Research on the Application Security Isolation Model

Lei Gong[1,2,3], Yong Zhao[3], and Jianhua Liao[4]

[1] Institute of Electronic Technology, Information Engineering University, Zhengzhou, China
[2] Key Lab of Information Network Security, Ministry of Public Security, Shanghai, China
gonglei_sky@sohu.com
[3] Institute of Computer Science, Beijing University of Technology, Beijing, China
zhaoyonge_mail@sina.com
[4] School of Electronics Engineering and Computer Science Peking University, Beijing, China
liao_jh@139.com

Abstract. With the rapid development of information technology, the secrutiy problems of information systems are being paid more and more attention, so the Chinese government is carrying out information security classified protection policy in the whole country. Considering computer application systems are the key componets for information system, this paper analyzes the typical security problems in computer application systems and points out that the cause for the problems is lack of safe and valid isolation protection mechanism. In order to resolve the issues, some widely used isolation models are studied in this paper, and a New Application Security Isolation model called NASI is proposed, which is based on trusted computing technology and the least privilege principle. After that, this paper introduces the design ideas of NASI, gives out formal description and safety analysis for the model, and finally describes the implementation of the prototype system based on NASI.

Keywords: Information security classified protection, Application security, Security model, Application isolation.

1 Introduction

Nowadays, information security problems are being paid more and more attention in the world. The Chinese government decreed classified criteria for security protection of computer information system in 1999, and since then a lot of regulations were being released, which confirm that information security classified protection is the basic policy for information security construction in China.

Computer application systems are the key components for information system. The typical security problems are followed. Firstly, hackers usually explore security vulnerabilities in application to compromise computer systems, promote their privileges, and then access sensitive information or tamper some significant data. Secondly, there is some interference among different application systems because of user's misoperation, mutual confusion system data and so on. Thirdly, malicious code (malware) such as viruses, worms and Trojan horses always infiltrates computer systems, and badly threats security of application systems.

The basic reasons of those security problems mentioned above are confusion of application environment and fuzzy application boundary. So the most effective way to resolve those security problems is application isolation [1].

2 Related Work

The typical security model focusing on application isolation mainly includes sandbox model, virtualization model and noninterference information flow model.

The sandbox model restricts the actions of an application process according to security policies, so the process can only influence limited areas. For instance, Java virtual machine [2][3], Sidewinder firewall [4] and Janus [5] are the typical sandboxes. The sandbox model can also record the behaviors of processes [6]. It utilizes copy-on-write technology to make the system recoverable after being attacked.

Virtualization model tends to project implementation. VM Ware, Virtual PC and Xen virtualization are on hardware layer, which virtualizes CPU, memory, peripheral interface and so forth. FreeBSD jail and Solaris Containers (including Solaris Zones) virtualization are on operating system, which intercepts system calls to build an independent execution environment.

Noninterference information flow model is based on noninterference theory, which is firstly proposed by Goguen and Meseguer [7]. Noninterference theories are significant means to analyze information flow among components and reveal covert channels [8], but it does not provide additional solution to isolate application.

In summary, sandbox model focuses on constraining behaviors of process and neglects the protection of sensitive objects. Virtualization model can carry out complete application isolation, but it is not easy to be deployed under the complex application circumstances. Noninterference information flow models are theory model and the interference behaviors in information system are very multiplex, so it is difficult to be implemented.

3 Application Security Isolation Model

In this section, we will introduce an application security isolation model called New Application Security Isolation (NASI) model, which is based on the trusted computing technology and the least privilege principle.

3.1 An Overview of NASI

The NASI model divides resources for application environment into several parts, and sets up trusted and untrusted domains. In Trusted Computer System Evaluation Criteria (TCSEC) [9], domain means objects set which subjects can access. While in NASI model, the concept of domain does not mean a single set of objects, but an execution environment in which subjects with the least privilege can access objects, as shows in figure 1. In the domain, subjects are the application processes and objects are the resources including memory spaces, configuration files, program files, data files and so on. Some of the resources are public and some of them are private, as a whole, both of which can be seen as an independent resource set mapping to a specific application program.

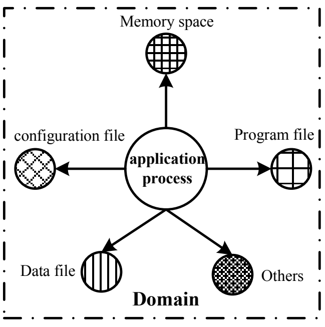

Fig. 1. Domain in NASI model can also be called application execution environment

The attribute of domains in NASI is either trusted or untrusted. In trusted domain, the program has normal and safe source, such as qualified software vendor. Processes in trusted domain can not only access the resources in the same domain, but also can access the resources in other trusted domains on the basis of security policies. In untrusted domain, the program has abnormal and unsafe source, such as Internet. Processes in untrusted domain can only access limited resources in their own and are unable to access resources in others.

3.2 Formal Description of NASI

Definition 1. Let Sub be a set of subjects in application environment, S for subjects in domains, then $Sub = \{S_1, S_2 \cdots S_n\}$; let Obj be a set of objects in application environment, O for objects in domains, O_{pub} for public objects, O_{pri} for private objects, then $O = O_{pub} + O_{pri}$, $Obj = \{O_1, O_2 \cdots O_n\}$; let $A = \{r, rw, w\}$ be a set of access modes, r for read only, rw for read/write, w for write; let R be requests for access, yes for allowed, no for deny, $error$ for illegal or error, so $D = \{yes, no, error\}$ denotes the set of outcomes for requests.

Definition 2 Trusted Domain. $TrustDom = \{N, S, O, A, P, TR\}$, N denotes domain ID, P denotes security policies, TR denotes trust relationship among domains.

Definition 3 unTrusted Domain. $unTrustDom = \{N, S, O, A, P\}$, the elements in untrusted domains are like trusted domains, except for lack of trust relationship TR .

Definition 4 Belonging relationship. $Host(O_i, t) = S_i, t \in T$, it means that the resources O_i belong to the process S_i at the moment t in a domain.

Property 1 Dynamic Property: $\exists (t_p, t_q \in T) t_p \neq t_q \wedge O_{it_p} \neq O_{it_q}$. This property means that during the procedure of program execution, some new resources will be created and some useless resources will be deleted.

Property 2 Belonging Property: $\forall(t_p \neq t_q \in T, O_i \in O)$, $Host(O_i, t_p) \equiv Host(O_i, t_q)$. This property indicates that although resources in domains are variational, they always belong to the process of their own, that is $Host(O_i, t) = Host(O_i)$.

Property 3 Base Property in Domains: if $\forall S_i, O_i \in unTrustDom, S_j, O_j \in TrustDom$, $\exists Host(O_i) = S_i, Host(O_j) = S_j$, then $S_i \times O_i \times A = \{yes\}$, $S_j \times O_j \times A = \{yes\}$. This property indicates that if processes and resources belong to the same domain, then the processes could access the resources.

Property 4 Base Property between Trusted Domain and Untrusted Domain: if $\forall S_i, O_i \in unTrustDom, S_j, O_j \in TrustDom$, $\exists Host(O_i) = S_i, Host(O_j) = S_j$, then $S_j \times O_i \times A = \{no\}$, $S_i \times O_j \times A = \{no\}$. This property indicates that there is no information flow between trusted and untrusted domain.

Property 5 Base Property among Trusted Domains:if $\exists TR_{ij} = TrustDom_i \succ TrustDom_j$, $\forall S_i, O_i \in TrustDom_i, S_j, O_j \in TrustDom_j$, $\exists Host(O_i) = S_i, Host(O_j) = S_j$,then $S_j \times O_i \times A = \{yes\}$. This property indicates that if one domain trusts another ($\succ$ means one-way trust), there will be information flow between them.

The properties above are very elementary, so NASI model has the following specific definitions and properties as complementarities.

Definition 5. Let C be a set of sensitivity level, L be the range of sensitivity level, and $L = \{[C_i, C_j] \mid C_i \in C \wedge C_j \in C \wedge (C_i \leq C_j)\}$, which means that its sensitivity level is between C_i and C_j .If $C_i = C_j$, then it represents single sensitivity level. Supposing $L_1 = [C_{1i}, C_{1j}] \in L$, $L_2 = [C_{2i}, C_{2j}] \in L$, then $L_2 \geq L_1 \Rightarrow (C_{2i} \geq C_{1j})$ and $L_2 \subseteq L_1 \Rightarrow (C_{2i} \geq C_{1i} \wedge C_{1j} \geq C_{2j})$; let $L_s L_o$ represent sensitivity level of subject and object respectively.

Definition 6. Let $V = B \times M \times F \times H$ be the set of system states, where $B \subseteq (Sub \times Obj \times A)$ denotes subjects access objects with privilege A , M is a set of access control matrices, $F \subseteq L_s \times L_o$ is the set of sensitivity levels for subjects and objects, $f = (f_s, f_o) \in F$, $f_s f_o$ denote sensitivity level of subject and object respectively, and H represents the set of hierarchy functions of objects. Furthermore, the set $W \subseteq R \times D \times V \times V$ is the set of behaviors of the system.

Property 6 Read Property: a state $v = (b, m, f, h) \in V$ satisfies this property if and only if, for each $(s, o, a) \in b$ the following holds: $a = r \Rightarrow$

$$\begin{cases} f_s(S) > f_o(O) \wedge S \in TrustDom_i \wedge O \in TrustDom_i \\ O = O_{pub} \wedge S \in TrustDom_i \wedge O \in TrustDom_j \wedge TrustDom_j \succ TrustDom_i \\ O = O_{pri} \wedge f_s(S) > f_o(O_{pri}) \wedge S \in TrustDom_i \wedge O \in TrustDom_j \wedge TrustDom_j \succ TrustDom_i \end{cases}$$

This property indicates that if processes and resources belong to the same trusted domain and subject dominates object, then S can read O . If processes and resources belong to different trusted domains, for the public resources, as long as the domains have trust relationship, S can read O ; for the private resource, besides the conditions above, subject that is trusted must dominate object which is in the other domain.

Property 7 Write Property: a state $v = (b, m, f, h) \in V$ satisfies this property if and only if, for each $(s, o, a) \in b$ the following holds: $a = w \Rightarrow$

$$\begin{cases} f_o(O) > f_s(S) \wedge S \in TrustDom_i \wedge O \in TrustDom_i \\ f_o(O) > f_s(S) \wedge S \in TrustDom_i \wedge O \in TrustDom_j \wedge TrustDom_j \succ TrustDom_i \end{cases}$$

This property indicates that if processes and resources belong to the same trusted domain and object dominates the subject, then S can write O . If processes and resources belong to different trusted domains, besides the condition above, the domains must have trust relationship.

$a = rw \Rightarrow$

$$\begin{cases} f_s(S) = f_o(O) \wedge S \in TrustDom_i \wedge O \in TrustDom_i \\ f_s(S) = f_o(O) \wedge S \in TrustDom_i \wedge O \in TrustDom_j \wedge TrustDom_j \succ TrustDom_i \end{cases}$$

If processes and resources belong to the same trusted domain and the subject's sensitivity level is equal to the object's level, then S can read and write O . If processes and resources belong to different trusted domains, besides the condition above, the domains must have trust relationship.

3.3 Security Analysis

1. Defending Attack towards Software Vulnerability

NASI model can well cope with attack towards software vulnerabilities. With the least privilege principle, it can constrain permissions of a process by property 3 and 4. NASI model cannot prevent processes from being compromised, but can ensure that attack could not do anything out of permissions of compromised processes. So the permission that attacker gets is limited and it has no right to destroy the system or access sensitive resources of other applications.

2. Reducing Interference among Processes

NASI can provide separate environment for each application, prevent users from destroying systems by misoperation, and reduce interference among processes. For example, supposing there are two application systems called App1 and App2, which are deployed in two trusted domains separately. App1 supplies database resources for App2 to display. According to property 6, App2 could read the data file belonged to App1, but if App2 tries to modify the database resource, it will conflict with property 7. From this point of view, NASI model can reduce interference among applications if we set the security policies properly.

3. Resisting Malware Attack
NASI model can resist malwares and reduce the damages even if they get a chance to run. Because only legal processes have permissions to access sensitive information under protection of NASI model, it can prevent sensitive information from illegally being accessed by malwares. For example, supposing there is a malware which runs as process M and it tries to access O which is in another domain. Because M and O are not in the same domain, and they don't have trust relationship, according to property 3, 4 and 5, the access will be denied.

4 Implementation of NASI

The architecture of NASI prototype system is divided into four layers which are hardware layer, OS kernel layer, system layer and application layer, as shown in Fig.2. The main security mechanism is implemented in OS kernel layer and it is supported by TPM (Trusted Platform Module) chip as the root of trust, so we can guarantee the initial environment for applications to be safe, the procedure of which is from hardware power on to OS loading.

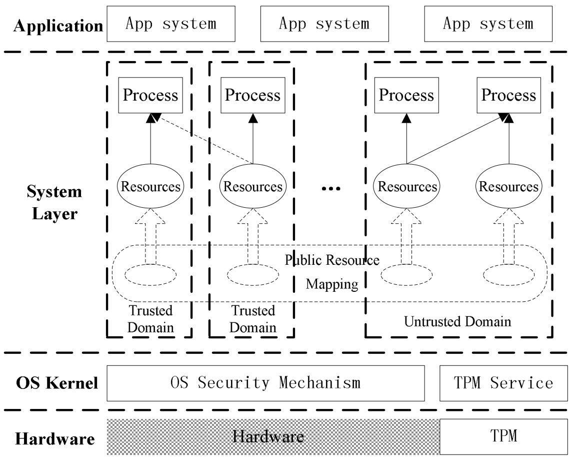

Fig. 2. The architecture of NASI prototype system

The NASI prototype system creates domains for each one of application. In the domain, the application process needs to utilize its own private resources and some of the public resources to accomplish the task effectively.

For private resources, the prototype system monitors them during the lifetime of application. The resources such as program files, configuration files and data files, which are created by application in deployment or in execution, belong to the same domain. For public resources, the prototype system uses virtualization technology to map public resources into different domains. When a process tries to access public resources, the prototype system will rename system resources at the OS system call interface [10]. For example, supposing an application in domain1 tries to access a file

/a/b, and then the prototype system will redirect it to access /domain1/a/b. When a process in domain2 accesses /a/b, it will try a different file /domain2/a/b, which is different from the file /a/b in domain1.

However, considering the performance overhead, a new created domain initially can share most of the public resources. Later on, if the processes in domain make only read requests, then they can directly access. But if they want to do some modification, the resources will be redirected to the domain to meet the requirement.

5 Conclusion and Future Work

In this paper, we introduced and implemented NASI model to satisfy the requirements of application security, which is very important in information security classified protection. From formalized description and security analysis, NASI can isolate application programs safely. Compared with other security model of related work, NASI can ensure not only security and validity, but also real feasibility.

In the future, we will pay more attention on how to measure trust degree for different domains and how to adjust trust degree during the application running.

Acknowledgement. This article is supported by the National High Technology Research and Development Program of China (2009AA01Z437), the National Key Basic Research Program of China (2007CB311100) and the Opening Project of Key Lab of Information Network Security, Ministry of Public Security.

References

1. Lampson, B.: A Note on the Confinement Problem. Communications of the ACM 16(10), 613–615 (1973)
2. Campione, M., Walrath, K., Huml, A.: and the Tutorial Team: The Java Tutorial Continued: The Rest of the JDK. Addison-Wesley, Reading (1999)
3. Gong, L., Mueller, M., Prafullchandra, H., Schemers, R.: Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2. In: Proceeding of the USENIX Symposium on Internet Technologies and Systems, pp. 103–112 (December 1997)
4. Thomsen, D.: Sidewinder: Combining Type Enforcement and UNIX. In: Proceedings of the 11th Annual Computer Security Application Conference, pp. 14–20 (December 1995)
5. Goldberg, I., Wagner, D., Thomas, R., Brewer, E.: A Secure Environment for Untrusted Helper Applications: Confining the Wily Hacker. In: Proceedings of the 6th USENIX Security Symposium, pp. 1–13 (July 1996)
6. Jain, S., Shafique, F., Djeric, V., Goel, A.: Application-level Isolation and Recovery with Solitude. In: EuroSys 2008, Glasgow, Scotland, UK, April 1-4 (2008)
7. Goguen, J., Meseguer, J.: Inference control and unwinding. In: Proc. Of the IEEE Symposium on Research in Security and Privacy, pp. 75–86 (1984)
8. Rushby, J.: Noninterference, Transitivity and Channel-Control Security Policies: Technical Report CSL-92-02, Computer Science Laboratory, SRI International, Menlo Park, CA (December 1992)
9. U.S. Department of Defense. Trusted Computer System Evaluation Criteria. DoD 5200.28-STD (1985)
10. Yu, Y., Guo, F., Nanda, S., Lam, L.-c.: A Feather-weight Virtual Machine for Windows Application. In: ACM Conference on VEE 2006, Ottawa, Ontario, Canada (2006)