ชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก

HONEYPOT PROGRAM SUITE

นาย ธีรชัย เรืองสูง

นาย ธนวัชร์ ลิมป์พัฒนชัย

นาย อาทิตยพงษ์ สุชินโรจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก

HONEYPOT PROGRAM SUITE

โดย

นาย ธีรชัย เรืองสูง

นาย ธนวัชร์ ลิมป์พัฒนชัย

นาย อาทิตยพงษ์ สุชินโรจน์

อาจารย์ที่ปรึกษา

อ. อัครเดช วัชระภูพงษ์

อ. ธนัญชัย ตรีภาค

ผศ. ธนา หงษ์สุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก

Honeypot Program Suite

ผู้จัดทำ

1. นายธนวัชร์ ลิมป์พัฒนชัย รหัสนักศึกษา 45010320
2. นายธีรชัย เรืองสูง รหัสนักศึกษา 45010350
3. นายอาทิตยพงษ์ สุชินโรจน์ รหัสนักศึกษา 45010965

อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

อาจารย์ที่ปรึกษา

(อาจารย์ ธนัญชัย ตรีภาค)

อาจารย์ที่ปรึกษา

(ผู้ช่วยศาตราจารย์ ธนา หงษ์สุวรรณ)

ชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก

นาย ธนวัชร์ ลิมป์พัฒนชัย 45010320

นาย ธีรชัย เรืองสูง 45010350

นาย อาทิตยพงษ์ สุชินโรจน์ 45010965

อ. อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา

อ. ธนัญชัย ตรีภาค อาจารย์ที่ปรึกษา

ผศ. ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา

ปีการศึกษา 2548

บทคัดย่อ

การบุกรุกเข้ามาในระบบในปัจจุบันมีความสลับสับซ้อน และกลวิธีใหม่ๆเกิดขึ้นอยู่ตลอดเวลา และถึงแม้ว่าเครือข่ายในองค์กรจะมีการป้องกันโดยใช้เครื่องมือป้องกันต่างๆ เช่น ไฟร์วอล, ระบบตรวจจับผู้บุกรุกและการปรับปรุงระบบให้ทันสมัยอยู่เสมอแล้ว ก็ยังพบว่ายังมีการเจาะระบบเข้ามาได้โดยอาศัยช่องโหว่ต่างๆ รวมถึงการตั้งค่าการใช้งานที่ไม่เหมาะสม และจากความสามารถอันจำกัดของเครื่องมือป้องกันระบบที่ส่วนใหญ่จะเป็นแบบRules-base ซึ่งจะอาศัยกฎและฐานข้อมูลที่มี เป็นตัวตัดสินใจว่าเข้าข่ายบุกรุกหรือจำเป็นต้องกำจัดแพ็คเกตนั้นทิ้งหรือไม่ ซึ่งการบุกรุกแบบใหม่ๆ ที่ไม่มีในกฎหรือในฐานข้อมูลก็จะสามารถกระทำได้โดยไม่ถูกตรวจพบ ดังนั้นนอกจากการปรับปรุงระบบและใช้เครื่องมือช่วยป้องกันระบบที่เหมาะสมแล้ว ยังมีประโยชน์อย่างยิ่งในการเข้าใจอย่างแท้จริงในแนวคิดและวิธีการของผู้บุกรุก ว่ามีการกระทำอย่างไรบ้าง ซึ่งจะสามารถนำมาประยุกต์เพื่อป้องกันระบบของเราได้ดียิ่งขึ้น ซึ่งตัวฮันนี่พ็อตโปรแกรม จะทำหน้าที่คัดแยกผู้ใช้งานที่มีพฤติกรรมที่น่าสงสัยออกจากระบบจริง เข้าสู่ระบบเสมือนเพื่อเฝ้าดูและศึกษาพฤติกรรม เพื่อทราบวิธีการที่ผู้บุกรุกกระทำ โดยที่การกระทำใดๆของผู้บุกรุกก็จะไม่ส่งผลกระทบหรือทำอันตรายต่อระบบจริง

HONEYPOT PROGRAM SUITE

Mr. Tanawach	Limpadtanachai	45010320
Mr. Theerachai	Ruangsoong	45010350
Mr. Arthittayapong	Suchinroj	45010965
Mr. Akkradach	Watcharapupong	Advisor
Mr. Tanunchai	Tripak	Co-Advisor
Asst.Prof. Thana	Hongsuwan	Co-Advisor

Academic Year 2005

ABSTRACT

In the present intruder have many new techniques and tools to intrude in our organization. Even if we use many tools to prevent our organization but it's work with it's knowledge base or rules that they set. All most security tools cannot protect and prevent all problems of security of organization. There is no tool which can learn behaviors of hackers, so Honeypot was introduced to solve these problems. Honeypot, a new generation tool , is a trap set to detect or deflect attempts at unauthorized use of information systems and about send the intruders to the virtual system at this we can observe their behavior and capture it without damage that may occurred with real system. So we can learn about their techniques and methods. That will help us to approve our organization.

กิตติกรรมประกาศ

เอกสารฉบับนี้และตัวชิ้นงานชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุกสำเร็จลุล่วงได้ด้วยดี ก็เนื่องมาจากการให้โอกาส การดูแล ให้คำแนะนำต่างๆ การสนับสนุน การให้คำสั่งสอน และให้คำปรึกษาเป็นอย่างดีเสมอมาจาก จากท่านอาจารย์ อัครเดช วัชระภูพงษ์ ท่านอาจารย์ ธนัญชัย ตรีภาค และท่าน ผศ. ธนา หงษ์สุวรรณ ซึ่งต้องขอขอบพระคุณอาจารย์ทั้ง 3 ท่านเป็นอย่างสูง ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ และสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เพื่อให้งานวิจัยดำเนินไปได้อย่างสะดวกและรวดเร็ว ขอบคุณห้องวิจัยไอแซค (ISAG) ที่เป็นแหล่งประสิทธิ์ประสาทวิชาให้ความรู้ความเข้าใจ เป็นสถานที่ ที่มีความอบอุ่น ท่านอาจารย์ พี่ๆ ที่เป็นผู้ให้แนวทางแก้ไขและที่ปรึกษาของชิ้นงานจนลุลวงด้วยดี

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นาย ธีรชัย เรืองสูง
นาย ธนวัชร์ ลิมป์พิฒนชัย
นาย อาทิตยพงษ์ สุชินโรจน์

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....I

บทคัดย่อภาษาอังกฤษ.....II

กิตติกรรมประกาศ.....III

สารบัญ.....IV

สารบัญตาราง.....VIII

สารบัญรูป.....IX

บทที่ 1 บทนำ.....1

1.1 ความสำคัญและที่มาของโครงงาน.....1

1.2 วัตถุประสงค์ของโครงงาน.....1

1.3 ขอบเขตของโครงงาน.....2

1.4 วิธีการดำเนินการ.....2

1.5 ประโยชน์ที่คาดว่าจะได้รับ.....2

1.6 ส่วนประกอบของปริญญานิพนธ์.....3

บทที่ 2 ทฤษฎีพื้นฐานที่ใช้ในการพัฒนาระบบฮันนี่พ็อต.....4

2.1 แนวคิดเริ่มต้นของฮันนี่พ็อต.....4

2.2 วิวัฒนาการของฮันนี่พ็อต.....4

2.4 ความรู้พื้นฐานเกี่ยวกับการบุกรุก.....6

2.5 ความรู้พื้นฐานเกี่ยวกับไฟล์วอล.....8

2.6 ความรู้พื้นฐานเกี่ยวกับระบบตรวจจับผู้บุกรุก.....12

บทที่ 3 ระบบฮันนี่พ็อต.....14

3.1 องค์ประกอบของระบบฮันนี่พ็อต.....14

3.2 หน้าที่การทำงานในแต่ละส่วนและเครื่องมือที่ใช้.....16

3.2.1 จำแนกผู้บุกรุกออกจากผู้ใช้งานทั่วไปและส่งเข้าใช้งานเครื่องกับดัก..... 16

3.2.2 ส่วนเฝ้าบันทึกพฤติกรรม และควบคุมขอบเขตของผู้บุกรุก.....19

3.2.3 ส่วนจัดเก็บข้อมูลกลาง.....23

3.2.4 ส่วนการจัดการและตรวจสอบสภาพความเสียหายของเครื่องกับดัก.....27

สารบัญ (ต่อ)

หน้า

3.2.5 ส่วนการควบคุมและแสดงผลการบันทึกข้อมูลต่างๆของระบบ 27
3.3 การทำงานร่วมกันของระบบ.....28
3.4 สรุปการทำงานของระบบโดยรวม.....30
3.5 ข้อจำกัดของฮันนี่พ็อตแบบเดิม.....31
3.6 รูปแบบระบบฮันนี่พ็อตที่เราได้พัฒนาขึ้นใหม่.....31
3.7 ข้อควรระวังในการใช้งานระบบฮันนี่พ็อต.....32

บทที่ 4 เครื่องมือที่นำมาประยุกต์ใช้งาน.....33
4.1 Sebek.....33
4.2 Samhain.....36
4.3 Snort_inline.....39
4.4 IPsec.....41
4.5 S2I.....45
4.6 Iptables.....46

บทที่ 5 เครื่องมือที่พัฒนาขึ้นใหม่.....53
6.I Tartarus Management.....53
6.2 Cage Prototype.....58

บทที่ 6 การทดลองและผลการทดลอง.....59
6.1 โครงสร้างระบบ.....59
6.2 ขั้นตอนการทดสอบชุดโปรแกรม.....60
6.3 ผลการทดลอง.....61

บทที่ 7 บทวิจารณ์และสรุป.....72
7.1 บทสรุป.....72
7.2 วิจารณ์สิ่งที่ได้จากโครงงาน.....72
7.3 ปัญหาอุปสรรคและแนวทางแก้ไข.....73

สารบัญ (ต่อ)

หน้า

7.4 แนวทางการพัฒนาต่อ.....73

บรรณานุกรม.....75

สารบัญตาราง

ตารางที่ หน้า

4.1 ตารางแสดงความสามารถของ AH, ESP และ AH+ESP.....41

6.1 ตารางกำหนดหมายเลขไอพีประจำเครื่องแต่ละเครื่อง.....61

สารบัญรูป

รูปที่ หน้า

2.1 แสดงการทำงานของระบบตรวจจับผู้บุกรุก.....12

3.1 แสดงโครงสร้างของโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก.....14

3.2 แสดงรูปแบบการวางทับข้อมูลในแพ็กเก็ตของระบบตรวจจับผู้บุกรุก.....21

3.3 แสดงการเชื่อมต่อระหว่างส่วนต่าง ๆ เพื่อทำการจำแนกผู้ใช้งานปกติกับผู้บุกรุกแบบเก่า.....28

3.4 แสดงการเชื่อมต่อระหว่างส่วนต่าง ๆ เพื่อทำการจำแนกผู้ใช้งานปกติกับผู้บุกรุกแบบใหม่....29

3.5 แสดงการทำงานของส่วนจำแนกและกำหนดทิศทางการเชื่อมต่อ.....29

4.1 แสดงการทำงานของ Sebek.....34

4.2 แสดงการcapture ข้อมูลที่ทำภายใน kernel.....34

4.3 แสดงการby-pass ไม่ผ่าน TCP stack.....35

4.4 แสดงองค์ประกอบการทำงาน.....41

4.5 แสดงรูปแบบแพ็คเกตของ IPsec.....42

4.6 แสดง Authentication Header.....43

4.7 แสดง Encapsulated Security Payload.....44

4.8 แสดงไดอะแกรมแสดงการทำงานของ S2I.....45

5.1 แสดงโปรแกรม TM (Tartarus Management).....53

5.2 แสดงรูปในส่วนที่ใช้ในการกำหนดกฎให้กับระบบตรวจจับผู้บุกรุก.....54

5.3 แสดงส่วนจัดการเครื่องกับดัก.....55

5.4 แสดงข้อมูลของเครื่องกับดัก.....56

5.5 แสดง log ที่เก็บอยู่ในฐานข้อมูลในเครื่อง Logserver.....57

6.1 แสดงโครงสร้างโครงระบบที่ทำการทดลอง.....59

6.2 แสดงสถานะกฎของไอพีเทเบิ้ล.....62

6.3 แสดงรายละเอียดของเครื่องกับดักต่างๆ.....62

6.4 แสดงรายละเอียดของการเชื่อมต่อ.....63

6.5 แสดงการเชื่อมต่อของเครื่องกับดัก.....63

6.6 แสดงถึงการพยายามบุกรุกเครื่อง FTP.....64

6.7 แสดงตาราง NAT ของเครื่อง honeywall.....64

6.8 ผู้บุกรุกได้เข้ามายังเครื่อง FTP ได้สำเร็จ.....65

6.9 แสดงสถานะของเครื่องกับดักในขณะที่ cage1 ได้ start แล้ว.....65

6.10 แสดงการเชื่อมต่อของเครื่องกับดักหลังจากผู้บุกรุกเข้ามาแล้ว.....66

สารบัญ (ต่อ)

หน้า

6.11 แสดงการใช้คำสั่ง route เพื่อตรวจสอบตัวตน.....66
6.12 แสดงการใช้คำสั่ง ping เพื่อตรวจสอบตัวตน.....67
6.13 แสดงการใช้คำสั่ง ifconfig เพื่อตรวจสอบตัวตน.....67
6.14 แสดงการใช้คำสั่ง vi ไฟล์ /etc/passwd.....68
6.15 แสดงข้อมูลที่ได้จากการเก็บพฤติกรรมมาจากเครื่องกับดัก.....68
6.16 แสดงข้อมูลในไฟล์ /etc/shadow ก่อนถูกเปลี่ยนแปลง.....69
6.17 แสดงการที่ผู้บุกรุกพยายามทำการแก้ไขไฟล์ /etc/shadow.....69
6.18 แสดงข้อมูลในไฟล์ /etc/shadow หลังเปลี่ยนแปลงแล้ว.....70
6.19 แสดงตาราง NAT ของเครื่อง Honeywall.....70
6.20 แสดงข้อมูลของเครื่องกับดักในเครื่อง Logserver.....71
6.21 แสดงข้อมูลของเครื่องต่างๆ ในระบบ.....71

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ในปัจจุบันการบุกรุกได้มีรูปแบบการบุกรุกที่สลับซับซ้อนมากขึ้น ทั้งเกิดจากผู้ใช้งานภายในองค์กรเอง และจากผู้ไม่ประสงค์ดีจากภายนอก ดังนั้นการดูแลระบบจำเป็นต้องอาศัยเครื่องมือช่วยในการตรวจสอบและป้องกันการบุกรุก แต่ด้วยข้อจำกัดของเครื่องมือป้องกันเหล้านี้จะป้องกันการบุกรุกที่เป็นที่รู้จักเท่านั้น นั่นคือหากนอกเหนือจากข้อมูลที่ใช้อ้างอิง เครื่องมือเหล้านี้ก็จะไม่สามารถป้องกันการบุกรุกได้ และในโลกแห่งความเป็นจริงก็ได้พบว่ามีวิธีการบุกรุกใหม่ๆอยู่เสมอ ทั้งจากช่องโหว่ของระบบ หรือจากการตั้งค่าการใช้งานอย่างไม่ปลอดภัย ดังนั้นการรับรู้วิธีการใหม่ๆ ของผู้บุกรุกจึงเป็นอีกหนทางหนึ่งที่จะทำให้เราสามารถปรับปรุงระบบให้มีความปลอดภัยมากขึ้น

โครงการนี้เป็นการพัฒนาโครงการต่อเนื่องจากปี 2547 โดยปรับปรุงโครงสร้างเพื่อสามารถดูแลจัดการองค์กรที่ใหญ่ขึ้น และพัฒนาโปรแกรมให้มีความแม่นยำในการจำแนกผู้บุกรุกออกจากผู้ใช้งานปกติ , สามารถดักจับโค้ดหนอนที่แพร่กระจายผ่านเครือข่ายได้, พัฒนาระบบเสมือนหรือเครื่องกับดัก (Cage) มีความแนบเนียนมากยิ่งขึ้นเพื่อมิให้ผู้บุกรุกทราบว่ากำลังอยู่ในเครื่องกับดัก, พัฒนาความปลอดภัยของระบบที่จะไม่ให้ผู้บุกรุกใช้เครื่องกับดักเป็นฐานการโจมตีไปยังเครื่องอื่นๆ รวมทั้งสร้างโปรแกรมควบคุมทั้งระบบเพื่อเป็นศูนย์กลางในการดูแลจัดการระบบฮันนี่พ็อต ซึ่งจะช่วยลดความยุ่งยากที่เกิดจากการใช้ระบบฮันนี่พ็อตเป็นอย่างมาก

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาวิธีการจำแนกผู้บุกรุกจากผู้ใช้งานปกติแล้วบันทึกพฤติกรรมของผู้บุกรุก
2. เพื่อปรับปรุงโปรแกรมให้สามารถดักจับโค้ดหนอนที่แพร่กระจายได้
3. เพื่อปรับปรุงโปรแกรมต้นแบบสำหรับจำแนกและบันทึกพฤติกรรมผู้บุกรุกให้มีความแนบเนียน
4. เพื่อปรับปรุงระบบหลอกให้มีความปลอดภัย ไม่ให้ผู้บุกรุกใช้ฮันนี่พ็อตเป็นฐานที่มั่นในการเข้าโจมตีระบบอื่นๆ
5. เพิ่มความง่ายและความสะดวกในการบำรุงรักษาระบบหลอก (Cage) ให้ดียิ่งขึ้น

1.3 ขอบเขตของโครงงาน

1. ศึกษาการจัดตั้งระบบฮันนี่พ็อตที่เหมาะสมในระบบเครือข่าย
2. ศึกษาวิธีการจำแนกผู้บุกรุกจากผู้ใช้งานปกติแล้วบันทึกพฤติกรรม
3. ระบบสามารถทำการดักจับโค้ดหนอนได้หรือผู้บุกรุกที่เป็นโปรแกรมอัตโนมัติได้
4. มีความแนบเนียนในการล่อหลอกผู้บุกรุกมากยิ่งขึ้น
5. เพิ่มความปลอดภัยของระบบหลอก(Cage) ให้มีความปลอดภัยมากขึ้น
6. มีความสามารถในการซ่อมแซมและบำรุงรักษาระบบหลอก (Cage) อย่างอัตโนมัติ

1.4 วิธีการดำเนินการ

1. ศึกษาความรู้พื้นฐานเพื่อให้เข้าใจระบบฮันนี่พ็อต
2. ศึกษาส่วนที่เพิ่มเติมเพื่อให้ระบบฮันนี่พ็อตมีประสิทธิภาพมากขึ้น
3. วิเคราะห์ความเหมาะสมในการวางตัวระบบฮันนี่พ็อต
4. หาแนวทางการจำแนกผู้บุกรุกจากผู้ใช้งานปกติ
5. ทดลองการใช้ระบบฮันนี่พ็อตร่วมกับส่วนเพิ่มเติมที่ศึกษามา
6. ทำรายงานและสรุปผล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้ความเข้าใจเกี่ยวกับวิธีการทำงานของระบบตรวจจับผู้บุกรุก
2. ได้รับความรู้ความเข้าใจเกี่ยวกับการทำงานของโปรแกรมไฟร์วอล
3. ได้รับความรู้ความเข้าใจเกี่ยวกับวิธีการจำแนกผู้บุกรุกออกจากผู้ใช้ทั่วไป
4. ได้รับความรู้ความเข้าใจเกี่ยวกับวิธีการบุกรุกระบบและวิธีการป้องกันการบุกรุก

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 8 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงาน ประกอบด้วยแนวคิดและวิวัฒนาการของ

ฮันนี่พ็อตตลอดจนการจัดทำเป็นระบบฮันนีพ็อต จุดเด่นและจุดด้อยของแต่ละแนวคิด รวมทั้งการพัฒนาเพิ่มเติมเพื่อแก้ไขจุดอ่อนของระบบเดิมและข้อควรระวังในการใช้งานระบบฮันนี่พ็อต

บทที่ 3 กล่าวถึงหน้าที่การทำงานทั้งหมดที่ระบบฮันนี่พ็อตรับผิดชอบ และองค์ประกอบของระบบที่ประกอบกันขึ้นเป็นระบบฮันนี่พ็อต

บทที่ 4 กล่าวถึงรายละเอียดของแต่ละองค์ประกอบของระบบฮันนี่พ็อต หน้าที่การทำงานและเครื่องมือและเทคนิคที่ใช้ในแต่ละองค์ประกอบ

บทที่ 5 กล่าวถึงโปรแกรมที่ได้พัฒนาขึ้นใหม่และการนำมาใช้ร่วมกัน ของส่วนต่างๆ

บทที่ 6 การทดลองและผลการทดลอง

บทที่ 7 กล่าวถึงบทวิจารณ์และสรุปผล รวมทั้งปัญหาและอุปสรรค์ที่พบ แนวทางในการพัฒนาต่อและข้อเสนอแนะ

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้ในการพัฒนาระบบฮันนี่พ็อต

ในหัวข้อนี้จะกล่าวถึงทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องในการวิจัย และพื้นฐานของระบบฮันนี่พ็อตซึ่งเนื้อหาในบทนี้จะกล่าวถึงแนวคิดของฮันนี่พ็อต. การทำงานของระบบตรวจจับผู้บุกรุก, การทำงานของไฟร์วอล, และรูปแบบการบุกรุกแบบต่างๆ ซึ่งเนื้อหาทั้งหมดนี้จำเป็นสำหรับการศึกษา และพัฒนาระบบฮันนี่พ็อต

2.1 แนวคิดของฮันนี่พ็อต

แนวคิดของฮันนี่พ็อตคือ ต้องการทราบการกระทำ และความพยายามต่างๆที่ผู้บุกรุกกระทำต่อระบบ ดังนั้นเพื่อให้ได้มา จำเป็นจะต้องสร้างระบบบางอย่างให้เหล่าผู้บุกรุกได้เข้ามาและระบบต้องสามารถจับตาดูได้ โดยที่ผู้บุกรุกไม่รู้ตัว ว่ากำลังโดนจับตามองอยู่ และโดยที่ไม่ส่งผลเสียหายใดๆต่อระบบจริง เปรียบเสมือนล่อหลอกให้ขโมยพยายามงัดเข้ามาในบ้าน แล้วใช้กล้องวงจรปิดถ่ายภาพไว้ทุกๆการกระทำตั้งแต่วิธีที่ใช้ในการงัดเข้ามาในบ้าน พยายามทำอะไรบ้าง สนใจสำรวจอะไรบ้างหรือได้นำอะไรออกไป โดยที่บ้านและของมีค่าใดๆ เป็นเพียงภาพลวงตาเท่านั้น ทำให้ไม่ต้องเสียทรัพย์สินหรือก่อให้เกิดอันตรายใดๆ ซึ่งทำให้เราสามารถเห็นวิธีที่ผู้บุกรุกใช้ ตลอดจนเครื่องมือใหม่ๆที่ผู้บุกรุกอาจนำมาใช้ ซึ่งเมื่อเราทราบวิธีและเครื่องมือของผู้บุกรุก ทำให้เราสามารถนำข้อมูลที่ได้ไปประยุกต์เพื่อปรับปรุงและป้องกันระบบจริงให้มีความปลอดภัยมากยิ่งขึ้น

2.2 วิวัฒนาการของฮันนี่พ็อต

เริ่มแรกมีความต้องการทราบการกระทำของผู้บุกรุก เพื่อเรียนรู้วิธีการและสิ่งที่ผู้บุกรุกสนใจ โดยในยุคแรกนี้ได้เริ่มสร้างเพียงการบริการเสมือน(serviceที่จำลองขึ้นมาให้เหมือนserviceจริงๆ) หรืออาจเรียกว่าเป็นแบบ Low-interaction ฮันนี่พ็อตที่มีลักษณะแบบนี้จะเป็นการจำลองบริการต่างๆขึ้นทำให้ดูเสมือนว่ามีเครื่องที่เปิดให้บริการต่างๆ เช่น FTP เป็นต้น ซึ่งฮันนี่พ็อตชนิดนี้จะมีความจำกัดในส่วนของการโต้ตอบการโจมตีของผู้บุกรก ซึ่งการโต้ตอบของฮันนี่พ็อตรูปแบบจะขึ้นอยู่กับการทำส่วนจำลองบริการและขึ้นอยู่กับระบบปฏิบัติการที่ใช้ เช่น อาจทำให้สามารถล็อกอินได้ หรือทำให้ใช้คำสั่งต่างๆ ได้เหมือนกับบริการจริงๆ ตัวอย่าง Honeypot ที่เป็นแบบ Low-interaction เช่น Specter, Honeyd และ KFSensor

ข้อดีของฮันนี่พ็อตแบบ Low-interaction นี้คือมีความยืดหยุ่นง่ายต่อการเปลี่ยนแปลงและบำรุงรักษาตัวฮันนี่พ็อต และมีอัตราเสี่ยงเล็กน้อย ซึ่งโดยทั่วไปแล้วมีระบบ Plug & Play ซึ่งจะทำให้การเปลี่ยนแปลงเป็นไปได้โดยสะดวก การจำลองบริการนั้นจะช่วยลดความเสี่ยงโดยสามารถบรรจุสิ่งที่จะใช้ตอบโต้ผู้บุกรุก ผู้บุกรุกจะไม่สามารถเข้ายึดครองระบบปฏิบัติการนั้น (honeypot) เพื่อใช้ในการโจมตีผู้อื่นต่อไปได้

ข้อเสียของฮันนี่พ็อตแบบ Low-interaction นี้คือสามารถออกแบบได้เฉพาะการตรวจจับการกระทำที่ทราบอยู่แล้วเท่านั้น ซึ่งมีข้อจำกัดมากมายในการเฝ้าดูพฤติกรรมผู้บุกรุก เพราะสิ่งที่เห็นจะแค่เพียงส่วนที่เกี่ยวข้องกับserviceนั้นเท่านั้น

ต่อมาจึงมีการสร้างเป็นเครื่องให้บริการเสมือน(เสมือนเป็นเครื่องคอมพิวเตอร์ที่ทำงานจริง ซึ่งจะมีเรื่องของระบบปฏิบัติการเข้ามาเกี่ยว)อาจเรียกว่าเป็นแบบ High-interaction ฮันนี่พ็อตที่มีลักษณะแบบนี้จะมีทั้งระบบปฏิบัติการ แอพพลิเคชั่น และบริการจริงๆ เพื่อตอบโต้กับผู้บุกรุก นั่นคือถ้าต้องการให้มีบริการใดก็ทำการลงโปรแกรม หรือเปิดบริการต่างๆ ที่เครื่องจริงๆ อย่างเช่น ถ้าจะเปิดบริการ FTP ก็ลงโปรแกรมอย่างเช่น Ftpd เป็นต้น

ข้อดีของฮันนี่พ็อตแบบ High-interaction คือข้อมูลที่ได้จากการกระทำของผู้บุกรุกนั้นมีความครอบคลุมมากกว่าเพราะว่ามีการโต้ตอบทุกอย่างที่เหมือนจริง ดังนั้นเราจึงสามารถสรุปผลข้อมูลได้จากการโจมตีจริง ระบบจะตอบโต้กับผู้บุกรุกจริงๆ เราจึงสามารถเรียนรู้ขอบเขตของพฤติกรรมของเหล่า ผู้บุกรุกได้ดี และข้อดีอีกข้อหนึ่งคือฮันนี่พ็อตแบบ High-interaction นั้นไม่ได้จำลองว่าการตอบโต้กลับไปยังผู้บุกรุกว่าควรจะตอบกลับแบบใดเพราะที่นั้นใช้เป็นตัวระบบจริงที่ติดตั้งอยู่บนฮันนี่พ็อต สามารถจับตาดูการกระทำต่อระบบซึ่งอาจส่งผลต่อบริการที่เครื่องนั้นๆให้บริการอยู่ หรือข้อมูลที่ผู้บุกรุกสนใจ และทำการสำรวจหรือนำออกไป โดยจะสามารถเฝ้าดูพฤติกรรมได้ครอบคลุมกว่าแบบเดิมมาก

ข้อเสียฮันนี่พ็อตแบบ High-interaction คือการเพิ่มอัตราเสี่ยงให้กับระบบ เนื่องจากผู้บุกรุกนั้นสามารถที่อาจสามารถเข้ายึดครองเครื่องที่ทำหน้าที่เป็นฮันนี่พ็อตแล้วใช้เครื่องนั้นๆ ในการจมตีระบบอื่นๆ ก็เป็นไปได้เช่นกัน ดังนั้นการใช้งานฮันนี่พ็อตที่เป็นแบบ High-interaction นั้นจึงต้องหาทางควบคุมพฤติกรรมของผู้บุกรุกให้อยู่ภายในขอบเขตที่เรากำหนดและไม่สามารถใช้ฮันนี่พ็อตเป็นฐานที่มั่นในการเข้าโจมตีระบบอื่นๆได้ ตัวอย่างของตัวฮันนี่พ็อคแบบ High-interaction คือ Honeynets เป็นต้น โดยการกระทำที่สามารถจับตาดูได้ในยุคนี้จะเป็นการกระทำที่ไม่มีเรื่องของการเข้ารหัสลับเข้ามาเกี่ยวข้อง เช่นการทำงานผ่านtelnetซึ่งไม่มีการเข้ารหัสข้อมูลก่อนทำการส่ง ดังนั้นเมื่อมีเทคโนโลยีหรือเครื่องมือใหม่ๆที่มีการเข้ารหัส เช่น SSL, SSH, SCP ก็จะไม่สามารถดักดูข้อมูลได้

ต่อมาได้มีการนำเครื่องมือที่มีลักษณะของRootkits ซึ่งมีการทำงานแบบ Kernel-base toolsเข้ามาใช้งานเพื่อให้สามารถเข้าถึงข้อมูลที่ทำการเข้ารหัส โดยที่เครื่องมือเหล่านี้ไม่ได้ไปนำข้อมูลมาถอดรหัส แต่จะเป็นการดูข้อมูลก่อนที่จะถูกเข้ารหัส(กรณีที่เป็นฝั่งส่ง)และอ่านข้อมูลหลังจากที่ถอดรหัสเรียบร้อยแล้ว(ในฝั่งรับ) ทำให้สามารถจับตาดูการกระทำของผู้บุกรุกได้ แม้จะใช้ SSHเข้ามายังเครื่องกับดัก ดังนั้นสิ่งที่เราสามารถบันทึกได้จะคลอบคลุม พฤติกรรมของผู้บุกรุกทั้งหมด เช่น การกดkeyboard การใช้คำสั่ง การถ่ายโอนไฟล์ การนำข้อมูลเข้าออก หรือการเรียกใช้โปรแกรม ได้มีการเพิ่มคุณสมบัติในการซ่อนตัวเองไม่ให้ผู้บุกรุกตรวจพบว่ากำลังทำงานอยู่ และในยุคนี้ได้เริ่มนำฮันนี่พ็อตเข้ามาเป็นส่วนหนึ่งของระบบจริง นั้นคือมีการทำการคัดแยกผู้บุกรุกออกจากผู้ใช้งานทั่วไป โดยมีการจัดเป็นโครงสร้างของระบบฮันนี่พ็อตขึ้น โดยประกอบขึ้นด้วยหลายๆส่วน ซึ่งทำหน้าที่ต่างๆกันไป เช่น Honeywall, Log server, Cage โดยแยกเครื่องกับดักออกจากเครื่องให้บริการจริง เพื่อไม่ให้การกระทำต่อเครื่องกับดักเกิดความเสียหายต่อเครื่องให้บริการจริง แต่ยังมีข้อจำกัดในการบันทึกผลการเปลี่ยนแปลงของระบบหลังจากมีการใช้คำสั่ง หรือการเรียกใช้โปรแกรม

2.3 ความรู้พื้นฐานเกี่ยวกับการบุกรุก

บุคคลที่พยายามเข้ามายังระบบโดยไม่ได้รับอนุญาตเราเรียกว่า "ผู้บุกรุก" หรือ "intruder" ซึ่ง

2.3.1 ผู้บุกรุกนั้นอาจแบ่งได้เป็น 2 ประเภทคือ

2.3.1.1 Outsider intruder (ผู้บุกรุกจากภายนอก) หมายถึง บุคคลภายนอกเครือข่ายที่พยายามเจาะเข้ามาในระบบหรือพยายามโจมตีระบบจากภายนอก เช่น การเจาะเข้ามายังเครื่องเว็บเซิร์ฟเวอร์แล้วเปลี่ยนหน้าเว็บไซต์ของเรา เป็นต้น ซึ่งการบุกรุกนี้อาจมาจากอินเทอร์เน็ต, การ dial-up หรือการบุกเข้าไปยังเครือข่ายของคู่ค้าแล้วเชื่อมต่อมายังเครือข่ายของเรา

2.3.1.2 Insider intruder (ผู้บุกรุกจากภายใน) หมายถึง บุคคลภายในเครือข่าย รวมทั้งผู้ใช้ที่ใช้สิทธิ์ในทางที่ผิด หรือการลักลอบใช้สิทธิ์ของผู้ใช้คนอื่นๆ ที่มีสิทธิ์เหนือกว่า

2.3.2 เส้นทางที่ผู้บุกรุกสามารถเข้าสู่ระบบมี 3 ทางหลักๆ คือ

2.3.2.1 Physical Intrusion คือการที่ผู้บุกรุกทำการเชื่อมต่อทางกายภาพกับเครื่องหรือระบบเครือข่ายเช่น มานั่งหน้าเครื่อง หรือนำเครื่องมาเสียบสายแลน เป็นต้น

2.3.2.2 System Intrusion คือการที่ผู้บุกรุกมีรหัสผ่านเรียบร้อยแล้วแต่เป็นรหัสผ่านของบัญชีผู้ใช้ที่มีสิทธิ์ต่ำ ถ้าระบบไม่ได้มีการอัพเดตแพท ผู้บุกรุกจะใช้ช่องโหว่ของระบบในการเข้าครอบครองสิทธิ์ของผู้ดูแลระบบได้

2.3.2.3 Remote Intrusion ผู้บุกรุกพยายามที่จะเจาะเข้าสู่ระบบข้ามเครือข่าย ซึ่งผู้บุกรุกจะไม่มีสิทธิ์ใดๆ เลยบนระบบเครือข่ายนั้น

2.3.3 ตัวอย่างการโจมตีระบบ

2.3.3.1 IP spoofing เป็นเทคนิคที่ใช้เพื่อให้ผู้ที่ไม่มีสิทธิ์เข้ามายังคอมพิวเตอร์ได้โดยผู้บุกรุกจะทำการส่งข้อความไปยังคอมพิวเตอร์ต่างๆ โดยหลอกว่าข้อความนั้นมาจากไอพีแอดเดรสที่เป็นที่น่าเชื่อถือ โดยอาจมีวิธีการโจมตีแตกต่างกันไปดังต่อไปนี้

1. Non-blind spoofing เป็นการโจมตีที่เกิดขึ้นได้เมื่อผู้บุกรุกอยู่บนซับเน็ตเดียวกันกับเครื่องเป้าหมายซึ่งผู้บุกรุกสามารถมองเห็นลำดับ sequence และ acknowledgement ของแพ็คเกตต่างๆ ได้ เมื่อผู้บุกรุกรู้ลำดับของแพ็คเกตซึ่งอาจทำได้โดยการใช้โปรแกรมดักจับแพ็คเกตที่วิ่งไปมาอยู่ในเครือข่ายแล้วนำมาพิจารณาแล้ว ผู้บุกรุกสามารถทำ session hijacking ได้จากนั้นผู้บุกรุกอาจทำการดักเอาส่วนของการยืนยันตนต่างๆ ที่ใช้ในการสถาปนาเชื่อมต่อไว้ ทำให้การสถาปนาเชื่อมต่อนั้นล้มเหลว จากนั้นจึงทำการสถาปนาการเชื่อมต่อมันโดยการแก้ไข sequence และ acknowledgement number ให้ตรงกับเครื่องของผู้บุกรุกก็ถือว่าการโจมตีนี้ทำได้สำเร็จ.

2. Bind spoofing เป็นการโจมตีที่เกิดขึ้นจากภายนอกซึ่งผู้บุกรุกไม่อาจรู้ลำดับ sequence และ acknowledgement ดังนั้นผู้บุกรุกจึงพยายามส่งแพ็คเกตจำนวนมากไปยังเครื่องเป้าหมายอย่างสม่ำเสมอตามลำดับของเลข sequence ที่สุ่มขึ้นมา แต่ในปัจจุบันนี้ระบบปฏิบัติการได้มีการพัฒนาเรื่องของการสุ่มเลข sequence ทำให้การคาดเดาให้ถูกต้องนั้นเป็นไปได้ยาก แต่ถึงอย่างไรก็ตามถ้าเลข sequence ที่ผู้บุกรุกสุ่มขึ้นมานั้นถูกต้องข้อมูลจากเครื่องผู้บุกรุกก็จะถูกส่งไปยังเครื่องเป้าหมายได้เช่นกัน

3. Man in the Middle Attack เรียกได้อีกอย่างหนึ่งว่า connection hijacking ซึ่งการโจมตีแบบนี้จะเป็นการเข้าไปขัดขวางการสื่อสารระหว่างเครื่องโฮสต์ 2 เครื่องและเข้าควบคุมการสื่อสารเพื่อสับเปลี่ยนแก้ไข หรือเพิ่มข้อมูลที่เครื่องทั้ง 2 ส่งถึงกันได้ โดยการที่ผู้บุกรุกจะทำเช่นนี้ได้ผู้บุกรุกเองต้องอยู่ในสถานที่เดียวกับเครื่องทั้ง 2 เครื่องที่ติดต่อกันโดยหัวใจหลักของการโจมตีแบบนี้คือการทำ

ให้การติดต่อระหว่างเครื่องโฮสต์ทั้ง 2 เครื่องอยู่ในสถานะที่เรียกว่า "desynchronized" นั่นคือการทำให้เลขลำดับ sequence ของแพ็คเกตที่ได้รับไม่ตรงกับเลขลำดับ sequence ที่ต้องการนั่นเอง

4. Denial of Service Attack คือ IP spoofing ที่ใช้บ่อยที่สุดในการโจมตีแบบ denial of service หรือ DoS ซึ่งผู้บุกรุกจะทำการใช้แบนวิดช์และทรัพยากรเครือข่ายให้หมดไปโดยการส่งแพ็คเกตจำนวนมากเท่าที่จะทำได้ไปยังเครื่องเป้าหมายในเวลาอันสั้น โดยผู้บุกรุกจะทำการเปลี่ยนไอพีต้นทางไปเรื่อยเพื่อทำให้การหยุด DoS ทำได้ยากยิ่งขึ้น

2.3.3.2 ICMP Smurfing เป็นของโปรแกรมอัตโนมัติที่จะทำการโจมตีเครือข่ายโดยใช้ไอพีบรอดคาสต์ ซึ่ง Smurf และโปรแกรมที่มีลักษณะคล้ายๆ กันนี้สามารถทำให้บางส่วนของเครือข่ายไม่สามารถทำงานได้ โดยทำให้โหนดต่างๆ จะคอยแลกเปลี่ยนข้อมูลเกี่ยวกับระบบเครือข่ายโดยใช้ ICMP อยู่ตลอดเวลา

โปรแกรม Smurf จะสร้างแพ็คเกตที่มีไอพีต้นทางเป็นเครื่องเป้าหมายโดยแพ็คเกตนั้นจะบรรจุ ICMP ping message เอาไว้ซึ่งมีแอดเดรสปลายทางเป็นไอพีบรอดคาสต์ นั่นก็หมายถึงว่าทุกๆ ไอพีแอดเดรสในเครือข่ายจะได้รับแพ็คเก็ตนี้ ซึ่งเมื่อเครื่องโฮสต์ส่วนใหญ่ได้รับแพ็คเกตดังกล่าวจะทำการตอบกลับด้วย ICMP echo reply ทำให้มีแพ็คเกตจำนวนมากกลับไปยังเครื่องเป้าหมายซึ่งถ้าการ ping มีมากพอก็อาจเป็นผลทำให้เครือข่ายไม่สามารถใช้งานได้

2.4 ความรู้พื้นฐานเกี่ยวกับไฟร์วอล

ไฟร์วอลนั้นเป็นเครื่องมือที่ใช้ในการป้องการการบุกรุกหรืออาจใช้เป็นเครื่องมือในการจัดสรร
การใช้ทรัพยากรของระบบเครือข่ายก็ได้

2.4.1 คุณสมบัติทั่วไปของไฟร์วอล

ไฟร์วอลเป็นเครื่องมือรักษาความปลอดภัยที่ทำงานในเชิงป้องกัน ซึ่งทำหน้าที่ในการควบคุมการเข้าถึงระบบเครือข่าย โดยอาศัยกฎเป็นพื้นฐาน (Rule based) สำหรับคุณสมบัติแต่ละอย่างของไฟร์วอลมีรายละเอียดดังนี้

2.4.1.1 การป้องกัน (Protect)

ไฟร์วอลเป็นเครื่องมือที่ใช้งานในเชิงป้องกัน โดยชิ้นข้อมูลที่ผ่านได้นั้น จะต้องเป็นชิ้น

ข้อมูลที่ไฟร์วอลเห็นว่ามีความปลอดภัย ชิ้นข้อมูลที่ไฟร์วอลเห็นว่าไม่ปลอดภัยหรืออาจจะนำมาซึ่งความไม่ปลอดภัยก็จะถูกกำจัด คือไม่ส่งต่อ โดยการที่ไฟร์วอลจะตัดสินใจว่าชิ้นข้อมูลใดปลอดภัยและชิ้นข้อมูลใดไม่ปลอดภัยนั้นจะขึ้นอยู่กับกฎของผู้ดูแลไฟร์วอล เป็นผู้กำหนดไว้ล่วงหน้า ซึ่งเงื่อนไขของกฎเหล่านี้เองทำให้ไฟร์วอลสามารถป้องกันชิ้นข้อมูลที่อาจจะส่งผลร้ายไม่ให้ผ่านเข้าไปถึงเครือข่ายคอมพิวเตอร์ได้

2.4.1.2 ควบคุมการเข้าถึง (Access Control)

การเข้าถึง หมายถึงการที่เครื่องใดเครื่องหนึ่งนั้นสามารถสื่อสารข้อมูลที่ต้องการไปยังเครื่องปลายทางได้สำเร็จ การเข้าถึงในแต่ละระดับจะมีวิธีการแตกต่างกันออกไป ทำให้การควบคุมการเข้าถึงสำหรับแต่ละระดับแตกต่างกันออกไปด้วย ไฟร์วอลเองจึงต้องมีการทำงานหลายลักษณะตามวิธีที่ไฟวอลใช้ควบคุมการเข้าถึง

2.4.1.3 กฎพื้นฐาน (Rule Based)

ไฟร์วอลจะควบคุมการเข้าถึงโดยอาศัยหลักการเปรียบเทียบคุณสมบัติของชิ้นข้อมูลที่จะผ่านไฟร์วอลกับกฎของการเข้าถึงที่ได้กำหนดไว้ หากพบว่าไม่มีกฎที่ห้ามไว้ก็จะอนุญาตให้ชิ้นข้อมูลนั้นผ่านได้ หากมีกฎที่ห้ามไว้ชิ้นข้อมูลนั้นก็จะถูกสกัดกั้นไว้ด้วยวิธีใดวิธีหนึ่ง

2.4.2 สเตตฟูลอินสเป็กชั่นไฟร์วอล (Stateful Inspection Firewall)

การสื่อสารโดยทั่วไปจะเป็นการสื่อสารแบบต่อเนื่อง โต้ตอบกันไปมาระหว่างผู้รับและผู้ส่งอยู่เสมอ โปรโตคอลที่อยู่ในชั้นที่สูงกว่าอินเตอร์เน็ตเลเยอร์ ไม่ว่าจะเป็นทรานสปอร์ตอย่างเช่น ทีซีพี, ยูดีพี หรือเลยไปถึงแอพพลิเคชั่นเลเยอร์ เช่น เอฟทีพี, เอชทีทีพี, เอสเอ็มทีพี ล้วนแล้วแต่จะต้องมีสถานะของการสื่อสาร (State) เสมอ สถานะนี้จะทำให้ทั้งสองฝั่งสามารถสื่อสารกันได้อย่างต่อเนื่อง คือทราบว่าตอนนี้กำลังอยู่ ณ จุดใดและจะต้องส่งหรือรับข้อมูลใดเป็นลำดับต่อไป

2.4.3 ความแตกต่างของการพิจารณาข้อมูลแบบแพ็คเก็ตฟิลเตอร์ริงกับแบบสเตตฟูล

อันที่จริงสองเรื่องนี้มิได้ขัดแย้งกันแต่ประการใด แพ็คเก็ตนั้นเป็นการสื่อสารที่เป็นส่วนย่อยของการสื่อสารทั้งหมด ผลของการสื่อสารข้อมูลก็คือผลรวมของการสื่อสารข้อมูลหลายๆ แพ็คเก็ตนั่นเอง แต่อย่างไรก็ตามการฟิลเตอร์หรือกรอง โดยพิจารณาทีละแพ็คเก็ตของทุกแพ็คเก็ตที่ผ่านเข้าออกนั้นอาจจะมีผลลัพธ์แตกต่างจากการฟิลเตอร์ของในแบบที่สองสถานะและภาพรวมหรือที่

เรียกว่าสเตตฟูล (Stateful) หากเปรียบเทียบการพิจารณาข้อมูลครั้งละแพ็คเก็ตกับการพิจารณาแบบสเตตฟูลแล้ว ตัวอย่างที่น่าจะช่วยให้เข้าใจได้ง่ายขึ้นคือ

เปรียบเทียบการสื่อสารข้อมูลทั้งหมดเสมือนภาพยนตร์ แพ็คเก็ตก็จะหมายถึงภาพนิ่งแต่ละภาพที่นำมาต่อรวมกันแล้วเปิดดูอย่างรวดเร็ว ภาพนิ่งเหล่านั้นจะกลายเป็นภาพเคลื่อนไหว ดังนั้นการพิจารณาแพ็คเก็ตก็เป็นเสมือนการดูภาพนิ่งทีละภาพ แต่จะไม่สามารถเซ็นเซอร์เนื้อเรื่องซึ่งเป็นสิ่งที่เกิดขึ้นจากความสัมพันธ์ของภาพหลายๆ ภาพได้มีโอกาสเป็นไปได้ว่ากิจกรรมบางชนิดที่หากดูเป็นภาพนิ่งแล้วจะไม่รู้สึกว่าสิ่งใดที่ไม่เหมาะสม แต่หากนำภาพนิ่งมาดูอย่างต่อเนื่องเป็นภาพเคลื่อนไหวแล้วก็อาจจะเป็นสิ่งที่ไม่พึงปรารถนาที่จะให้ปรากฏบนภาพยนตร์ก็เป็นได้

สเตตฟูลไฟร์วอลเป็นไฟร์วอลที่ทำงานโดยที่สามารถเข้าใจสถานะของการสื่อสารทั้งกระบวนการ เพราะถือว่าการสื่อสารข้อมูลจะสมบูรณ์ได้นั้นต้องมีทั้งการส่งและการรับอย่างสอดคล้องสัมพันธ์กันนั่นเอง หมายถึงหากไฟร์วอลจะสามารถควบคุมการสื่อสารได้จริงก็จะต้องสามารถเข้าใจกระบวนการของการสื่อสารตั้งแต่ต้นจนจบ โดยทั่วไปเราจะเรียกไฟร์วอลแบบนี้ว่า "สเตตฟูลอินสเป็กชั่นไฟร์วอล" (หรือเรียกย่อๆ ว่าสเตตฟูลไฟร์วอล) เป็นไฟร์วอลที่ทำการควบคุมแทรฟฟิกโดยใช้หลักการของแพ็คเก็จฟิลเตอร์ริง และการกำหนดแอคเซสรูลเช่นเดียวกับสกรีนิงเราเตอร์แต่สเตตฟูลไฟร์วอลจะมีความสามารถในการวิเคราะห์และรับรู้ความต่อเนื่องของแพ็คเก็ตในโปรโตคอลในระดับที่สูงขึ้นไปมากกว่า ไม่ว่าจะเป็น ทีซีพี, เอฟทีพี, เอชทีทีพี หรือแม้กระทั่งโปรโตคอลในระดับแอพพลิเคชั่น ที่จะมีวิธีการกำหนดสเตตของตนเอง อีกทั้งยังเป็นเครื่องมือที่ถูกออกแบบมาเพื่อทำหน้าที่ในการควบคุมแทรฟฟิกโดยเฉพาะไม่ได้เป็นการดัดแปลงการทำงานมาจากเราเตอร์จึงมีความสามารถในการควบคุมแทรฟฟิกการกำหนดแอคเซสรูล การบริการ รวมไปถึงความยืดหยุ่นจองการควบคุมแทรฟฟิก และประสิทธิภาพในการทำงานที่สูงกว่าสกรีนิ่งเราเตอร์เป็นอย่างมาก โดยทั่วไปหากพูดถึงไฟร์วอลจะหมายถึงไฟร์วอลประเภทนี้เอง

ตามที่ได้กล่าวไว้ข้างต้นว่า ความแตกต่างที่สำคัญของไฟร์วอลทั้งสองชนิดนี้ในแง่ของการตรวจสอบแทรฟฟิกคือ สเตตฟูลไฟร์วอล มีความสามารถในการวิเคราะห์แทรฟฟิกที่ผ่านไปมาในโปรโตคอลที่เลเยอร์สูงขึ้นไปไม่ว่าจะเป็น ทีซีพี, ยูดีพี, ไอซีเอ็มพี ได้อย่างสมบูรณ์ต่างจาก สกรีนิงเราเตอร์ที่สามารถวิเคราะห์ได้เฉพาะเท่าที่จะมีข้อมูลใน 1 แพ็คเก็ตเท่านั้นเพราะบางครั้งแทรฟฟิกที่อ่านไปมานั้นเชื่อมโยงกันหลายแพ็คเก็ต โดยเฉพาะ ทีซีพี ซึ่งจะมีลำดับของการติดต่อสื่อสารที่สัมพันธ์กันในแต่ลำแพ็คเก็ต การพิจารณาแพ็คเก็ตใดแพ็คเก็ตหนึ่งโดยไม่พิจารณาแพ็คเก็ตอื่นที่เกี่ยวข้องก็อาจจะไม่สามารถควบคุมแทรฟฟิกของ ทีซีพี ได้นอกจากนี้ยังรวมไปถึงการที่สเตตฟูลไฟร์วอลมีความสามารถในการประกอบรวมแฟร็กเมนต์เข้าด้วยกันให้เป็นดาต้าแกรมที่สมบูรณ์ ก่อนหลังจากนั้นจึงนำดาต้าแกรมนั้นมาทำการตรวจสอบเปรียบเทียบกับแอคเซสรูล

นอกจากการเชื่อมโยงกันของหลายแพ็คเก็ตสำหรับแพ็คเก็ตโปรโตคอล ทีซีพี ในทรานสปอร์ตเลเยอร์แล้ว ในแอพพลิเคชั่นเลเยอร์ก็มีแอพพลิเคชั่นบางชนิดที่จะต้องอาศัยการพิจารณาแทรฟฟิกอย่างต่อเนื่องเพื่อที่จะนำมากำหนดเป็นแอคเซสรูล ยกตัวอย่างเช่น การทำงานของเอฟทีพี ซึ่งในระหว่างการทำงานของแอพพลิเคชั่นนั้นโฮสที่เป็นไคลเอนต์จะสามารถกำหนดพอร์ตชั่วคราวขึ้นมาเป็นเซิร์ฟเวอร์พอร์ตใช้สำหรับรับ-ส่งไฟล์ได้ โดยพอร์ตเหล่านี้จะปิดลงเมื่อการรับ-ส่งข้อมูลเสร็จสิ้นสมบูรณ์ ซึ่งในกรณีนี้หากไม่มีการพิจารณาแทรฟฟิกที่มีมาก่อนหน้าแล้วไฟร์วอลอาจจะถือได้ว่าการเปิดให้บริการใหม่ขึ้นมาก็ได้ดังนั้นสเตตฟูลไฟร์วอลจึงมีการทำงานที่ค่อยข้างใกล้ชิดกับแอพพลิเคชั่นได้ค่อนข้างดี เพราะแอพพลิเคชั่นที่ใช้งานอยู่ในเน็ตเวิร์กไม่ได้มีเฉพาะแอพพลิเคชั่นพื้นฐานเท่านั้น มีแอพพลิเคชั่นอื่นๆ อีกมาก แต่หากแอพพลิเคชั่นใดมีการใช้งานอย่างแพร่หลาย และเป็นที่นิยมของผู้ใช้ โดยส่วนใหญ่ผู้ผลิตจะใส่บิวต์อิน การควบคุมแทรฟฟิกสำเร็จรูปมาให้อยู่ในไฟร์วอลเลย

2.4.4 ข้อดีของสเตตฟูลไฟร์วอล

1. ประสิทธิภาพในการทำงานสูง เนื่องจากออกแบบมาทำหน้าที่ไฟร์วอลโดยเฉพาะ สารมารถรองรับแอคเซสรูลที่ซับซ้อนได้ โดยที่ความสามารถในการทำงานไม่ลดลง

2. มีคุณสมบัติเพิ่มเติมให้ใช้ได้มากนอกเหนือจากการควบคุมแทรฟฟิก เช่นสามารถนำไปใช้ร่วมกับระบบการตรวจจับการบุกรุกหรือ IDS (Intrusion Detection System) เพื่อป้องกันการโจมตีได้อัตโนมัติ, สามารถบันทึกข้อมูลเอาไว้กลับมาดูภายหลังได้, สามารถใช้งานร่วมกับระบบป้องกันไวรัสได้เป็นต้น

3. การกำหนดแอคเซสรูลทำได้ง่ายเพราะไฟร์วอลมีความเข้าใจในโปรโตคอลระดับสูง ดังนั้นผู้ใช้อาจจะไม่จำเป็นต้องมีความเชี่ยวชาญในเรื่องระบบเครือข่ายมากนัก ก็พอจะใช้งานไฟร์วอลได้โดยกำหนดกฎพื้นฐานของแอพพลิเคชั่นที่ผู้ใช้รู้จัก มากกว่าการกำหนดกฎโดยใช้ข้อมูลบนแพ็คเก็ตโดยตรง เช่นแทนที่จะต้องกำหนดแอคเซสรูลให้อนุญาต ICMP Time exceed in Transit ให้ผ่านได้เพื่อใช้คำสั่ง Traceroute (ซึ่งโดยทั่วไปแล้วผู้ใช้ไม่ทราบว่าโปรแกรมใดใช้โปรโตคอลอะไร แต่จะรู้ว่าตนเองต้องการใช้โปรแกรมหรือแอพพลิเคชั่นอะไรบ้าง) ก็ระบุในไฟร์วอลว่าอนุญาตให้คำสั่ง Traceroute ทำงานได้หลังจากนั้นไฟร์วอลจึงกำหนดเป็นแอคเซสรูลที่ระบุโปรโตคอลนั่นเอง

4. สามารถเพิ่มเติมความปลอดภัยโดยระบบการตรวจสอบผู้ใช้ (Authenticate) ได้

5. การสื่อสารระหว่างไฟร์วอลกับแอดมินคอลโซล (Administration Console: เครื่องที่ทำหน้าที่ในการบริหารไฟร์วอล) จะมีความปลอดภัยสูง มีการตรวจสอบสิทธิ์ของผู้ที่เป็นแอดมินรวมทั้งการสื่อสารระหว่างไฟร์วอลกับคอลโซลจะมีการรักษาความปลอดภัยที่เข้มงวด มีการเข้ารหัสเพื่อนป้องกันการดักอ่านข้อมูล

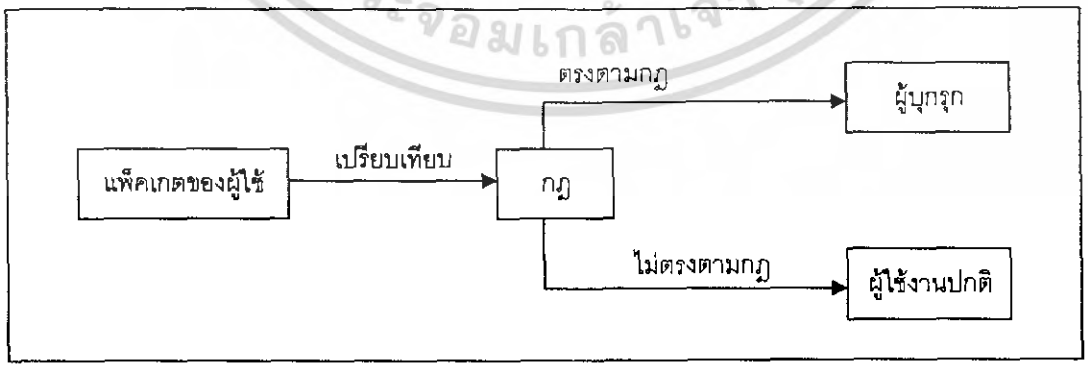
2.5 ระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

ระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์ (IDS : Intrusion Detection System) คือ ระบบที่ทำหน้าที่ติดตามดูการทำงานที่เกิดขึ้นบนระบบเครือข่ายคอมพิวเตอร์ เพื่อค้นหาร่องรอยที่บ่งบอกว่ากำลังมีผู้บุกรุกเข้ามาในระบบ หรือทำการค้นหาการกระทำที่เกินขอบเขตสิทธิ์ของผู้ใช้ระบบ

ผู้ดูแลระบบบางคนอาจคิดว่าระบบที่ตนดูแลอยู่นั้นไม่ได้มีข้อมูลที่สำคัญอะไร ถึงแม้ว่าจะโดนบุกรุกก็คงได้ไม่เป็นไร จึงละเลยและไม่ใส่ใจดูแลความปลอดภัยของระบบคอมพิวเตอร์ที่อยู่ในความรับผิดชอบของตนมากนัก แต่ในความเป็นจริงแล้วผู้บุกรุกนั้นอาจไม่ได้มีจุดประสงค์เพียงแค่บุกรุกเพื่อเข้ามาขโมยข้อมูลเพียงอย่างเดียว หากแต่อาจมีจุดประสงค์ในการที่จะใช้ระบบคอมพิวเตอร์ของเราในการโจมตีระบบอื่นต่อๆ ไปได้ และหากเป็นเช่นนั้น เมื่อมีการตรวจสอบจากระบบที่ถูกโจมตีกลับมาพบว่าการโจมตีนั้นๆ มาจากระบบของเราแล้ว ผู้ดูแลระบบจะต้องเป็นผู้รับผิดชอบความผิดดังกล่าวไม่ว่าจะเป็นความผิดเล็กน้อยหรือใหญ่หลวงเพียงใด อย่างหลีกเลี่ยงไม่ได้ ดังนั้นการรักษาความปลอดภัยของระบบของตนจึงเป็นสิ่งที่จำเป็นอย่างยิ่ง

2.5.1 รูปแบบของระบบตรวจจับผู้บุกรุกระบบคอมพิวเตอร์

วิธีเปรียบเทียบพฤติกรรมผู้ใช้กับวิธีการบุกรุกที่ระบบรู้จักเป็นวิธีการที่ระบบจะทำการเปรียบเทียบพฤติกรรมการใช้งานของผู้ใช้ว่าตรงกับรูปแบบการบุกรุกที่ระบบรู้จักหรือไม่ โดยระบบส่วนใหญ่จะใช้กฎ (Rule-base expert system) โดยตั้งกฎตามพฤติกรรมที่น่าสงสัยว่าจะเป็นการบุกรุก เช่น การตั้งกฎว่าเมื่อพบการเชื่อมต่อมายังพอร์ตหมายเลข 21 ซึ่งเป็นพอร์ตของบริการ FTP โดยใช้บัญชีผู้ใช้เป็น root เป็นต้น ซึ่งการทำเช่นนี้เข้าข่ายน่าสงสัย เพราะโดยปกติแล้วบัญชีผู้ใช้นี้มักจะถูกปิดการใช้งานไว้



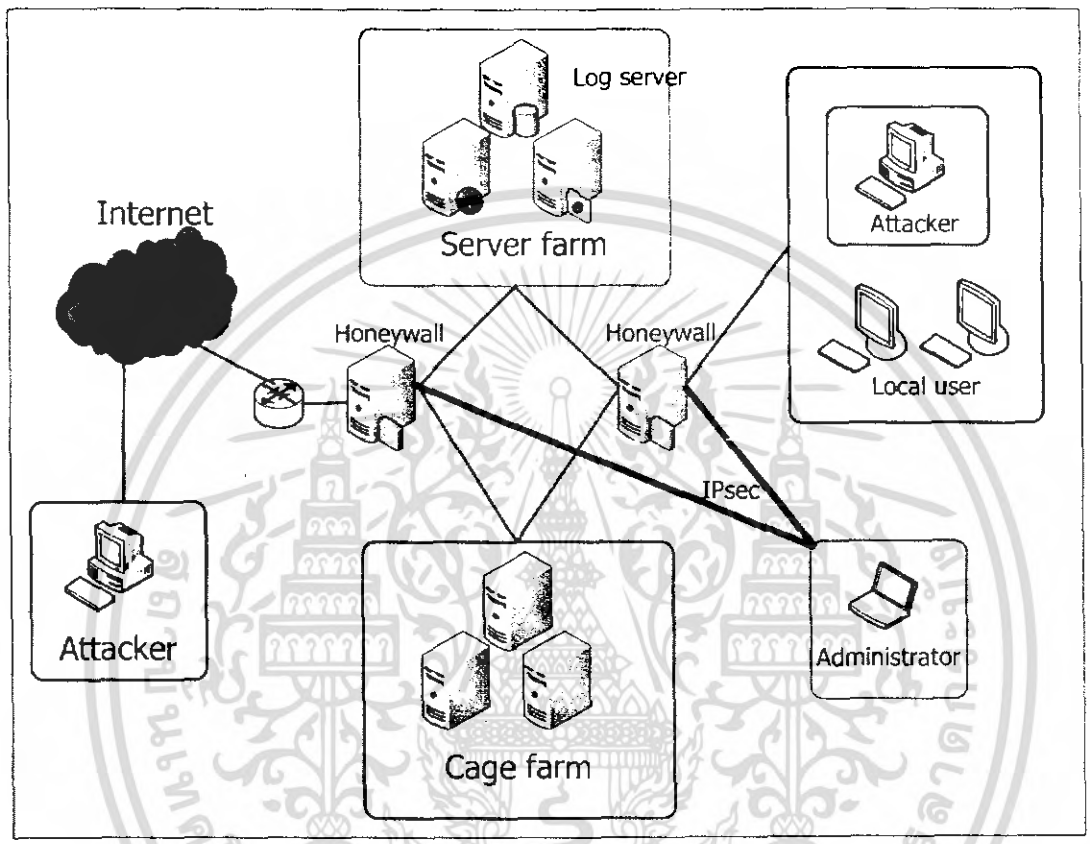
รูปที่ 2.1 แสดงการทำงานของระบบตรวจจับผู้บุกรุก

จากรูปแพ็คเกตที่มาจากการใช้งานเครือข่ายของผู้ใช้ต่างๆ จะถูกนำมาเปรียบเทียบกับกฎของระบบตรวจจับผู้บุกรุกถ้าตรงกับกฎก็จะถือว่าผู้ใช้นั้นเข้าข่ายเป็นผู้บุกรุกระบบ แต่ถ้าไม่ตรงกับกฎก็จะถือว่าเป็นผู้ใช้ปกติสามารถทำงานในระบบได้ต่อไปตามปกติ

2.5.2 จุดอ่อนของวิธีการเปรียบเทียบพฤติกรรมผู้ใช้กับวิธีการบุกรุกที่ระบบรู้จักนั้นคือ

1. ประสิทธิภาพในการตรวจจับผู้บุกรุกนั้นจะขึ้นอยู่กับการเลือกใช้กฎที่ใช้เปรียบเทียบ
2. หากตั้งกฎได้ไม่รัดกุมพอ เมื่อผู้ใช้ปกติทำพฤติกรรมที่ตรงกับกฎของระบบตรวจจับผู้บุกรุกโดยไม่ได้ตั้งใจระบบตรวจจับอาจเข้าใจผิดว่าผู้ใช้นั้นๆ เป็นผู้บุกรุกได้
3. รูปแบบการบุกรุกที่ไม่รู้จักจะไม่สามารถถูกตรวจพบได้

บทที่ 3
ระบบฮันนี่พ็อต



รูปที่ 3.1 แสดงโครงสร้างของโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก

3.1 องค์ประกอบของระบบฮันนี่พ็อต

3.1.1 Honeywall

ทำหน้าที่เป็นเครื่องGATEWAY ที่ข้อมูลการร้องขอหรือการกระทำใดๆต้องผ่านก่อนที่จะได้รับการให้บริการ โดยจะทำงานในรูปแบบ Bridge Mode นั่นคือผู้ใช้งานจะไม่ทราบว่าการใช้งานเครื่องให้บริการได้ส่งข้อมูลผ่านตัว Honeywall แล้วจึงค่อยผ่านไปยังเครื่องให้บริการ ซึ่งการตรวจสอบและคัดแยกผู้บุกรุกออกจากผู้ใช้ปกติ หรือการดักจับโค้ดหนอน ก็จะเกิดขึ้นในตัว Honeywall นี้เอง ตัวHoneywallได้อาศัยระบบ IDS (Intrusion Detection System) เพื่อให้มีความสามารถที่ได้กล่าวมาแล้ว และเมื่อตรวจพบว่าการกระทำเข้าข่ายการบุกรุก ตัว Honeywall จะทำ

การเปลี่ยนให้ผู้บุกรุกไปใช้งานเครื่องกับดักแทน ซึ่งได้เตรียมรอไว้แล้ว และมีสภาพแวดล้อมและค่าต่างๆเหมือนเครื่องให้บริการจริง และตัว Honeywall ก็จะส่งข้อมูลไปให้กับเครื่อง Log server

และที่ตัว Honeywall นี้เองจะเป็นตัวควบคุมระบบทั้งหมด โดยผ่านทางโปรแกรมที่ผู้พัฒนาได้จัดทำขึ้น นั้นคือโปรแกรมTM (Tartarus management) ซึ่งโปรแกรมนี้สามารถควบคุมจัดการและเรียกข้อมูลทุกอย่างได้ โดยเรียกจากโปรแกรมที่ผู้พัฒนาได้สร้างขึ้นและฝังการทำงานไว้ในส่วนต่างๆ นอกจากจะสามารถเรียกใช้โปรแกรมควบคุมและจัดการนี้ได้โดยตรงจากเครื่องHoneywall แล้ว ทางผู้พัฒนายังได้จัดเตรียมช่องทางที่ผู้ดูแลระบบจะสามารถเข้ามาบริหารจัดการได้ผ่านทางช่องทางที่ปลอดภัยโดยใช้ IPsec (IPsecurity) ซึ่งจะมีการAuthentication และ Encryption ทำให้สามารถจัดการระบบผ่านเครือข่ายได้ด้วย

3.1.2 เครื่องกับดัก (Cage)

ทำหน้าที่จำลองตัวเองให้เหมือนเครื่องให้บริการมากที่สุด และอาจมีการเปิดบริการเสริมหลอกล่อไว้ด้วย เพื่อให้ผู้บุกรุกหลงเชื่อและแสดงพฤติกรรมในการบุกรุกต่างๆ โดยการกระทำต่างๆ ไม่ว่าจะเป็นการกดแป้นพิมพ์ส่งคำสั่งเข้ามา การดาวน์โหลดหรือการนำข้อมูลเข้ามาในตัวเครื่องกับดัก หรือการนำข้อมูลออก จะถูกบันทึกพฤติกรรมไว้หมด ไม่ว่าจะส่งคำสั่งนี้จะใช้ผ่านทางช่องทางที่มีการเข้ารหัสก็ตาม เพราะเครื่องมือที่ใช้บันทึกพฤติกรรมทำงานในรูปแบบของ kernel-base และสามารถซ่อนตัวเองไม่ให้ถูกพบว่ากำลังทำงานอยู่ นอกจากนี้เครื่องกับดักจำเป็นที่จะต้องมีการรักษาความปลอดภัยในด้าน การถูกนำไปใช้เป็นฐานในการโจมตีเครื่องอื่นๆ โดยได้มีการตรวจสอบสภาพและความเสียหายอยู่ตลอดเวลา หากถึงจุดที่เครื่องกับดักมีความอ่อนแอถึงจุดที่กำหนด (ซึ่งผู้ดูแลระบบสามารถกำหนดได้ตามความสมควร) เครื่องกับดักก็จะถูกระงับการใช้งานโดยอัตโนมัติ

3.1.3 เครื่องแม่ข่ายฐานข้อมูล (Log server)

ทำหน้าที่เก็บบันทึกรายละเอียดที่ส่งเข้ามาจากที่ต่างๆ ลงในฐานข้อมูลอย่างมีระบบ ซึ่งข้อมูลที่ส่งมา จะมาจากทุกส่วน เช่น Honeywall, Cage ซึ่งเครื่องแม่ข่ายฐานข้อมูลนี้ก็จะอยู่ในส่วนที่ปลอดภัยจากการบุกรุกเข้ามา คืออยู่ภายใน DMZ ของระบบ ข้อมูลที่ถูกจัดเก็บในเครื่องนี้ก็ได้แก่ การกระทำของผู้บุกรุกต่างๆ, ไฟล์ที่ผู้บุกรุกได้นำเข้าหรือเอาออกจากระบบ, ข้อมูลการเปลี่ยนแปลงและสภาพของเครื่องกับดัก, ไฟล์อิมเมจของเครื่องกับดักที่ถูกระงับการใช้งาน เป็นต้น

3.2 หน้าที่การทำงานในแต่ละส่วน และเครื่องมือที่ใช้

การทำงานของโปแกรมจะแบ่งออกเป็น 5 ส่วนใหญ่ๆ ดังนี้

- จำแนกผู้บุกรุกออกจากผู้ใช้งานทั่วไปและส่งเข้าใช้งานเครื่องกับดัก
- ส่วนเฝ้าบันทึกพฤติกรรม และควบคุมขอบเขตของผู้บุกรุก
- ส่วนจัดเก็บข้อมูลกลาง
- ส่วนการจัดการและตรวจสอบสภาพความเสียหายของเครื่องกับดัก
- ส่วนการควบคุมและแสดงผลการบันทึกข้อมูลต่างๆของระบบ

3.2.1 จำแนกผู้บุกรุกออกจากผู้ใช้งานทั่วไปและส่งเข้าใช้งานเครื่องกับดัก

ในส่วนนี้จะรับผิดชอบโดยตัวHoneywall โดยใช้การจัดจำแนกแบบ Rules-base

3.2.1.1 การจำแนกว่าเป็นผู้บุกรุก

ในการจำแนกผู้บุกรุกนั้นจะพิจารณาว่าผู้ใดเป็นผู้บุกรุกหรือเข้าข่ายว่าเป็นผู้บุกรุก โดยใช้กฎของโปรแกรมตรวจจับผู้บุกรุก (Snort_inline) ซึ่งเป็นเครื่องมือประเภท IDS (Intrusion Detection System) โดยหากว่าการเชื่อมต่อใดมีลักษณะตรงตามกฎของโปรแกรมตรวจจับผู้บุกรุกจะถือว่าเจ้าของการเชื่อมต่อนั้นเข้าข่ายเป็นผู้บุกรุกจะต้องทำการเปลี่ยนทิศทางการเชื่อมต่อดังกล่าวไปยังเครื่องกับดักเพื่อดำเนินการบันทึกพฤติกรรมของผู้บุกรุกนั้นและเพื่อให้การจำแนกผู้บุกรุกของโปรแกรมตรวจจับผู้บุกรุกเป็นไปตามความต้องการของผู้พัฒนามากขึ้น ทางผู้พัฒนาจึงได้ทำการแก้ไขกฎการจำแนกบางส่วน เช่น Mysql cazz exploit

กฎของการตรวจสอบการบุกรุกของเครื่องมือที่ใช้เข้าโจมตีโปรแกรมฐานข้อมูล ที่ชื่อว่า มายเอสคิวแอล (mysql) โดยชื่อโปรแกรมที่เรียกว่า cazz เพื่อเข้าโจมตีแบบยึดครองระบบ เพื่อตั้งให้กฎเองมีความยืดหยุ่นที่ว่าหากเข้ามาเชื่อมต่อเข้ามายังช่องทางหมายเลข สามสามศูนย์หก ลักษณะการขอการสถาปนา ก็ให้ถือว่าเป็นการโจมตีจากผู้บุกรุก ซึ่งมีตัวอย่างดังนี้

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306
(msg:"MYSQL root login attempt"; flow:to_server,established;)
```

3.2.1.2 การจำแนกว่าเป็นผู้ใช้ปกติ

ในการจำแนกผู้ใช้ปกตินั้นจะพิจารณาโดยใช้กฎของโปรแกรมตรวจจับผู้บุกรุกเช่นเดียวกับการจำแนกผู้บุกรุก คือการเชื่อมต่อใดที่ไม่ตรงกับกฎให้ถือว่าเจ้าของการเชื่อมต่อนั้นๆ เป็นผู้ใช้ปกติสามารถให้ผ่านไปใช้บริการของเครื่องแม่ข่ายจริงได้ตามปกติ และกรณีที่ไม่ต้องการให้

มีการตรวจสอบเครื่องบางเครื่องที่เชื่อมต่อเข้ามา สามารถทำการกำหนดกฎที่ ไฟร์วอลให้การเชื่อมต่อนั้นสามารถผ่านเข้าไปได้โดยไม่ต้องมีการตรวจสอบ เนื่องจากว่าผู้ที่ควบคุมการเชื่อมต่อจริงแล้วคือไฟร์วอล ในการตรวจสอบของโปรแกรมตรวจจับผู้บุกรุกอาจมีผิดพลาดได้จึงจำเป็นต้องตั้งกฎของระบบตรวจจับผู้บุกรุกให้มีความรัดกุมและบ่งบอกชัดเจนว่าผู้ใดเป็นผู้บุกรุก เพื่อให้กระบวนการตรวจสอบนั้นไม่เกิดผลการเข้าใจผิดว่า ผู้ใช้งานปกตินั้นเป็นผู้บุกรุก

3.2.1.3 การจำแนกว่าเป็นโค้ดหนอน

ในการระบุได้ว่าเป็นโค้ดหนอน จะใช้การพิจารณาแบบ Rules-base เช่นเดียวกับการตรวจจับผู้บุกรุก โดยจะใช้ฐานข้อมูลที่เก็บSignatureของโค้ดหนอนชนิดต่างๆเอาไว้ มาเปรียบเทียบรูปแบบที่พบ ซึ่งจำเป็นจะต้องมีการปรับปรุงฐานข้อมูลSignatureนี้ให้ทันสมัยอยู่ตลอดเวลา ซึ่งทางผู้พัฒนาได้เพิ่มเข้าไปในส่วนของ Snort_inline เพื่อให้สามารถตรวจสอบโค้ดหนอนไปพร้อมกับตรวจจับผู้บุกรุก

3.2.1.4 เครื่องมือที่ใช้ในส่วนของ Honeywall

ทางผู้พัฒนาได้ประยุกต์เครื่องมือต่างๆให้สามารถใช้งานและติดต่อส่งผ่านคำสั่งถึงกันได้ ทำให้ได้ความสามารถตามที่ต้องการ

Iptables

ไอพีเทเบิลนั้นเป็นโปรแกรมไฟร์วอลแบบState Full Inspector ที่ทำงานใน Kernel-Mode โดย iptables จะมีหน้าที่ในการจัดการเชื่อมต่อ และการอนุญาตหรือไม่อนุญาตให้ชิ้นข้อมูลต่างๆ สามารถผ่านเข้าออกระบบเครือข่ายได้ ซึ่งในโครงงานนี้ได้นำโปรแกรม iptables มาใช้งานเพื่อให้ชิ้นข้อมูลที่ผ่านเข้ามาในระบบเครือข่ายถูกส่งไปให้ส่วนที่โปรแกรมตรวจจับผู้บุกรุกที่ทำงานในยูสเซอร์โหมดนำชิ้นข้อมูลเหล่านั้นไปพิจารณาต่อไป โดยชิ้นข้อมูลจะถูกส่งผ่านไปทางโมดูลที่ชื่อว่า ip_queue และอีกหน้าที่หนึ่งของ iptables คือการจัดเส้นทางการเชื่อมต่อให้กับผู้ใช้ที่ถูกจำแนกแล้วว่าเป็นผู้ใช้ปกติหรือเป็นผู้บุกรุกระบบ ให้เข้าไปใช้งานเครื่องให้บริการจริง หรือเครื่องกับดัก โดยตัวไอพีเทเบิลจำเป็นจะต้องทำงานร่วมกับ Snort_inline โดยผ่านการเชื่อมให้สามารถทำงานประสานกันได้โดย S2I

snort_inline

สนอร์ตอินไลน์เป็นโปรแกรมไอดีเอสที่ทำงานในยูสเซอร์โหมด ซึ่งจะทำหน้าที่เป็นระบบตรวจจับผู้บุกรุก ในโครงงานนี้จะนำ snort_inline มาใช้ในการจำแนกว่าผู้ใดเป็นผู้ใช้งานปกติและผู้ใดเป็นผู้บุกรุกระบบ ซึ่งจะจัดจำแนกตามข้อมูลหรือกฎที่ผู้ดูแลระบบใช้ ซึ่งในที่นี้โปรแกรมจะทำการอ่านค่าชิ้นข้อมูลที่ส่งมาทางมอดูล ip_queue แล้วนำมาเปรียบเทียบกับกฎที่ได้ตั้งไว้ ว่าเข้าข่ายเป็นการโจมตีหรือไม่ และทำการแจ้งเตือนในกรณีที่การตรวจสอบมีผลออกมาว่าชิ้นข้อมูลที่ส่งมามีลักษณะของการโจมตีระบบ

ซึ่งตัวสนอร์ตอินไลน์สามารถนำมาประยุกต์เพิ่มกฎที่เหมาะสมให้สามารถดักจับโค้ดหนอนที่แพร่ระบาดเข้ามายังระบบได้ โดยเพิ่มในส่วนของ Signature เข้าไปเพื่อใช้เป็นฐานข้อมูลให้ตัวโปรแกรมสามารถรู้จักและสามารถแยกแยะว่าเป็นโค้ดหนอนได้

S2I

เอสทูไอ (S2I: snort command to iptables) เป็นโปรแกรมที่สร้างมาจากภาษาเพิร์ลเพื่อให้สามารถแปลงการแจ้งเตือนที่ได้มาจากการไปคิวรี่กับฐานข้อมูล ไปเป็นคำสั่งของโปรแกรม iptables เพื่อโปรแกรมไฟร์วอลจัดเส้นทางให้กับชิ้นข้อมูลที่มาจากผู้ใช้งานปกติผ่านไปยังส่วนที่ส่วนที่เป็นเซิร์ฟเวอร์จริง และในส่วนของผู้บุกรุกเองนั้นก็ให้ส่งไปยังเครื่องกับดักที่ได้เตรียมไว้ นั่นก็คือเป็นตัวกลางที่จะทำให้โปรแกรม Snort_inline สามารถทำงานประสานกับโปรแกรมIptablesได้ ซึ่งสาเหตุที่ต้องมีเอสทูไอนั้นเนื่องมาจากโปรแกรมSnort_inline และโปรแกรมIptables ที่ใช้ในโครงงานนี้นั้นทำงานในโหมดที่ต่างกันจึงติดต่อกันเองโดยตรงไม่ได้ ดังนั้นจึงต้องมีโปรแกรมสื่อกลางช่วยในการติดต่อระหว่างกัน

TM (Tartarus Management)

Tartarus Management เป็นโปรแกรมในรูปแบบ GUI ใช้งานง่ายไม่จำเป็นต้องจดจำคำสั่งของแต่ละเครื่องมือ ซึ่งมีเป็นจำนวนมากในระบบ โดยเป็นโปรแกรมที่ใช้ควบคุมการทำงานทั้งหมดของ Honeypot Program Suite ซึ่งได้ช่วยแก้ปัญหาความยุ่งยากในการใช้งานระบบฮันนี่พ็อตที่ผ่านมาได้ เพราะได้มีรวมการบริหารทั้งหมด ทั้งในส่วนของการดูแลและจัดการเครื่องมือ การตั้งค่าการใช้งานต่างๆ เช่นค่าการใช้งานของ Snort_inline ทั้งในส่วนของserverและclient, ค่าการใช้งาน sebekทั้งในส่วนของserverและclient, ค่าการใช้งานของ samhainทั้งในส่วนของserverและclient, รวมทั้งการจัดการเครื่องที่ทำหน้าที่เป็นCage ทั้งการตรวจเช็คสภาพและการปรับเปลี่ยน ทุกอย่างมาไว้ในโปรแกรมควบคุมจัดการนี้

นอกจากนี้ยังใช้ในการเรียกแสดงข้อมูลทุกอย่างที่มีภายในระบบ เช่น ข้อมูลที่เก็บพฤติกรรมผู้บุกรุก หรือข้อมูลสภาพความเปลี่ยนแปลงของเครื่องกับดัก เป็นต้น

3.2.2 ส่วนเฝ้าบันทึกพฤติกรรม และควบคุมขอบเขตของผู้บุกรุก

ในส่วนนี้มีชื่อเรียกว่า "Cage" จะเป็นส่วนของกับดักหรือระบบหลอกที่ใช้เพื่อให้ผู้บุกรุกเข้ามาติดกับและเฝ้าคอยดูพฤติกรรมของผู้บุกรุกที่อยู่ในเครื่องกับดัก และมีการใช้การควบคุมข้อมูล Data control เพื่อความปลอดภัย นอกจากนี้ยังได้พัฒนาด้านความแนบเนียนของเครื่องกับดักให้มีสภาพแวดล้อมและค่าต่างๆ ให้เหมือนกับเครื่องที่ใช้งานจริง และมีความปลอดภัยไม่ให้เครื่องกับดักถูกใช้เป็นฐานในการโจมตีเครื่องอื่นๆ

Data control

Data control จะป้องกันเหล่าผู้โจมตีจะใช้ตัว honeypot เพื่อโจมตีหรือ ทำอันตรายในเครื่องอื่นๆที่ไม่ใช่ระบบHoneypot ตัว Data control จะลดอัตราเสี่ยง โดยมันทำการจำกัด จำนวนข้อมูลที่ส่งออก โดยใช้ NIPS(Network Intrusion Prevention System) ต่อเข้ากับตัวนับเพื่อนับว่าหากถึง limit ที่เข้ามาจำนวนมาก honeypot สามารถกำหนดได้ NIPSจะสามารถ block รูปแบบการกระทำได้ รวมทั้ง2 อย่างนี้เข้าไว้ด้วยกันจะทำให้เกิดความยืดหยุ่นที่ data control โดยติดตั้งไว้ที่ เครื่อง Honeywall เพื่อที่จะได้ควบคุมทั้งขาเข้าและขาออก โดยเริ่มแรกผู้ดูแลระบบจะต้องตัดสินใจว่าจะยอมรับให้มีการติดต่อเข้ามามากน้อยเพียงใดต่อช่วงเวลา โดยจะนับการ connect ที่มาจากภายนอก และจะให้ limit เท่าใดเพื่อป้องกันการ connect ที่จะมีมากมายเป็นการลดอัตราเสี่ยงของการ scan การโจมตีหรือการ denial of service ซึ่งยากที่จะป้องกันไม่ให้เกิดโจมตีจำนวนมาก แต่การจำกัดจำนวนของแพ็คเกตก็มีข้อเสียตรงที่ผู้บุกรุกอาจ สังเกตได้ว่า ถ้าหากพวกเขาถูก block หลังจากทำการส่ง packet เข้ามาเป็นจำนวนหนึ่ง เช่นจะถูก block เมื่อส่งpacket ที่ 11 ก็อาจทำให้ผู้บุกรุกทราบได้ว่ามีการทำ Data control เอาไว้ ค่าพื้นฐานสำหรับการเชื่อมต่อจะกำหนดได้ใน re.firewall ตัวค่าvariable อื่นๆ ที่เป็น IP Protocol มีNOT,TCP,UDP หรือ ICMP (เช่น IPsec,IPv6,tunneling,Network Voice Protocol และอื่นๆ) ตัวอย่างการกำหนดจำนวนpacket ที่เข้าออกเป็นดังนี้

Set the connection outbound limits for different protocols

SCALE="day"

TCPRATE="15"

UDPRATE="20"

ICMPRATE="50"

OTHERRATE="15"

มีการสนับสนุนจาก limit กับ rc.firewall ซึ่งขั้นต่อไปคือการทำให้สนับสนุน NIPS ซึ่งเป็น function ที่ว่าเป้าหมายของ NIPSคือการจำแนก และ block เมื่อรู้ว่านี้คือการโจมตี มันทำหน้าที่นี้โดยจะสอดส่องดู Packet อื่นๆที่ผ่านเข้ามายัง Honeywall ถ้าตัว packet นั้น match กับ IDS rules มันจะไม่เพียงแค่สร้างสัญญาณเตือน(เหมือน NIDS) แต่ packet จะถูก drop (ป้องกันการโจมตี) หรือเปลี่ยนแปลง(หยุดการโจมตี) จะมีประโยชน์ที่จะลดอัตราเสี่ยงของการโจมตีขาออกได้

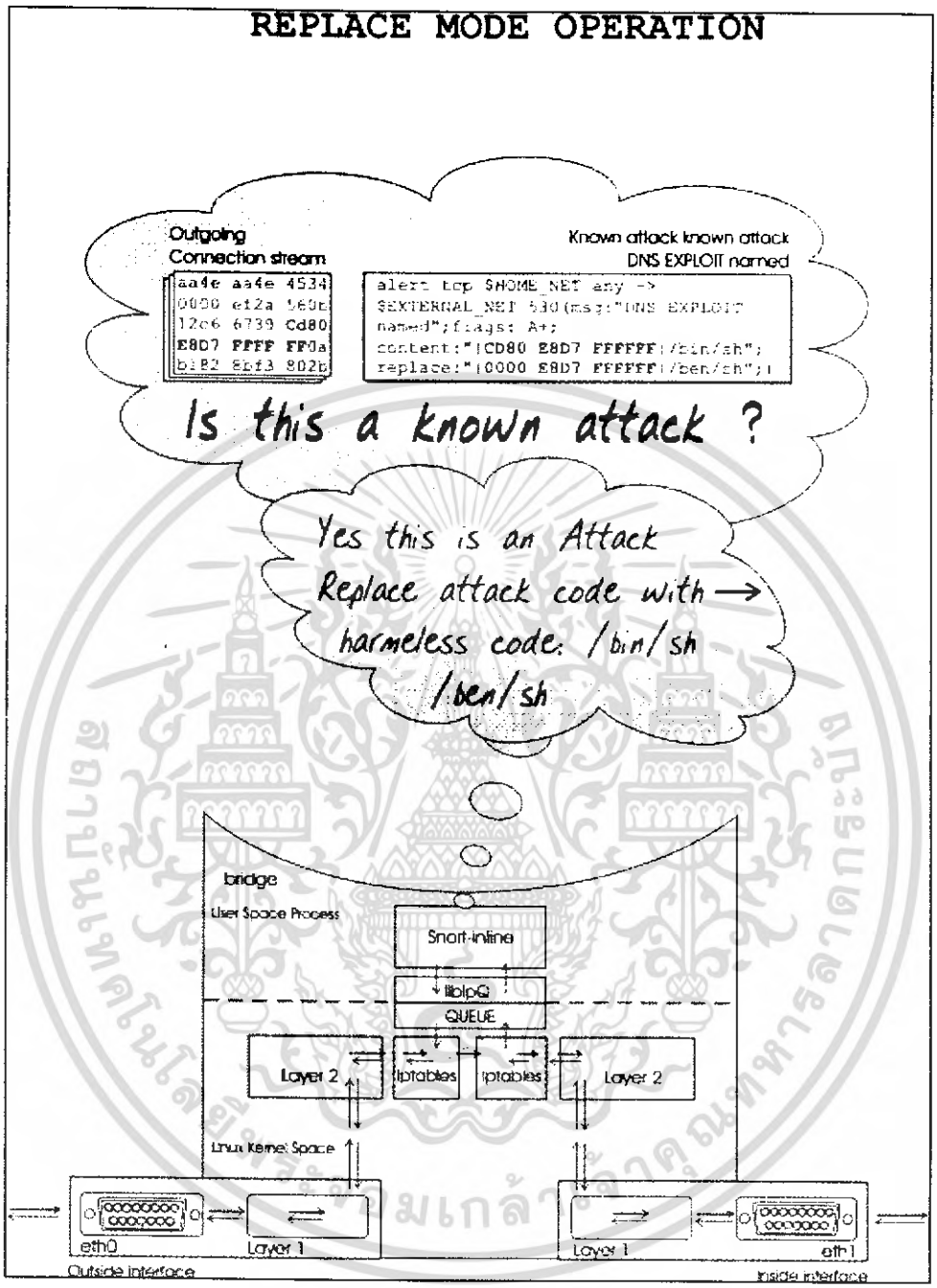
IPtables script can be used with the snort_inline filter

#QUEUE="no" **#Do not use experimental QUEUE support**

QUEUE="yes" **#Use experimental QUEUE support**

เราได้ตั้งค่าให้ยินยอม ให้เครื่องกับดักส่งข้อมูลออกได้ หรือทำการส่ง ICMP, Ping เพื่อเฝ้าดูพฤติกรรม เนื่องจากหากทำการใช้กฎที่มีทั้งหมด จะทำให้ผู้บุกรุกไม่สามารถทำอะไรได้เลย การใช้กฎของ snort เพื่อเป็นการป้องกันการโจมตีนั้น ในองค์กรต่างๆ อาจจะมีความต่างของการกำหนดว่าอะไรคือการโจมตี ดังนั้นพวกทางผู้พัฒนามุ่งเน้น ด้านการดัดแปลงตัวกฎของ snort_inline ให้ทำงานได้ตามที่ต้องการ ก่อนที่จะนำมากำหนดใช้ ดังนั้นกฎที่เราใช้จะมีส่วนที่กลับกันจากรูปแบบที่เป็นกฎทั่วๆ ไปของตัว snort_inline ที่เป็นกฎพื้นฐานโดยที่จะจับตาที่การโจมตีขาเข้า แต่ทางระบบฮันนี่พ็อต จะเพิ่มการจับตาดูที่การโจมตีขาออก เป้าหมายของการป้องกันจากโลกภายของจากระบบฮันนี่พ็อต

จะตั้งกฎเพื่อให้วัตถุประสงค์ที่เป็นส่วนของการวิเคราะห์ตัว packet และ ถ้าหากเห็นว่าเป็นการโจมตีจากภายนอก การ block หรือ drop ตัว packet จะบรรจุการโจมตีไว้ ทำการ block หรือ drop ตัว packet ดูรูป honeypot3 ที่บรรจุการโจมตีจะสามารถเห็นรูปแบบนี้ได้จากตัวอย่างของการ drop ตัว Code Red II ซึ่งเป็นการโจมตีที่เปรียบเทียบให้เห็นง่ายๆ โดยการใช้วิธีการวางทับ replace ruleset จะไม่ทำการ block ดังรูปแบบการdrop ของระบบตรวจจับผู้บุกรุก



รูปที่ 3.2 แสดงรูปแบบการวางทับข้อมูลในแพ็คเก็ตของระบบตรวจจับผู้บุกรุก

การแทนที่ในส่วนนี้คือการดัดแปลงตัว Contents ต่างๆ ที่เป็นการโจมตีในปัจจุบัน หยุดยั้งการเข้ายึดระบบ exploit นี้เป็นส่วนแยกของความต่างที่เป็นการควบคุมการกระทำสำหรับตรวจจับ attacker พวกเขาจะเห็น Attacker ต่างๆ นั้นไปถึงในเป้าหมายที่ได้เจตนาที่จะไปไว้ แต่ก็ไม่สามารถ

แสดงออกมาได้ว่าทำไม Attacker จึงพลาดในการกระทำนั้น ตัวเครื่องมือ snortconfig เป็นตัว script ที่ช่วยทำให้เกิดความยืดหยุ่นนี้กลับกันกับจากกฎพื้นฐานของตัว snort_inline ที่สามารถทำได้ เช่น การ drop การ sdrop หรือ การ replace โดยที่เครื่องมือนี้มีความสำคัญเป็นอันมากรักษากฎของกฎที่ได้จัดเตรียมไว้ของ snort_inline

สำหรับวัตถุประสงค์ของ แปลงของทางระบบฮันนี่พ็อต จะติดตั้งเป็นแบบ Drop ที่การจัดตั้งกฎ และ ในส่วน snort_inline.conf โดยการเข้าไป configure file ที่ /etc/snort_inline ซึ่งได้ใช้การแบ่งตัวแฟ้มข้อมูลจาก /etc/snort ดังนั้นจะไม่ทำให้สับสน ในการเริ่มต้นกับตัว snort_inline โดยจะใช้ snort_inline startup script ตัว script นี้เป็นตัวเริ่มต้นของการจัดค่าตัวแปรให้เหมาะสมกับ snort_inline ที่จะสามารถทำงานได้ จะช่วยให้การใช้งานง่ายขึ้น ซึ่งจะถูกรวมไว้ในชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก ในส่วนของโปรแกรม TM ไว้หมดแล้ว

Sebek_client

เซเบคไคล์เอ่นจะคอยซ่อนตัวอยู่ในกับดักเพื่อเฝ้าดูพฤติกรรมและการกระทำของผู้บุกรุก และคอยส่งข้อมูลนั้นไปยังเครื่องที่ใช้เก็บพฤติกรรมลงฐานข้อมูล โดยส่งผ่านช่องทางที่มีความปลอดภัย มีการเข้ารหัสข้อมูลในรูปแบบของ เซเบคเอง

เซเบคเป็นData Capture Toolแบบ Kernel-based โดยในการนำมาใช้จะประยุกต์ใช้เป็นแบบ LKM (Loaded Kernel Module) มีความโดดเด่นในความสามารถในการดักจับข้อมูลได้ทุกการกระทำของIntruderที่กระทำผ่าน read() system call ไม่ว่าจะเป็นในส่วนของ keystroke, files transfer, burneye passwords หรือหากมีการใช้ IRC clientหรือ e-mail client ตัวเซเบคก็สามารถดักจับได้ ดังเช่นเมื่อIntruderมีการนำfileส่งเข้ามา ตัวเซเบคก็สามารถอ่านและบันทึกไฟล์นั้นไว้ได้
ทั้งยังสามารถใช้เป็นGlass-box เมื่อเปรียบเทียบกับการทำงานของblack-boxที่เรารู้จัก นั้นคือเราสามารถใช้ติดตามการทำงานของโปรแกรมที่intruderเอามารันได้ด้วย แม้ว่าตัวintruder จะlog out ออกไปแล้ว

เซเบคมีรูปแบบวิธีการปิดซ่อนตัวเองได้อย่างมีประสิทธิภาพ และค่อนข้างยากที่จะตรวจสอบพบทั้งจากภายในและภายนอก โดยจะทำการปิดซ่อนโมดูลที่ใช้ และลบlinked list ที่จะแสดงการมีตัวตนออก โดยใช้ the cleaner ทำให้Intruder ไม่สามารถตรวจสอบพบว่าเซเบคกำลังทำงานอยู่ภายในเครื่อง ทุกpacketของเซเบค ตัวเซเบคจะมีfunctionในการสร้างและส่งเอง โดยไม่ใช้ TCP/IP stack ดังนั้นจึงไม่สามารถถูกมองเห็นและไม่สามารถblock traffic ของเซเบคได้ หลังจากสร้างpacketเสร็จpacketจะถูกส่งไปยังdevice driverโดยตรง ไม่ผ่าน raw socket socket interface ซึ่งพวกโปรแกรม sniffer ที่ใช้ libpcap เป็นฐานจะไม่สามารถเห็นได้เพราะมันไปดักจับข้อมูลที่ raw socket interface

และเพื่อป้องกันการตรวจจับในเครือข่าย scbekจะไม่ใช้ARP แต่จะกำหนดMAC Addressของserver ไว้เลย เพื่อไม่ต้องทำการ ARP request เพื่อป้องกันการทำ ARP spoofing ซึ่งก็จะลดการดักจับข้อมูลในเครือข่ายได้ ทั้งนี้เซเบคมีรูปแบบการส่งข้อมูลด้วยโปรโตคอล UDP

TMdaemon

TMdaemon เป็นเซอร์วิสที่ผู้พัฒนาได้สร้างขึ้นเพื่อคอยตรวจสอบสภาวะของเครื่องกับดักและจะทำการหยุดการทำงานของเครื่องกับดักโดยอัตโนมัติ หากเครื่องกับดักมีความอ่อนแอเกินกว่าที่จะสามารถควบคุมพฤติกรรมของผู้บุกรุกเอาไว้ได้ โดยการตรวจสอบข้อมูลดังกล่าวจะกระทำโดยใช้โปรแกรมที่เขียนขึ้นมาให้ไปอ่านค่าจากฐานข้อมูลต่างๆ เพื่อมาพิจารณาว่าตรงตามเงื่อนไขที่กำหนดไว้หรือไม่ ถ้าตรงก็จะทำการหยุดการทำงาน (suspend) ซึ่งจุดที่จะบ่งบอกว่าเครื่องกับดักจะถูกหยุดการทำงานผู้ดูแลระบบสามารถตั้งค่าได้ตามความเหมาะสม โดยเรียกใช้งานผ่านโปรแกรมควบคุม Tartarus Management (TM)

portsentry

พอร์ตเซนทรีจะเป็นตัวที่ทำหน้าที่ในการจำลองบริการที่ระบบจริงไม่ได้เปิดให้บริการอยู่ เพื่อเป็นการล่อหลอกผู้บุกรุกในกรณีที่ผู้บุกรุกร้องขอการเชื่อมต่อนั้นๆ เข้ามายังระบบ ซึ่งจะทำให้ผู้บุกรุกเชื่อว่ามีการเปิดบริการดังกล่าวอยู่ และเริ่มทำการโจมตี จากนั้นฮันนี่พ็อตก็จะทำการบันทึกพฤติกรรมต่างๆ ของผู้บุกรุกไว้

Samhain (samhain)

เซมเฮนเป็นเซอร์วิสที่ทำหน้าที่ในการตรวจสอบความถูกต้องของไฟล์สำคัญๆ ในระบบว่าผู้บุกรุกได้ทำการเปลี่ยนแปลงอะไรไปบ้าง เช่นไฟล์ใดถูกแก้ไข ไฟล์ใดโดนลบ โดยโปรแกรมจะทำการบันทึกการเปลี่ยนแปลงของไฟล์ต่างๆ และส่งไปเก็บลงในฐานข้อมูลในเครื่อง Log server ซึ่งทำให้เราทราบความเปลี่ยนแปลงของระบบได้ อีกทั้งตัวโปรแกรมนี้สามารถทำการซ่อนตัวจากการตรวจสอบจากผู้บุกรุกได้ด้วยในระดับหนึ่ง

3.2.3 ส่วนจัดเก็บข้อมูลกลาง (Log server)

จะทำการเก็บรวบรวมข้อมูล จัดเก็บลงฐานข้อมูล รับผิดชอบโดยเครื่องแม่ข่ายฐานข้อมูล (Log server) ซึ่งสิ่งที่ต้องพิจารณาในการจัดตั้งLog server คือ ความปลอดภัยของตัวเครื่องLog server เอง และรูปแบบการส่งข้อมูลเข้ามา เนื่องจากหากรูปแบบที่ใช้ในการส่งข้อมูลเข้ามาเป็นการส่งข้อมูล

แบบธรรมดา ผู้บุกรุกอาจใช้เครื่องมือช่วยในการดักจับ และทราบถึงการมีอยู่ของระบบฮันนี่พ็อต และอาจก่อให้เกิดปัญหาด้านความปลอดภัยของระบบจริง

ข้อมูลสำคัญที่จะทำการเก็บลงฐานข้อมูล เพื่อสามารถนำไปวิเคราะห์

1. ข้อมูลเบื้องต้นของผู้บุกรุก เช่น หมายเลขไอพีแอดเดรส
2. ข้อมูลการป้อนคำสั่ง หรือ keystroke ของผู้บุกรุก
3. ข้อมูลที่นำเข้า หรือ อาจเป็นไฟล์หรือเครื่องมือที่ผู้บุกรุกจะนำมาใช้
4. ข้อมูลที่ส่งออกจากเครื่องกับดัก อาจเป็นข้อมูลหรือเป็นการกระทำเพื่อโจมตีระบบ

ตัวfile logs ของ Firewall จะมีทุกๆ ข้อมูลพื้นฐานและพร้อมแล้วที่จะทำในส่วนนี้โดยการนำ rc.firewall script เข้ามาใช้โดยระบบจะพร้อมที่จะจับตาการเชื่อมต่อ ทั้งขาเข้าและขาออก ไปสู่ file /var/log/message นี้เองจะเป็นข้อมูลที่สำคัญ เพราะว่าจะเป็นเครื่องบ่งชี้ว่า Attacker ทำอะไรอยู่ มันจะเตือนครั้งแรกเลยเมื่อเริ่มมีการเชื่อมต่อขาออก หรือ การโจมตีขาออกโดยพื้นฐานการทำงานนี้ ตัว firewall จะตรวจสอบความล่อแหลมอย่างรวดเร็วๆในความต่างที่เป็นพฤติกรรมใหม่ หรือ พฤติกรรมที่ไม่รู้นั่นเอง ตัว script เองจะมี สี่ ความแตกต่างของการเดิน packet ดังนี้ TCP, UDP , ICMP และ OTHER จะมีตัว Data Control ตัว OTHER จะเข้ามาแทนทุกๆ อันที่เป็น non-IP proto 1, 6 หรือ 17 ชนิดเหล่านี้ มันจะมีความสนใจต่อเมื่อมีบางคนใช้ non-standard IP traffic เหล่านี้เหมือนกับเป็นการพยายามโจมตีใหม่ๆ หรือ วิธีการที่ไม่เคยพบมาก่อน อาจจะเป็นช่องทางของ Backdoor ที่เคยมีการค้นพบ อ่านได้ในบทความ Scan of the month 22 (http://www.honeynet.org/scans/scan22/)

ในส่วนที่สองเป็นการตรวจจับทุกๆ Packet และ มันมีประโยชน์มากในการจับทั้งเมื่อเข้ามาและ เมื่อออกไปจากตัวฮันนี่พ็อต โดยใช้มาตรฐานของ snort.conf configuration file นี้จะเป็นการ configure file เพื่อตรวจจับทุกๆ IP ที่สัญจรผ่านในเครือข่าย ไปสู่ tcpdump log file สำหรับนำมาวิเคราะห์ต่อในภายหลัง ดังนั้น การจะสั่งให้มีการเก็บข้อมูลจากsnortตามที่ต้องการ อย่างอัตโนมัติจะกระทำผ่าน snort.sh start script จะเป็นตัวเริ่มการทำงานเพื่อให้ไม่มี cron ในทุกวัน แจ้งให้ทราบว่า จะทำอย่าไรในการใช้ startup script ทางผู้พัฒนาได้สร้างการตรวจจับไว้ที่ส่วน interface

ในส่วนที่สามนี้เป็นการท้าทายการดักจับเหล่าผู้โจมตีบนตัว Honeypot เองมันเป็นแบบง่าย มันมีการใช้กันมานานแล้วกับระบบที่เป็นแบบ cleartext protocol เช่น FTP, HTTP และ Telnet คุณแค่เพียงมีตัว sniff ที่คักจับการกด keystroke อย่างไรก็ตาม ผู้โจมตี จะเหมือนกันกับคุณคือมีการเข้ารหัสของข้อมูลในปัจจุบันพวกเขาใช้ SSH หรือ 3DES เป็นช่องทางในการเชื่อมต่อการสื่อสารกันกับเครื่อง

computer พวกเราจะไม่สามารถจับการกดคีย์(keystrokes) ผ่านทางสายได้อีกต่อไปแทนการที่จะจับพวกเขาจากระบบที่เขาใช้หนึ่งในประโยชน์ของระบบที่มีการเข้ารหัสนั้นคือเครื่องปลายทางในที่นี้คือเครื่อง honeypot ของคุณ ถ้าหากว่ามีการจับข้อมูลบนเครื่อง honeypot ของคุณ ซึ่งข้อมูลเป็นการ decryption เราสามารถส่งข้อมูลผ่านช่องทางที่ encryption ได้ ตัว Sebek เป็นเครื่องมือที่กระทำการนี้ โดยตรงตัว Sebek จะซ่อนตัวอยู่ใน kernel ซึ่งเป็น module (หรือบางทีเรียกว่า patch) ความสามารถในการจับตาดูความเคลื่อนไหวของเหล่า Attacker หนึ่งในการติดตั้งตัว honeypot ตัว Sebek client จะทำงานอยู่ใน kernel ตัวข้อมูลจะถูกจับกลุ่มรวมกันโดย Sebek client โดยจะไม่เก็บไว้ในส่วนของตัว Honeypot เองเพราะว่ามันเป็นที่ซึ่งเหล่าผู้โจมตีจะสามารถเข้าไปค้นพบได้ แทนที่โดย ตัว Sebek client จะส่งข้อมูลผ่านทาง UDP เป็นการตรวจจับตัวเครื่องจักร เช่น ตัว Honeywall Gateway หรือการ Remote login ไปยังตัวระบบต่างๆ ผู้โจมตีจะไม่สามารถเห็น และ ก็ไม่เห็น packet ของตัว sniff เช่นกัน ราวกับว่าตัว Sebek Client นั้นซ่อนตัวอยู่บน Honeypot และหลบซ่อนไม่ให้เขาเหล่านั้นเห็น เหตุการณ์หากว่าผู้โจมตี download หรือ เขาใช้เครื่องมือในการดักจับ Sniffing Sebek เองจะทำงานโดยการซ่อนตัวจากพวกเขาเหล่านั้นนี้เป็นการทำโดยปรับปรุงตัว honeypot ให้มันไม่เห็นการดักจับ packet กับการออกแบบเริ่มต้นเป็นคล้ายกับเวทมนต์ที่ได้ทำไว้ไม่เห็นทั้งจำนวน และ port ของ UDP ที่เราเปิด ตัว Sebek เองได้ทำการส่งข้อมูลของเหล่าผู้โจมตีไปบนสายกับการตรวจจับโดยตัว Gateway ตั้งแต่นั้นมาตัว Honeypot เองก็ถูกควบคุมโดย Sebek ไม่มีผู้ใดที่จะสามารจับการกดคีย์ (keystroke) บนสายได้ (หมายเหตุ: ถ้าคุณมีตัว Honeypot แต่คุณเองไม่มี Sebek ติดตั้งอยู่ หรือ ตัว Sebek คุณติดตั้งแล้วทำการ Configure ไม่ถูกต้อง) และเหล่าผู้โจมตีจะเข้ามาควบคุมระบบของคุณได้ดังนั้นเมื่อเขาสามารถที่จะดักฟัง packet ที่เข้ามาจากระบบอื่นๆ เหล่า Packet จะไม่ถูกซ่อนตัวอีกต่อไป

ตัว Sebek นี้ทำงานใน Kernel โดยมีการ Compile มาสำหรับแต่ละชนิดของ OS และ Kernel version นั้นๆ ของ Honeypot ของคุณในขณะที่ Client รุ่นอื่นๆ ที่ต่างกัน และ ระบบปฏิบัติการที่ต่างกัน พวกมันจะมีการ Configure ระบบในลักษณะที่คล้ายกัน file ข้างล่างเป็นตัวอย่าง วัตถุประสงค์ของการConfigure ตัวระบบเพื่อกำหนดข้อมูลที่ถูกรวบรวม และอย่างไรก็ตามข้อมูลจะถูกส่งไปตามสาย โดยปกติแล้ว Sebek เองจะตรวจจับทุกๆ การกระทำบนระบบอย่างไรก็ตามคุณก็ยังมีทางเลือกที่จะเลือกตรวจจับการกดคีย์เพียงอย่างเดียวก็ได้ การรวบรวมระหว่างหมายเลขอันน่าอัศจรรย์ (แล้วแต่ชนิดของ Sebek) และ ปลายทางของหมายเลข UDP port ที่ตัดสินใจว่า packet ไหนบ้างที่จะซ่อนตัว ทุกๆ Honeypot ในกลุ่มเดียวกันจะมีการแบ่งการทำงานที่ร่วมกันได้เพื่อให้ได้มาซึ่งผลประโยชน์ที่แท้จริง

#----- INTERFACE:

INTERFACE="eth0"

#----- DESTINATION_IP:

DESTINATION_IP="10.0.0.1"

#----- DESTINATION_MAC:

DESTINATION_MAC="FF:FF:FF:FF:FF:FF"

#----- SOURCE_PORT:

SOURCE_PORT=1101

#----- DESTINATION_PORT:

DESTINATION_PORT=0

#----- MAGIC_VAL

MAGIC_VAL=0

#----- KEYSTROKE_ONLY:

KEYSTROKE_ONLY=0

#----- TESTING:

TESTING=0

หนึ่งในการ Configure Sebek เองจะนำข้อมูลทั้งหมดของระบบเข้าสู่ระบบเครือข่ายมี packet ที่พวกเขาเคยใช้สร้างขึ้นมาโดยเหล่าผู้โจมตี และ ทำให้สะดุดตาเมื่อมีผู้โจมตี โจมตีระบบของคุณมีการเรียนรู้เกี่ยวกับ Sebek สามารถอ่านได้จาก หัวข้อ Sebek

เครื่องมือที่ใช้ในส่วนของการเก็บบันทึกผล

MySQL server

ใช้ในการจัดเก็บข้อมูลลงฐานข้อมูล MySQL เพื่อความเป็นระบบ และสามารถเข้าใช้งานได้อย่างเป็นระบบ เพื่อเชื่อมต่อข้อมูลให้โปรแกรมควบคุมสามารถมาดึงไปใช้งานได้ และรองรับการเชื่อมต่อข้อมูลในการพัฒนาในอนาคต

Sebek_server

เซเบคเซิร์ฟเวอร์ เป็นส่วนที่ทำหน้าที่รับค่าข้อมูลที่เซเบคไคล์เอนดักจับได้จากเครื่องกับดักและระบบการทำงานจะส่งค่ามาเก็บยังฐานข้อมูลที่ได้จัดเตรียมไว้ ซึ่งจะส่งผ่านมาด้วยโปรโตคอล ยูดี

พี พร้อมทั้งเข้ารหัสลับไว้ โดยไม่จำเป็นจะต้องระบุไอพีปลายทาง เพื่อความปลอดภัยจะระบุเป็น MAC Addressของเครื่องserver ซึ่งเครื่องไคลเอนท์ไม่จำเป็นต้องมีการARP Request เพื่อหาปลายทาง ซึ่งจะช่วยลดความผิดสังเกตจากผู้บุกรุกได้ และการทำงานของ Sebek client จะทำการ by-pass ผ่าน TCP stack คือมันจะสร้างpacket เองเลย ทำให้โปรแกรมจำพวก sniffer ที่ผู้บุกรุกอาจนำเข้ามาติดตั้งในเครื่องกับดัก ไม่สามารถตรวจพบ

Samhain (Yule)

เซมเฮนในส่วนนี้จะเป็นเซอร์วิสที่ทำหน้าที่ในการรับข้อมูลจากเซมเฮนที่อยู่บนเครื่องกับดักมาบันทึกเก็บไว้ในฐานข้อมูลที่ได้จัดเตรียมไว้

3.2.4 ส่วนการจัดการและตรวจสอบความเสียหายของเครื่องกับดัก

ในส่วนนี้เราได้สร้างโปรแกรมดูแลจัดการและควบคุมตัวระบบฮันนี่พ็อตทั้งหมด เข้าไว้ในโปรแกรมเดียว Tartarus Management ซึ่งสามารถควบคุมเครื่องที่ทำหน้าที่ต่างๆทั้งหมดได้ รวมถึงความสามารถที่ใช้ในการตรวจสอบสภาพของเครื่องCage โดยได้สร้าง TMdaemon ซึ่งจะทำงานตลอดเวลาอยู่ที่เครื่องกับดัก คอยตรวจเช็คสภาพ และนำข้อกำหนดที่เราตั้งในตัวโปรแกรมดูแลจัดการไปเปรียบเทียบว่า ถึงจุดที่จะต้องหยุดให้บริการแล้วหรือยัง โดยในขั้นแรกนี้เราจะตรวจสอบว่าถ้าหาก Password ของ Administrator ถูกเปลี่ยนเราก็จะหยุดการให้บริการเครื่องกับดักเครื่องนั้นๆ โดยค่าที่ตั้งนี้หากนำไปใช้กับระบบอื่นๆก็สามารถเปลี่ยนแปลงได้เพื่อความเหมาะสมกับระบบขององค์กรนั้นๆ เราจำเป็นอย่างยิ่งที่จะต้องมีการตรวจสอบไม่ให้เครื่องCageอ่อนแอจนเกินไป เพื่อไม่ให้ผู้บุกรุกใช้เครื่องCage เป็นฐานที่มั่นในการโจมตีระบบอื่นๆ ซึ่งจะก่อให้เกิดความเสียหายต่อระบบจริงขององค์กร

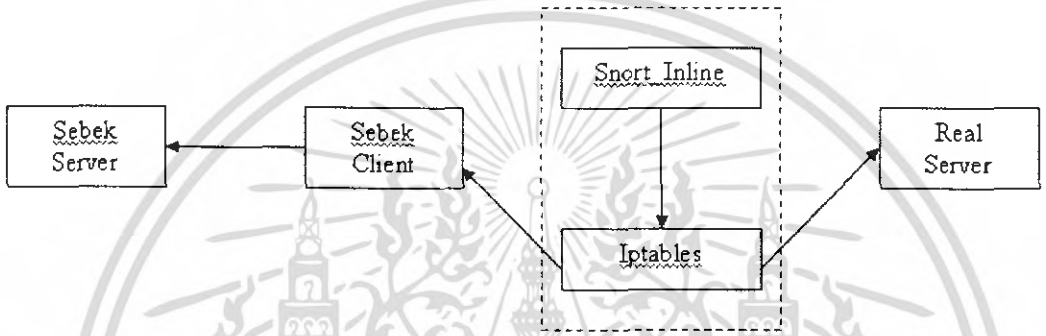
โดยเราสามารถเรียกดูสภาพของเครื่องCageได้ เช่นมีไฟล์ใดที่ถูกเปลี่ยน และสามารถหยุดการทำงานพร้อมทั้งสร้างCageตัวใหม่ได้โดยผ่านโปรแกรม TM และเครื่องกับดักที่หยุดให้บริการก็จะถูกเก็บไว้ เพื่อใช้ในการศึกษาต่อไป

3.2.5 ส่วนการควบคุมและแสดงผลการบันทึกข้อมูลต่างๆของระบบ

การดูแลและการจัดการระบบฮันนี่พ็อตทั้งหมดจะกระทำผ่านโปรแกรมTM (Tartarus Management) เพื่ออำนวยความสะดวกให้แก่ผู้ดูแลระบบ โดยโปรแกรมมีความสามารถควบคุมเครื่องมือในส่วนต่างๆของระบบได้ทั้งหมด เช่น การสั่งหยุดการให้บริการเครื่องกับดัก การสร้างเครื่องกับดักใหม่ หรือการเพิ่มหรือถอดถอนกฎที่ใช้ เป็นต้น

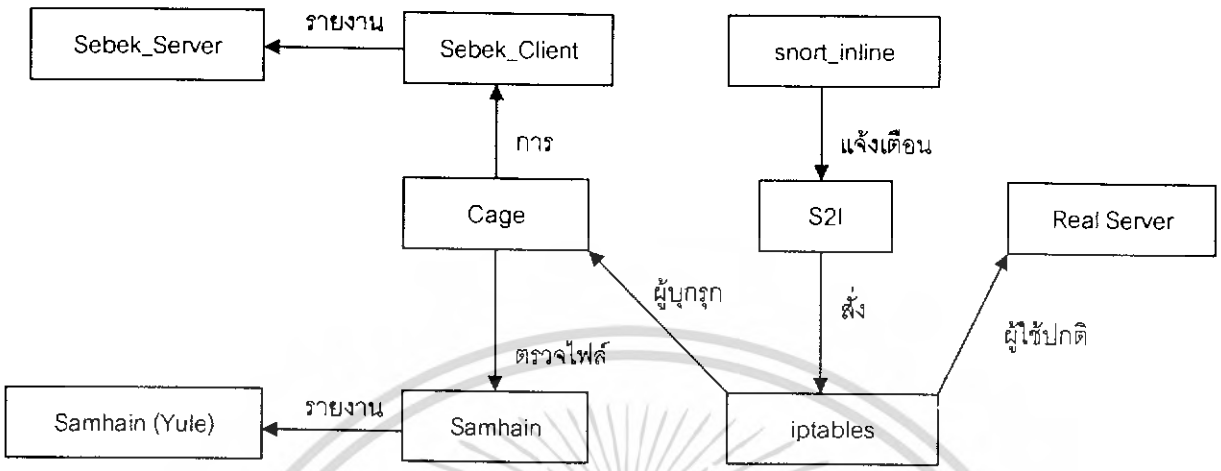
และได้จัดเตรียมช่องทางปลอดภัยในการให้ ผู้ดูแลระบบเข้ามาควบคุมระบบฮันนี่พ็อตได้โดยผ่าน IPsec ที่จะมีการ Authentication และการ Encryption ข้อมูลที่มีการส่งถึงกัน ดังนั้นผู้ดูแลระบบไม่จำเป็นต้องไปทำงานหน้าเครื่อง Honeywall แต่สามารถควบคุมระบบผ่านทางเครือข่ายได้อย่างปลอดภัย ช่วยให้สามารถแก้ปัญหาที่เกิดขึ้นได้ทันที หากเกิดเหตุการณ์ผิดปกติเกิดขึ้น

3.3 การทำงานร่วมกันของระบบ



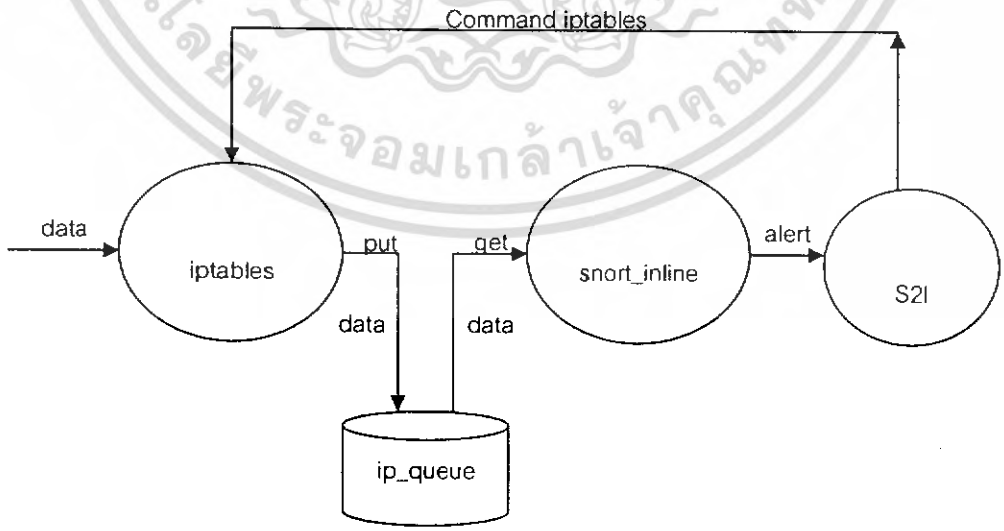
รูปที่ 3.3 แสดงการเชื่อมต่อระหว่างส่วนต่าง ๆ เพื่อทำการจำแนกผู้ใช้งานปกติกับผู้บุกรุกแบบเก่า

จากรูปเป็นลักษณะการทำงานของโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุกที่ได้สร้างขึ้นในปีการศึกษาที่ผ่านมา ซึ่งการทำงานจะเริ่มจาก snort_inline ทำการตรวจสอบและแยกแยะระหว่างผู้บุกรุกกับผู้ใช้งานทั่วไปจากนั้นจะติดต่อไปยังโปรแกรม iptables เพื่อให้จัดเส้นทางการใช้งานระบบเครือข่ายอย่างเหมาะสมตามการจำแนกที่ snort_inline โดยจัดการเส้นทางให้ผู้บุกรุกผ่านไปยังกับดักที่มี sebek client คอยดักจับและส่งการกระทำต่างๆ ของผู้บุกรุกไปให้ sebek server จัดเก็บลงฐานข้อมูลอยู่ตลอดเวลา ส่วนเส้นทางของผู้ใช้งานปกติให้ผ่านไปยังเซิฟเวอร์จริง แต่ในปีการศึกษานี้ได้มีการปรับแต่งและเพิ่มบางส่วนเพื่อให้การทำงานของโปรแกรมมีประสิทธิภาพมากขึ้น โดยได้ทำการเพิ่มโปรแกรม samhain ซึ่งเป็นส่วนที่จะทำงานกับเครื่องกับดักโดยโปรแกรม samhain จะคอยตรวจสอบความถูกต้องของไฟล์สำคัญในระบบของเครื่องกับดัก ซึ่งเราสามารถกำหนดได้ว่าจะให้ทำการตรวจสอบไฟล์ใด ซึ่งการตรวจสอบความถูกต้องของไฟล์บางไฟล์นั้นสามารถใช้เพื่อการตัดสินใจในการเปลี่ยนเครื่องกับดักเนื่องจากสภาพของเครื่องกับดักมีความอ่อนแอจนอาจที่จะไม่สามารถควบคุมพฤติกรรมของผู้บุกรุกให้อยู่ในขอบเขตที่ควรจะเป็น (ไม่สามารถใช้เครื่องกับดักเป็นฐานที่มั่นในการโจมตีเครื่องอื่นๆ ทั้งในระบบและนอกระบบ) ได้ เนื่องจากโปรแกรมมีการเก็บด้วยกว่าไฟล์ถูกเปลี่ยนแปลงไปอย่างไร เช่นถูกเปลี่ยนเจ้าของไฟล์จาก root ไปเป็นบุคคลอื่น เป็นต้น



รูปที่ 3.4 แสดงการเชื่อมต่อระหว่างส่วนต่าง ๆ เพื่อทำการจำแนกผู้ใช้งานปกติกับผู้บุกรุกแบบใหม่

ซึ่งในส่วนของการติดต่อกันระหว่าง snort_inline และ iptables นั้นเป็นโปรแกรมที่ทำงานคนละโหมดกันโดยโปรแกรม snort_inline นั้นจะทำงานในยูสเซอร์โหมดแต่โปรแกรม iptables นั้นทำการในเคอร์เนลโหมด ดังนั้นโปรแกรมทั้ง 2 โปรแกรมนี้จึงไม่สามารถที่จะติดต่อกันเองได้ ดังนั้นจึงต้องมีโปรแกรมสื่อกลางที่จะคอยแปลงคำแจ้งเตือนที่มาจาก snort_inline ไปเป็นคำสั่งที่โปรแกรม iptables นั้นเข้าใจ เพื่อให้การทำงานของระบบเป็นไปได้อย่างมีประสิทธิภาพ ซึ่งในส่วนนี้ยังคงใช้โปรแกรมสื่อกลางดังกล่าวจะยังคงใช้โปแกรม s2i ที่ได้มีการพัฒนาขึ้นในปีที่ผ่านมาซึ่งลักษณะการทำงานเป็นไปตามรูปด้านล่าง



รูปที่ 3.5 แสดงการทำงานของส่วนจำแนกและกำหนดทิศทางการเชื่อมต่อ

3.4 สรุปการทำงานโดยรวมของระบบ

ในการทำงานโดยรวมของระบบฮันนี่พ็อตจะเริ่ม จากส่วนการจำแนกและคัดแยกระหว่างผู้ใช้งานทั่วไปกับผู้บุกรุก จะเป็นหน้าที่การทำงานของตัวเกตเวย์ เรียกกันว่าฮันนี่วอล (Honeywall) ซึ่งเป็นส่วนที่ใช้เพื่อเป็นทางผ่านของชิ้นข้อมูลทุกชิ้นที่จะเข้ามาในระบบเครือข่าย ในลักษณะการติดตั้งตัวเกตเวย์ไว้ที่ส่วนที่เสมือนประตูทางเข้าออกนี้ เป็นส่วนที่สามารถใช้ประโยชน์ของระบบตรวจจับผู้บุกรุก (snort) เข้าไปเพื่อให้การวิเคราะห์และจำแนกผู้บุกรุกนั้นมีประสิทธิภาพสูงสุด แต่ในส่วนการทำงานเพียงลำพังของระบบตรวจจับผู้บุกรุก นั้นไม่เพียงพอที่จะจัดการให้ผู้บุกรุกเข้าสู่ระบบกับดักที่จัดเตรียมไว้ได้ เนื่องจากระบบตรวจจับผู้บุกรุกนั้นไม่มีความสามารถเพียงพอในการจัดการด้านช่องทางการเชื่อมต่อ ซึ่งความสามารถนี้เป็นความสามารถของไฟร์วอล ที่มีชื่อเรียกว่า IPtables เพื่อเป็นตัวที่เพิ่มขึ้นมาเพื่อเป็นตัวจัดการ การเชื่อมต่อสื่อสารเพื่อให้เป็นไปตามความต้องการของผู้พัฒนา แต่ในการทำงานจริงนั้น ทั้งสองโปรแกรมนี้ไม่สามารถพูดคุยเพื่อควบคุมกันได้ เพราะความสามารถที่มีนั้นเป็นแค่เพียง ไฟร์วอลนั้นส่งค่าของชิ้นข้อมูลไปยังส่วนที่โปรแกรมในส่วนยูเซอร์โหมด (user mode) ก็คือตัวระบบตรวจจับผู้บุกรุก (snort) เนื่องจากว่าไฟร์วอลเองเป็นโปรแกรมที่ทำงานในส่วนของเคอร์เนลโหมด (kernel mode)ดังนั้นจึงได้มีการใช้งานตัวกลางที่จะสามารถสื่อสารระหว่างระบบตรวจจับผู้บุกรุกกับไฟร์วอลเป็นตัวที่ใช้การอ่านค่าจากการแจ้งเตือนจากระบบตรวจจับผู้บุกรุก และอ่านค่าที่ได้มานั้นแปลงเป็นคำสั่งของไอพีเทเบิล อย่างเหมาะสมเพื่อสั่งให้จัดส่งชิ้นข้อมูลไปยังปลายทางที่ถูกต้องS2I(Snort command to Iptables)หลังจากนั้นผู้บุกรุกจะถูกเปลี่ยนแปลงการทำงานให้ไปมีผลต่อเครื่องกับดักแทน โดยเครื่องที่ทำหน้าที่เป็นกับดัก(cage)ทางผู้พัฒนาได้พัฒนาให้มีความปลอดภัยต่อองค์กร นั่นคือผู้บุกรุกจะไม่สามารถใช้เครื่องกับดักไปเป็นฐานโจมตีเครื่องอื่นๆ และพัฒนาให้มีความแนบเนียนเสมือนว่าเป็นเครื่องให้บริการจริงที่เป็นเป้าหมายของผู้บุกรุก โดยจะตรวจสอบไม่พบความผิดปกติของค่าต่างๆที่ระบุว่าเป็นเครื่องกับดัก เช่น ไอพีแอดเดรส แมคแอดเดรส หรือไม่พบเซอร์วิสที่ทำให้ทราบว่าได้มีการเฝ้าดูการกระทำต่างๆ เช่น ทุกๆkeystrokeที่ผู้บุกรุกสั่งงานเข้ามา แม้ผู้บุกรุกจะใช้คำสั่งผ่านการเข้ารหัสก็ตาม หรือแม้แต่การนำเข้าไฟล์หรือส่งออก packet ตัวเครื่องกับดักก็จะส่งข้อมูลทุกอย่างไปเก็บไว้ยังเครื่องแม่ข่ายฐานข้อมูล (Log serer) ซึ่งการส่งข้อมูลออกไปจะส่งไปแบบเข้ารหัสและไร้ตัวตน โดยที่ผู้บุกรุกไม่ทราบว่าได้มีการส่งข้อมูลออกไปด้วยเทคนิคของSebek ซึ่งทำงานในแบบ kernel-base นั่นคือแม้ว่าผู้บุกรุกจะนำโปรแกรมดักจับแพ็คเก็จมาติดตั้งก็จะไม่พบว่าเครื่องกับดักได้ส่งข้อมูลออกไป นอกจากนี้เครื่องกับดักยังได้ติดตั้งเครื่องมือที่ใช้ในการตรวจสอบความเปลี่ยนแปลง และตรวจสภาพความเสียหายเอาไว้ และทำการตรวจสอบอยู่ตลอดเวลา เมื่อถึงจุดที่มีความเสียหายตามที่กำหนดเครื่องกับดักก็จะหยุดให้บริการโดยอัตโนมัติ

ในส่วนของการจัดการและเรียกดูข้อมูลทางผู้พัฒนาได้สร้างชุดโปรแกรม TM (Tartarus Management) ที่สามารถติดต่อทำงานร่วมกับส่วนต่างๆ เพื่อสามารถจัดการและบริหารระบบได้โดยใช้เพียงโปรแกรมเดียว และยังสามารถเรียกดูข้อมูลทั้งหมดที่มีการบันทึกผ่านโปรแกรมนี้ได้ทันที โดยโปรแกรมนี้ยังสามารถสั่งการผ่านเครือข่ายได้โดยปลอดภัยโดยใช้ IPsec (IP security) ซึ่งจะเพิ่มความสะดวก และรวดเร็วในการเข้าแก้ปัญหาที่อาจเกิดขึ้นในระบบจริง เพราะผู้ดูแลระบบไม่จำเป็นจะต้องเข้ามาควบคุมหน้าเครื่องHoneywall แต่สามารถควบคุมผ่านทางเครือข่ายได้เลย

3.5 ข้อจำกัดและข้อด้อยของฮันนี่พ็อตเดิม

3.5.1 ความปลอดภัยของเครื่องกับดัก (Cage) โดยผู้บุกรุกอาจใช้เครื่องกับดักเป็นฐานที่มั่นในการไปโจมตีเครื่องอื่นๆ

3.5.2 ความแนบเนียนของเครื่องกับดัก (Cage) ผู้บุกรุกอาจตรวจสอบพบว่าข้อมูลและค่าประจำเครื่อง ไม่ได้เป็นเครื่องเดียวกับเครื่องให้บริการจริง ทำให้ผู้บุกรุกรู้ตัวและไม่กระทำการใดๆ

3.5.3 ความยุ่งยากในการนำมาใช้งานจริง เนื่องจากระบบฮันนี่พ็อตต้องใช้การทำงานของเครื่องที่ทำ หน้าที่ต่างๆ และแต่ละเครื่องต้องมีการใช้งานหลายเครื่องมือ ดังนั้นการบริหารจัดการและการดูแลสภาพของระบบจึงมีความซับซ้อนยุ่งยาก

3.5.4 การเปลี่ยนแปลงต่างๆที่เกิดจากผู้บุกรุก เมื่อผู้บุกรุกมีการใช้งานเครื่องมือต่างๆ แล้วไปเปลี่ยนแปลงไฟล์ใดๆ ทางฮันนี่พ็อตไม่สามารถติดตามได้ ตัวอย่างเช่น ทางระบบบันทึกได้ว่าผู้บุกรุกได้นำscript หรือเครื่องมือใดๆเข้ามา และทำการเรียกใช้งาน แต่ไม่ทราบว่าเครื่องมือเหล่านี้ไปเปลี่ยนแปลงไฟล์หรือกระทำการใดๆบ้าง

3.6 รูปแบบระบบฮันนี่พ็อตที่เราได้พัฒนาขึ้นใหม่

ระบบฮันนี่พ็อตที่ได้พัฒนาแล้วจะช่วยลดจุดด้อยที่มีอยู่และเพิ่มประสิทธิภาพของการทำงานเดิม ดังนี้

3.6.1 เครื่องกับดักมีความปลอดภัย จะไม่สามารถถูกใช้ไปโจมตีเครื่องอื่นๆได้ เพราะจะถูกระงับการใช้งานทันที โดยอัตโนมัติเมื่อถึงจุดที่เครื่องมีความอ่อนแอ ตามที่ผู้ดูแลระบบเห็นสมควร ซึ่งแต่เดิมผู้ดูแลระบบจะต้องหมั่นมาตรวจสอบสภาพเครื่องกับดักเอง

3.6.2 เครื่องกับดักมีความแนบเนียน โดยเมื่อผู้บุกรุกตรวจสอบเครื่องว่าเป็นเครื่องใด เช่น ใช้คำสั่งตรวจสอบไอพีแอดเดรส หรือตรวจสอบแมคแอดเดรส

3.6.3 ปัญหาความยุ่งยากด้านการควบคุมและดูแลระบบฮันนี่เนตถูกทำให้ลดน้อยลง โดยการมีโปรแกรมซึ่งเป็นศูนย์กลางควบคุมระบบทั้งหมด ซึ่งได้มีคำสั่งของเครื่องมือและองค์ประกอบที่จำเป็นอย่างครบถ้วน ผู้ใช้จึงไม่จำเป็นต้องทราบ คำสั่งต่างๆที่ใช้ในการควบคุมและสั่งงาน รวมทั้งการเรียกดูข้อมูลที่ได้จัดเก็บไว้ โดยโปรแกรมนี้จัดทำในแบบ Graphic User Interface จึงยิ่งง่ายในการใช้งานของระบบฮันนี่พ็อต

3.6.4 เพื่อติดตามการเปลี่ยนแปลงของระบบ จึงได้จัดหาเครื่องมือที่จะช่วยตรวจสอบความเปลี่ยนแปลงของระบบได้ ว่าไฟล์ใดถูกเปลี่ยนแปลง ไฟล์ใดถูกลบทำลาย จึงเก็บรายละเอียดการกระทำของผู้บุกรุกได้ละเอียดยิ่งขึ้น

3.6.5 สามารถทำการป้องกันและดักจับโค้ดหนอนที่แพร่กระจายในเครือข่ายได้ และสามารถดูพฤติกรรมต่อได้ ว่าโค้ดหนอนนั้นๆมีการกระทำใดบ้าง ไปแก้ไขไฟล์ใดบ้าง

3.6.6 เพิ่มขีดความสามารถในการรองรับจำนวนเครื่องกับดักให้มีมากและควบคุมได้ตามเท่าที่ต้องการ สามารถสั่งสร้างและเปลี่ยนเครื่องกับดักได้ทันที ที่ต้องการ

3.6.7 การเก็บรวบรวมข้อมูลทุกอย่างจัดทำขึ้นใหม่ในรูปแบบฐานข้อมูล ซึ่งจะช่วยในการจัดการข้อมูลได้มีอย่างเป็นระบบมากขึ้น

3.7 ข้อควรระวังในการใช้งานระบบฮันนี่พ็อต

ในการใช้งานระบบฮันนี่พ็อต การกำหนดค่าความปลอดภัยและจุดที่ถือว่าจำเป็นต้องหยุดการใช้เครื่องกับดักนั้นๆแล้ว เพื่อความคลอบคลุมและยืดหยุ่น เราได้ออกแบบให้ผู้ดูแลระบบสามารถกำหนดได้เองตามความเหมาะสม โดยได้จัดเตรียมกฎเกณฑ์ต่างๆไว้ให้เลือกใช้ แต่ไม่ว่าอย่างไรถ้าหากมีการกำหนดที่ไม่เหมาะสม หรือจัดวางตำแหน่งของเครื่องมือต่างๆอย่างไม่เหมาะสม ก็จะทำให้เครื่องกับดักอาจถูกใช้เป็นฐาน ในการโจมตีเครื่องอื่นๆได้

บทที่ 4
เครื่องมือที่นำมาประยุกต์ใช้งาน

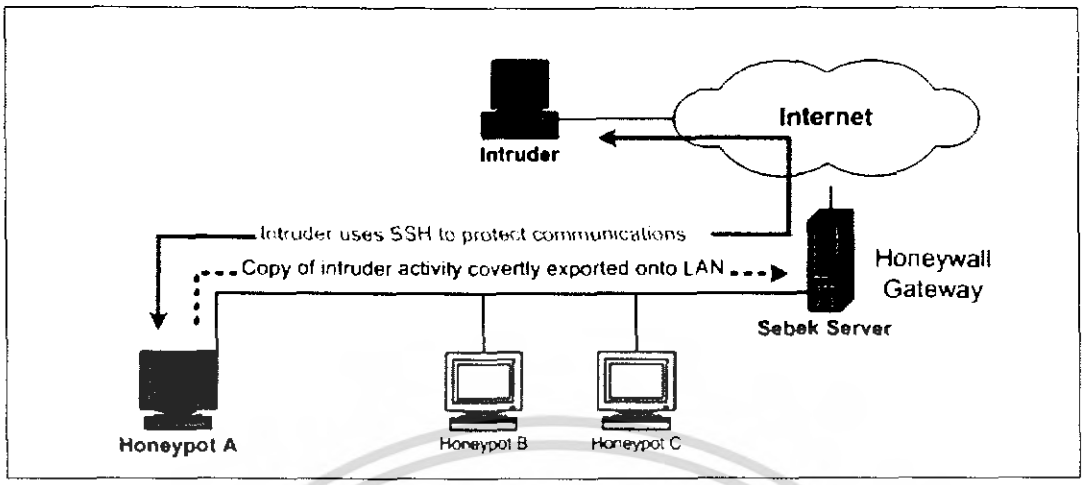
เนื่องจากโครงงาน ชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก ประกอบขึ้นด้วยส่วนประกอบหลายๆส่วน มีหน้าที่การทำงานในหลายๆด้าน จึงจำเป็นที่จะต้องใช้เครื่องมือที่เหมาะสม และสามารถทำงานได้อย่างที่ผู้พัฒนาคาดหวัง โดยแต่ละเครื่องมือก็จะมีจุดเด่น และข้อจำกัดในการใช้งาน ซึ่งการนำมาใช้งานร่วมกัน ให้สามารถสื่อสารและเชื่อมโยงกันได้นั้น จำเป็นที่จะต้องนำเครื่องมือต่างๆมาปรับปรุงและประยุกต์ใช้ เพื่อให้ได้ประสิทธิภาพของระบบที่ดีที่สุดโดย ในบทนี้จะให้รายละเอียด ของแต่ละเครื่องมือที่นำมาประยุกต์ใช้งานในระบบ

เครื่องมือต่างๆที่ได้นำมาประยุกต์ใช้งาน

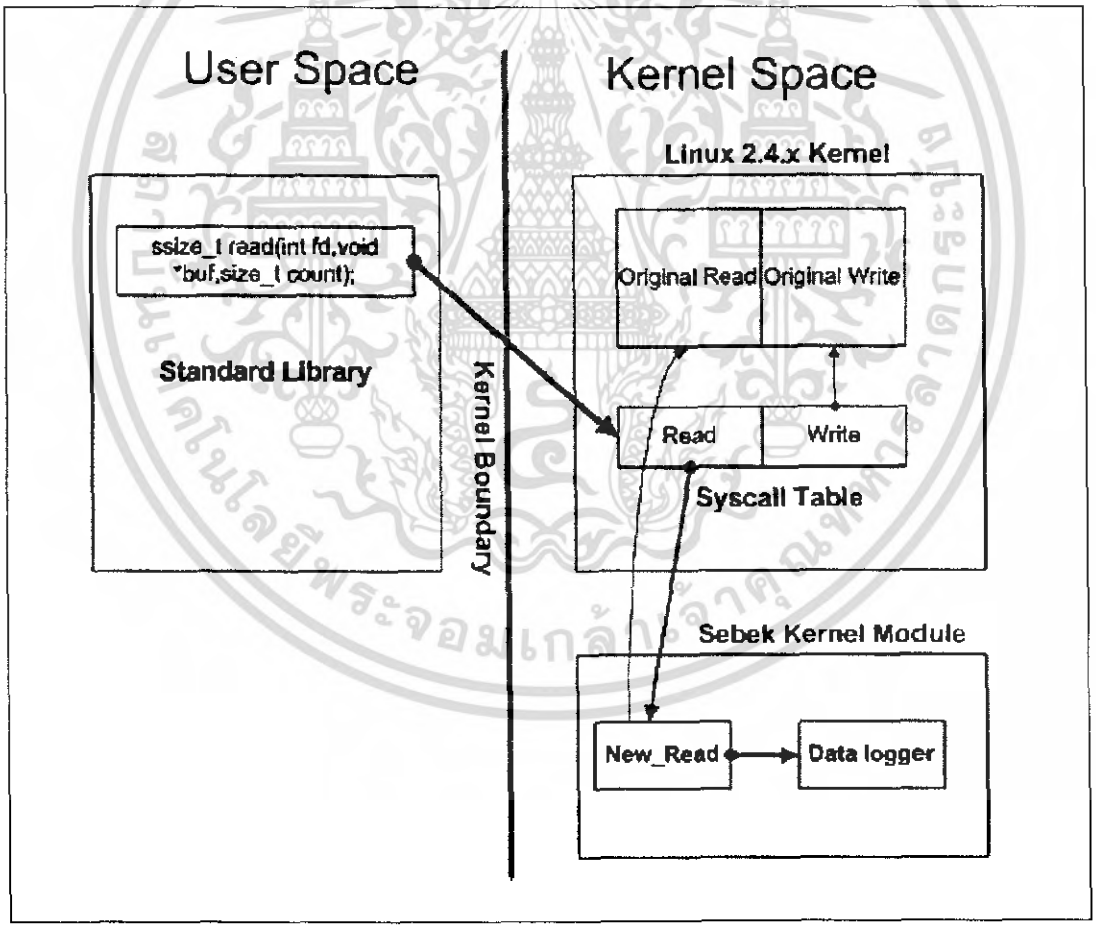
- Sebek
- Samhain
- Snort inline
- IPsec
- S2I
- Iptables

4.1 Sebek

เซเบคไคล์เอนจะคอยซ่อนตัวอยู่ในกับดักเพื่อเฝ้าดูพฤติกรรมและการกระทำของผู้บุกรุก และคอยส่งข้อมูลนั้นไปยังเครื่องที่ใช้เก็บพฤติกรรมลงฐานข้อมูล โดยส่งผ่านช่องทางที่มีความปลอดภัย มีการเข้ารหัสข้อมูลในรูปแบบของ เซเบคเอง

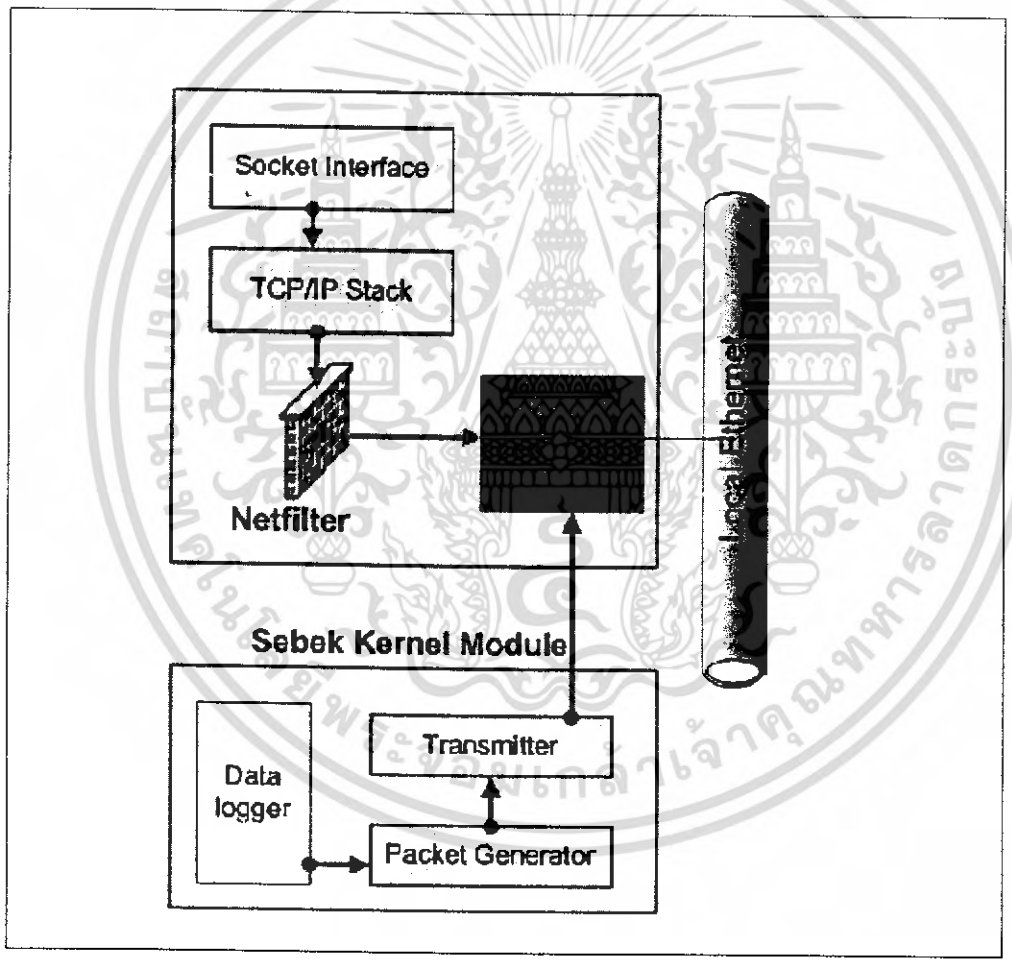


รูปที่ 4.1 แสดงการทำงานของ Sebek



รูปที่ 4.2 แสดงการcapture ข้อมูลที่ทำภายใน kernel

เซเบคเป็นData Capture Toolแบบ Kernel-based โดยในการนำมาใช้จะประยุกต์ใช้เป็นแบบ LKM (Loaded Kernel Module) มีความโดดเด่นในความสามารถในการดักจับข้อมูลได้ทุกการกระทำของIntruderที่กระทำผ่าน read() system call ไม่ว่าจะเป็นในส่วนของ keystroke, files transfer, burneye passwords หรือหากมีการใช้ IRC clientหรือ e-mail client ตัวเซเบคก็สามารถดักจับได้ ดังเช่นเมื่อIntruderมีการนำfileส่งเข้ามา ตัวเซเบคก็สามารถอ่านและบันทึกไฟล์นั้นไว้ได้ ทั้งยังสามารถใช้เป็นGlass-box เมื่อเปรียบเทียบกับการทำงานของblack-boxที่เรารู้จัก นั้นคือเราสามารถใช้ติดตามการทำงานของโปรแกรมที่intruderอามารันได้ด้วย แม้ว่าตัวintruder จะlog out ออกไปแล้ว



รูปที่ 4.3 แสดงการby-passไม่ผ่าน TCP stack

เซเบคมีรูปแบบวิธีการปิดซ่อนตัวเองได้อย่างมีประสิทธิภาพ และค่อนข้างยากที่จะตรวจสอบพบทั้งจากภายในและภายนอก โดยจะทำการปิดซ่อนโมดูลที่ใช้ และลบlinked list ที่จะแสดงการมี

ตัวตนออก โดยใช้ the cleaner ทำให้Intruder ไม่สามารถตรวจสอบพบว่าเซเบคกำลังทำงานอยู่ภายในเครื่อง ทุกpacketของเซเบค ตัวเซเบคจะมีfunctionในการสร้างและส่งเอง โดยไม่ใช้ TCP/IP stack ดังนั้นจึงไม่สามารถถูกมองเห็นและไม่สามารถblock traffic ของเซเบคได้ หลังจากสร้างpacketเสร็จ packetจะถูกส่งไปยังdevice driverโดยตรง ไม่ผ่าน raw socket socket interface ซึ่งพวกโปรแกรม sniffer ที่ใช้ libpcap เป็นฐานจะไม่สามารถเห็นได้เพราะมันไปดักจับข้อมูลที่ raw socket interface และเพื่อป้องกันการตรวจจับในเครือข่าย sebekจะไม่ใช้ARP แต่จะกำหนดMAC Addressของserver ไว้เลย เพื่อไม่ต้องทำการ ARP request เพื่อป้องกันการทำ ARP spoofing ซึ่งก็จะลดการดักจับข้อมูลในเครือข่ายได้ ทั้งนี้เซเบคมีรูปแบบการส่งข้อมูลด้วยโปรโตคอล UDP

4.2 Samhain

Samhain เป็นระบบที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลและใช้ในการแจ้งเตือนการบุกรุกระบบซึ่งสามารถใช้บนเครื่องโฮสต์เครื่องเดียว หรือใช้ในระบบเครือข่ายขนาดใหญ่ก็ได้ ซึ่งในโครงงานนี้ไม่ได้ใช้กับเครื่องโฮสต์เพียงเครื่องเดียว หากแต่จะใช้กับเครื่องหลายๆ เครื่องที่อยู่ในระบบเครือข่ายโดยจะมีเครื่องแม่ข่ายที่จะทำหน้าที่เป็นศูนย์กลางคอยรับข้อมูลจากเครื่องลูกข่ายทั้งหลายในระบบเครือข่าย ซึ่งจะได้อธิบายรายละเอียดของโครงสร้างและการทำงานในส่าวนถัดไป

4.2.1 การใช้งานในรูปแบบระบบเครือข่าย (network)

Samhain เป็นระบบที่ถูกออกแบบมาให้ง่ายต่อการมอนิเตอร์โฮสต์หลายๆ เครื่องในเครือข่าย ซึ่งมันจะประกอบไปด้วยโพรเซสที่ทำงานในลักษณะที่เป็นเดมอนในโฮสต์แต่ละเครื่อง และจะมีเครื่องแม่ข่ายกลางที่จะคอยเก็บบันทึกหรือรายงานต่างๆ จากเดมอนเหล่านั้นผ่านทางการเชื่อมต่อแบบทีซีพี/ไอพี ซึ่งในการส่งข้อมูลจากเครื่องลูกข่ายต่างๆ ไปให้เครื่องแม่ข่ายกลางนั้นก็จะต้องมีการยืนยันตนก่อนเพื่อป้องกันการปลอมแปลงข้อความที่ส่งไปให้เครื่องแม่ข่าย โดยในขั้นแรกจะต้องตกลงกันในเรื่องของโปรโตคอลที่จะใช้ในการยืนยันตนก่อน จากนั้นจะมีการแลกเปลี่ยน session key กัน การเชื่อมต่อที่มาจากเครื่องโฮสต์ที่ไม่ได้ทำการลงทะเบียนไว้จะถูกดรอปทิ้งทันที และในอีกกรณีหนึ่งที่จะถูกดรอปคือการเชื่อมต่อจากเครื่องโฮสต์ที่ได้มีการลงทะเบียนไปแล้วแต่เครื่องลูกข่ายไม่สามารถทำการยืนยันตนได้สำเร็จ

เมื่อ Session key ได้ถูกสร้างขึ้นมาเครื่องลูกข่ายจะใช้ session key นั้นในการเซ็นลงไปในข้อความของตนเพื่อใช้ในการยืนยันตนและเมื่อข้อความถูกส่งไปถึงเครื่องแม่ข่าย เครื่องแม่ข่ายจะทำ

การตรวจสอบลายเซ็นของเครื่องลูกข่ายจาก เมื่อตรวจสอบพบว่าถูกต้องแล้วเครื่องแม่ข่ายจะเอาลายเซ็นนั้นๆ ออกและนำเอาลายเซ็นของตนเซ็นลงไปเมื่อจะเก็บข้อความนั้นลงใน log file

ทั้งคอนฟิกูเรชันไฟล์และฐานข้อมูลสามารถเก็บไว้ที่ส่วนกลางบนฝั่งของเครื่องแม่ข่ายได้และสามารถดาวน์โหลดไปใช้โดยเครื่องลูกข่ายในขณะที่ทำการ Startup เครื่องได้อีกด้วย

4.2.2 การไม่แสดงตัวตนว่ามีการใช้งานอยู่ (stealth)

Samhain มีความสามารถในการอำพรางตัวไม่ให้ผู้บุกรุกสามารถรู้ได้ว่ามีโพรเซสของ samhain กำลังทำงานอยู่ในระบบซึ่งการทำงานในลักษณะนี้เรียกว่าทำงานแบบ stealth ซึ่งสามารถทำได้โดยการ compile samhain ให้สนับสนุนโหลด stealth แต่การทำงานในโหมด stealth นั้นก็ยังอาจสามารถถูกตรวจสอบได้จากการค้นหาคอนฟิกูเรชั่นไฟล์ หรือค้นหาจากสตริงที่อยู่ในดังนั้น samhain จึงมี option ที่เพิ่มเติมเข้ามาคือ

- สตริงที่สามารถอ่านได้ซึ่งอยู่ใน executable file และที่อยู่ใน log และฐานข้อมูลสามารถทำให้ดูไม่รู้เรื่องได้ตัวอย่างเช่น อาจทำให้มองดูแล้วเหมือนเป็นข้อมูลไบนารี่
- สามารถยกเลิกการใช้ command line parsing ได้
- สามารถทำ Configuration data ให้เป็นรูปภาพโดยใช้ steganography
- Executable สามารถบีบอัดและเข้ารหัสลับได้

ฐานข้อมูลและ log file อาจถูกนำไปซ่อนโดยการนำไปต่อท้ายภาพที่มีอยู่แล้วก็ได้โดยภาพนั้นยังคงดูเหมือนปกติ

4.2.3 คุณสมบัติในการทำงานของ Samhain

- สามารถchecksumได้หลากหลายรูปแบบ เช่น TIGER192, SHA-1 ,MD5 เป็นต้น โดยจะตรวจสอบได้จากหลาย ส่วน เช่น size, mode/permission, owner, group, creation/ modification/access time, inode, number of hardlinks, linked path(symbolic links) major/minor device number
- สามารถใช้ shell wildcard pattern ในการระบุไฟล์หรือไดเรกทอรี่ที่จะตรวจสอบได้
- สามารถตั้งและปรับแต่ง policies ของระบบได้ 8 รูปแบบที่แตกต่างกัน
- การตรวจสอบในเชิงลึกสามารถตั้งค่าให้ตรวจสอบทั้งหมด หรือตรวจสอบเฉพาะส่วนได้
- สามารถตรวจสอบความเปลี่ยนแปลงของระบบได้ ว่ามีไฟล์ใดถูกแก้ไข หรือถูกลบไป

4.2.4 ตารางการตรวจเช็คของโปรแกรม

- ตรวจสอบเมื่อผู้ใช้สั่ง
- ตรวจสอบเมื่อมีการส่ง signal ไปยัง samhain daemon โดย signal อาจมาจากเครื่องที่ทำหน้าที่เป็น Samhain server
- สามารถสั่งให้ทำงานได้อัตโนมัติ เมื่อมีการใช้งานเครื่อง

4,2,5 ความสามารถในการตรวจสอบที่โดดเด่นอื่นๆ

Kernel integrity สามารถตรวจสอบ integrity ของ kernel ที่กำลังใช้งานอยู่ได้ เพื่อตรวจสอบ rootkits(สามารถใช้ความสามารถนี้ได้เฉพาะกับระบบปฏิบัติการการ Linux หรือ FreeBSD แต่โดยไม่สามารถใช้กับ Fedora Core 2)

SUID/SGID files

- สามารถตรวจสอบการเปลี่ยนแปลงหรือสร้าง SUID/SGID files
- โดยสามารถสั่งให้ตรวจสอบอย่างสม่ำเสมอตามเวลาที่กำหนดได้
- สามารถลบหรือกักขังไฟล์ที่มีการกระทำดังกล่าวไว้ก่อนได้ โดยค่าที่เป็นค่าตั้งต้นจะกระทำเพียงแค่การเตือน

Mount check

- สามารถตรวจสอบการmount ต่างๆ รวมทั้งการ mount files system.

Login/logoff events

- สามารถตรวจสอบการ login/logoff ของผู้ใช้งานได้โดยใช้ system file ที่ชื่อว่า utmp

Log facilities

- การกำหนดให้มีการจัดเก็บและแสดงการเตือน สามารถกำหนดได้
- เครื่องแม่ข่ายฐานข้อมูลจะได้รับข้อมูลผ่านทาง TCP connection ที่มีการเข้ารหัสลับไว้ และมีการ Authentication ระหว่างกัน เพื่อให้แน่ใจได้ว่า เป็นตัวจริงทั้งคู่

- สามารถใช้งาน Syslog
- สามารถใช้งานผ่านConsole ถ้าได้มีการติดตั้งตัว daemonไว้
- Log files ที่จัดเก็บจะมีการ Sign ไว้เพื่อป้องกันการแก้ไขของผู้ไม่มีสิทธิ์
- E-mail ที่ใช้จะมีการ Sign ไว้เช่นเดียวกัน
- รองรับตัว RDBMS ที่หลากหลาย เช่น MySQL, PostgreSQL, และ Oracle

Integration with other monitoring systems

- **Nagios** : เป็นplugin เสริมที่สร้างขึ้นจาก Perl สำหรับตัว Nagios (check_samhain.pl)

- **Prelude** : สามารถ compiled เพื่อใช้งานในการตรวจสอบว่ามีแนวโน้มของระบบเป็นเช่นไร มีลางบอกเหตุว่าจะเกิดเหตุการณ์ร้ายแรงหรือไม่

Integrity of the file checking system

เป็นเรื่องที่มักจะต้องเลือกเสมอระหว่างความปลอดภัยกับความสะดวกในการใช้งาน ที่ผ่านมาก็เลยมีการเก็บ โปรแกรมที่ใช้ตรวจสอบไว้บนเครื่องนั้นๆ แล้วหวังว่าจะไม่ถูกผู้บุกรุกเข้าไปแก้ไขหรือปลอมแปลง ซึ่งการทำเช่นนี้ก่อให้เกิดความไม่ปลอดภัยเกิดขึ้น ดังนั้น Samhain จึงใช้การ Sign ทั้งส่วนของ ฐานข้อมูลและส่วนของ configuration file เพื่อให้มีความถูกต้องของการทำงานโดยได้ทำการSign โดยใช้ GNuPG

4.3 Snort inline

Snort inline เป็นโปรแกรมได้ถูกประยุกต์มาจากโปรแกรมSnort ซึ่งตัวSnort inlineจะติดต่อรับแพ็คเก็ตจาก Iptables ผ่านทาง libipq แทนที่จะของเดิมที่จะรับจาก libpcap และกฎที่ใช้ในตัวSnort inline ก็จะเป็นรูปแบบของกฎใหม่ โดยมีการทำงาน drop, sdrop, reject โดยสั่งงาน Iptablesได้ 3 คำสั่งดังกล่าว ตัวSnort inline จึงเหมือนทำหน้าที่เป็น IPS (Intrusion Prevention System)

การกระทำอันเนื่องจากกฎที่ใช้

- **Drop** **-The drop rule type**
  จะไปสั่งให้ iptables ทำการ drop แพ็คเกต และทำการเก็บ logไว้
- **Reject** **-The reject rule type**
  จะไปสั่งให้ iptables ทำการ drop แพ็คเกต และทำการเก็บ logไว้ พร้อมทั้งส่ง TCP reset หรือ UDP reset กลับไป แล้วแต่กรณี

- **Sdrop** **-The sdrop rule type**

  จะไปสั่งให้ iptables ทำการ drop แพ็คเกต โดยไม่ทำการบันทึกใดๆ

ลำดับในการพิจารณากฎของSnort inline

->activation->dynamic->drop->sdrop->reject->alert->pass->log

สามารถใช้ทางเลือก –o เพื่อเปลี่ยนลำดับเป็น

->activation->dynamic->pass->drop->sdrop->reject->alert->log

Stream4 option การทำงานใหม่ใน Snort inline

ซึ่งสามารถเปิดการใช้งานในทางเลือก ได้อีก 2 ความสามารถดังนี้

- **inline_state (no arguments)**

เมื่อเปิดใช้งาน Snort inlineจะทำการdrop แพ็คเกตที่ไม่เกี่ยวข้องหรือไม่สอดคล้องกับ TCP sessionที่มีอยู่ และรวมถึงแพ็คเกตที่มี TCP initiator ผิดรูปแบบ

- **midstream_drop_alerts (no arguments)**

โดยค่ากำหนดเริ่มต้นสำหรับ Snort inline จะทำการdrop โดยไม่แสดงการเตือนใดๆ ถ้าหากเราต้องการให้มีการแจ้งเตือนก็ต้องเปิดโหมดการทำงานอันนี้ด้วย

ความสามารถในการแก้ไขข้อมูลในแพ็คเกต

นั่นคือเราสามารถสั่งให้มีการตรวจสอบ ข้อมูลที่ผ่านเข้าออกที่ผ่านตัว Snort inline ว่ามีค่าที่ตรงกับที่เราตั้งไว้หรือไม่ และถ้ามีก็ทำการเปลี่ยนให้เป็นไปตามที่เราต้องการ เช่น

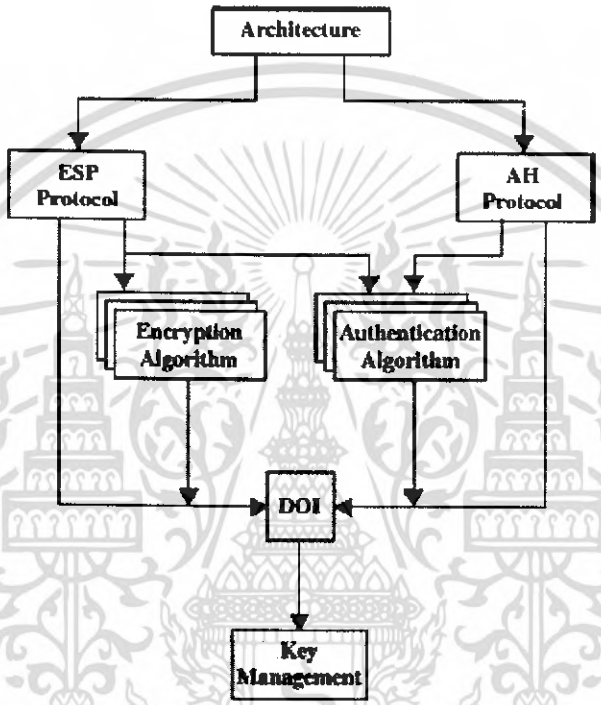
alert tcp any any <> any 80 (msg: "tcp replace"; content:"GET"; replace:"BET";)

alert udp any any <> any 53 (msg: "udp replace"; content: "yahoo"; replace: "xxxxx";)

ดังตัวอย่าง ตัวSnort inline ก็จะดูข้อมูลที่เป็นTCP ที่ใช้งาน port 80 โดยถ้าพบว่ามี ส่วนที่ตรงกับคำว่า GET ก็ให้เปลี่ยนเป็น BET และถ้าพบข้อมูลที่เป็น UDP ที่ใช้งาน port 53 ที่มีข้อมูลตรงกับคำว่า yahoo ก็ให้เปลี่ยนเป็น xxxxx

4.4 IPsec (IP security)

IPsec เป็นส่วนเพิ่มขยายของ Internet Protocol (IP) ในชุดโพรโตคอล TCP/IP IPsec ใช้โพรโตคอล 2 ชุดคือ Authentication Header (AH) และ Encapsulated Security Payload (ESP) เพื่อรองรับการพิสูจน์ตัวตน(Authentication) การรักษาความถูกต้องของข้อมูล (Integrity) และการรักษาความลับ (Confidentiality) ในระดับชั้นของ IP



รูปที่ 4.4 แสดงองค์ประกอบการทำงาน

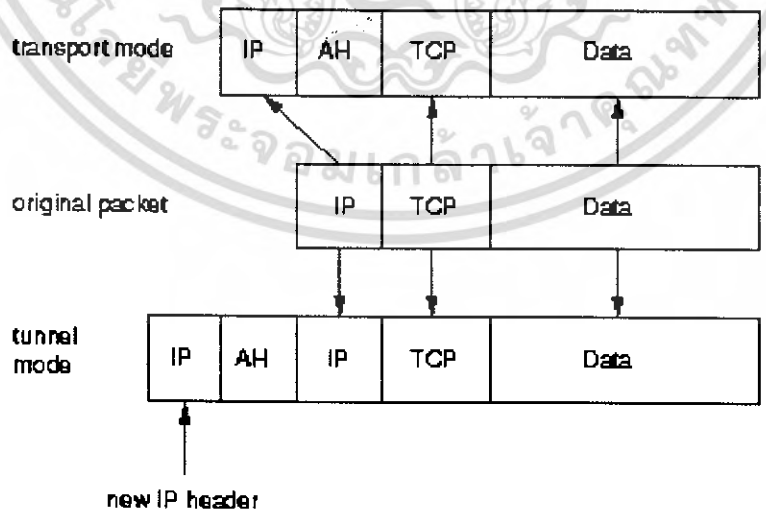
โดยการนำมาใช้งาน การทำส่วน AH จะแยกกับส่วน ESP โดยสามารถใช้ทั้งสองได้เพื่อประโยชน์ด้านความปลอดภัย โดยจะมีความสามารถดังตาราง

	AH	ESP (Encryption Only)	ESP (Encryption and Authentication)
Access Control	Y	Y	Y
Connectionless Integrity	Y		Y
Data Origin Authentication	Y		Y
Rejection of Replayed Packets	Y	Y	Y
Confidentiality		Y	Y
Limited Traffic Flow Confidentiality		Y	Y

ตารางที่ 4.1 ตารางแสดงความสามารถของ AH, ESP และ AH+ESP

โดยการใช้งานสามารถเลือกใช้ได้สองรูปแบบตามรูป

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



รูปที่ 4.5 แสดงรูปแบบแพ็คเกตของ IPsec

การรักษาความถูกต้องของข้อมูลของ IP ดาตาแกรม (IP Datagram) ในชุดโพรโตคอล IPsec ใช้ Hash Message Authentication Codes หรือ HMAC ด้วยฟังก์ชันแฮช เช่น MD5 หรือ SHA-1 ทุกครั้งที่มีการส่งแพ็คเก็ตจะมีการสร้าง HMAC และใช้การเข้ารหัสไปด้วยทุกครั้ง เพื่อให้ปลายทางสามารถตรวจสอบได้ตามหลักการลายเซ็นดิจิตอลว่าต้นทางเป็นผู้ส่งแพ็คเก็ตนั้นมาจริง

ส่วนการรักษาความลับของข้อมูลนั้น จะใช้การเข้ารหัส IP ดาตาแกรมด้วยวิธีการเข้ารหัสด้วยกุญแจสมมาตร ด้วยวิธีการมาตรฐานที่เป็นรู้จักกันดีเช่น 3DES AES หรือ Blowfish เป็นต้น

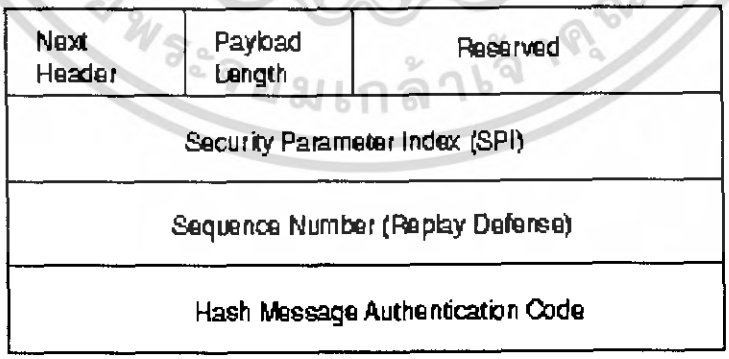
ปัญหาหนึ่งของ IPsec คือการส่งกุญแจที่ใช้ในการเข้ารหัสไปกับแพ็คเก็ต ซึ่งจัดว่าไม่ปลอดภัย นอกจากนี้การแลกเปลี่ยนกุญแจนำไปสู่ปัญหาของการดูแลระบบที่ใช้ IPsec เพราะทั้งระบบต้องสนับสนุนการใช้งานโพรโตคอล IPsec เดียวกัน จะทำอย่างไรให้สามารถส่งกุญแจในการเข้ารหัสไปกับแพ็คเก็ตถ้าไม่มีการเข้ารหัสแพ็คเก็ตแต่อย่างใด เพื่อแก้ปัญหาจึงได้พัฒนาโพรโตคอลในการแลกเปลี่ยนกุญแจหรือ Internet Key Exchange Protocol (IKE)

IKE จะทำการพิสูจน์ตัวตนของปลายทางก่อนการสื่อสาร ในขั้นตอนถัดมาจึงสามารถแลกเปลี่ยนและตกลง Security Association และกุญแจในการเข้ารหัสได้ด้วยวิธีการแลกเปลี่ยนกุญแจตามวิธีการแลกเปลี่ยนกุญแจด้วยการใช้กุญแจสาธารณะ เช่น Diffie-Hellmann เป็นต้น ซึ่งชุดโพรโตคอล IKE จะตรวจสอบกุญแจที่ใช้ในการเข้ารหัสระหว่างการติดต่อสื่อสารเป็นระยะตลอดการสื่อสารข้อมูลที่เกิดขึ้นแต่ละครั้ง

ชุดโพรโตคอล IPsec ประกอบจาก 2 โพรโตคอลหลักสองโพรโตคอลคือ

Authentication Header (AH) และ Encapsulated Security Payload (ESP)

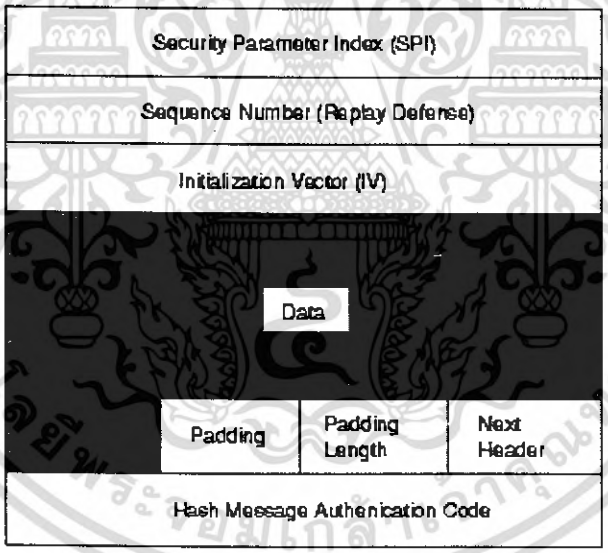
AH หรือ Authentication Header ทำหน้าที่รักษาความถูกต้องของ IP ดาตาแกรม โดยการคำนวณ HMAC กับทุก IP ดาตาแกรมตามรูป



รูปที่ 4.6 แสดง 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 สำหรับใช้ในการถอดรหัสแพ็คเก็ตเมื่อถึงปลายทาง
- Sequence Number ขนาด 32 บิตใช้บอกลำดับของแพ็คเก็ต
- **Hash Message Authentication Code (HMAC)** เป็นค่าที่เกิดจากฟังก์ชันแฮชเช่น MD5 หรือ SHA-1 เป็นต้น
- **ESP หรือ Encapsulated Security Payload** ใช้สำหรับรักษาความถูกต้องของแพ็คเก็ตโดยใช้ HMAC และการเข้ารหัสร่วมด้วย



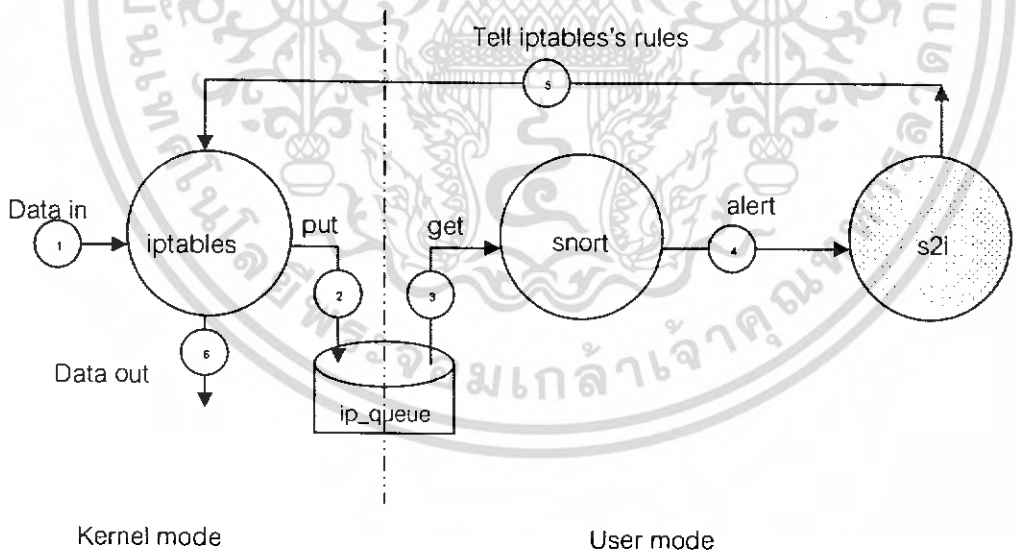
รูปที่ 4.7 แสดง Encapsulated Security Payload

- **Security Parameter Index (SPI)** ใช้ในการกำหนด Security Association (SA) ระบุ ESP ที่สอดคล้องกัน
- **Sequence Number** ระบุลำดับของแพ็คเก็ต

- **Initialization Vector (IV)** ใช้ในกระบวนการเข้ารหัสข้อมูล ป้องกันไม่ให้สองแพ็คเก็ตมีการเข้ารหัสที่ซ้ำกันเกิดขึ้น
- **Data** คือข้อมูลที่เข้ารหัส
- **Padding** เป็นการเติม Data เพื่อให้ครบจำนวนไบต์ที่เข้ารหัสได้
- **Padding Length** บอกความยาวของ Padding ที่เพิ่ม
- **Next Header** กำหนดเฮดเดอร์ถัดไป
- **HMAC** ค่าที่เกิดจากฟังก์ชันแฮชขนาด 96 บิต

4.5 S2I (S2I: Snort command to iptables)

เป็นโปรแกรมที่สร้างจากภาษาคอมไพเลอร์ เพื่อให้สามารถแปลงการแจ้งเตือนของตัวระบบตรวจจับผู้บุกรุก ให้แปลงเป็นคำสั่งของไฟร์วอล (ไอพีเทเบิล) ได้อย่างเหมาะสม เพื่อให้ชิ้นข้อมูลที่มาจากผู้ใช้งานปกติไปยังส่วนที่ถูกต้องตามต้องการ และในส่วนของผู้บุกรุกเองนั้นก็ได้ส่งไปยังเครื่องกับดัก ตัวเอสทูไอสามารถกำหนดเปลี่ยนแปลงค่า ต่างๆ ได้เพื่อให้เกิดความเหมาะสมในการใช้งานกับระบบเครือข่ายนั้นๆ ซึ่งมีโปรแกรมเสริมดังนี้



รูปที่ 4.8 แสดงไดอะแกรมแสดงการทำงานของ S2I

4.6 Iptables

ไอพีเทเบิลนั้นเป็นโปรแกรมไฟร์วอลแบบState Full Inspector ที่ทำงานใน Kernel-Mode โดย iptables จะมีหน้าที่ในการจัดการเชื่อมต่อ และการอนุญาตหรือไม่อนุญาตให้ชิ้นข้อมูลต่างๆ สามารถผ่านเข้าออกระบบเครือข่ายได้ ซึ่งในโครงงานนี้ได้นำโปรแกรม iptables มาใช้งานเพื่อให้ชิ้นข้อมูลที่ผ่านเข้ามาในระบบเครือข่ายถูกส่งไปให้ส่วนที่โปรแกรมตรวจจับผู้บุกรุกที่ทำงานในยูสเซอร์โหมดนำชิ้นข้อมูลเหล่านั้นไปพิจารณาต่อไป โดยชิ้นข้อมูลจะถูกส่งผ่านไปทางโมดูลที่ชื่อว่า ip_queue และอีกหน้าที่หนึ่งของ iptables คือการจัดเส้นทางการเชื่อมต่อให้กับผู้ใช้ที่ถูกจำแนกแล้วว่าเป็นผู้ใช้ปกติหรือเป็นผู้บุกรุกระบบ ให้เข้าไปใช้งานเครื่องให้บริการจริง หรือเครื่องกับดัก โดยตัวไอพีเทเบิลจำเป็นจะต้องทำงานร่วมกับ Snort_inline โดยผ่านการเชื่อมให้สามารถทำงานประสานกันได้โดย S2I

Linux สามารถใช้งานเป็นไฟร์วอลได้ตั้งแต่เคอร์เนล 1.1 ซึ่งเป็นเวอร์ชันแรก โดย Alan Cox ใช้ชื่อว่า ipfw (จาก BSD) ต่อมา Linux 2.0 ได้ถูกพัฒนาและปรับปรุงได้เครื่องมือที่มีชื่อว่า ipfwadm โดยเครื่องมือชิ้นนี้อนุญาตให้ผู้ใช้สามารถควบคุม filtering rule ได้ และต่อมา Linux 2.2 ก็ได้สร้างเครื่องมือตัวใหม่ชื่อ ipchains ซึ่งเผยแพร่ในปี 1998 โดย Rusty Russel และทีมงาน ทั้งนี้ ipchains นี้ถือได้ว่าเป็นพัฒนาการขั้นที่สามของ Linux Firewall จวบจนกระทั่งในปัจจุบัน ก็มี netfilter และ iptables ซึ่งถือว่าเป็นพัฒนาการขั้นที่สี่ของ Linux Firewall

Netfilter นั้นเป็นชื่อใหม่ของโค้ดที่ทำหน้าที่เป็น packet handler(stateful inspection) ใน Linux kernel 2.4 (จริงคือเวอร์ชัน 2.3.15 และเวอร์ชันต่อๆ มา) ซึ่งได้ถูกออกแบบและปรับปรุงใหม่จากเวอร์ชันก่อนหน้านี้ เป็นเรื่องที่น่ายินดีคือ netfilter นั้นสามารถทำงานย้อนหลังร่วมกับ ipchains และ ipfwadm ได้ และคำสั่งในการเรียกใช้งานคือ iptables

ความแตกต่างระหว่าง iptables และ ipchains

- ชื่อของ built-in chain (ประกอบไปด้วย INPUT, OUTPUT, FORWARD) เปลี่ยนจากตัวอักษรเล็ก (lowercase) เป็นตัวอักษรใหญ่ (uppercase)
- การใช้งานที่ต้องระบุ port ทั้ง TCP และ UDP นั้น ต้องใช้คำว่า --source-port หรือ --sport (--destination-port หรือ --dport) และต้องใช้ตามหลังจาก -p tcp หรือ -p udp
- TCP -y flag เปลี่ยนเป็น --syn และต้องใช้ร่วมกับ -p tcp
- target จาก DENY เปลี่ยนเป็น DROP

- chain ที่ไม่มี rule ใดๆ เลยก็สามารถทำงานได้
- การทำ zcroing built-in chain จะทำให้ byte counter ถูกล้างค่าไปด้วย
- ชื่อของ chain ยาวสูงสุดได้ 31 ตัวอักษร
- MASQ เปลี่ยนเป็น MASQUERADE และมีรูปแบบการใช้งานเปลี่ยนไป รวมทั้ง REDIRECT ก็มีการเปลี่ยนแปลงรูปแบบใหม่

รูปแบบการใช้งาน iptables เบื้องต้น

iptables จะมีรูปแบบการใช้งานดังนี้คือ

iptabies [table] <command> <match> <target/jump>

โดย rule ที่เขียนขึ้นจะเป็นเป็นตัวบอกเคอร์เนลว่าให้กระทำ action อย่างไร ในกรณีที่พบ packet ตรงตามที่ระบุไว้

- **[table]** หมายถึง ตารางหรือ table ที่ต้องการระบุ เช่น iptables -t nat หมายถึงให้ทำงานกับ nat table ในกรณีที่ไม่ได้ระบุตาราง iptables จะถือว่าคำสั่งดังกล่าวระบุถึง filter table โดยอัตโนมัติ
- **<command>** จะเป็นตัวสั่งให้ iptables ทำในสิ่งที่ต้องการ เช่น iptables -A INPUT ซึ่งหมายถึงให้สร้าง rule ต่อท้าย INPUT chain ใน filter table
- **<match>** เป็นส่วนที่ใช้ตรวจสอบว่า packet มีข้อมูลตรง (match) กับที่ระบุไว้หรือไม่ เช่น มี source ip address เป็น 1.2.3.4
- **<target/jump>** เป็นตัวระบุว่าเมื่อเจอ packet ที่ match ก็จะกระทำ (action) ตามที่ระบุไว้ เช่น ถ้า packet ใดมี source ip address เป็น 1.2.3.4 ให้ DROP packet นั้นทิ้งไป

Table

iptables สามารถทำงานได้กับตาราง(table) 3 ตารางหลัก สามารถระบุตารางได้โดยใช้ออปชัน -t ตามด้วยชื่อ table คือ

1. **Filter table** ใช้สำหรับกรอง packet มี 3 built-in chain คือ INPUT, OUTPUT, FORWARD ซึ่งจะได้อธิบายรายละเอียดต่อไป
2. **Nat table** ใช้สำหรับการแปลงแอดเดรส (Network Address Translation) มี 3 built-in chain คือ PREROUTING, POSTROUTING, OUTPUT ซึ่งรายละเอียดจะได้อธิบายต่อไป

3. **Mangle table** เป็นตารางที่ใช้เปลี่ยนแปลงหรือแก้ไข packet เช่น เปลี่ยนค่า TTL, MARK ซึ่งปกติจะใช้ในการทำ routing ที่มีความซับซ้อนสูง มี 2 built-in chain คือ PREROUTING chain (ใช้แก้ไข packet ก่อนที่จะเข้าสู่ไฟร์วอลและก่อนเข้าสู่ routing decision) และ OUTPUT chain (ใช้แก้ไข packet ที่ถูกสร้างโดยไฟร์วอลก่อนที่มันจะถูกส่งไปยัง routing decision) ทั้งนี้ไม่สามารถทำ network address translation หรือ masquerading ที่ table นี้ได้ และในเอกสารฉบับนี้จะไม่กล่าวถึง mangle อีก เนื่องจากเป็นส่วนที่ไม่นิยมนำไปใช้งาน

Match

การตั้งเงื่อนไขของการ match นั้นจะต้องอาศัยความเข้าใจในเรื่อง IP, TCP, UDP, และ ICMP มาบ้างพอสมควร จึงจะสามารถตั้งเงื่อนไขที่เหมาะสมและตรงตามความต้องการได้ ซึ่งมีรายละเอียดดังนี้

- การระบุ **source, destination IP address**

  สามารถระบุ source ip address ของ packet โดยใช้ -s หรือ **--source** หรือ **--src** และสำหรับ destination ip address ก็ใช้ **-d** หรือ **--destination** หรือ **--dst** การระบุไอพีแอดเดรสนั้นสามารถทำได้ 4 แบบด้วยกันคือ

  1. ใช้ชื่อเต็มแทน เช่น localhost หรือ www.nectec.or.th
  2. ระบุไอพีแอดเดรสโดยตรง เช่น 127.0.0.1 หรือ 202.44.204.33
  3. ระบุเป็น group ของไอพีแอดเดรส เช่น 202.44.204.0/24 ซึ่งหมายถึงไอพีแอดเดรสตั้งแต่ 202.44.204.0 - 202.44.204.255
  4. หรืออาจจะใช้ 202.44.204.0/255.255.255.0 แทน 202.44.204.0/24 ได้

- การทำ **Inversion**

  ในบางกรณีนั้นหากต้องการระบุเป็น inverse เช่น อนุญาตให้ทุกไอพียกเว้นไอพีที่ระบุไว้ ซึ่งการใช้คำสั่งดังกล่าวสามารถทำได้โดยใช้เครื่องหมาย ! นำหน้า argument ที่ต้องการ (เครื่องหมาย ! หมายถึง NOT) เช่น -p ! TCP ซึ่งจะ match กับโพรโตคอลทุกๆ ตัวที่ไม่ใช่ TCP หรือ -s ! localhost ซึ่งหมายถึง packet ที่มี source ip address อื่นๆ ยกเว้น localhost (127.0.0.1)

- **การระบุโพรโตคอล**

  สามารถระบุโพรโตคอลที่ต้องการได้ดังนี้คือ TCP, UDP, ICMP หรือสามารถใช้ตัวเลขแทนได้ (สำหรับ *NIX อ้างอิงได้จาก /etc/protocols) และยังสามารถใช้ได้ทั้งตัวอักษรเล็กหรือใหญ่ (ใช้ได้ทั้ง tcp และ TCP) เช่น **-p TCP** หรือ **-p ! tcp**

- **การระบุ interface**

  -i หรือ --in-interface ตามด้วยชื่อ interface ใช้เพื่อระบุ incoming interface ซึ่งหมายถึงว่า packet ที่จะ match กับ rule นี้ต้องเข้ามาจาก interface ที่กำหนด เช่น -i eth0 หมายความว่า ทุก packet ที่เข้ามาทาง eth0 จะ match กับ rule นี้ ทั้งนี้ชื่อ interface ที่สามารถใช้ได้นั้น สามารถตรวจสอบได้โดยใช้คำสั่ง ifconfig และ **-o หรือ --out-interface** ตามด้วยชื่อของ interface ใช้เพื่อระบุ outgoing interface ซึ่งหมายถึงว่า packet ที่จะ match กับ rule นี้ กำลังจะเดินทางผ่าน interface ที่ระบุไว้ เช่น -o eth1 หรือ -o ! eth1

ข้อสังเกต

- สำหรับ INPUT chain นั้นไม่มี output interface ดังนั้นหากใช้ -o ร่วมกับ INPUT chain ก็จะไม่มี packet ใดที่ match กับ rule นี้เลย
- ทำนองเดียวกันกับ OUTPUT chain ที่ไม่มี input interface ดังนั้นหากใช้ -i ร่วมกับ OUTPUT chain ก็ไม่มีประโยชน์อันใด
- FORWARD chain มีได้ทั้ง input และ output interface
- หากระบุ interface ที่ไม่มีอยู่จริง ก็จะไม่มี packet ใดที่ match กับ rule นั้นเลย
- หากใช้เครื่องหมาย + ร่วมกับ interface เช่น ppp+ นั้นจะหมายถึงทุกๆ ppp interface เช่น ppp0, ppp1

- **fragment packet**

  ในการส่งข้อมูลใน ip network นั้นเป็นเรื่องปกติที่จะเกิดการ fragment ของ packet เนื่องจากขนาดของ packet มีขนาดใหญ่เกินไปที่จะส่งไปในครั้งเดียว จำเป็นต้องมีการแบ่ง packet ออกเป็นหลายๆ ชิ้นทยอยส่งไป ซึ่งเรียกกันว่าการทำ fragment โดยเครื่องปลายทางจะทำหน้าที่ประกอบ fragment packet รวมกันเป็น packet ที่สมบูรณ์ดังเดิม ข้อมูลที่เป็น fragment packet นั้นจะมี header ที่สมบูรณ์แค่ packet แรกเท่านั้น ตัว packet ที่ตามมาจะมีแค่ header บางส่วนคือ ไอพีแอดเดรสเท่านั้น ไม่มีข้อมูลของโพรโตคอลแนบมาด้วย ดังนั้นการตรวจสอบข้อมูล header ของ TCP, UDP, ICMP จึงไม่สามารถทำได้ใน packet ที่สองเป็นต้นมา

หากใช้ NAT บรรดา fragment packet จะถูกประกอบเข้าด้วยกันจนสมบูรณ์ก่อนที่ packet จะเข้าไปถึง packet filtering ดังนั้นจึงไม่มีความจำเป็นที่จะต้องกังวลเกี่ยวกับ fragment packet

ดังนั้นถ้าไม่ได้ใช้ NAT ก็ควรทำความเข้าใจไว้ว่า iptables มีกระบวนการในการทำงานกับ fragment packet อย่างไร หลังจากที่ fragment packet แรกผ่านเข้ามาแล้ว iptables สามารถตรวจสอบได้ว่าจะอนุญาตให้ผ่านหรือไม่ ในขณะที่ fragment packet ที่สองและหลังจากนั้นที่ตามมานั้น จะไม่สามารถ match กับ rule ใดๆ เลย เช่น -p TCP --sport www หรือแม้แต่ -p TCP --sport ! www

อย่างไรก็ตาม สามารถเขียน rule ให้ตรวจสอบทั้ง fragment packet ตัวที่สองและหลังจากนั้นที่ตามมาได้ด้วยการใช้ -f หรือ --fragment ทั้งนี้อาจจะเขียนในทางตรงข้ามคือไม่ต้องตรวจสอบ fragment packet ที่สองและหลังจากนั้นโดยใช้ ! -f ก็ได้ ทั้งนี้โดยปกติแล้วมักจะปล่อยให้ fragment packet ผ่านไป เนื่องจากถ้าสามารถ DROP ตัว fragment packet ตัวแรกได้แล้ว มันก็ไม่สามารถถูกประกอบที่เครื่องปลายทางได้ แต่ทั้งนี้ fragment packet ที่ถูกปล่อยไปดังกล่าวอาจจะทำให้เครื่องที่ได้รับ hang หรือ crash ได้ หรืออาจจะเกิดการโจมตีแบบ Denial of Service โดยใช้ fragment packet ได้

คำแนะนำเพิ่มเติม

- ควรมีการป้องกันการปลอมไอพี(IP spoof) สำหรับเครื่องไฟร์วอล ซึ่งสามารถรัน seript ด้านล่างนี้เพื่อป้องกันปัญหาดังกล่าวได้

```
for x in lo eth0 eth1
do
        echo 1 > /proc/sys/net/ipv4/conf/${x}/rp_filter
done
```

แต่ script ดังกล่าวนี้ไม่ได้ป้องกันการปลอมไอพีสำหรับทั้งเครือข่าย ดังนั้นจึงต้องสร้าง rule ไว้ที่ PREROUTING chain เพื่อป้องกันปัญหานี้อีกครั้ง เช่น

```
$IPTABLES -t nat -A PREROUTING -i $INET_IFACE -s 10.0.0.0/8 -j DROP
$IPTABLES -t nat -A PREROUTING -i $INET_IFACE -s 127.0.0.0/8 -j
DROP
$IPTABLES -t nat -A PREROUTING -i $INET_IFACE -s 172.16.0.0/12 -j
DROP
```

SIPTABLES -t nat -A PREROUTING -i $INET_IFACE -s 192.168.1.0/16 -j DROP

การป้องกันการปลอมไอพีดัง rule ด้านบนนี้ สามารถลดปัญหาที่จะเกิดขึ้นจากการโจมตีแบบ Denial of Service เช่น Land attack ได้เป็นอย่างดี

- ควรปิดการทำงานของโพรโตคอล ICMP เพราะถือได้ว่าเป็นโพรโตคอลที่ไม่มีความปลอดภัย การโจมตีที่เกิดขึ้นแบบ Denial of Service ในช่วงที่ผ่านมาก็ใช้ ICMP เป็นหลัก เช่น Smurf attack, Ping flood(Ping of Death) แต่การยกเลิกการใช้งาน ICMP อาจจะก่อให้เกิดความไม่สะดวกในการใช้งาน ซึ่งก็ขึ้นอยู่กับวิจารณญาณของผู้ที่ดูแลระบบเครือข่ายด้วย

- เนื่องจากการสร้าง policy ของแต่ละ chain นั้นสามารถเลือกได้ 2 แบบ คือ DROP และ ACCEPT ซึ่งรูปแบบการเขียน rule สำหรับแต่ละ policy นั้นก็แตกต่างกันเล็กน้อย แต่ระดับของความปลอดภัยที่ได้ถือได้ว่ามีความแตกต่างกันพอสมควรคือ
    - ถ้าหากใช้ policy เป็น DROP แล้ว ก็เสมือนเป็นระบบปิด การสร้าง rule ก็เพียงแค่ ACCEPT สำหรับ port หรือ service ที่ต้องการให้เปิดใช้เท่านั้น
    - ในขณะที่หากใช้ policy เป็น ACCEPT แล้ว ก็เสมือนเป็นระบบเปิด จะต้องสร้าง rule สำหรับปิด service หรือ port บางส่วน ซึ่งมีโอกาสที่จะเกิดความผิดพลาดได้ง่ายกว่า

- ควรมีการทำ hardening OS สำหรับเครื่องที่จะนำมาใช้เป็นไฟร์วอล รวมทั้งจำกัดการเข้าถึงเครื่องนี้จากภายนอก ติดตั้ง service เฉพาะที่จำเป็นเท่านั้น และติดตั้งเครื่องมือช่วยสำหรับผู้ดูแลระบบเพื่อตรวจตราความผิดปกติที่อาจจะเกิดขึ้นในระดับ host based เพราะถ้าผู้บุกรุกสามารถเจาะเข้ามายังเครื่องไฟร์วอลได้แล้วก็เสมือนกับสามารถควบคุมเครือข่ายได้ทั้งหมด

- ติดตั้ง NIDS เช่น Snort ในเครือข่ายทั้งหน้าและหลังไฟร์วอล ซึ่งเครื่องที่จะติดตั้ง NIDS ก็ควรจะมีความแข็งแกร่งเช่นเดียวกับเครื่องที่ติดตั้งไฟร์วอล

- ภายหลังที่สร้าง rule เสร็จแล้ว ควรใช้ port scanner ทำการ scan เครื่องทั้งหมดจากภายนอก โดยให้ scan ทุกโพรโตคอลคือ TCP, UDP และ ICMP scan เพื่อเป็นการตรวจสอบเบื้องต้นว่าไฟร์วอลทำงานตามที่ได้ตั้งไว้หรือไม่

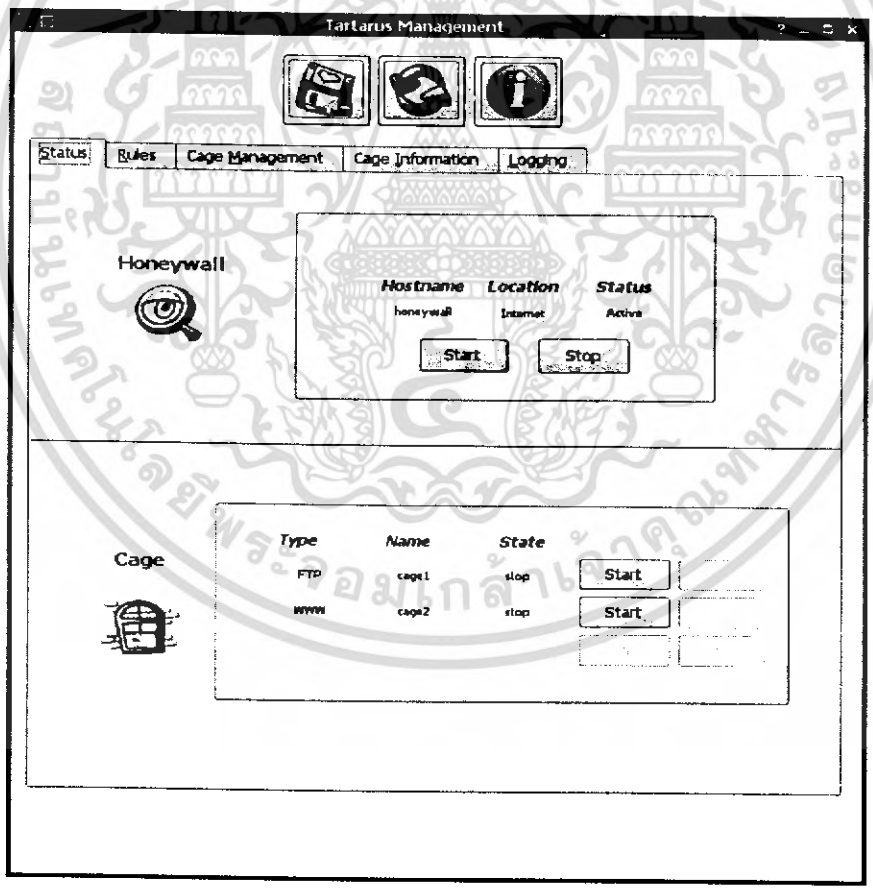
- การสร้าง rule บางอย่างนั้น จะต้องมีความรู้ในเรื่องของโพรโตคอลทั้ง TCP, UDP, ICMP เป็นอย่างดี เพราะไม่เช่นนั้นแล้วก็จะเป็นการเปิดช่องโหว่ในระบบโดยไม่รู้ตัว และ rule ในไฟร์วอลที่ดีนั้นจะต้องไม่มีจำนวนมากเกินไป เพื่อป้องกันไม่ให้เครื่องที่รันไฟร์วอลทำงานหนักเกินไป รวมทั้งป้องกันไม่ให้เกิด human error เนื่องจากความยาวที่มากเกินไป

- ขอให้ผู้ที่นำไฟร์วอลไปใช้งานระลึกไว้เสมอว่า ไฟร์วอลไม่สามารถป้องกันการบุกรุกระบบได้อย่างสมบูรณ์ 100% เพราะไฟร์วอลเป็นเพียงอุปกรณ์ที่ใช้เพื่อลดความเสี่ยงเท่านั้น

บทที่ 5
เครื่องมือที่พัฒนาขึ้นใหม่

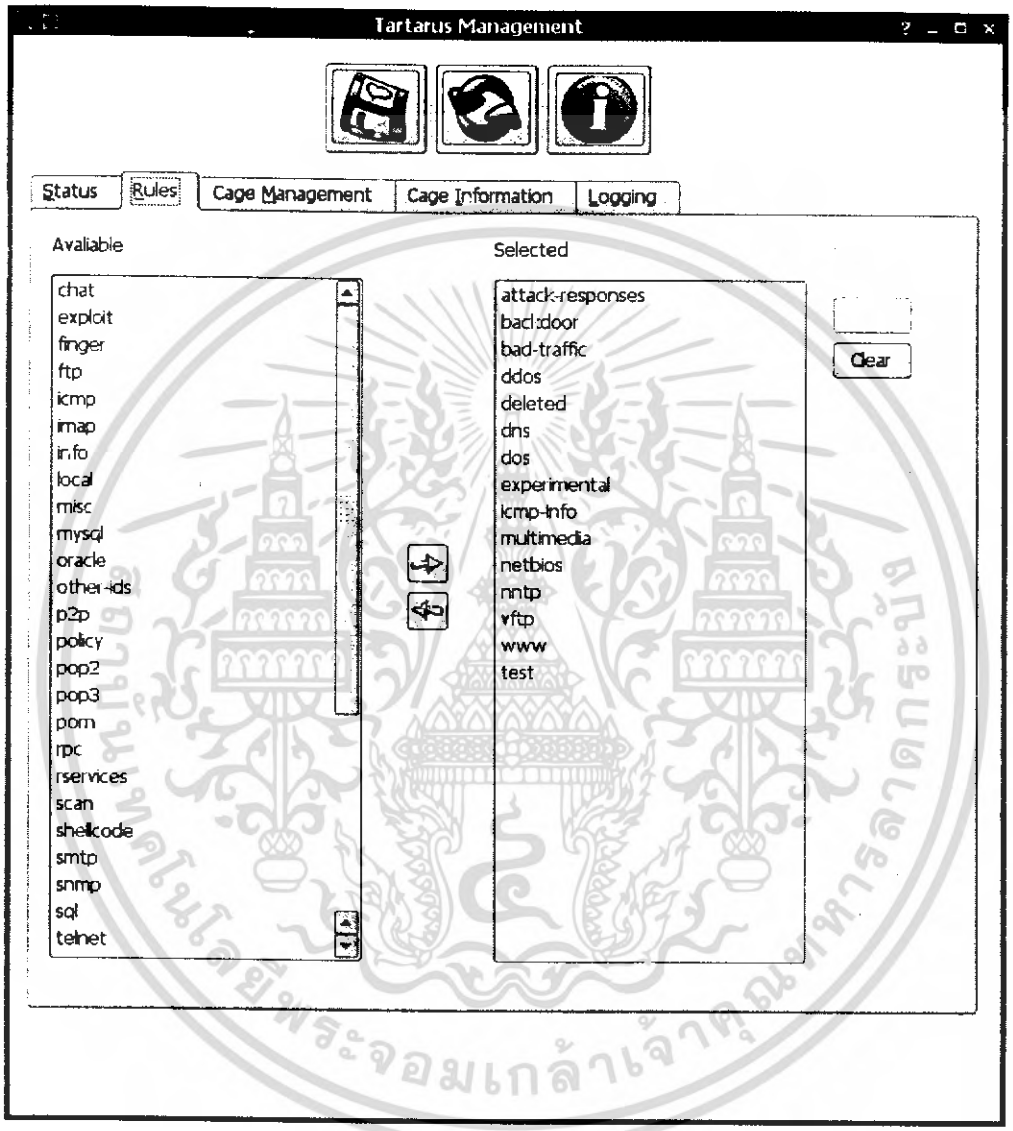
5.1 TM (Tartarus Management)

โปรแกรม Tartarus Management (TM) เป็นโปรแกรม Front-end ที่สร้างขึ้นเพื่อแก้ปัญหาความยุ่งยากในการดูแลและจัดการ ระบบฮันนี่พ็อต ซึ่งมีเครื่องมือและองค์ประกอบที่ทำงานร่วมกันอยู่เป็นจำนวนมาก ซึ่งทำให้เกิดความยุ่งยากในการดูแลเครื่องมือต่างๆให้ทำงานร่วมกันอย่างมีประสิทธิภาพ โดยตัวโปรแกรมTMสามารถควบคุม และกำหนดค่าการทำงานของเครื่องมือทั้งหมดในชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก โดยสามารถดูสถานะของระบบฮันนี่พ็อตและความเป็นไปของเครื่องกับดัก และสามารถเรียกดูบันทึกพฤติกรรมของผู้บุกรุกที่เก็บบันทึกไว้ได้



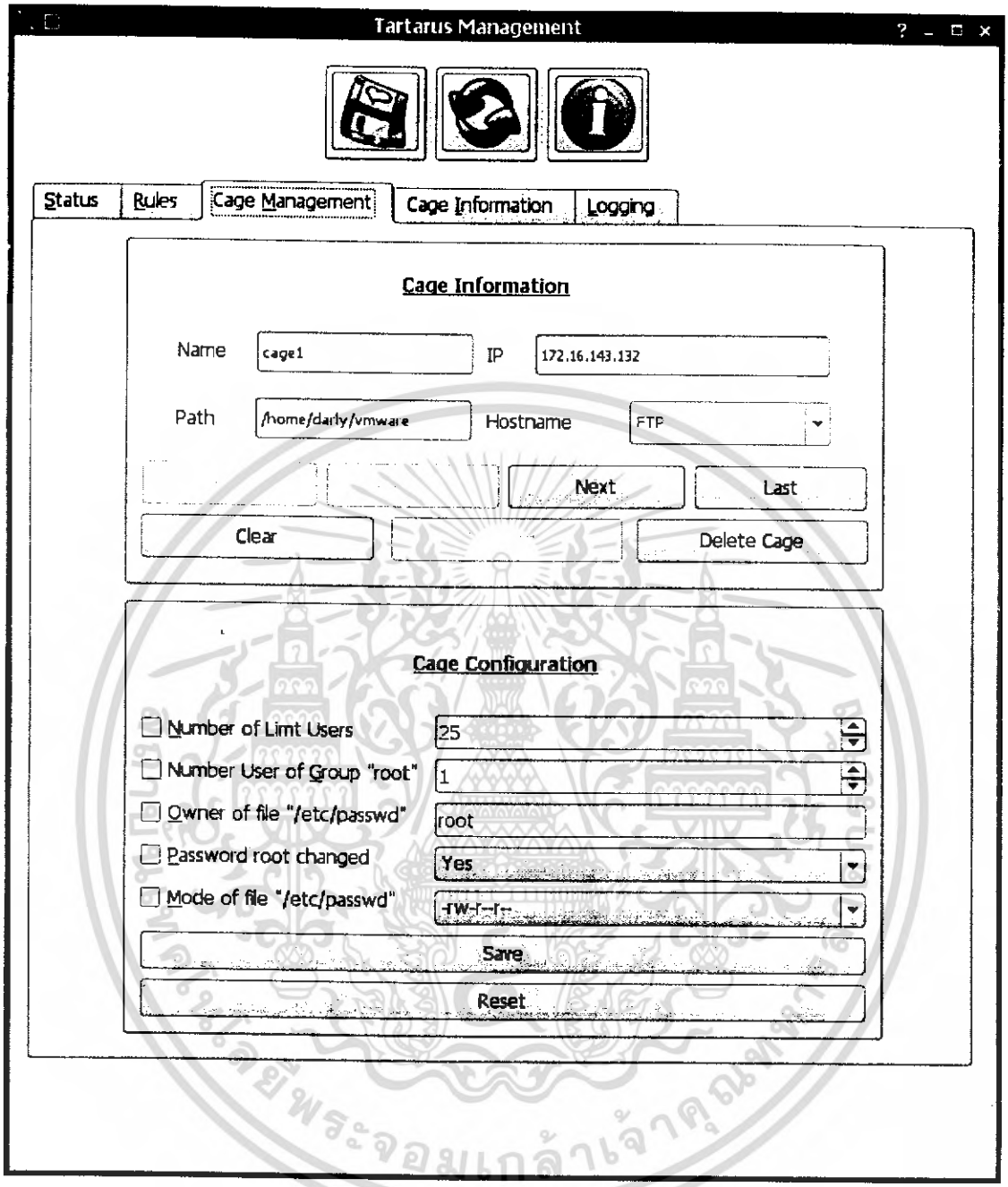
รูปที่ 5.1 แสดงโปรแกรม TM (Tartarus Management)

โดยในหน้าแรกดังรูป 5.1 ที่อยู่ด้านบนจะเป็นส่วนที่แสดงสถานะของระบบตรวจจับผู้บุกรุกในเครื่องเกตเวย์และสถานะของเครื่องกับดัก ซึ่งสามารถสั่งเปิดและปิดระบบตรวจจับผู้บุกรุกและเปิดและปิดเครื่องกับดักได้



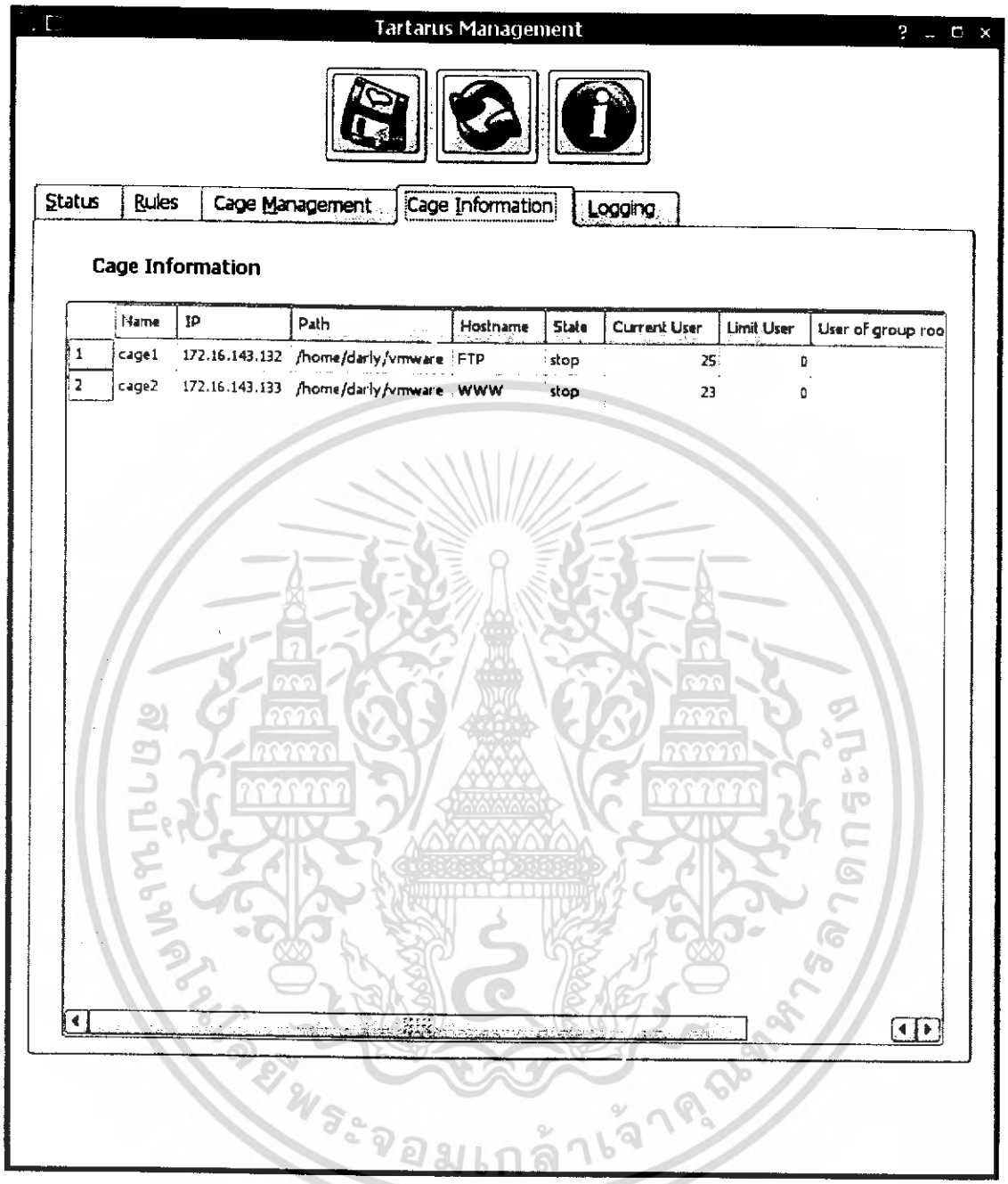
รูปที่ 5.2 แสดงรูปในส่วนที่ใช้ในการกำหนดกฎให้กับระบบตรวจจับผู้บุกรุก

ในรูป 5.2 เป็นส่วนของโปรแกรมที่ใช้ในการเพิ่มและลดกฎให้กับระบบตรวจจับผู้บุกรุกในเครื่องฮันนี่วอลเกตเวย์ ซึ่งจะเป็นกฎของตัว Snort inline โดยสามารถดาวน์โหลดกฎใหม่ๆ มาได้จากเวปไซต์หลักของโปรแกรม Snort



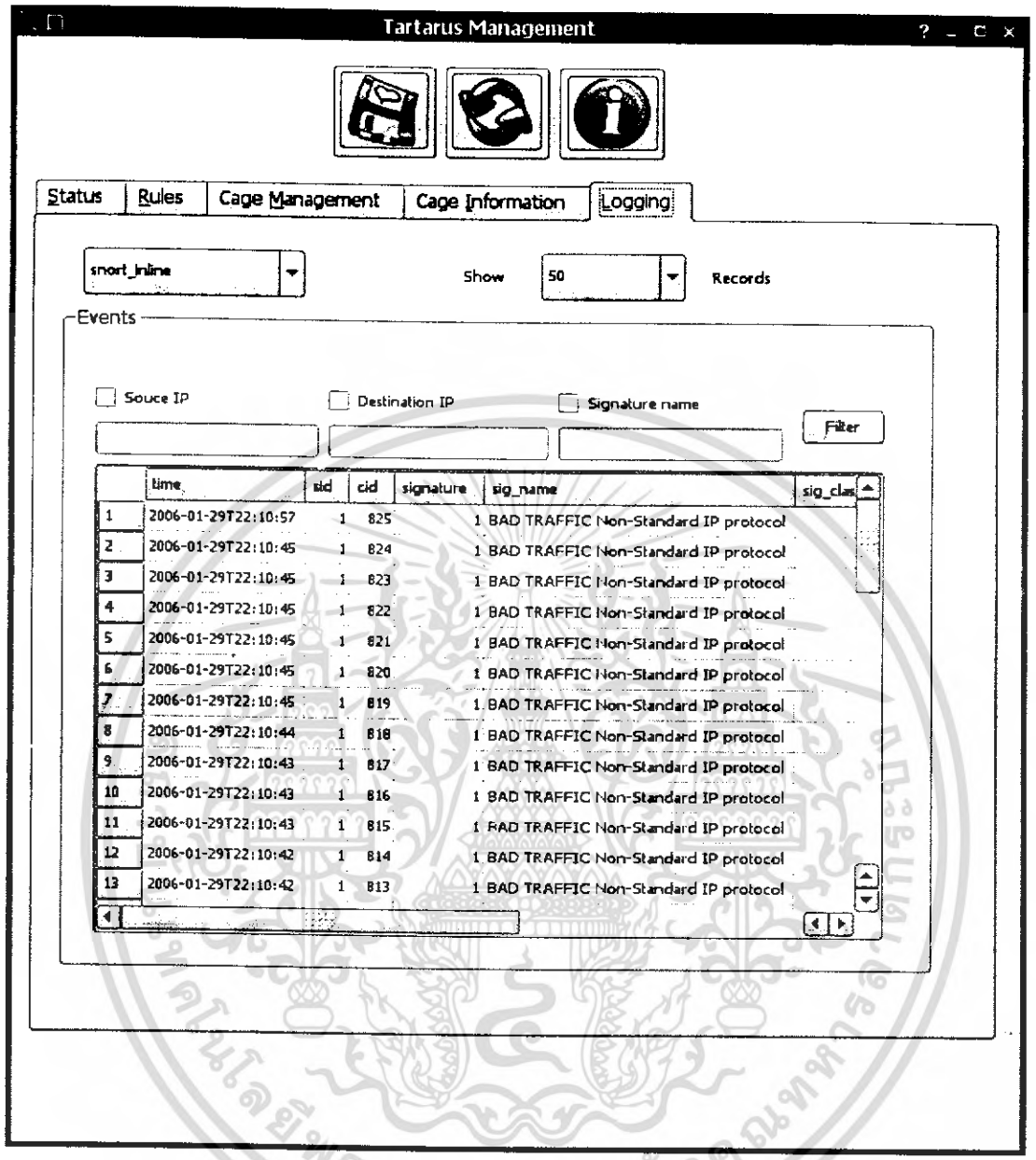
รูปที่ 5.3 แสดงส่วนจัดการเครื่องกับดัก

ในรูปที่ 5.3 เป็นส่วนที่ใช้ในการจัดการกับเครื่องกับดักทั้งในส่วนของการสร้าง ลบ เรียกดูข้อมูลของเครื่องกับดัก และกำหนดการตรวจสอบเครื่องกับดัก ซึ่งจะช่วยให้ผู้ดูแลระบบไม่จำเป็นต้องเข้าไปตรวจสอบด้วยตัวเอง เพราะโปรแกรมจะทำการตรวจสอบให้โดยอัตโนมัติอยู่ตลอดเวลาในการใช้งาน



รูปที่ 5.4 แสดงข้อมูลของเครื่องกับดัก

จากรูปข้างต้นเป็นส่วนที่แสดงข้อมูลของเครื่องกับดักที่เก็บอยู่ในฐานข้อมูลในเครื่อง Logserver ซึ่งจะแสดงถึงชื่อ, หมายเลขไอพีแอดเดรส, สถานะ, พารที่เก็บเครื่องกับดัก และข้อกำหนดที่ใช่ตรวจสอบเครื่องกับดักนั้นๆ



รูปที่ 5.5 แสดง log ที่เก็บอยู่ในฐานข้อมูลในเครื่อง Logserver

จากรูปแสดงหน้าที่ใช้ในการดูข้อมูล log ที่เก็บอยู่ในฐานข้อมูลในเครื่อง Logserver ซึ่งจะประกอบด้วยข้อมูลที่ได้จากโปรแกรม snort_inline, โปรแกรม samhain และโปรแกรม sebek ซึ่งทั้ง 3 โปรแกรมนี้จะทำการเก็บบันทึก ที่แตกต่างกัน

Sebek จะทำการบันทึกการกระทำของผู้บุกรุก เช่น keystroke, ข้อมูลการนำเข้า หรือ ส่งออก, ไฟล์ที่ได้นำเข้าหรือส่งออก

Samhain จะทำการบันทึกการเปลี่ยนแปลงของไฟล์ต่างๆ ในเครื่องกับดัก ซึ่งจะทำให้ทราบผลจากคำสั่งของผู้บุกรุกว่าไปมีผลต่อไฟล์ใด หรือทำการแก้ไข หรือลบไฟล์ใด

Snort inline จะทำการบันทึกข้อมูลของชิ้นข้อมูลที่มีการส่งผ่านเครื่อง Honeywall ที่ทำงานเป็นเกตเวย์ของระบบ

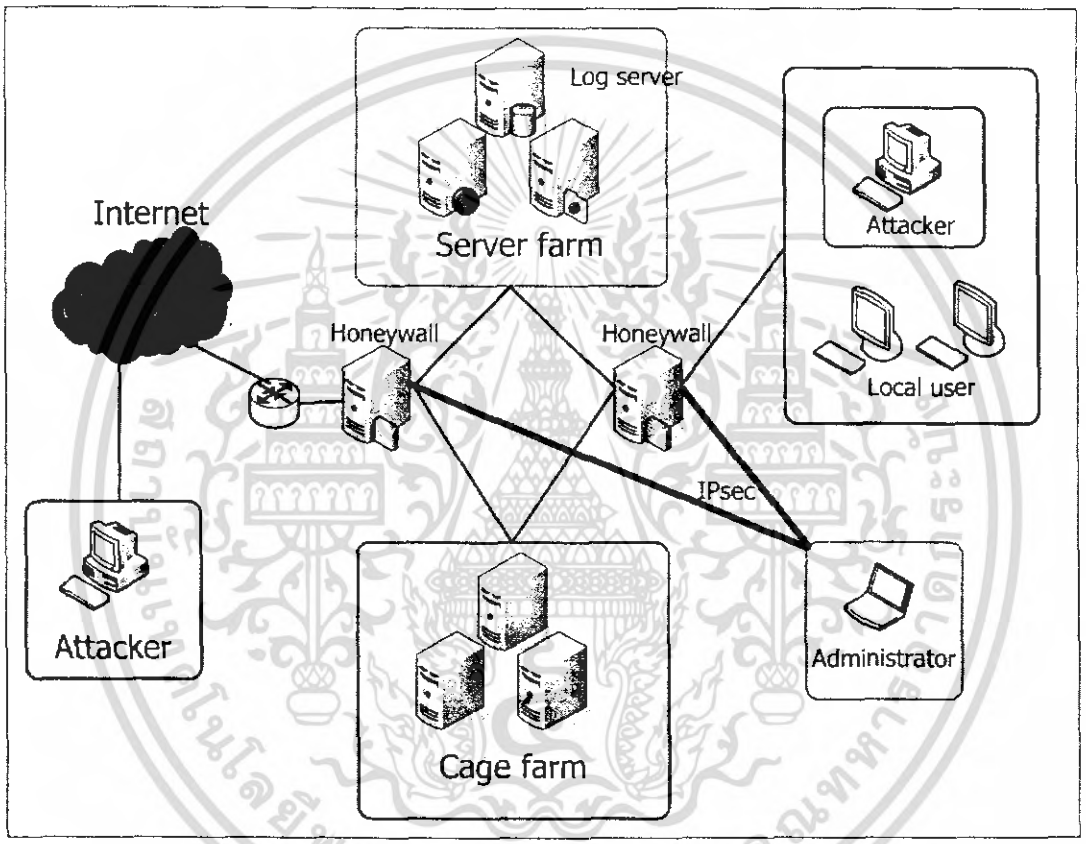
5.2 Cage Prototype

เครื่องกับดักในโครงงานนี้จะประกอบด้วยเครื่องที่ทำหน้าที่เป็น HTTP server และเครื่องที่ทำหน้าที่ FTP server โดยในที่นี้จะทำเป็นตัวอิมเมจของโปรแกรมวีเอ็มแวร์ซึ่งสามารถทำการคัดลอกไปใช้ได้เรื่อยๆ และเครื่องกับดักนั้นได้พัฒนาให้ให้มีความแนบเนียนมากขึ้นโดยการเพิ่มสคริปต์ที่เขียนขึ้นเพื่อไม่ให้ผู้บุกรุกตรวจสอบพบว่าเครื่องที่ผู้บุกรุกกำลังทำงานอยู่ด้วยนั้น แท้จริงแล้วไม่ใช่เครื่องให้บริการจริง โดยได้แก้ไขคำสั่งทั่วไปที่ผู้บุกรุกจะใช้ตรวจสอบ เช่น

- ifconfig เป็นคำสั่งที่ใช้ในการแสดงตัวตนซึ่งจะแสดง ip address, MAC Adress และรายละเอียดอื่นๆ ซึ่งโดยปกติแล้วถ้าใช้คำสั่ง ifconfig แล้วผู้บุกรุกจะทราบว่าตนเองไม่ได้อยู่ในเครื่องที่เป็นเครื่องแม่ข่ายจริงๆ ซึ่งในที่นี้ได้เขียนสคริปต์เพื่อไปเปลี่ยนผลลัพธ์ที่ได้จากการใช้คำสั่ง ifconfig ให้แสดงข้อมูลของเครื่องแม่ข่ายจริง ทำให้ผู้บุกรุกไม่ทราบว่าตนเองถูกส่งให้เข้าไปอยู่ในเครื่องกับดักแล้ว
- ping เป็นสคริปต์ที่ใช้ในการส่ง icmp packet ยังเครื่องอื่นๆ ซึ่งโดยปกติแล้วถ้าใช้คำสั่ง ping ไปยังเครื่องที่ไม่ได้เปิดอยู่จะขึ้น message error ว่า Host Unreachable และจะแสดงไอพีแอดเดรสของเครื่องที่ใช้คำสั่ง ping ด้วย ซึ่งก็จะทำให้ผู้บุกรุก รู้ว่าตนเองไม่ได้อยู่ในเครื่องแม่ข่ายจริง เช่นเดียวกับ ifconfig และในที่นี้ก็ได้เขียนสคริปต์เพื่อเปลี่ยนแปลงผลลัพธ์ที่ได้จากการใช้คำสั่ง ping ให้แสดงไอพีแอดเดรสของเครื่องแม่ข่ายจริงเพื่อไม่ให้ผู้บุกรุกทราบว่าถูกส่งเข้าไปยังเครื่องกับดักแล้ว
- route เป็นคำสั่งที่แสดง routing table ซึ่งถ้าผู้บุกรุกใช้คำสั่งนี้ก็จะทราบได้ว่าตนเองไม่ได้อยู่ในเครื่องแม่ข่ายเช่นกัน ดังนั้นในที่นี้จึงได้เขียนสคริปต์เพื่อไปเปลี่ยนแปลงผลลัพธ์ที่ได้จากคำสั่ง route เพื่อผู้บุกรุกไม่สามารถทราบได้ว่าตนเองถูกส่งไปยังเครื่องกับดักแล้ว
- netstat เป็นคำสั่งที่ใช้ดูการเชื่อมต่อต่างๆ ว่ามีการเชื่อมต่อเข้ามาจากภายนอกหรือไม่ หรือมีการเชื่อมต่อจากภายในออกไปยังภายนอกหรือไม่ และมีพอร์ตใดเปิดอยู่บ้าง

บทที่ 6
การทดลองและผลการทดลอง

6.1 โครงสร้างของระบบ



รูปที่ 6.1 แสดงโครงสร้างโครงระบบที่ทำการทดลอง

จากรูปเราแบ่งการทำงานของระบบเป็นสัดส่วนการทำงาน ดังนี้

- โซน Honeywall ทำหน้าที่เป็นเกตเวย์คอยตรวจสอบแพ็คเกจที่เข้ามา รวมทั้งเป็นส่วนที่จัดการตัวกับดักอย่างอัตโนมัติด้วย โดยในการติดต่อกับทั้งตัว Logserver และการติดต่อกับผู้ดูแลระบบได้ใช้ IP Sec เข้ามาช่วยเพื่อป้องกันการดักจับข้อมูล (sniff) ระหว่างทางอีกด้วย
- โซน DMZ เป็นส่วนของเว็บเซอร์วิสบริการต่างๆ เช่น Logserver, Web server, FTP server เป็นต้น

- โซน Cage เป็นส่วนของเครื่องกับดักที่ได้จัดเตรียมไว้ โดยเราสามารถเพิ่มเติม หรือจัดการเครื่องกับดักได้ผ่านทางโปรแกรม Tartarus Management อีกทั้งยังมีการจัดเก็บบันทึกพฤติกรรมของผู้บุกรุกอีกด้วย ในส่วนนี้จะมีโปรแกรม sebek client ในการตรวจจับพฤติกรรมแล้วส่งไปยังเครื่อง Logserver และโปรแกรม samhain ที่ทำหน้าที่ Integrity Checking เพื่อตรวจสอบว่าถึงจุดอ่อนแอของระบบหรือยัง พร้อมทั้งรายงานไปยังเครื่อง Logserver เช่นกัน

โปรแกรมต้นแบบที่สร้างขึ้นเป็นระบบรักษาความปลอดภัยตามแนวคิดของ Honeypot โดยอาศัยหลักการของไฟร์วอลและระบบตรวจจับการบุกรุกผ่านเครือข่ายเข้าร่วมจำแนก เพื่อให้ตรวจจับบันทึกและวิเคราะห์พฤติกรรมของผู้บุกรุกได้ง่าย ระบบนี้จะล่อหลอกผู้บุกรุกให้เข้าใจผิดว่าเป็นระบบที่อ่อนแอแต่กลับเป็นระบบที่มีการเฝ้าดูพฤติกรรมทุกระยะเมื่อผู้บุกรุกเข้าสู่ระบบ ระบบจะจัดวิถีทางให้ผู้บุกรุกดำเนินไปตามความเหมาะสม ตลอดช่วงดังกล่าวก็บันทึกเหตุการณ์ ที่ผู้บุกรุกกระทำ และเมื่อผู้บุกรุกออกจากระบบทุกอย่างก็จะกลับคืนสู่ภาวะเดิมเสมือนไม่มีการบุกรุกเกิดขึ้น อีกทั้งเมื่อตัวกับดักอ่อนแอมากเกินที่กำหนดแล้ว ตัวกับดักจะทำการสับเปลี่ยนเครื่องกับดักไปยังเครื่องใหม่ได้โดยอัตโนมัติ ซึ่งผู้บุกรุกไม่อาจทราบได้เลยว่าตนเข้าไปบุกรุกระบบที่ถูกจัดไว้เฉพาะและเก็บบันทึกหลักฐานเหตุการณ์ต่างๆ เรียบร้อยแล้ว

6.2 ขั้นตอนการทดสอบชุดโปรแกรม

6.2.1 ผู้ใช้เข้ามาด้วยการบุกรุก

- snort นำเนื้อความที่ตรวจจับได้ตามกฎที่ตั้งไว้ส่งเข้าไปยังเครื่อง Logserver ผ่านช่องทางของ IP Sec
- โปรแกรมสื่อสารกลางคิวรี่ข้อมูลจากเครื่อง Logserver เพื่อตั้งกฎให้ iptables จัดการให้ผู้บุกรุกเข้าไปยังเครื่องกับดักที่ได้จากการคิวรี่จากเครื่อง Logserver
- โปรแกรมสื่อสารกลางตั้งกฎ iptables ให้ไปยังเครื่องกับดักที่ทำหน้าที่เดียวกับเครื่องที่ผู้บุกรุกต้องการใช้งาน

6.2.2 กรณีเครื่องกับดักถึงจุดอ่อนแอ

- โปรแกรมสื่อสารกลางทำการเปิดเครื่องกับดักเครื่องใหม่ที่ได้จากการคิวรี่มาจาก Logserver
- ทำการตั้งกฎ iptables ใหม่เพื่อให้ผู้บุกรุกไปยังเครื่องกับดักเครื่องใหม่ที่ได้เตรียมไว้

- ทำการแช่แข็งเครื่องกับดักเครื่องเก่า เพื่อนำมาศึกษาพฤติกรรมของผู้บุกรุก

6.2.3 ผู้บุกรุกเข้ามายังกับดัก

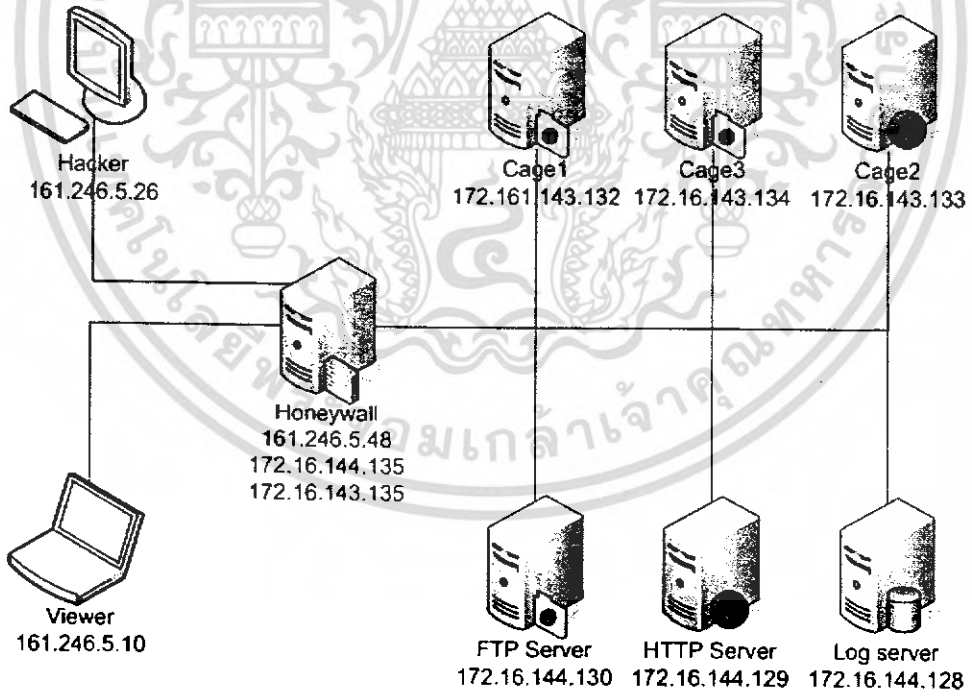
- กรณีที่เข้ามายังช่องทางที่มีบริการจริง(service) ให้ติดต่อกับบริการจริงโดยตรง
- sebek client ตรวจจับพฤติกรรมและส่งไปยัง sebek server เพื่อบันทึกพฤติกรรมไปยังเครื่อง Logserver
- samhain ทำการเช็ค Integrity ของไฟล์อยู่ทุกช่วงเวลา และบันทึกผลที่ได้ไปยังเครื่อง Logserver

6.2.4 ผู้ใช้เข้ามาอย่างถูกต้อง

- ส่งผู้ใช้เข้าติดต่อกับบริการจริงโดยตรง

6.3 ผลการทดลองมีดังนี้

6.3.1 รายละเอียดของเครื่องในระบบ มีดังนี้



รูปที่ 6.2 รูปแสดงการกำหนดหมายเลขไอพีประจำเครื่องแต่ละเครื่อง

6.3.2 ตรวจสอบสถานะของแต่ละเครื่อง

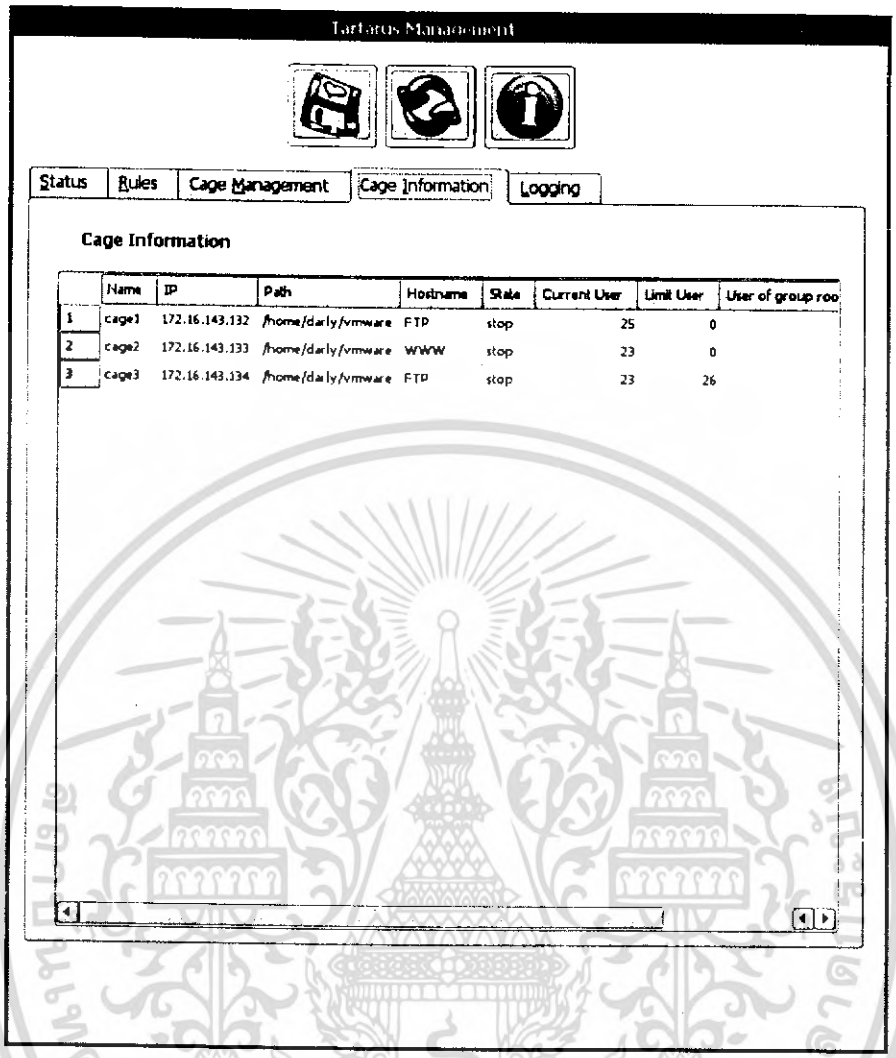
- **เครื่อง Honeywall**

```
honeywall:~# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
honeywall:~# _
```

รูปที่ 6.3 แสดงสถานะกฎของไอพีเทเบิ้ล

เมื่อตรวจสอบสถานะของกฎจะเห็นว่ายังไม่มีกฎใดที่สร้างขึ้นมา

- **เครื่อง Logserver**



รูปที่ 6.4 แสดงรายละเอียดของเครื่องกับดักต่างๆ

จะเห็นว่ายังไม่มีเครื่องกับดักเครื่องใด เปิดใช้งานอยู่

- **เครื่อง FTP Server**

```
ftp:~# netstat -aon
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Timer
tcp        0      0 0.0.0.0:879             0.0.0.0:*               LISTEN      o
ff (0.00/0/0)
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      o
ff (0.00/0/0)
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN      o
ff (0.00/0/0)
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN      o
ff (0.00/0/0)
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      o
ff (0.00/0/0)
tcp6       0      0 :::22                   :::*                    LISTEN      o
ff (0.00/0/0)
udp        0      0 0.0.0.0:873             0.0.0.0:*                           o
ff (0.00/0/0)
udp        0      0 0.0.0.0:876             0.0.0.0:*                           o
ff (0.00/0/0)
udp        0      0 0.0.0.0:111             0.0.0.0:*                           o
ff (0.00/0/0)
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  5      [ ]         DGRAM                    3852   /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     3981   /dev/printer
unix  2      [ ]         DGRAM                    4184
unix  2      [ ]         DGRAM                    3969
unix  2      [ ]         DGRAM                    3878
ftp:~# _
```

รูปที่ 6.5 แสดงรายละเอียดของการเชื่อมต่อ

- **เครื่องกับดัก**

```
cage1:~# netstat -aon |grep 161.246.5.26
cage1:~# _
```

รูปที่ 6.6 แสดงการเชื่อมต่อของเครื่องกับดัก

จะเห็นว่ายังไม่มีการเชื่อมต่อใดๆ เข้ามาทั้งเครื่อง Logserver และเครื่องกับดัก

6.3.3 ผู้บุกรุกทำการบุกรุกเข้ามา

```
root@isag26:/home/zeek# ftp 172.16.144.130
Connected to 172.16.144.130.
220 (vsFTPd 2.0.3)
Name (172.16.144.130:root):
331 Please specify the password.
Password:
530 Login incorrect.
Login failed.
ftp> exit
221 Goodbye.
root@isag26:/home/zeek#
```

รูปที่ 6.7 แสดงถึงการพยายามบุกรุกเครื่อง FTP

จากรูปจะเห็นได้ว่า ผู้บุกรุกพยายามจะทำการล็อกอินด้วยยูสเซอร์ root

- **เครื่อง Honeywall**

```
honeywall:~# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       all  --  isag26.ce.kmitl.ac.th  anywhere            to:172.16.143.132

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
honeywall:~#
```

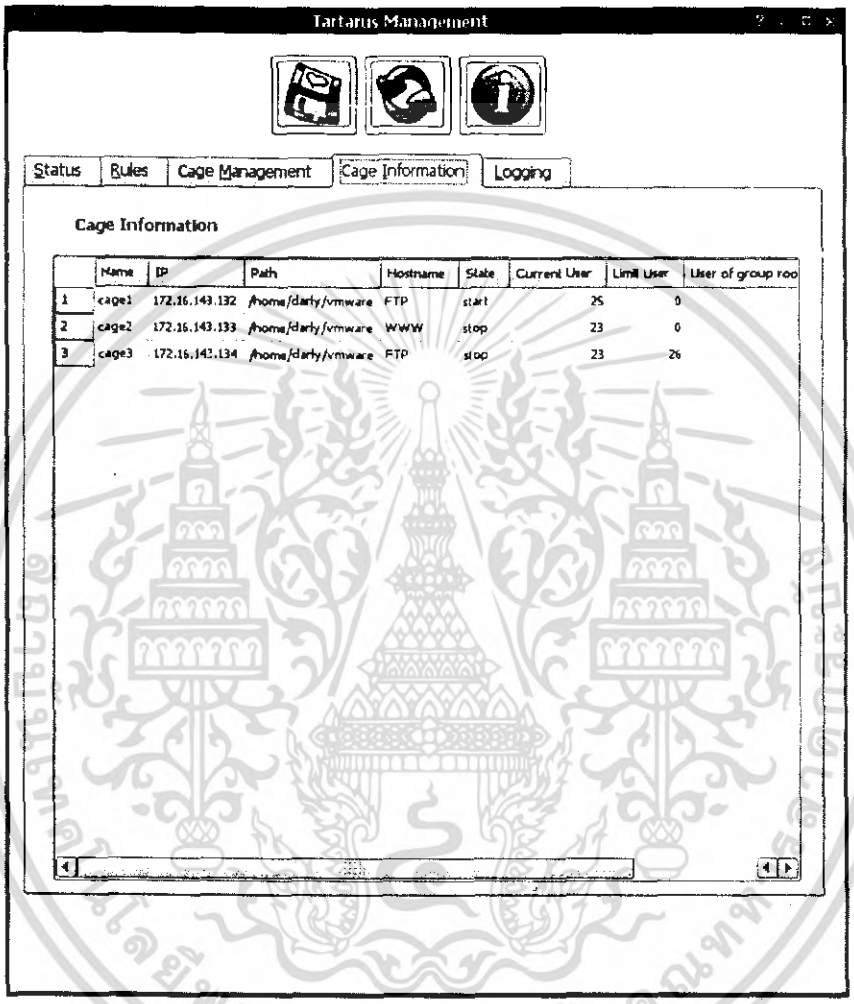
รูปที่ 6.8 แสดงตาราง NAT ของเครื่อง honeywall

เมื่อดูตาราง NAT ของเครื่อง Honeywall แล้ว จะเห็นได้ว่าได้ทำการสร้างกฎให้ผู้บุกรุกไปยังเครื่องกับดักหมายเลข 1 แล้ว

```
root@isag26:/home/zeek# ftp 172.16.144.130
Connected to 172.16.144.130.
220 (vsFTPd 2.0.3)
Name (172.16.144.130:root):
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

รูปที่ 6.9 ผู้บุกรุกได้เข้ามายังเครื่อง FTP ได้สำเร็จ

ผู้บุกรุกได้พยายามทำการล็อกอินด้วยยูสเซอร์ root อีกครั้ง และได้สำเร็จ โดยที่ผู้บุกรุกไม่ทราบว่าตัวเองนั้นได้ถูกล่อหลอกมายังเครื่องกับดักที่เตรียมเอาไว้แล้วนั่นเอง



รูปที่ 6.10 แสดงสถานะของเครื่องกับดักในขณะที่ cage1 ได้ start แล้ว

ในเครื่อง Logserver เมื่อตรวจสอบข้อมูลเครื่องกับดักจะเห็นได้ว่าเครื่องกับดัก cage1 ได้ start ใช้งานแล้ว

- **เครื่องกับดัก**

```
cage1:~# netstat -aon |grep 161.246.5.26
cage1:~# netstat -aon |grep 161.246.5.26
tcp        0      0 172.16.143.132:21       161.246.5.26:4292       TIME_WAIT   t
imewait (20.33/0/0)
tcp        0      0 172.16.143.132:21       161.246.5.26:4298       ESTABLISHEDk
eepalive (7167.46/0/0)
cage1:~# _
```

รูปที่ 6.11 แสดงการเชื่อมต่อของเครื่องกับดักหลังจากผู้บุกรุกเข้ามาแล้ว

จากรูป ได้แสดงให้เห็นว่าผู้บุกรุกได้เข้ามายังเครื่องกับดักเรียบร้อยแล้ว โดยที่ผู้บุกรุกไม่รู้ตัวเลย อีกทั้งเมื่อผู้บุกรุกได้เข้ามายังเครื่องกับดักและพยายามตรวจสอบว่าเป็นเครื่องอะไร จะไม่สามารถแสดงถึงเครื่องที่แท้จริงได้เพราะได้ทำการใส่สคริปต์หลอกไว้เรียบร้อยแล้วในเครื่องกับดัก

```
cage1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
172.16.144.0    *               255.255.255.0   U     0      0        0 eth0
default         172.16.144.135  0.0.0.0         UG    0      0        0 eth0
cage1:~# _
```

รูปที่ 6.12 แสดงการใช้คำสั่ง route เพื่อตรวจสอบตัวตน

```
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=4 Destination Host Unreachab
le
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=5 Destination Host Unreachab
le
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=6 Destination Host Unreachab
le
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=7 Destination Host Unreachab
le
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=8 Destination Host Unreachab
le
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=9 Destination Host Unreachab
le
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=10 Destination Host Unreacha
ble
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=11 Destination Host Unreacha
ble
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=12 Destination Host Unreacha
ble
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=13 Destination Host Unreacha
ble
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=14 Destination Host Unreacha
ble
From cage1.ce.kmitl.ac.th (172.16.144.130) icmp_seq=15 Destination Host Unreacha
ble
_
```

รูปที่ 6.13 แสดงการใช้คำสั่ง ping เพื่อตรวจสอบตัวตน

```
cage1:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:C2:F0:36
          inet addr:172.16.144.130  Bcast:172.16.144.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:153 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:44444 (43.4 KiB)  TX bytes:1938 (1.8 KiB)
          Interrupt:18 Base address:0x1080

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:39 errors:0 dropped:0 overruns:0 frame:0
          TX packets:39 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4032 (3.9 KiB)  TX bytes:4032 (3.9 KiB)

cage1:~# _
```

รูปที่ 6.14 แสดงการใช้คำสั่ง ifconfig เพื่อตรวจสอบตัวตน

นอกจากนี้ทางเครื่องกับดักเองได้บันทึกการกระทำทุกอย่างของผู้บุกรุกเอาไว้ เช่น ผู้บุกรุกทำการใช้คำสั่ง vi อ่านไฟล์ /etc/passwd ดังรูป

```
 1 root:x:0:0:root:/root:/bin/bash
 2 daemon:x:1:1:daemon:/usr/sbin:/bin/sh
 3 bin:x:2:2:bin:/bin:/bin/sh
 4 sys:x:3:3:sys:/dev:/bin/sh
 5 sync:x:4:65534:sync:/bin:/bin/sync
 6 games:x:5:60:games:/usr/games:/bin/sh
 7 man:x:6:12:man:/var/cache/man:/bin/sh
 8 lp:x:7:7:lp:/var/spool/lpd:/bin/sh
 9 mail:x:8:8:mail:/var/mail:/bin/sh
10 news:x:9:9:news:/var/spool/news:/bin/sh
11 uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
12 proxy:x:13:13:proxy:/bin:/bin/sh
13 www-data:x:33:33:www-data:/var/www:/bin/sh
14 backup:x:34:34:backup:/var/backups:/bin/sh
15 list:x:38:38:Mailing List Manager:/var/list:/bin/sh
16 irc:x:39:39:ircd:/var/run/ircd:/bin/sh
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
18 nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
19 Debian-exim:x:102:102::/var/spool/exim4:/bin/false
20 cage1:x:1000:1000:cage1,,,:/home/cage1:/bin/bash
21 identd:x:100:65534::/var/run/identd:/bin/false
22 sshd:x:101:65534::/var/run/sshd:/bin/false
23 samhain:x:1001:1001::/home/samhain:
24 yule:x:1002:1002::/home/yule:
cage1:~/sebek-linux-2.1.7# vi /etc/passwd_
```

รูปที่ 6.15 แสดงการใช้คำสั่ง vi ไฟล์ /etc/passwd

Sebek — Tue, 31 Jan 2006 22:08:30 +0700

Details	IP	PID	UID	COMMAND	FD	DATA
⊖	172.16.143.132	758	0	bash	0	[2006-01-31 02:03:50]# [U-ARROW]
⊖	172.16.143.132	777	0	vi	0	[2006-01-31 01:58:43]# [ESC] :q[BS] [BS] [ESC] :q
			0	vi	3	[2006-01-31 01:58:30]# \000
⊖	172.16.143.132	775	0	clear	3	[2006-01-31 01:57:55]# \000
⊖	172.16.143.132	4778	0	sh	255	[2006-01-30 17:18:19]#

รูปที่ 6.16 แสดงข้อมูลที่ได้จากการเก็บพฤติกรรมมาจากเครื่องกับดัก

จะเห็นได้ว่าทางเครื่อง Logserver (sebek server) ได้ทำการบันทึกพฤติกรรมที่ผู้บุกรุกกระทำทุกขั้นตอน

ต่อมาเมื่อผู้บุกรุกได้กระทำการที่ส่งผลให้เครื่องกับดักอ่อนแอ (ในที่นี้คือเปลี่ยนแปลงพาสเวิร์ด root) โปรแกรมสื่อสารกลางจะทำการสับเปลี่ยนผู้บุกรุกไปยังเครื่องกับดักเครื่องใหม่แล้วทำการแช่แข็ง (suspend) เครื่องกับดักเครื่องเก่าเอาไว้ เพื่อศึกษาพฤติกรรม หาแนวทางป้องกันต่อไป

```
root:$1$os9oUeWV$c./cjKgFgwy3QehR5PuLI1:13164:0:99999:7:::
daemon:*:13164:0:99999:7:::
bin:*:13164:0:99999:7:::
sys:*:13164:0:99999:7:::
sync:*:13164:0:99999:7:::
games:*:13164:0:99999:7:::
man:*:13164:0:99999:7:::
lp:*:13164:0:99999:7:::
mail:*:13164:0:99999:7:::
news:*:13164:0:99999:7:::
uucp:*:13164:0:99999:7:::
proxy:*:13164:0:99999:7:::
www-data:*:13164:0:99999:7:::
backup:*:13164:0:99999:7:::
list:*:13164:0:99999:7:::
irc:*:13164:0:99999:7:::
gnats:*:13164:0:99999:7:::
nobody:*:13164:0:99999:7:::
Debian-exim:!:13164:0:99999:7:::
cage1:$1$.qMr1YdC$eDB3BF8PQt2UJYPv9iCoK1:13164:0:99999:7:::
identd:!:13164:0:99999:7:::
sshd:!:13164:0:99999:7:::
samhain:!:13164:0:99999:7:::
yule:!:13164:0:99999:7:::
cage1:/etc#
```

รูปที่ 6.17 แสดงข้อมูลในไฟล์ /etc/shadow ก่อนถูกเปลี่ยนแปลง

ในไฟล์ /etc/shadow ตอนแรกค่า shadow ของ root คือ1os9oUeWV$c./cjKgFwy3QehR5PuLI1

```
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /etc
250 Directory successfully changed.
ftp> put shadow
local: shadow remote: shadow
200 PORT command successful. Consider using PASV.


421 Service not available, remote server has closed connection

send aborted
waiting for remote to finish abort
ftp> exit
:oot@isag26:/home/seek# ftp 172.16.144.130
Connected to 172.16.144.130.
220 (vsFTPd 2.0.3)
Name (172.16.144.130:root):
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

รูปที่ 6.18 แสดงการที่ผู้บุกรุกพยายามทำการแก้ไขไฟล์ /etc/shadow

จากรูปผู้บุกรุกได้เข้ามาทาง ftp แล้วทำการวางไฟล์ shadow ไว้ที่พาธ /etc จากนั้นจึงทำกา เชื่อมต่อเข้ามาใหม่

```
root:$1$t3.JbBKF$coNnaAyvVERUJtXusglD0.:13170:0:99999:7:::
daemon:*:13164:0:99999:7:::
bin:*:13164:0:99999:7:::
sys:*:13164:0:99999:7:::
sync:*:13164:0:99999:7:::
games:*:13164:0:99999:7:::
man:*:13164:0:99999:7:::
lp:*:13164:0:99999:7:::
mail:*:13164:0:99999:7:::
news:*:13164:0:99999:7:::
uucp:*:13164:0:99999:7:::
proxy:*:13164:0:99999:7:::
www-data:*:13164:0:99999:7:::
backup:*:13164:0:99999:7:::
list:*:13164:0:99999:7:::
irc:*:13164:0:99999:7:::
gnats:*:13164:0:99999:7:::
nobody:*:13164:0:99999:7:::
Debian-exim:!:13164:0:99999:7:::
cage1:$1$.qMr1YdC$eD83BF0PQt2UJYPv9iCoK1:13164:0:99999:7:::
identd:!:13164:0:99999:7:::
sshd:!:13164.0:99999:7:::
samhain:!:13164:0:99999:7:::
yule:!:13164:0:99999:7:::
cage1:/etc# _
```

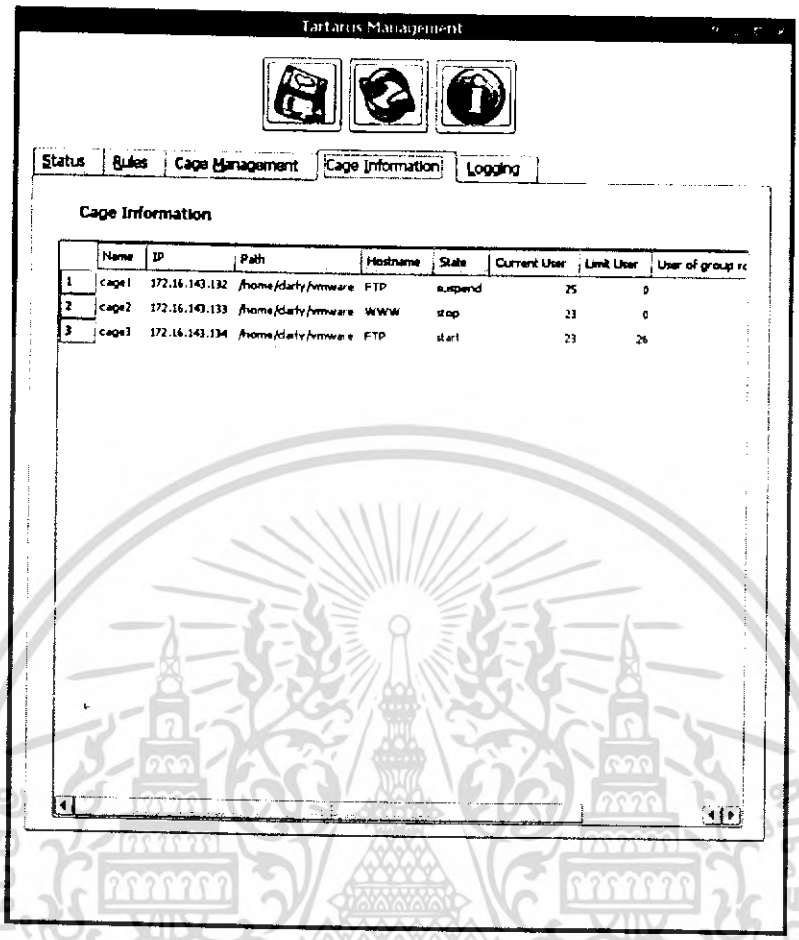
รูปที่ 6.19 แสดงข้อมูลในไฟล์ /etc/shadow หลังเปลี่ยนแปลงแล้ว

จะเห็นได้ว่าส่วนของยูสเซอร์ root ได้เปลี่ยนไปแล้ว คือ 1t3.JbBKF$cpNnaAyvVERUJtXusglD0.

```
honeywall:~# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       all  --  isag26.ce.kmitl.ac.th  anywhere            to:172.16.143.132

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
honeywall:~# iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
DNAT       all  --  isag26.ce.kmitl.ac.th  anywhere            to:172.16.143.134
```

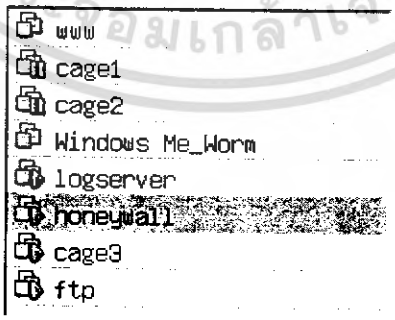
รูปที่ 6.20 แสดงตาราง NAT ของเครื่อง Honeywall

จากรูปแสดงให้เห็นว่าเครื่อง Honeywall ได้ทำการเปลี่ยนแปลงกฎไปยังเครื่องกับดักเครื่องใหม่ที่ได้มาจากการคิวรี่ข้อมูลมาจากเครื่อง Logserver



รูปที่ 6.21 แสดงข้อมูลของเครื่องกับดักในเครื่อง Logserver

ข้อมูลของเครื่องกับดักได้เปลี่ยนแปลงแล้ว คือ เครื่อง cage1 ได้ถูกแช่แข็ง (suspend) และเครื่องกับดักเครื่องใหม่ cage3 ได้ถูก start ขึ้นเพื่อใช้งานแทน



รูปที่ 6.22 แสดงข้อมูลของเครื่องต่างๆ ในระบบ

บทที่ 7
บทวิจารณ์และสรุป

7.1 บทสรุป

โปรแกรมจำแนกและบันทึกพฤติกรรมที่ได้ทำขึ้นในปีการศึกษานี้เป็นการปรับปรุงโปรแกรมต้นแบบที่ได้จัดทำขึ้นในปีการศึกษาที่ผ่านมา โดยปีการศึกษานี้ได้เพิ่มความสามารถในการทำงาน และปรับปรุงการทำงานเดิมให้มีประสิทธิภาพมากยิ่งขึ้น โดยได้พัฒนาเพื่อลดจุดด้อยของระบบฮันนี่พ็อต นั้นคือ เรื่องความปลอดภัยในการใช้งาน เรื่องความแนบเนียนของระบบ และเรื่องปัญหาความยุ่งยากในการใช้งานและการจัดการ โดยในส่วนของเครื่องกับดักได้พัฒนาให้มีการเก็บข้อมูลได้ครอบคลุม เพื่อให้สามารถเก็บรวบรวมข้อมูลได้ระเอียดยิ่งขึ้น และได้มีการเขียนโปรแกรมที่ทำงานเป็นเดมอนในการตรวจสอบสภาพของเครื่องกับดักอยู่ตลอดเวลา เพื่อประโยชน์ในด้านความปลอดภัยในการสั่งหยุดให้บริการเครื่องกับดักเครื่องนั้นๆ เมื่อมีความอ่อนแอถึงระดับที่กำหนด โดยข้อมูลทั้งหมดของระบบได้ถูกนำไปจัดเก็บด้วยระบบฐานข้อมูลกลาง ซึ่งจะทำให้การดูแลและจัดการข้อมูลได้เป็นระบบระเบียบมากขึ้น และได้จัดสร้างเครื่องกับดักที่ได้พัฒนาให้มีความแนบเนียนและปลอดภัยขึ้น นอกจากนี้ได้จัดทำโปรแกรมที่เป็นศูนย์กลางที่ทำงานประสานกับเครื่องมือทั้งหมดที่ใช้ให้สามารถดูแลจัดการได้ทั้งระบบผ่านโปรแกรมเดียวและโปรแกรมมีGUI (Graphics User Interface) ซึ่งทำให้มีความสะดวกต่อการใช้งานมากยิ่งขึ้น

7.2 วิจารณ์สิ่งที่ได้จากโครงงาน

ระบบฮันนี่พ็อตเป็นระบบที่มีความละเอียดอ่อนในการใช้งาน ซึ่งเวลานำไปประยุกต์ใช้นั้นแต่ละองค์กรก็จะมีรายละเอียดที่แตกต่างกันในการติดตั้งและกำหนดค่า ทางผู้พัฒนาจึงได้สร้างชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุกในเชิงกว้าง และให้มีความสามารถของเครื่องมือ ที่ผู้นำไปใช้งาน สามารถเขียนกฎที่เหมาะสมต่อรูปแบบการทำงานขององค์กรนั้นๆได้ โดยชุดโปรแกรมที่ได้สร้างมีความสามารถของ state-full firewall จาก Iptables และมีความสามารถของ IPS จาก Snort_inline จึงสามารถนำชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุกไปใช้งานในการดูแลระบบ ขององค์การจริงๆได้ แต่ด้วยการที่มีการใช้เครื่องกับดัก เป็นการเปิดโอกาสให้ผู้บุกรุกได้กระทำการต่อระบบเสมือน แทนที่จะทำการตัดการติดต่อไปเลยนั้น ก็เป็นจุดเริ่มต้นของความเสี่ยง ซึ่ง

แม้ชุดโปรแกรมจะได้สร้างความปลอดภัยไว้ให้แล้วในระดับหนึ่ง แต่เมื่อเวลามีการนำไปใช้งานจริง ผู้ดูแลระบบที่นำไปใช้ก็จะมีสิทธิ์ทุกอย่างในการกำหนดค่าหรือการจัดตั้งส่วนต่างๆขึ้นมา ซึ่งหากมีการกำหนดค่าที่ไม่เหมาะสม เช่น การกำหนดกฎของSnort inline, กฎของระดับความอ่อนแอของเครื่องกับดัก ก็จะก่อให้เกิดอันตรายต่อองค์กรนั้นๆ ดังนั้นแม้ทางผู้พัฒนาจะได้เพิ่มความปลอดภัยและความสะดวกในการใช้งานแล้ว ผู้ดูแลระบบที่จะนำชุดโปรแกรมจำแนกและบันทึกพฤติกรรมของผู้บุกรุกไปใช้งานนั้นจำเป็นต้องมีความรู้และความใส่ใจระมัดระวังเป็นพิเศษ

7.3 ปัญหาอุปสรรคและแนวทางแก้ไข

- เนื่องจากเครื่องมือแต่ละตัวที่นำมาใช้ มีความต้องการและข้อจำกัดของตัวเอง ดังนั้นการจะนำมาใช้งานร่วมกันได้ จึงจำเป็นมีการปรับปรุงหรือคอมไพล์ใหม่เพื่อให้ใช้งาน ร่วมกับระบบปฏิบัติการและเครื่องมืออื่นๆที่ใช้ได้อย่างมีประสิทธิภาพ

- ปัญหาในการติดต่อระหว่างกันระหว่างภาษาเพิร์ล, ภาษาซี และส่วนของฐานข้อมูล mySQL มีความยุ่งยากและเกิดบั๊กในโค้ดที่เขียนบ่อย จึงใช้เวลาค่อนข้างมากในการจัดการ

- ในการทดลองกับโค้ดหนอนนั้น เนื่องจากหาซอร์สโค้ดหนอน หรือตัวอย่างได้ยาก จึงไม่สามารถทดสอบได้อย่างมีประสิทธิภาพมากนัก

- Sebek version 3.0.3 ทางผู้พัฒนาไม่ได้เผยแพร่ code ในส่วนของการติดต่อ database คู่มือในการใช้งาน จึงต้องมีการศึกษาและแก้ไข source code เพื่อที่จะนำมาประยุกต์ใช้ได้

7.4 แนวทางการพัฒนาต่อ

เนื่องจากระบบของแต่ละองค์กร มีจุดประสงค์และรูปแบบการให้บริการ รวมทั้งการจัดโครงสร้างองค์กรที่แตกต่างกัน ดังนั้นเพื่อพัฒนาให้ชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุกสามารถนำไปใช้ ดูแลความปลอดภัยจริง จะต้องมีการกำหนดค่าต่างๆ ซึ่งเหมาะสมของแต่ละองค์กร จึงควรมีเครื่องมือช่วยให้สามารถกระทำได้ง่ายขึ้น เช่น

- เพิ่มการทำงานของ TM (Tartarus Management) ให้มีความสามารถในการควบคุมหรือจัดการการทำงานของ Firewall ผ่านGUI
- เพิ่มการทำงานของ TM ให้มีความสามารถในการสร้างกฎ สำหรับSnort inline ผ่านทางGUI
- เพิ่มความสามารถในการทำ Auto update Signature เนื่องจากความสามารถในการดักจับโค้ดหนอนนั้นจำเป็นที่จะต้องมีการปรับปรุง Signature ให้ทันสมัยอยู่เสมอ

- เพิ่มเครื่องมือที่ใช้ในการวิเคราะห์การกระทำของผู้บุกรุก ให้กับโปรแกรมTM เพื่ออำนวยความสะดวกในการหารูปแบบในการบุกรุก
- และอื่นๆ เพื่อความปลอดภัยและมีความน่าเชื่อถือของระบบจนสามารถนำไปปกป้ององค์กรจริง

บรรณานุกรม

ตัวอย่างเอกสารอ้างอิงที่เป็นหนังสือ

[1] Lance Spitzner., 2002, "Honeypots Tracking Hackers", Addison Wesley Professional.

ตัวอย่างเอกสารอ้างอิงที่เป็นวิทยานิพนธ์

[2] นพสรณ์ เกษจันทร์, ศุภกร ชะอุ่มดี, "การป้องกันเวบโดยใช้ไอพีเทเบิลส์" ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ปีการศึกษา 2545 คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[3] ศุภกร รวยวาสนา, อานนท์ ลิ้มสถิรานันท์, "ระบบตรวจจับผู้บุกรุกทางคอมพิวเตอร์" ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ปีการศึกษา 2546 คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[4] ดิเรก ยิ้มละมัย, สมพล พูนภัสสร, สันต์อาวี กรรณล้วน "ชุดโปรแกรมจำแนกและบันทึกพฤติกรรมผู้บุกรุก" ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ปีการศึกษา 2547 คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ตัวอย่างเอกสารอ้างอิงที่เป็น Web-site

[5] Lance Spitzner., 2003, "Definitions and Value of Honeypots", [Online] URL: http://www.tracking-hackers.com/papers/honeypots.html

[6] Honeypot Project, 2004, "Honeynet Definitions, Requirements, and Standards", [Online] URL: http://www.honeynet.org/alliance/requirements.html

[7] Honeypot Project, 2005, "Know Your Enemy: GenII Honeypots" [Online] URL: http://www.honeynet.org/papers/gen2/

[8] Honeypot Project, 2005, "Know Your Enemy: Honeynets" [Online] URL: http://www.honeynet.org/papers/honeynet/

[9] Honeypot Project, 2005, "Know Your Enemy: Leaning with VMware" [Online] URL: http://project.honeynet.org/papers/vmware/

[10] Honeypot Project, 2005, "Know Your Enemy: Statistics" [Online] URL: http://www.honeynet.org/papers/stats/

[11] Honeypot Project, 2005, "Know Your Enemy: Defining Virtual Honeynets" [Online] URL: http://www.honeynet.org/papers/virtual/

[12] Peter Harrison, 2005, "Linux Firewalls Using iptables" [Online] URL: http://www.siliconvalleyccie.com/linux-hn/iptables-intro.htm

[13] Suhas A Desai, 2005, "Introduction: IP Spoofing" [Online] URL: http://www.linuxsecurity.com/content/view/120225/49/

[14] Snort Project, 2005, "Snort User Manual" [Online] URL : http://www.snort.org/docs/snort_htmanuals/htmanual_2.4/rc1/

[15] http://www.bleedingsnort.org

[16] Burak DAYIOGLU, 2004, "How to Integrate/Use Bleeding Snort Rules" [Online] URL : http://www.dayioglu.net/?q=node/82

[17] Samhain Project, 2004, "Setting up a client/server samhain system" [Online] URL : http://www.la-samhna.de/samhain/HOWTO-client+server.html

[18] Samhain Project, 2004, "User manual" [Online] URL : http://www.la-samhna.de/samhain/manual/