บทที่ 5

การพัฒนาระบบ

5.1 บทนำ

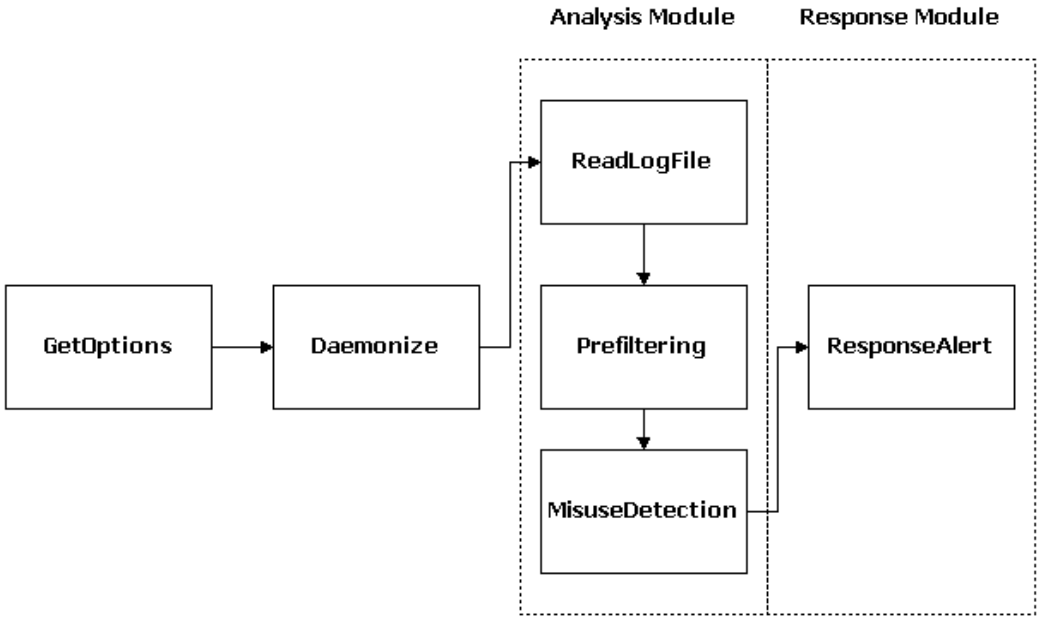
สำหรับบทนี้จะกล่าวถึงการพัฒนาโปรแกรมที่จะใช้ในการวิเคราะห์ล็อกไฟล์เพื่อตรวจจับการบุกรุกตามที่ได้ออกแบบไว้ในบทที่ผ่านมา โดยจะกล่าวถึงภาษาที่ใช้ในการพัฒนา การพัฒนาโปรแกรมให้ทำงานในลักษณะ daemon การพัฒนาโปรแกรมส่วนที่ทำหน้าที่ในการวิเคราะห์ข้อมูล การแจ้งเตือนการบุกรุก และตัวเลือกในการทำงานของโปรแกรม รวมทั้งการสร้างกฎเพื่อใช้สำหรับการตรวจจับการบุกรุก

5.2 ภาษาที่ใช้ในการพัฒนา

การพัฒนาโปรแกรมสำหรับวิเคราะห์ล็อกไฟล์ตามที่ได้ออกแบบนั้น พัฒนาโดยใช้ภาษา Perl ซึ่งภาษา Perl ได้นำเอาข้อดีหลาย ๆ อย่างของภาษา C, sed, awk และ sh (shell script) รูปแบบของภาษา Perl คล้ายกับภาษา C มาก และสาเหตุสำคัญที่เลือกภาษา Perl ในการพัฒนาเนื่องจากเป็นภาษาที่มีความยืดหยุ่นและมีประสิทธิภาพสูงในการจัดการข้อมูลประเภทข้อความโดยมีความสามารถในเรื่อง Regular Expression ซึ่งเป็นกฎเกณฑ์ที่ใช้การเปรียบเทียบข้อความเพื่อค้นหาข้อความที่มีรูปแบบเหมือนกัน จึงทำให้เหมาะสำหรับการวิเคราะห์ข้อมูลในล็อกไฟล์ซึ่งเป็นการค้นหาข้อมูลล็อกของเหตุการณ์ผิดปกติและการบุกรุก

5.3 แผนภาพโดยรวมของการพัฒนาโปรแกรม

การพัฒนาโปรแกรมตามที่ได้กล่าวไว้ในบทที่ 4 นั้น โปรแกรมจะทำงานในลักษณะของ daemon และครอบคลุมการทำงานในส่วนของ Analysis Module และ Response Module โดยแผนภาพรวมของการพัฒนาโปรแกรมแสดงได้ดังภาพประกอบที่ 5.1



ภาพประกอบ 5.1 แผนภาพรวมของการพัฒนาโปรแกรม

โดยโปรแกรมที่พัฒนาประกอบฟังก์ชันการทำงาน ดังนี้

- GetOptions เป็นส่วนที่รับตัวเลือกในการทำงานของโปรแกรม
- Daemonize การทำงานในลักษณะ daemon process
- โปรแกรมส่วน Analysis Module ประกอบด้วยฟังก์ชัน ReadLogFile, Prefiltering และ MisuseDetection
- โปรแกรมส่วน Response Module ประกอบด้วยฟังก์ชัน ResponseAlert

รายละเอียดในการพัฒนาของแต่ละส่วนจะกล่าวถึง ในหัวข้อที่ 5.4, 5.5, 5.6 และ 5.7 ตามลำดับ

5.4 ฟังก์ชันรับตัวเลือกการทำงานของโปรแกรม (GetOptions)

สำหรับการใช้โปรแกรมผู้ใช้สามารถระบุตัวเลือกสำหรับการทำงานได้ ซึ่งตัวเลือกสำหรับการสั่งให้โปรแกรมทำงานมีรายละเอียดดังตารางที่ 5.1

ตารางที่ 5.1 ตัวเลือกการทำงานของโปรแกรม

ตัวเลือก(Option)	การทำงาน
-t <log file> หรือ --tail <log file>	เป็นการระบุว่าให้โปรแกรมกระโดดไปยังส่วนท้ายของlog file ที่ระบุและคอยรับข้อมูลล็อกที่มีการเพิ่มเข้ามาในไฟล์ดังกล่าว แล้วทำการวิเคราะห์ข้อมูลเหล่านั้นเพื่อตรวจจับการบุกรุกไปเรื่อย ๆ
-f <PreFiltering Rule file> หรือ --filter <PreFiltering Rule file>	เป็นการกำหนดว่าให้อ่านข้อมูลกฎสำหรับการกรองเหตุการณ์ปกติจากไฟล์ที่กำหนด (PreFiltering Rule file)
-r <Misuse Rule file> หรือ --rules <Misuse Rule file>	เป็นการกำหนดว่าให้อ่านข้อมูลกฎสำหรับการตรวจจับและการแจ้งเตือน จากไฟล์ที่กำหนด (Misuse Rule file)

5.5 ฟังก์ชันการทำงานในลักษณะ daemon process (Daemonize)

daemon เป็นโปรเซสที่ทำงานอยู่เบื้องหลัง (background process) โดยถูกออกแบบให้ทำงานอย่างเป็นอิสระ สำหรับขั้นตอนในการพัฒนาโปรแกรมให้ทำงานในลักษณะ deamon ด้วยภาษา Perl มีขั้นตอนดังนี้

การทำงานในลักษณะ daemon process

1. Fork and Exit the Parent process

การเริ่มต้นสร้าง daemon เริ่มจากใช้ฟังก์ชัน fork() เพื่อสร้างโปรเซสลูก (child process) แล้วจบการทำงานของโปรเซสแม่ (parent process) ด้วยฟังก์ชัน exit() โดยมีจุดประสงค์เพื่อตัดความสัมพันธ์ของโปรเซสลูกที่สร้างขึ้นจากการควบคุมของ terminal หรือ login shell และเป็นการย้ายโปรเซสลูกที่ถูกสร้างขึ้นใหม่ออกไปจากกลุ่มที่โปรเซสแม่อยู่ ซึ่งจะส่งผลทำให้โปรเซสลูกมีโปรเซสแม่เป็น init process แทน โดยคำสั่งที่ใช้คือ

```
$pid = fork;
exit if $pid;
die "Couldn't fork: $!" unless defined($pid);
```

ฟังก์ชัน fork() จะส่งค่าหมายเลขโปรเซสของโปรเซสลูก ซึ่งจะมีค่าที่ไม่เท่ากับศูนย์ หรือส่งค่าเป็น -1 ในกรณีที่ ไม่สามารถสร้างโปรเซสลูกได้

2. Set File Creation Mode

สำหรับการกำหนดโหมดของไฟล์ที่สร้างขึ้นนั้นจะอาศัยค่าของ umask โดยค่าดังกล่าวใช้สำหรับการกำหนดค่า permission ในการเข้าถึงไฟล์ต่าง ๆที่สร้างขึ้นใหม่ โดยคำสั่งที่ใช้คือ

```
umask 0;
```

การกำหนดให้ค่า umask เป็น 0 จะทำให้มีสิทธิในการเข้าถึงไฟล์ที่สร้างจาก daemon เป็นแบบ full access นั่นคือผู้ใช้ทุกคนมีสิทธิในการอ่านและเขียนไฟล์ที่สร้างขึ้นได้ โดยไฟล์ที่สร้างขึ้นจะมีโหมดเป็น 666 หรือ ค่า permission mode เป็น -rw-rw-rw-

3. Create a Unique Session ID (SID)

สำหรับโปรเซสลูกที่ถูกสร้างขึ้นนั้นจะได้หมายเลขเซสชั่น (SID) จากเคอร์นอลเพื่อที่จะทำงานต่อไป โดยการเรียกใช้ฟังก์ชัน setsid() เพื่อกำหนดให้โปรเซสนั้นเป็น session leader และ group leader และเพื่อเป็นการตัดการควบคุมจาก terminal ซึ่งฟังก์ชันนี้เป็นฟังก์ชันที่อยู่ในโมดูล POSIX ซึ่งคำสั่งที่ใช้คือ

```
use POSIX;
# dissociate from the controlling terminal
POSIX::setsid() or die "Can't start new session: $!";
```

4. Change The Working Directory

สำหรับการเปลี่ยนไดเรคตอรี่ที่กำลังใช้งานอยู่นั้น หากโปรเซสมีการเริ่มต้นทำงานจากไดเรคตอรี่ที่อยู่ในพาร์ติชั่นที่ถูก mount อยู่ แต่หากต่อมาผู้ดูแลระบบต้องการ unmount พาร์ติชั่นนั้นก็จะไม่สามารถทำได้ จนกว่าโปรเซสนั้นจะสิ้นสุดการทำงาน ดังนั้นจึงควรเปลี่ยนไดเรคตอรี่ที่กำลังใช้งานอยู่ของ daemon ไปยังไดเรคตอรี่อื่น แต่ไม่แนะนำให้ใช้ไดเรคตอรี่ราก (/) เนื่องจากอาจจะทำให้ข้อมูลเต็มพื้นที่ดิสก์ที่มีอยู่ และทำให้เกิดปัญหากับระบบได้ ซึ่งคำสั่งที่ใช้สำหรับการเปลี่ยนไดเรคตอรี่ คือ

```
chroot("/var/daemon")
or die "Couldn't chroot to /var/daemon: $!";
```

5. Close Standard File Descriptors

ขั้นตอนสุดท้ายในการสร้าง daemon ก็คือการปิด standard file descriptors (STDIN, STDOUT, STDERR) เนื่องจาก daemon ไม่ได้ใช้ terminal ดังนั้นจึงควรปิด file

descriptors แต่ส่วนใหญ่แล้วแทนที่จะปิด file descriptors เหล่านี้ ซึ่งบางครั้งอาจจะมีการใช้งานอยู่ ก็จะใช้วิธีการย้ายการทำงานไปยัง /dev/null แทน โดยที่ STDIN ก็กำหนดให้ไปอ่านข้อมูลจาก /dev/null ส่วน STDOUT และ STDERR ก็ให้ย้ายไปเขียนที่ /dev/null คำสั่งที่ใช้คือ

```
open STDIN,  '/dev/null' or die "Can't read /dev/null: $!";
open STDOUT, '>/dev/null';
open STDERR, '>/dev/null';
```

5.6 การพัฒนาโปรแกรมส่วน Analysis Module

การพัฒนาโปรแกรมส่วนของ Analysis Module จะมีฟังก์ชันการทำงาน ดังนี้

- ReadLogFile เป็นส่วนของ Read Logfile Process ซึ่งทำหน้าที่คอยอ่านข้อมูลจาก ล็อกไฟล์โดยอ่านข้อมูลเฉพาะส่วนที่มีการบันทึกเพิ่มเติมและส่งข้อมูลล็อกต่อไปยังฟังก์ชัน Prefiltering
- Prefiltering เป็นส่วนของ Process Prefiltering Process ซึ่งทำหน้าที่กรองเอาข้อมูลล็อกของเหตุการณ์ปกติที่ไม่เกี่ยวข้องกับการรักษาความปลอดภัยออกไปและส่งข้อมูลต่อไปยังฟังก์ชัน MisuseDetection
- MisuseDetection เป็นส่วนของ Misuse Detection Engine ซึ่งทำหน้าที่ในการบ่งชี้ว่าเหตุการณ์ความผิดปกติที่ส่งมานั้นเป็นการบุกรุกรูปแบบใด

รายละเอียดของการพัฒนาโปรแกรมทั้ง 3 ส่วนนี้จะได้กล่าวในหัวข้อ 5.6.1, 5.6.2 และ 5.6.3

5.6.1 ฟังก์ชันอ่านข้อมูลจากล็อกไฟล์ (ReadLogFile)

ฟังก์ชัน ReadLogFile ทำหน้าที่คอยอ่านข้อมูลล็อกจากล็อกไฟล์นั้นจะอาศัยการทำงานของโปรแกรม tail เป็นโปรแกรมมาตราฐานที่มีใช้งานอยู่บนระบบปฏิบัติการยูนิกซ์ โดยโปรแกรมส่วนนี้จะที่ทำหน้าที่อ่านข้อมูลจากล็อกไฟล์ที่กำหนดแล้วส่งผลลัพธ์ต่อไปยังโปรแกรมส่วนทำหน้าที่กรองข้อมูลล็อกโดยใช้โปรแกรม pipe สำหรับการใช้งาน tail นั้นจะระบุตัวเลือกการทำงานเป็น tail –f ซึ่งหมายความว่าให้อ่านเฉพาะข้อมูลส่วนที่มีการบันทึกเพิ่มเติมเท่านั้น

5.6.2 ฟังก์ชันกรองข้อมูลล็อก (Prefiltering)

สำหรับโปรแกรมส่วนของ Process Prefiltering Process ซึ่งทำหน้าที่กรองเอาข้อมูลล็อกของเหตุการณ์ปกติที่ไม่เกี่ยวข้องกับการรักษาความปลอดภัยออกไปนั้นจะมีการเปรียบเทียบรูปแบบของข้อมูลล็อกกับกฎที่กำหนดไว้ว่าตรงกันหรือไม่ ตามที่ได้แสดงในภาพประกอบที่ 4.4 ซึ่งกฎเหล่านี้จะเขียนอยู่ในรูปแบบของ Regular Expression ซึ่งเป็นการเขียน

อธิบายถึงรูปแบบข้อความที่ต้องการค้นหา ข้อมูลกฎต่าง ๆที่จะใช้ในการทำงานเก็บอยู่ในไฟล์ที่มีรูปแบบเป็นไฟล์ข้อความ (text file) โดยเมื่อโปรแกรมเริ่มต้นการทำงานก็จะอ่านข้อมูลจากไฟล์มาเก็บไว้ในหน่วยความจำ และเก็บข้อมูลเหล่านี้ในโครงสร้างข้อมูลแบบอาร์เรย์ (array) ซึ่งโครงสร้างข้อมูลแบบอาร์เรย์ที่ใช้สำหรับการพัฒนาโปรแกรมส่วนนี้ ในภาษา Perl จัดเป็นอาร์เรย์แบบ associative array ซึ่งโปรแกรมส่วนนี้มีการอ่านข้อมูลกฎการกรองต่าง ๆ จากไฟล์กฎของเหตุการณ์ปกติ (Normal Event Rules) มาเก็บไว้ในหน่วยความจำ โดยจัดเก็บไว้ในตัวแปรอาร์เรย์ชื่อ filter_rules ดังภาพประกอบที่ 5.2

กฎข้อที่ 1	Regular Expressions สำหรับกฎข้อที่ 1
...	...
กฎข้อที่ n	Regular Expressions สำหรับกฎข้อที่ n

ภาพประกอบ 5.2 การเก็บข้อมูลในตัวแปรอาร์เรย์ filter_rules

ดังนั้นในการเขียนกฎต้องเขียนบรรทัดละ 1 กฎเท่านั้น ตัวอย่าง เช่น

```
sendmail\[\d+\]: .*.to=
sendmail\[\d+\]: .*.from=
```

เป็นกฎสองข้อ ที่ใช้อธิบายเหตุการณ์ปกติในการส่งและรับเมล์ของโปรแกรม sendmail และเมื่อกฎทั้งสองถูกอ่านมาเก็บไว้ที่ตัวแปรอาร์เรย์ filter_rules สามารถแสดงได้ภาพประกอบที่ 5.3

sendmail\[\d+\]: .*.to=
sendmail\[\d+\]: .*.from=

ภาพประกอบ 5.3 ตัวอย่างการเก็บข้อมูลในตัวแปรอาร์เรย์ filter_rules

ข้อมูลกฎเหล่านี้เมื่ออ่านมาเก็บไว้ในตัวแปร filter_rules แล้วก็จะเป็นข้อมูลคงที่ (static) นั่นคือจำนวนข้อมูลและข้อมูลต่าง ๆที่เก็บจะไม่มีการเปลี่ยนแปลงในระหว่างการทำงาน ส่วนรายละเอียดของกฎทั้งหมดดูได้จาก ภาคผนวก

5.6.3 ฟังก์ชันตรวจจับการบุกรุก (MisuseDetection)

โปรแกรมส่วนของ Misuse Detection Engine ซึ่งทำหน้าที่ในการบ่งชี้ว่าเหตุการณ์ความผิดปกติที่ส่งมานั้นเป็นการบุกรุกรูปแบบใดส่งข้อมูลการแจ้งเตือนไปยัง Response Module นั้นนั้นก็จะมีการอ่านข้อมูลกฎที่เก็บอยู่ในไฟล์ที่เรียกว่า Intrusion Signatures and Response Policy (สำหรับรายละเอียดของกฎได้กล่าวไว้ในหัวข้อที่ 4.3.2.3) มาเก็บไว้ในหน่วยความจำ ซึ่งจะมีตัวแปรอาร์เรย์ต่าง ๆ สำหรับเก็บข้อมูลในแต่ละส่วนของกฎ ซึ่งมีรายละเอียดชื่อตัวแปรอาร์เรย์และข้อมูลที่เก็บดังนี้

1 regexp เก็บ Regular Expressions ที่อธิบายถึงรูปแบบร่องรอยล็อกของเหตุการณ์ที่เป็นการบุกรุก เหตุการณ์ความผิดปกติ และเหตุการณ์ที่ต้องการตรวจจับ

2 info เก็บคำอธิบายของเหตุการณ์ที่ต้องการตรวจจับ และถ้าหากในส่วนของ regexp ในข้อ 1 ได้กำหนด Regular Expression ในลักษณะที่มีการใช้วงเล็บ ก็จะมีการสร้างตัวแปรพิเศษ $1, $2, ... $n (n คือจำนวนวงเล็บที่ใช้) ซึ่งแทนข้อมูลในวงเล็บแต่ละตัวเหล่านั้น ก็จะสามารถนำตัวแปรพิเศษเหล่านั้นมาใช้ในการระบุคำอธิบายนี้ได้ และตัวแปรพิเศษ $0 จะหมายถึงข้อมูลล็อกของทั้งบรรทัดที่ได้อ่านเข้ามา และ $1 ก็จะแทนข้อมูลที่อยู่ในวงเล็บตัวที่ 1

3 corr เก็บคำอธิบาย คำแนะนำวิธีการแก้ไขปัญหาของเหตุการณ์ที่ตรวจจับ

4 action เก็บ Alert Action List คือ ข้อมูลวิธีการในการแจ้งเตือน เมื่อตรวจจับเหตุการณ์ตามรูปแบบที่กำหนดในข้อ 2 สำหรับการแจ้งเตือนนั้นสามารถระบุได้หลายวิธีการ โดยใช้เครื่องหมายอัฒภาค (;) คั่นข้อมูลในแต่ละวิธีการแจ้งเตือนและสามารถใช้ตัวแปรพิเศษได้เช่นเดียวกับกรณีของตัวแปร info ในข้อ 2 ในกรณีที่การกำหนด Regular Expression มีการใช้วงเล็บ ซึ่งตัวแปรพิเศษเหล่านี้ก็จะสามารถนำมาใช้เป็นพารามิเตอร์ในการแจ้งเตือนได้ด้วย นอกจากนี้ยังสามารถใช้ตัวแปรพิเศษเหล่านี้เพิ่มเติมซึ่งเป็นตัวแปรพิเศษที่ใช้แทนข้อมูลในแต่ละส่วนของกฎที่ใช้ในการตรวจจับ ซึ่งมีตัวแปรพิเศษดังนี้

- %1 แทนข้อมูลส่วนของ regexp
- %2 แทนข้อมูลส่วนของ info
- %3 แทนข้อมูลส่วนของ corr
- %4 แทนข้อมูลส่วนของ action
- %5 แทนข้อมูลส่วนของ throttle
- %6 แทนข้อมูลส่วนของ threshold
- %7 แทนข้อมูลส่วนของ window
- %t แทนข้อมูลวันที่และเวลาปัจจุบันของระบบ

5 throttle เก็บช่วงเวลาที่ใช้หน่วงสำหรับการแจ้งเตือน มีหน่วยเป็นวินาที

6 threshold เก็บค่าขอบเขตของความถี่ของเหตุการณ์สำหรับการแจ้งเตือน

7 window เก็บช่วงเวลา (time window) ที่ใช้การนับจำนวนเหตุการณ์ (วินาที)

กฎข้อที่ 1	Regexp[1]	info[1]	Corr[1]	action[1]	throttle[1]	threshold[1]	window[1]
...	...	...	...	...	...	...	...
กฎข้อที่ n	Regexp[n]	info[n]	Corr[n]	action[n]	throttle[n]	threshold[n]	window[n]

ภาพประกอบ 5.4 ลักษณะการเก็บข้อมูลกฎการตรวจจับในตัวแปรอาร์เรย์ต่าง ๆ

จากภาพประกอบที่ 5.4 ข้อมูลกฎเหล่านี้เมื่ออ่านมาเก็บไว้ในตัวแปรอาร์เรย์ต่าง ๆ แล้วก็จะเป็นข้อมูลคงที่ (static) นั่นคือจำนวนข้อมูลและข้อมูลต่าง ๆที่เก็บจะไม่มีการเปลี่ยนแปลงในระหว่างการทำงานเช่นเดียวตัวแปรอาร์เรย์ filter_rules หลังจากนั้นเมื่อโปรแกรมแล้วทำการเปรียบเทียบรูปแบบข้อมูลล็อกที่ส่งมาจากส่วน Prefiltering กับรูปแบบร่องรอยการบุกรุกของกฎแต่ละข้อ และหากพบว่าข้อมูลล็อกนั้นตรงกับกฎข้อใด ก็จะเก็บข้อมูลเหตุการณ์นั้นไว้ในหน่วยความจำในตัวแปรอาร์เรย์ชื่อ events_list ซึ่งประกอบด้วยข้อมูลดังนี้

1 window เก็บช่วงเวลา(time window) ที่ใช้ในการนับจำนวนเหตุการณ์ มีหน่วยเป็นวินาที

2 event_times เก็บอาร์เรย์ของข้อมูลเวลาที่เกิดเหตุการณ์ในแต่ละครั้ง ข้อมูลเวลาที่เก็บนี้เป็นจำนวนวินาทีตั้งแต่ เวลา 00:00:00 UTC, January 1, 1970 ทั้งนี้นำข้อมูลเหล่านี้ไปใช้สำหรับการทำงานในการทำงานในลักษณะเลื่อนช่วงเวลา (Sliding Window)

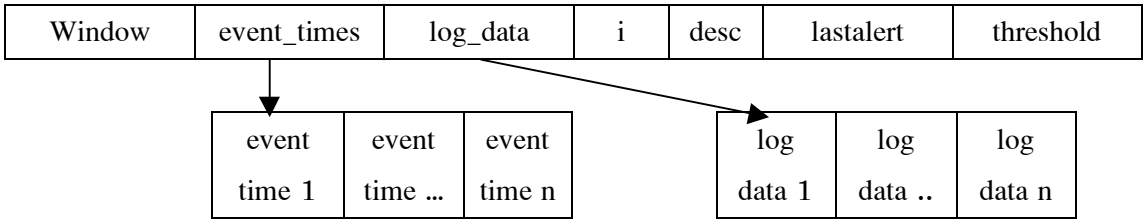
3 log_data เก็บอาร์เรย์ของข้อมูลล็อกของแต่ละเหตุการณ์ที่เกิดขึ้น

4 i เก็บหมายเลขของกฎ

5 desc เก็บคำอธิบายของเหตุการณ์ที่ต้องการตรวจจับ

6 lastalert เก็บเวลาที่ทำการแจ้งเตือนครั้งสุดท้าย

7 threshold เก็บค่าขอบเขตของความถี่ของเหตุการณ์สำหรับการแจ้งเตือน



ภาพประกอบ 5.5 ลักษณะการเก็บข้อมูลเหตุการณ์ในตัวแปรอาร์เรย์ events_list

จากภาพประกอบที่ 5.5 ข้อมูลเหตุการณ์ที่เก็บอยู่ในตัวแปร events_list เป็นข้อมูลแบบ dynamic ซึ่งจะมีการสร้างขึ้นเมื่อมีข้อมูลล็อกของเหตุการณ์ที่ตรงกับรูปแบบที่ได้ระบุไว้ในกฎเท่านั้น และจะมีการปรับปรุงและเปลี่ยนแปลงข้อมูลในทุกครั้งที่เกิดเหตุการณ์ดังกล่าวอีก ซึ่งค่า key หรือ index ที่ใช้สำหรับการเข้าถึงข้อมูลของแต่ละเหตุการณ์นั้น จะใช้ข้อมูล desc หรือคำอธิบายของเหตุการณ์ที่ต้องการตรวจจับที่ผ่านการแทนค่าของตัวแปรพิเศษแล้ว ดังนั้นจึงทำให้กฎสำหรับการตรวจจับแต่ละข้อ สามารถใช้ตรวจจับเหตุการณ์ที่สนใจได้หลายรูปแบบได้ ตัวอย่าง เช่น กฎการตรวจจับ ที่เขียนไว้ดังนี้

authentication failure.* rhost=(\S+) user=(\S+)::User $2 too many login failure from $1::*** Possible brute force password cracking ***::email=ssupacho@ratree.psu.ac.th ::300::3::60

กำหนดให้กฎนี้เป็นกฎข้อที่ 1 ดังนั้นข้อมูลของตัวแปรอาร์เรย์ต่าง ๆก็จะเป็นดังนี้

regexp[1] = authentication failure.* rhost=(\S+) user=(\S+)
info[1] = User $2 too many login failure from $1
corr[1] = *** Possible brute force password cracking ***
action[1] = email=ssupacho@ratree.psu.ac.th
throttle[1] = 300
threshold[1] = 3
window[1] = 60

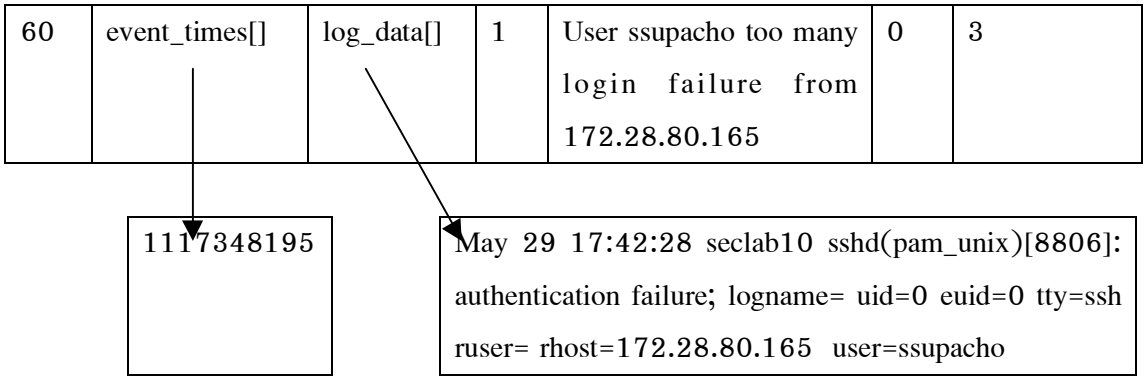
กฎข้อนี้ใช้สำหรับตรวจจับเหตุการณ์การเข้าระบบที่ไม่สำเร็จของผู้ใช้แต่ละคนจากแต่ละเครื่องที่มีหมายเลข IP ต่างกัน โดยถ้าหากตรวจจับพบว่ามีการ login ที่ไม่สำเร็จ 3 ครั้งในช่วงระยะเวลา 60 วินาทีก็จะทำการแจ้งเตือนโดยการส่งอีเมล์ไปยัง ssupacho@ratree.psu.ac.th โดยจะหน่วงเวลาสำหรับการแจ้งเตือนไว้ 300 วินาที หากพบเหตุการณ์ลักษณะเดียวกันนี้อีก หลังจากนั้นจึงทำการแจ้งเตือนไปใหม่อีกครั้ง ดังนั้นเมื่อโปรแกรมพบว่ามีข้อมูลล็อกเข้ามาเป็นดังนี้

May 29 17:42:28 seclab10 sshd(pam_unix)[8806]: authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.28.80.165 user=ssupacho

และเมื่อตรวจสอบข้อมูลล็อกพบว่าตรงกับข้อมูลกฎที่ได้กำหนดไว้ ดังนั้นโปรแกรมก็จะเพิ่มข้อมูลของเหตุการณ์นี้ในตัวแปรอาร์เรย์ event_list พร้อมทั้งแทนค่าตัวแปรพิเศษ $1 และ $2 ซึ่งจะมีค่าเป็น 172.28.80.165 และ ssupacho ตามลำดับ ดังนั้นข้อมูลส่วนของ desc ซึ่งเป็นคำอธิบายและเป็นข้อมูลที่ใช้เป็น key สำหรับการเข้าข้อมูลในตัวแปรอาร์เรย์ event_list ก็จะมีค่าเป็นดังนี้

User ssupacho too many login failure from 172.28.80.165

ซึ่งข้อมูลต่าง ๆ ของตัวแปรอาร์เรย์ event_list แสดงได้ดังภาพประกอบที่ 5.6



ภาพประกอบ 5.6 ตัวอย่างการเก็บข้อมูลเหตุการณ์ในตัวแปรอาร์เรย์ events_list (1)

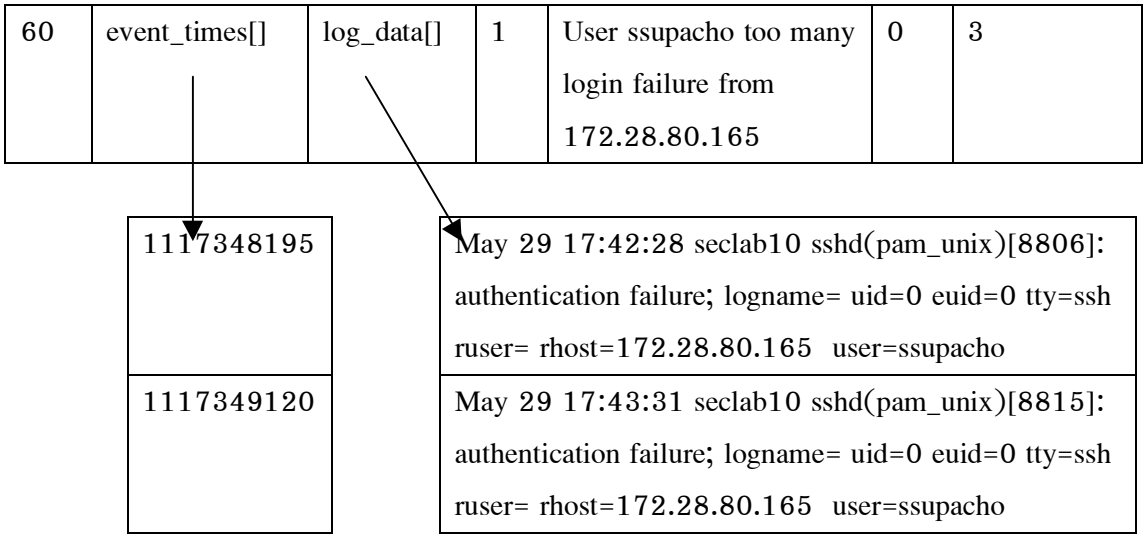
หลังจากนั้นหากโปรแกรมพบว่าข้อมูลล็อกที่เข้ามามีรูปแบบตรงกับกฎข้อนี้อีก ก็จะตรวจสอบค่า key ของเหตุการณ์ใหม่นั้นว่ามีอยู่ในตัวแปรอาร์เรย์ event_list แล้วหรือไม่ ถ้าหากพบว่ามีข้อมูลเหตุการณ์นั้นอยู่แล้วก็จะทำการเพิ่มข้อมูลเวลาที่เกิดเหตุการณ์และข้อมูลล็อกเข้าไปในอาร์เรย์ event_times[] และ log_data[] ตามลำดับเท่านั้น แต่หากพบค่า key ก็จะเพิ่มข้อมูลของเหตุการณ์ดังกล่าวในอาร์เรย์ even_list นั่นคือหากมีข้อมูลล็อกเข้ามาเป็นดังนี้

May 29 17:43:31 seclab10 sshd(pam_unix)[8815]: authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.28.80.165 user=ssupacho
May 29 17:45:12 seclab10 sshd(pam_unix)[8857]: authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=172.28.80.169 user=ssupacho

สำหรับข้อมูลล็อกของเหตุการณ์แรกมีค่า key เป็นดังนี้

User ssupacho too many login failure from 172.28.80.165

ซึ่งเมื่อตรวจสอบค่า key นี้กับข้อมูลในอาร์เรย์ event_list พบว่ามีข้อมูลนี้อยู่แล้ว ดังนั้นโปรแกรมจึงเพิ่มเฉพาะข้อมูลเวลาที่เกิดเหตุการณ์และล็อกของเหตุการณ์ในอาร์เรย์ event_times[] และ log_data[] ซึ่งแสดงได้ดังภาพประกอบที่ 5.7

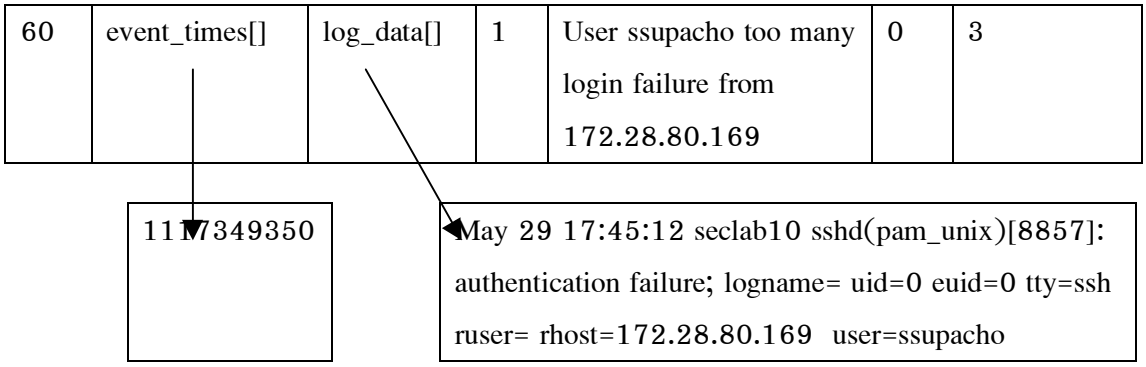


ภาพประกอบ 5.7 ตัวอย่างการเก็บข้อมูลเหตุการณ์ในตัวแปรอาร์เรย์ events_list (2)

ส่วนข้อมูลล็อกเหตุการณ์ที่สองนั้นมีค่า key เป็นดังนี้

User ssupacho too many login failure from 172.28.80.169

ซึ่งข้อมูลต่าง ๆ ของตัวแปรอาร์เรย์ event_list ก็จะเป็นดังภาพประกอบที่ 5.8



ภาพประกอบ 5.8 ตัวอย่างการเก็บข้อมูลเหตุการณ์ในตัวแปรอาร์เรย์ events_list (3)

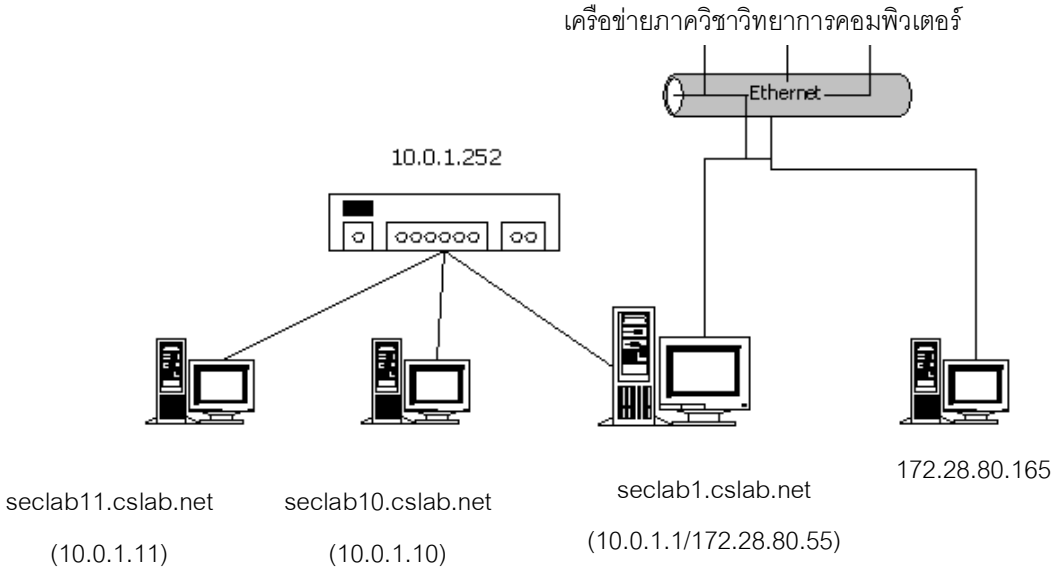
5.7 การพัฒนาโปรแกรมส่วน Response Module

โปรแกรมส่วนนี้ประกอบด้วยฟังก์ชัน ResponseAlert ซึ่งรับข้อมูลวิธีการในการแจ้งเตือนที่เก็บอยู่ในตัวแปรอาร์เรย์ชื่อ action แล้วจึงทำการแจ้งเตือนตามวิธีการที่ได้รับมา สำหรับวิธีการแจ้งเตือนนั้นมีทั้งหมด 3 วิธีการ คือ email, write และ exec ซึ่งรายละเอียดในการพัฒนาแต่ละวิธี มีดังต่อไปนี้

1. email การแจ้งเตือนโดยการส่งอีเมล์ตามที่อยู่ที่ได้ระบุไว้ในกฎ โดยในการส่งเมล์นั้นจะอาศัยโปรแกรม sendmail ในการทำหน้าที่ส่งอีเมล์ไปยังที่อยู่ตามที่กำหนด
2. write การบันทึกข้อมูลลงในไฟล์ที่ต้องการ โดยในการบันทึกข้อมูลลงในไฟล์จะเป็นการบันทึกข้อมูลในลักษณะเพิ่มข้อมูล (append) ในกรณีที่ไฟล์ที่ระบุมีอยู่แล้ว แต่หากยังไม่มีไฟล์ก็จะสร้างไฟล์ใหม่ขึ้นมาและบันทึกข้อมูลลงไปในไฟล์
3. exec การสั่งให้โปรแกรมที่ต้องการทำงาน โปรแกรมที่พัฒนาส่วนนี้จะเริ่มต้นจากการสร้างโปรเซสใหม่ขึ้นมาโดยใช้ฟังก์ชัน fork แล้วส่งคำสั่ง/โปรแกรมรับมาทำงานในโปรเซสดังกล่าวโดยการเรียกใช้ฟังก์ชัน system

5.8 การสร้างกฎเพื่อใช้สำหรับการทำงานของโปรแกรม

การสร้างกฎต่าง ๆที่ใช้สำหรับการทำงานโปรแกรมซึ่งมี 2 ส่วนคือ กฎของเหตุการณ์ปกติ (Normal Event Rules) และกฎของการตรวจจับการบุกรุก (Intrusion Signature and Response Policy) โดยกฎทั้ง 2 ส่วนสร้างโดยอาศัยข้อมูลที่ได้จากการทดสอบในเครือข่ายที่จัดทำขึ้นซึ่งประกอบด้วยเครื่องคอมพิวเตอร์ทั้งหมดจำนวน 4 เครื่อง และเครือข่ายทดสอบนี้เชื่อมต่อกับเครือข่ายของภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ ซึ่งแสดงได้ดังภาพประกอบ 5.9



ภาพประกอบ 5.9 เครือข่ายที่สร้างขึ้นเพื่อทดสอบการบุกรุก

จากภาพประกอบที่ 5.9 รายละเอียดของเครื่องคอมพิวเตอร์มีดังนี้

1. เครื่อง seclab1.cslab.net ทำหน้าที่เป็น Gateway ในการเชื่อมต่อระหว่างเครือข่ายที่จัดทำขึ้นกับเครือข่ายของภาควิชาวิทยาการคอมพิวเตอร์ ใช้ระบบปฏิบัติการ Linux RedHat 7.2
2. เครื่อง seclab10.cslab.net เป็นเครื่องเป้าหมายของการบุกรุก ใช้ระบบปฏิบัติการ Linux RedHat 7.2 โดยมีการติดตั้งโปรแกรม Sendmail 8.11.6-3 เพื่อทำหน้าที่เป็นเมล์เซิร์ฟเวอร์ และระบบตรวจจับการบุกรุก
3. เครื่อง seclab11.cslab.net เป็นเครื่องที่ใช้สำหรับการบุกรุก ใช้ระบบปฏิบัติการ Linux RedHat 7.2 และติดตั้งโปรแกรม Sendmail 8.11.6-3 ทำหน้าที่เป็นเมล์เซิร์ฟเวอร์
4. เครื่อง 172.28.80.165 เป็นเครื่องที่ใช้สำหรับการบุกรุก ใช้ระบบปฏิบัติการ Windows 98

5.8.1 การสร้างกฎของเหตุการณ์ปกติ

สำหรับวิธีในการสร้างกฎของเหตุการณ์ปกตินั้นทำโดยการจำลองการทำงานของเหตุการณ์ปกติในรูปแบบต่าง ๆบนเครื่องที่อยู่ในเครือข่ายที่จัดทำขึ้นสำหรับการทดสอบ แล้วเก็บรวบรวมข้อมูลล็อกที่เกิดขึ้นของแต่ละเหตุการณ์และนำมาเขียนเป็นกฎ โดยกฎแต่ละข้อใช้อธิบายถึงรูปแบบข้อมูลล็อคที่เกิดขึ้นของเหตุการณ์หนึ่งเหตุการณ์ ซึ่งกฎต่าง ๆ ที่ได้จากเหตุการณ์ปกติของโปรแกรมต่าง ๆ เป็นดังนี้

- โปรแกรม sendmail

```
sendmail\[\d+\]: .*.to=
sendmail\[\d+\]: .*.from=
(sendmail\[\d+\]: .*.relay=*.) && (!.*.reject.*.)
sendmail\[\d+\]: .*.: clone .*., owner=
sendmail\[\d+\]: NOQUEUE:*
sendmail\[\d+\]: .*.User unknown
sendmail\[\d+\]: .*.DSN: Service unavailable
sendmail\[\d+\]: .*.DSN: Host unknown
sendmail\[\d+\]: .*.DSN: Return receipt
sendmail\[\d+\]: .*.could not send message for past \d+ hours
sendmail\[\d+\]: .*.Domain of sender address .*. does not resolve
sendmail\[\d+\]: .*.Domain of sender address .*. does not exist
sendmail\[\d+\]: .*.Relaying temporarily denied
```

sendmail\[\d+\]: .*.Relaying denied

sendmail\[\d+\]: .*.timeout waiting for input from

sendmail\[\d+\]: .*.collect: premature EOM: Error 0

sendmail\[\d+\]: .*.collect: unexpected close on connection from

sendmail\[\d+\]: .*.lost input channel from

sendmail\[\d+\]: .*.forward .*. Group writable directory

sendmail\[\d+\]: .*.sender notify: Service unavailable

sendmail\[\d+\]: .*.return to sender: Service unavailable

sendmail\[\d+\]: .*.runqueue: Flushing queue from

sendmail\[\d+\]: .*.aliases rebuilt by root

sendmail\[\d+\]: aliases.*.longest

- โปรแกรม Mailscanner

update.virus.scanners

MailScanner\[\d+\]: .*.Uninfected

MailScanner\[\d+\]: .*.Virus and Content Scanning

MailScanner\[\d+\]: .*.New Batch

MailScanner\[\d+\]: .*.Using locktype

MailScanner\[\d+\]: .*.MailScanner .*. starting...

MailScanner\[\d+\]: .*.New Batch

MailScanner\[\d+\]: .*.RBL Check

MailScanner\[\d+\]: .*.SpamAssassin

MailScanner\[\d+\]: .*.Virus and Content Scanning: Starting

MailScanner\[\d+\]: .*.Virus Scanning

MailScanner\[\d+\]: .*.Saved infected

MailScanner\[\d+\]: .*.Cleaned: Delivered \d+ cleaned messages

MailScanner\[\d+\]: .*.Sender Warnings: Delivered \d+ warnings to virus senders

MailScanner\[\d+\]: .*.Notices: Warned about \d+ messages

MailScanner\[\d+\]: .*.Disinfection:* Attempting to disinfect \d+ messages

MailScanner\[\d+\]: .*.Virus Re-scanning: Sophos found \d+ infections

MailScanner\[\d+\]: .*.Disinfection: Rescan found only \d+ viruses

MailScanner\[\d+\]: .*.MailScanner child dying of old age

- โปรแกรม secure shell / login /su

  sshd(pam_unix)\[\d+\]: .*.session opened for user .*. by (uid=0)

  sshd(pam_unix)\[\d+\]: .*.session closed for user .*.

  su(pam_unix)\[\d+\]: .*.session opened for user .*. by .*.

  su(pam_unix)\[\d+\]: .*.session closed for user .*.

  login(pam_unix)\[\d+\]: .*.session opened for user .*. by .*.

  login(pam_unix)\[\d+\]: .*.session closed for user .*.

- โปรแกรม cron

  CRON\[\d+\]: \(root\) .*.

  CROND\[\d+\]: \(mailman\) .*.

5.8.2 การสร้างกฎของการตรวจจับการบุกรุก

การสร้างกฎสำหรับการตรวจจับการบุกรุกต้องอาศัยข้อมูลล็อกที่เป็นร่องรอยที่เกิดขึ้นจากการบุกรุกซึ่งในการเก็บร่องรอยการบุกรุกนั้นได้จากการทดสอบบุกรุกด้วยการโจมตีด้วยวิธีการต่าง ๆ ในเครือข่ายที่ใช้สำหรับการทดสอบ โดยสามารถแบ่งตามรูปแบบการโจมตีได้เป็นแบบ Active attack และ Passive attack

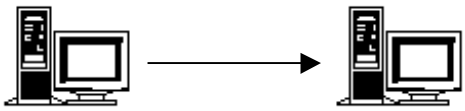
5.8.2.1 การเก็บร่องรอยการบุกรุกจากการโจมตีแบบ Active Attack

สำหรับวิธีการที่ใช้โจมตีแบบ Active Attack เพื่อเก็บร่องรอยการบุกรุก จะใช้วิธีการบุกรุกในรูปแบบต่าง ๆ คือการโจมตีโดยการใช้โปรแกรมไวรัส เวิร์ม โทรจัน DoS (Denial of Service และ Brute-force attack ซึ่งรายละเอียดการทดสอบเป็นดังนี้

- ไวรัส

  เครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการยูนิกซ์จะไม่ได้รับผลกระทบจากการโจมตีของไวรัสโดยตรง แต่ถ้าหากเครื่องดังกล่าวทำหน้าที่บริการเมล์ ซึ่งปัจจุบันไฟล์หรือโปรแกรมไวรัสมักจะถูกแนบมากับเมล์ที่ส่งไปยังผู้ใช้ต่าง ๆ ทำให้เครื่องคอมพิวเตอร์เหล่านี้ถูกใช้ให้เป็นตัวกลางในการแพร่กระจายของไวรัสไปยังเครื่องของผู้ใช้ที่ใช้ระบบปฏิบัติการ Windows ดังนั้นในการตรวจจับและป้องกันการแพร่กระจายของไวรัสที่ส่งมากับเมล์ จึงจะต้องทำการติดตั้งระบบการค้นหาและสกัดกั้นไม่ให้เมล์ที่มี

ไฟล์หรือโปรแกรมไวรัสถูกส่งไปยังผู้ใช้อื่น ๆ ซึ่งโปรแกรมที่ทำหน้าที่ให้บริการรับส่งเมล์ที่ใช้งานอยู่คือ sendmail ไม่มีความสามารถดังกล่าว ดังนั้นในการทดสอบจึงได้ติดตั้งโปรแกรมที่ทำหน้าที่ในการแยกไฟล์ต่าง ๆที่มีการแนบมากับอีเมล์เพื่อส่งไปให้โปรแกรมค้นหาไวรัสทำการค้นหาไวรัสในไฟล์ดังกล่าวแล้วรายงานผลของการค้นหาลงในล็อกไฟล์รวมทั้งสกัดกั้นไฟล์ที่มีไวรัสไว้เพื่อป้องกันการแพร่กระจายของไวรัส โดยในการทดสอบนี้จะใช้โปรแกรม MailScanner ซึ่งเป็นโปรแกรมประเภท opensource ทำงานร่วมกับโปรแกรมค้นหาไวรัสชื่อ Sophos Anti-virus รุ่นทดลองใช้งาน 30 วัน ที่ทำงานบนระบบปฏิบัติการ Linux ซึ่งโปรแกรมทั้งสองนี้จะติดตั้งไว้ที่เครื่อง seclab10.cslab.net ซึ่งทำหน้าที่เป็นเมล์เซิร์ฟเวอร์ และเครื่องนี้จะเป็นเครื่องเป้าหมายของการโจมตี และในการโจมตีจะใช้เครื่อง seclab11.cslab.net ดังภาพประกอบ 5.10



ภาพประกอบ 5.10 แผนภาพการทดสอบการโจมตีด้วยไวรัส

สำหรับการโจมตีทำโดยการส่งเมล์พร้อมกับแนบไฟล์ไวรัสจากเครื่อง seclab11.cslab.net ไปยังบัญชีผู้ใช้บนเครื่อง seclab10.cslab.net และการส่งเมล์พร้อมกับแนบไฟล์ไวรัสระหว่างบัญชีผู้ใช้ภายในเครื่อง seclab10.cslab.net โดยไวรัสที่ใช้ทดสอบการโจมตีมีดังนี้

- CIH/Chernobyl
- Melissa
- W32/MyDoom-A
- W32/Sobig-F
- eicar

ซึ่งผลการโจมตีพบว่าข้อมูลล็อกที่เป็นร่องรอยที่เกิดขึ้นการโจมตีของไวรัสมีลักษณะดังนี้

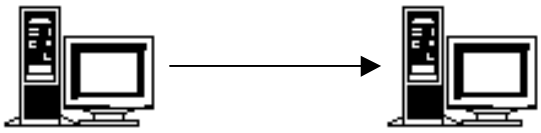
```
Jan 30 16:18:01 seclab10 sendmail[25546]: j0U9HoF25546: from=
<root@seclab10.cslab.net>, size=212643, class=0, nrcpts=1, msgid=
<Pine.LNX.4.33.0501301611420.25448-208000@seclab10.cslab.net>,
proto=ESMTP, daemon=MTA, relay=seclab10.cslab.net [127.0.0.1]
Jan 30 16:18:03 seclab10 MailScanner[29120]: New Batch: Scanning 1
messages, 213115 bytes
Jan 30 16:18:16 seclab10 MailScanner[29120]: Virus and Content Scanning:
Starting
Jan 30 16:18:19 seclab10 MailScanner[29120]: >>> Virus 'W32/Sobig-F' found in
file ./j0U9HoF25546/Win32.Sobig.Fmm.zip/your_details.pif
Jan 30 16:18:19 seclab10 MailScanner[29120]: >>> Virus 'W32/MyDoom-A'
found in file ./j0U9HoF25546/W32-Mydoom1.zip/doc.cmd
Jan 30 16:18:19 seclab10 MailScanner[29120]: >>> Virus 'W32/MyDoom-A'
found in file ./j0U9HoF25546/W32-Mydoom1.zip
Jan 30 16:18:19 seclab10 MailScanner[29120]: >>> Virus 'WM97/Melissa-BE'
found in file ./j0U9HoF25546/W97M.Melissa.zip/melissa.doc
Jan 30 16:18:19 seclab10 MailScanner[29120]: >>> Virus 'W32/Bagle-A' found in
file ./j0U9HoF25546/I-Worm.Bagle.a.zip/I-Worm.Bagle.a
Jan 30 16:18:20 seclab10 MailScanner[29120]: >>> Virus 'EICAR-AV-Test' found
in file ./j0U9HoF25546/eicar.com
Jan 30 16:18:20 seclab10 MailScanner[29120]: >>> Virus 'W95/CIH-10xx' found
in file ./j0U9HoF25546/chernobl.zip/CIH_14.EXE
Jan 30 16:18:20 seclab10 MailScanner[29120]: Virus Scanning: Sophos found 7
infections
Jan 30 16:18:21 seclab10 MailScanner[29120]: Virus Scanning: Found 7 viruses
Jan 30 16:18:21 seclab10 MailScanner[29120]: Filename Checks: Windows/DOS
Executable (eicar.com)
Jan 30 16:18:21 seclab10 MailScanner[29120]: Other Checks: Found 1 problems
Jan 30 16:18:21 seclab10 MailScanner[29120]: Saved infected "eicar.com" to
/var/spool/MailScanner/quarantine/20050130/j0U9HoF25546
Jan 30 16:18:21 seclab10 MailScanner[29120]: Saved infected "I-
Worm.Bagle.a.zip" to
/var/spool/MailScanner/quarantine/20050130/j0U9HoF25546
Jan 30 16:18:21 seclab10 MailScanner[29120]: Saved infected "W32-
Mydoom1.zip" to /var/spool/MailScanner/quarantine/20050130/j0U9HoF25546
Jan 30 16:18:21 seclab10 MailScanner[29120]: Saved infected
"Win32.Sobig.Fmm.zip" to
/var/spool/MailScanner/quarantine/20050130/j0U9HoF25546
Jan 30 16:18:21 seclab10 MailScanner[29120]: Saved infected "chernobl.zip" to
/var/spool/MailScanner/quarantine/20050130/j0U9HoF25546
Jan 30 16:18:21 seclab10 MailScanner[29120]: Saved infected
"W97M.Melissa.zip" to
/var/spool/MailScanner/quarantine/20050130/j0U9HoF25546
Jan 30 16:18:21 seclab10 MailScanner[29120]: Cleaned: Delivered 1 cleaned
messages
```

ข้อมูลล็อกเหล่านี้จะถูกส่งผ่านค่า facility ของ mail และค่า priority ตั้งแต่ระดับ info เป็นต้นไป หลังจากได้ข้อมูลร่องรอยการบุกรุกแล้วก็สามารถนำมาเขียนเป็นกฎสำหรับการตรวจจับไวรัสได้ เช่น กำหนดกฎในการตรวจจับว่า หากต้องการตรวจพบไวรัสชนิดใดเกิน 3 ครั้งในระยะเวลา 5 นาทีก็ให้ส่งอีเมล์ยังบัญชีชื่อ ssupacho ก็เขียนกฎการตรวจจับได้ดังนี้

Virus (\S+) found in file::Found virus $1::Correlation-*** be careful ***::email=ssupacho::0::3::300

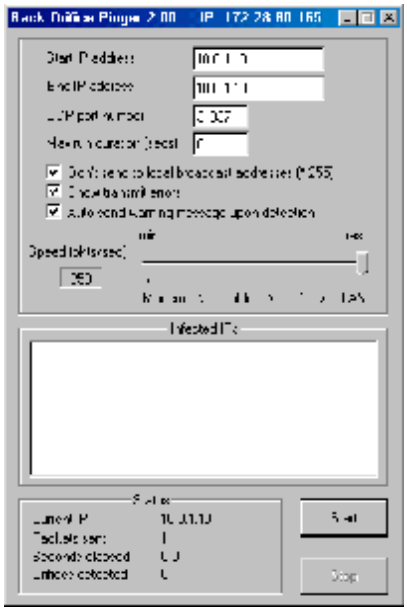
- **เวิร์มและโทรจัน**

สำหรับการเก็บร่องรอยการบุกรุกและทดสอบระบบโดยการโจมตีด้วยเวิร์มและโทรจันนั้น จะใช้วิธีการที่เหมือนกับการโจมตีด้วยไวรัส เนื่องจากปัจจุบันโปรแกรมประเภทเวิร์มและโทรจันใช้วิธีการแพร่กระจายผ่านการส่งเมล์พร้อมกับแนบไฟล์เวิร์มหรือโทรจันไปยังเป้าหมายซึ่งเหมือนกับกรณีของไวรัส ดังนั้นวิธีการเก็บร่องรอยและการตรวจจับจึงเหมือนกับกรณีของไวรัส นอกจากนี้ในกรณีของโทรจันนั้นหากเครื่องเป้าหมายของการโจมตีได้ถูกติดตั้งโปรแกรมม้าโทรจันแล้วก็อาจจะมีการเปิดช่องทางเพื่อใช้สำหรับการติดต่อเข้ามาของผู้บุกรุกในภายหลัง ดังนั้นในการตรวจจับก็สามารถทำได้โดยการตรวจสอบพอร์ตต่าง ๆที่เปิดใช้งานอยู่ว่ามีความผิดปกติไปจากเดิมหรือไม่ และในกรณีที่ผู้บุกรุกมีความพยายามติดต่อเข้ามายังเครื่องที่มีโปรแกรมโทรจันติดตั้งอยู่ ก็สามารถตรวจจับได้โดยอาศัยการตรวจสอบข้อมูลในแพ็กเก็ตที่ส่งมายังเครื่องคอมพิวเตอร์เป้ามายได้เช่นกัน ดังนั้นในการทดสอบการบุกรุกจึงติดตั้งโปรแกรม Snort ซึ่งเป็นโปรแกรมที่ใช้สำหรับตรวจจับการบุกรุกทางเครือข่าย (NIDS) เพิ่มเติมลงในเครื่องที่เป็นเป้าหมายของการโจมตี



ภาพประกอบ 5.11 แผนภาพการทดสอบการโจมตีด้วยโทรจัน

จากภาพประกอบ 5.11 เครื่องเป้าหมายของการโจมตีคือ seclab10.cslab.net ได้ติดตั้งโปรแกรม Snort เพิ่มเข้าไป และเครื่องที่ใช้โจมตีนั้นเป็นเครื่องที่อยู่ในเครือข่ายของภาควิชาวิทยาการคอมพิวเตอร์ หมายเลข IP เป็น 172.28.80.165 ใช้ระบบปฏิบัติการ Windows 98 และในการทดสอบการโจมตีจะใช้โปรแกรมชื่อ BOPing 2.00 ซึ่งโปรแกรมนี้ใช้สำหรับการค้นหาว่าเครื่องใดบ้างที่มีโปรแกรมม้าโทรจันชื่อ Back Orifice ทำงานอยู่ ซึ่งจอภาพโปรแกรม BOPing 2.00 แสดงได้ดังภาพประกอบ 5.12



ภาพประกอบ 5.12 จอภาพการทำงานของโปรแกรม BOPing 2.00

ซึ่งการทำงานทดสอบโจมตีโดยใช้โปรแกรมดังกล่าวพบว่าข้อมูลล็อกที่เป็นร่องรอยที่เกิดขึ้นการโจมตีมีลักษณะดังนี้

```
Jun 27 11:32:39 seclab10 snort: [105:1:1] spp_bo: Back Orifice Traffic detected
(key: 31337) {UDP} 172.28.80.165:4510 -> 10.0.1.10:31337
Jun 27 11:32:54 seclab10 snort: [105:1:1] spp_bo: Back Orifice Traffic detected
(key: 31337) {UDP} 172.28.80.165:4511 -> 10.0.1.10:2313
```

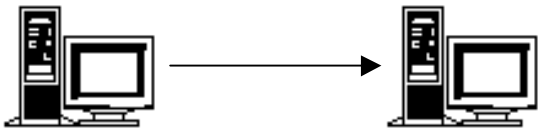
ข้อมูลล็อกเหล่านี้จะถูกส่งผ่านค่า facility ของ auth และค่า priority ตั้งแต่ระดับ alert เป็นต้นไป หลังจากได้ข้อมูลร่องรอยการบุกรุกแล้วก็สามารถนำมาเขียนเป็นกฎสำหรับการตรวจจับโทรจัน Back Orifice ได้ดังนี้

```
Back Orifice Traffic detected::Found trojan - Back Orifice::Correlation-*** plase
verify trojan in your system ***::email=ssupacho::0::0::0
```

- **Denial of Service (DoS)**

การโจมตีแบบ DoS เป็นการโจมตีที่จะพยายามทำให้งานที่ทำหรือบริการต่าง ๆที่ทำงานอยู่บนเครื่องเป้าหมายไม่สามารถทำงานได้โดยการส่งคำร้องขอ (requests) ไปยังเครื่องเป้าหมายเป็นจำนวนมากจนเกินความสามารถที่เครื่องนั้นจะให้บริการได้

จนหยุดการทำงานหรือไม่สามารถให้บริการได้ตามปกติ โดยการทดสอบโจมตีแบบนี้เพื่อเก็บร่องรอยและทดสอบระบบตรวจจับการบุกรุกที่พัฒนาขึ้นแสดงได้ดังภาพประกอบ 5.13



เครื่องที่ใช้โจมตี
172.28.80.165

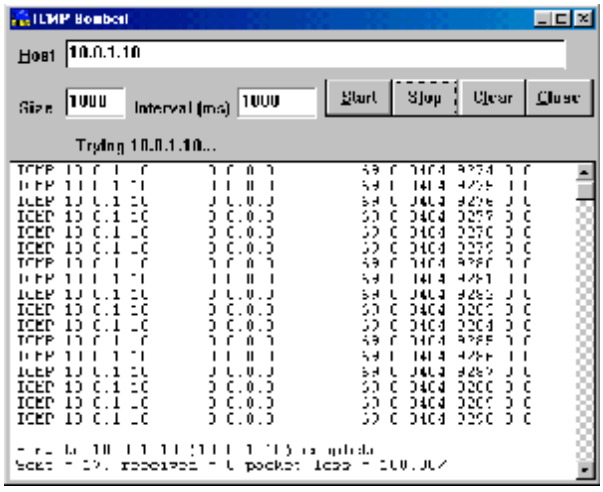
เครื่องเป้าหมาย
seclab10.cslab.net
(10.0.1.10)

ภาพประกอบ 5.13 แผนภาพการทดสอบการโจมตีแบบ DoS

การโจมตีแบบ DoS ส่วนใหญ่จะเป็นการโจมตีผ่านระบบเครือข่าย ดังนั้นในการทดสอบการบุกรุกจึงติดตั้งโปรแกรม Snort ซึ่งเป็นโปรแกรมที่ใช้สำหรับตรวจจับการบุกรุกทางเครือข่าย (NIDS) เพิ่มเติมลงในเครื่องที่เป็นเป้าหมายของการโจมตีด้วย สำหรับรายละเอียดวิธีการที่ใช้โจมตีมี 3 วิธีดังนี้

Ping Flood

เป็นการโจมตีโดยการส่งแพ็กเก็ต ICMP Echo Request ซึ่งเป็นแพ็กเก็ตแบบเดียวกับที่ได้จากคำสั่ง Ping ไปยังเครื่องที่ต้องการโจมตีเป็นปริมาณมาก ๆ อย่างรวดเร็ว ซึ่งในการทดสอบครั้งนี้ใช้โปรแกรมชื่อ ICMP Bomber ซึ่งได้จากอินเทอร์เน็ต ซึ่งจอภาพการทำงานของโปรแกรมแสดงได้ดังภาพประกอบที่ 5.14



ภาพประกอบ 5.14 จอภาพการทำงานของโปรแกรม ICMP Bomber

โดยทำการโจมตีจากเครื่องหมายเลข IP 172.28.80.165 ไปยังเครื่องเป้าหมายคือ 10.0.1.10 ผลข้อมูลล็อกที่เป็นร่องรอยการบุกรุกที่เกิดขึ้นเป็นดังนี้

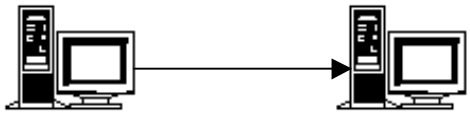
```
Jun 27 11:00:09 seclab10 snort: [1:499:3] ICMP Large ICMP Packet
[Classification: Potentially Bad Traffic] [Priority: 2]: {ICMP} 172.28.80.165 ->
10.0.1.10
Jun 27 11:01:00 seclab10 snort: [1:499:3] ICMP Large ICMP Packet
[Classification: Potentially Bad Traffic] [Priority: 2]: {ICMP} 172.28.80.165 ->
10.0.1.10
```

ข้อมูลล็อกเหล่านี้จะถูกส่งผ่านค่า facility ของ auth และค่า priority ตั้งแต่ระดับ alert เป็นต้นไป หลังจากได้ข้อมูลร่องรอยการบุกรุกแล้วก็สามารถนำมาเขียนเป็นกฎสำหรับการตรวจจับการโจมตี Ping Flood ได้ดังนี้

```
Large ICMP Packet::Found Ping Flood attack::Correlation-***  be careful
***::email=ssupacho::0::0::0
```

SYN Flood

การโจมตีวิธี SYN Flood ทำโดยการสร้างแพ็กเก็ต TCP ที่ตั้งค่า SYN เพื่อขอเริ่มการเชื่อมต่อพร้อมทั้งทำการปลอมหมายเลข IP ต้นทางให้เป็นของเครื่องที่ไม่มีอยู่จริงด้วย แล้วจึงส่งแพ็กเก็ตดังกล่าวในปริมาณมากไปยังเครื่องเป้าหมายตามรายละเอียดที่ได้กล่าวไว้ในหัวข้อที่ 2.3.8 โดยในการทดสอบได้ใช้โปรแกรมชื่อ synflood ที่เขียนด้วยภาษาซี โดยทำการโจมตีจากเครื่องหมายเลข IP 10.0.1.1 ไปยังเครื่อง 10.0.1.10 ดังภาพประกอบ 5.15



ภาพประกอบ 5.15 แผนภาพการทดสอบการโจมตีแบบ SYN Flood

โดยที่ในการโจมตีจะระบุพอร์ตที่จะโจมตีเป็นพอร์ตหมายเลข 80 และจำนวนที่ขอการเชื่อมต่อเป็น 1000 ครั้ง ซึ่งวิธีการเรียกใช้โปรแกรม synflood เป็นดังนี้

```
[root@seclab1 root]# ./synflood 10.0.1.1 10.0.1.10 80 1000
synflooding 10.0.1.10 from 10.0.1.1 port 80 1000 times
spoofing 129.62.13.81
spoofing 129.62.13.191
...
spoofing 129.62.13.27
spoofing 129.62.13.90
```

ซึ่งผลการทดสอบในตอนแรกไม่ปรากฏร่องรอยการบุกรุกใด ๆ เกิดขึ้นในล็อกไฟล์ของระบบ แต่จากการศึกษาพบว่าระบบปฏิบัติการ Linux มีกลไกในป้องกันและการตรวจจับการโจมตีด้วยวิธีนี้ แต่ปกติจะถูกกำหนดไว้ไม่ให้ทำงาน ซึ่งหากต้องการใช้งานกลไกดังกล่าวสามารถทำได้โดยการแก้ไขไฟล์ /etc/sysctl.conf โดยเพิ่มข้อมูล

```
net.ipv4.tcp_syncookies = 1
```

แล้วทำการโจมตีพบว่ามีข้อมูลล็อกที่เป็นร่องรอยของการโจมตีเกิดขึ้นเป็นดังนี้

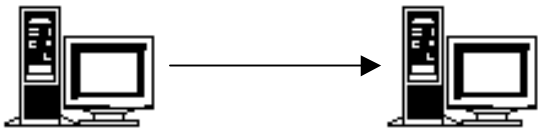
```
May  9 23:37:59 seclab11 kernel: possible SYN flooding on port 80. Sending
cookies.
```

ข้อมูลล็อกนี้จะส่งผ่านค่า facility ของ kernel และค่า priority ในระดับ info จากข้อมูลร่องรอยการบุกรุกนี้สามารถนำมาเขียนเป็นกฎในการตรวจจับได้ดังนี้

```
SYN Flood detected::possible SYN Flood attack::Correlation-***
SYN Flooding ***::email=ssupacho
```

- **Brute-force attack**

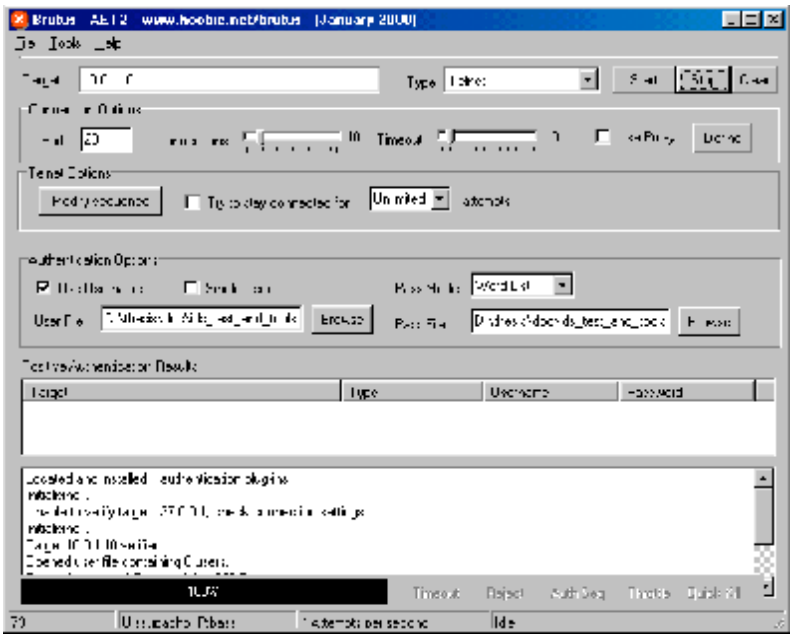
การโจมตีวิธีนี้เป็นการพยายามถอดรหัสข้อความที่มีการเข้ารหัสไว้ ซึ่งส่วนใหญ่ก็คือ รหัสผ่านของผู้ใช้ โดยการพยายามใช้ทุกค่าที่เป็นไปได้ ซึ่งในการทดสอบจะใช้วิธีการโจมตีผ่านทางพอร์ตของ Telnet ซึ่งหมายเลขพอร์ตคือ 23 โดยทำการโจมตีจากเครื่อง 172.28.80.165 ไปที่เครื่อง 10.0.1.10 ดังภาพประกอบที่ 5.16



เครื่องที่ใช้โจมตี 172.28.80.165 — เครื่องเป้าหมาย seclab10.cslab.net (10.0.1.10)

ภาพประกอบที่ 5.16 แผนภาพการโจมตีแบบ Brute-force attack

ในการโจมตีใช้โปรแกรม Brutus ในการทำงานซึ่งจอภาพของโปรแกรมนี้แสดงได้ภาพประกอบที่ 5.17



ภาพประกอบ 5.17 จอภาพการทำงานของโปรแกรม Brutus

โดยข้อมูลล็อกที่เป็นร่องรอยการบุกรุกที่เกิดขึ้นมีลักษณะดังนี้

Apr 6 14:01:49 seclab10 login(pam_unix)[5227]: authentication failure; logname= uid=0 euid=0 tty=pts/2 ruser= rhost=172.28.80.165 user=ssupacho
Apr 6 14:01:49 seclab10 login(pam_unix)[5228]: authentication failure; logname= uid=0 euid=0 tty=pts/3 ruser= rhost=172.28.80.165 user=ssupacho
Apr 6 14:01:49 seclab10 login(pam_unix)[5230]: authentication failure; logname= uid=0 euid=0 tty=pts/5 ruser= rhost=172.28.80.165 user=ssupacho
Apr 6 14:01:49 seclab10 login(pam_unix)[5229]: authentication failure; logname= uid=0 euid=0 tty=pts/4 ruser= rhost=172.28.80.165 user=ssupacho
Apr 6 14:01:49 seclab10 login[5221]: FAILED LOGIN 1 FROM 172.28.80.165 FOR ssupacho, Authentication failure

ซึ่งจากข้อมูลร่องรอยการบุกรุกดังกล่าว สามารถนำมาเขียนสำหรับการตรวจจับการโจมตีแบบ Brute-force ผ่านช่องทางของ Telnet ได้ดังนี้

authentication failure.* rhost=(\S+) user=(\S+)::User $2 login failure more than 10 times from $1::** Password Brute-force attack **::email=ssupacho,root::0::10::60

สำหรับการเขียนกฎเพื่อตรวจจับการเกิดเหตุการณ์ในลักษณะนี้ควรพิจารณาถึงเหตุการณ์ปกติที่บางครั้งผู้ใช้ปกติอาจจะมีการใส่รหัสผ่านที่ไม่ถูกต้อง ซึ่งจะทำให้เกิด

ล็อกในรูปแบบนี้ได้เช่นกัน แต่สิ่งที่แตกต่างระหว่างการใส่รหัสผ่านไม่ถูกต้องตามปกติกับการพยายามเดารหัสผ่านก็คือจำนวนครั้งของการเข้าสู่ระบบไม่สำเร็จในช่วงระยะเวลาหนึ่งจะมีปริมาณที่ไม่เท่ากัน โดยความถี่หรือจำนวนครั้งในกรณีที่เป็นการโจมตีจะมีค่าที่สูงกว่ามาก ดังนั้นในกฎดังกล่าวจึงได้กำหนดให้ต้องมีจำนวนเหตุการณ์เกิดขึ้น 10 ครั้งในระยะเวลา 1 นาที เท่านั้นจึงจะถือว่ามีการโจมตีเกิดขึ้นแล้วทำการแจ้งเตือน

5.8.2.2 การเก็บร่องรอยการบุกรุกจากการโจมตีแบบ Passive Attack

สำหรับวิธีการที่ใช้โจมตีแบบ Passive Attack เพื่อเก็บร่องรอยการบุกรุกและทดสอบระบบใช้วิธีการบุกรุกในรูปแบบต่างๆ คือ sniffer, Port Scanning, Exploit ซึ่งรายละเอียดการทดสอบเป็นดังนี้

- **sniffer**

การโจมตีแบบนี้เป็นการดักจับข้อมูลที่อยู่ในเครือข่าย ซึ่งในขณะที่โปรแกรมสนิฟเฟอร์ทำงาน ตัวโปรแกรมจะต้องกำหนดให้เน็ตเวิร์กอะแดปเตอร์อยู่ในโหมดการทำงานพิเศษที่เรียกว่า โพรมิสคูอัสโหมด (Promiscuous Mode) จึงจะทำให้สนิฟเฟอร์สามารถดักจับแพ็กเก็ตได้ทุกแพ็กเก็ตที่วิ่งไปมาในเครือข่ายที่มันถูกติดตั้งไว้ได้ ดังนั้นในการตรวจจับจึงใช้วิธีตรวจสอบว่าหากเน็ตเวิร์กอะแดปเตอร์ของเครื่องคอมพิวเตอร์ถูกกำหนดให้ทำงานในโพรมิสคูอัสโหมด ก็อาจจะเป็นไปได้ว่ามีโปรแกรมประเภทสนิฟเฟอร์ทำงานอยู่บนเครื่องคอมพิวเตอร์ ซึ่งในการทดสอบระบบจะได้สั่งให้โปรแกรมชื่อ sniffer ทำงานบนเครื่อง 10.0.1.10 พบว่าข้อมูลล็อกที่เป็นร่องรอยการบุกรุกที่เกิดขึ้นมีลักษณะดังนี้

```
Jan 31 01:49:57 seclab10 kernel: device lo entered promiscuous mode
Jan 31 01:50:48 seclab10 kernel: device eth0 entered promiscuous mode
```

โดยที่การบันทึกข้อมูลล็อกนี้ทำผ่านค่า facility ของ kernel และค่า priority ตั้งแต่ระดับ info ขึ้นไป ซึ่งหากนำข้อมูลร่องรอยการบุกรุกดังกล่าวมาเขียนเป็นกฎสำหรับการตรวจจับการโจมตีด้วยสนิฟเฟอร์ สามารถเขียนกฎได้ดังนี้

```
device (\S+) entered promiscuous mode::Device $1 entered promiscuous
mode::*** device $1 entered promiscuous mode, check sniffer program in  your
computer***::email=ssupacho
```

- **Surveillance/Probe**

สำหรับการโจมตีที่การสำรวจข้อมูลต่าง ๆของระบบที่ต้องการโจมตี ไม่ว่าจะเป็นค้นหาบริการที่จะสามารถบุกรุกหรือข้ามผ่านเข้าไปยังระบบได้ ด้วยวิธีการ Port Scanning หรือการตรวจบัญชีผู้ใช้ของระบบเมล์ด้วยคำสั่ง VRFY หรือ EXPN นั้น ในการทดสอบได้ใช้โปรแกรมชื่อ GFI LANguard Network Security Scanner โดยติดตั้งโปรแกรมนี้ไว้ในเครื่อง 172.28.80.165 แล้วทำการสแกนไปยังเครื่องเป้าหมายคือ 10.0.1.10 ซึ่งข้อมูลล็อกที่เป็นร่องรอยที่เกิดขึ้นของเหตุการณ์เป็นดังนี้

Apr 6 16:15:14 seclab10 xinetd[1095]: START: ftp pid=8749 from=172.28.80.165
Apr 6 16:15:14 seclab10 xinetd[1095]: START: telnet pid=8751 from=172.28.80.165
Apr 6 16:15:14 seclab10 xinetd[1095]: START: pop3 pid=8753 from=172.28.80.165
Apr 6 16:15:14 seclab10 xinetd[1095]: START: imap pid=8754 from=172.28.80.165
...
Apr 6 16:15:54 seclab10 sendmail[8752]: NOQUEUE: [172.28.80.165] did not issue MAIL/EXPN/VRFY/ETRN during connection to MTA
Apr 6 16:15:57 seclab10 sendmail[8789]: NOQUEUE: [172.28.80.165] did not issue MAIL/EXPN/VRFY/ETRN during connection to MTA
Apr 6 16:16:35 seclab10 ftpd[8802]: FTP session closed
Apr 6 16:16:35 seclab10 xinetd[1095]: EXIT: ftp pid=8802 duration=16(sec)
Apr 6 16:16:43 seclab10 ftpd[8803]: FTP session closed
Apr 6 16:16:43 seclab10 xinetd[1095]: EXIT: ftp pid=8803 duration=15(sec)
Apr 6 16:17:06 seclab10 sendmail[8804]: j369H6908804: [172.28.80.165]: EXPN root [rejected]
Apr 6 16:17:06 seclab10 sendmail[8804]: j369H6908804: [172.28.80.165]: VRFY root [rejected]

โดยที่การบันทึกข้อมูลของเหตุการณ์ลงในล็อกไฟล์ทำผ่านค่า priority ในระดับ info ซึ่งหากนำข้อมูลร่องรอยการบุกรุกดังกล่าวมาเขียนเป็นกฎสำหรับการตรวจจับการโจมตีด้วยวิธีการ Port Scanning ก็จะสามารถเขียนกฎได้ดังนี้

```
NOQUEUE: \[(.*?)\] did not issue .*? during connection to MTA::Client quit before communicating, may be scanning service from $1::Correlation-*** be careful ***::email=ssupacho

\[([0-9\.]+)]: (VRFY|vrfy) (\S+) \[rejected\]::rejected VRFY, may be scanning service from $1::Correlation-*** be careful ***::email=ssupacho
```

- **Exploit**

สำหรับการบุกรุกโดยอาศัยช่องโหว่ของโปรแกรมที่ทำงานอยู่บนเครื่องนั้น ส่วนใหญ่จะเป็นการพยายามอาศัยใช้ช่องโหว่เหล่านั้นในการบุกรุกเพื่อให้ได้ซึ่งสิทธิ

ของผู้ใช้ root ซึ่งในการทดสอบระบบได้ทำโจมตีเครื่อง 10.0.1.10 ซึ่งรายละเอียดการโจมตีแต่ละวิธีดังนี้

- **local exploit for sendmail 8.11.6**

เป็นการโจมตีในลักษณะ buffer overflow ของฟังก์ชัน prescan() เพื่อให้ได้สิทธิของ root ทำการทดสอบโจมตีบนเครื่อง 10.0.1.10 โดยใช้บัญชีผู้ใช้เป็นssupacho แล้วสั่งให้โปรแกรมชื่อ sendmail_prescan ทำงานโดยโปรแกรมนี้ได้จากอินเทอร์เน็ต ซึ่งขั้นตอนการทำงานเป็นดังนี้

```
[ssupacho@seclab10 ssupacho]$ ./sendmail_prescan -h
Local sendmail 8.11.6 exploit by sorbo (sorbox@yahoo.com)
Usage: ./sendmail_prescan <opts>
-h     this lame message
-t     target
-b     brute force

Id     Description     pvpbuf          zero          chunk          shellcode addr
0)     Slackware 8.0   0xbffdfef4      0xbffe15d6    0x80f30a0      0xbffe1f36
1)     Redhat 7.3      0xbffdfcd0      0xbffe19a6    0x80f30a0      0xbffe1f36
2)     Redhat 7.2      0xbffdfcd0      0xbffe19a6    0x80f30a0      0xbffe1f36

[ssupacho@seclab10 ssupacho]$ ./sendmail_prescan -t2
Local sendmail 8.11.6 exploit by sorbo (sorbox@yahoo.com)
Attempting to exploit Redhat 7.2
pvpbuf=         0xbffdfcd0
zero=          0xbffe19a6
chunk=          0x80f30a0
shellcode=      0xbffe1f36
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\... prescan: token too long
aliasing/forwarding loop broken (1094795585 aliases deep; 10 max)
sh-2.05# id
uid=0(root) gid=0(root) groups=500(ssupacho)
sh-2.05#
```

ซึ่งหลังจากสั่งให้โปรแกรมนี้ทำงานแล้ว และใช้คำสั่ง id เพื่อแสดงข้อมูลของผู้ใช้ (uid) และกลุ่มผู้ใช้ (gid) พบว่ามีค่าเป็นศูนย์ ซึ่งแสดงมีสิทธิเป็นผู้ใช้ root และเมื่อตรวจสอบข้อมูลล็อกเพื่อหาร่องรอยของเหตุการณ์นี้ พบว่าไม่มีร่องรอยใด ๆ เกิดขึ้นในล็อกไฟล์ ดังนั้นจึงไม่สามารถเขียนกฎเพื่อตรวจจับการบุกรุกด้วยวิธีนี้ได้

- **Linux kernel ptrace/kmod local root exploit**

การโจมตีวิธีนี้จะอาศัยช่องโหว่ของเหตุการณ์ race condition ในไฟล์ kernel/kmod.c ทำการทดสอบโจมตีบนเครื่อง 10.0.1.10 โดยใช้บัญชีผู้ใช้เป็น ssupacho แล้วสั่งให้โปรแกรมชื่อ ptrace-kmod ทำงานโดยโปรแกรมนี้ได้จากอินเทอร์เน็ต ซึ่งขั้นตอนการทำงานเป็นดังนี้

```
[ssupacho@seclab10 ssupacho]$ ./ptrace-kmod
sh-2.05# id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6
(disk),10(wheel)
sh-2.05#
```

ซึ่งหลังจากสั่งให้โปรแกรมนี้ทำงานแล้ว และใช้คำสั่ง id เพื่อแสดงข้อมูลของผู้ใช้ (uid) และกลุ่มผู้ใช้ (gid) พบว่ามีค่าเป็นศูนย์ ซึ่งแสดงมีสิทธิเป็นผู้ใช้ root และเมื่อตรวจสอบข้อมูลล็อกเพื่อหาร่องรอยของเหตุการณ์นี้ พบว่าไม่มีร่องรอยใด ๆ เกิดขึ้นในล็อกไฟล์ ดังนั้นจึงไม่สามารถเขียนกฎเพื่อตรวจจับการบุกรุกด้วยวิธีนี้ได้

- **DoS with too many fork()**

เป็นการโจมตีในลักษณะ DoS บนเครื่องที่กำลังใช้งานอยู่ โดยใช้โปรแกรมซึ่งได้จากอินเทอร์เน็ต พัฒนาด้วยภาษาซี ชื่อ superforker โดยโปรแกรมนี้จะสร้างโปรเซสขึ้นมาจำนวนมากโดยใช้ฟังก์ชัน fork() จนทำให้เกิดความเสียหายต่อทรัพยากรของระบบ ซึ่งการทดสอบโจมตีทำบนเครื่อง 10.0.1.10 โดยใช้บัญชีผู้ใช้เป็น ssupacho แล้วสั่งให้โปรแกรม superforker ทำงาน สำหรับขั้นตอนการทำงานเป็นดังนี้

```
[ssupacho@seclab10 ssupacho]$ ./superforker
Now crashing and blowing up this system.. have a nice day
You can safely logout, and let the proggie do its work
or you can stick around and watch lag go from 0 to bitch
in a matter of seconds
                              --CoViN
[ssupacho@seclab10 ssupacho]$ Making dirs..
 10:32pm  up 22 min,  3 users,  load average: 0.14, 0.61, 0.38
 10:32pm  up 22 min,  3 users,  load average: 0.14, 0.61, 0.38
 10:32pm  up 22 min,  3 users,  load average: 0.14, 0.61, 0.38
 10:32pm  up 22 min,  3 users,  load average: 0.14, 0.61, 0.38
```

หลังจากนั้นได้ตรวจสอบร่องรอยการบุกรุกในล็อกไฟล์ พบว่าการบันทึกข้อมูลของเหตุการณ์นี้จะทำผ่านค่า facility ของ mail และค่า priority ในระดับ info และร่อยรอยการบุกรุกเป็นดังนี้

```
Jun 26 22:33:15 seclab10 sendmail[1107]: rejecting connections on daemon
MTA: load average: 250
```

```
Jun 26 22:33:31 seclab10 sendmail[1107]: rejecting connections on daemon
MTA: load average: 326
Jun 26 22:33:46 seclab10 sendmail[1107]: rejecting connections on daemon
MTA: load average: 368
Jun 26 22:34:02 seclab10 sendmail[1107]: rejecting connections on daemon
MTA: load average: 400
Jun 26 22:34:17 seclab10 sendmail[1107]: rejecting connections on daemon
MTA: load average: 425
```

ซึ่งหากนำข้อมูลร่องรอยการบุกรุกดังกล่าวมาเขียนเป็นกฎสำหรับการตรวจจับการโจมตีด้วยวิธีนี้ ก็จะสามารถเขียนกฎได้ดังนี้

```
reject connections on deamon (\S+):load average: ([0-9]+)::too many load on
$1::Correlation-*** checking load/process on your machine
***::email=ssupacho
```

5.9 สรุป

สำหรับบทนี้ได้กล่าวถึงการพัฒนาโปรแกรมสำหรับใช้การวิเคราะห์ล็อก โดยการพัฒนาใช้ภาษา Perl และโครงสร้างข้อมูลแบบ associative array ในการเก็บข้อมูลกฎและเหตุการณ์ต่าง ๆ ในระหว่างการวิเคราะห์ข้อมูลรวมถึงการสร้างกฎต่าง ๆ เพื่อใช้ในการทำงานของโปรแกรม ซึ่งจุดเด่นของโปรแกรมที่พัฒนาอยู่ที่แนวทางในการตรวจจับซึ่งเป็นการผสมผสานระหว่างแนวทางการตรวจจับแบบ anomaly detection คือส่วนของการกรองข้อมูลเหตุการณ์ปกติออกไป และแนวทางการตรวจจับแบบ misuse detection ที่ใช้สำหรับการตรวจจับว่าเหตุการณ์ที่บุกรุกนั้นเป็นเหตุการณ์ประเภทใด โดยงานชิ้นนี้จะเน้นไปที่แนวทาง misuse detection นอกจากโปรแกรมที่พัฒนายังมีความสามารถในการนำตัวแปรพิเศษมาใช้ในการเขียนกฎสำหรับการตรวจจับ และยังสามารถกำหนดค่าจำนวนครั้งที่เกิดเหตุการณ์ในช่วงระยะเวลาหนึ่งได้ด้วย ซึ่งโปรแกรมวิเคราะห์ล็อกส่วนใหญ่ที่มีอยู่ในปัจจุบันไม่มีความสามารถดังกล่าว และต่อไปจะกล่าวถึงการทดสอบโปรแกรมที่พัฒนาขึ้นและผลการศึกษาเพื่อวิเคราะห์หาจุดสมดุลในการบันทึกข้อมูลล็อก