# ระบบความปลอดภัยบนลีนุกซ์

# Enterprise Security Based On Linux



นางสาวอรกร สุทธิรจิต
นางสาวอุทัยวรรณ รุ่งสว่าง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

# ระบบความปลอดภัยบนลีนุกซ์

**Enterprise Security Based On Linux**


โดย

นางสาวอรกร สุทธิรจิต

นางสาวอุทัยวรรณ รุ่งสว่าง



อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบรักษาความปลอดภัยบนลีนุกซ์

Enterprised Security Based on Linux

ผู้จัดทำ นส.อรกร สุทธิรจิต รหัส 43010526

นส.อุทัยวรรณ รุ่งสว่าง รหัส 43010551

_____ อาจารย์ที่ปรึกษา

( ดร.วรวัฒน์ ลิ้มโกคา )

_____ อาจารย์ที่ปรึกษา

( ดร.ชุติเมษฎ์ ศรีนิลทา )

# ระบบความปลอดภัยบนลีนุกซ์

นางสาวอรกร สุทธิรจิต 43010526
นางสาวอุทัยวรรณ รุ่งสว่าง 43010551
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

**บทคัดย่อ**

ปัจจุบันอินเตอร์เน็ตมีบทบาทสำคัญต่อการดำเนินกิจกรรมต่างๆ เป็นอย่างมาก ปัญหาที่ตามมาก็คือความปลอดภัยของระบบเน็ตเวิร์ก ดังนั้นทำให้ต้องมีการรักษาความปลอดภัย สิ่งที่จะช่วยลดความเสี่ยงนี้ได้ก็คือไฟร์วอลล์ แต่เนื่องจากการมีไฟร์วอลล์นั้นก็ไม่ได้หมายความว่าระบบจะปลอดภัยจากผู้บุกรุกแล้ว ดังนั้น จึงต้องมีระบบตรวจจับผู้บุกรุกขึ้นมา เพื่อสร้างความแข็งแกร่งให้กับไฟร์วอลล์มากยิ่งขึ้นซึ่งในระบบปฏิบัติการ ลีนุกซ์ก็ได้มีโปรแกรมทั้งไฟร์วอลล์และระบบตรวจจับผู้บุกรุกมาด้วยแล้ว

แต่เนื่องจากโปรแกรมทั้งไฟร์วอลล์และระบบตรวจจับผู้บุกรุกมีการเขียนคำสั่งที่ยุ่งยากและซับซ้อน ดังนั้นโครงงานนี้จึงเป็นการสร้างกราฟฟิกยูสเซอร์อินเตอร์เฟสขึ้นมาเพื่อให้ผู้ใช้สามารถใช้งานโปรแกรมได้ง่ายขึ้น

# Enterprise Security Based On Linux

Orakorn Suthirajit 43010526
Uthaiwan Rungsawang 43010551
Dr. Worawat Limpoka Advisor

## ABSTRACT

Today internet plays a very important role on our many activities. Problem that follows is the security on network system. What is needed is the security system to minimize potential risk. Firewall program can answer to this question but it does mean that the system will be free against the intruders.

As a result, it is essential to adopt an intrusion detection system to guard against the uninvited quests. On Linux operating system, it has already deployed the intrusion detection system combined with Firewall program to curve the intruders.

As both Firewall and intrusion detection programs system require confusing and complicate written instruction, this project 'Graphic User Interface' has been created to make it simple and more friendly for the users.

# กิตติกรรมประกาศ

ปริญญานิพนธ์นี้จะไม่สามารถเสร็จสมบูรณ์ได้หากไม่ได้รับความช่วยเหลือ และความร่วมมือจากหลายๆฝ่ายด้วยกัน ซึ่งพวกเราต้องขอขอบคุณ ได้แก่

ขอขอบคุณอาจารย์ที่ปรึกษา ดร.วรวัฒน์ ลิ้มโภคา ที่ให้คำปรึกษาและคำแนะนำจนกระทั่งโครงงานนี้เสร็จสิ้นลง

ขอขอบคุณอาจารย์ทุกท่านในสถาบันนี้ที่ได้สอนสั่งและให้ความรู้ตลอด 4 ปีที่ผ่านมา

ขอขอบคุณห้องปฏิบัติการ ESL , OLALA ที่เอื้อเฟื้อสถานที่ในการดำเนินงาน และสมาชิกห้อง ESL ที่ให้ความบันเทิง และคอยซื้อขนมมาแบ่งกันกินยามดึกดื่น

ขอขอบคุณพี่บีม ,เบริ์ด ,พี่หวาย,ดุ๊ก ที่ให้แนะนำและความช่วยเหลือในโครงงานนี้เวลาเกิดปัญหาขึ้นมา

และต้องขอขอบคุณบุคคลที่สำคัญที่สุดคือ บิดา มารดา ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา

นางสาวอรกร สุทธิรจิต

นางสาวอุทัยวรรณ รุ่งสว่าง

# สารบัญ

# สารบัญ(ต่อ)

# สารบัญ(ต่อ)

# สารบัญภาพ

## สารบัญภาพ(ต่อ)

**สารบัญภาพ(ต่อ)**

## สารบัญภาพ(ต่อ)

# บทที่ 1
# บทนำ

## 1.1 ความสำคัญและที่มา

เทคโนโลยีอินเตอร์เน็ต และเครือข่ายได้พัฒนาขึ้นทุกที องค์กรต่างๆมีการใช้งานคอมพิวเตอร์และเครือข่ายเพิ่มมากขึ้น ปัญหาที่ตามมาก็คือ ปัญหาความปลอดภัยในระบบคอมพิวเตอร์ ทั้งปัญหาความปลอดภัยของผู้ใช้งาน และปัญหาความปลอดภัยของตัวระบบคอมพิวเตอร์เอง จึงได้มีไฟร์วอลล์(Firewall) ขึ้น เพื่อป้องกันระบบขึ้นมา แต่ด้วยขีดจำกัดของไฟร์วอลล์คือไม่สามารถป้องกันระดับชั้นแอพพลิเคชันได้ ดังนั้นจึงมีพร็อกซี่ขึ้นมาซึ่งเป็นไฟร์วอลล์อีกแบบหนึ่ง เพื่อใช้ป้องกันในระดับชั้นแอพพลิเคชัน แต่แม้จะมีทั้งไฟร์วอลล์และพร็อกซี่ ก็ยังคงป้องกันผู้บุกรุกไม่ได้เต็มที่อยู่ดี ต่อมาจึงมีการพัฒนาระบบตรวจจับผู้บุกรุกขึ้น เพื่อรับผิดชอบการตรวจจับความผิดปกติต่างๆในระบบคอมพิวเตอร์ และเครือข่าย ซึ่งบนระบบปฏิบัติการลีนุกซ์ได้มีทั้งไฟร์วอลล์ พร็อกซี่ และระบบตรวจจับผู้บุกรุกมาให้แล้ว ได้แก่โปรแกรม Iptables , Proxy และ Snort ตามลำดับ แต่การเซ็ตคำสั่งต่างๆใน Iptables และ Proxy ค่อนข้างจะยุ่งยาก และคำสั่งในการเขียนกฎใน Snort ก็ค่อนข้างจะยุ่งยากเช่นเดียวกัน

แนวทางในการแก้ไขปัญหาดังกล่าวคือการพัฒนาส่วนติดต่อกับผู้ใช้ หรือกราฟฟิสกยูสเซอร์อินเตอร์เฟส(Graphic User Interface:GUI) ขึ้นมาเพื่อให้ความสะดวกในการเซ็ตคำสั่งต่างๆ

## 1.2 วัตถุประสงค์ของงานวิจัย

วัตถุประสงค์ของงานวิจัย คือการออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface:GUI) เพื่อให้ผู้ไม่ทราบคำสั่งต่างๆก็สามารถใช้งานได้ และเพื่อให้ความสะดวกแก่ผู้ใช้มากยิ่งขึ้นในการใช้งาน Iptables , Proxy และการออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส เพื่อใช้ในการสร้างกฎรูปแบบการโจมตีของผู้บุกรุกใน Snort เพื่อให้ผู้ใช้สามารถสร้างกฎใน Snort ได้สะดวกและง่ายขึ้น

## 1.3 ขอบเขตของงานวิจัย

ขอบเขตของงานวิจัยคือการศึกษาหลักการและการทำงานของไฟร์วอลล์โดยใช้ Iptables , Proxy โดยใช้ Squid และระบบตรวจจับผู้บุกรุก(Intrusion Detection System)โดยใช้ Snort แล้วออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface:GUI)

**1.4 เป้าหมายของโครงงาน**

1. ออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface:GUI) สำหรับ Iptables
2. ออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface:GUI) สำหรับ Squid
3. ออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface:GUI) สำหรับการสร้างกฎใน Snort

**1.5 วิธีการดำเนินงาน**

1. ศึกษาหลักการและการทำงานของไฟร์วอลล์โดยใช้ Iptables ,Proxy โดยใช้ Squid และระบบตรวจจับผู้บุกรุก(Intrusion Detection System)โดยใช้ Snort
2. Set up ระบบบนลีนุกซ์โดยใช้โปรแกรม Iptables , Squid และ Snort
3. ออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface:GUI) สำหรับ Iptables
4. ออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface:GUI) สำหรับ Squid
5. ออกแบบส่วนติดต่อกับผู้ใช้ หรือกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface:GUI) สำหรับการสร้างกฎใน Snort

# บทที่ 2
# ทฤษฎีและหลักการ

## 2.1 ลีนุกซ์คืออะไร

ลีนุกซ์ระบบปฏิบัติการแบบ 32 บิต ที่เป็นยูนิกซ์โคลน สำหรับเครื่องพีซี และแจกจ่ายให้ใช้ฟรี สนับสนุนการใช้งานแบบหลากงาน หลายผู้ใช้ (MultiUser-MultiTasking) มีระบบ X วินโดวส์ ซึ่งเป็นระบบการติดต่อผู้ใช้แบบกราฟฟิก(Graphic) ที่ไม่ขึ้นกับระบบปฏิบัติการ (Operating System) หรือฮาร์ดแวร์ใดๆ (มักใช้กันมากในระบบยูนิกซ์) และมาตรฐานการสื่อสาร TCP/IP ที่ใช้เป็นมาตรฐานการสื่อสารในอินเทอร์เนตมาให้ในตัว สามารถนำโค้ดต้นฉบับมาทำการแก้ไข ปรับปรุงตามความต้องการของผู้พัฒนา และยังสามารถทำสำเนาแจกจ่ายได้ ไม่ว่าจะเพื่อการศึกษาหรือในเชิงพาณิชย์ก็ตาม ลีนุกซ์ได้พัฒนาขึ้นภายใต้ลิขสิทธิ์แบบ GPL (GNU General Public License) ซึ่งเป็นลิขสิทธิ์ที่ยอมให้มีการเปลี่ยนแปลงต้นฉบับหรือแจกจ่ายได้โดยไม่จำกัดสิทธิ์ แต่ซอฟต์แวร์นั้นจะต้องยังคงเป็นลิขสิทธิ์แบบ GPL อยู่

### 2.1.1 ประวัติของลีนุกซ์

ลีนุกซ์ถือกำเนิดขึ้นในฟินแลนด์ ปี คศ. 1980 โดยลีนุส โทรวัลด์ส (Linus Trovalds) นักศึกษาภาควิชาวิทยาการคอมพิวเตอร์ (Computer Science) ในมหาวิทยาลัยเฮลซิงกิ ประเทศฟินแลนด์ ลีนุส เห็นว่าระบบมินิกซ์ (Minix) ที่เป็นระบบยูนิกซ์บนพีซีในขณะนั้น ซึ่งทำการพัฒนาโดย ศ.แอนดรูว์ ทาเนนบาวม์ (Andrew S. Tanenbaum) ยังมีความสามารถไม่เพียงพอแก่ความต้องการ จึงได้เริ่มต้นทำการพัฒนาระบบยูนิกซ์ของตนเองขึ้นมา โดยจุดประสงค์อีกประการ คือต้องการทำความเข้าใจในวิชาระบบปฏิบัติการคอมพิวเตอร์ด้วยเมื่อเขาเริ่มพัฒนาลีนุกซ์ไปช่วงหนึ่งแล้ว เขาก็ได้ทำการชักชวนให้นักพัฒนาโปรแกรมอื่นๆมาช่วยทำการพัฒนาลีนุกซ์ ซึ่งความร่วมมือส่วนใหญ่ก็จะเป็นความร่วมมือผ่านทางอินเทอร์เนต

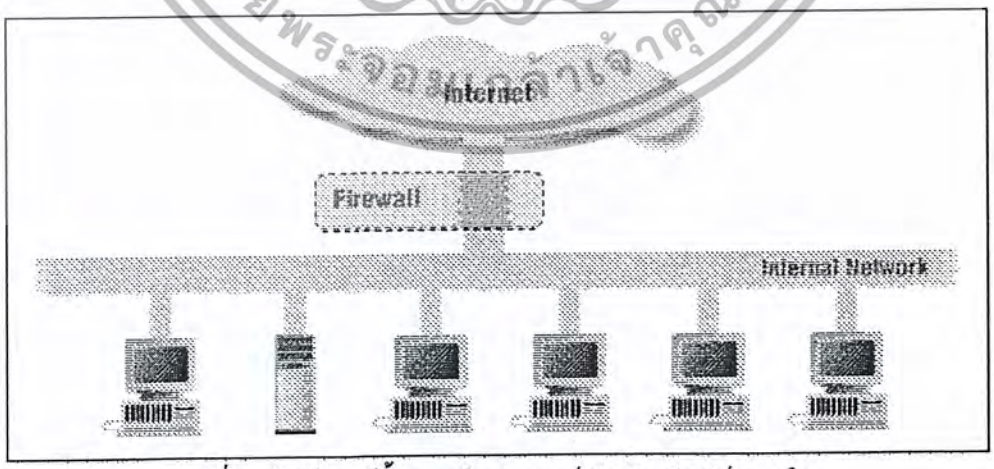
### 2.1.2 คุณสมบัติพื้นฐานของลีนุกซ์

- Multitasking ลีนุกซ์สามารถทำงานได้หลายโปรแกรมพร้อมๆกัน โดยที่โปรแกรมต่างๆทำงานเป็นอิสระจากกัน
- Multiuser ลีนุกซ์ยอมให้ผู้ใช้สามารถเข้ามาทำงานพร้อมๆกันได้ โดยผู้ใช้แต่ละคนทำงานได้เป็นอิสระจากกัน
- ทำงานได้ทั้งแบบ Text Mode และ Window Mode
- ทำงานได้ทั้งแบบหน้าเครื่อง (Console) หรือจากเครื่องอื่น (Remote)

**2.2 ไฟร์วอลล์ (Firewall)**

ไฟร์วอลล์คือเครื่องมือที่ใช้ป้องกันเน็ตเวิร์กจากการสื่อสารทั่วไปที่ไม่ได้รับอนุญาต ปัญหาพื้นฐานที่สุดในเรื่องความปลอดภัยบนเน็ตเวิร์กก็คือ การเข้าถึงระบบหรือข้อมูลภายในผ่านทางเน็ตเวิร์ก หรือที่เรียกว่า ลอจิคัลแอคเซส(Logical Access) ซึ่งมักเกิดขึ้นได้ง่ายกว่าการเข้าถึงทาง physical คือเข้ามาที่ตัวเครื่องจริงๆ การที่เรานำโฮสต์ใดๆมาต่อเข้ากับเน็ตเวิร์ก หมายถึงโฮสต์ของเราสามารถถูกแอคเซสได้จากทุกๆที่ตราบเท่าที่เน็ตเวิร์กนั้นจะครอบคลุมไปถึง อย่างไรก็ตามลอจิคัลแอคเซสจะเกิดขึ้นได้นั้นก็ต่อเมื่อโฮสต์จะต้องสามารถสร้าง "การเชื่อมต่อ" หรือ ลอจิคัลคอนเนคชั่น (Logical Connection) กับโฮสต์เป้าหมายปลายทางได้ ซึ่งความสามารถในการสร้างลอจิคัลนั้นจะขึ้นอยู่กับโปรโตคอลที่ใช้งานเป็นสำคัญ โปรโตคอลที่สำคัญที่สุดที่ต้องดูแลอย่างระมัดระวังก็คือ TCP/IP ซึ่งใช้งานอยู่บนอินเตอร์เน็ตในปัจจุบัน เพราะสามารถสร้างลอจิคัลคอนเนคชั่นได้โดยไม่มีขีดจำกัดในเรื่องระยะทาง อย่างไรก็ตามการแอคเซสกันได้ระหว่างโฮสต์ก็ใช่ว่าจะมีแต่เพียงประโยชน์เพียงด้านเดียว การแชร์ข้อมูลกันช่วยให้ทำงานร่วมกันได้อย่างสะดวกรวดเร็วถูกต้อง แต่ในทางกลับกันก็ทำให้ข้อมูลสามารถรั่วไหลไปตามเน็ตเวิร์กได้อย่างรวดเร็วเช่นกัน การมีเน็ตเวิร์กที่ครอบคลุมเฉพาะในองค์กรของตนเองนั้น ถึงแม้จะมีความเสี่ยงจากความไม่ปลอดภัยทางเน็ตเวิร์กอยู่บ้าง ก็จะจำกัดขอบเขตเฉพาะจากคอมพิวเตอร์ที่ต่ออยู่ภายใน แต่เมื่อองค์การของท่านต้องการติดต่อสื่อสารกับโลกภายนอก ความจำเป็นที่จะต้องการขยายเครือข่ายเน็ตเวิร์กให้ครอบคลุมไปยังภายนอกย่อมมีมากขึ้น ความเสี่ยงในการที่เน็ตเวิร์กอาจจะถูกแอคเซสได้จากบุคคลภายนอกก็มีมากขึ้น

อินเตอร์เน็ตเป็นเครือข่ายสื่อสารข้อมูลคอมพิวเตอร์สากลที่เชื่อมต่อเน็ตเวิร์กต่างๆทั่วโลกเข้าด้วยกัน อาศัยโปรโตคอล TCP/IP เป็นกลไลสำคัญในการสื่อสารข้อมูลข้ามเน็ตเวิร์กต่างๆไปจนถึงที่หมาย ด้วยเหตุที่โปรโตคอล TCP/IP เองขาดกลไลในการรักษาความปลอดภัยที่เพียงพอ ปัญหาที่ตามมาจึงมีอยู่มากมายและเริ่มส่งผลกระทบต่อความปลอดภัยของผู้ใช้มากขึ้นเป็นลำดับ และทำให้เกิดความจำเป็นที่จะต้องหาอุปกรณ์รักษาความปลอดภัยอื่นๆเข้ามาช่วยเพิ่มเติม เช่น ไฟร์วอลล์เป็นต้น




*รูปที่ 2-1 ไฟร์วอลล์กั้นระหว่างอินเตอร์เน็ตกับเน็ตเวิร์กภายใน*

**2.2.1 คุณสมบัติทั่วไปของไฟร์วอลล์**

ไฟร์วอลล์เป็นเครื่องมือรักษาความปลอดภัยที่ทำงานในเชิงป้องกัน (Protect) ซึ่งจะทำหน้าที่ควบคุมการเข้าถึงเน็ตเวิร์ก (Access Control) โดยอาศัยกฎเป็นพื้นฐาน (Rule base) สำหรับคุณสมบัติแต่ละอย่างของไฟร์วอลล์นั้นมีรายละเอียดดังนี้

1. **Protect :** ไฟร์วอลล์เป็นเครื่องมือที่ใช้ทำงานในเชิงป้องกัน โดยแพ็คเก็ตที่สามารถผ่านเข้า-ออกเน็ตเวิร์กได้นั้น จะต้องเป็นแพ็คเก็ตที่ไฟร์วอลล์เห็นว่ามีความปลอดภัย แพ็กเก็ตใดที่ไฟร์วอลล์เห็นว่าไม่ปลอดภัย หรืออาจจะนำมาซึ่งความไม่ปลอดภัยก็จะถูกดร็อป(drop) คือทิ้งไปเสียเฉยๆไม่ส่งต่อ โดยการที่ไฟร์วอล์จะตัดสินใจว่าแร็คเก็ตใดปลอดภัยและแพ็ตเก็ตใดไม่ปลอดภัยนั้นจะอยู่บนพื้นฐานกฎที่ผู้ดูแลไฟร์วอลล์(Firewall Administrator) เป็นผู้กำหนดไว้ล่วงหน้า ซึ่งเงื่อนไขของกฎเหล่านี้เองทำให้ไฟร์วอลลสามารถป้องกันแพ็คเก็ตที่อาจจะส่งผลร้ายไม่ให้ผ่านเข้าไปถึงเน็ตเวิร์กได้
2. **Access Control :** "แอคเซส" หมายถึงการที่โฮสต์ใดโฮสต์หนึ่งสามารถสื่อสารข้อมูลที่ต้องการไปยังโฮสต์ปลายทางได้สำเร็จ การแอคเซสในแต่ละระดับจะมีวิธีการแตกต่างกันออกไป ทำให้การควบคุมการแอคเซสสำหรับแต่ละระดับแตกต่างกันตามไปด้วย ไฟร์วอลล์จึงมีการทำงานหลายลักษณะตามวิธีที่ไฟร์วอลล์ใช้ควบคุมการแอคเซส
3. **Rule Base :** ไฟร์วอลล์จะควบคุมการแอคเซสโดยอาศัยการเปรียบเทียบคุณสมบัติของแร็คเก็ตที่จะผ่านไฟร์วอลล์กับกฎของการแอคเซสที่ได้กำหนดไว้ หากพบว่าไม่มีกฎที่ห้ามไว้ก็จะอนุฐาตให้แพ็คเก็ตนั้นผ่านไปได้ หากมีกฎที่ห้ามไว้แร็คเก็ตนั้นก็จะถูกสกัดกั้นไว้ด้วยวิธีใดวิธีหนึ่ง

**2.2.2 ประเภทของ Firewall**

**2.2.2.1 Packet Filtering Firewall**

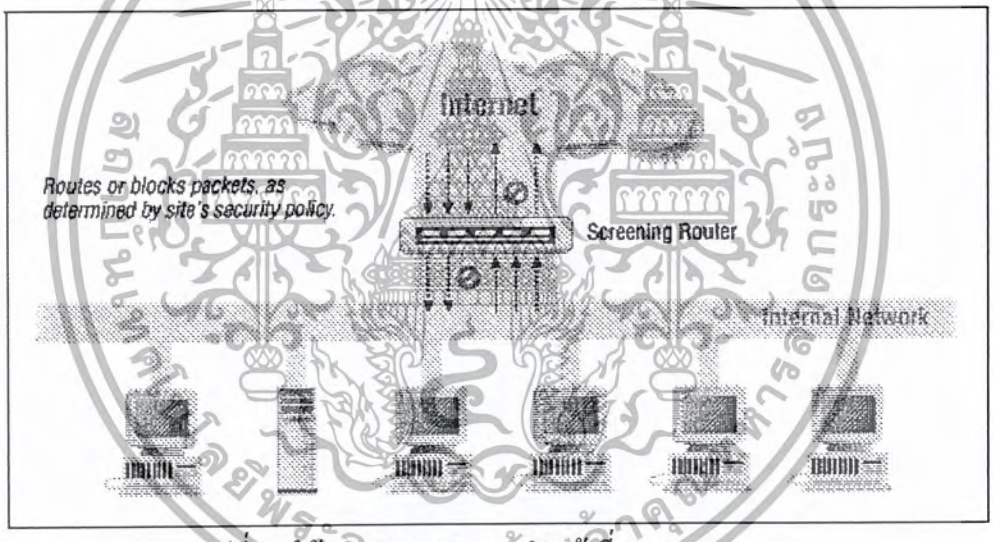
โดยทั่วไปจะเรียกว่า *Screening Router* เป็น ไฟร์วอลล์พื้นฐานที่มีความสามารถในการควบคุม traffic โดยอาศัยการตรวจสอบข้อมูลที่ปรากฏอยู่ในแพ็คเก็ต(Packet) ไฟร์วอลล์ประเภทนี้อาจจะเป็นความสามารถที่เพิ่มมาในฝั่งเราเตอร์(Router) โดยการทำงานจะอาศัยโครงสร้างพื้นฐานที่เราเตอร์มีอยู่ให้ทำหน้าที่มากกว่าการเราต์ เป็นเราเตอร์ที่ทำการหาเส้นทางและส่งต่อ (route) อย่างมีเงื่อนไข โดยจะพิจารณาจากข้อมูลส่วนที่อยู่ในเฮดเดอร์ (header) ของแพ็กเก็ตที่ผ่านเข้ามา เทียบกับกฎ (rules) ที่กำหนดไว้และตัดสินว่าควรจะทิ้ง (drop) แพ็กเก็ตนั้นไปหรือว่าจะยอม (accept) ให้แพ็กเก็ตนั้นผ่านไปได้ ในการพิจารณาเฮดเดอร์ Packet Filter จะตรวจสอบในระดับของอินเตอร์เน็ตเลเยอร์ (Internet Layer) และทรานสปอร์ตเลเยอร์ (Transport Layer) ในอินเตอร์เน็ตโมเดล ซึ่งในอินเตอร์เน็ตเลเยอร์จะมีแอตทริบิวต์ที่สำคัญต่อ Packet Filtering ดังนี้

- ไอพีต้นทาง
- ไอพีปลายทาง
- ชนิดของโปรโตคอล (TCP UDP และ ICMP)

และในระดับของทรานสปอร์ตเลเยอร์ มีแอตทริบิวต์ที่สำคัญคือ

- พอร์ตต้นทาง
- พอร์ตปลายทาง
- แฟล็ก (Flag ซึ่งจะมีเฉพาะในเฮดเดอร์ของแพ็กเก็ต TCP)
- ชนิดของ ICMP message (ในแพ็กเก็ต ICMP)

ซึ่งพอร์ตของทรานสปอร์ตเลเยอร์ คือทั้ง TCP และ UDP นั้นจะเป็นสิ่งที่บอกถึงแอพพลิเคชันที่แพ็กเก็ตนั้นต้องการติดต่อด้วยเช่น พอร์ต 80 หมายถึง HTTP, พอร์ต 21 หมายถึง FTP เป็นต้น ดังนั้นเมื่อ Packet Filter พิจารณาเฮดเดอร์ จึงทำให้สามารถควบคุมแพ็กเก็ตที่มาจากที่ต่างๆ และมีลักษณะต่างๆ (ดูได้จากแฟล็กของแพ็กเก็ต หรือ ชนิดของ ICMP ในแพ็กเก็ต ICMP) ได้ เช่น ห้ามแพ็กเก็ตทุกชนิดจาก crack.cracker.net เข้ามายังเน็ตเวิร์ก 203.154.207.0/24 , ห้ามแพ็กเก็ตที่มีไอพีต้นทางอยู่ในเน็ตเวิร์ก 203.154.207.0/24 ผ่านเราเตอร์เข้ามา




*รูปที่ 2-2 ใช้ Screening Router ทำหน้าที่ Packet Filtering*

Packet Filtering สามารถอิมพลีเมนต์ได้จาก 2 แพล็ตฟอร์ม คือ

- เราเตอร์ที่มีความสามารถในการทำ Packet Filtering (ซึ่งมีในเราเตอร์ส่วนใหญ่อยู่แล้ว)
- คอมพิวเตอร์ที่ทำหน้าที่เป็นเราเตอร์

**ข้อดีของ Screening Router**

1. ราคาถูกเพราะเป็นคุณสมบัติที่มักมีในเราเตอร์อยู่แล้ว อาศัยเพียงการกำหนดแอคเซสรูลที่เหมาะสมเท่านั้น หากยังไม่มีไฟร์วอลล์อยู่เลย ก็สามารถใช้เพื่อช่วยป้องกันเน็ตเวิร์กภายในได้ดีพอสมควรในระดับหนึ่ง
2. หากเน็ตเวิร์กภายในไม่ใหญ่มาก และมีการใช้งานอินเตอร์เน็ตอย่างจำกัด ก็สามารถใช้ทดแทนไฟร์วอลล์ได้ทันที
3. การป้องกันบางประเภทไม่สามารถป้องกันได้โดยไฟร์วอลล์ จะต้องทำโดยการกำหนดที่เราเตอร์เท่านั้น

**ข้อเสียของ Screening Router**

การกำหนดแอคเซสรูลทำได้ยาก ไม่มีระบบ user interface เพื่อช่วยในการทำงาน

1. คำสั่งในการทำงานจะผูกติดกับยี่ห้อของเราเตอร์ ไม่มีมาตรฐานของคำสั่ง หากเปลี่ยนยี่ห้อเราเตอร์ก็จะต้องศึกษารูปแบบของคำสั่งใหม่
2. ไม่สามารถกำหนดกฎที่ซับซ้อนได้ เพื่อจากขีดจำกัดของเราเตอร์ที่ทำงานโดยพิจารณาครั้งละแพ็คเก็ตเท่านั้น
3. เราเตอร์มีกำลังในการประมวลผลจำกัด

**2.2.2.2 Circuit-Level Firewall หรือ Stateful Firewall**

การสื่อสารข้อมูลโดยทั่วไปจะเป็นการสื่อสารแบบต่อเนื่อง โต้ตอบกันไปมาระหว่างผู้รับและผู้ส่งอยู่เสมอ โปรโตคอลที่อยู่ในเลเยอร์ที่สูงกว่าอินเตอร์เน็ตเลเยอร์ ไม่ว่าจะเป็นทรานสปอร์ตเลเยอร์อย่างเช่น TCP UDP หรือเลยไปจนถึงแอพพลิเคชั่นเลเยอร์ เช่น FTP,HTTP,SMTP ล้วนแล้วแต่จะต้องมีสถานะของการสื่อสาร(State) เสมอ สถานะนี้จะทำให้ทั้งสองฝั่งสามารถสื่อสารกันได้อย่างต่อเนื่องคือทราบว่าตอนนี้กำลังอยู่ ณ จุดใดและจะต้องส่งหรือรับข้อมูลใดเป็นลำดับต่อไป โดยทั่วไปหากพูดถึงไฟร์วอลล์จะหมายถึงไฟร์วอลล์ประเภทนี้นั่นเอง

Circuit-Level Firewall เป็นไฟร์วอลล์ที่ทำงานโดยที่สามารถเข้าใจสถานะการสื่อสารทั้งกระบวนการ เพราะว่าการสื่อสารข้อมูลจะสมบูรณ์ได้นั้นจะต้องมีทั้งการส่งและการรับอย่างสอดคล้องสัมพันธ์กันนั่นเอง หมายถึงหากไฟร์วอลล์จะสามารถควบคุมการสื่อสารได้จริงก็จะต้องสามารถเข้าใจกระบวนการของการสื่อสารตั้งแต่ต้นจนจบ โดยทั่วไปเราจะเรียกไฟร์วอลล์แบบนี้ว่า *"Stateful Inspection Firewall"* เป็นไฟร์วอลล์ที่ทำการควบคุม traffic โดยใช้หลักการของแพ็คเก็ตฟิลเตอริ่งและการกำหนดแอคเซสรูลเช่นเดียวกับสกรีนนิ่งเราเตอร์ แต่สเตทฟูลไฟร์วอลล์จะมีความสามารถในการวิเคราะห์และรับรู้ความต่อเนื่องของแพ็คเก็ตในโปรโตคอลในระดับสูงขึ้นไปมากกว่า ไม่ว่าจะเป็น TCP,FTP,HTTP

Stateful firewall เป็น stateful เพราะมันสามารถตรวจสอบ packet data โดยมีการเปรียบเทียบกับ history ของแต่ละ connection ได้ จะบันทึกข้อมูลเกี่ยวกับ connection ที่เกิดขึ้นลงใน state table ก่อนที่จะส่ง packet นั้นไปยัง IP stack ตัว table นี้จะมีส่วนสำหรับบันทึกข้อมูลสำหรับแต่ละ connection ที่ถูกต้อง

โดยปกติจะเก็บข้อมูล source and destination address, protocol, port, flag เมื่อ packet-filtering firewall ได้รับ packet มันจะตรวจสอบข้อมูลกับ state table ว่าเป็นส่วนของ connection ที่สร้างไว้แล้วหรือไม่ โดยจะพิจารณาจากข้อมูล source adderss, destination address, source port, destnation port จะต้องสอดคล้องกับ state table ซึ่งถ้าเป็นส่วนหนึ่งของ connection จริงก็ไม่มีความจำเป็นใดๆ ที่ต้องตรวจสอบซ้ำอีก ถ้า packet ที่ส่งมาไม่ตรงกับ connection ที่สร้างไว้แล้ว และไม่ใช่ SYN packet ตัว packet นั้นๆ ก็จะถูก drop ทิ้งไป และแม้แต่ packet ที่ผสม flag แปลกๆ เช่น SYN/FIN (เป็นกระบวนการหนึ่งในการทำ port scanning) ก็จะถูก drop ทิ้งไปเช่นเดียวกัน ทั้งนี้ไฟร์วอลล์ส่วนใหญ่สามารถบันทึกล็อกได้ด้วย ซึ่งขึ้นอยู่กับการตั้งค่าของผู้ดูแลระบบเองว่าต้องการเก็บข้อมูลใด ตัวอย่างผลิตภัณฑ์ทางการค้าที่ใช้ Stateful Inspection Technology ในส่วนที่เป็น open source แจกฟรีได้แก่ Netfilter ใน Linux (iptables ในลีนุกซ์เคอร์เนลตั้งแต่ 2.3 ขึ้นไป)

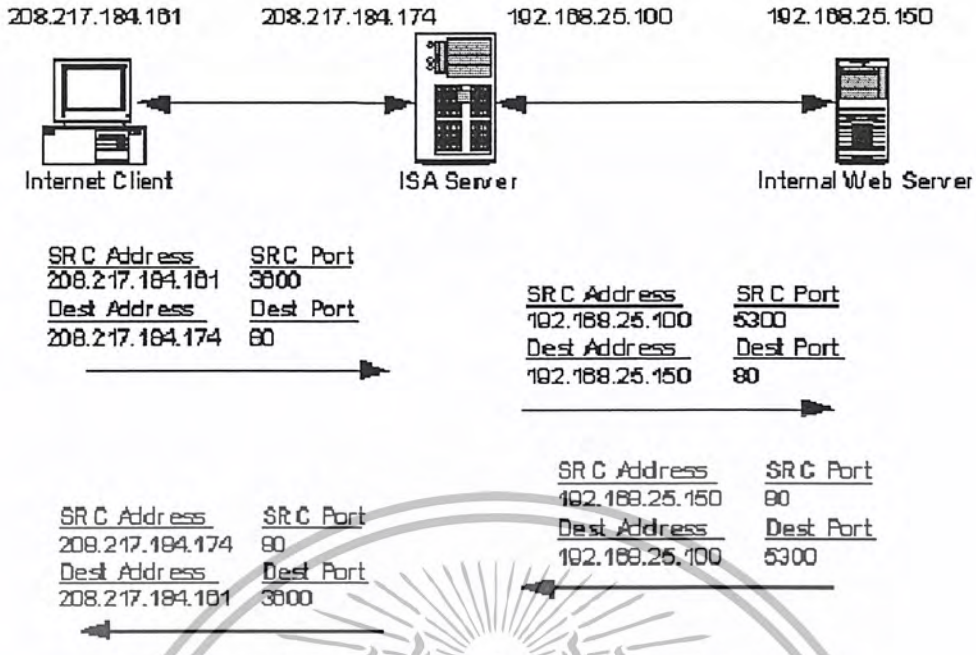
**ความแตกต่างของการพิจารณาข้อมูลแบบแพ็คเก็ตกับสเตทฟูล(Stateful)**

อันที่จริงสองเรื่องนี้มิได้ขัดแย้งกันแต่ประการใด แพ็คเก็ตนั้นเป็นการสื่อสารที่เป็นส่วนย่อยของการสื่อสารทั้งหมด ผลของการสื่อสารข้อมูลก็คือผลรวมของการสื่อสารข้อมูลหลายๆแพ็คเก็ตนั่นเอง แต่อย่างไรก็ตามการฟิลเตอร์หรือกรอง โดยพิจารณาทีละแพ็คเก็ตของทุกแพ็คเก็ตที่ผ่านเข้าออกนั้นอาจจะมีผลลัพธ์แตกต่างจากการฟิลเตอร์ของในแบบที่มองสถานะและภาพรวมหรือที่เรียกว่าสเตทฟูล(Stateful)

**2.2.2.3 Application Level Firewall(Proxy)**

พร็อกซี่เป็นเครื่องมือในการควบคุม traffic ชนิดหนึ่งซึ่งทำงานที่ระดับแอพพลิเคชั่น(Application layer) ในลักษณะที่เป็นตัวกลางในการสื่อสารระหว่างไคลเอนต์กับเซิร์ฟเวอร์ โดยทำหน้าที่ป้องกันไม่ให้มีการสื่อสารโดยตรงระหว่างไคลเอนต์กับเซิร์ฟเวอร์ แต่ยังคงให้ไคลเอนต์สามารถใช้งานแอพพลิเคชั่นบนเซิร์ฟเวอร์ได้ตามปกติ และผู้ใช้ซึ่งใช้งานแอพพลิเคชั่นชั้นๆจะไม่ได้รับผลกระทบแต่อย่างใด เมื่อไคลเอนต์ต้องการใช้เซอร์วิสภายนอก ไคลเอนต์จะทำการติดต่อไปยัง Proxy ก่อน ไคลเอนต์จะเจรจา (negotiate) กับ Proxy เพื่อให้ Proxy ติดต่อไปยังเครื่องปลายทางให้ เมื่อ Proxy ติดต่อไปยังเครื่องปลายทางให้แล้วจะมีการเชื่อมต่อ (connection) 2 การเชื่อมต่อ คือ ไคลเอนต์กับ Proxy และ Proxy กับเครื่องปลายทาง โดยที่ Proxy จะทำหน้าที่รับข้อมูลและส่งต่อข้อมูลให้ใน 2 ทิศทาง ทั้งนี้ Proxy จะทำหน้าที่ในการตัดสินใจว่าจะให้มีการเชื่อมต่อกันหรือไม่ จะส่งต่อแพ็กเก็ตให้หรือไม่

Application proxy จะทำการตรวจสอบข้อมูลในชั้น network layer และ transport layer และยังสามารถตรวจสอบความถูกต้องในชั้น application layer ได้อีกด้วย ซึ่งทำให้ Application proxy สามารถกลั่นกรอง commands, protocol, packet length, authorization, content, invalid header หรือสามารถส่งผ่าน packet ไปได้เลย Application proxy เป็น stateful firewall แต่สิ่งที่สร้างความแตกต่างคือ proxy server จะสร้าง IP packet ใหม่ เพื่อส่งต่อไปยังเป้าหมาย และ proxy จะสร้าง packet ใหม่เพื่อส่งต่อก็ต่อเมื่อ packet นั้นผ่านการตรวจสอบแล้ว




*รูปที่ 2-3 แสดงการทำงานของ Application Proxy*

**2.2.3 การจัดโซน(Zoning)**

การจัดโซนคือการจัดแบ่งโฮสต์ที่จะต้องอาศัยไฟร์วอลล์เป็นทางผ่านของ traffic ออกเป็นส่วนๆ ตามลักษณะของการใช้งานปกติของโฮสต์ในกลุ่มนั้นๆ โฮสต์ที่อยู่ในโซนเดียวกันสามารถติดต่อกันได้โดยตรงโดยไม่ต้องผ่านไฟร์วอลล์ แต่ถ้าโฮสต์อยู่ต่างโซนกันการสื่อสารใดๆที่เกิดขึ้นจะต้องผ่านไฟร์วอลล์เสมอ การสื่อสารระหว่างโซนจะต้องไม่สามารถติดต่อกันได้โดยตรงไม่ว่าด้วยช่องทางใด ประเภทของโซนนั้นจะมีการกำหนดไว้เป็นประเภทใหญ่ๆ 3 โซน

1. Internal network
2. External network
3. Demilitarized Zone

ลักษณะของโฮสต์นั้นแบ่งได้ออกเป็นประเภทใหญ่ๆคือ

1. **Trusted Host**

ทรัสต์โฮสต์ หมายถึงโฮสต์ที่เราสามารถไว้วางใจได้ สามารถทราบได้ว่าผู้ใช้เป็นใคร มีการควบคุมและป้องการการใช้งานของโฮสต์เป็นอย่างดี โดยส่วนใหญ่หมายถึงเครื่องคอมพิวเตอร์ที่อยู่ภายในองค์กร

2. **Untrusted Host**

อันทรัสต์โอสต์ หมายถึงโฮสต์ที่เราไม่สามารถไว้วางใจได้ ไม่ทราบว่าผู้ใช้เป็นใคร มีวัตถุประสงค์ใด อาจจะเป็นโฮสต์ใดๆก็ได้ที่อยู่บนอินเตอร์เน็ต

**2.2.3.1 Internal network (เน็ตเวิร์กภายใน)**

เป็นเน็ตเวิร์กที่มีเฉพาะทรัสต์โฮสต์ เนื่องจากภายในเน็ตเวิร์กนี้จะเป็นการใช้งานภายในองค์กร และโฮสต์ทั้งหมดล้วนเป็นโฮสต์ที่อยู่ภายในทั้งสิ้น ซึ่งโฮสต์เหล่านี้มีการสื่อสารข้อมูลกันอย่างใกล้ชิด เพราะจำเป็นต้องใช้ข้อมูลระหว่างกันเพื่อการทำงาน มีการควบคุมเพียงเล็กน้อย เนื่องจากต้องการให้ผู้ใช้ของแต่ละโฮสต์ใช้งานทรัพยากรร่วมกันมากที่สุดอยู่แล้ว เป็นการใช้งานปกติของทุกองค์กรที่มีเน็ตเวิร์กเชื่อมต่อถึงกัน

**2.2.3.2 External network(เน็ตเวิร์กภายนอก)**

หมายถึงโฮสต์อื่นๆที่ไม่ได้อยู่ภายใน Internal network เป็นโฮสต์ใดๆก็ตามที่อยู่บนอินเตอร์เน็ตก็ถือว่าเป็น Untrusted Host ทั้งหมด เนื่องจากไม่สามารถไว้วางใจและควบคุมผู้ใช้ได้นั่นไม่ได้หมายความว่าโฮสต์ที่อยู่ใน External network นั้นจะต้องเป็นโฮสต์ที่ผู้ใช้ประสงค์ร้ายแต่อย่างใด แต่การที่ไม่สามารถวางใจได้นั้น หมายถึงไม่สามารถทราบข้อมูลใดเกี่ยวกับผู้ใช้เพียงพอที่จะพิจารณาว่าควรไว้วางใจหรือไม่ กี่ประการ อีกประการหนึ่งก็คือองค์กรของเราไม่มีความสัมพันธ์เกี่ยวข้องใดๆกับโฮสต์เหล่าอันเป็นเหตุผลที่จะอนุญาตให้มีการสื่อสารกันโดยไม่มีการควบคุม

**2.2.3.3 Demilitarized Zone (DMZ)**

เป็นคำจำกัดความของโซนประเภทหนึ่งที่ไม่ใช่ทั้ง Internal และ External เนื่องจากด้วยความจำเป็นที่โฮสต์จะต้องสื่อสารโดยตรงกับทั้ง Internal Network และ External Network หรืออีกนัยหนึ่งคือโฮสต์ที่จะต้องติดต่อโดยตรงกับทั้งทรัสโฮสต์ (Trusted Host) และอันทรัสโฮสต์ (Untrusted Host) ดังนั้นการที่จะกำหนดให้โฮสต์ประเภทนี้อยู่ใน Internal Network ก็เกรงว่าจะทำให้โฮสต์อื่นๆพลอยเดือดร้อนไปด้วย เพราะโฮสต์อื่นๆที่อยู่ใน Internal Network นั้น แต่เดิมจะถูกกำหนดให้ไม่สามารถติดต่อภายนอกได้โดยตรง ในทำนองเดียวกันจะผลักไสให้โฮสต์ประเภทนี้ไปอยู่ใน External Network ก็ดูว่าจะขัดแย้งอยู่เช่นกัน เพราะโฮสต์เหล่านี้ยังคงต้องการการสื่อสารโดยตรงกับโฮสต์ที่อยู่ภายในอยู่ดี

**2.3 องค์ประกอบของแพ็คเก็ต(Packet)**

แพ็คเก็ตเป็นหน่วยพื้นฐานของการรับส่งข้อมูลของลิงค์เลเยอร์(Layer ที่2 ในโครงสร้างโปรโตคอล TCP/IP) การรับส่งข้อมูลแต่ละครั้งของลิงค์เลเยอร์จะส่งข้อมูลออกไปชุดหนึ่ง โดยที่ความยาวของชุดของข้อมูลนี้จะมีเท่าใดนั้นจะเป็นไปตามคุณสมบัติของลิงค์เลเยอร์นั้นๆ ข้อมูลแต่ละชุดนั้นเรียกว่าแพ็คเก็ต สำหรับโปรโตคอล TCP/IP นั้นจะใช้ IP เป็นโปรโตคอลหลักในการขนส่งข้อมูลระหว่างโฮสต์โดย IP ซึ่งอยู่ในอินเตอร์เน็ตเลเยอร์จะส่งข้อมูลลงไปยังลิงค์เลเยอร์ตามลำดับ โดยที่หากขนาดของดาต้าแกรม(IP Datagram) ที่จะส่งนั้นสามารถส่งไปโดยใช้แพ็คเก็ตเดียว IP ก็จะส่งดาต้าแกรมนั้นไปในทันทีและแพ็คเก็ตนั้นก็คือข้อมูลของ IP 1 ดาต้าแกรม แต่หากขนาดของ IP Datagram ใหญ่กว่าขนาดของลิงค์เลเยอร์แล้ว IP ก็จะต้องทำการแบ่งส่วนหรือแฟรกเมนเตชั่น คือกระจายดาต้าแกรมออกเป็นส่วนย่อยเสียก่อน

แล้วจึงค่อยส่งลงไปที่ลิงค์เลเยอร์ ซึ่งในกรณีนี้ ข้อมูล 1 แพ็คเก็ตจะเป็นเพียงส่วนย่อยหรือแฟรกเมนต์ (Fragment) หนึ่งของดาต้าแกรมเท่านั้น ดังนั้นข้อมูล 1 แพ็คเก็ตจึงไม่จำเป็นต้องเป็นข้อมูล 1 ดาต้าแกรมเสมอไป แต่อย่างไรก็ตามแพ็คเก็ตทุกแพ็คเก็ตที่ส่งมาจาก IP จะมีข้อมูลอย่างน้อยที่สุดคือ IP Address ต้นทางและปลายทางเสมอ ซึ่งจำเป็นสำหรับให้แพ็คเก็ตนั้นวิ่งต่อไปจนถึงที่หมายปลายทางได้

หากข้อมูล 1 แพ็คเก็ตนั้นบรรจุครบถ้วนทั้ง IP Datagram แล้วก็จะทำให้สามารถทราบถึงข้อมูลโปรโตคอลเลเยอร์ที่สูงขึ้นไปด้วยว่าเป็น ICMP, TCP UDP หรือโปรโตคอลอื่นใดที่อาศัยอยู่ใน IP Datagram นั้นแต่หากแพ็คเก็ตนั้นไม่สามารถบรรจุข้อมูลได้ครอบคลุมทั้งดาต้าแกรมแล้ว ก็จะทำให้เพียงแต่ทราบว่าแพ็คเก็ตนั้นเป็น IP แพ็คเก็ตท่านั้น ดังนั้นหากมีการกล่าวถึงแพ็คเก็ตของ TCP หรือของโปรโตคอลในเลเยอร์สูงขึ้นไปเมื่อใด นั่นหมายถึงการกล่าวถึงแพ็คเก็ตที่มีดาต้าแกรมสมบูรณ์จนทำให้สามารถทราบได้ถึงข้อมูลในเลเยอร์ TCP หรือโปรโตคอลเลเยอร์สูงขึ้นไปนั่นเอง

### 2.3.1 ข้อมูลที่สำคัญของแพ็คเก็ต

ภายในของแต่ละแพ็คเก็ตนั้นจะประกอบไปด้วยข้อมูลสำคัญซึ่งสามารถนำมาใช้เพื่อเป็นเงื่อนไขสำหรับการควบคุม traffic โดยไฟร์วอลล์ดังนี้

1. Source IP Address : IP Address ของต้นทาง เพื่อใช้ในการพิจารณาต้นทางของข้อมูลว่าอยู่ในเงื่อนไขที่อนุญาตหรือไม่
2. Destination IP Address : IP Address ของปลายทางเพื่อใช้ในการพิจารณาปลายทางของข้อมูลว่าอยู่ในเงื่อนไขที่อนุญาตหรือไม่
3. Protocol : ระบุโปรโตคอลที่อาศัยอยู่ใน IP Datagram ที่กำลังพิจารณานี้
4. Source Port : ระบุพอร์ตต้นทางสำหรับโปรโตคอลที่ใช้พอร์ตคือ TCP และ UDP
5. Destination Port : ระบุพอร์ตปลายทางที่แพ็คเก็ตนี้ต้องการติดต่อด้วย สำหรับโปรโตคอลที่ใช้พอร์ตคือ TCP และ UDP
6. ข้อมูลสำคัญอื่นๆตามลักษณะของโปรโตคอล เช่น TCP Flag,ICMP Message เป็นต้น

ข้อมูลทั้ง 6 ส่วนนี้จะมีได้อย่างครบถ้วนสมบูรณ์ก็ต่อเมื่อแพ็คเก็ตนั้นมีข้อมูลครบถ้วนทั้งหมดของ IP Datagram หากข้อมูลแพ็คเก็ตนั้นเป็นแฟรกเมนต์ อาจจะทำให้ข้อมูลในส่วนที่ 3 เป็นต้นไปซึ่งอยู่ในโปรโตคอลที่อยู่เลเยอร์สูงกว่า IP ไม่สมบูรณ์ จากข้อมูลที่สำคัญของแพ็คเก็ตข้างต้นนี้ จะสามารถนำมาใช้เป็นเงื่อนไขสำหรับการควบคุมการผ่านเข้าออกข้อมูลได้ โดยการพิจารณาข้อมูลทั้งหมดให้เป็นไปตามกฎที่ระบุไว้ ซึ่งเรียกว่า แอคเซสรูล(Access Rules) หรือกฎของการควบคุมการผ่านเข้าออกของแพ็คเก็ต

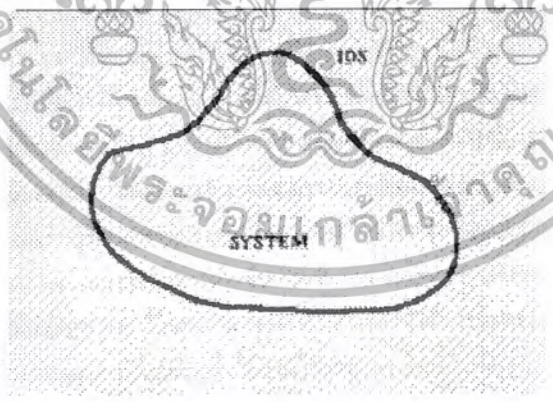
## 2.4 Intrusion Detection System

ปัจจุบันมีการใช้งานระบบคอมพิวเตอร์อย่างแพร่หลาย หลายๆบริษัท หรือหน่วยงานต่างๆได้นำระบบคอมพิวเตอร์มาใช้เพื่อพัฒนาศักยภาพการทำงานของตนให้มากขึ้น การทำงานของระบบคอมพิวเตอร์จะทำงานอย่างต่อเนื่อง และมีการออนไลน์ให้คนอื่นๆเข้ามาใช้งานระบบบางส่วนด้วย ซึ่ง

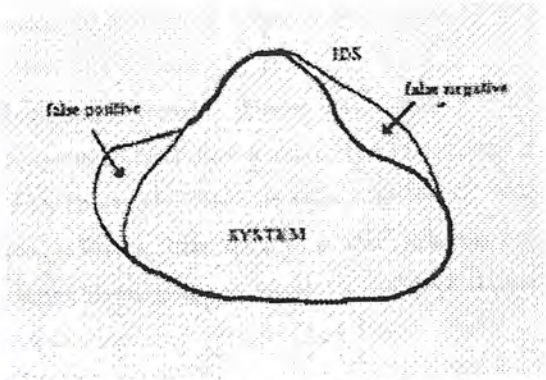
ปัญหาที่เกิดขึ้นตามมาก็คือปัญหาผู้บุกรุกเข้ามาสร้างความเสียหายให้กับระบบและอาจเข้ามาเพื่อขโมยข้อมูลที่สำคัญไป ปัญหาดังกล่าวจะเป็นปัญหากับผู้ดูแลระบบอย่างมาก เนื่องจากผู้ดูแลระบบต้องคอยป้องกันและแก้ไขปัญหาต่างๆอยู่เสมอๆ การทำงานของผู้ดูแลระบบนี้จะหนักมากหรือน้อยก็ขึ้นอยู่กับขนาดของระบบว่ามีขนาดใหญ่ และซับซ้อนมากน้อยเพียงใด ยิ่งระบบที่มีความซับซ้อนสูง มีความยุ่งยากในการดูแลมาก มีการออนไลน์ใช้งานอยู่ตลอดเวลา ผู้ดูแลระบบก็คงต้องอยู่ดูแลระบบตลอดเวลาซึ่งในทางปฏิบัติแล้วเป็นไปไม่ได้แน่นอน อีกทั้งปัญหาบางปัญหาผู้ดูแลระบบไม่สามารถตรวจสอบด้วยตาของตัวเองเลย ปัญหาการดูแลระบบจึงยุ่งยากขึ้นทุกวัน ทางออกที่จะช่วยแก้ปัญหาเหล่านี้ ก็คือการหาผู้ช่วยมาช่วยดูแลระบบตลอดเวลา สามารถมองเห็นในสิ่งที่ผู้ดูแลระบบมองไม่เห็น คอยแจ้งเตือนให้กับผู้ดูแลระบบยามมีเหตุการณ์ผิดปกติใดๆ และในบางครั้งก็สามารถแก้ไขปัญหาง่ายๆได้ด้วยตัวเองด้วย ผู้ช่วยที่สามารถช่วยแบ่งเบาภาระของผู้ดูแลระบบได้อย่างมากก็คือ ระบบตรวจจับผู้บุกรุก หรือ Intrusion Detection System

**2.4.1 ความหมาย Intrusion Detection System**

Intrusion Detection System คือ ระบบตรวจจับสัญญาณของความผิดปกติต่างๆที่เกิดขึ้นในระบบที่อยู่ในขอบเขตที่ระบบนี้มีหน้าที่ตรวจสอบ โดยทั่วไปจะหมายถึงระบบที่ตรวจจับความผิดปกติในบริษัท เขตทหาร โรงงาน หรือในเขตที่พักอาศัย แต่ในที่นี้เราจะหมายถึงโปรแกรมที่ใช้สำหรับตรวจจับความผิดปกติในระบบคอมพิวเตอร์ และระบบเครือข่ายเท่านั้น โดยตัวโปรแกรมจะมีความสามารถในการตรวจจับสัญญาณของความผิดปกติทุกๆอย่างที่เกิดขึ้นในระบบ ไม่ว่าจะเป็นภายในระบบคอมพิวเตอร์ ระบบปฏิบัติการ โปรแกรมที่รันอยู่ในเครื่อง การทำงานกับฐานข้อมูล หรือแม้แต่ข้อมูลที่วิ่งผ่านไปมาในเครือข่ายด้วย




*รูปที่2-4 ระบบ Intrusion Detection System*




*รูปที่ 2-5 ขอบเขตที่เหลื่อมกันของ IDS กับระบบทำให้เกิด flase positive และ false negative*

จากรูปที่ 2-4 ถ้าเรามองระบบเป็นเซตของการทำงานเซตหนึ่ง ระบบตรวจจับผู้บุกรุกที่แท้จริง (Ideal IDS) ต้องการทราบของเขตของเซตของระบบ ต้องทราบว่าระบบทำงานอะไรบ้าง การทำงานไหนปกติและผิดปกติ โดยระบบตรวจจับผู้บุกรุกที่มีประสิทธิภาพ จะทราบขอบเขตของระบบโดยไม่มีการเหลื่อมล้ำเข้าไปในระบบ หรือเหลื่อมล้ำออกนอกระบบอย่างเด็ดขาด

แต่ในระบบที่ใช้งานในโลกความเป็นจริง กรอบของระบบที่ IDS รับรู้อาจมีความเหลื่อมล้ำกันระบบที่ IDS ต้องตรวจสอบ ทำให้เกิดความผิดพลาดในการตรวจสอบได้ ซึ่งเราสามารถแบ่งความผิดพลาดในการตรวจสอบได้เป็นสองลักษณะคือ False Positive และ False Negative ดังรูปที่ 2-5 ซึ่งความผิดพลาดทั้งสองแบบมีรายละเอียดคือ

1. False Positive คือความผิดพลาดอันเนื่องมาจากการเกิดเหตุการณ์ซึ่งเป็นเหตุการณ์ปกติในระบบ แต่เป็นเหตุการณ์ที่ไม่ได้อยู่ในกรอบของระบบที่ IDS รับรู้ว่าเป็นปกติจึงทำให้ IDS คิดว่าเกิดเหตุการณ์ผิดปกติเกิดขึ้น ผลลัพธ์ก็คือ IDS จะแจ้งเตือนผู้ดูแลระบบว่าเกิดเหตุการณ์นั้นๆขึ้น แต่ก็เป็นการแจ้งเตือนเหตุการณ์ที่เป็นปกติ
2. False Negative คือความผิดพลาดอันเนื่องมาจากการเกิดเหตุการณ์ที่ผิดปกติเกิดขึ้น แต่เป็นเหตุการณ์ที่อยู่ในกรอบของระบบที่ IDS รับรู้ว่าเป็นเหตุการณ์ปกติในระบบ จึงทำให้ IDS ไม่แจ้งเตือนความผิดปกติที่เกิดขึ้น ถึงแม้ว่าจะเป็นเหตุการณ์ที่ผิดปกติก็ตาม

**2.4.2 ประเภทของระบบตรวจจับผู้บุกรุก**

เราสามารถแบ่งระบบตรวจจับผู้บุกรุกได้สองรูปแบบคือ ระบบที่ตรวจตราหาการทำงานที่ผิดไปจากการทำงานของระบบปกติเรียกว่า Anormaly Detection ซึ่งเป็นเหมือนกับการตรวจจับคนที่ไม่มีสิทธิ์ทำงานอยู่ในระบบ อีกรูปแบบหนึ่งของ IDS คือระบบที่ตรวจหาการทำงานที่ไม่ควรเกิดขึ้นในระบบเรียกว่า Misuse Detection ในที่นี้ก็เปรียบเสมือนเป็นบุคคลที่มีสิทธิ์ในระบบ สามารถเข้าออกในระบบได้ แต่เป็นผู้ที่ทำในสิ่งที่ระบบไม่อนุญาตให้ทำ หรือทำการใดๆที่อยู่นอกเหนือสิทธิ์ของตนในระบบ

**2.4.2.1 Anormaly Detection**

แนวความคิดของการทำ Anormaly Detection คือการหาเซตของการทำงานที่เป็นปกติย่อยๆขึ้นมาแล้วนำมารวมกันเพื่อให้ระบบ IDS ทราบข้อมูลของเซตการทำงานที่เป็นปกติทั้งหมดในระบบ หลังจากนั้นเมื่อให้ระบบ IDS ทำงาน ถ้าเกิดกรณีที่ IDS ตรวจจับการทำงานที่ไม่ได้อยู่ในเซตของการทำงานที่เป็นปกติ ระบบ IDS จะแจ้งเตือนต่อผู้ดูแลระบบทันที สำหรับการสร้างขอบเขตการทำงานของระบบนั้น อาจสร้างได้โดยการหาข้อมูลการทำงานที่เป็นปกติในระบบขึ้นมา โดยเอาข้อมูลการทำงานของผู้ใช้งานแต่ละคน เวลาที่มีการใช้งาน ทรัพยากรที่ผู้ใช้งานคนนั้นๆมักจะใช้บ่อยๆ หรือแม้กระทั่งข้อมูลในระบบ หรือในเครือข่ายก็สามารถนำมาสร้างเป็นเซตของระบบได้เช่นกันดังตัวอย่างรูปที่ 3 ในการหาเซตของการทำงานที่เป็นปกติทั้งหมด อาจเกิดการผิดพลาดขึ้นมาทำให้เกิดลักษณะของ false positive และ false negative ขึ้นมาได้เช่นกัน

**2.4.2.2 Misuse Detection**

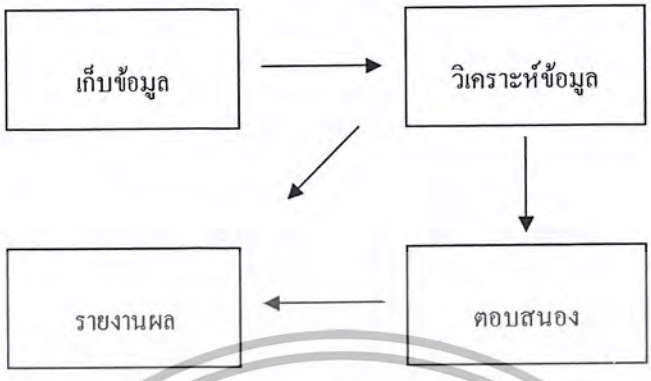
เป็นแนวความคิดที่ตรงข้ามกับ Anormaly Detection คือการทำ IDS รูปแบบนี้จะใช้ข้อมูลของการทำงานที่ผิดปกติต่างๆที่เคยเกิดขึ้นมาแล้ว สร้างเป็นฐานข้อมูลของการทำงานที่ผิดปกติให้ระบบ IDS จดจำไว้ ในการทำงานของ IDS ที่มีการทำงานแบบ Misuse จะนำข้อมูลที่อยู่ในระบบมาค้นหาในฐานข้อมูลว่ามีอยู่หรือไม่ ถ้าระบบ IDS มีข้อมูลของการทำงานรูปแบบนั้นๆอยู่ ก็แสดงว่าเกิดความผิดปกติขึ้นแล้ว ดังรูปที่ 4 มีการเก็บเซตของการทำงานที่ผิดปกติแล้วสร้างเป็นขอบเขตของการทำงานที่ผิดปกติ แต่ในการรวบรวมนี้อาจรวมเอาการทำงานที่เป็นปกติเข้าไปด้วย ทำให้เกิด false positive หรือในบางกรณีที่ไม่ได้เก็บข้อมูลความผิดปกติไว้ก็ทำให้เกิดกรณีของ false negative ได้เช่นกัน ซึ่งในการทำงานของ Misuse Detectin นี้ จะมีข้อเสียคือจะไม่สามารถตรวจจับการบุกรุกชนิดใหม่ๆได้ เนื่องจากต้องมีข้อมูลของการบุกรุกอยู่ก่อนจึงจะตรวจจับได้

**2.4.3 การทำงานของ Intrusion Detection System**

ระบบตรวจจับผู้บุกรุกแต่ละแบบจะมีหน้าที่การทำงานที่แตกต่างกันออกไป บางตัวจะตรวจจับความผิดปกติในระบบเครือข่าย บางตัวจะตรวจจับความผิดปกติในระบบฐานข้อมูล แต่โดยการทำงานทั้งหมดแล้วเราสามารถสรุปการทำงานของระบบตรวจจับผู้บุกรุกได้ 3 ขั้นตอนคือการเก็บข้อมูลระบบ การวิเคราะห์ข้อมูลที่เก็บได้ และการรายงานผลการทำงานให้ผู้ดูแลระบบหรือผู้ที่เกี่ยวข้องทราบ

ในการทำงานหลักๆของระบบตรวจจับผู้บุกรุก อาจมีขั้นตอนเสริมอยู่ขั้นตอนหนึ่ง คือการตอบสนองต่อการบุกรุกนั้นๆ การทำงานในขั้นตอนนี้จะใช้ในกรณีที่การบุกรุกเป็นรูปแบบการบุกรุกที่ระบบตรวจจับผู้บุกรุกสามารถแก้ไขด้วยตัวเองได้ ซึ่งในบางระบบอาจไม่มีการทำงานในส่วนนี้ ระบบที่ไม่ได้ทำงานแบบตอบสนองทันที(realtime) คือจะเก็บข้อมูลของระบบไว้ก่อน แล้วจึงวิเคราะห์ข้อมูลภายหลัง เมื่อทำการวิเคราะห์ข้อมูลแล้วพบว่ามีการบุกรุกเข้าสู่ระบบก็จะทำการแจ้งเตือนในขั้นตอนการทำรายงานผลการทำงาน ระบบตรวจจับผู้บุกรุกที่ไม่มีการตอบสนองต่อการบุกรุก ก็มักใช้ในงานที่ไม่มี

ความสำคัญมากนัก แต่ต้องการความถูกต้องสูง โดยลำดับการทำงานของระบบตรวจจับผู้บุกรุกสามารถมองเป็นขั้นตอนต่างๆได้ดังรูปที่ 5-5




*รูปที่ 2-6 การทำงานของ intrusion detection system*

**2.4.3.1 การเก็บข้อมูลในระบบ**

จากที่ได้กล่าวมาแล้วว่าระบบตรวจจับผู้บุกรุกจะมีการทำงานที่แตกต่างกันไป หน้าที่นการเก็บข้อมูลของระบบที่ต้องการตรวจสอบก็แตกต่างกันไปตามหน้าที่ของระบบตรวจจับผู้บุกรุกด้วย โดยเราสามารถแบ่งการเก็บข้อมูลของระบบที่ต้องการตรวจสอบออกเป็นกลุ่มต่างๆได้ 4 กลุ่มด้วยกันคือ มีการเก็บข้อมูลในชั้นแอปพลิเคชั่น(Application-based Approach) เพื่อนำมาตรวจสอบการทำงานของแอปพลิเคชั่นต่างๆว่าผิดปกติหรือไม่ , การเก็บข้อมูลของการทำงานของเครื่อง(Host-based Approach) เพื่อนำมาตรวจสอบการทำงานของระบบของเครื่องที่ใช้งานอยู่ , การเก็บข้อมูลการเปลี่ยนแปลงข้อมูลในระบบ (Target-based Approach) เพื่อนำมาตรวจสอบว่าข้อมูลมีการเปลี่ยนแปลงอย่างไร และการเก็บข้อมูลเครือข่าย (Network-based Approach) เพื่อนำมาตรวจสอบว่ามีการบุกรุกทางระบบเครือข่ายหรือไม่อย่างไร

**2.4.3.1.1 การเก็บข้อมูลในชั้นแอปพลิเคชั่น**

การเก็บข้อมูลในชั้นแอปพลิเคชั่นนั้น เป็นการเก็บข้อมูลที่โปรแกรมต่างๆสร้างขึ้นมาเพื่อรายงานผลการทำงานของโปรแกรมนั้นๆ เช่น log file หรือ error message ต่างๆของเว็บเซิร์ฟเวอร์ , ไฟร์วอลล์ หรือโปรแกรมบริหารฐานข้อมูล รวมถึงข้อมูลของการทำงานตอบสนองกันระหว่างผู้ใช้งานโปรแกรมและข้อมูลที่เกี่ยวข้อง ในการเก็บข้อมูลในลักษณะนี้ นอกจากเป็นการทำงานด้านการรักษาความปลอดภัยในระบบแล้ว ยังช่วยในการวิเคราะห์ระบบและปรับปรุงระบบเนื่องจากผลจากการวิเคราะห์ข้อมูลที่ได้ทำให้ทราบว่าการใช้งานโปรแกรมไหนในระบบมีมากน้อยอย่างไร และควรให้ความสำคัญกับการทำงานตรงส่วนไหน แต่การทำงานในส่วนนี้ก็ยังมีข้อเสีย ในกรณีที่มีผู้บุกรุกแล้วทำการเปลี่ยนแปลงข้อมูลดังกล่าว ทำให้การตรวจจับทำไม่ได้ ดังนั้นจึงควรเก็บข้อมูลดังกล่าวไว้ในที่ๆปลอดภัยด้วย

**2.4.3.1.2 การเก็บข้อมูลของการทำงานของเครื่อง**

สำหรับการเก็บข้อมูลของการทำงานของเครื่อง จะเน้นไปในการเก็บข้อมูลของระบบปฏิบัติการเป็นหลัก ข้อมูลที่เก็บได้จะอยู่ในรปของการแจ้งเตือนในระบบเช่นการตั้งค่าบางอย่างไม่สมบูรณ์ การทำงานของโพรเซสบางโพรเซสมีปัญหาหรือปัญหาของฮาร์ดแวร์เป็นต้น หรืออาจจะอยู่ในรูปข้อมูลของการทำงานโดยปกติของระบบปฏิบัติการนั้นๆ เช่นข้อมูลการใช้งานของยูสเซอร์แต่ละคน ใคร ทำอะไร เมื่อเวลาเท่าไร ซึ่งเมื่อนำข้อมูลเหล่านี้ไปวิเคราะห์แล้ว จะได้ผลการวิเคราะห์ในลักษณะมีการใช้งานอย่างไม่ถูกต้องหรือไม่ ถ้ามี ใครเป็นผู้ใช้งานนั้นๆ เมื่อเวลาเท่าไร จากที่ไหน ข้อดีอีกข้อหนึ่งก็คือการเก็บข้อมูลในลักษณะนี้สามารถเก็บข้อมูลที่ถูกเข้ารหัสได้ด้วย ส่วนข้อเสียของการเก็บข้อมูลการทำงานของเครื่องก็คือข้อมูลที่ได้มักจะมีขนาดใหญ่ ระบบที่ทำการเก็บข้อมูลลักษณะนี้จะมี overhead สูงขึ้น นอกจากนี้โปรแกรมที่ทำการเก็บข้อมูลและวิเคราะห์ข้อมูลยังขึ้นอยู่กับ platform และมีราคาสูงด้วย

**2.4.3.1.3 การเก็บข้อมูลการเปลี่ยนแปลงข้อมูลในระบบ**

การเก็บข้อมูลการเปลี่ยนแปลงข้อมูลในระบบจะใช้หลักการของ integrity analysis ในการตรวจสอบการเปลี่ยนแปลงข้อมูลต่างๆในระบบ ในระบบตรวจจับผู้บุกรุกบางระบบจะใช้ checksum เป็นตัวบ่งบอกการเปลี่ยนแปลงในระบบ วิเคราะห์ลักษณะนี้จะเริ่มจากการสร้างฐานข้อมูล signature ของไฟล์ต่างๆ ในระบบปกติไว้ เมื่อระบบทำงานก็จะทำการตรวจสอบค่า signature นี้ไปเรื่อยๆ อาจเป็นวันละครั้งสองครั้ง หรือบ่อยกว่านั้นแล้วแต่ความสำคัญของระบบ เมื่อมีข้อมูลไฟล์ไหนมีการเปลี่ยนแปลงก็จะทราบได้ ข้อดีของการทำ integrity analysis ลักษณะนี้ช่วยให้การตรวจจับการบุกรุกที่มีการเปลี่ยนแปลงระบบ เช่น ทำการเจาะระบบแล้วทำการวางโทรจัน หรือ Back Door ไว้ได้ หรับการแก้ไขเมื่อทราบว่ามีผู้บุกรุกมาเปลี่ยนแปลงไฟล์ข้อมูลในระบบก็ทำการแก้ไขเฉพาะไฟล์ที่ถูกแก้ไขเท่านั้น ไม่จำเป็นต้องทำการติดตั้งระบบใหม่ แต่ระบบนี้ก็มีข้อเสียในกรณีที่เมื่อมีไฟล์ในระบบเยอะมาก การเก็บข้อมูล signature ของไฟล์ต่างๆและการวิเคราะห์ข้อมูลก็จะใช้เวลานาน ระบบนี้จึงไม่เหมาะในการทำงาน real time เพราะทำให้เกิด overhead ในระบบสูงมาก

**2.4.3.1.4 การเก็บข้อมูลเครือข่าย**

การเก็บข้อมูลเครือข่ายนั้นนับวันจะมีความสำคัญขึ้นเรื่อยๆ เพราะการบุกรุกทางเครือข่ายมีมากขึ้นเรื่อยๆ การเก็บข้อมูลแบบนี้จะใช้การดักจับข้อมูลที่ผ่านไปมาในเครือข่าย โดยการทำให้เน็ตเวิร์คการ์ดอยู่ใน promiscuous mode เมื่อเน็ตเวิร์คการ์ดอยู่ในโหมดดังกล่าว จะสามารถรับข้อมูลทุกอย่างที่อยู่ในเครือข่ายได้ การเก็บข้อมูลเครือข่ายในลักษณะนี้สามารถตรวจจับการโจมตีทางเครือข่ายได้เช่นการทำ SYN flood การทำ port scan หรือการส่งแพ็คเก็ตปริมาณมากมารบกวนในระบบ แต่เนื่องจากการเก็บข้อมูลเครือข่ายนี้ใช้ลักษณะการทำสนิฟเฟอร์เป็นหลัก จึงไม่สามารถทำงานในเครือข่ายที่เป็นเครือข่ายสวิชชิ่ง ไม่สามารถทำงานในระบบเครือข่ายที่เขารหัสข้อมูล หรือไม่สามารถเก็บข้อมูลในเครือข่ายที่มีข้อมูลหนาแน่นได้ เพราะการทำงานในการเก็บข้อมูลอาจไม่เร็วพอที่จะเก็บข้อมูลทั้งหมดผ่านไปมาในระบบได้

ข้อเสียอีกข้อหนึ่งของการเก็บข้อมูลนี้ก็คือข้อมูลที่เก็บมีขนาดใหญ่มาก โดยเฉพาะอย่างยิ่งในระบบเครือข่ายที่มีการรับส่งแพ็กเก็ตปริมาณมากอยู่ตลอดเวลา

นอกจากนี้ยังมีการเก็บข้อมูลโดยทำการเก็บข้อมูลทั้ง Application-based , Host-based และ Network-based ร่วมกันด้วย เพื่อให้ได้ข้อมูลระบบอย่างครบถ้วน และใช้ข้อมูลจากทั้งสามแหล่งมาประกอบกันในการวิเคราะห์ความผิดปกติที่เกิดขึ้นในระบบด้วย การเก็บข้อมูลในลักษณะนี้เรารวมเรียกว่า "Integrated-based"

**2.4.3.2 การวิเคราะห์ข้อมูลระบบ**

เมื่อได้ข้อมูลของระบบที่จำเป็นแล้ว ในขั้นตอนต่อมาเราก็จะนำเอาข้อมูลที่ได้มาวิเคราะห์ว่าระบบของเรามีความผิดปกติเกิดขึ้นหรือไม่ การวิเคราะห์ข้อมูลเราสามารถแบ่งการทำงานตามรูปแบบการวิเคราะห์ข้อมูลได้ 2 รูปแบบคือ ทำการวิเคราะห์ในขณะที่เก็บข้อมูล (Real Time) หรือจะเก็บข้อมูลทั้งหมดไว้ก่อน แล้วจึงวิเคราะห์ข้อมูลนั้นๆภายหลัง (Batch) ในการวิเคราะห์ข้อมูลทั้งสองรูปแบบก็มีข้อดีข้อเสียแตกต่างกันไป

**2.4.3.2.1 การวิเคราะห์ในขณะที่เก็บข้อมูล(Real time)**

ในการวิเคราะห์ข้อมูลที่ได้ในขณะที่เก็บข้อมูล หรือแบบ Real Time นั้น ระบบจะจัดเก็บข้อมูลวิเคราะห์ข้อมูล และรายงานผลการวิเคราะห์ ในช่วงเวลาเดียวกัน เมื่อเกิดข้อผิดพลาดขึ้นสามารถตอบสนองได้ทันท่วงที ระบบที่ทำงานแบบ Real Time มีการแจ้งเตือนหลายๆแบบ เช่น E-mail หรือ Instant Messaging ให้กับผู้ดูแลระบบได้ในช่วงเวลาที่มีการบุกรุกได้ ในการตรวจสอบระบบแบบ Real Time ทำให้ระบบสามารถตรวจสอบข้อผิดพลาดได้อย่างรวดเร็ว แต่ก็ขึ้นอยู่กับความเร็วในการวิเคราะห์ข้อมูลด้วย ถ้าข้อมูลมีความซับซ้อนมากๆ ก็จะใช้เวลามากตามมาเช่นเดียวกัน เมื่อระบบทำการตรวจสอบการบุกรุกได้ในขณะที่เพิ่งเกิดการบุกรุกขึ้น ผู้ดูแลระบบ หรือระบบตรวจจับผู้บุกรุกเองสามารถแก้ไขปัญหาที่เกิดขึ้นได้ทันที แต่ทั้งนี้ก็ขึ้นอยู่กับความเร็วในการวิเคราะห์ข้อมูล และชนิดของปัญหาว่ามีความยุ่งยากในการแก้ปัญหามากน้อยเพียงไรด้วย

ระบบ Real Time ทำงานได้อย่างรวดเร็ว แต่การทำงานที่รวดเร็วดังกล่าวก็ต้องแลกกับการใช้หน่วยความจำปริมาณมาก และการประมวลผลที่รวดเร็วมากด้วย อีกทั้งการตอบสนองต่อการบุกรุกโดยอัตโนมัติ อาจทำให้เกิดความเสียหายกับระบบมากกว่าเดิม เพราะว่าในบางครั้งการทำงานที่รวดเร็วเกินไปของระบบนี้ ทำให้เกิดความผิดพลาดในการวิเคราะห์ข้อมูลจนประมวลผลการทำงานปกติ กลายเป็นการทำงานที่ผิดปกติ แล้วทำการแก้ไขตามข้อมูลที่มีอยู่ ก็ยิ่งทำให้ระบบมีความเสียหายมากกว่าเดิม ระบบตรวจจับผู้บุกรุกที่ทำงานแบบ Real time จึงเหมาะกับระบบที่มีข้อมูลที่ต้องพิจารณาน้อย ต้องการการรายงานอย่างรวดเร็วเมื่อผิดปกติ และข้อูลที่ต้องนำมาวิเคราะห์ไม่ซับซ้อนมากนัก

**2.4.3.2.2 การวิเคราะห์ข้อมูลภายหลังจากที่เก็บข้อมูล(Batch)**

อีกรูปแบบหนึ่งในการวิเคราะห์ระบบที่ใช้กันคือ การวิเคราะห์ข้อมูลภายหลังจากที่เก็บข้อมูลไว้แล้วหรือการทำงานแบบ Batch การทำงานในแบบนี้เหมาะกับงานที่ไม่จำเป็นต้องตอบสนองทันทีเมื่อเกิดความผิดปกติขึ้น แต่ให้มีการบันทึก และรายงานว่าเกิดความผิดปกติขึ้น การทำงานจะใช้หน่วยความจำและการประมวลผลน้อยกว่าแบบแรก แต่ก็ใช้เนื้อที่ในการเก็บข้อมูลมากกว่าแบบแรกแน่นอน เหมาะกับองค์กรที่มีบุคคลากรจำกัด ข้อเสียของการทำงานแบบ Batch คือมักแก้ปัญหาที่เกิดขึ้นไม่ทัน เพราะกว่าจะทราบว่าเกิดปัญหาขึ้น ปัญหานั้นก็เกิดขึ้นนานมาก ความเสียหายที่เกิดขึ้นก็แก้ไขได้ยาก

ไม่ว่าจะเป็นการวิเคราะห์ระบบจะเป็นแบบ Real Time หรือเป็นแบบ Signature Analysis เป็นการวิเคราะห์ระบบที่เหมือนกัน คือทำการหารูปแบบของการโจมตีในข้อมูลที่ได้รับมา (Signature Analysis) , วิธีการวิเคราะห์ทางสถิติ (Statistical Analysis) และวิธีการตรวจสอบการเปลี่ยนแปลงของระบบ (Integrity Analysis)

1) การหารูปแบบของการโจมตี (Signature Analysis)

ในวิธีการวิเคราะห์ระบบแบบ Signature Analysis เป็นการวิเคราะห์ข้อมูลโดยการหาสัญญาณของการโจมตี (Attack Signature) การทำงานจะทำโดยการเปรียบเทียบรูปแบบของข้อมูลกับรูปแบบของการโจมตีในฐานข้อมูล ว่ามีความคล้ายคลึงกันหรือ ไม่ ถ้ามีความคล้ายคลึงกันก็แสดงว่ามีการโจมตีเกิดขึ้นแล้ว ในการเปรียบเทียบอาจเป็นแบบอย่างง่ายคือการเปรียบเทียบข้อมูลว่ามีความเข้ากันได้กับข้อมูลของการโจมตีเพียงใด หรือเป็นแบบที่มีความสลับซับซ้อนขึ้นอีกเช่น การทำ state transition เป็นต้น

สำหรับโปรแกรมตรวจจับผู้บุกรุกที่มีจำหน่ายในท้องตลาด ส่วนใหญ่จะทำงานในลักษณะของการเปรียบเทียบรูปแบบของระบบกับการโจมตีในฐานข้อมูล ซึ่งบริษัทผู้ขายจะให้ฐานข้อมูลของการโจมตีไว้ด้วย ผู้ใช้งานสามารถป้อนข้อมูลการโจมตีหรือกฎในการตรวจจับได้ด้วย ซึ่งโปรแกรมในลักษณะนี้มักจะมีการอัพเดทข้อมูลในฐานข้อมูลบ่อยๆ เพื่อเพิ่มความสามารถในการวิเคราะห์ข้อมูลในระบบ การวิเคราะห์ระบบด้วยวิธี signature analysis นี้จะมี overhead ไม่มากนัก เพราะเป็นเพียงการเปรียบเทียบข้อมูลกับข้อมูลในฐานข้อมูลเท่านั้น และยังเพิ่มความรวดเร็วในการทำงานโดยรวมมากขึ้น เพราะสามารถนำข้อมูลในฐานข้อมูลมาเป็นกฎในการกรองข้อมูลที่จะเก็บให้น้อยลงด้วย แต่วิธีการนี้ก็มีข้อเสียเพราะฐานข้อมูลจะมีขนาดใหญ่ขึ้นเรื่อยๆ ต้องมีการอัพเดทฐานข้อมูลบ่อยๆ

2) วิธีการวิเคราะห์ทางสถิติ (Statistical Analysis)

วิธีการวิเคราะห์ทางสถิติ (Statistical Analysis) เป็นวิธีการวิเคราะห์ข้อมูลอีกแบบหนึ่งที่มีแนวคิดตรงข้ามกับวิธีการแรก คือจะหารูปแบบของการทำงานที่เป็นปกติ แล้วสร้างเป็นโพรไฟล์ (Profile) เก็บไว้ก่อน ในการวิเคราะห์ข้อมูล จะเปรียบเทียบข้อมูลกับโพรไฟล์ที่สร้างไว้ ถ้าไม่เข้ากันก็แสดงว่ามีความผิดปกติเกิดขึ้นแล้ว สำหรับโพรไฟล์นั้นอาจแยกเป็นโพรไฟล์สำหรับออปเจ็กต์ต่างๆ ในระบบเช่นยูสเซอร์ , ไฟล์ , ไดเร็กทอรีและอุปกรณ์ต่างๆ โดยรายละเอียดที่เก็บอยู่ในโพรไฟล์จะมีข้อมูลของจำนวนครั้งที่เข้าสู่ระบบ , จำนวนครั้งที่เข้าสู่ระบบผิดพลาด , เวลา และข้อมูลอื่นๆที่จำเป็น ค่าแต่ละค่าที่เก็บจะเป็นค่าของการใช้งานที่เป็นปกติ การตรวจจับว่าเกิดความผิดปกติขึ้นแล้ว จะดูจากค่าที่ไม่เข้ากับโพรไฟล์เป็นต้น

ยกตัวอย่างเช่น ในการทำงานปกติ ผู้ใช้งานฐานข้อมูลจะมีการเข้าใช้ข้อมูลในฐานข้อมูลตั้งแต่เวลา 8 โมงเช้าถึง 6 โมงเย็นเท่านั้น แต่ข้อมูลที่ตรวจจับได้มีการเข้าใช้ฐานข้อมูลตอนตีสอง ซึ่งก็สามารถบอกได้ว่ามีการบุกรุกเกิดขึ้นแล้ว

เนื่องจากวิธีการวิเคราะห์ข้อมูลแบบ Statistical Analysis เป็นการตรวจจับโดยใช้หลักการของการอนุญาตให้ใช้งานในการทำงานทั่วๆ ไปและไม่อนุญาติให้ใช้งานนอกเหนือจากที่เคยใช้โดยสามารถตรวจจับการบุกรุกในรูปแบบที่ซับซ้อนได้ด้วย เพราะเราถือว่าการทำงานที่สลับซับซ้อนมักจะไม่เหมือนกับการทำงานโดยปกติ แต่วิธีการวิเคราะห์ข้อมูลแบบนี้ก็มีข้อเสียเนื่องจากเป็นการเก็บข้อมูลการทำงานที่เป็นปกติไว้เพื่อเปรียบเทียบกับการทำงานที่ผิดปกติ เมื่อทำการบุกรุกในลักษณะเดิมๆเป็นเวลานาน ก็จะทำให้โพรไฟล์มีการเปลี่ยนแปลง ระบบตรวจจับก็จะเห็นว่าการโจมตีในลักษณะนั้นเป็นการทำงานที่เป็นปกติแทน และไม่สามารถตรวจจับการทำงานที่ผิดปกติในลักษณะนั้นได้อีกต่อไป และไม่เหมาะกับองค์กรที่มีการเปลี่ยนแปลงการทำงานบ่อยๆ เพราะทำให้โพรไฟล์มีขนาดใหญ่ ทำให้ระบบตรวจจับรวน และมีความผิดพลาดในการวิเคราะห์สูง

3) วิธีการตรวจสอบการเปลี่ยนแปลงของระบบ (Integrity Analysis)

วิธีสุดท้ายที่นิยมใช้กันในการวิเคราะห์ข้อมูลคือ วิธีการตรวจสอบการเปลี่ยนแปลงของระบบ (Integrity Analysis) ลักษณะการทำงานของวิธีการนี้คือการหาว่ามีการเปลี่ยนแปลงเกิดขึ้นในระบบหรือไม่ เช่นมีไฟล์ไหนมีการเปลี่นแปลง หรือมีออปเจ็กต์อะไรที่มีการเปลี่ยนแปลงคุณสมบัติบ้าง แล้วทำการแจ้งเตือนกับผู้ดูแลระบบ ในการวิเคราะห์ลักษณะนี้จะใช้แอชอัลกอริทึม (hash algorithm) เพื่อสร้างแมสเซชไดเจส (Message digest) ของข้อมูล แล้วทำการเปรียบเทียบแมสเซชไดเจสของข้อมูลในช่วงเวลาต่างๆว่าเหมือน หรือต่างกันหรือ ไม่ ถ้าแมสเซชไดเจสต่างกันก็แสดงว่าข้อมูลมีการเปลี่ยนแปลง Integrity Analysis สามารถตรวจจับการบุกรุกที่เข้ามาเปลี่ยนแปลงข้อมูลในระบบ หรือมีการติดตั้งโปรแกรมเช่น sniffer , rootkit ต่างๆ ในระบบได้ แต่ก็มีข้อเสียคือการวิเคราะห์ระบบในลักษณะนี้จะทำงานเป็นแบบ batch เท่านั้น ไม่เหมาะกับการทำ real time อย่างยิ่งเพราะจะทำให้เปลืองทรัพยากรมาก

**2.4.3.3 การตอบสนอง**

เมื่อมีการตรวจพบว่ามีการบุกรุกเกิดขึ้นในระบบ สำหรับระบบตรวจจับที่ทำงานแบบ real time จะมีการตอบสนองต่อการบุกรุกเพื่อไม่ให้เกิดความเสียหาย หรือบรรเท่าความเสียหายที่เกิดขึ้น สำหรับระบบที่ทำงานเป็นแบบ batch การตอบสนองอาจทำได้ไม่มากนัก เพราะการบุกรุกนั้นเกิดไปแล้ว ความเสียหายก็เกิดขึ้นแล้ว การตอบสนองอาจอยู่ในรูปการบรรเท่าไม่ให้ความเสียหายมีเพิ่มมากขึ้นเท่านั้น การตอบสนองต่อการบุกรุกนั้นแบ่งออกได้เป็นสามแบบด้วยกันคือการเปลี่ยนแปลงสภาพของระบบ การแก้ไขความผิดพลาดให้ถูก และการแจ้งเตือนผู้ดูแลระบบเมื่อถูกบุกรุก

**2.4.3.3.1 การเปลี่ยนแปลงสภาพของระบบ**

สำหรับการตอบสนองต่อการบุกรุก โดยการเปลี่ยนแปลงสภาพของระบบที่ถูกโจมตีก็เพื่อแก้ปัญหาหรือลดความเสียหายที่จะเกิดขึ้นเช่นตัดการเชื่อมต่อระหว่างระบบกับผู้บุกรุกออกจากกัน การตั้ง

ค่าอุปกรณ์เครือข่ายหรือไฟร์วอลล์ไม่ได้มีการติดต่อกับระบบของผู้บุกรุกอีกต่อไป และการหาข้อมูลเกี่ยวกับการโจมตีโดยอัตโนมัติเพื่อตรวจหาผู้บุกรุกต่อไป

**2.4.3.3.2 การแก้ไขความผิดพลาดให้ถูก**

การแก้ไขระบบ เป็นการตอบสนองต่อปัญหาที่เกิดขึ้นแล้วในระบบ โดยปกติแล้วผู้บุกรุกมักเปลี่ยนแปลงค่าต่างๆในระบบ โดยเฉพาะเข้ามาทำการเปลี่ยนแปลงข้อมูลในระบบตรวจจับผู้บุกรุกเพื่อไม่ให้สามารถตรวจจับการบุกรุกได้ การแก้ไขระบบก็เพื่อให้ระบบดังกล่าวสามารถทำงานได้อย่างเป็นปกติดี

**2.4.3.3.3 การแจ้งเตือนผู้ดูแลระบบ**

สุดท้ายเป็นการแจ้งเตือนผู้ดูแลระบบ โดยปกติมักแจ้งเตือนผู้ดูแลทันทีเมื่อทำการวิเคราะห์ได้ว่ามีความผิดปกติเกิดขึ้น เพื่อให้ผู้ดูแลระบบรับรู้และสามารถแก้ไขระบบได้ทันท่วงที สำหรับการแจ้งเตือนนี้ผู้ดูแลระบบสามารถเลือกได้ว่าจะแจ้งเตือนใครบ้าง และทำการแจ้งเตือนในรูปแบบไหนอาจเป็น E-mail , Pager หรือ Instant Messaging ต่างๆ

**2.4.3.4 การรายงานผลการทำงาน**

เมื่อระบบตรวจจับผู้บุกรุกทำการวิเคราะห์ระบบ และตรวจพบความผิดปกติในระบบ อาจมีการตอบสนองต่อความผิดปกตินั้นถ้าทำได้ จากนั้นระบบตรวจจับผู้บุกรุกต้องมีการรายงานผลให้กับผู้ดูแลระบบให้ทราบในรูปแบบต่างๆ โดยรายละเอียดของการรายงานผลนั้น จะบอกถึงช่องโหว่ในระบบ การแก้ไขปัญหาคร่าวๆบางครั้งอาจมีรายละเอียดของความรู้พื้นฐานบางอย่างของระบบ ที่ทำให้เกิดการบุกรุกลักษณะนั้นๆได้ การรายงานผลการทำงาน นอกจากเป็นการรายงานต่อผู้ดูแลระบบเพื่อให้ทราบการทำงานหรือจุดอ่อนในระบบแล้ว ยังเป็นประโยชน์ต่อการวิเคราะห์สถานะของระบบ และการวิเคราะห์ความปลอดภัยในระบบอีกด้วย

**2.3.4 ความสำคัญของ IDS**

เมื่อได้ทราบถึงการทำงานคร่าวๆของระบบตรวจจับผู้บุกรุกแล้ว อาจคิดว่าระบบตรวจจับผู้บุกรุกนั้นไม่มีความสำคัญเพราะในเมื่อมีการใช้งานไฟร์วอลล์อยู่แล้ว แต่ความเป็นจริงแล้วถึงแม้ว่าระบบจะมีไฟร์วอลล์อยู่แล้วก็ยังจำเป็นต้องใช้ระบบตรวจจับผู้บุกรุกด้วยเพราะในงานบางอย่างไฟร์วอลล์ไม่สามารถช่วยได้

จุดประสงค์ของการใช้งานไฟร์วอลล์นั้น สร้างขึ้นเพื่อเป็นเสมือนตัวป้องกันระบบให้แยกตัวออกมาจากเครือข่ายที่ไม่ปลอดภัย เป็นเมืองหน้าด่านของระบบ เป็นผู้ป้องกันการบุกรุกจากภายนอก แต่ระบบตรวจจับผู้บุกรุกนั้นมีจุดประสงค์ที่แตกต่างไป โดยเป็นผู้เฝ้าดูระบบ และเป็นผู้เตือนเมื่อเกิดความผิดปกติเกิดขึ้น ยกตัวอย่างในอาคารใหญ่ๆจะมีคนคอยตรวจตราอยู่ภายใน มีกริ่งสัญญาณเตือนเมื่อเกิดความผิดปกติเกิดขึ้น ซึ่งก็ช่วยให้แก้ปัญหาได้ทันท่วงที และในกรณีที่มีความผิดปกติเกิดขึ้น แต่ไม่สามารถ

ตรวจจับได้ในขณะนั้น ระบบตรวจจับผู้บุกรุกก็มีการจัดเก็บข้อมูลการใช้ระบบไว้ จึงสามารถนำข้อมูลดังกล่าวมาวิเคราะห์หาความผิดปกติได้ภายหลัง โดยจุดประสงค์ในการสร้างระบบตรวจจับผู้บุกรุกและไฟร์วอลล์ จึงต่างกันโดยสิ้นเชิง แต่ถึงแม้ว่าจุดประสงค์การทำงานของระบบตรวจจับผู้บุกรุกและไฟร์วอลล์จะแตกต่างกัน แต่ทั้งสองก็สามารถทำงานร่วมกันและทำให้ประสิทธิภาพการรักษาความปลอดภัยในระบบดีขึ้นด้วย

# บทที่ 3

# IPTABLES

Linux สามารถใช้งานเป็นไฟร์วอลล์ได้ตั้งแต่เคอร์เนล 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

## 3.1 รูปแบบการใช้งาน iptables เบื้องต้น

iptables จะมีรูปแบบการใช้งานดังนี้คือ

```
iptables [table] <command> <match> <target/jump>
```

*รูปที่ 3-1 รูปแบบการใช้งาน iptables*

โดย rule ที่เขียนขึ้นจะเป็นเป็นตัวบอกเคอร์เนลว่าให้กระทำ action อย่างไร ในกรณีที่พบ packet ตรงตามที่ระบุไว้

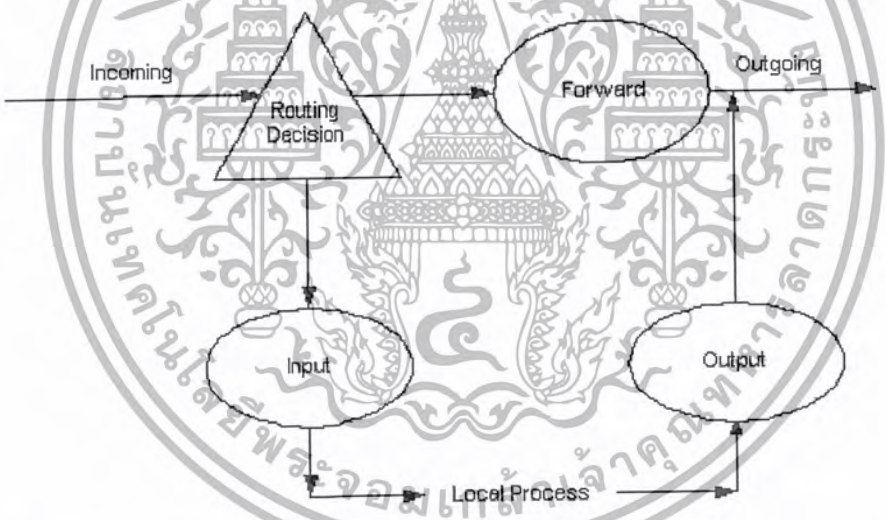
- **[table]** หมายถึง ตารางหรือ table ที่ต้องการระบุ เช่น iptables -t nat หมายถึงให้ทำงานกับ nat table
  ในกรณีที่ไม่ได้ระบุตาราง iptables จะถือว่าคำสั่งดังกล่าวระบุถึง filter table โดยอัตโนมัติ
- **<command>** จะเป็นตัวสั่งให้ iptables ทำในสิ่งที่ต้องการ เช่น iptables -A INPUT ซึ่งหมายถึงให้สร้าง rule 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

**3.2 Table**

iptables สามารถทำงานได้กับตาราง(table) 3 ตารางหลัก สามารถระบุตารางได้โดยใช้

**3.2.1 Filter table** ใช้สำหรับกรอง packet มี 3 built-in chain คือ INPUT, OUTPUT, FORWARD ซึ่งเป็นตารางที่ใช้งานมากที่สุด เป็นจุดที่ใช้ในการตรวจสอบและควบคุมการผ่านเข้าออกของ packet ถ้าหากจะพิจารณาการไหลเวียนของ packet เฉพาะในส่วนของ filter table โดยไม่สนใจ table อื่นๆ นั้น ก็พอจะแสดงให้เห็นได้ดังภาพที่ 1 โดยเมื่อ packet เข้ามาในระบบ มันจะเข้าไปยัง routing decision เพื่อตัดสินใจว่า packet จะถูกส่งไปที่ใด

- ในกรณีที่ packet ถูกส่งผ่านไปยังเครื่องอื่น packet นั้นจะต้องถูกตรวจสอบโดย rule ใน FORWARD chain
- ถ้า packet นั้น มีเป้าหมายเป็นเครื่องปัจจุบัน (เครื่องที่รัน iptables อยู่นี้ เรียกอีกอย่างว่า linux box) ตัว packet จะถูกตรวจสอบโดย rule ใน INPUT chain
- และในกรณีที่ packet ถูกสร้างจากเครื่องปัจจุบัน (linux box) ตัว packet จะถูกตรวจสอบจาก rule ใน OUTPUT chain ก่อนที่จะถูกส่งออกไป




*รูปที่ 3-2 แสดงให้เห็นว่า packet มีเส้นทางการเดินทางอย่างไรเมื่อเข้ามาในระบบ (filter table)*

ดังภาพ iptables ประกอบไปด้วย built-in chain จำนวน 3 chain ซึ่งไม่สามารถลบได้คือ INPUT, OUTPUT, FORWARD เมื่อเครื่องคอมพิวเตอร์เริ่มทำงานในครั้งแรก ทั้งสาม chain จะมี default policy เป็น ACCEPT ซึ่งหมายความว่าอนุญาตให้ทุกอย่างผ่านเข้าออกได้หมด และสำหรับ FORWARD chain นั้น ถึงแม้จะได้กำหนดให้ policy เป็น ACCEPT แล้ว packet ก็จะยังไม่สามารถถูก forward ไปยังจุดหมายที่ต้องการได้ ตราบใดที่ยังไม่ได้เซ็ตให้ enable IP forwarding

**3.2.2 Nat table** ใช้สำหรับการแปลงแอดเดรส (Network Address Translation)มี 3 built-in chain คือ PREROUTING, POSTROUTING, OUTPUT ซึ่งเป็นตารางที่ใช้สำหรับทำ network address translation เช่น เปลี่ยนค่า source ip address, destination ip address จุดสำคัญอีกอย่างหนึ่งที่ต้องรู้ก็คือ มีเพียง packet แรกเท่านั้นที่เข้ามาที่ chain นี้ ส่วน packet ถัดไปนั้นจะถูกกระทำเหมือนที่ packet แรกได้รับ ดังนั้นจึงไม่ควรทำ packet filtering ที่ chain เหล่านี้

การใช้งาน Nat table นั้นก็เพียงแต่ใช้ออปชัน -t nat เท่านั้น และ target ที่สามารถใช้งานได้คือ SNAT, DNAT ซึ่งมีรายละเอียดดังนี้

**3.2.2.1 SNAT**

การทำ source NAT จะทำที่ POSTROUTING chain โดย หลักๆ คือทำการเปลี่ยน source address ก่อนที่จะส่ง packet นั้นออกไป ซึ่งสามารถใช้ออปชัน **-o** (outgoing interface) ร่วมด้วยได้ นอกจากนี้ยังใช้ **-j SNAT** และ **--to--source** หรือ **--to** เพื่อเปลี่ยนไอพีแอดเดรสหรือ port ไปตามต้องการได้ เช่น

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4
```

*รูปที่ 3-3 เปลี่ยน source ip address เป็น 1.2.3.4*

```
iptables -t nat -A POSTROUING -o eth0 -j SNAT --to 1.2.3.4-1.2.3.6
```

*รูปที่ 3-4 เปลี่ยน source address เป็น 1.2.3.4, 1.2.3.5 หรือ 1.2.3.6*

```
# iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to 1.2.3.4:1-1023
```

*รูปที่ 3-5 เปลี่ยน source address เป็น 1.2.3.4 port 1-1023*

**3.2.2.2 DNAT**

การทำ Destination NAT จะทำภายใต้ PREROUTING chain หลักๆ คือการเปลี่ยนค่า destination address หรือ port ก่อนที่จะส่ง packet ไปยัง routing decision โดยปกติการใช้งานจะระบุ **-j DNAT** และใช้ **--to-destination** หรือ **--to** และยังสามารถใช้ **-i** (incoming interface) ร่วมด้วยได้ เช่น

```
iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 192.168.1.20
```

*รูปที่ 3-6 เปลี่ยน destination address เป็น 192.168.1.20*

```
iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 192.168.1.20-192.168.1.22
```

*รูปที่ 3-7 เปลี่ยน destination address เป็น 192.168.1.20, 192.168.1.21 หรือ 192.168.1.22*

```
iptables -t nat -A PREROUING -p tcp --dport 80 -ieth0 -j DNAT --to 192.168.1.50:80
```

*รูปที่ 3-8 เปลี่ยน destination address ของ web traffic เป็น 192.168.1.50 port 8080*

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

**3.3 Command**

- -A เพิ่ม rule ใหม่ต่อท้าย chain (Append rule) เช่น
  **# iptables -A INPUT -p ALL -i eth0 -j ACCEPT**

- -D ลบ rule (Delete rule) เช่น
  **# iptables -D INPUT --dport 80 -j DROP**

- -I เพิ่ม rule ใหม่ ใน chain (Insert rule) เช่น
  **# iptables -I OUTPUT -p ALL -s 127.0.0.1/32 -j ACCEPT**

- -R แทนที่ rule เดิม ด้วย rule ใหม่ (Replace rule)

- -L แสดง rule ทั้งหมดใน chain (ถ้าไม่ระบุ chain จะแสดง rule ทั้งหมดใน filter table ทั้งสาม built-in chain) เช่น
  **# iptables -L**

**# iptables -L -t nat**

**# iptables -L INPUT**

- -F ลบ rule ทั้งหมดใน chain ทิ้ง เช่น

  **# iptables -F INPUT**

  **# iptables -F mychain**

- -Z ใช้ reset byte counter สำหรับทุก rule ใน chain ที่กำหนด เช่น

  **# iptables -Z INPUT**

- -N ใช้สร้าง chain ใหม่ เช่น

  **# iptables -N mychain**

- -X ลบ chain ที่ไม่มี rule ซึ่งสามารถลบ user-defined chain ที่ไม่มี rule ได้ แต่ไม่สามารถลบ built-in chain ได้ เช่น

  **# iptables -X emptychain**

- -P เปลี่ยน default policy ของ chain ค่าที่ใช้ได้คือ ACCEPT, DROP ทั้งนี้ค่านี้มีความสำคัญอย่างมากเพราะหาก packet ถูกส่งเข้ามาใน chain แล้ว และไม่ match กับ rule ใดๆ เลย packet นั้นก็ต้องถูกตัดสินใจโดย policy ของ chain นั้นๆ เช่น

  **# iptables -P FORWARD DROP**

  ซึ่งหาก packet ถูกส่งเข้ามายัง FORWARD chain และไม่ match กับ rule ใดๆ ใน FORWARD chain นี้เลย มันก็จะถูก DROP ทันที

- -E ใช้เปลี่ยนชื่อ chain ใหม่ เช่น

  **# iptables -E myoldchain mynewchain**

**3.4 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

**3.5 การระบุ target**

เมื่อมี packet ที่ match กับ rule แล้ว ต้องกำหนด target สำหรับ packet ไว้ด้วย โดยปกติจะใช้กัน 2 target คือ DROP และ ACCEPT

จะเห็นได้ว่าคำสั่งในการเซตไฟร์วอลล์โดยใช้ Iptables มีมากมาย ทำให้เกิดความยุ่งยากในการเขียนคำสั่งต่างๆให้ทำงานได้ตามต้องการ ดังนั้นแนวทางแก้ปัญหาก็คือ การสร้างกราฟฟิกยูสเซอร์อินเตอร์เฟสขึ้นมาเพื่อให้การสร้างคำสั่งต่างๆเพื่อใช้ในการสร้างไฟร์วอลล์ทำให้รวดเร็วและง่ายขึ้น

# บทที่ 4

# Squid

Squid มีคุณสมบัติเป็น Web proxy cache ออกแบบมาเพื่อ run บน Unix systems
คำว่า Proxy จะเป็นเหมือนตัวแทน ที่มีอำนาจหน้าที่แทนคนอื่น เพื่อไปเอาข้อมูลมา ทุกอย่างที่ไป proxy มา เก็บลงใน cache จะเรียกว่า objects ส่วน cache จะเป็นที่เอาไว้เก็บข้อมูลที่ proxy ไปหามา แต่จะไม่เหมาะกับ web site ที่เป็นข้อมูลแบบ dynamic เพราะจะต้องการผลลัพธ์ที่ update อยู่เสมอ

Squid จะทำตัวเป็นตัวแทน ที่ทำการรับ request มาจาก client(เช่น browser) และส่งต่อไปยัง Internet และเมื่อมีข้อมูลส่งกลับมาก็จะเก็บไว้บน disk cache

ประโยชน์ของ squid คือ เมื่อมีการ request ข้อมูลที่ซ้ำๆกันเกิดขึ้น แล้วจะส่งข้อมูลบน disk cache กลับไปที่ client แทน ทำให้เร็วในการ access Internet และ ลด bandwidth ที่ต้องใช้

squid proxy จะแตกต่างจาก firewall proxy เพราะ firewall proxy ส่วนใหญ่จะไม่เก็บข้อมูลต่างๆ เอาไว้ และเมื่อมีการ request เข้ามาก็จะไป fetch ข้อมูลมาทุกครั้ง

ใน cache จะทำการเก็บ object ต่างๆเอาไว้ เพื่อหลีกเลี่ยงที่ object ที่เก็บจะเป็นข้อมูลเก่า squid

## 4.1 หลักการทำงานของ Proxy - Caching Server

1. proxy-caching จะทำงานเมื่อมีการ requests มาจาก users เพื่อจะเข้าไปที่ web page ก็จะเข้าไปหาที่ cache ก่อน ถ้ามี web page นั้นอยู่ใน cache แล้วก็จะนำ web page นั้นreturn กลับไป แต่ถ้าไม่มีใน cache ก็จะไปหาที่ original site ซึ่งการเข้าไปหาที่ proxy-caching ก่อนจะทำให้การ access Internet ได้เร็วขึ้น เพราะว่าทำการ access ไปยัง original site โดยตรงจะทำให้เกิด traffic เยอะกว่า

2.ความสัมพันธ์ของ proxy-caching แบบ parent & siblingจะมีลักษณะ ดังนี้

- parent และ child จะมีลักษณะ คือถ้า server A เป็น parent และ server B เป็น child เมื่อมีการ request จาก B ไป A ถ้าที่ A มีข้อมูลก็จะส่ง ข้อมูลกลับไปให้ B แต่ถ้าที่ A ไม่มีข้อมูล A ก็จะทำการไปหาข้อมูลมา แล้วส่งให้ B

- sibling จะมีลักษณะ คือ ถ้า server A และ server B เป็น sibling กัน ถ้ามี clients request ข้อมูลมาที่ B แต่ B ไม่มีข้อมูลให้ ก็จะไปถามที่ A ก่อน ถ้า A มีก็จะทำการส่งข้อมูลนั้นให้กับ B แต่ถ้าไม่มี B ก็จะต้องไปหาข้อมูลมาเอง

ก็จะมีการตั้งค่า refresh ไว้ เพื่อจะได้ไม่ให้ข้อมูลเก่าๆ ส่งไปให้ user

### 4.1.1 การติดต่อระหว่าง Proxy - Caching Server

ระหว่าง cache ด้วยกันจะทำการติดต่อกันโดยใช้ protocal ที่เรียกว่า ICP(Internet Cache Protocal) เพื่อจะดูว่าจะใช้ cache จากที่ใด ในกรณีที่มีข้อมูลอยู่ทั้ง parent และ sibling ก็จะดูจากความเร็วใน response กลับมา แต่ถ้าไม่มีทั้ง 2 ที่เลย ก็จะส่ง request ข้อมูลไปยัง

parent cache เพื่อให้ไปหาต่อไป

**4.2 security**

**4.2.1 Access Control**

- squid ใช้ในการควบคุมว่าจะให้ใคร access web servers ได้บ้าง

ทำโดยใช้ ACL(Access Control List) สามารถแบ่งกลุ่มได้ว่า กลุ่มไหน deny กลุ่มไหน allow

**4.2.2 Encryption**

- squid สามารถ support encrypt protocal SSL ด้วยเพื่อจะได้ส่งไปโดยไม่มีใครอ่านได้

**4.3 Squid Version beta**

DEVEL ออกมาสำหรับผู้ใช้ Squid ที่คุ้นเคยกับ Squid ดีแล้ว ต้องการจะหา bug และปัญหาต่างๆ ใน DEVEL Version ที่ออกไป

PRE เป็น beta Version ที่ให้ผู้ใช้ลองก่อนที่จะออกเป็น version stable

STABLE เป็น production ที่ออกมา เมื่อมีคุณสมบัติใหม่ๆและมี bug ที่แก้ไขไปหมดแล้ว

**4.4 Squid Support**

**proxying and caching of HTTP, FTP**

Squid จะสามารถ proxy ผ่าน protocal ที่เป็น HTTP (HyperText Transfer Protocal) และ FTP (File Transfer Protocol)ได้ โดยที่ HTTP จะเป็นเอกสารต่างๆที่อยู่ในรูปแบบของ HTML ส่วน FTP จะเป็น file ต่างๆ

**proxying for SSL**

Squid จะสามารถส่งผ่าน protocal ที่เป็น SSL ได้ โดยจะใช้ได้ตั้งแต่ squid version 2.5 ขึ้นไป

Squid จะทำการสร้างช่องทางติดต่อโดยเฉพาะ โดยในการติดต้องจะส่งผ่านทาง https_port ที่จะกำหนดใน squid เมื่อต้องการจะติดต่อผ่านทาง port นี้ squid จะทำการเปิดช่องการในการติดต่อ SSL โดยตรงไปที่ original server และจะrequest ผ่านทาง CONNECT method ใน squid

โดย CONNECT method ใน squid จะเป็นทางที่ทำให้ทุกการติดต่อผ่าน HTTP proxy ได้จะทำการส่งไปและกลับระหว่าง client และ server

**Definition Of SSL**

SSL (Secure Socket Layer)จะเป็น protocal ที่ใช้ในการติดต่อข้อมูลที่ต้องการความปลอดภัยในการติดต่อผ่านทาง web โดยที่ SSL จะทำการ encrypt ข้อมูลต่างๆที่ต้องการส่งไป ทำการ authenticate กับ server ว่าสามารถจะติดต่อผ่านได้หรือไม่ มี integrity เกี่ยวกับ message ที่ส่งไป และการ authenticate ของ client ในการติดต่อผ่าน TCP/IP protocol

**4.5 Cache Hierarchies**

เป็นการที่จะเชื่อมต่อ cache เข้าด้วยกันทำให้ Squid สามารถติดต่อกับ cache อื่นๆใน Internet ได้ ทำให้แบ่งข้อมูลต่างๆ ระหว่าง cache ได้ และจะทำให้ประสิทธิภาพของ cache ดีขึ้นอีกด้วย จะแบ่ง cache ออกเป็น 2 ประเภทคือ parent และ sibling จะต้องมี protocal ระหว่าง cache ที่ใช้ในการติดต่อ คือ ICP (Inter-Cache Protocol),Cache-Digests,HTCP (Hyper-Text Cache Protocol) และ CARP (Cache Array Routing Protocol)

ICP (Internet Cache Protocol) จะเป็น protocol ที่ใช้ในการติดต่อระหว่าง cache ต่างๆ โดยจะดูว่ามีข้อมูลที่ client ต้องการอยู่หรือไม่ ถ้าไม่มีก็จะส่ง ICP ไปยัง cache ถ้ามีก็จะส่ง HIT กลับมา แต่ถ้าไม่มีก็จะส่ง MISS กลับมา ในการติดต่อระหว่าง cache ก็จะติดต่อระหว่าง icp_port ที่กำหนดไว้ใน squid

HTCP(Hyper Text Caching Protocol) จะเป็น protocol เหมือนกับ ICP ไม่ค่อยใช้แล้ว

CARP(Cache Array Routing Protocol) เป็น protocol ที่ใช้ hash function ในการตัดสินใจว่าจะไปเอา request ที่เข้ามาส่งไปที่ cache ไหน

Cache Digests ใช้ในการรับ index ของ object ต่างๆจาก cache อื่นๆ เมื่อที่ cache ไม่มีข้อมูลที่ user request มาจะไปดูที่ index ว่า cache ไหนมี object นั้นอยู่บ้าง แล้วติดต่อไปยัง cache นั้น ทำให้ไม่ต้องเสียพื้นที่มากในการเก็บ object

**4.6 ตัวอย่างไฟล์คอนฟิกใน Squid**

ไฟล์คอนฟิกของ Squid มีชื่อว่า Squid.conf จะอยู่ที่ /usr/local/squid/etc/squid.conf ในกรณีที่ลงโปรแกรมแบบไบนารีซอร์ส (Binary Source) และจะอยู่ที่ /etc/squid/squid.conf ในกรณีที่ลงโปรแกรมแบบอาร์พีเอ็ม (RPM)

**ตัวอย่างไฟล์คอนฟิกของ Squid**

```
# TAG: maximum_object_size_in_memory (bytes)
#       Objects greater than this size will not be attempted to kept in
#       the memory cache. This should be set high enough to keep objects
#       accessed frequently in memory to improve performance whilst low
#       enough to keep larger objects from hoarding cache_mem .
#
#Default:
# maximum_object_size_in_memory 8 KB


# TAG: ipcache_size     (number of entries)
# TAG: ipcache_low      (percent)
```

```
# TAG: ipcache_high       (percent)
#         The size, low-, and high-water marks for the IP cache.
#
#Default:
# ipcache_size 1024
# ipcache_low 90
# ipcache_high 95


# TAG: fqdncache_size     (number of entries)
#         Maximum number of FQDN cache entries.
#
#Default:
# fqdncache_size 1024


# TAG: cache_replacement_policy
#         The cache replacement policy parameter determines which
#         objects are evicted (replaced) when disk space is needed.
```

จากตัวอย่าง จะเห็นได้ว่าคำสั่งในการเซตคอนฟิกต่างๆในไฟล์ Squid.conf มีมากมาย ทำให้เกิดความยุ่งยากในการเซตค่าคอนฟิกต่างๆ ดังนั้นแนวทางแก้ปัญหาก็คือ การสร้างกราฟฟิกยูสเซอร์อินเตอร์เฟสขึ้นมาเพื่อให้การเซตคอนฟิกต่างๆทำให้รวดเร็วและง่ายขึ้น

# บทที่ 5

# SNORT

SNORT (Roesch, M. : 1999.) คือโปรแกรมตรวจจับผู้บุกรุกทางเครือข่ายที่มีการใช้งานอย่างแพร่หลาย เนื่องจากเป็นระบบตรวจจับผู้บุกรุกที่ใช้งานง่าย แต่มีประสิทธิภาพสูง สามารถตรวจจับการบุกรุกทางเครือข่ายได้หลากหลาย โดยผู้ใช้งานเพียงแค่สร้างกฎขึ้นมาให้มีรูปแบบของกฎตามที่โปรแกรม SNORT ได้ให้ไว้ ตัวแปรในกฎมีความหลากหลายทำให้สามารถสร้างกฎที่ซับซ้อนได้ อีกทั้งยังมีการแบ่งปันกฎที่ใช้ในการตรวจสอบการบุกรุกแบบต่างๆในอินเตอร์เน็ต ผู้ใช้งานสามารถดาวน์โหลดมาใช้งานได้ทันที นอกจากนี้ในส่วนของการพัฒนาตัวโปรแกรมนั้นก็มีการพัฒนาอย่างรวดเร็วเพราะเป็นโปรแกรมที่เปิดเผยซอร์สโค้ดทำให้มีผู้ร่วมพัฒนามากมาย

## 5.1 โครงสร้างของ SNORT

การออกแบบโปรแกรม SNORT นั้นออกแบบโดยคำนึงถึงประสิทธิภาพในการทำงานและการใช้งานง่ายเป็นหลัก โดยมีส่วนการทำงานหลักๆอยู่สามส่วนด้วยกันคือ ส่วนของการแปลความหมายข้อมูลแพ็กเก็ต(packet decoder), ส่วนของการตรวจสอบกฎ (detection engine) และการบันทึกผลการทำงานและแจ้งเตือน(logging and alerting subsystem) โดยมีการทำงานร่วมกับไลบรารี่ Libcap ซึ่งเป็นไลบรารี่ในการอ่านข้อมูลของแพ็กเก็ตที่ผ่านไปมาในเครือข่าย โดยการทำงานแต่ละส่วนจะมีรายละเอียดดังนี้

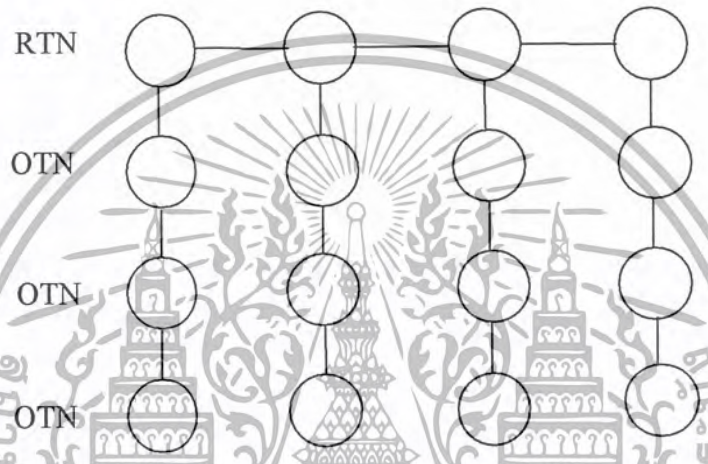
### 5.1.1 ส่วนของการแปลความหมายข้อมูลแพ็กเก็ต (Packet Decoder)

ในส่วนของการแปลความหมายแพ็กเก็ตนั้นจะทำงานอยู่บริเวณโพรโตคอลสแตก ในชั้นดาตาลิงค์เลเยอร์ (Datalink Layer) และทีซีพี/ไอพี (TCP/IP) ซึ่งจะถูกแบ่งเป็นโปรแกรมย่อยต่างๆ แต่ละโปรแกรมย่อยของการแปลความหมายแพ็กเก็ตก็ทำงานโดยการนำเอาข้อมูลของแพ็กเก็ต มาเทียบค่ากับโครงสร้างข้อมูลของแพ็กเก็ตแบบต่างๆ ส่วนของโปรแกรมที่แปลความหมายแพ็กเก็ตจะถูกเรียกให้ทำงานเมื่อโปรแกรม SNORT ได้รับแพ็กเก็ตจากเครือข่าย โดยจะแปลความหมายของโพรโตคอลตั้งแต่ชั้นดาตาลิงค์ , ชั้นเน็ตเวิร์ค , ชั้นทรานสพอร์ต ขึ้นไปเรื่อยๆจนถึงชั้นแอปพลิเคชัน

การทำงานในส่วนนี้ต้องทำงานด้วยความเร็วสูงมาก ซึ่งโปรแกรม SNORT ใช้วิธีการสร้างพอยเตอร์หลายๆตัว เพื่อชี้ในส่วนต้นของเฮดเดอร์ ในโพรโตคอลชั้นต่างๆ ทำให้การแปลความหมายข้อมูลทำได้อย่างรวเร็ว สำหรับการตีความแพ็กเก็ตนั้น โปรแกรม SNORT สามารถตีความแพ็กเก็ตในโพรโตคอลได้หลายๆตัว เช่น Ethernet , SLIP , raw (PPP) datalink protocol และโพรโตคอลอื่นๆที่มีการใช้งานกันอย่างแพร่หลาย

**5.1.2 ส่วนของการตรวจสอบกฎ(detection engine)**

สำหรับโครงสร้างที่เป็นลิงค์ลิสท์ 2 มิตินั้นประกอบด้วยโหนด 2 รูปแบบด้วยกันคือ Rule Tree Node (RTN) และ Option Tree Node (OTN) ดังรูปที่ 5.1 โดย RTN จะเป็นโหนดที่บรรจุส่วนของกฎที่เป็นรูปแบบการบุกรุกหลายๆแบบจะต้องมี เช่น ไอพีแอดเดรสต้นทาง , ไอพีแอดเดรสปลายทาง , พอร์ตต้นทาง , พอร์ตปลายทางและโพรโตคอลที่ใช้ในแพ็กเก็ตนั้นๆ เช่น ทีซีพี (TCP) , ยูดีพี (UDP) หรือไอซีเอ็มพี (ICMP) เป็นต้น ส่วน OTN ประกอบด้วยส่วนเฉพาะของกฎ ที่เป็นส่วนที่แตกต่างกันในการโจมตีแต่ละแบบเช่น โค้ดของไอซีเอ็มพีแพ็กเก็ต (ICMP Packet) , ทีซีพีแฟล็ก (TCP Flag) และข้อมูลในแพ็กเก็ตที่ใช้บุกรุก เป็นต้น




*รูปที่ 5-1 โครงสร้างข้อมูลแบบลิงค์ลิสต์สองมิติ*

สำหรับจุดประสงค์ในการสร้างโครงสร้างกฎในลักษณะนี้คือหลีกเลี่ยงการตรวจสอบกฎทุกๆกฎ โดยพยายามแบ่งกลุ่มของกฎออกเป็นกลุ่มๆแทน ถ้าตรวจสอบข้อมูลในแพ็กเก็ตกับ RTN แล้วไม่เหมือนก็ไม่จำเป็นต้องตรวจสอบกฎกับ OTN ที่ติดกับ RTN นั้นอีก สำหรับ SNORT ในกรณีที่ตรวจสอบกับเงื่อนไขใน RTN แล้วไม่เหมือน ก็จะไปตรวจสอบกับข้อมูลใน RTN อื่นๆต่อไปจนหมด

ในกรณีที่ตรวจสอบกับข้อมูลใน RTN แต่ละส่วนแล้วตรงกับเงื่อนไข จะต้องตรวจสอบกับเงื่อนไขใน OTN อีกครั้งหนึ่ง ในส่วนของการเปรียบเทียบ content ของกฎนั้น โปรแกรม SNORT ใช้อัลกอริทึมในการตรวจสอบกฎโดยใช้อัลกอริทึมของบอยเออร์-มัวร์ ถ้าตรวจสอบใน OTN แล้วไม่พบความผิดปกติ ก็จะตรวจสอบแพ็กเก็ตดังกล่าวกับ OTN ถัดไปจนหมดรายการของ OTN กระบวนการดังกล่าวเป็นกระบวนการที่เสียเวลามาก ถึงแม้ว่าบอยเออร์-มัวร์จะช่วยให้การค้นหาและตรวจสอบทำได้เร็วขึ้นก็ตามเพราะการทำงานในการตรวจสอบกฎก็ต้องทำทีละกฎไปเรื่อยๆจนครบทุกกฎ และทุกครั้งของการตรวจสอบจะต้องอ่านแพ็กเก็ตข้อมูลจนหมดด้วย

**5.1.3 การบันทึกผลการทำงานและแจ้งเตือน (Logging and alerting subsystem)**

ส่วนการบันทึกผลการทำงานและแจ้งเตือนต่อผู้ดูแลระบบ ผู้ใช้งานโปรแกรม SNORT สามารถเลือกได้จากการป้อนพารามิเตอร์ ขณะรันโปรแกรม SNORT ในการทำงานในส่วนนี้ สามารถเลือกการบันทึกผลการทำงานได้ 3 รูปแบบและสามารถเลือกการแจ้งเตือนได้ 5 รูปแบบ สำหรับการทำงานอื่นๆที่เป็นตัวเลือกในการเก็บข้อมูลแพ็กเก็ต สามารถเลือกได้ว่าจะเก็บข้อมูลไว้ในรูปแบบอย่างไร โดยสามารถเลือกรูปแบบได้เป็น แพ็กเก็ตที่ถูกแปลความหมายแล้ว , แพ็กเก็ตที่อยู่ในรูปแบบที่สามารถอ่านค่าต่างๆได้ (human readable) หรืออยู่ในรูปแบบของโปรแกรม tcpdump (tcpdump binary format) เก็บไว้ในไฟล์ข้อมูลเพียงไฟล์เดียว

โดยการบันทึกแพ็กเก็ตที่ถูกแปลความหมายแล้วจะสามารถนำเอาข้อมูลไปวิเคราะห์เร็วกว่าการเก็บข้อมูลที่ยังไม่ได้แปลความหมาย ส่วนการเก็บข้อมูลในรูปแบบของ tcpdump จะสามารถเก็บข้อมูลได้เร็วกว่า ซึ่งเหมาะกับการใช้งานในกรณีที่ต้องการประสิทธิภาพการทำงานที่สูงกว่า ในบางครั้งการบันทึกการทำงานนั้นอาจไม่จำเป็นต้องทำ โดยผู้ดูแลระบบจะใช้เพียงการแจ้งเตือนเท่านั้นเพื่อให้โปรแกรมทำงานน้อยลง และได้ความเร็วในการทำงานเพิ่มมากขึ้น

สำหรับการแจ้งเตือนผู้ดูแลระบบ โปรแกรม SNORT สามารถส่งผลลัพธ์ไปยังระบบการเก็บข้อมูลการทำงานของระบบปฏิบัติการหรือ syslog ได้ โดยการทำงานนี้สามารถบันทึกผลลัพธ์ได้ในลักษณะของเท็กไฟล์หรือสามารถส่งการแจ้งเตือนในลักษณะ Winpopup โดยใช้โปรแกรม Samba Client ส่งการแจ้งเตือนไปยังหน้าต่างของ Microsoft Windows การส่งข้อมูลเพื่อแจ้งเตือนผู้ดูแลระบบโดยส่งไปเก็บไว้ในเท็กซ์ไฟล์นั้นสามารถทำได้ 2 รูปแบบคือแบบเต็มรูปแบบ (Full Alert) และแบบแจ้งเตือนแบบรวดเร็ว (Fast Alert)

ในการแจ้งเตือนแบบเต็มรูปแบบจะเขียนข้อมูลที่แจ้งเตือนและข้อมูลตั้งแต่เฮดเดอร์ของไอพีแพ็กเก็ตจนถึงข้อมูลในชั้นทรานส์พอร์ตไว้ในเท็กไฟล์ ส่วนการแจ้งเตือนแบบรวดเร็วจะเขียนข้อมูลเพียงบางส่วนของเฮดเดอร์เก็บไว้ในไฟล์ เพื่อให้การทำงานรวดเร็วขึ้น ส่วนการทำงานรูปแบบสุดท้ายที่สามารถเลือกได้ในการแจ้งเตือนนั้นคือการ disable alerting ใช้ในกรณีที่ไม่ต้องการให้มีการแจ้งเตือนใดๆ เกิดขึ้น เช่นในกรณีที่ทำการทดสอบเครือข่าย เป็นต้น

**5.2 การทำงานของ SNORT**

การทำงานของโปรแกรม SNORT สามารถแบ่งเป็น 3 ขั้นตอนหลักๆ คือ ส่วนการทำงานก่อนการตรวจจับ , ส่วนการทำงานขณะทำการตรวจจับ และการทำงานหลังการตรวจจับ

**5.2.1 การทำงานก่อนการตรวจจับ**

การทำงานก่อนการตรวจจับ จะเป็นกระบวนการเตรียมสิ่งแวดล้อมให้พร้อมต่อการทำงาน โดยสิ่งที่ต้องเตรียมคือโครงสร้างของกฎที่จำเป็นต้องใช้ในการตรวจสอบ กระบวนการนี้เริ่มต้นเมื่อมีการรันโปรแกรม SNORT โปรแกรมจะอ่านค่าพารามิเตอร์ต่างๆ เช่น อ่านไฟล์ที่เป็นกฎจากไฟล์ใด อ่านแพ็กเก็ต

จากไฟล์หรือจากเครือข่าย ให้แสดงผลแบบใด ทำงานเป็นเดมอนหรือไม่ ฯลฯ โดยเฉพาะไฟล์ของกฎที่จะให้ตรวจสอบนั้น จะต้องถูกอ่านในกระบวนการนี้เมื่อ SNORT อ่านไฟล์ของกฎแล้ว จะสร้างโครงสร้างข้อมูลของกฎเพื่อใช้ในการวิเคราะห์ข้อมูล โดยโครงสร้างข้อมูลของกฎมีลักษณะเป็นลิงค์ลิสต์ 2 มิติ แล้วจึงเปลี่ยนโหมดของเน็ตเวิร์คการ์ดให้เป็น promiscuous mode เพื่อใช้ในการอ่านข้อมูลในแพ็กเก็ตที่ผ่านไปมาในเครือข่าย

**5.2.2 ส่วนการทำงานขณะทำการตรวจจับ**

การทำงานในส่วนของการตรวจจับนั้นเริ่มจาก โปรแกรม SNORT รับแพ็กเก็ตข้อมูลจากเครือข่าย ซึ่งใช้ฟังก์ชันในไลบรารี่ Libpcap เพื่อดึงข้อมูลจากเครือข่าย หลังจากนั้นโปรแกรมจะตีความส่วนต่างๆของข้อมูลที่ได้รับเข้ามาว่าเป็นข้อมูลของ โพรโตคอลใดประกอบกันบ้าง แล้วจึงนำมาเปรียบเทียบกับกฎซึ่งได้สร้างเป็นโครงสร้างข้อมูลแบบลิงค์ลิสต์ 2 มิติไว้แล้วในส่วนการทำงานแรก เมื่อเปรียบเทียบกฎต่างๆเรียบร้อยแล้ว ถ้าแพ็กเก็ตไหนที่ตรงตามกฎ จะตอบสนองตามที่กฎได้ตั้งค่าไว้ เช่น แจ้งเตือนผู้ดูแลระบบหรือบันทึกการทำงานในการตรวจจับ ซึ่งการทำงานของโปรแกรมจะทำงานไปเรื่อยๆจนกว่าจะมีการส่งสัญญาณต่างๆไปหยุดโพรเซส เช่น SIGHUB หรือ SIGKILL เป็นต้น

**5.2.3 ส่วนการทำงานหลังการตรวจจับเสร็จสิ้น**

ในส่วนการทำงานนี้จะเป็นการสรุปผลการรับข้อมูลว่าข้อมูลที่ได้รับนั้นเป็นแพ็กเก็ตข้อมูลโพรโตคอลอะไรบ้าง เป็นจำนวนเท่าไหร่ และคิดเป็นกี่เปอร์เซ็นต์ของแพ็กเก็ตที่รับเข้ามาทั้งหมดดังตัวอย่างในรูปที่ 2

**Breakdown by protocol:**

TCP: 25 (18.248%)

UDP (20.438%)

ICMP:6 (4.380%)

FRAGS:0 (0.000%)

ARP:61 (44.526%)

IPv6:0 (0.000%)

IPX:0 (0.000%)

OTHER:17 (12.409%)

***รูปที่ 5-2 ตัวอย่างการทำงานของโปรแกรม SNORT เมื่อสิ้นสุดการตรวจจับ***

จากรูปตัวอย่างหมายถึง ตลอดระยะเวลาทำงานของโปรแกรม SNORT ได้รับแพ็กเก็ต TCP 25 แพ็กเก็ต คิดเป็น 18.248% เปอร์เซ็นต์ของทั้งหมด แพ็กเก็ต UDP 28 แพ็กเก็ต คิดเป็น 20.438 เปอร์เซ็นต์ของทั้งหมด แพ็กเก็ต ICMP 6 แพ็กเก็ตคิดเป็น 4.380 เปอร์เซ็นต์ของทั้งหมด แพ็กเก็ต ARP 61 แพ็กเก็ต

คิดเป็น 44.526 เปอร์เซ็นต์ของทั้งหมด แพ็กเก็ตที่ใช้โพรโตคอลอื่นๆอีก 17 แพ็กเก็ตคิดเป็น 12.409 เปอร์เซ็นต์ของแพ็กเก็ตทั้งหมด ในการทำงานครั้งนี้ไม่มีแพ็กเก็ตที่เป็น IPV6, IPX และแพ็กเก็ตที่เป็นแฟรกเมนต์เตชั่น (fragmentation)

**5.3 กฎและโครงสร้างของกฎที่ใช้ใน SNORT**

กฎในโปรแกรม SNORT นั้นคือประโยคที่เป็นเงื่อนไขในการทำงานของโปรแกรม SNORT โดยจะเป็นประโยคที่บอกว่าแพ็กเก็ตลักษณะไหนที่เป็นแพ็กเก็ตผิดปกติ แล้วโปรแกรม SNORT จะต้องทำงานอย่างไรกับกฎนั้น ในช่วงการทำงานของโปรแกรม SNORT กฎจะถูกอ่านในช่วงของการทำงานก่อนการตรวจจับ เพื่อสร้างโครงสร้างของกฎในรูปลิงค์ลิส 2 มิติแล้วนำไปใช้ในช่วงการทำงานขณะทำการตรวจจับ

สำหรับกฎที่ SNORT ใช้นั้นมีไวยกรณ์ของกฎประกอบด้วย ส่วนที่อธิบายการตอบสนองของ SNORT ต่อแพ็กเก็ตผิดปกติ , ส่วนเงื่อนไขของเฮดเดอร์ของข้อมูลและพอร์ต และส่วน Option field ยกตัวอย่างในรูปที่ 5.3

```
Alert tcp any any -> any 80 (msg:"IIS-cmd?";flags:PA;content:".cmd?&";nocase;)
```

*รูปที่ 5-3 ตัวอย่างกฎของ SNORT*

จากรูปกฎตัวอย่างหมายถึงโปรแกรมจะแจ้งเตือนต่อผู้ดูแลระบบถ้าแพ็กเก็ตที่รับเข้ามาเป็นแพ็กเก็ตที่ทำงานกับโพรโตคอลทีซีพี โดยส่งจากไอพีและพอร์ตใดๆไปยังไอพีปลายทางใดๆที่พอร์ต 80 โดยในทีซีพีเฮดเดอร์จะเซตแฟลก PSH และ ACK และในแพ็กเก็ตมีคำว่า ".cmd?&" อยู่ในแพ็กเก็ต ในการตรวจสอบข้อมูลในแพ็กเก็ตจะไม่คำนึงถึงนัยสำคัญของตัวอักษรตัวพิมพ์เล็กและตัวพิมพ์ใหญ่

โดยรายละเอียดของกฎ ในส่วนของการตอบสนองของโปรแกรม SNORT ในกรณีที่ข้อมูลในแพ็กเก็ตตรงตามกฎ จะมีการตอบสนอง 3 รูปแบบด้วยกันคือ แจ้งเตือนผู้ดูแลระบบ (alert) , เก็บข้อมูลแพ็กเก็ต (log) และไม่สนใจแพ็กเก็ตนั้น (pass) ในส่วนถัดมาเป็นส่วนเงื่อนไขของเฮดเดอร์ข้อมูล สามารถตั้งค่าได้สามส่วนคือไอพีแอดเดรส , พอร์ต และ direction operator

1. **ไอพีแอดเดรส**

ในส่วนนี้จะเป็นรายละเอียดเกี่ยวกับหมายเลขไอพีแอดเดรสที่อยู่ในแพ็กเก็ต โดยมีค่าพิเศษคือ any ซึ่งจะหมายถึงไอพีแอดเดรสใดๆ ในส่วนนี้โปรแกรม SNORT จะไม่สามารถป้อนค่าเป็นชื่อโฮสได้ เนื่องจากไม่มีกระบวนการในการเปลี่ยนชื่อโฮสให้กลายเป็นไอพีแอดเดรส สำหรับค่าหมายเลขไอพีแอดเดรสที่ใช้ได้นั้น สามารถเป็นหมายเลขไอพีแอดเดรสหรือหมายเลขไอพีแอดเดรสตามด้วย CIDR block ก็ได้ โดย CIDR block จะเป็นการบอก netmask ของแพ็กเก็ต โดย CIDR block/24 จะหมายถึง

เครือข่ายคลาสซี , /16 หมายถึงเครือข่ายคลาสบี และ /32 จะหมายถึงหมายเลขแอดเดรสของเครื่องนั้นๆ ยกตัวอย่างเช่น 192.168.1.0/24 จะหมายถึงช่วงของแอดเดรสตั้งแต่ 192.168.1.1 ถึง 192.168.1.255

นอกจากนี้ยังมีโอเปอร์เรเตอร์อีกตัวหนึ่งที่ใช้งานควบคู่กับหมายเลขไอพีแอดเดรสได้คือ Negation "!" โดยโอเปอร์เรเตอร์ดังกล่าวจะหมายถึงให้โปรแกรม SNORT ตรวจสอบหมายเลขไอพีแอดเดรสทุกๆหมายเลขยกเว้นหมายเลขไอพีแอดเดรสนั้น ยกตัวอย่างเช่น การตั้งกฎเพื่อให้โปรแกรม SNORT แจ้งเตือนเมื่อมีข้อมูลจากภายนอกส่งเข้ามาในเครือข่าย สามารถทำได้โดยตั้งกฎดังตัวอย่างรูปที่ 4

```
alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 any …
```

*รูปที่ 5-4 ตัวอย่างกฎที่ใช้ Negation กับไอพีแอดเดรส*

จากตัวอย่างมีความหมายว่าให้โปรแกรม SNORT แจ้งเตือนเมื่อมีแพ็กเก็ตที่เป็นโพรโตคอล TCP โดยมีหมายเลขไอพีต้นทางไม่ได้อยู่ในเครือข่าย แต่หมายเลขไอพีปลายทางอยู่ในเครือข่าย

2. พอร์ต

จะเป็นรายละเอียดเกี่ยวกับหมายเลขพอร์ตที่ต้องการดักจับ ซึ่งในกรณีที่เป็นพอร์ตเดี่ยวๆสามารถใช้เป็นตัวเลขได้ หรือสามารถดักจับในช่วงโดยใช้เครื่องหมาย ":" คั่นระหว่างช่วงพอร์ตเช่น 1:1024 และในกรณีที่ต้องการดักทุกๆพอร์ตจะให้กำว่า "any" ยกตัวอย่างในรูปที่ 5 เป็นตัวอย่างการตั้งกฎโดยในกฎแรกจะให้โปรแกรม SNORT บันทึก UDP แพ็กเก็ตที่มาจากทุกๆไอพีและทุกๆพอร์ต และมีแอดเดรสปลายทางอยู่ในเครือข่าย 192.168.1.0/24 ที่พอร์ตน้อยกว่าหรือเท่ากับ 6000 ส่วนกฎที่สามเป็นกฎที่ให้โปรแกรม SNORT บันทึก TCP แพ็กเก็ตที่ส่งมาจากไอพีแอดเดรสใดๆที่พอร์ตน้อยกว่าหรือเท่ากับ 1024 ส่งมายังเครื่องในเครือข่าย 192.168.1.0/24 ที่พอร์ตมากกว่าหรือเท่ากับ 500

```
log udp any any -> 192.168.1.0/24 1:1024
log tcp any any -> 192.168.1.0/24 :6000
log tcp any :1024 -> 192.168.1.0/24 500:
```

*รูปที่ 5-5 ตัวอย่างการใช้กฎที่มีการใช้พอร์ตรูปแบบต่างๆ*

นอกจากนี้ยังสามารถใช้ Negation "!" เพื่อให้มีความหมายตรงข้ามได้ เช่นในรูปที่ 6 ตัวอย่างการใช้กฎที่มีการใช้ Negation กับพอร์ต โดยกฎนี้จะให้โปรแกรม SNORT บันทึกแพ็กเก็ตทุกๆแพ็กเก็ตที่ส่งเข้ามาในเครือข่าย 192.168.1.0/24 ที่ไม่ใช่พอร์ตของ X Windows (พอร์ต 6000 ถึง 6010) แต่การใช้ Negation นี้จะไม่สามารถใช้กับ "any" ได้

```
log tcp any any -> 192.168.1.0/24 !6000:6010
```

รูปที่ 5.6 ตัวอย่างการใช้กฎที่มีการใช้ Negation กับพอร์ต

**3. direction operator**

Direction Operator หรือเครื่องหมาย "->" เป็นเครื่องหมายในการบอกทิศทางของข้อมูลที่กฎจะนำไปใช้ โดยหมายเลขไอพีแอดเดรสและหมายเลขพอร์ตที่อยู่ด้านซ้ายมือของเครื่องหมายเป็นการบอกที่มาของแพ็กเก็ตว่าต้นทางคือหมายเลขไอพีแอดเดรสอะไร และส่งมาจากพอร์ตไหน ส่วนหมายเลขไอพีแอดเดรสและหมายเลขพอร์ตที่อยู่ด้านขวามือของเครื่องหมายจะหมายถึงหมายเลขไอพีแอดเดรสและพอร์ตของเครื่องปลายทาง

สำหรับ Direction Operator นี้จะมีอีกรูปแบบหนึ่งคือ bi-directional operator ซึ่งเป็นเครื่องหมาย "<>" ซึ่งหมายถึงให้โปรแกรม SNORT พิจารณาคู่หมายเลขไอพีแอดเดรสและพอร์ต ไม่ว่าจะเป็นต้นทางหรือปลายทาง ซึ่งเหมาะกับการวิเคราะห์ข้อมูลทั้งไปและกลับ เช่น เซสชั่นของ telnet และ POP3 รูปที่ 7 เป็นตัวอย่างกฎที่ใช้ bi-directional operator โดยกฎนี้จะให้โปรแกรม SNORT บันทึกแพ็กเก็ตที่ส่งไปมาระหว่างภายนอกเครือข่าย 192.168.1.0/24 ที่พอร์ตใดๆกับในเครือข่าย 192.168.1.0/24 ที่พอร์ต 23 ซึ่งก็คือการเก็บข้อมูลการทำงานต่างๆในเซสชั่น telnet นั่นเอง

```
log !192.168.1.0/24 any <> 192.168.1.0/24 23
```

*รูปที่ 5-7 ตัวอย่างกฎที่ใช้ bi-directional operator*

ในส่วนสุดท้ายเป็นส่วนของ Option field ซึ่งเป็นส่วนของการตั้งกฎให้โปรแกรม SNORT ตรวจสอบค่าปลีกย่อยต่างๆในแพ็กเก็ต ซึ่งใน SNORT version 2.0.1 จะมี 41 option field คือ

1. msg
2. logto
3. ttl
4. tos
5. id
6. ipoption
7. Fragbits
8. Dsize
9. Flags
10. Seq

11. ACK
12. Window
13. Itype
14. Icode
15. Icmp_id
16. Icmp_sea
17. Content-list
18. Offset
19. Depth
20. nocase
21. Session
22. rpc
23. Resp
24. React
25. Reference
26. Reference
27. Sid
28. Rev
29. Classtype
30. priority
31. Uricontent
32. Tag
33. Ip Proto
34. Same IP
35. Flow
36. Fragoffset
37. Rawbytes
38. distance
39. Within
40. Byte_test
41. Byte_Jump

**5.4 ตัวอย่างกฎของโปรแกรม SNORT**

จากตัวอย่างกฎที่ใช้งานจริงในโปรแกรม SNORT นั้น เราจะเห็นได้ว่ามีการใช้งานหลายๆ รูปแบบเพื่อดักจับการบุกรุกหลายๆแบบได้ ดังตัวอย่าง

กฎที่ 1 : alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"DDOS TFN Probe"; id: 678; itype: 8; content: "1234";reference:arachnids,443; classtype:attempted-recon; sid:221; rev:1;)

กฎที่ 2 : alert tcp any any -> any 139 (msg:"Virus - Possible QAZ Worm Infection"; flags:A; content: "|71 61 7a 77 73 78 2e 68 73 71|"; reference:MCAFEE,98775; sid:732; classtype:misc-activity; rev:3;)

กฎที่ 3 : alert tcp $EXTERNAL_NET any -> $HOME_NET 110 (msg:"POP3 DELE overflow attempt"; flow:to_server,established; content:"DELE"; nocase; content:!"|0a|"; within:10; classtype:attempted-admin; sid:2111; rev:1;)

กฎที่ 4 : alert tcp $HOME_NET 23 -> $EXTERNAL_NET any (msg:"INFO TELNET Bad Login"; content: "Login failed"; nocase; flow:from_server,established; classtype:bad-unknown; sid:492; rev:6;)

กฎที่ 5 : alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306 (msg:"MYSQL show databases attempt"; flow:to_server,established; content:"|0f 00 00 00 03|show databases"; classtype:protocol-command-decode; sid:1776; rev:1;)

กฎที่ 6 : lert udp $EXTERNAL_NET any -> $HOME_NET 162 (msg:"SNMP trap udp"; reference:cve,CAN-2002-0012; reference:cve,CAN-2002-0013; sid:1419; rev:2; classtype:attempted-recon;)

กฎที่ 7 : alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC source route lssre"; ipopts:lsrre; reference:bugtraq,646; reference:cve,CVE-1999-0909; reference:arachnids,420; classtype:bad-unknown; sid:501; rev:2;)

กฎที่ 8 : alert tcp $HOME_NET any -> $EXTERNAL_NET 6881:6889 (msg:"P2P BitTorrent transfer"; flow:to_server,established; content:"|13|BitTorrent protocol"; offset:0; depth:20; classtype:policy-violation; sid:2181; rev:1;)

*รูปที่ 5-8 รูปตัวอย่างกฎใน Snort*

จะเห็นได้ว่าการสร้างกฎใน Snort จะค่อนข้างซับซ้อนและยุ่งยาก เนื่องจากออปชัน(Option) มีถึง 41 ออปชัน ดังนั้นแนวทางแก้ปัญหาคือ การสร้างกราฟฟิกยูสเซอร์อินเตอร์เฟสขึ้นมาเพื่อแก้ปัญหาตรงส่วนนี้ ทำให้ผู้ใช้สามารถสร้างกฎได้อย่างรวดเร็วและง่ายขึ้น

# บทที่ 6
# การวิเคราะห์และออกแบบระบบ

การวิเคราะห์และการออกแบบระบบ เป็นส่วนสำคัญอย่างยิ่งอีกส่วนหนึ่งในทำการโครงงาน การวิเคราะห์และการออกแบบที่ดี จะทำให้การสร้างระบบ (Implement) เป็นไปได้อย่างรวดเร็วและตรงตามความต้องการ ภาษาที่ใช้ช่วยในการวิเคราะห์และออกแบบระบบ คือ ภาษา UML ซึ่งเป็นภาษาที่ช่วยสร้างไดอะแกรม (Diagram) ต่างๆในการออกแบบระบบ ซึ่งทำให้การออกแบบระบบมีมาตรฐาน และง่ายต่อความเข้าใจของผู้พัฒนาระบบต่อๆไปอีกด้วย

## 6.1 การวิเคราะห์ความต้องการของระบบ

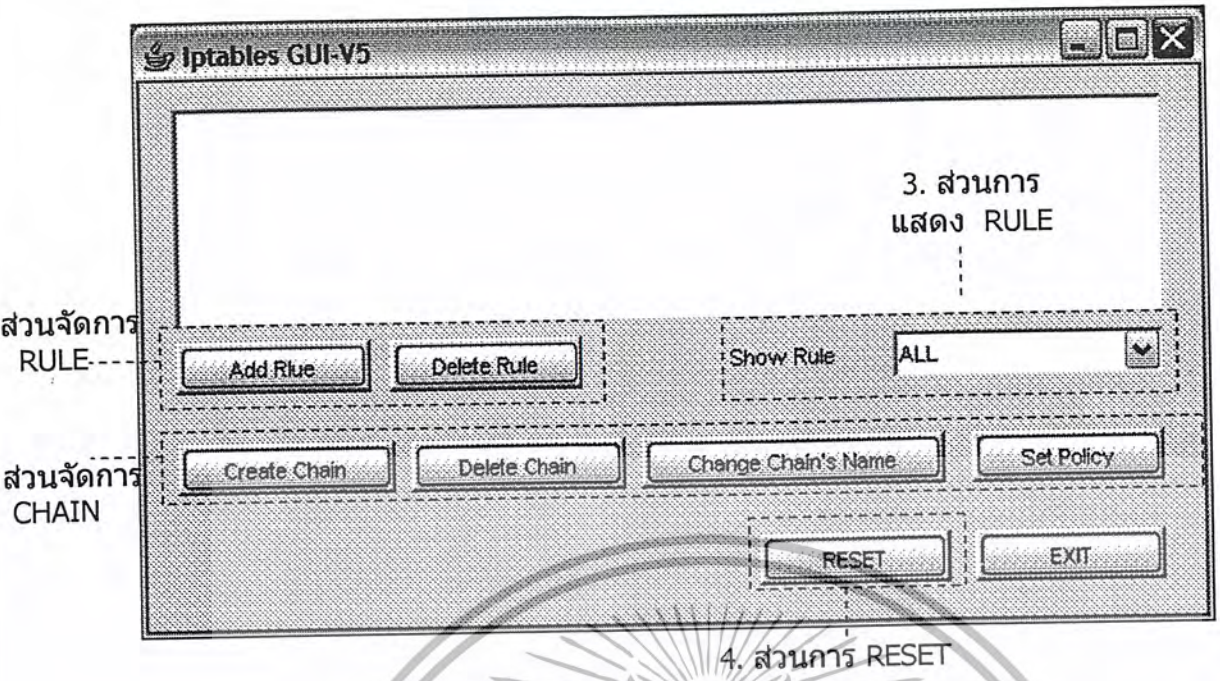
การวิเคราะห์ระบบ เป็นการรวบรวมรายละเอียด และความต้องการต่างๆที่ได้ศึกษามามาทำการรวบรวมและสรุปเป็นความต้องการของระบบ ซึ่งในขั้นตอนการวิเคราะห์นี้ จะใช้ยูสเคสไดอะแกรม (Use Case Diagram) เป็นไดอะแกรมแสดงความต้องการของระบบ ซึ่งเป็นหน้าที่การทำงานต่างๆที่ต้องมีในระบบที่ได้จากการศึกษาและวิเคราะห์มา

### 6.1.1 หลักการออกแบบ

การออกแบบกราฟฟิกยูสเซอร์อินเตอร์เฟส (Graphic User Interface) ซึ่งประกอบไปด้วย กราฟฟิกยูสเซอร์อินเตอร์เฟสของ Iptables , กราฟฟิกยูสเซอร์อินเตอร์เฟสของ Squid และกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Snort เป็นการนำเอาคำสั่งต่างๆที่ใช้ในการสร้างไฟร์วอลล์โดยใช้ Iptables , การสร้าง proxy โดยใช้ Squid และคำสั่งในการสร้างกฎและจัดการของ Snort มาสร้างเป็นกราฟฟิกยูสเซอร์อินเตอร์เฟสเพื่อให้ผู้ใช้สามารถใช้งานได้ง่ายขึ้น

#### 6.1.1.1 หลักการออกแบบ Iptables

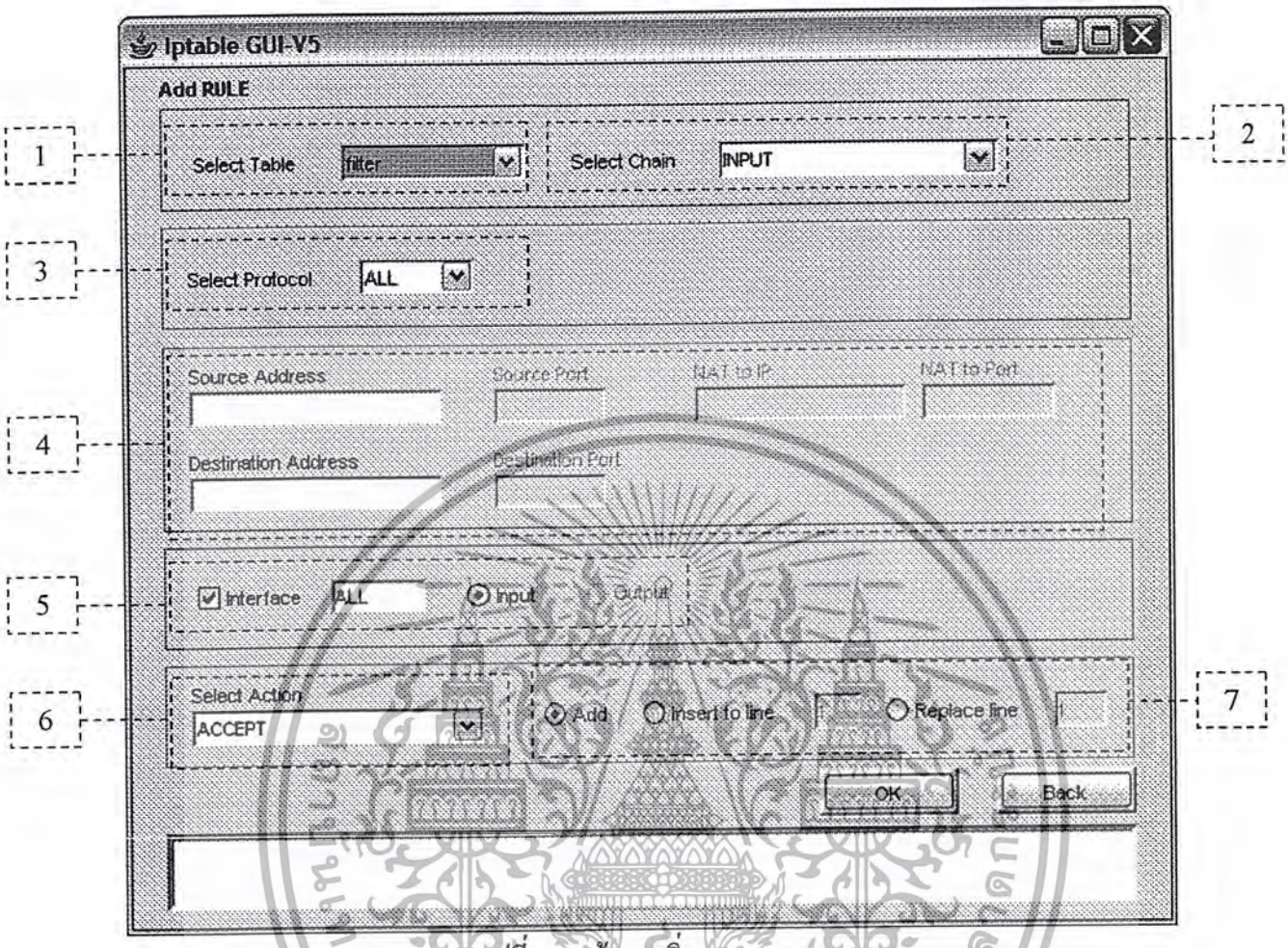
หน้า main แบ่งออก เป็น 4 ส่วน คือ ส่วนของการจัดการ RULE, การจัดการ CHAIN, การแสดง RULE, และการ RESET




*รูปที่ 6-1 หน้าหลักของ9 Iptables GUI*

1. ส่วนการจัดการ RULE แบ่งเป็น การ สร้าง และลบ RULE
2. ส่วนการจัดการ CHAIN แบ่งเป็นการ สร้าง, ลบ, เปลี่ยนชื่อ และการ ตั้งนโยบาย (Policy) ของ CHAIN
3. ส่วนการแสดง RULE โดยเลือกดู RULE ทั้งหมด หรือ RULE ในแต่ละ CHAIN ได้
4. การ RESET เพื่อทำการตั้งค่าให้กลับสู่ค่าเดิมของ firewall คือ ลบ RULE ในทุกๆ CHAIN ออกทั้งหมด และตั้งนโยบายเป็น DROP ทั้งหมด

การเพิ่ม RULE




*รูปที่ 6-2 หน้าการเพิ่มกฎ*

หลักการสร้าง RULE

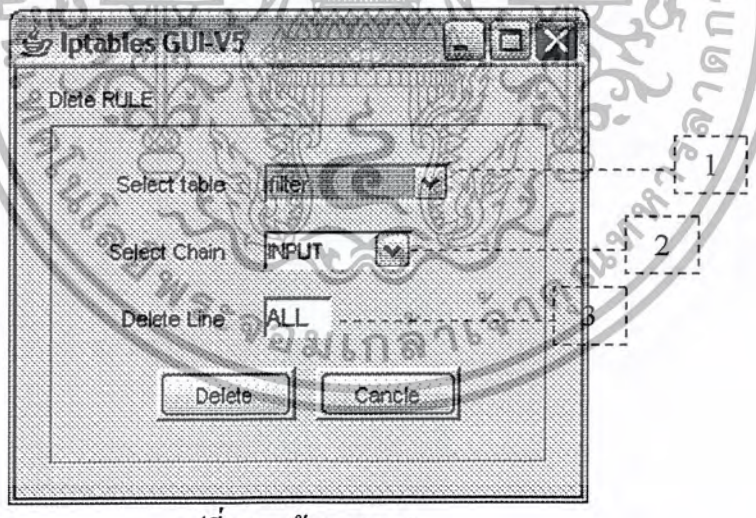
1. เลือก table ที่จะทำการสร้าง RULE เนื่องจาก table มี สาม table หลักดังกล่าวข้างต้น และการเลือก table ก็ต้องเลือกเพียง table ใด table หนึ่ง จึงใช้ interface เป็น list แบบ popup โดยให้ Filter table เป็น default เนื่องจาก table นี้จะถูกใช้บ่อยที่สุด
2. เลือก chain ด้วยเหตุผลเดียวกันกับ table คือต้องเลือกเพียง chain ใด chain หนึ่ง จึงใช้ interface เป็น list แบบ popup
3. เลือก Protocol ใช้ interface เป็น list แบบ popup ด้วยเหตุผลเดียวกับข้อ 1และ2 แต่จะแตกต่างไปจาก 1และ2 คือ สามารถที่จะเลือกใช้ทุกๆ Protocol ได้ จึงเพิ่ม ตัวเลือก ALL ไว้ใน List ด้วย
4. ระบุ หมายเลข IP address และ Port ของ เครื่องต้นทางและปลายทาง เนื่องจากการระบุหมายเลข IP address และ Port นั้นมีได้หลายค่า และยังสามารถระบุเป็นชื่อได้ เช่น การระบุ Port เป็น http ดังนั้น การใช้ interface เป็น text field จึงสดวกและเหมาะสมกว่า

interface แบบ อื่นๆ และอีกประการหนึ่ง การระบุ Port oนั้นจะสามารถ การทำได้ก็ต่อเมื่อ ต้องมีการเลือก Protocol เป็น tcp หรือ udp เท่านั้นในเบื้องต้นจึงต้องทำหาร disenable ไว้ก่อน

ส่วนของ NAT IP และ Port จะใช้ระบุ หมายเลข IP และ Port ในการทำ NAT (ทำ ใน NAT table)

5. ระบุ interface การเข้าออกของ packet เนื่องจาก ส่วน interface จะระบุหรือไม่ก็ได้จึงทำ check box ไว้ให้เลือกว่าจะระบุหรือไม่ระบุ interface ชื่อของ interface จะระบุเป็นชื่อ เช่น eth0, eth1, ppp0, ppp1 ฯลฯ ดังนั้นจึงใช้ text field และต้องเลือกด้วยว่าจะเป็นขาเข้าหรือออกอย่างใดอย่างหนึ่งจึงใช้ radio button
6. การเลือก action เนื่องจาก firewall โดย default policy แล้วจะเป็น DROP คือ ไม่อนุญาติให้ใครผ่านเข้าออกทั้งนั้น ดังนั้นการตั้งกฎโดยส่วนใหญ่ จึงเป็นการตั้งเพื่อเลือกอนุญาติเท่านั้น จึงตั้ง default เป็น ACCEPT
7. เลือกว่าจะเพิ่ม(add) RULE ต่อท้าย RULE เดิม หรือแทรก RULE (insert) หรือแทนที่ RULE เดิมที่มีอยู่(replace) ต้องเลือกอย่างใดอย่างหนึ่งจึงใช้ radio button ซึ่งโดยปกติแล้วจะเป็นการเพิ่มต่อท้ายจึงให้เลือกเพิ่มเป็น default

การลบ RULE

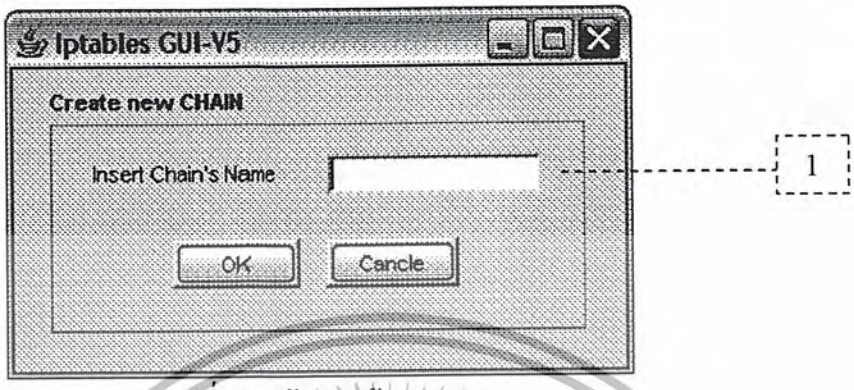



*รูปที่ 6-3 หน้าการลบกฎ*

1. เลือก table ที่จะทำการสร้าง RULE เนื่องจาก table มี สาม table หลักดังกล่าวข้างต้น และการเลือก table ก็ต้องเลือกเพียง table ใด table หนึ่ง จึ่งใช้ interface เป็น list แบบ popup
2. เลือก chain ด้วยเหตุผลเดียวกันกับ table คือต้องเลือกเพียง chain ใด chain หนึ่ง จึ่งใช้ interface เป็น list แบบ popup

3. ระบุลำดับของ RULE ที่ต้องการลบ ซึ่งการใส่ค่าสามารถใส่ได้ตั้งแต่ 1, 2, 3, ... จึงใช้ text field

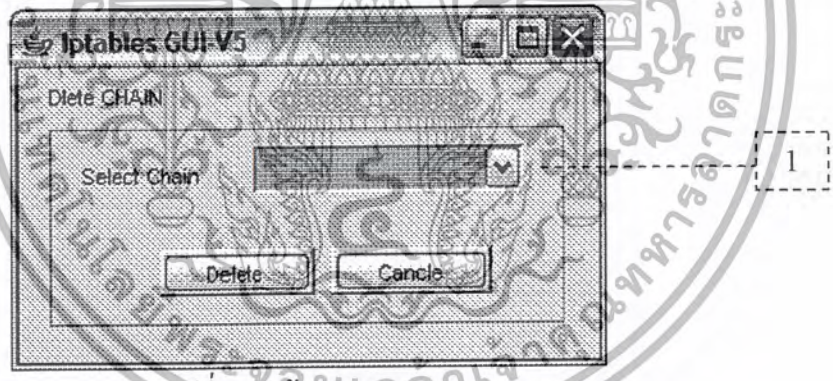
**การสร้าง CHAIN**




*รูปที่ 6-4 หน้าการสร้าง Chain*

1. ใส่เชื่อ CHAIN ที่ต้องการสร้าง ซึ่งการใส่ค่าสามารถใส่ได้หลากหลาย จึงใช้ text field

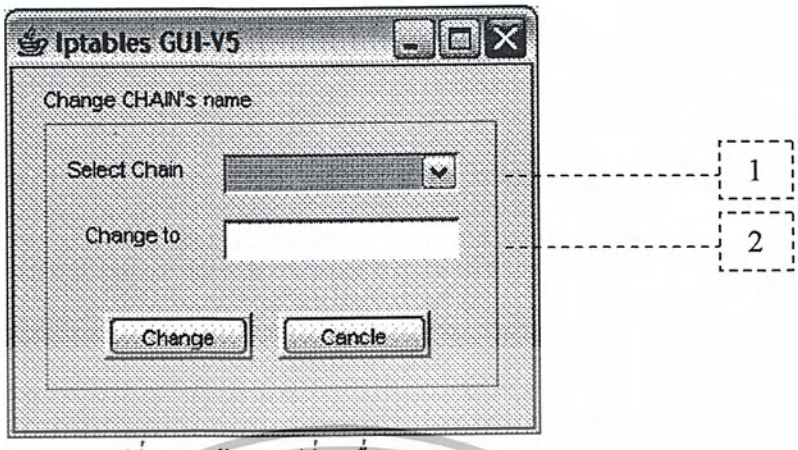
**การลบ CHAIN**




*รูปที่ 6-5 หน้าการลบ Chain*

1. เลือก CHAIN ที่ต้องการลบ ซึ่งต้องเลือกจาก CHAIN ที่มีอยู่ จึงใช้ interface เป็น list แบบ popup

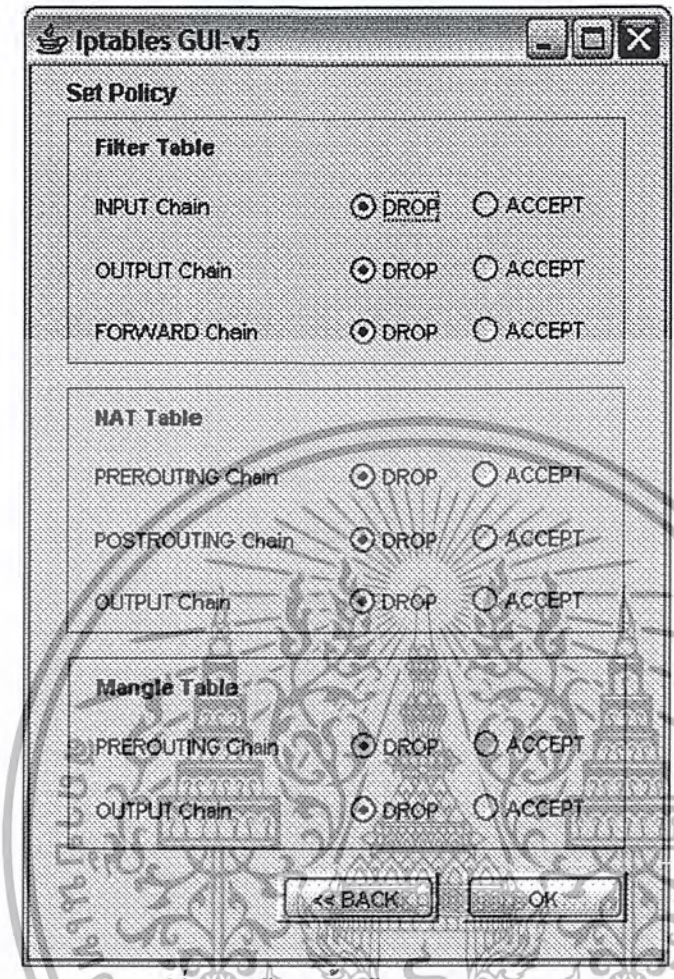
**การเปลี่ยนชื่อ CHAIN**




*รูปที่ 6-6 หน้าการเปลี่ยนชื่อ Chain*

1. เลือก CHAIN ที่ต้องการลบ ซึ่งต้องเลือกจาก CHAIN ที่มีอยู่ จึงใช้ interface เป็น list แบบ popup
2. ใส่ชื่อที่ต้องการเปลี่ยน ซึ่งการใส่ค่าสามารถใส่ได้หลากหลาย จึงใช้ text field

การตั้งนโยบายของ CHAIN

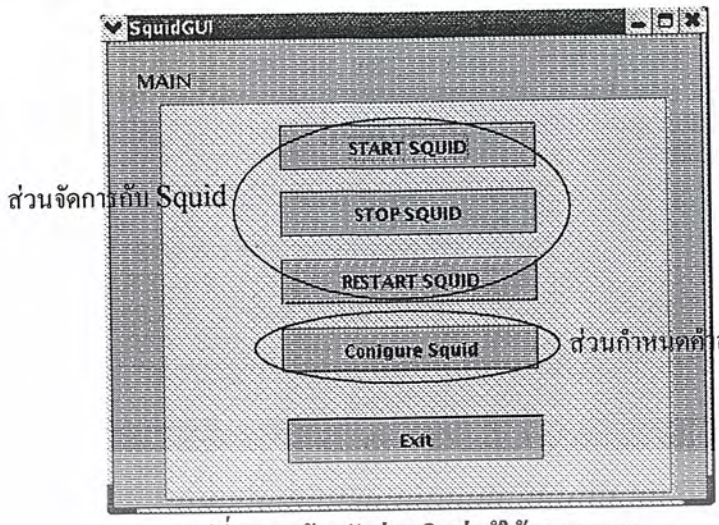



*รูปที่ 6-7 หน้าการตั้งนโยบายของ Chain*

เนื่องจากมีเพียงสองลักษณะคือ DROP หรือ ACCEPT เท่านั้นจึงใช้ radio button

**6.1.1.2 หลักการออกแบบ Squid**

เนื่องจากการกำหนดค่าต่างๆใน Squid จำเป็นต้องติดต่อกับไฟล์ที่มีชื่อว่า Squid.conf ซึ่งไฟล์ Squid.conf นั้นมีคำสั่งที่หลากหลายและมีเป็นพันบรรทัด หลักการออกแบบก็คือการจัดกลุ่มคำสั่งต่างๆที่มีลักษณะเหมือนกันไว้ด้วยกัน เพื่อให้ง่ายต่อการใช้มากยิ่งขึ้น ซึ่งในหน้าหลักนี้ในการเลือกที่จะทำสิ่งที่ต้องการไม่ว่าจะเป็น การจัดการเกี่ยวกับ Squid และการกำหนดคำสั่งต่างๆใน Squid สามารถเลือกทำได้ทีละอย่างเท่านั้นจึงใช้ ปุ่ม (Button) ในการเลือก

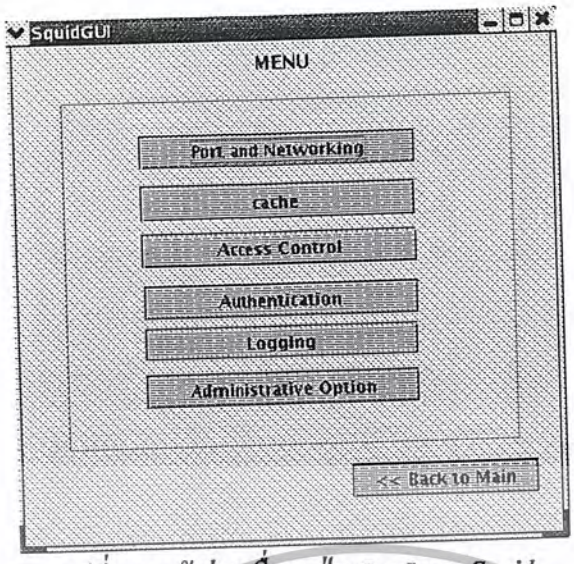



*รูปที่ 6-8 หน้าหลักส่วนติดต่อผู้ใช้ของ Squid*

โดยจากรูปที่ 6-9 จะแบ่งเป็นการจัดการกับ Squid ซึ่งผู้ใช้จะสามารถสตาร์ท , สต็อป หรือ รีสตาร์ท เซอร์วิสได้ และผู้ใช้สามารถกำหนดค่าต่างๆตามต้องการได้ที่ Configure Squid

สำหรับหน้า Configure Squid จะสามารถแบ่งออกได้เป็นอีก 6 หน้าย่อยซึ่งเป็นหน้าที่จัดกลุ่มคำสั่งต่างๆในไฟล์ Squid.conf ไว้ ซึ่งแบ่งเป็นกลุ่มดังนี้

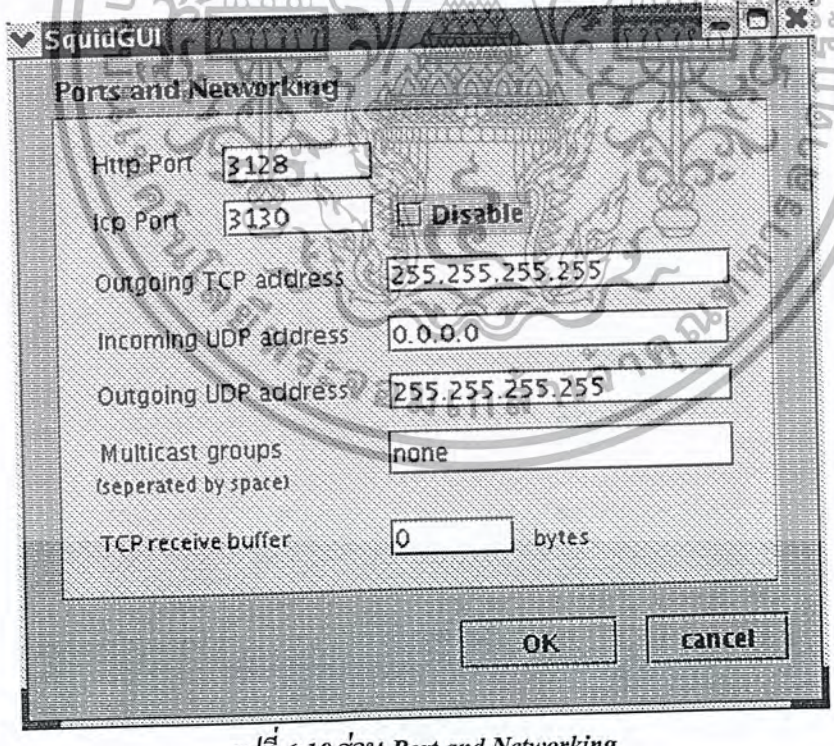
- Port and Networking ซึ่งเป็นกลุ่มที่ให้ผู้ใช้กำหนดคำสั่งที่เกี่ยวกับพอร์ตต่างๆและส่วนที่เกี่ยวข้องกับเน็ตเวิร์ก
- Cache ซึ่งเป็นกลุ่มที่ให้ผู้ใช้กำหนดค่าต่างๆเกี่ยวกับแคช
- Access Control ซึ่งเป็นกลุ่มที่ใช้กำหนดการกำหนดสิทธิ์ในการเข้าถึง
- Authentication ซึ่งเป็นกลุ่มที่ให้ผู้ใช้กำหนดการพิสูจน์ตน
- Logging ซึ่งเป็นกลุ่มที่ใช้กำหนดไฟล์ล็อกต่างๆ
- Administrative Option ซึ่งเป็นกลุ่มที่ให้ผู้ใช้กำหนดค่าต่างๆที่ไว้แสดงเมื่อ Squid ไม่สามารถให้บริการสิ่งที่ผู้เรียกใช้ต้องการได้




*รูปที่ 6-9 หน้าย่อยเมื่อกดปุ่ม Configure Squid*

**ส่วน Port and Networking**

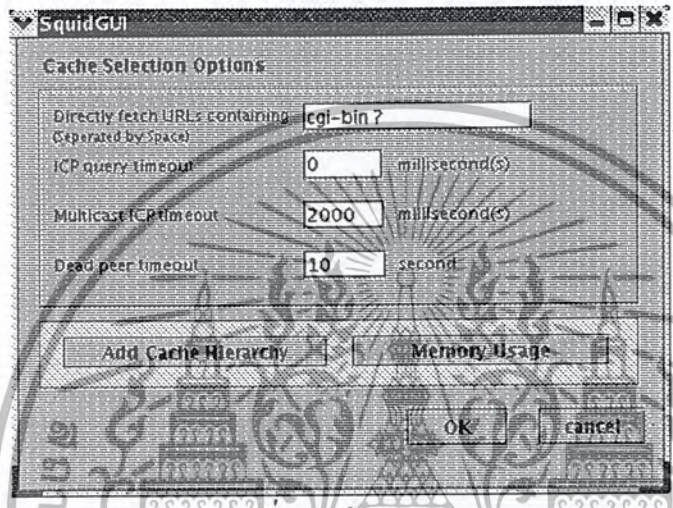
กลุ่ม Port and Networking เป็นกลุ่มที่ให้ผู้ใช้กำหนดคำสั่งที่เกี่ยวกับพอร์ตต่างๆและส่วนที่เกี่ยวข้องกับเน็ตเวิร์ก ซึ่งจะแยกออกเป็นแต่ละคำสั่งให้ผู้ใช้ใส่ค่าต่างๆตามที่ต้องการ ได้โดยใช้ TextField เป็นตัวที่ให้ผู้ใช้กรอกค่าต่างๆ




*รูปที่ 6-10 ส่วน Port and Networking*

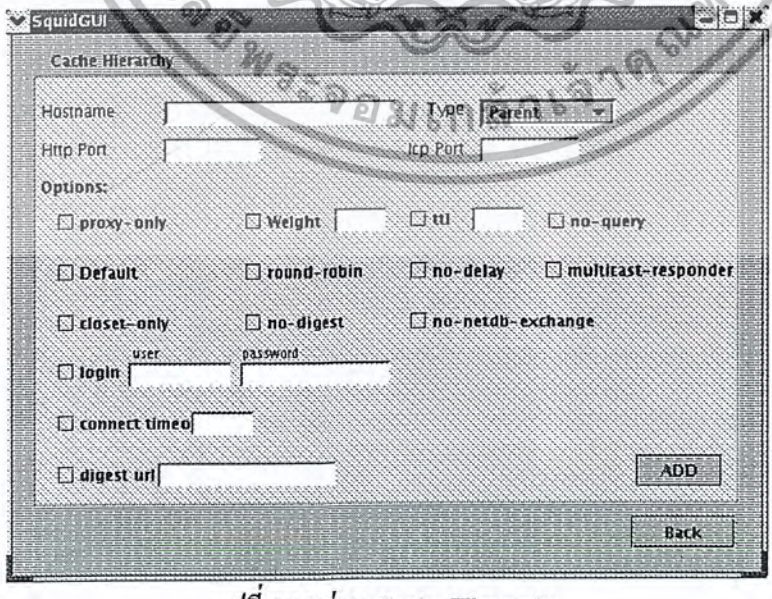
**ส่วน cache**

กลุ่ม Cache ซึ่งเป็นกลุ่มที่ให้ผู้ใช้กำหนดค่าต่างๆเกี่ยวกับแคช ซึ่งจะแยกออกเป็นแต่ละคำสั่งให้ผู้ใช้ใส่ค่าต่างๆตามที่ต้องการได้โดยใช้ TextField เป็นตัวที่ให้ผู้ใช้กรอกค่าต่างๆ ซึ่งจะสามารถแบ่งได้ 2 กลุ่มในกลุ่มแคชได้แก่ Add Cache Hierarchy ซึ่งเป็นคำสั่งต่างๆที่ให้ผู้ใช้กำหนดค่าต่างๆในการติดต่อกับแคชอื่นๆ และ Memory Usage ซึ่งเป็นคำสั่งต่างๆที่ให้ผู้ใช้กำหนดค่าต่างๆเกี่ยวกับเมมโมรี่ ซึ่งจะใช้ TextField ในการที่จะให้ผู้ใช้กรอกค่าต่างๆที่ต้องการ



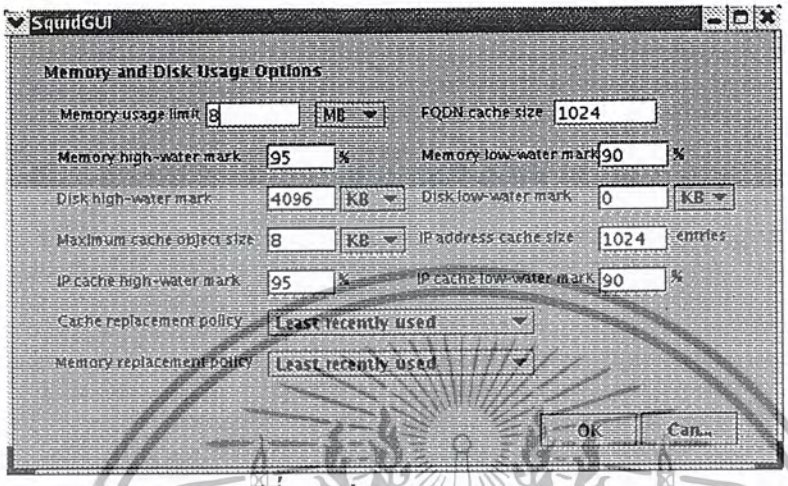

*รูปที่ 6-11 ส่วน cache*

ในส่วนของ Cache Hierarchy จะเป็นส่วนที่ติดต่อกับ Cache อื่นๆ ซึ่งในส่วนของ Options สามารถเลือกได้หลายๆ Option ใน 1 คำสั่งดังนั้นจึงใช้ CheckBox ในการที่จะให้ผู้ใช้เลือก Option ต่างๆ




*รูปที่ 6-12 ส่วน Cache Hierarchy*

ในส่วนของ Memory Usage จะเป็นส่วนที่ให้ผู้ใช้กำหนดค่าต่างๆเกี่ยวกับเมมโมรี่ ซึ่งจะใช้ TextField ในการที่จะให้ผู้ใช้กรอกค่าต่างๆที่ต้องการลงไป ในส่วนที่เป็นค่าที่ตายตัวจะใช้ List ให้ผู้ใช้เลือกแทน

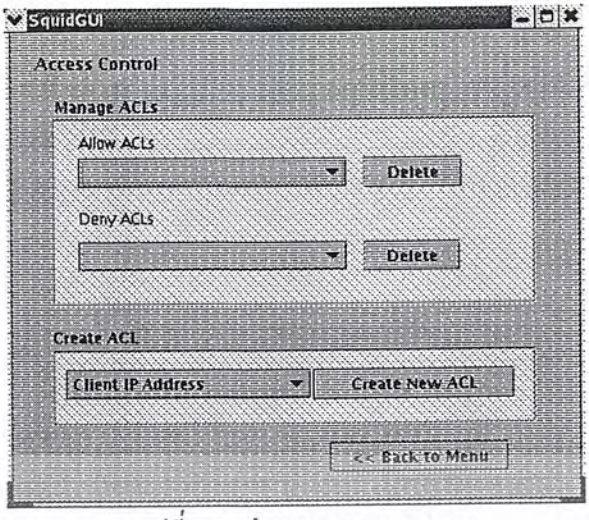



*รูปที่ 6-13 ส่วน Memory Usage*

**ส่วน Access Control**

ในส่วนของ Access Control เป็นกลุ่มที่ใช้กำหนดการกำหนดสิทธิ์ในการเข้าถึงต่างๆ โดยจะแบ่งกลุ่มเป็น

- **Manage ACLs** เป็นส่วนที่ใช้จัดการกับการชื่อของประเภท Access Control ต่างๆ ที่ผู้ใช้ได้กำหนดไว้ โดยจะแบ่งเป็น ส่วนอนุญาต(Allow) และ ส่วนปฏิเสธ(Deny) ซึ่งเมื่อผู้ใช้กำหนดค่าต่างๆที่ต้องการแล้วชื่อของ ACL ที่ได้กำหนดไว้จะมาปรากฏอยู่ที่ ACL ที่อนุญาต(Allow ACLs) หรือ ACL ที่ปฏิเสธ(Deny ACLs) ซึ่งเมื่อผู้ใช้ต้องการลบก็สามารถลบได้จากส่วน Manage ACLs นี้โดยกดปุ่ม Delete ได้
- **Create ACL** เป็นส่วนที่ให้ผู้ใช้กำหนดประเภทต่างๆของ ACL ทำได้โดยการเลือกประเภท ACL ที่กำหนด และกดปุ่ม Create New ACL ซึ่งประเภท ACL นั้นตายตัวจึงใช้ List

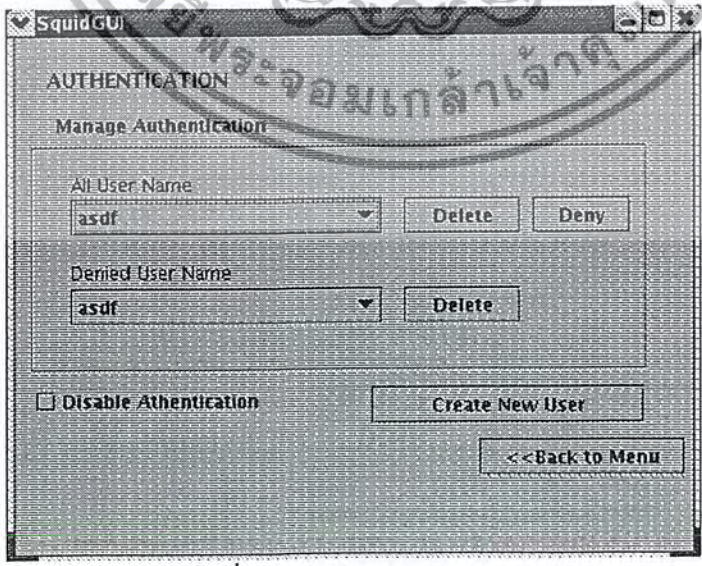



*รูปที่ 6-14 ส่วน Access Control*

**ส่วน Authentication**

ในส่วนการ Authentication เป็นกลุ่มที่ให้ผู้ใช้กำหนดการพิสูจน์ตน โดยจะแบ่งกลุ่มเป็น

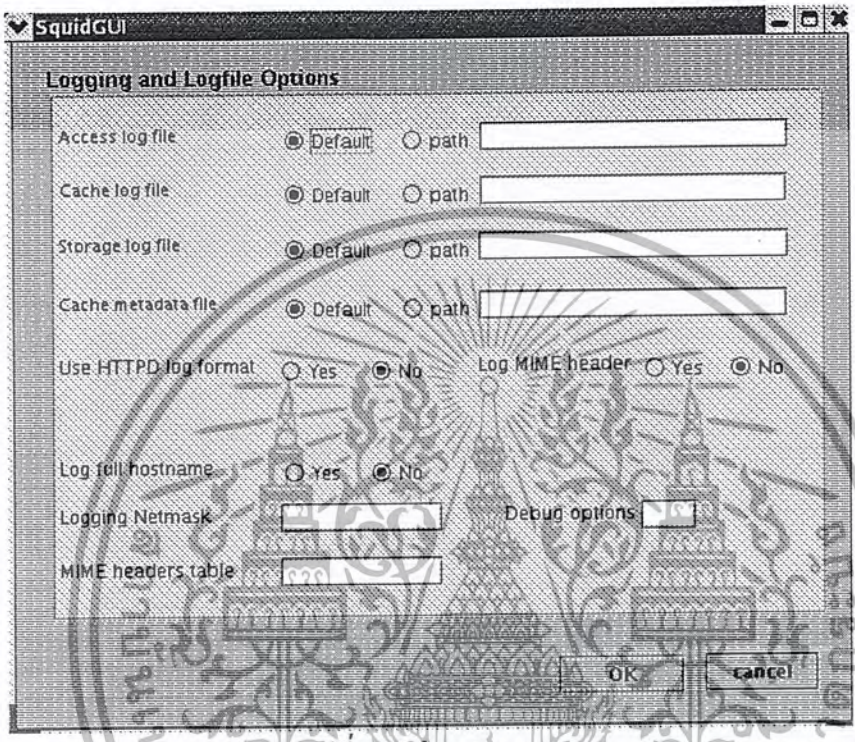
- Manage Authentication เป็นส่วนที่ใช้จัดการกับ User Name ที่ผู้ใช้ได้ทำการสร้างไว้ โดยสามารถสร้างที่ Create New User โดยจะแบ่งเป็น รายชื่อ(All UserName) และส่วนปฏิเสธ(Deny) ซึ่งเมื่อผู้ใช้ได้กำหนด Username แล้ว ชื่อนั้นๆจะมาปรากฏที่ All UserName ซึ่งชื่อที่ผู้ใช้กำหนดนั้นตายตัวจึงใช้ List และผู้ใช้สามารถลบได้โดยกดปุ่ม Delete
- Denied User Name เป็นส่วนที่เก็บชื่อ User Name ที่ได้ถูกปฏิเสธ(Deny)ในการติดต่อไว้ ซึ่งชื่อที่ได้ถูกปฏิเสธนั้นตายตัวจึงใช้ List และผู้ใช้สามารถลบได้โดยกดปุ่ม Delete




*รูปที่ 6-15 ส่วน Authentication*

**ส่วน Logging**

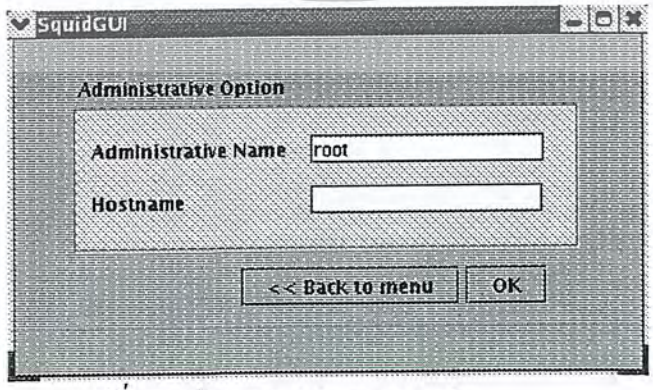
เป็นกลุ่มที่ใช้กำหนดไฟล์ล็อกต่างๆ ซึ่งไฟล์ล็อกต่างๆ โดยปกติแล้วจะมีค่าที่ถูกตั้งไว้แล้ว ดังนั้นจึงใช้ Radio Button ให้ผู้ใช้เลือกว่าจะใช้ค่าที่ได้ตั้งไว้เดิมหรือจะใช้ค่าใหม่




*รูปที่ 6-16 ส่วน logging*

**ส่วน Administrative Option**

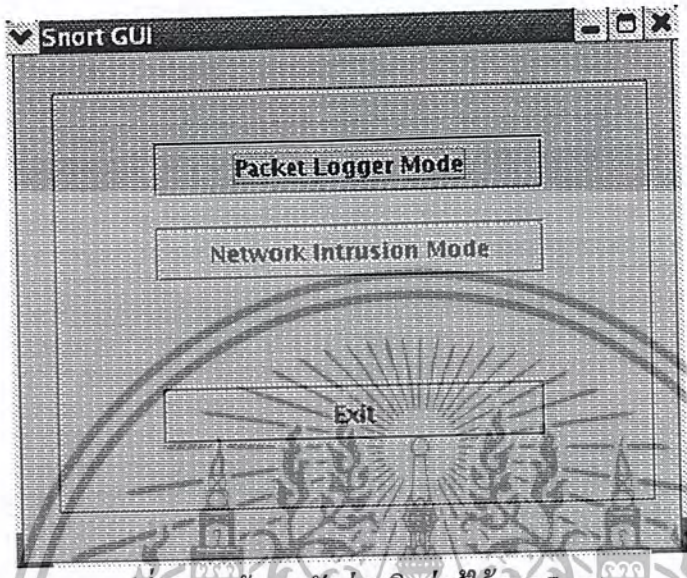
เป็นกลุ่มที่ให้ผู้ใช้กำหนดค่าต่างๆที่ไว้แสดงเมื่อ Squid ไม่สามารถให้บริการสิ่งที่ผู้เรียกใช้ต้องการได้ ดังนั้นจึงใช้ TextField




*รูปที่ 6-17 ส่วน Administrative Option*

**6.1.1.3 หลักการออกแบบ Snort**

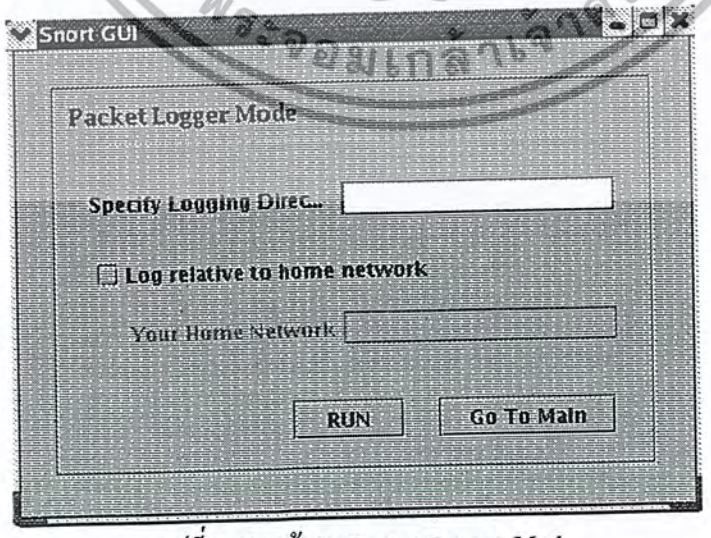
ในการออกแบบ Squid ผู้ใช้สามารถเลือกทำได้ 2 โหมดคือ โหมด Packet Logger และโหมด Network Intrusion ซึ่งผู้ใช้สามารถเลือกโหมดได้ทีละโหมดดังนั้น จึงใช้ Button




*รูปที่ 6-18 หน้าจอหลักส่วนติดต่อผู้ใช้ของ Snort*

โหมด **Packet Logger (Packet Logger)**

เป็นโหมดที่ให้ผู้ใช้สามารถดู packet ที่วิ่งผ่านไปมาใน Network ได้ โดยจะให้ผู้ใช้กำหนด Path ที่ต้องการว่าให้เก็บไว้ในไดเรกทอรีใด จึงใช้ TextField และผู้ใช้สามารถเลือกได้ว่าจะให้เก็บล็อกเฉพาะในเน็ตเวิร์กตัวเองแต่ไม่จำเป็นว่าต้องใช้ ดังนั้น จึงใช้ CheckBox



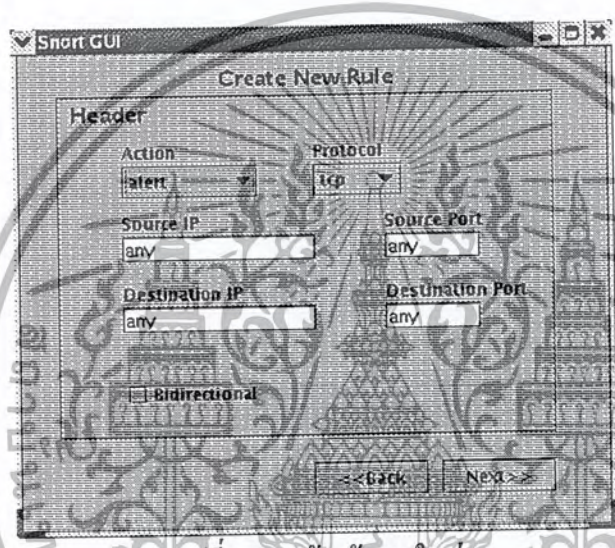

*รูปที่ 6-19 หน้าจอ Packet Logger Mod*

**โหมด Network Intrusion**

เป็นโหมดที่ให้ผู้ใช้กำหนดกฎของ Snort ได้ ซึ่งจะแบ่งได้ทั้งหมด 3 หน้าด้วยกันได้แก่

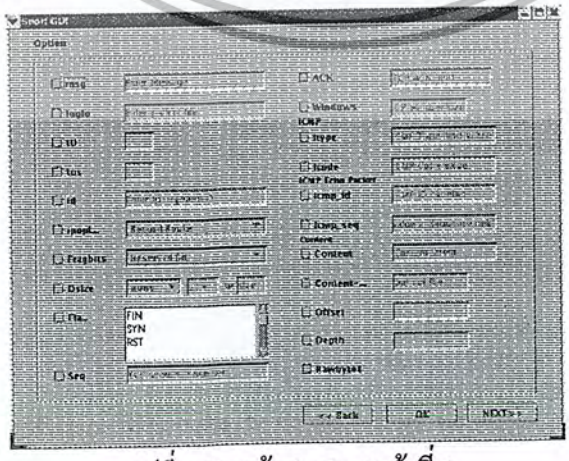
- หน้าสร้างกฎใหม่(Create New Rule)
- หน้า Option หน้าที่ 1
- หน้า Option หน้าที่ 2
- ส่วนเพิ่ม Option

หน้าสร้างกฎใหม่จะสามารถให้ผู้ใช้ใส่ค่าที่ตัวเองต้องการลงไป ดังนั้นจึงใช้ TextField และสำหรับค่าที่ตายตัวจะใช้ List

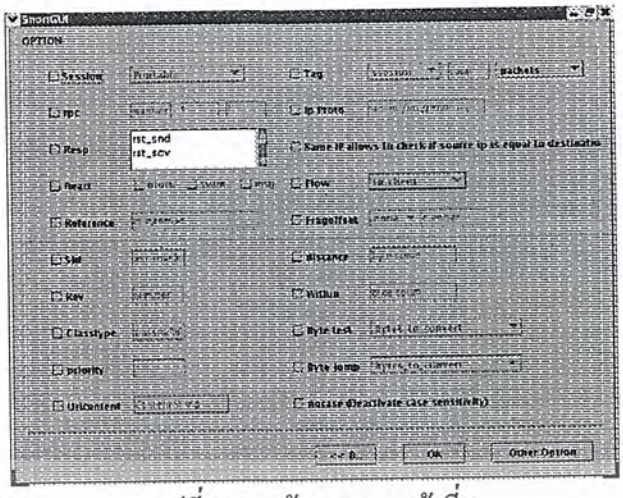



*รูปที่ 6-20 หน้าสร้างกฎใหม่*

หน้า Option เป็นหน้าที่ให้ผู้ใช้สามารถเลือก Option ต่างๆที่ต้องการใช้ได้ โดยไม่จำเป็นต้องใช้ทุก Option ดังนั้นจึงใช้ CheckBox และสำหรับค่าที่ตายตัวจะใช้ List



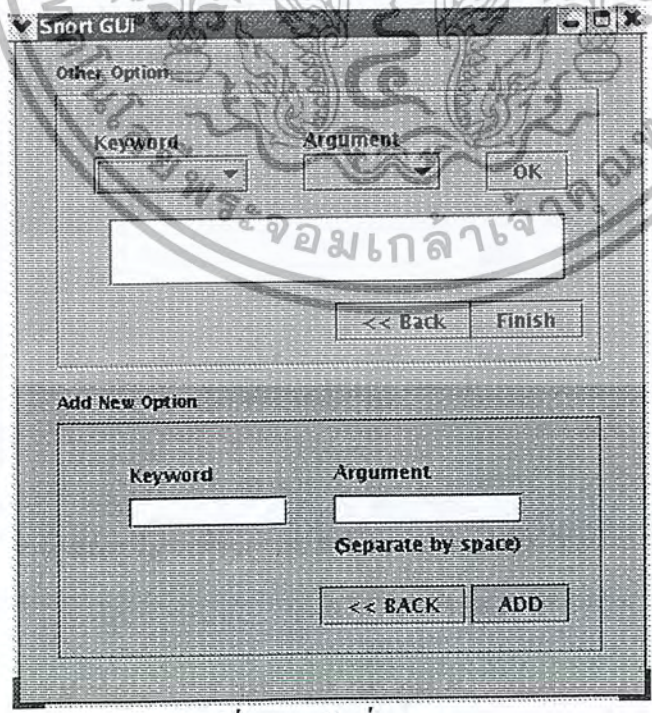
*รูปที่ 6-21 หน้า Option หน้าที่ 1*




*รูปที่ 6-22 หน้า Option หน้าที่ 2*

เนื่องจากโปรแกรม Snort ในแต่ละเวอร์ชันจะมี Option ไม่เท่ากัน เวอร์ชันยิ่งสูงเท่าไหร่ Option จะเพิ่มมากขึ้นตามไปด้วย ดังนั้นจึงออกแบบหน้านี้มาเพื่อให้ผู้ใช้สามารถเพิ่ม Option ที่ต้องการไปได้ เพื่อให้การออกแบบในการสร้างกฎยืดหยุ่นมากยิ่งขึ้น

Option จะแบ่งเป็น Keyword และ Argument ผู้ใช้สามารถใส่ค่าที่ต้องการลงไปได้อย่างไม่จำกัดดังนั้นจึงใช้ TextField และเมื่อ Option ได้ถูกกำหนดขึ้นแล้วก็จะไปขึ้นที่ List Keyword และ Argument และ Option ที่ผู้ใช้สร้างขึ้นมาจะสามารถดูได้ จึงใช้ TextArea ในการแสดง Option ที่สร้างขึ้นมา

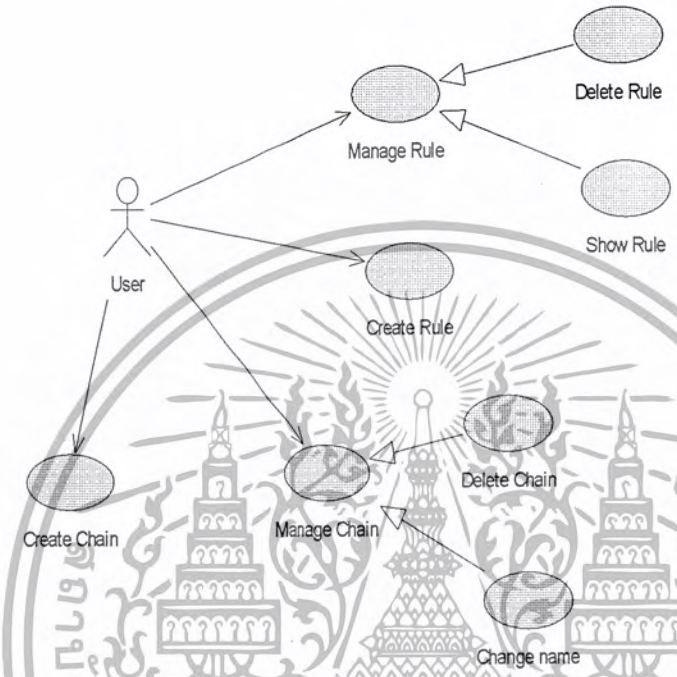



*รูปที่ 6-23 หน้าเพิ่ม Option*

**6.2 Iptables**

**6.2.1 Use Case Diagram**

แผนภาพยูซเคส แสดงฟังก์ชั่นการทำงานที่สำคัญๆของระบบ ซึ่งมีรายละเอียดแบ่งตามกราฟฟิกยูสเซอร์อินเตอร์เฟสดังนี้




*รูปที่ 6-24 Use Case Diagram ของ Iptables*

จากรูปฟังก์ชันการทำงานหลักที่สำคัญจะประกอบไปด้วยฟังก์ชันต่างๆดังนี้

1. Create Rule

   เป็นฟังก์ชันในการสร้างกฎใน Iptables โดยผู้ใช้สามารถกำหนดกฎที่ต้องการได้

2. Manage Rule

   เป็นฟังก์ชันจัดการกฎที่ได้สร้างขึ้นมาไว้แล้ว โดยผู้ใช้สามารถลบกฎที่สร้างไว้แล้ว และสามารถดูกฎที่ได้สร้างไว้

3. Create Chain

   เป็นฟังก์ชันในการสร้าง Chain ที่นอกเหนือจาก Chain ที่ได้มีการกำหนดไว้แล้วใน Iptables

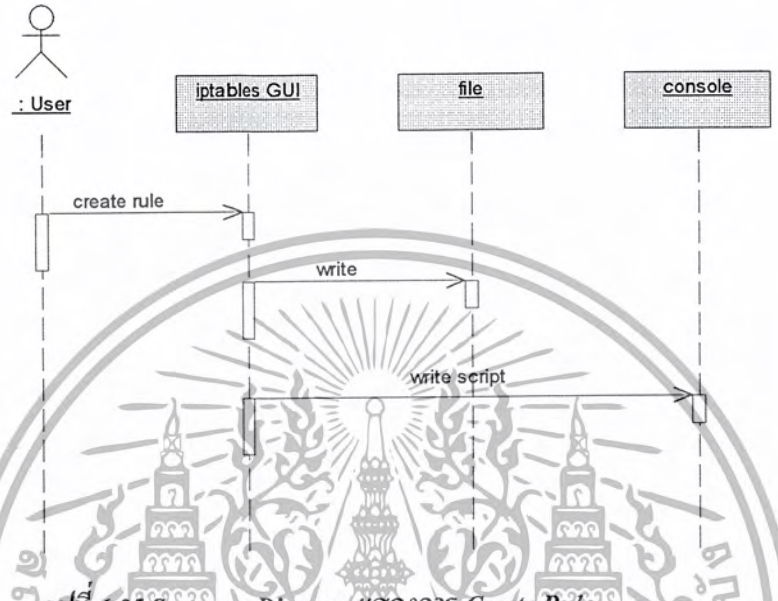
4. Manage Chain

   เป็นฟังก์ชันในการจัดการ Chain ที่ได้สร้างเอาไว้ โดยผู้ใช้สามารถลบ Chain ที่ได้สร้างไว้ สามารถเปลี่ยนชื่อ Chain ที่ได้สร้างไว้ได้

### 6.2.2 Iptables Sequence Diagram

#### 6.2.2.1 Sequence Diagram แสดงการ Create Rule

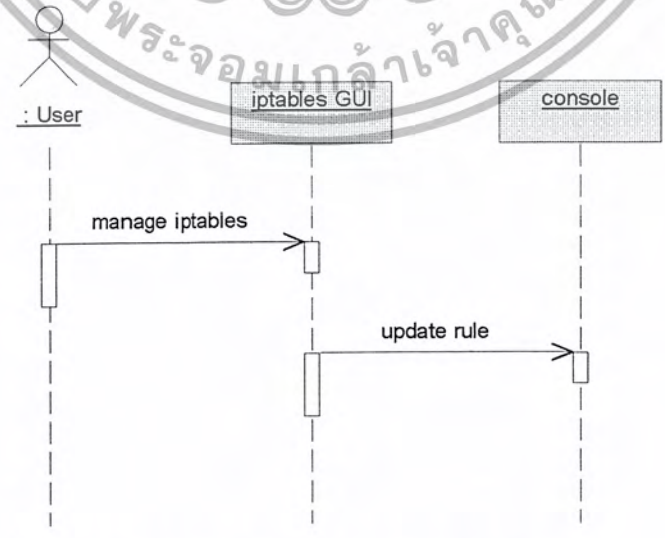
เป็นการจำลองสถานการณ์ขณะที่ผู้ใช้ทำการสร้างกฎขึ้นมาใหม่ โดย Iptables GUI จะทำการเขียนกฎนั้นๆเก็บไว้ในไฟล์และจากนั้นจะเจนเนอเรทคำสั่งส่งไปเขียนยัง console




*รูปที่ 6-25 Sequence Diagram แสดงการ Create Rule*

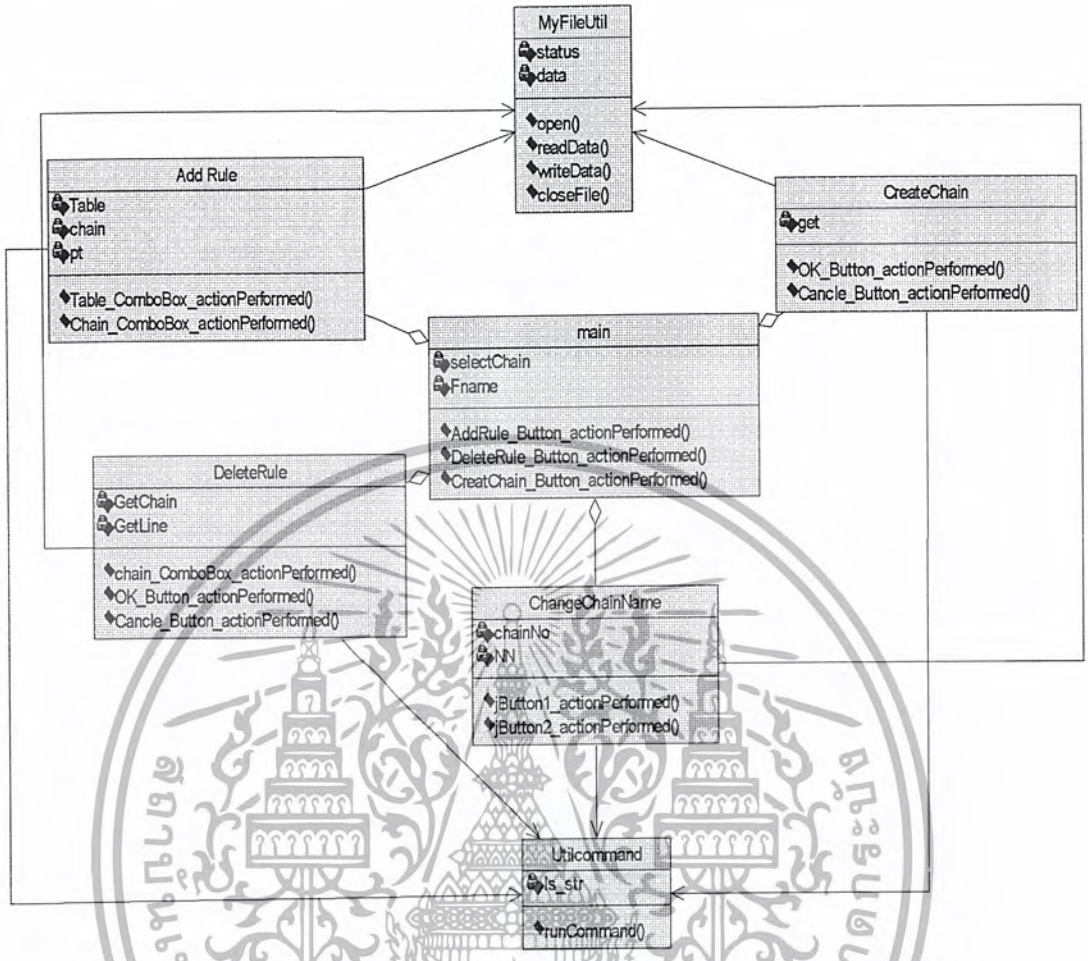
#### 6.2.2.2 Sequence Diagram แสดงการ Manage Rule

เป็นการจำลองสถานการณ์ขณะที่ผู้ใช้จัดการเกี่ยวกับกฎไม่ว่าจะเป็นการจัดการเกี่ยวกับกฎหรือการจัดการเกี่ยวกับ Chain ซึ่งเมื่อผู้ใช้จัดการเกี่ยวกับกฎ Iptables GUI ก็จะทำการเขียนคำสั่งสร้างกฎนั้นๆไปที่ console




*รูปที่ 6-26 Sequence Diagram แสดงการ Manage Rule*

### 6.2.3 Iptables Class Diagram



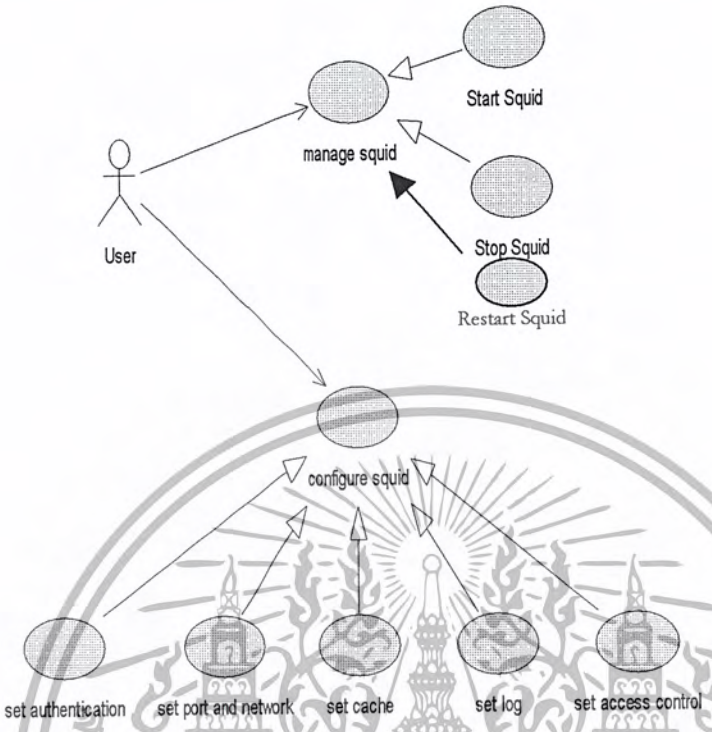

*รูปที่ 6-27 แสดง class diagram ของ Iptables*

อธิบายการทำงาน

- MyFileUtil เป็น class ที่จัดการเกี่ยวกับไฟล์
- Main เป็น class ที่เป็นตัวหลักที่ทำการเรียกหน้าต่างๆในกราฟฟิกยูสเซอร์อินเตอร์เฟส
- Add Rule เป็น class ที่ให้ผู้ใช้เพิ่มกฎต่างๆ
- DeleteRule เป็น class ที่ให้ผู้ใช้ทำการลบกฎต่างๆ
- CreateChain เป็น class ที่ให้ผู้ใช้สร้าง Chain ที่ต้องการได้
- ChangeChainName เป็น class ที่ให้ผู้ใช้เปลี่ยนชื่อ Chain ที่ได้สร้างขึ้นมา
- UtilCommand เป็น class ที่ทำให้ Iptables GUI ไปติดต่อกับ console ได้

## 6.3 Squid

### 6.3.1 Use Case Diagram




*รูปที่ 6-28Use Case Diagram ของ Squid*

จากรูปฟังก์ชันการทำงานหลักที่สำคัญจะประกอบไปด้วยฟังก์ชันต่างๆดังนี้

1. Manage Squid

   เป็นฟังก์ชันในการจัดการ Squid โดยผู้ใช้สามารถ Start และ Stop Squid ได้

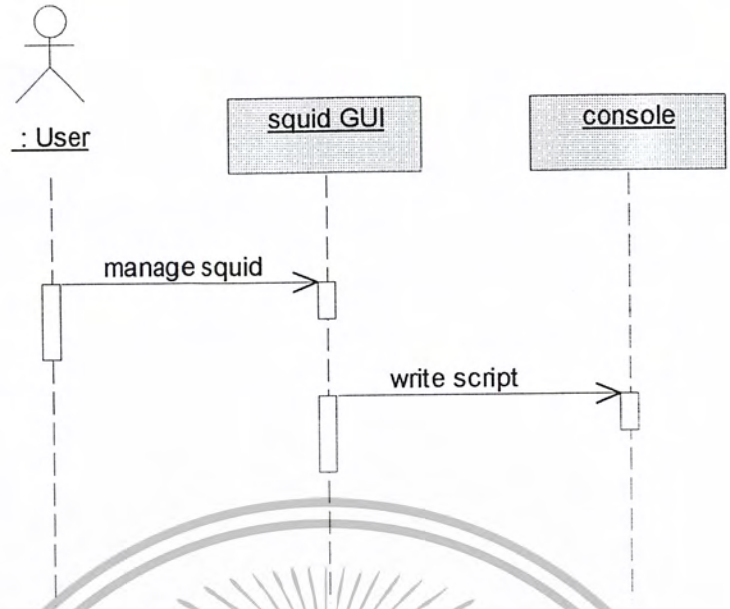
2. Configure Squid

   เป็นฟังก์ชันในการคอนฟิก Squid โดยผู้ใช้สามารถคอนฟิกการพิสูจน์ตน (authentication) , การเซตพอร์ตและเน็ตเวิร์ก , การเซ็ตแคช , เซ็ตล็อค , เซ็ตแอคเซสคอนโทรล

### 6.3.2 Sequence Diagram

#### 6.3.2.1 Sequence Diagram แสดงการ Manage Squid

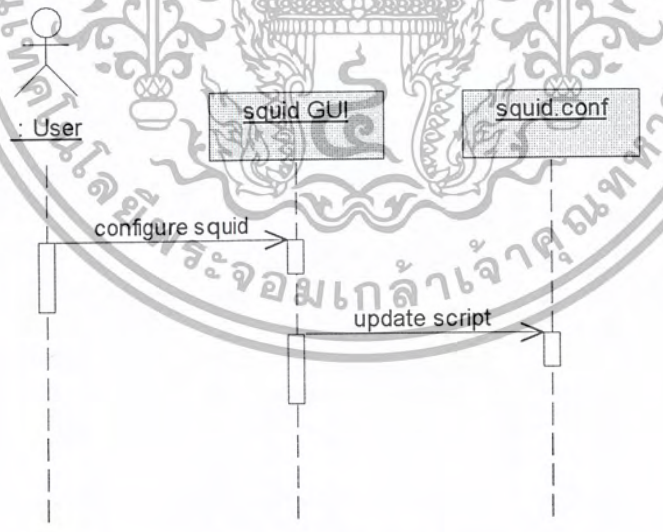
เป็นการจำลองสถานการณ์ขณะที่ผู้ใช้จัดการเกี่ยวกับ Squid คือ Start และ Stop Squid ซึ่งเมื่อผู้ใช้เลือกไม่ว่าจะเป็น Start หรือ Stop Squid จากนั้น squid GUI จะทำการเขียนคำสั่งไปที่หน้า console




*รูปที่ 6-29 Sequence Diagram แสดงการ Manage Squid*

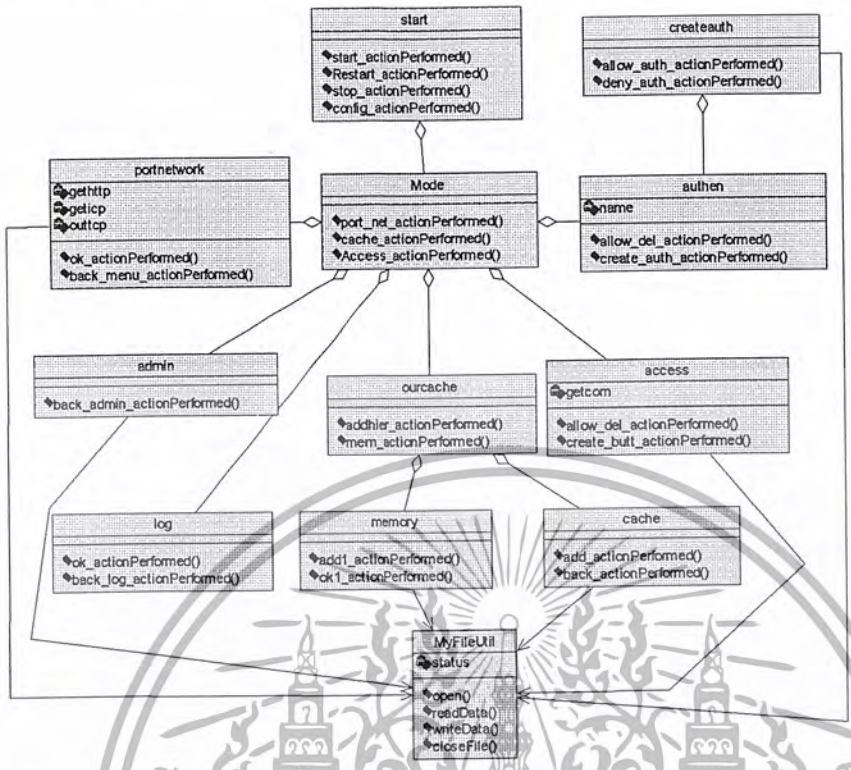
**6.3.2.2 Sequence Diagram แสดงการ Configure Squid**

เป็นการจำลองสถานการณ์ขณะที่ผู้ใช้ปรับแต่งค่าต่างๆ ในไฟล์ squid.conf เมื่อผู้ใช้ทำการปรับแต่ง Squid Squid GUI ก็จะทำการเขียนคำสั่งนั้นๆ เข้าไปยังไฟล์ squid.conf




*รูปที่ 6-30 Sequence Diagram แสดงการ Configure Squid*

### 6.3.3 Squid Class Diagram



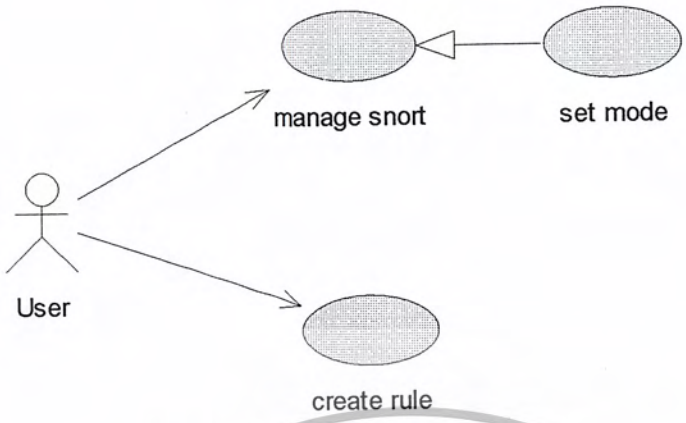

*รูปที่ 6-31 แสดง class diagram ของ Squid*

อธิบายการทำงาน

- start เป็น class ที่เป็นตัวหลักที่ทำการเรียกหน้าต่างๆ ในกราฟฟิกยูสเซอร์อินเตอร์เฟส
- mode เป็น class ที่เป็นตัวหลักที่ทำการเรียกหน้าต่างๆเกี่ยวกับการคอนฟิกในยูสเซอร์อินเตอร์เฟส
- portnetwork เป็น class ที่ให้ผู้ใช้คอนฟิกพอร์ตและเน็ตเวิร์ค
- admin เป็น class ที่ให้ผู้ใช้คอนฟิกส่วนของผู้ดูแลระบบ
- ourcache เป็น class ที่ให้ผู้ใช้จัดการเกี่ยวกับแคช
- log เป็น class ที่ให้ผู้ใช้จัดการเกี่ยวกับล็อค
- access เป็น class ที่ให้ผู้ใช้จัดการเกี่ยวกับแอคเซสคอนโทรล
- authen เป็น class ที่ให้ผู้ใช้จัดการเกี่ยวกับการพิสูจน์ตน
- memory เป็น class ที่ให้ผู้ใช้จัดการเกี่ยวกับเมมโมรี่
- cache เป็น class ที่ให้ผู้ใช้จัดการเกี่ยวกับ cache hierarhy
- createauth เป็น class ที่ให้ผู้ใช้จัดการเกี่ยวกับการเพิ่มผู้ใช้ในการพิสูจน์ตน
- MyFileUtil เป็น class ที่ให้ผู้ใช้จัดการเกี่ยวกับไฟล์ต่างๆ

## 6.4 Snort

### 6.4.1 Use Case Diagram




*รูปที่ 6-32 Use Case Diagram ของ Snort*

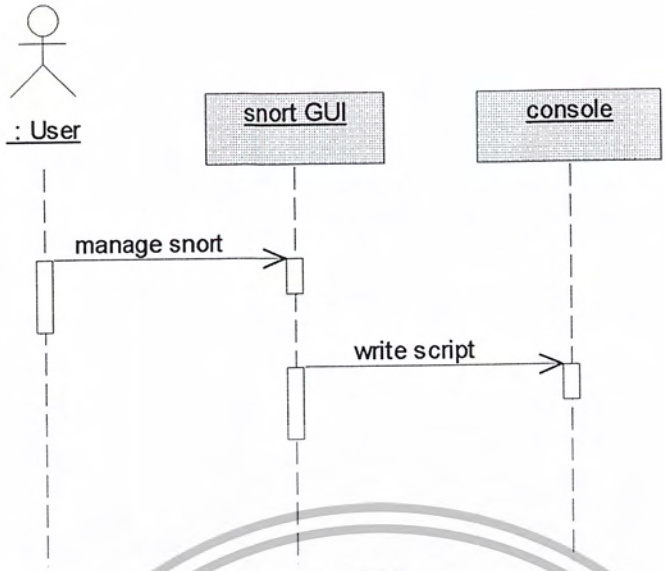
จากรูปฟังก์ชันการทำงานหลักที่สำคัญจะประกอบไปด้วยฟังก์ชันต่างๆดังนี้

1. manage snort
   เป็นฟังก์ชันในการจัดการ snort ซึ่งผู้ใช้สามารถเซตโหมดได้ว่าจะต้องการใช้โหมดใด
2. create rule
   เป็นฟังก์ชันในการเพิ่มกฎ ซึ่งผู้ใช้สามารถเพิ่มกฎที่ต้องการได้

### 6.4.2 Seqeunce Diagram

#### 6.4.2.1 Sequence Diagram แสดงการ Manage Snort

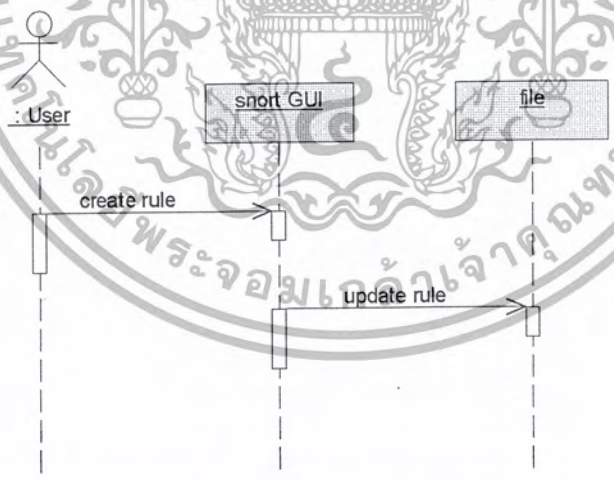
เป็นการจำลองสถานการณ์ขณะที่ผู้ใช้จัดการเกี่ยวกับ Snort คือในการเลือก mode ต่างๆ จากนั้น snort GUI จะทำการเขียนคำสั่งนั้นๆ ไปยัง console




*รูปที่ 6-33 Sequence Diagram แสดงการ Manage Snort*

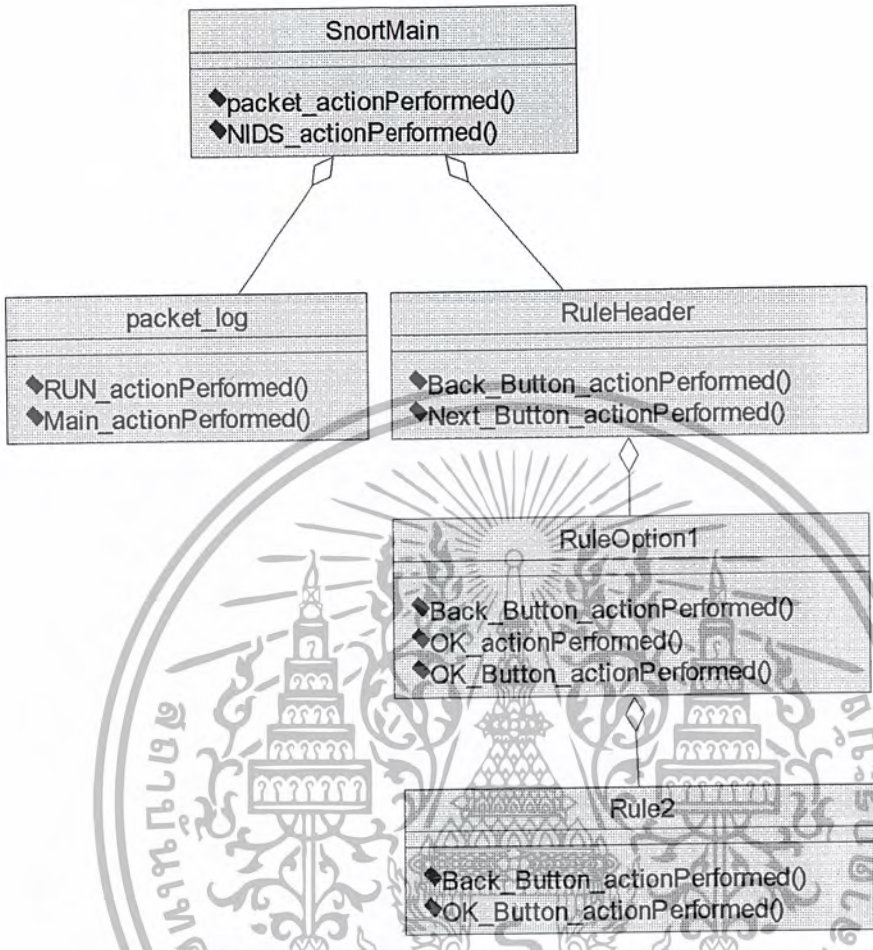
**6.4.2.2 Sequence Diagram แสดงการ Create Rule**

เป็นการจำลองสถานการณ์ขณะที่ผู้ใช้สร้างกฎขึ้นมาใหม่ โดย snort GUI จะทำการเขียนกฎที่ผู้ใช้ต้องการลงไปที่ snort.conf




*รูปที่ 6-34 Sequence Diagram แสดงการ Create Rule*

### 6.4.3 Class Diagram




*รูปที่ 6-35 แสดง class diagram ของ Snort*

อธิบายการทำงาน

- SnortMain เป็น class ที่เป็นตัวหลักที่ทำการเรียกหน้าต่างๆในกราฟฟิกยูสเซอร์อินเตอร์เฟส
- Packet_log เป็น class ที่ให้ผู้ใช้เซตโหมดการทำงานเป็นแบบ Packet logging
- RuleHeader เป็น class ที่เป็นตัวหลักที่ทำการเรียกหน้าต่างๆในการสร้างกฎ
- RuleOption1 เป็น class ที่ให้ผู้ใช้เลือกออกชันต่างๆที่ต้องได้การได้หน้าที่1
- Rule2 เป็น class ที่ให้ผู้ใช้เลือกออกชันต่างๆที่ต้องได้การได้หน้าที่2

# บทที่ 7
# การทดสอบและวิเคราะห์

เนื้อหาในบทนี้เป็นการทดสอบเพื่อประเมินผลโครงงาน ว่าสามารถบรรลุวัตถุประสงค์ของโครงงานที่ได้ตั้งไว้หรือไม่ โดยได้มีการตั้งวัตถุประสงค์ของการทดสอบดังนี้

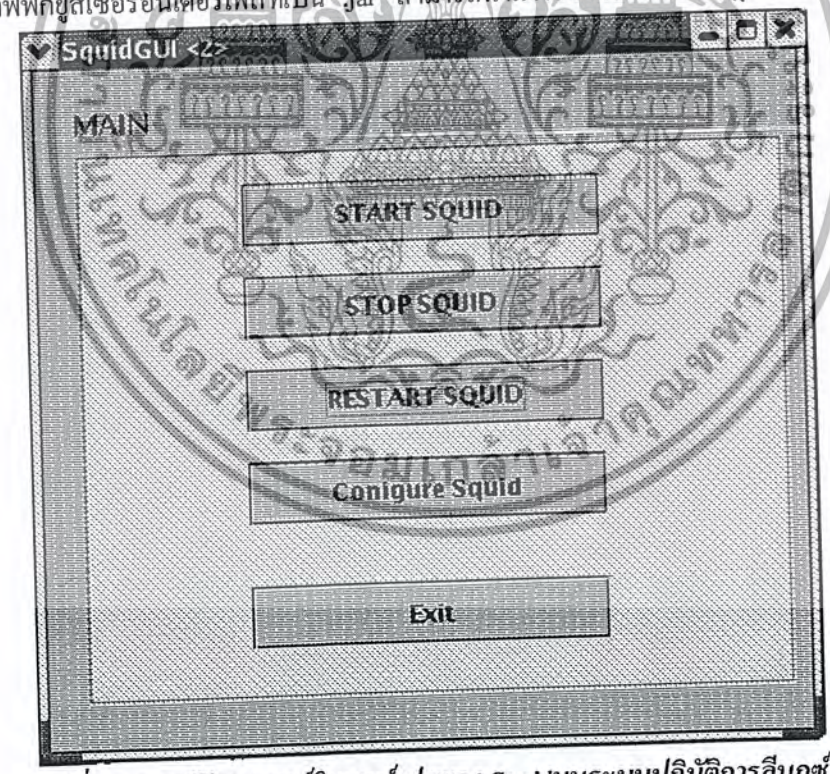
**วัตถุประสงค์การทดสอบ**

1. เพื่อให้สามารถทำงานได้ตามที่ได้ออกแบบไว้
2. เพื่อแสดงถึงการนำโครงงานนี้ไปใช้ได้จริง

กราฟฟิกยูสเซอร์อินเตอร์เฟสของ Squid , Iptables และ Snort ได้พัฒนาขึ้นบนระบบปฏิบัติการวินโดว์ ซึ่งในการใช้งานบนลีนุกซ์นั้นจะต้องสร้างไฟล์ให้เป็น "jar" แล้วค่อยนำมาใช้บนระบบปฏิบัติการลีนุกซ์

## 7.1 การทดสอบการทำงานของกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Squid

กราฟฟิกยูสเซอร์อินเตอร์เฟสที่เป็น "jar" สามารถนำมาใช้งานบนระบบปฏิบัติการลีนุกซ์ได้

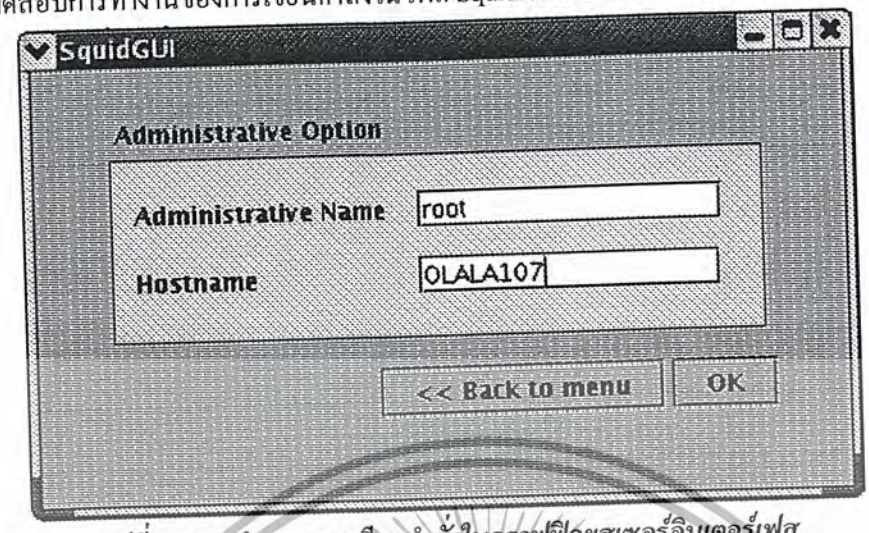



*รูปที่ 7-1 กราฟฟิกยูสเซอร์อินเตอร์เฟสของ Squidบนระบบปฏิบัติการลีนุกซ์*

ต่อมาได้นำกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Squid มาวิเคราะห์กรณีทดสอบดังต่อไปนี้

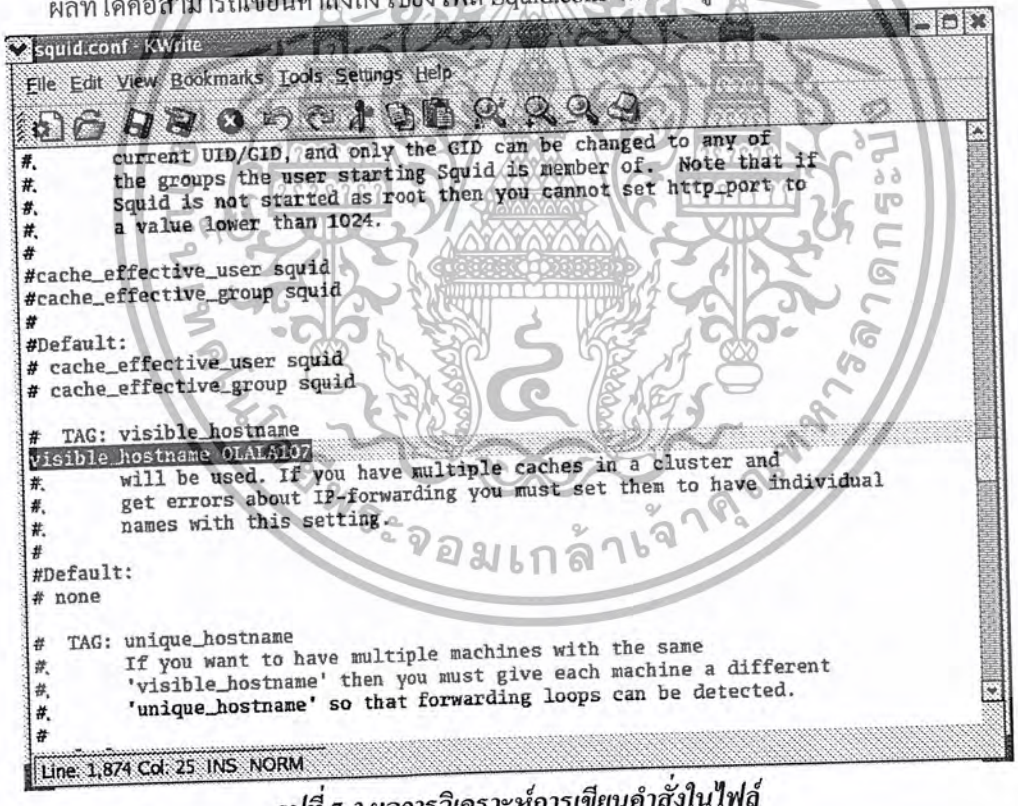
กรณีทดสอบที่ 1

ทดสอบการทำงานของการเขียนคำสั่งในไฟล์ Squid.conf ซึ่งเป็นไฟล์ที่ใช้กำหนดคำสั่งต่างๆ




*รูปที่ 7-2 การกำหนดการเขียนคำสั่งในกราฟฟิกยูสเซอร์อินเตอร์เฟส*

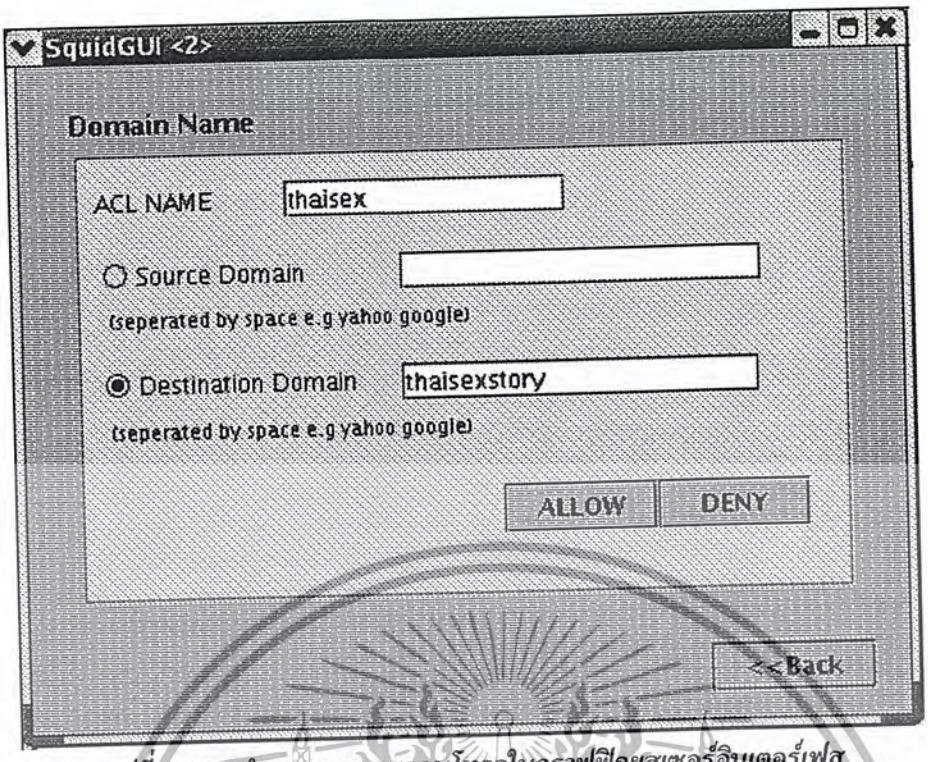
ผลที่ได้คือสามารถเขียนคำสั่งลงไปยังไฟล์ Squid.conf ได้อย่างถูกต้อง




*รูปที่ 7-3 ผลการวิเคราะห์การเขียนคำสั่งในไฟล์*

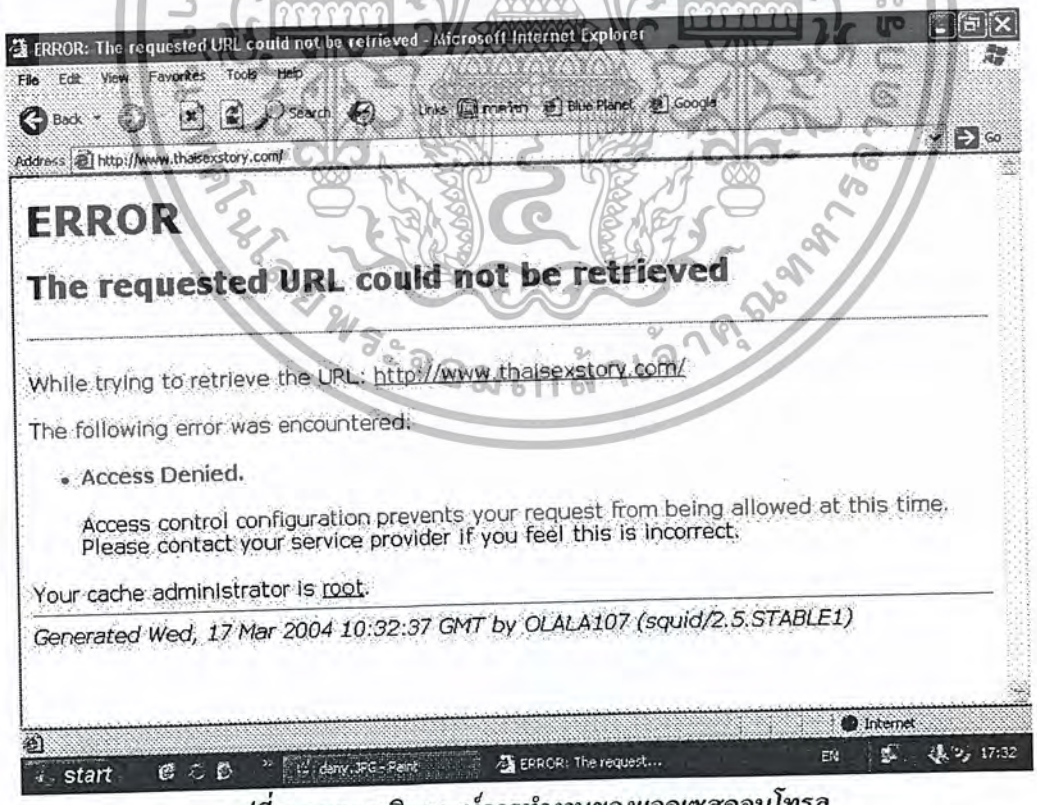
กรณีทดสอบที่ 2

ทดสอบการทำงานของการกำหนดแอคเซสคอนโทรล(Access Control) โดยการไม่ให้ผู้ใช้เข้าใช้งานเว็บ http://www.thaisexstory.com ได้




*รูปที่ 7-4 การกำหนดแอคเซสคอนโทรลในกราฟฟิกยูสเซอร์อินเตอร์เฟส*

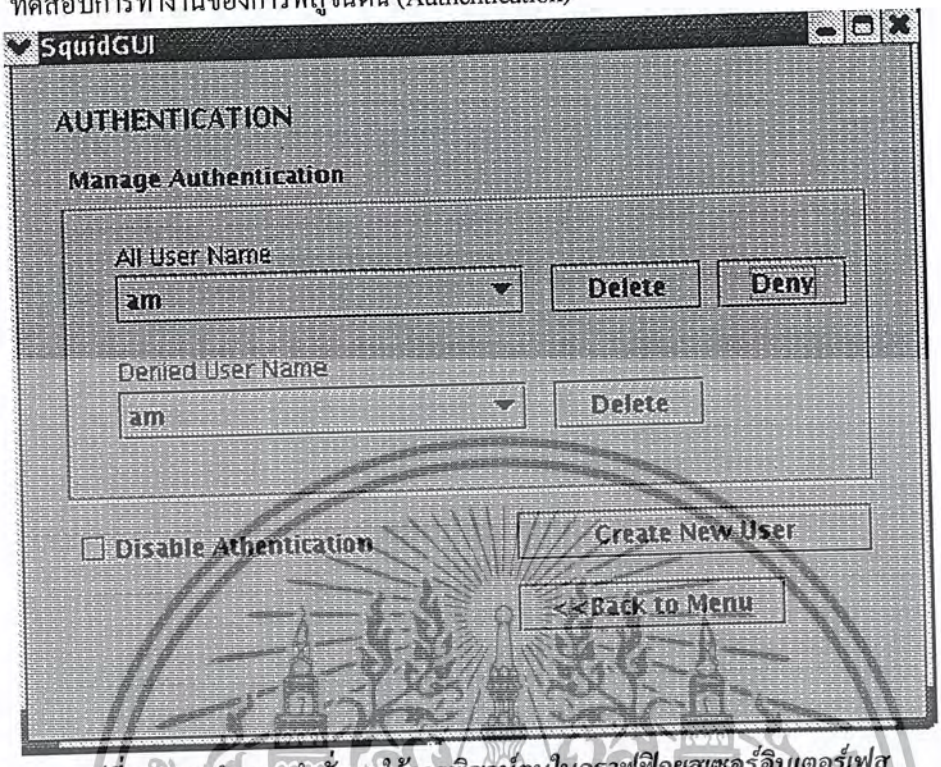
ผลที่ได้คือสามารถปฏิเสธการเชื่อมต่อไปยัง http://www.thaisexstory.com ได้




*รูปที่ 7-5 ผลการวิเคราะห์การทำงานของแอคเซสคอนโทรล*

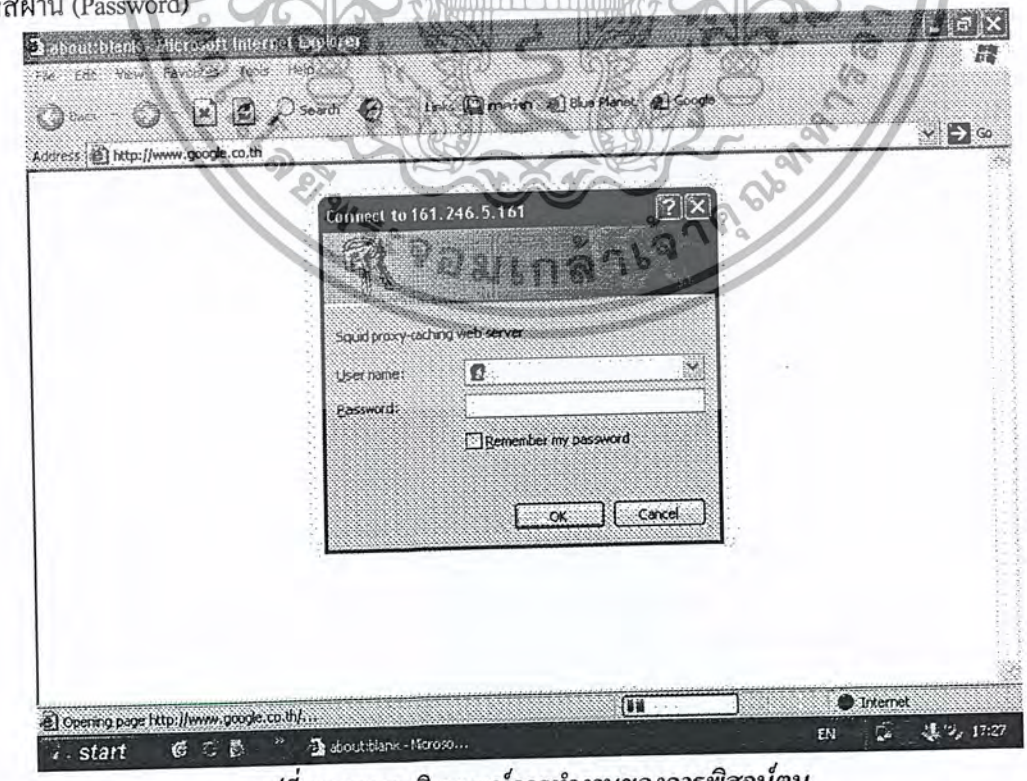
กรณีทดสอบที่ 3

ทดสอบการทำงานของการพิสูจน์ตน (Authentication)




*รูปที่ 7-6 การกำหนดคำสั่งการใช้การพิสูจน์ตนในกราฟฟิกยูสเซอร์อินเตอร์เฟส*

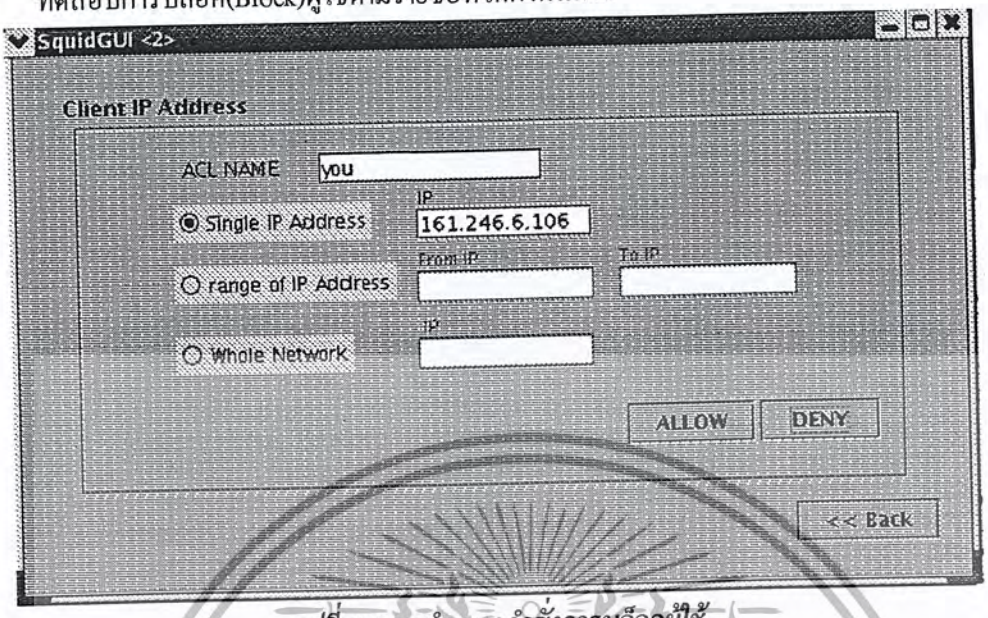
ผลที่ได้คือสามารถใช้การพิสูจน์ตนได้ โดยจะมีป๊อปอัพ (Pop-up) ขึ้นมาถามชื่อผู้ใช้ (User)และรหัสผ่าน (Password)




*รูปที่ 7-7 ผลการวิเคราะห์การทำงานของการพิสูจน์ตน*

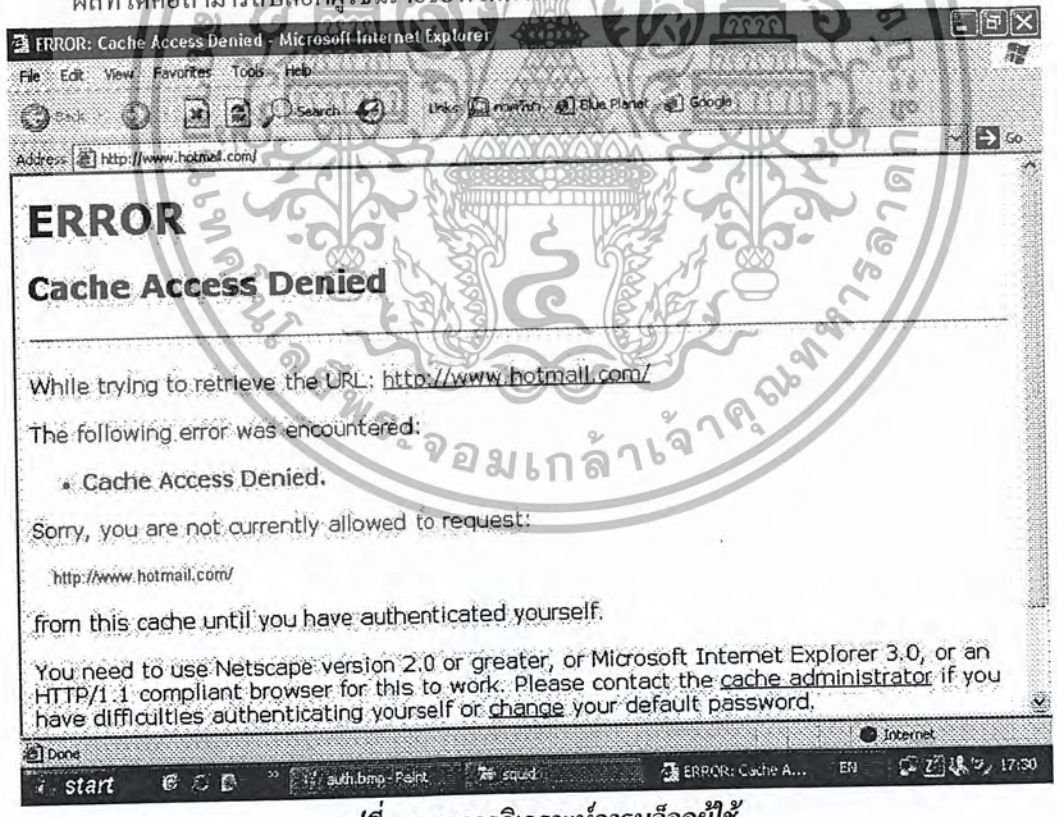
กรณีทดสอบที่ 4

ทดสอบการบล็อค(Block)ผู้ใช้ตามรายชื่อที่ได้กำหนดไว้




*รูปที่ 7-8 การกำหนดคำสั่งการบล็อกผู้ใช้*

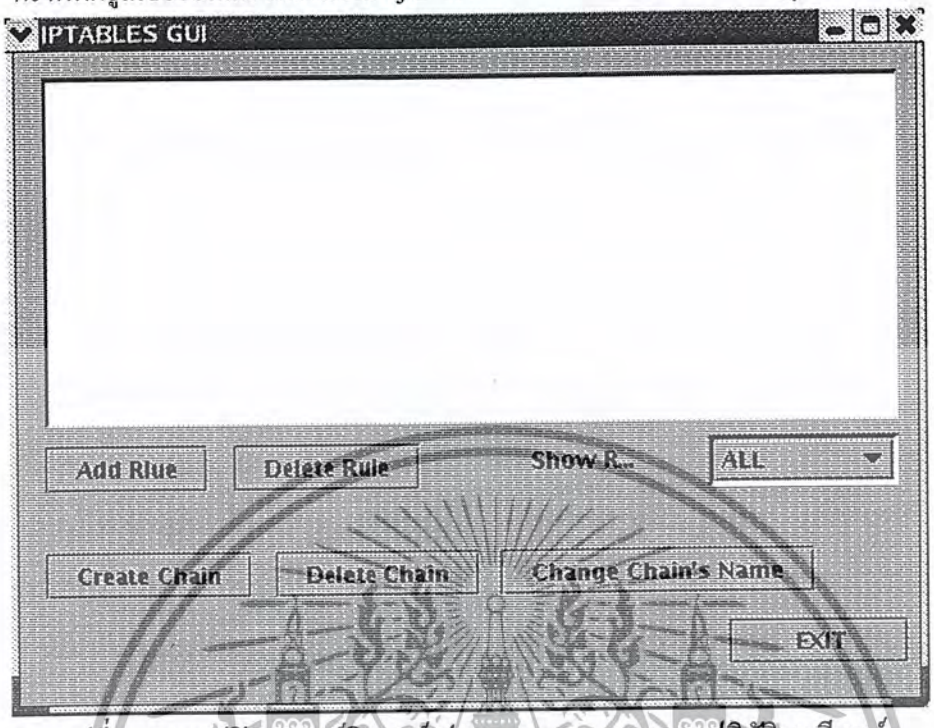
ผลที่ได้คือสามารถบล็อกผู้ใช้มีรายชื่อที่ได้กำหนดไว้




*รูปที่ 7-9 ผลการวิเคราะห์การบล็อคผู้ใช้*

**7.2 การทดสอบการทำงานของกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Iptables**

กราฟฟิกยูสเซอร์อินเตอร์เฟสที่เป็น ".jar" สามารถนำมาใช้งานบนระบบปฏิบัติการลีนุกซ์ได้

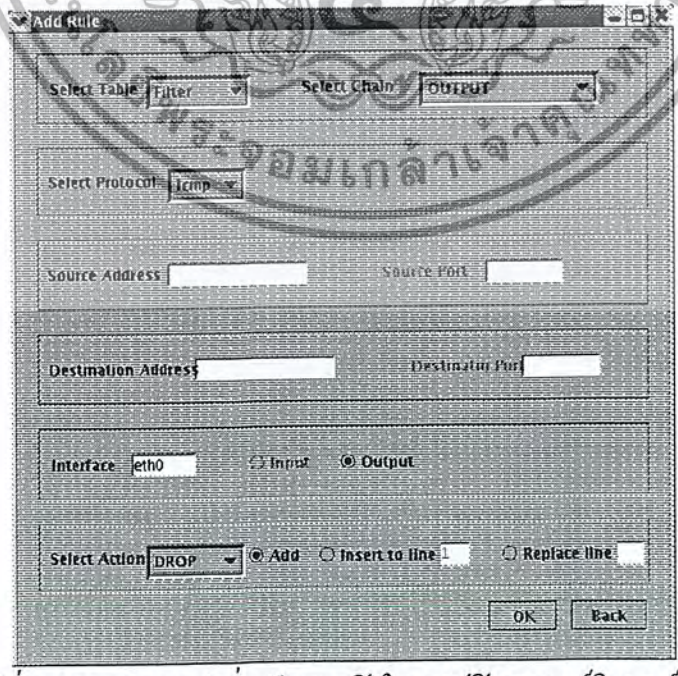



*รูปที่ 7-10 กราฟฟิกยูสเซอร์อินเตอร์เฟสของ Iptablesบนระบบปฏิบัติการลีนุกซ์*

ต่อมาได้นำกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Squid มาวิเคราะห์กรณีทดสอบดังต่อไปนี้

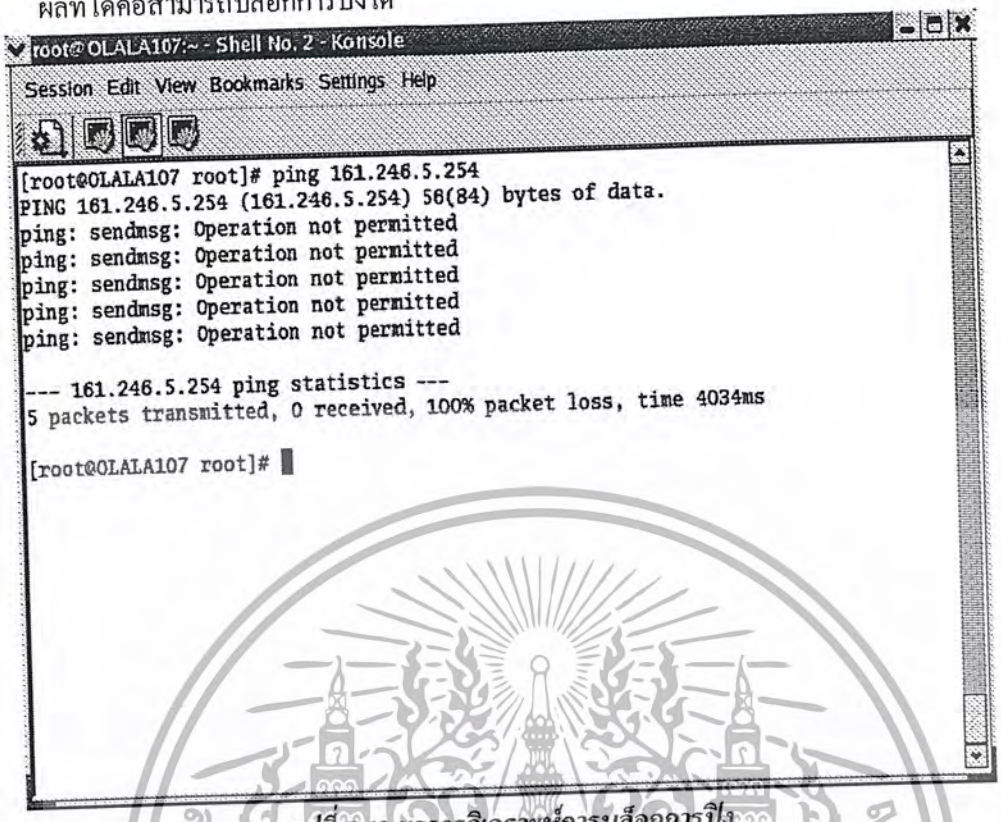
กรณีทดสอบที่ 1

ทดสอบการบล็อคการปิง (Ping)




*รูปที่ 7-11 การกำหนดคำสั่งบล็อกการปิงในกราฟฟิกยูสเซอร์อินเตอร์เฟส*

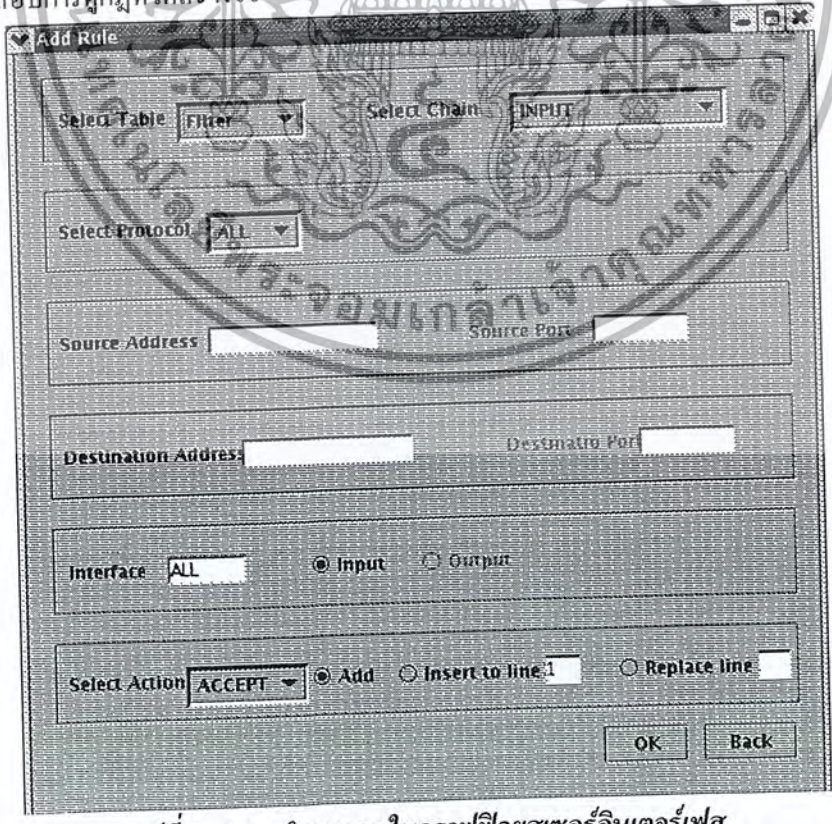
ผลที่ได้คือสามารถบล็อกการปิงได้




*รูปที่ 7-12 ผลการวิเคราะห์การบล็อกการปิง*

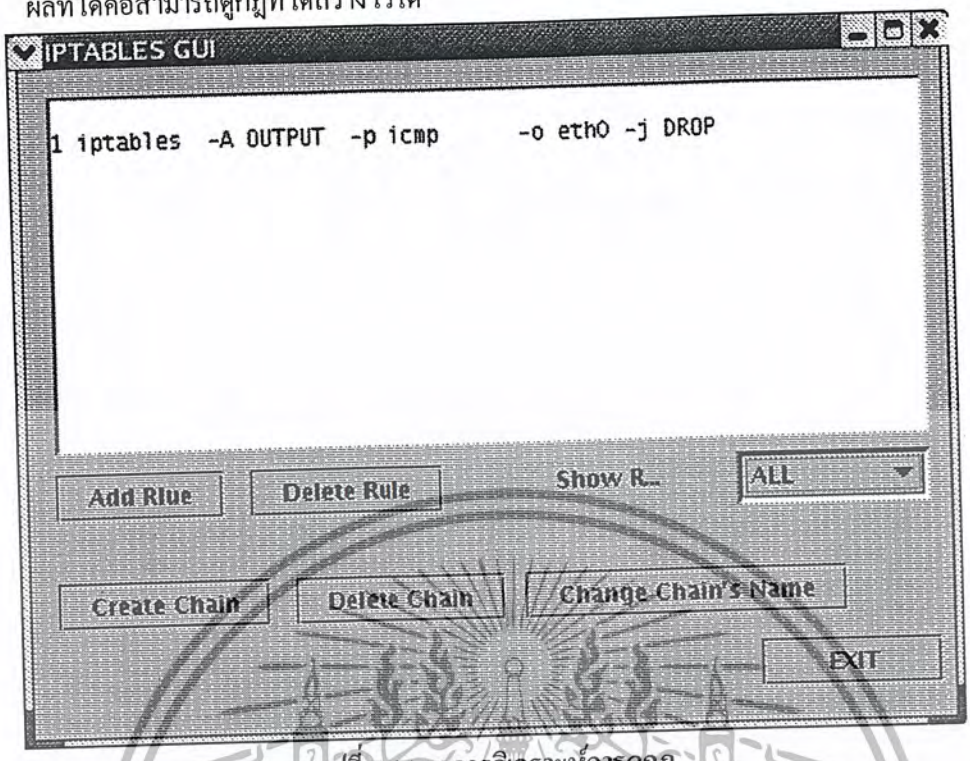
กรณีทดสอบที่ 2

ทดสอบการดูกฎที่ได้สร้างไว้




*รูปที่ 7-13 การกำหนดกฎในกราฟฟิกยูสเซอร์อินเตอร์เฟส*

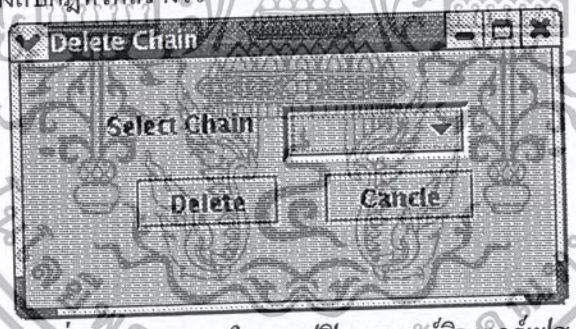
ผลที่ได้คือสามารถดูกฎที่ได้สร้างไว้ได้




*รูปที่ 7-14 ผลการวิเคราะห์การดูกฎ*

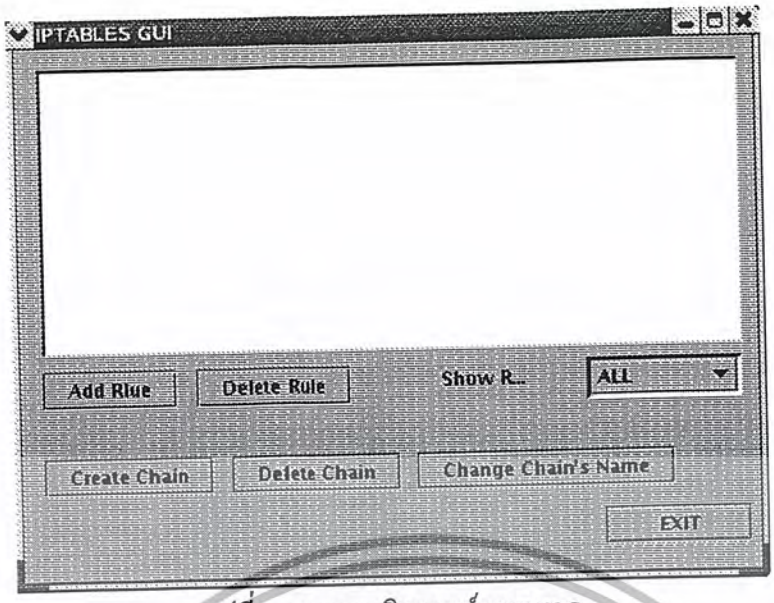
กรณีทดสอบที่ 3

ทดสอบการสร้างลบกฎที่ได้สร้างไว้




*รูปที่ 7-15 การลบกฎในกราฟฟิกยูสเซอร์อินเตอร์เฟส*

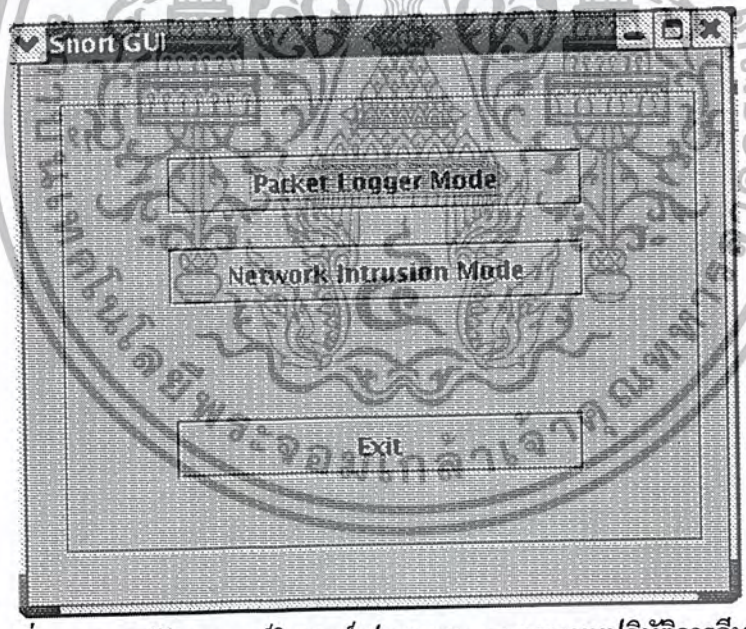
ผลที่ได้คือสามารถลบกฎที่ได้สร้างไว้ได้




*รูปที่ 7-16 ผลการวิเคราะห์การลบกฎ*

**7.3 การทดสอบการทำงานของกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Snort**

กราฟฟิกยูสเซอร์อินเตอร์เฟสที่เป็น "jar" สามารถนำมาใช้งานบนระบบปฏิบัติการลีนุกซ์ได้

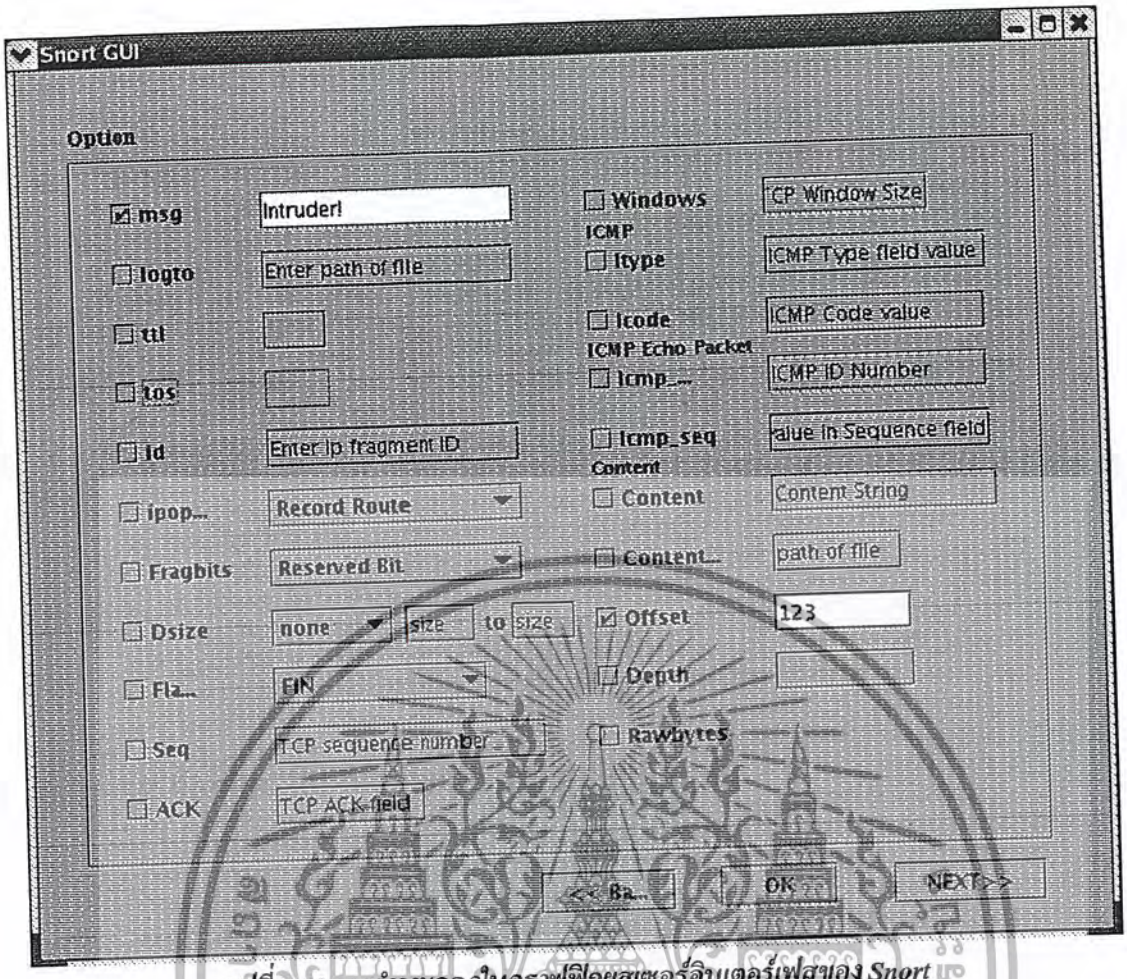



***รูปที่ 7-17 กราฟฟิกยูสเซอร์อินเตอร์เฟสของ Snortบนระบบปฏิบัติการลีนุกซ์***

ต่อมาได้นำกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Snort มาวิเคราะห์กรณีทดสอบดังต่อไปนี้

กรณีทดสอบที่ 1

ทดสอบการกำหนดกฎ




*รูปที่ 7-18 การกำหนดกฎในกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Snort*

ผลที่ได้คือสามารถเขียนกฎที่ผู้ใช้ต้องการเข้าไปในไฟล์ได้


*รูปที่ 7-19 ผลการวิเคราะห์การกำหนดกฎ*

# บทที่ 8
# บทวิจารณ์และสรุป

## 8.1 บทวิจารณ์และสรุป

กราฟฟิกยูสเซอร์อินเตอร์เฟสสำหรับไฟร์วอลล์และระบบตรวจจับผู้บุกรุกนั้น ทำให้ผู้ใช้สามารถกำหนดคำสั่งที่ตนเองต้องการ โดยไม่ต้องทราบข้อกำหนดในการเขียนคำสั่งใดๆ มีส่วนที่จะเสริมความรู้ทางด้านเน็ตเวิร์ก (Network) , หลักการทำงานของไฟร์วอลล์แบบต่างๆและระบบตรวจจับผู้บุกรุก และการใช้ทำงานบนระบบปฏิบัติการลีนุกซ์นั้น ทำให้ผู้ศึกษาได้ศึกษาระบบปฏิบัติการที่ต่างออกไป และวิธีการใช้งานต่างๆ ทำให้ผู้ศึกษามีความรู้เพิ่มมากขึ้น

## 8.2 ปัญหาที่พบ

1. หนังสือที่จำเป็นต้องใช้ในการศึกษาไม่มีเป็นภาษาไทย หนังสือที่เป็นภาษาอังกฤษมีราคาแพงมาก และบางเล่มก็ไม่สามารถหาได้ ต้องสั่งจากต่างประเทศเข้ามาเท่านั้น
2. เนื่องจากระบบปฏิบัติการลีนุกซ์เป็นระบบที่ผู้ศึกษา ไม่เคยใช้มาก่อน จึงต้องใช้เวลาในการศึกษาพอสมควร
3. ความไม่เชี่ยวชาญในการใช้ระบบปฏิบัติการลีนุกซ์ ทำให้เมื่อเกิดปัญหาขึ้นต้องใช้เวลาแก้ไขนาน
4. คู่มือการใช้โปรแกรม Squid , Iptables , Snort มีในอินเตอร์เน็ตเท่านั้น และคู่มือนั้นก็ไม่มีความละเอียดเพียงพอ บ้างก็พิมพ์ผิดหรือ ไม่ชัดเจน ทำให้เกิดปัญหาบ่อยมาก
5. ผู้เชี่ยวชาญในการใช้งานโปรแกรม Squid , Iptables , Snort หาได้ยากมาก ทำให้หาคำปรึกษาได้ยากมาก
6. โน้ตบุ้กของผู้ศึกษาไม่สามารถลงระบบปฏิบัติการลีนุกซ์ได้
7. เกิดปัญหาในระบบปฏิบัติการลีนุกซ์บ่อย ทำให้ผู้ศึกษาต้องลงระบบปฏิบัติการลีนุกซ์ใหม่เป็นจำนวนหลายรอบ

## 8.3 ข้อจำกัด

1. กฎ Iptables ที่กำหนดไว้เมื่อรีสตาร์ทเครื่องกฎจะหายไปทั้งหมด ต้องกำหนดกฎใหม่
2. กราฟฟิกยูสเซอร์อินเตอร์เฟสของ Squid ไม่สามารถลบกฎที่ได้กำหนดไว้ที่กราฟฟิกยูสเซอร์อินเตอร์เฟส
3. ในการกำหนดกฎใน Snort ผู้ใช้จำเป็นที่จะต้องรู้ว่าออปชันต่างๆใน Snort มีไว้ใช้เพื่ออะไร
4. กราฟฟิกยูสเซอร์อินเตอร์เฟสของ Snort ไม่สามารถลบกฎที่ได้กำหนดไว้ที่กราฟฟิกยูสเซอร์อินเตอร์เฟส

**8.4 แนวทางในการพัฒนาต่อ**

1. ปรับปรุงกราฟฟิกยูสเซอร์อินเตอร์เฟสที่มีความสวยงามกว่านี้
2. ควรพัฒนาให้ไม่ต้องกำหนดกฎ Iptables ทุกครั้งที่มีการรีสตาร์ทเครื่อง
3. ควรพัฒนาให้ผู้ใช้สามารถลบคำสั่งที่ผู้ใช้ได้กำหนดไว้ที่กราฟฟิกยูสเซอร์อินเตอร์เฟสได้

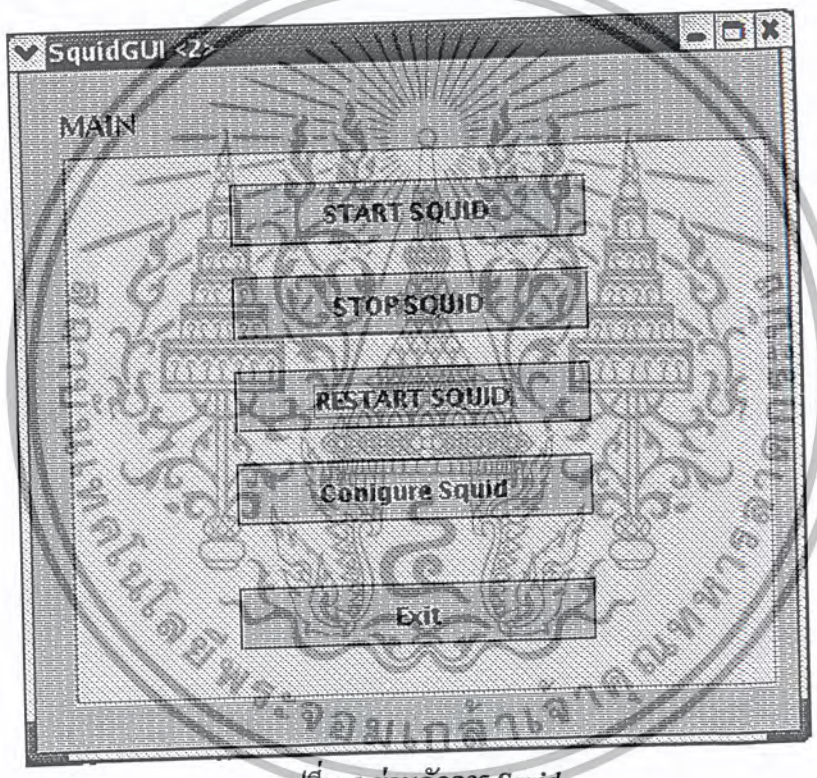
# ภาคผนวก ก.

# วิธีการใช้งานกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Squid

Squid GUI ประกอบไปด้วย 2 ส่วนได้แก่

**1. ส่วนจัดการกับ Squid ประกอบไปด้วย**

- Start Squid ให้ทำงาน
- Stop Squid ให้หยุดทำงาน
- Restart Squid ให้ Squid หยุดทำงานแล้วเริ่มต้นการทำงานใหม่
- Configure Squid ให้ไปส่วนเซ็ตค่าต่างๆ ใน Squid



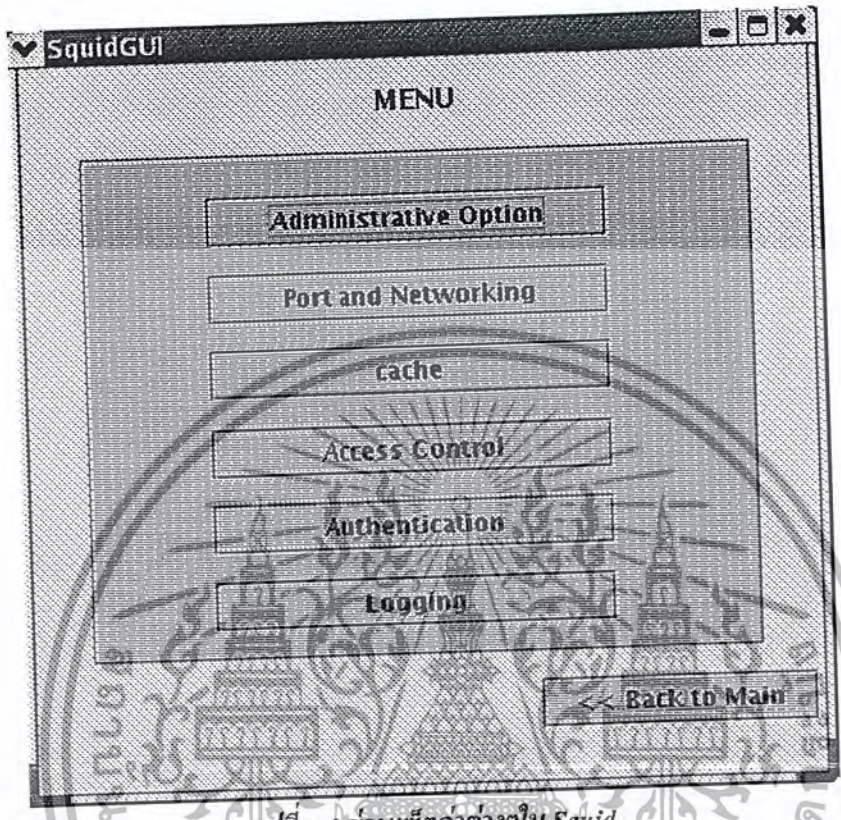

*รูปที่ ก-1 ส่วนจัดการ Squid*

**2. ส่วนเซ็ตค่าต่างๆใน Squid (Configure Squid)**

เมื่อกดปุ่ม Configure Squid ในส่วนจัดการกับ Squid จะมีรายการต่างๆให้ผู้ใช้สามารถเซ็ตค่าได้ แบ่งเป็น

- Administrative Option เป็นส่วนจัดการของผู้ควบคุมระบบ
- Port and Networking จะเป็นส่วนจัดการเกี่ยวกับ Port ต่างๆและส่วนที่เกี่ยวข้องกับเน็ตเวิร์ก
- Cache เป็นส่วนจัดการเกี่ยวกับแคช

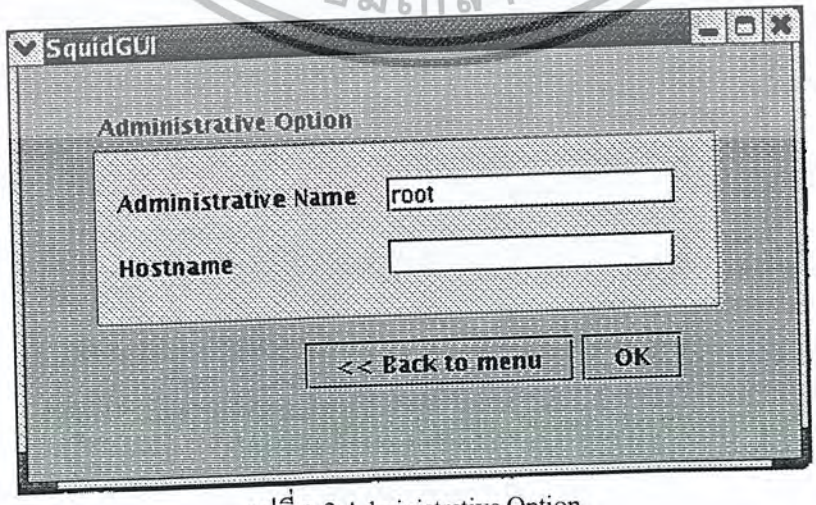
- Access Control เป็นส่วนจัดการเกี่ยวกับสิทธิ์ของผู้ใช้งาน
- Authentication เป็นส่วนจัดการเกี่ยวกับการพิสูจน์ตนของผู้ใช้งาน
- Logging เป็นส่วนจัดการเกี่ยวกับล็อก




*รูปที่ ก-2 ส่วนเซ็ตค่าต่างๆใน Squid*

**2.1 Adminstrative Option**

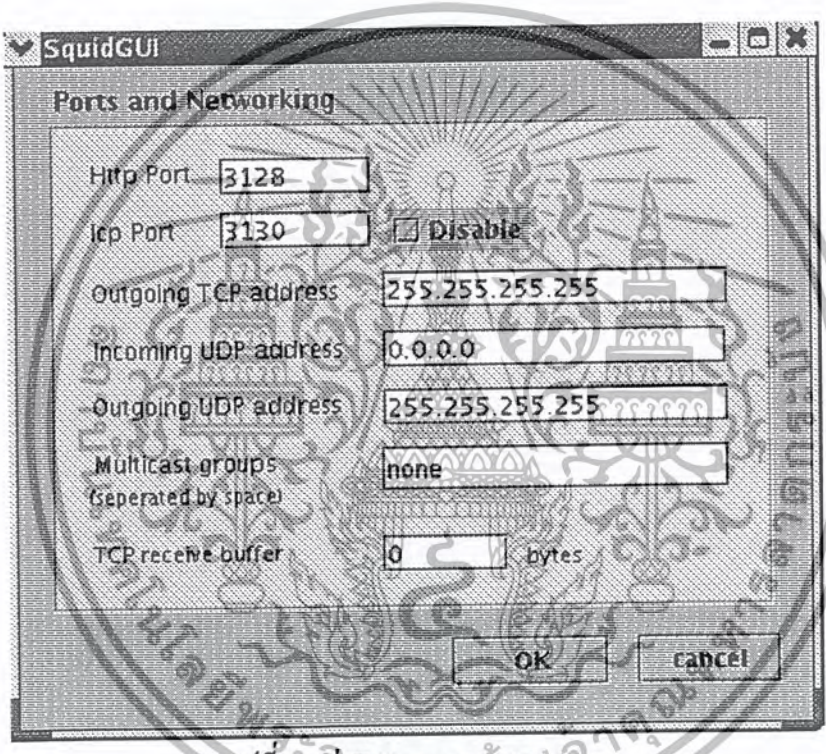
ให้ผู้ใช้กำหนดชื่อ และ hostname มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField




รูปที่ ก-3 Administrative Option

**2.2 Port and Networking ประกอบไปด้วย**

- Http Port ระบุ socket address ที่ให้ Squid listen สำหรับ HTTP client request
- Icp Port Port ที่ Squid ส่งและรับ ICP queries ที่ส่งไปและรับมาจากแคชเพื่อนบ้าน
- Outgoing TCP Address ใช้สำหรับการสื่อสารที่ถูกส่งไปยัง remote server
- Incoming UDP Addressใช้สำหรับ ICP socket เพื่อรับแพ็คเก็ตจากแคชอื่นๆ
- Outgoing UDP Address ใช้สำหรับ ICP socket เพื่อส่งแพ็คเก็ตไปแคชอื่นๆ
- Multicast group Multicast group ที่ server จะร่วมด้วยเพื่อรับ ICP queries ที่ Multicast ออกไป
- TCP receive buffer



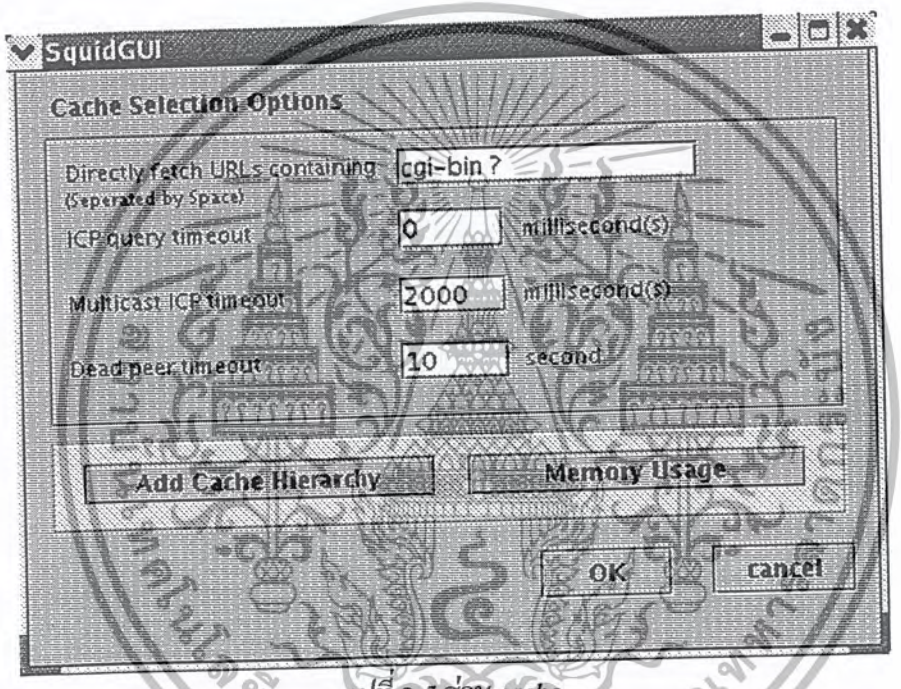

*รูปที่ ก-4 ส่วน Port and Networking*

โดยจะมีค่า Default ที่ทาง Squid ตั้งไว้ให้แล้ว สามารถใช้ค่าเหล่านี้ได้เลย สำหรับส่วนของ Icp Port สามารถ ไม่ใช้งาน (Disable)ได้ ส่วนนี้มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField

**2.3 Cache ประกอบไปด้วย**

- Directly fetch URLs containing กำหนดคำที่มีอยู่ใน URL ให้ถูกจัดการโดยแคชนี้โดยตรง
- ICP query timeout กำหนดค่า timeout ของการทำ ICP query
- Multicast ICP timeout กำหนดค่า timeout ของการ Multicast ICPDead peer
- Timeout กำหนดค่า timeout สำหรับแคชที่ติดต่อด้วย ถ้าไม่มีการตอบกลับ ICP จะถือว่าแคชที่ติดต่อด้วยนั้น dead

ส่วนนี้มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField




*รูปที่ ก-5 ส่วน cache*

และในส่วนของ cache นี้ยังแบ่งออกเป็น 2 ส่วนย่อยได้แก่

- ส่วน Cache Hierarchy (Add Cache Hierarchy)
- ส่วนจัดการ Memory (Memory Usage)

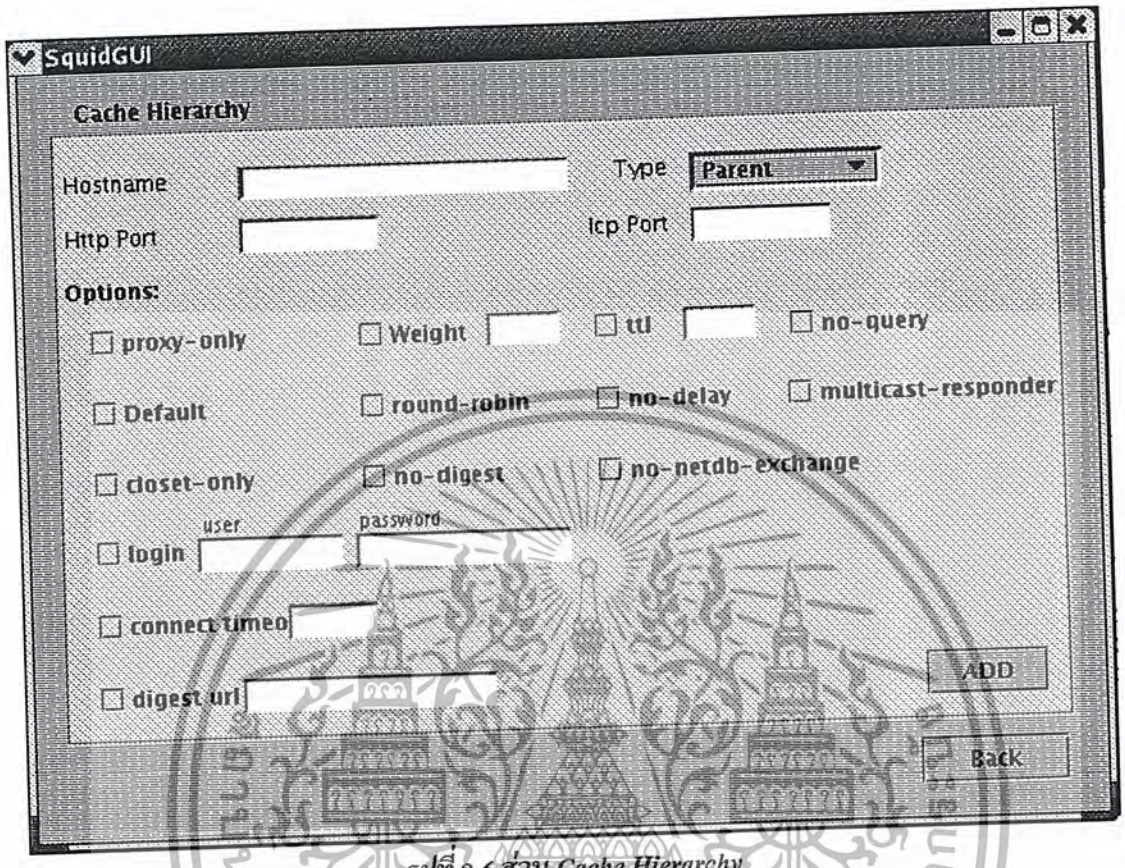
ส่วนนี้มีความประสงค์เพื่อให้ผู้ใช้เลือกทำรายการดังนั้นจึงใช้ปุ่มกด(button)

**2.3.1 ส่วน Cache Hierarchy (Add Cache Hierarchy)**

ให้ผู้ใช้กำหนด Cache Hierarchy

- ส่วน Hostname, Http port, Icp port มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField

- ส่วน Type ผู้ใช้สามารถเลือกได้ 3 แบบเท่านั้นจึงใช้ ComboBox
- ส่วน Option ผู้ใช้สามารถเลือกได้หลาย Option ดังนั้นจึงใช้ checkbox




*รูปที่ ก-6 ส่วน Cache Hierarchy*

**2.3.2 ส่วนจัดการ Memory (Memory**

- ส่วนMemory usage limit , FQDN cache
- size, Memory high-low water mark, Maximum Cache Object size, IP address cache

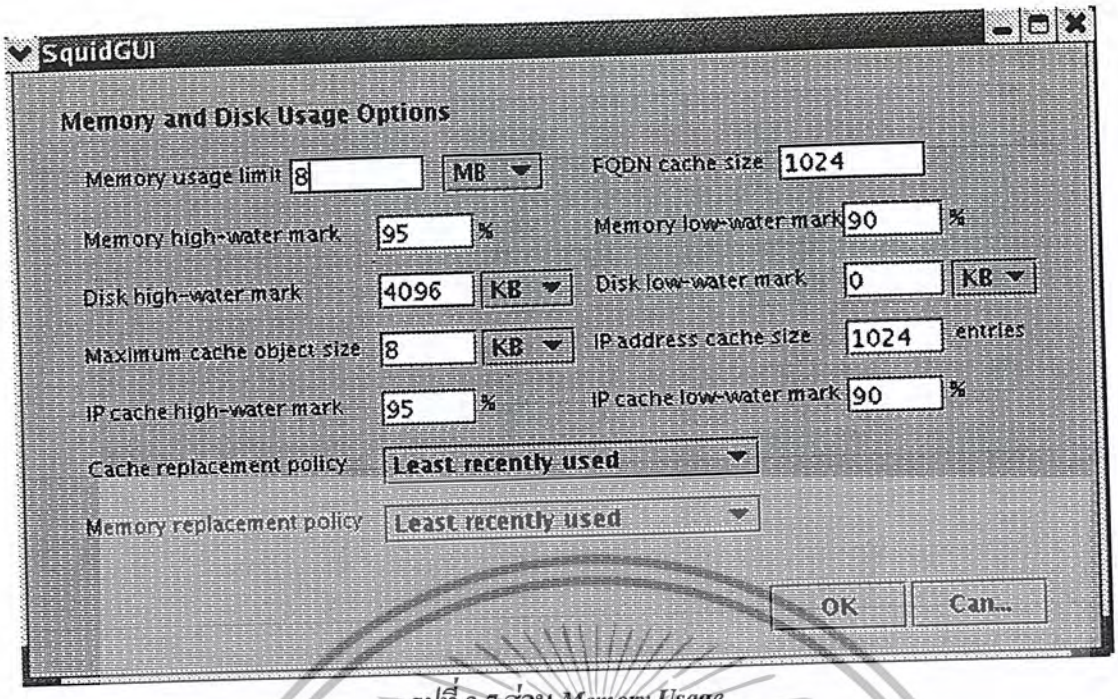
size , IP cache high-low water mark มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField

- ส่วนหน่วยของ Memory usage limit , Disk High-low watermark , maximum cache

object size สามารถกำหนดค่าได้ 3 แบบเท่านั้นจึงใช้ ComboBox

- ส่วนของ Disk-Momory replacement Policy สามารถกำหนดค่าได้ 4 แบบเท่านั้นจึงใช้

ComboBox




*รูปที่ ก-7 ส่วน Memory Usage*

2.4 ส่วน Access Control แบ่งเป็น 2 ส่วนย่อย ได้แก่

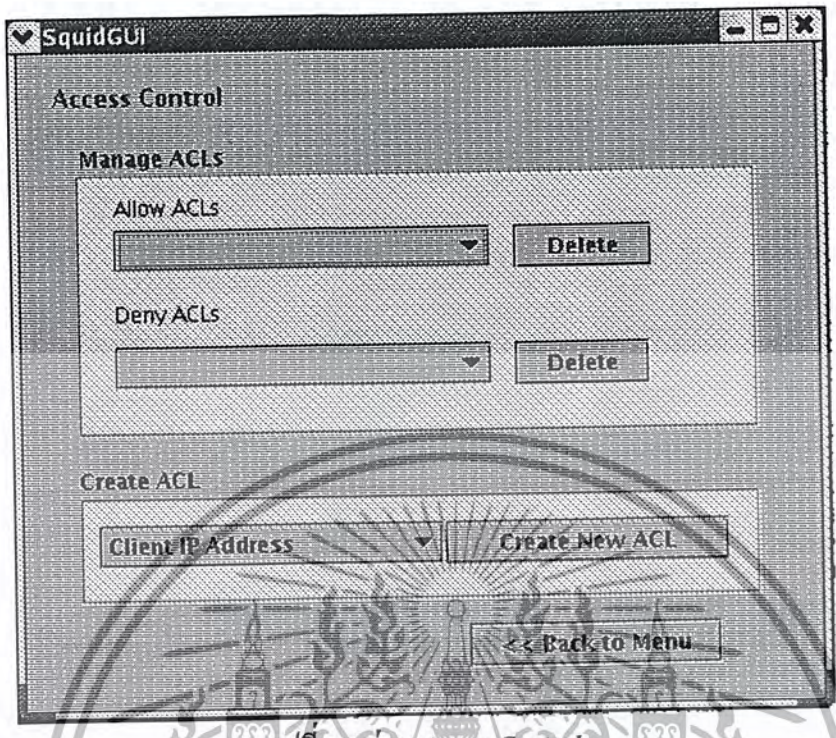
- ส่วนจัดการ Access Control list (Manage ACLs)

เป็นส่วนที่ให้ผู้ใช้สามารถลบกฎที่ได้ตั้งเอาไว้โดยจะแบ่งไว้เป็นกฎที่อนุญาต(Allow) กับกฎที่ไม่อนุญาต (Deny) โดยจะให้เลือกตามที่ผู้ใช้ได้กำหนดกฎเอาไว้ดังนั้นจึงใช้ ComboBox

- ส่วนเพิ่ม Access Control list (Create ACL)

เป็นส่วนที่ให้ผู้ใช้เพิ่มกฎที่ต้องการ และผู้ใช้สามารถเลือกสร้างกฎได้ 11 แบบดังนั้นจึงใช้ ComboBox

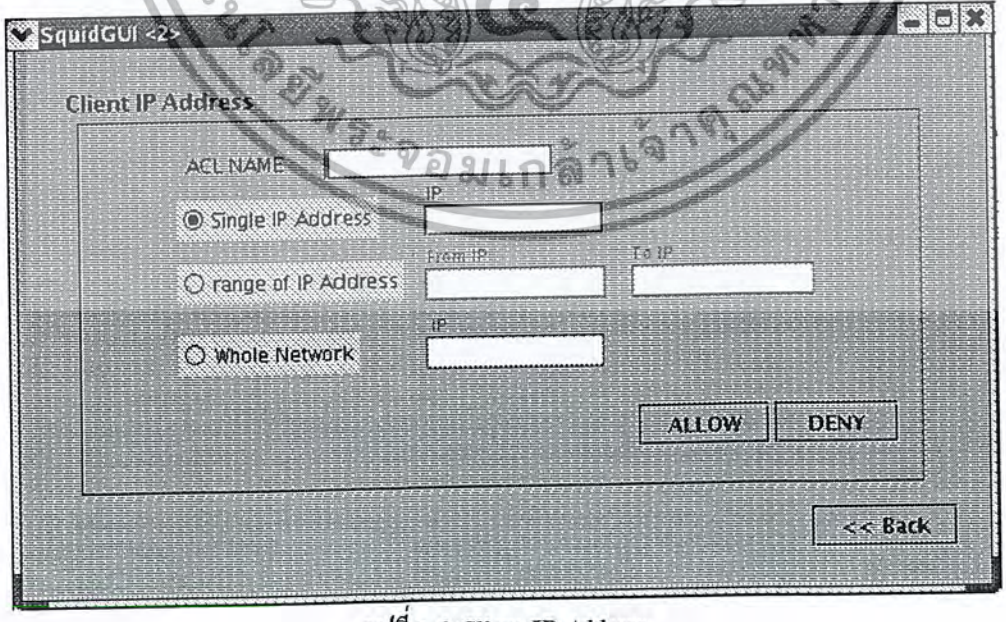
### 2.4.1 ส่วนเพิ่ม Access Control list (Create ACL)




*รูปที่ ก-8 ส่วน Access Control*

#### 2.4.1.1 Client IP Address

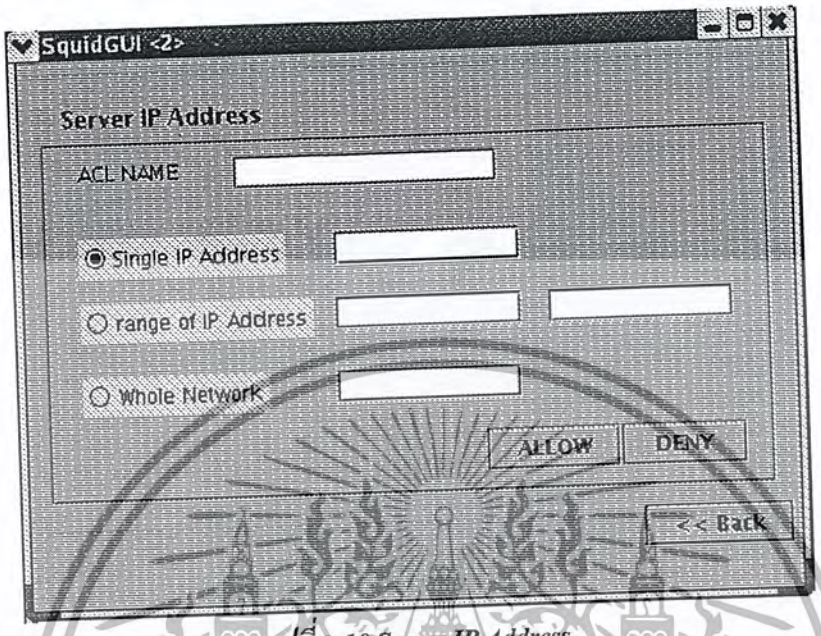
กำหนด IP ของ Client แบ่งได้เป็น 3 แบบ และสามารถเลือกได้ทีละแบบเท่านั้นจึงใช้ radiobox




*รูปที่ ก-9 Client IP Address*

**2.4.1.2 Server IP Address**

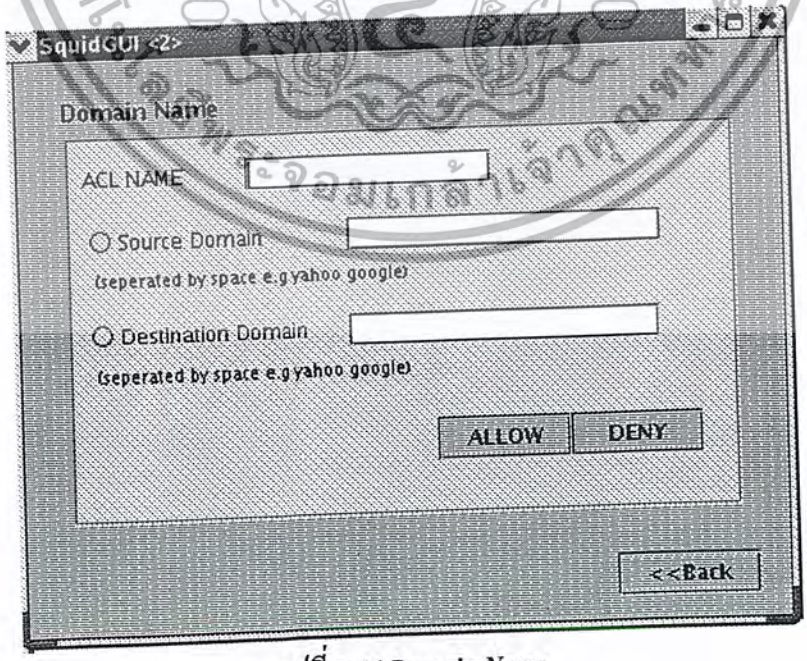
กำหนด IP ปลายทาง มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField




*รูปที่ ก-10 Server IP Address*

**2.4.1.3 Domain Name**

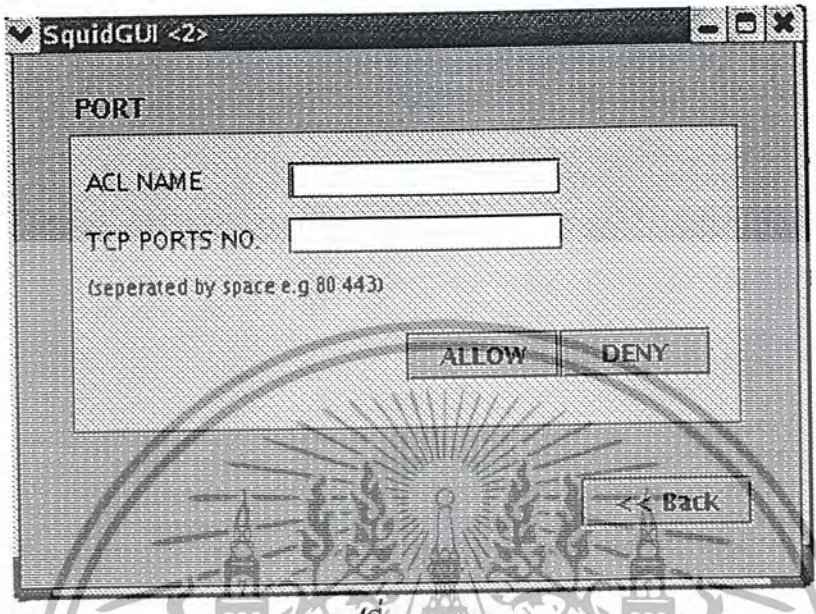
กำหนด domain name สามารถเลือกเป็น source หรือ Destination แต่เลือกได้ทีละแบบจึงใช้ radiobutton




*รูปที่ ก-11 Domain Name*

**2.4.1.4 Port**

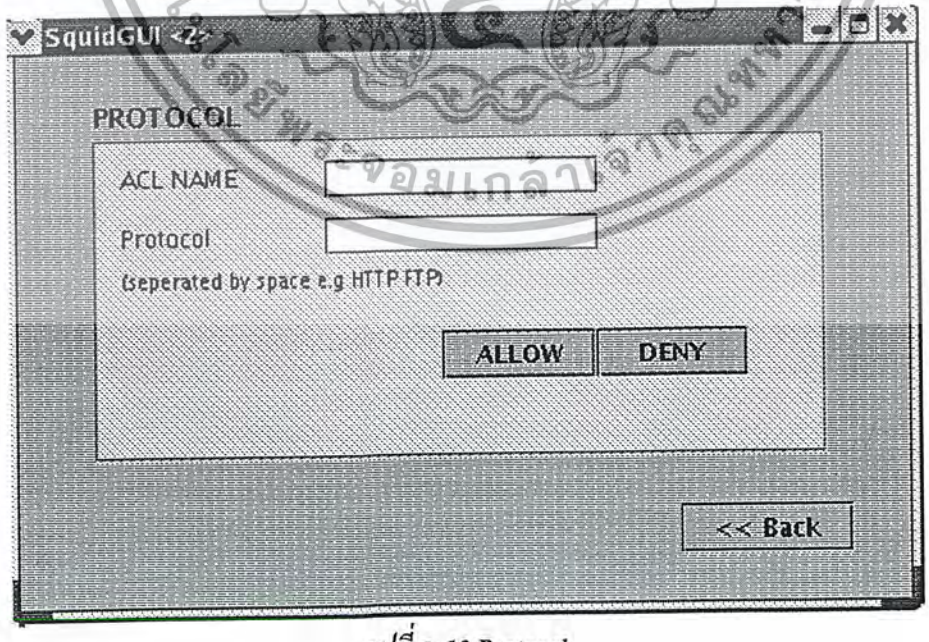
ให้ผู้ใช้กำหนด หมายเลข port มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField




*รูปที่ ก-12 Port*

**2.4.1.5 Protocol**

ให้ผู้ใช้กำหนดชนิด protocol มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField

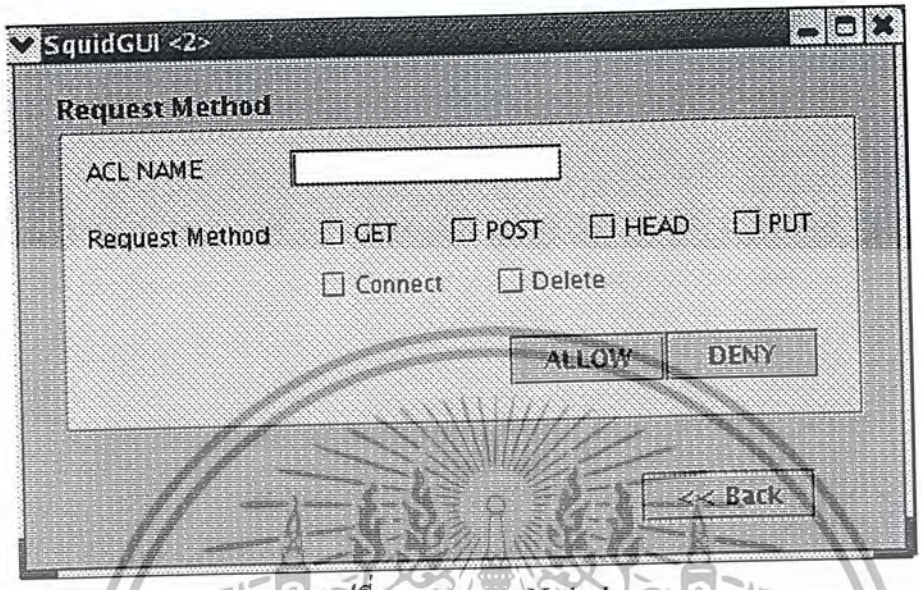



*รูปที่ ก-13 Protocol*

**2.4.1.6 Request Method**

ให้ผู้ใช้กำหนดชนิด Request Method สามารถเลือกได้หลายแบบจึงใช้

CheckBox

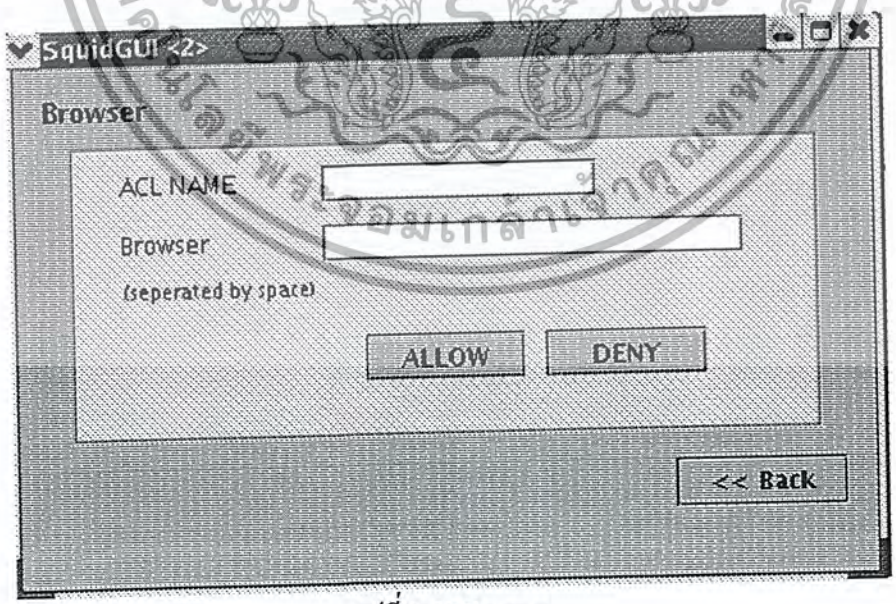



*รูปที่ ก-14 Request Method*

**2.4.1.7 Browser**

ให้ผู้ใช้กำหนดชนิด browser มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไป

จึงใช้ TextField

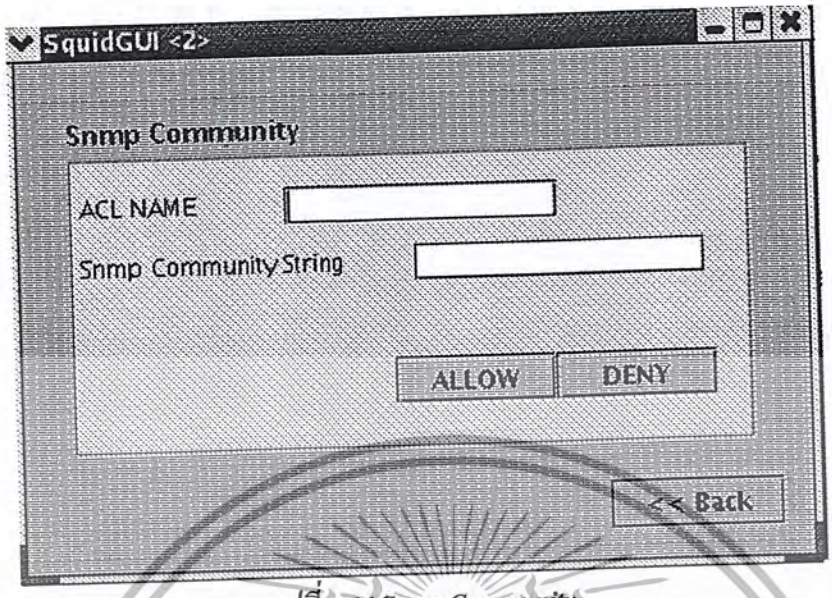



*รูปที่ ก-15 Browser*

**2.4.1.8 snmp Community**

ให้ผู้ใช้กำหนดค่า Snmp Community มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่

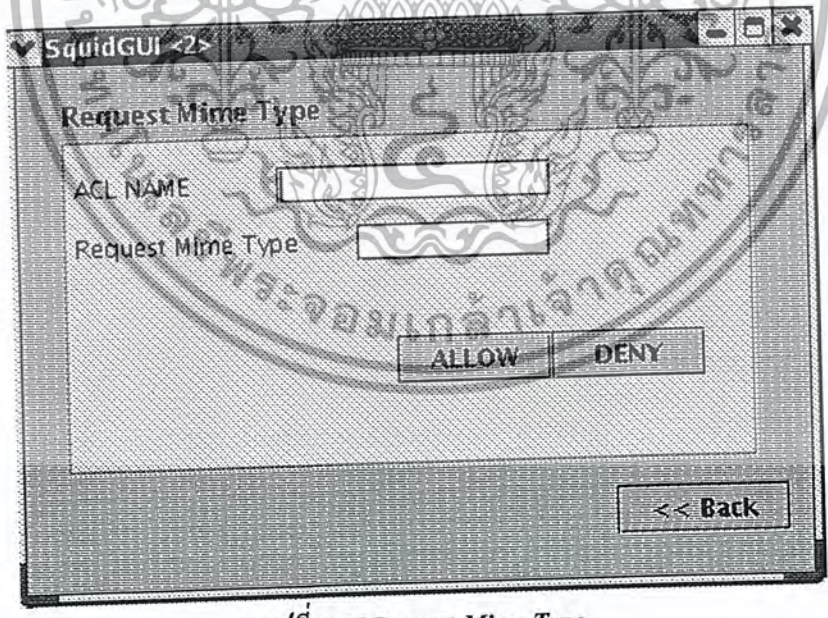
ต้องการลงไปจึงใช้ TextField




*รูปที่ ก-16 Snmp Community*

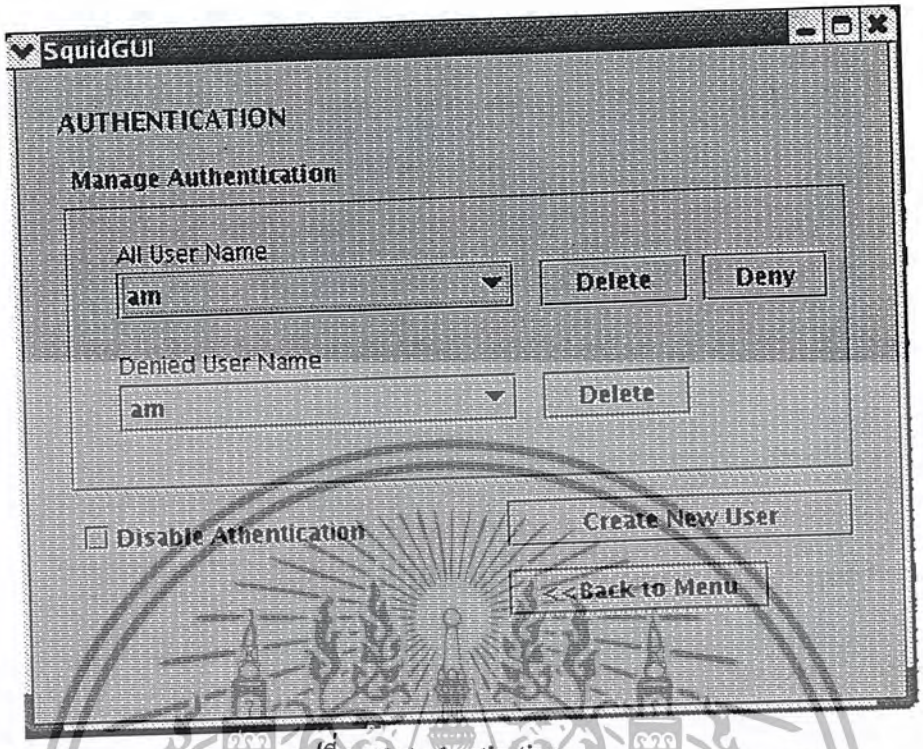
**2.4.1.9 request Mime Type**

ให้ผู้ใช้กำหนดค่า Snmp Community มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField



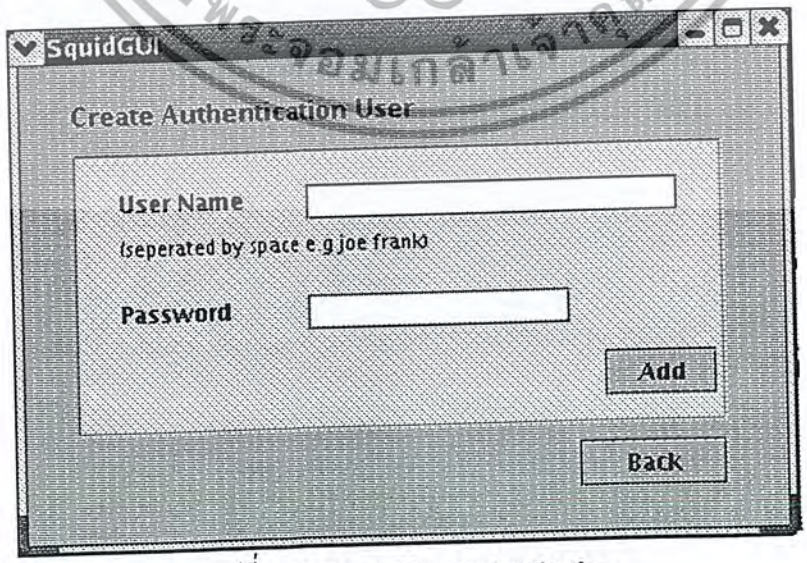

*รูปที่ ก-17 Request Mime Type*

2.5 ส่วน Authentication แบ่งเป็น 2 ส่วนได้แก่




*รูปที่ ก-18 Authentication*

- ส่วนจัดการ Authentication (Manage Authentication) เป็นส่วนที่ให้ผู้ใช้สามารถลบ username ที่ได้กำหนดไว้ สามารถเลือกได้จาก username ที่มีอยู่ จึงใช้ ComboBox
- เพิ่ม Authentication (Create New Authentication) เป็นส่วนที่ให้ผู้ใช้เพิ่ม username ที่ใช้ในการ Authentication มีความประสงค์เพื่อให้ผู้ใช้ใส่ข้อมูลที่ต้องการลงไปจึงใช้ TextField

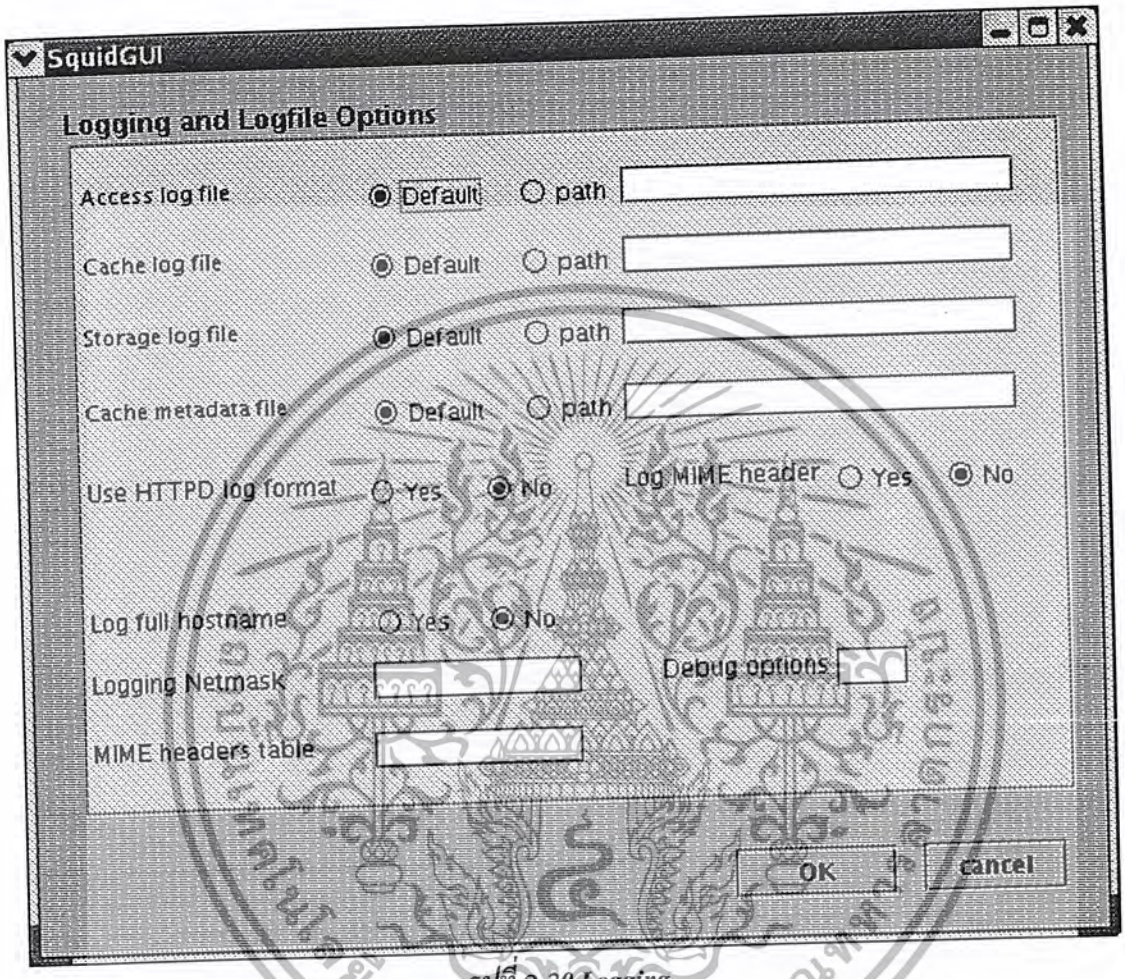



*รูปที่ ก-19 Create New Authentication*

**2.6 ส่วน Logging กำหนดข้อมูลการ log ต่างๆ**

ข้อมูลของ file การ log ต่างๆสามารถเลือกได้จาก default ซึ่งมีอยู่แล้ว หรือจะกำหนดเอง ดังนั้นจึงใช้ radio button

ข้อมูลการ log ประเภทให้เลือก yes หรือ no เท่านั้นจึงใช้ radio button




*รูปที่ ก-20 Logging*

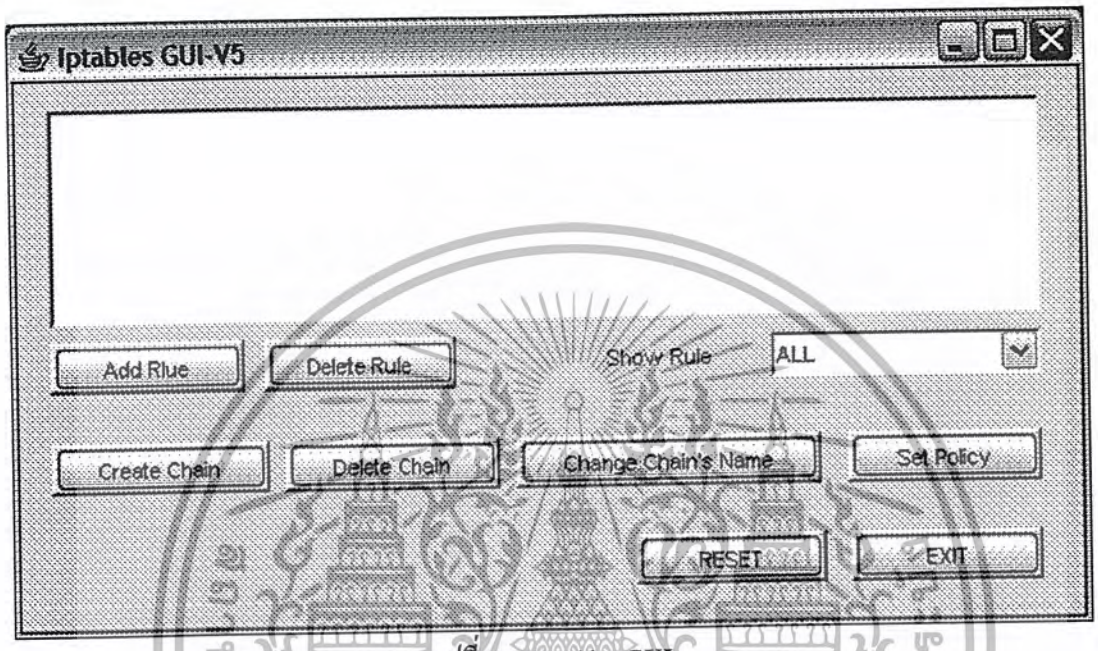
**Squid Graphic User Interface(GUI) Analysis**

จาก GUI ที่อธิบายมาข้างต้นสามารถทำงานแทน script ได้ทุกประการ

# ภาคผนวก ข.

# วิธีการใช้งานกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Iptables

## 1. Iptables GUI Interface



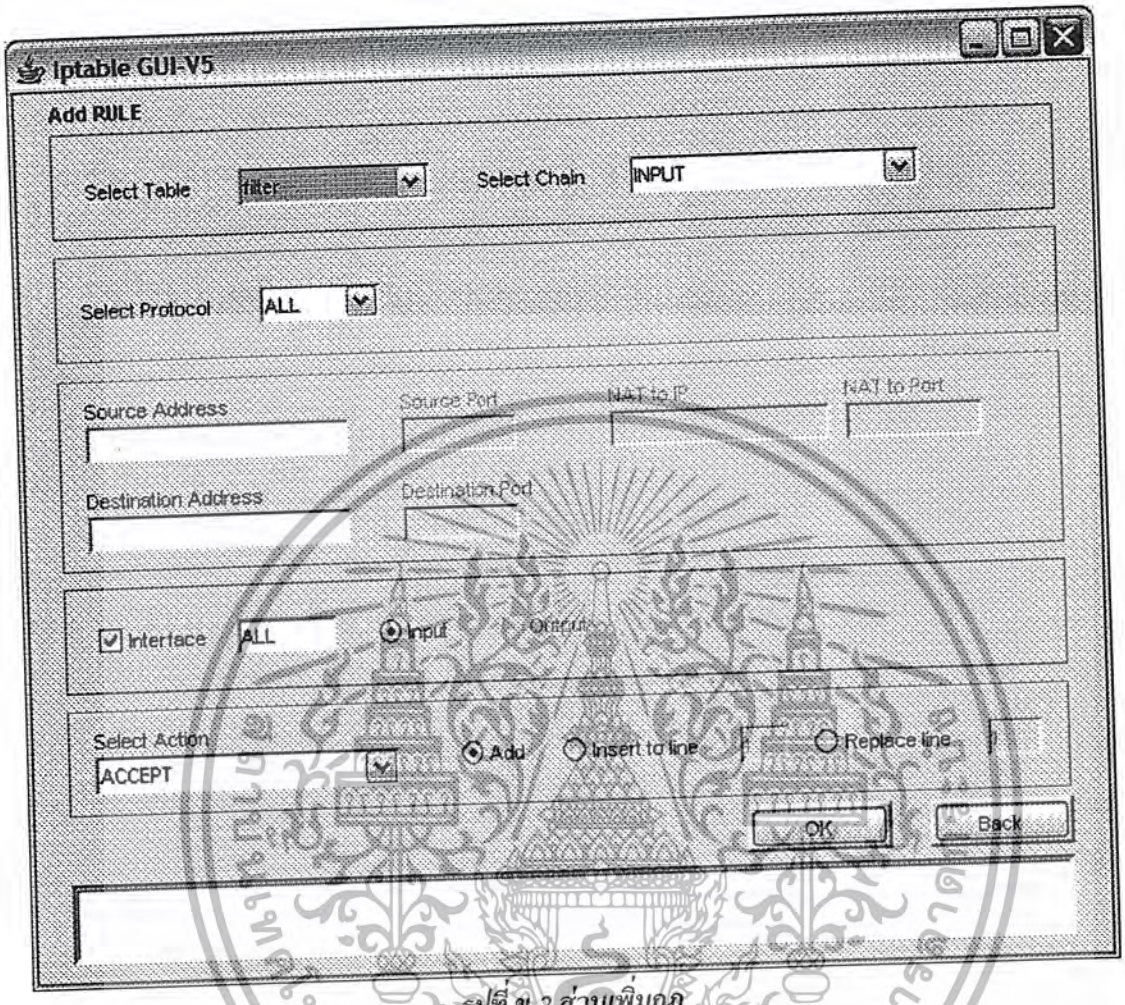

*รูปที่ ข-1 Iptables GUI*

TextArea จะเป็นพื้นที่แสดง Rule ที่ได้ set ไว้ซึ่งจะใช้ ComboBox ซึ่งประกอบไปด้วย

- ALL เป็นการแสดง Rule ทั้งหมด
- INPUT เป็นการแสดงเฉพาะ Rule ที่ Input chain
- OUTPUT เป็นการแสดงเฉพาะ Rule ที่ Output chain
- Forward เป็นการแสดงเฉพาะ Rule ที่ Forward chain
- PREROUTING เป็นการแสดงเฉพาะ Rule ที่ Prerouting chain
- POSTROUTING เป็นการแสดงเฉพาะ Rule ที่ Input chain

2. ส่วนเพิ่มกฎ

เมื่อต้องการเพิ่ม Rule สามารถกดปุ่ม ADD ซึ่งจะได้เป็น Frame ลักษณะดังนี้

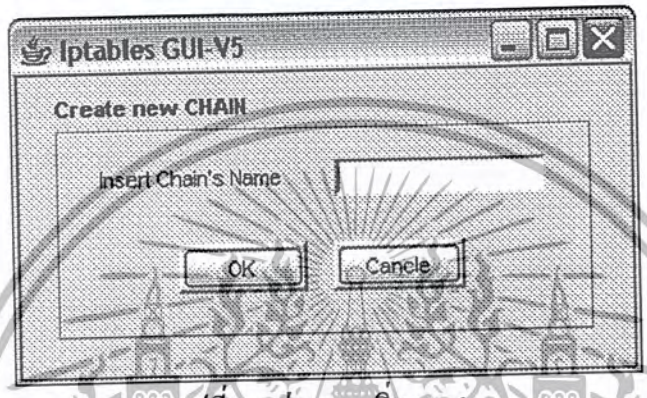



*รูปที่ ข-2 ส่วนเพิ่มกฎ*

- สามารถเลือก TABLE ได้ 3 TABLE ซึ่งประกอบไปด้วย Filter , Nat , Mangle ซึ่งสามารถเลือกได้ทีละประเภทเท่านั้น จึงใช้ ComboBox
- สามารถเลือก Built-in chain ได้ 5 chain ซึ่งประกอบไปด้วย Input , Output , Forward , Prerouting , Postrouting และ chain ที่ได้สร้างไว้เอง ซึ่งสามารถเลือกได้ทีละประเภทเท่านั้น จึงใช้ ComboBox
- สามารถเลือก Protocol ได้ 3 protocol ซึ่งประกอบไปด้วย ICMP , UDP และ TCP หรือจะเลือกทั้งหมด (ALL) ก็ได้ซึ่งสามารถเลือกได้ทีละประเภทเท่านั้น จึงใช้ ComboBox
- Source Address , Destination Address เป็น field ให้ใส่ IP Address
- Source Port , Destination Port เป็น field ให้ใส่หมายเลข Port ที่ user ต้องการ
- Interface เป็น field ให้ใส่ชื่อ Interface ซึ่งสามารถเลือกได้อย่างใดอย่างหนึ่งคือ Input หรือ Output จึงใช้ Radio Button

- สามารถเลือก Action ได้ 2 แบบคือ ACCEPT กับ DROP จึงใช้ ComboBox ซึ่ง Rule ที่เขียนเลือกได้ 3 แบบซึ่งเลือกได้อย่างใดอย่างหนึ่งจึงใช้ Radio Button คือ
- Add เป็นการต่อท้าย Rule ที่มีอยู่เดิม
- Insert to line เป็นการเพิ่มไปในบรรทัดที่ต้องการ
- Replace line เป็นการแทนที่บรรทัดที่ต้องการ

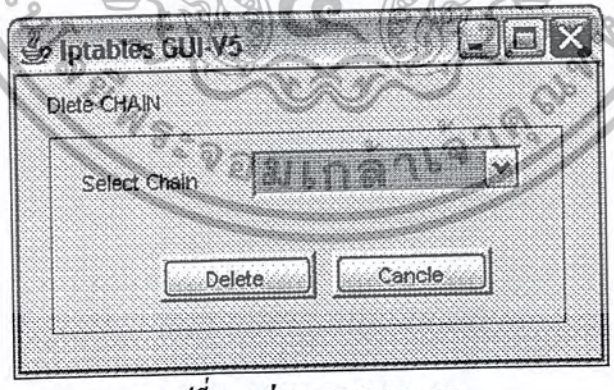
**3. ส่วนการ Create Chain**




*รูปที่ ข-3 ส่วนการเพิ่ม Chain*

ผู้ใช้สามารถสร้าง Chain ใหม่โดยสามารถตั้งชื่อ Chain ได้ตามต้องการ จึงใช้ TextField

**4. ส่วนการ Delete Rule**

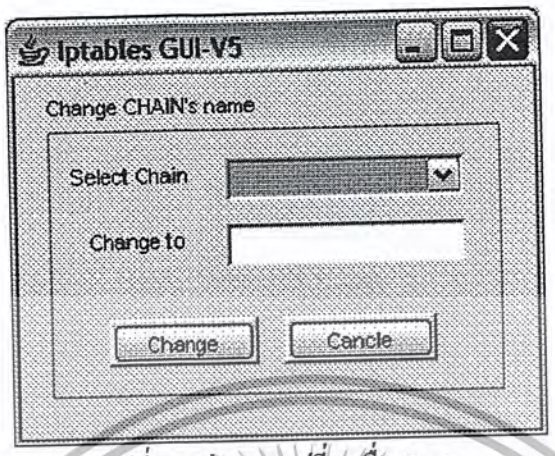



*รูปที่ ข-4 ส่วนการ Delete กฎ*

สามารถเลือกได้ว่าจะลบที่ Chain ไหนจึงใช้ ComboBox และ Delete Line เป็น field ให้ใส่บรรทัดที่ต้องการจะลบ โดยถ้าจะลบทั้งหมดให้ใส่ ALL

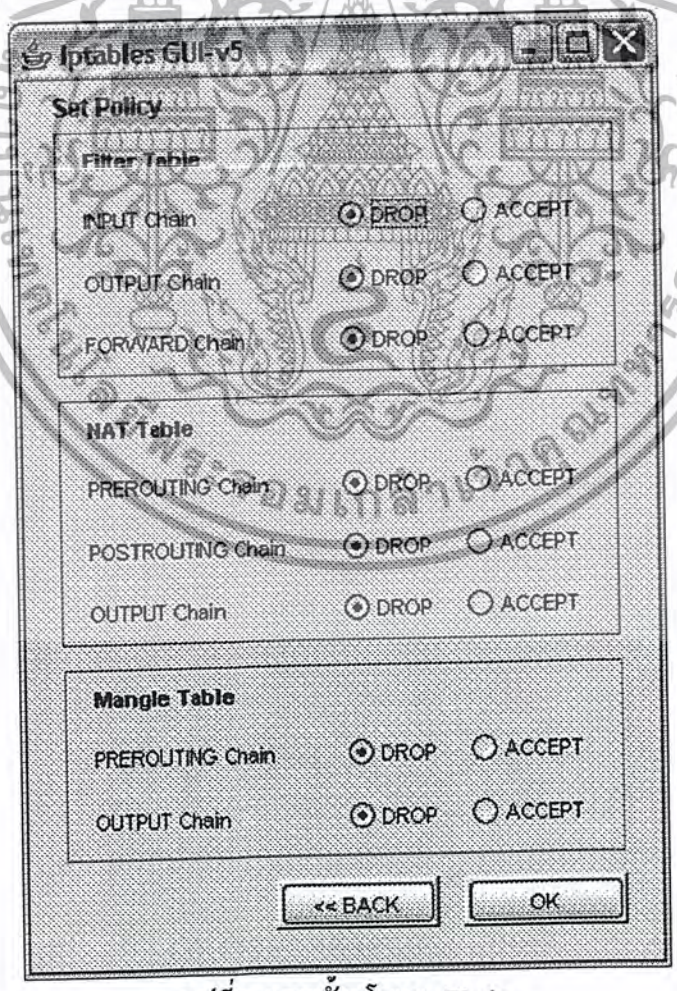
สามารถเปลี่ยนชื่อ Chain ได้จาก ComboBox ที่มีชื่อ Chain ทั้งหมด เปลี่ยนเป็น(Change to) เป็น Textfield ให้ใส่ชื่อที่ต้องการ

5. ส่วนการเปลี่ยนชื่อ Chain




*รูปที่ ข-5 ส่วนการเปลี่ยนชื่อ Chain*

6. การตั้งนโยบาย Chain




*รูปที่ ข-6 การตั้งนโยบาย Chain*

การตั้งนโยบายของ Iptables สามารถเลือกได้ 2 แบบ ได้แก่ ACCEPT และ DROP โดยสามารถเลือกได้ทุกตาราง ผู้ใช้สามารถเลือกได้ว่าจะให้ตารางต่างๆเป็นนโยบายแบบใด

**Iptables Graphic User Interface(GUI) Analysis**

สามารถทำงานได้เหมือน script ทุกประการ

# ภาคผนวก ค.

# วิธีการใช้งานกราฟฟิกยูสเซอร์อินเตอร์เฟสของ Snort

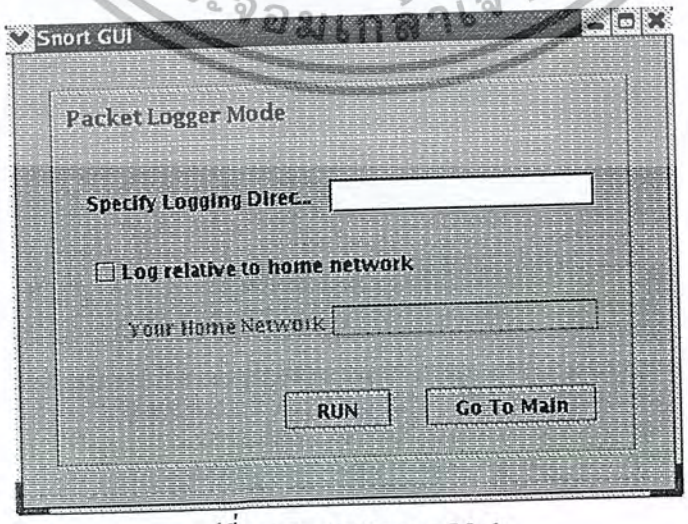
**1. Snort GUI Mode**

snort สามารถ run ได้ 2 mode จึงมีหน้าแรกให้เลือกว่าต้องการทำ mode ไหนโดยเป็น button ให้เลือก


*รูปที่ ค-1 Snort GUI Mode*

**1.2 Packet Logger Mode**

ถ้ากดปุ่ม Packet Logger Mode จะเข้าสู่ Frame ดังนี้




*รูปที่ ค-2 Packet Logger Mode*

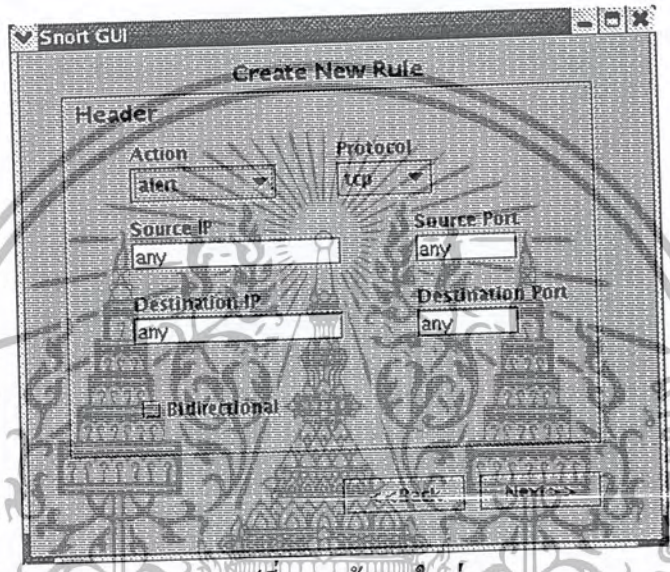
user ต้องใส่ directory ที่จะเก็บ log ก่อน จึงใช้เป็น TextField ให้ user กรอก directory ที่ต้องการลงไป

ส่วนของ Log relative to home network ไม่จำเป็นต้องมีก็ได้จึงใช้ CheckBox

Your Home Network เป็น TextField ให้ user กรอก Home Network ของตัวเองลงไป

**1.3 Network Intruaion Mode**

เมื่อกดปุ่ม Network Intrusion Mode จะเข้าสู่ Frame ดังนี้




*รูปที่ ค-3 สร้างกฎใหม่*

ซึ่งเป็นหน้าที่ให้ user สามารถสร้าง Rule เพิ่มเข้าไปใน snort.conf ได้ โดย Rule จะประกอบไปด้วย 2 ส่วนคือ

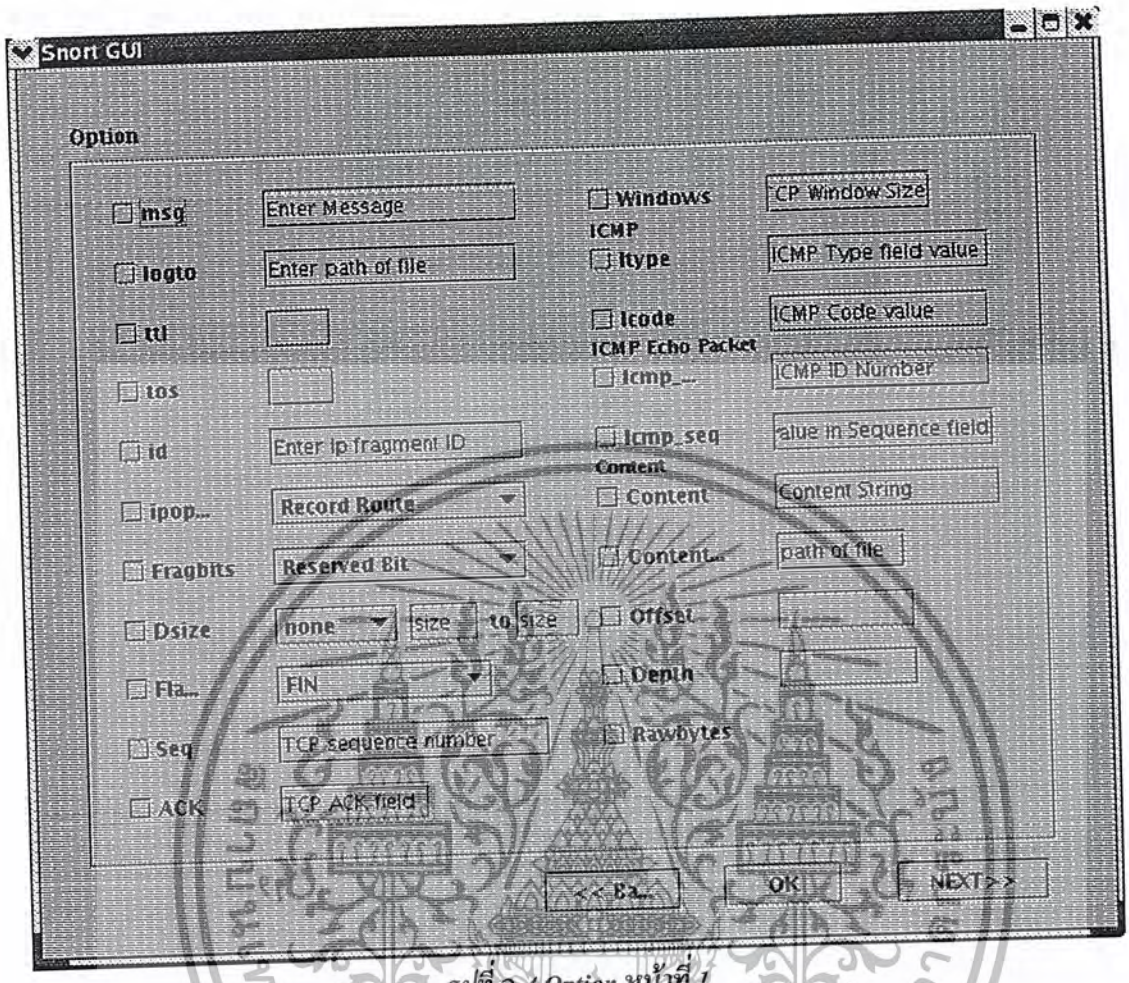
1. Rule Header
2. Rule Option

ส่วนของ Header จะประกอบไปด้วย

- Action สามารถเลือกได้ 5 ประเภทจึงใช้ ComBoBox ซึ่งจะเลือกได้ทีละแบบเท่านั้น
- Protocol สามารถเลือกได้ 4 ประเภทจึงใช้ ComBoBox ซึ่งจะเลือกได้ทีละแบบเท่านั้น

Source IP , Destination IP , Source Port และ Destination Port เป็น TextField ให้ user กรอก IP หรือ Port ที่ต้องการลงไป

เมื่อกดปุ่ม Next จะเข้าสู่หน้า Option ดังนี้




*รูปที่ ค-4 Option หน้าที่ 1*

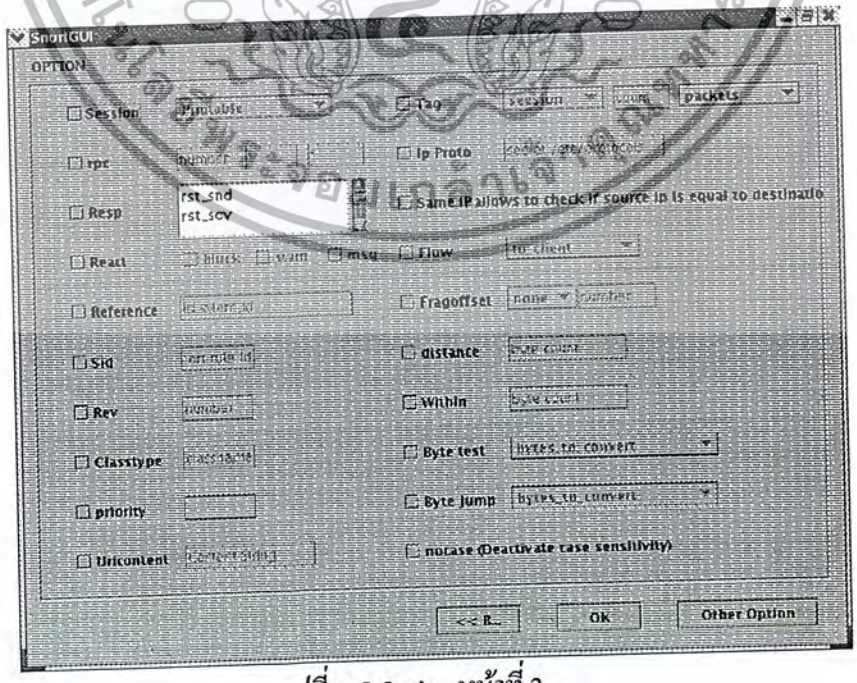
แต่เนื่องจาก Option ในการเขียน Rule ของ Snort มีทั้งหมด 41 Option จึงได้แบ่งออกเป็น 2 Frame เพื่อความสะดวกในการใช้และเนื่องจาก Option มีเป็นจำนวนมาก จึงใช้วิธีเขียนวิธีใส่ output ไปในช่องที่จะให้ user กรอกเลยเพื่อความเรียบง่ายในการใช้

รายละเอียด Rule Option มีดังนี้(สามารถเลือกได้ทีละหลายๆ Rule จึงเลือกใช้ CheckBox)

1. msg เป็น TextField ให้ user กรอก message ที่ต้องการ
2. logto เป็น TextField ให้ user กรอก path ที่ต้องการใส่ log
3. ttl เป็น TextField ให้ user ใส่ค่า ttl ที่ต้องการ
4. tos เป็น TextField ให้ user ใส่ค่า tos ที่ต้องการ
5. id เป็น TextField ให้ user ใส่ค่า Ip Fragment ที่ต้องการ
6. ipoption มีอยู่ 8 ประเภทแต่เนื่องจากสามารถเลือกได้ทีละประเภทเท่านั้นจึงเลือกใช้ ComboBox size ให้ user ใส่ค่าที่ต้องการลงไปเป็น TextField

7. Fragbit มี 3 ประเภทแต่เนื่องจากสามารถเลือกได้ทีละประเภทเท่านั้นจึงเลือกใช้ ComboBox
8. DSize สามารถเลือกว่าจะให้ใส่เป็น มากกว่า(<) , น้อยกว่า(>) หรือไม่ใส่ก็ได้ (none) และสามารถเลือกได้ทีละประเภทเท่านั้นจึงเลือกใช้ ComboBox
9. Flags มีอยู่ 9 ประเภทสามารถเลือกได้หลายๆประเภทจึงเลือกใช้ ListBox
10. Windows เป็น TextField ให้ user ใส่ค่า TCP Window Size ได้ตามต้องการ
11. (ส่วน ICMP) Itype เป็น TextField ให้ user ใส่ค่า TCP Type field value ได้ตามต้องการ
12. (ส่วน ICMP) Icode เป็น TextField ให้ user ใส่ค่า Code Value ได้ตามต้องการ
13. (ส่วน ICMP Echo Packet) Icmp_id เป็น TextField ให้ user ใส่ค่า Icmp id ได้ตามต้องการ
14. (ส่วน ICMP Echo Packet) Icmp_seq เป็น TextField ให้ user ใส่ค่า sequence ใน sequence field ได้ตามต้องการ
15. (ส่วน Content) Content เป็น TextField ให้ user ใส่ Content ที่ต้องการ
16. (ส่วน Content) Content list แทนที่จะใช้ Content user สามารถใช้ Content list แทนได้โดยการใส่ path ของ file ที่มี content ที่ต้องการอยู่เป็น TextField
17. (ส่วน Content) Offset เป็น TextField ให้ user ใส่ค่าที่ต้องการ
18. (ส่วน Content) Depth เป็น TextField ให้ user ใส่ค่าที่ต้องการ
19. (ส่วน Content) Rawbytes ไม่ต้องใส่ค่าอะไรลงไป
20. Seq เป็น TextField ให้ user ใส่ค่า TCP sequence ที่ต้องการ
21. ACK เป็น TextField ให้ user ใส่ค่า TCP ACK ที่ต้องการ

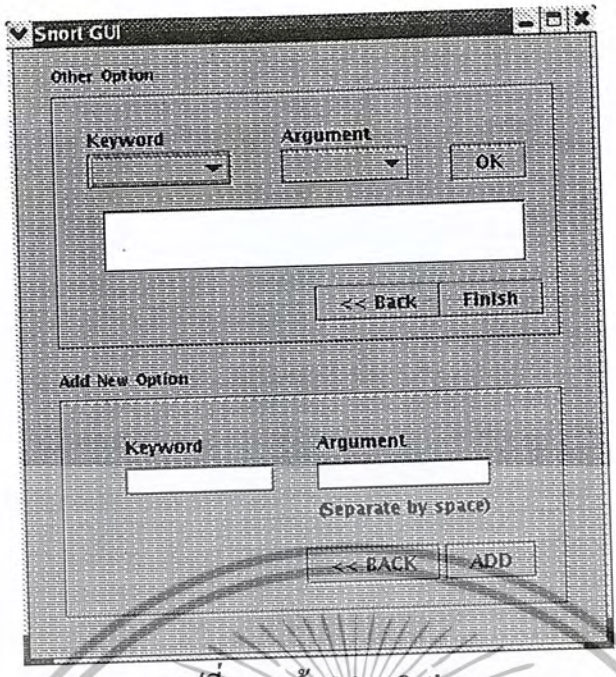
Option ยังไม่ครบ user สามารถเลือกไปใช้ Option หน้าที่เหลือได้โดยการกดปุ่ม Next เมื่อกดปุ่ม Next จะมี Frame ขึ้นมาดังนี้




*รูปที่ ค-5 Option หน้าที่ 2*

22. session สามารถเลือกได้ 2 ประเภทแต่สามารถเลือกได้ทีละประเภทเท่านั้นจึงใช้ ListBox
23. rpc เป็น Textfield ให้ user ใส่ค่าที่ต้องการ และสามารถใส่ได้ 3 ค่า โดยช่องแรกต้องใส่ค่า แต่ 2 ช่องหลังจะไม่ใส่ค่า หรือใส่ * ก็ได้
24. Resp สามารถเลือกได้ 7 ประเภท แต่สามารถเลือกได้ทีละประเภทเท่านั้นจึงใช้ ListBox
25. React สามารถเลือกได้ 3 ประเภทและสามารถเลือกได้หลายๆประเภทจึงใช้ CheckBox
26. Reference เป็น TextField ให้ user ใส่ id system, id ที่ต้องการได้ไม่จำกัด
27. Sid เป็น TextField ให้ user ใส่ค่าSnort rule id ที่ต้องการ
28. Rev เป็น TextField ให้ user ใส่ค่า Rev ที่ต้องการ
29. ClassType เป็น TextField ให้ user ใส่ classname ที่ต้องการ
30. Priority เป็น TextField ให้ user ใส่ค่า priority ที่ต้องการ
31. Uricontent เป็น TextField ให้ user ใส่ค่า Content String ที่ต้องการ
32. Tag ประกอบไปด้วย 3 ส่วน ได้แก่
    - Type สามารถเลือกได้ 3 ประเภทแต่สามารถเลือกได้ทีละประเภทเท่านั้นจึงใช้ ComboBox
    - count เป็น TextField ให้ user ใส่ค่าที่ต้องการ
    - metric สามารถเลือกได้ 2 ประเภทแต่สามารถเลือกได้ทีละประเภทเท่านั้นจึงใช้ ComboBox
33. Ip proto เป็น TextField ให้ user ใส่ protocol ที่ต้องการ โดยสามารถดูได้ที่ /etc/protocols
34. SameIp ไม่ต้องการค่าใดๆ
35. Flow สามารถเลือกได้ 7 ประเภทแต่สามารถเลือกได้ทีละประเภทเท่านั้นจึงใช้ ComboBox
36. Fragoffset สามารถเลือกน้อยกว่า (<) , (>) หรือไม่เลือกก็ได้แต่สามารถเลือกได้ทีละประเภท เท่านั้นจึงเลือกใช้ ComboBox และ เป็น TextField ให้ user ใส่ค่าที่ต้องการ
37. distance เป็น TextField ให้ user ใส่ค่า byte count ที่ต้องการ
38. Within เป็น TextField ให้ user ใส่ค่า byte count ที่ต้องการ
39. ByteTest สามารถเลือกได้ 11 ประเภทแต่สามารถเลือกได้ทีละประเภทเท่านั้นจึงใช้ ComboBox
40. ByteJump สามารถเลือกได้ 10 ประเภทแต่สามารถเลือกได้ทีละประเภทเท่านั้นจึงใช้ ComboBox
41. nocase ไม่ต้องการค่าใดๆ

เมื่อกดปุ่ม Other Option จะมีหน้าอีกหน้าหนึ่งขึ้นมาดังรูปที่ ค-6 ซึ่งผู้ใช้สามารถกำหนด Option เพิ่มเติมได้จากหน้า Other Option นี้



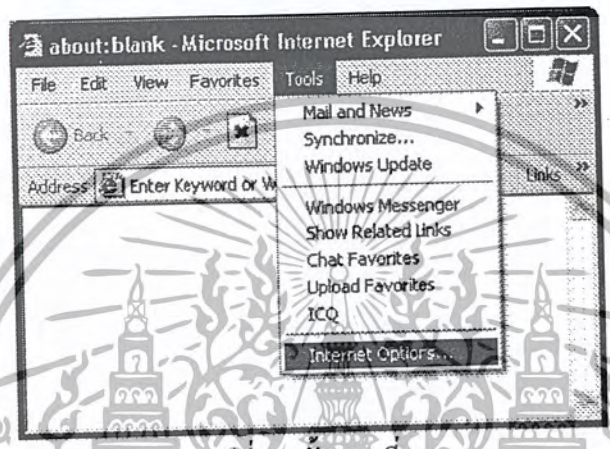

*รูปที่ ค-6 หน้า Other Option*

# ภาคผนวก ง.

# วิธีกำหนดให้เครื่องไคลเอนท์ใช้พร็อกซี่

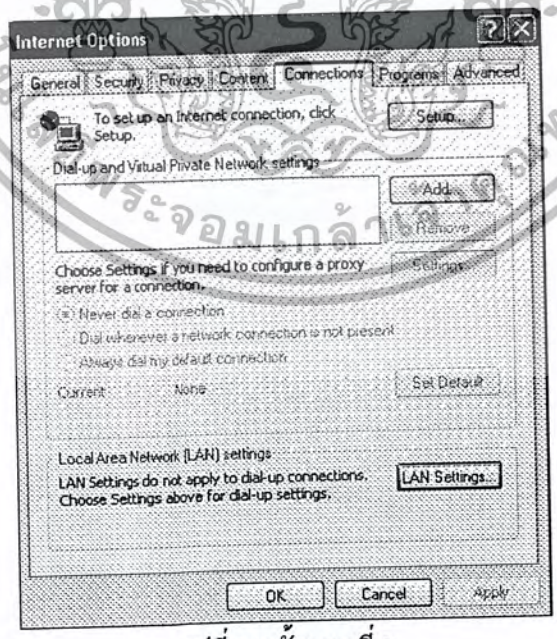
การกำหนดให้ client ใช้ squid ทำได้โดย

1. ไปที่หน้า browser เลือก Tools -> Internet Options…



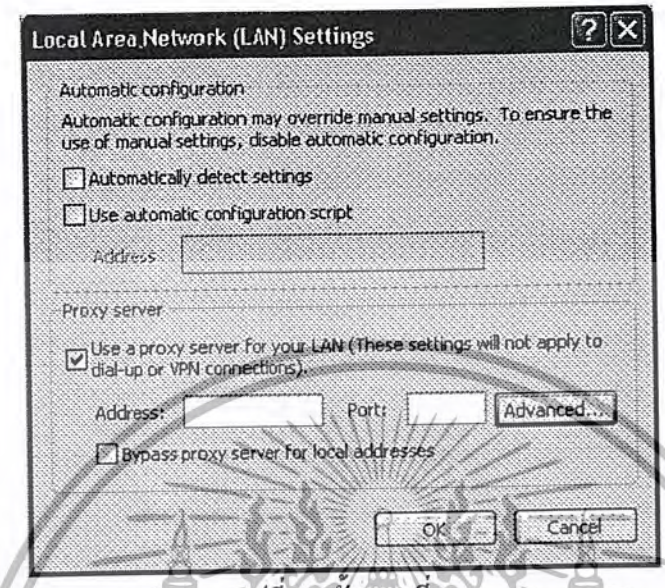

*รูปที่ ง-1 ขั้นตอนที่ 1*

2. ในหน้า Internet Options เลือก LAN Settings…



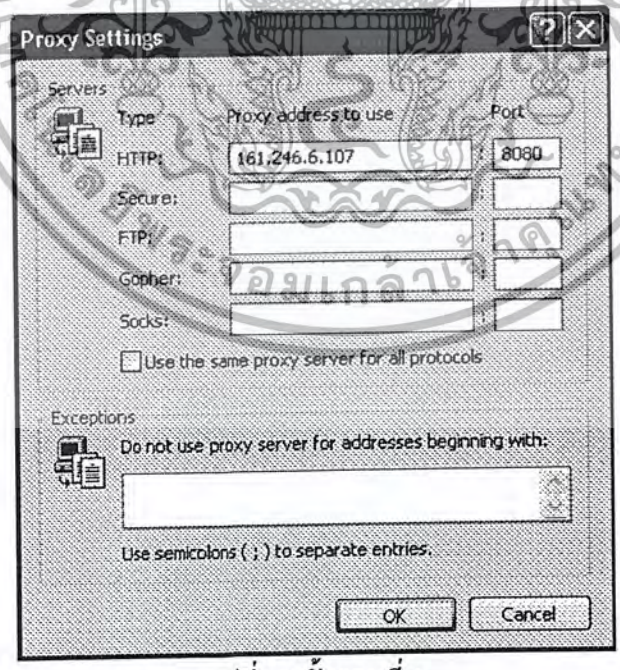

*รูปที่ ง-2 ขั้นตอนที่ 2*

3. คลิกถูกเพื่อเลือกที่ use a proxy server for your lan... แล้วเลือก Advanced...




*รูปที่ ง-3 ขั้นตอนที่ 3*

4. พิมพ์ ip หรือ url address ที่จะใช้เป็น Proxy Server (ในที่นี้ใช้ ip 161.246.6.107 เป็น Proxy Server) port : 8080 ซึ่งเป็น port ของ http คลิก OK




*รูปที่ ง-4 ขั้นตอนที่ 4*

# บรรณานุกรม


**หนังสืออ้างอิง**

[1] เรืองไกร รังสิพล, "*เปิดโลก Firewall และ Internet Security*", โปรวิชั่น, 2545.

[2] ก่อกิจ วีระอาชากุล, สัจจะ จรัสรุ่งรวีวร, "*ติดตั้งและปรับแต่งเซิร์ฟเวอร์ Linux สำหรับ Admin Linux โดยเฉพาะ*", อินโฟเพรส, 2545.

[3] ณัฐดนัย สุขรัตน์, "*คัมภีร์ Java เล่ม 2*", เคทีพี คอมพ์ แอนด์ คอมซัลท์, 2546.

[4] ดร. วีระศักดิ์ ซึงถาวร, "*Java Programming Volumn I*", ซีเอ็ดยูเคชั่น, 2546.

[5] ดร. วีระศักดิ์ ซึงถาวร, "*Java Programming Volumn II*", ซีเอ็ดยูเคชั่น, 2546.

**เว็ปไซต์อ้างอิง**

[1] http://www.thaicert.nectec.or.th

[2] http://www.squid-cache.org

[3] http://www.snort.org

[4] http://java.sun.com

[5] http://javaalmanac.com

[6] http://freatmeat.net

[7] http://sourceforge.net

[8] http://www.iptables.org