Integrating Intrusion Alert Information to Aid Forensic Explanation: An Analytical Intrusion Detection Framework for Distributive IDS

Bon K. Sy[1, 2]

[1]**Queens College**
Computer Science Department
65-30 Kissena Blvd.
Flushing NY 11367
U.S.A.

[2]**Graduate Center/CUNY**
Computer Science Department
365 Fifth Ave
New York NY 10016
U.S.A.

bon@cs.qc.cuny.edu

Abstract

The objective of this research is to show an analytical intrusion detection framework (AIDF) comprised of (i) a probability model discovery approach, and (ii) a probabilistic inference mechanism for generating the most probable forensic explanation based on not only just the observed intrusion detection alerts, but also the unreported signature rules that are revealed in the probability model. The significance of the proposed probabilistic inference is its ability to integrate alert information available from IDS sensors distributed across subnets. We choose the open source Snort to illustrate its feasibility, and demonstrate the inference process applied to the intrusion detection alerts produced by Snort. Through a preliminary experimental study, we illustrate the applicability of AIDF for information integration and the realization of (i) a distributive IDS environment comprised of multiple sensors, and (ii) a mechanism for selecting and integrating the probabilistic inference results from multiple models for composing the most probable forensic explanation.

1. Introduction

Current intrusion detection technology provides rich information about an attack. However, the primary focus of intrusion detection is on one-bit information about whether the security has been compromised. Forensic analysis, on the other hand, attempts to understand and explain what has happened to the security environment, and how a security breach could happen [1-3].

We present an analytical intrusion detection framework (AIDF) with an underlying structure comprised of a probability model discovery and inference mechanism. The purpose of AIDF is to bridge intrusion detection with forensic analysis based on inferring and integrating alert information from distributive IDS sensors; whereas the outcomes of an inference are referred to as forensic explanations. It is important to note that forensic explanations typically are *not* what one will be seeking as the final product of a forensic analysis. Rather, forensic explanations are useful *stepping stones* for an analyst to narrow the possible attack goals to model, and provide intermediate snapshots on events useful for modeling attacks in a forensic analysis.

The relationship between intrusion detection and forensic explanation could be perceived as the following: an attack signature rule of a misuse based IDS such as Snort can be conceived as a manifestation of an illegal activity, and different combinations of n illegal activities can be represented by n-bit binary strings; whereas each binary string pattern of the n-bit constitutes a forensic explanation on the corresponding underlying illegal activities. In other words, there is a high level linguistic description of a forensic explanation, which has a semantic interpretation using predict calculus that corresponds to the truthfulness of the n-bit binary string pattern about the conjunctive occurrence of the signature rules. The utility of a forensic explanation is its potential for offering a basis on security policy modification, and on improving intrusion prevention capability; e.g., an “offline incremental” rule revision in the case of misuse based intrusion detection.

The significance of this research is a novel probabilistic based approach for uncovering hidden information during the course of intrusion detection useful for generating forensic explanations. Specifically, we will illustrate how the novel approach is applied to generating the most probable forensic explanation from the intrusion alerts produced by distributive Snort IDS sensors [4, 5].

In section 2 a brief survey on forensics, and the relationship between forensic explanation and analysis are discussed. In section 3 different types of intrusion detection technology and alarm alerts are discussed. In sections 4 the details on signature rules for misuse based intrusion detection and network monitoring are described. In section 5 an analytical intrusion detection framework (AIDF) is introduced. We will discuss an algebraic constraint formulation within the AIDF for deriving multiple models based on incomplete observations, and an inference process for discovering hidden inhibiting negative for a forensic analysis. In section 6 the theoretical foundation behind the probability model discovery for the AIDF is detailed. In section 7 the scenario of applying AIDF to the real world situation for integrating intrusion alert information from multiple sensors are discussed. The scalability issue of the proposed AIDF is presented in section 8. An illustration on distributive coverage to address the scalability issue is presented in section 9. In section 10 an experimental study and its evaluation are presented, followed by the conclusion and future work in section 11.

2. Brief Survey on Forensics

Forensic analysis [6] is commonly perceived as "the process of understanding, re-creating, and analyzing arbitrary events that have previously occurred." As of this writing, the field of computer forensics is still considered largely ad hoc [6]. Ideally, forensic analysis will explain: (1) what has happened? (2) When did it happen? (3) How did it happen? (4) Who caused it? Furthermore, it would be desirable if the forensic explanation is based on a systematic extraction and analysis of evidence that is admissible in a legal proceeding.

Unfortunately, as pointed out by Sommer [7], there is a gap between the purposes of IDSs of various types and the needs of the legal system. The gap exists not only at a purposive and functional level, but also in philosophical approach as to what is sufficient or constitutes a "proof" of an intrusion offense. Therefore, in a legal proceeding, IDS alerts or alert correlation in an ESM (Enterprise Security Management) system may contribute merely one step towards proving guilt of a criminal offense or liability in a civil matter.

To the best of our knowledge, there is no one single intrusion detection/prevention solution that can claim its alert evidence being sufficient and reliable, nor can it always satisfy the stringent requirement of the chain of custody in order for it to be admissible in a legal proceeding. This issue is further complicated by the international jurisdiction when the source of an intrusion violation and the targeted victim are from different countries. For example, forensic evidence is legally admissible in the U.S. only if (1) it is permissible (via lawful search/seizure), (2) preserved (following the chain of custody on how the evidence is collected, possessed, controlled, and made accountable), (3) trustworthiness (i.e., evidence collected during normal business operation, not aftermath), and (4) relevant. The forensic aspect of this research is by no means sufficient for legal admissibility. Rather, it is meant to help better understand what and how could happen, thereby offers insights into preventing potential malicious attacks.

One active research area in this direction is the development of models towards forensic analysis. For example, one idea is to develop the model for back tracking as exemplified by BackTracker [8]. BackTracker records system calls and processes in form of a "dependency" graph. Such a “dependency” graph reviews the dependencies and traces of the sequence of the compromised systems events. However, the criticism [9] on BackTracker is its requirement on an analyst to provide the system events to examine. Yet the suspicious files and process IDs as related to potentially compromised system events may not be easily discovered if the analysis takes place long after the intrusion.

Forensix [10] is another tool similar to BackTracker with the capability of recording system calls. Although Forensix possesses an augmented feature to allow an analyst to make database query on the activities of system calls, it is subjected to a similar constraint; i.e., an analyst has to determine what specific files to investigate based on her belief on whether such files may have been modified by an intruder's code. In other words, the success of both BackTracker and Forensix depends on the ability of an analyst to pinpoint "root cause" in an ad hoc exploratory process of a forensic analysis.

Peisert and Bishop et al [9] proposed an approach towards modeling forensic analysis ─ referred to as PB forensic approach hereafter in this paper. PB approach models a generic attack in form of a three-layer directed graph. The first layer is comprised of nodes representing events that signify the beginning of an attack. The third layer is comprised of nodes representing the end goals of an attack. An end goal of an attack is defined as the success on achieving a particular violation. The middle layer is comprised of many interconnected nodes; whereas each node in the middle layer represents an intermediate step of an attack and a unique traversal is a possible realization of an attack achieving the end goals of an intruder. In other words, one unique traversal can be thought of as a revelation of "how could an attack happen." The objective of a forensic analysis is to identify the actual traversal path among many possible ones.

PB approach performs a goal-oriented attack modeling based on the so-called requires/provides pre-conditions. Each directed path encodes the temporal order of the system events occurred over time. A transition from one node to the next node signifies the "capabilities" in the requires/provides model. Their modeling approach is based on two important assumptions. First, forensic analysis is computationally tractable if one can carefully integrate the useful information at different abstraction layers (such as low level system calls and high level alerts). In doing so, the search space of the path traversals in the middle layer can be kept manageable. Second, their model implicitly assumes absolute certainty, and sufficiency, on the "capability" attribute set for a build-out of the nodes in the middle layer. As such, their modeling approach explicitly discards the notion of implication, which estimates the theoretical possibility of a hypothesized outcome.

In this research, we argue that extending the model for forensic analysis to incorporate uncertainty can enhance its utility. Our argument is based on three observations. First, a sophisticated forensic model has to incorporate low level and high level events. The relationship between these events in different levels is often many-to-many. One way to capture the potential ambiguity of the many-to-many relationship is through the concept of entropy in information theory, which is grounded on probability theory. Second, uncertainty measure can be useful in a forensic analysis to offer a more granular forensic explanation; e.g., without a study on the level of consistency of certain time-varying statistical behavior, source/destination pair over time is not sufficient to decipher whether similar attacks are from the same or different parties. Third, a useful forensic model may render an exponentially large search space (in the middle layer). Proper probability assignment could improve the search efficiency by incorporating heuristic strategy that may bring us closer to a realistic real-time forensic analysis.

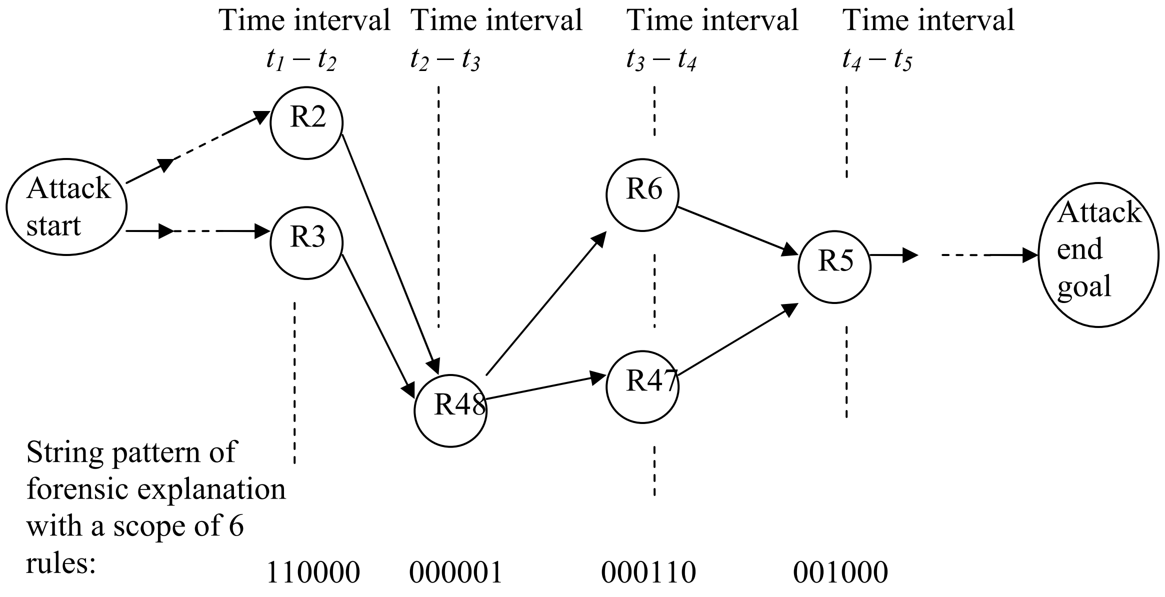
One natural way to generalize a forensic model for incorporating uncertainty is to consider Markov modeling. The advantages of extending and generalizing a forensic model to a kind of Markov model are (1) the existence of efficient search techniques (e.g., Vertibi-like search algorithm based on the joint probability of a traversal path) for a forensic analysis, and (2) the ability to reflect through uncertainty estimate the nature of many-to-many relationship between the alert events that appear at different levels of abstraction (e.g., packet stream characteristics such as protocol and payload contents on one extreme and the one-bit intrusion alert on the other extreme).

To realize the advantages just mentioned, we need to map the elements of a forensic model to that of a Markov model. An essential step for the mapping is the task of deriving probability models for the nodes characterizing the intermediate attack goals in the middle layer of a forensic model. This is necessary because each node in a Markov model encompasses a probability distribution (i.e., a probability model by itself). Time stamps of the events define the temporal constraints on the order of the nodes represented by the transitional directed path in the model. Below is an example rule set that will be used to illustrate the case in point and forensic explanations:

2 BLEEDING-EDGE VIRUS HTTP Challenge/Response Authentication
3 (http_inspect) OVERSIZE REQUEST-URI DIRECTORY
5 BLEEDING-EDGE EXPLOIT Awstats Remote Code Execution Attempt

6 BLEEDING-EDGE EXPLOIT XML-RPC for PHP Remote Code Injection
47 BLEEDING-EDGE Exploit Suspected PHP Injection Attack
48 BLEEDING-EDGE EXPLOIT WEB PHP remote file include exploit attempt

Suppose rules 6 and 47 were reported in an intrusion alert, a forensic explanation with a scope covering these six rules has a corresponding 6-bit binary string pattern of 000110 that characterizes an intrusion event targeted at the vulnerability of PHP. As in another example, forensic explanation "web exploit vulnerability" could have a semantic interpretation on a 6-bit binary string corresponding to 100001. Any of such binary string with matching access attributes (qualified by, for example, source and destination address pair, session/process ID, etc.) can be used to represent a snapshot of an intermediate stage of an attack. The signature rules in the binary string can be mapped to low-level system events, or used directly in modeling an attack during a forensic analysis session. Below shows an attack model based on PB approach; where the directed arcs signify the order of the events occurred over time:


Time interval $t_1 - t_2$
Time interval $t_2 - t_3$
Time interval $t_3 - t_4$
Time interval $t_4 - t_5$
Attack start
R2
R3
R48
R6
R47
R5
Attack end goal
String pattern of forensic explanation with a scope of 6 rules:
110000
000001
000110
001000

Our research effort in this paper does not attempt to address the full extent of a forensic analysis. Instead, our focus is on the snapshot of each time interval between t_i and t_j. Nevertheless, our effort contributes to extending a forensic model to incorporate uncertainty by providing the followings:

1. Probability model discovery technique for deriving the probability distribution of nodes in each time interval based on constraints taking into the consideration of *inhibiting negative* ─ a concept of alert alarms that will be presented in the next section.
2. Inferring missing information due to *inhibiting negative* that may reveal critical system events for modeling in a forensic analysis.
3. Probabilistic inference technique for generating the forensic explanations.

3. IDS and Alert Alarm Types

Intrusion detection could be realized as host based or network based. This paper is focused on network based intrusion detection. Readers interested in the application layer intrusion detection are referred to our previous work elsewhere [11]. Network based intrusion detection could be broadly classified into two general approaches; namely, misuse based or anomaly based [12, 13, 34]. Anomaly based and misuse based intrusion detection each has its own strength and weakness. As reported elsewhere [13], hybrid approach combining misuse based and anomaly based intrusion detection has been proposed. One of the main thrusts on the hybrid approach is the potential on combining high-level alerts such as alert correlation or aggregation.

In order to fully realize the potential of hybrid approach for intrusion detection, it is desirable to have a rich language for abstracting and integrating high-level information from alert logs of multiple sensors. In particular, it is desirable to have a language that is based on multilevel abstractions and mechanisms for expressing uncertainty in characterizing events, and for restructuring the representational space of events in ways that reduce the likelihood of successful evasive attempts. As succinctly mentioned by Dolye et al. [14], the limitations of misuse and anomaly based intrusion detection stem not from the methods themselves but from reliance on inadequate expressive languages for describing significant patterns.

The language issue just mentioned is a factor that affects the success on event characterization and recognition during intrusion detection. For example, Snort does not provide a convenient way to encode in a signature rule that can express an event such as "the probability of a DoS (Denial of Service) attack in a subnet with two or more mission critical servers increases by more than 10% per hour over a period of six hours." Since the language such as the one in Snort has an inherent limitation on the expressiveness and semantics to describe temporally related events characterized by uncertainty, the extent of possible success on integrating alert information to aid forensic analysis would have to be considered within this scope. Obviously this research could not overcome the limitation just mentioned without focusing on the development of a language for describing and characterizing intrusion events. Nonetheless, this research attempts to explore one aspect of the forensic analysis permitted by the language of Snort; namely, event characterization that is based on linguistic sentences that are composed of the normal conjunction and disjunction of the occurrence of activities captured in the signature rules. As we have noted, the need for a possible cooperation between IDS and forensic systems is timely in light of the industrial efforts in the development of forensic tools such as InfiniStream (by Network Associates) [15] and NetIntercept (by Sandstorm Enterprise) [16], as well as academic research effort on the development of ForNet [17].

Since Snort relies on signature rules for intrusion detection, the implementation of its search engine must consider the possibility of DoS attack due to an exploit on the nature of the multi-rule pattern matching [18]. In other words, one may exploit the nature of multi-rule pattern matching by deliberately sending network traffic that will trigger a large number of rules, and as such, overflows the processor and consumes the memory of an inspection engine. Current anti-DoS strategy [19] to prevent this potential exploit is to perform a set partition on the rules. The basic idea behind the set partition on the rules is to group the rules into categories such as protocol field rules, generic content rules, packet anomaly rules, and IP rules. The inspection engine will first decide within which category of the rule set to perform a search. It will then conduct a search for matching rules against network traffic. Afterwards, it will store the matching rules up to the limit of a queue with some pre-defined size, and report a small subset of the matching rules stored in the queue. For example, the default configuration for Snort implementation has a queue size of eight; i.e., it stores up to eight matching rules in a queue even if there may be more than eight matching rules. The default configuration also has a limit of reporting up to three matching rules stored in the queue; in which the three rules may be chosen based on some pre-defined priority criteria, or in a probabilistic manner to prevent an attack that anticipates the behavior of the search engine. In other words, if there are more than three rules in the queue, an end-user will only be able to observe up to three matching rules in the queue. As such, there are three different kinds of *negative* behavior of misuse based intrusion detection:

1. *True negative* ─ When there is no attack and the misuse based intrusion detection system reports no attack alert.
2. *False negative* ─ When there is an attack and the misuse based intrusion detection system fails to detect and fails to generate an alert. This occurs when there is no signature rule to match the attack patterns, or the encoded signature rules fail due to either the lack of scope coverage on the new attack or practical processor limitation.
3. *Inhibiting negative* ─ When there is an attack and the misuse based intrusion detects, but fails, to report the attack. This occurs when the set of encoded signature rules indeed detects the attack, but the inspection engine does not report all the matching rules because of the anti-DoS strategy. Referring to the previous discussion, anti-DoS strategy imposes capturing and reporting limit. Consequently, when the reporting limit is reached, the matching rules that are reported *inhibit* other matching rules to be reported.

While false negative is a concept that has been widely used for possible evaluation of an intrusion detection approach, we also recognize its limitation in terms of its applicability to only simulation test. In the real world scenario, unless one could detect deviation of the behavior of the computing/networking environment and could link the deviation to the external disturbance, false negative will be very difficult, if not impossible, to detect. This is particularly so in an environment where there is no additional source(s) of IDS data for cross reference. On the other hand, false positive is a concept that has been widely used for characterizing false alarms; i.e., intrusion detection system generates an alert when there is no attack. False alarms could be particularly prevailing and subject to “base rate fallacy” [20] if the signatures rules are not properly tuned to the objective of the security policy to be enforced.

In this research, our focus is on inhibiting negative and some aspects of false negative. The premise is that the network traffic patterns encoded in the signature rules could reveal information, if combined appropriately, to provide useful forensic explanations. Thus the objective is to uncover through a discovery process the hidden matching rules that are not observable, and to develop a model through which probabilistic inference could be carried out to derive the most probable forensic explanation; whereas the model incorporates the observable matching rules and the hidden ones that are uncovered.

One may argue that generating forensic explanation through inhibiting negative is not necessarily viable. It is because, as technology progresses, one may afford to set a large queue size and to increase the number of matching rules to be reported while preserving anti-DoS strategy. In this case, the proposed research may not be valuable. While this may be acceptable from the practical point of view at the present moment, this "quick fix" solution only works when one accepts the queue size to grow linearly with respect to the number of matching rules. Irrespective to whether the size of the rule set follows Moore's Law, it is pointed out elsewhere [21] that the simple technique of linearly searching through the set of rules is becoming increasingly infeasible.

Furthermore, one may wonder whether any existing IDS other than Snort may also be subject to inhibiting negative. As we have noted, there are other IDS appliances based on Snort implementation; e.g., NinjaBox Z-series system by Endace [22] and Sourcefire IS3000 V4.0.2 [23]. While it is presented as an advantageous feature, the technical manual of Sourcefire IS3000 system has also pointed out the importance of determining the type of background traffic and the maximum load that a sensor can sustain. This is emphasized because the system will begin to drop packets/miss alerts when the maximum load is exceeded ─ suggesting the potential issues caused by inhibiting negative. In other words, the issue is not whether there is a board array of IDS systems subject to inhibiting negative. But rather the significance of the problem is whether industry-standard IDSs are subject to inhibiting negative.

Model-based probabilistic inference is proposed to address inhibiting negative. However, the motivation and utility of probabilistic inference is to explore more than just the intrusion detection methodology that decouples the dependency of its performance from the size of the rule set database. Rather, it has additional utilities from the real time perspective in regard to its ability to integrate information from the collaborative IDS sensors that realize a scaleable distributive IDS environment. This will become clearer after the analytical detection framework is introduced in sections 5 and 7. Furthermore, the proposed technique could offer insights into evasive attempts of malicious attacks. We will show two such examples using Fragroute and web based ftp exploit.

Fragroute is a tool for manipulating packets of data so that malicious attacks could be camouflaged to bypass an intrusion detection system. This tool has gained attentions few years ago when it was released. The basic idea is to exploit the fact that IDS does not perform data integrity check as stringent as a server software. An example "insertion" attacks is to send a command to a server that could be disguised by adding extraneous, illegitimate data.

Yet another example of an intrusion bypassing IDS is a simple web exploit such as ***http://ftp.sloppery_setup.com/pub/../../*** In a sloppy ftp site accessible via the http, if directory access protection is not properly configured at the OS and server setup, "./pub/../../" allows one to traverse up to a directory level that often is not supposed to be shared with the public. Many such similar patterns may exist for an exploit. It is a challenge for IDS to determine the nature of the "unusual" activities and to catch this kind of patterns via simply an explicit encoding of the "patterns" as signatures.

We argue the proposed probabilistic inference for discovering inhibiting negative could be easily extended to offer a possible avenue to address the above issues related to (1) assessing the ambiguity of the "unusual activities" through likelihood assessment of the event from the discovered probability model(s), and (2) discovering these activities of which there may not even be any explicit signature rule.

Using the "sloppy ftp site" example just discussed, one could note that the possible success on bypassing IDS is an exploit on the lack of explicit signature rule encoding in layer 3. Yet the web log at layer 7 would reveal abnormal directory listing and traversal. When such additional information is made available for this approach, one could construct a model that incorporates information made available from different sources. With such a model, probabilistic inference would become possible to help construct conjectures on possible evasive attempts of malicious attacks.

4. Some Details on Snort Signature Rules

In this section we use Snort as a basis to explain the mechanism behind the signature rules used for intrusion detection. Three simplified sample Snort signature rules are shown below:

<u>Signature rule R1:</u>

```
alert tcp any any -> $HOME_NET 22 (msg: "SSH Successful user               \
connection"; dsize: 52; flags: AP; threshold:  type both, track by_src, count 3,  \
seconds 60; classtype: successful-user; sid: 2001637; rev:3; )
```

<u>Signature rule R2:</u>

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg: "Potential SSH   \
Scan"; flags: S; flowbits: set, ssh.brute.attempt; threshold: type threshold,        \
track by_src, count 5, seconds 120; classtype: suspicious-login; sid: 2001219;  \
rev:12; )
```

<u>Signature rule R3:</u>

```
alert tcp $EXTERNAL_NET 23 -> $HOME_NET any                                  \
(msg:"TELNET client ENV OPT escape overflow attempt";                        \
flow:to_client,established; content:"|FF FA|'|01|";                          \
rawbytes; pcre:"/(\x02([\x01\x02\x03]|\xFF\xFF)){100,}/RBsm";                \
content:"|FF F0|"; distance:0; rawbytes; reference:bugtraq,12918;            \
 reference:cve,2005-0469;  classtype:attempted-user; sid:3537; rev:2;)
```

Briefly, rule R1 shows a user behavior access rule that belongs to the class "successful-user." This rule encodes a network traffic pattern that is connection-oriented using TCP with a packet payload size 52, and the flow of the traffic is from any IP using any port in the Internet to a host in the local area network specified in $HOME_NET at port 22. In addition, this rule is triggered to send an alert only if the payload packet size is 52, and the TCP flag bits "ACK" and "PSH" are checked. Furthermore, the number of login attempts from the source IP must be 3 times (or less) within the 60 seconds interval.

Rule R2, on the other hand, is an attack signature rule and belongs to the class "suspicious-login." As could be noted, R2 is mutually exclusive to rule R1, while rule R3 can occur as a standalone, or in combination with R1 or R2.

When a Snort signature rule is triggered to send an alert, the log (using, for example, CSV format) will contain the information about <TimeStamp: date/hh24:min:sec> <Protocol> <Source IP address> <Source port> <Destination IP address> <Destination port> <Signature rule>. Below is an example:

```
Dec-12-2005/03:43:01 TCP 139.103.65.163 54708 149.4.211.170.236 22 R1
```

As the size of the signature rule set grows inevitably over time because of the discovery of the new attacks, one may incline to think that the effectiveness of an IDS will remain so long as the processing power could keep up proportionally ─ thus providing a ground for re-examining the significance of the proposed AIDF. Unfortunately, this would be an over simplification on the relationship between the processing power and the size of signature rule set. We will use a configurable feature of Snort to illustrate this point.

Snort could be configured to enable or disable different kinds of preprocessors such as stream4/5 and http. While Snort treats IP/TCP/UDP traffic in the packet level, Snort could be configured (via snort.conf) to enable stream preprocessor to generate evasion alerts and/or to reassemble packets into larger "psuedopackets" for an inspection. Similarly, Snort could be configured to invoke http preprocessor to perform deep packet content inspection based on various flow depth at predefined ports. For the sake of illustration, let's consider the following signature rule in a rule set database:

```
alert tcp any any -> any any (msg: "Test rule"; content:"someTestString"; nocase; \
                              classtype: attempted-user; sid: 2400000; rev 1;)
```

When the http preprocessor is enabled with default flow depth 1460, an outbound URL request "http://www.google.com/someTestString" will trigger the rule just shown above. However, if "someTestString" appears in the http response content section but not in the URL, the rule just shown would not necessary be triggered. It is because the maximum value of the flow depth (i.e., 1460) does not guarantee an inspection on the entire server response ─ unless the flow depth is set to assume a value of 0. However, when the flow depth parameter is set to 0, Snort could hang or crash depending upon the type and the amount of traffic being inspected. In order for Snort to operate with sufficient stability, it is common to set the flow depth to be 1460, even if it means to miss reporting a rule that should have been triggered in an intrusion detection session. In other words, a database with even just one single rule could lead to inhibiting or false negative irrespective to the processing power of an inspection engine.

5. Analytical Intrusion Detection Framework

In our Analytical Intrusion Detection Framework (AIDF) that supports forensic inference and explanation generation, the basis is a binary-valued table indicating any match between network data packet and alert patterns encoded in a signature rule. As shown in the previous section, whenever there is one or more rule(s) matching the network traffic, Snort will report the network traffic information as well as the rule(s) being triggered. Below is a binary-valued representation of an example Snort log file:

Timestamp	R1	R2	R3
`2005-12-12 03:43:01`	0	1	0
…	…	…	…
`2005-12-12 03:44:03`	0	0	0
`2005-12-12 03:44:04`	1	0	0
`2005-12-12 03:44:05`	0	0	1

Table 1: Snort log in form of binary-valued table

In the above table, 0 indicates no match and 1 indicates a match of the corresponding rule. Referring to the previous section, R1 and R2 would not co-occur simultaneously. It could be formulated probabilistically as Pr(R1:1 | R2:1) = 0. If the configuration is set to have a limit of reporting only one matching rule from the queue, then one would not be able to observe the co-occurrence of R1 and R3. This is because only one of them will be reported even if there is a match for both rules and the rules are stored in the queue. In other words, the observation of any row entry with a 1 does not necessarily reveal the true underlying phenomenon. The only situation that the observation reveals the true underlying phenomenon without the "inhibiting factor" is a row with all zeros.

To further illustrate this, let's assume simple frequency count is used for estimating the occurrence likelihood of the data in table 1, and the following is obtained:

$Pr(R1:1|R2:1) = 0 \quad \Leftrightarrow \quad \sum_{R3} Pr(R1:1, R2:1, R3) = 0$

$Pr(R1:1) \geq 150/30000 \quad \Leftrightarrow \quad \sum_{R2,R3} Pr(R1:1, R2, R3) \geq 0.005$

$Pr(R2:1) \geq 60/30000 \quad \Leftrightarrow \quad \sum_{R1,R3} Pr(R1, R2:1, R3) \geq 0.002$

$Pr(R3:1) \geq 90/30000 \quad \Leftrightarrow \quad \sum_{R1,R2} Pr(R1, R2, R3:1) \geq 0.003$

$Pr(R1:0\ R2:0\ R3:0) = 0.99 \quad \Leftrightarrow \quad Pr(R1:0, R2:0, R3:0) = 0.99$

$\sum_{R1,R2,R3} Pr(R1, R2, R3) = 1$

Prob Solution model M: Pr(R1 R2 R3)		***Variable instantiation for R1 R2 R3:***							
		0 0 0	***0 0 1***	***0 1 0***	***0 1 1***	***1 0 0***	***1 0 1***	***1 1 0***	***1 1 1***
M1: Pr(R1 R2 R3)		0.99	0	0.002	0.003	0.005	0	0	0
M2: Pr(R1 R2 R3)		0.99	0.001	0.002	0.002	0.005	0	0	0
M3: Pr(R1 R2 R3)		0.99	0.002	0.002	0.001	0.005	0	0	0
M4: Pr(R1 R2 R3)		0.99	0.003	0.002	0	0.005	0	0	0
M5*: Pr(R1 R2 R3)		0.99	0.0025	0.000	0.0025	0.005	0	0	0
M6*: Pr(R1 R2 R3)		0.99	0.0017	0.000	0.0033	0.005	0	0	0

Table 2: Multiple probability model solutions

Table 2 shows six possible probability models consistent with the formulation just shown. These models are derived following the probability model discovery technique discussed elsewhere [24], which will be detailed in the next section. Suppose an alert due to rule 2 (SSH scan) is reported by the IDS, one may be interested in whether ESC overflow attempt (rule 3) also exists. In other words, one may be interested in $ArgMax_{R3}[Pr(R3|R2:1)]$. Let's assume after the models are derived, an additional piece of information is available: "ESC overflow attempt (R3) as a standalone activity without a successful SSH connection session as specified in R1 always exceeds 0.4%.” Then only the last two models (marked with *) satisfy the additional condition Pr(R1:0 R3:1) ≥ 0.004. In both cases, Pr(R3:1|R2:1) ≥ Pr(R3:0|R2:1); i.e., the most probable explanation suggests that whenever there is a SSH scan (R2), the activity of ESC overflow attempt may also occur. In other words, the most probable explanation based on the derived probability models allows us to deduce additional information about R3:1 (ESC overflow attempt) during a forensic inference session.

6. Probability Model Discovery for AIDF

In general, one can formulate a set of probability constraints for model discovery based on the available information from IDS sensor(s). If only partial information is available because of, for example, inhibiting negative, the set of probability constraints defines an under-determined algebraic system. Each probability constraint can be expressed as a linear combination of joint probability terms; whereas each joint probability term is treated as a non-negative unknown in the algebraic system. Probability model discovery is then reduced to solving an under-determined algebraic system. For example, the probability constraint set shown right before table 2 in the previous section can be written in the matrix form below:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} po \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \\ p6 \\ p7 \\ s0 \\ s1 \\ s2 \end{bmatrix} = \begin{bmatrix} 0.005 \\ 0.002 \\ 0.003 \\ 0 \\ 0.99 \\ 1 \end{bmatrix} \Leftrightarrow A \cdot \underline{x} = \underline{b}$$

where *pi ≥0, i = 0 .. 7,* the binary representation of the index *i* in *pi* indicates the instantiation of R1, R2, R3, and *sj (*for *j = 0 .. 2*) are non-negative slack variables. For example, *p6* corresponds to the probability term *Pr(R1:1 R2:1 R3:0)* because $6_{10} = 110_2$.

The technique for solving the under-determined algebraic system is based on singular value decomposition (SVD), and when necessary, the method developed by Kuenzi, Tzschach and Zehnder [25] for obtaining an initial feasible solution. To illustrate the key idea behind the singular value decomposition and the method by Kuenzi, Tzschach and Zehnder, the matrix form of an algebraic system is re-written as below:

$$A \cdot \underline{x} = \underline{b} \Leftrightarrow [a_{i,j}]_{i=1..m,\,j=1..n} \cdot \underline{x} = \underline{b}$$

Using the example in the previous section, *m=6* and *n=11*. $\underline{x}$ is a *11*x*1* vector and $\underline{b}$ is a *6*x*1* vector. The *(i,j)* entry of the matrix *A*, represented by $a_{i,j}$, is -1, 0, or 1 in the example. The objective of the singular value decomposition is to decompose the matrix *A* into three matrices, *U, D*, and *V*, such that $UDV^T=A$; whereas *U* and *V* are orthogonal matrices and *D* is a diagonal matrix. A matrix *M* is orthogonal if $M^TM = MM^T = I$ (identity matrix). Since *U* and *V* are orthogonal,

$$[u_{k,j}]_{k=1..n,\,j=1..m} \cdot [u_{k',j}]_{k'=1..m,\,j=1..n} = I$$

$$\Leftrightarrow \left[\sum_{j=1}^{m} u_{j,k'} \cdot u_{j,k} \right]_{k'=1..n,k=1..n} = I \Leftrightarrow \sum_{j=1}^{m} u_{j,k'} \cdot u_{j,k} = \begin{cases} 0 & when\ k' \neq k \\ 1 & when\ k' = k \end{cases}$$

$$[v_{k',j}]_{k'=1..n,j=1..n} \cdot [v_{k,j}]_{k=1..n,j=1..n} = I$$

$$\Leftrightarrow \left[\sum_{j=1}^{n} v_{k',j} \cdot v_{k,j} \right]_{k'=1..n,k=1..n} = I \Leftrightarrow \sum_{j=1}^{n} v_{k',j} \cdot v_{k,j} = \begin{cases} 0 & when\ k' \neq k \\ 1 & when\ k' = k \end{cases}$$

The mechanism behind the decomposition of A in SVD is to apply householder transformation [26] to achieve QR decomposition [26] on A, and iteratively decompose the R matrix of the QR decomposition using QR iteration. The householder transformation as applied to achieve QR decomposition guarantees the orthogonal property of the matrix Q. The goal of QR iteration is to iteratively decompose the R matrix from triangular matrix to bi-diagonal matrix, and then further from bi-diagonal matrix to the diagonal matrix D as desired in the SVD. The following steps summarize the QR iteration for achieving Singular Value Decomposition:

Step 0:

Initialize $A' = A$.

Step 1:

Apply QR decomposition on A'; i.e., $A' = Q_{Ui}R_i$.; where i initially equals 1 and represents the iteration index.

Step 2:

Apply QR decomposition on the transpose of R_i (if it is not already diagonal at $i > 1$); i.e., $(R_i)^T = Q_{Vi}R'_i$.

Step 3:

Determine whether R'^T_i is diagonal. If R'^T_i is diagonal, then the iteration is completed, otherwise let $A' = R'^T_i$ and go to step 1.

When R'^T_i becomes a diagonal matrix at the completion of the kth QR iteration, $A = (\Pi_i Q_{Ui})R'^T_i(\Pi_{i=k} Q_{Vi}^T) = (\Pi_i Q_{Ui})R'^T_i(\Pi_{i=1} Q_{Vi})^T$. Since each Q_{Ui} and Q_{Vi} is orthogonal, $\Pi_i Q_{Ui}$ and $\Pi_k Q_{Vi}$ are orthogonal. $\Pi_i Q_{Ui}$ can then be used to define the U matrix of the SVD. $\Pi_{i=k} Q_{Vi}$ can then be used to define the V matrix of the SVD. And R'^T_i is used to define the diagonal matrix D of the SVD. A complete detailed illustration of applying householder transformation and QR decomposition to realize the singular value decomposition of A could be found online elsewhere [27].

Upon the completion of the singular value decomposition, one can easily obtain the inverse of D by taking the reciprocal of the non-zero diagonal entry in D while zero diagonal entry remains zero. Once U, V, and D^{-1} are known, solving $A\underline{x} = \underline{b}$ is equivalent to solving $UDV^T\ \underline{x} = \underline{b}$. One can then easily derive $\underline{x} = V D^{-1} U^T\ \underline{b}$.

Recall that the probability constraint set A is often an under-determined algebraic system. Many solutions may exist in the model discovery process. We will now show an important property of SVD that permits efficient derivation of the multiple solutions.

Theorem 1: Let UDV^T be the singular value decomposition of A. If the i^{th} diagonal entry of the D matrix is zero, then the corresponding column vector V_i in the V matrix is a null vector with respect to A; i.e., $AV_i = 0$.

Proof:

For $A = \left[\sum_{k=1}^{m} u_{i,k} d_{k,k} v_{k,j} \right]_{i=1..m, j=1..n}$, let the $k'th$ diagonal entry of D is zero;

i.e., $d_{k'k'} = 0$. We need to prove $AV_{k'} = 0$; whereas $V_{k'} = \left[v_{k',i} \right]_{i=1..n}$ is a column vector.

$$AV_{k'} = \left[\sum_{k=1}^{m} u_{i,k} d_{k,k} v_{k,j} \right]_{i=1..m, j=1..n} \left[v_{k',i} \right]_{i=1..n}$$

$$= \left[\sum_{j=1}^{n} \left(\sum_{k=1}^{m} u_{i,k} d_{k,k} v_{k,j} \right) v_{k',j} \right]_{i=1..m}$$

$$= \left[\sum_{j=1}^{n} \sum_{k=1, k \neq k'}^{m} u_{i,k} d_{k,k} v_{k,j} v_{k',j} \right]_{i=1..m} + \left[\sum_{j=1}^{n} u_{i,k'} d_{k',k'} v_{k',j} v_{k',j} \right]_{i=1..m}$$

$$= \left[\sum_{k=1, k \neq k'}^{m} u_{i,k} d_{k,k} \left(\sum_{j=1}^{n} v_{k',j} v_{k,j} \right) \right]_{i=1..m} + \left[\sum_{j=1}^{n} u_{i,k'} d_{k',k'} v_{k',j}^{\;2} \right]_{i=1..m}$$

By orthogonal definition stated earlier, the first term on the right-hand side above is a zero vector because $\sum_{j=1}^{n} v_{k',j} \cdot v_{k,j} = 0$. The second term on the right-hand side above is also a zero vector because $d_{k'k'}$ is zero. Therefore, $AV_{k'}$ is a zero vector. This completes the proof. **Q.E.D.**

By the concept of eigenvector, if A is an under-determined algebraic system, some k' diagonal entry/entries would be zero; i.e., $\sum_{k'} AV_{k'} = 0 \Leftrightarrow A(\sum_{k'} c_{k'} V_{k'}) = 0$ for some arbitrary constant $c_{k'}$. Since $A\underline{x} = \underline{b}$, $A\underline{y} = \underline{b}$; where $\underline{y} = \underline{x} + \sum_{k'} c_{k'} v_{k'}$. By changing the constant $c_{k'}$ in the linear combination, we obtain different values of $\underline{y}$. The multiple probability model solutions can then be obtained from those solutions of $\underline{y}$ that are non-negative.

In applying the Singular Value Decomposition to discovering multiple probability models, all joint probability terms and the slack variables, if any, should be non-negative.

Slack variables allow the conversion of inequality constraints to equality constraints. In the worst case scenario, we will need to produce an initial probability model solution that satisfies the non-negativity requirement if the initial solution $\underline{x}$ derived from the SVD fails the non-negativity requirement. Once an initial solution satisfies the non-negativity requirement, the process of generating multiple probability model solutions based on the initial solution and the linear combination of the null vectors extracted from the column vectors in the matrix V of SVD can proceed as just described. In order to make sure that the probability model discovery process will not fail when a set of proper probability constraints is presented, the method by Kuenzi, Tzschach and Zehnder is used to generate an initial solution when the worst case scenario occurs. The detail of the mechanism behind the method by Kuenzi, Tzschach and Zehnder will not be discussed in this paper because it has been extensively discussed elsewhere [25]. Nonetheless, an example illustration is provided in appendix 2.

For complete details on applying Singular Value Decomposition and the method by Kuenzi, Tzschach, and Zehnder for generating multiple models, readers are referred to chapter 9 of [24] as well as [27]. Nonetheless, the algorithmic steps for the model discovery are summarized as below for the completeness of this paper:

Step 1:

Construct the algebraic system of equations defined by the probability constraint set in the form of $A\underline{x} = \underline{b}$; i.e., each constraint, with proper slack variable introduced when necessary, accounts for a row in matrix A.

Step 2:

Apply the Singular Value Decomposition (SVD) algorithm on the matrix A and obtain a feasible solution. If the solution of the SVD does not satisfy non-negativity requirement, apply the numerical method proposed by Kuenzi, Tzschach, and Zehnder to obtain a feasible solution that satisfies non-negativity requirement.

Step 3:

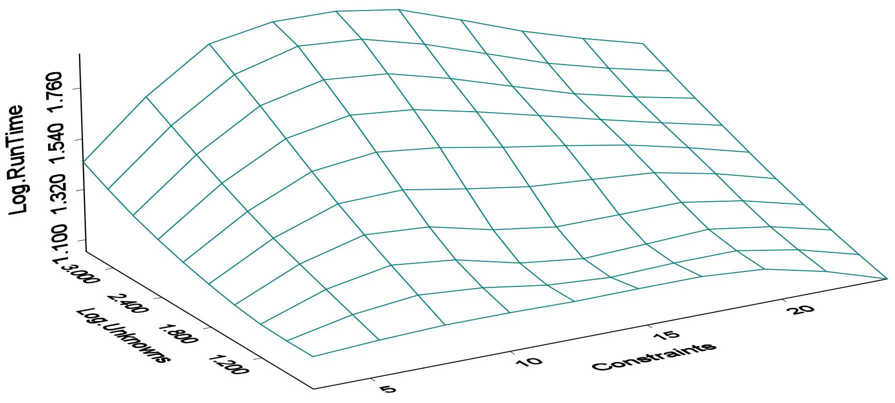
Identify column vectors $\{V_{k'} \mid k'=1,2...\}$ in the matrix V that correspond to the zero entries in the diagonal matrix D of the SVD of A; where the SVD of $A = UDV^T$, U and V are orthogonal matrices.

Step 4:

Obtain multiple alternative solutions $\underline{y}$ by constructing the linear combination of the initial solution $\underline{x}$ with the $V_{k'}$; i.e., $A\underline{y} = \underline{b}$ where $\underline{y} = \underline{x} + \Sigma_{k'}\, c_{k'}\, V_{k'}$ for some constants $c_{k'}$. Retain those solutions satisfying non-negativity requirement.

The pivoting mechanism of Kuenzi, Tzschach, and Zehnder approach applied in step 2 is quite common in linear programming (LP) solutions such as Simplex. However, our algorithm extends beyond solving linear optimization problems. One of the key unique features of our algorithm is a "speculative fast" search based on SVD. This is *not* part of the LP solutions. If in the worst case the SVD "speculative fast" search could not derive a solution satisfying the non-negativity constraints, then both our and LP solutions would have to revert back to the selection of a pivot element based on incremental improvement of a modified cost function. However, LP is restricted to searching only the vertices of the polytope (solution space), which *cannot* reach any optimal solution that does not reside at some vertex of the polytope. On the other hand, the SVD approach uses the linear combination of the null vector (step 4) to search through the inner space of the polytope using a linear path trajectory. Therefore, our approach overcomes the limitation of the traditional LP solutions, and is capable of solving a non-linear optimization problem where the best solution does not reside at a vertex.

The complexity of the SVD described above is in the order of *O(mnr)*; where *m* and *n* are the number of rows and columns in a *m*x*n* matrix, and *r* is the rank of the matrix (proportional to the number of nonzero column entries). However, the number of unknown joint probability terms for model discovery grows exponentially with respect to the number of observed triggering rules. Therefore, discovering joint probability models using the SVD is exponentially hard with respect to the number of joint probability unknowns. Although the nature of the problem is exponentially hard, the SVD based approach is well within the range for its application to the discovery of probability models of intrusion alerts. As we have experimented and reported elsewhere [24], the SVD, together with the method by Kuenzi, Tzschach, and Zehnder, have been successfully applied to discover probability models for solving many algebraic problems with an average size of 200 – 300 unknowns and a set of 12 – 24 (in)equality constraints. The most difficult problem being solved is one with 2187 unknowns and 10 constraints. When solving an algebraic problem of this size, the model discovery process can be completed almost instantly in a laptop with a 1.6GHZ CPU and 512MB RAM using standard error tolerance 10^{-6} and convergence iteration limit 30 as suggested elsewhere [28, 29]. Specifically, the average time taken for solving the algebraic problem is 14.71 seconds for problems with 16 unknowns or less, 17.99 seconds for that with 128 unknowns, 22.12 seconds for that with 256 unknowns, and 86.32 seconds for that with 2187 unknowns. The average time just reported includes the time taken for a human user to interact with the graphical user interface of the software, as well as the time taken for a network connection and the authentication/authorization process (for using our software). The plot below shows the relationship among the log of the run-time (in seconds), the log of the number of unknowns, and the number of constraints, for different sizes of the algebraic problems:



In addition to relying on only the SVD decomposition to handle the exponential nature of the problem from the aspect of algorithmic efficiency and computing resources, opportunities also exist to reduce the exponential problem to sub-exponential by considering the possible independency among the signature rules. Furthermore, we will explain in the latter section how the rule set partition in the AIDF could also improve the scalability of the model discovery by taking advantage on the rule set partition to be distributed across multiple IDS sites rather than at one aggregated IDS site. This is in addition to using domain knowledge (if desired) on the possible independence and the mutually exclusiveness of the rules for deriving set partition for the detection engine.

7. AIDF for Integrating Alert Information from Multiple Sensors

The Analytical Intrusion Detection Framework is introduced in section 5 for discovering probability model and for inferring hidden alerts that are not reported due to inhibiting negative. However, AIDF has a broader impact beyond just identifying inhibiting negative. We could apply the probabilistic approach of the AIDF for integrating information from different sensor sources, not just Snort, in a distributive Network based IDS (NIDS) environment.

Typically, it is desirable for the placement of only minimal NIDS sensors for the maximal coverage of a network topology. Let's assume there is a network environment of three subnets SN1, SN2, and SN3. The placement of a single IDS sensor (with a database of, say, 3000 rules) at a common network gateway/router may be sufficient to monitor cross subnet traffic activities. However, any local unicast traffic activities based on, for example, MAC address information stored in the ARP cache, could bypass the IDS sensor. On the other hand, host based IDS approach may not be effective ─ in terms of rule matching performance as well as the cost of implementation. It is because any host-to-host packet traversal in the same subnet will be examined at least twice. And if the packet traverses cross subnet, it may be examined once more if a NIDS sensor is placed at the layer 3 edge device (e.g., common gateway/router).

One alternative is to consider a distributive IDS environment. In a distributive IDS environment, a small number of sensors could be placed to collectively cover a local (subnet) environment. Each sensor is configured to handle a partial set of rules. Specifically, we would like to leverage on the information gathered from individual sensors for reconstructing the activities that may otherwise require a costly placement of extensive number of multiple IDS sensors. That is, we may opt to place n IDS sensors, as opposed to one sensor, in a subnet with m hosts; where $n << m$. For example, suppose m is 20 and n is 3, each IDS sensor is loaded with, say, 1200 (out of the 3000) rules with 300 overlapping rules between a designated pair of sensors. Any packet traversal broadcasting over the subnet will be received by all three IDS sensors, and altogether they cover the space of all 3000 rules. In addition, any unicast packet traversal will have 28% chance (assuming uniform distribution) reaching at least one of the three sensors.

However, there is one scenario the distributive IDS strategy may fail; i.e., a unicasting packet crafted for an attack targets at a victim sensor (referred to as the ignorant sensor) that has no matching signature rule(s). Yet the matching signature rule(s) exist(s) at another sensor (referred to as the partner sensor). A sensor is a "partner" of another sensor if overlapping rules exist between these two sensors. Our proposed approach provides a technique that could address the drawback of the scenario just mentioned. Specifically, the attack just mentioned could be identified through a probabilistic inference using the model encapsulating the coupling/association statistical characteristics of the ignorant and partner sensors as defined by the overlapping rules.

In general, the technique described in the AIDF provides a method for integrating information from different IDS sensor sources, and could be applied to realize the IDS sensor collaboration not just across local subnets, but wide area networks in which IDS sensors are equipped with different signature rules. Deploying IDS sensors equipped with different signature rules will be detailed later in the "experimental study" section.

8. Scalability and Distributive Coverage of AIDF

Although probability model discovery process is just an exercise on solving underdetermined algebraic system, the number of unknown joint probability terms grows exponentially with respect to the number of signature rules. Scalability is a complexity issue that needs to be addressed for practical purpose.

To address the scalability issue, we introduce a distributive coverage framework. Let R be a set of n signature rules {R1, …, Rn}; i.e., |R| = n. Let R' be a small subset of R (i.e., |R'| = m << n). With respect to R', let $G_{R'}$ = {$G1_{R'}$, …, $Gk_{R'}$} be k (> 1) subsets of R' satisfying the following conditions:

1. $\cup_{j=1}^{k} Gj_{R'} = R'$
2. For each Rr $\in$ R', Rr $\in \cap_j Gj_{R'}$ where each $Gj_{R'} \in G^{R',r}$ and $G^{R',r}$ is a subset of $G_{R'}$ with $|G^{R',r}| = s$ for some s > 1

In a typical real world scenario for Snort, n is in the order of 3000; i.e., n ≈ 3000. Suppose m = 8, k = 5, s =3, the subset R' consists of 8 signatures out of the set of 3000 some rules. R' typically is the set of signature rules reported in intrusion alerts of a short moving time window. In the set $G_{R'}$ there are 5 elements, and each is a subset of R'. For each rule Rr in R', it also appears as an element in three of the five subsets of R' in $G_{R'}$. Below shows an example of the formulation just described:

R' = {R1, … R8} $G_{R'}$ = {$G1_{R'}$, …, $G5_{R'}$}
$G1_{R'}$ = {R1, R2, R3, R6, R8} $G2_{R'}$ = {R2, R3, R4, R6, R8}
$G3_{R'}$ = {R3, R4, R5, R7, R8} $G4_{R'}$ = {R1, R4, R5, R7} $G5_{R'}$ = {R1, R2, R5, R6, R7}

R1 $\in G1_{R'} \cap G4_{R'} \cap G5_{R'}$ = {R1}   where $G^{R',1}$ = { $G1_{R'}$, $G4_{R'}$, $G5_{R'}$}
R2 $\in G1_{R'} \cap G2_{R'} \cap G5_{R'}$ = {R2, R6}   where $G^{R',2}$ = { $G1_{R'}$, $G2_{R'}$, $G5_{R'}$}
R3 $\in G1_{R'} \cap G2_{R'} \cap G3_{R'}$ = {R3, R8}   where $G^{R',3}$ = { $G1_{R'}$, $G2_{R'}$, $G3_{R'}$}
R4 $\in G2_{R'} \cap G3_{R'} \cap G4_{R'}$ = {R4}   where $G^{R',4}$ = { $G2_{R'}$, $G3_{R'}$, $G4_{R'}$}
R5 $\in G3_{R'} \cap G4_{R'} \cap G5_{R'}$ = {R5, R7}   where $G^{R',5}$ = { $G3_{R'}$, $G4_{R'}$, $G5_{R'}$}
R6 $\in G1_{R'} \cap G2_{R'} \cap G5_{R'}$ = {R2, R6}   where $G^{R',6}$ = { $G1_{R'}$, $G2_{R'}$, $G5_{R'}$}
R7 $\in G3_{R'} \cap G4_{R'} \cap G5_{R'}$ = {R5, R7}   where $G^{R',7}$ = { $G3_{R'}$, $G4_{R'}$, $G5_{R'}$}
R8 $\in G1_{R'} \cap G2_{R'} \cap G3_{R'}$ = {R3, R8}   where $G^{R',8}$ = { $G1_{R'}$, $G2_{R'}$, $G3_{R'}$}

As shown in the previous section, the exponential complexity for the probability model discovery depends on $|Gj_{R'}|$. Therefore, $|Gj_{R'}|$ should be kept small for scalability reason. In the above example, $|Gj_{R'}| = 5$ for j = 1...5 except 4, and $|G4_{R'}| = 4$. The size of $|Gj_{R'}|$ can remain small by increasing the size of $|G_{R'}|$ when the number of rules increases. In addition, it is desirable to keep the size of $|G^{R',r}|$ for different rules similar to each other so that the coverage for each rule has minimal bias.

Recall an intrusion alert reported by Snort is in form of a rule trigger. When a rule Rr is triggered, its corresponding cover set $G^{R',r}$ defines a number of $Gj_{R'}$ that each forms a basis for probability model discovery. In other words, probability model(s) can be derived for each $Gj_{R'} \in G^{R',r}$, and probabilistic inference can be conducted on each $Gj_{R'}$ to derive the most probable explanation $E[Gj_{R'}|Rr]$ based on $ArgMax[Pr(Gj_{R'}|Rr)]$. Within the distributive coverage framework, a forensic explanation is composed out of all the most probable explanations $E[Gj_{R'}|Rr]$.

9. An illustration on Distributive Coverage

We illustrate the concept of distributive coverage framework just discussed for forensic inference using one-week log data from one of our local subnets. Specifically, the distributive coverage framework is applied to scale the complexity for discovering multiple models for one IDS sensor. Discovering multiple models for multiple IDS sensors will be illustrated in the next section. In this illustration, a Snort sensor was installed in a local subnet 192.168.0.0/24. A common configuration was used for the Snort installation; i.e., the queue size of Snort is eight, while the reporting limit is two. The rule set of the experiment is the latest release prior to Nov 28, 2005 from the official Snort [30] and BleedingSnort [31] web sites. Altogether there are about 3000 rules with automatic daily update.

The testing period of the experiment is from Nov 28, 2005 to Dec 5, 2005. During the testing period, five rules shown below are triggered:

R1: Potential SSH Scan
R2: SCAN NMAP –f –sS
R3: Oversize Request –urid
R4: SCAN NMAP –sS
R5: Bare Byte Unicode.

The frequency occurrence of each rule is shown below with zero referring to no rule match while one referring to a rule match:

R1	R2	R3	R4	R5	Count
0	1	0	1	0	1
0	0	1	0	1	308
1	0	0	0	0	67
0	0	0	0	0	1557858

Table 3: Frequency count summary information

In the above table, the traffic volume is the sum of all the counts on the rightmost column. Referring to the distributive coverage framework formulation discussed in the previous section with m = 5, k = 6, s = 2 … 5.

$R' = \{R1, \ldots R5\}$ $G_{R'} = \{G1_{R'}, \ldots, G6_{R'}\}$

$G1_{R'} = \{R1, R2, R3\}$ $G2_{R'} = \{R1, R3, R4\}$ $G3_{R'} = \{R1, R3, R5\}$

$G4_{R'} = \{R1, R4, R5\}$ $G5_{R'} = \{R2, R3, R4\}$ $G6_{R'} = \{R3, R4, R5\}$

$R1 \in G1_{R'} \cap G2_{R'} \cap G3_{R'} \cap G4_{R'} = \{R1\}$ where $G^{R',1} = \{ G1_{R'}, G2_{R'}, G3_{R'}, G4_{R'}\}$

$R2 \in G1_{R'} \cap G5_{R'} = \{R2\}$ where $G^{R',2} = \{ G1_{R'}, G5_{R'}\}$

$R3 \in G1_{R'} \cap G2_{R'} \cap G3_{R'} \cap G5_{R'} \cap G6_{R'} = \{R3\}$ where $G^{R',3} = \{G1_{R'}, G2_{R'}, G3_{R'}, G5_{R'}, G6_{R'}\}$

$R4 \in G2_{R'} \cap G4_{R'} \cap G5_{R'} \cap G6_{R'} = \{R4\}$ where $G^{R',4} = \{ G2_{R'}, G4_{R'}, G5_{R'}, G6_{R'}\}$

$R5 \in G3_{R'} \cap G4_{R'} \cap G6_{R'} = \{R5\}$ where $G^{R',5} = \{ G3_{R'}, G4_{R'}, G6_{R'}\}$

Under the condition of the reporting limit = 2, the constraint set for the model $M(G1_{R'})$ covering the scope (R1, R2, R3) becomes:

$Pr(R1: 1) \geq 67/1558234=0.0000429974$
$Pr(R2: 1) \geq 1/(1+308+67+1557858) = 0.0000006418$
$Pr(R3: 1) \geq 308/(1+308+67+1557858) = 0.0001976$
$(67 + 309)/(1+308+67+1557858) = 376/1558234 = 0.0002413 \geq Pr(R1: 1)$
$(1 + 308)/1558234 = 0.0001983 \geq Pr(R2: 1)$
$(1 + 308)/1558234 = 0.0001983 \geq Pr(R3: 1)$
$Pr(R1: 1\ R2:0\ R3:0) = 67/(1+308+67+1557858) = 0.000043$
$Pr(R1:0\ R2:0\ R3:0) = 1557858/(1+308+67+1557858) = 0.9997587$

The upper bound for Pr(R1: 1) is arrived by observing 67 times occurrence of R1:1, plus the number of potential inhibiting negative ─ hereafter referred to as inhibiting zero ─ on R1 due to the patterns of at least two "1"s in the log data. These patterns are (R1:0 R2:1 R3:0 R4:1 R5:0) occurring once and (R1:0 R2:0 R3:1 R4:0 R5:1) occurring 308 times. Similarly, the upper bound for Pr(R2: 1) is due to the one-time occurrence of R2:1, plus the number of potential inhibiting zero resulted from the pattern (R1:0 R2:0 R3:1 R4:0 R5:1) occurring 308 times. And the upper bound for Pr(R3: 1) is due to 308 times occurrence of R3:1, plus the number of potential inhibiting zero resulted from the pattern (R1:0 R2:1 R3:0 R4:1 R5:0) occurring once. Note that the pattern (R1:1 R2:0 R3:0 R4:0 R5:0) cannot introduce inhibiting zero when the reporting limit is two. In general, the number of potential inhibiting zero for a rule variable Ri is the sum of the number of occurrences of all patterns that has Ri being 0, and there are at least h "1"s in each of these patterns ─ where h is the reporting limit. In the example, h is equal to two.

Using the model discovery technique described previously, the model optimizing the cost function based on maximum Shannon entropy measurement (to minimize bias) is shown below:

$Pr(R1:0\ R2:0\ R3:0) = 0.9997587$
$Pr(R1:0\ R2:1\ R3:0) = 0.0000007$
$Pr(R1:1\ R2:0\ R3:0) = 0.000043$
$Pr(R1:1\ R2:0\ R3:1) = 0.0001976$
All other probability terms being zero

Repeating the above process for $M(G2_{R'})$ through $M(G6_{R'})$, the following results are obtained:

Variable Instantiation	000	001	010	011	100	101	110	111
$M(G1_{R'})$:Pr(R1 R2 R3)	0.999759	0	7E-07	0	4.3E-05	1.98E-04	0	0
$M(G2_{R'})$:Pr(R1 R3 R4)	0.999759	6.42E-07	1.98E-04	0	0	0	4.30E-05	0
$M(G3_{R'})$:Pr(R1 R3 R5)	0.999759	4.30E-05	4.30E-05	1.12E-04	0	0	0	4.30E-05
$M(G4_{R'})$:Pr(R1 R4 R5)	0.999759	1.98E-04	0	0	0	4.24E-05	0	6.42E-07
$M(G5_{R'})$:Pr(R2 R3 R4)	0.999802	6.31E-07	1.97E-04	0	0	0	6.31E-07	1.07E-08
$M(G6_{R'})$:Pr(R3 R4 R5)	0.999802	6.56E-07	0	0	6.56E-07	1.96E-04	0	6.42E-07

Table 4: Probability models of different scope coverage

Suppose during the deployment the rules R1 (Potential SSH Scan) and R3 (Oversize Request –urid) are triggered, we are interested in conducting a forensic inference using the information in the models $M(G1_{R'})$ through $M(G6_{R'})$. In other words, we are interested in identifying event patterns that satisfy the following conditions:

When k =1 and 3,

(1) $\text{ArgMax}_{Rj\in M(GiR'),Rj\neq Rk}[\Pr(M(Gi_{R'}) \mid Se: Rk=1)]$ where i = 1..6

(2) $\Pr(arg \mid Se: Rk=1)/\Pr(arg) \geq 1$

where $arg = \text{ArgMax}_{Rj\in M(GiR'),Rj\neq Rk}[\Pr(M(Gi_{R'}) \mid Se: Rk=1)]$, i = 1..6

Condition (2) shown above basically imposes a requirement on the admissibility of the model to participate on contributing the best explanation only if the model covers the observable triggering rule(s). Event patterns that satisfy the conditions are shown in the following table:

Model	Se	Best Explanation satisfying(1) and(2)
$M(G1_{R'})$	R1:1 R3:1	ArgMax[Pr(R1,R2,R3|R1=1,R3=1)] = R2:0
$M(G2_{R'})$		ArgMax[Pr(R1,R3,R4|R1=1,R3=1)] = R4:0
$M(G3_{R'})$		ArgMax[Pr(R1,R3,R5|R1=1,R3=1)] = R5:1
$M(G4_{R'})$		ArgMax[Pr(R1,R4,R5|R1=1)] = R4:0 R5:1
$M(G5_{R'})$		ArgMax[Pr(R2,R3,R4|R3=1)] = R2:0 R4:0
$M(G6_{R'})$		ArgMax[Pr(R3,R4,R5|R3=1)] = R4:0 R5:1

Table 5: Best explanation results from probabilistic inference

When R1:1 and R3:1 are observed, the best explanations from all six models — ($M(G1_{R'})$ through $M(G6_{R'})$) — satisfy all conditions; thus contributing to new information for participating in the majority voting scheme. (R2:0) from $M(G1_{R'})$, (R4:0) from $M(G2_{R'})$, (R5:1) from $M(G3_{R'})$, (R4:0 R5:1) from $M(G4_{R'})$, (R2:0 R4:0) from $M(G5_{R'})$, and (R4:0 R5:1) from $M(G6_{R'})$ are six pieces of new information uncovered during the forensic inference process. Using these six pieces of new information to construct majority voting result, the following is obtained:

Observation source	Model source(s)	Event	# of votes
R1:1 R3:1	$M(G1_{R'})$, $M(G5_{R'})$	R2:0	2
R1:1 R3:1	$M(G3_{R'})$, $M(G4_{R'})$, $M(G6_{R'})$	R5:1	3
R1:1 R3:1	$M(G2_{R'})$, $M(G4_{R'})$, $M(G5_{R'})$, $M(G6_{R'})$	R4:0	4

Table 6: Voting result given R1:1

As mentioned previously, the queue size is two. Given the observation of R1 and R3 being triggered, we will seek for the event from the above table that has the largest number of votes with an instantiation value of 1. In this case, R5:1 is the winner. By observing R1 and R3 in the intrusion alert, the forensic inference will further suggest to investigate the activities that may also have triggered R5 (table 7).

Observation source	New event discovery	Forensic analysis result
R1:1 R3:1	R2:0 R4:0 R5:1	R1:1 R2:0 R3:1 R4:0 R5:1

Table 7: Result of forensic inference

10. Experimental Study

An experimental study was conducted to demonstrate the realization of AIDF in a multiple sensor environment and its ability on extracting forensic explanations based on the intrusion alerts of multiple sensors. The diagram in figure 1 depicts the set up for the experimental simulation.

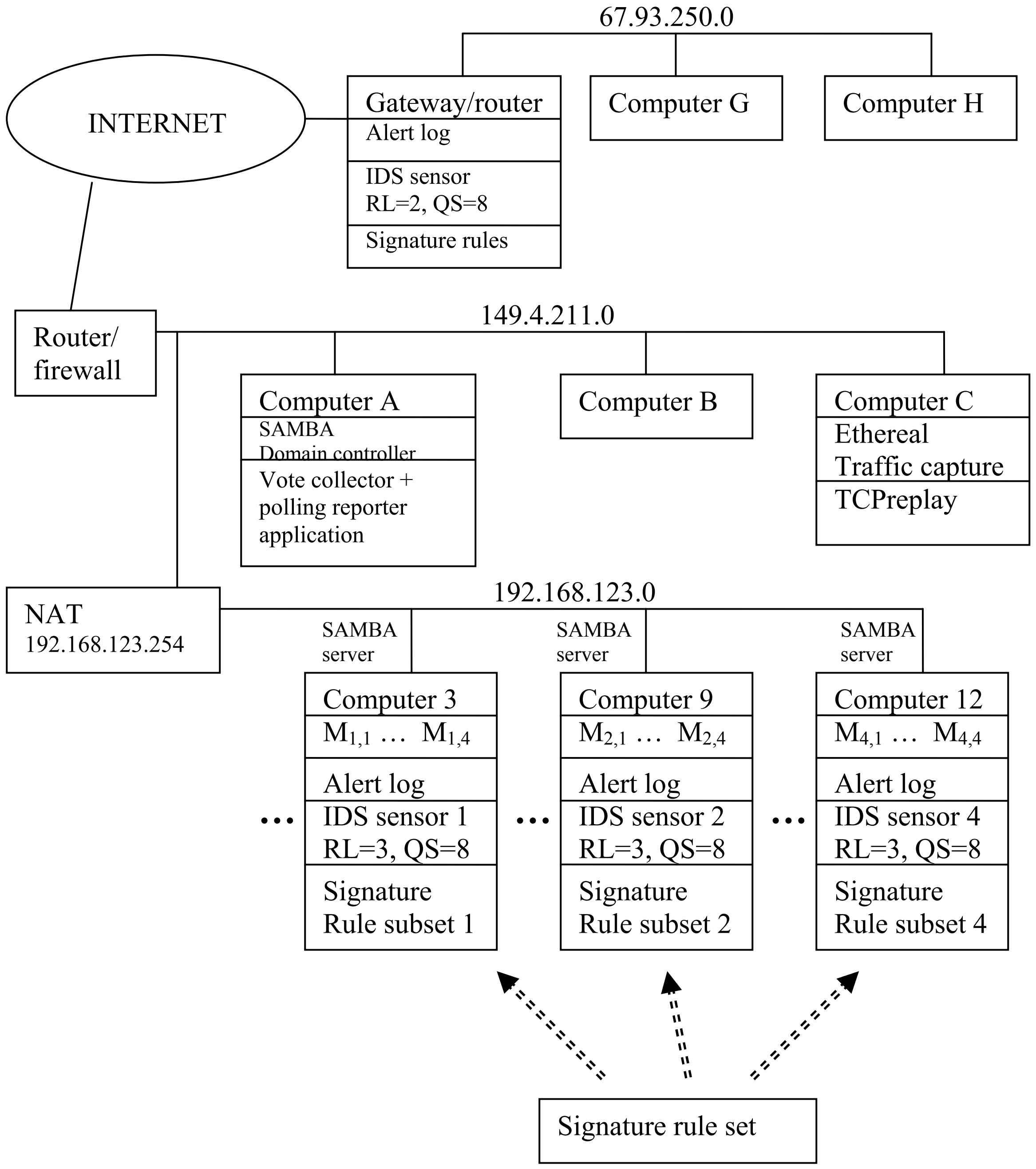


Figure 1: Network environment for the experimental simulation

As shown in figure 1, the IDS sensors are placed behind a simple NAT firewall environment. Ideally, this experiment should be conducted in the public network as opposed to the virtual private network. Due to the security policy restriction in our University environment, the experiment can only be conducted in a restricted virtual private network with simple NAT firewall enabled. As a consequence, should a DoS attack occur, the NAT firewall would have filtered the attack packets that target at a large number of closed ports. Therefore, a large number of signature rules that should have been triggered would not appear in an intrusion alert. In other words, only attack packets targeting at the open server ports could pass through to trigger a limited number of signature rules. Because of this, we refer the study as a "simulated experiment."

The environment of the simulated experiment consists of a dozen of networked computers behind a NAT router serving as a gateway (192.168.123.254) for the network 192.168.123.0. Among the dozen of computers, four are Snort IDS sensors for the network. At the time of the experiment, about 3000 signature rules were available. For each IDS sensor, a default setting of a queue size = 8 and a reporting limit = 3 were used.

In this simulated experiment, each one of the four IDS sensors is loaded with a database consisting of 1200 unique signature rules. 33% of the signature rules are overlapped between a pair of the IDS sensors. The duration of this experiment is one-month period between the end of Dec 2005 and the end of Jan 2006. 12 unique signatures were reported. The complete detail on the distribution of the signature rules appeared in the alert log as reported by each IDS sensor can be found in the online appendix table 1 [32]. Below is the list of the 12 unique signature rules appeared in the alert log:

1 BLEEDING-EDGE Potential SSH Scan
2 BLEEDING-EDGE VIRUS HTTP Challenge/Response Authentication
3 (http_inspect) OVERSIZE REQUEST-URI DIRECTORY
5 BLEEDING-EDGE EXPLOIT Awstats Remote Code Execution Attempt
6 BLEEDING-EDGE EXPLOIT XML-RPC for PHP Remote Code Injection
8 BLEEDING-EDGE Web Proxy GET Request
21 (portscan) TCP Portsweep
46 (http_inspect) BARE BYTE UNICODE ENCODING
47 BLEEDING-EDGE Exploit Suspected PHP Injection Attack
48 BLEEDING-EDGE EXPLOIT WEB PHP remote file include exploit attempt
49 BLEEDING-EDGE Proxy CONNECT Request
50 BLEEDING-EDGE THCIISLame IIS SSL Exploit Attempt

As noted, only 12 unique signature rules were triggered even the size of the entire signature database is about 3000. Given the relatively small size of the unique signature rules being triggered, it might be reasonable to consider data mining approach such as association rule discovery in form of Pr(X|Y) (where X is a possible set of "missing" rules and Y is a set of observable rules). However, while the approach based on mining the association rules is a logical extension of this approach, it typically also requires additional/stronger conditions. For example, association rule mining approach could rely on the same statistical information to determine the degree of association as measured by not just support measure (defined by joint event likelihood), but also additional criteria such as confidence measure, lift measure, leverage, or level of interestingness (as measured by mutual information measure) [24, 35, 36]. If one reduces association rule discovery to base on only support measure with a priori derived from simple frequency count, then our approach could be considered as a kind of association rule discovery. With stronger conditions as typically required in association rule discovery, the key to the successful application of association rule mining to tackle the same IDS problem will be on identifying additional/stronger conditions that would accept association rules leading to improving the false positive rate without worsening the false negative rate. In this research, we only rely on posterior Bayesian inference.

Furthermore, during the progress of this paper an issue re-occurring a number of times is the contention on the reasonableness of the dozen of intrusion alerts. Specifically, is it reasonable to expect only a dozen of alerts in the one-month period covered by this experimental study? Unfortunately the expected number of intrusion alerts is more complicated than just a function of time.

We assert that successful deployment of IDS requires an on-going tuning process. Initially an IDS is not uncommon to produce an excessive number of false positives or false negatives depending upon the choice of the rule set. Overtime the rule set should be adjusted to improve the false alarms; e.g., administratively prohibited ICMP alert is a “famous” false alarm among certain network infrastructure set up involving Cisco Pix device and DNS server. Until the IDS is tuned to minimize the false alarms, the IDS alerts may be too noisy, and may not be meaningful in regard to offering insights into the weakness of a current set up and how a malicious attack could penetrate through a security zone. In other words, the reasonableness of the number of intrusion alerts in relation to the expected number of intrusion alerts is a difficult question because of its time varying nature. Nonetheless, there are at least two important aspects to consider for answering this difficult question: (i) the current state of security zone refinement, and (ii) the environment/source from which it is based on for deriving the expected number of intrusion alerts. With respect to the first aspect, the number of observable alerts from any IDS set up that is still under refinement and entails an unacceptable false positive or false negative rate would not be appropriate for establishing a benchmark for comparison purpose. With respect to the second aspect, the number of intrusion alerts observed in one environment is not necessarily appropriate to be used as a reference for comparison if the inherent environment on the kind of server services (thus the opening ports) is not at least similar. As such, we consider our finding on the size of unique signature rule set over the one-month period of the experimental study as “best effort”.

Experiment procedure

The one-month intrusion alert data set was used to create seven training and testing data sets in the running periods. Each data set consists of one week training and one week testing, with half of a week overlapping between two successive data sets. In addition, the testing period is immediately right after the training period. The specific date intervals could be found in the online appendix table 2 [32].

In the experimental study, the training data set was the signature rules reported in the intrusion alerts by each sensor. These signature rules were used to define the scope of the probabilistic inference models to be built for each sensor. In general, if *n* signature rules were reported and the scope of a model is to focus on few rules, let's say *k*, there are *C(n,k)* numbers of possible models for each sensor. However, in this simulated experiment we focus on only the models that satisfy the following condition:

> On any given test period, a model for this simulated experiment must include at least one signature rule reported in some intrusion alert(s) as appeared in the training data set, and at least one signature rule reported in some intrusion alert(s) as appeared in the testing data set.

The reason for the above condition is that if a model does not include at least one signature rule reported in some intrusion alert(s) as appeared in the testing data set, the model is irrelevant in regard to gaining insights into answering a fundamental question related to the objective of this simulated experiment; namely, how effective is the AIDF framework and the voting mechanism described in the previous sections?

It is also important to point out that the scope of the rule coverage in the probability model discovery process is limited by the number of rules triggered during the training period. Even if each IDS sensor is loaded with thousands of rules, the size of the rule database of an IDS sensor only increases the likelihood of "catching" an attack. If during the test period only a dozen of rules triggered, the scope coverage of the probability model could only be at most a dozen of rules as opposed to the number of the rules in the database. It is because probability model discovery is only applied to cover the scope of the rules triggered. In this regard, one must be mindful that extracting forensic explanation from intrusion alerts is opportunistic.

For the purpose of this simulated experiment, four models were built for each IDS sensor using only the training data in a given test period. Each model (satisfying the conditions above) consists of three signature rules that have been reported. The distribution of the rule occurrence patterns that have at least one triggered rule is shown in the online appendix table 3 [32]. This distribution information is then used for formulating constraints for probability model discovery.

Probability constraints based on the frequency count information were formulated for the discovery of inference models of each IDS sensor of each test. An example of the formulation of the probability constraints is shown in the appendix 1. This example uses the data from the test set 3, IDS sensor 2, and covers the scope involving R1, R47, and R48 (shown in the online appendix table 4 [32]). As shown in this example, the probability constraints result in an under-determined algebraic system. The illustration shown in appendix 1 presents three of the multiple solutions of the under-determined algebraic system. Whenever multiple solutions may be available as a result of an under-determined algebraic system, the optimal solution is selected as the probability model for inferring the most probable forensic explanation, where the optimization criterion is the minimal bias as defined mathematically using Shannon entropy.

For comparison purpose, the same constraint set is also used to derive a naïve Bayes model under the assumption of independence. In other words, the model discovery process is applied to obtain the marginal probability information. The joint probability information is then derived from the product of the marginal probabilities. Naïve Bayes is supposed to offer computational advantage. Therefore, the basic idea is to approximate the joint probability model by a naïve Bayes model. This is to establish a benchmark, and to *speculate* whether the probabilistic inference using the naïve Bayes model could achieve a result similar to the joint probability model discovered in the AIDF.

During the test period, intrusion alerts being reported were used as the testing data for the evaluation purpose. Recall that the default reporting/log limit is three at the data collection phrase. Based on the default setting, there are two possibilities of the real world situation. First, three rules are reported in an alert. Second, less than three rules are reported in an alert. An alert will never report more than three rules because reporting/log limit is three.

When three rules are reported in an alert, there could be more than three rules being triggered but not reported. Therefore, when an alert reports three rules, one will not be able to tell in reality how many rules are triggered. If the exact number of triggered rules is not known in an alert, one could not use it to establish a "ground truth" for an experimental evaluation. As such, only a subset of the real world alerts qualifies as the "ground truth" for an experimental evaluation; i.e., alerts that report less than three rules.

The subset constituting the ground truth guarantees the observed real world alerts to have no hidden unreported rules. This subset of alerts is used to derive *test cases* by masking triggered rules away. Since an alert in the ground truth has less than three rules, it could have at most two triggered rules. In the alerts where two rules were reported, each rule was masked one-at-a-time to generate a test case. We do not mask both rules away at the same time because this is equivalent to have no alert (no rule gets triggered). To be precise, even though the real world alerts are captured using the default setting with a log limit = 3, only the subset constituting the ground truth is useful for the experimental evaluation, and the evaluation is (and can be) based only on a scenario where at most two rules are triggered, or an equivalence of a "simulated" log limit = 2.

During the test phase, the unmasked rule in the alert of a test case is used as an evidence/observation in the probabilistic inference. The outcome of the probabilistic inference is a "belief statement." This belief statement produced by each of the four models of each sensor was then used to construct the "votes" for generating a "forensic hypothesis." The forensic hypothesis of each sensor is then used to construct the "votes" for generating a forensic explanation. Ideally, the forensic explanation should contain no further claim other than the signature rule that has already been reported and that was masked. Otherwise, it results in a false positive in the forensic explanation. Similarly, if the forensic explanation fails to discover the masked rules, it results in a false negative.

Preliminary experimental result and evaluation metrics

In the one-month period, a total of about 1.3 million packets traversed through, and inspected by, Snort. Spanning over the testing periods of the experiment as shown in the online appendix table 4 [32], there were 47 test cases originated from the 12 unique signatures. The complete experimental results are shown in the online appendix tables 4 (for AIDF) and 5 (for naïve Bayes) [32]. In reference to the result of all the test cases, the information about the log report limit is not provided to probabilistic inference during the course of generating a forensic explanation. Table 8 shows a snapshot of the experimental result extracted from the online appendix table 4 [32] to facilitate a further discussion.

		T2							
Jan 7-13 (Train)	**Jan 14-20 (Test)**	**IDS1**	**Model belief**	**IDS2**	**Model belief**	**IDS3**	**Model belief**	**IDS4**	**Model belief**
8,6,5,(2,3),1, (47,48),49	(mask 3),2	M_{11}(2,3,5)	111	M_{21}(2,47,48)	P(se)= 0	M_{31}(3,5,49)	N/A	M_{41}(8,47,48)	N/A
		M_{12}(2,3,6)	111	M_{22}(1,2,48)	010	M_{32}(3,21,49)	N/A	M_{42}(6,47,48)	N/A
		M_{13}(2,5,6)	100	M_{23}(1,2,47)	010	M_{33}(5,49,50)	N/A	M_{43}(47,48,49)	N/A
		M_{14}(3,5,6)	N/A	M_{24}(1,47,48)	N/A	M_{34}(3,5,50)	N/A	M_{44}(6,8,47)	N/A
		Forensic hypothesis	111/ 100	**Forensic hypothesis**	010	**Forensic hypothesis**	N/A	**Forensic hypothesis**	N/A
Forensic explanation	R1:0 R2:1 R3:1 R47:0 R48:0								

Table 8: Snapshot of experimental result

The snapshot shown in table 8 is for a test case in T2 with the data in Jan 7 through 13 used for training purpose. During the training period, five unique rules and two pairs of rules (triggered simultaneously) were reported in the intrusion alerts. This test case occurred between Jan 14 and 20. In this test case R2 is provided as an evidence, whereas R3 is masked away to emulate the side effect of reporting/log limit. As depicted above, there are four IDS sensors (IDS1 – IDS4). Four models are derived for each of the sensors. For IDS sensor 1, the first model M11 covers the scope characterized by (R2, R3, R5). Likewise, the first model of the second and third IDS sensor ─ M21 and M31, covers the scope characterized by (R2, R47, R48) and (R3, R5, R49) respectively.

One important point to note is that the scalability of our proposed approach does not depend on the number of rules or the combination of (triggered) rules. Rather, the scalability of our proposed approach can always be made manageable through the rule set partitions of the AIDF discussed in section 9. Referring to table 8, there are 16 models. These 16 models collectively cover 11 rules (a combination of 2^{11}) while each model has only a representational complexity in the order of 2^3. This complexity can remain the same if we just increase the number of models to cover an expanded scope of increased number of rules.

When R2 is reported as a simulated alert, it is used by each model as an evidence for the probabilistic inference whenever possible. Note that the most probable explanation generated by the second model of the first IDS sensor M_{12} is (R2:1 R3:1 R6:1) ─ represented by the string "111" in table 8. Furthermore, the first model of second IDS sensor M_{21} and the third IDS sensor M_{31} will not be utilized for the probabilistic inference. The first model of the third IDS sensor M_{31} will not be utilized for the probabilistic inference because the evidence R2:1 is out of scope with respect to M_{31} ─ represented by "N/A." Note that a model could be out of scope in one test case but will be "in scope" for at least one test case in this simulated experiment because of the condition stated previously (under the description of experimental procedure). Also, despite M_{21} covers the scope consisting of R2, Pr(R2:1) is zero. Therefore, it is not possible to conduct probabilistic inference for deriving ArgMax[Pr(R47,R48|R2:1)]. Thus M_{21} and M_{31} are not utilized in conducting probabilistic inference in this case.

In the process of producing a forensic hypothesis produced by the IDS sensor 1, there are three counts of R2:1 and two counts of R3:1 while there is zero count for R2:0 and R3:0. In addition, there is equal number of counts for R5:0 and R5:1, as well as R6:0 and R6:1; thus neutralizing each other. As a result, the forensic hypothesis resulted from IDS sensor 1 will be (R2:1, R3:1). Likewise, the forensic hypothesis resulted from IDS sensor 2 will be (R1:0, R2:1, R47:0, R48:0). The forensic explanation derived from these two forensic hypotheses is then (R1:0, R2:1, R3:1, R47:0, R48:0).

In the snapshot shown in table 8 one model of IDS sensor 1, two models of IDS sensor 2, and all models of IDS sensors 3 and 4 are not utilized in probability inference given the case (R2:1). Ideally, it is desirable to utilize a model as often as possible. Therefore, one aspect of the evaluation would be on ***model utility*** as measured by the percentage of a model being utilized for probabilistic inference with respect to the number of test cases. In other words, it measures how often a model is relevant and used in a probabilistic inference, as well as participating in voting.

Furthermore, it is also desirable for a model to produce an "accurate" probabilistic inference result. The accuracy could be measured by conventional metrics using false positives and false negatives. However, since a model may not participate in probabilistic inference due to either out of scope or Pr(Se)=0, two additional metrics are introduced when evaluating the probabilistic inference accuracy of a model; namely, neglected negative and false silent. Neglected negative refers to the scenario where a rule is not triggered but is not considered due to out of scope. An example is R49 that appears in the training data, but not triggered in the particular case shown in table 8, and is not considered by M_{11} due to out of scope. False silent refers to the scenario where a rule is triggered but is not considered due to out of scope. Generally speaking, we could consider neglected negative and false silent as the scenario where a model remains neutral by not committing to a statement about whether a rule is triggered or not. Note that when model utility is low, we could expect a relatively high percentage of neglected negative and/or false silent.

In addition, we should be aware that the probabilistic inference does not have to be accurate in order to generate an accurate forensic explanation. The snapshot shown above illustrates one such case. Although models M_{11} and M_{12} did not generate the correct inference statement about R5 and R6, it was compensated by the inference statement of M_{13} (that allows us to cancel R5:0 with R5:1, and R6:0 with R6:1). In other words, the evaluation of accuracy should be conducted in both the model level as well as in the case level (note that there are 47 cases in this experiment).

In summary, the evaluation will be conducted in both model level and case level. In the model level, each model will be evaluated for its utility, and its accuracy as defined by false positive, false negative, false silent, and neglected negatives. In the case level, we will evaluate whether the forensic explanation for a test case is a perfect discovery, imperfect discovery, or an unsuccessful discovery. A perfect discovery is the case where the masked element (rule) is successfully uncovered with no incorrect conclusion on the other rules. Imperfect discovery is the case where the masked element (rule) is successfully uncovered but it also includes incorrect conclusion about other rules. Unsuccessful discovery is the case where it fails to uncover the masked element.

For comparison purpose, naïve Bayes reasoning was also conducted so that it could serve as a benchmark reference. Since the scope coverage of the naïve Bayes model is identical to that of the model discovery, the model utility percentage remains the same in both cases. (It is further noted that out of scope cases due to Pr(Se)=0 happen to be the same.) In regard to the inference accuracy and case level evaluation, the results of naïve Bayes reasoning were tabulated side-by-side with that using the joint probability model discovery method described in section 6. This allows us to gain insights into the validity and the effectiveness of the probability model discovery process for AIDF.

In additional to the evaluation criteria just described, the experimental results are also used to derive two pieces of summary information for better understanding the potential benefits of the AIDF as related to the integration of the intrusion alerts from multiple IDS sensors:

1. What is the percentage of perfect discovery that is attributed to combining alert information from multiple IDS sensors?
2. What is the percentage of cases where the scope coverage of the forensic explanations is improved upon by combining information from multiple IDS sensors rather than relying on the information from a single IDS sensor?

Summary evaluation result

1. Model level evaluation:

Criteria — model utility, inference accuracy

1a. Model utility percentage

	IDS1	IDS2	IDS3	IDS4
M1	49%	38%	40%	19%
M2	49%	36%	32%	23%
M3	45%	30%	19%	19%
M4	45%	28%	30%	21%

Table 9: Model utility percentage

1b. Inference accuracy

	Inferred: 0	Inferred: 1	Inferred: Neutral
Actual: 0	True negative: 42.6% (NB 47.46%)	False positive: 1.8% (NB: 0.2%)	Neglected negative: 55.6% (NB: 52.34%)
Actual: 1	False negative: 7% (NB: 30.14%)	True positive: 87.4% (NB: 65.75%)	False silence: 5.6% (NB: 4.11%)

Table 10: Inference accuracy

2. Case level evaluation:

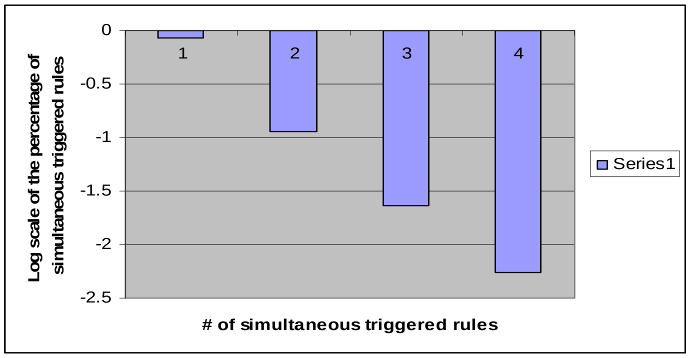
Criterion: Discovery success rate

Total # of cases: 47	Prob model discovery	Naïve Bayes
Perfect discovery	40%	4%
Imperfect discovery	28%	12%
Unsuccessful discovery	32%	84%

Table 11: Discovery success rate

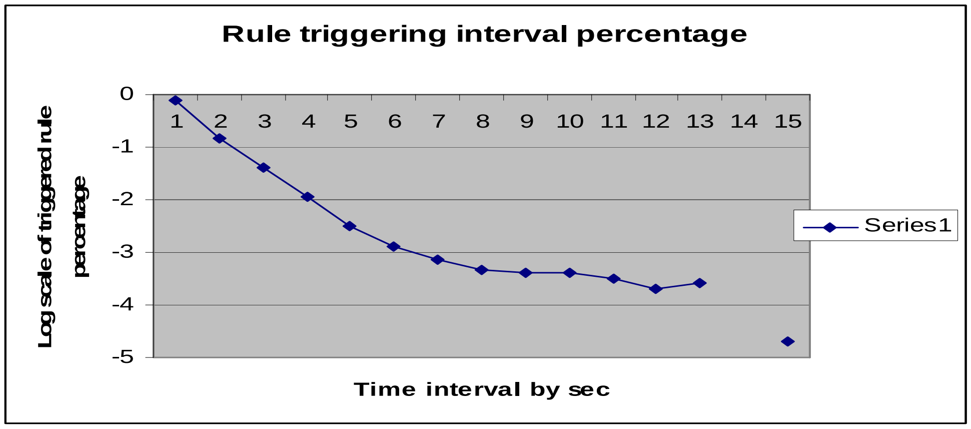
3. Significance of AIDF as related to information integration for a forensic inference:
    a. Percentage of perfect discovery that is attributed to combining alert information from multiple IDS sensors: 16.67%
    b. Percentage of cases where the scope coverage of the forensic explanations is improved upon by combining information from multiple IDS sensors rather than relying on the information from a single IDS sensor: 87%

Note that the results just shown are based on masking off one or another alert for instances of multiple alert triggers because of the need to establish a ground truth for the experimental validation with report limit = 2. In retrospect, the experimental validation could be improved upon to better reflect the ground truth of the inhibiting negative caused by the real traffic data if we could afford to perform multiple replays on the traffic data. Specifically, the traffic data could be replayed multiple times by incrementally increasing the report limit up to the queue size, and then by repeating again for increased queue size. In doing so, the amount of effort required for the experimental study would increase in a polynomial order with respect to the level of improvement on the ground truth. Despite the additional burden, further experimental study was conducted to simulate the replay for a better understanding on the possible range of the number of "true inhibiting negative." This is achieved by setting the reporting limit identical to the size of the queue and we extrapolate the number of true inhibiting negative based on the time that the rules are triggered. The results are shown in the graph below:



The graph above shows a log-linear relationship between the number of rules triggered simultaneously as related to causing inhibiting negative and their relative percentage. While the graph exhibits a log-linear relationship, readers are cautioned for not rushing into a conclusion on power law distribution. In this study we do not have sufficient data to conduct a statistical hypothesis test to validate/invalidate whether the distribution of the percentage of simultaneous triggered rules follows power law property with an underlying binomial or Poisson distribution.

Yet another aspect that we consider is the pre-condition for the occurrence of inhibiting negative; i.e., inhibiting negative could only occur if packets or packet streams could trigger multiple rules simultaneously. In other words, packet streams that will not trigger multiple rules can not cause inhibitive negative. Consequently the existence of packet streams that could trigger multiple rules over some pre-defined unit time may be an indication of increased likelihood of the occurrence of inhibiting negative. To gain further insight into the likelihood of the occurrence of inhibiting negative, we also analyzed the number of rules triggered by the (attack) traffic over short term windows (i.e., time intervals in seconds). An analysis of the traffic data accumulated from our network shows that the number of unique rules triggered over different sizes of short term window indicates some piece-wise log-linear relationship:



<u>Discussion</u>

The model utility evaluation result shows that the utility of the models constructed for this experiment ranges from 19% to 49%. Recall that a model covers only a very small subset of all the available signature rules of an IDS sensor. The choice of signature rule set partition for an IDS sensor, and the selection of small subset of rules for modeling, have a direct impact on the model utility percentage. Unless historical network traffic bears a predictable relationship with the future network traffic for estimating the distribution of the triggered rule set, it remains as an open question as to the choice of signature rule set partition for an IDS sensor and the selection of small subset of rules for optimizing the model utility percentage.

In reference to the inference accuracy, the results are shown in table 10. The performance of the naïve Bayes is shown in the bracket labeled as “NB.” It is noted that the proposed method significantly outperforms naïve Bayes in terms of true positive and false negative categories. The two methods perform relatively similar to each other in the other categories. While the proposed approach achieves an encouraging high accuracy on the true positive, it is also noted that the choice of the rule coverage by a model and the number of models chosen for each IDS sensor tends to be too conservative in lieu of a relatively high percentage of neglected negative ─ suggesting that the proposed approach tends to remain neutral and refrain from contributing towards the generation of a forensic explanation on rules that are not actually triggered. Although it is conceptually conceivable to improve the neglected negative and false silence rate by *twinkling* the model utility factor via different choices of scope coverage, the redistribution of the rules for the models, however, does not guarantee the outcomes to fall into the categories of true positive or true negative. Furthermore, it remains as a challenge in a real world deployment environment to know whether it is a true negative when a rule is not triggered, except the potential inhibiting negative in certain cases.

In reference to the case level evaluation, the discovery rate is very encouraging when the results of AIDF are compared against that of naïve Bayes. As discussed previously, one should expect that naïve Bayes would not perform as well because of the nature of the problem. Note that extracting forensic explanation from intrusion alerts takes advantage on the possible association among the rules that do not just co-occur independently. Yet the underlying independence assumption of the naïve Bayes will naturally ignore the useful co-occurrence information. This is confirmed by the experimental result that shows a significantly lower true positive rate. Although theoretically naïve Bayes has an advantage in terms of computational simplicity and efficiency, such an advantage is marginal given the size of the problem and the consideration of the model discovery complexity that involves inequality constraint.

Imperfect discovery rate, however, seems high even though it's still better than that of the naïve Bayes. It is because the imperfect discovery rate accounts for about 40% of the successful discovery of the hidden rule. There are two aspects that could be further investigated for improving the imperfect discovery rate. Note that imperfect discovery originates from the cases that include partially incorrect statement on the occurrence of the irrelevant rules in addition to the correct claim on the masked element (rule). There are three open questions that could be further investigated for improving the imperfect discovery rate. First, the success on generating forensic explanation depends on the ability of reasoning the accurate mathematical constraints for model discovery. How do we automate the process for generating the optimal constraint set? Second, alternative voting strategy other than the "equal-weight" strategy may help to cancel the effect of incorrect voting bias. What domain knowledge could be used to devise voting strategy that improves the imperfect and unsuccessful discovery in the process of generating a forensic explanation? Third, if operational knowledge (based on, for example, security policy and firewall rule implementation) on the maximum number of triggering conditions is known a priori, one could carefully select a queue size matching the maximum number of rules that could be triggered simultaneously, thus offering additional insights into improving inference accuracy and imperfect discovery rate. In other words, could one derive the relationship between the choice of queue size and a given rule set when operational knowledge is available?

In regard to the potential contribution of AIDF to integrating information from multiple IDS, and particularly in the area of forensic inference based on intrusion alerts, the results are encouraging. 16.67% of successful (perfect) discovery of hidden signature rules would not be possible without integrating intrusion alerts from multiple IDS sensors. Furthermore, in 87% of the test cases the scope coverage of the forensic explanations is improved upon by combining the information about the intrusion alerts produced by the multiple sensors.

In summary, although the experimental results demonstrate the success on the analytical intrusion detection framework and distributive intrusion detection, the following five factors bear interesting research open questions for further investigation:

Choice of signature rule set partition
Number of models per IDS
Scope of rule coverage for each model
Voting strategy
Queue size with respect to operational knowledge

Despite the open research questions just raised, perhaps the most valuable real world forensic inference that we have learned from the one-month experimental study is a common attack on our site targeting at the exploit of the weakness of PHP and Perl applications that provide web interaction and database access without performing input validation — a vulnerability that could be exploited for remote arbitrary code execution. This explanation has an equivalent semantic interpretation of the conjunctive combination of (R2:1 R3:1), (R5:1), (R6:1), (R5:1 R6:1) and (R47:1 R48:1). In particular, this explanation is derived based on the discovery of three consistent events shown in table 4 of the online appendix [28]: (a) signature rules 2 and 3 co-occurred throughout the one-month period, (b) rule 5 and rule 6 occurred separately and jointly in half of the one-month period, as well as co-occurred together with rules 2 and 3, and (c) rules 47 and 48 co-occurred after the co-occurrence of rules 2 and 3.

11. Conclusion

It is rare for an intrusion detection system to be operated in full logging mode for the purpose of forensic data gathering because of the practical consideration and implementation of anti-DoS strategy in a real world deployment. Consequently, there is a gap between the information useful for forensic and that revealed by intrusion detection. This paper presents a novel approach for generating the most probable forensic explanation from intrusion alerts and brings forth two specific contributions. First, a probabilistic inference approach is proposed for uncovering hidden information due to, for example, inhibiting negative — thus bridging intrusion detection with forensic analysis. Second, the proposed approach is generalized for an analytical intrusion detection framework for realizing distributive intrusion detection with a granularity at the level of rule partition, and a voting mechanism for integrating intrusion alerts from different IDS sensors distributed cross the same subnet or different subnets.

To demonstrate the relevancy of this research to the state-of-the-art intrusion detection technology, we investigated the underlying implementation of rule inspection engine in Snort. Current implementation of the rule inspection engine for intrusion detection in Snort reports only a partial list of all matching rules due to the consideration of the anti-DoS strategy and the need of real time performance. We show how our proposed model discovery and probabilistic inference mechanism could be applied to uncovering the hidden matching rules for generating forensic explanations from intrusion alerts. A preliminary experimental study is conducted to better understand its practicability and scalability. The results of this proposed approach is compared favorably against naïve Bayes approach. In our future research, we plan to further investigate the open questions raised from the five factors discovered in our experimental study. These open questions bear an impact on further improving the effectiveness and usefulness of the proposed framework for (i) realizing distributive IDS, and (ii) integrating intrusion alerts from different IDS sensor sources for the purpose of forensic inference and generating forensic explanation.

Acknowledgement: The author is indebted to the reviewers for many useful suggestions. This research is supported in part under the grants from PSC-CUNY Research and Queens College/CUNY Graduate Initiative Investment. Some initial findings of this paper appeared in the 2006 International Workshop on the Application of Data Mining to Security Control [33]. The author is grateful to Professor Giorgio Giacinto for his guidance on the emphasis of the paper suitable for this special issue, and the discussion and comments offered by Professors Soon Chun of the College of Staten Island, Douglas Salane of John Jay College, and Jun Zheng of Queens College.

12. References

[1] A. Valdes, K. Skinner, "Probabilistic Alert Correlation," LNCS, 2212, *Recent Advances in Intrusion Detection, (RAID 2001)*, Springer-Verlag.
[2] P. Stephenson, "The Application of Intrusion Detection Systems in a Forensic Environment," *RAID 2000*, Springer-Verlag Berlin Heidelberg.
[3] B. Schneier, J. Kelsey, "Secure Audit Logs to Support Computer Forensics," *ACM Transactions on Information and System Security*, V. 2, #2, May 1999, pp. 159-176. ISSN: 1094-9224.
[4] M. Roesch, "Snort – Lightweight Intrusion Detection for Networks," http://www.Snort.org/docs/lisapaper.txt
[5] T.H. Ptacek, T.N. Newsham, "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection," Jan 1988, http://www.Snort.org/docs/idspaper/
[6] http://www.sdsc.edu/~peisert/research/peisert-dissertation.pdf 2007
[7] P. Sommer, "Intrusion Detection System as Evidence," *Proc. of the First International Workshop on Recent Advances in Intrusion Detection (RAID),* 1998.
[8] S.T. King, P.M. Chen, "Backtracking Intrusions," *ACM Transactions on Computer Systems*, 23(1):51-76, Feb 2005.
[9] S. Peisert, M. Bishop. S. Karin, K. Marzullo, "Toward Models for Forensic Analysis," *Proc. of the 2nd International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp 3-15, 2007, ISBN: 0-7695-2808-2.
[10] A. Goel, W. Feng, D. Maier, J. Walpole, "Forensix: A Robust, High-Performance Reconstruction System, " *Proc. of the 25th International Conference on Distributed Computing Systems Workshops*, pp 155-162, 2005.

[11] B. K. Sy, H. Chan, "Data Mining Approach for Anomaly Detection: A Signature-based Approach," *Proc. of the 2005 International Conference on Data Mining*, Las Vegas, Nevada, June 20 – 23, 2005.
[12] D. Ellis, J. Aiken J. K. Attwood, S. Tenaglia, "A Behavioral Approach to Worm Detection," *Proc. of the ACM workshop on Rapid Malcode*, 2004.
[13] S. Axelsson, *Research in Intrusion-Detection Systems: A survey*, Technical Report 98-17, Dec, 1998.
[14] J. Doyle, I. Kohane, W. Long, H. Shrobe, P. Szolovits, "Event Recognition Beyond Signature and Anomaly," *Proc. of 2001 IEEE Workshop on Information Assurance and Security*, Wert Point, NY, June 2001, pp 17 -23.
[15] P. Roberts, Nai goes Forensic with Infinistream, *InforWorld*, Feb 2003, http://www.infoworld.com/article/03/02/10/HNnai_1.html
[16] Sanstorm Enterprise: Netintercept. http://www.sandstorm.com/products/netintercept/, Feb 2003.
[17] K. Shanmugasundaram, N. Memon, A. Savant, H. Bronnimann, "ForNet: A Distributed Forensics Network," *2nd International Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security*, LNCS 2776, Springer 2003, ISBN 3-540-40797-9.
[18] S. Wu, U. Manber, A Fast Algorithm for Multi-Pattern Searching, May 1994.
[19] M. Norton, D. Roelker, "Hi-Performance Multi-rule Inspection Engine," Sourcefire Network Security Inc., April 2004.
[20] S. Axelsson, "The Base-rate Fallacy and its Implications for the Difficulty of Intrusion Detection," *Proc. of the ACM Conference on Computer and Communication Security*, Nov 1999 (also http://www.raid-symposium.org/raid99/PAPERS/Axelsson.pdf)
[21] N. Tuck, T. Sherwood, B. Calder, G. Varghese, "Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection," *Proceedings of the IEEE Infocom Conference*, Hong Kong, China, March 2004.
[22] http://www.endace.com/
[23] http://www.nss.co.uk/grouptests/gigabitids/edition3/sourcefire/sourcefire.htm
[24] B.K. Sy, A.Gupta, *Information-statistical Data Mining: Warehouse Integration with Examples of Oracle Basics*, ISBN 1-4020-7650-9, Kluwer Academic Publishing, 2003/2004.
[25] H.P. Kuenzi, H.G. Tzchach, C.A. Zehnder, *Numerical Methods of Mathematical Optimization*, New York, Academic Press, 1971.
[26] M.T. Health, *Scientific Computing: An Introduction Survey*, McGraw Hill, ISBN 0-07-027684-6, 1997.
[27] http://www.qcwireless.net/dids_paper/svd_mechanism_illustration.pdf
[28] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition, Cambridge University Press, 1992, ISBN 0-521-43108-5.
[29] B.K. Sy, "Probability Model Selection Using Information-Theoretic Optimization Criterion," *The Journal of Statistical Computation and Simulation, Gordon and Breach Publisher*, Vol 69(3), p 203-224, 2001. (Also U.S. Patent 6,961,685)
[30] http://www.Snort.org/rules/
[31] http://www.bleedingSnort.com/bleeding-all.rules
[32] http://www.qcwireless.net/dids_paper/online_appendix.pdf

[33] B. K. Sy, N. Mullodzhanov, "Extracting Forensic Explanation from Intrusion Alerts," *Proc. of 2006 International Conference on Data Mining (DMIN'06),* Las Vegas, Nevada, June 2006.
[34] H. Debar, M. Dacier, A. Wepsi, *A Revised Taxonomy for Intrusion-Detection Systems*, IBM Research Report, 1999.
[35] J. Han, M. Kamber, *Data Mining – Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.
[36] R. Duda, P. Hart, D. Stock, *Pattern Classification*, John Wiley & Sons, 2001.

13. Appendix

Appendix 1

Data source: Online appendix table 4 ─ Period 3, IDS2
Model scope coverage: (1 47 48), log/report limit = 2, data size for period 3 =152399
Data size of rule occurrence pattern (R1:0 R2:0 R21:0 R46:0 R47:0 R48:0) = 152234
Probability constraint formulation:

C1:(143+15)/152399 = 0.00103675 ≥ Pr(R1:1)
C2: Pr(R1:1) ≥ 143/152399 = 0.000938
C3: Pr(R47:1, R48:1) = 15/152399 = 0.000098
C4: Pr(R1:0 R47:1 R48:0) = 6/152399 = 0.000039
C5: Pr(R1:0 R47:0 R48:0) =(1+152234)/152399 = 0.99892
C6: $\sum_{R1,R47,R48}$ Pr(R1 R47 R48) = 1

Denote Pr(R1:0 R47:0 R48:0) by P0, Pr(R1:0 R47:0 R48:1) by P1, ... , Pr(R1:1 R47:1 R48:1) by P7, the constraints C1 – C5 could be reformulated as below:

$$\begin{array}{llr}
C1: & P4 + P5 + P6 + P7 + S1 & = 0.00103675 \\
C2: & P4 + P5 + P6 + P7 - S2 & = 0.000938 \\
C3: & P3 + P7 & = 0.000098 \\
C4: & P2 & = 0.000039 \\
C5: & P0 & = 0.99892 \\
C6: & P0 + P1 + P2 + P3 + P4 + P5 + P6 + P7 & = 1
\end{array}$$

Below shows three model solutions by solving the above under-determined algebraic system using SVD decomposition:

R1	R47	R48	M1: Pr(R1 R47 R48)	M2: Pr(R1 R47 R48)	(Optimal) M3: Pr(R1 R47 R48)
0	0	0	0.99892	0.99892	0.99892
0	0	1	1.03E-04	1.3E-05	1.3E-05
0	1	0	3.9E-05	3.9E-05	3.9E-05
0	1	1	0	0	2.00E-05
1	0	0	8.40E-04	8.40E-04	8.40E-04
1	0	1	0	0.00002	0.00007
1	1	0	0	0.00007	0.00002
1	1	1	9.80E-05	9.80E-05	7.80E-05

Appendix 2

In this appendix the example shown in section 6 will be used to illustrate the application of the Kuenzi, Tzschach, and Zehnder (KTZ) approach to derive an initial feasible solution. The basic idea behind the KTZ approach for solving an algebraic system of linear constraints is to reformulate the constraint set by introducing additional variables — one for each constraint. Using the example in section 5 (and the algebraic formulation shown in section 6),

$Z_0 = 0.005 - P_4 - P_5 - P_6 - P_7 + S_0$
$Z_1 = 0.002 - P_2 - P_3 - P_6 - P_7 + S_1$
$Z_2 = 0.003 - P_1 - P_3 - P_5 - P_7 + S_2$
$Z_3 = - P_6 - P_7$
$Z_4 = 0.99 - P_0$
$Z_5 = 1 - P_0 - P_1 - P_2 - P_3 - P_4 - P_5 - P_6 - P_7$
$Z_i \geq 0 \quad i = 0 .. 5$

The above (in)equalities can be thought of as a constraint set of an optimization problem with a cost function $Min[Z_0+Z_1+Z_2+Z_3+Z_4+Z_5]$. Note that a feasible solution of this new optimization problem is a vector of 17 parameters $[Z_0\ Z_1\ Z_2\ Z_3\ Z_4\ Z_5\ P_0\ P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7\ S_0\ S_1\ S_2]$. If the global minimum can be achieved in this new optimization problem, this is equivalent to $Z_0 = Z_1 = \ldots = Z_5 = 0$, which in turn gives a feasible solution satisfying the non-negativity requirement for the original problem. That is, P_is in the global optimal solution $[0\ 0\ 0\ 0\ 0\ 0\ P_0\ P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7\ S_0\ S_1\ S_2]$ is a non-negative feasible solution to the original model discovery problem.

To illustrate the mechanism behind the method by Kuenzi, Tzschach, and Zehnder, the example shown above is reformulated in tableau form below:

Minimize $Z = Z_0+Z_1+Z_2+Z_3+Z_4+Z_5 = \sum_i Z_i$
$\leftrightarrow$ Minimize $Z = 2 - 2P_0 - 2P_1 - 2P_2 - 3P_3 - 2P_4 - 3P_5 - 4P_6 - 5P_7 + S_0 + S_1 + S_2$
$\leftrightarrow$ Maximize $-Z = Z' = -2 + 2P_0 + 2P_1 + 2P_2 + 3P_3 + 2P_4 + 3P_5 + 4P_6 + 5P_7 - S_0 - S_1 - S_2$

(Row 1) Maximize $Z' = -2 + 2P_0 + 2P_1 + 2P_2 + 3P_3 + 2P_4 + 3P_5 + 4P_6 + 5P_7 - S_0 - S_1 - S_2$
(Row 2) $Z_0 = 0.005 - P_4 - P_5 - P_6 - P_7 + S_0$
(Row 3) $Z_1 = 0.002 - P_2 - P_3 - P_6 - P_7 + S_1$
(Row 4) $Z_2 = 0.003 - P_1 - P_3 - P_5 - P_7 + S_2$
(Row 5) $Z_3 = 0 - P_6 - P_7$
(Row 6) $Z_4 = 0.99 - P_0$
(Row 7) $Z_5 = 1 - P_0 - P_1 - P_2 - P_3 - P_4 - P_5 - P_6 - P_7$

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**
		Constant	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	S_0	S_1	S_2
R1	**Z'**	-2	2	2	2	3	2	3	4	5	-1	-1	-1
R2	**Z0**	0.005					-1	-1	-1	-1	1		
R3	**Z1**	0.002			-1	-1			-1	-1		1	
R4	**Z2**	0.003		-1		-1		-1		-1			1
R5	**Z3**	0							-1	-1			
R6	**Z4**	0.99	-1										
R7	**Z5**	1	-1	-1	-1	-1	-1	-1	-1	-1			

Examine Z' or row R1 to determine the entry/entries below a positive Z'-row that is/are negative. In this case, we have (R2 C7), (R2 C8), (R2 C9), (R2 C10), (R3 C5), (R3 C6), (R3 C9), (R3 R10), (R4 C4), (R4 C6), (R4 C8), (R4 C10), (R5 C9), (R5 C10), (R6 C3), (R7 C3) through (R7 C10).

Note that altogether there are 14 variables plus three slack variable (P_0, ... P_7, Z_0, ... Z_5, S_0, S_1, S_2) and six equations (rows 2 - 7) not including the cost function (row 1). It is always possible to set the variables appearing at the Z'-row (row 1) to be zero while solving the equations in row 2 – row 7 for the remaining variables. For example, in the tableau shown above, P_0, ... P_7, S_0, S_1, and S_2 appear at Z'-row (row 1). If $P_0 = ... = P_7 = S_0 = ... = S_2 = 0$, equations on rows 2-7 can be used to solve for Z_0, through, Z_5. In such a case, the value for Z_0, through Z_5 can readily be read off from the "constant" column (column 2). In other words, maximizing Z' on row1 can be reduced to merely focusing on increasing the "constant" at row1 as much as possible.

Therefore, the objective is to increase the "constant" at row1 as much as possible. Evidently the limiting increase is given by dividing the element in the right-hand column (referred to as the pivot element) into the element in the constant column (column 2) of the pivot element's row. A value that is small in magnitude is most restrictive. The increase in the objective function for this choice of pivot element is then that value multiplied by the Z'-row entry of that column. We repeat this procedure on all possible right-hand columns (columns 3 – 13) to find the most appropriate pivot element.

For example, if we choose (row3, column5) --- abbreviated by [R3 C5] --- as the pivot element, this pivot will allow P2 to be increased by 2/|-1| = 2 (i.e., [R1 C5]/[R3 C5]), which results in a net increase of the cost function (i.e., Z'-row) by an amount 0.002·2/|-1|=0.004 (i.e., [R3 C2]· [R1 C5]/[R3 C5]). The mechanism behind the method by Kuenzi, Tzschach and Zehnder is to always choose the pivot element that will lead to the smallest increase in the cost function among all the possible choices of the pivot elements. In this case, the pivot element is [R3 C5]. This is used to "swap in" Z_1 and to "swap out" P_2, resulting in the following tableau form:

	C1	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*C8*	*C9*	*C10*	*C11*	*C12*	*C13*
		Constant	**P0**	**P1**	**Z1**	**P3**	**P4**	**P5**	**P6**	**P7**	**S0**	**S1**	**S2**
R1	**Z'**	*-1.996*	*2*	*2*	*-2*	*1*	*2*	*3*	*2*	*3*	*-1*	*1*	*-1*
R2	**Z0**	*0.005*					*-1*	*-1*	*-1*	*-1*	*1*		
R3	**P2**	*0.002*			*-1*	*-1*			*-1*	*-1*		*1*	
R4	**Z2**	*0.003*		*-1*		*-1*		*-1*		*-1*			*1*
R5	**Z3**	*0*							*-1*	*-1*			
R6	**Z4**	*0.99*	*-1*										
R7	**Z5**	*0.998*	*-1*	*-1*	*1*		*-1*	*-1*				*-1*	

The pivot element swap process just described is repeated until there is no more positive in the Z' row (or row1) for columns C3 - C13, which signals no further increase for Z' in row1 is possible. The answer for the optimal solution can then be read from the constant column (column 2) of the final tableau.