Piranha Audit: A Kernel Enhancements And Utilities To Improve Audit/Logging

Vincenzo Cutello, Emilio Mastriani, Francesco Pappalardo
Department of Mathematics and Computer Science
University of Catania, Catania, Italy
{cutello, mastriani, francesco}@dmi.unict.it

Abstract

This paper presents a mechanism to enrich logging as required in TCSEC [1] document to detect and stop possible intrusions based on typical attacks and to protect the sensible audit data from deletion/modification even in root compromise situation.

After installing Piranha Audit, administrators will have a solid infrastructure for improving security and resistance to penetration, with only modest performance penalties.

We present experimental results of the advantages of this solution and the performance impact of the mechanism.

1 Introduction

The main purpose of this work is to present a systematic solutions to the persistent problems of securing and improving the Audit and Logging capabilities.

Moreover, we will present a collection of suites to perform Intruder Detection and a proposal to protect the system against Buffer Overflow Attacks. Covert Storage Channel Analisys is currently under study.

The basic problem is that, in a root compromise case, all audit data can be deleted or altered, trashing the collected informations even if they respect the TCSEC requirements.

An important question is: does anybody have the time to inspect hundreds of lines generated by Audit/Logging system? We provide a collection of utilities that analyze in real time such data and take the least disruptive action to terminate the event that may corrupt the system integrity.

In doing so we will try to meet Dision B, Class 3 TCSEC requirements.

Section 2 decribes the state of the art in Linux about Audit and Logging, the typical attacks against the integrity and security of the system and what are the TCSEC requirements in detail.

Section 3 shows how Piranha Audit helps system administrators to detect what has happened and how Intruder Detection System defends against some more dangerous attacks. Kernel patches applied and a quick description of the suite of user utilities will be also provided.

Section 4 presents performance and penetration testing. Section 5 describes related works. Finally section 6 presents our conclusions.

2 General overview

The standard Linux Kernel meets Division C, Class 2 "partially" in Audit context, since there is no system routine which records events of object introduction or deletion.

Once this problem was solved, to reach Division B, Class 1:

(a) the audit record will have to include, for each event that either introduces an object into a user's address space ot it deletes an object, the name of the object and the object's security level.

(b) Moreover, the system manager would have to be able to selectively audit the actions of any one or more users based on individual identity and/or object security level.

(c) Finally, it must be possible to audit any override of human-readable output markings.

To reach Class 3,

(d) one of the required features is the presence of a mechanism that is able to monitor the occurrence or accumulation of security auditable events that may indicate an imminent violation of security policy. This mechanism will have to be able to immediately notify the security administrator when thresholds are exceeded, and, if the occurrence or accumulation of these security relevant events continues, the system will have to take the least disruptive action to terminate the event.

(e) Moreover, we would need some mechanisms for the identification of events that may be used in the exploitation of the usage of covert storage channels.

In this paper, we will describe an extension of the standard Linux Kernel to reach Division C, Class 2 and that solves problems (a)-(d) as well. Problem (e) currently is solved for a particular case: File Flag Communication[1], but this needs more work.

Now we will describe a list of typical attacks [2].

- A system cracker telnets to the next site on his hit list. "guest – guest", "root – root", and "system – manager" all fail. It does not matter. A lot of sites have easy passwords to crack, based on user name, birth date and so on.

- NFS-Attacks. For instance, running showmount on a target reveals that /export/foo is exported to the world. In this case you can put an .rhosts entry in the remote guest home directory, which will allow you to login to the target machine without having to supply a password!

- Anonymous ftp attacks. Vulnerabilities in ftp are often a matter of incorrect ownership or permissions of key files or directory.

- X windows attacks. If not protected properly (i. e. via xhost or magic cookie mechanisms) window displays can be captured or watched.

- DoS[2] attacks. These type of attacks do not involve a penetration in a system. They slow or block a net service or the entire system.

- Sendmail attacks. Sendmail is a very complex program that has a long history of security problems, i. e. running the "decode" alias is a security risk: it allows potential attackers to overwrite any file that is writable by the owner of that alias, often daemon, but potentially any user.

- "hosts.equiv" attacks. The hosts recorded in this file are trusted: for example if a login request come from a site recorded in hosts.equiv file, there is no need to supply a password. Any form of trust can be spoofed.

- Buffer exploit attacks. If a malicious user finds a buffer overflow in a suid utility, he can gain root privilege.

- Password sniffing. The telnet sessions do not use any form of encryption; so an attacker can sniff the password during a telnet session.

New forms of attacks appear every day. This list can only be a short example.

3 Piranha Audit details

Why would you want to meet the TCSEC requirements? An Audit/Logging file that respects TCSEC layout provides detailed informations as described above. Moreover Piranha Audit protects sensible data against deletion/modification at root level and phisycal disk management (fdisk, format, kernel image replacement, boot the system from floppy). To allow these operations and dumping the Piranha_Audit.log, it's needed Pirahna Manager operator.

He/she is a trusted person that knows the Piranha password that is needed to complete Piranha Audit management sessions.

Only he/she can change the Piranha password. We emphasize that just the root or just Piranha Manager cannot assolve these rules: the execution of any Piranha management session (needs root privileges) requires the Piranha password.

Table 1 shows the files used and kernel protected by Piranha Audit.

This high level of protection has been obtained by applying patches to 2.2.14 Linux Kernel shown in table 2, where PM stands for Piranha Manager and SU for Super User.

[1] With this term we intend a illegal communication from root to user processes based on file presence that indicates, for example, a bit information.

[2] Denial of Service

Table 1: Piranha Audit files.

Files	Description
Piranha_Audit.log	Contains all sensible data from Audit/Logging System
syslog.conf	Configuration file for syslogd daemon
Piranha_FSCF_DB.md5	Collects MD5-fingerprint for critical file system objects
Piranha_SETUID-GID.db	Maps all SETUID-GID root files
Piranha_MD5_Digest_Creator	Utility that uses MD5 algorithm to create digital sign
Piranha_System_Shutdown	Utility to shutdown the machine in critical events
Piranha_Password	Contains the password for Piranha Manager operator

Table 2: Protection modes.

Protected Files	Patched Files	User Level	SU Level	SU+PM Level
Piranha_Audit.log	namei.c, open.c	—	r–	rd-
syslog.conf	namei.c, open.c	—	r–	rw-
Piranha_FSCF_DB.md5	name.c, open.c	—	r–	rw-
Piranha_SETUID-GID.db	namei.c, open.c	—	r–	rw-
Piranha_MD5_Digest_Creator	namei.c, open.c	—	r–	rx-
Piranha_System_Shutdown	namei.c, open.c	—	r–	rx-
Piranha_Password	namei.c, open.c	—	r–	rx-

r=read
d=dumping
x=execute

In “namei.c” and “open.c” we have also introduced a C routine that allows syslogd daemon to open Piranha_Audit.log in append only mode. The TCSEC layout is kept byinserting “printk” calls in “namei.c”, “open.c”, “pipe.c” in correct locations.

The “exec.c” has been patched to detect possible buffer exploit attacks. Suppose that a malicious user has exploited a setuid program. He/she produces “a.out” program that uses this bug to obtain root access. The program does its work and executes a root shell. Piranha Audit detects a particular situation: UID –> 500, GID –> 100, ***EUID –> 0,*** EGID –> 100. There is an anomaly: an inconsistence between UID and EUID; a kernel trap is executed. The user session will be terminated and the account will be locked.

The patched “signal.c” does not allow to kill the Piranha Guardian, detailed below in table 3 with a quick description of Intruder Detection Suite, where IDS stands for Intruder Detection System.

The Simple Watcher utility allows an automatic log analysis detecting patterns that implies an anomaly status.

When it is detected, Simple Watcher sends an Alert Message to Piranha Audit subsystem that takes the least disruptive action to terminate the event.

It is possible to configure rensponses to certain auditable events and to make the PM protection of key files configurable setting the Simple Watcher config file.

4 Performances and penetration testing

Examples 1, 2, 3, 4 and 5 show the behavior of Piranha Audit in some cases of intrusions attempts.

Example 1: Gaining a root shell.

- *Jul 9 10:07:32 SecureHost kernel: Piranha Audit: Warning: the object /bin/su, executed by UID: 500, GID: 100 is set-uid!*
- *Jul 9 10:07:33 SecureHost su[3196]: + tty3 emilio-root*

Example 2: Attempt to remove Audit/Logging archive.

- *Jul 9 10:07:46 SecureHost kernel: Piranha Audit: Object delete command issued from UID 0, GID 0, object name: Piranha_Audit.log.*
- *Jul 9 10:07:46 SecureHost kernel: Piranha Audit: Owner of object is: UID 0, GID 0.*
- *Jul 9 10:07:46 SecureHost kernel: Piranha Audit: Object i-node mode is: 33188.*
- *Jul 9 10:07:46 SecureHost kernel: Piranha Audit: Unauthorized access (delete command) to security event file from UID: 0 GID: 0 detected.*

Example 3: Attempts to delete/link/overwrite Piranha_Audit.log.

- *Jul 9 10:08:59 SecureHost kernel: Piranha Audit: Read access of security event file detected from UID: 0 GID: 0.*
- *Jul 9 11:16:42 SecureHost kernel: Piranha Audit: Hard link not allowed for security event file from UID: 0 GID: 0.*
- *Jul 9 11:17:04 SecureHost kernel: Piranha Audit: Unauthorized or incorrect use of security event file detected from UID: 0 GID 0.*
- *Jul 9 11:18:14 SecureHost kernel: Piranha Audit: Unauthorized or incorrect use of security event file detected from UID: 0 GID: 0.*
- *Jul 9 11:18:36 SecureHost kernel: Piranha Audit: Unauthorized or incorrect use of security event file detected from UID: 0 GID 0.*
- *Jul 9 11:18:55 SecureHost kernel: Piranha Audit: Attempt to create confusion with special object (mknod system call) and protected Piranha Audit files detected from UID 0, GID 0.*

Example 4: Fdisk attempt.

- *Jul 9 11:23:45 SecureHost kernel: Piranha Audit: (ALERT LEVEL 3) Attempt of disk management without correct security procedure detected from UID: 0 GID: 0.*

Example 5: Satisfying TCSEC layout.

- *Jul 12 15:41:30 SecureHost kernel: Piranha Audit: Object introduction detected from UID 500, GID 100, object name is: trial.*
- *Jul 12 15:41:30 SecureHost kernel: Piranha Audit: Owner of object is: UID 500, GID 100.*

Table 3: IDS utilities.

Utility	Quick description
Piranha_Account_Locker	Locks an account after compromised events
Piranha_Intruder_Killer	Terminates work session of a buffer exploit compromised user
Piranha_MD5_Digest_Creator	Creates md5 finger-print
Piranha_PWD_Creator	Sets the Piranha Manager Password
Piranha_SETUID-GID_Checker	Controls every 60 minutes the root SETUID-GID map
Piranha_SETUID-GID_Init	Initializes root SETUID-GID database file
Simple Watcher [9]	Instructs Piranha about Alert Level reactions
Piranha_System_Shutdown	Halts the machine in critical situation
Piranha_Dumper	Allow under root+PM privileges file system management
Piranha_FSC	Protects critical files against modification/trojan horse attacks
Piranha_FSC_Init	Initializes the database with MD5 signs of critical files
Piranha_Guardian	Controls that all IDS works correctly. It cannot be killed
Piranha_Init	Script that coordinates the execution of IDS
Piranha_Overflow_Checker	Checks for dimension overflow of Piranha_Audit.log
Piranha_PG_PID_Search	Searches for suitable PID for Piranha_Guardian
Piranha_PID-UID_Finder	Gets from PID its owner (UID)

- *Jul 12 15:41:35 SecureHost kernel: Piranha Audit: Object delete command issued from UID 500, GID 100, object name: trial.*
- *Jul 12 15:41:35 SecureHost kernel: Piranha Audit: Owner of object is: UID 500, GID 100.*
- *Jul 12 15:41:35 SecureHost kernel: Piranha Audit: Object i-node mode is: 33188.*
- *Jul 12 15:42:42 SecureHost kernel: Piranha Audit: Special object introduction (mknod system call) detected from UID 500, GID 100, object name is: pipe.*
- *Jul 12 15:42:42 SecureHost kernel: Piranha Audit: Device number of object is: 0.*
- *Jul 12 15:42:42 SecureHost kernel: Piranha Audit: Object i-node mode is: 33261.*

Below we show the Piranha Audit System behavior to underline the performances under different conditions.

5 Related works

Anderson [3] first proposed using audit trails to monitor system activity. The use of existing audit records suggested the development of simple tools to check for unauthorized access to systems and files.

Bonyun [4] argued that a single, well-unified logging process was an essential component of computer security mechanisms.

Picciotto [5] presents a sophisticated audit capability for a Compartmented Mode Workstation.

Intrusion detection systems that focus on anomalous behavior have also driven research in auditing and logging. Axent Technologies [7] has presented IDS in Unix and NT platforms, but nothing for Linux.

Tripwire facility from the COAST [8] project at Purdue University can take care of the file system, but it can only report problems: it does not take any action to terminate the dangerous event.

6 Conclusions

We have presented Piranha_Audit, a systematic solution to the persistent problems of securing and improving the Audit and Logging capabilities, that prevents a broad class of buffer overflow security attacks from succeeding.

Its most important futures are that it denies the deletion/modification of protected files even in a root compromised situation; with TCSEC layout, the system administrator has a powerful method to investigate; intrusion detection is critical in today's complex enterprises. Attempting to manually review audit trails is hopelessly

Table 4: Main memory details (in bytes).

Total	Used	Free	Shared	Buffers	Cached
64716800	63213568	1503232	24977408	1347584	41750528

Table 5: CPU Info.

processor	0
vendor_id	GenuineIntel
cpu family	586
model	2
model name	Pentium MMX
stepping	3
cpu MHz	200.457340
fdiv_bug	no
hlt_bug	no
sep_bug	no
f00f_bug	yes
coma_bug	no
fpu	yes
fpu_exception	yes
cpuid level	1
wp	yes
flags	fpu vme de pse tsc msr mce cx8
bogomips	80.08

Table 6: Stress Testing.

CPU Stress Test	PASSED
Disk Stress Test	PASSED
CPU+Disk Stress Test	PASSED

CPU STATES: 62 processes: 48 sleeping, 14 running, 0 zombie, 0 stopped 98.2% user, 1.7% system, 0.0% nice, 0.0% idle
DISK USE: dd if=/dev/zero of=trial count=400000
PASSED means that Piranha Audit System behavior is correct how in no stress situation.

time-consuming and a losing battle given the number of systems and different types of audit trails. Today we need automated intrusion detection tools. Digitals finger print have produced with MD5 [6] algorithm, one of the best in its area.

All this with little performances degradation how is showed in the following figure.

7 Acknowledgements

We would like to thank the anonymous referees for their valuable comments.

Figure 1: Performances.

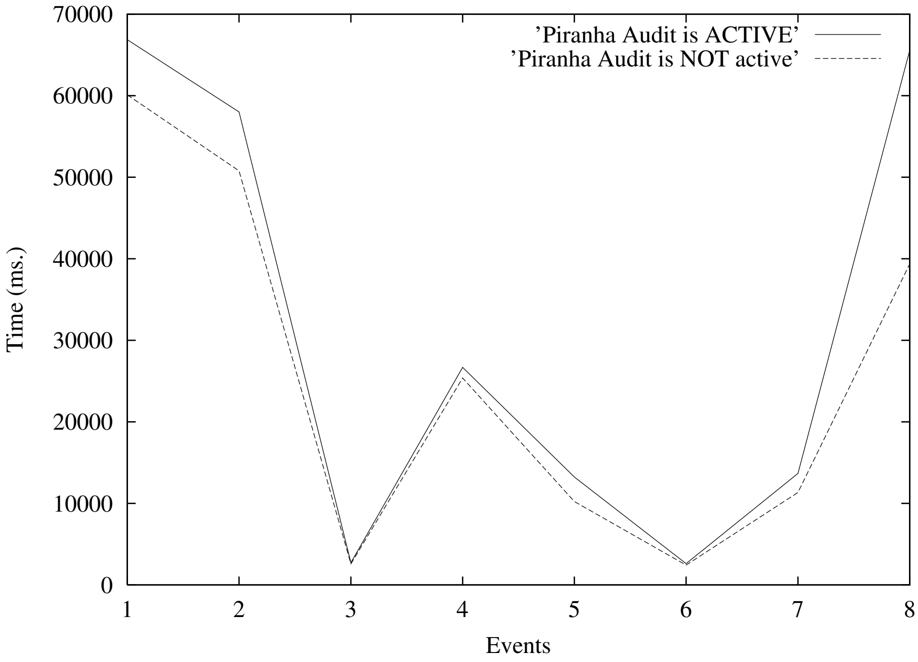

Table 7: Performance keywords

Event	**Keywords**
find | grep lyx	1
Pirannha Audit compile process	2
latex work.tex	3
Starting an X session	4
netscape	5
lyx	6
gimp	7
Linux Boot	8

References

[1] *Department of Defense Trusted Computer Systems Evalutation Criteria*. DoD 5200.28-STD, December 1985.

[2] D. Farmer and W. Venema, *Improving the Security of Your Site by Breaking Into It*, USENET posting, December 1993.

[3] James P. Anderson, *Computer Security Threat Monitoring and Surveillance,* James P. Anderson Co. , Fort Washington, PA (1980).

[4] David Bonyun, *The Role of a Well-Defined Auditing Process in the Enforcement of Privacy and Data Security,* Proceedings of the 1989 IEEE Symposium on Security and Privacy, pp. 206-214 (1989).

[5] J. Picciotto, *The design of an Effective Auditing Subsystem,* Proceedigns of 1987 IEEE Symposium on Security and Privacy, pp. 2-10 (1982.

[6] R. L. Rivest, *The MD5 Message Digest Algorithm*, MIT Laboratory for Computer Science and RSA Data Security, Inc., Request for Comments nr. 1321, April 1992.

[7] Axent Technologies, *Guide to Intrusion Detection*, 1995.

[8] COAST, Computer Operations, Audit and Security Technology, *www.cs.purdue.edu/ar97/research/coast.html*.

[9] Todd Atkins, *Simple Watcher*, Todd.Atkins@CAST.Stanford.edu, Stanford University.