บทที่ 4

การทดสอบระบบตรวจจับการบุกรุก

4.1 บทนำ

ในบทนี้จะกล่าวถึงการทดสอบและผลจากการทดสอบระบบการตรวจจับการบุกรุกที่ได้พัฒนาขึ้น โดยในส่วนแรกจะกล่าวถึงสภาพแวดล้อมที่ใช้ในการทดสอบถัดมาเป็นวิธีการและผลจากการทดสอบ โดยแบ่งการทดสอบเป็นสองวิธีคือโดยการเขียนโปรแกรมขึ้นมาเพื่อทดสอบกฎแต่ละข้อ และการทดสอบโดยใช้เครื่องมือการบุกรุกเพื่อทดสอบการทำงานของโปรแกรมจริง โดยภาพรวมและเป็นการทดสอบในสภาพแวดล้อมจริง หลังจากนั้นจะเป็นการทดสอบการตัดสินผิดทางบวก และในส่วนสุดท้ายจะกล่าวถึงการทดสอบประสิทธิภาพการทำงานของระบบเมื่อมีการทำงานของระบบตรวจจับการบุกรุกอยู่เบื้องหลังเพื่อวิเคราะห์ผลกระทบที่โปรแกรมนี้อาจจะมีต่อระบบโดยรวมในเรื่องของเวลาทรัพยากรที่ถูกใช้

4.2 สภาพแวดล้อมในการทดสอบระบบ

ระบบตรวจจับการบุกรุกถูกพัฒนาขึ้นบนระบบปฏิบัติการ NetBSD 1.6X โดยมีรายละเอียดดังแสดงในตารางที่ 4.1

ตารางที่ 4.1 รายละเอียดระบบปฏิบัติการ NetBSD ที่ใช้ในการทดสอบระบบ

Machine Processor architecture name	i386
Node name	tualek
Operating system name	NetBSD
Operating system release	1.6X
Operating system version	NetBSD 1.6x (TUALEK) #18: Thu May 20 00:24:08 ICT 2004 add@tualek:/usr/src/sys/arch/i386/compile/TUALEK
CPU	Intel Pentium 4 (686-class) 2400.20 MHz
Total/avail memory	223 / 199 MB

ในส่วนถัดไปจะกล่าวถึงการทดสอบโปรแกรมซึ่งผู้วิจัยได้แบ่งการทดสอบออกเป็นสองส่วนคือในส่วนแรกเป็นการทดสอบที่ผู้วิจัยได้พัฒนาโปรแกรมขึ้นมาเองเพื่อใช้ในการทดสอบกฎที่ใช้สนับสนุนการตรวจจับในแต่ละข้อ และส่วนที่สองเป็นการทดสอบโดยใช้โปรแกรมการบุกรุกที่ได้จากอินเตอร์เน็ต รายละเอียดของการทดสอบทั้งสองส่วนจะกล่าวถึงในหัวข้อที่ 4.3 และ หัวข้อที่ 4.4 ตามลำดับดังนี้

4.3 การทดสอบการตรวจจับการบุกรุกตามเงื่อนไขสำหรับกฎแต่ละข้อ

ในการทดสอบการตรวจจับการบุกรุกสำหรับกฎแต่ละข้อ กระทำโดยการเขียนโปรแกรมการบุกรุกสำหรับเหตุการณ์ที่เข้าข่ายการบุกรุกตามเงื่อนไขของกฎแต่ละข้อขึ้นมาโดยแยกอธิบายได้ดังนี้

4.3.1 การทดสอบการตรวจจับการบุกรุกตามกฎข้อที่ 0

กฎข้อที่ 0 เมื่อโปรเซสอยู่ในสถานะสิทธิพิเศษอนุญาตให้ system call *setreuid()* และ *setregid()* เท่านั้นที่สามารถเปลี่ยน UID หรือ GID ได้ บนระบบปฏิบัติการยูนิกซ์มีคำสั่ง "su" ซึ่งเป็นคำสั่งที่ใช้เปลี่ยนสิทธิของผู้ที่เรียกใช้คำสั่งให้มีสิทธิเทียบเท่าสิทธิที่ถูกระบุ หรือในกรณีที่มีการเรียกใช้ "su" โดยไม่ระบุสิทธิที่ต้องการเทียบเท่าจะหมายถึงการเรียกใช้คำสั่ง "su" เพื่อให้มีสิทธิเทียบเท่า "root" นั่นหมายความว่าผู้ที่เรียกใช้คำสั่งนี้จะมีสิทธิในการเรียกใช้ทรัพยากรต่าง ๆ ในระบบ เนื่องจากในกระบวนการทำงานของคำสั่ง "su" มีการเรียกใช้ systemcall *setgid()* และ systemcall *setuid()* ผู้ใช้ที่ติดต่อเข้าใช้บริการหรือทรัพยากรของระบบไม่ว่าจากเครื่องใดก็สามารถที่จะเรียกใช้คำสั่งนี้ได้โดยไม่จำเป็นจะต้องนั่งใช้งานอยู่ที่เครื่องที่ให้บริการโดยตรง ฉะนั้นจะเห็นได้ว่าคำสั่งนี้เป็นคำสั่งที่อันตรายหากมีการนำไปใช้ในทางที่ผิด การป้องกันซึ่งสามารถป้องกันได้โดยกฎข้อที่ 0 นั่นคือในระบบที่คำนึงถึงเรื่องของการรักษาความปลอดภัยผู้ใช้จะไม่สามารถเรียกใช้คำสั่ง "su" เพื่อเปลี่ยนสิทธิของตัวเองให้เทียบเท่า root ได้ หากผู้ใช้ต้องการที่จะใช้บริการหรือทรัพยากรที่ต้องใช้สิทธิเทียบเท่า root ก็สามารถใช้ได้โดยการนั่งหน้าเครื่อง server แล้วล็อคอินเป็น root เพื่อเข้าใช้งานโดยตรง

<u>เหตุการณ์</u> : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) ใช้คำสั่ง su เพื่อเปลี่ยนสิทธิของตัวเองให้มีสิทธิการใช้งานเทียบเท่าผู้ใช้ที่มีสิทธิสูงสุด

ตัวบ่งชี้ : กระบวนการทำงานของคำสั่ง su มีการเรียกใช้ system call *setgid() setuid()* ในขณะที่ โปรเซสอยู่ในสถานะสิทธิพิเศษ

NetBSD 1.6X (TUALEK) #18: Thu May 20 00:24:08 ICT 2004

Welcome to NetBSD!

% su

Password:

NetBSD/i386 (tualek) (ttyE1)

Login:

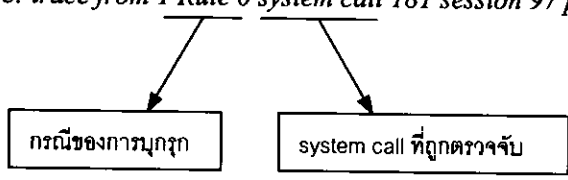
การตรวจจับ : หลังจากที่ผู้ใช้ ใช้คำสั่ง su เมื่อถึงขั้นตอนการใส่รหัสผ่านแล้วเคาะ Enter โปรเซสจะถูกหยุดการทำงานเนื่องจากตรวจพบว่ามีการเรียกใช้ system call *setgid()* ข้อมูลที่ได้จากการตรวจจับถูกบันทึกลง ล็อกแฟ้ม /var/log/ids.log โดยในส่วนแรกของแฟ้ม (บรรทัดที่ 1-9) แสดงถึงโปรเซสที่ถูกติดตาม หมายเลขโปรเซส หมายเลขโปรเซสแม่ หมายเลขกลุ่มโปรเซส หมายเลขเซสชันโปรเซส และหมายเลขของโปรเซสที่เป็นตัวติดตาม ซึ่งจะถูกบันทึกในครั้งแรกเมื่อโปรแกรมตรวจจับถูกเรียกใช้งาน ในส่วนถัดมา (บรรทัดที่ 10) แสดงรายละเอียดของโปรเซสที่ถูกตรวจจับ

Log file (/var/log/ids.log)

1 Aug 26 17:47:57 tualek idssysc: my own pid sid 399 12
2 Aug 26 17:47:57 tualek idssysc: pid 372 ppid 1 pgid 372 sid 372 trcpid 408
3 Aug 26 17:47:57 tualek idssysc: pid 319 ppid 1 pgid 319 sid 319 trcpid 409
4 Aug 26 17:47:57 tualek idssysc: pid 342 ppid 1 pgid 342 sid 342 trcpid 401
5 Aug 26 17:47:57 tualek idssysc: pid 304 ppid 1 pgid 304 sid 304 trcpid 405
6 Aug 26 17:47:57 tualek idssysc: pid 188 ppid 1 pgid 188 sid 188 trcpid 378
7 Aug 26 17:47:57 tualek idssysc: pid 157 ppid 1 pgid 157 sid 157 trcpid 370
8 Aug 26 17:47:57 tualek idssysc: pid 12 ppid 1 pgid 12 sid 12 trcpid 414
9 Aug 26 17:47:57 tualek idssysc: pid 1 ppid 0 pgid 1 sid 1 trcid 366

---

10 Aug 26 17:48:25 tualek idssysc: trace from 1 Rule 0 system call 181 session 97 pid 100 is killed



หมายเหตุ : จากการทดสอบการตรวจจับตามกฎข้อที่ 0 ทดสอบในกรณีที่ผู้ใช้เข้าใช้ระบบทั้งทางคอนโซล (console) และผ่านทางการใช้โปรแกรม *ssh*

4.3.2 การทดสอบการตรวจจับการบุกรุกตามกฎข้อที่ 1

กฎข้อที่ 1 เมื่อโปรเซสอยู่ในสถานะสิทธิพิเศษ ไม่อนุญาตให้มีการเรียกใช้ system call *execve()*

เหตุการณ์ : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้โปรแกรม rule1 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม (ดังแสดงด้านล่าง)โดยที่ตัวโปรแกรมนี้จะเรียกใช้ system call *execve()* เพื่อรันคำสั่งอื่น

ตัวบ่งชี้ : มีการเรียกใช้ system call *execve()* ในขณะที่โปรเซสอยู่ในสถานะสิทธิพิเศษ

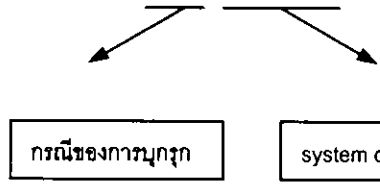
```
% cd ids
% cd testfile
% ls -la rule1
-r-sr-xr-x  1 root  users  5656 Aug 26 21:25 rule1
% ./rule1
Connection to tualek.eng.psu.ac.th closed.
```

ลักษณะโปรแกรมที่ใช้ทดสอบ

```
/* This program for test rule 1 */

#include <unistd.h>
int main() {
        char*  myarg[] = {"ls", "-l", (char *)0};

        execv("/bin/ls",myarg);
}
```

<u>การตรวจจับ</u> : หลังจากที่ผู้ใช้ เรียกใช้งานคำสั่ง rule1 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรมทำให้ในขณะที่โปรแกรมทำงาน โปรเซสจะอยู่ในสถานะของสิทธิพิเศษ setuid state ในขณะเดียวกันมีการเรียกใช้ ฟังก์ชัน execv() เพื่อรันคำสั่ง "/bin/ls" ในกระบวนการทำงานของคำสั่งฟังก์ชัน execv() นี้จะมีการเรียกใช้ system call *execve()* ดังนั้นโปรแกรมตรวจจับจะหยุดการทำงานของโปรเซสนั้นทำให้ไม่สามารถทำงานต่อได้ และบันทึกเหตุการณ์การบุกรุกนี้ในแฟ้มบันทึกข้อมูลการ บุกรุก ดังแสดงด้านล่าง

Aug 26 21:29:08 tualek idssysc: trace from 342 Rule 1 system call 59 session 829 pid 986 is killed



หมายเหตุ : จากการทดสอบการตรวจจับตามกฎข้อที่ 1 ทดสอบในกรณีที่ผู้ใช้เข้าใช้ระบบทั้งทาง console และผ่านทางการใช้โปรแกรม *ssh* ในตัวอย่างนี้เป็นการทดสอบโดยที่ผู้ใช้ล็อคอิน เข้าสู่ระบบโดยเรียกใช้โปรแกรม *ssh* เมื่อโปรเซสถูกหยุดการทำงานตัวโปรแกรมจะถูกตัดการติดต่อกับตัวเครื่องที่ผู้ใช้ล็อคอินเข้าไป

4.3.3 การทดสอบการตรวจจับการบุกรุกตามกฎข้อที่ 2

กฎข้อที่ 2 ขณะที่โปรเซสอยู่ในสถานะสิทธิพิเศษไม่อนุญาตให้มีการสร้าง setuid/setgid โปรแกรม อนุญาตเฉพาะ ผู้ใช้ที่มีสิทธิสูงสุด เท่านั้นสำหรับการทดสอบการบุกรุกตามกฎข้อที่ 2 แบ่งออกเป็น 4 เหตุการณ์ คือ

<u>เหตุการณ์ที่ 1</u> : ผู้ใช้ปกติ (uid=1000(add) gid=100(user)) เรียกใช้โปรแกรม rule2 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม (ดังแสดงด้านล่าง) โดยการทำงานของโปรแกรมนี้จะเรียกใช้คำสั่ง creat() เพื่อสร้างแฟ้มโดยกำหนดโหมดเป็น 04000 ซึ่งเป็นโหมดในการสร้าง setuid โปรแกรม

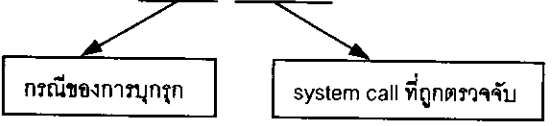
```
% cd ids
% cd testfile
% ls -la rule2
-r-sr-xr-x  1 root  users  5746 Aug 26 22:01 rule2
% ./rule2
Connection to tualek.eng.psu.ac.th closed
```

ลักษณะโปรแกรมที่ใช้ทดสอบ

```
int main() {
    int fd;
    fd=creat("newfile.txt", 04000);
        close(fd);
    return 0;
}
```

การตรวจจับ : หลังจากที่ผู้ใช้ เรียกใช้งานโปรแกรม rule2 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม นั่นคือในขณะที่โปรแกรมทำงานโปรเซสจะอยู่ในสถานะของสิทธิพิเศษ setuid state ในขณะเดียวกันมีการเรียกใช้ฟังก์ชัน creat() ซึ่งในกระบวนการทำงานของฟังก์ชัน creat() มีการเรียกใช้system call *open()* ในการสร้างแฟ้มและมีการกำหนดโหมดเป็น 04000 หรือ S_ISUID ซึ่งเป็นการกำหนดโหมดเพื่อให้แฟ้มนั้นเป็นแฟ้มประเภท setuid root โปรแกรม ดังนั้น เมื่อการตรวจจับพบว่ามีการกำหนดโหมดเช่นนี้ ตัวโปรแกรมจึงหยุดการทำงานของโปรเซสและบันทึกผลของการตรวจจับ ดังแสดงด้านล่าง

Aug 26 22:03:32 tualek idssysc: trace from 342 Rule 21 system call 5 session 927 pid 1102 is killed



หมายเหตุ : Rule 21 ในที่นี้หมายถึงเข้าข่ายการบุกรุกตามเงื่อนไขในกฎข้อที่ 2 กรณีที่ 1 นั่นคือเรียกใช้ system call *open()* มีกำหนดค่าแฟลกเพื่อสร้างแฟ้มและมีการกำหนดโหมด S_ISUID

เหตุการณ์ที่ 2 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้โปรแกรมrule22 ซึ่งเป็นโปรแกรมประเภท setgid โปรแกรมโดยการทำงานของโปรแกรมนี้จะเรียกใช้คำสั่ง creat เพื่อสร้างแฟ้มโดยกำหนดโหมดเป็น 02000

```
% cd ids
% cd testfile
% ls -la rule22
-r-sr-xr-x  1 root  users  5746 Aug 26 22:01 rule22
% ./rule22
Connection to tualek.eng.psu.ac.th closed.
```

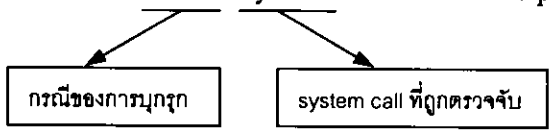
ลักษณะโปรแกรมที่ใช้ทดสอบ

```
int main() {
    int fd;
    fd=creat("newfile.txt", 02000);
        close(fd);
    return 0;
}
```

การตรวจจับ : หลังจากที่ผู้ใช้เรียกใช้งานโปรแกรม rule22 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม นั่นคือในขณะที่โปรแกรมทำงานโปรเซสจะอยู่ในสถานะของสิทธิพิเศษ setuid state ในขณะเดียวกันมีการเรียกใช้ฟังก์ชัน creat() ซึ่งในกระบวนการทำงานของฟังก์ชัน creat() มีการเรียกใช้system call *open()* ในการสร้างแฟ้มและมีการกำหนดโหมดเป็น 02000 หรือ S_ISGID ซึ่งเป็นการกำหนดโหมดเพื่อให้แฟ้มนั้นเป็นแฟ้มประเภท setgid โปรแกรม ดังนั้นเมื่อ

การตรวจจับพบว่ามีการกำหนดโหมดเช่นนี้ ตัวโปรแกรมจึงหยุดการทำงานของโปรเซสและบันทึกผลของการตรวจจับ ดังแสดงด้านล่างนี้

Aug 26 23:37:31 tualek idssysc: trace from 342 Rule 22 system call 5 session 927 pid 1377 is killed



หมายเหตุ : Rule 22 ในที่นี้หมายถึงเข้าข่ายการบุกรุกตามเงื่อนไขในกฎข้อที่ 2 กรณีที่ 2 นั่นคือเรียกใช้ system call *open()* มีกำหนดค่าแฟลกเพื่อสร้างแฟ้ม และมีการกำหนดโหมด S_ISGID

<u>เหตุการณ์ที่ 3</u> : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้โปรแกรมrule23 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม โดยตัวโปรแกรมจะเรียกใช้ system call *chmod()* เพื่อเปลี่ยนโหมดของแฟ้มเป็น S_ISUID โหมด

$ cd ids

$ cd testfile

$ ls -la rule23

-r-sr-xr-x 1 root users 5625 Aug 27 16:45 rule23

$ ./rule23

Killed

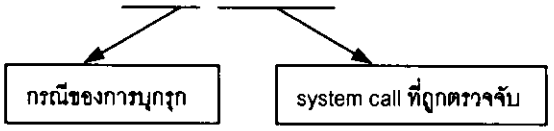
Connection to tualek.eng.psu.ac.th closed.

<u>ลักษณะโปรแกรมที่ใช้ทดสอบ</u>

```
int main() {
    int fd;
    chmod("newfile.txt",S_IRUSR|S_IXUSR|S_IXGRP|S_IXOTH|S_ISUID);
    return 0;
}
```

การตรวจจับ : หลังจากที่ผู้ใช้เรียกใช้งานโปรแกรม rule23 ซึ่งเป็นโปรแกรมประเภท setuid rootโปรแกรม นั่นคือในขณะที่โปรแกรมทำงานโปรเซสจะอยู่ในสถานะของสิทธิพิเศษ setuid state ในขณะเดียวกันมีการเรียกใช้ system call *chmod()* และมีการกำหนดโหมดเป็น S_ISUID ซึ่งเป็นการกำหนดโหมดเพื่อให้แฟ้มนั้นเป็นแฟ้มประเภท setuid root โปรแกรม ดังนั้น เมื่อการตรวจจับพบว่ามีการกำหนดโหมดเช่นนี้ ตัวโปรแกรมจึงหยุดการทำงานของโปรเซสและบันทึกผลของการตรวจจับ ดังแสดงด้านล่างนี้

Aug 27 16:48:24 tualek idssysc: trace from 342 Rule 23 system call 15 session 2068 pid 3469 is killed



หมายเหตุ : Rule 23 ในที่นี้หมายถึงเข้าข่ายการบุกรุกตามเงื่อนไขในกฎข้อที่ 2 กรณีที่ 3 นั่นคือเรียกใช้ system call *chmod()* และมีการกำหนดค่าโหมด S_ISUID

เหตุการณ์ที่ 4 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้โปรแกรมrule24 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม โดยตัวโปรแกรมจะเรียกใช้ system call *chmod()* ซึ่งมีการกำหนดโหมดของแฟ้มเป็น S_ISGID โหมด

$ cd ids

$ cd testfile

$ ls -la rule24

-r-sr-xr-x 1 root users 5625 Aug 27 16:45 rule24

$ ./rule24

Killed

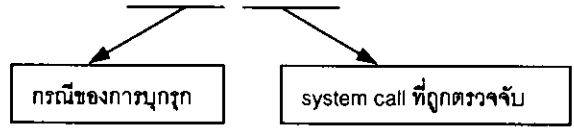
Connection to tualek.eng.psu.ac.th closed.

ลักษณะโปรแกรมที่ใช้ทดสอบ

```
int main() {
    int fd;
    chmod("newfile.txt",S_IRUSR|S_IXUSR|S_IXGRP|S_IXOTH|S_ISGID);
    return 0;
}
```

การตรวจจับ : หลังจากที่ผู้ใช้เรียกใช้งานโปรแกรม rule24 ซึ่งเป็นโปรแกรมประเภท setuid rootโปรแกรม นั่นคือในขณะที่โปรแกรมทำงานโปรเซสจะอยู่ในสถานะของสิทธิพิเศษ setuid state ในขณะเดียวกันมีการเรียกใช้ system call *chmod()* และมีการกำหนดโหมดเป็น S_ISGID ซึ่งเป็นการกำหนดโหมดเพื่อให้แฟ้มนั้นเป็นแฟ้มประเภท setgid โปรแกรม ดังนั้นเมื่อการตรวจจับพบว่ามีการกำหนดโหมดเช่นนี้ ตัวโปรแกรมจึงหยุดการทำงานของโปรเซสและบันทึกผลของการตรวจจับ ดังแสดงด้านล่างนี้

Aug 27 17:00:27 tualek idssysc: trace from 342 Rule 24 system call 15 session 3497 pid 4030 is killed



หมายเหตุ : Rule 24 ในที่นี้หมายถึงเข้าข่ายการบุกรุกตามเงื่อนไขในกฎข้อที่ 2 กรณีที่ 4 นั่นคือเรียกใช้ system call *chmod()* และมีการกำหนดค่าโหมด S_ISGID

4.3.4 การทดสอบการบุกรุกตามกฎข้อที่ 3

กฎข้อที่ 3 ไม่อนุญาตให้โปรเซสแก้ไขโปรแกรมระบบ

เหตุการณ์ที่1 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้โปรแกรม rule31 ซึ่งเป็น setuid rootโปรแกรม การทำงานของโปรแกรมมีการเรียกใช้ system call *open()* เพื่อแก้ไขแฟ้ม /usr/src/sys/kern/testrule3.txt ซึ่ง /usr/src/sys/kern/ ถูกกำหนดเป็นที่อยู่ของ system file โดยมีการเรียกใช้แฟ้มในลักษณะของ *absolute path*

open("/usr/src/sys/kern/testrule3.txt",O_WRONLY)

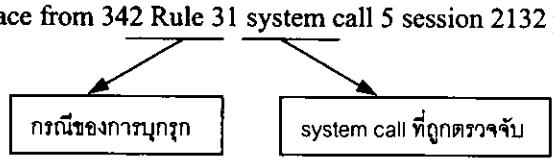
ลักษณะโปรแกรมที่ใช้ทดสอบ

```
int main() {
    int fd;
    fd=open("/usr/src/sys/kern/testrule3.txt", O_WRONLY);
    close(fd);
    return 0;
}
```

การตรวจจับ : หลังจากที่ผู้ใช้เรียกใช้งานโปรแกรม rule31 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม นั่นคือในขณะที่โปรแกรมทำงานโปรเซสจะอยู่ในสถานะของสิทธิพิเศษ setuid state ในขณะเดียวกันมีการเรียกใช้ system call *open()* เปิดแฟ้ม "/usr/src/sys/kern/testrule3.txt" ซึ่งอยู่ในไดเรกทอรีที่ถูกกำหนดให้ทุกแฟ้มที่อยู่ในไดเรกทอรีนั้นเป็นแฟ้มระบบ และมีการเรียกใช้แฟลก "O_WRONLY" ดังนั้นเมื่อการตรวจจับพบว่ามีการเรียกใช้แฟ้มจากไดเรกทอรีที่ถูกกำหนดเพื่อทำการแก้ไขจึงถือว่าเข้าข่ายการบุกรุก ตัวโปรแกรมจึงหยุดการทำงานของโปรเซสและบันทึกผลของการตรวจจับ

```
$ cd ids
$ cd testfile
$ ls -la rule31
-r-sr-xr-x  1 root  users  5625 Aug 27 16:45 rule31
$ ./rule31
Killed
Connection to tualek.eng.psu.ac.th closed.
```

Aug 27 17:25:29 tualek idssysc: trace from 342 Rule 31 system call 5 session 2132 pid 356 is killed



หมายเหตุ : Rule 31 ในที่นี้หมายถึงเข้าข่ายการบุกรุกตามเงื่อนไขในกฎข้อที่ 3 กรณีที่ 1 นั่นคือเรียกใช้ system call *open()* และมีการกำหนดค่าแฟลกเป็น O_WRONLY

เหตุการณ์ที่ 2 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้โปรแกรม r3symb ซึ่งเป็น setuid root โปรแกรม การทำงานของโปรแกรมมีการเรียกใช้ system call *open()* เพื่อแก้ไขแฟ้ม ./usr_sys/testrule3.txt ซึ่งเป็นแฟ้มที่เชื่อมโยงไปยัง /usr/src/sys/kern/testrule3.txt โดยมีการอ้างถึงแฟ้มแบบ *relative path*

fopen("./usr_sys/kern/testrule3.txt","ORDWR")

ลักษณะโปรแกรมที่ใช้ทดสอบ

```
int main() {
    int fd;
    fd=open("./usr_sys/kern/testrule3.txt", O_RDWR);
    close(fd);
    return 0;
}
```

การตรวจจับ : หลังจากที่ผู้ใช้เรียกใช้งานโปรแกรม r3symb ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม นั่นคือในขณะที่โปรแกรมทำงานโปรเซสจะอยู่ในสถานะของสิทธิพิเศษ setuid state ในขณะเดียวกันมีการเรียกใช้ system call *open()* เปิดแฟ้ม "./usr_sys/kern/testrule3.txt" ซึ่งเป็นแฟ้มที่เชื่อมโยงไปยัง /usr/src/sys/kern/testrule3.txt ซึ่งอยู่ในไดเรกทอรีที่ถูกกำหนดให้ทุกแฟ้มที่อยู่ในไดเรกทอรีนั้นเป็นแฟ้มระบบโดยมีการอ้างถึงแฟ้มแบบ *relative path* และมีการเรียกใช้แฟลก "O_RDWR" ดังนั้นเมื่อการตรวจจับพบว่ามีการเรียกใช้แฟ้มจากไดเรกทอรีที่ถูกกำหนดเพื่อทำการแก้ไขจึงถือว่าเข้าข่ายการบุกรุก ตัวโปรแกรมจึงหยุดการทำงานของโปรเซสและบันทึกผลของการตรวจจับในแฟ้มบันทึกข้อมูลการบุกรุก

$ cd ids

$ cd testfile

$ ls -la r3symb

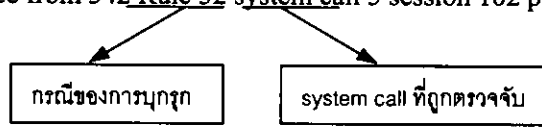
-r-sr-xr-x 1 root users 5746 Aug 29 18:33 r3symb

$ ./r3symb

Killed

Connection to tualek.eng.psu.ac.th closed.

Sep 1 23:00:12 tualek idssysc: trace from 342 Rule 32 system call 5 session 102 pid 467 is killed



หมายเหตุ : Rule 32 ในที่นี้หมายถึงเข้าข่ายการบุกรุกตามเงื่อนไขในกฎข้อที่ 3 กรณีที่ 2 นั่นคือเรียกใช้ system call *open()* และมีการกำหนดค่าแฟลกเป็น O_RDWR

การตรวจจับ : หลังจากที่ผู้ใช้เรียกใช้งานโปรแกรม rule31 หรือ r3symb ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรม นั่นคือในขณะที่โปรแกรมทำงานโปรเซสจะอยู่ในสถานะของสิทธิพิเศษ setuid state ในขณะเดียวกันมีการเรียกใช้ system call *open()* เพื่อแก้ไขแฟ้มซึ่งอยู่ในไดเรกทอรีที่ถูกกำหนดให้ทุกแฟ้มที่อยู่ในไดเรกทอรีนั้นเป็นแฟ้มระบบ ดังนั้นเมื่อการตรวจจับพบว่ามีการเรียกใช้แฟ้มจากไดเรกทอรีที่ถูกกำหนดเพื่อทำการแก้ไขเข้าข่ายการบุกรุก ตัวโปรแกรมจึงหยุดการทำงานของโปรเซสและบันทึกผลของการตรวจจับ

4.3.5 การทดสอบการตรวจจับการบุกรุกตามกฎข้อที่ 4

กฎข้อที่ 4 ผู้ใช้ที่มีสิทธิสูงสุดเท่านั้นที่มีสิทธิในการสร้างบัญชีผู้ใช้ใหม่ได้

เหตุการณ์ : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้โปรแกรม rule4 ซึ่งเป็นโปรแกรมประเภท setuid root โปรแกรมโดยการทำงานของโปรแกรมนี้จะเรียกใช้ system call *opem()* เพื่อเปิดแฟ้ม /etc/master.passwd

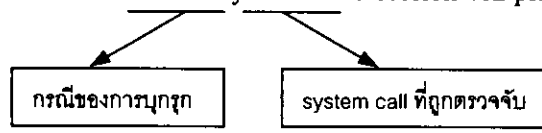
```
% cd ids
% cd testfile
% ./rule4
Connection to tualek.eng.psu.ac.th closed.
```

ลักษณะโปรแกรมที่ใช้ทดสอบ

```
int main()
{
  f = fopen("/etc/master.passwd","r");
  fclose(f);
}
```

การตรวจจับ : หลังจากผู้ใช้เรียกใช้งานโปรแกรม rule4 และตรวจจับพบว่ามีการเรียกใช้ system call *open()* โดยแฟ้มที่ถูกเรียกใช้คือ /etc/master.passwd ซึ่งเป็นแฟ้มสำคัญที่ถูกเรียกใช้เมื่อมีการสร้างรายชื่อผู้ใช้ใหม่ ในระบบ โปรเซสที่เข้าข่ายการบุกรุกนั้นจะถูกหยุดการทำงาน และถูกบันทึกรายละเอียดลงในแฟ้มบันทึกข้อมูลการบุกรุกดังแสดงด้านล่าง

```
Sep  2 05:08:49 tualek idssysc: trace from 342 Rule 4 system call 5 session 462 pid 568 is killed
```



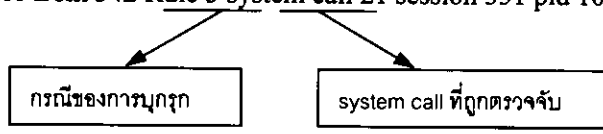
4.3.6 การทดสอบการตรวจจับการบุกรุกตามกฎข้อที่ 5

กฎข้อที่ 5 เมื่อโปรเซสอยู่ในสถานะที่มีสิทธิพิเศษไม่อนุญาตให้มีการเรียกใช้ system call *mount(), unmount(), nfssvc(), quotactl(), reboot(), settimeofday(), swapon()* ซึ่งสามารถเรียกใช้ได้โดยผู้ใช้ที่มีสิทธิสูงสุดเท่านั้นสำหรับการทดสอบการตรวจจับการบุกรุกสำหรับกฎข้อที่ 5 จะแบ่งการทดสอบออกตามเหตุการณ์การเรียกใช้ system call แต่ละตัว

<u>เหตุการณ์ที่ 1</u> : ผู้ใช้ซึ่งมีค่า EUID เท่ากับ 0 เรียกใช้คำสั่ง mount เพื่อทำการเรียกใช้งาน floppy disk ในการจำลองเหตุการณ์นี้ได้ทดลองโดยการใช้คำสั่ง "su" เพื่อให้ผู้ใช้ในขณะนั้นมีค่า EUID เท่ากับ 0 หลังจากนั้นเรียกใช้ คำสั่ง mount ดังนี้

mount –t msdos /dev/fd0a /floppy

Nov 10 21:49:41 tualek idssysc: trace from 342 Rule 5 system call 21 session 351 pid 168 is killed

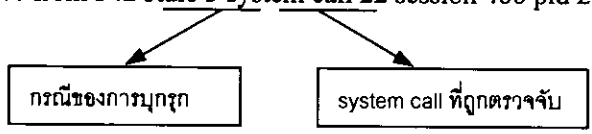


<u>การตรวจจับ</u>: เมื่อมีการเรียกใช้คำสั่ง mount เพื่อจะใช้งาน floppy disk ระบบตรวจจับพบว่ามีการเรียกใช้ system call *mount()* ดังนั้นกระบวนการในการเรียกใช้งาน floppy disk จึงถูกระงับเนื่องจากในขณะนั้นโปรเซสมีค่า EUID เท่ากับ 0 ซึ่งถือว่าอยู่ในสถานะสิทธิพิเศษ เป็นการตรวจจับตามกฎที่ใช้สนับสนุนการตรวจจับในข้อที่ 5

<u>เหตุการณ์ที่ 2</u> : ผู้ใช้ซึ่งมีค่า EUID เท่ากับ 0 เรียกใช้คำสั่ง umount เพื่อยกเลิกการเรียกใช้งาน floppy disk ในการจำลองเหตุการณ์นี้ได้ทดลองโดยการใช้คำสั่ง "su" เพื่อให้ผู้ใช้ในขณะนั้นมีค่า EUID เท่ากับ 0 หลังจากนั้นเรียกใช้ คำสั่ง umount ดังนี้

umount /floppy

Nov 10 21:50:57 tualek idssysc: trace from 342 Rule 5 system call 22 session 466 pid 204 is killed

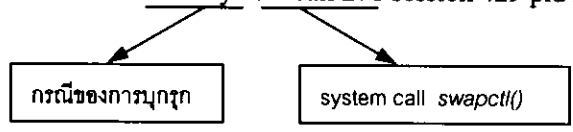


<u>การตรวจจับ</u>: เมื่อมีการเรียกใช้คำสั่ง umount เพื่อจะใช้งาน floppy disk ระบบตรวจจับพบว่ามีการเรียกใช้ system call *umount()* ดังนั้นกระบวนการในการยกเลิกการใช้งาน floppy disk จึงถูกระงับเนื่องจากในขณะนั้นโปรเซสมีค่า EUID เท่ากับ 0 ซึ่งถือว่าอยู่ในสถานะสิทธิพิเศษ เป็นการตรวจจับตามกฎที่ใช้สนับสนุนการตรวจจับในข้อที่ 5

<u>เหตุการณ์ที่ 3</u> : ผู้ใช้ซึ่งมีค่า EUID เท่ากับ 0 เรียกใช้คำสั่ง swapctl เพื่อเอารายการเส้นทางอุปกรณ์ swap ออกจากรายชื่อในเคอเนล ในการจำลองเหตุการณ์นี้ได้ทดลองโดยการใช้คำสั่ง "su" เพื่อให้ผู้ใช้ในขณะนั้นมีค่า EUID เท่ากับ 0 หลังจากนั้นเรียกใช้ คำสั่ง swapctl ดังนี้

```
# swapctl –d /etc
```

Nov 10 22:07:02 tualek idssysc: trace from 342 Rule 5 system call 271 session 429 pid 230 is killed



<u>การตรวจจับ</u>: เมื่อมีการเรียกใช้คำสั่ง swapctl เพื่อยกเลิกเส้นทาง /etc ของอุปกรณ์ swap ระบบตรวจจับพบว่ามีการเรียกใช้ system call *swapctl()* ดังนั้นกระบวนการในการยกเลิกเส้นทางของอุปกรณ์ swap จึงถูกระงับเนื่องจากในขณะนั้นโปรเซสมีค่า EUID เท่ากับ 0 ซึ่งถือว่าอยู่ในสถานะสิทธิพิเศษ เป็นการตรวจจับตามกฎที่ใช้สนับสนุนการตรวจจับในข้อที่ 5

4.4 การทดสอบการบุกรุกโดยใช้โปรแกรมการบุกรุกจากอินเตอร์เน็ต

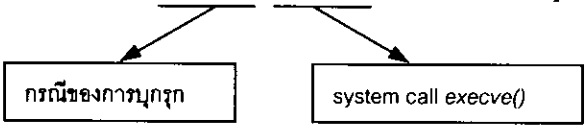
สำหรับในส่วนนี้จะกล่าวถึงการทดสอบและผลจากการทดสอบระบบตรวจจับการบุกรุกโดยใช้โปรแกรมการบุกรุกซึ่งได้มาทางอินเตอร์เน็ต โดยที่ลักษณะการทำงานโดยรวมของโปรแกรมการบุกรุกเหล่านี้จะอาศัยจุดอ่อนในเรื่องของการใช้ buffer หรือการโจมตีประเภท buffer overflow ซึ่งส่วนมากจะเป็นโปรแกรมประะเภท setuid root

<u>เหตุการณ์ที่1</u> : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม libc-language_su.c ซึ่งเป็นโปรแกรมที่พยายามจะรันคำสั่ง "/bin/su" เพื่อให้ได้สิทธิเทียบเท่าผู้ใช้ที่มีสิทธิสูงสุด

```
$ ./libc-language_su
glibc xploit for /bin/su - by Doing <jdoing@bigfoot.com>

Usage: ./libc-language_su [options]
-o offset [default: 5000]
-n nops   [default: 12000]
-m path to msgfmt [default: /usr/bin/msgfmt]
-O path to objdump [default: /usr/bin/objdump]
-e eat:pad set eat and pad values [default: calculate them]
-l language set language used in env var [default: search it]
Enjoy!

Phase 1. Checking paths and write permisions
Checking for /usr/bin/msgfmt...Ok
Checking for /usr/bin/objdump...Ok
Checking write permisions on /tmp...Ok
Checking read permisions on /bin/su...Ok
Checking for a valid language... [using bg_BG.CP1251] Ok
Checking that /tmp/LC_MESSAGES does not exist...Ok
Phase 2. Calculating eat and pad values

Killed
Connection to tualek.eng.psu.ac.th closed.
```

ผลของตัวโปรแกรมแสดงวิธีการใช้งาน และการทำงานของโปรแกรมในแต่ละขั้นโปรแกรม

Sep 3 04:53:39 tualek idssysc: trace from 342 Rule 1 system call 59 session 1804 pid 1917 is killed



<u>การตรวจจับ</u> :จากตัวอย่างด้านบน โปรแกรมบุกรุกจะทำงานจนกระทั่งมีการเรียกใช้ system call *execve()* เพื่อรัน "/bin/su" โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

เหตุการณ์ที่2 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม local.c ซึ่งเป็นโปรแกรมที่มีการทำงานคล้ายกับโปรแกรม libc-language_su.c นั่นคือพยายามจะใช้งานคำสั่ง "/bin/su"

$ ./local

Usages: ./local <RETloc> <offset> <num> <align> <buffsize> <eggsize>

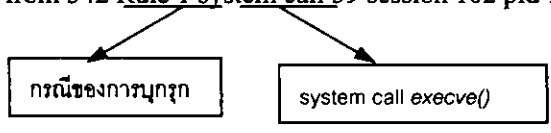
Using RET location address: 0xbfffd250

Using Shellcode address: 0xbfbffd3e

Killed

Connection to tualek.eng.psu.ac.th closed.

Sep 3 05:29:21 tualek idssysc: trace from 342 Rule 1 system call 59 session 102 pid 107 is killed



การตรวจจับ :จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call *execve()* เพื่อรันคำสั่ง "/bin/su" โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

เหตุการณ์ที่3 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม sortrace.c ซึ่งภายในแกรมบุรุก sortace.c จะมีการเรียกใช้ system call *execve()* เพื่อเรียกใช้งาน traceroute ซึ่งผิดกฎข้อที่ 1 ซึ่งไม่อนุญาตให้โปรเซสในสถานะสิทธิพิเศษเรียกใช้ system call *execve()* ลักษณะการทำงานและตัวอย่างโปรแกรมแสดงในภาคผนวก ก.

$ ./sortrace

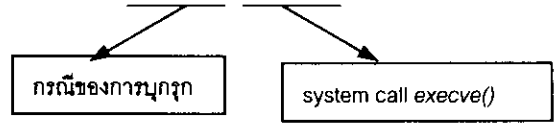
Traceroute v1.4a5 exploit by sorbo (sorbox@yahoo.com)

Checking vuln...

Killed

% Connection to tualek.eng.psu.ac.th closed.

Sep 3 05:34:46 tualek idssysc: trace from 342 Rule 1 system call 59 session 402 pid 253 is killed



จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call *execve()* เพื่อรันคำสั่ง /usr/sbin/traceroute" โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

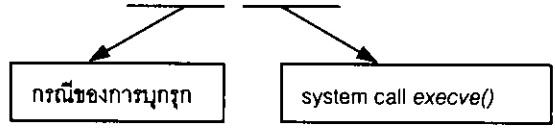
<u>เหตุการณ์ที่4</u> : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม cwho ซึ่งเป็นโปรแกรมที่พยายามจะรันโปรแกรม /bin/sh เพื่อให้ได้เชลล์พร้อมท์ของผู้ใช้ที่มีสิทธิสูงสุด ซึ่งทำให้ผู้บุกรุกสามารถใช้งานคำสั่งได้เสมือนเป็นผู้ใช้สูงสุด การทำงานและลักษณะของโปรแกรมจะแสดงในภาคผนวก ก.

$ ./cwho -u add

Killed

Connection to tualek.eng.psu.ac.th closed.

Sep 3 05:40:01 tualek idssysc: trace from 342 Rule 1 system call 59 session 164 pid 140 is killed



<u>การตรวจจับ</u> : จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call *execve()* เพื่อรันคำสั่ง /bin/sh/" โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

<u>เหตุการณ์ที่2</u> : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม crnovf ซึ่งเป็นโปรแกรมที่พยายามจะรันโปรแกรม /usr/bin/crontab ในขณะที่โปรเซส อยู่ในสถานะสิทธิพิเศษ

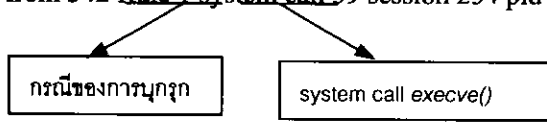
$ ./crnovf

Using offset (0xc7c48508)

Killed

% Connection to tualek.eng.psu.ac.th closed.

Sep 3 05:49:36 tualek idssysc: trace from 342 Rule 1 system call 59 session 234 pid 227 is killed

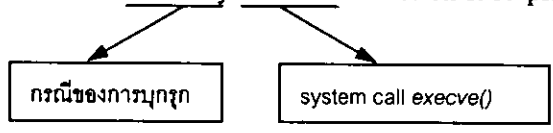


การตรวจจับ : จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call *execve()* เพื่อรันคำสั่ง "/usr/bin/crontab" โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

เหตุการณ์ที่5 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม rdist ซึ่งเป็นโปรแกรมที่พยายามจะรันโปรแกรอื่นในขณะที่โปรเซสอยู่ในสถานะสิทธิพิเศษ

```
$ ./rdist
Using offset of esp + 50 (bfbffb1a)
Killed
% Connection to tualek.eng.psu.ac.th closed.
```

Sep 15 04:17:08 tualek idssysc: trace from 342 Rule 1 system call 59 session 1959 pid 2700 is killed

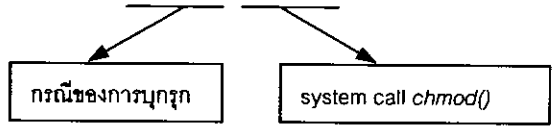


การตรวจจับ : จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call *execve()* เพื่อรันคำสั่ง "/usr/bin/rdist" โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

เหตุการณ์ที่6 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม chmod_pw เพื่อพยายามจะเปลี่ยนแปลงโหมดของแฟ้ม /etc/passwd ให้เป็นโหมด 777 ซึ่งอนุญาตให้ทุกคนเขียนลงแฟ้มได้ ลักษณะการทำงานและตัวอย่างโปรแกรม แสดงไว้ในภาคผนวก ก.

```
$ cd program_bug/
$ ./chmod_pw
Killed
Connection to tualek.eng.psu.ac.th closed.
```

Sep 3 05:52:31 tualek idssysc: trace from 342 Rule 23 system call 15 session 574 pid 633 is killed



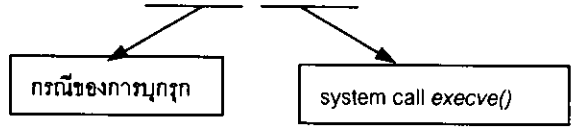
การตรวจจับ : จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call chmod() เพื่อเปลี่ยนโหมดเป็น 777 โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 2 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

เหตุการณ์ที่7 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม 4man.c ซึ่งมีการเรียกใช้ ./man ในโปรแกรม นั่นคือมีการเรียกใช้ system call *execve()*

ลักษณะการทำงานและตัวอย่างโปรแกรมแสดงในภาคผนวก ก.

$ ./4man
RET: 0xbfbff260 len: 4076
Killed
Connection to tualek.eng.psu.ac.th closed.

Sep 3 05:56:18 tualek idssysc: trace from 342 Rule 1 system call 59 session 625 pid 464 is killed



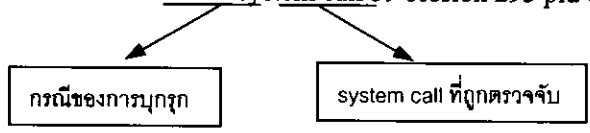
การตรวจจับ : จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call *execve()* เพื่อเรียกใช้./man ในขณะที่โปรแซสอยู่ในสถานะสิทธิพิเศษ โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

เหตุการณ์ที่8 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม umnt ซึ่งตัวโปรแกรมจะค้นหาตำแหน่งใน process stack ต่อจากนั้นจะทำการเพิ่มโค้ดเพื่อเรียก /bin/sh และเรียกใช้คำสั่ง umount

---

$ cd program_bug/
$ ./umnt
Killed
Connection to tualek.eng.psu.ac.th closed.

Sep 3 06:10:47 tualek idssysc: trace from 342 Rule 1 system call 59 session 293 pid 606 is killed



การตรวจจับ : จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call execve() เพื่อเรียกใช้คำสั่ง umount ในขณะที่โปรแซสอยู่ในสถานะสิทธิพิเศษ โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

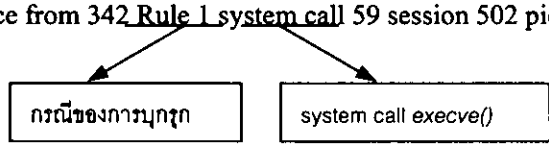
เหตุการณ์ที่9 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม bsd_lpr_exploit.c ซึ่งเป็นโปรแกรมที่มีการเรียกใช้คำสั่ง "/usr/bin/lpr" โดยรายละเอียดของโปรแกรมจะนำเสนอในภาคผนวก ก.

$ ./bsd_lpr_exploit

Killed

Connection to tualek.eng.psu.ac.th closed.

Sep 3 05:45:32 tualek idssysc: trace from 342 Rule 1 system call 59 session 502 pid 527 is killed



การตรวจจับ : จากผลการทดสอบโปรแกรมเมื่อโปรแกรมบุกรุกมีการเรียกใช้ system call execve() เพื่อเรียกใช้คำสั่ง /usr/bin/lpr ในขณะที่โปรแซสอยู่ในสถานะสิทธิพิเศษ โปรแกรมจะถูกตรวจจับตามกฎข้อที่ 1 และหยุดการทำงานของโปรแกรมนี้ไม่ให้ทำงานต่อได้สำเร็จ

เหตุการณ์ที่10 : ผู้ใช้ปกติ (uid=1000(add) gid=100(users) groups=100(users)) เรียกใช้งานโปรแกรม trojan_suid_shell ซึ่งพยายามจะสร้าง SUID root shell ในสถานะสิทธิพิเศษ

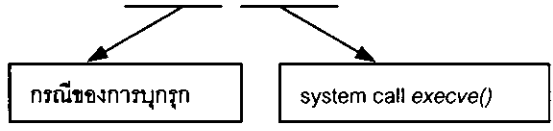
$ cd trojan/

$ ./trojan_suid_shell industry3

Killed

Connection to tualek.eng.psu.ac.th closed.

Sep 10 22:06:07 tualek idssysc: trace from 342 Rule 1 system call 59 session 6636 pid 3610 is killed



4.5 ผลจากการทดสอบการตรวจจับการบุกรุก

จากการทดสอบการตรวจจับการบุกรุกโดยโปรแกรมการบุกรุกสามารถนำมาสรุป เป็นตารางแสดงจำนวนครั้งที่ทดสอบ และจำนวนครั้งที่สามารถตรวจจับได้ดังตารางที่ 4.2 ดังนี้

ตารางที่ 4.2 แสดงผลจากการทดสอบการตรวจจับการบุกรุก

เครื่องมือ	จำนวนครั้งที่ทดสอบ	จำนวนครั้งที่จับได้	จำนวนครั้งที่จับไม่ได้
libc-language_su.c	10	10	0
local.c	10	10	0
Sortrace.c	10	10	0
cwho.c	10	10	0
crontab_overflow.c	10	10	0
rdist	10	10	0
chmod_pw.c	10	10	0
4man.c	10	10	0
bsd_lpr_exploit.c	10	10	0
umnt.c	10	10	0
trojan_suid_shell	10	10	0
รวม	**110**	**110**	**0**

4.6 การทดสอบการตัดสินผิดทางบวก (false positive)

สำหรับการทดสอบการตัดสินใจผิดทางบวก ได้เลือกการทดสอบจากการเรียกใช้คำสั่งปกติในขณะที่ระบบตรวจจับการบุกรุกทำงานอยู่เบื้องหลังกระทำโดยทดลองสุ่มคำสั่งที่ใช้งานอยู่บนระบบซึ่งเป็นคำสั่งที่ไม่เข้าข่ายการบุกรุก ผลการทดสอบดังแสดงในตารางที่ 4.3

ผลการตรวจจับ : จากข้อมูลในตารางจะเห็นได้ว่าในขณะที่โปรแกรมการตรวจจับการบุกรุกทำงานอยู่เบื้องหลัง การเรียกใช้คำสั่งในระบบจะไม่ได้รับผลกระทบจากตัวโปรแกรม นั่นคือคำสั่งที่ไม่เข้าข่ายการบุกรุกจะยังใช้งานได้ตามปกติ คิดเป็นร้อยละ 100

ตารางที่ 4.3 แสดงผลการทดสอบการเรียกใช้งานคำสั่งในระบบ

ลำดับที่	คำสั่ง	*จำนวนครั้งที่ทดสอบ	จำนวนครั้งที่เกิด False Positive
1	ls	30	0
2	passwd	20	0
3	ps	30	0
4	gcc	20	0
5	compile kernel	5	0
6	login	20	0
7	tar	20	0
8	sshd	20	0
9	ftpd	20	0
10	man	20	0
11	rm	20	0
12	cp	30	0
13	startx	20	0
14	find	30	0
15	ktrace	20	0
16	ssh	20	0
17	ftp	20	0
18	vi	30	0
19	who	20	0
20	ktruss	20	0
รวม		**435**	**0**

4.7 การทดสอบประสิทธิภาพของระบบ

การทดสอบประสิทธิภาพของระบบในที่นี้จะแบ่งเป็นการทดสอบในส่วนของเวลาที่ใช้และทรัพยากรที่ถูกใช้เมื่อมีการใช้งานระบบตรวจจับการบุกรุก

4.7.1 เวลาที่ใช้

เวลาที่ใช้ในที่นี้หมายถึงเวลาที่โปรแกรมในระบบใช้ตั้งแต่เริ่มทำงานจนจบคำสั่ง เปรียบเทียบเมื่อมีการใช้งานโปรแกรมตรวจจับการบุกรุกอยู่เบื้องหลัง และเมื่อไม่มีการใช้ระบบตรวจจับการบุกรุก ในการทดสอบได้เลือกใช้การคอมไพล์เคอเนลเป็นตัวเปรียบเทียบเนื่องจากเวลาที่ใช้ในการคอมไพล์เคอเนลใช้เวลาค่อนข้างนานทำให้เห็นความแตกต่างในการใช้เวลาได้ชัดเจน ผลแสดงดังตารางที่ 4.4

ตารางที่ 4.4 ตารางเปรียบเทียบเวลาที่ใช้ในการคอมไพล์เคอร์เนล

ครั้งที่	เวลาที่ใช้		
	ไม่รันโปรแกรมการตรวจจับ	รันโปรแกรมการตรวจจับ	ค่าความต่าง (นาที)
1	13.30	15.34	2.04
2	13.30	15.28	1.98
3	13.30	15.24	1.94
4	13.30	15.27	1.97
5	13.30	15.27	1.97
ค่าเฉลี่ย			1.98
จำนวนโปรเซสที่เกิดขึ้นในการคอมไพล์เคอเนลแต่ละครั้ง			**9149 โปรเซส**

4.7.2 การเรียกใช้หน่วยความจำภายในระบบ

พิจารณาจากขนาดของแฟ้มโดยดูจากผลของการใช้คำสั่ง *"top"* ดูการเปลี่ยนแปลงขนาดของโปรแกรม ตรวจจับ โดยรวบรวมข้อมูลดังแสดงในตารางที่ 4.5 – 4.11 ผลที่ได้ แสดงให้เห็นว่าจากตารางที่ 4.5 ซึ่งเป็นข้อมูลที่ได้หลังจากที่เริ่มต้นใช้ระบบตรวจจับการบุกรุกขนาดเริ่มต้นของโปรแกรมตรวจจับการบุกรุกที่ติดตามแต่ละโปรเซสมีขนาด 100K เมื่อเวลาผ่านไปจนถึงในตารางที่ 4.11 ซึ่งเป็นวันที่สามที่โปรแกรมตรวจจับการบุกรุกทำงานอยู่เบื้องหลังมีบางโปรแกรมที่มีขนาดของแฟ้มเพิ่มขึ้นในวันที่สอง และวันที่สามหลังจากนั้นจนกระทั่งวันที่ห้าขนาดคงที่ ขนาดที่เพิ่มขึ้นคิดรวมจากวันแรกที่เริ่มต้นใช้โปรแกรมตรวจจับจนถึงวันที่สามคิดเป็น 4% ของวันเริ่มต้น รวมขนาดทั้งหมดเท่ากับ 728 K จากเริ่มต้นรวม 700 K

ตารางที่ 4.5 ผลจากการใช้โปรแกรม top เพื่อดูขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุกในวันที่ 1 ครั้งที่ 1 (Tue Sep 14 06:22:01 ICT 2004)

PID	USERNAME	PRI	NICE	*SIZE*	RES	STATE	TIME	WCPU	CPU	COMMAND
366	root	-6	0	***100K***	628K	RUN	0:00	0.00%	0.00%	idssysc
401	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
378	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
408	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
370	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
405	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc
409	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc

ตารางที่ 4.6 ผลจากการใช้โปรแกรม top เพื่อดูขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุกในวันที่ 1 ครั้งที่ 2 (Tue Sep 14 20:46:22 ICT 2004) เมื่อเวลาผ่านไป 13 ชั่วโมง

PID	USERNAME	PRI	NICE	*SIZE*	RES	STATE	TIME	WCPU	CPU	COMMAND
366	root	-6	0	***100K***	628K	RUN	0:16	0.00%	0.00%	idssysc
408	root	-6	0	***100K***	616K	piperd	0:01	0.00%	0.00%	idssysc
378	root	-6	0	***116K***	652K	piperd	0:00	0.00%	0.00%	idssysc
370	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
401	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
409	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc
405	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc

ตารางที่ 4.7 ผลจากการใช้โปรแกรม top เพื่อดูขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุกในวันที่ 2 ครั้งที่ 1 (Wed Sep 15 04:18:34 ICT 2004) เมื่อเวลาผ่านไป 20 ชั่วโมง

PID	USERNAME	PRI	NICE	*SIZE*	RES	STATE	TIME	WCPU	CPU	COMMAND
366	root	-6	0	***100K***	628K	RUN	0:27	0.00%	0.00%	idssysc
401	root	-6	0	***100K***	628K	piperd	0:13	0.00%	0.00%	idssysc
408	root	- 6	0	***100K***	616K	piperd	0:02	0.00%	0.00%	idssysc
378	root	-6	0	***128K***	664K	piperd	0:00	0.00%	0.00%	idssysc
405	root	-6	0	***100K***	628K	piperd	0:00	0.00%	0.00%	idssysc
370	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
409	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc

ตารางที่ 4.8 ผลจากการใช้โปรแกรม top เพื่อดูขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุกในวันที่ 2 ครั้งที่ 2 (Wed Sep 15 17:06:02 ICT 2004) เมื่อเวลาผ่านไป 37 ชั่วโมง

PID	USERNAME	PRI	NICE	*SIZE*	RES	STATE	TIME	WCPU	CPU	COMMAND
366	root	-6	0	***100K***	628K	RUN	0:42	0.00%	0.00%	idssysc
401	root	-6	0	***100K***	628K	piperd	0:13	0.00%	0.00%	idssysc
408	root	-6	0	***100K***	616K	piperd	0:04	0.00%	0.00%	idssysc
378	root	-6	0	***128K***	664K	piperd	0:00	0.00%	0.00%	idssysc
405	root	-6	0	***100K***	628K	piperd	0:00	0.00%	0.00%	idssysc
370	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
409	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc

ตารางที่ 4.9 ผลจากการใช้โปรแกรม top เพื่อดูขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุกในวันที่ 3 ครั้งที่ 1 (Wed Sep 15 23:17:47 ICT 2004) เมื่อเวลาผ่านไป 43 ชั่วโมง

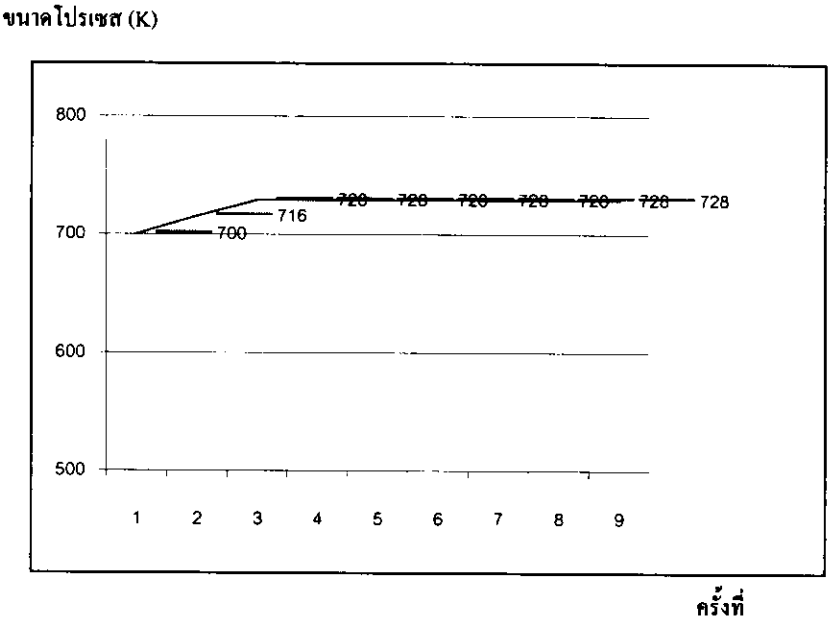
PID	USERNAME	PRI	NICE	*SIZE*	RES	STATE	TIME	WCPU	CPU	COMMAND
366	root	-6	0	***100K***	628K	RUN	0:49	0.00%	0.00%	idssysc
401	root	-6	0	***100K***	628K	piperd	0:13	0.00%	0.00%	idssysc
408	root	-6	0	***100K***	616K	piperd	0:05	0.00%	0.00%	idssysc
378	root	-6	0	***128K***	664K	piperd	0:00	0.00%	0.00%	idssysc
405	root	-6	0	***100K***	628K	piperd	0:00	0.00%	0.00%	idssysc
370	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
409	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc

ตารางที่ 4.10 ผลจากการใช้โปรแกรม top เพื่อดูขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุก ในวันที่ 3 ครั้งที่ 2 (Thu Sep 16 02:28:23 ICT 2004) เมื่อเวลาผ่านไป 47 ชั่วโมง

PID	USERNAME	PRI	NICE	*SIZE*	RES	STATE	TIME	WCPU	CPU	COMMAND
366	root	-6	0	***100K***	628K	RUN	0:53	0.00%	0.00%	idssysc
401	root	-6	0	***100K***	628K	piperd	0:13	0.00%	0.00%	idssysc
408	root	-6	0	***100K***	616K	piperd	0:05	0.00%	0.00%	idssysc
378	root	-6	0	***128K***	664K	piperd	0:00	0.00%	0.00%	idssysc
405	root	-6	0	***100K***	628K	piperd	0:00	0.00%	0.00%	idssysc
370	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
409	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc

ตารางที่ 4.11 ผลจากการใช้โปรแกรม top เพื่อดูขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุกใน วันที่ 5 ครั้งที่ 1 (Sat Sep 18 06:34:47 ICT 2004) เมื่อเวลาผ่านไป 90 ชั่วโมง

PID	USERNAME	PRI	NICE	*SIZE*	RES	STATE	TIME	WCPU	CPU	COMMAND
366	root	-6	0	***100K***	628K	RUN	1:51	0.00%	0.00%	idssysc
401	root	-6	0	***100K***	628K	piperd	0:58	0.00%	0.00%	idssysc
408	root	-6	0	***100K***	616K	piperd	0:11	0.00%	0.00%	idssysc
378	root	-6	0	***128K***	664K	piperd	0:00	0.00%	0.00%	idssysc
405	root	-6	0	***100K***	628K	piperd	0:00	0.00%	0.00%	idssysc
370	root	-6	0	***100K***	616K	piperd	0:00	0.00%	0.00%	idssysc
409	root	-6	0	***100K***	612K	piperd	0:00	0.00%	0.00%	idssysc



กราฟรูปที่ 4.1 แสดงการเปลี่ยนแปลงขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุก

จากกราฟรูปที่4.1 แสดงให้เห็นแนวโน้มการเพิ่มขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุกโดยใช้ตัวเลขที่ได้จากโปรแกรม top ในการสังเกตในระยะเวลา 5 วัน

4.8 สรุป

จากการทดสอบโปรแกรมตรวจจับการบุกรุกทั้งหมด จะเห็นได้ว่าระบบตรวจจับการบุกรุกสามารถตรวจจับพฤติกรรมที่เป็นลักษณะของการบุกรุก โดยที่ในส่วนของโปรแกรมบุกรุกไม่ว่าจะเป็นโปรแกรมบุกรุกประเภทม้าโทรจัน buffer overflow หรือ program bug การทำงานของโปรแกรมบุกรุกเหล่านี้ส่วนใหญ่เป็นการทำงานในลักษณะที่มีการเรียกใช้ system call *execve()* เพื่อทำงานอื่นอย่างเช่น เพื่อเรียกใช้ /bin/sh เพื่อให้ได้มาซึ่งเชลล์ของผู้ใช้ที่มีสิทธิสูงสุด โปรแกรมบุกรุกเหล่านี้จะถูกตรวจจับได้ตามเงื่อนไขในกฎข้อที่ 1 นั่นคือไม่อนุญาตให้มีการเรียกใช้ system call *execve()* ในขณะที่โปรเซสอยู่ในสถานะสิทธิพิเศษ เมื่อพิจารณาในส่วนของการตัดสินผิดทางบวก จะเห็นได้ว่าเมื่อใช้การตรวจจับการบุกรุกที่ใช้เทคนิคการพิจารณาการเปลี่ยนแปลงสถานะร่วมกับการพิจารณาการเรียกใช้ system call การตัดสินผิดทางบวกแทบจะไม่เกิดขึ้นเลย เนื่องจากการพิจารณาจาก system call และการเปลี่ยนแปลงสถานะค่อนข้างแน่นอนไม่มีการเปลี่ยนแปลงเหมือนกันการพิจารณาจากลำดับของ system call ที่ถูกเรียกใช้ซึ่งอาจจะเกิดการตัดสินใจผิดทางบวกได้ง่าย นอกจากนั้นการพิจารณาเฉพาะในสถานะที่โปรเซสอยู่ในสถานะสิทธิพิเศษและเลือกพิจารณาเฉพาะ system call บางตัวยังช่วยเป็นตัวกรองให้โปรแกรมทำงานได้เร็วและขนาดของโปรเซสที่ทำหน้าที่ตรวจจับการบุกรุกมีขนาดไม่ใหญ่มากนัก ดังจะเห็นได้จากข้อมูลในตารางที่ 4.4 ซึ่งแสดงให้เห็นเวลาที่ใช้ในการประมวลโปรเซสถึง 9,149 โปรเซส เมื่อมีการรันโปรแกรมตรวจจับการบุกรุกใช้เวลาเพิ่มขึ้นเฉลี่ยประมาณ 2 นาทีหรือคิดเป็น 14% และกราฟที่ 4.1 ซึ่งแสดงการเปลี่ยนแปลงขนาดของโปรเซสที่ใช้ตรวจจับการบุกรุก ในบทถัดไปจะเป็นการอภิปรายและสรุปผลเกี่ยวกับระบบตรวจจับการบุกรุกที่นำเสนอนี้