Indian Journal of Research in Applied Sciences Engineering (IJRASE)
Vol.2.No.1 2014pp14-17.
available at: www.goniv.com
Paper Received :05-03-2014
Paper Published:28-03-2014
Paper Reviewed by: 1. John Arhter 2. Hendry Goyal
Editor : Prof. P.Muthukumar

THE SECURE COMMITMENT PROTOTYPE FOR VIRTUAL MACHINES TO IDENTIFY STATE OF THE MALICIOUS BEHAVIOR

A.REVATHY, **Dr.M.SENTHIL**,
Department Of CSE
S.K.P Engineering College
Tiruvannamalai, India
revathycse11@gmail.com senthil.wiseman@gmail.com

ABSTRACT

The virtual machine provides the software implementation process where the task created and closed. Basically virtual machine has two types that is process virtual machine and system virtual machine. It provides the environment for single program execution and system platform. The problem of virtual machine is there is no malicious activity detection only the benign updates send to the vm host environment. In this paper we propose the VM commitment system called Secom for provide automatic elimination of state changes. State elimination processed in os level information flow. It consist of three steps there are grouping state changes into cluster, differentiate between benign and malicious state and commit the benign cluster. Mainly secom has three novel features first one is use the os level information flow instead of huge log data , it detect compromised os object one by one and convert object into cluster then identify malicious .Then finally we filter the particular malware code in this clustered flow then update it to host. It reduces the false-positive rate and information flow loss when identifying malicious clusters.

1. INTRODUCTION

Malware is a critical problem in OS-Level virtual machine operating system. NET-WORM, EMAIL-WORM, P2PWORM,IM WORM,IRC WORM from these resources the files get corrupted. Anti-Virus companies receive upto thousands of new malware samples every day. The financial loss caused by malware has been as high as 14.2 billion US dollars in the year 2005[1].

It can able to only remove a piece of Malware in the remediation process. By monitoring each URL List, we discover a malicious. From a URL List, it drives browser to visit each URL and detects browser exploits by monitoring unexpected state changes. In the SECOM[4] approach, normally it compares the benign changes and malicious changes. It implements the concept that was filtering the malicious changes and committing the benign cluster. In this approach it only detects the known harmful virus.

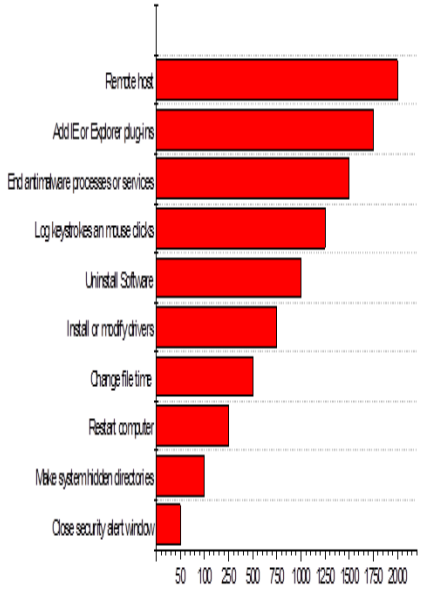
To overcome this problem, we are implementing TTAnalyze tool that directly interacted to the operating system to rapidly analyze an unknown malware sample and understood its behavior to protect the system. TTAnalyze detects the malicious program from the internet and automatically delete the malware. We are using this tool for dynamically analyzing the remediation procedure. We can rectify the state of the affected malware by three ways, detecting and tracing the malware from the core, resources may be completely removed from the system, and scalable website testing.

2. RELATED WORK

MALWARE BEHAVIOUR: This system proposes a novel MAC model called Tracer. Mandatory Access Control traces all critical malware samples. Tracer does not confine the suspected intruders. We are analyzing 10 critical real world malware behaviors by implementing tracer in Windows OS. The disadvantage of this system have higher false positive rate to our system. By analyzing remote host is severely attacking by malware.

Here we are using tracer to detect malware in Windows,that shows how severely affect each application.

BACKTRACING INTRUSION: The virtual



machine shatters some attacks by shrinking the virtual address space.[7] We use backtracker to analyze several real attacks against computers. Backtracking is used torectify the malware in an operating system.

REMUS: Remus provides an extremely high degree fault tolerance. By using this Remus technique, our system discretizes the execution of a VM into series of replicated snapshots[8]. It does not guarantee that the process will be effective and efficient. In our system we can ale to produce frequently running OS to a physical machine with high frequencies.

LEARNING AND CLASSIFICATION OF MALWARE BEHAVIOUR: Here we are finding the ratio between intra-cluster and inter-cluster variation. The clustering algorithm generated from many malware samples in single executable[9]. Malware Binaries are executed and monitored in a Sandbox environment; it would be an accurate automates analysis of malware behavior.

3. IMPLEMENTATION

System model

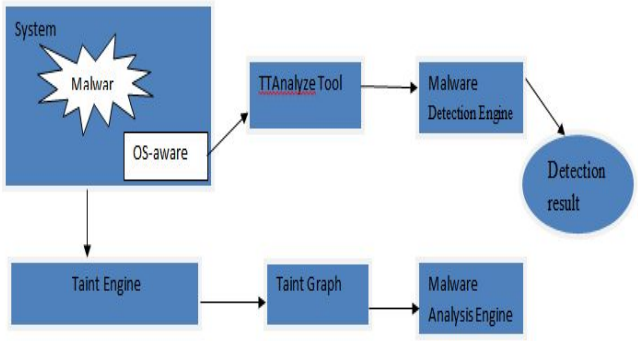


Fig 1.Automated Testing in Taint Engine

Table 1. TTAnalyze malware detection

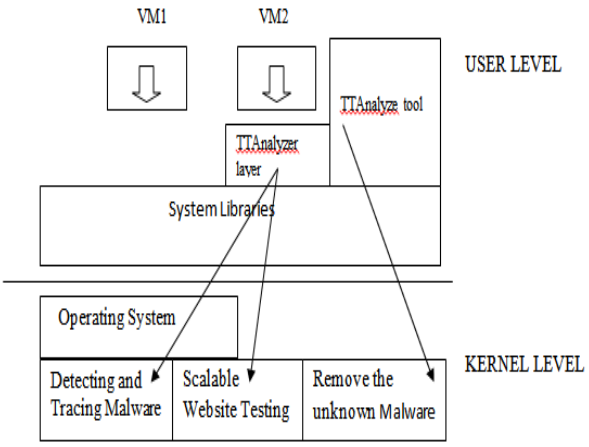
Malware name	File	Registry	Process	Service
Email-Worm.Win32.Doombot.B	no	No	No	no
Email-Worm.Win32.Netsky.B	yes	Yes	Yes	yes
Email-Worm.Win32.Netsky.D	yes	Yes	Yes	yes
Email-Worm.Win32.Netsky.Q	yes	Yes	Yes	yes
Email-Worm.Win32.Sober.Y	yes	Fail	Yes	yes
Email-Worm.Win32.Zafi.D	yes	Fail	Yes	yes
Net-Worm.Win32.Mytob.BD	no	No	No	no
Net-Worm.Win32.Mytob.BK	yes	Yes	Yes	yes
Net-Worm.Win32.Mytob.C	yes	Fail	Yes	yes
Net-Worm.Win32.Mytob.J	no	No	No	no

We propose two specific types of test cases used in our experiment. We need to monitor show tainted data propagates throughout the whole system including the OS with the TTAnalyze tool. All sensitive information that is introduced into the system in the automated tests is marked as a taint source [10]. Maintaining a mapping between addresses in memory and modules requires information from the guest operating system. Taint Engine is used to detect the malware and the TTAnalyze tool is used to clean the malware. The

implementation of the system is mainly for malware clearance by direct connection with operating system. In Fig 2. The system tested with two cases, first with the Taint Engine it produces a graph that performs detection of malware analysis using Malware Analysis engine. In the second cases with the TTAnaltze tool that detect using Malware detection engine and also clear the malware and produce the result that shows in the following table. In the table consist of File, Registry, Process and Service.

The implementation of the system is mainly for malware clearance by direct connection with operating system. In Fig 2. The system tested with two cases, first with the Taint Engine it produces a graph that performs detection of malware analysis using Malware Analysis engine. In the second cases with the TTAnaltze tool that detect using Malware detection engine and also clear the malware and produce the result that shows in the following table. In the table consist of File, Registry, Process and Service.

4. ARCHITECTURE:



To demonstrate the suitable TTAnalyze approach, we have successfully developed a TTAnalyzer layer. This layer consists of kernel level and user level. We are using two Virtual Machine for this implementation. TTAnalyze tool use to directly remove the Malware from the core of the resources in the system.

5. EXPERIMENTAL SETUP:

Our experiments were performed a overall execution overhead of real world applications. WE have tested two machines. Machine 1 contains a Intel Core 3, 2-Ghz CPU with 2-GB memory, an runs webserver an Local host. Machine 2 contains an Intel Pentium -4 2-GHz CPU with 512-MB memory an runs with the real world applications. We installed Windows 7 on both Machines. We carry out an experiment to measure the overhead of system interruption. To ingress the performance impact of TTAnalyze on network-facing server applications, we measure the throughput of many webserver. The applications WinZip 32, BCC 32 are running in Machine1.Telnet cd, Telnet data are running in Machine 2. In Fig.4 We can see that TTAnalyze imposes more overhead on the system compared to that of Secom Approach and Feather Weight Virtual Machine. We measured the performance using Web bench, Win stone and Net Bench. Finally all experimental results demonstrate that enforcing TTAnalyze acquire a sufficient performance overhead on Windows OS.

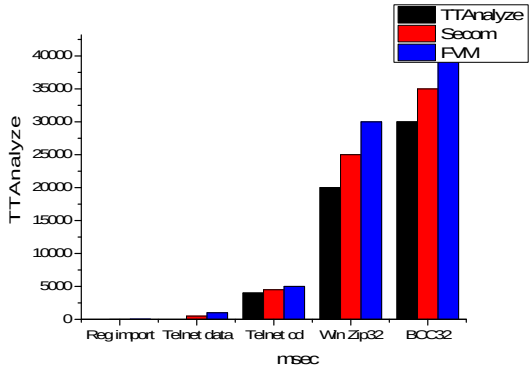


Fig 3. Performance of TTAmalyzer in independent application

It solving the problem of system getting slow down and smoothly runs two OS in the system.

6. CONCLUSION:

Although a considerable amount of research effort has gone into MALWARE clearance, malicious code is an important threat on the Internet. In this paper, we propose 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. By analyzing TTAnalyze tool that dynamically analyzing the flow which present in the particular cluster. This approach shares a peculiar feature and performs the operation. The critical challenge in the Existing System is difficult to detect the unknown malware. To address this challenge, we propose a TTAnalyze tool for quickly getting an understanding of the behavior of unknown malware. We analyzing the malware by 3

ways; detecting and tracing the malware from the core, scalable website testing and resources may be completely removed from the system. Compared with other approach this can remove the malware completely and produce a lower false-positive rate.

REFERENCES

[1]. R. Paleari, L. Martignoni, E. Passerini, D. Davidson, M.Fredrikson, J. Giffin, and S. Jha, "Automatic Generation of Remediation Procedures for Malware," Proc. USENIX Conf. Security, Aug. 2010.

[2]. Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski,
Shuo Chen, and Sam King. Automated Web Patrol with Strider
HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities.
In Proceedings of 13th Annual Network and Distributed System
Security Symposium, February 2006.

[3]. Y. Yu, H.K. Govindarajan, L. Lam, and T. Chiueh, "Applications of Feather-Weight Virtual Machine," Proc. Int'l Conf. Virtual Execution Environments (VEE), Mar. 2008.

[4]. U. Bayer, C. Kruegel, and E. Kirda. TTAnalyze: A tool for analyzing malware. In 15th European Institute for Computer Antivirus Research (EICAR) Annual Conference, Hamburg, Germany, Apr. 2006.

[5]. Malware Clearance for Secure Commitment of OS-Level Virtual Machines Zhiyong Shan, Xin Wang, and Tzi-cker Chiueh

[6]. Z. Shan, X. Wang, and T. Chiueh, "Tracer: Enforcing Mandatory Access Control in Commodity OS with the Support of Light- Weight Intrusion Detection and Tracing," Proc. Sixth ACM Symp. Information, Computer and Comm. Security (ASIACCS), pp. 135-144 Mar. 2011.

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

[8]. B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A.Warfield, "Remus: High Availability via Asynchronous Virtual Machine Replication," Proc. Fifth USENIX Symp. Networked Systems Design and Implementation (NSDI), 2008.

[9] K. Rieck, T. Holz, C. Willems, P. Du¨ ssel, and P. Laskov, "Learning and Classification of Malware Behavior," Proc. Fifth Int'l Conf.Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), pp. 108-125, June 2008.

[10]. H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda, "Panorama:Capturing System-Wide Information Flow for Malware Detection and Analysis," Proc. 14th ACM Conf. Computer and Comm. Security (CCS), 2007.

[11]. Y.-M. Wang, R. Roussev, C. Verbowski, A. Johnson, M.-W. Wu, Y.Huang, and S.-Y. Kuo, "Gatekeeper: Monitoring Auto-Start Extensibility Points (ASEPs) for Spyware Management," Proc. 18th USENIX Conf. System Administration, 2004.

[12]. M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda, "Defending Browsers against Drive-By Downloads: Mitigating Heap-Spraying Code Injection Attacks," Proc. Sixth Int'l Conf. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), July 2009.

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

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

[15]. M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda, "Defending Browsers against Drive-By Downloads: Mitigating Heap-Spraying Code Injection Attacks," Proc. Sixth Int'l Conf. Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), July 2009.