Reasoning about Complementary Intrusion Evidence*

Yan Zhai, Peng Ning, Purush Iyer, Douglas S. Reeves
Cyber Defense Laboratory
Department of Computer Science
North Carolina State University
Raleigh, NC 29695-8207
{yzhai, pning, purush, reeves}@ncsu.edu

Abstract

This paper presents techniques to integrate and reason about complementary intrusion evidence such as alerts generated by intrusion detection systems (IDSs) and reports by system monitoring or vulnerability scanning tools. To facilitate the modeling of intrusion evidence, this paper classifies intrusion evidence into either event-based evidence *or* state-based evidence*. Event-based evidence refers to observations (or detections) of intrusive* actions *(e.g., IDS alerts), while state-based evidence refers to observations of the* effects *of intrusions on system states. Based on the interdependency between event-based and state-based evidence, this paper develops techniques to automatically integrate complementary evidence into Bayesian networks, and reason about uncertain or unknown intrusion evidence based on verified evidence. The experimental results in this paper demonstrate the potential of the proposed techniques. In particular, additional observations by system monitoring or vulnerability scanning tools can potentially reduce the false alert rate and increase the confidence in alerts corresponding to successful attacks.*

1. Introduction

It is well-known that current intrusion detection systems (IDSs) produce large numbers of alerts, including both actual and false alerts. The high volume and the low quality of those alerts (i.e., missed attacks and false alerts) make it very hard for human users or intrusion response systems to understand the alerts and take appropriate actions.

Several alert correlation techniques have been proposed to facilitate the analysis of intrusion alerts, including those based on the similarity between alert attributes [7, 10, 26, 29], previously known (or partially known) attack scenarios [11, 12], and prerequisites and consequences of known attacks [8, 21]. However, most of these correlation methods focus on IDS alerts, overlooking other intrusion evidence provided by system monitoring tools (e.g., anti-virus software) and vulnerability scanning tools (e.g., Nessus [3], SATAN [14], Nmap [15]). Since none of the above methods can perfectly construct attack scenarios due to the imperfection of the IDSs, it is desirable to include additional, complementary intrusion evidence to further improve the performance of intrusion analysis.

Several researchers recently investigated ways to consider multiple information sources during intrusion analysis [20,23]. A formal model named M2D2 was proposed to represent data relevant to alert correlation, including characteristics of monitored systems, properties of security tools, and observed events [20]. Though quite useful for alert correlation, M2D2 does not provide a specific mechanism to automatically reason about information provided by multiple sources. Another mission-impact-based method [23] reasons about the relevance of alerts by fusing alerts with the targets' topology and vulnerabilities, and ranks alerts based on their relationships with critical resources and users' interests. Though the mission-impact based method can automate the analysis of intrusion alerts, the construction of a mission-impact based model requires substantial human intervention, and the constructed model is highly dependent on the monitored systems. Thus, it is desirable to seek other effective mechanisms that can handle complementary intrusion evidence automatically.

In this paper, we develop techniques to automatically integrate and reason about complementary intrusion evidence, including IDS alerts, reports from system monitoring or vulnerability scanning tools, and human observations. Our approach is based on the interdependency between attacks and

* This work is partially supported by the National Science Foundation (NSF) under grants ITR-0219315 and CCR-0207297, and by the U.S. Army Research Office (ARO) under grant DAAD19-02-1-0219. The authors would like to thank the anonymous reviewers for their valueable comments.

system states. That is, an attack may need certain system states to be successful, and will modify the system states as a result. However, IDS alerts, which represent detected attacks, are uncertain due to the imperfection of current IDSs. To reason about uncertain IDS alerts, our approach automatically builds Bayesian networks that consist of variables representing IDS alerts and system states. With additional, complementary evidence about system states provided by system monitoring tools, vulnerability scanning tools, and human observations, we can then make further inference about uncertain IDS alerts. As a result, we can increase our confidence in alerts corresponding to successful attacks, and at the same time reduce the confidence in false alerts. Moreover, by combining system state evidence, we can make reasonable hypotheses about attacks possibly missed by IDSs.

The main contribution of this paper is a reasoning framework for complementary intrusion evidence. To our best knowledge, this is the first attempt to *automatically* integrate and reason about complementary intrusion evidence such as IDS alerts and vulnerability scanning reports. We also performed a series of experiments to validate our approach and gain further insights into the problem. The experimental results demonstrate the potential of the proposed approach as well as the effectiveness of our techniques.

The rest of this paper is organized as follows. The next section describes our techniques to integrate and reason about complementary intrusion evidence. Section 3 presents the results of our initial experiments. Section 4 discusses related work. Section 5 concludes this paper and points out some future research directions.

2. Reasoning Framework

In this section, we present our techniques to reason about complementary intrusion evidence, including IDS alerts and reports from system monitoring tools or vulnerability scanning tools. In the following, we first describe our representation of intrusion evidence, and then present the framework to reason about complementary intrusion evidence using Bayesian networks.

2.1. Modeling Intrusion Evidence

We classify intrusion evidence into two categories: *event-based evidence* and *state-based evidence*. Event-based evidence refers to observations (or detections) of attacks. For example, an IDS alert of a buffer overflow attack against `sshd` is event-based evidence. State-based evidence refers to observations of the *effect* of attacks on system states. For example, the existence of a rootkit[1] on a machine is state-based evidence indicating that the machine has been compromised.

1 A rootkit is a collection of tools (programs) that a hacker uses to mask intrusion and obtain administrator-level access to a computer or network (`http://searchsecurity.techtarget.com`).

2.1.1. System Attributes and State-Based Evidence We follow [6, 25] to represent system states (e.g., vulnerabilities, user access privileges, and network connectivities) as *system attributes* (or simply *attributes*), each of which is a boolean variable representing the system's state. Notation-wise, we use a system attribute directly to represent that it is True, and use its negation to represent that it is False. There may be implication relationships between attributes, which also come from expert knowledge. For example, `RootPrivilege` implies `FileTransferPrivilege`, which indicates that an attacker having the root privilege also has the privilege to transfer files from/to the system. Note that such a representation can be extended to include variables to provide more flexibility. For example, we may use `RootPrivilege(x)` to represent the attacker has acquired root privilege on host `x`. However, for simplicity, we do not do so in this paper.

State-based evidence consists of observations on system attributes related to possible attacks. They may be collected by system scanning tools. We refer to the change of an attribute as an *attribute alteration*. Attribute alterations are detected by system monitoring tools, comparing the system scanning reports, and human observations. The *timestamp* of an attribute alteration is the time when the alteration is detected or inferred. Such a timestamp can be stored together with each attribute alteration.

For convenience, we refer to the probability for a system attribute to be True as the *confidence* in the attribute. When a system attribute is in negation form, the confidence in the attribute is the probability that the negation form is True. Compared with IDS alerts, reports by scanning/monitoring tools are more reliable due to the verifiable nature of most system attributes. We can assume the confidence in a verifiable attribute is 1. However, some system attributes may not be verifiable because of the absence of an appropriate scanner. In addition, some system attributes are difficult to check due to the security policy on the target system or performance reasons. In such cases, unless we have any further knowledge or evidence about the attribute, we assume the confidence in such an attribute is 0.5. Intuitively, this represents the lack of information about the state of the attribute.

2.1.2. Event-Based Evidence Typical sources of event-based evidence include event logs, IDS alerts, network traffic logs, system call logs, etc. Different kinds of logs provide event-based evidence on the system in different granularities and toward different aspects of the system. In this paper, the only event-based evidence we consider is IDS alert, which is in a coarser granularity but more understandable by human compared with other types of system logs. We will use IDS alerts and event-based evidence interchangeably in the rest of the paper. Our representation of IDS alerts is closely related to our model of attacks. Thus, we first introduce our representation of attacks before discussing IDS alerts.

Similar to [6,25], we model an attack as an atomic transformation that establishes a set of system attributes called *postcondition*, given a logical condition called *precondition* over system attributes. Intuitively, if the precondition of an attack is satisfied, the attack can then transform the system into the state specified by its postcondition. IDS alerts are not exactly the attacks launched toward the target due to the imperfection of current IDSs. An IDS may report a false alert or miss an actual attack. To facilitate the reasoning about IDS alerts, we use the prior confidence of each attack to represent its quantitative property. The *prior confidence of an attack type T*, denoted *Pr(T)*, is the prior belief we have about the probability for a corresponding alert to represent an actual type *T* attack. The prior confidence of each type of attack can be gathered by analyzing historical data. It represents our prior knowledge about IDS alerts based on our previous experience. One may observe that the probability for each attack type varies during different time period as they are dependent on not only the quality of the IDSs, but also the attack frequency and background activities in the network. However, in the later part of this paper, we will see that our reasoning approach is still useful despite the dynamic nature of the prior confidences, as it reduces the uncertainty of intrusion evidence when additional verified evidence is considered. In some sense, *Pr(T)* is the *belief* that a type *T* alert is a real instance of attack, and our reasoning framework is to increase or decrease our belief in alerts based on complementary intrusion evidence. Similar to the confidence in a system attribute, we refer to the probability that an IDS alert corresponds to a *successful attack* as the *confidence* in the alert.

We summarize our prior knowledge about IDS alerts and attacks below:

- An IDS alert *e* of attack type *T* has the probability *Pr(T)* to be a real attack;
- A real attack *E* has probability 1 to be successful when its precondition is satisfied by the system attributes before the attack happens;
- A real attack *E* has probability 0 to be successful if its precondition is not satisfied by the system attributes before the attack happens;
- The attributes in the postcondition of a successful attack *E* are True after the attack happens.

2.2. Basic Reasoning framework

In normal situations, a system should stay in a legitimate state. Starting from a legitimate system state, an attacker may launch a sequence of attacks to get the system into some intermediate states, and finally into the attacker's objective state. It is easy to see that there exist causal relationships among attacks and system attributes. Our approach is to use these causal relationships to reason about complementary IDS alerts and system attributes reported by scanning/monitoring tools. Specifically, we organize IDS alerts and system attributes into Bayesian networks [16] based on those causal relationships, and use these Bayesian networks to reason about complementary intrusion evidence.

2.2.1. Network Structure To identify and represent these causal relationships, we integrate IDS alerts with system attributes based on the preconditions and postconditions of attacks. Specifically, we place IDS alerts, available system attributes, and system attributes possibly modified by the corresponding attacks into a directed graph, which we call an *alert-attribute network*.

Each node in such a graph is a binary variable representing either an IDS alert or a system attribute. When a node represents a system attribute, it can denote either a piece of state-based evidence (e.g., scan report), or an inferred attribute alteration caused by an IDS alert. Each node is timestamped. The timestamp of an alert node is the time when the corresponding activities take place, while the timestamp of an attribute node is the time when the attribute alteration is observed or inferred.

All edges in the graph are directed. An edge from an alert node to an attribute node represents that the corresponding attack changes the system attribute into this new state. An edge from an attribute node to an alert node represents that the attribute is a part of the precondition of the corresponding attack. An edge from an attribute node to another attribute node represents that the first attribute implies the second attribute. There are no edges that connect two alert nodes together directly.

We construct such a graph starting with the initial system state, which is represented in the graph as a set of attribute nodes corresponding to the initial attributes. As time goes by, new IDS alerts and system monitoring reports are raised. When a new IDS alert is reported, a corresponding alert node is added into the graph only if the alert's precondition is evaluated to be True given the attributes presented in the graph by the time. Also, edges are added from the latest attribute nodes corresponding to the attributes in the alert's precondition to the newly generated alert node (to represent the causal relationships). To serve the same purpose, edges from the alert node to its postcondition attribute nodes are also established when they are created. For each attribute node in the alert's postcondition, if nodes related to the same attribute already exist in the graph, which could either be caused by some previous alerts or reported by system monitoring tools, an edge from the latest such node to the new node is added to represent the implication relationship. By doing so, each attribute node in the graph represents the accumulative effects on the attribute of all the prior related alerts. Note that the construction and analysis processes can be done offline following the time sequence of the evidence in IDS alert logs and scan reports.

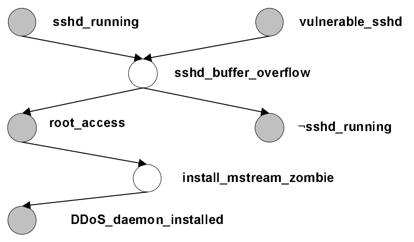


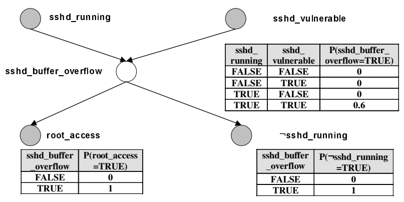
Figure 1. An attack-attribute network example

Figure 1 shows an example alert-attribute network, which is constructed as discussed above. The gray nodes represent initial or updated system attributes, and the white nodes represent IDS alerts. For simplicity, we do not show the initial system attributes that are not involved in the precondition or postcondition of the corresponding attacks. Alert `sshd_buffer_overflow` indicates an attempt to compromise the system through the vulnerable sshd. The precondition of `sshd_buffer_overflow` is `sshd_running`$\wedge$`vulnerable_sshd`, and the postcondition is {¬`sshd_running`, `root_access`}. Thus, this attempt can be successful since its precondition is satisfied in the system state. As a result, this attack introduces two attribute alterations: ¬`sshd_running` and `root_access`. In other words, the attacker stops the sshd daemon and gains root access to the system. As shown in Figure 1, the attacker then installs a mstream zombie program, changing the attribute `DDoS_daemon_installed` from False to True.

2.2.2. Conditional Probabilities A Bayesian network is a directed acyclic graph (DAG), where each directed edge represents a causal relationship between the two ends of the edge, and each node stores a conditional probability table describing the statistical relationships between the node and its parent nodes [16].

Based on the construction of the alert-attribute network, it is easy to see that a graph constructed in that way is acyclic. Indeed, all the edges are from previously existing nodes to newly added nodes, and thus will not result in any cycle. From our discussion above, the causal relationships among the nodes in an alert-attribute network are obvious. Now we discuss how to determine each node's conditional probability table so that the alert-attribute network becomes a Bayesian network.

When an IDS alert e is reported, the probability for the alert e to be a real attack is $Pr(e)$. The variable e being True represents that the corresponding attack is successful. We assume an attack will succeed if its precondition is satisfied. Thus, the probability of e being True is the prior confidence of the corresponding IDS alert when its precondition is satisfied, or 0 otherwise. Since the precondition of an attack is a logic formula of system attributes, the conditional probability of an alert node can be easily derived. The conditional probability table associated with node `sshd_buffer_overflow` in Figure 2 shows such an example, where we assume Pr(`sshd_buffer_overflow`) = 0.6. Note that the probability of an IDS alert variable being False under these preconditions can be easily computed from the above probabilities. Thus, we do not include them here.



sshd_ running	sshd_ vulnerable	P(sshd_buffer_ overflow=TRUE)
FALSE	FALSE	0
FALSE	TRUE	0
TRUE	FALSE	0
TRUE	TRUE	0.6

sshd_buffer _overflow	P(root_access =TRUE)
FALSE	0
TRUE	1

sshd_buffer _overflow	P(¬sshd_running =TRUE)
FALSE	0
TRUE	1

Figure 2. Conditional probability tables

Conditional probability tables associated with system attributes are even simpler to compute. Indeed, if an IDS alert e represents a successful attack, all the system attributes in its postcondition should turn to True. Otherwise, the system attributes that are False before the IDS alert should remain False. If two attribute nodes of the same attribute are connected together with an edge representing implication relationship, and the earlier one is True, the latter one should also be True. Thus, the conditional probability of a system attribute a being True would be 1 if at least one of its parent variables (either alert nodes or attribute nodes) is True, and 0 if all its parent variables are False (unless it is reported by system scanning/monitoring tools). The tables associated with `root_access` and ¬`sshd_running` show examples of such conditional probabilities. Similar to the above example, we only show the probabilities for the attributes to be True, from which the probabilities for the attributes to be False can be easily computed.

2.2.3. Reasoning about Intrusion Evidence The Bayesian networks constructed in this way offer an excellent opportunity to reason about the uncertain intrusion evidence, particularly the IDS alerts. We call those attributes with a confidence value of 1 the verified attributes. The report of such verified attributes are observations of facts. When new verified attributes are reported by system monitoring/scanning tools, we can use these observations to re-compute the confidence values in the related previous objects in the network with Bayesian inference. And for each node in the Bayesian network, its final probability value is the combined result of all the evidence and knowledge. Take the Bayesian network shown in Figure 2 as an example. We may be uncertain about an IDS alert reporting a buffer overflow attack against `sshd`, since the IDS has reported the same type of alerts incorrectly in the past. However, if by scanning the system we find that `sshd` is not running properly after the IDS reports this alert, we can then update the confidence in $\neg sshd\_running$ to be 1. Thus, we are more certain about the alert, which caused the attribute alteration. Though human users would do the same reasoning, placing these evidence into Bayesian networks offers additional benefits, since such a reasoning process can then be performed automatically and systematically. Also such reasoning could become too difficult for human users when dealing with very complicated scenarios.

It is easy to see that the more verified state-based evidence we have, the better judgment we can make by reasoning about the uncertain IDS alerts and system states. This suggests that we should monitor the system closer and scan the system more frequently, as system monitoring tools and vulnerability scanning tools usually generate evidence with high confidence value. However, such monitoring and scanning are often expensive and may hurt the other applications by consuming resources. Thus, it is important to determine the right balance for system monitoring and scanning activities. Nevertheless, this problem is out of the scope of this paper. We leave it for future consideration.

2.2.4. Merging Attribute Nodes As discussed earlier, there may be edges between attribute nodes corresponding to the same attribute, which represent implication relationships between them. We observe that in certain cases, such attribute nodes can be merged without affecting the reasoning about intrusion evidence in alert-attribute networks. This observation is reflected by Lemma 1, which is presented next. For the sake of presentation, if two attribute nodes `A` and `B` are connected with edge `(A, B)`, we refer to the action of removing node `A` with all its outgoing edges and redirecting all its incoming edges to node `B` as *merging* `A` *into* `B`.

Lemma 1 *Consider two attribute nodes* `A` *and* `B` *corresponding to the same attribute and connected by an edge* `(A, B)`*. If either there is no other outgoing edge from node* `A` *or* `A` *is instantiated (verified), merging* `A` *into* `B` *does not change the probability of any other node when reasoning about intrusion evidence.*

For space reasons, the proof of this lemma is not provided here but in the full version of this paper [31]. With Lemma 1, we can recursively merge attribute nodes that satisfy the condition specified in Lemma 1 to reduce the complexity of the network structure without affecting the reasoning result.

2.3. Alert Aggregation and Abstraction

In reality, IDSs often generate a large number of alerts for the same type of attack. Also, IDSs usually raise different alerts for similar attacks, or variations of the same attack. AS a result, many alert nodes share the same parent nodes and child nodes in the Bayesian network. This introduces two problems. First, the number of entries in conditional probability table of their children nodes of those alerts is exponential to the number of alerts, which makes it difficult to take advantage of existing Bayesian network tools. Second, the effect of additional evidence will spread over these alerts as we do not know which of them indeed contributes to the modification of system attributes. In practice, we usually do not care about the subtle difference between the alert variations and which particular alert is the actual successful one, but whether at least one of them is successful. Thus, a natural approach to addressing the above problem is to abstract alert variations into one common alert and aggregate such alerts together into one single node, which represents "at least one of the component alerts corresponds to a successful attack".

The conditional probability table of an aggregated alert node can be computed similarly. However, we need to use aggregated prior confidence value Pr_a, which represents the probability that at least one of its component alerts corresponds to an actual attack. Given n component alerts of an abstract attack type T that are merged into one aggregated alert, the aggregated prior confidence $Pr_a(T)$ can be computed as

$$Pr_a(T) = 1 - \prod_{i=1}^{n}(1 - Pr(Type_{a_i})),$$

where $a_1, ..., a_n$ are the alerts to be aggregated, and $Type_{a_i}$ is the attack type of a_i.

2.4. Hypothesizing about Missed Attacks

When there are missed attacks, the effect of the attacks on the system will not be reflected in the alert-attribute network. As a result, some later alerts corresponding to successful attacks may be considered false. In other words, the current model only works when there are no missed attacks. (Note that this is a common problem shared by almost all alert correlation methods.)

We observe that when successful attacks are missed by IDSs, it is still possible for the system monitoring tools to catch the impact of the attacks on the system states. In other words, we may observe unexpected attribute alterations. Such a case essentially causes *inconsistency* in the alert-attribute networks, where a new attribute node is added without any node leading to it.

Inconsistencies are almost always caused by missed attacks: An "unexpected" attribute alteration causing the inconsistencies can either be directly caused by some successful attack missed by IDSs, or by a detected successful attack whose precondition is not satisfied in the network due to previously missed attacks. The only exception is that it could be caused by false alerts if the monotonicity property of attacks does not hold for some particular types of attacks. That is, a successful attack disables other attacks' preconditions. According to [25], this kind of attacks are very rare. We can always recognize such attacks and pay additional attention in the investigation when they are involved. Thus, we propose to hypothesize about missed attacks based on the inconsistencies in alert-attribute networks.

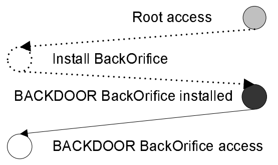


Figure 3. An example of hypothesized attack

Figure 3 shows an example to hypothesize about missed attacks to resolve an inconsistency. When the system monitoring tool reports the fact that a backdoor "BackOrifice" was found in the local system, the system adds node "BACKDOOR BackOrifice installed" to the graph immediately, which activates the precondition of the later alert "BACKDOOR BackOrifice access". However, there is no previous node possibly causing the "BackOrifice installed" attribute set to True. To fill in this gap, we look up the graph structure for established attributes and attacks, the knowledge base for possible attacks that can cause this attribute alteration, and the log of previously dropped alerts for possible related attacks. According to the above information, we make a hypothesis of a possibly missed attack "Install BackOrifice", linking the attribute nodes "Root access" and "BackOrifice installed". The hypothesized node and edges are presented with dotted lines in the figure.

A hypothesis of a possibly missed attack infers that (1) the attack has happened, (2) the attack has been missed by IDSs, and (3) the attack is successful. The probability of a hypothesized attack being a correct one can be estimated as $P_{hypothesis} = P_{happened} \cdot P_{missed} \cdot P_{successful}$, where $P_{happened}$ is the probability for the attack to have happened, P_{missed} is the prior probability for the IDS to miss the attack, and $P_{successful}$ is the probability for the attack to succeed if it happens. As P_{happen} fully depends on the attacker's knowledge and personal preference and is unpredictable, we only study the value $P_{missed} \cdot P_{successful}$, which represents the probability for the hypothesis to be `True` given the condition that it has actually happened. From our previous discussion, the successfulness of an attack depends on whether its precondition is satisfied by the system attributes. Thus, the conditional probability table of a hypothesis node over the attributes in the attack's precondition is similar to a normal alert node except that the non-zero values in the conditional probability table is P_{missed} instead of *Pr*. Accordingly, we refer to the probability computed from the Bayesian network with this conditional probability table the confidence in the hypothesized attack. Although this confidence value has a different meaning from those for normal alert nodes, it still shows which hypothesis is more possible given the available evidence.

We add the the hypothesized attacks with the corresponding conditional probability tables into the alert-attribute network. From the earlier discussion, we can see that such a hypothesis is made and placed into the alert-attribute network only if the attack is possible given the system state at the time. With the Bayesian network's belief update, we can always keep the hypotheses consistent with the newest observations. For example, we may find negative evidence against a hypothesis, and then the Bayesian inference process will update the probability of the hypothesized attack to 0, implying that the hypothesis cannot be a successful attack.

Validation is necessary for all hypotheses. From the above discussion about Bayesian inference about the hypotheses, we can see that the validation process is already embedded in the Bayesian inference process.

As more IDS alerts are reported, the Bayesian network will grow larger and larger. Some techniques (e.g., sliding time window) can be adopted to deal with this issue. We leave it as our future work.

3. Experimental Results

We have performed a series of experiments to evaluate the effectiveness of the proposed techniques. In our experiments, we connected three PCs through a hub in an isolated network. For convenience, we refer to them as *attacker*, *victim*, and *IDS*. We launched attacks from the attacker against the victim, while monitoring the attacks on the IDS.

System Setup: We use Snort version 1.9.1 [24] as the IDS sensor. We also use Nessus [3] and XScan [30] as the vulnerability scanning tools. We evaluate our techniques with five attack scenarios. The goals of these attack scenarios vary from modifying the target's web page to converting the target machine into a part of attacker's own distributed network. Some attack scenarios target MS Windows systems, while the others target Linux systems. Accordingly, the victim runs either Windows or Linux, depending on the attack scenarios. We run TripWire [5] (for MS Windows) and Samhain [4] (for Linux) on the victim as the file system integrity monitoring tools. We also run Trojan horse scanning tools Tauscan [27] (for MS Windows) and chkrootkit 0.43 [1] (for Linux) on the victim as additional system scanning tools. We developed a program to automatically generate alert-attribute networks from the IDS alerts and the reports of these scanning tools, and then use JavaBayes [2] to make inference using these networks.

To simulate the realworld system administration, we configure the file system integrity monitoring tools (Tripwire and Samhain) to monitor important files and directories only, i.e., system configurations files, service configuration files, and the main webpage files.

To mimic an operational network, we also inject background traffic into the network during our experiments. We randomly select one of the training datasets (the training dataset on Monday in the third week) in the 1999 DARPA datasets [19] as the background traffic in the experiments, as it is attack free. This background traffic triggers 325 alerts in Snort, which are all false alerts of course.

In the rest of this section, we first present the analysis of Scenario 0 in detail, and then summarize the results of all five attack scenarios. Additional details of the other four attack scenarios are included in the Appendix of the full version of this paper.

Scenario Detail: In this attack scenario, the attacker exploits the remote buffer overflow vulnerability in some old versions of Serv-U ftp server to get administrative access. The victim machine is a Windows box running a vulnerable Serv-U 5.0 ftp server with default public anonymous access. The victim also runs Norton anti-virus with file system real-time protection. When the system attempts to access a file containing known virus or backdoor, the file system real-time protection will quarantine the file.

The attack scenario includes five steps:

1. remote buffer overflow attack against the Serv-U,

2. attempt to install BackOrifice on the victim, which was quarantined by the Norton anti-virus,

3. kill the Norton anti-virus process with system process tools through the remote administrative shell,

4. install the BackOrifice again (successful), and

5. changing the web page through BackOrifice.

The initial system attributes include Serv-U 5.0 running on port 21, anonymous ftp access, and Norton Anti-virus running with file system real-time protection.

During the attack process, Snort reported 2 alerts:

- 1 `FTP command overflow attempt` alert
- 1 `BACKDOOR BackOrifice access` alert

Norton also logged that BackOrifice was found in the file system and quarantined successfully during the attack period. In the end, Tripwire logged and reported the modification to the web page file and the system logged that Norton anti-virus was shut down.

Reasoning: Our alert-attribute network generation tool generated the network shown in Figure 4 based on the above information and the prior probabilities and attack type information, which are included in the appendix of the full version of the paper.

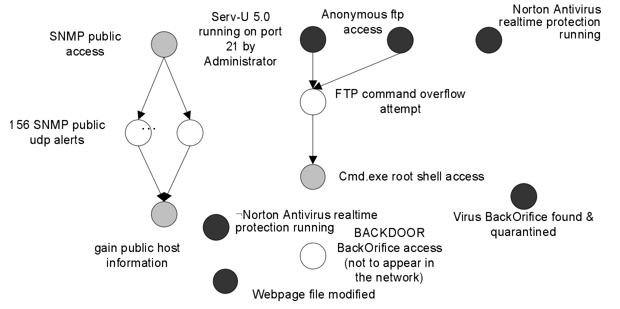

SNMP public access
Serv-U 5.0 running on port 21 by Administrator
Anonymous ftp access
Norton Antivirus realtime protection running
156 SNMP public udp alerts
...
FTP command overflow attempt
Cmd.exe root shell access
gain public host information
¬Norton Antivirus realtime protection running
BACKDOOR BackOrifice access (not to appear in the network)
Virus BackOrifice found & quarantined
Webpage file modified

Figure 4. Initial alert-attribute network

To distinguish between different types of nodes in a Bayesian network, we use white nodes to denote IDS alerts, gray nodes to denote unverified system attributes, and black nodes to denote verified system attributes. The relative vertical position of nodes in the graph represents the relative time order among nodes.

Note that Figure 4 includes 156 "`SNMP public access udp`" alerts, which results in 2^{156} entries in the conditional probability table of `gain public information`. Computing with such a conditional probability table is out of JavaBayes' handling capacity. However, after alert aggregation, the 156 nodes are aggregated into a single node and thus can be handled easily by JavaBayes.

Now let us look at possible missed attacks. There are several obvious inconsistencies in Figure 4. There are no detected alerts causing the verified attributes "`Norton Anti-virus not running`", "`Virus BackOrifice found & quarantined`", and "`Webpage file modified`". Based on our knowledge about attacks, "`Shut down Norton Anti-virus via cmd.exe shell`" and "`Install BackOrifice`" are the only possible hypotheses that can fill in the first two gaps. For the attribute "`Webpage file modified`", it could be done through remote control via either cmd.exe shell or BackOrifice access. The first option implies hypothesized remote control via cmd.exe, while the second one implies hypothesized installation of BackOrifice after Norton was shut down. These hypotheses lead to a new alert-attribute network in Figure 5. In Figure 5, the dotted nodes and edges denote hypothesized attacks and corresponding causal relationships. Conditional probability table of each node can be generated automatically given the network structure and prior probability values. Then JavaBayes generates updated confidence values of each node in this Bayesian network. The confidence values of the related alerts before and after reasoning are shown in Figure 6. We can see significant increases in the confidence values of successful attacks; however, all the false alerts have either decreased or unchanged confidence.

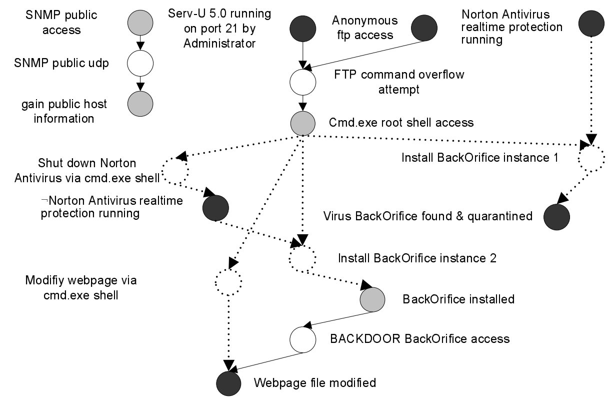

SNMP public access
Serv-U 5.0 running on port 21 by Administrator
Anonymous ftp access
Norton Antivirus realtime protection running
SNMP public udp
FTP command overflow attempt
gain public host information
Cmd.exe root shell access
Shut down Norton Antivirus via cmd.exe shell
Install BackOrifice instance 1
¬Norton Antivirus realtime protection running
Virus BackOrifice found & quarantined
Install BackOrifice instance 2
Modifiy webpage via cmd.exe shell
BackOrifice installed
BACKDOOR BackOrifice access
Webpage file modified

Figure 5. Updated alert-attribute network

We also find some interesting observations in Table 6. The confidence values in three of the hypothesized nodes turned into 1, and two of them are the two options to resolve the same inconsistency. As we have discussed in Section 2.2.2, unless a hypothesis is the only option to solve the inconsistency, a confidence value of 1 for a hypothesized attack does not mean that the attack must have happened. Instead, it implies that *if* that attack has happened, it must be successful. Thus, although the confidence values for the two hypothesized nodes are both 1, it does not mean that both attacks must have happened. However, comparing the probability of the path from the initial verified attributes to the later verified attribute (by multiplying the probabilities of all the intermediate nodes along the path), we find that the one through "`Modify web page via cmd.exe`" has a greater probability than the other one. Although it is not what exactly happened in our experiment, it shows that both methods can achieve the goal of modifying web page without being detected, and modifying through established remote cmd.exe shell is simpler and easier compared to the other option, which requires several extra attack steps. Also, the probability of a hypothesized node being 0 means either it is not missed by the IDS, or it is a failed attack attempt.

ALERT NAME	BEFORE	AFTER	INCREASE
FTP command overflow	0.3	1	233.30%
BACKDOOR BackOrifice access	0.3	0.6	100%
Individual SNMP public access udp	0.075	0.075	0%
aggregated SNMP public access udp	0.5	0.5	0%
other 169 alerts	0.25	0	-100%
Shut down Norton Antivirus (Hypothesized)	N/A	1	N/A
Install BackOrifice Instance 1 (Hypothesized)	N/A	0	N/A
Install BackOrifice Instance 2 (Hypothesized)	N/A	1	N/A
Modify web page (Hypothesized)	N/A	1	N/A

Figure 6. Changes in confidence

Using Confidence for Intrusion Detection: With the reasoning framework for intrusion evidence, we are able to associate a quantitative measure (i.e., confidence) with each IDS alert.

In our experiments, we used a confidence threshold to determine whether an IDS alert is a successful attack or not. Specifically, if the confidence in an alert is greater than or equal to the threshold, we accept the alert. Otherwise, we simply drop it. We change the threshold value between 0 and 1, and collect the detection rates and false alert rates. To compare the results in different situations, we repeated the above process in two cases: (1) without alert aggregation and abstraction, (2) with alert aggregation and abstraction. The performance graphs for the five attack scenarios are very similar.

In our evaluation, we abuse the notions of detection rate and false alert rate to represent the *detection rate of successful attacks* and *false alert and failed attack rate*, respectively. Figure 7 shows the detection rate and false alert rate w.r.t. different thresholds in all cases for one of our scenarios. (Since the meaning of the confidence in a hypothesized attack is different from that in an IDS alert, we do not consider hypothesized attacks in this evaluation.) This figure shows that the Bayesian reasoning with verified evidence can significantly increase the detection rate and decrease the false alert rate with appropriate threshold values.

Summary of Results: In the following, we summarize the results obtained from all the five attack scenarios. We first discuss the impact of the proposed techniques on alerts, and then describe the results about hypothesized attacks.

We use a simple metric named confidence ratio to examine the usefulness of the proposed techniques. Specifically, a *confidence ratio* is the ratio between the average confidence of alerts corresponding to successful attacks and the average confidence of the other alerts (i.e., false alerts and alerts corresponding to failed attack attempts).

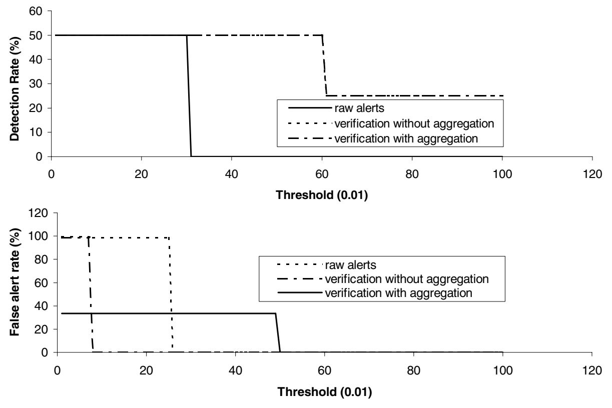


Figure 7. Detection rate and false alert rate

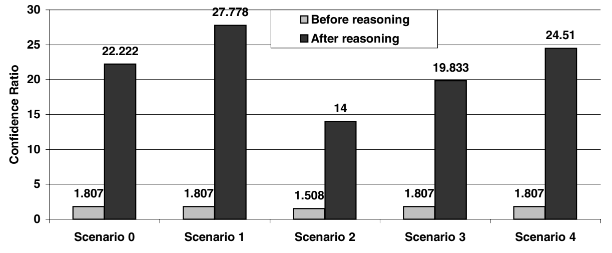


Figure 8. Confidence ratio changes

Figure 8 shows the confidence ratios in all five attack scenarios before and after using the proposed techniques. (We have discussed one scenario in the previous subsection; details of the other scenarios can be found in the full paper [31].) These results indicate that with the proposed techniques, the average confidence in alerts of successful attacks are greatly increased compared with the average confidence in the other alerts (false alerts and alerts for failed attack attempts). In fact, the average confidence in the other alerts either remain the same or decrease.

Scenario	0	1	2	3	4
Accuracy	75%	100%	50%	100%	50%

Table 1. Accuracy of hypotheses

We made ten hypotheses during the analysis of the five attack scenarios. Table 1 shows the accuracy of these hypotheses. In the experiments, six out of the ten hypothesized attacks are actual successful attacks missed by Snort, and one out of the other 4 hypotheses is a failed attack attempt. Among the seven real attacks, we have definite confidence that four of them must have happened from the alert-attribute networks. The result shows that with sufficient local system evidence, our model is efficient and effective in discovering some missed attacks.

4. Related Work

The techniques closest to ours are M2D2 [20] and the mission-impact-based correlation method [23], which have been briefly discussed in the introduction. However, M2D2 is intended to provide a formal model to represent intrusion related information, while the mission-impact-based method requires human experts to specify the correlation models. In contrast, our method can automatically construct Bayesian networks of IDS alerts and other complementary evidence based on the knowledge of individual attacks.

Another approach was proposed in [22] to hypothesize about missed attacks based on the pre/post-conditions of known attacks. Our approach differs in that the hypotheses made in our model is based on not only the pre/post-conditions of known attacks but also the available system states. The techniques proposed in [8,21,28] are also based on modeling individual attacks. However, they focus on IDS alerts without exploiting other information sources. Our approach can potentially get more concrete results due to the additional, complementary information.

There are other alert correlation techniques. Some approaches cluster alerts on the basis of the similarities between the alert attributes (e.g., [7,29]), or hierarchies of concepts (e.g., [17,18]). Some others correlate alerts by matching alerts against known attack scenarios, using languages such as STATL [13] and LAMBDA [9]. These methods are potentially complementary to our approach.

Our approach is also related to the recent results on vulnerability analysis (e.g., [25]). However, our approach is aimed at reasoning about intrusion evidence rather than identifying possible sequences of attacks.

5. Conclusion and Future Work

In this paper, we developed a method to integrate and reason about complementary intrusion evidence, including IDS alerts, reports of system monitoring or vulnerability scanning tools, and even human observations. By using the interdependency between attacks and system states, we combine various kinds of intrusion evidence into Bayesian networks, and infer about uncertain IDS alerts based on additional observations of system states. We further proposed to refine these Bayesian networks through alert aggregation and abstraction, so that we can focus on the reasoning about existences of successful attacks and use complementary intrusion evidence more effectively. Our initial experimental results have demonstrated the potential of the proposed techniques.

This paper is only the starting point of our effort to integrate and reason about complementary intrusion evidence. In our future work, we will investigate additional techniques to improve the performance. In particular, we will study the use of dynamical Bayesian networks in processing streams of IDS alerts and other intrusion evidence, investigate approaches to handling attacks missed by IDSs, and perform experiments with large sets of intrusion evidence.

References

[1] checkrootkit. `http://www.checkrootkit.org`. Accessed on Feb. 4, 2004.

[2] Javabayes. `http://www-2.cs.cmu.edu/~javabayes/Home/`. Accessed on Oct 10, 2003.

[3] Nessus. `http://www.nessus.org`. Accessed on Feb. 4, 2004.

[4] Samhain. `http://la-samhna.de/samhain/`. Accessed on April 4, 2004.

[5] Tripwire. `http://www.tripwire.com`. Accessed on Feb. 4, 2004.

[6] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224, November 2002.

[7] F. Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th Annual Computer Security Applications Conference*, December 2001.

[8] 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.

[9] F. Cuppens and R. Ortalo. LAMBDA: A language to model a database for detection of attacks. In *Proc. of Recent Advances in Intrusion Detection (RAID 2000)*, pages 197–216, September 2000.

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

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

[12] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Recent Advances in Intrusion Detection*, LNCS 2212, pages 85 – 103, 2001.

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

[14] D. Farmer and W. Venema. SATAN: Security administrator tool for analyzing networks. `http://142.3.223.54/~short/SECURITY/satan.html`.

[15] Fyodor. Nmap free security scanner. `http://www.insecure.org/nmap`, 2003.

[16] F.V. Jensen. *Bayesian Networks and Decision Graphs*. Statistics for Engineering and Information Science. Springer, 2001.

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

[18] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *The 8th ACM International Conference on Knowledge Discovery and Data Mining*, July 2002.

[19] MIT Lincoln Lab. 1999 DARPA intrusion detection scenario specific datasets. `http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html`, 1999.

[20] B. Morin, L. Mé, H. Debar, and M. Ducassé. M2D2: A formal data model for IDS alert correlation. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, pages 115–137, 2002.

[21] P. Ning, Y. Cui, and D. S Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 245–254, Washington, D.C., November 2002.

[22] P. Ning, D. Xu, C. Healey, and R. St. Amant. Building attack scenarios through integration of complementary alert correlation methods. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS '04)*, pages 97–111, February 2004.

[23] P.A. Porras, M.W. Fong, and A. Valdes. A mission-impact-based approach to INFOSEC alarm correlation. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, pages 95–114, 2002.

[24] M. Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 1999 USENIX LISA conference*, 1999.

[25] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2002.

[26] S. Staniford, J.A. Hoagland, and J.M. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2):105–136, 2002.

[27] Tauscan. `http://www.agnitum.com/products/tauscan/`.

[28] S. Templeton and K. Levitt. A requires/provides model for computer attacks. In *Proceedings of New Security Paradigms Workshop*, pages 31 – 38. ACM Press, September 2000.

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

[30] X-scan. `http://www.xfocus.org`.

[31] Y. Zhai, P. Ning, P. Iyer, and D.S. Reeves. Reasoning about complementary intrusion evidence. Technical Report TR-2004-25, Department of Computer Science, North Carolina State University, 2004.