IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 2, April 2014.
www.ijiset.com

ISSN 2348 - 7968

A LIGHTWEIGHT SECURE PROCESS AUTHENTICATION MECHANISM USING ENCRYPTION TECHNIQUE

R. Bhavani
Department of Computer Science and Engineering
Rajalakshmi Engineering College
Chennai, India.

***Abstract* — This paper highlights the need for process authentication for user-level applications in modern operating system. In most systems, the process name and installation path is used for process authentication purpose which is not reliable. We propose a lightweight secure credential generation mechanism for process authentication in which user-level applications need to prove their identity to the kernel at runtime and it is stored in the encrypted space of the disk. The process identity is encrypted and stored in credential registrar when application is run for first time. Subsequent run of the application generates new credential and is verified with the stored value in credential registrar for authenticity. Our prototype evaluation results in low overhead and proves that it is feasible approach for process authentication in operating system.**

Keywords- Operating system; process authentication; credential registrar; Authentication framework

I. INTRODUCTION

Operating system normally provides a certain level of security to the applications. Process identifiers are mainly process id and process names. We assume that kernel does not contain any malicious code. Many types of authenticating an application are being provided by MAC in which the administrator manages the access control. It was used mainly in the system where confidentiality was considered as the main constraint. Some of the existing MAC systems are SELinux, grsecurity and AppArmor which overcomes the drawbacks of traditional MAC system. Commonly MAC uses the installation path for which the access rights are given 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.

II. RELATED WORKS

G. Xu, C. Borcea, and L. Iftode [1] 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. 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. 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 motivation of this work is to prevent from 3 real-life threats namely Service Spoofing, Service Tampering, and Post-Request Attack.

H.M.J. Almohri, D. Yao and D. Kafura [2] 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. The secret key is created for the legitimate application, tokens are created for secure authentication of the application. Monitoring of the application is done to prevent the unauthenticated application to access the resources. The tokens are bind with the appropriate application with their access rights. The main advantage of this paper is there is no performance penalty.

P. Loscocco and S. Smalley [3] 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. 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

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 2, April 2014.
www.ijiset.com
ISSN 2348 - 7968

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. 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.

Z.M. Hong Chen, Ninghui Li [4] 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.

H. Zang, W. Banick, D. Yao, and N. Ramakishnan [5] 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 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. 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.

K. Xu, H. Xiong, D. Stefan, C. Wu and D. Yao [6] 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.

III. SYSTEM ARCHITECTURE

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.

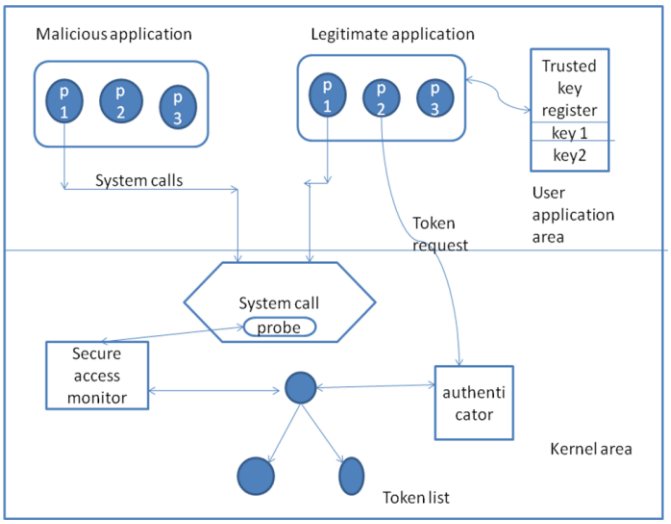

Fig. 1 Process Authentication

A. *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.

B. *Authenticator*

Authenticator is 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.

C. *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 list namely credential list and status list. All the valid credentials are maintained in the credential list.

IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 2, April 2014.
www.ijiset.com
ISSN 2348 - 7968

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. *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.

IV. Discussion

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 unforgeability, 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 192 bit. AES 192 bit has 12 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 192 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 L. 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 p 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 p.name. It returns the value if it has for the corresponding process p 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 p.
5. The process p 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.

IV. Implementation And Results

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

V. Conclusion

Our work on credential generation mechanism for process authentication 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. In future, this mechanism will be incorporated in android based handsets.

References

[1] G. Xu, C. Borcea, and L. Iftode, "Satem: Trusted service code execution across transactions," in Proceedings of 25th IEEE Symposium on Reliable Distributed Systems(SRDS). Washington, DC, USA: IEEE computer society, 2006 ,pp. 452-457.

[2] H.M.J. Almohri, D. Yao, and D. Kafura, "Identifying native applications with high assurance," in Proceedings of ACM Conference on Data and Application Security and Privacy (CODASPY), February 2012.

[3] 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.

[4] 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.

[5] 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.

[6] 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.