Real Time Malware Defense System

Dilip Pandit, Dineshkumar Kongonda, Kabita Ghosh, Ravikumar Wagh, Tushar B Kute

Information Technology
Sandip Institute of Technology and Research Centre, Nashik

panditdilipr@gmail.com
dineshkongonda60@gmail.com
kabita18ghosh@gmail.com
wagh1.ravi@gmail.com
tbkute@gmail.com

***Abstract*: - In this paper, we present about the "Real Time Malware Defense System based on Linux Operating System". The basic idea of this system is that it will classify a process as a malicious or benign. This system uses the information in the kernel structures of a process to do the classification of processes and to do run-time analysis of the behaviour of an executing program. And if the malicious process get detected it will kill that process, thus securing the user's information to get harm. The purpose of this system is that, to prevent the unauthorized programs to run and avoid being attacked maliciously. As we know that while working on our workstation, we face many problems and among them malware is one of the most familiar problem. It doesn't matter whether you are working on Windows or on other's Operating System like UNIX, Linux or MAC. Mostly we face this kind of problem with Windows OS, but same problem we are facing in Linux Operating System also but in different ways. Linux has its own advantages like it avoids Windows malwares up to 80% to 90%. But in our cyber world there are mind hunters, who find their own way to crack your system. In case of Linux like Operating System, if you know the file system very well you can do anything. In Linux everything works by processes, that's why many times we face the problems related to malware and we cannot identify them. To solve this problem the "Real-Time Malware Defense System" will be helpful.**

Keywords
Malware, Kernel, System calls, intruder, code

I. INTRODUCTION

With the rapid development of information technology, the computer has been more and more important in our lives. Especially our working, learning and entertainments, and we can say each and every work has been dependent on computers because it has bought a lot of convenience and flexibility while it also makes us to face with lot of security risks. Computer malware, viruses and Trojans become the most important factor of the hazards of computer security. The Windows Operating System is frequently been attacked by the viruses and makes destruction to the system as it is not so robust. But as we know that Linux is a very robust and more secure Operating system but still there are security risks. The Linux Operating System does not support .exe file so threat of viruses are not found but malwares are present in Linux. Since 1991 Linux have 1000s of distributions and each one is designed in different ways for commercial and non-commercial as well as personal usage but for all them the Kernel of Operating System is common for all. Linux can be categories into four users 1.Root user 2.Admin User 3.Limited User 4.Guest User. Only Root and Admin have all authorities to do anything in Operating System but still Admin users have some limitation than Root User. Root user can do anything in Operating System. Because of this kind of privilege authentication in Linux is much more secure than other Operating Systems, but as we know that in computer system there is no complete security and these are few reasons why Linux distributions also have malware activities. Malware is very simple word to say but it is somewhat very difficult concept to understand. Including malware there are many other things on which we have to focus, like virus, worm, Trojan etc. People always get confused and they cannot differentiate among them. They consider malware, virus, worm and Trojan in same group or they think they are same but things are different here. Malware can't be considering in same group because malware covers virus, worm and Trojan etc. For more clarity, consider a big umbrella as malware and virus, worm and Trojan comes under that umbrella. So we can say that malware can cover all that malicious processes. Malware, short for malicious software, is software used or programmed by attackers to disrupt computer operations, gathers sensitive information, or gains access to private computer systems. It can appear in the form of code, scripts, active content, and other software. Malware is a general term used to refer to a variety of forms of hostile or intrusive software. In law, malware is sometimes known as a computer contaminant. Malware is different from defective software, which is legitimate software but contains harmful bugs that were not corrected before release. However, some malware is disguised as genuine software, and may come from an official company website in the form of a useful or attractive program which has harmful malware embedded in it along with additional tracking software that gathers marketing statistics. Software such as anti-virus, anti-malware, and firewalls are relied upon by the users at home, small and large organizations around the globe to safeguard against malware attacks which helps in identifying and preventing the further

spread of malware in the network. Today, malware is used primarily to steal sensitive information of personal, financial or business importance by black hat hackers with harmful intentions. Malware is sometimes used broadly against government or corporate websites to gather guarded information, or to disrupt their operation in general. However, malware is often used against individuals to gain personal information such as social security numbers, bank or credit card numbers and so on. Left unguarded, personal and network computers can be at considerable risk against these threats. Computer malware is becoming an increasingly significant threat to the computer systems and networks world-wide. In recent years, security experts have observed a massive increase in the number and sophistication of new malware. According to a recent threat report by Symantec, in 2008 alone, 5491 new software vulnerabilities were reported, 1.6 million new malware signatures were created, 245 million new attacks were reported, and the financial losses caused by malware soared to more than 1 trillion dollars. It is, however, interesting to note that though 50% of new malware are simply repacked versions of known malware. So computer security has become an urgent problem to be solved.

In Linux, process is an active body of the computer system. Any safety issue is eventually caused by a process of the computer system. So malwares can attack these processes and harm the computer system. Usually these destructive behaviour always implement sabotage by deriving the new illegal process so as to achieve the purpose of destroying the Computer system.

II. RELATED WORK

By using run time analysis of the behaviour of processes we are able to classify malware and benign process. In past by analysis information of system call, the malware processes are detected, this is known as system call based run time malware detection. Commonly, the sequence of system calls is used to detect malwares. Malware commonly attacks on vulnerabilities of process. Vulnerabilities means loopholes or we can call it as mistakes in software. There are different types of techniques which are used to detect the malware based on system calls:

1. **Signature based analysis:** This is commonly used technique by antivirus software's to detect malwares. In this technique the system maintains database to detect worms (malwares). This database includes data set of threats (virus, worms, malware etc). This system is continuously running as daemon process, whenever system identifies the same pattern in database it will inform or activate an alarm. The system match patterns by using data mining algorithms. Data mining technique for malware detection usually starts with first step of generating feature set. This technique fails to identify, when new malware that has not been yet registered. As soon as new malware comes in a network or host it is must, to update the database of system. Database contains signature defined as API sequence of calls. In Microsoft Windows Operating System the Windows API calls are traced to identify malwares, because Windows API reflects the functional behaviour of programs. In Windows based application system, the application software interacts with KERNEL32.DLL, NTDLL.DLL, and USER32. DLL. this is known as Windows API interface. [3]

2. **Signature free analysis:** In previous malware detection systems, we obtain information from API interfaces or system call for both Linux and Windows. But we only get result processed by functions. That mean we could not get required information to classify malware and benign process. Also this process may take longer time to detect malwares. It is difficult to quickly decide the process as malware even malware is running. The older methods continuously perform memory access based on some criteria. The signature free system is used to detect polymorphic malware and unknown malware based on Windows API calls sequences called as Intelligent Malware Detection System (IMDS). In 2005, researchers (Kolter and Maloof, 2006) described new technique based on machine learning. There is no need to store signature of malwares in database, because in this technique we identify malwares by getting information from kernels data structures such as process descriptor.[3]

The Linux real-time monitoring real-time system consists of four modules as:

(a) **Process White List:** The process in white list indicates that processes in it are safe processes and processes in blacklist are not to be creditable. [2]

(b) **Process Logs:** Process logs contains list of usernames which are allowed to run process. [2]

(c) **Process White List Detection System:** The process white list detection module performs detection of processes, matching of processes in white list, process start/ban. [2]

(d) **Process Dialog Box:** The process dialog box controls human machine interface in Linux application layer. The task (process) structure (descriptor) maintained in the kernel of an operating system contains record of every action and resource usage of a process. Task register contains lots of fields near about 118. By keeping watch periodically on 11 out of those fields in task structure we can get sufficient information about processes to classify process as malware or benign. Some of the fields from these are CPU usage, memory usage, Users, process-id, parent process-id, Process group id, session id. Generation of signatures for the kernels data structure and to build process using their fields in task structure is proposed. [2]

3. The In-Execution Malware Detection using Task Structure of Linux Process architecture is classified into four modules listed and explained below:

(a) **Feature Logger:** Feature logger periodically gets image of task structure of processes. They dump 118 fields of task structure. From this dump, they extract information which is required to classify processes. [2]

(b) **Feature Analyzer and Pre-processor:** In order to differentiate malware and benign process, the parameters collected in above phase are pre-processed in two phases. In first phase, the parameters which are flags, constants, static identifiers, empty or zero valued fields are eliminated. In the second phase time-series analysis of the short-listed parameters is performed. This system collects data from both malware and benign process at variable time. After that, system discards those parameters which are not required to classify. That means it takes into account only those parameters which are essential to classify malware and benign process. [2]

(c) **Classification:** Classification system works in two phases: Training and Testing. In training phase, system uses classification algorithm like Naive Bayes, Bayes Net, J48 and JRip. This system uses two or more classifier algorithm to increase accuracy. Finally, the testing is performed after every millisecond on the basis of extracted values of short-listed task structure parameters. [2]

III. PROPOSED WORK

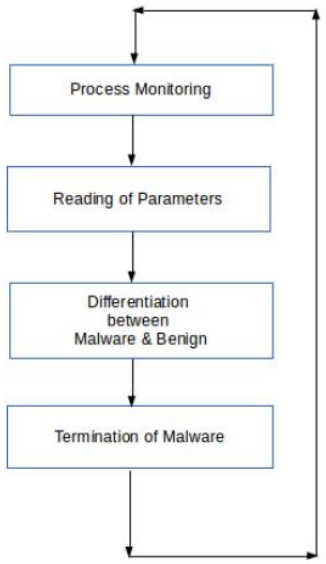

Figure1. Architecture of RTMDS

The above architecture is the overview of the Real Time Malware Defense System. In the figure, the first block shows the Process Monitoring which describes the real-time monitoring of the process which tries to execute on your system. The second block shows the Reading of Parameters which describes the fetching of the information about the process from kernel's task structure. The third block describes about classifying of processes whether the process is a malicious process or benign process based on the parameters fetched. After classification of the processes, if the process is detected as malicious process then that process is kill or we can say that the process is terminated.

In this paper, run-time analysis of model behaviour of an executing process is done. The information in the kernel structures of a process can be used to discriminate between a malicious and benign process. The task structure maintained in kernel of an operating system contains records of every action and resource usage of a process; therefore, the actions and resource usage patterns of a malicious are expected to be different from that of a benign process. As the process descriptor contains the data that describes the executing programs such as open _les, the process address space, pending signals, the process state etc, we can use the following parameters for discrimination between malicious and benign process.

1. The number of pages in executable memory mappings (task st→mem manager →exec vm).
2. The number of page tables of a process (task st→Mem_manager→nr ptes).

Now we can get the process information using the Structure which defines the task structure of a process.

```
typedef struct
{
    Char Name [16]; //Process name
    _u32 PID; //Process ID
    _u32 PPID; //Parent process ID
    _u32 UID; //Process user ID
    _u32 GID; 15//Process group ID
    int static prio; //Process static priority
    Char CmdLine [256]; //Command name of running
    Process
} process info;
```
[7]

In the simplest case strace runs the specified command until it exits. It intercepts and records the system calls which are called by a process and the signals which are received by a process. The ptrace () system call provides a means by which one process (the "tracer") may observe and control the execution of another process (the "tracee"), and examine and change the tracee's memory and registers. It is primarily used to implement breakpoint debugging and system call tracing. A tracee first needs to be attached to the tracer. Attachment and subsequent commands are per thread: in a multithreaded process, every thread can be individually attached to a (potentially different) tracer, or left not attached and thus not debugged. Therefore, "tracee" always means "(one) thread", never "a (possibly multithreaded) process". Ptrace commands are always sent to a specific tracee using a call of the form ptrace (PTRACE foo, PID).

The various information or parameters about the processes are stored in the /proc directory. So we can fetch those information from /proc directory. Like, we can get memory details of the total processes by executing the command **cat /proc/meminfo**. The following figure shows the entire memory details.

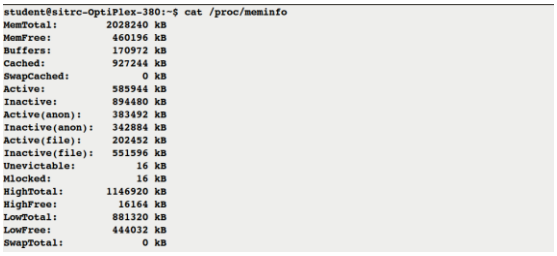

Figure 2. /proc/meminfo

There are many other information that we can fetch, like in the following figure which shows all the directories present in the /proc directory. It can be seen by using the command **ls /proc**.

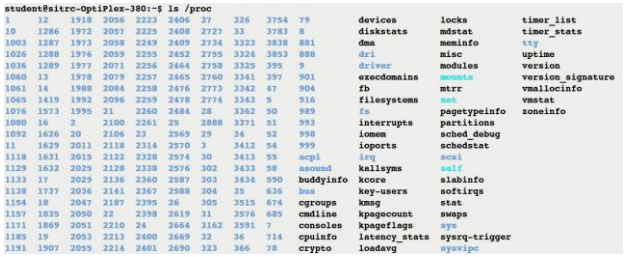

Figure 3. /proc directory

As in the above figure, all the blue color numbers represent the directories which contain the complete information about the processes. This number is identical to the process_id of the processes in execution.

We can obtain information about each and every process from the directories of the process. Like in the following figure, it shows the status of the particular process_id 3576.

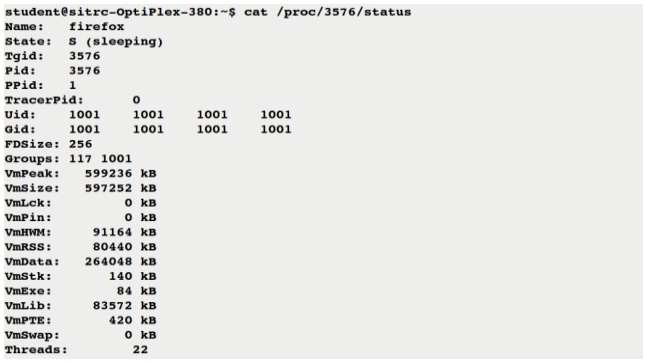

Figure 4. /proc/process_id/status

If we want to look for the users of the system, the registry of those user is present in the shadow file of the /etc directory. In that we can identify the invalid user, if any. In the following figure we can see that the users present in the system like daemon, bin, sys etc. are shown. In that one invalid user has been detected which can be malicious. This invalid user is being hidden from the system and which are rarely being identified.

Figure 5. Invalid User

IV. CONCLUSION

In this project, we propose in-execution detection scheme that detects a malicious process during its execution and subsequently kill them on Linux platform. As a result, our framework can be embedded in the kernel of Linux. Moreover, our features are system specific and we must estimate them for benign and malware processes in each system. The features can only be extracted by a super user; as a result, it becomes difficult to evade our detection system.

ACKNOWLEDGMENT

We take this opportunity to express our sincere gratitude towards our Project Guide Prof. Tushar B Kute for his generous assistance, without whose considerate approach and insight, this paper would never have been possible. We would like to thank the staff members of SITRC who helped us to select this topic and prepare this paper. We extend our sincere thanks to our college library staff that extended their full cooperation for obtaining the necessary material for this report. Our whole hearted thanks to all our friends who helped us by giving their valuable suggestions in preparing the paper.

.REFERENCES

[1] Wu Kehe and Ge Yueguang, Chen Wei, Zhang Tong, *The research and implementation of the; Linux process real-time monitoring technology*, 2012 Fourth International Conference on Computational and Information Sciences.

[2] JFarrukh Shahzad, Sohail Bhatti, Muhammad Shahzad and Muddassar Farooq, *In Execution Malware Detection using Task Structures of Linux Processes*. 2011 International Conference on Computational and Information Sciences.

[3] Christopher Kruegel Technical University Vienna chris@auto.tuwien.ac.at,Willam Robertson and Giovanni Vigna Reliable Software Group University of California, Santa Barbara wkr,vigna@cs.ucsb.edu;*"Detecting Kernel-Level Rootkit through binary analysis";* Proceeding of the 20th Annual Computer Security Application Conference.

[4] Gerard Wagener Radu State Alexender Dulaunoy; *"Malware Behavior And Analysis";* Received:1 July

2007/Revised:11 November 2007/Accepted:21 November 2007 Springer-Verlag France 2007.

[5] Mamoun Alazab,Sitalakshmi Venkataramana, Paul Watters, and Moutaz Alazab; *"Zero-day Malware Detection based on supervised learning Algorithm of API call Signatures";* Proceeding of the 9th Australian DataMining Conference.

[6] Konrad Rieck, Thorsten Holz, Carsten Willems , Patrick D¨ssel1 , and Pavel Laskov1;*"*
Learning and Classification of Malware Behavior ".

[7] Linux.Kernel.Development.3rd.Edition;