บทที่ 5
กลไกการตรวจจับการบุกรุกของระบบปฏิบัติการ

5.1 บทนำ

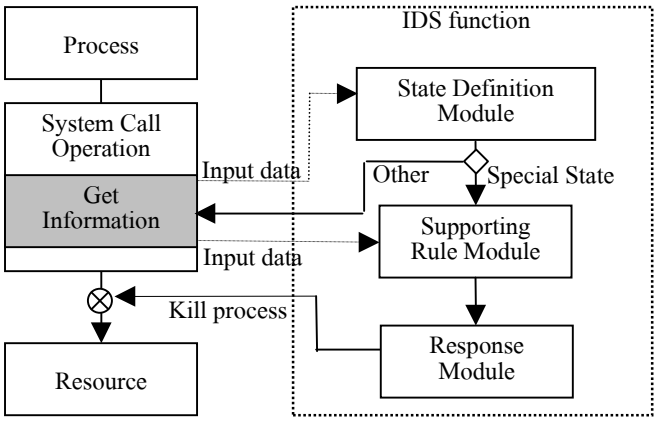
จากการทดสอบผลการวิเคราะห์กฎสนับสนุนโดยพัฒนาโปรแกรมตรวจจับการบุกรุกในบทที่ 4 ได้ข้อสรุปว่าโปรแกรมตรวจจับการบุกรุกสามารถตรวจจับเหตุการณ์ต่างๆ ที่ขัดต่อกฎสนับสนุน แต่โปรแกรมดังกล่าวยังมีข้อจำกัดบางประการเช่น ข้อมูลที่ใช้สำหรับการพิจารณาเหตุการณ์นั้นจะเกิดขึ้นหลังจากที่การบุกรุกเกิดขึ้นเรียบร้อยแล้วอีกทั้งข้อมูลที่เกิดขึ้นมีจำนวนมากจนโปรแกรมตรวจจับการบุกรุกไม่สามารถจัดการข้อมูลเหล่านั้นได้ทั้งหมดซึ่งเป็นผลให้โปรแกรมตรวจจับการบุกรุกหยุดการทำงาน

ในบทนี้จึงกล่าวถึงการพัฒนากลไกการตรวจจับการบุกรุกของระบบปฏิบัติการเพื่อปรับปรุงประสิทธิภาพของการทำงานของระบบปฏิบัติการ โดยการแก้ไขซิซเท็มคอล ในลำดับถัดไปจะถึงสถาปัตยกรรมของกลไกการตรวจจับการบุกรุกของระบบปฏิบัติการ และวิธีการแก้ไขซิซเท็มคอล

5.2 สถาปัตยกรรมของกลไกตรวจจับการบุกรุก

สถาปัตยกรรมของกลไกตรวจจับการบุกรุกปรับปรุงจากสถาปัตยกรรมของโปรแกรมตรวจจับการบุกรุกในบทก่อนหน้า การตรวจจับการบุกรุกทั้งสองสถาปัตยกรรมมีขั้นตอนวิธีที่คล้ายคลึงกันแต่วิธีการอ่านข้อมูลนำเข้าแตกต่างกัน กลไกการตรวจจับการบุกรุกในระบบปฏิบัติการสามารถอ่านข้อมูลนำเข้าจากซิซเท็มคอลเป้าหมายได้โดยตรง ในขณะที่โปรแกรมตรวจจับการบุกรุกเดิมอ่านข้อมูลจากซิซเท็มคอล ktrace()

เมื่อซิซเท็มคอลที่ได้รับการแก้ไขถูกเรียกใช้ กลไกการตรวจสอบจะเริ่มทำงานก่อนที่จะเข้าสู่การทำงานของซิซเท็มคอล กลไกตรวจจับการบุกรุกจะนิยามสถานะของโปรเซส ถ้าหากโปรเซสดังกล่าวอยู่ในสถานะพิเศษแล้วจะถูกส่งเข้าสู่โมดูลพิจารณาตามกฎสนับสนุนต่อไป แต่ถ้าหากเป็นสถานะอื่นแล้วโปรเซสเป้าหมายสามารถเข้าใช้ทรัพยากรตามปกติ ในกรณีที่พิจารณาแล้วว่าโปรเซสเป้าหมายเป็นโปรเซสบุกรุก โปรเซสดังกล่าวพร้อมด้วยโปรเซสอื่นๆ ที่มีค่า UID เท่ากับโปรเซสบุกรุกจะถูกทำลายและบันทึกเหตุการณ์ไว้ในล็อกของระบบ กระบวนการทำงานที่กล่าวมาทั้งหมดนี้แสดงไว้ในภาพประกอบที่ 5.1



ภาพประกอบ 5.1 สถาปัตยกรรมของกลไกการตรวจจับการบุกรุกของระบบปฏิบัติการ

5.3 การแก้ไขซิซเท็มคอลของระบบปฏิบัติการ

ในหัวข้อนี้กล่าวถึงการแก้ไขซิซเท็มคอลที่เกี่ยวข้องกับความปลอดภัยซึ่งได้จากการวิเคราะห์กฎสนับสนุนในบทที่ 3 โดยเพิ่มกลไกการตรวจจับการบุกรุกเข้าในซิซเท็มคอล ก่อนที่จะแก้ไขระบบนั้นจะกล่าวถึงโครงสร้างของซิซเท็มคอลและวิธีการอ่านข้อมูลนำเข้า กลไกการตรวจจับการบุกรุกทำงานใน kernel space ดังนั้นการตรวจจับการบุกรุกจึงสามารถเข้าถึงโครงสร้างข้อมูลได้ทุกโครงสร้าง หลังจากนั้นกล่าวถึงโมดูลต่างๆ ที่เกี่ยวข้องกับการตรวจจับการบุกรุกและวิธีการแก้ไขระบบปฏิบัติการ

5.3.1 โครงสร้างของซิซเท็มคอลในระบบปฏิบัติการเน็ทบีเอสดี

โครงสร้างของของระบบปฏิบัติการเน็ทบีเอสดีในส่วนของซิซเท็มคอลประกอบไปด้วย 3 ส่วนหลักได้แก่ ส่วนโพรโตไทป์ ส่วนของการประกาศตัวแปรและส่วนของการดำเนินการของซิซเท็มคอล โดยแต่ละส่วนมีรายละเอียดดังนี้

- **ส่วนของโปรโตไทป์** เป็นส่วนที่กำหนดรายละเอียดของซิซเท็มคอล รายละเอียดส่วนนี้จะเหมือนกันทุกซิซเท็มคอล ประกอบไปด้วย โปรเซสที่เรียกใช้ พารามิเตอร์ที่ส่งเข้าซิซเท็มคอล ค่าคืนกลับของซิซเท็มคอล
- **ส่วนของการกำหนดตัวแปร** เป็นส่วนของการประกาศและกำหนดค่าตัวแปรที่ใช้ในการดำเนินงานของซิซเท็มคอล

- **ส่วนของการดำเนินงานของซิซเท็มคอล** เป็นชุดคำสั่งสำหรับดำเนินงานตามหน้าที่ของซิซเท็มคอลแต่ละตัว

จากตัวอย่างของโครงสร้างของซิซเท็มคอลที่แสดงต่อไป เป็นโครงสร้างของซิซเท็มคอล open() เมื่อพิจารณาตามส่วนประกอบที่กล่าวไปแล้วข้างต้นอธิบายได้ว่า

```
1.      int sys_open(l, v, retval)
2.      struct lwp *l;
3.      void *v;
4.      register_t *retval;
5.      {
6.              struct sys_open_args /* {
7.                      syscallarg(const char *) path;
8.                      syscallarg(int) flags;
9.                      syscallarg(int) mode;
10.             } */ *uap = v;
11.     struct proc *p = l->l_proc;
12.     struct cwdinfo *cwdi = p->p_cwdi;
13.     struct filedesc *fdp = p->p_fd;
        //////// CUT////////
14.     flags = FFLAGS(SCARG(uap, flags));
15.     if ((flags & (FREAD | FWRITE)) == 0)
16.             return (EINVAL);
        //////// CUT////////
17.     *retval = indx;
18.     return (0);
19.     }
```

1 2 3

ส่วนที่ 1 (บรรทัดที่ 1-10) เป็นส่วนของโพรโตไทป์ เป็นรายละเอียดของซิซเท็มคอลได้แก่ struct lwp *p เป็นโครงสร้างข้อมูลที่ใช้เก็บข้อมูลสำคัญของโปรเซส void *v เป็นตัวที่เก็บค่าพารามิเตอร์สำหรับส่งข้อมูลเข้าซิซเท็มคอล ในแต่ละซิซเท็มคอลจะปรับค่าตัวแปรดังกล่าว

ให้เป็นโครงสร้างข้อมูลที่เฉพาะเจาะจงกับซิซเท็มคอล จากตัวอย่างตัวแปร *v จะถูกปรับเป็นโครงสร้างข้อมูล struct sys_open_args ท้ายที่สุด retval เป็นตัวแปร ใช้สำหรับเก็บค่าคืนกลับของซิซเท็มคอล

ส่วนที่ 2 (บรรทัดที่ 11-13) เป็นส่วนของการประกาศตัวแปรสำหรับการใช้งานภายในซิซเท็มคอลรายละเอียดดังกล่าวขึ้นอยู่กับชนิดของซิซเท็มคอล

ส่วนที่ 3 (บรรทัดที่ 14-19) เป็นส่วนของการดำเนินงานของซิซเท็มคอลที่จำเป็นต้องแก้ไขเพื่อโดยเพิ่มกลไกของการตรวจจับการบุกรุกลงไปในซิซเท็มคอล

5.3.2 การอ่านข้อมูลนำเข้าสำหรับกลไกการตรวจจับการบุกรุก

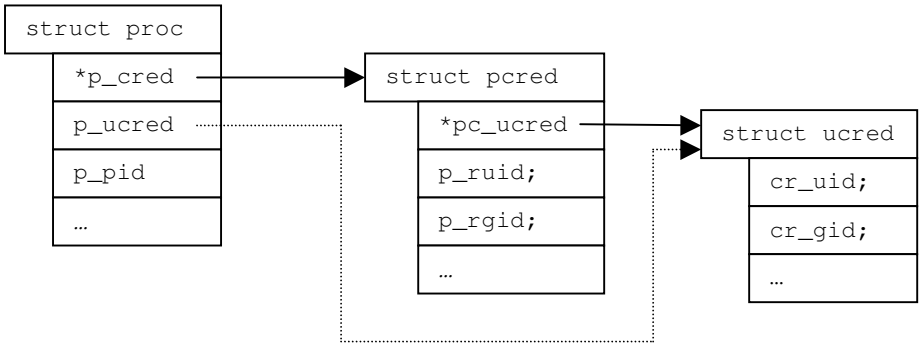
ข้อมูลนำเข้าของกลไกการตรวจจับการบุกรุกเป็นข้อมูลชุดเดียวกับข้อมูลที่ใช้ในโปรแกรมตรวจจับการบุกรุก แต่แตกต่างกันที่วิธีการอ่านข้อมูล กลไกชุดนี้จะอ่านข้อมูลมาจากตัวแปร *p และ *v ซึ่งแสดงอยู่ในส่วนที่ 2 ของรหัสต้นฉบับของซิซเท็มคอล สำหรับวิธีการอ่านค่าที่ต้องการนั้นแสดงไว้ในตารางที่ 5.1

ตารางที่ 5.1 แสดงชื่อข้อมูลและวิธีการอ่านข้อมูลนำเข้า

ตัวแปร	ความหมาย	ชนิดข้อมูล	วิธีการอ่านค่า
UID	User ID	uid_t	p->p_cred->p_ruid
GID	Group ID	uid_t	p->p_cred->p_rgid
EUID	Effective User ID	gid_t	p->p_ucred->cr_uid
EGID	Effective Group ID	gid_t	p->p_ucred->cr_gid
PID	Process ID	pid_t	p->p_pid
Parameter	Parameter	-	SCARG(v, argv1)

จากตารางที่ 5.1 แสดงชื่อและวิธีการอ่านข้อมูลนำเข้าประกอบด้วยค่า user credential ซึ่งอ่านจากโครงสร้างข้อมูล struct proc *p โครงสร้างข้อมูลดังกล่าวมีโครงสร้างข้อมูลย่อยชื่อ struct pcred และ struct ucred ดังภาพประกอบที่ 5.2 โครงสร้างข้อมูล struct porc จัดเก็บข้อมูลเกี่ยวกับรายละเอียดของโปรเซส เช่น หมายเลขโปรเซส (p_pid) ค่า user credential ของโปรเซสโดยอ้างผ่านสมาชิกชื่อ *p_cred โดยมีข้อมูลแบบโครงสร้างของ struct pcred โครงสร้างดังกล่าวมีสมาชิกที่สำคัญ 3 ตัวคือค่า UID และ GID โดยอ่านข้อมูลมาจาก p_ruid และ p_rgid ตาม

ลำดับ สำหรับสมาชิกอีกตัวคือ *pc_ucred เป็นตัวแปรแบบโครงสร้างซึ่งจัดเก็บค่า EUID และ EGID ของโปรเซส แต่โปรเซสสามารถอ้างถึงข้อมูลทั้งสองตัวนี้ได้จากสมาชิกชื่อ *p_ucred ของโครงสร้าง struct proc เช่น ถ้าหากต้องการอ้างถึงค่า EUID ของโปรเซสผ่านตัวแปรชื่อ *p อ้างได้ว่า p->p_ucred->cr_uid แทนการอ้าง p->p_cred->pc_ucred->cr_uid เป็นต้น



ภาพประกอบ 5.2 โครงสร้างข้อมูล struct proc, struct pcred และ struct ucred

สำหรับค่าพารามิเตอร์ของแต่ละซิซเท็มคอลจะอ่านจากตัวแปร *v ซึ่งถูกปรับค่าเป็นชนิดของข้อมูลตามโครงสร้างข้อมูลของซิซเท็มคอลเป้าหมาย เช่น ซิซเท็มคอล open*("/tmp/x", O_CREAT, S_IRWX)* เมื่อพิจารณาจากตัวอย่างข้างต้นโครงสร้างข้อมูลของ *v ประกอบไปด้วย

```
struct sys_open_args /* {
        syscallarg(const char *) path;
        syscallarg(int) flags;
        syscallarg(int) mode;
} */ *uap = v;
```

เมื่ออ่านค่าพารามิเตอร์ ของตัวแปร *v ซึ่งกำหนดให้ตัวแปร uap ด้วยมาโคร SCARG() ผลที่ได้คือค่าพารามิเตอร์ของซิซเท็มคอล open() ตามตารางที่ 5.2

ตาราง ที่ 5.2 แสดงการอ่านค่าพารามิเตอร์ของซิซเท็มคอล open()

พารามิเตอร์	ความหมาย	ค่าของพารามิเตอร์
SCARG(uap, path)	ชื่อแฟ้มที่ต้องการเปิด	"/tmp/x"
SCARG(uap, flags)	ค่า flags	O_CREAT
SCARG(uap, mode)	ค่า mode	S_IRWX

5.3.3 โมดูลสำคัญที่เกี่ยวข้องกับกลไกการตรวจจับการบุกรุก

แต่ละซิซเท็มคอลจะประกอบไปด้วยกลไกการตรวจจับการบุกรุก 3 โมดูลได้แก่ โมดูลนิยามสถานะ โมดูลพิจารณาตามกฎสนับสนุน และโมดูลตอบสนอง สำหรับโมดูลนิยามสถานะและโมดูลตอบสนองมีวิธีการทำงานเหมือนกันทุกซิซเท็มคอล แต่การทำงานของโมดูลของการพิจารณากฎสนับสนุนจะแตกต่างกัน ในหัวข้อนี้กล่าวถึงเฉพาะโมดูลนิยามสถานะและโมดูลตอบสนองเท่านั้น สำหรับโมดูลพิจารณากฎสนับสนุนนั้นจะกล่าวในหัวข้อถัดไป

โมดูลนิยามสถานะ

โมดูลนิยามสถานะอ่านค่า user credential จากโครงสร้างข้อมูล struct proc หลังจากนั้นนิยามสถานะตามอัลกอริทึม โมดูลนี้จะคืนค่าสถานะของโปรเซสให้แก่ซิซเท็มคอลที่เรียกใช้ ขั้นตอนการทำงานของโมดูลนิยามสถานะมีดังนี้

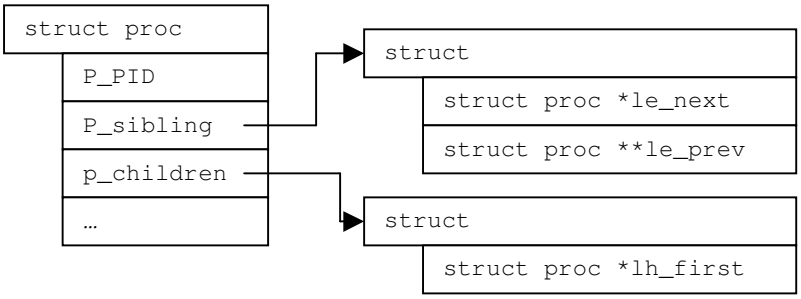
```
int state(p)
struct proc *p;
{
        int uid = p->p_cred->p_ruid;
        int gid = p->p_cred->p_rgid;
        int euid = p->p_ucred->cr_uid;
        int egid = p->p_ucred->cr_gid;
        int st = 0x0;

        if (!uid && !euid && !gid  !egid) return NORMAL;
        if (!gid && !egid )st |= SYSTEM;
        if (!uid && !euid )st |= SUPERUSER;
        if (st) return st;
        st = 0x0;
```

```
        if (!euid ) st |= SETUID;
        if (!egid ) st |= SETGID;
        if (!uid )  st |= SETREUID;
        if (!gid )  st |= SETREGID;
        return st;
}
```

โมดูลตอบสนอง

โมดูลนี้มีหน้าที่บันทึกเหตุการณ์และทำลายโปรเซสที่ละเมิดกฎสนับสนุน ซึ่งมีขั้นตอนวิธีของการทำลายโปรเซสแบบเรียกตัวเอง เนื่องจากโครงสร้างความสัมพันธ์ของโปรเซสในระบบปฏิบัติการยูนิกซ์เป็นแบบโครงสร้างแบบต้นไม้ (tree structure) โดยเชื่อมโปรเซสแต่ละโปรเซสด้วยสมาชิก 2 ตัว คือ **p->p_sibling** มีหน้าที่สำหรับเก็บรายการของโปรเซสพี่น้อง และ **p->p_children** มีหน้าที่สำหรับเก็บรายการของโปรเซสลูก ดังแสดงไว้ในภาพประกอบที่ 5.3



ภาพประกอบ 5.3 โครงสร้างแสดงความสัมพันธ์ของโปรเซส

จากภาพประกอบที่ 5.3 แสดงความสัมพันธ์ของโปรเซสเป็นแบบต้นไม้ เมื่อพิจารณาโปรเซสบุกรุกเป็นโปรเซสหลักแล้ว โปรเซสพี่น้องของโปรเซสบุกรุกอ้างถึงโดยโครงสร้างลิงค์ลิสต์แบบสองทาง (double link list) ด้วยสมาชิกชื่อ p_sibling->le_next และ p_sibling->le_prev ในขณะที่โปรเซสลูกของโปรเซสบุกรุกอ้างถึงด้วยโครงสร้างลิงค์ลิสต์แบบทางเดียว (single link list) ด้วยสมาชิกชื่อ p_children->lh_first

การทำลายโปรเซสที่มีความสัมพันธ์บุกรุกจะเป็นการเข้าถึงสมาชิกซึ่งหมายถึงโปรเซสของโครงสร้างข้อมูลด้วยอัลกอริทึมของ tree transversal แบบ depth-first-search เพื่อค้นหาโปรเซสที่มีเจ้าของคนเดียวหรือมีค่า UID เท่ากันกับโปรเซสบุกรุก ซึ่งเริ่มต้นที่โปรเซส init (PID 1) ตามอัลกอริทึมต่อไปนี้

```
KillGroup(PROC1)
        # when PROC1 and PROC2 is Process tabel
        If PROC1 is null then return
        PROC2 is child process of PROC1
        do
                KillGroup(PROC2)
                If Owner of PROC2 is not ROOT then kill it
        while PROC2 has Sibling Process
```

5.3.4 การแก้ไขซิซเท็มคอลที่เกี่ยวข้องกับความปลอดภัย

จากการวิเคราะห์กฎสนับสนุนและทดสอบแนวคิดดังกล่าวโดยการพัฒนาโปรแกรมตรวจจับการบุกรุกจึงได้ซิซเท็มคอลที่จะต้องแก้ไขเพื่อเพิ่มประสิทธิภาพทางด้านความปลอดภัยของระบบปฏิบัติการ หลังจากที่ศึกษาโครงสร้างของซิซเท็มคอลเหล่านั้นแล้วจึงจัดกลุ่มของซิซเท็มคอลตามวิธีการแก้ไขได้โดยแสดงไว้ในตารางที่ 5.3

ตารางที่ 5.3 แสดงกลุ่มของซิซเท็มคอลซึ่งจัดจำแนกตามวิธีการแก้ไข

กลุ่ม	ซิซเท็มคอล
I	setuid() setgid() setreuid() setregid()
II	chmod() fchmod() lchmod() open()
III	execve()
IV	mount() unmount() reboot() quotactl() settimeofday() swapon() และ nfssvc()

จากตารางที่ 5.3 ซิซเท็มคอลในกลุ่ม I เป็นซิซเท็มคอลที่ใช้สำหรับการเปลี่ยนแปลงค่า user credential ของโปรเซส ซิซเท็มคอลในกลุ่ม II เป็นซิซเท็มคอลที่ใช้สำหรับสร้างโปรแกรมแบบ setuid นอกจากนี้ซิซเท็มคอล open() ยังเป็นช่องทางให้ผู้บุกรุกสามารถแก้ไขโปรแกรมระบบหรือฐานข้อมูลผู้ใช้ ซิซเท็มคอลในกลุ่ม III ใช้สำหรับสั่งงานโปรเซสอื่นและถ่ายทอดสิทธิพิเศษให้แก่โปรเซสใหม่ถ้าโปรเซสที่สั่งงานเป็นโปรเซสแบบ setuid ซิซเท็มคอลกลุ่ม IV ช่วยสนับสนุนให้การบุกรุกประสบความสำเร็จหรือใช้สำหรับกลบเกลื่อนร่องรอยของการบุกรุก ซิซเท็มคอลแต่ละกลุ่มมีรายละเอียดของการแก้ไขดังนี้

ซิซเท็มคอลที่ใช้สำหรับเปลี่ยนแปลงค่า user credential ได้แก่ซิซเท็มคอล setuid (), setgid(), seteuid() และ setegid() ซิซเท็มคอลในกลุ่มนี้จะเรียกใช้ฟังก์ชัน do_setresuid() หรือ do_setresgid() เพื่อเปลี่ยนแปลงค่า user credential จากการวิเคราะห์กฎสนับสนุนพบว่า ถ้าหากมี

การเปลี่ยนแปลงค่า user credential แล้วเป็นผลให้โปรเซสอยู่ในสถานะผู้ใช้สูงสุดหรือกลุ่มระบบ ถือว่าโปรเซสดังกล่าวเป็นโปรเซสบุกรุก ดังนั้นสำหรับซิซเท็มคอลในกลุ่มนี้ ให้เพิ่มกระบวนการตรวจจับการบุกรุกไว้ตอนท้ายของฟังก์ชัน do_setresuid() และ do_setresgid() ดังนี้

```
do_setresuid(l, p, retval){
        If state(PROC) is SYSTEM_GROUP or SUPER_USER_GROUP then
                response(PROC)
        return
}
```

ซิซเท็มคอลสำหรับการสั่งงานโปรเซส ได้แก่ซิซเท็มคอล execve() ซึ่งจะถูกตรวจสอบสถานะและค่าพารามิเตอร์ได้แก่ ชื่อโปรเซสที่จะเรียกใช้ ถ้าหากโปรเซสดังกล่าวไม่อยู่ในกลุ่มของโปรเซสที่น่าเชื่อถือแล้ว ถือว่าโปรเซสที่เรียกใช้ซิซเท็มคอลเป็นโปรเซสบุกรุก

```
sys_execve(l, p, retval){
        /* ส่วนของการประกาศตัวแปร*/
        if state(PROC) is Special State then
                if PROG is not trust program then
                        response(PROC)
                end if
        end if
        /* ส่วนของการดำเนินงานของซิซเท็มคอล */
}
```

ซิซเท็มคอลวิกฤต ได้แก่ซิซเท็มคอล settimeofday(), reboot(), mount(), unmount (), quotactl() และ nfssvc() ซิซเท็มคอลในกลุ่มนี้ได้เพิ่มกระบวนการนิยามสถานะไว้ส่วนต้นของซิซเท็มคอลแต่ละตัว ถ้าหากโปรเซสที่เรียกใช้ซิซเท็มคอลอยู่ในสถานะพิเศษถือว่าโปรเซสดังกล่าวเป็นโปรเซสบุกรุก (sys_xxxx หมายถึงชื่อซิซเท็มคอลแต่ละตัว เช่น sys_reboot แทนซิซเท็มคอล reboot())

```
sys_xxxx(l, p, retval){
        /* ส่วนของการประกาศตัวแปร*/
        if state(PROC) is Special State then
                        response(PROC)
        end if
        /* ส่วนของการดำเนินงานของซิซเท็มคอล */
}
```

ซิซเท็มคอลในกลุ่มของการจัดการแฟ้ม แบ่งออกเป็นสองกลุ่มย่อยคือการแก้ไขซิซเท็มคอล open() และการแก้ไขซิซเท็มคอลในกลุ่มของ chmod() สำหรับซิซเท็มคอลในกลุ่มของ chmod() มีวิธีการแก้ไขเหมือนกันเนื่องจากซิซเท็มคอลทั้งสามนั้นเรียกใช้ฟังก์ชัน change_mode() เพื่อเปลี่ยนแปลงค่า permission ซึ่งแก้ไขตามอัลกอริทึมต่อไปนี้

```
int change_mode()
        /* ส่วนของการประกาศตัวแปร*/
        if state(PROC) is Special State then
                if mode is Setuid/Setgid then
                        response(PROC)
                end if
        end if
        /* ส่วนของการดำเนินงานของซิซเท็มคอล */
```

สำหรับซิซเท็มคอล open() ซึ่งเป็นซิซเท็มคอลที่ผลกระทบต่อการทำงานของระบบปฏิบัติการมากที่สุด เนื่องจากกิจกรรมของระบบปฏิบัติการทุกกิจกรรมเกี่ยวข้องกับแฟ้ม อีกทั้งซิซเท็มคอลนี้มีความเกี่ยวข้องกับกฎสนับสนุนหลายข้อ การตรวจสอบค่าพารามิเตอร์จึงซับซ้อนมากกว่าซิซเท็มคอลอื่นๆ

```
sys_open(l, p, retval){
        /* ส่วนของการประกาศตัวแปร*/
        If state(PROC) is Special State then
                If FLAG is Modify then
                        If PATH is User Database then response(PROC)
                        If PATH is System Command then response(PROC)
                        If MODE is SETUID then response(PROC)
                End if
        End if
        /* ส่วนของการดำเนินงานของซิซเท็มคอล */
}
```

5.4 บทสรุป

กลไกการตรวจจับการบุกรุกของระบบปฏิบัติการประกอบด้วย 3 โมดูลหลักซึ่งเรียกใช้โดยซิซเท็มคอลที่อาจจะก่อให้เกิดผลกระทบทางด้านความปลอดภัยของระบบเมื่อเรียกใช้ซิซเท็มคอลเหล่านั้นในขณะอยู่ในสถานะพิเศษ หลังจากที่แก้ไขซิซเท็มคอลเรียบร้อยแล้วจำเป็นต้องคอมไพล์เคอร์เนลและรีสตาร์ทระบบใหม่ เพื่อเริ่มสั่งงานระบบปฏิบัติการชุดใหม่

ในบทถัดไปเป็นการทดสอบประสิทธิภาพของกลไกการตรวจจับการบุกรุกโดยทดสอบทั้งความแม่นยำและความคงทน ทดสอบหาผลกระทบที่อาจจะเกิดขึ้นต่อระบบปฏิบัติการภายหลังการแก้ไขซิซเท็มคอล และท้ายที่สุดวัดประสิทธิภาพของระบบปฏิบัติการหลังจากที่ได้รับการแก้ไขเพื่อเปรียบเทียบเวลาทั้งก่อนและหลังแก้ไขระบบปฏิบัติการ