SIMS: A Modeling and Simulation Platform for Intrusion Monitoring/Detection Systems *

Ashish Garg, Shambhu Upadhyaya, Ramkumar Chinchani
Dept. of Computer Science and Engineering
University at Buffalo, SUNY
Buffalo, NY 14260 USA
Email: {ashish, shambhu, rc27}@cse.buffalo.edu

Kevin Kwiat
Air Force Research Laboratory
525 Brooks Rd.
Rome, NY 13441 USA
kwiatk@rl.af.mil

Keywords: intrusion detection, simulation tool.

Abstract

Computer security is becoming an ever-growing concern in the current world. Intrusion monitoring and detection systems form a large class of tools that are deployed to combat the scourge of attacks. While there are many such systems available, the typical development cycle of such a system involves significant time and effort. In this paper, we propose and develop a modeling and simulation platform constructed over a popular operating system simulator that can aid in the modeling and rapid testing of intrusion detection systems. In order to achieve this goal, we have studied a number of intrusion detection systems and identified the various features that are essential to the successful construction of such a platform. We demonstrate the capabilities of this system, called Modeling and Simulation Platform for Intrusion Monitoring/Detection Systems (SIMS), by developing and simulating two intrusion detection system models. In the long run, we also hope that this would become a widely used academic and commercial tool.

1 INTRODUCTION

The purpose of using modeling and simulation platforms is; (a) to educate people with the emerging technology, (b) to reduce costs in terms of time and labor before deployment of a real system and (c) for rapid development, testing and evaluation of a concept. These platforms not only grant the flexibility of testing large variety of scenarios before they are implemented, but also provide an adequate idea about the feasibility of the system to be implemented [1]. In this paper, we identify an area which is attracting significant attention but greatly lacks in modeling and simulation tools, viz., computer and network systems security.

Security is a major concern for business and service critical systems. In order to address these concerns, a wide variety of security mechanisms such as intrusion detection systems, firewalls, etc., are deployed. A complete security system can take a long time to evolve, sometimes a few years; for example the EMERALD (Event Monitoring Enabling Responses to Anomalous Live Disturbances) system [2] developed at SRI International took 3-5 years to become a robust system. Modeling and simulation platforms, that provide support for rapidly testing the feasibility and evaluating the performance of these security systems before they are deployed, can reduce the time to develop such a system. Although modeling and simulation tools are available for other domains such as computer networks [3, 4, 5], there are none available for testing security systems. In this paper, we focus primarily on modeling and simulating one category of security mechanisms, i.e., intrusion detection. Intrusion detection is an important line of defense since it has the potential to detect and stop attacks occurring on a computer system [6].

Our primary contribution in this paper involves development of a tool for modeling, simulating and testing an intrusion detection system (IDS). This is accomplished by surveying various available IDSs and identifying a set of general features supported by them. These features form the basis of our tool. Our tool primarily serves three purposes; (1) IDS developers can use this tool to simulate and test their intrusion detection models before deploying the real system, (2) educational institutions can use this tool to demonstrate a working IDS with real attack scenarios, without having to compromise the system security and (3) an organization can use the tool to test various IDSs before deploying the one with least overhead on their systems, while providing adequate security. As background, Section 2 of this paper presents the basic terminology of intrusion detection, and surveys various types of IDSs

*Research supported in part by U.S. Air Force Research Laboratory, Rome, NY, under contract: F30602-00-10507

in order to identify the general features. Section 3 explains how the features extracted from various IDSs are integrated in SIMS (A Modeling and Simulation Platform for Intrusion Monitoring/Detection Systems) architecture. Section 4 begins the description of the tested IDS models and attack scenarios, and then discusses the performance evaluation of our tool. Section 5 concludes the paper.

2 BACKGROUND AND MOTIVATION

We present an overview of some of the notable research efforts in the field of intrusion detection and prevention in order to put our work in perspective. Any modeling platform that claims to allow rapid development and testing of intrusion detection system models should provide support for general IDS features. According to the Software Engineering Institute of Carnegie Mellon University, an intrusion is defined as follows:

> "An intrusion has taken place when an attack is considered successful from the victim's perspective, i.e., the victim has experienced some loss or consequence" [7].

In general, an *intrusion* is defined as successfully gaining unauthorized access into a system. An attempt to break into a system without proper authorization is called an *attack*. Intrusion detection is the process of detecting attacks or unauthorized use of a system. The history of intrusion detection dates back to 1980, when Anderson proposed the idea of audit trails for monitoring the threats to computer systems [8]. The events or operations on a computer system are collected in the form of audit data. Almost all IDSs are based on this audit data generated by computer systems. Intrusion detection has been studied in two major classes. They are:

Misuse Detection. Misuse detection is done by detecting "specific, precisely representable techniques of computer system abuse" [9]. An IDS doing misuse detection contains a collection of signatures, which are specifications of features, conditions, arrangements and interrelationships among *events* that signify a break-in or other misuse, or their attempt [10]. The intrusions are detected by searching for these intrusion signatures in the user activities. For example, the signature for a SYN attack is a large number of half open TCP connections [11]. Misuse detection systems (MDSs) are categorized based on the source of audit data. Host-based MDSs analyze audit data from a single host, see Haystack [12] and MIDAS (Multics Intrusion Detection and Alerting System) [13]. On the other hand, network based MDSs detect the intrusions based on the audit data from a collection of hosts in a network, see NIDES (Network Intrusion Detection Expert System) [14] and NSM (Network Security Monitor, now called Network Intrusion Detector or NID) [15].

Anomaly Detection. Anomaly detection is based on the premise that an attack on a computer system (or network) will be noticeably different from normal system (or network) activity, and an intruder (possibly masquerading as a legitimate user) will exhibit a pattern of behavior different from the normal user [16]. IDSs based on this approach attempt to characterize user's normal behavior, often by maintaining statistical profiles of each user's activity [17, 18]. These activity profiles include information about user's computing behavior such as time of login, duration of the login session, disk usage, CPU usage, favorite applications etc.

Some systems [19, 20] use access control to define the bracket of allowed activity (also called as *sandboxing*) based on a *security policy*. The security policy is specified by the information assurance analyst and contains various rules that allow or deny actions or object accesses. The various aspects of this security policy are enforced at the appropriate points in the system. These access control mechanisms prevent the system from being misused by unauthorized people.

The above analysis of various intrusion detection systems helps us in identifying the following standout features of IDSs:

Event Monitoring. An IDS is primarily an event-driven system and thus it requires a comprehensive event generation and management system. Events generated by a system are generally in the form of login failures, file access or any other observable activity. Typically, probes inserted at strategic points in the system produce this audit-data specific to generated events. These generated events are consumed by an event listener or monitoring module, which analyses the audit trails [1] to determine whether an intrusion has occurred or not. This audit data can be used either to detect patterns of misuse or construct a statistical profile for anomaly detection.

Access Control. Intrusion prevention systems use static access control mechanisms to control the access to various system objects. Actions and object accesses that are allowed or disallowed are specified as a security policy and various aspects of this policy are enforced at the appropriate points in the system. For example, file-based access control is enforced in the file system code, process-based access control such as signal delivery is implemented in the process subsystem. Relevant issues in this context are locating proper points in the various subsystems to enforce access control and the correctness of the security policy.

We conclude this section on a comparative note between simulators for operating systems and computer security systems. The evolution of modern operating systems such as Unix [21] is as old as mid 1960's and the modeling/simulation tools for operating systems such as Nachos [22] came as late as early 1990's. On a similar note, while the history of intru-

[1] a stream of events in a system

sion detection dates back to the early 1980's, tools for modeling/simulating the real-world scenarios of intrusion detection are beginning to appear in 2000 [23, 24, 25, 26].

3 ARCHITECTURE OF SIMS

An IDS runs over a host and an operating system. In order for us to be able to model and simulate an IDS, we need to use a simulated operating system kernel. Nachos [22] has been a de-facto standard as an open-source instructional operating system. It simulates a real operating system kernel and hence was well suited to our needs for simulating and testing various IDS models.

Figure 1 shows the preliminary architecture of SIMS. The IDS model features extracted in the previous section are integrated into Nachos. SIMS is essentially this comprehensive package. The IDS models to be simulated and tested run over SIMS. In case of event monitoring, all the subsystems of Nachos send notifications of events to the event management system (EMS), when each event is appropriately redirected to its listener. The EMS also logs these events, (see Figure 1 (a)). EMS receives event listener registration information from an analyst [2] requesting specific events of interest. When these events are generated, EMS notifies the analyst about them.

The access control model for controlling access to system objects and preventing intrusions is implemented by providing the support for ACLs and their enforcement. The access rights or permissions associated with the system objects are specified by the analyst as a security policy. During the execution of user processes, objects are accessed and if an ACL was specified, then it is checked and enforced, (see Figure 1 (b)). The two main components of SIMS are described as follows:

- **Event Management System (EMS)**

  The analyst registers the events of interest with EMS. Probes placed at specific parts of Nachos kernel generate the events when system objects are accessed. When events are generated, corresponding subsystem of Nachos reports them to EMS. The analyst is notified about the events of interest to him. EMS logs all the events along with the information such as the time when the event occurred, the system call made, the object which was accessed and the access check. A sample log file is shown in Table 1. The information gathered by EMS is used for misuse detection as well as for anomaly detection based on the statistics derived from event log.

- **Access Control Mechanisms**

  The analyst can also define various access control mechanisms for system objects. These access control mechanisms are used to control the access to various system objects including files (using access control lists), processes etc. These access control entities are checked by the corresponding subsystem of Nachos to determine whether an object has enough permissions to allow the access request. The subsystem logs this access check information to the log file (see Table 1).

[2] A person seeking to develop and test a specific IDS model.

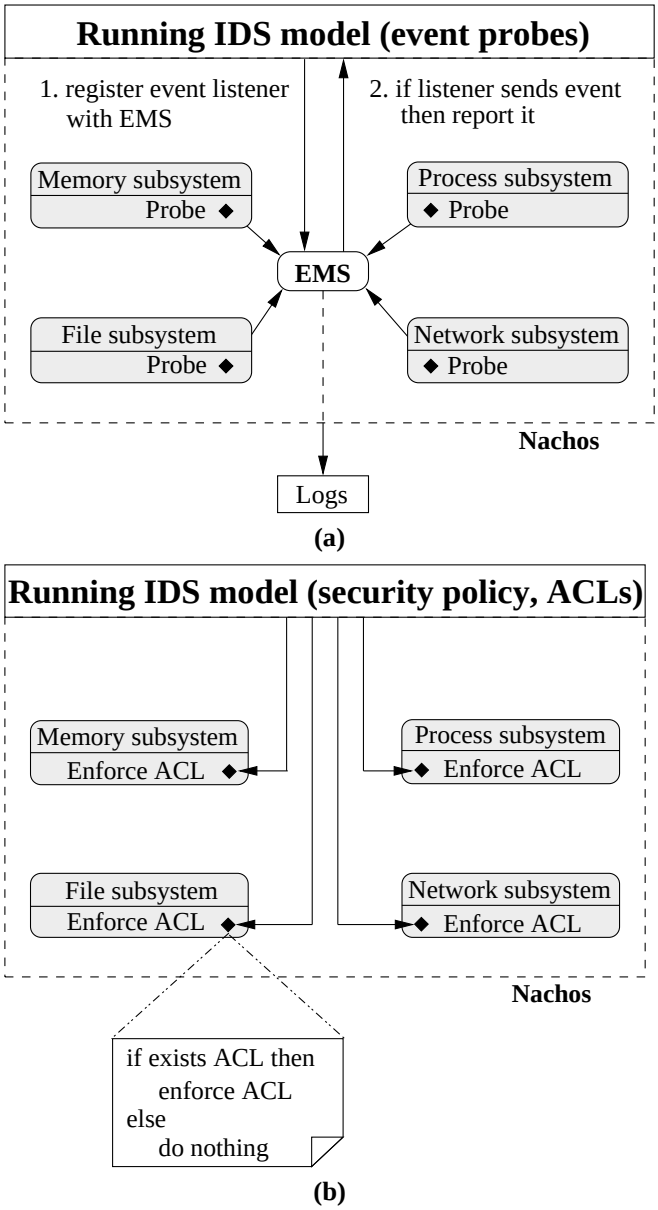

Figure 1: Architecture of SIMS

A user interface (UI) (not shown in Figure 1) is provided to the analyst for communicating with SIMS. The analyst sends the registration information to the EMS and receives the event notifications using this UI. The ACLs for enforcing access control to system objects are also specified by the analyst using

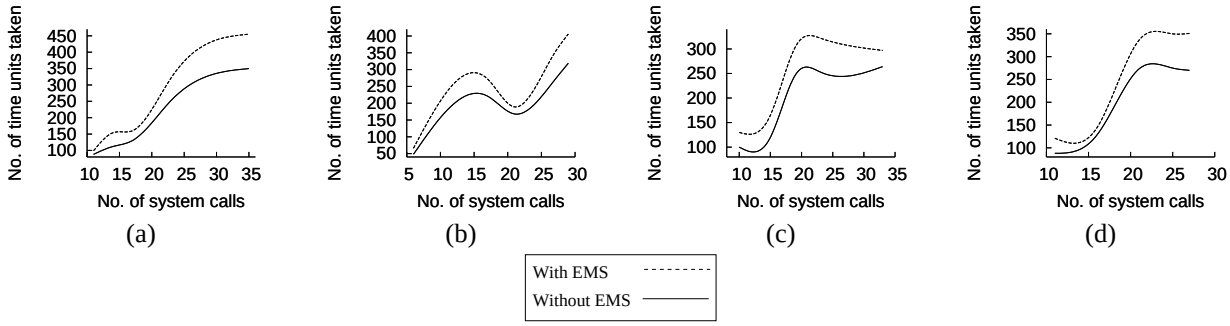

Figure 2: System overhead due to the integration of IDS models

Table 1: A sample log file

Timestamp	System call	Object	Access check
1034217770	open()	test1.txt	Allowed
1034217898	open()	test2.txt	Denied
1034218232	read()	test1.txt	Allowed
1034218439	write()	test1.txt	Denied
1034218732	open()	test3.txt	Allowed
1034219103	read()	test3.txt	Denied

the UI.

4 MODELING AND EXPERIMENTATION

The SIMS tool allows modeling and simulation of various IDSs and attack scenarios. In this section we describe our experimental setup and measure the performance of the tool. We implemented two IDS models to test the capabilities of SIMS. These IDS models were tested by running a rogue process which simulates various attack scenarios. We also show the overhead on the system due to the integration of EMS and access control mechanisms.

4.1 IDS Models

The first IDS model we implemented is for monitoring the events and making a decision based on them. We developed this model by inserting probes in specific parts of the Nachos kernel. These probes generate events which are delivered by the respective sub-system to a central entity (called EMS). The second model utilizes access control mechanisms, which are used for controlling access to the system objects. This IDS model sets the ACLs on particular files in the Nachos file system. When these files are accessed, ACLs on these files are checked to determine whether access is allowed or denied.

4.2 Attack Scenarios

We designed a rogue process that generates a large number of file object accesses. This process is adequate to demonstrate the functionality of both of the above mentioned IDS models. The target files are of different types and the access permissions on these files are defined adequately to test the effectiveness of detection process. The following three types of attacks were simulated for testing SIMS: (1) misuse of system resources, by using the rogue process to generate large number of object accesses, (2) anomalies, by determining statistical deviations using event logs, and (3) sandboxing or access control, by specifying ACLs on files.

4.3 Performance Measurements

We measured the performance of the IDS models using two factors: (1) detection accuracy, and (2) the system overhead due to the integration of the IDS model. The IDS model was able to detect misuse as long as its signature was specified. In other cases, we observed many false negatives. For anomaly detection, the accuracy of detection improved with time. Large periods of data acquisition and processing resulted in more accurate statistical profiles. The access control mechanism provided a way to quickly test the completeness of a security policy.

When a large number of events are generated, the simulation becomes less responsive due to the overheads of processing this audit data. This is generally the experience with the real world IDSs and therefore, audit processing is done offline [2, 27, 28]. The overhead of integrating the EMS and access control mechanisms in Nachos kernel is evaluated by plotting the graphs of time taken by different system calls with and without their integration. Figure 2 shows some of these graphs. These graphs (a, b, c and d) show four different scenarios with a random mix of system calls in each case. A random mix makes sure that the experiment is not tied to a specific user pattern.

We also calculated the overhead caused by individual Nachos system calls on the system. This is shown in Table 2. It should be noted that the overhead due to an individual system call only depends on the object on which it has been called and not on the calling sequence. It thus remains the same for

the IDS model using ACLs for files. It is also interesting to note that the dip in the Figure 2 (b) is due to the lower time units taken by certain system calls during the execution. The average overhead over all system calls was calculated to be approximately 26%. This is the worst-case overhead, when all allowable checks were used to check file access permissions. Further optimizations of the code are possible to reduce the overhead.

Table 2: Average overhead per Nachos system call

Nachos System call	Average overhead (%age)
CreateFile()	29.856
Open()	28.173
Read()	23.788
Write()	27.545
Close()	20.977

The overhead due to the integration of EMS and access control mechanisms in the system is low. Proposed framework can be plugged into an existing operating system, which supports the integration of an event subsystem, without increasing the system load enormously. The log file generated by the EMS contains detailed information about the event, such as the time and accessed object etc. This log information can be used to construct the statistics about the events and decisions can be made based on these statistics.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we described a modeling and simulation platform for intrusion detection systems. After surveying the available implemented IDSs, we were able to extract some essential features to form the core components of SIMS. We have demonstrated the usefulness of SIMS using some attack scenarios and corresponding IDS models. In its current form, the basic foundations for SIMS have been laid and we continue to investigate issues that will eventually make it more complete and appealing.

Simulators almost always abstract out much of the details of the entities of their models in order to make the simulations tractable. However, attackers usually try to exploit vulnerabilities whose presence is due to some flaw in the detailed design of the system. Thus, simulating IDSs, as SIMS proposes to do, is a challenging problem and runs against the grain of most computer simulation strategies.

We would like to develop and test more IDS models for fair evaluation of this tool. Also, due to the current limitations of Nachos to simulate a single user program, SIMS can only model the IDSs running on a single host and a single user. We would like to test our architecture for the multi-host/multi-user environment. The network support in Nachos is weak, so we would like to integrate the network support in Nachos. Such enhancement will enable the modeling of intrusions at the network level too. Our current study also involves the investigation of probability density functions (pdfs) [29], which will accurately reflect the real world system activity and attack scenarios.

References

[1] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in Network Simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[2] P. A. Porras and P. G. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proceedings of 20th NIST-NCSC National Information Systems Security Conference*, pages 353–365, 1997.

[3] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.

[4] X. Chang. Network Simulations with OPNET. In *Proceedings of the 31st conference on Winter simulation*, pages 307–314. ACM Press, 1999.

[5] Information Sciences Institute, USC School of Engineering. The Network Simulator: ns2, http://www.isi.edu/nsnam/ns/. Accessed on Jan 05, 2003.

[6] C. A. Carver, J. R. Surdu, J. M. D. Hill, D. Ragsdale, S. D. Lathrop, and T. Presby. Military Academy Attack/Defense Network. In *Proceedings of the 2002 IEEE Workshop on Information Assurance, United States Military Academy*, June 2002.

[7] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the Practice of Intrusion Detection Technologies. Technical Report CMU/SEI-99-TR-028, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA 15213-3890, January 2000.

[8] J. P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical Report Contract 79F296400, James P. Anderson Co., Box 42, Fort Washington, PA 19034, April 1980.

[9] S. Kumar and E. H. Spafford. A Software Architecture to Support Misuse Intrusion Detection. In *Proceedings of the 18th National Information Security Conference*, pages 194–204, 1995.

[10] S. Kumar. *Classification and Detection of Computer Intrusions*. PhD thesis, Purdue University, 1995.

[11] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A., and D. Zamboni. Analysis of a Denial of Service Attack on TCP. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 208–223. IEEE Computer Society Press, May 1997.

[12] S. E. Smaha. Haystack: An Intrusion Detection System. In *Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL., December 1988.

[13] M.M. Sebring, E. Shellhouse, M.E. Hanna, and R.A. Whitehm'st. Expert Systems in Intrusion Detection: A Case Study. In *Proceedings of the 11th National Computer Security Conference*, pages 74–81, Baltimore, MD., October 1988.

[14] D. Anderson, T. Frivold, A. Tamaru, and A. Valdes. Next-generation Intrusion Detection Expert System (NIDES). Technical Report SRI–CSL–95–07, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, May 1994.

[15] 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*, pages 296–304, May 1990.

[16] D. Denning. An Intrusion Detection Model. *IEEE Transactions of Software Engineering*, 13(2):222–232, February 1987.

[17] H. S. Javitz and A. Valdes. The SRI IDES Statistical Anomaly Detector. In *Proceedings of the IEEE Research in Security and Privacy*, pages 316–376, ,Oakland, CA, MAY 1991.

[18] T. Lunt and R. Jagannathan. A Prototype Real-time Intrusion-Detection Expert System. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 59–66, 1988.

[19] S. N. Chari and P. Cheng. BlueBoX: A Policy-driven, Host-Based Intrusion Detection system. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, San Diego, California, February 2002.

[20] C. Wright, C. Cowan, and J. Morris. Linux Security Modules: General Security Support for the Linux Kernel. In *Proceedings of 11th USENIX Security Symposium*, August 2002.

[21] D. M. Ritchie. The Evolution of the Unix Time-sharing System. In *Lecture Notes in Computer Science No. 79: Language Design and Programming Methodology*. Springer-Verlag, 1980.

[22] W. A. Christopher, S. J. Procter, and T. E. Anderson. The Nachos Instructional Operating System. In *USENIX Winter*, pages 481–488, 1993.

[23] S. Razak, M. Zhou, and S. Lang. Network Intrusion Simulation Using OPNET. In *OPNETWORK2002 conference*, September 2002.

[24] N. C. Rowe and S. Schiavo. An Intelligent Tutor for Intrusion Detection on Computer Systems. In *Computers and Education*, volume 31, pages 395–404, 1998.

[25] S. Schiavo. An Intrusion-Detection Tutoring System Using Means-ends Analysis. Masters thesis, Department of Computer Science, U.S. Naval Postgraduate School, Month 1995.

[26] C. Roberts. Plan-Based Simulation of Malicious Intruders on a Computer System. Masters thesis, Department of Computer Science, U.S. Naval Postgraduate School, March 1995.

[27] P. Spirakis, S. Katsikas, D. Gritzalis, F. Allegre, J. Darzentas, C. Gigante, D. Karagiannis, P. Kess, H. Putkonen, and T. Spyrou. SECURENET: A Network Oriented Intrusion Prevention and Detection Intelligent System. In *Proceedings of the 10th International Conference on Information Security, IFIP SEC94*, The Netherlands, May 1994.

[28] T. Spyrou and J. Darzentas. Intrusion Modeling: Approximating Computer User Intentions for Detection and Predictions of Intrusions. In *Proceedings of Information Systems Security Conference*, pages 319–335, May 1996.

[29] R. A. Maxion. Measuring Intrusion-Detection Systems. In *Proceedings of The First International Workshop on Recent Advances in Intrusion Detection (RAID-98)*, Louvain-la-Neuve, Belgium, 14-16 September 1998.