บทที่ 6
ประสิทธิภาพของกลไกการตรวจจับการบุกรุก

6.1 บทนำ

ในบทนี้เป็นการทดสอบการทำงานของกลไกการตรวจจับการบุกรุกโดยแบ่งการทดสอบออกเป็น 3 ประเด็นหลักได้แก่ การทดสอบประสิทธิภาพของกลไกการตรวจจับการบุกรุกโดยทดสอบความแม่นยำของการตรวจจับการบุกรุกและความคงทนของการทำงานในสภาวะเครียด การทดสอบในประเด็นต่อมาคือการหาทดสอบเพื่อผลกระทบของการตรวจจับการบุกรุกที่มีผลต่อระบบปฏิบัติการ ท้ายที่สุดเป็นการทดสอบเพื่อวัดประสิทธิภาพของระบบปฏิบัติการหลังจากการแก้ไขซิซเท็มคอล เนื่องจากการแก้ไขซิซเท็มคอลในบทก่อนหน้านั้นอาจจะทำให้ประสิทธิภาพของระบบปฏิบัติการเปลี่ยนแปลง

6.2 การทดสอบประสิทธิภาพของกลไกการตรวจจับการบุกรุก

การทดสอบประสิทธิภาพของการทำงานของกลไกการตรวจจับการบุกรุกเป็นประเด็นหลักของบทนี้ซึ่งแบ่งออกเป็นสองส่วนหลักได้แก่ การทดสอบความแม่นยำของการทำงานและการทดสอบความคงทนของการตรวจจับการบุกรุก การทดสอบความแม่นยำเป็นการประเมินความสามารถในการตรวจจับการบุกรุกของระบบปฏิบัติการ ในบางสภาวะการตรวจจับการบุกรุกอาจจะทำงานไม่เต็มที่หรือไม่สามารถทำงานได้ จึงจำเป็นต้องทดสอบความคงทนของกลไกการตรวจจับการบุกรุกว่ากลไกดังกล่าวสามารถทำงานในทุกสภาวะ สำหรับรายละเอียดของวิธีการทดสอบระบบและผลการทดสอบระบบนั้นจะกล่าวในลำดับถัดไป

6.2.1 การทดสอบความแม่นยำของกลไกการตรวจจับการบุกรุก

การทดสอบความแม่นยำของการตรวจจับการบุกรุกเป็นการวัดความถูกต้องในการทำงานซึ่งแบ่งออกเป็น 3 ขั้นตอน ได้แก่ การทดสอบความแม่นยำในระดับโมดูล การทดสอบความแม่นยำด้วยโปรแกรมบุกรุก และการทดสอบความแม่นยำในสภาวะแวดล้อมจริง

6.2.1.1 การทดสอบความแม่นยำในระดับของโมดูล

การทดสอบความแม่นยำในระดับโมดูลเป็นการทดสอบเพื่อตรวจสอบความถูกต้องของการทำงานของโมดูลต่างๆ จากสถาปัตยกรรมของกลไกการตรวจจับการบุกรุกที่กล่าวถึงในหัวข้อที่ 5.2 ซึ่งแบ่งออกเป็น 3 โมดูลหลักได้แก่ การนิยามสถานะ การพิจารณาตามกฎสนับสนุนและการตอบสนองต่อเหตุการณ์บุกรุก การทดสอบการทำงานของโมดูลต่างๆ มีรายละเอียดดังนี้

การทดสอบโมดูลนิยามสถานะ

การทดสอบในหัวข้อนี้ทดสอบโดยพัฒนาโปรแกรมทดสอบเพื่อเรียกใช้โมดูลนิยามสถานะ หลังจากนั้นทดสอบโมดูลด้วยกรณีทดสอบ เพื่อพิจารณาผลที่ได้จากการดำเนินงานของแต่ละกรณีทดสอบ ผลการทดสอบแสดงไว้ในตารางที่ 6.1

ตารางที่ 6.1 แสดงกรณีทดสอบและผลการทดสอบสำหรับโมดูลการนิยามสถานะ

	ค่า User credentials				ผลการทดสอบ
	UID	EUID	GID	EGID	
1	0	0	0	0	Normal state
2	1000	1000	100	100	Normal state
3	1000	0	100	100	Setreuid state
4	0	1000	100	100	Setuid state
5	1000	1000	100	0	Setgid state
6	1000	1000	0	100	Setregid state
7	0	0	100	100	Super user state
8	1000	1000	0	0	System group state

<u>หมายเหตุ</u> ค่า user credential แต่ละค่ามีความหมายดังนี้

	ผู้ดูแลระบบ	ผู้ใช้ 1
ค่าประจำตัว	0 (root)	1000 (user1)
ค่าประจำกลุ่ม	0 (wheel)	100 (users)

จากผลการทดสอบในตารางที่ 6.1 แสดงให้เห็นว่าโมดูลดังกล่าวทำงานได้ถูกต้องตามที่นิยามไว้ จากการการนิยามสถานะตามงานวิจัยของ Nuansri [Nuansri, 1999] โปรเซสของผู้ดูแลระบบจะถูกกำหนดเป็นสถานะผู้ใช้สูงสุดและกลุ่มระบบ แต่ถ้าหากโปรเซสอยู่ในสถานะดังกล่าวแล้วโปรเซสไม่สามารถดำเนินการใดๆ ได้ อีกทั้งโปรเซสที่มีค่าประจำสถานะเป็น [0, 0, 0, 0] นั้นจะเกิดขึ้นกรณีเดียวคือผู้ใช้ได้เข้าสู่ระบบด้วยชื่อบัญชี root เท่านั้น ดังนั้นจึงจำเป็นที่จะปรับกฎของการนิยามสถานะเพิ่มเติมโดยกำหนดให้ผู้ดูแลระบบซึ่งมีค่าประจำสถานะเป็น [0, 0, 0, 0] ให้อยู่ในสถานะปรกติ สำหรับการพิจารณาสถานะของค่าประจำสถานะอื่นๆ ของโมดูลนิยามสถานะถือว่าถูกต้องการกฎของการนิยามสถานะ

การทดสอบโมดูลพิจารณากฎสนับสนุน

การทดสอบโมดูลพิจารณากฎสนับสนุน เป็นการทดสอบความถูกต้องของการแก้ไขซิซเท็มคอลตามกฎสนับสนุน การทดสอบในหัวข้อนี้จะทดสอบโดยเปลี่ยนสถานะของโปรเซสให้อยู่สถานะพิเศษหลังจากนั้นสั่งงานกรณีทดสอบจำนวน 10 กรณี โดยแต่ละกรณีทดสอบนั้นเป็นการดำเนินการที่ขัดต่อกฎสนับสนุน แสดงไว้ในตารางที่ 6.2

ตารางที่ 6.2 แสดงกรณีทดสอบสำหรับการทดสอบโมดูลพิจารณากฎสนับสนุน

	คำสั่ง	ซิซเท็มคอลที่เรียกใช้	กฎสนับสนุน
1	su	setgid()	0
2	exec bash	execve()	1
3	chmod +s msh	chmod()	2
4	./creat_suid	open()	2
5	cp msh /bin	open()	3
6	vipw	open()	4
7	mount /dev/sd0e /mnt	mount()	5
8	eject /dev/sd0e	umount()	5
9	date 0101051200	settimeofday()	5
10	reboot	reboot()	5

ผลของการทดสอบระบบตามวิธีการข้างต้นพบว่า กลไกการตรวจจับการบุกรุกสามารถตรวจจับการกรณีทดสอบได้ทุกกรณีซึ่งรวมไปถึงกรณีทดสอบที่ 10 ที่ไม่สามารถตรวจจับได้โดยโปรแกรมตรวจจับการบุกรุกในบทก่อนหน้า แต่สำหรับกลไกการตรวจจับการบุกรุกในบทนี้สามารถตรวจจับเหตุการณ์ได้ก่อนที่การบุกรุกจะสำเร็จ เนื่องจากการพิจารณาเหตุการณ์เกิดขึ้นภายในซิซเท็มคอลที่ต้องเฝ้าระวัง และทราบผลการพิจารณาก่อนที่ซิซเท็มคอลจะทำงาน

การทดสอบโมดูลตอบสนองต่อเหตุการณ์บุกรุก

เมื่อโปรเซสที่ถูกติดตามถูกพิจารณาว่าเป็นโปรเซสบุกรุก โมดูลนี้จะทำลายโปรเซสดังกล่าวและโปรเซสอื่นๆ ที่เจ้าของโปรเซสคนเดียวกันโดยพิจารณาจากค่า UID ของโปรเซสนั้น แต่ละกรณีทดสอบประกอบไปด้วยโปรเซสบุกรุกและโปรเซสอื่นๆ มีค่า UID เป็นค่าเดียวกัน ซึ่งได้แก่โปรเซสแม่ โปรเซสลูก และโปรเซสพี่น้อง (sibling process) แต่ละโปรเซสมีจำนวนโปรเซสที่แตกต่างกันในแต่ละกรณีทดสอบ เมื่อโปรเซสบุกรุกเริ่มทำงานกลไกการตรวจจับการบุกรุกจะต้องทำลายโปรเซสทุกโปรเซสตามเงื่อนไขที่กล่าวไปแล้วข้างต้น กรณีทดสอบของโมดูลนี้แสดงในตารางที่ 6.3

ตารางที่ 6.3 แสดงกรณีทดสอบของการทดสอบโมดูลตอบสนอง

กรณี	จำนวนโปรเซสที่เกี่ยวข้อง		
ทดสอบ	โปรเซสแม่	โปรเซสลูก	โปรเซสพี่น้อง
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

จากตารางที่ 6.3 แสดงจำนวนโปรเซสที่เกี่ยวข้องกับโปรเซสบุกรุกในแต่ละกรณีทดสอบซึ่งเป็นการแสดงให้เห็นว่า โปรเซสบุกรุกมีโปรเซสอื่นๆ ที่เกี่ยวข้อง เช่นกรณีทดสอบที่ 1 มีเฉพาะโปรเซสบุกรุกเพียงโปรเซสเดียว ในกรณีทดสอบที่ 7 นอกจากจะมีโปรเซสบุกรุกแล้วยังมี

โปรเซสแม่และโปรเซสลูกของโปรเซสบุกรุกซึ่งมีค่า UID เท่ากัน สำหรับจำนวนโปรเซสของกรณีทดสอบในที่นี้ได้กำหนดไว้เป็น 1 เพื่อใช้ในกระบวนการทดสอบเท่านั้น แต่สำหรับสถานการณ์จริงโมดูลตอบสนองนี้สามารถทำลายโปรเซสได้มากกว่า 1 โปรเซส เนื่องจากสำหรับกระบวนการทำงานนั้นจะดำเนินการแบบเรียกตัวเอง จากการทดสอบตามกรณีทดสอบข้างต้น กลไกการตรวจจับการบุกรุกสามารถทำงานได้ถูกต้องตามเงื่อนไข

แต่บางกรณีโปรเซสบุกรุกอาจจะป้องกันการทำลายโปรเซสโดยพัฒนาโปรแกรมเพื่อตรวจสอบสัญญาณจากระบบปฏิบัติการ ถ้าโปรเซสบุกรุกได้รับสัญญาณชื่อ SIGKILL (หมายถึงการทำลายโปรเซสที่ได้รับสัญญาณ) แล้วโปรเซสบุกรุกจะสร้างโปรเซสบุกรุกใหม่ขึ้นมาทันที การทำลายโปรเซสในกรณีนี้เกิดเหตุการณ์ที่เรียกว่า race condition ระหว่างกระบวนการทำลายโปรเซสและกระบวนการสร้างโปรเซสของระบบปฏิบัติการ แต่เมื่อจำลองเหตุการณ์ดังกล่าวกลไกตรวจจับการบุกรุกสามารถทำลายโปรเซสเหล่านั้นได้แม้ว่าโปรเซสบุกรุกได้ป้องกันตามวิธีการข้างต้น เนื่องจากกระบวนการสร้างโปรเซสใช้เวลาในการดำเนินการมากกว่ากระบวนการทำลายโปรเซส นั่นคือ ในกระบวนการสร้างโปรเซส ระบบปฏิบัติการต้องตรวจสอบว่าทรัพยากรของเคอร์เนลว่างหรือไม่ จองเนื้อที่สำหรับสร้างตารางโปรเซส ตรวจสอบว่าในระบบปฏิบัติการมีโปรเซสทำงานอยู่มากกว่าที่กำหนดหรือไม่ คัดลอกตารางโปรเซสของโปรเซสแม่ไปยังโปรเซสใหม่ คัดลอกโครงสร้างของโปรเซสโปรเซสแม่ไปยังโปรเซสลูก ท้ายที่สุดเปลี่ยนสถานะของโปรเซสลูกเข้าสู่สถานะพร้อม (ready state) ในขณะที่กระบวนการทำลายโปรเซสจะเกิดขึ้นเมื่อได้รับสัญญาณ SIGKIL โปรเซสเป้าหมายจะคืนทรัพยากรทุกอย่างที่จองไว้ แล้วส่งสัญญาน SIG_CHILD ไปยังโปรเซสแม่เพื่อแจ้งให้ทราบว่าโปรเซสลูกได้ตายไปแล้ว เมื่อพิจารณาจากลำดับการทำงานของทั้งสองกระบวนการข้างต้น อีกทั้งการทดสอบโดยการพัฒนาโปรแกรมจึงสรุปได้ว่า กระบวนการสร้างโปรเซสใช้เวลามากกว่ากระบวนการทำลาย ดังนั้นระบบปฏิบัติการชุดใหม่นี้จึงสามารถทำลายโปรเซสบุกรุกที่เกิดขึ้นได้ทั้งหมดทุกกรณี [Bach, 1986]

จากการทดสอบการความแม่นยำของกลไกการตรวจจับการบุกรุกในระดับของโมดูลทั้งโมดูลกฎนิยามสถานะ โมดูลพิจารณากฎสนับสนุน และโมดูลตอบสนองจึงสรุปได้ว่ากลไกตรวจจับการบุกรุกสามารถทำงานได้อย่างถูกต้องตามที่ได้ออกแบบระบบไว้ทุกประการ

6.2.1.2 การทดสอบด้วยโปรแกรมบุกรุก

จากผลของการศึกษาถึงผลกระทบของการบุกรุกระบบในหัวข้อที่ 2.2.2 และการจัดอนุกรมวิธานของการบุกรุกในหัวข้อที่ 2.3 วิทยานิพนธ์ชุดนี้จึงจัดจำแนกวิธีการบุกรุกระบบคอมพิวเตอร์ตามผลกระทบที่เกิดขึ้นซึ่งครอบคลุมเฉพาะการบุกรุกคอมพิวเตอร์หรือ host-based intrusion เท่านั้น ไม่ครอบคลุมการโจมตีคอมพิวเตอร์ผ่านเครือข่าย ซึ่งแบ่งออกเป็น 3 กลุ่ม ได้แก่

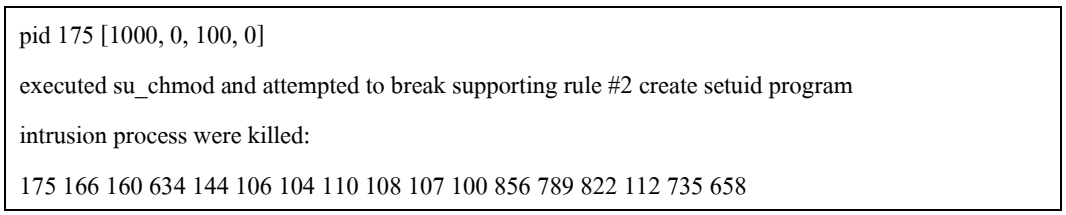
1. การบุกรุกที่พยายามเปลี่ยนสิทธิ์ของผู้บุกรุกให้เทียบเท่ากับผู้ดูแลระบบหรือเปลี่ยนค่า UID เป็น 0 ในระบบปฏิบัติการยูนิกซ์
2. การบุกรุกที่พยายามดำเนินการกับแฟ้มของระบบ เช่น อ่าน เขียน แก้ไข หรือการเปลี่ยนสิทธิ์การเข้าถึงแฟ้มเป้าหมาย เป็นต้น
3. การบุกรุกที่พยายามติดตั้งโปรแกรมโทรจัน โปรแกรม rootkits หรือก่อกวนระบบจนระบบไม่สามารถทำงานได้ตามปกติ

การทดสอบฟังก์ชันตรวจจับการบุกรุกในหัวข้อนี้ได้จัดกรณีทดสอบซึ่งเป็นโปรแกรมบุกรุกจำนวน 12 โปรแกรม โดยจัดจำแนกตามวิธีการข้างต้นซึ่งแสดงไว้ในตารางที่ 6.4

ตารางที่ 6.4 แสดงกรณีทดสอบของการทดสอบด้วยโปรแกรมบุกรุกซึ่งจัดจำแนกตามผลกระทบ

ผลกระทบของการบุกรุก	ซิซเท็มคอลที่เรียกใช้	จำนวนโปรแกรมบุกรุก
ผู้บุกรุกได้รับสิทธิ์ของ root	execve()	10
	setuid()	1
ผู้บุกรุกแก้ไขสิทธิ์การเข้าถึงแฟ้ม	chmod()	1

จากตารางที่ 6.4 เป็นการจัดจำแนกกรณีทดสอบตามผลกระทบที่อาจจะเกิดขึ้นถ้าหากการบุกรุกระบบประสบความสำเร็จ ซึ่งพบว่ามีกรณีทดสอบจำนวน 10 กรณีทดสอบพยายามที่จะเปลี่ยนสิทธิ์ของโปรเซสบุกรุกให้เทียบเท่ากับ root โดยสั่งงานโปรเซส /bin/sh ด้วยซิซเท็มคอล execve() หรือเรียกใช้ซิซเท็มคอล setuid() เพื่อเปลี่ยนสถานะเข้าสู่สถานะผู้ใช้สูงสุดและสั่งงานโปรเซส /bin/sh นอกจากนี้มีกรณีทดสอบที่พยายามเปลี่ยนสิทธิ์ของแฟ้ม /bin/sh เป็นโปรแกรมแบบ setuid เพื่อที่จะสั่งงานคำสั่งอื่นๆ ภายหลัง ตัวอย่างผลการทำงานของกลไกตรวจจับการบุกรุกแสดงไว้ในภาพประกอบที่ 6.1



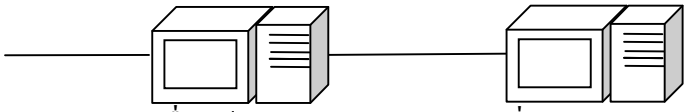
ภาพประกอบ 6.1 ผลการทำงานของกลไกการตรวจจับการบุกรุก

จากภาพประกอบที่ 6.1 แสดงผลการตรวจจับการบุกรุก อธิบายได้ว่า โปรเซสหมายเลข 175 ซึ่งอยู่ในสถานะพิเศษ [1000, 0, 100, 0] พยายามที่จะกระทำการใดๆ ที่ขัดกับกฎสนับสุนนข้อที่ 2 โดยสร้างโปรแกรมแบบ setuid กลไกการตรวจจับการบุกรุกทำลายโปรเซสและโปรเซสที่เกี่ยวข้องดังแสดงไว้ในภาพประกอบที่ 6.1

เมื่อสั่งงานกรณีทดสอบตามกรณีข้างต้นและกลไกการตรวจจับการบุกรุกสามารถตรวจจับการทำงานของกรณีทดสอบได้ทั้งหมดจึงสรุปได้ว่า กลไกการตรวจจับการบุกรุกของระบบปฏิบัติการชุดนี้สามารถตรวจจับการบุกรุกได้ทุกวิธีการที่ละเมิดต่อกฎสนับสนุน

6.2.1.3 การทดสอบระบบในสภาวะแวดล้อมจริง

การทดสอบการทำงานในหัวข้อนี้เป็นการเปิดระบบให้บุคคลอื่นร่วมทดสอบโดยจัดตั้งเครื่องให้บริการซึ่งติดตั้งระบบปฏิบัติการที่ได้รับการแก้ไขและอนุญาตให้บุคคลภายนอกตรวจหาช่องทางและจุดอ่อนของระบบปฏิบัติการ ก่อนที่จะดำเนินการข้างต้นจำเป็นที่จะต้องป้องกันข้อมูลสำคัญบางส่วนเช่น ล็อกไฟล์ของระบบปฏิบัติการ ข้อมูลชุดนี้ป้องกันได้โดยจัดตั้ง log server ซึ่งมีหน้าที่สำหรับจัดเก็บล็อกไฟล์ไว้ในกรณีที่ไม่สามารถกู้ข้อมูลจากเครื่องเป้าหมายได้ อีกทั้งต้องป้องกันข้อมูลในเครื่อง log server มิให้ถูกทำลายเช่นกัน การทดสอบระบบในส่วนนี้จึงจัดตั้งเครื่องคอมพิวเตอร์จำนวน 2 ชุดโดยแต่ละเครื่องมีการเชื่อมต่อและติดตั้งโปรแกรมดังภาพประกอบที่ 6.2



เครื่องเป้าหมาย เครื่อง Log server

IP Address	172.25.3.111	172.25.3.48
Operating system	NetBSD (modified version)	NetBSD (modified version)
Service	Apache, FTP, SSH, and MySQL	Syslogd, Firewall

ภาพประกอบ 6.2 การจัดเตรียมระบบเพื่อทดสอบการทำงานของกลไกการตรวจจับการบุกรุก

จากภาพประกอบที่ 6.2 แสดงการจัดเตรียมเครื่องคอมพิวเตอร์สำหรับการทดสอบ โดยแบ่งเครื่องคอมพิวเตอร์จำนวน 2 ชุดโดยเครื่องแรกหมายเลข IP 172.25.3.111 เป็นเครื่องเป้าหมายซึ่งติดตั้งระบบปฏิบัติการที่ได้รับการแก้ไข พร้อมทั้งให้บริการข้อมูลทั้งในรูปแบบของเว็บไซต์ แฟ้ม โปรแกรม ssh จดหมายอิเล็กทรอนิกส์ และฐานข้อมูล MySQL นอกจากนี้สร้างชื่อบัญชี test เพื่อเพิ่มความสะดวกให้แก่ผู้ทดสอบ แต่ไม่ประกาศรหัสผ่านของชื่อบัญชีดังกล่าว

สำหรับเครื่อง log server ซึ่งมีหมายเลข IP เป็น 172.25.3.48 เครื่องดังกล่าวได้ติดตั้งให้บริการสำรองข้อมูลล็อก และติดตั้งโปรแกรมไฟร์วอลซึ่งอนุญาตให้เครื่องหมายเลข IP 172.25.3.111ติดต่อเข้าสู่ระบบผ่านพอร์ตของ syslogd เท่านั้น

การทดสอบระบบในประเด็นนี้ได้ทดสอบเมื่อวันที่ 21 มีนาคม พ.ศ. 2549 เวลา 12.00 น. ถึง วันที่ 23 มีนาคม พ.ศ. 2549 เวลา 11.59 น. ผลการทดสอบสังเกตได้จากข้อมูลในล็อกไฟล์ของระบบปฏิบัติการ และโปรแกรมประยุกต์ต่างๆ ดังต่อไปนี้

- /var/log/vsftpd.log สำหรับจัดเก็บผลการทำงานของโปรแกรม vsftpd
- /var/log/authlog สำหรับจัดเก็บผลของการเข้าสู่ระบบ
- /var/log/message สำหรับจัดเก็บผลการทำงานของระบบปฏิบัติการ

รายละเอียดที่แสดงต่อไปนี้เป็นผลของการทดสอบระบบซึ่งตัดรายละเอียดบางส่วนที่ไม่เกี่ยวข้องกับการทดสอบออกไป เช่น ผลการทำงานของการเข้าสู่ระบบจากเครื่องทดสอบหรือเหตุการณ์ที่รายงานโดยระบบปฏิบัติการ แต่ไม่เกี่ยวข้องกับกลไกการตรวจจับการบุกรุก เป็นต้น ซึ่งเป็นข้อมูลล็อกของแฟ้ม vsftpd.log, authlog และ messages ตามลำดับ

1. **Tue Mar 21 21:54:31 2006 [pid 13054] [ftp] OK LOGIN: Client "58.136.100.103", anon password "IEUser@"**
2. Tue Mar 21 22:32:19 2006 [pid 11093] [test] FAIL LOGIN: Client "58.136.100.103"
3. Tue Mar 21 22:32:41 2006 [pid 14889] [test] FAIL LOGIN: Client "58.136.100.103"
4. Tue Mar 21 22:33:25 2006 [pid 28123] [test] FAIL LOGIN: Client "58.136.100.103"
5. **Wed Mar 22 00:35:57 2006 [pid 14225] [test] OK LOGIN: Client "61.7.157.128"**
6. Wed Mar 22 00:38:58 2006 [pid 16378] [admin] FAIL LOGIN: Client "61.7.156.202"
7. Wed Mar 22 00:38:58 2006 [pid 26310] [admin] FAIL LOGIN: Client "61.7.156.202"
8. Wed Mar 22 00:38:58 2006 [pid 28114] [admin] FAIL LOGIN: Client "61.7.156.202"
9. Wed Mar 22 00:38:58 2006 [pid 937] [admin] FAIL LOGIN: Client "61.7.156.202"
10. Wed Mar 22 00:38:58 2006 [pid 28608] [admin] FAIL LOGIN: Client "61.7.156.202"
11. Wed Mar 22 00:38:58 2006 [pid 5418] [admin] FAIL LOGIN: Client "61.7.156.202"
12. Wed Mar 22 00:38:59 2006 [pid 7272] [admin] FAIL LOGIN: Client "61.7.156.202"
13. Wed Mar 22 00:38:59 2006 [pid 28615] [admin] FAIL LOGIN: Client "61.7.156.202"
14. Wed Mar 22 00:38:59 2006 [pid 28149] [admin] FAIL LOGIN: Client "61.7.156.202"
15. Wed Mar 22 00:38:59 2006 [pid 1065] [admin] FAIL LOGIN: Client "61.7.156.202"
16. Wed Mar 22 00:38:59 2006 [pid 28836] [admin] FAIL LOGIN: Client "61.7.156.202"
17. Wed Mar 22 00:39:08 2006 [pid 21015] [admin] FAIL LOGIN: Client "61.7.157.128"
18. Wed Mar 22 00:39:08 2006 [pid 29947] [admin] FAIL LOGIN: Client "61.7.157.128"
19. Wed Mar 22 00:43:25 2006 [pid 4088] [test] FAIL LOGIN: Client "61.7.157.128"
20. Wed Mar 22 00:43:25 2006 [pid 13800] [test] FAIL LOGIN: Client "61.7.157.128"
21. Wed Mar 22 00:43:25 2006 [pid 27287] [test] FAIL LOGIN: Client "61.7.157.128"
22. Wed Mar 22 00:43:28 2006 [pid 12010] [test] FAIL LOGIN: Client "61.7.157.128"
23. **Wed Mar 22 16:08:22 2006 [pid 17309] [ftp] OK LOGIN: Client "203.113.76.11", anon password "IEUser@"**
24. **Wed Mar 22 16:09:21 2006 [pid 18402] [test] OK LOGIN: Client "203.113.76.11"**
25. Wed Mar 22 16:15:28 2006 [pid 11219] [test] FAIL LOGIN: Client "203.113.76.11"
26. Wed Mar 22 20:44:43 2006 [pid 12061] [root] FAIL LOGIN: Client "61.7.156.202"
27. Wed Mar 22 20:44:43 2006 [pid 5390] [root] FAIL LOGIN: Client "61.7.156.202"
28. Wed Mar 22 20:44:57 2006 [pid 9376] [root] FAIL LOGIN: Client "61.7.156.202"
29. Wed Mar 22 20:44:57 2006 [pid 26271] [root] FAIL LOGIN: Client "61.7.156.202"
30. Wed Mar 22 20:44:57 2006 [pid 2714] [root] FAIL LOGIN: Client "61.7.156.202"

ภาพประกอบ 6.3 ผลการทดสอบระบบในแฟ้ม /var/log/vsftpd.log

31. Wed Mar 22 20:45:01 2006 [pid 13718] [root] FAIL LOGIN: Client "61.7.156.202"
32. Wed Mar 22 20:46:48 2006 [pid 29373] [test] FAIL LOGIN: Client "61.7.157.128"
33. Wed Mar 22 20:46:48 2006 [pid 13098] [test] FAIL LOGIN: Client "61.7.157.128"
34. Wed Mar 22 20:46:48 2006 [pid 24424] [test] FAIL LOGIN: Client "61.7.157.128"
35. Wed Mar 22 20:46:48 2006 [pid 9896] [test] FAIL LOGIN: Client "61.7.157.128"
36. Wed Mar 22 20:46:48 2006 [pid 11128] [root] FAIL LOGIN: Client "61.7.157.128"
37. Wed Mar 22 20:46:48 2006 [pid 16408] [root] FAIL LOGIN: Client "61.7.157.128"
38. Wed Mar 22 20:46:48 2006 [pid 11180] [test] FAIL LOGIN: Client "61.7.157.128"
39. Wed Mar 22 20:46:49 2006 [pid 5092] [test] FAIL LOGIN: Client "61.7.157.128"
40. Wed Mar 22 20:46:49 2006 [pid 1833] [test] FAIL LOGIN: Client "61.7.157.128"
41. Wed Mar 22 20:46:49 2006 [pid 18467] [test] FAIL LOGIN: Client "61.7.157.128"
42. Wed Mar 22 20:46:49 2006 [pid 19909] [test] FAIL LOGIN: Client "61.7.157.128"
43. Wed Mar 22 20:46:49 2006 [pid 29756] [test] FAIL LOGIN: Client "61.7.157.128"
44. Wed Mar 22 20:46:49 2006 [pid 13988] [test] FAIL LOGIN: Client "61.7.157.128"
45. Wed Mar 22 20:46:49 2006 [pid 5675] [root] FAIL LOGIN: Client "61.7.157.128"
46. Wed Mar 22 20:46:49 2006 [pid 4719] [root] FAIL LOGIN: Client "61.7.157.128"
47. Wed Mar 22 20:46:49 2006 [pid 14226] [root] FAIL LOGIN: Client "61.7.157.128"
48. Wed Mar 22 20:46:50 2006 [pid 7793] [root] FAIL LOGIN: Client "61.7.157.128"
49. Wed Mar 22 20:46:50 2006 [pid 11169] [root] FAIL LOGIN: Client "61.7.157.128"
50. Wed Mar 22 20:46:59 2006 [pid 60] [root] FAIL LOGIN: Client "61.7.157.128"
51. Wed Mar 22 20:47:00 2006 [pid 17535] [root] FAIL LOGIN: Client "61.7.157.128"
52. Wed Mar 22 20:47:00 2006 [pid 15842] [root] FAIL LOGIN: Client "61.7.157.128"
53. Wed Mar 22 20:47:00 2006 [pid 20929] [root] FAIL LOGIN: Client "61.7.157.128"
54. Wed Mar 22 20:47:00 2006 [pid 14923] [root] FAIL LOGIN: Client "61.7.157.128"
55. Wed Mar 22 20:47:00 2006 [pid 6959] [root] FAIL LOGIN: Client "61.7.157.128"
56. Wed Mar 22 20:47:00 2006 [pid 16468] [root] FAIL LOGIN: Client "61.7.157.128"
57. **Wed Mar 22 20:48:50 2006 [pid 11460] [test] OK LOGIN: Client "61.7.157.128"**
58. **Wed Mar 22 20:49:39 2006 [pid 19276] [test] OK LOGIN: Client "61.7.157.128"**
59. **Wed Mar 22 20:50:14 2006 [pid 15520] [ftp] OK LOGIN: Client "61.7.156.202", anon password "IEUser@"**
60. **Wed Mar 22 20:50:26 2006 [pid 21375] [ftp] OK LOGIN: Client "61.7.156.202", anon password "IEUser@"**

ภาพประกอบ 6.3 (ต่อ) ผลการทดสอบระบบในแฟ้ม /var/log/vsftpd.log

1. Mar 21 16:23:33 victim last message repeated 3 times
2. Mar 21 23:32:16 victim sshd[9448]: Failed password for test from 58.8.192.230 port 3670 ssh2
3. Mar 21 23:32:43 victim last message repeated 2 times
4. Mar 21 23:33:03 victim sshd[26463]: Failed password for test from 58.8.192.230 port 3781
5. Mar 21 23:33:17 victim sshd[26463]: Failed password for test from 58.8.192.230 port 3781
6. **Mar 22 00:46:52 victim sshd[11944]: Accepted password for test from 61.7.157.128 port 1809**
7. Mar 22 00:46:56 victim su: BAD SU test to root on /dev/ttyp0: authentication error
8. Mar 22 00:47:18 victim su: BAD SU test to root on /dev/ttyp0: authentication error
9. Mar 22 07:56:06 victim sshd[19144]: fatal: Timeout before authentication for 61.8.153.114
10. Mar 22 07:59:02 victim sshd[10156]: Did not receive identification string from 211.157.104.19
11. Mar 22 11:12:15 victim sshd[3237]: Did not receive identification string from 61.91.101.207
12. **Mar 22 16:10:38 victim sshd[13939]: Accepted keyboard-interactive/pam for test from 203.113.76.11 port 1611 ssh2**
13. Mar 22 16:15:39 victim su: BAD SU test to root on /dev/ttyp0: authentication error
14. **Mar 22 20:51:22 victim sshd[9237]: Accepted password for test from 61.7.156.202 port 1594**
15. Mar 22 20:52:08 victim su: BAD SU test to root on /dev/ttyp0: authentication error
16. **Mar 22 20:52:32 victim sshd[16609]: Accepted password for test from 61.7.156.202 port 1596**
17. Mar 22 23:26:15 victim sshd[17224]: Did not receive identification string from 210.208.231.21

ภาพประกอบ 6.4 ผลการทดสอบระบบในแฟ้ม /var/log/authlog

จากภาพประกอบที่ 6.2 และ 6.3 เป็นบันทึกการทำงานของโปรแกรม vsftp และกระบวนการตรวจสอบตัวตนของระบบปฏิบัติการ ภาพประกอบที่ 6.2 แสดงผลการพยายามติดต่อเข้าสู่ระบบผ่านบริการ ftp ซึ่งแสดงให้เห็นว่า (บรรทัดที่ 2-22 และ 25-56) ผู้ทดสอบพยายามเดารหัสผ่านของชื่อบัญชี root, admin และ test ซึ่งพบว่าในช่วงเวลาสั้นๆ ผู้ทดสอบอาจจะใช้เครื่องมือบางอย่างสุ่มรหัสผ่านของชื่อบัญชีข้างต้นสำหรับการติดต่อกับระบบปฏิบัติการ แต่ผลการกระทำดังกล่าวไม่ประสบผลสำเร็จโดยสังเกตจาก ข้อความ “[root] FAIL LOGIN: Client "61.7.157.128" ซึ่งแสดงว่าชื่อบัญชี root เข้าสู่ระบบผ่านเครื่อง 61.7.157.128 ไม่สำเร็จ แต่สำหรับผลการทดสอบในบรรทัดที่ 1, 5, 23 และ 57-60 นั้นผู้ทดสอบสามารถสุ่มรหัสผ่านของชื่อบัญชี test และ ftp ได้ แต่ชื่อ

บัญชี ftp นั้นใช้สำหรับอนุญาตให้ผู้ใช้แบบ anonymous เข้าใช้ระบบจึงไม่ก่อให้เกิดผลกระทบทางด้านความปลอดภัย แต่สำหรับชื่อบัญชี test นั้น เป็นชื่อบัญชีของผู้ใช้ระบบ ผู้ทดสอบได้ทำการทดสอบระบบต่อไปโดยใช้ชื่อบัญชี test และรหัสผ่านที่สุ่มได้ติดต่อเข้าสู่ระบบผ่านบริการ ssh หรือ telnet ซึ่งแสดงรายละเอียดของการทำงานไว้ในภาพประกอบที่ 6.4

ภาพประกอบที่ 6.4 แสดงผลการติดต่อเข้าสู่ระบบผ่านบริการของ sshd และ telnet ซึ่งแสดงได้เห็นว่า ผู้ทดสอบพยายามที่จะติดต่อเข้าสู่ระบบโดยการสุ่มรหัสผ่านของชื่อบัญชี test จากผลการกระทำข้างต้น ผู้ทดสอบไม่สามารถเข้าสู่ระบบได้ ยกเว้นในบรรทัดที่ 6, 12, 14 และ 16 ผู้ทดสอบสามารถติดต่อเข้าสู่ระบบได้ ด้วยชื่อบัญชี test เมื่อพิจารณาผลการทดสอบต่อไปจะพบว่าในแต่ละบรรทัด เมื่อผู้ทดสอบเข้าสู่ระบบได้แล้ว ผู้ทดสอบพยายามที่จะเปลี่ยนสิทธิ์ของชื่อบัญชี test ให้เป็น root ด้วยคำสั่ง su แต่จากการทดสอบข้างต้น ผู้ทดสอบไม่สามารถดำเนินการได้สำเร็จ เนื่องจากผู้ทดสอบป้อนรหัสผ่านของ root ผิดพลาดโดยสังเกตจากข้อความ “su: BAD SU test to root on /dev/ttyp0: authentication error” แต่ในที่สุดผลการทดสอบในบรรทัดที่ 16 ผู้ทดสอบสามารถที่จะเปลี่ยนสิทธิ์ของผู้ใช้เป็น root ด้วยคำสั่ง su ได้ แต่กระบวนการข้างต้นถูกทำลายโดยกลไกการตรวจจับการบุกรุก เนื่องจากขัดต่อกฎสนับสนุนข้อที่ 0 ซึ่งดังแสดงผลการทำงานไว้ดังต่อไปนี้

```
Mar 22 20:52:16 victim /netbsd: pid 7464 [1003, 0, 0, 0] executed su and attempted to break supporting rule #0
call setuid/setgid
Mar 22 20:52:16 victim /netbsd: intrusion process were killed:7464 14047 22327
```

จากผลการทดสอบความแม่นยำของกลไกการตรวจจับการบุกรุกในหัวข้อที่ 6.2.1 โดยทดสอบในระดับของโมดูล ทดสอบด้วยโปรแกรมบุกรุก และทดสอบในสถานการณ์จริงจึงสรุปได้ว่า ระบบปฏิบัติการที่ได้รับการเพิ่มสมรรถนะทางด้านความปลอดภัยสามารถตรวจจับการบุกรุกได้ตามเงื่อนไขและกรณีทดสอบได้ทุกประการ

6.2.2 การทดสอบความคงทนของกลไกตรวจจับการบุกรุก

กระบวนการตรวจจับการบุกรุกจะต้องทำงานตลอดเวลาและคงทนต่อการรบกวนจากเหตุการณ์ต่างๆ เช่น กรณีของทรัพยากรของระบบปฏิบัติการเหลืออยู่น้อย มีจำนวนโปรเซสบุกรุกมากกว่าหนึ่งโปรเซส เป็นต้น การทดสอบระบบในหัวข้อนี้เป็นการทดสอบการทำงานของระบบในขณะที่เกิดเหตุการณ์ทั้งสองเงื่อนไขซึ่งจะกล่าวในลำดับถัดไป

6.2.2.1 การทดสอบระบบเมื่อทรัพยากรของระบบเหลืออยู่น้อย

โปรเซสในระบบปฏิบัติการใช้ทรัพยากรร่วมกันโดยมีหน่วยประมวลผลกลางทำหน้าที่จัดลำดับของโปรเซสเพื่อที่จะเข้าใช้ทรัพยากร ถ้าหากในขณะนั้นมีโปรเซสจำนวนมากทำงานพร้อมกัน หน่วยประมวลผลกลางจะทำงานหนักขึ้น การใช้งานหน่วยความจำหลักย่อมสูงขึ้นเช่นกัน จากสถานการณ์ข้างต้น หากเกิดการบุกรุกระบบแล้วกลไกการตรวจจับการบุกรุกของระบบปฏิบัติการจะต้องทำงานได้ตามปกติ

การทดสอบชุดนี้สร้างสภาวะเครียดให้แก่ระบบปฏิบัติการโดยสั่งงานโปรแกรมสำหรับคำนวณหาผลคูณเมทริกซ์ขนาด 400x400 จำนวน 156 โปรเซส และเปรียบเทียบปริมาณทรัพยากรของระบบปฏิบัติการทั้งก่อนและหลังสั่งงานโปรเซสคูณเมทริกซ์ ไว้ในตารางที่ 6.5

ตารางที่ 6.5 เปรียบทรัพยากรของระบบปฏิบัติการระหว่างเหตุการณ์ปรกติและสภาวะเครียด

ทรัพยากร	สภาวะปรกติ	สภาวะเครียด
load average	0.06	156.91
จำนวนโปรเซสที่กำลังทำงาน	28	184
หน่วยความจำที่ถูกใช้ไป	25 MB	213 MB
หน่วยความจำที่เหลือ	198 MB	2576 KB

จากตารางที่ 6.5 อธิบายได้ว่า ในสภาวะปกติ ระบบปฏิบัติการมีค่า load average (ซึ่งเป็นค่าที่บ่งบอกถึงระดับของการใช้งานหน่วยประมวลผลกลาง) เป็น 0.06 มีโปรเซสกำลังทำงานอยู่จำนวน 28 โปรเซส หน่วยความจำหลักของระบบปฏิบัติการถูกใช้ไป 25 MB จากจำนวนทั้งหมด 256 MB เมื่อสั่งงานโปรเซสคูณเมทริกซ์จำนวน 156 โปรเซสแล้ว สภาวะของระบบปฏิบัติการมีความเปลี่ยนแปลงอย่างเห็นได้ชัดซึ่งได้แก่ ในสภาวะเครียดหน่วยประมวลผลกลางทำงานหนักกว่าสถานะปรกติถึง 2600 เท่าโดยสังเกตจากค่า load average เนื่องจากหน่วยประมวลผลกลางต้องคำนวณหาผลคูณและจัดการกับโปรเซสจำนวน 184 โปรเซส หน่วยความจำหลักถูกใช้ไป 213 MB และเหลืออยู่ประมาณ 3 MB นอกจากนี้ในขณะที่โปรเซสคำนวณดังกล่าวกำลังทำงานอยู่นั้น การปฏิสัมพันธ์ระหว่างผู้ใช้กับระบบปฏิบัติการจะหยุดชะงักและก่อให้เกิดผลกระทบด้านการแสดงผลการทำงานซึ่งจะแสดงได้ช้ากว่าปรกติ

เมื่อสั่งงานโปรเซสบุกรุกในโปรเซสคำนวณเมทริกซ์กำลังทำงานนั้น กลไกการตรวจจับการบุกรุกสามารถตรวจจับเหตุการณ์ที่เกิดขึ้นและทำลายโปรเซสบุกรุกได้ แต่การแสดงผลการทำงานทางจอภาพจะเกิดขึ้นช้าลงกว่าปกติอย่างเห็นได้ชัด

6.2.2.2 การทดสอบระบบเมื่อมีโปรเซสบุกรุกจำนวนมาก

การทดสอบระบบเมื่อมีโปรเซสบุกรุกจำนวนมาก เป็นการทดสอบประสิทธิภาพของกลไกการตรวจจับผู้บุกรุกโดยทดสอบความสามารถในการตรวจจับการโปรเซสบุกรุก ในขณะที่โปรเซสบุกรุกมีหลายโปรเซสและโจมตีระบบพร้อมกัน

การทดสอบระบบในหัวข้อนี้ได้สร้างสร้างเหตุการณ์จำลองขึ้นมาโดยสั่งงานโปรเซสบุกรุกให้ทำงานพร้อมกันจำนวน 20 โปรเซส โปรเซสบุกรุกแต่ละโปรเซสมีค่า UID ที่แตกต่างกัน

ผลของการทดสอบพบว่า เมื่อสั่งงานโปรเซสบุกรุกพร้อมกันหลายโปรเซส กลไกตรวจจับการบุกรุกสามารถทำลายโปรเซสบุกรุกได้ทุกโปรเซสรวมทั้งโปรเซสอื่นๆ ที่เกี่ยวข้อง เนื่องจากการตรวจจับการบุกรุกทำงานในระดับระบบปฏิบัติการซึ่งแตกต่างจากโปรแกรมตรวจจับการบุกรุกที่ทำงานในระดับของโปรเซส เมื่อระบบตรวจจับการบุกรุกทำงานในระดับของโปรเซส โปรเซสดังกล่าวย่อมมีโอกาสที่จะถูกเลือกที่จะเข้าใช้หน่วยประมวลผลกลางเช่นเดียวกันกับโปรเซสบุกรุกและโอกาสที่โปรเซสบุกรุกจะถูกเลือกให้ทำงานย่อมมีสูงกว่า แม้ว่าผู้ดูแลระบบจะปรับระดับความสำคัญ (priority) ของโปรเซสตรวจจับการบุกรุกอยู่ลำดับสูงสุด แต่โปรเซสบุกรุกยังมีโอกาสที่จะเข้าใช้หน่วยประมวลผลกลางเพราะมีจำนวนมากกว่า ดังนั้นถ้าหากเพิ่มกลไกการตรวจจับการบุกรุกให้แก่ระบบปฏิบัติการแล้วปัญหาที่กล่าวมาข้างต้นจะไม่เกิดขึ้น เนื่องจากการตรวจจับการบุกรุกเป็นส่วนหนึ่งของระบบปฏิบัติการอีกทั้งกระบวนการตรวจสอบจะทำงานก็ต่อเมื่อมีการเรียกใช้ซิซเท็มคอลที่เกี่ยวข้องกับความปลอดภัยซึ่งได้รับการแก้ไขไปแล้ว

6.3 การทดสอบเพื่อหาผลกระทบที่มีต่อชุดคำสั่งของระบบปฏิบัติการ

หลังจากที่เพิ่มกลไกการตรวจจับการบุกรุกให้แก่ซิซเท็มคอลแล้วย่อมก่อให้เกิดผลกระทบต่อชุดคำสั่งเดิมของระบบ การทดสอบระบบในส่วนนี้เป็นการหาผลกระทบต่างๆ ที่เกิดขึ้นและปรับแต่ง (tune up) กลไกการตรวจจับการบุกรุกให้สร้างผลกระทบแก่ระบบปฏิบัติการให้น้อยที่สุด โปรแกรมที่อาจจะได้รับผลกระทบคือโปรแกรมแบบ setuid เนื่องจากโปรแกรมเหล่านี้ทำงานในสถานะพิเศษ เพื่อให้การทดสอบสามารถครอบคลุมเหตุการณ์ได้ทุกเงื่อนไข จึง

จัดการทดสอบอีกประการหนึ่งคือการใช้งานระบบปฏิบัติการที่ได้รับการแก้ไขในระบบงานจริง การทดสอบส่วนนี้ได้แบ่งสถานการณ์ออกเป็นหลายรูปแบบ สำหรับรายละเอียดและผลการทดสอบนั้นจะกล่าวถึงตามลำดับดังนี้

6.3.1 การทดสอบหาผลกระทบต่อโปรแกรมแบบ setuid

โปรแกรมแบบ setuid อาจจะได้รับผลกระทบจากกลไกการตรวจจับการบุกรุก เนื่องจากโปรเซสของโปรแกรมเหล่านี้ทำงานในสถานะพิเศษ ดังนั้นจึงจำเป็นที่จะต้องทดสอบเพื่อหาผลกระทบที่อาจจะเกิดขึ้นแก่โปรแกรมเหล่านี้ทุกโปรแกรม โดยจัดรูปแบบและวิธีการทดสอบเช่นเดียวกับการทดสอบโปรแกรมตรวจจับการบุกรุกในหัวข้อที่ 4.6.3

จากการศึกษาถึงผลกระทบที่มีต่อโปรแกรมระบบสรุปได้ว่ากลไกการตรวจจับการบุกรุกชุดนี้สร้างผลกระทบต่อโปรแกรมของระบบบางตัวซึ่งแบ่งผลกระทบและโปรแกรมออกเป็นกลุ่มดังนี้

- **โปรแกรมที่ขัดต่อกฎสนับสนุนข้อที่ 0** ได้แก่โปรแกรม su เนื่องจากโปรแกรมดังกล่าวใช้สำหรับเปลี่ยนค่า user credential ของผู้ใช้ปรกติให้มีค่าเท่ากับค่า user credential ของ root เมื่อพิจารณาตามกฎสนับสนุนพบว่า โปรแกรม su เป็นโปรแกรมแบบ setuid และเรียกใช้ซิซเท็มคอล setgid() เพื่อเปลี่ยนค่า GID ของผู้ใช้เป็น 0 ผลกระทบดังกล่าวไม่สามารถแก้ไขในขณะนี้เนื่องจำเป็นต้องพัฒนาโปรแกรมนี้ใหม่ดังนั้นจึงต้องยกเลิกโปรแกรมนี้ชั่วคราว ถ้าหากผู้ดูแลระบบต้องการทำงานด้วยสิทธิพิเศษแล้วผู้ดูแลระบบจำเป็นต้องล๊อกอินเข้าสู่ระบบที่หน้า console ของระบบเท่านั้น

- **โปรแกรมที่เกี่ยวข้องกับการจัดการฐานข้อมูลผู้ใช้** เนื่องจากการจัดการฐานข้อมูลผู้ใช้ต้องดำเนินการผ่านคำสั่ง pwd_mkdb เท่านั้น โปรแกรมของระบบปฏิบัติการที่ดำเนินงานดังกล่าวได้แก่โปรแกรม chfn, chsh, chpass และ passwd เนื่องจากโปรแกรม passwd เป็นโปรแกรมพื้นฐานที่จะเป็นต่อผู้ใช้ จึงมีความจำเป็นที่จะแก้ไขกฎสนับสนุนเพิ่มเติม โดยอนุญาตให้โปรแกรมแบบ setuid เรียกใช้โปรแกรม pwd_mkdb โดยที่การยกเว้นดังกล่าวจะไม่ก่อให้เกิดผลกระทบต่อระบบเนื่องจาก โปรแกรมดังกล่าวเป็นโปรแกรมระบบที่เชื่อถือได้อีกทั้งได้รับการป้องกันการแก้ไขจากกฎสนับสนุนข้อที่ 3

สำหรับโปรแกรมแบบ setuid ที่จะสร้างขึ้นมาใหม่ ควรคำนึงถึงความปลอดภัย เนื่องจากจุดอ่อนที่เกิดจากโปรแกรมแบบ setuid มีโอกาสที่ผู้บุกรุกอาศัยช่องโหว่ดังกล่าวโจมตีระบบ Bishop [Bishop,1999] ได้เสนอแนวทางของการสร้างโปรแกรมแบบ setuid ให้ปลอดภัยเพื่อป้องกันปัญหาด้านความปลอดภัยที่เกิดจากจุดอ่อนที่เกิดขึ้นจากกระบวนการพัฒนาระบบ

6.3.2 การทดสอบหาผลกระทบจากการทำงานในสถานการณ์จริง

การทดสอบระบบในสถานการณ์จริงเป็นการทดสอบเพื่อหาผลกระทบที่เกิดขึ้นจากเหตุการณ์ที่ไม่สามารถทำนายล่วงหน้าได้เพื่อที่จะปรับปรุงการทำงานกลไกการตรวจจับการบุกรุก โดยจัดเครื่องแม่ข่าย (server) สำหรับบริการต่างๆ หลังจากนั้นบันทึกผลการใช้งาน

การใช้งานระบบปฏิบัติการยูนิกซ์โดยส่วนใหญ่แล้วจะใช้สำหรับจัดตั้งเป็นเครื่องแม่ข่าย เพื่อรองรับการเข้าสู่ระบบระยะไกล (remote login) ค้นหาเส้นทาง (PC router) หรือเป็นเครื่องไฟร์วอล (firewall) เป็นต้น วิทยานิพนธ์ชุดนี้จึงแบ่งการทดสอบออกเป็น 3 กลุ่มได้แก่

- **เครื่องให้บริการเมล์ เว็บและแฟ้ม** โดยเปิดบริการโพรโตคอล SMTP สำหรับการส่งเมล์ โพรโตคอล IMAP และ POP สำหรับการอ่าน E-mail และเปิดบริการเว็บซึ่งสนับสนุนการทำงานของสคริปต์ PHP ฐานข้อมูล MySQL และติดตั้งโปรแกรม vsftp ในช่วงภาคเรียนที่ 1 ปีการศึกษา 2548 ภายใต้ชื่อโดเมน lilac.cs.psu.ac.th
- **การให้บริการการเข้าสู่ระบบระยะไกล** โดยสร้างชื่อบัญชีสำหรับนักศึกษาที่ลงทะเบียนในรายวิชา structure programming จำนวน 96 คน เพื่อใช้ระบบปฏิบัติเน็ทบีเอสดีสำหรับการพัฒนาโปรแกรมด้วยภาษาซี และกลุ่มนักศึกษาซึ่งลงทะเบียนในรายวิชา computer network system นักศึกษากลุ่มนี้ได้พัฒนาโปรแกรมเครือข่าย (socket programming) ด้วยภาษาซีและสร้างเว็บไซต์ซึ่งสนับสนุนการทำงานของ php และฐานข้อมูล MySQL นักศึกษาทั้งสองกลุ่มติดต่อเข้าสู่ระบบผ่านโปรแกรม ssh และ telnet การทดสอบในหัวข้อนี้ได้ทดสอบในภาคการศึกษาที่ 1 ปีการศึกษา 2548
- **ติดตั้ง PC Router** สร้างเครือข่ายจำลองโดยติดตั้ง NAT และโปรแกรมไฟร์วอลให้แก่เครือข่ายในระหว่างวันที่ 14 ถึง 30 พฤศจิกายน 2548 โดยมีเครื่องคอมพิวเตอร์อยู่ภายใต้การดูแลจำนวน 4 เครื่อง ภายในช่วงเวลาดังกล่าวมีการใช้เครือข่ายอินเทอร์เน็ตตามปรกติ

จากการทดสอบระบบในแต่ละกลุ่มข้างต้นผลปรากฏว่า กิจกรรมต่างๆ สามารถดำเนินงานได้ตามปรกติและผู้ใช้สามารถใช้งานระบบได้โดยไม่ได้รับผลกระทบที่เกิดจากการแก้ไขระบบปฏิบัติการแต่อย่างใด เพราะฉะนั้นจึงสรุปได้ว่า เมื่อเพิ่มกลไกการตรวจจับการบุกรุกแก่ระบบปฏิบัติการ กลไกดังกล่าวไม่ก่อให้เกิดผลกระทบแก่ระบบงานเดิมยกเว้นคำสั่ง su

6.4 การทดสอบหาประสิทธิภาพของระบบปฏิบัติการหลังจากที่แก้ไขซิซเท็มคอล

การทดสอบระบบในลำดับถัดไปเป็นการเปรียบเทียบประสิทธิภาพของระบบปฏิบัติการโดยวัดเวลาในการทำงานของซิซเท็มคอลที่ได้รับการแก้ไขทั้งก่อนและหลังการแก้ไขระบบปฏิบัติการ หลังจากนั้นนำเวลาที่ได้มาวิเคราะห์และเปรียบเทียบในเชิงสถิติ ซึ่งลำดับถัดไปจะกล่าวถึงการทดสอบระบบ สรุปและวิเคราะห์ผลการทดสอบระบบ

6.4.1 กรณีทดสอบ

การทดสอบชุดนี้เลือกทดสอบบางซิซเท็มคอลโดยจัดกลุ่มซิซเท็มคอลตามวิธีการพัฒนาโปรแกรม ซึ่งจัดกลุ่มซิซเท็มคอลออกได้ 4 กลุ่มได้แก่

กลุ่มของซิซเท็มคอล setuid()

ซิซเท็มคอลในกลุ่มนี้ประกอบไปด้วยซิซเท็มคอล setuid(), setgid(), seteuid(), setegid() และซิซเท็มคอลในกลุ่มซิซเท็มคอลวิกฤต เช่น settimeofday() เป็นต้น เนื่องจากรูปแบบของการพัฒนาระบบเหมือนกันคือ แก้ไขเพิ่มเติมเฉพาะการนิยามสถานะของโปรเซสซึ่งไม่มีการตรวจสอบค่าพารามิเตอร์เหมือนกับซิซเท็มคอลกลุ่มอื่นๆ

กลุ่มของซิซเท็มคอล fchmod()

การวัดเวลาการทำงานซิซเท็มคอลที่ดำเนินงานเกี่ยวกับแฟ้ม ค่าที่ได้จากการวัดแต่ละครั้งมีความแตกต่างกันมากเนื่องจากเวลาที่ได้จากทดสอบเป็นเวลาที่ใช้ในการทำงานของซิซเท็มคอลรวมกับเวลาที่ใช้ในการเข้าถึงแฟ้ม ถ้าหากเวลาที่ใช้ในการเข้าถึงแฟ้มเปลี่ยนแปลงไป 1 มิลลิวินาที จะเป็นผลให้เวลาการทำงานรวมของซิซเท็มคอลเปลี่ยนแปลงไปเกือบ 100% ดังนั้นเนื่องจากเวลาการทำงานของซิซเท็มคอล chmod() นั้นรวมเวลาของการเปิดแฟ้มและเวลาการทำงานของซิซเท็มคอลเข้าด้วยกัน แต่เวลาการทำงานของซิซเท็มคอล fchmod() เป็นเวลาการทำงานของซิซเท็มคอลซึ่งไม่รวมเวลาของการเปิดแฟ้ม อีกทั้งการแก้ไขซิซเท็มคอลในกลุ่มนี้แก้ไขที่จุดเดียวกันคือภายในฟังก์ชัน change_mode() ดังนั้นการวัดเวลาในกลุ่มนี้จึงวัดเวลาการทำงานของซิซเท็มคอล fchmod()

กลุ่มของซิซเท็มคอล execve()

ซิซเท็มคอล execve() ใช้เวลาทำงานมากกว่าซิซเท็มคอลอื่นๆ เนื่องจากซิซเท็มคอลนี้มีหน้าที่สั่งงานโปรเซส การวัดเวลาการทำงานของซิซเท็มคอลนี้จะเริ่มจับเวลาเมื่อเริ่มสั่งงานซิซเท็มคอลจนกระทั้งซิซเท็มคอล execve() ส่งค่ากลับ เวลาได้จากการทดสอบคือเวลาการทำงานของซิซเท็มคอลและเวลาการทำงานของโปรเซสที่ถูกสั่งงานรวมเข้าด้วยกัน อีกทั้งการสั่งงาน

โปรแกรมต้องใช้เวลาสำหรับการค้นหาแฟ้มในดิสก์ เพื่อเป็นการเลี่ยงปัญหาข้างต้นการทดสอบจึงเลือกใช้คำสั่งภายใน (internal command) ของโปรแกรมเชลล์ซึ่งได้แก่คำสั่ง pwd

กลุ่มของซิซเท็มคอล open()

การวัดเวลาการทำงานของซิซเท็มคอล open() แตกต่างจากการทดสอบซิซเท็มคอลอื่นๆ นั่นคือกำหนดให้โปรแกรมทดสอบได้เพื่อเปิดแฟ้มจำนวน 5000 รอบก่อนที่จะเริ่มวัดเวลาการทำงานหลังจากนั้นเริ่มวัดเวลาการทำงานตามปกติ จนกระทั่งได้ข้อมูลครบตามจำนวนแล้วจึงบันทึกผลการทดสอบ การเปิดแฟ้มก่อนที่จะวัดเวลาการทำงานจริงเป็นการความคลาดเคลื่อนของเวลาที่เกิดจากหน่วยความจำแคชของอุปกรณ์

6.4.2 วิธีการวัดเวลาการทำงานของซิซเท็มคอล

ข้อจำกัดของการวัดเวลาการทำงานของซิซเท็มคอลคือ ซิซเท็มคอลแต่ละตัวใช้เวลาในการทำงานในระดับของไมโครวินาทีซึ่งไม่สะดวกแก่การวัดเวลาการทำงาน ดังนั้นจำเป็นต้องเรียกใช้ซิทเท็มคอลนั้นหลายๆ รอบเพื่อวัดเวลาการทำงานรวมแล้วนำมาหาค่าเฉลี่ย การทดสอบหัวข้อนี้แบ่งข้อมูลออกเป็น 10 กลุ่มตามจำนวนรอบของการทดสอบ ได้แก่การเรียกใช้ซิซเท็มคอลจำนวน 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000 และ 50000 รอบ จากการวัดเวลาแต่ละรอบจะได้ข้อมูลมาหนึ่งชุดซึ่งเป็นเวลาที่ใช้ในการทำงานของซิซเท็มคอล สำหรับแต่ละจำนวนรอบของการทำงานได้ทดสอบซ้ำจำนวน 5000 ครั้ง แต่ซิซเท็มคอล execve() ใช้เวลาในการทำงานมากกว่าซิซเท็มคอลตัวอื่น ดังนั้นจึงจำเป็นที่จะลดจำนวนรอบของการทดสอบเป็น 5 10 15 20 25 30 35 40 45 และ 50 รอบ เพื่อช่วยลดเวลาในการทดสอบระบบ

ค่าเฉลี่ยของเวลาการทำงานคำนวณโดยนำข้อมูลดิบทั้ง 5000 ชุดตัดค่าสุดโต่ง (outlier) หลังจากนั้นนำข้อมูลที่เหลือมาหาค่าเฉลี่ย ผลที่ได้จากการเฉลี่ยคือ เวลาการทำงานของซิซเท็มคอ

6.4.3 ผลการวัดเวลาการทำงานของซิซเท็มคอลแต่ละกลุ่ม

เมื่อวัดเวลาการทำงานของซิซเท็มคอลตามวิธีการข้างต้นแล้วนำผลของการทดสอบที่ได้หาค่าเฉลี่ยพร้อมทั้งสร้างกราฟแสดงความสัมพันธ์ระหว่างจำนวนรอบของการทดสอบและเวลาของการทำงาน ผลการทดลองแสดงไว้ในตารางที่ 6.6 และแสดงผลการเปรียบเทียบในรูปแบบของกราฟดังภาพประกอบที่ 6.3, 6.4, 6.5 และ 6.6

ข้อมูลแต่ละตารางแสดงเวลาการทำงานของซิซเท็มคอลในหน่วยของมิลลิวินาที (ms) ซึ่งเป็นเวลาที่ใช้ในการทำงานทั้งก่อนและหลังการแก้ไข หลังจากนั้นคำนวณหาผลต่างของเวลาแล้วแสดงข้อมูลในรูปแบบของร้อยละ หลังการแก้ไขซิซเท็มคอลประสิทธิภาพของระบบปฏิบัติการไม่ควรลดลงมากกว่าร้อยละ 5

สำหรับกราฟแสดงความสัมพันธ์ของจำนวนรอบการทำงานและเวลาที่ใช้ของซิซเท็มคอลเป็นการสร้างกราฟจากข้อมูลในตาราง หลังจากนั้นหาเส้นแนวโน้มของข้อมูลจากกราฟ ซึ่งพบว่า กราฟที่ได้จากการทดลองนั้นมีแนวโน้มการเพิ่มขึ้นเป็นเส้นตรงทั้งหมด

ตัวอย่างกราฟในภาพประกอบที่ 6.3 สมการพยากรณ์ 2 ชุด และค่า R^2 เป็น 1.000 ซึ่งหมายความว่า จากการประมาณการเติบโตของข้อมูลสองชุดเมื่อพิจารณาจากสมการแล้วข้อมูลทั้งสองชุดมีอัตราการเติบโตที่ใกล้เคียงกัน โดยข้อมูลชุดหลังการแก้ไขมีอัตราการเพิ่มสูงกว่าเล็กน้อย เมื่อพิจารณาจากค่า R^2 แล้ว ข้อมูลทั้งสองชุดแทบจะไม่มีความคลาดเคลื่อนจากสมการพยากรณ์เนื่องจากค่า R^2 มีค่าเป็น 1.000

ตารางที่ 6.6 เวลาที่ใช้ในการทำงานของซิซเท็มคอลก่อนและหลังการแก้ไข

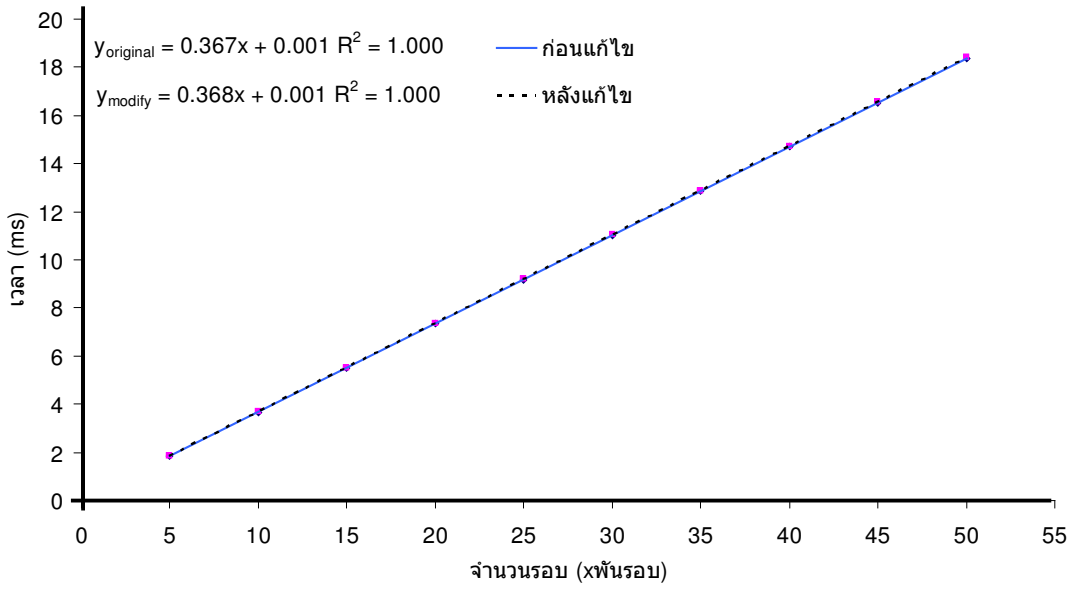
(พัน) รอบ	เวลาการทำงานของ setuid() (ms)		
	ก่อน	หลัง	%
5	1.8375	1.8408	0.183
10	3.6741	3.6808	0.182
15	5.5108	5.5208	0.182
20	7.3484	7.3618	0.181
25	9.1844	9.2013	0.184
30	11.0207	11.0408	0.182
35	12.8573	12.8807	0.182
40	14.6939	14.7207	0.182
45	16.5305	16.5606	0.182
50	18.3678	18.4012	0.182

(พัน) รอบ	เวลาการทำงานของ fchmod() (ms)		
	ก่อน	หลัง	%
5	10.447	10.486	0.375
10	20.881	20.970	0.425
15	31.317	31.455	0.436
20	41.750	41.939	0.450
25	52.177	52.423	0.470
30	62.619	62.907	0.458
35	73.040	73.391	0.478
40	83.484	83.877	0.468
45	93.913	94.364	0.478
50	104.335	104.848	0.489

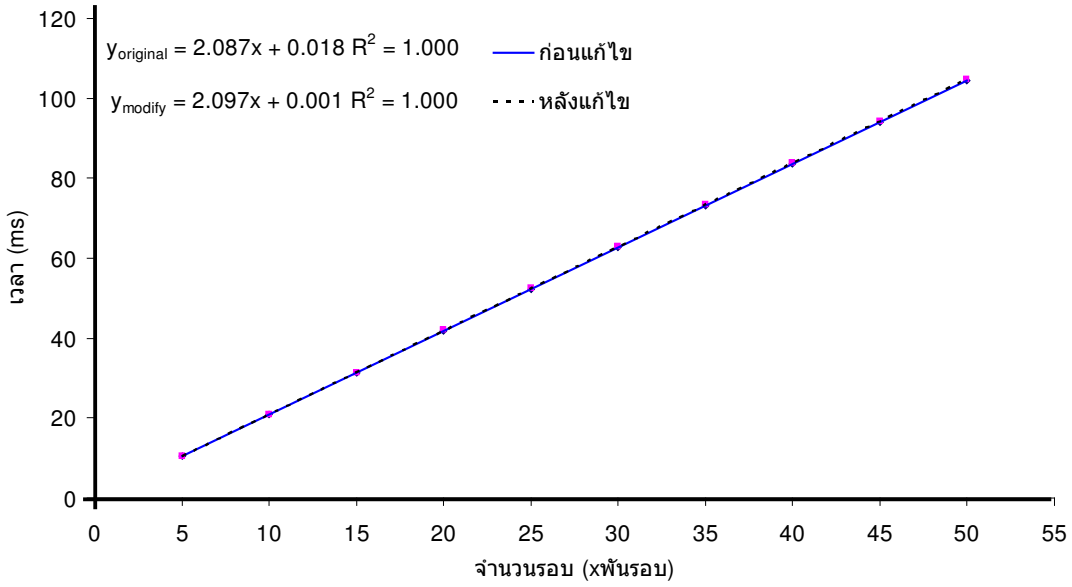
ตารางที่ 6.6 (ต่อ) เวลาที่ใช้ในการทำงานของซิซเท็มคอลก่อนและหลังการแก้ไข

(พัน)	เวลาการทำงานของ open() (ms)		
รอบ	ก่อน	หลัง	%
5	10.483	10.825	3.159
10	20.964	21.657	3.199
15	31.446	32.490	3.212
20	41.927	43.323	3.222
25	52.409	54.141	3.199
30	62.891	64.974	3.207
35	73.372	75.811	3.216
40	83.853	86.643	3.220
45	94.336	97.474	3.219
50	104.816	108.316	3.231

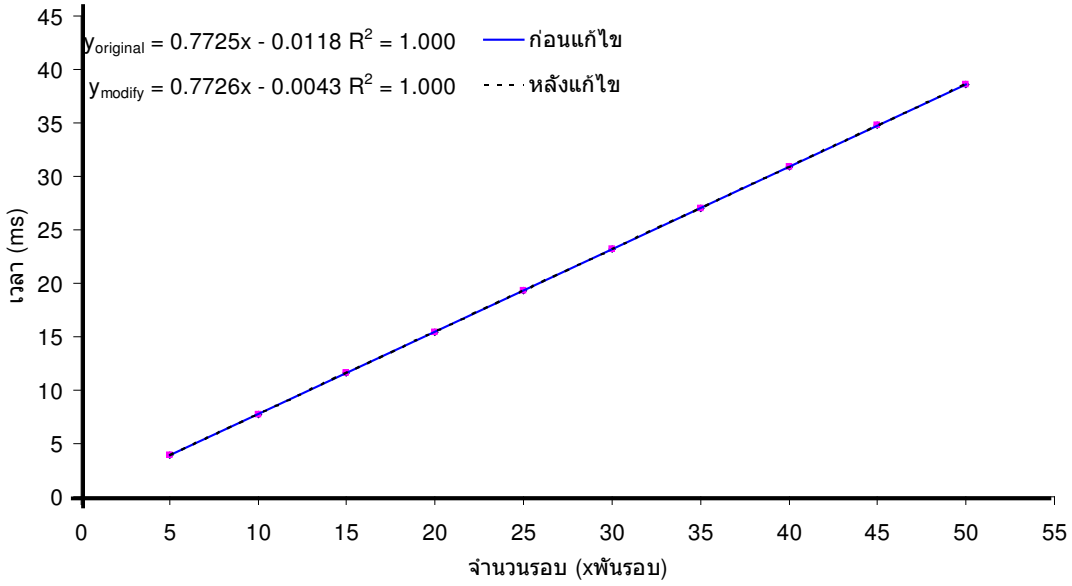
	เวลาการทำงานของ execve() (ms)		
รอบ	ก่อน	หลัง	%
5	3.8570	3.8605	0.093
10	7.7165	7.7210	0.058
15	11.5696	11.5814	0.102
20	15.4375	15.4460	0.055
25	19.2972	19.3217	0.127
30	23.1600	23.1781	0.078
35	27.0299	27.0312	0.005
40	30.8782	30.8973	0.062
45	34.7557	34.7642	0.024
50	38.6191	38.6310	0.031



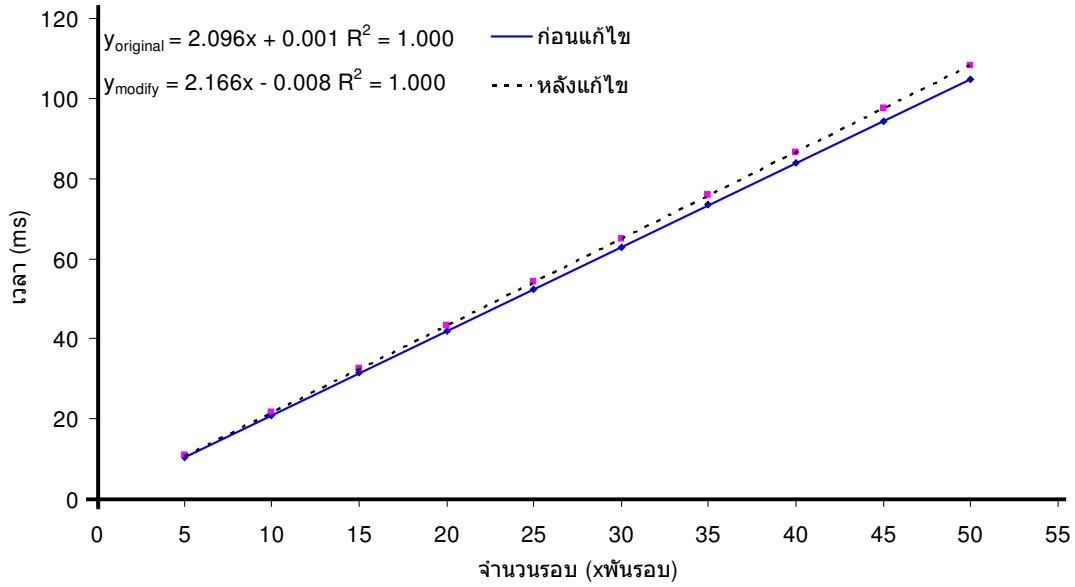
ภาพประกอบ 6.5 เวลาที่ใช้ในการทำงานของซิซเท็มคอล setuid() ก่อนและหลังการแก้ไข



ภาพประกอบ 6.6 เวลาที่ใช้ในการทำงานของซิซเท็มคอล fchmod() ก่อนและหลังการแก้ไข



ตารางที่ 6.7 เวลาที่ใช้ในการทำงานของซิซเท็มคอล execve() ก่อนและหลังการแก้ไข



ภาพประกอบ 6.8 เวลาที่ใช้ในการทำงานของซิซเท็มคอล open() ก่อนและหลังการแก้ไข

เมื่อเปรียบเทียบเวลาการทำงานของซิซเท็มคอล setuid() ทั้งก่อนและหลังการแก้ไขพบว่า เวลาที่ใช้ในการทำงานของซิซเท็มคอลมีความแตกต่างกันน้อยมาก นั่นคือร้อยละ 0.18 (น้อยกว่าร้อยละ 5) เมื่อพิจารณาจากค่าความชันของกราฟซึ่งแสดงอัตราการเติบโตของเวลาหลังการแก้ไขจะสูงกว่าก่อนแก้ไขเพียง 0.001 ดังนั้นจึงสรุปได้ว่าการแก้ไขซิซเท็มคอล setuid() ทำให้ประสิทธิภาพของระบบปฏิบัติการเปลี่ยนแปลงไปและยอมรับได้

เวลาการทำงานของซิซเท็มคอล fchmod() ทั้งก่อนและหลังมีความใกล้เคียงกันมากเช่นเดียวกับซิซเท็มคอล setuid() เมื่อพิจารณาจากภาพประกอบที่ 6.4 พบว่า กราฟทั้งสองเส้นมีอัตราการเพิ่มขึ้นของเวลาใกล้เคียงกัน เวลาการทำงานหลังการแก้ไขมากกว่าเวลาก่อนการแก้ไขร้อยละ 0.41 (น้อยกว่าร้อยละ 5) ถือว่ายอมรับได้ เช่นเดียวกันกับการทดสอบซิซเท็มคอล exevce() เวลาที่ใช้ในการทำงานหลังการแก้ไขเพิ่มขึ้นจากเดิมแต่ยังน้อยกว่าร้อยละ 5 ซึ่งถือว่ายอมรับได้เช่นกัน ดังนั้นจึงสรุปได้ว่าการแก้ไขซิซเท็มคอล fchmod() และ execve() ทำให้ประสิทธิภาพของระบบปฏิบัติการเปลี่ยนแปลงไปและยอมรับได้

แต่ผลการทดสอบของซิซเท็มคอล open() มีความแตกต่างกันมากกว่าซิซเท็มคอลอื่นๆ เนื่องจากการทดสอบส่วนนี้ได้รับผลกระทบมาจากปัจจัยภายนอกที่ไม่สามารถป้องกันได้ แต่เมื่อพิจารณาถึงค่าที่แตกต่างนั้นจะเห็นได้ว่า มีแนวโน้มของการเพิ่มขึ้นของเวลาในทิศทางเดียวกัน

อีกทั้งผลต่างของเวลามีความแตกต่างมากกว่าซิซเท็มคอลตัวอื่นแต่ยังอยู่ในระดับที่ยอมรับได้เนื่องจากให้เวลาการทำงานเพิ่มขึ้นร้อยละเปลี่ยนแปลงไปไม่ถึง 5%

6.5 บทสรุป

การทดสอบความแม่นยำของการทำงานของฟังก์ชันตรวจจับการบุกรุกสรุปได้ว่าระบบชุดนี้สามารถตรวจจับการบุกรุกได้ทุกวิธีการที่ขัดต่อกฎสนับสนุน และสามารถทำงานได้ในสภาวะเครียด สรุปได้ว่า ระบบตรวจจับการบุกรุกสามารถทำงานได้ทั้งสภาวะปกติและสภาวะเครียด การทดสอบข้างต้นสามารถพิสูจน์ได้ว่า ระบบดังกล่าวใช้ทรัพยากรสำหรับการดำเนินงานน้อยมาก อีกทั้งสามารถจัดการโปรเซสบุกรุกจำนวนมากได้ในเวลาเดียวกัน

ระบบตรวจจับการบุกรุกชุดนี้สร้างผลกระทบให้แก่คำสั่งแบบ setuid ได้แก่คำสั่ง su ผู้ดูแลระบบจำเป็นที่จะต้องยกเลิกคำสั่งนี้ชั่วคราวจนกว่าจะได้รับการพัฒนาโปรแกรมชุดนี้ใหม่ คำสั่งอีกกลุ่มที่ได้รับผลกระทบคือ กลุ่มของคำสั่งที่เกี่ยวข้องกับการแก้ไขฐานข้อมูลผู้ใช้ แต่ผลกระทบในข้อนี้สามารถแก้ไขได้โดยยกเว้นการตรวจสอบการเรียกใช้โปรแกรม pwd_mkdb ส่วนผลกระทบที่อาจจะเกิดขึ้นจากการใช้งานปกตินั้นได้ทดสอบแล้วว่าไม่ก่อให้เกิดผลกระทบใดๆ กับระบบงานเดิม

เมื่อพิจารณาถึงประสิทธิภาพของระบบปฏิบัติการที่เปลี่ยนแปลงไปสรุปได้ว่าเวลาที่เพิ่มขึ้นในแต่ละซิซเท็มคอลที่ได้รับการแก้ไขนั้นยังอยู่ในระดับที่ยอมรับได้ สำหรับซิซเท็มคอล open() นั้นเมื่อพิจารณาในเชิงสถิตินั้นเวลาการทำงานแตกต่างกันประมาณ 3% ยังถือว่ายอมรับได้