IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 9, Issue 3 (Mar. - Apr. 2013), PP 11-15
www.iosrjournals.org

Execution of Critical Application in Corrupted Environment Using Vmm

S.Valarmathi[1], Mr.S.Sathishkumar[2]
[1]*M.E. (Cse), Srinivasan Engg College, Perambalur,Tamilnadu,India.*
[2]*AP/ It, Srinivasan Engg College, Perambalur,Tamilnadu,India.*

Abstract: *A resilient execution environment can be developed for a critical application even in the presence of corrupted OS kernel. The attacker tries to capture the application content by corrupting the OS when an application is executing. In previous case the attacker corrupts the OS by injecting code then the application terminates immediately without executing it. In this current system even in the presence of corruption the application is executed with out any interception and it provide a resilient authenticated execution of critical application in entrusted environment by using Virtual Machine Monitor (VMM). VMM is a monitoring technique to monitor all the activities during execution and it is one of the online based recovery schemes to identify any such corruption. It repairs the memory corruption and allows the process for normal execution. VMM solutions generally broadcast into two categories they are memory authentication and memory duplication. Memory authentication is to check the integrity of an application and memory duplication is to rectify the corruption. The proposed system is applied for military application, hospitals, and colleges and for all critical applications.*

Keywords*- Memory corruption, operating systems, security, Virtual machine monitors (VMM).*

I. Introduction

Computing platforms are encompassing an ever-growing range of applications supporting an ever-growing range of hardware and providing tremendous functionality. In many critical applications such as military, college, health or infrastructure monitoring software are highly critical compromising their correct execution may have dire consequences. Typically such applications are originate at trustworthy sources and they undergo a thorough testing process possibly including formal verification. Most software applications lack the ability to repair themselves during an attack especially when attacks are delivered via previously unseen inputs or exploit previously unknown vulnerabilities. Security vulnerabilities in the entrusted applications and OS destroy the guarantees of isolation and may lead to compromise of the trusted code. Memory corruption attacks are among the most frequently occurring security violations accounting for some percent of the total number of Memory attacks can be especially devastating if the target is the OS itself. If the critical application computes missile trajectories for a military operation the adversary's goal may be simply to skew the results. These systems recognize a threat or attack has occurred orient the system to this threat by analyzing it to adapt the threat by constructing appropriate or changes in state respond to the threat by verifying and deploying those adaptations. Many techniques have been developed to eliminate vulnerabilities but none of them provide an ultimate solution. New static-analysis tools are capable of finding many programming errors but lack of run-time information limits their capabilities preventing them from finding all errors. It also produces large number of false positives making them expensive to deploy in practice. Dynamic and run-time tools are often not effective either because they do not have a reference for comparison in order to detect misbehavior. Multi-variant code execution is a run-time monitoring technique that prevents malicious code execution and addresses the problems mentioned above. Vulnerabilities that allow the injection of malicious code are among the most dangerous form of security flaws since they allow attackers to gain complete control over the targeted system. Multi-variant execution environment protects against execution of malicious code by running two or more slightly different variants of the same program in lock step. Many programs that are executed on modern multi-core and multi-processor systems are sequential and thus cannot exploit the parallel features of the hardware they are running on. Rollback recovery treats a distributed system as a collection of application processes that communicate through a network. Fault tolerance is achieved by using stable storage to save the processes states during failure-free execution. In case of failure a failed process restarts from one of its saved states thereby reducing the amount of lost computation.

II. Problem Statement And Challenges

VMM is used to monitor the critical application. It executes the application even in the presence of OS corruption. The attacker try to corrupt the OS to capture the application running in kernel OS. Intruder add any code in the kernel OS to capture the application content. In previous system if any of this corruption is detected it terminate the application instead of correcting them. It leads to loss in the application data it is also one type of denial of service attack. The goal of an attacker is to capture the content of the application or terminate the application running in kernel OS. In previous system application based check point is used to recover the application from an error that is during the execution of an application there are several checkpoints in between that once the application reaches the checkpoint it get saved in that particular point before reaching the next checkpoint if any error is detected mean it is recovered from the previous saved checkpoint it leads to loss of time and more computation is needed it is not sure that it is fully recovered. In this system it is recovered from last checkpoint that is from the previous state before an attack is injected. It also provides an error recovery mechanism to recover from that error. VMM is used to monitor the application during the time of execution and it provides an recovery mechanism to recover from that error.

III. Related Work

3.1 Multi-VariantExecution Environment

Multi-variant code execution is a run-time monitoring technique that prevents malicious code execution and addresses many problems. Multi-variant execution protects application against malicious code execution attacks by running two or more slightly different variants of the same program in lock step. At certain synchronization points their behavior is compared against each other. Variation among the behavior of the variants is an indication of an anomaly in the system and raises an alarm. A multi-variant execution environment (MVEE) can engage the idle cores in these systems to improve security with little performance overhead [9].In a Multi-Variant Execution Environment several slightly different versions of the same program are executed in particular lockstep. While this is done a monitoring technique compares the behavior of the versions at certain synchronization points with the aim of detecting discrepancies which may indicate attacks. The monitor can be implemented entirely in user space eliminating the need for kernel modifications. A fully functioning MVEE named Orchestra and evaluated its effectiveness. The results show that the overall performance of fast execution and monitoring of two variants on a multi core system to unprotected conventional execution.

3.2 Competitive Parallel Execution

Competitive parallel execution (CPE) an approach to leverage multi-processor and multi core systems for the execution of sequential programs. CPE is a technique for modifying and executing existing sequential applications to increase their performance on parallel systems [12]. The central idea of CPE is to facilitate the introduction of multiple variants for parts of a program where different variants are suited for different run-time conditions. Dynamic factors such as input data or the characteristics of the platform the program is executed on determine which variant executes the fastest. Instead of trying to determine a single best strategy offline before program execution, a program contains different variants that compete against each other at run-time. The purpose of creating variants is to make the program adaptive to variable run-time conditions. A CPE-aware run-time system is responsible for orchestrating the execution of the variant-augmented program on the available processor cores. The goal is thereby that the program progresses at the rate of the fastest variant for each execution phase. Competitive parallel execution is a simple yet attractive technique to improve the efficiency of sequential programs on multi-core and multi-processor systems. A sequential program is transformed as CPE-enabled program by introducing multiple variants for a program. The run-time system ensures that the behavior and outcome of a CPE-enabled program is not distinguishable from the one of its original sequential counterpart. It evaluate a run-time system that is implemented as a user-space library and that closely interacts with the operating system. It discusses two strategies for the generation of variants and investigates the applicability of CPE for two usage scenarios one is computation-driven CPE a simple and straightforward parallelization of heuristic algorithms and another one is compiler-driven CPE: generation of CPE-enabled programs as part of the compilation process using different optimization strategies.

IV. Functional Design And Analysis

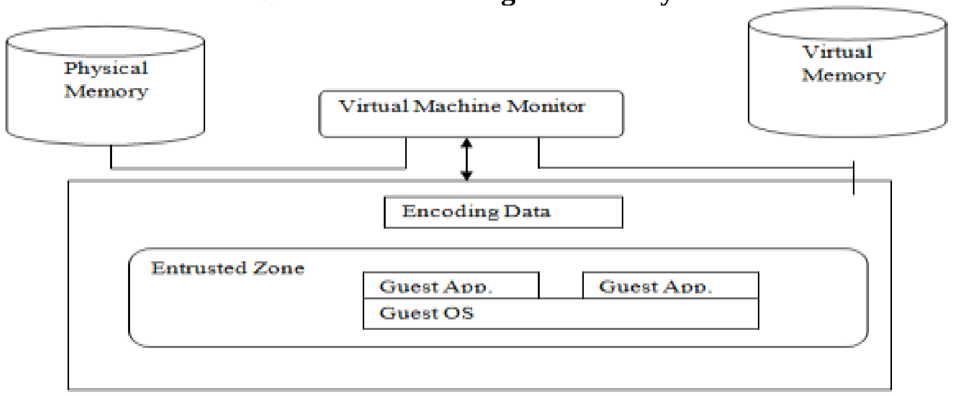

Fig 1: Overview architecture

Four set of functional steps needed as follows
Project is going to be divided into the following modules.
1) Memory Authentication
2) OS Corruption
i) VMM
ii) DOS attack
3) Communication Overhead
4) Reed-Solomon Code-Based Approach

4.1 Memory authentication

In this category a trusted component that is the VMM applies cryptographic techniques to validate the integrity of the application memory image during execution. It is used to check if there is any change happened in an application by using snapshot taken when an application switches from user mode to kernel mode. It also checks the integrity of an application. During the time of switching from user mode to kernel mode encoding can be take place. It is used protect the application content from unauthorized user.

4.2 OS corruption

4.2.1 VMM

Virtual Machine Monitor is a Monitoring technique to monitor all the activities during execution. It has access to the physical computer processor and manages resources between the two environments preventing malicious or poorly designed applications running in a guest operating system from requesting excessive hardware resources from the host operating system.

4.2.2 Denial of service

The system assumes that the guest OS may be compromised, and assuming that it is still somewhat functional. That is if the attacker’s goal is merely to shut down the system or corrupt the OS beyond repair. This attack is outside scope as focusing the scenarios where the attacker’s goal is to keep the system working in a compromised manner. A more subtle attack could aim to prevent the OS from scheduling the trusted application. To defend against this attack it suggests a technique similar to a watchdog timer technique. The application may send a report to an external monitor at least once within a certain time period. At that point external administration would be required to restart or repair the corruption. To prevent the guest OS from forging such reports cryptographic techniques can be applied to network data.

4.3 Communication overhead

Every virtual machine running on an host consumes some memory overhead additional to the current usage of its configured memory. This extra space is needed by exchange for the internal VM kernel data structures like virtual machine frame buffer and mapping table for memory translation (mapping physical virtual machine memory to machine memory).During the time of translation that is from physical memory to virtual memory for checking the integrity of the application that leads to communication overhead. Snapshot of the application is taken during translation to check the integrity of the application

4.4 Reed-Solomon Code-Based Approach

Reed-Solomon (RS) codes is based on using multiple trials of a simple RS decoding algorithm in combination with erasing or flipping a set of symbols or bits in each trial. This technique presents a framework based on rate-distortion (RD) theory to analyze these multiple-decoding algorithm’s code is an error correcting mechanism and it is one of online error correcting technique to correct the error occurred during execution of an application. By defining an appropriate distortion measure between an error pattern and an erasure code the successful decoding condition for a single errors and erasure decoding trial becomes equivalent to distortion being less than a fixed threshold. Finding the exact set of erasure patterns also turns into a covering problem that can be solved asymptotically by RD theory.

V. Experimental Result

This experiment shows that our approach is practical and could be used in a real deployment for that VMM can be used to monitor the entire application and it repair the corrupted application memory image by using error recovery code. The empirical results show that both the memory and performance overhead imposed by this design is reasonable.

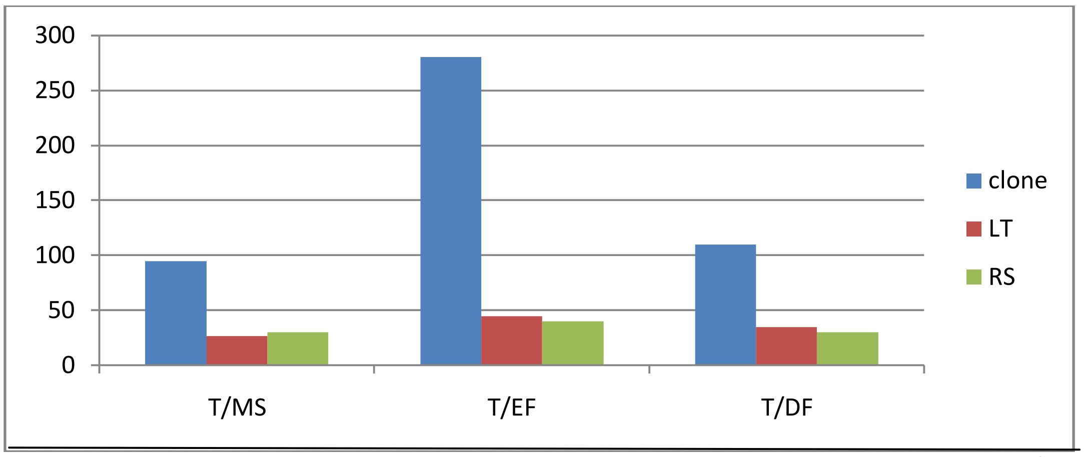

Fig (2) throughout this experiment, it is tried that how to protect a critical application even in the presence of entrusted environment. For this virtual machine monitoring technique is used it monitor both the application and OS.

VI. Conclusion

VMM (Virtual Machine Monitor) is a monitoring technique to provide resilient authenticated execution for critical applications. It monitors the entire application and protects the application from intruders. If it detects any corruption in an application, it recovers the application from that corruption by using error recovery mechanism like LT (Luby Transform) and RS (Reed Solomon) code and also it protect the application from memory corruption attack. In future VMM with overshadow technique can be used to provide confidentiality and authentication for a critical application in an entrusted environment.

Acknowledgments

This work was presented in part at the IEEE International Conference on Communications (ICC), 2009. This work was can be done in part of in our institution and support of all staff members.

References

[1] Azab.A.M,Ning.P,Wang.Z,Jiang.X,Zhang.X and Skalsky.N.C, "Hypersentry: Enabling Stealthy In-Context Measurement of Hypervisor Integrity," Proc. 17th ACM Conf. Computer and Comm. Security (CCS), pp. 38-49, 2010.
[2] Huang.R,Deng.D.Y and Suh.G.E, "Orthrus: Efficient SoftwareIntegrity Protection on Multi-Cores," Proc. Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 371-384, 2010.
[3] Kirkpatrick.M.S,Ghinita.G and Bertino.E,"Resilient Authenticated Execution of Critical Applications in Untrusted Environments,"IEEETransation on dependable and secure computing,july/august 2012.
[4] Litty.L,Lagar-Cavilla.H.A and Lie.D,"Hypervisor Support for Identifying Covertly Executing Binaries," Proc.17th USENIX Conf. SecuritySymp., pp.243-258,2008.
[5] Piromsopa.K and Enbody.R.J,"SecureBit:Transparent,Hardware Buffer Overflow Protection," IEEE Trans.Dependable and Secure Computing, vol.3,no.4,pp.365-376,Oct.-Dec.2006.
[6] Rhee.J,Riley.R,Xu.D and Jiang.X,s"Defeating Dynamic Data Kernel RootkitAttacks via VMM-Based Guest-Transparent Monitoring," Proc.Fifth Int'l Conf.Availability,Reliability and Security(ARES), 2009.
[7] Riley.R,Jiang.X and Xu.D, "Guest-Transparent Prevention of Kernel Rootkits with VMM-Based Memory Shadowing," Proc.11th Int'l Symp. Recent Advances in Intrusion Detection (RAID),pp. 1-20, 2008.
[8] Rinard.M,Cadar.C,Dumitran.D,Roy.D.M,Leu.T and Beebee.W.S,"Enhancing Server Availability and Security Through Failure-Oblivious Computing," Proc. Sixth Conf. Symp. Operating Systems Design and Implementation (OSDI), pp. 21-21, 2004.
[9] Salamat.B, Gal.A, Jackson.T, Manivannan.K, Wagner.G, and Franz.M, "Multi-Variant Program Execution: Using Multi-Core Systems to Defuse Buffer-Overflow Vulnerabilities," Proc. Int'l Conf. Complex, Intelligent and Software Intensive Systems, pp. 843-848, 2008.
[10] Salamat.B,Jackson.T,Gal.A and Franz.M, "Orchestra: Intrusion Detection Using Parallel Execution and Monitoring of Program Variants in User-Space," Proc. Fourth ACM European Conf.Computer Systems (Eurosys), pp. 33-46, 2009.
[11] Sidiroglou.S,Laadan.O,Perez.C,Viennot.N,Nieh.J and Keromytis.A.D,"ASSURE: Automatic Software Self-Healing Using Rescue Points," Proc.14th Int'l Conf.Architectural Support for Programming Languages and Operating Systems(ASPLOS), pp.37-48,2009.
[12] Trachsel.Oand Gross.T.R, "Variant-Based Competitive Parallel Execution of Sequential Programs," Proc. Seventh ACM Int'l Conf. Computing Frontiers, pp. 197-206, 2010.

AUTHORS PROFILE

S.Valarmathi received the B.Tech Degree Information Technology and now she is an M.E student in the Department of Computer Science & Engineering, Srinivasan Engineering College – Dhanalakshmi Srinivasan Group of Institutions, Perambalur, TN, India.
Her research interest includes Network Security and Mobile Computing.

S.Sathishkumar is working as Assistant Professor/IT, Srinivasan Engineering College – Dhanalakshmi Srinivasan Group of Institutions, Perambalur, TN, India.
His research interest includes pervasive computing, Wireless Networks and Image Processing.