A Real-time Alert Correlation Method Based on Code-books for Intrusion Detection Systems

Ehsan Mahdavi , Ali Fanian , Fatima Amini

PII: S0167-4048(19)30203-2
DOI: https://doi.org/10.1016/j.cose.2019.101661
Reference: COSE 101661

To appear in: *Computers & Security*

Received date: 5 August 2019
Revised date: 29 October 2019
Accepted date: 4 November 2019

A Real-time Alert Correlation Method Based on Code-books for Intrusion Detection Systems

Ehsan Mahdavi[1]
e.mahdavi@ec.iut.ac.ir

Ali Fanian[1]*
a.fanian@cc.iut.ac.ir

Fatima Amini[2]
f_amini2000@sco.iaun.ac.ir

1- Abstract

Alert Correlation is the process of analyzing alerts to reduce their number, eliminate false positives, detect the scenarios behind them and generate a higher perspective of the incidents. Making this process online will upgrade the classic role of alert correlation from being a post-process step to a key part of intrusion detection systems. In this article, we propose a novel two-phase model called a Real-time Alert Correlation method based on Code-books (RACC) for intrusion detection systems. First, in the offline phase, RACC pre-processes a knowledge base to propose some matrices as the main data structure of the method that we call them code-books. Instead of keeping alerts in the memory, those matrices just hold keys to the corresponding meta-alerts. An index that is based upon red-black trees is used to access matrix elements. Generating the matrices and mentioned index are independent from the alerts, so utilizing them can facilitate the alert correlation process in an online manner in phase two of the proposed model. The experiments show that compared to similar methods, RACC can significantly reduce the alert correlation time and can enable real-time alert correlation.

Keywords: Network Security, Intrusion Detection Systems, Alert, Online alert correlation, Attack Scenario, Causal relationships, Code-books.

* Corresponding Author
[1] Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran
[2] Department of Electrical and Computer Engineering, Islamic Azad University, Najafabad branch, Iran

2- Introduction

As the Internet and computer network usage grows, security threats become more frequent. Appearance of modern technologies like mobile or Internet of Thing (IoT) and their issues which are studied in literatures like [1] and [2], affects cyber-threat landscape. As cyber-attacks evolve, it is important to evolve protection systems as well.

Intrusion Detection System (IDS) is one of the most important tools to prevent intrusion into computer networks. In literature, any activity to undermine the confidentiality, integrity, accessibility, or attempting to circumvent security mechanisms of a computer network is called intrusion [3]. In contrast, intrusion detection is the process of monitoring the incidents in a computer system or network and analyzing them to find the signs of an intrusion. Any software or hardware system or combination of both of them which is responsible to detect intrusions, is known as intrusion detection system.

As a negative side effect, huge volume of raw alerts generated by intrusion detection systems, confuses security managers. It is proven that a large amount of those alerts are commonly incorrect [4]. To overcome this problem, alert correlation techniques are usually employed. Alert correlation tries to provide a higher level view of security incidents by aggregating and combining alerts into a denser and more valuable form. This view can reveal attack scenarios and attacker's intentions. In other words, by extracting useful information from the huge volume of generated alerts, the alert correlation can be a supplement to intrusion detection systems. In most researches, during the alert correlation, raw alerts are changed into some other forms of alerts that are usually called meta-alert or hyper alert. These alerts are usually a denser form of the original alerts that might point to a collection of alerts, a simplified form of alerts or an aggregation of them. The output of the process might be a set of these intra process hyper alerts or undergo more changes.

As mentioned, the key purpose of alert correlation is to create a higher level view of attacker's intentions and how they are achieved. As most alert correlation methods reduce the volume of the alerts, this can be considered as a main by-product of this process. In a general taxonomy given in [5], three important strategies including similarity, sequential and case based approaches are proposed to correlate the alerts.

Real-time alert correlation has not been focus for most of the researches in this field. A brief search in online scientific index engines would clarify that real-time alert correlation was discussed in less than 50% of the literature. Therefore, in this article, using the causal relationships between the alerts, we create some code-book matrices in order to detect the attack scenarios. The code-books were inspired from network management systems (NMS) techniques [6]. In those matrices, the rows represent the possible problems in the network and the columns represent a sign of a problem. The use of matrix for alert correlation and a Red-Black tree to achieve the indices of those matrices, boosts the method speed. This enables the proposed method as called RACC to be used in online applications. Simulations show that used resources for the method makes it suitable for online correlation of the alerts.

In the following a review of previous works and a general taxonomy of existing methods is presented. Section 4 describes the RACC method. Implementation results in section 5 shows that the proposed method can be used for online alert correlation purposes. Finally, section 6 will conclude the research.

3- Related work

As discussed in the previous section, IDS alert correlation methods employ three major strategies. In this section, we review the relevant researches from these categories.

1-3 Similarity based methods

These methods usually try to reduce the total number of similar alerts through clustering and aggregation.

Valdes and Skinner [7] have proposed a probabilistic approach that can combine alerts from multiple sensors. This method uses a mathematical framework to estimate the minimum requirement to specify the similarity between the alerts. The basis of this approach is to define a particular similarity measurement function for each characteristic type. Since the values in an alert may take a range of values, the similarity measurement function must be able to determine how the corresponding values of a characteristic overlap for different alerts. Using such calculations, alerts with high similarity scores are considered to be correlated.

To correlate the alerts while identifying steps of attacks, a two-tier feature selection method has been presented in [8]. This method tries to select the appropriate features of the alerts. In the first layer, features are sorted in descending order with respect to their information gain. Then a subset of them with most scores form the early candidates. In the second layer, features that have a better separation capability than the earlier ranked features are added. This feature selection tries to detect scenario.

Daneshgar et al. [9] have proposed a model that consists of an online as well as an offline module. The online module clusters the alerts according to their similarities to some fuzzy patterns. In the offline module, a statistical analysis is performed based on statistical characteristics. The output of the offline module is considered as input to the online module and the iterative fuzzy patterns are extracted. The presence of these two modules contribute to the accuracy of the proposed method. It can be considered as an online method which extracts the scenario of complex attacks.

The method in [10] uses the integration of Particle Swarm Optimization (PSO) and k-means to correlate the alerts. In order to balance the dataset, the nominal features are converted and scaled before clustering. Also the numerical features are scaled to the range of [-1, 1]. The method uses PCA to reduce the features describing the alerts. After all, it integrates PSO and k-means clustering to aggregate the alerts.

As can be seen in this section, similarity based methods for alert correlation are a subset of general methods including probabilistic solutions, statistical methods and even machine learning methods which are vastly used in different areas of computer science. They are adapted to solve the correlation problems. Their general nature makes them suitable to be adjusted for solving different network security problems like detecting last generation attacks [11] from raw network traffic.

Despite the low complexity, the methods in this category have proven their efficiency for reducing the alerts. Inability to find the causal relationships between the alerts and the origin of the problems is the main weakness of these methods. Finding causal relationship of alerts with shared origin, can play an important role in identifying the motives and effects of different attacks.

2-3 Sequential based methods

Special interest on causal relationships between the alerts is the most important characteristic of such methods. The prerequisites of incidents as well as their possible consequences are presented in these approaches by different means.

Causal graphs are widely used to model those relations. In such methods, relations between the alerts are commonly represented using Directed Acyclic Graphs (DAG). Each node in a DAG represents an alert or meta-alert. Each connecting edge, presents a relationship between those nodes.

Roschke et al. [11] have used Floyd-Warshall algorithm to find shortest paths in the attack graph. This made it possible to detect different attack scenarios. Each node of their graph represents a step in a scenario and is equivalent to an alert. Every attack step requires occurrence of one or more previous steps. Input edges represent those prerequisites. On the other hand, output edges are connected to incidents that need this step as their prerequisite to occur.

In [12], a method is proposed to correlate the alerts for Early Warning systems (EWS). It consists of an online as well as an offline phase. In the offline phase, the alerts are first aggregated together based on their similarity. This aggregation creates some hyper alerts. Then, the hyper alerts are categorized into episodes with a specific maximum length. Afterwards, the frequent critical and benign episodes are extracted. Critical episodes are used to create the attacks tree. This tree is a model of the attacker behavior. Benign episodes might be used to identify new attacks. By this means, multi-step attack scenarios are recognizable. Considering the models generated in the previous (offline) phase, online phase generates trees locating current alerts. Another data structure in this method is a matrix called CCM[1]. Each element in CCM determines the correlation strength.

Soleimani et al. [13] have presented a multi-layer framework. It reduces the search load among large number of alerts to explore the attack scenarios. The method first aggregates the alerts based on their similarity. An episode mining algorithm is then used to explore possible combinations of alerts. The episode miner, creates episodes with various lengths using aggregated alerts. Similar to [12], their work is based upon critical episode concept. They also utilize decision-trees to determine multi-step attacks.

In a study by Zali et al., causal relation graph (CRG) is proposed. In CRG two types of vertices are defined. Condition vertices and alert signature vertices[14]. For each vertex in CRG, Forward Queue and Backward Queue Trees are extracted. CRG is likely to be a template and correlation results are stored by locating alerts in instantiated graphs and corresponding trees. Thus, in this method for each typical alert, there are two trees, a signature node and a condition vertex which stores alert details. All this information is stored in the main memory.

[1] Causal Correlation Matrix

Additionally, for each incoming alert, a backward search is carried out. It might require significant processing or the search range should be limited.

With the goal of processing a large amount of alerts, a real-time alert correlation approach is presented in [15] which adopts [14]. The approach is based on the Attack Planning Graph (APG). In order to generate an attack scenario, the APG model is created based on the prior knowledge of the attacks. APG is a directed acyclic graph that contains the causal relationship between attacks. According to the generated graph, the alerts can be analyzed. APG includes a set of ST and AT nodes. The ST nodes indicate the attributes or states and the AT nodes specify the attack types. To correlate the alerts, BFS and backward BFS are applied on the APG. These searches create two trees (Forward Queue and Backward Queue Trees) for each APG node. In this approach, the attack maps and planning trees are constructed by correlating the similar alerts in offline phase. Duplicate alerts will be reduced significantly in this phase. Based on the opinion of authors of this paper, the method presented in [14] cannot correlate some kinds of similar alerts and the storage overhead is high. These drawbacks of [14] motivated them to propose improvements.

In [16], Normalizing and reducing the alerts volume is the first course of action. This step tries to keep important alerts features. Afterwards, the alerts are processed within certain window length. In each window, the most frequent sequences of alerts are determined. Then a causal correlated matrix (CCM) rules the sequences to be correlated together. The correlated alerts are listed in table called CFSP[1]. Every time a number of alerts form a window, the most frequent sequences are obtained. Then, larger CFSPs are extracted based on the values of current CFSP table and the existing relationships in the CCM matrix. The CCM matrix is also updated based on the frequent sequences of each window and some rules are established. To be more accurate, after processing the alerts, the attacks and future behaviors of the attacker are predictable based on a learned hidden Markov model.

Kliger et al. [6] have presented a method using code-books to correlate alerts in network management systems (NMS). The method’s salience is to perform well while tolerating high volumes of false positives particularly for the cases when the alert loss is high. To do so, the authors have reduced the code-book size and tried to consider a subset of signs that can distinguish the issues. They have used two types of measures. The first one is the Hamming distance which is used for deterministic alert correlation when the matrix elements are equal to zero or one. The other measure employs a probabilistic method in which the code-book matrix contains weight coefficients in the interval of [0, 1]. Each of weights indicates the probability of a relationship between a specific sign and the investigated incident.

Steinder and Sethi [17] have employed Bayesian methods on network errors. Their system effectively solves the issues of false positive alerts and corrupted or lost alerts. They used two Bayesian inference algorithms to distinguish between different errors.

Zhu et al. [18] have used the combination of a multi-layer neural network (MLP), a support vector machine (SVM) and a knowledge representation scheme called Alert Correlation Matrix (ACM). It should be noted that ACM shows the correlation strength.

In sequential based methods, no matter how much the scenarios follow, the steps usually link to each other in a chain. This provides scalability of these methods. Moreover, comparing the prerequisite of an incident with the consequence of the preceding incidents indicates the causal relations appropriately. At a more abstract level, sequential based methods can identify new scenarios. For this purpose, it is enough to ignore the incidents and derive possible scenarios using the prerequisites and consequences. Usually, the results of these types of alert correlation methods are easily understandable and they directly express the possible scenarios of attack. However, defining poor logical relations or using IDSs with low quality alerts may result in wrong correlation scenarios.

3-3 Case based methods

These methods commonly rely on the existence of a knowledge base system to provide them with appropriate scenarios.

Liu et al. [19] have proposed an alert correlation system based on finite automata. This method investigates the scenarios in three types of high-level views. However, to process critical scenarios, an NFA is used to reconstruct the scenarios between an attacker and a target.

Case based methods are very efficient to correlate the known scenarios, but the creation of a database of all feasible scenarios is almost impossible. So the concern about unknown scenarios always exists. On the other hand, expanding the set of

[1] Correlated Frequent Sequential Pattern

scenarios will increase the search cost and this challenges these methods for online utilization.

−4 Proposed method

Sequential methods particularly try to model cause/effect relationships between incidents. This becomes prominent for prerequisite/consequence based subcategory. An ideal situation for such methods is to be able to predict next moves of malicious users by analyzing current incidents. This brings ability to embed suitable reactions into systems. The goal is not met unless such analyses can be done in real time. In this paper, we propose a new online method to provide such advantage for prerequisite/consequence based methods. Figure 1 shows the overall schema of the proposed method RACC. In the next sub-sections, we will describe the method in more detail.

1-4 Description of the proposed RACC model

In this section, we describe the proposed method in detail. Our main concern for the proposed method is to provide alert correlation online. Considering the ability of sequential based methods to express attack scenarios, the proposed method tries to create a real-time prerequisite/consequence based method. To do so, we first extract possible scenarios from a prerequisite/consequence knowledge base. This process is performed offline and the results will be placed in a specific data structure. The data structure should be designed properly to be used quickly. In RACC, it is composed of a collection of matrices. In more definite terms, there will be a matrix for every scenario extracted from the prerequisite/consequence relationships. A Red-Black tree is also used as a map to ease and speed up access to those matrices and specific indices. Additionally, instead of keeping all the alerts and meta-alerts in the main memory, this method only keeps the indices for them in the main memory. This will effectively reduce required memory. Once the indices were created in the offline phase, the alert correlation can be performed online in real time.

As mentioned earlier, the main data structure in the proposed model consists of a number of code-book matrices. Section 3-2 cited such matrices which were used for alert correlation in the network management systems (NMS). The code-books in those systems are constructed with possible problems in a network on the matrix rows and the signs of different problems appear as the matrix columns. In a given matrix, an element with the value of 1 represents the occurrence of a sign for a problem and indicates a cause/effect relationship between them. Then, to correlate the various alerts, usually a distance criterion between the matrix rows is used. This criterion can determine how much a problem is in relation with other problems for the same network.

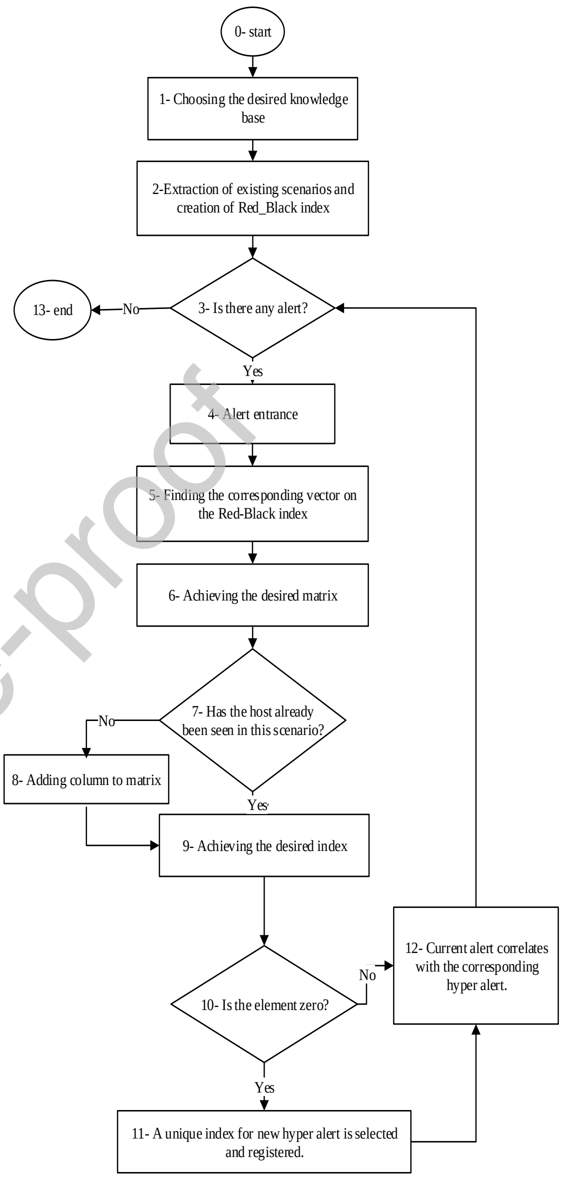
0- start
1- Choosing the desired knowledge base
2-Extraction of existing scenarios and creation of Red_Black index
3- Is there any alert?
No
13- end
Yes
4- Alert entrance
5- Finding the corresponding vector on the Red-Black index
6- Achieving the desired matrix
7- Has the host already been seen in this scenario?
No
8- Adding column to matrix
Yes
9- Achieving the desired index
10- Is the element zero?
No
12- Current alert correlates with the corresponding hyper alert.
Yes
11- A unique index for new hyper alert is selected and registered.

Figure 1: Overview of the proposed RACC method

To the best of our knowledge, code-books have not been used for alerts correlation in the intrusion detection systems. So we declare the code-book as a type of matrix which we propose to correlate the alerts in the intrusion detection systems.

1−1−4 Proposed code-book

The *Mx* matrix as a code-book in the proposed method, is a matrix that maps occurrence of different steps from a multi-step scenario (the scenario x^{th}) to different systems protected by the IDS. To do so, each matrix row should represent a step of the intended scenario and the matrix columns represent a protected system. For instance, matrix (1) shows such a matrix for an attack scenario that consists of four steps.

$$Mx = \begin{array}{c} \\ Email\ Almail\ Overflow \\ FTP\ Put \\ Mstream\ Zombie \\ Stream\ Dos \end{array} \begin{array}{c} \begin{array}{cccc} IP1 & IP2 & \cdot & \cdot \end{array} \\ \left[\begin{array}{cccc} 1 & 0 & \cdot & \cdot \\ 1 & 0 & \cdot & \cdot \\ 0 & 1 & \cdot & \cdot \\ 1 & 1 & \cdot & \cdot \end{array}\right] \end{array} \quad (1)$$

Therefore, the value of 1 on the ith row and jth column of the matrix, shows the step i from the scenario x is occurred on the system j (IP_j). Obviously on a column, one or more 0 valued elements between two elements with a value of 1, indicates that the corresponding alerts are somehow lost. This could be due to IDS weaknesses or a clever attacker has hidden his moves by omitting some of them. However, in this method, the meta-alerts are obtained by aggregating the alerts associated with elements with a value of 1 on same column. Hence such meta-alerts highlight three key concepts. **Which** steps of **which** scenario occurred on **which** host so far. For a progressing scenario, a quick glance on the entries of a column of the proposed matrix, can reveal next possible steps. This can be done simultaneously with the current alert processing. The output can be presented as feedback to the intrusion detection system.

To distinguish relationship between the alerts and corresponding meta-alerts, it suffices to store an integer instead of a bit as matrix elements. In this case, the value of each element is either zero (which mean the corresponding step has not reported on a host) or a unique index for the corresponding meta-alert. This index can reveal the index in a relational database or is considered to be unique file name. So in the proposed method, each column indicates the execution of different steps of a scenario on a particular host. The corresponding alerts will be correlated as a meta-alert. Thus, each scenario on each host is specified with a unique index. These indices are the main information required to correlate the alerts. Consequently, by having them in the main memory, RACC doesn't keep the alerts and meta-alerts in the main memory. This results in a significant reduction of required memory. Utilizing this mechanism, if there is a specific attack in different matrices, it means that the attack exists in several scenarios. As with the most alert correlation methods, there is a meta-alert for each scenario, but only the corresponding index will be stored in the proper matrix.

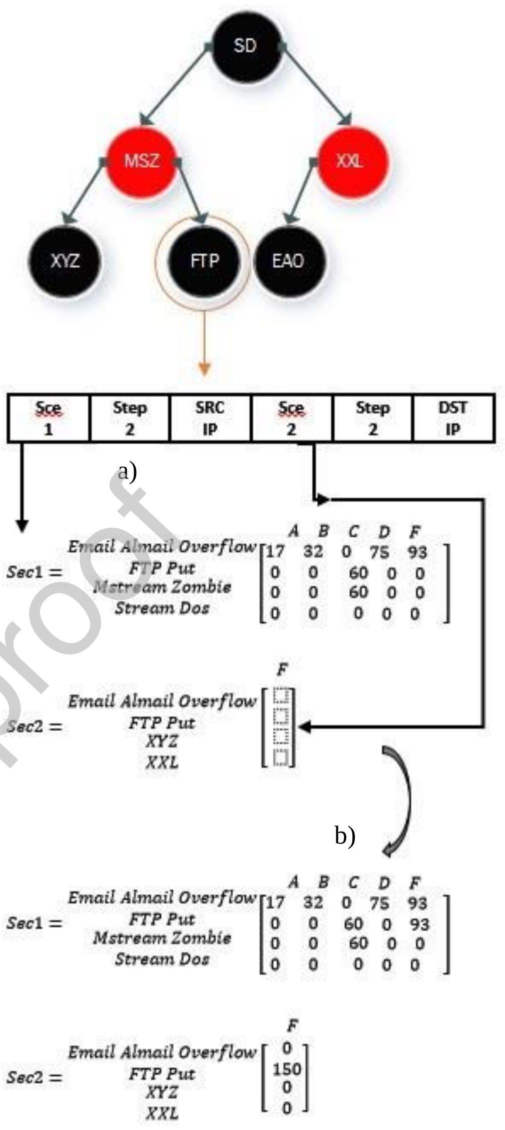

Figure 2: Accessing and modifying of code-books when an alert occurs

2−1−4 Details of the proposed model

As stated before, there is a matrix for every scenario and a typical alert might be used in multiple matrices (scenarios). To be suitable for real-time applications, RACC must provide quick access to locations that refer to typical alert. In more definite terms, all row indices corresponding a typical alert. For this purpose, a recursive program, pre-processes the prerequisite/consequence knowledge base. It explores existing prerequisite/consequence definitions and each extracted scenario is stored in a vector. The knowledge base used in this phase, describes the typical alerts of IDS with XML structure. Each XML entry contains the prerequisites and consequences of the intended alert. The preprocessor, also stores the vector index and the position of the typical alert in that vector, to another data structure. The vector index corresponds to each typical alert. The typical alert is called hyper alert in the literature.

Another challenge is to find these indices. To do so, a Red-Black tree is used. The red-black trees are suitable as search trees. They ensure there is no path which is larger than any other route by more than two steps [20]. Therefore, these trees are almost balanced and their search complexity reveals maximum order of $\log_2 n$. The key index in this scenario is a Red-Black tree. Each node points to a vector of ordered pairs (vector index, index in the vector). This pair reveals (the matrix corresponding to a scenario, index of the matrix corresponding to the scenario step). It is notable that the hyper alert text is the key to sort the Red-Black tree.

Following is the explanation of working of RACC. For each input alert, RACC retrieves vector of the corresponding element from the mentioned red black tree. The vector, points to the intended element in the code-book matrix. Supposing the case that there are *n* types of different alerts in data set, the Red-Black tree search will be carried out in the order of $\log_2 n$. In this phase, according to the limited number of scenarios, the search in the corresponding vector is done at constant time. Assuming there are *r* scenarios in the knowledge base and each one has maximum *m* steps, the search will be done in the order of $(r \times m)$. Evaluating or modifying the intended element is also a constant process and can be performed in the order of $O(1)$. Therefore, the entire process can be performed in order of $O(\log_2 n) + O(r \times m) + O(1)$.

It should be noted that all these values are constant; they are not related to the number of security incidents and corresponding alerts in the execution time. So, these values appear as a constant factor to correlate each alert and the general order of alert correlation is linear with respect to the number of alerts.

When an alert in reported on a target host, the red black tree guides to the codebook and corresponding row. If there is a column in the matrix corresponding to that host, this column's elements are checked. If they are all zero, it means that no step of the corresponding scenario has been reported on the host. Therefore, it is necessary to create a meta-alert and a unique index as the corresponding element in the code-book. The unique incident number can also be stored as a link between meta-alert and current alert in a relational database or corresponding files. If a non-zero value is found on the column, then the corresponding meta-alert is already generated and its index should be used directly for the current element. For memory optimization purpose, the columns might be dynamically added to the code-books. In this case, if there is no column corresponding to the intended host, then a column should be added to the matrix when an alert is reported in a particular scenario.

Figure 2 shows how to use the Red-Black index and the code-books at the point of an alert entrance of type FTPPut. In this example, this type of alert exists in scenarios *Sce1* and *Sce2*. A search in the Red-Black tree yields these scenarios (code-books). The corresponding vector and the matrices (via this vector) are consequently available. For example, suppose the mentioned alert is reported on host *F*. Part *a* shows that the previous step (Email Almail Overflow) has occurred for the same host in the first scenario and the corresponding meta-alert key is 93. So in part *b* this key is considered for the corresponding row. There is no alert in part *A* for the *Sce2* scenario. Therefor a column for the host *F* will be added into the matrix and meta-alert number, for example, is considered to be 150.

So far, the proposed method has been described in details. The use of code-book and the Red-Black index will result in dramatic speed of alert correlation in the online phase. To verify the validity of our claim, we will evaluate the proposed method in Section 5.

−5 Evaluation of the proposed method

As mentioned earlier, the main goal of the proposed method is online enforcement capability. To increase speed, the main action was to use several matrices as well as a Red-Black index. The proposed components are also designed to make the least use of resources like main memory. To investigate the feasibility of online application, in this section we examine implementation results of the method. In the following, we first examine the required main memory and then we estimate the execution speed of this method in terms of CPU time.

1-5 Required memory

Perhaps the first concern about the proposed method is the vast memory consumption. With an analysis based on a pessimistic practical experience, it will be specified that this concern does not hold. Suppose that a large network with 1000 protected hosts has been selected to use the RACC method. For 10000 scenarios where the average length of each scenario is 20 steps, we will need 10000 matrices of size (1000×20). It is notable that this is a strict mode. Because each element of the matrix occupies only 1 bit of memory space, we need $1000 \times 1000 \times 20$ bits. So only 25 MB of memory space will be occupied. Even using the integer numbers as indices of meta-alerts, the required memory space will not exceed 800 MB. Typical personal computers these days are assembled with 8 to 16 GB of main memory. Thus the required memory space is likely to occupy 5 to 10 percent of the main memory and the proposed matrix operation can be executed very quickly. In addition, the columns of the matrices can dynamically be added to the data structure. That is, if one of the alerts related to one step from a scenario is seen on a new host, a column is added to the corresponding matrix. So in real applications, the required main memory will stay below 10% for RACC.

2-5 Implementation method and datasets

The proposed method has been implemented using C++(11) in an Ubuntu distribution of Linux environment. In the pre-processing phase, the method requires a prerequisite/consequence knowledge base to extract the attack scenarios. To do so, the two knowledge bases corresponding to the alerts of two known intrusion detection systems have been used. The IDSs are SNORT [21] and RealSecure [22] and dataset are obtained from [23]. The first knowledge base provides the prerequisite/consequence relationships between around 3000 hyper alerts in SNORT and the second knowledge base contains prerequisite/consequence relationships between 30 different hyper alerts in RealSecure. As shown by Table 1, the proposed method is implemented for three data sets using these two knowledge bases.

Table 1: Used datasets and databases

IDS	Knowledge base	Dataset
RealSecure	[23]	DARPA2000 [24]
SNORT	[25]	DARPA2000 [24]
		TH [26]

DARPA2000 dataset consists of two scenarios of DDoS[1] attack. The first scenario has 5 steps and its final goal is to conduct a DDoS attack. Figure 3 illustrates the steps involved in this scenario. In the second scenario of DARPA2000, a smarter attack scenario has been carried out. In this scenario, instead of sending requests to all the hosts as in the first step of the previous scenario, requests of type HINFO are sent to a DNS[2] provider. Another difference in this scenario is that the attacker, instead of performing all the attack steps from his/her machine, first penetrates into another machine and then uses it to perform the latter steps. Figure 4 indicates the steps of the attack.

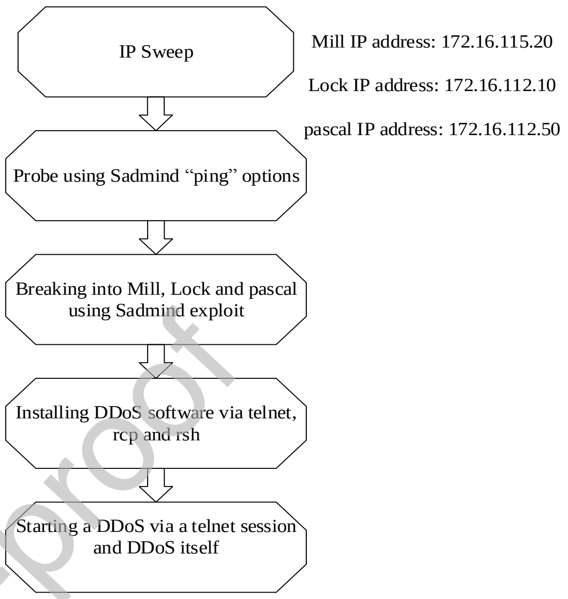
IP Sweep
Mill IP address: 172.16.115.20
Lock IP address: 172.16.112.10
pascal IP address: 172.16.112.50
Probe using Sadmind "ping" options
Breaking into Mill, Lock and pascal using Sadmind exploit
Installing DDoS software via telnet, rcp and rsh
Starting a DDoS via a telnet session and DDoS itself

Figure 3: Steps for implementing LLDOS 1.0 in DARPA2000

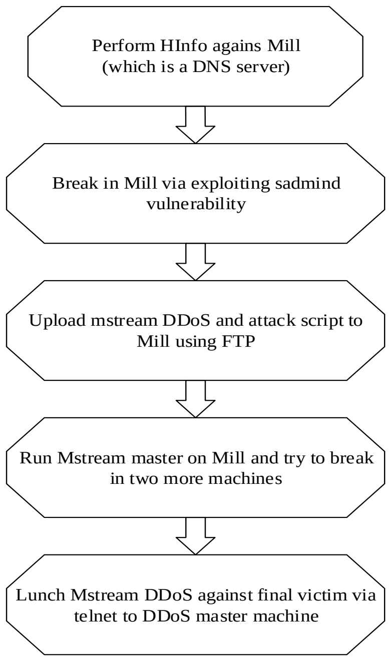
Perform HInfo agains Mill (which is a DNS server)
Break in Mill via exploiting sadmind vulnerability
Upload mstream DDoS and attack script to Mill using FTP
Run Mstream master on Mill and try to break in two more machines
Lunch Mstream DDoS against final victim via telnet to DDoS master machine

Figure 4: Steps for implementing LLDOS 2.0 in DARPA2000

[1] Distributed Denial of Service

[2] Domain Name Service

The dataset TH [26] contains alerts generated by Snort IDS for real traffic of a private corporation which is provided for this research.

3-5 Alert correlation results of the proposed RACC method

The result of applying RACC on the SNORT alerts for the first scenario of DARPA2000 is the matrix shown in Figure 5. It depicts that the first step of the attack has been applied on all the hosts. The next two steps of the attack have been performed on three of those hosts. This can be seen that they are correlated and generated corresponding hyper alerts. The hyper alert index is stored in related columns of code-book.

So far the code-book mechanism only considers the destination address of the attack as matrix columns. In some scenarios like the second of DARPA, the victim computer(s) stay to be attack destination up to a certain step. From that point onward, when the host is exposed, the attacker exploits it and performs next steps with that host. From the alerts point of view, this means that the abused destination is the source of next malicious actions. However, these alerts should be correlated together. For example, in second DARPA scenario, the final step generates an alert with destination IP 202.77.162.213. So this address appears as a column in the matrix with no prior steps performed. The results are not suitable because RACC won't correlate the last step with prior steps. It is possible to correct this defect by using information in knowledge base. The existing knowledge base, outlines the argument type that relates each prerequisite and consequence. This can be reflected by considering another bit in the index for each hyper alert. This bit specifies if the argument type is destination or the source IP. Considering this correction, the matrix in Figure 5 can be corrected to the matrix in Figure 6. This can be generalized so that type of the columns may vary. This has been reflected in Figure 6 as triplet vectors. Replacing the mentioned bit with a byte can extend the method coverage to 256 types of key arguments. However, as can be seen in figures, the proposed matrix just keeps some indices of the information which are necessary to fast alert correlation. The code-books are compact enough to maintain memory efficiency while preserving high speed capabilities. To support these claims, Figure 7 illustrates the alert correlation graph of the method proposed in [14] for RealSecure alerts on the first DARPA2000 scenario and Figure 8 shows the corresponding matrix in the RACC. As you can see, Figure 7 shows the overall alert correlation schema with no information such as what was the target host or in which meta-alert the correlated alerts were aggregated. Contrarily, RACC matrix contains such information at the same time, it provides them more compact and more efficient manner. Furthermore, accessing the alerts of the graph presented in Figure 7 and also processing them requires cumbersome graph traversals. This is while similar tasks in the corresponding RACC code-books are simpler and faster. As the typical number of alerts in knowledge base grows, such as SNORT, graph dimensions are subject to considerable increase in size so they cannot be figured in this paper. Thus such visual comparisons can't be performed. The comparison of RACC with numerous and large graphs proposed in [14] and other graph based methods illustrates the advantages of the proposed method.

Figure 8 and Figure 9 show the matrix derived from correlating RealSecure alerts for both DARPA2000 scenarios. As illustrated, all the steps of attack are specified in the proposed matrix.

	IP = 135.15.216.191	**IP = 172.16.112.110**	**IP = 172.16.112.10**	**IP = 172.16.112.105**	**IP = 172.16.112.149**	**IP = 172.16.112.194**	**IP = 172.16.112.20**	**IP = 172.16.112.207**	**IP = 172.16.112.50**	**IP = 172.16.112.105**	**IP = 172.16.112.204**	**IP = 172.16.113.50**	**IP = 172.16.115.20**	**IP = 172.16.115.30**	**IP = 202.77.162.213**
ICMP PING	73	180	283	334	385	436	487	530	589	640	844	895	946	1150	**0**
RPC Sadmind UDP PING	0	0	283	0	0	0	0	0	589	0	0	0	946	0	**0**
RPC Sadmind query with root credentials attempt	0	0	283	0	0	0	0	0	589	0	0	0	946	0	**0**

RSERVICES rsh root	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1230**

Figure 5: Attack matrix on the SNORT alerts for the first scenario of DARPA2000

	IP = 135.15.216.191	**IP = 172.16.112.110**	**IP = 172.16.112.10**	**IP = 172.16.112.105**	**IP = 172.16.112.149**	**IP = 172.16.112.194**	**IP = 172.16.112.20**	**IP = 172.16.112.207**	**IP = 172.16.112.50**	**IP = 172.16.112.105**	**IP = 172.16.112.204**	**IP = 172.16.113.50**	**IP = 172.16.115.20**	**IP = 172.16.115.30**
ICMP PING	73	180	283	334	385	436	487	530	589	640	844	895	946	**1150**
RPC Sadmind UDP PING	0	0	283	0	0	0	0	0	589	0	0	0	946	**0**
RPC Sadmind query with root credentials attempt	0	0	283	0	0	0	0	0	589	0	0	0	946	**0**
RSERVICES rsh root	0	0	283	0	0	0	0	0	589	0	0	0	946	**0**

Figure 6 : New attack matrix on the SNORT alerts for the first scenario of DARPA2000

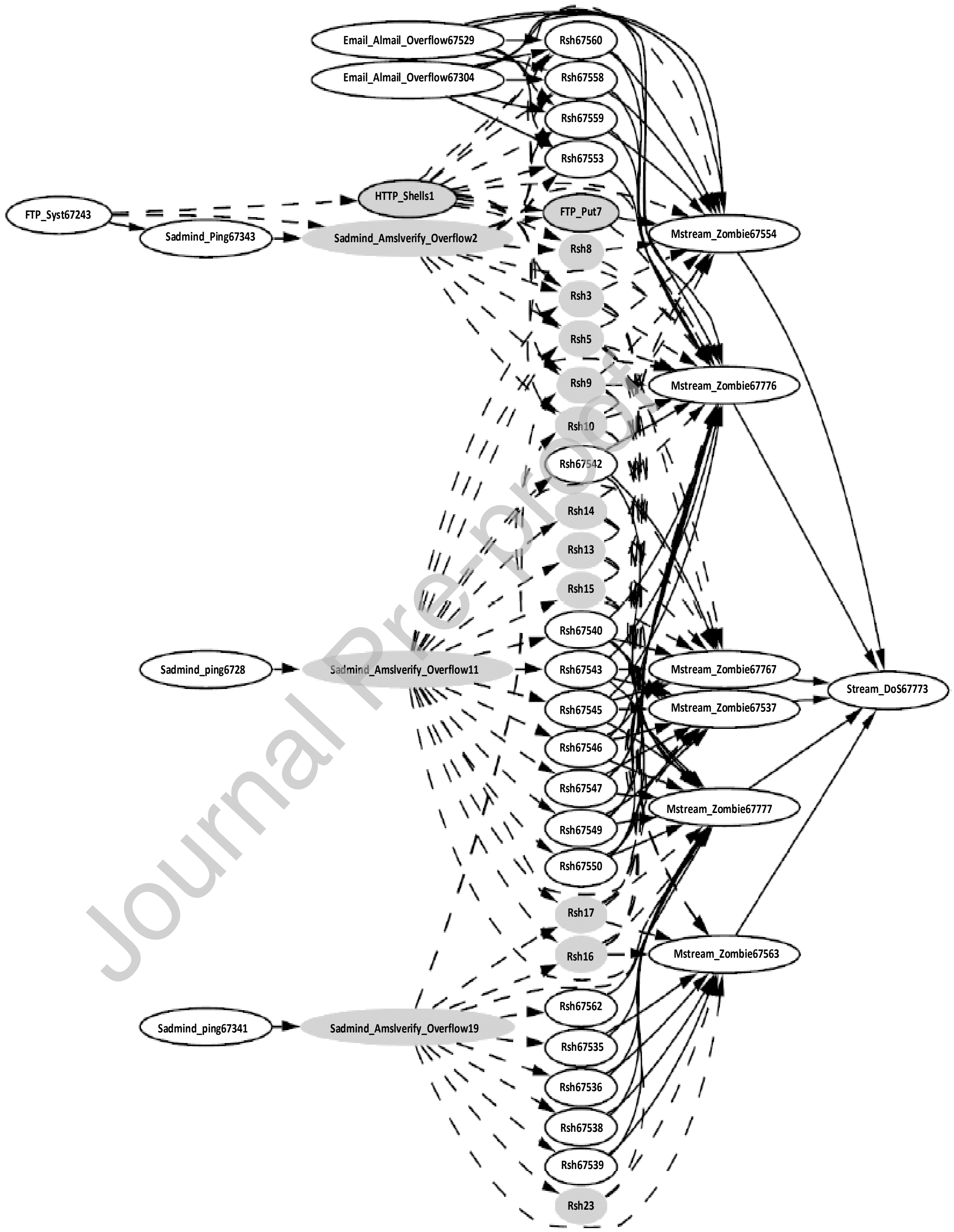

Figure 7: Proposed graph in [14] to correlate RealSecure alerts for the first scenario of DARPA2000

	IP = 172.16.112.10	**IP = 172.16.112.50**	**IP = 172.16.114.10**	**IP = 172.16.114.20**	**IP = 172.16.114.30**	**IP = 172.16.115.20**
Sadmind Ping	54	64	66	69	100	**103**
Sadmin Amslverify Overview	54	64	66	69	100	**103**
Rsh	54	64	66	69	100	**103**
Mstream Zombie	54	64	0	0	0	**103**
Stream DoS	54	64	0	0	0	**103**

Figure 8: Attack matrix on the RealSecure alerts for the first scenario of DARPA2000

	IP = 131.84.1.13	**IP = 172.16.112.194**	**IP = 172.16.113.105**	**IP = 172.16.113.148**	**IP = 172.16.113168**	**IP = 172.16.113.169**	**IP = 172.16.113.204**	**IP = 172.16.113.207**	**IP = 172.16.115.20**
Port Scan	0	0	0	0	0	0	0	0	**0**
Email Almail Overflow	55	5	91	93	95	97	99	115	**133**
FTP Put	55	0	0	0	0	0	0	115	**0**
Mstream Zombie	55	0	0	0	0	0	0	115	**0**
Stream DoS	55	0	0	0	0	0	0	115	**0**

Figure 9: Attack matrix on the RealSecure alerts for the second scenario of DARPA2000

1−3−5 Evaluating the online performance

After evaluating the validity of the proposed method, we examine the validity of the high-speed claim. The most important metric used to examine the speed of the proposed method is the CPU time to execute the desired operations. Research conducted by Zali et al [14], has the most similar goal with RACC. The method was proposed to make a prerequisite/consequence base approach online. To compare with [14], experiments were first performed on a personal computer similar to [14]. The SNORT knowledge base, whose prerequisite/consequence relationships are available, has about 3800 hyper alerts and 644 possible scenarios. In RealSecure case, there are 30 hyper alerts and 45 scenarios, respectively.

Figure 10 and Figure 11 show the mean and maximum processing times for the proposed RACC method and the method in [14]. The methods have been applied on the SNORT alerts and its knowledge base for the first scenario of DARPA2000. The first observed difference is a significant reduction of processing time in the proposed RACC method. It should be noted that Chrono library in C++(11) provides the most accurate measure of CPU time in nano seconds.

Observations show that the processing time in the method presented in [14] increases continuously. This is due to the queuing nature of the warnings in the quoted trees. In this case, for the scenarios with more steps, the number of alerts are increased in queues. This delay is cumulatively propagated to latter alert processing. Additionally, this alert queuing procedure is intensified because they should be replaced in graphs with new hyper alerts. Another factor is to add new nodes to the resulting graph of the mentioned method. On the other hand, there is no queue in RACC. The trends are uphill for processing scenarios with more steps, but the charts are immediately descending for short scenarios. Considering maximum CPU time in Figure 11 depicts that the highest processing time for each 25 consecutive alerts is about 50 microseconds, while [14] is 6 times slower and processes same episode in about 300 microseconds.

According to Figure 10, between arriving alert number 700 and 1000, RACC has progressed almost without any delay, but the time consumed in [14] has increased continuously. Checking those alerts, shows that they are those alerts that have not been used in any scenario. In this case, the RACC simply does nothing. But [14] performs all queuing, searching, traversing graphs and locating them in graphs.

In another experiment, the proposed method and the method in [15] are compared with each other using similar hardware. Figure 12 and Figure 13 depict the mean and maximum processing times for RACC method and the method in [15]. The methods have been applied on the SNORT alerts using the knowledge base for the first scenario of DARPA2000. The method in [15] can be considered as an extension of [14], so it shows similar behavior as it uses queuing, backward searches and graph traversals. Thus the same observation and explanations will hold. As is clear, RACC has a significant reduction in alerts processing time.

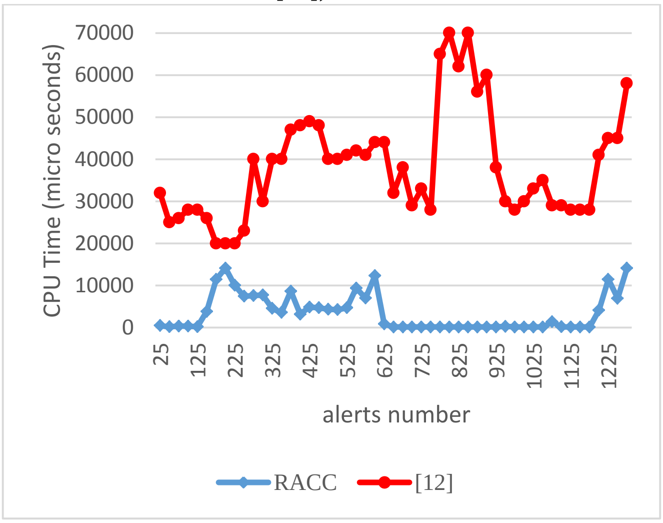

Figure 10 : Comparison of average CPU time to process every 25 consecutive SNORT alerts for the first scenario of DARPA2000 between RACC and the method in [14] with similar hardware

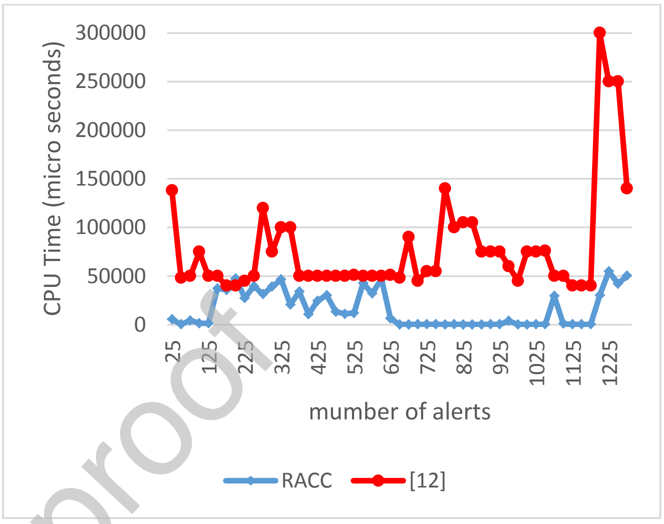

Figure 11 :Comparison of maximum CPU time in the processing of every 25 consecutive SNORT alerts for the first scenario of DARPA2000 between RACC and the method in [14] with similar hardware

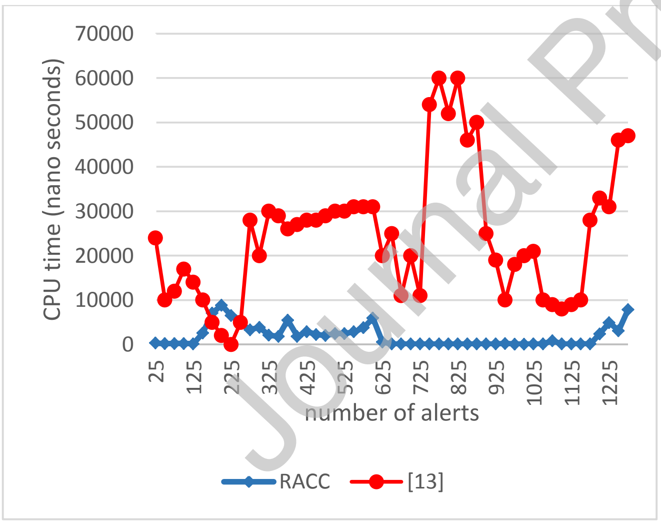

Figure 12 :Comparison of average CPU time to process every 25 consecutive SNORT alerts for the first scenario of DARPA2000 between RACC and the method in [15] with similar hardware

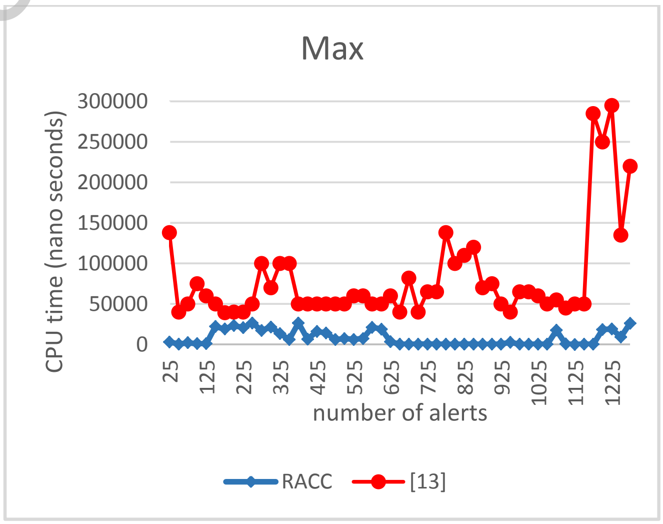

Figure 13 : Comparison of maximum CPU time in the processing of every 25 consecutive SNORT alerts for the first scenario of DARPA2000 between RACC and the method in [15] with similar hardware

In another evaluation in terms of processing time, we compared our method with the method proposed in [10]. Using similar hardware, given that the average processing time of all alerts in RACC is 1.57 seconds versus 2.56 seconds of [10], it can be said that our method is about 40 percent faster. It is noted that both methods have been applied on the alerts for the first scenario of DARPA2000.

Figure 14 and Figure 15 are the results of experiments with a newer hardware. The new hardware had main memory of 8 GB. The processing speed of each CPU thread is 3.6 GHz. The obvious thing about these experiments is the reduced CPU times for the same data. This is obtained from CPU processing power. Another notable item is the impact of the number of typical hyper alerts. As mentioned, there are 30 typical alerts in RealSecure, with respective 45 possible scenarios. Considerable difference in CPU time for processing first scenario of DARPA2000 using RealSecure versus SNORT can be seen. This is affected by the fact that there are 3000 typical alerts and more than 600 possible attack scenarios in the second knowledge base of SNORT.

In addition, for the first scenario of DARPA2000, there is a significant difference between the processing time for (eth2) alerts from TH dataset, as compared to SNORT alerts. This is depicted in Figure 14 and Figure 15. Considering that the knowledge base is the same for both experiments, the only affecting difference is the order in which the alerts appear.

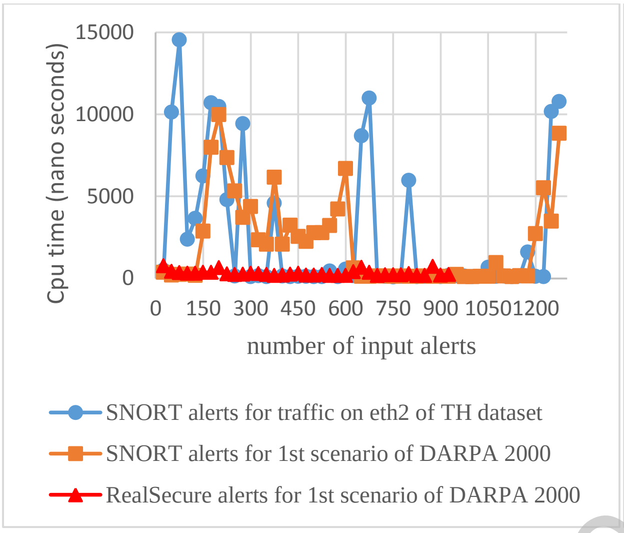

Figure 14 :Evaluating the average processing time of the proposed method on three alert sets using an up to date hardware

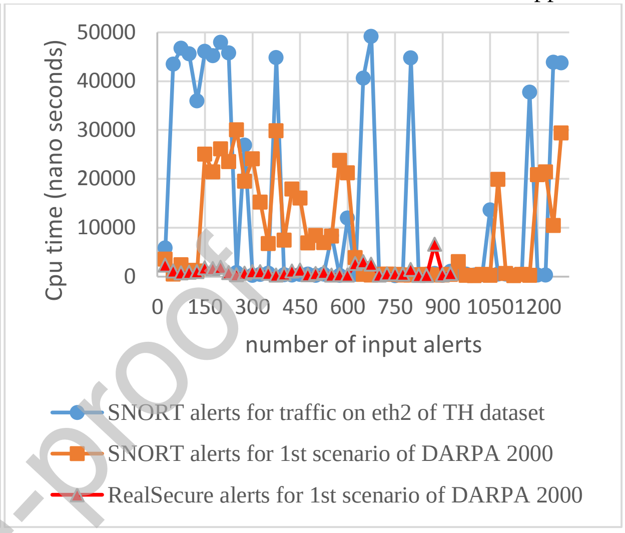

Figure 15 : Evaluating the maximum processing time of the proposed method on three alert sets using an up to date hardware

To further investigate the on-line performance of the proposed method, we also compared the required CPU time for the RACC with [12], [13] and [16]. Aforementioned methods were discussed in 3.2. These methods have reported CPU time to process the alerts in episodes of length 30, 60, 90 and 120 minutes. To compare with RACC, we followed the experiments performed by Ramki et al. [12]. Episodes of mentioned lengths were created on RealSecure alerts for the first scenario of DARPA2000. In these episodes, the processing time of alerts was calculated on the similar hardware. The result of the experiments are shown in Figure 16. As it is clear, the proposed method can process the alerts and correlate scenarios, in significantly less time than the methods presented in [12], [13] and [16]. The results in Figure 16 are calculated using similar hardware under same evaluation conditions.

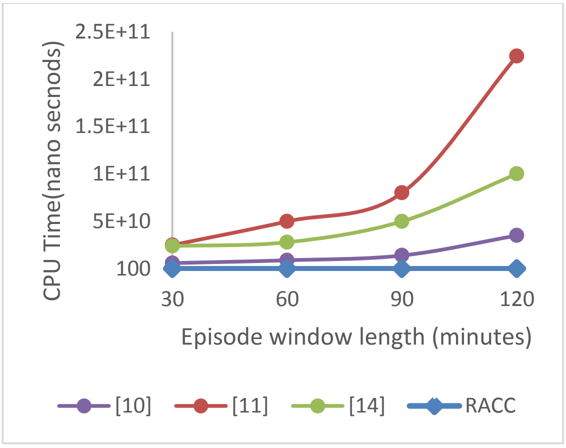

Figure 16: Comparison of CPU time with similar hardware in processing Realsecure alerts for the first scenario of DARPA2000 between RACC and [12], [13] and [16], acquired from [12]

In this section, validity, memory efficiency and execution performance of RACC was examined in terms of CPU time. These measures were examined on the alerts generated by 2 famous IDSs on 3 different set of data to prove online functionality of the method. Moreover, the speed of RACC was compared with 6 online methods. In the following, we will discuss supplementary notes about the proposed method.

4-5 Supplementary topics

Similar to other prerequisite/consequence based methods, RACC obtains its scenarios from an external knowledge base of causal relationships. Therefore, as with other prerequisite/consequence based methods, the accuracy of the proposed method is heavily dependent on the accuracy of the knowledge base in hand. Moreover, redundant communications between the alerts, might result in increasing both number of scenarios and their average length. This will cause direct impact on the runtime speed and the resources required.

Another issue for alert correlation is to predict the next steps in the scenario. Obviously, using the proposed matrix, with a quick review on a column, the next step of a scenario in progress is obtained. The prediction can be done simultaneously with the current alert processing. The output is a desired feedback to the intrusion detection system.

On the other hand, in the most prerequisite/consequent based correlation methods, such as correlation graphs, missing alerts prevent inducting the chain of causal relationships. This totally results in graph traversal abortion. This is equal to stopping of alert correlation process in certain graph paths. A quick review of the corresponding column of a scenario remedies the problem in RACC.

This study neglects alert correlation effect in false positive reduction. Each column of a code-book with just one non-zero value, generates a meta-alert. Considering the fact that the scenarios with bigger portion of carried out steps, are more likely to represent a dishonest purpose, by imposing thresholds on the number of steps, it is possible to reduce final meta-alerts. This can be done by disregarding sparse columns.

After reviewing these notes, the next section will summarize and conclude the research.

−6 Conclusion

In this article, a new alert correlation model was proposed. The model is based on some matrices which we called them code-books. Each code-book in RACC corresponds to an attack scenario. The columns of this matrix represent scenario progress on a particular host. Quick access to the matrix elements was accomplished by creating an index based on Red-Black trees. The algorithm has two phases. In the offline phase, it extracts the scenarios from a knowledge base and creates the mentioned codebooks and the corresponding Red-Black indexes. In the online phase, when an alert comes, it accesses the codebooks via a search in the Red-Black tree and does the correlation via matrix operations. We stated how this search and the matrix operations are not related to the number of security incidents and will appear as constant values in the online phase. So the correlation can be done in linear order of $O(n)$ concerning the number of alerts. Common IDSs like snort perform at the same computational order for the number of packets. So in theory, for the IDSs that perform online at the computational order of $O(n)$, RACC can be integrated with it to perform an online alert correlation. Experiments have shown considerable performance increases and less resource consumption against other real-time alert correlation methods.

As discussed in 5-4, the effectiveness of RACC is highly dependent on the identification and building of appropriate models and scenarios. In this article, scenarios were extracted from existing knowledge bases and the method tries in reducing the lag time for alert correlation and resources required to do so. Utilizing machine learning techniques to extract scenarios can extend the method and decrease the costs while improving the accuracy.

7- References

1] P. Farina, E. Cambiaso, G. Papaleo and M. Aiello, "Mobile Botnets Development: Issues and Solutions," *International Journal of Future Computer and Communication,* vol. 3, no. 6, pp. 385-390, 2014.

2] I. Vaccari, E. Cambiaso and M. Aiello, "Remotely Exploiting AT Command Attacks on ZigBee Networks," *Security and Communication Networks,* 2017.

3] R. Bace and P. Mell, "Intrusion detections systems," National Institute of Standards and Technology, 2001.

4] K. Julisch and M. Dacier, "Mining intrusion detection alarms for actionable knowledge," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining,* New York, NY, USA, 2002.

5] e. a. Saeed Salah, "A model-based survey of alert correlation techniques," *Computer Networks,* vol. 57, no. 5, pp. 1289-1317, 2013.

[16] S. Kliger, S. Yemini, D. Ohsie and S. Stolfo, "A Coding Approach to Event Correlation," in *FIP — The International Federation for Information Processing*, 1996.

[17] K. S. A. Valdes, "Probabilistic Alert Correlation," in *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, London, 2001.

[18] T. A. Alhaj, M. M. Siraj, A. Zainal, H. T. Elshoush and F. Elhaj, "Feature selection using information gain for improved structural-based alert correlation," *PloS one,* vol. 11, no. 11, p. e0166017, 2016.

[19] F. Faraji Daneshgar and M. Abbaspour, "Extracting fuzzy attack patterns using an online fuzzy adaptive alert correlation framework," *Security and Communication Networks,* vol. 9, no. 14, pp. 2245-2260, 2016.

[110] H. H. W. Hua, M. M. Siraj and M. M. Din, "Integration of PSO and K-Means Clustering Algorithm for Structural-Based Alert Correlation Model," *International Journal of Innovative Computing,* vol. 7, no. 2, 2017.

[111] S. Roschke, F. Cheng and C. Meinel, "A new alert correlation algorithm based on attack graph," in *Proceedings of the 4th international conference on Computational intelligence in security for information systems*, Berlin, Heidelberg, 2011.

[112] A. A. Ramaki, M. Amini and R. E. Atani, "RTECA: Real time episode correlation algorithm for multi-step attack scenarios detection," *computers & security,* vol. 49, pp. 206-219, 2015.

[113] M. Soleimani and A. A. Ghorbani, "Multi-layer episode filtering for the multi-step attack detection," *Computer Communications,* vol. 35, no. 11, pp. 1368-1379, 2012.

[114] Z. Zali, H. Saeedi and M. Hashemin, "Real-Time Attack Scenario Detection via Intrusion Detection Alert Correlation," in *2012 9th International ISC Conference on Information Security and Cryptology*, Tabriz, 2012.

[115] Z. Jing, L. Xiaopeng and W. Hengjun, "Real-time alert correlation approach based on attack planning graph," *Journal of Computer Applications,* vol. 36, no. 6, pp. 1538-1543, 2016.

[116] H. Farhadi, M. AmirHaeri and M. Khansari, "Alert Correlation and Prediction Using Data Mining and HMM," *The ISC International Journal of Information Security,* vol. 3, no. 2, pp. 77-101, 2011.

[117] M. Steinder and A. Sethi, "Probabilistic fault localization in communication systems using belief networks," *IEEE/ACM Transactions on Networking,* vol. 12, no. 5, pp. 809-822, 2004.

[118] B. Zhu and A. Ghorbani, "Alert correlation for extracting attack strategies," *International Journal of Network Security,* vol. 3, p. 244–258., 2006.

[119] L. Liu, K. Zheng and Y. Yang, "An Intrusion Alert Correlation Approach Based on Finite Automata," in *2010 International Conference on Communications and Intelligence Information Security (ICCIIS)*, 2010.

[120] T. . H. Cormen, C. . E. Leiserson, R. Rivest and C. Stein, Introduction to Algorithms.

[121] [Online]. Available: https://www.snort.org/.

[122] [Online]. Available: https://www.ibm.com/services/us/iss/pdf/realsecure_server_sensor_datasheet.pdf.

[123] [Online]. Available: http://discovery.csc.ncsu.edu/Projects/AlertCorrelation/index.html.

[124] [Online]. Available: https://www.ll.mit.edu/ideval/data/2000data.html.

[125] D. Xu and P. Ning, "Alert Correlation Through Triggering Events and Common Resources," in *20th Annual Computer Security Applications Conference*, 2004.

[126] A. Co, "ASPA Co dataset for IDS alert correlation," ASPA Co, 2019. [Online]. Available: https://aspaco.org/resource/research/ids/alarm-correlation/Datasets.zip.

Ehsan Mahdavi is a Ph.D. candidate with Isfahan University of technology. He received his B.S. and M.S. degrees in computer science from Shahid Beheshti University, Tehran, Iran. He then returned back to his hometown Isfahan and pursuit Ph.D. study in the field of network security with Isfahan University of Technology which is ongoing now.

Ali Fanian received the B.S., M.S. and Ph.D. degrees in computer engineering in 1999, 2001 and 2011, respectively from Isfahan University of Technology, Isfahan, Iran. He started his work in the same department as an assistant professor from that time. Different aspects of computer architecture and net- work security are his research interests; specially, ad-hoc networks, wireless network security and hardware design.

Fatima Amini received her B.S. in computer engineering from the department of electrical and computer Engineering, Islamic Azad University, Mobarakeh branch. She continued her study in intrusion detection system alert correlation and received her M.S. degrees at Department of Electrical and Computer Engineering, Islamic Azad University, Najafabad branch, Iran.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.