An Online Adaptive Approach to Alert Correlation

Hanli Ren, Natalia Stakhanova, and Ali A. Ghorbani

Information Security Center of eXcellence
University of New Brunswick
Fredericton, New Brunswick, Canada
{e8vwe,natalia,ghorbani}@unb.ca

Abstract. The current intrusion detection systems (IDSs) generate a tremendous number of intrusion alerts. In practice, managing and analyzing this large number of low-level alerts is one of the most challenging tasks for a system administrator. In this context alert correlation techniques aiming to provide a succinct and high-level view of attacks gained a lot of interest. Although, a variety of methods were proposed, the majority of them address the alert correlation in the off-line setting. In this work, we focus on the online approach to alert correlation. Specifically, we propose a fully automated adaptive approach for online correlation of intrusion alerts in two stages. In the first online stage, we employ a Bayesian network to automatically extract information about the constraints and causal relationships among alerts. Based on the extracted information, we reconstruct attack scenarios on-the-fly providing network administrator with the current network view and predicting the next potential steps of the attacker. Our approach is illustrated using both the well known DARPA 2000 data set and the live traffic data collected from a Honeynet network.

Keywords: alert correlation, Bayesian network.

1 Introduction

The rapid increase in the number, sophistication and impact of computer attacks makes the computer systems unpredictable and unreliable, emphasizing the importance of intrusion detection technology. Intrusion detection systems (IDS) are generally designed to provide a system administrator with sufficient information to handle intrusion incidents. In practice, with the ever increasing capacity of networks this often translates to a large number of low-level alerts produced by an IDS. A tremendous volume of alerts coupled with their low quality makes it challenging for a system administrator to handle intrusions in timely manner.

In this context, the alert correlation techniques aiming to consolidate relevant IDS alerts in a concise high-level format gained a special interest. Several alert correlation techniques have been proposed in the recent past including approaches based on feature similarity analysis [1], attack rules and scenarios [2,3,4,5,6,7] and analysis of alert statistics [8,9,10,11]. Generally, these

C. Kreibich and M. Jahnke (Eds.): DIMVA 2010, LNCS 6201, pp. 153–172, 2010.

techniques follow one of two directions: they either rely on expert knowledge or infer relationships among alerts using statistical or machine learning analysis. Although the expert knowledge based approaches appear to produce accurate results, they are generally limited in their ability to integrate novel alerts. Moreover, accurately defining all possible relationships among existing alerts can be prohibitively tedious and time-consuming, and thus not always feasible. The inference approaches on the other hand allow to accommodate novel alerts through automatic alert analysis. However, they might not fully discover the causal relationships between related alerts. Given the complementary nature of the expert knowledge-based and inference approaches, it is highly desirable to combine their strengths, while avoiding their weaknesses.

In this paper, we present a method for automatic correlation of intrusion detection alerts that brings together the strengths of expert knowledge-based and inference approaches. Instead of relying on user expertise, we propose to analyze the casual relationships among alerts using a Bayesian network. Through this analysis coupled with network configuration information and expert knowledge we automatically extract the constraints and alert relationships that characterize attack steps. The extracted information is essentially attributed to various attacks and thus can be employed to piece together an attack strategy in the online setting.

To facilitate real-time intrusion analysis, we further develop a technique for the adaptive online alert correlation. To allow the correlation procedure to automatically adjust to the new previously unseen (in the offline setting) behavior, the approach monitors the alerts behavior to reflect any significant changes that might potentially influence the causal relationships among alerts. This online correlation strategy not only provides a picture of the current intrusive activity on the network, but also predicts a potential next step of an attacker.

Although the online component is specifically developed for runtime correlation of alerts, both components can be applied in the offline setting.

The contributions of this paper can be summarized as follows:

- *A Bayesian correlation feature selection model* that allows to automatically retrieve causal relationships and relevant features among alerts without expert or domain knowledge. The proposed feature selection method explicitly shows the relationships among alerts and provides reasoning behind these relations.
- *An adaptive method for online attack scenario construction* that allows a user to extract attack patterns in real time. The proposed method provides a dynamic adaptation of the correlation procedure to the temporal changes in alerts' behavior. This allows to address previously unseen alerts, and consequently, discover new attack steps.
- *Implementation* of the proposed approach that allows a user to generate attack scenarios from a large amount of raw alerts on-the-fly.

The reminder of the paper is organized as follows. A brief overview of related work is given in Section 2. Section 3 provides an overview of the proposed approach for alert correlation. Section 4 describes the detailed design of the al-

ter correlation feature selection algorithm based on Bayesian causality theory. Section 5 describes the design of online alert correlation. Experimental results are given in Section 6. Section 7 concludes the paper with our future work.

2 Related Work

In recent years, a variety of alert correlation techniques have been proposed aiming to reduce the overwhelming number of alerts and to provide a global and condensed view of the network security status. Broadly speaking, these approaches can be divided into several groups: *alert aggregation techniques* [1,12,13] that cluster similar alerts; the methods focused on *detection accuracy improvement* [14,15,16] that aim to improve the accuracy of intrusion detection often through filtering of false positive and low-interest alerts; the methods for *alert prioritization* [17,18,19] that focus on adjusting priority of alerts based on their severity; and *alert causality* analysis. Since our work employs the alert causality analysis, we will primarily focus on the related work in this area.

Research on causality analysis trends can be considered within the three categories: *scenario-based correlation*, *rule-based correlation* and *statistical correlation*. In general, the studies in the first two categories rely on expert knowledge to find related alerts, while the approaches in the last category aim to infer logical relationships among alerts using statistical or machine learning analysis.

Scenario-based correlation methods [5,6,7,20] find relationships among alerts based on the known attack scenarios. The goal of alert correlation in this case is to find a sequence of alerts that match pre-specified scenarios. Attack scenarios can be specified using an attack language (e.g., SRARL [6], LAMDBA [5], ADeLe [7]) or learned using machine learning techniques [20]. One of the major downsides of these approaches is the necessity to develop all attack scenarios in advance, which is not only a time-consuming and error-prone process, but also requires a considerable expert knowledge.

Rule-based correlation approaches [2,3,4] are based on the observation that the majority of alerts are related, i.e., they either represent the early stages of an attack or intermediate steps of more advanced attack behavior. Thus, analyzing alerts based on the predefined rules containing prerequisites and consequences of attack steps is sufficient to identify related alerts. Similar to the scenario-based correlation methods, these approaches require specific attack knowledge. Although they can explicitly show the logical relationship between the alerts, they cannot handle novel attacks since their prerequisites and consequences are undefined.

As opposed to rule-based methods, statistical approaches [8,9,10] statistically analyze relationships among alerts based on their co-occurrence within certain time period, and thus, are generally independent of prior domain knowledge. As such, Qin et al. [8,21] presented a Bayesian correlation engine to discover the strong statistical dependency among alerts. Based on the assumption that alerts are causally related if there exists a strong statistical dependency among them, Qin analyzes statistical patterns among the aggregated alerts, i.e. *hyper alerts*. The degree of relevance of alerts is evaluated by calculating the conditional probability

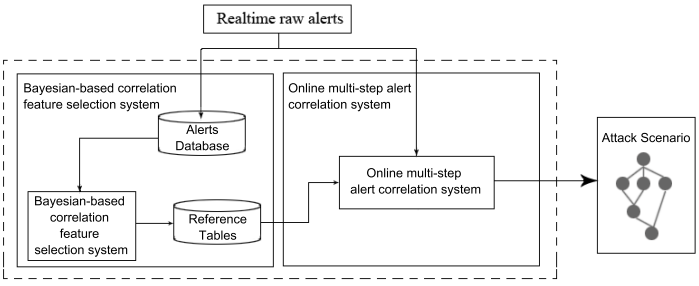


Fig. 1. Overview of the alert correlation system

among each pair of hyper alerts. The approach builds the attack scenarios by evaluating the causal relationship between each pair of hyper alerts. Since the number of all possible combinations of hyper alerts can be extremely large, a straightforward application of this approach in the online setting becomes infeasible.

Motivated by this idea, we propose to generate attack strategies on-the-fly. Specifically, rather than evaluating statistical patterns among the hyper alerts, we focus on extracting the constraints and causal relationship between each pair of alert types (even if the number of raw alerts is large, the number of alert types is usually limited to a small number). We take an advantage of this strategy in the online attack scenarios construction stage by analyzing only the most relevant features.

There have been several works [9,10] specifically focused on the methods for estimating correlation probability among alerts using Multilayer Perceptron (MLP) [22], Support Vector Machines (SVM) [23] and frequent structure mining technique [10]. Unlike scenario-based and rule-based methods, statistical approaches do not require expert knowledge and are capable of representing unknown attacks; they however cannot explicitly show the causal relationship among alerts. Requiring heavy statistical analysis, these techniques are generally time consuming and thus not applicable in the online setting.

In summary, all the above approaches focus on the offline alert correlation. In our work, we attempt to address this issue by performing alert correlation in two stages. Unlike scenario-based and rule-based methods, we use statistical analysis to automatically extract prerequisites and consequences of attack steps. Since the statistical analysis is a time-consuming process, we perform it in an offline mode. Then, based on the extracted information, the online component then connects related alerts and constructs attack scenarios on-the fly. Contrary to the statistical methods, our approach can explicitly show the causal relationships between alerts.

3 Overview

The alert correlation aims to consolidate IDS alerts based on their causal relationships. These relationships can be discovered relatively easy if attack strategy,

prerequisites and consequences are known in advance. In practice, manual generation of such attack information requires expert knowledge and experience, and thus is not only time-consuming, but also error-prone. An alternative approach to realize these relationships is through the automatic analysis of raw alerts. However, the main challenge that arises in this context is to extract sufficient number of constraints and conditions pertinent to the considered attack strategy in order to accurately characterize the instances of this attack in the future. In this work, we adopt the latter approach and attempt to extract casual alert relationships automatically. To achieve this goal we propose a two-component correlation model. The two components of our model, namely, offline Bayesian correlation feature selection component, and online multi-step alert correlation components, are shown in Figure 1.

In the offline component, we aim to extract relevant alert information that can be later employed in the online alert correlation. First, we aggregate alerts that belong to the same attack step. Based on the Bayesian causality we then analyze the relevance of alerts representing different attack steps. Finally, we extract the features that define relevancy of attack steps. As the result of the offline correlation, the system produces reference tables (namely, the correlation and relevance tables) that contain information necessary to identify causal relationships among alerts on-the-fly. The online alert correlation component processes the raw low-level alerts to extract attack scenarios based on the attack information provided by the reference tables. To dynamically adjust correlation process to the current alerts' behavior that might incorporate previously unseen alerts, the online module monitors the changes in the alerts' behavior. These temporal changes are automatically accounted for in the attack scenario construction.

Note that the offline component primary allows to speed up the online correlation process. Thus when necessary both components can be applied offline for analysis of historical data.

4 Bayesian Correlation Feature Selection

Figure 2 shows the first step of the proposed alert correlation framework, which is the offline analysis of low-level alerts. This analysis aims to produce sufficient information for the following automatic real-time attack reconstruction.

In the proposed framework, this analysis is performed in two steps: preprocessing and feature extraction. In the preprocessing step, the aim is to reduce redundancy by bringing alerts into a standard format of a hyper alert [8]. Generally, intrusion detection sensors produce a number of alerts for each suspicious event. Most of these alerts are repetitive and provide the same information about an event. This information can be concisely represented through hyper alerts. A hyper alert is essentially a set of low-level alerts aggregated based on the values of their attributes and clustered into a group.

In the next step, we infer the causal relationships between the hyper alerts through a Bayesian probability analysis. Based on the correlation probability, we extract hyper alert attributes that significantly influence the degree of relevance

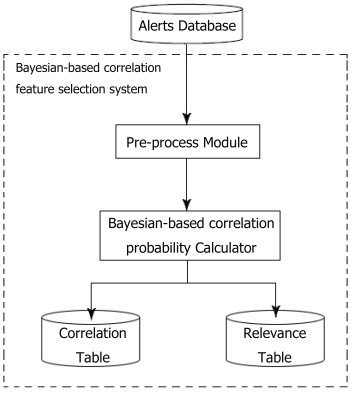


Fig. 2. Overview of the offline alert correlation component

of two hyper alerts. Analyzing relevance of the extracted features allows to reason about the relationships between hyper alerts and consequently between alert types. One of the requirements in this context is to ensure that the inferred relationships reflect specific properties of the network environment. This information may downgrade the relevance of alerts not applicable in a given environment or emphasize critical relationships. For example, by introducing the network asset group information, the relevancy of alerts whose target IP addresses belong to the same asset group is emphasized, even though their exact destination IP address are different. The network specific properties are incorporated through the generalization hierarchies built using expert knowledge.

4.1 Alert Preprocessing

The preprocessing step follows a straightforward strategy to compress information represented in low-level alerts. For the causal relationships analysis we adopt a hyper alert format discussed by [8].

A *low-level alert* a_i is an alert produced by intrusion detection sensor (e.g. IDS) with a set of alert attributes, i.e, features such as destination IP address, port number etc., denoted by $f_1 \dots f_j$. The value that a feature f_j assumes in alert a_i is denoted by $a_i[f_j]$. The range of possible values, i.e., domain of feature f_j is denoted by $dom(f_j)$.

Given a set of low-level alerts $a_1 \dots a_n$, a *hyper alert* A_i is a group of low-level alerts a_k, $1 \leq k \leq n$ with the same features' values (except timestamp), i.e, for each $a_k, a_l \in A_i$, $a_k[f_j] = a_l[f_j]$. The hyper alerts, as well as low-level alerts can be also distinguished based on the *type* of alert that denotes a certain attack class/step. To differentiate between the types of alerts, we refer to low-level alerts using small letters and to hyper alerts using capital letters. As such, low-level alerts a_i, b_i and hyper alerts A_i, B_i indicate alerts of type a and b, respectively. Then, $A = [A_1, A_2, A_3, \cdots, A_m]$ denotes a group of all hyper alerts of type a

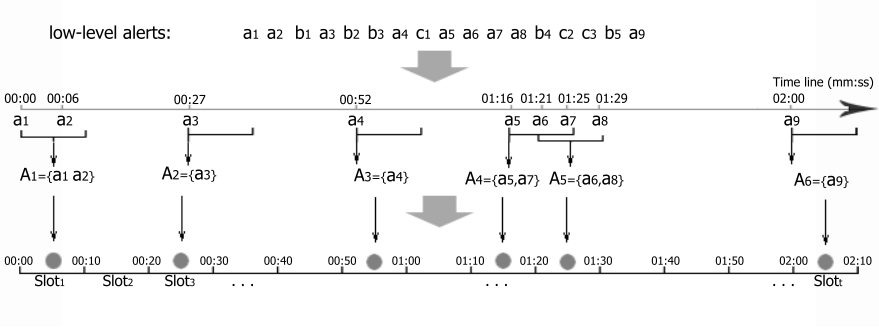


Fig. 3. The pre-processing procedure

and $dom(A, f_j) \subset dom(f_j)$ denotes the range of values that feature f_i assumes in alerts $a \in A$.

To form a set of hyper alerts, a stream of low-level alerts is initially broken into time windows using a sliding window approach. The alerts within each window is clustered based on the alert type and then merged into hyper alerts based on the alert feature values. The example of preprocessing procedure for alerts of type a is illustrated in Figure 3. For example, the alerts a_1, a_2 fall into the same time window and happened to have the same feature values, thus they are merged into a hyper alert A_1. Similarly, a_5, a_6, a_7 occurred in the same time window. The feature values of alert a_5 match alert a_7, but do not match alert a_6, thus only a_5 and a_7 are merged into a hyper alert A_4. In the next time window, we consider alerts not previously merged into hyper alerts, i.e., a_6 and a_8. These two alerts have the same feature values, so we merge them into a hyper alert A_5. The resulting stream of hyper alerts is also broken into time windows, $Slot_1, Slot_2, \cdots, Slot_t$. The result of preprocessing stage is the groups of hyper alerts of different types $A, B, \ldots, Z$.

4.2 Feature Selection

Evaluating relationships between intrusion detection alerts essentially means analyzing alert attributes, i.e. features. Since not all attributes equally contribute to the relationship between two alerts, it is desirable to identify the attributes that are necessary to be analyzed. Thus, given a set of hyper alerts, the goal is to extract alert features that are the main contributors to the relationships between intrusion detection alerts.

We perform this process in three steps as follows:

1. *Feature construction:* derive additional features based on the basic alert attributes such as IP address, port, and protocol.
2. *Correlation probability calculation:* estimate the correlation probability of alerts and determine the alert features that contribute the most to this probability.
3. *Construction of correlation and relevance tables.*

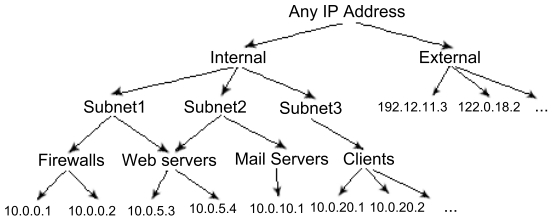


Fig. 4. An example of a generalization hierarchy of IP address

Step 1: Feature construction. The IDS alert features capture intrinsic alert properties, such as the IP address of an alert, its port, protocol information, etc. While the values of these features are the same for low-level alerts grouped in one hyper alert (except time stamp), their values vary among the hyper alerts of the same type. At the same time feature values of hyper alerts share common patterns that allow to describe the hyper alert type. For example, a set of hyper alerts representing an attack on a subnet, although has a different destination IP addresses, shares the same subnet address.

Extracting the basic alert attributes may not be sufficient to fully discover these patterns. We, thus, derive additional features from the available attributes, called *extended features*. To derive extended features we follow the idea of *generalization hierarchy* introduced by [13]. The generalization hierarchy is a directed acyclic graph that spawns elements of an alert attribute domain. For example, Figure 4 shows a sample generalization hierarchy for IP addresses. We employ the following generalization hierarchies:

- *Alert IP address hierarchy* includes source and destination IP addresses and is generalized into asset groups (e.g. firewalls, mail servers), subnets and network domain (e.g. internal, external network).
- *Alert port number hierarchy* includes both source and destination ports and is generalized according to a service assigned to a port (e.g., DNS, FTP, HTTP) and into privileged and non-privileged ports.

Generally, generating generalization hierarchy is network-specific and thus requires expert knowledge.

Step 2: Correlation probability calculation. The probability inference engine is based on Bayesian network [24], one of the most widely used models for understanding the causal relationships among a large number of variables. A Bayesian network is essentially a graphical model that represents probabilistic relationships among all variables.

A Bayesian network model consists of: (1) a network structure that describes analyzed variables via a directed acyclic graph (DAG) and (2) a set of probabilities associated with each variable and presented in conditional probability tables (CPTs). Together, these components describe causal or dependent relationships among variables and the strengths of these relationships [24]. Figure 5 shows a

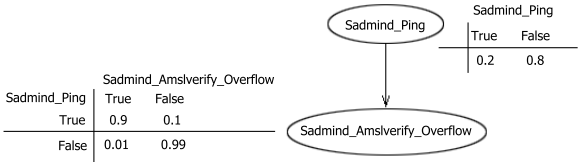


Fig. 5. An example of a causal network

snapshot of an alert Bayesian network. The occurrence of children alerts is primarily influenced by the state of their parents. In this context, the children alerts can be viewed as a direct cause of parent alerts, i.e., a consequent attack step. To evaluate the probability of child alert occurrence given its parents' state a conditional probability $P(child|parent)$ has to be computed. This step is known as probabilistic inference and can be assessed as follows:

$$P(child = c|parent = p) = \frac{P(child = c \wedge parent = p)}{P(parent = p)} \tag{1}$$

Propagating the probabilistic inference calculation from parents to children allows to infer dependencies among alerts and estimate the strengths of their relationships.

Our interest in Bayesian inference model is not limited to relationships between alert types. To be able to assess the alert relevance in the online setting, we need to reduce the amount of features analyzed. Thus, we employ Bayesian model to determine the influence of the individual features on the causal relationships among alerts.

The pseudocode for estimating Bayesian correlation probability among alert types is given in Algorithm 1. Given a pair of hyper alert groups $\langle A, B\rangle$, the procedure aims to analyze the causal relationship between the hyper alert of type a and b, and specifically, the influence of feature f_j on this relationship. The procedure returns the probability of the occurrence of type b alerts given that type a alerts happened, denoted by $P(B|A[f_j] = dom(B, f_j))$. This process requires calculation of three components:

- the prior probability of alerts of type b, $P(B)$, which essentially indicates a probability of type b alerts happening (Line 2). Note that prior probabilities of alert types can be extracted during alert preprocessing step.
- the probability of occurrence of alerts of type a with a specific value, $P(A[f_j] = dom(B, f_j))$ (Lines 4-7).
- the probability of occurrence of type b alerts with the given feature value, $P(B \wedge A[f_j] = dom(B, f_j))$ (Lines 9-14). Intuitively, it is clear that the relationship between alerts of two types have to be analyzed through some temporal constraints. We employ an *expirePeriod* forcing the algorithm to consider only alerts following type a alert within this specified time period. Theoretically, a large time window allows us to find the relationships between alerts of slowly developing attacks. However, it also increases the analysis

Algorithm 1. Causal Relationship Analysis procedure

1: **function** $P(B|A[f_j])$ relAnalysis $(< A, B >, f_j)$
2: calculate $P(B)$;
3: $AF \leftarrow \emptyset$;
4: **for** each $A_i \in A$ **do**
5: If $A_i[f_j] \in dom(B, f_j)$, add A_i into AF;
6: **end for**
7: calculate $P(A[f_j] = dom(B, f_j))$;
8: $ABF \leftarrow \emptyset$;
9: **for** each $A_i \in AF$ **do**
10: $w \leftarrow$number of time windows covered by the $expirePeriod$;
11: $TW_{A_i} \leftarrow$ time window of A_i ;
12: $TW \leftarrow \{TW_{A_i}, TW_{A_i+1}, \cdots, TW_{A_i+w}\}$;
13: Within TW, if $\exists B_{ii} \in B$ s.t. $B_{ii}[f_j] = A_i[f_j]$, then add A_i into ABF;
14: **end for**
15: calculate $P(B \wedge A[f_j] = dom(B, f_j))$;
16: $P(B|A[f_j] = dom(B, f_j)) \leftarrow \frac{P(B \wedge A[f_j]=dom(B,f_j))}{P(A[f_j]=dom(B,f_j))}$;
17: **return** $P(B)$,$P(B|A[f_j] = dom(B, f_j))$;
18: **end function**

Algorithm 2. FeatureSubsetSelection procedure

1: **function** F_{subset} featureSelection$(< A, B >, F)$
2: $F_{subset} \leftarrow \emptyset$;
3: **for** each $f_j \in F$ **do**
4: **if** $P(B|A[f_j] = dom(B, f_j)) > t$ **then**
5: add f_j into F_{subset};
6: **end if**
7: **end for**
8: $n \leftarrow$ number of elements in F_{subset};
9: $k \leftarrow 2$;
10: **while** $k \leq n$ **do**
11: $G \leftarrow$ all k size combinations from F_{subset};
12: $tempSet \leftarrow \emptyset$;
13: **for** each $g_i \in G$ **do**
14: **if** $P(B|A[G] = dom(B, G)) > t$ **then**
15: add $\forall f_j \in g_i$ into $tempSet$;
16: **end if**
17: **end for**
18: **if** $tempSet \neq \emptyset$ **then**
19: $F_{subset} \leftarrow tempSet$;
20: $n \leftarrow$ number of elements in F_{subset};
21: $k \leftarrow k + 1$;
22: **end if**
23: **end while**
24: **return** F_{subset};
25: **end function**

time. Thus, the *expirePeriod* size selection should be depend on the system performance.

The main advantage of the casual relationship procedure (Algorithm 1) is that it requires no knowledge of attack scenarios or constraints. Applied to the stream of alerts, the algorithm can distinguish influential and irrelevant features for proceeding alerts; it cannot, however, identify the degree of this influence and determine whether combinations of certain features can be associated with the more significant influence. This type of analysis is performed by Algorithm 2.

Based on the relevancy strength and a predefined threshold t, we can distinguish four kinds of features:

- If $P(B|A[f_j] = dom(B, f_j)) = P(B)$, then f_j is an *irrelevant feature*. In other words, feature f_j does not influence the occurrence probability of alerts of type b.
- If $P(B|A[f_j] = dom(B, f_j)) < P(B)$, then f_j is *a relevant feature with negative influence*. The presence of certain values of feature f_j in alerts of type a decreases the occurrence probability of type b alerts.
- If $P(B) < P(B|A[f_j] = dom(B, f_j)) < t$, then f_j is *a relevant feature with positive influence*. The presence of certain values of feature f_j in alerts of type a slightly increases the occurrence probability of type b alerts.
- If $P(B|A[f_j] = dom(B, f_j)) > t$, then f_j is *a relevant feature with critical influence*. The presence of certain values of feature f_j in alerts of type a significantly increases the occurrence probability of type b alerts.

To analyze the significance of subsets of features, only relevant features with critical influence are analyzed. Algorithm 2 follows a greedy approach by analyzing all possible combinations of features (Lines 10-23). Starting with pairs of

Table 1. An example of Correlation Table

Alert Type Pair	Correlation probability	Relevant Features/constrains
$<T_1, T_2>$	70%	f_2,f_4,f_6
$<T_1, T_3>$	65%	f_1,f_3,f_4,f_6
...	...	...
$<T_i, T_n>$	80%	$f_2 = f_4$

Table 2. An example of Relevance Table

Alert type	Occurrence Probability of T_i Alerts	Relevant Alert Types	
		Strongly relevant	Weakly relevant
T_1	5%	T_2, T_3, T_5, T_6	T_4, T_7, T_8, T_9
...	...	...	...
T_n	1%	T_7	T_1, T_2, T_8, T_9

features, the procedure randomly adds a feature to each subset whose probability exceeds the specified threshold t (Lines 14-16).

Step 3: Construction of correlation and relevance tables. Once the correlation probabilities are evaluated, we construct reference tables, specifically, *correlation* and *relevance* tables that allow to hypothesize about causal relationships among alerts. The correlation table contains for all alert type pairs: the correlation probability, the relevant features that significantly influence this probability and the constrains that characterize the relationship of this pair. We denote $T = [T_1, T_2, ...T_z]$ as a set of alert types. Table 1 gives an example of the correlation table.

As oppose to the correlation table, the relevance table contains information per alert type. An example of the relevance table given in Table 2 shows that each alert type is associated with occurrence probability and the sets of weakly or strongly relevant alert types.

5 Online Alert Correlation

One of the challenges in applying alert correlation in practice is the ability of the system to extract attack strategies on-the-fly. This is mainly due to the amount of information necessary to process and draw a meaningful conclusions about the relationships among alerts. In Section 4 we introduced the Bayesian correlation engine that performs this analysis in the offline setting and outputs probability information of alert types and the corresponding relevant features. The goal of the online component is to identify "causally" related alerts and construct attack scenarios on-the-fly based on the relationships and constraints identified in the *Correlation Table*.

The online alert correlation component is given in Figure 6 and consists of the two primary modules: an *alert correlation module* responsible for identifying alert causality, and an *attack scenario module* that produces an attack graph based on the pairs of causally-related alerts.

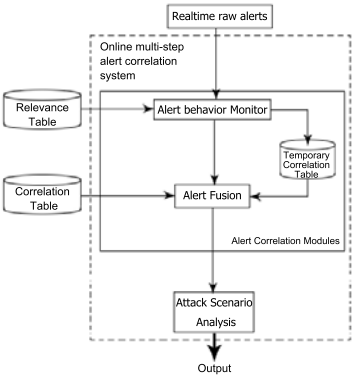


Fig. 6. Overview of online multi-step alert correlation system

Algorithm 3. Online alert correlation procedure

```
1: function AttackList onlineCorrelation (Table_corr, Table_rel, t)
2:   TempTable_corr ← OccurProbCheck(Table_rel);
3:   RecordedAlerts ← ∅;
4:   AttackList ← ∅;
5:   for each incoming alert b do
6:     Type_B ← Type of b;
7:     for each alert a ∈ RecordedAlerts do
8:       Type_A ← Type of a;
9:       TypePair ←< Type_A, Type_B > ;
10:      if ∃ TypePair in TempTable_corr then
11:        p ← getCorProb(TypePair, TempTable_corr);
12:      else
13:        p ← getCorProb(TypePair, Table_corr);
14:      end if
15:      if p > t then
16:        F ← getRelFeatures(TypePair, Table_corr);
17:        if a and b have the same value of all F then
18:          if a ∈ attack:∃attack ∈ AttackList then
19:            Add b into attack;
20:          else
21:            Create a new attack;
22:            Add a and b into attack;
23:            Add attack into AttackList;
24:          end if
25:        end if
26:      end if
27:    end for
28:    Add b into RecordedAlerts;
29:  end for
30:  return AttackList;
31: end function
```

Algorithm 4. Occurrence probability check procedure

```
1: function Table_rel OccurProbCheck (TempTable_corr)
2:   alertsList ← all alerts happened in the last hour
3:   for each alert type t do
4:     P_1 ← occurrence probability of all type t alerts in alertsList
5:     P_2 ← occurrence probability of type t alerts in Table_rel
6:     if P_1 − P_2 > 100% then
7:       weaklyRel ← getWeaklyRelType(t, Table_rel);
8:       for each alert type wt ∈ weaklyRel do
9:         recomputeCorProb(⟨t, wt⟩);
10:        update TempTable_corr;
11:      end for
12:    end if
13:    if P_2 − P_1 > 100% then
14:      stronglyRel ← getStronglyRelType(t, Table_rel);
15:      for each alert type st ∈ stronglyRel do
16:        recomputeCorProb(⟨t, st⟩);
17:        update TempTable_corr;
18:      end for
19:    end if
20:  end for
21:  return TempTable_corr;
22: end function
```

Adaptive alert correlation module. To discover "causally" related alerts, the correlation module relies on the reference tables. The reference tables maintain stable alert information that represents alerts' behavior in the past. As we see in practice, the majority of attacks have established patterns. Thus, it is reasonable to assume that if an alert was linked to certain attack steps in the past, it is likely to be related to those steps in the future. Following this intuition, the

online correlation module analyzes alert pairs with a high correlation probability based on the information provided by the correlation table.

While this strategy works well for the known patterns frequently seen in attacks, it does not allow to discover new attack steps or to incorporate in attack scenario alerts with less obvious relationships (e.g., due to their low presence in the data during the offline analysis). In order to account for these alerts in the online step, the online correlation module monitors the alerts' behavior, specifically the changes in the occurrence probability of alerts. Any sudden and significant change in the frequency of known alerts or appearance of new ones may indicate potential change in the "strength" of relationships of the corresponding alert pairs. These temporal changes to relationships, i.e, correlation probability of alerts are maintained in the *Temporal correlation table*, which can be viewed as a snapshot of alerts behavior at a given period of time. Since this table has a temporal nature, it only serves as an intermediate step between scheduled runs of the offline feature selection process.

The main advantage of such approach is that it allows to discover new alerts' relationships without any domain or expert knowledge and incorporate them into the attack scenario on-the-fly.

The *Online alert correlation* procedure function in Algorithm 3 presents the pseudocode for online correlation component. It takes as arguments a correlation table $Table_{corr}$, a relevant table $Table_{rel}$, a threshold t and an alert stream *stream*. The function returns the $AttackList$, a list of attacks where each attack is given as a set of correlated alerts.

The online correlation of alerts is performed in two steps: first, alerts' occurrence probability is analyzed to determine if any deviation in alerts' behavior has occurred. Second, the causal relationships among alerts are determined.

Step 1 - Alert behavior analysis: a temporary correlation table is maintained to monitor the significant and sudden changes in the alerts' behavior. First, we calculate the occurrence probability of each alert type in the last period of time (e.g. last one hour) (Line 4). Then we compare this probability with the one stored in the Relevance Table. If the occurrence probability of T_i alerts suddenly increases, the correlation probability of the alert type pairs grouped by T_i and its weakly relevant alert types is re-calculated. In case the produced result does not match the Correlation Table, the corresponding information is logged in the Temporary Correlation Table (Lines 6-12). On the other hand, if the occurrence probability of T_i alerts suddenly decreases, the correlation probability of the alert type pairs grouped by T_i and the corresponding strongly relevant alert types is re-calculated (Lines 13-19).

Step 2 - Alert fusion: before hyper alerts correlation probability can be computed, we pair the alerts that have shown strong connection in the past according to both the Correlation Table and the Temporary Correlation Table. When fusing two alerts, the alert type pair information is queried from the Temporary Correlation Table first, if no record found, the original Correlation Table is used. In practice, there is a number of alerts that do not have clear relationships to

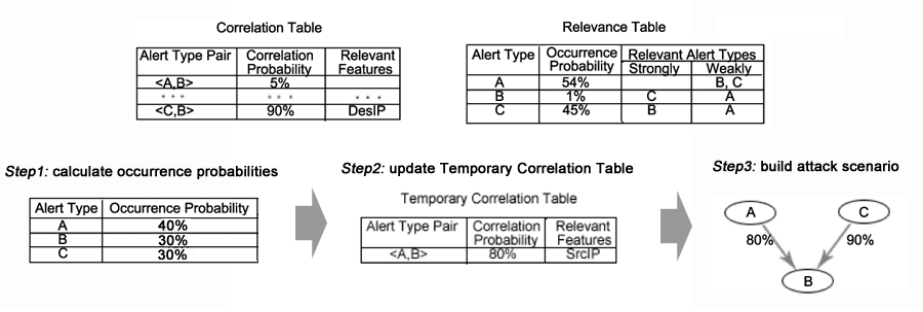


Fig. 7. An example of Correlation Process

other alerts. Although these alerts are preserved in the Correlation Table, their contribution to attack scenario is insignificant and even misleading. Thus, we apply a probability threshold that allows to navigate to the alert pairs that have the strongest relationships, i.e, are more likely to represent a meaningful step in an attack. Thus, given a probability threshold t, two alerts A_i and A_j are linked together, if $CorProb\langle A_i.A_j\rangle > t$, that is if the correlation probability of $\langle A_i.A_j\rangle$ exceeds the threshold t.

Illustrative example. Let a_1, b_1, c_1, a_2, c_2, a_3, b_2, b_3, a_4, c_3 be the latest alert steam to be analyzed by the online component in the example in Figure 7. Assume that the provided Correlation and Relevance tables have been built in the offline component. First, the occurrence probabilities of each alert type observed in the incoming stream are calculated and compared with the contents of the Relevance table ($Step1$). Let us assume that the probability of alerts of type b suddenly increased ($P(B) = 30\%$ compared to the previously recorded $P(B) = 1\%$). This increase first of all influences the correlation probability between the alerts of type b and the weakly relevant alert types. The strongly relevant types are not considered as the probability increased and thus their relevancy cannot become weak. In this case, the correlation probability of the pair $\langle A, B\rangle$ is re-calculated and the result is recorded in the *Temporary Correlation Table* as shown in $Step3$. In $Step4$, the attack scenario is built based on information contained in both the *Correlation Table* and the *Temporary Correlation Table*.

Attack scenario analysis. An attack scenario is generated based on the pairs of causally related alerts. Figure 8 shows a simple example of an attack graph. As shown in the attack graph, the $Port\_Scan, Buffer\_Overflow, FTP\_User$ alerts have already been grouped. Moreover, even though FTP_Pass alert was not reported yet, the online component is able to predict it based on the known causal relationships of $Buffer\_Overflow$ alert (i.e., FTP_Pass attack has the same source IP and destination IP address with the $Buffer\_Overflow$ attack).

Alert Type Pair	Correlation Probability	Selected Features
<Port_Scan, Buffer_Overflow>	75%	DesIP,DesPort
<Buffer_Overflow, FTP_User>	90%	SrcIP,DesIP
<Buffer_Overflow. FTP_Pass>	90%	SrcIP,DesIP

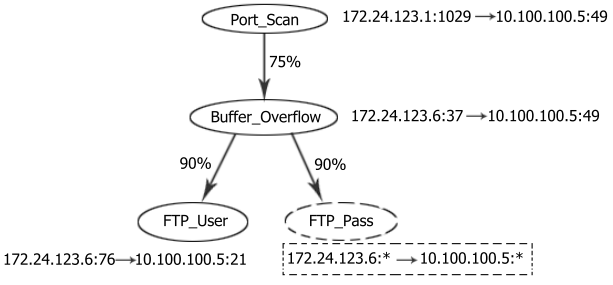


Fig. 8. Attack Graph Example

6 Experimental Results

To evaluate the effectiveness of the proposed alert correlation approach, we have implemented the algorithms described in Sections 4 and 5 and performed a series of experiments focusing on the following issues: (1) the ability of the proposed approach to select relevant features, (2) to construct accurate attack scenarios, (3) the ability to discover new attack steps, and (4) performance efficiency of the approach.

Feature selection. For our experiments we employed 2000 DARPA/Lincoln Lab offline evaluation data [25], in particular LLDOS 1.0 scenario which includes a distributed Denial-of-Service (DDoS) attack. In this scenario, the attacker first scans the network to determine which hosts are "up", then uses the "ping" option of the *sadmind* exploit program to determine which of the discovered hosts are running the *sadmind* service. Eventually, the attacker launches the *sadmind* Remote-to-Root exploit in order to compromise the vulnerable machines, and uses telnet, *rcp* and *rsh* to install a DDoS program in the compromised machines.

The low level alerts for a given data set were generated using the signature-based Snort IDS [25] by replaying "Inside-tcpdump" data. Among 15 different alert types produced by Snort, 5 alert types are directly related to the LLODS1.0 scenario. The correlation probability and the selected features of these 5 alert types are shown in Table 3.

As the results show, alert types generally exhibit specific attack patterns. For example, *Sadmind_Ping* alerts usually share the same source IP address and destination subnet, which means the attacker probes several target machines in a subnet from one source to detect hosts running *Sadmind* service. On the other hand, *Mstream_Zombie* alerts usually share the same destination port, which means the attacker issues same attacks (attacks against the same port) on various targets from different sources.

Table 3. Correlation Table

Alert Type Pair	Correlation probability	Relevant Features/constrains
<Sadmind_Ping, Sadmind_Ping>	0.96	SrcIP, DesSubnet
<Sadmind_Ping, Sadmind_Overflow>	1.0	SrcIP,DesIP,DesPort
<Sadmind_Ping, Admind>	1.0	srcIP,desIP,SrcPort,DesPort
<Sadmind_Ping, Rsh>	1.0	SrcIP,DesIP
<Sadmind_Ping, Mstream_Zombie>	1.0	DesIP of Sadmind_Ping equals scrIP of Mstream_Zombie
<Sadmind_Overflow, Sadmind_Overflow>	0.86	SrcIP,DesSubnet,DesPort
<Sadmind_Overflow, Sadmind_Ping>	0.0	
<Sadmind_Overflow, Admind>	1.0	srcIP,desIP,SrcPort,DesPort
<Sadmind_Overflow, Rsh>	0.86	SrcIP,DesIP
<Sadmind_Overflow, Mstream_Zombie>	1.0	DesIP of Sadmind_Overflow equals scrIP of Mstream_Zombie
<Admind,Admind>	0.81	SrcIP,DesSubnet,DesPort
<Admind, Sadmind_Ping>	0.13	SrcIP,DesIP
<Admind, Sadmind_Overflow>	0.87	SrcIP,DesIP,DesPort
<Admind, Rsh>	0.75	SrcIP,DesIP
<Admind, Mstream_Zombie>	1.0	DesIP of Admind equals scrIP of Mstream_Zombie
<Rsh, Rsh>	0.81	SrcSubnet,DesSubnet,DesPort
<Rsh, Sadmind_Ping>	0.0	
<Rsh, Sadmind_Overflow>	0.0	
<Rsh, Admind>	0.0	
<Rsh, Mstream_Zombie>	1.0	DesIP of Rsh equals scrIP of Mstream_Zombie
<Mstream_Zombie, Mstream_Zombie>	0.79	DesPort
<Mstream_Zombie, Sadmind_Ping>	0	
<Mstream_Zombie, Sadmind_Overflow>	0	
<Mstream_Zombie, Admind>	0	
<Mstream_Zombie, Rsh>	0	

The results in Table 3 also show the causal relationships between different alert types. Take $\langle Sadmind\_Ping, Sadmind\_Overflow \rangle$ as an example, *Sadmind_Overflow* alerts usually happen after *Sadmind_Ping*, and they share the same source IP, target IP and port, which means after using *Sadmind_Ping* to probe several targets running *Sadmind* service, the attacker issues *Sadmind_Overflow* attacks on the same targets. While for $\langle Rsh, Mstream\_Zombie \rangle$, the destination IP of Rsh alerts usually the same as the source IP address of *Mastream_Zombie* alerts, which means after the attacker compromises a target machine by *Rsh* attacks, *Mastream_Zombie* attacks will be launched against the final victim from the target machine.

Accuracy. To evaluate the accuracy of our system, we use two criteria: *True positive correlated* and *False positive correlated* rates.

True positive correlated (TPC) rate indicates the percentage of the correctly correlated alert type pairs ($True\_Correlated\_Pairs$) among all the alert type pairs that have causal relationships ($Related\_Pairs$).

$$TPC = \frac{number\ of\ True\_Correlated\_Pairs}{number\ of\ Related\_Pairs}$$

False positive correlated (FPC) rate indicates the percentage of the incorrectly correlated alert type pairs ($False\_Correlated\_Pairs$) among all the correlated alert type pairs ($Correlated\_Pairs$).

$$FPC = \frac{number\ of\ False\_Correlated\_Pairs}{number\ of\ Correlated\_Pairs}$$

Among all those 225 possible alert type pairs generated from the 15 types of alerts reported by Snort, 63 alert type pairs are labeled as causally related based on the attack scenario description given by DARPA dataset. If we set the correlation threshold to 50%, there are 70 alert type pairs correlated by our system. Among them, 9 pairs are falsely correlated, so the TPC rate of our approach on DARPA dataset is 96.8%, and the FPC rate is 12.9%. All the 9 pairs are generated among four alert types: *FTP_User, FTP_Pass, Email_Almail_Overflow and Email_Debug*. The close analysis of these pairs reveals that incorrect correlation of these alerts happens due to the high occurrence probability numbers (4% 10% compared to the probability for other types of less than 1%). This mainly due to the fact that most of these alerts share the same source or destination IP addresses as the number of different IP addresses used in DARPA experiment is very small.

Attack scenario construction. Figure 9 shows the complete attack scenario extracted from Table 3 and a group of alerts involved in this attack. A node in the attack scenario graph indicates an alert type (attack step), the edges in the graph are associated with the corresponding correlation probabilities. The rest of correlated alert type pairs, which are not shown in Table 3, also produce several connected attack graphs. But, there is no connection between these graphs and the DDoS attack scenario.

New attack steps discovery. In order to evaluate our method's ability to adapt to the temporal changes, we employed the live network traffic collected by the netForensics Honeynet team [26]. The provided logs spawn over 7 days of network traffic that triggered overall 15602 Snort alerts.

By scanning the traffic of the first day, Snort generated 1508 alerts belonging to 27 different alert types. This produced 729 alert pairs. Based on the available description and expert knowledge, out of these 729 pairs, 198 were labeled as causally related.

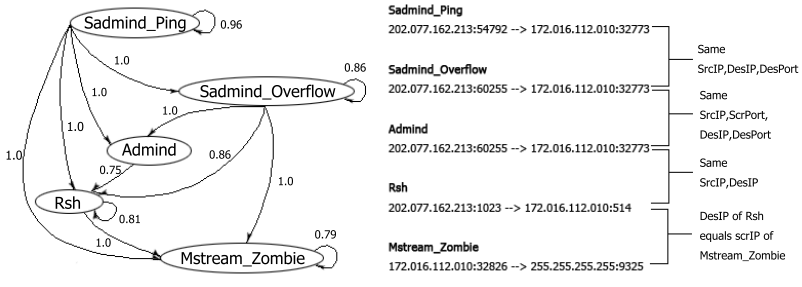
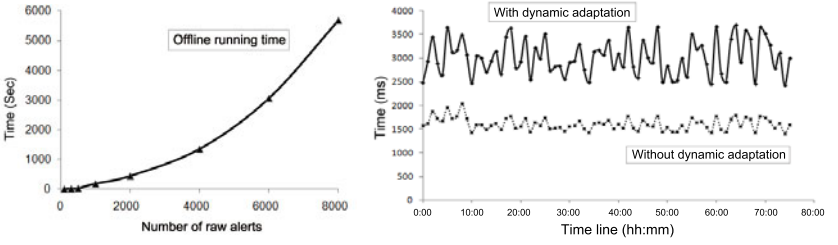


Fig. 9. Attack scenario



(a) (b)

Fig. 10. Performance of the offline and online components

Applying our Bayesian offline analysis to the alerts generated by Snort, the *CorrelationTable* shows 226 correlated alert type pairs, among them 191 pairs are correctly correlated and 36 pairs are incorrectly correlated. Thus, the TPC rate is 96.5%, and the FPC rate is 15.9%. Most of these false positive correlations are caused by the false alarms produced by Snort IDS. Specifically, Snort reported a large number of alerts of types: *ICMP Destination Unreachable, MS-SQL Worm* and *WEB-MISC WebDAV*. Due to the high frequency of these alerts and the similarity of their IP addresses, the occurrence probability of these alert types was high which resulted in their incorrect correlation.

By scanning the first hour traffic of the second day, Snort generated 221 new alerts. Among them, 2 new alert types were found with 82 alerts all together. This requires the recalculation of correlation probability of these two new alert types with the existing ones. The TPC rate without such recalculation is 93.2%, while after recalculation the TPC rate increases to 96.1%. Moreover, since the occurrence probability of some types of alerts suddenly decreases, 4 alert type pairs appear to be no longer correlated. Thus, the FPC rate decreases to 14%.

Performance. In this experiment, we focused on the evaluation of performance of both offline and online components. The experiments were run on Intel(R) Core(TM)2 with CPU of 2.4GHz. For the evaluation we employed the Honeynet traffic. The offline component was trained based on the first 8000 alerts generated by Snort and the online correlation was performed on the other half of alerts (7602 alerts) which constitutes around 80 hours. The results are presented in Figure 10. The online component was run in two modes with the dynamic adaptation to the current alert behavior and without. For this experiment, the online correlation module was configured to perform incremental correlation on one hour basis, i.e., the attack scenarios were continuously updated during one hour period, after this period expired new graphs were started. As Figure 10 (b) shows, the correlation process with the dynamic adaptation on average takes 3015 ms to process alerts triggered during one hour (on average 96 alerts). This is a reasonable processing time requirement that we consider suitable for an offline as well as an online analysis.

7 Conclusion and Future Work

In this paper, we presented a novel approach for adaptive online alert correlation. The approach incorporates two components: the offline module that is responsible for retrieving relevant attack information from the previously observed alerts based on Bayesian causality mechanism; and the online component that is based on the extracted information correlates raw alerts and constructs attack scenarios online. In addition to historic alert information, the proposed approach is able to dynamically adjust to the current alert behavior and reflect it in the correlation process. The advantages of the proposed correlation method were examined using DARPA 2000 data sets and live Honeynet data. In the future, we plan to deploy the proposed approach in the real world environment, which might bring new insights into our approach advantages.

Acknowledgements

This work was funded by the Atlantic Canada Opportunity Agency (ACOA) through the Atlantic Innovation Fund (AIF) and NSERC through grant RGPIN-2277441 to Dr. Ali A. Ghorbani.

References

1. Valdes, A., Skinner, K.: Probabilistic alert correlation. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 54–68. Springer, Heidelberg (2001)
2. Ning, P., Cui, Y., Reeves, D.S.: Constructing attacks scenarios through correlation of intrusion alerts. In: Proceedings of the 9th ACM conference on Computer and communications security, pp. 245–254 (2002)
3. Cheung, S., Lindqvist, U., Fong, M.: Modeling multistep cyber attacks for scenario recognition. In: DARPA Information Survivability Conference and Exposition, vol. 1, pp. 284–292 (2003)
4. Cuppens, F., Miege, A.: Alert correlation in a cooperative intrusion detection framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 202–215 (2002)
5. Cuppens, F., Ortalo, R.: A language to model a database for detection of attacks. In: Debar, H., Mé, L., Wu, S.F. (eds.) RAID 2000. LNCS, vol. 1907, pp. 197–216. Springer, Heidelberg (2000)
6. Eckmann, S.T., Vigna, G., Kemmerer, R.A.: Statl: An attack language for state-based intrusion detection. Journal of Computer Security 10, 71–103 (2002)
7. Totel, E., Vivinis, B., Mé, L.: A language driven IDS for event and alert correlation. In: SEC, pp. 209–224 (2004)
8. Qin, X.: A probabilistic-based framework for INFOSEC alert correlation. In: Proceedings of the Symposium on Recent Advances in Intrusion Detection, vol. 2820, pp. 73–93 (2003)
9. Zhu, B., Ghorbani, A.A.: Alert correlation for extracting attack strategies. International Journal of Network Security, 244–258 (2006)
10. Sadoddin, R., Ghorbani, A.A.: An incremental frequent structure mining framework for real-time alert correlation. Computers and Security 28, 153–173 (2009)

11. Zhang, S., Li, J., Chen, X., Fan, L.: Building network attack graph for alert causal correlation. Computers and Security 27, 188–196 (2008)
12. Maggia, F., Matteuccia, M., Zanero, S.: Reducing false positives in anomaly detectors through fuzzy alert aggregation. Information Fusion 10, 300–311 (2009)
13. Julisch, K.: Clustering intrusion detection alarms to support root cause analysis. ACM Transactions on Information and System Security 6, 443–471 (2002)
14. Pietraszek, T.: Using adaptive alert classification to reduce false positives in intrusion detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 102–124. Springer, Heidelberg (2004)
15. Manganaris, S., Christensen, M., Zerkle, D., Hermiz, K.: A data mining analysis of rtid alarms. Computer Networks: The International Journal of Computer and Telecommunications Networking 34, 571–577 (2000)
16. Viinikka, J., Debar, H., Mé, L.: Processing intrusion detection alert aggregates with time series modeling. Information Fusion 10, 312–324 (2009)
17. Morin, B., Mé, L., Debar, H., Ducasse, M.: M2d2: A formal data model for IDS alert correlation. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 115–137. Springer, Heidelberg (2002)
18. Morin, B., Mé, L., Debar, H., Ducasse, M.: A logic-based model to support alert correlation in intrusion detection. Information Fusion 10, 285–299 (2009)
19. Porras, P.A., Fong, M.W., Valdes, A.: A mission-impact-based approach to infosec alarm correlation. In: Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection, pp. 95–114 (2002)
20. Dain, O.M., Cunningham, R.K.: Fusing a heterogeneous alert stream into scenarios. In: Proceeding of the 2001 ACM Workshop on Data Mining for Security Applications, pp. 1–13 (2001)
21. Qin, X., Lee, W.: Discovering novel attack strategies from INFOSEC alerts. In: Proceedings of the 9th European Symposium on Research in Computer Security, Sophia Antipolis, pp. 439–456 (2004)
22. Haykin, S.: Neural networks: A comprehensive foundation, 2nd edn. (1998)
23. Cristianini, N., Taylor, J.S.: An introduction to support vector machines and other kernel-based learning methods (2000)
24. Heckerman, D.: A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research (1995)
25. Laboratory, M.L.: 2000 darpa intrusion detection scenario specific datasets (2000)
26. netForensics Honeynet team: Honeynet traffic logs, `http://old.honeynet.org/scans/scan34/`