บทที่ 4

การพัฒนาระบบ

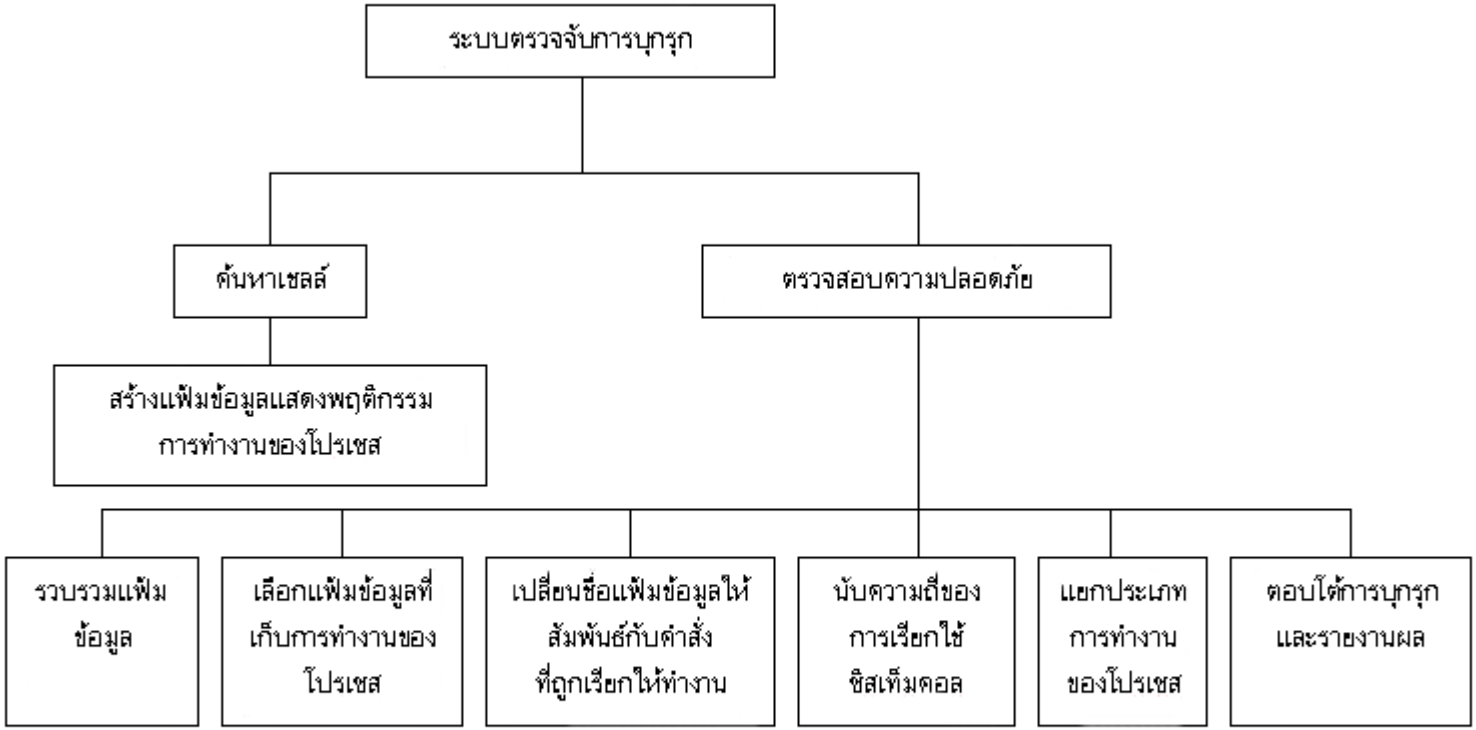
ในบทนี้จะกล่าวถึงการพัฒนาระบบตรวจจับการบุกรุกซึ่งมีเนื้อหาในบทประกอบด้วยโครงสร้างการดำเนินงาน ขั้นตอนวิธี การพัฒนาโปรแกรม การพัฒนาข้อมูลฝึกสอนระบบเพื่อใช้งาน แฟ้มข้อมูลหลักที่ระบบตรวจจับการบุกรุกใช้ในระหว่างการทำงาน และสารบบแฟ้มข้อมูล ตามลำดับ

ระบบตรวจจับการบุกรุกที่พัฒนาขึ้นนี้เป็นต้นแบบของระบบตรวจจับการบุกรุกที่เกิดจากการนำวิธีการในการตรวจจับการบุกรุกมาผสมกันระหว่างวิธีการแบบ anomaly detection และ misuse detection โดยที่ระบบที่พัฒนาขึ้นนี้เป็นโปรแกรมการทำงานอย่างหนึ่งซึ่งเมื่อถูกเรียกใช้งานภายใต้ระบบปฏิบัติการลินุกซ์เรดแฮ็ทจะถูกเปลี่ยนไปเป็นโปรเซสหนึ่งของระบบปฏิบัติการแต่เป็นโปรเซสที่ขึ้นอยู่กับการทำงานของระบบปฏิบัติการโดยตรงเนื่องจากเป็นโปรเซสแบบดีมอน (daemon process) การทำงานของระบบตรวจจับการบุกรุกจะไม่มีส่วนติดต่อกับผู้ใช้งานภายในระบบ

4.1 โครงสร้างการดำเนินงาน

จากแนวคิดในการออกแบบระบบตรวจจับการบุกรุกในบทที่ 3 ทำให้เกิดระบบตรวจจับการบุกรุกที่มีส่วนการทำงานต่อเนื่องกันกลายเป็นระบบใหญ่ระบบหนึ่งที่ทำงานโดยไม่มีการหยุด กรณีที่จะทำให้ระบบตรวจจับการบุกรุกหยุดการทำงานตามแนวคิดที่ออกแบบไว้คือเมื่อระบบปฏิบัติการหยุดทำงาน แต่ในกรณีที่ต้องการให้ระบบตรวจจับการบุกรุกหยุดทำงานโดยตรงผู้จัดทำได้สร้างโปรแกรมขึ้นมาอีกชุดหนึ่งเพื่อใช้หยุดการทำงานของระบบตรวจจับการบุกรุกซึ่งจะแสดงรายละเอียดไว้ในส่วนของภาคผนวก

ระบบตรวจจับการบุกรุกประกอบด้วยโปรเซสหลัก 2 โปรเซสคือ โปรเซสค้นหาเชลล์ในระบบปฏิบัติการเพื่อติดตามการทำงานของเชลล์ และโปรเซสที่ทำงานตรวจสอบความปลอดภัยให้กับระบบปฏิบัติการจากการทำงานของโปรเซสซึ่งสามารถแสดงการทำงานเป็นระดับที่มีความสัมพันธ์กันดังภาพประกอบ 4.1



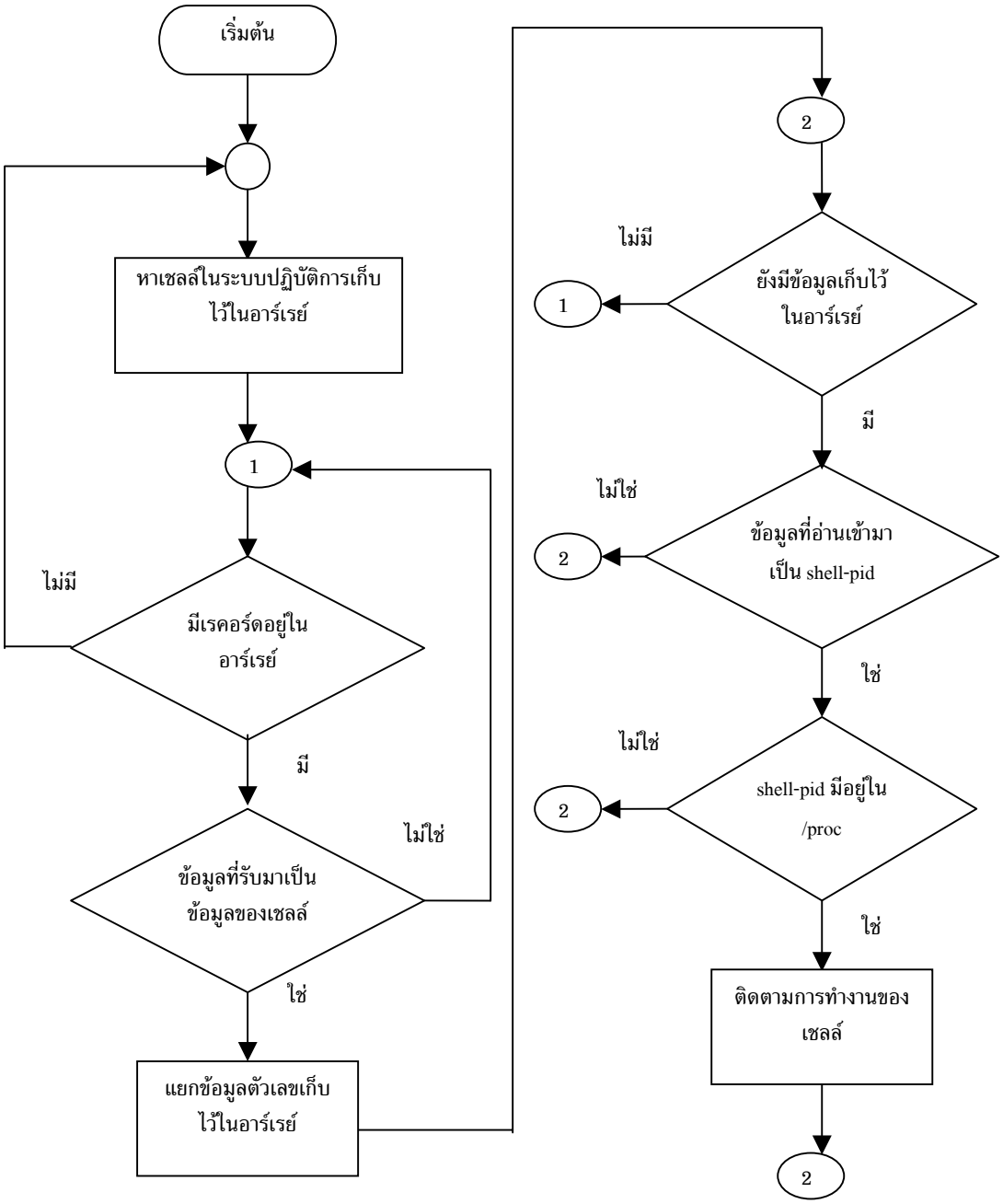
ภาพประกอบ 4.1 โครงสร้างระบบตรวจจับการบุกรุก

4.2 ขั้นตอนวิธี

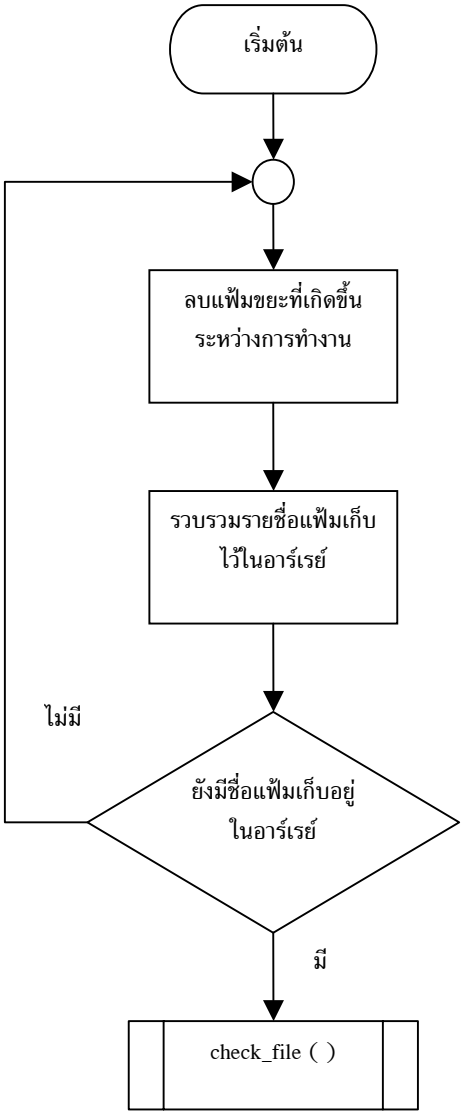
ระบบตรวจจับการบุกรุกที่พัฒนาขึ้นมีขั้นตอนการทำงานของกระบวนการต่าง ๆ ซึ่งสามารถอธิบายโดยเขียนให้อยู่ในลักษณะของ flow chart สำหรับขั้นตอนวิธีที่สำคัญมีดังนี้

การค้นหาเซลล์ที่เกิดขึ้นในระบบปฏิบัติการ ส่วนหนึ่งของระบบตรวจจับการบุกรุกที่เป็นการค้นหาเซลล์ที่เกิดขึ้นในระบบปฏิบัติการเพื่อติดตามการสร้างโปรเซสของเซลล์มีขั้นตอนวิธีสามารถแสดงได้ดังภาพประกอบ 4.2

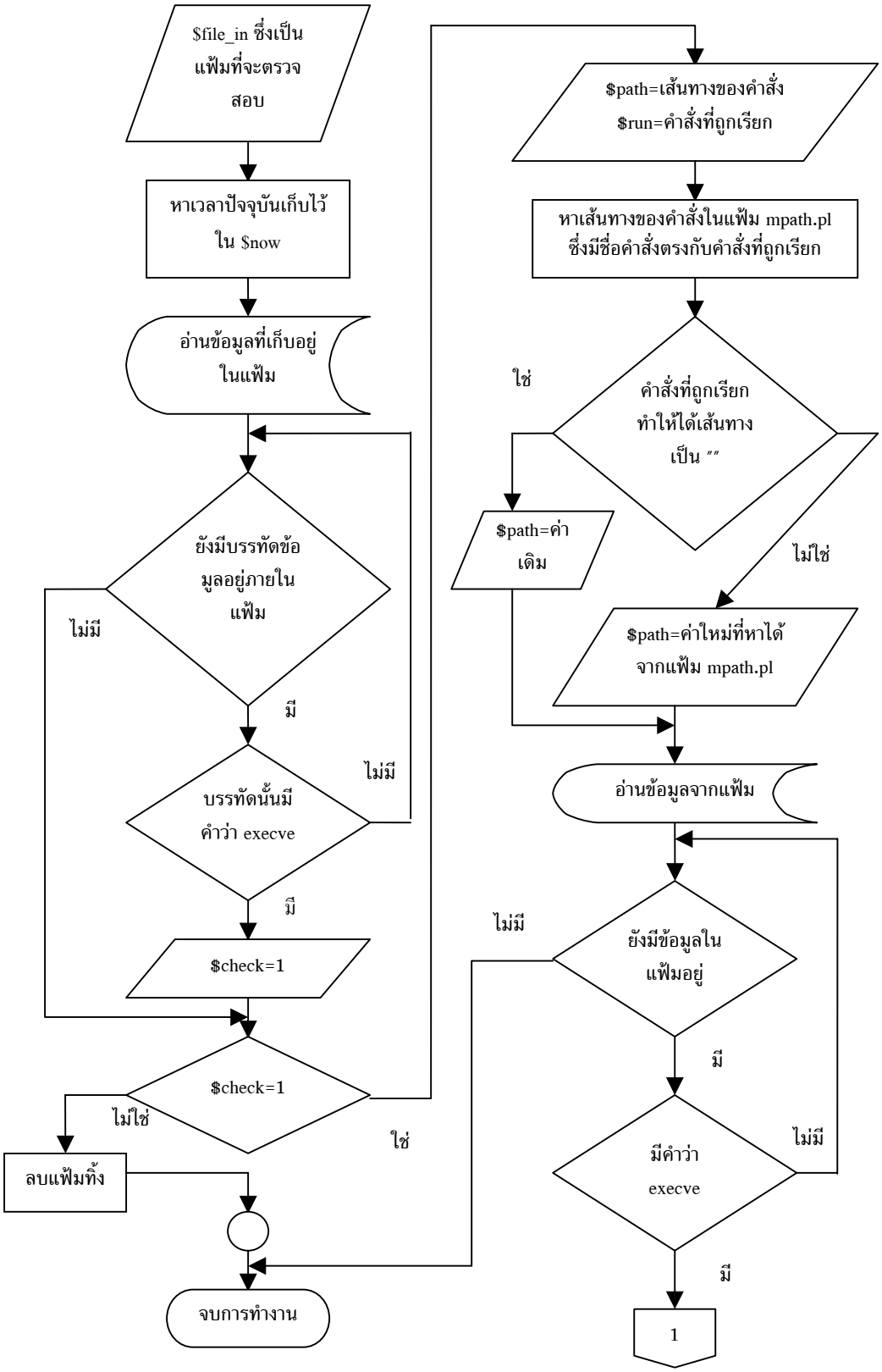
การตรวจสอบความปลอดภัยให้กับระบบปฏิบัติการ ในการตรวจสอบความปลอดภัยจากการทำงานของโปรเซส การทำงานโดยรวมจะเป็นการรวบรวมพฤติกรรมการทำงานของโปรเซสและแยกประเภทคลาสการทำงานของโปรเซส หากเป็นคลาสการทำงานที่ผิดปกติจะดำเนินการเก็บหลักฐานการบุกรุกระบบและตัดผู้บุกรุกออกจากการเชื่อมต่อกับระบบปฏิบัติการ ในส่วนนี้ได้กำหนดกระบวนการทำงานให้เป็นฟังก์ชันที่ดำเนินการร่วมกันมีขั้นตอนวิธีของฟังก์ชันดังภาพประกอบ 4.3 ถึงภาพประกอบ 4.9 ตามลำดับ



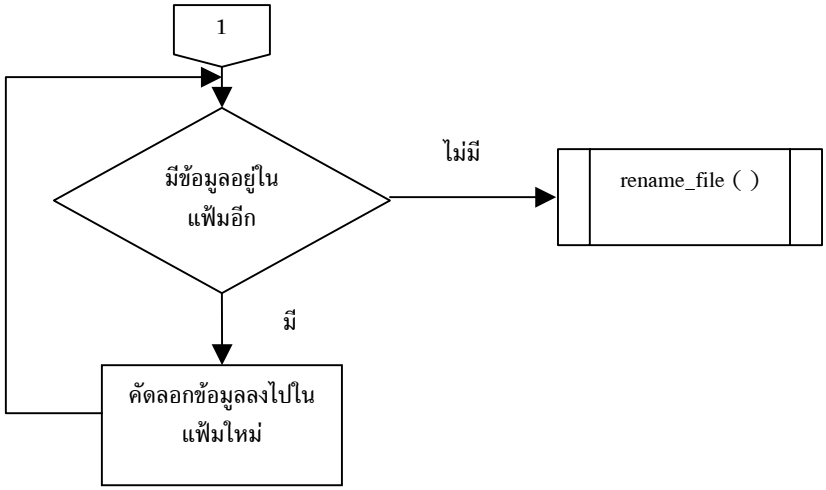
ภาพประกอบ 4.2 การทำงานของโปรแกรม find_shell



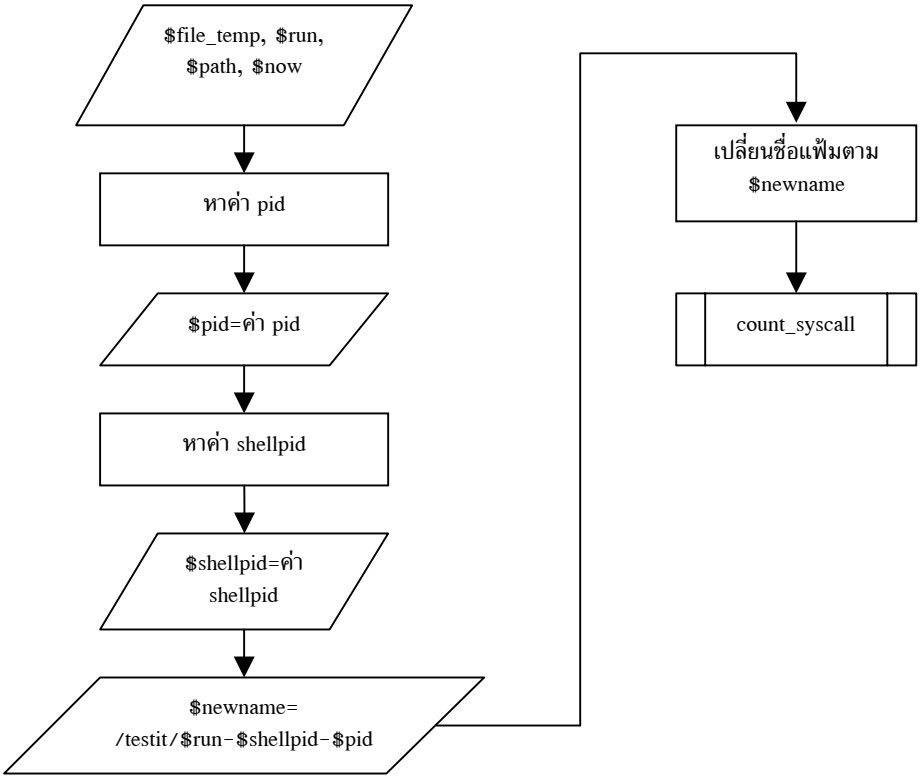
ภาพประกอบ 4.3 แสดงส่วนเริ่มต้นการทำงานของโปรแกรม monitor.pl



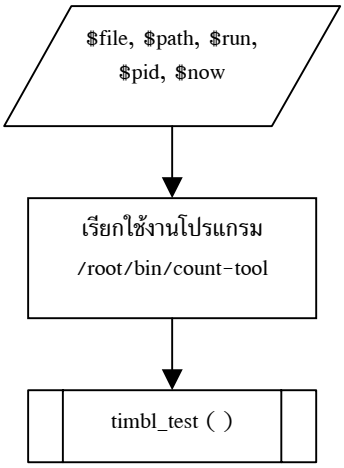
ภาพประกอบ 4.4 แสดงฟังก์ชัน check_file ของโปรแกรม monitor.pl



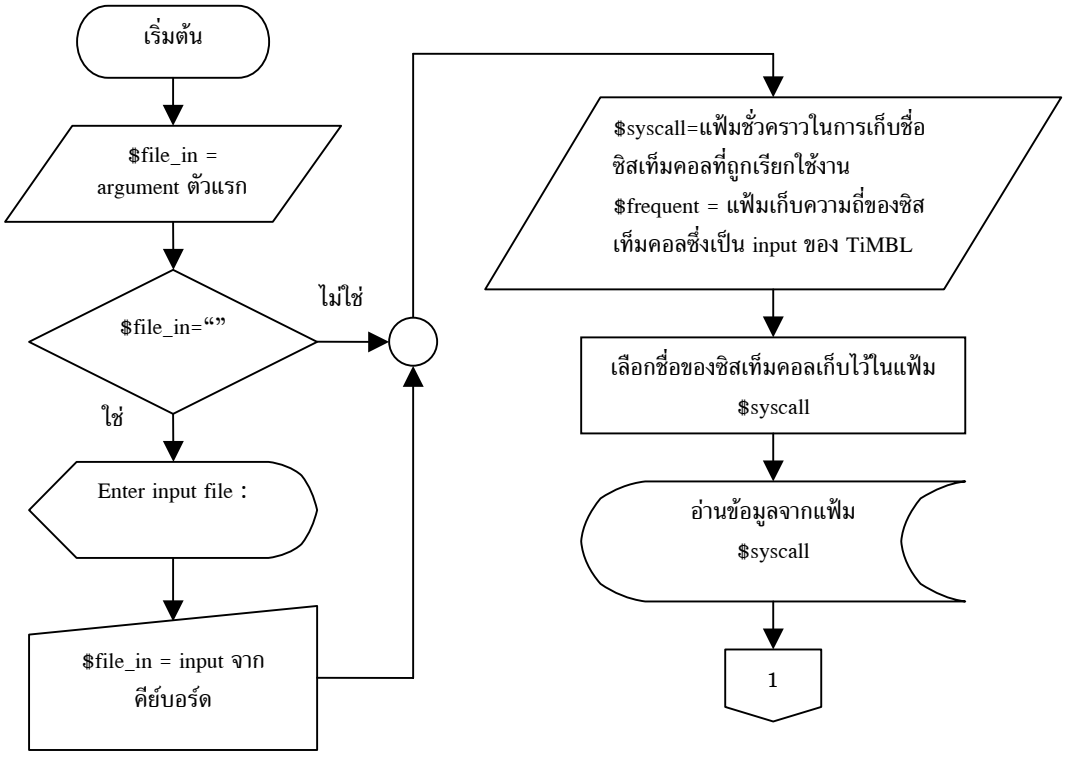
ภาพประกอบ 4.4 แสดงฟังก์ชัน check_file ของโปรแกรม monitor.pl (ต่อ)



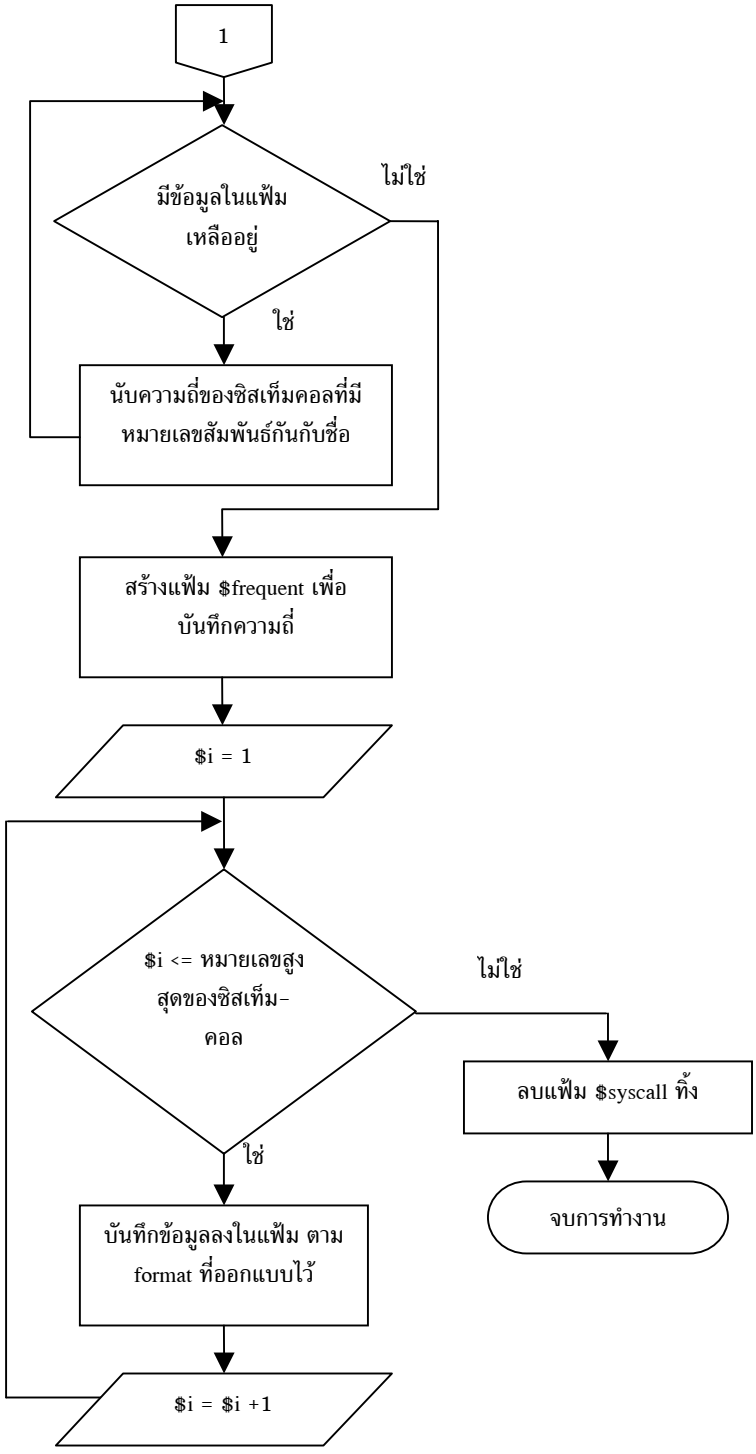
ภาพประกอบ 4.5 แสดงฟังก์ชัน rename_file ของโปรแกรม monitor.pl



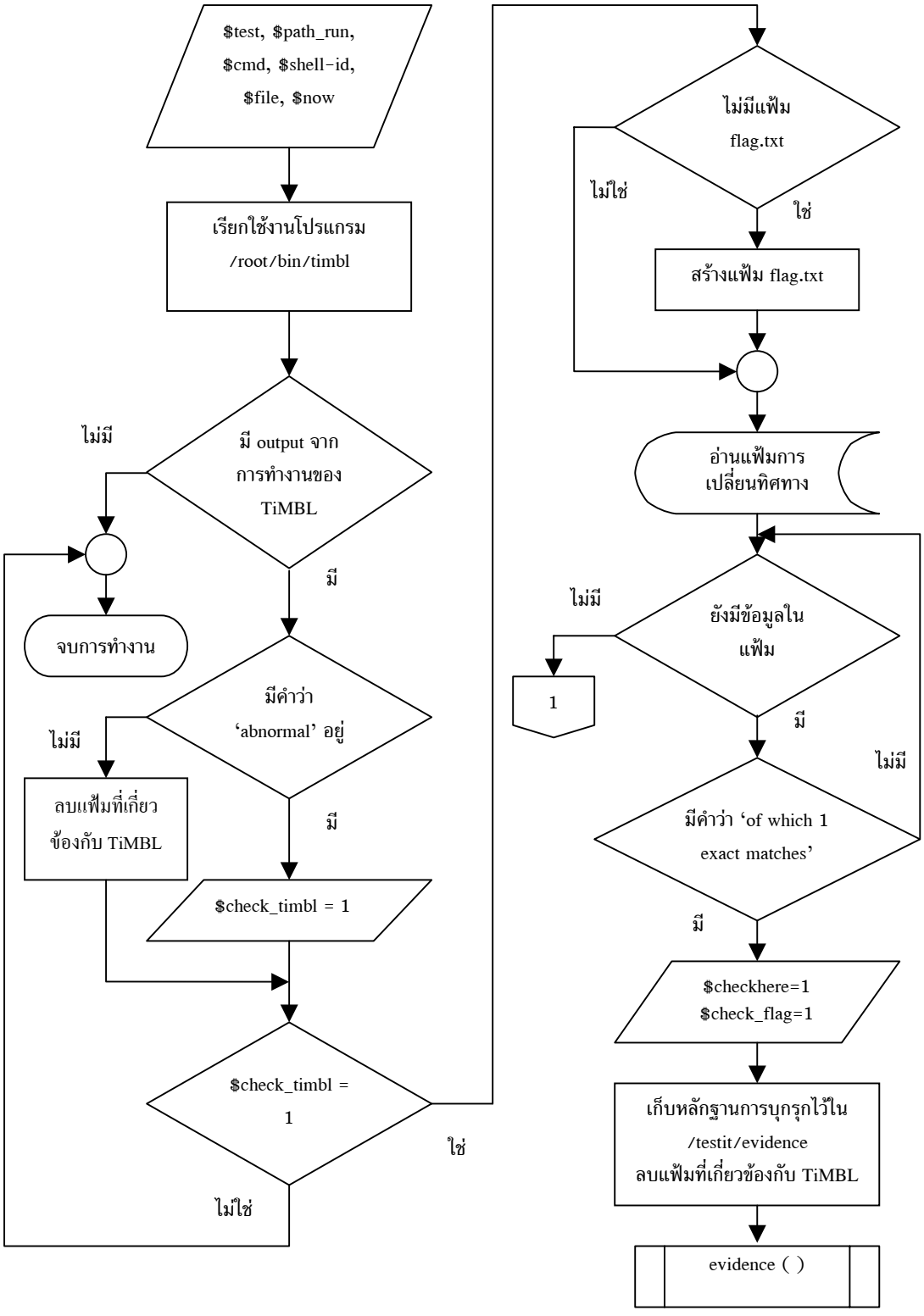
ภาพประกอบ 4.6 แสดงฟังก์ชัน count_syscall ของโปรแกรม monitor.pl



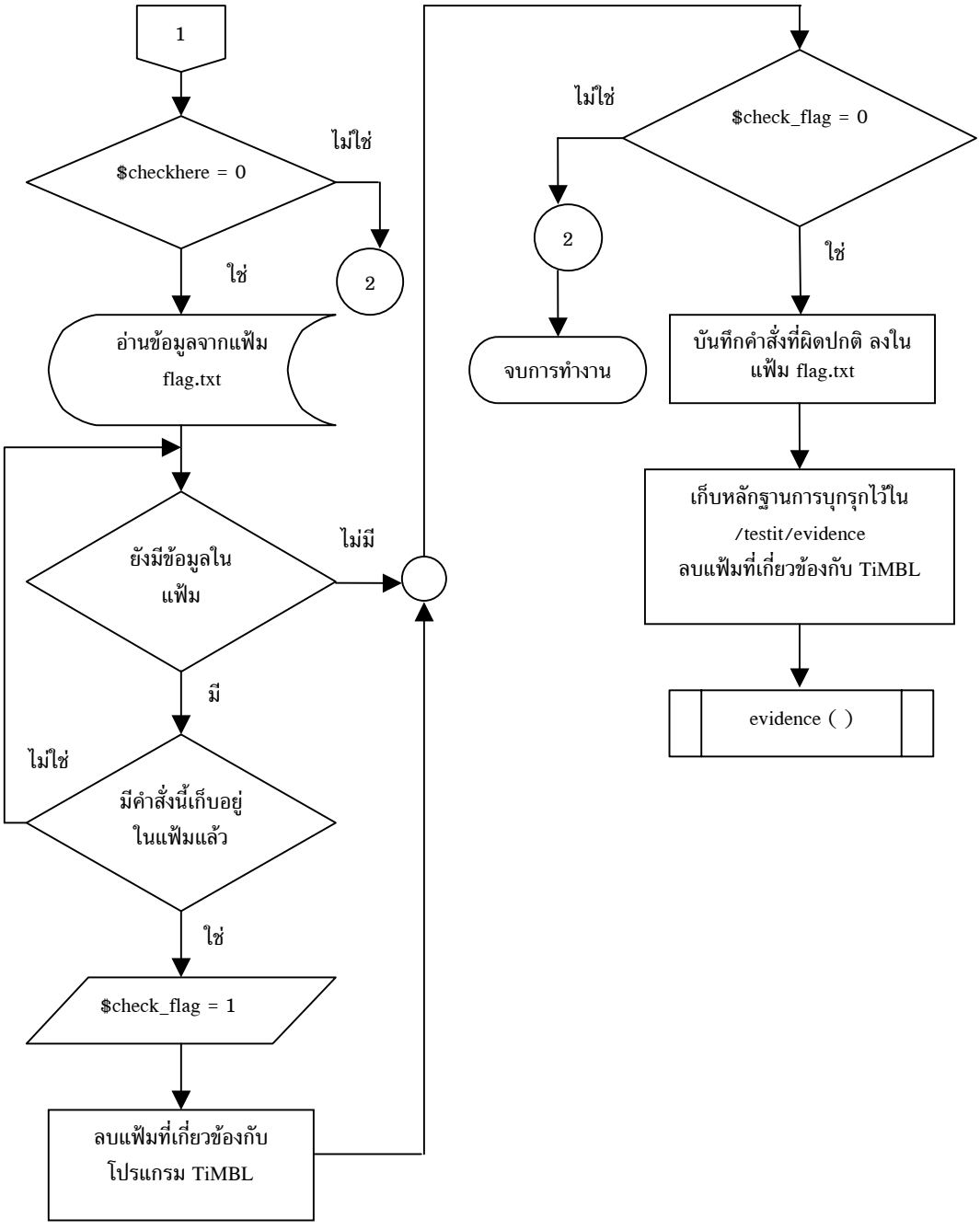
ภาพประกอบ 4.7 แสดงการทำงานของโปรแกรม count-tool



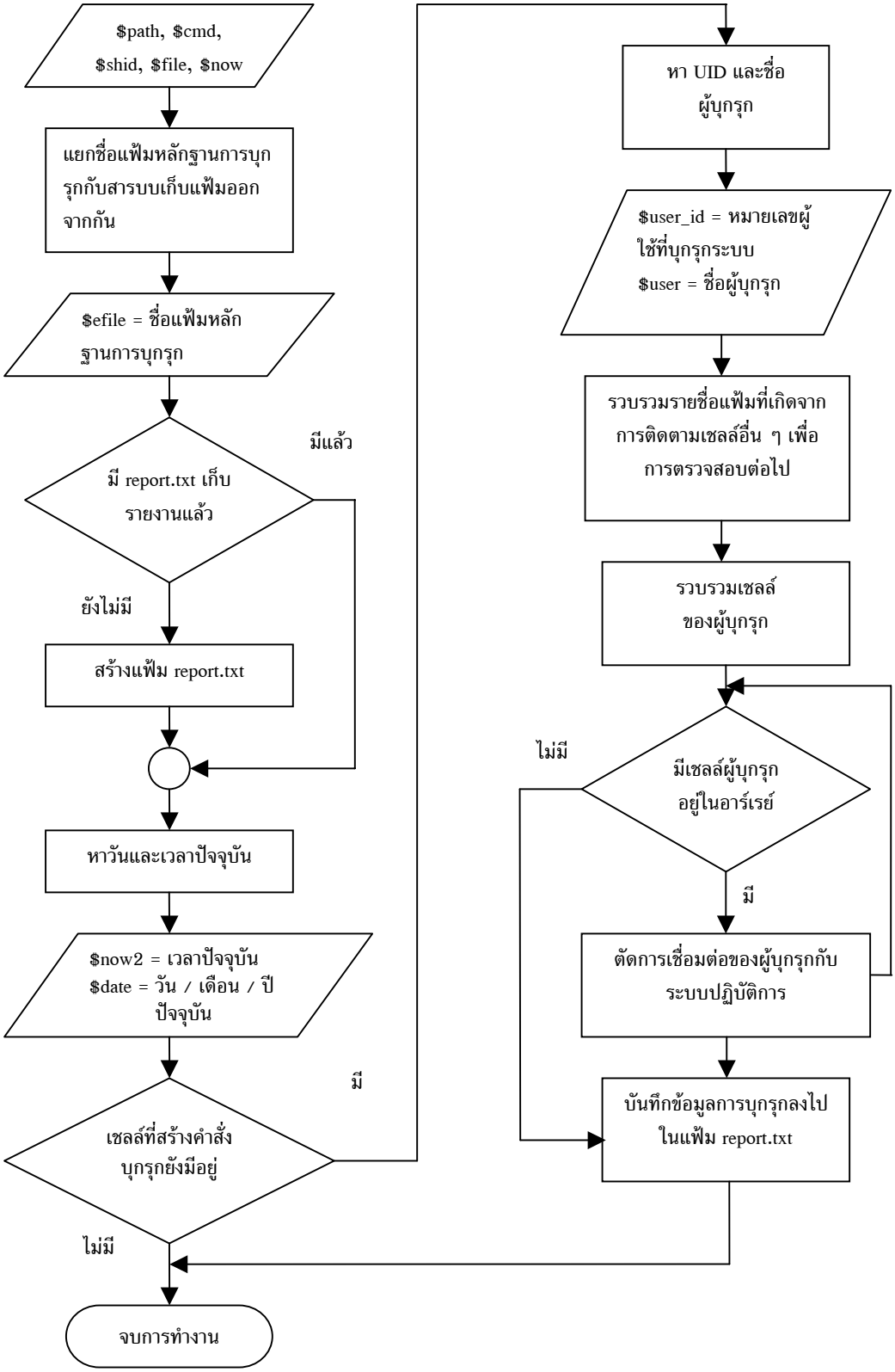
ภาพประกอบ 4.7 แสดงการทำงานของโปรแกรม count-tool (ต่อ)



ภาพประกอบ 4.8 แสดงฟังก์ชัน timbl_test ของโปรแกรม monitor.pl



ภาพประกอบ 4.8 แสดงฟังก์ชัน timbl_test ของโปรแกรม monitor.pl (ต่อ)



ภาพประกอบ 4.9 แสดงฟังก์ชัน evidence ของโปรแกรม monitor.pl

4.3 การพัฒนาโปรแกรม

ในการพัฒนาระบบตรวจจับการบุกรุกสำหรับวิทยานิพนธ์ชิ้นนี้ได้นำส่วนต่าง ๆ ที่ออกแบบไว้ในบทที่ 3 มาพัฒนาเป็นระบบตรวจจับการบุกรุก ระบบได้รับการพัฒนาขึ้นเพื่อใช้งานบนระบบปฏิบัติการลินุกซ์เรดแฮ็ทโดยใช้ภาษาเพิร์ลซึ่งมาพร้อมกับระบบปฏิบัติการลินุกซ์เรดแฮ็ทในระหว่างการติดตั้ง โปรแกรมตรวจจับการบุกรุกมีฟังก์ชันในการทำงานดังนี้

check_file() ฟังก์ชันนี้ทำหน้าที่ตรวจสอบข้อมูลที่ถูกจัดเก็บไว้ในแฟ้มข้อมูลว่าเป็นการทำงานของโปรเซสหรือไม่ โดยการเปิดแฟ้มข้อมูลขึ้นมาอ่านตั้งแต่บรรทัดแรกจนถึงบรรทัดสุดท้ายหากแฟ้มข้อมูลนั้นไม่มีคำว่า execve อยู่แสดงว่าเป็นแฟ้มข้อมูลที่ไม่ใช่การทำงานของโปรเซสซึ่งจะถูกลบทิ้งไป แต่หากมีคำว่า execve อยู่ก็จะเก็บข้อมูลตั้งบรรทัดที่มีคำว่า execve เป็นต้นไปจนกระทั่งถึงบรรทัดสุดท้ายของข้อมูลที่เก็บไว้ในแฟ้มข้อมูลแล้วบันทึกข้อมูลลงไปในแฟ้มข้อมูลใหม่อีกแฟ้มหนึ่ง แล้วส่งต่อการทำงานให้กับฟังก์ชัน rename_file

รูปแบบการเรียกใช้งาน

check_file($file)

โดยที่ $file คือ ตัวแปรที่เก็บชื่อแฟ้มข้อมูลที่เกิดจากการติดตามการทำงานของโปรเซสด้วยคำสั่ง strace ของระบบปฏิบัติการลินุกซ์เรดแฮ็ท

rename_file() ฟังก์ชันนี้ทำหน้าที่เปลี่ยนชื่อแฟ้มข้อมูลที่เก็บพฤติกรรมการทำงานของโปรเซสไว้ให้มีชื่อสัมพันธ์กับการทำงาน โดยกำหนดให้รูปแบบของชื่อแฟ้มข้อมูลมีลักษณะเรียงตามลำดับดังนี้คือ ชื่อคำสั่งหรือโปรแกรมที่ถูกเรียก-หมายเลขเชลล์ที่ทำงาน-หมายเลขโปรเซสที่ทำงาน จากนั้นส่งการทำงานต่อให้กับฟังก์ชัน count_syscall

รูปแบบการเรียกใช้งาน

rename_file($file, $command_path, $command, $time)

โดยที่ $file คือ ชื่อแฟ้มข้อมูลที่จะเปลี่ยนชื่อ

$command_path คือ เส้นทางการเรียกใช้คำสั่งหรือโปรแกรม

$command คือ คำสั่งหรือโปรแกรมที่ถูกเรียกใช้งาน

$time คือ เวลาที่พบแฟ้มข้อมูลที่จะตรวจสอบ

count_syscall() ฟังก์ชันนี้ทำหน้าที่เรียกใช้งานโปรแกรม count-tool ซึ่งทำหน้าที่สร้างแฟ้มข้อมูลนำเข้าที่ต้องการให้โปรแกรม TiMBL แยกประเภทคลาสการทำงาน

แฟ้มข้อมูลทดสอบที่เกิดขึ้นนี้เป็นแฟ้มข้อมูลที่มีรูปแบบการจัดเก็บข้อมูลแบบเดียวกับแฟ้มข้อมูลฝึกสอนระบบที่จะกล่าวถึงในหัวข้อถัดไป เมื่อโปรแกรม count-tool ทำงานเสร็จเรียบร้อยแล้ว ฟังก์ชัน count_syscall จะส่งการทำงานต่อให้กับฟังก์ชัน timbl_test

รูปแบบการเรียกใช้งาน

count_syscall($file, $command_path, $command, $shell_id, $time)

โดยที่ $file คือ ชื่อแฟ้มข้อมูลที่จะนับความถี่ของซิสเท็มคอล

$command_path คือ เส้นทางการเรียกใช้คำสั่งหรือโปรแกรม

$command คือ คำสั่งหรือโปรแกรมที่ถูกเรียกใช้งาน

$shell_id คือ หมายเลขของเชลล์ที่เป็นผู้สร้างโปรเซส

$time คือ เวลาที่พบแฟ้มข้อมูลที่จะตรวจสอบ

timbl_test( ) ฟังก์ชันนี้ทำหน้าที่เรียกใช้งานโปรแกรม TiMBL ซึ่งเป็นเครื่องมือที่นำมาใช้งานแยกประเภทคลาสการทำงานที่เกิดขึ้นภายในระบบซึ่งคลาสที่เป็นไปได้มีด้วยกัน 2 คลาสคือ คลาสการทำงานแบบปกติ (normal) และคลาสการทำงานที่ผิดปกติ (abnormal) นอกจากฟังก์ชัน timbl_test จะกำหนดคลาสการทำงานให้โปรเซสแล้วยังมีการทำสัญลักษณ์การบุกรุกให้กับคำสั่งของระบบปฏิบัติการที่มีความผิดปกติอีกด้วย หากการทำงานแยกประเภทได้คลาสที่เป็นการทำงานแบบปกติระบบตรวจจับการบุกรุกจะอนุญาตให้ผู้ใช้งานทำงานอื่น ๆ ในระบบต่อไปได้ แต่หากได้คลาสซึ่งเป็นการทำงานที่ผิดปกติจะส่งการทำงานต่อให้กับฟังก์ชัน evidence

รูปแบบการเรียกใช้งาน

timbl_test($test_file, $command_path, $command, $shell_id, $file, $time)

โดยที่ $test_file คือ ชื่อแฟ้มข้อมูลที่ใช้ทดสอบกับ TiMBL

$command_path คือ เส้นทางการเรียกใช้คำสั่งหรือโปรแกรม

$command คือ คำสั่งหรือโปรแกรมที่ถูกเรียกใช้งาน

$shell_id คือ หมายเลขของเชลล์ที่เป็นผู้สร้างโปรเซส

$file คือ แฟ้มข้อมูลที่เก็บรวบรวมพฤติกรรมการทำงานของโปรเซส

$time คือ เวลาที่พบแฟ้มข้อมูลที่เก็บรวบรวมพฤติกรรมการทำงานของโปรเซส

ฟังก์ชัน timbl_test เรียกโปรแกรม TiMBL ให้ทำงานโดยใช้คำสั่งดังนี้

/root/bin/timbl -mM -f /testit/train/tool_train -t $test > $test.out

คำสั่งนี้เป็นการสั่งให้โปรแกรม TiMBL ทำงานแยกประเภทข้อมูลซึ่งมีรายละเอียดแสดงไว้ในตาราง 4.1

ตาราง 4.1 อธิบายคำสั่งการแยกประเภทข้อมูล

ส่วนของคำสั่ง	ความหมาย
/root/bin/timbl	เรียกใช้งานโปรแกรม TiMBL
-mM	option ของโปรแกรม TiMBL ซึ่งเป็นการวัดปริมาณโดยใช้ความเหมือนกันที่มากที่สุดของสมาชิกในหน่วยความจำ
-f	กำหนดให้ TiMBL เรียนรู้ข้อมูลจากแฟ้มข้อมูลฝึกสอนระบบคือ /testit/train/tool_train
-t	กำหนดให้โปรแกรม TiMBL ทำงานแยกประเภทคลาสโดยใช้แฟ้มทดสอบที่เก็บอยู่ในตัวแปร $test
>	กำหนดให้เปลี่ยนทิศทางการแสดงข้อมูลจากจอภาพไปเก็บไว้ในแฟ้มข้อมูล โดยที่ชื่อของแฟ้มข้อมูลถูกเก็บไว้ในตัวแปร $test.out

ในการทำงานแยกประเภทข้อมูลของ TiMBL เมื่อพฤติกรรมของโปรเซสที่นำมาทดสอบตรงกันกับพฤติกรรมของโปรเซสที่จัดเก็บไว้ในแฟ้มข้อมูลฝึกสอนระบบ TiMBL จะแสดงผลการทำงานไว้ในแฟ้มข้อมูลผลลัพธ์ด้วยข้อความว่า “0/1 (0.000000), of which 1 exact matches” แต่ในกรณีที่พฤติกรรมของโปรเซสที่นำมาทดสอบไม่ตรงกันกับพฤติกรรมของโปรเซสที่จัดเก็บไว้ในแฟ้มข้อมูลฝึกสอนระบบ TiMBL จะแสดงผลการทำงานไว้ในแฟ้มข้อมูลผลลัพธ์ด้วยข้อความว่า “0/1 (0.000000)” ในงานวิจัยนี้ได้ทดสอบความสามารถในการจำแนกคลาสของ TiMBL ด้วย โดยจะกล่าวถึงผลในการแยกประเภทไว้ในส่วนของภาคผนวก

การทำงานแยกประเภทข้อมูลของ TiMBL จะมีการรายงานการทำงานออกมาทางจอภาพ แต่จะถูกเปลี่ยนทิศทางการแสดงผลไปเก็บไว้ในแฟ้มข้อมูลผลลัพธ์ซึ่งตัวอย่างการรายงานผลบางส่วนถูกแสดงไว้ในภาพประกอบ 4.10

evidence() ฟังก์ชันนี้ทำหน้าที่บันทึกข้อมูลการบุกรุกเก็บไว้เป็นหลักฐาน และตัดการเชื่อมต่อของผู้บุกรุกจากระบบปฏิบัติการ

รูปแบบการเรียกใช้งาน

evidence($command_path, $command, $shell_id, $file, $time)

โดยที่ $command_path คือ เส้นทางการเรียกใช้คำสั่งหรือโปรแกรม

$command คือ คำสั่งหรือโปรแกรมที่ถูกเรียกใช้งาน

$shell_id คือ หมายเลขของเชลล์ที่เป็นผู้สร้างโปรเซส

$file คือ ชื่อแฟ้มข้อมูลที่เก็บพฤติกรรมการทำงานบุกรุกระบบของโปร-เซส

$time คือ เวลาที่พบแฟ้มข้อมูลที่เก็บรวบรวมพฤติกรรมการทำงานของโปรเซส

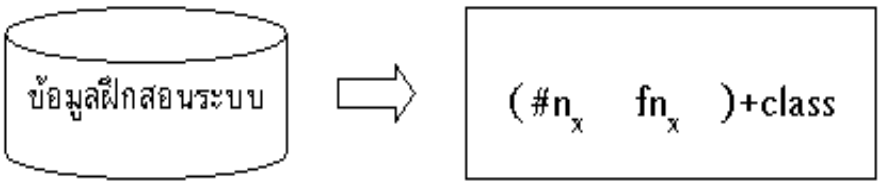
```
Phase 1: Reading Datafile: /testit/train/tool_train
Start:          0 @ Wed Jun  8 09:31:45 2005
Finished:    2290 @ Wed Jun  8 09:31:48 2005
Calculating Entropy          Wed Jun  8 09:31:48 2005
Lines of data     : 2290
DB Entropy        : 5.4691583
Number of Classes : 73

Feats      Vals      X-square      Variance      InfoGain      GainRatio
   1      1  0.0000000        0.0000000     0.0000000     0.0000000
   2      1  0.0000000        0.0000000     0.0000000     0.0000000
                                    ...
Size of MVDM[516] = 32 Bytes
Total Size of MVDM matrices 1096368 Bytes

Weighting     : No Weighting

Tested:       1 @ Wed Jun  8 09:31:50 2005
Ready:        1 @ Wed Jun  8 09:31:50 2005
Seconds taken: 1 (1.00 p/s)
0/1 (0.000000), of which 1 exact matches
```

ภาพประกอบ 4.10 ตัวอย่างรายงานผลการทำงานของ TiMBL ซึ่งเป็นข้อมูลที่เกิดจากการเปลี่ยนทิศทางการแสดงผลทางจอภาพมาเก็บไว้ในแฟ้มข้อมูล

ในส่วนนี้ได้นำเสนอการพัฒนาโปรแกรมซึ่งแสดงฟังก์ชันต่าง ๆ ในการทำงานและพารามิเตอร์ในแต่ละฟังก์ชัน รวมทั้งคำสั่งเรียกใช้งานและผลที่ได้รับจากการทำงานของโปรแกรม TiMBL แล้ว ในหัวข้อต่อไปจะกล่าวถึงการพัฒนาข้อมูลฝึกสอนระบบเพื่อใช้งานและแฟ้มข้อมูลหลักที่ระบบตรวจจับการบุกรุกใช้ในระหว่างการทำงาน

4.4 การพัฒนาข้อมูลฝึกสอนระบบเพื่อใช้งาน

จากหัวข้อ 3.4 ของบทที่ 3 ที่ได้กล่าวถึงแนวคิดการสร้างข้อมูลฝึกสอนระบบมาแล้วว่าจะต้องมีการเลือกคำสั่งที่ต้องการ และมีขั้นตอนการรวบรวมพฤติกรรมการทำงานของโปรเซสที่ต้องการอย่างไร ในส่วนนี้เป็นการพัฒนาข้อมูลฝึกสอนระบบเพื่อนำไปใช้งานโดยหลังจากที่ทราบความถี่ในการเรียกใช้งานซิสเท็มคอลหมายเลขต่าง ๆ แล้วจะต้องนำเอาข้อมูลทั้งหมดมาเปรียบเทียบกันว่าควรเลือกซิสเท็มคอลหมายเลขใดบ้างมาใช้เป็นตัวแทนพฤติกรรมการทำงานของโปรเซสเก็บไว้ในแฟ้มข้อมูลฝึกสอนระบบ ในการเลือกหมายเลขซิสเท็มคอลมาเป็นตัวแทนนี้ใช้ลักษณะเด่นจากความถี่ของซิสเท็มคอลเป็นประการสำคัญ โดยเลือกความถี่ของซิสเท็มคอลที่ทำให้เกิดความแตกต่างกันอย่างชัดเจนระหว่างคลาสการทำงานแบบปกติ และคลาสการทำงานที่ผิดปกติซึ่งในที่นี้คือ คลาส normal และ abnormal ตามลำดับ รูปแบบในการจัดเก็บข้อมูลในแฟ้มข้อมูลฝึกสอนระบบแสดงไว้ด้วยภาพประกอบ 4.11 และข้อมูลที่เก็บไว้ในแฟ้มข้อมูลฝึกสอนระบบบางส่วนแสดงไว้ด้วยภาพประกอบ 4.12



ภาพประกอบ 4.11 รูปแบบการจัดเก็บข้อมูลในแฟ้มข้อมูลฝึกสอนระบบบนลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1

จากภาพประกอบ 4.11 รายละเอียดของสัญลักษณ์เป็นดังนี้

- $\#n_x$ หมายถึง หมายเลขซิสเท็มคอลที่ถูกเรียกใช้งานซึ่งเป็นหมายเลขจากค่าน้อยไปหาค่ามาก ค่าที่มีความเป็นไปได้อยู่ระหว่าง 1 - 190
- fn_x หมายถึง ความถี่ของการเรียกใช้งานซิสเท็มคอลหมายเลขที่ n จากการทำงานของโปรเซสที่กำลังพิจารณาอยู่

- (#n_x fn_x)+ หมายถึง รูปแบบของข้อมูลซึ่งประกอบด้วยหมายเลขซิส-เท็มคอล อักขระว่าง (แท็บ) ความถี่ของการเรียกใช้งานซิสเท็มคอลหมาย-เลขที่ n จากการทำงานของโปรเซสที่กำลังพิจารณาอยู่ และอักขระว่าง (แท็บ) ที่ได้จัดเก็บไว้ ซึ่งเป็นรูปแบบที่ปรากฏอยู่อย่างน้อยที่สุดหนึ่งครั้งภายในแฟ้มข้อมูลฝึกสอนระบบ
- class หมายถึง ประเภทของพฤติกรรมในการทำงานซึ่งในที่นี้มีอยู่ 2 กลุ่ม คือ normal และ abnormal

```
3   10  5   18  6   14  15  0   20  5   ...  174 11  175 0   190 0   normal
3   188 5   53  6   54  15  0   20  1   ...  174 1   175 0   190 0   normal
3   379 5   34  6   35  15  0   20  1   ...  174 1   175 0   190 0   normal
3   18  5   25  6   27  15  0   20  1   ...  174 1   175 0   190 0   normal
3   0   5   0   6   0   15  0   20  0   ...  174 0   175 0   190 0   normal
3   2   5   3   6   3   15  0   20  1   ...  174 0   175 0   190 0   normal
3   1   5   3   6   2   15  0   20  1   ...  174 0   175 0   190 0   normal
3   5   5   7   6   7   15  0   20  1   ...  174 0   175 0   190 0   normal
3   106 5   57  6   33  15  0   20  1   ...  174 0   175 0   190 0   normal
3   39  5   28  6   36  15  0   20  1   ...  174 56  175 8   190 0   normal
3   1   5   3   6   2   15  0   20  1   ...  174 8   175 0   190 0   normal
3   26  5   20  6   21  15  0   20  3   ...  174 38  175 9   190 0   normal
3   20  5   33  6   30  15  0   20  1   ...  174 0   175 0   190 0   normal
3   18  5   23  6   18  15  0   20  2   ...  174 0   175 0   190 0   normal
                                         ...
3   1   5   3   6   2   15  0   20  1   ...  174 2   175 1   190 1   abnormal
3   19  5   24  6   22  15  0   20  2   ...  174 0   175 0   190 0   abnormal
3   8   5   14  6   11  15  0   20  3   ...  174 26  175 5   190 1   abnormal
3   1   5   3   6   4   15  0   20  1   ...  174 4   175 2   190 1   abnormal
3   1   5   3   6   2   15  0   20  1   ...  174 2   175 1   190 1   abnormal
3   6   5   16  6   13  15  0   20  2   ...  174 0   175 0   190 0   abnormal
3   1   5   3   6   4   15  0   20  2   ...  174 26  175 5   190 1   abnormal
3   11  5   18  6   19  15  0   20  1   ...  174 0   175 0   190 0   abnormal
3   29  5   14  6   28  15  0   20  2   ...  174 48  175 75  190 0   abnormal
3   11  5   14  6   13  15  0   20  2   ...  174 26  175 9   190 0   abnormal
3   14  5   14  6   14  15  0   20  2   ...  174 28  175 13  190 0   abnormal
3   16  5   14  6   17  15  0   20  2   ...  174 31  175 45  190 0   abnormal
```

ภาพประกอบ 4.12 ตัวอย่างข้อมูลพฤติกรรมการทำงานของโปรเซสที่เก็บไว้ในแฟ้มข้อมูลฝึกสอนระบบบนลินุกซ์เรดแฮ็ท เวอร์ชัน 6.1

4.5 แฟ้มข้อมูลหลักที่ระบบตรวจจับการบุกรุกใช้ในระหว่างการทำงาน

การทำงานของระบบตรวจจับการบุกรุกมีการดำเนินงานที่เกี่ยวข้องกับแฟ้มข้อมูลจำนวนมาก ในการดำเนินงานกับแฟ้มข้อมูลต่าง ๆ นั้นมีแฟ้มข้อมูลจำนวนหนึ่งเป็นแฟ้มที่มี

ความสำคัญซึ่งระบบตรวจจับการบุกรุกจะต้องใช้ในการตรวจสอบการบุกรุกซึ่งแฟ้มเหล่านี้ประกอบด้วยแฟ้มที่ใช้สำหรับตรวจสอบเส้นทางของคำสั่งที่ถูกเรียกใช้งานว่าเป็นคำสั่งของระบบปฏิบัติการหรือไม่ แฟ้มสำหรับบันทึกคำสั่งของระบบปฏิบัติการที่แสดงพฤติกรรมการทำงานที่ผิดปกติ และแฟ้มสำหรับบันทึกข้อมูลที่สามารถใช้เป็นหลักฐานเมื่อมีการบุกรุกระบบเกิดขึ้น แฟ้มข้อมูลที่กล่าวมานี้มีรายละเอียดดังต่อไปนี้

แฟ้มข้อมูลเพื่อการตรวจสอบเส้นทาง แฟ้มข้อมูลเพื่อการตรวจสอบเส้นทางนี้เป็นแฟ้มที่เก็บคำสั่งและเส้นทางการเรียกใช้งานคำสั่งของลินุกซ์เรดแฮ็ทเอาไว้ซึ่งคำสั่งเหล่านั้นคือคำสั่งที่เลือกมาเก็บข้อมูลพฤติกรรมการทำงานของโปรเซสที่ได้กล่าวไว้ในหัวข้อ 3.4 ของบทที่ 3 รูปแบบข้อมูลที่จัดเก็บลงในแฟ้มข้อมูลเพื่อการตรวจสอบเส้นทางเป็นดังนี้คือ

%cmd_path=(“command”,“path”, …, “command”, “path”);

รายละเอียดของการจัดเก็บเป็นดังนี้

- %cmd_path หมายถึง ชื่อของอาร์เรย์ที่กำหนดขึ้นเพื่อใช้เรียกแทนสมาชิกที่เป็นคำสั่งของระบบปฏิบัติการและเส้นทางการเรียกใช้งาน
- command หมายถึง คำสั่งของระบบปฏิบัติการ
- path หมายถึง เส้นทางการเรียกใช้งานคำสั่งของระบบปฏิบัติการลินุกซ์เรด-แฮ็ท

ตัวอย่างข้อมูลที่บันทึกไว้ในแฟ้มข้อมูลเพื่อการตรวจสอบเส้นทางเป็นไปดังภาพประกอบ 4.13

%path=("ls","/bin","ps","/bin", … , ″who","/usr/bin","whoami","/usr/bin");

ภาพประกอบ 4.13 ตัวอย่างข้อมูลที่จัดเก็บไว้ในแฟ้มข้อมูลเพื่อการตรวจสอบเส้นทาง

แฟ้มข้อมูลสัญลักษณ์บ่งชี้การบุกรุก แฟ้มข้อมูลสัญลักษณ์บ่งชี้การบุกรุกนี้เป็นแฟ้มที่เก็บคำสั่งของระบบปฏิบัติการที่มีพฤติกรรมไปในทางการบุกรุกระบบ เมื่อระบบตรวจจับการบุกรุกพบว่าคำสั่งของระบบปฏิบัติการมีพฤติกรรมที่ผิดปกติ ระบบตรวจจับการบุกรุกจะตรวจสอบแฟ้มข้อมูลนี้ก่อนว่าคำสั่งที่มีพฤติกรรมผิดปกตินั้นได้รับการบันทึกเก็บไว้หรือยัง ในกรณีที่ยังไม่ได้บันทึกไว้ระบบตรวจจับการบุกรุกจะบันทึกคำสั่งนั้นลงไปในแฟ้มนี้เพื่อให้ผู้ดูแลระบบสามารถตรวจสอบความถูกต้องในภายหลังได้และดำเนินการกับคำสั่งนั้นในฐานะการ

คุกคามระบบ แต่ถ้าคำสั่งที่มีพฤติกรรมผิดปกตินั้นได้รับการบันทึกไว้ในแฟ้มข้อมูลนี้แล้วระบบตรวจจับการบุกรุกจะไม่ดำเนินการกับคำสั่งนั้นเพราะถือว่าเป็นคำสั่งที่รอการพิจารณาอยู่ซึ่งอาจจะเป็นคำสั่งใหม่ของระบบปฏิบัติการที่ถูกเรียกใช้งานโดยที่ยังไม่ได้จัดเก็บพฤติกรรมการทำงานมาก่อน จึงรอให้ผู้ดูแลระบบเข้ามาพิจารณาตรวจสอบเพื่อมิให้กระทบกระเทือนกับการทำงานของผู้ใช้ในระบบต่อไปอีก รูปแบบของการจัดเก็บข้อมูลในแฟ้มสัญลักษณ์การบุกรุกเป็นดังนี้

command, path, file_evidence

รายละเอียดของการจัดเก็บคือ

- command หมายถึง คำสั่งของลินุกซ์เรดแฮ็ทที่มีพฤติกรรมการทำงานผิดปกติ
- path หมายถึง เส้นทางของคำสั่งบนระบบปฏิบัติการ
- file_evidence หมายถึง ชื่อแฟ้มข้อมูลที่เก็บพฤติกรรมการทำงานคุกคามระบบของคำสั่งนั้น

ตัวอย่างข้อมูลที่บันทึกไว้ในแฟ้มสัญลักษณ์การบุกรุกเป็นไปดังภาพประกอบ 4.14

```
sh,/bin/sh,/testit/sh-9013-7893
ktop,/usr/bin/ktop,/testit/ktop-20441-20981
grep,/bin/grep,/testit/grep-28355-28386
tput,/usr/bin/tput,/testit/tput-6042-6096
sshd,/usr/sbin/sshd,/testit/sshd-1841-2247
bash,/bin/bash,/testit/bash-2662-5979
```

ภาพประกอบ 4.14 ตัวอย่างข้อมูลที่บันทึกไว้ในแฟ้มสัญลักษณ์การบุกรุก

จากภาพประกอบ 4.14 ความหมายของข้อมูลในแต่ละบรรทัดสามารถอธิบายได้จากตัวอย่างในบรรทัดแรกที่ได้บันทึกไว้ดังนี้

- sh เป็นคำสั่งของลินุกซ์เรดแฮ็ทที่มีพฤติกรรมการทำงานผิดปกติ
- /bin/sh เป็นเส้นทางการทำงานของคำสั่ง sh
- /testit/sh-9013-7879 เป็นแฟ้มที่เก็บพฤติกรรมการทำงานของคำสั่ง sh ที่ผู้ดูแลระบบสามารถเข้าไปตรวจสอบได้ในสารบบ /testit/evidence

แฟ้มข้อมูลหลักฐานการบุกรุก แฟ้มข้อมูลหลักฐานการบุกรุกเป็นแฟ้มที่เก็บรวบรวมข้อมูลการบุกรุกที่เกิดขึ้น รูปแบบของการจัดเก็บข้อมูลในแฟ้มหลักฐานการบุกรุกเป็นดังนี้

uid user command evidence date start-time stop-time

รายละเอียดของการจัดเก็บคือ

- uid หมายถึง หมายเลขผู้ใช้งานซึ่งเป็นผู้บุกรุกระบบ
- user หมายถึง ชื่อผู้บุกรุกระบบ
- command หมายถึง คำสั่งหรือโปรแกรมที่มีพฤติกรรมการทำงานในเชิงคุกคามระบบ
- evidence หมายถึง แฟ้มข้อมูลที่เก็บบันทึกพฤติกรรมการทำงานของคำสั่งหรือโปรแกรมที่ทำงานในเชิงคุกคามระบบ
- date หมายถึง วัน เดือน ปีที่พบการคุกคามระบบ
- start-time หมายถึง เวลาที่พบการบุกรุก
- stop-time หมายถึง เวลาที่ตรวจสอบเสร็จและยืนยันได้ว่าเป็นการบุกรุกระบบ

ตัวอย่างข้อมูลที่บันทึกไว้ในแฟ้มหลักฐานการบุกรุกเป็นไปดังภาพประกอบ 4.15

UID	USER	COMMAND	EVIDENCE	DATE	START-TIME	STOP-TIME
503	user4	adv1	adv1-930-9286	Wed 27/Apr/2005	22:36:0	22:36:6
500	user1	adv3	adv3-20403-21949	Wed 27/Apr/2005	22:39:52	22:39:58
502	user3	cwho	cwho-25257-14017	Wed 27/Apr/2005	23:0:47	23:0:54
501	user2	dsr	dsr-11521-24010	Wed 27/Apr/2005	23:14:27	23:14:33
503	user4	irc	irc-9249-13434	Wed 27/Apr/2005	23:20:38	23:21:23
500	user1	lsof	lsof-14084-17052	Thu 28/Apr/2005	0:0:49	0:0:56
502	user3	lsb	lsb-929-2314	Thu 28/Apr/2005	9:15:21	9:15:27
502	user3	reh	reh-22898-24319	Thu 28/Apr/2005	9:21:59	9:22:6
501	user2	wts	wts-9172-10859	Thu 28/Apr/2005	9:28:3	9:28:9
502	user3	xwho	xwho-30557-31707	Thu 28/Apr/2005	9:32:52	9:32:58
500	user1	ptrac	ptrac-1533-3015	Thu 28/Apr/2005	10:16:7	10:16:24
501	user2	sdi	sdi-13476-14752	Thu 28/Apr/2005	10:41:21	10:41:27

ภาพประกอบ 4.15 ตัวอย่างข้อมูลที่บันทึกไว้ในแฟ้มหลักฐานการบุกรุก

จากภาพประกอบ 4.15 ความหมายของข้อมูลในแต่ละบรรทัดสามารถอธิบายได้จากตัวอย่างในบรรทัดแรกที่ได้บันทึกไว้คือ

- 503 เป็นหมายเลขผู้ใช้งานที่เรียกคำสั่งหรือโปรแกรมซึ่งมีพฤติกรรมการทำงานในเชิงคุกคามระบบ

- user4 เป็นชื่อผู้ใช้งานที่มีหมายเลขผู้ใช้เท่ากับ 503 ซึ่งเป็นผู้เรียกใช้คำสั่งหรือโปรแกรมที่มีพฤติกรรมการทำงานในเชิงคุกคามระบบ
- adv1 เป็นชื่อโปรแกรมหรือคำสั่งที่มีพฤติกรรมการทำงานคุกคามระบบ
- adv1-930-9286 เป็นแฟ้มที่เก็บพฤติกรรมการบุกรุกระบบของโปรแกรมหรือคำสั่งที่ชื่อว่า adv1
- Wed 27/Apr/2005 เป็นวัน เดือน ปีที่พบว่าโปรแกรมหรือคำสั่ง adv1 ทำงานคุกคามระบบ
- 22:36:0 เป็นเวลาที่ตรวจพบแฟ้มข้อมูลที่เก็บพฤติกรรมการทำงานของคำ-สั่งหรือโปรแกรม adv1
- 22:36:6 เป็นเวลาที่ตรวจสอบการทำงานเสร็จและพบว่า adv1 มีพฤติ-กรรมการทำงานบุกรุกระบบ

ในหัวข้อที่ผ่านมาได้กล่าวถึงการพัฒนาข้อมูลฝึกสอนระบบและแฟ้มข้อมูลที่ระบบตรวจจับการบุกรุกใช้งานในการตรวจสอบการบุกรุกแล้ว ในหัวข้อถัดไปจะกล่าวถึงสารบบและแฟ้มข้อมูลสำหรับงานวิจัยนี้

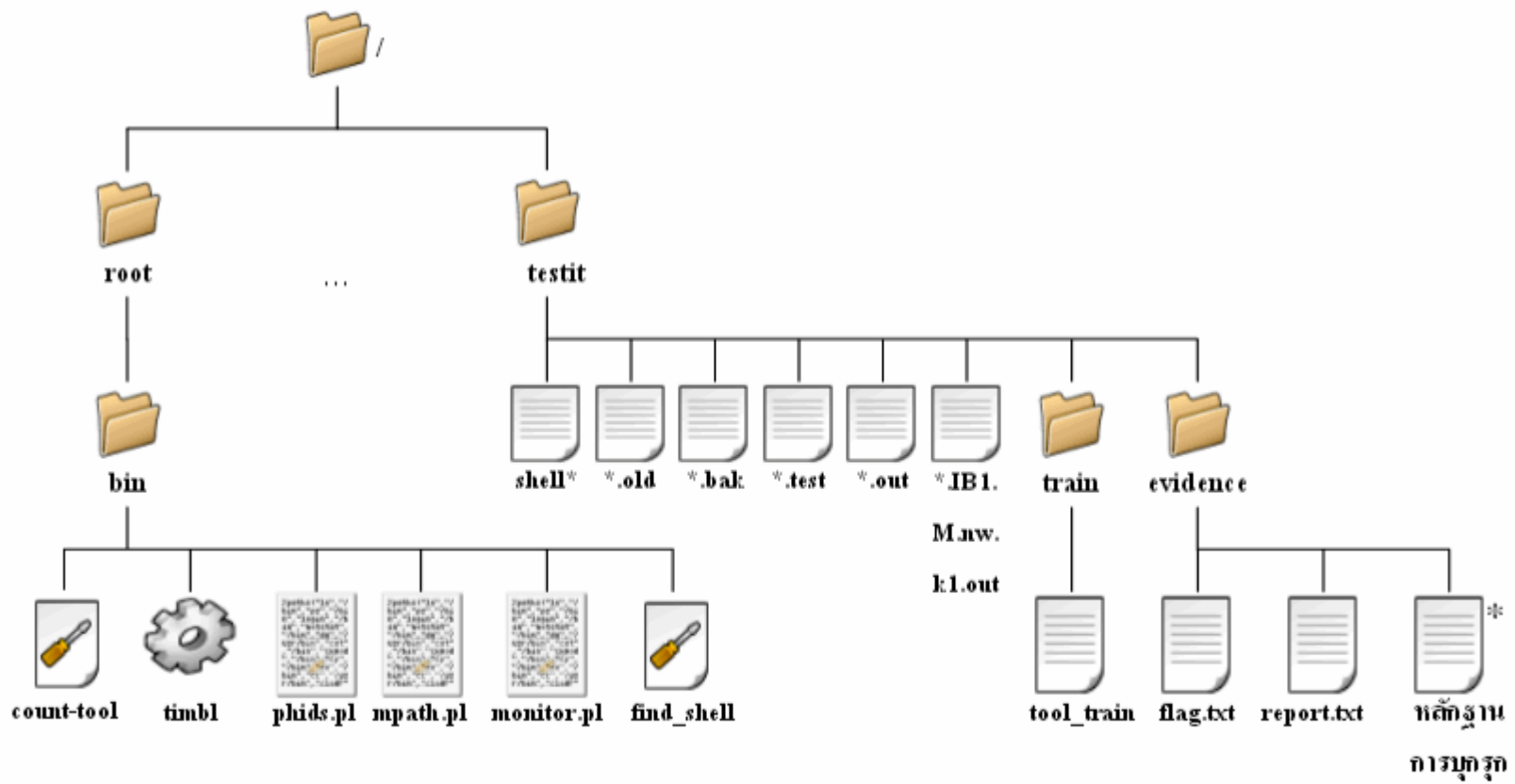
4.6 สารบบและแฟ้มข้อมูล

สารบบและแฟ้มข้อมูลที่เกี่ยวข้องกับการดำเนินงานเพื่อตรวจสอบการบุกรุกระบบปฏิบัติการที่ได้พัฒนาขึ้นมีรายละเอียดดังต่อไปนี้

- สารบบ /testit เป็นทีเก็บรวบรวมแฟ้มข้อมูลที่เกิดจากการติดตามการทำงานของโปรเซส และแฟ้มข้อมูลชั่วคราวในระหว่างการทำงานตรวจสอบการบุกรุก
- สารบบ /testit/evidence เป็นที่เก็บรวบรวมแฟ้มข้อมูลสัญลักษณ์บ่งชี้ แฟ้มข้อมูลรายงานการบุกรุก และแฟ้มข้อมูลที่เก็บพฤติกรรมการบุกรุก
- สารบบ /testit/train เป็นที่เก็บแฟ้มข้อมูลฝึกสอนระบบ
- สารบบ /root/bin เป็นที่เก็บรวบรวมโปรแกรมแยกประเภทการบุกรุก โปรแกรมนับความถี่ของซิสเท็มคอล โปรแกรมเริ่มต้นการทำงานตรวจจับการบุกรุก แฟ้มข้อมูลคำสั่งและเส้นทาง โปรแกรมค้นหาเชลล์ และโปรแกรมตรวจสอบความปลอดภัยให้กับระบบปฏิบัติการ
- แฟ้มข้อมูล shell* เป็นแฟ้มข้อมูลที่ใช้เก็บข้อมูลที่เกิดจากการติดตามการทำงานของโปรเซสก่อนที่จะถูกเปลี่ยนชื่อในระหว่างการทำงาน

- แฟ้มข้อมูล *.old เป็นแฟ้มข้อมูลที่ถูกเปลี่ยนชื่อมาจาก shell* เพื่อใช้ตรวจสอบการทำงานของโปรเซส
- แฟ้มข้อมูล *.bak เป็นแฟ้มข้อมูลที่เก็บข้อมูลการทำงานของโปรเซสตั้งแต่คำสั่ง execve จนกระทั่งสิ้นสุดการทำงานของโปรเซสซึ่งอ่านข้อมูลมาจากแฟ้มข้อมูล *.old
- แฟ้มข้อมูล *.test เป็นแฟ้มข้อมูลที่เก็บข้อมูลที่จะนำไปทดสอบกับโปรแกรมแยกประเภท
- แฟ้มข้อมูล *.out เป็นแฟ้มข้อมูลที่เก็บข้อมูลการเปลี่ยนทิศทางของข้อมูลจากจอภาพ
- แฟ้มข้อมูล *.IB1.M.nw.k1.out เป็นแฟ้มข้อมูลที่เก็บคลาสจากการแยกประเภทข้อมูลของโปรแกรม TiMBL
- แฟ้ม phids.pl เป็นแฟ้มของโปรแกรมเริ่มต้นการทำงานของระบบตรวจจับการบุกรุก
- แฟ้มข้อมูล mpath.pl เป็นแฟ้มข้อมูลที่เก็บคำสั่งและเส้นทางในการทำงานบนระบบปฏิบัติการลินุกซ์เรดแฮ็ท
- แฟ้ม monitor.pl เป็นแฟ้มของโปรแกรมที่ใช้ทำงานตรวจจับการบุกรุกเพื่อตรวจสอบความปลอดภัยให้กับระบบปฏิบัติการ
- แฟ้ม find_shell เป็นแฟ้มของโปรแกรมที่ใช้ติดตามการทำงานของเชลล์และการสร้างโปรเซสลูกของเชลล์
- แฟ้มข้อมูล count-tool เป็นแฟ้มของโปรแกรมที่ใช้ทำงานนับความถี่ของซิสเท็มคอลจากการเรียกใช้ของโปรเซสแยกตามชื่อของซิสเท็มคอล
- แฟ้ม timbl เป็นแฟ้มของโปรแกรม TiMBL ที่ใช้ทำงานแยกประเภทข้อมูล
- แฟ้มข้อมูล tool_train เป็นแฟ้มข้อมูลฝึกสอนระบบ

แผนผังต้นไม้ (Tree Diagram) ของสารบบและแฟ้มข้อมูลที่เกี่ยวข้องกับการทำงานของระบบตรวจจับการบุกรุกที่ได้พัฒนาขึ้นเป็นดังภาพประกอบ 4.16



ภาพประกอบ 4.16 สารบบและแฟ้มข้อมูลของระบบตรวจจับการบุกรุก

4.7 สรุป

ในบทนี้ได้กล่าวถึงการพัฒนาระบบตรวจจับการบุกรุกซึ่งได้แสดงโครงสร้างการดำเนินงานที่เป็นโปรเซสหลักของระบบตรวจจับการบุกรุก ขั้นตอนการทำงานของโปรเซสหลักและฟังก์ชันต่าง ๆ ของระบบตรวจจับการบุกรุก การพัฒนาโปรแกรมตรวจจับการบุกรุกซึ่งแสดงวิธีการเรียกใช้งานฟังก์ชันในการทำงานของระบบตรวจจับการบุกรุก การพัฒนาข้อมูลฝึกสอนระบบเพื่อใช้ฝึกสอนระบบตรวจจับการบุกรุก แฟ้มข้อมูลที่ระบบตรวจจับการบุกรุกใช้ระหว่างการทำงาน ตลอดจนสารบบและแฟ้มข้อมูลของระบบตรวจจับการบุกรุกที่ได้พัฒนาขึ้น ในบทถัดไปจะนำเสนอในส่วนของการทดสอบระบบตรวจจับการบุกรุกและการปรับปรุงระบบตรวจจับการบุกรุกต่อไป