2010

A graph oriented approach for network forensic analysis

Wei Wang
Iowa State University

A graph oriented approach for network forensic analysis

by

Wei Wang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Thomas E. Daniels, Major Professor
Yong Guan
Doug W. Jacobson
Robyn R. Lutz
Steve F. Russell

Iowa State University

Ames, Iowa

2010

To my parents and the four unforgettable years in Ames

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost I owe my deepest gratitude to my advisor Dr. Thomas Daniels, who has supported me throughout my Ph.D study with his patience, enthusiasm and knowledge, while always encouraging me to explore in my own way. Without his guidance this dissertation would not have been completed. I simply could not imagine for a better or friendlier advisor.

I would also like to express my sincere thanks to Dr. Yong Guan, Dr. Doug W. Jacobson, Dr. Robyn R. Lutz and Dr. Steve F. Russell for serving on my committee. Their encouragement and insightful comments are truly valuable to my work.

The research work in this dissertation was supported by the National Science Foundation under Grant No.0627409, the DOI under contract No. NBCHC030107, and by the Iowa State University Information Assurance Center.

I am deeply indebted to my parents Zhenquan Wang and Qijuan Chen for their never ending love and support. I also want to thank my wife Ying Cui, for all her inspiration and companion.

Finally, I would like to thank all my dear friends at Iowa State University. My memories of days in Ames will always be cherished because of you.

ABSTRACT

Network forensic analysis is a process that analyzes intrusion evidence captured from networked environment to identify suspicious entities and stepwise actions in an attack scenario. Unfortunately, the overwhelming amount and low quality of output from security sensors make it difficult for analysts to obtain a succinct high-level view of complex multi-stage intrusions.

This dissertation presents a novel graph based network forensic analysis system. The evidence graph model provides an intuitive representation of collected evidence as well as the foundation for forensic analysis. Based on the evidence graph, we develop a set of analysis components in a hierarchical reasoning framework. Local reasoning utilizes fuzzy inference to infer the functional states of an host level entity from its local observations. Global reasoning performs graph structure analysis to identify the set of highly correlated hosts that belong to the coordinated attack scenario. In global reasoning, we apply spectral clustering and Pagerank methods for generic and targeted investigation respectively. An interactive hypothesis testing procedure is developed to identify "hidden attackers" from non-explicit-malicious evidence. Finally, we introduce the notion of target-oriented effective event sequence(TOEES) to semantically reconstruct stealthy attack scenarios with less dependency on ad-hoc expert knowledge. Well established computation methods used in our approach provide the scalability needed to perform post-incident analysis in large networks. We evaluate the techniques with a number of intrusion detection datasets and the experiment results show that our approach is effective in identifying complex multi-stage attacks.

CHAPTER 1. INTRODUCTION

With the growing reliance on computer networks for information and resource sharing, information security has also been an increasingly serious concern for everyone in today's cyber-world. Goals of information security are defined by three basic principles: (1) Confidentiality: keep secrecy of sensitive information against unauthorized access; (2) Integrity: prevent improper changes to information/resources and protect the trustworthiness; (3) Availability: guarantee legitimate access to information and resources desired. To ensure that information systems are protected against threats in all these three aspects, a number of techniques have been developed and applied in security operations. For instance, firewalls, authentication, encryption and access control systems are designed for *Prevention of attacks*. However, the world of information security is an endless arms race where capability of cyber attackers constantly on the rise. Software vulnerabilities, misconfiguration in security mechanisms and insider threats always opens up possibilities for intruders to break in, making it impossible to provide bullet-proof protection of confidentiality, integrity and availability in practice.

On another front, security sensors such as intrusion detection systems(IDSs) are widely used for *Detection of attacks*. However, current intrusion detection techniques are still far from satisfactory. Firstly, the sheer volume of low-level security alerts generated by these sensors is often overwhelming. Secondly, IDSs inevitably make two types of mistakes: false positives, i.e. the alerts generated do not correspond to real malicious activities; false negatives, i.e. stealthy intrusions are missed without triggering alerts. These problems prevent security administrators from making meaningful analysis and initiate timely response.

In summary, *Prevention* and *Detection* mechanisms are not sufficient to eliminate cyber attack threats. It is important to develop effective post-incident investigation mechanisms

that hold attackers responsible for their malicious actions – this is the realm of network forensics. More formally, network forensics is a subfield of digital forensics which aims to identify suspicious entities in the scene of attack and reconstruct stepwise actions of the attacker by reasoning with intrusion evidence captured from networked environments. The area of network forensics presents a rich problem space, including evidence collection, preservation, analysis and presentation. Our research focus on the analysis phase of network forensics. Specifically, this work is motivated by the following major challenges in network forensic analysis:

1. Forensic analysts are overwhelmed by huge volume of low quality evidence. Evidence from standard security sensors such as IDS alerts contain large amount of "background noise" that comes from the following two sources: 1. false positives triggered by benign activities and 2. irrelevant attacks that are not part of the foreground attack scene of interest. Both of them tend to obscure the important information for identifying attackers and reconstructing the attack scenario.

2. Cyber attacks are becoming increasingly sophisticated. There exist many multi-stage attacks that consist of several evolving phases and span over large number of hosts. However, low level alerts from current security sensors only represent isolated steps in a complex attack scenario. From a forensic analyst's point of view, it is important to intelligently correlate the atomic pieces of evidence to reveal a high-level abstracted view of what occurred in the attack.

3. Most of the existing proposals for multi-stage attack analysis rely on predefined causality rules of attack conditions and consequences, or a priori knowledge of system vulnerabilities. Intuitively the number of casuality rules is very large and difficult to encode. Moreover, they are not able to recognize a coordinated scenario when the relationship between attacks are new. For post-incident investigation it is also infeasible to get accurate vulnerability state of the systems involved at the time of attack. Therefore in practice, it is desirable to have an effective method that is applicable to existing security sensor alarms with less dependency on expert knowledge modeling.

4. There exists no unified approach to investigate non-explicit-malicious activities that could evade the detection of usual security sensors, i.e. actions thought benign by IDS but could have played a role in the coordinated attack. To get a comprehensive view of the attack, it is important to detect the correlation between alerted behaviors and these non-explicit malicious activities. Given evidence of explicit intrusions, analysts need the methodology to generate plausible hypotheses that could be used to guide extended evidence discovery from heterogeneous sensors. Finally, quantitative evaluation is needed to identify potential "hidden attack members" from the non-explicit-malicious evidence.

In contrast to these challenges, current practices in network forensic analysis are still mainly done by manual ad-hoc methods, a time-consuming, error prone and non-scalable process [25]. There exists an increasing need for effective, automated and extensible analysis methods.

1.1 Problem Statement

Major objectives of network forensic analysis can be summarized into two interrelated problems: *Attack group identification* and *Attack scenario reconstruction. Attack group identification* is the task of discovering the group of entities involved in the attack scenario and determining the role of each entity in the group, such as *Attackers, Victims, Stepping Stones* and *Background Attackers. Attack scenario reconstruction* is the process of inferring step-wise actions taken by the attacker to achieve his malicious objective.

This dissertation presents a set of techniques for both *Attack group identification* and *Attack scenario reconstruction.* In essence, we study the problem of

How to effectively analyze forensic evidence from heterogeneous information sources to identify entities and events relevant to multi-stage attack scenarios in a systematic and scalable approach?

More specifically, this high level problem formulation can be extended into the following subproblems:

- How to effectively manage the huge volume of intrusion evidence and filter out irrelevant noise for a focused analysis?

- How to perform post-incident intrusion investigation in a scalable, structured and computationally sound framework?
- How to reconstruct stealthy multi-stage attack scenarios with less dependency on hard coded expert knowledge?
- How to integrate information from heterogeneous information sources into analysis to identify non-explicit malicious activities of attackers?

Our research presents a modular analysis process, which supports the integration of multiple analysis components to combat the above mentioned challenges. The evidence graph model is developed to provide the basis for automated analysis. A set of analysis components are proposed in a hierarchical reasoning framework based on the evidence graph. The components utilize techniques that perform analysis from different views of forensic evidence. More specifically, we integrate domain knowledge based reasoning with well established mathematical methods to form a complete analysis flow. We develop a prototype system that implements the proposed techniques and evaluation is made with various attack traces.

1.2 Research Contributions

The objective of this dissertation is to provide innovative solutions to the challenges identified in the problem statement. Our approach is distinctively different from traditional intrusion investigation methods. By performing analysis on a graph representation of intrusion evidence, we integrate various techniques into a hierarchical analysis framework, including fuzzy reasoning, graph clustering, Pagerank methods as well as semantic attack scenario reconstruction. Major advantages of our approach include:

Effectiveness and Flexibility: In an integrated graph-based reasoning framework we provide methods for different analysis purposes. By using spectral clustering for generic investigation, coordinated attack scenarios buried in predominant amounts of false positives are effectively extracted as significant graph clusters. With the personalized Pagerank algorithm we are able to incorporate prior known watch list for targeted investigation. Moreover, we

apply Pagerank methods in hypothesis testing to identify non-explicit "hidden attackers" for a comprehensive view of the attack. Finally, our cluster evaluation based on the notion of target-oriented effective event sequence(TOEES) effectively filters out false positives and reconstructs stealthy multi-stage attack scenarios.

Applicability: Attack modeling in our graph based analysis process is straightforward. Neither a priori knowledge of host vulnerability nor the labor intensive process of defining hard coded causality dependence rules between exploits is required. Information needed for evidence graph construction, aggregation and cluster evaluation can be retrieved from atomic alerts of current network IDS sensors such as Snort [54]. Compared to traditional causality based reasoning methods, our proposed techniques can be well applied in many post-incident analysis cases where comprehensive vulnerability and causality information are not available.

Scalability: We effectively transforms intrusion evidence into a abstracted graph space for analysis. Spectral clustering and Pagerank methods are performed on the host-centric aggregated evidence graph where each node represents a host level network entity. With existing techniques for high performance spectral graph clustering and Pagerank computation [30], our approach can efficiently handle intrusion investigation in large networks.

Visualization: Our evidence graph model intuitively presents intrusion evidence and analysis results in multiple levels of abstraction. The raw primary evidence graph built from pre-processed evidence is a multigraph that embeds detailed information of exploits. Through evaluation of attack phase and relevant resources, the aggregated evidence graph presents the attack correlation between distinct hosts in a simple undirected graph for structure analysis. Combination of the two views produce a compact and comprehensive visualization to investigators. Evidence graphs can be readily visualized using tools such as GraphViz.

The solutions presented in this dissertation represent contributions to the field of network forensic analysis in the following aspects:

- An extensible evidence graph model that incorporates evidence from heterogeneous evidence sources for analysis and presentation;
- A flexible aggregation scheme for redundancy reduction in raw evidence;

- A hierarchical reasoning framework that provides integrated analysis results from different views:
  - Local reasoning that performs functional analysis with fuzzy inference;
  - Global reasoning that performs structure analysis on abstracted graph space with spectral clustering and Pagerank methods.
- An effective and low-modeling-cost correlation scheme based on target-oriented effective event sequence(TOEES) to reconstruct complex multi-stage attack scenarios.
- A prototype system that implements all the techniques proposed and provides experiment results with well-known public attack datasets.

This research enriches the state-of-art in network forensic analysis and leads to the development of more systematic and automated methods with a mathematically well grounded model.

1.3 Organization of the Dissertation

The reminder of the dissertation is organized as follows: Chapter 2 reviews the background knowledge and related research work. Chapter 3 first presents the architecture of our evidence analysis system, then describes the evidence graph model and hierarchical reasoning framework. Chapter 4 introduces our spectral graph clustering approach for generic intrusion investigation and discusses its robustness to background noise. Chapter 5 describes the application of personalized Pagerank methods for targeted investigation and hidden attacker identification. Chapter 6 presents automated cluster evaluation and scenario reconstruction schemes based on target-oriented effective event sequence(TOEES). Finally, Chapter 7 summarizes the thesis and outlines ideas for future work.

CHAPTER 2. RELATED WORK

Our proposed approach is closely related to research in network forensics, intrusion detection, alert correlation, as well as techniques in graph based security analysis. In the following, we will briefly introduce related work in these areas.

2.1 Network Forensics

Network forensics generally refers to the collection and analysis of network data. It is an important part of digital forensics that focuses on networked environments. In 2001, the first Digital Forensic Research Workshop(DFRWS) produced a working definition of Network Forensics as

"The use of scientifically proved techniques to collect, fuse, identify, examine, correlate, analyze, and document digital evidence from multiple, actively processing and transmitting digital sources for the purpose of uncovering facts related to the planned intent, or measured success of unauthorized activities meant to disrupt, corrupt, and or compromise system components as well as providing information to assist in response to or recovery from these activities [22]."

As suggested by the definition, analyze and correlate evidence is one of the major challenges of network forensics. Currently it is still generally an ad-hoc process heavily relies on the expertise of investigators. To our knowledge, little work has been done in automated network forensic analysis. Well-known forensics investigation tools like EnCase [19] and Safeback [48] focus on capture and analysis of evidence from storage media on a single host. Various software tools such as tcptrace [60], tcpflow [59], flowtools [21] and netcat [36] support network traffic capture and session analysis, but lack the mechanism to help investigators understand and correlate the pieces of evidence they provide. Commercial tools like eTrust network foren-

sics tool [20] and NetDetector [37] captures raw network data and investigate breaches inside cooperate networks, however the analysis procedure is still mostly manual and ad-hoc.

ForNet [49] is a distributed logging mechanism that could be deployed in wide area networks to aid network forensics. ForNet transforms a set of raw network data into a succinct form named a Synopsis that can be stored for a prolonged period of time for forensic analysis. However, the ForNet system is limited to the evidence capture stage and does not address the critical problem of how to automate the process of reasoning about attack scenarios from saved synopses.

Brian Carrier described a model of event reconstruction process for digital crime investigation [4]. Although the work is mostly theoretical, it sheds light on the requirements for developing tools that transform collected evidence into digital crime scenes. Our work is a starting point to explore the technical considerations of automated network forensic analysis.

2.2 Intrusion Detection Systems

Intrusion detection is deemed as the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network. Generally intrusion detection systems could be categorized based on audit source or the detection technique used.

Based on the audit source, IDSs can be classified into two major categories: host-based detection and network-based detection [15]. Host-based IDSs reside on individual hosts and processes audit data generated by the host's operating system and applications. On the other hand, network-based IDSs collect traffic over the network monitored. Generally they inspect packets promiscuously captured through the network interfaces for analysis. In practice network-based IDSs are widely deployed to monitor large networks as it is often impractical to maintain host-based monitoring on each host in the network. In this work we use the open-source network IDS Snort as a major source of explicit intrusion evidence.

Based on the detection technique, IDSs can be classified into two major categories: misuse

detection and anomaly detection [15]. Misuse detection encodes known attacks as a set of patterns and detects attacks by matching the audit data against these predefined patterns. Misuse detection systems can be further divided into stateless and state-based. Stateless systems such as Bro [3] only match the current event against rules to determine if it is malicious or not. In comparison, state-based systems such as USTAT [24] define more complex attack patterns with state transition diagrams, which includes system states and the actions causing state transitions. Not only the current event but also the system state are taken into consideration to see if the event would lead the system into an alarm-triggering state. Misuse detection systems are usually faster and has fewer false positives when compared with anomaly detection, however they are not able to catch novel attacks which do not match know signatures.

Anomaly detection systems generally operates in two phases: training and detection. The training phase builds profiles to model the "normal" behavior of the system or network being monitored. In detection phase, alerts are triggered when deviation from the normal profile is above a certain threshold. Anomaly detection is able to detect novel attacks as they do not rely on predefined knowledge of specific attacks. However, it is difficult to build adequate "normal" profiles, as the pattern of non-malicious activities may have wide fluctuations from time to time and seemingly anomalous behavior is benign in many cases. Moreover, attackers can even attempt to "train" the anomaly detection system to accept intrusive activities as normal. As a matter of fact, anomaly detection systems are more difficult to tune and prone to false negatives as well as false positives.

Although intrusion detection systems(IDS) are not developed for forensic analysis, investigators have recognized that IDS's are an important evidence source for network forensics because it is often the most applicable source to provide first hand information about security violations in the network. However, the overwhelming volume of low-level alerts and large quantity of false positives make it difficult for forensics investigators to understand the overall attack scenario.

2.3 Intrusion Alert Correlation

From a forensics investigator's point of view, it is important to reduce the redundancy of alerts, filter out false positives and intelligently combine individual alerts to identify a clear picture of what happened in a complex multi-stage attack. The answer to this need is intrusion alert correlation, the process of identifying alerts that belong to the same attack scenario. By linking alerts that satisfy certain relationships together, alert correlation aims to reduce the volume of alerts, suppress the effect of false positives and present the attack scenario on a higher abstraction level. Past work in alert correlation can be classified into the following categories.

Correlation methods based on similarity of alert attributes [9, 12, 27, 28, 61] are based on the notion that alerts belong to the same attack often have similar attributes. Valdes and Skinner [61] defined attributes to be used for comparison, how to evaluate the extent of similarity and the approach to assign weights for different attributes. We note that correlation of this category can be better depicted as alert clustering and aggregation. Correlation algorithms of this type are not able to capture the causal relationships between events. Thus the system in itself is not sufficient to reconstruct an attack scenario.

The second category of correlation methods are based on attack scenarios predefined by experts [16, 35] or learned from training datasets [13]. The correlation process is essentially signature based: IDS alerts are matched with predefined patterns to formulate an attack scenario. High level languages like STATL [18] and the chronicles formalism [35] have been developed to describe scenarios. However, an inherent limitation of these techniques is that they are not able to identify unknown novel attack scenarios. In addition, the large number of scenario variations makes defining a comprehensive knowledge base of scenarios difficult.

In recent works [10, 39, 40], a new correlation approach of defining pre-conditions (prerequisites) and post-conditions(consequences) was introduced. Alerts are correlated together if an earlier alert's post-condition matches or contributes to the pre-condition of an later alert. This method has the potential to identify new variations of attack scenarios as we do not need to specify complete attack scenarios. However, this method requires that appropriate pre and

post conditions must be manually pre-defined for all individual attacks in the event stream. If a relevant precondition or postcondition is not modelled, some causal relationships between alerts could go undetected. Given the large number of attacks and the site-specific nature of pre and post conditions, effective alert reduction would require a substantial modeling effort, similar to the effort required to develop complete attack rule sets for misuse-based detection systems.

Correlation methods based on statistical analysis greatly differ from the previous three in that it does not require any prior knowledge of attacks. Qin and Lee use the Granger Causality Test time series algorithm to correlate alerts [44, 45]. The underlying notion is that in multi-step attack scenarios, alerts should have statistical similarities in their attributes. Theoretically this method has the ability to recognize novel attack scenarios, however itself is not sufficient for the complete correlation process. Further verification and analysis is needed to reduce the effect of false positives and reconstruct the attack scenario.

Alert correlation is a challenging task. We believe that current alert correlation methods are not satisfactory for network forensics analysis needs due to the following limitations:

- Most of the proposed approaches have limited capabilities in practice because the modeling requirement for domain knowledge is non-trivial and requires a great deal of interaction and maintenance on the part of security analysts. In many cases when the detailed causality information required for correlation modeling is not readily available, these systems become ineffective.

- Lack of situational awareness in the correlation process make these methods incapable of discovering non-explicit events that seem fairly innocent individually, but are malicious when examined in correlation with certain context.

- Performance of all alert correlation methods are strictly limited by the performance of IDS. The correlation methods are not robust to deal with fragmentary and incomplete evidence.

Our evidence analysis approach integrates alert correlation methods and aims to address

some of their limitations. In our evidence preprocessing module, we extend the attribute-based method [61] into our flexible evidence aggregation process. We present a simple and applicable correlation scheme in attack phase and relevant resource based graph aggregation. Finally in our cluster evaluation module, we develop the target oriented effective sequence(TOEES) that filters out false positives and reconstructs multi-stage attack scenarios.

2.4 Graph Methods in Security Analysis

Our work is also related to a variety of graph based methods in network security analysis. *Attack graphs* are widely used to analyze whether certain goals of the target network can be achieved by attackers [43,46,47,50,51]. The goals are usually to obtain unauthorized privileges on targeted network entities. In an attack graph, a starting node represents an attacker at a specified network location. Nodes represents actions the attacker takes and edges denotes the changes in network state caused by the actions. Some researchers also use a different notation that each node represents a state of the network under attack and each edge represents a malicious activity that leads to network state transition. A full attack graph enumerates all possible sequences of exploits that could be used to achieve the intrusion goal.

Attack graphs have been mainly constructed to model network vulnerabilities. Researchers have proposed various methods to generate and analyze attack graphs. Phillips and Swiler [43] proposed to construct attack graphs based on predefined attack scenarios. Formal languages such as CAML [57] and LAMBDA [11] are developed to describe describe actions and states in attack graphs. These languages generally define the preconditions necessary for an attacker action to succeed and the post-conditions as a result of the attacker step. In recent works, model-checking methods have been used to automatically generate attack graphs [50, 51].

Attack graphs are also used in alert correlation context [58]. For this purpose attack graphs are first built for the network protected by IDS. Then the incoming IDS alert sequences are matched against paths in the pre-built graph in order to identify high-level multi-stage attacks.

The application of attack graphs in practice has several major limitations.

- The first problem of most past methods is scalability. Due to the huge volume of low

level system states and exploits involved, analytical complexity of the attack graph grows exponentially and become computationally infeasible for realistic large networks. Most past algorithms have only been able to generate attack graphs on small networks with fewer than 20 hosts. One of the most scalable research algorithm to date is developed at George Mason University [1], which requires computation grows at N^6(N is the number of hosts in the network). It is practically infeasible to compute attack graphs for large enterprise networks with thousands of hosts.

- The second major problem of many past work in attack graphs is that the attack modeling process often requires prohibitive low-level details of host, network, and exploits. For example, information used to describe pre and post conditions for exploits are not readily available from current vulnerability databases. Therefore extensive human analysis of vulnerabilities is needed to model the attacks, which is labor intensive and error prone. Moreover, methods that integrate alerts with attack graphs [58] require accurate up-to-date host based vulnerability information of the network, which is not realistic at the time of post-incident investigation.

Graph related methods have also been proposed in intrusion detection and correlation systems. GrIDS [56] aggregate network data into *activity graphs* to approximately represent the causal structure of network activity. *Alert correlation graphs* are defined in pre/post alert correlation works [39,40] to represent the causal relationship between intrusion detection alerts, where nodes represent hyper alerts and edges represent prepare-for relationships between alerts.

Our evidence graph model has a major conceptual difference with graphs in previous work that it is host-centric. With nodes represent host-level network identities and edges represent forensic evidence from heterogeneous network sources, the evidence graph model provides higher level of abstraction and better scalability. Efficient algorithms for the spectral clustering and Pagerank methods used in our graph analysis approach are readily to support analysis on large networks. Though the model is less fine-grained, it provides a practical view of the attack that is commonly applicable in post-incident intrusion analysis.

Graph clustering methods and the Pagerank algorithm have been extensively studied out-

side of security context([7,30]). To the best of our knowledge, our approach is the first approach that applies graph spectral clustering techniques to multi-stage attack analysis. We adapted the recursive clustering approach in [17] to minimize the need for parameter tuning and a priori knowledge of the clusters. The Pagerank algorithm [30] is initially developed for ranking web pages. Mehta et al. [32] proposed to apply the Pagerank algorithm for ranking security states in attack graphs. In comparison, we adapt the personalized Pagerank algorithm [67] on evidence graphs for targeted investigation and hidden-attacker-identification.

CHAPTER 3. GRAPH BASED HIERARCHICAL REASONING FRAMEWORK

3.1 Network Forensic Analysis System Overview

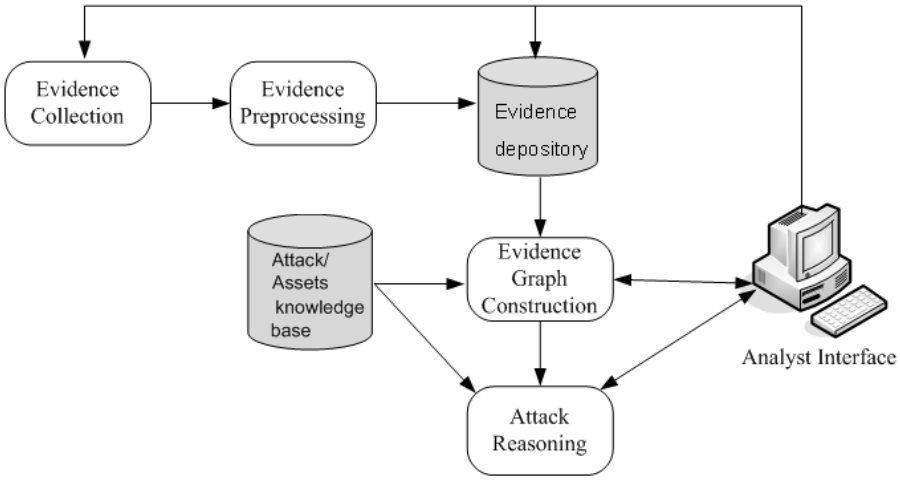


Figure 3.1 Network forensic analysis system architecture

Figure 3.1 shows the basic architecture of our proposed network forensic analysis system. Functionality of each component in the architecture is briefly described as follows:

- **Evidence collection module** collects digital evidence from heterogeneous sensors deployed on the networks and hosts under investigation.
- **Evidence preprocessing module** transforms collected evidence into standardized format and reduce the redundancy in raw evidence through aggregation.
- **Attack/Assets knowledge base** provides knowledge of known exploits as well as the

networks and hosts under investigation, which are used in graph construction and attack reasoning.

- **Evidence graph construction module** generates the evidence graph by retrieving preprocessed evidence from the depository. Hypotheses and out-of-band information are also instantiated into the evidence graph through graph edit operations.

- **Attack reasoning module** performs hierarchical reasoning based on the evidence graph to identify entities in multi-stage attacks and reconstruct the scenario. Techniques used in the reasoning process includes fuzzy inference, variations of clustering methods, Pagerank methods and target-oriented cluster evaluation for scenario reconstruction.

The core of our analysis system is the evidence graph construction module and attack reasoning module. The graph construction module constructs the evidence graph with preprocessed evidence and updates graph attributes with information retrieved from the knowledge bases. Following that, the attack reasoning module performs attack scenario inference based on the graph structure. Through the analyst interface, expert opinions and out-of-band information can be instantiated through graph edit operations. The analyst can also incorporate hypothesis into the evidence graph by formulating specific searches for extended secondary evidence. In return the reasoning process is performed on the updated graph to further evaluate the hypotheses.

Figure 3.2 illustrates our forensic analysis system from the data flow perspective. The graph construction module transforms the set of preprocessed primary evidence E_P into the initial raw evidence graph G_P, a directed weighted multigraph. Note that the attack reasoning module consist of six functional components: *Local Fuzzy Inference, Graph Aggregation, Primary Attack Clustering, Targeted Ranking, Hidden Member Identification* and *Cluster Evaluation.* Functionalities of the components are as follows:

Local Fuzzy Inference: We apply fuzzy reasoning to infer the functional states of a graph node from its local observations, which provide an initial view of the roles the host may play in an attack as well as the context for evidence evaluation.

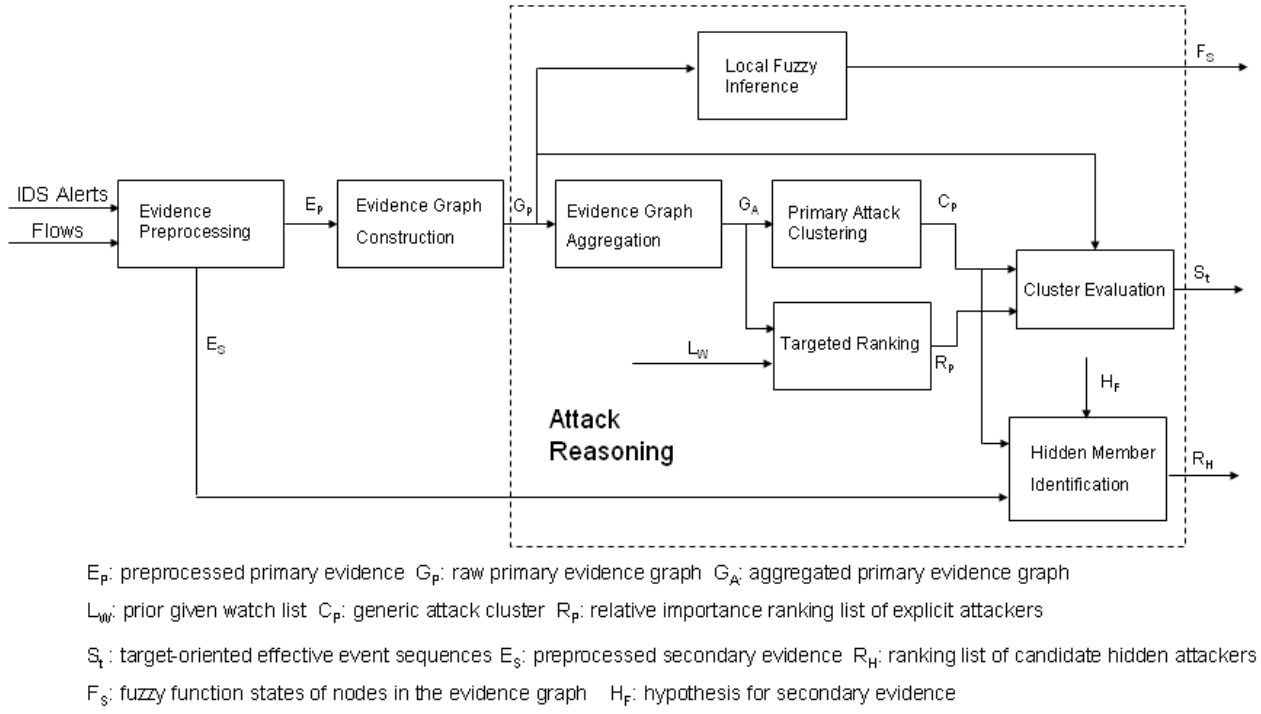


Figure 3.2 System data flow

Evidence Graph Aggregation: To facilitate interpretation and analysis of graph structure, we develop an aggregation scheme based on attack phase and relevant resources to convert the raw evidence graph G_P into the simple aggregated evidence graph G_A.

Primary Attack Clustering: In general investigation cases without additional upfront knowledge, we use clustering techniques to extract significant clusters from the aggregated evidence graph G_A. The set of clusters C_P indicate intensive attack activities and provide an approximation of potential multi-stage intrusion scenarios. Two types of clustering techniques are explored in our work: seed-based single-linkage clustering and spectral graph clustering.

Targeted Ranking: In cases when a watch list L_W is given for targeted investigation, we apply personalized Pagerank algorithm on the aggregated evidence graph G_A to provide rankings of potential attackers in terms of their relative importance to the specified watch list.

Hidden Member Identification: In this component we aim to identify hidden entities associated with malicious non-explicit-attack activities in a coordinated attack scenario. We build an enriched secondary evidence graph based on the primary attack clusters C_P and

hypothesis H_F. The list of potential hidden attackers are ranked with respect to their relative importance in the secondary evidence graph.

Cluster Evaluation: Inputs to the cluster evaluation module are entities identified from *Primary Attack Clustering* and *Targeted Ranking* as well as associated events retrieved from raw evidence graph G_P. We develop an automated approach based on the notion of *Target Oriented Effective Event Sequence* to further filter out false positives in the candidate cluster and reconstruct the attack scenario.

3.2 Evidence Category

In network forensic analysis, an event is an occurrence that changes the state of entities in the network. A malicious event is an event that serves certain malicious intentions of the attacker. Therefore we define evidence for network forensics investigation as digital data that provides information of malicious events that have occurred in a cyber attack scenario, which can be classified into two categories: primary (foreground) evidence and secondary (background) evidence.

- Primary evidence refers to information that directly indicates attacks or security policy violations. Primary evidence generally comes from sensors designed for specific security purposes such as IDSs.

- Secondary evidence refers to information that is not an explicit indication of exploits or security violations, but could represent malicious events in certain context. Secondary evidence may come from various general purpose sensors and in a much higher volume.

Intuitively primary evidence is the triggering point of forensic investigation. Searching for secondary evidence usually has two objectives: to discover hidden suspicious events that are not detected by specific security sensors and to evaluate the trustworthiness of primary evidence. However the huge volume and wide range of secondary evidence present a challenging problem for forensics analysis, as most event logs are not related to security events. We propose a flexible approach that uses preliminary analysis results from primary evidence as the guideline

to formulate hypotheses towards secondary evidence, which is presented in Section 3.6. In our current proof-of-concept prototype, we focus on two network based evidence sources in each category: network IDS alerts are used as primary evidence and network flow records are used as secondary evidence.

3.3 Evidence Preprocessing

Tasks in the preprocessing stage include normalization and aggregation. All types of evidence from sensors are normalized with predefined templates. Following that, aggregation is performed to reduce redundancy in applicable types of evidence. The pre-processed evidence are stored into the evidence depository for retrieval in graph construction and attack reasoning.

3.3.1 Evidence Normalization

Objective of the normalization process is to transform evidence from heterogeneous sensors into a uniform format for analysis. Although attributes of evidence vary with different sources, the following set of essential fields apply to all categories of evidence and have to be instantiated in the normalization process:

- *ID*: Unique index for the evidence record;
- *Subject*: The set of attributes that define the initiator of underlying event;
- *Object*: The set of attributes that define the target of underlying event;
- *Action*: Description of nature of the event such as its classification and possible consequence;
- *Time*: Time stamp(s) of the underlying event.

For intrusion alerts, we define a simplified template derived from IDMEF [23] to capture essential alert attributes. The result is denoted as raw alert. Format of the raw alert template is $\{AlertID, Classification, SrcIP, DesIP, DetectTime, HyperID\}$. The *HyperID* field is reserved for the aggregation procedure presented below.

Similarly, we define a simplified template based on Cisco NetFlow [38] format to uniquely define a flow record. Format of the network flow template is $\{FlowID$, $SrcIP$, $DesIP$, $SrcPort$, $Desport$, $StartTime$, $EndTime$, $Protocol$, $ByteCount$, $ServiceType$, $MonitorID\}$.

3.3.2 Evidence Aggregation

The large amount of redundancy in current IDS alerts makes it difficult to analyze the underlying attacks in an efficient manner. For example, a single event often generates many duplicate alerts in a short period and only one instance need to be kept for analysis purpose. In addition to redundancy reduction, aggregation also provides a more conceivable view of the attack. For example, an aggressive scanner would produce thousands of probing related alerts with the same source address and distinctive target addresses. By using an abstracted classification for the probe and aggregating all target addresses for the same source address during certain time frames, we can group these raw alerts into a single hyper alert that clearly demonstrate the scanner's activity and intention. In summary the evidence aggregation process aims to remove the duplicates and generate hyper alerts that are easy to analyze without losing relevant information.

We use alert aggregation based on similarity of attributes and context requirements to merge raw alerts into hyper alerts. Format of hyper alerts is $\{HyperID$, $Classification$, $SrcIP$, $DesIP$, $StartTime$, $EndTime$, $Count\}$. Each hyper alert has a one-to-many relationship with raw alerts. In the hyper alert template, the $Count$ field records the number of raw alerts that are merged into the hyper alert for statistical evaluation. The $HyperID$ field in the raw alert template records the unique identification number of the hyper alert it merged into. The index mapping between raw alerts and hyper alerts enable analysts to backtrack and examine alerts in finer scale.

We apply a flexible alert aggregation algorithm based on the $Leader - Follower$ model. In essence it adapts the similarity-based alert correlation method proposed by Valdes and Skinner [61]. The aggregation criteria is to combine alerts that have the same source-destination pair, belong to the same attack class and whose time stamp falls in a self-extending time window.

The time window of hyper alerts is self-extending in that if time stamp of the raw alert falls outside the [start time, end time] window of hyper alert but the difference is within a predefined limit T, the corresponding bound of the hyper alert time window is updated with time stamp of the raw alert. This implies that we are able to merge continuous duplicate raw alerts that span over a long period into a single hyper-alert with a proper T value. The $Leader-Follower$ alert aggregation procedure is shown in Algorithm 1.

input : A set of raw alerts $r_1 \ldots r_n$, time limit T
output: A set of hyper alerts $h_1 \ldots h_m$
begin
  $h_1 \leftarrow r_1$;
  $m \leftarrow 1$;
  for $i \leftarrow 2$ **to** n **do**
    $merged \leftarrow 0$;
    for $j \leftarrow 1$ **to** m **do**
      if $r_i.sourceaddr = h_j.sourceaddr$ && $h_j.destaddr = r_i.destaddr$ && $h_j.class = r_i.class$ && $h_j.starttime - T \leqslant r_i.detecttime \leqslant h_j.endtime + T$ **then**
        $h_j.starttime \leftarrow min(h_j.starttime, r_i.detecttime)$;
        $h_j.endtime \leftarrow max(h_j.endtime, r_i.detecttime)$;
        $r_i.hyperid \leftarrow h_j.id$;
        $h_j.count \leftarrow h_j.count + 1$;
        $merged \leftarrow 1$;
        $break$;
      end
    end
    if $merged = 0$ **then**
      $m \leftarrow m + 1$;
      $h_m \leftarrow r_i$;
      $h_m.count \leftarrow 1, h_m.HyperID \leftarrow m$;
    end
  end
end

Algorithm 1: Leader-Follower alert aggregation

Effectiveness of the $Leader-Follower$ aggregation process depends on the specific scenario being analyzed. Therefore when the initial outcome with Algorithm 1 is not satisfactory, we also consider the following flexibilities in aggregation criteria for improved results:

Level of abstraction on classification: One important variant that affects the results of

aggregation is the evaluation of attack classification. A common practice is to take the classifications from well known exploit collections such as CVE (Common Vulnerabilities and Exposures). However, it is common to observe that multiple different attack classes are defined for attacks exploiting the same vulnerability or having similar result. It is often desirable to evaluate attack classifications on a higher abstraction level such that trivial differences are ignored. For example, a "SCAN Nmap TCP" alert and a "SCAN Nmap XMAS" alert generated by Snort can be merged into one hyper alert with the same abstracted class "SCAN Attack". Although higher level of abstraction reduces the hyper alert space, we are aware that it results in loss of granularity in classification information and may lead to inappropriate aggregation.

Constraint on source/target: The general aggregation criteria in Algorithm 1 needs to be adapted for specific contexts. For example, by evaluating traces of attacks we discover that a large portion of raw alerts is triggered by port scan activity, which is usually of little significance. Therefore we can improve the efficiency of aggregation by using a strategy that only cares for hosts that are either source or target of large number of scan-related alerts. The former *one to many* characteristic often represents a scanner while the latter *many to one* characteristic often indicates a potential victim of attack. Consequently the flexible aggregation criteria uses an abstract type that represents all scan-related alerts and only require match of either *sourceaddr* or *destaddr*. Similarly, alerts triggered by DDOS attacks often exhibit a *many to one* pattern and should use an abstract type of "DDOS" class abstraction and require match only on *destaddr*.

3.4 Evidence Graph Model

The evidence graph model is the foundation of our forensic analysis process. Functionalities of the evidence graph model include:

1. The evidence graph provides an intuitive representation of collected forensic evidence.
2. The evidence graph provides the basis for both local functional and global structural analysis in attack reasoning.

3. The evidence graph provides a friendly interface that allows the analyst to incorporate hypotheses and out-of-band information into the reasoning process.

3.4.1 Model Description

Definition 1. *An evidence graph is a quadruple $G = (N, E, L_N, L_E)$, where N is the set of nodes, E is the set of directed edges, L_N is the set of labels that indicate the attributes of nodes and L_E is the set of labels that indicate the attributes of edges. In the evidence graph, each node n_i represents a host-level entity and each edge e_i represents a piece of preprocessed forensic evidence.*

In forensics analysis, entities are subjects or objects in the attack scenario that can be modelled at different granularity. Here we choose to represent entities on the host level. Therefore, each node in the evidence graph is characterized by the following labels:

1. *ID*: Unique identification of the host-level identity.

2. *States*: Each node is characterized by a set of fuzzy functional states to provide context for evidence evaluation and attack scenario analysis. In our prototype, a simple set of fuzzy states are defined as S={*Attacker*, *Victim*, *Stepping Stone*, *Affiliated*}. The fuzzy variable *Attacker(AT)* indicates the belief that the current node is a source of attack. The fuzzy variable *Victim(VI)* indicates the belief that the current node is a target of attacks. The fuzzy variable *Stepping Stone(SS)* indicates the belief that the current node is controlled by an attacker and used as a stepping stone in the scenario. By notion of "suspicion by association", the fuzzy variable *Affiliated(AF)* indicates the belief that the current node is suspicious because it has certain types of interactions with an attacker, victim or stepping stone. The set of functional states could be refined, but it is kept simple in this paper for illustration purposes. Note that these states are not mutually exclusive. For example, in a data theft scenario, a victim host that was compromised in an attack could be used as a storage relay to transfer stolen files. Consequently states of the host will evolve from *Victim* to both *Victim* and *Affiliated*.

3. *Time stamps*: Each functional state is associated with two time stamps: $T_{activate}$ records the initial time the state is activated above a certain threshold and T_{latest} records the latest time of update to the state.

4. *Value*: Each node is associated with a value $V \in [0, 1]$, indicating asset value of the host. For example, an important file server in the protected private network has a higher importance than a public web server in the DMZ. We note that host value is a quite site specific metric and extensive knowledge of the network under investigation is needed for proper assignment.

In addition to general attributed fields assigned in the normalization and aggregation process, each edge in the evidence graph is characterized by the following set of attributes:

1. *Weight*: Weight is a fuzzy value $w \in [0, 1]$ that represents the seriousness of evidence. For example, a probe attempt is assigned lower weight than a buffer overflow exploit as the former has little impact on the target while the latter could result in unauthorized access privilege. As shown in Table 3.1, in our prototype weights of known attacks are coarsely defined w.r.t to its *Attack Phase*, which can be directly obtained through classifications in public exploit databases. Details of *Attack Phase* are described in section 3.5.2.

2. *Relevancy*: Relevancy value of an edge represents the belief that the underlying action indicated by the evidence would successfully achieve its expected impact. For intrusion alerts, there could be three specific cases: relevant true positive, false positive and non-relevant true positive [29]. Relevant true positive refers to alerts that truly represent an attack and the attack achieves its expected impact. False positive refers to alerts that identify a legitimate event as an alert by mistake. Non-relevant true positive refers to alerts that truly represent an attack but the attack does not achieve its expected impact.

We define relevancy value for these three cases as follows:

$$r = \begin{cases} 0, & \text{false positive/non-relevant true positive;} \\ 0.5, & \text{unable to verify;} \\ 1 & \text{relevant true positive.} \end{cases} \tag{3.1}$$

The process to check the relevancy of an alert is denoted as alert verification. A static evaluation approach is to compare the prerequisites of an attack with configuration of the target host. If all prerequisites are completely satisfied, the alert is labelled as relevant and its relevancy value is assigned as 1. If one or more contradicting configurations are found, the alert is labelled as non-relevant and its relevancy value is assigned as 0; otherwise we are unable to determine the relevancy value is assigned as 0.5. This static approach could rule out attacks that are bound to fail because the target host is not vulnerable. However it cannot guarantee that attacks tagged as relevant are truly successful, as various other factors could lead to failure of the attack. For example, because the attack could still fail because of an incorrect parameter. Recently Kruegel and Robertson [29] proposed an active verification mechanism that checks for traces that match the attack's expected outcome on the victim host, but effectiveness of the method needs further study.

3. *Host Importance*: Host importance $h \in [0, 1]$ is a fuzzy parameter to relate importance of evidence to value of associated hosts. Our rationale for defining host importance lies in the idea that events associated with a highly valued host should receive more attention in intrusion analysis. The host importance for an edge e can be determined as follows:

$$h(e) = max(V_{src}, V_{dst}) \tag{3.2}$$

where V_{src} is value of the source node and V_{dst} is value of the destination node.

3.4.2 Discussions

In essence our evidence graph $G = (V, E)$ is a directed multigraph. Given an alarm with source ip address s and destination ip address d, two vertexes $s \in V$ and $d \in V$ are added to the evidence graph. A directed edge $e \in E = (s, d)$ is then drawn between s and d corresponding to the direction of the detected alarm.

The evidence aggregation process aggregates raw alerts into a reduced number of hyper alerts. Then we draw an edge corresponding to each distinctive hyper alert to create a multigraph. As shown in figure 3.3, the number of parallel edges between a pair of hosts B and C is decided by the set of unique hyper alerts between them. The rationale behind this approach is that our evidence graph not only maintains the adjacency link structure, but also preserves detailed information of specific attacks detected. More specifically, our objective is to address both the needs for structure based and knowledge based analysis. As we will show in chapter 4 and chapter 5, abstracted link structure of the evidence graph is sufficient for structure based analysis methods such as spectral clustering and personalized Pagerank methods. Through the link structure we are able to identify candidate identities involved in the attack scenario, i.e. *Attack group Identification*. However, knowledge based analysis is always required to evaluate the candidate identities and filter out false positives, perform *Hypothesis testing* and eventually achieve *Attack Scenario Reconstruction*. For these tasks details of the specific attacks need to be conveniently embedded in the evidence graph in addition to link structure.

Theoretically, when the alarm is generated by a host-based sensor that resides on the attacked host, s and d are equivalent and the alarm would be represented by a self-loop in the evidence graph as shown on host D in figure 3.3. As in this work we only consider network based sensors, self-loops of this kind will not appear in the evidence graph constructed.

Finally, we calculate priority score for an edge to indicate the overall importance of the intrusion evidence. The priority score $p(e)$ of an edge e is calculated as the product of its weight, relevancy and host importance:

$$p(e) = w(e) \times r(e) \times h(e) \tag{3.3}$$

As an example, a Windows DCOM buffer overflow attack is observed to initiate from at-

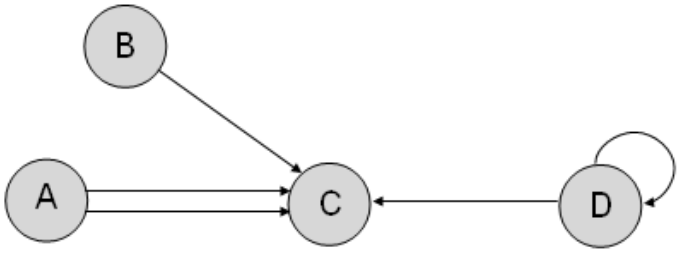


Figure 3.3 Evidence graph as a directed multigraph

tacker s against target t. By prior knowledge we know that t is a Linux server and consequently the Windows DCOM buffer overflow will have no impact on it. Therefore though weight of the edge $w(e)$ is high, priority score of the edge is zero as relevancy of the edge is $r(e) = 0$.

Weight and priority score are defined to provide different views in our reasoning process. The priority score $p(e)$ is used for attacker and scenario identification. In the above example, the failed attack has $p(e) = 0$, which indicates that it has little significance in the attack scenario because the attacker did not achieve his goal. However, edges with a zero priority score are not removed from the evidence graph because weight of the edge $w(e)$ is used in local reasoning for functional state of associated hosts. Though the attack is irrelevant, the unsuccessful attack attempt still indicates malicious intent of the source host s and consequently the "*Attacker*" state of s should be activated.

Note that in post-attack investigation, it is difficult to obtain accurate vulnerability and network state information for attack verification. Therefore to be conservative we regard all alerts as relevant for evidence graph construction. On the other hand, if vulnerability information is available, it could certainly be used to prune away irrelevant edges in the evidence graph structure and only improve the accuracy of analysis results.

3.4.3 Building Evidence Graph

To construct the evidence graph, the sequence of intrusion evidence is processed in time order, starting from the first evidence in the record and moving towards the latest evidence. Evidence with time intervals is added to the graph in order of the start time in their interval.

```
input  : Stream of evidence in time order
output: Evidence graph G
begin
    foreach evidence E in stream do
        foreach host V affected by E do
            if V does not exist in G then
                CreateNode (G, V);
            end
        end
        CreateEdge (G, E);
        foreach host V as subject or object in E do
            UpdateNode (E, V);
        end
    end
end
```

Algorithm 2: Constructing an evidence graph

For each evidence, we evaluate which nodes in the current evidence graph it will affect and create nodes that do not exist, then create the edge accordingly. The algorithm for building the evidence graph is listed in Algorithm 2. The *UpdateNode* function performs inference for the states of node by causal reasoning via Rule-Based Fuzzy Cognitive Maps(RBFCM), which we will describe in our hierarchical reasoning framework.

Finally, expert opinions and out-of-band information can also be directly incorporated into the evidence graph for automated reasoning through the following graph edit operations:

1. Insert a new node n: This represents adding a new suspicious host to the evidence graph.
2. Remove a node n: This represents removing an irrelevant host from the evidence graph. Note that removing a node implies removing edges associated with the node.
3. Update a node n: This represents changing one or more functional states of the node.
4. Insert a new edge e: This represents adding new intrusion evidence between existing nodes in the evidence graph.
5. Remove an edge e: This represents removing irrelevant evidence from the evidence graph.

6. Update an edge e: This represents adjusting the weight or relevancy value of the corresponding evidence.

3.5 Hierarchical Reasoning Framework

Based on the evidence graph, we develop a hierarchical reasoning framework for automated evidence analysis. In this section we describe two levels of the framework: local reasoning for functional analysis and global reasoning based on structure analysis.

3.5.1 Local Reasoning: Functional Analysis

The objective of local reasoning is to infer the functional states of an entity from its local observations. In the evidence graph space, "local" means that the inference is only based on information of the node itself and its direct neighbors. We argue that keeping track of host states has significant importance in forensics analysis.

1. Host states provide context for evaluating evidence. There is no absolute "suspicious value" of events. Actions of attackers are often represented by events that do not seem suspicious when examined individually without context. For example, legitimate file transfer connections associated with a healthy host generally do not indicate suspicious activity. However, an outbound ftp connection initiated from a victim host is likely to be a data exfiltration attempt that leads back to a host controlled by the attacker. Therefore, host states provide the context to discover hidden events for further investigation.

2. Host states derived with local observations provide an initial view of the roles the host may play in an attack. The evolution of host states helps to display the advancing stages of an attack to the forensics analyst.

The complexity of host systems and cyber attacks makes it difficult to reach a precise statement about host states. Therefore we use a fuzzy approach towards the problem because it is powerful in approximating human decision making process that involves qualitativeness

and inexactness. In the current prototype, we develop causal inference via Rule-Based Fuzzy Cognitive Maps(RBFCM) to model the states of nodes.

Fuzzy Cognitive Maps(FCM) are actually fuzzy directed graphs that combine neural networks and fuzzy logic to predict changes of the system. [5]. Nodes in a FCM are concepts that change over time and edges represent the causality link between nodes. The weight of an edge measures how much one concept influences the other. FCM has been used for decision support in many different domains, including network security and intrusion detection systems [53]. However, FCMs are limited in their capacity to model real-world scenarios for two reasons. First, generic FCMs can only represent simple monotonic cause-effect relations. Secondly, FCMs cannot cooperate with traditional fuzzy rules [5]. To express the strength of causality, domain expert knowledge is required to assign numerical edge weights. However due to the highly ad-hoc nature of attack traces and lack of training data, it is impractical to obtain appropriate weights in the network forensics domain.

Rule Based Fuzzy Cognitive Maps(RBFCM) are an evolution of FCM. A RBFCM is essentially a standard rule based fuzzy system plus feedback and mechanisms to deal with causal relations [6]. Compared with the generic FCM, RBFCM is better adapted for modelling complex dynamic systems because changes to the concept are not simply determined by the weight of the edges, but are defined by the fuzzy rules relating the concepts and inputs. This is an important advantage to incorporate non-symmetric and non-monotonic causal relationships. Fuzzy rules provide the analyst an effective approach to present domain knowledge. As shown in Figure 3.4, a RBFCM consists of fuzzy concepts and fuzzy rule bases. In our context, concepts are the defined set of functional states $\{Attacker, Victim, Stepping\ Stone, Affiliated\}$. For each concept, the respective fuzzy rule bases $\{RBA, RBV, RBS, RBA\}$ consist of "$IF...Then...$" fuzzy rules that define how each concept is affected by values of other concepts and incoming new evidence.

In the RBFCM shown in Figure 3.4, fuzzy rules are used to map multiple inputs (current value of states and new evidence) to the output (updated value of states). Note that the edges among concepts indicate that it is possible to activate any functional state from any other

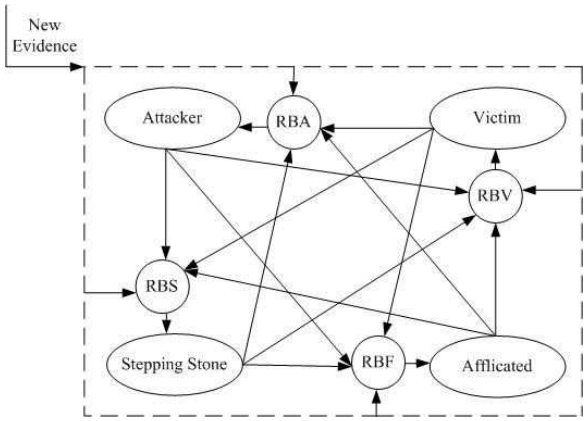


Figure 3.4 RBFCM model for local reasoning

functional state. The states are updated in an incremental manner. State values at time $t+1$ are determined by the states at time t and new evidences observed during the time interval $[t, t+1)$. The fuzzy rules are designed from expert knowledge. Below we present several examples of rules in our RBFCM model. The mapping of numerical ranges for states in the fuzzy rules is shown in Figure 3.5.

If *Back Orifice* is detected on host n
Then $Victim$ state of n is highly activated.

If attack of weight w initiates from host n
Then *Attacker* state value of n is increased by w.

If $Victim$ state is high and
Attacker state is high and
$T_{activate}(AT) > T_{activate}(VI)$
Then *Stepping Stone* state is highly activated.

If ftp connection to host n is detected and
host n's $Victim$ state is high and

$0 < T_{ftp} - T_{activate}(VI) < T_{limit}$

Then $Affiliated$ state is medium.

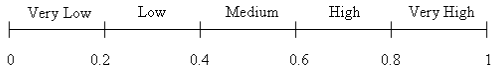


Figure 3.5 Mapping the fuzzy states

In local reasoning, we apply the assumption that states of hosts are monotonic. For example, once a host's *Victim* state is activated, it will never return to normal unless specifically instructed by out-of-band knowledge.

The total causal influence on each functional state is defined within the interval [0,1]. To monotonically map the concept value into the normalized range [0,1], we apply the sigmoid function in the following form where c is a positive constant.

$$f(x) = \frac{1}{1 + e^{-cx}} \tag{3.4}$$

3.5.2 Attack Phase and Relevant Resource Based Aggregation

We note that the raw evidence graph is a rudimentary presentation of intrusion evidence. However the multigraph structure make it difficult for efficient structure based analysis. Many graph structure analysis algorithms and tools do not provide full support for multigraphs [31]. Therefore an important prerequisite for our global reasoning based on graph structure is to aggregate the parallel edges between neighbor nodes into a weighted "super edge". In essence the edge aggregation process provides an abstraction of information encoded in the original set of edges. How to assign an appropriate weight to the "super edge" is critical, as the aggregated weight should be a good approximation of combined effects indicated by the original edge set.

For a pair of neighbor nodes (s, t) in the evidence graph G, let $E(s, t)$ represent the set of incident edges between s and t. Then we evaluate weight of the "super edge" $C(s, t)$ as a numeric abstraction of information encoded in the set $E(s, t)$ with higher weight represents

stronger correlation between s and t. As shown in equation 3.5, a straightforward aggregation approach is to calculate $C(s,t)$ as the sum of priority scores in the original set of parallel edges $E(s,t)$.

$$C(s,t)_{SUM} = C(t,s)_{SUM} = \sum_{e \in E\{s,t\}} p(e) \tag{3.5}$$

However, this simple aggregation often results in biased weights. Consider the scenario of large number of network sweeps vs. a single buffer overflow exploit. Intuitively the latter denotes greater potential for the attacker and more severe adverse effects. In contrast, in the simple summation approach the former would result in higher aggregated weight.

We develop an enhanced edge aggregation process with two considerations: (1) ease of modeling and (2) provide a reasonable abstraction of combined effects indicated by the original set of events.

Our graph aggregation scheme is based on augmenting the primary intrusion evidence with two attributes: attack phase and relevant resource. Intuitively, a multistage attack scenario is characterized by a sequence of intrusion events that belong to different attack phases such as *Reconnaissance, Intrusion, Privilege Escalation, Denial of Service, Post Compromise*. Here we define a simple set of attack phases with a name scheme such that the beginning part represents the general category and the following parts represent refined sub categories.

1. *probe_map(Reconnaissance)*: This is an initial reconnaissance step in which the attacker tries to map topology of the target network and find out live hosts.

2. *probe_vulnerability(Reconnaissance)*: This is a follow on reconnaissance step employed to discover exploitable vulnerabilities on the target systems.

3. *attack_privilege(Intrusion or Privilege Escalation)*: In this phase the attacker attempts to gain unauthorized access privilege to the target system by exploiting a known vulnerability. More detailed classification could be based on the level of access privilege such as *attack_privilege_user* and *attack_privilege_admin*.

4. *attack_dos(Denial of Service)*: In this phase the attacker attempts to prevent legitimate access to the target system by corrupting services or exhausting resources.

5. *post_misc(Post Compromise)*: In this phase the attacker performs goal-oriented activities after getting hold of the target system. There exist many subcategories in this category. For example, *post_consolidation* represents the attacker leverages his access to install tools such as an backdoor on the victim; *post_access* indicates communication through persistent control channel to the compromised system; *post_exfiltration* indicates data exfiltration attempts from compromised host to some external site; *post_integrity* represents modification or deletion attempts that tamper integrity of the compromised host.

The second attribute utilized in our augmented evidence model is relevant resource of attacks. Relevant resources encode information about the system resources utilized by the attack, i.e. the set of network and host properties the attack attempts to exploit. Currently we adopt a simple model to succinctly represent relevant resource R as (*OS*, *Service*, *Application*) which denotes the operating system, type of service and specific application.

Information modeling in our aggregation scheme is straightforward. Both attack phase classifications and relevant resources are defined in the Intrusion Detection Message Exchange Format(IDMEF) [23] data model. Practically they can be retrieved from attributes of current IDS alerts such as Snort [54]. Resource information can also be obtained from vulnerability databases like OSVDB [42]. Examples of extending primary intrusion evidence(Snort alerts) with attack phase and relevant resource information are shown in Table 3.1.

Alert	Attack Phase	Relevant Resource	Weight
ICMP PING NMAP	probe_map	any	0.5
RPC sadmind UDP PING	probe_vul	unix_sadmind	0.6
WEB-MISC apache DOS attempt	attack_dos	unix_apache	0.7
WEB-IIS IDA-IDQ exploit	attack_privilege_admin	windows_iis	0.8
WEB-FRONTPAGE rad fp30reg.dll access	attack_privilege_admin	windows_frontpage	0.8
RPC sadmind query with root credential	attack_privilege_admin	unix_sadmind	0.8
RSERVICES rsh root	post_access	unix_rsh	1.0

Table 3.1 Examples of extended snort alerts

input : A pair of adjacent nodes s and t in primary evidence graph G_P
output: Weight $C(s,t)_{PR}$ for edge (s,t) in aggregated evidence graph G_A

begin
  $E_{s,t} \leftarrow$ set of edges between s and t in G_P;
  Initialize $C(s,t)_{PR} \leftarrow 0$, attack phase and resource set $PR \leftarrow \emptyset$;
  while $E_{s,t}$ *is not empty* **do**
    Get $e \in E_{s,t}$ with earliest StartTime ;
    Look up attack phase P_e, relevant resource R_e and weight w_e ;
    if $(P_e, R_e) \subseteq PR$ **then**
      continue;
    end
    else
      $PR \leftarrow PR \cup (P_e, R_e)$;
      $C(s,t)_{PR} \leftarrow C(s,t)_{PR} + w_e$;
    end
    $E_{s,t} \leftarrow E_{s,t} \setminus e$;
  end
  return $C(s,t)_{PR}$;
end

Algorithm 3: Graph aggregation based on attack phase and relevant resource

We define the attack phase and relevant resource model for graph aggregation such that the aggregated weight would provide a reasonable representation of effects indicated by the original set of events. The rationale behind our aggregation scheme is that from the target's view, only events with the potential to increase the attacker's knowledge or capability should be counted towards the aggregated weight. We note that in multi-stage attacks, each effective step can be intuitively characterized by the advancement of attack phase or expansion of relevant resources, i.e. the attacker either proceeds to the next attack phase or attempts to achieve objective of the current attack phase by expanding the set of resources exploited. Therefore the integrated view of attack phase and relevant resources provides a unique measure of the attacker's increasing knowledge and capability with regards to the specific target. We illustrated the notion through the following scenarios:

1. The attacker advances in attack phase and continues with the same set of relevant resources. For example, a "RPC Sadmind Ping" alert is followed by a "RPC Sadmind Overflow" alert. The former alert belongs to phase *probe_vulnerability* while the latter

advances to *attack_privilege_admin*. However, both alerts have the same relevant resource {*Unix_rpc_sadmind*}.

2. The attacker advances in attack phase with different relevant resources. For example, the "RPC Sadmind Overflow" alert is followed by a "Telnet Access" alert. The former alert belongs to phase *attack_privilege_admin* while the latter advances to *post_access*. Also the two alerts have different relevant resources where R1 ={*Unix_rpc_sadmind*} and R2={*All_command_telnet*}.

3. The attacker remains in the same attack phase and continues with the same relevant resources. For example, Snort alerts "SNMP request tcp" and "SNMP trap tcp" both belong to attack phase *probe_vulnerability* and have the same resource R = {*all_snmp*}.

4. The attacker remains in the same attack phase with different relevant resources. For example, a "MS-SQL version overflow attempt" alert is followed by a "WEB-IIS fpcount attempt" alert. The two events both belong to the same phase *attack_admin*, while R1={*windows_database_sql*} and R2={*windows_web_iis*}.

What scenarios 1, 2 and 4 have in common is that advancement in attack phases and/or expansion of relevant resources lead to potential increase of the attacker's knowledge and capabilities. In both scenario 1 and 2, advanced attack phases show the attacker's continued progress in obtaining higher level of capabilities. Scenario 4 indicates an ongoing process of trying different means to achieve a certain capability, in which more knowledge of the target is revealed to the attacker. In contrast, for scenario 3 the attacker's knowledge or capability are not likely to increase because phase and relevant resource of the attacks overlap each other. Based on these analysis, detailed procedure for edge aggregation is presented in Algorithm 3. The stream of primary evidence is evaluated in time order with respect to their phase and relevant resources. Assuming hash-based lookups take constant time in the inside loop, the overall computational cost is $O(n)$ where n is the number of edges to aggregate.

With the attack phase and relevant resource based aggregation procedure we transform the raw evidence graph G_P into the aggregated primary evidence graph G_A. The aggregated

primary evidence graph $G_A = (V, E)$ is an undirected graph where V is the identical set of nodes from corresponding evidence graph G_P. For each pair $s, t \in V$, there is an undirected edge $e \in E$ if and only if s, t are neighbors in the corresponding raw evidence graph G_P. Weight of an edge e in G_A is denoted by $w(e) = C(s, t)_{PR}$.

3.5.3 Global Reasoning: Graph Structure Analysis

The global reasoning process aims to identify the set of highly correlated hosts that belong to the coordinated attack scene from structure of the evidence graph. Global reasoning is based on the assumption that during the procedure of attack, there must be a strong correlation between members of the attack group, and this correlation is exhibited through certain structural characteristics in the evidence graph. For example, a DDOS attack or an active scanner would be presented by a star-like topology in the evidence graph, while an unusual long path could be indication of a stepping stone chain. These distinctive graph components provide a first approximation of the coordinated attack scenario together with the functional state estimates from local reasoning.

We approach the global reasoning task as a group detection problem, which is to discover potential members of an attack group given the intrusion evidence observed. The attack group detection procedure works in two different phases: (1) create new attack groups by generating seed for the group and (2) expand existing groups by membership testing. Note that as shown in Fig 3.2, primary attack clustering analysis is based on the aggregated evidence graph G_A.

3.5.3.1 Seed Generation

In the seed generation phase, we aim to discover important nodes in the evidence graph as initial seeds of attack group. In essence, we would like to select entities that are both functionally significant and structurally important. From the functional perspective, the investigator generally has two options in seed selection. In a forward search manner, it is straightforward to select external hosts with *Attacker* state highly activated in the local reasoning process as initial seeds. In a backward search manner, hosts in the trusted domain with *Victim* or

Stepping Stone state highly activated are good candidates of initial seeds.

From the graph structure perspective, we use the eigenvector centrality metric to evaluate importance of nodes in the evidence graph. Eigenvector centrality is a refined version of the simple degree metric. Instead of just counting the number of edges incident to a node, eigenvector centrality score is based on both the number and the quality of edges. The intuition is that an edge to a node having a high eigenvector centrality score would contribute more than an edge to a node having a low score. Let centrality score of node i be denoted by x_i, which is proportional to the sum of the centrality score of n's neighbors:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{i,j} x_j \tag{3.6}$$

where A is the adjacency matrix of aggregated evidence graph G_A and λ is a constant. The equation can be rewritten in matrix form as $Ax = \lambda x$ where x becomes an eigenvector of the adjacency matrix with eigenvalue λ. In practice, we apply the power iteration method to obtain the dominant eigenvector of evidence graph's adjacency matrix, which represents the eigenvector centrality scores of corresponding graph nodes. By integrating the eigenvector centrality rank with functional state analysis, we can easily formulate queries for efficient seed selection. For example, "Among all nodes that have *Attacker* state activated higher than T, which one has the highest structural significance?"

3.5.3.2 Group Expansion

In group expansion process, we expect to discover nodes that have strong correlation with the initial seeds and add them to the attack group. Here we define the distance between two neighbor nodes s and t as the reciprocal of edge weight in aggregated evidence graph G_A, i.e. $d(s,t) = d(t,s) = \frac{1}{w(e)}$. Smaller distance values represent stronger correlation between two nodes. Group expansion is an iterative process which consists of three basic steps. First, we identify all external neighbors of current seed members as the set of candidate nodes. Second, a ranked list is formed based on the distance between each candidate node to current group members. Finally, the ranked list is cut at a predefined threshold and candidate nodes within

the distance threshold are added as new seed members of the group. If no candidate node is within the distance threshold, the group expansion procedure terminates. The procedure is listed in Algorithm 4.

```
input  : Evidence graph G, initial seed node v_s, distance threshold D, step size n
output: The derived attack group group
begin
    group ← v_s;
    neighbors ← ∅;
    candidates ← ∅;
    repeat
        foreach node v in the set group do
            neighbors ← FindNeighbour (G, v);
            candidates ← candidates ∪ neighbors;
        end
        foreach node v in the set candidates do
            v.distance ← GetDistance (v, group);
        end
        new ←RankCandidates (candidates, D, n);
        group ← group ∪ new;
    until no new member is found;
end
```

Algorithm 4: Basic attack group expansion process

The *FindNeighbors* function returns the set of external neighbor nodes of current seed members. In the *GetDistance* function, we evaluate the distance between the candidate node to its nearest seed member. In the *RankCandidates* function, candidate nodes whose distance exceeds the threshold are discarded. A ranked list is formed for the remaining candidate nodes in the order of increasing distance. Given the step size n, we take a greedy approach and the top n candidate nodes in the ranked list are added to the attack group as new seed members.

In essence the group expansion process can be regarded as a *seed based single link clustering* approach. It belongs to the class of hierarchical and agglomerative clustering algorithms, where each node start being its own cluster and clusters are merged iteratively. The conceptual difference is that we are only interested in clusters formed around "seed nodes" that possess both functional and structure significance. We apply the single linkage distance metric in group expansion, i.e. comparing minimum distance between candidate node and current seeds to the

distance threshold for membership testing. Other distance metric options such as complete and average linkage are less appropriate because in scenarios like a stepping stone chain, the suspicious entity may have strong correlation with single member of the current seed group.

Due to the vast difference in attack traces, selecting appropriate thresholds is not straightforward. Lowering the threshold generally leads to higher rate of false positives while raising the threshold could result in more false negatives. In practice the analyst usually need to compare the results of a set of incremental thresholds for evaluation. Moreover, the ranking list often explains more than trying to find the best cut-off threshold.

3.6 Interactive Hypothesis testing

Network forensic analysis is not complete without hypotheses testing. Intuitively, hypotheses are "what if" propositions made for possible explanations of inconsistencies in preliminary analysis results. For example, "What if the backdoor was uploaded to the victim by FTP?", "What if the buffer overflow alert between host s and t is an irrelevant background artifact?". Hypothesis testing is inherently an interactive process. As shown in Figure 3.6, the process of making and evaluating hypothesis based on the evidence graph model consists of the following steps:

1. Formulate hypotheses based on initial analysis results and out-of-band information.
2. Instantiate formulated hypotheses into the evidence graph with graph edit operations and extended queries on non-explicit secondary evidence.
3. Perform analysis on the updated evidence graph. Examine the updated results and reiterate from step 1 for new hypotheses.

Hypotheses can be classified into two major categories: removing irrelevant evidence and adding missed evidence. We note that the former essentially boils down to the relevancy evaluation problem discussed in evidence graph model construction. The latter case be further grouped into the following two types:

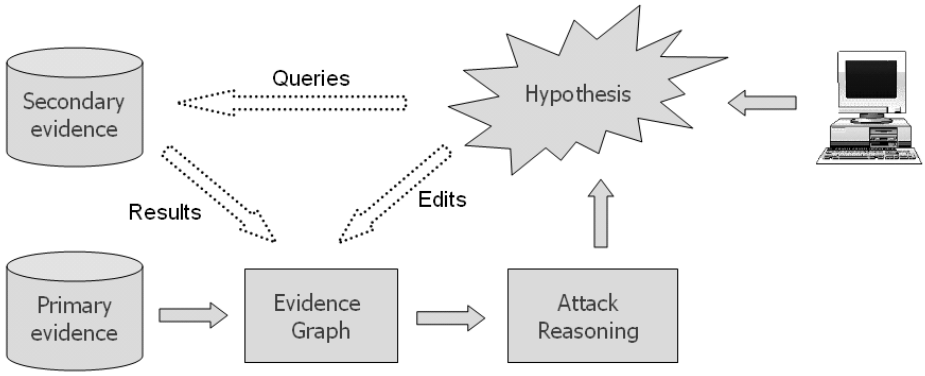


Figure 3.6 Hypothesis testing process overview

- *Missed Primary Evidence*: Serious attacks could be missed by security sensors due to their limited coverage and detection rate. In this case, entities in the attack scenario that should be in the same attack group are split into several isolated components in the aggregated evidence graph, each of which only represents a part of the complete attack scenario.

- *Missed Secondary Evidence*: Steps in the attack scenario could be legitimate operations that do not trigger security alerts on their own. For example, the attacker could use a separate host other than the ones employed in direct attacks as the warez server. After successfully compromising a target, the attacker uploads backdoor and exploit tools from the warez server to the victim by ftp. We are able to discover the group of attacker and victims as explicit attacks are captured in primary evidence. However the warez server will evade detection because the file transfer flows appear benign and are absent from IDS alerts.

Automated hypotheses generation for missed primary evidence is certainly a direction for future research, however is out of the scope of this work. Ning et al. [41] proposed a scheme to hypothesize for missed attacks by (1) similarity of alert attributes and (2) potential prepare-for relations encoded in equality constraints, which can be integrated with our approach. Through graph edit operations defined in section 3.4.3, hypotheses can be instantiated as new nodes

and edges into the evidence graph for local and global reasoning.

We note that it is impractical to apply all secondary evidence in building the initial evidence graph as most of them are irrelevant to the attack. The overwhelming amount of innocuous events will not only result in an explosion in graph size but also obscure the foreground attack scene. Therefore we propose a flexible approach that uses analysis results of primary evidence to formulate hypotheses for missed secondary evidence. Specifically, the following factors are considered:

- *Association:* Benign activities associated with a misfeasor could be suspicious. For example, a ftp attempt from the $Victim$ to an outside host could represent the exfiltration of sensitive data. Moreover, the belief of suspicion could be strengthened with *Non-Coincidence* in terms of association. For example, a host that has connections with several identified $SteppingStones$ in the attack group is more likely to be relevant to the attack scenario than incidental.

- *Classification* denotes the event type(s) of interest. It is more prudent to give greater importance to certain classes of events than others. The classifications are often derived from specific host based information and consequences of attack. For example, http connections associated with a web server are arguably not as suspicious as a ssh login. Also, backdoor tools discovered on a trusted internal host implicitly indicate that file transfer activities should be examined with more attention.

- *Time:* In addition to *Classification* and *Association*, the range of potentially suspicious secondary evidence is further narrowed down with temporal constraints, which could be inferred from activation and update time stamps of functional states in local reasoning results.

Therefore after the initial evidence graph is constructed from primary evidence, we can formulate hypotheses for missed secondary evidence based on the triple (*Association*, *Classification*, *Time*). By querying the secondary evidence repository, the hypotheses are then instantiated into the enriched evidence graph. Note that defining *Classification* is more hypothesize-

and-test than deterministic in nature. Our objective is to allow the investigator flexibly evaluate various possibilities of *Classification* in an interactive manner.

3.7 Experimental Results

In our prototype system for experiments, Snort is used as the network IDS sensor to generate intrusion alerts. We use softflowd [55] to process the TCPDUMP traces and generate datagrams in NetFlow format. The OSU flow-tools [21] is used as the collector to capture NetFlow datagrams. Both raw and preprocessed evidence are stored in a MySQL database. The local and global reasoning modules are developed with Perl. We also implement a GUI application based on LEDA [31] for visualization and manipulation of evidence graphs.

Traces of three multi-stage attack scenarios are used to validate our proposed hierarchical reasoning framework. The first dataset contains a small-scale attack implemented in our own testbed. To further evaluate the ability of our techniques, we perform experiments on two public intrusion detection datasets, the MIT Lincoln Lab DARPA dataset [14] and dataset provided by ARDA contract No. NBCHC030107.

In the preprocessing phase, we use reduction rate to evaluate the efficiency of reducing redundancy in raw evidence. For intrusion alerts, the reduction rate is defined as one minus the ratio between number of hyper alerts over number of raw alerts. Similarly for network flows, the reduction rate is one minus the ratio between size of NetFlow records over that of raw traffic trace.

To examine the output of our hierarchical reasoning process, we consider the following possibilities. If a host actually involved in the attack scenario is included in the attack group and its role is correctly recognized, we define it as a *true correlation*. A *false correlation* denotes that a host not related to the attack scenario is included in the attack group or its role is misclassified. We call it a *missing correlation* when a host actually plays a role in the attack scenario but is not correlated into our attack group. With sufficient ground truth for the attack scenario, we are able to identify true, false and missing correlations. Two quantitative metrics are used to evaluate the overall effectiveness of our attack reasoning methods. The reasoned

attack group represents the set of hosts obtained in group expansion result, while the actual attack group represents the set of hosts truly involved in the attack scenario.

$$Accuracy = 1 - \frac{\text{number of false correlations}}{\text{number of hosts in the reasoned attack Group}} \tag{3.7}$$

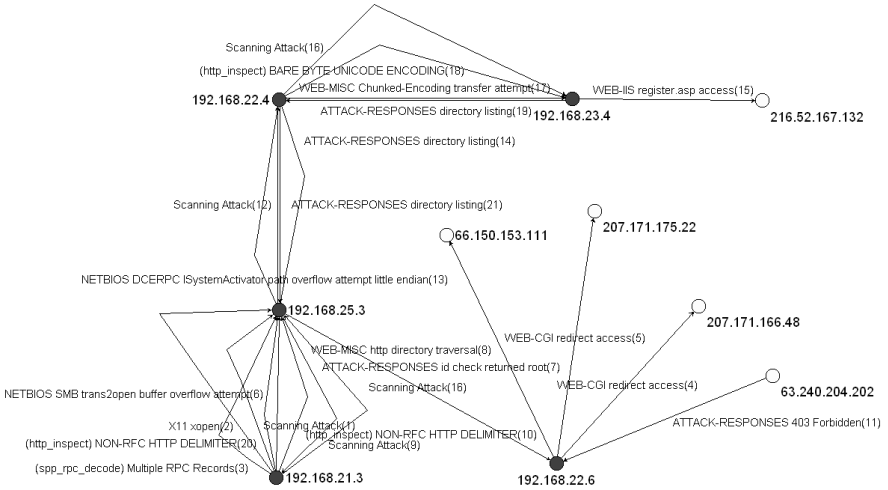
$$Coverage = 1 - \frac{\text{number of missing correlations}}{\text{number of hosts in the actual attack group}} \tag{3.8}$$

3.7.1 Scenario 1

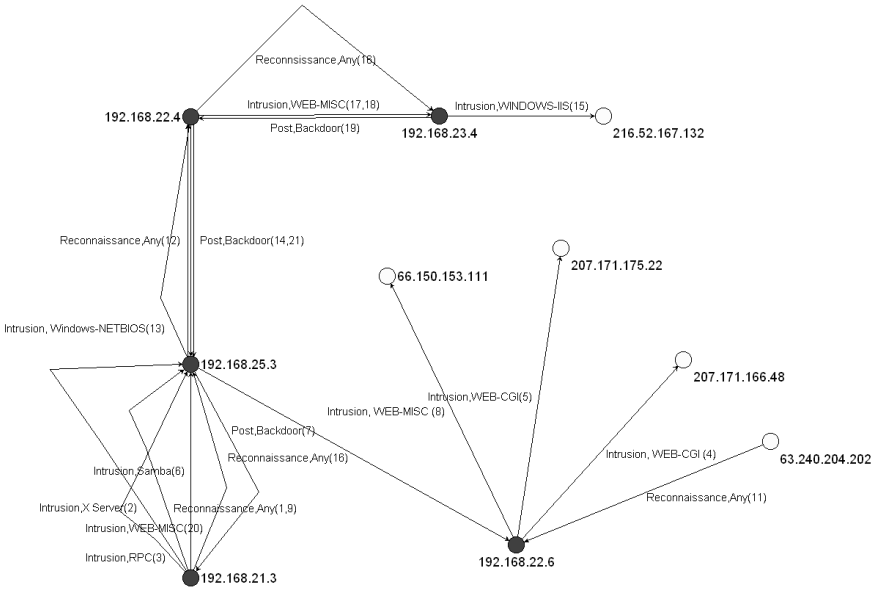
The first multi-stage attack scenario is implemented in our own testbed of around 30 physical hosts and 5 subnets. Multiple exploits, backdoor programs and scanning tools are used to bring more variety to the scenario. In addition, background traffic including http, ftp, sftp, ssh and telnet are generated throughout the whole attack process to simulate a more realistic network environment. Irrelevant random attacks are also included to obscure the coordinated attack scenario.

For notation convenience we adopt the following labels for hosts involved in the attack scenario *Attacker*:192.168.21.3, *Stepping Stone 1*: 192.168.25.3, *Stepping Stone 2*:192.168.22.4 , *Victim*: 192.168.23.4 and *Ftp Relay*: 192.168.24.4. The scenario includes the following steps:

1. *Attacker* initiates Samba buffer overflow attack against *Stepping Stone 1*.
2. Attack tools are uploaded from *Ftp Relay* to *Stepping Stone 1*; also an Netcat backdoor is started on *Stepping Stone 1*.
3. *Stepping Stone 1* initiates Windows DCOM buffer overflow attack against *Stepping Stone 2*. Attack tools are uploaded from *Ftp Relay* to *Stepping Stone 2*.
4. *Stepping Stone 2* initiates Frontpage Server 2000 buffer overflow attack against the *Victim*. After break in backdoor tool is uploaded from *Ftp Relay* to *Victim*.
5. Sensitive data is transferred from *Victim* to the *Ftp Relay* and backdoor connections are closed.



(a) Original raw primary evidence graph



(b) Attack phase and resource for correlation evaluation

Figure 3.7 Scenario 1 initial raw evidence graph from primary evidence

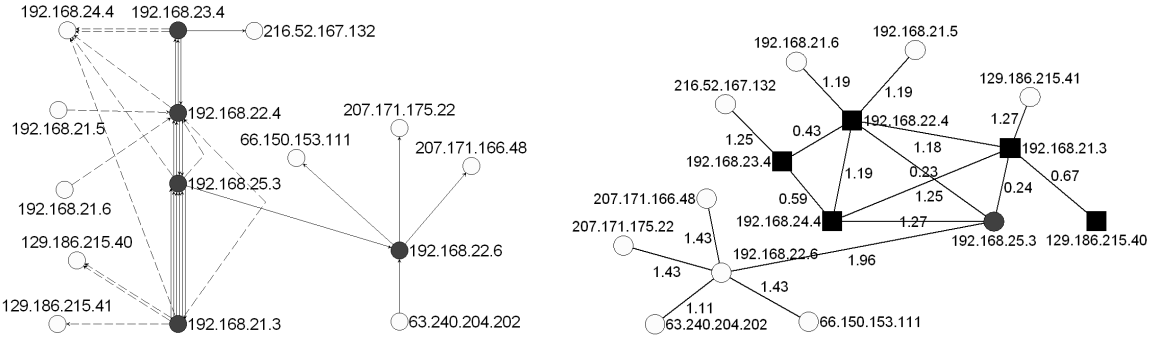
3.7.1.1 Evidence Graph Construction

In preprocessing phase, we use a self-extending time window T of 60 seconds for alert aggregation. The overall preprocessing results are shown in Table 3.8. The initial raw primary evidence graph shown in Figure 3.7(a) is then generated from the aggregated hyper alerts. The number attached to each edge denotes the sequence of corresponding events in time order. Figure 3.7(b) illustrates the corresponding attribute set *Attack Phase, Relevant Resource* for each alert, which are used for correlation evaluation in global reasoning process. Note that the *Ftp Relay* does not show up in the initial evidence graph as file transfer activities do not trigger Snort alerts.

In the next step, hypotheses can be formulated for potentially missed secondary evidence and instantiated to build the enriched evidence graph, which offers a more comprehensive view of what happened in the network. Hypotheses for missed secondary evidence are formed with the triple {*Association, Classification, Time*}. By observing the existence of backdoor tools on exploited hosts, it is reasonable to define *Classification* as all file transfer flows(ftp,tftp). *Association* is defined as any host with *Victim* state activated and *Time* is extracted from activation time of the *Victim* state. The enriched evidence graph is shown in Figure 3.8(a) where the solid edges represent primary evidence while the dotted edges represent events from secondary evidence. We notice that several potentially suspicious hosts include the *Ftp Relay* are brought up in the enriched evidence graph.

3.7.1.2 Local Reasoning

In the first step, the analyst would examine states of hosts from the local reasoning process. Based on the RBFCM local reasoning procedure, states of nodes in the evidence graph are inferred and shown in Table 3.2. The hosts that have "*Attacker*" state activated are highlighted in Figure 3.7(a), 3.7(b) and Figure 3.8(a).



(a) Scenario 1: enriched evidence graph with secondary evidence

(b) Scenario 1: aggregated evidence graph

Figure 3.8 Scenario 1: enriched evidence graph and aggregated evidence graph

3.7.1.3 Global Reasoning

First, we compute eigenvector centrality scores of nodes in the initial primary evidence graph for seed generation. From Table 3.3 we can see that 192.168.25.3 has the highest score. Also note that in local reasoning it has all states activated. Therefore host 192.168.25.3 is chosen as the initial seed based on both functional and structural importance.

In the next step, the attack group is constructed incrementally from the initial seed. Distances between neighbor nodes are shown in the aggregated evidence graph (figure 3.8(b)). The highlighted oval node is the initial seed and the highlighted square nodes are members added to the attack group during the group expansion process with distance threshold 1.

By observing the states of members in the attack group we can see that host 192.168.21.3 has *Attacker* state activated. Hosts 192.168.25.3 and 192.168.22.4 both have *Stepping Stone* state activated. Host 192.168.23.4 has both *Attacker* and $Victim$ state activated but not *Stepping Stone*. Further examination of the activation time of states clearly suggests that host 192.168.21.3 is the initial start point of attack and hosts 192.168.25.3,192.168.22.4 are used as stepping stones. As the result of hypothesizing for missed secondary evidence, the set of nodes 192.168.24.4, 192.168.21.5, 192.168.21.6, 129.186.215.40 and 129.186.215.41 all have $Affiliated$ state activated, which suggest that they are possibly involved in malicious non-explicit-attack activities. However, 192.168.24.4 is affiliated with all attackers and stepping stones in the attack

Host	AT	VI	SS	AF
192.168.22.4	0.85	0.85	0.87	0.84
192.168.25.3	0.85	0.80	0.94	0.84
192.168.21.3	0.80	0	0	0.84
192.168.23.4	0.69	0.85	0	0.82
192.168.24.4	0	0	0	0.84
192.168.22.6	0.85	0.50	0	0
129.186.215.40	0	0	0	0.81
129.186.215.41	0	0	0	0.69
192.168.21.5	0	0	0	0.70
192.168.21.6	0	0	0	0.70
207.171.166.48	0	0.67	0	0
207.171.175.22	0	0.67	0	0
66.150.153.111	0	0.67	0	0
216.52.167.132	0	0.69	0	0
63.240.204.202	0	0.71	0	0

Table 3.2 Scenario 1: local functional states

Host	Eigenvector centrality
192.168.25.3	0.68
192.168.21.3	0.53
192.168.22.4	0.46
192.168.23.4	0.21
192.168.22.6	0.07
216.52.167.132	0.03
207.171.166.48	0.01
207.171.175.22	0.01
66.150.153.111	0.008
63.240.204.202	0.008

Table 3.3 Scenario 1: eigenvector centrality scores

group while any other node is only affiliated with one member. Though more investigation is needed, it is reasonable to conclude that 192.168.24.4 is more suspicious than 129.186.215.40 due to its higher degree of non-coincidence. We also note that although 192.168.22.6 is labelled as an attacker in local reasoning, it is isolated from the major attack group and regarded as a irrelevant background attacker. The coverage and accuracy of reasoning is shown in Table 3.9.

3.7.2 Scenario 2

The second dataset we examine is the MIT Lincoln Lab 2000 DARPA intrusion detection scenario 2.0.2. The LLDOS 2.0.2 dataset contains a series of attack sessions that can be grouped into the following stages:

1. The attacker probes public DNS server of the target network via a HINFO query.

2. The attacker compromises the DNS server by exploiting the Solaris Sadmind vulnerability.

3. The attacker uploads attack scripts and mstream DDoS software to the compromised DNS server via FTP.

4. The attacker telnets to the compromised DNS server. Then the probing and Sadmind exploit process is repeated towards hosts in the network. After successfully breaking into one Solaris host, the attacker uploads mstream software via FTP.

5. The attacker access the compromised hosts via telnet and initiates DDoS attack towards an external web server.

3.7.2.1 Evidence Graph Construction

The attack scenario lasts about 1 hour and 45 minutes. In preprocessing, a self-extending time window T of 60 seconds is used in alert aggregation. We note that the general aggregation criteria does not perform well because most alerts focus on a limited number of targets. Thus we free up the constraint on *Desaddr* field to better model the many-to-one pattern in raw alerts. The overall preprocessing results are shown in Table 3.8.

Given the initial evidence graph, the analyst can filter out irrelevant events for a clearer view. In this scenario we arguably ignore all *Reconnaissance* alerts that are not followed by more advanced *Intrusion* or *Post Compromise* alerts. In formulating hypotheses for missed secondary evidence, we define *Classification* as all file transfer(ftp,tftp) and remote access (ssh,rlogin,telnet) flows. *Association* is defined as any host with *Attacker* or *Victim* state activated. and *Time* is extracted from activation time of the corresponding functional states. For clearer illustration, we only show part of the enriched evidence graph corresponding to the reasoned attack group in Figure 3.9.

3.7.2.2 Local and Global Reasoning Results

In local reasoning process, functional states of nodes in Figure 3.9 are shown in Table 3.4.

In global reasoning phase, we first compute eigenvector centrality to evaluate the importance of each node in the evidence graph. Top nodes in terms of eigenvector centrality score are shown in Table 3.5.

In the next step, we examine the effectiveness of seed based single link clustering. As shown in Table 3.5, node 131.84.1.31 has the highest eigenvector centrality score. Local rea-

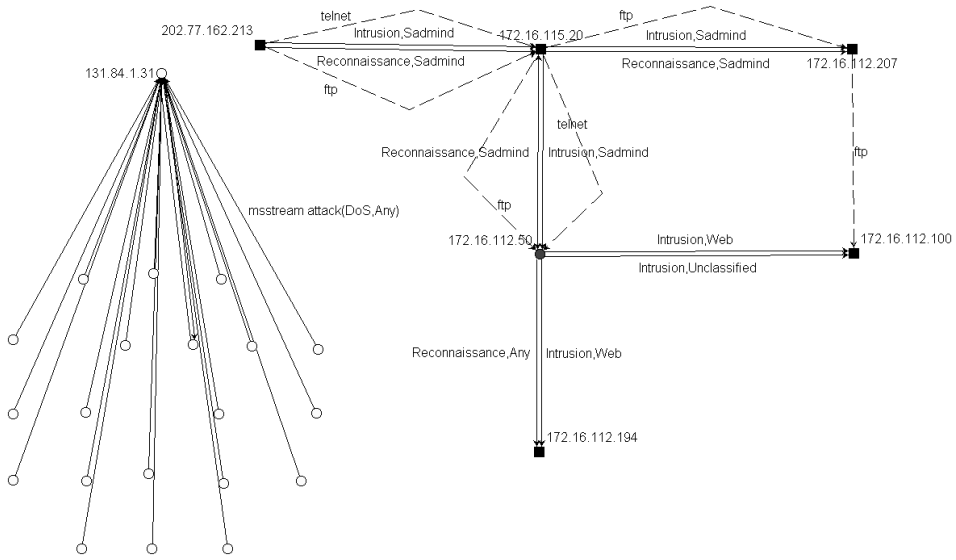


Figure 3.9 Scenario 2: enriched evidence graph with extracted attack group

Host	AT	VI	SS	AF
202.77.162.213	0.84	0	0	0.84
172.16.115.20	0.84	0.84	0.99	0.84
172.16.112.50	0.82	0.84	0.99	0.84
172.16.112.207	0	0.65	0	0
131.84.1.31	0	0.99	0	0
172.16.116.194	0.82	0	0	0
172.16.112.100	0.80	0.80	0	0.84

Table 3.4 Scenario 2: functional states from local reasoning

soning shows that 131.84.1.31 has a highly activated *Victim* state. Thus we start by choosing 131.84.1.31 as the initial seed. As shown in the left side of Figure 3.9, the result cluster has a star topology in which edges correspond to mstream DOS alerts, which intuitively indicates the many-to-one DDoS attack pattern. However it is an isolated component in the evidence graph due to spoofed IP addresses. To further explore the attack scenario, we choose 172.16.112.50 as the seed, which has the second highest eigen centrality score and an active *Stepping Stone* state. With threshold set to 80 percent of all correlation scores, the result attack group is shown in Figure 3.9 where the highlighted oval node represents the seed and

Host	Eigenvector centrality
131.84.1.31	0.78
172.16.112.50	0.25
172.16.112.100	0.18
172.16.114.50	0.17
172.16.113.148	0.14
172.16.115.20	0.14
172.16.112.207	0.13

Table 3.5 Scenario 2: eigenvector centrality scores in global reasoning

highlighted square nodes represent members in group expansion. From Table 3.4 we can see that host 202.77.162.213 has *Attacker* state activated. Hosts 172.16.115.20 and 172.16.112.50 both have *SteppingStone* state activated while host 172.16.112.207 have only $Victim$ state activated. By comparing the activation time of these states, it is straight forward to see that 202.77.162.213 started the attack and the attacker initially broke into the protected network through 172.16.115.20. Following that 172.16.115.20 fans out attacks against 172.16.112.50 and 172.16.112.207. Though host based information is needed for validation, 172.16.112.50 appears more likely to be compromised as its *SteppingStone* state has been activated.

We can see that the attack group and major attack steps in the ground truth are successfully extracted. The false correlations 172.16.116.194 and 172.16.112.100 are caused by irrelevant background attacks. Further host based investigation is needed to filter out these artifacts. Note that for target of the DDoS attack 131.84.1.31, its $Victim$ state is activated immediately after the activation time of *SteppingStone* state of 172.16.115.20 and 172.16.112.50, which suggests a strong logical relation between the two separated attack components. Overall coverage and accuracy of the scenario is shown in Table 3.9.

3.7.3 Scenario 3

Finally, we perform experiments on an attack dataset prepared under ARDA contract No.NBCHC030117 which initially funded this work. In the ARDA "Bankshot" dataset a multistage attack scenario is implemented in a network with multiple subnets and several hundred hosts. Background traffic and exploits are calibrated to the level of real-world traces

collected from a large corporate network. Compared with previous scenarios, this is a more realistic example of network forensic investigation in a large scale network where the foreground attacks are masqueraded by huge volume of background noise. **Specifically, over 99.9% of alerts generated from the dataset are false alarms or irrelevant background attacks.** We note that this multi-stage attack scenario is quite stealthy in that the attacker changes identity at different stages and uses compromised host inside the trusted domain as the "Stepping Stone". From the ground truth, the whole attack can be split into the following phases:

1. The attacker probes the protected network, then launches a failed exploit attempt against a Windows host from a different source.

2. The attacker launches attack against a Linux host from a different source and successfully compromises it through a web server exploit.

3. The attacker uses the compromised Linux host to probe other hosts inside the protected network, then launch attacks and breaks into one of them.

4. The attacker exfiltrates sensitive data from compromised hosts and exits.

3.7.3.1 Evidence Graph Construction

The attack scenario lasts about 60 minutes. In the preprocessing stage, we use the general aggregation criteria with a self-extending time window T of 60 seconds. The overall preprocessing results are shown in Table 3.8.

Similar to scenario 2 we also ignore *Reconnaissance* alerts without correlation to more advanced *Intrusion* or *Post Compromise* alerts. Only significant scanners that initiate intensive probing activities are kept in the evidence graph. For missed secondary evidence, we define *Classification* as all file transfer(ftp,tftp) and remote access (ssh,rlogin,telnet) flows. *Association* is defined as any host with *Attacker* or *Victim* state activated and *Time* is extracted from activation time of the functional states. The part of enriched evidence graph corresponding to reasoned attack group is shown in Figure 3.10.

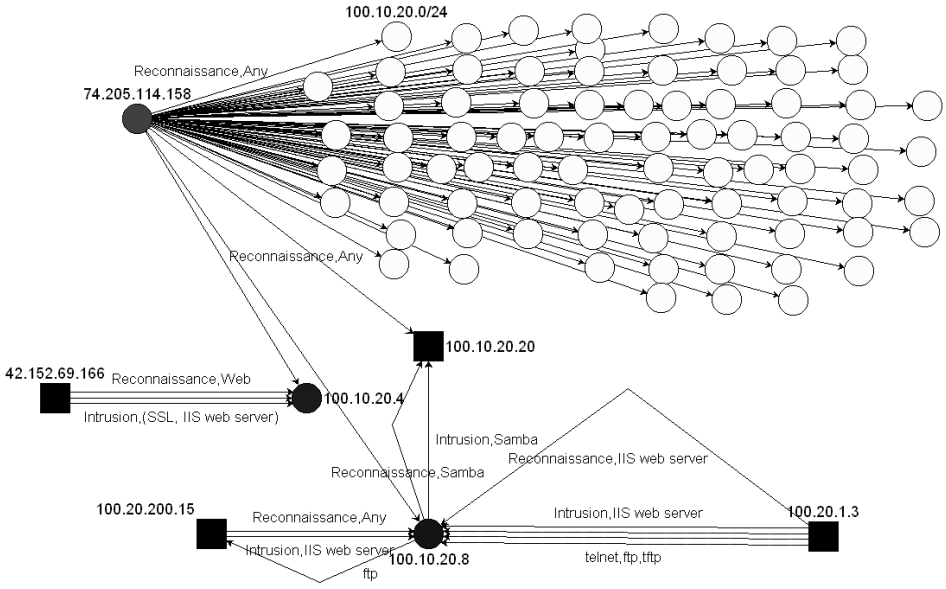


Figure 3.10 Scenario 3: enriched evidence graph with extracted attack groups

3.7.3.2 Local and Global Reasoning Results

States of nodes inferred in the local reasoning process are shown in Table 3.6. Top nodes in terms of eigenvector centrality score are shown in table 3.7. In global reasoning, 74.205.114.158 is selected as the initial seed as it has the highest centrality score and *Attacker* state activated. However, the group expansion process fails to find other members of the attack group because edges incident to 74.205.114.158 all represent scanning activities that have a low correlation score. To further explore the dataset, we select node 100.10.20.8 as the second seed, which has the next highest centrality score and both *Attacker* and $Victim$ state activated. By setting the threshold at 90 percent of all correlation scores, the result attack group is shown in Figure 3.10 where the highlighted square nodes represent members expanded from 100.10.20.8.

In the next step, local and global reasoning results are combined to extract the attack group and scenario. It is intuitive to observe that 74.205.114.158 initiates intensive probing activities and has *Attacker* state activated, which correctly identifies one external attacker in step 1. The second seed 100.10.20.8 is also correctly identified as the $Victim$ in step 3, but its *Attacker* state is incorrectly activated due to irrelevant background attacks against

Host	AT	VI	SS	AF
74.205.114.158	0.69	0	0	0
42.152.69.166	0.80	0	0	0
100.10.20.4	0	0.69	0	0
100.20.200.15	0.69	0	0	0
100.10.20.8	0.79	0.80	0	0
100.20.1.3	0.80	0	0	0
100.10.20.20	0	0.69	0	0.84

Table 3.6 Scenario 3: functional states from local reasoning

Host	Eigenvector centrality
74.205.114.158	0.93
100.10.20.8	0.21
100.10.20.4	0.08
100.20.200.15	0.07
100.10.20.20	0.06
42.152.69.166	0.02
100.20.1.3	0.01

Table 3.7 Scenario 3: eigenvector centrality scores in global reasoning

100.10.20.20. Host 100.20.1.3 is also a false correlation caused by background noise, the similar pattern of attack against the Windows IIS vulnerability makes it difficult to identify it without host based information. Two external attackers 168.225.9.78 and 91.13.103.83 in step 2 are absent from the evidence graph because of missed primary evidence. As no further connections were established between the two absent attackers to other members of the attack group, we are not able to identify them even with the enriched evidence graph. According to the ground truth, the attacker breaks into host 100.20.200.15 and used it as a stepping stone. In our analysis, though 100.20.200.15 is correctly included in the attack group, it has only *Attacker* state activated because the earlier exploits against it evaded detection of Snort. The analyst has to rely on out-of-band information that 100.20.200.15 is an internal trusted server to reason it was potentially compromised. Finally, we note that host 100.10.20.4 has the third highest eigenvector centrality score and $Victim$ state activated, however it is not related to the attack group expanded from 100.10.20.8. By selecting 100.10.20.4 as the initial seed, the external attacker 42.152.69.166 is extracted in group expansion, which reveals the failed exploit

		Scenario 1	Scenario 2	Scenario 3
Traffic	TCPDUMP	300 MB	118 MB	3.6 GB
	NetFlow	6.4 MB	10.6 MB	11.1 MB
	Reduction	97.87%	91.0%	99.6%
Alerts	Raw Alerts	7502	52815	58715
	Hyper Alerts	21	150	625
	Reduction	99.72%	99.7%	98.9%

Table 3.8 Preprocessing results for the 3 scenarios

	Scenario 1	Scenario 2	Scenario 3
Accuracy	83.3%	71.4%	57.1%
Coverage	100%	100%	71.4%

Table 3.9 Accuracy and coverage for 3 scenarios

attempt in attack step 1.

From Table 3.9 we can see that false negatives in primary evidence significantly degrade the coverage and accuracy of our analysis results. However, our functional and structural metrics still effectively extract the major identities and steps in the attack scenario. Moreover, a strong association between the three attack groups can be observed from the fact that the activation time of *Attacker* state in 74.205.114.158, 42.152.69.166 and 100.20.200.15 closely follows each other.

CHAPTER 4. GLOBAL REASONING WITH SPECTRAL CLUSTERING METHODS

In our hierarchical reasoning framework, global reasoning relies on analyzing structure of the evidence graph, which contains both the coordinated attack and irrelevant background noise. In the following two chapters we present techniques to discover multi-stage intrusions from a graph structure perspective, including spectral clustering and Pagerank methods.

4.1 Motivation

The global reasoning process in Chapter 3 is performed in two phases: the seed generation phase aims to discover important suspicious entities as initial seeds of attack group and the group expansion phase is based on the invariant that entities belong to the attack scenario of interest should be strongly correlated by suspicious relations. We first start with a seed host chosen based on functional state and graph centrality metrics. Following that the attack group is extended by iteratively adding nodes with correlation strength above a predefined threshold as new members of the group. The attack scenario is reconstructed from the extracted subgraph of correlated nodes and corresponding functional states from local fuzzy reasoning. Limitation of this approach is that the reasoning process is inefficient for large scale analysis. Especially, the iterative group expansion phase is computationally expensive for massive graphs.

Intrusion investigation in its most generic form is analogous to finding the needle in the haystack. Now that we have transformed collected evidence into the aggregated evidence graph, advanced graph clustering techniques can be used to discover multi-stage attacks embedded in the graph structure. In recent graph theory [34] and link analysis [26] work, graph spectral methods have been well studied to evaluate topological structure of large complex graphs.

We find graph spectral methods will provide more systematic, flexible and efficient solutions for analyzing structural characteristics of the evidence graph. More specifically, we aim to consolidate the two phases of global reasoning process to provide a single-step solution for *Attack Group Identification*.

In this chapter we present how spectral clustering methods are applied on our evidence graph for generic multi-stage attack investigation.

4.2 Spectral Clustering for Generic Investigation

Clustering methods have been widely used in exploratory data analysis, with the intuition that similar or closely related objects should be grouped together. In the evidence graph space, our motivation is based on the reasoning that coordinated multistage attacks would form significant natural graph clusters, which differentiate them from irrelevant background noise and false positives. Our perception of "natural graph cluster" could be justified by the following two views.

- The Random Walk View: A random walk on a graph could be regarded as a Markov process which randomly jumps from one node to another. Thus graph clustering can be interpreted as finding clusters such that a random walk will mostly stay inside the same cluster and seldom jump between clusters [17]. The analogy to random walks could be found in many multistage attack scenarios where the attacker jumps to the compromised host and initiates attacks in the next step. Automated attacks such as worms are also designed to follow this behavior pattern. Intuitively, the group of hosts involved in a coordinated attack will form a cluster in our aggregated evidence graph such that a random walk will visit most of the corresponding nodes before wandering out.

- The Graph Cut View: From the graph cut view, clustering is to find clusters in the graph structure such that weights of edges between different clusters are minimized while weights of edges within the cluster are maximized [17]. In the aggregated evidence graph, the equivalent interpretation is to find groups of hosts with a higher scale of attack

correlation between group members, while the scale of attack correlation between group members and outsiders is lower.

We can see that the two views are strongly related to each other. In the random walk model, let the edge weight between node v_i and v_j be w_{ij}, then the transition probability of a random walk from v_i to v_j is given by $P_{ij} = w_{ij}/d_i$, where $d_i = \sum_{j=1}^{n} w_{ij}$ is the weighted degree of node v_i. Consequently for clusters that have minimal weights between each other, the probability of a random walk jumping from one cluster to another is also low.

4.2.1 Graph Laplacian Spectrum

Spectral clustering methods are based on the Laplacian spectrum of graphs. It is known that many principal structure properties of a graph are closely related to its graph spectrum [7,34,68]. To characterize the properties of a graph and extract information from its structure, we compute the graph spectrum using its Laplacian matrix representation. Let the aggregated primary evidence graph of size n be denoted by $G_A = (V, E)$ with node set $V = v_1, v_2, ..., v_n$ and edge set $E \subseteq V \times V$. The weighted adjacency matrix A of G_A is defined as:

$$A(i,j) = \begin{cases} \text{w(i,j)}, & \text{if (i,j)} \in E \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Note that the aggregated evidence graph G_A is an undirected graph. Therefore the adjacency matrix A is symmetric and has only real eigenvalues. $w(i,j)$ represents the edge weight between node i and j in G_A. Then the degree matrix D is constructed as a $n \times n$ diagonal matrix with degrees $d_1, d_2, ...d_n$ on the diagonal where degree of a node $d_i = \sum_{j=1}^{n} w(i,j)$. The unnormalized Laplacian representation is given by $L = D - A$. In this work we adopt the normalized graph Laplacian as follows:

$$L_n := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{4.2}$$

The Laplacian spectrum of a graph is given by eigendecomposition of its normalized Laplacian matrix L_n, i.e. $L_n = \Phi \Lambda \Phi^T$ where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ is a diagonal matrix of

eigenvalues and $\Phi = (\phi_1, \phi_2, ..., \phi_{|V|})$ is the matrix composed with eigenvectors as columns. The graph Laplacian spectrum then refers to the set of eigen values $(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ ordered by magnitude, which provides an insight into global structure properties of the graph [7]. For example, the multiplicity of eigenvalue 0 indicates the number of connected components in the graph. Note that in the ideal case when all intrusion activities are captured in primary evidence, hosts involved in the same attack scenario should belong to the same connected component in our aggregated evidence graph G_A.

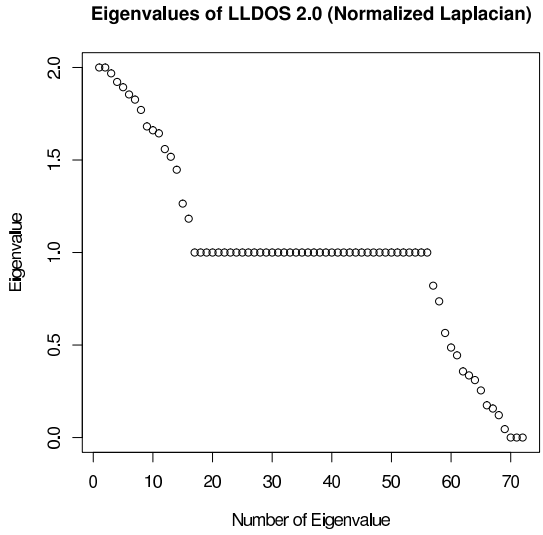


Figure 4.1 Graph Laplacian spectrum for LLDOS 2.0 dataset

Our objective is to extract the attack scenario from large complex evidence graphs through graph Laplacian spectrum analysis. Here an initial experiment is conducted on the Lincoln lab LLDOS 2.0 dataset we analyzed in Section 3.7.2. A heuristic spectral clustering procedure was applied where components of significant magnitude in each of the k leading eigenvectors are extracted as a cluster [65].

Figure 4.1 shows eigenvalues of the aggregated primary evidence graph's normalized Laplacian. We truncate the graph spectrum by picking the leading eigenvectors corresponding to the n smallest eigenvalues $v_1, v_2, ..., v_n$ as they are more informative of graph structure. Distribution of eigenvectors v_1, v_2, v_3, v_4 is shown in Figure 4.2. In each eigenvector we identify the components with significant magnitude and extract corresponding subgraphs from the raw

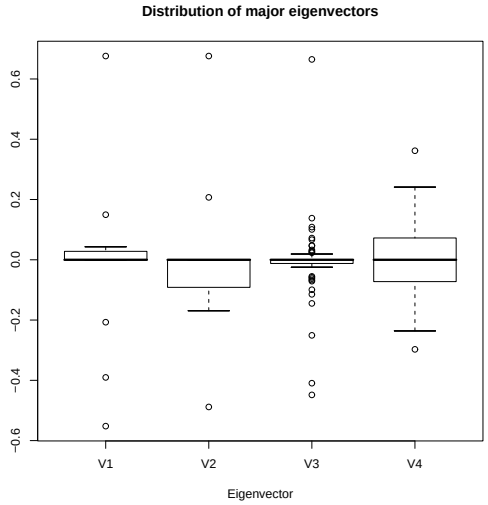


Figure 4.2 Distribution of leading eigenvectors from LLDOS 2.0 graph

evidence graph G_P for validation.

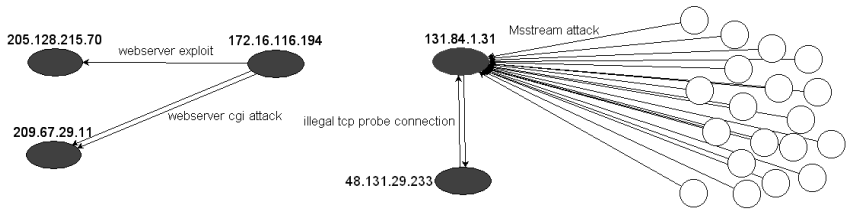
- For principal eigenvector v_1, the corresponding subgraph is shown in Figure 4.3(a). The largest magnitude component in v_1 identifies host 131.84.1.31, which is target of the DDOS attack. An isolated background attack component is also extracted.

- Figure 4.3(b) shows the subgraph corresponding to significant components in eigenvector v_2. We observe that it represents a subset of the components represented in principal eigenvector v_1.

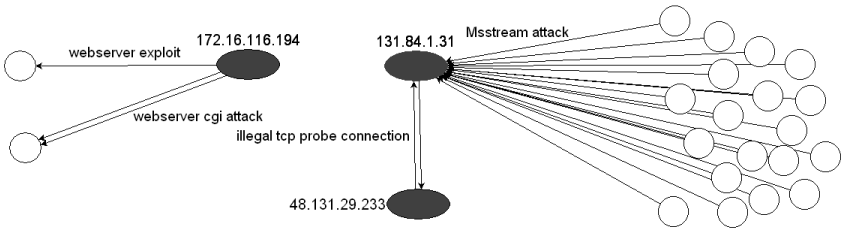
- As can be seen in Figure 4.3(c), components of significant magnitude in eigenvector v_3 maps to several routers and service hosts in the DMZ, which are brought up by background traffic artifacts.

- In eigenvector v_4, we observe that components of most significant magnitudes maps to the two DDOS agent hosts 172.16.115.20 and 172.16.112.50. The other nodes shown in Figure 4.3(d) are also caused by background artifacts.

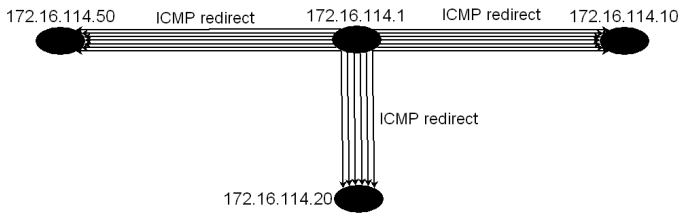
The initial basic graph spectrum analysis shows that important identities of the attack scenario are successfully extracted with the leading eigenvectors. However a more systematic



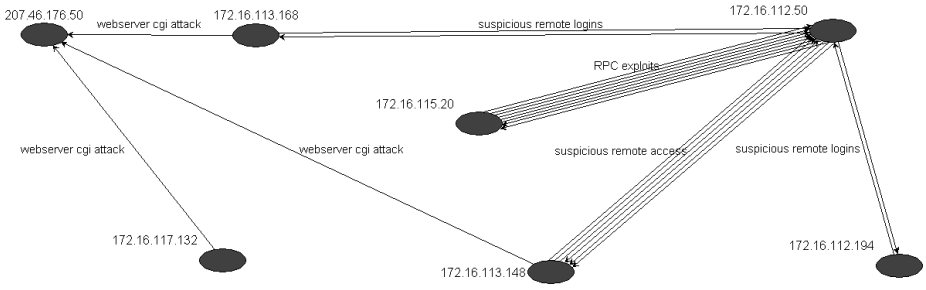
(a) Subgraph for significant components in v_1



(b) Subgraph for significant components in v_2



(c) Subgraph for significant components in v_3



(d) Subgraph for significant components in v_4

Figure 4.3 Sub evidence graphs extracted from leading eigenvectors

clustering scheme is needed to reduce the amount of irrelevant identities in the extracted subgraph.

4.2.2 Recursive Spectral Clustering Algorithm

Generally a clustering scheme on the graph structure is defined by choosing clusters that optimize an objective function of graph partitions. Specifically, spectral clustering is used as an approximation of the optimal normalized edge cut [52]. For two disjoint clusters A and B, edge cut is defined as sum of the edge weights between nodes in cluster A and B, i.e. $cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$. From the graph cut view, the normalized cut proposed by Shi and Malik [52] has the advantage of considering both connectivity between clusters and connectivity inside each cluster to avoid trivial clusters. Its objective function is defined as

$$Ncut(A, B) = \frac{cut(A, B)}{vol(A)} + \frac{cut(A, B)}{vol(B)} \tag{4.3}$$

where vol denotes the total weight of edges originating from nodes in the cluster, i.e. $vol(A) = \sum_{i \in A} d_i$. We note that the normalized edge cut also been proved to have formal equivalence with random walk in graphs [33]. To find a clustering that minimize $Ncut$ has been proved to be NP-hard [17]. However, it has been shown that relaxing $Ncut$ leads to spectral clustering using the normalized Laplacian [52], i.e. spectral clustering can be used as an approximation of the optimal normalized cut. Compared to traditional clustering methods such as the single linkage approach used in [63,64,66], spectral clustering not only often provides better results, but is also more computationally efficient with current linear algebra packages such as LAPACK [2].

Generally speaking, spectral graph clustering can be summarized into three basic stages:

1. Matrix formulation: Construct Laplacian matrix representation of the graph.

2. Spectrum computation: Compute the graph Laplacian spectrum by eigen decomposition. With k eigenvectors the graph is represented in the subspace spanned by the selected eigenvectors. In essence, this is to map nodes in the graph to points in a lower dimensional space.

input : Aggregated primary evidence graph $G_A = (V_A, E_A)$, eigenvector magnitude threshold P, minimal graph componet size threshold S
output: Primary attack clusters C_P

begin
  if G_A *has more than one connected component* **then**
    divide G_A into its components and recurse on each component;
  end
  if $|V_A| < S$ **then**
    stop and return;
  end
  Construct normalized Laplacian matrix L_n of G_A;
  Compute spectrum of L_n and select the principal eigenvector v_1;
  Initialize set of nodes $N \leftarrow \emptyset$, set of edges $E \leftarrow \emptyset$;
  $N \leftarrow$ set of nodes corresponding to points in principal eigenvector v_1 with magnitude greater than P;
  $E \leftarrow$ set of edges in G_A with both endpoints in N;
  Output $C_P \leftarrow (N, E)$;
  $V'_A \leftarrow V_A \setminus N$,$E'_A \leftarrow E_A \setminus E$;
  Recurse on $G'_A \leftarrow (V'_A, E'_A)$;
end

Algorithm 5: Recursive spectral clustering algorithm

3. Cluster assignment: Form clusters of points in the lower dimensional space and assign corresponding cluster membership to nodes in the original graph.

Several spectral clustering algorithms have been proposed in past work [8, 17]. For the normalized Laplacian L_n, the leading eigenvectors that correspond to the smallest non-zero eigenvalues are generally used in graph clustering [7]. If the desired number of clusters k is known, a common approach is to utilize the normalized Laplacian matrix L_n in step 1, select k leading eigenvectors from the Laplacian spectrum in step 2 and use the traditional k-means clustering algorithm on the $R^{n \times k}$ matrix to form k clusters in the lower dimensional space. However, in general intrusion analysis we have no prior knowledge for the optimal size of cluster or desired number of clusters.

In this work we develop a recursive spectral clustering approach to avoid the bias of selecting an arbitrary number of clusters. The detailed procedure is presented in Algorithm 5. As illustrated in Figure 4.4(a), the initial round of clustering is performed on Laplacian spectrum

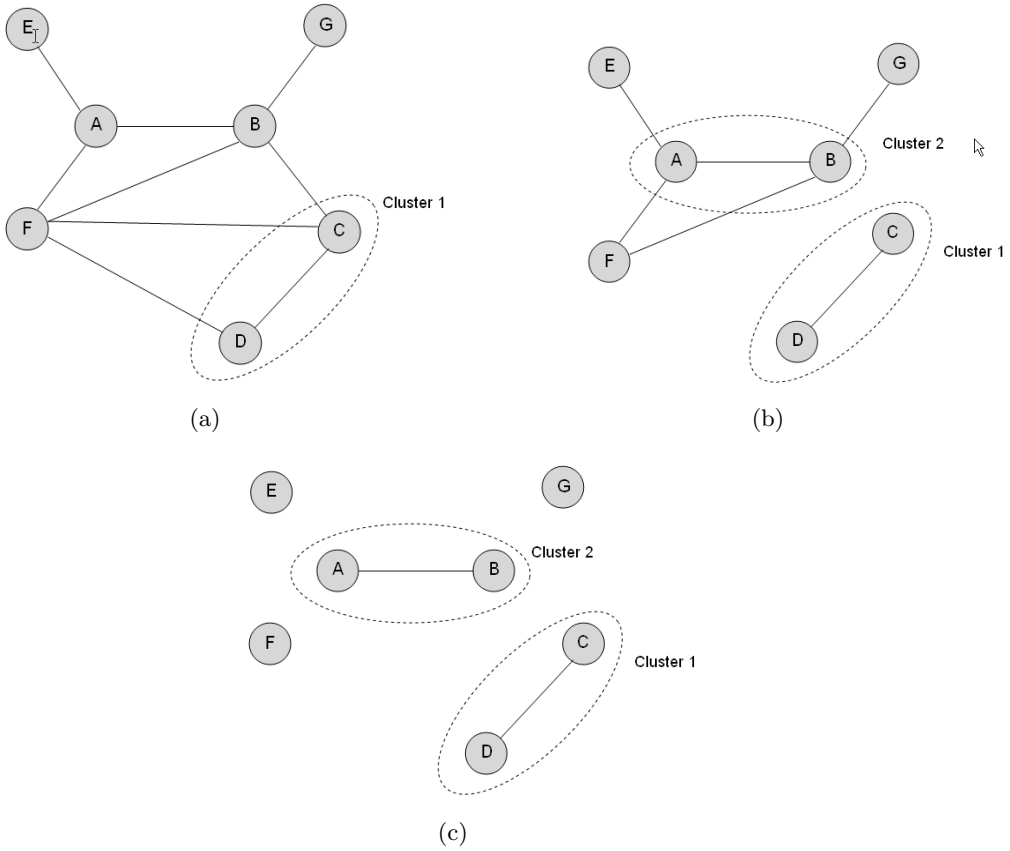


Figure 4.4 Recursive spectral clustering illustration

of the original aggregated primary evidence graph G_A. One significant cluster is extracted from the principal eigenvector v_1, i.e. eigenvector corresponding to the smallest non-zero eigenvalue λ_1. As shown in figure 4.4(a) and figure 4.4(b), the extracted significant cluster is then truncated from the original aggregated evidence graph G_A. Consequently the result graph G'_A could be consisted of multiple disconnected components. Then in the next round the Laplacian spectrum is recomputed for each of the remaining non-trivial graph component whose size of each graph component is above the minimum graph component size threshold S. The recursive clustering process continues until all remaining graph components are too trivial for inspection. The significant clusters extracted are candidates of potential coordinated attack scenarios for further evaluation.

In summary major objective of our spectral clustering approach is to identify **coordinated**

multi-stage attack scenarios because they naturally form significant clusters in the aggregated primary evidence graph. Throughout the process we are not relying on any a priori knowledge of host vulnerability or causal dependency between exploits to prune away background noise and false positives. However if the attack process is just a "single shot" exploit, our approach would not be effective as the cluster structure would not be significant and additional verification is needed to differentiate the genuine atomic attack from background noise.

4.2.3 Experimental Results

We evaluate our spectral clustering approach with the attack dataset prepared under ARDA contract No. NBCHC030117 as well as the MIT Lincoln Lab DARPA 2000 LLDOS1.0 dataset.

The evaluation metrics used here are defined to solely focus on structure-based clustering results. We perform receiver operating characteristics (ROC) analysis to evaluate performance of the proposed spectral clustering approach, which is based on two quantitative metrics: true positive rate and false positive rate. If a host is extracted in clustering and it is actually a member of the true attack group, it is counted as a true positive; if the host is not a member of the true attack group but is included in the extracted cluster, it is counted as a false positive. As both datasets contain a ground truth file that could be used to check against our clustering results, we calculate true positive rate and false positive rate as follows. The set of hosts correspond to the set of nodes in the graph component on which spectral clustering is performed.

$$\text{True positive rate} = \frac{\#\text{ of true positives}}{\#\text{ of hosts involved in the true attack scenario}} \tag{4.4}$$

$$\text{False positive rate} = \frac{\#\text{ of false positives}}{\#\text{ of hosts irrelevant to the true attack scenario}} \tag{4.5}$$

4.2.3.1 Experiment Results with ARDA Bankshot Dataset

Here we first apply spectral clustering to reexamine the ARDA "Bankshot" dataset described in Section 4.2.3.1. According to the ground truth, the multi-stage attack consists of the following phases:

(1) The attacker probes the protected domain 100.10.20.0/24 from 74.205.
114.158 and launches several failed exploit attempts from 42.152.69.166 against a Windows host 100.10.20.4;

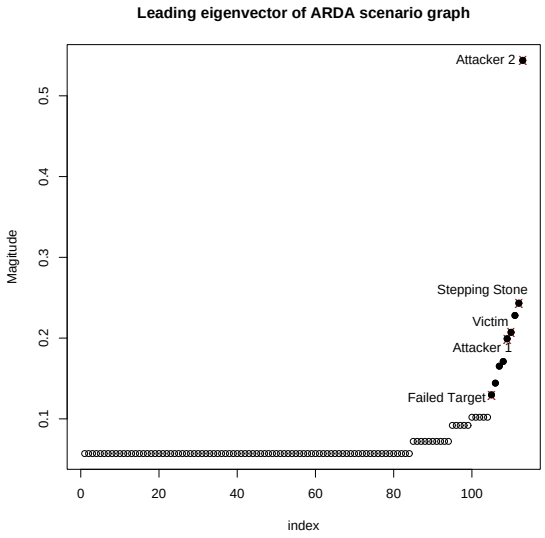
(2) The attacker probes the network from 74.205.114.158, then successfully compromises a Linux host 100.20.200.15 through a local Apache exploit.

(3)The attacker uses 100.20.200.15 as the launching point to probe hosts inside the trusted domain 100.10.20.0/24, then launches exploits to break into host 100.10.20.8.

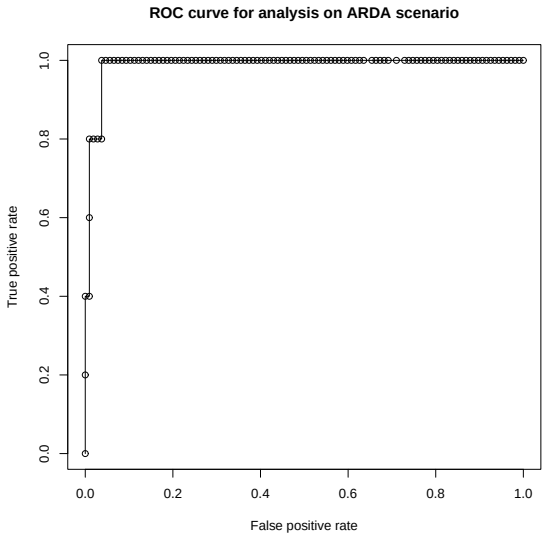
For convenience of illustration we denote the following labels for important hosts in the attack truth. *Attacker 1*: 42.152.69.166; *Attacker 2*: 74.205.114.158; *Failed Target*: 100.10.20.4; *Victim*: 100.10.20.8 and *Stepping Stone*: 100.20.200.15.

Through the preprocessing scheme presented in Algorithm 1, a total of 58715 Snort alerts are reduced into 625 hyper alerts for constructing the initial evidence graph G_P, which consists of 205 nodes and 625 edges. After performing graph aggregation based on attack phases and relevant resources as described in Algorithm 3, the aggregated evidence graph G_A consists of 205 nodes and 356 edges. We also observe that in G_A, 30 out of a total of 31 connected components are of trivial size. Therefore spectral clustering is applied on the only significant component C_{10}, which contains 112 nodes and 232 edges.

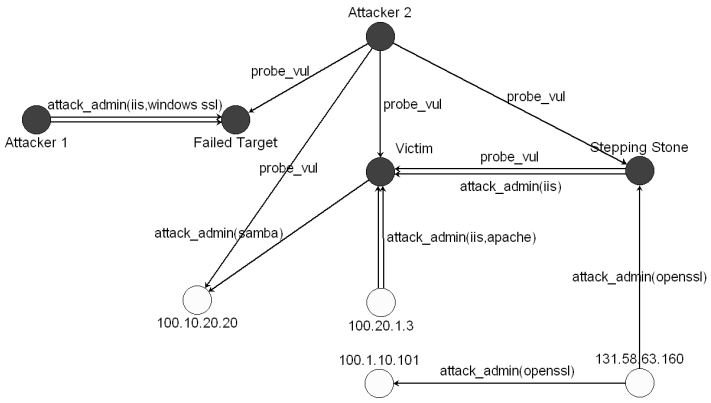
Following Algorithm 5, we perform spectral clustering on the principal eigenvector v_1 from Laplacian spectrum of C_{10}. As shown in Figure 4.5(a), the set of highlighted nodes are the cluster we extracted in the very first iteration and the recursion terminates as all remained graph components are trivial. We can see from the cross marked nodes that hosts involved in the attack truth have significant magnitudes in the principal eigenvector v_1. Just for further verification and illustration purpose, the sub-evidence graph corresponding to the extracted cluster is shown in Figure 4.5(c). We can see that all hosts involved in the attack scenario are



(a) Principal eigenvector v_1 for component C_{10}



(b) ROC curve of clustering on v_1



(c) Cluster extracted from principal eigenvector v_1

Figure 4.5 Spectral clustering analysis of component C_{10} in ARDA graph

successfully identified through our spectral clustering approach. The local Apache exploit on *Stepping Stone* is not detected because only network based evidence is used in analysis. Though the four hosts 100.10.20.20, 100.20.1.3, 100.1.10.101 and 131.58.63.160 are false positives caused by background noise, we will perform further filtering in cluster evaluation phase. True positive rate for the extracted cluster is 100% and false positive rate is 3.7%. To evaluate overall performance of spectral clustering on the ARDA attack dataset, we plot the ROC curve shown in Figure 4.5(b), which illustrates the trade off between true positives and false positives.

4.2.3.2 Experiment Results with LLDOS1.0 Dataset

The MIT Lincoln Lab DARPA 2000 LLDOS 1.0 dataset presents a multi-stage attack scenario that consists of five phases. For convenience, we denote the following labels for important hosts in the attack truth. Origin of the exploits is *Attacker*: 202.77.162.213. The three hosts that have been successfully compromised in step 3 are *Victim_A*: 172.16.112.50, *Victim_B*: 172.16.112.10 and *Victim_C*: 172.16.115.20. The final target of DDoS attack is *Victim_D*: 131.84.1.31.

1. Network Sweep: The *Attacker* performs IP sweeps to discover live hosts in the target networks.

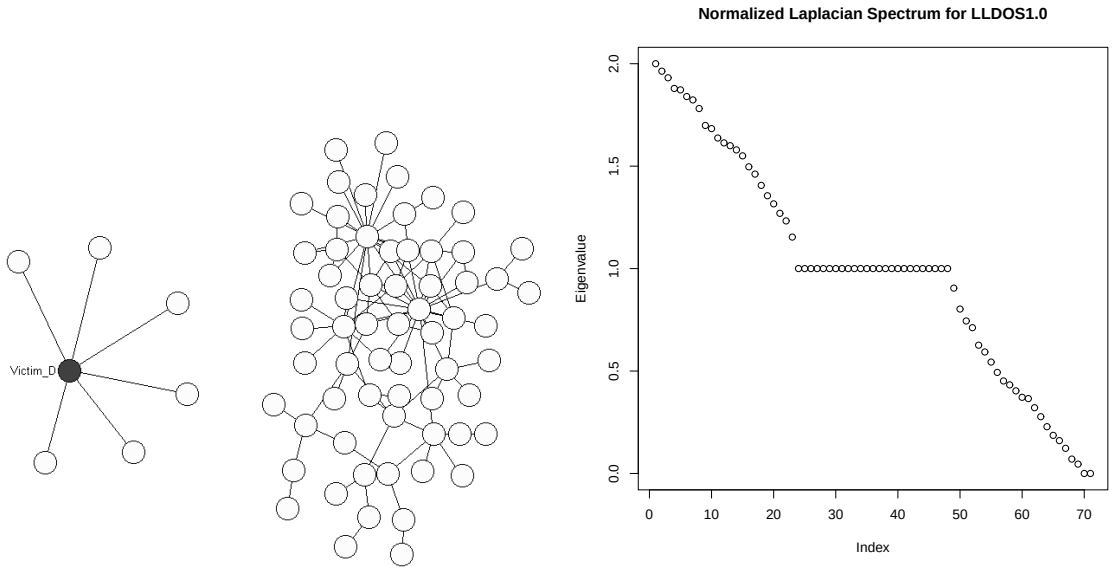
2. Vulnerability Scan: The *Attacker* probes the live hosts discovered to find exploitable Sadmind daemons.

3. Break in: The *Attacker* breaks into *Victim_A*, *Victim_C* and *Victim_C* through the Sadmind vulnerability.

4. Installation: The *Attacker* installs Mstream DDoS software on the three compromised hosts.

5. DDoS Attack: The *Attacker* access the compromised hosts by telnet and initiates DDoS attack towards *Victim_D*.

The attack lasts around 3 hours and about 200M tcpdump packet traces are collected from the DMZ and inside network, most of which are innocuous background traffic. Through



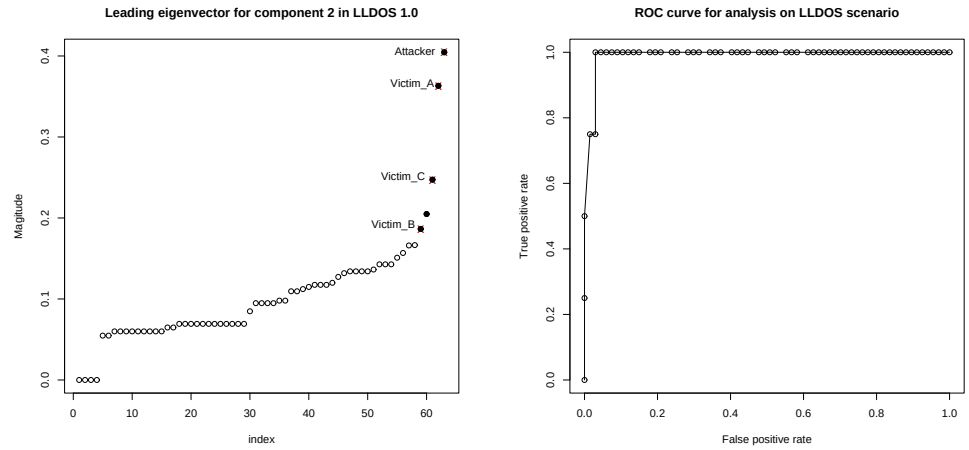
(a) Aggregated primary evidence graph (b) Laplacian spectrum of the whole graph

Figure 4.6 Evidence graph and Laplacian spectrum for LLDOS 1.0 dataset

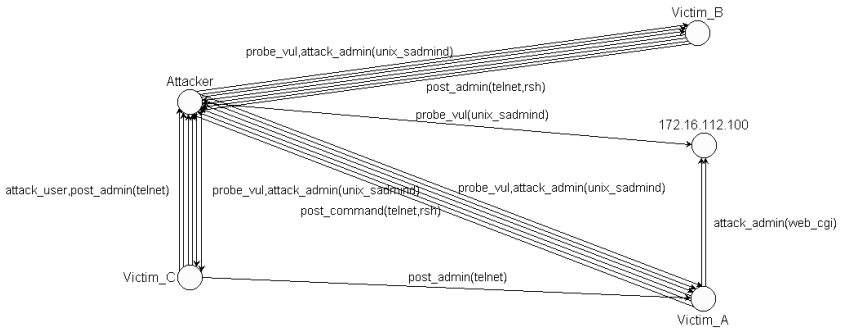
preprocessing, a total of 36555 Snort alerts are reduced into 291 hyper alerts for constructing the initial raw primary evidence graph G_P, which consists of 71 nodes and 291 edges. After graph aggregation as described in Algorithm 3, the aggregated evidence graph G_A consists of 71 nodes and 100 edges.

Figure 4.6(b) shows eigenvalues of the normalized Laplacian for the aggregated evidence graph G_A. The 2 zero eigenvalues denotes that there exist two disconnected components in the graph. We denote the smaller graph component as C_1 and the larger component as C_2. Intuitively we can see that the trivial component C_1 exhibits a star pattern centered around *Victim_D*, which represents the DDoS attack. Note that C_1 is disconnected from C_2 because of spoofed IP addresses used in the DDoS attack. Therefore we move on to perform spectral clustering on component C_2.

Figure 4.7(a) shows the principal eigenvector v_1 from the Laplacian spectrum of C_2, from which we expect to extract important entities in the coordinated attack scenario. Following Algorithm 5, the cluster of highlighted nodes are extracted in the first iteration and the recursion terminates. Graph nodes corresponding to $Attacker, Victim\_A, Victim\_B$ and $Victim\_C$ have



(a) Principal eigenvector v_1 of component C_2 (b) ROC curve of clustering on v_1



(c) Cluster extracted from principal eigenvector v_1

Figure 4.7 Spectral clustering analysis component C_2 in LLDOS 1.0 graph

significant magnitudes in the principal eigenvector v_1 and can be easily extracted in clustering. Just for verification purpose, the sub-evidence graph correspond to the extracted cluster is shown in Figure 4.7(c), which successfully identifies the complete coordinated attack scenario except for the DDoS stage. Host 172.16.112.100 is the only false positive extracted in the cluster. Therefore in graph component C_2, the true positive rate is 100% and false positive rate is 1.7%.

An interesting observation is that relaxing the threshold leads to an immediate set of false positives 172.16.114.50, 172.16.114.20, 172.16.114.30, 172.16.114.10 and 172.16.114.1, which are routers and service hosts in the DMZ. By examining the primary evidence graph we observe a large number of ICMP related alerts among these servers, which is not related to the

foreground attack. To our best knowledge, this non-malicious anomaly has not been uncovered in previous analysis. From this observation we note that clusters extracted from primary evidence could represent benign anomalies such as network misconfiguration. In Chapter 6 we present automated cluster evaluation to differentiate the benign anomalies from true coordinated attack clusters.

4.3 Graph Perturbation Analysis

We have shown that entities involved in a coordinated attack can be effectively extracted with spectral clustering. For validation and comparable evaluation with previous works, experimental results presented are based on public synthetic attack datasets. In many cases, the major controversy with artificial datasets is not the arrangement of sophisticated foreground attacks, but the robustness of background traffic. In real-world networks, the noise level in background traffic varies dramatically and the number of false positives in IDS alerts could be overwhelming. Therefore an immediate concern comes forth: How would the spectral clustering approach perform in a more adverse operational environment? In particular, how would the increase of false positives affect the effectiveness of clustering results? In order to evaluate our spectral clustering approach with regard to different levels of background noise, we first analyzed the scenarios of false positives in evidence graph context. Based on the analysis, we developed a graph perturbation algorithm to inject simulated noise into an given evidence graph. Robustness of spectral clustering results is then quantitatively evaluated at multiple noise levels.

4.3.1 Background Noise Analysis

An ideal evidence graph consists of true attacks, and only true attacks. In other words, no false positives are built into the evidence graph and no true positives are missing. Consequently natural connected components would readily reveal entities and events in the true attack scenario. However, in real world networks attack traffic is generally a tiny drop in the tide of regular background traffic. In most IDS deployments, the amount of false positives triggered

by innocent network activities are often overwhelming. Intuitively, each false positive would result in a spurious edge in our raw primary evidence graph. As the aggregated evidence graph is constructed such that each undirected edge is a numeric representation of corresponding edge(s) in the raw evidence graph, its structure would also be affected in different ways.

Let G be the aggregated evidence graph constructed from primary evidence to perform spectral clustering. Then as shown in Figure 4.8, G is consisted of two distinctive components: the ideal component S, which is generated from true positives; and the noise component N, which represents IDS false positives generated by background traffic. The result graph G is then decided by the correlation between S and N. To evaluate the robustness of our spectral clustering approach, we first study how S would be affected by different categories of background noise in the evidence graph model.

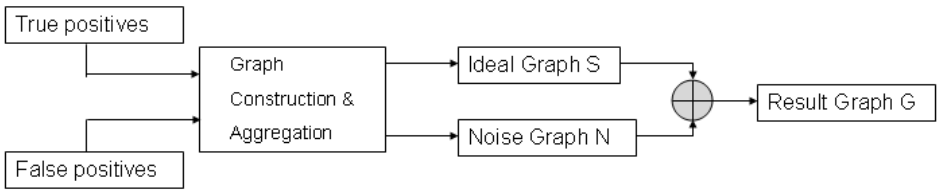


Figure 4.8 Background noise in evidence graph model

When false positives are mingled with true alerts, the aggregated evidence graph could be affected in two aspects: (1) topology and (2) correlation strength. Topology changes refer to adding new nodes and edges to the original graph structure while correlation strength changes refer to weight changes of existing edges. In accordance we differentiate background noise from the following two perspectives: (1) locality correlation and (2) functional correlation.

4.3.1.1 Locality Correlation

Let the ideal evidence graph be $S = (n_t, e_t)$ where n_t is the set of entities in the true attack scenario and e_t represents the set of true attack events. For the example shown in Figure 4.9(a), $n_t = \{n_1, n_2, n_3, n_4\}$ and $e_t = \{e_1, e_2, e_3, e_4, e_5\}$. Similarly, e_f represents a false positive alarm and its associated entities are denoted as n_f. Based on locality relationship between e_f

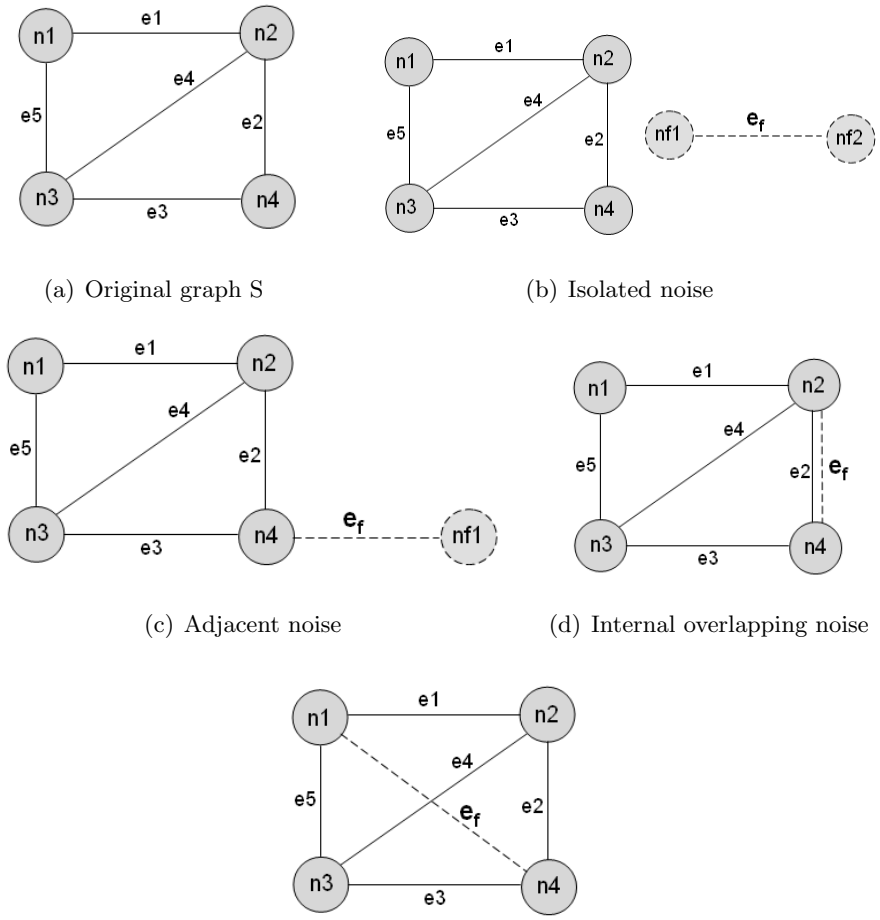
and S, background noise can be classified into the following types:

- Isolated noise: As shown in Figure 4.9(b), the noise edge e_f is completely separated from the true cluster S.
- Adjacent noise: In Figure 4.9(c), the noise edge e_f is adjacent to the true cluster S.
- Internal overlapping noise: As shown in Figure 4.9(d), the noise edge e_f locates inside the true cluster S and overlaps with existing true edge e_2.
- Internal independent noise: As shown in Figure 4.9(e), the background noise edge e_f locates inside the true cluster S but does not overlap with any existing true edge.

The aforementioned four types of background noise have different implications to graph clustering.

Firstly, *Isolated Noise* refers to the scenario where neither source nor target of the false positive belongs to the true attack cluster. Essentially the spurious edges added to the raw evidence graph is isolated from the true cluster. In the aggregated evidence graph both topology and weight of the true cluster are unaffected. Note that when multiple instances of isolated noise are coincidentally connected, spurious clusters of significant size could be created. However, as our spectral clustering is performed independently on each individual graph component, clustering results for the true attack scenario could only be masqueraded when the spurious cluster resides in the same graph component with the true cluster. Spurious clusters in different graph components would not affect the true cluster.

Secondly, *Adjacent Noise* refers to the scenario where either source or target of the false alert belongs to the true attack cluster. In other words false positives in this category are triggered between entities in the true attack group and their innocent neighbors. In this scenario, topology and correlation strength of existing edges inside the true cluster are not affected. However, connection density between the true cluster and innocent neighbor nodes would be increased, which potentially could increase the structural importance of these noise nodes. Note that structure importance of the innocent neighbor depends on the number and



(a) Original graph S

(b) Isolated noise

(c) Adjacent noise

(d) Internal overlapping noise

(e) Internal independent noise

Figure 4.9 Noise classifications by locality correlation

weight of edges connecting to hosts in the true attack cluster. For example, when one innocent neighbor has false positives associated with multiple members of the true attack group, it is highly likely to be extracted as a member of the true cluster. Consequently *Adjacent Noise* would lead to higher false positive rates in spectral clustering results.

Thirdly, *Overlapping Internal Noise* refers to the scenario where (1) both source and target of false alerts are members of the true attack cluster and (2) true attack events already exist between the source and target. Therefore in the raw evidence graph the noise edge overlaps with true edges. In the aggregated evidence graph, topology of the true cluster remains the same. However, correlation strength between the source and target host could be changed.

The new correlation strength would be decided by the combination of true and noise edges, i.e. how the attack phase and resource of the overlapping noise edge are correlated with that of original true edges. In next step we will further discuss this scenario from the perspective of functional correlation.

Finally, *Independent Internal Noise* refers to the scenario where (1) both source and target of false alerts are members of the true attack group and (2) there exist no true attack events between the source and target. Therefore in the raw evidence graph the noise edge is independent to existing true alarm edges. In the aggregated evidence graph, topology of the true cluster is changed such that additional spurious edges are created between existing nodes. Aggregated weight of the spurious edge is solely decided by attack phase and resource definition of the false positive. Note that *Independent Internal Noise* increases internal intensity of the true cluster and potentially would make the true cluster more structurally significant in the graph.

4.3.1.2 Functional Correlation

In the aggregated evidence graph, weight of an edge is computed through the attack phase and relevant resource based aggregation process to represent the combined effect of attacks between the pair of hosts. Let a pair of hosts in the true attack cluster be $n1, n2$ and the set of true positives between them be e_t. A false positive edge e_f between $n1, n2$ can be classified in terms of its functional correlation with the set of true positives e_t:

- Uncorrelated noise: The false positive edge e_n has no functional correlation with e_t , i.e. the noise event neither advances attack phases nor extends relevant resources of true attack events. Therefore uncorrelated noise would be ignored in the attack phase and resource based aggregation process. Aggregated edge weight in the evidence graph would not be affected.

- Correlated noise: The false positive edge e_n is functionally correlated to e_t, i.e. the noise event advances attack phase or extends relevant resources of true attacks. Therefore

correlated noise would be merged with true attack events and consequently increase the edge weight in aggregated evidence graph.

4.3.1.3 Integrated Classifications

Both locality and functional correlation types are defined based on relationship between false positives and true positives. We do not explicitly consider the correlation between instances of false positives. Functional correlation between false positives would increase aggregated weight of spurious edges. Locality correlation among random false positives could result in significant graph clusters. However, noise clusters formed purely by false positives are not mingled with true attack clusters. After graph clustering, we perform semantic scenario evaluation on each extracted cluster to filter out the noise clusters.

From an integrated view of both locality and functional correlations, background noise in the evidence graph model can be further classified into the categories shown in Table 4.1.

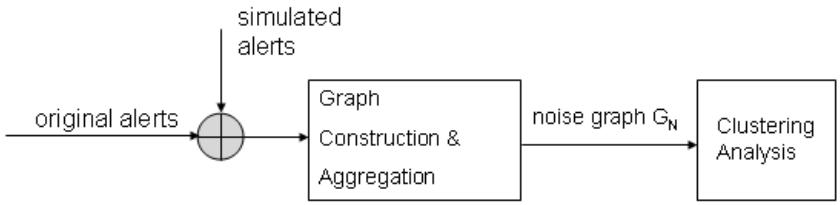
Category	Isolated	Adjacent	Overlapping Internal	Independent Internal
Uncorrelated	✓	✓	✓	✓
Correlated	N/A	N/A	✓	N/A

Table 4.1 Integrated noise categories

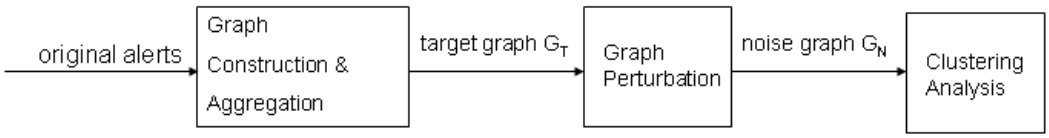
By definition, *Uncorrelated* is the only functional correlation type applicable to locality correlation categories *Isolated Noise*, *Adjacent Noise* and *Independent Internal Noise*. On the other hand, both *Uncorrelated* and *Correlated* types of functional correlation are applicable to *Overlapping Internal Noise*. *Uncorrelated and Overlapping Internal Noise* does not affect the topology or edge weights of true attack cluster. *Correlated and Overlapping Internal Noise* would increase the aggregated weight of true attack edges.

4.3.2 Noise Simulation through Graph Perturbation

To evaluate spectral clustering results at increased noise levels, we develop a simulation approach that could conveniently inject specified level and type of background noise into the given attack dataset.



(a) Noise injection by raw false alerts



(b) Noise injection by graph perturbation

Figure 4.10 Approaches for injecting noise into given attack dataset

As shown in Figure 4.10, there exist two approaches for noise simulation. In the first approach, simulated noise is generated as random raw IDS alerts. Attack phase and relevant resource based aggregation is then performed on the mixed set of injected false positives and original true positives to construct the aggregated graph. Noise generation in this approach would be more complicated and less controllable. To satisfy functional correlation requirements, the random alerts need to be selected based on the attributes of existing true positives. On the other hand, any variation in raw alerts would eventually be represented by structure and weight changes in the aggregated evidence graph to affect spectral clustering results. Therefore in a second approach, we can generate background noise by directly perturbing the original aggregated evidence graph. Fusing background noise with the true attack scenario is then transformed into perturbation operations on the original aggregated evidence graph. As seen in Figure 4.8, G can also be regarded as the result of a series of perturbation operations on the ideal graph component S.

Algorithm 6 presents our graph perturbation process, which is inspired by the approach proposed in [62]. For coverage and controllability considerations, parameters of our graph

input : Target graph component $G_T = (V_T, E_T)$, weight scale factor w, size scale factor s
output: Perturbed graph component $G_N = (V_N, E_N)$

begin
  $G_N \leftarrow G_T$;
  $num \leftarrow |V_N| \times s$;
  repeat
    Pick a random node n_{p1} from V_N ;
    Pick a random node n_{p2} from $V_N \setminus n_{p1}$;
    Create edge $e_{curr} = (n_{p1}, n_{p2})$ with weight scale factor w ;
    if *there exists edge* e_{prev} *between* n_{p1} *and* n_{p2} **then**
      $weight_{e_{curr}} = weight_{e_{curr}} + weight_{e_{prev}}$;
      $E_N = E_N \setminus e_{prev}$;
    end
    $E_N = E_N \bigcup e_{curr}$;
    $num \leftarrow num - 1$;
  until $num > 0$;
  Output $G_N = (V_N, E_N)$;
end

Algorithm 6: Graph perturbation for noise simulation

perturbation algorithm are defined as follows:

- Target component G_T: As spectral clustering is performed individually on each connected graph component in the aggregated evidence graph, target component is the graph component that contains the true attack cluster.
- Weight scale factor w: The weight scale factor w controls the weight of an newly created perturbation edge, which is calculated as random uniform noise $U(0, 1)$ scaled by 10^w.
- Size scale factor s: The size scale factor s is the ratio between number of perturbation edges to be generated and number of edges in the target component. Higher s denotes that more spurious edges would be created.

The objective of Algorithm 6 is to inject simulated background noise into an existing target graph component which contains the true attack cluster. Structure of G_T would be changed by the different categories of noise generated in permutation. From aforementioned analysis we can summarize that:

- *Isolated Noise* could potentially form significant independent cluster in the perturbed component.
- *Adjacent Noise* would increase the inter-cluster density between the true cluster and its neighboring nodes in the perturbed component.
- Both *Overlapping Internal Noise* and *Independent Internal Noise* would increase the intro-cluster density of true attack cluster in the perturbed component.

We can see that *Isolated Noise* and *Adjacent Noise* would result in higher level of masquerading to the true cluster and adversely affect spectral clustering results. On the other hand, *Overlapping Internal Noise* and *Independent Internal Noise* are benign for extracting the true cluster.

Noise Type	Probability
Isolated	$(1 - \frac{N_{true}}{N_{target}})^2$
Adjacent	$\frac{N_{true}}{N_{target}} \times (1 - \frac{N_{true}}{N_{target}})$
Overlapping Internal	$(\frac{N_{true}}{N_{target}})^2 \times (\frac{E_{true}}{{}_{N_{true}}C_2})$
Independent Internal	$(\frac{N_{true}}{N_{target}})^2 \times (1 - \frac{E_{true}}{{}_{N_{true}}C_2})$

Table 4.2 Distribution of noise categories in graph permutation

In the perturbation process presented in Algorithm 6, connectivity of the target graph G_T is maintained such that the perturbed graph G_N remains a connected graph. Let the number of nodes in target component be N_{target}, the number of nodes in the true cluster be N_{true} and the number of true attack edges be E_{true}. Distribution of different categories of noise generated in graph permutation is shown in Table 4.2, where ${}_{N_{true}}C_2 = \frac{N_{true}!}{2(N_{true}-2)!}$.

4.3.3 Experimental Results

We perform noise simulation analysis on the ARDA "Bankshot" attack dataset described in section 4.2.3.1. The only significant component C_{10} in the aggregated evidence graph is used as the target component for perturbation as it contains the true attack cluster. The ARDA dataset already contains background noise at the level of real-world traces collected

from a large corporate network. We further inject background noise by graph perturbation to evaluate the robustness of spectral graph clustering results.

In following experiments, we apply ROC analysis to evaluate clustering results with respect to different noise levels through variations in weight scale factor w and size scale factor s.

4.3.3.1 Variation in weight scale factor

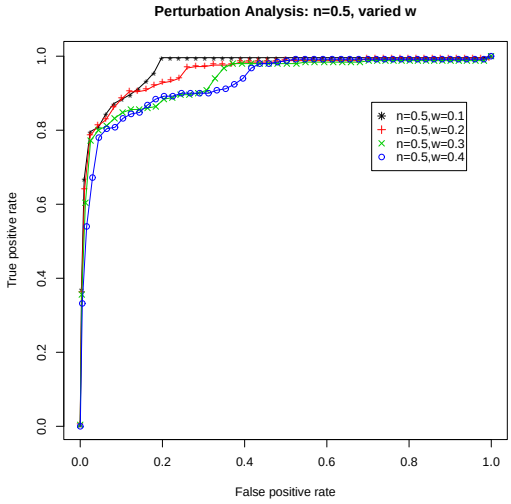
Performance of spectral graph clustering w.r.t variations in weight scale factor w is shown in Figure 4.11. In Figure 4.11(a), 4.11(b), 4.11(c)and 4.11(d), the ROC curves are obtained with w increasing from 0.1 to 0.4 in steps of 0.1, i.e. weight of perturbation edges are random uniform noise $U(0,1)$ scaled by increasing magnitude from $10^{0.1}$ to $10^{0.4}$. In each figure the size scale factor n is fixed as $n = 0.5$, $n = 1$, $n = 1.5$, $n = 2$ respectively. Each curve in figure 4.11 is the average of 50 random perturbations with the specified set of (n, w).

4.3.3.2 Variation in size scale factor

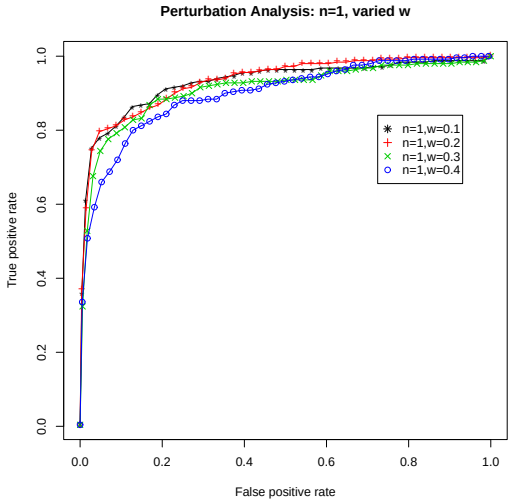
Performance of spectral graph clustering w.r.t variations in size scale factor n is shown in Figure 4.12. In Figure 4.12(a), 4.12(b), 4.12(c) and 4.12(d), the ROC curves are obtained with n increasing from 0.5 to 2 in steps of 0.5, i.e. the number of perturbation edges injected are increased from 50% In each figure the weight scale factor w is fixed as $w = 0.1$, $w = 0.2$, $w = 0.3$ and $w = 0.4$ respectively.

As shown in Figure 4.11 and Figure 4.12, our spectral clustering approach demonstrates robustness to different types of injected background noise. In all perturbation scenarios with different combinations of n and w, we are still able to extract the cluster of coordinated attackers at acceptable cost. For example in Figure 4.12(a), when the number of noise edges in the original graph component C_{10} is further increased by 60%(i.e. $n = 0.6$), spectral clustering is able to achieve true positive rate of 86% before the false positive rate reaches 9%. Even at a higher size scale factor $n = 1.5$, true positive rate remains over 80% with false positive rate below 10%. In Figure 4.11(a), similar results can be observed at a high weight scale $w = 0.4$. As expected, performance of spectral clustering gradually degrades with the increase of n and

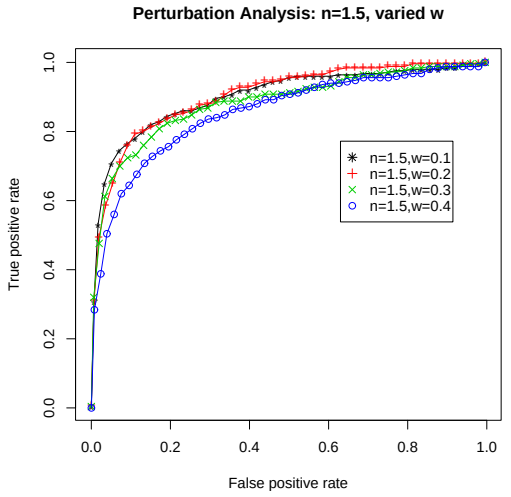
w. However, with our phase and resource based aggregation scheme, random background attacks and false positives are not likely to result in high aggregated weights as coordinated attacks do, therefore strengthens the immunity of clustering analysis to background noise.



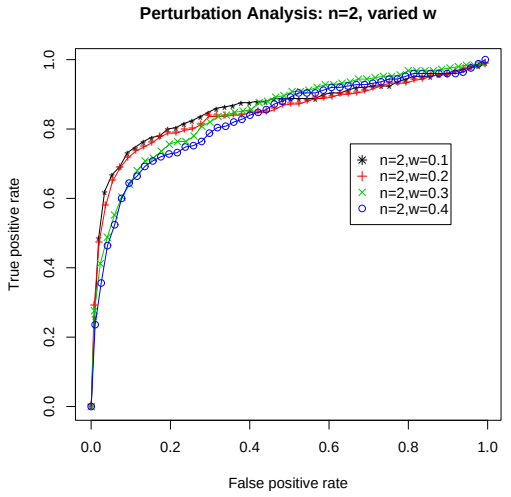
(a) $n = 0.5$, varied w



(b) $n = 1$, varied w

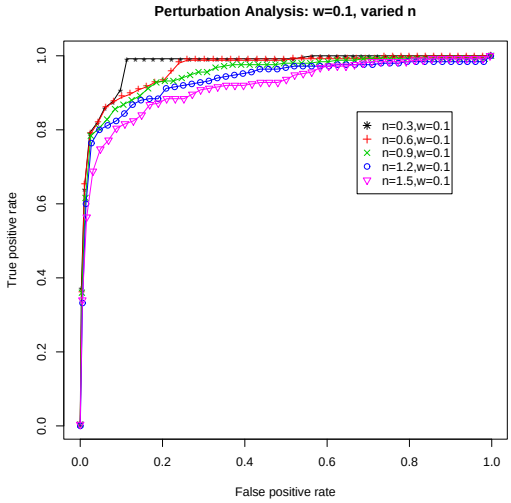


(c) $n = 1.5$, varied w

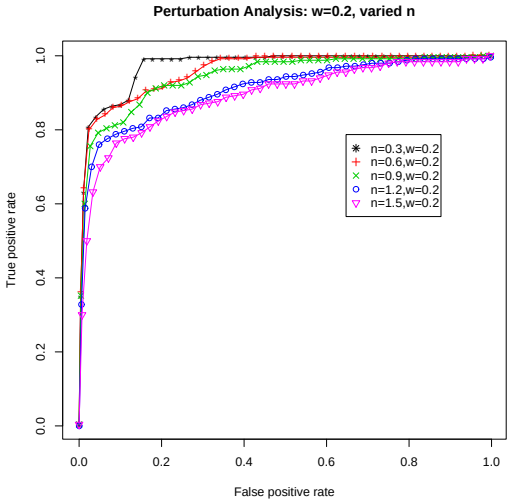


(d) $n = 2$, varied w

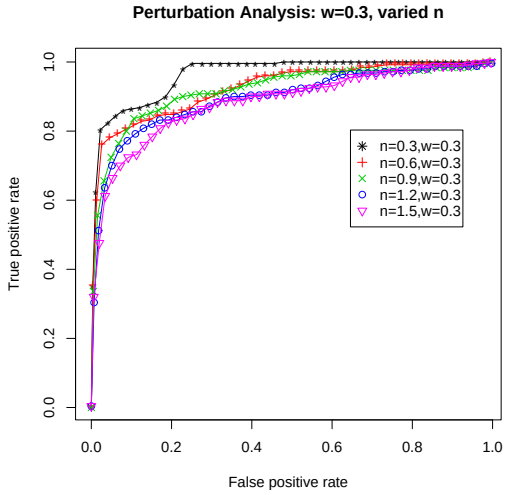
Figure 4.11 Perturbation analysis on ARDA graph: variations in w



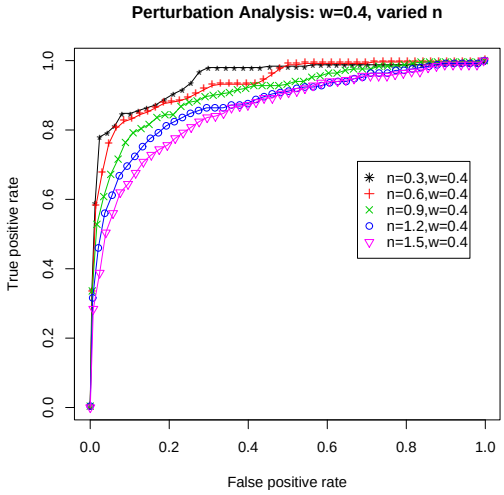
(a) $w = 0.1$, varied n



(b) $w = 0.2$, varied n



(c) $w = 0.3$, varied n



(d) $w = 0.4$, varied n

Figure 4.12 Perturbation analysis on ARDA graph: variations in n

CHAPTER 5. PAGERANK MODEL FOR TARGETED INVESTIGATION AND HIDDEN MEMBER IDENTIFICATION

5.1 Background and Motivation

In Chapter 4 we note that natural graph clusters formed by multi stage attacks could be justified by the analogy to a *Random Walk*. In the evidence graph model, each edge represents a potential path the attacker could take, i.e. the attacker initiates attacks from the source node and moves forward as he takes possession of the compromised target node.

A similar notion has been followed by the web graph which models the World Wide Web structure. In the Pagerank model proposed by Page and Brin [30], web graph is viewed as representation of a first order Markov process. In the web graph model, states of the Markov process are represented by nodes(web pages) and transitions are represented by edges(hyperlinks). In the adjacency matrix P of the web graph, each element P_{ij} denotes the transition probability of the surfer moves from node v_i to v_j. If all outgoing edges of v_i are treated equal then $P_{ij} = \frac{1}{d^i_{out}}$ for each node v_j that has an incoming edge from v_i. The Pagerank score is calculated based on the transition matrix P as follows:

$$\overline{P} = \alpha P + \frac{(1-\alpha)ee^T}{n} \tag{5.1}$$

where e is a vector of all ones and n is size of the web graph. In Equation 5.1, $\alpha \in [0, 1]$ is called the *damping factor*, which represents the probability that the surfer continue to successors of current page. Likewise $1-\alpha$ denotes the probability that the surfer stops clicking links on current page and restarts from a random new page. Generally, each node in the graph is assigned an uniform probability $\frac{1-\alpha}{n}$ that it will be chosen when the surfer makes a fresh

new start. Let $PR = (r_1, r_2, \cdots, r_n)$ be the dominant eigenvector of $\overline{P}$, then r_i is the Pagerank value for node i, which represents its global importance in the web graph.

We note that in this sense our evidence graph is in analogy to the web graph and the random walk of attacker in essence fits into the Pagerank model. If we leverage Pagerank methods on the aggregated evidence graph, higher Pagerank score would intuitively denote higher probability of being visited in a multi-stage attack scenario. In this chapter we will present how to apply customized Pagerank model for two specific cases of attack analysis: targeted investigation and hidden member identification.

5.2 Ranking for Targeted Investigation

In many cases it is desirable to incorporate out-of-band domain knowledge for more "targeted" investigation. For instance, the analyst might only be interested in attack scenarios related to a given watch list of hosts. Hosts on the watch list could be ones with high asset value or ones deemed as suspicious from upfront knowledge. While the spectral clustering approach can be applied to generically extract attack clusters in the evidence graph, it is more intuitive to directly identify the entities that have significant "relative importance" with respect to hosts on the watch list. In our evidence graph model, "relative importance" denotes that importance of a candidate node is computed with respect to a set of "root nodes" instead of the whole graph.

Definition 2. *Given graph $G = (V, E)$, a set of root nodes R and a set of candidate nodes C where $R, C \subset V$, the relative importance problem is to rank the importance of nodes in C with respect to R.*

Inspired by the work of White et al. [67], we retrofit an extension to the basic Pagerank model named "personalized Pagerank" to compute relative importance for our targeted investigation purpose. As shown in Equation 5.2, the personalized Pagerank model replaces $\frac{ee^T}{n}$ with ev^T where v^T is a probability vector called the personalization vector [30]. In the basic Pagerank model illustrated in Equation 5.1, the probability of a node being selected as the new start is uniformly distributed over the whole graph. For the personalized Pagerank model, the

probability is explicitly specified by the personalization vector instead. The intuition behind personalized Pagerank is that by constructing the personalization vector with respect to the given watch list, we are able to rank potential attackers in terms of their relative importance.

$$\overline{\overline{P}} = \alpha P + (1-\alpha)ev^T \tag{5.2}$$

To detect potential attackers with significant relative importance to the given watch list L_W, we perform personalized Pagerank evaluation on the aggregated evidence graph G_A. Referring to Equation 5.2, P is the normalized adjacency matrix of G_A. The root set is defined as $R = V_{G_A} \bigcap L_W$ and the set of candidate nodes is defined by $C = V_{G_A} \setminus R$. We construct the personalization vector as $v = \{p_1, \cdots, p_n\}$ where p_i denotes the prior importance assigned to node i and $p_1 + p_2 + \cdots + p_n = 1$. In this work we simply set p_i based on root set membership and all nodes in the root set R are treated equal. In other words, if node $i \in R$, then $p_i = \frac{1}{|R|}$, otherwise $p_i = 0$. The personalized Pagerank scores are then given by the dominant eigenvector of $\overline{\overline{P}}$. The result ranking indicates the relative importance with respect to the root set, i.e. level of suspicion for potential attackers associated with the given watch list. Hosts identified with high relative importance scores then are selected for further cluster evaluation.

Here we illustrate the application of personalized Pagerank model for targeted investigation with the ARDA "Bankshot" scenario analyzed in Section 4.2.3.1. Suppose the watch list is defined as the set of three exploited internal servers shown in Figure 4.5(c): L_W={*Failed Target, Stepping Stone, Victim*}. Personalized Pagerank is then applied on the graph component C_{10} of aggregated evidence graph G_A with root set $R = L_W$ to compute the relative importance rank. The top 5 ranked hosts out of 112 are shown in Table 5.1.

Rank	1	2	3	4	5
Host	Attacker 2	100.10.20.20	Attacker 1	100.20.1.3	131.58.63.160
Score	0.52	0.10	0.07	0.06	0.06

Table 5.1 Personalized Pagerank scores for ARDA scenario

From Table 5.1 we can see that *Attacker 2* stands out with a score significantly higher than that of other suspects, as in ground truth it successfully compromises *Stepping Stone*

and *Victim*. *Attacker 1* is not as obvious because it only initiates a failed attack attempt towards the *Failed Target*. Therefore it is difficult to be differentiated with the false positive hosts 100.10.20.20, 100.20.1.3 and 131.58.63.160. In summary, the relative importance ranking effectively reveals the attackers and their respective risks to hosts on the given watch list.

5.3 Ranking for Hidden Member Identification

Clusters extracted from the aggregated primary evidence graph may only provide a limited view of the coordinated attack scenario. Intrusion investigation cannot be completed without considering "hidden members" in an attack scene that are characterized by non-explicit-attack malicious activities. As another high level example, the attacker uses a different host other than the ones involved in explicit attacks as the control master. After successfully compromising several targets and installing agents on them, the master sends out commands to the agents to launch further attacks. We are able to identify the cluster of attacker and victims because explicit attacks are captured in IDS alerts and represented in the primary evidence graph. However the master will evade detection because the control traffic appears benign to primary evidence sensors(e.g. simply some telnet connections from the master to the agents). These "conduits" of malicious behavior need to be effectively identified through secondary evidence analysis.

In Section 3.6, we propose an interactive hypothesis testing approach to integrate secondary evidence into our analysis process. For the above example, an intuitive hypothesis to discover the hidden master could be formed as "In terms of telnet connections, which hosts in the network are the most important with respect to the set of victims?".

Our approach evaluates the hypothesis in two steps. As presented in Section 3.6, the first step is to perform knowledge based secondary evidence filtering based on the triple filter(*Association*, *Classification*, *Time*). Based on the notion that benign activities associated with a misfeasor could be suspicious, we note that the set of clusters extracted from primary evidence provides a flexible guide for defining *Association*. The *Association* set could be a single cluster C_i, a subset of C_i or union of multiple clusters. Using analysis results from primary

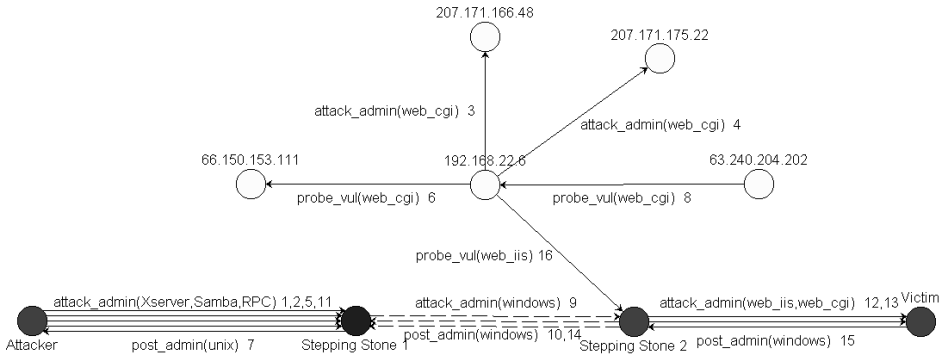
evidence greatly reduce the search space of secondary evidence. The secondary evidence graph then can be constructed with the filtered events.

In the second step, the Personalized Pagerank model can be used to rank potential hidden attackers in terms of their relative importance to the defined *Association* set as "root nodes" in the secondary evidence graph. The relative importance rank essentially provides a quantitative measure for the notion of *Non-coincidence*, i.e. activities of certain patterns increase the chance of being intentional rather than incidental. In the previous example, the master should be labelled with a higher scale of suspicion because it has connections with multiple stepping stones.

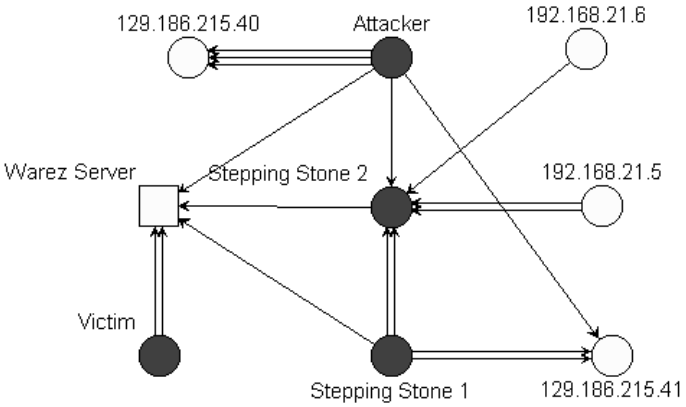
Let A be the *Association* set defined for secondary evidence filtering. The secondary evidence graph for relative importance evaluation is denoted as $G_r = (V_r, E_r)$ where E_r is the set of edges corresponding to the filtered non-explicit-malicious events and V_r represents the hosts associated with these events. The set of root nodes can be defined by $R = A \bigcap V_r$ and the set of candidate nodes is thus defined by $C = V_r \setminus R$. Potential hidden members then can be evaluated with the relative importance score computed by Equation 5.2.

Here we illustrate the application of personalized Pagerank model for hidden member identification with the multi stage attack scenario illustrated in Figure 5.1(a). For convenience we adopt the following notations: *Attacker*, *Stepping Stone 1*, *Stepping Stone 2*, *Victim*: and *Warez Server*. The scenario is constructed as follows: *Attacker* initiates a Samba remote buffer overflow exploit against *Stepping Stone 1*; After break in attack tools are uploaded from *Warez Server* to *Stepping Stone 1*; Then *Stepping Stone 1* initiates Windows DCOM remote buffer overflow exploit against *Stepping Stone 2*; After break in attack tools are also uploaded from *Warez Server* to *Stepping Stone 2*; *Stepping Stone 2* then initiates Frontpage Server 2000 buffer overflow exploits against *Victim*. Finally sensitive data is transferred from *Victim* to the *Warez Server*.

The original primary evidence graph G_P is shown in Figure 5.1(a). The edge labels indicate attack phases and relevant resources of corresponding alerts. Also the numbers represent the sequence of alerts in time order. The highlighted nodes are hosts involved in the true coordi-



(a) Primary evidence graph with attack clusters



(b) Secondary evidence graph for relative importance evaluation

Figure 5.1 Example scenario for hidden member identification

nated attack scenario while the rest are background artifacts. To introduce more intricacy we deliberately drop the alerts captured between *Stepping Stone 1* and *Stepping Stone 2*. As such it is intuitive to see that two clusters c_1 ={*Attacker, Stepping Stone 1*} and c_2 ={*Stepping Stone 2, Victim*} can be identified in the graph. However the *Warez Server* is not identified because ftp flows do not trigger Snort alerts. To construct the secondary evidence graph G_r for hidden member identification, it is straightforward to use the clusters from primary evidence graph to define *Association*, i.e. $A = c_1 \cup c_2$. Suppose traces of exploit codes is found on the stepping stones, it is reasonable to hypothesize the occurrence of file transfers in the attack and

therefore *Classification* = {all file transfer flows}. Figure 5.1(b)shows the secondary evidence graph constructed with the filtered events. The highlighted circular nodes denote the root set R and the other nodes belong to the candidate set C. The relative importance computation by Equation 5.2 is then applied on G_r. As shown in Table 5.2, personalized Pagerank score of the *Warez Server* is significantly higher than that of other hosts, which reveals its anomalous connection pattern with extracted attack clusters and suggests that it is a potential hidden member in the attack scenario.

Rank	1	2	3	4	5
Host	Warez Server	129.186.215.41	129.186.215.40	192.168.21.5	192.168.21.6
Score	0.244	0.084	0.036	0.032	0.032

Table 5.2 Personalized Pagerank scores for hidden member identification

In summary, the personalized Pagerank model provides a systematic and quantitative metric that well fits into our interactive hypothesis testing procedure. Based on different hypothesis in secondary event filtering and root set definition, the relative importance ranks could lead to different views of potential hidden members for further investigation. As an immediate extension, the investigator can also assign weighted importance to nodes in the root set according to their roles and asset value. Finally, current techniques in efficient computation of Pagerank enable us to handle large scale intrusion analysis with ease.

5.4 Discussions

In addition to network events, evidence in an attack scenario could also include host events such as account creation, backdoor installation, system binary replacement and data tampering. These host based evidence could also be integrated into our hypothesis testing and hidden member identification scheme. For a host that is genuinely involved in the attack, here we give a brief summary of the analysis scenarios based on the availability of different types of evidence:

1. The host is revealed in network based primary evidence (with or without host based evidence).

In this case the host is readily incorporated in the primary evidence graph, therefore we are able to apply the spectral clustering procedure to extract it out. Hidden member identification with personalized Pagerank methods would not be necessary.

2. The host is revealed in host based primary evidence but not in any network based primary evidence.

   In this case it is highly likely that the remote exploits are missed by network based sensors. In the primary evidence graph, corresponding node to the host becomes a isolated "hot spot": its local state is activated as suspicious from host based evidence, but no event that causes the state change is available from network based evidence. Therefore we can apply interactive hypothesis testing to evaluate potential hidden attackers associated with these "hot spots". Firstly, the set of suspicious isolated nodes would define *Association*, i.e. the root set $|R|$ in personalized Pagerank. Secondly, clues found in host based events would be used to formulate *Classification* and *Time*) for filtering flows. For instance, if we find a listening port associated with a suspicious process on the host, we can only consider flows associated with this particular port.

3. The host is not revealed in any type of primary evidence.

   In this case we don't have much choice other than hypothesize-and-test for hidden member identification. One concern is that the Pagerank result could bias towards network entities with intense normal activities. For instance, an innocent public web server could have high relative importance score because of its intense connectivity in secondary evidence graph of certain hypothesis. Multiple trials and additional out-of-band information are needed for further evaluation.

CHAPTER 6. CLUSTER EVALUATION WITH ATTACK SCENARIO RECONSTRUCTION

6.1 Motivation

In this chapter we present schemes for automated cluster evaluation and semantic scenario reconstruction. Spectral clustering and Pagerank methods aim at *Attack Group Identification*, i.e. extract clusters of hosts from the evidence graph as candidate members of coordinated attacks. Naturally the next important step of forensic analysis is to help investigators further examine the candidate clusters and reconstruct genuine intrusion scenarios. Our cluster evaluation module is motivated by the following observations:

1. Candidate clusters extracted by graph structure analysis methods contain undesired artifacts. As presented in Section 4.3.1, *Adjacent noise* and *Internal noise* could result in irrelevant entities and events to be extracted in the same cluster with those of the true attack scenario. Further semantic attack analysis is needed to separate the mixture and reconstruct the true scenario.

2. Significant clusters in the evidence graph could represent atypical but benign activities, such as a network misconfiguration. We note that clusters caused by benign activities tend to have persistent patterns. For example, in the MIT Lincoln Lab DARPA 2000 dataset LLDOS1.0 we observed a significant cluster that consists of a set of routers in the DMZ, with a huge number of ICMP related alerts among them. Clusters like this will be persistently generated until the misconfiguration is corrected. In contrast, clusters caused by genuine coordinated attacks appear in a more spontaneous manner and therefore worth more attention. Moreover, considering the fact that IDS alerts are

predominantly false positives, it is quite likely that false alarms could randomly form densely connected clusters in the evidence graph. In both cases the whole extracted cluster should be identified as irrelevant, i.e.*Isolated noise*.

Given candidate clusters extracted from the evidence graph, the challenges of cluster evaluation essentially can be summarized as:

How to differentiate clusters formed by genuine coordinated attacks from those formed by random false alarms or persistent benign activities? Furthermore, how do we further filter out background noise in true attack clusters and reconstruct the coordinated attack scenario?

Considering the large number of exploitable vulnerabilities and many variations of attack strategies, manual inspection of all candidate clusters would be a heavy burden for investigators. Moreover, establishing comprehensive expert rule sets for the task would also be non-trivial and error prone. The major contribution presented in this chapter is a systematic and automated approach for cluster evaluation and scenario reconstruction. Our approach is based on the notion of target-oriented effective event sequence(TOEES). Through scenario analysis and data trace driven experiments, we demonstrate that clusters resulted from coordinated attack activities possess characteristics inherently different from that of benign or random noise clusters. Moreover, we present how to reconstruct multistage attack scenarios with target-oriented effective event sequence.

6.2 Target-Oriented Effective Event Sequence

To transform multiple parallel edges in the raw primary evidence graph G_P into a single unified numeric representation in the aggregated evidence graph G_A for structure based analysis, we have developed an aggregation scheme based on two generalized attributes of primary evidence: attack phase and relevant resource. Firstly, intrusion events naturally fall into different attack phases, which indicate the attacker's respective objectives throughout the intrusion process; Secondly, relevant resources represents the set of network and host properties exploited, which indicate the attacker's means to achieve his objective in a certain attack phase. Essentially, multi-stage attacks can be regarded as an increasing process of the at-

tacker's knowledge and capability towards its target, which could be uniquely characterized by the advancement of attack phase and/or expansion of relevant resources. Each effective step to increase the attacker's capability and knowledge would be revealed by either proceeding to a more advanced attack phase or exploiting alternate resources to achieve objective of the current attack phase. To represent the combined effect of multiple observed events between a pair of nodes in the raw evidence graph, only the intrusion events that demonstrate either advancement of attack phase and/or expansion of relevant resources are counted towards the aggregated edge weight. In the aforementioned graph aggregation phase, the evaluation for attack phase and relevant resource is restricted to the sequence of events between a source/target pair.

In the cluster evaluation module, we extend the attack phase and relevant resource evaluation to the whole set of hosts and events in the given cluster through an integrated metric. Specifically, the integrated metric is designated to effectively detect attackers' strategy of subversion by *Role Perturbation*. Role perturbation refers to the scenario that the attacker employs multiple identities to perform different functional tasks in a multi-stage attack. It is commonly used to masquerade the intention and procedure of multi-stage attacks, in the mean time make it more difficult for analysts to identify them from background noise. Here we illustrate the variations of role perturbation with the following case analysis.

First, we examine the unperturbed scenario as the starting point. Figure 6.1(a) shows an sequence of attack events between a pair of hosts A and B. Intuitively, the attacker use host A to collects information of target B(event 1), exploits the target to gain access privilege(event 2) and eventually obtains data from the target(event 3). Without any role perturbation, we are able to label the sequence of events between hosts A and B as highly likely to be an intentional multi-stage attack, in which host A is identified as the scanner, attacker and affiliated data warehouse.

However, the attacker could easily distribute the functional roles to different identities to disguise the attack process. As shown in Figure 6.1(b), the attacker use host A to collect target information (event 1), then uses host B to launch privilege access exploit (event 2) and finally

uses host D to obtain data from the target(event 3). Consequently the functional roles are distributed to 3 different identities: A as scanner, B as attacker and D as the affiliated data warehouse. We regard this type of perturbation as *Role Distribution*.

In *Role Distribution* scenarios, variations in attack phases and relevant resources are initiated from different sources to the same target. When viewed from any individual (source,target) pair, the isolated event does not qualify for the criteria of attack phase advance or relevant resource expansion. Therefore it is difficult to judge whether it is part of a coordinated attack or background noise. Consider the scenario illustrated in figure 6.1(c), suppose two identical alerts are reported between (source,target) pairs (B,C) and (C,E). The former is a false alarm while the latter is part of a coordinated attack(event 2). When examined individually, it is difficult to tell which alert is more suspicious. However from a target oriented view, the alarm between (B,C) should be recognized with higher priority as host C is more likely to be the target of a coordinated multi-stage attack.

The second type of role perturbation is *Role Alternation*. In *Role Alternation* scenarios, one identity could play multiple roles in attacks against different targets. As shown in Figure 6.1(d), the cluster contains multiple attacks against different targets. Hosts A, C and E are involved in the attacks against target B while hosts C and F participate in the attacks against target D. Note that host C is involved in both attacks but takes different functional roles, i.e. *Attacker* towards target D and *Affiliated* in the attack towards target B.

Another related scenario to consider is *Simultaneous Attacks*. In this case, targets and time spans of multiple independent attack scenarios overlap with each other. Consequently the significant cluster extracted from evidence graph would consist a mixture of identities and interleaved sequence of events from different attack scenarios. Our objective then is to differentiate the identities and events of simultaneous independent attacks. By "independent" we are assuming that there are no collaborations between the attackers, i.e. they are not aware the existence of each other and do not share achievements(for example, the backdoor installed by one attacker cannot be used by other attackers). Otherwise the collaborated attacks could be classified as *Role perturbation* cases from a single attacker.

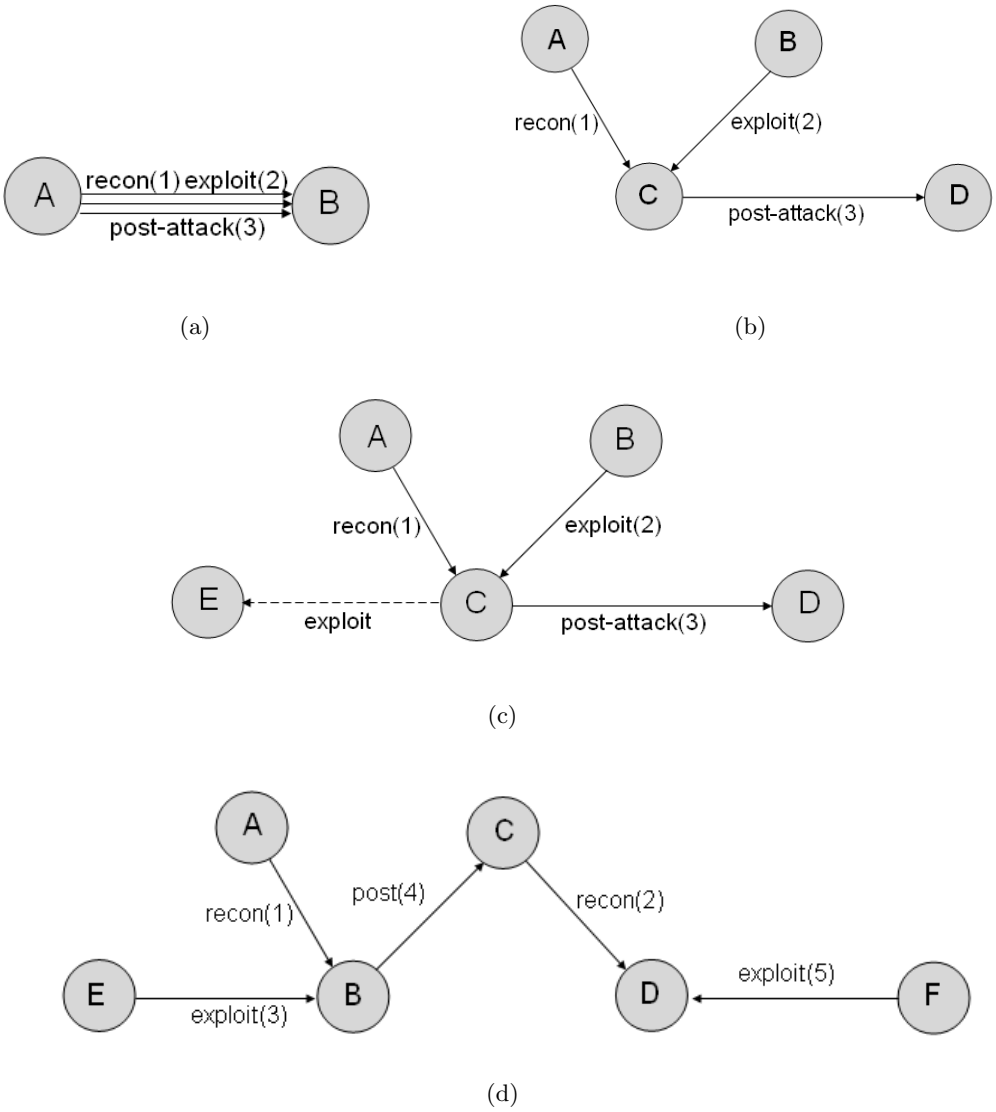


Figure 6.1 Examples of role perturbation

In recognition of the aforementioned *Role Perturbation* scenarios, we develop the notion of target-oriented effective event sequence(TOEES) for automated cluster evaluation and scenario reconstruction. A target-oriented effective event sequence is defined by the following components:

- **Attackers** H: Each TOEES possesses a set of entities that are attackers involved in the event sequence.

- **Target** T: Each TOEES has one unique entity that represents target of the event sequence.

- **Sequence of effective events** E: For a set of TOEES $S_1, \ldots, S_i, \ldots, Sn$ and a candidate event e, e is considered as an effective event for S_i if and only if (1) target of e matches T of S_i; (2) comparing to existing events in S_i, e either proceeds to a more advanced attack phase or attempts to achieve objective of the current attack phase by expanding the set of relevant resources exploited. (3) source of e does not exist in H of any other S_j that has the same target T.

Note that our definition of target-oriented effective sequence is essentially a depiction of the attack from the "victim's view". More specifically, it focuses on the following manifestations:

- Where and when did the intrusion originate?

- How did the intrusion happen, i.e. what are the relevant resources exploited?

- What are the likely consequences, i.e. what capabilities have been possibly achieved by the intrusion?

The process for identifying target-oriented effective event sequences in a given candidate cluster is shown in Algorithm 7. Here we illustrate steps of the algorithm in detail with the example scenario shown in Figure 6.2. Each event is tagged with abstracted attack phase(R: recon, E: exploit, P: post-attack) and sequence number in increasing time order. Relevant resources of the exploits are left out to simplify the illustration. Table 6.2 shows how the target-oriented effective event sequences are formulated in correspondence to each developing event.

Events R1 and E2 initialized TOEES I. Event R3 does not satisfy the attack phase and relevant resource advancement requirement for TOEES I, therefore a new TOEES II is initiated. Event E4 belongs only to TOEES I because its source B already exists in the attackers set H of TOEES I. On the other hand, events E5 and P7 potentially could be added to TOEES I and II because their sources does not exist in the attacker set H of both TOEES. Also note

input : candidate cluster $C = (V_C, E_C)$
output: set of target-oriented effective event sequences S

begin
  $count \leftarrow 0$;
  $S \leftarrow \emptyset$;
  $E_t \leftarrow$ sort E_C in increasing time order ;
  foreach *event* $e \in E_t$ **do**
    Look up attack phase P_e, relevant resource R_e ;
    if S *is empty* **then**
      $S_{count} \leftarrow e$;
      $H_{count} \leftarrow e_{src}$, $T_{count} \leftarrow e_{tgt}$;
      $PR_{count} \leftarrow (P_e, R_e)$;
      $H \leftarrow H_{count}$;
      $count++$;
    end
    else
      $foundMatch \leftarrow FALSE$;
      foreach S_i *where* $i \leq count$ **do**
        if $e_{src} \nsubseteq H_i \&\& e_{src} \subseteq H$ **then**
          continue;
        end
        if $T_i \equiv e_{tgt} \&\& (P_e, R_e) \nsubseteq PR_i$ **then**
          $S_i \leftarrow S_i \cup e$;
          $H_i \leftarrow H_i \cup e_{src}$;
          $PR_i \leftarrow PR_i \cup (P_e, R_e)$;
          $H \leftarrow H \cup H_i$;
          $foundMatch \leftarrow TRUE$;
        end
      end
      if $!foundMatch$ **then**
        $count++$;
        $S_{count} \leftarrow e$;
        $H_{count} \leftarrow e_{src}$, $T_{count} \leftarrow e_{tgt}$;
        $PR_{count} \leftarrow (P_e, R_e)$;
      end
    end
  end
end

Algorithm 7: Identify target-oriented effective event sequences in a candidate cluster

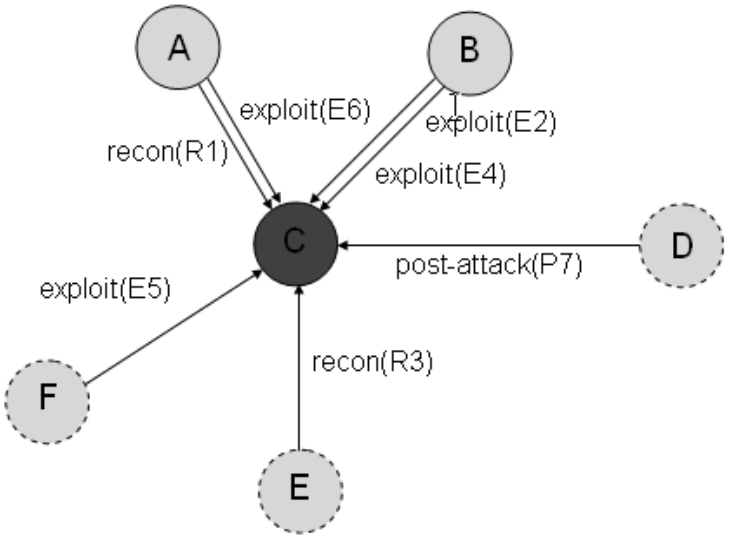


Figure 6.2 Example to identify target-oriented effective event sequence

Event	TOEES I	TOEES II
R1	E:{R1} H:{A} T:{C}	N/A
E2	E:{R1,E2} H:{A,B} T:{C}	N/A
R3	skip	E:{R3} H:{E} T:{C}
E4	E:{R1,E2,E4} H:{A,B} T:{C}	skip
E5	E:{R1,E2,E4,E5} H:{A,B,F} T:{C}	E:{R3,E5} H:{E,F} T:{C}
E6	E:{R1,E2,E4,E5,E6} H:{A,B,F} T:{C}	skip
P7	E:{R1,E2,E4,E5,E6,P7} H:{A,B,F,D} T:{C}	E:{R3,E5,P7} H:{E,F,D} T:{C}

Table 6.1 Steps of constructing target-oriented effective event sequence

that event E6 could be skipped by both sequences: it cannot be added to TOEES II because its source A exists in the host set H of TOEES I. Whether it could be added to TOEES I depends on comparison of attack phase/relevant resource comparisons with the existing set of events {R1,E2,E4,E5}.

After event P7, two target-oriented effective attack sequences are identified in the candidate cluster. Each TOEES readily reveals a coordinate attack scenario in spite of the attacker's role perturbation strategies. In addition, the two TOEESs present the two attack threads as potential *Simultaneous Attacks* in the same cluster, though ambiguity for host F and D needs to be further clarified with additional evidence.

The notion of attack phase and relevant resource evaluation, when extended to the extracted candidate graph clusters, provides an useful tool for methodically analyzing multi-stage attack scenarios. In addition to cluster suspicion evaluation, TOEES is also applied as the means for *Attack Scenario Reconstruction*. Conceptually wise, each TOEES is a natural depiction of multi-stage attacks towards a specific target. Construction of TOEES allows us to effectively identify identities in a coordinated attack scenario in spite of *Role Perturbation* strategies. Because random false alarms generally do not have the coordinated nature to form TOEES, we are able to further filter out false positives in the candidate cluster. Note that in scenarios that involve multiple exploited targets, further correlation needs to be established between multiple TOEES. For example in the stepping-stone attack scenario shown in figure 6.1(d), two separate TOEESs are identified with target hosts B and D respectively. However host C is shared in the set of *Attackers* in both TOEESs and can be used to correlate the event sequences into a complete scenario.

6.3 Experiments and Discussion

As a proof of concept, our cluster evaluation prototype is tested on the ARDA "Bankshot" attack dataset, which contains one genuine multi-stage attack scenario and random background attacks. The candidate attack cluster extracted with our spectral clustering approach (as shown in Figure 4.5(c)) is applied as the input for cluster evaluation.

Through Algorithm 7, two multi- event TOEESs are identified as representations of suspicious multi-stage attacks. As shown in Table 6.3, TOEES 1 reconstructs the intrusion attempts targeted at *Failed Target* and TOEES 2 captures the attacks towards *Victim*. We can see that though the attacker changes identity at different stages, the multi-stage attack scenario is clearly reconstructed with the two TOEESs. Moreover, the process of constructing TOEES further filters out the false positive identities 100.10.20.20, 100.1.10.101 and 131.58.63.160 in the candidate cluster. It can be seen that most random false alarms do not satisfy the attack phase and relevant resource correlation requirement for TOEES construction. The only false positive 100.20.1.3 included in TOEES 2 would need additional host-based attack verification

to be filtered out.

	TOEES 1	TOEES 2
H	{Attacker 1, Attacker 2}	{Attacker 2, Stepping Stone, 100.20.1.3}
T	{Failed Target}	{Victim}
E	{probe_vul,attack_admin(iis),attack_admin(windows_ssl)}	{probe_vul,attack_admin(iis),attack_admin(apache)}

Table 6.2 TOEESs identified from candidate attack cluster

For comparison with the true attack cluster, we generate synthetic non-malicious clusters as the baseline. As presented in Algorithm 8, the input target evidence graph G_T is built from attack-free background traffic collected on the target network. The input parameter X specifies size of the the random cluster generated. Note that we implicitly enforce the restriction that the extracted cluster will always be a connected graph.

```
   input  : Target evidence graph G_T = (V_T, E_T), cluster size X
   output: Random cluster C = (V_C, E_C)
 1 begin
 2     V_C ← ∅, E_C ← ∅ ;
 3     Pick a random seed node n_s from V_T ;
 4     V_C ← n_s ;
 5     repeat
 6         V_N ← unique external neighbors of all nodes in V_C ;
 7         if |V_N| ¿ 0 then
 8             Pick a random node n_t from V_N ;
 9             E_C ← E_C∪ all edges between V_C and n_t ;
10             V_C ← V_C ∪ n_t ;
11         end
12         else
13             break ;
14         end
15         X ← X − 1 ;
16     until X > 1;
17     Output C = (V_C, E_C) ;
18 end
```

Algorithm 8: Algorithm for extracted random cluster from given evidence graph

In our experiment 100 random clusters are generated from attack-free background traffic trace collected on the same network where the genuine multistage attacks happen. Then Algorithm 7 is applied to identify TOEESs in each random cluster. The result shows that

no multi-event TOEES is identified, i.e. there is no suspicion of multi-stage attacks in the random clusters. The observation illustrates that our TOEES based evaluation is effective in distinguishing between true attack cluster and random attack-free clusters.

We also apply TOEES evaluation to the cluster of DMZ hosts identified in MIT Lincoln Lab DARPA 2000 LLDOS 1.0 dataset. The cluster is extracted in spectral clustering results because of a large number of ICMP related alerts caused by network misconfiguration, which is irrelevant to the true multi-stage attack scenario. Based on Algorithm 7, only 4 single-event TOEES is identified, which shows that the cluster is a non-malicious anomaly. Furthermore, the same TOEES pattern is repeatedly observed at different time slots, which suggests that the atypical cluster is caused by certain benign persistent activities.

Finally, our target-oriented evaluation is also suitable for automated monitoring on a given watch list of assets to be protected. Periodical procedures can be set up to identify TOEESs targeted at the specified assets and alert for further examination when TOEES of multiple events are detected.

CHAPTER 7. CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize the thesis contributions and briefly discuss some areas that of interest for future research.

7.1 Conclusion

With the increasing threat of cyber attacks, prevention and detection techniques are not sufficient for information security needs. Post-incident network forensics analysis has become an important practice to extract meaningful evidence from security sensors and understand the attack scenario. However, limitations of current evidence sources and attack correlation techniques impose great challenges for investigators to perform effective, scalable and systematic analysis.

This dissertation improves the state of art for network forensic analysis with a novel graph based approach that is applicable for practical, large scale analysis and less dependent on ad hoc expertise. Our prototype system utilizes a variety set of techniques to provide an effective, flexible and scalable solution for analyzing complicated multi-stage attacks. We also integrate non-explicit secondary evidence from heterogeneous information sources into the correlation process to provide a comprehensive view of the attack.

In summary, we have addressed challenges in network forensic analysis with the following approaches:

Evidence Graph Model and Hierarchical Reasoning Framework. We first present the evidence graph model which provides an intuitive representation of collected forensic evidence as well as the foundation for forensic analysis. After evidence preprocessing, hyper alerts are used to construct the raw primary evidence graph. Based on the evidence graph,

we develop a hierarchical reasoning framework that consists of two levels: local reasoning and global reasoning. Local reasoning employs fuzzy inference to infer the functional states of an host level entity from its local observations. On the other hand, global reasoning performs graph structure analysis to identify the set of highly correlated hosts that belong to the coordinated attack scenario. For global reasoning, we develop a graph aggregation scheme to evaluate the combined effects of multiple exploits based on attack phase classifications and relevant resources. The raw primary evidence graph is abstracted into the aggregated primary evidence graph, an undirected simple graph efficient for structure based analysis.

Primary Attack Group Identification. Building upon the aggregated primary evidence graph, we provide graph structure based analysis for attack group identification in two investigation modes. Firstly, for generic find-the needle-in-the-haystack investigation, we develop a recursive spectral graph clustering algorithm to identify multistage attacks by extracting significant clusters from the graph Laplacian spectrum. Our approach effectively differentiates coordinated multistage attacks from irrelevant background noise. Furthermore, we analyze different categories of background noise in the evidence graph model and evaluate the robustness of clustering results through graph perturbation analysis. Secondly, for targeted investigation with out-of-band information, we apply the personalized Pagerank model to quantitatively evaluate the relative importance of potential attackers with regard to the given watch list.

Hidden Attacker Identification and Hypothesis Testing. As an unique contribution, we integrate secondary evidence into intrusion investigation to identify "hidden attackers" that are characterized by non-explicit-attack malicious activities. We propose a flexible hypothesis testing procedure for missing secondary evidence. In the interactive hypothesis testing process, analysis results of primary evidence are used to guide the exploration of secondary evidence. Filtered secondary evidence are then instantiated to build the enriched secondary evidence graph. In addition, we customized the personalized Pagerank algorithm to quantitatively rank potential hidden attackers in terms of their relative importance to entities identified in primary attack analysis.

Cluster Evaluation with Scenario Reconstruction. Finally, we develop automated

suspicion evaluation with semantic scenario reconstruction for the candidate clusters extracted in attacker group identification. Specifically, we present the notion of Target-Oriented Effective Event Sequence(TOEES), which allows us to further filter out false positives and effectively reconstruct multi-stage attack scenarios. Our approach is capable of identifying stealthy *Role Perturbation* attack strategies and scenarios of *Simultaneous Attacks*.

We believe the system presented represents an important contribution to network forensic analysis research. In comparison to previous works, our methods perform well for post-incident multi-stage attack analysis with less demand on domain-expert -knowledge based attack modeling. Information required to build and analyze the evidence graph can be obtained from current security sensors, without the need for detailed a priori host vulnerability knowledge and hard coded causality rules. Spectral clustering and Pagerank methods are performed on the host-centric graph where each graph node represents a host level network entity. With existing techniques for high performance spectral graph clustering and Pagerank computation, our methods can efficiently handle intrusion investigation in large networks.

7.2 Future Work

There are still many open problems in our network forensic analysis approach that remain to be addressed. In the following, we introduce several possible directions for further study.

Evaluation with Real World Datasets One of the biggest open challenges is the evaluation with more realistic attack datasets. There have been many controversies on evaluation results with synthetically generated attack datasets, mainly because background traffic and network model in current artificial datasets are often overly simplistic when compared to those in real world networks. However, there are two problems associated with using real-world attack datasets. Firstly, privacy issue is a big concern for organizations to make real world attack datasets publicly available. A flexible data sanitization process needs to be developed to provide privacy protection as well as the preservation of critical information for analysis usability. Secondly, real world attack datasets do not readily have ground truth. Without a common baseline it is difficult to make a fair comparison between performance of different

analysis methods. Therefore a challenging task is to generate ground truth for real world datasets in an automated approach. It is especially difficult because of the dilemma that the ground truth has to be universally accepted but verification from the real attacker can hardly be obtained.

Distributed Analysis Forensic evidence may be scattered across a large number of networks and administrative domains. It is often not efficient and realistic to perform analysis at a centralized evidence repository. It is important to develop a distributed algorithm to perform independent initial analysis and exchange local results to reach a unified global view of evidence in an incremental and iterative manner. We will conduct analytical study to evaluate the reasoning accuracy and information exchange overhead of the distributed reasoning procedure. In addition, performance of the reasoning algorithm in partial and incremental deployment of agents needs to be studied.

Attack Case Profiling Attack case profiling is based on the promise that attackers tend to repeat certain behavioral patterns or strategies. Given a new extracted scenario, the natural question to ask is whether it is similar to patterns seen in the past or elsewhere. Knowing the recurrence of attack scenarios would help the investigator form appropriate responses. As our evidence graph model presents a graph based description of attack patterns, this is similar to the decision problem of subgraph isomorphism. One potential is to develop efficient graph spectra characterizations for encoding and matching of attack patterns. Here a major challenge is to minimize the cospectrality effect of graphs and provide an effective spectral signature. Adequate attack strategy based abstraction is needed prior to encoding and matching of graph structure. In a noisy dataset, a more forgiving mechanism such as graph edit distance could also be considered. We will do cross validation tests to evaluate how well the profiling methods can relate known similar scenarios and differentiate unrelated scenarios.

BIBLIOGRAPHY

[1] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 217–224, New York, NY, USA, 2002. ACM.

[2] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammerling, J. Demmel, C. Bischof, and D. Sorensen. Lapack: a portable linear algebra library for high-performance computers. In *Supercomputing '90: Proceedings of the 1990 ACM/IEEE conference on Supercomputing*, pages 2–11, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[3] Bro intrusion detection system. http://www.bro-ids.org.

[4] Brian D. Carrier and Eugene H. Spafford. Defining event reconstruction of digital crime scenes. *Journal of Forensic Sciences*, November 2004.

[5] J. P. Carvalho and J. A. B. Tome. Rule Based Fuzzy Cognitive Maps and Fuzzy Cognitive Maps - A Comparative Study. In *Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society(NAFIPS99)*, New York, 1999.

[6] J. P. Carvalho and J. A. B. Tome. Rule Based Fuzzy Cognitive Maps: Fuzzy Causal Relations. In *Proceedings of the 8th International Fuzzy Systems Association World Congress(IFSA99)*, Taiwan, 1999.

[7] Fan R. K. Chung. *Spectral Graph Theory.* American Mathematical Society, 1997.

[8] Ying Cui, Xiaoli Z. Fern, and Jennifer G. Dy. Non-redundant multi-view clustering via orthogonalization. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 133–142, Washington, DC, USA, 2007. IEEE Computer Society.

[9] F. Cuppens. Managing alerts in a multi-intrusion detecttion environment. In *Proceedings of the 17th Annual Computer Security Applications Conference(ACSAC)*, 2001.

[10] F. Cuppens and A. Miege. Alert Correlation in a Cooperative Intrusion Detection Framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.

[11] Frédéric Cuppens and Rodolphe Ortalo. Lambda: A language to model a database for detection of attacks. In *RAID '00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, pages 197–216, London, UK, 2000. Springer-Verlag.

[12] O. Dain and R. Cunningham. Building scenarios from a heterogeneous alert stream. In *Proceedings of the 2001 IEEE workshop on Information Assurance and Security*, pages 231–235, 2001.

[13] O. Dain and R. Cunningham. Fusing a heterogeneous alert stream into scenarios. In *Proceedings of the 2001 ACM workshop on Data Mining for Security Applications*, pages 1–13, 2001.

[14] MIT Lincoln Lab 2000 DARPA intrusion detection scenario specific datasets. http://www.ll.mit.edu/IST/ideval/data/2000/index.html.

[15] H. Debar, M. Dacer, and A. Wespi. A revised taxonomy for intrusion-detection systems. In *IBM Research Report*, 1999.

[16] Herve Debar and Andreas Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection(RAID)*, October 2001.

[17] Chris Ding. A tutorial on spectral clustering. Talk presented at ICML 2004. Slides available at http://crd.lbl.gov/ cding/Spectral/.

[18] S. Eckmann, G. Vigna, and R. Kemmerer. Statl: An attack language for state-based intrusion detection. Dept. of Computer Science, University of California, Santa Barbara., 2000.

[19] EnCase Forensic Tool. Available at http://www.guidancesoftware.com.

[20] eTrust Network Forensics Solution. Available at http://www3.ca.com/.

[21] flow-tools. http://www.splintered.net/sw/flow-tools/.

[22] G.Palmer. A roadmap for digital forensics research. In *Proceedings of the first Digital Forensic Research Workshop*, Utica, New York, USA, 2001.

[23] Intrusion Detection Message Exchange Format. Internet draft available at http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-14.txt.

[24] Koral Ilgun. Ustat: A real-time intrusion detection system for unix. In *SP '93: Proceedings of the 1993 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 1993. IEEE Computer Society.

[25] Institute for Security Technology Studies. Law enforcement tools and technologies for investigating cyber attacks: Gap analysis report. http://www.ists.dartmouth.edu, February 2004.

[26] Takahiko Ito, Masashi Shimbo, Taku Kudo, and Yuji Matsumoto. Application of kernels to link analysis. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 586–592, New York, NY, USA, 2005.

[27] K. Julisch. Mining alarm clusters to improve alarm handling efficiency. In *Proceedings of the 17th Annual Computer Security Applications Conference(ACSAC)*, pages 12–21, 2001.

[28] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. In *ACM Transactions on Information and System Security*, pages 443–471, Nov 2003.

[29] C. Kruegel and W. Robertson. Alert Verification: Determing the success of intrusion attempts. In *Proceedings of the 1st Workshop on the Detection of Intrusions and Malware Vulnerability Assessment (DIMVA)*, Dortmund, Germany, July. 2004.

[30] Amy N. Langville and Carl D. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

[31] LEDA graph library. http://www.algorithmic-solutions.com/enleda.htm.

[32] V. Mehta, C. Bartzis, H. Zhu, E.M. Clarke, and J.M. Wing. Ranking Attack Graphs. In *Proceedings of Recent Advances in Intrusion Detection 2006(RAID'06)*, Hamburg, Germany, September 2006.

[33] M. Meila and J. Shi. A random walks view of spectral segmentation. In *Proceedings of 8th International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 744–750, 2001.

[34] Bojan Mohar. The laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, pages 871–898, 1991.

[35] B. Morin and H. Debar. Correlation of intrusion symptoms: an application of chronicles. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection(RAID'03)*, 2003.

[36] netcat. http://netcat.sourceforge.net/.

[37] NetDetector. Available at http://www.niksun.com/Products-NetDetector.htm.

[38] Cisco IOS NetFlow protocol. http://www.cisco.com/en/US/products/ps6601/home.html.

[39] P. Ning, Y. Cui, and D. S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *9th ACM Conference on Computer and Communicaitons Security*, November 2002.

[40] P. Ning and D. Xu. Learning attack stratagies from intrusion alerts. In *Proceedings of the 10th ACM Conference on Computer and Communicaitons Security*, 200-209, 2003.

[41] Peng Ning and Dingbang Xu. Hypothesizing and reasoning about attacks missed by intrusion detection systems. *ACM Transactions on Information and System. Security.*, 7(4):591–627, 2004.

[42] The Open Source Vulnerability Database. http://osvdb.org/.

[43] C. Phillips and L. Swiler. A graph-based system for network vulnerability analysis. In *Proceedings of the New Security Paradigm Workshop*, Charlottesville, VA, USA, 1998.

[44] X. Qin and W. Lee. Statistical causality analysis of INFOSEC alert data. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection(RAID)*, September 2003.

[45] X. Qin and W. Lee. Discovering novel attack strategies from INFOSEC alerts. In *Proceedings of the 9th European Symposium on Research in Computer Security*, September 2004.

[46] C. Ramakrishnan and R. Sekar. Model-based vulnerability analysis of computer systems. In *Proceedings of the 2nd International Workshop on Verification, Model Checking and Abstract Interpretation*, 1998.

[47] Ronald W. Ritchey and Paul Ammann. Using model checking to analyze network vulnerabilities. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, Washington, DC, USA, 2000.

[48] SafeBack Bit Stream Backup Software. Available at http://www.forensics-intl.com/safeback.html.

[49] Kulesh Shanmugasundaram, Nasir Memon, Anubhav Savant, and Herve Bronnimann. ForNet: A Distributed Forensics Network. In *Proceedings of the Second International*

Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security, 2003.

[50] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2002.

[51] Oleg Sheyner and Jeannette M. Wing. Tools for generating and analyzing attack graphs. In *Proceedings of International Symposium on Formal Methods for Components and Objects*, 2005.

[52] J. Shi and J. Malik. Normalized cuts and image segmentation. *Internet Mathematics*, 22(8):888–905, 2000.

[53] Ambareen Siraj, Susan M.Bridges, and Rayford B.Vaughn. Fuzzy cognitive maps for decision support in an intelligent intrusion detection system. Technical report, Department of Computer Science, Mississippi State University, 2001.

[54] Snort network intrusion detection system. http://www.snort.org.

[55] Softflowd. http://www.mindrot.com/softflowd.html.

[56] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – A graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, 1996.

[57] Ulf Lindqvist Steven Cheung and Martin Fong. Modeling multistep cyber attacks for scenario recognition. In *Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, pages 284–292, Washington, D.C., 2003.

[58] Eric Robertson Steven Noel and Sushil Jajodia. Correlating intrusion events and building attack scenarios through attack graph distances. In *Proceedings of the 20th Annual Computer Security Applications Conference(ACSAC'04)*, December 2004.

[59] tcpflow. http://www.circlemud.org/ jelson/software/tcpflow/.

[60] tcptrace. http://www.tcptrace.org/.

[61] A. Valdes and K. Skinner. Probablistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection(RAID)*, October 2001.

[62] D. Verma and M. Meila. A comparison of spectral clustering algorithms. *Technical report uw-cse-03-05-01, university of washington.*, 2005.

[63] Wei Wang and Thomas E. Daniels. Building evidence graphs for network forensics analysis. In *Proceedings of the 21st Annual Computer Security Applications Conference(ACSAC'05)*, December 2005.

[64] Wei Wang and Thomas E. Daniels. Network forensics analysis with evidence graphs. In *Proceedings of the 5th Digital Forensic Research Workshop (DFRWS)*, New Orleans, LA, August 2005.

[65] Wei Wang and Thomas E. Daniels. Diffusion and graph spectral methods for network forensic analysis. In *Proceedings of the 12th New Security Paradigms Workshop*, Schloss Dagstuhl, Germany, September 2006.

[66] Wei Wang and Thomas E. Daniels. A graph based approach toward network forensics analysis. *ACM Trans. Inf. Syst. Secur.*, 12:4:1–4:33, October 2008.

[67] Scott White and Padhraic Smyth. Algorithms for estimating relative importance in networks. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 266–275, 2003.

[68] Ping Zhu and Richard C. Wilson. A study of graph spectra for comparing graphs. In *The 16th British Machine Vision Conference*, Oxford Brookes University, Oxford, September 2005.