SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2015; **8**:822–836
Published online 19 July 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1039

RESEARCH ARTICLE

E-correlator: an entropy-based alert correlation system

Mohammad GhasemiGol* and Abbas Ghaemi-Bafghi

Data and Communication Security Lab, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

ABSTRACT

With the rapid size and complexity growth of computer networks, network supervisors are now facing a new problem, which is to analyze and manage the large amounts of security alerts that can be generated by security devices. Alert correlation systems attempt to solve this problem by finding the similarity and causality relationships between raw alerts and providing high-level view of the network under surveillance. Several alert correlation methods have been proposed recently to detect known attack scenarios. This paper focuses on how to develop an intrusion-alert correlation system according to the information existed in the raw alerts without using any predefined knowledge. For this purpose, first, we define the concept of alert partial entropy to find the alert clusters with the same information. Then, we represent the alert clusters by an intelligible notation called hyper-alerts. The network supervisor can reduce the number of hyper-alerts based on the principle of maximum entropy or by using the concept of hyper-alerts partial entropy. For more visualization, we define the hyper-alerts graph, which provides a global view of intrusion alerts. Our results show that the proposed entropy-based alert correlation system (E-correlator) can simplify the analysis of large number of alerts. We achieved the promising reduction ratio of 99.98% in LLS_DDOS_1.0 attack scenario in DARPA2000 dataset while the constructed hyper-alerts have enough information to discover the attacker, the victim, and the attack scenario. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS

intrusion detection system; alert correlation; information theory; alert partial entropy; hyper-alert partial entropy

***Correspondence**

Mohammad GhasemiGol, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
E-mail: ghasemigol@wali.um.ac.ir

1. INTRODUCTION

Intrusion detection systems (IDSs) are usually considered as a second line of defense for computer and network systems to protect them against malicious activities along with the prevention-based systems such as firewalls and access controls [1]. If an intrusion is detected, the IDS generates a warning known as alert or alarm [2]. Traditional IDSs focus on low-level attacks and generate overwhelming number of alerts per day. Analysis and management of these intrusion alerts is a troublesome and time-consuming task for network supervisors or intrusion response systems. In addition, the logical connections between alerts or attacking strategies behind the number of alerts cannot be discovered by IDSs. Hence, to improve the representation of security threats, alert correlation is a necessary and critical process in intrusion detection and response [3].

Alert correlation is defined as a process that contains multiple components with the purpose of analyzing alert and providing a high-level insight on the security state of the network under surveillance. Recent research on alert correlation can be classified into the following techniques [4]:

- Alert correlation based on feature similarity.
- Alert correlation based on known scenarios.
- Alert correlation based on prerequisite and consequence relationship.

Similarity-based correlation methods correlate alerts according to the similarities of some selected features, such as source IP addresses, destination IP addresses, protocols, and port numbers. Alerts with higher degree of overall feature similarity will be considered as correlated. The common weakness of these approaches is that they cannot fully discover the causal relationships between related alerts [4].

Scenario-based correlation methods correlate alerts according to the known attack scenarios. The attack scenario is either specified by an attack language such as STATL [5] or LAMDBA [6] or learned from training data sets using data mining approach [7]. Whenever a new alert is received, it is compared with the existing scenarios and then added to the most likely candidate scenario [8]. A common weakness of the scenario-based correlation techniques is that they are all restricted to known situation. In other words, the scenarios have to be predefined by an expert or be learned from labeled training examples [4].

The last type of alert correlation technique is based on the assumption that most alerts are not isolated, but related to different stages of attacks, with the early stages preparing for the later ones. Intuitively, the prerequisite of an attack is the necessary condition to launch an attack successfully, and the consequence of an attack is the possible outcome if an attack succeeds [9]. This kind of approach requires specific knowledge about the attacks in order to identify their prerequisites and consequences. On the basis of this information, alerts are considered to be correlated by matching the consequences of some previous alerts and the prerequisites of later ones [10]. However, the major limitation of this class of approaches is that they cannot correlate unknown attacks because its prerequisites and consequences are not defined. Even for known attacks, it is difficult to define all prerequisites and all of their possible consequences [4].

This paper proposes a new similarity correlation system based on entropy called E-correlator. The main idea of this paper is that the huge number of raw alerts contains some information that can be displayed by fewer hyper-alerts. At first, we defined the concept of alert partial entropy to find the alert clusters with the same information. Then, we represent the alert clusters by intelligible notation called hyper-alerts. The obtained hyper-alerts can be reduced by using the principle of maximum entropy or applying the concept of hyper-alerts partial entropy (APE). Finally, we generate the hyper-alerts graph (HG), which provides high-level view of intrusion alerts.

In Section 2, some of the related works in alert correlation are reviewed. The detail of proposed alert correlation system is presented in Section 3, while its performance in alert correlation is discussed in Section 4. Finally, the conclusions and some suggestions for future work are given in Section 5.

2. RELATED WORK

In this section, we review the related work in the literature, which address alert correlating techniques. In the similarity correlation methods, alerts are put into a group based on the similarity of their corresponding features. The most common attributes of alerts are Source IP, Destination IP, Source Port, Destination Port, Attack Class, and Timestamp. According to Valdes and Skinner, a probabilistic approach to alert correlation correlates attacks over time, over multiple attempts, and from multiple sensors. Their used features are based on alert content that anticipates evolving Internet Engineering Task Force standards. Their probabilistic approach provides a unified mathematical framework for correlating alerts that match closely but not perfectly, where the minimum degree of match required to fuse alerts is controlled by a single configurable parameter. Only features in common are considered in the fusion algorithm. For each feature, they define an appropriate similarity function. The overall similarity is weighted by a specifiable expectation of similarity [11].

Julisch proposed a clustering technique for grouping all the alerts, which share the same root causes. The clustering technique proposed by Julisch has hierarchy structures, which decompose the attributes of the alerts from the most general values to the most specific ones. These generalization hierarchies are later used for measuring the distance between alerts in a clustering algorithm [12]. We use the Julisch's generalization hierarchies to represent the hyper-alerts in our correlator system. Siraj *et al*. proposed a hybrid clustering model based on improved unit range, principal component analysis, and unsupervised learning algorithm to aggregate similar alerts and to reduce the number of alerts [13]. Perdisci *et al*. proposed a new on-line alarm clustering system to introduce a concise view about attacks and to reduce the volume of alarms. Their proposed system consists of three main modules, namely, an alarm management interface (AMI), an alarm classifier, and an alarm clustering module. The AMI receives alarms from multiple IDS and translates them in a standard format. Then, the alarm classifier assigns a class label to the received alarms and sends them to the alarm clustering module, where the alarms are fused to obtain meta-alarms [14].

In the known scenarios correlation methods, whenever a new alert is received, it is compared with the current existing scenarios and then added to the most likely candidate scenario. Some of the previous works in this category have used formal models for specifying attack scenarios, such as LAMBDA, STATL, and ADeLe [5,6,15]. However, some correlation research works are based on predefined attack scenarios. For example, Dain and Cunningham proposed a real-time alert clustering scheme, which fuses the alerts produced by multiple IDSs into scenarios. In this system, they use a probabilistic algorithm in which a new alert belongs to a given scenario (the scenario constructed by their algorithm does not necessarily indicate malicious behavior). Whenever a new alert is received, it is compared with current existing scenarios and then assigned to the scenario that yields the highest probability score [16].

In the prerequisite and consequence relationship alert correlation, we require specific knowledge about the attacks in order to identify their prerequisites and consequences. On the basis of this information, alerts are considered to be correlated by matching the consequences of some previous alerts and the prerequisites of later ones. Ning and Cui [17] proposed an alert correlation method to identify the prerequisites (e.g., existence of vulnerable services) and the consequences (e.g., discovery of vulnerable services) of each type of attacks and correlate the attacks by matching the consequences of some previous attacks and the prerequisites of some later ones. For example, if a UDP port scan is followed by a buffer overflow attack against one of the scanned ports, they can be correlated as the same series of attacks. They introduce the notion of *hyper-alert type*, which is used to represent the prerequisite and consequence of each type of alerts. A hyper-alert type T is a triple (fact, prerequisite, and consequence) where *fact* is a set of attribute names, each with an associated domain of values, *prerequisite* is a logical formula whose free variables are all in fact, and *consequence* is a set of logical formulas such that all the free variables in consequence are in fact. For example, consider the buffer overflow attack against the sadmind remote administration

tool; this can be represented using the following hyper-alert type:

SadmindBufferOverflow = (fact, prerequisite, and consequence) for such attacks, where

- Fact = (IP, port),
- Prerequisite = ExistHost (IP) and VulnerableSadmind (IP), and
- Consequence = (GainRootAccess(IP)).

Generally, the scenario-based and prerequisite-consequence methods are limited to a predefined knowledge base, whereas the similarity techniques are capable of correlating alerts that may contribute to unknown attacks. On the other hand, the common weakness of the similarity approaches is that they cannot fully discover the causal relationships between related alerts.

3. THE PROPOSED ENTROPY-BASED ALERT CORRELATION SYSTEM

In this section, we proposed a new similarity correlation system based on entropy (E-correlator). The main idea of this work is that the massive number of alerts is correlated so that the correlated alerts have the same quantity of information as the original. Figure 1 shows the architecture of E-correlator, which has the following procedure:

Input: Raw alerts
Output: Hyper-alerts graph
Step 1: Modeling the raw alerts by APE matrix.
Step 2: Run the density-based spatial clustering of applications with noise (DBSCAN) algorithm on the APE matrix to cluster the raw alerts based on their obtained partial entropy as the similarity measure.
Step 3: Generate the alert clusters by intelligible notation called hyper-alerts.
Step 4: The network supervisor can reduce the number of hyper-alerts based on the principle of maximum entropy or use the concept of hyper-APE.
Step 5: Generate the HG for more visualization.

We will describe the component of proposed correlator system and its complexity in greater detail in the following subsections.

3.1. Modeling the raw alerts by alerts partial entropy matrix

We first introduce the concept of entropy, which is a measure of the uncertainty of a random variable. Let X be a discrete random variable with alphabet X and probability mass function $P(X) = \Pr(X = x_i)$ and $x_i \in X$. The entropy of this random variable with the probability mass function $P(X)$ is defined by [18]:

$$H(X) = -\sum_{x_i \in X} P(x_i) \log_2 P(x_i) \tag{1}$$

According to this equation, each value $x_i \in X$ has its portion in obtained entropy. We named each of these portions as partial entropy. The formal definition of partial entropy is given in the succeeding text.

Definition 1 (Partial entropy). *Consider a discrete random variable with alphabet X and probability mass function P(X). Let H(X) be its entropy. The partial entropy of X is the portion of each value $x_i \in X$ in H(X), it can be written as*

$$H_P(X = x_i) = -P(x_i) \log_2 P(x_i) \tag{2}$$

So we have the following equation:

$$H(X) = \sum_{x_i \in X} H_P(X = x_i) \tag{3}$$

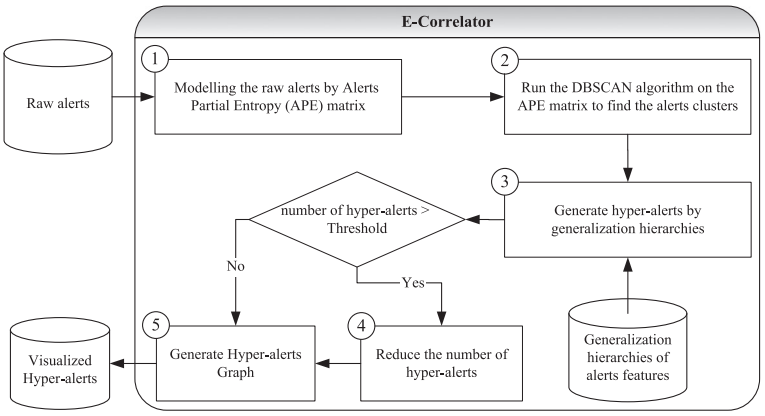

Figure 1. Architecture of E-correlator.

Now, suppose that the set of alerts is defined by $\psi = \{A_1, A_2, \ldots, A_n\}$, and the set of alert features, such as source IP address, destination IP address, protocol, source port number, destination port number, time, and duration, is shown by $F = \{F_1, F_2, \ldots, F_k\}$. Each feature F_j is a discrete random variable with the set of value $\{f_j\}$ and the probability mass function of $P_j(f_j)$. Hence, we can calculate the entropy of each feature F_j as the following equation:

$$H(F_j) = -\sum_{f_j \in F_j} P_j(f_j) \log_2 P_j(f_j) \tag{4}$$

According to Equation (2), the partial entropy of feature F_j for $f_j \in F_j$ is defined by

$$H_P(F_j = f_j) = -P_j(f_j) \log_2 P_j(f_j) \tag{5}$$

Now, we can consider two assumptions to compute the partial entropy of each alert. First, suppose that the alert features are independent, so we have the following definition for partial entropy of each alert.

Definition 2 (Alert partial entropy for independent features). *Suppose that we have the set of alerts $\psi = \{A_1, A_2, \ldots, A_n\}$, and the set of alert features is shown by $F = \{F_1, F_2, \ldots, F_k\}$, and the alert features are independent. For each alert $A_i = [f_{i1}, f_{i2}, \ldots, f_{ik}]$, its partial entropy is a vector, which is defined by*

$$\begin{aligned} H_P(\psi = A_i) &= H_P([F_1, F_2 \ldots, F_k] = [f_{i1}, f_{i2} \ldots, f_{ik}]) \\ &= [H_P(F_1 = f_{i1}), H_P(F_2 = f_{i2}), \ldots, H_P(F_k = f_{ik})] \end{aligned} \tag{6}$$

Whereas $f_{ij} \in F_j$ and

$$H_P(F_j = f_{ij}) = -P_j(f_{ij}) \log_2 P_j(f_{ij})$$

According to Definition 2, we can calculate the partial entropy of alerts and fill the following APE matrix from the set of alerts (Each row shows the partial entropy of an alert):

$$\begin{matrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{matrix} \begin{bmatrix} H_P(F_1 = f_{11}) & H_P(F_2 = f_{12}) & \cdots & H_P(F_k = f_{1k}) \\ H_P(F_1 = f_{21}) & H_P(F_2 = f_{22}) & \cdots & H_P(F_k = f_{2k}) \\ \vdots & \vdots & \cdots & \vdots \\ H_P(F_1 = f_{n1}) & H_P(F_2 = f_{n2}) & \cdots & H_P(F_k = f_{nk}) \end{bmatrix}$$

Now, suppose that some of the alert features are not independent, for example, the destination port number may be dependent on the source port number. Here, we define the concept of partial conditional entropy to compute the partial entropy of alerts with more precisely. For this purpose, first, we should define the partial joint entropy. The entropy definition (Equation (1)) can be easily extended to a pair of random variables. Let (X, Y) be two discrete random variables with a joint distribution $P(X, Y)$. The joint entropy of these random variables is defined by [18]

$$H(X, Y) = -\sum_{x_i \in X} \sum_{y_i \in Y} p(x_i, y_i) \log p(x_i, y_i) \tag{7}$$

According to this equation, each value $(x_i, y_i) \in (X, Y)$ has its portion in obtained joint entropy. We named each of these portions as partial joint entropy. The formal definition of partial joint entropy is given in the succeeding text.

Definition 3 (Partial joint entropy). *Consider two discrete random variables X* and *Y, with a joint distribution $P(X, Y)$. Let $H(X, Y)$ be their joint entropy. The partial joint entropy of (X, Y) is the portion of each value $(x_i, y_i) \in (X, Y)$ in $H(X, Y)$, it can be written as*

$$H_P(X = x_i, Y = y_i) = -p(x_i, y_i) \log p(x_i, y_i) \tag{8}$$

According to the chain rule for entropy,

$$H(X_1, X_2 \cdots, X_n) = \sum_{j=1}^{n} H(X_j | X_{j-1}, \cdots X_1) \tag{9}$$

the entropy of a pair of random variables is the entropy of one plus the conditional entropy of the other [18]. So we have the following equation for conditional entropy:

$$H(Y|X) = H(X, Y) - H(X) \tag{10}$$

Definition 4 (Partial conditional entropy). *Consider two discrete random variables X and Y, with a joint probability mass function $P(X, Y)$ and marginal probability mass functions $P(X)$* and *$P(Y)$. The partial conditional entropy of (X, Y) is the portion of each value $(x_i, y_i) \in (X, Y)$* in *$H(Y|X)$, it can be written as*

$$H_P(Y = y_i | X = x_i) = H_P(X = x_i, Y = y_i) - H_P(X = x_i) \tag{11}$$

So if we assume that some of the alert features are not independent, we can extend the mentioned alert partial entropy definition as follows:

Definition 5 (Alert partial entropy including some independent features). *Suppose that we have the set of alerts $\psi = \{A_1, A_2, \ldots, A_n\}$, and the set of alert features is shown by $F = \{F_1, F_2, \ldots, F_k\}$, and some of the alert features are not independent. Let $F_d \in F$ be the only dependent feature in the set of alert features,*

and it conditioned on the feature F_j. For each alert $A_i = [f_{i1}, f_{i2}, \ldots, f_{id}, \ldots, f_{ik}]$, its partial entropy is a vector, which is defined by

$$H_P(\psi = A_i) = [H_P(F_1 = f_{i1}), H_P(F_2 = f_{i2}), \ldots, H_P(F_d = f_{id} | F_j = f_{ij}), \ldots, H_P(F_k = f_{ik})] \quad (12)$$

So we can modify the APE matrix to support this dependent feature as follows:

$$\begin{matrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{matrix} \begin{bmatrix} H_P(F_1 = f_{11}) \cdots H_P(F_d = f_{1d} | F_j = f_{1j}) & \cdots & H_P(F_k = f_{1k}) \\ H_P(F_1 = f_{21}) & \cdots H_P(F_d = f_{2d} | F_j = f_{2j}) \cdots & H_P(F_k = f_{2k}) \\ \vdots & \vdots & \vdots \\ H_P(F_1 = f_{n1}) & \cdots H_P(F_d = f_{nd} | F_j = f_{nj}) \cdots & H_P(F_k = f_{nk}) \end{bmatrix}$$

Similarly, we can add the other dependent features into the APE matrix. Even there may be more complex relationships between the features, for example, $F_d \in F$ can be conditioned on two features or more. In these cases, we can use the chain rule entropy (Equation (9)) to compute the partial conditional entropy of feature.

However, each alert can be represented by a k-dimensional vector as the alert partial entropy, and the set of alerts can be shown by APE matrix. While the alerts with the same information have similar partial entropy, we can reduce the number of alerts by a simple clustering algorithm on the APE matrix to find the similar rows.

3.2. Review of density-based spatial clustering of applications with noise algorithm

After calculating the APE matrix, we need a method to cluster the alerts based on their partial entropy. Here, we use DBSCAN [19] as a well-known density-based clustering algorithm to aggregate the similar alerts into specific clusters. It offers several advantages compared with the other clustering methods. DBSCAN does not require one to specify the number of clusters in the data *a priori*. Also, it is better at finding arbitrarily shaped clusters and detecting the noise or outliers. DBSCAN requires two parameters: density threshold (eps) and minimum size (MinPts). It starts with an arbitrary starting point that has not been visited yet. Then, it finds all the neighbor points within distance eps of the starting point. If the number of neighbors is less than MinPts, the point is marked as noise. But when the number of neighbors is greater than or equal to MinPts, a cluster is formed. The starting point and its neighbors are added to this cluster, and the starting point is marked as visited. The algorithm then repeats the evaluation process for all the neighbors recursively. This process continues until the density-connected cluster is completely found. Then, a new unvisited point is retrieved and processed to discover a further cluster or noise. In the following, we present the pseudocode of DBSCAN:

```
DBSCAN(D, eps, MinPts)
  C = 0
  for each unvisited point P in dataset D
    mark P as visited
    NeighborPts = regionQuery(P, eps)
    if sizeof(NeighborPts) < MinPts
      mark P as NOISE
    else
      C = next cluster
      expandCluster(P, NeighborPts, C, eps, MinPts)

expandCluster(P, NeighborPts, C, eps, MinPts)
  add P to cluster C
  for each point P' in NeighborPts
    if P' is not visited
      mark P' as visited
      NeighborPts' = regionQuery(P', eps)
      if sizeof(NeighborPts') >= MinPts
        NeighborPts = NeighborPts joined with NeighborPts'
    if P' is not yet member of any cluster
      add P' to cluster C

regionQuery(P, eps)
  return all points within P's eps-neighborhood (including P)
```

3.3. Generate hyper-alerts by generalization hierarchies

In this section, we want to display the alerts located in a cluster by intelligible notation. For this purpose, first, we define a meaningful hierarchy for each alert feature. A hierarchy defines a sequence of mappings from a set of concepts to their higher-level correspondences [20]. A good example of this technique proposed by Pietraszek [21] as generalization hierarchies. He labeled the IP addresses according to their role (Workstation, Firewall, HTTPServer, etc.) and then grouped according to their network location (Intranet, DMZ, Internet, etc.) with a final top-level generalized address AnyIP (Figure 2). When these classification hierarchies are not known, the IP addresses can be generalized according to the hierarchies in the addressing structure; For example, an IP address 195.176.20.45 can be generalized to the corresponding class C network: 195.176.20.0/24, followed by the class-B generalization 195.176.0.0/16, class-A generalization 195.0.0.0/8, and, finally, AnyIP.

Furthermore, the other attributes will have different generalization hierarchies, depending on the type and our interests. For example, the source and destination ports of port-oriented IP connections can be generalized into privileged (0–1023) and non-privileged (1024–65 535), with a top-level category of AnyPort. In addition, the

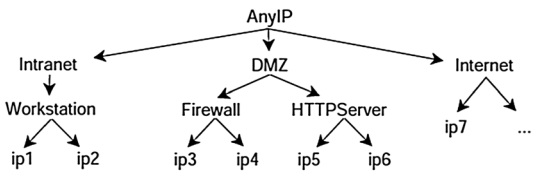

Figure 2. A sample generalization hierarchies for IP address.

well-known destination ports (0–1023) can comprise a number of hierarchies describing their function, for example, httpPorts (80, 443, 8080, and 9090), mailPorts (25, 110, 143, 993, and 995), and chatPorts (194, 258, 531, and 994). By this generalization hierarchy, the hyper-alerts can be constructed from the set of alert clusters.

Here, we use the following 14 features to construct the hyper-alerts:

- Source address
- Frequency of source address
- Destination address
- Frequency of destination address
- Protocol
- Frequency of protocol
- Source port number
- Frequency of source port number
- Destination port number
- Frequency of destination port number
- Time interval (lower bound of time)
- Time interval (upper bound of time)
- Duration
- Contained number of alerts

3.4. Reduce the number of hyper-alerts

This is an optional step to reduce the number of hyper-alerts when we have many of them. Here, we can select the specified number of hyper-alerts that contain the most of information about the set of alerts. In this section, two methods for this purpose are discussed. The first method is based on the principle of maximum entropy, and the second one uses the concept of hyper-APE.

According to the principle of maximum entropy, when estimating the probability distribution, you should select that distribution which leaves you the largest remaining uncertainty consistent with your constraints. Here, we want to estimate the hyper-alerts probability distribution. So we should select a subset of hyper-alerts with the maximum entropy subject to the constraint that the number of desired hyper-alerts is specified by the network supervisor. In other words, we have the following optimization problem:

$$\begin{aligned} &\max && Entropy(hyperalerts) \\ &s.t. && \\ & && number\ \ of\ \ desired\ \ hyperalerts = x \end{aligned} \tag{13}$$

We can use the genetic algorithm to solve the aforementioned optimization problem. But before that, the entropy of hyper-alerts should be defined.

Definition 6 (Hyper-alerts entropy). *Suppose that the set of hyper-alerts is defined by* $\hbar = \{h_1, h_2, \ldots, h_l\}$, *and the set of hyper-alert features, such as source address, destination address, protocol, source port number, destination port number, time interval, duration, and contained number of alerts, is shown by* $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_m\}$. *So each hyper-alert* h_i *is displayed by a vector with m different features* ($h_i = [\gamma_{i1}, \gamma_{i2}, \ldots, \gamma_{im}]$). *Now, suppose that each feature* Γ_j *is a discrete random variable on the set* $\{v_{1j}, v_{2j}, \ldots, v_{pj}\} \cup \{g_{1j}, g_{2j}, \ldots, g_{qj}\}$ *so that* $\{v_{1j}, v_{2j}, \ldots, v_{pj}\}$ *is the non-generalized value set and* $\{g_{1j}, g_{2j}, \ldots, g_{qj}\}$ *is the generalized one for feature* Γ_j. Let $\{g(v_{1j}), g(v_{2j}), \ldots, g(v_{pj})\}$ *be the generalized value for* $\{v_{1j}, v_{2j}, \ldots, v_{pj}\}$, where $g(v_{ij})$ *is the generalized value for* v_{ij}. *So we can calculate the entropy of* Γ_j *as the following equation:*

$$\begin{aligned} H(\Gamma_j) = &-\sum_{i=1}^{R} P(v_{ij}) \log_2 P(v_{ij}) \\ &-\sum_{i=1}^{R} P(g(v_{ij})) \log_2 P(g(v_{ij})) \\ &-\sum_{i=1}^{q} P(g_{ij}) \log_2 P(g_{ij}) \end{aligned} \tag{14}$$

and the hyper-alerts entropy can be defined by

$$H(\hbar) = [H(\Gamma_1), H(\Gamma_2), \ldots, H(\Gamma_m)] \tag{15}$$

Similar to alert partial entropy definition, we can define the hyper-APE as follows.

Definition 7 (Hyper-APE). *For each hyper-alert* $h_i = [\gamma_{i1}, \gamma_{i2}, \ldots, \gamma_{im}]$, *its partial entropy is a vector, which is defined by*

$$\begin{aligned} H_P(\hbar = h_i) &= H_P([\Gamma_1, \Gamma_2, \ldots, \Gamma_m] = [\gamma_{i1}, \gamma_{i2}, \ldots, \gamma_{im}]) \\ &= [H_P(\Gamma_1 = \gamma_{i1}), H_P(\Gamma_2 = \gamma_{i2}), \ldots, H_P(\Gamma_m = \gamma_{im})] \end{aligned} \tag{16}$$

Whereas $\gamma_{ij} \in \Gamma_j$, $H_P(\Gamma_j = \gamma_{ij})$ is the partial entropy, which is computed by one of the following two equations. If γ_{ij} is a non-generalized value, we use

$$\begin{aligned} H_P\left(\Gamma_j = \gamma_{ij}\right) = &-P_j(v_{ij}) \log_2 P_j(v_{ij}) \\ &-P_j(g(v_{ij})) \log_2 P_j(g(v_{ij})) \end{aligned} \tag{17}$$

and if γ_{ij} is a generalized value, we have

$$H_P\left(\Gamma_j = \gamma_{ij}\right) = -P_j(g_{ij}) \log_2 P_j(g_{ij}) \tag{18}$$

We can use the hyper-APE to determine the importance of hyper-alerts based on the amount of information contained. In other words, we can compute the weight of each hyper-alert h_i by the following equation:

$$Weight(h_i) = \sum_{j=1}^{m} H_P\left(\Gamma_j = \gamma_{ij}\right) \tag{19}$$

Clearly, a hyper-alert with more weight has more importance among the others. However, the network supervisor

can sort the hyper-alerts on the basis of their partial entropy and select the desired number of hyper-alerts from the top of the ordered list.

3.5. Generate hyper-alerts graph

For more visualization, we define the HG, which displays the flow of HGs and provides high-level view of the network under surveillance.

Definition 8 (Hyper-alerts graph). *Hyper-alerts graph is a directed graph that its nodes are hyper-addresses (generalized or non-generalized, source or destination IP addresses) contained in the set of hyper-alerts and its directed edges display the flow of alerts between two IP addresses. The edge label has the following five partitions to show the detail of hyper-alerts (Figure 3):*

(A) number of outgoing alerts for each source addresses
(B) protocol(s)
(C) source port(s)
(D) destination port(s)
(E) number of input alerts for each destination addresses

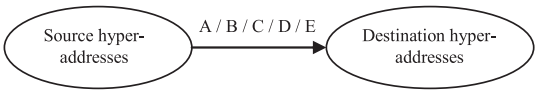

Figure 3. Relation between nodes in the hyper-alerts graph.

3.6. Complexity analysis

Here, we analyze the time complexity of the proposed alert correlation system. Suppose that n is the number of alerts and k is the alert dimension. In the first step, the probability mass function can be computed with time complexity $O(n.\log n)$. Then, we represent the partial entropy of each alert with a k-dimensional vector, which takes $O(nk)$. Because in real systems $n >> k$, the computational complexity of this step is approximately $O(n.\log n)$.

In the second step, we use DBSCAN algorithm, which is the most time-consuming part in our framework. In DBSCAN algorithm, each alert is visited possibly multiple times. The computational complexity of DBSCAN without any special structure is $O(n^2)$. Moreover, if a spatial index such as R*-tree or KD-tree is used, the complexity can be reduced to $O(n.\log n)$ [19].

In the third step, we generate the hyper-alerts by generalization hierarchies. For this purpose, we need to visit each alert only once. So the computational complexity of this step is $O(n)$.

Now, let m be the number of generated hyper-alerts. In the fourth step, which is an optional process, the number of hyper-alerts can be reduced in two ways. One way is based on the principle of maximum entropy in which we use the genetic algorithm to solve Equation (13), and it takes

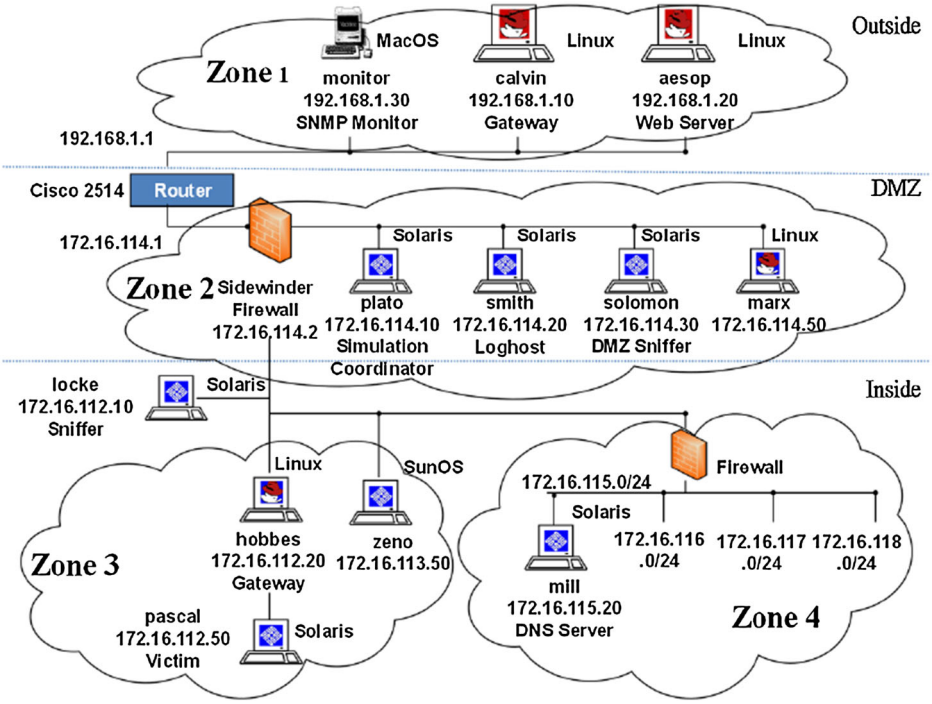

Figure 4. DARPA 2000 topology.

Table I. Generated hyper-alerts for LLS_DDOS_1.0 attack scenario (MinPts = 5 and eps = 0.0001 for density-based spatial clustering of applications with noise algorithm).

ID	Source address	fr[1]	Destination address	fr	Protocol	fr	Port source	fr	Port destination	fr	Min time	Max time	Mean duration	Number of alerts	Phase indicator
1	202.77.162.213 172.16.114.50 Zone3 172.16.115.20	7 1 6 7	202.77.162.213 172.16.114.50 Zone3 172.16.115.20	14 1 4 2	icmp-d-u tcp udp	1 18 2	Invalid privileged registered	1 18 2	Invalid privileged	1 20	'10:11:09 AM'	'11:34:21 AM'	2.2E-06	21	Noise alerts
2	202.77.162.213	767	Zone2 Zone3 Zone4	5 508 254	icmp-req	767	−1	767	−1	767	'9:51:35 AM'	'9:52:01 AM'	0	767	Phase 1
3	Zone2 Zone3 Zone4	5 10 3	202.77.162.213	18	icmp-rep	18	−1	18	−1	18	'9:51:35 AM'	'9:51:59 AM'	0	18	Phase 1
4	202.77.162.213	10	172.16.114.2 Zone3 Zone4	1 7 2	UDP	10	Privileged	10	111	10	'10:07:07 AM'	'10:17:10 AM'	0.000637	10	Phase 2
5	202.77.162.213	76	Zone2 Zone3 172.16.115.20	42 20 14	UDP	76	Privileged	76	Dynamic privileged registered	17 38 21	'10:08:06 AM'	'10:34:58 AM'	0	76	Phases 2 and 3
6	202.77.162.213	20	Zone2 Zone3 172.16.115.20	9 5 6	TCP	20	Privileged registered	1 19	Privileged	20	'10:33:14 AM'	'11:26:14 AM'	0.000127	20	Phases 3 and 4
7	131.84.1.31	32838	Outside	32838	TCP	32838	Invalid registered	4 32834	Registered	32838	'11:27:50 AM'	'11:27:55 AM'	0	32838	Phase 5
8	Outside	1069	131.84.1.31	1069	TCP	1069	Registered	1069	Invalid privileged	1 1068	'11:27:50 AM'	'11:27:55 AM'	0	1069	Phase 5

UDP, user datagram protocol; TCP, transmission control protocol. icmp-d-u, icmp-destination-unreachable; icmp-req, icmp-echo-request; icmp-rep, icmp-echo-reply.

$O(gpi)$ time to be solved where g is the number of generations, p is the population size, and i is the size of the individuals, which is equal to the number of hyper-alerts. So it can be rewritten as $O(gpm)$. The other way to reduce the number of hyper-alerts is using hyper-APE, which has $O(m.\log m)$ computational cost.

Finally, in the last step, we can create the HG with time complexity $O(m)$. Therefore, the total computational complexity of our framework is $2O(n.\log n)+O(kn)+\{O(m.\log m) \mid O(gpm)\}+O(m)$. According to the fact that $n>>m$ and $n>>k$, the total computational complexity can be approximately summarized as $O(n.\log n)$.

4. EXPERIMENTS

In this section, we test the proposed entropy-based correlator by using DARPA2000 [22] dataset to demonstrate how it works. Even after 13 years of its generation, it has been used in many papers because it was introduced and it is the only choice to compare alert correlation methods [4,23–26].

4.1. DARPA2000 dataset

DARPA 2000 is a well-known IDS evaluation dataset created by MIT Lincoln Laboratory over the topology shown in Figure 4. There are two attack scenarios in DARPA2000 dataset, LLS_DDOS_1.0, and LLDOS2.0.2. In both scenarios, the attacker tries to install components necessary to run a distributed denial-of-service (DDoS) and then launch a DDoS at a US government site. The main difference between them is that the attacker uses IPSweep and Sadmind Ping to find out the vulnerable hosts in LLS_DDOS_1.0 while DNS HInfo is used in LLDOS2.0.2; second, the attacker attacks each host individually in LLS_DDOS_1.0, while in LLDOS2.0.2, the attacker breaks into one host first and then fans out from it.

In this paper, we only show the results of evaluation on LLS_DDOS_1.0. In this scenario, the attacker first sends Internet control message protocol (ICMP) echo-requests to many IP addresses and listens for ICMP echo-replies to determine which hosts are "up" and then uses the "ping" option of the sadmind exploit program to determine which of the discovered hosts are running the sadmind service. In the next phase, the attacker tries to break into the hosts found to be running the sadmind service in the previous phase and launches the sadmind remote-to-root exploit several times against each host, each time with different parameters. After gaining root access in each host, the attacker uses telnet, rcp, and rsh to install a DDoS program in the compromised machines. However, the five phases of the attack scenario are as follows:

Phase 1: Perform IPsweep to look for live hosts from a remote site.
Phase 2: Probe live hosts to look for vulnerable ones running sadmind service.
Phase 3: Break in these hosts via sadmind vulnerability, both successful and unsuccessful.
Phase 4: Install Trojan mstream DDoS software on three hosts.
Phase 5: Launch DDoS.

Here, we have performed the experiments, with the DMZ network traffic of LLS_DDOS_1.0 that contains 34 819 alerts, which indicated the five steps of DDoS attack on the target IP address 131.84.1.31. The mentioned E-correlator system is applied to this set of alerts. According to the network topology used to capture the DARPA dataset (Figure 4), we suppose the following generalization hierarchies:

- The IP addresses can be generalized into Zone1, Zone2, Zone3, Zone4, and outside.
- The source and destination ports of port-oriented IP connections can be generalized into privileged (1–1024), registered (1025–49 151), and dynamic (49152–65 535).

Table II. The result of E-correlator for LLS_DDOS_1.0 attack scenario (MinPts = 5 and eps = 0.0001 in density-based spatial clustering of applications with noise algorithm).

	Number of raw alerts	Number of generated hyper-alerts (without running step 4)	Reduction ratio (%)
Phase 1	785	Two	99.74
Phase 2	25	Two (one of them is shared between Phases 2 and 3)	92.00
Phase 3	80	Two (one of them is shared between Phases 2 and 3) (one of them is shared between Phases 3 and 4)	97.50
Phase 4	19	One (shared between Phases 3 and 4)	94.74
Phase 5	33 910	Two	99.99
The total of raw alerts	34 819	Seven	99.98

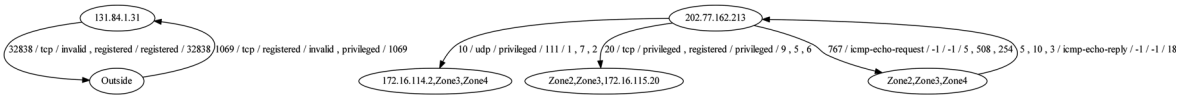

Figure 7. Generated hyper-alerts graph for LLS_DDOS_1.0 attack scenario (MinPts = 5 and eps = 0.0001 in density-based spatial clustering of applications with noise algorithm).

The generated hyper-alerts are shown in Table I while we assume that MinPts = 5 and eps = 0.0001 in the DBSCAN algorithm. With a brief review on these hyper-alerts, we can reach the following results (Table II):

- Getting the promising reduction ratio of $(34819 - 7)/34819 \cong 99.98\%$ in LLS_DDOS_1.0 attack scenario without running step 4. We measured the reduction ratio similar to [27] by dividing the difference between the number of raw alerts and generated hyper-alerts into the total number of raw alerts.
- The obtained seven hyper-alerts cover the general information existed in each of the five phases of attack senario and provides a more global view of what is happening in the network.

If the number of produced hyper-alerts exceeds a certain threshold, the network supervisor can reduce the number of reported hyper-alerts by running step 4. We have repeated the experiment with two different values of MinPts and report the results in Appendix. If we select a large value for MinPts parameter, the alerts reduction ratio will be increased. But in this state, we will lose many details, and the generated HG cannot be useful to guess the attack scenario (Tables III and IV and Figure 5). On the other hand, decreasing the value of MinPts may lead to increase the number of produced hyper-alerts (Tables V and VI and Figure 6). So we have to make a trade-off between the number of hyper-alerts and the amount of detail lost.

For more visualization, we can generate the HG from the list of produced hyper-alerts. Figure 7 shows the HG of LLS_DDOS_1.0 attack scenario, which is generated from the hyper-alerts listed in Table I. By this representation of hyper-alerts, the network supervisor can easily find the flow of alerts and detect the suspicious nodes, victim nodes, and the attack strategy from HG. In addition, the given HG can help us in selecting a suitable response strategy.

5. CONCLUSION

This paper presents a new alert correlation system based on entropy called E-correlator. The main idea of the proposed system is to correlate the raw alerts without any predefined knowledge. In addition, the correlated alerts have the same quantity of information as the original. For this purpose, we defined the concept of alert partial entropy. The alerts with the similar partial entropy indicate the same information; hence, we can correlate them into a specific cluster and report them by an intelligible hyper-alert. If the number of produced hyper-alerts exceeds a certain threshold, the network supervisor can reduce them by the principle of maximum entropy or usage of the concept of hyper-APE. Finally, for more visualization, we create the HG from the list of hyper-alerts. We validated E-correlator system on attack scenario LLS_DDOS_1.0 in DARPA 2000 dataset. The reduction ratio with the experiments was 99.98% while the generated hyper-alerts have the enough information to discover the attack scenario. In addition, the HG provides high-level view of intrusion alerts. In our future research, we plan to investigate additional information sources other than the raw alerts (such as attack graph and vulnerabilities existed in application, services, and protocols of hosts) to use in E-correlator system.

REFERENCES

1. Ning P, Jajodia S. Intrusion detection basics. In *Handbook of Computer Networks*, Bidgoli H (ed). John Wiley & Sons, Inc., Hoboken, New Jersey, 2006.
2. Njogu HW, Jiawei L, Kiere JN, *et al.* A comprehensive vulnerability based alert management approach for large networks. *Future Generation Computer Systems* 2013; **29**:27–45.
3. Xu D, Ning P. Privacy-preserving alert correlation: a concept hierarchy based approach. In The 21st Annual Computer Security Applications Conference, Tucson, AZ, USA, 2005; 537–546.
4. Ghorbani AA, Lu W, Tavallaee M. *Network Intrusion Detection and Prevention Concepts and Techniques*. : Springer US, 2009.
5. Eckmann ST, Vigna G, Kemmerer RA. STATL: an attack language for state-based intrusion detection. In 1st ACM Workshop on Intrusion Detection Systems, Athens, Greece, 2000.
6. Cuppens F, Ortalo R. A language to model a database for detection of attacks. In Recent Advances in Intrusion Detection, Toulouse, France, 2000.
7. Xiang G, Dong X, Yu G. Correlating alerts with a data mining based approach. In The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service, Hong Kong, 2005.
8. Al-mamory SO, Zhang HL. A survey on IDS alerts processing techniques. In 6th WSEAS International Conference on Information Security and Privacy, Wisconsin, USA, 2007.
9. Taha AEE. Intrusion detection correlation in computer network using multi-agent system. Ain Shams University, 2011.
10. Ning P, Reeves DS, Cui Y. Correlating alerts using prerequisites of intrusions. North Carolina State University, Department of Computer Science, 2001.
11. Valdes A, Skinner K. Probabilistic alert correlation. In Recent Advances in Intrusion Detection, Davis, CA, USA, 2001; 54–68.
12. Julisch K. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security* 2003; **6**(4):443–471.
13. Siraj MM, Maarof MA, Hashim SZM. A hybrid intelligent approach for automated alert clustering and filtering in intrusion alert analysis. *International Journal of Computer Theory and Engineering* 2009; **1**(5):539–545.

14. Perdisci R, Giacinto G, Roli F. Alarm clustering for intrusion detection systems in computer networks. *Engineering Applications of Artificial Intelligence* 2006; **19**:429–438.
15. Totel E, Vivinis B, Mé L. A language driven ids for event and alert correlation. In SEC, 2004; 209–224.
16. Dain O, Cunningham RK. Fusing a heterogeneous alert stream into scenarios. In ACM Computer and Communications Security, Philadelphia, Pennsylvania, USA, 2001.
17. Ning P, Cui Y. An intrusion alert correlator based on prerequisites of intrusions, Technical Report TR-2002-01, Department of Computer Science, North Carolina State University, 2002.
18. Cover TM, Thomas JA. *Elements of Information Theory*. John Wiley & Sons, Inc., New York, NY, 2006.
19. Ester M, Kriegel H-P, Sander J *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise. In 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA, 1996.
20. Al-Mamory SO, Zhang H. Intrusion detection alarms reduction using root cause analysis and clustering. *Computer Communications* 2009; **32**:419–430.
21. T Pietraszek. Alert classification to reduce false positives in intrusion detection. Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany, 2006.
22. Darpa. Intrusion detection evaluation datasets, 2000 http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data/2000data.html.
23. Ahmadinejad SH, Jalili S, Abadi M. A hybrid model for correlating alerts of known and unknown attack scenarios and updating attack graphs. *Computer Networks* 2011; **55**:2221–2240.
24. Xuewei F, Dongxia W, Guoqing M *et al*. Research on the key technology of reconstructing attack scenario based on state machine. In 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, China, 2010; 42–46.
25. Xu D. Correlation analysis of intrusion alerts. North Carolina State University, 2006.
26. Soleimani M, Ghorbani AA. Multi-layer episode filtering for the multi-step attack detection. *Computer Communications* 2012; **35**:1368–1379.
27. Sadoddin R, Ghorbani AA. An incremental frequent structure mining framework for real-time alert correlation. *Computers & security* 2009; **28**:153–173.

APPENDIX

In this section, we have repeated the experiment with two different values of MinPts and report the results in the following. If we select a large value for MinPts parameter, the alerts reduction ratio will be increased. But in this state, we will lose many details and the generated HG cannot be useful to guess the attack scenario. On the other hand, decreasing the value of MinPts may lead to increase the number of produced hyper-alerts.

Table III. Generated hyper-alerts for LLS_DDOS_1.0 attack scenario (MinPts = 10 and eps = 0.0001 in density-based spatial clustering of applications with noise algorithm).

ID	Source address	fr[1]	Destination address	fr	Protocol	fr	Port source	fr	Port destination	fr	Min time	Max time	Mean duration	Number of alerts	Phase indicator
1	202.77.162.213	10	202.77.162.213	15	icmp-d-u	1	Invalid	10	Invalid	10	'9:51:35 AM'	'10:50:52 AM'	1.06E-05	25	Noise
	172.16.114.50	1	Zone2	6	icmp-req	3	privileged	12	privileged	15					alerts
	Zone3	8	Zone3	2	tcp	6	registered	3							
	172.16.115.20	6	172.16.115.20	2	icmp-rep	15									
2	202.77.162.213	761	Zone2	2	icmp-req	761	−1	761	−1	761	' 9:51:35 AM'	'9:52:01 AM'	0	761	Phase 1
			Zone3	506											
			Zone4	253											
3	Zone2	5	202.77.162.213	15	icmp-rep	15	−1	15	−1	15	'9:51:35 AM'	' 9:51:59 AM'	0	15	Phase 1
	Zone3	8													
	Zone4	2													
4	202.77.162.213	12	Zone2	2	UDP	12	Privileged	12	111	12	'10:07:07 AM'	'10:17:10 AM'	0.00053	12	Phase 2
			Zone3	8											
			Zone4	2											
5	202.77.162.213	97	202.77.162.213	2	TCP	23	Privileged	81	Dynamic	17	'10:08:06 AM'	'11:34:21 AM'	2.34E-05	99	Phases
	172.16.115.20	2	Zone2	48	UDP	76	registered	18	privileged	61					2, 3, 4,
			Zone3	28					registered	21					and 5
			172.16.115.20	21											
6	Outside	33907	Outside	33907	TCP	33907	Invalid	4 33903	Invalid	1	'11:27:50 AM'	'11:27:55 AM'	0	33907	Phase 5
							registered		privileged	1068					
									registered	32838					

UDP, user datagram protocol; TCP, transmission control protocol; icmp-d-u, icmp-destination-unreachable; icmp-rep, icmp-echo-reply; icmp-req, icmp-echo-request.

Table IV. The result of E-correlator For LLS_DDOS_1.0 attack scenario (MinPts = 10 and eps = 0.0001 in density-based spatial clustering of applications with noise algorithm).

	Number of raw alerts	Number of generated hyper-alerts (without running step 4)	Reduction ratio (%)
Phase 1	785	Two	99.74
Phase 2	25	Two (one of them is shared between Phases 2 and 3 and Phases 4 and 5)	92.00
Phase 3	80	One (shared between Phases 2 and 3 and Phases 4 and 5)	98.75
Phase 4	19	One (shared between Phases 2 and 3 and Phases 4 and 5)	94.73
Phase 5	33 910	Two (one of them is shared between Phases 2 and 3 and Phases 4 and 5)	99.99
The total of raw alerts	34 819	Five	99.99

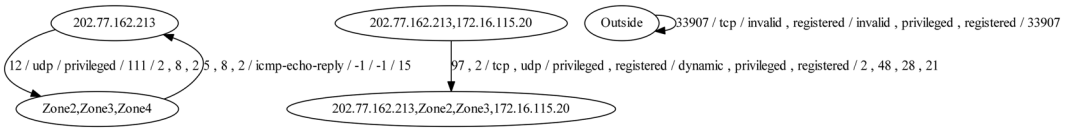

Figure 5. Generated hyper-alerts Graph for LLS_DDOS_1.0 attack scenario (MinPts = 10 and eps = 0.0001 in density-based spatial clustering of applications with noise algorithm).

Table V. Generated hyper-alerts for LLS_DDOS_1.0 attack scenario (MinPts = 2 and eps = 0.0001 in density-based spatial clustering of applications with noise algorithm).

ID	IP source	fr[1]	IP destination	fr	Protocol	fr	Port source	fr	Port destination	fr	Min time	Max time	Mean duration	Number of alerts	Phase indicator
1	202.77.162.213 172.16.114.50	2 1	202.77.162.213 172.16.114.50 172.16.113.50	1 1 1	icmp-d-u udp	1 2	Invalid privileged	1 2	Invalid privileged	1 2	'10:11:09 AM'	'10:12:08 AM'	0	3	Noise alerts
2	202.77.162.213	767	Zone2 Zone3 Zone4	5 508 254	icmp-req	767	−1	767	−1	767	'9:51:35 AM'	'9:52:01 AM'	0	767	Phase 1
3	Zone2 Zone3 Zone4	5 10 3	202.77.162.213	18	icmp-rep	18	−1	18	−1	18	'9:51:35 AM'	'9:51:59 AM'	0	18	Phase 1
4	202.77.162.213	10	172.16.114.2 Zone3 Zone4	1 7 2	UDP	10	Privileged	10	111	10	'10:07:07 AM'	'10:17:10 AM'	0.000637	10	Phase 2
5	202.77.162.213	76	Zone2 Zone3 172.16.115.20	42 20 14	UDP	76	Privileged	76	Dynamic privileged registered	17 38 21	'10:08:06 AM'	'10:34:58 AM'	0	76	Phases 2 and 3
6	202.77.162.213	20	Zone2 Zone3 172.16.115.20	9 5 6	TCP	20	Privileged registered	1 19	Privileged	20	'10:33:14 AM'	'11:26:14 AM'	0.000127	20	Phases 3, 4, and 5
7	202.77.162.213	5	Zone3 172.16.115.20	3 2	TCP	5	Privileged registered	3 2	Privileged	5	'10:35:00 AM'	'11:34:21 AM'	0	5	Phases 3, 4, and 5
8	Zone3 172.16.115.20	3 1	202.77.162.213	4	TCP	4	1023	4	514	4	'10:50:01 AM'	'10:50:38 AM'	0	4	Phase 4
9	172.16.112.10 172.16.115.20	1 3	202.77.162.213	4	TCP	4	1023	4	514	4	'10:50:01 AM'	'10:50:20 AM'	1.16E-05	4	Phase 4
10	Zone3 172.16.115.20	2 3	202.77.162.213	5	TCP	5	Privileged	5	Privileged	5	'10:50:07 AM'	'11:27:04 AM'	0	5	Phases 3, 4, and 5
11	131.84.1.31	32 838	Outside	32 838	TCP	32838	Invalid registered	4 32 834	Registered	32838	'11:27:50 AM'	'11:27:55 AM'	0	32838	Phase 5
12	Outside	1069	131.84.1.31	1069	TCP	1069	Registered	1069	Invalid privileged	1 1068	'11:27:50 AM'	'11:27:55 AM'	0	1069	Phase 5

UDP, user datagram protocol; TCP, transmission control protocol; icmp-d-u, icmp-destination-unreachable; icmp-rep, icmp-echo-reply; icmp-req, icmp-echo-request; fr, Frequency.

Table VI. The result of E-correlator for LLS_DDOS_1.0 attack scenario (MinPts = 2 and eps = 0.0001 in density-based spatial clustering of applications with noise algorithm).

	Number of raw alerts	Number of generated hyper-alerts(without running step 4)	Reduction ratio (%)
Phase 1	785	Two	99.74
Phase 2	25	Two (one of them is shared between Phases 2 and 3)	92.00
Phase 3	80	Four (one of them is shared between Phases 2 and 3) (three of them are shared between Phases 3, 4, and 5)	95.00
Phase 4	19	Five (three of them are shared between Phases 3, 4, and 5)	73.68
Phase 5	33 910	Five (three of them are shared between Phases 3, 4, and 5)	99.99
The total of raw alerts	34 819	11	99.97

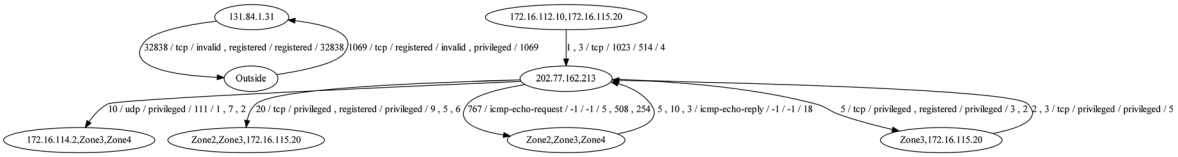

Figure 6. Generated hyper-alerts graph for LLS_DDOS_1.0 attack scenario (MinPts = 2 and eps = 0.0001 in density-based spatial clustering of applications with noise algorithm).