Doing intrusion detection using embedded sensors—Thesis proposal*

Diego Zamboni

Center for Education and Research in Information Assurance and Security
1315 Recitation Building
Purdue University
West Lafayette, IN 47907-1315

CERIAS Technical Report 2000-21

October 18, 2000

Abstract

Intrusion detection systems have usually been developed using large host-based components. These components impose an extra load on the system where they run (sometimes even requiring a dedicated system) and are subject to tampering or disabling by an intruder.

Additionally, intrusion detection systems have usually obtained information about host behavior through indirect means, such as audit trails or network packet traces. This potentially allows intruders to modify the information before the intrusion detection system obtains it, making it possible for an intruder to hide his activities.

In this document I propose work that will attempt to show that it is possible to perform intrusion detection using small sensors embedded in a computer system. These sensors will look for signs of specific intrusions. They will perform *target monitoring* by observing the behavior of the system directly, instead of through an audit trail or other indirect means. Furthermore, by being built into the code of the operating system and its programs, they may not impose a considerable extra load on the host they monitor.

I will also explore the possibility of applying a group of sensors built to detect known intrusions, to detecting new intrusions. If this is shown to be possible, it would be a step towards determining the types of data that need to be collected to successfully detect new intrusions.

The work I propose is divided in four stages: a) building the necessary infrastructure for the implementation of the sensors, b) implementing sensors for detecting known intrusions, c) testing new attacks against the group of implemented sensors, and d) performing analysis on the data obtained in step (c) to determine if the existing sensors can be used to detect new attacks.

1 Introduction

The field of intrusion detection has received increasing attention in recent years. One reason for this is the explosive growth of the Internet and the large number of networked systems that exist in all types of organizations. The increase in the number of networked machines has lead to an increase in unauthorized activity [9], not only from external attackers, but also from internal attackers, such as disgruntled employees and people abusing their privileges for personal gain [42].

To detect and counteract unauthorized activity, it is desirable for network and system administrators to monitor activity in their networks. Because even a single host can generate several megabytes of logging and audit data in a single day, it is desirable to have tools that can automatically monitor computer systems and detect when unauthorized activity is taking place. These tools are commonly known as intrusion detection systems.

*Portions of this work were supported by sponsors of CERIAS. This thesis proposal was defended on January 14, 2000.

In the last few years a number of intrusion detection systems have been developed, both in the commercial and academic sectors. These systems have used different approaches to detecting unauthorized activity, and have given us some insight into the problems that still have to be solved before we can have intrusion detection systems that are useful and reliable in production settings for detecting a wide range of intrusions.

Most of the existing intrusion detection systems have used central data analysis engines [e.g. 11, 31] or per-host data collection and analysis components [e.g. 23, 41]. Even systems designed using software agents [e.g. 2, 3] have in practice implemented agents as separate processes. All of these approaches are subject to the following problems:

- They continuously use additional resources in the system they are monitoring, even when there are no intrusions occurring.
- Because the components of the intrusion detection system are implemented as separate processes, they are subject to tampering. An intruder can potentially disable or modify the programs running on a system, rendering the intrusion detection system useless or unreliable.

In this document I propose the use of embedded sensors to perform intrusion detection. These sensors would be built into the code of the operating system or into the system programs. This would make them use additional resources only when they are executed or triggered, and be more resistant to tampering.

2 Background

I start by introducing concepts that are used throughout this paper.

2.1 Intrusion Detection

Intrusion detection is defined as "the problem of identifying individuals who are using a computer system without authorization (i.e., 'crackers') and those who have legitimate access to the system but are abusing their privileges (i.e., the 'insider threat')" [39]. I add to this definition the identification of *attempts* to use a computer system without authorization or to abuse existing privileges. My working definition matches the one given by Heady et al. [22], where an intrusion is defined as "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource," disregarding the success or failure of those actions.

The dictionary definition of the word *intrusion* [36] does not include the concept of an insider abusing his or her privileges to perform unauthorized actions, or attempting to do so. A more accurate phrase to use is *intrusion and insider abuse detection*. In this document I use the term *intrusion* to mean both intrusion and insider abuse.

I also use the categorization of intrusion detection as provided by Kumar [29]:

Signature-based detection: Detection is performed by looking for well-defined patterns of attack that exploit weaknesses in the system. The attack patterns are usually referred to as the "signature" of an intrusion.

This type of detection was called "misuse detection" by Kumar.

Anomaly detection: Detection is performed by looking for anomalous behavior and use of the computer system. Anomaly detection assumes that intrusive activity is a subset of anomalous activity, therefore by detecting the latter it is possible to detect the former.

An intrusion detection system is a computer program that attempts to perform intrusion detection by either signature-based or anomaly detection, or a combination of techniques. An intrusion detection system should preferably perform its task in real time [39].

Intrusion detection systems are usually classified as host-based or network-based [39]. Host-based systems base their decisions on information obtained from a single host (usually audit trails), while network-based systems obtain data by monitoring the traffic in the network to which the hosts are connected.

A *distributed intrusion detection system* is one where data is both collected and analyzed in a distributed fashion, as opposed to a *centralized intrusion detection system* , in which data may be collected in a distributed fashion, but is processed centrally. A distributed intrusion detection system may use host- or network-based techniques, or a combination of the two.

The definition of an intrusion detection system does not include preventing the intrusion from occurring, only detecting it and reporting it to an operator. There are some intrusion detection systems that try to react when they detect an unauthorized action occurring. This reaction usually includes trying to contain or stop the damage, for example, by terminating a network connection.

2.2 Desirable characteristics of an intrusion detection system

Crosbie and Spafford [13] define the following characteristics as desirable for an intrusion detection system:

- It must *run continually* with minimal human supervision.
- It must be *fault tolerant* by being able to recover from system crashes, either accidental or caused by malicious activity. Upon startup, the intrusion detection system must be able to recover its previous state and resume its operation unaffected.
- It must *resist subversion.* The intrusion detection system must be able to monitor itself and detect if it has been modified by an attacker.
- It must impose a *minimal overhead* on the system where it is running, to avoid interfering with the system's normal operation.
- It must be *configurable* according to the security policies of the system that is being monitored.
- It must be *adaptable* to changes in system and user behavior over time. For example, new applications being installed, users changing from one activity to another or new resources being available can cause changes in system use patterns.

As the number of systems to be monitored increases and the chances of attacks increase we also consider the following characteristics as desirable:

- It must be *scalable* to monitor a large number of hosts while providing results in a timely and accurate manner.
- It must provide *graceful degradation of service.* If some components of the intrusion detection system stop working for any reason, the rest of them should be affected as little as possible.
- It must allow *dynamic reconfiguration*, allowing the administrator to make changes in its configuration without the need to restart the whole intrusion detection system.

3 Related work

The first work performed on intrusion detection is commonly considered to be the one reported by Anderson [1], which introduced, among others, the idea of doing anomaly detection by creating profiles of normal use and detecting deviations from those profiles. This idea was later formally presented by Denning [14] in what is considered to be the seminal paper for modern intrusion detection.

In the area of host-based intrusion detection there has been substantial work using different methods for analyzing data generated by the host. One of the first host-based intrusion detection systems implemented was IDES [15, 16, 31, 34], which used both a statistical detection engine based on Denning's model [14] and a rule-based expert system for detecting known intrusions by their signatures [35]. Kumar [29] used pattern matching techniques to detect and classify attacks. More recently, Forrest et al. [18] have applied classification techniques to sequences of Unix system calls to identify anomalous behavior in Unix processes. Also, Lane and Brodley [30] have used classification of command sequences to perform automatic user identification. Lunt [32, 33] has surveyed the most common host-based intrusion detection and audit trail analysis techniques.

The area of network-based intrusion detection has also seen a good amount of work. One of the first implemented network-based intrusion detection system was the Network Security Monitor (NSM) [23], which applies both statistical models and rule-based detection to network connections, using the traffic source, destination and service as relevant features for classification. One system that is interesting because it has been in use in a production environment for several years is NADIR [24], deployed at the Los Alamos National Laboratory, and which also uses both statistical and rule-based methods to analyze data collected at a number of different places within LANL's network. A survey of some existing network-based intrusion detection systems has been done by Mukherjee, Heberlein, and Levitt [39].

In distributed intrusion detection data is both collected and analyzed in a distributed fashion. One of its earliest exponents is DIDS [44, 45], which uses some NSM components and has the ability to do both local and global analysis of the data. At the local level it uses both statistical and rule-based detection, and at the global level it uses a rule-based expert system. In this sense, DIDS can be described as a number of host-based and network-based intrusion detection systems that can communicate and share results with one another. This is the form of most intrusion detection systems that call themselves distributed: the same techniques used in host-based and network-based intrusion detection systems are used, but the results are shared and can be analyzed at different levels.

Other examples of distributed intrusion detection systems are GrIDS [46], ASAX [21, 38], EMERALD [41], AAFID [2, 12] and a system proposed by Barnett and Vu [3]. All of these systems use different sources for data and different mechanisms for analyzing it. However, they share a similar general structure: a hierarchical arrangement where host-local components perform some part of the work and relay their results to components higher in the hierarchy. This continues until the partial results reach the top-level components, which have a network-wide view of the systems. Because all these systems ultimately depend on a centralized component, we could argue that they are not truly distributed [17].

Furthermore, even these distributed intrusion detection systems operate based on "heavy" host components, usually implemented as separate processes. These components, as those in all the intrusion detection systems previously mentioned, are subject to tampering and disabling by an intruder, and impose a continuous extra load on the system being monitored.

The problem of analyzing data coming from many different sources is not new. It has been the subject of study for several years in the physical world. For example, the military have studied for a long time the problem of making sense of data coming simultaneously from different points in a battlefield. This is the subject of study of a field called Multisensor Data Fusion (MDF). It was not until recently that this field has been related to intrusion detection [4, 5]. One of the objectives of MDF is the creation of *situational awareness*. The application of MDF to intrusion detection promises to create intrusion detection systems that do not only follow certain analysis techniques on the data they collect, but that can infer knowledge about their environment and the threats they face.

Machine learning, data mining and artificial intelligence techniques such as clustering, classifier trees and neural networks can be applied in MDF for knowledge discovery. However, it is not the first time that these techniques have been applied to the field of intrusion detection. Automatic classification techniques have been applied to the analysis of sequences of events [18, 30], with good success rates, and Tan [47] used neural networks to classify network traffic. Frank [20] has surveyed some of these techniques and their possible uses in intrusion detection.

In the area of innovative intrusion detection techniques, research has been made lately by Hofmeyr [26] on "adaptive intrusion detection", which uses an analogy to the human immune system to provide a highly distributed intrusion detection system that can detect sequences of events that do not belong to the normal behavior of the system (called "self"). This system is interesting because it requires little communication between the hosts. Hofmeyr provides some theoretical results that show that under certain assumptions, a distributed intrusion detection system that requires communication between its components is less robust than one that does not require communication. In this model, a system is considered robust if it does not produce false positives and does not completely fail to detect intrusions. This model builds upon previous work [18, 19], and although its demonstrated application uses a specific type of event for intrusion detection, in general it can be applied to sequences of any type of events.

4 Thesis proposal

I propose to work on determining the validity of the following two thesis hypotheses:

1. It is possible to instrument sensors in a computer system in a way that is useful to perform intrusion detection. The sensors can have limited processing and logging capabilities to avoid disturbing the normal

operation of the system.

2. A group of sensors designed for detecting known intrusions can also be used to detect new ones. This assumes that a large enough number of sensors exist, and that the sensors have been properly designed.

If these hypotheses are valid, their verification would contribute to the design of lighter, more distributed and more resilient intrusion detection systems (first hypothesis) and to determine the types of data that need to be collected to detect certain types of intrusions (second hypothesis).

In these hypotheses the following terms are used:

- *Intrusion detection* is defined as in Section 2.1, to mean "intrusion and insider abuse detection."
- A *sensor* is defined as a piece of software (conceivably aided by some hardware component) that monitors a specific variable, activity or condition of a host. Because the sensor monitors the system directly and not through an audit trail or through packets on a network, we say that it performs *target monitoring*. The sensor's observations are reported as numeric values, and can be discrete or continuous.
- *Limited logging and processing capabilities* means that each sensor can do some processing on the values it observes (for example, range comparisons, normalizations or other simple transformations). The sensor may also keep a limited amount of state information. Its logging capabilities are limited to reporting the value (possibly after a transformation) of its observation. The sensor may also decide not to report a specific observation.
- The definitions of *large enough number* and *proper design* of sensors are addressed in Sections 6.1 and 5.1, respectively.

5 Proposed work

One or more Unix systems will be instrumented with sensors designed to detect specific known attacks. One possible source of information about such attacks is the CVE (Common Vulnerability and Exposure enumeration) [37] from MITRE. Once a sufficient number of sensors is implemented and tested, new attacks will be performed against the instrumented system to determine if and how the existing sensors will react to the new attacks. Some analysis is going to be performed on the data produced by the sensors under the new attacks, as well as on the patterns of sensors triggered by the attacks.

The work will be divided into the following phases:

1. Implementation of base mechanisms for the implementation of the sensors.
2. Implementation of sensors based on a list of existing attacks. See Section 5.1 for details on how the sensors will be built.
3. After an initial population of sensors is built, it will be tested using new attacks. This will be done by monitoring the usual sources of information for new security attacks (such as CERT Advisories [8], BugTraq [6] and the SecurityFocus database [43]) and executing those attacks against the instrumented systems. The responses from the sensors will be recorded for further analysis.
4. Analysis of the data obtained. The study of different analysis techniques is not the main focus of this work, but some analysis will be done to aid in showing the validity of the hypotheses proposed.

5.1 Design of sensors

Sensors will initially be built to test for the specific conditions of the attack they are designed to detect, so that the triggering of a sensor will indicate the occurrence of the corresponding attack. However, I expect that as more sensors are implemented some patterns will be detected in their construction. This might allow sensors to evolve towards monitoring more general types of information that, although still designed for detecting a specific attack, may be useful for detecting other types of attacks.

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Average
9	14	10	4	21	22	18	21	24	14	27	16.73

Table 1: Per-month and average number of new Unix-related vulnerabilities entered into the SecurityFocus vulnerability database from January to November of 1999. The list used for this count was obtained from the URL `http://www.securityfocus.com/vdb/middle.html?vendor=&title=&version=any`. From the full list all vulnerabilities related to non-Unix and closed-source products were removed, to leave the vulnerabilities for which it would be possible to produce sensors.

An example may clarify and justify this claim. Let us assume that a sensor is built to detect a specific buffer-overflow attack in **sendmail**. This sensor may be a piece of code inserted immediately before the affected buffer is used, to measure the length of the string being copied into it. Now assume that later a sensor for another buffer overflow attack on **sendmail** is being implemented, and while that is being done, it is discovered that the attack occurs on the same buffer, but in a different place in the code. In this case, the two sensors may be fused into one by moving the check to the location where the offending data enters the program. Sensors may also be split, for example, into a general sensor that detects a buffer overflow attack, and more specialized sensors that detect the specific attack that is taking place.

Note that this sensor "fusion and splitting" process is not guaranteed to occur, and it is not necessary for the success of this project. I expect the insight and considerations about the design of sensors to be one of the main contributions of this work.

6 Completion conditions

The completion conditions for each phase (as described in Section 5) are defined as follows:

1. This phase will be completed when a base set of mechanisms (such as system calls) are implemented that can be used to instrument sensors both in the kernel and in the applications of the target systems. The minimum such set is a call to perform logging of events. However, as the work progresses, the convenience or need of other mechanisms may become apparent. The logging mechanisms will likely be implemented in a way that allows interaction with the AAFID system (see Section 9.1).

2. This phase will be completed when a large enough number of sensors have been built. See Section 5.1 for more details on how the sensors will be built. See Section 6.1 for the precise definition of "large enough" that I will use.

   The completion of this phase will provide the necessary evidence to support the validity of the first thesis hypothesis proposed in Section 4.

3. This phase will be complete when 35 new attacks have been tried against the system. According to the SecurityFocus vulnerability database [43], the monthly number of new vulnerabilities reported from January to November of 1999 is as shown in Table 1. According to these statistics, the number of 35 attacks was chosen to represent approximately two months of new attacks.

4. Although the exhaustive study of analysis methods is not the focus of this thesis, some interpretation and analysis will be performed to verify the validity or incorrectness of the second thesis hypothesis. This phase will be complete when enough results are collected to provide substantial evidence in favor or against the statement that new attacks can be detected using the existing sensors. In this respect, I will consider "substantial evidence" as one of the following:

   - Finding at least one analysis technique that, using the data produced by the sensors, is able to detect a significant percentage of the new attacks attempted against the system.
   - Finding that none of the analysis techniques used (the techniques to be used and their number has not been determined) can detect a significant percentage of the new attacks performed against the systems. This could be interpreted as evidence that the second hypothesis is not true.

Part of the analysis will also be a study of which sensors are more significant in detecting attacks. It may be possible to apply feature selection techniques similar to the ones used in the machine learning field [e.g. 27] to determine which of the existing sensors are more important for the detection of new intrusions.

6.1 Number of sensors needed

Sensors will be built based on the vulnerability and attack list provided by the CVE [37]. This list is not a taxonomy or a classification scheme, and is used only as a fairly complete list of known vulnerabilities.

Frequently, the discovery and publication of an attack triggers exploration in the same direction or in the same program, and causes the discovery of similar or related problems. For this reason, attacks that are published close in time may be correlated. To avoid these correlations and get a representative sample of the "attack space"[1], attacks will be selected from the list randomly without replacement.

This process will continue until the last 15 attacks selected from the list are already detected by previously implemented sensors (following the sensor implementation design considerations described in Section 5.1), or until the CVE list is exhausted. The last version (September 9, 1999) of the CVE list contains 321 entries. From this list, all the entries for non-Unix and closed-source programs were removed, leaving 226 entries. The number 15 was chosen to represent approximately 5% of this list. The criterion behind this is that if we start finding redundancy in the sensors needed, it is time to move on to verifying the second hypothesis.

6.2 Implementation platform

The "attack space" is considerably large, because each attack can be characterized by a number of features, as shown by Krsul [28]. To simplify the practical study of this space, I will eliminate one of the features with the most variance: the operating system and platform of the attack. In the current version of the CERIAS vulnerability database [7], of the eighteen multiple-choice features defined, "operating system" has originally 47 defined values, making it the second largest one. Of these values, 31 correspond to Unix-style systems.

To reduce the number of different systems that have to be instrumented, all the sensors will be implemented on the same platform. Because sensors will be designed to detect the attack attempts and not their success, the attack does not need to exist in the implementation platform anymore, and in fact may be an attack for a completely different platform. In this sense, the instrumented systems will work like *universal honeypots*[2] because they will be able to accept and monitor attacks for different platforms.

7 Expected results

According to the two thesis hypotheses on which this work will be based, the following two results are expected:

Hypothesis 1: To find that is is possible to build sensors into a system in a way that allows the detection of known intrusions. Furthermore, to find that some of the sensors can be general enough to work for more than one intrusion.

Hypothesis 2: To find that once a large enough number of sensors is built, the existing sensors will be triggered when new intrusions are attempted against the system. I expect to find that the sensors will be triggered in ways that will make it possible to detect the new intrusions.

8 Contributions

The first contribution of my work (from the first thesis hypothesis) will be to show whether it is possible to perform intrusion detection using sensors embedded in a computer system. If this is shown to be possible, it will provide a step towards the design of intrusion detection systems that are lightweight and that do not affect the normal operation of the systems being monitored. Also, by their nature, the sensors are more resistant to tampering

[1] The term "attack space" is used loosely in this context to represent all the existing attacks and vulnerabilities. My work is not tied to any particular classification or representation of this space. One such classification scheme was provided by Krsul [28].

[2] For an interesting application of honeypots, see the paper by Cheswick [10]. The term "universal honeypot" was coined by Benjamin Kuperman from CERIAS, to the best of my knowledge.

and disabling by an intruder, making the intrusion detection system more resistant to attacks performed against itself.

Most of the host-based intrusion detection systems that exist today [e.g. 11, 25, 31] have been designed around the data that the operating system makes available in the form of audit trails, process accounting logs, etc. Another contribution of my work will be to show whether it is possible to design an intrusion detection system the other way around: by deciding which data we want to collect and implementing the mechanisms to collect it. By going into the operating system and into the affected programs to instrument the appropriate sensors, we can monitor any activity that takes place in the system, and obtain any information that may contribute to the detection of an attack. Furthermore, by instrumenting the sensors into the affected code itself, we are doing *target monitoring*. This is, we are monitoring the activities of the object under study directly, instead of indirectly through audit trails. Target monitoring has the advantage of reducing the risk of an intruder modifying the information before it gets to the sensor.

If it can be shown that it is possible to detect new attacks using a set of sensors designed for detecting known attacks (second thesis hypothesis), this would provide insight for determining, at least experimentally, the types of data that need to be collected to detect certain types of intrusions.

If I find that the second thesis hypothesis cannot be substantiated, this would also be a useful result because it would provide evidence that there are no general types of data that can be collected to detect new attacks, and that each new attack may need the development of specialized sensors for its detection.

Independently of the results of the two proposed hypotheses, my work will also provide the following contributions:

- Through the process of implementing sensors, it will provide insight into the considerations of how to design and build these sensors to make them efficient and reliable. The experience obtained by building the sensors for this project will be a significant contribution to the intrusion detection field in general.
- Through the process of implementing and deploying the sensors, it will provide experience about proper placement of the sensors in the computer systems. The placement of the sensors may affect their impact on system performance, as well as their reliability and resistance to tampering.
- For the purposes of better understanding the issues involved in this work, I have started to develop a functional model of intrusion detection (see Section 9.4.1). By itself, this could be a contribution by providing a clearer understanding of the general intrusion detection process, how the existing intrusion detection systems fit into it, and how to make new intrusion detection systems satisfy the characteristics described in Section 2.2.

9 Current state

In the recent months I have been working on different aspects that serve as foundation and preliminary research for the work proposed in this document.

9.1 Development of infrastructure

The AAFID (Autonomous Agents for Intrusion Detection) [2] architecture has been developed at CERIAS, together with a prototype that implements all the essential aspects of the architecture. The AAFID architecture establishes a hierarchical structure for the collection, transmission and analysis of data using entities known as agents (data collectors), transceivers (host-local data analyzers) and monitors (network-wide data analyzers). The AAFID architecture is general enough to provide the infrastructure necessary for implementing different types of intrusion detection systems. I will use the AAFID prototype as a supporting platform for the collection of data necessary for my work.

9.2 Selection of implementation platform

The implementation platform for the sensors has already been selected. The operating system selected is OpenBSD [40], one of the three main variations of the free operating systems based on the BSD distribution. This operating system was chosen for the following reasons:

- It is an open-source operating system, which makes it easy to instrument the sensors both in the kernel and in the system programs.

- The source tree is centrally managed and distributed as a single directory tree. This makes it more manageable than Linux, for example, where source code for different components of the operating system are distributed as separate packages. The source tree of OpenBSD closely mimics the layout of the system itself, making it easy to locate the code for different programs and subsystems.

- The OpenBSD project is known for its attention to security, and is the first operating system that has gone through an extensive security audit process. Most of the security problems for which sensors will be implemented have already been fixed in OpenBSD. By looking at the security patches and at the change log for each file it may be easier to locate the portions of code where the problems existed. This could help in determining where the sensors for each intrusion have to be placed. Also, because the problems themselves no longer exist, it will be easier to try attacks against the instrumented system without worrying about the adverse effects they could have on the system.

- It runs on multiple architectures, including low-end Intel and Sparc systems, which may make it easier to set up a network of instrumented systems for the realization of the proposed experiments.

9.3 Study of data analysis techniques for intrusion detection

I have done some preliminary work in the analysis of data produced by AAFID agents [48]. The data collected by AAFID agents was analyzed using clustering techniques, and the results are promising, showing that the clustering assignments of the data points changed during periods in which attacks were being performed against the machines in the network, and remained mostly stable otherwise. These are preliminary results and work is still in progress in this respect.

Although data analysis is not the main aspect of the work I propose, the results of using clustering for analyzing AAFID data show that it might be possible to use novel analysis techniques to centrally analyze the data collected by distributed sensors. This is can be related to my first thesis hypothesis.

9.4 Development of models for intrusion detection

One difficulty for theoretical research in intrusion detection has been the lack of proper models for the design, deployment and use of an intrusion detection system. Most of the past intrusion detection work focuses on how intrusions will be detected. While this is the main objective of an intrusion detection system, the detection task is based on decisions about what will be detected, how data will be collected, and what will be done with the results of the analysis. In most existing intrusion detection work these decisions have been made implicitly by the authors or the designers.

I have started working on developing a model that will serve as a framework for subsequent analysis and study. My claim is that all existing intrusion detection systems follow (either implicitly or explicitly) this model. I also present some discussion about how different models may be needed for representing the design process of intrusion detection systems.

9.4.1 A functional model for intrusion detection

This model represents how intrusion detection systems are conceptually designed, implemented, deployed and used. It is divided into several phases. Note that the problems and issues discussed are general problems that need to be addressed, and I will not address all of these problems in my work. The main reason for the development of this model is to further my understanding of the intrusion detection process.

1. *Design and Planning:* This phase includes planning of the features that the intrusion detection system will have in terms of its detection capabilities and the way in which they will be deployed to make more efficient use of the resources available.

   Completion of this phase includes answering the following questions:

(a) What types of intrusions must the intrusion detection system detect? (determining the detection requirements)

(b) What types of data need to be collected for the intrusion detection system to detect the intended types of problems? (determining the data collection needs)

(c) How are the detection requirements and data collection needs expressed, and how are they dependent upon one another?

(d) How are data collection and data analysis components organized and deployed?

In all existing intrusion detection systems this is an off-line phase because it occurs before the intrusion detection system starts running. However, the planning and deployment phase could also occur on-line (while the intrusion detection system is running), giving the intrusion detection system the capability of dynamically re-analyzing its needs and capabilities and re-deploying (even re-creating) components as needed.

2. *Data Collection:* This phase includes collecting the data that will be used to look for intrusions. This is the first on-line phase, which means that it occurs during the execution of the intrusion detection system.

   Successful implementation of this phase includes answering the following questions:

   (a) How is the information about data collection needs obtained? This is, how to let the data collection mechanisms know about the data collection needs determined in the previous phase? The answer to this question is important if we want to implement an intrusion detection system that is able to modify its data collection capabilities as part of a dynamic response to its environment. In most existing intrusion detection systems, this question is answered off-line, and the information about which data needs to be collected is hard-coded in the data collection mechanisms themselves.

   (b) How and where is the data collected? Different types of data may be better collected at different parts in the host machine or the network. For example, it may be more efficient to collect low level process data at some internal point in the operating system, whereas the log of commands executed by a user needs to be collected at the user level.

   (c) How is the data represented? Using a common format for the data may be useful for processing (and is what most existing intrusion detection systems do), but it may also impose an unnecessary overhead in the data collection mechanisms. Furthermore, it may not be feasible to efficiently represent different types of data using a common format.

   (d) Where and how is the data stored for later use?

   (e) How is the data pre-processed or filtered? It may be useful to do some form of pre-processing on the data before passing it to the data analysis mechanisms.

   (f) How is the data sent to the data analysis phase?

3. *Data Analysis:* In this phase, the data collected is processed and results are produced. This is the phase that performs the intrusion detection, and is possibly the most complex of the on-line phases.

   To correctly implement this phase, the following questions must be answered:

   (a) How is the data obtained from the data collection phase? In most existing intrusion detection systems this question is answered off-line by the intrusion detection system designers and the data transfer mechanisms are hard-coded in the implementation of the intrusion detection system.

   (b) Which data analysis mechanisms to use? Each existing method has its own strengths and weaknesses. With the increasing importance of distributed intrusion detection systems, new data analysis mechanisms may be needed to process different types of data coming from different hosts.

   (c) Where and how is the analysis implemented? The data analysis may be performed at a single place (centralized) or in different places (distributed). However, it is not clear under which circumstances it may be better to use one or the other. Can they be combined? If distributed analysis is used, the question remains of how to best communicate between the distributed components to exchange information.

(d) How and where are the results stored for later use?

(e) How is the analysis verified? If the intrusion detection system is compromised by an attacker, is it possible to detect this? Is there a way to verify the results so that trust in the intrusion detection system can be increased?

4. *Presentation of Results:* This is the phase where the results from the data analysis phase are presented to the user. As such, it has to answer the following questions:

   (a) How to get the results from the data analysis phase in a manner that is both efficient and secure?

   (b) How to present the data to the user so that it can be used to make accurate and timely decisions? This is a data visualization and reduction problem.

9.4.2 Design models for intrusion detection

The model described in Section 9.4.1 is a functional model that shows the logical flow of decisions and actions in the design, implementation, deployment and use of an intrusion detection system.

However, in practice, the design of an intrusion detection system may not follow the functional model. In most cases, the designers of the intrusion detection system face constraints imposed by the environment in which the intrusion detection system is going to operate, or by policy. For example, the system described by Anderson [1] was limited to operate on data obtained from the pre-existing audit mechanisms in the operating system. Using these data, the intrusion detection system had to perform its functions as best as possible. Therefore the data available influences or determines the detection capabilities, the analysis techniques, and other factors of the implementation, deployment and use of the intrusion detection system. I call this a *data-driven design*.

In other cases, the intrusion detection system may be built with the explicit purpose of testing a specific data analysis technique (for example, a neural network, as described by Tan [47]). In this case, the analysis technique will dictate the data that is needed, where it needs to be collected, and other factors. I call this a *analysis-driven design*.

These models may be useful for further understanding the design process of an intrusion detection system, and may be fully developed and described in more detail as my work progresses.

10 Future work

The following are some of the issues that I do not plan to address on my work, but that could be based on it for future research:

- Developing a formal description or analysis of the types of data that need to be collected to detect certain types of attacks. If my second thesis hypothesis can be determined to be true, a future step could be to develop a classification for sensor data, and use it to classify the most relevant sensors.

- Based on the previous item, it might be possible to link the sensor classification scheme to one or more of the existing vulnerability classification schemes [e.g. 28]. This could allow for some interesting capabilities, such as:

  – Semi-automatic selection of the sensors necessary to monitor for certain vulnerabilities;

  – If each sensor had resource usage information, it could be used to optimize the use of the resources available to the intrusion detection system.

- If it is possible to use embedded sensors to detect intrusions on the victim system, it might be interesting to study the possibility of using sensors to detect the origination of an attack in the source system. This may only be possible for some types of attacks, and has other concerns such as user privacy that would need to be explored.

References

[1] J. P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, James P. Anderson Co., Fort Washington, PA, April 1980.

[2] Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez, Eugene Spafford, and Diego Zamboni. An architecture for intrusion detection using autonomous agents. Technical Report 98-05, COAST Laboratory, Purdue University, May 1998. URL `ftp://coast.cs.purdue.edu/pub/COAST/papers/diego-zamboni/zamboni9805.ps`.

[3] Bruce Barnett and Dai N. Vu. Vulnerability assessment and intrusion detection with dynamic software agents. In *Proceedings of the Software Technology Conference*, April 1997.

[4] Tim Bass. Multisensor data fusion for next generation distributed intrusion detection systems. In *Proceedings of the IRIS National Symposium on Sensor and Data Fusion*, May 1999. URL `http://www.silkroad.com/papers/html/iris/`.

[5] Tim Bass. Intrusion detection systems & multisensor data fusion: Creating cyberspace situational awareness. *Communications of the ACM*, April 2000. URL `http://www.silkroad.com/papers/acm.fusion.ids.ps`.

[6] BugTraq. Mailing list archive. Web page at `http://www.securityfocus.com/`, 1999–2000.

[7] Center for Education and Research in Information Assurance and Security. Vulnerability database. Stored in the CERIAS systems under `/u/coast4/proj-vdb`, 1999. For availability information, contact `spaf@cerias.purdue.edu`.

[8] CERT Coordination Center. CERT advisories. Web page at `http://www.cert.org/advisories/`, December 1999.

[9] CERT Coordination Center. CERT/CC statistics. Web page at `http://www.cert.org/stats/cert_stats.html`, 1999.

[10] Bill Cheswick. An evening with berferd: In which a cracker is lured, endured, and studied. In *Proceedings of the Winter 1992 Usenix conference*. Usenix, 1992. URL `http://cm.bell-labs.com/who/ches/papers/berferd.ps`.

[11] Mark Crosbie, Bryn Dole, Todd Ellis, Ivan Krsul, and Eugene Spafford. IDIOT—users guide. CSD-TR 96-050, COAST Laboratory, Purdue University, 1398 Computer Science Building, West Lafayette, IN 47907-1398, September 1996. URL `http://www.cerias.purdue.edu/techreports/public/96-04.ps`.

[12] Mark Crosbie and Eugene Spafford. Defending a computer system using autonomous agents. In *Proceedings of the 18th National Information Systems Security Conference*, pages –, Oct 1995. URL `http://www.best.com/~mcrosbie/Research/NISSC95.ps`.

[13] Mark Crosbie and Gene Spafford. Active defense of a computer system using autonomous agents. Technical Report 95-008, COAST Group, Department of Computer Sciences, Purdue University, West Lafayette, IN 47907-1398, Feb 1995. URL `http://www.cerias.purdue.edu/homes/spaf/tech-reps/9508.ps`.

[14] Dorothy E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, 13(2): 222–232, February 1987.

[15] Dorothy E. Denning, D. L. Edwards, R. Jagannathan, T. F. Lunt, and P. G. Neumann. A Prototype IDES— A Real-Time Intrusion Detection Expert System. Technical report, Computer Science Laboratory, SRI International, 1987.

[16] Dorothy E. Denning and Peter G. Neumann. Requirements and Model for IDES – A Real-Time Intrusion Detection System. Technical report, Computer Science Laboratory, SRI International, August 1985.

[17] Stephanie Forrest. Personal communication, 1999. Department of Computer Sciences, University of New Mexico.

[18] Stephanie Forrest, Steven Hofmeyr, Anil Somayaji, and Thomas Longstaff. A sense of self for Unix processes. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*. IEEE Computer Press, 1996. URL `ftp://ftp.cs.unm.edu/pub/forrest/ieee-sp-96-unix.ps`.

[19] Stephanie Forrest, Steven A. Hofmeyr, and Anil Somayaji. Computer Immunology. *Communications of the ACM*, 40(10):88–96, October 1997. ISSN 0001-0782. URL `http://www.acm.org/pubs/citations/journals/cacm/1997-40-10/p88-forrest/`.

[20] Jeremy Frank. Artificial intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th National Computer Security Conference*, Baltimore, MD, October 1994. URL `http://seclab.cs.ucdavis.edu/papers/ncsc.94.ps`.

[21] Naji Habra, B. Le Charlier, A. Mounji, and I. Mathieu. ASAX: Software Architecture and Rule-based Language for Universal Audit Trail Analysis. In *Proceedings of ESORICS 92*, Toulouse, France, November 1992. URL `ftp://coast.cs.purdue.edu/pub/doc/intrusion_detection/HabraCharlierEtAl92.ps`.

[22] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The Architecture of a Network Level Intrusion Detection System. Technical report, University of New Mexico, Department of Computer Science, August 1990.

[23] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A Network Security Monitor. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, May 1990.

[24] Judith Hochberg, Kathleen Jackson, Cathy Stallings, J. F. McClary, David DuBois, and Josephine Ford. NADIR: An automated system for detecting network intrusion and misuse. *Computers and Security*, 12(3): 235–248, May 1993.

[25] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6:151–180, 1998.

[26] Steven Andrew Hofmeyr. *An Immunological Model of Distributed Detection and Its Application to Computer Security*. PhD thesis, University of New Mexico, May 1999. URL `ftp://coast.cs.purdue.edu/pub/doc/intrusion_detection/hofmeyer-distributed-detection.ps.gz`.

[27] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proc. 11th International Conference on Machine Learning*, pages 121–129. Morgan Kaufmann, 1994.

[28] Ivan Krsul. *Software Vulnerability Analysis*. PhD thesis, Purdue University, 1998. URL `ftp://coast.cs.purdue.edu/pub/COAST/papers/ivan-krsul/krsul-phd-thesis.ps.Z`.

[29] Sandeep Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, West Lafayette, IN 47907, 1995. URL `ftp://coast.cs.purdue.edu/pub/COAST/papers/sandeep-kumar/kumar-intdet-phddiss.ps.Z`.

[30] Terran Lane and Carla E. Brodley. Temporal sequence learning and data reduction for anomaly detection. In *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, pages 150–158. ACM, 1998. URL `http://mow.ecn.purdue.edu/~terran/facts/research/pubs/acm_ccs98.ps`.

[31] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. G. Neumann, H. S. Javitz, A. Valdes, and T. D. Garvey. A Real-Time Intrusion Detection Expert System (IDES) – Final Technical Report. Technical report, SRI Computer Science Laboratory, SRI International, Menlo Park, CA, February 1992.

[32] Teresa F. Lunt. Automated Audit Trail Analysis and Intrusion Detection: A Survey. In *Proceedings of the 11th National Computer Security Conference*, October 1988.

[33] Teresa F. Lunt. A Survey of Intrusion Detection Techniques. *Computers & Security*, 12(4):405–418, June 1993.

[34] Teresa F. Lunt, R. Jagannathan, Rosanna Lee, Sherry Listgarten, D. L. Edwards, P. G. Neumann, H. S. Javitz, and A. Valdes. Development and Application of IDES: A Real-Time Intrusion-Detection Expert System. Technical report, SRI International, 1988.

[35] Teresa F. Lunt, R. Jagannathan, Rosanna Lee, Alan Whitehurst, and Sherry Listgarten. Knowledge based Intrusion Detection. In *Proceedings of the Annual AI Systems in Government Conference*, Washington, DC, March 1989.

[36] Merriam-Webster. “intrusion”. Merriam-Webster OnLine: WWWebster Dictionary. `http://www.m-w.com/dictionary`, 1998. Accessed on May 16, 1998.

[37] MITRE. Common vulnerabilities and exposures. Web page at `http://cve.mitre.org`, 1999–2000.

[38] Abdelaziz Mounji, Baudouin Le Charlier, Denis Zampunieris, and Naji Habra. Distributed audit trail analysis. Technical Report RP-94-007, Institut d’Informatique, FUNDP, Rue Grandgagnage 21, Namur, Belgium, November 1994. URL `ftp://coast.cs.purdue.edu/pub/doc/intrusion_detection/MounjiCharlierEtAl94.ps.gz`.

[39] Biswanath Mukherjee, Todd L. Heberlein, and Karl N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, May/June 1994.

[40] OpenBSD. Web page at `http://www.openbsd.org/`, 1999–2000.

[41] Phillip A. Porras and Peter G. Neumann. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th National Information Systems Security Conference*, pages 353–365. National Institute of Standards and Technology, 1997.

[42] Richard Power. 1999 CSI/FBI computer crime and security survey. *Computer Security Journal*, Volume XV (2), 1999.

[43] SecurityFocus. Web page at `http://www.securityfocus.com/`, 1999–2000.

[44] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, Tim Grance, L. Todd Heberlein, Che-Lin Ho, Karl N. Levitt, Biswanath Mukherjee, Douglass L. Mansur, Kenneth L. Pon, and Stephen E. Smaha. A System for Distributed Intrusion Detection. In *Proceedings of COMPCON Spring ’91, 36th IEEE Computer Society International Conference*, pages 170–176, San Francisco, CA, February 25 – March 1 1991. IEEE Computer Society, IEEE, IEEE Service Center, Piscataway, NJ.

[45] Steven R. Snapp, James Brentano, Gihan V. Dias, Terrance L. Goan, L. Todd Heberlein, Che lin Ho, Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (distributed intrusion detection system) - motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176, Washington, DC, October 1991. National Institute of Standards and Technology.

[46] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS: A graph based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, volume 1, pages 361–370. National Institute of Standards and Technology, October 1996.

[47] Kymie Tan. An application of neural networks to UNIX computer security. In *Proceedings of the IEEE International Conference on Neural Networks*, November 1995. URL `ftp://coast.cs.purdue.edu/pub/doc/intrusion_detection/neuralnetworks.ps`.

[48] Diego Zamboni. Using clustering to detect abnormal behavior in a distributed intrusion detection system. Under preparation, draft available by request, 1999.