ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

Enhanced Kernel Based Process Validation for High Level System Assurance

Prof. Sonawane V.D.[1], Amit Mishra[2], Chattar Vishal[3], Jawale Vinay[4]

Assistant Professor, Dept. of Computer Engineering, AAEMF's COE & MS, Koregoan Bhima, SPPU, Pune, India[1,2,3]

B.E. Students, Dept. of Computer Engineering, AAEMF's COE & MS, Koregoan Bhima, SPPU, Pune, India[4]

ABSTRACT: The existing operating system doesn't have kernels level security and they are protected from malicious activity through either by using Mandatory Access Control(MAC) or Firewall and antivirus. The existing systems use the authorization mechanisms. But this is not sucient for achieving system assurance. The operating system Kernel is going to perform process level validation ,where user level applications process proves its identity to kernel. The present process validation is performed using process names or an executable path used by OS to identify a process,which makes it unreliable. It results in the form of malware that may impersonate the other processes thus violating the system assurance . Nowadays we heavily rely on mission critical high computing machine to complete our day to day online services and facilities. Therefore, these mission critical computing machines are very critical and organization don't want the servers to be down because of virus attacks and hacking of those systems. High assurance systems are now in demand and everybody is looking for new security techniques on top of general Antivirus systems available in the market. These days hackers and viruses on internet are smart enough,that's why the mission critical systems having antivirus or firewalls are not sufficient. They are looking for new security techniques at process execution level to protect them against any malware attacks and system downtime. The system must do validation process before execution and that should be based with some trusted internal mechanism.

KEYWORDS: Operating system security,Kernel,Process validation

I. INTRODUCTION

This paper points out the need in modern operating systems kernels for a process authentication mechanism, where a process of a user-level applications shows its identity to the kernel. Process authentication and process identification are two different works of OS. The identification describes a principal; PIDs or process names are identifiers for processes in an OS environment. The information like process names or executable paths that is conventionally used by OS to identify a process is unreliable. Therefore, the malware may impersonate other processes, making system prone to hackers. We propose a secure process authentication framework prototype in which user-level applications are required to prove its identity during runtime to get authenticated from the kernel. For process authentication, we a system monitoring framework is developed to prevent unauthorized use or access of system resources. It verifies the processes identity before it completes requested system calls. We will implement and evaluate a prototype of our monitoring architecture in Linux. The prototype gives reasonably low overhead, makes it feasible to cryptographically authenticating applications and their processes in the operating system.

II. RELATED WORK

It is proposed in [1] the system of Identifying Native Applications with High Assurance. The implemented system gives the address for identification problem by proposing a secure application identifier model in which user level applications are must present identification proofs during run time to get authenticated by kernel.

[2] developed a strong ,flexible mandatory access control(MAC) architecture used Type Enforcement. It is called the Flask,was developed for the Mach and Fluke research operating systems. The NSA has added the Flask architecture into the Linux operating system for developers and users for deploying the technology.

[3] have proposed Linux Security Modules (LSM). The Linux is prone to malwares, as seeing past years by software security vulnerabilities. The Linux Security Modules (LSM) project provides the Linux kernel a framework for access control's enables loading enhanced security policies as kernel modules.

III.PROPOSED SYSTEM

We will develop prototype system that will identify malicious applications before installation and untrusted or malicious process before execution. A system is developed that will give secure computing for mission critical systems. It will provide additional security by doing process level validation.

A. Validation Technique

1.Application classification while installation of new application to determine whether it is trusted application or malicious.
2.For trusted application process generate secrete credentials from Kernel.
3.Form Code Capsule of secrete credentials along with Process Name.
4.Add the Code Capsule into Credential List.
5.When any new process initiate for execution then Process Authenticator intercepts system call requests and asks process secrete credentials.
6.In Parallel Process Authenticator gets stored credentials with system from Credentials List with the help of Register.
7.Process Authenticator pass on credentials received from Process to Secrete Verifier for validation of process.
8.Secrete Verifier takes credential from process and saved credentials in credentials list.
9.If both Secrete Credentials are same then system calls that process as validated process and allowed for execution of process. Otherwise process is not allowed for execution.
10.Validated process names are maintained in the Status List as trusted process.

B. Process Validation Algorithm

Let P represent Process, A represent Process Authenticator module,
Let p.pid represent process ID of the process.
Let h is credential stored with credential stored with credential list L.
Let H is recomputed credentials from application process.
Let T is status list and Cs is code capsule contains encrypted secrete credentials.

Step 1: P sends validation request (p.name) to A. A will initiate process validation.
Step 2: A does following action:
2.1. A queries to Registrar as query p.cred, Registrar checks on credential list L. If p.cred is equal to NULL i.e. application does not have registered credentials.
2.2. Generates a unique random number from kernel and formed Secrete credentials along with application name. Encrypts all credentials using encryption algorithm.
Step 3: Encrypted credentials is code capsule and stored into Credential List and along with application.
Step 4: While validation of process, A compares secrete credentials:
If (h = = H)
P, process validation successful.
Else,
P, process validation is unsuccessful.
(Validation is unsuccessful since stored credentials in L and secrete credentials with process are not matched)
Step 5: A checks to see if p.pid belongs to T (Status list). If yes P, reports as suspicious.
Step 6: When P terminates, A is notified to and delete p.pid from the status list T.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

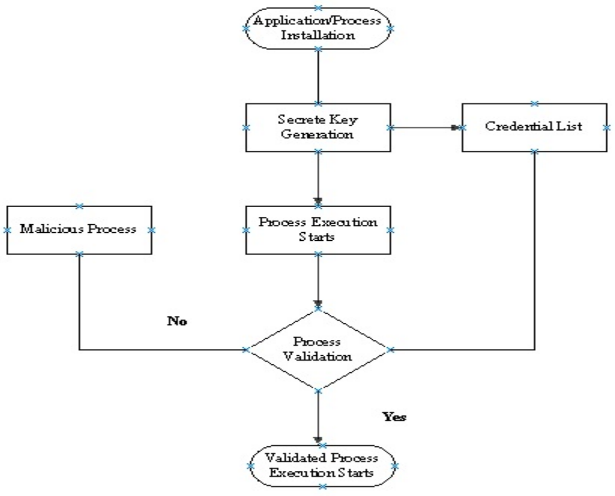

Figure 1. Workflow Diagram of Process Validation

IV. SYSTEM ARCHITECTURE

There are four main parts of the process validation as follows:Figure 1. Workflow Diagram of Process Validation
A) Application Classification
B) Secrete Credential Generation
C) Process Validation
D) Runtime Monitoring

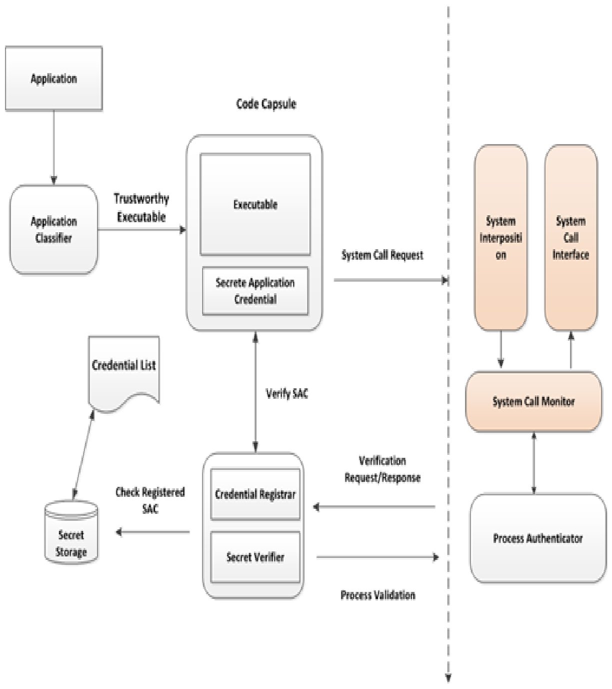

Figure 2. Process Validation Architecture

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

A. **Application Classification**

This operation deals with the initial level of application or process validation before installation on Windows operating system. The Classifier is the module takes care of application classification. When user tried to install any new application on Windows operating system, then classifier first check the whether application is trusted or not. This is initial assessment done by Classifier based on the some specific criteria's of the executable. Till the Classifier checks the dependability aspect of application, installation is paused and once Classifier gives result as trusted application then installation proceeds.

B. **Secrete Credential Generation**

This operation deals in two different scenarios. In first scenario once Classifier module any application as trusted application based on the initial assessment then during installation of application registrar creates secrete credentials for the respective application process. In second scenario for the already installed application before process validation system deployment, when process tries to execute then it first checks the credential list and if in credential list code capsule is not present for process , then new secrete credentials will be issued for respective process. The secrete credentials are a unique random number generated by the kernel. Once random number gets from kernel a code capsule formed with secrete credentials.

C. **Process Validation module**

This is main operation in process validation and deals with validation of process before eIt is proposed in [1] the system of Identifying Native Applications with High Assurance.The implemented system gives the address for identification problem by proposing a secure application identifier model inwhich user level applications are must present identification proofs during run time to get authenticated by kernel.xecution or making any system call. When any new process tries to execute Authenticator ask secrete credentials to process for validation. Once Authenticator ask for secrete credentials then process shows secrete credentials available with process, other hand Authenticator make request to Register to get secrete credentials to get credentials for respective process. All the secrete credentials in the form of code capsules are maintained in the Status list Register. gives list of available credentials for respective process to Secrete Verifier to validate secrete credentials provided by the process and available with the system. If both secrete credentials are matched then system conclude as trustworthy process for the execution. Once process is validated then process name is listed in Status list where all the validated process list is maintained.

D. **Runtime Monitoring**

This operation deals with runtime monitoring of the process validation status, this operation takes reference as Status List where all successful validated list of process is mentioned. In every new process execution validation system first check that respective process is validated successfully in the past or not. If respective process name is present into Status List then process validation is skip for the respective process and directly allowed for the execution.This operating runtime monitor what all process has been validated and maintain list of validated process in the form of Status List.

VI.CONCLUSION

The proposed system will work to provide enhanced level process validation before execution as compared to existing systems. There are two levels of security provide for mission critical system. The first level of security checks application trustworthiness before installation and avoid of any malicious program to enter into system. While the second level security provided through secure computing, process validation does for every process before execution and making any system call to consume any systemresources, this gives confidence to user towards only valid process will use of system resources and high level system assurance for mission critical systems. The implemented system will add value to exiting security measures with secure computing and since system will run in background minimal user intervention is needed. Proposed solution argues and demonstrates the kernel must where the identity of a process can be proved. Proposed system gives us guarantee for high assurance system and work on process level validation hence execution of any untrusted code can be prevented.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 10, October 2015

REFERENCES

[1] H.M.J. Almohri, D. Yao, and D. Kafura, "Identifying Native Applications with High Assurance," Proc. ACM Conf. Data and Application Security and Privacy (CODASPY '12)," Feb. 2012.
[2] P. Loscocco and S. Smalley, "Integrating Flexible Support for Security Policies into the Linux Operating System," Proc. USENIX Ann. Technical Conf., 2001.
[3] Z.M.H. Chen and N. Li, "Analyzing and Comparing the Protection Quality of Security Enhanced Operating Systems," Proc. 16th Ann. Network and Distributed System Security Symp. 2009.
[4] C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman, "Linux Security Module Framework," Proc. 11th Ottawa Linux Symp., 2002.
[5] K. Xu, H. Xiong, D. Stefan, C. Wu, and D. Yao, "Data-Provenance Veri_cation for Secure Hosts," IEEE Trans. Dependable and Secure Computing, vol. 9, no. 2, pp. 173-183,Mar./Apr. 2012.
[6] W. Dai, T.P. Parker, H. Jin, and S. Xu, "Enhancing Data Trustworthiness via Assured Digital Signing," IEEE Trans. Dependable and Secure Computing, vol. 9, no. 6, pp. 838-851, Nov./Dec. 2012.
[7] G. Xu, C. Borcea, and L. Iftode, "Satem: Trusted Service Code Execution across Transactions," Proc. IEEE 25th Symp. Reliable Distributed Systems (SRDS '06), pp. 321-336, 2006.
[8] A.M. Fiskiran and R.B. Lee, "Runtime Execution Monitoring (REM) to Detect andPrevent Malicious Code Execution," Proc. IEEE Int'l Conf. Computer Design: VLSI inComputers and Processors (ICCD '04), pp. 452-457, 2004.
[9] T. Jaeger and R. Sandhu, Operating System Security. Morgan & Claypool, 2008. [11] K.Xu, P. Butler, S. Saha, and D. Yao, "DNS for Massive-Scale Command and Control," IEEE Trans. Dependable and Secure Computing, vol. 10, no. 3, pp. 143-153, May/June. 2013.
[10] X. Shu and D. Yao, "Data-Leak Detection as a Service,â€. Proc.Eighth Int'l Conf.Security and Privacy in Communication Networks (SECURECOMM '12), Sept. 2012.