บทที่ 3
การวิเคราะห์และออกแบบกลไกการตรวจจับการบุกรุก

3.1. บทนำ

วิทยานิพนธ์ชุดนี้เป็นการเพิ่มสมรรถนะทางด้านความปลอดภัยให้แก่ระบบปฏิบัติการด้วยการเพิ่มกลไกการตรวจจับการบุกรุกโดยการแก้ไขระบบปฏิบัติการ แต่เนื่องจากการแก้ไขระบบปฏิบัติการนั้นเป็นเรื่องที่ละเอียดอ่อนจึงจำเป็นต้องศึกษา ออกแบบและทดสอบแนวคิดในระดับโปรแกรมประยุกต์ก่อนเพื่อให้แน่ใจว่า แนวคิดดังกล่าวสามารถทำงานได้จริงและไม่ก่อให้เกิดผลกระทบต่อระบบปฏิบัติการก่อนที่จะแก้ไขระบบปฏิบัติการ วิทยานิพนธ์ชุดนี้จึงแบ่งส่วนของการออกแบบระบบออกเป็น 2 บทนั้นคือบทที่ 3 และบทที่ 4 โดยที่บทที่ 3 จะกล่าวถึงการวิเคราะห์กฎของการนิยามสถานะและกฎสนับสนุน (ซึ่งกล่าวถึงในหัวข้อที่ 2.6) เพื่อหาแนวทางในการแก้ไขระบบปฏิบัติการ หลังจากนั้นบทที่ 4 จะกล่าวถึงการทดสอบผลการวิเคราะห์โดยการพัฒนาโปรแกรมขึ้นมาเพื่อตรวจสอบกฎ แล้วทดสอบผลการทำงานของโปรแกรมว่าเป็นไปตามที่คาดหวังไว้หรือไม่

ในบทนี้จึงเป็นการกล่าวถึงการติดตามการทำงานของโปรเซสเพื่อศึกษาถึงพฤติกรรมของโปรเซสแบบปกติและโปรเซสแบบ setuid แล้วนำผลที่ได้ไปใช้ในการวิเคราะห์หาวิธีการตรวจจับการบุกรุก หลังจากนั้นวิเคราะห์กฎของการนิยามสถานะและกฎสนับสนุนแต่ละข้ออย่างละเอียดพร้อมทั้งหาซิซเท็มคอลที่เกี่ยวข้องกับกฎสนับสนุนแต่ละข้อเพื่อใช้ในกระบวนการตรวจจับการบุกรุก ท้ายที่สุดวิเคราะห์หาข้อมูลนำเข้าสำหรับการตรวจจับการบุกรุกเพื่อใช้ในการพัฒนาระบบต่อไป

3.2. การติดตามการทำงานของโปรเซส

จากการศึกษาถึงวิธีการตรวจจับการบุกรุกโดยการวิเคราะห์การเปลี่ยนแปลงสถานะของโปรเซสในหัวข้อที่ 2.6 พบว่า การตรวจจับการบุกรุกจะต้องติดตามการทำงานของโปรเซสตลอดการทำงานจนกว่าโปรเซสดังกล่าวเรียกใช้ซิซเท็มคอล exit() เพื่อจบการทำงาน เนื่องจากในกรณีที่โปรเซสนั้นอยู่ในสถานะพิเศษ การตรวจจับการบุกรุกต้องติดตามการเรียกใช้ฟังก์ชันของโปรเซสเป้าหมายว่ากระทำการใดๆ ที่ขัดต่อกฎสนับสนุนหรือไม่ การศึกษาในหัวข้อนี้

จึงเป็นการติดตามการทำงานของโปรเซสแต่ละแบบเพื่อวิเคราะห์หาข้อแตกต่างของโปรเซสปกติ โปรเซสแบบ setuid และโปรเซสบุกรุก พร้อมทั้งหาวิธีการพิจารณาว่าโปรเซสที่กำลังติดตามอยู่นั้นเป็นโปรเซสบุกรุกหรือไม่

3.2.1. วิธีการติดตามการทำงานของโปรเซส

เมื่อโปรเซสต้องการเข้าถึงทรัพยากรใดๆ ของระบบ โปรเซสดังกล่าวจะร้องขอทรัพยากรนั้นผ่านซิซเท็มคอลของระบบปฏิบัติการ การดำเนินงานเหล่านี้เกิดขึ้นในสภาวะแวดล้อมของเคอร์เนล ซึ่งมีข้อมูลเกิดจากการดำเนินงานของโปรเซสในระดับเคอร์เนลได้แก่ ซิซเท็มคอลที่โปรเซสเรียกใช้ แฟ้มที่เกี่ยวข้อง และอุปกรณ์นำเข้าและส่งออก เป็นต้น ข้อมูลที่กล่าวมาแล้วข้างต้นเป็นข้อมูลที่ถูกจัดเก็บอยู่ในโครงสร้างข้อมูลที่ไม่สามารถเข้าถึงได้โดยโปรเซสของผู้ใช้ปกติ นอกจากนี้ข้อมูลจะเกิดขึ้นเฉพาะตอนที่โปรเซสกำลังทำงานเท่านั้นและจะถูกทำลายเมื่อโปรเซสหยุดการทำงาน

ในระบบปฏิบัติการยูนิกซ์ตระกูลบีเอสดี (BSD) เช่น ระบบปฏิบัติการเน็ทบีเอสดี ได้ให้บริการซิซเท็มคอล ktrace() ซึ่งใช้สำหรับติดตามพฤติกรรมของโปรเซสโดยอ่านข้อมูลดังที่กล่าวมาแล้วข้างต้น และระบบปฏิบัติการได้อนุญาตเฉพาะ root เท่านั้นที่สามารถติดตามการทำงานของโปรเซสอื่นด้วยซิซเท็มคอล ktrace() ตัวอย่างของผลการติดตามการทำงานของโปรเซสแสดงไว้ในตารางที่ 3.1

ตารางที่ 3.1 แสดงผลของการติดตามการเรียกใช้ซิซเท็มคอลของโปรเซส ls ด้วยคำสั่ง ktrace

line#	PID	process name	trace point	system call and parameter
1	….			
2	174	bash	CALL	__stat13(0x80d88a8,0xbfbff440)
3	174	bash	NAMI	/bin/ls
4	174	bash	RET	__stat13 0
5	….			
6	174	bash	CALL	fork
7	174	bash	RET	fork 337/0x151
8	174	bash	CALL	wait4(0xffffffff,0xbfbff658,2,0)
9	….			
10	337	bash	RET	fork 0
11	337	bash	EMUL	netbsd
12	….			
13	337	bash	CALL	execve(0x80d88a8,0x80d8868,0x80df008)
14	337	bash	NAMI	/bin/ls
15	337	bash	NAMI	/libexec/ld.elf_so
16	337	ls	EMUL	netbsd

จากตารางที่ 3.1 เป็นการแสดงผลการติดตามการทำงานของโปรเซส ls ด้วยซิซเท็มคอล ktrace() ในแต่ละบรรทัดจะประกอบไปด้วยส่วนต่างๆ ดังนี้ หมายเลขโปรเซส ชื่อโปรเซส จุดที่ถูกติดตาม (trace point) ซิซเท็มคอล และค่าพารามิเตอร์ต่างๆ ส่วนประกอบแต่ละอย่างมีรายละเอียดดังนี้

- **Process ID** หรือ **PID** หมายถึงหมายเลขของโปรเซสที่ถูกติดตามการทำงาน
- **ชื่อโปรเซส** (process name) หมายถึงชื่อของโปรเซสที่กำลังถูกติดตาม
- **จุดที่ถูกติดตาม** (trace point) หมายถึงชนิดของข้อมูลในโครงสร้างข้อมูล เนื่องจากข้อมูลที่ได้จากการติดตามการทำงานมีหลายประเภทเช่น ชื่อซิซเท็มคอล การคืนค่าของซิซเท็มคอล เส้นทางของแฟ้มที่ถูกเลือกใช้ เป็นต้น จึงสรุปจุดของการติดตามโปรเซสไว้ในตารางที่ 3.2

ตารางที่ 3.2 แสดงความหมายของ trace point ของซิซเท็มคอล ktrace()

trace point	ความหมาย
CALL	เป็นการติดตามการเรียกใช้ซิซเท็มคอลโดยจะแสดงชื่อและค่าพารามิเตอร์ของซิซเท็มคอลในขณะที่โปรเซสกำลังทำงาน
RET	เป็นการติดตามค่าที่ส่งกลับมาจากการทำงานของซิซเท็มคอล
NAMEI	เป็นการติดตาม path ของแฟ้มในขณะที่กำลังประมวลผล โดยจะแปลงค่าในพารามิเตอร์ของซิซเท็มคอลให้แสดงผลเป็นชื่อแฟ้ม
GENIO	เป็นการติดตามการเรียกใช้อุปกรณ์นำเข้าและส่งออกของระบบ
PSIG	เป็นการติดตามการประมวลผลสัญญาณของโปรเซส
CSW	เป็นการติดตามการดำเนินการ context switch ของโปรเซส

- **ชื่อของซิซเท็มคอล** เนื่องจากการทำงานของโปรเซสในระดับเคอร์เนลนั้นเป็นการดำเนินการผ่านซิซเท็มคอล ดังนั้นผลของการติดตามการทำงานของโปรเซสในแต่ละบรรทัดจะแสดงชื่อซิซเท็มคอลอยู่เสมอ
- **พารามิเตอร์** (parameter) จะปรากฏในบางซิซเท็มคอลที่ต้องการค่าพารามิเตอร์สำหรับการดำเนินงาน

จากการศึกษาถึงการนิยามสถานะในหัวข้อ 2.6 พบว่า ข้อมูลสำคัญที่ใช้สำหรับการนิยามสถานะคือค่าประจำสถานะของโปรเซสซึ่งประกอบไปด้วย [UID, EUID, GID, EGID] แต่ซิซเท็มคอล ktrace() ไม่สามารถแสดงข้อมูลเหล่านี้ออกมาจึงจำเป็นที่จะต้องแก้ไขโครงสร้างข้อมูลและซิซเท็มคอลดังกล่าวโดยเพิ่มส่วนของการแสดงค่าประจำสถานะออกมาในขณะที่กำลังติดตามการทำงานของโปรเซส หลังจากนั้นต้องคอมไพล์เคอร์เนลใหม่ วิธีการติดตามการทำงานข้างต้นเรียกว่า "UID Monitoring" [Nuansri, 1999]

3.2.2. โปรเซสแบบปกติ

โปรเซสแบบปกติหมายถึงโปรเซสที่ไม่มีการเปลี่ยนแปลงค่า user credential ตั้งแต่เริ่มทำงานจนเรียกใช้ซิซเท็มคอล exit() เพื่อจบการทำงาน หรือพิจารณาตามกฎการนิยามสถานะจะได้ว่า โปรเซสแบบปกติคือโปรเซสที่ไม่มีการเปลี่ยนแปลงสถานะในขณะที่โปรเซสกำลังทำงาน ตัวอย่างการติดตามทำงานของโปรเซส ls ด้วยคำสั่ง ktrace หลังจากที่แก้ไขซิซเท็มคอล ktrace() ดังตารางที่ 3.2

ตารางที่ 3.3 แสดงผลของการติดตามการเรียกใช้ซิซเท็มคอลของโปรเซส ls ด้วยคำสั่ง ktrace

line	UID	EUID	GID	EGID	PID	Process	trace point	system call & parameter
1	1000	1000	100	100	176	bash	CALL	__stat13(0x80d88a8, 0xbfbff440)
2	1000	1000	100	100	176	bash	NAMI	"/bin/ls"
3	1000	1000	100	100	176	bash	RET	_stat13 0
4	1000	1000	100	100	176	bash	CALL	fork
5	1000	1000	1000	100	100	bash	RET	fork 338/0x152
6	1000	1000	100	100	176	bash	CALL	wait4(0xffffffff,0xbfbff658,2,0)
7	1000	1000	100	100	338	bash	RET	fork 0
8	1000	1000	100	100	338	bash	EMUL	netbsd
	....							
9	1000	1000	100	100	338	bash	CALL	execve(0x80d88a8,0x80d8868,…)
10	1000	1000	100	100	338	bash	NAMI	/bin/ls
11	1000	1000	100	100	338	bash	NAMI	/libexec/ld.elf_so
12	1000	1000	100	100	338	ls	EMUL	netbsd
13	1000	1000	100	100	338	ls	CALL	write(1, 0x8053000, 0x2f)
14	1000	1000	100	100	338	ls	GIO	fd 1 wrote 74 bytes

จากตารางที่ 3.2 แสดงผลการทำงานของซิซเท็มคอล ktrace() ซึ่งประกอบด้วยข้อมูลสองส่วนได้แก่ค่าประจำสถานะของโปรเซสซึ่งเขียนแทนด้วย [UID, EUID, GID, EGID] และผลของการทำงานของซิซเท็มคอล สำหรับวิทยานิพนธ์ชุดนี้พิจารณาเหตุการณ์เฉพาะ trace point ชื่อ CALL (เนื่องจากต้องการติดตามการใช้งานซิซเท็มคอลของโปรเซส) ส่วนหมายเลขบรรทัดนั้นเพิ่มมาเพื่อความสะดวกในการอ้างถึงเท่านั้น ผลการติดตามการทำงานแต่ละส่วนมีรายละเอียดดังนี้

ส่วนที่ 1 (บรรทัดที่ 1-3) เป็นกระบวนการตรวจสอบสถานะของโปรแกรม ls ด้วยซิซเท็มคอล stat13() เพื่อตรวจสอบว่าโปรแกรมดังกล่าวมีอยู่จริงและผู้ใช้มีสิทธิ์ในการเข้าถึงและสั่งงานโปรแกรมนี้

ส่วนที่ 2 (บรรทัดที่ 4-6) เป็นขั้นตอนการทำงานของโปรแกรมเชลล์ของระบบปฏิบัติการ (ซึ่งในที่นี้คือ bash) โดยโปรแกรมเชลล์จะสร้างโปรเซสใหม่ขึ้นมาด้วยซิซเท็มคอล fork () โปรเซสใหม่มีค่า PID เป็น 338 หลังจากนั้นก็เรียกใช้ซิซเท็มคอล wait() เพื่อรอรับสัญญาณจากโปรเซสลูก เนื่องจากการโปรเซส ls นั้นทำงานแบบเบื้องหน้า (foreground process)

ส่วนที่ 3 (บรรทัดที่ 7-12) โปรเซสลูกเรียกใช้ซิซเท็มคอล execve() เพื่อโหลดชุดคำสั่งของโปรแกรม ls เข้าสู่หน่วยความจำหลังจากนั้นโปรเซส ls ดำเนินการตามชุดคำสั่งของโปรเซสเพื่อที่จะอ่านชื่อแฟ้มในระบบมาเก็บไว้ในหน่วยความจำ

ส่วนที่ 4 (บรรทัดที่ 13-14) โปรเซสแสดงผลการทำงานด้วยซิซเท็มคอล write() ซึ่งแสดงไว้ในบรรทัดที่ 14 ท้ายที่สุดโปรเซส ls เรียกใช้ซิซเท็มคอล exit() เพื่อหยุดการทำงานของโปรเซสและส่งสัญญาณไปยังโปรเซสแม่ (parent process) ซึ่งมี PID เป็น 176

จากการศึกษาถึงการทำงานของโปรเซส ls ข้างต้นพบว่า กระบวนการทำงานของโปรเซส ls ซึ่งสั่งงานโดยผู้ใช้ที่มีค่า UID และค่า GID เป็น 1000 และ 100 ตามลำดับ ตั้งแต่เริ่มต้นการทำงาน (บรรทัดที่ 9) จนจบการทำงาน (บรรทัดที่ 16) ค่าประจำสถานะของโปรเซสมีค่าเป็น [1000, 100, 1000, 100] ตลอดช่วงเวลาการทำงาน ซึ่งแสดงให้เห็นว่าการทำงานของโปรเซสแบบปกติเนื่องจากค่าประจำสถานะไม่มีการเปลี่ยนแปลงตลอดการทำงาน

3.2.3. โปรเซสแบบ setuid

จากการนิยามไว้ในหัวข้อที่ 2.5.5. โปรแกรมแบบ setuid คือโปรแกรมที่ได้รับการกำหนดสิทธิพิเศษ เมื่อผู้ใช้ปกติสั่งงานโปรแกรมดังกล่าว โปรแกรมนั้นจะทำงานด้วยสิทธิพิเศษชั่วคราวเพื่อที่จะดำเนินงานพิเศษบางอย่างที่ผู้ใช้ปกติของระบบไม่สามารถทำได้

การติดตามการทำงานของโปรเซสแบบ setuid ในที่นี้ยกตัวอย่างการทำงานของคำสั่ง passwd และถูกสั่งงานโดยผู้ใช้มีค่า UID และ GID เป็น 1000 และ 100 ตามลำดับ ผลของการติดตามการทำงานแสดงไว้ในตารางที่ 3.3

ตารางที่ 3.4 แสดงผลของการติดตามการเรียกใช้ซิซเท็มคอลของโปรเซส passwd ด้วยคำสั่ง ktrace

line#	UID	EUID	GID	EGID	PID	process	trace point	system call & parameter
1	1000	1000	100	100	602	csh	CALL	execve(0x806ec00,0x806b650,…)
2	1000	1000	100	100	602	csh	NAMI	"/usr/bin/passwd"
3	1000	1000	100	100	602	csh	NAMI	"/usr/libexec/ld.elf_so"
4	1000	0	100	100	602	passwd	EMUL	"netbsd"
5	1000	0	100	100	602	passwd	RET	execve JUSTRETURN
6	1000	0	100	100	602	passwd	CALL	write(0x3,0x8067000,0xd)
7	1000	0	100	100	602	passwd	GIO	fd 3 wrote 13 bytes
8	1000	0	100	100	602	passwd	RET	write 13/0xd
9	1000	0	100	100	602	passwd	CALL	read(0x3,0x8067000,0x10000)
10	1000	0	100	100	602	passwd	GIO	fd 3 read 7 bytes
11	1000	0	100	100	602	passwd	RET	read 7
12	1000	0	100	100	607	pwd_mkdb	CALL	open(0xbfbff9da,0,0x1b6)
13	1000	0	100	100	607	pwd_mkdb	NAMI	"/etc/ptmp"
14	1000	0	100	100	607	pwd_mkdb	NAMI	"/etc/ptmp"
15	1000	0	100	100	607	pwd_mkdb	RET	open 3
15	1000	0	100	100	602	passwd	CALL	exit(0)
16	1000	1000	100	100	600	passwd	RET	wait4 602/0x25a

ส่วนที่ 1 (บรรทัดที่ 1-5) พบว่าก่อนสั่งงานโปรเซส passwd ค่าประจำสถานะของโปรเซสคือ [1000, 1000, 100, 100] แต่หลังจากที่เรียกใช้ซิซเท็มคอล execve() แล้วค่า EUID ของโปรเซสเปลี่ยนจาก 1000 เป็น 0 ซึ่งหมายถึงโปรแกรมทำงานด้วยสิทธิ์ของ root

ส่วนที่ 2 (บรรทัดที่ 6-11) โปรเซสได้ดำเนินการต่างๆ เพื่อเตรียมพร้อมสำหรับการแก้ไขฐานข้อมูลผู้ใช้

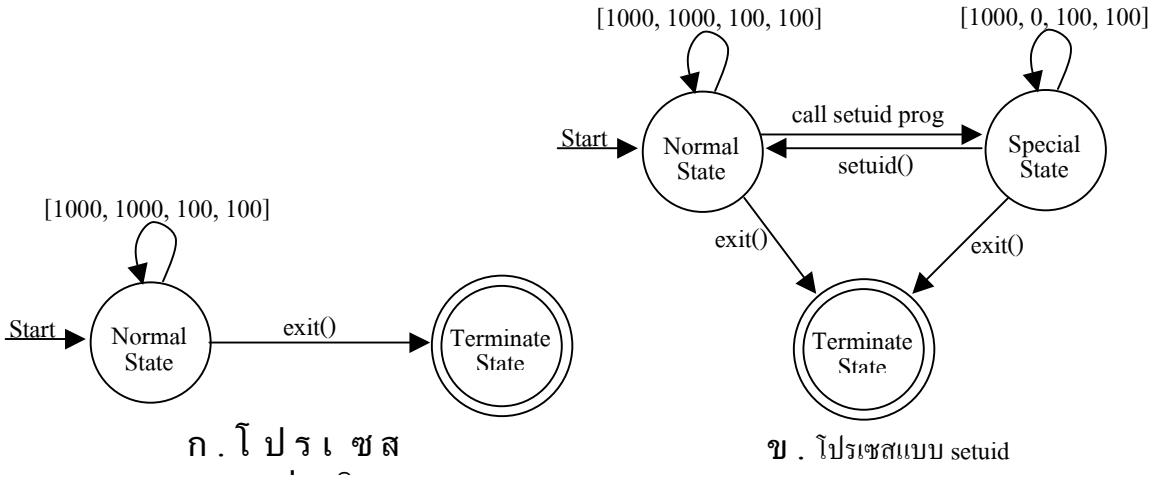
ส่วนที่ 3 (บรรทัดที่ 12-15) โปรเซส passwdได้สร้างโปรเซสใหม่ในขณะที่โปรเซสดังกล่าวมีค่าประจำสถานะเป็น [1000, 0, 100, 100] ผลที่ตามมาคือโปรเซสใหม่ก็ได้รับการถ่ายทอดค่าประจำสถานะนั้นไปด้วย

ส่วนที่ 4 (บรรทัดที่ 16-17) เมื่อโปรเซส passwd ดำเนินการเรียบร้อยแล้ว โปรเซสดังกล่าวเรียกใช้ซิซเท็มคอล exit() เพื่อหยุดการทำงานของโปรเซส

จากการศึกษาถึงการทำงานของโปรเซส passwd ข้างต้นพบว่า กระบวนการทำงานของโปรเซส passwd ตั้งแต่เริ่มต้นจนจบการทำงานค่าประจำสถานะของโปรเซสได้เปลี่ยนจากค่า [1000, **1000**, 100, 100] ซึ่งเป็นสถานะปกติ ไปเป็นค่า [1000, **0**, 100, 100] ซึ่งเป็นสถานะพิเศษเพื่อแก้ไขแฟ้มชื่อบัญชีผู้ใช้ เมื่อโปรเซสดังกล่าวดำเนินการจนจบการทำงานก็เรียกใช้ซิซเท็มคอล exit()

3.2.4. สรุปผลการศึกษาโปรเซสแบบปกติและโปรเซสแบบ setuid

จากการศึกษาถึงโปรเซสแบบปกติและโปรเซสแบบ setuid ในหัวข้อที่ 3.2.2 และ 3.2.3 พบว่าโปรแกรมทั้งสองแบบมีความแตกต่างกันโดยพิจารณาจากค่าประจำสถานะ สำหรับค่าประจำสถานะของโปรเซสแบบปกติจะไม่เปลี่ยนแปลงตลอดการทำงาน แต่สำหรับโปรเซสแบบ setuid แล้วค่าประจำสถานะจะเปลี่ยนแปลงหากมีการเรียกใช้ซิซเท็มคอลที่เปลี่ยนค่า user credential เพื่อให้เข้าใจถึงความแตกต่างของโปรเซสแบบปกติและโปรเซสแบบ setuid จึงแทนการทำงานของโปรเซสข้างต้นด้วยด้วยแผนภาพการเปลี่ยนแปลงสถานะในภาพประกอบที่ 3.1



ภาพประกอบ 3.1 แผนภาพการเปลี่ยนแปลงสถานะของโปรเซสแบบปกติและแบบ setuid [Nuansri, 1999]

แผนภาพประกอบที่ 3.1ก แสดงให้เห็นว่าสถานะของโปรเซสแบบปกติจะไม่เปลี่ยนแปลงตลอดการทำงานของโปรเซส จนกระทั่งโปรเซสดังกล่าวเรียกใช้ซิซเท็มคอล exit() เพื่อจบการทำงาน ในขณะที่ภาพประกอบที่ 3.1ข แสดงให้เห็นถึงการทำงานของโปรเซสแบบ setuid เมื่อโปรเซสเริ่มทำงาน โปรเซสก็จะเข้าสู่สถานะปกติและดำเนินกิจกรรมต่างๆ จนกระทั่งโปรเซสเรียกใช้ซิซเท็มคอล execve() เพื่อสั่งงานโปรเซสแบบ setuid โปรเซสจะเปลี่ยนสถานะเข้าสู่สถานะพิเศษเนื่องจาก ค่า EUID ของโปรเซสเปลี่ยนเป็นค่า 0 หมายความว่าโปรเซสดังกล่าวได้รับสิทธิพิเศษในการทำงานจนกระทั่งเรียกใช้ซิซเท็มคอล setuid() เพื่อเปลี่ยนค่า EUID ให้เป็น 1000 โปรเซสจะเปลี่ยนสถานะกลับไปอยู่ในสถานะปกติ แล้วดำเนินงานอื่นต่อไปจนกว่าเรียกใช้ซิซเท็มคอล exit() เพื่อหยุดการทำงาน อีกกรณีหนึ่งคือโปรเซสอาจจะเรียกใช้ซิซเท็มคอล exit() ในขณะที่อยู่ในสถานะพิเศษ ซึ่งจะเป็นผลให้โปรเซสเปลี่ยนเข้าสู่สถานะยุติการทำงาน

3.3. การวิเคราะห์การเปลี่ยนแปลงสถานะของโปรเซสและกฎสนับสนุน

สำหรับเอกสารในหัวข้อนี้เป็นการวิเคราะห์กฎการนิยามสถานะและกฎสนับสนุนเพื่อหาวิธีการนิยามสถานะของโปรเซสและการตรวจสอบโปรเซสด้วยกฎสนับสนุน

3.3.1. การวิเคราะห์การเปลี่ยนแปลงสถานะของโปรเซส

โดยปกติเมื่อโปรเซสเริ่มทำงาน โปรเซสจะเข้าสู่สถานะปกติ หากมีการสั่งงานโปรแกรมแบบ setuid โปรเซสจะเปลี่ยนสถานะเป็นสถานะ setuid ซึ่งดำเนินกิจกรรมต่างๆ ด้วยสิทธิพิเศษ เมื่อโปรเซสอยู่ในสถานะนี้โปรเซสดังกล่าวสามารถเปลี่ยนสถานะเป็นสถานะอื่นได้ เช่นเปลี่ยนสถานะเข้าสู่สถานะ setreuid() ด้วยซิซเท็มคอล setreuid() เรียกใช้ซิซเท็มคอล exit() เพื่อจบการทำงาน หรือกลับไปสู่สถานะปกติด้วยซิซเท็มคอล setuid() แต่ถ้าหากโปรเซสดังกล่าวเปลี่ยนสถานะเข้าสู่สถานะผู้ใช้สูงสุดถือว่าโปรเซสดังกล่าวพยายามที่จะบุกรุกระบบ ดังภาพประกอบที่ 2.5

จากการศึกษาการทำงานของซิซเท็มคอลทั้งหมดในระบบปฏิบัติการยูนิกซ์พบว่ามีเพียงซิซเท็มคอลบางตัวเท่านั้นที่สามารถเปลี่ยนแปลงค่าประจำตัวผู้ใช้ได้ นั่นคือซิซเท็มคอล setuid (), setgid(), seteuid() และ setegid() สำหรับการเปลี่ยนสถานะของโปรเซสส่วนใหญ่จะเปลี่ยนเพียงชั่วคราวเพื่อวัตถุประสงค์ในการทำงานบางอย่างเท่านั้น เมื่อโปรเซสนั้นทำงานตามวัตถุประสงค์เรียบร้อยแล้ว โปรเซสดังกล่าวจะกลับเข้าสู่สถานะปกติ

ดังนั้นการตรวจสอบการบุกรุกระบบในกรณีนี้จะตรวจสอบการเรียกใช้ซิซเท็มคอลที่กล่าวมาแล้วข้างต้น ถ้าหากเรียกใช้ซิซเท็มคอล setuid(), setgid(), seteuid() หรือ setegid() แล้วเป็นผลให้โปรเซสเปลี่ยนสถานะเข้าสู่สถานะผู้ใช้สูงสุดถือว่าโปรเซสดังกล่าวเป็นโปรเซสบุกรุก

3.3.2. การนิยามสถานะ

จากการศึกษาถึงการนิยามสถานะในหัวข้อที่ 2.6.1 โปรเซสมีสถานะทั้งหมด 6 สถานะแต่สำหรับการออกแบบระบบเพื่อพัฒนาโปรแกรมนั้นนิยามสถานะเพียง 5 สถานะนั้นคือ สถานะปกติ สถานะพิเศษ สถานะผู้ใช้สูงสุด สถานะกลุ่มระบบ และสถานะผู้ใช้อื่น ซึ่งแต่ละสถานะจะมีค่าประจำสถานะที่แตกต่างกัน ซึ่งได้แก่ uid, gid, sid, sgid, oid และ ogid ในระบบปฏิบัติการเน็ทบีเอสดีค่าต่างๆ ข้างต้นได้รับการกำหนดค่าดังตารางที่ 3.4 โดยกำหนดตามค่าปริยายของระบบ

ตารางที่ 3.5 แสดงการกำหนดค่าให้สัญลักษณ์ในระบบปฏิบัติการเน็ทบีเอสดี

สัญลักษณ์	ความหมาย	ค่าที่เป็นไปได้
uid	ค่าประจำตัวของผู้ใช้ปกติ	จำนวนเต็มตั้งแต่ 1000 เป็นต้นไป
gid	ค่าประจำกลุ่มของผู้ใช้ปกติ	จำนวนเต็มตั้งแต่ 100 เป็นต้นไป
sid	ค่าประจำตัวของผู้ดูแลระบบ	0
sgid	ค่าประจำกลุ่มของผู้ดูแลระบบ	จำนวนเต็มตั้งแต่ 0 ถึง 9
oid	ค่าประจำตัวของผู้ใช้ปกติ	จำนวนเต็มตั้งแต่ 1000 เป็นต้นไป
ogid	ค่าประจำกลุ่มของผู้ใช้ปกติ	จำนวนเต็มตั้งแต่ 100 เป็นต้นไป

จากตารางที่ 3.4 เป็นการการกำหนดค่าให้แก่สัญลักษณ์ในระบบปฏิบัติการเน็ทบีเอสดี ซึ่งอธิบายได้ว่า โดยปริยายแล้วระบบปฏิบัติการจะกำหนดค่า 1000 ให้แก่ผู้ใช้คนแรก และเพิ่มค่าขึ้นโดยอัตโนมัติสำหรับผู้ใช้คนถัดไป เว้นเสียแต่ว่าผู้ดูแลระบบกำหนดให้เป็นอย่างอื่น สำหรับค่า gid ของผู้ใช้นั้นถ้าหากผู้ดูแลไม่ได้กำหนดให้นั้น ระบบปฏิบัติการจะกำหนดค่า gid เป็น 100 ซึ่งมีชื่อว่า users ค่า oid และ ogid จะหมายถึงค่าประจำตัวผู้ใช้หรือค่าประจำกลุ่มที่มีค่าไม่เท่ากับค่า uid, gid, sid และ sgid ท้ายที่สุดสำหรับค่า sgid นั้นขึ้นอยู่กับระบบปฏิบัติการ

ตารางที่ 3.5 เป็นตัวอย่างของการนิยามสถานะของโปรเซสเมื่อโปรเซสมีค่า user credential เป็นค่าต่างๆ โดยกำหนดให้เจ้าของโปรเซสที่มีค่า UID และ GID เป็น 1000 และ 100 ตามลำดับ และค่า X และ Y นั้นเป็นค่าอื่นๆ ที่ไม่เท่ากับ 1000 100 และ 0

ตารางที่ 3.6 แสดงการนิยามสถานะเมื่อโปรเซสมีค่าประจำสถานะเป็นค่าต่างๆ

Classification				Example				State
UID	EUID	GID	EGID	UID	EUID	GID	EGID	
Uid	uid	gid	gid	1000	1000	100	100	Normal state
Sid	uid	gid	gid	**0**	1000	100	100	Setreuid state
Uid	**sid**	gid	gid	1000	**0**	100	100	Setuid state
Uid	uid	**sgid**	gid	1000	1000	**0**	100	Setgid state
Uid	uid	gid	**sgid**	1000	1000	100	**0**	Setregid state
Sid	**sid**	gid	gid	**0**	**0**	100	100	Super user state
Uid	uid	**sgid**	**sgid**	1000	1000	**0**	**0**	System group state
Oid	oid	gid	gid	X	X	20	20	Another user state
Uid	uid	ogid	ogid	1000	1000	Y	Y	

จากการวิเคราะห์กฎของการนิยามสถานะข้างต้นจึงเขียนอัลกอริทึมสำหรับการนิยามสถานะได้ดังนี้

```
define_state(uid, euid, gid, egid)
        If uid = euid and gid = egid then return NORMAL
        If gid < 9 and egid <9 then return SYSTEM_GROUP
        If uid = 0 and gid = 0 then return SUPER_USER
        If euid = 0  then  return SETUID
        If egid < 9  then  return SETGID
        If uid = 0  then   return SETREUID
        If gid < 9  then  return SETREGID
```

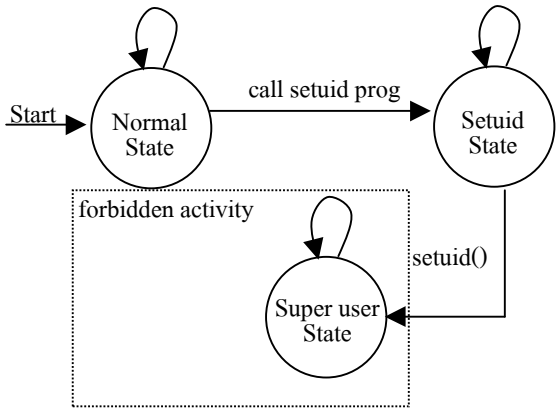
3.3.3. การวิเคราะห์กฎสนับสนุน

ในกรณีที่โปรเซสอยู่ในสถานะพิเศษซึ่งมีค่าประจำสถานะค่าใดค่าหนึ่งเป็นศูนย์ ถือว่าโปรเซสดังกล่าวมีสิทธิ์ในการเข้าใช้ทรัพยากรเป็นครึ่งหนึ่งของ root ดังนั้นจึงจำเป็นที่จะต้องติดตามการทำงานของโปรเซสดังกล่าวโดยตรวจสอบการเรียกใช้ซิซเท็มคอลของโปรเซส ว่าโปรเซสดังกล่าวดำเนินกิจกรรมใดๆ ที่ขัดต่อกฎสนับสนุนหรือไม่ ในลำดับถัดไปเป็นการศึกษาและวิเคราะห์กฎสนับสนุนเพื่อหาซิซเท็มคอลวิกฤตที่มีผลต่อความปลอดภัยของระบบ

3.3.3.1. กฎสนับสนุนข้อที่ 0

Rule 0: Only the special system calls setreuid() and the setregid() are permitted to change the (real) UID or GID respectively.

กฎสนับสนุนข้อนี้สืบเนื่องมาจากหัวข้อ 3.3.1 การวิเคราะห์การเปลี่ยนแปลงสถานะ จากการวิเคราะห์ข้างต้นพบว่า กลุ่มของซิซเท็มคอลที่ได้ใช้สำหรับเปลี่ยนแปลงค่า user credential ได้แก่ ซิซเท็มคอล setuid(), setgid(), seteuid() และ setegid() ในระบบปฏิบัติการรุ่น 4.3BSD ซิซเท็มคอลเหล่านี้ไม่มีกระบวนการตรวจสอบสิทธิ์การของการเปลี่ยนค่า user credential จนกระทั่งระบบปฏิบัติการรุ่น 4.4BSD ได้มีการสร้างซิซเท็มคอลเพิ่มเติมอีก 2 ตัวได้แก่ซิซเท็มคอล setreuid() และซิซเท็มคอล setregid() ซึ่งมีการเพิ่มกระบวนการตรวจสอบสิทธิ์ก่อนเปลี่ยนค่า user credential เป็นค่าอื่นๆ



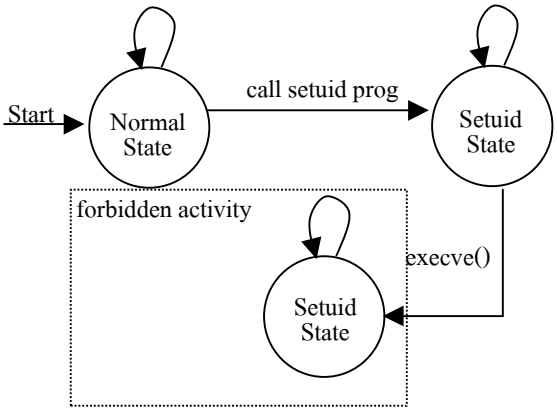
ภาพประกอบ 3.2 แผนภาพการเปลี่ยนแปลงสถานะของการตรวจจับการบุกรุกซึ่งละเมิดกฎสนับสนุนข้อที่ 0

การตรวจจับการละเมิดกฎสนับสนุนแสดงไว้ในภาพประกอบที่ 3.1 อธิบายได้ว่าเมื่อโปรเซสใดๆ เปลี่ยนสถานะจากสถานะปกติเข้าสู่สถานะพิเศษแล้วเรียกใช้ซิซเท็มคอลสำหรับเปลี่ยนแปลงค่าประจำสถานะ (ในที่นี้ยกตัวอย่างซิซเท็มคอล setuid()) เป็นผลให้โปรเซสเปลี่ยนสถานะเข้าสู่สถานะผู้ใช้สูงสุดแล้ว ถือว่าโปรเซสดังกล่าวเป็นโปรเซสบุกรุก

3.3.3.2. กฎสนับสนุนข้อที่ 1

Rule 1: No execve() call is allowed in a special privileged state.

กฎสนับสนุนข้อกำหนดขึ้นมาเพื่อป้องกันการบุกรุกระบบผ่าน buffer overflow (ซึ่งอธิบายไปแล้วในหัวข้อที่ 2.2.1.2) เนื่องจากซิซเท็มคอล execve() มีหน้าที่สำหรับสั่งงานโปรแกรมตามที่ระบุในพารามิเตอร์ โดยการคัดลอกชุดคำสั่งของโปรแกรมใหม่ไปทับชุดคำสั่งในส่วนของโค้ด (text segment) ในหน่วยความจำของโปรเซสเดิมที่เรียกใช้ซิซเท็มคอลตัวนี้ แต่ข้อมูลเดิมในตารางโปรเซส (process table) ของโปรเซสไม่ถูกเปลี่ยนแปลง ซึ่งรวมไปถึงค่า user credential ของโปรเซส หากโปรเซสเก่าเป็นโปรเซสที่ทำงานภายใต้สิทธิ์ของ root แล้วโปรเซสใหม่ที่เกิดขึ้นจะทำงานด้วยสิทธิ์ของ root ด้วยการตรวจจับการละเมิดกฎสนับสนุนแสดงไว้ในภาพประกอบที่ 3.3 อธิบายได้ว่าถ้าโปรเซสอยู่ในสถานะพิเศษและเรียกใช้ซิซเท็มคอล execve() เพื่อสั่งงานโปรแกรมใดๆ แล้วให้ถือว่าโปรเซสดังกล่าวพยายามละเมิดกฎสนับสนุนข้อที่ 1 ให้ถือว่าโปรเซสดังกล่าวเป็นโปรเซสบุกรุก



ภาพประกอบ 3.3 แผนภาพการเปลี่ยนแปลงสถานะของการตรวจจับการบุกรุกซึ่งละเมิดกฎสนับสนุนข้อที่ 1

3.3.3.3. กฎสนับสนุนข้อที่ 2

Rule 2: A process in a special privileged state is not allowed to create a setuid/setgid program.

ผู้บุกรุกอาจจะสร้างโปรแกรมแบบ setuid ผ่านช่องโหว่ของระบบด้วยวิธีการต่างๆ เพื่อใช้สำหรับการสั่งงานโปรแกรมนั้นภายหลัง ถ้าหากโปรเซสใดอยู่ในสถานะพิเศษโปรเซสดังกล่าวสามารถสร้างโปรแกรมที่สิทธิ์ในการทำงานด้วยสิทธิ์ของ root ได้ สำหรับผลกระทบที่จะเกิดขึ้นนั้นขึ้นอยู่กับวัตถุประสงค์ของโปรแกรมที่สร้างขึ้นมาในขณะนั้น วิธีการสร้างโปรแกรมแบบ setuid แบ่งออกเป็น 2 วิธีคือ สร้างโปรแกรมแบบ setuid ด้วยซิซเท็มคอล open() หรือสร้างโปรแกรมแบบ setuid ด้วยซิซเท็มคอลใน กลุ่มของ chmod

การสร้างโปรแกรมแบบ setuid ด้วยซิซเท็มคอล open()

ซิซเท็มคอล open() ใช้สำหรับการเปิดแฟ้มเพื่อสร้าง อ่าน หรือแก้ไขซึ่งขึ้นอยู่กับค่าพารามิเตอร์ของซิซเท็มคอล สำหรับซิซเท็มคอล open มีโพรโตไทป์ (prototype) ดังนี้

*open(const char *path, int flag, int mode)*

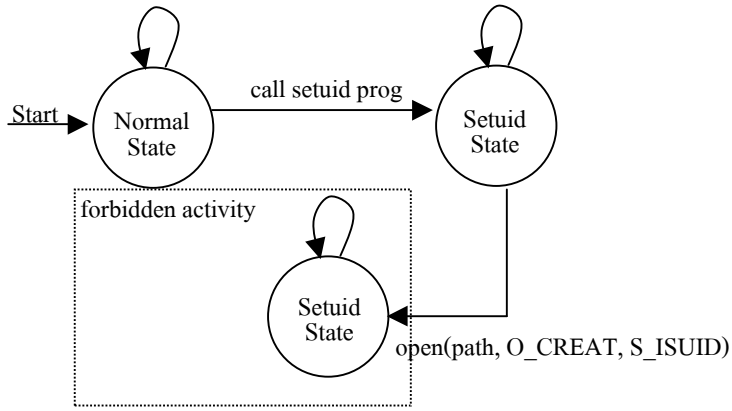
ซึ่งค่าพารามิเตอร์ประกอบไปด้วย

path หมายถึงชื่อแฟ้มข้อมูลที่ต้องการเปิด

flag หมายถึงเป้าหมายของการเปิดแฟ้มดังกล่าว เช่นการสร้างแฟ้มใหม่ การอ่านหรือการเขียน

mode หมายถึงการกำหนดสิทธิ์ของแฟ้มหลังจากที่สร้างแฟ้มแล้ว เช่น การกำหนดให้อ่านอย่างเดียว เขียนอย่างเดียวหรือกำหนดให้แฟ้มดังกล่าวเป็นแฟ้มแบบ setuid

การตรวจจับการละเมิดกฎสนับสนุนแสดงไว้ในภาพประกอบที่ 3.4 สามารถอธิบายได้ว่าถ้าโปรเซสอยู่ในสถานะพิเศษและเรียกใช้ซิซเท็มคอล open() เพื่อสร้างแฟ้มใหม่ (ค่าของ flag เป็น O_CREAT) และกำหนดให้แฟ้มดังกล่าวเป็นแฟ้มแบบ setuid (ค่าของ mode เป็น S_ISUID) เมื่อการดำเนินการข้างต้นสำเร็จ โปรเซสดังกล่าวจะสร้างโปรแกรมแบบ setuid ถ้าหากผู้ใช้คนใดสั่งงานโปรแกรมข้างต้นแล้วโปรเซสที่เกิดขึ้นจะทำงานด้วยสิทธิ์ของ root ดังนั้นจึงถือว่าโปรเซสที่สร้างโปรแกรมแบบ setuid ด้วยซิซเท็มคอล open() แล้วกำหนดค่าพารามิเตอร์ตามเงื่อนไขข้างต้นเป็นโปรเซสบุกรุก



ภาพประกอบ 3.4 แผนภาพการเปลี่ยนแปลงสถานะของการตรวจจับการบุกรุกซึ่งละเมิดกฎสนับสนุนข้อที่ 2 ด้วยซิซเท็มคอล open()

การสร้างโปรแกรมแบบ setuid ด้วยซิซเท็มคอลในกลุ่มของ chmod

ซิซเท็มคอลในกลุ่มของ chmod ใช้สำหรับการเปลี่ยนค่า permission ของแฟ้มที่ระบุในพารามิเตอร์ ซิซเท็มคอลในกลุ่มนี้ประกอบไปด้วย 3 ซิซเท็มคอล ได้แก่ซิซเท็มคอล chmod (), fchmod() และ lchmod() แต่ละซิซเท็มคอลมีโพรโตไทป์ (prototype) ดังนี้

*int chmod (const char *path, mode_t mode)*

int fchmod (int fd, mode_t mode)

*int lchmod (const char *path, mode_t mode)*

ซึ่งค่าพารามิเตอร์ประกอบไปด้วย

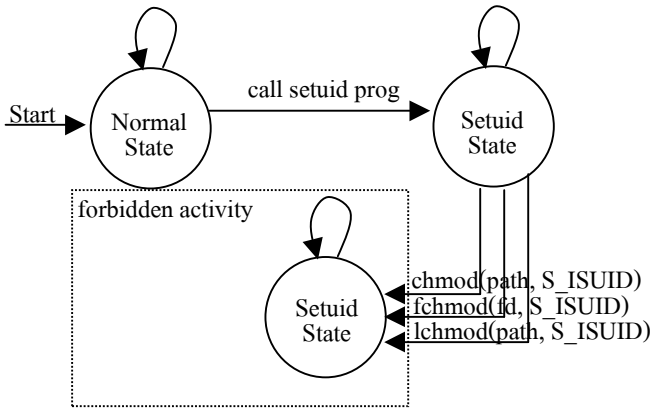
path หมายถึงชื่อแฟ้มข้อมูลที่ต้องการปรับค่า permission โดยแสดงเส้นทางแบบเต็ม

fd หมายถึง หมายเลข file descriptor ของแฟ้มที่ต้องการปรับค่า ซึ่งต้องเปิดแฟ้มดังกล่าวด้วยซิซเท็มคอล open() ก่อน หลังจากนั้นเรียกใช้ซิซเท็มคอลดังกล่าว

mode หมายถึงการกำหนดสิทธิ์ของแฟ้มที่ระบุ เช่น การกำหนดให้อ่าน/เขียนอย่างเดียว หรือกำหนดให้แฟ้มดังกล่าวเป็นแฟ้มแบบ setuid

การตรวจจับการละเมิดกฎสนับสนุนแสดงไว้ในภาพประกอบที่ 3.5 สามารถอธิบายได้ว่าถ้าโปรเซสอยู่ในสถานะพิเศษและเรียกใช้ซิซเท็มคอลในกลุ่มนี้เพื่อเปลี่ยนค่า permission ของโปรแกรมที่มีอยู่แล้วให้เป็นโปรแกรมแบบ setuid (ค่าของ mode เป็น S_ISUID)

เมื่อการดำเนินการข้างต้นสำเร็จ โปรเซสดังกล่าวจะสร้างโปรแกรมแบบ setuid ถ้าหากผู้ใช้คนใดสั่งงานโปรแกรมข้างต้นแล้วโปรเซสที่เกิดขึ้นจะทำงานด้วยสิทธิ์ของ root ดังนั้นจึงถือว่าโปรเซสที่สร้างโปรแกรมแบบ setuid ด้วยซิซเท็มคอลในกลุ่มนี้แล้วกำหนดค่าพารามิเตอร์ตามเงื่อนไขข้างต้นเป็นโปรเซสบุกรุก



ภาพประกอบ 3.5 แผนภาพการเปลี่ยนแปลงสถานะของการตรวจจับการบุกรุกซึ่งละเมิดกฎสนับสนุนข้อที่ 2 ด้วยซิซเท็มคอลกลุ่ม chmod

3.3.3.4. กฎสนับสนุนข้อที่ 3

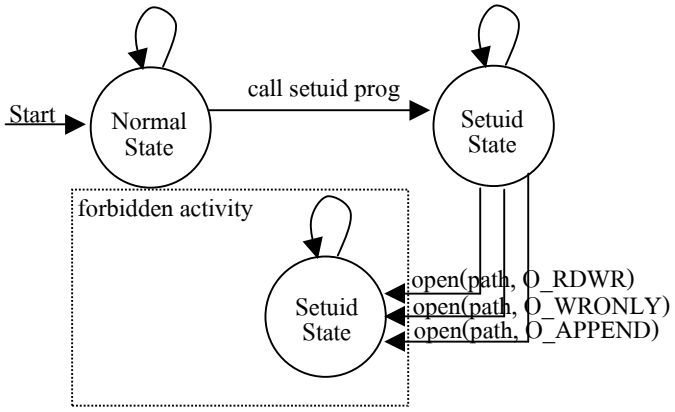
Rule 3: A process is not allowed to modify system programs.

โปรแกรมระบบ (system program) หมายถึงโปรแกรมหรือคำสั่งของระบบปฏิบัติการที่ติดตั้งมากับระบบปฏิบัติการตั้งแต่ขั้นตอนการติดตั้งระบบ เพื่อให้ได้ระบบปฏิบัติการที่เชื่อถือได้ว่าปลอดภัย ต้องมั่นใจว่าที่มาของชุดติดตั้งระบบปฏิบัติการและช่องทางของการดาวน์โหลดต้องน่าเชื่อถือเพียงพอ เช่น ดาวน์โหลดจากเว็บไซต์ของระบบปฏิบัติการโดยผ่านช่องทางที่มีการป้องกันหรือเข้ารหัส และตรวจสอบความถูกต้องของข้อมูลหลังการดาวน์โหลด เมื่อติดตั้งระบบปฏิบัติการเสร็จสมบูรณ์แล้วโปรแกรมระบบจะถูกจัดเก็บไว้ที่ /bin, /sbin, /usr/bin และ /usr/sbin

ผู้ใช้ปกติในระบบไม่จำเป็นต้องแก้ไขโปรแกรมระบบ เว้นแต่ผู้ดูแลระบบที่มีความรู้และต้องการปรับแต่งโปรแกรมให้ตรงตามวัตถุประสงค์ของระบบปฏิบัติการที่จัดตั้งขึ้น กฎสนับสนุนข้อนี้ถูกกำหนดขึ้นมาเพื่อป้องกันการบุกรุกระบบด้วยโปรแกรมโทรจัน rootkits (อธิบายไว้ในหัวข้อที่ 2.2.1.3) หรือวิธีการใดๆ ที่พยายามที่จะแก้ไขคำสั่งของระบบ ตัวอย่างเช่นผู้บุกรุก

อาศัยช่องทางบางประการ เพื่อที่สร้างโปรแกรมสำหรับดักจับรหัสผ่าน หลังจากนั้นแก้ไขคำสั่ง ls ของระบบปฏิบัติการเพื่อมิให้แสดงชื่อโปรแกรมเปล่านั้นขึ้นมา เป็นต้น

การตรวจจับการบุกรุกที่ละเมิดกฎสนับสนุนกฎข้อนี้จะตรวจสอบการเปิดแฟ้มโดยโปรเซสที่อยู่ในสถานะพิเศษ ถ้าหากแฟ้มที่ถูกเปิดขึ้นมาด้วยซิซเท็มคอล open() โดยที่แฟ้มดังกล่าวถูกจัดเก็บอยู่ใน path ของโปรแกรมระบบ และกำหนดค่าพารามิเตอร์ flag ของซิซเท็มคอล open() ให้มีค่าเป็น O_RDWR O_WRONLY หรือ O_APPEND ถือว่าโปรเซสดังกล่าวพยายามที่จะแก้ไขโปรแกรมระบบ การตรวจจับการละเมิดกฎสนับสนุนที่กล่าวมาข้างต้นสามารถแทนได้ด้วยแผนภาพแสดงการเปลี่ยนแปลงสถานะดังภาพประกอบที่ 3.6



ภาพประกอบ 3.6 แผนภาพการเปลี่ยนแปลงสถานะของการตรวจจับการบุกรุกซึ่งละเมิดกฎสนับสนุนข้อที่ 3

3.3.3.5. กฎสนับสนุนข้อที่ 4

Rule 4: Only the superuser should be allowed to create new accounts.

การสร้างชื่อบัญชีผู้ใช้เป็นอีกวิธีการหนึ่งที่ผู้บุกรุกเลือกใช้เมื่อผู้บุกรุกสามารถโจมตีระบบสำเร็จ โปรเซสบุกรุกสามารถแก้ไขแฟ้มต่างๆ ในระบบได้ซึ่งรวมไปถึงฐานข้อมูลบัญชีผู้ใช้ ผู้บุกรุกพยายามที่จะสร้างชื่อบัญชีผู้ใช้ที่มีสิทธิ์เป็น root นั่นคือกำหนดค่า UID ของชื่อบัญชีที่สร้างขึ้นมาใหม่ให้มีค่าเป็น 0 เมื่อผู้บุกรุกติดต่อเข้าสู่ระบบด้วยวิธีการปกติ เช่นติดต่อผ่านโปรแกรม telnet หรือ ssh ด้วยชื่อบัญชีที่ถูกสร้างใหม่ ผู้บุกรุกจะได้รับสิทธิ์เป็น root เสมือนว่าผู้ดูแลระบบ

ติดต่อเข้าสู่ระบบด้วยชื่อบัญชี root บางกรณีผู้บุกรุกอาจไม่ต้องสร้างชื่อบัญชีใหม่ แต่ผู้บุกรุกได้แก้ไขค่า UID ของผู้ใช้ที่มีอยู่แล้วให้มีค่าเป็น 0 ผลของการเข้าสู่ระบบด้วยชื่อบัญชีดังกล่าวจะได้ผลเหมือนกับวิธีการก่อนหน้านี้

กระบวนการตรวจสอบการละเมิดกฎสนับสนุนข้อนี้มีกระบวนการตรวจสอบที่คล้ายคลึงกับการตรวจสอบการละเมิดกฎข้อที่ 3 ในระบบปฏิบัติการเน็ทบีเอสดี ฐานข้อมูลที่เกี่ยวข้องกับชื่อบัญชีผู้ใช้ถูกจัดเก็บไว้ในแฟ้มจำนวนสองแฟ้ม คือ /etc/passwd และ /etc/master.passwd แต่ละแฟ้มนั้นมีหน้าที่แตกต่างกันดังนี้

- **/etc/passwd** เป็นแฟ้มที่ใช้สำหรับจัดเก็บข้อมูลของผู้ใช้จะไม่ถูกนำมาใช้ในกระบวนการพิสูจน์ตัวตน (authentication) แฟ้มดังกล่าวถูกสร้างโดยคำสั่ง /usr/sbin/pwd_mkdb เมื่อเรียกใช้คำสั่งในกลุ่มของการจัดการชื่อบัญชีผู้ใช้
- **/etc/master.passwd** เป็นแฟ้มที่ใช้สำหรับจัดเก็บรหัสผ่านของผู้ใช้ซึ่งใช้สำหรับการติดต่อเข้าสู่ระบบ แฟ้มดังกล่าวอนุญาตให้เฉพาะ root เท่านั้นที่สามารถอ่านและแก้ไขได้

ตารางที่ 3.7 แสดงผลการศึกษาความสัมพันธ์ของแฟ้ม /etc/passwd และ /etc/master.passwd สำหรับการสร้างชื่อบัญชีผู้ใช้

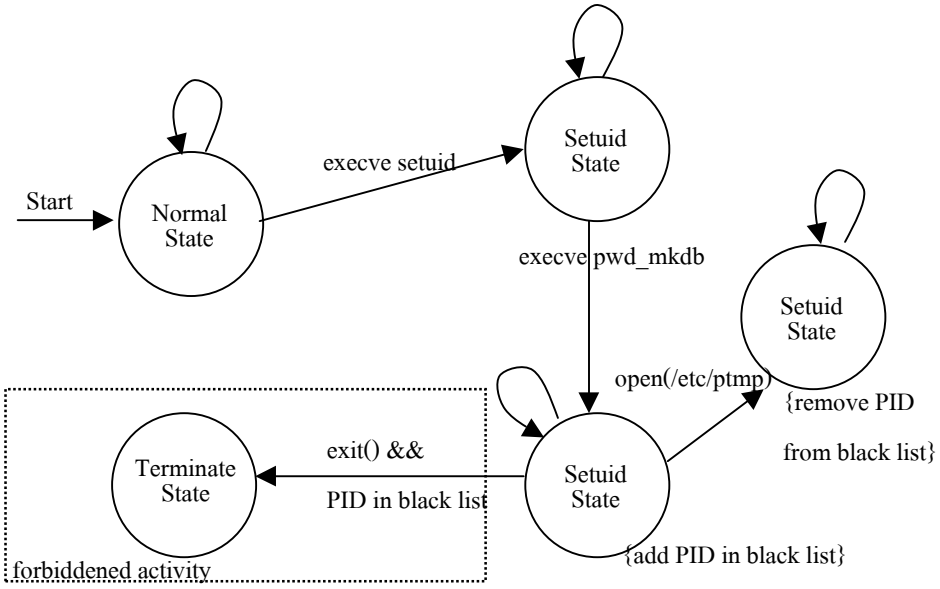
การทดลองที่	/etc/passwd	/etc/master.passwd	กำหนดรหัสผ่าน	ผลการทดลอง
1	เพิ่มชื่อบัญชี		ไม่กำหนด	
2		เพิ่มชื่อบัญชี	ไม่กำหนด	
3		เพิ่มชื่อบัญชี	กำหนด	ใช้เข้าสู่ระบบได้
4	เพิ่มชื่อบัญชี	เพิ่มชื่อบัญชี	ไม่กำหนด	
5	เพิ่มชื่อบัญชี	เพิ่มชื่อบัญชี	กำหนด	ใช้เข้าสู่ระบบได้

ตารางที่ 3.6 เป็นการศึกษาถึงความสัมพันธ์ของแฟ้มทั้งสองสำหรับกระบวนการตรวจสอบตัวตน โดยสร้างกรณีทดสอบจำนวน 5 กรณีทดสอบ โดยแต่ละกรณีมีเงื่อนไขจำนวน 3 ข้อคือ การเพิ่มชื่อบัญชีในแฟ้ม /etc/passwd การเพิ่มชื่อบัญชีในแฟ้ม /etc/master.passwd และการกำหนดรหัสผ่านแก่ชื่อบัญชีนั้น หลังจากนั้นเข้าสู่ระบบโดยใช้ชื่อบัญชีที่สร้างใหม่ ผลการทดลอง

พบว่า ชื่อบัญชีที่ได้จากการทดลองที่ 3 และ 5 สามารถใช้ติดต่อเข้าสู่ระบบได้ แสดงให้เห็นว่า ชื่อบัญชีที่อยู่ในแฟ้ม /etc/master.passwd และได้รับการกำหนดรหัสผ่านเท่านั้นที่สามารถนำมาใช้สำหรับการติดต่อเข้าสู่ ระบบ

การตรวจจับการละเมิดกฎสนับสนุนด้วยการตรวจจับการเรียกใช้ซิซเท็มคอล open() เพื่อแก้ไขแฟ้ม /etc/master.passwd ซึ่งพบว่าการตรวจสอบด้วยวิธีดังกล่าวสามารถตรวจจับการละเมิดกฎข้อนี้ได้ แต่อย่างไรก็ตามวิธีการนี้สร้างผลกระทบแก่คำสั่ง passwd เนื่องจากวิธีการทำงานของคำสั่งดังกล่าวตรงตามเงื่อนไขของวิธีการตรวจจับการบุกรุก แต่เมื่อศึกษาถึงกระบวนการสร้างชื่อบัญชีผู้ใช้จะพบว่า ไม่ว่าจะเป็นการแก้ไขฐานข้อมูลด้วยคำสั่งใดๆ คำสั่งเหล่านั้นจะสั่งงานคำสั่ง pwd_mkdb เพื่อแก้ไขฐานข้อมูล สำหรับขั้นตอนการเพิ่มชื่อบัญชีผู้ใช้ และการเปลี่ยนรหัสผ่าน การทำงานของคำสั่ง pwd_mkdb มีความแตกต่างกัน ในกรณีของการเปลี่ยนรหัสผ่าน คำสั่งดังกล่าวจะคัดลอกแฟ้ม /etc/master.passwd ไปยัง /etc/ptmp แล้วแก้ไขรหัสผ่านในแฟ้ม /etc/ptmp หลังจากนั้นคำสั่ง pwd_mkdb จะคัดลอกแฟ้ม /etc/ptmp กลับไปเป็นแฟ้ม /etc/master.passwd กระบวนการข้างต้นจะพบในคำสั่ง passwd เท่านั้น จะไม่พบในคำสั่งจัดการฐานข้อมูลผู้ใช้อื่นๆ

จากที่กล่าวมาข้างต้น วิธีการตรวจสอบการแก้ไขฐานข้อมูลผู้ใช้สามารถตรวจสอบได้คือ เมื่อโปรเซสเรียกซิซเท็มคอล execve() เพื่อสั่งงานคำสั่ง pwd_mkdb โปรเซสดังกล่าวอาจมีแนวโน้มที่จะบุกรุกแต่ยังไม่สามารถสรุปได้ในทันที จึงบันทึกค่า PID ไว้ตรวจสอบภายหลัง หลังจากนั้นติดตามการทำงานต่อไปจนกระทั่งโปรเซสที่กำลังติดตามเรียกใช้ซิซเท็มคอล open() เพื่อเปิดแฟ้ม /etc/ptmp ซึ่งสรุปได้ว่าโปรเซสดังกล่าวไม่ใช่โปรเซสบุกรุก แต่ถ้าโปรเซสที่ถูกติดตามเรียกใช้ซิซเท็มคอล exit() เพื่อออกจากโปรแกรมนั่นหมายความว่า โปรเซสดังกล่าวเป็นโปรเซสบุกรุก การตรวจจับการละเมิดกฎสนับสนุนข้อนี้สามารถแทนด้วยแผนภาพการเปลี่ยนแปลงสถานะได้ดังภาพประกอบที่ 3.7



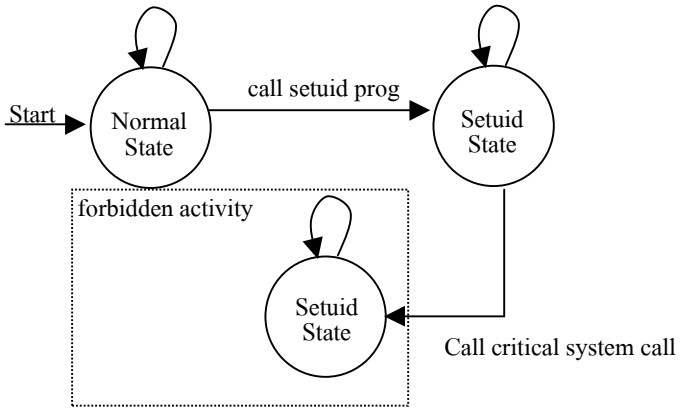
ภาพประกอบ 3.7 แผนภาพการเปลี่ยนแปลงสถานะของการตรวจจับการบุกรุกซึ่งละเมิดกฎสนับสนุนข้อที่ 4

3.3.3.6. กฎสนับสนุนข้อที่ 5

Rule5 : Some system call functions are strictly limited to superuser(root). These system calls are mount(), umount(), nfssvc(), quotactl(), reboot(), settimeofday() and swapon().

ระบบปฏิบัติการเน็ทบีเอสดีมีซิซเท็มคอลบางตัวที่เป็นซิซเท็มคอลวิกฤติ (critical system call) ซึ่งอาจจะสร้างความเสียหายให้แก่ระบบหากนำไปใช้เพื่อประสงค์ร้าย เช่นผู้บุกรุกเรียกใช้ซิซเท็มคอล mount() เพื่อติดตั้งระบบแฟ้ม procfs ของระบบปฏิบัติการในกลุ่มบีเอสดี [Wojtczuk, 2000] เนื่องจาก procfs เป็นส่วนของการจัดการหน่วยความจำของโปรเซส เมื่อผู้บุกรุกสามารถเข้าถึงระบบแฟ้มดังกล่าวสำเร็จ ผู้บุกรุกสามารถอ่านหรือแก้ไขข้อมูลในหน่วยความจำของโปรเซสเป้าหมายได้ หรือบางกรณีผู้บุกบุกเรียกใช้ซิซเท็มคอลเหล่านี้เพื่อทำลายร่องรอยของการบุกรุก ดังนั้นซิซเท็มคอลบางตัวควรจำกัดสิทธิ์ในการเรียกใช้โดยอนุญาตให้เฉพาะ root เท่านั้น ซึ่งได้แก่ซิซเท็มคอล mount(), umount(), nfssvc(), quotactl(), reboot(), settimeofday() และ swapon ()

การตรวจจับการละเมิดกฎสนับสนุนในข้อนี้สามารถแทนเหตุการณ์ได้ด้วยแผนภาพการเปลี่ยนแปลงสถานะในภาพประกอบที่ 3.10 อธิบายได้ว่า ในขณะที่โปรเซสอยู่ในสถานะพิเศษแล้วเรียกใช้ซิซเท็มคอลวิกฤติที่กล่าวมาข้างต้น ถือว่าโปรเซสดังกล่าวเข้าข่ายโปรเซสบุกรุก



ภาพประกอบ 3.8 แผนภาพการเปลี่ยนแปลงสถานะของการตรวจจับการบุกรุกซึ่งละเมิดกฎสนับสนุนข้อที่ 5

3.3.3.7. ซิซเท็มคอลที่มีผลกระทบต่อความปลอดภัยต่อระบบ

ในขณะที่โปรเซสอยู่ในสถานะพิเศษ โปรเซสดังกล่าวมีสิทธิ์ในการเข้าถึงทรัพยากรเป็นครึ่งหนึ่งของ root ดังนั้นจึงจำเป็นต้องติดตามการทำงานและตรวจสอบการเรียกใช้ซิซเท็มคอลต่างๆ จากการศึกษาและวิเคราะห์การเปลี่ยนแปลงสถานะของโปรเซสและกฎสนับสนุนพบว่ามีซิทเท็มคอลบางตัวที่มีผลกระทบต่อความปลอดภัยของระบบ ซิซเท็มคอลเหล่านั้นได้แบ่งกลุ่มออกเป็น 4 กลุ่มตามหน้าที่ของซิซเท็มคอลซึ่งได้แก่

- ซิซเท็มคอลสำหรับการเปลี่ยนแปลงค่า user credential
- ซิซเท็มคอลสำหรับการจัดการโปรเซส
- ซิซเท็มคอลสำหรับการจัดการแฟ้ม
- ซิซเท็มคอลกลุ่มอื่นๆ

สำหรับรายชื่อซิซเท็มคอลแต่ละกลุ่มนั้นจะแสดงไว้ในตารางที่ 3.7 ภายในตารางประกอบไปด้วยกลุ่มของซิซเท็มคอล ชื่อซิซเท็มคอล และเงื่อนไขของการพิจารณาว่าเป็นโปรเซสบุกรุก

ตารางที่ 3.8 แสดงชื่อซิซเท็มคอลที่มีผลต่อกระทบต่อความปลอดภัยของระบบ

ชื่อซิซเท็มคอล	กฎข้อที่	เงื่อนไขของการพิจารณาว่าเป็นโปรเซสบุกรุก
ซิซเท็มคอลสำหรับการเปลี่ยนแปลงค่า user credential		
setuid() seteuid() setgid() setegid()	0	เมื่อโปรเซสเรียกใช้ซิซเท็มคอลเหล่านี้แล้วโปรเซสเปลี่ยนสถานะเข้าสู่สถานะผู้ใช้สูงสุดหรือกลุ่มระบบ
ซิซเท็มคอลสำหรับการจัดการโปรเซส		
execve()	1	เมื่อโปรเซสเรียกใช้ซิซเท็มคอลนี้โดยไม่ต้องตรวจสอบค่าพารามิเตอร์
	4	ถ้าหาก path คือแฟ้ม pwd_mkdb ให้เพิ่มค่า PID ไว้ในลิสต์
ซิซเท็มคอลสำหรับการจัดการแฟ้ม		
open()	2	ตรวจสอบค่าพารามิเตอร์ต่อไปนี้ หากเป็นจริงถือว่าบุกรุก - mode มีค่าเป็น O_CREAT - flag มีค่าเป็น S_ISUID
	3	ตรวจสอบค่าพารามิเตอร์ต่อไปนี้ หากเป็นจริงถือว่าบุกรุก - path คือแฟ้มใดๆ ที่อยู่ในสาระบบ /bin/, /sbin/, /usr/bin และ /usr/sbin/ - mode มีค่าเป็น O_WRONLY, O_RDWR และ O_APPEND
	4	ถ้าหาก path คือ /etc/ptmp ให้ลบ PID ออกจากลิสต์เนื่องจากโปรเซสดังกล่าวเป็นโปรเซสปกติ

ตารางที่ 3.8 (ต่อ) แสดงชื่อซิซเท็มคอลที่มีผลต่อกระทบต่อความปลอดภัยของระบบ

ชื่อซิซเท็มคอล	กฎข้อที่	เงื่อนไขของการพิจารณาว่าเป็นโปรเซสบุกรุก
exit()	4	ถ้าหากโปรเซสเรียกใช้ซิซเท็มคอลนี้ในขณะที่ PID ยังอยู่ในลิสต์ถือว่าโปรเซสดังกล่าวเป็นโปรเซสบุกรุก
chmod() fchmod() lchmod()	3	ตรวจสอบค่าพารามิเตอร์ต่อไปนี้ หากเป็นจริงถือว่าบุกรุก - flag ซึ่งมีค่าเป็น S_ISUID
ซิซเท็มคอลกลุ่มอื่นๆ		
mount() umount() nfssvc() quotactl() reboot () settimeofday() swapon()	5	เมื่อโปรเซสเรียกใช้ซิซเท็มคอลนี้โดยไม่ต้องตรวจสอบค่าพารามิเตอร์

3.4. การวิเคราะห์ข้อมูลนำเข้า

กระบวนการตรวจสอบพฤติกรรมของโปรเซสว่าเป็นโปรเซสบุกรุกหรือไม่นั้น แบ่งออกเป็นสองส่วนหลักคือ การนิยามสถานะ และการตรวจสอบตามกฎสนับสนุน ซึ่งได้กล่าวไปแล้วก่อนหน้าที่ สำหรับการนิยามสถานะต้องการข้อมูลสำหรับการพิจารณาจำนวน 4 ค่าได้แก่ UID, EUID, GID และ EGID ส่วนการพิจารณาตามกฎสนับสนุนนั้นต้องการชื่อซิซเท็มคอลและค่าพารามิเตอร์ ซึ่งใช้ในการตรวจสอบในกรณีที่โปรเซสอยู่ในสถานะพิเศษ

3.5. บทสรุป

เนื้อหาในบทนี้กล่าวถึงการศึกษาและวิเคราะห์ระบบปฏิบัติการเน็ทบีเอสดีและวิธีการตรวจจับการบุกรุก ซึ่งพบว่าโปรแกรมที่มีผลกระทบทางด้านความปลอดภัยคือโปรแกรมแบบ setuid เนื่องจากในบางช่วงเวลาโปรแกรมดังกล่าวทำงานด้วยสิทธิ์ของ root วิทยานิพนธ์ชุดนี้จึง

ศึกษาและวิเคราะห์วิธีการตรวจจับการบุกรุกตามแนวคิดของการวิเคราะห์การเปลี่ยนแปลงสถานะของโปรเซส และสามารถสรุปได้ว่า การระบุว่าโปรเซสใดเป็นโปรเซสบุกรุกนั้นพิจารณาสองกรณี คือ โปรเซสเปลี่ยนจากสถานะปกติเข้าสู่สถานะผู้ใช้สูงสุดและกลุ่มระบบ หรือโปรเซสอยู่ในสถานะพิเศษ และเรียกใช้ซิซเท็มคอลต่างๆ พร้อมกับค่าพารามิเตอร์ตามที่วิเคราะห์ข้างต้น

รายละเอียดของบทถัดไปจึงกล่าวการทดสอบผลการวิเคราะห์ระบบในบทนี้โดยพัฒนาโปรแกรมตรวจจับการบุกรุก และทดสอบโปรแกรมดังกล่าวในประเด็นของความแม่นยำในการตรวจจับและผลกระทบที่อาจจะเกิดขึ้นกับระบบปฏิบัติการเมื่อทดลองสั่งงานโปรแกรมตรวจจับดังกล่าว