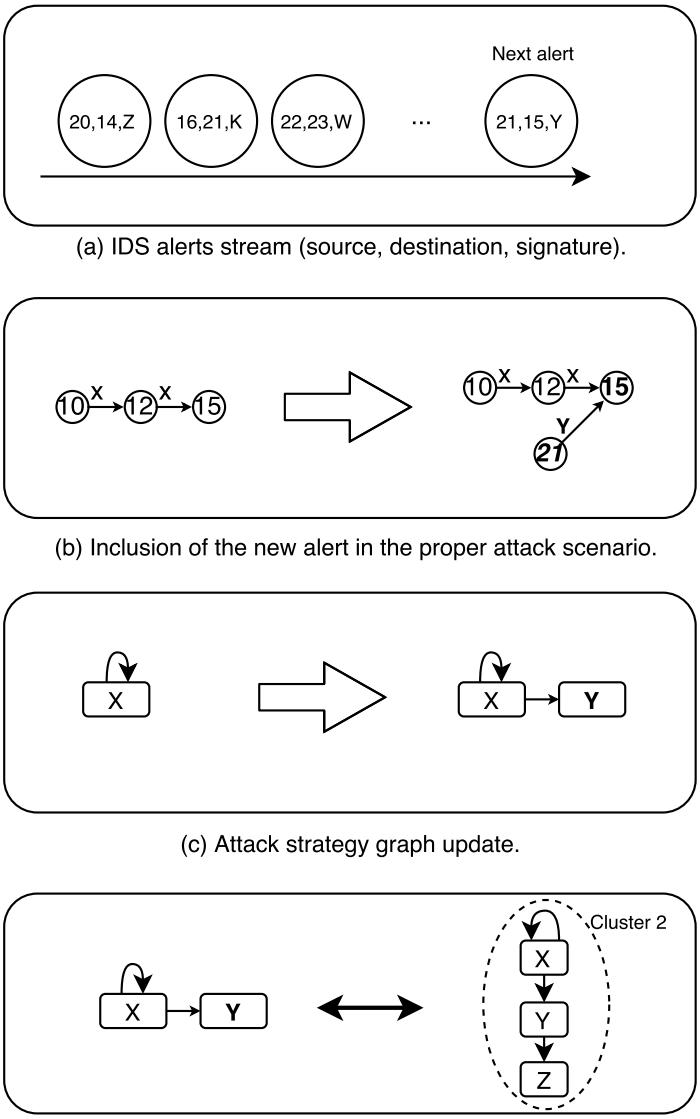

Figure 10. Online correlator overview.

Associating the updated attack strategy graph to a cluster provides useful information for the security analyst, since each cluster contains information about typical attack strategies that attackers have adopted against the network. This way, this association allows anticipating the nature of the current attack by analyzing the attack strategy pattern that the offline correlator found earlier.

As a result, the online correlator outputs hyper-alerts that describe each current attack scenario. Also, the hyper-alert information is updated in real time as it receives a new low-level alert. With this new information, the security analyst has a better situational awareness of the environment in real time to react faster to a security event.

Table 3. Hyper-alert information.

Attribute Category	Attributes	Description
Status information	Status	The hyper-alert is closed if no alert is added in x minutes. Otherwise, it is under construction. If the hyper-alert is closed, it means that it will not receive more alerts. Then, this is the final state of the hyper-alert
Cluster information	Associated cluster and increased error	Reports the associated cluster for the hyper-alert and the new error of the cluster, considering that the attack strategy graph of this hyper-alert has been added to this cluster (using Ward's method). The bigger the error, the greater the difference between the cluster and the attack strategy graph
Time information	Start time, end time and duration	Reports the timestamp of the first and the last alert of the hyper-alert and the difference in minutes between these timestamps
Quantitative information	Quantity of alerts, quantity of distinct attackers, quantity of distinct targets and quantity of distinct signatures	Counts the number of alerts, attackers, targets and signatures that compose the hyper-alert
Address information	The number of alerts triggered per attacker IP address and the number of alerts triggered per target IP address	This information is useful to determine which IP addresses trigger more alerts and which IP addresses receive more attacks
Signature information	The number of alerts with each signature and the attack strategy graph	This information is useful to determine which signature happens more often and to understand the causal relationship between these signatures
Relationship information	The connected component and a list of relationships	One relationship consists of one source IP address, one destination IP address, and the number of times that these IP addresses appear in the same alert of the attack scenario. This information is useful to understand the relationship of the IP addresses
Attack characteristic information	The mean of the targets per attacker and the mean of attackers per target	This information is useful to understand the characteristic of the attack. For example, if the mean of targets per attacker is high, it may indicate a reconnaissance attack. Alternatively, if the mean of attackers per target is high, it may indicate a DDoS attack

5. Experiments and Results

The IDS alerts used in the experiments were recorded in July 2012 by a commercial network-based IDS deployed in a network with about 40,000 computers from the University of Maryland. Two experiments were performed. The first experiment (presented in Section 5.1) uses all alerts of July 2012, the first half was applied in the offline correlator and the second half was applied in the online correlator. The aim of the first experiment is to analyze the general behavior of the proposed correlator. Therefore, it describes the main clusters the offline correlator identified and the main hyper-alerts the online correlator generated. On the other hand, the second experiment (presented in Section 5.2) investigates how the size of the alert sample used in the offline correlation affects the online correlation results.

5.1. First Experiment – General experiments

The alerts triggered in the first 14 days of July 2012 were the input of the offline correlator to generate the cluster model of attack strategies. The alerts triggered in the other days of the month (from 15th to 31st of July 2012) were used to test the online correlator. The maximum time difference between two alerts of an attack scenario was defined as 60 minutes for the offline and online correlator.

For the offline correlator, a total of 20,509 alerts were used as input. The aggregation step separated these 20,509 alerts into 895 attack scenarios. Then, the attack scenarios that represent exceptional situations were filtered. An attack scenario was considered as an exceptional situation if it contains only one alert or contains only alerts with the same signature. These cases do not depict attack strategies used by attackers, as they show an attack flow with a single step and do not provide enough information on the behavior of the attacker. With this filtering, 3,796 alerts were removed, with 16,713 remaining alerts. 57 attack scenarios represent these 16,713 alerts. For each one of these 57 attack scenarios, the attack strategy graph was generated, and the similarity matrix was computed using the proposed metric based on Jaccard index. Finally, the 57 attack strategy graphs were separated into clusters by the hierarchical clustering technique.

A total of 12 clusters were generated, representing 12 attack strategy patterns. Therefore, the problem of analysing 16,713 low-level alerts was simplified into the analysis of 12 meaningful clusters. The two clusters (Cluster 1 and Cluster 7) that represent the most of the alerts were chosen to be further explained in this experiment. Cluster 1 covers 14,138 alerts (84.59% of the 16,713 alerts) and has 10 attack scenarios. Cluster 7 covers 2,317 alerts (13.86% of the 16,713 alerts) and has 12 attack scenarios. This means that these two clusters cover 98.46% of the 16,713 alerts.

Cluster 1 is characterized by buffer overflow attacks. Figure 11 shows an example of an attack strategy graph that composes Cluster 1. On the other hand, Cluster 7 is characterized by reconnaissance attacks, such as the port scan Possible Nmap Scan (XMAS (FIN PSH URG)), and the Fingerprinting Probe, which tries to ascertain the operational system of the target. Figure 12 shows an example of an attack strategy graph that composes Cluster 7.

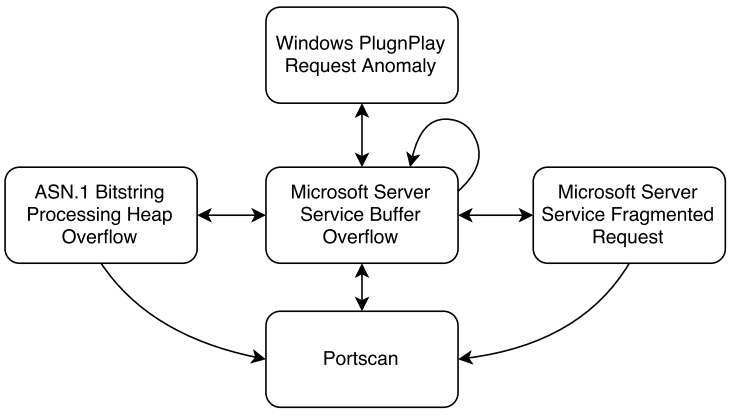

Figure 11. Example of an attack strategy graph of the Cluster 1.

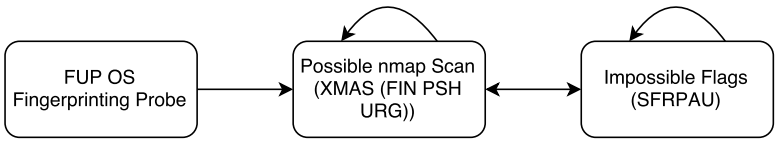

Figure 12. Example of an attack strategy graph of the Cluster 7.

The union of all attack strategy graphs of a cluster is performed to represent an attack strategy pattern visually. Figure 13 shows the attack strategy pattern of Cluster 7.

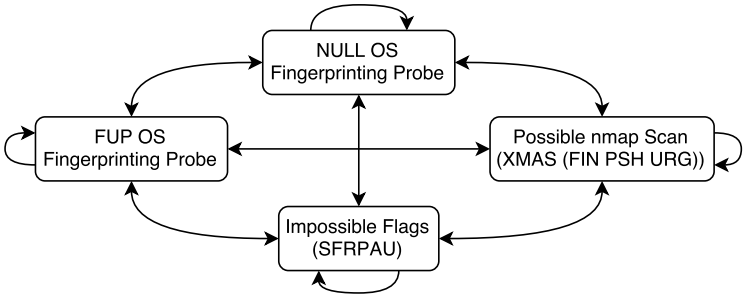

Figure 13. Union of the attack strategy graphs of the Cluster 7.

The alerts triggered From 15th to 31st of July 2012 were used for the online correlator validation. In this period, 19,933 alerts were generated. To test the online correlation, a simulator was developed to release the alerts one by one. As explained in Section 4, for each new alert, it is verified to which attack scenario it belongs. After determining to which attack scenario the new alert belongs, the attack strategy graph of this attack scenario is updated with the new alert. Then, the updated attack strategy graph is associated with one of the clusters generated by the offline correlator.

Therefore, this is a process where each attack scenario (and consequently each attack strategy graph) changes as a new alert is added to it. A new attack strategy graph (updated by only a few alerts) can be associated with a cluster at the beginning and, after receiving a few more alerts, it can be associated with another cluster because the more alerts are used to update the attack strategy graph, the more precise is its association.

A hyper-alert is considered *closed* when no alerts are added to its attack scenario during 60 minutes. After that, no more alerts can be added to this attack scenario, and, hence, to this hyper-alert. The closed hyper-alert is in its final state (its attributes will not receive more updates). The cluster associated with the closed hyper-alert is denominated *final cluster*.

After the experiments, the attack strategy graphs showed fast convergence into their final cluster: we observed that an attack strategy graph is correctly associated with its final cluster after an average of 1.54 associations with a standard deviation of 0.7. The average of associations for an attack strategy graph is 8.73 with the standard deviation of 10.05 (the high standard deviation is due to the difference in the attack strategy graphs size). Each new association is performed after the addition of a new alert that changes the attack strategy graph.

Interesting results were observed after analyzing the online correlator output: 16 out of 59 hyper-alerts were associated with the Cluster 1 and contained between 748 and 1209 alerts. Figure 14 represents one of the 16 hyper-alerts associated with Cluster 1. By analyzing the time information of the hyper-alerts, we observed that these 16 hyper-alerts were built during the night. Also, each one of the 16 hyper-alerts was generated once a day (from 15th to 31st of July 2012), except in the day 29th. Moreover, by analyzing the address and signature information of the hyper-alerts, we identified that the same source IP address was responsible for about 70% of these alerts, which have the same signature. The security analyst could have identified this pattern in the first days by using the proposed approach. Since this problem is responsible for a significant amount of alerts, the number of future alerts could have been reduced with the mitigation of this situation.

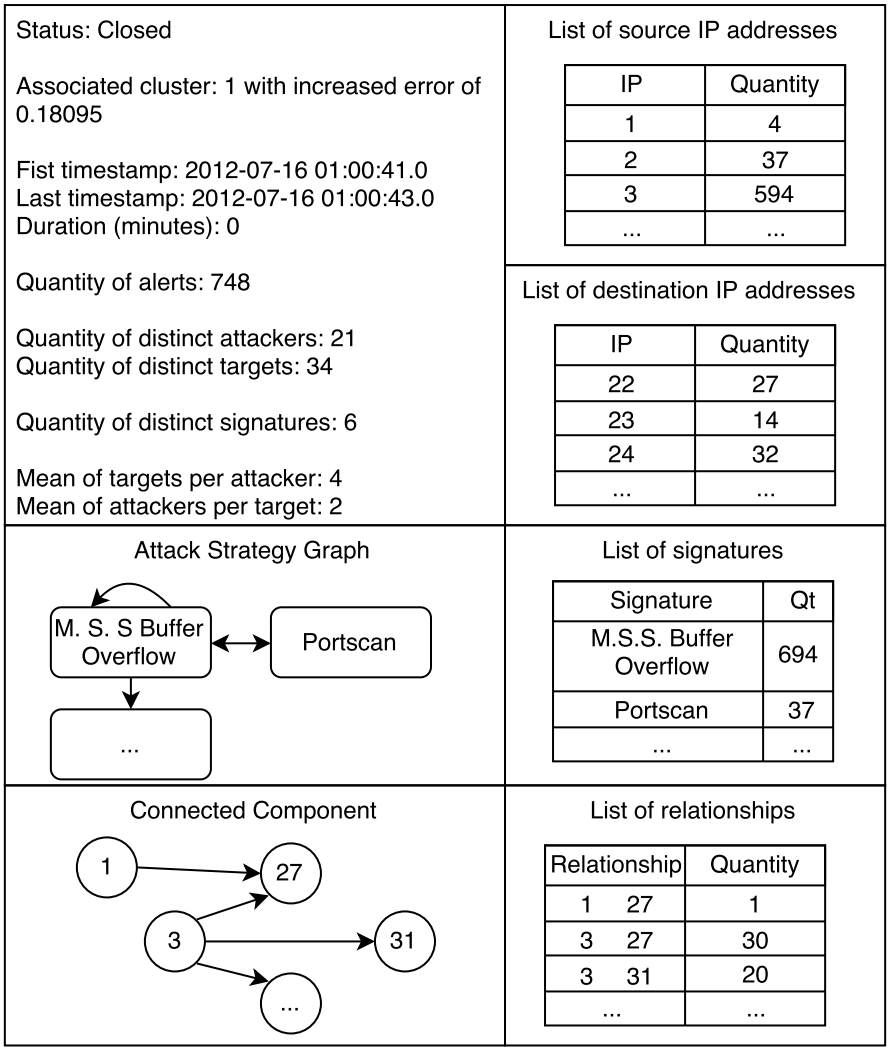

Figure 14. Example of hyper-alert associated with Cluster 1.

13 hyper-alerts were associated with the Cluster 7 and had between 6 and 32 alerts. A significant pattern found in each of these hyper-alerts is the presence of a single source IP address attacking multiple destination IP addresses. Figure 15 demonstrates

the evolution of a hyper-alert associated with the Cluster 7 by showing its quantity of alerts, best cluster, connected component and attack strategy graph. The IP addresses of the connected components were changed to the labels from A to H for privacy purposes. When the hyper-alert had only three alerts, it was associated with the Cluster 2. After adding a new alert, the association of the hyper-alert changed to Cluster 7 and remained on it until the end (when the hyper-alert was closed with seven alerts). This hyper-alert shows an interesting pattern when its connected component is investigated: only one source IP address is attacking other IP addresses. The mean of targets per attacker is 7, which reinforces the characteristic of a reconnaissance attack. The attack strategy graph of this hyper-alert is very similar to the attack strategy graph of the Cluster 7 (Figure 13). Therefore, with 4 alerts, it was possible to infer that this hyper-alert had the same behavior of other attacks on the Cluster 7.

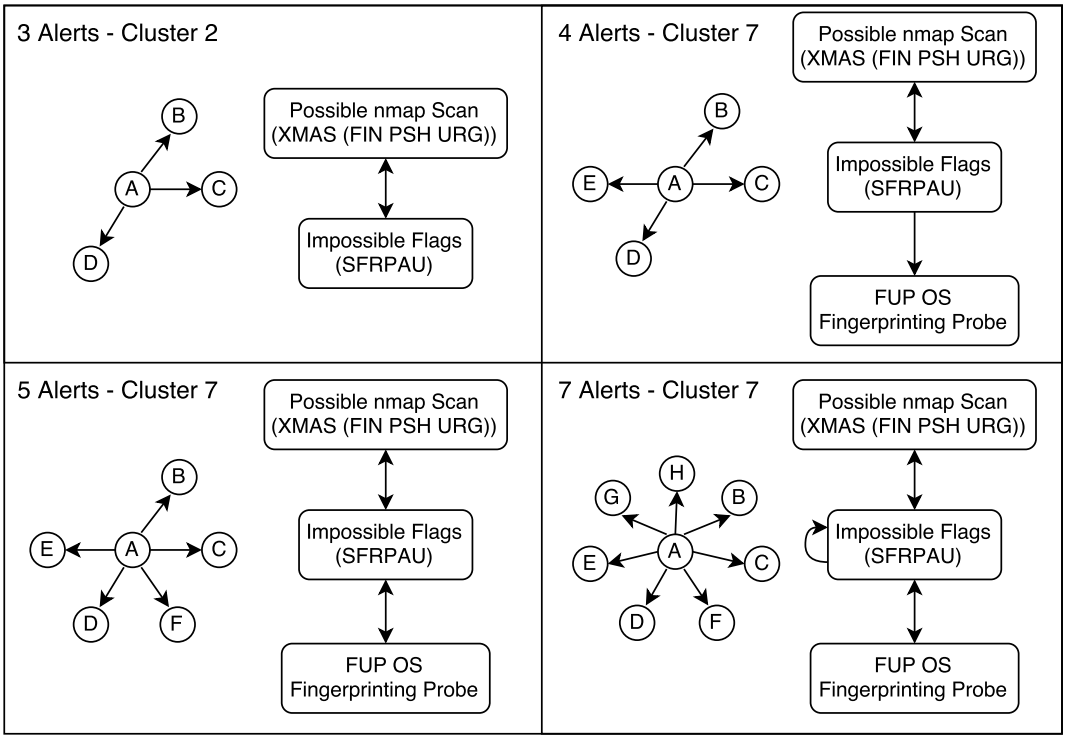

Figure 15. Evolution of one hyper-alert associated with the Cluster 7.

Moreover, 74.58% of the hyper-alerts (44 of 59) presented association error lower than 0.39. The lower the error, the better the similarity between the attack strategy and the associated cluster. Figure 16 presents an example of an attack strategy associated with the Cluster 7 with 0.42 association error. The attack strategy of the Figure 16 resembles the Cluster 7 (Figure 13) even with association error higher than 0.39. These 44 hyper-alerts covers 95.52% of the alerts (13468 of 14099 alerts). This result indicates that 74.58% of the hyper-alerts (consequently 95.52% of the alerts) have similar attack strategies to the attack strategy patterns identified by the offline correlator. Therefore, the hypothesis that a previous attack strategy has good chances to repeat in the future is reinforced.

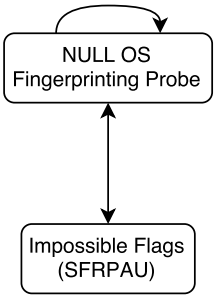

Figure 16. Attack strategy graph associated with Cluster 7 with 0.42 association error.

The offline correlator was able to organize the historical intrusion alerts into clusters, simplifying the problem of analyzing many low-level alerts into the analysis of few meaningful clusters. These clusters show the summarized information for each attack pattern found in the related alerts and they were useful to support the online correlator. Two main clusters that represent the majority of the alerts (98.46%) were identified, which means that the majority of the attacks directed to this network are similar. The online correlator was able to correlate the alerts in real time, organizing them into hyper-alerts with meaningful information that may assist the response to security events. Through the manual analysis of the hyper-alerts, we were able to find interesting patterns and understand the attacks behavior, what would not be possible through the manual analysis of the raw alerts. Also, we were able to understand the attack scenarios with only their first alerts by analyzing the associated cluster of a given hyper-alert.

5.2. Second Experiment – Variations of sample size for offline correlation

This experiment aimed to investigate how the variations in the size of the alert sample for offline correlation, affect the results of the online correlation.

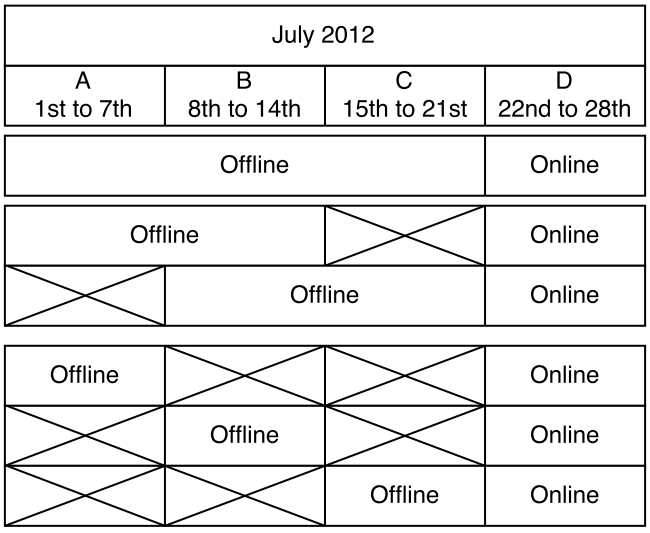

Figure 17. Organization of the IDS alerts from July 2012 for each test.

In the beginning, six tests were performed. Figure 17 shows how they were organized: the alerts generated in July 2012 were separated into four weeks, i.e. A, B, C and

D. In all six tests, the last week (D) was used in the online correlator to achieve a fair comparison between all tests.

For the offline correlator, the test $1a$ used three weeks (A, B, C). The test $1b$ and $1c$ used two weeks: (A, B) and (B, C) respectively. The other three tests used one week: (A), (B) and (C) respectively. This organization was intended to test how the amount and the age of alerts (used in the offline phase) affect the association error in the online phase.

As an example, the test $1a$ was performed as following: the 23,121 alerts triggered in the weeks A, B and C were used in the offline correlator to uncover the attack strategy patterns that the attackers adopted in this period. The 23,121 alerts were separated into 91 attack scenarios using the connected component method, and the 91 attack scenarios generated 18 clusters of similar attack strategies. Therefore, 18 attack strategy patterns were identified in these three weeks. Then, the week D was used in the online correlator. The week D has 5,908 alerts distributed into 10 attack scenarios, and consequently 10 hyper-alerts. Using the Ward's method to compute the increased association error (presented in Section 4.2), each one of the 10 hyper-alerts was associated with one attack strategy pattern. A low error indicates that the hyper-alert could be properly matched to one of the 18 attack strategy patterns. The mean of the 10 association errors (one for each hyper-alert) and their standard deviation were computed. This entire process was performed for all six periods, generating the results of Table 4.

Table 4. Results of the six tests.

Test	Weeks Used for Offline Phase	Alerts Count	Attack Scenarios Count	Clusters Count	Mean of Errors	Standard Deviation of Errors
1a	ABC	23,121	91	18	0.15	0.08
1b	AB	16,713	57	13	0.10	0.07
1c	BC	12,659	65	16	0.12	0.08
1d	A	10,462	26	7	0.11	0.08
1e	B	6,251	31	10	0.11	0.07
1f	C	6,408	34	9	0.13	0.09

Table 4 shows the results for all six tests. For each test, this table shows the test name, the total amount of alerts used, the amount of attack scenarios constructed by the connected component method, and the number of clusters in which those attack scenarios were distributed in the offline correlator. The last two columns indicate the mean and standard deviation of association errors for the hyper-alerts in the online correlator.

The biggest mean of association error was noticed in the test $1a$, using all three weeks (A, B, C) as input for the offline correlator. It means that adopting a longer period for offline correlation does not necessarily generate lower association errors between the hyper-alerts and attack strategy patterns.

Moreover, analyzing the last three tests ($1d$, $1e$ and $1f$), the test that used only week A (older alerts) showed a smaller error than the test that used only week C (with more recent alerts). Therefore, we observed that using more recent alerts in the offline correlation do not necessarily give better results.

It is noteworthy that the error differences between all six tests are very low. This

happened because the online correlator generated 7 hyper-alerts with the buffer overflow pattern and 3 hyper-alerts with the reconnaissance pattern. Both attack strategy patterns were present in the weeks A, B or C. Therefore, none hyper-alert in the week D had a novel attack strategy pattern and all the 10 hyper-alerts were properly associated with an attack strategy pattern with low association error, regardless the week used for offline correlation.

Due to the low association errors for the six previous tests, we performed two new tests ($2a$ and $2b$), using only one or two days instead of weeks. The aim of these two new tests is to use a small sample of alerts and verify the influence of the age of alerts in the association error. Test $2a$ was performed using the first day of July 2012, totalizing 1,917 older alerts. Test $2b$ was performed using the days 20th and 21st of July 2012, totalizing 1,820 more recent alerts. The days from 22nd to 28th were used in the online correlator (same days of week D in the previous tests).

Table 5 shows the results for both tests. We observed that both tests identified the buffer overflow and reconnaissance attack patterns in their clusters. Therefore, even with a smaller sample of alerts in the offline correlation than in tests $1a$ to $1f$, the results still presented low association error (0.2 for the first day and 0.14 for the days 20th and 21st). Furthermore, even with a greater amount of alerts, the test $2a$ (with older alerts) presented higher association error when compared to the test $2b$ (with newer alerts).

Table 5. Results of the tests with days.

Test	Days Used for Offline Phase	Alerts Count	Attack Scenarios Count	Clusters Count	Mean of Errors	Standard Deviation of Errors
2a	1st	1,917	6	3	0.20	0.07
2b	20th and 21st	1,820	11	4	0.14	0.04

Thereby, with a much smaller sample of alerts, the offline correlator was able to identify enough attack strategy patterns to properly anticipate the nature of all the upcoming hyper-alerts in the online correlator. It was possible due to the repetitive behavior of the attacks on the same network. As the most of the attack strategies are repetitive, two main benefits could be identified. The first one is the great capacity of anticipating the nature of known attacks. The second one is related to the few attack strategies that do not match any attack strategy pattern extracted in the offline phase. These attack strategies can be easily disclosed since they present higher association errors than repetitive ones. It is important to find these attack strategies because they represent unusual or novel attacker behavior, which may pose a significant risk to the monitored systems.

6. Conclusion

To support the information security management, this paper addressed the problem of correlating and analyzing the massive amount of intrusion alerts recorded by an IDS device. To deal with the challenges related to the huge amount of alerts generated by IDSs and the low-level information provided, a new approach to assist the security analyst in the intrusion alert analysis was proposed. The proposed approach is composed of two

correlators: offline and online. The offline correlator can group past IDS alerts into attack scenarios using the connected component separation method. Then, the attack strategy graph of each scenario can be identified, and similar attack strategies are clustered to uncover the attack strategy patterns. For this, a new method to measure the similarity between two attack strategy graphs was proposed. The online correlator separates the alerts into attack scenarios as the IDS triggers them. To describe each attack scenario, one hyper-alert with many attributes is generated. The proposed approach was evaluated using a real dataset from 2012 provided by the University of Maryland. The results of the first experiment showed that the offline correlator simplified the problem of analyzing many low-level alerts into the analysis of few clusters. Moreover, the online correlator organized the upcoming alerts into a small amount of meaningful hyper-alerts, which are information structures that describe a group of correlated alerts. The results of the second experiment showed that, even with a small amount of data (i.e., data generated in two days), the offline correlator could identify enough attack strategy patterns to support the online correlator in generating hyper-alerts.

As future work, we intend to: (a) evaluate the proposed approach in other datasets extracted from both real and experimental environments; (b) eliminate the necessity of periodical execution of the offline correlator, which is possible using the online correlator to update the offline knowledge base; (c) explore streaming mining techniques, that may help to improve the aggregation step of our approach; (d) explore new statistical methods to completely eliminate the necessity of manual parameters.

7. Acknowledgment

The authors would like to thank Gerry Sneeringer and the Division of Information Technology at the University of Maryland for allowing and supporting the described research.

References

Ahmad, A., Hadgkiss, J., and Ruighaver, A. B. (2012). Incident response teams - challenges in supporting the organisational security function. *Comput. Secur.*, 31(5):643–652.

Alvarenga, S. C., Junior, S. B., Miani, R. S., Cukier, M., and Zarpelao, B. B. (2015). Discovering attack strategies using process mining. In *AICT 2015 : The Eleventh Advanced International Conference on Telecommunications*, pages 119–125.

Brown, C., Cowperthwaite, A., Hijazi, A., and Somayaji, A. (2009). Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–7.

Debar, H. (2002). An introduction to intrusion-detection systems. In *Proceedings of Connect 2000*.

GhasemiGol, M. and Ghaemi-Bafghi, A. (2015). E-correlator: an entropy-based alert correlation system. *Security and Communication Networks*, 8(5):822–836.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323.

Julisch, K. (2003). Clustering intrusion detection alarms to support root cause analysis. *ACM Trans. Inf. Syst. Secur.*, 6(4):443–471.

Kawakani, C. T., Junior, S. B., Miani, R. S., Cukier, M., and Zarpelao, B. B. (2016). Intrusion alert correlation to support security management. In *XII Brazilian Symposium on Information Systems - Information Systems in the Cloud Computing Era*, pages 313–320.

Kerns, G. J. (2011). *Introduction to Probability and Statistics Using R*. Free Software Foundation, first edition.

Lagzian, S., Amiri, F., Enayati, A., and Gharaee, H. (2012). Frequent item set mining-based alert correlation for extracting multi-stage attack scenarios. In *Telecommunications (IST), 2012 Sixth International Symposium on*, pages 1010–1014.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., and Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16 – 24.

Liu, L., Zheng, K., and Yang, Y. (2010). An intrusion alert correlation approach based on finite automata. In *Communications and Intelligence Information Security (ICCIIS), 2010 International Conference on*, pages 80–83.

Liu, Z., Wang, C., and Chen, S. (2008). Correlating multi-step attack and constructing attack scenarios based on attack pattern modeling. In *Information Security and Assurance, 2008. ISA 2008. International Conference on*, pages 214–219.

Macfarlane, P. A. (1996). Kansas geological survey, dakota aquifer program - ward's method.

McHugh, J. (2000). Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294.

Mitchell, R. and Chen, I.-R. (2014). A survey of intrusion detection in wireless network applications. *Computer Communications*, 42(0):1 – 23.

Ning, P. and Xu, D. (2003). Learning attack strategies from intrusion alerts. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 200–209, New York, NY, USA. ACM.

Niwattanakul, S., Singthongchai, J., Naenudorn, E., and Wanapu, S. (2013). Using of jaccard coefficient for keywords similarity. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, page 6.

Patel, A., Qassim, Q., and Wills, C. (2010). A survey of intrusion detection and prevention systems. *Information Management & Computer Security*, 18(4):277–290.

Ruefle, R., Dorofee, A., Mundie, D., Householder, A., Murray, M., and Perl, S. (2014). Computer security incident response team development and evolution. *Security Privacy, IEEE*, 12(5):16–26.

Scarfone, K. and Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). Technical report, National Institute of Standards and Technology. Special Publication 800-94.

Shameli-Sendi, A., Aghababaei-Barzegar, R., and Cheriet, M. (2016). Taxonomy of information security risk assessment (isra). *Computers & Security*, 57:14 – 30.

Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357 – 374.

Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., and Rajarajan, M. (2015). Intrusion alert prioritisation and attack detection using post-correlation analysis. *Computers & Security*, 50:1 – 15.

Spathoulas, G. P. and Katsikas, S. K. (2013). Enhancing ids performance through comprehensive alert post-processing. *Comput. Secur.*, 37:176–196.

Stallings, W. and Brown, L. (2007). *Computer Security: Principles and Practice*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition.

Stavroulakis, P. P. and Stamp, M., editors (2010). *Handbook of Information and Communication Security*. Springer Science & Business Media.

Treinen, J. J. and Thurimella, R. (2006). A framework for the application of association rule mining in large intrusion detection infrastructures. In *Proceedings of the 9th International Conference on Recent Advances in Intrusion Detection*, RAID'06, pages 1–18, Berlin, Heidelberg. Springer-Verlag.

Vacca, J. (2013). *Computer and information security handbook*. Morgan Kaufmann, Amsterdam.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Xu, R. and Wunsch, D., I. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678.

Xuewei, F., Dongxia, W., Minhuan, H., and Xiaoxia, S. (2014). An approach of discovering causal knowledge for alert correlating based on data mining. In *Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on*, pages 57–62.

Zuech, R., Khoshgoftaar, T. M., and Wald, R. (2015). Intrusion detection and big heterogeneous data: a survey. *Journal of Big Data*, 2(1):1–41.