

# Visualizing and Identifying Intrusion Context from System Calls Trace

Zhuowei Li, Amitabha Das  
School of Computer Engineering  
Nanyang Technological University  
50, Nanyang Avenue, Singapore, 639798  
Email: zhwei.li@pmail.ntu.edu.sg, asadas@ntu.edu.sg

## Abstract

*Anomaly-based Intrusion Detection (AID) techniques are useful for detecting novel intrusions without known signatures. However, AID techniques suffer from higher false alarm rate compared to signature-based intrusion detection techniques. In this paper, the concept of intrusion context identification is introduced to address the problem. The identification of the intrusion context can help to significantly enhance the detection rate and lower the false alarm rate of AID techniques. To evaluate the effectiveness of the concept, a simple but representative scheme for intrusion context identification is proposed, in which the anomalies in the intrusive datasets are visualized first, and then the intrusion contexts are identified from the visualized anomalies. The experimental results show that using the scheme, the intrusion contexts can be visualized and extracted from the audit trails correctly. In addition, as an application of the visualized anomalies, an implicit design drawback in t-stide is found after careful analysis. Finally, based on the identified intrusion context and the efficiency comparison, several findings are made which can offer useful insights and benefit future research on AID techniques.*

## 1. Introduction

Since the concept of intrusion detection in computer systems was proposed by Anderson [1], many research studies have been carried out to find appropriate intrusion detection techniques to protect the resources in computers or networks [11] [12]. However, the network disaster caused recently by Nimda, MSBlast, and MSSasser highlights the shortcomings of the intrusion detection techniques deployed in our network infrastructures [9], and indicates that intrusion detection techniques still have a long

way to go before they can provide effective protection to computing resources.

In general, intrusion detection techniques can be categorized into *signature-based intrusion detection (SID)* and *anomaly-based intrusion detection (AID)*. SID techniques build (or update) intrusion signature bases that include signatures of all known intrusions. Then, the resource behavior that matches any intrusion signature in the signature base is labeled as an intrusion. Obviously, previously unknown intrusions cannot be detected by these techniques. Besides this drawback, the requirement of instant updating of the intrusion signature bases imposes a severe performance bottleneck on SID techniques.

With the implicit assumption that violations or anomalies are indications of intrusions, anomaly-based intrusion detection techniques have become a focus of intense research as they offer an useful alternative to SID, that is capable of detecting novel intrusions into computing resources [9] [11] [7]. Almost all AID techniques work as follows. First, a model of normally behaving users and/or processes [7] [12] is built. Then, an intrusion is detected by comparing the actual current behaviors against the normal model and taking actions according to some predetermined security policies [3]. However, although many AID techniques have been proposed to date, no single AID technique can effectively detect all types of intrusions into the resources under various scenarios [9]. More specifically, as of now, AID techniques suffer from high false alarm rate that makes it largely ineffective for use in an Intrusion Detection System (IDS) [2].

It is well known that suppressing the false alarms is an effective (or even the only) method for the successful application of AID techniques in real environments. To achieve it, there are already three well-known approaches within the context of intrusion detection. The first one is the multi-sensor fusion [10] in which the outputs of different IDS sensors are aggregated to produce a single alarm. This approach is based on the assumption that any intrusion detection tech-

nique can not classify a set of events as intrusion with sufficient confidence. The second approach uses alert/alarm correlations [4] [14], where the alerts/alarms with different timestamps and/or in different IDS sensors are correlated to build the attack scenarios. The third approach is to cluster the alarms to study their ‘root causes’ [8], and then formulate specific rules to filter the corresponding alarms. However, these three solutions are mainly based on the SID techniques in which the alerts will provide more accurate information about intrusions. When they are applied to AID techniques, the accuracy of the alerts generated by these AID techniques is unavoidably degraded, and under the worst scenario, it will lead to making these three solutions invalid in AID techniques. In stide, an AID technique, a locality frame count scheme is proposed by Forrest et al. [6] to suppress the false alarms assuming that a true alarm will cause a cluster of anomalies in the audit trails. However, the assumption lacks a formal justification due to the dynamic behaviors of diverse computing resources.

In this paper, we introduce a relatively new concept of *intrusion context identification (ICI)* which significantly helps to filter the false alarms for every AID technique. In ICI, using the foundations of an AID technique, the contexts of intrusions are extracted from the audit trails automatically. After a careful study about the intrusion contexts, the false positives can be distinguished from the true positives. Therefore, the false alarms can be filtered out significantly from the alarms. Simultaneously, this information can then be used to improve/modify the current intrusion detection techniques for future detection.

As an application of the intrusion context identification, we propose a scheme to visualize the anomalies in the audit trails left by the intrusions, and then to identify the intrusion contexts. Next, based on the mined intrusion contexts and our knowledge about the audit trails, we can distinguish the false positives from the true positives in the intrusive dataset, reducing false alarms, and increasing the detection rate significantly. As a result, the efficiency of the AID system is enhanced.

Among the proposed AID techniques in the past [7] [12], in this paper, we concentrate on sequence analysis based techniques, and in particular sequence time-delay embedding (*stide*), first proposed by Forrest et al [7] for privileged Unix processes. Although the experiments performed are based on stide and uses system logs of the intruded system, the concept is more general and applicable to other methods and even audit trails of network traffic.

The main contributions of this paper are:

- The concept of *intrusion context identification* is introduced to overcome the shortcomings of AID techniques, i.e. the high false alarm rate in it.
- Based on a typical AID technique, stide, a scheme for intrusion context identification is designed to visual-

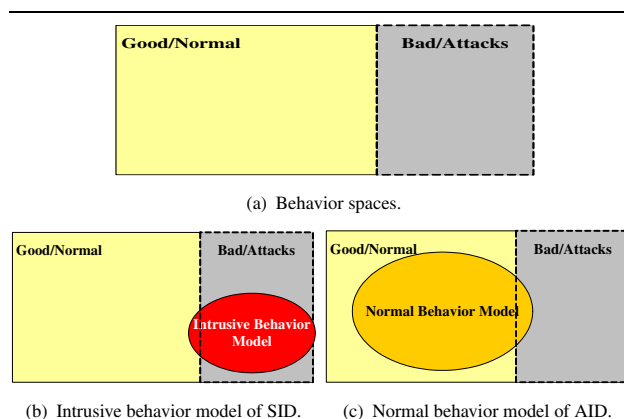
ize the anomalies in the intrusive dataset, and then to identify the intrusion contexts.

- In our experimental results, several meaningful findings from the minimum foreign sequences of intrusions are made, which will benefit future research on AID techniques.
- The efficiency of stide and t-stide are compared by visualizing the anomalies in the intrusive datasets, and a design flaw in t-stide is found.

The remaining paper is organized as follows. Section 2 introduces the concept of intrusion context identification. In section 3, we provide a formal description of the AID technique, stide, which is chosen to illustrate the application of ICI. The intrusion context visualization and identification scheme for stide is proposed and described in section 4. In the following section, several experiments, which are conducted to verify the scheme and to compare the efficiency of stide and t-stide, are described and the results are presented. Finally, conclusions are drawn and future work on intrusion context identification is discussed.

## 2. Intrusion context identification

In a computing resource, there are two behavior spaces for intrusion detection (Figure 1.a): *good/normal behavior space*, and *bad/attacks behavior space*, and they are complementary to each other. Conceptually, signature-based



**Figure 1. Behavior Spaces and Models.**

intrusion detection is based on knowledge of bad/attacks behavior space, and anomaly-based intrusion detection is based on knowledge of good/normal behavior space [3]. Perfect detection of intrusion can be achieved only if we have a complete model of any one of the two behavior spaces, because what is not bad is good and vice versa.

However, it is difficult to model any such behavior space completely and correctly in reality, and Figure 1 (b) and (c) illustrate real behavior models for SID (i.e., intrusive behavior model) and for AID (i.e., normal behavior model) [3]. It is obvious that there exist errors in the behavior model for SID or AID. For example, a part of intrusive behavior model in SID falls into good/normal behavior space.

## 2.1. Intrusion context

Before describing the reason why the intrusion context identification can suppress the false alarms, let us define the intrusion context of an alarm in AID.

**DEFINITION 2.1 (Intrusion Context)** *In anomaly-based intrusion detection, the **intrusion context** corresponding to an alarm is the part of audit trails falling outside the normal behavior model that causes the alarm to be raised.*

**EXAMPLE 2.1** *Assume that the normal behavior model in stide [7] is  $\{1-2-3, 3-1-2, 3-2-4\}$ , and a test trace is '1-2-3-4'. In the test trace, the first 3-gram '1-2-3' is detected as 'normal' but the second 3-gram '2-3-4' will be detected as an alarm, so the intrusion context of the alarm is '2-3-4'.*

## 2.2. Significance of intrusion context identification

Let us first discern what causes the false alarms in anomaly-based and signature-based intrusion detection. Obviously, the part of the intrusive behavior model falling into the good/normal behavior space is the root cause for the false alarms in SID, and the part of the good/normal behavior space outside the normal behavior model is the root cause for the false alarms in AID. In SID, the intrusive behavior model is gleaned from the audit trails of well-known intrusions, so its accuracy can be guaranteed and thus its false alarm rate is very small. However, the normal behavior model in AID is gleaned from the audit trails of the known behaviors, so there lacks the distinguishing ability for the future behaviors falling outside the normal behavior model, i.e., the knowledge scarcity about future behaviors is the main cause for false alarms. As of now, to offset the knowledge scarcity, most of AID techniques try to generalize the normal behavior model to cover more good behavior space [5], and then to reduce the false alarms, but the generalization will cover more bad behavior space as well, which degrades the detection rate. In other words, the generalization is two-edged for intrusion detection.

Intrusion context identification is to meet up the knowledge scarcity in the normal behavior model incrementally, and then to reduce the false alarms in AID by including the intrusion contexts that causes false alarms to augment the normal behavior model. It can be illustrated as follows.

**EXAMPLE 2.2** *Assume that there are two 3-grams in a test trace, which causes false alarms: '2-4-3' and '3-1-1' (i.e., the identified intrusion contexts), and their frequency are 200 and 100 respectively. Without ICI, the number of false alarms in the test trace is 300, but, with ICI, the number of false alarms will fall down dramatically to 2 since the two 3-grams will be inserted in the normal behavior model after the SSO's inspection once they cause their first (false) alarms.*

Using this approach, similar to the intrusive behavior model in SID, the normal behavior model gleaned from the audit trails of known normal behaviors can be guaranteed to be very accurate without two-edged generalization. Furthermore, the normal behavior model can even start with zero-knowledge initially, and incrementally built up using identified intrusion contexts as shown in the following example.

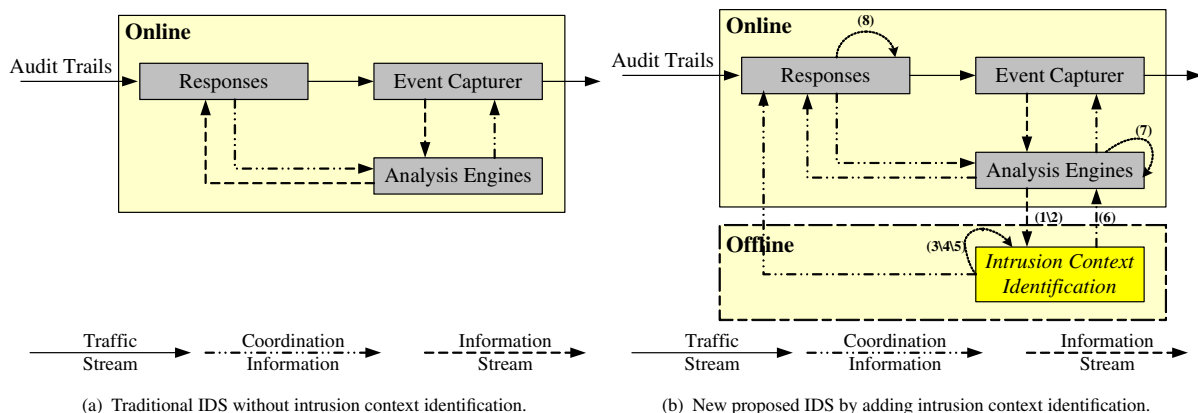
**EXAMPLE 2.3** *Take the same scenario as in example 2.1. Suppose that the test trace is '1-2-3-4' but the normal behavior model is empty. Consequently, the first 3-gram '1-2-3' will be detected as an alarm, and from the identified intrusion context '1-2-3', it is a false alarm, so '1-2-3' is inserted into the normal behavior model. Lastly, if the second 3-gram '2-3-4' triggers a false alarm as well, the normal behavior model will be augmented by two 3-grams: '1-2-3' and '2-3-4'.*

## 2.3. A new IDS architecture with ICI

As a process of intelligently monitoring events in a computer or network system, an intrusion detection system consists of three functional components (Figure 2): (1) the *information sources* that provide the stream of audit trails; (2) the *analysis engines* that identify signs of intrusions; and (3) the *response components* that generate reactions based on the outcome of the analysis engines.

To practically reduce the high false alarm rate in the analysis engines, an extra ICI module is added into the architecture of an IDS as an offline module (Figure 2). For every alarm produced by analysis engines, the ICI module will start to identify the intrusion context for determining whether it is a false alarm. In addition, the ICI module will take considerable amount of time to identify the intrusion contexts, so it is reasonable to propose it as an offline one.

**2.3.1. The Workflow of ICI.** The function of the ICI module is to collaborate with analysis engines to distinguish false alarms from true alarms, and to fine-tune the normal model in the analysis engines to avoid the same false alarms in the future. At the same time, corresponding response strategies should be triggered to avoid the loss due to intrusions, which are detected as true 'alarms'. The procedures involved in the ICI module are as follows:



**Figure 2. The architecture of an intrusion detection system.**

1. Every event processed by analysis engines is copied to the ICI module to store in an audit trails buffer with a predefined context depth;
2. Every *alarm* generated by analysis engines is sent to the ICI module as well;
3. The ICI module extracts the intrusion context of the alarm, and further processes the intrusion context, such as visualizing it or clustering it;
4. The ICI module searches the ‘similar’ intrusion contexts identified before the current alarm;
5. With GUI, a site security officer inspects the intrusion context, and then gives a *decision* on whether the alarm is false and the *confidence* level of the decision considering the similar past intrusion contexts;
6. The ICI vector  $\langle \text{alarm}, \text{decision}, \text{confidence} \rangle$  of the alarm is sent to analysis engines and/or response components to trigger response strategies;
7. The analysis engines fine-tune the normal model using ICI vector to avoid the same false alarms in the future.
8. The response components start corresponding response strategies using the ICI vector.

In general, the root cause analysis in [8] can be applied here to simplify the work in the ICI module, in which the intrusion contexts for alarms can be clustered to avoid the repeatable work done by the SSO. At the same time, the logic in alert correlation [13] can also be applied here to rebuild (normal/intrusive) scenarios. In addition, an intrusion context database is necessary to store ICI vectors and corresponding intrusion contexts for future reference, and it is similar to the intrusion signature set in SID.

## 2.4. Other uses of intrusion context identification

Other than the main objective of reducing false alarms, the ICI module can be utilized as an analysis tool for a number of tasks as listed below, to increase the efficiency or the detection rate, of an AID technique.

- to dig out the intrusion signatures for signature-based IDS since the SID system is computationally more efficient than the AID system.
- to strengthen the alert correlation for AID techniques since one alert cannot reliably determine an intrusion, and its intrusion context can give the exact reasons.
- to study the intrusive characteristics of intrusions, and then to improve the efficiency of existing AID techniques, and to propose more efficient AID techniques.
- to apply it into other anti-intrusion fields, such as computer forensics.

In the following sections, as a first step to develop such an ICI module, we will propose a simple but representative intrusion context identification scheme to show the effectiveness of the ICI module. Using the theoretical basis behind an AID technique, the ICI scheme is designed as follows. First, the anomalies in the intrusive dataset are visualized by foreign sequence graphs. Secondly, the intrusion contexts are mined from these visualized anomalies. After studying the identified intrusion contexts, several useful conclusions are drawn.

## 3. A Formal Description of *stide*

### 3.1. Notations and definitions

*Sequences and Sequence Sets:* Let  $\Sigma$  denote the dataset for a process, which consists of event logs with the identity of the associated running process.  $SS(\Sigma, l)$  denotes the set

of all the sequences of length  $l$  ( $l \geq 0$ ), which are collected from  $\Sigma$ . As a special case,  $SS(\Sigma, 0) = \{\phi\}$ . Furthermore,  $SS(\Sigma) = \bigcup_{l=0}^{+\infty} SS(\Sigma, l)$ , and  $S$  is a sequence in  $SS(\Sigma)$ .

*Set Operations:* For given datasets  $\Sigma_a$  and  $\Sigma_b$  of a process and corresponding sequence sets  $SS(\Sigma_a, l)$  and  $SS(\Sigma_b, l)$  ( $l \geq 0$ ), the set operations ( $\cup, -$ ) are defined as follows:

$$\begin{aligned} (1) \quad & SS(\Sigma_a, l) \cup SS(\Sigma_b, l) \\ &= \{S | (S \in SS(\Sigma_a, l)) \vee (S \in SS(\Sigma_b, l))\} \\ (2) \quad & SS(\Sigma_a, l) - SS(\Sigma_b, l) \\ &= \{S | (S \in SS(\Sigma_a, l)) \wedge (S \notin SS(\Sigma_b, l))\} \end{aligned}$$

**3.1.1. Foreign sequences and self sequences.** Let  $\Sigma_a$  be the *reference dataset* or *training dataset*, and  $\Sigma_b$  be the *target dataset* or *test dataset*. For any sequence  $S \in SS(\Sigma_b)$ , if  $S$  is also in  $SS(\Sigma_a)$ ,  $S$  will be called a *self sequence*, otherwise, it is a *foreign sequence* to  $\Sigma_a$ . For the convenience of expression,  $FRGN(\Sigma_b|\Sigma_a)$  is defined as the set of foreign sequences of  $\Sigma_b$  w.r.t.  $\Sigma_a$ . Similarly, the set of self sequences is defined as  $SELF(\Sigma_b|\Sigma_a)$ . Obviously,  $\phi \in SELF(\Sigma_b|\Sigma_a)$ , and  $FRGN(\Sigma_b|\Sigma_a) \cup SELF(\Sigma_b|\Sigma_a) = SS(\Sigma_b)$ .

A sequence  $S$  in  $FRGN(\Sigma_b|\Sigma_a)$  will be called a *minimum foreign sequence (MFS)* [15] if none of its subsequences<sup>1</sup> is in  $FRGN(\Sigma_b|\Sigma_a)$ , i.e. all of its subsequences are in  $SELF(\Sigma_b|\Sigma_a)$ . The set of all minimum foreign sequences is denoted as  $MFS(\Sigma_b|\Sigma_a)$ .

**3.1.2. Sequence length.** In the set of  $MFS(\Sigma_b|\Sigma_a)$ , the smallest length of all its sequences is defined as:

$$\begin{aligned} |MFS|_{min}(\Sigma_b|\Sigma_a) &= \min_{S \in MFS(\Sigma_b|\Sigma_a)} length(S) \\ &= \min(\{l | l \geq 0; SS(\Sigma_b, l) - SS(\Sigma_a, l) \neq \Phi\}) \end{aligned} \quad (1)$$

where, for convenience, we use the notation  $|\dots|$  to represent the length of any member sequence in the sequence set MFS, instead of the size of the set. In words,  $|MFS|_{min}(\Sigma_b|\Sigma_a)$  represents the minimum length of any sequence  $S \in \Sigma_b$  that is not a sequence in  $\Sigma_a$ . From this definition,  $|MFS|_{min}(\Sigma_b|\Sigma_a) \geq 1$ .

### 3.2. Formalizing stide

In the experimental setup for stide [18], there are two datasets for every process, the normal dataset  $\Sigma_{nml}$ , and the intrusive dataset  $\Sigma_{int}$ , which are defined below.

**DEFINITION 3.1 (Normal Dataset)** *A normal dataset is a dataset  $\Sigma_{nml}$  that **MUST** be collected in the normal run of the process without any intrusion.*

<sup>1</sup> In this paper, the terms *subsequence* and *supersequence* will always imply contiguity.

**DEFINITION 3.2 (Intrusive Dataset)** *An intrusive dataset  $\Sigma_{int}$  is a dataset that was collected when one or more intrusions were occurring targeting on the process.*

In terms of these two datasets from the same process, stide can be formally described as follows. Let  $DW$  ( $\geq 1$ ) denote the size of the detector window. In the modeling phase, the normal model of the process is obtained as:  $SS(\Sigma_{nml}, DW)$ . Then, in the detecting phase, the foreign sequences in the intrusive dataset  $\Sigma_{int}$ ,  $FMS(\Sigma_{int}|\Sigma_{nml}, DW)$ , are enumerated:

$$FMS(\Sigma_{int}|\Sigma_{nml}, DW) = SS(\Sigma_{int}, DW) - SS(\Sigma_{nml}, DW) \quad (2)$$

If  $FMS(\Sigma_{int}|\Sigma_{nml}, DW) \neq \Phi$ , the intrusion(s) in the intrusive dataset  $\Sigma_{int}$  can be detected with the detector length  $DW$  [7] [15] [18]. It is evident that the sequence set  $FMS(\Sigma_{int}|\Sigma_{nml}, DW)$  is strongly related to  $MFS(\Sigma_{int}|\Sigma_{nml})$  via  $|MFS|_{min}(\Sigma_{int}|\Sigma_{nml})$ :

$$\begin{aligned} & |MFS|_{min}(\Sigma_{int}|\Sigma_{nml}) \leq DW \\ \iff & FMS(\Sigma_{int}|\Sigma_{nml}, DW) \neq \Phi \end{aligned} \quad (3)$$

Practically, even though the underlying principle is very simple, stide can detect most of the intrusions into the processes (*the datasets from UNM*[7]). For this reason, it is accepted as a typical and effective anomaly-based intrusion detector in many research studies. It is obvious that any sequence  $S$  in  $MFS(\Sigma_{int}|\Sigma_{nml})$  will cause an alarm, and  $S$  is the intrusion context of the alarm.

## 4. Visualizing and Identifying intrusion contexts

From the above formal description of stide, we know that the anomalies in the intrusive dataset will be indicated by a foreign sequence set  $FMS(\Sigma_{int}|\Sigma_{nml}, DW)$ . Therefore, it will be helpful to visualize these foreign sequences in our ICI scheme. Furthermore, the detector window size  $DW$  is also critical to the foreign sequence set as the foreign sequence will not be detected if its length is larger than  $DW$ . Therefore, in the graphs that we produced for the visualization of the intrusion context, the length of every foreign sequence in the intrusive dataset will be shown to indicate whether it can be detected by the stide detector with length  $DW$ . Specifically, the graph shows the foreign sequence length of the current event (vertical) against its index (horizontal). For convenience, we call these graphs as *foreign sequence graphs (FSG)*. Ultimately, to identify intrusion contexts correctly, we should try to identify minimum foreign sequences, and the events that are associated with the MFS can then be identified as the intrusion context.

To achieve it, the intrusive dataset can be processed in two ways: (1) splitting it into blocks, and to evaluate every block; (2) evaluating every event in it. According to the principles of stide, the first method of splitting will have the

possibility to break the foreign sequences. Therefore, we will use the second approach, i.e. every event in the intrusive dataset will be evaluated for its significance for intrusion context identification.

#### 4.1. FSG: foreign sequence graph

An FSG graph is built as follows. Let us assume that the total length of the audit trails for a process is  $p$ , thus, it can be represented as  $\{e_1, e_2, \dots, e_p\}$ , and the lengths of stide detectors range from 1 to  $N$ . First, the stide detectors are built from the normal dataset with the detector window size  $DW$  varying from 1 to  $N$ . Then, for every event  $e_i$ , if  $e_{i-l+2} \dots e_i \in SSEQ(\Sigma_{in}|\Sigma_{no})$  but  $e_{i-l+1} \dots e_i \in FSEQ(\Sigma_{in}|\Sigma_{no})$ , the length of the precedent foreign sequence  $FSL(e_i) = length(e_{i-l+1} \dots e_i) = l$  will be shown in the FSG graph. Specifically, the following algorithm can be used to calculate the foreign sequence length series. To illustrate the intrusion context, the detection re-

---

**Algorithm 1** Calculating the foreign sequence length series.

---

**Require:** The event sequence of the process  $e_1, e_2, \dots, e_p$ ; and the stide self model series  $stide_1, stide_2, \dots, stide_N$ ;

```

for  $i=1$  to  $p$  do
   $seq=$ NULL;  $FSL(e_i) = N + 1$ ;
  for  $j=0$  to  $N-1$  do
    if  $i-j \leq 0$  break;
    Insert  $e_{i-j}$  to  $seq$  as the first element;
    if  $seq$  is-not-in  $stide(j+1)$  then
       $FSL(e_i) = length(seq)$ ; break;
    end if
  end for
end for

```

Output  $FSL(e_1), FSL(e_2), \dots, FSL(e_p)$

---

sults  $FSL(e_1), FSL(e_2), \dots, FSL(e_p)$  are used to draw the foreign sequence graph for the intrusive process.

An intrusion context will be determined from the foreign sequence graph when the events near it have smaller foreign sequence length values compared with its neighbours. In practice, if one minimum foreign sequence is  $e_m e_{m+1} \dots e_n$  ( $n - m + 1 \leq N$ ), it will be identified by the local lowest point in its foreign sequence graph, which is expressed as  $FSL(e_n) = n - m + 1$ , since  $FSL(e_{n-1}) \geq n - m + 1$ , and  $FSL(e_{n+1}) \geq n - m + 1$ . Thus, the precedent  $n - m + 1$  events consist of the intrusion context, that is, the minimum foreign sequence.

#### 4.2. MFSs identified from FSGs

In the formal description of stide, an intrusion will be detected if one minimum foreign sequence appears in the intrusive dataset at least. In general, all the foreign sequences are supersequences of the minimum foreign sequences in the audit trails of a process [15]. Therefore, it is useful to

collect the minimum foreign sequences of an intrusion from its foreign sequence graph.

According to the definition of minimum foreign sequences, the non-MFS foreign sequences contain one or more minimum foreign sequences. For example, suppose that one minimum foreign sequence is  $x_1, x_2, \dots, x_l$ , the foreign sequence can be constituted as follows:  $y_1 \dots y_m x_1 x_2 \dots x_l y_{m+1} \dots y_{m+n}$ , where,  $m \geq 0, n \geq 0$ ,  $y_1 \dots y_m$  and  $y_{m+1} \dots y_{m+n}$  are non-MFS sequences. Because the non-MFS foreign sequences provide no additional information in comparison to the MFS(s) in them [15], they will be filtered out.

Fortunately, in the generation of the foreign sequence graph (Algorithm.1), the prefix sequence  $y_1 \dots y_m$  is filtered out in the beginning. Therefore, to collect the minimum foreign sequence, only the suffix sequence  $y_{m+1} \dots y_{m+n}$  should be filtered out. Therefore, from the foreign sequence graphs, the method for trimming down non-MFS foreign sequences can be inferred as follows: if  $FSL(e_i) = FSL(e_{i-1}) + 1$ , the foreign sequence identified by  $FSL(e_i)$  will be filtered out since it is included in the foreign sequence identified by  $FSL(e_{i-1})$ .

The direct use of the identified intrusion contexts is to find out the characteristics of intrusions in the intrusive dataset. Secondly, after careful study, the intrusion contexts that lead to false alarms in stide will be identified as well, and they can be used to fine-tune the normal model by including them (it is worth nothing that the fine-tune methods are different with respect to different AID techniques). If there is any response strategy issued, some anti-response directions should be send out to make the strategy invalid.

In summary, the scheme will be useful to study the characteristics of intrusions, to remove the false alarms in the detection phase, and to improve the efficiency of AID techniques. However, every coin has two sides. The identified intrusion context can be utilized to create cleverer intrusions, such as the information hiding techniques [16] and the mimicry attacks [17]. It is slightly said that the competition between the security researchers and the hackers is endless without a predestined winner.

## 5. Experiments and evaluations

The experiments are conducted on a pentium III 1.0G computer with 256MB memory, in which the popular Red-Hat Linux operating system is installed, and the programming language is C++. To avoid unnecessary anomalies (or intrusions) into our experiments, the experiments are done without any physical network connection to our system.

Normal Datasets	Intrusive Datasets	No. of Traces	No. of System Calls
live-named-UNM	---	142	9230572
---	buffer overflow-1	3	969
---	buffer overflow-2	2	831
live-lpr-MIT	---	2703	2926304
---	lprcp	1001	165248
sendmail-CERT	---	294	1576086
---	syslog-local-1	6	1516
---	syslog-local-2	6	1574
---	syslog-remote-1	7	1861
---	syslog-remote-2	4	1553
---	cert-sm565a	3	275
---	cert-sm5x	8	1537
sendmail-UNM	---	346	1799764
---	decode	36	3067
---	forward loops	36	2569
---	sunsendmailcp	3	1119
syn-wu-ftpd	---	8	180315
---	misconfiguration	5	1363
syn-xlock-UNM	---	71	339177
---	buffer overflow-1	1	489
---	buffer overflow-2	1	460

**Table 1. The Dataset Specifications.**

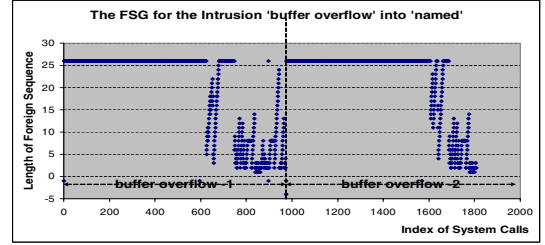
## 5.1. Evaluation datasets

The datasets from the University of New Mexico that are used in [15] [18] have been used in our experiments. The normal and intrusive datasets for selected processes are specified in Table 1. From the table, our selected datasets represent most processes and intrusions. Furthermore, to analyze the characteristics of every intrusion, its intrusive dataset, even into the same process, is treated as an individual dataset since every intrusion will leave its own characteristic intrusion contexts in its audit trails.

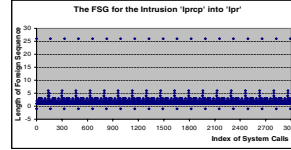
## 5.2. FSG: visualizing intrusion contexts

Figure 3 summarizes the foreign sequence graphs for every pair of an intrusion and a process. For the convenience of comparison, some FSG graphs are compressed into one subfigure, and their borders are split with vertical lines for different intrusive datasets. To easily identify the boundaries, we introduce dummy values of  $FSL=-4$  between different intrusive datasets, and  $FSL=-1$  between different processes in one dataset. For instance, in subfigure (a), the vertical line at '974' splits the two intrusive datasets for 'buffer overflow' into 'named'. In addition,  $N=25$  in our experiments, so the maximum value of the foreign sequence length is 26 (Algorithm 1). It is worth noting that every point in FSGs is positioned by the index of a system call and its corresponding foreign sequence length.

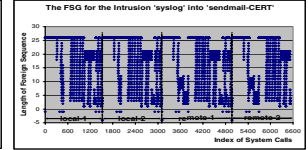
From these FSG graphs, the number of foreign sequences in every dataset is significant, and most of the intrusions can be detected by stide with a proper detec-



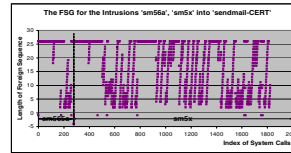
(a) Process 'named' and Intrusion 'buffer overflow'.



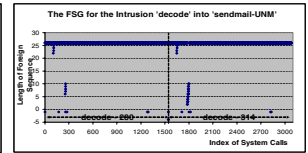
(b) Process 'lpr', and Intrusion 'lprcp'.



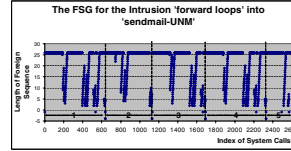
(c) Process 'sendmail-CERT', and Intrusion 'syslog'.



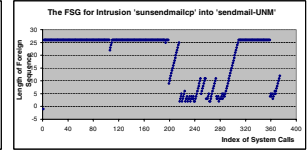
(d) Process 'sendmail-CERT', and Intrusions 'sm565a' and 'sm5x'.



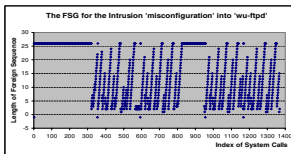
(e) Process 'sendmail-UNM', and Intrusion 'decode'.



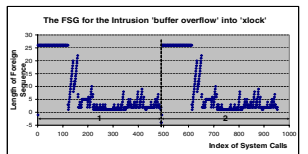
(f) Process 'sendmail-UNM', and Intrusion 'forward loops'.



(g) Process 'sendmail-UNM', and Intrusion 'sunsendmailcp'(3 processes).



(h) Process 'ftpd', and Intrusion 'misconfiguration'.



(i) Process 'xlock', and Intrusion 'buffer overflow'.

**Figure 3. Foreign Sequence Graphs.**

tion window. Furthermore, one significant phenomenon reflected in most of FSG graphs except subfigure (b) is that the intrusions will not cause foreign sequences in the first stage. For example, in subfigure (a), only after 600 system calls, there are some foreign sequences generated. To some degree, it is indicated that there is a 'warmup' stage for most intrusions, at least for the intrusions detected by stide-like AID detectors.

Secondly, three instances of the intrusion 'sunsendmail' in subfigure (g) have caused identical FSG graphs. That is to say, they have the same intrusion characteristics. At the

same time, for the different instances of the same intrusion in subfigures (a)(c)(f) or (i), their FSG graphs are very similar as well. This phenomenon will benefit the research on AID techniques since it is unnecessary to design specific technique to detect different runs of an intrusion.

Thirdly, for different intrusions into a process, their FSG graphs are different. For example, for the process ‘sendmail’ from CERT, the FSG graph of the intrusion syslog in subfigure (c) is different from the FSG graph of the intrusion sm565a or sm5x in subfigure (d); the FSG graphs of the intrusions into the same process ‘sendmail’ from UNM, ‘decode’ in subfigure (e), ‘forward loops’ in subfigure (f) and ‘sunsendmailcp’ in subfigure (g), are different mutually as well. This fact proves that there are no general detection strategies to detect all such intrusions into a process, and specific detection strategies (e.g. stide detectors with different length) are needed to detect all these intrusions.

Lastly, it is obvious that, especially in subfigures (c)(d)(e)(f) and (g), there are precursor foreign sequences with larger length in the FSG graphs of some intrusions. Therefore, under these scenarios, the larger the stide detector size is, the sooner the first alarm to indicate the occurrence of an intrusion. However, it is well known that the larger stide detector window will degrade the efficiency of stide. That is, there is a clear tradeoff between the MTTA (Mean Time To Alarm) and the efficiency of stide: if we want to detect the intrusion earlier (thereby lowering the MTTA), we need a larger detector window, degrading the detection efficiency.

### 5.3. Identifying intrusion contexts: MFSs

The minimum foreign sequences in the audit trails are one of the characteristics left by an intrusion, which is utilized by stide-like anomaly detectors to detect the intrusion. As mentioned earlier, MFSs are the intrusion contexts of alarms in stide. Based on the identified intrusion contexts in these datasets, their numerical statistics are summarized in Figure 4 and Table 2.

*Numerical statistics of the identified MFSs.* From the total number of non-duplicated minimum foreign sequences in every intrusive dataset (Figure 4), we can infer that (1) The detector window  $DW = 2$  is enough to detect most of the intrusions except the intrusion instance ‘decode-280’; (2) From the FSG in Figure 3(e) and the numerical statistics in Figure 4 of the intrusion instance ‘decode-280’, we observe that the minimum length of MFSs is 6. From its positional index, we can easily find that this corresponds to the sequence ‘2-95-6-6-95-5’. The following list shows all the MFSs occurring in the two decode instances and the corresponding system calls:

- *decode-280*  
process 283: 2-95-6-6-95-5

- *decode-314*  
process 317: 112-6, 6-19, 2-95-1, 2-95-6-6-95-5.  
\* 1:exit, 2:fork, 5:open, 6:close, 19:lseek, 95:connect, 112:vtrac

From this observation and Eqn (3), we can now confidently assert that the intrusion instance ‘decode-280’ leads to the magic number 6 for stide [18][15] since  $|MFS|_{min}(\Sigma_{int}|\Sigma_{nml}) = 6$ . (3) As there are only a small number of minimum foreign sequences with their lengths beyond 7, the efficiency of stide detectors will not be improved much if the detection window  $DW > 7$ .

MFS Length	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	sum
named-1	5	34	11	0	3	1	0	2	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	59
named-2	5	19	6	1	1	1	0	1	0	0	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	38
lpr-1	26	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	37
sm-cert-1	3	20	13	5	1	4	2	0	0	1	0	0	1	1	0	0	2	2	0	0	0	0	0	0	0	55
sm-cert-2	3	27	20	5	2	3	4	0	0	1	0	0	1	1	0	0	2	2	0	0	0	0	0	0	0	71
sm-cert-3	3	30	20	5	2	3	4	2	0	1	1	0	2	1	0	0	2	2	0	0	0	0	0	0	0	78
sm-cert-4	2	9	9	4	1	3	4	2	0	1	1	0	2	0	0	0	2	2	0	0	0	0	0	0	0	42
sm-cert-5	0	8	2	2	0	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	15
sm-cert-6	0	23	25	17	6	4	5	2	1	0	0	1	0	0	0	4	2	4	0	0	0	0	0	0	0	96
sm-unm-1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
sm-unm-2	0	2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
sm-unm-3	0	11	9	5	2	0	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33
sm-unm-4	0	3	1	2	0	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11
sm-unm-5	0	11	9	2	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30
sm-unm-6	0	10	10	5	2	0	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33
sm-unm-7	0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
sm-unm-8	0	7	7	4	2	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24
sm-unm-9	0	7	7	4	2	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24
sm-unm-10	0	7	7	4	2	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24
ftp-1	3	34	14	8	4	2	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	68
xlock-1	10	41	8	2	2	3	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	68
xlock-2	10	41	8	2	2	3	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	68

named-1,2: the buffer overflow-1,2  
sm-cert-1..6: syslog-local-1,2, syslog-local-3,4 sm565a and sm5x  
sm-unm-1..10: decode-280,314, forward loops-1,2,3,4,5, sunsendmailcp-10763,10801,10814  
ftp-1: misconfiguration  
xlock-1,2: buffer overflow-1,2

Figure 4. The Number of Minimum Foreign Sequences (non-duplicated) in all Intrusive Datasets.

Intrusion	Num. of Runs	Num. of MFSs of Each Run	Num. of Shared MFSs
decode	2	{2,5}	2
buffer overflow into xlock	2	{68,68}	68
buffer overflow into named	2	{59,38}	33
sunsendmailcp	3	{24,24,24}	24
forward loops	5	{33,11,30,33,4}	0
syslog-local	2	{55,71}	52
syslog-remote	2	{78,42}	42

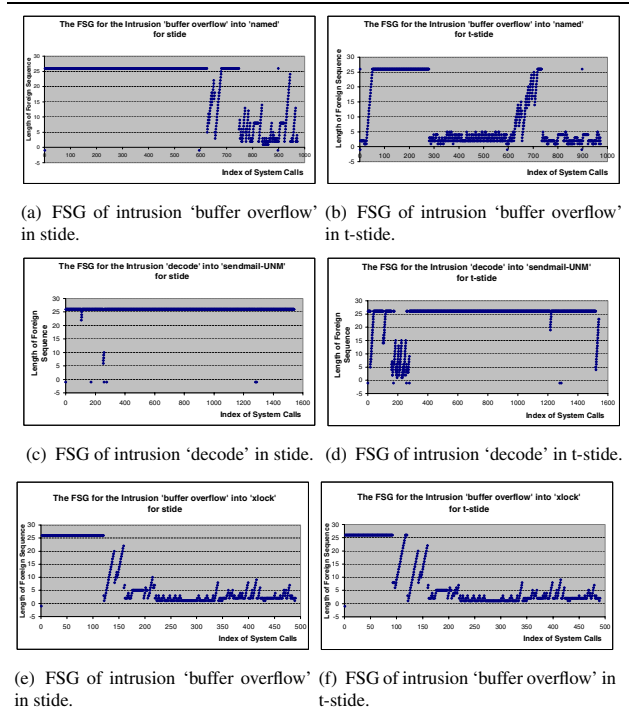
Table 2. Shared Minimum Foreign Sequences By the Same Intrusion into the Same Process.

In addition, we also note that different runs of an intrusion will share most of the minimum foreign sequences (Table 2). Especially, different runs for ‘sunsendmailcp’ have the same set of minimum foreign sequences. This discovery benefits the research on AID techniques because the diversity of different runs of the same intrusion is not too large to warrant the designing of one specific (or ad-hoc) IDS system for each of its runs. At the same time, it also strengthens our earlier assertions that the FSG graphs are intrusion-

specific, and that different runs of an intrusion almost have the same intrusion characteristics, i.e. the intrusion context.

Also, from Table 2 and Figure 4, the large number of minimum foreign sequences, especially for the intrusion ‘buffer overflow’, will discourage the mimicry attacks [17] and the information hiding paradigm [16] greatly.

#### 5.4. Visually comparing stide and t-stide



**Figure 5. FSGs for comparing stide and t-stide.**

In [18], a variance of stide, t-stide, is proposed by discarding any sequence whose overall frequency is less than a threshold  $t$ . However, from the experimental results in [18], the efficiency of t-stide is not better than, or even worse than, the efficiency of stide. To identify the reasons for it, we visually compare the FSGs of stide and t-stide with  $t = 0.00001$ , as shown in Figure 5. For convenience, the FSGs of the same intrusive dataset for stide and t-stide are positioned side by side.

(A) It is quite apparent from a visual comparison that the foreign sequences and MFSs produced by t-stide constitute a superset of the foreign sequences generated by stide. This implies that the detection ability of t-stide is exactly the same as that of stide, but t-stide is likely to generate many more false alarms. This conclusion coincides exactly with the experimental results on the performance of t-stide [18].

(B) Interestingly, stide and t-stide provide identical detection performance for the program ‘ftpd’. This is perfectly consistent with our finding that their FSGs for ‘ftpd’ are also identical (therefore, they are not shown in Figure 5, c.f. Figure 3.h).

(C) But for the process ‘named’ in Figure 5 (a) vs (b), t-stide presents a very different picture. The reason for this phenomenon becomes clear when one examines the t-stide scheme. In t-stide, a sequence  $S_k$  is defined as rare and discarded if  $N_k \leq t \times \sum_{i=1}^n N_i$ , where  $\{S_1, S_2, \dots, S_n\}$  are the normal sequences in the training dataset,  $N_i$  is the frequency of sequence  $S_i$ , and  $k \in \{1, 2, \dots, n\}$ . However, this scheme leads to a serious problem when the number of unique sequences,  $n$ , becomes large. This can be seen as follows. In general, the mean frequency of the sequence set is  $N_{mean} = \frac{1}{n} \sum_{i=1}^n N_i$ , therefore,

$$N_k \leq t \times \sum_{i=1}^n N_i = t \times n \times N_{mean} \quad (4)$$

From the above equation, if  $t \times n \geq 1$ , then a sequence can be designated as rare and discarded even if its frequency is larger than the mean frequency of the sequence set. Since  $t$  is predefined in t-stide, this possibility becomes larger if  $n$  is very large (which is possible if the process is complex). Obviously, this makes the condition in t-stide for identifying rare sequences quite unreasonable. In [18], the dilemma is avoided by setting an unusually small value for the threshold  $t$  ( $=0.00001$ ) but limiting  $n$  in the experimental datasets. To overcome this drawback, the ‘rare’ sequence can be redefined as the sequence  $S_{rare}$  if  $N_{rare} \leq t' \times \frac{1}{n} \sum_{i=1}^n N_i$ , that is,  $N_{rare} \leq t' * N_{mean}$ .

(D) It is worth noting that a program normally consists of a beginning transaction, a processing transaction, and an end transaction. Since the processing transaction is always repetitive, the related system call sequences tend to dominate, making the sequences in the beginning and end transactions infrequent (i.e. rare). However, these sequences in the beginning and end transactions are critical for the running of every program. Therefore, for the completeness of the AID detector, such ‘rare’ sequences should not be discarded.

## 6. Conclusions and future work

In this paper, the concept of intrusion context identification is first introduced, and one such simple but representative scheme is proposed based on one typical AID technique, stide. In this scheme, the anomalies in the intrusive datasets are visualized in foreign sequence graphs.

From the visualization graphs (FSGs), the minimum foreign sequences in the intrusive datasets are identified as intrusion contexts. The experimental results show that the intrusion contexts can be identified from the audit trails successfully. Furthermore, a visual comparison (with the visualized anomalies) is done between stide and t-stide as well. From the comparison, a design drawback in t-stide is found.

In addition, a number of important facts are discovered, which can offer useful insight and benefit the research on AID techniques. They are listed below:

1. Different runs of an intrusion almost have the same intrusion characteristics;
2. Different intrusions into one process will cause different anomalies in the intrusive datasets;
3. Most of the intrusions have precursors, which are useful to provide short MTTA;
4. Some intrusions cannot be detected in the first stage when no anomalies are caused.
5. There is diminishing rate of return in terms of efficiency with the increase of the detector window.
6. The 'rare' sequences in the normal model are critical for the efficiency of an AID technique.

In our future work, the practical and promising concept of intrusion context visualization and identification will be further applied to different environments (e.g. the networks, the windows platform) to analyze the intrusion characteristics. Since the intrusion contexts extraction must be founded on sound principles, a proper formal framework for AID techniques is a necessary step for that. At the same time, in the new proposed IDS (Figure 2), finer design details need to be worked out for the intrusion context identification module. Most importantly, an user-friendly GUI is needed for SSO to manipulate the intrusion contexts. Ultimately, an AID technique with a little generalization in training the normal behavior model will be combined with intrusion context identification to build an efficient solution for intrusion detection.

## 7. Acknowledgements

The datasets utilized in our experiments, and the documentations about them are downloaded from University of New Mexico. We are grateful to Professor Forrest and her research group for their scientific generosity.

## References

- [1] J. Anderson. Computer security threat monitoring and surveillance. Technical report, James P Anderson Co., Fort Washington, Pennsylvania, April 1980.
- [2] S. Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 1–7, 1999.
- [3] S. Chari and P. Cheng. BlueBox: A Policy-Driven, Host-based Intrusion Detection System. *ACM Transaction on Information and System Security*, 6(2):173–200, May 2003.
- [4] F. Cuppens and A. Mieke. Alert correlation in a cooperative intrusion detection framework. In *Proceedings. 2002 IEEE Symposium on Security and Privacy*, pages 187 – 200, May 2002.
- [5] H. Debar, M. Dacier, and A. Wespi. A revised taxonomy for intrusion detection systems. *Annales des Telecommunications*, 55(7–8):361–378, 2000.
- [6] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, pages 120–128. IEEE Computer Society Press, 1996.
- [7] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6(3):151–180, 1998.
- [8] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transaction on Information and System Security*, 6(4):443–471, 2003.
- [9] R. Kemmerer and G. Vigna. Intrusion detection: a brief history and overview. *IEEE Computer*, 35(4):supl27 – supl30, April 2002.
- [10] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In *19th Annual Computer Security Applications Conference*, Las Vegas, Nevada, December 08 - 12 2003.
- [11] W. Lee and S. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3(4):227–261, Nov. 2000.
- [12] M. Mahoney and P. Chan. Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks. In *SIGKDD 2002*, July 23-26 2002.
- [13] P. Ning, Y. Cui, and D. R. fand D. Xu. Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):274–318, May. 2004.
- [14] P. Ning, Y. Cui, and D. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 245–254, 2002.
- [15] K. Tan and R. Maxion. Determining the operational limits of an anomaly-based intrusion detector. *IEEE Journal on selected areas in communications*, 21(1):96–110, Jan. 2003.
- [16] K. Tan, J. Mchugh, and K. Killourhy. Hiding intrusions: From the abnormal to the normal and beyond. In *Proceedings of Information Hiding 2002*, pages 1–17, 2002.
- [17] D. Wagner and P. Soto. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 255–264, 2002.
- [18] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145, 1999.