Provisioning encryption for system call to secure the communication between user application and kernel mode

Shobana L*, Muthumanickam K,
Department of Computer Science and Engineering, Arunai Engineering College, Tiruvannamalai,
***Corresponding author email: shobana.cse25@gmail.com**

ABSTRACT

Operating system is the essential part of the computer system. Windows is one of the operating system used by all most all of the computer system. In windows system, there is some limitation; it does not have kernel level security that leads to malicious activities to acquire the system services. The process of a user level application needs to prove its identity to the kernel before accessing the system services. And the information such as process name or process id is not enough to identify a process by an operating system. That's results; malware may make another process to acquire the system service. In proposing approach the process requesting the services through a system call needs to achieve a secure computing with the help of cryptographic algorithm called RSA algorithm. For that, a secret key will be issued to encrypt the process id and the key will be maintained in the system, the encrypted process id will be invoked at the kernel end and decrypts it for validation purpose. If it is a valid process, then it allows acquiring the service. However, the system avoids the kernel crashing.

KEY WORDS: operating system, kernel security, root kits, user application, RSA algorithm.

1. INTRODUCTION

An operating system is a most important component of system software and controls the computer hardware resources and provides familiar services for computer programs. The operating system is an important component of the system software in a computer system. The hardware must provide appropriate mechanisms to make sure the correct operation of the computer system and to prevent user programs from the proper operation of the system. An OS is the program that, after being initially loaded into the computer by a boot program, controls all the other programs in a computer. Windows is worldwide accepted user friendly operating system almost used by 90% of system users in the world. But root kits have been a major concern for windows users and it is difficult to protect the operating system. Because, the source code of the windows is not made publicly available. Also it does not have kernel level security that leads to malicious activities to acquire the system services. Operating systems have been a common target of attackers who try to break their protection mechanisms and modify the system according to their advantage. A program that allowed a non-authorized user to access the system and become a system administrator was called a root kits. Now a day the root kits were developed as software that hides the presence of attackers in the system and take overall control of the system. Kernel mode root kits present at the core of an operating system and thus they may find more difficult to detect and remove from the system. Root kits can occur through user application. For that, the process of a user level application needs to prove its identity to the kernel before accessing the system services. And the information such as process name or process id is not enough to identify a process by an operating system. That's results; malware may make other process to acquire the system service. The process requesting the services through a system call needs to achieve a secure computing with help of cryptographic algorithm that improve the integrity of the system which refers to the consistency and accuracy of data to ensure that unauthorized users are prevented from modifying data and maintains integrity.

Motivation and Objective: In windows system architecture, we identified some of the weakness such as there is no protection between the user mode and kernel mode, when the system service requests transfer from the user to kernel mode. The activity of viruses, worms, Trojans and other forms of malware permit an attacker to take control of a computer system. An attacker will gain administrative privilege by installing a root kits which is developed as software. The attacker uses these root kits to steal personal data such as bank account details, passwords and credit card numbers. But general purpose anti-virus software failed to detect such type of malware activities. API hooking mechanism cannot determine the system service requests is called from user mode or infected code. To design a system by using kernel mode API hooking to avoid a code injection attack as well as to avoid system crash is challenging. Authentication is capable of identifying interpreted programs running as stand- alone processes. Interpreted languages such as JavaScript, Adobe action script and word document macros are not comes under this type of security model. The detection of malicious code injected into authenticated process as opposed to running as stand-alone process is out of the scope of A2 attack model. Classification of trustworthiness of program is challenging. In order to avoid this problem, providing integrity has been established (i.e.) scrambling technique was used by the suspicious code.

The main objective of this system is to protect the system from rootkits attack. Process of a user level application needs to prove its identity to the kernel before accessing the system services. And the information such as process name or process id is not enough to identify a process by an operating system. That's results; malware may make other process to acquire the system service. So, the process id has to validate to check whether it is legitimate process or suspicious process with the help of RSA algorithm. For that, a secret key will be issued to encrypt the process id and the key will be maintain in the system, the encrypted process id will be invoked at the kernel end and decrypts it for validation purpose. If it is a valid process then it allows acquiring the service. However, the system avoids the kernel crashing.

Literature survey: Almohri (2014) proposed a lightweight secure application authentication framework in which user level application are need to be proofs their authentication during runtime. Also they developed a system call monitoring framework to protect the system resource from unauthorized access by the users. The security of A2 framework is depends on the confidentiality of the application credentials. Pradnya patil (2014), proposed a component that provide a confidential key for every created process and validated component. And both can be authenticate on the first time creation of the process. Two different key lists are maintained in the system for credentials and status lists. Proposed solution will create a secret key for newly created and already created processes. Expected user using this proposed solution will be authenticating only. Masadesh (2014) proposed a novel security technique which uses new encryption and decryption algorithm to achieve authenticated communication and enhanced data integrity along with PGP, sWIFI and HMAC systems. The proposed system is very complex for attackers to decode and it is applicable to client-server architecture. Jinpeng Wei, Francesco gadaleta (2013), proposed a Hello root kitty concept; it is an invariance enforcing framework which takes advantages of current virtualization technology to protect a guest operating system against root kits. By application the abstraction of invariance to ascertain maliciously adapted atom abstracts structures and restores them to their aboriginal accepted values. Effectively attention article operating arrangement from avant-garde basis kits. Miao Wang (2011), introduces a Built-in API for advance apprehension system. The atom akin API hidden in the NT architectonics arrangement is alleged built-in API. The bisect the captured sequences with accelerate window adjustment to authorize accustomed arrangement database. They prove that windows built-in APIs are allegedly accessible abstracts antecedent for host aberration apprehension arrangement beneath windows platform.

System Implementation

User Application: User mode is the unprivileged mode. Every user I/O request involves an exchange of information between user and kernel. But the kernel doesn't assure the application arrived from the user space. It has to validate all data and address passed from user mode. If there is a root kits present in the system, then the kernel cannot able to validate whether the process is legitimate or not. Ntdll.dll acts as an entrance to kernel mode. All user-mode API's are expected to reach this entry point for accessing the system services. Ntdll.dll passes the requests to the kernel driver then to the service dispatcher in kernel, known as KiSystem Service, where the requests are redirected to the suitable kernel service providers. By reviewing the behavior of how API's are called, these methods intend to allow the legitimate code to call system services, at the same time, disallow the injected code.

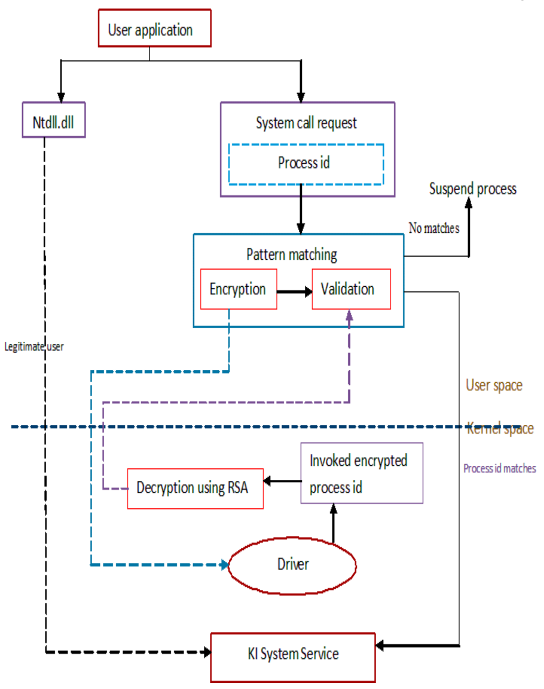

Figure.1. Proposed System Architecture B. Encryption function

Both the encryption and validation function is responsible for the integrity of the system. Integrity refers to the consistency and accuracy of data ensures that unauthorized users are prevented from modifying data authentication. Data transmission process will be protected to avoid any known or unknown changes of the data. In encryption function we need to concentrate on three mission caching registers, preparing a password and saving the stack pointer. A process reaches this function eventually enter to kernel driver and return to the validation function. The register might be modified and causes data integrity problem so that we need to back up the registers in advance by pushing them into the stack. After backing up, we further encrypt the process id of the process which is requesting for the system service with the help of RSA algorithm. We use two keys for encryption and decryption respectively. Encryption function is done with the help public key e, along with the 23bit process id. The encrypted process id will be stored in the stack, which will be invoked during the validation function.

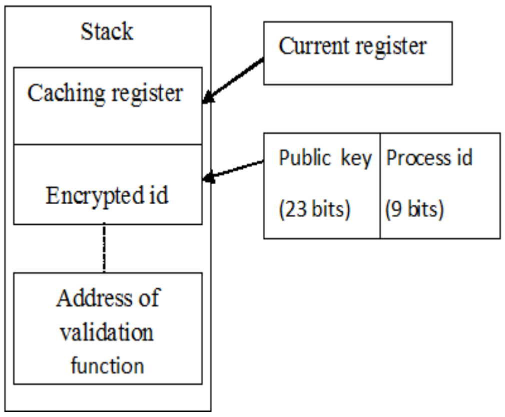

Figure.2. Process in encryption function

RSA Algorithm: The RSA algorithm is proposed by Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. RSA is a public key cryptosystem and widely used for secure data transmission. It is an asymmetric cryptographic algorithm. Asymmetric key means there is two different keys used. Public key is known by everyone and it is used for encryption and private key which is known only to the receipt for decryption. In our approach we use this algorithm to encrypt the process ID of the suspicious process and decrypt the same process in kernel mode. This behavior clearly analyzes the hooking behavior of the external activities by the root kits.

Steps,

- Choose two different prime numbers p and q.
- Compute $n = pq$,
- Calculate phi, $\varphi = (p-1)(q-1)$.
- Choose an integer e, such that $e < n$, $\gcd(e, \varphi) = 1$.
- Compute $d = e^{-1} \bmod \varphi$.
- Encryption, $c = m^e \bmod n$, $1<m<n$.
- Decryption, $m = c^d \bmod n$.

In proposed algorithm, the two large prime numbers will be chosen randomly.

Driver: The driver is responsible for the protection mechanism in the kernel mode. Windows driver model (WDM) effectively shields the driver. Before handling a request, the driver has to determine whether the request is coming from legitimate code or from suspicious code by validating the request. If the request comes from the legitimate code it is allowed to get the system service directly. If this request has not authenticated already means, the driver has to record the encrypted id, its process ID, and thread ID of the incoming process request. Then decrypt the encrypted id with the private key taken from the SSDT. Then the Driver redirects the control to the validation function.

Validation function: The validation function fetches the encrypted id stored in user mode stack of Encryption function and then decrypted id it to compare comes from the driver. If it matches, then it is considered as a legitimate process and get the system service directly, otherwise the request may come from malicious process. Hence it will be suspended before it gets the system service.

2. EXPERIMENTAL ANALYSIS

The first experiment investigated that the overhead will differs for each system call with hooks. The average numbers of CPU cycles spent for calling the system call by the c program were recorded. Therefore, after passing the SSDT the system calls were stopped and it will not carrying the actual request. For the hooked test, most of the running time should be spend on the proposed protection mechanism. This can help us to estimate the actual running time of the proposed mechanism. The second experiment illustrated the overhead when protected system calls are called by C functions. In each function one API was hooked. Each experiment was conducted for 10 times in both the original system and the system with the protection mechanism. We measured CPU cycles consumed and took the average as the experimental results. The third experiment is conducted to measure the performance of the system while running some real-life applications. In this experiment, we measure the time consumed by running several applications that are both computation and I/O demanding. They included Win RAR, internet explorer, and regedit.exe files. This should reflect the overhead of our scheme incurred to the system. At last, an experiment was analyzed the loading times of different programs. The first one did not apply any protection mechanism and reflected the time taken to invoke a process by the OS. The second one loaded along with the protection mechanism. Popular applications include Internet Explorer, Win RAR, Visual C++, MS Word, and Cute FTP are subjected to the experiment. Number of CPU cycles consumed in loading different programs for each case will be recorded.

Table.1. Experimental Analysis of Cpu Cycles For Various Real Time Application

Program being examined	Protected system call	Results		
		Before hook	After hook	%
WinRAR	Zwcreatefile Zwopenfile Zwopenkey	32.5	34.6	6.5
Ms- word	Zwcreatefile Zwopenfile Zwenumeratekey	13.2	16.6	9.7
Cute ftp	Zwopenkey Zwopenfile	36.7	40.5	1.3

3. RESULTS AND DISCUSSION

The process request for system service will be list out into active process and suspicious process. The active process is the legitimate user so it directly accesses the system service from the KI system service dispatch. But the suspicious process will be going for encryption function. The encryption function will encrypt the process id with the help of public key and stored in stack. Finally the control is forwarded to kernel driver there the encrypted id will get decrypted with the help of private key stored in SSDT. After validating the process id the suspicious process will be suspended.

The proposed system contain some properties and they are,

- Not requiring the source code of the protected program.
- Not needed to reboot Windows kernel.
- It can able to stop malicious code in runtime.
- Incurring less than 10 percent of running time overhead.

This scheme does not crash the system; instead, the suspicious process will be stopped by the validation function directly. Next, attacker can guess the key of a particular service after several trials. In fact, the attacker only needs to try at most 284 times to locate the original ID. This scheme does increase the security strength as 23-bit password and also by providing random key generation the attacker find hard to guess the key. Third, their scheme has difficulty to permute system services with different number of parameters. This scheme also can avoid this problem.

4. CONCLUSION

Process of a user level application needs to prove its identity to the kernel before accessing the system services. And the information such as process name or process id is not enough to identify a process by an operating system. That's results; malware may make other process to acquire the system service. In the proposed system, the process id has to validate to check whether it is legitimate process or suspicious process with the help of RSA algorithm. For that, a secret key will be issued to encrypt the process id and the key will be maintain in the SSDT of the system, the encrypted process id will be invoked at the kernel end and decrypts it for validation purpose. If it is a valid process then it allows acquiring the service. However, the system avoids the kernel crashing.

REFERENCES

Deian Stefan1, Chehai Wu, Danfeng (Daphne) Yao and Gang Xu, Knowing Where Your Input is From, Kernel-Level Data-Provenance Verification, This work has been supported in part by REU programs, NSF grant CCF-0728937,CNS-0831186, CNS-0953638, 2014.

Hussain M.J, Almohri, Danfeng (Daphne) Yao and Dennis Kafura, Process Authentication for High System Assurance, IEEE transactions on dependable and secure computing, 11(2), 2014.

Jinku Li, Zhi Wang, Tyler Bletsch, Deepa Srinivasan, Michael Grace and Xuxian Jiang, Comprehensive and Efficient Protection of Kernel Control Data, IEEE transactions on information forensics and security, 6(4), 2011.

Jinpeng Wei1, Feng Zhu1 and Calton Pu2, KQguard, Binary-Centric Defense against Kernel Queue Injection Attacks, Springer-Verlag Berlin Heidelberg, 2013.

Mohd Anuar Mat Isa, Jamalul-lail Ab Manan, Raja Mariam Ruzila Raja Ahmad Sufian, Azhar Abu Talib, An Approach to Establish Trusted Application, Second International Conference on Network Applications, Protocols and Services, 2010,

Patrice Clemente, Jonathan Rouzaud-Cornabas, and Christian Toinard, From a Generic Framework for Expressing Integrity Properties to a Dynamic MAC Enforcement forOperating Systems, M.L. Gavrilova, (Eds.): Trans. on Comput. Sci. XI, LNCS 6480, 2010, 131–161.

PENG Chao, YANG Xing-qiang, NIU Zhen-zhou, LIU Xiang-peng, Research on Windows Operating System Education, Supported by university relations of Microsoft research Asia, 2010.

Pradnya Patil, Shubham Joshi, Kernel Based Process Level Authentication Framework for Secure Computing and High Level System Assurance, International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) 2(12), 2014.

Shadi R, Masadeh, Ahmad Azzazi, Bassam A.Y, Alqaralleh and Ali, Mousa.Al Sbou, A Novel Paradigm In Authentication System Using Swifi Encryption /Decryption Approach, International Journal of Network Security & Its Applications (IJNSA), Vol.6(1), 2014.

Takamasa Isohara, Keisuke Takemori, Yutaka Miyake,Ning Qu, Adrian Perrig, LSM-based Secure System Monitoring Using Kernel Protection Schemes, International Conference on Availability, Reliability and Security, 2010.