ISSN 2320-6802
ICIRET-2014

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY

WINGS TO YOUR THOUGHTS.....

PROCESS AUTHENTICATION USING PROC FILE SYSTEM

R. Bhavani
Department of Computer Science and Engineering
Rajalakshmi Engineering College
Chennai- 600128
bhavaniramamurthi@gmail.com

Abstract: *This paper states the need for process authentication in modern operating system. PID or process names are identifier of a process in Operating System. In general, the process name and installation path is used for authenticating a process which is not reliable. We propose a lightweight secure process authentication mechanism in which the process identity is encrypted and stored in kernel space when application is run for first time, and is verified with the stored value in credential registrar for authenticity. The process authentication is done before the requested system call gets completed. Our approach results in low overhead and proves that it is feasible approach for process authentication in operating system.*
Keywords: *Operating system; process authentication; process encryption; LUKS.*

1. INTRODUCTION

In general certain level of security is provided to the application by the modern operating systems. Process identifiers are mainly process id and process names. We assume that kernel does not contain any malicious code and is considered to be trusted. Many methods are being established for application authentication given by MAC in which the administrator manages the access control. It was used mainly in the system where confidentiality was considered as the main term. Some of the existing MAC systems are SELinux, grsecurity and AppArmor which overcomes the drawbacks of traditional MAC system. Commonly installation path are used by existing MAC in which the access rights are given by the user which is weaker and gives way for the malware to invade. Public key cryptography can also be applied which uses keys to verify. These traditional methods can be overcome by creating an application framework which verifies the application before the system call request ends. Session II describes about the literature survey.

2. LITERATURE REVIEW

L. Lu, V. Yegneswaran, P.Porras and W. Lee [1] proposed that web based malware infections are the major reason for the delivery of malicious software into the internet. To avoid this drawback a newly designed kernel extension based browser is created namely BLADE (Block All Drive by Download Exploits). This paper states that a mere connection to a web server can result in the installation of malware on the client machine. These infections are forced by the browser to download them and execute the malicious application. These malware are used to perform Denial of Service attack and botnet activity etc. BLADE does not perform any additional changes to browser instead it collects information about all the User Interface (UI) which are present in the browser. The browser separates the received data into two different files like supported file type and unsupported file type. BLADE intercept and impose by preventing the execution of the content which is not been consent to any user to browser interaction. BLADE introduces three OS-level capabilities like user interaction, consent correlation, disk I/O redirection. BLADE is proposed on child processes like plug-ins. The drive by exploits have 3 phases such as shell code injection phase, shell code execution phase, covert binary instant phase. The supervisor is considered to be the coordinator to carry out the task of BLADE, it does the internal communication too. Once when a notification is given to the downloaded, the status change is insisted to all of them.

S. Forrest, S. Hofmeyr, and A. Somayaji [2] proposed how system call monitoring was used in anomaly intrusion detection mechanism. Immune system are used to detect the foreign particles and misbehavior of the internal particles. This paper brings out the similarity between computer security and problem of protecting the human body from an threats either externally or internally. IDs are used to detect or identify a threat model. A distinguish between a

normal and abnormal system call is brought out. Analomy detection which relies on system call is able to detect buffer overflow, Trojan horses, and configuration errors. The different types of principles are generic mechanism, autonomy, Adaptable, graduated response, diversities. One of the most important problem to be considered is the external attack in which server programs run with the privileges in mail servers, remote login servers they maintain a gateways for the remote entry into the system, and hence it provides a vital role to protect against attacks. To develop a normal profile for the analomy detection two methods are followed like: synthetic normal profile by applying it in all normal usages. Collecting real normal profile is done for online usages.

M. Rajagopalan, M. A. Hiltunen, T.Jim, and R. D. Schilichting [3] proposed that system call monitoring based on authenticating system calls. The authenticated system call information is used by the verify the kernel to verify the system call. The authenticated system calls reads the application binary and uses the static analysis to generate the policies, and it rewrites the binary with the authenticated calls. System call monitoring techniques are used to detect the malicious applications. A SCM (System call Monitoring) is created in which the system call is monitored during runtime, which decides about the either allowing or blocking the application. Authenticated System Calls is used as the key factor. MAC is needed to support confidentiality and integrity. The limitations of MAC is over come by Flask. MAC uses cryptographic key which is done at runtime by the kernel. System call policy is satisfied when the call is executed. System call policies are used to trace the system call graph, system call policies are also used to mean to capture the legitimacy of the system call. The checking of the system call is done either at the user space or at the kernel space. The system call monitoring technique specified here does the various steps like generating the policies, replacement of the system calls with the authenticated ones of the same and runtime checking is done by the kernel. The installation process is done by the first two procedures. Authenticated system calls are used as the key factor.

G. Xu, C. Borcea, and L. Iftode [4] proposed about Satem, a Service-aware trusted execution monitor ensuring the trustworthiness of the executable code across client-service transactions. Satem architecture has an execution monitor in specified with Trusted Platform Module (TPM). For a transaction to be successful the user demanded service must be proven before transaction starts. The existing method lags in measuring and protecting the integrity of the service code at the runtime. Then it creates a commitment, the commitment is ensured with the third-party trusted authority. The client verifies the commitment with the third party and make sure that it is not being compromised. The operating system kernel of the service provider is registered with the Trusted Platform Module (TPM). The main drawback of this method results in false positives. BIND emerges to solve this problem due to the high complexity the code is applied to a certain part of the code instead of the entire part of service code. Before the transaction starts the client requests the trusted execution monitor. The main motivation of this work is to prevent from 3 real-life threats namely Service Spoofing, Service Tampering, and Post-Request Attack.

P. Loscocco and S. Smalley [5] proposed that DAC (Discretionary Access Control) is mainly based on user identity and ownership. It ignores security relevant information such as user role, trustworthiness of the program and sensitivity, integrity of data. NSA created Security- Enhanced Linux or SELinux integrating this enhanced architecture into the existing Linux operating system. SELinux supports the following restrict access to the classified data, minimizes the damaged caused by the virus and malicious code. DAC does not provide any protection against the malicious software. DAC mechanism is used by malicious or flawed application can easily cause failure in system security. DAC is inadequate for strong security. DAC supports trusted administrators and completely untrusted ordinary users. MAC is added to the existing vulnerability, it is based on labels. The limitations of traditional MAC is addressed by the National Security Agency (NSA), with the help of Secure Computing Corporation (SCC) developed two mach based prototypes, DTMach and DTOS, they developed strong, flexible security architecture.

H. Zang, W. Banick, D. Yao, and N. Ramakishnan [6] proposed a framework to analyze user actions and the network related events, this helps to identify the anomalous events caused by the malicious program. To provide malware protection to the existing system we propose a light weight cryptographic mechanism to provide message authentication. Parasitic malware uses same process ID as that of the host program. The unwanted traffic is noted as these can leak the user related information. To test the user based options CR-Miner is created to test the security, accuracy and also the efficiency of the user activities. The main goal of the CR-Miner is to identify the dependencies in the network traffic. A semantic based approach is created to detect the anomaly traffic on hosts. They

observe the dependencies between the user activities. First the CR-Miner framework analyzes the model and a tree based structure is maintained. The false positive rate of the CR-Miner is also calculated.

H.M.J. Almohri, D. Yao and D. Kafura [7] proposed that process identification at runtime is authenticated to the kernel using the secret key. This secret key is registered with the kernel at the time of installation, which is authenticated in unique way. Existing MAC system is combined with the system call monitoring. There are two main concepts namely application identification and application monitoring. Process names are dynamic and can be changed by the user or an attacker at specific time. Monitoring of the application is done to prevent the unauthenticated application to access the resources. The secret key is created for the legitimate application, tokens are created for secure authentication of the application. The tokens are bind with the appropriate application with their access rights. The main advantage of this paper is there is no performance penalty.

K. Xu, H. Xiong, D. Stefan, C. Wu and D. Yao [8] proposed that their main goal was to improve trustworthiness of the host resources. System data integrity approach is created at the system level. Two applications are created such as keystroke integrity verification and malicious traffic detection. Data-provenance integrity states that source from which a piece of data is generated can be verified. The outbound network packets states that the packets are generated by user-level application and it cannot be injected in the middle of the network stack, they can be prevented by providing a firewall at the transport layer without being bypassed by malware. Keystrokes are used in external keyboard devices in client-server architecture. It includes authentication of two important data types user inputs and network flow. Signatures scheme is involved in the malware detection, the signer and verifier are kernel module.

Z.M. Hong Chen, N. Li [8] proposed that VulSAN (Vulnerable Surface ANalyser) is used for measurement of protection quality for analyzing and comparing protection of MAC system in Linux, the tool results in the creation of the graph. VulSAN uses various tools such as Attack Path Analyzer, Fact Collector, Host Attack Graph Generator. Fact collector collects security policies, system state details and the details about the running processes. Host attack uses the scenario and generates the host attack graph. SELinux policy defines processes of which domain can access objects of different operations. AppArmor is an access control system which describes about the access permission it maintains a profile list. The profile list consists of the file access details. If there is no profile it says that it is not confined by default. The file permission is defined in the profile list.

3. APPLICATION AND AUTHENTICATION

The system architecture has three major functions described in it such as trusted key registrar; the applications are classified as the legitimate and the trusted application to identify the intrusion of the malware in the application and to secure the use of system resources. Authenticator and Service access monitor. The architecture is explained below.

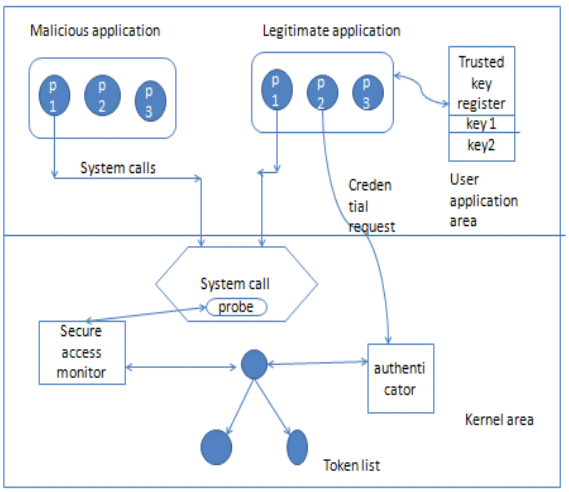

Fig. 1 Process Authentication

a. Service Access Monitor (SAM)

Service access monitor is responsible for verifying the process authentication at runtime and enforces application-level access rights. Since the tokens are maintained by the authenticator, SAM realizes its task by coordinating with the authenticator through a shared data structure. SAM enforces application level access rights based on user specified application policy. SAM maintains two lists namely credential list and status list. All the valid credentials are maintained in the credential list.

b. Trusted Key registrar

Trusted key registrar is a kernel helper responsible for installing the key for the application and registering the application with the kernel. The application interacts with the trusted key registrar to receive a secret key. The trusted key registrar stores the same key and registers it for corresponding application with a secure storage to be used for the authentication of the process at runtime.

c. *Proc file system*

Proc file provides information about the processes and run time system information. /proc is a virtual file system. It is mapped to a mount point named /proc at boot time. The proc file acts as a intermediate between the internal data structure in the kernel.

d. *Credentials Generation*

Secret credentials are created for the trusted applications. Application Authentication can be done at any time either at the installation or at the run time. Generated copies of the credentials are maintained by the application and the registrar. On the removal of the credential the application is no longer valid. Random numbers can be generated to make the guess harder; the major drawback of the credential storage is maintaining the credential list in secret. The credentials are bind with the executable file from which the authenticator authorizes or verifies the application. The encrypted application is accessed only using the given passphrase at the time of encryption

e. *Process Authenticator*

Authenticator is mainly responsible for authenticating a process when it loads first. The authenticator generates identity tokens based on a token generation protocol. Credentials are referred to cryptographic keys and passwords. Credentials are the control access to the information and other resources. These credentials are being verified by the authenticator.

f. *LUKS Encryption*

Linux Unified Key Setup is used for encrypting the required partition which provides high level of security with the low level of attacks and supports multiple keys. LUKS includes the Cryptsetup. The Encrypted part is made available with the Passphrase which protects the illegal use of the resources available

4. PROCESS AUTHENTICATION

There are three major operations in authenticating a process- Credential Generation, Authenticating Process and Runtime Monitoring. The process is accredited with credentials and the credential list is accessed by the kernel. The process authentication must satisfy the following un forge ability, Anti-replay, Uniqueness etc. The credential is registered for the process which is considered to be the legitimate application. The credential is maintained secret; the secrecy of the credential is done by the code capsule. The code capsule is a piece of code which is not read or write accessible to user in any way while it is accessible only to the kernel, this is being attached with the executable code. The major role of the code capsule is to combine the credential with the corresponding executable file. The process is authenticated using AES 256 bit. AES 256 bit has 14 rounds of encryption. It resists quantum computer attack. There are 4 major rounds in general they are Sub bytes, Shift rows, Mix columns, Add Round key. AES 256 bit encryption provides a high level of security; it can be efficiently implemented in hardware and also in software and also has a less memory utilization.

The credential list maintains the name of the application and their corresponding credential. There exist a authenticator module which maintains the status list S and the credential list with the above mentioned features. The authenticator requests the registrar with the name of the application which in turn replies the request with the credential saved in the list T. The credential list is maintained with the encrypted value for the process ID.

The authentication protocol is between the authenticator and a process at the time of the process creation. The procedure at the time of creation is explained as follows.

1. The process s sends the authentication request to the application to claim that it is the exact process.
2. The authenticator A request the registrar for the process with s.name. It returns the value if it has for the corresponding process s if there is no match found it returns null.
3. When there is no credential saved it alerts a sign for suspicious application.
4. The authenticator A creates a random once and sends it to the process s.
5. The process s secret credential is obtained from the code capsule.
6. If the authenticator time t exceeds the threshold value the request for the authentication of the process is terminated is indicated to A.

5. IMPLEMENTATION AND RESULT

We have implemented credential registrar in c in Linux operating system. The process name, PID & file details are encrypted with AES 256 bit key and the cipher text generated by the LUKS encryption is saved in the credential registrar. Luks encryption method is performed to avoid illegal access of the resources. The saved details are accessed using the proc file system.

6. CONCLUSIONS AND FUTURE WORK

Our work on credential generation mechanism for authenticating a process in Linux operating system has been implemented successfully. Only legitimate processes are allowed to access the system resources and illegal access is notified to the administrator through mail. The process can be invoked through a front end framework. In future this can be implemented in the android mobile phone.

REFERENCES

[1] L. Lu, V. Yegneswaran, P. Porras, and W. Lee, "BLADE: an attack-agnostic approach for preventing drive-by malware infections," in Proceedings of 17th ACM conference on computer and communication Security, ser. CCS'10. New York, NY, USA: ACM,2010, pp. 440-450.

[2] S. Forrest, S. Hofmeyr, and A. Somayaji, "The evolution of system-call monitoring," in Proceedings of 2008 Annual Computer Security Applications Conference, ser ACSAC'08. Washington, DC, IEEE computer society, 2008, pp. 418-430.

[3] M. Rajagopalan, M. A. Hiltunen, T. Jim, and R. D. Schilichting, "System call monitoring using authenticated system calls," IEEE Transaction on Dependable and Secure Computing, vol. 3, pp. 216-229, July 2006.

[4] G. Xu, C. Borcea, and L. Iftode, "Satem: Trusted service code execution across transactions," in Proceedings of the 25th IEEE Transaction on Reliable Distributed Systems (SRDS). Washington, DC, USA: IEEE computer society, 2006 ,pp. 321 – 336.

[5] P. Loscocco and S.Smalley, "Integrating flexible support for security policies into the Linux operating system," in Proceedings of the 2001 USENIX Annual Technical Conference. Berkeley, CA: USENIX Association, 2001.

[6] H. Zang, W. Banick, D.Yao, and N. Ramakrishnan, "User intention-based traffic dependence analysis for anomaly detection," in Proceedings of Workshop on Semantics and Security (WSCS), May 2001.

[7] K. Xu, H. Xiong, D. Stefan, C. Wu, and D. Yao, "Data provenance verification for secure hosts," IEEE Transaction on Dependable and Secure Computing (TDSC), vol. 9 (6), pp. 838 – 851.

[8] Z.M. Hong Chen, Ninghui Li, "Analyzing and comparing the protection quality of security enhanced operating system," in Proceedings of the 16th Annual Network & Distributed System Security.