The ISC Int'l Journal of Information Security
ISeCure

July 2012, Volume 4, Number 2 (pp. 125–136)
http://www.isecure-journal.org

Real-Time Intrusion Detection Alert Correlation and Attack Scenario Extraction Based on the Prerequisite-Consequence Approach[☆]

Zeinab Zali[1,*], Massoud Reza Hashemi[1], and Hossein Saidi[1]

[1] *Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran*

A R T I C L E I N F O.

Article history:
Received: 30 December 2012
Revised: 20 May 2013
Accepted: 21 July 2013
Published Online: 27 August 2013

Keywords:
Attack, Intrusion, Attack Scenario, Intrusion Detection System, IDS, Alert, Alert Correlation, Graph.

A B S T R A C T

Alert correlation systems attempt to discover the relations among alerts produced by one or more intrusion detection systems to determine the attack scenarios and their main motivations. In this paper a new IDS alert correlation method is proposed that can be used to detect attack scenarios in real-time. The proposed method is based on a causal approach due to the strength of causal methods in practice. To provide a picture of the current intrusive activity on the network, we need a real-time alert correlation. Most causal methods can be deployed offline but not in real-time due to time and memory limitations. In the proposed method, the knowledge base of the attack patterns is represented in a graph model called the Causal Relations Graph. In the offline mode, we construct Queue trees related to alerts' probable correlations. In the real-time mode, for each received alert, we can find its correlations with previously received alerts by performing a search only in the corresponding tree. Therefore, the processing time of each alert decreases significantly. In addition, the proposed method is immune to deliberately slowed attacks. To verify the proposed method, it was implemented and tested using DARPA2000 dataset. Experimental results show the correctness of the proposed alert correlation and its efficiency with respect to the running time.

© 2012 ISC. All rights reserved.

1 Introduction

Computer networks are an essential part of today's information society. These networks are usually connected to the global internet network. Since security had not been considered as one of the original internet design goals, in recent decades securing networks against attacks has become very important. Nowadays various security systems and tools such as Intrusion Detection Systems (IDS) are deployed in networks to provide security. When an IDS observes any suspicious event representing an unauthorized access, or any kind of abusive or harmful activity which may result in damaging systems and computer networks, it produces alerts. But extracting useful information from these alerts is not that easy due to the following reasons:

- IDS may flag a large number of alerts every day, resulting in a flood of alerts that may overwhelm security officers.
- Among the alerts produced by IDS, false alerts are mixed with true ones.

☆ This article is an extended/revised version of an ISCISC'12 paper.
* Corresponding author.

Email addresses: z.zali@ec.iut.ac.ir (Z. Zali), hashemim@cc.iut.ac.ir (M. R. Hashemi), hsaidi@cc.iut.ac.ir (H. Saidi).

ISSN: 2008-2045 © 2012 ISC. All rights reserved.

- IDS cannot detect all the attacks and may miss some alerts.
- There are some causal relationships between continuous steps of an attack scenario, but IDS does not detect correlations among the alerts.

Therefore, it is necessary to analyze the thousands of alerts produced by one or more IDSs to extract useful information about intrusion attempts from thousands of alerts produced by one or more IDSs. We can assume each alert as a symptom of a low-level attack. Alert correlation systems receive alerts from various IDS sensors and after analyzing them produces a high level view of the attack attempts against the network. This analysis includes methods for alert aggregation, alert fusion, alert reduction, removing false alerts, missed alerts discovery and extracting alerts correlations and its objective is detecting or predicting attempted attack scenarios.

The reminder of the paper is organized as follows. After reviewing existing methods in the next section, we describe our model in Section 3. Our real-time method for attack scenario extraction, based on the new proposed model, is represented in Section 4. We analyze the proposed model and algorithm in Section 5. Experimental results are given in Section 6. Section 7 concludes the paper with our future work.

2 Related Work

Valeur *et al.* [1] have proposed a framework for an alert correlation system. It includes a comprehensive set of components (including normalization, preprocessing, alert fusion, alert verification, thread reconstruction, attack session reconstruction, focus recognition, multistep correlation, impact analysis and prioritization) for an alert analyzing system. They have considered all required preprocessing and postprocessing steps of an alert analyzing system and proposed a method for each of the alert analyzing model components. We can replace each component with our proposed method for satisfying that component's objective. We will review some of the proposed methods for different phases of alert analysis.

Some methods such as probabilistic, data mining or neural networks methods, classify or correlate the alerts based on the similarity of their chosen common features [2–6]. This class of methods is useful for the preprocessing steps of alert correlation such as alerts classification, false alerts detection or alert reduction; nevertheless, they are not sufficient (effective) for attack scenarios detection. Some research works have attempted to design and implement a language for describing known attack scenarios [7–9]. Although these languages provide standard ways to describe the attack scenarios, they limit the user to identify specific and known scenarios. Thus, if a scenario is slightly different from the defined scenarios it will not be detected in this way.

In comparison with attack description languages, causal approaches [10–13] have particularly important advantages. By using a knowledge base of low-level attacks and their prerequisites and consequences, it is possible to detect any correlated sequences of low-level attacks. Precise and complete definition of the knowledge base is the main requirement of these methods. Farhadi *et al.* [14] use a Causal Correlation Matrix (CCM) which includes the correlation probability of each pair of alerts. It uses association rules in data mining to determine and update the CCM.

Most of the casual approaches are implementable only offline. For real-time applications we need to consider a time window for received alerts such as [14]. But in this method we will not be able to detect slowed or hidden attacks. For example, the attacker can prevent correlation of the attack scenario alerts by increasing the time intervals between the successive steps of the attack scenario or alternatively by filling the time window with useless and misleading alerts. So, in real-time applications we must use some measures in the knowledge base description and also the correlation algorithm to address this problem.

Due to the strength of causal methods in practice, we construct our method based on these methods and propose a suitable model and algorithm for real-time applications. The proposed method uses the "attack graph" idea which has been used in the TVA approach [15].

The causal methods have a knowledge base which is usually considered as a table of records [11, 16]. Some new works like [17] use methods to construct the knowledge base automatically. Authors in [17] employ a Bayesian network to automatically extract information about the constraints and causal relationships among alerts. In our proposed method the knowledge base is represented as a graph called the Causal Relations Graph. This graph contains low-level attack patterns in the form of their prerequisites and consequences. In addition, it is a clear representation of causal relations among the low-level attacks. A new search is performed upon the arrival of each new alert, in the correlation phase. Thus, the search time in real-time is decreased significantly by constructing some trees in the offline mode, before the correlation system starts up.

In [11], looking for correlated alerts with a given alert is performed using a query to the previously received alert tables. This method has a high run time and is useful only for offline applications. It should be noted that some optimization is proposed in [18] for this method. In other real-time causal methods (except for the TVA method), the search leads to a nested loop for pervious received alerts. The method presented in [16] looks in previous correlated sets of alerts for an existing alert set which can be correlated to the new alert. Finding the desired set is an NP-hard problem and so heuristic algorithms should be used to solve it. There are some other new real-time methods like [17, 19]. But we show in the experimental results that our method is more time efficient. The proposed method in this paper uses the benefits of the TVA model, but does not require extracting the vulnerability graph of the topological network.

3 Causal Relation Graph Model

The proposed model has been defined based on a graph called the Causal Relation Graph (CRG). Describing the CRG requires defining *Condition* and *Alert Signature* (AS) concepts. We can represent prerequisites or consequences of a low level attack using a set of conditions. Each attack pattern in the IDS which produces an alert is considered as the representative of a low level attack event. Therefore every attack pattern is defined with its prerequisites and consequences which themselves are compositions of conditions. Our definition for some concepts are similar to the concepts in some other papers in the literature including [13, 16, 20, 21], but not exactly the same. In the following sections we provide our definition for these concepts according to our proposed model and algorithm which are capable of real-time alert correlation, implicit classification and aggregation of large volumes of alerts.

3.1 Definition 1: Condition

Every *Condition* represents satisfaction of a condition or property for an IP address or a port number, i.e., a predicate with an IP address or a port number variable. For example, $OSSolaris(VictimIP)$ says that the operating system of a host with IP address $victimIP$ is Solaris. Another example $RootAccess(victimIP)$ means that there is the capability of root access to a host with $victimIP$ address.

3.2 Definition 2: Alert Signature (AS)

Each AS is defined as AS = (Prerequisites, Consequences). Prerequisites is the set of all required conditions for launching the attack related to AS and Consequences is the set of all the attack consequence conditions. Every AS has four variables including source IP address, source port number, destination IP address, and destination port number. Every participant condition variable in the Prerequisites and Consequences sets is one of the four AS variables. For example the Alert Signature as_1 which is related to the *Sadmind* vulnerability exploit is represented as $as_1 = (\{RPCService(dstIP), OSSolaris(dstIP), SadmindService(dstIP)\}, \{RootAccess(dstIP)\})$.

3.3 Definition 3: Causal Relation Graph (CRG)

CRG is a directed graph with labeled edges and two types of vertices.

- **AS**: a vertex in CRG for every AS in a specific IDS.
- **Condition**: a vertex in CRG for every defined condition participant in at least one AS.

There are also two types of edges in CRG.

- **Prerequisite edges**: There are some edges in CRG from every Condition vertices related to prerequisites of an AS to that AS vertex.
- **Consequence edges**: There is an edge in CRG from every AS vertex to all the Condition vertices related to their consequences.

The label of each edge is the condition variable that can be the source IP address, source port number, destination IP address, or destination port number of the related AS. Please note that there is no edge between any pairs of conditions or pairs of ASs. We should define an AS for every attack pattern that exists in the IDS. A complete knowledge base on low level attacks is required for producing Alert Signature sets. For constructing the CRG, a vertex is considered for every AS related to an attack pattern in the knowledge base. There is also a Condition vertex for every Condition in AS definitions in CRG. As vertices are added to the graph, related edges link the graph vertices according to the definition. Figure 1 represents a part of CRG which includes vertices related to as_1.

3.4 Definition 4: Causal Relation Queue Tree (CRQT)

CRQT is a data structure based on CRG. For constructing CRQT, we consider a queue with the length of q for every CRG vertex. The elements which are placed in queues of AS and Condition vertices are respectively alerts and condition variables (IP address or port number). Then for every AS vertex as_x, two trees with root as_x are extracted from CRG.

- **Forward Queue Tree (FQT)**: If a breadth first search is done starting from vertex as_x and in the direction of edges' direction, a tree called FQT is obtained. Please note that the tree vertices are

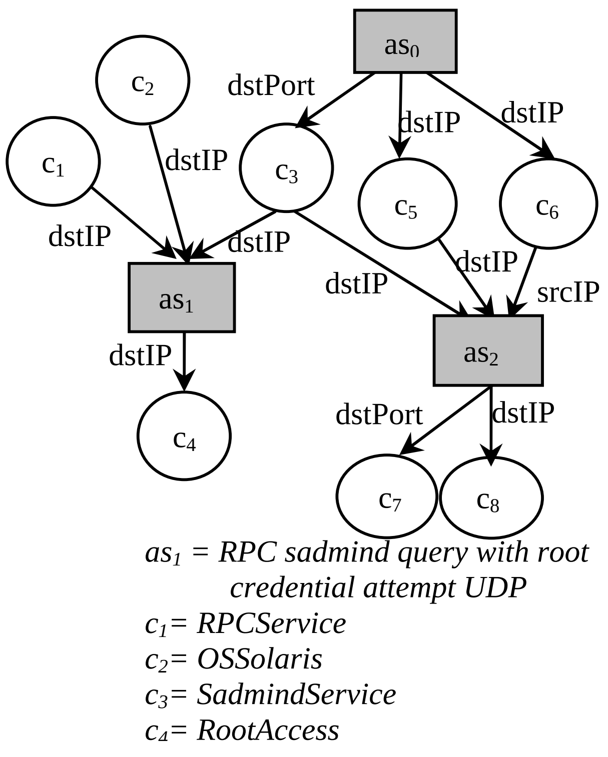


Figure 1. A part of sample CRG

a subset of CRG vertices. It is sufficient to have some pointers linking the desired vertices of the CRG to construct every FQT.

- **Backward Queue Tree (BQT)**: This tree is constructed in the same way as FQT is constructed, with the difference that the search is done in the reverse direction of CRG edges.

The ideas of CRG and FQT (BQT) are not the same as the ones in [15, 21]. These are the main concepts of our distinct model. We use them efficiently in our real-time correlation.

4 The Proposed Algorithm Using CRG

The proposed model for representing our knowledge base about low level attacks related to distinct IDS was described in the previous section. In this section, we focus on the proposed algorithm for real time alert correlation and attack scenario extraction using our model.

After constructing Queue Trees, the alert correlation system is ready to receive alerts and perform alert correlation. For this purpose, any arriving alert in real time, enters the queue of corresponding AS vertex. Every alert in IDS has a unique ID. In addition, there is exactly one AS in CRG for every type of alert in IDS. Therefore we can construct an index of alert IDs which maps each type of alert to its related vertex in CRG. In this way, upon the arrival of each alert, the corresponding vertex is specified without consuming time. Over time, all the vertices' queues and implicitly queue trees are gradually filled. Simultaneously with the arrival of alerts, alert analysis and correlation is performed.

4.1 Attack Scenario Extraction Algorithm

Step1- With the arrival of each alert a_i, the CRG vertex related to AS as_i is determined. If in the queue of the determined AS vertex, alert a_j exists which is replaceable with a_i, then alert a_j will be dequeued and a_i will replace it. Otherwise, a_i is added to the queue. We can consider various criteria to examine the possibility of replacing two alerts. Propositional formulas (1) and (2) are two such criteria where their validity satisfies a_j and is replaceable with a_i.

$$(a_i.dstIP = a_j.dstIP)\&(a_i.srcIP = a_j.srcIP) \quad (1)$$

$$(a_i.dstIP = a_j.dstIP)\&(|a_i.t - a_j.t| < \delta) \quad (2)$$

In these formulas, $dstIP$ and $srcIP$ are destination IP address and source IP address of the alert, respectively. By using (1), repeated alerts will be excluded. In (2), $a_i.t$ and $a_j.t$ are respectively, a_i and a_j generation time (timestamp) and δ is a constant timing threshold. Formula (2) replaces an alert with the previous one if both are related to the same attack destined to same destination. The equality of source addresses requirement is not mentioned in (2). This criterion is considered for aggregating alerts related to cooperation between two attackers. Consider a_i is the q_i's element of as_i vertex queue.

Step 2- After the arrival of alert a_i to as_i queue, all the consequent conditions queues will be updated as explained in the following. As was described in Definition 1 (Section 3.1), every condition has one of the IP address or port number variables. If the label of the connected edge to the condition vertex is source or destination IP address, the value of the element entering into the condition queue should be respectively, source or destination IP address of AS a_i. Similarly, the values should be port numbers if the edges' labels are port numbers. If there are not any other elements in the queue with the stated value, the element will be added to the queue.

Step 3- To extract correlated alerts with alert a_i(i.e., the scenario which has led to a_i), a breadth first search should be done. This search is started from the root of the BQT with the root as_i and the q_ith element of the as_i queue. In each phase, two continuous edges will be traversed. In this search, a_i will be correlated with a_r if edges (as_i, c_k) and (c_k, as_r) are traversed continuously, such that there is q_kth element in condition c_k queue, that matches with alert a_i and the matched element will be matched with element a_r in as_r queue. A condition element is matched with alert a_i in one of the following situations:

- If the variable of condition element is IP address (port number) and the label of edge connecting Condition vertex to AS vertex is destination IP address (destination port number), then the value of condition element will be equal to a_i destination IP address (destination port number).
- If the variable of condition element is an IP address (port number) and the label of the edge connecting Condition vertex to AS vertex is a source IP address (destination port number), then the value of the condition element will be equal to the a_i source IP address (destination port number).

As alerts arrive and correlations are extracted, results can be entered in a graph like the graph proposed in [21]. The graph in [21] has two types of edges, but we construct a graph which has only correlation edges. We call it the attack scenario graph.

The pseudo code of correlation algorithm is shown in Figure 2. Figure 3 shows attack scenario graph construction pseudo code.

When a new alert a_i is correlated to alert a_j (which was received before), if a_j has already been added to the result graph, it is sufficient to add vertex a_i and an edge between a_j and a_i to it. If a_j has not been added to the result graph, vertices a_i and a_j and the edge between them should be added to the result graph. The result graph at any moment can include more than one connected graph, each one is an extracted attack scenario detected via our real time system.

```
Procedure CRG_Alert_Correlation
Input: Correlation Relations Graph $CRG(AS \cup C, E)$,
$|AS|=n$, $|C|=m$
    Array $CQ$ of $m$ queue of condition variables
    Array $ASQ$ of $n$ queue of alerts
    Array $BQT$ of $n$ Backward Queue Tree
     ( $BQT[i](AS_i \cup C_i, E_i)$ is traced tree of backward BFS
search
     from $i$'th Alert Signature vertex of CRG, $\forall i \quad n \le i \le 1$ )
   $a_{new}$ s.t $ASig(a_{new})=as_i$, $1 \le i \le n$
  The initial Result Graph $RG(V,E)$
Output: The updated ResultGraph $RG(V,E)$
Begin_Method
    1. if $ASQ[i]$ Contains an old $a_{old}$ s.t $a_{new}$ is a replaced of $a_{old}$
           replace $a_{old}$ with $a_{new}$
           $a_{new}.event\_id \leftarrow a_{old}.event\_id$
       else
           Enqueue $a_{new}$ into $ASQ[i]$
    2. for each $c_j \in C$ s.t $(as_i,c_j) \in E$
          cv $\leftarrow$ an condition variable s.t is matched with $a_{new}$
          if $cv$ does not exist in $CQ[j]$
               Enqueue $cv$ into $CQ[j]$
    3. start from root of $BQT[i]$
             for each $k$ & $l$
                 if $ASQ[k]$ contains $a_k$ & $CQ[l]$ contains $cv_l$ s.t
                 $(a_{new},cv_l) \in E_i$ & $(cv_l,a_k) \in E_i$
                 & $match(a_{new},cv_l)$ & $match(cv_l,a_k)$
                       RG $\leftarrow$ Insert_into_RG($RG,a_k,a_{new}$)
     return $RG$
End Method
```

Figure 2. Pseudo code of correlation algorithm

4.2 Missed Alert Detection and Attack Scenario Prediction Using CRG

During alert correlation, false positive alerts will be removed in the scenario extraction function automatically. In this section we talk about detecting false negative alerts, i.e., missed alerts.

With the arrival of each new alert the consequence conditions queues of an AS vertex are updated. If the attack related to that alert was launched with the same prerequisites as defined in the Alert Signature, all its prerequisites would be satisfied. So when an alert is received, there will be a matched element with it in all the prerequisite Condition queues of that alert. Therefore, if there is no such an element in one of the prerequisite conditions queues it can be assumed that an alert has been missed. This is the base of the missed alert detection solution. If we assume that all the definitions of Alert Signatures are accurate and correct, this solution will be useful. If the attacker achieves the prerequisite Condition in a different way from the Alert Signature definition, our assumption for missed alert will not be correct.

To predict attacks which their alerts have not yet been received, we can use Forward Queue Trees. A search should be done in FQT starting from the AS vertex of the new alert. In this way, the edges leading to conditions that are not satisfied, will be traversed. In this search we can traverse only the edges with the highest probability values, i.e., the edges which are connected to AS vertices in which related low level attacks are correlated with a high probability. We can assume such probabilities in CRG. Papers [22, 23] have proposed probability methods for attack incident prediction. We can use their proposed methods in our model.

5 Analysis of the Proposed Model and Algorithm

In addition to proper alert correlation, the most important goal that the CRG method follows is real-time attack scenario detection simultaneously to IDS alert

```
Procedure Insert_into_RG
Input: alert, correlated_alert
  Initial Attack Scenario Graph ASG(V,E),
  V is an array of alerts
Output: updated Result Graph ASG(V,E)
Begin_Method
   1. alert_vertex $\leftarrow$ NULL
     correlated_alert_vertex $\leftarrow$ NULL
     for each $v \in V$
       if v.event_id=alert.event_id
          alert_vertex $\leftarrow$ v
       else if v.event_id=correlated_alert.event_id
          correlated_alert_vertex $\leftarrow$ v
   2. if alert_vertex!=NULL & correlated_alert_vertex!=NULL
      if (alert_vertex, correlated_alert_vertex) $\notin$ E
         insert (alert_vertex, correlated_alert_vertex) into E
      else if (alert_vertex=NULL &
           correlated_alert_vertex=NULL)
            initialize alert_vertex for alert and insert it into V
            initialize correlated_alert_vertex for correlated_alert
           and insert it into V
            insert (alert_vertex, correlated_alert_vertex) into E
      else if alert_vertex=NULL
            initialize alert_vertex for alert and insert it into V
            insert (alert_vertex, correlated_alert_vertex) into E
      else if correlated_alert_vertex=NULL
            initialize correlated_alert_vertex for correlated_alert
            and insert it into V
            insert (alert_vertex, correlated_alert_vertex) into E
     return ASG
End_Method
```

Figure 3. Pseudo code of attack scenario graph construction

generation. Therefore the algorithm should be efficient in time and memory. Time is the key parameter in real-time applications. If a system has a precise result but high response time, it is useless for real-time applications. In the following section we explain how the proposed method is efficient in time and memory.

With the arrival of each new alert, the time for searching the correlated alerts is at most equal to the time for searching all the edges of related BQT and their vertices' queues. This time is at most $2q \times (m+n)$, where m is the number of all the CRG Condition vertices, n is all the AS vertices and q is the maximum length of each queue. On the other hand, the required time for adding the correlation result to the attack scenario graph equals to the time it takes to search the existing alerts in the graph.

By creating an index on the alerts ID numbers in result graph, it is not required to search in attack scenario graph.

The proposed method uses BQTs for alert correlation. The number of these trees equals the number of AS, i.e., n. So, it is necessary to have n trees in the memory so that each one has at most $n + m$ pointers. On the other hand there should also be $n + m$ queues in the memory with the maximum length of q. Therefore, the total required memory size for the system is $n \times (n+m) + q \times (n+m)$. So, the memory complexity of the algorithm is $O(n^2)$. Please note that the result graph occupies a fixed amount of the system memory.

It should be mentioned that the vertex queue can be considered as time window in some methods like [1]. As was previously stated, considering time window in some real-time applications is one of their weaknesses. However, considering a separate queue for each of the vertices is as a separate window for each type of alert, while in other methods usually one window is considered for all the alerts. Additionally, in CRG only one alert is saved in the queue among all the replaceable alerts. This results in alert aggregation and prevents rapid filling of the queues. Therefore, the proposed method reduces the limitations of a time window. It should be noted that over time, most of the existing alerts in queues are not useful anymore and they can be removed. However, there is the possibility of keeping the alerts in the queues for even more than one day. Thus there is not any concern for slowed attacks. Choosing a criterion for removing earlier alerts depends on the average number of received alerts, the accuracy of IDS alert generation and the network conditions. The network condition is important in terms of traffic, network sensitivity level and the amount of threatening risks of the network. The removal criterion can be decided based on the situation.

CRG is like the attack graph in [21], but the nature of the vertices is totally different in CRG. The graph data structure is also different in CRG. The knowledge which is represented via the CRG model includes causal relations of low level attacks which are also used in the offline method [11]. Also, this knowledge is the same as the one used in [10, 16]. The graph in [15, 21] is also a model of network topological structure and its vulnerabilities. The CRG model and algorithm which proposes to use a casual relations knowledge base, have some advantages compared to the casual relation based methods described in [10, 11, 16].The method in [11] can only be used for offline applications. Even in the offline mode, [11] is not time efficient and needs optimization [18]. The correlation problem in [16] has been converted to a set coverage problem which is NP-hard. Even though it is introduced as a real-time method, the time complexity of the solution is not suitable for real time applications.

The method proposed in [15] has an advantage compared to CRG which is related to the nature of the queue graph proposed in [15]. In a queue graph only the alerts related to vulnerabilities of a given network are analyzed. In other words, an implicit alert validation takes place during the correlation process. So the search is performed on only valid alerts and useless alerts related to irrelevant or unsuccessful attacks are excluded from the search procedure. However, it should be noted that constructing such an attack graph is another hard problem in network security. Also, the size of this graph is larger than CRG, because there is a vertex for each vulnerability on each host in this graph.

Another advantage of CRG compared to the method in [15] is related to the queue length. In [15] the queue length of each alert vertex is one and so there is no choice but to replace a new alert with the earlier one. Therefore, two similar low level attacks which are launched from different sources are assumed to be two cooperating attacks, although each one may be related to a different attack scenario. According to (1), in the CRG method, two alerts of the same attack type destined to the same destination are replaced if their source addresses are also the same. Two attacks destined to the same destination are considered in cooperation when using criteria (2), if they happen at a short interval from each other.

Keeping the alerts in the queues for a long time may result in problems such as correlation of two irrelevant scenarios. So, considering a criterion for alerts expiration time is necessary. Reference [15] uses some temporal constraints to periodically dequeue some alerts. The method of [15] has a linear time complexity, but CRG's complexity is $O(nq)$. This is because the queue length can be more than one in CRG. Despite the presence of the parameter q in the time order of the algorithm, implementation results show that the processing time for a real time system is quite acceptable.

As mentioned earlier, the result graph in our method is constructed so that it does not have any redundancy and so compression is not required. For all the attacks which are replaced with each other, there is only one vertex in the result graph. To be informed of the number of repeated attempts, we can consider a distinct field for each result graph vertex. The value of this field should be equal to the number of alerts replaced with each other.

6 Experimental Results

We implemented the proposed method in C++ under Linux and tested it with the DARPA 2000 intrusion detection evaluation dataset [MIT Lincoln Lab 2000]. DARPA 2000 dataset is often used to evaluate correlation systems. Sufficient knowledge and skill is required for writing a knowledge base to construct the CRG. For using any IDS as alert generator for our system, we require a knowledge base that includes all the Alert Signatures of that IDS. The definition of each Alert Signature should include the prerequisites and consequences conditions corresponding to Definition 2. We used the knowledge base that Ning et al. used for TIAA [24]. Two knowledge bases have been provided for TIAA. The first one is for RealSecure IDS. This knowledge base has only 28 Alert Signatures [12]. The second one is for Snort which is a relatively complete knowledge base with about 3000 Alert Signatures.

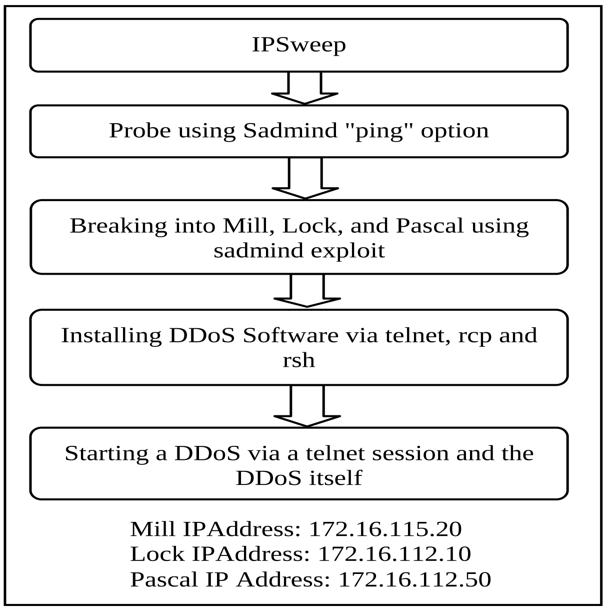

IPSweep
Probe using Sadmind "ping" option
Breaking into Mill, Lock, and Pascal using sadmind exploit
Installing DDoS Software via telnet, rcp and rsh
Starting a DDoS via a telnet session and the DDoS itself
Mill IPAddress: 172.16.115.20
Lock IPAddress: 172.16.112.10
Pascal IP Address: 172.16.112.50

Figure 4. DARPA 2000's first attack scenario

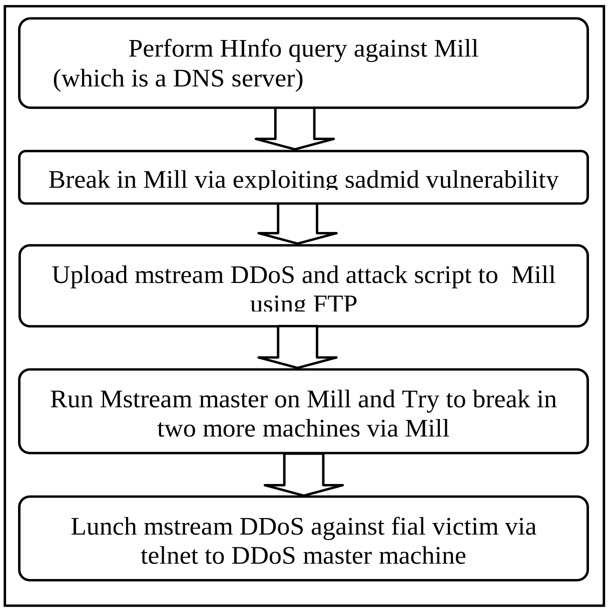
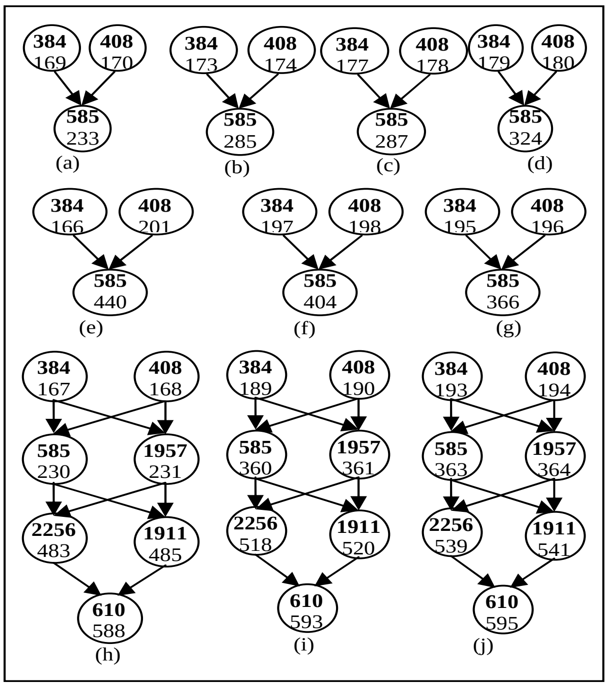

Perform HInfo query against Mill (which is a DNS server)
Break in Mill via exploiting sadmid vulnerability
Upload mstream DDoS and attack script to Mill using FTP
Run Mstream master on Mill and Try to break in two more machines via Mill
Lunch mstream DDoS against fial victim via telnet to DDoS master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

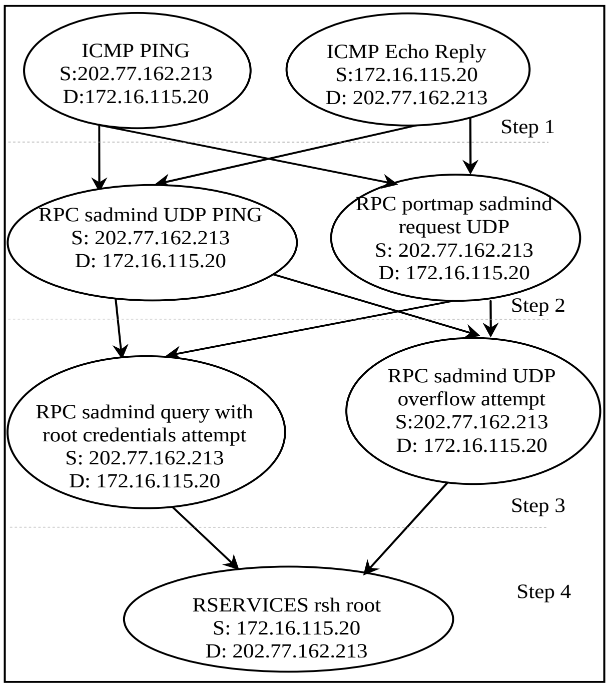
Figure 5. DARPA 2000's second attack scenario

DARPA 2000 includes two DDOS attack scenarios. Figure 4 represents the first scenario's steps and Figure 5 represents the second scenario. There are five steps



(a) 384 169, 408 170 → 585 233
(b) 384 173, 408 174 → 585 285
(c) 384 177, 408 178 → 585 287
(d) 384 179, 408 180 → 585 324
(e) 384 166, 408 201 → 585 440
(f) 384 197, 408 198 → 585 404
(g) 384 195, 408 196 → 585 366
(h) 384 167, 408 168 → 585 230, 1957 231 → 2256 483, 1911 485 → 610 588
(i) 384 189, 408 190 → 585 360, 1957 361 → 2256 518, 1911 520 → 610 593
(j) 384 193, 408 194 → 585 363, 1957 364 → 2256 539, 1911 541 → 610 595

Figure 6. The result graph of snort alerts correlation for LLDOS 1.0 Inside traffic (In each connected subgraph, every ellipse represents an alert. The upper number in the ellipse is the ID number of snort related AS, the lower number is the incidence number of alert)



ICMP PING S:202.77.162.213 D:172.16.115.20
ICMP Echo Reply S:172.16.115.20 D: 202.77.162.213
Step 1
RPC sadmind UDP PING S: 202.77.162.213 D: 172.16.115.20
RPC portmap sadmind request UDP S: 202.77.162.213 D: 172.16.115.20
Step 2
RPC sadmind query with root credentials attempt S: 202.77.162.213 D: 172.16.115.20
RPC sadmind UDP overflow attempt S:202.77.162.213 D: 172.16.115.20
Step 3
Step 4
RSERVICES rsh root S: 172.16.115.20 D: 202.77.162.213

Figure 7. Scenario attack (h) of Figure 7 in details. Source and destination addresses can be seen in this figure

in each scenario. MIT Lincoln Lab has sniffed the traffic results from launching each of the scenarios using two sensors, one sensor on inside network and another one on the DMZ network. Our results for both of the networks were successful and satisfactory. As in previous literature, we also state the inside sensor results here.

6.1 Attack Scenario Extraction Results

Figure 6 shows the result graph of snort alerts correlation for the first scenario. We expect the result graph to include all the low level attacks related to various steps of the scenario. Every connected sub-graph represents an extracted scenario. In graphs (a) to (e) there are only two steps of LLDOS 1.0. These imperfect scenarios are those unsuccessful attacks which have terminated in intermediate steps. As you can see in Figure 4, at the third step three vulnerable hosts are detected and exploited. Every sub-graph (h) to (j) in Figure 6 represents an attack scenario destined to one of these victim hosts. You can see scenario (h) in details in Figure 7. Only the last step has not been extracted. The reason is that the related alert was missed by snort.

Figure 8 represents the result graph of RealSecure alerts correlation for the first scenario. As you can see, almost all the key steps of the scenario have been extracted. The extracted scenario is more complete than the scenario extracted from snort alerts because of the more accurate RealSecure knowledge base. On the other hand, only three successful scenarios have been extracted for RealSecure and the failed scenarios

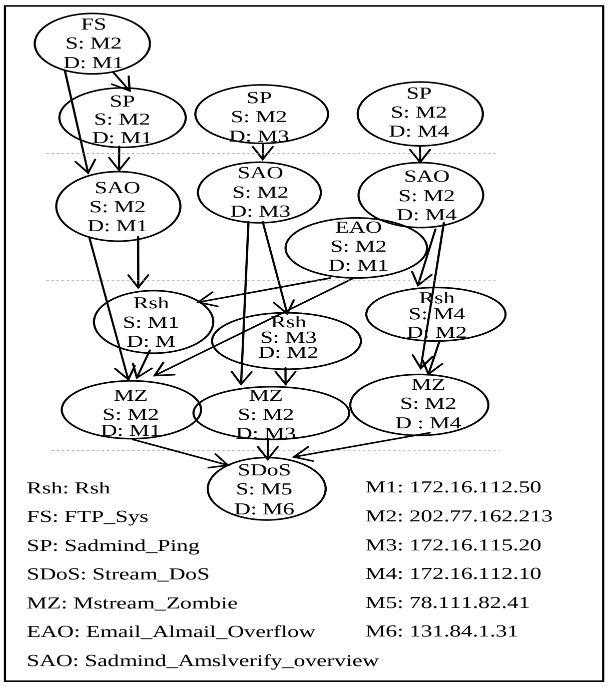


Figure 8. The result graph of RealSecure alerts correlation for LLDOS 1.0 Inside traffic in CRG method

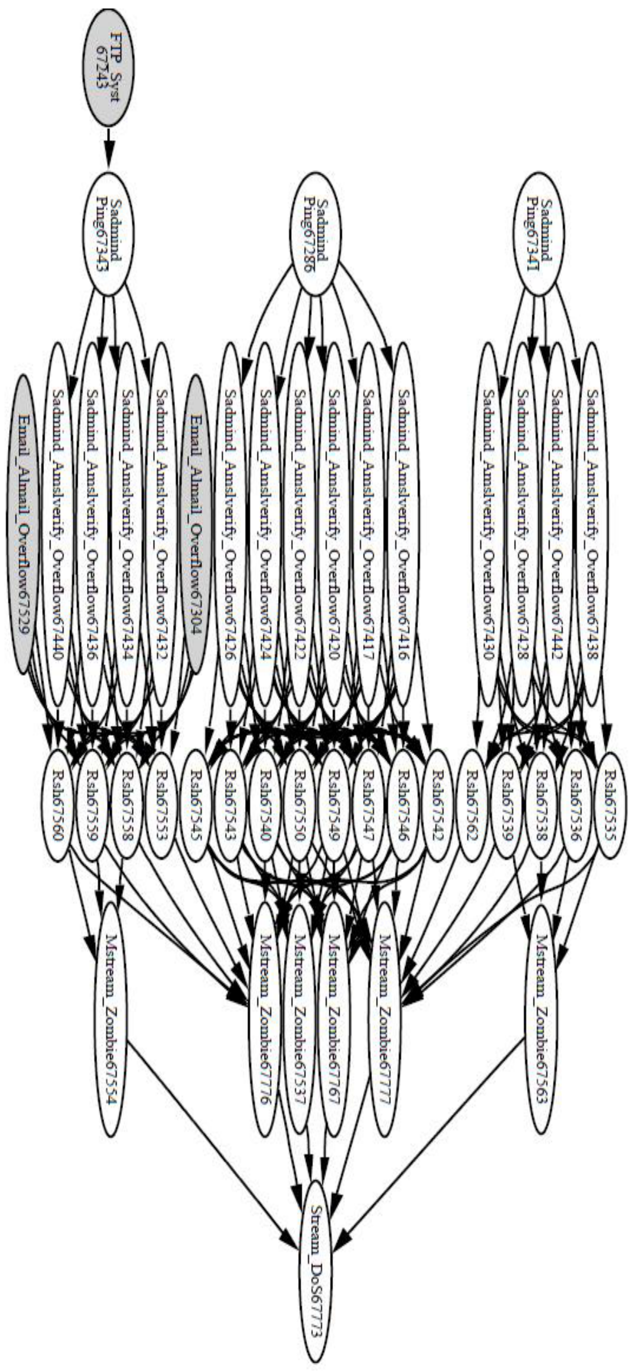

FTP_Syst 67243
Sadmind Ping67343
Sadmind Ping67286
Sadmind Ping67341
Email_Almail_Overflow67529
Sadmind_Amslverify_Overflow67440
Sadmind_Amslverify_Overflow67436
Sadmind_Amslverify_Overflow67434
Sadmind_Amslverify_Overflow67432
Email_Almail_Overflow67304
Sadmind_Amslverify_Overflow67426
Sadmind_Amslverify_Overflow67424
Sadmind_Amslverify_Overflow67422
Sadmind_Amslverify_Overflow67420
Sadmind_Amslverify_Overflow67417
Sadmind_Amslverify_Overflow67416
Sadmind_Amslverify_Overflow67430
Sadmind_Amslverify_Overflow67428
Sadmind_Amslverify_Overflow67442
Sadmind_Amslverify_Overflow67438
Rsh67560
Rsh67559
Rsh67558
Rsh67553
Rsh67545
Rsh67543
Rsh67540
Rsh67550
Rsh67549
Rsh67547
Rsh67546
Rsh67542
Rsh67562
Rsh67539
Rsh67538
Rsh67536
Rsh67535
Mstream_Zombie67554
Mstream_Zombie67776
Mstream_Zombie67537
Mstream_Zombie67767
Mstream_Zombie67777
Mstream_Zombie67563
Stream_DoS67773

Figure 9. The result graph of RealSecure alerts for LLDOS1.0 Inside traffic in Ning method [11]

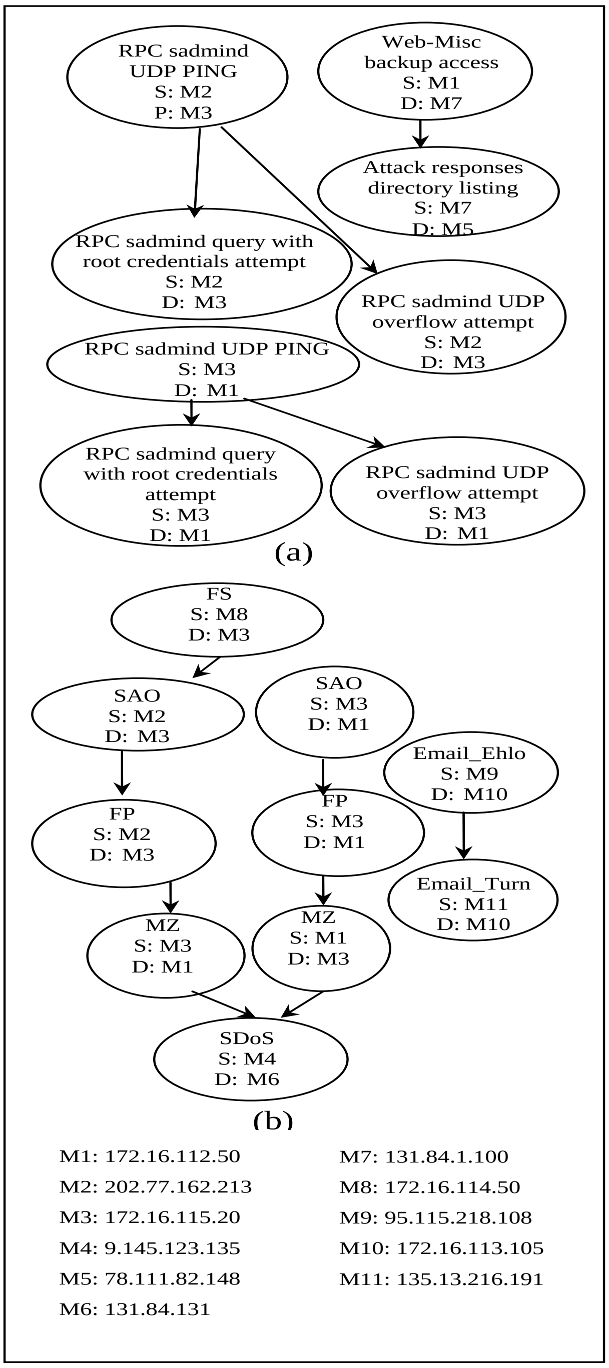


Figure 10. Results of alert correlation for the second DARPA 2000 scenario

have not been extracted. The reason is the differences among the two IDSs. RealSecure itself is more accurate than Snort. It removes some unsuccessful attacks. The first step is not represented in Figure 8, because RealSecure does not produce ICMP alerts.

Figure 9 represents the scenario extraction result of the Ning [11] method. You can compare our result graph in Figure 8 with Figure 9. As you can see our result graph is more compact and explicit. It is because our method can aggregate identical alerts to the same destination.Then again this attack graph has been obtained in the offline mode.

The result of alert correlation for the second DARPA 2000 scenario is shown in Figure 10. It is not very accurate or complete. The reason is that the attacker tries to hide the steps in the second scenario and launches the attack scenario with more skill than the

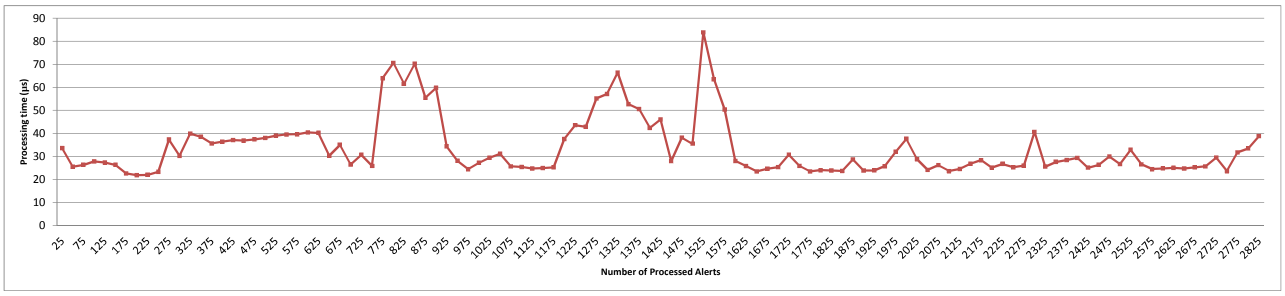


Figure 11. Average processing time per 25 consecutive alerts

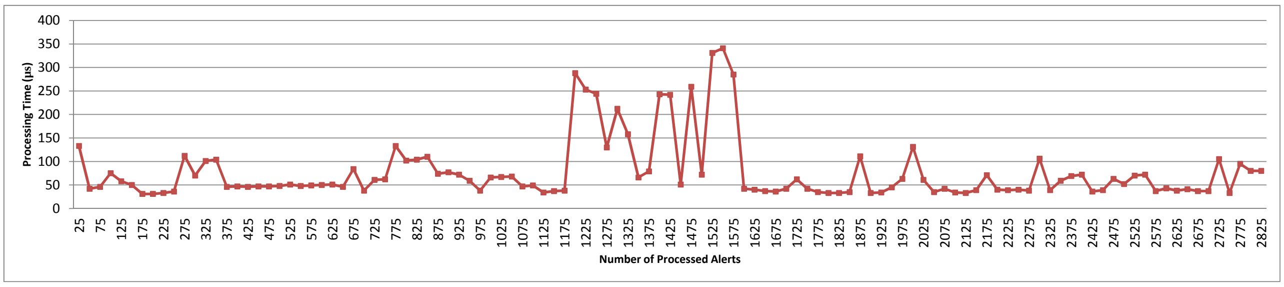


Figure 12. Maximum processing time per 25 consecutive alerts

first scenario.

6.2 Performance Evaluation

Since the main goal of the proposed method is its capability for real time operation, the time evaluation of the system is very important. We have tested the system on a Pentium III dual core 2.2GHz server with 1GB RAM running Suse 11. The CRG used in our evaluations has 903 condition vertices and 2895 AS vertices for snort and 31 condition vertices and 28 AS vertices for RealSecure.

The diagrams of Figures 11 and 12 show alert processing time. Figure 11 shows the average processing time of each 25 consecutive snort alerts. The horizontal axis represents the number of received alerts until the arrival time of a new alert. As you can see the processing time does not depend on the number of received alerts and over time after the start of the system, system performance has not decreased. Figure 12 shows the maximum processing time of every 25 consecutive alerts. As you see the maximum processing time of all the processed alerts is about 0.35ms. In Figure 11 and Figure 12, Alerts which take much more processing time are related to the AS vertices that have many connected edges to other vertices.

Whereas the maximum processing time of our method is about 0.35ms, this time is about 0.1s in [15] (Figure 13 of [15]), and about 3s in [17, 19].

Figure 13 shows the processing time of snort and RealSecure alerts in one diagram. This diagram has been drawn according to the number of existing alerts in queues at the arrival time of a new alert.

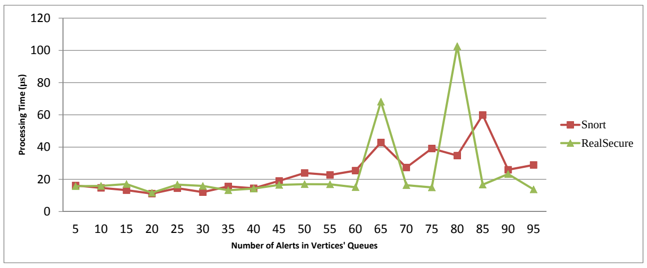


Figure 13. Processing time per number of alerts in vertices queues (for snort and RealSecure)

7 Conclusion and Future Work

In this paper we first proposed a suitable model for representing a knowledge base of causal relations between low level attacks, and then proposed an efficient algorithm for real-time attack scenario detection.

The proposed method is based on the causal approach due to the strength of causal methods in practice; moreover it benefits from the advantages of TVA model [15]. Therefore, in addition to proper alert correlation, our proposed algorithm has polynomial time and memory complexity (of degree 2) in terms of the number of CRG vertices and so it is independent of the number of received alerts. Experimental results approved the correctness of our algorithm for alert correlation and its efficiency in real-time. It should be noted that the completeness and accuracy of the knowledge base is required in our method as in any other causal method.

As a future work, we will investigate the efficiency of our proposed model for attack prediction and missed alert detection. We will also consider using artificial intelligence and data mining approaches to create and update the low level attacks knowledge base.

References

[1] F. Valeur, G. Vigna, C. Kruegel, and R. Kemmerer. A Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE Trans. on Dependable and Secure Computing*, 1(3):146–169, July 2004.

[2] F. Cuppens. Managing Alerts in a Multi-Intrusion Detection Environment. In *Proceedings of 17th Computer Security Applications Conference*, pages 22–31, 2001.

[3] S. Staniford, J.A. Hoagland, and J.M. McAlerney. Practical Automated Detection of Stealthy Portscans. *Journal of Computer Security*, 10(1-2):105–136, 2002.

[4] A. Valdes and K. Skinner. Probabilistic Alert Correlation. In *Proceedings of the 4th Int. Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, pages 54–68, 2001.

[5] B. Zhu and A. Ghorbani. Alert Correlation for Extracting Attack Strategies. *Int. Journal of Network Security*, 3(3):244–258, 2006.

[6] S.O. Al-Mamory, H. Zhang, and A.R. Abbas. IDS Alarms Reduction Using Data Mining. In *IEEE World Congress on Computational Intelligence*, pages 3564–3570, June 2008.

[7] F. Cuppens and R. Ortalo. LAMBDA: A Language to Model a Database for Detection of Attacks. In *Proceedings of the 3th Int. Workshop on the Recent Advances in Intrusion Detection (RAID 2000)*, pages 197–216, June 2008.

[8] S. Eckmann, G. Vigna, and R. Kemmerer. STATL: An Attack Language for State-based Intrusion Detection. *Journal of Computer Security*, 10(1-2):71–104, 2002.

[9] O. Dain and R. Cunningham. Building Scenarios from a Heterogeneous Alert Stream. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, pages 231–235, June 2001.

[10] S.J. Templeton and K. Levitt. A Requires/Provides Model for Computer Attacks. In *Proceedings of the 2000 Workshop on New Security Paradigms*, pages 31–38, Sep. 2000.

[11] P. Ning, Y. Cui, and D.S. Reeves. Constructing Attack Scenarios through Correlation of Intrusion Alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 245–254, Nov. 2002.

[12] D. Xu and P. Ning. Alert Correlation through Triggering Events and Common Resources. In *Proceedings of the 20th Annual Computer Security Applications Conference*, pages 360–369, Dec. 2004.

[13] F. Cuppens and A. Miege. Alert Correlation in a Cooperative Intrusion Detection Framework. In *Proceedings of IEEE Security and Privacy Symposium*, pages 202–215, 2002.

[14] H. Farhady, M. Amirhaeri, and M. Khansari. Alert Correlation and Prediction Using Data Mining and HMM. *ISeCure - The ISC International Journal of Information Security*, 3(2): 77–102, 2011.

[15] L. Wang, A. Liu, and S. Jajodia. Using Attack Graphs for Correlating, Hypothesizing, and Predicting Intrusion Alerts. *Journal of Computer Communications*, pages 2917–2933, Vol. 29, No. 15, 2006.

[16] J. Zhou, M. Heckman, B. Reynolds, A. Carlson, and M. Bishop. Modeling Network Intrusion Detection Alerts for Correlation. *ACM Trans. on Information and System Security*, 10(1):1–31, Feb. 2007.

[17] Hanli Ren, Natalia Stakhanova, and Ali A. Ghorbani. An online adaptive approach to alert correlation. In *Proceedings of the 7th Int. Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA'10, 2010.

[18] D. Xu. *Correlation Analysis of Intrusion Alerts.* PhD thesis, Department of Computer Science, University of North Carolina State, 2006.

[19] L. Zhaowen, L. Shan, and M. Yan. Real-Time Intrusion Alert Correlation System Based on Prerequisites and Consequence. In *Proceedings of the 6th Int. Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, pages 1–5, 2010.

[20] N.K. Pandey, S.K. Gupta, S. Leekha, and J. Zhou. ACML: Capability Based Attack Modeling Language. In *Proceedings of 4th Int. Conference on Assurance and Security*, pages 147–154, Sep. 2008.

[21] S. Jajodia and S. Noel. Topological Vulnerability Analysis: A Powerful New Approach for Network Attack Prevention, Detection, and Response. in Algorithms, Architectures, and Information Systems Security, B. Bhattacharya, S. Sur-Kolay, S. Nandy, and A. Bagchi (eds.), 2007.

[22] X. Qin and W. Lee. Discovering Novel Attack

Strategies from INFOSEC Alerts. In *Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS 2004)*, pages 439–456, Sep. 2004.

[23] S. Zhang, J. Li, X. Chen, and L. Fan. Building Network Attack Graph for Alert Causal Correlation. *Journal of Computers and Security*, 27 (5-6):188–196, Oct. 2008.

[24] P. Ning, Y. Cui, and D.S. Reeves. Techniques and Tools for Analyzing Intrusion Alerts. *ACM Trans. on Information and Systems Security*, 7 (2):274–318, May 2004.

Zeinab Zali was born in 1983. She received her BS and MS degrees in Computer Engineering in 2006 and 2009, respectively, both from Isfahan University of Technology (IUT), Isfahan, Iran. She is currently a Ph.D. student of computer engineering at IUT, and is working on Information Centric Networking. Her research interests include information centric networking, intrusion detection systems and alert correlation, software defined networks, and applied game theory in computer science.

Massoud Reza Hashemi received his BS and MS degrees from Isfahan University of Technology in 1986 and 1988, respectively, and his Ph.D. from the University of Toronto in 1998, all in Electrical and Computer Engineering. From 1988 to 1993 he was with Isfahan University of Technology as a faculty member. From 1998 to 1999 he was a postdoctoral fellow at the University of Toronto. From 1999 to 2002 he was a founding member and lead systems architect of AcceLight Networks, where he developed some of the key system elements of a multi-terabit multiservice core switch. Since 2003 he has been with Isfahan University of Technology. His current research interests include software defined networks, data centric networks, DSM in smart grid, and alert correlation in network intrusion detection.

Hossein Saidi was born in 1961, received BS and MS degrees in Electrical Engineering in 1986 and 1989, respectively, both from Isfahan University of Technology (IUT), Isfahan, Iran. He also received DSc in Electrical Engineering from Washington University in St. Louis, USA, in 1994. Since 1995 he has been with the Department of Electrical and Computer Engineering at IUT, where he is currently an Associate Professor of Electrical and Computer Engineering. His research interest includes high speed switches and routers, communication networks, QoS in networks, security, queueing system and information theory.
He was the co-founder and R&D director at MinMax Technology Inc. (1996-1998) and Erlang Technology Inc. (1999-2006) both in USA, and SarvNet Telecommunication Inc. (2003). During these years, he was the main architect of three generation of Switch ASIC chips: WUMCS, SeC and XeC with respectively 2.5, 10, and 80 Gb/s capacity per chip and up to 560 Gb/s total system capacity. He was also the principal architect of SeT, the network processor chip of Erlang Technology.
He holds four US and one international patents, and has published more than 100 scientific papers. He is the recipient of several awards including: 2006 ASPA award (The 1st Asian Science Park Association leaders award) and the Certificate award at the 1st National Festival of Information and Communication Technology (ICT 2011), both as the CEO of SarvNet Telecommunication Inc. He also received the 2011 Isfahan University of Technology Distinguished Researcher Award.