MALWARE CLEARANCE FOR SECURE COMMITMENT OF OS-LEVEL VIRTUAL MACHINE

A.VIDHYA[*]
R.RAJESWARI[**]

Abstract— The number and complexity of attacks on computer systems are increasing. This growth necessitates proper defense mechanisms. Intrusion detection systems play an important role in detecting and disrupting attacks before they can compromise software. The Secom prototype can effectively eliminate malicious state changes while committing a VM with small performance degradation. The Secom prototype has a smaller number of false negatives and thus can more thoroughly clean up malware side effects. In addition, the number of false positives of the Secom prototype is also lower than that achieved by the online behavior-based approach of the commercial tools. Multivariant execution is an intrusion detection mechanism that executes several slightly different versions, called variants,of the same program in lockstep. The variants are built to have identical behavior under normal execution conditions. However, when the variants are under attack, there are detectable differences in their execution behavior. At runtime, a monitor compares the behavior of the variants at certain synchronization points and raises an alarm when a discrepancy is detected. The project presents a monitoring mechanism that does not need any kernel privileges to supervise the variants. Many sources of inconsistencies, including asynchronous signals and scheduling of multithreaded or multiprocess applications, can cause divergence in behavior of variants. These divergences cause false alarms.

Keywords— False positives, Kernel privileges, Malicious state changes, Multivariant execution.

[*] Currently pursuing masters degree program in computer science engineering.

[**] M.E.,(Ph.D).,Assistant Professor of CSE Dept

I. INTRODUCTION

An OS-level VM has minimal startup/shutdown cost, low resource requirement and high scalability due to its sharing of the execution environment of the host operating system and confining state changes within the VM's environment. A user has to redo all the work in her actual workspace since there is no secure commitment mechanism to save the benign changes within the VM back to the host environment. Secure commitment means merging only benign changes into the host environment but filtering out malicious changes when committing a VM.

The main aim to provide a lightweight commitment approach, named Secom, for an OS-level virtual machine to prevent malicious changes from being merged into the host.
To our best knowledge, this is the first effort toward building a secure and practical VM commitment mechanism. Secom focuses on filtering out malware impacts which impose a severe security threat to computers. As the issue of resolving inconsistency from concurrent modifications of the same resource by multiple processes running within different VMs . The approach consists of three stages. it correlates suspicious objects within a VM into a number of clusters by tracking OS-level information flows and attaching a cluster label to each object. Objects in a cluster are only possible to be either all benign or all malicious, it determines malicious clusters using an online malware detection engine to monitor malicious behaviors, it merges benign changes in a VM to the host environment while discarding malicious clusters.

2. RELATED WORKS

The malware make the changes within an OS-level VM include files, directories, and registry entries that are created, modified, and deleted by the processes running in the VM. So we need the secure commitment. Example applications of such secure commitment include, Committing running results of enterprise applications. Synchronizing states of fault tolerant applications. Committing working results of a user. Committing installation results of untrusted software. The objectives are data collection is easy. The clustering of suspicious object and malicious object is very easy. Less time is required for data consolidation. To reduce the false-positive rate. Since the application is web service oriented and platform independent data from various data sources are possible. To detect multiple malware behavior at simultaneously. Eliminate malicious state changes. Its three novel features allow it to complete the task in a lightweight but efficient manner.

The clustering approach to differentiate the benign and malicious changes within a system. The approach trace OS-level information flows to gathered the changes, and group gathered the changes into clusters. . It traces only suspicious processes and executable files that represent malware program program themselves rather than tracing all kinds of files, directories, and IPC objects. This can avoid the disadvantages of a classical malware tracing approach, which imposes a heavy performance impact on the system or makes the entire system floating with suspicious labels.

It proposes a new behavior-based malware detection approach. A suspicious cluster is considered to be malicious only when it exhibits at least two types of malware behaviors. It Checks for malware program existence in single machine only.It requires a training stage to generate a remediation program for cleaning the impacts of a specific malware program. It cannot control the file, folders or registry entries created by malware. A user has to redo all the work in actual workspace. It's only on tracing OS-level information flows and monitoring malware behaviors. Secom focuses on filtering out malware impacts which impose a severe security threat to computers.

2.1 PROPOSED WORK

Security vulnerabilities in software have been a significant problem for the computer industry for decades. While the use of safer programming languages such as Java and C# has alleviated the problem, there are still many software packages that are created and maintained in C and C++. This is primarily driven by concerns about performance and access to low-level constructs, which is not always possible in languages executed in a managed environment.

On the other hand, writing safe and secure programs in C and C++ is difficult, despite an increase in education and the availability of safer APIs designed to help detect errors. As a result, the challenge of finding mechanisms to detect and remove vulnerabilities persists. With the large amount of code written every year, it should be noted that despite the fact that the vulnerability density is decreasing, the overall number of vulnerabilities is increasing.

For example, the number of buffer errors listed by the statistics feature of the National Vulnerabilities Database at the time of writing increased from 409 in 2007 to 563 in 2008, and 565 in 2009. Many techniques have been developed to eliminate vulnerabilities, but none of them provides a complete solution.

Modern static analysis tools are capable of finding many varieties of programming errors, but a lack of runtime information limits their abilities. Some also have a relatively high false positive rate, making them expensive to use in practice. Dynamic and runtime tools are often not effective because they lack a baseline to use for detection. Also, the performance overhead of sophisticated algorithms used by such runtime tools is often prohibitively high in some production systems.

Multivariant code execution is a runtime monitoring technique that prevents system damage resulting from malicious code execution and addresses the above problems with dynamic detection tools. Multivariant execution protects against malicious code execution attacks by running two or more slightly different versions of the same program, called variants, in lockstep. At defined synchronization points, the variants' behavior is compared against each other. Divergence among the behavior is an indication of an anomaly and raises an alarm.

An obvious drawback of multivariant execution is the extra processing overhead, since at least two variants of the same program must be executed in lockstep to provide the benefits mentioned above. Our experimental results show that this overhead is in the range afforded by most security sensitive applications where performance is not the first priority, such as government and banking software. Besides, the large amount of parallelism that inherently exists in multivariant execution helps it take advantage of multicore processors.

Currently, cores are often idle due to the lack of extractable parallelism in many applications or due to the bottlenecks imposed by memory or I/O devices. Moreover, the number of cores is increasing rapidly. For instance, Intel has promised 80 cores by 201. A multivariant execution environment (MVEE) can engage the idle cores in these systems to improve security with little performance overhead.

3 SYSTEM ARCHITECTURE

One malicious process is creating files randomly in a shared folder. Other malicious process is creating folder is created with random numbers as name. Another malicious process in creating registry key is random names. The malware programs are allowed to execute in the system and this module watches the programs so that their activities are tracked out. Then the related processes are grouped into clusters.The processes grouped into malicious cluster are found out and listed. The files created by the processes, folders and registry entries are deleted in this module.The client node's ID, IP address and system name are keyed in and stored in the database table. The start monitor starts running, the server application is activated and task can be assigned as well as records, nodes list, task list and attacks list can be viewed.The start self defense starts running, the server self defence activity is made such that the files in the server shared folder is not affected by the client nodes. Assign code execution task to nodes is, the node is selected, an executable file is copied to the node's share folder and a number of parameters are saved in database table. After find the malware it will be delete.

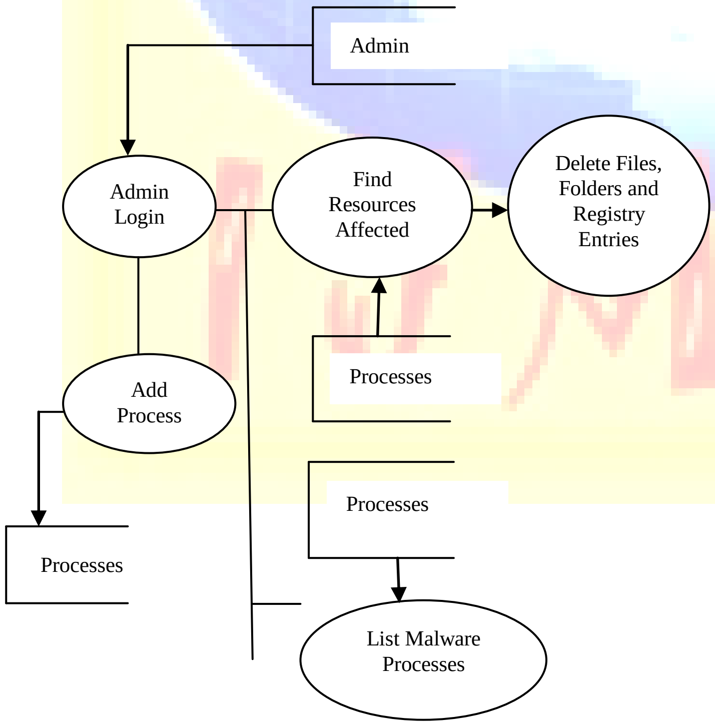


Fig .3.1 Architecture

4 MECHANISM

SECOM approach consists of three stages:

- It correlates suspicious objects within a VM into a number of clusters by tracking OS-level information flows and attaching a cluster label to each object. Objects in a cluster are only possible to be either all benign or all malicious.
- It determines malicious clusters using an online malware detection engine to monitor malicious behaviors.
- Last, it merges benign changes in a VM to the host environment while discarding malicious clusters.

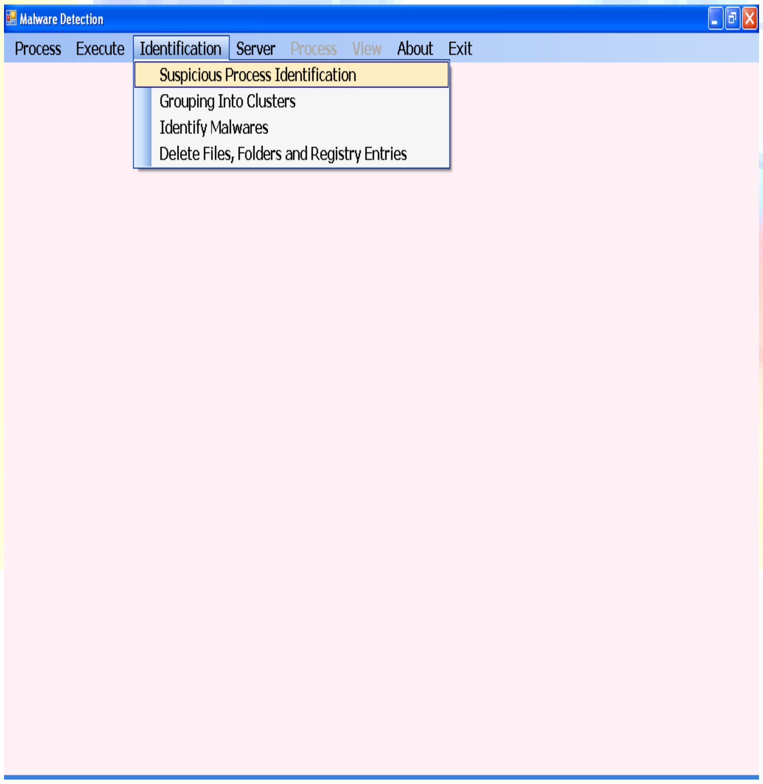


Fig 4.1: Malware clustering process

5 EXPERIMENTAL RESULTS

The virtual machine connected to the LAN. The server machine assingn the task to client and check whether the result is right or wrong. If the result is right there is no malware attack. If the result is wrong there will be malware attack. After find the malware it will deleted. This action perform by the server.

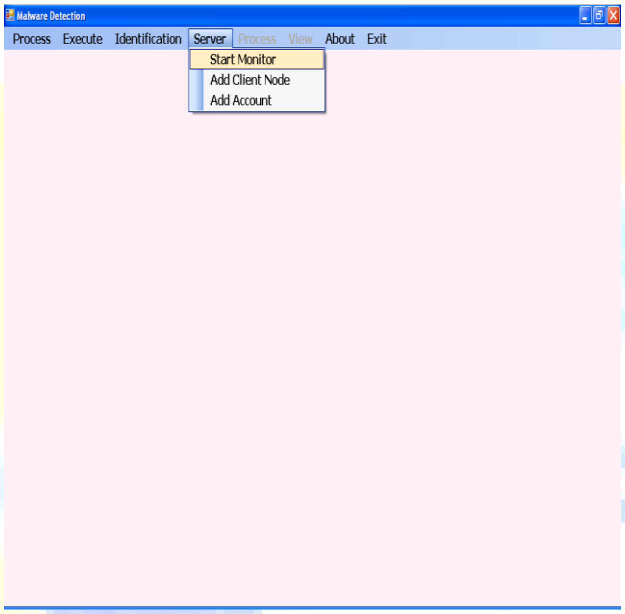
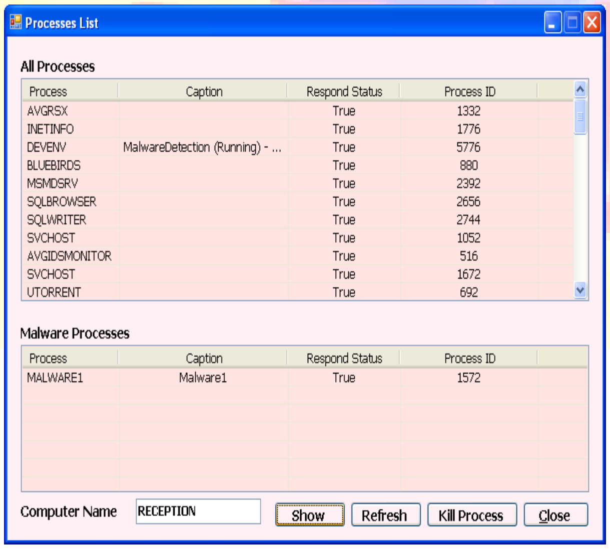


Fig: 5.1 Add the client nodes



All Processes

Process	Caption	Respond Status	Process ID
AVGRSX		True	1332
INETINFO		True	1776
DEVENV	MalwareDetection (Running) - ...	True	5776
BLUEBIRDS		True	880
MSMDSRV		True	2392
SQLBROWSER		True	2656
SQLWRITER		True	2744
SVCHOST		True	1052
AVGIDSMONITOR		True	516
SVCHOST		True	1672
UTORRENT		True	692

Malware Processes

Process	Caption	Respond Status	Process ID
MALWARE1	Malware1	True	1572

Computer Name: RECEPTION

Fig: 5.2 Delete the malware

VI CONCLUSION AND FUTURE WORK

This project proposes Secom, a scheme toward securely committing OS-level virtual machines, which is required by intrusion-tolerant applications and system administrations to save benign changes within a VM to the host environment. The critical challenge behind securely committing VM is to identify compromised objects thoroughly and lightly. To address the challenge, this project proposes Secom that consists of three steps.

First, it correlates suspicious OS objects within a VM together by tracking OS-level information flows and grouping them into clusters by intelligently attaching different labels to objects. Then, it recognizes a malicious cluster by a behavior-based malware detection engine. Last, it commits VM while discarding malicious clusters. Secom has three novel features. First, Secom can lightly commit VM using OS-level information flows and malware behaviors. Second, Secom detects and discards malicious changes in a cluster fashion to clean up malware impacts quickly and thoroughly. Finally, to reduce the false-positive rate, Secom considers two malware behaviors which are of different types and the originator of the processes which exhibit the behaviors when identifying a malicious cluster.

In addition, the proposed multi-variant execution environment runs multiple versions of a program simultaneously and monitors their behavior. Instead of finding and removing the vulnerabilities, the new method accepts the inevitable existence of vulnerabilities and prevents their exploitations.

In future, false position rate deduction can be studied in mis-identifying the executable files as malware or code-injected.

References

[1] S.T. King and P.M. Chen (2003) “Backtracking Intrusions”, Proc. ACM Symp. Operating Systems Principles (SOSP), pp. 223-236.

[2] D. Price and A. Tucker (2004) “Solaris Zones: Operating System Support for Consolidating Commercial Workloads”, Proc. 18th Large Installation System Administration Conf., pp. 241-254.

[3] Poul-Henning Kamp, Robert N. M. Watson (2000) “Jails: Confining the omnipotent root”, The FreeBSD Project.

[4] R. Paleari, L. Martignoni, E. Passerini, D. Davidson, M. Fredrikson, J. Giffin, and S. Jha (2010) “Automatic Generation of Remediation Procedures for Malware”, Proc. USENIX Conf. Security.

[5] W. Sun, R. Sekar, and V.N. Venkatakrishnan (2005) “One-Way Isolation: An Effective Approach for Realizing Safe Execution Environments”, Proc. 12th ISOC Network and Distributed Systems Symp. (NDSS), pp. 265-278.

[6] S. Soltesz, H. Po¨tzl, M.E. Fiuczynski, A. Bavier, and L. Peterson (2007) “Container-Based Operating System Virtualization: A Scalable, High-Performance Alternative to Hypervisors”, Proc. Second ACM European Conf. Computer Systems.

[7] Y. Yu, F. Guo, S. Nanda, L. Lam, and T. Chiueh (2006) “A Feather-Weight Virtual Machine for Windows Applications”, Proc.Second Int’l Conf. Virtual Execution Environments (VEE), pp. 24-34.

[8] Zhendong Su, Chen, Hsu, Jason Li, Ristenpart, (2005) “Back to the Future: A Framework for Automatic Malware Removal and System Repair”, Proc.University of California, Davis.

Author's Detail:

[1]Vidhya.A received degree B.Tech from Sasurie Engineering College , Anna university in 2011. Now pursuing M.E Computer Science and Engineering in MPNMJ Engineering College, Anna university,

[2]Rajeswari.R received the M.E degree in computer science engineering from vellalar college of engineering & Technology in 2009.Currently, pursuing Ph.D in Anna university,Chennai. Currently, she is an assistant professor in the computer science department, MPNMJ Engineering College, Anna university.