A Strong Execution Environment for a Critical Application Even in the Presence of Corrupted Environment

Valarmathi.S, Sathishkumar.S, Venkatesan.D

Abstract-A strong execution environment is created for a critical application even in the presence of entrusted environment. Generally in an entrusted environment if any application is going to be executed mean suddenly it terminates an application or data loss is occurred. To overcome this drawback some of the existing technique was developed such as variant based and replication technique and it is not much effective because overhead problem is occurred. A new technique called Virtual Machine is going to be developed. In this technique VM is used as a secondary storage to store all the details. Two modes are created one is user mode and another one is kernel mode. In user mode user can view the file name only they do not have the rights to view the file content. In kernel mode only the user have the rights to view the content of the file. Virtual memory monitors and displays the user details that are when the user comes. This technique is mainly used for critical applications such as colleges, bank and hospitals and so on.

Index terms-Memory corruption, Operating System, Security, Virtual machine.

I. INTRODUCTION

Network security is used to prevent modification and unauthorized access of data from intruders. The main aim of attackers is to capture the content of the file or to modify the file content. Attackers try to corrupt the operating system by injecting any code in the system. If any application is going to be executed in that environment mean data loss or modification in data content is occurred. So our aim is to avoid such data loss and modification for that error recovery mechanism is used to get back the original content.

Suppose intruders inject any code in the content of the file mean it is identified by using the memory usage. The original size of the file is stored in the virtual machine. If the intruders inject any code mean definitely the size of the memory is increased from that we can identify that corruption is present. By using error recovery mechanism we can recover from that error. Rollback mechanism is used if any application is going to be executed mean there are several checkpoints are present once the application reach any checkpoint mean it is saved in that point, before reaching next checkpoint if any corruption take place mean it is recovered from the previous checkpoint. In this technique the application is recovered from last context switch. So we are proposing a new technique that provides a strong execution environment for an application even in the presence of entrusted environment.

Manuscript received on April, 2013.

Ms.S.Valarmathi, M.E. (Computer science and Enginerring),Srinivasan Enginerring college,Perambalur,India.
Mr.S.Sathishkumar, Assistant professor (Information Technology),Srinivasan Enginerring college, Perambalur, India.
Mr.D.Venkatesan, M.E. (Computer science and Enginerring),Srinivasan Enginerring college,Perambalur,India.

II. PROBLEM STATEMENT

The main aim of an attacker is to inject any code in the operating system to corrupt the OS. If an application is going to be executed in an entrusted environment mean modification can take place. The attackers try to capture the data to avoid that the application is encoded and allowed to execute in a corrupted environment. For encoding an application unique key is used for each file so definitely overhead problem is occurred. To avoid that problem single key is used to encode all the files. But the intruders aim is to modify the file content for that they try to inject code in the encoded format. So definitely modification can take place our aim is recover from that modification and to provide strong execution environment for an application.

III. RELATED WORK

A. *Failure Oblivious Computing*

The concept of failure-oblivious computing describes the program discards illegal writes manufactures values for illegal reads and continues to execute through memory errors without address space or data structure corruption. Instead of terminating or throwing an exception the generated code simply discards invalid writes and manufactures values to return for invalid reads enabling the server to continue its normal execution path [8]. The results show that this technique make these servers invulnerable to known security attacks that exploit memory errors and enable the servers to continue to operate successfully to service legitimate requests and satisfy the needs of the users even after attacks trigger their memory errors. Failure-oblivious computation enhances availability, resilience and security by continuing to execute through memory errors while ensuring that such errors do not corrupt the address space or data structures of the computation. In many cases failure-oblivious computing can automatically convert unanticipated and dangerous inputs or data into anticipated error cases that the program is designed to handle correctly. Failure-oblivious computing enables a program to continue execution despite memory error.

B. *Reactive Approach*

A reactive approach for handling a wide variety of software failures ranging from remotely exploitable vulnerabilities to more mundane bugs that cause abnormal program termination. This system monitors an application during its execution using a variety of external software probes trying to localize observed faults. When a fault is detected it recovers program execution to a safe control [11].

Once a fault has been detected it restore control to a safe state by forcing the function containing the fault to return an error value and rolling back any memory modification the emulated code has made during its execution. That most applications are written well enough to catch the majority of errors but fail to consider some boundary conditions that allow the fault to manifest itself. This approach is an exploration into a reactive system that allows quick automated reaction to software failures thereby increasing service availability in the presence of general software bugs. That this approach can be used to catch a variety of software failures not just malicious attacks. Reactive approach handles wide variety of software failures.

IV. FUNCTIONAL DESIGN AND ANALYSIS

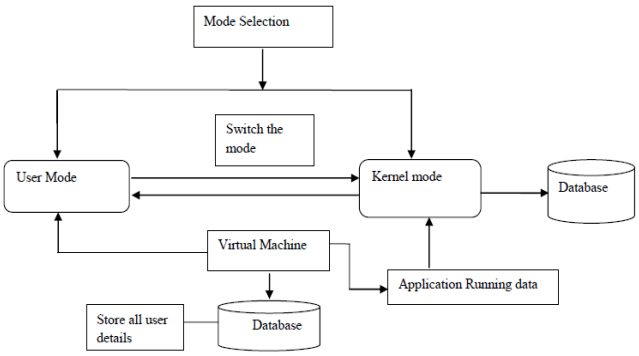

Fig.1 System Architecture

Four set of functional steps needed as follows

Project is going to be divided into the following modules.

1) Mode creation.
2) Data Upload to Virtual Machine.
3) Single authentication key generation.
4) Data access.
5) Error Recovery.

4.1 Mode Creation

Two modes are created- user mode and kernel mode.

User Mode- login to a user mode .In this mode user can only view what are the files present in the system. It does not allow the user to view the content of the file. VMM accepts the user details. It calculates the amount of memory used for the file.

Kernel Mode- login to a kernel mode. In this mode only, the user can view the content of the file.VMM accepts the user details. It also calculates the amount of memory used for the file.

4.2 Upload Data

Here virtual machine starts to work. Virtual machine accepts user mode and kernel mode details. It calculates the CPL (Current Privilege Level) for each mode. It provides a CPL value as three for user mode. It provides a CPL value as zero for kernel mode. The data is uploaded into virtual machine. Virtual machine display the uploaded data details.VMM help to know when the user was access the file and who access the file when it happens. If suppose in a kernel mode login if any intruders access the file mean it display which file and when it is happen .It will help to know intruders are come or not.

4.3 Single authentication key generation

In kernel mode login also the person who having the key can only view the content of the file otherwise it is displayed in encoded format. If any intruders come in kernel mode login mean they can view the file only in encoded format. So they could not view the content of the file. Here I am using single key to encode all the files.

4.4 Data Access

If intruders try to access the file mean it is displayed in encoded format so they cannot view the content but they try to modify the file content for that intruders inject any extra code in the encoded format or the modify the encoded format without inject any extra code. The user who having the key has the rights to view the content of the file and perform any change mean that person have the rights.

4.5 Error Recovery

If intruders perform any modification in the encode file mean it can be identified by using Virtual machine. The authorized user can know when it is happen by using virtual machine because in VM it display the time where the user is accessed if suppose I am a user mean I know when I am doing that change. If intruders inject any code in the encoded format mean memory size is increased from that i can identify that intruders inject code so I can recover from that error and get the original content. Suppose instead of adding any code in the encoded format they try to modify the encoded format without inject any code mean how can i identify that modification can take place and where. If any such situation is happen mean I can recover from that error also. If these things are happen mean we can recover from that modification and get the original content back.

V. EXPEIMENTAL RESULT

This experiment shows that our technique is mainly used to recover the critical application from corruption. Here virtual machine is used as a secondary storage to store all the details about the user whether the user is from user mode or from kernel mode. It monitors and displays the accessing details of the user that is where the user comes what they corrupt by using VM we can identify the corrupted file and we can recover the file from corruption by using error recovery code. In this technique for encoding the file only single key is used so overhead is reduced.

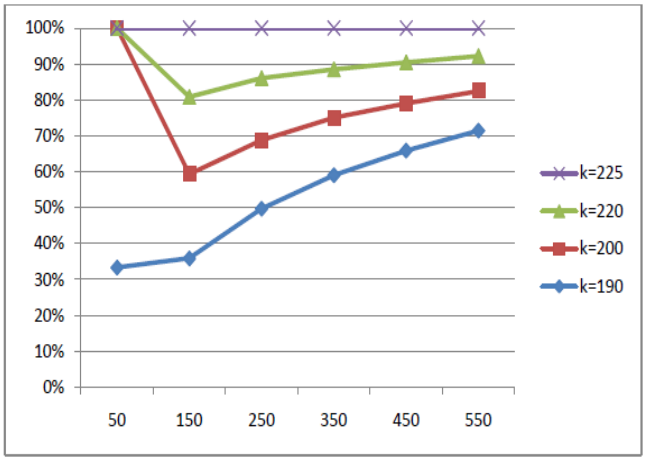

Fig 2.Rate of successful data recovery

VI. CONCLUSION

It is conclude that we are providing the strong execution environment for a critical application even in the presence of entrusted environment. If the intruders inject any code in the encoded file mean it can be recovered by using error recovery mechanism. Here the application is recovered from the last context switch that is before corruption. This technique also reduce the overhead by using single key for encoding the file content instead of using unique key for each file in the system and also it protects the application content from intruders.

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 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,LocastoM.E, Boyd.S.W and KeromytisA.D,"Building A Reactive Immune System for Software Services,"Proc. USENIX Ann. Technical Conf., pp. 149-161, 2005.
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.

D.Venkatesan received the B.Tech Degree Information Technology and now he is an M.E student in the Department of Computer Science & Engineering, Srinivasan Engineering College – Dhanalakshmi Srinivasan Group of Institutions, Perambalur, TN, India. His research interest includes Wireless Networks and Image Processing.