

An Implementation of Extended-Role Based Access Control on an Embedded System

Shin, Wook†

Kim, Hong Kook†

Sakurai, Kouichi‡

†Department of Information and Communications,
Gwangju Institute of Science and Technology,
1 Oryong-dong Buk-gu, Gwangju, 500-712, Rep. of Korea

{sunihill, hongkook}@gist.ac.kr

‡Department of Computer Science and Communication Engineering,
Kyushu University, Hakozaki, Fukuoka 812-8581, Japan

sakurai@csce.kyushu-u.ac.jp

Abstract This paper addresses the complexity issues of extended-role based access control (E-RBAC) implemented under an embedded environment. Although E-RBAC can provide more trusted environment than the traditional trusted operating systems by prohibiting the attacks consisting of ordinary operations, it is expected that its implementation has performance overhead due to the procedural constraints of E-RBAC. An implementation of E-RBAC suggested in this paper reduces the overhead of E-RBAC, and it is also shown that the overhead is not significant compared to that of the previous Intrusion Detection System (IDS) solutions.

1 Introduction

Traditional UNIX compatible systems have been operated under Discretionary Access Control (DAC) policy. Although DAC was a flexible and general purpose scheme, insufficiencies in security have been pointed out [1]. Therefore, other security policies such as mandatory access control and role based access control have been adopted into the design of operating systems to establish more trusted computing base. Mandatory Access Control (MAC) was introduced and it provides a more concrete trusted environment by controlling the flow of information [2] - [4]. Role Based Access Control (RBAC) was adopted to provide a more flexible execution environments while the security information is administrated in a centralized manner [5], [6].

Although several access control policies have been introduced for the development of Trusted Operating Systems (TOS), the ability of the security kernel in TOS is still limited. Current security kernels cannot deny some attacks consisting of ordinary operations, which is due to

the manner of a decision making process in traditional access controls. In the current access control schemes, access control information is extracted from the access subject and object at the moment of an access, and its validity is lost after a decision. In the decision process, the legality of an access is decided based on the traditional access matrix information or the pre-defined relation between access subjects and access objects, while the associated information or hidden information between the operations is not considered at all.

The Extended Role Based Access Control (E-RBAC) was proposed to overcome the limitation and to extend the functionality of traditional access controls [7]. E-RBAC controls accesses based on associated access control information as well as the traditional access control information.

On the contrary to the traditional access controls, an E-RBAC system considers an additional information to make access decisions, thus it is expected that its implementation has more performance overhead than the implementations of the previous security kernels.

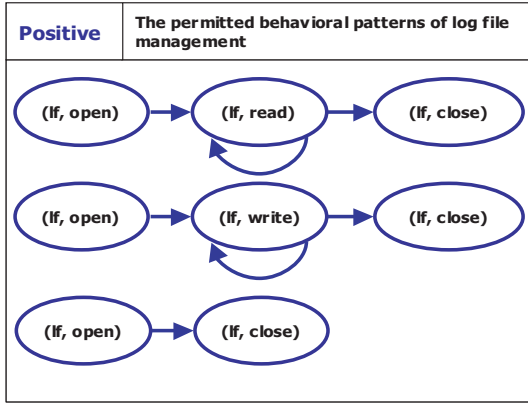


Figure 1: Examples of several operation sequences having a positive PC.

Therefore, in a designer viewpoint, it is necessary to make sure how much overheads are introduced in the implementation. In this paper, we implement a simple E-RBAC system under an embedded environment and evaluate its performance overhead.

This paper is organized as follows. In Sect. 2, we briefly review the E-RBAC concept and its model. In Sect. 3, the implementation architecture is discussed and the results of the performance evaluations are presented in Sect. 4. In Sect. 5, we conclude this paper.

2 The Extended Role Based Access Control (E-RBAC)

E-RBAC limits accesses based on the associated access control information. The information is represented as an ordered set of operations that are specified as procedural constraints (PC) in the E-RBAC model [7]. PC is defined as a partially ordered set of permissions with a positive or a negative value. A positive PC unit describes a permissible sequence of executions but a negative PC unit represents a dangerous one.

Fig. 1 shows examples of operation sequences having the positive PC. The positive PCs describe the permitted execution sequences related with log file management. As shown in Fig. 1, there are three allowed execution sequences: First, open a log file, read it several times, and close the file. Second, users can

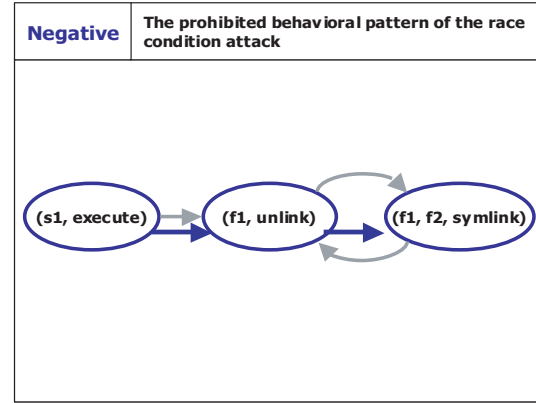


Figure 2: An example of an operation sequence having a negative PC.

write some data after opening the log file, and close it. Third, open the file, and then close the log file without any action. In this system, ‘read-write’ access is not permitted for the log file. It is reasonable because read-only and write-only operations are enough to record and inquire audit data. Moreover, some attacks erase the log record that includes attacker’s information. The attacks read and find the record of the log data, and then remove the record or replace it with the other information. Putting all accounts together, it is reasonable not to support ‘read-write’ operations. This kind of context can be a part of the security policy of the system, and the positive PCs support its enforcement effectively.

Fig. 2 shows an example of the negative PC. The negative PC models a race condition attack against a sendmail daemon [8]. The attack executes the sendmail program, and repeats linking and unlinking to redirect the binding between the sendmail program and a temporary file. The core execution sequence of the attack can be modeled as the state-transition diagram. E-RBAC detects and denies the attack if the execution sequence is specified.

E-RBAC considers procedural information as well as the traditional access information to judge the legality of each access so that the system prohibits system intrusions more effectively. An operation is denied if it accomplishes a negative execution procedure, or if it deviates from the defined positive sequences. E-RBAC has its formal model to specify an E-

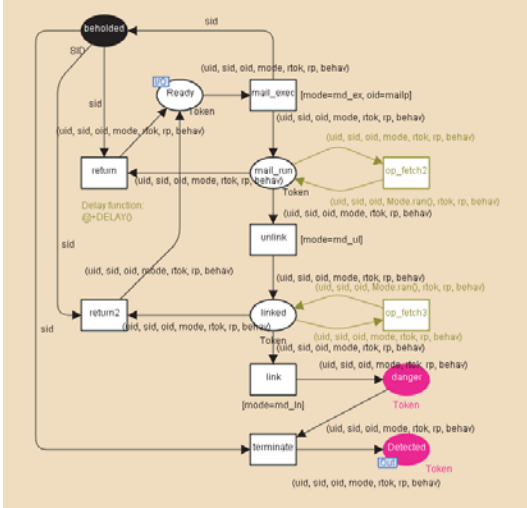


Figure 3: An example of a negative PC represented by CCPN.

RBAC system and to verify correctness of security configuration of the system. The Coloured Petri Net (CPN) based formal model, Constrained CPN (CCPN) was proposed and it specifies the E-RBAC system including the positive and negative PCs [7]. The specification can be verified with automated tools such as CPN Tools [9]. Fig. 3 shows an example of negative PC which is modeled with CCPN formalism.

3 Implementations

TOS has been implemented based on various kernels [2] - [6], [10]. There was no even a de facto standard for the implementation, although the core function of access control and the security policy of the implementations are not so different. However, there was a remarkable movement in the field of TOS based on monolithic Linux kernels. The Linux Security Module (LSM) [11] was developed to provide a bottle-neck interface for access controls. Various access control policies can be enforced by the policies are implemented as the Loadable Kernel Modules. LSM is accepted officially as one of the security mechanisms of the Linux kernel from the kernel version 2.6. Additionally, SELinux, the most representative security kernel based on Linux, adopted it as its core framework of access controls. Therefore, LSM

consolidates itself as the standard access control framework in Linux systems.

However, the LSM approach is not so reasonable for our implementation, because the LSM is too heavy mechanism to be ported into our tiny system. We implement our system on an embedded board, IFC-ETK100 [13], using the se3208 32 bit EISC processor [14]. The se3208 processor is a 16-bit processor actually, but it emulates 32-bit operations internally. Since there is no Memory Management Unit (MMU), the operating system is uClinux version 2.4.19 that can support the environment without MMU. The target embedded system has limitations in computational power and functions being compared to the Linux systems on desktop machines.

Many functions defined in LSM are not necessary in the embedded target. Some technologies in LSM are not supported in the system such as Loadable Kernel Module (LKM). Therefore, it can be an overhead to adopt the LSM architecture into such lightweight systems. Moreover, the LSM approach is not proven as the best or the most efficient solution for TOS development [12].

Therefore, we implement our access control structure by directly modifying kernel functions of the system instead of taking the LSM architecture. At first, we simply add the field of a permission vector to the data structures of process and files. The permission vectors are calculated in terms of permitted roles. The relation between processes, roles, behaviors, and permissions is also evaluated as E-RBAC model which is described in [7], and the set of allowed permissions for the roles is calculated finally. Access control decision functions (ADF) decide the legality of each access comparing the roles of a process and a file. They are implemented as kernel functions. Their functions are not so different from those of the other TOS implementations.

In addition, we add fields to trace the current state of the process in term of the CCPN model. Each process has its state information, and the state is changed by the execution of operations of the process. ADF calculates the next state according to the current state and

```

kr: /home/sunhill
drwxr-xr-x  2 0      0          12288 Sep  2 2004 lost+found
drwxr-xr-x  2 0      0          1024 Jul  8 2003 mnt
drwxr-xr-x  2 0      0           0 Apr 17 16:55 proc
drwxr-xr-x  2 0      0          1024 Jul  8 2003 root
drwxr-xr-x  2 0      0          1024 Jul  8 2003 sbin
drwxr-xr-x  2 0      0          1024 Apr 17 16:55 tmp
drwxr-xr-x  3 0      0          1024 Sep  2 2004 usr
drwxr-xr-x  5 0      0          1024 Jul  8 2003 var
/ > /bin/themis_forkattack
attack starts
fork to race!m the child
1. Exec sendmail
Sending mails...
To... johndoe@dummy.net
I'm the parent, child has pid 10
2. unlink
3. link
DEBUG> An attack is detected
-->attack finished
pid 9: failed 4096
/ > Messages: hello
[영어] [원성] [두블식]

```

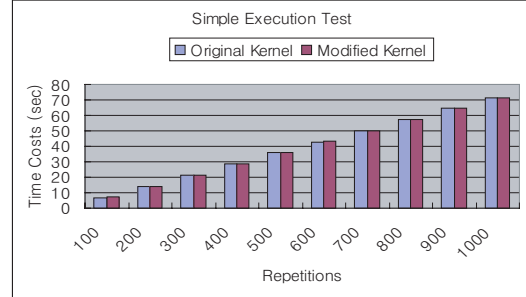
Figure 4: Examples of an attack execution and its detection.

the action that the process wants to execute. All processes iterate CCPN by executing operations. From the traces, we can investigate the execution sequence of operations of each access subject.

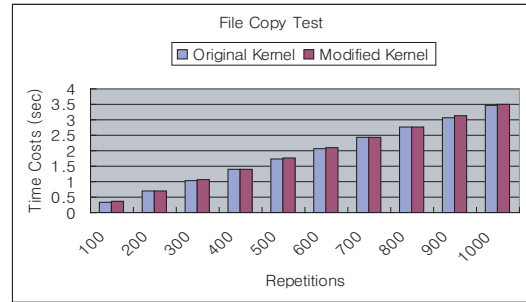
Fig. 4 shows an execution result of a race condition attack and its detection. The exploit program spawns two processes; One executes a sendmail process and the other repeats linking and unlinking to redirect the temporary file. The result shows that our implementation detects the attack successfully.

For the performance evaluation of our simple implementation, we will execute three programs. The first two testing programs are a simple execution program and a copy program, respectively. The simple execution program executes the program that prints a short sentence in the LCD panel of the embedded system. The copy program makes copies of a 512-byte file. No procedural constraint is applied to the two programs. The third program is a log file read program that is executed under a procedural constraint. The state-transition graph of the action consists of three sequential states.

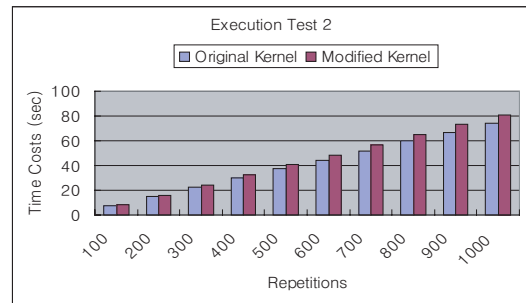
Fig. 5(a) shows the execution time of the simple execution program and Fig. 5(b) shows the results of the copy program. Each program is executed repeatedly from 100 times to 1000 times, and the execution time is measured by using the wait3() function. As shown in the figures, without procedural constraints, there is no significant overhead in the E-RBAC sys-



(a)



(b)



(c)

Figure 5: Comparison of the execution time between the original kernel and the modified kernel using E-RBAC from (a) a simple execution program, (b) a copy program, and (c) a log read program.

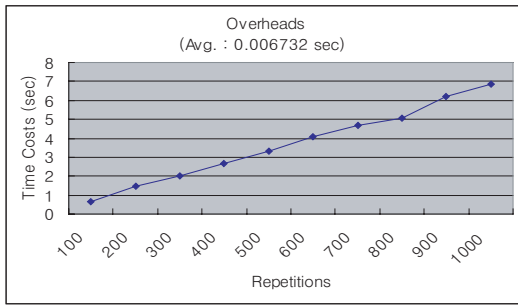


Figure 6: The overheads

tem compared to the original kernel.

On the other hand, Fig. 5 (c) shows the different execution times between original kernel and our implementation. With procedural constraints, there is an overhead caused from the state tracing. It is shown from Fig. 6 that the overhead linearly increases as the repetition increases and the average of the overhead is about 7 milliseconds which corresponds to 10 percent increase in overhead. The overhead is almost equal to or larger than the overheads of SELinux [10]. However, it is not larger than the overhead of an application level IDS solution, snort [15], when the overhead of operations in snort is measured in a firewall-applied environment.

4 Conclusions

In this paper, we introduced an E-RBAC implementation and discussed its performance evaluation results. The extended access control efficiently limits attack trials consisting of allowed operations. For example, the implementation detected the sendmail race condition attack successfully.

Also, the performance overhead was evaluated with three simple programs. When operations were executed without procedural constraints, there was no significant overhead. On the contrary, the performance overhead was observed with procedural constraints. However, the overhead was not larger than the current IDS solution. Considering that E-RBAC provides more fundamental security mechanisms which cannot be by-passed, this amount of overhead can be a bearable overhead. Moreover, usually in

lightweight systems such as our embedded environment, it is impossible to execute heavy programs such as IDS.

Acknowledgement

This research was partly supported by Joint Forum for Strategic Software Research (SSR) of International Information Science foundation. Also, it was partly supported by the University Research Program of the Ministry of Information and Communication, Republic of Korea.

References

- [1] Sandhu, R. and Samarati, P.: Access Control: Principles and Practice. IEEE Communications, Volume 32, Number 9, September 1994.
- [2] UNICOS Multilevel Security (MLS) Feature User's Guide. SG-2111 10.0, Cray Research, Inc. (1990)
- [3] Branstad, M., Tajalli, H., Mayer, F.: Security issues of the Trusted Mach system. Proc. of 4th Aerospace Computer Security Applications Conference (1998), 362-367
- [4] Flask: <http://www.cs.utah.edu/flux/fluke>
- [5] Ott, A.: The Rule Set Based Access Control (RSBAC) Linux Kernel Security Extension, 8th Int. Linux Kongress, Enschede (2001)
- [6] Trusted Solaris: <http://www.sun.com/software/solaris/trusted-solaris/index.html>
- [7] Shin, W., Lee, J.G., Kim, H.K., and Sakurai, K.: Procedural Constraints in the Extended RBAC and the Coloured Petri Net Modeling. IEICE Transactions on Fundamentals, Special Section on Cryptography and Information Security(to be issued), Vol.E88-A, No.1, Jan. 2005.
- [8] [8lgm]-Advisory-20.UNIX.SunOS-sendmailV5.1-Aug-1995.README

- [9] CPN Tools:
<http://wiki.daimi.au.dk/cpntools/>
- [10] Loscocco, P., Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System. Proc. of the FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX '01) (2001)
- [11] LSM: <http://lsm.immunix.org/>
- [12] grsecurity:
<http://www.grsecurity.net/lsm.php>
- [13] InterFC: <http://www.interfc.co.kr>
- [14] Advanced Digital Chips Inc.:
<http://www.adc.co.kr>
- [15] Snort: <http://www.snort.org>