Recovery Based Architecture To Protect Hids Log Files Using Time Stamps

Surinder S. Khurana
Punjab Engg. College, Chandigarh, India
surindersingh.cs07@pec.edu.in

Divya Bansal, Prof. Sanjeev Sofat
Punjab Engg. College, Chandigarh, India

***Abstract* – After the great revolution in the field of Information Technology, many applications made necessity to run computer systems (either servers or client machines) all the time. Along with improvements and new inventions in technology, the threat of attacks through computer networks becomes a large issue. Host Based Intrusion Detection is a part of security system that protects hosts from various kinds of attacks. It also provides a great degree of visibility (of system activities). It is quite widest that HIDS are vulnerable to attacks. An adversary, if successfully enters in a system can disable HIDS or modify HIDS rules to hide its existence. One can easily evade HIDS. In [7] we propose a new architecture that protects HIDS from such attacks. In this paper, we have proposed a new mechanism to check integrity of log files. We have discussed its affects on performance of system.**

I. INTRODUCTION

Intrusion Detection System [5] is an imperative ingredient in network computer security which plays a vital role in detecting the intrusive activities before they occur. Intrusion Detection is usually done through scanning network traffic and/or hosts data and activities. These intrusive activities can be defined as - activities performed by some adversary for gaining unlawful benefits. The adverse affects of such intrusion activities are in terms of loss of confidentiality, integrity and availability of resources or services.

IDSs can be classified under various categories. Figure-1 illustrates the various classifications of Intrusion Detection Systems.

Response Based	• Active IDS • Passive IDS
Domain Of Detection Based	• HIDS • NIDS
Underlying Detection Technique Based	• Anomaly based IDS • Signature based IDS

Figure-1: IDS Classification

An IDS can be active or passive. Passive IDS detect the attacks and logs information or raise alarms. Active IDS takes action in response to an already detected attack. Active IDSs are also known as Intrusion Prevention System.

Section 2 discusses Detection and Recovery based Architecture to protect HIDS. Section 3 describes time stamping based protocol to check integrity of log files. In section 4 and 5, we discuss implementation details. Affects of our architecture on system performance has described in section 6. In section 7, we discuss some related works. We present directions for future work in 6 and our conclusion in section 8.

II. OVERVIEW OF DETECTION AND RECOVERY BASED ARCHITECTURE

In [1], architecture has proposed to detect attacks on HIDS and recover HIDS to its previous healthy state. The architecture protects HIDS from two type of attacks: first is that it does not allow adversary to kill the HIDS process. And the other is it does not allow unauthorized modification of rule or signature database. The basic idea behind the architecture is to allow the adversary to perform attack on HIDS and then recover the HIDS to its previous healthy state. A new process called MonitorIDS has been introduced in the proposed architecture. MonitorIDS is a lightweight system process which detects the attacks that affects HIDS and takes required actions to recover HIDS from affects of that attack. MonitorIDS takes care of both HIDS process and integrity of HIDS related information. However, this architecture does not consider underlying technique used by HIDS to detect intrusions. It can be used with either signature based or anomaly based IDS.

Figure 2 depicts working of proposed architecture. As shown in figure 2, a backup of HIDS related files has been created immediately after the installation. MonitorIDS process is embedded with HIDS. This process regularly monitors the HIDS process and files after a small fraction of time gap. If MonitorIDS found HIDS process dead (detect unauthorized kill of HIDS) it immediately restarts the HIDS. It also monitors integrity of files related to HIDS. These files may include rule or signature database. If it found any unauthorized modification of HIDS files it replace the modified files

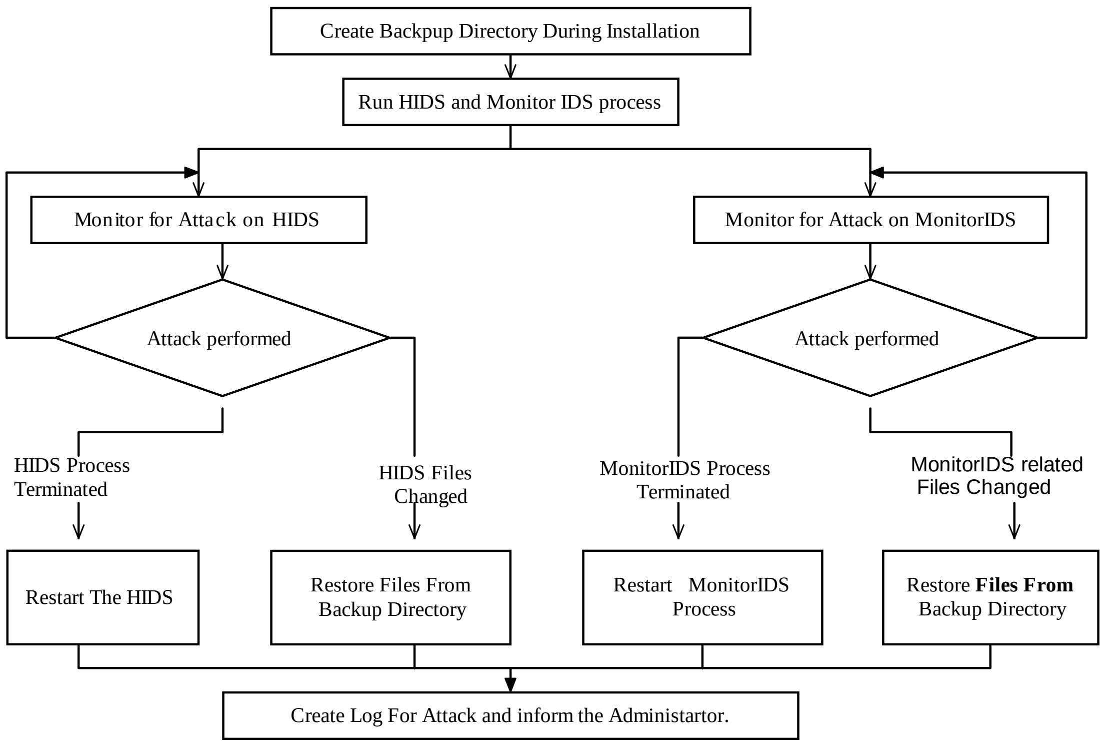
Create Backpup Directory During Installation
Run HIDS and Monitor IDS process
Monitor for Attack on HIDS
Monitor for Attack on MonitorIDS
Attack performed
Attack performed
HIDS Process Terminated
HIDS Files Changed
MonitorIDS Process Terminated
MonitorIDS related Files Changed
Restart The HIDS
Restore Files From Backup Directory
Restart MonitorIDS Process
Restore Files From Backup Directory
Create Log For Attack and inform the Administartor.

Figure 2 : Block Diagram of proposed architecture

with files from backup directory. It also logs the attack information and informs the administrator.

The architecture also takes care of backup directory and MonitorIDS process related information, so that these cannot be modified by some adversary. One important point to note about MonitorIDS is that it is a System process. To run this process as system process an entry labeled with respawn has made in /etc/inittab file. That cannot be stopped. If some adversary makes attempt to kill this process, operating system kernel automatically restarts it. MonitorIDS process also takes care of /etc/inittab file so that entry related to it cannot be removed.

III. TIME STAMPING BASED MECHANISM TO CHECK INTEGRITY OF LOG FILES

The main goal is to detect and recover unauthorized modification of log files by any process that is not related to HIDS. Such detection is made based on the last modification time (a file attribute, changed when the file was modified) of the log file.

As shown in Figure-3, when HIDS is running under this mechanism it should follow the below given sequence of steps to write an entry in a file.

1. Write log entry in log file and in backup copy of log file.
2. Encrypt Last Modification Time of log file and its backup copy.
3. Write encrypted version of last Modification time of log file and its backup copy into a file.

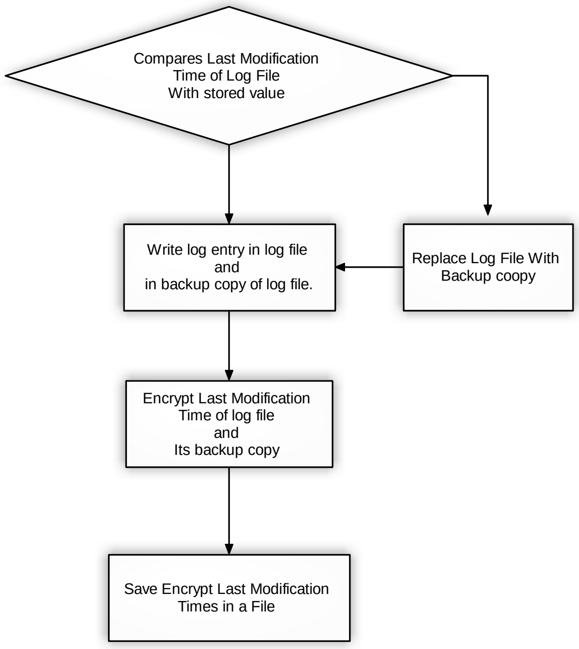
Compares Last Modification Time of Log File With stored value
Write log entry in log file and in backup copy of log file.
Replace Log File With Backup coopy
Encrypt Last Modification Time of log file and Its backup copy
Save Encrypt Last Modification Times in a File

Figure - 3: Sequence of Steps to Write Entry in Log File

Before writing any entry IDS process compares the current last modification time of log file and last modification time that was stored in a file as specified in step 3 given above. A mismatch between these two values represents that any other process changed the log file means unauthorized modification of log file. In such a case log file is replaced with its backup copy or if unauthorized access to the backup file is detected, it will be replaced with log file.

IV. IMPLEMENTATION

The proposed architecture has been implemented in Red Hat Linux environment. Our architecture does not emphasis on underlying technology used by host based intrusion detection system to detect intrusion. Rather than construct a HIDS from scratch, we decided to leverage an HIDS within current implementation of our architecture.

The two hypotheses that underlie this dissertation are practical in nature. First, they intend to show that it is feasible to protect HIDS using proposed architecture. Second, proposed architecture does not affect system performance adversely. Therefore, an implementation was a center point for the development of this dissertation and was used both for practical verification of the intended features of the architecture and for aiding in reasoning about and experimenting with its characteristics.

V. NEW PROCESSES PROPOSED IN ARCHITECTURE

Practically MonitorIDS process (introduced in our proposed architecture) has implemented with two sub processes :

1. MonitorProcesses
2. MonitorFiles

To ensure that MoitorIDS processes live forever we run these processes as system processes. When the adversary tries to kill this process kernel automatically restarts it. A respawn labeled entry related to these processes has been written in system file ‘\etc\inittab’ to run these processes as system processes.

MonitorProcesses process ensures that all processes related to OSSEC are always running. If it found any process dead it restarts the process corresponding HIDS process. MonitorProcesses also prevents modification related to these entries in ‘\etc\inittab’ file. In case of deletion or modification of these entries from ‘\etc\inittab’ file, this process writes new entries. The purpose of MonitorFiles sub-process is to protect HIDS files from unauthorized modification. If it detects any modification or deletion, it restores the victim file with genuine file from backup directory.

VI. AFFECTS ON PERFORMANCE OF THE SYSTEM

To evaluate the affects on the performance on system following steps are carried out :

1. To evaluate the affects on execution time of basic Linux commands, some time measurements were carried out.
2. System Monitoring Utility was used to check the changes in utilization of processor due to processes related to our proposed architecture.

A. Affects on execution of Linux commands

For this purpose, most of the time measurements were carried out regarding basic Linux commands (ps, find, who). Each Command was executed 100 times and all corresponding 100 execution times were recorded. The average of these timings has been considered as the Average Execution Time(ATE). Average Execution Time (AET) was calculated twice. In first run, we calculate execution time (ATE1) without running processes (such as MonitorIDS process) related to our architecture. And in second run we calculate execution time (ATE2) while processes related to our architecture were running.

Table-1 Average Execution Time in milliseconds

Command	ps –ef	find / >/dev/null
Number of system calls	536	10055
(a) Average Execution Time (AET 1) (time required to execute on machine while not running processes related to proposed architecture)	25.9	63.1
(b) Average Execution Time (AET 2) (time required to execute on machine while processes related to proposed architecture are running)	30.1	65.5
Overhead relative to (a)	0.16%	0.03%

Table 1 represents average execution time (in milliseconds) and corresponding overhead. Very small fraction of time was observed as overhead due to the MonitorIDS process.

B. Changes in processor utilization due MonitorIDS Process

MonitorIDS process is a very lightweight process that works in iterative manner. In each iteration checks the state (process terminated or running) of HIDS processes and integrity of HIDS related files. One iteration requires approximately .001 Second time for execution.

Figure 3 represents the CPU utilization graph when MonitorIDS process was not running. Because the processor in use is Dual Core two lines represents utilizations of processor. As shown in the graph CPU utilization is between 0% to 20%. Most of the time the CPU utilization is below 10%.

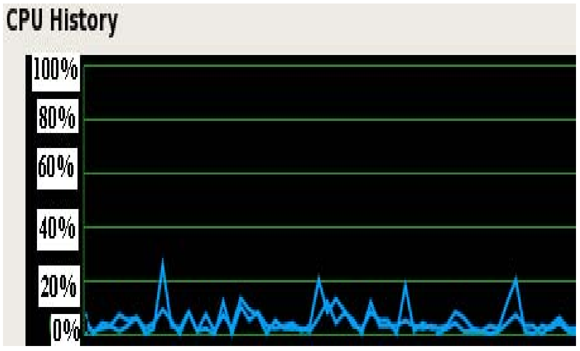

Figure 3: Graph showing CPU utilization while MonitorIDS not running

But as shown in graph given in figure 4, while MonitorIDS process has been running the CPU use is increased to some extent. Most of the time one of the CPU utilization lies between 15 to 20 % and other CPU utilization lies between 5 to 15%.

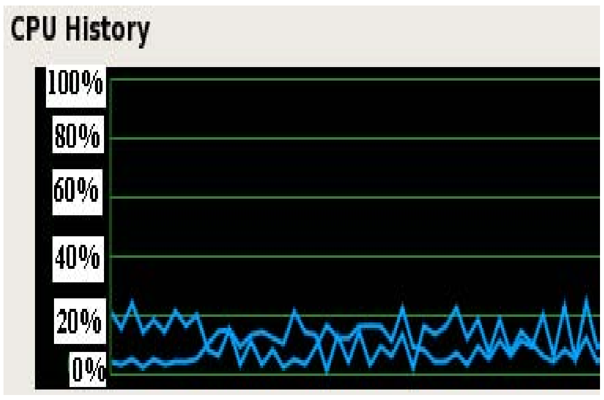

Figure 4 Graph showing CPU utilization while MonitorIDS running
.

As in comparison to graph in figure3 the CPU use is increased about approximately 8 to 10% of total CPU utilization while MonitorIDS process has been running.

VII. RELATED WORK

To protect the HIDS many researchers have given the proposals. In [2], Laureano and Jamhour define the use of virtual machine to protect the HIDS. Virtual machine is used because of its inherent properties like separation of execution space. Other benefits [3] of virtual machine, that are useful in system security are isolation (a process running outside the VM cannot be accessed internal process of VM), inspection (VM state can be accessed by VM Monitor), and interposition (any operation issued by VM can be modified by VM Monitor). The idea behind the architecture is to encapsulate the system activities (both user and guest activities) in virtual machine and place the Intrusion Detection System outside the scope of virtual machine. Now any process has not been able to access the IDS. IDS monitors the activities performed in the virtual machine and identify the intrusive activities. Their approach use type II virtual machine. The architecture protects the IDS from attacks by placing it out from the scope of other processes. However, this approach has one basic problem regarding with theperformance issue. The overhead due to virtual machine degrades the system performance to very worse status. As results given by them if we compare the execution timings of basic routines under an actual machine and virtual environment, there is a large variation. The time taken by routines under virtual machine is much more than time taken on an actual machine.

Table 2 represents a subset of results given by Laureano and Jamhour in [2]. The virtual machine overhead is so high that each routine requires double or even more time for execution as in comparison to time it requires to execute on a normal machine.

Table-2 Virtual Machine Overhead on execution time

Command	ps –ef	find / >/dev/null
Number of system calls	536	10055
(a) Host Time (time required to execute On real machine)	25	125
(b) Guest Time (time required to execute On real machine)	68	484
Overhead relative to (a)	172%	287%

The architecture also affects the network performance. The performance of applications such as FTP and HTML etc. is also turned down.

The work in [4] represents the use of virtual machine type-I. The basic idea is same as to run the HIDS in execution space that cannot be accessible by other processes. However, virtual machine overheads also affect this architecture.

The use of virtual machines for the security of systems has defined by G. Dunlap & et. al.[6]. The proposal defines an intermediate layer between the monitor and the host system, called Revirt. This layer captures the data sent through the syslog process (the standard UNIX logging daemon) of the virtual machine and sends it to the host system for storing and later analysis. However, if the virtual system is compromised, the guest syslog process can be terminated and/or the log messages can be manipulated. by the intruder, and consequently they are no longer reliable.

VIII. FUTURE WORK

Still there are many issues to be addressed about how the proposed architecture can be best implemented and used. MonitorIDS process checks HIDS continuously in iterative way. To check HIDS in an iteration MonitorIDS process requires .001 seconds.

On an average, the adversary has .0005 (.0001/2) seconds to disable our architecture and HIDS by executing following steps :

1. Terminate the MonitorIDS process
2. Remove the entry related to MonitorIDS process from /etc/inittab file to stop the Operating system to restart MonitorIDS process.
3. Terminate HIDS processes or change files related to HIDS.

A DFA (Deterministic Finite Automata) shown in Figure 4, represent such sequences of steps. State q1 is the initial healthy state and q4 is the final state. After reaching at q4 state attacker can tamper the HIDS. To avoid this case, there should be some mechanism that prevents transitions of system states from q1 to q4. As discussed above in average case transition from q1 to q4 is only possible if made in .0005 seconds. Such mechanism can be based on detecting system call sequence required for this transition. If any such sequence is detected it should delay (approximately .0005 execution of these system calls so that before the system state will change from q2 or q3 to state q4, transition from q1 or q2 to state q1(initial healthy state) took place. This mechanism can be implemented by changing operating system kernel.

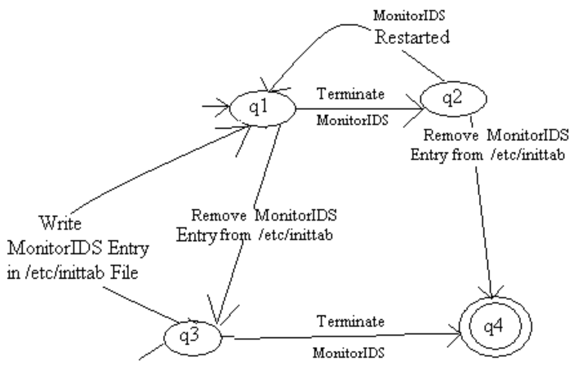
MonitorIDS Restarted
q1
Terminate MonitorIDS
q2
Remove MonitorIDS Entry from /etc/inittab
Write MonitorIDS Entry in /etc/inittab File
Remove MonitorIDS Entry from /etc/inittab
q3
Terminate MonitorIDS
q4

Figure-4: DFA representing sequence of steps to disable our architecture.

Issue of authorization and integrity of genuine updates of HIDS information is another important issue to be addressed. In this approach, we did not consider any technique to authorize of update. Because IDS's rules/signature database requires frequent updates, inclusion of a strong authorization mechanism would be required.

IX. CONCLUSION

In this paper, we propose a new time stamping based approach to check integrity of log files of HIDS. This approach can be combined with Detection and Auto Recovery based architecture to Protect Host Based IDS. As discussed in section 4, unlike other approaches our approach does not use virtual machine and hence does not affect system performance adversely.

Our mechanism ensures that HIDS process always live (cannot be killed by adversary) and the information related to HIDS can never be updated by any adversary. This architecture also ensures the integrity of frequently updated log files of HIDS.

REFERENCES

[1] A. Abraham, C. Grosan and C.M. Vide. Evolutionary Design of Intrusion Detection Programs. International Journal of Network Security, Vol. 4, No. 3, 2007.

[2] M Laureano, C Maziero, E Jamhour, Protecting host-based intrusion detectors through virtual machines- Computer Networks- Elsevier, 2007.

[3] P. Chen, B. Noble, When Virtual Is Better Than Real, Workshop on Hot Topics in Operating Systems, 2001.

[4] T. Garfinkel, M. Rosenblum, A virtual machine introspection based architecture for intrusion detection, ISOC Network and Distributed System Security Symposium (2003).

[5] S. Axelsson. Research in intrusion detection systems: A survey. Technical report, Chalmers University of Technology, 1999.

[6] G. Dunlap, S. King, S. Cinar, M. Basrai, P. Chen, ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay, USENIX Symposium on Operating Systems Design and Implementation, 2002.

[7] Surinder Singh khurana, Ms. Divya Bansal, Prof. Sanjeev Sofat "Detection and Auto Recovery Approach to Protect Host Based IDS" 2009 IEEE International Advance Computing Conference (IACC 2009)