A Privilege Separation Method for Security Commercial Transactions

Yasha Chen[1,2], Jun Hu[3], Xinmao Gai[4], and Yu Sun[3]

[1] Department of Electrical and Information Engineering, Naval University of Engineering, 430033, Wuhan, Hubei, China
[2] Key Lab of Information Network Security, Ministry of Public Security, 201204, Shanghai, China
cys925@hotmail.com
[3] School of Computer, Beijing University of Technology, 100124, Beijing, China
[4] School of Computer, National University of Defense Technology, 410073, Changsha, Hunan, China

Abstract. Privilege user is needed to manage the commercial transactions, but a super-administrator may have monopolize power and cause serious security problem. Relied on trusted computing technology, a privilege separation method is proposed to satisfy the security management requirement for information systems. It authorizes the system privilege to three different managers, and none of it can be interfered by others. Process algebra Communication Sequential Processes is used to model the three powers mechanism, and safety effect is analyzed and compared.

Keywords: privilege separation, fraud management, security commercial transactions, formal method.

1 Introduction

Information systems are widely used in commerce activities, business transactions and government services. Privilege user is needed to manage the commercial transactions in those systems, but a super-administrator may have monopolize power and cause serious security problem. In order to avoid it, security criteria is specified in GB17859[1] and TCSEC[2], in which stringent figuration management controls are imposed, and trusted facility management is provided in the form of support for system administrator and operator functions. Privilege control mechanism provides appropriate security assurance for commercial transactions system.

"Separation of privilege" is one of the eight principles Saltzer and Schroeder [3] specified for the design and implementation of security mechanisms. Separations of duty rules are normally associated with integrity policies and models [4, 5, 6]. Recent work in security management [7, 8, 9] designed multi-layered privilege control mechanism and implemented in security operating system. However, formal methods are hardly used to describe their methods, and the effects is not well proved.

Process algebra is a structure in the sense of universal algebra that satisfied a particular set of axioms, which was coined in 1982 by Klop and Bergstra [10]. Its tools are algebraically languages for the specification of processes and the formulation of statements about them, together with calculi for the verification of these statements. Communicating Sequential Processes (CSP) [11] specify system as a set of parallel state machines that sometimes synchronize on events, so it can mathematically precise statement of a security policy. [12] modeled the first noninterference security argument for a practical security operating system using the CSP formalism, and proved that the model fulfills an end-to-end property that protects secrecy and integrity even against subtle covert channel attacks.

The paper is organized as follows. Section 2 analysis the relationship with privilege control, security management and commercial transactions system, and then give a review of the three powers separation mechanism. Section 3 specifies each manager's privilege and the assembled model with CSP. Section 4 analyzes the security effect of the method, and proves that is safer than monopolize power mechanism.

2 Review of the Model

The separation of powers, is a model for the governance of democratic states, which constituted by the separation of "executive, legislative, and judicial powers". With the help of information system, commercial transactions can be conducted automatically, but privilege user is needed to manage the system. We have introduced this approach to implement privilege control mechanism which provides three different types of managers to exercise "decision making, enforcement, audit" privilege respectively, thus avoiding power of abuse. If a system exist only one monopolize administrator, he can subvert easily the security of the system, in Section 4, we'll prove that after adopting the approach of "separation of power" no administrator can use his own privilege to subvert the security of system.

The privilege users undertaking such a logic function named as "*system manager*, s*ecurity manage*r and *audit manager*". Their responsibility specified as

security manager-Unified mark all the subjects and objects of the system, and manage authorization of subjects.

system manager-Manage the system subject's identities and resources, configure the system.

audit manager-Manage various types of audit records of the storage, management and query, etc.

A monopolized privilege user can do anything he likes, so it is evident then, that our three powers mechanism is safer than monopolize power mechanism.

3 Formal Description

3.1 Mechanism Analysis

Reference monitor is a part of trusted computing base (TCB), always running, temper-resistant, and cannot be bypassed. In our model, the relationship between the reference monitor, operators and three types of managers are described as Fig 1:

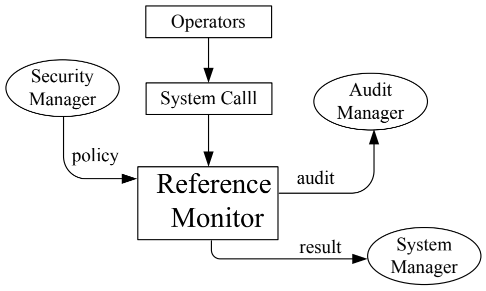

Fig. 1. Relation of all users and reference monitor

a). *Security manager* specify the policy that the reference monitor need to excute;
b). Reference monitor executes the policy and send the result to *system manager*;
c). *Audit manager* audits all system actions through reference monitor.

3.2 Communicating Sequential Processes

CSP is well suited to the description and analysis of operation system, because operation system and the relevant aspects of the users all can be described as processes within CSP. Our model investigates three managers' actions and their interactions, and then verifies certain aspects of their behavior through use of the theory.

The word *process* stand for the behavior pattern of an object [11]. A process is a mathematical abstraction of the interactions between a system and its environment. The set of names of events which are considered relevant for a particular description of the object is called its *alphabet*. Processes can be assembled together into systems, in which the components interact with each other and with their external environment.

In the traces model of CSP, a *trace* of a process is a finite sequence of symbols recording the events in which the process has engaged up to some moment in time.

We offer a brief glossary of symbols here:

αP	the alphabet of process *P*
$a \to P$	*a* then *P*
$(a \to P \vert b \to Q)$	*a* then *P* choice *b* then *Q* (provide $a \neq b$)
$P \vert\vert Q$	*P* in parallel with *Q*
$P \sqcap Q$	*P* or *Q*(non-deterministic)
$P \Box Q$	*P* choice *Q*
$P \setminus C$	*P* without *C*(hiding)
$P \vert\vert\vert Q$	*P* interleave *Q*
$P // Q$	*P* subordinate to *Q*
$b!e$	on channel *b* output *e*
$b?x$	on channel *b* input *x*

3.3 Privilege Separation

System manager, *security manager* and *audit manager* are denoted as CSP processes M_{SE}, M_{AU}, M_{SY}. We define their privilege as follows;

1) The privilege of *security manager* is defined as.

$$M_{SE} = TAGMGR|AUTMGR$$
$$TAGMGR = NEWTAG| \\ DELTAG| \\ MODTAG$$
$$NEWTAG = choose : CHOOSE|| \\ |_{\forall i}(i.g?(t,+)) \rightarrow \\ i.g!t \rightarrow TAGMGR$$
$$DELTAG = choose : CHOOSE|| \\ |_{\forall i}(i.g?(t,-)) \rightarrow \\ i.g!t \rightarrow TAGMGR$$
$$MODTAG = choose : CHOOSE|| \\ |_{\forall i}(i.g?(t,t^{'))\rightarrow} \\ i.g!t^{'} \rightarrow TAGMGR$$
$$AUTMGR = AUTHORIZE| \\ WITHDRAW$$

TAGMGR and *AUTMGR* are sub processes of M_{SE}.*TAGMGR* tags any system subject and object, which received from *SYSMGR*. It uses *NEWTAG* to create a tag, *DELTAG* to delete a tag and *MODTAG* to modify the original tag. *AUTMGR* uses *AUTHORIZE* to give access right for a subject, and uses *WITHDRAW* to cancel it.

2) The privilege of *audit manager* is defined as.

$$M_{AU} = ADATAMGR$$
$$ADATAMGR = EXPORT| \\ DELETE| \\ CHECK$$
$$EXPORT = choose : SELECT|| \\ |_{\forall i}(i.a?m) \rightarrow \\ i.a!m \rightarrow ADATAMGR$$
$$DELETE = choose : SELECT|| \\ |_{\forall i}(i.a?(m,-)) \rightarrow \\ i.g!r \rightarrow ADATAMGR$$
$$CHECK = choose : SELECT|| \\ |_{\forall i}(i.a?m) \rightarrow \\ i.a!m \rightarrow ADATAMGR$$

AUMGR uses *ADATAMGR* to manger audit data, *EXPORT* to export audit data, *DELETE* to clear useless audit files. *CHECK* can browse all data.

3) The privilege of *system manager* is defined as.

$$\begin{aligned} M_{SY} &= USERMGR|PROMGR|COFMGR \\ USERMGR &= ADDUSER| \\ &\quad DELUSER \\ ADDUSER &= choose : LIST|| \\ &\quad |_{\forall i}(i.u?(m,+)) \rightarrow \\ &\quad i.u!m \rightarrow USERMGR \\ DELUSER &= choose : LIST|| \\ &\quad |_{\forall i}(i.u?(m,-)) \rightarrow \\ &\quad i.u!m \rightarrow USERMGR \\ PROMGR &= INSTALL| \\ &\quad UNISTALL \end{aligned}$$

USERGMGR, *PROTMGR* and *COFMGR* are sub-processes of M_{SE}. *USERMGR* manages the information of system users. *ADDUSER* add a new system user and *DELUSER* delete a user and corresponding resource. *PROMG* manages the application program. It uses setup to add new program, and use *UNISTLL* to remove it. *COFMGR* manages all the configuration files.

3.4 Communication

Except executing self-responsibility, those three managers need to interact with others. The communication events between them can be clearly showed (Fig.2). Those communications are:

1) All the operations of *system manager* will be audited by *audit manager.*
2) All the operations of *security manager* will be audited by *audit manager.*
3) *system manage*r submit request to *audit manager* before the state transition.

We split each process $M_i(i \in \{SE, SY, AU\})$ into two logical components: a *application half* A_i, and a *TOOL half* $i\!:\!T$. A_i represents the behaviors of people (similarly to a user interface). $i\!:\!T$ represents the trusted system tool, it behaves according to a strict state machine. The two halves of the same manager communicate via the channel *s*. (CSP processes use *channel* to communicate. A channel is used in only one direction and between only two processes).

Those communications can be specified as processes *SEND* and *SWITCH.*

$$\begin{aligned} |_{\forall i}SEND_i &= (i.p?(j,m)) \rightarrow \\ &\quad ((AU.p!(i.m) \rightarrow SKIP)||| \qquad (i \in \{SE, SY\}) \\ &\quad SWITCH) \end{aligned}$$

M_{SE} and M_{SY} use process *SEND* to communicate with M_{AU}, where *m* is an arbitrary string.

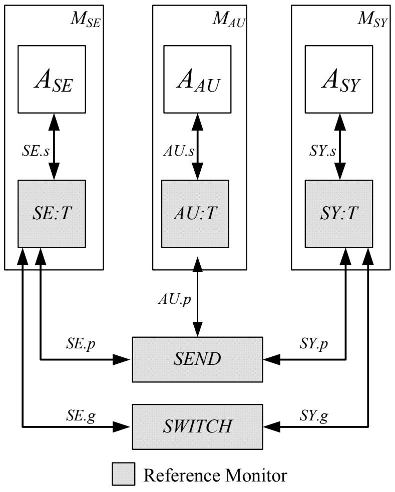

Fig. 2. Events of communication

$$\begin{aligned} SWITCH = {} & |_{\forall i}(i.g?(j,S,O,M)) \rightarrow \\ & ((j.g!(i.r) \rightarrow SKIP)||| \quad (i, \in \{SY, SE\}) \\ & SWITCH) \end{aligned}$$

M_{SE} and M_{SY} use process *SWITCH* to communicate, where *S* is subject sets and *O* is object set. *M* is a tabular data. This information as M_{SE}*can provide identification of* M_{SY}.

3.5 Cooperate Functioning

System manager, *security manager* and *audit manager* have to cooperate. for functioning An interleaving of all these processes specified as follows. Let *P* be the assembled process,

$$\begin{aligned} P = {} & M_{SE}|||M_{AU}|||M_{SY}||| \\ & A_{SE}|||A_{AU}|||A_{SY}||| \\ & (SEND||(SE:T|AT:T) \setminus \alpha SEND)||| \\ & (SE:T||SY:T) \setminus \alpha SWITCH) \end{aligned}$$

The direct evidence of internal state transitions will not be shown as the CSP hiding operator ("\") can hide the events in the alphabet.

4 Security Analysis and Implement

We will prove the three powers privilege separation mechanism can avoid the damage which happened in the monopolize power mechanism system.

Definition 1 (Secure manager state). The manager is secure if and only if:

$$\begin{gathered}\forall i \neq j \cdot tr, tr' \in traces(M_j) : \\ tr \upharpoonright \alpha M_i = tr' \upharpoonright \alpha M_i \Rightarrow \\ \mathcal{SF}[[M_j/tr]] \upharpoonright \alpha M_i = \mathcal{SF}[[M_j/tr']] \upharpoonright \alpha M_i\end{gathered}$$

This definition of equivalence follows the *stable failures model*. For a process *P*, the stable failure of *P*, [13] [14] written $\mathcal{SF}[[P]]$, is defined as:

$$\begin{gathered}\mathcal{SF}[[P]] = \{(s, X) \mid s \in traces(P) \wedge P/s \downarrow \wedge \\ X \in refusals(P/s)\}\end{gathered}$$

For each pair of traces, two experiments traces by M_j. If two resulting processes look equivalent from the managerM_j's perspective, than M_i is secure.

Definition 2 (Safe initial state). The initial state is safe if and only if:

$$\begin{gathered}\forall\ M_i \in M \cdot M_{i_0} = (b?m) \rightarrow M_i \ \wedge\ m \in TRUST \\ M = \{M_{SE}, M_{AU}, M_{SY}\}\end{gathered}$$

M_{i_0} is the start process of M_i. M_{i_0} uses channel *b* to listen on for its initial message *m*. Relied on Trusted Computing technology, the set *TRUST* can be fully trusted, any message picked from it is safe.

As the initial state of manager and the definition of a secure state of the manager have give, and the way in which the manager progresses from one state to another defined in section 3, than all future states of manager will be secure.

We implement this mechanism in Debian5.0 with LSM architecture. LSM provides a solution for security access control model in Linux. Based on operating system security mechanisms, our security management framework replaces original Linux hooks with loadable module in order to implement our security mechanism. Major security capability of the system meets the Structured-Protection criteria in [1] and [2].

5 Discussion

Although our privilege separation mechanism is safer than monopolize power, there is still much work to research. First, the formal prove of our mechanism has not be done in this paper, and we should do a machine-checkable proof (using FDR theorem proof checker in our future work. Second, conspiring situation has not been considered, which deserved to investigate.

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. Classified criteria for security protection of computer information system. GB17859-1999 (1999)
2. Trusted Computer System Evaluation Criteria (TCSEC), DoD (1985)

3. Saltzer, J., Schroeder, M.: The Protection of Information in Computer Systems. Proceedings of the IEEE 63(9), 1278–1308 (1975)
4. Clark, D.D., Wilson, D.R.: A Comparison of Commercial and Military Computer Security models. In: Proceedings 1987 Symposium on Security and Privacy. IEEE Computer Society, Oakland (1987)
5. Lee, T.M.P.: Using Mandatory Integrity to Enforce "Commercial Security". In: 1988 IEEE Symposium on Security and Privacy. IEEE Computer Society, Oakland (1988)
6. Shockley, W.R.: Implement Clark/Wilson Integrity Policy Using Current Technology. In: Proceedings 11th National Computer Security Conference (October 1988)
7. Qing, S.H., Shen, C.X.: Designing of High Security Level Operating System. Science in China Ser. E. Information Sciences 37(2) (2007)
8. Ji, Q.G., Qing, S.H., He, Y.P.: A New Privilege Control Formal Model Supporting POSIX. Science in China Ser. E. Information Sciences 34(6) (2004)
9. Sheng, Q.M., Qing, S.H., Li, L.P.: Design and Implementation of a Multi-Layered Privilege Control Mechanism. Journal of Computer Research and Development (3) (2006)
10. Bergstra, J.A., Klop, J.W.: Fixed Point Semantics in Process Algebras, Report IW 206. Mathematisch Centrum, Amsterdam (1982)
11. Hoare, C.A.R.: Communicating Sequential Processes. Prentice/Hall International, Englewood Cliffs (1985)
12. Krohn, M., Tromer, E.: Non-interference for a Practical DIFC-Based Operating System. In: 2009 IEEE Symposium on Security and Privacy. IEEE Computer Society, Oakland (2009)
13. Roscoe, A.W.: A Theory and Practice of Concurrency. Prentice Hall, London (1998)
14. Schneider, S.: Concurrent and Real-Time Systems: The CSP Approach. John Wiley & Sons, LTD., Chichester (2000)