Discovering Novel Attack Strategies from INFOSEC Alerts

Xinzhou Qin and Wenke Lee

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332, USA
{xinzhou, wenke}@cc.gatech.edu

Abstract. Correlating security alerts and discovering attack strategies are important and challenging tasks for security analysts. Recently, there have been several proposed techniques to analyze attack scenarios from security alerts. However, most of these approaches depend on *a priori* and hard-coded domain knowledge that lead to their limited capabilities of detecting new attack strategies. In this paper, we propose an approach to discover novel attack strategies. Our approach includes two complementary correlation mechanisms based on two hypotheses of attack step relationship. The first hypothesis is that attack steps are directly related because an earlier attack enables or positively affects the later one. For this type of attack relationship, we develop a Bayesian-based correlation engine to correlate attack steps based on security states of systems and networks. The second hypothesis is that for some related attack steps, even though they do not have obvious and direct relationship in terms of security and performance measures, they still have temporal and statistical patterns. For this category of relationship, we apply time series and statistical analysis to correlate attack steps. The security analysts are presented with aggregated information on attack strategies from these two correlation engines. We evaluate our approach using DARPA's Grand Challenge Problem (GCP) data sets. The results show that our approach can discover novel attack strategies and provide a quantitative analysis of attack scenarios.

1 Introduction

A large-scale deployment of information security (INFOSEC) mechanisms can provide in-depth protection for systems and networks. However, the sheer quantity of low-level or incomplete alerts output by INFORSEC devices can overwhelm and prevent security analysts from making thorough analysis and rapid response. Therefore, it is important to develop an advanced alert correlation system that can reduce the redundancy of alarms, intelligently correlate security alerts, and detect attack strategies. Alert correlation is therefore a core component in a security management system.

Recently, there have been several alert correlation proposals. With respect to correlation techniques, most of the proposed approaches (e.g., [5, 9, 12, 22]) rely on various forms of prior knowledge of individual attacks such as attack pre-conditions and consequences. It is difficult for these approaches to recognize *new* attack strategies where the attack or the relationship between attacks is new. It is obvious that the number of

possible correlations is very large, potentially a combinatorial of the number of (known and new) attacks. Therefore, it is infeasible to know *a priori* and encode all possible matching conditions between attacks. In fact, dangerous and intelligent adversaries will invent new attacks and novel attack strategies especially in information warfare.

We have two motivations in our work. First, we want to develop an alert correlation system that can discover *new* attack strategies without relying solely on domain knowledge. Second, we want to incorporate more evidence or indicators from other non-security monitoring systems to correlate alerts and detect attack strategies. For example, we can incorporate alerts from network management systems (NMS) into the security alert correlation. Although alerts from NMS may not directly tell us what attacks are present, they provide us information on the state of protected domains.

Our main contribution in this paper is the design of an integrated correlation system to discover novel attack strategies from INFOSEC alerts. The system includes two complementary correlation engines based on two hypotheses of relationships between attack steps. The first hypothesis is that attack steps are directly related because an earlier attack enables or positively affects the later one. For example, a port scan may be followed by a buffer overflow attack against a scanned service port. For this type of direct relationship, we develop a Bayesian-based correlation mechanism to reason and correlate attack steps based on security states of systems and networks. Our Bayesian-based correlation mechanism uses probabilistic reasoning technique and incorporates domain knowledge of individual attacks to reason and correlate alerts. Our approach does not rely on the strict pre-/post-condition matching and can also function on the partial correlation evidence. The second hypothesis is that some related attack steps still have temporal and statistical patterns even though they do not have an obvious or direct relationship in terms of security and performance measures. For this category of relationship, we apply statistical analysis to correlate attack steps. This correlation mechanism does not rely on prior knowledge of attacks. It correlates alerts by investigating and testing the temporal and statistical patterns of attack steps. Therefore, it is analogous to *anomaly detection*.

We evaluate our methods using DARPA's Grand Challenge Problem (GCP) data sets [8]. The results show that our approach can successfully discover new attack strategies and provide a quantitative analysis method to analyze attack strategies.

The remainder of this paper is organized as follows. Section 2 discusses the related work. We present our alert correlation approach in Section 3. In Section 4, we report the experiments and results on the GCP. We summarize the paper and point out some ongoing and future work in Section 5.

2 Related Work

Recently, there have been several proposed techniques of alert correlation and attack scenario analysis.

Valdes and Skinner [30] use probabilistic-based reasoning to correlate alerts by measuring and evaluating the similarities of alert attributes. Alert aggregation and scenario construction are conducted by enhancing or relaxing the similarity requirements in some attribute fields. Goldman et al. [12] build a correlation system based on Bayesian

reasoning. The system predefines the relationship between mission goals and corresponding security events for further inference and correlation.

Porras et al. design a "mission-impact-based" correlation system with a focus on the attack impacts on the protected domains [26]. The system uses clustering algorithms to aggregate and correlate alerts. Security incidents are ranked based on the security interests and the relevance of attack to the protected networks and systems.

Debar and Wespi [9] apply backward and forward reasoning techniques to correlate alerts with *duplicate* and *consequence* relationship. They use clustering algorithms to detect attack scenarios and situations. This approach pre-defines consequences of attacks in a configuration file.

Morin and Debar [21] apply chronicle formalism to aggregate and correlate alerts. The approach performs attack scenario pattern recognition based on *known* malicious event sequences. Therefore, this approach is similar to *misuse detection* and cannot detect new attack sequences.

Ning et al. [22], Cuppens and Miège [7] and Cheung et al. [5] build alert correlation systems based on matching the pre-/post-conditions of individual alerts. The idea of this approach is that prior attack steps prepare for later ones. Therefore, the consequences of earlier attacks correspond to the prerequisites of later attacks. The correlation engine searches alert pairs that have a consequence and prerequisite matching. Further correlation graphs can be built with such alert pairs [22]. One challenge to this approach is that a new attack cannot be paired with any other attacks because its prerequisites and consequences are not defined. Recently, Ning et al. [24] have extended the pre-/post-condition-based correlation technique to correlate some isolated attack scenarios by hypothesizing missed attack steps.

Our approach aims to address the challenge of how to detect *novel* attack strategies. Our approach differs from other work in the following aspects. First, our approach integrates two complementary correlation engines to discover attack scenario patterns. We apply a Bayesian-based correlation engine to the attack steps that are directly related because prior attack enables the later one. Our Bayesian-based correlation engine differs from previous work in that we incorporate knowledge of attack step transitions as a constraint when conducting probabilistic inferences. The correlation engine makes the inference about the correlation based on broad indicators of attack impacts without using the strict hard-coded pre-/post-condition matching. We apply a statistical-based correlation engine to attack steps with temporal and statistical patterns. This approach differs from previous work in that it does not rely on prior knowledge of attack strategies or pre-/post-conditions of individual attacks. Therefore, this approach can be used to discover *new* attack strategies. In this respect, our approach is analogous to *anomaly detection* technique. To the best of our knowledge, this is the first approach to detecting new attack strategies. Our integrated approach also provides a quantitative analysis of the likelihood of various attack paths. With the aggregated correlation results, security analysts can perform further analysis and make inferences about high-level attack plans.

3 Alert Correlation

In this section, we introduce our two complementary correlation mechanisms based on probabilistic and statistical reasoning techniques respectively. In particular, we apply the Bayesian network to probabilistic inference and use *Granger Causality Test* (GCT) [13][1] for statistical analysis.

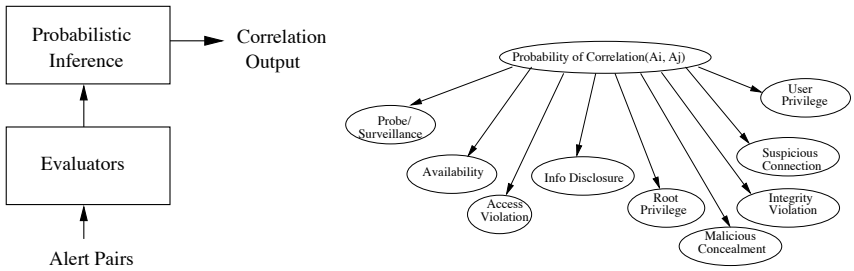
In our framework, we first *aggregate and cluster* raw alerts, then *prioritize* the aggregated alerts before conducting further alert correlation. The corresponding algorithms for alert aggregation and prioritization can be found in our prior work [27]. Briefly, alert aggregation and clustering reduces the redundancy of raw alerts while retaining important alert attributes, such as *time stamp*, *source IP*, *destination IP*, *port(s)*, *attack class*. In this step, alerts corresponding to the same attacks from heterogeneous security sensors are aggregated. Aggregated alerts with the same attributes (except time stamps) are grouped into one cluster, called **hyper alert**. Alert prioritization is to rank each hyper alert based on its relevance to the configuration of protected networks and hosts, as well as the severity of the corresponding attack assessed by the security analyst. The relevance check downgrades the impacts of some alerts unrelated to the protected domains. For example, an attacker may blindly launch a buffer overflow attack against a host without knowing if the corresponding service exists or not. In practice, it is quite possible that a signature-based IDS will output an alert once the packet contents match the detection rules even though the service does not exist on the target host. This type of alert has a low priority.

3.1 Probabilistic Reasoning on Alert Correlation

Motivation In practice, we observe that when a host is compromised by an attacker, it usually becomes the target of further attacks or a stepping-stone for launching attacks against other systems. Therefore, the consequences of an attack on a compromised host can be used to reason about a possible matching with the goals of another attack. It is possible to address this correlation by defining pre-/post-conditions of individual attacks and applying condition matching. However, it is infeasible to enumerate and precisely encode all possible attack consequences and goals into pre-/post-conditions. Therefore, we apply probabilistic reasoning to alert correlation by incorporating system indicators of attack consequences and prior knowledge of attack transitions. In this section, we discuss how to apply probabilistic reasoning to attack consequences and goals in order to discover the subtle relationships between attack steps in an attack scenario.

Model Description Figure 1(a) shows the procedure of correlation inference. Given a stream of alerts, *evaluators* first analyze one or more features of alert pairs and output results as evidence to the *inference module*. The *inference module* combines the individual opinions expressed by the evaluators into a single assessment of the correlation by computing and propagating correlation beliefs within the inference network.

[1] In alert correlation, "causality" should be interpreted as correlation instead of conventional meaning of "causality". The term "cause" used between attack steps should be interpreted as attack step transition.



(a) Inference flowchart (b) Bayesian-based correlation model

Fig. 1. Probabilistic reasoning model

In our inference module, we use a Bayesian network [25] as our reasoning engine. Bayesian networks are usually used as a principle method to reason uncertainty and are capable of leveraging prior expert opinions with the learned information from data. A Bayesian network is usually represented as a directed acyclic graph (DAG) where each node represents a variable that has a certain set of states, and the directed edges represent the causal or dependent relationships among the variables. A Bayesian network consists of several parameters, i.e., prior probability of parent node's states (i.e., $P(parent\_state = i)$), and a set of conditional probability tables (CPT) associated with child nodes. CPT encodes the prior knowledge between child node and its parent node. Specifically, an entry of the CPT at a child node is defined by $CPT_{ij} = P(child\_state = j | parent\_state = i)$.

Figure 1(b) shows the structure of our Bayesian inference model for pair-wise correlation. Since we depend on domain knowledge to correlate directly related alert pairs, we design a one-level Bayesian network that is good enough to perform inference. In the inference model, the root node represents the *hypothesis* that two attacks are correlated. Specifically, the root node has two hypothesis states, i.e., "high correlation" and "low correlation". Each child node represents a type of attack consequences on the host. The evaluator on each child node detects the condition matching between the consequences and the necessary conditions of the two alerts being correlated. The evaluation result on each leaf node is mapped to a state of the child node. Each child node has three states: "matched", "not matched" and "unknown". The state "unknown" handles the case that there is no need of condition matching, e.g., some attacks do not necessarily have any pre-conditions in order to launch attacks. The output of the inference engine represents the probability or confidence of the correlation between two alerts being correlated, i.e., $P(correlation = high | evidence)$, based on the evidence (e.g., "matched" or "unmatched") provided by the leaf nodes. The inference is conducted by propagating belief messages among leaf nodes and the root node. Specifically, we denote e^k as the k^{th} leaf node and H_i as the i^{th} hypothesis of the root node. Given the evidence from the leaf nodes, assuming conditional independence with respect to each H_i, the belief in hypothesis at the root is: $P(H_i \mid e^1, e^2, \ldots, e^N) =$

$\gamma P(H_i) \prod_{k=1}^{N} P(e^k|H_i)$, where $\gamma = [P(e^1, e^2, \ldots, e^N)]^{-1}$ and γ can be computed using the constraint $\sum_i P(H_i|e^1, e^2, \ldots, e^N) = 1$ [25]. Since the belief computation can be performed incrementally instead of being delayed until all the evidence is collected, the Bayesian inference engine can also function on partial evidence, and the lack of evidence input from an evaluator does not require special treatment.

As Figure 1(b) shows, each leaf node represents an attack consequence on the attack victim. We consider broad aspects of attack consequences when reasoning about the correlation between two alerts.

Probe/Surveillance: information on system or network has been gained by an attacker, e.g., a probing attack can get information on open ports. *Availability*: the system is out of service or the service is negatively affected by the attack, e.g., because of a DoS attack. *Access Violation*: an illegal access to a file or data of a system. *Information Disclosure*: the attacker exports (sensitive) data to external site. *Root Privilege* has been obtained by an attacker, for example, by a buffer overflow attack. *Malicious Concealment*: malicious binary codes have been installed on the system, e.g., a Trojan horse. *Integrity Violation*: the file on a system has been modified or deleted, violating the security policy. *Suspicious Connection*: a covert channel has been set up by the attack. *User Privilege* has been obtained by the attacker.

FailService	DegradeService	FailProcess
DegradeProcess	ModifyData	DeleteData
GainUserPrivilege	GainRootPrivilege	GainServiceInfo
GainOSInfo	InstallMaliciousDaemon	InstallTrojan
SetupCovertChannel	FailCovertChannel	ExportData
GainFile	AccessSystem	LeakInformation

Table 1. Predicates used in impact evaluation

Table 1 shows the set of predicates that we defined to assess the consequences of attack. Each attack impact shown in Figure 1(b) has been associated with a set of predicates defined in Table 1. For example, predicates "FailService" and "DegradeService" represent the attack impacts on the availability of the target's service. The definition of predicates is a broad template and each predicate can be instantiated to a specific consequence instance according to information provided by alerts. For example, when a port scan alert is output, its corresponding impact instance is *GainServiceInfo.TargetIP*. Each alert has also been defined a *pre-condition(s)* using the predicates shown in Table 1. Like the definition of impact of attack, *pre-condition(s)* of each alert can also be instantiated based on alert specific attributes. Each alert can provide the necessary information from its attributes, such as *source IP*, *target IP*, *attack class*.

Correlating two alerts includes the following steps. First, each alert first initializes its corresponding *pre-condition* and *impact* fields. Second, alert pairs are checked to see if they comply with certain constraints, e.g., an implicit temporal constraint between these two alerts is that alert A_i occurs before alert A_j. Third, evaluations are conducted by comparing the "*causal*" alert's impacts and *effected* alert's pre-conditions on each of the leaf nodes as shown in Figure 1. Fourth, results of evaluations are mapped to the

states of leaf nodes, i.e., "matched", "unmatched" and "unknown". Finally, an overall probability computation is conducted based on the state evidence of each leaf node.

For example, alert $portscan$ has a consequence defined as *GainServiceInfo.targetIP*. Alert *imap buffer overflow* has a pre-condition as *GainServiceInfo.targetIP*, where predicate "GainServiceInfo" is associated with attack consequence *Probe* shown in Figure 1(b). If $portscan$ alert occurs before alert *imap buffer overflow* and they have the same target IP addresses, then their pre-/post-conditions are matched. The corresponding state of leaf node *Probe/Surveillance* in Figure 1(b) will be set as "matched". The Bayesian-model computes the evidence and outputs the probability or confidence of the correlation of these two alerts.

Parameters in Bayesian Model When using a Bayesian model for inference, we need to set up two types of parameters, i.e., prior probability of root's states and CPT associated with each child node.

The prior probability of root states (e.g., $P(correlation = high)$) used in the inference engine is set based on the attack class of alerts being correlated. It indicates the *prior knowledge* estimation of the possibility that one attack class reasonably transits to another one. For example, it is reasonable for us to have a higher estimation of the possibility that an exploit attack follows a probe than the other way around. We use domain-specific knowledge based on prior experience and empirical studies to estimate appropriate probability values. Related work [30] also helps us on the probability estimation.

In alert correlation, the pair of alerts being evaluated in the correlation engine (as shown in Figure 1(b)) is only known at run-time. Therefore, we cannot use an inference engine with a fixed set of CPT parameters. Instead, we set up a set of CPTs based on each pair of attack classes (e.g., *Malicious Concealment* and *DoS*). At run-time, when correlating a pair of alerts A_i and A_j with respective corresponding attack classes $C(A_i)$ and $C(A_j)$ (e.g., alert *imap buffer overflow* with attack class $Super\ Privilege\ Violation$ and alert $illegal\ file\ access$ with attack class $Access\ Violation$), the inference engine selects the corresponding CPT parameters for the attack classes $C(A_i)$ and $C(A_j)$, and computes the overall probability that A_j is "caused" by A_i given the evidence from the evaluators, i.e., $P(correlation = high|e = evidence)$. An implicit temporal constraint between these two alerts is that alert A_i occurs before A_j. In this example, we can interpret the correlation as: the *imap buffer overflow* attack is followed by an illegal access to a file after the attacker gets root privileges on the target. Initial values of CPTs are pre-defined based on our experience and domain knowledge.

CPT values associated with each node adapt to new evidence and therefore can be updated accordingly. We apply an adaptive algorithm originally proposed by [1] and further developed by [6]. The motivation of using adaptive Bayesian network is that we want to fine-tune the parameters of the model and adapt the model to the evidence to fix the initial CPTs that may be pre-defined inappropriately. The intuition of the algorithms proposed by [1] is that we want to adapt the new model by updating CPT parameters to fit the new data cases while balancing movement away from the current model.

Specifically, we denote X as a node in a Bayesian network, and let U be the parent node of X. X has r states with values of x_k, where $k = 1, ..., r$ and U has q states

with values of u_j, where $j = 1, ..., q$. An entry of CPT of the node X can be denoted as: $\theta_{jk} = P(X = x_k|U = u_j)$. Given a set of new data cases, denoted as D, $D = y_1, ..., y_n$, and assuming there is no missing data in evidence vector of y_t, where evidence vector y_t represents the evidence at the t^{th} time, the CPT updating rules are:

$$\theta^t_{jk} = \eta + (1 - \eta)\theta^{t-1}_{jk}, \ \ for\ P(u_j|y_t) = 1\ and\ P(x_k|y_t) = 1. \tag{1}$$

$$\theta^t_{jk} = (1 - \eta)\theta^{t-1}_{jk}, \ \ for\ P(u_j|y_t) = 1\ and\ P(x_k|y_t) = 0. \tag{2}$$

$$\theta^t_{jk} = \theta^{t-1}_{jk}, \ otherwise. \tag{3}$$

η is the learning rate. The intuition of the above updating rules is that, for an entry of CPT, e.g., θ_{mn}, we either increase or decrease its value (i.e., $P(X = x_n|U = u_m)$) based on the new evidence received. Specifically, given the evidence vector y_t, if the parent node U is observed in its m^{th} state, i.e., $U = u_m$, and X is in its n^{th} state, i.e., $X = x_n$, we regard the evidence as *supporting evidence* of the CPT entry θ_{mn}. We then increase its value (i.e., $P(X = x_n|U = u_m)$), which indicates the likelihood that X is in its n^{th} state given the condition that parent node U is in its m^{th} state, as shown in Eq. (1). By contrast, if node X is not in its n^{th} state while its parent node U is in the m^{th} state, we then regard the evidence as *un-supporting evidence* of θ_{mn} and decrease θ_{mn}'s value as shown in Eq. (2). We do not change the value of θ_{mn} if no corresponding evidence is received. The learning rate η controls the rate of convergence of θ. η equaling 1 yields the fastest convergence, but also yields a larger variance. When η is smaller, the convergence is slower but eventually yields a solution to the true CPT parameter [6]. We build our inference model based on above updating rules.

We also need to point out that the adaptive capability of the inference model does not mean that we can ignore the accuracy of initial CPT values. If the initial values are set with a large variance to an appropriate value, it will take time for the model to converge the CPT values to the appropriate points. Therefore, this mechanism works for fine-tuning instead of changing CPT values dramatically.

For an alert pair, (A_i, A_j), if its correlation value computed by the Bayesian-based model, denoted as P_{bayes}, is larger than a pre-defined threshold, e.g., 0.5, then we say Bayesian-based correlation engine identifies that alert A_j is "caused" by alert A_i.

Alert correlation with Bayesian networks has several advantages. First, it can incorporate prior knowledge and expertise by populating the CPTs. It is also convenient to introduce partial evidence and find the probability of unobserved variables. Second, it is capable of adapting to new evidence and knowledge by belief updates through network propagation. Third, the correlation output is probability rather than a binary result from a logical combination. We can adjust the correlation engine to have the maximum detection rate or a minimum false positive rate by simply adjusting the probability threshold. By contrast, it is not directly doable when using a logical combination of pre-/post-condition matching. Finally, Bayesian networks have been studied extensively and successfully applied to many applications such as causal reasoning, diagnosis

analysis, event correlation in NMS, and anomaly detection in IDS. We have confidence that it can be very useful to INFOSEC alert correlation.

3.2 GCT-based Alert Correlation

The motivation to develop another complementary correlation mechanism is that many existing correlation techniques depend on various forms of domain knowledge of attack scenario patterns. This is similar to *misuse detection*. In order to discover *new* attack strategies that are beyond the scope of *prior* knowledge on attack scenarios, we develop another correlation engine based on statistical analysis, in particular, the Granger Causality Test (GCT) [13]. In this section, we briefly introduce our GCT-based correlation mechanism. Details can be found in [27].

Granger Causality Test (GCT) is a time series-based *statistical* analysis method that aims to test if a time series variable *X* correlates with another time series variable *Y* by performing a *statistical hypothesis test*. Although GCT was originally proposed and applied in econometrics, it has been widely applied in other areas, such as weather analysis (e.g., [18]), automatic control system (e.g., [4, 11]) and neurobiology (e.g., [17, 16]). In our prior work [3, 2], we have applied GCT-based analysis for pro-active detection of Distributed-Denial-of-Service (DDoS) attacks using MIB II [29] variables. The results have demonstrated the correlation strength of GCT in network security context.

In this work, we apply the GCT to alert streams for alert correlation and analysis. The hypothesis and intuition is that attack steps that do not have well-known patterns or obvious relationships may nonetheless have some *temporal and statistical* correlations in the alert data. For example, two attacks are associated when one or more alerts for one attack also occurs with one or more alerts for another attack. We can apply time series and statistical correlation analysis to correlate such alert pairs and describe the attack scenario.

Applying GCT to data analysis requires a series of statistical tests including testing if an individual data set is statistically stationary, if two time series variables are statistically independent of each other, and if they are co-integrated. Briefly, when applying the GCT to alert correlation, we test the statistical correlation of alert instances to determine the relationship between hyper alerts (i.e., aggregated alerts with same attributes except time stamp). Specifically, the correlation includes the following steps when identifying the "causal" alert with respect to hyper alert A. (1) For each pair of hyper alerts $(B_i, A), i = 1, 2, \ldots, l$, we compute the value of Granger Causality Index (GCI) g_i, which represents the strength of the "causal" relationship between B_i and A. (2) Given a significance level, we record the alerts whose GCI values have *passed* the F-test as the "causal" candidate alerts, and rank the candidate alerts according to their GCI values. (3) We then select the top *m* candidate alerts and regard them as being "causally" related to alert A. (4) These (candidate) "causal" relationships can be subject to more inspection by other analysis techniques.

The main advantage of GCT-based correlation engine is that it does not require *a priori* knowledge about attack behaviors and how the attacks can be related. This approach can identify the correlation between two attack steps if they have a statistical pattern, e.g., they repeatedly occur together. We believe that there are a large number of attacks, e.g., worms, with such attack steps. Thus, we believe that causality analysis is a

very useful technique. As also discussed in [3,2], when there is sufficient training data available, we can use GCT off-line to compute and validate very accurate "causal" relationships from alert data. We can then update the knowledge base with these "known" correlations for efficient pattern matching in run-time. When GCT is used in real-time and finds a new "causal" relationship, the top m candidates can be selected for further analysis by other techniques.

As a statistical data analysis tool, GCT also has its limitations because it studies the correlation between variables from a *statistical* point of view. Like any other statistical analysis techniques, the analysis result depends on the existence of statistical patterns in the data. GCT can also result in false causality if two unrelated alerts happen to have a strong statistical pattern. Lee et al. [19] empirically report the "pitfalls" of GCT when applying it to *co-integrated* time series variables. In our analysis, we test the co-integration of data sets before applying GCT to avoid the inaccuracy.

3.3 Integration of GCT-Based and Probabilistic Reasoning Correlation Mechanisms

Integration Process of Two Correlation Engines Our two correlation engines are built on different techniques and focus on different correlation aspects. GCT-based correlation engine is similar to *anomaly detection*. Bayes-based correlation engine is analogous to an extension of pattern matching-based detection. We apply and integrate the two correlation mechanisms with the following steps:

(1) First, we apply Bayesian-based correlation engine on target hyper alerts. Target alerts are hyper alerts with high priorities computed by the *alert priority computation module* [27]. Thus, they should be the main interests in the correlation analysis to correlate with all the other hyper alerts. The result of this step can be a set of isolated correlation graphs.

(2) Second, for each target alert, we run GCT to correlate it with other hyper alerts that have not been identified as "causally" related to the target alert by Bayesian correlation engine. That is, GCT is used to attempt to discover more correlation between alerts and link the isolated graphs together.

For example, we have five hyper alerts, denoted as A_1, A_2, A_3, A_4, A_5. Alerts A_1, A_2 are target alerts. After applying Bayesian-based correlation engine, i.e., the first step of correlation, we get two isolated correlation graphs, as shown in Figure 2. The directed edge indicates the direction from "causal" alerts to the target alerts. Alert A_1 is correlated with alerts A_3 and A_4. Alert A_2 is correlated with alert A_5. In step 2, for alert A_1, we run GCT(A_2, A_1) and GCT(A_5, A_1) to check if they have any relationships. For alert A_2, we run GCT(A_1, A_2), GCT(A_3, A_2) and GCT(A_4, A_2) to test if there are any correlation. If we can find new relationship in step 2, then we can link these two isolated graphs. For example, the bold line in Figure 2 shows the new "causal" relationship from A_4 to A_2 identified in step 2.

The rationale of our integration process in alert correlation is analogous to intrusion detection where security analysts usually first apply *pattern-based detection*, then *anomaly detection* to cover the attack space that pattern-matching method cannot discover.

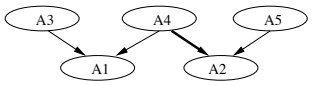


Fig. 2. An example of integration process. The bold line represents a new correlation found in the second step of integration process.

Probability/Confidence Integration In Section 3.1, we introduced our Bayesian-based correlation engine that outputs the correlation probability/confidence of two alerts, denoted as P_{bayes}. In practice, we have a threshold t, and when P_{bayes} is over the threshold t, we say the corresponding alert pair has a "causal" relationship identified by the Bayesian-based correlation engine. As discussed in Section 3.2, GCT Index (GCI) represents the strength of correlation between two alerts being correlated. It conforms to F-distribution with parameters of p and $N-3p-1$, where p is the number of history values of the time series variable used in the GCT computation, and N is the size of the time series variable. Therefore, for any two correlated alerts identified by GCT-based correlation engine, we can compute the corresponding F-distribution probability values, i.e., $P_{gct} = CDF_{F-distribution}(p, N-3p-1, GCI)$, where CDF represents the *cumulative distribution function*. P_{gct} represents the probability/confidence of correlation between two alerts.

When integrating the two correlation engines, we can normalize the confidence output from GCT-based engine as:

$$P_{gct\_normalized} = (P_{gct} - t) * \omega + t \tag{4}$$

In Eq. (4), t is the threshold defined in Bayesian-based correlation engine, and ω is a weight value that is determined based on prior experience and performance measurements of the two correlation engines. The normalized value of $P_{gct\_normalized}$ is in the range of $[0, t+\epsilon]$, where ϵ is a small positive number. The intuition of this normalization is that we want to downgrade the output of GCT-based correlation engine a little because it is based on statistical analysis that is less accurate than the domain-knowledge-based Bayesian correlation engine.

Therefore, for a correlated alert pair, e.g., (A_i, A_j), we can have a probability or confidence of its correlation (i.e., attack transition from A_i to A_j) computed by either Bayesian correlation engine or GCT-based correlation mechanism. We denote it as $correlation\_prob(A_i, A_j)$, which equals P_{bayes} when their "causal" relationship is identified by Bayesian engine or equals $P_{gct\_normalized}$ when GCT discovers its relationship.

We also note that two different approaches have been proposed to integrate isolated correlation graphs. Ning [23] et al. apply graph theory to measure and merge similar correlation graphs. In [24], Ning et al. link isolated correlation graphs based on attack pre-/post-conditions. Our approach is different from their work in that our integration method is based on the correlation probability evaluated by our two complementary correlation engines instead of graph or pre/post-condition-based merging algorithms.

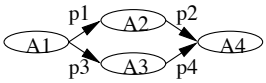


Fig. 3. An example of correlation graph

Attack Strategy Analysis A scenario/correlation graph can be constructed based on pairs of correlated alerts. A scenario graph is defined as a directed graph where each edge E_{ij} represents a "causal" relationship from alert A_i to A_j. Alerts with "causal" relationship compose the nodes in the scenario graph. We denote the node corresponding to the "causal" alert as *causal node*, and the node corresponding to the "effected" alert as *effected node*. A threshold t is pre-defined and alert A_j is considered to be "caused" by alert A_i only when $correlation\_prob(A_i, A_j) > t$. In constructing scenario graphs, we only include the correlated alert pairs whose $correlation\_prob$ values are over the threshold t.

In a correlation graph, each edge is associated with a correlation probability (i.e., $correlation\_prob$) from *causal node* to *effected node*. Therefore, we can perform *quantitative* analysis on the attack strategies. Each path in the graph is potentially a subsequence of an attack scenario. Each path can be seen as a Markov chain [10, 28]. Therefore, based on the probability associated with each edge, for any two nodes in the graph that are connected by multiple paths, e.g., nodes A_1 and A_4 in the Figure 3, assuming the conditional independence of A_4 and A_1, we can compute the overall probability of each path, e.g., $P(A_1, A_2, A_4) = P(A_4|A_2)P(A_2|A_1)P(A_1) = p_1 * p_2 * p_{A1}$ [28], and then rank order and select the one with the highest overall correlation probability as the most likely sequence connecting the two alerts. Combining all the probability along each edge, an overall probability of two nodes connected with multiple paths can also be computed. For example, in the Figure 3, $P\{A_1\ to\ A_4\} = 1-(1-p_1*p_2)(1-p_3*p_4)$.

4 Experiments

To evaluate the effectiveness and validity of our alert correlation mechanisms, we applied our algorithms to the data sets of the Grand Challenge Problem (GCP) version 3.1 provided by DARPA's Cyber Panel program [8, 15]. In this section, we describe our experiments with a focus on the analysis of GCP I.

4.1 The Grand Challenge Problem (GCP)

GCP version 3.1 includes two innovative worm attack scenarios to specifically evaluate alert correlation techniques. In addition to the complicated attack scenarios, the GCP data sets also include many background alerts that make alert correlation and attack strategy detection more challenging. In GCP, multiple heterogeneous security systems, e.g., network-based IDSs, host-based IDSs, firewalls, and network management systems, are deployed in several network enclaves. Therefore, GCP alerts are from both security systems and network management system. GCP alerts are in the Intrusion Detection Message Exchange Format (IDMEF) defined by IETF [14].

In order to compare the performance between our current integrated correlation system and the GCT-alone approach used in [27], we used the same data sets and pre-processed the raw alerts the same way as in [27]. According to the GCP documents that include detailed configurations of protected networks and systems, we established a configuration database. Information on mission goals enables us to identify the servers of interest and assign interest score to corresponding alerts targeting at the important hosts. The alert priority is computed based on our model described in [27].

For performance evaluation, we define two measures: *true positive correlation rate*, (i.e., $(\#\ of\ correct\ correlated\ alerts)/(total\ \#\ of\ correlated\ relationships)$) and *false positive correlation rate*, (i.e., $(\#\ of\ incorrect\ correlated\ alerts)/(total\ \#\ of\ correlated\ alerts)$). Here, *correlated alerts* refer to the correlated alert pairs output by correlation engines. We refer to the documents with the ground truth to determine the *correlated relationships* among the alerts. Scenario graph is constructed based on alerts that have causal relationship identified by our correlation engines.

In formulating hyper alert time series, we set the unit time slot to 60 seconds. In the GCP, the entire time range is 5 days. Therefore, each hyper alert time series $x(k)$ has a size of 7,200 (units), i.e., k=0, 1, 2, ..., 7199.

GCP Scenario I In the GCP Scenario I, there are multiple network enclaves in which attacks are conducted separately. The attack scenario in each network enclave is almost same. We select a network enclave as an example to show the correlation process.

The alert correlation processing is the following:

First, **alert aggregation**. We conduct raw alert aggregation and clustering in order to have aggregated hyper alerts. In scenario I, there are a little more than 25,000 low-level raw alerts output by heterogeneous security devices in all enclaves. After alert fusion and clustering, we have around 2,300 hyper alerts. In our example network enclave, there are 370 hyper alerts after low-level alert aggregation.

Second, **alert noise detection**. We apply the *Ljung-Box* statistical test [20] with significance level $\alpha = 0.05$ to all hyper alerts in order to identify background alerts. In scenario I, we identify 255 hyper alerts as background alerts using this mechanism. Most of background alerts are "HTTP_Cookie" and "HTTP_Posts". Therefore, we have 115 non-noise hyper alerts for further analysis.

Third, **alert prioritization**. The next step is to select the alerts with high priority values as the target alerts. The priority computation is described in [27]. In this step, we set the threshold $\beta = 0.6$. Alerts with priority scores above β are regarded as important alerts and are selected as target alerts. In this step, we identified 15 hyper alerts whose priority values are above the threshold.

Fourth, **alert correlation**. When applying correlation algorithms, we correlate each target alert with all other non-background alerts (i.e., the background alerts identified by the *Ljung-Box* test are excluded.). As described in Section 3.3, we have two steps in correlating alerts. First, we apply Bayesian-based correlation engine on each target hyper alert and discover its "causal" alerts. Figure 4 shows the resulting correlation graphs. Second, for each target hyper alert, we apply GCT-based correlation algorithm to correlate it with other hyper alerts, which are not its "causal" alerts after running Bayesian correlation mechanism in the first step. The resulting correlation graph is shown in

Figure 5. The dotted line in Figure 4 and Figure 5 represent false positive "causal" relationship. The correlation probability or confidence of each alert-pair is associated with the edge in the correlation graph. In Eq. (4), ω equals 0.3 and t equals 0.6.

Fifth, **attack path analysis**. As discussed in Section 3.3, for any two nodes in the correlation graph that are connected on multiple paths, we can compute the probability of attack transition along each path, then rank and select the one with highest overall value. For example, from node *DB_FTP_Globbing_Attack* to node *DB_NewClient* in the graph shown in Figure 5, there are 6 paths that connect these two nodes. Based on the probability or confidence associated on the edge, we can compute the value of each path and rank the order.

For example, the overall confidence for the attack path *DB_FTP_Globbing_Attack*→ *Loki*→*DB_NewClient* is: $P(DB\_FTP\_Globbing\_Attack, Loki, DB\_NewClient) = P(DB\_FTP\_Globbing\_Attack) * P(Loki|DB\_FTP\_Globbing\_Attack) * P(DB\_NewClient|Loki) = P(DB\_FTP\_Globbing\_Attack) * 0.7 * 0.72 = P(DB\_FTP\_Globbing\_Attack) * 0.5$. Table 2 shows the ordered multi-paths according to the corresponding path values. From the table, we can see that it is more confident to say that the attacker is more likely to launch *FTP Globbing Attack* against the Database Server, then *New Client* attack from the Database Server that denotes a suspicious connection to an external site (e.g., set up a covert channel).

Sixth, **attack strategy analysis**. In this phase, we perform attack strategy analysis by abstracting the scenario graphs. Instead of using hyper alerts representing each node, we use the corresponding attack class (e.g., *DoS* and *Access Violation*) to abstractly present attack strategies. While analyzing attack strategy, we focus on each target and abstract the attacks against the target. Figure 6(a) shows the high-level attack strategy on the Plan Server extracted from attack scenario graphs shown in Figure 5. From Figure 6(a), we can see that the attacker uses a covert channel (indicated by *Connection Violation*) to export data and import malicious code to root the Plan Server. The attacker accesses to the data stored on the Plan Server (indicated by *Access Violation*) to steal the data, then export the information. The activity of *Surveillance* has impacted the server on the performance (indicated by *Asset Distress*). Figure 6(b) shows the attack strategy on the Database Server. It is easy to see that the attacker launches an exploit attack against the Database Server in order to get root access. Then the attacker sets up a covert channel, accesses data and exports the data. The mutual loop pattern between attack class *Connection Violation*, *Access Violation* and *Exfiltration* indicates the attack continuously accesses file, exports data and downloads the malicious code.

4.2 Discussion

Applying our integrated correlation mechanism can discover more attack step relationships than using a single approach. Figure 4 shows that when we apply Bayesian-based approach alone, we can only discover partial attack step relationships. The reason is that the Bayesian-based correlation engine relies on domain knowledge to correlate alerts. Therefore, it is only capable of discovering the direct attack step transitions, e.g., attack *Mail_RootShareMounted* followed by attack *Mail_IllegalFileAccess*. When the alert relationship is new or has not been encoded into the correlation engine, such relationship cannot be detected. Figure 5 shows that we can discover some more attack re-

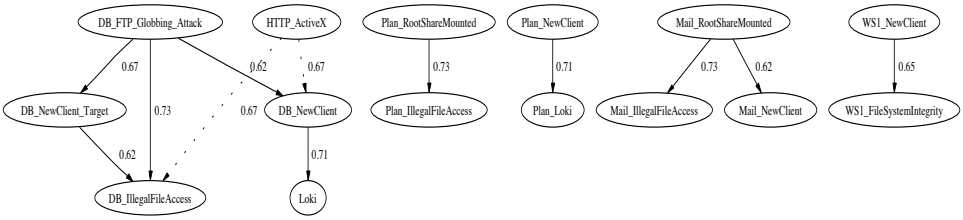


Fig. 4. The GCP scenario I: The correlation graph discovered by Bayesian-based approach.

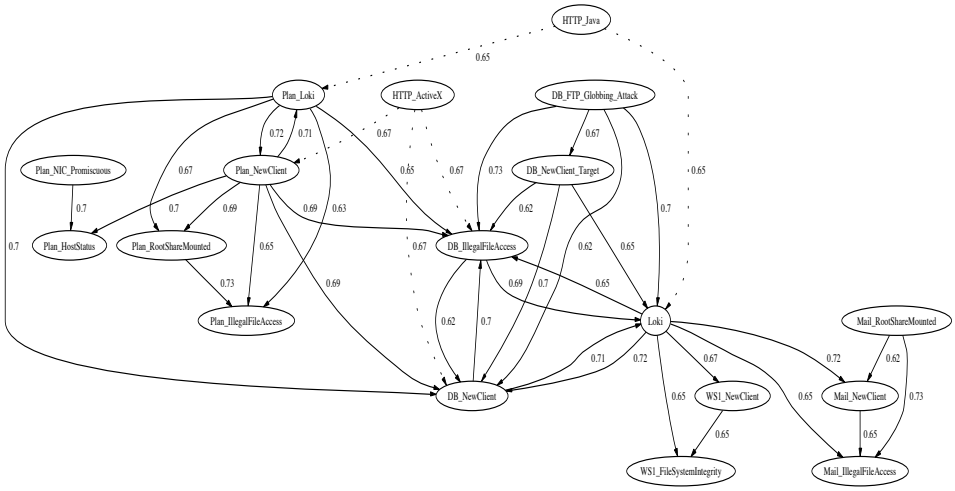
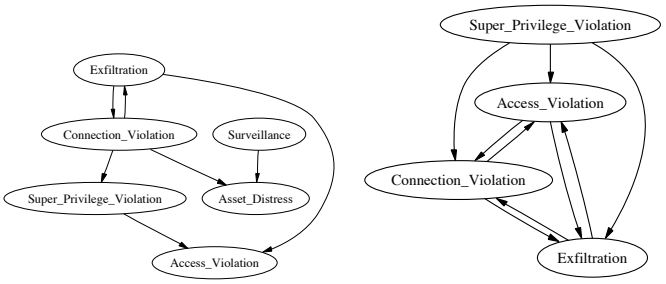


Fig. 5. The GCP scenario I: The correlation graph discovered by the integrated approach.

Order	Nodes Along the Path	Score
Path1	DB FTP Globbing Attack→DB NewClient	P*0.62
Path2	DB FTP Globbing Attack→Loki→DB NewClient	P*0.50
Path3	DB FTP Globbing Attack→DB NewClient Target→DB NewClient	P*0.47
Path4	DB FTP Globbing Attack→DB IllegalFileAccess→DB NewClient	P*0.45
Path5	DB FTP Globbing Attack→DB NewClient Target→Loki →DB NewClient	P*0.31
Path6	DB FTP Globbing Attack→DB NewClient Target→DB IllegalFileAccess →DB NewClient	P*0.23

Table 2. Ranking of paths from node *DB FTP Globbing Attack* to node *DB NewClient*. P denotes $P(DB\ FTP\ Globbing\ Attack)$

lationships using GCT-based correlation method so that we can link the isolated graphs output by Bayesian-correlation engine. The reason is that GCT-based correlation mechanism correlates attack steps based on the temporal and statistical relationship between attack steps, e.g., the loop pattern of attack transitions among attack *DB_NewClient*,



(a) GCP scenario I: attack strategy on Plan Server

(b) GCP scenario I: attack strategy on Database Server

Fig. 6. GCP I: Attack strategy graph

DB_IllegalFileAccess and *Loki*. This correlation engine does not rely on prior knowledge. On the other hand, given GCT-based correlation analysis does not use domain knowledge, it is less accurate than Bayesian-based correlation analysis for the direct attack step transitions. By incorporating the two correlation engines, in this experiment, we can improve the true positive correlation rate from 95.06% (when using GCT-based correlation engine alone) to 97.53%. False positive correlation rate is decreased from 12.6% (when using GCT-based correlation engine alone) to 6.89%.

Our correlation approach can also correlate non-security alerts, e.g., alerts from network management system (NMS), to detect attack strategy. Although NMS alerts cannot directly tell us what attacks are unfolding or what damages have occurred, they can provide us some useful information about the state of system and network health. So we can use them in detecting attack strategy. In this scenario, NMS outputs alert *Plan_Host_Status* indicating that the Plan Server's CPU is overloaded. Applying our GCT-based and Bayesian-based correlation algorithms, we can correlate the alert *Plan_HostStatus* with alert *Plan_NewClient* (i.e., suspicious connection) and *Plan_NIC_Promiscuous* (i.e., traffic surveillance).

We are aware of the limitations of using synthesized data only in our experiments although we believe that the simulation was integrated to be as realistic as possible. The real world will have more complicated and subtle attack strategies with more noisy attacks. We plan to apply our algorithms to real-life data so that we can further improve our work.

5 Conclusion and Future Work

In this paper, we presented an integrated correlation system to analyze INFOSEC alerts and detect novel attack strategies. We develop and integrate two complementary alert correlation mechanisms: (1) correlation based on Bayesian inference with a broad range

of indicators of attack impacts, and (2) correlation based on the Granger Causality Test, a statistical-based correlation algorithm. Our Bayes-based correlation mechanism can discover alerts that have direct "causal" relationships according to domain knowledge. This correlation engine can also relax the strict hard-coded pre-/post-condition matching and handle the partial input evidence. GCT-based correlation engine can discover new attack relationships when attack steps have statistical relationship. Attack scenarios are analyzed by constructing correlation graphs based on the correlation results. A quantitative analysis of attack strategy is conducted using the outputs of our integrated correlation engines. Attack strategies are analyzed using correlation graphs. The results show that our approach can discover novel attack strategies with high accuracy.

We will continue to study alert correlation with a focus on attack plan recognition and prediction. We will also study situation assessment, e.g., damage assessment and situation analysis. We also note the limitation of the synthesized alerts in our experiments. Therefore, we will apply our algorithms to alert streams collected from live networks to improve our work.

6 Acknowledgments

This work is supported in part by NSF grants CCR-0133629 and CCR-0208655 and Army Research Office contract DAAD19-01-1-0610. The contents of this work are solely the responsibility of the authors and do not necessarily represent the official views of NSF and the U.S. Army.

References

1. E. Bauer, D. Koller, and Y. Singer. Update rules for parameter estimation in Bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 3–13, Providence, RI, August 1997.
2. J. B. D. Cabrera, L. Lewis, X. Qin, W. Lee, and R. K. Mehra. Proactive intrusion detection and distributed denial of service attacks - a case study in security management. *Journal of Network and Systems Management*, vol. 10(no. 2), June 2002.
3. J. B. D. Cabrera, L. Lewis, X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran, and R. K. Mehra. Proactive detection of distributed denial of service attacks using mib traffic variables - a feasibility study. In *Proceedings of IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, May 2001.
4. P. E. Caines and C. W. Chan. Feedback between stationary stastic process. *IEEE Transactions on Automatic Control*, 20:495–508, 1975.
5. S. Cheung, U. Lindqvist, and M. W. Fong. Modeling multistep cyber attacks for scenario recognition. In *Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, Washington, D.C., April 2003.
6. I. Cohen, A. Bronstein, and F. G. Cozman. Online learning of bayesian network parameters. *Hewlett Packard Laboratories Technical Report, HPL-2001-55(R.1)*, June 2001.
7. F. Cuppens and A. Miège. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 202–215, Oakland, CA, May 2002.
8. DAPRA Cyber Panel Program. DARPA cyber panel program grand challenge problem (GCP). http://www.grandchallengeproblem.net/, 2003.

9. H. Debar and A. Wespi. The intrusion-detection console correlation mechanism. In *4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, October 2001.
10. C. W. Geib and R. P. Goldman. Plan recognition in intrusion detection system. In *DARPA Information Survivability Conference and Exposition (DISCEX II)*, June 2001.
11. M. R. Gevers and B. D. O. Anderson. Representations of jointly stationary stochastic feedback processes. *International Journal of Control*, 33:777–809, 1981.
12. R. P. Goldman, W. Heimerdinger, and S. A. Harp. Information modleing for intrusion report aggregation. In *DARPA Information Survivability Conference and Exposition (DISCEX II)*, June 2001.
13. C. W. J. Granger. Investigating causal relations by econometric methods and cross-spectral methods. *Econometrica*, 34:424–428, 1969.
14. IETF Intrusion Detection Working Group. Intrusion detection message exchange format. http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-09.txt, 2002.
15. J. Haines, D. K. Ryder, L. Tinnel, and S. Taylor. Validation of sensor alert correlators. *IEEE Security & Privacy Magazine*, January/February, 2003.
16. W. Hesse, E. Moller, M. Arnold, H. Witte, and B. Schack. Investigation of time-variant causal interactions between two eeg signals by means of the adaptive granger causality. *Brain Topography*, 15:265–266, 2003.
17. M. Kaminski, M. Ding, W.A. Truccolo, and S. L. Bressler. Evaluating causal relations in neural systems: Granger causality, direct transfer function (dtf) and statistical assessment of significance. *Biological Cybernetics*, 85:145–157, 2001.
18. R. K. Kaufamnn and D. I. Stern. Evidence for human influence on climate from hemispheric temperature relations. *Nature*, 388:39–44, July 1997.
19. H. Lee, K. S. Lin, and J. Wu. Pitfalls in using granger causality tests to find an engine of growth. *Applied Economics Letters*, 9:411–414, May 2002.
20. G. M. Ljung and G. E. P. Box. On a measure of lack of fit in time series models. In *Biometrika 65*, pages 297–303, 1978.
21. B. Morin and H. Debar. Correlation of intrusion symptoms: an application of chronicles. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, Pittsburgh, PA, September 2003.
22. P. Ning, Y. Cui, and D.S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *9th ACM Conference on Computer and Communications Security*, November 2002.
23. P. Ning and D. Xu. Learnign attack strategies from intrusion alerts. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003.
24. P. Ning, D. Xu, C. G. Healey, and R. A. Amant. Building attack scenarios through integration of complementary alert correlation methods. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS'04)*, San Diego, CA, February 2004.
25. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc, 1988.
26. P. A. Porras, M. W. Fong, and A. Valdes. A Mission-Impact-Based approach to INFOSEC alarm correlation. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*, October 2002.
27. X. Qin and W. Lee. Statistical causality analysis of infosec alert data. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, Pittsburgh, PA, September 2003.
28. S. M. Ross. *Introduction to Probability Models*. Harcourt Academic Press, 7th edition, 2000.
29. W. Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley, 1999.
30. A. Valdes and K. Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, October 2001.