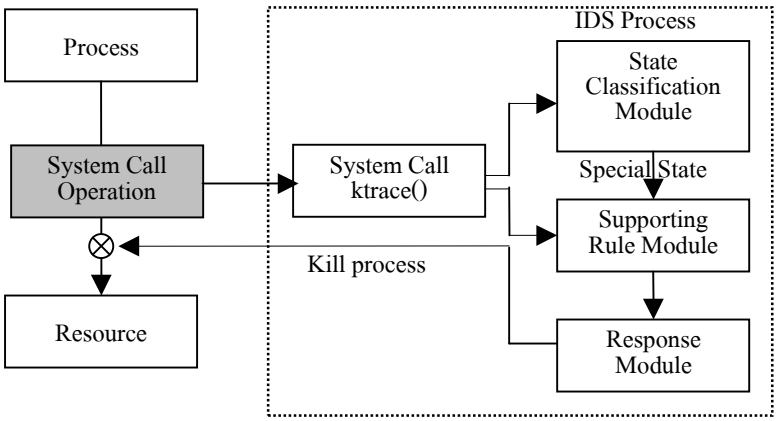
บทที่ 4
ระบบตรวจจับการบุกรุกในระดับของโปรแกรมประยุกต์

4.1. บทนำ

การออกแบบระบบของวิทยานิพนธ์ชุดนี้แบ่งออกเป็นสองส่วนได้แก่ ส่วนของการวิเคราะห์กฎสนับสนุน และส่วนของการทดสอบผลของการวิเคราะห์ โดยการพัฒนาวิธีการตรวจจับการบุกรุกในระดับของโปรแกรมประยุกต์ สำหรับการวิเคราะห์กฎต่างๆ นั้นได้อธิบายไปแล้วในบทที่ 3 และในบทนี้จะอธิบายถึงการพัฒนาโปรแกรมตรวจจับการบุกรุกซึ่งจะกล่าวถึงสถาปัตยกรรมของโปรแกรม วิธีการอ่านข้อมูลนำเข้าจากโปรเซสเป้าหมาย วิธีการติดตามการทำงานของโปรเซสในระบบปฏิบัติการ การทดสอบประสิทธิภาพของระบบตรวจจับการบุกรุกและผลกระทบจากระบบตรวจจับการบุกรุกที่มีผลต่อระบบระบบปฏิบัติการ

4.2. สถาปัตยกรรมของโปรแกรมตรวจจับการบุกรุก

โปรแกรมตรวจจับการบุกรุกชุดนี้จะตรวจสอบการทำงานของโปรเซสทุกโปรเซสในระบบปฏิบัติการ โดยติดตามการเปลี่ยนแปลงค่าประจำสถานะ การเรียกใช้ซิซเท็มคอลของโปรเซสเป้าหมาย ตามสถาปัตยกรรมของโปรแกรมตรวจจับการบุกรุกซึ่งแสดงไว้ในภาพประกอบที่ 4.1



ภาพ

ประกอบ 4.1 สถาปัตยกรรมของโปรแกรมตรวจจับการบุกรุก

ภาพประกอบที่ 4.1 แสดงสถาปัตยกรรมของระบบตรวจจับการบุกรุกในระดับโปรแกรมประยุกต์ โดยปกติแล้วโปรเซสจะร้องขอทรัพยากรผ่านซิซเท็มคอล ระบบตรวจจับการบุกรุกชุดนี้จึงติดตามการทำงานของโปรเซสโดยอ่านข้อมูลจากซิซเท็มคอลและทำงานแบบขนานไปกับการทำงานของโปรเซสที่กำลังติดตาม ระบบตรวจจับการบุกรุกพิจารณากิจกรรมของโปรเซสทีละซิซเท็มคอล ระบบตรวจจับการบุกรุกประกอบด้วยโมดูลหลัก 3 โมดูล ได้แก่ โมดูลนิยามสถานะ โมดูลกฎสนับสนุน และโมดูลตอบสนอง โดยแต่ละโมดูลมีรายละเอียดดังนี้

- **โมดูลนิยามสถานะ** (state classification module) เป็นโมดูลที่ใช้สำหรับการนิยามสถานะของโปรเซสในขณะนั้นว่าโปรเซสดังกล่าวอยู่ในสถานะใดใน 5 สถานะดังที่กล่าวมาแล้วในหัวข้อที่ 3.3.2 โมดูลนี้จะนิยามสถานะของโปรเซสจากค่า user credential จำนวน 4 ค่า ได้แก่ UID, EUID, GID และ EGID โดยอ่านค่ามาจากโปรเซสเป้าหมายด้วยซิซเท็มคอล ktrace()
- **โมดูลกฎสนับสนุน** (supporting rule module) เป็นโมดูลที่ใช้สำหรับการตรวจสอบการเรียกใช้ซิซเท็มคอลในขณะที่โปรเซสเป้าหมายอยู่ในสถานะพิเศษ โดยพิจารณาเฉพาะซิซเท็มคอลและค่าพารามิเตอร์ที่ได้มาจากการวิเคราะห์กฎสนับสนุนในหัวข้อที่ 3.3.3 ข้อมูลที่จำเป็นในการพิจารณาโปรเซสของโมดูลนี้คือหมายเลขซิซเท็มคอลและค่าพารามิเตอร์
- **โมดูลตอบสนอง** (response module) เป็นโมดูลที่ใช้สำหรับการจัดการกับโปรเซสบุกรุก สำหรับการพัฒนาโปรแกรมตรวจจับการบุกรุกในบทนี้เป็นเพียงการทดสอบผลของการวิเคราะห์เท่านั้นจึงไม่ทำลายโปรเซสบุกรุกในโปรแกรมชุดนี้ แต่จะบันทึกเหตุการณ์ไว้ในล็อกไฟล์ของระบบเพื่อใช้สำหรับการตรวจสอบภายหลังเท่านั้น แต่ถ้าในกรณีของการแก้ไขระบบปฏิบัติการแล้ว โมดูลนี้จะทำลายโปรเซสบุกรุกและโปรเซสอื่นที่มีเจ้าของคนเดียวกันกับโปรเซสบุกรุก

เมื่อโปรเซสเป้าหมายเริ่มทำงาน โปรเซสตรวจจับการบุกรุกจะติดตามการทำงานโปรเซสเป้าหมายโดยอ่านข้อมูลที่จำเป็นสำหรับการพิจารณาด้วยซิซเท็มคอล ktrace() หลังจากนั้นส่งข้อมูลที่ได้ให้แก่โมดูลนิยามสถานะเพื่อกำหนดสถานะให้แก่โปรเซส ถ้าหากโปรเซสเป้าหมายอยู่ในสถานะอื่นที่ไม่ใช่สถานะพิเศษแล้วโปรเซสดังกล่าวสามารถเข้าใช้ทรัพยากรตามที่ต้องการ แต่ถ้าหากโปรเซสเป้าหมายอยู่ในสถานะพิเศษ โปรเซสตรวจจับการบุกรุกตรวจสอบซิซเท็มคอลและพารามิเตอร์ ในกรณีที่โปรเซสเป้าหมายกระทำการใดที่ขัดต่อกฎสนับสนุนแล้วถือว่าโปรเซสดังกล่าวเป็นโปรเซสบุกรุกและบันทึกข้อมูลของโปรเซสไว้ในล็อกไฟล์ของระบบ

4.3. ข้อมูลนำเข้าของโปรแกรมตรวจจับการบุกรุก

ระบบตรวจจับการบุกรุกจะนิยามสถานะของโปรเซสและพิจารณาการเรียกใช้งานซิซเท็มคอลและค่าพารามิเตอร์ นอกจากนี้ยังต้องการหมายเลขโปรเซสของโปรเซสบุกรุกเพื่อใช้ในโมดูลตอบสนอง จากที่กล่าวมาข้างต้นสามารถสรุปได้ว่า ข้อมูลนำเข้าสำหรับโปรแกรมตรวจจับการบุกรุกประกอบไปด้วย

1. **PID** หมายถึง หมายเลขประจำโปรเซสที่กำลังติดตาม
2. **[UID, EUID, GID, EGID]** หมายถึงค่าประจำสถานะของโปรเซส
3. **System call ID** หมายถึงหมายเลขซิซเท็มคอลที่ถูกเรียกใช้โดยโปรเซสและถูกติดตามในขณะนั้น
4. **Parameter** หมายถึงค่าพารามิเตอร์ของซิซเท็มคอลที่ถูกเรียกใช้โดยโปรเซสและถูกติดตามในขณะนั้น

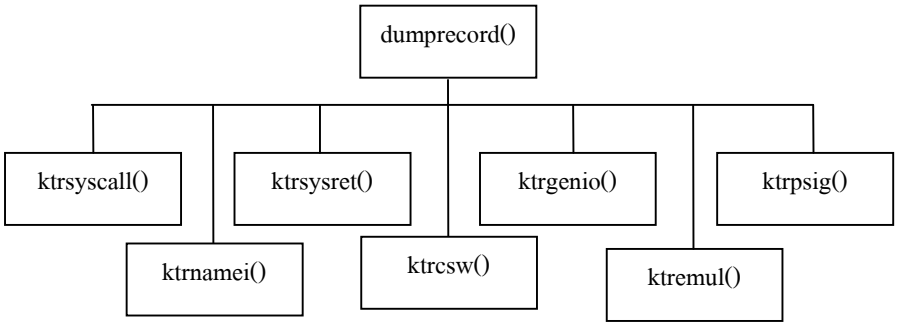
เนื่องจากข้อมูล ที่กล่าวมาข้างต้นจะเกิดขึ้นเพียงชั่วขณะที่โปรเซสกำลังทำงานและจัดเก็บอยู่ในหน่วยความจำของระบบที่โปรเซสปกติไม่สามารถเข้าถึงได้ ยกเว้นผู้ดูแลระบบเท่านั้นที่สามารถอ่านข้อมูลเหล่านี้ได้ด้วยซิซเท็มคอล ktrace() ตัวอย่างของอ่านข้อมูลนำเข้าซิซเท็มคอล ktrace() จะแสดงในหัวข้อที่ 4.4.2

4.4. การพัฒนาโปรแกรมตรวจจับการบุกรุก

สำหรับระบบปฏิบัติการเน็ทบีเอสดีมีคำสั่งของระบบที่เรียกใช้ซิซเท็มคอล ktrace ()จำนวน 2 คำสั่ง ได้แก่คำสั่ง ktrace และ ktruss ผลการทำงานของโปรแกรม ktrace ถูกบันทึกอยู่ในแฟ้มแบบไบนารี (binary file) ชื่อ ktrace.out ผู้ดูแลระบบต้องใช้โปรแกรม kdump เพื่อแปลงแฟ้มแบบไบนารีให้อยู่ในรูปแบบที่สามารถอ่านได้ แต่สำหรับผลการทำงานของโปรแกรม ktruss นั้นสามารถอ่านได้ทันที วิทยานิพนธ์ชุดนี้จึงเลือกโปรแกรม ktruss เป็นเครื่องมือสำหรับการพัฒนาโปรแกรมตรวจจับการบุกรุก

4.4.1. โครงสร้างของโปรแกรม ktruss

โปรแกรม ktruss ประมวลผลซิซเท็มคอลของโปรเซสทีละซิซเท็มคอล โดยประมวลผลตามลำดับการเรียกใช้งานจริง โครงสร้างของโปรแกรม ktruss ประกอบด้วยโมดูลหลักที่สำคัญดังแสดงไว้ในภาพประกอบที่ 4.2



ภาพประกอบ 4.2 โมดูลของโปรแกรม ktruss

จากภาพประกอบที่ 4.2 โปรแกรม ktrace มีโมดูลหลักคือ dumprecord() ซึ่งมีหน้าที่สำหรับการประมวลชุดคำสั่งของโปรเซสที่กำลังติดตามทีละซิซเท็มคอลจนกว่าจะจบโปรเซส ภายในโมดูลดังกล่าวจะตรวจสอบชนิดของ trace point (คำอธิบายของ trace point กล่าวไว้ในหัวข้อที่ 3.2.1) เพื่อเรียกใช้โมดูลมารองรับการทำงานที่แตกต่างกันตามชนิดของ trace point เช่น ถ้า trace point มีค่าเป็น syscall หมายความว่าโปรแกรม ktruss กำลังติดตามการใช้งานซิซเท็มคอล โมดูล dumpreccord() จะเรียกใช้โมดูล ktrsyscall() มาอ่านข้อมูลที่ได้จากซิซเท็มคอล ktrace() แล้วแสดงชื่อพร้อมค่าพารามิเตอร์ของซิซเท็มคอล สำหรับโปรแกรมตรวจจับการบุกรุกชุดนี้สนใจเฉพาะโมดูล ktrsyscall() และ ktrsysret() เนื่องจากโปรแกรมดังกล่าวจะติดตามและผลการทำงานของซิซเท็มคอล

4.4.2. การอ่านข้อมูลนำเข้าจากโปรแกรม ktruss

ภายในโมดูล ktrsyscall() และ ktrsysret() มีการอ้างถึงโครงสร้างข้อมูลชื่อ struct ktr_syscall และ struct ktr_sysret ตามลำดับ โครงสร้างข้อมูลทั้งสองนี้มีไว้สำหรับจัดเก็บข้อมูลสำคัญของซิซเท็มคอลซึ่งเป็นข้อมูลที่ใช้ในโปรแกรมตรวจจับการบุกรุก แต่ละโมดูลมีรายละเอียดดังนี้

โมดูล ktrsyscall()

ภายในโมดูลนี้มีการดำเนินงานสองขั้นตอนคือการนิยามสถานะของโปรเซส และการติดตามการเรียกใช้ซิซเท็มคอลคอลต่างๆ โดยพิจารณาจาก ค่าประจำสถานะ ซิซเท็มคอล และค่าพารามิเตอร์ ซึ่งถูกจัดเก็บอยู่ในโครงสร้างข้อมูลชื่อ struct ktr_syscall และ struct ktr_header สำหรับ struct ktr_syscall ประกอบไปด้วยสมาชิกจำนวน 3 ตัวได้แก่

```
struct ktr_syscall {
        int     ktr_code;               /* syscall number */
        int     ktr_argsize;            /* size of arguments */
        /* followed by ktr_argsize/sizeof(register_t) "register_t"s */
};
```

สมาชิกแต่ละตัวมีความหมายดังนี้

- **ktr_code** เป็นตัวแปรแบบ integer ใช้สำหรับเก็บหมายเลขซิซเท็มคอลซึ่งถูกนิยามไว้ในแฟ้มชื่อ /usr/src/sys/kern/master.syscall
- **ktr_argsize** เป็นตัวแปรแบบ integer ใช้สำหรับเก็บขนาดของพารามิเตอร์
- **ค่าพารามิเตอร์** เป็นข้อมูลแบบ register_t แต่ละซิซเท็มคอลมีพารามิเตอร์จำนวน ktr_argsize/sizeof(register_t) ตัว

ตารางที่ 4.1 เป็นการแสดงชื่อข้อมูล และวิธีการอ่านข้อมูล ข้อมูลนำเข้าซึ่งสามารถอ่านออกมาจากโครงสร้างข้อมูลทั้งสองโดยตรง ยกเว้นค่าพารามิเตอร์นั้นไม่สามารถอ่านออกมาได้ ต้องอาศัยชุดคำสั่งต่อไปนี้เพื่ออ่านค่าพารามิเตอร์แต่ละตัวออกมา

```
register_t * ap;
long arg_1, arg_2;
ap = (register_t *)((char *)ktr + sizeof(struct ktr_syscall))
arg_1 = (long) *ap; // สำหรับ พารามิเตอร์ตัวที่ 1
ap++;
arg_2 = (long) *ap; // สำหรับ พารามิเตอร์ตัวที่ 2
```

ตารางที่ 4.1 แสดงข้อมูลนำเข้าของระบบตรวจจับการบุกรุก

ชื่อข้อมูล	โครงสร้างข้อมูล	ชนิดข้อมูล	ชื่อตัวแปร
UID	struct ktr_header	uid_t	ktr_uid
EUID	struct ktr_header	uid_t	ktr_euid
GID	struct ktr_header	gid_t	ktr_gid
EGID	struct ktr_header	gid_t	ktr_egid
SysID	struct ktr_syscall	int	ktr_code
Parameter	-	register_t *	ap

โมดูล ktrsysret()

ในบางกรณีโปรแกรมตรวจจับการบุกรุกพิจารณาโปรเซสจากผลของทำงานของซิซเท็มคอล ได้แก่ ซิซเท็มคอล setuid(), seteuid(), setgid() และ setegid() โปรแกรมตรวจจับการบุกรุกจะไม่สามารถตรวจสอบการทำงานได้จากโมดูล ktrsyscall() เนื่องจากผลการดำเนินงานของซิซเท็มคอลยังไม่เสร็จสมบูรณ์ แต่สามารถตรวจสอบผลการทำงานได้จากโมดูล ktrsysret()

โมดูลนี้อ่านข้อมูลจากโครงสร้างข้อมูล struct ktr_header เพื่อใช้สำหรับการนิยามสถานะเช่นเดียวกับโมดูลก่อนหน้า แต่อ่านหมายเลขซิซเท็มคอลและค่าที่ถูกคืนกลับมาจากโครงสร้างข้อมูล struct ktr_sysret ประกอบไปด้วยสมาชิกดังต่อไปนี้

```
struct ktr_sysret {
        short     ktr_code;
        int       ktr_error;
        register_t ktr_retval;
        register_t ktr_retval_1;
};
```

โครงสร้างข้อมูลนี้มีสมาชิกหลายตัว แต่จะกล่าวถึงเฉพาะสมาชิกที่เกี่ยวโปรแกรมชุดนี้เท่านั้นซึ่งได้แก่

- **ktr_syscode** เป็นตัวแปรแบบ integer ใช้สำหรับเก็บหมายเลขซิซเท็มคอลซึ่งถูกนิยามไว้ในแฟ้มชื่อ /usr/src/sys/kern/master.syscall เช่นเดียวกับ struct ktr_syscall
- **ktr_retval** เป็นตัวแปรแบบ register_t ใช้สำหรับเก็บค่าที่ถูกคืนกลับมา โดยค่าดังกล่าวจะแสดงถึงผลการดำเนินงานของซิซเท็มคอล

สำหรับวิธีการอ่านข้อมูลนั้นสามารถอ่านออกมาจากโครงสร้างได้โดยตรงทั้งสองค่านั้นคือ หมายเลขซิซเท็มคอลซึ่งอ่านจากค่า ktr_syscode และค่าที่คืนกลับอ่านจากค่า ktr_retval สำหรับค่าประจำสถานะนั้นมีวิธีการอ่านเช่นเดียวกับโมดูล ktrsyscall() ซึ่งแสดงไว้ในตารางที่ 4.1

4.4.3. โมดูลที่สำคัญของโปรแกรมตรวจจับการบุกรุก

โปรแกรมตรวจจับการบุกรุกประกอบด้วยโมดูลหลักจำนวน 3 โมดูล ได้แก่ โมดูลนิยามสถานะ โมดูลกฎสนับสนุน และโมดูลตอบสนอง นอกจากนี้ยังมีโมดูลย่อยที่ออกแบบขึ้นมาเพื่ออำนวยความสะดวกของการทำงาน โมดูลที่เกี่ยวข้องกับโปรแกรมตรวจจับการบุกรุกต่อไปนี้

int state(struct ktr_header *kth)

ข้อมูลนำเข้า kth คือโครงสร้างข้อมูลซึ่งเก็บค่า user credential ของโปรเซส

ค่าคืนกลับ สถานะของโปรเซส ซึ่งนิยามไว้ในแฟ้ม ids.h

คำอธิบาย เป็นฟังก์ชันที่ใช้สำหรับนิยามสถานะของโปรเซส

อัลกอริทึม แสดงไว้ในหัวข้อที่ 3.3.2

void ktr_syscall(struct ktr_header *kth, struct ktr_syscall*ktr)

ข้อมูลนำเข้า kth คือโครงสร้างข้อมูลซึ่งเก็บค่า user credential ของโปรเซส
ktr คือโครงสร้างข้อมูลซึ่งเก็บรายละเอียดของซิซเท็มคอลที่กำลังตรวจสอบ

คำอธิบาย เป็นฟังก์ชันสำหรับการตรวจสอบการเรียกใช้ซิซเท็มคอลเมื่อโปรเซสอยู่ในสถานะพิเศษ

อัลกอริทึม

```
ktr_syscall(kth, ktr)
  uid is uid(kth)
  pid is pid(kth)
  if syscall_id is execve() then
     if path is "pwd_mkdb" then
       call set(pid, uid)
     else
       call response(uid, "execve()", R#1)
     end if
     return
  end if
  if syscall_id is exit() then
     if search(pid)>=0 then
       call response(uid, "exit()", R#4)
     end if
  return
end if

if syscall_id is in critical system call then
  call response(uid, "exit()", R#5)
  return
end if

if syscall_id is in chmod_group and flag is setuid
then
  call response(uid, "exit()", R#2)
  return
end if

if syscall_id is open()  then
  call support_open(kth, filename, flag, mode)
  return
end if
```

Void ktr_sysret(struct ktr_header *kth, struct ktr_sysret*ktr)

ข้อมูลนำเข้า kth คือโครงสร้างข้อมูลซึ่งเก็บค่า user credential ของโปรเซส
ktr คือโครงสร้างข้อมูลเก็บรายละเอียดของซิซเท็มคอลที่กำลังตรวจสอบ

คำอธิบาย เป็นฟังก์ชันสำหรับการตรวจสอบผลของการเรียกใช้ซิซเท็มคอลเมื่อโปรเซสอยู่ในสถานะพิเศษ

อัลกอริทึม

```
ktrsysret(kth, ktr)
if syscall is in chage_id_group  and state(kth) is Super user group then
        call response(uid, "seuid/setgid()", R#0)
end if
return
```

void support_open(struct ktr_header *kth, char *path, long flag, long mode);

ข้อมูลนำเข้า kth คือโครงสร้างข้อมูลซึ่งเก็บค่า user credential ของโปรเซส
path คือชื่อแฟ้มแบบเต็ม (Full path)
flag คือรูปแบบของการเปิดแฟ้ม เช่นเปิดแฟ้มเพื่อเขียน อ่าน หรือแก้ไข
mode คือการกำหนดสิทธิ์ของการเข้าถึงแฟ้ม เช่น แฟ้มแบบ setuid

คำอธิบาย เป็นฟังก์ชันที่ใช้สำหรับการพิจารณาโปรเซสเมื่อโปรเซสเรียกใช้ซิซเท็มคอล open() ในกรณีที่โปรเซสอยู่ในสถานะพิเศษ

อัลกอริทึม

```
Support_open(kth, path, flag, mode)
    If path is "/etc/spwd.db" then
        call unset(uid)
        return
    End if
  If flag is CREAT and mode is SETUID then
        call response(uid, "open()", R#2)
        return
    End if
  If flag is MODIFY  and path is command then
        call response(uid, "open()", R#3)
        return
End if
        Return
```

สำหรับกฎสนับสนุนข้อที่ 4 จำเป็นต้องมีโครงสร้างข้อมูลเพื่อจัดเก็บโปรเซสที่มีแนวโน้มว่าเป็นโปรเซสบุกรุกแต่ต้องรอการตรวจสอบ โครงสร้างข้อมูลดังกล่าวมีสมาชิกดังนี้

- **uid** เป็นข้อมูลแบบ uid_t ใช้สำหรับเก็บหมายเลขประจำตัวผู้ใช้ซึ่งเป็นเจ้าของโปรเซสที่กำลังถูกติดตาม
- **pid** เป็นข้อมูลแบบ pid_t ใช้สำหรับเก็บหมายเลขโปรเซสที่กำลังติดตาม

void setCR_PS(pid_t pid, uid_t uid)

ข้อมูลนำเข้า pid คือหมายเลขโปรเซส

uid คือหมายเลขประจำตัวผู้ใช้

คำอธิบาย เป็นฟังก์ชันที่ใช้สำหรับเพิ่มข้อมูลในโครงสร้างข้อมูล

อัลกอริทึม

```
set (pid, uid)
  For each record in list
    If uid( record)<0  then
        pid(record) = pid
        uid(record) = uid
        exit from loop
    End if
Return
```

int searchCR_PS(pid_t pid)

ข้อมูลนำเข้า pid คือหมายเลขโปรเซสที่ต้องการค้นหา

ค่าคืนกลับ ตำแหน่งของข้อมูลในโครงสร้างข้อมูล หรือคืนค่า -1 ในกรณีค้นหาไม่พบ

คำอธิบาย เป็นฟังก์ชันที่ใช้สำหรับค้นหาข้อมูลในโครงสร้างข้อมูล

อัลกอริทึม

```
search (key)
For each record in list
    If key = uid( record) then
      Return REC#(record)
Return -1
```

void unsetCR_PS(uid_t uid)

ข้อมูลนำเข้า uid คือหมายเลขประจำตัวผู้ใช้

คำอธิบาย เป็นฟังก์ชันที่ใช้สำหรับลบข้อมูลในโครงสร้างข้อมูล

อัลกอริทึม

unset (key)

For each record in list

If key = uid(record) then

pid(record)= -1

uid(record) = -1

End if

Return

void response(uid_t uid, char *syscall, int rule)

ข้อมูลนำเข้า	uid คือหมายเลขประจำตัวผู้ใช้ syscall คือชื่อซิซเท็มคอล rule คือลำดับของกฎสนับสนุน
คำอธิบาย	เป็นฟังก์ชันที่ใช้สำหรับตอบสนองต่อโปรเซสที่ถูกพิจารณาว่าเป็นโปรเซสบุกรุก
อัลกอริทึม	สำหรับการพัฒนาระบบตรวจจับการบุกรุกในระดับของโปรเซสจะบันทึกข้อมูลของโปรเซสบุกรุกไว้ในล็อกไฟล์ของระบบด้วยฟังก์ชัน syslog() เท่านั้น และไม่ทำลายโปรเซสดังกล่าว

4.5. การติดตามการทำงานของโปรเซสในระบบปฏิบัติการ

โปรแกรมตรวจจับการบุกรุกจะต้องทำงานตลอดเวลา ทุกครั้งที่ระบบปฏิบัติการเริ่มทำงานโปรแกรมตรวจจับการบุกรุกจะต้องถูกสั่งงานขึ้นมาเช่นเดียวกับโปรแกรมอื่นๆ เพื่อตรวจสอบการทำงานของโปรเซสทุกโปรเซส แต่บางกรณีโปรเซสตรวจจับการบุกรุกจะยกเว้นการตรวจสอบโปรเซสบางโปรเซสเช่น โปรเซสของระบบปฏิบัติการ โปรเซสที่มีเจ้าของเป็น root เป็นต้น โปรแกรมตรวจจับการบุกรุกชุดนี้จะตรวจสอบโปรเซสเดมอน (daemon process) ของระบบปฏิบัติการซึ่งเป็นโปรเซสเริ่มต้น และตรวจสอบการทำงานของโปรเซสลูกของโปรเซสเดมอนทุกโปรเซส

PID	TT	STAT	TIME	COMMAND
1	??	Ss	0:00.00	init
118	??	Ss	0:00.00	/sbin/dhclient
124	??	Ss	0:00.03	/usr/sbin/syslogd -s
338	??	Rs	0:00.09	/usr/sbin/sshd
344	??	Ss	0:00.00	/usr/sbin/inetd -l
365	??	Rs	0:00.00	/usr/sbin/cron
11	E0	Ss+	0:00.04	sh /etc/rc autoboot

ภาพประกอบ 4.3 โปรเซสเริ่มต้นของระบบปฏิบัติการ

ภาพประกอบที่ 4.3 เป็นตัวอย่างของโปรเซสเริ่มต้นของระบบปฏิบัติการเน็ทบีเอสดี ซึ่งแสดงให้เห็นว่า มีโปรเซสหลักที่ต้องตรวจสอบเพียงไม่กี่โปรเซสเท่านั้น เช่น init (PID 1) และ inetd (PID 344) เป็นต้น เนื่องจากโปรเซส init เป็นโปรเซสหลักของระบบปฏิบัติการมีหน้าที่สร้างโปรเซสลูกอื่นๆ โปรเซส inetd เป็นโปรเซสหลักที่คอยสร้างโปรเซสสำหรับการให้บริการของระบบเช่น ftp หรือ telnet เป็นต้น นอกจากนี้ถ้าหากระบบปฏิบัติการเป็นเครื่องให้บริการเว็บ จดหมายอิเล็กทรอนิกส์ หรือบริการอื่นๆ โปรเซสตรวจจับการบุกรุกจะเฝ้าติดตามการทำงานโปรเซสเดมอนของบริการเหล่านั้นเพิ่มเติม ดังตัวอย่าง ระบบปฏิบัติการได้ให้บริการ remote login ผ่านโปรเซส ssh (PID 338)

4.6. การทดสอบประสิทธิภาพของโปรแกรมตรวจจับการบุกรุก

การออกแบบและพัฒนาโปรแกรมตรวจจับการบุกรุกชุดนี้จัดทำขึ้นมาเพื่อหาซิซเท็มคอลและวิธีการแก้ไขซิซเท็มคอลจึงจำเป็นที่จะต้องทดสอบประสิทธิภาพของโปรแกรมทั้งด้านความแม่นยำของการตรวจจับการบุกรุกและการทดสอบหาผลกระทบที่อาจจะเกิดขึ้นแก่ระบบระบบปฏิบัติการ

การทดสอบระบบในหัวข้อนี้แบ่งออกเป็น 3 หัวข้อได้แก่ การทดสอบความแม่นยำตามกฎสนับสนุน การทดสอบความแม่นยำด้วยโปรแกรมบุกรุก และการทดสอบเพื่อหาผลกระทบของโปรแกรมตรวจจับการบุกรุกที่มีผลต่อระบบปฏิบัติการ

4.6.1. การทดสอบความแม่นยำตามกฎสนับสนุน

การทดสอบในหัวข้อนี้เป็นการทดสอบความแม่นยำของกฎสนับสนุน ทดสอบโดยสั่งงานโปรแกรมบุกรุกซึ่งพัฒนาโดยผู้ทำวิทยานิพนธ์ แล้วกำหนดค่าให้โปรเซสบุกรุกอยู่ในสถานะพิเศษและสั่งงานชุดคำสั่งบุกรุกผ่านจุดอ่อนของระบบปฏิบัติการ

กรณีทดสอบและผลการทดสอบประกอบด้วย 3 ส่วนได้แก่ กรณีทดสอบ ผลกระทบ และผลการทดสอบ โดยแต่ละหัวข้อนั้นมีความหมายดังนี้

- **กรณีทดสอบ** หมายถึงชุดคำสั่งทดสอบซึ่งจะสั่งงานคำสั่งเหล่านั้นในขณะที่โปรเซสอยู่ในสถานะพิเศษ สำหรับเครื่องหมาย '#' หมายถึงโปรเซสอยู่สถานะพิเศษ
- **ผลกระทบ** หมายถึงผลลัพธ์ที่จะเกิดขึ้นถ้าหากการโจมตีระบบประสบความสำเร็จ
- **ผลการทดสอบ** หมายถึงผลการทำงานของโปรเซสตรวจจับการบุกรุกซึ่งจะแสดงข้อความที่ถูกบันทึกไว้ในล็อกไฟล์ของระบบ

กรณีทดสอบ	$ su
คำอธิบาย	เป็นคำสั่งสำหรับการเปลี่ยนสิทธิ์ของผู้ใช้ปกติให้มีสิทธิ์เท่ากับ root
ผลกระทบ	ผู้บุกรุกได้รับสิทธิ์ของ root เท่ากับ root
ผลการทดสอบ	*User 1000 called setgid() and attempted to compromise supporting rule #0*

กรณีทดสอบ	# exec bash
คำอธิบาย	ในขณะที่โปรเซสอยู่ในสถานะพิเศษ โปรเซสดังกล่าวสั่งงานคำสั่ง bash
ผลกระทบ	โปรแกรม bash ทำงานด้วยสิทธิพิเศษเทียบเท่ากับ root
ผลการทดสอบ	*User 1000 called execve() and attempted to compromise supporting rule #1*

กรณีทดสอบ	# chmod +s msh
คำอธิบาย	เปลี่ยนสิทธิ์ของโปรแกรม msh ให้เป็นโปรแกรมแบบ setuid
ผลกระทบ	โปรแกรม msh ทำงานด้วยสิทธิของ root
ผลการทดสอบ	*User 1000 called chmod() and attempted to compromise supporting rule #2*

กรณีทดสอบ	#./create_suid
คำอธิบาย	โปรแกรม create_suid สร้างโปรแกรมแบบ setuid ชื่อ /tmp/attack
ผลกระทบ	โปรแกรม /tmp/attack ทำงานด้วยสิทธิ์ของ root
ผลการทดสอบ	*User 1000 called open() and attempted to compromise supporting rule #2*

กรณีทดสอบ # cp msh /bin/sh

คำอธิบาย ผู้บุกรุกสร้างโปรแกรม /bin/sh ใหม่ซึ่งคัดลอกจากโปรแกรม msh

ผลกระทบ โปรแกรม /bin/sh ใหม่ทำงานตามชุดคำสั่งที่ผู้บุกรุกได้โปรแกรมขึ้นใหม่

ผลการทดสอบ *User 1000 called open() and attempted to compromise supporting rule #3*

กรณีทดสอบ # vipw

คำอธิบาย ผู้บุกรุกแก้ไขฐานข้อมูลผู้ใช้ด้วยคำสั่ง vipw

ผลกระทบ ผู้บุกรุกเปลี่ยนค่า UID ของผู้ใช้ให้มีค่าเป็น 0

ผลการทดสอบ *User 1000 called exit() and attempted to compromise supporting rule #4*

กรณีทดสอบ # mount /dev/sd0e /mnt

คำอธิบาย ติดตั้งดิสก์ใหม่แก่ระบบปฏิบัติการ

ผลกระทบ ผู้บุกรุกสามารถสั่งงานโปรแกรมบนดิสก์หรือคัดลอกข้อมูลออกจากระบบ

ผลการทดสอบ *User 1000 called mount() and attempted to compromise supporting rule #5*

กรณีทดสอบ # eject /dev/sd0e

คำอธิบาย ผู้บุกรุกสามารถยกเลิกดิสก์ที่ถูกติดตั้งอยู่ในระบบได้

ผลกระทบ ดิสก์ที่ถูกยกเลิกอาจจะเป็นดิสก์ที่เก็บข้อมูลสำคัญของระบบ

ผลการทดสอบ *User 1000 called unmount() and attempted to compromise supporting rule #5*

กรณีทดสอบ # date 0101051200

คำอธิบาย ผู้บุกรุกสามารถกำหนดเวลาของระบบใหม่

ผลกระทบ เกิดผลกระทบต่อกิจกรรมของระบบที่มีเวลามาเกี่ยวข้อง

ผลการทดสอบ *User 1000 called settimeofday() and attempted to compromise supporting rule #5*

กรณีทดสอบ # reboot

คำอธิบาย ผู้บุกรุกสามารถสั่งรีสตาร์ท ระบบปฏิบัติการได้

ผลการทำงาน เป็นการก่อกวนระบบ ทำให้กิจกรรมของระบบหยุดชะงัก

ผลการทดสอบ ไม่สามารถตรวจจับการบุกรุกได้

ตัวอย่างการบันทึกผลการทำงานในล็อกไฟล์ของระบบแสดงข้อความว่า

User 1000 called setuid() and attempted to compromise supporting rule #0

ข้อความข้างต้นหมายความว่า ผู้ใช้หมายเลข 1000 ได้พยายามที่จะบุกรุกระบบโดยการเรียกใช้ซิซเท็มคอล setuid() จากการกระทำดังกล่าวขัดต่อกฎสนับสนุนข้อที่ 0

จากผลการทดสอบพบว่าโปรแกรมตรวจจับการบุกรุกสามารถตรวจจับเหตุการณ์ผิดปกติได้เกือบทั้งหมด ยกเว้นทดสอบระบบด้วยคำสั่ง reboot ซึ่งเรียกใช้ซิซเท็มคอล reboot() โปรแกรมตรวจจับการบุกรุกไม่สามารถตรวจจับการกระทำข้างต้นได้ได้เนื่องจากซิซเท็มคอล ktrace() ได้รับข้อมูลจากซิซเท็มคอลหลังจากที่ทำงานเสร็จเรียบร้อยแล้ว เมื่อซิซเท็มคอล reboot() ทำงานเรียบร้อยแล้วเป็นผลให้ระบบปฏิบัติการถูกรีสตาร์ท โปรเซสต่างๆ ของระบบรวมไปถึงโปรเซสตรวจจับการบุกรุกถูกทำลายโดยระบบปฏิบัติการ

4.6.2. การทดสอบความแม่นยำด้วยโปรแกรมบุกรุกที่ได้จากการสืบค้นทางอินเทอร์เน็ต

ปัญหาของการทดสอบระบบในหัวข้อนี้คือ โปรแกรมบุกรุกแต่ละตัวมีความจำเพาะเจาะจงกับโปรแกรมเป้าหมาย รุ่นของโปรแกรม และระบบปฏิบัติการ อีกทั้งจุดอ่อนที่ตรวจพบในปัจจุบันได้รับการแก้ไขเรียบร้อยแล้ว ดังนั้นโปรแกรมบุกรุกที่สืบค้นมาไม่สามารถโจมตีจุดอ่อนเหล่านั้นได้สำเร็จ

การทดสอบระบบหัวข้อนี้จึงแก้ไขปัญหาดังกล่าวโดยจำลองสถานการณ์ว่า ผู้บุกรุกสั่งงานชุดคำสั่งบุกรุกผ่านจุดอ่อนของโปรแกรมแบบ setuid กรณีทดสอบในหัวข้อนี้เป็นชุดคำสั่งที่ได้มาจากโปรแกรมบุกรุกใช้สำหรับการทดสอบระบบของผู้ดูแลระบบซึ่งได้จากการสืบค้นทางอินเทอร์เน็ต จำนวน 12 กรณีทดสอบ ผลการทดสอบแสดงไว้ในตารางที่ 4.2

ตารางที่ 4.2 แสดงผลการทดสอบความแม่นยำด้วยโปรแกรมบุกรุก

กรณีทดสอบ	จำนวน	ผลการทดสอบระบบ
1. เป้าหมายของผู้บุกรุกคือสิทธิ์ของ root ดังนั้นผู้บุกรุกจึงพยายามที่	1	ตรวจจับการบุกรุกได้

	จะเปลี่ยนค่า UID ของผู้บุกรุกให้มีค่าเท่ากับ 0		
2.	ผู้บุกรุกสร้างโปรแกรมแบบ setuid ด้วยซิซเท็มคอล chmod()	1	ตรวจจับการบุกรุกได้
3.	ผู้บุกรุกสั่งงานโปรเซสบุกรุกด้วยซิซเท็มคอล execve()	10	ตรวจจับการบุกรุกได้

จากผลการทดสอบระบบด้วยโปรแกรมบุกรุกสรุปได้ว่า โปรแกรมตรวจจับการบุกรุกสามารถตรวจจับโปรเซสใดๆ ที่พยายามที่จะขัดต่อกฎสนับสนุน

4.6.3. การทดสอบเพื่อหาผลกระทบของโปรแกรมตรวจจับการบุกรุกที่มีต่อระบบปฏิบัติการ

คำสั่งที่อาจจะได้รับผลกระทบจากโปรแกรมตรวจจับการบุกรุกชุดนี้คือโปรแกรมแบบ setuid เนื่องจากโปรเซสเหล่านี้ทำงานในสถานะพิเศษจึงมีโอกาสที่จะได้รับผลกระทบดังกล่าว โปรแกรมแบบ setuid ของระบบปฏิบัติการเน็ทบีเอสดีรุ่น 3.0 มีทั้งหมด 63 คำสั่ง บางคำสั่งมีความเสี่ยงต่อความปลอดภัยของระบบและถูกยกเลิกคำสั่งเหล่านั้นไปแล้วได้แก่ คำสั่งในกลุ่ม uucp เช่น uucp, uustat และ uux เป็นต้น คำสั่งในกลุ่ม “R Command” เช่น rlogin, rcmd และ rsh เป็นต้น อีกทั้งในบางคำสั่งถูกจัดกลุ่มเป็นกลุ่มเดียวกัน เช่น chsh, chfn และ chpass เป็นต้น หรือคำสั่งในกลุ่มของ passwd, kpasswd หรือ yppasswd เป็นต้น ดังนั้นการทดสอบเพื่อหาผลกระทบทดสอบโดยสั่งงานโปรแกรมแบบ setuid ของระบบปฏิบัติการ เช่น passwd traceroute ping su lock เป็นต้น หลังจากนั้นตรวจสอบผลการทำงาน

เมื่อสั่งงานโปรแกรมแบบ setuid แล้วผลปรากฏว่า โปรแกรมตรวจจับการบุกรุกไม่มีกระทบต่อคำสั่งปกติ แต่เกิดผลกระทบกับโปรแกรมแบบ setuid ในบางคำสั่งได้แก่

- **คำสั่ง su** เนื่องจากคำสั่งดังกล่าวได้เรียกใช้ซิซเท็มคอล setgid() เพื่อเปลี่ยนค่า GID ของโปรเซสแล้วเป็นผลให้โปรเซสเปลี่ยนสถานะเป็นสถานะกลุ่มระบบ จากการกระทำข้างต้นซึ่งละเมิดกฎสนับสนุนข้อที่ 0
- **คำสั่ง passwd chpass chsh และ chfn** เนื่องจากในขณะที่โปรเซสอยู่ในสถานะพิเศษได้เรียกใช้ซิซเท็มคอล execve() เพื่อสั่งงานคำสั่ง pwd_mkdb ซึ่งมีหน้าจัดการฐานข้อมูลผู้ใช้ การกระทำข้างต้นได้ละเมิดกฎสนับสนุนข้อที่ 1 แต่เนื่องจากปัญหาในข้อนี้สามารถแก้ไขได้โดยแก้ไขกฎสนับสนุนเพื่อยกเว้นการสั่งงานโปรแกรมบางโปรแกรมที่เป็นโปรแกรมที่น่าเชื่อถือ เช่น pwd_mkdb เป็นต้น

นอกจากนี้ในระหว่างการทดลอง โปรแกรมตรวจจับการบุกรุกใช้เนื้อที่หน่วยความจำจำนวนมากเก็บข้อมูลที่มาจากซิซเท็มคอลที่กำลังติดตามการทำงาน จนในที่สุดผลให้โปรเซสตรวจจับการบุกรุกหยุดการทำงาน เนื่องจากโปรแกรมชุดนี้ถูกใช้เป็นเพียงตัวทดสอบความถูกต้องและความแม่นยำของการตรวจจับการบุกรุกและหาผลกระทบของระบบตรวจจับการบุกรุกที่มีผลต่อระบบปฏิบัติการ

เท่านั้นไม่ได้นำมาใช้งานจริงจึงไม่จำเป็นต้องสนใจข้อจำกัดในส่วนนี้มากนักแต่ในหัวข้อถัดไปซึ่งเป็นการแก้ไขระบบปฏิบัติการ จำเป็นที่จะต้องตระหนักถึงประสิทธิภาพของระบบตรวจจับการบุกรุกและการใช้ทรัพยากรของระบบตรวจจับการบุกรุก

4.7. บทสรุป

จากการศึกษา วิเคราะห์กฎสนับสนุนในบทก่อนหน้านี้สามารถนำแนวคิดดังกล่าวมาพัฒนาโปรแกรมตรวจจับการบุกรุกได้ดังรายละเอียดที่กล่าวไปแล้วในบทนี้ เมื่อทดสอบระบบตรวจจับผู้บุกรุกตามกฎสนับสนุนและโปรแกรมบุกรุกพบว่า ระบบตรวจจับการบุกรุกสามารถตรวจจับเหตุการณ์ที่เกิดขึ้นจากการโจมตีระบบผ่านจุดอ่อนได้ หากวิธีการบุกรุกนั้นละเมิดกฎสนับสนุน ยกเว้นการโจมตีระบบด้วยซิซเท็มคอล reboot() เนื่องจากการตรวจสอบการทำงานของโปรแกรมจะเกิดขึ้นหลังจากที่ซิซเท็มคอลแต่ละตัวทำงานเสร็จสิ้นแล้วเป็นผลให้ระบบปฏิบัติการรีสตาร์ทระบบใหม่ ผลที่ตามมาคือโปรเซสตรวจจับผู้บุกรุกถูกทำลาย

สำหรับผลกระทบที่เกิดขึ้นต่อกระบวนการทำงานปรกติของระบบปฏิบัติการเมื่อติดตั้งระบบตรวจจับผู้บุกรุกจะมีผลกระทบต่อโปรแกรมแบบ setuid เท่านั้น ซึ่งได้แก่คำสั่ง su ผู้ดูแลระบบต้องกำหนดนโยบาย ยกเลิกคำสั่งดังกล่าวชั่วคราว จนกว่าจะพัฒนาคำสั่ง su ขึ้นใหม่ที่ไม่ขัดต่อกฎสนับสนุนและห้ามมิให้ผู้ใช้ใดๆ ในระบบใช้คำสั่งนี้เปลี่ยนสิทธิ์การทำงานเป็น root หากผู้ใช้ต้องการใช้สิทธิ์ของ root นั้นจำเป็นต้องเข้าสู่ระบบผ่าน console เท่านั้น

โปรเซสตรวจจับการบุกรุกเป็นทำงานในระดับของผู้ใช้ทั่วไป การเข้าถึงตารางโปรเซส และตารางอื่นๆ ที่เกี่ยวข้องนั้นต้องอาศัยซิซเท็มคอล อีกทั้งในขณะที่โปรเซสตรวจจับการบุกรุกกำลังทำงาน ระบบปฏิบัติการต้องคอยจัดการกับหน่วยความจำจำนวนมากจนโปรเซสตรวจจับการบุกรุกหยุดการทำงาน นอกจากนี้จำนวนโปรเซสตรวจจับการบุกรุกขึ้นอยู่กับจำนวนของโปรเซส ณ เวลาที่สั่งงานโปรแกรมตรวจจับผู้บุกรุก

ในบทต่อไปเป็นการนำวิธีการตรวจจับการบุกรุกซึ่งได้พิสูจน์แล้วในระดับของโปรแกรมประยุกต์มาแก้ไขระบบปฏิบัติการโดยแก้ไขซิซเท็มคอลต่างๆ ที่เกี่ยวข้องกับความปลอดภัยของระบบปฏิบัติการเพื่อให้ระบบปฏิบัติการสามารถตรวจจับและป้องกันการบุกรุกได้