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 We implement the Extended Role Based Access Control(E-RBAC) in an embedded environment. E-RBAC prohibits attacks which consist of ordinary operations. Although the implemented access control scheme provides more trusted environment, the performance overhead is not significant.

1 Introduction

Traditional UNIX compatible systems have been operated under Discretionary Access Control (DAC) policy. Although DAC was flexible and general purpose scheme, insufficiencies in security have been pointed out. Therefore other security policies have been adopted to establish more trusted computing bases. Mandatory Access Control (MAC) was introduced and provides more concrete trusted environment by controlling the flow of information [1], [2], [3]. Role Based Access Control (RBAC) was applied and supports more flexible execution environments while the security information is administrated in a centrailized manner [4], [5].

Although several access control policies have been introduced to the development of Trusted Operating Systems (TOS), the ability of the security kernels in TOS still has a kind of limitation. The limitation is security kernels cannot deny some attacks which consist of ordinary operations. It is due to the traditional access controls decide the legality of accesses based on instant access control information. Access control information is extracted from the access subejct and object at the moment of an access, and losts its validity after the decision. In the decision process, the association information between the operations is not considered at all.

The Extended Role Based Access Control (E-RBAC) was proposed to overcome the limitation and extend the functionality of traditional access controls [6]. E-RBAC controls accesses based on associated access control information as well as the traditional access control information.

On the other hand, it is expected a performance overhead exists, because E-RBAC system considers the additional information to make access decisions. It is necessary to confirm how much the overhead is generated. In this paper, we implement a simple E-RBAC system in an embedded environment and test the performace overhead.

This paper is organized as follows. In Sect. 2, we beriefly introduce the E-RBAC concept and its model. In Sect. 3, the implementation architecture is discussed and the results of the performance test are presented in Sect. 4. Sect. 5 is the conclusions.

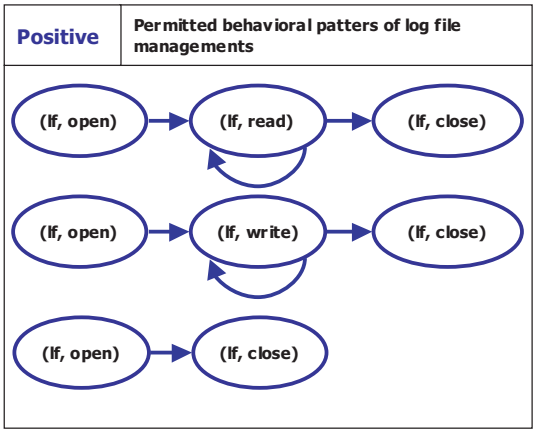

Figure 1: The example of a positive PC

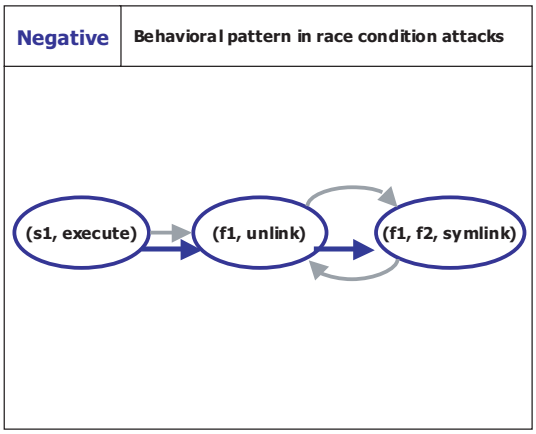

Figure 2: The example of a negative PC

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

E-RBAC limits accesses based on the associated access control information as well as the traditioanl access matrix information. The associated access control information represented as the ordered set of operations, and they are specified in E-RBAC model as Procedural constraints (PC) [6].

PC is the partially ordered set of permissions having the property of positive or negative value. A positive PC unit describes a permissible sequence of executions. A negative PC unit represents a dangerous sequence of executions.

Fig. 1 shows an example of positive PC. The positive PC describes the permitted execution sequences in a log file manipulations. As the state diagram presented, there are three allowed execution sequences: First, open the log file, read it several times, and close it. Second, users can write some data after open the log file, and close it. Third, open and close the log file with nothing. In other words, for the log file, 'read and write' access is not permitted. In the system, it is assumed that read only or write only operation is enough to provide. Considering some attacks remove or replace the log record which includes attacker's information after they read and find the record of the log data, it is reasonable to not support the 'read and write' operation on log files. The positive PC describes the security policy.

Fig. 2 shows an example of negative PC. The negative PC models a race condition attack to a sendmail daemon [7]. The attack executes the sendmail program, and repeats linking and unlinking for a redirection of the object binding. The core execution sequence of the attack can be modeled as the figure, and E-RBAC can deny the attack if it is specified in its access control model.

E-RBAC system checks the legality of an operation more effectively against various system threats by considering the procedural information as well as the traditional access matrix information, An execution of an operation will be denied if it accomplishes a negative execution procedure, or if it deviates from the defined positive sequences.

E-RBAC has its formal model for the its specification and verification. The CCPN formal model was proposed for E-RBAC based on CPN formalism [6]. It can specify E-RBAC system including the positive and negative PCs, and the specification can be verified with authomated tools such as CPN Tools [8] Fig. 3 shows an example of negative PC which is modeled with CCPN formalism.

3 Implementations

Currently, several implemenatations of Trusted Operating System have been exist [1], [2], [3]. [4], [5], [9]. They are implemented based on various types of kernels.

On the other hand, there was a remarkable movement in the field of researches based on monolithic Linux kernels. SELinux, one of

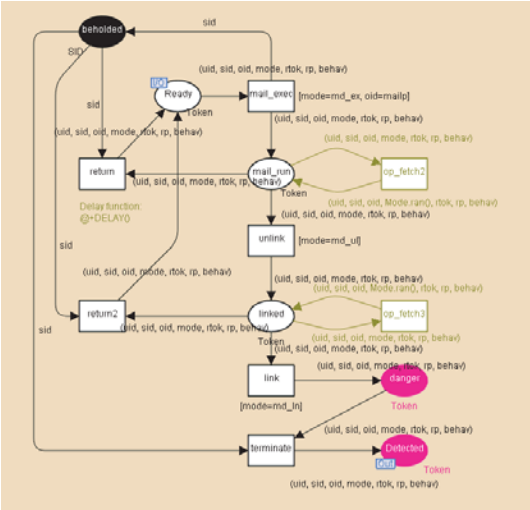

Figure 3: An example of CCPN of a negative PC

the most representitive security kernels based on Linux, adopted the Linux Security Module (LSM) [10] as their core framework of access controls. Also, LSM is accepted officially as one of the security mechanisms of Linux kernel from the kernel version 2.6 officially. To sum up, the current trend of development of TOS is using LSM modules. It can be a good news for the companies because it is need to establish a kind of standard in TOS deveelopment.

However, the approach using LSM was not reasonable for our implementation. In our implementation, the target system is an embedded system, IFC-ETK100 [12] using se3208 32bit EISC processo r[13]. The operating system is uClinux version 2.4.19. The embedded systems usually have a kind of limitations in computational power and functions being compared to Linux operating systems and desktop machines. Therefore it needs not bunch of access control functions defined in LSM, and it can be an overhead to adopt the LSM architecture into light-weight systems. Many embedded systems does not support for loadable modules, it is unnecessary that the flexible LSM architecture also. Moreover, LSM approach is not proven as the best or efficient solution for TOS development [11].

Therefore, we implement our access control structure directly modifying kernel functions

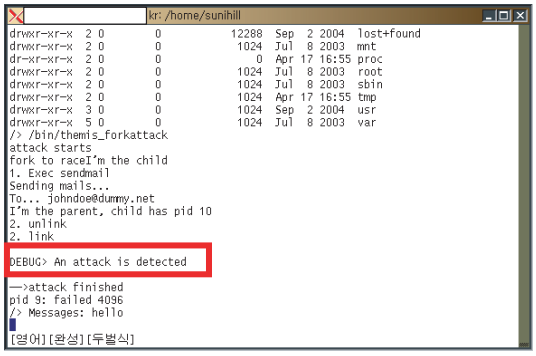

Figure 4: Examples of an attack execution and its detection

of the system instead of taking LSM architecture. At first, we simply added the field of permission vector to the data structures of process and files. The permission vectors are calculated in terms of permitted roles. The relation between process, roles, behaviors, and permissions are considered as E-RBAC model describes [6], and finally the set of allowed permissions for a process are calculated. Access control decision functions (ADF) are decide the legality of each accesses comparing the sets of roles between process and files, and it is implemented as a kernel function. The framework provides access controls based on traditional access matrix information.

For the investigating the execution sequences of operations, we also simply added a field to the data structure of processes, and the value of the field is the current state in CCPN. ADF calculates the next state based on the current states of each processes and actions which the processes want to execute. Therefore, all processes itinerate CCPN by executing operations.

Fig. 4 shows an execution result of a race condition attack and its detection. The attack spawns two processes of executing a sendmail process and linking and unlinking repeatedly. The E-RBAC system detects the attack successfully.

The main reason of the implementation is the test of overhead. The testing program was a simple execution program and a copy program. The simple execution program executes the other program which prints a short sen-

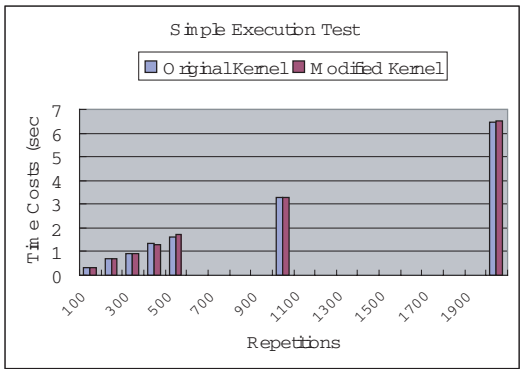

Figure 5: The execution time of the simple execution program

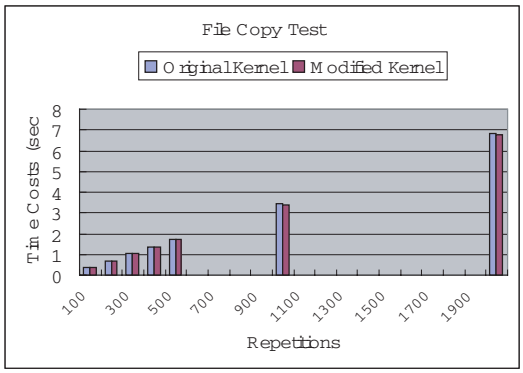

Figure 6: The execution time of the copy program

tence in the LCD panel of the embedded system. The copy program make a copy of a 512 byte file. Considering the access enforcement functions intervene all of file operations, it is reasonable to measure the execution time of the two programs.

Fig. 5 shows the execution time of the simple execution program and the Fig. 5 shows the results of the copy program. Each program is executed repeatedly from 100 times to 2000 times, and the execution times are measured. As the figures show, there is no significant overhead in the E-RBAC system.

4 Conclusions

In this paper, we introduced the E-RBAC concept and its implementation. The extended access control efficiently limits attack trials which consist of allowed operations. The implementation result of the access control in an embedded system showed an example case of the detection of the sendmail race condition attack.

Also, the performance overhead is tested with two simple file manipulation examples. The test result showed that there is no significant overhead in E-RBAC system.

Acknowlegement

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

References

[1] UNICOS Multilevel Security (MLS) Feature User's Guide. SG-2111 10.0, Cray Research, Inc. (1990)

[2] Branstad, M., Tajalli, H., Mayer, F.: Security issues of the Trusted Mach system. Proc. of 4th Aerospace Computer Security Applications Conference (1998), 362–367

[3] Flask: http://www.cs.utah.edu/flux/fluke

[4] Ott, A.: The Rule Set Based Access Control (RSBAC) Linux Kernel Security Extension. 8th Int. Linux Kongress, Enschede (2001)

[5] Trusted Solaris: http://wwws.sun.com/software/solaris/trustedsolaris/index.html

[6] 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.

[7] [8lgm]-Advisory-20.UNIX.SunOS-sendmailV5.1-Aug-1995.README

[8] CPN Tools: http://wiki.daimi.au.dk/cpntools/

[9] 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)

[10] LSM: http://lsm.immunix.org/

[11] gresecurity: http://www.grsecurity.net/lsm.php

[12] InterFC: http://www.interfc.co.kr

[13] Advanced Digital Chips Inc.: http://www.adc.co.kr