ซอฟแวร์ติดตั้งไฟร์วอลล์ที่ใช้งานโดยติดต่อทางกราฟฟิคด้วยเทคโนโลยี XML

An XML – Based GUI Tool For Firewall Configuration

นายนิวัตร นิมมานศักดา
นายประกาศิต อัศวภักดีพันธุ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัญฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
สาขาวิศวกรรมคอมพิวเตอร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

ซอฟแวร์ติดตั้งไฟร์วอลล์ที่ใช้งานโดยติดต่อทางกราฟฟิคด้วยเทคโนโลยี XML

An XML – Based GUI Tool For Firewall Configuration

โดย

นายนิวัตร นิมมานศักดา รหัส 41014228

นายประกาศิต อัศวภักดีพันธุ์ รหัส 41014246

อาจารย์ที่ปรึกษา

ดร. วิศิษฏ์ หิรัญกิตติ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ซอฟแวร์ติดตั้งไฟร์วอลล์ที่ใช้งานติดต่อทางกราฟฟิคด้วยเทคโนโลยี XML

An XML – based GUI Tool for firewall Configuration

ผู้จัดทำ

1. นาย นิวัตร นิมมานศักดา รหัส 41014228
2. นาย ประกาศิต อัศวภักดีพันธุ์ รหัส 41014246

_____ อาจารย์ที่ปรึกษา

(ดร. วิศิษฏ์ หิรัญกิตติ)

ซอฟแวร์ติดตั้งไฟร์วอลล์ที่ใช้งานติดต่อทางกราฟฟิกด้วยเทคโนโลยี XML

นายนิวัตร นิมมานศักดา 41014228
นายประกาศิต อัศวภักดีพันธุ์ 41014246
ดร. วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ปริญญานิพนธ์นี้พัฒนาโปรแกรมที่ช่วยในการติดตั้ง ไฟร์วอลล์ไอพีเทเบิลให้ง่ายต่อการใช้งานของผู้ใช้ที่ขาดความรู้ในด้านนี้ ซึ่งโปรแกรมนี้จะต้องใช้กับซอฟต์แวร์ไฟร์วอลล์ไอพีเทเบิลและใช้ภาษาจาวาและเทคโนโลยีของ XML มาใช้ในการพัฒนาโปรแกรม ซึ่งภาษา XML จะเป็นตัวบรรยายนโยบายของไฟร์วอลล์ และใช้ GUI ในการเข้าถึงและเปลี่ยนแปลง XML อิลิเมนต์ ความสามารถของโปรแกรม เช่น การควบคุมและติดตั้งไฟร์วอลล์ระยะไกล การเข้ารหัสของไฟล์กฎ การค้นหากฎที่คอนฟิกแล้ว การเลือกใช้กฎต่าง ๆ โดยมีออฟชั่นให้เลือกได้สะดวก การเซทบริการต่างๆ อาทิ บริการเอฟทีพี บริการเทลเน็ท บริการเอชทีทีพี และอื่น ๆ ได้อย่างสะดวก ความง่ายและความสะดวกของจะทำให้ผู้ใช้โปรแกรมสามารถติดตั้งไฟร์วออล์ได้อย่างง่ายดาย

An XML – based GUI Tool for firewall Configuration

Mr. Niwar Nimmansakda 41014228
Mr. Prakasit Asavapakdepan 41014246
Dr. Visit Hirankitti Advisor

ABSTRACT

This project developed a software tool for easy configurating of IPtables Firewall. we made the tool so easy that a novice can use it. To use this software tool the user must have IPtables Firewall in use. To make this firewall product configurable we adopted XML to describe network policies of a firewall and provided GUI to easily access to and modify these XML elements.

This tool allows the user to config a firewall remotely. The user can search for any policies previously set for viewing and modification. The user are given with options to set policies, Ftp service, Telnet service, HTTP service, and so on

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้ปริญญานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ให้ความเอาใจใส่แนะนำ และช่วยเหลือเสมอมาซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอขอบคุณเพื่อน ๆ ที่คอยแนะนำเสมอมา ขอบคุณป้อม, โจ้ดี ด้วยครับและต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่งซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมาในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

นาย นิวัตร นิมมานศักดา

นาย ประกาศิต อัศวภักดีพันธุ์

สารบัญ

หน้าที่

สารบัญภาพประกอบ

หน้าที่

หน้าที่

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

อินเตอร์เน็ต เป็นเครือข่ายคอมพิวเตอร์ขนาดใหญ่ ที่เชื่อมโยงเอาเครือข่ายคอมพิวเตอร์ย่อยๆ ทั่วโลกเข้าด้วยกัน มีการแลกเปลี่ยนข่าวสารกันตลอด 24 ชั่วโมง การคุยกับคอมพิวเตอร์ในอีกซีกโลกหนึ่งจะง่ายพอๆ กับการคุยกับคอมพิวเตอร์ที่อยู่ในห้องติดกัน เมื่อไรก็ตามที่คอมพิวเตอร์ของคุณต่อเข้ากับอินเตอร์เน็ตก็เปรียบเสมือนกับตัวแมงมุมกลางใยแมงมุมขนาดมหึมา ซึ่งใยแมงมุมแต่ละเส้นนำข้อมูลมาจากคอมพิวเตอร์เครื่องอื่นมาสู่คอมพิวเตอร์ของคุณปัจจุบันนี้หลายๆ องค์กรกำลังให้ความสนใจและกำลังเตรียมพร้อมเพื่อนำองค์กรของตนเข้าสู่อินเตอร์เน็ต เพื่อรองรับการติดต่อสื่อสารและการค้นคว้าหาข้อมูลต่างๆ กระนั้นก็ตามมีผู้ไม่ประสงค์ดีบางคนบางกลุ่ม คอยฉวยโอกาสนี้ในการโจรกรรมหรือทำลายข้อมูลระหว่างการทำธุรกรรม แม้กระทั่งการทำลายระบบเครือข่ายขององค์การของเรา สิ่งที่ต้องนำมาใช้ป้องกันจากผู้ไม่ประสงค์ดีนั้นก็คือในเรื่องของระบบรักษาความปลอดภัย โดยในด้านการปฏิบัติแล้วเราอาจกำหนดนโยบายขององค์กรให้ทั่วถึงและจริงจัง โดยต้องมีการแจ้งข้อปฏิบัติให้ พนักงานในองค์กรและสร้างเสริมให้พนักงานตระหนักถึงความสำคัญของระบบการรักษาความปลอดภัยและภัยจากการถูกบุกรุก การปฏิบัติตามข้อปฏิบัติอาจไม่เพียงพอดังนั้นจึงต้องมีระบบรักษาความปลอดภัยเพื่อป้องกันภัยจากผู้บุกรุก

ในเรื่องเกี่ยวกับการป้องกันเครือข่ายจากผู้บุกรุกไฟร์วอลล์ (Firewall) เป็นทางเลือกที่ดีทางหนึ่งที่จะลดความเสี่ยงภัยจากผู้ไม่ประสงค์ดี

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อสร้างระบบไฟร์วอลล์เพื่อป้องเป็นการลดความเสี่ยงของการบุกรุกและมุ่งหวังให้เครือข่ายภายในของเรามีความปลอดภัยมากขึ้น โดยไฟร์วอลล์จะสามารถกำหนดกฎการเข้าออกของข้อมูลบนเครือข่ายได้ อีกทั้งระบุบริการที่จะให้บริการหรือปิดบริการการได้ อีกทั้งยังมีเนื้อหาอื่นๆ อีกที่จะทำให้เครือข่ายของเรามีความปลอดภัยมากยิ่งขึ้น สุดท้ายจะเป็นการทดสอบระบบไฟร์วออล์ที่สร้างขึ้นมาสร้างความปลอดภัยให้กับเครือข่ายแค่ไหน และอย่างไรบ้างที่ทำให้ไม่มีความปลอดภัย เพื่อหาแนวทางแก้ไขต่อไป

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์ที่จัดทำขึ้นนี้ จัดทำภายใต้วัตถุประสงค์หลัก 5 ประการได้แก่

1. แสดงเรื่องความปลอดภัยบนเครือข่ายอินทราเน็ต
2. บอกถึงการทำงานของระบบเครือข่ายที่ใช้โปรโตคอล (TCP/IP)
3. แสดงการบุกรุกทางเครือข่ายคอมพิวเตอร์และรูปแบบการโจมตีในแบบต่างๆ ของผู้ประสงค์ร้าย

4. แสดงการทำงานของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต โดยครอบคลุมตั้งแต่โครงสร้างของระบบรักษาความปลอดภัยบนเครือข่ายอินทราเน็ตชนิดต่างๆ ข้อดีข้อเสียของของระบบรักษาความปลอดภัยชนิดต่างๆ
5. บอกถึงข้อบกพร่องของเครือข่ายที่จะนำไฟร์วอลล์มาใช้งานด้วย และกำหนดคุณสมบัติต่างๆ ให้กับเครือข่ายเพื่อให้เครือข่ายมีความปลอดภัยมากขึ้น
6. พัฒนาโปรแกรมเพื่อคอนฟิกระบบไฟร์วอลล์
7. ทดสอบการทำงานของระบบรักษาความปลอดภัยบนเครือข่ายอินทราเน็ต

1.3 ขอบเขตของงาน

งานวิจัยศึกษาชิ้นนี้จะศึกษาความปลอดภัยของเครือข่ายคอมพิวเตอร์ จากนั้นจะทำการออกแบบระบบไฟร์วอลล์ที่เหมาะสมที่จะทำให้ครือข่ายมีความปลอดภัยขึ้นได้ โดยการทำงานของแต่ละบริการจะต้องมีการทำงานได้เหมือนเดิมและจะมีความปลอดภัยมากขึ้น

ทำการพัฒนาโปรแกรมควบคุมระบบไฟร์วอลล์โดยการใช้ภาษาจาวาและ XML สามารถที่จะควบคุมไฟร์วอลล์จากระยะไกลได้

ส่วนการทดสอบระบบรักษาความปลอดภัยของเครือข่ายนั้น จะทดสอบกับรูปแบบการโจมตีที่จะทำให้เครือข่ายไม่ปลอดภัย และถึงแนวทางที่จะแก้ไขให้เครือมีความปลอดภัยมากขึ้น ซึ่งในงานวิจัยนี้พยายามลดช่องโหว่ของเครือข่าย

1.4 ขั้นตอนในการดำเนินงาน

1. ศึกษาและค้นคว้าข้อมูลของการสื่อสารระหว่างเครือข่าย
2. ศึกษาการทำงานของโปร โตคอล ทีซีพี/ไอพี (TCP/IP)
3. ศึกษาถึงข้อบกพร่องของโปร โตคอล ทีซีพี/ไอพี เพื่อหาแนวทางป้องกัน
4. ศึกษารูปแบบการบุกรุกและโจมตีเครือข่ายในแบบต่างๆ
5. ศึกษารายละเอียดและการทำงานของไฟร์วอลล์ (Firewall)
6. ศึกษารายละเอียดและการใช้งาน iptables เพื่อนำมาสร้างกฎเกณฑ์ต่างๆ สำหรับไฟร์วอลล์
7. ศึกษาความหมายและรายละเอียดของ Network Address Translation (NAT) และการอิมพลีเมนต์โดยใช้ iptables
8. ศึกษาภาษาจาวาและ XML เพื่อมาใช้ในการพัฒนาโปรแกรม
9. สร้างระบบไฟร์วอลล์เพื่อรักษาความปลอดภัยให้กับเครือข่าย
10. ทดสอบการทำงานของระบบไฟร์วอลล์ที่สร้างขึ้นมา

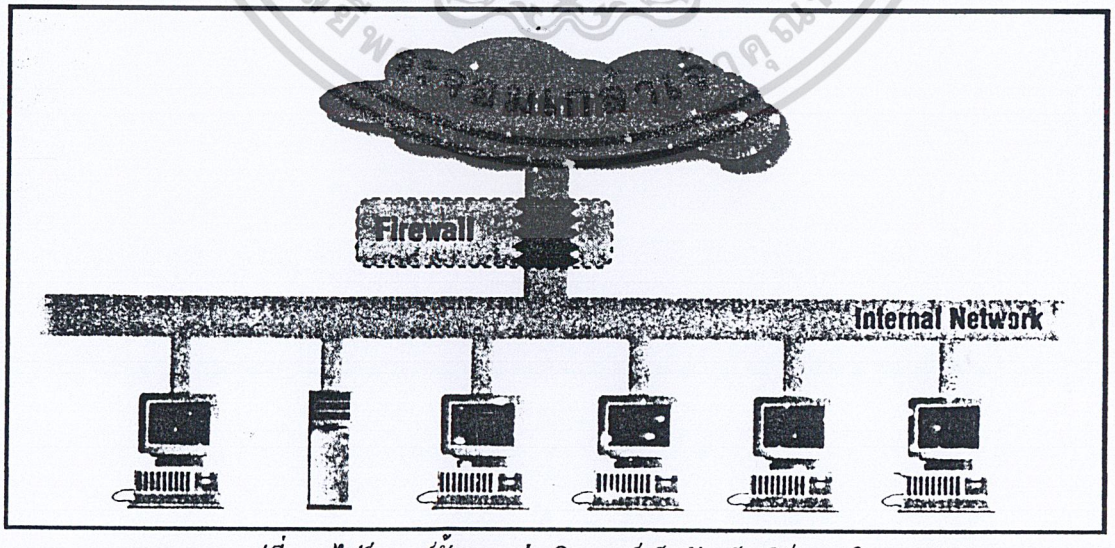
บทที่ 2
ไฟร์วอลล์ (Firewall)

ไฟร์วอลล์ (Firewall): เครื่องมือที่ใช้ในการควบคุมการสื่อสารข้อมูลระหว่างเน็ตเวิร์กหนึ่งกับอีกเน็ตเวิร์กหนึ่ง โดยทั่วไปแล้วไฟร์วอลล์จะถูกติดตั้งกั้นระหว่างเน็ตเวิร์กภายในองค์กรกับอินเตอร์เน็ต เพื่อป้องกันการใช้งานที่ไม่ได้รับอนุญาต

ไฟร์วอลล์มีหน้าที่หลักที่สำคัญ คือ ควบคุมการสื่อสารและเป็นเครื่องมือที่เปิดโอกาสให้สามารถจำกัดขอบเขตการติดต่อระหว่างโฮสต์กับเน็ตเวิร์กอื่นๆ ได้ การมีไฟร์วอลล์จะทำให้สามารถควบคุมได้ว่าโฮสต์ใด เน็ตเวิร์คใด และพอร์ตใดที่เป็นที่อนุญาตและประเภทใดที่ไม่เป็นที่อนุญาต

ดังนั้นไฟร์วอลล์จึงเป็นเครื่องมือรักษาความปลอดภัยขั้นพื้นฐานที่สำคัญที่สุดสำหรับเน็ตเวิร์ก แต่การมีไฟร์วอลล์จะต้องอาศัยการกำหนดกฎการเข้าถึงเน็ตเวิร์ก (Access Rules) ที่ถูกต้องด้วยจึงจะทำงานได้อย่างสมบูรณ์ การมีไฟร์วอลล์แต่ขาดการกำหนดกฎที่รัดกุมจะทำให้ความสามารถในการรักษาความปลอดภัยลดต่ำลงจนอาจจะไม่มีความปลอดภัยเลยก็เป็นได้ ดังนั้นการที่มีไฟร์วอลล์อยู่ไม่ได้เป็นการรับประกันว่าเน็ตเวิร์กจะได้รับการป้องกันเสมอไป

อีกประการหนึ่งคือ ถึงแม้จะมีไฟร์วอลล์อยู่และมีการกำหนดกฎที่ถูกต้องรัดกุมแล้วก็ตาม แต่ถ้าโฮสต์นั้นมีความจำเป็นที่จะต้องให้บริการกับภายนอก กฎของไฟร์วอลล์ก็จำเป็นต้องอนุญาตให้มีการสื่อสารข้อมูลผ่านทางพอร์ตที่กำหนดนั้น ดังนั้นหากภัยคุกคามหรือการโจมตีใดๆ ใช้ช่องทางที่ไฟร์วอลล์อนุญาตอย่างถูกต้อง แพ็กเก็ตที่ใช้เพื่อการนั้นก็จะสามารถผ่านไฟร์วอลล์เข้าไปยังโฮสต์ได้อยู่ดี



รูปที่ 2-1 ไฟร์วอลล์กั้นระหว่างอินเตอร์เน็ตกับเน็ตเวิร์กภายใน

สิ่งที่ไฟร์วอลล์ช่วยได้

- บังคับใช้นโยบายด้านความปลอดภัย โดยการกำหนดกฎให้กับไฟร์วอลล์ว่าจะอนุญาต หรือไม่อนุญาตในการใช้บริการต่างๆ ภายในเครือข่าย
- ทำให้การพิจารณา ดูแล และการตัดสินใจด้านความปลอดภัยของระบบเป็นไปได้ง่ายขึ้น เนื่องจากการสื่อสารทุกชนิดระหว่างเน็ตเวิร์กภายนอกกับเน็ตเวิร์กภายในจะต้องผ่านไฟร์วอลล์ การดูแลที่จุดนี้เป็นการดูแลความปลอดภัยในระดับเน็ตเวิร์ก (Network-based Security)
- เก็บรายละเอียดของกิจกรรมต่างๆ ระหว่างเน็ตเวิร์กภายนอกและเน็ตเวิร์กภายในได้อย่างมีประสิทธิภาพ ทำให้สามารถตรวจสอบได้ง่าย
- ป้องกันเน็ตเวิร์กภายในบางส่วนจากการเข้าถึงของเน็ตเวิร์กภายนอก เช่น มีบางส่วนของเน็ตเวิร์กภายในที่ต้องการให้ภายนอกเข้ามาใช้บริการได้ แต่ส่วนที่เหลือไม่ต้องการให้ภายนอกเข้ามาได้
- สามารถทำฟังก์ชันอื่นๆ เช่น NAT ได้

สิ่งที่ไฟร์วอลล์ช่วยไม่ได้

- อันตรายที่เกิดจากเน็ตเวิร์กภายใน ไฟร์วอลล์ไม่สามารถป้องกันได้เนื่องจากอยู่ภายในเน็ตเวิร์กเอง ไม่ได้ผ่านไฟร์วอลล์เข้ามา
- อันตรายจากเน็ตเวิร์กภายนอกที่ไม่ได้ผ่านเข้ามาทางไฟร์วอลล์ เช่น การ Dial-up เข้ามายังเน็ตเวิร์กภายในโดยตรงโดยไม่ได้ผ่านไฟร์วอลล์
- การโจมตีโดยอาศัยช่องโหว่ของแอพพลิเคชั่น เช่น ถ้าอนุญาตให้แพ็กเก็ตของ HTTP ผ่านได้ อาจมีการใช้ CGI ในการโจมตีได้
- ไวรัส เพราะไฟร์วอลล์ไม่สามารถตรวจดูรายละเอียดข้อมูลภายในแพ็กเก็ตได้ว่ามีไวรัสหรือไม่

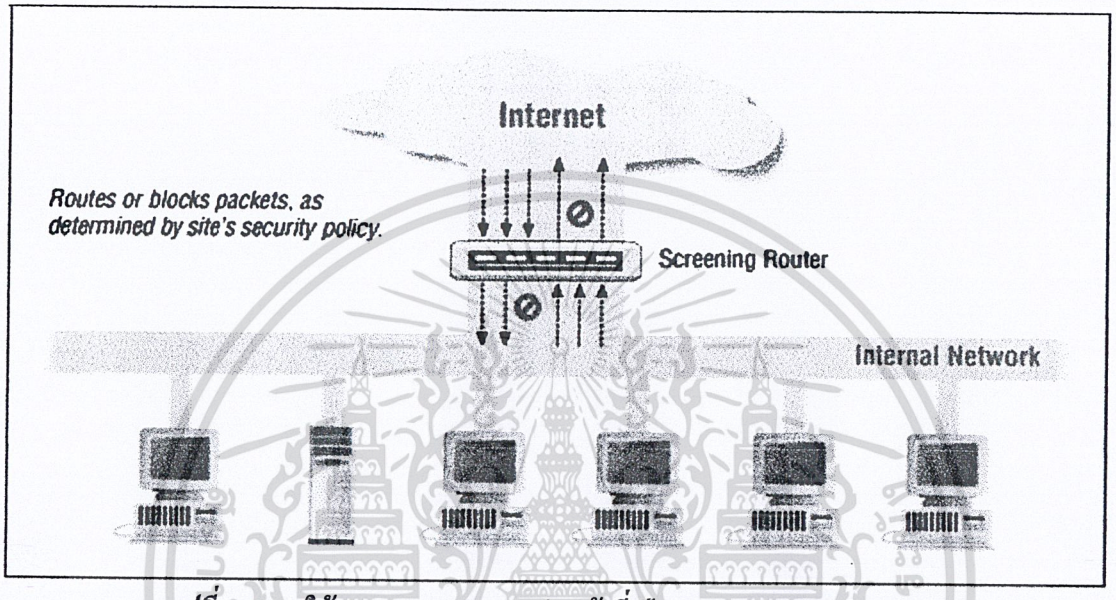
2.1 ชนิดของไฟร์วอลล์

ไฟร์วอลล์สามารถแบ่งตามเทคโนโลยีที่ใช้ในการตรวจสอบและควบคุมได้เป็น

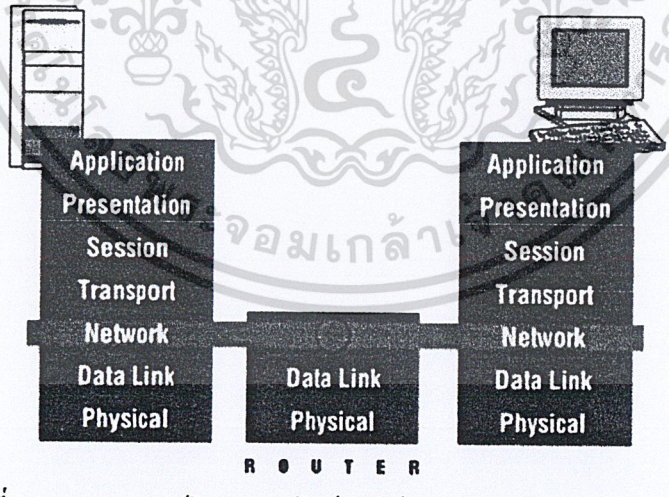
- *Packet-Filtering Router*
- *Stateful Inspection (หรือ Dynamic Packet Filtering)*
- *Application Proxy*
- *Circuit-Level Gateway*

2.1.1 Packet-Filtering Router

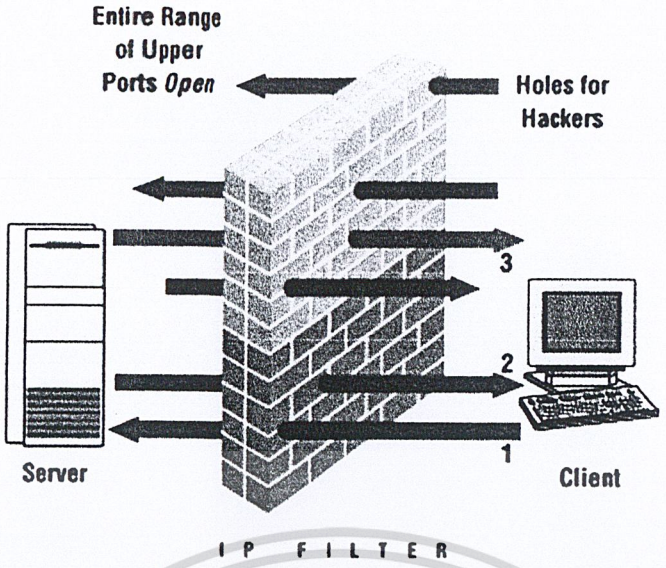
Packet-Filtering Router คือ เราเตอร์ที่ทำการหาเส้นทางและส่งต่ออย่างมีเงื่อนไข โดยจะพิจารณาจากข้อมูลในส่วนของเฮดเดอร์ของแพกเก็ตนั้นๆ เช่น แอดเดรสต้นทาง แอดเดรสปลายทาง พอร์ตหรือโพรโตคอล เทียบกับกฎเกณฑ์ที่กำหนดไว้และตัดสินใจว่าควรจะทำอย่างไรกับแพ็กเก็ต โดยหากเข้ากับกฎใดกฎหนึ่ง ก็จะดูว่ากฎบอกไว้ว่าให้ส่งแพ็กเก็ตนั้นต่อไปหรือดร็อปแพ็กเก็ตนั้นทิ้ง และหากไม่เข้ากับกฎใดเลย ก็จะพิจารณาจากค่า default ของนโยบายว่าจะส่งต่อหรือดร็อปแพ็กเก็ตนั้นทิ้งไป



รูปที่ 2-2 การใช้ Screening Router ทำหน้าที่เป็น Packet-Filtering Router



รูปที่ 2-3 แสดงเลเยอร์โมเดลของไฟร์วอลล์ชนิด Packet-Filtering Router



รูปที่ 2-4 แสดงช่องทางการเข้าออกของแพ็กเก็ตผ่านไฟร์วอลล์ชนิด Packet-Filtering Router

Packet Filtering สามารถนำมาประยุกต์ใช้งานได้จาก 2 แพลตฟอร์ม คือ

1. คอมพิวเตอร์ที่ทำหน้าที่เป็นเราเตอร์
2. เราเตอร์ที่มีความสามารถในการทำ Packet Filtering (ซึ่งมีอยู่ในเราเตอร์ทั่วไปอยู่แล้ว)

ซึ่งแต่ละรูปแบบมีข้อได้เปรียบและเสียเปรียบกันดังนี้

รูปแบบของเราเตอร์	ข้อดี	ข้อเสีย
คอมพิวเตอร์ที่ทำหน้าที่เป็นเราเตอร์	มีความยืดหยุ่นในการทำงานไม่จำกัด สามารถเพิ่มฟังก์ชันการทำงานได้ไม่จำกัด	- ประสิทธิภาพปานกลาง - จำนวนการเชื่อมต่อมีน้อย - อาจมีความเสี่ยงอันเกิดจากความอ่อนแอของตัวระบบปฏิบัติการ
เราเตอร์ที่มีความสามารถในการทำ Packet Filtering	- ประสิทธิภาพสูง - จำนวนการเชื่อมต่อมาก	- ความยืดหยุ่นในการทำงานต่ำ เพิ่มฟังก์ชันการทำงานได้ยาก - อาจต้องเพิ่มจำนวนหน่วยความจำให้มากขึ้น

ตารางที่ 2-1 เปรียบเทียบข้อดีและข้อเสียในการเลือกแพลตฟอร์มสำหรับ Packet Filtering

ข้อดีของ Packet-Filtering Router คือ ไม่ขึ้นกับแอพพลิเคชั่น, ทำงานได้เร็ว และสามารถรองรับการขยายตัวได้ดี ส่วนข้อเสียของ Packet-Filtering Router ก็คือ ยากในการกำหนดกฎเกณฑ์ต่างๆ ให้ถูกต้องและรัดกุม

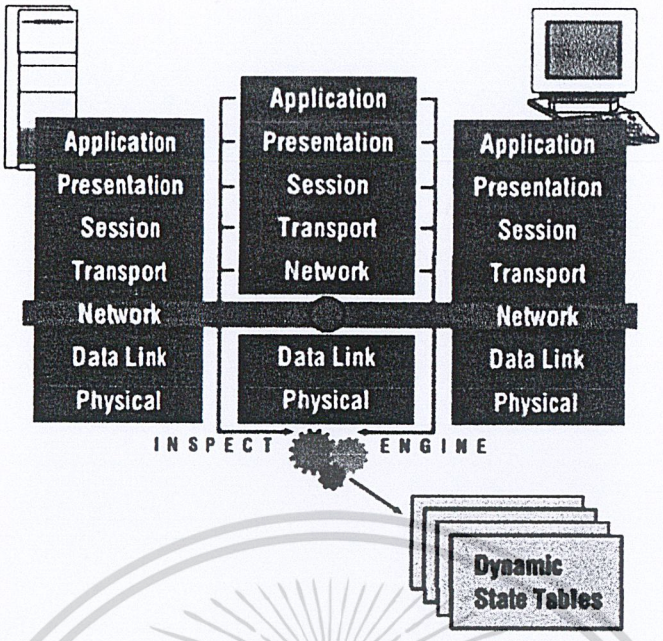
รูปแบบการโจมตีต่างๆ ที่ Packet-Filtering Router สามารถป้องกันได้

- **IP address spoofing:** เป็นการโจมตีจากภายนอก แต่ปลอมแปลงแอดเดรสต้นทางให้เป็นแอดเดรสภายในแทน ซึ่งหากไม่มีการป้องกัน อาจเข้าใจได้ว่าเป็นแพ็กเก็ตที่ส่งมาจากภายใน สามารถยอมรับและไว้ใจได้ ซึ่งอาจก่อให้เกิดอันตรายได้ ตัว Packet-Filtering Router จะป้องกันการโจมตีในลักษณะนี้โดยการดร็อปแพ็กเก็ตที่มีแอดเดรสต้นทางเป็นแอดเดรสภายในแต่กลับมากจากอินเตอร์เฟสภายนอก
- **Source routing attacks:** เครื่องต้นทางจะระบุข้อมูลการเราต์ของแพ็กเก็ต โดยที่หวังว่าจะสามารถผ่านส่วนรักษาความปลอดภัยที่ไม่ได้สนใจในส่วนของข้อมูลการเราต์ได้ ตัว Packet-Filtering Router จะป้องกันด้วยการดร็อปแพ็กเก็ตทั้งหมดที่เป็นรูปแบบเช่นนี้
- **Tiny fragment attacks:** ผู้โจมตีจะใช้วิธี IP fragmentation สร้างแฟรกเมนต์ขนาดเล็กๆ จำนวนมากและบังคับให้ข้อมูลของ TCP header กระจายอยู่ในแฟรกเมนต์ต่างๆ ทำให้สามารถผ่านกฎต่างๆ ที่ขึ้นอยู่กับข้อมูล TCP header ได้ ตัว Packet-Filtering Router จะป้องกัน โดยการดร็อปแพ็กเก็ตของ TCP ที่มี IP Fragment Offset เท่ากับ 1

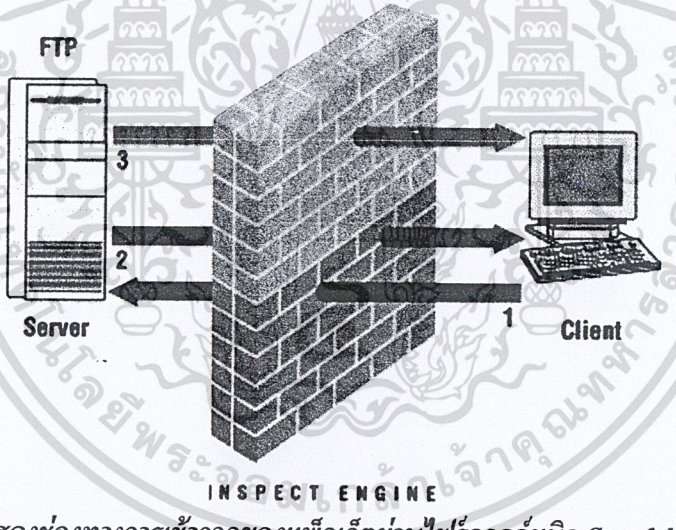
2.1.2 Stateful Inspection

Stateful Inspection เป็นไฟร์วอลล์ที่พัฒนาขึ้นมาจากไฟร์วอลล์แบบ Packet-Filtering ซึ่งโดยปกติแล้ว Packet-Filtering แบบธรรมดา (ที่เป็น Stateless แบบที่มีอยู่ในเราเตอร์ทั่วๆ ไป) จะควบคุมการเข้าออกของแพ็กเก็ตโดยการพิจารณาจากข้อมูลในเฮดเดอร์ของแต่ละแพ็กเก็ตเท่านั้น และจะพิจารณาเฉพาะแพ็กเก็ตนั้นๆ โดยไม่ได้คำนึงถึงแพ็กเก็ตอื่นๆ ด้วย จากนั้นนำข้อมูลมาเปรียบเทียบกับกฎที่มีอยู่ ซึ่งกฎที่มีอยู่นั้นก็เป็นกฎที่สร้างจากข้อมูลในส่วนของเฮดเดอร์เท่านั้น ดังนั้น Packet-Filtering แบบธรรมดาจึงไม่สามารถทราบได้ว่า แพ็กเก็ตนี้อยู่ส่วนใดของการเชื่อมต่อ เป็นแพ็กเก็ตที่เข้ามาติดต่อใหม่เลยหรือเปล่า หรือว่าเป็นแพ็กเก็ตของการเชื่อมต่อที่เกิดขึ้นอยู่แล้ว

แทนที่จะดูตรงส่วนเฮดเดอร์อย่างเดียว Stateful Inspection จะนำเอาข้อมูลในส่วนของเพย์โหลดและข้อมูลที่ได้จากแพ็กเก็ตก่อนหน้านั้นที่ได้บันทึกเอาไว้ มาพิจารณาด้วย โดยสามารถดูลักษณะของการเชื่อมต่อ การโต้ตอบของแต่ละโพรโตคอลที่มีรูปแบบที่แตกต่างกัน โดยสามารถแยกแยะโพรโตคอลที่ถูกต้องและไม่ถูกต้องออกจากกันได้ นอกจากนั้นไฟร์วอลล์ชนิดนี้ยังสามารถปิดพอร์ตที่มีหมายเลขมากกว่า 1023 ได้ด้วย ซึ่งจะเปิดเฉพาะเวลาที่มีการเชื่อมต่อผ่านพอร์ตนั้นๆ เท่านั้น ทำให้ปลอดภัยกว่าไฟร์วอลล์ชนิด Packet-Filtering



รูปที่ 2-5 แสดงเลเยอร์โมเดลของไฟร์วอลล์ชนิด Stateful Inspection



รูปที่ 2-6 แสดงช่องทางการเข้าออกของแพ็กเก็ตผ่านไฟร์วอลล์ชนิด Stateful Inspection

ข้อดีและข้อเสียของไฟร์วอลล์ชนิดนี้จะเหมือนๆ กับไฟร์วอลล์ชนิด Packet-Filtering แต่จะมีความปลอดภัยเพิ่มสูงขึ้น และสามารถควบคุมการเข้าออกของแพ็กเก็ตได้ลึกซึ้งกว่า

ตัวอย่างผลิตภัณฑ์ทางการค้าที่เป็น Stateful Inspection

- Check Point Firewall-1
- Cisco Secure Pix Firewall
- SunScreen Secure Net

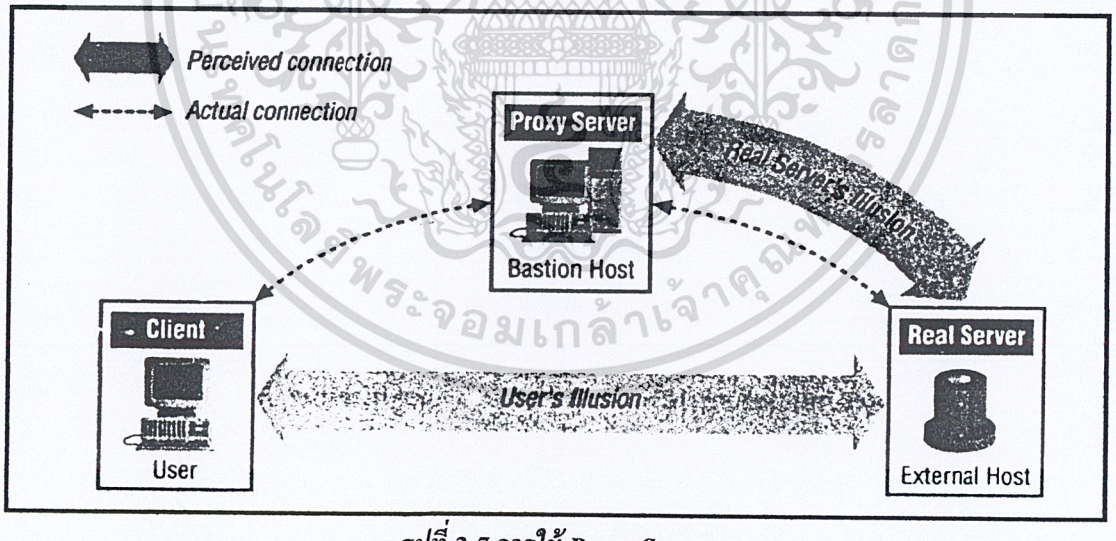
และส่วนที่เป็นผลิตภัณฑ์ที่แจกฟรี

- NetFilter ใน Linux (IPtables ในเคอร์เนล 2.4 ขึ้นไป)

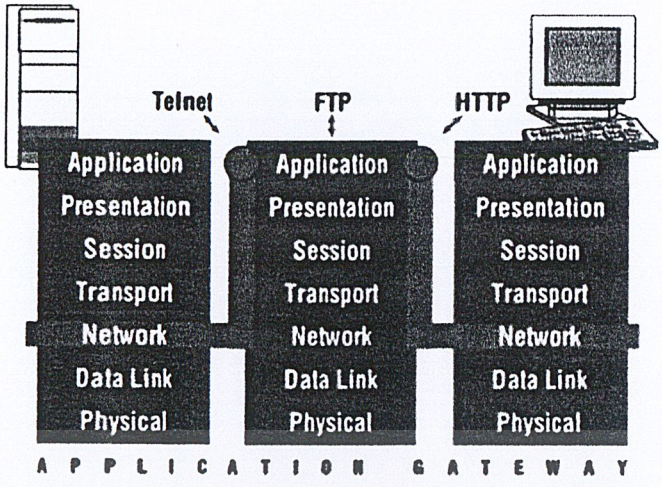
2.1.3 Application Proxy

Application Proxy เป็นโปรแกรมแอพพลิเคชั่นที่ทำงานอยู่บนไฟร์วอลล์ที่กั้นอยู่ระหว่างเน็ตเวิร์ก 2 เน็ตเวิร์ก ทำหน้าที่เพิ่มความปลอดภัยให้กับระบบเน็ตเวิร์ก โดยการควบคุมการเชื่อมต่อระหว่างเน็ตเวิร์กภายนอกและภายใน Proxy จะช่วยเพิ่มความปลอดภัยได้มากเนื่องจากมีการตรวจสอบข้อมูลถึงระดับแอพพลิเคชั่นเลเยอร์ (Application Layer) เลย

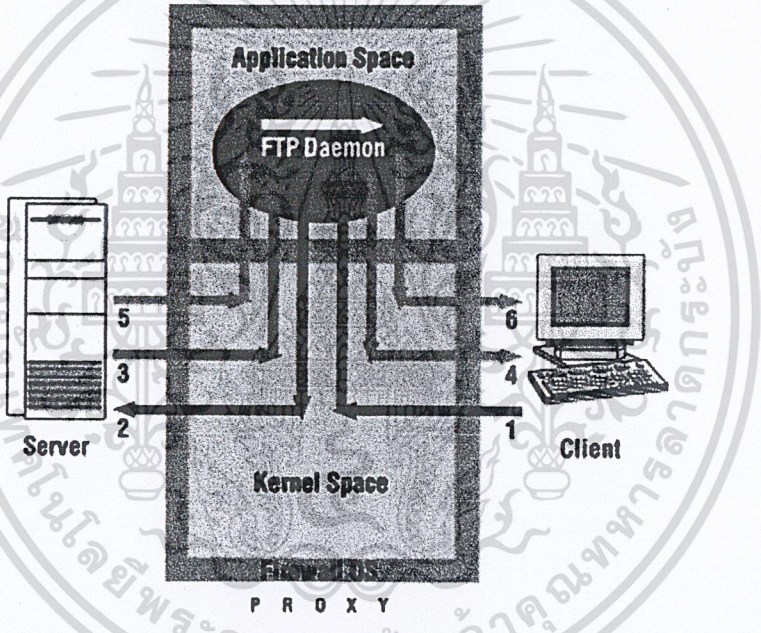
เมื่อไคลเอนต์ต้องการใช้บริการภายนอก ไคลเอนต์จะต้องทำการติดต่อไปยัง Proxy ก่อน จากนั้นไคลเอนต์จะต้องเจรจากับ Proxy เพื่อให้ Proxy ติดต่อไปยังเครื่องปลายทางให้ และเมื่อ Proxy ติดต่อไปยังเครื่องภายทางให้เรียบร้อยแล้ว จะมีการเชื่อมต่อขึ้น 2 การเชื่อมต่อ คือ ไคลเอนต์กับ Proxy และ Proxy กับเครื่องปลายทาง โดยที่ตัว Proxy จะเป็นผู้ทำหน้าที่รับและส่งต่อข้อมูลให้ทั้ง 2 ทิศทาง ทั้งนี้ Proxy จะทำการตัดสินใจว่าจะให้มีการเชื่อมต่อกันหรือไม่ จะส่งต่อแพ็กเก็ตนั้นๆ ให้หรือไม่ โดยพิจารณาจากเฮดเดอร์ของแพ็กเก็ตเทียบกับกฎเกณฑ์ที่กำหนดไว้ และในกระบวนการทำงานนั้น จะมีการพิจารณาลักษณะของการโต้ตอบของโพรโตคอลด้วยว่ามีรูปแบบการโต้ตอบถูกต้องตามมาตรฐานหรือไม่ โดยอาจมีการตรวจสอบลึกลงไปในข้อมูลของแพ็กเก็ตด้วยว่ามีส่วนที่เป็นการโจมตีหรือมุ่งร้ายหรือไม่ด้วยก็ได้ ขึ้นอยู่กับความสามารถของไฟร์วอลล์แต่ละยี่ห้อ



รูปที่ 2-7 การใช้ Proxy Server



รูปที่ 2-8 แสดงเลเยอร์โมเดลของไฟร์วอลล์ชนิด Application Proxy



รูปที่ 2-9 แสดงช่องทางการเข้าออกของแพ็กเก็ตผ่านไฟร์วอลล์ชนิด Application Proxy

ข้อดีของไฟร์วอลล์ชนิดนี้คือ มีความปลอดภัยสูงมาก เพราะพิจารณาข้อมูลถึงในระดับแอพพลิเคชั่นเลเยอร์เลย ส่วนข้อเสียนั้นก็คือ การทำงานของไฟร์วอลล์ชนิดนี้จะช้ามาก เพราะมีการทำงานหลายขั้นตอนมาก และไม่สามารถใช้งานได้กับทุกโพรโตคอล จะสามารถใช้งานได้กับเฉพาะโพรโตคอลที่รู้จักเท่านั้น

2.1.4 Circuit-Level Gateway

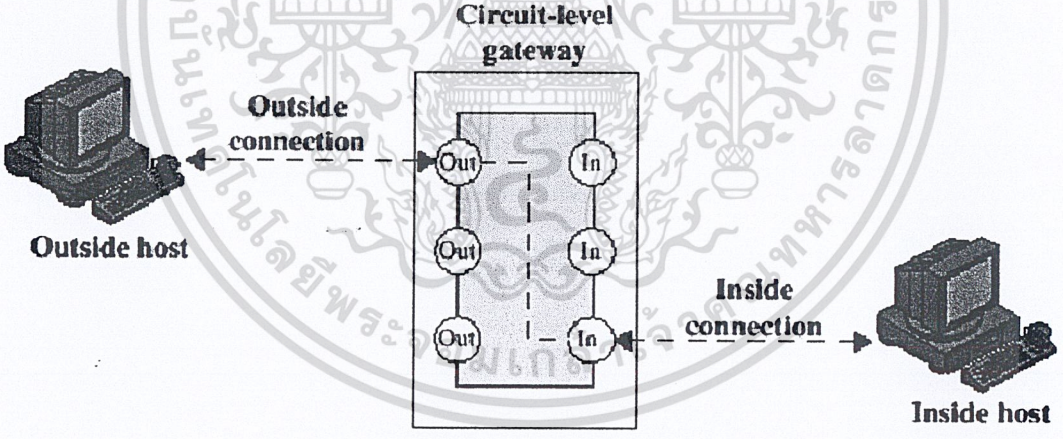
แตกต่างจากไฟร์วอลล์ชนิดอื่นๆ คือ ไฟร์วอลล์ชนิดอื่นๆ จะทำงานในลักษณะของเราเตอร์ หากเครื่องในเน็ตเวิร์กภายในต้องการติดต่อไปเน็ตเวิร์กภายนอก ก็จะทำการติดต่อผ่านไฟร์วอลล์ และเมื่อเน็ตเวิร์กภายนอกต้องการติดต่อเข้ามาเน็ตเวิร์กภายใน ก็ต้องติดต่อผ่านไฟร์วอลล์ด้วยเช่นกัน ซึ่งไฟร์วอลล์จะไม่เข้าไปยุ่งเกี่ยวในส่วนของคอนเนคชันของเครื่องทั้งภายนอกและภายในเลย จะทำแค่ตรวจสอบและพิจารณาว่าจะดร็อปหรือส่งต่อแพ็กเก็ตเท่านั้น

แต่ในไฟร์วอลล์แบบ Circuit-Level Gateway นั้น จะต้องมีการสร้างคอนเนคชันขึ้นมา เพื่อทำการเชื่อมต่อกับไฟร์วอลล์ทั้ง 2 ด้าน และทำหน้าที่กำหนดว่ามีการเชื่อมต่อลักษณะใดบ้างที่ยอมให้เกิดการเชื่อมต่อได้ โดยตัวอย่างของไฟร์วอลล์ชนิดนี้ คือ SOCKS

SOCKS ประกอบด้วย

- SOCKS server ซึ่งทำงานบน UNIX-based firewall
- SOCKS client library ซึ่งทำงานบนโฮสต์ภายในที่ถูกป้องกันโดยไฟร์วอลล์
- SOCKS-ified version ของไคลเอนต์โปรแกรม เช่น FTP และ TELNET

ซึ่งการที่จะสามารถใช้ SOCKS ได้นั้น ทั้งเน็ตเวิร์กภายในและภายนอกจำเป็นต้องรู้จักกับ SOCKS ซึ่งเป็นโพรโตคอลพิเศษนี้ด้วย เพราะจะต้องเชื่อมต่อกับไฟร์วอลล์โดยผ่านทางโมดูลของ SOCKS



รูปที่ 2-10 Circuit-Level Gateway

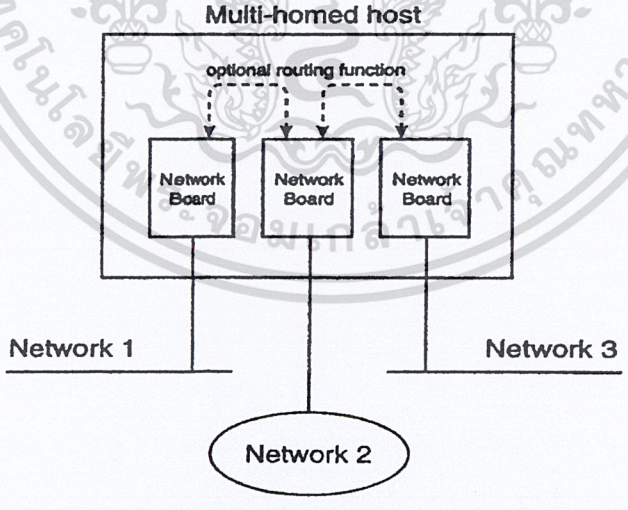
ในปัจจุบันไฟร์วอลล์ประเภทนี้ มีผู้ใช้น้อยมาก (ไม่น่าจะมีใครใช้แล้ว) เพราะว่ามีจุดด้อยอยู่เยอะโดยเฉพาะอย่างยิ่ง การที่แอพพลิเคชั่นต่างๆ จำเป็นต้องรู้จักโพรโตคอลพิเศษที่ใช้อยู่ด้วย จึงจะสามารถใช้งานได้ ส่วนข้อดีของไฟร์วอลล์ชนิดนี้ก็คือ จะให้ความปลอดภัยมากยิ่งขึ้น

2.2 สถาปัตยกรรมของไฟร์วอลล์ (Firewall Architecture)

2.2.1 สถาปัตยกรรมดูอัลโฮมโฮสต์ (Dual-homed host architecture)

สถาปัตยกรรมดูอัลโฮมโฮสต์ (Dual-homed host architecture) คือคอมพิวเตอร์ที่แยกสองเครือข่ายออกจากกันและต้องมีอย่างน้อยสองเน็ตเวิร์กอินเตอร์เฟส อินเตอร์เฟสอันหนึ่งเชื่อมต่อยังภายในเครือข่าย อีกอันเชื่อมต่อไปยังที่อื่นๆ หรืออินเตอร์เน็ต คอมพิวเตอร์นี้อาจจะทำหน้าที่เป็นเราเตอร์ที่เชื่อมระหว่างสองเครื่องข่าย ทั้งยังสามารถหาเส้นทางของไอพีแพ็กเก็ต (IP packet) จากเครือข่ายไปยังที่อื่น การอิมพลีเมนท์ (implement) สถาปัตยกรรมดูอัลโฮมโฮสต์ อาจจะไม่ทำหน้าที่เป็นตัวหาเส้นทางก็ได้ โดยถ้ามีไอพีแพ็กเก็ตมาจากเครือข่ายใดๆ เช่น อินเตอร์เน็ต มันจะไม่ถูกหาเส้นทางโดยตรงไปที่เครือข่ายอื่นๆ (เช่น เครือข่ายที่มีการป้องกัน หรือเครือข่ายภายใน) เครือข่ายภายในไฟร์วออล์สามารถติดต่อกับดูอัลโฮมโฮสต์และเครือข่ายภายนอกไฟร์วอลล์ สามารถติดต่อกับดูอัลโฮมโฮสต์ได้ แต่เครือข่ายเหล่านี้ไม่สามารถติดต่อได้โดยตรง การติดต่อระหว่างไอพีของเครือข่ายเหล่านี้อาจมีบางส่วนที่ถูกป้องกันเป็นได้

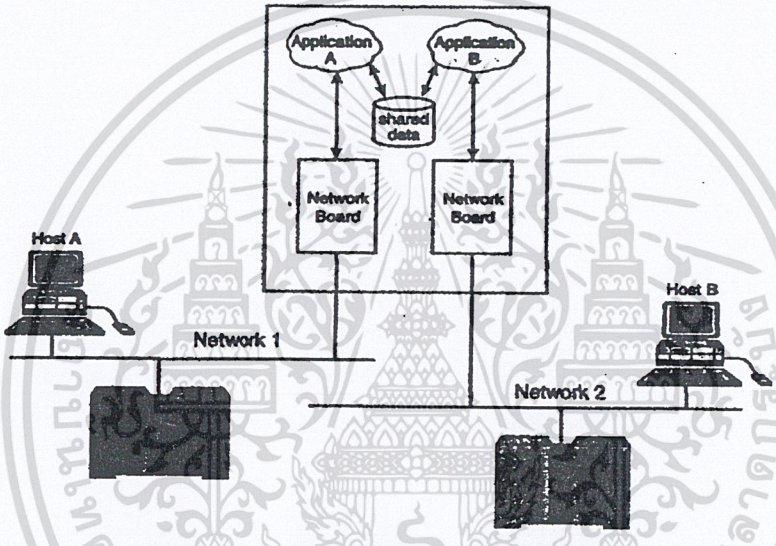
สถาปัตยกรรมดูอัลโฮมโฮสต์ จะเป็นระบบไฟร์วอลล์ที่ค่อนข้างเรียบง่าย ดูอัลโฮมโฮสต์ จะตั้งอยู่ระหว่างการเชื่อมต่อของสองเครือข่าย เช่น เครืออินเตอร์เน็ตและเครือข่ายภายในของเรา ดูอัลโฮมโฮสต์จะให้การควบคุมในระดับสูงๆ เช่น ถ้าเราไม่อนุญาตแพ็กเก็ตให้เข้าออกระหว่างภายนอกกับภายในเครือทั้งหมด เราก็สามารถที่จะมั่นใจได้ว่าภายในเครือข่ายของเรามีความปลอดภัยจากการถูกโจมตีจากนอกระบบได้ ดูอัลโฮมโฮสต์จะให้บริการโฮสต์ที่เชื่อมต่อไปยังเครือข่ายหลายๆ เครือข่าย (แต่การหาเส้นทางได้ถูกปิดลง) ขณะที่สถาปัตยกรรมสกรีนโฮสต์จะให้บริการโฮสต์เพื่อเชื่อมต่อไปยังเครือข่ายภายในเท่านั้น



รูปที่ 2-11 แสดงคลาสสิคมัลติโฮมโฮสต์ (Classic multi-homed host)

ถ้าหน้าที่ในการหาเส้นทาง (routing function) ในมัลติโฮมโฮสต์ไม่สามารถทำงานได้โฮสต์จะทำการแบ่งแยกเส้นทางของเครือข่ายระหว่างเครือข่ายที่มันติดต่ออยู่ และเครือข่ายอื่นๆ ที่สามารถจัดการกับแอพพลิเคชั่น (applications) บนมัลติโฮมโฮสต์โดยเฉพาะอย่างยิ่งถ้าแอพพลิเคชั่นอนุญาตให้เครือข่าย

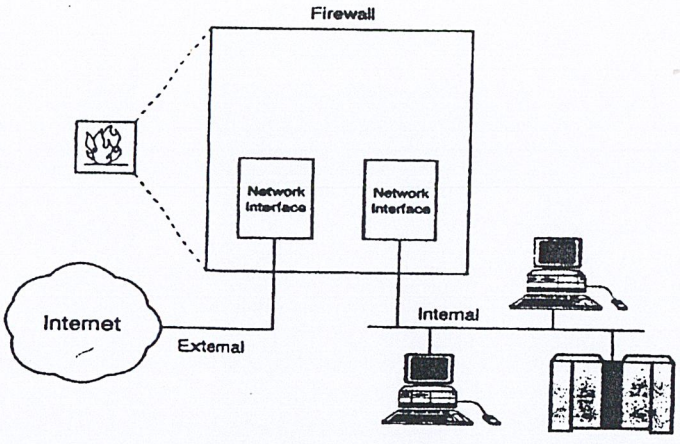
สามารถที่จะแบ่งปันข้อมูล (share data) ได้ ดูอัลโฮมโฮสต์ (Dual-homed host) เป็นตัวอย่างพิเศษของมัลติโฮมโฮสต์ซึ่งมีสองเครือข่ายมาอินเตอร์เฟส (interface) กัน โดยที่ไม่มีการระบุหน้าที่ในการหาเส้นทาง ดังรูปที่ 3-12 แสดงตัวอย่างของดูอัลโฮมโฮสต์ (Dual-homed host) ซึ่งไม่มีการใช้การระบุหน้าที่ในการหาเส้นทาง โฮสต์เอ (host A) อยู่บนเครือข่ายที่หนึ่งซึ่งสามารถเข้าถึงแอพพลิเคชั่นเอบนดูอัลโฮมโฮสต์ที่คล้ายคลึงกัน, โฮสต์บี (host B) สามารถเข้าถึงแอพพลิเคชั่นบี (application B) บนดูอัลโฮมโฮสต์ แอพพลิเคชั่นทั้งสองบนดูอัลโฮมโฮสต์สามารถแบ่งปันข้อมูลได้ จึงเป็นไปได้ว่าโฮสต์เอและ โฮสต์บีสามารถแลกเปลี่ยนข้อมูลผ่านข้อมูลที่ถูกแบ่งปันบนดูอัลโฮมโฮสต์และยังไม่แลกเปลี่ยนเส้นทางของเครือข่ายระหว่างสองส่วนของเครือข่ายที่เชื่อมต่อกับดูอัลโฮมโฮสต์



รูปที่ 2-12 แสดงดูอัลโฮมโฮสต์ (Dual-homed host)

- **ดูอัลโฮมโฮสต์ (Dual-homed host) ที่เป็นไฟร์วอลล์**

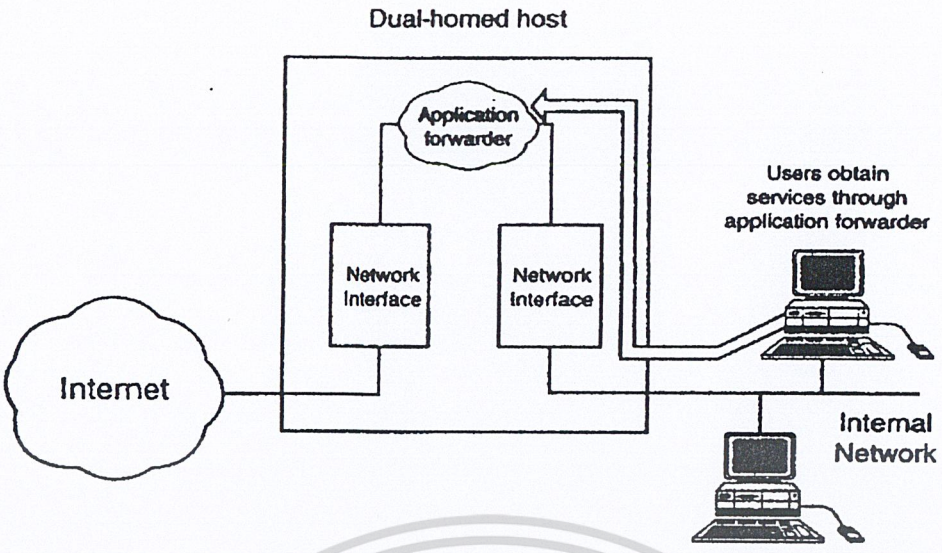
ดูอัลโฮมโฮสต์สามารถแยกเครือข่ายภายใน (ซึ่งมีเครือข่ายอย่างน้อย 2 เครือข่าย) และเครือข่ายภายนอก ดังรูปที่ 3-13 เพราะดูอัลโฮมโฮสต์ไม่ขึ้นกับการส่งแบบทีซีพีไอพี (TCP/IP) มันสามารถขัดขวางทุกๆ ไอพี (IP) ระหว่างเครือข่ายภายในและเครือข่ายภายนอกที่ไม่น่าไว้วางใจ



รูปที่ 2-13 แสดงดูอัลโฮมโฮสต์ (Dual-homed host) ที่เป็นไฟร์วอลล์

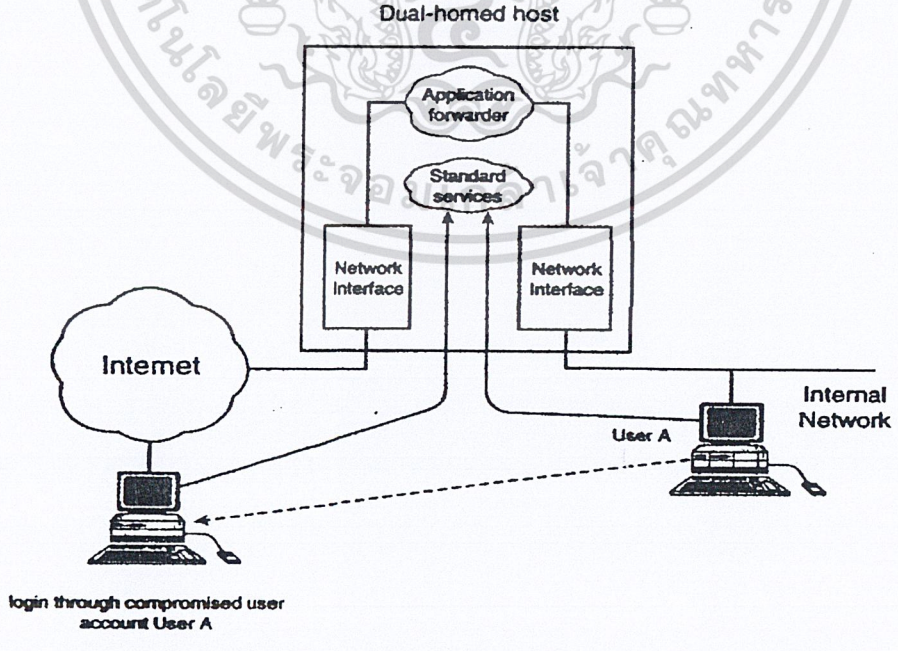
การบริการของอินเตอร์เน็ต เช่น จดหมาย (mail) และข่าวสาร (news) เป็นสิ่งที่จำเป็นพื้นฐานในการบริการเก็บและส่ง ถ้าการบริการเหล่านี้ทำงานบนดูอัลโฮมโฮสต์จะทำให้สามารถกำหนดโครงสร้างในการส่งแอพพลิเคชั่นจากเครือข่ายหนึ่งถึงที่อื่นๆ ถ้าข้อมูลแอพพลิเคชั่นจำเป็นที่จะต้องข้ามไฟร์วอลล์ ผู้ส่งต่อแอพพลิเคชั่น (application forwarder) สามารถติดตั้งให้ทำงานบนดูอัลโฮมโฮสต์ได้ดังรูปที่ 3-14 ผู้ส่งต่อแอพพลิเคชั่น (application forwarder) คือซอฟท์แวร์ (software) พิเศษที่ถูกใช้ส่งระหว่างสองเครือข่ายที่ติดต่อกันการเข้าถึงอย่างอื่น คืออนุญาตให้ผู้ใช้เข้ามาล็อกอิน (login) ในดูอัลโฮมโฮสต์และเข้าถึงบริการภายนอกจากเครือข่ายภายนอกที่เชื่อมติดต่อกับดูอัลโฮมโฮสต์ ดังรูปที่ 3-15

ถ้าผู้ส่งต่อแอพพลิเคชั่น (application forwarders) ที่ถูกใช้การสัญจรของแอพพลิเคชั่นจะไม่สามารถผ่านดูอัลโฮมโฮสต์ไฟร์วอลล์นอกเสียจากว่าผู้ส่งต่อแอพพลิเคชั่นถูกกำหนดไว้บนระบบไฟร์วอลล์ ถ้าผู้ใช้ได้รับการอนุญาตจากไฟร์วอลล์โดยตรง ดังรูปที่ 3-15 ระบบรักษาความปลอดภัยของไฟร์วอลล์สามารถประนีประนอมได้ เพราะว่าดูอัลโฮมโฮสต์ไฟร์วอลล์เป็นจุดศูนย์รวมการติดต่อระหว่างเครือข่ายภายนอกและภายใน ดังนั้นดูอัลโฮมโฮสต์ไฟร์วอลล์ก็จะอยู่ในบริเวณที่มีความเสี่ยง (zone of risk) ถ้าผู้ใช้เลือกรหัสผ่าน (password) ที่อ่อนแอ ในบริเวณที่มีความเสี่ยงก็สามารถขยายมาที่ระบบเครือข่ายภายในทำให้ผิดกับจุดประสงค์ของดูอัลโฮมโฮสต์ไฟร์วอลล์ได้



รูปที่ 2-14 แสดงดูอัลโฮมโฮสต์ที่มีตัวส่งต่อแอพพลิเคชั่น (application forwarder)

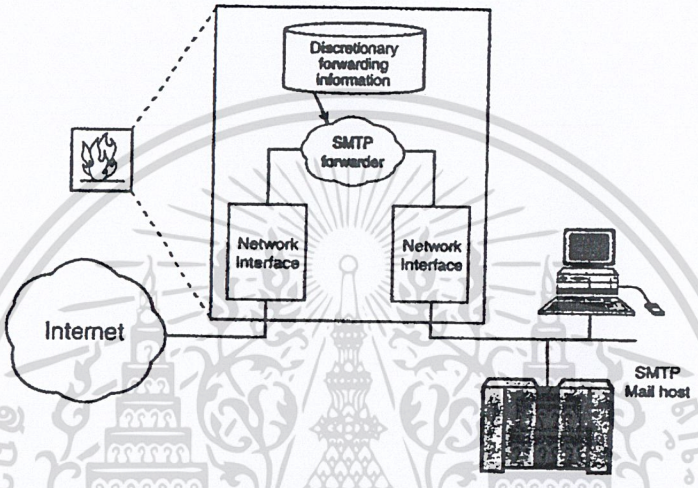
ผู้จัดการระบบรักษาความปลอดภัย จะไม่ทำการสร้างบัญชีของผู้ใช้ (user account) ให้เข้ามาที่ ไฟร์วอลล์ ดังนั้นไฟร์วอลล์ควรจะถูกใช้ได้เฉพาะผู้ใช้ที่มีสิทธิโดยแท้จริงเท่านั้นที่จะผ่านเข้ามาได้



รูปที่ 2-15 แสดงความไม่ปลอดภัยเมื่อผู้ใช้ล็อกอิน (login) เข้ามาในดูอัลโฮมโฮสต์

ถ้าหากมีการเก็บรายละเอียดของผู้ใช้การล็อกอินของผู้ใช้ (users login) ไว้อย่างดี มันจะถูกบุกรุกเมื่อมีการผิดเพี้ยนของระบบรักษาความปลอดภัย ดังนั้นควรที่จะมีการบันทึกไว้ว่ามีใครเข้ามาบ้าง

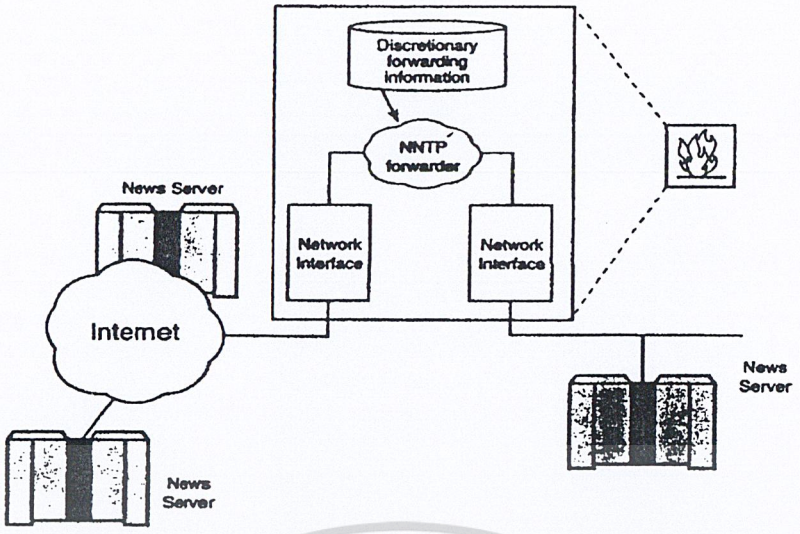
ตัวอย่างของการบริการเก็บและส่ง คือ เอสเอ็มทีพี (SMTP-mail) และเอ็นเอ็นทีพี (NNTP-news) ดังรูปที่ 3-16 แสดงการติดตั้งตัวระมัดระวังในการส่งข้อมูล (discretionary forwarding) ของข้อความเมล์ (mail) ระหว่างเครือข่ายภายนอกที่ไม่น่าไว้วางใจกับเครือข่ายภายใน ดังรูปที่ 3-17 แสดงการติดตั้งตัวระมัดระวังในการส่งข้อมูลสำหรับข้อความข่าวสาร (news) ระหว่างเซิร์ฟเวอร์ข่าวสาร (news Servers) บนเครือข่ายภายนอกและเครือข่ายภายใน



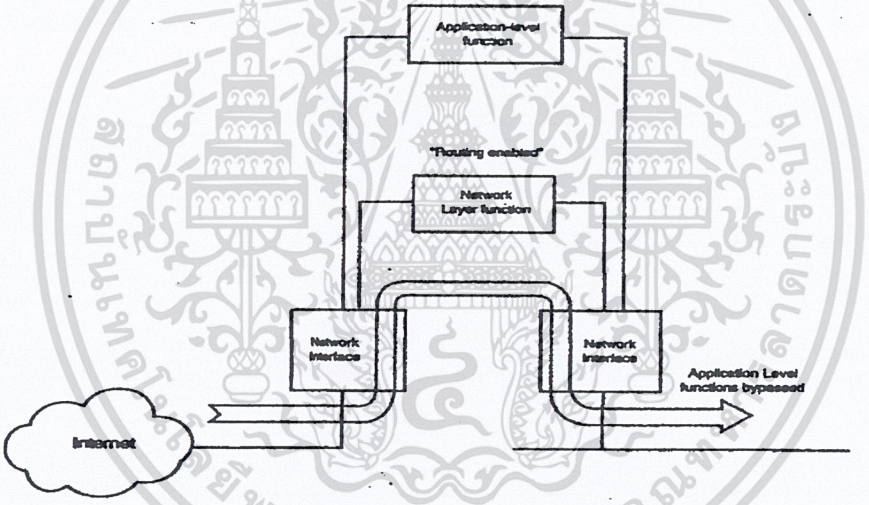
รูปที่ 2-16 แสดงดูอัลโฮมโฮสต์ที่มีตัวส่งต่อเมล์ (mail forwarder)

ดูอัลโฮมโฮสต์เป็นโครงสร้างพื้นฐานที่ใช้ในไฟร์วอลล์ ความสามารถที่ใช้เมื่อฉุกเฉินของดูอัลโฮมโฮสต์ คือ เมื่อการหาเส้นทาง (routing) ใช้ไม่ได้ และมีเพียงเส้นทางระหว่างส่วนหนึ่งของเครือข่าย (network segment) ที่ผ่านทางชั้นแอพพลิเคชั่น (application layer) เมื่อตัวหาเส้นทางมีปัญหา ไอพี (IP) ก็สามารถส่งไปได้ มันคือไม่ผ่านชั้นแอพพลิเคชั่นของดูอัลโฮมโฮสต์ไฟร์วอลล์ ดังรูปที่ 3-18

ไฟร์วอลล์ส่วนมากจะสร้างขึ้นรอบๆ ระบบยูนิกซ์ (UNIX) ซึ่งบนการทำงานของระบบยูนิกซ์นั้นหน้าที่การจัดหาเส้นทาง (rounting function) จะถูกจัดตั้งให้ทำงานได้ ดังนั้นมันจึงเป็นสิ่งสำคัญในการตรวจสอบว่า หน้าที่การจัดหาเส้นทางในดูอัลโฮมโฮสต์ไฟร์วอลล์ทำงานได้หรือไม่ และถ้ามันไม่ทำงานคุณควรจะรู้ว่ามันไม่ทำงานได้อย่างไร



รูปที่ 2-17 แสดงดูอัลโฮมโฮสต์ที่มีตัวส่งต่อข่าวสาร (news forwarder)



รูปที่ 2-18 แสดงการคอนฟิก (config) ที่ผิดพลาดบนดูอัลโฮมโฮสต์ไฟร์วอลล์

- **ข้อดี-ข้อเสียของสถาปัตยกรรมดูอัลโฮมโฮสต์**

  ข้อดีของสถาปัตยกรรมดูอัลโฮมโฮสต์

  - ง่ายต่อการติดตั้ง
  - จะเป็นการควบคุมการติดต่อทั้งหมดที่จุดเดียว ดังนั้นผู้ดูและระบบสามารถควบคุมการการเข้าออกได้โดยง่าย
  - มีความยุ่งยากน้อยต่อกฎการกรอง
  - มีการบันทึกการเข้าออกของเครือข่าย (log)

ข้อเสียของสถาปัตยกรรมดูอัลโฮมโฮสต์

- เพราะดูอัลโฮมโฮสต์จะมีการป้องกันที่จุดเดียวดังนั้น จะเป็นการง่ายที่จะโจมตีเพื่อเข้ามาฝ่ายในเครือข่ายได้โดยง่าย
- ประสิทธิภาพจะลดต่ำลง เมื่อมีการติดตั้งบนเครื่องที่ช้า

2.2.2 สถาปัตยกรรมสกรีนโฮสต์ (Screened host architecture)

สถาปัตยกรรมสกรีนโฮสต์ (Screened host architecture) จะป้องกันและจัดการทางด้านบริการ (service) จากโฮสต์ที่อยู่ภายนอกเครือข่ายที่มีเข้ามาติดต่อกับโฮสต์ภายในเครือข่ายของเรา โดยจะมีการป้องกันคือจะใช้เราเตอร์ซึ่งทำหน้าที่เป็นแพ็กเก็ตฟิลเตอร์ริ่ง (packet filtering) คอยกรองข้อมูลที่เข้าออกของเครือข่ายโดย สถาปัตยกรรมสกรีนโฮสต์ (Screened host architecture) จะมีแบสชั่นโฮสต์ (bastion host) ซึ่งจะเป็นโฮสต์ที่คอยให้บริการพร็อกซี่ (Proxy) เราเตอร์ (rounter) จะทำหน้าที่บังคับให้เครื่องภายในเครือข่าย (network) ที่จะร้องขอบริการต้องข้องขอบริการผ่านพร็อกซี่ โดยจะไม่ยอมให้ติดต่อบริการจากภายนอกเน็ตเวิร์คได้โดยตรง

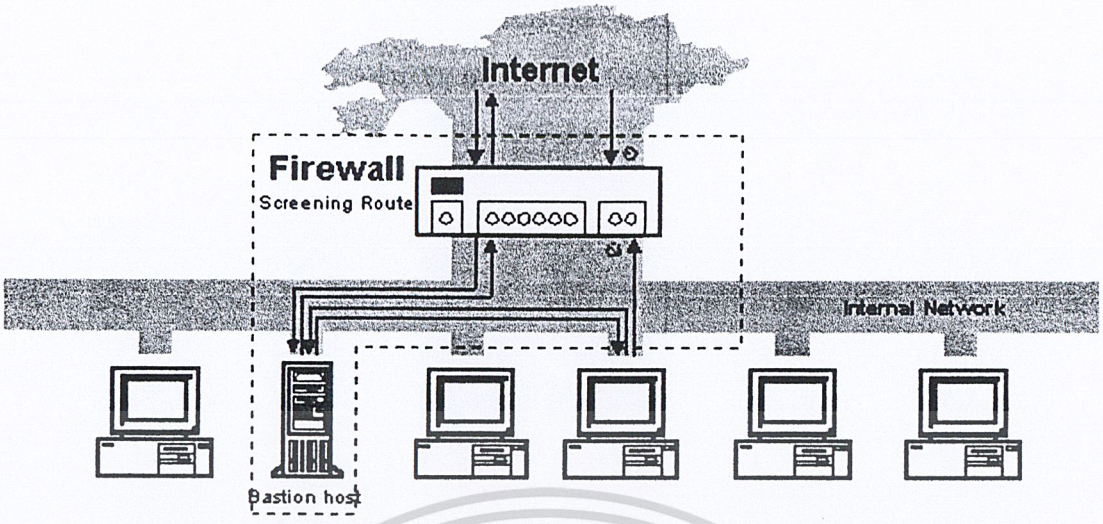
โครงสร้างของแพ็กเก็ตฟิลเตอร์ริ่งในสกรีนนิ่งเราเตอร์อาจจะกระทำอย่างใดอย่างหนึ่งดังต่อไปนี้

- สำหรับโฮสต์ที่แน่นอน จะอนุญาตให้โฮสต์ภายใน (internal host) ติดต่อกับโฮสต์บนอินเตอร์เน็ตได้
- บางบริการอาจมีการติดต่อกับโอสภายนอกได้โดยตรง
- ไม่อนุญาตทุกการติดต่อจากโฮสต์ภายใน
- บริการอื่นๆ จะไม่ยอมให้เครื่องภายในติดต่อผ่านไปภายนอกได้โดยตรง มีเพียง (Bastion host) เท่านั้นที่สามรถติดต่อกับเครือข่ายภายนอกได้โดยตรง ทั้งนี้จะเป็นการบังคับให้ใช้บริการพร็อกซี่ผ่านแบสชั่นโฮสต์เท่านั้น

เราสามารถปรับเปลี่ยนแก้ไขบริการต่างๆ ที่มีการป้องกันเอาไว้ โดยบางบริการอาจจะอนุญาติให้ผ่านแพ็กเก็ตฟิลเตอร์ริ่งโดยตรง และบางบริการก็อาจจะไม่อนุญาติให้ผ่านได้โดยตรงก็ได้ ขึ้นอยู่กับระบบของเราว่าจะมีการเปิดบริการไหนบ้าง

ความไม่ปลอดภัยของสถาปัตยกรรมสกรีนโฮสต์ ถึงแม้สถาปัตยกรรมสกรีนโฮสต์ จะมีทั้งเราเตอร์และพร็อกซี่ ที่ทำหน้าที่เป็น Packet Filtering แต่ก็ไม่ปลอดภัยเพราะว่าเราเตอร์จะยอมให้ติดต่อกับ Bastion Host ซึ่งตรงนี้จะมีความไม่ปลอดภัย

ในสถาปัตยกรรมนี้แพ็กเก็ตผ่านจากอินเตอร์เน็ตเข้าสู่เครือข่ายภายในโดยตรง จึงดูเหมือนว่าจะมีความเสี่ยงมากกว่าดูอัลโฮมโฮสต์ที่ไม่มีแพ็กเก็ตภายนอกเข้าถึงเครือข่ายภายในได้ ในทางปฏิบัติดูอัลโฮมโฮสต์อาจจะเสียหายได้ง่าย ซึ่งทำให้แพ็กเก็ตผ่านเข้ามาได้ นอกจากนี้มันง่ายกว่าในการป้องกันด้วยเราเตอร์ (router) ซึ่งสามารถจำกัดกลุ่มของการบริการมากกว่าป้องกันด้วยโฮสต์ด้วยจุดนี้สกรีนโฮสต์จึงรักษาความปลอดภัยได้ดีกว่าและนิยมใช้มากกว่าดูอัลโฮมโฮสต์



รูปที่ 2-19 แสดงสถาปัตยกรรมแบบสกรีนโฮสต์ (Screened host architecture)

- **ข้อดี-ข้อเสียของสถาปัตยกรรมสกรีนโฮสต์**

  ข้อดีของสถาปัตยกรรมสกรีนโฮสต์

  - มีความยืดหยุ่นมากกว่า dual-home host firewall
  - ความปลอดภัยจะเพิ่มขึ้นอีกเล็กน้อย (อาจกล่าวได้ว่าอยู่ในระดับเดียวกับ dual-home host ก็ได้)

  ข้อเสียของสถาปัตยกรรมสกรีนโฮสต์

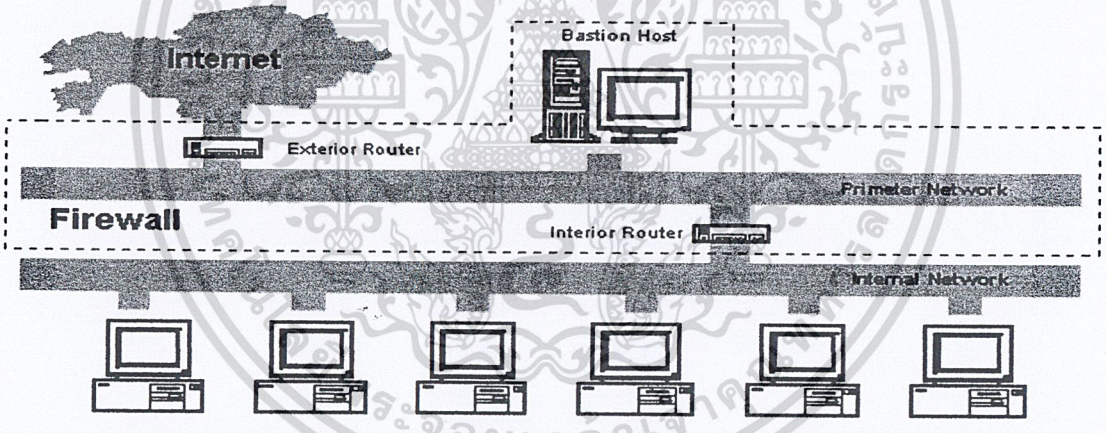
  - บางครั้งอาจต้องการคอมโพเนนท์ที่เพิ่มเติม ทั้งแบสชั่นโฮสและสกีนนิ่งเราเตอร์
  - มันยากต่อการคอนฟิกงานที่ถูกเพิ่มจะถูกเพิ่มเติมที่สกีนนิ่งเราเตอร์
  - มีปัญหา IP spoofing
  - มีความไม่ปลอดภัยที่ให้สามารถเข้าแบสชั่นโฮสต์ได้

  สถาปัตยกรรมสกรีนโฮสต์เหมาะสำหรับ

  - เครือข่ายที่มีการติดต่อกับเครือข่ายภายนอกน้อย
  - เครือข่ายที่ป้องกันภัยในระดับของโฮสต์คือยู่แล้ว

2.2.3 สถาปัตยกรรมสกรีนสับเน็ต (Screened subnet architecture)

สถาปัตยกรรมแบบสกรีนสับเน็ต (Screened subnet architecture) จะเกิดจากการรวมกันขึ้นจากหลายๆ ส่วนที่ประกอบกันขึ้นมาเป็นระบบ ได้เพิ่มชั้นพิเศษของความปลอดภัยบนสถาปัตยกรรมแบบสกรีนโฮสต์โดยเพิ่มชั้นของเครือข่ายเพอริมิเตอร์ (perimeter network) เพื่อแบ่ง เครือข่ายภายในจากอินเตอร์เน็ต โดยวิธีการนี้สามารถเพิ่มความปลอดภัยได้มาก เนื่องจากโดยหากมีการป้องกันเพียงจุดเดียวแล้วถ้าหากเกิดความเสียหากหรือมีสิ่งแปลกปลอมขึ้นมาแล้ว ระบบทั้งหมดก็อาจเป็นอันตรายได้แต่ถ้าหากมีการป้องกันหลายชั้น จะเป็นดารลดความเสี่ยงที่อาจจะเกิดขึ้นกับระบบด้วย โดยถ้าหากชั้นแรกเกิดความเสียหายหรือมีสิ่งแปลกปลอมเข้ามาแล้ว ระบบยังมีอีกชั้นคอยป้องกันระบบของเราจากสิ่งเหล่านั้นไม่ให้ได้มาทำลายระบบของเรา เป็นการเพิ่มความปลอดภัยให้กับระบบของเราด้วยที่อาจถูกโจมตี อย่างไรก็ตามถ้าระบบฟิลเตอร์ริ่ง (filtering) ระหว่างแต่ละชั้นอนุญาตในสิ่งเดียวกัน ชั้นที่เพิ่มขึ้นก็ไม่ได้เพิ่มความปลอดภัยเลย ส่วนประกอบของสถาปัตยกรรมแบบสกรีนสับเน็ต ประกอบด้วยเราเตอร์ 2 ตัว ตัวนึงอยู่ระหว่างอินเตอร์เน็ตกับ Perimeter Network ส่วนอีกตัวหนึ่งอยู่ระหว่าง Perimeter Network กับเน็ตเวิร์กภายใน ถ้าหากจะมีผู้มาเจาะเน็ตเวิร์กภายในต้องผ่านเราเตอร์เข้ามาถึง 2 ตัวด้วยกัน ถึงแม้ว่าจะเจาะชั้นแรกเข้ามายังแบสชั่นโฮสต์ได้ แต่ก็ยังต้องผ่านเราเตอร์ตัวในอีก ถึงจะเข้ามายังเครือข่ายภายในได้ ดั้งนั้นความปลอดภัยของเครื่องภายในเครือข่ายของเราก็มีความปลอดภัยมากขึ้น



รูปที่ 2-20 แสดงสถาปัตยกรรมแบบสกรีนสับเน็ต (Screened subnet architecture) โดยใช้เราเตอร์ 2 ตัว

โดยทั่วไปแบสชั่นโฮสต์จะตั้งอยู่บนเครือข่ายเพอริมิเตอร์ ทำหน้าที่ให้บริการพรอ็กซี่ ซึ่งโดยทั่วไปแล้วจะเป็นเครื่องที่มีความเสี่ยงสูงต่อการถูกโจมตี และมักเป็นเป็นเครื่องที่เปิดบริการให้กับโฮสต์อื่นๆ เราจึงต้องให้ความสำคัญดูแลเป็นพิเศษ ถึงแม้ว่าจะพยายามป้องกันแต่ก็มักจะถูกบุกรุกสำหรับในสกรีนโฮสต์ถ้าแบสชั่นโฮสต์ถูกบุกรุก จะไม่มีอะไรที่จะป้องกันระหว่างมันกับเครื่องกลภายใน ดังนั้นถ้ามีคนบุกรุกเข้าไปในแบสชั่นโฮสต์ของสกรีนโฮสต์ได้ถือว่าเขาได้รางวัลใหญ่ (jackpot)

โดยการแบ่งแบสชั่นโฮสต์ไว้บนเครือข่ายเพอริมิเตอร์ จะสามารถลดผลกระทบที่เกิดจากการที่มีคนบุกรุกแบสชั่นโฮสต์ มันจะช่วยป้องกันผู้บุกรุกในการเข้าถึงแต่ไม่ถึงกับทั้งหมด

- **ข้อดี-ข้อเสียของสถาปัตยกรรมสกรีนสับเน็ต**

ข้อดีของสถาปัตยกรรมสกรีนสับเน็ต

- การป้องกันที่ยืดหยุ่นมากกว่า เพราะทั้งแพ็กเก็ตฟิลเตอร์ริ่งและพร็อกซีเซิร์ฟเวอร์จะถูกใช้ การเลือกจะใช้ในการติดต่ออาจใช้ทั้งอันใดอันหนึ่งหรือทั้งสอง
- มีระดับความปลอดภัยที่สูง ผู้บุกรุกต้องผ่านทั้งหมดสามชั้นก่อนที่จะเข้าเครือข่ายภายในได้
- สนับสนุนการซ่อนเน็ตเวิร์ก (network hiding) เพราะมีแค่เน็ตเวิร์ก DMZ เท่านั้นที่จำเป็นต้องเปิดเผยต่อผู้ใช้อินเตอร์เน็ต

ข้อเสียของสถาปัตยกรรมสกรีนสับเน็ต

- ต้องการทรัพยากรเป็นพิเศษ คือต้องการทั้งแบสชั่นโฮสและเราเตอร์สองตัวคือสกีนนิ่งเราเตอร์
- ยากต่อการคอนฟิก สำหรับสกีนสับเน็ตที่จะทำงานได้นั้น ต้องมีการคอนฟิกสกีนนิ่งเราเตอร์ทั้งสองตัว

บทที่ 3

IPtables

เป็นทูล Built-In ของลีนุกซ์ที่มีเคอร์เนลตั้งแต่ 2.4 เป็นต้นไป ใช้สำหรับสร้าง ควบคุม และตรวจสอบกฎเกณฑ์ในการกรองข้อมูลที่ผ่านไปมาระหว่างเน็ตเวิร์กภายนอกและเน็ตเวิร์กภายใน และยังป้องกันเน็ตเวิร์กและข้อมูลสำคัญๆ จากผู้ที่ไม่ได้รับอนุญาตที่แอบเข้ามาในระบบเน็ตเวิร์กของเรา

หน้าที่หลักที่สำคัญของ IPtables ที่ต้องทำ คือ

- ให้สิทธิข้อมูลที่ได้รับอนุญาตให้เข้ามาถึงที่หมายได้และปฏิเสธข้อมูลที่ไม่ได้รับอนุญาตให้เข้ามาในเน็ตเวิร์ก หรือเครื่องคอมพิวเตอร์ภายในเน็ตเวิร์ก
- ให้บริการในลักษณะของยามรักษาการอิเล็กทรอนิกส์ ที่คอยหยุดความพยายามของแฮกเกอร์หรือบุคคลใดก็ตามที่พยายามเจาะเข้ามายังเน็ตเวิร์กของเรา จากเครื่องคอมพิวเตอร์ที่ไม่รู้จัก (Untrusted Computer) บนเน็ตเวิร์กภายนอก

3.1 CHAINS

Chain แบ่งออกเป็น 4 chain ดังนี้

- **INPUT**
  - ใช้ตรวจสอบแพ็กเก็ตที่มีแอดเดรสปลายทางอยู่ที่เครื่องไฟร์วอลล์ (แพ็กเก็ตที่ถูกส่งมาให้ไฟร์วอลล์)
- **OUTPUT**
  - ใช้ตรวจสอบแพ็กเก็ตที่มีแอดเดรสต้นทางเป็นเครื่องไฟร์วอลล์ (แพ็กเก็ตที่ไฟร์วอลล์ส่งออกมา)
- **FORWARD**
  - ใช้ตรวจสอบแพ็กเก็ตที่ไม่มีแอดเดรสต้นทางหรือปลายทางตรงกับเครื่องไฟร์วอลล์ (แพ็กเก็ตที่ส่งผ่านไฟร์วอลล์)
- **User-Defined chain**
  - เป็นกลุ่มของกฎเกณฑ์ที่ผู้ใช้เป็นผู้กำหนดขึ้นเอง

โดยที่แต่ละกลุ่มของกฎเกณฑ์นั้นจะถูกนำมาใช้แยกกัน ถ้าตรวจสอบแล้วไม่ตรงกับกฎแรก ก็จะเลื่อนไปตรวจสอบการทำงานกับกฎต่อๆ ไป และถ้าไม่ตรงกับกฎใดๆ เลย ก็จะไปตรวจสอบที่นโยบาย (Policy) ของ chain นั้นๆ ซึ่งโดยทั่วๆ ไปแล้วจะเป็นการครอปแพ็กเก็ตนั้นทิ้งไป

หมายเหตุ: INPUT, OUTPUT และ FORWARD ถือเป็น Built-In chain

3.2 TABLES

-t, –table

ใช้สำหรับระบุ packet matching table โดยที่แบ่งออกเป็น 3 table ดังนี้

- **filter**
  - เป็น default table ประกอบด้วย Built-In chain คือ INPUT, OUTPUT และ FORWARD
- **nat**
  - เป็น table ที่ใช้สำหรับการทำ Network Address Translation (NAT) ประกอบด้วย Built-In chain คือ PREROUTING, POSTROUTING และ OUTPUT
- **mangle**
  - เป็น table ที่ใช้สำหรับการเปลี่ยนแปลงแพ็กเก็ตแบบพิเศษ ประกอบด้วย Built-In chain คือ PREROUTING และ OUTPUT

3.3 TARGETS (สำหรับ filter table)

การตรวจสอบแพ็กเก็ต iptables จะนำรายละเอียดของแพ็กเก็ตไปตรวจสอบกับกฎต่างๆ ที่กำหนดเอาไว้ตามลำดับ โดยถ้าหากว่าไม่ตรงกับกฎนั้น ก็จะเลื่อนไปตรวจสอบกับกฎในลำดับต่อไป แต่หากว่ารายละเอียดของแพ็กเก็ตตรงกับกฎนั้น iptables จะปฏิบัติกับแพ็กเก็ตนั้น ตามที่ระบุไว้ตรงส่วนของ target

Target สามารถระบุได้หลายอย่าง ทั้งแบบที่กำหนดเอง (User-Defined chain) หรือเลือกเอาหนึ่งใน Built-In chain ต่อไปนี้

- **ACCEPT**
  - ส่งผ่านแพ็กเก็ตนั้นต่อไปยังที่ระบุไว้ในแอดเดรสปลายทาง
- **DROP**
  - ทิ้ง (ลบ) แพ็กเก็ตนั้นไปจากเน็ตเวิร์ก
- **REJECT**
  - ทิ้งแพ็กเก็ตเช่นเดียวกับ DROP แต่จะมีการส่งข้อความกลับไปที่โฮสต์ต้นทาง เพื่อให้โฮสต์ต้นทางทราบว่าแพ็กเก็ตที่ส่งมานั้น โดนลบทิ้ง
- **QUEUE**
  - เป็น target พิเศษ ใช้สำหรับส่งแพ็กเก็ตไปรอคิวสำหรับ userspace processing นอกจากว่ามีบางโปรแกรมกำลังรอแพ็กเก็ตนั้นอยู่ แพ็กเก็ตนั้นก็จะถูกดรอปทิ้งเลย
- **RETURN**
  - ถ้าเป็น Built-In chain เช่น ‘iptables –A FORWARD –p tcp –j RETURN’ จะปฏิบัติต่อแพ็กเก็ตเสมือนว่าไม่มีกฎใดที่ตรงเลย นั่นคือ จะไปใช้ policy ของ chain แทน

- ถ้าเป็น User-Defined chain เช่น 'iptables –A my-new-chain –p tcp –j RETURN' จะส่งแพ็กเก็ตกลับไปตรวจสอบกับกฎต่อไปใน chain ก่อนหน้านี้แทน

- **LOG**
  - บันทึกข้อมูลลง log

3.4 OPTIONS

- **COMMANDS (สำหรับจัดการกับ chain)**

-h, –help

Help ดูคำอธิบาย และ syntax ของ command ต่างๆ

Example: *iptables -h*

-N, –new-chain

สร้าง chain ใหม่ (User-Defined chain) ขึ้นมา

Example: *iptables –N my-new-chain*

-X, –delete-chain

ลบ chain ที่สร้างขึ้นเอง (User-Defined chain) ซึ่งไม่มีกฎเกณฑ์ใดๆ อยู่เลย และต้องไม่เป็น target ของกฎเกณฑ์ใดๆ ด้วย (ไม่สามารถลบ Built-In chain ได้)

Example: *iptables –X my-new-chain* *#ลบ my-new-chain*

iptables –X *#ลบ chain ทั้งหมดที่ถูกสร้างขึ้น และสามารถลบได้*

-E, –rename-chain

เปลี่ยนชื่อของ chain

Example: *iptables –E old-name new-name*

-P, –policy

เปลี่ยน policy ของ chain สำหรับกรณีที่รายละเอียดของแพ็กเก็ตไม่ตรงกับกฎเกณฑ์ใดๆ เลย

Example: *iptables –P DROP* *#ดร็อปทุกแพ็กเก็ต*

iptables –P FORWARD DROP *#ดร็อปทุกแพ็กเก็ตที่ฟอร์เวิร์ด*

-L, –list

แสดงกฎเกณฑ์ทั้งหมดที่อยู่ใน chain

Example: *iptables -L*

-F, –flush

ลบกฎเกณฑ์ภายใน chain นั้นๆ

Example: *iptables –F* *#ลบกฎเกณฑ์ทั้งหมดที่อยู่ในทุกๆ chain*

iptables –F FORWARD *#ลบกฎเกณฑ์ทั้งหมดในกลุ่ม FORWARD*

-Z, –zero

รีเซ็ตค่าตัวนับจำนวนกฎเกณฑ์ให้เป็น 0

Example: *iptable –Z FORWARD*

- **COMMANDS (สำหรับจัดการกับกฎเกณฑ์ต่างๆ ภายใน chain)**

-A, –append

เพิ่มกฎเข้าไปที่ลำดับท้ายสุดใน chain

Example: *iptables –A FORWARD –p tcp –j ACCEPT*

-I, –insert

แทรกกฎเข้าไปยังลำดับที่ระบุด้วยหมายเลขที่กำหนด (ถ้าหมายเลขเป็น 1 จะทำให้กฎนั้นเป็นกฎแรกสุดของ chain เลย)

Example: *iptables –I FORWARD 1 –p udp –j ACCEPT*

-R, –replace

นำกฎใหม่ไปแทนที่กฎเดิมตรงลำดับที่ระบุด้วยหมายเลขที่กำหนด

Example: *iptables –R FORWARD 1 –p udp –j DROP*

-D, –delete

ลบกฎภายในกลุ่มของกฎเกณฑ์ตรงตำแหน่งที่ระบุด้วยหมายเลขที่กำหนด หรือกฎแรกที่ตรงกับที่กำหนดไว้

Example: *iptables –D FORWARD 1* *#ลบกฎแรกในกลุ่ม FORWARD*

iptables –D FORWARD –p udp –j DROP *#ลบกฎแรกที่ตรง*

ข้อสังเกต: หากเราเพิ่มกฎด้วย option '-A' แล้ว เราสามารถลบกฎข้อนั้นได้ด้วย Option '-D' ที่เป็น mirror กัน เช่น

iptables –A INPUT –p icmp –j DROP #กฎที่เพิ่มด้วย Option '-A'

iptables –D INPUT –p icmp –j DROP #ลบกฎที่เพิ่มเข้าไปได้โดยใช้ mirror

- **PARAMETERS**

Parameter ต่อไปนี้สามารถนำไปใช้เพื่อการระบุข้อแม้ต่างๆ สำหรับกฎเกณฑ์

-p, –protocol [!] *protocol-name*

สำหรับระบุโพรโตคอลของแพ็กเก็ต โดยสามารถระบุได้ 4 แบบ คือ *tcp*, *udp*, *icmp* หรือ *all* และสามารถใช้ "!" เพิ่มที่ด้านหน้าของ *protocol-name* เพื่อให้มีความหมายในทางตรงกันข้ามได้

Example: *iptables –A FORWARD –p tcp –j ACCEPT*

-s, –source, –src [!] *address*[*/mask*]

สำหรับระบุแอดเดรสต้นทางของแพ็กเก็ต

Example: *iptables –A FORWARD –s 161.246.5.119 –j ACCEPT*

-d, –destination, dst [!] *address*[*/mask*]

สำหรับระบุแอดเดรสปลายทางของแพ็กเก็ต

Example: *iptables –A FORWARD –d 161.246.5.120 –j DROP*

หมายเหตุ:

แอดเดรสที่ใช้ระบุนั้น สามารถระบุได้ 4 รูปแบบด้วยกัน ดังนี้

1. ระบุเป็นชื่อเต็ม เช่น '*localhost*'
   '*www.ce.kmitl.ac.th*'
2. ระบุเป็นไอพีแอดเดรส เช่น '*127.0.0.1*'
   '*161.246.4.2*'
3. ระบุเป็นช่วงของไอพีแอดเดรส โดยใช้การระบุซับเน็ตมาส์คแบบย่อ
   เช่น '*161.246.5.0/24*' #ระบุช่วงแอดเดรสตั้งแต่ 161.246.5.0 จนถึง
   #161.246.5.255 โดยมีซับเน็ตมาส์คเป็น '1'
   #24 บิต (255.255.255.0)
4. ระบุเป็นช่วงของไอพีแอดเดรส โดยใช้การระบุซับเน็ตมาส์คแบบเต็ม

–tcp-flag [!] *mask comp*

สำหรับระบุแฟลกของแพ็กเก็ตที่ต้องการตรวจสอบ โดยแบ่งเป็น 2 ชุด โดยที่แฟลกชุดแรกจะหมายถึง ชุดแฟลกที่เราต้องการตรวจสอบ และแฟลกชุดที่สองจะหมายถึง ชุดแฟลกที่ถูกเซต ซึ่งสามารถระบุได้ด้วย *SYN*, *ACK*, *FIN*, *RST*, *URG*, *PSH*, *ALL* และ *NONE*

Example: *iptables –A INPUT –p tcp –tcp-flag ALL SYN,ACK –j DROP*

[!] **–syn**

สำหรับระบุแพ็กเก็ตที่การร้องขอ TCP connection ซึ่งจะเป็นแพ็กเก็ตที่ SYN flag ถูกเซต และ ACK flag กับ FIN flag ถูกเคลียร์ (มีความหมายเช่นเดียวกับการใช้ '*--tcp-flag SYN,ACK,FIN SYN*')

Example: *iptables –A FORWARD –p tcp –syn –j ACCEPT*

–tcp-option [!] *number*

สำหรับระบุแพ็กเก็ตที่มีการเซตค่า TCP option (ข้อมูลเพิ่มเติม) เป็นหมายเลขตามที่กำหนด ซึ่งแพ็กเก็ตใดมี TCP header ที่ไม่สมบูรณ์ก็จะถูกดร็อปไปโดยอัตโนมัติ

Example: *iptables –A FORWARD –p tcp –tcp-option 536 –j ACCEPT*

- **UDP EXTENSIONS**

UDP Extension จะถูกโหลดขึ้นมา ถ้ามีการระบุ '-p udp' หรือ '-protocol udp' และไม่มีกฎเกณฑ์ใดที่ตรงกันอีกแล้ว

–sport, –source-port [!] [*port*[*:port*]]

ดูคำอธิบายได้ในส่วน **–sport, –source-port** [!] [*port*[*:port*]] ของ **3.5.1 TCP EXTENSIONS** เพราะเหมือนกัน

Example : *iptables –A OUTPUT –p udp --sport 1077 –j ACCEPT*

–dport, –destination-port [!] [*port*[*:port*]]

ดูคำอธิบายได้ในส่วน **–dport, –destination-port** [!] [*port*[*:port*]] ของ **3.5.1 TCP EXTENSIONS** เพราะเหมือนกัน

Example: *iptables –A INPUT –p udp --dport 12345 –j DROP*

- **ICMP EXTENSION**

ICMP Extension จะถูกโหลดขึ้นมา ถ้ามีการระบุ '-p icmp' หรือ '-protocol icmp' และไม่มีกฎเกณฑ์ใดที่ตรงกันอีกแล้ว

--icmp-type [!] *type*

สำหรับระบุแพ็กเก็ตที่มี ICMP type ตรงตาม type ที่กำหนด ซึ่งสามารถกำหนดเป็นหมายเลขหรือชื่อ type ก็ได้

Example: *iptables –A FORWARD –p icmp --icmp-type 3 –j DROP*

หมายเหตุ: สามารถเรียกดู ICMP type ต่างๆ ได้ด้วย '*iptables –p icmp –h*'

- **OTHER MATCH EXTENSIONS**

Extension กลุ่มนี้ จะต้องขึ้นด้วย **–m** หรือ **–match**

mac

--mac-source [!] *address*

สำหรับระบุแมคแอดเดรสของอีเธอร์เน็ตดีไวซ์ต้นทางของแพ็กเก็ต ซึ่งจะอยู่ในรูปแบบ XX:XX:XX:XX:XX:XX โดยการระบุในแบบนี้ จะใช้ได้กับ INPUT, FORWARD หรือ PREROUTING chain เท่านั้น

Example: *iptables -A INPUT -m mac --mac-source 00:60:67:30:AC:E5 -j DROP*

limit

สำหรับระบุจำนวนการแมทช์กับกฎนั้นๆ โดยมักใช้ควบคู่กับ LOG target เพื่อใช้จำกัดจำนวนการบันทึก log

–limit *rate*

สำหรับค่าเฉลี่ยของจำนวนการแมทช์กับกฎนั้นๆ ต่อช่วงเวลาหนึ่งๆ โดยสามารถเลือกช่วงเวลาได้หลายแบบ ดังนี้ '*/second*', '*/minute*', '*/hour*' หรือ '*/day*' โดยค่า default คือ *3/hour*

หมายเหตุ: '*/s*' = '*/second*'
'*/m*' = '*/minute*'
'*/h*' = '*/hour*'

–limit-burst *number*

สำหรับระบุค่าสูงสุดของจำนวนการแมทช์กับกฎนั้นๆ โดยจะถูกรีชาร์จทีละหนึ่ง ทุกๆ ครั้งที่เรทการแมทช์กันของกฎนั้นๆ น้อยกว่าที่ระบุไว้โดย **–limit** ซึ่งค่า default คือ *5*

Example: *iptables –A INPUT –m limit --limit 3/m --limit-burst 3 –j LOG*

multiport

สำหรับระบุพอร์ตหลายๆ พอร์ตในกฎ ซึ่งสามารถระบุถึง 15 พอร์ตต่อหนึ่งกฎ แต่สามารถใช้งานได้กับโพรโตคอล TCP และ UDP เท่านั้น

–sport, –source-port [*port*[,*port*]]

สำหรับระบุพอร์ตต้นทางของแพ็กเก็ต

Example: *iptables –A FORWARD –p tcp –m multiport –sport 21,23 –j DROP*

–dport, –destination-port [*port*[,*port*]]

สำหรับระบุพอร์ตปลายทางของแพ็กเก็ต

Example: *iptables –A FORWARD –p udp –m multiport --dport 21,23 –j ACCEPT*

–port [*port*[,*port*]]

สำหรับระบุพอร์ตต้นทางและปลายทางของแพ็กเก็ต

Example: *iptables –A FORWARD –p tcp –m multiport --port 123,1234 –j DROP*

state

สำหรับระบุสถานะการเชื่อมต่อของแพ็กเก็ต โดยสามารถระบุเป็นชุดได้โดยใช้ ‘,’ คั่น

–state *state*

สามารถเลือกระบุสถานะการเชื่อมต่อของแพ็กเก็ต ได้ดังนี้

- **INVALID**
  - สำหรับแพ็กเก็ตที่ไม่ทราบสถานะการเชื่อมต่อ
- **ESTABLISHED**
  - สำหรับแพ็กเก็ตที่อยู่ในสถานะที่มีการสร้างการเชื่อมต่อระหว่างทั้งสองฝั่งเรียบร้อยแล้ว

- **NEW**
  - สำหรับแพ็กเก็ตที่ใช้เริ่มต้นเชื่อมต่อใหม่
- **RELATED**
  - สำหรับแพ็กเก็ตที่ใช้เริ่มต้นเชื่อมต่อใหม่ แต่เกี่ยวข้องกับการเชื่อมต่อที่มีอยู่แล้ว เช่น ICMP error

    Example: *iptables -A FORWARD –m state --state NEW,ESTABLISHED –j DROP*

unclean

โมดูลนี้ ไม่มี option ใดๆ เพียงใช้สำหรับระบุแพ็กเก็ตที่ดูแปลก ไม่ปกติ โดยพิจารณาจากผลที่ได้จากการทดสอบ

Example: *iptables –A FORWARD –m unclean –j DROP*

3.6 TARGET EXTENSIONS

- **LOG**

–log-level ***level***

บันทึก log เฉพาะ level ที่กำหนดเท่านั้น

Example: *iptables –A FORWARD –d 127.0.0.1 –j LOG --log-level 7*

–log-prefix ***prefix***

ใส่ prefix ที่กำหนดไปที่ log message ด้วย เพื่อให้เข้าใจและสังเกตได้ง่าย โดย prefix สามารถยาวได้ถึง 14 ตัวอักษร

Example: *iptables –A INPUT –p tcp --dport 12345 –j LOG --log-prefix "Netbus Scan"*

–log-tcp-sequence

บันทึก TCP sequence number ด้วย การใช้การบันทึก log แบบนี้ เสี่ยงต่อการเกิดอันตรายทางด้านความปลอดภัยได้ เพราะอาจมีผู้ไม่ประสงค์ดีมาอ่านข้อมูลตรงนี้ได้

Example: *iptables –A FORWARD –p tcp –j LOG --log-tcp-sequence*

–log-tcp-options

บันทึก option ที่อยู่ใน TCP header ของแพ็กเก็ตด้วย

Example: *iptables –A FORWARD –p tcp –j LOG --log-tcp-options*

--log-ip-options

บันทึก option ที่อยู่ใน IP header ของแพ็กเก็ตด้วย

Example: *iptables –A FORWARD –p tcp –j LOG --log-ip-options*

- **MARK**

--set-mark *mark*

สำหรับเซตค่า *netfilter mark* ที่เกี่ยวข้องกับแพ็กเก็ต ใช้ได้เฉพาะกับ mangle table เท่านั้น

Example: *iptables -t mangle -p tcp --dport 80 -j MARK --set-mark 2*

- **REJECT**

--reject-with *type*

สำหรับระบุชนิดของข้อความที่ส่งกลับไปตอน REJECT โดยมีชนิดของข้อความที่สามารถระบุได้ ดังนี้

- *ICMP error message (ค่า default คือ icmp-port-unreachable)*
  - **icmp-net-unreachable**
  - **icmp-host-unreachable**
  - **icmp-port-unreachable**
  - **icmp-proto-unreachable**
  - **icmp-net-prohibited**
  - **icmp-host-prohibited**
- *Other*
  - **echo-reply**
  - **tcp-reset**

Example: *iptables –A INPUT –d 192.168.1.0 –j REJECT --reject-with icmp-net-unreachable*

- **TOS**

--set-tos *tos*

สำหรับเซตค่าให้กับส่วน Type Of Service (8-bit) โดยจะเซตเป็นตัวเลขหรือชื่อของ TOS ก็ได้ และใช้ได้เฉพาะกับ mangle table เท่านั้น

Example: *iptables –t mangle –A OUTPUT –p tcp --dport 80 –j TOS --set-tos 8*

หมายเหตุ: สามารถดูรายชื่อของ TOS ที่ได้ โดยใช้ '*iptables –j TOS –h*'

- **MIRROR**

ทำการสลับค่าของแอดเดรสต้นทางและแอดเดรสปลายทางใน IP header แล้วทดลองส่งแพ็กเก็ตออกไป ใช้เพื่อพิสูจน์ผล สามารถใช้ได้กับ INPUT, OUTPUT, FORWARD chain หรือ User-Defined chain ที่ถูกเรียกจาก chain เหล่านั้นเท่านั้น

Example: *iptables –A FORWARD –p tcp –j MIRROR*

- **SNAT**

เป็น target ที่ใช้ได้กับ nat table และ POSTROUTING chain เท่านั้น ใช้สำหรับระบุแอดเดรสต้นทางของแพ็กเก็ตที่ควรจะต้องถูกแก้ไข

–to-source <*ipaddr*>[-<*ipaddr*>][*:port-port*]

สำหรับระบุไอพีแอดเดรสต้นทางใหม่ของแพ็กเก็ต โดยสามารถระบุเป็นช่วงของแอดเดรสได้ และสามารถระบุพอร์ตเพิ่มเติมได้อีกด้วย (การระบุพอร์ตใช้ได้กับโพรโตคอล TCP หรือ UDP เท่านั้น)

Example: *iptables –t nat –A POSTROUTING –s 192.168.0.7 –j SNAT --to-source 161.246.5.7*

- **DNAT**

เป็น target ที่ใช้ได้กับ nat table และ PREROUTING, OUTPUT chain หรือ User-Defined chain ที่ถูกเรียกจาก chain เหล่านั้นเท่านั้น ใช้สำหรับระบุแอดเดรสปลายทางของแพ็กเก็ตที่ควรจะต้องถูกแก้ไข

–to-destination <*ipaddr*>[-<*ipaddr*>][*:port-port*]

สำหรับระบุไอพีแอดเดรสปลายทางใหม่ของแพ็กเก็ต โดยสามารถระบุเป็นช่วงของแอดเดรสได้ และสามารถระบุพอร์ตเพิ่มเติมได้อีกด้วย (การระบุพอร์ตใช้ได้กับโพรโตคอล TCP หรือ UDP เท่านั้น)

Example: *iptables –t nat –A OUTPUT –d 161.246.5.7 -j DNAT --to-destination 192.168.0.7*

- **MASQUERADE**

เป็น target ที่ใช้ได้กับ nat table และ POSTROUTING chain เท่านั้น ใช้สำหรับการกำหนดไดนามิกไอพีแอดเดรสในการเชื่อมต่อแบบ Dial-Up โดยถ้าต้องการกำหนดสแตติกไอพีแอดเดรสควรใช้ SNAT target

--to-ports <*port*>[-<*port*>]

สำหรับระบุพอร์ตต้นทาง โดยสามารถระบุเป็นช่วงของพอร์ตได้ด้วย แต่สามารถใช้ได้กับโพรโตคอล TCP หรือ UDP เท่านั้น

Example: *iptables –t nat –A POSTROUTING –p tcp *
–s 192.168.1.0/24 –j MASQUERADE --to-ports 1200-1455

- **REDIRECT**

เป็น target ที่ใช้ได้กับ nat table และ PREROUTING, OUTPUT chain หรือ User-Defined chain ที่ถูกเรียกจาก chain เหล่านั้นเท่านั้น สำหรับเปลี่ยนแปลงแอดเดรสปลายทางเพื่อให้ส่งแพ็กเก็ตให้ตัวเอง นั่นคือแอดเดรสปลายทางถูกเปลี่ยนเป็น 127.0.0.1 นั่นเอง

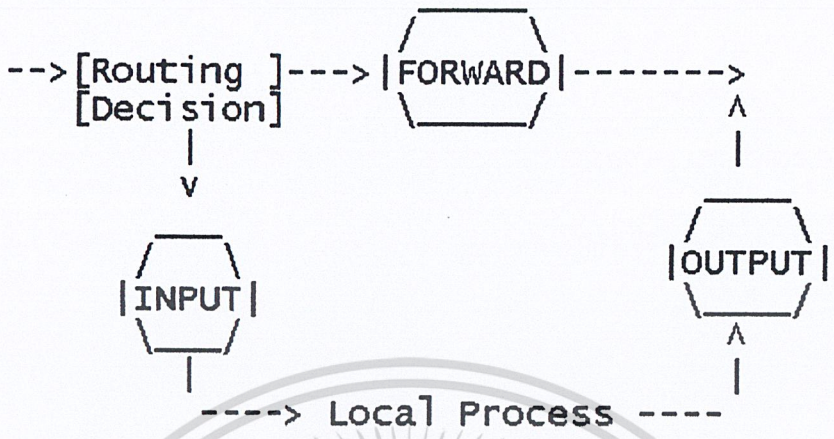
--to-ports <*port*>[-<*port*>]

สำหรับระบุพอร์ตปลายทาง โดยสามารถระบุเป็นช่วงของพอร์ตได้ โดยถ้าไม่ใช้ option นี้ พอร์ตปลายทางก็จะไม่ถูกเปลี่ยนแปลง สามารถใช้ได้กับโพรโตคอล TCP หรือ UDP เท่านั้น

Example: *iptables –t nat PREROUTING –p tcp –d ! 192.168.1.0/24 *
--dport www –j REDIRECT --to-ports 3128

3.7 แพ็กเก็ตเดินทางผ่าน chain ต่างๆ อย่างไร

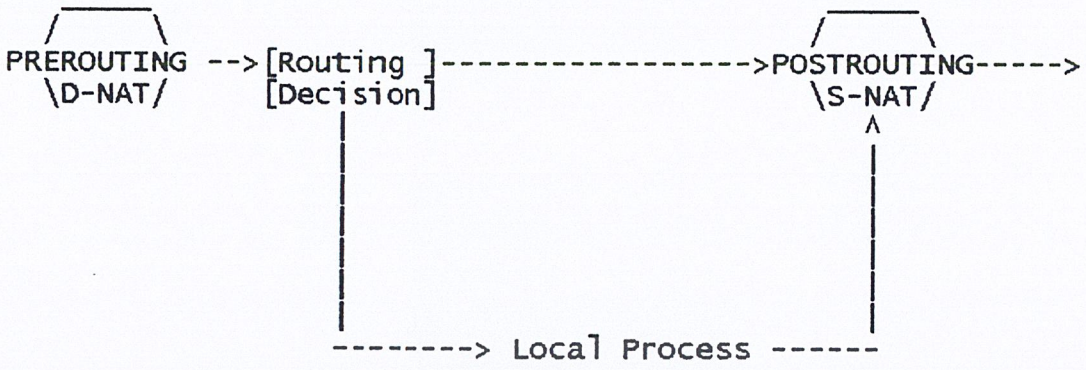
3.7.1 FILTER TABLE



รูปที่ 3-1 Diagram แสดงการเดินทางผ่าน chain ต่างๆ ใน filter table

1. เมื่อแพ็กเก็ตเข้ามาในเครื่องไฟร์วอลล์ (เข้ามาทางเน็ตเวิร์กอินเทอร์เฟส) ต้องผ่าน Routing Decision ซึ่งเป็นขั้นตอนการตัดสินของตัวเคอร์เนลว่า แพ็กเก็ตควรเดินทางต่อไปที่ใด โดยเคอร์เนลจะมองดูจากแอดเดรสปลายทางของแพ็กเก็ต
2. ถ้าแอดเดรสปลายทาง คือ แอดเดรสของเครื่องไฟร์วอลล์นี้เอง (ถ้าไม่ใช่ดู 3. ต่อไป) แพ็กเก็ตจะเดินทางไปตรวจสอบที่ INPUT chain โดยเมื่อตรวจสอบแล้ว สามารถยอมให้แพ็กเก็ตผ่านได้ โพรเซสที่กำลังรอแพ็กเก็ตนี้อยู่ก็จะได้รับแพ็กเก็ตเพื่อไปประมวลผลต่อไป (ข้ามไปดูข้อ 4. ต่อ) แต่ถ้าไม่สามารถยอมให้แพ็กเก็ตผ่านได้ แพ็กเก็ตก็จะถูกดร็อปทิ้ง
3. แต่ถ้าแอดเดรสปลายทาง ไม่ใช่แอดเดรสของเครื่องไฟร์วอลล์ แพ็กเก็ตนั้นจะถูกดร็อป ถ้าเครื่องไฟร์วอลล์ไม่มีความสามารถในการทำ forwarding แต่ถ้ามีแพ็กเก็ตก็จะเดินทางไปตรวจสอบที่ FORWARD chain โดยถ้ายอมให้แพ็กเก็ตผ่านได้ แพ็กเก็ตก็จะถูกส่งออกไปทางเน็ตเวิร์กอินเทอร์เฟส (คนละอินเทอร์เฟสกับตอนเข้ามา) แต่ถ้ายอมให้ผ่านไม่ได้ ก็จะถูกดร็อปทิ้ง
4. และเมื่อโปรแกรมในเครื่องไฟร์วอลล์ต้องการส่งแพ็กเก็ตออกไปภายนอก แพ็กเก็ตนั้นๆ ก็จะเดินทางไปตรวจสอบที่ OUTPUT chain โดยถ้ายอมให้ผ่านได้ ก็จะถูกส่งออกไป แต่ถ้ายอมให้ผ่านไม่ได้ ก็จะถูกดร็อปทิ้ง

3.7.2 NAT TABLE



รูปที่ 3-2 Diagram แสดงการเดินทางผ่าน chain ต่างๆ ใน nat table

1. เมื่อแพ็กเก็ตเข้ามาในเครื่องไฟร์วอลล์ (เข้ามาทางเน็ตเวิร์กอินเทอร์เฟส) แพ็กเก็ตต้องเดินทางไปตรวจสอบที่ PREROUTING chain ก่อนเพื่อทำการ NAT แอดเดรส
2. จากนั้นแพ็กเก็ตจะต้องผ่าน Routing Decision เพื่อให้เคอร์เนลตัดสินใจว่าจะให้แพ็กเก็ตเดินทางต่อไปที่ใด
3. โดยถ้าแพ็กเก็ตนี้เป็นการส่งต่อ แพ็กเก็ตจะเดินทางต่อไปที่ POSTROUTING chain เพื่อ NAT แอดเดรสเพื่อทำการส่งต่อออกไปอีกที แต่ถ้าแพ็กเก็ตเป็นที่ต้องการของโพรเซสภายใน แพ็กเก็ตก็จะถูกส่งไปให้โพรเซสที่รออยู่เพื่อนำไปประมวลต่อ
4. และเมื่อต้องมีการส่งแพ็กเก็ตออกไปภายนอก แพ็กเก็ตนั้นๆ ก็จะต้องเดินทางไปที่ POSTROUTING chain เพื่อทำการ NAT แอดเดรสก่อน

บทที่ 4

Network Address Translation (NAT)

4.1 NAT

การเปลี่ยนแปลงแอดเดรสต้นทางหรือแอดเดรสปลายทางที่ผ่านมา เช่น การแปลงแอดเดรสต้นทางที่เป็นไอพีแอดเดรสภายใน ให้กลายเป็นไอพีแอดเดรสจริงก่อนที่แพ็กเก็ตจะออกสู่อินเตอร์เน็ต หรือในทางกลับกัน คือ แปลงแอดเดรสปลายทางจากไอพีแอดเดรสจริง ให้กลายเป็นไอพีแอดเดรสภายในก่อนที่จะเข้าสู่เน็ตเวิร์กภายใน

- **ข้อดีและข้อเสียของ NAT**

ข้อดี

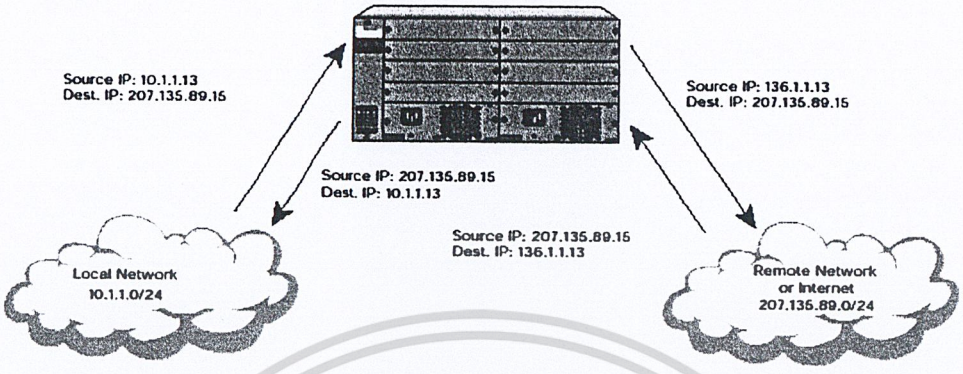
1. สามารถนำเน็ตเวิร์กภายในที่ใช้ไอพีแอดเดรสภายในเชื่อมต่อเข้ากับเน็ตเวิร์กภายนอกหรืออินเตอร์เน็ตได้ ซึ่งทำให้ประหยัดไอพีแอดเดรส
2. สามารถทำการปิดบังหรือย้ายเครื่องที่ให้บริการสาธารณะได้ ซึ่งมีผลดีทางด้านการรักษาความปลอดภัย
3. สามารถใช้ไอพีแอดเดรสจริงเพียง 1 ไอพี แทนเครื่องที่ให้บริการสาธารณะได้หลายๆ เครื่อง โดยใช้ Port-Forwarding
4. สามารถช่วยในการทำ Transparent Proxy ได้

ข้อเสีย

1. NAT ต้องใช้กำลังของโพรเซสเซอร์มาก ซึ่งจะกับเน็ตเวิร์คขนาดใหญ่ ที่ต้องมีการใช้ NAT มากๆ
2. เพิ่มความเป็นไปได้ของ miss-addressing โดยเฉพาะกับแอพพลิเคชั่นโพรโตคอลที่มีความซับซ้อนมาก เช่น FTP

4.2 STATIC NAT

การแมพระหว่างแอดเดรสของเน็ตเวิร์กภายในกับแอดเดรสของเน็ตเวิร์กภายนอกแบบคงที่ (กำหนดตายตัว)

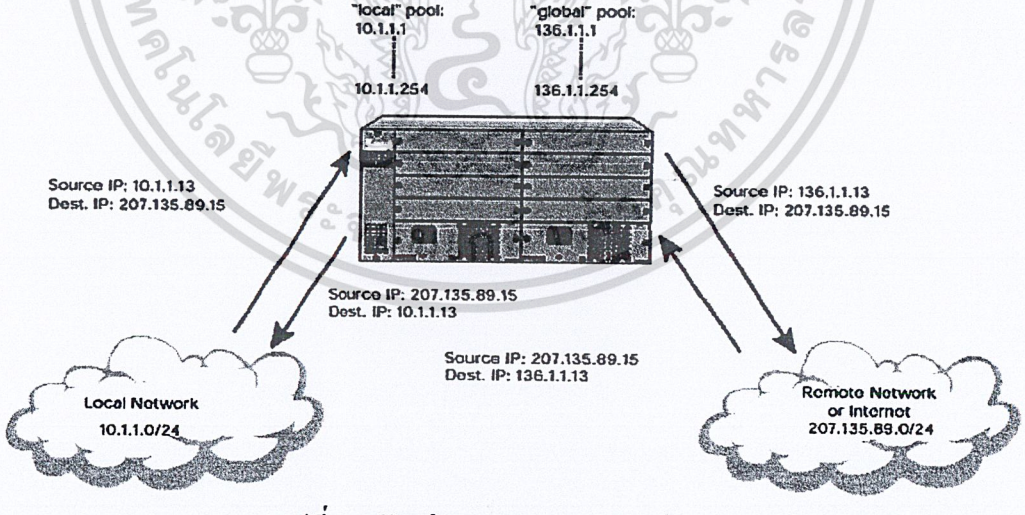


รูปที่ 4-1 ตัวอย่างของ Static NAT

(แอดเดรสต้นทางถูกแปลงจาก 10.1.1.13 เป็น 136.1.1.13 และแอดเดรสปลายทางถูกแปลงจาก 136.1.1.13 เป็น 10.1.1.13)

4.3 DYNAMIC NAT

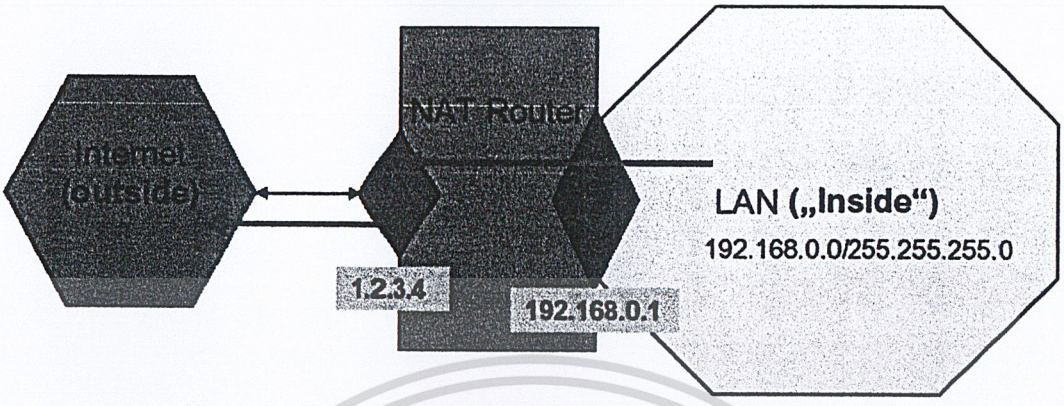
การแมพระหว่างแอดเดรสของเน็ตเวิร์กภายในกับแอดเดรสของเน็ตเวิร์กภายนอกแบบไม่คงที่ คือ จะระบุแอดเดรสที่จะใช้ไว้เป็นช่วงๆ หรือเป็นกลุ่มๆ ซึ่งผู้ใช้จะไม่ทราบว่าแอดเดรสใดที่ถูกเลือกใช้



รูปที่ 4-2 ตัวอย่างของ Dynamic NAT

4.4 SOURCE NAT (SNAT)

การเปลี่ยนแปลงแอดเดรสต้นทางของแพ็กเก็ต ใช้ได้กับทั้ง Static NAT และ Dynamic NAT



รูปที่ 4-3 การแปลงแอดเดรสต้นทางของแพ็กเก็ตที่ส่งจากเน็ตเวิร์กภายในออกไปเน็ตเวิร์กภายนอก จาก 192.168.0.1 เป็น 1.2.3.4

- **SNAT สำหรับ IPtables**

การทำ SNAT สำหรับ IPtables นั้น สามารถใช้ได้กับ POSTROUTING chain เท่านั้น

ตัวอย่างการเปลี่ยนแอดเดรสต้นทางเป็น 1.2.3.4 (Static NAT)

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

ตัวอย่างการเปลี่ยนแอดเดรสต้นทางเป็น 1.2.3.4, 1.2.3.5 หรือ 1.2.3.6 (Dynamic NAT)

iptables –t nat –A POSTROUTING –o eth0 –j SNAT --to 1.2.3.4-1.2.3.6

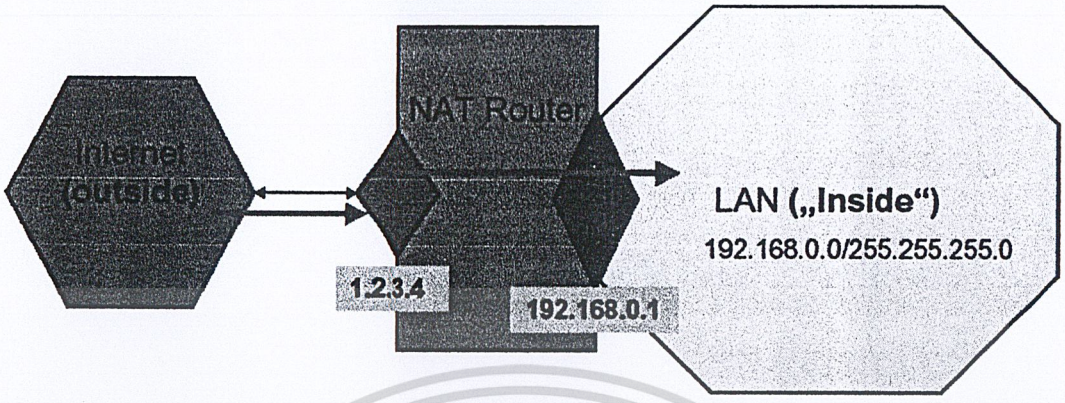
ตัวอย่างการเปลี่ยนแอดเดรสต้นทางเป็น 1.2.3.4 และเปลี่ยนพอร์ตต้นทางเป็นพอร์ตใดพอร์ตหนึ่งใน 1-1023

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

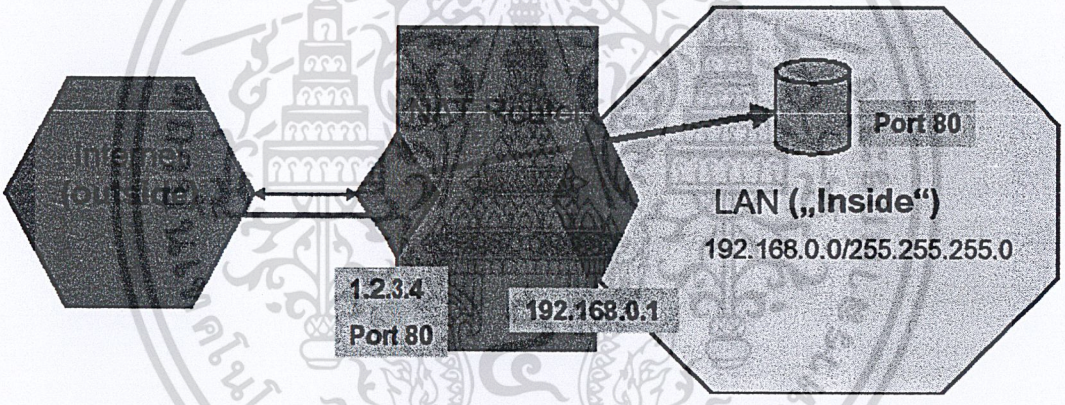
หมายเหตุ: สามารถดูการใช้ option ต่างๆ ได้ในเรื่อง IPtables

4.5 DESTINATION NAT (DNAT)

การเปลี่ยนแปลงแอดเดรสปลายทางของแพ็กเก็ต ใช้ได้กับทั้ง Static NAT และ Dynamic NAT



รูปที่ 4-4 การแปลงแอดเดรสปลายทางของแพ็กเก็ตที่ส่งจากเน็ตเวิร์กภายนอกเข้ามาในเน็ตเวิร์กภายใน จาก 1.2.3.4 เป็น 192.168.0.1



รูปที่ 4-5 การแปลงแอดเดรสปลายทางของแพ็กเก็ตโดยใช้ Port-Forwarding เพื่อให้เน็ตเวิร์กภายนอกสามารถใช้บริการจากเซิร์ฟเวอร์ที่ใช้แอดเดรสภายในได้

- **DNAT สำหรับ IPtables**

การทำ DNAT สำหรับ IPtables นั้น สามารถใช้ได้กับ PREROUTING, OUTPUT chain หรือ User-Defined chain ที่ถูกเรียกโดย chain เหล่านั้นเท่านั้น

ตัวอย่างการเปลี่ยนแอดเดรสปลายทางเป็น 5.6.7.8 (Static NAT)

iptables –t nat –A PREROUTING –i eth1 –j DNAT --to 5.6.7.8

ตัวอย่างการเปลี่ยนแอดเดรสปลายทางเป็น 5.6.7.8, 5.6.7.9 หรือ 5.6.7.10 (Dynamic NAT)

iptables –t nat –A PREROUTING –i eth1 –j DNAT --to 5.6.7.8-5.6.7.10

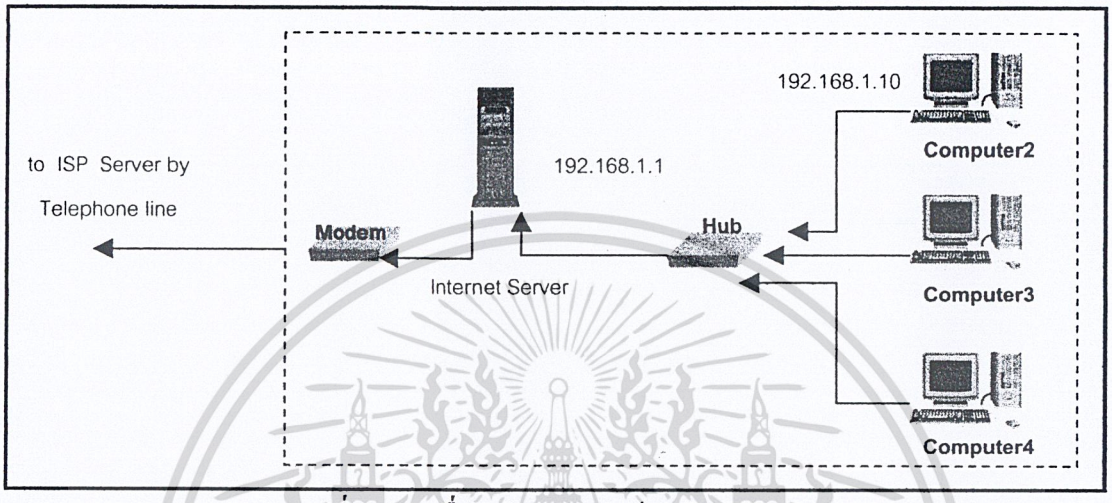
ตัวอย่างการเปลี่ยนแอดเดรสปลายทางเป็น 5.6.7.8 และเปลี่ยนพอร์ตต้นทางเป็นพอร์ต 8080

iptables –t nat –A PREROUTING –p tcp --dport 80 –i eth1 –j DNAT --to 5.6.7.8:8080

หมายเหตุ: สามารถดูการใช้ option ต่างๆ ได้ในเรื่อง IPtables

4.6 MASQUERADING

เป็นกรณีพิเศษของ SNAT ใช้สำหรับ Dynamic NAT (หากต้องการ Static NAT ควรใช้ SNAT) เช่น การเชื่อมต่อแบบ Dial-Up (การเชื่อมต่อโดยใช้โมเด็ม)



รูปที่ 4-6 การเชื่อมต่อกับอินเตอร์เน็ตโดยใช้โมเด็ม

- **MASQUERADING สำหรับ IPtables**

การทำ MASQUERADING สำหรับ IPtables นั้น สามารถใช้ได้กับ POSTROUTING chain เท่านั้น

ตัวอย่างการทำ MASQUERADING ทุกๆ แพ็กเก็ตที่ผ่านออกทาง ppp0 (Modem Interface)

iptables –t nat –A POSTROUTING –o ppp0 –j MASQUERADE

หมายเหตุ: สามารถดูการใช้ option ต่างๆ ได้ในเรื่อง IPtables

4.7 REDIRECTION

เป็นกรณีพิเศษของ REDIRECTION ใช้สำหรับแปลงแอดเดรสปลายทางให้เป็น 127.0.0.1 เพื่อให้ส่งแพ็กเก็ตนั้นให้กับตัวเอง

- **REDIRECTION สำหรับ IPtables**

การทำ REDIRECTION สำหรับ Iptables นั้น สามารถใช้ได้กับ PREROUTING, OUTPUT chain หรือ User-Defined chain ที่ถูกเรียกโดย chain เหล่านั้นเท่านั้น

ตัวอย่างการทำ REDIRECTION จากการใช้บริการเว็บไปใช้บริการ Proxy แทน

iptables –t nat –A PREROUTING –I eth1 –p tcp --dport 80 –j REDIRECT --to-ports 3128

หมายเหตุ: สามารถดูการใช้ option ต่างๆ ได้ในเรื่อง IPtables

บทที่ 5
ความปลอดภัยของระบบเครือข่าย

ปัจจุบันมีความเสี่ยงที่จะเกิดอันตรายต่อเน็ตเวิร์กคอมพิวเตอร์เพิ่มสูงขึ้นเป็นอย่างมาก ดังนั้นการรักษาความปลอดภัยให้กับเน็ตเวิร์กของเรา จึงเป็นเรื่องสำคัญอย่างมากที่จะต้องคำนึงถึง และสิ่งที่ควรจะทราบไว้ ก็คือ อันตรายต่างๆ ที่เกิดขึ้นกับเน็ตเวิร์กคอมพิวเตอร์ส่วนใหญ่นั้น มักเกิดจากผู้บุกรุก ซึ่งรับรู้ถึงข้อบกพร่องต่างๆ ของเน็ตเวิร์กเป้าหมาย เช่น ข้อบกพร่องของโพรโตคอลที่ใช้ เป็นต้น ดังนั้นการจะรักษาความปลอดภัยระบบเน็ตเวิร์กของเราได้นั้น จำเป็นต้องรู้เท่าทันตัวเอง การที่ระบบถูกเจาะได้ มิได้หมายถึงว่า ผู้ที่บุกรุกเป็นผู้ที่เก่งกาจแต่อย่างใด แต่หมายความถึงว่า ผู้ดูแลเน็ตเวิร์กมิได้รู้เท่าทันตัวเองซะมากกว่า

5.1 ภัยคุกคาม (Threat)

โดยทั่วไปภัยคุกคามจะแบ่งออกตามลักษณะของการเข้าถึงและใช้งานเน็ตเวิร์ก ซึ่งสามารถแบ่งออกได้เป็น 4 ระดับ ดังนี้

- **ระดับกายภาพ (Physical Level)**

ภัยคุกคามในระดับนี้จะอาศัยการเข้าถึงระบบคอมพิวเตอร์และเน็ตเวิร์กในระดับกายภาพ (อุปกรณ์ต่างๆ ที่สามารถมองเห็นหรือสัมผัสได้จริงๆ) แล้วจึงทำลายให้เสียหาย เช่น การตัดสายเคเบิล การขโมยเครื่องเซิร์ฟเวอร์ และอาจรวมถึงภัยจากธรรมชาติด้วย เช่น น้ำท่วม (ทำให้ไฟลัดวงจรได้)

- **ระดับเน็ตเวิร์ก (Network Level)**

ภัยคุกคามในระดับนี้จะอาศัยช่องทางการสื่อสารที่มีอยู่เพื่อเข้าถึง แล้วจึงสร้างความเสียหาย หรือในเน็ตเวิร์กไปเพื่อวัตถุประสงค์ไม่ดี โดยที่การคุกคามจะใช้เทคนิคในระดับเน็ตเวิร์กอาศัยข้อบกพร่องของโพรโตคอลและวิธีการสื่อสารเป็นหลัก ภัยคุกคามในระดับนี้ ไม่จำเป็นต้องเข้าถึงอุปกรณ์ต่างๆ ในระดับกายภาพจริงๆ แค่เข้าถึงแบบลอจิคอลก็เพียงพอแล้ว และไม่ผูกติดอยู่กับแอพพลิเคชั่นใดๆ โดยเฉพาะ จะผูกติดอยู่กับโพรโตคอลและวิธีการสื่อสารเท่านั้น

- **ระดับแอพพลิเคชั่น (Application Level)**

เป็นภัยคุกคามที่เกิดขึ้นกับแอพพลิเคชั่นหรือระบบปฏิบัติการโดยตรง โดยอาศัยข้อบกพร่องของตัวแอพพลิเคชั่นมาใช้เป็นช่องทางในการบุกรุก ก่อกวน หรือทำลาย เช่น แอพพลิเคชั่นที่มีการตรวจสอบการรับอินพุทที่ไม่ดี อาจทำให้ผู้ใช้สามารถป้อนอินพุทที่ผิดรูปแบบได้ ส่งผลให้ตัวแอพพลิเคชั่นทำงานผิดพลาดหรือหยุดทำงานได้ เป็นต้น การที่มีแอพพลิเคชั่นที่มีความบกพร่องอยู่มาก จะยิ่งทำให้มีช่องโหว่เกิดขึ้นอยู่ในระบบเน็ตเวิร์กของเรามากด้วยเช่นกัน

- **ระดับผู้ใช้ (User Level)**

ผู้ใช้ธรรมดาทั่วๆ ไปนั้น มักคิดว่าตัวเองไม่น่าจะก่อให้เกิดภัยคุกคามต่อระบบเน็ตเวิร์กได้ แต่จริงๆ แล้วผู้ใช้หลายประเภทที่ก่อให้เกิดภัยคุกคามขึ้นได้ เช่น ผู้ใช้ไม่รักษาส่วนที่เป็นความลับของตัวเอง เปิดเผยรหัสผ่านให้ผู้อื่นรับทราบ หรือทำให้ผู้อื่นรับทราบรหัสผ่านของตัวเอง โดยไม่ได้ตั้งใจ ผู้ใช้มีความรู้ไม่เพียงพอ ประมาทเลินเล่อ

5.2 ภัยคุกคามระดับเน็ตเวิร์กและการป้องกัน

ภัยคุกคามระดับเน็ตเวิร์กสามารถแบ่งได้ตามผลที่ได้รับออกเป็น 3 กลุ่มใหญ่ๆ ดังนี้

- Confidentiality & Privacy Violation การล่วงละเมิดข้อมูลอันเป็นความลับของผู้อื่น
- Denail of Services (DoS) โจมตีส่วนใดส่วนหนึ่งหรือทั้งหมดของระบบ เพื่อให้ไม่สามารถทำงานได้อย่างสมบูณ์
- Unauthorized Access การลักลอบเข้ามาใช้งานเน็ตเวิร์ก หรือสอบถามข้อมูลจากเน็ตเวิร์กโดยไม่ได้รับอนุญาต

ทางด้านการป้องกันนั้น ภัยคุกคามบางชนิดสามารถใช้สูตรแบบตายตัวได้เลย แต่สำหรับภัยคุกคามบางชนิดนั้น ไม่สามารถจะใช้สูตรแบบตายตัวได้ ต้องดูข้อมูลแวดล้อมอื่นๆ ด้วย เพราะการป้องกันอาจส่งผลกระทบต่อการให้บริการบางอย่างของเน็ตเวิร์กด้วย จึงจำเป็นต้องดูข้อมูลแวดล้อมอื่นๆ ด้วย ว่าต้องให้ความสำคัญกับเรื่องใดมากกว่า

5.2.1 Confidentiality & Privacy Violation

Packet Sniffer

รูปแบบ:

ดักอ่านข้อมูลที่สื่อสารอยู่บนเน็ตเวิร์กเพื่อให้ได้มาซึ่งข้อมูลของผู้อื่น จะมีลักษณะใกล้เคียงกับการดักฟังโทรศัพท์

การป้องกัน:

- **การป้องกันโดยลดการกระจายของข้อมูล** ซึ่งสามารถทำได้หลายวิธี คือ
    1. **ไม่เชื่อมสายที่ไม่มีผู้ใช้งานเข้าไปยังฮับ** เพื่อป้องกันการเชื่อมต่อจากบุคคลภายนอก แต่ไม่สามารถป้องกันบุคคลภายในที่เชื่อมต่ออยู่แล้วได้
    2. **เปลี่ยนจากโทโพโลยี่แบบบัสเป็นโทโพโลยี่แบบสตาร์** ในโทโพโลยี่แบบบัส โฮสต์ทุกตัวจะต้องใช้บัสร่วมกัน ซึ่งบัสเป็นแหล่งกระจายข้อมูลให้กับโฮสต์ทุกๆ ตัวอย่างทั่วถึงโดยไม่สามารถป้องกันได้เลย

3. **เปลี่ยนจากฮับธรรมดาเป็นสวิทช์** ฮับธรรมดาจะกระจายข้อมูลไปยังทุกโฮสต์ที่ต่ออยู่บนตัวมัน แต่สวิทช์จะกระจายข้อมูลให้กับเฉพาะโฮสต์ที่เป็นคู่สนทนาเท่านั้น
4. **เปลี่ยนแบ็คโบนจากสายโคแอคเชียลเป็นสายใยแก้วนำแสง** ถึงแม้ว่าจะใช้โทโพโลยี่แบบสตาร์ในแต่ละเน็ตเวิร์กแล้วก็ตาม อย่างไรก็ต้องมีบัสที่เรียกว่าแบ็คโบนเพื่อเชื่อมเน็ตเวิร์กเข้าด้วยกัน ซึ่งหากใช้สายโคแอคเชียลจะสามารถเป็นช่องทางที่ให้มีการนำสนิฟเฟอร์มาเกาะกับแบ็คโบนได้โดยไม่รู้ตัว แต่ถ้าเป็นใยแก้วนำแสงแล้ว โอกาสที่จะนำสนิฟเฟอร์มาเกาะนั้นเป็นไปไม่ได้เลย

ข้อดี:

เป็นการป้องกันที่ต้นเหตุ สามารถลดความเสี่ยงได้มาก

ข้อเสีย:

อาจจำเป็นต้องมีการปรับปรุงสภาพเน็ตเวิร์ก ซึ่งถ้าต้องปรับปรุงมากก็จะทำให้เสียเวลาและค่าใช้จ่ายมากด้วย

- **การป้องกันโดยการเข้ารหัสข้อมูลในระดับแอพพลิเคชั่น**

ไม่ได้เป็นการป้องกันการดักอ่านข้อมูลโดยสนิฟเฟอร์โดยตรง แต่เป็นการป้องกันเนื้อหาของข้อมูลมากกว่า คือ ดักเอาไปได้ แต่ก็อ่านไม่รู้เรื่อง โดยวิธีการเข้ารหัสนี้จะกระทำในระดับแอพพลิเคชั่น จะแนกได้เป็น 2 วิธี คือ

1. **เข้ารหัสเฉพาะข้อมูลแต่ใช้แอพพลิเคชั่นเดิม** ใช้แอพพลิเคชั่นเดิม ใช้วิธีการสื่อสารและส่งข้อมูลเหมือนเดิมทุกอย่าง เพียงแต่นำข้อมูลมาเข้ารหัสด้วยแอพพลิเคชั่นที่ใช้สำหรับเข้ารหัสโดยเฉพาะก่อน จึงค่อยส่งข้อมูล
2. **เปลี่ยนแอพพลิเคชั่น** จากเดิมที่ใช้แอพพลิเคชั่นที่สื่อสารข้อมูลด้วยวิธีธรรมดา ก็เปลี่ยนมาใช้แอพพลิเคชั่นใหม่ที่มีการสื่อสารข้อมูลที่เข้ารหัส เช่น การเปลี่ยนจากการใช้โปรแกรมเทลเน็ตมาเป็น SSH (Secure Shell) แทน เป็นต้น

ข้อดี:

- ช่วยลดความเสี่ยงจากการถูกอ่านข้อมูลที่เป็นความลับได้ระดับหนึ่ง
- กระจายการป้องกันข้อมูลให้ครอบคลุมหลายระดับ ถึงแม้ว่าการป้องกันในระดับอื่นจะผิดพลาด ก็ยังมีการป้องกันในระดับแอพพลิเคชั่นอยู่

ข้อเสีย:

- ต้องใช้แอพพลิเคชั่นที่ใช้เข้ารหัสโดยเฉพาะเพิ่มเติม

- ต้องเก็บรักษากุญแจถอดรหัสอย่างรัดกุม
- ต้องเสียเวลามากขึ้นในการสื่อสาร เพราะต้องใช้เวลาในการเข้ารหัสและถอดรหัสด้วย
- แอพพลิเคชั่นอาจไม่รองรับการเข้ารหัส ซึ่งอาจทำให้ไม่สามารถเข้ารหัสได้ หรือทำได้แต่ยุ่งยากมาก

- **การป้องกันโดยการเข้ารหัสข้อมูลในระดับเน็ตเวิร์ก**

การเข้ารหัสข้อมูลในระดับเน็ตเวิร์กนี้ เป็นที่รู้จักกันทั่วไปในนาม *VPN (Virtual Private Network)* จะต่างจากการเข้ารหัสข้อมูลในระดับแอพพลิเคชั่นตรงที่การเข้ารหัสข้อมูลในระดับเน็ตเวิร์กจะไม่ส่งผลกระทบใดๆ ต่อแอพพลิเคชั่น แอพพลิเคชั่นเดิมสามารถทำงานได้ตามปกติ โดยที่ไม่ต้องแก้ไขหรือเปลี่ยนแปลงอะไรเลย เพราะข้อมูลที่ถูกส่งให้กับแอพพลิเคชั่นนั้น จะถูกถอดรหัสมาเรียบร้อยแล้วตั้งแต่ในระดับเน็ตเวิร์ก และในทางกลับกันการส่งข้อมูลออกของแอพพลิเคชั่น ก็สามารถทำได้ตามปกติ โดยข้อมูลจะไปถูกเข้ารหัสในระดับเน็ตเวิร์กเอง

การเข้ารหัสข้อมูลในระดับเน็ตเวิร์กนั้น จะอาศัย TCP/IP ในการรับส่งข้อมูล แต่เพิ่มการเข้ารหัสและสร้างช่องทางการสื่อสารที่ปลอดภัยขึ้นอีกชั้นหนึ่ง หรือเรียกอีกอย่างหนึ่งว่า *ทันเนลลิ่ง (Tunnelling)* ซึ่งโพรโตคอลที่ใช้เพื่อการทันเนลลิ่งก็มีอยู่หลายโพรโตคอล เช่น

IPSec (Internet Protocol Security, RFC 2401-11, RFC 2501)
PPTP (Point to Point Tunnelling Protocol, RFC 2637)
L2F (Layer 2 Forwarding, RFC 2341)
L2TP (Layer 2 Tunnelling Protocol, RFC 2661)

ข้อดี:

- มีความปลอดภัยของข้อมูลสูงมาก
- ไม่ขึ้นอยู่กับแอพพลิเคชั่น
- สามารถป้องกันโฮสต์ทุกโฮสต์ที่ใช้ช่องทางนี้ในการสื่อสารได้โดยอัตโนมัติ

ข้อเสีย:

- ต้องใช้แบนวิดธ์มากกว่าการส่งข้อมูลแบบธรรมดา
- ต้องใช้กำลังของโฮสต์และอุปกรณ์ที่อยู่ในเน็ตเวิร์กในการเข้ารหัสและถอดรหัสสูงมาก

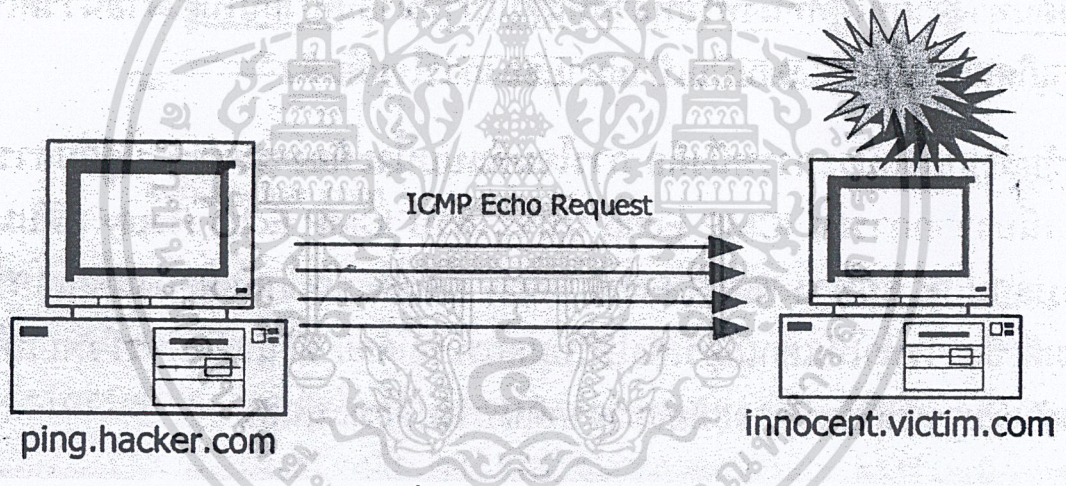
- ประสิทธิภาพในการสื่อสารข้อมูลต่ำ

5.2.2 Denial of Services (DoS)

5.2.2.1 Ping Flood Attack

รูปแบบ:

ส่ง ICMP Echo Request (แบบเดียวที่ได้จากคำสั่ง Ping) ปริมาณมากๆ ไปยังเป้าหมายอย่างรวดเร็ว ทำให้โฮสต์ที่ถูกโจมตีจะต้องคอยตอบ ICMP Echo Reply ตลอดเวลาจนไม่เวลาจะทำงานอื่น ความรุนแรงของการโจมตีจะมากหรือน้อย ขึ้นอยู่กับความเร็วในการส่ง ICMP แพ็กเก็ตเป็นหลัก หากผู้โจมตีมีแบนวิดธ์สูง และมีเครื่องที่มีสมรรถนะสูง ก็จะสามารถโจมตีได้ด้วยแพ็กเก็ตจำนวนมากในเวลาอันสั้น เครื่องเป้าหมายก็อาจจะหยุดทำงานลงได้ (การโจมตีแบบนี้จะเสมือนกับผู้ที่มีทรัพยากรสูงกว่าทำลายเป้าหมายที่มีทรัพยากรต่ำกว่า)



รูปที่ 5-1 Ping Flood Attack

การป้องกัน:

กำหนดอัตราการรับ ICMP Echo Request ที่เข้ามาในเน็ตเวิร์กให้อยู่ในระดับที่สามารถควบคุมได้ และไม่มากเกินไป โดยสามารถกำหนดได้ที่เราเตอร์หรือไฟร์วอลล์ก็ได้

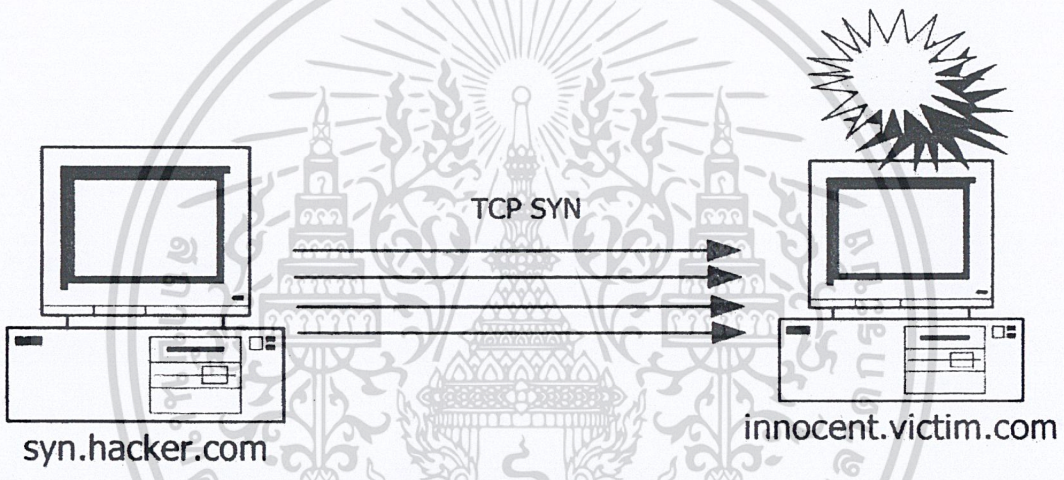
ตัวอย่างการกำหนดกฎใน IPtables

iptables –A INPUT –p icmp --icmp-type echo-request –m limit –j ACCEPT

5.2.2.2 SYN Flood Attack

รูปแบบ:

ส่งสัญญาณ TCP SYN ปริมาณมากๆ ไปยังเครื่องเป้าหมาย เสมือนการทำ 3-way handshake แต่พอเครื่องเป้าหมายส่งสัญญาณ TCP SYN/ACK กลับมา จะไม่มีการส่งสัญญาณ TCP ACK กลับไปเพื่อจบขั้นตอนการร้องขอเชื่อมต่อ ดังนั้นจะทำให้เครื่องเป้าหมายต้องรอสัญญาณจนกว่าจะถึงเวลาที่กำหนดจึงจะยุติการรอได้ ซึ่งถ้ามีปริมาณของ TCP SYN จำนวนมากๆ ในระยะเวลาอันสั้น ก็จะทำให้เครื่องเป้าหมายต้องจัดสรรทรัพยากรของเครื่องมาเพื่อรองรับขั้นตอนการเชื่อมต่อ (ปลอมๆ) เป็นจำนวนมาก จนส่งผลให้ไม่สามารถจัดสรรทรัพยากรของเครื่องให้กับกิจกรรมอื่นได้อีก จนอาจหยุดการทำงานลงได้



รูปที่ 5-2 SYN Flood Attack

การป้องกัน:

- **การป้องกันโดยเราเตอร์**

กำหนดอัตราการรับข้อมูล CAR (Committed Access Rate) ของแพ็กเก็จ TCP SYN ที่ ACL (Access Control List) ของเราเตอร์ เช่นเดียวกับการป้องกัน Ping Flood เพื่อให้ระดับร้องขอเชื่อมต่ออยู่ในระดับที่สามารถควบคุมได้ และไม่มากเกินไป

หรืออีกวิธีหนึ่ง ซึ่งเป็นความสามารถของเราเตอร์ที่ใช้ในการป้องกัน SYN Flood โดยเฉพาะ เรียกว่า การอินเตอร์เซ็พท์ (TCP Intercept)

การอินเตอร์เซ็พท์ คือ การที่เราเตอร์ทำกระบวนการ 3-way handshake แทนโฮสต์ปลายทางก่อน จนสิ้นสุดกระบวนการแพ็กเก็ตจึงจะถูกส่งไปยังโฮสต์ปลายทาง

- **การป้องกันโดยไฟร์วอลล์**

ใช้ความสามารถของไฟร์วอลล์ในส่วนที่ใช้ป้องกันการโจมตีแบบนี้โดยเฉพาะ โดยมี 2 สิ่งที่จะต้องพิจารณาเครื่องไฟร์วอลล์ ดังนี้

- ไฟร์วอลล์ต้องมีทรัพยากรเพียงพอที่จะให้บริการ โดยเฉพาะหน่วยความจำที่ใช้สำหรับการจัดการการเชื่อมต่อ และความเร็วในการให้บริการ
- ไฟร์วอลล์ต้องผ่านการทดสอบว่าสามารถรับมือกับการโจมตีในลักษณะนี้ได้จริง และสามารถทำงานได้แม้ว่าจะโดนโจมตีอยู่ก็ตาม

ตัวอย่างการกำหนดกฎใน IPtables

iptables –A INPUT –p tcp --syn –m limit –j ACCEPT

- **การป้องกันโดยโฮสต์**

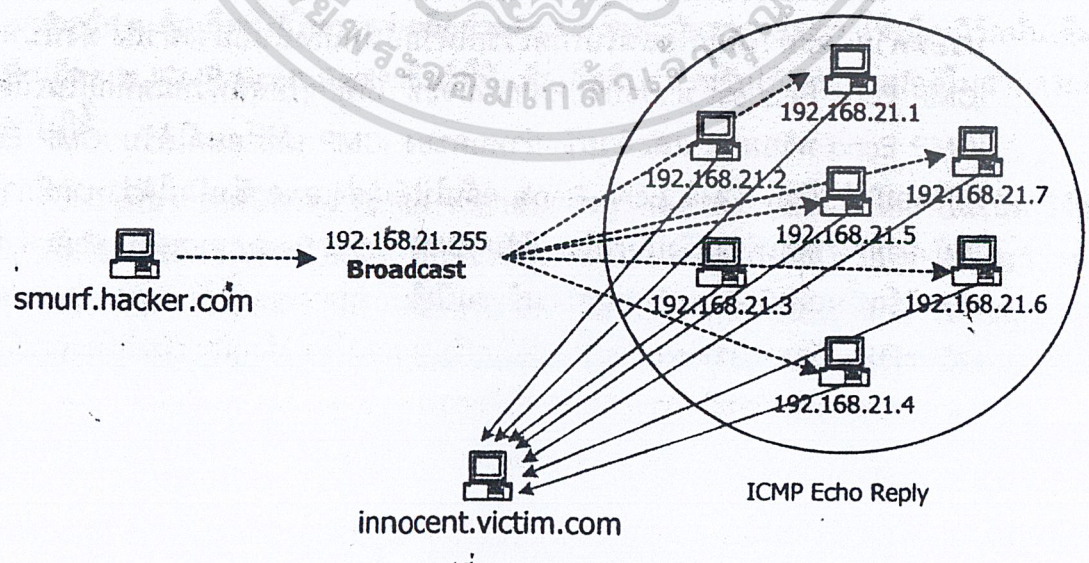
ติดตั้งและปรับปรุงระบบปฏิบัติการให้เป็นเวอร์ชั่นใหม่ที่สุดเท่าที่จะทำได้ เพราะระบบปฏิบัติการเวอร์ชั่นใหม่ๆ จะถูกปรับปรุงในส่วนจัดการ TCP/IP ให้มีประสิทธิภาพสูงขึ้น

เลือกใช้เซิร์ฟเวอร์ที่มีทรัพยากรเหมาะสมกับปริมาณงาน โดยอาจจะมีทรัพยากรสำรองเผื่อไว้สำหรับรองรับการขยายตัวของการให้บริการด้วยยิ่งดี

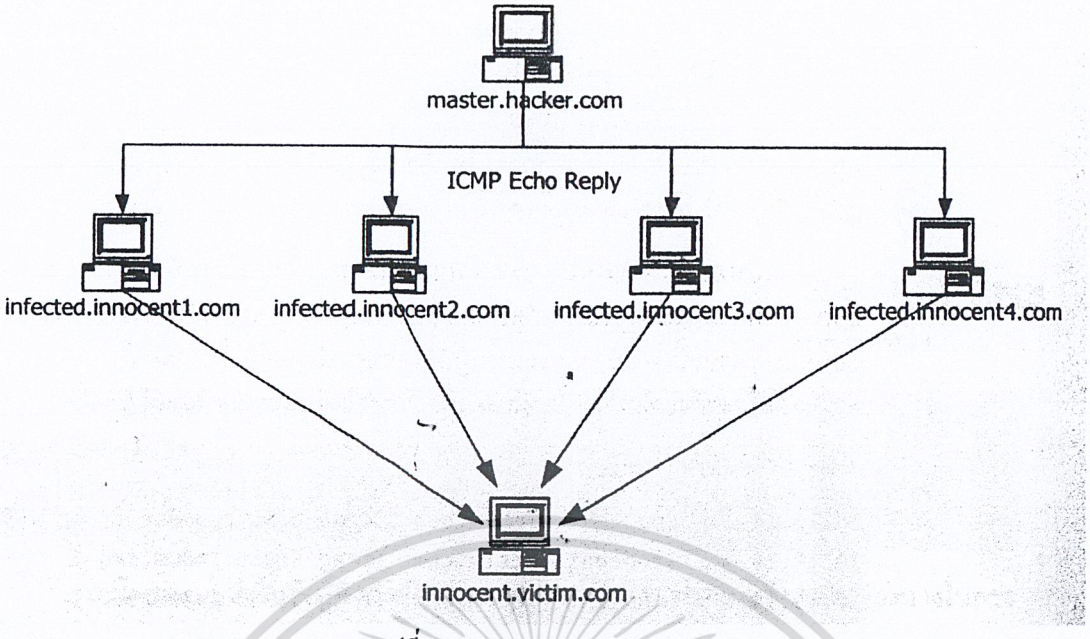
5.2.2.3 Smurf Attack, Tribe Flood Network

รูปแบบ:

ใช้เทคนิคการขยายสัญญาณโดยใช้คุณสมบัติบรอดคาสต์ร่วมกับการใช้ ICMP Echo Request-Reply ดังนั้นการป้องกันจะใช้วิธีการเดียวกับการป้องกัน Ping Flood เพียงแต่เพิ่มเติมในส่วนของการป้องกันการใช้คุณสมบัติบรอดคาสต์ด้วย



รูปที่ 5-3 Smurf Attack



รูปที่ 5-4 Tribe Flood Network

การป้องกัน:

ป้องกันโดยใช้วิธีการเดียวกับการป้องกัน Ping Flood เพียงแต่เพิ่มเติมในส่วนของการป้องกันการใช้คุณสมบัติบรอดคาสต์ด้วย

ตัวอย่างการกำหนดกฎใน IPtables

iptables –A FORWARD –p icmp –d 192.168.39.255 –j DROP

5.2.2.4 Diagnostic Port Attack

รูปแบบ:

เป็นการอาศัยบริการประเภท Small Services ที่เราเตอร์หรือโฮสต์เป้าหมายเปิดไว้ โดย Small Services เป็นบริการที่มีอยู่ใน TCP/IP ตั้งแต่ยุคแรกๆ เพื่อใช้สำหรับตรวจสอบสถานะการเชื่อมต่อ โดยที่มีรายละเอียดการให้บริการดังแสดงในตารางต่อไปนี้

ชื่อบริการ	หมายเลขพอร์ต	รายละเอียดการให้บริการ
Echo	7/UDP 7/TCP	เซิร์ฟเวอร์จะทำการตอบกลับด้วยข้อมูลเดียวกับที่ได้รับมาจากไคลเอนต์
Discard	9/UDP 9/TCP	เซิร์ฟเวอร์จะไม่ตอบรับข้อมูลใดๆ ที่ส่งมาจากไคลเอนต์
Daytime	13/UDP 13/TCP	เซิร์ฟเวอร์จะตอบวันที่และเวลา กลับไปยังไคลเอนต์ในรูปแบบที่สามารถอ่านได้เข้าใจ

Chargen	19/UDP	เซิร์ฟเวอร์จะตอบกลับไปยังไคลเอนต์ด้วย ASCII Character อย่างต่อเนื่องจนกว่าไคลเอนต์จะยุติการติดต่อ
Time	34/UDP 34/TCP	เซิร์ฟเวอร์จะตอบค่าเวลาในรูปแบบของ 32 บิต

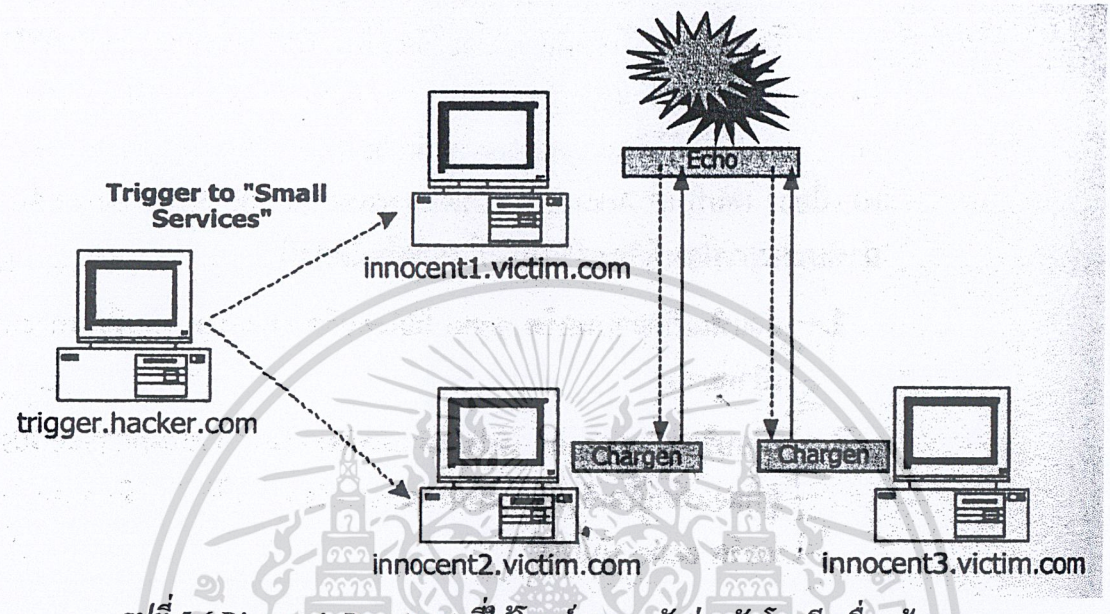
ตารางที่ 5-1 แสดงรายละเอียดการให้บริการของบริการประเภท Small Services

การโจมตีแบบนี้มักจะใช้ควบคู่ไปกับการปลอมหมายเลข IP ของแอดเดรสต้นทางและปลายทางของแพ็กเก็ต โดยวิธีการโจมตีนั้น จะเริ่มจากปลอมแอดเดรสต้นทางและปลายทางของแพ็กเก็ตที่ใช้โจมตีให้เป็นแอดเดรสของเป้าหมาย จากนั้นจะส่งแพ็กเก็ตจากพอร์ตต้นทาง Echo ไปยังพอร์ตปลายทาง Chargen (ใช้พอร์ตต้นทางเป็น Time หรือ Daytime ก็ได้ด้วย แต่ Chargen ให้ประสิทธิภาพในการโจมตีสูงที่สุด) จากนั้นจะเกิดการโจมตีตัวเองของเป้าหมายเกิดขึ้นเองโดยอัตโนมัติ ดังนี้

1. เมื่อบริการ Chargen ได้รับแพ็กเก็ตก็จะเริ่มทำงานด้วยการส่ง ASCII Character กลับไปยังผู้ที่ส่งแพ็กเก็ตเข้ามาตามหมายเลขพอร์ตที่ระบุไว้ ซึ่งก็คือการส่งกลับไปยังพอร์ต Echo ของเครื่องเป้าหมายนั่นเอง
2. เมื่อแพ็กเก็ตจาก Chargen มาถึงพอร์ต Echo บริการ Echo ก็จะทำงานโดยการตอบข้อมูลที่ได้รับกลับไปยังพอร์ต Chargen อีก
3. กลับไปยังข้อที่ 1 อีกครั้ง แล้วจะวนลูปเช่นนี้แบบไม่รู้จบ จนในที่สุดเครื่องเป้าหมายก็จะหยุดทำงานไปเอง

รูปที่ 5-5 Diagnostic Port Attack

นอกจากนี้ถ้าหากพบว่ามีเครื่องอื่นๆ นอกเหนือจากเครื่องเป้าหมาย เปิดบริการเหล่านี้อยู่ด้วยแล้ว จะสามารถเพิ่มประสิทธิภาพการโจมตีให้มากขึ้นอีก ได้ด้วยการให้เครื่องอื่นๆ มาร่วมโจมตีเครื่องเป้าหมายด้วย โดยการให้ Echo ของเครื่องเป้าหมายโต้ตอบกับ Chargen ที่เปิดบริการอยู่ในเครื่องอื่นๆ ด้วย จะทำให้เครื่องหยุดทำงานเร็วขึ้น



รูปที่ 5-6 Diagnostic Port Attack ที่ให้โฮสต์หลายๆ ตัวร่วมกันโจมตีเครื่องเป้าหมาย

และด้วยวิธีการโจมตีเช่นนี้ หากว่าภายในเน็ตเวิร์กนั้นมีโฮสต์ที่เปิดให้บริการ Small Services อยู่หลายๆ เครื่องแล้ว อาจทำให้เน็ตเวิร์กเป็นอัมพาตได้เลย

การป้องกัน:

เนื่องจาก Small Services เหล่านี้ เป็นบริการที่ไม่จำเป็นอีกต่อไปแล้วดังนั้น จึงป้องกันได้ด้วยการปิดบริการเหล่านี้ที่เราเตอร์และโฮสต์ต่างๆ ภายในเน็ตเวิร์ก และปิดพอร์ตของบริการเหล่านี้ไม่ให้ผ่านไฟร์วอลล์มาได้

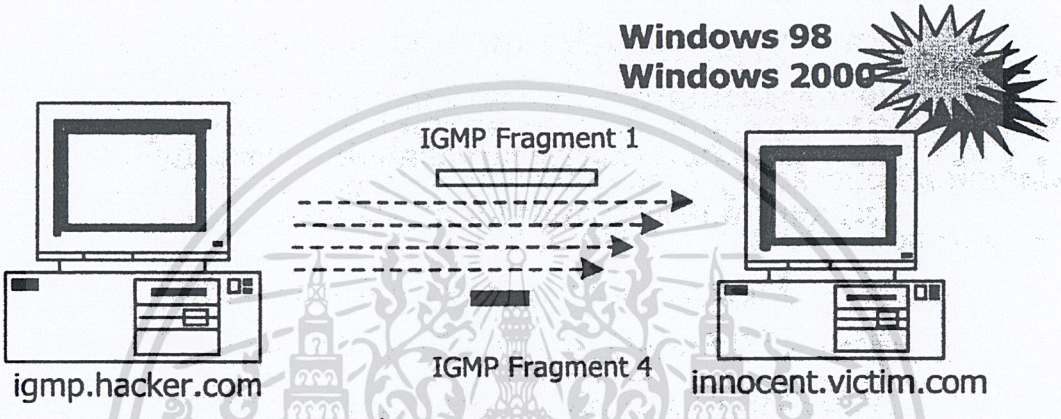
ตัวอย่างการกำหนดกฎใน IPtables

iptables –A FORWARD –p tcp –m multiport --dport 7,9,13,19,34 –j DROP

5.2.2.5 Fragmented IGMP (Internet Group Management Protocol) Attack

รูปแบบ:

เป็นการโจมตีที่ข้อผิดพลาดภายในระบบปฏิบัติการ Microsoft Windows 98 และ Microsoft Windows 2000 โดยการส่งชุดของแฟรกเมนต์แพ็กเก็ตของ IGMP ซึ่งมี 4 แฟรกเมนต์ จำนวน 15 ชุด ไปให้เครื่องเป้าหมายที่ใช้ระบบปฏิบัติการ Windows 98 หรือ Windows 2000 ซึ่งจะส่งผลให้ระบบปฏิบัติการหยุดทำงานและแสดงจอภาพสีฟ้า (Blue Screen) ทำให้ผู้ดูแลระบบนั้นเข้าใจผิดคิดว่าเกิดจากความผิดปกติของฮาร์ดแวร์



รูปที่ 5-7 Fragmented IGMP Attack

การป้องกัน:

- **การป้องกันโดยเราเตอร์หรือไฟร์วอลล์**

จากหลักการที่ว่า แพ็กเก็ต IGMP ไม่ใช่แพ็กเก็ตสำหรับการสื่อสารกับโฮสต์ที่ใช้แอดเดรสยูนิคาสต์ (Unicast) และบรอดคาสต์ (Broadcast) แต่ใช้สำหรับการสื่อสารกับโฮสต์ที่ใช้แอดเดรสมัลติคาสต์ (Multicast) เท่านั้น ดังนั้นแพ็กเก็ตที่เป็น IGMP และมีแอดเดรสปลายทางเป็นโฮสต์ภายในจึงสามารถดร็อปทิ้งไปได้เลย

ตัวอย่างการกำหนดกฎใน IPtables

iptables –A FORWARD –p igmp –d 192.168.39.0/24 –j DROP

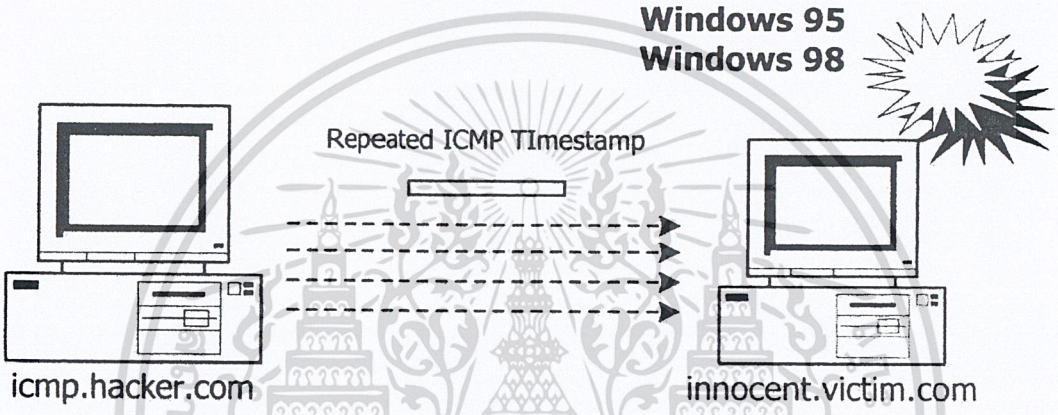
- **การป้องกันโดยโฮสต์**

ปรับปรุงระบบปฏิบัติการด้วยโปรแกรมแก้ไข (Patch) จากผู้ผลิตให้ครบถ้วนและทันสมัยที่สุดเท่าที่ทำได้

5.2.2.6 ICMP Timestamp Attack

รูปแบบ:

เป็นการโจมตีที่ข้อผิดพลาดภายในระบบปฏิบัติการเช่นเดียวกับ Fragmented IGMP Attack แต่เป็นการโจมตีที่ Microsoft Windows 95 และ Microsoft Windows 98 ด้วยการส่ง ICMP Timestamp Request (ซึ่งโดยปกติจะใช้เพื่อการสอบถามเวลาจากโฮสต์ปลายทาง) จำนวนมากและอย่างรวดเร็วไปยังเครื่องเป้าหมายที่ใช้ระบบปฏิบัติการ Windows 98 หรือ Windows 95 ซึ่งจะส่งผลให้เครื่องเป้าหมายหยุดทำงาน และแสดงหน้าจอสีฟ้า (Blue Screen) ได้



รูปที่ 5-8 ICMP Timestamp Attack

การป้องกัน:

- **การป้องกันโดยเราเตอร์หรือไฟร์วอลล์**

ICMP Timestamp Request นี้ไม่จำเป็นต่อการใช้งาน ดังนั้นสามารถกำหนดกฎให้เราเตอร์หรือไฟร์วอลล์ดร็อปทิ้งได้เลย ไม่มีผลกระทบใดๆ ต่อการทำงานตามปกติแต่อย่างใด

ตัวอย่างการกำหนดกฎใน IPtables

iptables –A FORWARD –p icmp --icmp-type timestamp-request –j DROP

5.2.2.7 DoS รูปแบบอื่นๆ

การ DoS ในรูปแบบอื่นๆ เหล่านี้ มักจะทำการโจมตีโดยอาศัยข้อบกพร่องโดยตรงของระบบปฏิบัติการเป็นหลัก ดังนั้นปัจจัยสำคัญที่จะส่งผลกระทบเมื่อถูกโจมตีจึงขึ้นอยู่กับความแข็งแรงของระบบปฏิบัติการ การโจมตีบางวิธีอาจจะใช้ได้ผลรุนแรงกับระบบปฏิบัติการหนึ่ง แต่ไม่ส่งผลระคายเคืองกับอีกระบบปฏิบัติการหนึ่งก็เป็นได้

ตัวอย่าง Dos ในรูปแบบอื่นๆ

- **Land Attack**

ส่งแพ็กเก็ต SYN ไปยังเครื่องเป้าหมาย ตรงพอร์ตที่เปิดอยู่ โดยทำการปลอมแปลงแอดเดรสให้ต้นทางและปลายทางเป็นแอดเดรสเดียวกัน คือ แอดเดรสของเป้าหมาย และพอร์ตต้นทางกับปลายทางก็เป็นพอร์ตเดียวกันด้วย

ส่งผลให้มีการตอบรับไปมาของ TCP วนอยู่ในตัวของเครื่องเป้าหมาย ทำให้คอมพิวเตอร์ใช้ทรัพยากรจนหมด จนต้องหยุดการทำงานลง

การโจมตีแบบนี้ จะได้ผลกับระบบปฏิบัติการรุ่นเก่าๆ ในยุคแรกๆ เท่านั้น เพราะในระบบปฏิบัติการรุ่นใหม่ๆ จะมีการปรับปรุงแก้ไข TCP Stack ให้ดียิ่งขึ้น ทำให้การโจมตีแบบนี้ไม่ได้ผลแล้ว

- **Teardrop Attack**

อาศัยข้อบกพร่องของการแฟรกเมนต์รีแอสเซมเบิล (การรวมแฟรกเมนต์แพ็กเก็ตหลายๆ แพ็กเก็ตกลับมาเป็น IP datagram เดียวกัน) ของระบบปฏิบัติการเป้าหมาย

โจมตีโดยใช้การเหลื่อมกันของแพ็กเก็ตในระหว่างที่มีการแฟรกเมนต์รีแอสเซมเบิล ซึ่งไม่เกิดขึ้นในการใช้งานปกติ ทำให้ระบบปฏิบัติการที่ไม่ได้ตรวจสอบเงื่อนไขตรงนี้ดีพอ อาจทำงานผิดปกติ หรือหยุดทำงานไปเลย

- **Ping of Death Attack**

พยายามส่งแฟรกเมนต์ของ ICMP Echo Request ไปยังเป้าหมายเสมือนการ Ping ธรรมดา แต่ตั้งใจทำให้ผลรวมของแฟรกเมนต์นั้นเกินกว่า 64 K ซึ่งเป็นขนาดที่ระบบปฏิบัติการจัดสรรหน่วยความจำไว้ จึงทำให้ตอนทำการรีแอสเซมเบิล ข้อมูลล้นออกจากหน่วยความจำที่จัดสรรไว้ (overflow) ซึ่งทำให้ระบบปฏิบัติการ แต่โปรแกรมต่างๆ ที่รันอยู่ทำงานผิดพลาด หรือหยุดทำงานในทันที

- **UDP Bomb**

โจมตีโดยอาศัยข้อบกพร่องของ UDP โดยทำให้เครื่องเป้าหมายรับ UDP datagram ที่มีขนาดผิดปกติ และหากระบบปฏิบัติการของเครื่องเป้าหมายไม่สามารถจัดการกับแพ็กเก็ตที่ผิดปกติเหล่านี้ได้ดีพอ เครื่องเป้าหมายก็จะหยุดทำงานลง

- **Winfreeze**

อาศัย ICMP redirect message ในการโจมตี โดยปกติแล้ว ICMP redirect message ใช้ในการแจ้งให้เราเตอร์ เพื่อให้ทำการปรับปรุงเส้นทางในตารางเราต์ติ้ง (routing table)

โจมตีโดยการส่ง ICMP redirect ไปให้กับเครื่องเป้าหมายจำนวนมากอย่างต่อเนื่อง จนทำให้เครื่องเป้าหมายปรับปรุงข้อมูลในตารางเราต์ติ้งไม่ทัน จนเกิดความผิดพลาด

การโจมตีชนิดนี้ มักจะใช้ได้ผลดีกับระบบปฏิบัติการในตระกูล Windows

- **Jolt**

โจมตีโดยอาศัยการแฟรกเมนต์ทั่วๆ ไป โดยทำการส่งแฟรกเมนต์แพ็กเก็ตซ้ำๆ จำนวนมากๆ อย่างต่อเนื่องไปยังเครื่องเป้าหมาย หากมีความเร็วในการส่งและปริมาณไม่มากพอ ก็อาจจะแค่ทำให้เครื่องเป้าหมายทำงานช้าลงเท่านั้น แต่ถ้าเพิ่มความเร็วในการส่งข้อมูลได้ ก็จะทำให้เครื่องเป้าหมายหยุดทำงานในที่สุด

การป้องกัน DoS ในรูปแบบเหล่านี้ ขึ้นอยู่กับความแข็งแรงของระบบปฏิบัติการเป็นสำคัญ จึงควรปรับปรุงให้ระบบปฏิบัติการที่ใช้อยู่เป็นเวอร์ชั่นที่มีความเสถียรที่สุด หรือใหม่ที่สุดเท่าที่เป็นไปได้

5.2.2.8 การป้องกัน DoS ด้วยการตรวจสอบต้นทางของแพ็กเก็ต

การ DoS มักมีคุณสมบัติที่สำคัญร่วมด้วย คือ มักจะมีการปลอมแปลงแอดเดรสต้นทางของแพ็กเก็ตที่ใช้ในการโจมตีเสมอ ด้วยเหตุผลที่สำคัญ ดังต่อไปนี้

1. การโจมตีแบบ DoS บางชนิดจะมีแพ็กเก็ตสะท้อนกลับไปยังผู้ส่งด้วย จึงอาจจะทำให้ตัวเองเดือดร้อนเองได้ หากผู้โจมตีไม่ปลอมแปลงแอดเดรสต้นทาง
2. การโจมตีแบบ DoS ไม่ต้องการการตอบรับใดๆ จากเป้าหมาย จึงไม่มีความจำเป็นต้องใช้แอดเดรสจริงๆ
3. เพื่อเป็นการอำพรางตนเอง ไม่ให้ถูกติดตามได้

ด้วยเหตุผลที่กล่าวมาข้างต้น จึงพอที่จะสามารถอนุมานได้ว่าแพ็กเก็ตที่ใช้ในการโจมตีแบบ DoS นั้น ส่วนใหญ่แล้วจะถูกปลอมแปลงแอดเดรสต้นทางทั้งสิ้น ซึ่งการ

ปลอมแปลงแอดเดรสต้นทางนั้น สามารถที่จะแปลงแอดเดรสต้นทางได้ 2 รูปแบบ คือ Valid IP Address และ Invalid IP Address

การแปลงแอดเดรสต้นทางให้อยู่ในรูปแบบของ Valid IP Address นั้น อาจจะไม่สามารถป้องกันอะไรได้มากนัก เพราะเป็นแอดเดรสที่ถูกใช้อยู่ในอินเตอร์เน็ตอยู่แล้ว แต่การแปลงแอดเดรสต้นทางให้อยู่ในรูปแบบของ Invalid IP Address นั้น สามารถป้องกันได้เลย ด้วยการดร็อปแพ็กเก็ตทั้งหมดที่มีแอดเดรสต้นทางเป็น Invalid IP Address ทิ้งได้เลย เพราะไม่ใช่แอดเดรสที่จะถูกใช้ในอินเตอร์เน็ต ดังนั้นแพ็กเก็ตที่มีแอดเดรสต้นทางเป็น Invalid IP Address จึงแพ็กเก็ตที่มุ่งร้ายแน่นอน

การป้องกันการ DoS ด้วยการตรวจสอบต้นทางนี้ สามารถที่ลดความเสี่ยงจากการโจมตีได้มากทีเดียว เพราะว่าผู้ที่โจมตีแบบ DoS ส่วนใหญ่จะปลอมแอดเดรสต้นทางเป็น Invalid IP Address

ตัวอย่างการกำหนดกฎใน IPtables

iptables –A FORWARD –s 192.168.1.0/24 –j DROP

5.2.3 Unauthorized Access

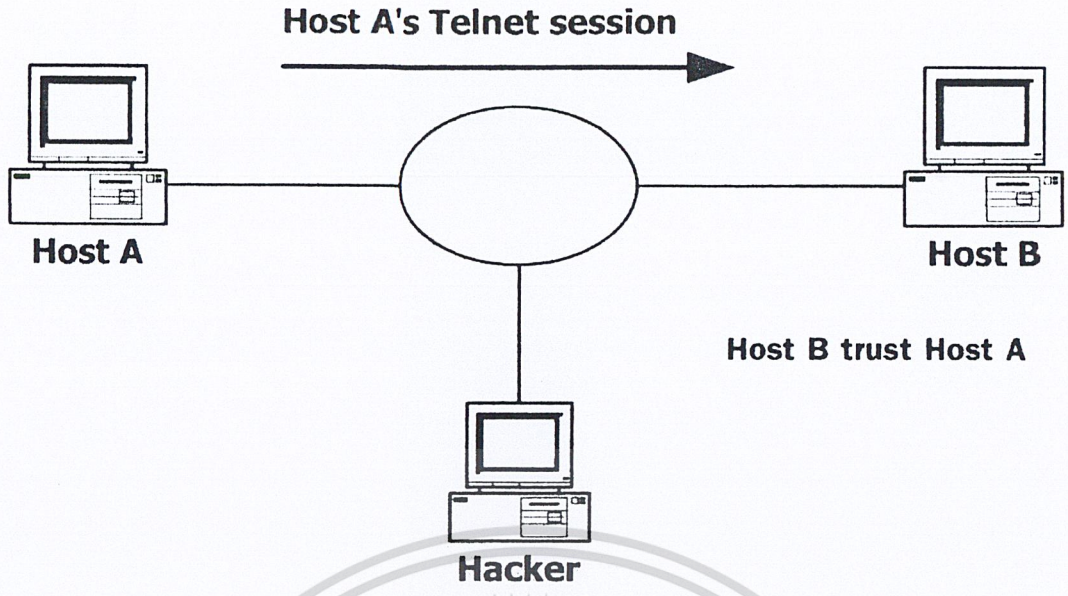
5.2.3.1 Session Hijacking

รูปแบบ:

เป็นการใช้เทคนิคของการปลอมแปลงแอดเดรสและโจมตีด้วย DoS ร่วมกัน เพื่อสวมรอยใช้งานคอนเน็คชั่นที่โฮสต์อื่นกำลังใช้งานอยู่ โดยมีขั้นตอน ดังนี้

1. **Gathering Information**

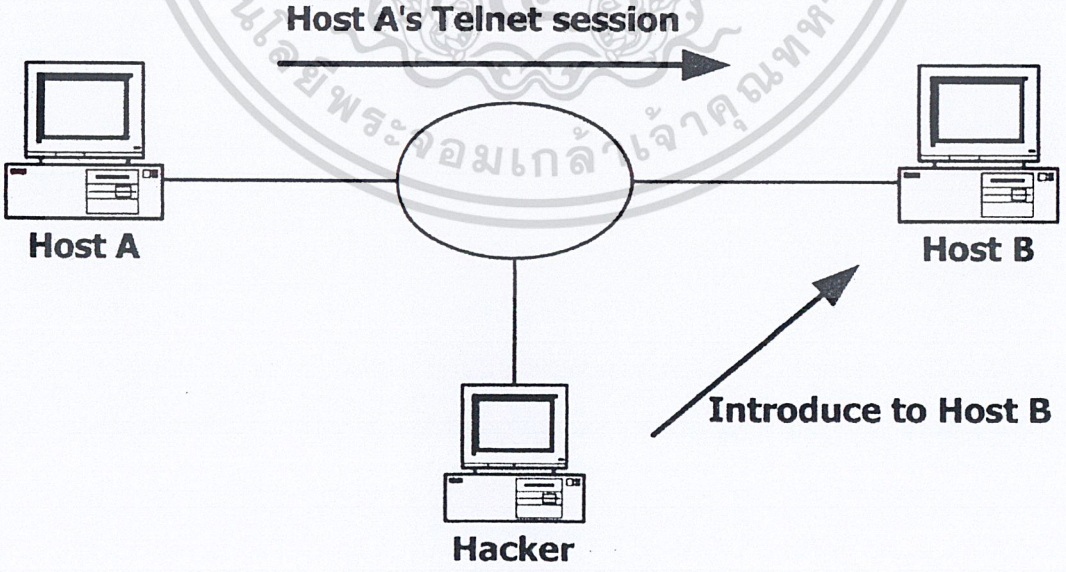
เป็นขั้นตอนการเก็บข้อมูลของแพ็กเก็ตที่ใช้สื่อสารในคอนเน็คชั่นที่ต้องการสวมรอย โดยเฉพาะอย่างยิ่ง TCP Sequence Number เพื่อนำมาใช้ในการสวมรอยแทน โดยการเก็บข้อมูลอาจจะทำได้โดยการใช้สนิฟเฟอร์ดักจับเอามาโดยตรง หรือในกรณีที่ไม่สามารถดักจับโดยตรงได้ ก็จะต้องอาศัยการคาดการณ์ TCP Sequence Number เอาเอง โดยวิเคราะห์จาก TCP Sequence Number ปัจจุบัน และ ความสัมพันธ์กับ TCP Sequence Number อื่นๆ (การคาดการณ์เป็นวิธีที่ทำได้ยากมากๆ แต่ก็มีผู้ที่เคยทำได้)



รูปที่ 5-9 Hijacking – Gathering Information

2. Interception

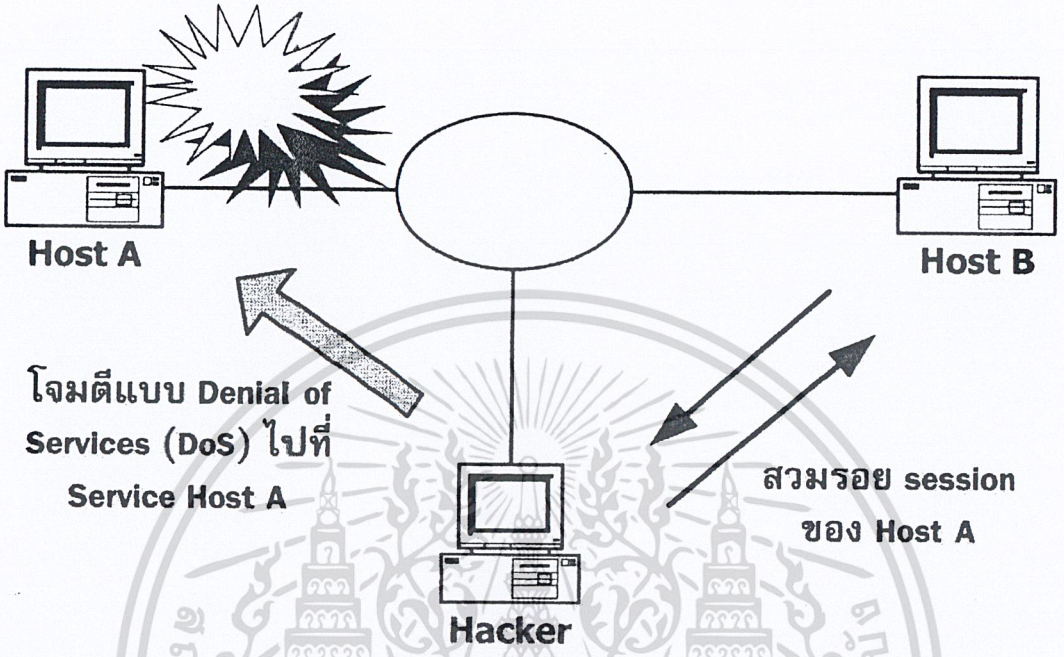
อาศัยจังหวะ แล้วทำการปลอมแอดเดรสสวมรอยเป็นเครื่องเป้าหมาย จากนั้นส่งแพ็กเก็ตไปยังเครื่องที่เป็นคู่สื่อสารของเครื่องเป้าหมาย และทำการสื่อสารต่อไปเรื่อยๆ เพื่อสวมรอยยึดคอนเน็คชั่น โดยข้อมูลของการสื่อสารจะต้องสอดคล้องกับข้อมูลของการสื่อสารเดิมระหว่างเครื่องเป้าหมายและคู่สื่อสารด้วย ซึ่งข้อมูลตรงนี้ก็ได้จากขั้นตอน Gathering Information



รูปที่ 5-10 Hijacking - Interception

3. Denial of Services

โจมตีเครื่องเป้าหมายให้หยุดทำงานด้วยการ DoS เพื่อให้การยึดคอนเน็คชั่นสมบูรณ์



รูปที่ 5-11 Hijacking – Denail of Services

การป้องกัน:

ป้องกันการปลอมแอดเดรสด้วยการตรวจสอบแอดเดรสต้นทางที่เป็น Invalid IP Address เช่นเดียวกับการป้องกันการ DoS ดังนั้นจะเห็นได้ว่า การป้องกัน Invalid IP Address นั้นสามารถลดความเสี่ยงจากภัยคุกคามได้มากทีเดียว

ป้องกันที่โฮสต์ โดยปรับปรุงระบบปฏิบัติการให้เป็นเวอร์ชั่นที่ใหม่และทันสมัยที่สุด เพราะระบบปฏิบัติการใหม่ๆ นั้น จะมีรูปแบบของการสร้าง TCP Sequence Number ที่ซับซ้อน ทำให้คาดการณ์ได้ยาก

5.2.3.2 Reconaisance: Network, Host, Application Scanning

ตราบใดที่เรายังต้องเปิดบริการบนอินเตอร์เน็ต การป้องกันการสแกนนั้นทำได้ไม่ง่ายนัก ไม่ว่าอย่างไร โฮสต์ก็ยังคงถูกสแกนได้ด้วยวิธีใดวิธีหนึ่งอยู่ดี ข้อสำคัญอยู่ที่ โฮสต์ที่ไม่ได้เปิดให้บริการภายนอกไม่สมควรถูกสแกนได้จากภายนอก ส่วนโฮสต์ที่เปิดให้บริการนั้น แม้จะหลีกเลี่ยงไม่ได้ แต่อย่างน้อยการทราบอย่างชัดเจนว่ามีโฮสต์ใดที่ถูกสแกนได้บ้าง ก็ทำให้เราพุ่งความสนใจและทำการป้องกันโฮสต์นั้นๆ ได้มากขึ้น

การป้องกัน:

1. การสแกนเน็ตเวิร์กด้วย ICMP สามารถป้องกันได้โดยการไม่เปิดให้ ICMP Echo Request และ Reply แพ็กเก็ตผ่านเข้ามาในเน็ตเวิร์กได้
2. การป้องกันการสแกนพอร์ตต่างๆ ของโฮสต์ โดยปกติจะถูกบล็อคด้วยกฎทั่วไปของไฟร์วอลล์อยู่แล้ว เพราะแพ็กเก็ตที่มีพอร์ตปลายทางที่ไม่ได้รับอนุญาตจะถูกบล็อคแต่เนิ่นๆ
3. การสแกนแอพพลิเคชั่นโดยเฉพาะเจาะจง หากเป็นการสแกนที่มีเป้าหมายเป็นโฮสต์ที่เปิดให้บริการอยู่แล้ว เช่น HTTP, SMTP จะไม่สามารถป้องกันจากสแกนได้ เพราะเราจำเป็นต้องอนุญาตให้โฮสต์ให้บริการด้วยพอร์ตนี้

บทที่ 6
จาวาและ XML

6.1 จาวา

สำหรับจาวาแอพพลิเคชั่นคือรูปแบบการใช้งานภาษาจาวาเพื่อสร้างโปรแกรมขึ้นมาใช้งาน โดยโปรแกรมที่ว่านี้จะถูกใช้งานได้ก็ต่อเมื่อนำไปติดตั้งบนเครื่องของผู้ใช้ก่อน และเครื่องของผู้ใช้ก็จะต้องปรับสภาพแวดล้อมให้พร้อมกับการใช้งานโปรแกรมดังกล่าวด้วย JVM (Java Virtual Machine) บนระบบปฏิบัติการ (Operating System) สำหรับชุดคำสั่งที่ใช้สร้างจาวาแอพพลิเคชั่น ต้องเป็นคลาสที่มีเมธอด main อยู่ภายใน ดังนั้นชุดคำสั่งภาษาจาวาที่ถูกเขียนไว้ในการสร้างคลาสและมีเมธอด main อยู่ภายในจะถือว่าไบต์โค้ดที่ได้จากการคอมไพล์เป็นไบต์โค้ดซึ่งนำไปใช้งานบน JVM ในลักษณะจาวาแอพพลิเคชั่นได้

ข้อดี : คือทำให้เกิดสภาวะที่สามารถนำไบต์โค้ดไปใช้วานได้ทุกที่ หรือ write Once, Run AnyWhere บนเครื่องคอมพิวเตอร์ที่มี JVM อยู่ เพราะ JVM ของแต่ละระบบปฏิบัติการก็จะแปลงไบต์โคด์ให้เป็นภาษาเครื่องของแต่ละระบบปฏิบัติการนั้น ๆ ทำให้ไบต์โค้ดถูกใช้งานที่ไหนก็ได้

ข้อเสีย : คือการทำงานที่ต้องแปลงไบต์โค้ดก่อนใช้งาน จึงทำให้เสียเวลาส่วนหนึ่งกับแปลงคำสั่งซึ่งมีผลต่อเวลาโดยรวม ทำให้โปรแกรมทำงานได้ช้า

6.1.1 จาวาสวิงค์ (JAVA SWING)

โดยทั่วไปทุกโปรแกรมที่ถูกใช้งานโดยผู้ใช้ ต้องสร้างส่วนตอดต่อกับผู้ใช้ (User Interface:UI) หรือเรียกว่าอินเตอร์เฟส เพื่อใช้ผู้ใช้สามารถกรอกข้อมูลหรือสั่งงานทางหน้าจอได้ ภาษาจาวาได้กำหนดชุด API เพื่อสนับสนุนอินเตอร์เฟส ซึ่งประกอบไปด้วยคอมโพเน็นต์ต่างๆ มากมายให้เลือกใช้ ยกตัวอย่าง เช่น JTextBox, JButton, JPanel, JRadio เป็นต้น

บริษัท Sun ได้กำหนดอินเตอร์เฟสทั้งหมดอยู่ในชื่อของ JFC (Java Foundation Classes) โดยมีสวิงซ์ (Swing API) เป็นส่วนหนึ่งด้วย ในชุดของ JFC ยังประกอบไปด้วย AWT API, Drag and Drop API, Accessibility API, Look and Feel API ยังมาช่วยสร้างส่วนอินเตอร์เฟสใช้มีประสิทธิภาพมากขึ้น

ในลักษณะของสวิงซ์ นอกจากจะเป็นเครื่องมือสำหรับสร้างส่วนอินเตอร์เฟสดังที่กล่าวแล้ว สวิงซ์ยังให้ความแตกต่างจากรูปแบบอินเตอร์เฟสเดิมที่จาวาใช้คือ AWT กล่าวคือใน AWT ให้ลักษณะของอินเตอร์เฟสที่เหมือนกันในการใช้งาน แต่มีข้อจำกันที่ให้ลักษณะที่แตกต่างกันเมื่อนำไปใช้งานในระบบปฏิบัติงานที่แตกต่างกัน เนื่องจากการวาดคอมโพเน็นต์ของแต่ระบบใช้วิธีที่แตกต่างกัน

สวิงค์จะลดจุดบกพร่องเหล่านี้ให้หน้าตาอินเตอร์เฟสหมือนกัน ไม่ว่าจะนำโปรแกรมไปใช้ในระบบปฏิบัติการใดๆ ก็ตาม

6.1.2 JDBC Data Access

โปรแกรมที่ใช้งานส่วนใหญ่มักจะเป็นโปรแกรมที่ใช้งานติดต่อกับระบบฐานข้อมูล ถ้าโปรแกรมใช้งานและฐานข้อมูลเป็นผลิตภัณฑ์ที่มาจากผู้ขายรายเดียวกัน การสร้างระบบโปแกรมขึ้นมาก็สามารถทำได้อย่างไม่ยุ่งยาก เนื่องจากผู้ขายย่อมจะสร้างซอฟแวร์ของตัวเองให้สามารถทำงานประสานกันได้อย่างดีที่สุดอยู่แล้ว แต่ในความเป็นจริงของการใช้งานทั่วไปเป็นไปได้ยากที่จะเลือกมาจากผลิตภัณฑ์ที่มาจากเจ้าของเดียวกันเดียวกัน ผู้สร้างโปรแกรมจะเลือกใช้ฐานข้อมูลที่ดีที่สุดในความคิดของตัวเอง และเลือกเครื่องมือที่ดีที่สุดในการสร้างโปรแกรม ดังนั้นจะเห็นได้ว่าเครื่องมือที่ใช้มาจากผู้ขายคนละรายกัน จากรูปแบบดังกล่าว จึงเกิดมีผู้ทำหน้าที่เป็นตัวกลางของการสื่อสารระหว่างโปรแกรมใช้งานกับฐานข้อมูลขึ้นมา ซึ่งเรียกว่ามิดเดิลแวร์ (Middleware)

ODBC (Open Database Connectivity) ถือเป็นมิดเดิลแวร์ตัวหนึ่งที่เป็นตัวกลางการสื่อสารและมีใช้อยู่ในระบบปฏิบัติการวินโดว์ ODBC พัฒนามาจากภาษาซีเป็นหลัก และเมื่อต้องการสร้างโปรแกรมด้วยภาษาจาวา จึงต้องมีมิดเดิลแวร์ที่สามารถสื่อสารกับจาวาได้ ดังนั้นบริษัท Sun จึงได้พัฒนาชุด API ขึ้นมาจัดการด้านนี้โดยเฉพาะเรียกว่า JDBC

JDBC จัดเป็นชุด API ที่ให้โปรแกรมเมอร์ภาษาจาวาสามารถสร้างแอพพลิเคชั่นหรือแอปเพล็ตเพื่อติดต่อกับฐานข้อมูลด้วย JDBC Driver เพื่อเข้าถึงระบบฐานข้อมูลเชิงสัมพันธ์จากผลิตภัณฑ์ได้หลากหลาย เช่น Oracle, SQL Server, Informix, DB2, mySQL โดยเป็นการใช้คำสั่ง SQL ซึ่งเป็นคำสั่งที่สนับสนุนอยู่ทุก ๆ ระบบปฏิบัติการ

6.2 XML

XML คือคำที่มาจะคำย่อของประโยคต่อไปนี้ eXtensible Markup Language เป็นมาตราฐานใหม่ของการจัดทำเอกสารอย่างแน่นอนซึ่งนอกจากจะเป็นมาตราฐานในการทำเอกสารบนเว็บแล้วยังจะรวมถึงเอกสารในรูปแบบอื่น ๆ ด้วย อาทีการแลกเปลี่ยนข้อมูลระหว่างข้อมูลจากฐานข้อมูลต่างแฟลตฟอร์มกัน การสร้างไฟล์คอนฟิก เป็นต้น

โครงสร้างของเอกสาร XML เอกสาร XML ประกอบไปด้วย 2 ส่วนหลักคือ Prolog Element และ Document Element (หรือเรียกอีกชื่อว่า Root Element)

6.2.1 ส่วนแรก (Prolog) ส่วนของโปรล็อกประกอบไปด้วย 3 บรรทัดต่อไปนี้

ในบรรทัดแรกเรียกว่า XML Declaration เป็นการประกาศให้รู้ว่าเอกสารนี้คือเอกสาร XML และเป็นการประกาศเวอร์ชั่นของ XML (ในปัจจุบันเวอร์ชั่นใหม่สุดคือ 1.0) การใส่ค่า XML Declaration เป็นทางเลือกที่จะประกาศหรือไม่ประกาศก็ได้ แต่ตามมาตรฐานขอกำหนดแล้ว ควรรวมการประกาศนี้ไว้ในเอกสารด้วย หากไม่ประกาศ XML Declaration ในเอกสารจะต้องประกาศไว้ ณ จุดเริ่มต้นเสมอ

บรรทัดที่สองของโปรล็อกคือ บรรทัดว่างเพื่อช่วยให้เอกสารอ่านง่ายขึ้น เราจะสามารถเพิ่มบรรทัดว่างระหว่างรายการในส่วนของโปรล็อกกี่บรรทัดก็ได้ ตัวประมวลผล XML จะข้ามและไม่นำบรรทัดว่าเหล่านั้นมาประมวลผล

6.2.2 ส่วนที่สอง (Document Element)

ในส่วนที่สองของเอกสาร XML คือ Element เดี่ยวหรือเรียกว่า Document Element หรือ Root Element ซึ่งสามารถบรรจุ Element เพิ่มเติมในเอกสาร XML ได้

ในเอกสาร XML นั้น Element จะแสดงลักษณะโครงสร้างของเอกสาร และแสดงส่วนประกอบของเนื้อหาอยู่ภายใน ลักษณะของ Element ประกอบไปด้วยแท็กเริ่มต้น, เนื้อหาภายใน Element และแท็กปิดท้าย เนื้อหาภายใน Element จะสามารถเป็นเป็นได้ทั้งข้อมูลหรือ Element อื่น ๆ ที่ซ้อนกันอยู่ภายในหรือทั้งสองแบบ

บรรทัดที่สามของโปรล็อกเรียกว่าหมายเหตุหรือคอมเมนท์ หารเพิ่มคอมเมนท์ในเอกสาร XML นับเป็นทางเลือกหนึ่ง การใส่คอมเมนท์ช่วยให้เอกสารน่าอ่านยิ่งขึ้น โดยเริ่มต้นด้วย <! - - และปิดท้ายด้วย - -> ผู้สร้างเอกสารสามารถที่จะพิมพ์ข้อความตามที่ต้องการได้ และตัวประมวลผล XML จะไม่นำส่วนมาประมวลผล

6.2.3 กฎพื้นฐานของ XML

ในส่วนนี้เป็นกฎพื้นฐานเล็กน้อยสำหรับใช้สร้างเอกสารที่มีรูปแบบที่ถูกต้อง (Well-Formed XML Document) เป็นกฎข้อหนึ่งที่ต้องปฏิบัติตาม จึงจะทำให้เอกสารนั้นมีรูปแบบที่ถูกต้อง จะต้องปฏิบัติตามกฎข้อบังคับเหล่านี้

1. เอกสารต้องมี Element ระดับบนสุดเพียง Element เดียวเท่านั้น (ซึ่งเรียกว่า Document Element หรือ Root Element) โดย Element อื่น ๆ ทั้งหมดจัดซ้อนกันอยู่ภายใน
2. Element จะต้องซ้อนกันอยู่อย่างถูกต้อง นั่นหมายความว่าถ้ามี Element ซ้อนอยู่ภายใน Element อื่น ๆ จะต้องปิด Element นั้นภายใน Element เดียวกับที่ Element นั้นซ้อยอยู่
3. แต่ละ Element จะต้องมีแท็กเริ่มต้นและแท็กปิดท้าย ซึ่งจะต่างกับ HTML ตรงที่ XML ไม่สามารถให้คุณละเว้นแท็กปิดท้ายได้ เพราะเมือแท็กไม่ครบคู่จะทำให้เกิดดารผิดพลาดของการแสดงผลออกมา
4. ชื่อของ Element ในแท็กเริ่มต้นจะต้องมีชื่อตรงกับชื่อของแท็กปิดท้ายเท่านั้น
5. ชื่อของ Element มีลักษณะ Case-sensitive (การแยกระหว่างตัวอักษรตัวใหญ่กับตัวอักษรตัวเล็กไม่เหมือนกัน)

6.2.4 ข้อกำหนดโครงสร้างสำหรับ XML

ส่วนนี้ จะมาทำความรู้จักกับ เอกสารในรูปแบบการจัดสร้างที่ถูกต้อง และส่วนควบคุมโครงสร้างเอกสารที่เรียกว่า DTD

> "DTD คือแฟ้มข้อมูล (หรือหลายแฟ้มข้อมูลที่ใช้งานร่วมกัน) ซึ่งบรรจุข้อกำหนด และกฎเกณฑ์ของเอกสาร ชุดข้อกำหนดเหล่านี้ สำหรับการกำหนดรูปแบบอิลิเม็นต์ ตัวอย่างเช่น หากต้องการเอกสารที่มีอิลิเม็นต์ <LIST> ที่มีอิลิเม็นต์ <ITEM> บรรจุอยู่ภายใน ข้อกำหนดในแฟ้มข้อมูล DTD จะมีรูปแบบดังนี้
>
> <!ELEMENT item (#pcdata)>
>
> <!ELEMENT list (item)+>
>
> ซึ่งอธิบายความหมายคือ อิลิเม็นต์ items บรรจุข้อความใดๆ และอิลิเม็นต์ list บรรจุอิลิเม็นต์ item อีกที
>
> ดังนั้น DTD เป็นรูปแบบภาษา ซึ่งทำให้สามารถตรวจสอบเอกสาร ที่นำเอาข้อกำหนด DTD ไปใช้ ว่าถูกจัดสร้างตามความต้องการหรือไม่ ทำให้ระบบการ rendering สามารถเข้าใจตัวเอกสารได้ดี และดึงไปใช้งานได้ถูกต้อง"

DTD มีความซับซ้อน

จากคำอธิบายในส่วนต้น DTD มีความซับซ้อน หมายถึงการสร้าง DTD มีรายละเอียดและขั้นตอนที่ยุ่งยากพอสมควร

แม้ DTD จะทำให้การสร้างเอกสารมีความยุ่งยากก็ตาม แต่มีเหตุผลอยู่ 2 ข้อที่ควรรู้ก่อน

- XML ไม่จำเป็นต้องใช้ DTD เสมอไป
- ถ้าแม้จะต้องใช้ DTD แต่ก็จะมีการสร้าง DTD ที่เป็นมาตรฐานให้ใช้อยู่แล้ว

DTD ที่เป็นมาตรฐาน ถูกพัฒนาโดยผู้ที่สนใจ ในเอกสารเฉพาะด้าน และมีให้พวกเรา ได้ดึงมาใช้งาน เพื่อให้เกิดรูปแบบมาตรฐานที่ชัดเจน และเข้าใจข้อมูลกัน ในเอกสารประเภทเดียวกัน

3 ส่วนสำคัญ

เอกสาร XML มีส่วนประกอบที่สำคัญอยู่สามส่วน สองส่วนเป็นสิ่งที่จำเป็น ในขณะที่อีกส่วนเป็นทางเลือก ที่จะมีหรือไม่ก็ได้

- ส่วนแรก คือส่วนของเนื้อหา หรือคอนเท็นซ์นั่นเอง ซึ่งทำให้เอกสารมีข้อมูลสำหรับดูโดยผู้อ่าน คอนเท็นซ์นี้เป็นได้ทั้งข้อความ รูปภาพ ส่วนนี้ถูกสร้างขึ้นมาจาก อิลิเม็นต์ ในที่นี้คือไฟล์ XML

- ส่วนที่สอง คือกฎเกณฑ์และข้อกำหนด โครงสร้างของเอกสาร ในที่นี้คือไฟล์ DTD ส่วนนี้ถือเป็นทางเลือก ซึ่งจะเลือกใช้หรือไม่ก็ได้
- ส่วนสุดท้าย คือ Stylesheet คือ ลักษณะข้อกำหนดสำหรับการแสดงผลลัพธ์นั้นเอง ในที่นี้คือไฟล์ XSL

เอกสารที่ถูกต้อง

เรื่องของ DTD เอกสารที่อยู่ในรูปแบบ well-formed เป็นเอกสารที่เรียกว่าถูกต้องตามข้อกำหนดของ XML

แต่เอกสารที่ไม่ถูกต้องตามรูปแบบ XML ก็สามารถเป็นเอกสารที่ดีได้เช่นกัน เนื่องจากในปัจจุบันข้อมูล บนระบบอินเตอร์เน็ตมีอยู่มากมาย ที่ไม่ได้สร้างเป็นเอกสารที่ถูกต้องตามแบบ XML

นั่นคือความต้องการส่วนหนึ่ง ที่ DTD จะสามารถบังคับเอกสาร XML ที่สร้างมาจาก DTD เดียวกัน ให้เป็นเอกสารที่ถูกต้องตามแบบ XML (ต้องเป็น DTD ที่ถูกต้องด้วยนะ)

เอกสารรูปแบบ well-formed

แนวคิดของเอกสารแบบ well-formed ถูกนำมาใช้สำหรับ XML ในกรณีที่ ไม่สามารถสร้างเอกสาร XML ด้วยข้อกำหนด DTD

> "จากตัวอย่างของแท็ก <IMG> ใน HTML เป็นแท็กที่เรียกว่าอิลิเม็นต์เปล่า เนื่องจากว่าแท็กดังกล่าว ไม่จำเป็นต้องอาศัยแท็กปิด เช่นเดียวกัน หาก XML มีอิลิเม็นต์ที่มีแท็กปิด จึงไม่เป็นเอกสารในรูปแบบ '***well-formed***' "

เอกสาร HTML ไม่เป็นเอกสารในรูปแบบ well-formed

เนื่องจากคำสั่ง หรือแท็กในภาษา HTML มีหลายแท็ก ที่มีเฉพาะแท็กเปิด โดยที่ไม่จำเป็นต้องใช้แท็กปิดเลย ทำให้เอกสารที่สร้างด้วยแท็ก HTML จึงไม่ถือเป็นเอกสาร well-formed

แต่เอกสาร XML ต้องเป็นเอกสารรูปแบบ well-formed

หากคุณกำลังจำสร้างเอกสาร XML แล้วจำไว้เสมอว่า จะต้องสร้างเอกสารที่เป็นรูปแบบ well-formed

6.2.5 ข้อบังคับเอกสาร well-formed

1. อิลิเม็นต์ที่ถูกต้อง อิลิเม็นต์ต้องแท็กเปิด และต้องปิดด้วยด้วยแท็กปิดเสมอ เช่น <...> ..... </...> ยกเว้นอิลิเม็นต์ว่าง

2. ข้อมูลแอททริบิวต์ต้องอยู่ภายในเครื่องหมาย '...' หรือ "..." หากแอททริบิวต์ไม่ได้อยู่ในสัญลักษณ์ดังกล่าว เอกสารจะไม่ถือว่าเป็นแบบ well-formed

3. อิลิเม็นต์ว่าง หากต้องการใช้งานอิลิเม็นต์ว่าง ต้องสร้างอิลิเม็นต์ว่างให้ถูกต้อง

เอกสาร XML ไม่จำเป็นต้องมีอิลิเม็นต์ที่มีแท็กเปิดและแท็กปิดเสมอ เรายังสามารถสร้างอิลิเม็นต์ว่าง หรือ Empty Element บรรจุลงในเอกสารได้ด้วย

อิลิเม็นต์ว่าง จะต้องไม่บรรจุตัวอักษรใดๆ ลงภายในอิลิเม็นดังกล่าว สามารถเขียนได้ในสองรูปแบบ

- อิลิเม็นต์ที่มีแท็กเปิดและแท็กปิด โดยที่ไม่มีข้อความ หรือคอนเท็นซ์ อยู่ระหว่างแท็ก (ยกเว้นตัวอักษรขึ้นบรรทัดใหม่)

  <VALUE></VALUE>

- อิลิเม็นต์เฉพาะ ที่เรียกว่า อิลิเม็นต์ว่างตามรูปแบบ <.../>

  <RATE/>

อิลิเม็นต์ว่างที่มีแอททริบิวต์ แน่นอน อิลิเม็นต์ว่าง สามารถบรรจุข้อมูลแอททริบิวต์ได้ด้วย ดังตัวอย่างต่อไปนี้

<BL number="0">..... </BL>

4. ต้องไม่มีตัวอักษรประเภท Markup อยู่ในส่วนคอนเท็นซ์ คือข้อมูล **< หรือ &**
เนื่องจากตัวอักษร < ถูกใช้สำหรับการสร้างแท็กแล้ว และ & ถูกใช้สำหรับการแสดงสัญลักษณ์พิเศษ
หากต้องการแสดงข้อมูลตัวอักษรดังกล่าว ในส่วนคอนเท็นซ์จริงๆ ให้ใช้สัญลักษณ์ ใช้งาน **<** สำหรับแสดงตัวอักษร < และ **&** สำหรับแสดงตัวอักษร &

5. อิลิเม็นต์ต้องซ้อนกันอย่างมีรูปแบบ หากมีการสร้างเอกสาร XML ที่มีการวางซ้อนกันของอิลิเม็นต์ในหลายๆชั้น การวางซ้อนกันต้องมีรูปแบบ เป็นชั้นๆ ห้ามวางอิลิเม็นต์ในลักษณะที่ชั้นไขว้กัน

<MULTIPORT>
<VALUE>multiport</VALUE>
<SPORT>7, 1, 2, 3</SPORT>

<DPORT>null</DPORT>

<PORT>null</PORT>

</MULTIPORT>

ตัวอย่างด้านบนเป็นเอกสารที่อิลิเม็นต์วางซ้อนกันอย่างถูกต้อง

<MULTIPORT>

<VALUE>multiport</VALUE>

<SPORT>7, 1, 2, 3</SPORT>

<DPORT>null</PORT>

<PORT>null </DPORT>

</MULTIPORT>

ตัวอย่างด้านบนเป็นเอกสารที่วางซ้อนอิลิเม็นต์อย่างไม่ถูกต้อง

6.2.6 การมองโครงสร้างของเอกสารเป็นทรีด้วย DOM

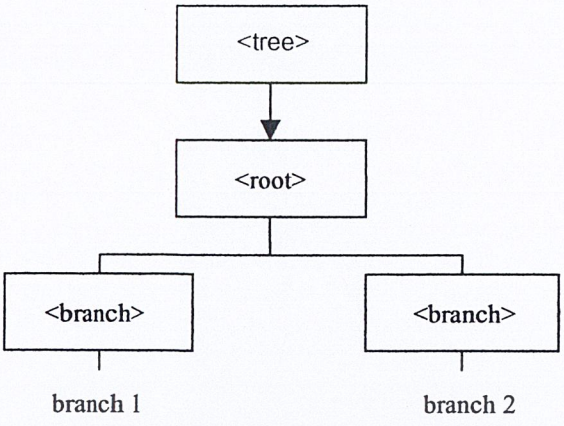
XML DOM คือแอพลิเคชั่นโปรแกรมมิ่งอินเทอร์เฟซ (Application Programming Interface, API) สำหรับเอกสาร XML ซึ่งจะกำหนดว่าเอกสาร XML จะมีการเข้าถึงและการโยกย้ายรวมไปถึงการจัดการต่างๆ ได้อย่างไรบ้าง DOM เป็นมาตราฐานของ W3C โดยมีเป้าหมายคือต้องการสร้างอินเทอร์เฟซที่เป็นมาตรฐานและจะต้องใช้ได้กับแอพพลิเคชั่นได้หลากหลาย ซึ่ง DOM ก็สามารถใช้ได้กับภาษาโปรแกรมหรือระบบปฏิบัติการใดก็ได้ ด้วยวิธีการแบบ DOM โปรแกรมเมอร์จะสามารถสร้างเอกสาร XML ขึ้นมา เข้าไปในโครงสร้างระดับต่างๆ ได้รวมทั้งเพิ่ม, ลบ และแก้ไขอิลิเม็นต์ต่างๆของเอกสารได้

มุมมองของ DOM

การทำงานของ DOM จะเริ่มต้นเมื่อโปรแกรมหรือแอพพลิเคชั่นไปเรียก พาร์สเซอร์ให้ไปโหลดเอกสาร XML เข้าไปสู่ในหน่วยความจำคอมพิวเตอร์ เมื่อเอกสารถูกโหลดเสร็จเรียบร้อย แล้ว DOM ก็จะทำให้โปรแกรมสามารถดึงข้อมูลมาใช้หรือดำเนินการต่าง ๆ ได้ กับข้อมูล

DOM จะมีมุมมองไปยังเอกสาร XML เป็นแบบโครงสร้างต้นไม้หรือทรี (Tree) ระดับบนสุดของทรีเราจะเรียกว่า “documentElement” ซึ่งอีลิเมนต์ดังกล่าวจะแตกสาขาออกไปเป็นอีลิเมนต์ย่อย ๆ ตั้งแต่ 1 อีลิเมนต์เป็นต้นไป ซึ่ง Element ย่อยที่แตกออกจาก documentElement เราจะเรียกว่า “โหนดลูก” (childNodes)

DOM จะมองเอกสารเป็นทรีดังนี้คือ



รูปที่ 6-1 DOM มอง XML ในรูปแบบทรี

การจะเข้าถึงในแต่ละโหนดในทรีเราจะใช้ Node Interface Model ซึ่งภายได้การทำงานของบราวเซอร์ของไมโครซอฟต์ (อินเทอร์เน็ตเอ็กซ์พลอเรอร์เวอร์ชัน 5.0 ขึ้นไป) จะมี Microsoft จะมี parser ฝังอยู่ในตัวอยู่แล้ว โดยตัว Microsoft XML parser จะสนับสนุนฟังชั่นก์ที่จำเป็นทั้งหมดที่ใช้ในการเข้าถึงแต่ละโหลดของทรี, ค่าแอตทริบิวต์ของโหนด และจัดการกับค่าต่างๆ ของโหนด อาทิเพิ่ม, ลบ และแก้ไขข้อมูล

จุดเด่นของ DOM

1. โปรแกรมจะพาร์สเอกสารเพียงครั้งเดียว เมื่อ DOM ได้สร้างทรีขึ้นมาแล้ว ทรีนั้นก็จะอยู่ในหน่วยความจำตลอดจนกว่าเราจะสั่งยกเลิก และในขณะที่ทรีอยู่ในหน่วยความจำเราก็สามารถเข้าถึงข้อมูลภายในทรีได้อย่างรวดเร็ว
2. ข้อมูลที่สร้างเป็นทรีจะทำให้เกิดความสัมพันธ์ ทำให้การเรียกใช้งานทำได้ง่าย

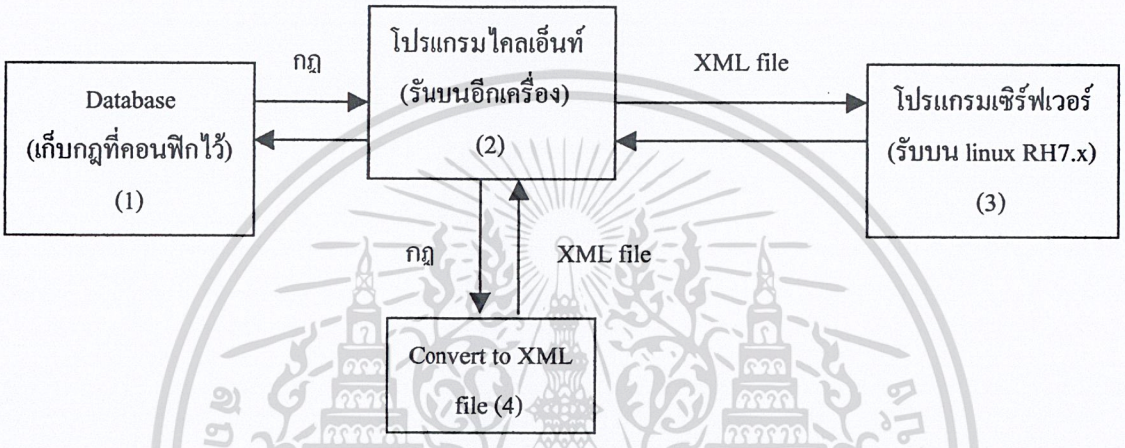
จุดด้อยของ DOM

1. ในกรณีที่เอกสารมีขนาดใหญ่และซับซ้อน การสร้างทรีจะทำได้ช้า และกินหน่วยความจำมาก
2. ผู้พัฒนาโปรแกรมจำเป็นที่จะต้องเข้าใจถึงโครงสร้างโดยรวมของเอกสารเป็นอย่างดี จึงจะสามารถเขียนโปรแกรมโดยใช้ DOM ได้ กล่าวคือในส่วนของการเขียนโค้ดด้วยวิธีการ DOM นั้นค่อนข้างยุ่งยาก

บทที่ 7
โครงสร้างของโปรแกรม

ในส่วนของการออกแบบโปรแกรมจะมีส่วนประกอบด้วยกัน 4 ส่วนคือ

1. ส่วนที่เป็นฐานข้อมูล (database) ที่เอาไว้เป็นกฎต่าง ๆ ที่คอนฟิกเอาไว้
2. ส่วนของโปรแกรมไคลเอ็นท์ (รับบนอีกเครื่องหรือเครื่องเดียวกันเพื่อจัดการคอนฟิกกฎ)
3. ส่วนของโปรแกรมเซิร์ฟเวอร์
4. ส่วนตัวที่ทำหน้าที่แปลงจากกฎในฐานข้อมูลไปเป็นไฟล์ XML



รูปที่ 7-1 แสดงโครงสร้างโดยรวมของโปรแกรมไฟร์วอลล์

ลักษณะของโปรแกรมแบ่งออกเป็นสองส่วนคือส่วนของโปรแกรมไคลเอ็นท์และส่วนของโปรแกรมเซิร์ฟเวอร์ซึ่ง โดยโปรแกรมไคลเอ็นท์นั้นจะเป็นโปรแกรมหลักที่ทำหน้าคอนฟิกกฎต่าง ๆ ของไฟร์วอลล์ โดนตัวโปรแกรม สามารถที่จะนำไปรับบนเครื่องไฟร์วอลล์เอง เพื่อเซทกฎบนเครื่องนั้นหรือจะนำไปรับอีกเครื่องหนึ่งก็ได้ โดยถ้านำไปรับอีกเครื่องโปรแกรม ไคลเอนท์จะติดต่อไปยังโปรแกรมเซิร์ฟเวอร์เพื่อส่งค่าของกฎในรูปของไฟล์ XML แล้วตัวโปรแกรมเซิร์ฟเวอร์เองนั้นก็จะมีตัวที่ทำหน้าที่แปลงจากไฟล์ XML ไปเป็นในรูปแบบของ iptables ไฟร์วอลล์และยังจัดการใช้ไฟล์ที่แปลงนั้นรันและสามารถใช้กฎนั้นได้ทันที

โดยในแต่ละส่วนจะออกแบบและมีสร้างดังต่อไปนี้

7.1 ส่วนของฐานข้อมูล

ในโปรแกรมนี้จำเป็นที่จะต้องใช้ฐานข้อมูลเพื่อเก็บกฎต่าง ๆ ในการคอนฟิกไว้ ฐานข้อมูลที่เราเลือกใช้นี้คือ mySQL ซึ่งเป็นฐานข้อมูลที่ไม่ต้องเสียค่าใช่จ่ายและสามารถรันบนหลาย ๆ ระบบปฏิบัติการได้ เช่น Windows, linux เป็นต้น

การติดตั้ง MySQL สำหรับ Unix Platfrom (การติดตั้งจาก source code)

ระบบปฏิบัติการที่จะคอมไพล์เพื่อติดตั้ง MySQL จะต้องมียูทิลิตี้ ดังต่อไปนี้ติดตั้งก่อน

- GNU gunzip
- GNU tar
- GNU make
- gcc เวอร์ชั่น 2.95.2 ขึ้นไปสำหรับการคอมไพล์กับ Mysql v3.23.x

การเลือกใช้แพ็กเกจที่ติดตั้งควรเลือกแพ็กเกจที่สนับสนุนทรานแซ็คชั่น เพื่อที่โปรแกรมที่ใช้จะสามารถ ใช้การยืนยันในกรณีที่อาจเปลี่ยนใจไม่ใช้กฎนั้น (การ commit หรือ rollback) เช่น ใน Windows ใช้เวอร์ชั่นที่เป็นเวอร์ชั่น max คือ mysql-max-3.23.47-win หรือถ้าใน linux จะเป็น mysql-4.0.1-alpha.tar.gz

และจะต้องดูว่าฐานข้อมูลที่ลงนั้นมีประเภทของตารางข้อมูลหรือไม่ โดยใช้คำสั่ง

mysql> show variables;

โดยที่ชนิดของตารางที่สนับสนุนทรานแซ็คชั่นคือ INNOBASE, BDB เมื่อใช้คำสั่งดูแล้วจะต้องมีค่าของตัวแปรปรากฏ YES

```
have_bdb                | YES
have_gemini             | NO
have_innodb             | DISABLED
have_isam               | YES
have_raid               | NO
have_openssl            | NO
```

หรือถ้าที่ have_bdb ไม่ปรากฏ YES อาจจะเปลี่ยน type เป็น INNOBASE ได้

การติดตั้ง Mysql โดยการ compile source code ตามปกติจะมีขั้นตอนและคำสั่ง ดังนี้

```
# groupadd mysql
# useradd –g mysql mysql
# gunzip < mysql-VERSION.tar.gz |tar –xvf –
# cd mysql-VERSION
# ./configure –prefix=/usr/local/mysql
# make
# make install
# scripts/mysql_install_db
# chown –R mysql /usr/local/mysql
# chgrp –R  mysql /usr/local/mysql
```

```
# /usr/local/mysql/bin/safe_mysqld –user=mysql &
```

หรือการใช้ rpm ดังนี้

```
# rpm –rebuild MYSQL-VERSION.src.rpm
```

เมื่อติดตั้ง mySQL ได้แล้วต่อไปก็จะเป็นในส่วนของการสร้างฐานข้อมูล ที่ mysql prompt เราจะสร้างได้ดังนี้

```
// ตาราง Black List
create table blacklist
(
ID int(5) not null default '1' auto_increment,
HostAddr text default NULL,
Status text default NULL,
primary key(id)
)
type=BDB;

// ตาราง DefineVar
create table definevar
(
ID int(5) not null default '1' auto_increment,
Name text default NULL,
Value text default NULL,
Type text default NULL,
Comment text default NULL,
primary key(id)
)
type=BDB;

// ตาราง DefineChain
create table definechain
(
ID int(5) not null default '1' auto_increment,
```

```
Name text default NULL,
Policy text default NULL,
Comment text default NULL,
primary key(id)
)
type=BDB;

// ตาราง Confige
create table confige
(
ID int(5) not null default '1' auto_increment,
Chain text default NULL,
Service text default NULL,
src_IP text default NULL,
src_Port text default NULL,
src_Device text default NULL,
des_IP text default NULL,
des_Port text default NULL,
des_Device text default NULL,
Protocal text default NULL,
Action text default NULL,
notSrcIP text default NULL,
notSrcPort text default NULL,
notSrcDevice text default NULL,
notDesIP text default NULL,
notDesPort text default NULL,
notDesDevice text default NULL,
en_dis text default NULL,
tablename text default NULL,
notprotocal text default NULL,
jump text default NULL,
frag text default NULL,
syn text default NULL,
notfrag text default NULL,
```

notsyn text default NULL,

flagmark text default NULL,

notflagmark text default NULL,

flagcomp text default NULL,

nottcp_op text default NULL,

tcp_op text default NULL,

noticmptype text default NULL,

icmptype text default NULL,

matchs text default NULL,

msource text default NULL,

limits text default NULL,

limitrate text default NULL,

limitB text default NULL,

limitBnum text default NULL,

sdport text default NULL,

markvalue text default NULL,

userid text default NULL,

groupid text default NULL,

processid text default NULL,

sessionid text default NULL,

state text default NULL,

tos text default NULL,

level text default NULL,

prefix text default NULL,

logseq text default NULL,

logop text default NULL,

logipop text default NULL,

setmark text default NULL,

rejecttype text default NULL,

settos text default NULL,

tosource text default NULL,

todes text default NULL,

toport text default NULL,

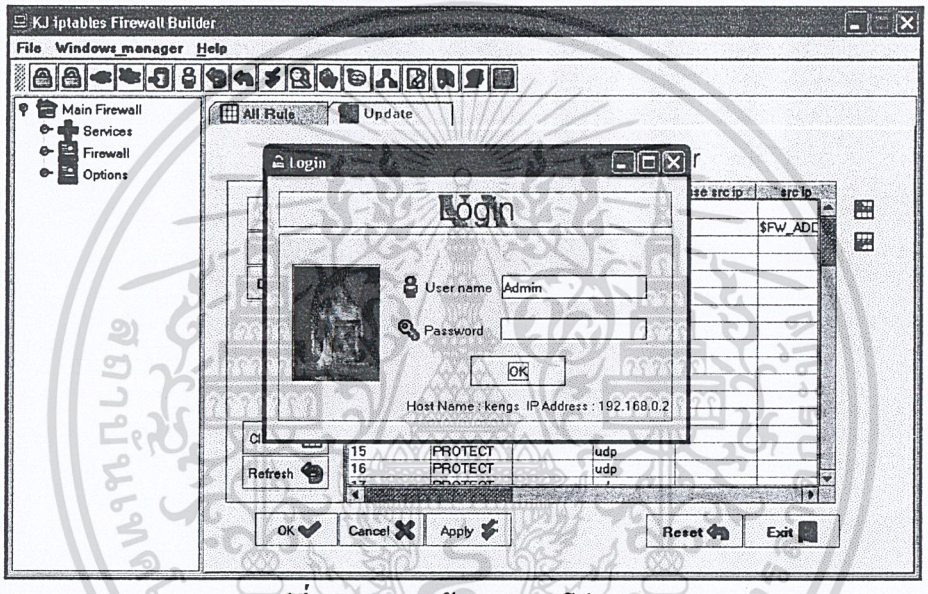
notmsource text default NULL,

```
natto text default NULL,
mul_sport text default NULL,
mul_dport text default NULL,
primary key(id)
)
type=BDB;

// ตาราง User
create table user
(
ID int(5) not null default '1' auto_increment,
username text default NULL,
password text default NULL,
primary key(id)
)
type=MyIsam;

// ตาราง Options
create table options
(
ID int(5) not null default '1' auto_increment,
drespping text default NULL,
drespbroad text default NULL,
drespicmpredi text default NULL,
enbaderror text default NULL,
dipspoof text default NULL,
entcpsyn text default NULL,
dsrcrouting text default NULL,
logmar text default NULL,
dydrop text default NULL,
primary key(id)
)
type=BDB;
```

7.2 ส่วนของการโปรแกรมไคล์เอนท์

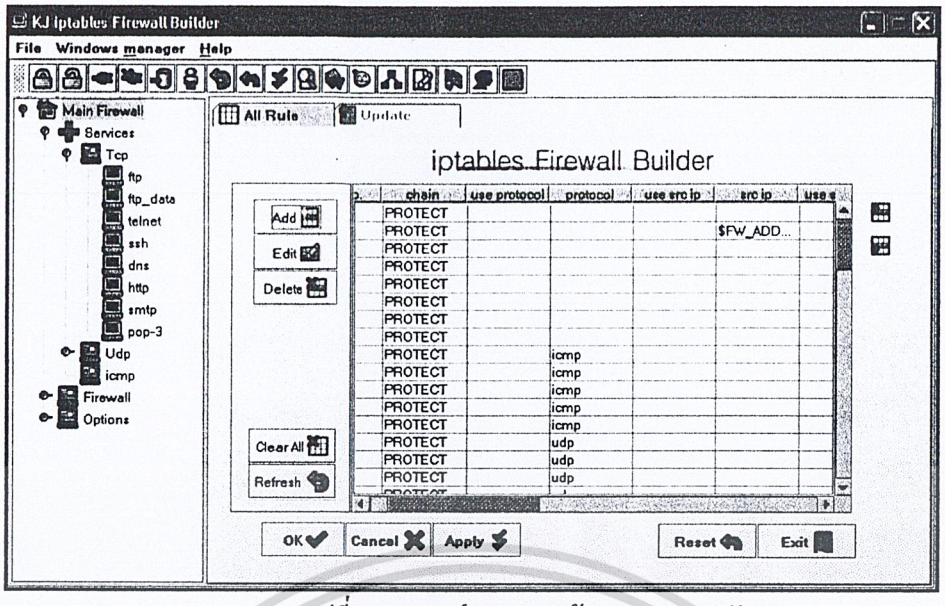
โปรแกรมนี้จะสร้างขึ้นโดยใช้ภาษาจาวา ในการเขียนเพื่อที่จะสามารถันบนได้ทุก ๆ ระบบปฏิบัติการ โดย ในส่วนของโปรแกรมนั้นจะพิจารณาจากสคริปท์ของ iptables ว่ามีข้อมูลอะไรบ้างที่ต้องนำไปใช้ในเพื่อที่จะสร้างโปรแกรม ซึ่งโดยสคริปท์ของ iptables แล้วจะเป็นการยากที่ใช้ใช้เพราะต้องเขียนอยู่บน Text Editor ซึ่ง สร้างปัญหาให้กับผู้ใช้พอสมควร จึงได้สร้างโปรแกรมขึ้นมาจัดการกับสคริปท์ของ iptables นี้

ส่วนของโปรแกรมไคลเอ็นท์ที่นำมาจัดการกับไฟร์วอลล์ iptables ให้มีสามารถใช้ได้สะดวกยิ่งขึ้น มีส่วนประกอบดังนี้



รูปที่ 7-2 แสดงหน้าจอ login โปรแกรม

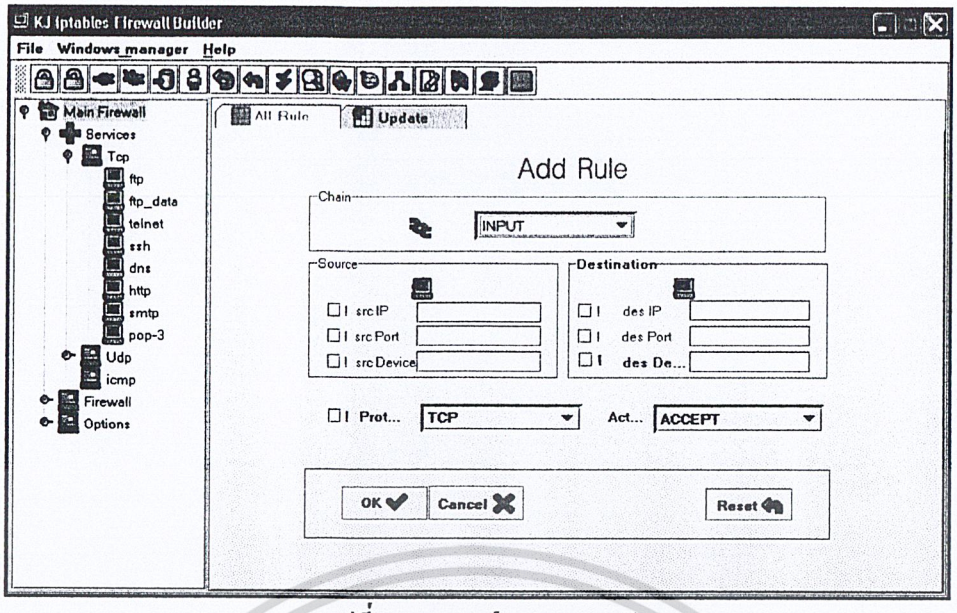
ก่อนเข้าสู่โปรแกรมจะต้องทำการ login โดยค่าของ Username และ Password ก่อนการเปลี่ยนแปลงจะเป็น Admin



รูปที่ 7-3 แสดงส่วนของหน้าจอควบคุมหลัก

ส่วนของโปรแกรมหน้านี้จะมี

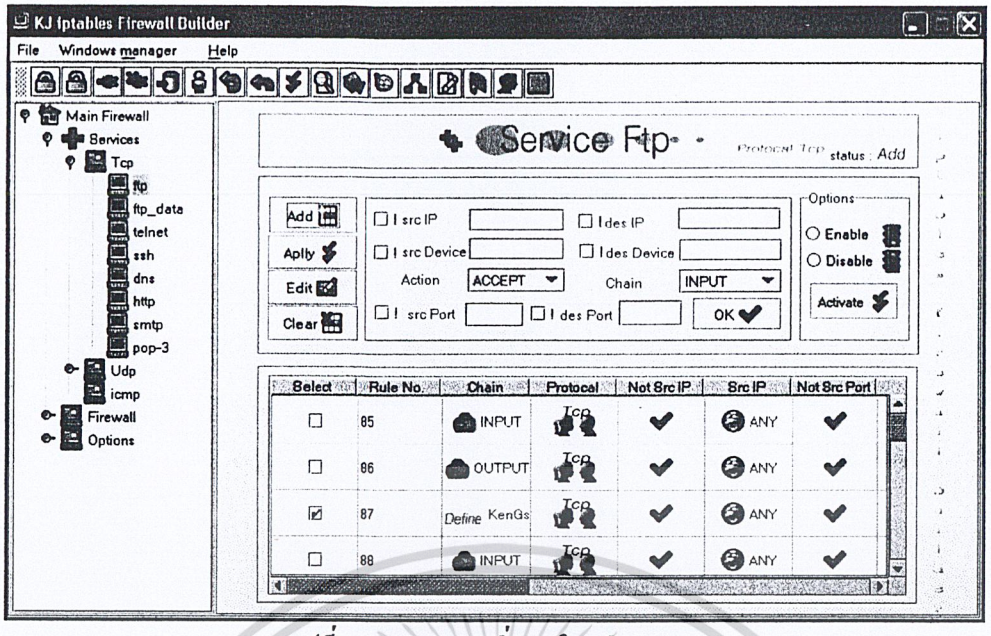
- Add ซึ่งจะการเพิ่มกฎ (rule)
- Edit จะเป็นการแก้ไขกฎ
- Delete จะเป็นการลบกฎเฉพาะในส่วนที่ได้เลือก
- Clear All จะเป็นการลบกฎทั้งหมดออกการโปรแกรม
- Refresh จะเป็นการรีเฟรชกฎที่เราได้เพิ่ม, ลบ, แก้ไขให้ปรากฏในหน้าจอ
- OK จะเป็นการยืนยันที่จะใช้กฎนั้นๆ ที่เราได้เพิ่ม, ลบ, แก้ไขหรือไม่ ก่อนออกโปรแกรม
- Cancel จะเป็นการออกจากโปรแกรมเหมือนปุ่ม Exit แต่จะตรวจสอบว่าได้มีการแก้ไขกฎหรือไม่ ถ้ามีจะมีหน้าจอเตือนเราให้ยืนยันจะใช้กฎนั้น ๆ หรือไม่
- Apply จะเป็นการยืนยันที่จะใช้กฎนั้นๆ ที่เราได้เพิ่ม, ลบ, แก้ไขหรือไม่
- Reset จะเป็นการย้อนกลับไปใช้กฎเดิมก่อนที่จะมีการยืนยันใช้กฎนั้น ๆ ด้วยการใช้ปุ่ม Apply
- Exit ออกจากโปรแกรมทันที่โดยไม่มีข้อความยืนยันใด ๆ เลย



รูปที่ 7-4 แสดงส่วนของการอัพเดทกฎ

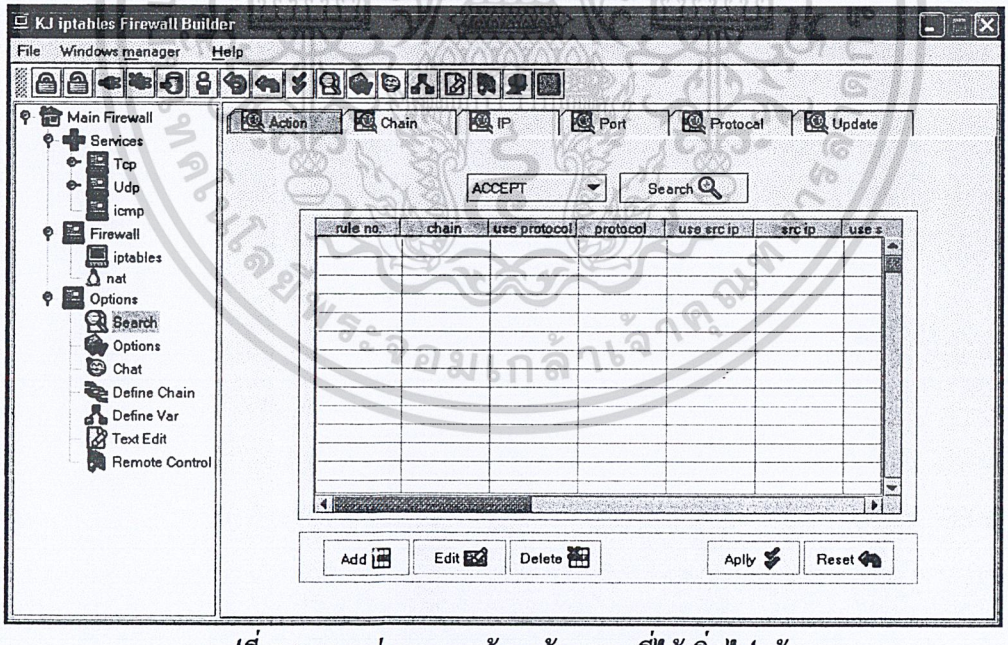
ในการเพิ่มกฎนั้นจะประกอบไปด้วยฟิลด์ต่าง ๆ ดังนี้

- Chain จะเป็นการบอกว่าในกฎเราจะใช้เชนลักษณะเช่นใด เช่น INPUT, OUTPUT, FORWARD หรือ เชนที่เรากำหนดเองขึ้นมา
- src IP หมายถึง IP ต้นทางของแพ็กเกจ
- src Port หมายถึง Port ต้นทาง
- src Device หมายถึง Device ต้นทาง
- des IP หมายถึง IP ปลายทาง
- des Port หมายถึง Port ปลายทาง
- des Device หมายถึง Device ปลายทาง
- ! ในแต่ละตัวถ้ามีการเช็ค (Check) หมายถึง ทุกตัวยกเว้นตัวนั้น
- Protocol โปรโตคอลของแพ็กเกจ
- Action รูปแบบการกระทำเมื่อมีแพ็จเกจนั้นเข้ามา
- OK เป็นการยืนยันที่จะใช้กฎนั้น
- Cancel เป็นการกลับไปหน้าหลักโดยไม่มีการเปลี่ยนแปลงกฎเลย
- Reset เป็นการเอารีเซทกฎนั้นก่อนการเปลี่ยนแปลง



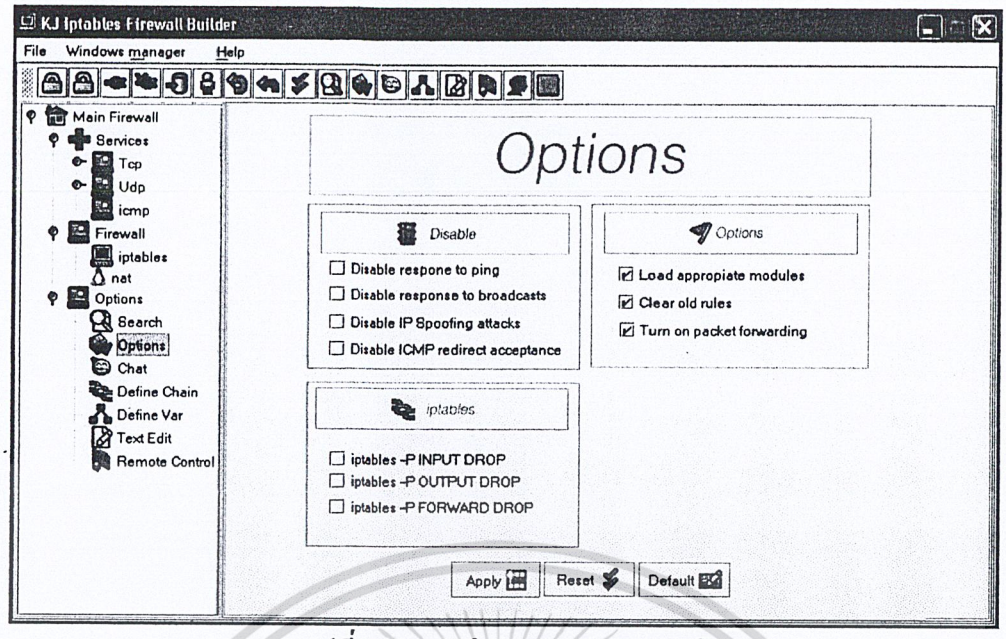
รูปที่ 7-5 แสดงการเพิ่มกฎในบริการ Ftp

ส่วนของ Service จะประกอบไปด้วยโปรโตคอล TCP, UDP, ICMP โดยโปรโตคอล TCP, UDP จะแบ่งออกเป็น Ftp, Ftpdata, Telenet, Ssh, Dns, Http, Smtp, Pop3 โดยแต่ละบริการจะมีลักษณะนี้ดังรูป



รูปที่ 7-6 แสดงส่วนของหน้าจอค้นหากฎที่ได้เพิ่มไปแล้ว

ส่วนของการค้นการค้นหา (Search) โดยจะเป็นการค้นหากฎ มีทั้งการค้นหา Action, Chain, IP Address, Port, Protocol นอกจากนั้นแล้วยังสามารถเพิ่มกฎต่าง ๆ ได้จากหน้าจอนี้อีกด้วย



รูปที่ 7-7 แสดงส่วนของ Options ของโปรแกรม

ส่วนของ Options จะมีออปชั่นให้เลือกเพื่อปิดบริการหรือเลือกใช้บริการดังต่อไปนี้

ส่วนของ Disable

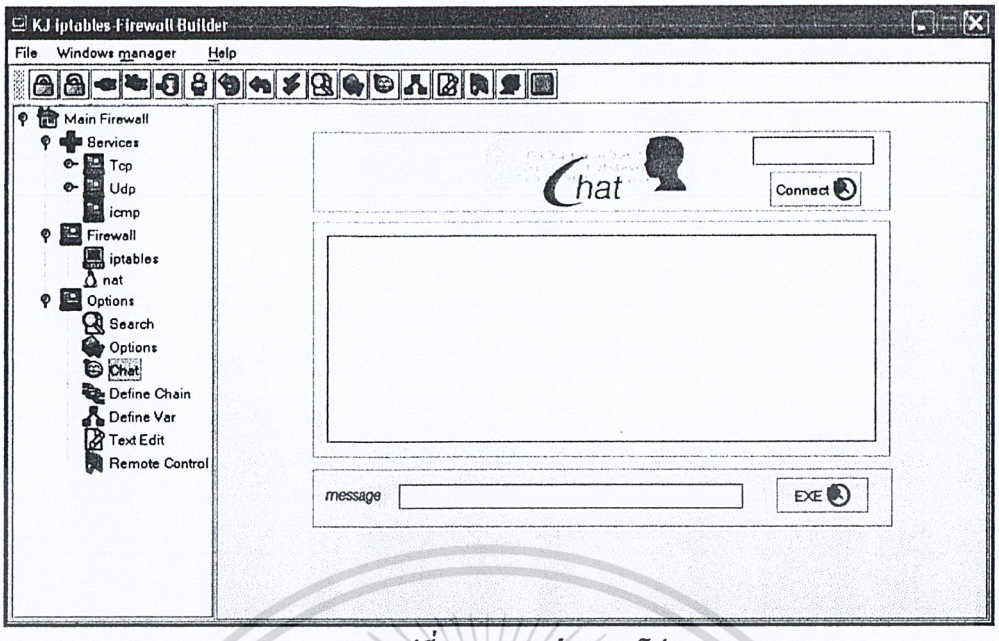
- Disable respone to ping จะเป็นการปิดบริการตอบรับ (Reply) การ ping ที่เข้ามายังเครื่อง
- Disable respone to broadcasts ปิดบริการการ broadcasts
- Disable IP Spoofing attacks ปิดบริการการโจมตีแบบ IP Spoofing
- Disable ICMP redirect acceptance ปิดบริการ ICMP redirect

ส่วนของ iptables

- iptables -P INPUT DROP ดร็อปแพ็จเกจที่เข้ามายังเครื่อง
- iptables –P OUTPUT DROP ดร็อปแพ็จเกจที่ออกจากเครื่อง
- iptables –P FORWARD DROP ดร็อปแพ็จเกจที่จะส่งต่อไป

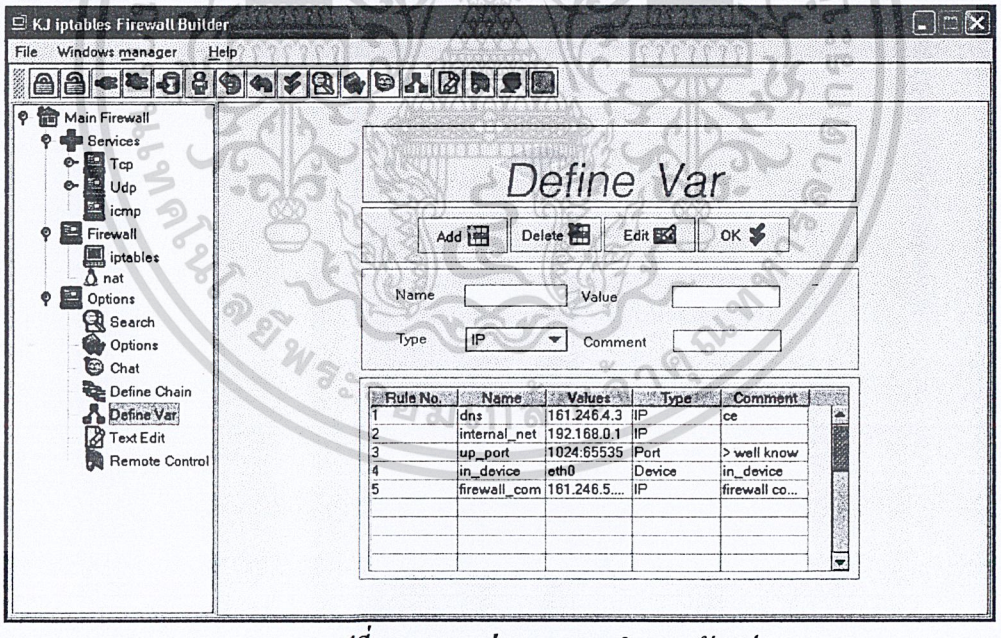
ส่วนของ Options

- Load appropiate module จะเป็นการโหลดโมดูลที่จำเป็นต่อการใช้ iptables
- Clear Old rule จะเป็นการ Clear rule เก่าออกทั้งหมด
- Turn on package forwarding จะเป็นการเปิดให้สามารถส่งต่อ แพ็จเกจได้



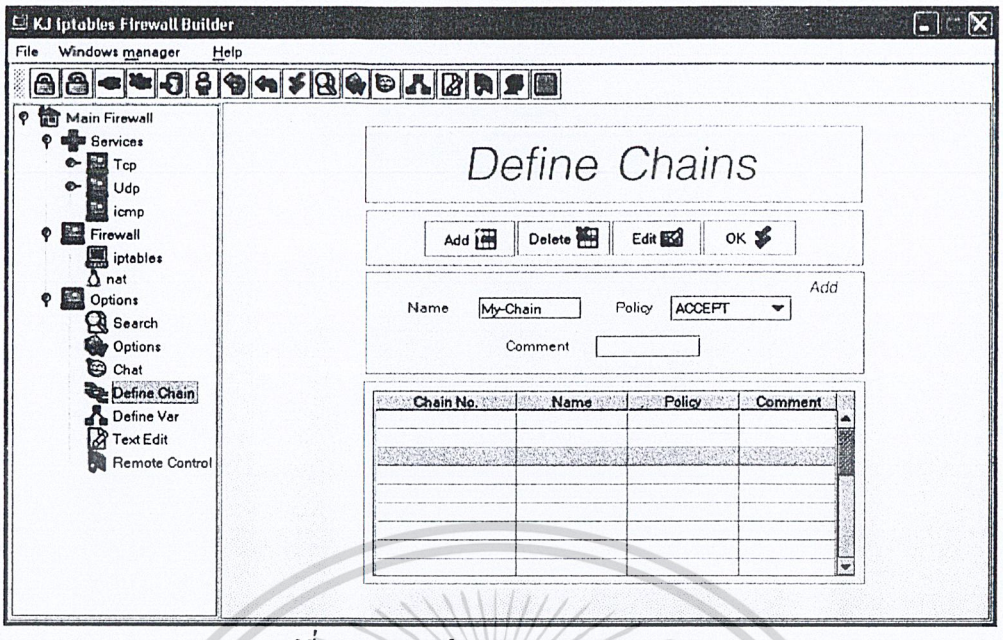
รูปที่ 7-8 แสดงส่วนของโปรแกรม Chat

เป็นส่วนของ Chat ตรงนี้โปรแกรมจะติดต่อไปยังโปรแกรมเซิร์ฟเวอร์ที่อยู่อีกฝั่งหนึ่งซึ่งจะสามารถพูดคุยติดต่อกันได้และข้อความที่คุยกันนั้นจะมีการเข้ารหัส



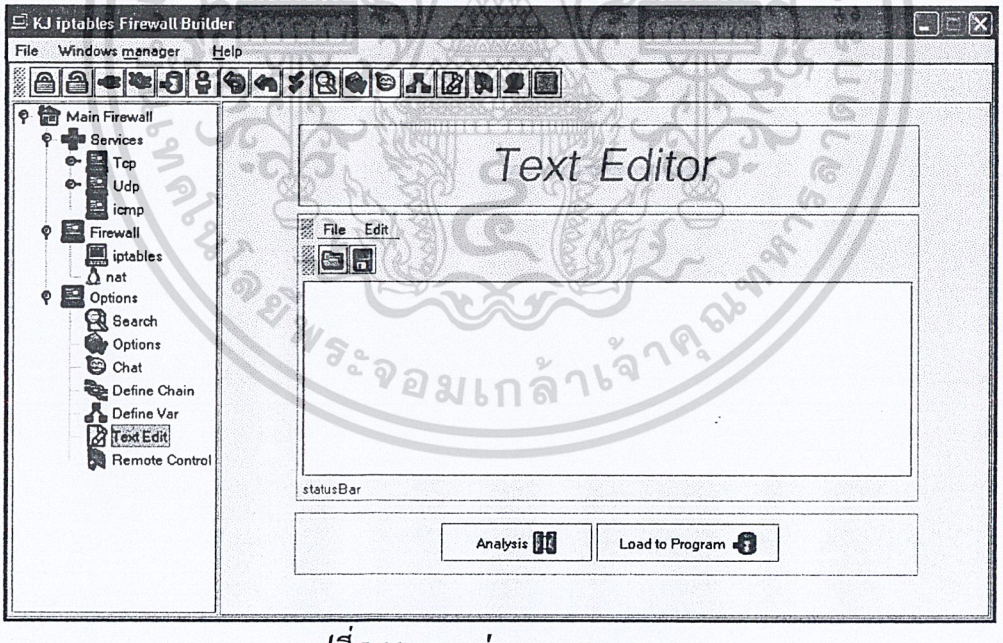
รูปที่ 7-9 แสดงส่วนของการกำหนดตัวแปร

ส่วนของการกำหนดตัวแปร (Define Var) ในโปรแกรมสามารถที่จะกำหนดตัวแปรได้ เพื่อความสะดวกในการจำจำเช่น อาจจะกำหนดว่า ตัวแปรชื่อ Dns มีค่าเป็น 161.246.4.3 ซึ่งจะทำให้สะดวกในการใช้งานโปรแกรม หรืออาจจะกำหนดชื่อ port ที่เราใช้อยู่ประกำเช่น My-Port เป็น 5555 เป็นต้น



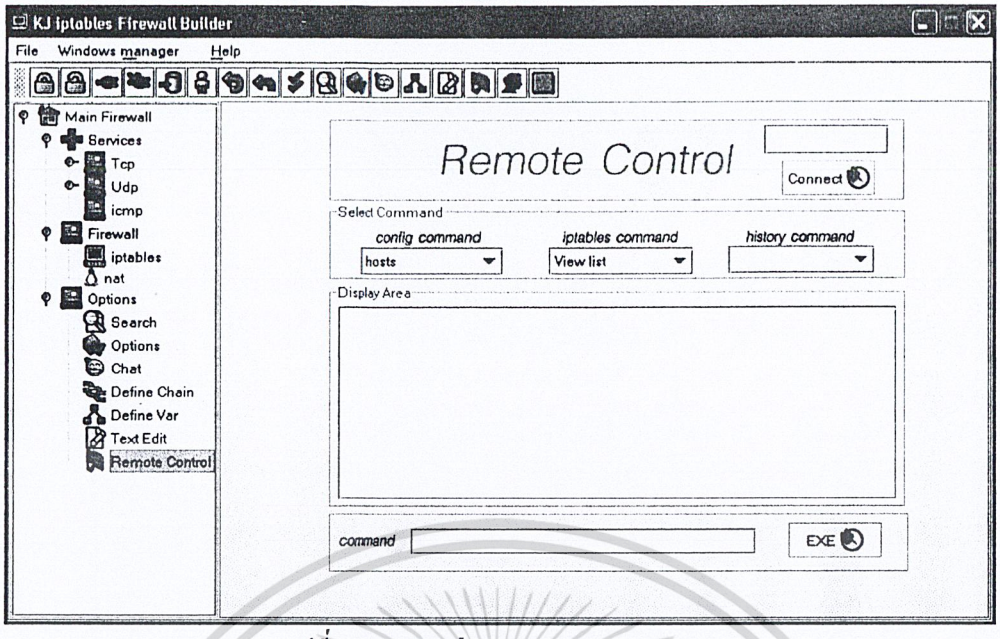
รูปที่ 7-10 แสดงส่วนของการ Define Chains

ส่วนของการกำหนดเชน (Define Chains) สิ่งที่กำหนดตรงนี้จะไปปรากฏอยู่ในส่วนของ Chain และ Action



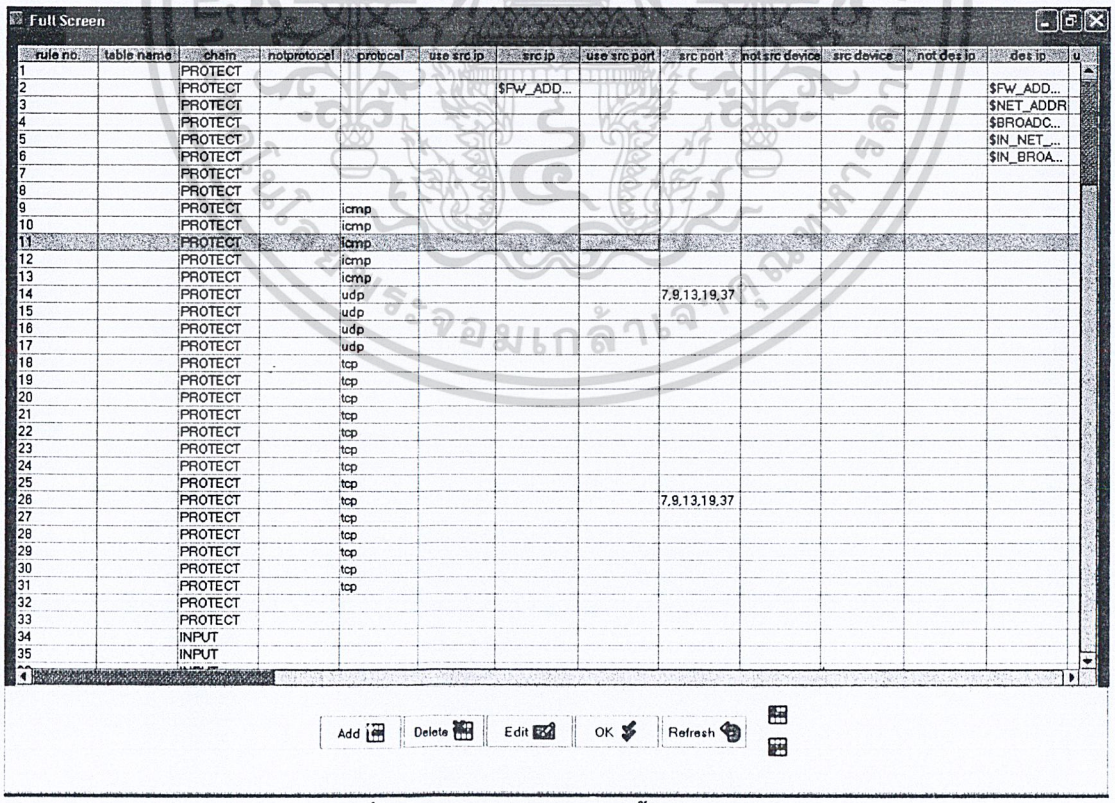
รูปที่ 7-11 แสดงส่วนของ Text Editor

ส่วนของ Text Editor ตรงนี้จะเป็นการนำไฟล์ที่เขียนกฎเอาไว้แล้วตามสคริปของ iptables แล้วสามารถโหลดเข้าสู่โปรแกรมได้และยังติดต่อกับโปรแกรมในส่วนของโปรแกรมเซิร์ฟเวอร์เพื่อตรวจสอบความถูกต้องซึ่งถ้าสคริปนั้นผิดก็จะส่งข้อความที่ผิดนั้นกลับมาที่โปรแกรมไคลเอ็นท์



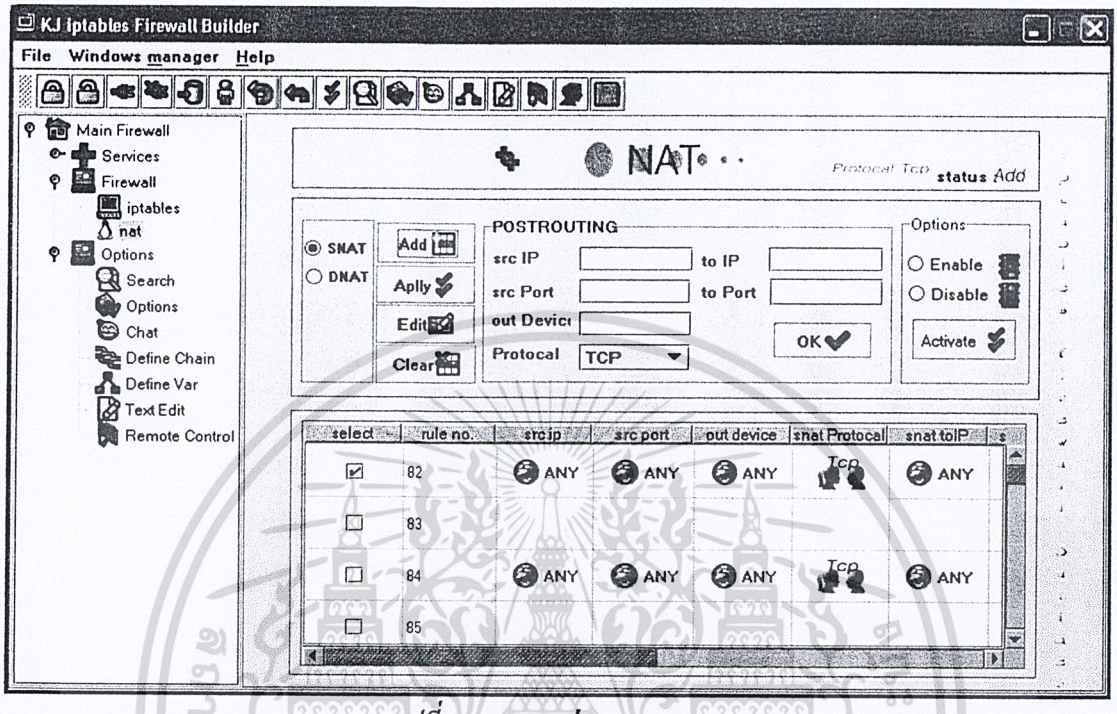
รูปที่ 7-12 แสดงส่วนของ Remote Control

ส่วนของ Remote Control จะเป็นส่วนที่สามารถความไปยังเครื่องไฟล์วอลล์ได้ เช่น การดูรายละเอียดของไฟล์, การดูรายละเอียดของ iptables ที่เราตั้งไว้, หรือการดู configuration ของเครื่องไฟล์วอลล์ เป็นต้น



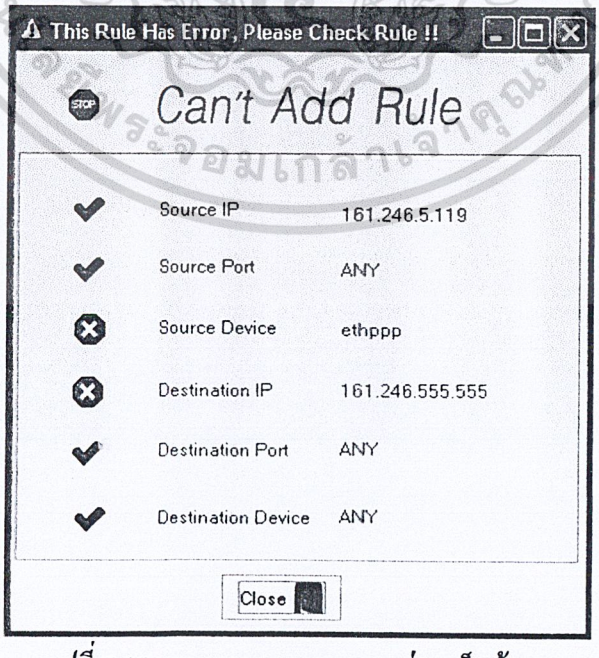
รูปที่ 7-13 แสดงภาพโดยรวมทั้งหมดของกฎ

ส่วนนี้จะเป็นส่วนที่เอาไว้ดูรายละเอียดทั้งหมดของกฎที่ตั้งขึ้น เนื่องจาก iptables มีแท็กที่ต้องใช้เป็นจำนวนมากซึ่ง จะเป็นการยุ่งยากมากเกินไปถ้า โปรแกรมคอนฟิกจะมีฟิลด์ทั้งหมด ส่วนนี้จึงเป็นส่วนที่มาจากการคอนฟิกโดย Text Editor



รูปที่ 7-14 แสดงส่วนของ NAT

ส่วนของการคอนฟิก NAT ประกอบไปด้วย SNAT, DNAT



รูปที่ 7-15 แสดงการตรวจสอบกฎก่อนเก็บข้อมูล

ทุกครั้งที่มีการเพิ่มหรือแก้ไขจะมีการตรวจสอบความถูกต้องของกฎเสมอ

7.3 ส่วนของโปรแกรมเซิร์ฟเวอร์

ในโปรแกรมเซิร์ฟเวอร์นั้นจะคอยรับบริการการร้องจากโปรแกรมเซิร์ฟเวอร์ ซึ่งโปรแกรมเซิร์ฟเวอร์นี้จะเป็นตัวที่คอยไปควบคุยเครื่องที่ไฟร์วออล์อีกต่อหนึ่ง การติดต่อกับโปรแกรมไคล์เอนท์นั้นจะเป็นไปได้ทั้ง การติดต่อเพื่อรับสคริปท์ การติดต่อการคุยกัน หรือการติดต่อแบบรีโมทเพื่อดูรายละเอียดข้อมูลต่างๆ ของเครื่องไฟร์วอลล์ต้น

รูปที่ 7-16 แสงส่วนของโปรแกรมเซิร์ฟเวอร์

7.4 ส่วนของตัวแปลงกฎเป็น XML

ส่วนนี้จะรวมอยู่ในส่วนของการโปรแกรมไคล์เอนท์ โดยจะมีหน้าที่ในการแปลงกฎที่เราได้คอนฟิกไว้ไปเป็น XMLเพื่อที่จะส่งต่อไปให้โปรแกรมเซิร์เวอร์ทำการแปลงเป็นสคริปท์ของ iptables

- **รูปแบบของ DTD ที่นำมาใช้ในโปรแกรม**

```
<!DOCTYPE FIREWALL_CONFIG [
<!ELEMENT FIREWALL_CONFIG
(VAR_LIST*,CHAIN_LIST*,BLACK_LIST*,CONFIG_CHAIN*)>
<!ELEMENT VAR_LIST (VAR*)>
<!ELEMENT VAR (VALUE,COMMENT)>
```

```
<!ELEMENT VALUE (#PCDATA)>
<!ELEMENT COMMENT (#PCDATA)>
<!ELEMENT CHAIN_LIST (CHAIN*)>
<!ELEMENT CHAIN (POLICY,COMMENT)>
<!ELEMENT POLICY (#PCDATA)>
<!ELEMENT BLACK_LIST (BL*)>
<!ELEMENT BL (ADDR,STATUS)>
<!ELEMENT ADDR (#PCDATA)>
<!ELEMENT STATUS (#PCDATA)>
<!ELEMENT CONFIG_CHAIN (RULE*)>
<!ELEMENT RULE
(TABLE,CHAIN_NAME,PROTOCOL,SRC,DES,IN_IF,OUT_IF,FRAGMENT,MATCH,MARK,UID
_OWNER,GID_OWNER,PID_OWNER,SID_OWNER,JUMP_TARGET,TARGET_EXT)>
<!ELEMENT TABLE (#PCDATA)>
<!ELEMENT CHAIN_NAME (#PCDATA)>
<!ELEMENT PROTOCOL
(NOT,VALUE,SRC_PORT,DES_PORT,TCP_FLAG,SYN,TCP_OP,ICMP_TYPE)>
<!ELEMENT NOT (#PCDATA)>
<!ELEMENT SRC_PORT (NOT,VALUE)>
<!ELEMENT DES_PORT (NOT,VALUE)>
<!ELEMENT TCP_FLAG (MASK,COMP)>
<!ELEMENT MASK (NOT,VALUE)>
<!ELEMENT SYN (NOT,VALUE)>
<!ELEMENT TCP_OP (NOT,VALUE)>
<!ELEMENT ICMP_TYPE (NOT,VALUE)>
<!ELEMENT SRC (NOT,VALUE)>
<!ELEMENT DES (NOT,VALUE)>
<!ELEMENT IN_IF (NOT,VALUE)>
<!ELEMENT OUT_IF (NOT,VALUE)>
<!ELEMENT FRAGMENT (NOT,VALUE)>
<!ELEMENT MATCH (MAC,LIMIT,MULTIPORT,STATE,UNCLEAN,TOS)>
<!ELEMENT MAC (VALUE, SOURCE)>
<!ELEMENT SOURCE (NOT ,VALUE)>
<!ELEMENT LIMIT (VALUE, RATE, BURST)>
```

<!ELEMENT RATE (#PCDATA)>

<!ELEMENT BURST (#PCDATA)>

<!ELEMENT MULTIPORT (VALUE, SPORT, DPORT, PORT)>

<!ELEMENT SPORT (#PCDATA)>

<!ELEMENT DPORT (#PCDATA)>

<!ELEMENT PORT (#PCDATA)>

<!ELEMENT STATE (VALUE, STATE_LIST)>

<!ELEMENT STATE_LIST (#PCDATA)>

<!ELEMENT UNCLEAN (#PCDATA)>

<!ELEMENT TOS (VALUE,TYPE)>

<!ELEMENT TYPE (#PCDATA)>

<!ELEMENT MARK (#PCDATA)>

<!ELEMENT UID_OWNER (#PCDATA)>

<!ELEMENT GID_OWNER (#PCDATA)>

<!ELEMENT PID_OWNER (#PCDATA)>

<!ELEMENT SID_OWNER (#PCDATA)>

<!ELEMENT JUMP_TARGET (#PCDATA)>

<!ELEMENT TARGET_EXT
(LOG_LEVEL,LOG_PREFIX,LOG_TCP_SEQ,LOG_TCP_OPT,LOG_IP_OPT,SET_MARK,REJECT
_WITH,SET_TOS,TO,TO_SRC,TO_DES,TO_PORT)>

<!ELEMENT LOG_LEVEL (#PCDATA)>

<!ELEMENT LOG_PREFIX (#PCDATA)>

<!ELEMENT LOG_TCP_SEQ (#PCDATA)>

<!ELEMENT LOG_TCP_OPT (#PCDATA)>

<!ELEMENT LOG_IP_OPT (#PCDATA)>

<!ELEMENT SET_MARK (#PCDATA)>

<!ELEMENT REJECT_WITH (#PCDATA)>

<!ELEMENT SET_TOS (#PCDATA)>

<!ELEMENT TO (#PCDATA)>

<!ELEMENT TO_SRC (#PCDATA)>

<!ELEMENT TO_DES (#PCDATA)>

<!ELEMENT TO_PORT (#PCDATA)>

<!ATTLIST BL

number CDATA #IMPLIED>

```
<!ATTLIST RULE
        n CDATA #IMPLIED>
]>
```

- **ELEMENT ที่ใช้ในโปรแกรม**

```
<FIREWALL_CONFIG>
<VAR_LIST>
<VAR name="">
<VALUE></VALUE>
<COMMENT></COMMENT>
</VAR>
<VAR name="">
<VALUE></VALUE>
<COMMENT></COMMENT>
</VAR>
</VAR_LIST>
<CHAIN_LIST>
<CHAIN name="">
<POLICY></POLICY>
<COMMENT></COMMENT>
</CHAIN>
</CHAIN_LIST>
<BLACK_LIST>
<BL number="">
<ADDR></ADDR>
<STATUS></STATUS>
</BL>
<CONFIG_CHAIN>
<RULE n="0">
<TABLE></TABLE>
<CHAIN_NAME></CHAIN_NAME>
<PROTOCOL>
<NOT></NOT>
<VALUE></VALUE>
```

<SRC_PORT>

<NOT></NOT>

<VALUE></VALUE>

</SRC_PORT>

<DES_PORT>

<NOT></NOT>

<VALUE></VALUE>

</DES_PORT>

<TCP_FLAG>

<MASK>

<NOT></NOT>

<VALUE></VALUE>

</MASK>

<COMP></COMP>

</TCP_FLAG>

<SYN>

<NOT></NOT>

<VALUE></VALUE>

</SYN>

<TCP_OP>

<NOT></NOT>

<VALUE></VALUE>

</TCP_OP>

<ICMP_TYPE>

<NOT></NOT>

<VALUE></VALUE>

</ICMP_TYPE>

</PROTOCOL>

<SRC>

<NOT></NOT>

<VALUE></VALUE>

</SRC>

<DES>

<NOT></NOT>

<VALUE></VALUE>

</DES>

<IN_IF>

<NOT></NOT>

<VALUE></VALUE>

</IN_IF>

<OUT_IF>

<NOT></NOT>

<VALUE></VALUE>

</OUT_IF>

<FRAGMENT>

<NOT></NOT>

<VALUE></VALUE>

</FRAGMENT>

<MATCH>

<MAC>

<VALUE></VALUE>

<SOURCE>

<NOT></NOT>

<VALUE></VALUE>

</SOURCE>

</MAC>

<LIMIT>

<VALUE></VALUE>

<RATE></RATE>

<BURST></BURST>

</LIMIT>

<MULTIPORT>

<VALUE></VALUE>

<SPORT></SPORT>

<DPORT></DPORT>

<PORT></PORT>

</MULTIPORT>

<STATE>

<VALUE></VALUE>

<STATE_LIST></STATE_LIST>

</STATE>

<UNCLEAN></UNCLEAN>

<TOS>

<VALUE></VALUE>

<TYPE></TYPE>

</TOS>

</MATCH>

<MARK></MARK>

<UID_OWNER></UID_OWNER>

<GID_OWNER></GID_OWNER>

<PID_OWNER></PID_OWNER>

<SID_OWNER></SID_OWNER>

<JUMP_TARGET></JUMP_TARGET>

<TARGET_EXT>

<LOG_LEVEL></LOG_LEVEL>

<LOG_PREFIX></LOG_PREFIX>

<LOG_TCP_SEQ></LOG_TCP_SEQ>

<LOG_TCP_OPT></LOG_TCP_OPT>

<LOG_IP_OPT></LOG_IP_OPT>

<SET_MARK></SET_MARK>

<REJECT_WITH></REJECT_WITH>

<SET_TOS></SET_TOS>

<TO></TO>

<TO_SRC></TO_SRC>

<TO_DES></TO_DES>

<TO_PORT></TO_PORT>

</TARGET_EXT>

</RULE>

</CONFIG_CHAIN>

</FIREWALL_CONFIG>

ในแต่ละ Element นั้นมีความหมายดังต่อไปนี้

ชื่อ Tag	รายละเอียด
<FIREWALL_CONFIG>	เป็นส่วน root ของ XML File
<VAR_LIST>	เป็นเริ่มต้นของแท็ก Variable List
<VAR name="">	เป็นชื่อของตัวแปรที่เราตั้งขึ้น
<VALUE></VALUE>	เป็นค่าของตัวแปร
<COMMENT></COMMENT>	เป็น comment ของตัวแปรที่ตั้งขึ้น
</VAR>	เป็นส่วนปิดของแท็ก Var
</VAR_LIST>	เป็นปิดของแท็ก Variable List
<CHAIN_LIST>	เป็นส่วนเริ่มต้นของ Chain List
<CHAIN name="">	เป็นชื่อของ Chain
<POLICY></POLICY>	เป็นการระบุ Policy
<COMMENT></COMMENT>	เป็นการระบุ Comment ของ Policy
</CHAIN>	เป็นส่วนปิดท้ายของแท็ก Chain
</CHAIN_LIST>	เป็นส่วนปิดท้ายของ Chain List
<BLACK_LIST>	เป็นส่วนเริ่มต้นของ Black List
<BL number="">	เป็นการระบุ BlaclList Number
<ADDR></ADDR>	ระบุค่า Address
<STATUS></STATUS>	ระบุค่า Status
</BL>	เป็นแท็กปิดท้ายของ BlackList Number
<CONFIG_CHAIN>	ส่วนเริ่มต้นของแท็ก Config
<RULE n="0">	ระบุ Rule No.
<TABLE></TABLE>	ใช้สำหรับระบุ packet matching table เช่น nat
<CHAIN_NAME></CHAIN_NAME>	เป็นส่วนที่บอกชื่อของ Chain
<PROTOCOL>	เป็นส่วนเริ่มต้นของแท็กโปรโตคอล
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุโพรโตคอลของแพ็กเก็ต โดยสามารถระบุได้ 4 แบบ คือ *tcp*, *udp*, *icmp*
<SRC_PORT>	เป็นส่วนเริ่มต้นของแท็ก
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุพอร์ตต้นทางของแพ็กเก็ต สามารถระบุเป็นชื่อบริการ
</SRC_PORT>	เป็นส่วนปิดท้ายของแท็ก Source Port

<DES_PORT>	เป็นส่วนเริ่มต้นของแท็ก Destination Port
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุพอร์ตปลายทางของแพ็กเก็ต สามารถระบุเป็นชื่อบริการ
</DES_PORT>	เป็นส่วนปิดท้ายของแท็กDestination Port
<TCP_FLAG>	เป็นส่วนเริ่มต้นของแท็ก TCP Fragment
<MASK>	เป็นส่วนเริ่มต้นของแท็ก MASK
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	ระบุแฟลกของแพ็กเก็ตที่ต้องการตรวจสอบ
</MASK>	เป็นส่วนปิดท้ายของแท็ก MASK
<COMP></COMP>	ระบุแฟลกของแพ็กเก็ตที่ต้องการตรวจสอบ
</TCP_FLAG>	เป็นส่วนปิดท้ายของแท็ก TCP Flag
<SYN>	เป็นส่วนเริ่มต้นของแท็ก SYN
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุแพ็กเก็ตที่การร้องขอ TCP connection ซึ่งจะเป็นแพ็กเก็ตที่ SYN flag ถูกเซต และ ACK flag กับ FIN flag ถูกเคลียร์
</SYN>	เป็นส่วนปิดท้ายของแท็ก SYN
<TCP_OP>	เป็นส่วนเริ่มต้นของแท็ก TCP_OP
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุแพ็กเก็ตที่มีการเซตค่า TCP option
</TCP_OP>	เป็นส่วนปิดท้ายของแท็ก TCP Option
<ICMP_TYPE>	เป็นส่วนเริ่มต้นของแท็ก PROTOCAL
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุแพ็กเก็ตที่มี ICMP type ตรงตาม type ที่กำหนด ซึ่งสามารถกำหนดเป็นหมายเลขหรือชื่อ type ก็ได้
</ICMP_TYPE>	เป็นส่วนปิดของแท็ก ICMP_TYPE
</PROTOCOL>	เป็นส่วนปิดของแท็ก PROTOCAL
<SRC>	เป็นส่วนเริ่มต้นของแท็ก Source Address
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	ระบุค่า Address
</SRC>	เป็นส่วนปิดของแท็ก Source Address
<DES>	เป็นส่วนเริ่มต้นของแท็ก Destination Address

<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	ระบุค่า Address
</DES>	เป็นส่วนปิดของแท็ก Destination Address
<IN_IF>	เป็นส่วนเริ่มต้นของแท็ก IN_IF
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุการ์ดอินเทอร์เฟสที่แพ็กเก็ตผ่านเข้ามา โดยที่ option นี้สามารถใช้กับ INPUT, FORWARD และ PREROUTING chain ได้เท่านั้น
</IN_IF>	เป็นส่วนปิดของแท็ก IN_IF
<OUT_IF>	เป็นส่วนเริ่มต้นของแท็ก OUT_IF
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุการ์ดอินเทอร์เฟสที่แพ็กเก็ตผ่านออกไป โดยที่ option นี้สามารถใช้กับ FORWARD, OUTPUT และ POSTROUTING chain ได้เท่านั้น
</OUT_IF>	เป็นส่วนปิดของแท็ก OUT_IF
<FRAGMENT>	เป็นส่วนเริ่มต้นของแท็ก FRAGMENT
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	ระบุว่าเป็นกฎที่ใช้สำหรับแฟรกเมนต์
</FRAGMENT>	เป็นส่วนปิดของแท็ก FRAGMENT
<MATCH>	เป็นส่วนเริ่มต้นของแท็ก MATCH
<MAC>	เป็นส่วนเริ่มต้นของแท็ก MAC
<VALUE></VALUE>	สำหรับระบุแมคแอดเดรสของอีเธอร์เน็ตดีไวซ์ต้นทางของแพ็กเก็ต ซึ่งจะอยู่ในรูปแบบ XX:XX:XX:XX:XX:XX โดยการระบุในแบบนี้ จะใช้ได้กับ INPUT, FORWARD หรือ PREROUTING chain เท่านั้น
<SOURCE>	เป็นส่วนเริ่มต้นของแท็ก SOURCE
<NOT></NOT>	สามารถใช้ "!" เพื่อให้มีความหมายในทางตรงกันข้ามได้
<VALUE></VALUE>	สำหรับระบุค่า Address
</SOURCE>	เป็นส่วนปิดของแท็ก SOURCE
</MAC>	เป็นส่วนปิดของแท็ก MAC
<LIMIT>	เป็นส่วนเริ่มต้นของแท็ก LIMIT
<VALUE></VALUE>	สำหรับระบุค่า limit
<RATE></RATE>	สำหรับค่าเฉลี่ยของจำนวนการแมทช์กับกฎนั้นๆ ต่อช่วงเวลา

	หนึ่งๆ โดยสามารถเลือกช่วงเวลาได้หลายแบบ ดังนี้ *'/second'*, *'/minute'*, *'/hour'* หรือ *'/day'* โดยค่า default คือ *3/hour*
<BURST></BURST>	สำหรับระบุค่าสูงสุดของจำนวนการแมทช์กับกฎนั้นๆ โดยจะถูกรีชาร์จทีละหนึ่ง ทุกๆ ครั้งที่เรทการแมทช์กันของกฎนั้นๆ น้อยกว่าที่ระบุไว้โดย **–limit** ซึ่งค่า default คือ *5*
</LIMIT>	เป็นส่วนปิดของแท็ก
<MULTIPORT>	เป็นส่วนเริ่มต้นของแท็ก MULTIPORT
<VALUE></VALUE>	สำหรับระบุพอร์ตหลายๆ พอร์ตในกฎ ซึ่งสามารถระบุถึง 15 พอร์ตต่อหนึ่งกฎ แต่สามารถใช้งานได้กับโพรโตคอล TCP และ UDP เท่านั้น
<SPORT></SPORT>	สำหรับระบุพอร์ตต้นทางของแพ็กเก็ต
<DPORT></DPORT>	สำหรับระบุพอร์ตปลายทางของแพ็กเก็ต
<PORT></PORT>	สำหรับระบุพอร์ตต้นทางและปลายทางของแพ็กเก็ต
</MULTIPORT>	เป็นส่วนปิดของแท็ก MULTIPORT
<STATE>	เป็นส่วนเริ่มต้นของแท็ก STATE
<VALUE></VALUE>	สำหรับระบุสถานะการเชื่อมต่อของแพ็กเก็ต โดยสามารถระบุเป็นชุดได้โดยใช้ ',' คั่น
<STATE_LIST></STATE_LIST>	สามารถเลือกระบุสถานะการเชื่อมต่อของแพ็กเก็ต
</STATE>	
<UNCLEAN></UNCLEAN>	โมดูลนี้ ไม่มี option ใดๆ เพียงใช้สำหรับระบุแพ็กเก็ตที่ดูแปลก ไม่ปกติ โดยพิจารณาจากผลที่ได้จากการทดสอบ
<TOS>	เป็นส่วนเริ่มต้นของ TOS
<VALUE></VALUE>	สำหรับระบุค่าเป็น TOS
<TYPE></TYPE>	สำหรับเซตค่าให้กับส่วน Type Of Service (8-bit) โดยจะเซตเป็นตัวเลขหรือชื่อของ TOS ก็ได้ และใช้ได้เฉพาะกับ mangle table เท่านั้น
</TOS>	เป็นส่วนปิดของแท็ก TOS
</MATCH>	เป็นส่วนปิดของแท็ก MATCH
<MARK></MARK>	สำหรับเซตค่า *netfilter mark* ที่เกี่ยวข้องกับแพ็กเก็ต ใช้ได้เฉพาะกับ mangle table เท่านั้น
<UID_OWNER></UID_OWNER>	ระบุค่าของ uid owner
<GID_OWNER></GID_OWNER>	ระบุค่าของ gid owner

<PID_OWNER></PID_OWNER>	ระบุค่าของ pid owner
<SID_OWNER></SID_OWNER>	ระบุค่าของ sid owner
<JUMP_TARGET></JUMP_TARGET>	สำหรับระบุ target ของแพ็กเก็ต คือ ควรจะทำอย่างไรเวลาที่รายละเอียดของแพ็กเก็ตตรงกับกฎที่กำหนดไว้
<TARGET_EXT>	เป็นแท็กเริ่มต้นของ TARGET EXT
<LOG_LEVEL></LOG_LEVEL>	บันทึก log เฉพาะ level ที่กำหนดเท่านั้น
<LOG_PREFIX></LOG_PREFIX>	ใส่ prefix ที่กำหนดไปที่ log message ด้วย เพื่อให้เข้าใจและสังเกตได้ง่าย โดย prefix สามารถยาวได้ถึง 14 ตัวอักษร
<LOG_TCP_SEQ></LOG_TCP_SEQ>	บันทึก TCP sequence number ด้วย การใช้การบันทึก log แบบนี้ เสี่ยงต่อการเกิดอันตรายทางด้านความปลอดภัยได้ เพราะอาจมีผู้ไม่ประสงค์ดีมาอ่านข้อมูลตรงนี้ได้
<LOG_TCP_OPT></LOG_TCP_OPT>	บันทึก option ที่อยู่ใน TCP header ของแพ็กเก็ตด้วย
<SET_MARK></SET_MARK>	สำหรับเซตค่า *netfilter mark* ที่เกี่ยวข้องกับแพ็กเก็ต ใช้ได้เฉพาะกับ mangle table เท่านั้น
<REJECT_WITH></REJECT_WITH>	สำหรับระบุชนิดของข้อความที่ส่งกลับไปตอน REJECT โดยมีชนิดของข้อความที่สามารถระบุได้ ดังนี้
<SET_TOS></SET_TOS>	สำหรับเซตค่าให้กับส่วน Type Of Service (8-bit) โดยจะเซตเป็นตัวเลขหรือชื่อของ TOS ก็ได้ และใช้ได้เฉพาะกับ mangle table เท่านั้น
<TO></TO>	เป็นส่วนของ nat ที่เก็บ Address
<TO_SRC></TO_SRC>	เป็นส่วนของ nat ที่เก็บ Source Address
<TO_DES></TO_DES>	เป็นส่วนของ nat ที่เก็บ Destination Address
<TO_PORT></TO_PORT>	เป็นส่วนของ nat ที่เก็บ PORT
</TARGET_EXT>	เป็นส่วนปิดของแท็ก TARGET EXT
</RULE>	เป็นส่วนปิดของแท็ก RULE
</CONFIG_CHAIN>	เป็นส่วนปิดของแท็ก CONFIG CHAIN
</FIREWALL_CONFIG>	เป็นส่วนปิดของแท็ก FIREWALL CONFIG

ตารางที่ 7-1 แสดงรายละเอียดแต่ละอิลิเมนต์

แสดงการแสดงผลของไฟล์ XML เมื่อมีการสร้างกฎขึ้นมา

```
<?xml version="1.0" ?>
<!DOCTYPE FIREWALL_CONFIG (View Source for full doctype...)>
- <FIREWALL_CONFIG>
  - <OPTIONS>
      <DISABLE_BROADCASTS>/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts</DISABLE_BROADCASTS>
      <DISABLE_PINGS>/bin/echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_all</DISABLE_PINGS>
      <DISABLE_ICMP_REDIRECT>/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects</DISABLE_ICMP_REDIRECT>
      <ENABLE_PROTECTION>/bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses</ENABLE_PROTECTION>
      <DISABLE_SPOOFING>/bin/echo "1" > /proc/sys/net/ipv4/conf/all/rp_filter</DISABLE_SPOOFING>
      <ENABLE_SYN_COOKIES>/bin/echo "1" > /proc/sys/net/ipv4/tcp_syncookies</ENABLE_SYN_COOKIES>
      <DISABLE_SROUTING>/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route</DISABLE_SROUTING>
      <LOG_MARTIANS>/bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians</LOG_MARTIANS>
    </OPTIONS>
  - <VAR_LIST>
    - <VAR name="dns">
        <VALUE>161.246.4.3</VALUE>
        <COMMENT>ce</COMMENT>
      </VAR>
    - <VAR name="internal_net">
        <VALUE>192.168.0.1</VALUE>
        <COMMENT>null</COMMENT>
      </VAR>
    - <VAR name="up_port">
        <VALUE>1024:65535</VALUE>
        <COMMENT>> well know</COMMENT>
      </VAR>
    - <VAR name="in_device">
        <VALUE>eth0</VALUE>
        <COMMENT>in_device</COMMENT>
      </VAR>
    - <VAR name="firewall_com">
        <VALUE>161.246.5.119</VALUE>
        <COMMENT>firewall computer</COMMENT>
      </VAR>
    </VAR_LIST>
  - <CHAIN_LIST>
    - <CHAIN name="INPUT">
        <POLICY>DROP</POLICY>
        <COMMENT>input-chain</COMMENT>
      </CHAIN>
    - <CHAIN name="OUTPUT">
        <POLICY>ACCEPT</POLICY>
        <COMMENT>output-chain</COMMENT>
```

รูปที่ 7-17 แสดงเอกสาร XML ที่มีการสร้างกฎแล้ว (1)

```
          <NOT>null</NOT>
          <VALUE>icmp</VALUE>
        - <ICMP_TYPE>
            <NOT>null</NOT>
            <VALUE>echo-request</VALUE>
          </ICMP_TYPE>
        </PROTOCOL>
      - <MATCH>
        - <LIMIT>
            <VALUE>limit</VALUE>
            <RATE>10/s</RATE>
            <BURST>15</BURST>
          </LIMIT>
        </MATCH>
        <JUMP_TARGET>ACCEPT</JUMP_TARGET>
        <TARGET_EXT />
      </RULE>
    - <RULE n="4">
        <CHAIN_NAME>INPUT</CHAIN_NAME>
      - <PROTOCOL>
          <NOT>null</NOT>
          <VALUE>icmp</VALUE>
        - <ICMP_TYPE>
            <NOT>null</NOT>
            <VALUE>echo-request</VALUE>
          </ICMP_TYPE>
        </PROTOCOL>
        <JUMP_TARGET>DROP</JUMP_TARGET>
        <TARGET_EXT />
      </RULE>
    - <RULE n="5">
        <CHAIN_NAME>FORWARD</CHAIN_NAME>
      - <PROTOCOL>
          <NOT>null</NOT>
          <VALUE>icmp</VALUE>
        - <ICMP_TYPE>
            <NOT>null</NOT>
            <VALUE>echo-request</VALUE>
          </ICMP_TYPE>
        </PROTOCOL>
      - <MATCH>
        - <LIMIT>
            <VALUE>limit</VALUE>
            <RATE>10/s</RATE>
```

รูปที่ 7-18 แสดงเอกสาร XML ที่มีการสร้างกฎแล้ว (2)

```
      </FRAGMENT>
      <MARK>null</MARK>
      <UID_OWNER>null</UID_OWNER>
      <GID_OWNER>null</GID_OWNER>
      <PID_OWNER>null</PID_OWNER>
      <SID_OWNER>null</SID_OWNER>
      <JUMP_TARGET>SNAT</JUMP_TARGET>
    - <TARGET_EXT>
        <LOG_LEVEL>null</LOG_LEVEL>
        <LOG_PREFIX>null</LOG_PREFIX>
        <LOG_TCP_SEQ>null</LOG_TCP_SEQ>
        <LOG_TCP_OPT>null</LOG_TCP_OPT>
        <LOG_IP_OPT>null</LOG_IP_OPT>
        <SET_MARK>null</SET_MARK>
        <REJECT_WITH>null</REJECT_WITH>
        <SET_TOS>null</SET_TOS>
        <TO>null</TO>
        <TO_DES>null</TO_DES>
        <TO_PORT>null</TO_PORT>
      </TARGET_EXT>
    </RULE>
  - <RULE n="16">
      <CHAIN_NAME>INPUT</CHAIN_NAME>
    - <PROTOCOL>
        <NOT>null</NOT>
        <VALUE>tcp</VALUE>
      - <TCP_FLAG>
        - <MASK>
            <NOT>null</NOT>
            <VALUE>null</VALUE>
          </MASK>
          <COMP>null</COMP>
        </TCP_FLAG>
      - <SYN>
          <NOT>null</NOT>
          <VALUE>null</VALUE>
        </SYN>
      - <TCP_OP>
          <NOT>null</NOT>
          <VALUE>null</VALUE>
        </TCP_OP>
      - <ICMP_TYPE>
          <NOT>null</NOT>
          <VALUE>null</VALUE>
```

รูปที่ 7-19 แสดงเอกสาร XML ที่มีการสร้างกฎแล้ว (3)

```
        <VALUE>$IFACE</VALUE>
      </IN_IF>
      <JUMP_TARGET>syn-flood</JUMP_TARGET>
      <TARGET_EXT />
    </RULE>
  - <RULE n="29">
      <CHAIN_NAME>syn-flood</CHAIN_NAME>
    - <MATCH>
      - <LIMIT>
          <VALUE>limit</VALUE>
          <RATE>1/s</RATE>
          <BURST>4</BURST>
        </LIMIT>
      </MATCH>
      <JUMP_TARGET>RETURN</JUMP_TARGET>
      <TARGET_EXT />
    </RULE>
  - <RULE n="70">
      <CHAIN_NAME>INPUT</CHAIN_NAME>
    - <PROTOCOL>
        <NOT>null</NOT>
        <VALUE>icmp</VALUE>
      </PROTOCOL>
    - <IN_IF>
        <NOT>null</NOT>
        <VALUE>$IFACE</VALUE>
      </IN_IF>
      <JUMP_TARGET>LOG</JUMP_TARGET>
    - <TARGET_EXT>
        <LOG_PREFIX>"IPTABLES ICMP-IN: "</LOG_PREFIX>
      </TARGET_EXT>
    </RULE>
  - <RULE n="81">
      <CHAIN_NAME>OUTPUT</CHAIN_NAME>
    - <OUT_IF>
        <NOT>null</NOT>
        <VALUE>$IFACE</VALUE>
      </OUT_IF>
      <JUMP_TARGET>DROP</JUMP_TARGET>
      <TARGET_EXT />
    </RULE>
  </CONFIG_CHAIN>
</FIREWALL_CONFIG>
```

รูปที่ 7-20 แสดงเอกสาร XML ที่มีการสร้างกฎแล้ว (4)

- **library ที่จะเป็นที่จะต้องนำไปใช้ในการ CLASSPATH ของโปรแกรม คือ**
    - jbcl.jar เป็นของ Jbuilder5
    - xerces.jar เป็น library สำหรับ DOM ในส่วนของ XML
    - mm.mysql-2.0.4-bin.jar เป็นไดร์เวอร์ที่นำไปใช้ในการติดต่อฐานข้อมูล mySQL
    - jce-jdk13-112.jar นำไปใช้ในการเข้ารหัส

บทที่ 8
ลงมือสร้างไฟร์วอลล์ โดยใช้ IPtables

8.1 กำหนดกฎ

```
# ═══════════════════════════════════════════════
# ปิด ipchains
modprobe –r ipchains
# load module ต่างๆ
modprobe ip_tables
modprobe iptable_filter
modprobe iptable_nat
modprobe conntrack
modprobe conntrack_ftp

#enable forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# ═══════════════════════════════════════════════
# เคลียร์กฎที่มีอยู่ออกทั้งหมด
iptables -F
iptables -X
iptables -Z

# เคลียร์ที่เทเบิล nat ด้วย
iptables -t nat -F
iptables -t nat -X
iptables -t nat -Z

# policy
iptables -P INPUT DROP
iptables -P FORWARD DROP
## เครื่อง firewall สามารถออกได้อย่างอิสระ
```

```
iptables -P OUTPUT ACCEPT

# ============================================================

# constant variables

IN_IF="eth0"

EX_IF="eth1"

IN_HOST="192.168.39.0/24"

FW_ADDR_IN="192.168.39.1"

FW_ADDR_EX="161.246.39.1"

NET_ADDR="161.246.39.0"

BROADCAST="161.246.39.255"

IN_NET_ADDR="192.168.39.0"

IN_BROADCAST="192.168.39.255"

IN_WEB_ADDR="192.168.39.2"

EX_WEB_ADDR="161.246.39.2"

IN_MAIL_ADDR="192.168.39.3"

EX_MAIL_ADDR="161.246.39.2"

CLASS_A="10.0.0.0/8"

CLASS_B="172.16.0.0/12"

CLASS_C="192.168.0.0/16"

CLASS_D_MULTICAST="224.0.0.0/4"

CLASS_E_RESERVED_NET="240.0.0.0/5"

LOOPBACK="127.0.0.0/8"

# ============================================================

# protect chain มีไว้ตรวจสอบรูปแบบแพ็กเก็ตต่างๆ ที่เข้ามาภายใน

iptables –N PROTECT

## ป้องกันการ attack

### land attack

#### ด้วยการห้ามแพ็กเก็ตที่มี source address = destination address = firewall address

iptables -A PROTECT -s $FW_ADDR_EX -d $FW_ADDR_EX -j DROP

### smurf attack

#### ด้วยการห้ามส่งแพ็กเก็ตมาที่ net address และ broadcast address

iptables -A PROTECT -d $NET_ADDR -j DROP
```

```
iptables -A PROTECT -d $BROADCAST -j DROP
iptables -A PROTECT -d $IN_NET_ADDR -j DROP
iptables -A PROTECT -d $IN_BROADCAST -j DROP
### fragments
#### บันทึก log เก็บไว้ก่อน จึงค่อย drop
iptables -A PROTECT -f -m limit -j LOG --log-prefix "FRAGMENTS: "
iptables -A PROTECT -f -j DROP

## ป้องกันการ attack โดยอาศัย icmp
### ping flood
#### โดยการจำกัดปริมาณการ ping (icmp echo-request)
iptables -A PROTECT -p icmp --icmp-type echo-request -m limit \
        --limit 3/s --limit-burst 15 -j ACCEPT
iptables -A PROTECT -p icmp --icmp-type echo-request -j DROP
### icmp timestamp attack
iptables -A PROTECT -p icmp --icmp-type timestamp-request –j DROP

## icmp logging
### เพื่อใช้ตรวจสอบว่ามี icmp packet รูปแบบใดอีกหรือเปล่าที่ต้องป้องกัน
iptables -A PROTECT -p icmp -m limit -j LOG --log-prefix "ICMP: "
### ยอมรับ icmp packet รูปแบบอื่นๆ ที่ไม่ใช่ ping
iptables -A PROTECT -p icmp -j ACCEPT

## ป้องกันการ attack โดยอาศัย udp
### diagnostic port attack
#### disable small services at "/etc/inetd/conf"
#### และ drop แพ็กเก็ตที่ใช้ small services port
#### 7-echo
#### 9-discard
#### 13-daytime
#### 19-chargen
#### 37-time
iptables -A PROTECT -p udp -m multiport --sport 7,9,13,19,37 -j DROP
```

```
iptables -A PROTECT -p udp -m multiport --dport 7,9,13,19,37 -j DROP

## udp logging
### เพื่อใช้ตรวจสอบว่ามี udp packet รูปแบบใดอีกหรือเปล่าที่ควรยอมรับ
iptables -A PROTECT -p udp -m limit -j LOG --log-prefix "UDP: "
iptables -A PROTECT -p udp -j DROP

## ป้องกันการ attack โดยอาศัย tcp
### trojans
### โดยการบันทึก log ไว้ แล้ว drop ทิ้ง
#### deepthroat
iptables -A PROTECT -p tcp -m multiport --dport 41,999,2140,3150,6670,6771,60000 \
        -m limit -j LOG --log-prefix "DeepThroat: "
iptables -A PROTECT -p tcp -m multiport --dport 41,999,2140,3150,6670,6771,60000 \
        -j DROP
#### subseven
iptables -A PROTECT -p tcp -m multiport --dport 2773,6711,6712,6713,7215,27374,27573,54283 \
        -m limit -j LOG --log-prefix "SubSeven: "
iptables -A PROTECT -p tcp -m multiport --dport 2773,6711,6712,6713,7215,27374,27573,54283 \
        -j DROP
#### netbus
iptables -A PROTECT -p tcp -m multiport --dport 12345,12346,20034 \
        -m limit -j LOG --log-prefix "NetBus: "
iptables -A PROTECT -p tcp -m multiport --dport 12345,12346,20034 \
        -j DROP
#### back orifice (bo)
iptables -A PROTECT -p tcp -m multiport --dport 8787,31337,54320 \
        -m limit -j LOG --log-prefix "Back Orifice: "
iptables -A PROTECT -p tcp -m multiport --dport 8787,31337,54320 \
        -j DROP
### diagnostic port attack
iptables -A PROTECT -p tcp -m multiport --sport 7,9,13,19,37 -j DROP
iptables -A PROTECT -p tcp -m multiport --dport 7,9,13,19,37 -j DROP
### syn flood
```

โดยการจำกัดปริมาณ syn packet

iptables -A PROTECT -p tcp --syn -m limit --limit 3/s --limit-burst 15 -j ACCEPT

iptables -A PROTECT -p tcp --syn -j DROP

tcp logging

เพื่อใช้ตรวจสอบว่ามี tcp packet รูปแบบใดอีกหรือเปล่าที่ควรยอมรับ

iptables -A PROTECT -p tcp -m limit -j LOG --log-prefix "TCP: "

iptables -A PROTECT -p tcp -j DROP

logging

iptables -A PROTECT -m limit -j LOG --log-prefix "OTHERS: "

iptables -A PROTECT -j DROP

═══════════════════════════════════════

 # input chain

drop invalid and illegal packets

iptables -A INPUT -m unclean -j DROP

iptables -A INPUT -m state --state INVALID -j DROP

allow unlimited traffic on the loopback interface.

iptables -A INPUT -i lo -j ACCEPT

ยอมรับแพ็กเก็ตทั้งหมดจาก internal network

iptables -A INPUT -i $IN_IF -s $IN_HOST -j ACCEPT

drop แพ็กเก็ตที่ร้องขอการเชื่อมต่อจากภายนอกทั้งหมด

คือ ไม่อนุญาตให้ภายนอกสร้างการเชื่อมต่อได้

iptables -A INPUT -i $EX_IF -m state --state NEW -j DROP

ป้องกันการปลอมแปลง ip (spoofing)

เครื่องภายนอกไม่มีทางใช้ ip address เดียวกับเครื่องไฟร์วอลล์ได้

iptables -A INPUT -i $EX_IF -s $FW_ADDR_EX -j DROP

เครื่องภายนอกไม่มีทางใช้ private ip ได้

iptables -A INPUT -i $EX_IF -s $CLASS_A -j DROP

```
iptables -A INPUT -i $EX_IF -s $CLASS_B -j DROP
iptables -A INPUT -i $EX_IF -s $CLASS_C -j DROP
iptables -A INPUT -i $EX_IF -s $CLASS_D_MULTICAST -j DROP
iptables -A INPUT -i $EX_IF -s $CLASS_E_RESERVED_NET -j DROP
### เครื่องภายนอกไม่มีทางใช้ loopback address ได้
iptables -A INPUT -i $EX_IF -s $LOOPBACK

## ยอมรับแพ็กเก็ตจากภายนอก
## ที่เป็นส่วนหนึ่งของการเชื่อมต่อที่สร้างโดยเครื่องไฟร์วอลล์เอง
iptables -A INPUT -i $EX_IF -m state --state ESTABLISHED -j ACCEPT

## RELATED packet jump to PROTECT
iptables -A INPUT -i $EX_IF -m state --state RELATED -j PROTECT

# ═══════════════════════════════════════════════════
# forward chain
## drop invalid and illegal packets
iptables -A FORWARD -m unclean -j DROP
iptables -A FORWARD -m state --state INVALID -j DROP

# (from external to internal network)
## ป้องกันการปลอมแปลง ip (spoofing)
### เครื่องภายนอกไม่มีทางใช้ ip address เดียวกับเครื่องไฟร์วอลล์ได้
iptables -A FORWARD -i $EX_IF -o IN_IF -s $FW_ADDR_EX -j DROP
### เครื่องภายนอกไม่มีทางใช้ private ip ได้
iptables -A FORWARD -i $EX_IF -o $IN_IF -s $CLASS_A -j DROP
iptables -A FORWARD -i $EX_IF -o $IN_IF -s $CLASS_B -j DROP
iptables -A FORWARD -i $EX_IF -o $IN_IF -s $CLASS_C -j DROP
iptables -A FORWARD -i $EX_IF -o $IN_IF -s $CLASS_D_MULTICAST -j DROP
iptables -A FORWARD -i $EX_IF -o $IN_IF -s $CLASS_E_RESERVED_NET -j DROP
### เครื่องภายนอกไม่มีทางใช้ loopback address ได้
iptables -A FORWARD -i $EX_IF -o $IN_IF -s $LOOPBACK

## ยอมรับแพ็กเก็ตที่เป็นส่วนหนึ่งของการเชื่อมต่อที่สร้างโดยภายในเอง
```

```
iptables -A FORWARD -i EX_IF -o $IN_IF -m state --state ESTABLISHED -j ACCEPT

#(from external to server)
## ยอมให้ภายนอกใช้บริการ web and mail server
iptables -A FORWARD -i EX_IF -o $IN_IF -d $IN_WEB_ADDR -p tcp --dport 80 -m state --state
NEW -j PROTECT
iptables -A FORWARD -i EX_IF -o $IN_IF -d $IN_MAIL_ADDR -p tcp --dport 25 -m state --state
NEW -j PROTECT

#(from external to internal host)
## drop แพ็กเก็ตที่ร้องขอการเชื่อมต่อจากภายนอกทั้งหมด
## คือ ไม่อนุญาติให้ภายนอกสร้างการเชื่อมต่อได้
iptables -A FORWARD -i $EX_IF -o $IN_IF -d $IN_HOST -m state --state NEW -j DROP

## RELATED packet jump to PROTECT
iptables -A FORWARD -i $EX_IF -o $IN_IF -m state --state RELATED -j PROTECT

# (from internal network to external)
## ป้องกัน trojans
## โดยการบันทึก log ไว้ แล้ว drop ทิ้ง
### deepthroat
iptables -A FORWARD -i $IN_IF -o $EX_IF -p tcp -m multiport --sport
41,999,2140,3150,6670,6771,60000 \
        -m limit -j LOG --log-prefix "DeepThroat: "
iptables -A FORWARD -i $IN_IF -o $EX_IF -p tcp -m multiport --sport
41,999,2140,3150,6670,6771,60000 \
        -j DROP
### subseven
iptables -A FORWARD -i $IN_IF -o $EX_IF -p tcp -m multiport --sport
2773,6711,6712,6713,7215,27374,27573,54283 \
        -m limit -j LOG --log-prefix "SubSeven: "
iptables -A FORWARD -i $IN_IF -o $EX_IF -p tcp -m multiport --sport
2773,6711,6712,6713,7215,27374,27573,54283 \
        -j DROP
```

```
### netbus
iptables -A FORWARD -i $IN_IF -o $EX_IF -p tcp -m multiport --sport 12345,12346,20034 \
        -m limit -j LOG --log-prefix "NetBus: "
iptables -A FORWARD -i $IN_IF -o $EX_IF -p tcp -m multiport --sport 12345,12346,20034 \
        -j DROP
### back orifice (bo)
iptables -A FORWARD -i $IN_IF -o $EX_IF -p tcp -m multiport --sport 8787,31337,54320 \
        -m limit -j LOG --log-prefix "Back Orifice: "
iptables -A FORWARD -i $IN_IF -o $EX_IF -p tcp -m multiport --sport 8787,31337,54320 \
        -j DROP

## ยอมรับแพ็กเก็ตจากภายนอกออกสู่ภายนอกทั้งหมด
## แต่บันทึก log ไว้ก่อน เผื่อไว้ใช้ตรวจสอบ
iptables -A FORWARD -i $IN_IF -o $EX_IF -m limit -j LOG --log-prefix "FORWARD-OUT: "
iptables -A FORWARD -i $IN_IF -o $EX_IF -j ACCEPT

# ═══════════════════════════════════════════════
# output chain
## ป้องกัน trojans
## โดยการบันทึก log ไว้ แล้ว drop ทิ้ง
### deepthroat
iptables -A OUTPUT -p tcp -m multiport --sport 41,999,2140,3150,6670,6771,60000 \
        -m limit -j LOG --log-prefix "DeepThroat: "
iptables -A OUTPUT -p tcp -m multiport --sport 41,999,2140,3150,6670,6771,60000 \
        -j DROP
### subseven
iptables -A OUTPUT -p tcp -m multiport --sport 2773,6711,6712,6713,7215,27374,27573,54283 \
        -m limit -j LOG --log-prefix "SubSeven: "
iptables -A OUTPUT -p tcp -m multiport --sport 2773,6711,6712,6713,7215,27374,27573,54283 \
        -j DROP
### netbus
iptables -A OUTPUT -p tcp -m multiport --sport 12345,12346,20034 \
        -m limit -j LOG --log-prefix "NetBus: "
iptables -A OUTPUT -p tcp -m multiport --sport 12345,12346,20034 \
```

```
        -j DROP
### back orifice (bo)
iptables -A OUTPUT -p tcp -m multiport --sport 8787,31337,54320 \
        -m limit -j LOG --log-prefix "Back Orifice: "
iptables -A OUTPUT -p tcp -m multiport --sport 8787,31337,54320 \
        -j DROP

## ยอมรับแพ็กเก็ตที่ออกจากเครื่องไฟร์วอลล์ทั้งหมด
## แต่บันทึก log ไว้ก่อน เผื่อไว้ใช้ตรวจสอบ
iptables -A OUTPUT -m limit -j LOG --log-prefix "OUTPUT: "
iptables -A OUTPUT -j ACCEPT

# ═══════════════════════════════════════════
# NAT
## snat
iptables -t nat -A POSTROUTING -s $IN_HOST -i $IN_IF -o $EX_IF -j SNAT --to $FW_ADDR_EX
## dnat
### web server
iptables -t nat -A PREROUTING -d $EX_WEB_ADDR -p tcp --dport 80 -j DNAT --to
$IN_WEB_ADDR
### mail server
iptables -t nat -A PREROUTING -d $EX_MAIL_ADDR -p tcp --dport 25 -j DNAT --to $IN_MAIL_ADDR
```

8.2 การเซ็ตให้รันไฟร์วอลล์ขณะบูต ใน Linux Redhat 7.1 (รันจากไฟล์)

สำหรับ Linux Redhat 7.1 กฎต่างๆ ที่เราสร้างขึ้นใน 9.1 นั้น สามารถกำหนดให้รันโดยอัตโนมัติขณะบูตเครื่องได้โดย

1. การสร้างเป็นไฟล์สคริปต์ (script) โดยบันทึกเป็นชื่อไฟล์อะไรก็ได้ แต่ในที่นี้จะใช้ชื่อว่า “rc.firewall”
2. จากนั้นนำไฟล์ “rc.firewall” ไปเก็บไว้ในพาร์ท (path) /etc/rc.d
3. กำหนดเพอร์มิสชั่น (permission) ให้ไฟล์ “rc.firewall” เป็น 700 เพื่อให้สามารถเอ็กซีคิวซ์ (execute) ได้ โดยใช้คำสั่งต่อไปนี้

   chmod rc.firewall 700

4. จากนั้นบันทึกตำแหน่งที่อยู่ของ “rc.firewall” ลงในไฟล์ rc.local เพื่อให้สคริปต์ “rc.firewall” ถูกรันทุกครั้งที่มีการบูต */etc/rc.d/rc.firewall*

บทที่ 9
ทดสอบการทำงานของไฟร์วอลล์ที่สร้างขึ้น

9.1 เครื่องที่ใช้ในการทดสอบ

- *เครื่อง A*

  CPU: Intel Pentium II 350 MHz

  RAM: 256 MB

  OS: Microsoft Windows 98 Second Edition

- *เครื่อง B*

  CPU: Intel Pentium II 233

  RAM: 32 MB

  OS: Microsoft Window 98

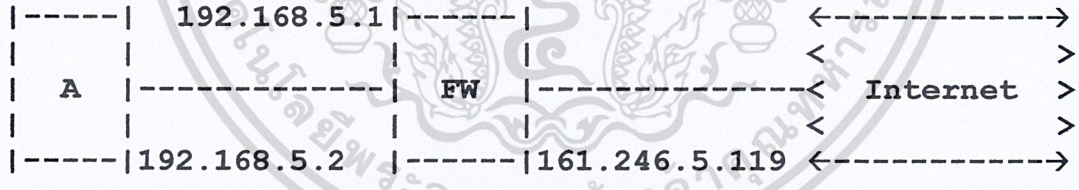
- *เครื่อง FW*

  CPU: Intel Pentium III 500 MHz

  RAM: 128 MB

  OS: Linux Redhat 7.1

9.2 ทดสอบการใช้งานอินเตอร์เน็ตสำหรับเครื่องภายใน



รูปที่ 9-1 ทดสอบการใช้งานอินเตอร์เน็ตสำหรับเครื่องภายใน

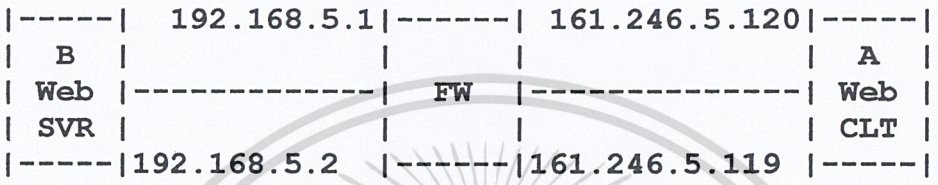
- *กำหนดให้*
  - *เครื่อง FW* เป็นเครื่องไฟร์วอลล์ กั้นกลางระหว่างเครือข่ายภายใน 192.168.5.0 กับเครือข่ายภายนอก 161.246.5.0 โดยใช้แอดเดรสของอินเตอร์เฟสภายในเป็น 192.168.5.1 และแอดเดรสของอินเตอร์เฟสภายนอกเป็น 161.246.5.119
  - *เครื่อง A* เป็นเครื่องภายใน ใช้ไอพีแอดเดรสภายใน คือ 192.168.5.2 และตั้งเกตเวย์ไปที่เครื่อง FW
- *วิธีทดสอบ*

ทดสอบโดยการใช้เครื่อง A เปิดเว็บต่างๆ เช่น www.kmitl.ac.th, www.ce.kmitl.ac.th, www.sanook.com และ www.google.com

- *ผลการทดสอบ*

  เครื่อง A สามารถเปิดเว็บต่างๆ ได้ ดังนั้น **การใช้งานอินเตอร์เน็ตสำหรับเครื่องภายในสามารถใช้ได้ตามปกติ** เมื่อใช้ไฟร์วอลล์

9.3 ทดสอบการให้บริการของเซิร์ฟเวอร์



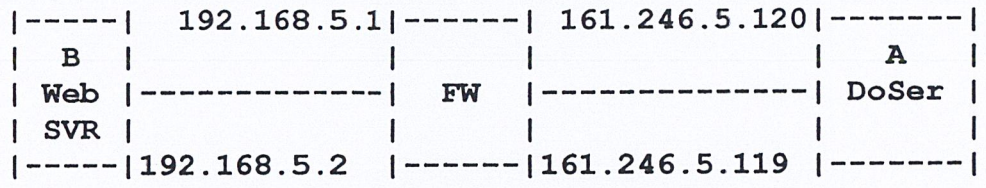
รูปที่ 9-2 ทดสอบการให้บริการของเซิร์ฟเวอร์

- *กำหนดให้*
  - ***เครื่อง FW*** เป็นเครื่องไฟร์วอลล์ กั้นกลางระหว่างเครือข่ายภายใน 192.168.5.0 กับเครือข่ายภายนอก 161.246.5.0 โดยใช้แอดเดรสของอินเตอร์เฟสภายในเป็น 192.168.5.1 และแอดเดรสของอินเตอร์เฟสภายนอกเป็น 161.246.5.119
  - ***เครื่อง A*** เป็นเครื่องภายนอก ใช้ไอพีแอดเดรสจริง คือ 161.246.5.120
  - ***เครื่อง B*** ทำงานเป็นเครื่องเว็บเซิร์ฟเวอร์ ใช้ไอพีแอดเดรสภายใน คือ 192.168.5.2 แต่ภายนอกสามารถขอใช้บริการโดยใช้ไอพี 161.246.5.99 เพราะใช้ NAT เป็นตัวแปลงไอพีแอดเดรสให้
- *วิธีทดสอบ*

  ทดสอบโดยการใช้เครื่อง A ขอใช้บริการจากเครื่อง B ซึ่งเป็นเว็บเซิร์ฟเวอร์ โดยพิมพ์ “161.246.5.99” ที่ช่อง Address ของบราวเซอร์
- *ผลการทดสอบ*

  เครื่อง A สามารถเปิดเว็บได้ ดังนั้น **การให้บริการของเซิร์ฟเวอร์ เป็นไปตามปกติ** เมื่อใช้ไฟร์วอลล์

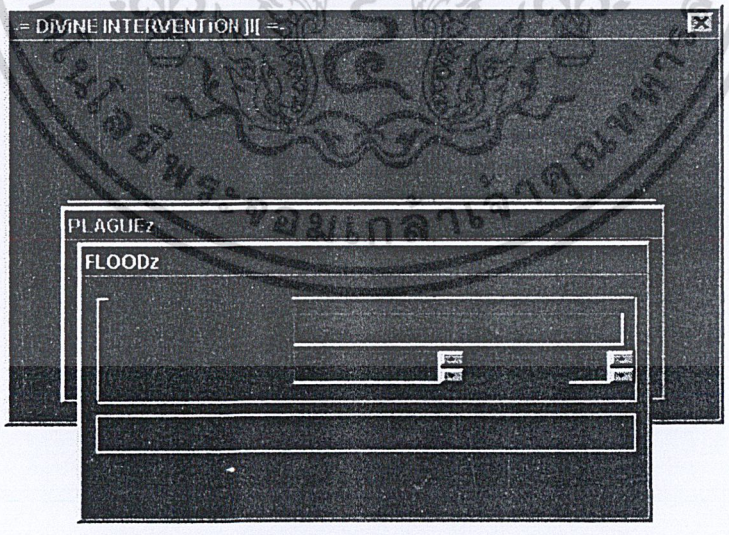
9.4 ทดสอบการโจมตีแบบ DoS ที่เซิร์ฟเวอร์จากภายนอก



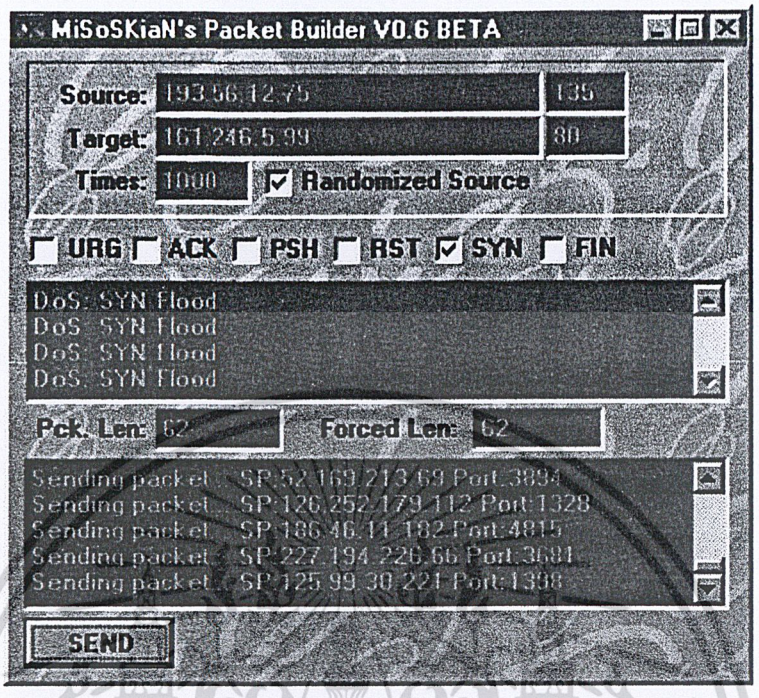
รูปที่ 9-3 ทดสอบการโจมตีแบบ DoS ที่เซิร์ฟเวอร์จากภายนอก

- *กำหนดให้*
  - *เครื่อง FW* เป็นเครื่องไฟร์วอลล์ กั้นกลางระหว่างเครือข่ายภายใน 192.168.5.0 กับเครือข่ายภายนอก 161.246.5.0 โดยใช้แอดเดรสของอินเตอร์เฟสภายในเป็น 192.168.5.1 และแอดเดรสของอินเตอร์เฟสภายนอกเป็น 161.246.5.119
  - *เครื่อง A* เป็นเครื่องภายนอก ใช้ไอพีแอดเดรสจริง คือ 161.246.5.120
  - *เครื่อง B* เป็นเครื่องภายใน ใช้ไอพีแอดเดรสภายใน คือ 192.168.5.2
- *วิธีทดสอบ*

  ทดสอบโดยการใช้เครื่อง A ทำการ Ping Flood ด้วยโปรแกรม *DiViNE INTERVENTiON][* และ SYN Flood ด้วยโปรแกรม *MiSoSKiaN's Packet Builder V0.6 BETA* ไปที่เครื่อง B ซึ่งเป็นเว็บเซิร์ฟเวอร์ (161.246.5.99)



รูปที่ 9-4 โปรแกรม DiViNE INTERVENTiON][



รูปที่ 9-5 โปรแกรม MiSoSKiaN's Packet Builder V0.6 BETA

- *ผลการทดสอบ*

  เครื่อง B สามารถทำงานได้ตามปกติ ไม่ได้รับผลกระทบจาก Ping Flood และ SYN Flood เลย ดังนั้น **ไฟร์วอลล์สามารถป้องกันการโจมตีแบบ DoS จากภายนอกได้**

- **การทดสอบ Dos โปรแกรมลักษณะอื่น ๆ**

โปรแกรม	ลักษณะ	ผลการทดสอบ
Targa.c	Unclean package	√
Smurf.c	ICMP broadcast	√
Pingflood.c	Ping flood	√
Smack.c	ICMP unreachable	√
Synful.c	SYN flood	√
Synk4.c	IP Spoofing	√
MiSoSKian	SYN flood	√

ตารางที่ 9-1 โปรแกรมอื่น ๆ Dos ที่ใช้ในการทดสอบ

บทที่ 10
สรุปและวิจารณ์

ในการใช้ไฟล์วอลล์ของลีนุกส์โดยปกติแล้วจะต้องเขียนโดยใช้การเขียนสคริปท์ไฟร์วอลล์ ซึ่งจะยากต่อการใช้งานของผู้ใช้ ปริญญานิพนธ์นี้จึงได้พัฒนาโปรแกรมที่ช่วยในการติดตั้งไฟร์วอลล์ไอพีเทเบิลให้ง่ายต่อการใช้งาน และพัฒนาเข้าใจการใช้ script ไฟร์วอลล์ไอพีเทเบิลในลีนุกส์ให้มากขึ้น และนำเอาเทคโนโลยีของ XML มาใช้ในการอธิบายกฎในการคอนฟิก

ข้อดี : 1. จะมีความสะดวกต่อการติดตั้งไฟร์วอลล์เพราะภายในโปรแกรมจะมีช่องให้กรอกรายละเอียดในการคอนฟิกจะทำให้ผู้ใช้ ไม่สับสนในการกรอกกฎ
2. การคอนฟิกกฎระยะไกล จะทำให้ผู้ใช้ไม่จำเป็นต้องคอนฟิกที่เครื่องไฟร์วอลล์
3. การแจ้งเตือนผู้ใช้เมื่อมีผู้บุกรุกเข้ามา
4. มีการเข้ารหัสเพื่อความปลอดภัย

ข้อเสีย : 1. ผู้ใช้โปรแกรมจะเป็นต้องศึกษารายละเอียดของสคริปท์ไฟร์วอลล์ไอพีเทเบิลอยู่เพราะว่าในภายในโปรแกรมยังอ้างอิงรูปแบบของ สคริปท์ในบางส่วน
2. การเข้ารหัสจะเป็นการเข้าแบบ RSA ซึ่งจะมีความปลอดภัยมาก แต่ข้อเสียคือจะช้ามาก

ข้อจำกัด : ในโปรแกรมอาจไม่สนับสนุนบางแท็กของสคริปท์ และสคริปท์ไอพีเทเบิลมีการพัฒนาอย่างต่อเนื่องซึ่งโปรแกรมอาจจะใช้ไม่ได้ในบางจุด

วิธีแก้ไข : จำเป็นต้องมีการพัฒนาโปรแกรมเพื่อสนับสนุนสคริปท์

ปัญหาและอุปสรรค : ปัญหาการใช้สวิงค์ในภาษาจาวา ซึ่งบางคอมโพเน็นต์จะใช้ต้องมีการศึกษาพอสมควร

แนวทางการพัฒนาต่อไป : ศึกษาสคริปท์ไฟร์วอลล์ไอพีเทเบิลอย่างต่อเนื่องเมื่อมีการออกเวอร์ชันใหม่ ๆ แล้วพัฒนาโปรแกรมให้ครอบคลุมแท็ก

บรรณานุกรม

[1] Behrouz Forouzan with Catherine Coombs and Sophia Chung Fegan: "*Introduction to Data Communications and Networking*", The McGraw-Hill Companies., 1998

[2] Joel Scambray, Stuart Maclure and George Kurtz: "*Hacking Exposed: Network Security Secrets & Solutions (Second Edition)*", The McGraw-Hill Companies., 2001

[3] Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman: "*Building Internet Firewalls Second Edition*", O'Reilly & Associates, Inc., 2000

[4] H. M. Deitel and P. J. Deitel: "*Java How to Program Fourth Edition*", Prentice Hall, Inc., 2002

[5] Merlin Hughes, Michael Shoffner and Derek Hamner: "*Java Network Programming Second Edition*", Manning Publications Co., 1999

[6] Elliotte Rusty Harold: "*Java Network Programming Second Edition*", O'Reilly & Associates, Inc., 2000

[7] Marco Pistoia, Duane F. Reller, Deepak Gupta, Milind Nagnur and Ashok K. Ramani: "*Java 2 Network Security*", Prentice Hall, Inc., 1999

[8] Brett McLaughlin: "*Java and XML*", O'Reilly & Associateds, Inc., 2000

[9] Michael J. Young: "*Step by Step XML*", Microsoft Corporation, 2000

[10]นิรุธ อำนวยศิลป์: "*Linux Red Hat ฉบับเพื่อการใช้งานจริง*", บริษัท ซัคเซส มีเดีย จำกัด, 2544

[11]รศ.ดร. ควงแก้ว สวามิภักดิ์: "*ระบบดำเนินงานยูนิกซ์*", บริษัท ซีเอ็ดยูเคชั่น จำกัด,

[12]สันติ ศรีลาศักดิ์ และ เกศมณี เที่ยงธรรม: "*คุณทำได้... เซ็ตอัพอินเตอร์เน็ตเซิร์ฟเวอร์บนลีนุกซ์*", บริษัท ออฟเซ็ท เพรส จำกัด

[13]เรืองไกร รังสิพล: "*เจาะระบบ TCP/IP จุดอ□น□อง□□ร□ตคอล□ละวิ□□องกัน*", บริษัท โปรวิชั่น จำกัด

[14]สงกรานต์ ทองสว่าง: "*MyS□L ระบบ□าน□้อ□ูล□ำ□รับอินเตอร์เน็ต*", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 2544

[15]รุ่งโรจน์ โพนคำ และ ปราลี มณีรัตน์: "*□dvanced Java Programming*", บริษัท ซัคเซส มีเดีย จำกัด