Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 06-004

A Multi-Step Framework for Detecting Attack Scenarios

Mark Shaneck, Varun Chandola, Haiyang Liu, Changho Choi, Gyorgy Simon, Eric Eilertson, Yongdae Kim, Zhi-li Zhang, Jaideep Srivastava, and Vipin Kumar

February 21, 2006

A Multi-Step Framework for Detecting Attack Scenarios

Mark Shaneck, Varun Chandola, Haiyang Liu, Changho Choi, György Simon,
Eric Eilertson, Yongdae Kim, Zhi-Li Zhang, Jaideep Srivastava, Vipin Kumar

University of Minnesota - Twin Cities
{shaneck, chandola, hliu, choi, gsimon, eric, kyd, zhzhang, srivasta, kumar}@cs.umn.edu

Abstract. With growing dependence upon interconnected networks, defending these networks against intrusions is becoming increasingly important. In the case of attacks that are composed of multiple steps, detecting the entire attack scenario is of vital importance. In this paper, we propose an analysis framework that is able to detect these scenarios with little predefined information. The core of the system is the decomposition of the analysis into two steps: first detecting a few events in the attack with high confidence, and second, expanding from these events to determine the remainder of the events in the scenario. Our experiments show that we can accurately identify the majority of the steps contained within the attack scenario with relatively few false positives. Our framework can handle sophisticated attacks that are highly distributed, try to avoid standard pre-defined attack patterns, use cover traffic or "noisy" attacks to distract analysts and draw attention away from the true attack, and attempt to avoid detection by signature-based schemes through the use of novel exploits or mutation engines.

Keywords Intrusion Detection, Attack Scenarios, False Alarms, Missed Attacks

1 Introduction

As the threat of attacks by network intruders increases, it is important to correctly identify and detect these attacks. However, network attacks are frequently composed of multiple steps, and it is desirable to detect all of these steps together, as it 1) gives more confidence to the analyst that the detected attack is real, 2) enables the analyst to more fully determine the effects of the attack, and 3) enables the analyst to be better able to determine the appropriate action that needs to be taken. Traditional IDSs face a major problem in dealing with these multi-step attacks, in that they are designed to detect single events contained within the attack, and are unable to determine relationships between these events.

Many alert correlation techniques have been proposed to address this issue by determining higher level attack scenarios [4, 6, 24, 27, 34]. However, if the data that is being protected by the network is highly valuable, an attacker can spend more time, money, and effort to make his attacks more sophisticated in order to bypass the security measures and avoid detection. Attackers, then, may use techniques to prevent their attacks from being reconstructed, such as making their attacks *highly distributed*; *avoiding standard pre-defined attack patterns*; using *cover traffic or "noisy" attacks* to distract analysts and draw attention away from the true attack; and attempting to avoid

detection by signature-based schemes through the use of *novel attacks or mutation engines* [38]. In these more sophisticated attacks, many of these correlation techniques face certain difficulties. In the case of matching against attack models [4] or analysis of prerequisites/consequences [6, 24, 27, 34], attackers can (and often do) perform unexpected or novel attacks to confuse the analysis. In addition, the information for these schemes must be specified ahead of time, and thus the analyst must be careful to specify complete information and not miss any possible situation.

Furthermore, these correlation approaches, as well as traditional IDS techniques, suffer from a fundamental problem, in that they try to achieve both a low false positive rate and a low false negative rate simultaneously. These goals, however, are inherently conflicting. If the mechanism used is set to be too restrictive then there will be many false negatives, yet if the mechanism is set to be less restrictive, many false positives will be introduced. Also, if signature-based systems, such as Snort [32], are used with many rules, too much time will be spent processing each packet, resulting in a high rate of dropping packets [30]. If these dropped packets contain attacks, then they will be missed. While some of the approaches have techniques to deal with missed attack steps [4, 24, 27], they cannot handle the absence of many of the steps in the attack.

Contribution In this paper, we propose an analysis framework that addresses this tradeoff between false positives and negatives by decomposing the analysis into two steps. In the *first step*, the analysis is performed in a highly restrictive fashion, which selects events that have a **very low false positive rate**. In the *second step*, these events are expanded into a complete attack scenario by using a **less restrictive analysis**, with the condition that the events added are related somehow to the events detected in the first step. We describe how this framework is suitable for this problem as it addresses the tradeoff between false positives and false negatives. In addition, our framework is 1) flexible, as it allows the analyst to exercise control over the results of the analysis, 2) designed to be modular and extensible, and thus makes it easy to improve the individual components of the analysis and incorporate new sources of data. We also implemented and evaluated our framework on a dataset that contained several attack scenarios, and we were able to successfully detect the majority of the steps within those scenarios.

Organization The remainder of this paper is organized as follows. In Section 2 we describe our framework. In Section 3 we describe our implementation of this framework and show its experimental results in Section 4. Next we discuss whether the framework achieves the goals set forth in the design and discuss the limitations of our approach in Section 5. We then describe some areas of related research in Section 6 and draw some conclusions and outline directions for our future work in Section 7.

2 Framework Design

The goals for our analysis framework are as follows: First, the system should address the inherent tradeoff between false positives and false negatives. Second, the system should be able to detect the majority of the steps contained within an attack and make connections between these steps to form the attack scenario. For this we assume that at least one step in the attack is visible (if none of the attack steps are visible to any lower

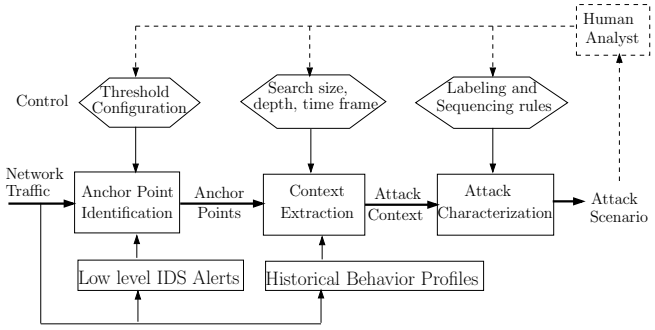

Fig. 1. The different phases of the analysis framework

level IDS, and thus the attack is perfectly stealthy, then we will be unable to detect the attack). Third, our analysis framework should provide high coverage of attacks (meaning that most or all of the attacks are detected). Finally, the system should be modular by design, thus making it simple to incrementally improve our approach.

The main challenge faced in designing this kind of system is balancing false positives and false negatives. To address this problem, our analysis framework is composed of two main steps. The first step, *Anchor Point Identification*, is focused on detecting a set of events (anchor points) in a very restrictive fashion, such that the set contains very few false positives. However, this will inevitably result in a large number of missed attack steps. To deal with this, the second step, *Context Extraction*, relaxes the restrictions conditionally; for a (potential) attack step to be examined in this step, it must meet the lower requirements as set by the detection mechanism, and it must also be connected in some way to an event captured in the first step. The overall framework is shown in Figure 1. Note that in Figure 1 there are three steps, where the third step, *Attack Characterization*, is concerned with giving semantic meaning to the steps in the overall attack scenario, as detected by the first two steps. This step is out of scope for this paper and thus it is not addressed in the description of our framework. In addition, the analysis scheme incorporates domain specific knowledge to further refine the results, which it does by keeping a human analyst in the loop. The analyst can control the output of Anchor Point Identification and Context Extraction by specifying the sensitivity of the tools which they utilize or applying domain knowledge in the rules that are used.

In addition, the analyst can control his view, in that he can specify the events that he is interested in seeing. For example, if the analyst is securing a specific machine that contains important data, he can set that machine to be the anchor point and search for relevant context that is related to that machine; or if the analyst knows about a certain activity that occurred on the network, or has a list of known bad hosts in a blacklist, he can specify the hosts involved in that activity.

2.1 Anchor Point Identification

The first phase of the multi-step analysis involves the identification of starting points (*anchor points*) for analysis. This is done by taking a set of low-level IDS alerts from

one or more (preferably independent) sources and selecting from this set a number of anchor points, such that we have high confidence that the set contains very few false positives. This can be done in many ways. One way is to use a single IDS configured to operate in a very restrictive manner, resulting in a high confidence yet incomplete set of attack events. Another way of doing this is through correlation techniques. It is well known that if an alert can be correlated with many other alerts, we can be more confident that this alert corresponds to a true positive [24]. Thus, in this manner, alerts from multiple sources can be combined together, where only the alerts which have high confidence are selected. However, there is a difference between the goal of this step and the goal of traditional alert correlation techniques. The difference is that we are not trying to balance false positives versus false negatives. Instead, Anchor Point Identification attempts to aggressively reduce false positives while maintaining high coverage of attack scenarios (where an attack scenario is considered ”covered” if at least one attack event in the scenario is selected in this step). The low false positive requirement is needed to ensure that the subsequent context extraction starts from a highly trusted base thus can focus on reducing false negatives. Because high attack coverage can accommodate high false negatives, this challenge is a relaxation of the more stringent requirement on traditional techniques that require low false positives and low false negatives simultaneously.

2.2 Context Extraction

The anchor points generated in the previous step are comprised of events in which there is high confidence that they are part of an attack. The *Context Extraction* step generates a suspicious context around these anchor points, both temporally and spatially. This step detects events related to the anchor points which are also anomalous or suspicious, but not enough so to be detected by the previous step. The goal of this phase is to add to the context only those activities that are part of the attack, thus filling in the attack steps missed by the previous step, while keeping the low false positive rate achieved by the Anchor Point Identification. This is done by relaxing the restrictions conditionally, i.e. “lowering the bar”, but only for those events that are connected somehow to an anchor point.

The major requirement for this step is some type of ranking for each network connection. One way this is accomplished is by an anomaly detection system. In this type of system, all connections are ranked according to how anomalous they are as compared to all other network connections, and this is typically done using data mining techniques. This can also be done by building historical behavior profiles for each host, determining which machines are servers and clients for particular services. When using historical behavior profiles, connections would be added to the context if they deviated from the historical behavior profiles for the hosts that they involved, for example if a web server started initiating connections, which it had never done before. This must be done carefully, however, for example in the case of peer-to-peer connections, which can be difficult to profile. If this type of traffic is not carefully profiled then the context can expand rapidly, effectively invalidating the result. One way to deal with this is to use peer-to-peer detection techniques [19] and ignore the peer-to-peer traffic when profiling.

This step also makes use of domain knowledge in the form of rules. Certain behavior patterns are known to be signs of malicious activity. For example, attackers often scan a network on a particular port to look for vulnerable machines. These scans most often result in failed connection attempts, as most machines will not have a service on that port. Thus, these machines will not respond (or will reject the connection attempt), and therefore are not vulnerable to being attacked on this port. This can be captured in a rule which states that all scans that do not result in a full connection (no successful reply from the scanned host) should be ignored, and all scans which do receive a successful response should be included.

3 Implementation Details

We implemented our framework to evaluate its effectiveness. Our framework could be instantiated in many ways, however we chose to implement it using simple components, in order to see how the framework performed even with simple components. As seen in Section 4, even with the simple components, our analysis framework successfully detected the attacks contained within the data on which we tested. These components, however, leave much room for improvement and, since the framework is designed to be modular, newer and more sophisticated techniques can be easily designed and inserted. In our implementation, we also utilized certain "primitives", such as low level IDS systems. The choice of these systems were driven by simplicity and practicality, and could easily be replaced by any other system that achieves the same goals.

3.1 Data Sources

Our framework requires certain data sources to be present in order to perform the analysis. We evaluated our framework on a specific data set (which is described in Section 4), and thus many of the choices for primitives were driven by this dataset. First, the network traffic was in tcpdump format, which we then converted into a netflow format [33]. Thus all the analysis we performed was done on aggregated network header information. Also, along with the traffic, Snort alerts were included. Thus, our implementation used these alerts as one low-level IDS. In addition, we also used our MINDS anomaly detector [11, 12] and MINDS scan detector [10]. Note that these choices were made based on practical reasons and could be replaced by other systems. For example, Snort could be replaced by any other signature-based system, such as ISS Real Secure [16]. Also, any scan detector could be used in place of the MINDS scan detector, such as TRW [18], and the following host/service profiler could be replaced by a more systematic host/service profiler such as the port pattern anomaly detector used in the EMERALD system [29, 35].

For the context extraction, we implemented a simple historical behavior profiler (e.g. host/service profiler), which examines the network traffic and determines which machines run which services, and which machines are clients for particular services. How it was used for context extraction is described in Section 3.3. It is based on the fact that machines typically exhibit the same behavior repeatedly. Thus, a web server might accept many connections on port 80 and 443, and rarely have any connection

requests on other ports or make outgoing connections on any ports. The profiler constructs a probability distribution of services which have been accessed on each host. The probability is calculated for each host by dividing the number of connections made to/from a particular port by the total number of connections to/from that host. If this probability is greater than a configurable threshold, then it is declared to be a server (or client, depending on the direction of the connections) on that port. In addition the profiler only profiles valid connections that have bidirectional flows (i.e. incoming flow and corresponding outgoing flow). This prevents the profiler from being skewed, for example by receiving scan packets on a port on which it does not offer any services. In our implementation, only internal hosts which have a degree of connectivity greater than some threshold (e.g. T_s for the server, T_c for client) are profiled. Once the profiles have been generated, each connection is examined and matched against the profile for the host involved. If it matches the profile (e.g. the connection in incoming on port 80 to a machine that has been profiled as a server on port 80), then the connection is assigned a score of 0, meaning normal. If the connection does not match any profile for the host, then it is assigned a score of 1, meaning anomalous.

3.2 Anchor Point Identification

The Anchor Point Identification step takes the output of multiple alert sensors and produces the set of events involved in attacks with higher confidence than relying on any single low-level IDS tool. In order for the alert combination to be effective, the data should be as orthogonal as possible, thus maximizing the overall information. In our implementation, we achieved this through the use of Snort alerts and the MINDS anomaly detector [11]. These two IDSs use vastly different mechanisms to flag traffic, and thus fit the requirement that the sources be orthogonal. We combined these two data sources in a simple manner, selecting Snort alerts to be anchor points if either the source or destination machine was also involved in a highly ranked anomaly. Note that the anomalous activity need not be the same connection that was flagged by Snort. The intuition behind this mechanism to combine the data is as follows. If a machine is truly attacked and compromised, it is likely that the attacker would use the machine in a way that it normally does not behave, causing anomalous traffic to/from this host. The threshold for determining if a flow is considered to be highly anomalous is configurable. Details on how sensitive this threshold was and how effective this technique was can be found in Section 4.

3.3 Context Extraction

The next step in the analysis process is the *Context Extraction* step. The main goal of this step is to add events from the set of all network traffic that are related to the attack(s) represented by the anchor points detected in the previous step. The main challenge faced by this step of the analysis is to properly refine the context so as to add the maximum number of attack steps to the context, while adding the minimum number of unrelated events. As noted in Section 2.2, we made use of two main techniques, rules drawn from domain expertise and host/service profiling. The rules used are as follows: First, we ignored all traffic that was flagged as a scan in which the scanned host did not reply

(i.e. did not successfully open a TCP connection). Conversely, we selected all scanning traffic that did result in a full bidirectional connection. Finally, each connection for which the previous rules did not apply was selected or ignored based on its host/service profiling score. If the score was above a configurable threshold, then the connection would be selected and added to the context, otherwise it would be ignored. Note that for a connection to be considered for addition to the context it must be related somehow to the anchor points. For our implementation we define this relation such that a connection is related to the anchor points if one of the IPs in the connection is already contained within the context, where the initial context is the set of anchor points.

Once we have a method to define which network events are to be selected for the context, the algorithm for context extraction is quite simple. The algorithm goes through a series of iterations. At the beginning of each iteration, there is a list of all the IPs contained within the context. During the iteration, each flow is processed. If one of the IPs involved in the flow in contained within the context already, and if the flow passes the specified rules (and the flow is not already in the context) then the flow is added to the context (and any IPs not already contained within the context will be added). The iterations continue until no more flows are added to the context (i.e. the transitive closure has been reached). The Context Extraction could also be limited to add only a set number of flows or distinct hosts to the context, however this could result in the loss of some of the attack. In addition, the threshold for the host/service profiling score can be dynamically adjusted, to require, for example, that connections added in later iterations (and thus more loosely connected to the original anchor points) have higher profile anomaly scores.

4 Experimental Evaluation

We evaluated our proposed framework using datasets generated by Skaion corporation [1]. These datasets are simulated to be statistically similar to the traffic found in Intelligence Community. This dataset has several scenarios with attacks injected that follow different patterns. In the following sections we first describe the nature of the Skaion dataset, then discuss methods we used to evaluate our framework, and finally we show our results. As can be seen in the following results, even though our approach currently uses only simple implementations for each component, our overall analysis captures the major attack steps successfully.

4.1 Skaion Dataset

As part of the ARDA P2INGS research project, the Skaion Corporation has released several sets of simulated network traffic data. This data includes various scenarios of multi-step sophisticated attacks on resources within a protected network. The scenarios for which they have generated data include single stage attacks (a simple scan or exploit or data exfiltration scenario), bank shot attacks (where an internal host is compromised and used to attack another internal host), and misdirection attacks (where a “noisy” attack is staged on one part of the network while the true attack takes place in a more stealthy manner in another part of the network). In addition to the main attack, there

are other background attacks (none of which are successful) and scans. To date, they have released 3 datasets to date, including many instances of these scenarios. However, for the sake of space, we will describe our results on one scenario in detail and present a summary of our results on other scenarios. The network topology in these scenarios is comprised of the following four domains: (i) the target protected domain, BPRD (Bureau of Paranormal Research and Defense) comprising of various servers which are the typical targets for attacks; (ii) a secondary internal domain which is not as protected as the protected domain and comprises of servers as well as clients. The hosts inside this domain have additional privileges to access the protected domain, BPRD; (iii) a set of external hosts which consists of attackers as well as normal users and (iv) a trusted domain which consists of remote users access the protected network with additional privileges over a dialup or a VPN connection. All traffic entering and leaving the entire internal network is captured by tcpdump. Snort alerts are collected for traffic exchanged between the external network and entire internal network.

Single Stage Attacks These scenarios are compromised of a simple attack made up of four steps. First, scanning is used to determine the IP addresses in the target network that are actually associated with live hosts. Typically in these scenarios, this is done by an attacker performing reverse DNS lookups to see which IPs have domain names associated with them. The next step consists of an attacker (or multiple attackers) probing these live hosts to determine certain properties, such as which OS and version is running on the host. Then one of these hosts is attacked (possibly by a host that was not involved in any previous steps) and compromised. Finally, a backdoor is opened, to which the attacker connects, and performs various malicious activities, such as data exfiltration or the downloading and installation of attack tools.

Bank-Shot Attacks These attacks are aimed at avoiding detection by using an "insider" host to launch the actual attack. In this scenario, initial scanning is done, and then an attack is launched against a host in the BPRD network. This attack fails, and the attacker then scans and compromises a host in the secondary internal domain. From this server, the attacker scans and launches attacks on hosts in the protected BPRD network. A host is then compromised, from which data is exfiltrated.

Misdirection Attacks The attacker attempts to draw the attention of the analyst away from the real attack. He does this by launching a noisy attack (one which sets off many IDS alerts) on a particular set of hosts in the protected network. Then using a previously compromised host in the trusted domain, he attacks and compromises another host in the BPRD network, from which he exfiltrates data.

4.2 Evaluation Methodology

Before discussing the results of our experiments, we first describe how we performed the experiments and the methods we used to evaluate our framework. For a given scenario, we first ran all low-level IDS tools to generate the alerts, anomaly scores, etc. For

profiling, we used ten and five connections for T_s and T_c respectively. This means that a host was profiled as a server only if it had more than 10 inbound connections. Similarly, a host was profiled as a client only if it had more than 5 outbound connections. In addition we only profiled ports with more than two connections. We then ran Anchor Point Identification using multiple rules for detecting the anchor points in order to compare the performance and sensitivity of each set of rules. First, we used Snort alone, where each Snort alert was selected as an anchor point. Next, we used the MINDS anomaly detector alone, where the connections that ranked in the top k% of anomalies were selected as anchor points. Finally, we combined Snort and MINDS in the method described in Section 3.2. The anchor points selected were those Snort alerts in which at least one of the IPs was involved in a highly ranked anomaly (ranked within the top k% of MINDS Anomaly Detector output). The evaluation criteria for the anchor points is twofold: first, whether it covers the attack (i.e. did it have any true positives), and second, whether it has low false positives (the lower the better). The Anchor Point Identification step generates a set of events (anchor points) which represents a connection between two hosts. An anchor point is related to the attack scenario if the connection it represents is a part of some attack step. In our results section, the results of this step are represented by the number of attack related hosts detected (true positives) and number of non-attack related hosts detected (false positives). A host is counted as attack related if it is present in an attack related anchor point (in this case we call it covered, as introduced in section 2). If a host is present only in non-attack related anchor points, it is counted as a false positive.

Following the Anchor Point Identification step, Context Extraction was run with each set of anchor points found by different rules utilized by Anchor Point Identification. No other parameters were varied for this step, since the parameters mainly consist of limiting the expansion, and for our experiments this step was run until no more context was added. The goal for this step is to detect all attack related steps (with emphasis on the more important steps, e.g. initial scanning is less important then exploits or backdoor accesses) while reducing the number of non-attack related steps. Note that there are two types of non-attack related hosts that could be added to the context. First, they could be part of background attacks, which are still interesting for the analyst. Second, there are real false positives, which are not a part of the actual attack scenario or the background attacks.

All the tables for the results follow the following notation :

AS:*Attack Steps* This represents the high level attack steps like probing (information gathering), actual exploit, backdoor access, or data exfiltration.

AH:*Attack-related Hosts* This includes all hosts related to the attack scenario including external scanners, external attackers, internal hosts scanned by the attackers for information and the eventual victims which get compromised.

BA:*Background Attack Related Hosts* This involves all hosts related to the background attacks in the traffic as attackers or victims.

FP : *False Positives* This counts all hosts that are not related to the actual attack scenario or to the background attacks but are wrongly detected by our framework.

4.3 Detailed Analysis: Skaion Scenario - 3s6

We present our detailed analysis on one of the bank shot attack scenarios. The scenario we evaluated (called 3s6) had 122,331 connections in the traffic, involving 4516 unique IPs, on which there were 6974 Snort alerts.

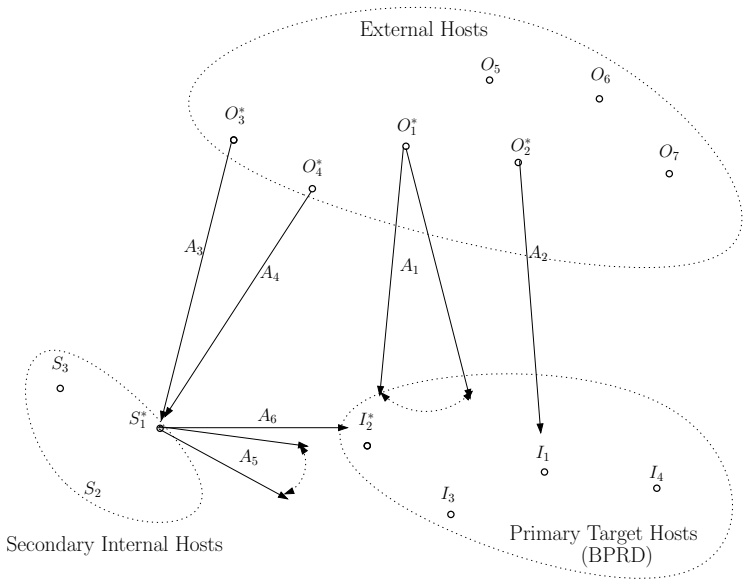

Fig. 2. Different steps and hosts involved in the attack scenario 3s6

The attack graph for the scenario 3s6 is shown in figure 2. The various steps involved (in chronological order) are :

- A_1 : O_1 (74.205.114.158) scans 92 hosts *(936 flows)* inside the BPRD network.
- A_2 : O_2 (42.152.69.166) attacks internal server, I_1 (100.10.20.4) four times (*17 flows*) and fails each time.
- A_3 : O_3 (168.225.9.78) port scans *(18 flows)* secondary internal host, S_1 (100.20.20.15 *alias* 100.20.1.3).
- A_4 : O_4 (91.13.103.83) attacks S_1 *(78 flows)* using *Apache OpenSSL SSLv2 Exploit* [3] and succeeds.
- A_5 : S_1 port scans 6 servers in the BPRD network *(895 flows)* including the eventual victim, I_2 (100.10.20.8).
- A_6 : S_1 launches attacks on I_2 using *IIS IDA-IDQ exploit* [2] and succeeds. It also browses through the files of I_2 *(4 flows)*.

The attackers try to confuse the analyst by first scanning and unsuccessfully attempting to attack the internal network (Steps A_1 and A_2). Most of the attack related Snort alerts are on this traffic. Another attacker then attacks the secondary network and compromises an internal host (S_1). This host is then used to scan the BPRD network and

Table 1. Results for anchor point identification on bank-shot scenario 3s6

Config		AH	FP
Snort		96	169
Top *k*%-anomalies	0.2	5	5
	0.5	8	67
	1.0	50	114
Snort + Top *k*%-anomalies	0.2	93	0
	0.5	95	39
	1.0	98	83

Table 2. Results for context extraction on bank-shot scenario 3s6

Config		#Iterations	AS	AH	BA	FP
Snort		2	5(A_1,A_2,A_4,A_5,A_6)	24	3	75
Top *k*%-anomalies	0.2	2	5(A_1,A_2,A_4,A_5,A_6)	24	3	43
	0.5	2	5(A_1,A_2,A_4,A_5,A_6)	24	3	58
	1.0	2	5(A_1,A_2,A_4,A_5,A_6)	24	3	93
Snort + Top *k*%-anomalies	0.2	2	5(A_1,A_2,A_4,A_5,A_6)	24	3	45
	0.5	2	5(A_1,A_2,A_4,A_5,A_6)	24	3	47
	1.0	2	5(A_1,A_2,A_4,A_5,A_6)	24	3	47

launches an attack on I_2. Since this traffic is internal, it is not detected by Snort. The results of context extraction in Table 2 show that the framework succeeds in capturing a large portion of the attack scenario (5 out of 6 attack steps). The context also captures some background attacks present in the traffic. The false alarms arise because of following reasons - 1) *Mislabeled Flows* - These arise because of errors in the data converting component due to which initiating flows might be labeled as replies and vice versa. 2) *False alarms from Our Profiler* - Host/service profiler has an associated false alarm rate due to which some non-attack related flows are added to the context.

All configurations for anchor points result in detecting a portion of the scanning activity by O_1 as anchor points in Table 1. From these anchor points, the scanning activity A_1 is added to the context. Since I_1 is scanned by O_1, its traffic is analyzed. This results in adding the failed attack attempts, A_2 to the context. I_2 is also scanned by O_1. Since I_2 is attacked by S_1, this attack step A_6, is added to the context. On analyzing the traffic to and from S_1, the scanning activity A_5 is added to the context. Similarly the attack step, A_4 on S_2 is also added to the context. The attack step A_3, is not captured since it involves probing of S_1 on ports on which it is a server. However, we capture all those attack steps from which we can construct the core attack scenario.

We observe from Table 1 that if we use a correlation of Anomaly Detector and Snort we get less number of false positives as anchor points. As we relax the constraints in Anchor Point Identification step, we detect more attack related hosts, but the number of false positives also increases. However, from the context extraction results in Table 2 we observe that we still detect the major portion of the attack scenario even if we start with a less number of anchor points. Moreover, the presence of false positives in anchor points results in a high false positive rate for context extraction.

4.4 Results for Other Scenarios

The results of our analysis on other scenarios are summarized in Table 3. The configuration used for Anchor Point Identification was the combination of Snort Alerts and top 0.5% of MINDS Anomaly Detector Output. From the table we observe that our implementation is able to capture all important steps of each attack scenario except for the scenario - *Five by Five* (In this case, the volume of traffic related to the victim host was not enough to be profiled, thereby that host was not added to the context). The attack steps which were missed in all cases involved failed attack attempts or probes before attacks. Our implementation captured all the important attack events, such as the actual exploit, data exfiltration for all but one scenario from which the core attack scenario can be generated. From the results we can observe that by using strict thresholds for Anchor Point Identification, we are able to detect some attack related events (as anchor points) while keeping the number of false positives very low. Using these anchor points, we successfully detect the core attack scenario in all but one scenario along with some background attack activity. Since the number of non attack related anchor points are low, the false positives in the context extraction step are also very few.

Table 3. Summary of results for different Skaion scenarios

	Scenario	Ground Truth					Anchor Points		Context Extraction			
		# Conn	# Hosts	# Alerts	**AS**	**AH**	AH	FP	**AS**	**AH**	BA	FP
Single Stage	Naive	1739	581	27	**4**	**10**	2	0	**4**	**3**	0	0
	Simple Ten	12040	2616	114	**4**	**246**	4	0	**4**	**6**	0	1
	Five by Five	7853	2101	177	**3**	**13**	5	45	**0**	**0**	0	5
	Ten by Ten	9459	1435	54	**4**	**16**	5	11	**4**	**5**	0	1
	s9	4833	472	53	**3**	**2**	2	3	**3**	**2**	0	0
	s10	4792	582	58	**4**	**3**	2	6	**3**	**2**	0	0
	s14	8915	1210	95	**3**	**2**	2	9	**3**	**2**	12	4
	s16	5711	368	1372	**4**	**3**	2	4	**3**	**2**	2	3
	s24	4334	699	452	**6**	**10**	2	4	**4**	**4**	1	3
	3s10	47490	3084	3150	**3**	**6**	5	21	**3**	**6**	1	5
Bankshot	s1	45161	12292	10896	**6**	**7**	4	32	**6**	**7**	11	3
	s37	23970	1517	7671	**6**	**5**	4	18	**6**	**4**	0	0
Misdirection	s29	10926	627	451	**7**	**6**	5	1	**7**	**6**	0	4

A brief description of our results on each scenario is given below:

Naive Attacker All attack related steps are detected. The 7 attack-related hosts that are not detected are the hosts inside BPRD which are scanned by the attacker as part of the probe, but do not reply back. Thus they do not supply any information to the external attackers.

Simple Ten All attack related steps are detected. The 240 attack-related hosts not detected are again the scanned hosts which do not reply back.

Five by Five We fail to detect any attack steps or any attack related hosts. In this scenario, the victim host inside the network was not involved in any traffic with external world apart from the attacks launched by outside attacker. There was no profile

generated for this host and hence the attacks could not be distinguished from normal traffic. The attack would have been detected if there was enough traffic which would meet the thresholds related to profiling of internal servers.

Ten by Ten All attack related steps are detected. 11 attack-related hosts not detected include 6 scanned hosts which do not reply back and 5 external scanners who never get a reply back from the hosts which they scan. Thus effectively, these external scanners never get any information about the internal network and hence do not contribute to the actual attack scenario.

s9 All attack related steps and attack related hosts are detected without any false positives.

s10 One attack step is missed in this scenario. The missed step is a failed attack launched by one external attacker on an internal host which is not the eventual victim. Thus this step is not an important part of the whole attack scenario.

s14 All attack related steps and attack related hosts are detected. We also detect some of the background attacks in the traffic. The false positives detected in this scenario arise due to mislabelled connections (replies labelled as initiating connections). This occurs during the conversion of tcpdump data to netflow format.

s16 One attack step is missed in this scenario. The reason for this is same as in scenario *s10*. We also detect two background attacks as a part of the context. The false positives arise because of two outside hosts involved in traffic on random high ports with internal servers which does not conform to the normal profile of those internal servers.

s24 In this scenario three external attackers did a distributed scanning of the internal network. One of the scanners got a reply back from the eventual victim while the other two did not get any replies from the hosts which they scanned. These two scanning steps which did not contribute any information were missed. The false positives occurred because of the same reason as in scenario *s16*.

3s10 All attack related steps and attack related hosts are detected. We also detect some of the background attacks in the traffic. The false positives detected in this scenario arise due to mislabelled connections (replies labelled as initiating connections) or due to outside hosts accessing internal servers on random high ports.

s1 All attack related steps and attack related hosts are detected. We also detect some of the background attacks in the traffic. The false positives detected in this scenario arise because of external hosts accessing internal servers on random high ports.

s37 In this scenario, one of the attackers port scans two internal servers but gets reply only from one which is eventually attacked. The other server does not supply any information back to the attacker. Only this server is not detected while all other involved hosts and attack steps are detected.

s29 All attack steps except for one initial probe, which did not get any replies, were detected. The false alarms occur for the same reason as in scenario *s1*.

5 Discussion

In Section 2 we described the main design goals for our system. The first goal was that the analysis framework should address the inherent tradeoff between false positives and

false negatives. We address this issue in the design of our framework by decomposing the problem into two steps. In the first, we focus on the reduction of false positives, by selecting network events in such a way that gives us high confidence that the events are part of an attack. This was achieved in our simple implementation, through the use of Snort alerts combined with the MINDS anomaly detector. Second, we fill in the missed attack steps by extracting the context from the set of anchor points. By requiring that the anchor points be of high quality (low false positives) we can relax the restrictions on what we add to the context if they are connected to the anchor points. This was also achieved by our simple Context Extraction module, in that relatively few false positives were added to the context when the anchor points contained few false positives.

The second goal was to detect the majority of attack steps in the attack scenario. Evaluating this is not completely straightforward, since not all attack steps would be considered equal, and thus a measure such as a straight percentage of attack-related connections would not be sufficient. This is due to the fact that not all attack steps are of the same importance. For example, in the scenario described in Section 4.3, if we had detected all of the scans and nothing else, we would have detected the vast majority of network connections that were relevant to the attack (95%), but this information would be useless to the analyst. A better measure would be aggregating the connections together into steps (using techniques such as those proposed in [9, 36]), and measuring the number of attack steps that were detected. In this experiment, however, we managed to detect all major attack steps (including attacks, internal stepping stones, and data exfiltration) and many connections in the scanning. Thus we achieved the goal of detecting the majority of the attack steps.

The third goal set forth in Section 2 was high coverage of attacks. In the Skaion scenarios, however, only one attack was present in each scenario. Thus, while not fully tested, this goal was initially achieved in the fact that we were able to detect the main attack in each scenario, except for the one with insufficient profiling information.

The final goal was to make the system modular by design. This design goal was achieved as seen in Section 2. First, the two components in our framework are independent of each other. Thus the implementation of one can be changed without affecting the other. Context Extraction does not depend on how the anchor points are found, as long as they are of high quality. Also, Anchor Point Identification is not concerned with how the anchor points are used, and thus any algorithm can be used to implement the Context Extraction. Also, the system is not tied to any low-level IDS system. None of the design of our framework hinges on the types of alerts available. For example, in our implementation Snort alerts were used. However, any other signature-based system could replace it. The only restriction is that the information needed by the particular implementation of the later stages needs to be present in some form. In addition, we could incorporate other types of information that could be used to detect intrusions, such as system logs [14, 15] and host based IDS alerts [8, 17, 20].

5.1 Limitations and Improvements

This leads us to consider the limitations of our framework. The biggest limitation is that it has greater storage requirements than most IDSs. Snort, for example, examines traffic in real time, and creates alerts based on what it finds. All that needs to be stored

are the alerts. However, our system needs storage of both the low-level IDS alerts as well as the actual network traffic (in some form). The more detailed the data and longer time frame for which the data is stored, the better our system will perform. Also, detecting sophisticated attacks may require the capture of traffic between internal hosts. Capturing the traffic between every host within the network would be difficult, and in many cases infeasible. This storage requirement can be greatly mitigated by storing the data in the net-flow format, where only header information is aggregated and kept. In the University of Minnesota campus network, 1 year of net-flow information can be stored in 0.5 TB, whereas 1 week of tcpdump data requires 2-3 TB of storage. On the other hand, if complete forensic analysis is to be performed, it would be very desirable for the tcpdump data to be present, and thus our framework would pose no extra storage requirement. A operational system designed to store relevant raw network data for forensic analysis is described in [21].

One last important point to discuss is the effect of a framing attack, that is, how an attacker can attack the analysis framework itself. If an attacker knows the rules used by Anchor Point Identification, then he would be able to generate spurious anchor points. However, the amount of anchor points he can generate depends greatly upon the rules used by Anchor Point Identification. If Snort alerts alone were used, then the attacker could easily generate an arbitrary amount of anchor points involving every internal machine [13, 22]. This would basically reduce our framework to a low-level IDS system with a low threshold, flagging much of the traffic as part of an attack. If the rules used for Anchor Point Identification were Snort combined with the MINDS anomaly detector, which was shown to be effective in our experiments, then the effect that the attacker can impose on the anchor points is more limited. Again the attacker can send packets that cause Snort alerts to all the hosts in the network, but to be flagged as anchor points, each of these flows must also be in the top tier of anomalies. By the definition of anomaly, all of these packets would have to be unique with respect to the attributes used by the anomaly detector to rank the network connections. This is a difficult thing to do, since the attacker would have to know a priori what will be considered anomalous for the time period that will be examined. Also the attacker would have to be careful not to send too many packets with certain similarities, since while they may be abnormal when compared to the rest of the traffic, they may form their own cluster and be considered normal with respect to themselves. Also, if too much abnormal traffic is sent, there could be enough abnormal traffic that the abnormal traffic becomes the “norm”, making it very hard to predict what will be flagged as abnormal. For example, if the attacker sent large packets to cause the anomalies, after too many such packets, large packets will be considered normal. In addition, for the Snort and MINDS combination, there is an upper limit on the number of anomalies that will be used for selecting anchor points (due to the cutoff threshold). Thus, the Anchor Point Identification step can, through careful design and implementation, provide some measure of resistance against this type of attack. Another useful aspect of Anchor Point Identification is that it is dynamically configurable (depending on the available data sources), so if it generates too many false positives, it can be run again with a different set of anchor point selection criteria. This opens up two lines of future work on this component. First, we plan to investigate and design better approaches to combine the data sources in order to select anchor points.

Second, we plan to test the designs against these types of attacks to better understand the effects that they can have on the results of our system.

Context Extraction is less resistant to this framing attack, especially when using a port profiling technique, as it is difficult to cause a machine to act outside of its profile, without actually compromising it (to do this effectively, the attacker would need some insider knowledge of the port usage of the internal machines, such as ports on which the internal machines actually offered service but had low enough volume so as to not be profiled). However, Context Extraction would be affected by the false anchor points generated by attacks on the Anchor Point Identification step.

6 Related Work

The most related area of research to this work is IDS alert aggregation and correlation. Alert aggregation has to do with taking alerts from multiple sensors and merging them into one higher level alert. Generally this is done on single events that trigger alerts across multiple sensors. For example, if a subnetwork is set up such that traffic going between it and the outside internet would pass through two Snort sensors, then an attack that triggers a Snort alert would trigger two such alerts. If an analyst is looking at these alerts, it is more efficient if the analyst only looks at the alert once. This gets more difficult when the sensors are not the same type of sensor and report different sets of information, and often a probabilistic approach must be taken [36].

Correlation has two main aspects to it. One is the fusion of different alerts that refer to different events in an attack but are highly related. For example, if there is a DOS attack, and each probe sets off an alert, there will be many alerts from a certain source IP to a certain destination IP. Thus all of these alerts could be merged into one higher level "DOS" alert. This type of fusion can be achieved by clustering alerts based on specific fields in the alert containing matching information [28].

The second area of correlation is in the realm of relating alerts together that fit into an attack scenario. This is the most closely related work in correlation to our approach. Much of the work done in this area has been done in matching prerequisites and consequences of alerts [5, 9, 23–26]. In this approach, the analyst defines the set of actions that must take place before a given alert can occur (its prerequisites), and then once an alert has happened what actions can subsequently take place (its consequences). By placing this information with each alert, a system can match them together (along with extra information such as IP addresses or time-stamps) to form sequences of attacks, or attack scenarios. One limitation of this approach is that it requires extensive expert domain knowledge to determine exactly what is required for an action to take place and what its consequences can be. In addition to prerequisites and consequences, there have also been probabilistic matching approaches proposed [7], and matching detected events against attack models [4].

There have been many other approaches proposed to correlating alerts, and many of these have been incorporated in the comprehensive system in [37].

7 Conclusion and Future Work

We have shown how the multi-step analysis approach can be beneficial in analyzing network traffic and IDS alerts to discover multi-step, sophisticated attacks. One of the most important directions for future work is to utilize the output of the context extraction module in a way that allows for easy analysis. This is the task of the Attack Characterization step, which was ignored in the description of the framework. Even if the output of the context extraction is 100% accurate, it is still a (potentially large) collection of raw network traffic data. Presenting this information to the analyst in a easy to use format, perhaps using visualization techniques, would be beneficial to the analysis, and could help to reduce the effect of false positives from the Context Extraction. Thus, we intend to investigate mechanisms to infer semantic meaning from these connections to determine the full scope of the attack. One way to accomplish this is to use alert aggregation techniques [9, 36]. A second mechanism that could be useful is attack graphs [31], which are possible paths of attack and are generated based on vulnerability assessment and network connectivity information. Matching the detected context against full attack graphs could provide more information to the Attack Characterization step. The final way that we are investigating is the use of visualization techniques to be able to "see" the data, from which an analyst can infer the attack scenario. Another area of future research is to create better and more sophisticated components for the individual steps in the analysis framework. Our approach worked well with the simple components, and improving them will improve the overall result. In addition we plan to test our framework using data captured from a live network.

References

1. Skaion corporation. http://www.skaion.com/news/rel20031001.html.
2. IIS IDA-IDQ Exploit, Cert Advisory CA-2001-13. http://www.cert.org/advisories/CA-2001-13.html.
3. Apache OpenSSL SSLv2 Exploit, Cert Advisory CA-2002-23. http://www.cert.org/advisories/CA-2002-23.html.
4. S. Cheung, U. Lindqvist, and M. Fong. Modeling Multistep Cyber Attacks for Scenario Recognition. In *Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX-III 2003)*, 2003.
5. F. Cuppens and A. Miege. Alert Correlation in a Cooperative Intrusion Detection Framework. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.
6. F. Cuppens and R. Ortalo. LAMBDA: A Language to Model a Database for Detection of Attacks. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, 2000.
7. O. Dain and R. Cunningham. Building Scenarios from a Heterogeneous Alert Stream. In *IEEE Transactions on Systems, Man and Cybernetics*, 2002.
8. H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion detection systems. In *Computer Networks*, 1999.
9. H. Debar and A. Wespi. Aggregation and Correlation of Intrusion Detection Alerts. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, 2001.

10. L. Ertoz, E. Eilertson, P. Dokas, V. Kumar, and K. Long. Scan Detection - Revisited. Technical Report 127, Army High Performance Computing Research Center, 2004.
11. L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, P. Dokas, J. Srivastava, and V. Kumar. Detection and Summarization of Novel Network Attacks Using Data Mining. Technical report, University of Minnesota, 2003.
12. L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, J. Srivastava, V. Kumar, and P. Dokas. *Next Generation Data Mining*, chapter 11. MIT Press, 2004.
13. C. Giovanni. Fun with packets: Designing a stick. http://www.eurocompton.net/stick/papers/Peopledos.pdf.
14. S. Hansen and E. Atkins. Automated System Monitoring and Notification With Swatch. In *Proceedings of the Seventh Systems Administration Conference (LISA'93)*, 1993.
15. D. Hughes. Tklogger. *ftp://coast.cs.purdue.edu/pub/tools/unix/tklogger.tar.Z*.
16. ISS RealSecure. http://www.iss.net.
17. R. Jagannathan, T. Lunt, D. Anderson, C. Dodd, F. Gilham, C. Jalali, H. Javitz, P. Neumann, A. Tamaru, and A. Valdes. System Design Document: Next-Generation Intrusion Detection Expert System (NIDES). Technical report, SRI International, 1993.
18. J. Jung, V. Paxson, A. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proceedings IEEE Symposium on Security and Privacy*, 2004.
19. T. Karagiannis, A. Broido, M. Faloutsos, and K. Claffy. Transport Layer Identification of P2P Traffic. In *Proceedings of the ACM SIGCOMM/USENIX Internet Measurement Conference*, 2004.
20. G. H. Kim and E. H. Spafford. The design and implementation of tripwire: a file system integrity checker. In *CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security*. ACM Press, 1994.
21. K. Long. Catching the Cyberspy: ARL's Interrogator. In *Army Science Conference*, 2004.
22. D. Mutz, G. Vigna, and R. Kemmerer. An Experience Developing an IDS Stimulator for the Black-Box Testing of Network Intrusion Detection Systems. In *Proceedings of the Annual Computer Security Applications Conference*, 2003.
23. P. Ning, Y. Cui, and D. Reeves. Analyzing Intensive Intrusion Alerts via Correlation. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, 2002.
24. P. Ning, Y. Cui, and D. Reeves. Constructing Attack Scenarios Through Correlation of Intrusion Alerts. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2002.
25. P. Ning, D. Reeves, and Y. Cui. Correlating Alerts Using Prerequisites of Intrusions. Technical report, North Carolina State University, Department of Computer Science, 2001.
26. P. Ning and D. Xu. Learning Attack Strategies from Intrusion Alerts. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2003.
27. P. Ning, D. Xu, C. Healey, and R. S. Amant. Building Attack Scenarios through Integration of Complementary Alert Correlation Methods. In *Network and Distributed System Security Symposium*, 2004.
28. P. Porras, M. Fong, and A. Valdes. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, 2002.
29. P. Porras and P. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Distrurbances. *National Information Security Conference*, 1997.
30. L. Schaelicke, T. Slabach, B. Moore, and C. Freeland. Characterizing the Performance of Network Intrusion Detection Sensors. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, 2003.

31. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated Generation and Analysis of Attack Graphs. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.
32. Snort - The Open Source Network Intrusion Detection System. http://www.snort.org.
33. C. Systems. Netflow services and applications. http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps wp.htm.
34. S. Templeton and K. Levit. A Requires/Provides Model for Computer Attacks. In *Proceedings of New Security Paradigms Workshop*, 2000.
35. A. Valdes. Detecting Novel Scans Through Pattern Anomaly Detection. In *Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX-III 2003)*, 2003.
36. A. Valdes and K. Skinner. Probabilistic Alert Correlation. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, 2001.
37. F. Valeur, G. Vigna, C. Kruegel, and R. Kemmerer. A Comprehensive Approach to Intrusion Detection Alert Correlation. In *IEEE Transactions on Dependable and Secure Computing*, 2004.
38. G. Vigna, W. Robertson, and D. Balzarotti. Testing Network-based Intrusion Detection Signatures Using Mutant Exploits. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2004.