An attack detection system for secure computer systems - Outline of the solution

Ioanna Kantzavelou[1] *and Sokratis K. Katsikas*[2]

[1] *Department of Informatics,*
Technological Educational Institution (T.E.I) of Athens,
Ag. Spiridonos St., Aegaleo, Athens 12210, Greece
tel.: +30-1-6437918, fax: +30-1-6437918
e-mail: ioanna@hol.gr

[2]*Department of Mathematics,*
University of the Aegean,
Karlovassi 83200, Samos, Greece
tel.: +30-273-35482, fax: +30-273-35483
e-mail: ska@aegean.gr

Abstract

The spread of distributed information technology has increased the number of opportunities for crime and fraud in computer systems. Despite the fact that computer systems are typically protected by a number of security mechanisms, attacks continue to occur. In addition, it seems infeasible to close all the known security loopholes of today's systems. No combination of technologies can prevent legitimate users from abusing their authority in a system. Thus, new lines of defence are required to ensure safe operation of computer systems as well as data protection. Attack Detection Systems are an approach to enhancing the security of a computer system. The Attack Detection System (ADS) which is the subject of this paper, is a real-time attack detection system which allocates points to users who are attempting to attack the target system, detects attacks by examining the number of points each user has been given, and takes countermeasures according to this number of points. The outline of the solution that implements the ADS is described in detail in this paper.

Keywords

Attack Detection, Event Analysis, Security Relevant Events, Security Relevant Errors, Risk Levels.

1 INTRODUCTION

In the last few years, many organisations have adopted the use of *auditing systems*. Auditing systems capture all events that occur on a computer system, and keep logs of the audit data in special files for security analysis. In the beginning, the analysis of log files was carried out by the security officer of the system, who had to search all the printed audit data to detect security violations. The large volume of data made this difficult. The need for tools for automated security analysis of audit data became evident. Such a system is called an *attack* (or *intrusion) detection system* and must have the following goals: to provide a trail of computer system events; to determine how the system was breached; to determine who was responsible for a breach; and to take action to prevent further breaches.

In conclusion, there is a need for an attack detection system that can provide protection to a computer system by detecting security violations in real-time. Therefore, the problem to be solved was defined as stated in the next paragraph.

1.1 Problem Definition

The overall goal of the work carried out was to provide a real-time attack detection system which will detect attacks on a computer system and will instruct the computer system to take action to prevent further security violations.

The problem to be solved was the design and implementation of a real-time attack detection system for secure computer systems which could: monitor all events that occur on a computer system, log the events, analyse each event in order to determine whether it is of potential relevance from a security point of view, store the security relevant events separately, examine security relevant events against rules stored in a rule base, decide (in real-time) if an attack is taking place, send a signal to inform the security officer of a system when an attack occurs, and finally take action to prevent further attacks.

These requirements define the problem that was solved by the implementation of the Attack Detection System. The next paragraph presents the essential results of this implementation.

1.2 Results

An attack detection system for secure computer systems, called the *Attack Detection System* (ADS), has been implemented. This system is a real-time rule-based system which provides an audit trail for all computer system events, detects attacks by analysing audit data, and takes measures to prevent additional attacks when an attack occurs.

This attack detection system uses a novel method for detecting attacks, the *point allocation method* (Kantzavelou 1996). According to this method, the ADS allocates points to users who are attempting to attack a computer system. Based on these points, the ADS takes countermeasures to protect the computer system.

Furthermore, the Attack Detection System is modifiable. This allows the administrator of the attack detection system to improve its effectiveness. The concept of the Attack Detection System is described in the next paragraph.

1.3 The Concept of the Attack Detection System (ADS)

The Attack Detection System (ADS) which is the subject of this paper aims at providing enhanced security in a computer system called the *target system*. The ADS carries out the main functions described below in order to fulfil its goal. Figure 1 depicts these functions and the inter-function communication within the ADS (Kantzavelou 1994). The ADS modules are discussed extensively in (Kantzavelou 1996).

Event Collection

The Attack Detection System monitors all target system activities called *events*, and logs these events in a data base called *Event Data Base* (EDB). Furthermore, it examines each event in order to filter the events which are of potential relevance from a security point of view.

Attack Detection

Analysis of the audit records and detection of attacks in real-time. The ADS applies a rule-based technique to detect attacks, which implies the use of a rule base called *Rule Base* (RB). When the ADS detects that a user is acting suspiciously, it counteracts by automatically deciding upon an action and instructing the target system to take this action.

Attack Detection System Access

The ADS informs the Security Officer (SO) of the target system about attacks detected and suspicious users. It also gives to the SO a picture of all events that have occurred on the target system.

Rule Base Access

This function allows the administrator of the ADS to modify the Rule Base in order to adjust the ADS to the target system.

Event Data Base Maintenance

The ADS provides this special function to maintain the Event Data Base (EDB) which is the collection of the audit data files. In particular, the purpose of this function is to retrieve and store records in the EDB.

Rule Base Maintenance

The ADS provides also a function to maintain the Rule Base (RB) which consists of rules. In particular, this function retrieves, stores, updates, and deletes records from the RB.

2 OUTLINE OF THE SOLUTION

The Attack Detection System is a rule-based system (Kantzavelou 1994). In particular, a rule base has been defined to characterise the state of audit data which constitutes an attack. The method of the examination of audit data is an important part of the design of this system. This section is divided into two parts to describe two methods of examination of audit data: the *examination of commands* and the *examination of service points*. It also gives reasons why the second method has been chosen as the most appropriate for the design and implementation of the Attack Detection System.

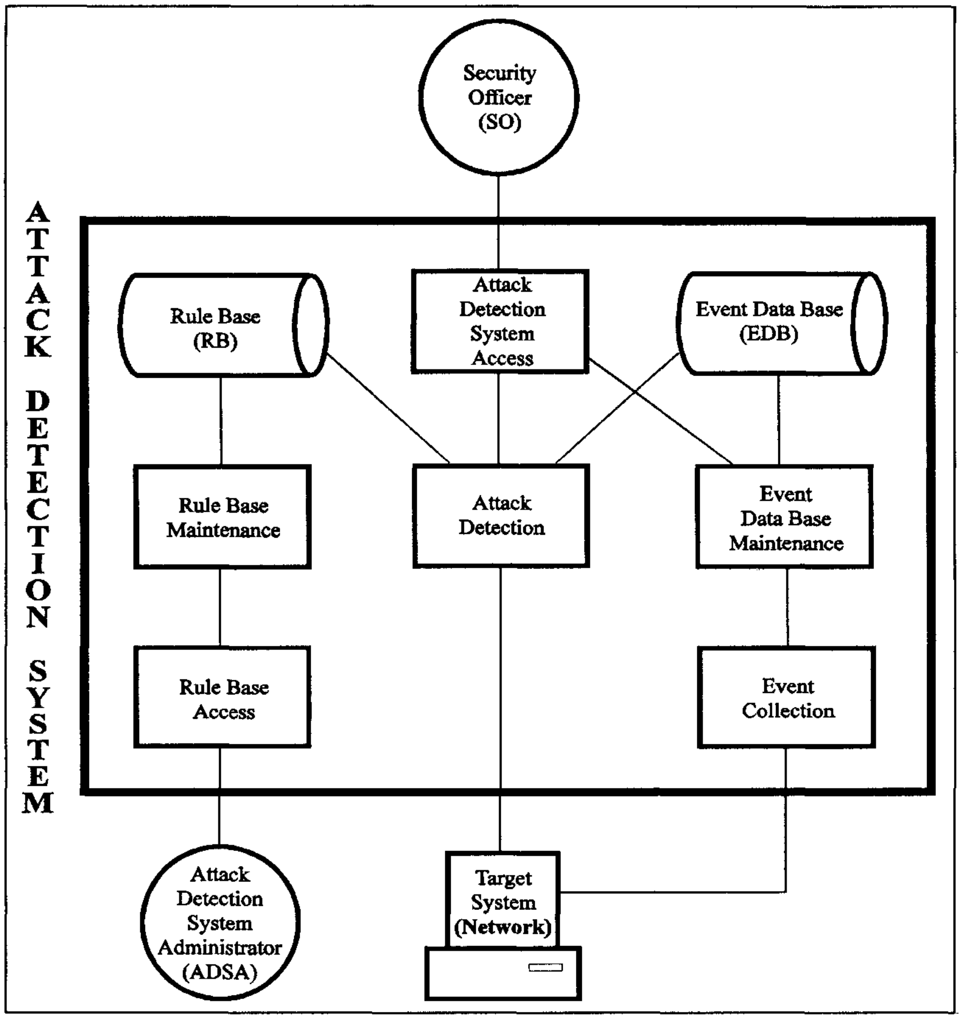

Figure 1: The Main Functions of the Attack Detection System

2.1 Examination of Commands

When a user requests the execution of a command (or a program) from the Target System (e.g. a UNIX[1] -based system), the Event Reception Module collects data about this activity and stores it in the Event Data Base. The key information of each collected activity is the command line which shows the actual event that took place on the Target System and its results. Therefore, a possible method for the examination of audit data would be analysis of the syntax of each command a user types. An approach to implement this method might be the following:

[1] UNIX is a trademark of AT&T Bell Laboratories

- The Event Data Base will include the command line and the status of its execution, as well as information about the user such as his login name, his host machine name, the date and the time of his activity, etc. An example of an Event Data Base record could be the following:

date	time	user	tty	host	command	status
27/4/93	18:26:35	ioanna	tty0	hobbes	cat > /myfile	Permission denied

- According to the above example, user *ioanna* attempted the creation of a file which failed because the user had no write permission for the accessed directory. The reason of the command failure is indicated by the *status* field.
- The Rule Base will consist of sets of rules. Each set of rules will refer to one command type and will include as many rules as there are possible syntaxes and status of this command. An example of the Rule Base construction could be the following:

set	syntax	status	volume	preventing action
cat	cat > filename	Permission denied	3	logout
cat	cat > filename	File exists	4	lock screen
cat	cat filename	Unreadable	4	logout

- According to the above example, the first rule refers to the creation of a file using the *cat* command. It defines that if a user attempts three times to create a file without write permission on the accessed directory, then the ADS will take the action *logout* to prevent the user from further attacks. The volume specified in a rule indicates the expected number of attempts of the associated command that will trigger the rule. This number is the command risk level and derives from risk analysis of the target system commands.
- When the Attack Detection Module (ADM) - which is responsible for analysing and examining the audit data (Kantzavelou 1996) - retrieves records of activities from the Event Data Base which belong to the same user and refer to one syntax of a command, then it will get the rule which matches the syntax of the typed command and the command status. In the above example, the ADM will examine the activity against the first rule, and will take the preventing action that has been defined, if the user has failed three times to execute this command.

Unfortunately, this method has a number of disadvantages and weaknesses:

- The implementation of the rule base requires the analysis of all possible syntaxes and reasons of failure of each command. Considering that the UNIX system for instance supports more than 400 user commands, such an implementation demands too much effort and the ADS effectiveness might prove the lack of rule base completeness.
- A user may rename a command. In this case no rule can match such a command, so that an attack would pass through the Attack Detection Module undetected.
- The rule base cannot include rules for individual user programs. This means that an attacker who uses his own program to damage a system will evade detection.

- The use of an editor would only show that a user called the editor to edit a file. Information about the file status (the file was changed or not) would be available only if the Attack Detection System performs additional examination of the file characteristics (date and time last saved).
- Detection of attacks will be in non-real time because the Attack Detection Module will expect a number of commands (volume) in order to characterise an activity as an attack.

The mentioned disadvantages and weaknesses of this method show that this method is incapable to fulfil some of the primary requirements stated in the problem definition paragraph. Therefore, another approach is required for the examination of audit data. The chosen method for the design and implementation of the Attack Detection System which is analysed in the next paragraph provides the required alternative solution.

2.2 Examination of Service Points

The alternative solution for the examination of audit data, is based on the design of operating systems. All operating systems provide *service points* (Peterson 1985) through which commands and programs request services from the kernel for their execution. These service points are elementary functions which are traditionally defined in the assembler language of the machine in older operating systems, whereas, recent operating systems define them in C language. Thus, in most operating systems, for each service point there is a C function which names the service point.

The UNIX system uses the term *system call* for a service point (Sun Microsystems 1990), the DOS system names it *software interrupt* (Keller 1988), etc. Although the names differ, the basic philosophy is held in common. The term *system call* will be used throughout the rest of this paper, because the implementation platform was UNIX-based. This term is also equivalent to the term *event* in the context of the ADS. In addition, the term *activity* is used here to describe the attempted execution of a command or program.

2.2.1 Service Points Under UNIX

When a command (or program) is requested by a user, then a number of system calls are requested by the command from the kernel. Each of these system calls is responsible for performing an elementary operation required for the execution of the command, and might be called more than once. Given that a system call is actually a function, a value that indicates the exit status of this function is returned when a system call is requested. This value might be:

- *'-1'* if an error occurred and the system call failed.
- *'0'* or greater than zero if the system call succeeded. The number in this case is associated with the requested system call, e.g. a successful *read()* system call will return the number of bytes read.
- *'?'* if the system call never returns a value

When an error occurs, the execution of the relevant command stops, and the associated system call fails returning an error code. This code indicates the reasons of the system call failure. There are a total of 128 system calls and 90 system call errors(Kantzavelou 1994) currently supported by the Sun operating system Release 4.1.3. An example of the list of system calls that are required for an attempt to view the contents of a file without read permission using the 'cat *filename*' command is presented in Figure 2.

System Call	Return Value	Error Code
open()	3	---------
read()	32	---------
mmap()	0	---------
mmap()	0	---------
open()	4	---------
getrlimit()	0	---------
System Call	**Return Value**	**Error Code**
mmap()	0	---------
close()	0	---------
getuid()	82	---------
getgid()	10	---------
open()	3	---------
fstat()	0	---------
mmap()	0	---------
close()	0	---------
open()	3	---------
read()	32	---------
mmap()	0	---------
mmap()	0	---------
close()	0	---------
open()	3	---------
read()	32	---------
mmap()	0	---------
mmap()	0	---------
close()	0	---------
close()	0	---------
fstat()	0	---------
open()	-1	EACCES
write()	7	---------
writev()	25	---------
close()	0	---------
close()	0	---------
close()	0	---------
exit()	?	---------

Figure 2: List of System Calls Required for a 'cat *filename*' Command

Among the system calls requested for the execution of this command, only the *open()* system call failed once returning the *EACCES* error code to indicate that the user had no read permission for the accessed file.

The method chosen for the design and implementation of the Attack Detection System examines system calls instead of commands for the following reasons:

i) The list of system calls in all operating systems is limited. This fact allows the implementation of a complete rule base which assures the effectiveness of the ADS.
ii) System call names cannot be changed by a user, because system calls belong to the operating system.
iii) Individual user programs rely on system calls for their execution. Therefore, auditing of the system calls of a program will permit detection of an attacker who attempts to damage the target system using his own program.
iv) Auditing of editing a file will make available information of the file status.
v) A system call might be requested more than once for the execution of a command. Thus, a number of system call request records may exist after the execution of a command. This fact may allow the characterisation of a single command (which may constitute an attack) thus facilitating real-time detection of attacks.

2.3 Filtering of Security Relevant Events

The Event Reception Module (ERM) is responsible for collecting all target system events (Kantzavelou 1996). Due to the fact that the volume of collected audit data is large, the examination of all events by the Attack Detection Module (ADM) becomes difficult. Filtering of audit data aims to reduce the large volume of audit data collected by the ERM. Thus, security relevant information is retained, while the bulk of innocuous event data is ignored by the ADM.

The principle of filtering security relevant events described below was based on the following criteria:

- A successful security relevant system call is a security relevant event
- An unsuccessful system call that has returned a security relevant error is a security relevant event.

Thus, the filtering of events requires the filtering of system calls and errors. The basic steps that were followed to determine which system call errors and system calls are of potential relevance from a security point of view, are described below (Pfleeger 1989):

1. Filtering of system call errors

As described in the previous section, a system call might fail if an error occurs. In this case, the error indicates the reasons for the system call failure. In particular, a system call error might indicate one or more threats that could endanger the target system. Therefore, filtering of the system call errors was based on the following criteria:

- what is the impact of the action indicated by a system call error on the target system
- what is the frequency of a system call error, i.e. how many system calls might have a specific error.

Among the list of these errors, only ten errors related to security were found. These security relevant errors are described below (Sun Microsystems 1990):

EACCES
An attempt was made to access a file in a way forbidden by the protection system.

EADDRNOTAVAIL
An attempt to create a socket with an address not on this machine was made.

EBADF
Either a file descriptor does not refer to an open file, or a read (or write) request is made to a file that is open only for writing (or reading).

EDQUOT
A *write()* system call to an ordinary file, the creation of a directory or symbolic link, or the creation of a directory entry failed because the user's quota of disk blocks was exhausted, or the allocation of an inode for a newly created file failed because the user's quota of inodes was exhausted.

EEXIST
An existing file was mentioned in an inappropriate context, for example, **link**.

EFAULT
The system encountered a hardware fault in attempting to access the arguments of a system call.

ENOMEM
During an *execve()*, *sbrk()*, or *brk()* system call, a program asks for more address space or swap space than the system is able to supply, or a process size limit would be exceeded.

ENOTEMPTY
An attempt was made to remove a directory with entries other than '.' and '..' by performing a **rmdir()** system call or a **rename()** system call with that directory specified as the target directory.

EPERM
Typically this error indicates an attempt to modify a file in some way forbidden except to its owner or super-user. It is also returned for attempts by ordinary users to do things allowed only to the super-user.

EROFS
An attempt to modify a file or directory was made on a file system which was mounted read-only.

Figure 3 shows the type of threat(s)(Kantzavelou 1995) that each of the security relevant system call errors indicates.

Error Nam Error Name	Disclosure of Information	Corruption of Information	Unauthorised Use of Resources	Misuse of Resources	Unauthorised Information Flow	Denial of Service
EADDRNOTAVAIL				✱	✱	
EFAULT						✱
ENOMEM				✱		
EDQUOT				✱		
EEXIST		✱		✱		
EBADF	✱	✱	✱			
ENOTEMPTY		✱				
EACCES	✱	✱	✱		✱	
EROFS		✱				
EPERM		✱	✱			

Figure 3: Security Relevant System Call Errors - Types of Threats Map

Due to the fact that not all the system call errors are equally serious, a risk level was assigned to each system call error to rate them. The scale used for this rating was 1 to 10, as it is presented in Figure 4. Risk level '1' refers to the lowest risk whereas risk level '10' refers to the greatest risk.

Error Name	Risk Level
EADDRNOTAVAIL	1
EFAULT	2
ENOMEM	3
EDQUOT	4
EEXIST	5
EBADF	6
ENOTEMPTY	7
EACCES	8
EROFS	9
EPERM	10

Figure 4: Risk Levels of System Call Errors

2. *Filtering of system calls*

The filtering of the system calls was based on the following criteria:

- what is the impact of a successful system call on the target system
- what is the frequency of a system call, i.e. how many commands might request a specific system call for their execution
- what is the *total risk level* of a system call. The total risk level was calculated by adding up the individual risk levels of the security relevant errors that a system call might return. Due to the fact that not all the system calls are equally serious, the total risk level was used to rate them.

	successful system call	**non successful** system call	**security relevant** system call	**security irrelevant** system call	**security relevant** error	**security irrelevant** error
successful system call	▨	▨	Y	N	▨	▨
non successful system call	▨	N	N	N	Y	N
security relevant system call	Y	N	N	▨	Y	N
security irrelevant system call	N	N	▨	N	▨	N
security relevant error	▨	Y	Y	▨	Y	▨
security irrelevant error	▨	N	N	N	▨	N

Figure 5: Decision Matrix for Filtering of Security Relevant Events

Among the list of system calls, there were 79 found related to security. The security relevant system calls and errors that they might return are listed in (Kantzavelou 1994). Furthermore, the assigned total risk levels of the above system calls are also presented in (Kantzavelou 1994).

Finally, the rationale of the decision as to whether an event is security relevant or not was defined. Figure 5 depicts a decision matrix which presents this rationale. The shadowed boxes represent impossible cases; the symbol *'Y'* represents cases of security relevant events; and the symbol *'N'* represents cases of security irrelevant events.

Upon the determination of the security relevant events that are to be examined by the Attack Detection Module, a list of monitoring actions was specified to represent these security relevant events. This list is provided in (Kantzavelou 1994).

3 CONCLUSION

This paper has addressed the problem definition of the design and implementation of an attack detection system for secure computer systems, called the Attack Detection System (ADS). In comparison with other attack detection systems, the ADS described herein is a real-time system which provides flexibility in order to be more effective in detecting attacks.

More specifically, this paper defined the problem which was solved, presented the most significant results of this work and the overall concept of the ADS. The outline of the solution was extensively discussed by example and other solutions presented in comparison to the chosen one, to show the reasons that the method chosen for the design and implementation of the Attack Detection System was the most appropriate.

4 REFERENCES

Kantzavelou, I *An Attack Detection System for Secure Computer Systems*, M.Sc. Thesis, 1994.

Kantzavelou I, Patel A *'Issues of Attack in Distributed Systems - A Generic Attack Model', Proc. of the Joint Working Conference IFIP TC-6 TC-11 and Austrian Computer Society*, September 20-21, 1995, Graz, Austria, pp. 1-16.

Kantzavelou I, Patel A *'An Attack Detection System for secure computer systems - Design of the ADS', Proc. of the 12th International Information Security Conference (IFIP SEC '96)*, May 21-24, 1996, Samos, Greece, pp. 337-347.

Keller, L *Operating Systems: Communicating with and Controlling the Computer*, Prentice Hall (1988).

Peterson, J and Silbverschatz, A *Operating System Concepts*, Addison-Wesley Publishing Company, Second Edition (1985).

Pfleeger, C *Security in Computing*, Prentice-Hall International Editions (1989).

Sun Microsystems, Inc. *'System Calls' SunOS Reference Manual*, Vol II, Printed in USA, Revision A (1990).

5 BIOGRAPHIES

Ioanna Kantzavelou holds a B.Sc. in Informatics from the Technological Educational Institute of Athens, Greece, and a M.Sc. in Computer Science (by research, on security in computer networks) from the University College Dublin, Ireland. She has been involved with many European R&D projects in the area of security, in particularly medical information systems security. She currently is Visiting Assistant Professor with the Department of Informatics of the Technological Educational Institute of Athens, Greece.

Sokratis K. Katsikas (Dip. Eng., M.S., Ph.D) was born in Athens, Greece, in 1960. He received the Diploma in Electrical Engineering degree from the University of Patras, Greece, in 1982, the M.S. in Electrical & Computer Engineering from the University of Massachusetts at Amherst, USA in 1984, and the Ph.D. in Computer Engineering from the University of Patras, Greece, in 1987.
He has held teaching and research positions with the University of Massachusetts at Amherst, the University of Patras, the Computer Technology Institute, Patras, Greece, the University of La Verne Athens Campus, the Office of Naval Research, Hellenic Navy and the Technological Education Institute of Athens. In 1990 he joined the Department of Mathematics of the University of the Aegean, Greece, where he now is Associate Professor of Informatics and Associate Head of the Department.

He has been involved with many CEC funded R&D projects in the areas of computer security, robotics, and artificial intelligence. He has authored or coauthored more than 65 technical papers and conference presentations in his areas of research interest, which include computer security, estimation theory, adaptive control, and artificial intelligence. He has served as chairman or member of program and organizing committees of many international conferences and is a reviewer for several technical journals.

He is a member of IEEE, of the ACM, of the Greek Computer Society (Vice-President), of the Technical Chamber of Greece and of the Hellenic Association of Electrical and Mechanical Engineers. He is also a member of the New York Academy of Sciences. He is listed in "Who's Who in the World" and in "Who's Who in Science and Engineering".

He is the National Representative of Greece in IFIP General Assembly, a member of the CEPIS Network on Legal & Security Issues, a member of CEN TC251 (Medical Informatics) - WG6 (Security, Privacy, Safety), a member of IMIA (International Medical Informatics Association) - WG4 (Security, Privacy), the chairman of IFIP TC11 (Information Systems Security) - WG 4 (Network Security), the chairman of the Greek Computer Society Special Interest Group on IT Security and the secretary of the Greek Computer Society Special Interest Group on Software Reliability, Quality and Safety.