A Probabilistic-Based Framework for INFOSEC Alert Correlation

A Thesis
Presented to
The Academic Faculty

by

Xinzhou Qin

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

College of Computing
Georgia Institute of Technology
August 2005

A Probabilistic-Based Framework for INFOSEC Alert Correlation

Approved by:

Dr. Wenke Lee, Advisor
College of Computing
Georgia Institute of Technology

Dr. Mustaque Ahamad
College of Computing
Georgia Institute of Technology

Dr. Ralph Merkle
College of Computing
Georgia Institute of Technology

Dr. Henry Owen
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Chuanyi Ji
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: July 15, 2005

To my dear family:

Thank you for all of your love, support and encouragements.

ACKNOWLEDGEMENTS

I would like to express my sincere and deep gratitude to my advisor, Dr. Wenke Lee, for his great support, guidance, patience and encouragement during the past several years. Wenke has not only guided and helped me on my research work, but also taught me important values of life. He can always directly point out my weakness that I need to overcome and also always give me cheers when I have achievedf milestones. The thesis would not have been possible without help of Wenke and many other people. I would also like to thank Dr. S. Felix Wu at UC Davis for introducing me to Wenke and for his great help when I had difficulties.

Many thanks to Dr. Mustaque Ahamad, Dr. Ralph Merkle, Dr. Henry Owen and Dr. Chuanyi Ji for being my proposal and dissertation committees and providing insightful suggestions on my research. I would also like to thank Dr. João B. D. Cabrera for his friendship and help on my research work. Also thanks to Dr. Lundy Lewis for his support and encouragement during my Ph.D. studies.

I have also been very fortunate to have a great team at InfoSec Lab at Georgia Tech. Many thanks to our terrific team members, Yi-an Huang, David Dagon, Guofei Gu, Mohamad Kone, Prahlad Fogla, Oleg M. Kolesnikov and Monirul Sharif, for their great help on my research and bringing me the wonderful and enjoyable graduate student life at Tech. I will never forget our discussions on ideas, collaborations on research, travels on conferences and wonderful chats on fun. Special thanks to David Dagon. David is an energetic researcher, also like an elder brother, helping us on everything that he can, patiently, warmly and unselfishly. I believe each of the team members will become a super star in the InfoSec community.

I would also like to thank Dr. Marsha Duro, Dr. Alexander Bronstein and Ms. Julie

Symons for mentoring me during my summer internship at HP Labs.

Many thanks to Dr. Fengmin Gong at McAfee for being my mentor and bringing me to the information security field during my internship at MCNC.

Finally, I would like to dedicate this dissertation to my family: my parents, my wife and my brother. I would never have made it through the whole Ph.D. process without their priceless love, encouragements and support. Special thanks to my wife for her great love, patience and understanding during the past several years. She has shared the hardship that I have endured and enjoyed the success that I have achieved. Thank you, my dear family!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Deploying a large number of information security (INFOSEC) systems can provide in-depth protection for systems and networks. However, the sheer number of security alerts output by security sensors can overwhelm security analysts and keep them from performing effective analysis and initiating timely response. Therefore, it is important to develop an advanced alert correlation system that can reduce alarm redundancy, intelligently correlate security alerts and detect attack strategies. Alert correlation is therefore a core component of a security management system.

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 addition, these approaches focus more on the aggregation and analysis of raw security alerts, and build basic or low-level attack scenarios.

This thesis focuses on discovering novel attack strategies via analysis of security alerts. Our framework helps security administrator aggregate redundant alerts, filter out unrelated attacks, correlate security alerts, analyze attack scenarios and take appropriate actions against forthcoming attacks.

In alert correlation, we have developed an integrated correlation system with three complementary correlation mechanisms based on two hypotheses of attack step relationship. The first hypothesis is that some attack steps are directly related because an earlier attack enables or positively affects the later one. We have developed a probabilistic-based correlation engine that incorporates domain knowledge to correlate alerts with direct causal relationship. The second hypothesis is that some related attack steps, even though they do

not have obvious or direct (or known) relationship in terms of security and performance measures, still exhibit statistical and temporal patterns. For this category of relationship, we have developed two correlation engines to discover attack transition patterns based on statistical analysis and temporal pattern analysis, respectively. Based on the correlation results of these three correlation engines, we construct attack scenarios and conduct attack path analysis. The security analysts are presented with aggregated information on attack strategies from the integrated correlation system.

In attack plan recognition, we address the challenges of identifying attacker's high-level strategies and predicting upcoming attack intentions. We apply graph-based techniques to correlating isolated attack scenarios derived from low-level alert correlation based on their relationship in attack plans. We conduct probabilistic inference to evaluate the likelihood of attack goal(s) and identify potential attacker's intentions based on observed attack activities.

We evaluate our approaches using DARPA's Grand Challenge Problem (GCP) data sets and live data sets collected from our department backbone network. Our evaluation shows that our approach can effectively discover novel attack strategies, provide a quantitative analysis of attack scenarios and identify attack plans.

CHAPTER I

INTRODUCTION AND MOTIVATION

1.1 Intrusion Detection

The information security industry has been very active in recent years. In order to counter security threats to computer systems and networks, many technologies have been developed and applied in security operations, such as firewall, encryption, authentication and access control. However, all these technologies have their own limitations that may allow intruders to break in. An Intrusion Detection System (IDS) is a security mechanism that can intelligently monitor computer systems and networks to detect intrusions in real time, and then respond to attacks quickly and effectively.

"Intrusion detection is the process of identifying and responding to malicious activity targeted at computing and networking resources [2]." Intrusion detection can be *host-based* or *network-based*. Host-based IDS can protect critical network devices storing sensitive and security information. Intrusions are detected by analyzing operating system and application audit trails, e.g., BSM [81]. Network-based IDS monitors all activities over a network connection or segment and performs an analysis of the traffic by using the parameters or rules set up by the experts.

Two main techniques have been applied in IDS. One is *misuse detection*, the other is *anomaly detection*. Misuse detection uses signatures of known attacks, i.e., the patterns of attack behavior or effects, to identify the matched activity as an attack. Anomaly detection uses the established normal profiles to identify any unacceptable deviations as possible results of an attack. Misuse and anomaly detection both have advantages and disadvantages. Misuse detection systems usually have a relative high accuracy in intrusion detection and usually produce only a few false positives. However, these systems can only detect attacks

that have already been modeled. Anomaly detection systems, by contrast, have the advantage of being able to detect new intrusions that have not had any known signatures yet. However, anomaly detection has relatively high false positives because some legitimate behaviors may be regarded as intrusions due to their deviations from normal profiles. Another challenge to anomaly detection system is that it is usually difficult to train a system in a very dynamic environment.

IDS has been an active research area for about two decades. James Anderson published an influential paper [3] in 1980. In 1987, Dorothy Denning provided a methodological framework of IDS in her milestone paper [28]. Since then, both academia and industry have begun to study intrusion detection technologies and develop IDS products. There are several influential research IDSs. For example, EMERALD [68] uses statistical techniques for anomaly detection and expert system rules for misuse detection. STAT [42] uses state transition analysis for anomaly detection. The assumption in STAT is that the unauthorized activity can be reflected by a certain sequence of actions that indicate the system has been moved from an initial authorized state to a compromised state by the intruder. Bro [64] filters network streams into a series of events, and executes scripts that contain site-specific ID rules.

The industry has also been actively developing commercial IDS products, e.g., Net Ranger by Cisco Systems and RealSecure by Internet Security Systems. Recently, many companies have started developing Intrusion Prevention System (IPS) based on IDS techniques with more proactive response capabilities. In addition to traditional IDS or IPS appliance, there has appeared a trend to integrate IDS with other security devices (e.g., Firewall, SSL, VPN) into one security appliance or into a switch or router, e.g., Cisco Systems and Juniper Networks. From the perspective of intrusion detection technique, most IDS or IPS products are built based on signature pattern matching.

1.2 Alert Correlation and Attack Plan Recognition

Information security (INFOSEC) is a complex process with many challenging problems. As more security systems are developed, deploying a large scale of INFOSEC mechanisms, e.g., authentication systems, firewalls, intrusion detection systems (IDSs), antivirus software, network management and monitoring systems, can provide protection in depth for the IT infrastructure. INFOSEC sensors often output a large quantity of low-level or incomplete security alerts because there is a large number of network and system activities being monitored and multiple INFOSEC systems can each report some aspects of security events. The sheer quantity of alerts from these security systems and sensors can overwhelm security administrators and prevent them from performing comprehensive security analysis of the protected domains and initiating timely response.

From a security administrator's point of view, it is important to reduce the redundancy of alarms, intelligently integrate and analyze security alerts, construct attack scenarios (defined as a sequence of related attack steps) and present high-level aggregated information from multiple local-scale events. To address this issue, researchers and security product vendors have proposed *alert correlation*, a process to analyze and correlate security alerts to provide an aggregated information on the networks and systems under protection. Applying alert correlation techniques to identifying attack scenarios can also help forensic analysis, response and recovery, and even prediction of forthcoming attacks. Therefore, alert correlation is a core component in a security management system.

Recently there have been several proposals on alert correlation, including alert similarity measurement [83], probabilistic reasoning [33], clustering algorithms [27], pre- and post-condition matching of known attacks [15, 23, 58], and chronicles formalism approach [57]. Most of these proposed approaches have limited capabilities because they rely on various forms of predefined knowledge of attack conditions and consequences. They cannot recognize a correlation when an attack is new or the relationship between attacks is new. In other words, these approaches in principle are similar to *misuse detection*

techniques, which use the "signatures" of known attacks to perform pattern matching and cannot detect new attacks. It is obvious that the number of possible correlations is very large, potentially a combinatorial of the number of known and new attacks. It is infeasible to know *a priori* and encode all possible matching conditions between attacks. To further complicate the matter, the more dangerous and intelligent adversaries will always invent new attacks and novel attack sequences. Therefore, we must develop significantly better alert correlation algorithms that can discover sophisticated and new attack sequences.

In addition, all these approaches focus on the aggregation and analysis of raw security alerts, and build basic or low-level attack scenarios. However, in practice, an alert correlation system should have a hierarchical architecture. The analysis is conducted from low-level alert correlation to abstract scenario analysis at high levels. In addition, there can exist isolated attack scenarios derived from low-level alert correlation due to various reasons, e.g., IDSs miss detecting critical attacks. Therefore, in addition to the low-level correlation analysis, it is necessary to develop algorithms and tools for security analysts to further analyze and correlate attack scenarios so that they can make situation and mission assessment accurately, and take appropriate responses to minimize the damages. In addition, threat analysis and attack prediction are also helpful and important for security operators to take actions in advance to avoid potential attacks and damages.

Recognizing attack plans is one of the goals of security analysts. Plan recognition has been a research area in artificial intelligence (AI) for decades. In AI, plan recognition is a process of inferring the goals of an agent from observations of the agent's activities. Plan recognition can be characterized as *keyhole recognition* and *intended recognition* based on the role of an agent whose plan is being inferred [19]. In *keyhole recognition*, the agent is not aware that its action is being observed, i.e., the agent is only engaged in the task and does not attempt to impact the recognition process. In *intended recognition*, the agent attempts to perform actions that can aid the recognition of its plan, e.g., a language understanding system [19].

Unfortunately, traditional plan recognition techniques cannot be applied to attack plan recognition. Unlike the traditional agent that either *aids* the recognition of its plan or *does not attempt to impact* the recognition of the process, attackers can perform activities to escape detection and avoid the recognition of their attack strategies. Therefore, this type of recognition process can be categorized as *adversary recognition* that is more challenging and more uncertain in the recognition process. In addition, some assumptions of traditional plan recognition techniques are not valid anymore. First, in plan recognition, there is always an assumption that there exists a valid plan in the plan library that the agent can reach. In network security, we cannot assume that we have a complete attack plan library that includes all the possible strategies of attackers. Therefore, we have to deal with the case that the observed attacker's activity is beyond or partially matched with our pre-defined attack plans. Second, plan recognition assumes a complete, ordered set of tasks for a plan. However, we cannot always observe all of the attacker's activities, and often can only detect incomplete attack steps due to the limitation or deployment of security sensors. Therefore, the attack plan recognition system should address partial order and unobserved activities.

1.3 An Overview of Our Alert Correlation System

The main objective of the correlation process is to provide an aggregated information on the security-related activities on the network under protection. In this section, we present an overview of our alert correlation process. This process consists of a collection of components that focus on different aspects of the overall correlation task and transform sensor alerts into aggregated intrusion reports.

Figure 1 shows the graphical representation of the framework of our alert correlation process. The core of the correlation process consists of several components including alert aggregation and clustering, alert verification and prioritization, alert correlation, attack plan recognition and intention prediction.

Alert aggregation and clustering reduces the redundancy of raw security alerts. This

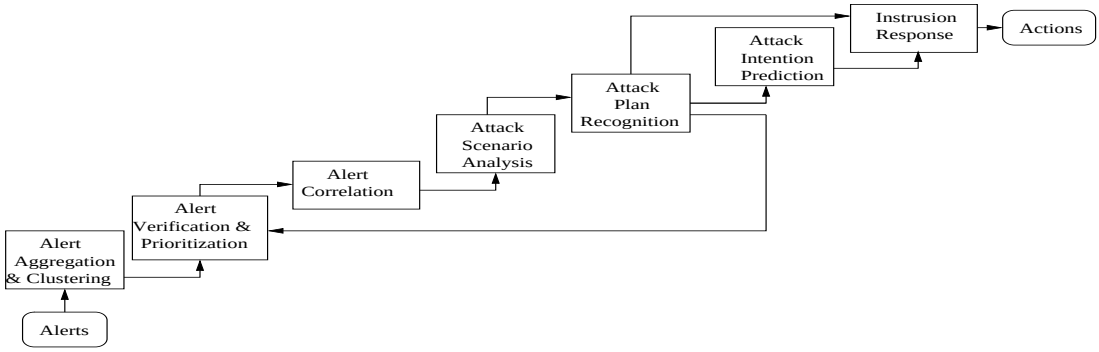


Figure 1: Framework of Alert Correlation Process

component combines alerts corresponding to the same attack instance detected by the same intrusion detection system, as well as alerts corresponding to the same attack detected by different intrusion detection systems.

Alert verification and prioritization verifies and prioritizes each alert based on its success and relevance to the mission goals as well as the severity assessed by security analysts. The verification component filters out false positive alert cross checked by multiple security sensors. The prioritization component computes an appropriate priority value to each alert. This priority information is important for security analysts to quickly discard information that is irrelevant or of less importance to a particular site.

Alert correlation discovers the relationship among attacks or attack steps in a coordinated attack, and constructs attack scenarios. Our alert correlation mechanism integrates three different correlation methods based on two hypotheses of attack step relationships to discover and analyze relationships among alerts. *Bayesian-based correlation engine* [71] applies probabilistic reasoning to correlate alerts that have direct causal relationships according to some domain knowledge. This correlation mechanism is based on the hypothesis that some attack steps have direct relationship because prior attack step enables the later one. *Causal discovery theory-based correlation mechanism* [72] performs alert correlation using statistical analysis of attack occurrences to identify the dependency between alerts. *Time series-based* correlation engine [69] conducts alert correlation using statistical test and investigating temporal relationship between alerts. These two statistical and

temporal-based correlation mechanisms are based on the hypothesis that some attack steps have temporal or statistical patterns even though they may not have direct or obvious (or known) relationships in terms of security or performance measures. We integrate results of these three correlation engines to construct attack scenarios and detect attack strategy. The result of alert correlation is a set of candidate attack plans corresponding to the intrusions executed by the attacker. The outputs of this phase can be used for further analysis in the later phase, i.e., *attack plan recognition* [70].

Attack scenario analysis analyzes attack scenarios resulted from prior alert correlation. In this step, we provide an approach to quantitatively analyze and rank various attack paths and inform security analysts of the ones with the highest likelihood.

Attack plan recognition and intention prediction identifies the attacker's intentions by analyzing and correlating the candidate attack plans or scenarios output by the prior alert correlation phases. In this phase, security analysts conduct situation assessment and threat analysis based on the scenario correlation. This phase should provide a global analysis on the attacker's activities of past, current and future (i.e., intention prediction). Security analysts depend on the results of this phase to take appropriate countermeasures and actions to protect systems and networks, and prevent further attacks.

1.4 Dissertation Overview

In this section, we define our thesis and highlight the contributions presented in this dissertation.

1.4.1 Problem Statement and Formalization

This thesis studies the problem of

How to effectively analyze and correlate security alerts to identify novel attack strategies and plans.

In the process of attack strategy identification, alert correlation is an important and

major analysis component. According to our knowledge and understanding of the problem, we have identified the properties of causal alert pairs and the problem space of alert correlation.

Definition 1 *Alert correlation is a process to identify the causal relationship between alerts.*

More formally, we denote a set of security alerts as $A = \{a_1, a_2, ..., a_i, ..., a_n\}$, in which each alert $a_i \in A$ is associated with a time stamp t_{a_i}.

In addition to time information, each alert has a number of other attributes, such as *source IP*, *destination IP*, *port(s)*, *user name*, *process name*, *attack class*, and *sensor ID*, which are defined in a standard document, "Intrusion Detection Message Exchange Format (IDMEF)", drafted by the IETF Intrusion Detection Working Group [35]. Alert correlation is to investigate if any two alerts a_i and a_j has a causal relationship.

When two alerts have a causal relationship, e.g., a_i causes a_j (denoted as $a_i \rightarrow a_j$), they have all the following three properties.

- First, they have a cause-effect relationship. It means that the causal attack (as represented by a_i) has a positive impact or make a preparation for the attacker to launch a follow-up attack (as represented by a_j). More formally, for an attack A (represented by alert a), we denote $P(a)$ as attack A's *prerequisite* or *pre-conditions* (e.g., existence of vulnerable services), and $C(a)$ as attack A's *consequences* (e.g., getting root privilege). The cause-effect or *direct causal relationship* between a causal attack A_i and an effect attack A_j can be interpreted as A_i's consequences (i.e., $C(a_i)$) has contributed to the prerequisite of the later attack A_j (i.e., $P(a_j)$). Such causal relationship can be represented in terms of security and performance measures.

- Second, they have a sequential relationship. It implies a time constraint between a causal alert and an effect alert. A causal alert should appear prior to an effect

alert, i.e., $t_{a_i} < t_{a_j}$. A sequence of causally related alerts can be constructed from the cause-effect alert pairs. For example, given three causally related alert pairs, $a_i \rightarrow a_j$, $a_j \rightarrow a_m$ and $a_m \rightarrow a_n$, we can have the sequence of causally related alert stream as $a_i \rightarrow a_j \rightarrow a_m \rightarrow a_n$. The associated time constraint of the sequence is that $t_{a_i} < t_{a_j} < t_{a_m} < t_{a_n}$.

- Third, there exists a high *statistical one-way dependence* from the effect alert (a_j) to the causal alert (a_i), i.e., $P(a_j|a_i) > \theta$, where, θ ($0 \leq \theta \leq 1$) is a dependence threshold. In other words, the probability that an effect alert occurs whenever a cause alert occurs is high.

  In addition to the above three properties, in our security operations and data analysis, we have noticed that attack step transition can also have a temporal pattern in the time intervals of attack steps. For example, when an attacker runs a script to launch a series of attacks, the time interval between attack steps is relatively stable. Although such temporal pattern of time intervals does not necessarily exist in all causally related alerts (i.e., it is not an inherent property of cause-effect alerts), incorporating it into the correlation analysis is helpful for us to discover and improve the correlation accuracy when alerts under the correlation have a temporal pattern.

  More formally, we use Δt_{ij} as the time interval between a cause alert (a_i) and an effect alert (a_j), i.e., $\Delta t_{ij} =\mid t_{a_i} - t_{a_j} \mid$. When the causally related alert pairs have occurred multiple k times, we denote ΔT_{ij} to represent the corresponding time interval set, i.e., $\Delta T_{ij} = \{\Delta t_{ij}^1, \Delta t_{ij}^2, ..., \Delta t_{ij}^k\}$.

  According to the variation of ΔT_{ij}, we have the following definition of pairwise temporal patterns.

  Definition 2 *Two causally related alerts ($a_i \rightarrow a_j$) have a strong temporal pattern if their time intervals have relatively stable values. In other words, the variance of*

their time lags has a small value, i.e., $Var(\Delta T_{ij}) \leq \epsilon$*, where,* ϵ *is a small positive value.*

Two causally related alerts ($a_i \rightarrow a_j$*) have a loose temporal pattern if the variance of their time intervals has a large value, i.e.,* $Var(\Delta T_{ij}) > \epsilon$*.*

Intuitively, the *loose temporal pattern* characterizes the situation that alert a_i leads to a_j, but the time is not precise. The *strong temporal pattern* specifies that alert a_j happens after a_i with a relatively stable time distance.

Having stated the properties of causally related alerts, we can define the problem space of alert correlation as follows.

- Alert pairs have *direct causal relationship*. As defined before, *direct causal relationship* means a prior attack makes preparation for a later one. The study of this problem subspace is to find a way to represent the prerequisite and consequence of each attack and evaluate such preparation-for relationship using security and performance measures. It is also needed to evaluate the properties of *sequential time constraints* and *statistical one-way dependence* between alert pairs under analysis.

- Alert pairs have *no known direct causal relationship* in terms of security and performance measures. In this problem subspace, we aim to correlate alerts with strong statistical dependence and temporal patterns found in alert data. The study of this problem subspace is to design algorithms to perform statistical and temporal correlation.

  From a statistical analysis point of view, statistical dependence between two alerts (e.g., a_i and a_j) can be either *one-way dependence* or *mutual dependence*. A strong *one-way dependence* from a_j to a_i implies that whenever a_i occurs, a_j occurs (denoted as $a_i \rightarrow a_j$), i.e., $P(a_j|a_i) > \theta$, where, θ ($0 \leq \theta \leq 1$) is a dependence threshold. Alerts a_i and a_j have strong *mutual dependence* (denoted as $a_i \rightleftarrows a_j$) if

$P(a_j|a_i) > \theta$ and $P(a_i|a_j) > \theta$, which means whenever a_i occurs, a_j occurs and vice versa.

As described before, the goal of alert correlation is to construct the attack scenario in order to identify attack strategies. An attack scenario is usually represented by a directed graph to represent the dependency or correlation among attack steps. We denote such directed graph as an alert correlation graph. Dependencies among attack steps represented in an alert correlation graph can have two structure forms. The first one is a non-loop one-way dependency structure, e.g., $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4$. The second one has a loop-structured dependency form that can be composed by either a series of one-way dependence that form a loop (e.g., $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow a_1$) or mutual dependence (e.g., $a_1 \rightleftarrows a_2$). The loop dependency can be formed, for example, in a series of attack steps that has a repeat pattern.

Based on the structure of dependency among alerts (i.e., loop-structured or non-loop structured) and the temporal pattern of time intervals between the correlated alerts, we can further classify this problem space to four subspaces as follows.

- Dependency among alerts has a *non-loop structure*.
  - The alert pair has a *strong temporal* pattern.
  - The alert pair has a *loose temporal* pattern.
- Dependency among alerts has a *loop structure*.
  - The alert pair has a *strong temporal* pattern.
  - The alert pair has a *loose temporal* pattern.

Figure 2 shows the classification of the problem space of alert correlation. We use the direct or indirect causal relationship between alerts as a criteria to divide the problem space to two sub-spaces.

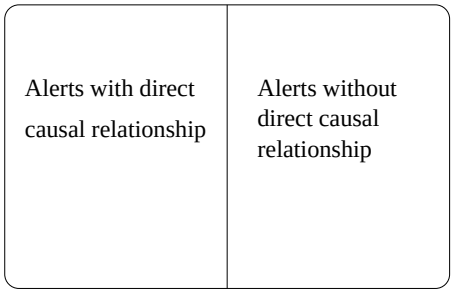


Figure 2: Classification of the problem space of alert correlation

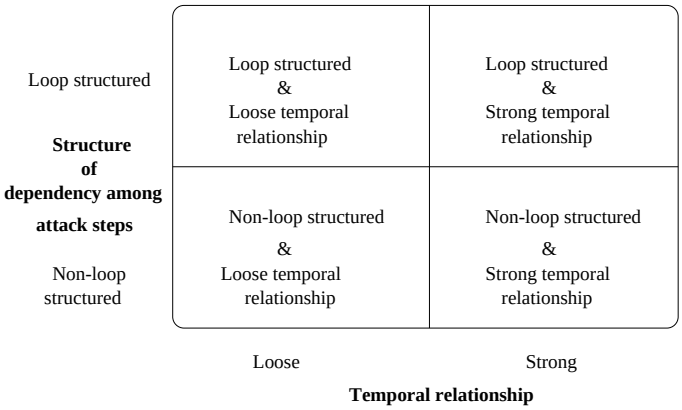


Figure 3: Granular classification of the problem subspace: alert pair without direct causal relationship.

Figure 3 shows the granular classification of the problem subspace in which alert pairs do not have direct causal relationship. In this problem subspace, we use the structure of dependency among attack steps and the strength of temporal pattern between alert pairs as criteria to further identify four problem sub-spaces.

We have studied and developed alert correlation algorithms based on the above problem space definition and classification.

This thesis proposes a framework of alert correlation that helps security administrator aggregate alerts output by security sensors, filter out spurious or incorrect alerts, analyze attack scenarios and take proactive actions against forthcoming attacks. The proposed research includes an integrated correlation system, an attack plan recognition and intention prediction system. There are four major contributions expected in the proposed framework.

1.4.2 Thesis Contributions

This thesis research contributes to the intrusion detection and security management fields in the following areas:

Knowledge-based Probabilistic Correlation Model. In this research, we study how to incorporate domain knowledge into security alert analysis and correlation to discover alert pairs that have *direct causal relationship*, i.e., 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. In particular, we have developed a Bayesian-based correlation mechanism [71] to correlate alerts and identify the causally related alerts if they conform to the three properties of cause-effect alerts as described in Section 1.4.1. Specifically, this correlation engine uses predicates to represent attack prerequisite and consequence, and applies probabilistic reasoning to evaluating the *preparation-for relationship* between alerts based on security states of systems and networks. We also apply time constraints to testing if causally related alert pair candidates conform to the property of *sequential relationship*. In the process of probabilistic correlation, we evaluate the property of *statistical one-way dependence* between alerts being correlated using an attack transition table that shows the likelihood of attack step transitions from different attack classes. We pre-defined the attack transition table based on our domain knowledge, experiment evaluation and related work. Our approach does not rely on the strict pre-/post-condition matching and can also function on the partial correlation evidence.

Statistical and Temporal-based Alert Correlation Models. Besides the knowledge-based correlation model that identifies alert pairs with direct causal relationship, we have developed two statistical and temporal-based correlation models to discover *novel* and *new* attack transition patterns. The development of these two correlation techniques is based on the hypothesis that attack steps can still exhibit statistical dependency patterns (i.e., the third property of cause-effect alerts) or temporal patterns even though they do not have an obvious or *known* preparation-for relationship (i.e., the first property of cause-effect

alerts) in terms of security and performance measures. Therefore, these two correlation engines aim to discover correlated alerts based on statistical dependency analysis and temporal pattern analysis with sequential time constraints (i.e., to ensure the conformity to the second property of cause-effect alerts). More formally, these two engines actually perform *correlation analysis* instead of a direct causality analysis because the preparation-for relationship between alerts are either indirect or unknown. In theory, causality is a subset of correlation [38], which means that a causally related alert pair is also correlated, however, the reverse statement is not necessarily true. Therefore, the correlation output is actually a super set of correlated alerts that can include the causally related alert pairs as well as some correlated but non-causally related alerts. Our goal is to apply these two correlation engines to identifying the correlated alerts that have strong statistical dependencies and temporal patterns, and also conform to the sequential time constraint property. We present these correlated alert candidates to the security analysts for further analysis. In the correlation process, these two correlation mechanisms do not rely on prior knowledge of attack causal relationship.

- **Causal Discovery Theory-based Alert Correlation Model**. We have developed an alert correlation engine based on causal discovery theory [72]. This correlation engine aims to discover the strong statistical dependence among alerts. Since this correlation engine is based on the assumption that causality among variables can be represented by a causal Bayesian network (represented by a directed acyclic graph (DAG) in which there is no directed cycle path), this correlation mechanism can discover the dependency among attack steps that has a non-loop structure. Since this model only applies and depends on probability theory and computation to investigate and identify the statistical one-way dependence, and the temporal pattern of time interval between alerts are not involved in the correlation process, therefore, this model covers the problem subspaces of {*Non-loop structured, Loose temporal relationship*} and {*Non-loop structured, Strong temporal relationship*} as shown in Figure 3.

- **Granger-Causality-based Alert Correlation Model**. In addition to developing an correlation engine to discover the statistical one-way dependency (i.e., no dependency loop), we have studied and developed another statistical and temporal-based correlation mechanisms using Granger-Causality Test analysis [69]. This correlation engine investigates and tests the statistical dependency and temporal patterns of alert pairs to identify attack relationship. This correlation engine aims to discover the statistical dependency (including mutual dependence and one-way dependence that forms a dependency loop in the attack step dependency graph), i.e., the problem subspace of {*Loop structured, Strong temporal relationship*} as shown in Figure 3. In addition, since this correlation engine performs pairwise correlation, it can also complement the causal discovery theory-based correlation engine in the problem subspace of {*Non-loop structured, Strong temporal relationship*}, as shown in Figure 3, to identify the non-loop dependency pattern missed by causal discovery-based correlation engine.

System Integration and Attack Strategy Analysis. We integrate three complementary correlation engines to perform alert analysis and correlation. We construct attack scenarios and conduct attack path analysis based on the output of three correlation engines. We evaluate and rank the overall likelihood of various attack paths and identify those with higher probabilities.

Attack Plan Recognition and Intention Prediction. We have developed techniques applied to attack plan recognition and intention prediction [70]. We have developed a series of techniques to solve three problems. First, we consider how to correlate isolated attack scenarios derived from low-level alert correlation. Second, we address how to recognize the attacker's attack plan and intentions. Third, we discuss how to make predictions of potential attacker's intentions based on current observations and analysis. In our approach, we apply graph-based techniques to correlating isolated attack scenarios and identifying their relationship. Based on the correlation results, we further apply probabilistic reasoning

technique to recognizing the attack plans and evaluating the likelihood of potential attack intentions.

1.4.3 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 reviews the related research work on alert correlation and plan recognition. Chapter 3 describes two components of our correlation system, i.e., alert aggregation and prioritization. Chapter 4 describes our probabilistic-based correlation mechanism. Chapter 5 introduces an approach to alert correlation based on statistical analysis. Chapter 6 describes our correlation approach based on temporal analysis of security alerts. Chapter 7 discusses the integration strategy of three correlation engines. Chapter 8 introduces our attack plan recognition techniques. Chapter 9 reports the experimental performance. Finally, Chapter 10 summarizes the thesis and outlines ideas for future work.

CHAPTER II

RELATED WORK

2.1 Alert Correlation in Security Management System

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

Valdes and Skinner [83] proposed probabilistic-based approach to correlate security alerts by measuring and evaluating the similarities of alert attributes. In particular, the correlation process includes two phases. The first phase aggregates low-level events using the concept of attack threads. The second phase uses a similarity metric to fuse alerts into meta-alerts to provide a higher-level view of the security state of the system. Alert aggregation and scenario construction are conducted by enhancing or relaxing the similarity requirements in some attribute fields.

Porras et al. designed a "mission-impact-based" correlation system with a focus on the attack impacts on the protected domains [67]. The work is an extension to the prior system proposed in [83]. 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.

Some correlation research work are based on pre-defined attack scenarios or association between mission goals and security events. Goldman et al. [33] built a correlation system based on Bayesian reasoning. The system predefines the causal relationship between mission goals and corresponding security events as a knowledge base. The inference engine relies on the causal relationship library to investigate security alerts and perform alert correlation.

Debar and Wespi [27] applied backward and forward reasoning techniques to correlate

alerts. Two alert relationships were defined, i.e., *duplicate* and *consequence*. In a correlation process, backward-reasoning looks for *duplicates* of an alert, and forward-reasoning determines if there are any *consequences* of an alert. They used clustering algorithms to detect attack scenarios and situations. This approach pre-defines consequences of attacks in a configuration file.

Other researchers use attack modeling languages to describe attack transition patterns or attack scenarios to perform alert correlation. Templeton and Levitt [82] defined an attack description language JIGSAW to model computer attacks. In JIGSAW, *capabilities* and *concepts* are used to represent attack conditions. In particular, capabilities are used to describe prerequisite information that the attacker needs to know in order to take an effective attack. Concepts are used to model components or fragments of complex attacks. The attack prerequisites and impacts are expressed in terms of capabilities. A complex attack scenario can be identified by composing the capability provided by one concept with the capability required by another one. In fact, this is an example of pre- and post-condition matching approach to detect attack scenarios based on knowledge modeling. However, other than the capability of detecting multi-step attacks, JIGSAW does not provide support for functionalities of other correlation components.

Krügel et al. [51] proposed a distributed pattern matching scheme based on an attack specification language that describes various attack scenario patterns. Alert analysis and correlation are based on the pattern matching scheme.

Morin and Debar [57] applied *chronicles formalism* to aggregating and correlating alerts. Chronicles provide a high level language to describe the attack scenarios based on time information. Chronicles formalism approach has been used in many areas to monitor dynamic systems. The approach performs attack scenario pattern recognition based on *known* malicious event sequences. Therefore, this approach is analogous to *misuse intrusion detection*.

Ning et al. [58], Cuppens and Miège [23] and Cheung et al. [15] built alert correlation

systems based on matching the pre- and 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. In addition to the alert pre- and post-condition matching, the approach in [23] also has a number of phases including alert clustering, alert merging, and intention recognition. In the first two phases, alerts are clustered and merged using a similarity function. The intention recognition phase is referenced in their model, but has not been implemented. Having the correlation result, the approach in [58] further builds correlation graphs based on correlated alert pairs [58]. Recently, Ning et al. [60] have extended the pre- and post-condition-based correlation technique to correlate some isolated attack scenarios by hypothesizing missed attack steps.

2.2 Alert Correlation in Network Management System

In the field of network management, alert or event correlation has been an active research topic and a subject of numerous scientific publications for over 10 years. The objective of alert correlation in a network management system (NMS) is to localize the faults occurred in communication systems. The problem of alert correlation in NMS is also referred as *root cause analysis*. During the past 10 more years, many solutions have been proposed that derive from different areas of computer science including artificial intelligence (AI), graph theory, neural networks, information theory, and automata theory. In this section, we introduce several well-known approaches that have been implemented in real network management systems to analyze and correlate alerts.

Case-based systems base their decisions on experience and past situations [53]. In a case-based reasoning system, successful solutions to prior alert correlation cases are stored in a knowledge base, called *case base*. When a new problem brings up, the system searches the case base for similar problems. When similar cases are retrieved, the various solutions to the prior cases must be adapted to the case at hand. If this adapted solution successfully

solves the problem, the present problem is added to the case base with the adapted solution for future use.

Model-based approaches incorporate deep knowledge in the form of a model of the underlying system. Model-based systems reason using explicit representation of the system being diagnosed. Deep knowledge of the system may describe its structure (static knowledge) and function (dynamic knowledge). The model-based approaches differ with each other in terms of technologies used to define the system model. Model-based reasoning systems usually use rules to represent heuristic knowledge of the communication networks for system diagnosis. In [61,62], the correlation system performs alert analysis and correlation based on pre-defined causal relationship trees. The correlation tree shows the cause-effect relationship between events. A rule language was developed to express the correlation trees and used to correlate alerts. Another example of model-based correlation system is IMPACT [43,44]. IMAPCT is an expert system specifically designed for event correlation in telecommunications networks. The system uses rule-based reasoning as its inference engine and an object-oriented frame hierarchy as a general knowledge representation.

Code book-based technique [50] applies information-theory to the process of fault localization. It uses a code book to represent the causal relationship between every possible fault and its corresponding symptoms (i.e., alerts). In this approach, event propagation is conceptually regarded as data transmission over a channel. A set of optimal codes is input and the output alphabet is a set of all possible symptom combinations [50]. Therefore, event correlation is equivalent to decoding a received output symbol to one of the valid input symbols. Spurious and lost symptoms are analogous to channel errors. The number of errors that may be detected or corrected depends on the code book and the decoding scheme. Extensions to this correlation technique have been proposed [55, 56] to identify several simultaneous root causes and improve the overall performance.

2.3 Problems in Current Alert Correlation Systems

Alert correlation is a challenging task in security management. A correlation system should effectively reduce the alert redundancy, intelligently analyze alerts and correctly identify the attack strategies.

Most of the proposed approaches have limited capabilities because they rely on various forms of predefined knowledge of attacks or attack transition patterns using attack modeling language or pre- and post-conditions of individual attacks. Therefore, those approaches cannot recognize a correlation when an attack is new or the relationship between attacks is new. In other words, these approaches in principle are similar to misuse detection techniques, which use the signatures of known attacks to perform pattern matching and cannot detect new attacks. It is obvious that the number of possible correlations is very large, potentially a combinatorial of the number of known and new attacks. It is infeasible to know *a priori* and encode all possible matching conditions between attacks. In practice, the more dangerous and intelligent adversaries will always invent new attacks and novel attack sequences. Therefore, we must develop significantly better alert correlation algorithms that can discover sophisticated and new attack sequences.

In the network management system (NMS), most event correlation techniques also depend on various knowledge of underlying networks and the relationship among faults and corresponding alerts. In addition, in an NMS, event correlation focuses more on alerts resulted from network faults that often have fixed patterns. Therefore, modeling-based or rule-based techniques are mostly applied in various correlation systems. Whereas in security, alerts are more diverse and unpredictable because the attackers are intelligent and can use flexible strategies. Therefore, it is difficult to apply correlation techniques developed in network management system to the analysis of security alerts.

2.4 Plan Recognition

In artificial intelligence (AI), plan recognition has been an active research area. Different types of inference techniques have been applied to plan recognition, e.g., *deduction* and *abduction*. In particular, the earliest work in plan recognition was rule-based inference system [74, 84]. A milestone work of plan recognition was done by Kautz and Allen in 1986 [49]. In [49], they defined the problem of plan recognition as finding a minimal set of top-level actions (i.e., plan goals) that were sufficient to explain the observed actions. The inference was conducted by going through the rule sets. Charniak and McDemott [13] proposed that the plan recognition problem can be solved by *abduction*, or reasoning to the best explanation. Charniak and Goldman [11, 12] applied Bayesian networks to plan recognition. Carberry [10] applied Dempster-Shafer theory [76] to computing the combined support by multiple evidences to hypotheses plans. Albrecht et al. [1] proposed to construct a plan recognition inference system based on *Dynamic Belief Networks* [26]. In Dynamic Belief Networks, the influence of temporal aspects is represented by multiple nodes to indicate the status of a variable at different instances of time.

There are some challenges in applying traditional plan recognition techniques to security applications. First, traditional plan recognition techniques are usually applied in non-adversary situation. The recognition process can be either aided or non-interfered by the agent being observed. However, in the security application, the plan recognition process is an *adversary recognition* where attackers are trying to avoid or interfere with any recognition process on their intrusion activities.

Second, the assumptions used in traditional plan recognition are not valid in adversary recognition anymore. For example, in non-adversary plan recognition, a single agent and a single plan have to be determined. The observed activities are conducted by a single agent toward a single plan. Although there are some works on multi-agent plan recognition, they also share that assumption. In attack plan recognition, by contrast, it is possible that an attacker has multiple dynamic attack plans. There also exist coordinated attacks

conducted by multiple attackers. In addition, in non-adversary plan recognition, there is a complete, ordered and correct set of activities. The observations available are correct and corresponding to a determined plan. Every action that is performed is observed. In adversary plan recognition, this assumption is not valid anymore.

The most related work to ours is [31] in which Geib and Goldman applied probabilistic reasoning to recognizing the attacker's intentions. The approach conducts the plan recognition from raw security alerts. The plan library is defined by detailed specific attacks. This definition method has the limitation that it can increase the computation complexity of inference. In addition, it also requires a complete and ordered attack sequence (if there are missing attack steps, it inserts hypothesized attack steps in order to have a complete activity sequence) when conducting the plan recognition.

2.5 Our Approach

Our approach aims to address the challenge of how to detect *novel* attack strategies that can consist of a series of unknown patterns of attack transitions. In alert correlation techniques, our approach differs from other work in the following aspects. Our approach integrates three complementary correlation engines to discover attack scenario patterns. It includes both knowledge-based correlation mechanisms and statistical and temporal-based correlation methods.

We apply a Bayesian-based correlation engine to the attack steps that are directly related, e.g., a 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 inference. The correlation engine performs the inference about the correlation based on broad indicators of attack impacts without using the strict hard-coded pre-/post-condition matching.

In addition to domain knowledge-based correlation engine, we have developed two statistical and temporal-based correlation engines. The first one applies causal discovery

theory to alert analysis and correlation. This approach identifies alert relationship based on statistical analysis of attack dependence. Having observed that many attack steps in a complicated attack strategy often have a strong temporal relationship, we have developed a correlation engine using temporal analysis. In particular, we applied Granger-Causality Test technique to discovering attack steps that have strong temporal and statistical patterns.

These two statistical and temporal-based correlation techniques differ from other related work in that they do not rely on prior knowledge of attack strategies or pre- and post-conditions of individual attacks. Therefore, these two statistical and temporal-based approaches can be used to discover *new* attack strategies that can have unknown attack transition patterns. To the best of our knowledge, our approach is the first approach that detects new attack strategies without relying on pre-defined knowledge base.

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.

In attack plan recognition, our approach is unique in the following aspects. First, we build our plan recognition system after a low-level alert correlation step that includes alert aggregation, alert prioritization and alert correlation. The advantage of this approach is that it can reduce the computation complexity when performing the high-level attack scenario correlation and probabilistic inference. Second, we do not require a complete ordered alert sequence for inference. We have the capability of handling partial order and unobserved activity evidence sets. In practice, we cannot always observe all of the attacker's activities, and can often only detect partial order of attack steps due to the limitation or deployment of security sensors. For example, security sensors such as IDSs can miss detecting intrusions and thus result in an incomplete alert stream. Third, we provide an approach to predict potential attacks based on observed intrusion evidence.

CHAPTER III

ALERT AGGREGATION AND PRIORITIZATION

In this chapter, we describe two major components in our alert correlation system, i.e., alert aggregation and alert prioritization.

3.1 Alert Aggregation and Clustering

One of the issues with deploying multiple security devices is the large number of alerts output by the devices. The large volume of alerts make it very difficult for the security administrator to analyze attack events and handle alerts in a timely fashion. Therefore, the first step in alert analysis is alert aggregation and volume reduction.

In our approach, we use alert fusion and clustering techniques to reduce the redundancy of alerts while keeping the important information. Specifically, each alert has a number of attributes such as *time stamp*, *source IP*, *destination IP*, *port(s)*, *user name*, *process name*, *attack class*, and *sensor ID*, which are defined in a standard document named "Intrusion Detection Message Exchange Format (IDMEF)" drafted by IETF Intrusion Detection Working Group [35].

IDMEF has defined alert formats and attributes. IDMEF is intended to be a standard data format that intrusion detection systems can use to report alerts about suspicious events. A Document Type Definition (DTD) has been proposed to describe IDMEF data format by XML documents.

In IDMEF, three temporal attributes have been defined to be associated to an alert. *detect-time* refers to the time that the attack occurs, *create-time* represents the time when the attack is detected and *analyzer-time* is the time when the alert is output by an IDS. Create-time and analyzer-time are fully dependant on the characteristics of the IDS. Therefore,

we use *detect-time* attributes in our alert aggregation process. In other words, two alerts might be considered similar even though their create-time and analyzer-time are completely different.

In the IDMEF format, the structures of attributes source and target are similar. They can be described by a node, a user, a process and a service. A node might be identified by its IP address (typically by a network-based IDS) or by its host name (typically by a host-based IDS). Similarly, some IDSs provide service names or port numbers. We create and use two correspondence tables between host names and IP addresses, and between services and port numbers. For most alerts output by a host-based IDS, we specify that a similarity exists between alerts' source and target attributes if both their nodes, users, services and processes are similar. And for most network attacks, we compare the nodes and services.

Alert fusion has two phases, i.e., aggregation of alerts of the same IDS and aggregation of alerts of different sensors. Specifically, we first combine alerts that have the same attributes except time stamps. This step is intended to aggregate alerts that are output by the same IDS and are corresponding to the same attack but have a small delay, i.e., the time stamps of those alerts can be slightly different, e.g., two seconds apart. Second, based on the results of step 1, we aggregate alerts with the same attributes but are reported from different heterogeneous sensors. The alerts varied on time stamp are fused together if they are close enough to fall in a pre-defined time window.

Alert clustering is used to further group alerts after alert fusion. Based on various clustering algorithms, we can group alerts in different ways according to the *similarity* among alerts, (e.g., [83] and [46]). Currently, based on the results of alert fusion, we further group alerts that have same attributes except time stamps into one cluster. After this step, we have further reduced the redundancy of alerts.

Definition 3 *A* hyper alert *is defined as a time ordered sequence of alerts that belong to the same cluster.*

For example, after alert clustering, we have a series of aggregated alert instances,

$a_1, a_2...a_n$, in one cluster that have the same attributes along the time axis. We use hyper alert A to represent this sequence of alerts, i.e., $A = \{a_1, a_2, ..., a_n\}$.

3.2 Alert Verification and Prioritization

The next phase of alert processing is to verify and prioritize each hyper alert based on its success and relevance to the mission goals.

When a correlation engine receives false positives as input, the quality of correlation results can degrade significantly. Therefore, the reduction of false positive and irrelevant alerts is an important prerequisite to achieve a good correlation results.

The task of alert verification is to examine an alert and determine the success or failure of the corresponding attack. It aims to filter out the false positive alerts output by security sensors.

We apply evidence cross checking to identifying the false positive alert. In other words, we use alerts or evidence output by other security sensors to cross check the validity of an alert. In particular, for an alert generated by a security sensor (e.g., an IDS), we check if there are any similar alerts output by other security sensors or if there are any alerts or evidence corresponding to the impact of the attack. For example, when a network-based IDS output a buffer overflow alert targeting a specific process running on the target host, and if the host-based IDS installed on the target machine also generated an alert representing an abnormal running of that process or other abnormal activities (e.g., illegal file access) corresponding to the evidence of the attack impact, then we can enforce the validity of the buffer overflow alert.

Priorities are important to classify alerts and quickly discard information that is irrelevant or of less importance to a particular site. The alert prioritizing component has to take into account the security policy and the security requirements of the site where the correlation system is deployed. The objective is that, with the alert priority rank, security analyst can select important alerts as the target alerts for further correlation and analysis.

Specifically, the priority score of an alert is computed based on the relevance of the alert to the configuration of the protected networks and hosts as well as the severity of the corresponding attack assessed by the security analyst. In practice, a correlation system uses the information from the impact analysis and the asset database to determine the importance of network services to the overall mission goals of the network.

Porras et al. proposed a more comprehensive mechanism of incident/alert rank computation model in a "mission-impact-based" correlation engine, named M-Correlator [67]. Since we focus on alert correlation and scenario analysis instead of alert priority ranking, and alert prioritization is just an intermediate step to facilitate further alert analysis, we adapted the priority computation model of M-Correlator with a simplified design.

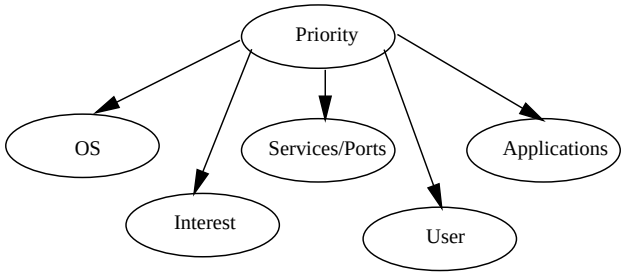


Figure 4: Alert Priority Computation Model

Figure 4 shows our priority computation model that is constructed based on Bayesian networks [65]. We use Bayesian inference to obtain a belief over states (hypotheses) of interests. A Bayesian network is usually represented as a directed acyclic graph (DAG) where each node represents a variable, and the directed edges represent the causal or dependent relationships among the variables. A conditional probability table (CPT) [65] is associated with each child node. It encodes the prior knowledge between the child node and its parent node. Specifically, an element of the CPT at a child node is defined by $CPT_{ij} = P(child\_state = j | parent\_state = i)$ [65]. The belief in hypotheses of the root is related to the belief propagation from its child nodes, and ultimately the evidence at the leaf nodes.

Specifically, in our priority computation model, the root represents the priority with

two hypothesis states, i.e., "high" and "low". Each leaf node has three states. For node "Interest", its three states are "low", "medium" and "high". For other nodes, the three states are "matched", "unmatched" and "unknown". The computation result is a value in [0,1] where 1 is the highest priority score.

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 $\sum_i P(H_i|e^1, e^2, \ldots, e^N) = 1$. For example, for the hyper alert of ***FTP Globbing Buffer Overflow*** attack, we get evidence [*high, matched, matched, unknown, unknown*] from the corresponding leaf nodes, i.e., Interest, OS, Services/Ports, Applications and User, respectively. As Figure 4 shows, the root node represents the priority of hyper alert. Assume that we have the prior probabilities for the hypotheses of the root, i.e., $P(Priority = high) = 0.8$ and $P(Priority = low) = 0.2$, and the following conditional probabilities as defined in the CPT at each leaf node, $P(Interest = high|Priority = high) = 0.70$, $P(Interest = high|Priority = low) = 0.10$, $P(OS = matched|Priority = high) = 0.75$, $P(OS = matched|Priority = low) = 0.20$, $P(Services = matched|Priority = high) = 0.70$, $P(Services = matched|Priority = low) = 0.30$, $P(Applications = unknown|Priority = high) = 0.15$, $P(Applications = unknown|Priority = low) = 0.15$, $P(User = unknown|Priority = high) = 0.10$, $P(User = unkown|Priority = low) = 0.10$, we then can get $\gamma = 226.3468$, therefore, $P(Priority = high|Interest = matched, OS = matched, Service = matched, Applications = matched, User = unknown) = 0.9959$. We regard this probability as the priority score of the alert. The current CPTs are predefined based on our experience and domain knowledge.

To calculate the priority of each hyper alert, we compare the dependencies of the corresponding attack represented by the hyper alert against the configurations of target networks and hosts. We have a knowledge base in which each hyper alert has been associated with

a few fields that indicate its attacking OS, services/ports and applications. For the alert output from a host-based IDS, we will further check if the target user exists in the host configuration. The purpose of relevance check is that we can downgrade the importance of some alerts that are unrelated to the protected domains. For example, an attacker may launch an individual buffer overflow attack against a service blindly, without knowing if the service exists. It is quite possible that a signature-based IDS outputs the alert once the packet contents match the detection rules even though such service does not exist on the protected host. The relevance check on the alerts aims to downgrade the impact of such kind of alerts on further correlation analysis. The interest of the attack is assigned by the security analyst based on the nature of the attack and missions of the target hosts and services in the protected domain.

CHAPTER IV

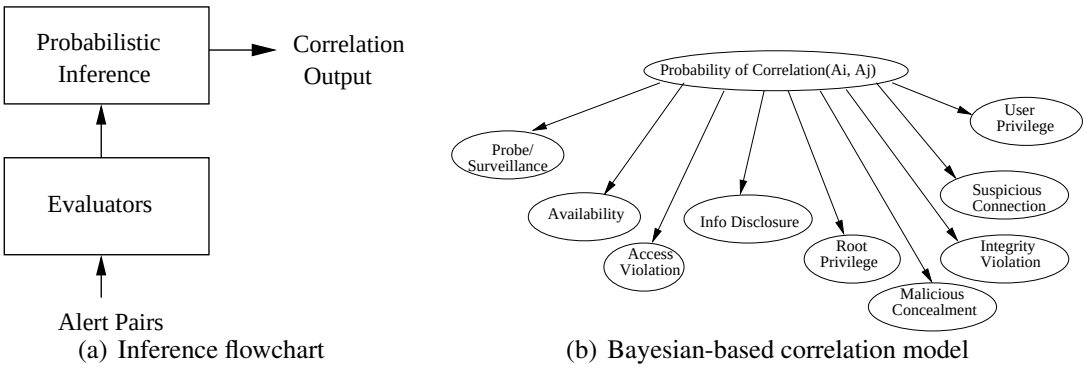
PROBABILISTIC-BASED ALERT CORRELATION

4.1 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. In a series of attacks where the attackers launch earlier attacks to prepare for later ones, there are usually strong connections between the consequences of the earlier attacks and the prerequisites of the later ones. If an earlier attack is to prepare for a later attack, the consequence of the earlier attack should at least partly satisfy the prerequisite of the later attack.

It is possible to address this type of correlation by defining pre- and 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- and post-conditions. In addition, in practice, an attacker does not have to perform early attacks to prepare for a later one, even though the later attack has certain prerequisites. For example, an attacker can launch an individual buffer overflow attack against a service blindly without knowing if the service exists or not. In other words, the prerequisite of an attack should not be mistaken for the necessary existence of an earlier attack. A hard-coded pre- and post-conditions matching approach cannot handle such cases.

Having the challenges in mind, we apply probabilistic reasoning to alert correlation by incorporating system indicators of attack consequences and prior knowledge of attack transitions. In this chapter, we discuss how to apply probabilistic reasoning to attack consequences and goals in order to discover the subtle relationships between attack steps in an



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

Figure 5: Probabilistic reasoning model

attack scenario.

4.2 Model Description

Figure 5(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.

In our inference module, we use a Bayesian network [65] 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 $CPT_{ij} = P(child\_state = j|parent\_state = i)$. We have more discussions on probability properties of a Bayesian network and two specific

Bayesian models in Section 5.2.1 and Section 8.4.2 respectively.

Figure 5(b) shows the structure of our Bayesian inference model for pairwise 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 be launched. The output of the inference engine represents the probability or confidence of the correlation between two alerts being analyzed (i.e., $P(correlation = high|evidence)$) based on the evidence (e.g., "matched" or "unmatched") provided by the leaf nodes.

The belief computation 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 $\sum_i P(H_i|e^1, e^2, \ldots, e^N) = 1$ [65]. 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 5(b) shows, each leaf node represents an attack consequence on the attack victim.

When reasoning about the correlation between two alerts, we consider broad aspects of attack consequences, in particular, (1) *Probe/Surveillance*: information on system or network has been gained by an attacker, e.g., a probing attack can get information on open ports. (2) *Availability*: the system is out of service or the service is negatively affected by the attack, e.g., because of a DoS attack. (3) *Access Violation*: an illegal access to a file or data of a system. (4) *Information Disclosure*: the attacker exports (sensitive) data to external site. (5) *Root Privilege* has been obtained by an attacker, for example, by a buffer overflow attack. (6) *Malicious Concealment*: malicious binary codes have been installed on the system, e.g., a Trojan horse. (7) *Integrity Violation*: the file on a system has been modified or deleted, violating the security policy. (8) *Suspicious Connection*: a covert channel has been set up by the attack. (9) *User Privilege* has been obtained by the attacker.

For each attack, it may result in one or more of those impacts on the victim host or network. Each attack may also need some pre-conditions prepared by prior attack(s) in one or more above fields. Therefore, when correlating two alerts, we compare the causal alert candidate's consequences with effected alert's pre-conditions in each leaf nodes of Figure 5(b).

Table 1: Predicates used in impact evaluation

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

Table 1 shows the set of predicates that we defined to assess the consequences of attack. Each attack impact shown in Figure 5(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*. For another example, an attack may result in compromise of the *root privilege* and modification of the *password* file at a victim host. The corresponding attack consequence can be represented by $\{$*GainRootPrivilege.targetIP*, *ModifiyData.passwordFile*$\}$.

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 5. 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* that is associated with attack consequence *Probe/Surveillance* as shown in Figure 5(b). Alert *imap buffer overflow* has a pre-condition as *GainServiceInfo.targetIP*, where predicate "GainServiceInfo" is associated with attack consequence *Probe/Surveillance* shown in Figure 5(b). If $portscan$ alert occurs before alert *imap buffer overflow* and they have the same target IP addresses, then their pre- and post-conditions are matched. The corresponding state of leaf node *Probe/Surveillance* in Figure 5(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.

4.3 Parameters in Bayesian Model

When using a Bayesian model for inference, we need to set two types of parameters, i.e., prior probability of root's states and CPT associated with each child node. In this section, we describe how we set the parameters used in our Bayesian model.

4.3.1 Parameters in Bayesian Model I: Prior Probability or Estimation on Attack Transition

In this section, we describe the attack classes used in our work and the prior probability estimation on attack transition, i.e., the root states in our model.

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 [83] also helps us on the probability estimation.

In our work, we denote the attack classes as the follows.

C1: Super Privilege Violation. C2: User Privilege Violation. C3: DoS. C4: Probe. C5: Access Violation. C6: Integrity Violation. C7: Asset Distress. C8: Connection Violation. C9: Malicious Binary Installation. C10: Exfiltration.

In alert correlation, the pair of alerts being evaluated in the correlation engine (as shown in Figure 5(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 $Super\ PrivilegeViolation$ 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.

Table 2: Prior Estimation on Attack Transition

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
C1	0.5	0.6	0.3	0.7	0.6	0.6	0.3	0.7	0.5	0.4
C2	0.6	0.5	0.3	0.6	0.5	0.5	0.3	0.1	0.5	0.4
C3	0.3	0.3	0.5	0.6	0.3	0.3	0.5	0.1	0.6	0.3
C4	0.2	0.2	0.3	0.5	0.7	0.3	0.3	0.8	0.3	0.3
C5	0.6	0.3	0.5	0.6	0.5	0.6	0.3	0.1	0.5	0.5
C6	0.5	0.3	0.5	0.4	0.8	0.5	0.3	0.1	0.5	0.4
C7	0.3	0.3	0.6	0.3	0.3	0.3	0.5	0.4	0.3	0.2
C8	0.1	0.1	0.3	0.4	0.3	0.3	0.5	0.5	0.3	0.7
C9	0.3	0.3	0.3	0.3	0.6	0.6	0.3	0.1	0.5	0.6
C10	0.5	0.5	0.3	0.3	0.6	0.6	0.3	0.3	0.6	0.5

Table 2 shows the estimated possibility that how reasonably an attack with class C_i (i.e., i^{th} column in the matrix) may progress to another attack with class C_j (i.e., j^{th} row in the matrix). The table entry is used as the prior probability of the root state in our model. The estimation is based on our prior experience and empirical studies. In the process of estimation, we also refer to the related work in [83].

4.3.2 Parameters in Bayesian Model II: Adaptive CPT Update

Another important parameter in Bayesian model is the CPT associated with each node. CPT values associated with each node adapt to new evidence and therefore can be updated accordingly. We apply an adaptive algorithm originally proposed by [4] and further developed by [18]. 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 [4] 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 [18]. 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.

4.4 Summary

Our alert correlation engine using Bayesian network has several advantages. First, we 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.

There are also several limitations in our approach. First, our correlation engine relies

on the underlying security sensors (e.g., IDSs) to provide alerts. If the security sensors miss a critical attack that links two stages of a series of attacks, the related attack steps may be split into two correlated groups. Therefore, we need some other techniques (e.g., attack plan recognition) to link isolated alert sets that includes correlated alerts. Second, our approach is based on domain knowledge of attack transition patterns. If there are new attack transition patterns or two related alerts have no direct causal relationship, our approach is not fully effective. Therefore, we need to develop complementary correlation techniques (e.g., statistical-based correlation technique) and use them along with our Bayesian-based correlation engine.

CHAPTER V

CAUSAL DISCOVERY-BASED ALERT CORRELATION

5.1 Motivation

Knowledge-based alert correlation system depends on attack transition patterns to correlate security alerts. It has the advantage of efficiency and accuracy. However, the signature-based correlation system lacks the capability of detecting the attack transitions whose scenario patterns are unknown. In practice, security analysts are more interested in those *novel* attack strategies that can easily evade signature-based correlation analysis and can potentially cause more damages due to the lack of knowledge about them.

Bearing this challenge in mind, we have studied and built a correlation technique based on statistical analysis. This correlation engine is based on the hypothesis that for some attack steps, even though they do not have direct causal relationship, they can have statistical dependence patterns. For example, a malicious daemon keeps uploading sensitive information to an external site and downloading new malicious code updates from the external site. For this type of attack transition patterns, our correlation engine applies causal discovery theory [66] to correlating alerts. Our goal is to identify *new* attack transition patterns beyond the limitation of domain-knowledge.

In this chapter, we introduce and describe our correlation mechanism using causal discovery theory.

5.2 Introduction to Causal Discovery

Causal discovery has been an active research topic in the fields of artificial intelligence (AI) and social science. The goal of causal discovery is to test and identify causal relationships among variables under study. Researchers have developed and shown that causal Bayesian

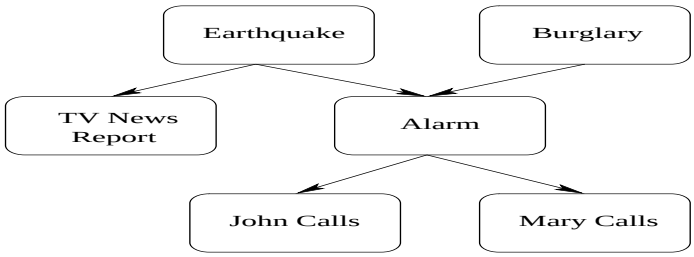


Figure 6: An example of a causal network

network can be used to represent the causal relationships between variables [66].

5.2.1 Causal Bayesian Network

A Bayesian network is usually represented as a directed acyclic graph (DAG) where each node represents a variable, and the directed edges represent the causal or dependent relationships among the variables.

Figure 6 shows an example of a causal network adapted from [65]. Here, a house alarm may sound as a result of either a burglary or an earthquake. An earthquake may also result in a TV news report. Neighbors John or Mary may report a call when the alarm sounds. The directed edge represents the cause-effect between variables.

In practice, causal discovery can be regarded as a task of constructing causal Bayesian networks from observational data.

Learning a Bayesian network from data includes two subtasks, i.e., learning the structure of the Bayesian network and learning the parameters of the network. The first subtask learns the causal relationship between variables and the second one represents the strength of these dependencies, which are encoded in conditional probability tables (CPTs) associated with each child node. Specifically, an element of the CPT at a child node is a conditional probability defined as $CPT_{ij} = P(child\_state = j|parent\_state = i)$ [65]. Since it is relatively straightforward to learn the parameters given observational data and a causal network structure, the challenge in causal discovery is the first task, i.e., learning the network structure from data sets.

In the causal discovery theory, the fundamental assumption is *causal Markov condition*.

Causal Markov condition means that, in a causal Bayesian network, any node is conditionally independent of its non-descendants (i.e., non-effect nodes) given its parent nodes (i.e., direct causes) [79]. The independence relationships represented by the structure of a causal Bayesian network are given by the causal Markov condition.

Figure 6, the independence properties implied by the network intuitively satisfy the notion of causality. For example, when an earthquake or burglary happens, the probability of neighbor John or Mary hearing the alarm will increase. An example of casual Markov condition is that, when the alarm has sounded (i.e., given the direct cause of John's calling), the belief that John will report the alarm sound is independent of an earthquake or burglary (i.e., a non-effect node of John's calling).

The conditional independence properties of a causal network can be deduced from the structure of the DAG by the *d-separation* criterion as defined in [65].

Definition 4 *If X, Y and Z are three disjoint subsets of nodes in a causal network D with a DAG structure, then Z is said to* **d-separate** *X from Y, denoted as $I < X|Z|Y >_D$, if along every path between a node in X and a node in Y there is node w satisfying one of the following conditions: (1) w has converging arrows and none of w or its descendants are in Z, or (2) w does not have converging arrows and w is in Z.*

In the example of Figure 6, if $X = \{John\ Calls\}$, $Y = \{Mary\ Calls\}$ and $Z = \{Alarm\}$, according to second criteria in *d-separation* definition, X and Y are *d-separated* by Z because there is no converging arrows at $w = \{Alarm\}$ along the path $X - Z - Y$ and $Alarm$ also belongs to Z. In other words, given the fact of $Alarm$ sounds, the probability of John's call is independent with Mary's activity. On the other hand, applying *d-separation* criterion, we can deduce that the belief on $Earthquake$ and $Burglary$ are dependent on the evidence of $Alarm$. It fits our intuitive notions of causality. When $Alarm$ has occurred, our increasing confidence on $Earthquake$ reduces the belief that $Burglary$ causes $Alarm$.

The structure of a causal network under discovery is a directed acyclic graph (DAG) that encodes conditional independencies via the causal Markov assumption. Learning the

Bayesian network structure from the data actually is the process of identifying the conditional independency among variables.

5.2.2 Approaches to Causal Discovery

Based on causal Markov assumption, there have been many research work on causal discovery. Generally, there are two approaches to discovering causal Bayesian networks.

One causal discovery approach is based on score functions, e.g., Bayesian computation [21,30,39]. Intuitively, this approach computes the probability that the causal relationship exists among the variables. For each pair of variables, a probabilistic computation is conducted to exam the dependence or independence between the two variables. In looking for the structures that fit for the conditional independence constraints, the approach in [39] makes *probabilistic* inferences about the conditional-independence constraints and the goal is to find the Bayesian network structures that have maximum score.

In [39], the score is defined as the posterior probabilities $p(m|D)$, where m corresponds to the causal network models learned from the given data D. This Bayesian-based approach can give a quantitative evaluation of causal network structures constructed from data. The goal is to identify a causal network structure $\tilde{m}$ ($\tilde{m} \in m$) so that $p(\tilde{m}|D)$ has the maximum value among all other causal network structures learned from data D. One challenge to this approach is model search and selection. Researchers usually use *model selection* method to select the best fitted model among others (i.e., the one with highest posterior probability $p(m|D)$) or *selective model averaging* method to average a number of better fitted models from all models [39]. There are still challenges to these two model selection methods, in particular, the accuracy issue [39]. In practice, people use some heuristic search algorithms to solve the model selection problems. However, those heuristic search algorithms may not give the best causal Bayesian network structures. Some scoring-based algorithms also have the issues that the different input ordering of variables can generate very different causal network structures.

Another category of causal discovery mechanism is *constraint-based* or *dependency analysis-based* approach (e.g., [14, 79]). This category of approaches usually apply statistical tests (e.g., χ^2 test, a statistical test for accepting or rejecting an hypothesis [38], and mutual information, a measure of dependency between variables [22]) to discovering conditional independence and dependence among variables and use these relationships as constraints to construct a Bayesian network. Specifically, for each pair of variables, this approach tests if any dependence exists. If so, an edge will be added between these two variables accordingly. Further tests will be conducted on each edge to examine if the two end-nodes are found to be conditionally independent. If the conditional independence is identified, the edge will be removed. The intuition on this approach is that a pair of nodes with larger test score (e.g., mutual information that measures the dependency between variables) is more likely to represent a direct connection (an edge) than a pair with smaller test score, which may represent an indirect connection. Search and scoring methods can be applied to identifying the directions of edges.

In our work, we applied *constraint-based* approach using mutual information for conditional independence test [14].

In information theory [22], *mutual information* is defined and used to measure the statistical dependence between two random variables.

Definition 5 *For two random variables X and Y with a joint probability distribution $P(x, y)$ and marginal probability distributions $P(x)$ and $P(y)$, mutual information $I(X, Y)$ is defined as [22]*

$$I(X, Y) = \sum_{x,y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \tag{4}$$

Definition 6 *For three random variables X, Y and Z with a joint probability distribution $P(x, y, z)$ and conditional probability distributions $P(x, y|z)$, $P(x|z)$ and $P(y|z)$, the conditional mutual information $I(X, Y|Z)$ is defined as [22]*

$$I(X, Y|Z) = \sum_{x,y,z} P(x, y, z) \log \frac{P(x, y|z)}{P(x|z)P(y|z)} \tag{5}$$

Intuitively, mutual information $I(X, Y)$ measures the information of X that is shared by Y. If X and Y are independent, then X contains no information about Y and vice versa, so their mutual information is zero. If X and Y are dependent, knowing the value of one variable can give us some information about the value of the other. In building the causal Bayesian network, we can apply mutual information to test if two variables are dependent and evaluate the strength of corresponding dependence.

Similarly, *conditional mutual information* $I(X, Y|Z)$ is used to test if two variables (i.e., X and Y) are dependent given the condition variable Z.

In theory, we claim X and Y are independent when $I(X, Y) = 0$ given the actual distributions of corresponding variables. In practice, given a data set D, we use empirical instead of theoretic distributions of variables when computing mutual information. Therefore, the normal practice is usually to set up a small threshold ϵ and claim X and Y are independent when $I(A, B) < \epsilon$. Similarly, we declare X and Y are conditionally independent given Z when $I(X, Y|Z) < \epsilon$.

The intuition of applying mutual information to causal discovery is that when two variables have a strong statistical dependency pattern, mutual information can detect it. The causality direction is determined by the conditional mutual information measure under the assumption that, in a causal network, two cause nodes are independent with each other, but conditionally dependent with each other given a common effect node. Specifically, based on mutual information measure, if we have identified variable A, B are mutually independent (i.e., $I(A, B) < \epsilon$), and are dependent with C respectively (i.e., $I(A, C) > \epsilon$, $I(B, C) > \epsilon$), and if we have also identified variable A and B are conditionally dependent given C based on conditional mutual information measure (i.e., $I(A, B|C) > \epsilon$), then we can determine that A, B are causes to C, i.e., $\{A \rightarrow C, B \rightarrow C\}$. Such structure is called

V-structure [39]. Such determination intuitively satisfies the notion of causality because when an effect is determined (i.e., given C), the increasing confidence on cause A reduces the belief that B causes C. In our example of Figure 6, we have seen such cause-effect pattern among *Earthquake*, *Burglary* and *Alarm*.

In our work, we did not to select the score function-based approach (e.g., [39]) because that approach usually requires prior knowledge (e.g., prior probability) of causal network models in the model construction, model comparison and final model selection. According to our experience it is difficult to get such prior knowledge in the security application. In fact, our goal is to identify novel attack transition patterns that can be totally unknown in the past. In [14], the researchers have developed algorithms to avoid complex conditional independence tests based on mutual information divergence. The enhanced test algorithms have eliminated the need for an exponential number of conditional independence tests that is an issue in earlier constraint-based algorithms.

5.3 Applying Causal Discovery Analysis to Alert Correlation

Before we apply causal-discovery approach to alert correlation, raw alerts need to get aggregated and clustered into *hyper alerts* as described in Section 3.1 so that we can investigate the statistical patterns between alerts.

After the above process, we formulate transaction data for each hyper alert. Specifically, we set up a series of time slots with equal time interval, denoted as t_{slot}, along the time axis. Given a time range T, we have $m = T/t_{slot}$ time slots. Recall that each hyper alert A includes a set of alert instances with the same attributes except time stamps, i.e., $A = [a_1, a_2, \ldots, a_n]$, where a_i represents an alert instance in the cluster. We denote $N_A = \{n_1, n_2, ..., n_m\}$ as the variable to represent the occurrence of hyper alert A during the time range T, where n_i is corresponding to the occurrence (i.e., $n_i = 1$) or un-occurrence (i.e., $n_i = 0$) of the alert A in a specific time slot $slot_i$. In other words, if there is one or more

instances of alert A (e.g., a) occurring in the time slot $slot_i$, then $n_i = 1$; otherwise, $n_i = 0$.

Using the above process, we can create a set of transaction data and input them to the causal discovery engine for analysis. Table 3 shows an example of the transaction data corresponding to hyper alert A, B and C. The correlation engine will output the causal network model based on transaction data set.

Table 3: An example of transaction data set

Time slot	$Alert_A$	$Alert_B$	$Alert_C$
$slot_1$	1	0	1
$slot_2$	0	0	1
...	...	...	...
$slot_i$	1	0	0
$slot_m$	0	0	1

Algorithm 1 Alert correlation using causal discovery theory

1. For each alert pair A_i, A_j
if A_i and A_j are dependent using mutual information measure, i.e., $I(A_i, A_j) > \epsilon$, where ϵ is a small threshold, **then**
Connect A_i and A_j directly.
end if
2. For any three alerts A_m, A_n, A_k that have the connection pattern that A_m and A_n, A_n and A_k are directly connected, and A_m and A_k are not directly connected (i.e., $A_m - A_n - A_k$)
if A_m and A_k are conditionally dependent given A_n using conditional mutual information measure, i.e., $I(A_m, A_k|A_n) > \epsilon$ **then**
Let A_m be the parent node of A_n, and A_k be the parent node of A_n, respectively, (i.e., $A_m \rightarrow A_n$ and $A_k \rightarrow A_n$).
end if
3. For any three alerts A_m, A_n, A_k that have a partially directed pattern ($A_m \rightarrow A_n - A_k$), i.e., A_m is a parent node of A_n, A_n and A_k are directly connected (edge (A_n, A_k) is not oriented), and A_m is not directly connected with A_k
if A_m and A_k are conditionally independent given A_n, i.e., $I(A_m, A_k|A_n) < \epsilon$, **then**
Let A_n be the parent node of A_k, i.e., $A_n \rightarrow A_k$.
else if A_m and A_k are conditionally dependent given A_n, i.e., $I(A_m, A_k|A_n) > \epsilon$, **then**
Let A_k be the parent node of A_n, i.e., $A_k \rightarrow A_n$.
end if

Algorithm 1 shows the steps to apply causal discovery theory to correlating alerts. In

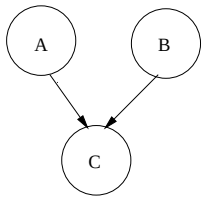


Figure 7: An example of the causal network model of alert A, B and C

step 1, we apply mutual information measure to identify alerts with strong statistical dependence. In step 2, we identify alert triplets that have a *V-structure* (i.e., $X \rightarrow Z, Y \rightarrow Z$, as described in Section 5.2). The causality directions in a *V-structure triplets* are determined by the conditional mutual information measure under the assumption that, in a causal network, two cause nodes are respectively dependent with a common effect node. These two cause nodes are mutually independent with each other, but conditionally dependent with each other given a common effect node. In step 3, for the partially directed alert triplets, since A_m and A_k are not directly connected, it means A_m and A_k are mutually independent (otherwise they should have been connected in step 1). The causality direction between A_n and A_k is tested based on the causal Markov assumption (i.e., in a causal network, a node X is independent to other nodes (except its direct effect node) given X's direct cause). Therefore, if A_m and A_k are also conditionally independent given A_n, we can identify the causality direction between A_n and A_k (i.e., $A_n \rightarrow A_k$). Otherwise, if A_m and A_k are conditionally dependent given A_n, the triplet has a *v-structure*, then A_k is the parent node of A_n (i.e., $A_k \rightarrow A_n$).

Table 4: An example of CPT associated with node B

	$AB = 0$	$AB = 01$	$AB = 10$	$AB = 11$
$C = 0$	p_1	p_2	p_3	p_4
$C = 1$	p_5	p_6	p_7	p_8

Figure 7 shows an example of the causal network model among alert A, B and C of which A and B are two causal alerts of C. As described in Section 5.2.1, in a causal network, each non-root node is associated with a conditional probability table (CPT) that

shows the strength of the causal relationship between the node and its parent node. Table 4 shows the CPT entries associated with alert C in which "1" represents the occurrence of the alert and "0" represents the nonoccurrence. Among the CPT entries as shown in Table 4, we are more interested in p_6 and p_7. The value of p_6 represent the probability of the occurrence of alert C given that alert B has already occurred, i.e., $p_6 = P(C = 1|B = 1)$. Similarly, the entry of p_7 shows the dependency of alert C to the causal alert A, i.e., $p_7 = P(C = 1|A = 1)$. In practice, we can regard p_6 and p_7 as the likelihood of attack step transition from attack B to attack C and from attack A to attack C, respectively.

Given the transaction data, computing the CPT entries is more straightforward. For example, the value of p_6 can be empirically computed as $P(C = 1|B = 1) = \frac{\#\ of\ (B{=}1,C{=}1)}{\#\ of\ (B{=}1)}$. We can also apply the algorithm of adaptive CPT updates as described in Section 4.3.2 to update the parameters.

CHAPTER VI

TEMPORAL-BASED ALERT CORRELATION

6.1 Motivation

The motivation to develop another complementary correlation mechanism is to discover more attack step dependency that the prior correlation engines have missed. Our Bayesian-based correlation engine focuses on discovering alert pairs with direct causal relationship. The causal discovery theory-based correlation engine identifies attack steps with strong statistical dependence (in particular, the dependency among attacks with a non-loop one-way dependency structure). In order to discover attack steps that have loop dependency pattern and strong temporal patterns, we develop another correlation engine based on statistical and temporal analysis, in particular, the Granger Causality Test (GCT) [34]. In this section, we introduce our GCT-based correlation mechanism.

6.2 Time Series Analysis

Time series analysis aims to identify the nature of a phenomenon represented by a sequence of observations. The objective requires the study of patterns of the observed time series data.

There are two main goals of time series analysis: (a) identifying the nature of the phenomenon represented by the sequence of observations, and (b) forecasting (predicting future values of the time series variable). Both goals require that the pattern of observed time series data is identified and more or less formally described. Once the pattern is established, we can interpret and integrate it with other techniques to extrapolate future events.

A time series is an ordered finite set of numerical values of a variable of interest along the time axis. It is assumed that the time interval between consecutively recorded values is

constant. We denote a univariate time series as $x(k)$, where $k = 0, 1, \ldots, N-1$, and N denotes the number of elements in $x(k)$.

Time series causal analysis deals with analyzing the correlation between time series variables and discovering the causal relationships. Causal analysis in time series has been widely studied and used in many applications, e.g., economy forecasting and stock market analysis.

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*. In time series analysis theory, although there exist some other simple lagged correlation analysis, e.g., computing correlation coefficients between two time series variables, GCT has been proved to be more rigorous. GCT was originally proposed and applied in econometrics, it has been widely applied in other areas, such as weather analysis (e.g., [48]), automatic control system (e.g., [9, 32]) and neurobiology (e.g., [40, 47]).

Network security is another application in which time series analysis can be very useful. In our prior work [6, 7], we have used time series-based causality analysis for pro-active detection of Distributed-Denial-of-Service (DDoS) attacks using MIB II [80] variables. We based our approach on the Granger Causality Test (GCT) [34]. Our results showed that the GCT is able to detect the "precursor" events, e.g., the communication between Master and Slave hosts, without prior knowledge of such communication signatures, on the attacker's network before the victim is completely overwhelmed (e.g., shutdown) at the final stage of DDoS.

In this work, we apply the GCT to INFOSEC alert streams for alert correlation and scenario analysis. The intuition is that attack steps that do not have well-known patterns or obvious relationships may nonetheless have some temporal correlations in the alert data. For example, there are one or more alerts for one attack only when there are also one or more alerts for another attack within a certain time window. We can apply temporal

causality analysis to find such alerts to identify an attack scenario. We next give some background on the GCT.

6.3 Granger Causality and Granger Causality Test

The intuition of Granger Causality is that if an event *X* is the cause of another event *Y*, then the event *X* should precede the event *Y*. Formally, the Granger Causality Test (GCT) uses statistical functions to test if *lagged* information on a time-series variable *x* provides any statistically significant information about another time-series variable *y*. If the answer is yes, we say variable *x* Granger-causes *y*. We model variable *y* by two auto-regression models, namely, the Autoregressive Model (AR Model) and the Autoregressive Moving Average Model (ARMA Model). The GCT compares the residuals of the AR Model with the residuals of the ARMA Model. Specifically, for two time series variables *y* and *x* with size *N*, the Autoregressive Model of *y* is defined as:

$$y(k) = \sum_{i=1}^{p} \theta_i y(k-i) + e_0(k) \tag{6}$$

The Autoregressive Moving Average Model of *y* is defined as:

$$y(k) = \sum_{i=1}^{p} \alpha_i y(k-i) + \sum_{i=1}^{p} \beta_i x(k-i) + e_1(k) \tag{7}$$

Here, *p* is a particular lag length, and parameters α_i, β_i and θ_i $(1 \leq i \leq p)$ are computed in the process of solving the Ordinary Least Square (OLS) problem (which is to find the parameters of a regression model in order to have the minimum estimation error). The residuals of the AR Model is $R_0 = \sum_{k=1}^{T} e_0^2(k)$, and the residuals of the ARMA Model is $R_1 = \sum_{k=1}^{T} e_1^2(k)$. Here, $T = N - p$.

The AR Model, i.e., Eq.(6), represents that the current value of variable y is predicted by its past p values. The residuals R_0 indicate the total sum of squares of error. The ARMA Model, i.e., Eq.(7), shows that the current value of variable y is predicted by the

past p values of both variable y and variable x. The residuals R_1 represents the sum of squares of prediction error.

The Null Hypothesis H_0 of GCT is $H_0 : \beta_i = 0, i = 1, 2, \cdots, p$. That is, *x* does not affect *y* up to a delay of *p* time units. We denote *g* as the Granger Causality Index (GCI):

$$g = \frac{(R_0 - R_1)/p}{R_1/(T - 2p - 1)} \sim F(p, T - 2p - 1) \tag{8}$$

Here, $F(a, b)$ is Fisher's F distribution with parameters a and b [37]. F-test is conducted to verify the validity of the Null Hypothesis. If the value of *g* is larger than a critical value in the *F*-test, then we reject the Null Hypothesis and conclude that *x* Granger-causes *y*. Critical values of *F*-test depends on the degree of freedoms and significance value. The critical values can be looked up in a mathematic table [38].

The intuition of GCI (g) is that it indicates how better variable y can be predicted using histories of both variable x and y than using the history of y alone. In the ideal condition, the ARMA model precisely predicts variable y with residuals $R_1 = 0$, and the GCI value g is infinite. Therefore, the value of GCI (g) represents the strength of the causal relationship. We say that variable $\{x_1(k)\}$ is more likely to be causally related with $\{y(k)\}$ than $\{x_2(k)\}$ if $g_1 > g_2$ and both have passed the *F*-test, where g_i, $i = 1, 2$, denotes the GCI for the input-output pair (x_i, y).

6.4 Procedure of Data Processing in GCT

Before applying GCT to data sets, we propose a procedure of data processing. In each step, there are multiple possible testing techniques and we chose the one that is most commonly used and conveniently implemented.

Step 1: testing for individual stationary. This step is to statistically test if each data set is stationary. A stationary time series means the probability distribution is stable during the stochastic process. In this step, we use testing technique proposed by Dickey-Fuller [29].

Step 2: data transformations. For non-stationary data sets, we can apply transform functions to change a non-stationary time series into a stationary one. The most common used transformations are log transformation and the differencing transformation. They can be also used together. For example, an initial log transformation is followed by first differencing, i.e., $(1-L)Log(x(t)) = Log(x(t)) - Log(x(t-1))$, where L represents *lag operator* defined as $Lx(t) = x(t-1)$ and $(1-L)x(t) = x(t) - x(t-1)$.

Step 3: testing for multivariate independence. This step is to test if two time series variables are *statistically independent* of each other. The available test techniques are proposed by Chitturi [16] and Hosking [41].

In practice, we can go through this step and then conduct the GCT for the non-independent bivariates. Results of GCT can tell us if they are causally related and the causal order or direction. As an alternative, we can also skip this step and conduct GCT directly because we can also infer the variable relationship from GCT output that can tell if they are independent of each other or if there are any causal relationships.

Step 4: testing for co-integration of data sets. In this step, we can apply multivariate version of Dickey-Fuller Test or Johansen Test [45] to test the existence of co-integration between two time series. Theoretically, GCT can be conducted on two co-integrated time series variables. However, as Lee et al. [52] empirically pointed out, GCT can result in spurious causality when testing co-integrated variables. Therefore, we recommend not to apply GCT on co-integrated time series in order to avoid the inaccuracy.

Step 5: testing Granger Causality. As described in Section 6.3, we conduct the statistical hypothesis test with a significance level, e.g., 5% or 1%.

Step 6: confidence computation. This step is to compute the probability or confidence of correlation. As GCI conforms to F-distribution, i.e., $F(p, T-2p-1)$, therefore, we can compute the corresponding probability as: $P_{gct} = CDF_{F-distribtuion}(p, T-2p-1, GCI)$, which represents the correlation confidence between two variables.

6.5 Applying GCT in Alert Correlation

6.5.1 Alert Time Series Formulation

Before applying GCT to alert correlation, we need to formulate each hyper alert into a univariate time series.

Specifically, we set up a series of time slots with equal time interval, denoted as t_{slot}, along the time axis. Given a time range T, we have $m = T/t_{slot}$ time slots. Recall that each hyper alert or cluster A include a set of alert instances with the same attributes except time stamps, i.e., $A = [a_1, a_2, \ldots, a_n]$, where a_i represents an aggregated alert instance in the cluster, we denote $\tilde{A}$ as the corresponding time series variable of hyper alert A. $\tilde{A} = \{n_1, n_2, ..., n_m\}$, where each value n_i represents the number of alert instances of hyper alert A occurring within a specific time slot $slot_i$.

Table 5: An example of alert time series formulation

Time slot	Number of A's alert instances	$\tilde{A}$'s value
$slot_1$	1	1
$slot_2$	5	5
...	...	...
$slot_i$	9	9
$slot_m$	0	0

Table 5 is an example that shows how we formulate a time series variable for each hyper alert. From Table 5, we can see that the time variable $\tilde{A}$'s value equals the number of alert instances of hyper alert A occurring within a time slot.

We currently do not use categorical variables such as port accessed and pattern of TCP flags as time series variables in our approach.

6.5.2 GCT-based Alert Correlation

Applying the GCT to alert correlation, the task is to determine which hyper alerts among $A_1, A_2, ..., A_l$ most likely have the causal relationship with hyper alert B (a hyper alert represents a sequence of alerts in the same cluster). Based on alert priority value and

mission goals as described in Chapter 3, the security analyst can specify a hyper alert as a target (e.g., alert *Mstream_DDOS* against a database server) which other alerts are correlated with. The GCT algorithm is applied to the corresponding alert time series. The formulation of alert time series is described in Section 6.5.1.

As described in Section 6.5.1, values of a hyper alert's time series (e.g., $\tilde{B}$) represent the number of alert instances occurring within a certain time period. Specifically, given a hyper alert B, for each hyper alert pair, i.e., $(A_i, B), i = 1, 2, \ldots, m$, we apply GCT to their corresponding time series variables, i.e., $GCT(\tilde{A}_i, \tilde{B})$. In other words, we are testing the temporal correlation of alert instances to determine if $\tilde{A}_i$ has a causal relationship with $\tilde{B}$.

As described in Section 6.3, the GCT index (GCI) g returned by the GCT function represents the evidence strength of the cause-effect relationship, and GCI also conforms to F-distribution. In practice, after performing GCT computation on each pair of alert time series variables (e.g., $GCT(\tilde{A}_i, \tilde{B})$, $i = 1, 2, \ldots, m$), we record the alert time series variables whose GCI values have passed the F-distribution test (e.g., $\tilde{A}_1, \tilde{A}_5, \tilde{A}_9$), then select the corresponding hyper alerts (e.g., A_1, A_5, A_9) as candidates of causal alerts w.r.t. alert B. We rank order the candidate alerts according to their GCI values, then select the top *m* candidate alerts and regard them as being causally related to alert B. These candidate relationships can be further inspected by other techniques or security analyst based on expertise and domain knowledge. The corresponding attack scenario is constructed based on the correlation results.

In alert correlation, identifying and removing background alerts is an important step. We use *Ljung-Box* [54] test to identify the background alerts. The assumption is that background alerts have characteristic of randomness. The *Ljung-Box* algorithm tests for such randomness via autocorrelation plots. The Null Hypothesis is that the data is random. The test value is compared with critical values to determine if we reject or accept the Null Hypothesis.

When applying GCT, one important parameter is the variable p as shown in Eq.(6) and Eq.(7). This parameter represents the number of history values (or the length of lagged time window) needed when performing the GCT.

Given two hyper alerts A and B that have corresponding time series $\tilde{A} = \{\tilde{a_1}, \tilde{a_2}, ..., \tilde{a_i}, ..., \tilde{a_n}\}$ and $\tilde{B} = \{\tilde{b_1}, \tilde{b_2}, ..., \tilde{b_j}, ..., \tilde{b_n}\}$ respectively, we want to identify if A Granger-causes B or not. As described in Section 6.2, a time series variable is under the assumption that the time interval between consecutively recorded values is constant. Therefore, the position difference between time series instances can be regarded as the time delay between alert instances.

In our work, We denote the corresponding parameter p as $p_{\tilde{A}\tilde{B}}$. We set the parameter $p_{\tilde{A}\tilde{B}}$ as follows.

Definition 7 *Given a time series variable instance $\tilde{a_i}$ ($\tilde{a_i} \in \tilde{A}$ and $\tilde{a_i} \neq 0$) and its most adjacent time series instance $\tilde{b_j}$ ($\tilde{b_j} \in \tilde{B}$, $\tilde{b_j} \neq 0$ and $j > i$), we denote $\Delta d_{i,j}$ as the adjacent time delay between $\tilde{a_i}$ and $\tilde{b_j}$.*

$\Delta d_{i,j} = j - i$

We denote $d_{adjacent\_time\_gap}$ as a set variable that unions all the time delays between adjacent time series instances in$\tilde{A}$ and $\tilde{B}$.

$d_{adjacent\_time\_gap} = \bigcup \{\Delta d_{i,j}\}$

where $i, j = 1, 2, ..., n$.

We then set $p_{\tilde{A}\tilde{B}}$ as $p_{\tilde{A}\tilde{B}} = \max\{d_{adjacent\_time\_gap}\}$.

The intuition of the method of setting parameter p is that we want to have a time window with an enough length so that we can include all potential causal alerts with respect to an effect alert.

Figure 8 shows an example how we set the parameter p. In the figure, time series variable $\tilde{A}$ has 3 non-zero instances at $k = 2, 15, i$ (i.e., $\tilde{a_2}, \tilde{a_{15}}, \tilde{a_i}$), time series variable $\tilde{B}$ has 3 non-zero instances at $k = 4, 17, j$ (i.e., $\tilde{b_4}, \tilde{b_{17}}, \tilde{b_j}$). We set p as the maximum value of delays between adjacent time series instance as shown in Figure 8.

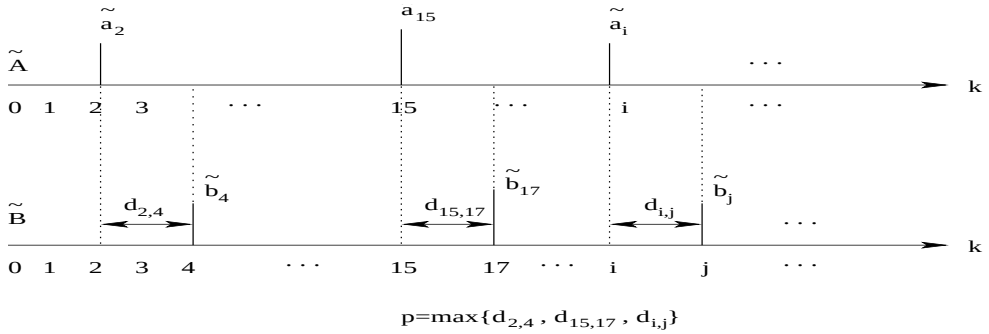


Figure 8: An example of time delay between time series instances

The main advantage of using statistical causality test such as GCT for alert correlation is that the approach does not require *a priori* knowledge about attack behaviors and how the attacks could be related. This approach can identify the correlation between two attack steps as long as the two have a temporal pattern (not necessarily high frequency) when occurring together. We believe that a large number of attacks, e.g., worms, have attack steps with such characteristics. Thus, we believe that causal analysis is a very useful technique. As discussed in [6–8], when there are 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, as discussed above, the top m candidates can be selected for further analysis by other techniques.

CHAPTER VII

SYSTEM INTEGRATION AND ATTACK SCENARIO ANALYSIS

7.1 Integration Process of Three Correlation Engines

Our three correlation engines are built on different techniques and focus on different correlation aspects. Bayesian-based correlation engine is analogous to an extension of pattern matching-based detection. Causal discovery theory-based correlation mechanism investigates statistical pattern of attack step occurrences to identify causal relationship between alerts. GCT-based correlation engine focuses on temporal pattern of attacks to discover new attack transition patterns.

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.

In practice, we integrate and apply the three correlation mechanisms with the following steps.

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* as described in Section 3.2. Thus, they should be the main interests in the correlation analysis to correlate with all the other hyper alerts. The goal of this step is to correlate alerts that have direct relationship based on prior knowledge of attack step transitions. The result of this step can be a set of isolated correlation graphs. For those alert pairs that have not got any causal relationship, we leave them to be processed in the next step.

Second, for those uncorrelated alert pairs, we run causal discovery-based correlation

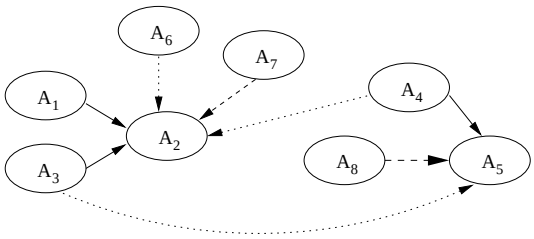


Figure 9: An example of integration process. The solid line represents a correlation identified by Bayesian-based correlation engine. The dotted line shows the causal relationship found by causal discovery-based correlation engine. The dashed line represents a new correlation specified by GCT-based correlation engine.

engine to correlate them. The goal of this step is to discover more correlation between alerts that have not been identified in the prior step.

Third, for each alert pair that has not established any cause-effect relationship from prior correlation engines, we apply GCT to it. That is, GCT is used to correlate alerts that have strong temporal relationship and link the isolated correlation results together.

Figure 9 shows an example of our integration process. For example, we have 8 hyper alerts, denoted as $A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8$. Assuming we have identified alert A_2 and A_5 as target alerts and we want to identify causal alerts w.r.t. A_2 and A_5 respectively. After applying Bayesian-based correlation engine, i.e., the first step of correlation, we have got two groups of correlated alerts, i.e., $\{A_1 \rightarrow A_2, A_3 \rightarrow A_2\}$ and $\{A_4 \rightarrow A_5\}$, as shown by solid lines in Figure 9. We then apply causal discovery algorithm to the rest isolated alerts that have not been correlated with A_2 and A_5 respectively. In particular, we check if causal relationship exists between alerts $\{A_1, A_5\}$, $\{A_2, A_5\}$, $\{A_3, A_5\}$, $\{A_6, A_5\}$, $\{A_7, A_5\}$, $\{A_8, A_5\}$, $\{A_4, A_2\}$, $\{A_5, A_2\}$, $\{A_6, A_2\}$, $\{A_7, A_2\}$ and $\{A_8, A_2\}$. Assuming after this step, we have got 3 more causal-related alert pairs, i.e., $\{A_3 \rightarrow A_5\}$, $\{A_6 \rightarrow A_2\}$, $\{A_4 \rightarrow A_2\}$ as represented by dotted lines in the figure. We finally apply GCT to check if the rest isolated alert pairs $\{A_1, A_5\}$, $\{A_2, A_5\}$, $\{A_6, A_5\}$, $\{A_7, A_5\}$, $\{A_8, A_5\}$, $\{A_4, A_2\}$, $\{A_5, A_2\}$ and $\{A_8, A_2\}$ have the causality w.r.t. A_5 and A_2 respectively. Figure 9 shows that GCT identifies the causality of $\{A_7 \rightarrow A_2\}$ and $\{A_8 \rightarrow A_5\}$ as shown by the dashed line.

7.2 Probability/Confidence Integration

In Section 4.2, we introduced our Bayesian-based correlation engine that outputs the correlation probability or 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 described in Section 5.3, the CPT associated with each child node in a causal network shows the strength of relationship between the child node and its parent node. Particularly, one CPT entry (i.e., $P(childnode = 1|parentnode = 1)$ can be interpreted as the probability of attack transition from parent node (attack) to child node (attack), e.g., the p_{11} in Table 4. We denote such attack transition probability as $P_{causal-discovery}$.

As discussed in Section 6.3, 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 three correlation engines, we can adjust the confidence output from GCT-based engine as:

$$P_{gct\_final} = (P_{gct} - t) * \omega + t \tag{9}$$

In Eq. (9), 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 adjusted value of P_{gct_final} is in the range of $[0, t + \epsilon]$, where ϵ is a small positive number. The intuition of this adjustment is that we want to

downgrade the output of GCT-based correlation engine a little because it is based on temporal 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 Bayesian correlation engine (i.e., P_{bayes}), causal discovery algorithm (i.e., $P_{causal\_discovery}$) or GCT-based correlation mechanism (i.e., P_{gct_final}) depending on which correlation engine identifies the causal relationship. We denote the probability of alert correlation (or attack transition) as $P_{correlation}(A_i, A_j)$, i.e.,

$$P_{correlation}(A_i, A_j) = \begin{cases} P_{bayes}, & \text{if causality found by Bayesian-based correlation engine} \\ P_{causal\_discovery}, & \text{if causality found by causal discovery-based correlation engine} \\ P_{gct\_final}, & \text{if causality found by GCT-based correlation engine} \end{cases} \tag{10}$$

We also note that two different approaches have been proposed to integrate isolated correlation graphs. Ning [59] et al. apply graph theory to measure and merge similar correlation graphs. In [60], 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 three complementary correlation engines instead of graph or pre- /post-condition-based merging algorithms.

7.3 Attack Transition Table Updates

Statistical and temporal-based alert correlation has the advantages of discovering attack transition steps without depending on prior domain knowledge. However, compared with pattern-matching correlation techniques, it is has relatively high positive false rate and the computation cost is also relatively high.

In practice, we periodically incorporate newly discovered attack transition patterns into our domain knowledge so that we can use our Bayesian-based correlation engine to analyze and correlate alerts efficiently. Also based on new analysis results and data sets, we update the attack transition table as shown in Table 2.

Denote θ as an original entry in Table 2, $\theta^{'}$ as the corresponding new value computed based on new analysis results and data after a regular period T, the current table update policy is that we do not update the table entry until the new value $\theta^{'}$ has varied from θ by a certain percentage β, e.g., 5%.

7.4 Attack Strategy Analysis

Attack strategy analysis is an important component in a correlation system. It can provides security analysts an aggregated information about what has happened and what is happening to the protected IT infrastructure.

Having correlated alert pairs output by correlation engines, we can construct attack scenarios represented by correlation graph to represent the attack strategies. A correlation 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 *cause node*, and the node corresponding to the effected alert as *effect node*. A threshold t is pre-defined and alert A_j is considered to be caused by alert A_i only when $P_{correlation}(A_i, A_j) > t$. In constructing scenario graphs, we only include the correlated alert pairs whose $P_{correlation}$ values are over the threshold t.

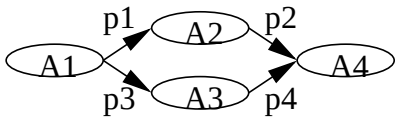


Figure 10: An example of correlation graph

In a correlation graph, each edge is associated with a correlation probability (i.e., $P_{correlation}$) from *cause node* to *effect node*, which can be also regarded as the probability of attack step transition. Having such information, we can perform *quantitative* analysis on the attack strategies. In a correlation graph, each path is potentially a subsequence of an attack scenario and can be seen as a Markov chain [31, 73]. Having the probability associated with each edge, for any two nodes in the graph that are connected by multiple paths, we can compute the overall probability of each path [73].

In the example of Figure 10, nodes A_1 and A_4 have to paths to connect each other. Assuming the conditional independence of A_4 and A_1, we can compute the overall probability of each path, e.g., $P(A_1, A_2, A_4) = P(A_4|A_2)P(A_2|A_1)P(A_1) = p_2 * p_1 * p_{A_1}$.

We then rank order and select the path(s) with the highest overall correlation probability as the most likely sequence(s) connecting two nodes.

Combining all the probability along each edge, we can also compute an overall probability of two nodes connected with multiple paths. For example, in the Figure 10, $P\{A_1\, to\, A_4\}$ $= 1 - (1 - p_1 * p_2)(1 - p_3 * p_4)$.

CHAPTER VIII

ATTACK PLAN RECOGNITION

In this chapter, we introduce our models and algorithms for correlating isolated alert sets and attack plan recognition.

8.1 Attack Tree Analysis

In security operations, security analysts usually pre-define a set of attack plans or attack libraries that incorporate the domain knowledge of attacks or attack scenario patterns, and the knowledge of the networks and systems under protection. Attack plans or libraries are usually represented by graphs (i.e., attack graphs) that show all paths through a system that end in a state where an intruder can successfully achieve his goal. Schneier [75] described *attack tree analysis* that quantifies the security or vulnerability of a system based on the goals of the attacker. When defining the attack trees, security analysts first evaluate the vulnerabilities of the systems and networks, then pretend to be attackers and work out attack plans to achieve the intrusion goals. In this process, an attack tree is extended and branches are built to identify the different subgoals of the attacker and penetration points available to the attacker. The process continues by decomposing or expanding the means of penetration to the lowest level of intrusion, known as the leaves. An attack tree can represent each opportunity for an attack against a computer system or network. Computer systems and networks potentially contain numerous penetration points and vulnerabilities. An attack forest is defined as a consolidation of numerous attack trees [75].

Figure 11 shows an example of an attack tree that indicates attack methods to steal the data stored on a server and export it to the external. In the Figure 11, the "OR" node represents *different ways* to achieve the goals. In practice, in addition to the "OR" node,

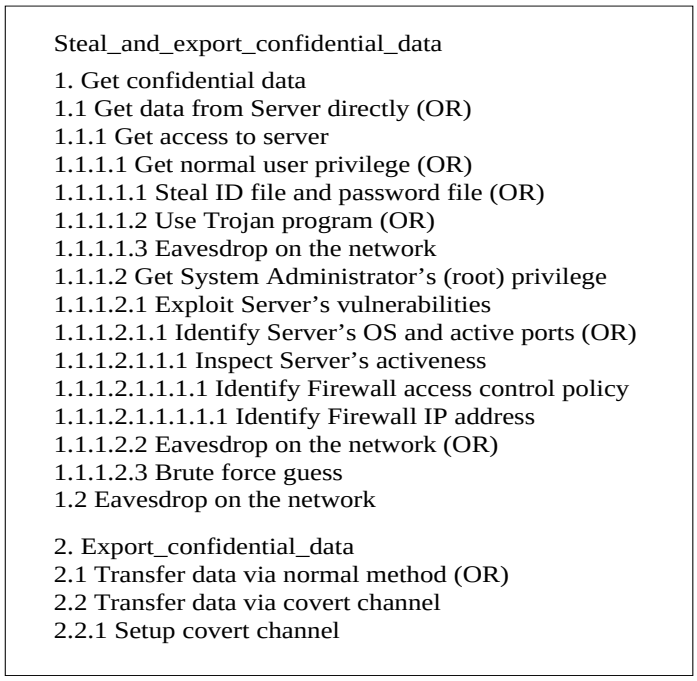


Figure 11: An example of attack tree

the "AND" node is also always used in an attack tree to represent *different steps* to achieve the intrusion goals.

Attack tree analysis can serve as a basis for intrusion detection, defense, response and forensic analysis. However, defining attack trees is a very challenging task. It is usually done manually and is very time consuming. Recently, Sheyner et. al [77] proposed a model checking-based technique to automatically construct attack graphs. Although it helps facilitate the task of defining attack graphs, the approach still has the limitation of scalability, in particular, when defining the attack graphs for a large network and computer systems.

In our approach, we first use attack trees to define attack plan libraries to correlate isolated alert sets. We then convert attack trees into causal networks on which we can assign probability distribution by incorporating domain knowledge to evaluate the likelihood of attack goals and predict future attacks. Figure 12 shows an example of the causal network converted from the attack tree as shown in Figure 11. In defining attack trees, instead of using various specific attacks to define the nodes of an attack tree, we use the abstract attack

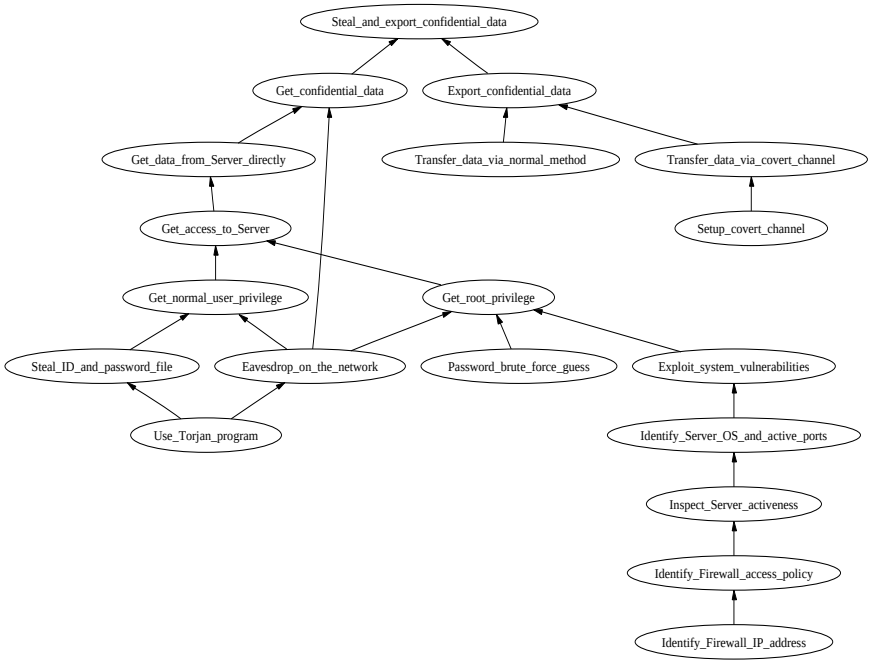


Figure 12: An example of a causal network converted from an attack tree

class or type to represent an attack approach. For example, we use *Exploit Server Vulnerability* instead of a specific buffer overflow attack to indicate the method to break into a server to get the root access. The advantage of using attack classes to represent attack tree nodes is that it can reduce the computation complexity of probabilistic inference on the causal network that is converted from attack trees. It is well known that querying an arbitrary causal network is an NP-hard problem [20]. Therefore, in practice, a causal network is usually defined in the form of *causal polytrees* (i.e., singly-connected causal networks in which no more than two paths exist between any two nodes) so that the probabilistic reasoning can be conducted in polynomial time [65].

8.2 Converting An Attack Tree to A Causal Network

In Section 5.2.1, we have introduced the causal Bayesian network and its probability properties. A causal network (or 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., $P(parent\_state = i)$), 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 element of the CPT at a child node is defined by $CPT_{ij} = P(child\_state = j | parent\_state = i)$.

In our study, we build the causal networks based on attack trees and apply probabilistic inference. The root node of a causal network represents the final goal of an attack plan, non-leaf nodes represent subgoals, and leaf nodes indicate the nodes receiving evidence. We define each node of the causal network to have a binary state, i.e., 1 or 0. The value of 1 represents the goal is achieved for goal or subgoal nodes, while the value of 0 indicates the failure of the goal or subgoals. When a leaf node has a state value of 1, it indicates that the leaf node has received evidence. Otherwise, the leaf node has a value of 0.

When converting attack trees to a causal network, we can map "OR" nodes from an attack tree directly to the causal network while keeping the "OR" logical relationship. As "AND" nodes in an attack tree represent different *attack steps* to reach a goal ("OR" nodes indicate different *attack ways* to achieve an attack goal), there always exists an implicit dependent and sequential relationship between "AND" nodes in an attack tree. Therefore, we should keep such "causal" order when constructing the causal network. For example, we can define an attack tree for getting access to a server with the following attack steps which have "AND" relationship, i.e., *exploit vulnerability* AND *identify server OS* AND *identify Firewall access control policy* AND *Identify Firewall IP address*. In the causal network, we can keep the implicit sequential order among the nodes, i.e., *identify IP address*, *identify firewall access control policy*, *identify server's OS*, *exploit vulnerability* in order of the causal sequence.

When using a causal network (or Bayesian network) for inference, we need to set up two types of parameters, i.e., prior probability of parent node's states and CPT associated with each child node.

The prior probability of parent node's states (e.g., $P(parent\ node\ state = 1)$) used in the inference engine is set based on the *prior knowledge* estimation of the possibility. We used domain-specific knowledge based on prior experience and empirical studies to estimate appropriate probability values. In particular, we computed the probability of parent node's states based on historical data.

In our approach, CPT values associated with each node are adaptive to new evidence and therefore can be updated accordingly. We apply an adaptive algorithm originally proposed by [4] as described in Section 4.3.2. The motivation of using an 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 [4] is that we want to adapt the new model by updating CPT parameters to fit the new data cases while balancing the extent that we move away from the current model.

8.3 Correlating Isolated Alert Sets

As discussed in Section 1.3, after processing raw alerts with alert aggregation, prioritization and correlation, we can reduce the large volume of raw alerts and correlate some of related alerts into different sets (or scenarios). However, it is possible that there exist some isolated correlated alert sets after the raw alert correlation due to various reasons. For example, for pattern-matching-based correlation approach, if the security sensors fail to detect some intermediate attacks in a series of coordinated attacks, the missing alerts can result in the un-match between observed alert sequences with known attack sequence patterns. The result is a set of isolated attack scenarios that belong to the same attack sequences. In addition, applying different correlation approaches together can also result in different correlation results due to the difference between correlation techniques. In such a case, it is also necessary to integrate correlation results output by different correlation engines and further correlate isolated alert sets. Third, from the security analyst's point of view, it is

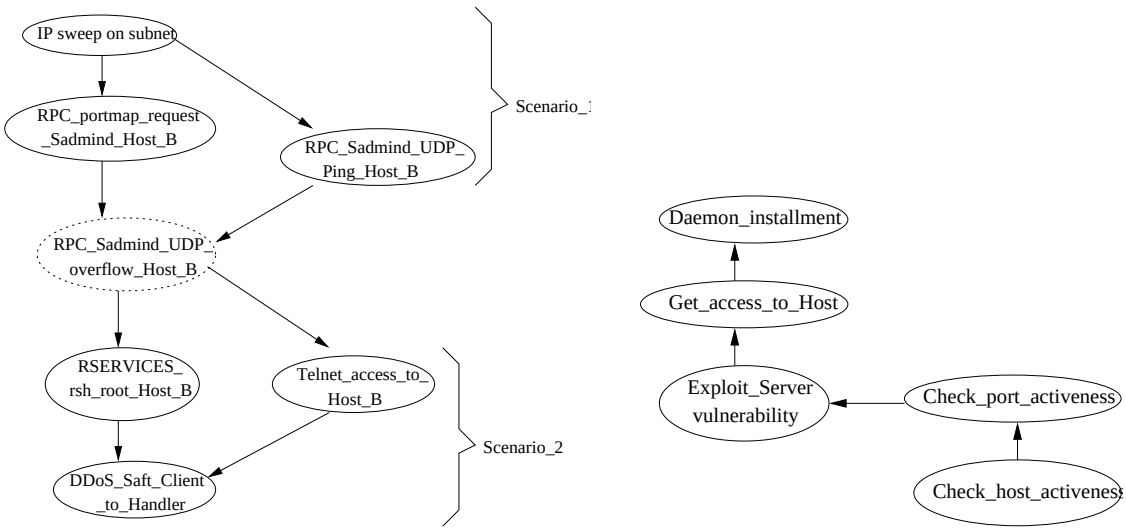
necessary to combine the local or low-level correlation results, investigate and assess the attack situation in order to make timely and appropriate response or prevention.

Given two individual isolated scenarios being studied, denoted as S_1 and S_2, where $S_1 = \{e_1, e_2, ..., e_i, ..., e_m\}$, $S_2 = \{e'_1, e'_2, ..., e'_i, ..., e'_n\}$ and e_i represents an alert (i.e., evidence), and given a set of attack plans, denoted as P, where $P = \{P_1, P_2, ..., P_k, ..., P_f\}$, and P_k is denoted as a specific attack plan that is represented by a causal network converted from attack trees, the problem is to find the relationship between S_1 and S_2. Algorithm 2 shows the method of correlating two isolated scenarios.

Algorithm 2 Correlation of isolated attack scenarios

Let $TPSet_1 = \{$Predecessor nodes of S_1 in $P_k\}$.
Let $TPSet_2 = \{$Predecessor nodes of S_2 in $P_k\}$.
Let P_k be an attack plan represented by a causal network, where $S_1 \in P_k$, and $S_2 \in P_k$.
Let $PSet_i = \{$Predecessor nodes of e_i in $P_k\}$, where $e_i \in S_1$.
Let $PSet'_i = \{$Predecessor nodes of e'_i in $P_k\}$, where $e'_i \in S_2$.
if $\exists e_j \in S_1$ and $e_j.attackClassNode \in PSet'_i$ and $e'_i.time < e_j.time$ and $\{e'_i.target = e_j.target$ or $e'_i.target = e_j.source\}$ **then**
  S_1 is a subgoal of S_2; S_1 and S_2 are directly related.
else if $\exists e'_j \in S_2$ and $e'_j.attackClassNode \in PSet_i$ and $e_i.time < e'_j.time$ and $\{e_i.target = e'_j.target$ or $e_i.target = e_j.source\}$ **then**
  S_2 is a subgoal of S_1; S_2 and S_1 are directly related.
else if $\{TPSet_1 \cap TPSet_2\} \neq \phi$ and they have the same target **then**
  S_1 and S_2 have indirect relationship with the same goal.
end if
Group all scenarios that are related in P_k into one evidence set.

The intuition of the correlation algorithm is to find out the relationship between two isolated attack scenarios. One relationship is that one scenario is a direct subgoal of another. This is indicated by the fact that one attack step in one scenario is a predecessor node of alerts of another scenario in the plan library. In such a case, a time constraint is applied to ensure that subgoal attack happens after the prior attacks. Another relationship is that two scenarios have an indirect relationship but have the same goal. For example, they both target at a same victim.



(a) Two correlated scenarios (b) The corresponding attack plan

Figure 13: An example of correlation of two isolated scenarios

Figure 13(a) shows an example of how we correlate two isolated attack scenarios derived from low-level alert correlation. Assuming that $scenario_1$ includes alerts *IP sweep*, *RPC_Portmap_request_Sadmind_ Host_B* and *RPC_Sadmind_UDP_Ping_Host_B*. $Scenario_2$ contains alerts *RSERVICES_rsh_root_Host_B*, *Telnet_access_to_Host_B* and *DDoS_Saft_ Client_to_Handler*. In this case, we assume the alert *RPC_Sadmind_UDP_Overflow_Host_B* is missed that results in the isolation of $scenario_1$ and $scenario_2$. According to the corresponding attack plan as shown in Figure 13(b), attacks in $scenario_1$ have corresponding attack class nodes *Check_host_activeness* and *Check_port_activeness* in Figure 13(b). Similarly, the attacks in $scenario_2$ have corresponding abstract attack class nodes *Get_access_to _host* and *Daemon_installment* in Figure 13(b). From Figure 13(b), we can see $scenario'_2s$ abstract class nodes are predecessors of $scenario'_1s$. Therefore, $scenario_2$ is actually a goal of $scenario_1$, i.e., the attacker gets ready to exploit the host vulnerability by launching attacks in $scenario_1$ so that he can access the host and install the DDoS daemon (as shown by attack steps in $scenario_2$). Although the alert corresponding to *Exploit host vulnerability* is missed, we can hypothesize the existence of such alert and apply the scenario correlation technique to correlate the two isolated scenarios, i.e., $scenario_1$ and $scenario_2$.

8.4 Probability Evaluation and Attack Plan Recognition

We apply probabilistic inference to the causal network (or Bayesian network) to compute and evaluate the likelihood of goal and subgoals based on observed attack activities, and predict the potential upcoming attacks.

8.4.1 Iterative Belief Propagation Concepts

The inference process of a Bayesian network can be conducted by a series of belief propagations via message passing [65]. Specifically, in a tree-structured Bayesian network, assuming a node X has a parent node U and m children nodes, $V_1, V_2, ..., V_m$, the belief updating can be accomplished in three steps.

First X receives a prior or causal support message from its parent node U, denoted as $\pi_X(u)$, where $\pi_X(u) = P(u)$. Node X also receives evidence or diagnostic support message $\lambda_{Vj}(x), j = 1, 2, ..., m$ from each of its child nodes, where $\lambda_{Vj}(x) = P(v_j|x)$. In run-time, when a node X is activated, it first updates the belief ***Belief(x)***, i.e., the probability of X's states ($P(X = x|evidence)$), based on the evaluation values and the message $\pi_X(u)$ communicated with its parent node and $\lambda_{Vj}(x)$ sent by its child nodes, as shown in Eq.(11). This is called *belief updating*.

$$Belief(x) = \alpha\lambda(x)\pi(x) \tag{11}$$

where,

$$\lambda(x) = \prod_j \lambda_{Vj}(x) \tag{12}$$

$$\pi(x) = \sum_u P(x|u)\pi_X(u) \tag{13}$$

where α is a normalizing constant rendering $\sum_x Belief(x) = 1$.

Next, the node X computes its own λ message based on λ messages received from its child nodes, then sends it to its parent node, as shown in Eq.(14). This phase is *bottom-up propagation*.

$$\lambda_X(u) = \sum_x \lambda(x) P(x|u) \tag{14}$$

Finally, the node X computes its own π messages and sends them to each of its child nodes, e.g., to its j^{th} child node V_j, as shown in Eq.(15). The final step is *top-down propagation*.

$$\pi_{Vj}(x) = \alpha\pi(x) \prod_{k \neq j} \lambda_{Vk}(x) \tag{15}$$

In practice, the local belief update on node X can be executed by these three steps in any order. The belief propagation algorithm in polytrees starts from the evidence node and propagates the changed belief along the graph edges by computing *Belief(x)*, $\lambda_X(u)$ and $\pi_{Vj}(x)$ at every visited node. More detailed information can be found in [65].

8.4.2 Link Matrix at "noisy-OR" and "noisy-AND" Causal Networks

In our work, we convert an attack tree to a causal network while keeping the logical "OR" and "AND" relationship between attack steps. The corresponding causal network is usually called "noisy-OR" and "noisy-AND" causal network [65].

In this section, we introduce the CPTs (Link Matrix) at a "noisy-OR" or a "noisy-AND" causal network. The special characteristics of "noisy-AND" and "noisy-OR" structures have made the computation of belief propagation more efficient. In our work, we assume the causal network has a polytree structure.

8.4.2.1 "noisy-OR" Model

In a "noisy-OR" causal network, as in the case of the logical *OR*, an effect node X is presumed to be false (i.e., $P(X) = 0$) if all the condition that cause E are false. However,

unlike a logical *OR*, if one of the causes of the event X is true, it does not necessarily imply that X is definitely true. We can consider each cause node has an associated inhibitory influence that is active with a probability q ($q = P(X = 0|U = 1)$), where U is the cause of X. Therefore, if a cause node U is the only cause of X, then $P(X = 1|U = 1) = 1 - q$.

Consider a "noisy-OR" causal network, assuming a node X has a set of parent nodes, denoted as $U = \{U_1, U_2, ..., U_n\}$, and a set of child nodes, denoted as $V = \{V_1, V_2, ..., V_m\}$. Parent nodes have a logical "OR" relationship w.r.t. node X. Assuming each node (X and U_i) has a binary state 0 or 1. Denote q_i as a conditional probability, i.e., $P(X = 0|U_i = 1)$, and $c_i = 1 - q_i$ (i.e., $c_i = P(X = 1|U_i = 1)$), $T_u = \{i : U_i = 1\}$, we can have the following [65]:

$$P(X = x|U) = \begin{cases} \prod_{i \in T_u} q_i, & \text{if } x = 0 \\ 1 - \prod_{i \in T_u} q_i, & \text{if } x = 1 \end{cases} \tag{16}$$

$P(X|U)$ is named link matrix that actually represents the CPT associated with node X in a "noisy-OR" polytree. The intuition of q_i is that it acts as an inhibitory factor while c_i is the degree of belief that a cause node U_i can endorse the effect node of X. Having known the link matrix $P(X|U)$, we can follow the message propagation method as described in Section 8.4.1 for belief update. In a "noisy-OR" model, belief of X ($Belief(x)$) can computed using the following equations [65].

$$Belief(x) = \begin{cases} \alpha \prod_j \lambda_{Vj}(x) \prod_{i \in T_u} (1 - c_i \pi_X(u_i)), & \text{if } x = 0 \\ \alpha \prod_j \lambda_{Vj}(x)[1 - \prod_{i \in T_u} (1 - c_i \pi_X(u_i))], & \text{if } x = 1 \end{cases} \tag{17}$$

and α is a normalizing constant. Details of theoretic analysis can be referred to [65].

For example, considering a simple "noisy-OR" polytree structure as shown in Figure 14. Node X has two parent nodes U_1 and U_2 which have a logical OR relationship with X. Assuming each node has two states, i.e., 0 or 1. Denote $q_1 = P(X = 0|U_1 = 1)$ and $q_2 = P(X = 0|U_2 = 1)$.

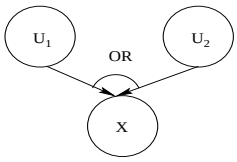


Figure 14: An example of a "noisy-OR" structure

Table 6: An example of CPT in a "noisy-OR" polytree

U_1U_2=	00	01	10	11
$X = 0$	1	q_2	q_1	q_1q_2
$X = 1$	0	$1 - q_2$	1-q_1	$1 - q_1q_2$

Table 6 shows the CPT entries associated with node X.

8.4.2.2 "noisy-AND" Model

Similarly, a *"noisy-AND"* causal network is a generalization of a logical *And*. As in the case of a logical *And*, an effect node X is true (i.e., $P(X = 1)$) if all the conditions that cause X are true. However, unlike the logical *And*, if one of the causes of node X are false, it does not imply that X is definitely false. In other words, we can regard each cause node of X as having an associated enabling influence that is active with a probability c ($c = P(X = 1|U = 0)$. Therefore, if a cause node U is the only cause of X and U is false (i.e., $U = 0$), then X is false with a probability of $P(X = 0|U = 0) = 1 - c$.

Assuming that node X has a set of parent nodes, denoted as $U = \{U_1, U_2, ..., U_n\}$, and a set of child nodes, denoted as $V = \{V_1, V_2, ..., V_m\}$. Keeping the prior definition of q_i and c_i (i.e., $q_i = P(X = 0|U_i = 0)$, $c_i = P(X = 1|U_i = 0)$ and $F_u = \{i : U_i = 0\}$, for a generic "noisy-AND" causal polytree, we can have the link matrix (or CPT associated with node X) as follows.

$$P(X = x|U) = \begin{cases} 1 - \prod_{i \in F_u} c_i, & \text{if } x = 0 \\ \prod_{i \in F_u} c_i, & \text{if } x = 1 \end{cases} \tag{18}$$

In a "noisy-AND" model, denote $T_u = \{i : U_i = 1\}$, the belief of X ($Belief(x)$) can be computed using the following equation [65].

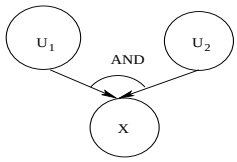


Figure 15: An example of a "noisy-AND" structure

$$Belief(x) = \begin{cases} \alpha \prod_j \lambda_{Vj}(x)(1 - \prod_{i \in T_u} (1 - c_i(1 - \pi_X(u_i)))), & \text{if } x = 0 \\ \alpha \prod_j \lambda_{Vj}(x) \prod_{i \in T_u} (1 - c_i(1 - \pi_X(u_i))), & \text{if } x = 1 \end{cases} \tag{19}$$

and α is a normalizing constant. Details of theoretic analysis can be referred to [65].

For example, considering a simple "noisy-AND" polytree structure as shown in Figure 15. Node X has two parent nodes U_1 and U_2 which have a logical "AND" relationship with X. Assuming each node has two states, i.e., 0 or 1. Denote $q_1 = P(X = 0|U_1 = 0)$, $c_1 = 1 - q_1$, $q_2 = P(X = 0|U_2 = 0)$, $c_2 = 1 - q_2$.

Table 7: An example of CPT in a "noisy-AND" polytree

U_1U_2=	00	01	10	11
$X = 0$	$1 - c_1c_2$	$1 - c_1$	$1 - c_2$	0
$X = 1$	c_1c_2	c_1	c_2	1

Table 7 shows the CPT entries associated with node X. More details of theoretic analysis can be found in [65].

8.4.3 Attack Evaluation and Prediction

Given a stream of evidence (i.e., alerts), and a causal network (i.e., attack plan) P, the inference through iterative belief updating is shown in Algorithm 2.

Intuitively, given a set of correlated alerts as observed evidence, we input the evidence into the causal network so that we can make inference and compute the likelihood for each non-leaf node, denoted as Z_i, as shown in Algorithm 3. The computation result is used to infer the likelihood that a node can be the attack goal or intention, i.e., $P(Z_i = 1|evidence)$.

Algorithm 3 Likelihood computation of attack goal and subgoal

Let P be a causal network and each node in the P has a binary state, $\{0,1\}$.
for all node $Y_i \in P$ that receives evidence e_i **do**
  Mark Y_i as an observed node with value of 1
  Let M be a collection nodes resulted from breath-first search starting from Y_i
  for all node $X \in M$ **do**
    Receive $\lambda(v)$ from all $X's$ child nodes, V.
    Receive $\pi(x)$ from all $X's$ parent nodes, U.
    Compute $\lambda_X(u)$ for all $X's$ parent nodes, U.
    Compute $\pi_v(x)$ for all $X's$ child nodes, V.
  end for
end for
for all non-leaf node $Z_i \in P$ **do**
  Compute $P(Z_i = 1|evidence)$ (i.e., the likelihood of Z_i)
end for

Algorithm 4 Attack intention recognition

for all non-leaf nodes $Z_i \in P$ **do**
  if $P(Z_i = 1|evidence)$ is the maximum or $P(Z_i = 1|evidence) > threshold$ **then**
    select Z_i as potential upcoming attack
  end if
end for

As shown in Algorithm 4, in the final selection of possible attack goal(s) and intention(s), we can either select the node(s) that has the maximum belief value or the one(s) whose belief value is over a threshold.

CHAPTER IX

EXPERIMENTS AND PERFORMANCE EVALUATION

9.1 The Grand Challenge Problem (GCP)

To evaluate the effectiveness of our alert correlation mechanisms, we applied our correlation algorithms to the data sets of the Grand Challenge Problem (GCP) version 3.1 provided by DARPA's Cyber Panel program [25, 36]. In this section, we describe and report our experiment results.

GCP version 3.1 is an attack scenario simulator. It can simulate the behavior of security sensors and generate alert streams. GCP 3.1 includes two innovative worm attack scenarios to specifically evaluate alert correlation techniques. 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. 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. GCP alerts are in the Intrusion Detection Message Exchange Format (IDMEF) defined by IETF [35].

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 Section 3.2.

To better understand the effectiveness of our correlation system, we have defined two performance measures, *true positive correlation rate* and *false positive correlation rate*.

$$True\ positive\ correlation\ rate = \frac{\sharp\ of\ correctly\ correlated\ alert\ pairs}{\sharp\ of\ related\ alert\ pairs} \quad (20)$$

and

$$False\ positive\ correlation\ rate = \frac{\sharp\ of\ incorrectly\ correlated\ alert\ pairs}{\sharp\ of\ correlated\ alert\ pairs} \quad (21)$$

In Eq.(20), *related alert pairs* represents the alerts that have cause-effect relationship. In Eq.(21), *correlated alert pairs* refer to the correlation result output by a correlation system.

True positive correlation rate examines the completeness of alert correlation techniques. It measures the percentage of related alert pairs that an correlation system can identify. It is analogous to *true positive rate* or *detection rate* commonly used in intrusion detection.

False positive correlation rate measures the soundness of an alert correlation system. It examines how correctly the alerts are correlated. It is analogous to *false positive rate* used in intrusion detection.

In our experiments, we refer to the documents with the ground truth to determine the correctness of the alert correlation. 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 A, its corresponding time series variable $\tilde{A}$ has a size of 7,200 instances, i.e., $\tilde{A} = \{\tilde{a_0}, \tilde{a_1}, ..., \tilde{a_{7,199}}\}$.

9.1.1 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 procedure of alert correlation is shown as follows.

First, **alert aggregation**. We conducted 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 applied the *Ljung-Box* statistical test [54] with significance level $\alpha = 0.05$ to all hyper alerts in order to identify background alerts. In scenario I, we identified 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 Section 3.2. In this step, we set the threshold $\beta = 0.6$. Alerts with priority scores above β were regarded as important alerts and were selected as target alerts of which we had much interest. In this step, we identified 15 hyper alerts whose priority values are above the threshold

Fourth, **alert correlation**. When applying correlation algorithms, we correlated 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 7.1, we have three steps in correlating alerts. First, we applied Bayesian-based correlation engine on hyper alerts and discover the correlated alert pairs. Figure 16 shows the correlation results related to the hyper alerts that we identified as most interested alerts. Second, we applied causal discovery-based correlation engine to alerts that have not been identified to be correlated with others in the first step. Third, we applied GCT-based correlation algorithm to further correlate alert pairs which have not been correlated after prior two steps. Figure 17 shows the correlation results after the three-step correlation process. The dotted line in Figure 16 and Figure 17 represent false positive correlation. The correlation probability or confidence of each alert-pair is associated with the edge in the correlation graph. In Eq. (9), ω equals 0.3 and t equals

0.6.

Fifth, **attack path analysis**. As discussed in Section 7.4, 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 17, 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 8 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 performed attack strategy analysis by abstracting the scenario graphs. Instead of using hyper alerts representing each node, we used the corresponding attack class (e.g., *DoS* and *Access Violation*) to abstractly present attack strategies. While analyzing attack strategy, we focused on each target and abstracted the attacks against the target. Figure 18(a) shows the high-level attack strategy on the Plan Server extracted from attack scenario graphs shown in Figure 17. From Figure 18(a), we can see that the attacker uses a covert channel (indicated by *Connection Violation*) to export

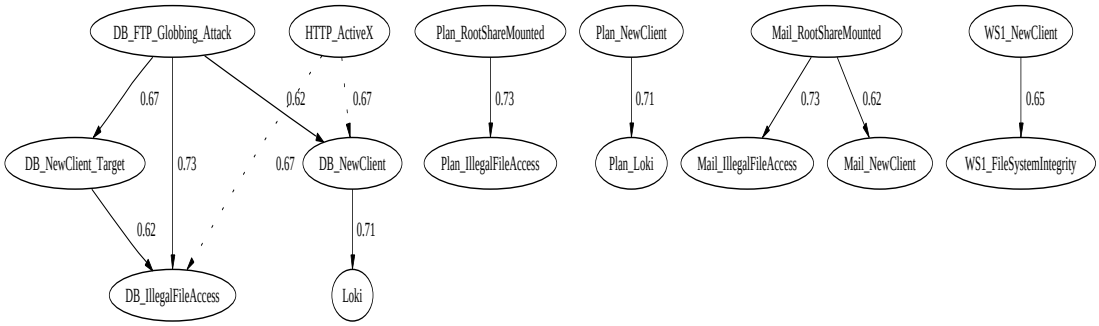


Figure 16: The GCP scenario I: The correlation graph discovered by Bayesian-based approach.

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 18(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 classes *Connection Violation*, *Access Violation* and *Exfiltration* indicates the attack continuously accesses file, exports data and downloads the malicious code.

Table 8: Ranking of paths from node *DB FTP Globbing Attack* to node *DB NewClient*. $P = P(DB\ FTP\ Globbing\ Attack)$

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

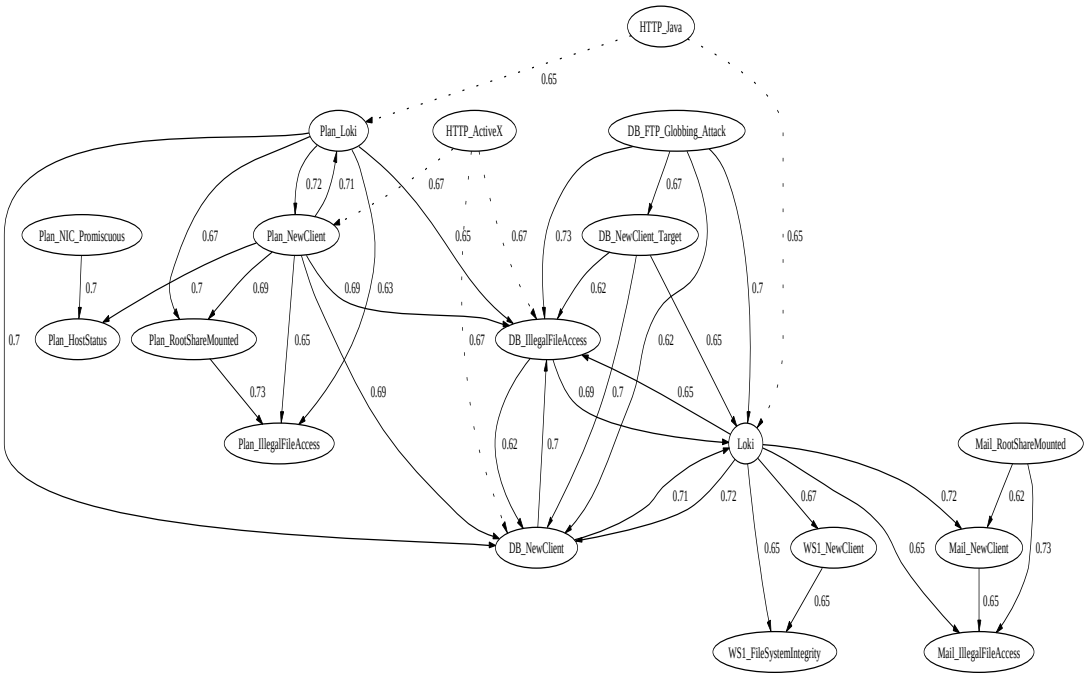
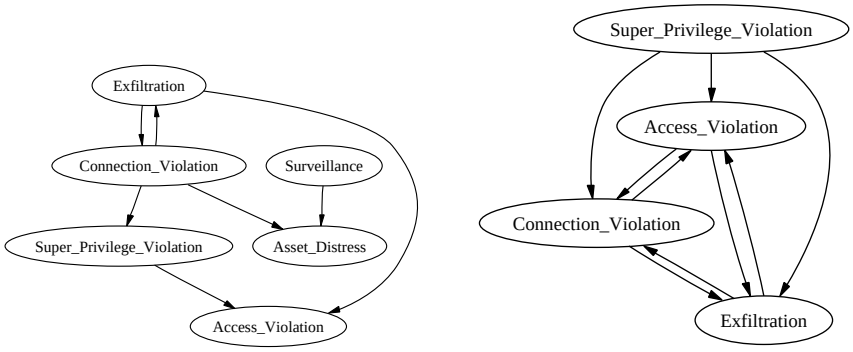


Figure 17: The GCP scenario I: The correlation graph discovered by the integrated approach.



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

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

Figure 18: GCP I: Attack strategy graph

9.1.2 Discussion on GCP Scenario I

Applying our integrated correlation mechanism can discover more attack step relationships than using a single approach. Figure 16 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 17 shows that we can discover more attack relationships after applying causal discovery-based and GCT-based correlation methods. Using complementary correlation engines enable us to link isolated correlation graphs output by Bayesian-correlation engine. The reason is that our statistical and temporal-based correlation mechanisms correlate attack steps based on the analysis of statistical and temporal patterns between attack steps. For example, the loop pattern of attack transitions among attack *DB_NewClient*, *DB_IllegalFileAccess* and *Loki*. This correlation engine does not rely on prior knowledge. By incorporating the three correlation engines, in this experiment, we can improve the true positive correlation rate from 95.06% (when using GCT-based correlation engine alone [69]) to 97.53%. False positive correlation rate is decreased from 12.6% (when using GCT-based correlation engine alone [69]) 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).

9.1.3 GCP Scenario II

In GCP scenario II, there are around 22,500 raw alerts. We went through the same process steps as described in Section 9.1.1 to analyze and correlate alerts.

After alert aggregation and clustering, we got 1,800 hyper alerts. We also use the

same network enclave used in Section 9.1.1 as an example to show our results in the GCP Scenario II.

In this network enclave, there are a total of 387 hyper alerts. Applying the *Ljung-Box* test to the hyper alerts, we identify 273 hyper alerts as the background alerts. In calculating the priority of hyper alerts, there are 9 hyper alerts whose priority values are above the threshold $\beta = 0.6$, meaning that we have more interest in these alerts than others.

As described in Section 7.1, we apply three correlation engines sequentially to the alert data to identify the alert relationship. For example, we select two alerts, *Plan_Service_Status_Down* and *Plan_Host_Status_Down*, as target alerts, then apply the GCT algorithm to correlating other alerts with them.

Table 9: Alert Correlation by the GCT on the GCP Scenario II: Target Alert: *Plan Service Status Down*

$Alert_i$	Target Alert	GCT Index
Plan_Registry_Modified	Plan_Service_Status_Down	20.18
HTTP_Java	Plan_Service_Status_Down	17.35
HTTP_Shells	Plan_Service_Status_Down	16.28

Table 10: Alert Correlation by the GCT on the GCP Scenario II: Target Alert: *Plan Server Status Down*

$Alert_i$	Target Alert	GCT Index
HTTP_Java	Plan_Server_Status_Down	7.73
Plan_Registry_Modified	Plan_Server_Status_Down	7.63
Plan_Service_Status_Down	Plan_Server_Status_Down	6.78
HTTP_RobotsTxt	Plan_Server_Status_Down	1.67

Table 9 and Table 10 show the corresponding GCT correlation results. In the tables, we list alerts whose *GCI* values have passed the F-test. The alerts *Plan_Host_Status* and *Plan_Service_Status* are issued by a network management system deployed on the network.

Figure 19 shows the correlation graph of *Plan Server*. The solid lines indicate the correct alert relationship while dotted lines represent false positive correlation. Figure 19 shows that *Plan_Registry_Modified* is causally related to alerts *Plan_Service_Status_Down*

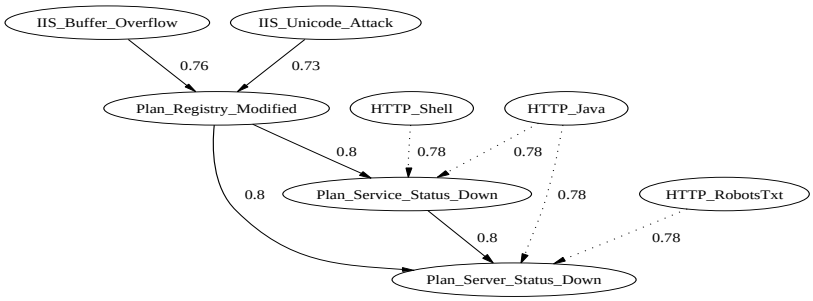


Figure 19: The GCP Scenario II: Correlation graph of the plan server

and *Plan_Server_Status_Down*. The GCP document verifies such relationship. The attacker launched *IIS_Unicode_Attack* and *IIS_Buffer_Overflow* attack against the Plan Server in order to traversal the root directory and access the plan server to install the malicious executable code. The Plan Server's registry file is modified (alert *Plan_Registry_Modified*) and the service is down (alert *Plan_Service_Status*) during the daemon installation. Alert *Plan_Host_Status_Down* indicates the "down" state of the plan server resulted from the reboot initiated by the malicious daemon. Plan server's states are affected by the activities of the malicious daemon installed on it. The ground truth described in the GCP document also supports the causal relationships discovered by our approach. In this experiment, the true positive correlation rate is 94.25% (vs. 93.15% using GCT-engine alone [69]) and false positive correlation rate is 8.92% (vs. 13.92% using GCT-engine alone [69]).

Table 11: Ranking of paths from node *IIS Buffer Overflow* to node *Plan Server Status Down*. $P = P(IIS\_Buffer\_Overflow)$

Order	Nodes Along the Path	Score
Path 1	IIS_Buffer_Overflow → Plan_Registry_Modified → Plan_Server_Status_Down	P* 0.61
Path 2	IIS_Buffer_Overflow → Plan_Registry_Modified Plan_Service_Status_Down	P*0.49

For nodes with multiple paths in the correlation graph, we can also perform path analysis quantitatively. For example, there are two paths connecting node *IIS_Buffer_Overflow* and node *Plan_Server_Status_Down* as shown in Figure 19. We can rank these two paths according to score of the overall likelihood, as shown in Table 11.

9.1.4 Discussion on GCP Scenario II

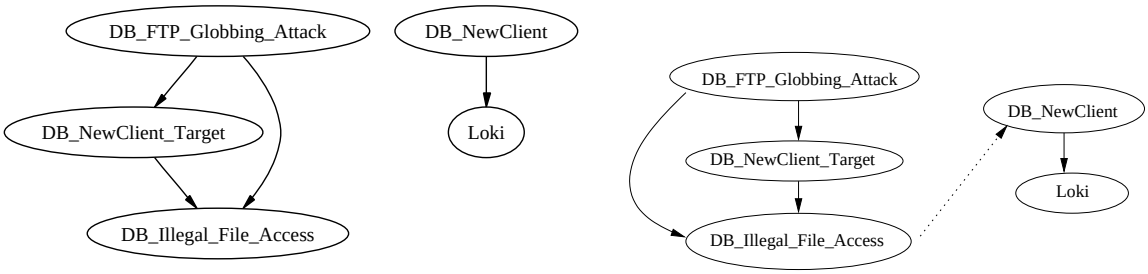
Similar to our analysis in GCP Scenario I, our integrated correlation engine enables us to detect more cause-effect relationship between alerts. For example, in Figure 19, if using knowledge-based correlation engine, we can only detect the causal relationship between alerts *IIS_Buffer_Overflow* and *Plan_Registry_Modified*, as well as between alerts *IIS_Unicode_Attack* and *Plan_Registry_Modified*. With complementary temporal-based GCT alert correlation engine, we can detect other cause-effect relationship among alerts. For example, GCT-based correlation engine detected causality between a security alert (e.g., *Plan_Registry_Modified*) and an alert output by the network management system (e.g., *Plan_Server_Status_Down*). In practice, it is difficult to detect such causality between security activity and network management fault using a knowledge-based correlation approach, unless such knowledge has been priory incorporated to the knowledge base.

Compared with GCP Scenario I, GCP Scenario II is more challenging due to the nature of the attack. Our correlation result in the GCP Scenario II is not comprehensive enough to cover the complete attack scenarios. By comparing the alert streams with the GCP document, we notice that many malicious activities in the GCP Scenario II are not detected by the IDSs and other security sensors. Therefore, some intermediate attack steps are missed, which is another challenge in GCP Scenario II.

Our approach depends on alert data for correlation and scenario analysis. When there is a lack of alerts corresponding to the intermediate attack steps, we cannot construct the complete attack scenario. In practice, IDSs or other security sensors can miss some attack activities. One solution is to apply attack plan recognition techniques that can partially link isolated attack correlation graphs resulted from missing alerts.

9.1.5 Attack Plan Recognition and Prediction

In the GCP Scenario I, there are multiple network enclaves in which attacks are conducted separately. We select a network enclave as an example to show the process of scenario



(a) Two isolated scenarios (b) Correlated scenarios

Figure 20: Correlation of isolated scenarios

correlation and attack prediction.

Figure 20(a) shows an example of two isolated attack scenarios derived from low-level alert correlation, where *DB_FTP_Globbing_Attack* represents an buffer over flow attack against the database server, *DB_NewClient_Target* indicates an suspicious incoming connection to the database server from another server, *DB_Illegal_File_Access* represents the illegal access (write or read) to the database server, *DB_NewClient* indicates a suspicious outbound connection from database to an external host, and *Loki* means a suspicious data export via covert channel. In this case, we use the attack plan as defined in Figure 11. The corresponding causal network is shown in Figure 12. According to Figure 12, we can see that the alert sets {*DB_FTP_Globbing_Attack*, *DB_NewClient_Target*, *DB_Illegal_File_Access*} are corresponding to the attack steps with goals to get access to the server and get the data directly from the host, i.e., the attacker first applies buffer over flow attack against the database server, then sets up a covert channel to the host and export malicious code that is used to access to the database server to get the data. Alert sets {*DB_NewClient*, *Loki*} are attack steps that aim to set up covert channel and export confidential data to the outside. Figure 12 also shows that these two sets of alerts have an *indirect relationship* but the *same eventual goal* that is to steal the data from the database server and export it to the external. Therefore, applying the scenario correlation technique as described in Section 8.3, we can correlate these two scenarios as one integrated scenario, i.e., they are correlated with the same eventual goal, as shown in Figure 20(b), and group

them together as one evidence set.

The advantage of correlating isolated scenarios is that we can accumulate more comprehensive evidence that can be used for further analysis, e.g., likelihood evaluation of each subgoal or final goal and attack prediction.

Table 12: Likelihood evaluation of sub-goals and final goal with different evidence. Denote e_1: *DB FTP Globbing Attack*, e_2: *DB NewClient Target*, e_3: *DB Illegal File Access*, e_4: *DB NewClient*, e_5: *Loki*, $subgoal_1$: *Get confidential data*, $subgoal_2$: *Export confidential data*, $goal$: *Steal and export confidential data*.

Evidence set	$P(subgoal_1 = 1\|$ $evidence)$	$P(subgoal_2 = 1\|$ $evidence)$	$P(goal = 1\|$ $evidence)$
e_1	0.58	0.55	0.56
e_1, e_2	0.58	0.71	0.63
e_1, e_2, e_3	0.78	0.71	0.74
e_1, e_2, e_3, e_4	0.78	0.81	0.77
e_1, e_2, e_3, e_4, e_5	0.78	0.85	0.81

Also using database server as an example, based on the integrated evidence set, we apply probabilistic inference to the causal network as shown in Figure 12 to compute the likelihood of each subgoal and final goal. Table 12 shows the assessment of likelihood of some subgoals and final goal based on the evidence set. In Table 12, we show the probability result of two subgoals, i.e., *Get_confidential_data* and *Export_confidential_data*, and the final goal, i.e., *Steal_and_export_confidential_data*. We can see that probabilities of the success of subgoals and the final goal increases with the support of incoming evidence corresponding to the attack steps aimed to get data from the database server and export data to the external.

The likelihood of each node at causal network based on on-going evidence can also be used to predict the attacks. For example, after getting the evidence of *DB_FTP_Globbing_Attack*, the probability of the subgoal *Get_data_from_Server_directly* (as shown in Figure 12) is increased and equals 0.67. Therefore, we expect a future attack that enables the attacker to access the file stored in the database server. For another example, when we get the evidence of *DB_NewClient*, the likelihood of *Transfer_data_via_covert_channel* (as shown in

Figure 12) is computed as 0.71 that means it is quite likely that we will see another attack with which the attacker can export data via the covert channel in the future. In the GCP Scenario I, the attacker did launch the attack to access the confidential data stored in database server (indicated by the alert *DB_Illegal_File_Access*) after getting the root access to the database server, and also transferred the stolen data to the external (indicated by the alert *Loki*) after setting up the covert channel (indicated by alert *DB_NewClient*).

In our approach, one of the most important components is the library of attack plans (defined as attack trees). It is the basis for automatically correlating isolated attack scenarios at a higher level and conducting probabilistic inference for attack prediction. It is true that there exists a limitation in this approach due to the (limited) library of attack plans. If the attack strategies are beyond the definition of attack plans, we cannot automatically correlate isolated scenarios or make an inference on the future attacks based on existing attack plan library. Such a task requires the involvement of security experts. However, we argue that, in practice, the plan library can be defined as comprehensively as possible by security experts with their knowledge of attacks and attack strategies, as well as the understanding of networks and systems under protection and the mission goals. The attack plan library can be expanded or re-defined with the new knowledge of attacks or attack scenarios. Therefore, we believe that our approach is practical and has the potential to provide security operators a way to automatically correlate isolated attack scenarios and predict future attacks based on the observed evidence and networks under protection.

9.1.6 Discussion on Statistical and Temporal Correlation Engines

In our alert correlation system, we have designed three correlation engines. The Bayesian-based correlation aims to discover alerts that have direct causal relationship by evaluating and checking the three properties of cause-effect alerts as described in Section 1.4.1. Specifically, this correlation engine uses predicates to represent attack prerequisite and consequence, applies probabilistic reasoning to evaluating the property of *preparation-for*

relationship between alerts. It applies time constraints to testing if the alert pair candidate conforms to the property of *sequential relationship*, and uses the pre-defined probability table of attack step transitions to evaluate the property of *statistical one-way dependence* between alerts under correlation. Alert pairs that have matched these three properties are identified as having direct causal relationship.

In order to discover alerts that have no *known* direct causal relationship, we have also developed two statistical and temporal-based correlation models to discover *novel* and *new* attack transition patterns. The development of these two correlation techniques is based on the hypothesis that attack steps can still exhibit statistical dependency patterns (i.e., the third property of cause-effect alerts) or temporal patterns even though they do not have an obvious or *known* preparation-for relationship (i.e., the first property of cause-effect alerts). Therefore, these two correlation engines aim to discover correlated alerts based on statistical dependency analysis and temporal pattern analysis with sequential time constraints (i.e., to ensure the conformity to the second property of cause-effect alerts). More formally, these two engines actually perform *correlation analysis* instead of a direct causality analysis because the preparation-for relationship between alerts are either indirect or unknown.

In theory, causality is a subset of correlation [38], which means that a causally related alert pair is also correlated, however, the reverse statement is not necessarily true. Therefore, the correlation output is actually a super set of correlated alerts that can include the causally related alert pairs as well as some correlated but non-causally related alerts. Our goal is to apply these two correlation engines to identifying the correlated alerts that have strong statistical dependencies and temporal patterns, and also conform to the sequential time constraint property. We present these correlated alert candidates to the security analysts for further analysis.

As an extra experiment, we applied GCP data sets to causal discovery-based correlation engine and GCT-based correlation engine only in order to test if the output of these two

correlation engines can include the causally related alert pairs identified by Bayesian-based correlation engine. Our experiment results have shown that the correlated alerts identified by causal discovery-based correlation engine and GCT-based correlation engine have included those causally related alerts discovered by Bayesian-based correlation engine. In practice, we still use Bayesian-based correlation engine to identify causally related alerts in order to decrease the false positive correlation rate.

However, it does not necessarily mean that those two correlation engines (i.e., casual-discovery and GCT-based engines) can discover all the correlated alerts that have strong statistical and temporal patterns because of their limitations.

As described in Section 5.2, causal discovery-based correlation engine assumes that causality between variables can be represented by a causal Bayesian network that has a DAG structure. The statistical dependency between variables can be measured, for example, by mutual information. As described in Algorithm 1, causality direction among variables are identified by the assumption of causal Markov condition (i.e., a node X is independent with other nodes (except its direct effect nodes) given X's direct cause node) and the properties of *V-structure* as described in Section 5.2.2.

Due to the assumptions and properties used by causal discovery theory, in the process of alert correlation, the causal discovery-based correlation engine can result in cases that the causality direction cannot be identified among dependent alerts.

For example, for three variables A, B and C, after applying mutual information measures, we have got a dependency structure as $A - B - C$, which means A and B, B and C are mutually dependent respectively, A and C are mutually independent. If we apply conditional mutual information measure to A, B and C and get the result that A and C are conditionally independent given the variable B, then, without any other information, the causal discovery-based correlation engine actually cannot identify the causality among these three variables. In fact, with the above statistical dependency information, we can have the following three different causality structures, i.e., $A \rightarrow B \rightarrow C$, $A \leftarrow B \leftarrow C$

and $A \leftarrow B \rightarrow C$. These three causality structures have the same statistical dependency properties if no other information has been provided or extra causality has been identified (e.g., A or B or C has some dependency with another variable D, etc.). For the simplest dependency structure, i.e., $A - B$, without any extra information, causal discovery-based algorithm cannot identify the causality direction between A and B either.

By contrast, GCT-based correlation engine performs pairwise statistical dependency analysis and identify corresponding pairwise dependency direction. However, GCT-based correlation algorithm also incorporates the temporal information in the process of correlation. In particular, the GCT-correlation engine has the limitation of identifying correlated alert pairs whose time intervals have a *loose temporal pattern* (as defined in Definiton 2) even though they may have a strong statistical dependency pattern.

In summary, considering the strength and limitations of causal discovery-based and GCT-based correlation engines, from the perspective of statistical dependency and temporal pattern analysis, we can have a good correlation performance in identifying alerts that have a strong statistical dependency and strong temporal pattern because these two correlation engines can complement and enhance each other in this correlation space. If alerts that have a strong statistical dependency pattern but a loose temporal pattern, the correlation performance may be weak because GCT-based correlation engine has limitations in the loose temporal pattern space and causal discovery-based correlation engine also has its own limitations in the causality identification.

9.2 Experiments on Backbone Data

To evaluate our alert correlation mechanisms, we applied our algorithms to the alert data sets collected on our department backbone network. The backbone is an OC-48 (2Gbps) link, and our span is a 1Gbps link. We normally get an average traffic at around 170Mbps-200Mbps with bursts up to 400Mbps-500Mbps.

In practical security operations, there is no ground truth document that is usually only

available in a self-managed test environment or tests using simulation data sets. In order to have more accurate evaluation, we chose the way to deploy various security sensors, including network-based IDSs and host-based intrusion detection agents, to cross check and verify alerts. Office of Information Technology (OIT) at Georgia Institute of Technology have also helped us evaluate the validity of alerts by providing extra and more evidence of malicious network activities. In the experiments, we also set up a few servers (e.g., a web server) and desktops to which we have administration privilege and full access.

In the process of setting up our empirical ground truth, the main task is to filter out false positive alerts and alerts that are non-relevant with the protected hosts or network (e.g., a Microsoft IIS-related attack targeting at a Linux-based web server). In the step of filtering false positive alerts, we applied evidence cross checking to identifying the false positive alerts. In other words, we used alerts or evidence output by other security sensors to cross check the validity of an alert. In particular, for an alert generated by a security sensor (e.g., an IDS), we checked if there were any similar alerts output by other security sensors or if there were any alerts or evidence (e.g., audited data recorded at the victim machine) corresponding to the impact of the attack. For example, when a network-based IDS output a buffer overflow alert targeting a specific process running on the target host, and if the host-based IDS installed on the target machine also generated an alert representing an abnormal running of that process or other abnormal activities (e.g., illegal file access) corresponding to the evidence of the attack impact, then we could enforce the validity of that buffer overflow alert. In the step of identifying non-relevant alerts, we compared reported alerts with the configurations of target hosts to see if the corresponding vulnerabilities existed on the victim machines. In our experiments, we had the knowledge of the configurations of hosts in our network (e.g., OS, services running on the machine, etc.). We also had the knowledge of the alerts (e.g., the vulnerabilities or OS that the corresponding attack aims to compromise) with reference to the CVE (Common Vulnerabilities and Exposures) [24] and documents of corresponding IDSs installed on our network and hosts. When the target

hosts did not have corresponding vulnerabilities of the reported attack, we regarded the reported alert(s) as non-relevant. Using the above methods, we set up our empirical ground truth and evaluated the performance of our algorithms.

In the experiments, we deployed Snort 2.3.3 [78] and 0.9 [5, 63], two open source network IDSs, on the backbone to monitor the traffic and output alerts. Snort is a signature-based network IDS using string pattern matching to identify intrusions. Bro analyzes and filters network traffic into a series of events describing network activities, and executes scripts that contain site-specific intrusion detection rules or security policies to identify attacks. Bro also includes and uses attack signatures expressed as regular expressions to detect known attacks or access to known vulnerabilities. Bro has the facility to store and use the information of past activity to analyze new activity, and Bro's specific language can also allow it to understand the context of the signature. These capabilities have helped Bro reduce the number of false positives.

We fine-tuned Snort and Bro using network traffic traces (including attack traffic) that we have collected and analyzed before.

In addition to network-based IDS, we have also deployed Cisco Security Agent [17], a host-based intrusion detection agent, on critical servers and some end-user desktops.

Knowledge of alerts is referred to Snort, Bro, Cisco Security Agent documents and CVE (Common Vulnerabilities and Exposures) [24], a well known project to standardize the names of publicly known vulnerabilities.

For performance evaluation, we use *true positive correlation rate* and *False positive correlation rate* as defined in Eq.(20) and Eq.(21) respectively. In the experiment, we used our empirical ground truth to determine if the correlation output made sense or not.

We analyzed alerts collected within a time window of 24 hours, and set a time slot as 1 minute that was used to aggregate raw alerts, generate input data set for alert time series variables (as described in Section 6.5.1) and causal discovery-based alert correlation engine (as described in Section 5.3).

The size of daily alert data varies depending on the traffic, in particular, attack traffic.

Table 13: An example of a weekly alert numbers

Day	Total Alert Numbers
Day 1	640,553
Day 2	602,743
Day 3	637,127
Day 4	647,456
Day 4	621,837
Day 5	634,182
Day 6	587,831
Day 7	571,913

Table 13 shows an example of total alert numbers in a week. Among the attack categories, exploit attack is the majority. Web servers have attracted many malicious activities.

In this section, we present some results using one of our daily alert data set.

The procedure of alert correlation is shown as follows.

First, **alert aggregation**. In this data set, the IDS output 621,837 raw alerts. We conducted raw alert aggregation and clustering in order to have aggregated hyper alerts. After alert fusion and clustering, we had around 63,457 hyper alerts.

Second, **alert noise detection**. We applied the *Ljung-Box* statistical test [54] with significance level $\alpha = 0.05$ to all hyper alerts in order to identify background random alerts. After this step, we identified around 14,300 hyper alerts as background random alerts.

Third, **alert prioritization and focus identification**. The next step is to select the alerts with high priority values as the target alerts. The priority computation is described in Section 3.2. In this step, we identified some "inland" network segments and some important servers (e.g., web servers, ftp servers and mail servers, etc.) and self-managed desktops as our focused protection network and targets. In computing the priority, we set the threshold $\beta = 0.6$. Alerts with priority scores above β were regarded as important alerts and were selected as target alerts of which we have much interest. In this step, we identified 24,130 hyper alerts whose priority values are above the threshold for further analysis.

Fourth, **alert correlation**. As described in Section 7.1, we first applied our domain knowledge-based Bayesian correlation engine to discovering direct cause-effect relationship between alert pairs, then applied causal discovery-based correlation mechanism and finally used GCT-based correlation algorithm to further identify alert causal relationship based on statistical and temporal analysis.

With alert correlation result, from a protection (or target) point of view, we can investigate several specific scenarios that reflect various nature of attack situations. In particular, (1) Attacks come from the same source to the same target with the same attack class. This allows us to detect a scenario that an attacker is launching a series of attacks against a specific service of a server. (2) Attacks come from the same source and target the same destination. This situation is intended to detect the situation that an attacker launches series of attacks against the various services available on the target machine. (3) Attacks with the same class target at the same target. For example, a DDoS attack belongs to this situation. (4) Attacks target at the same host. It can give us an overview attack situation on the target.

Figure 21 shows a correlation graph in which alerts are from an external host $H1$ to an internal host $H2$ that has some vulnerabilities. The correlation result is based on the analysis of Bayesian-based correlation engine. In this figure, we can see the target host $H2$ in our network got a series of attacks from an external host $H1$. The graph shows the attack scenarios. The attacker first used the attack tool $NMAP$ to scan and get service information of the host $H2$, then attempted two major buffer overflow attacks (i.e., $RPC\_CMSD\_TCP\_Overflow$ and $FTP\_Command\_Overflow$) to get access to $H2$. The buffer overflow attack activities were reported by network-based IDSs and our security agent installed on the host $H2$ also reported alarms related to abnormal process running. Figure 21 also shows two DDoS attack-related attempts launched from $H1$ to $H2$, i.e., $DDOS\_Trin00\_Master\_to\_Daemon\_default\_password\_attempt$ and $DDOS\_mstream\_client\_to\_handler$. These two alerts represent the communication between the master and the handler (daemon or zombie) in a DDoS attack. The alerts imply

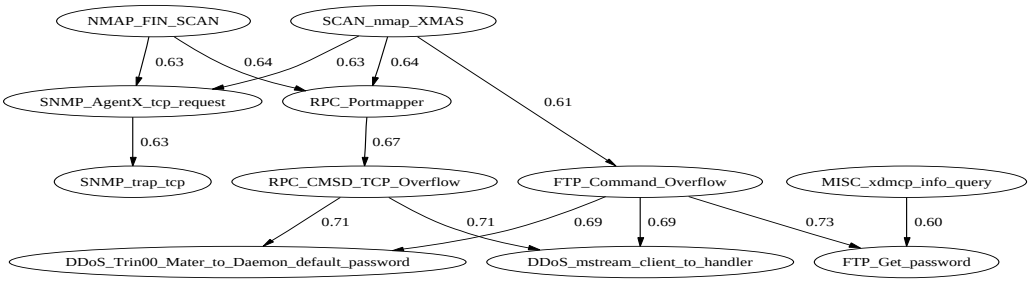


Figure 21: An example of alert correlation graph constructed by Bayesian-based correlation engine

that the host $H2$ was probably being used as a handler to participate in a DDoS attempt. These two DDoS related client-handler control attack alerts were also found in other days. It is quite likely that the attacker was trying to compromise and control our host and make them as a DDoS handler.

Table 14 shows a snapshot of the correlation result at the web server. In particular, we show the alert causal relationship identified by either causal discovery algorithm or GCT-based correlation algorithm and corresponding correlation score output by the engines. From the figure, we can see that various attacks had been launched targeting the web server in order to get access to it, e.g., alert $web\_php\_viewtopic\_php\_access$, and traverse the directory, e.g., alert $web\_IIS\_directory\_transversal\_attempt$. As described in Section 1.3, the major difference between causal discovery algorithm and GCT-based correlation algorithm is that GCT is more focused on temporal analysis while causal discovery method depends on statistical analysis of attack dependence. In addition, according to our observation, web server is a hot target of the daily attack traffic.

In our experiments, we used our empirical ground truth to evaluate the performance of the correlation results. In the data set we presented in this paper, the *true positive correlation rate* is 90.3% and the *false positive correlation rate* is 10.6%.

Fifth, ***scenario correlation***. In this step, we correlated isolated scenarios using plan recognition technique.

Table 14: An example of alert correlation at Web Server. CD: Causal Discovery-based correlation engine; GCT: Granger-Causality-based correlation engine

Cause Alert	Effect Alert	Detected by (correlation score)
http_php_overflow	WEB_MISC_weblogic_tomcat	CD (0.68)
WEB_MISC_robots_txt_access	WEB_MISC_Invalid_HTTP _Version_String	GCT (0.71)
WEB_ATTACKS_perl_execution	WEB_IIS_Directory _transversal	GCT (0.81)
WEB_PHP_viewtopic_php_access	WEB_MISC_weblogic_tomcat	CD (0.81)
WEB_ATTACKS_bin_ls _command_attempt	WEB_CGI_webplus_directory _traversal	CD (0.73)
WEB_MISC_Invalid_HTTP_String	WEB_CGI_calendar_access	CD (0.80)
WEB_MISC_Invalid_HTTP_String	WEB_CGI_webplus_directory _traversal	CD (0.67)
http_inspect_OVERSIZE _REQUEST	WEB_MISC_doc_access _URI_DIRECTORY	GCT (0.68)
WEB_ATTACKS_chmod _command_attempt	WEB_CGI_webplus_directory _traversal	CD (0.86)
WEB_CGI_php_cgi_access	WEB_IIS_Directory _transversal	GCT (0.77)
WEB_CGI_webplus_directory _traversal	WEB_IIS_directory _listing	GCT (0.73)

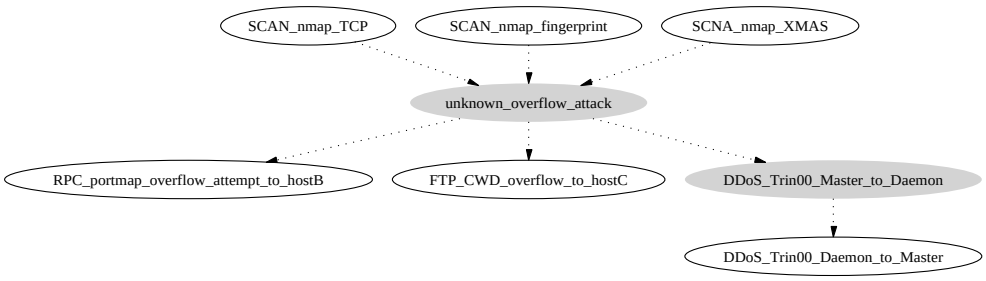


Figure 22: An example of correlating isolated scenarios

Figure 22 shows an example of our approach to correlating isolated alerts (scenarios). In Figure 22, nodes with solid lines represent the alerts output by the IDS. Specifically, alerts *SCAN_nmap_TCP*, *SCAN_nmap_fingerprint*, *SCAN_nmap_XMAS* are attacks

against one of our hosts, denoted as H. Alerts *RPC_portmap_oeverlow_attempt_to_hostB* and *FTP_CWD_overflow_to_hostC* represent two outbound buffer overflow attacks from host H to server B and server C respectively. Alert *DDoS_Trin00_Daemon_to_Master* represents an outbound communication between host H and an external host outside our network. Those alerts are isolated because we have not been able to correlate them after prior steps.

Alerts *RPC_protmap_oeverlow_attempt_to_hostB*, *FTP_CWD_overflow_to_hostC* and *DDoS_Trin00_Daemon_to_Master* actually tell us that host H had been compromised because these attacks were initiated from host H. However, we do not have any alert evidence showing the inbound attacks that have compromised H. In other words, the IDS may have missed detecting the attack that compromise the host H due to some reasons (e.g., the attack is a new attack and the IDS does not know the signature yet). Therefore, we have to hypothesize that there were some intermediate attacks against the host H before it launched those two buffer overflow attacks to host B and C. In addition, having the knowledge that a Master-handler (daemon) communication of a DDoS attack relies on a prior buffer overflow attack on the handler host, we can also hypothesize that there was an attack step that had enabled the *Master* (a host intended to control a distributed set of servers) to install a malicious daemon on the host H before H could further send communication with $Master$. In Figure 22, we use a light grey node to represent the hypothesis attack steps. In practice, such knowledge can be represented and incorporated into knowledge base (e.g., attack plan library) so that we can apply attack plan recognition technique to correlating isolated scenarios (alerts) and identify the attack goal(s). In Figure 22, we use dotted lines to represent the correlation between attack steps.

Predicting attack intention is a very challenging problem in practice. Our current approach relies on knowledge of attack scenarios (e.g., causal network model as shown in Figure 12) and the alert evidence provided by the IDS. In our experiments, we also performed prediction on attacker intentions. For example, in Figure 22, having seen the evidence of

scanning alerts, the prediction confidence on further attacks, in particular, DDoS attack is only 8%. When having alert *DDoS_Trni00_Daemon_to_Master*, the prediction confidence on DDoS attack is increased to 71%. In practice, we have observed there have been some intensive outbound *BAD-Traffic_IP_Proto* attacks from host H.

In our experiments, we made analysis with incorporating alerts output from network-based IDSs (Snort and Bro) that detects malicious activities observed on the network, and host-based security agent to collect security information about local malicious activities on victim hosts. Combining both network-based and host-based IDS alerts have improved and enhanced the attack scenario analysis.

One open problem is the selection of correlation time window. Currently, we analyze and correlate alerts with daily schedule. Theoretically, a large correlation window includes more security alerts that can provide more helpful information for security analysts to identify attack strategies. However, a large correlation window can result in computation cost and bring more noise that can affect the correlation accuracy. Currently, how to scientifically set up an optimum correlation window is still an open problem.

CHAPTER X

CONCLUSION AND FUTURE WORK

This dissertation has described a framework of an alert correlation system to analyze INFOSEC alerts, detect novel attack strategies, recognize attack plans and predict forthcoming attacks.

In this chapter, we summarize the thesis, review the thesis contributions and briefly describe some areas that merit future research.

10.1 Research Summary

Numerous and various cyber attacks require a large scale of deployment of information security systems to ensure the security and reliability of IT infrastructures. The sheer number of low-level or incomplete security alerts motivate the need for an alert correlation system that can help security analysts effectively manage security alerts, identify attack strategies and take a timely actions to avoid further attacks.

We provided a background introduction to intrusion detection and security alert correlation. An examination of existing alert correlation techniques, including a variety of knowledge-based correlation techniques, has revealed that current approaches cannot provide a comprehensive solution to detect attack strategies. Most notably, current knowledge-based correlation techniques cannot meet the demands of detecting novel attack scenarios because they cannot identify attack step transitions that are new or not yet incorporated in the knowledge base.

To meet the needs of detecting novel attack strategies, we have developed an integrated correlation system based on three complementary correlation techniques. Our correlation techniques are developed based on three hypothesis of attack step transitions. (1) The first

hypothesis is that some attack steps have directly related connection, i.e., a prepare-for relationship. For this type of attack steps, we have developed a Bayesian-based correlation engine. It identifies alert causal relationship with a broad range of indicators of attack impacts. This correlation engine can also relax the strict hard-coded pre- and post-condition matching and handle the partial input evidence. (2) The second hypothesis is that some attack steps have statistical dependence patterns. We have developed and presented a statistical-based correlation engine based on causal discovery theory. (3) The third hypothesis is that attack steps have temporal patterns in their time intervals. For this type of attack relationship, we have built a correlation engine based on the Granger Causality Test. The major benefit provided by statistical and temporal correlation engines is that they can discover new attack transition patterns without relying on the domain knowledge.

We also described how to perform attack scenarios analysis by constructing correlation graphs based on 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.

We have also developed an approach to identifying attack plans and predicting upcoming attacks. We have developed a graph-based technique to correlate isolated attack scenarios derived from low-level alert correlation based on their relationship in attack plans. We conducted probabilistic inference to evaluate the likelihood of attack goal(s) and predict potential upcoming attacks based on causal network converted from attack trees.

Finally, we have validated our correlation approach using DARPA Grand Challenge Problem (GCP) data set and alert data collected from a department backbone network on a university campus. The results have shown that our approach can effectively discover novel attack strategies with high accuracy.

10.2 Thesis Contribution

In summary, the contributions of this thesis are:

- **Knowledge-based Probabilistic Correlation Model**. In this research, we have developed a Bayesian-based correlation model to correlate attack steps that have direct causal relationship (i.e., the problem space of $\{$*Alerts with direct causal relationship*$\}$ as shown in Figure 2). This correlation engine uses predicates to represent attack prerequisite and consequence, and applies probabilistic reasoning to evaluating the property of *preparation-for relationship* between alerts based on security states of systems and networks. It applies time constraints to testing if the alert pair candidate conforms to the property of *sequential relationship*, and uses the pre-defined probability table of attack step transitions to evaluate the property of *statistical one-way dependence* between alerts under correlation. Alert pairs that have matched these three properties are identified as having direct causal relationship. Our approach does not rely on the strict pre- and post-condition matching and can also function on the partial correlation evidence.

- **Statistical and Temporal-based Alert Correlation Models**. In addition to the knowledge-based correlation model, in order to discover alerts that have no *known* direct causal relationship, we have developed two statistical and temporal-based correlation models to discover *novel* and *new* attack transition patterns. The development of these two correlation techniques is based on the hypothesis that attack steps can still exhibit statistical dependency patterns (i.e., the third property of cause-effect alerts) or temporal patterns even though they do not have an obvious or *known* preparation-for relationship (i.e., the first property of cause-effect alerts). Therefore, these two correlation engines aim to discover correlated alerts based on statistical dependency analysis and temporal pattern analysis with sequential time constraints (i.e., to ensure the conformity of the second property of cause-effect alerts). More formally, these two engines actually perform *correlation analysis* instead of a direct causality analysis because the preparation-for relationship between alerts are either

indirect or unknown. These two correlation mechanisms do not rely on prior knowledge of attack causal relationship.

- **Causal Discovery Theory-based Alert Correlation Model**. We have developed a causal discovery theory-based alert correlation engine. This correlation engine discovers the strong statistical dependence (in particular, the non-loop one-way dependence) among alerts. This correlation mechanism is based on the assumption that the non-loop one-way dependence among alerts can be represented by a causal Bayesian network that is a directed acyclic graph (DAG). Since this model only applies and depends on probability theory and computation to investigate and identify the statistical one-way dependence, and the temporal pattern of time interval between alerts are not involved in the correlation process, therefore, this correlation model covers the problem subspaces of {*Non-loop structured, Loose temporal relationship*} and {*Non-loop structured, Strong temporal relationship*} as shown in Figure 3.
- **Granger-Causality-based Alert Correlation Model**. We have studied and developed another statistical and temporal-based correlation mechanisms using Granger-Causality Test. This correlation engine investigates and tests the statistical dependency and temporal patterns of alert pairs to identify attack step relationship. This correlation engine discovers the statistical dependency among attack steps that has a loop structure (including mutual dependence and one-way dependence that forms a dependency loop in the attack step dependency graph), i.e., the problem subspace of {*Loop structured, Strong temporal relationship*} as shown in Figure 3. In addition, since this correlation engine performs pairwise correlation, it can also complement causal discovery-based correlation engine in the problem subspace of {*Non-loop structured, Strong temporal relationship*}, as shown in Figure 3, to identify the non-loop dependency pattern missed by causal discovery-based correlation engine.

- **System Integration and Attack Strategy Analysis**. We integrate three complementary correlation engines to perform alert analysis and correlation. We construct attack scenarios and conduct attack path analysis based on the output of three correlation engines. We evaluate and rank the overall likelihood of various attack paths and provide the security analysts the ones with higher probabilities.

- **Attack Plan Recognition and Intention Prediction**. We have developed techniques applied to attack plan recognition and intention prediction. We have developed a series of techniques to correlate isolated attack scenarios derived from low-level alert correlation, to recognize the attacker's attack plan and intentions as well as make predictions of potential attack intentions based on current observations and analysis.

10.3 Future Work

There are several interesting and important future directions:

- Development of an extra correlation engine to effectively cover a problem subspace. According to our problem space definition as shown in Figure 2 and Figure 3, we have developed a Bayesian-based correlation engine to discover alert pairs that have *direct causal relationship*. For alert pairs without direct causal relationship, we have developed two correlation engines to cover the rest of the problem space. In particular, the correlation engine using causal discovery theory identifies alert pairs that have *strong statistical one-way dependence*. GCT-based correlation engine discovers alert pairs with *strong temporal relationship*. As discussed in Section 9.1.6, causal discovery-based and GCT-based correlation engines have their own limitations in various statistical and temporal pattern space. According to our problem space definition as shown in Figure 3, the existing three correlation engines do not effectively cover the problem subspace of $\{$*Loop structured, Loose temporal relationship*$\}$ as shown in Figure 3. Therefore, it is desired to develop an extra correlation engine to cover that problem subspace.

- Open Questions in Alert Correlation and Attack Plan Recognition. One open problem is the selection of correlation time window. Currently, we analyze and correlate alerts with daily schedule. Theoretically, a large correlation window includes more security alerts that can provide more helpful information for security analysts to identify attack strategies. However, a large correlation window can result in computation cost and bring more noise that can affect the correlation accuracy. How to scientifically set up an optimum correlation window is still an open problem. In attack plan recognition, one open problem is that how we effectively distinguish the deceptive plan and the real goal of the attackers. That is, we need to develop a mechanism to identify and avoid misleading by attackers.

- Early Warning and Attack Prediction. We have done some research on identifying attacker's intentions and early warning based on attack transition models and probabilistic inference. The current approach relies on the domain knowledge and attack evidence. Early warning and attack prediction in a broad scale is very important and challenging, e.g., early warning of a worm spread in the Internet. Currently, early warning of attacks (e.g., worm spread) still relies on human intelligence and intervention (e.g., interaction with underground hacker world about new vulnerabilities or new attack tools). It will be an interesting research area to study and develop new techniques to provide early warning of intrusions, in particular, those zero-day attacks. In addition, there are still much work to do to improve the accuracy of attack predictions based on knowledge models.

- Anomaly Detection. Anomaly detection has been studied in intrusion detection community for many years. Although various techniques have been proposed, the biggest challenge is still how to decrease the false positive rate. It involves the normal profile construction and detection models. Among various anomaly detection fields, I have an interest in application-level anomaly detection. Current approach to application

intrusion detection relies on checking application protocol compliance or using attack signatures for pattern matching. How to effectively detect new attacks against applications beyond the limitation of domain knowledge is a research of interest.

10.4 Conclusions

This dissertation has studied the challenges of alert management and novel attack strategy detection. We have presented a framework of alert correlation to analyze and correlate alerts. We have described three complementary correlation engines to discover attack transition patterns. These three correlation engines analyze attack step relationship from three different perspectives, i.e., direct causal relationship between alerts, statistical dependence and temporal relationship. We have also presented an approach to analyze attack scenarios based on attack path analysis and correlation graphs. Our model of attack plan recognition and prediction enable attack security analysts to correlate isolated attack scenarios, identify attack intentions and have an early warning to forthcoming intrusions. We have examined the effectiveness of our approach using DARPA GCP data sets and live data from our backbone network.

REFERENCES

[1] ALBRECHT, D. and NICHOLSON, A., "Bayesian models for keyhole plan recognition in an adventure game," *User Modeling and User-Adapted Interaction*, pp. 5–47, 1998.

[2] AMOROSO, E., *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Traps, Trace Back, and Response*. Intrusion.Net Books, 1999.

[3] ANDERSON, J. P., "Computer security threat monitoring and surveillance," tech. rep., James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.

[4] BAUER, E., KOLLER, D., and SINGER, Y., "Update rules for parameter estimation in Bayesian networks," in *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, (Providence, RI), pp. 3–13, August 1997.

[5] BRO, "Bro intrusion detection system." http://bro-ids.org, 2005.

[6] CABRERA, J. B. D., LEWIS, L., QIN, X., LEE, W., and MEHRA, R., "Proactive intrusion detection and distributed denial of service attacks - a case study in security management," *Journal of Network and Systems Management*, vol. vol. 10, June 2002.

[7] CABRERA, J. B. D., LEWIS, L., QIN, X., LEE, W., PRASANTH, R. K., RAVICHANDRAN, B., and MEHRA, R. K., "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.

[8] CABRERA, J. B. D. and MEHRA, R. K., "Extracting precursor rules from time series - a classical statistical viewpoint," in *Proceedings of the Second SIAM International Conference on Data Mining*, (Arlington, VA, USA), pp. 213–228, April 2002.

[9] CAINES, P. E. and CHAN, C. W., "Feedback between stationary stastic process," *IEEE Transactions on Automatic Control*, vol. 20, pp. 495–508, 1975.

[10] CARBERRY, S., "Incorporating default inferences into plan recognition," in *Proceedings of the Eighth National Conference on Artifici al Intelligence*, (Boston, Massachusetts), pp. 471–478, 1990.

[11] CHARNIAK, E. and GOLDMAN, R. P., "A probabilistic model of plan recognition," in *Proceedings of the Ninth National Conference on Artifici al Intelligence*, (Anaheim, California), pp. 160–165, 1991.

[12] CHARNIAK, E. and GOLDMAN, R. P., "A bayesian model of plan recognition," *Artificial Intelligence*, vol. 64, pp. 53–79, Novemeber 1993.

[13] CHARNIAK, E. and MCDEMOTT, D., *Introduction to Artificial Intelligence*. Addison Wesley, 1985.

[14] CHENG, J., GREINER, R., KELLY, J., BELL, D., and LIU, W., "Learning bayesian networks from data: An information-theory based approach," *Artificial Intelligence*, vol. vol.137, pp. 43–90, 2002.

[15] CHEUNG, S., LINDQVIST, U., and FONG, M. W., "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.

[16] CHITTURI, R. V., "Distribution of residual autocorrelations in multiple autoregressive schemes," *Journal of American Statistician Association*, vol. 69, pp. 928–934, 1974.

[17] CISCO SYSTEMS, "Cisco security agent." http://www.cisco.com, 2005.

[18] COHEN, I., BRONSTEIN, A., and COZMAN, F. G., "Online learning of bayesian network parameters," *Hewlett Packard Laboratories Technical Report, HPL-2001-55(R.1)*, June 2001.

[19] COHEN, P. R., PERRAULT, C. R., and ALLEN, J. F., "Beyond question answering," in *Strategies for Natural Language Processing* (LEHNERT, W. and RINGLE, M., eds.), pp. 245–274, 1981.

[20] COOPER, G. F., "Probabilistic inference using belief networks is np-hard," Tech. Rep. KSL-87-27, Stanford University, 1988.

[21] COOPER, G. F. and HERSKOVITS, E., "A bayesian method for constructing bayesian belief networks from databases," in *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, 1991.

[22] COVER, T. and THOMAS, J., *Elements of Information Theory*. John Wiley, 1991.

[23] CUPPENS, F. and MIÈGE, A., "Alert correlation in a cooperative intrusion detection framework," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, (Oakland, CA), pp. 202–215, May 2002.

[24] CVE http://www.cve.mitre.org, 2005.

[25] DAPRA CYBER PANEL PROGRAM, "DARPA cyber panel program grand challenge problem (GCP)." http://www.grandchallengeproblem.net/, 2003.

[26] DEAN, T. and WELLMAN, T., *Planning and Control*. Morgan Kaufmann, 1991.

[27] DEBAR, H. and WESPI, A., "The intrusion-detection console correlation mechanism," in *4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, October 2001.

[28] DENNING, D., "An intrusion detection model," *IEEE Transactions on Software Engineering*, vol. 13, February 1987.

[29] DICKEY, D. A. and FULLER, W. A., "Distribution of the estimators for autoregressive time series with a unit root," *Journal of American Statistician Association*, vol. 74, pp. 427–431, 1979.

[30] FRIEDMAN, N., NACHMAN, I., and PEER, D., "Learning bayesian network structure from massive datasets: The sparse candidate algorithm," in *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, 1999.

[31] GEIB, C. W. and GOLDMAN, R. P., "Plan recognition in intrusion detection system," in *DARPA Information Survivability Conference and Exposition (DISCEX II)*, June 2001.

[32] GEVERS, M. R. and ANDERSON, B. D. O., "Representations of jointly stationary stochastic feedback processes," *International Journal of Control*, vol. 33, pp. 777–809, 1981.

[33] GOLDMAN, R. P., HEIMERDINGER, W., and HARP, S. A., "Information modeling for intrusion report aggregation," in *DARPA Information Survivability Conference and Exposition (DISCEX II)*, June 2001.

[34] GRANGER, C. W. J., "Investigating causal relations by econometric methods and cross-spectral methods," *Econometrica*, vol. 34, pp. 424–428, 1969.

[35] GROUP, I. I. D. W., "Intrusion detection message exchange format." http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-09.txt, 2002.

[36] HAINES, J., RYDER, D. K., TINNEL, L., and TAYLOR, S., "Validation of sensor alert correlators," *IEEE Security & Privacy Magazine*, vol. January/February, 2003.

[37] HAMILTON, J., *Time Series Analysis*. Princeton University Press, 1994.

[38] HAYTER, A. J., *Probability and Statistics for Engineers and Scientists*. Duxbury Press, 2002.

[39] HECKERMAN, D., MEEK, C., and COOPER, G. F., "A bayesian approach to causal discovery," in *Book of Computation, Causation, and Discovery, C. Glymour and G. Cooper, editors*, MIT Press, 1999.

[40] HESSE, W., MOLLER, E., ARNOLD, M., WITTE, H., and SCHACK, B., "Investigation of time-variant causal interactions between two eeg signals by means of the adaptive granger causality," *Brain Topography*, vol. 15, pp. 265–266, 2003.

[41] HOSKING, J. R. M., "Lagrange multiplier tests of multivariate time series models," *Journal of The Royal Statistical Society Series B*, vol. 43, pp. 219–230, 1981.

[42] ILGUN, K., KEMMERER, R. A., and PORRAS, P. A., "State transition analysis: A rule-based intrusion detection approach," *IEEE Transactions on Software Engineering*, vol. 21, pp. 181–199, March 1995.

[43] JAKOBSON, G. and WEISSMAN, M., "Real-time telecommunication network management: Extending event correlation with temporal constraints," in *Proceedings of the Fourth IFIP/IEEE International Symposium on Integrated Network Management (IM 1995)*, May 1995.

[44] JAKOBSON, G. and WEISSMAN, M. D., "Alarm correlation," *IEEE Network Magazine*, November 1993.

[45] JOHANSEN, S., "Statistical analysis of co-integration vectors," *Journal of Economic Dynamics and Control*, vol. 1, pp. 321–346, 1988.

[46] JULISCH, K. and DACIER, M., "Mining intrusion detection alarms for actionable knowledge," in *The 8th ACM International Conference on Knowledge Discovery and Data Mining*, July 2002.

[47] KAMINSKI, M., DING, M., TRUCCOLO, W. A., and BRESSLER, S. L., "Evaluating causal relations in neural systems: Granger causality, direct transfer function (dtf) and statistical assessment of significance," *Biological Cybernetics*, vol. 85, pp. 145–157, 2001.

[48] KAUFAMNN, R. K. and STERN, D. I., "Evidence for human influence on climate from hemispheric temperature relations," *Nature*, vol. 388, pp. 39–44, July 1997.

[49] KAUTZ, H. and ALLEN, J. F., "Generalized plan recognition," in *Proceedings of the Fifth National Conference on Artificia l Intelligence*, pp. 32–38, September 1986.

[50] KLIGER, S., YEMINI, S., YEMINI, Y., OSHIE, D., and STOLFO, S., "A coding approach to event correlations," in *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management*, May 1995.

[51] KRUGEL, C., TOTH, T., and KERER, C., "Decentralized event correlation for intrusion detection," in *Proceedings of the 4th International Conference on Information Security and Cryptology*, 2001.

[52] LEE, H., LIN, K. S., and WU, J., "Pitfalls in using granger causality tests to find an engine of growth," *Applied Economics Letters*, vol. 9, pp. 411–414, May 2002.

[53] LEWIS, L., "A case-based reasoning approach to the management of faults in communication networks," in *Proceedings of the IEEE INFOCOM*, 1993.

[54] LJUNG, G. M. and BOX, G. E. P., "On a measure of lack of fit in time series models," in *Biometrika 65*, pp. 297–303, 1978.

[55] LO, C. C. and CHEN, S.-H., "A scheduling-based event correlation scheme for fault identification in communications network," *Computer Communications*, vol. 22, no. 5, pp. 432–438, 1999.

[56] LO, C. C., CHEN, S.-H., and LIN, B.-Y., "Coding-based schemes for fault identification in communication networks," in *Proceedings of Military Communications International Symposium*, (Atlantic City, NJ), Nov. 1999.

[57] MORIN, B. and DEBAR, H., "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.

[58] NING, P., CUI, Y., and REEVES, D. S., "Constructing attack scenarios through correlation of intrusion alerts," in *9th ACM Conference on Computer and Communications Security*, November 2002.

[59] NING, P. and XU, D., "Learnign attack strategies from intrusion alerts," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003.

[60] NING, P., XU, D., HEALEY, C., and AMANT, R. A., "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.

[61] NYGATE, Y. A., "Event correlation using rule and object based techniques," in *Proceedings of the 6th IFIP/IEEE International Symposium on Integrated Network Management*, May 1995.

[62] NYGATE, Y. A. and STERLING, L., "Aspen - designing complex knowledge based systems," in *Proceedings of the 10th Israeli Symposium on Artificial Intelligence, Computer Vision, and Neural Netowrk*, December 1993.

[63] PAXSON, V., "Bro: A system for detecting network intruders in real-time," in *Proceedings of the 7th USENIX Security Symposium*, (San Antonio, TX), 1998.

[64] PAXSON, V., "Experiences learned from bro," *;login: The USENIX Association Magazine*, September 1999.

[65] PEARL, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc, 1988.

[66] PEARL, J., *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[67] PORRAS, P. A., FONG, M. W., and VALDES, A., "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.

[68] PORRAS, P. A. and NEUMANN, P. G., "EMERALD: Event monitoring enabling responses to anomalous live disturbances," in *National Information Systems Security Conference*, (Baltimore MD), October 1997.

[69] QIN, X. and LEE, W., "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.

[70] QIN, X. and LEE, W., "Attack plan recognition and prediction using causal networks," in *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 2004)*, (Tucson, AZ), December 2004.

[71] QIN, X. and LEE, W., "Discovering novel attack strategies from INFOSEC alerts," in *Proceedings of the 9th European Symposium on Research in Computer Security*, (Sophia Antipolis, France), September 2004.

[72] QIN, X. and LEE, W., "Causal discovery-based alert correlation," in *submission to the 21th Annual Computer Security Applications Conference (ACSAC 2005)*, (Tucson, AZ), December 2005.

[73] ROSS, S. M., *Introduction to Probability Models*. Harcourt Academic Press, 7th ed., 2000.

[74] SCHMIDT, C., SRIDHARAN, N., and GOODSON, J., "The plan recognition problem: an intersection of psychology and artificial intelligence," *Artificial Intelligence*, vol. 11, pp. 45–83, 1978.

[75] SCHNEIER, B., *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, August 2000.

[76] SHAFER, G., *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[77] SHEYNER, O., HAINES, J., JHA, S., LIPPMANN, R., and WING, J. M., "Automated generation and analysis of attack graphs," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, (Oakland, CA), May 2002.

[78] SNORT, "Snort–open source intrusion detectin system." http://www.snort.org, 2005.

[79] SPIRTES, P., GLYMOUR, C., and SCHEINES, R., *Causation, Prediction, and Search*. Springer-Verlag NY, Inc., 1993.

[80] STALLINGS, W., *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley, 1999.

[81] SUNSOFT, *SunSHIELD Basic Security Module Guide*. SunSoft, Mountain View, CA, 1995.

[82] TEMPLETON, S. J. and LEVITT, K., "A requires/provides model for computer attacks," in *Proceedings of New Security Paradigms Workshop*, 2000.

[83] VALDES, A. and SKINNER, K., "Probabilistic alert correlation," in *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, October 2001.

[84] WILENSKY, R., *Planning and Understanding*. Addison Wesley, 1983.

VITA

Xinzhou Qin was born in Beijing, China. In 1996, Xinzhou received his Bachelor of Engineering degree in Electrical Engineering and graduated with honors from Beijing University of Technology in Beijing, China. Subsequently, he was awarded his Masters of Science in Electrical Engineering degree from North Carolina State University in Raleigh, NC in 1998. In 2001, Xinzhou was awarded North Carolina Network Initiatives Fellowship. Xinzhou joined College of Computing at the Georgia Institute of Technology in 2001. During his study at Georgia Tech, Xinzhou was an award winner of Cisco Systems Information Assurance Scholarship in 2003.

Xinzhou's research efforts and interests include network and system security, in particular, intrusion detection, alert correlation and security management.