White List Security Management Mechanism based on Trusted Computing Technology

Baohua Zhao1,a, Hao Zhang2,a, Hao Guo1,a, Yue Qi2,a
1 China Electric Power Research Institute, Beijing 100192, China
2 State Grid JIBEI Electric Power Company Limited, Beijing 100053, China
aemail: cuizhanhua@163.com

Abstract

A security management mechanism with white list is proposed based on trusted computing technology. The mechanism runs dynamic measurements to verify their integrity when the software or the program starts, which is an active defense mechanism based on trusted computing technology. It can effectively prevent unknown malicious programs, progresses or codes running to get sensitive information, which does not have disadvantages of some traditional antivirus software, such as feature code or virus database update, patching or bugs fix, etc.

Keywords: Trusted Computing; White List; Dynamic Measurement

Introduction

A "white list" is a set of concepts in contrast to "blacklist". The so-called "white list" refers to rules set up in the allowed list, which are "good" or "allowed". By contrast, the "blacklist" means "bad" or "not allowed". "Application white list" is a group of applications which are allowed to run in the system[1-6].

The antivirus software principles will help us understand the application white list management system. In fact, antivirus software virus is a blacklist. When the program runs, the antivirus program matches the corresponding rules in the blacklist, it proves corresponding programs may be viruses, Trojans and other harmful programs if the match is successful[8-11]. However, the blacklisting antivirus software mechanism has two drawbacks. First, the scope of viruses or Trojans is determined in the blacklist, it will easily being threatened by "Zero-day" attacks when a new threat is not on the blacklist list and the virus database has not being updated. Second, more and more viruses and Trojans will result in unlimited expansion of blacklisting. Rules matching will be a time-consuming job when blacklisting reaches a certain scale, which is the reason the computer with anti-virus software installed will run comparatively slow after running a period of time.

Known as the "next generation of information security products", the application white list technology can become a substitute for antivirus software, preventing malicious software attacks and unauthorized programs running, and so on. Meanwhile, it can solve the "Zero-day" attacks and performance problems providing the system's safety and system operating in control from the application level. Application white list management system can meet the user demand from the aspect of safety, performance and functionality, therefore, there are some advantages to develop white list security management mechanism based on trusted computing technology.

First of all, such a mechanism is the need of application white list to manage system security [12-14]. It is a very challenging task to protect the integrity and consistency of a software system if the application white list system is developed on the basis of pure software systems. In contrast, the trusted computer system is a hardware-based security platform, which can solve the insecurity of computer and network infrastructure, providing high security with the trust chain established from chip, hardware and operating systems.

Second, it is needed to ensure that the operating system loads procedures and core files with integrity and security. Currently, many viruses and Trojans target are for vulnerabilities of the operating system itself, infecting core operating system files and installed programs. Application white listing management systems based on trusted computing technology apply the principle of mandatory access controls and mechanisms and implement the operating system integrity checking, which enhances the security level of the entire operating system.

Trusted computing platform and trust mechanism

Trustworthy refers to "behave at realization of the given target is always of an entity as well as expected results." Trusted computing [15-19] is a trusted component such that operation or process in any operating conditions is predictable, and it is able to resist the bad code and the devastation caused by a physical disturbance. Trusted computing is the foundation of security, starting from the trusted root and solving the security problems resulted from the structure of the PC. It has the following features:(1) providing the identity, competence, integrity and availability of workspace of users; (2)providing the confidentiality and integrity of the storage, processing, transmission; (3) protecting the hardware environment configuration and the integrity of the operating system kernel, services and applications; (4)providing the safety of key operations and storage. Trusted computing passwords support platform for trusted cryptography module run as a trusted root and platform for its security management functions through the following three types of mechanisms:(1)Starting with the credible measurement, computing system platform integrity measurements, establishing trust chain computer system platform, ensure that the system credible;(2)Credible reporting credible identifies platform, uniqueness, based on credible reports, and platform identity and integrity of reporting;(3)Based on trusted storage root key management, platform data security features to provide the appropriate password service.

As we know, trusted computer works with Trusted Platform Module security chips, which run as a trusted root to measure the hardware configuration, operating system, applications, and the overall platform integrity. Trusted Platform Module stores integrity data to protect integrity of the platform status report and judge the credibility of a platform to ensure that the interactive platform is not infected by malicious programs.

The concept and principle of white list

White list security management system based on trusted computing [22-26] ensure that the operating conditions are safe in its full life-cycle. Before the system is loaded, the system verification and protection is implemented by the trusted computing platform from a hardware perspective; When the system is loaded, it is verified by trusted roots layer by layer, which guarantee the safety and consistency of white list itself; After the system is loaded, the white list runs to protect the safety of the system. The proposed white list security management system provided security in whole life cycle, covering the system loading, ruing and using.

The white list security management mechanism based on trusted computing performs security check before and after the system startup with the following procedures:

Step 1: After the computer starts, its control it taken over by the underlying trust root;

Step 2: Trusted root verifies the integrity of BIOS, and transfers control to BIOS after its integrity check is successful;

Step 3: BIOS MBR verifies the integrity of the operating system loader, and transfers control to the operating system loader after its integrity check is successful;

Step 4: The operating system loader verifies the integrity of the operating system kernel and critical components, transfers control to the operating system after successful;

Step 5: Operating system completes the white list validation and loads system;

Step 6: White list management system checks the security of operating system core.

When the computer runs in normal operation phase, white list security management mechanism is enabled to protect individual applications running in computer.

White list management based on trusted computing

White list mechanism is mainly used for dynamic measurement during the programs startup. Trusted computing technology is used to develop white list management mechanism, which runs an executable program integrity check to prevent them from malicious code and other attacks.

To control all executable code loaded from the system, trusted white list scan tools are developed to record full execution path and abstract value. Before loading the code, all executable files have to be checked and compared with trusted white list, they will not be allowed to run if the executive summary does not match. The white list mechanisms are implemented in the kernel loop filter layer 2, as shown in figure 1.

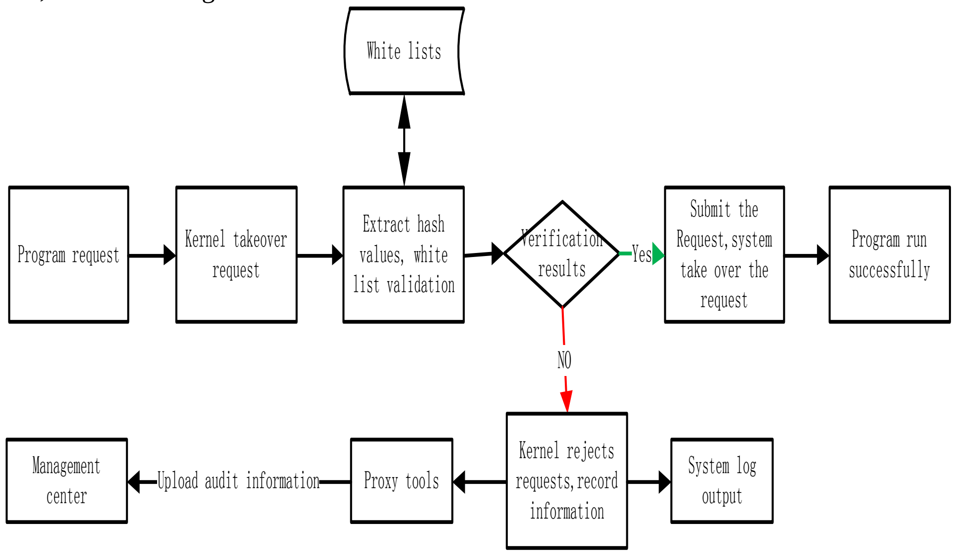

Fig.1. White list security management mechanism

(1) Definition of white list

White list is made up of three categories (shown in figure 2), including local white list, network white list, a temporary white list:

Category1: Local white list refers to system white list and installed program white list (software right acquisition mode), which is maintained by the client. System white list is generated by scanning interface when it is installed and it cannot be used as software templates. Installed program white list is generated by installed program interface, software scanning interface and network control scanning interface in software acquisition privilege mode.

Category 2: Network white list refers to software template white list and abnormal program white list which is maintained by the Central Administration. Software Templates white list is generated by management center when the client program group reports to the management center. The client can use software template to install template software without software acquisition privilege mode. Abnormal program white list is generated by the unknown program approval process.

Category3: Temporary white list refers to upgrade program white list which is automatically generated when the software upgrades and they will be automatically added to the white list when the system starts. Upgrade program white list is generated by the upgrade process.

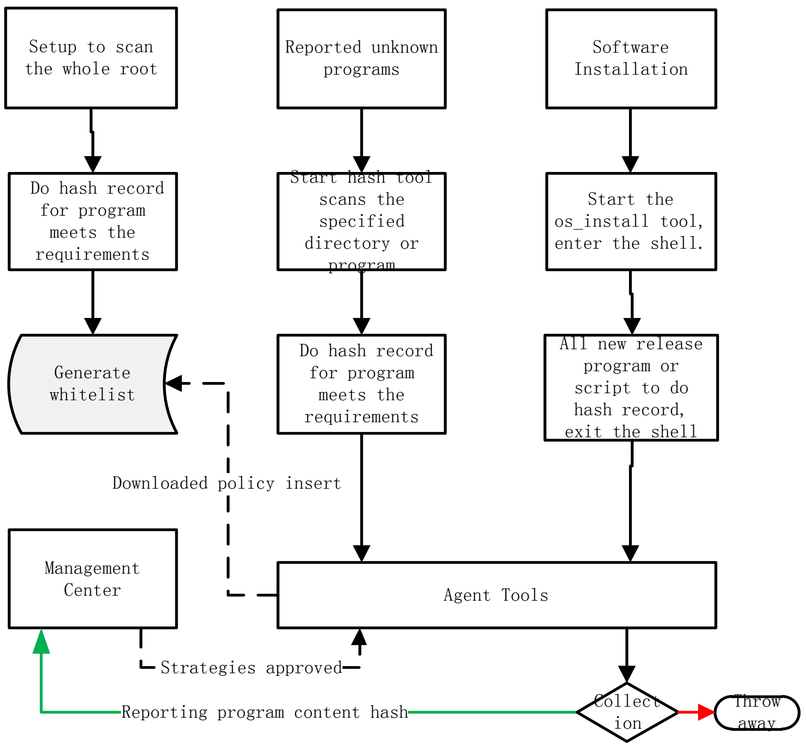
Setup to scan the whole root
Reported unknown programs
Software Installation
Do hash record for program meets the requirements
Start hash tool scans the specified directory or program
Start the os_install tool, enter the shell.
Generate whitelist
Do hash record for program meets the requirements
All new release program or script to do hash record, exit the shell
Downloaded policy insert
Management Center
Agent Tools
Strategies approved
Reporting program content hash
Collection
Throw away

Fig.2. White list generation Business Process

All white list files in Linux are stored in the "white list" file. For programs in white list repository, the mechanism provides tamper-proof protection and prohibition of non-authorized modification so that they cannot be renamed, changing position, modified and deleted. In the default case, only upgrade programs or executive programs created by upgrade programs can perform operation of add, deletion and changing for all executive programs in the white list repository. The modified programs can run immediately and stored into a temporary file, which will be written into white list after the operating system restarts. Network white list does not have anti-tampering function.

4.2 Software installation control by white list mechanism

Software installation control with white list mechanism is shown in figure 3. The mechanism provides software installation interface to install applications for the way of the software installation package. During the installation of the software, white list is generated by scanning interface and permit them to run.

Client software are installed in two ways: with software acquisition right and without software acquisition right, which will help control the installment permissions in desired granularity.

Situation 1: software installment with software acquisition right. When the client is the acquisition terminal, it can authorize the client with "software acquisition" right by management center. Then, software could be installed by installment interface of the client. Moreover, these programs will be added into local white list and run immediately. Meanwhile, with except of system white list, the installed software white list can be exported to as policy files, which will be reported to management center to determine whether they will be deployed into other clients.

Situation 2: software installment without software acquisition right. In default, the client does not have acquisition right. Only template software from management center can be installed by software installment interface in such situation, and they will be automatically added into network white list and run immediately.

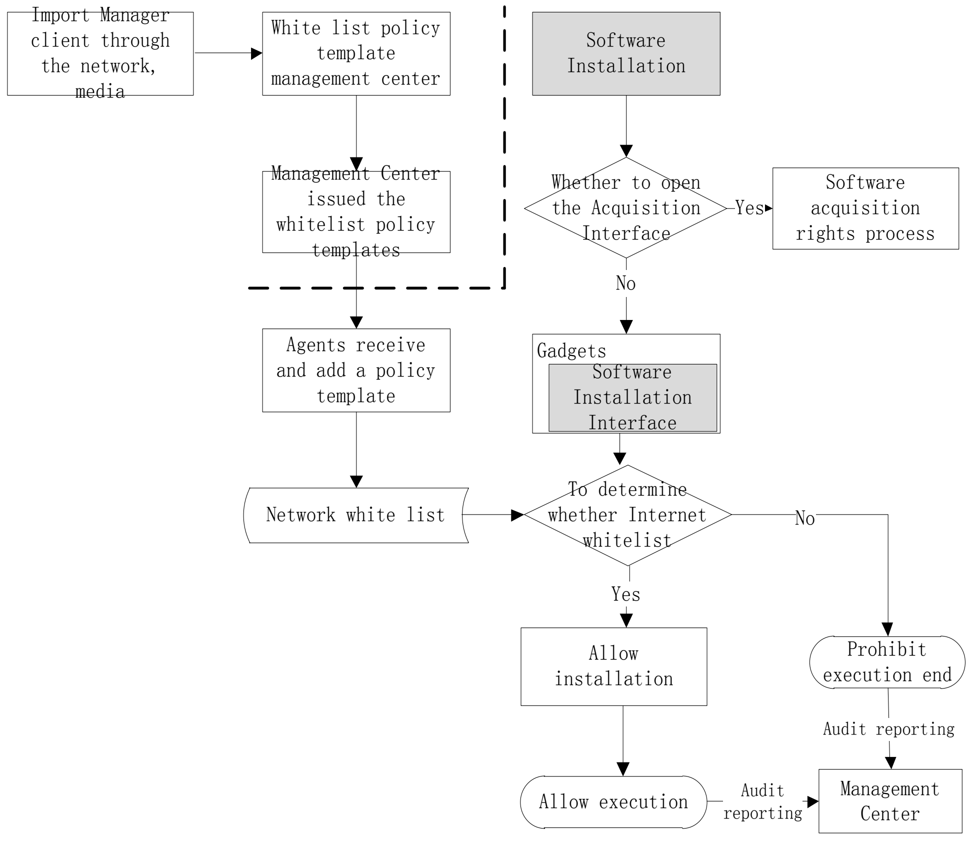
Import Manager client through the network, media
White list policy template management center
Software Installation
Management Center issued the whitelist policy templates
Whether to open the Acquisition Interface
Yes
Software acquisition rights process
No
Agents receive and add a policy template
Gadgets
Software Installation Interface
Network white list
To determine whether Internet whitelist
No
Yes
Allow installation
Prohibit execution end
Audit reporting
Allow execution
Audit reporting
Management Center

Fig.3. Software installation control by white list mechanism

Experiment results

The white list management mechanism is tested with NetLogo simulation software based on a software startup policy control. The experiments results showed high untrusted software detection rate, which proved that the mechanism developed runs effectively. The experiments are completed with Intel(R)

Pentuim(R) 2.90G, 4G RAM, and Win7. Table 1 shows the experimental parameters in details.

Table 1 The simulation parameters

	Initial	Parameters Description
Network N	100	Number of entities in the network environment
Environmental parameters M	300	A snapshot of the number of group members
Algorithm parameters	m	The value 0 or 1,indicating whether the matching white list

Within a set period of time, the number of the software to be detected is m and the initial credibility is 60%. The proportion of untrusted software in the software library over time is as follows:

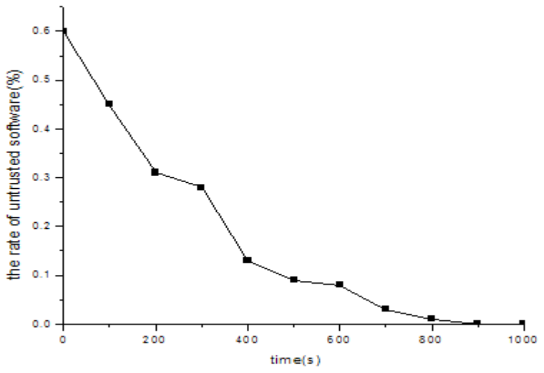

Fig.4. Detection efficiency of white list mechanism

Conclusion

The white list security management mechanism for software installment is developed base on trusted computing technology, which is an active protection defense mechanism. All software will be dynamically measured by such mechanism to check their integrity to protect them from tampering, modifying and changing. The simulation results proves that the mechanism could provide high secure protection for software installment and operation, which will prevent them from infected or destroyed by malicious program or codes..

References

[1] Masoom Alam, Xinwen Zhang, Mohammad Nauman, Model-based Behavioral Attestation [A]. Proceedings of 13th ACM symposium on Access control models and technologies[C], New York: ACM Press, 2008:175-184.

[2] Masoom Alam, Xinwen Zhang, Mohammad Nauman, Tamleek Ali. Behavioral Attestation for Web Services (BA4WS)[A]. Proceedings of the 2008 ACM workshop on Secure web services[C], New York :ACM, 2008:21-28.

[3] Masoom Alam, Mohammad Nauman, Xinwen Zhang, Tamleek Ali, Patrick C.K. Hung. Behavioral Attestation for Business Processes (BA4BP)[A]. Proceedings of 2009 IEEE International Conference on Web Services[C], IEEE Press, 2009:343-350.

[4] Mohammad Nauman, Masoom Alam, Xinwen Zhang, and Tamleek Ali. Remote Attestation of Attribute Updates and Information Flows in a UCON System[A]. Proceedings of the 2nd International Conference on Trusted Computing[C], Berlin, Heidelberg: Springer-Verlag, 2009:63-80.

[5] Clark Weissman. Security Controls in the ADEPT-50 Time Sharing System[A]. Proceedings of the 1069 AFIPS Fall Joint Computer Conference[C], AFIPS Press, 1969:119-133.

[6] E. I. Organick. The Multics System: An Examination of Its Structure[M]. Cambridge, Mass. MIT Press, 1972.

[7] National Computer Security Center. Final Evaluation Report of Multics[R]. MR11.0, CSC-EPL-85/003, Ft. George G. Meade, MD, 1985.

[8] Vito B. L. D, Palmquist P. H, et al. Specification and Verification of the ASOS Kernel[A]. Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy[C], Oakland, California, 1990:61-74.

[9] Waldhart N. A. The Army Secure Operating System[A]. Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy[C]. Oakland, California, 1990:50-60.

[10] Blotcky, K. Lynch, S Lipner. SE/VMS: Implementing Mandatory Security in VAX/VMS[A]. Proceedings of the 9th National Computer Security Conference. Gaithersburg[C], Md. National Bureau of Standards, 1986:47-54.

[11] B. Pfitzmann, J. Riordan, et al. The PERSEUS System Architecture[R]. IBM Technical Report NO.93381, IBM Research Division, Zurich, 2001.

[12] Secure Computing Corporation. Assurance in the Fluke Microkernel: Final Report[R]. CDRL Sequence NO.A002, Secure Computing Corporation, 1999.

[13] Secure Computing Corporation. DTOS Lessons Learned Report[R]. CDRL Sequence No.A008, Secure Computing Corporation, Rosevile, Minnesota, 1997.

[14] Secure Computing Corporation. DTOS Generalized Security Policy Specification [R]. DTOS CDRL A019, Secure Computing Corporation, Roseville, Minnesota, 1997.

[15] A. L. Peter, D. S. Stephen. Integrating Flexible Support for Security Policies into the Linux Operating System[R]. NSA and NAI labs, 2001. [16] Intel. LaGrande: Technology Architectural Overview[R]. Intel White Paper, Intel, 2003.

[17] G. David. LaGrande Architecture[R]. Intel White Paper, Intel, SCMS-18, 2003.

[18] Microsoft, Microsoft Next-Generation Secure Computing Base: An Overview[EB/OL], http://www.micro soft.com/resource/ngscb/ngscb_overview.mspx, 2003-4-1.

[19] P. England, B. Lampson, et al. A Trusted Open Platform[J]. IEEE Computer, 2003,36(7): 55-62.

[20] R. S. Sandhu, E. Coyne, et al. Role-Based Access Control Models[J]. IEEE Computer, IEEE Press, 1996:29(2):38-47.

[21] H. Mantel, D. Sands. Controlled declassification based on intransitive non-interference[A]. Proceedings of APLAS[C], 129-145, 2004.

[22] J Park. Towards usage control models: beyond traditional access control [A]. Proceedings of seventh ACM symposium on Access control models[C], ACM Press, 2002.

[23] L. Badger, D. F. Swme, et al. Practical domain and type enforcement for UNIX [A]. Proceedings of IEEE Symposium on Security and Privacy[C], 1995: 66-77.

[24] Safford D, Zohar M. A trusted Linux client[A]. Proceedings of 2004 Annual Computer Security Applications Conference[C], Hilton Tucson, 2004.

[25] Ahmed M. Azab, Peng Ning, Emre C. Sezer, Xiaolan Zhang. HIMA: A Hypervisor-Based Integrity Measurement Agent[A]. In Proceedings of the 25th Annual conference on Computer Security Applications[C], Piscataway NJ:IEEE, 2009:461-470.

[26] Lionel Litty, H. Andres Lagar-Cavilla, David Lie. Hypervisor Support for Identifying Covertly Executing Binaries[A]. Proceedings of the 17th conference on Security Symposium[C]. Berkeley: USENIX, 2008:243-258.