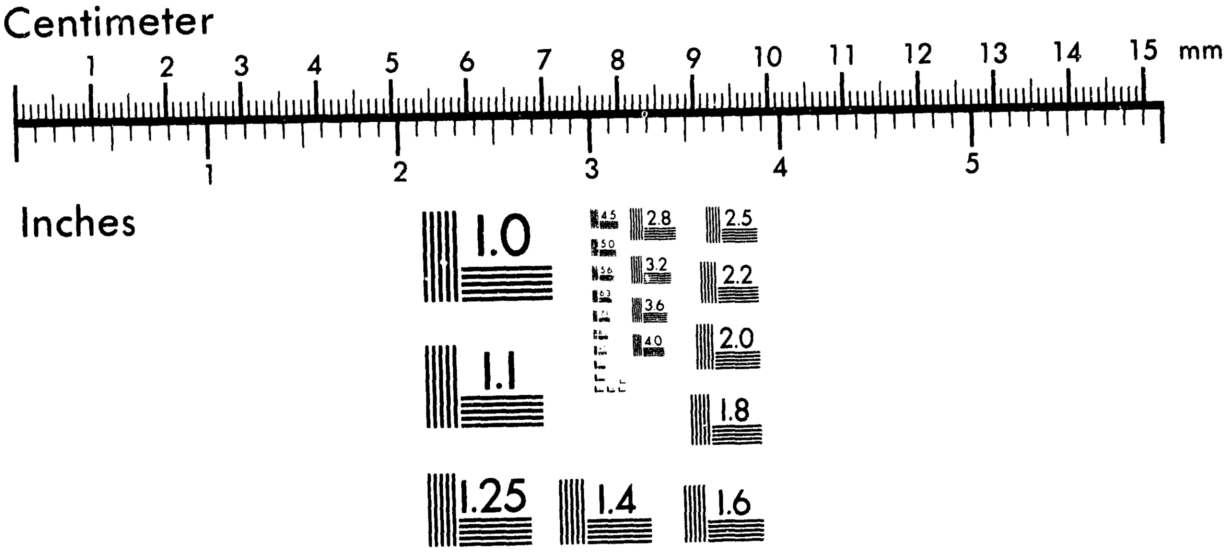
Centimeter
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 mm
1 2 3 4 5
Inches
1.0 1.1 1.25 1.4 1.6 1.8 2.0 2.2 2.5 2.8 3.2 3.6 4.0 4.5 5.0 5.6 6.3

1 of 1

2/

Conf-9305151--9

UCRL-JC-113562
PREPRINT

Using Unix System Auditing for Detecting Network Instrusions

Marvin J. Christensen

This paper was prepared for submittal to the
15th Department of Energy Computer Security Group Training Conference
Albuquerque, New Mexico
May 3-6, 1993

March 1993

Lawrence Livermore National Laboratory

Using Unix System Auditing for Detecting Network Intrusions

Marvin J. Christensen
Computer Incident Advisory Capability (CIAC)
Computer Security Technology Center (CSTC)
Lawrence Livermore National Laboratory[1]
P.O. Box 808 MS. L-303
Livermore, CA. 94550.
(510) 423-5173
mjchristensen@llnl.gov

ABSTRACT

Intrusion Detection Systems (IDSs) are designed to detect actions of individuals who use computer resources without authorization as well as legitimate users who exceed their privileges. Although significant progress is being made in IDS research, at present IDSs are no panacea.

This paper describes a different approach, namely a decision aiding approach to intrusion detection. The introduction of a decision tree represents the logical steps necessary to distinguish and identify different types of attacks. This tool, the Intrusion Decision Aiding Tool (IDAT), utilizes IDS-based attack models and standard Unix audit data. Since attacks have certain characteristics and are based on already developed signature attack models, experienced and knowledgeable Unix system administrators know what to look for in system audit logs to determine if a system has been attacked. Others, however, are usually less able to recognize common signatures of unauthorized access. Users can traverse the tree using available audit data displayed by IDAT and general knowledge they possess to reach a conclusion regarding suspicious activity. IDAT is an easy-to-use window based application that gathers, analyzes, and displays pertinent system data according to Unix attack characteristics. IDAT offers a more practical approach and allows the user to make an informed decision regarding suspicious activity.

[1] This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

1.0 Introduction

Computer system administrators have realized that computer systems are vulnerable to abuse and penetration by both legitimate users who abuse their authority and individuals who are not authorized to use the computers [1]. Intrusion Detection Systems (IDSs) [2,3,4,5,6,7] and many host-based security applications [8,9,10] are designed to detect unauthorized use and enable system administrators to better protect their computer systems.

The ever expanding networking capabilities of computer systems have influenced computer security. Networking has re-directed general security concerns from unauthorized reproduction of files by authorized system users towardconcern with remote intrusion attempts that may result in loss of: 1) secrecy or confidentiality (unauthorized disclosure of information), 2) integrity, (unauthorized alteration), and 3) availability, (as in denial-of-service attacks), [11, 12].

The development of IDSs have been motivated by four main factors: 1) most existing systems are vulnerable to security breaches due to security flaws, 2) replacing systems with more secure systems is expensive due to the dependence on existing systems, 3) developing an absolutely secure system is not possible, and 4) even the most secure systems are vulnerable to insiders [1].

2.0 Current Solutions

To determine if a system has been breached, current host-based IDS technology scans audit data generated by the system looking for suspicious activity. This vast amount of data can be collected at each level of abstraction and analyzed by tools to examine for both suspicious user behavior and unexpected system behavior [1]. IDS research is identifying what level of audit log detail and method of detection will reduce false negatives and false positives. Most IDSs require additional auditing and accounting capabilities to be installed and enabled on each system.

The objectives of many IDSs are to: 1) present to a system security officer (SSO) a brief report on suspicious activities, 2) generate few false alarms (i.e., false positives), and 3) miss few attacks (i.e., false negatives). It may happen that legitimate actions of legitimate users are incorrectly flagged by signature analysis programs. Therefore, additional automated, or human, analysis may be required to decide whether or not an event is suspicious and, if so, whether it is indicative of an intrusion [13].

3.0 Signature Analysis Technique

Attack signature analysis attempts to define attack-type behavior and recognize sequences of events that match (or closely match) predefined signatures. Snapp [13] identifies four components for each event being considered as an attack signature. These components are:

Subject:	owner of the event record
Object:	target of the event record
Action:	that which caused the event record
Context:	conditions preceeding the event record.

All four components of the representation can be used to define the signature. However, the context is the main factor in deciding the severity of the attack.

An attack is not homogeneous over time. The intruders objectives and behavior will change as the attack progresses. Dias [14] identifies three phases of an attack which are:

1)	Preparation:	the intruder gathers information about the target
2)	Execution:	when the attack actually takes place
3)	Aftermath:	the intruder is no longer active, but the system may show signs of the incident.

Aftermath of an incident may be: (1) passive, such as logs in an audit file, (2) static, such as modified file permissions or changed passwords, or (3) active, such as a running process sending information to the intruder.

4.0 Problems with IDSs

Although IDSs have been used for the past few years, none have been formally validated or certified by an authorized agency. Researchers have made claims about each and have reported success, but the inherent nature of IDSs is subject to numerous limitations. There are several caveats, therefore, which should accompany any discussion of IDSs [1]:

1) IDSs cannot prevent intruders from breaching system and network security
2) IDSs cannot replace the SSO
3) IDSs do not make the target system more secure
4) IDSs may not be effective when the target system is untrusted.

Not only is there a large investment in the development of IDSs, but there is an investment in the daily operation. IDSs require computer resources to generate, collect, and analyze audit data. Audit data must be stored for analysis by approved personnel, but must be protected from unauthorized disclosure or release. Personnel must be trained in the operation of IDSs, particularly in the relationship between IDS output and the computer audit data (i.e., how to install and enable auditing capabilities) as well as the legal/social aspects of IDSs. Computer users regard their use of the computer as a personal matter and a system that monitors their activity could be seen as a violation of privacy [1].

Automated routines do not have the intellegince to infer the obvious. Because IDSs are automated programs looking for specific events, knowledgeable intruders can circumvent detection [15]. For example, network monitors are usually searching for string patterns that are suspicious or that have been used in prior attacks. By transmitting compressed data, the network monitor cannot match plain text strings with compressed text.

5.0 Decision Aiding

Although IDSs represent an important development in detecting intrusions, the IDS approach is not the only logical approach. For instance, virtually every operating system (including Unix) has auditing capabilities. Auditing is very useful when there is a suspected breach in security. Since Unix already has auditing capabilities, experienced Unix system administrators can detect suspicious or abnormal activities via audit logs. Unfortunately, Unix auditing data resides in a number of different audit logs, making these data difficult to physically access and conceptually integrate to determine whether or not an attack has occurred. The information available in Unix audit logs is conducive to a new approach to the problem of intrusion detection--decision aiding.

This paper introduces a decision aiding approach embodied in a tool named IDAT (Intrusion Detection Aiding Tool). The decision aiding approach does not fully automate intrusion detection, but is designed to automate certain parts of data collection and aggregation. Determining attack activity by matching attack signatures with audit data may not be possible in all cases because of lack of some audit data. Instead of implementing expert system techniques to resolve the available audit data, system users must use their own discretion.

6.0 Intrusion Decision Aiding Tool (IDAT)

The Intrusion Decision Aiding Tool (IDAT) is a decision aiding tool that can be operated by any Unix user. IDAT gathers, analyzes, and displays the appropriate data which allows the user to detect suspicious behavior. For users not versed in Unix who wish to detect suspicious behavior, IDAT is designed to answer typical questions which an experienced system administrator would ask when trying to determine if a system has been compromised. Along with displaying appropriate data, IDAT includes a decision tree for each type of attack with additional help text that can be displayed on demand. In IDAT, the user must scan the audit logs and use a decision tree to make decisions. The decision aiding approach circumvents many of the limitations of IDSs, yet it can capitalize on the attack models on which some IDSs are based. As with any IDS, IDAT does not stop intruders, but alerts users to their presence.

A requirement of IDAT is that it run on a generic Unix system as delivered from the vendor with its standard auditing capabilites. Current IDSs require modification to the kernel, daemons, or configuration files to allow for additional auditing capabilities that must then be maintained. IDAT uses the default auditing capabilities of SunOS along with standard SunOS commands.

Five attack types are recognized in the current prototype of IDAT [16]. The attacks involve exploitation of five Unix services and utilities:

1)	tftp:	trivial file transfer protocol
2)	doorknob:	attempt to guess username and password
3)	su:	super user
4)	sendmail:	the program that sends and receives e-mail
5)	rsh:	remote shell.

These attacks were selected because of the activity observed on the Internet by members of the Department of Energy's Computer Incident Advisory Capability (CIAC) team. Many system administrators are familiar with all of these attacks, but many are still occasional victims to intruders who launch attacks against them. IDAT was designed specifically to help system administrators and system users detect these types of attacks.

Due to the limited length of this article, only the doorknob attack will be covered (refer to [16] for a full description).

7.0 Doorknob Attack Description

One of the most notorious types of network attacks is the doorknob attack. An attacker, or process, on the remote system attempts to login to the target system by simply guessing a valid user's username and password. The doorknob attack is not limited to intruders. It was used successfully by the internet (RTM) worm [17] and WANK/OILZ worms [18]. For discussion purposes in this paper, the doorknob attack is limited to an intruder using telnet(1C) to attack a system.

Default accounts and active accounts are common ways by which an intruder can gain access to a system. Passwords are used to authenticate users, and they are usually the only authentication process on Unix systems. If someone presents a valid username and password combination, the system assumes that the user is valid and grants the user access. Direct logins to privileged accounts are serious security concerns and are considered part of a doorknob attack.

8.0 Doorknob Attack Characteristics

The key to detecting the doorknob attack is failed login attempts. This approach assumes that the intruder is unlikely to guess the correct username and password combination on the first attempt. Typically, if a valid user fails once, they try again, more carefully pressing the correct keys. Also, if a user has changed a password, the login failure message gives relevant feedback so that the user can try again with the new password. However, repeated failed login attempts are a good indication that a doorknob attack has occurred.

For many Unix systems, only after five login failures does login generate a login failure message. When repeated attempts are necessary, a common technique used to defeat the repeated login failure action is to try three or four times, then break the connection. By breaking the connection or letting the connection time-out (typically 60 seconds), no login failure audit data is generated.

Knowing the username and password combination for each failed login attempt would help in determining if the system was under attack. Given that intruders often guess passwords derived from the username or system name, passwords that differ from each other by only one or a few characters are suspicious. Also, attempts using dictionary names (which are known by intruders to be selected as passwords) are suspicious.

The doorknob attack can be identified by a number of characteristics, including:

- information gathering
- failed login attempts
- prior connections
- username and password combinations
- remote system
- attempted local user account
- time of connection

Figure 1 Characteristics of a doorknob attack

One dimension that is difficult to define is time. Information about the target system can be gathered over a long period of time measured in seconds to years.

9.0 Audit Data Available on Unix[2] Systems

For every telnet(1C) session where the user is prompted for a username and/or password, the telnet daemon telnetd(8C) makes an entry in `/var/log/syslog` via the syslog daemon syslogd(8C). The time of the connection, remote system, and processes ID are listed for each entry. An example of a telnet entry in `/var/log/syslog` is:

```
Jan 16 08:07:46 target in.telnetd[9352]: connect
from  source.llnl.gov (128.115.19.45)
```

Figure 2 Telnet entry in `/var/log/syslog`

The telnet audit entry does not identify the remote user account, local user account, or even if the telnet connection was successful or not. To determine if the telnet connection was successful and which local user account was used, wtmp(5V) file records all logins and logouts. Output of the wtmp(5V) file can be listed with the last(1) command. An example of the wtmp(5V) entry for the previous telnet connection is:

```
christen ttyp1 source.llnl.gov Sat Jan 16 08:08
- 08:17 (00:09)
```

Figure 3 Wtmp(5V) entry via last(1) command

By searching wtmp(5V) for a successful login from the remote system within the permitted login time (e.g., 60 seconds), a correlation between telnet connections and possible local users can be made.

Repeated failed login attempts will result in multiple telnet connections with no successful logins. If the intruder is bold enough to attempt five logins (which generates a repeated failed login message) then searching `/var/adm/messages` for login failures is adequate. The following is an example of repeated login failures by a remote system attempting to login to the target system.

```
Oct 13 17:32:55 target login: REPEATED LOGIN
FAILURES ON ttypa FROM source.llnl.gov, username
```

Figure 4 Repeated login failures

Default system configuration permits network logins to privileged accounts in much the same way someone would log into a normal account. Because network logins into privileged

---

[2] Sun Microsystems SunOS 4.1.1. SunOS are trademarks of Sun Microsystems, Inc.

accounts are privileged processes, there is additional auditing by syslogd(8C). By default, the "ROOT LOGIN" messages are directed to /var/adm/messages. Logins can originate from the local host or from a remote system. The messages for both local and remote logins are:

```
Oct 15 08:23:40 target login: ROOT LOGIN console

Oct 15 08:57:51 target login: ROOT LOGIN ttyp7
FROM source.llnl.gov
```

Figure 5 Login to a privileged account

One problem with the default configuration is that "ROOT LOGIN" audit entries are generated only when successful logins occur to privileged accounts. If a doorknob attack were used, these messages would only record such activity only after the attacker successfully gained access as a privileged user.

Changing the configuration of ttytab(5) to deny direct network logins to privileged accounts is highly recommended. By not allowing network logins to privileged accounts, the user must first login as a normal user and then su(1V) to a privileged account. By editing ttytab(5), terminals can be identified as non-secure, which will deny any network login into a privileged account. Even if the correct password is provided, the connection will be refused just as if they entered an incorrect password and a syslogd(8C) message will be generated. The message is directed to /var/adm/messages. Following are data from two login attempts to privileged accounts where the terminal is configured non-secure. In the first example a correct password has been entered; in the second example an incorrect password was used.

```
Oct 17 09:33:29 target login:ROOT LOGIN REFUSED
ON ttyp6 FROM source.llnl.gov

Oct  1 08:57:51 target login:ROOT LOGIN REFUSED
ON ttyp7 FROM source.llnl.gov
```

Figure 6 Root login to an unsecure terminal

10.0 Detecting Doorknob Attacks with IDAT

A direct approach for detecting doorknob attacks is searching for repeated login failures. These failure messages are located in /var/adm/messages and are generated after five failed login attempts. These entries identify the initiating host and the fifth username tried. This approach is not foolproof since breaking the connection before five failures will not produce a failure message.

The output format of IDAT identifies repeated failed login attempts, telnet attempts where no user has logged in successfully, and refused privileged login attempts. Following is an example of the IDAT format for the doorknob attack:

```
                                      Tue Nov 10 09:04:33 PST 1992

                          Doorknob Attack

Month Date    Time     Activity        Source            Username
----- ---- --------  --------  ------------------   --------
Nov    9  08:12:20  REFUSED   one.remote.system      ?
Nov    9  08:12:10  TELNET    one.remote.system      -no one-
Nov    8  08:12:39  FAILURE   two.remote.system      stan
Nov    8  08:12:09  TELNET    two.remote.system      -no one-
Nov    7  08:12:23  TELNET    three.remote.system    -no one-
-----------------------------------------------------------------
```

Figure 8 IDAT output for doorknob attacks

In this example, IDAT output shows that system "one.remote.system" tried to login to a privileged account and was refused because the terminal was set to un-secure. A user from system "two.remote.system" tried more than five times to enter the correct username and password combination with "stan" being the fifth username tried. Someone from "three.remote.system" established a telnet(1C) connection and then broke or dropped the connection.

11.0 Implementation

Implementation of IDAT is on a stand-alone Sun Microsystems SPARCstation 2 running SunOS 4.1.1 [19], which is connected to the Internet. IDAT assumes vendor default configurations for auditing capabilities. Although the default configuration for the development system within CIAC has been altered to a more secure state, IDAT does not depend on any of these changes.

The first section of code written was the perl [20] scripts used to demonstrate the feasibility of the concepts expressed in this paper. The five perl scripts have been tested on two different systems running SunOS 4.1 and SunOS 4.1.1 which both have perl version 4 (patch 35) installed. The size of the raw (i.e., ASCII) perl scripts range from 80 lines to 350 lines (including comment lines).

The last section of code written was the XView [21] source for the windowing environment. A requirement for IDAT is that it must support OpenLook, since OpenLook is Sun Microsystems user-interface standard. The total number of lines for all six of the XView modules is approximately 1800 lines.

12.0 Conclusion

This paper introduced decision aiding into the field of intrusion detection. Decision aiding builds on the accepted signature attack models and extends it to the understanding of Unix audit data as it relates to known signature attacks.

The visual diagram of the decision tree, combined with the easy-to-use graphical user-interface, provides a simple means of viewing audit data that is otherwise complicated, time consuming, and requires a near-expert level understanding of Unix. IDAT allows any user on the system to quickly view the current status of that system.

Testing and validation of IDAT only requires Unix audit logs and system files in their appropriate format. This also implies that to defeat IDAT, Unix auditing capabilities must be defeated. Although it is possible to defeat Unix auditing (evident by intruders who cover their tracks), IDAT could still aid in the decision that suspicious behavior has occurred, thus accomplishing its original task.

As an aside, when a security tool is developed such as IDAT, one of the many aspects to be considered is the "threat" level it in itself may represent. Threat here is defined as "How useful would it be if the tool fell into the hands of a perpetrator?" Many security tools available today are considered "high threat tools." IDAT, on the other hand, is considered a "low threat tool" because IDAT does not provide intruders with a directly useful tool for attacking systems.

IDAT raises the understanding of computer and network security. IDAT is a tool available today that helps detect intrusions that might have gone undetected and might otherwise continue for days, weeks, or even months.

References

1. United States Air Force, *Computer System Intrusion Detection, E002: Final Technical Report,* TIS report #348., September 11, 1990.

2. Javitz, Harold S., and Alfonso Valdes, "The SRI IDES Statistical Anomaly Detector," *Proc. 1991 Symposium on Research in Security and Privacy,* Oakland, CA, May 1991.

3. Garvey, Thomas G., and Lunt, Teresa, "Model-based Intrusion Detection," *Proc. 14th National Computer Security Conference,* pp. 372-385, Washington, D.C., October 1991.

4. Lunt, Teresa F., "IDES: An Intelligent System for Detecting Intruders," *Proc. of the Symposium: Computer Security, Threat and Countermeasures,* Rome, Italy, November 1990.

5. Snapp, S. R., J. Brentano, G.V. Dias, T.L. Goan, L.T. Heberlein, C. Ho, K.N. Levitt, B. Mukherjee, S.E. Smaha, T. Grance, D.M. Teal, and D. Mansur, "DIDS (Distributed Intrusion Detection System)- Motivation, Architecture, and an Early Prototype," *Proc of the 14th National Computer Security Conference,* pp. 167-176, Washington, D.C., October 1991.

6. Heberlein, Todd L., Gihan V. Dias, Karl N. Levitt, Biswanath Mukherjee, Jeff Wood, and David Wolber, "A Network Security Monitor," *Proc. 1990 Symposium on Research in Security and Privacy,* pp. 296-304, Oakland, CA, May 1990.

7. Lankewicz, Linda, and Mark Benard, *Real-Time Anomaly Detection Using a Nonparametric Pattern Recognition Approach,* Technical Report TUTR 91-106, Tulane University, New Orleans, LA 70118, May 1991.

8. Venema, Wietse, "TCP Wrapper - Network Monitoring, Access Control, and Booby Traps," *Proc. of the Third USENIX Unix Security Symposium,* pp. 85-92, Baltimore, MD, September 1992.

9. Farmer, Daniel, and Eugene H. Spafford, *The COPS Security Checker System,* Technical Report CSD-TR-993, Purdue University, West Lafayette, IN 47907-1398, September 1991.

10. Bartoletti, Tony, *User's Guide for SPI/UNIX version 2.1.,* Department of Energy by Lawrence Livermore National Laboratory. UCRL-MA-103440, Rev. 3., July 1992.

11. McLean, John, "The Specification and Modeling of Computer Security," *Computer,* v23, n1. pp. 9-17, January 1990.

12. Garfinkel, Simson, and Gene Spafford, *Practical UNIX Security.* O'Reilly & Associates, Inc., 1991.

13. Snapp, Steven R., Biswanath Mukherjee, and Karl N. Levitt, "Detecting Intrusions Through Attack Signature Analysis," *Proc. Third Workshop on Computer Security Incident Handling,* Herndon, VA, August 1991.

14. Dias, Gihan V., Karl N. Levitt, and Biswanath Mukherjee, *Modeling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection,* Technical Report CSE-90-41, University of California, Davis., July 1990.

15. Heberlein, Todd L., K. N. Levitt, and B. Mukherjee, "A Method to Detect Intrusive Activity in a Networked Environment," *Proc. 14th National Computer Security Conference,* pp. 362-371, Washington, D.C., October 1991.

16. Christensen, Marvin, "A Unix System Decision Aiding Tool for Detecting Network Intrusions," *Master's Thesis*, University of California, Davis, March 1993.

17. Seeley, Donn, "Password Cracking: A Game of Wits," *Communications of the ACM*, 32(6):700-703, June 1989.

18. Longstaff, Thomas A., and E. Eugene Schultz, "Beyond Preliminary Analysis of the WANK and OILZ Worms: A Case Study of Malicious Code," *Computers and Security* (to appear).

19. Sun Microsystems, *SunOS Reference Manual*, Sun Release 4.1, PN: 800-3827-10, Revision A, March 1990.

20. Wall, Larry, and Randal L. Schwartz, *Programming in perl*, O'Reilly & Associates, Inc., reprint March 1992.

21. Heller, Dan, *XView Programming Manual*, O'Reilly & Associates, Inc., October 1990.

DATE
FILMED
9/20/93

END