โปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่าย

Network Traffic Replay and Generator Tool

นายธีรศักดิ์ ปัญญาเกริกไกร

นางสาวกุลวดี อิงคพันธ์พงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

โปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่าย

Network Traffic Replay and Generator Tool

โดย

นายธีรศักดิ์ ปัญญาเกริกไกร

นางสาวกุลวดี อิงคพันธ์พงศ์

อาจารย์ที่ปรึกษา

อาจารย์ธนัญชัย ตรีภาค

อาจารย์อัครเดช วัชระภูพงษ์

อาจารย์ธนา หงษ์สุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชา วิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่าย

Network Traffic Replay and Generator Tool

ผู้จัดทำ

1. นายธีรศักดิ์ ปัญญาเกริกไกร รหัสประจำตัว 44010224
2. นางสาวกุลวดี อิงคพันธ์พงศ์ รหัสประจำตัว 44010032

_____ อาจารย์ที่ปรึกษา

(อาจารย์ธนัญชัย ตรีภาค)

_____ อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

_____ อาจารย์ที่ปรึกษา

(อาจารย์ธนา หงษ์สุวรรณ)

โปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่าย

นายธีรศักดิ์ ปัญญาเกริกไกร รหัส 44010224
นางสาวกุลวดี อิงคพันธ์พงศ์ รหัส 44010032
อาจารย์ธนัญชัย ตรีภาค อาจารย์ที่ปรึกษา
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
อาจารย์ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

โดยทั่วไปอุปกรณ์เน็ตเวิกต้องมีความสามารถในการประมวลผล การไหลของข้อมูลหรือทราฟฟิกที่มาจากหลายๆทางได้และเพื่อตรวจสอบการทำงานที่เหมาะสมของตัวอุปกรณ์จึงทำการทดสอบโดยการสร้างทราฟฟิกไปยังอุปกรณ์เหล่านั้นโดยที่มีการกำหนดลักษณะการทำงานที่ซับซ้อนได้

วิทยานิพนธ์นี้อธิบายหน้าที่ของโปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่ายซึ่งถูกใช้เพื่อแสดงลักษณะของประสิทธิภาพของพิธีการสื่อสารเครือข่ายแบบแพ็กเก็ตสวิชต์ โดยสร้างหรือดักจับแพ็คเก็ตแบบทางเดียวที่ผ่านไปมาบนเครือข่ายและการทดสอบโปรโตคอลที่แตกต่างกันเพื่อหาความแตกต่างกันของประสิทธิภาพซึ่งผลของการทำงานของโปรแกรมจะช่วยในการวิเคราะห์ประสิทธิภาพของระบบ,ประเมินประสิทธิภาพของอุปกรณ์บนเครือข่ายเพื่อใช้ในการปรับปรุงเครือข่ายให้ทำงานได้อย่างเหมาะสมนอกจากนั้นยังใช้เพื่อศึกษาการทำงานของโปรโตคอลในระดับชั้นต่างๆในระบบเครือข่ายศึกษาลักษณะการโจมตีแบบต่างๆในระบบเครือข่า เพื่อใช้ในการวิเคราะห์ลักษณะของเครือข่ายและแก้ไขให้เหมาะสม

Network Traffic Replay and Generator Tool

Mr.Teerasak Panyakrekkai
Miss.Kulwadee Engkapunpong
Mr.Thanunchai Threepak Advisor
Mr.Akkradach Watcharapupong Advisor
Mr.Thana Hongsuwan Adviser

ABSTACT

Networking devices must be capable of processing traffic flows from multiple sources. In order to verify that such device operates properly, a network test bench can be used to inject traffic into the device. The specification of the traffic flows can be difficult.

For this project describes The network traffic replay and generator tool that can be used to characterize the performance of packet-switched network communication protocol and can generates and receives one-way packet traffic streams transmitted form the UNIX user level process between traffic source and traffic sink nodes in a network. Different protocol may be tested to ascertain differences in performance. This operation can be used to analyze the system efficiency by the consideration of network blocking characteristics and evaluate the performance of network devices. With all of this information, we can analyze the network and adjust the system to make them work properly.

In addition, this program can help us to understand the working concept of each protocol in OSI model and attack patterns in computer network system.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี เนื่องจากได้รับการแนะนำ สนับสนุน และให้คำปรึกษาเป็นอย่างดีจากอาจารย์ธนัญชัย ตรีภาค, อาจารย์อัครเดช วัชระภูพงษ์ และอาจารย์ธนา หงษ์สุวรรณอาจารย์ที่ปรึกษาปริญญานิพนธ์ซึ่งต้องขอขอบพระคุณเป็นอย่างสูงรวมทั้งอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่านที่ให้การอบรมสั่งสอนวิชาความรู้แก่คณะผู้จัดทำมาโดยตลอดและขอขอบพระคุณเป็นอย่างสูงสำหรับบุคคลที่สำคัญที่สุดที่ทำให้คณะผู้จัดทำมีวันนี้ คือ บิดา มารดา ผู้เป็นที่เคารพรักยิ่งของคณะผู้จัดทำ ซึ่งท่านให้การอบรมสั่งสอนเลี้ยงดู และให้โอกาสในการศึกษาอย่างเต็มที่ จึงขอกราบขอบพระคุณมา ณ ที่นี้

สุดท้ายนี้ขอขอบพระคุณผู้ดูแลระบบคอมพิวเตอร์ภาควิชาวิศวกรรมศาสตร์และสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่อำนวยความสะดวกในการใช้งานเครือข่ายและขอขอบคุณเพื่อนๆ ที่ให้ข้อคิดเป็น และเป็นกำลังใจให้เสมอมา

คณะผู้จัดทำ

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญภาพประกอบ	VII
บทที่ 1 บทนำ	1
1.1 ความสำคัญ และที่มา	1
1.2 วัตถุประสงค์ของวิทยานิพนธ์	1
1.3 ขอบเขตของวิทยานิพนธ์	1
1.4 ขั้นตอนการดำเนินงาน	1
บทที่ 2 โพรโตคอลทีซีพี/ไอพี	2
2.1 ความเป็นมาของโพรโตคอลทีซีพี/ไอพี	3
2.2 การเชื่อมต่อของโพรโตคอลทีซีพี/ไอพี	3
2.3 โพรโตคอลแสตก	3
2.4 โพรโตคอลทีซีพี	5
2.5 โพรโตคอลยูดีพี	6
2.6 โพรโตคอลไอพี	8
2.7 โพรโตคอลเออาร์พี	9
2.8 โพรโตคอลไอซีเอ็มพี	12
2.9 ข้อพกพร่องของทีซีพีไอพี	13
2.9.1 ขาดกลไกด้านความปลอดภัย	14
2.9.2 การตอบรับเป็นสิ่งที่คาดหมายได้	14
2.9.3 การตอบรับไว้ไม่ครอบคลุมทุกเงื่อนไข	14
บทที่ 3 สถาปัตยกรรมเครือข่ายและระดับชั้นโพรโตคอล	15
3.1 แบบอ้างอิงโอเอสไอ (OSI Reference Model)	16
3.2 ชุดโพรโตคอลทีซีพี/ไอพี (TCP/IP Protocol suite)	16
3.2.1 เลเยอร์เน็ตเวิร์คของทีซีพี/ไอพี	18
3.2.2 เลเยอร์ทรานสปอร์ตของทีซีพี/ไอพี	19
3.2.3 เลเยอร์แอพพลิเคชั่นของทีซีพี /ไอพี	33
	37

บทที่ 4 การโจมตีเครือข่าย 41

4.1 การโจมตีเพื่อให้ปิดบริการ (Denial of Services : DoS) 41

4.2.รายละเอียดการโจมตีแบบต่างๆ 41

4.2.1 ประเภทอยู่ในชั้นเน็ตเวิร์ก 41

4.2.2 ประเภทอยู่ในชั้นทรานสปอร์ต หรือชั้นอินเทอร์เน็ต 46

4.2.3 ประเภทอยู่ในชั้นแอพพลิเคชัน 50

4.3โปรแกรมที่ใช้โจมตีเพื่อให้ปิดบริการ 52

4.4 IP-Spoofing 55

บทที่ 5 Tcpdump 56

5.1 tcpdump คืออะไร? 56

5.2 การใช้งานโปรแกรม tcpdump 57

5.2.1 OPTION ของโปรแกรม tcpdump 57

5.2.2 EXPRESSION ของโปรแกรม tcpdump 58

5.3 ตัวอย่างการใช้งานและผลลัพธ์ของโปรแกรม tcpdump 60

5.3.1 เวลา (Time Stamp) 61

5.3.2 ลิงค์เลเยอร์ (Link Layer) 61

5.3.3 เน็ตเวิร์คเลเยอร์ (Network Layer) 62

5.3.4 ทรานสปอร์ตเลเยอร์ (Transport Layer) 63

บทที่ 6 หลักการทำงานย่อยของโปรแกรม 65

6.1หลักการสร้างแพ็คเก็ต 65

6.1.1 ความหมายและหลักการทำงานของ libnet 65

6.1.2ขั้นตอนการดำเนินการ 66

6.2 การทำซ้ำทราฟฟิกบนเครือข่าย 67

6.2.1 ความหมายของการทำซ้ำแพ็กเก็ตลงบนเครือข่าย (Replay packet) 67

6.2.2 องค์ประกอบของการทำซ้ำทราฟฟิกบนเครือข่าย 67

6.2.3 การทำงานของการทำซ้ำทราฟฟิกบนเครือข่าย 67

6.2.4 ประโยชน์ของการทำซ้ำทราฟฟิกบนเครือข่าย 68

6.3หลักการดักจับแพ็กเก็ต 68

6.3.1 ความหมายของแพ็คเก็ตแคปเจอร์ (Packet sniffer) 68

6.3.2 องค์ประกอบของแพ็กเก็ตแคปเจอร์ 69

6.3.3 การทำงานของแพ็กเก็ตแคปเจอร์ 70

6.3.4 ประโยชน์จากแพ็กเก็ตแคปเจอร์ 72

6.4 การออกแบบโปรแกรม 73
บทที่ 7 การทำงานของโปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่าย 75
7.1 Generate Packet 75
7.1.1 Mode Configuration file 75
7.1.2 Mode Template Packet 76
7.1.3 Mode Editable Packet 77
7.2 Replay Packet 79
7.3 Capture Packet 83
บทที่ 8 สรุปและวิจารณ์ 83
8.1 สรุปผลการทำงาน 84
8.2 ปัญหาและอุปสรรค 84
8.3 แนวทางการวิจัยและพัฒนาต่อ 85
ภาคผนวก ก 86
ภาคผนวก ข 89
ภาคผนวก ค 92
บรรณานุกรม 94

สารบัญภาพประกอบ

	หน้า
รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอ	4
รูปที่ 2-2 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี	5
รูปที่ 2-3 โพรโตคอลสแต็กของทีซีพี/ไอพี	6
รูปที่ 2-4 แสดงการทำ 3-ways Handshake	7
รูปที่ 2-5 แสดงแพ็กเก็ตทีซีพี	8
รูปที่ 2-6 แสดงแพ็กเก็ตยูดีพี	9
รูปที่ 2-7 แสดงการรีแอสเซมเบิ้ล	10
รูปที่ 2-8 แสดงแพ็กเก็ตไอพี	12
รูปที่ 2-9 เออาร์พีดาทาแกรม	13
รูปที่ 2-10 ฟอร์แมตของ ไอซีเอ็มพี	14
รูปที่ 3-1 แบบอ้างอิงโอเอสไอ	16
รูปที่ 3-2 สถาปัตยกรรมทีซีพี/ไอพี	18
รูปที่ 3-3 การส่งผ่านข้อมูลระหว่างเลเยอร์	19
รูปที่ 2-4 สถาปัตยกรรมไอพี	19
รูปที่ 3-5 แสดงไอพีใน (a) เฟรมอีเธอร์เน็ตทู (b) เฟรม SNAP	20
รูปที่ 3-6 รูปแบบของดาต้าแกรมไอพีเวอร์	21
รูปที่ 3-7 ออปชั่นในการวัดและความปลอดภัย	22
รูปที่ 3-8 รูปแบบดาต้าแกรมเออาร์พี	24
รูปที่ 3-9 แสดงไอซีเอ็มพีชนิดต่าง ๆ ถูกเอ็นแคปซูเลทในไอพี	26
รูปที่ 3-10 เฮดเดอร์พื้นฐานของโพรโตคอลไอซีเอ็มพี	26
รูปที่ 3-11 รูปแบบดาต้าแกรมไอซีเอ็มพี	26
รูปที่ 3-12 ค่า destination unreachable ในฟิลด์ Code	28
รูปที่ 3-13 (a) รูปแบบของแมสเสจ route change request (b) ค่าในฟิลด์ Code ที่แมสเสจประเภทนี้ใช้	30
รูปที่ 3-14 รูปแบบแมสเสจไอซีเอ็มพี router advertisement	30
รูปที่ 3-15 รูปแบบแมสเสจไอซีเอ็มพี router solicitation	31
รูปที่ 3-16 รูปแบบแมสเสจไอซีเอ็มพี parameter problems	31
รูปที่ 3-17 รูปแบบแมสเสจไอซีเอ็มพี time stamp request/reply	32
รูปที่ 3-18 รูปแบบแมสเสจไอซีเอ็มพี information requests	32
รูปที่ 3-19 รูปแบบแมสเสจไอซีเอ็มพี address mask request	33
รูปที่ 3-20 หมายเลขพอร์ตที่ใช้ในยูดีพีและทีซีพี	34
รูปที่ 3-21 ฟิลด์ในเฮดเดอร์ของยูดีพี	34

รูปที่ 3-22 ค่าเช็คซัมประกอบด้วยเฮดเดอร์และซูโดเฮดเดอร์ 35
รูปที่ 3-23 เฮดเดอร์ทีซีพี 36
รูปที่ 4-1 แสดงการโจมตีด้วย Ping Flood Attack 42
รูปที่4-2 รูปแบบแพ็กเก็ตที่เกิดขึ้นจากการโจมตีจาก Ping Flood Attack 42
รูปที่ 4-3 แสดง การโจมตีด้วย Smurf 43
รูปที่ 4-4 แสดงการรีแอสเซมบลีแบบปกติ 44
รูปที่ 4-5 แสดงแพ็กเก็ตสุดท้ายที่ต้องรอแพ็กเก็ตก่อนหน้า 44
รูปที่ 4-6 แสดงการรีแอสเซมบลีแบบแพ็กเก็ตมีขนาดเหลื่อมกัน 45
รูปที่4-7 รูปแสดงการโจมตีด้วย Land Attack 46
รูปที่ 4-8 แสดงแผนภูมิแสดงประเภทของการโจมตีเพื่อให้ปิดบริการสำหรับสแต็กทีซีพี/ไอพี 46
รูปที่ 4-9 แสดงการส่งแพ็กเก็ตแบบ SYN Flood 47
รูปที่ 4–10 แพ็กเก็ตที่เกิดขึ้นจากการโจมตีแบบ SYN Flood 47
รูปที่ 4-11 แสดงการสแกนของผู้บุกรุก 49
รูปที่ 5-1 แสดงความหมายของแพ็กเกจที่ได้จากโปรแกรม tcpdump 64
รูปที่ 6-1 แสดงโปรโตคอลที่ลิปเน็ตสนับสนุน 65
รูปที่ 6-2 แสดงการขั้นตอนการสร้างและส่ง Packet ลงบนเครือข่าย 67
รูปที่ 6-3 การทำงานของแพ็กเก็ตแคปเจอร์ 71
รูปที่ 6-4 แสดงการสร้างแบบ LIBNET_LINK 72
รูปที่ 7-1 แสดงส่วนของ Configure file 75
รูปที่ 7-2 แสดงผลของการสร้างจาก Configure file 76
รูปที่ 7-3 ตัวอย่างการฟอร์แมตของแพ็คเก็ตที่ต้องการสร้าง 76
รูปที่ 7-4 แสดงรายระเอียดของการสร้าง Packet TCP ที่สามารถแก้ไขได้ 77
รูปที่ 7-5 แสดงการสร้าง Syncflood 77
รูปที่ 7-6 แสดงผลของการสร้าง Syncflood 78
รูปที่ 7-7 แสดงการสร้างแพ็กเก็ตชนิด ICMP 78
รูปที่ 7-8 แสดงผลของการสร้างแพ็กเก็ตชนิด ICMP 79
รูปที่ 7-9 ตัวอย่าง การ Replay packet แบบเลือก match 80
รูปที่ 7-10 ผลของการทำ Replay packet แบบเลือก match 80
รูปที่ 7-11 ตัวอย่างฟังก์ชั่นของ การ Replay packet แบบเลือกไม่ match 81
รูปที่ 7-12 แสดงผลของการทำ Replay packet แบบเลือกไม่ match 81
รูปที่ 7-13 แสดงการทำ Replay packet แบบเลือกใน list 82
รูปที่ 7-14 แสดงผลของการทำ Replay packet แบบเลือกใน list 82
รูปที่ 7-15 แสดงการใช้ฟังก์ชั่นในการดักจับแพ็คเก็ต 83
รูปที่ 7-16 แสดงผลของการดักจับแพ็คเก็ต 83

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เนื่องจากในปัจจุบันการใช้งานอินเตอร์เน็ตมีการใช้งานกันอย่างแพร่หลายและข้อมูลที่ส่งในระบบเครือข่ายก็มีมากมายหลายแบบ ระบบความปลอดภัยจึงเป็นสิ่งสำคัญ เนื่องจากมีคนใช้งานอินเตอร์เน็ตมากมาย มีทั้งผู้ใช้งาน และผู้ที่บุกรุก ทำให้ต้องมีจัดหาอุปกรณ์ที่หรือระบบที่จะป้องกันผู้ที่จะบุกรุกเข้ามา

สิ่งที่ต้องคำนึงถึงในโลกของอินเตอร์เน็ตในปัจจุบันอย่างหนึ่งคือเรื่องของความปลอดภัยทางเครือข่ายคอมพิวเตอร์ ดังนั้นก่อนที่จะเปิดให้บริการบนระบบเครือข่าย เราจึงต้องมีการทดสอบอุปกรณ์หรือระบบรักษาความปลอดภัยต่างๆ ก่อนที่จะเปิดให้บริการ โดยการจำลองสภาวะนั้นๆ กับอุปกรณ์หรือระบบที่เราต้องการทดสอบ และ ทำซ้ำสภาวะเหล่านั้นเพื่อที่จะได้นำเอาผลของการตรวจสอบมาเปรียบเทียบหาข้อบกพร่องต่างๆและนำมาวิเคราะห์แก้ไขได้ก่อนที่จะเปิดให้บริการจากเหตุผลข้างต้น จึงจำเป็นที่จะต้องมีโปรแกรมที่ช่วยในการทดสอบและสร้างสภาวะต่างๆ ของระบบเครือข่าย และทำซ้ำสภาวะนั้นๆเพื่อเป็นการเปรียบเทียบผลที่ออกมา ทำให้ระบบนั้นหรืออุปกรณ์นั้นสามารถทำงานได้ตามที่เราต้องการหรือตามที่เรากำหนดไว้

ปริญญานิพนธ์นี้จึงมุ่งเน้นการศึกษาขั้นตอนการสร้างแพ็กเก็ตในรูปแบบต่างๆและการทำซ้ำสภาวะทางครือข่ายซึ่งสามารถจำลองการทำงานบนเครือข่ายในช่วงเวลาหนึ่งๆ ได้อย่างถูกต้องแม่นยำรวมถึงการศึกษาลักษณะของแพ็กเก็ตการโจมตีแต่ละประเภทเพื่อพัฒนาโปรแกรมสร้างและทำซ้ำทราฟฟิกของเครือข่ายบนระบบปฏิบัติการลีนุกซ์ให้สามารถจำลองสภาวะดังกล่าวได้อย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์ที่จัดทำขึ้นนี้ จัดทำภายใต้วัตถุประสงค์หลัก 4 ประการ ได้แก่

(1) เพื่อศึกษาการทำงานของโปรโตคอลในระดับชั้นต่างๆ

(2) เพื่อสร้างแพ็คเกจในรูปแบบต่างๆ ได้ตามที่ต้องการ

(3) เพื่อทำซ้ำข้อมูลของระบบเครือข่ายจากไฟล์ที่ดักจับมาได้

(4) เพื่อใช้ตรวจสอบอุปกรณ์และทดสอบระบบเครือข่ายในสภาวะแวดล้อมที่เหมือนเดิมหลายๆ ครั้งได้ เพื่อใช้ในการเปรียบเทียบ ทำให้ระบบมีประสิทธิภาพมากขึ้น

1.3 ขอบเขตของปริญญานิพนธ์

ขอบเขตการทำงานของปริญญานิพนธ์นี้ ได้แก่

(1) โปรแกรมสามารถสร้างเพ็จเก็ต ในรูปแบบต่างๆ ทั้งที่เป็นรูปแบบปกติ และ รูปแบบการโจมตีเพื่อสร้างสภาวะแวดล้อมในการทดสอบประสิทธิภาพของระบบการรักษาความปลอดภัยขององค์กรและความสามารถในการรองรับการทำงานของอุปกรณ์ในระบบเครือข่ายได้

(2) โปรแกรมสามารถนำข้อมูลในเครือข่ายที่บันทึกไว้มาส่งซ้ำได้ตามที่ต้องการ

(3) เป็นโปรแกรมที่ทำงานบนระบบปฏิบัติการที่เป็นลีนุกซ์ (Linux)

(4) โปรแกรมสามารถใช้งานง่าย โดยมีการทำกราฟฟิกยูสเซอร์อินเทอร์เฟซที่เหมาะสม

1.4 ขั้นตอนการดำเนินงาน

(1) ศึกษารายละเอียดเกี่ยวกับทีซีพี/ไอพีเบื้องต้น

(2) ศึกษาการสร้างแพ็คเก็ต (packet) โดยใช้ library libnet

(3) ศึกษารายละเอียดและการลักษณะของการโจมตีเพื่อจำลองสภาวะเครือข่ายได้

(4) ศึกษาโครงสร้างของไฟล์ที่ดักจับข้อมูลมาได้โดยโปรแกรม ethereal หรือ tcpdump

(5) ศึกษาการทำซ้ำของไฟล์ข้อมูลที่ดักจับมาได้จาก library tcpreplay

(6) ออกแบบโครงสร้างของโปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่ายคอมพิวเตอร์

(7) ศึกษาการเขียนโปรแกรมผ่านเครือข่าย

(8) ออกแบบขั้นตอนการทำงานของโปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่ายคอมพิวเตอร์

(9) พัฒนาโปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่ายคอมพิวเตอร์

(10) ทดสอบและปรับปรุงโปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่ายคอมพิวเตอร์

บทที่ 2
โพรโตคอลทีซีพี/ไอพี

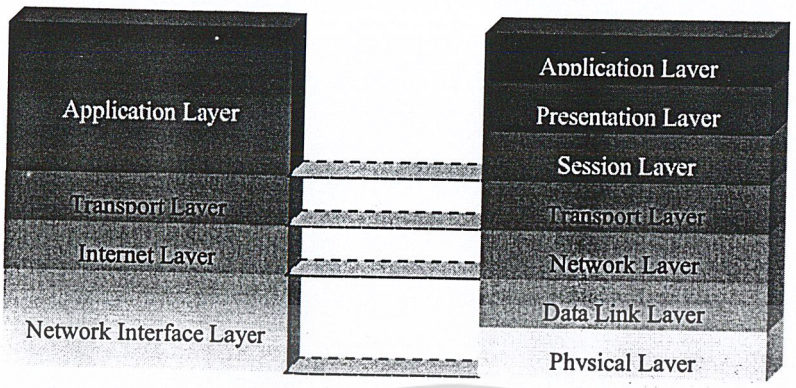
2.1 ความเป็นมาของโพรโตคอลทีซีพี/ไอพี

ทีซีพี/ไอพีเป็นมาตรฐานการรับส่งข้อมูลระหว่างคอมพิวเตอร์สองระบบที่มีขึ้นเมื่อกระทรวงกลาโหมสหรัฐฯหรือ Department Of Defense (DOD) ทำการทดลองในปี ค.ศ.1969 เชื่อมโยงคอมพิวเตอร์ ทางทหารของแต่ละหน่วย ซึ่งเป็นคอมพิวเตอร์ต่างชนิดกันให้สามารถติดต่อรับส่งข้อมูลกันได้ โครงการนี้มีชื่อว่า Advanced Research Projects Agency Network หรือ ARPANET ซอฟต์แวร์ที่ใช้ควบคุมการรับส่งข้อมูลของ ARPANET ประกอบด้วยส่วนหลักๆ 2 ส่วน คือ ทีซีพี (Transmission Control Protocol หรือ TCP) และ ไอพี (Internet Protocol หรือ IP) ซึ่งทีซีพี มีหน้าที่ตรวจสอบการรับส่งข้อมูลระหว่างคอมพิวเตอร์ผู้รับและผู้ส่ง ให้ได้รับข้อมูลถูกต้องครบถ้วน ส่วนไอพี จะมีหน้าที่เลือกเส้นทางที่ใช้รับส่งข้อมูลผ่านระบบเครือข่าย และตรวจสอบที่แอดเดรสของผู้รับ ซึ่งเรียกว่า ไอพีแอดเดรส (IP Address) ต่อมาในปี ค.ศ.1983 ทีซีพี/ไอพี ถูกกำหนดให้เป็นมาตรฐานการรับส่งข้อมูลของกระทรวงกลาโหมสหรัฐฯ จึงถือว่า ทีซีพี/ไอพี มีต้นกำเนิดมาจากโครงการ ARPANET นั่นเอง และต่อมาก็ได้ถูกรวมเป็นส่วนหนึ่งของระบบปฏิบัติการ UNIX และได้ถูกใช้กันอย่างแพร่หลายต่อมาซึ่งปัจจุบันใช้งานอยู่ในแทบทุกเครือข่าย ไม่ว่าจะเป็นเครือข่ายเฉพาะที่หรือเครือข่ายในบริเวณกว้าง ทีซีพี/ไอพีเชื่อมกลุ่มเครือข่ายย่อยเข้าด้วยกันเป็นเครือข่ายขนาดใหญ่ หรือ อินเทอร์เน็ต (Internet)

ทีซีพี/ไอพี ผ่านการออกแบบให้เป็นอิสระจากชนิดคอมพิวเตอร์ ฮาร์ดแวร์ และระบบปฏิบัติการ กลไกของโพรโตคอล มีความเชื่อถือได้สูงและทำงานได้แม้ในบางภาวะที่การสื่อสารมีความผิดปกติ รวมทั้งสามารถเลือกเส้นทางส่งข้อมูลตามสภาพเครือข่ายได้ในกรณีที่บางเส้นทางชำรุด โดย ทีซีพี/ไอพี ไม่ได้เป็นเพียงสอง โพรโตคอล ที่มีอยู่เท่านั้น หากแต่ยังมีโพรโตคอลสนับสนุนอีกเป็นจำนวนมาก และจัดรวมกันเป็นชุดโพรโตคอล ทีซีพี/ไอพี

2.2 การเชื่อมต่อของโพรโตคอลทีซีพี/ไอพี (TCP/IP Linking)

ทีซีพี/ไอพี (TCP/IP หรือ Transmission Control Protocol/Internet Protocol) เป็นโพรโตคอลในการสื่อสารในระบบอินเทอร์เน็ตและอินทราเน็ต การทำงานของทีซีพี/ไอพีสามารถเปรียบเทียบกับโมเดลอ้างอิงโอเอสไอ (Open System Interconnection Reference Model: OSI) ตามมาตรฐานไอเอสโอ (International Organization for Standardization: ISO) ได้ดังรูปที่ 2-1

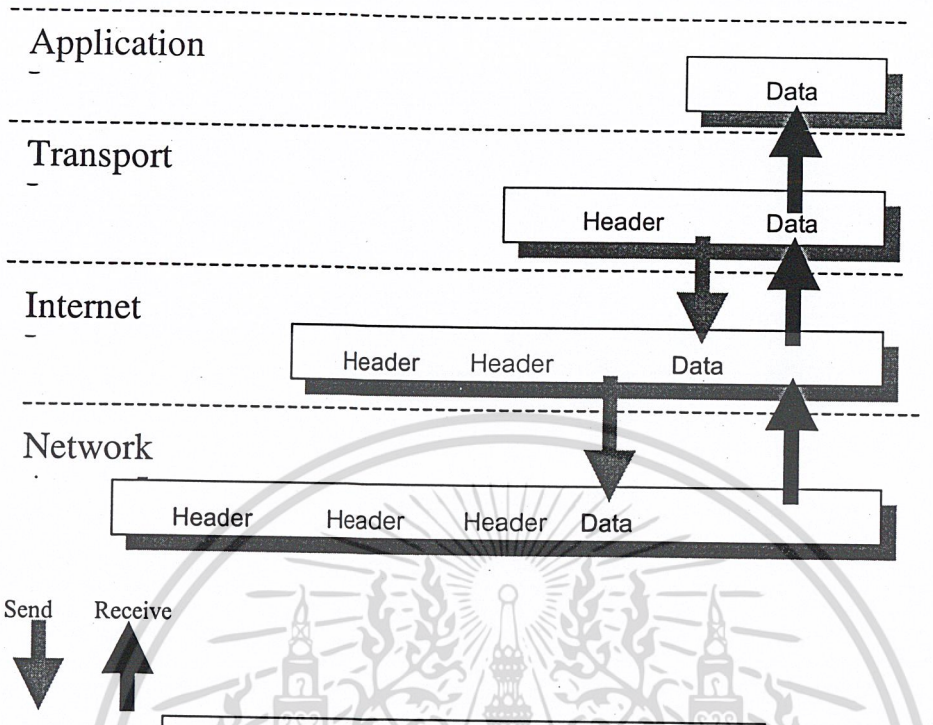


รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอ

ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชันจนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังตารางที่ 2-1

ชื่อระดับชั้น	หน้าที่
1. ชั้นแอพพลิเคชัน (Application Layer)	ชั้นนี้รองรับการทำงานของแอพพลิเคชันต่างๆ ที่ทำงานเป็นโพรเซสอยู่ในเครื่องต้นทางและปลายทาง โดยจัดการเชื่อมต่อระหว่างโพรเซส หรือแอพพลิเคชันที่อยู่ต่างเครื่องกัน โดยการทำงานของแอพพลิเคชันต่างๆมีการติดต่อกันตามแต่ละโพรโตคอลเฉพาะแล้วแต่แอพพลิเคชันที่ใช้งาน ซึ่งจะขอบริการจากชั้นทรานสปอร์ตอีกทีหนึ่ง
2. ชั้นทรานสปอร์ต (Transport Layer)	มีการสร้างการเชื่อมต่อกันระหว่างแอพพลิเคชันแบบ end-to-endโดยจุดที่เชื่อมต่อกันเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ต (port) หรือซ็อกเก็ต (Socket) ในชั้นนี้มีบริการหลักอยู่ 2 แบบ คือ Connection Oriented โดยเรียกผ่านโพรโตคอลทีซีพี(TCP: Transmission Control Protocol) และ Connectionless ซึ่งเรียกผ่านโพรโตคอลยูดีพี (UDP: User Datagram Protocol) ซึ่งกล่าวถึงในหัวข้อถัดไป
3. ชั้นอินเทอร์เน็ต (Internet Layer)	ชั้นนี้มีหน้านี้ส่งผ่านข้อมูลระหว่างเครือข่าย โดยมีโพรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆ ในอินเทอร์เน็ต คือ ไอพี (Internet Protocol: IP) ซึ่งกล่าวถึงในหัวข้อถัดไป นอกจากนี้ในชั้นนี้ยังมีโพรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ ไอซีเอ็มพี (Internet Control Message Protocol: ICMP) และเออาร์พี (Address Resolution Protocol: ARP)
4. ชั้นเน็ตเวิร์กอินเทอร์เฟส(Network Interface Layer)	ทำหน้าที่ในการแปลงข้อมูลให้อยู่ในรูปที่เหมาะสมกับเครือข่ายแต่ละแบบ ซึ่งแตกต่างกันออกไป และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่าย

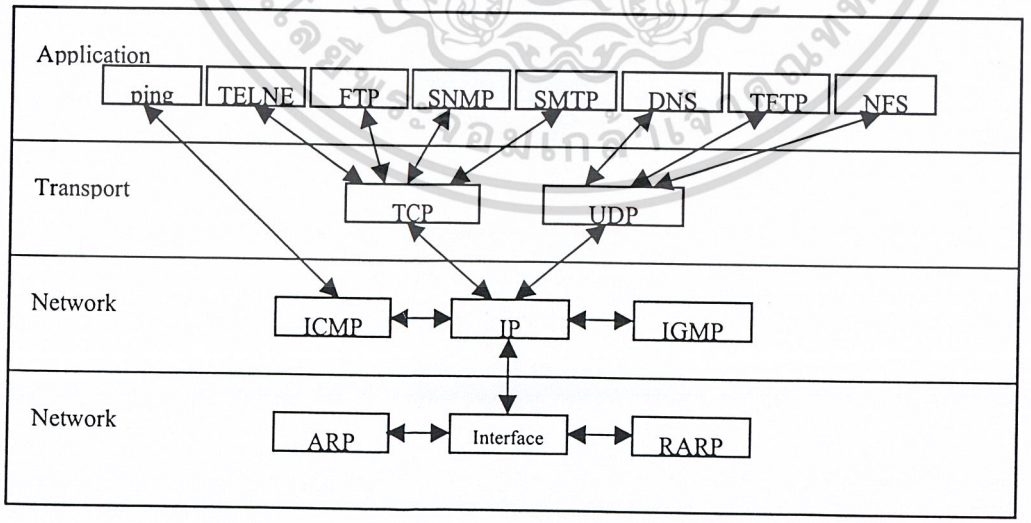
ตารางที่ 2-1 การทำงานของแต่ละระดับชั้นของทีซีพี/ไอพี



รูปที่ 2-2 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี

2.3 โพรโตคอลสแต็ก

การทำงานตามโปรแกรมประยุกต์หนึ่งๆไม่ได้ใช้โพรโตคอลพร้อมกันทั้งชุด หากแต่ใช้เพียงโพรโตคอล ที่สัมพันธ์กันไปในแต่ละระดับชั้นของแบบอ้างอิง ตัวอย่างเช่น เทลเน็ต (Telnet) จะอาศัยทีซีพีและไอพี ตามลำดับ การซ้อนทับของ โพรโตคอล จากระดับชั้นบนไปชั้นล่างเรียกว่า โพรโตคอลสแต็ก (Protocol Stack) ดังรูปที่ 2-3



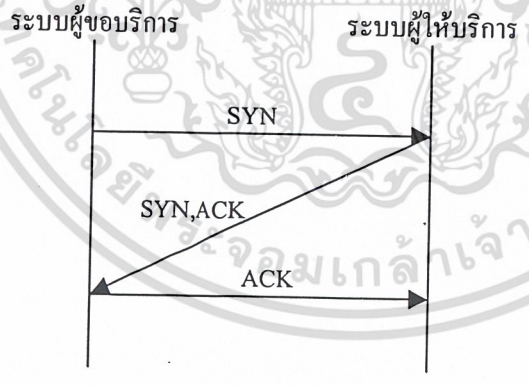
รูปที่ 2-3 โพรโตคอลสแต็กของทีซีพี/ไอพี

ไอพีซึ่งอยู่ในระดับชั้นเน็ตเวิร์กตามรูป เป็นแกนสำคัญของโพรโตคอลสแต็ก เนื่องจากทั้ง ทีซีพี และ ยูดีพี ต้องใช้ไอพีเพื่อเลือกเส้นทางส่งแพ็กเก็ต ในระดับชั้นเน็ตเวิร์กยังมีไอซีเอ็มพีสนับสนุนการทำงานของไอพีเพื่อรายงานข้อผิดปกติที่เกิดขึ้นเนื่องจากการส่งแพ็กเก็ต และมีไอจีเอ็มพีดูแลการจัดกลุ่มโฮสต์ในเครือข่ายมัลติคาสต์ ระดับชั้นทรานสปอร์ตมี 2 โพรโตคอลที่สำคัญ คือ ทีซีพีและยูดีพี แอพลิเชันจะเลือกใช้ทีซีพีหรือยูดีพีตามลักษณะงาน โพรโตคอลระดับล่างถัดจากไอพีได้แก่โพรโตคอลระดับ เน็ตเวิร์กอินเตอร์เฟสซึ่งกำหนดการทำงานตามเทคโนโลยีเครือข่ายที่ใช้งาน ในระดับชั้นนี้มีโพรโตคอลในชุดของ ทีซีพี/ไอพี ทำหน้าที่สนับสนุนการทำงานอยู่สอง โพรโตคอล คือ เออาร์พีและอาร์เออาร์พีทั้งสองโพรโตคอลทำหน้าที่แปลงค่าระหว่างแอดเดรสไอพี กับ ฮาร์ดแวร์แอดเดรส

ในชุดโพรโตคอลทีซีพี/ไอพีนี้ มีโพรโตคอลหลักที่ขอกล่าวถึง 5 โพรโตคอล ได้แก่ โพรโตคอลทีซีพี โพรโตคอลยูดีพี ซึ่งทำงานในชั้นทรานสปอร์ต และโพรโตคอลไอพี โพรโตคอลเออาร์พี โพรโตคอลไอซีเอ็มพี ซึ่งทำงานในชั้นอินเทอร์เน็ต โดยมีรายละเอียดดังต่อไปนี้

2.4 โพรโตคอลทีซีพี (TCP: Transmission Control Protocol)

การทำงานที่สำคัญอย่างหนึ่งของโพรโตคอลทีซีพี คือ การทำ “3-way Handshake” ซึ่งเป็นกระบวนการเริ่มต้นในการสร้างการเชื่อมต่อในชั้นทรานสปอร์ต กล่าวคือ ในการติดต่อกันระหว่างระบบในเครือข่ายต้องมีการสร้างการเชื่อมต่อไปยังระบบที่ให้บริการก่อน โดยผู้ขอบริการส่งสัญญาณ SYN เพื่อขอบริการ จากนั้นผู้ให้บริการจะส่งสัญญาณ ACK เพื่อตอบรับการเชื่อมต่อที่ร้องขอมาจึงสามารถรับส่ง ข้อมูลกันได้ ดังรูปที่ 2-5



รูปที่ 2-4 แสดงการทำ 3-ways Handshake

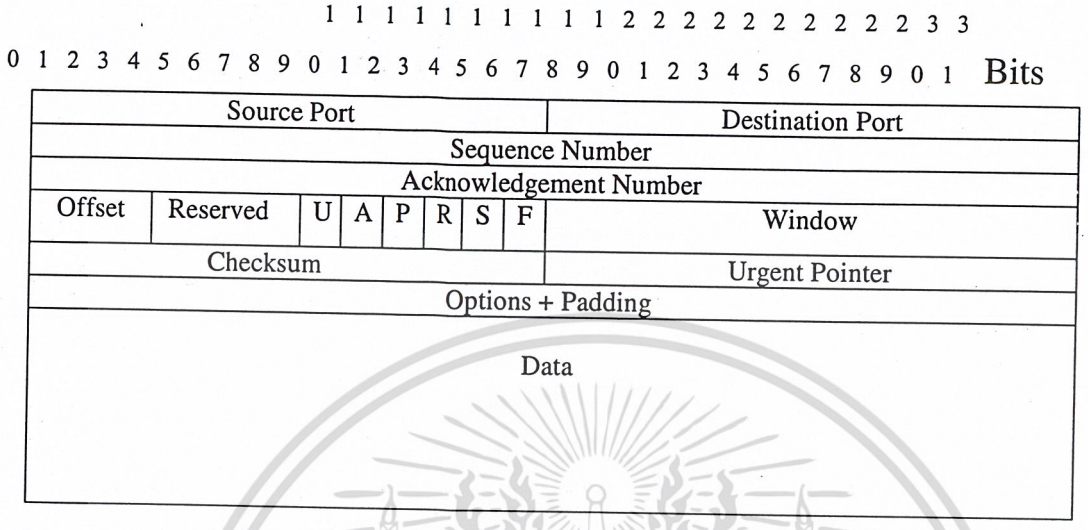
การเชื่อมต่อแบบ 3-way handshake นี้ เป็นการตรวจสอบความพร้อมของทั้งฝ่ายส่งและฝ่ายรับ และการกำหนดค่าเริ่มต้นของพารามิเตอร์ต่างๆ ของทั้งสองฝ่ายให้ตรงกัน หลังจากกระบวนการทำ 3-way handshake สิ้นสุด ทั้งสองฝ่ายจึงสามารถรับและส่งข้อมูลซึ่งกันและกันได้

ดังนั้นโพรโตคอลทีซีพีจึงเป็นโพรโตคอลที่มีการรับส่งข้อมูลแบบ “Connection Oriented” ทำให้การทำงานของทีซีพีมีความน่าเชื่อถือมากขึ้น หน้าที่การทำงานของทีซีพีในการรับส่งข้อมูลมีหน้าที่หลัก 6 ข้อคือ

1. ควบคุมการรับส่งข้อมูล (Basic Data Transfer)
2. ความน่าเชื่อถือในการรับส่งข้อมูล (Reliability)
3. ควบคุมการไหลของข้อมูล (Flow Control)
4. การทำมัลติเพล็กซ์ (Multiplexing)
5. ควบคุมการเชื่อมต่อ (Connection)
6. ความปลอดภัยในการรับส่งข้อมูล (Security)

<u>ส่วนประกอบของทีซีพีเฮดเดอร์</u>

1. Source Port : เป็นหมายเลขพอร์ตของบริการที่เครื่องต้นทาง
2. Destination Port : เป็นหมายเลขพอร์ตของบริการเครื่องปลายทาง
3. Sequence Number : เป็นหมายเลขที่บอกลำดับของการรับส่งข้อมูลของเครื่องที่ต้องการขอส่งข้อมูล
4. Acknowledgement Number : เป็นหมายเลขที่บอกลำดับของการรับส่งข้อมูลที่ฝั่งรับข้อมูล ปกติ ค่าของ Acknowledgement Number มีค่าเท่ากับ Sequence Number (ของอีกฝั่งหนึ่ง) + 1 เสมอ
5. Data Offset : เป็นตัวบอกค่าออฟเซตของข้อมูล เพราะทีซีพีนั้นไม่มีการกำหนดความยาวที่แน่นอนของข้อมูล จึงต้องมีออฟเซตเป็นตัวบอก
6. Flag : เป็นบิตที่บอกชนิดของข้อมูล ได้แก่
   - URG : Urgent Pointer Field Significant - แสดง Urgent Pointer
   - ACK : Acknowledgement Field Significant – แสดงการ Acknowledgement
   - PSH : Push Function
   - RST : Reset The Connection - แสดงเมื่อรีเซตการเชื่อมต่อ
   - SYN : Synchronize Sequence Number - หมายเลขแพ็กเก็ตที่ส่งแบบซิงโครนัส
   - FIN : No more data from sender - แสดงว่าไม่มีข้อมูลที่ส่งจากผู้ส่งแล้ว
7. Window : เป็นเลขบอกจำนวนของอ็อกเต็ต (octet) ของข้อมูลจัดการในส่วนของ end-to-end flow control
8. Checksum : เป็นส่วนที่ตรวจสอบความถูกต้องของข้อมูล
9. Urgent Pointer : เป็นตัวชี้ตำแหน่งของ Urgent Data
10. Option and Padding : เป็นตัวบอกออปชันของโพรเซสที่ใช้ทีซีพี
11. Data : เนื้อข้อมูลที่ต้องการสื่อสาร มีขนาดได้ไม่ต่ำกว่า 5 32-บิตเวิร์ด (6 บิตแรกสงวนไว้ และกำหนดให้เป็นศูนย์)



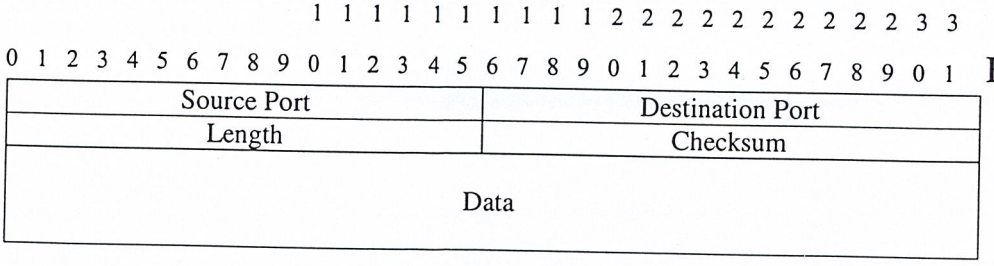
รูปที่ 2-5 แสดงแพ็กเก็ตทีซีพี

2.5 โพรโตคอลยูดีพี (UDP: User Datagram Protocol)

โพรโตคอลยูดีพีเป็นโพรโตคอลในการติดต่อสื่อสารในชั้นทรานสปอร์ต (Transport Layer) การทำงานคล้ายกับทีซีพีมาก คือ จัดการเกี่ยวกับการสื่อสารระหว่างเครื่อง แต่เป็นแบบคอนเนคชันเลส (Connectionless) คือ ทั้งฝ่ายส่งและฝ่ายรับไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกันโดยไม่ต้องมีการแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโพรโตคอลทีซีพี และไม่มีการส่งสัญญาณตรวจสอบว่า ข้อมูลถึงเครื่องปลายทางอย่างถูกต้องครบถ้วนในการส่งข้อมูลแต่ละครั้ง จึงไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล

<u>ส่วนประกอบของ UDP Frame</u>

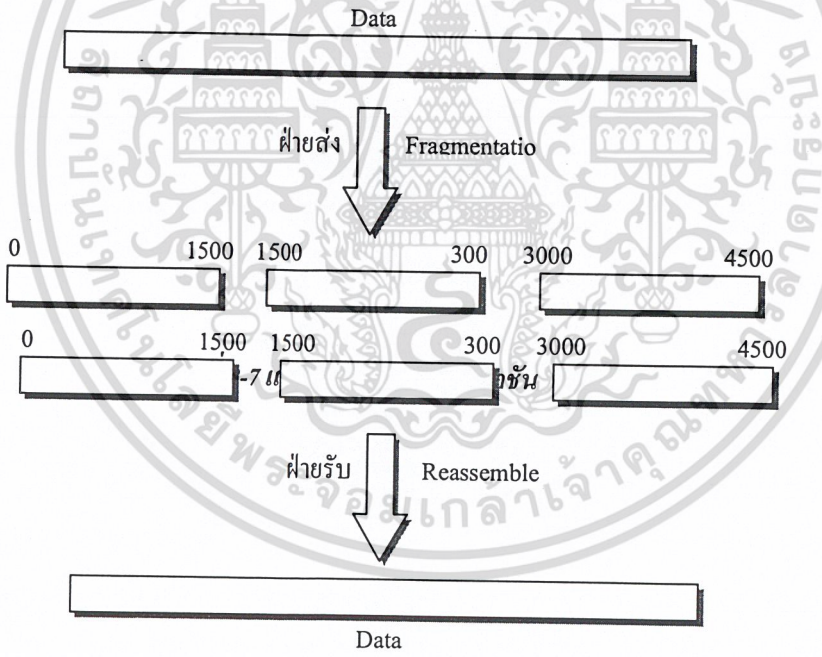
1. Source Port : เป็นค่าตัวเลข 16 บิต บอกพอร์ตของบริการที่เครื่องต้นทาง
2. Destination Port : เป็นค่าตัวเลข 16 บิต บอกพอร์ตของบริการที่เครื่องปลายทาง
3. Length : เป็นค่าตัวเลข 16 บิต บอกความยาวของข้อมูล
4. Checksum : เป็นค่าตัวเลข 16 บิต ตรวจสอบความถูกต้องของข้อมูลที่ส่ง



รูปที่ 2-6 แสดงแพ็กเก็ตยูดีพี

2.6 โพรโตคอลไอพี (IP: Internet Protocol)

โพรโตคอลไอพีเป็นโพรโตคอลที่จัดการเกี่ยวกับแอดเดรสของแต่ละแพ็กเก็ต เพื่อให้ส่งแพ็กเก็ตต่างๆ ไปยังเป้าหมายได้ถูกต้อง การทำงานของไอพีเป็นเพียงการส่งข้อมูลไปยังเครื่องเป้าหมายเท่านั้น ไม่มีการส่งสัญญาณขอบริการ หรือสัญญาณให้บริการระหว่างกันเหมือนทีซีพี เรียกว่าการเชื่อมต่อแบบคอนเนกชันเลส ซึ่งระบบทั้งสองตั้งสมมติฐานว่า การเชื่อมต่อระหว่างกันไม่มีความผิดพลาดเกิดขึ้น แน่เนื่องจากมาตรฐานในเครือข่ายมีหลากหลาย ขนาดของแพ็กเก็ตในแต่ละมาตรฐานจึงมีความแตกต่างกันออกไป ทำให้การส่งข้อมูลระหว่างอุปกรณ์ในเครือข่ายนั้นอาจมีการแบ่งข้อมูลออกเป็น แพ็กเก็ตย่อยๆ ในระหว่างการส่ง เรียกว่า การทำแฟร็กเมนเตชัน (Fragmentation) เช่น แพ็กเก็ตของ FDDI มีขนาด 4,500 ไบต์ หากเครื่องปลายอยู่ในเครือข่าย อีเทอร์เน็ต (Ethernet) ซึ่งมีขนาดของแพ็กเก็ตสูงสุดเพียง 1,500 ไบต์ ดังนั้นการส่งแพ็กเก็ตไปยังเครื่องปลายทางจึงต้องมีการแบ่งเป็นแพ็กเก็ตย่อย และเมื่อแพ็กเก็ตย่อยมาถึงเครื่องเป้าหมายก็จะมารวมกันเป็นแพ็กเก็ตเดิมที่มีขนาด 4,500 ไบต์อีกครั้ง เรียกการรวมกันนี้ว่า การ รีแอสเซมเบิ้ล (Reassemble) ซึ่งทำให้ได้ข้อมูลเหมือนที่ส่งมาจากเครื่องต้นทาง



รูปที่ 2-7 แสดงการรีแอสเซมเบิ้ล

ส่วนประกอบของแพ็กเก็ตไอพี

1. version : เป็นค่าตัวเลข 4 บิต บอกเวอร์ชันของมาตรฐานไอพีที่ใช้ โดยปกติมีค่าเป็น 4 ซึ่งหมายถึง IPv4

2. Internet Header Length (IHL) : เป็นตัวบอกความยาวเฮดเดอร์ของไอพี

3. Type of Service : เป็นส่วนที่บอกการทำงานของแพ็กเก็ตที่ส่งว่าทำหน้าที่อะไร มีทั้งหมด 8 บิต โดย

Bit 0-2 : บอกรายละเอียดการทำงานของแพ็กเก็ตนั้นๆ

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

Bit 3 : บอกถึงลักษณะของดีเลย์ (delay)

0 = Normal Delay - มีดีเลย์ปกติ

1 = Low Delay - มีดีเลย์ต่ำ

Bit 4 : บอกถึงประเภทของทรูพุต

0 = Normal Throughput – มีทรูพุตปกติ

1= High Throughput – มีทรูพุตสูง

Bit 5 : บอกถึงประเภทของความน่าเชื่อถือ

0 = Normal Reliability - มีความน่าเชื่อถือพอประมาณ

1 = High Reliability - มีความน่าเชื่อถือสูง

Bit 6-7 : กันไว้ใช้ในอนาคต

4. Total Length : มีขนาด 16 บิต บอกถึงความยาวในดาต้าแกรมของไอพี

5. Identification field : เป็นตัวเลข 16 บิต เป็นค่าประจำตัวของไอพีนั้น โดยโฮสต์ที่ส่งเป็นผู้กำหนด และเพิ่มค่าขึ้นหนึ่งเมื่อมีการส่งดาต้าแกรมของไอพีใหม่ ซึ่งใช้ในการประกอบกลับ

6. Flag : เป็นตัวเลข 3 bit บอกลักษณะของแพ็กเก็ตว่ามีการแฟร็กเมนต์หรือไม่

Bit 0 : สงวนไว้ ปกติเป็น 0

Bit 1 : 0 = บอกว่าแพ็กเก็ตมีการแตกแพ็กเก็ตย่อย

1 = บอกว่าแพ็กเก็ตไม่มีการแตกแพ็กเก็ตย่อย

Bit 2 : 0 = บอกว่าแพ็กเก็ตนั้นเป็นแพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

1 = บอกว่าแพ็กเก็ตนั้นยังไม่ใช่แพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

7. Fragment Offset : เป็นค่าตัวเลข 13 บิต บอกออฟเซต(offset)ของแฟร็กเมนต์เมื่อเทียบในดาต้าแกรม

8. Time To Live (TTL) : เป็นตัวเลข 8 บิต บอกช่วงเวลาของแพ็กเก็ตที่ยังอยู่ในเครือข่ายได้ โดยกำหนดค่าเป็นจำนวนเราเตอร์(router)สูงสุดที่ดาต้าแกรมผ่านได้ ซึ่งโดยทั่วไปทีค่าระหว่าง 32 ถึง 64 และลดค่าลงเรื่อยๆ เมื่อผ่านเราเตอร์ เพื่อเป็นการป้องกันแพ็กเก็ตล้นเครือข่าย

9. Protocol : เป็นตัวเลข 8 bit บอกถึงโพรโตคอลที่อยู่เหนือขึ้นไป ว่าเป็นโพรโตคอลระดับสูงกว่าประเภทใด

10.Header Checksum : เป็นค่าตัวเลข 32 บิต ใช้ตรวจสอบความถูกต้องของเฮดเดอร์

11.Source Address : เป็นค่าตัวเลข 32 บิต บอกถึงไอพีแอดเดรสของเครื่องต้นทาง

12.Destination Address : เป็นค่าตัวเลข 32 บิต บอกถึงไอพีแอดเดรสของเครื่องปลายทาง

Bit 0: สงวนไว้ ปกติเป็น 0

Bit 1: 0 = บอกว่าแพ็กเก็ตมีการแตกแพ็กเก็ตย่อย

1 = บอกว่าแพ็กเก็ตไม่มีการแตกแพ็กเก็ตย่อย

Bit 2: 0 = บอกว่าแพ็กเก็ตนั้นเป็นแพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

1 = บอกว่าแพ็กเก็ตนั้นยังไม่ใช่แพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 Bits

Ver	IHL	Type of Service	Total Length	
Identifier			Flags	Fragment
Time to Live		Protocol	Header Checksum	
Source Address				
Destination Address				
Options + Padding				
Data				

2.7 โพรโตคอลเออาร์พี(ARP:Address Resolution Protocol)

โพรโตคอลเออาร์พีเป็นโพรโตคอลที่ออกแบบมาเพื่อใช้ในเครือข่ายที่สนับสนุนการบอร์ดคาสต์ ถูกเรียกใช้งานโดยโพรโตคอลไอพีเพื่อช่วยแปลงหมายเลขไอพี ไปเป็นหมายเลขฮาร์ดแวร์ปลายทาง ตัวอย่างเช่น เว็บเซิร์ฟเวอร์เครื่องหนึ่งเชื่อมต่ออยู่ในเครือข่ายอินเทอร์เน็ต และในการเชื่อมต่อนี้ต้องอาศัยการ์ดแลน(LAN card) ติดตั้งอยู่ ที่แลนการ์ดนี้จะมีหมายเลขเฉพาะประจำฮาร์ดแวร์ที่ไม่ซ้ำกับใคร เพื่อใช้อ้างอิงการส่งข้อมูลในเครือข่าย แต่เมื่อมาใช้งานใน โพรโตคอล ทีซีพี/ไอพี ก็ต้องมีการกำหนดหมายเลขแอดเดรสไอพี ประจำตัวเพื่อใช้อ้างอิงกัน และโพรโตคอลเออาร์พี จะทำหน้าที่แปลงค่าหมายเลขไอพีให้เป็นหมายเลขฮาร์ดแวร์จริงในระดับการทำงานที่ชั้นอินเทอร์เน็ตนี้ ซึ่งกลไกการแปลงนี้เรียกว่าแอดเดรสรีโซลูชัน (address resolution)

Header	ARP/RARP datagram	FCS

hardware		protocol
HLEN	PLEN	operation
Sender HA (octets 0-3)		
Sender HA (octets 4-5)		Sender IA (octets 0-1)
Sender IA (octets 2-3)		Target HA (octets 0-1)
Target HA (octets 2-5)		
Target IA (octets 0-3)		

รูปที่ 2-9 เออาร์พีดาทาแกรม

ส่วนประกอบของเออาร์พีดาทาแกรม

1. Hardware 16 *บิต* : กำหนดชนิดของฮาร์ดแวร์เครือข่ายที่เออาร์พีทำงานอยู่ ค่าใช้งานมีตัวอย่างดังต่อไปนี้

   1 อีเทอร์เน็ต

   4 โทเค็นริง

   5 เคออส(chaos)

   6 เครือข่าย IEEE 802

   7 อาร์คเน็ต

   12 โลคัลทอล์ค

2. protocol 16 *บิต* : ชนิดของโพรโตคอลที่ร้องขอใช้เออาร์พี

3. HLEN 8 *บิต* : ขนาดของฮาร์ดแวร์แอดเดรสเป็นจำนวนไบต์ ค่าปกติที่ใช้งาน คือ 6 ซึ่งเท่ากับขนาด 6 ไบต์ของอีเทอร์เน็ตฮาร์ดแวร์แอดเดรส

4. PLEN *8 บิต* : ขนาดของแอดเดรสระดับเน็ตเวิร์กเป็นจำนวนไบต์ ค่าปกติที่ใช้ คือ 4 ซึ่งเท่ากับขนาด 4 ไบต์ของไอพีแอดเดรส

5. Operation 16 *บิต* : กำหนดรูปแบบการใช้ดาทาแกรม ค่าในฟิลด์นี้ใช้กำหนดการทำงานของทั้งเออาร์พีและอาร์เออาร์พี ซึ่งมี 4 ค่า คือ

   ARP request (ค่าเท่ากับ 1)

   ARP reply (ค่าเท่ากับ 2)

   RARP request (ค่าเท่ากับ 3)

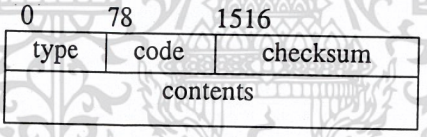
   RARP reply (ค่าเท่ากับ 4)

6. Address : ฟิลด์แอดเดรสเรียงลำดับจากฮาร์ดแวร์และเน็ตเวิร์กแอดเดรสของสถานีที่ร้องขอตามด้วยฮาร์ดแวร์และเน็ตเวิร์กแอดเดรสของสถานีที่ตอบรับ

2.8 โพรโตคอล ไอซีเอ็มพี (ICMP : Internet Control Message Protocol)

หน้าที่หลักของ โพรโตคอล ไอซีเอ็มพี คือการแจ้งหรือแสดงข้อความจากระบบ เพื่อบอกให้ผู้ใช้ทราบว่าเกิดอะไรขึ้นในการส่งผ่านข้อมูลนั้น ซึ่งปัญหาส่วนมากที่พบ คือส่งไปไม่ได้ หรือปลายทางรับข้อมูลไม่ได้ เป็นต้น นอกจากนี้โพรโตคอลไอซีเอ็มพี

ยังถูกเรียกใช้งานจากเครื่องเซิร์ฟเวอร์ และเราเตอร์ อีกด้วย เพื่อแลกเปลี่ยนข้อมูลที่ใช้ควบคุม ส่วนรูปแบบการทำงานของโพรโตคอลไอซีเอ็มพีนั้นจะทำงานควบคู่กับโพรโตคอลไอพีในระดับเดียวกัน และข้อความต่างๆที่แจ้งให้ทราบจะถูกผนึกอยู่ภายในข้อมูลของไอพี(ไอพีดาทาแกรม)อีกทีหนึ่ง ข้อความที่โพรโตคอลไอซีเอ็มพีส่งนั้น แบ่งออกได้ 2 แบบ คือ ICMP error message หรือข้อความแจ้งข้อผิดพลาด และ ICMP query หรือข้อความเรียกขอข้อมูลเพิ่ม ตัวอย่างกลไกการทำงานของโพรโตคอลไอซีเอ็มพี เช่น เมื่อมีการส่งผ่านข้อมูลจากผู้ใช้ไปยังปลายทางที่ไม่ถูกต้อง หรือขณะนั้นเครื่องปลายทางเกิดปัญหาจนไม่สามารถรับข้อมูลได้ที่เราเตอร์จะส่งข้อความแจ้งเป็นไอซีเอ็มพีเมสเซจ (ICMP message) ที่ชื่อ destination unreachable ให้กับผู้ส่งข้อมูล นอกจากนี้ตัว ข้อมูลที่แจ้งข้อความก็จะมีส่วนของข้อมูลไอพีดาทาแกรมที่เกิดปัญหาด้วยดังนั้นเมื่อผู้ส่งข้อมูลได้รับข้อความแจ้งแล้วก็จะได้ทราบว่าจุดที่เกิดปัญหานั้นอยู่ที่ใดดังนั้นโพรโตคอลไอซีเอ็มพี จึงกลายมาเป็นเครื่องมืออย่างหนึ่งในการช่วยทดสอบเครือข่าย เช่น คำสั่ง Ping ที่เรามักใช้ทดสอบว่าเครื่องเซิร์ฟเวอร์ที่ให้บริการหรืออุปกรณ์ที่ต่ออยู่ในเครือข่าย อินเทอร์เน็ตนั้นยังทำงานเป็นปกติหรือไม่ แล้วคำสั่ง ping มีการเรียกใช้งานโพรโตคอล ไอซีเอ็มพี แจ้งเป็นข้อความให้ทราบอีกต่อหนึ่ง



รูปที่ 2-10 ฟอร์แมตของ ไอซีเอ็มพี

1. Type *ขนาด 8 บิต* : กำหนดค่าความผิดพลาดและการรายงานสถานะ การใช้งานในปัจจุบันมีทั้งหมด 15 ประเภท
2. code *ขนาด 8 บิต* : รหัสความผิดพลาดย่อย
3. Checksum *ขนาด 16 บิต* : ค่าผลรวมตรวจสอบแบบ 1's complement สำหรับใช้ตรวจสอบความผิดพลาด โดยคำนวณผลรวมของ type,code และ contents
4. Contents *ขนาดไม่คงที่* : ฟิลด์นี้ใช้บรรจุข้อมูลข่าวสารเพิ่มเติมเพื่อแจ้งกลับซึ่งจะขึ้นอยู่กับค่า typeและ code

2.9 ข้อบกพร่องของทีซีพีไอพี

2.9.1 ขาดกลไกด้านความปลอดภัย

สำหรับทีซีพีไอพีแล้วหากมีการกระตุ้นที่ถูกต้องตามโพรโตคอลแล้วจะต้องตอบรับเสมอหรือถ้าก็คือหากถูกถามจะต้องตอบเสมอ โพรโตคอลจะกำหนดไว้ชัดเจนว่าหากมีการกระตุ้นที่ถูกต้องตามโพรโตคอลจะต้องตอบรับ โฮสต์ผู้รับจะทำหน้าที่ตรวจสอบความถูกต้องเพียงว่ามันถูกต้องตาม

โพรโตคอลหรือไม่เท่านั้น ไม่ได้กำหนดในเรื่องอื่นเช่น ปริมาณมากเกินไปหรือไม่ ต่อเนื่องหรือไม่ ไม่มีกลไกในการตรวจสอบผู้ถาม และการจัดการกับการถามที่ผิดปกติ เช่นในกรณีปกติ หากโฮสต์ของเราได้รับ ICMP Echo Request มาโฮสต์ของเราไม่สามารถตรวจสอบได้ว่าผู้ส่งมีสิทธิ์ได้รับคำตอบหรือไม่ หรือผู้ส่งถามซ้ำมากเกินไปแล้ว แต่สิ่งที่ทำได้ก็คือตอบกลับไปด้วย ICMP Echo Reply เท่านั้น และต้องตอบกลับไปทุกกรณี ทั้งนี้เพราะกลไกด้านความปลอดภัยไม่ได้มีอยู่ในโพรโตคอล

ลักษณะนี้ต่างกับในระบบปฏิบัติการและแอพพลิเคชั่นต่างๆที่มีกลไกด้านความปลอดภัยโดยจะอนุญาตให้ผู้มีสิทธิ์ในระบบเท่านั้นที่จะสามารถใช้ทรัพยากรของระบบได้ แต่ตราบใดที่โพรโตคอลยังคงต้องทำงานอยู่บนทีซีพีไอพีและอาศัยทีซีพีไอพี ในการสื่อสารข้อมูล ช่องว่างของทีซีพีไอพีก็ยังคงอยู่และไม่สามารถแก้ไขได้ด้วยการรักาความปลอดภัยในระดับที่สูงขึ้น

ด้วยช่องว่างนี้เองที่ถูกนำมาใช้ประโยชน์โดยผู้บุกรุก ไม่ว่าจะเป็นการสำรวจเป้าหมายด้วยการสแกนพอร์ต สแกนเน็ตเวิร์ก การ DoS ด้วย SYN Flood, Ping Flood เป็นต้น ซึ่งสามารถกระทำได้โดยที่โฮสต์แทบจะไม่สามารถป้องกันตัวเองได้เลย

2.9.2 การตอบรับเป็นสิ่งที่คาดหมายได้

แน่นอนที่สุดว่าการตอบรับจะต้องเป็นสิ่งที่สามารถคาดหมายได้ เพราะการตอบรับใดๆจะต้องเป็นตามข้อกำหนดในโพรโตคอลซึ่งเผยแพร่แก่สาธารณชนอยู่แล้ว หากการตอบสนองคาดหมายไม่ได้การสื่อสารสองทางก็คงไม่เกิดขึ้น ผู้บุกรุกจึงอาศัยการวิเคราะห์การตอบรับของเป้าหมายในสถานการณ์ต่างๆมาใช้ให้เป็นประโยชน์ถึงแม้ผู้ที่จะใช้ประโยชน์จากการตอบรับในลักษณะนี้ได้ต้องมีความเข้าใจในโพรโตคอลเป็นอย่างดีและค่อนข้างละเอียด เพราะต้องทราบว่าการกระตุ้นแต่ละชนิดจะได้รับคำตอบจากเป้าหมายอย่างไร การตอบรับแต่ละแบบมีความหมายอย่างไร แต่มิได้เกินความสามารถของผู้บุกรุกทั้งหลายในการได้มาซึ่งข้อมูลที่สำคัญของเป้าหมาย จะพบได้ว่าการกระตุ้นและวิเคราะห์เป้าหมายเทคนิคต่างๆจะเป็นขั้นตอนที่สำคัญของผู้บุกรุกส่วนใหญ่จำเป็นต้องใช้ เพราะเป็นช่องทางที่ไม่มีการรักษาความปลอดภัย ในปัจจุบันเครื่องมือต่างๆที่ใช้ในการบุกรุกเจาะระบบได้พัฒนาไปมาก มีผู้ผลิตโปรแกรมที่ทำหน้าที่กระตุ้นและวิเคราะห์เป้าหมายแพร่หลายอยู่ทั่วไป ช่วยให้การเจาะระบบและสำรวจเป้าหมายยุ่งยากน้อยลง

2.9.3 การตอบรับกำหนดไว้ไม่ครอบคลุมทุกเงื่อนไข

ด้วยเป้าหมายของโพรโตคอลในเบื้องต้นคือ ความถูกต้อง ความมีเสถียรภาพและประสิทธิภาพของการสื่อสาร ดังนั้นเงื่อนไขข้อกำหนดต่างๆ ที่ระบุไว้นั้นก็เป็นไปเพื่อรับใช้วัตถุประสงค์ดังกล่าว จุดสำคัญอยู่ที่หากมีการสื่อสารที่ถูกต้องตามโพรโตคอล คอมพิวเตอร์ทั้ง 2 ฝั่งจะต้องอำนวยความสะดวกให้ทำการการสื่อสารด้วยความถูกต้อง มีเสถียรภาพและประสิทธิภาพสูงสุด แต่โพรโตคอลกลับละเลยในส่วนนี้ หากมีการสื่อสารที่ไม่ถูกต้องตามโพรโตคอลแล้วจะจัดการต่อไปอย่างไร อาจจะมีกำหนดไว้ในจุดที่สำคัญ แต่ก็ยังคงเหลือเงื่อนไขอื่นๆ อีกมากมายที่โพรโตคอลไม่ได้ระบุ ตัวอย่างเช่น ทีซีพีกำหนดให้ส่ง SYN ในตอนสร้างการเชื่อมต่อและส่ง FIN ในตอนยกเลิกการติดต่อ ถามว่าหากมีการส่งทีซีพีที่มีทั้ง SYN และ FIN พร้อมกันจะเกิดอะไรขึ้น ผู้รับจะตอบรับกลับไปอย่างไร

แน่นอนว่าการส่งสัญญาณหรือข้อมูลใดๆ ที่ผิดข้อกำหนดในโพรโตคอลนั้นมิสามรถกระทำได้ในการทำงานปกติ เช่น หากคุณจะส่งข้อมูลผ่านทีซีพี คุณอาจจะเรียก API หรือซ็อกเก็ต มาทำงานและที่เหลือ API เหล่านั้นก็ไปจัดการข้อมูลในระดับล่างเองให้ถูกต้องตามโพรโตคอล แต่ API หรือ ซ็อกเก็ตเหล่านั้นก็เป็นแค่โปรแกรมชนิดหนึ่งเท่านั้นที่ทำหน้าที่จักระเบียบข้อมูลแล้วส่งไปยังอุปกรณ์ในระดับล่างเท่านั้น หากผู้บุกรุกมีโปรแกรมที่ทำหน้าที่ดังกล่าว ก็สามารถส่งข้อมูลไปยังอุปกรณ์ต่างๆ ได้โดยตรงเช่นกัน ถึงแม้อาจจะต้องมีความรู้ความรู้เรื่องภาษาระดับต่ำและฮาร์ดแวร์บ้างก็ตาม แต่ก็เป็นสิ่งที่ทำได้ และเมื่อเราสามารถส่งข้อมูลไปยัง ดีไวซ์(device)ได้ ข้อมูลที่ส่งก็ไม่จำเป็นต้องเป็นไปตามที่ถูกกำหนดไว้แล้วในโพรโตคอลเหมือนที่ ซ็อกเก็ต ทำข้อมูลอาจจะมีเงื่อนไข หรือแฟล็ก ที่แปลกประหลาดและไม่สามารถจัดการด้วยโพรโตคอลทั่วๆไปก็เป็นได้ และจุดนี้เองที่ทำให้ข้อมูลที่ส่งไปยังปลายทางบางครั้งหากไม่อยู่ในเงื่อนไขปกติที่ระบุไว้ในโพรโตคอล ผู้รับข้อมูลอาจจะไม่สามารถจัดการกับข้อมูลได้อย่างถูกต้อง และข้อมูลเหล่านั้นก็จะแปรสภาพเป็นเครื่องมือในการโจมตีได้ อย่างเช่น Windows NT ที่เคยเกิดปัญหาแฮงค์และขึ้นจอฟ้าทันทีเมื่อถูกโจมตีด้วยวิธีนี้

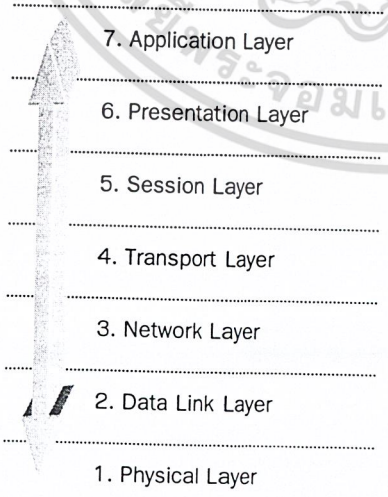
บทที่ 3
สถาปัตยกรรมเครือข่ายและระดับชั้นโพรโตคอล

3.1 แบบอ้างอิงโอเอสไอ (OSI Reference Model)

ในการกล่าวถึงระบบเครือข่ายคอมพิวเตอร์ เราจะต้องใช้คำที่มีความหมายเฉพาะเจาะจงเกี่ยวกับระบบสื่อสารข้อมูล ซึ่งแต่ละคนอาจใช้ไม่เหมือนกัน การมีสิ่งไว้อ้างอิงจึงเป็นสิ่งที่จำเป็นในการทำความเข้าใจความหมายของคำที่ใช้ในการสื่อสารข้อมูล โมเดลแบบสถาปัตยกรรม(Architectural model) ที่ถูกพัฒนาโดยองค์การมาตรฐานโลก (International Standards Organization: ISO) ได้ถูกใช้อย่างแพร่หลายในการอธิบายถึงโครงสร้างและหน้าที่ของโพรโตคอลในการสื่อสารข้อมูล โมเดลดังกล่าวนี้มีชื่อว่าแบบอ้างอิงการเชื่อมต่อของระบบเปิด (Open System Interconnection: OSI Reference Model)

โมเดลนี้ประกอบไปด้วยเลเยอร์จำนวน 7 ชั้นที่กำหนดหน้าที่ของโพรโตคอลการสื่อสารข้อมูล แต่ละเลเยอร์แสดงถึงหน้าที่เมื่อข้อมูลถูกส่งระหว่างแอพพลิเคชั่นที่ทำงานร่วมกันข้ามเครือข่าย ดังรูป 2-1 .ที่แสดงถึงแต่ละเลเยอร์พร้อมทั้งคำอธิบายเกี่ยวกับหน้าที่ของมันสั้น ๆ มักเรียกโครงสร้างนี้ว่าสแต็กหรือโพรโตคอลสแต็ก

หนึ่งเลเยอร์ไม่ได้กำหนดโพรโตคอลเดียวแต่กำหนดหน้าที่ในการสื่อสารข้อมูลที่สามารถทำได้โดยโพรโตคอลจำนวนเท่าไหร่ก็ได้ ดังนั้นแต่ละเลเยอร์จึงมีได้หลายโพรโตคอลซึ่งให้บริการที่เหมาะสมกับหน้าที่ของเลเยอร์นั้น ตัวอย่างเช่นโพรโตคอลไฟล์ทรานสเฟอร์และโพรโตคอลอิเลคทรอนิคส์เมล์ซึ่งทั้งสองให้บริการแก่ผู้ใช้ จึงเป็นส่วนหนึ่งของเลเยอร์แอพพลิเคชั่น



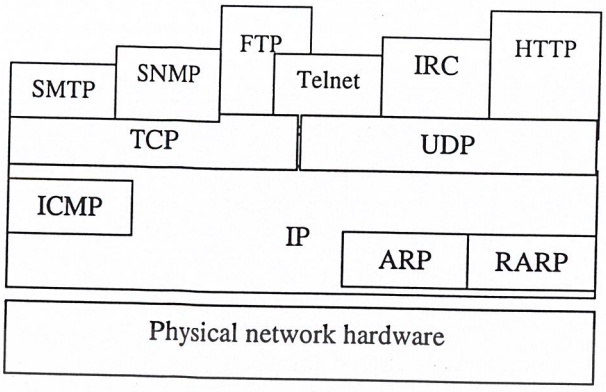
รูปที่ 3-1 แบบอ้างอิงโอเอสไอ

ทุกโพรโตคอลติดต่อสื่อสารกับเพียร์(peer) ของมัน เพียร์คือการนำโพรโตคอลเดียวกันในเลเยอร์ที่เทียบเท่ากันบนระบบไกล(remote system) มาใช้งานจริง เช่น โพรโตคอลไฟล์ทรานสเฟอร์ของเราคือเพียร์ของโพรโตคอลไฟล์ทรานสเฟอร์ที่ใช้อยู่อีกระบบหนึ่ง โดยนามธรรมแล้วแต่ละโพรโตคอลจะ เกี่ยวข้องกับการสื่อสารกับเพียร์ของมันโดยที่ไม่ต้องสนใจเลเยอร์ที่อยู่เหนือหรือต่ำกว่ามัน แต่การส่งข้อมูลจริง ๆ จะเกี่ยวข้องกับทุกเลเยอร์ที่จะต้องส่งข้อมูลจากแอพพลิเคชั่นที่เราใช้ไปยังแอพพลิเคชั่นที่เทียบเท่ากันในระบบไกล เลเยอร์ที่สูงกว่าจะเกี่ยวข้องกับเลเยอร์ที่ต่ำกว่าในการส่งข้อมูลไปบนเครือข่ายที่มีอยู่ ข้อมูลถูกส่งลงมาจากสแต็กจากเลเยอร์หนึ่งสู่เลเยอร์ถัดไป จนกระทั่งข้อมูลถูกส่งไปบนเครือข่ายโดยโพรโตคอลในเลเยอร์ฟิสิคัล(physical layer) ที่ปลายอีกข้างหนึ่งก็จะทำในลักษณะตรงกันข้ามคือข้อมูลจะถูกส่งไปบนสแต็กให้แก่แอพพลิเคชั่นที่ต้องการ แต่ละเลเยอร์ไม่จำเป็นต้องรู้ว่าเลเยอร์บนและล่างมันทำหน้าที่อะไร มันจำเป็นต้องรู้เฉพาะว่าทำอย่างไรจึงจะส่งข้อมูลไปให้ได้ การแบ่งหน้าที่ของการสื่อสารข้อมูลออกเป็นหลาย ๆ เลเยอร์จะช่วยลดผลกระทบที่จะต้องเปลี่ยนแปลงทั้งชุดโพรโตคอลเมื่อเทคโนโลยีเปลี่ยนไป ตัวอย่างเช่น เราสามารถเพิ่มแอพพลิเคชั่นเข้าไปใหม่โดยไม่ต้องเปลี่ยนเครือข่ายในระดับฟิสิคัล หรือฮาร์ดแวร์ใหม่สามารถติดตั้งได้โดยไม่ต้องเขียนซอฟต์แวร์ใหม่

แบบอ้างอิงโอเอสไอประกอบด้วยเลเยอร์ต่าง ๆ 7 ชั้นดังนี้

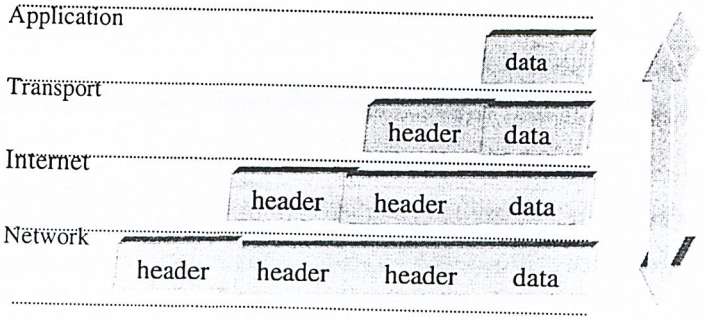
Application Layer	เป็นเลเยอร์ที่ติดต่อกับผู้ใช้ ประกอบด้วยแอพพลิเคชั่นโปรแกรมที่ใช้เครือข่าย
Presentation Layer	การที่แอพพลิเคชั่นจะทำงานร่วมกันได้จะต้องทำการตกลงว่าจะแทนข้อมูลกันอย่างไร เลเยอร์นี้จะกำหนดมาตรฐานการแทนข้อมูลให้
Session Layer	จัดการการติดต่อระหว่างแอพพลิเคชั่นที่ทำงานร่วมกัน
Transport Layer	รับประกันว่าผู้รับจะต้องได้รับข้อมูลอย่างถูกต้องครบถ้วน
Network Layer	จัดการการติดต่อข้ามเครือข่ายและแยกโพรโตคอลในเลเยอร์ที่สูงกว่าออกจากรายละเอียดของเครือข่าย
Data Link Layer	ดูแลความน่าเชื่อถือของการส่งข้อมูลข้ามเครือข่ายฟิสิคัล
Physical Layer	กำหนดคุณลักษณะของฮาร์ดแวร์ที่จำเป็นต้องใช้ในการพาสัญญาณการสื่อสารข้อมูล เช่น ระดับโวลเตจ จำนวนและตำแหน่งของพิน(pins) ที่ใช้ในการอินเทอร์เฟส ตัวอย่างมาตรฐานในเลเยอร์นี้คือ คอนเน็คเตอร์ในการอินเทอร์เฟส เช่น RS232C และ V.35 และมาตรฐานในการเชื่อมสายของเครือข่าย เช่น ไอ ทริปเปิ้ลอี 802.3

3.2 **ชุดโพรโตคอลทีซีพี/ไอพี** (TCP/IP Protocol suite)



รูปที่ 3-2 สถาปัตยกรรมทีซีพี/ไอพี

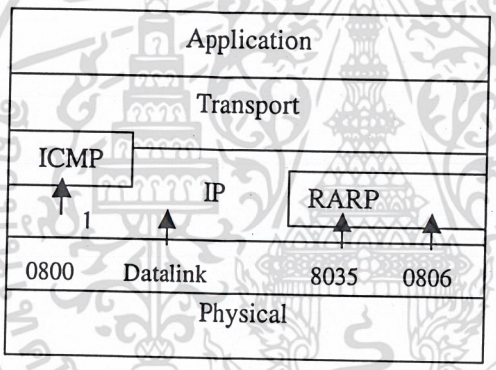
ประกอบด้วยเลเยอร์หลายชั้นเช่นเดียวกับแบบอ้างอิงโอเอสไอดังรูป 2-2 ถ้าพิจารณาจากหน้าที่ของแต่ละเลเยอร์แล้ว 4 เลเยอร์ล่างของทีซีพี/ไอพีสามารถนำมาเปรียบเทียบกับ 4 เลเยอร์ล่างของแบบอ้างอิงโอเอสไอได้ โดยเลเยอร์ 1 และ 2 เป็นเลเยอร์ที่ใช้ร่วมกันได้(compatible) เพราะโอเอสไอกำหนดระบบตัวกลางหลายระบบในเลเยอร์ดังกล่าว และทีซีพี/ไอพีก็ถูกออกแบบให้ใช้ตัวกลางใดก็ได้(medium independent) เมื่อพิจารณาเลเยอร์ 3 และ 4 คืออินเทอร์เน็ตและทรานสปอร์ตของทีซีพี/ไอพี และเน็ตเวิร์คและทรานสปอร์ตของโอเอสไอ จะเห็นได้ว่าโอเอสไอมีทางเลือกมากมายที่สามารถใช้ในเลเยอร์ทั้งสองนี้ได้ และบางตัวก็ทำหน้าที่ในลักษณะคล้ายคลึงกับในทีซีพี/ไอพี ข้อแตกต่างสำคัญระหว่างทีซีพี/ไอพีกับโอเอสไอคือเลเยอร์แอพพลิเคชั่นของทีซีพี/ไอพีซึ่งในโอเอสไอแล้วจะเท่ากับ 3 เลเยอร์บน ถึงแม้จะมีความแตกต่างในการแบ่งเป็นเลเยอร์อยู่บ้าง แต่ลักษณะในการส่งข้อมูลของทีซีพี/ไอพีจะเหมือนกับของโอเอสไอคือข้อมูลจะถูกส่งลงมาจากสแต็กหรือส่งขึ้นไปบนสแต็ก ขณะที่ข้อมูลถูกส่งลงมาแต่ละเลเยอร์ในสแต็กก็จะเพิ่มข้อมูลควบคุม(control information) เข้าไปเพื่อจะแน่ใจได้ว่าการส่งเกิดขึ้นอย่างเหมาะสม ข้อมูลควบคุมนี้เรียกว่าเฮดเดอร์เพราะมันถูกใส่ไว้หน้าข้อมูลที่ถูกส่ง แต่ละเลเยอร์ปฏิบัติกับข้อมูลทั้งหมดที่มันได้รับมาจากเลเยอร์ที่สูงกว่าเหมือนเป็นข้อมูลจริง ๆ และเพิ่มเฮดเดอร์ของมันเองไว้ข้างหน้าข้อมูลทั้งหมดนั้น การทำดังกล่าวนี้เรียกว่าเอ็นแคปซูเลชั่น(encapsulation) ดังรูป 2-3 เมื่อผู้รับได้รับข้อมูลก็จะทำตรงกันข้ามกับที่กล่าวมาคือแต่ละเลเยอร์จะนำเฮดเดอร์ออกมาก่อนที่จะส่งข้อมูลขึ้นไปยังเลเยอร์ที่สูงกว่า ข้อมูลก็จะไหลขึ้นไปบนสแต็ก ข้อมูลที่ได้รับก็จะถูกแปลความหมายทั้งเฮดเดอร์และข้อมูลเลเยอร์ต่าง ๆ และโพรโตคอลที่เกี่ยวข้องในโพรโตคอลชุดทีซีพี/ไอพีดังนี้



รูปที่ 3-3 การส่งผ่านข้อมูลระหว่างเลเยอร์

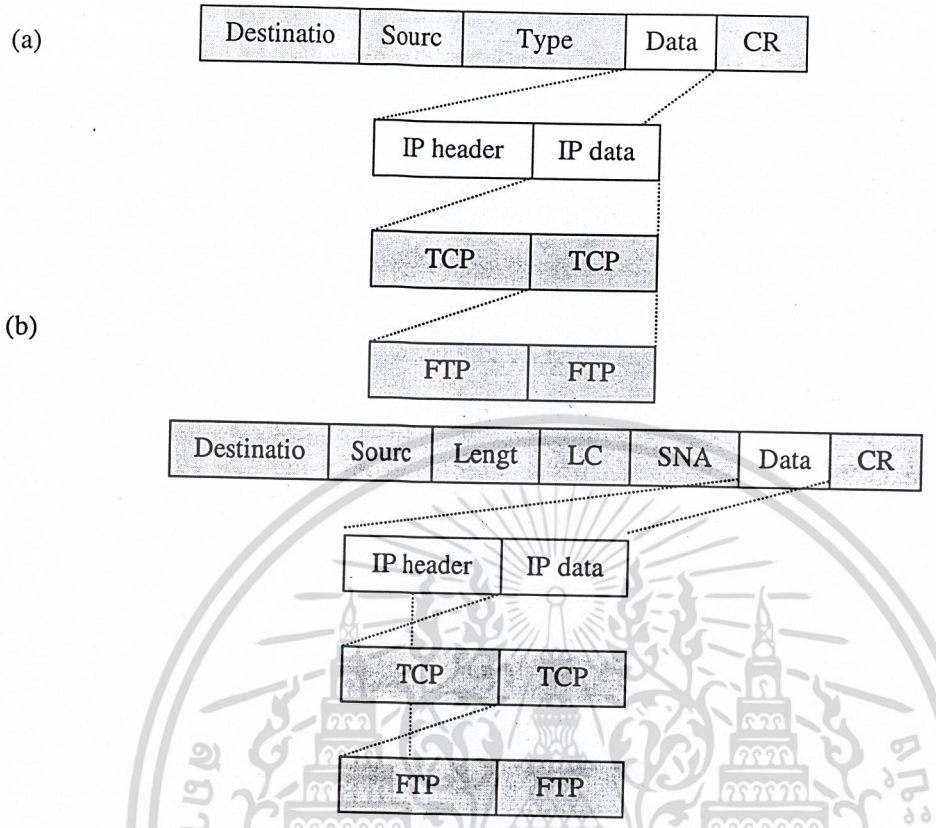
3.2.1 เลเยอร์เน็ตเวิร์คของทีซีพี/ไอพี

มีโพรโตคอลไอพี (Internet Protocol: IP) ทำหน้าที่พื้นฐานในการส่งโพรโตคอลชั้นสูงกว่าของทีซีพี/ไอพีไปบนเครือข่ายฟิสิคัลทั้งหมด ซึ่งทำให้โพรโตคอลในเลเยอร์ที่สูงกว่าไม่จำเป็นต้องรู้อะไรเกี่ยวกับความสามารถของตัวกลางเลย และยังทำให้ผู้พัฒนาแอพพลิเคชั่นที่เขียนโปรแกรมเหนือเลเยอร์ทรานสปอร์ตทำงานได้ง่ายขึ้นด้วยเหตุผลเดียวกัน นอกจากนี้โพรโตคอลไอพียังเกี่ยวข้องกับการส่งข้อมูลไปยังเครื่องและเครือข่ายที่ต้องการอย่างถูกต้องหรือการเร้าท์เส้นทางนั่นเอง ไอพีเป็นบริการแบบไม่ต้องการก่อตั้งการเชื่อมต่อก่อน(connectionless หรือ connectionless datagram service) เพราะไม่มีการเรียก(call) หรือก่อตั้งวงจรเสมือนก่อนที่จะเริ่มส่งข้อมูล เนื่องจากแต่ละดาต้าแกรมมีข้อมูลทั้งหมดที่จำเป็นต้องใช้ในการเร้าท์เส้นทางอยู่แล้ว และระหว่างโหนด 2 โหนดก็ไม่มีเส้นทางที่เฉพาะเจาะจงซึ่งทำให้ง่ายในการเร้าท์ใหม่แม้จะเสียเวลาในการสวิตชิ่งเล็กน้อย เมื่อเครือข่ายเกิดข้อผิดพลาด ส่วนแอดเดรสปลายทางที่ใช้มีทั้งคนเดียว(unique) เป็นกลุ่ม(multicast) หรือทุกคน(broadcast)



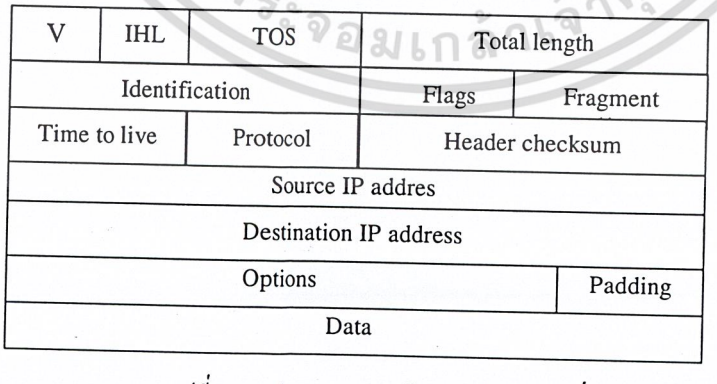
รูปที่ 2-4 สถาปัตยกรรมไอพี

นอกจากไอพีแล้วยังมีโพรโตคอลอื่นเลเยอร์นอีกคือ โพรโตคอลเออาร์พี(Address Resolution Protocol: ARP), โพรโตคอลอาร์เออาร์พี(Reverse Address Resolution Protocol: RARP), โพรโตคอลไอซีเอ็มพี(Internet Control Message Protocol: ICMP) ดังรูป 2-4 โพรโตคอลเออาร์พีและอาร์เออาร์พีแสดงที่ตำแหน่งล่างของชั้นไอพีเพราะโพรโตคอล 2 ตัวนี้ไม่ได้ใช้ไอพีและเป็นที่รู้จักโดยชั้นดาต้าลิงค์ที่สนับสนุนเหมือนเป็นโพรโตคอลที่แยกออกมาต่างหาก ไอซีเอ็มพีแสดงไว้ตำแหน่งบนของชั้นไอพีเพราะมันถูกส่งข้ามเครือข่ายโดยอยู่ในดาต้าแกรมไอพี ชั้นไอพีรู้ว่าเป็นดาต้าแกรมไอซีเอ็มพีโดยค่าโพรโตคอลที่เท่ากับ 1 ทั้งฟิลด์เฮดเดอร์และฟิลด์ข้อมูลของดาต้าแกรมไอพีจะกลายเป็นฟิลด์ข้อมูลของเฟรมในชั้นดาต้าลิงค์ ลักษณะเช่นนี้เรียกว่าการเอ็นแคปซูเลชั่นดังได้กล่าวแล้ว หรือบางครั้งก็เรียกการเอ็นเวลล็อปปิ้ง(enveloping) ซึ่งฟิลด์ข้อมูลของดาต้าแกรมไอพีเองก็บรรจุเฮดเดอร์ของโพรโตคอลในชั้นที่สูงกว่าเช่นเดียวกันดังรูป 2-5 ซึ่งแสดงการเอ็นแคปซูเลชั่นในเฟรมอีเธอร์เน็ตทูและเฟรม SNAP



รูปที่ 3-5 แสดงไอพีใน (a) เฟรมอีเธอร์เน็ตทู (b) เฟรม SNAP

ลักษณะดาต้าแกรมไอพีเวอร์ชั่น 4 เป็นดังรูป 2-6 ซึ่งแสดงในลักษณะกว้าง 32 บิต เมื่อดาต้าแกรมนี้ถูกส่งไปบนเครือข่ายลำดับการส่งจะเป็นจากซ้ายบนไปขวาล่างซึ่งเรียกว่าลำดับไบต์ของเครือข่าย (network byte order) และตัวเลขในทีซีพี/ไอพีจะถูกส่งโดยให้บิตที่สำคัญสูงสุด (most significant octet) ไปก่อน แต่ละฟิลด์ในรูป 2-6 สามารถอธิบายได้ดังต่อไปนี้



รูปที่ 3-6 รูปแบบของดาต้าแกรมไอพีเวอร์ชั่น

1. Version มีขนาด 4 บิต แสดงถึงเวอร์ชั่นของโพรโตคอลไอพี ขณะนี้คือเวอร์ชั่น 4

2. Internet Header Length มีขนาด 4 บิต แสดงถึงความยาวของเฮดเดอร์ หน่วยเป็น 32 บิตเวิร์ด ทำให้พบจุดเริ่มต้นของข้อมูลได้ง่ายถ้ามีการใช้ฟิลด์ Option แต่ปกติจะมีค่าเป็น 5 คือไม่มีการใช้ฟิลด์ Option

3. Type of Service มีขนาด 8 บิต ประกอบไปด้วยแฟล็กที่ใช้สำหรับทีโอเอส (TOS) และลำดับความสำคัญ โดย 3 บิตแรกใช้บ่งถึงลำดับความสำคัญ 8 ระดับซึ่งทำให้โหนดของไอพีรู้ว่าดาต้าแกรมใดมีความสำคัญมากกว่าดาต้าแกรมอื่น แต่เราท์เตอร์บางตัวก็ไม่สนใจแฟล็กนี้แฟล็กดี(D-flag)เป็นการร้องขอการเชื่อมต่อที่มีการเสียเวลา (delay) ต่ำแฟล็กที(T-flag)บอกถึงความต้องการทรูพุธ (throughput) สูง แฟล็กอาร์(R-flag) บอกถึงความต้องการความเชื่อถือ (reliability) สูง หมายถึงความน่าจะเป็นในการละทิ้ง (discard) ดาต้าแกรมมีต่ำกว่า แฟล็กซี(C-flag) บอกถึงความต้องการเสียค่าใช้จ่ายที่ต่ำกว่า บิตสุดท้ายไม่ใช้

4. Total Length มีขนาด 16 บิต เป็นการวัดทั้งเฮดเดอร์และข้อมูลในหน่วยออคเต็ด (octets) ซึ่งทำให้คำนวณขนาดข้อมูลโดยคิดจากฟิลด์ Total Length และฟิลด์ IHL ได้ จะเห็นได้ว่าฟิลด์นี้มีขนาด 16 บิตซึ่งหมายความว่าขนาดดาต้าแกรมที่ใหญ่ที่สุดมีขนาดเป็น 65,535 ออคเต็ดซึ่งใหญ่กว่าที่เครือข่ายฟิสิคัลสนับสนุนมาก ถ้าดาต้าแกรมถูกแบ่งย่อย (fragment) ค่าในฟิลด์นี้คือค่าใหม่ ไม่ใช่ค่าเก่าของขนาดดาต้าแกรม

5. Identification มีขนาด 16 บิต บ่งถึงการแบ่งย่อยทั้งหมดของดาต้าแกรม มีลักษณะเฉพาะของใครของมัน (unique) สำหรับแต่ละดาต้าแกรมใหม่ที่ถูกส่งโดยโฮส ฟิลด์นี้ไม่ใช่หมายถึงหมายเลขลำดับ (sequence number) เพราะไอพีเป็นบริการแบบไม่ต้องก่อตั้งการติดต่อก่อนส่งข้อมูล แต่เป็นเพราะไอพีสนับสนุนบริการการเชื่อมต่อของเลเยอร์ทรานสปอร์ตได้หลายแบบ

6. Flags มีขนาด 3 บิต ใช้ในการควบคุมการแบ่งย่อย ถ้าบิตลำดับต่ำมีค่าเป็น 0 หมายถึงเป็นส่วนสุดท้ายของดาต้าแกรมที่ถูกแบ่งย่อย บางครั้งจึงเรียกบิตนี้ว่า More flag หรือบิต MF บิตกลางใช้บ่งถึงว่าดาต้าแกรมนี้ห้ามแบ่งย่อยจึงเรียกว่า Do not fragment หรือบิต DF บิตลำดับสูงไม่ถูกใช้

7. Fragment offset มีขนาด 13 บิต ฟิลด์นี้ใช้ร่วมกับดาต้าแกรมที่ถูกแบ่งย่อยเพื่อบอกถึงตำแหน่งของข้อมูลในดาต้าแกรมเดิม วัดในหน่วย 8 ออคเต็ด ดังนั้นการแบ่งย่อยดาต้าแกรมจึงต้องทำให้หน่วยนี้

8. Time to live มีขนาด 8 บิต ฟิลด์นี้ถูกเซ็ตโดยผู้ส่งดาต้าแกรมและจะถูกลดค่าโดยเราท์เตอร์เมื่อดาต้าแกรมผ่านมัน ถ้าฟิลด์ TTL ถูกลดค่าจนเป็น 0 ดาต้าแกรมนั้นจะถูกละทิ้งเพื่อป้องกันไม่ให้ดาต้าแกรมถูกเร้าท์เป็นลูป (loop) ตลอดไป

9. Protocol มีขนาด 8 บิต บ่งถึงว่าดาต้าแกรมนั้นบรรจุโพรโตคอลใดของเลเยอร์ทรานสปอร์ต ค่าปกติคือ

17 ยูดีพี(UDP)

6 ทีซีพี(TCP)

1 ไอซีเอ็มพี(ICMP)

7 อีจีพี(EGP)

89 โอเอสพีเอฟ(OSPF)

10. Header checksum มีขนาด 16 บิต ป้องกันเฉพาะเฮดเดอร์ไม่รวมข้อมูล เพราะจะต้องคำนวณใหม่ทุกครั้งที่ผ่านเราท์เตอร์ เพราะค่าฟิลด์ TTL Flags และ Fragment offset เปลี่ยนไป ถ้าคำนวณข้อมูลด้วยจะทำให้เสียเวลามากขึ้น

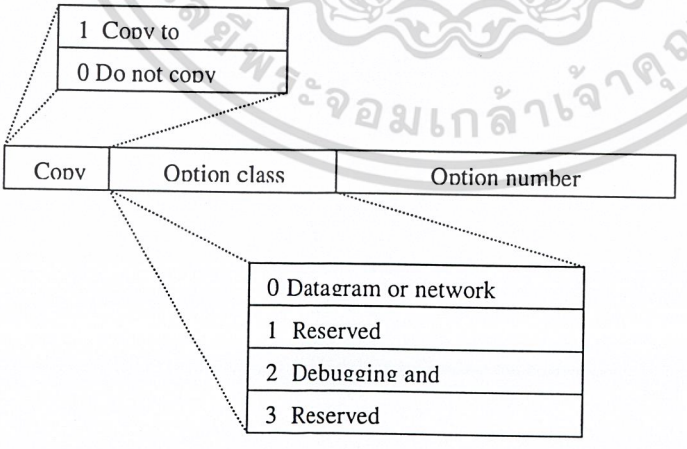
11. Source IP address มีขนาด 32 บิต

12. Destination IP address มีขนาด 32 บิต

13. Data มีขนาดไม่แน่นอน ซึ่งฟิลด์นี้จะรวมเฮดเดอร์ของโพรโตคอลในเลเยอร์ที่สูงกว่าไว้กับข้อมูลจริง ๆ ด้วย

14. Padding มีขนาดไม่แน่นอน ค่าของฟิลด์นี้จะแทนด้วย 0 ใช้เพื่อต่อเฮดเดอร์ให้ครบ 32 บิตเวิร์ด ซึ่งทำให้ IHL บอกถึงจุดเริ่มต้นของข้อมูลได้ถูกต้องเมื่อมีการใช้ฟิลด์ Options ซึ่งความยาวไม่คงที่

15. Options สนับสนุนการดีบัก (debugging)การวัด (measurement) และความปลอดภัย (security) ซึ่งสามารถมีหลาย ออปชั่นได้ในดาต้าแกรมเดียวดังรูป 2-7 ซึ่งฟิลด์นี้ประกอบด้วย



รูปที่ 3-7 ออปชั่นในการวัดและความปลอดภัย

15.1 Copy มีขนาด 1 บิต ใช้ตัดสินใจว่าออปชั่นจะอยู่ในทุกส่วนดาต้าแกรมที่ถูกแบ่งหรือไม่ ถ้ามีค่าเป็น 0 หมายถึงออปชั่นจะปรากฏในส่วนย่อยแรก (fragment) เท่านั้น ตัวอย่างของออปชั่นที่จำเป็นต้องคัดลอก (copy) ให้แก่ทุกส่วนย่อยคือออปชั่นเกี่ยวกับความปลอดภัย

15.2 Option class มีขนาด 2 บิต บอกถึงคลาสของออปชั่น ซึ่งออปชั่นไทม์แสตมป์ (Time stamp option) มีคลาสเป็น 2 นอกจากนั้นคลาสปกติจะเป็น 0

15.3 Option numbers มีขนาด 5 บิต

15.3.1 Security: คือ Option 2 มีการกำหนดระดับของดาต้าแกรมจากธรรมดาไปจนถึงดาต้าแกรมที่เป็นความลับมาก ซึ่งช่วยให้เราท์เตอร์รู้ว่าดาต้าแกรมใดบรรจุข้อมูลที่สำคัญและป้องกันข้อมูลเหล่านั้นไม่ให้ออกจากสิ่งแวดล้อมที่ปลอดภัย

15.3.2 Time stamp: คือ Option 4 ทำให้ดาต้าแกรมที่ถูกส่งไปในเครือข่ายสามารถรวบรวมไทม์แสตมป์จากแต่ละเราท์เตอร์ที่มันผ่านซึ่งเราสามารถนำสิ่งที่ได้นี้มาใช้ประเมินการเสียเวลาและความเปลี่ยนแปลงในเครือข่ายของเราท์เตอร์ได้

15.3.3 Loose source route: คือ Option 3 ในการกำหนดค่าของเราท์เตอร์เพื่อให้ดาต้าแกรมผ่านตามไอพีแอดเดรสของเราท์เตอร์ ออปชั่นนี้จะอนุญาตให้ใช้เราท์เตอร์อื่นได้ในระหว่างลิสต์ของเราท์เตอร์ที่ถูกกำหนด

15.3.4 Record route: คือ Option 7 ทำให้แต่ละเราท์เตอร์ใส่ไอพีแอดเดรสของมันในฟิลด์ Option ของดาต้าแกรมเมื่อดาต้าแกรมเดินทางผ่านเครือข่าย ซึ่งทำให้ค้นหาทางที่ดาต้าแกรมใช้ในการไปถึงโฮสหรือเราท์เตอร์ใด ๆ ได้

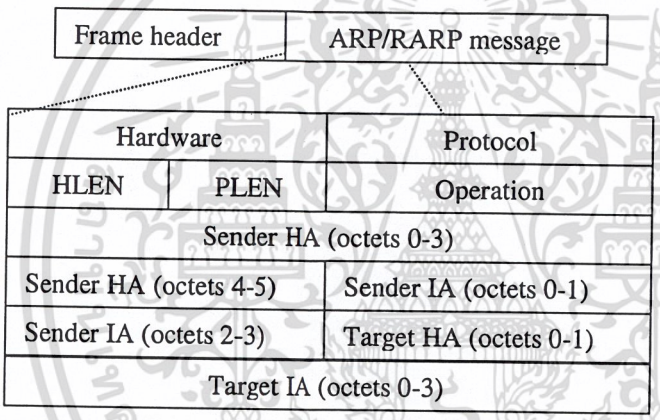
15.3.5 Strict source route: Option 9 คล้ายกับ loose source route ยกเว้นว่าเฉพาะเราท์เตอร์ที่กำหนดในลิสต์เท่านั้นที่สามารถใช้ได้

<u>โพรโตคอลเออาร์พี</u>(Address Resolution Protocol: ARP)

การ์ดแลนส่งและรับแฟรมโดยใช้แมคแอดเดรส(MAC address) แต่ทีซีพี/ไอพีใช้ไอพีแอดเดรสที่กำหนดโดยผู้ดูแลระบบเครือข่าย ณ เวลาติดตั้ง ซึ่งไม่มีความสัมพันธ์โดยตรงกับแมคแอดเดรส การสื่อสารแบบปลายถึงปลาย(end-to-end) ใช้ไอพีแอดเดรส แต่แบบฮ็อพถึงฮ็อพ(hop-to-hop) ใช้แมคแอดเดรส ดังนั้นเลเยอร์แมค(MAC) จึงต้องการแมคแอดเดรสของฮ็อพถัดไประหว่างไอพีแอดเดรสต้นทางและปลายทาง เราสามารถรู้แมคแอดเดรสของไอพีแอดเดรสที่กำหนดโดยใช้โพรโตคอลเออาร์พี แต่จะใช้ได้เฉพาะบนตัวกลางที่สนับสนุนการบรอดคาสท์เท่านั้น และแต่ละโหนดจะมีแคช(cache) ที่เรียกว่าแคชเออาร์พีซึ่งเก็บไอพีแอดเดรสและแมคแอดเดรสที่สัมพันธ์กัน เมื่อไอพีจะส่งดาต้าแกรมไปยังไอพีแอดเดรสอื่นมันจะหาแมคแอดเดรสของไอพีแอดเดรสที่เลเยอร์ดาต้าลิงค์จำเป็นต้องใช้ในการส่งจากแคชเออาร์พีก่อน ถ้าไม่พบมันจะพยายามหาแมคแอดเดรสจากไอพีแอดเดรสโดยใช้โพรโตคอลเออาร์พี ซึ่งการทำดังกล่าวนี้โพรโตคอลเออาร์พีจะส่งดาต้าแกรมร้องขอ(ARP request datagram) ไปยังทุกการ์ดแลนโดยใช้แมคแอดเดรสสำหรับการบรอดคาสท์(0xFFFF_FFFF_FFFF) พร้อมทั้งไอพีแอดเดรสของแมคแอดเดรสที่ต้องการ การ์ดในเครือข่ายจะอ่านคำขอนี้และทุกการ์ดที่รู้คำตอบจะตอบกลับ(ARP response) ซึ่ง

เมื่อได้รับคำตอบ คำตอบนี้ก็จะถูกเก็บไว้ในแคชเพื่อใช้ต่อไปในอนาคต แต่ถ้าไม่ได้รับคำตอบภายในเวลาไม่กี่วินาทีเออาร์พีรีเควสก็จะถูกส่งซ้ำ เพราะเออาร์พีอาจถูกละทิ้งได้เนื่องจากความผิดพลาดในการส่งหรือความคับคั่งของบริดจ์(bridge) เพื่อลดความจำเป็นในการบรอดคาสท์เออาร์พี โหนดที่ตอบกลับจะคัดลอกไอพีแอดเดรสและแมคแอดเดรสของผู้ร้องขอเก็บไว้ในแคชเออาร์พีของมันด้วย ดาต้าแกรมเออาร์พีนี้ไม่สามารถผ่านเราท์เตอร์ได้เพราะเราท์เตอร์ทำงานที่เลเยอร์ไอพี และไม่รีเลย์(relay) การบรอดคาสท์โดยใช้แมคแอดเดรสซึ่งทำให้ป้องกันการเกิดข้อมูลจำนวนมาก(flooding) วิ่งไปทั่วทั้งระบบได้

รูปแบบของดาต้าแกรมเออาร์พีแสดงดังรูป 2-8 สามารถใช้กับเครือข่ายแบบใดก็ได้ ไม่เฉพาะทีซีพี/ไอพีเท่านั้น แต่ต้องมีตัวกลางที่สามารถส่งเฟรมบรอดคาสท์ได้ เออาร์พีทำงานโดยตรงบนเลเยอร์ดาต้าลิงค์ ดังนั้นจึงถูกเอ็นแคปซูเลชั่นโดยแฟรมดาต้าลิงค์เท่านั้น ทำให้มันต้องการฟิลด์ Ethernet type ของมันเองคือ 0x0806



รูปที่ 3-8 รูปแบบดาต้าแกรมเออาร์พี

ฟิลด์ในดาต้าแกรมเออาร์พีประกอบด้วย

1. Hardware บอกถึงชนิดของฮาร์ดแวร์ที่ใช้ในเครือข่ายซึ่งสร้างดาต้าแกรมนี้ขึ้นมา ชนิดที่ใช้ได้คือ

Type	Description
1	Ethernet (10 Mbps)
2	Experimental Ethernet (3 Mbps)
3	Ameteur radio AX.25
4	Proteon ProNET Token Ring
5	Chaos
6	IEEE 802 networks
7	ARCNET
8	Hyperchannel
9	Lanstar
10	Autonet Short address
11	LocalTalk
12	LocalNet (IBM PCNet or Sytek Inc. LocalNet)

2. Protocol แสดงถึงโพรโตคอลที่ร้องขอ ค่าที่ใช้ในฟิลด์นี้จะเหมือนกับฟิลด์ Ethernet type ในเฟรมอีเธอร์เน็ต ซึ่งก็คือ 0x0800 สำหรับ IP

3. HLEN บอกถึงความยาวของฮาร์ดแวร์แอดเดรสในหน่วยออคเต็ด ปกติจะมีค่าเป็น 6 สำหรับแมคแอดเดรสของแลนไอทริปเปิ้ลอี

4. PLEN บอกถึงความยาวของแอดเดรสในเลเยอร์เน็ตเวิร์คในหน่วยออคเต็ด ปกติมีค่าเป็น 4 สำหรับ IP

5. Operation มีค่าเป็น 1 สำหรับเออาร์พีรีเควส และ 2 สำหรับเออาร์พีเรสปอนด์(ARP response) และยังใช้กับอาร์เออาร์พีด้วยโดยมีค่าเป็น 3 สำหรับอาร์เออาร์พีรีเควส และ 4 สำหรับอาร์เออาร์พีเรสปอนด์

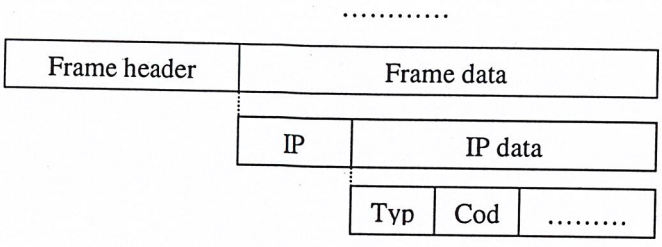
6. Addresses ประกอบด้วยฮาร์ดแวร์แอดเดรสของผู้ส่ง (แมคแอดเดรสต้นทาง), ไอพีแอดเดรสต้นทาง, ฮาร์ดแวร์แอดเดรสเป้าหมาย (แมคแอดเดรสปลายทาง), และไอพีแอดเดรสปลายทาง

Reverse Address Resolution Protocol(RARP)

ใช้สำหรับอุปกรณ์ที่ไม่สามารถเก็บไอพีแอดเดรสของตัวเองได้ เช่นเวิร์คสเตชั่นที่ไม่มีฮาร์ดดิสค์ อาร์เออาร์พีทำงานในลักษณะตรงกันข้ามกับเออาร์พีคือหาไอพีแอดเดรสจากแมคแอดเดรสที่กำหนด อาร์เออาร์พีทำงานโดยตรงกับเลเยอร์ดาต้าลิงค์โดยมีหมายเลขชนิดอีเธอร์เน็ตเท่ากับ 0x8035 โหนดที่ทำหน้าที่เป็นเซิร์ฟเวอร์อาร์เออาร์พี (RARP server) ที่พบแมคแอดเดรสที่กำหนดจะตอบกลับโดยอาร์เออาร์พีเรสปอนด์พร้อมทั้งไอพีแอดเดรสที่ต้องการ รูปแบบของดาต้าแกรมจะเหมือนเออาร์พีแต่ฟิลด์ Operation จะใช้ค่าเป็น 3 สำหรับรีเควส และ 4 สำหรับเรสปอนด์ ถึงแม้ว่าอาร์เออาร์พีจะทำงานได้ดีแต่ก็มีข้อจำกัดมากในทางปฏิบัติจึงถูกแทนที่โดยโพรโตคอลบูท (Boot Protocol: BOOTP) ซึ่งสามารถทำงานผ่านเราท์เตอร์และหาข้อมูลที่เป็นประโยชน์ได้มากกว่าอาร์เออาร์พีเมื่อเวิร์คสเตชั่นที่ไม่มีฮาร์ดดิสค์ทำการบูท

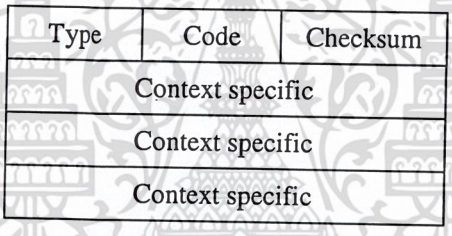
Internet Control Message Protocol(ICMP)

ถึงแม้ว่าไอพีจะไม่รับรองในการส่งข้อมูล แต่ไอซีเอ็มพีซึ่งใช้ได้ในไอพีสามารถสร้างแมสเสจเกี่ยวกับความผิดพลาดเพื่อช่วยเลเยอร์ไอพีในการให้บริการส่งข้อมูลให้ดีที่สุดและยังช่วยผู้ดูแลระบบในการวิเคราะห์หาสาเหตุเกี่ยวกับการทำงานของเครือข่าย ไอซีเอ็มพีใช้ดาต้าแกรมไอพีในการส่งแมสเสจระหว่างโหนด แมสเสจแสดงข้อผิดพลาดของไอซีเอ็มพีจะถูกสร้างโดยโหนดที่พบว่ามีปัญหาในการส่งเกิดขึ้นและจะส่งแมสเสจนี้กลับไปยังแอดเดรสที่เป็นต้นทางของดาต้าแกรมที่ทำให้เกิดปัญหา รูปที่ 2-9 แสดงถึงแมสเสจไอซีเอ็มพีที่ถูกเอ็นแคปซูเลทในดาต้าแกรมไอพีและแมสเสจแบบต่าง ๆ ที่เป็นไปได้ ไอซีเอ็มพีมีหมายเลขโพรโตคอล (protocol number) ของตัวเองซึ่งเท่ากับ 1 ทำให้ไอพีรู้ว่าได้รับไอซีเอ็มพี ถึงแม้ว่าไอซีเอ็มพีจะใช้เลเยอร์ไอพีแต่มันถูกมองว่าอยู่ภายในไอพีทั้งหมดเพราะไม่ได้ให้บริการแก่เลเยอร์ที่อยู่เหนือมัน



รูปที่ 3-9 แสดงไอซีเอ็มพีชนิดต่าง ๆ ถูกเอ็นแคปซูเลทใน

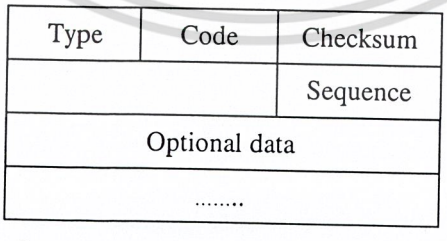
รูปแบบพื้นฐานของดาต้าแกรมไอซีเอ็มพีแสดงได้ดังรูป 2-10 แต่ฟิลด์ต่าง ๆ จะต่างกันซึ่งขึ้นอยู่กับชนิดที่ใช้อยู่ ฟิลด์ Type บ่งถึงชนิดของแมสเสจไอซีเอ็มพี ฟิลด์ Code ใช้แสดงถึงข่าวสารที่ละเอียดมากขึ้น ฟิลด์ Checksum ใช้เพราะไอพีไม่ได้ป้องกันข้อมูลของมันด้วยเช็คซัม (checksum) แต่เมื่อทำงานบนเครือข่ายฟิสิคัลซึ่งมีเฟรมเช็คซีเควันส์(Frame Check Sequence: FCS) เช็คซัมของไอซีเอ็มพีอาจเท่ากับ 0 หมายถึงไม่ถูกคำนวณ



รูปที่ 3-10 เฮดเดอร์พื้นฐานของโพรโตคอลไอซีเอ็มพี

1. ไอซีเอ็มพีชนิด 0 และ 8 – echo

ใช้เพื่อจุดประสงค์ในการหาสาเหตุ ถูกสร้างจากโปรแกรมอรรถประโยชน์(Utility program) ที่รู้จักกันดีคือ ping ซึ่งจะส่งไอซีเอ็มพีชนิด 8 ไปยังโหนดและคาดว่าจะได้รับไอซีเอ็มพีชนิด 0 ตอบกลับมา รูปแบบของไอซีเอ็มพีสองชนิดนี้เป็นดังรูป 2-11



รูปที่ 3-11 รูปแบบดาต้าแกรมไอซีเอ็มพีชนิด

ฟิลด์ Identifier และ Sequence number ใช้ในการทำให้ดาต้าแกรมนี้แตกต่างจากดาต้าแกรมอื่น ถ้ามีข้อมูลส่งในฟิลด์ Optional data มันจะต้องถูกส่งกลับในการตอบกลับ

2. ไอซีเอ็มพีชนิด 3 – destination unreachable

ถ้าเราท์เตอร์ไม่สามารถส่งดาต้าแกรมได้ มันจะส่งแมสเสจไอซีเอ็มพีชนิด Destination unreachable เพื่อบอกถึงสาเหตุ ฟิลด์ Code จะถูกใช้บอกถึงสาเหตุ ส่วนเฮดเดอร์อินเตอร์เน็ตรวมทั้งดาต้าแกรมพรีฟิกซ์ (datagram prefix) 64 บิตจะใช้ในการบอกถึงดาต้าแกรมที่เป็นสาเหตุของปัญหาดังรูป

Type	Code	Checksu
Unused (must be 0)		
Internet header+64		
.....		

Code value	Meaning
0	Network unreachable
1	Host unreachable
2	Protocol unreachable
3	Port unreachable
4	Fragmentation needed and the do not fragment bit set
5	Source route failed
7	Destination host unknown
11	Network unreachable for type of service
12	Host unreachable for type of service
13	Communication administratively prohibited e.g. firewall blocked
14	Host precedence violation
15	Precedence cut-off in effect

รูปที่ 3-12 ค่า destination unreachable ในฟิลด์ Code

2.1 Network unreachable: หมายถึงเครือข่ายที่ระบุในไอพีแอดเดรสไม่สามารถพบได้ ควรจะตรวจที่ไอพีแอดเดรสหรืออาจเกิดความผิดพลาดในตารางเราท์เส้นทาง (routing table) ของเราท์เตอร์ระหว่างทาง แมสเสจแสดงข้อผิดพลาดนี้ถูกสร้างโดยเราท์เตอร์เท่านั้น จุดที่เกิดข้อผิดพลาดจะทราบได้จากแอดเดรสต้นทางในเฮดเดอร์ของไอพีที่บรรจุแมสเสจของไอซีเอ็มพีซึ่งก็คือเราท์เตอร์ที่เจอข้อผิดพลาดนั่นเอง

2.2 Host unreachable: ดาต้าแกรมที่เป็นสาเหตุของความผิดพลาดได้ไปถึงเราท์เตอร์ที่ต่อตรงกับเครือข่ายปลายทางแล้ว แต่เมื่อเราท์เตอร์พยายามที่จะส่งดาต้าแกรมมันกลับไม่สามารถสื่อสารกับโฮสนั้นได้ ซึ่งอาจเกิดจากเออาร์พีล้มเหลวในดาต้าแกรมแรก โฮสดาวน์ หรือเหตุอื่นใดก็ตามซึ่งอาจเพราะไม่มีไอพีแอดเดรสนั้น เช่นเดียวกับ Network unreachable คือแมสเสจนี้จะถูกสร้างจากเราท์เตอร์เท่านั้น จุดที่เกิดข้อผิดพลาดก็คือจุดที่เป็นแอดเดรสต้นทางในเฮดเดอร์ไอพีที่บรรจุแมสเสจไอซีเอ็มพี ซึ่งก็คือเราท์เตอร์ที่พบข้อผิดพลาดนั่นเอง

2.3 Protocol unreachable: ในกรณีนี้ดาต้าแกรมได้ไปถึงโฮสปลายทางแล้วแต่ไม่สามารถใช้โพรโตคอลที่ถูกพามาในดาต้าแกรมไอพีได้ ซึ่งพบได้ไม่บ่อยนัก แต่ก็เป็นไปได้ถ้าเครื่องระยะไกล (remote machine) นั้นถูกกำหนดติดตั้งผิด

2.4 Port unreachable: ส่งจากโฮสเพื่อบอกว่าบริการในเลเยอร์แอพพลิเคชั่นที่โฮสระยะไกลที่การติดต่อกำลังถูกก่อตั้งอยู่นั้นไม่พร้อมที่จะใช้งานได้ (not available) ในแต่ละบริการของแอพพลิเคชั่น (application service) ที่โฮสนั้นจะถูกอีแนเบิ้ล (enable) และดิสแอเบิ้ล (disable) ขณะโฮสเริ่มทำงานโดยไฟล์คอนฟิคกูเรชั่น (configuration file) ดังนั้นเมื่อเกิดข้อผิดพลาดขึ้นก็ควรจะตรวจสอบที่ไฟล์นี้

2.5 Fragmentation needed and the do not fragment bit set: ปกติจะมาจากเราท์เตอร์ บ่งถึงความจำเป็นที่จะต้องแบ่งย่อยดาต้าแกรมแต่บิต Do not fragment หรือบิต DF ในฟิลด์ Flags ของเฮดเดอร์ไอพีไม่ยอมให้ทำ

2.6 Source route failed: ออปชั่นสำหรับการจัดการในไอพีอนุญาตให้ดาต้าแกรมไอพีไปตามเส้นทางที่กำหนดไว้ก่อนได้ แมสเสจนี้จะบ่งถึงข้อผิดพลาดที่ดาต้าแกรมไปตามเส้นทางที่กำหนดนี้ไม่สำเร็จ

2.7 Destination host unknown: ถูกสร้างขึ้นมาจากเราท์เตอร์เมื่อรู้จากซอฟต์แวร์สำหรับเชื่อมเลเยอร์ (link layer software) ว่าไม่มีโฮสปลายทาง

2.8 Network unreachable for type of service: สร้างจากเราท์เตอร์เมื่อพาธ (path) ที่จะไปถึงปลายทางไม่สอดคล้องกับที่ TOS ต้องการหรือไม่สอดคล้องกับ TOS ปกติ

2.9 Communication administratively prohibited: ถูกสร้างเมื่อเราท์เตอร์ไม่สามารถส่งแพ็คเกจต่อไปได้เนื่องมาจากการกรองแพ็คเกจ ตัวอย่างเช่น เหตุผลในความปลอดภัยหรือการคิดค่าบริการจากระบบ (local charging)

2.10 Host precedence violation: ถูกส่งจากเราท์เตอร์ที่เป็นฮ็อพแรกไปยังโฮสเพื่อบอกว่าไม่สามารถกำหนดลำดับความสำคัญดังที่ร้องขอได้สำหรับโฮสต้นทางหรือปลายทาง เครือข่ายต้นทางหรือปลายทาง โพรโตคอลในเลเยอร์ที่สูงกว่า และพอร์ตต้นทางหรือปลายทาง

2.11 Precedence cut-off in effect: บ่งถึงว่าผู้ดูแลระบบเครือข่ายได้กำหนดค่าระดับความสำคัญที่เส้นทางนี้ต้องการไว้ต่ำสุด คือดาต้าแกรมถูกส่งด้วยลำดับความสำคัญที่ต่ำกว่าที่ต้องการ

3. ไอซีเอ็มพีชนิด 4 และโค้ด 0 – source quence

รูปแบบของไอซีเอ็มพีชนิดนี้จะเหมือนกับไอซีเอ็มพีชนิด Destination unreachable แต่จะมี Type เป็น 4 และ Code เป็น 0 เท่านั้น ไอซีเอ็มพีแบบนี้ใช้ในการทำโฟลวคอนโทรล (flow control) เราท์เตอร์ที่พบว่าเครือข่ายหรือโปรเซสเซอร์ถูกใช้งานหนักเกินไปจะส่งแมสเสจไอซีเอ็มพีนี้ไปยังโฮสที่เป็นสาเหตุหลักของการใช้งานมาก ซึ่งโฮสดังกล่าวเมื่อได้รับแมสเสจก็จะลดอัตราการสร้างแพ็คเกจไปสู่ปลายทางที่ระบุมา

4. ไอซีเอ็มพีชนิด 5 – route change request

มีรูปแบบดังรูปที่ 2-13 ถูกใช้โดยเราท์เตอร์เท่านั้น สำหรับเราท์เตอร์ที่รู้ว่ามันไม่ใช่เราท์เตอร์ที่เหมาะสมที่สุดสำหรับการไปถึงปลายทางที่กำหนดก็จะใช้แมสเสจนี้แนะนำเราท์เตอร์ที่เหมาะสมกว่าแก่ผู้ส่ง คือไอพีแอดเดรสต้นทางของดาต้าแกรม เพื่อความต่อเนื่องในการส่งข้อมูลเราท์เตอร์จะส่งดาต้าแกรมที่เป็นสาเหตุให้เกิดแมสเสจนี้ไปยังเราท์เตอร์ที่เชื่อว่าเข้าถึงเส้นทางที่ดีกว่าด้วย

(a)

Type	Code	Checksum
Internet address of a more suitable		
Internet header+64 bits of		
.....		

(b)

Code value Meaning

0 Redirect datagrams to go to that network
1 Redirect datagrams to reach that host
2 Redirect datagrams for that network with that TOS
3 Redirect datagrams for that host with that TOS

รูปที่ 3-13 (a) รูปแบบของแมสเสจ route change request

(b) ค่าในฟิลด์ Code ที่แมสเสจประเภทนี้ใช้

5. ไอซีเอ็มพีชนิด 9 – router advertisement

ทำให้เราท์เตอร์ประกาศตัวกับโฮสบนระบบเครือข่ายได้ดังรูปที่ 2-14 ซึ่งจะส่งทุก ๆ 7-10 นาที หรือเพื่อตอบสนองโฮสจากแมสเสจไอซีเอ็มพีชนิด 10 แมสเสจนี้ไม่มีข้อมูลว่าจะติดต่อเราท์เตอร์นี้ผ่านเส้นทางไหน เพราะฉะนั้นถ้าโฮสเลือกเราท์เตอร์แรกไม่เหมาะสมก็จะได้รับแมสเสจไอซีเอ็มพีชนิด 5

Type	Code	Checks
Num	Addr	Life
Router address [1]		
Preference level [1]		
Router address [2]		
Preference level [2]		
.....		

รูปที่ 3-14 รูปแบบแมสเสจไอซีเอ็มพี router advertisement

6. ไอซีเอ็มพีชนิด 10 – router solicitation

สามารถถูกส่งได้โดยโฮสเวลาใดก็ได้ แต่มักจะเป็นตอนเปิดเครื่อง(Start-up) เพื่อหาเราท์เตอร์ที่สามารถใช้ได้ในเครือข่ายที่โฮสอยู่ เราท์เตอร์จะตอบสนองแมสเสจนี้ด้วยแมสเสจไอซีเอ็มพีชนิด 9(router advertisement response) หลังจากส่งแมสเสจนี้ไปแล้ว โฮสจะรอ unsolicited advertisements จากเราท์เตอร์ทุก ๆ 7-10 นาที รูปแบบของแมสเสจชนิดนี้เป็นดังรูป 2-15

Type	Code	Checksum
Reserved		

รูปที่ 3-15 รูปแบบแมสเสจไอซีเอ็มพี router solicitation

7. ไอซีเอ็มพีชนิด 11 – time exceeded for datagram

รูปแบบของไอซีเอ็มพีชนิดนี้จะเหมือนกับแมสเสจไอซีเอ็มพี Destination unreachable ซึ่งแมสเสจชนิดนี้จะถูกส่งในสถานการณ์ดังต่อไปนี้

7.1 จากเราท์เตอร์: ใช้บอกถึงว่าค่าฟิลด์ TTL ในเฮดเดอร์ไอพีถูกลดจนมีค่าเป็น 0 ในกรณีนี้ค่า Code จะเป็น 0 ทำให้ดาต้าแกรมถูกละทิ้งก่อนถึงปลายทาง ซึ่งส่วนใหญ่จะแสดงถึงว่าฟิลด์ TTL ที่ตั้งค่าเอาไว้ตอนเริ่มต้นไม่เหมาะสมหรือเกิดจากความเสียหายที่เกิดขึ้นกับเครือข่าย ซึ่งทำให้ความยาวของเส้นทางไม่ปกติ

7.2 จากโหนดปลาย: ค่าในฟิลด์ Code จะเป็น 1 บอกถึงการพยายามรวมส่วนย่อยเพื่อให้เป็นดาต้าแกรมเดิมไม่สำเร็จ การรวมดาต้าแกรมอีกครั้งโดยใช้เวลามากจนเกินไปถ้าเกิดขึ้นไม่บ่อยนักก็ไม่ถือว่าเป็นปัญหาร้ายแรงแต่อย่างใด

8. ไอซีเอ็มพีชนิด 12 – parameter problem

ใช้อาร์กิวเมนท์ผิดในฟิลด์ Option ของเฮดเดอร์ไอพี แต่ถ้าร้ายแรงกว่านั้นก็คือเกิดข้อผิดพลาดในการทำงานของไอพี มันแสดงถึงว่ามีค่าในเฮดเดอร์ที่ไม่สามารถเข้าใจได้ แมสเสจนี้จะไม่ถูกส่งถ้าดาต้าแกรมไม่ถูกละทิ้ง ฟิลด์ pointer บ่งถึงตำแหน่งของอ็อคเต็ดที่สงสัย เช่น ถ้ามีค่าเป็น 1 ก็หมายถึงฟิลด์ TOS ถ้ามีค่าเป็น 20 ก็จะหมายถึงอ็อคเต็ดแรกของฟิลด์ Option เป็นต้น รูปแบบของแมสเสจชนิดนี้เป็นดังรูปที่ 2-16

Type	Code	Checksum
Pointer	Unused (must be 0)	
Internet header+64 bits of		
.....		

รูปที่ 3-16 รูปแบบแมสเสจไอซีเอ็มพี parameter problems

9. ไอซีเอ็มพีชนิด 13 และ 14 – time stamp request and reply

ใช้เก็บเวลาจากนาฬิกา (Clock) ของเครื่องระยะไกล ผู้ร้องขอจะส่งแมสเสจไอซีเอ็มพีชนิด 13 และปลายทางจะตอบด้วยแมสเสจชนิด 14 ฟิลด์ Original time stamp จะถูกเติมก่อนดาต้าแกรมถูกส่ง ฟิลด์ Receive time stamp จะเติมทันทีที่ได้รับการร้องขอ และฟิลด์ Transmit time stamp จะถูกเติมทันทีก่อนที่จะตอบกลับไปยังต้นทาง รูปแบบของแมสเสจไอซีเอ็มพีชนิดนี้เป็นดังรูป 2-17 โดยแมสเสจชนิดนี้จะถูกใช้สำหรับเก็บสถิติเกี่ยวกับสมรรถนะของการเชื่อมต่อไปยังโฮสหรือเพื่อการเข้าจังหวะกันของนาฬิกาในโฮส

Type	Code	Checksum
Identifier		Sequence
Originate time stamp		
Receive time stamp		
Transmit time stamp		

รูปที่ 3-17 รูปแบบแมสเสจไอซีเอ็มพี time stamp request/reply

10. ไอซีเอ็มพีชนิด 15 และ 16 – information request

โฮสใช้เพื่อหาหมายเลขของเครือข่าย(Network number) ถ้าโฮสนั้นไม่รู้ แอดเดรสที่ใช้ในเฮดเดอร์ไอพีจะมีค่าเป็น 0 หมายถึงเครือข่ายนี้ ซึ่งจะถูกเติมอย่างถูกต้องโดยปลายทางและถูกส่งกลับมา รูปแบบของแมสเสจไอซีเอ็มพีชนิดนี้เป็นดังรูป 2-18 กลไกนี้ใช้กับระบบไดอัลอิน (dial-in) ที่ใช้สลิป (SLIP) เป็นวิธีในการกำหนดเน็ตเวิร์กแอดเดรสที่เหมาะสมให้กับแต่ละปลายของการเชื่อมต่อ

Type	Code	Checksum
Identifier		Sequence

รูปที่ 3-18 รูปแบบแมสเสจไอซีเอ็มพี information requests

11. ไอซีเอ็มพีชนิด 17 และ 18 - address mask request

ใช้ร่วมกับการกำหนดแอดเดรสเป็นสับเน็ต(Subnet addressing) ได้เพื่อให้โหนดรู้ถึงสับเน็ตมาสค์ (subnet mask) ของเครือข่ายที่มันต่ออยู่ด้วย โหนดสามารถส่งคำร้องขอไปยังแอดเดรสที่มันรู้จักซึ่งอาจเป็นเราท์เตอร์ หรือทำการบรอดคาสท์ไปยังเครือข่ายก็ได้ คำตอบกลับ(reply) จะส่งตรงถ้าโหนดรู้แอดเดรสของมันหรือทำการบรอดคาสท์ก็ได้ สับเน็ตมาสค์จะถูกใส่มาในฟิลด์ Address mask ของการตอบกลับดังรูป 2-19

Type	Code	Checksu
Identifier		Sequence
Address mask		

รูปที่ 3-19 รูปแบบแมสเสจไอซีเอ็มพี address mask request

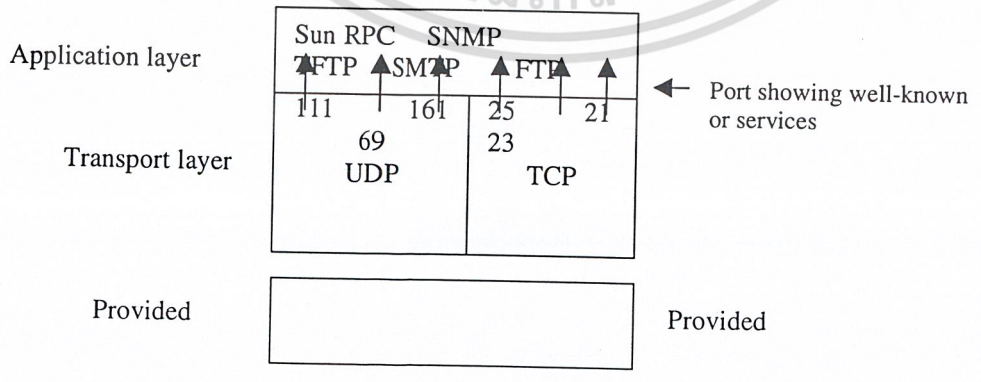
3.2.2 เลเยอร์ทรานสปอร์ตของทีซีพี/ไอพี

บริการที่ไอพีมีให้ยังต้องการให้คาต้าแกรมส่งไปยังบริการในเลเยอร์แอพพลิเคชั่นที่เหมาะสมโดยตรงและต้องการบริการที่เชื่อถือได้สำหรับแอพพลิเคชั่นที่จำเป็นต้องใช้มัน ซึ่งหน้าที่เหล่านี้เป็นของเลเยอร์นี้ทำโดยโพรโตคอลทรานสปอร์ต 2 ตัวคือโพรโตคอลยูดีพี (User Datagram Protocol: UDP)

และโพรโตคอลทีซีพี (Transmission Control Protocol: TCP) ซึ่งการเลือกใช้นั้นขึ้นอยู่กับประเภทของบริการที่แอพพลิเคชั่นของผู้ใช้ต้องการ

เมื่อข้อมูลถูกส่งไปยังเครื่องที่ต้องการโดยไอพีมันจะถูกส่งไปยังบริการแอพพลิเคชั่นที่เกี่ยวข้องบนเครื่องนั้น ๆ การมัลติเพล็กซ์และดีมัลติเพล็กซ์ข้อมูลไปยังหรือจากเลเยอร์ไอพีและส่งข้อมูลไปยังแอพพลิเคชั่นที่ถูกต้องเป็นหน้าที่อย่างหนึ่งของเลเยอร์ทรานสปอร์ตนี้ รวมทั้งการทำให้ไม่มีข้อผิดพลาด(error-free) และบริการส่งข้อมูลไปยังแอพพลิเคชั่นที่ถูกต้องแบบต้องก่อตั้งหรือไม่ก่อตั้งการเชื่อมต่อก่อนก็เป็นหน้าที่ของเลเยอร์นี้เช่นกัน โพรโตคอลยูดีพีจะให้บริการแบบไม่ต้องก่อตั้งการเชื่อมต่อก่อน(connectionless) ซึ่งเป็นบริการที่ไม่มีความน่าเชื่อถือ เพราะว่ามันจะอนุญาตให้ส่งข้อมูลไปยังเครื่องหรือกลุ่มของเครื่องโดยไม่จำเป็นต้องก่อตั้งการเชื่อมต่อก่อนดังนั้นดาต้าแกรมหนึ่งจะถูกส่งไปยังโหนดอื่นได้โดยไม่ต้องการการตอบสนองว่าดาต้าแกรมดังกล่าวได้ไปถึงแล้วหรือยัง ในสิ่งแวดล้อมบางอย่างบริการแบบนี้ก็เป็นวิธีที่มีประสิทธิภาพในการทำงานมาก บริการแอพพลิเคชั่นที่ใช้โพรโตคอลนี้ได้แก่ ทีเอฟทีพี(TFTP) เอ็นเอฟเอส(NFS) และการบรอดคาสท์ เป็นต้น โพรโตคอลทีซีพีจะให้บริการแบบจำเป็นต้องก่อตั้งการเชื่อมต่อก่อน(connection-oriented) การเชื่อมต่อมีลักษณะคล้ายท่อของข้อมูลที่อยู่ระหว่างจุด 2 จุด ไม่มีการทำบรอดคาสท์หรือมัลติคาสท์ในโพรโตคอลนี้ โพรโตคอลทีซีพีมีฟีเจอร์(feature) ในการให้บริการที่น่าเชื่อถือระหว่างคอมพิวเตอร์ 2 เครื่อง เพื่อให้มีความน่าเชื่อถือ โพรโตคอลทีซีพียังได้เพิ่มโอเวอร์เฮดจำนวนมากเพื่อใช้ในการทำแอคโนวเลดเมนท์(acknowledgement), โฟลวคอนโทรล(flow control), ไทม์เมอร์(timers) และความสะดวกสบายในการจัดการการเชื่อมต่อ(connection management facilities) ทีซีพีมีโอเวอร์เฮดมากกว่ายูดีพีในแง่ของการประมวลผลที่ต้องการและขนาดของเฮดเดอร์ที่ต้องใช้ ตัวอย่างของแอพพลิเคชั่นที่ต้องการบริการแบบนี้ได้แก่ เทลเน็ต(Telnet) และเอฟทีพี(FTP) เป็นต้น

ทั้งทีซีพีและยูดีพีต่างก็ใช้การอ้างแอดเดรสเป็นพอร์ต(port addressing) ในการส่งข้อมูลไปยังบริการในชั้นแอพพลิเคชั่นที่สัมพันธ์กัน ซึ่งพอร์ตก็คือแอดเดรสขนาด 16 บิตซึ่งหมายเลขของพอร์ตที่เป็นที่รู้จักกันดี(well-know port) ถูกกำหนดเป็น 0-255 ดังรูป 2-20 นอกจากนี้ยังมีการใช้ซ็อคเก็ต(socket)



รูปที่ 3-20 หมายเลขพอร์ตที่ใช้ในยูดีพีและทีซีพี

ในทีซีพี/ไอพีอีกด้วย ซึ่งซ็อคเก็ตก็คือการนำไอพีแอดเดรสมาต่อกับหมายเลขพอร์ต เมื่อไอพีแอดเดรสนั้นไม่มีใครซ้ำในแต่ละโหนดและหมายเลขพอร์ตก็ไม่ซ้ำบนโหนดนั้น ดังนั้นซ็อคเก็ตก็จะเป็นการระบุถึงบริการในชั้นแอพพลิเคชั่นที่ไม่มีการใช้ซ้ำกัน เพราะซ็อคเก็ตไม่ซ้ำดังนั้นทั้งโพรโตคอลทีซีพีและยูดีพีจึงได้รวมไอพีแอดเดรสกับหมายเลขพอร์ตเข้าไปคำนวณเช็คซัมด้วยเพื่อให้แน่ใจได้ว่าดาต้าแกรมที่ส่งไปถึงโฮสผิดจะไม่เป็นที่ยอมรับโดยเลเยอร์ทรานสปอร์ตของโฮสนั้นแม้หมายเลขพอร์ตนั้นจะเป็นที่รู้จักกันดีก็ตาม บริการในชั้นแอพพลิเคชั่นส่วนใหญ่จะอนุญาตให้มีหลายเซสชั่น(session) ได้ซึ่งทำให้จำเป็นต้องแยกเซสชั่นเหล่านี้ให้ได้เพื่อให้แน่ใจได้ว่าข้อมูลจะถูกส่งกลับคอมพิวเตอร์ที่เหมาะสม ตัวอย่างเช่น ผู้ใช้ทุกคนที่ใช้เทลเน็ตจะติดต่ออยู่กับโฮสเดียวกันด้วยหมายเลขพอร์ตเดียวกันคือ 23 ทางหนึ่งที่จะแยกแยะได้คือดูว่าดาต้าแกรมมาจากไหน แต่ก็เป็นไปได้ที่ผู้ใช้ 2 คนจะติดต่อมาจากโฮสเดียวกัน การแก้ปัญหานี้ทำได้โดยใช้พอร์ตที่รู้จักกันดีกับบริการชั้นแอพพลิเคชั่นของเซิร์ฟเวอร์เท่านั้น และให้โปรแกรมไคลเอ็นท์เลือกหมายเลขพอร์ตที่ยังไม่มีใครใช้ในเครื่องนั้นมาใช้ ด้วยวิธีนี้ถึงแม้ว่า 2 เซสชั่นจะใช้เซิร์ฟเวอร์เดียวกันและยังมาจากโฮสเดียวกันก็ง่ายในการแยกแยะ 2 เซสชั่นนี้

<u>โพรโตคอลยูดีพี</u>(User Datagram Protocol: UDP)

ยูดีพีเพิ่มความสามารถเข้าไปในการให้บริการของไอพีเพียงเล็กน้อย และใช้ฟิลด์ Source Port และ Destination Port ในเฮดเดอร์ของมันเพื่อส่งข้อมูลไปยังบริการในเลเยอร์แอพพลิเค ชั่น ฟิลด์ต่าง ๆ ในโพรโตคอลยูดีพีเป็นดังรูป 2-21 ประกอบด้วย

Source port	Destination
Length	UDP
Data	

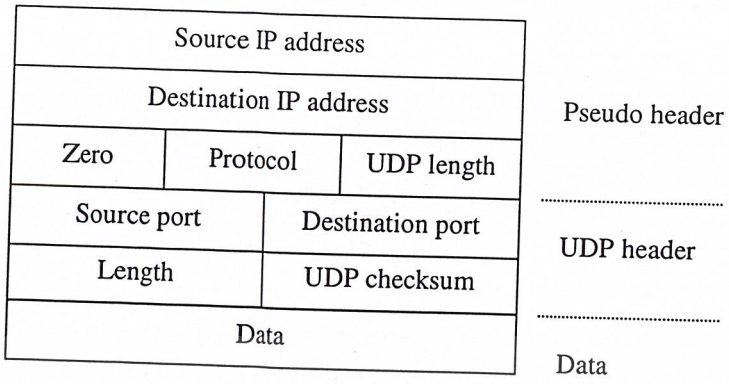
รูปที่ 3-21 ฟิลด์ในเฮดเดอร์ของยูดีพี

1. Source Port บอกถึงว่าดาต้าแกรมมาจากพอร์ตหมายเลขอะไร ซึ่งก็คือบริการใดในชั้นแอพพลิเคชั่น มีขนาด 16 บิต

2. Destination Port บอกถึงว่าดาต้าแกรมนั้นต้องการส่งไปยังพอร์ตหมายเลขอะไร บริการใดในชั้นแอพพลิเคชั่น มีขนาด 16 บิต

3. Length ความยาวของดาต้าแกรมยูดีพี มีขนาด 16 บิต

4. Checksum เป็นการป้องกันข้อมูลที่ยูดีพีบรรทุกมาซึ่งคิดทั้งฟิลด์ยูดีพีและซูโดเฮดเดอร์ (Pseudo header) ดังรูป 2-22 มีขนาด 16 บิต



รูปที่ 3-22 ค่าเช็คซัมประกอบด้วยเฮดเดอร์และซูโดเฮดเดอร์

<u>โพรโตคอลทีซีพี</u>(Transmission Control Protocol: TCP)

ช่วยเพิ่มความน่าเชื่อถือให้ไอพีและใช้พอร์ตในการกำหนดแอดเดรสของเลเยอร์แอพพลิเคชั่นเช่นเดียวกันกับยูดีพี โพรโตคอลทีซีพีเป็นโพรโตคอลที่ต้องการการเชื่อมต่อก่อน(connection-oriented) ดังกล่าวมาแล้ว คือจะต้องเปิดการติดต่อก่อนส่งและเมื่อส่งเสร็จก็จะต้องปิดการติดต่อนั้นด้วย ข้อมูลที่ทีซีพีส่งให้ไอพีนั้นจะประกอบด้วยเฮดเดอร์ของทีซีพีรวมกับข้อมูลจากเลเยอร์แอพพลิเคชั่น ซึ่งรวมกันแล้วเรียกว่าเซ็กเมนท์ (segment) เพื่อความน่าเชื่อถือที่มากขึ้นเราต้องการสิ่งดังต่อไปนี้

- การตรวจและแก้ไขข้อผิดพลาด: ซึ่งเกี่ยวข้องกับความเป็นไปได้ที่เซกเมนท์จะเสียหายจากสายสื่อสารหรือซอฟต์แวร์ในเลเยอร์ที่สูงกว่า
- โฟลวคอนโทรล: ใช้ในการป้องกันไม่ให้ผู้ส่งทำให้ผู้รับประสบปัญหาเนื่องมาจากข้อจำกัดในทรัพยากร
- การจัดลำดับการส่ง: จำเป็นต้องมีเพราะเลเยอร์ไอพีสามารถส่งดาต้าแกรมซึ่งมีเซกเมนท์ทีซีพีในลำดับใดก็ได้ ซึ่งเกิดขึ้นเมื่อดาต้าแกรมถูกส่งคนละเส้นทาง
- การกำจัดเซกเมนท์ที่ซ้ำ: เกิดเพราะกลไกกู้คืน (error-recovery) ที่ทีซีพีใช้ทีซีพีได้ทำการเพิ่มความสามารถที่กล่าวข้างต้นได้โดย
- ใช้หมายเลขแสดงลำดับในการแยกแยะข้อมูล
- ต้องได้รับเอคโนวเลดเมนท์ของการส่งข้อมูลในลำดับที่ถูกต้อง
- มีการส่งเซกเมนท์ซ้ำเมื่อไม่ได้รับการตอบกลับในเวลาที่กำหนด

Source port			Destination port
Sequence number			
Acknowledgement number			
Data	Reserv	Code	Window
Checksum			Urgent pointer
Option			Padding
Data			
.....			

รูปที่ 3-23 เฮดเดอร์ทีซีพี

และเพื่อให้เกิดหน้าที่ที่ได้กล่าวข้างต้น เฮดเดอร์ทีซีพีจึงซับซ้อนมีฟิลด์มากกว่าเฮดเดอร์ยูดีพี ดังรูป 2-23 ทีซีพีมีฟิลด์ Source port และ Destination port ด้วยเหตุผลเดียวกันกับยูดีพีคือใช้แยกแยะแอพพลิเคชั่น ฟิลด์ที่เหลือส่วนมากมีเพื่อความน่าเชื่อถือและเกี่ยวข้องกับการควบคุมการติดต่อ ดังนี้

1. Sequencing number มีขนาด 32 บิต ในทีซีพีจะนับออคเต็ดในการส่ง แต่โพรโตคอลอื่นที่ใช้เลขลำดับเพื่อควบคุมความผิดพลาดจะนับเซ็กเมนท์เลขลำดับในเฮดเดอร์นี้จะกำหนดตำแหน่งในข้อมูลทั้งหมด ของออคเต็ดแรกในเซ็กเมนท์ ซึ่งช่วยในทีซีพีใส่เซ็กเมนท์ในตำแหน่งที่ถูกต้องในข้อมูลได้ แม้ไอพีจะส่งข้อมูลไม่เป็นลำดับก็ตามการที่มันมี 32 บิตทำให้ไม่เกิดการซ้ำของค่าแม้เวลาในการส่งจะเร็วมากก็ตาม คือเซ็กเมนท์ที่ได้รับอาจมีเลขลำดับซ้ำกับเซ็กเมนท์ซึ่งแอคโนวเลดไปเรียบร้อยแล้วได้แต่ถ้าเกิดข้อผิดพลาดเนื่องจากการซ้ำก็ไม่เป็นปัญหา เพราะแค่ไม่สนใจโดยไม่ต้องทำอะไร

2. Acknowledgement number มีขนาด 32 บิต บอกให้รู้ว่าได้รับออคเต็ดทั้งหมดอย่างถูกต้องจนถึงเลขแอคโนวเลดลบด้วย 1 เมื่อผู้ส่งได้รับค่านี้ก็ไม่จำเป็นต้องเก็บข้อมูลไว้เพื่อส่งใหม่อีกต่อไป ซึ่งเลขแอคโนวเลดนี้จะใช้ได้เมื่อเซ็ตแฟลก ACK

3. Data Offset วัดออฟเซ็ตที่เป็นจุดเริ่มต้นของฟิลด์ข้อมูลในหน่วย 32 บิตเวิร์ดค่าปกติคือ 5 ซึ่งก็คือเฮดเดอร์ 20 ออคเต็ดเมื่อไม่ใช้ฟิลด์ออปชั่น ฟิลด์นี้มีขนาด 4 บิต

4. Flagsมีขนาด 6 บิต ต่อจากฟิลด์ Reserve ซึ่งมีขนาด 6 บิตเช่นกัน ใช้บอกว่าฟิลด์อื่นใช้ได้หรือไม่ และสำหรับการควบคุมการติดต่อ ประกอบด้วย 6 แฟลกดังนี้

- URG บอกว่าใช้ฟิลด์ urgent pointer ได้ซึ่งฟิลด์นี้ชี้ไปยังออคเต็ดในฟิลด์ข้อมูลซึ่งเป็นปลายของข้อมูล Urgent ซึ่งไม่ถูกมองเป็นข้อมูลปกติและควรจะถูกประมวลผลก่อนข้อมูลอื่น ๆ

- ACK บอกว่าฟิลด์ Acknowledge ใช้ได้ ซึ่งฟิลด์นี้จะใช้ไม่ได้เมื่อก่อตั้งการเชื่อมต่อคือก่อนที่แต่ละโหนดจะสามารถตัดสินใจได้ว่าจะใช้ค่า Sequence และ acknowledge ใด

- PSH คือแฟลก push ซึ่งทำให้เลเยอร์ทีซีพีที่ระยะไกลส่งเซ็กเมนท์นี้ไปให้เลเยอร์แอพพลิเคชั่นอย่างทันทีทันใด ปกติทีซีพีจะหันมาสนใจข้อมูลจากเซ็กเมนท์ที่เข้ามาและส่งข้อมูลนี้ไปยังเลเยอร์แอพพลิเคชั่นในบัฟเฟอร์ที่ใหญ่กว่าเพื่อลดโอเวอร์เฮดในการประมวลผล

- RST คือแฟลก reset ใช้เมื่อเกิดข้อผิดพลาดอื่น ๆ บอกถึงว่ามีข้อผิดพลาดเกิดขึ้นและการควรจะหยุดการติดต่อ

- SYN คือแฟลก synchronize ใช้ขณะเริ่มต้นการก่อตั้งการเชื่อมต่อระหว่าง 2 โหนดซึ่ง ณ เวลานั้นทั้ง 2 โหนดไม่รู้ว่าควรจะใช้เลขแอคโนวเลดใดการก่อตั้งการเชื่อมต่อจะประกอบด้วยการแลกเปลี่ยนเซ็กเมนท์แบบ 2 ทาง (2-way exchange of segment) พร้อมทั้งเซ็ตแฟลก SYN ซึ่งแต่ละอันจะถูกแอคโนวเลดในเซ็กเมนท์โดยเซ็ตแฟลก ACK

- FIN ใช้ในการเลิกการติดต่อเมื่อข้างใดข้างหนึ่งไม่มีข้อมูลที่จะส่งก็จะส่งเซ็กเมนท์มีการเซ็ตแฟลก FIN เมื่อทั้ง 2 ข้างส่งแฟลก FINการติดต่อก็จะปิดลง

5. Window มีขนาด 16 บิต บอกถึงขนาดเนื้อที่ในบัฟเฟอร์ของโหนดนี้ที่ใช้ได้ในการเชื่อมต่อนี้ โหนดอื่นจะต้องไม่ส่งข้อมูลที่ยังไม่แอคโนวเลดมา เกินเนื้อที่ใน บัฟเฟอร์ที่ระบุนี้

6. Checksum มีขนาด 16 บิต ใช้ตรวจสอบเฮดเดอร์และข้อมูล

7. Urgent pointer มีขนาด 16 บิต ค่าในฟิลด์นี้ชี้ไปยังปลายของข้อมูลในฟิลด์ข้อมูลที่เร่งด่วนและต้องการความสนใจทันที จะใช้ได้เมื่อมีการเซ็ตแฟลก URG

8. Optionsขนาดไม่คงที่มีออปชั่นเดียวที่ใช้เป็นปกติในทีซีพีคือขนาดเซ็กเมนท์มากที่สุด(Maximum Segment Size:MSS) เพื่อออกเลเยอร์ทีซีพีปลายทางถึงขนาดเซ็กเมนท์มากที่สุดที่ควรจะส่งซึ่งรวมเฮดเดอร์ทีซีพีแล้ว

9. Padding ถ้าใช้ฟิลด์ออปชั่นแพดดิ้ง จะทำให้มั่นใจได้ว่าข้อมูลเริ่มที่ขอบ 32 บิตอย่างที่ออปเซ็ตข้อมูลชี้อย่างถูกต้อง

3.2.3 เลเยอร์แอพพลิเคชั่นของทีซีพี /ไอพี

บริการของเลเยอร์แอพพลิเคชั่นจะรับผิดชอบในการเชื่อมต่อ(Interface) ระหว่างแอพพลิเคชั่นของผู้ใช้ และบริการในชั้นทรานสปอร์ต บริการของแอพพลิเคชั่นไม่ใช่แอพพลิเคชั่นของผู้ใช้แต่เป็นการเชื่อมต่อกับแอพพลิเคชั่นนั้นกับเครือข่ายสื่อสาร มีบริการของแอพพลิเคชั่นหลายตัวที่เหมาะสมกับแอพพลิเคชั่นหลายชนิด และยังมียูทิลิตี้ในการจัดการอีกจำนวนหนึ่ง ตัวอย่างของบริการในชั้นนี้คือ

1. FTP ย่อมาจาก File Transfer Protocol ใช้พอร์ตหมายเลข 20 เป็นโพรโตคอลมาตรฐานและเป็นวิธีที่ง่ายที่สุดในการแลกเปลี่ยนไฟล์กันในอินเทอร์เน็ต FTP เป็นโพรโตคอลแอพพลิเคชั่นที่ใช้โพรโตคอลชุดทีซีพี/ไอพี และมักจะถูกใช้เสมอในการส่งไฟล์เว็บเพจจากผู้สร้างเว็บเพจไปยังคอมพิวเตอร์ที่ทำตัวเป็นเซิร์ฟเวอร์เพื่อให้ใครก็ตามในอินเทอร์เน็ตสามารถใช้ได้ นอกจากนี้ FTP ยังใช้ในการดาวน์โหลดโปรแกรมและไฟล์อื่น ๆ จากเซิร์ฟเวอร์มายังคอมพิวเตอร์ของเราได้ด้วย ในฐานะผู้ใช้เราสามารถใช้ FTP ด้วยอินเทอร์เฟสแบบคอมมานไลน์ง่าย ๆ เช่นจากหน้าต่างดอสพรอมท์ หรือด้วยโปรแกรมที่มีขายซึ่งจะเสนออินเทอร์เฟสแบบกราฟฟิกให้ นอกจากนี้เว็บบราวเซอร์ของเราก็ยังสามารถใช้สำหรับดาวน์โหลดโปรแกรมที่เราเลือกจากเว็บเพจโดยส่ง FTP request ได้ด้วย ในการใช้ FTP เรายังสามารถอัพเดทหมายถึงการลบ การเปลี่ยนชื่อ การย้ายตำแหน่ง และการคัดลอกไฟล์ที่เซิร์ฟเวอร์ได้อีกด้วยแต่เราจำเป็น

ต้องล็อกออนเข้าไปในเซิร์ฟเวอร์นั้นก่อน อย่างไรก็ตามไฟล์ที่ให้ใครก็ได้ใช้สามารถเข้าถึงได้โดย anonymous FTP

2. Telnet ใช้พอร์ตหมายเลข 23 Telnet คือทางที่จะช่วยให้เราสามารถเข้าถึงคอมพิวเตอร์ของใครก็ตามถ้าเขาอนุญาตซึ่งมักจะเรียกคอมพิวเตอร์ลักษณะนี้ว่าโฮสคอมพิวเตอร์ หรือถ้าจะกล่าวให้ลึกกว่านั้นก็สามารถจะพูดได้ว่า Telnet คือคำสั่งของผู้ใช้บนโพรโตคอลทีซีพี/ไอพีสำหรับการเข้าถึงคอมพิวเตอร์ระยะไกล โพรโตคอลเว็บหรือ HTTP และ FTP นั้นอนุญาตให้เราร้องขอไฟล์ที่เจาะจงจากคอมพิวเตอร์ระยะไกลแต่ไม่ได้ให้เราล็อกออนจริง ๆ ในฐานะผู้ใช้ของคอมพิวเตอร์เครื่องนั้น แต่ด้วย Telnet เราสามารถล็อกออนเหมือนเป็นผู้ใช้ปกติด้วยสิทธิอะไรก็ตามที่เราได้รับอนุญาตให้ทำบนเครื่องคอมพิวเตอร์เครื่องนั้น

3. SMTP ย่อมาจาก Simple Mail Transfer Protocol ใช้พอร์ตหมายเลข 25 SMTP คือโพรโตคอล TCP/IP ที่ถูกใช้ในการส่งหรือรับอีเมล์(e-mail) อย่างไรก็ตามเพราะข้อจำกัดในความสามารถของมันในการจัดคิวข่าวสารที่ฝั่งผู้รับ เราจึงมักจะใช้โพรโตคอลอื่นแทนเช่น POP3 หรือ IMAP ซึ่งจะให้ผู้ใช้เก็บข่าวสารในกล่องจดหมาย(mail box) ของเซิร์ฟเวอร์และดาวน์โหลดข่าวสารเหล่านั้นจากเซิร์ฟเวอร์เป็นระยะ ๆ หรือกล่าวได้อีกอย่างว่าปกติแล้วผู้ใช้จะใช้โปรแกรมที่ใช้ SMTP สำหรับการส่งอีเมล์และใช้ POP3 หรือ IMAP สำหรับการรับอีเมล์ โปรแกรมเกี่ยวกับการเมล์ส่วนใหญ่ เช่น Eudora จะให้เราระบุทั้งเซิร์ฟเวอร์ SMTP และเซิร์ฟเวอร์ POP

4. Gopher ใช้พอร์ตหมายเลข 70 Gopher เป็นโพรโตคอลชั้นแอพพลิเคชั่นในเซิร์ฟเวอร์ซึ่งโครงสร้างไฟล์ถูกจัดการเรียงเป็นลำดับชั้น Gopher ได้จัดหาทางที่จะนำเท็กซ์ไฟล์จากทั่วโลกมายัง viewer บนคอมพิวเตอร์ของเรา โกเฟอร์ได้รับความนิยมเป็นเวลาหลายปีโดยเฉพาะอย่างยิ่งในมหาวิทยาลัย และยังเป็นก้าวหนึ่งที่นำไปสู่ HTTP แต่ด้วยไฮเปอร์เท็กซ์ลิงค์ ภาษา HTML และการปรากฏตัวของบราวเซอร์แบบกราฟฟิกทำให้โกเฟอร์เสื่อมความนิยมลงอย่างรวดเร็ว โครงสร้างไฟล์แบบดั้งเดิมจำนวนหนึ่งโดยเฉพาะในมหาวิทยาลัยยังคงใช้อยู่และสามารถเข้าถึงโดยเว็บบราวเซอร์ส่วนใหญ่เพราะมันยังคงสนับสนุนโพรโตคอลโกเฟอร์ Gopher ถูกพัฒนาที่มหาวิทยาลัยมินเนโซต้า(the University of Minnesota) ถึงแม้ว่าบราวเซอร์โกเฟอร์และไฟล์จะเป็นเท็กซ์แต่บราวเซอร์โกเฟอร์ก็ได้ถูกพัฒนาให้แสดงรูปกราฟฟิกได้คือไฟล์ GIF และ JPEG ซึ่งถูกรวมไว้ในไฟล์ไดเรคทอรี่โกเฟอร์

5. HTTP ย่อมาจาก Hypertext Transfer Protocol (HTTP) ใช้พอร์ตหมายเลข 80 HTTP เป็นชุดของกฎสำหรับการแลกเปลี่ยนไฟล์ ซึ่งมีทั้งเท็กซ์ กราฟฟิก ภาพ เสียง วีดีโอ และไฟล์มัลติมีเดียอื่น ๆ บนเวิลด์ไวด์เว็บ(World Wide Web) เมื่อเปรียบเทียบกับชุดโพรโตคอลทีซีพี/ไอพีซึ่งเป็นพื้นฐานสำหรับการแลกเปลี่ยนข้อมูลข่าวสารบนอินเทอร์เน็ต HTTP ก็คือแอพพลิเคชั่นโพรโตคอล คอนเซ็ปท์(concepts) สำคัญที่เป็นส่วนหนึ่งของ HTTP ประกอบไปด้วยความคิดที่ว่าไฟล์สามารถอ้างอิงไปยังไฟล์อื่นได้ โดยเว็บเซิร์ฟเวอร์ใด ๆ ก็ตามนอกจากจะเก็บไฟล์ HTML และไฟล์อื่น ๆ แล้วยังมี HTTP daemon ซึ่งเป็นโปรแกรมที่ถูกออกแบบมาให้รอ HTTP requests และจัดการเมื่อคำร้องขอมาถึง เว็บบราวเซอร์ของเราก็คือไคลเอ็นต์ HTTP ซึ่งจะส่งคำร้องขอไปยังเซิร์ฟเวอร์ เมื่อผู้ใช้บราวเซอร์ใส่การร้อง

ขอไฟล์โดยการเปิดเว็บไฟล์(โดยการพิมพ์ URL: Uniform Resource Locator) หรือคลิ๊ก(click) บน hypertext link บราวเซอร์ก็จะสร้าง HTTP request และส่งไปยังไอพีแอดเดรสที่ระบุโดย URL หลังจากนั้น HTTP daemon ที่เซิร์ฟเวอร์ปลายทางจะได้รับคำร้องขอและเมื่อทำการประมวลผลสิ่งที่จำเป็นแล้วไฟล์ที่ถูกร้องขอก็จะถูกส่งกลับ

6. POP3 ย่อมาจาก Post Office Protocol 3 ใช้พอร์ตหมายเลข 110 POP3 เป็นเวอร์ชั่นล่าสุดของโพรโตคอลมาตรฐานสำหรับการรับอีเมล์ POP3 เป็นโพรโตคอลแบบไคลเอ็นต์/เซิร์ฟเวอร์ ซึ่งจะรับอีเมล์ที่ถูกส่งมาและเก็บเอาไว้ให้ในเซิร์ฟเวอร์ของเรา เราสามารถดูเมล์ได้ที่เซิร์ฟเวอร์และดาวน์โหลดได้ นอกจาก POP3 แล้วยังมีโพรโตคอลที่ทำงานคล้ายคลึงกันคือโพรโตคอล IMAP (Interactive Mail Access Protocol) ด้วย IMAP เราสามารถดูอีเมล์ที่เซิร์ฟเวอร์เหมือนกับว่าอีเมล์เหล่านั้นอยู่บนเครื่องคอมพิวเตอร์ของเราเอง อีเมล์ที่ถูกลบที่เครื่องเราจะยังอยู่บนเซิร์ฟเวอร์เช่นเดิม นอกจากนี้อีเมล์ยังสามารถเก็บและค้นหาได้ที่เซิร์ฟเวอร์ เราสามารถคิดได้ว่า POP คือบริการแบบเก็บและส่งต่อไป(store-and-forward) ส่วน IMAP ก็คือไฟล์เซิร์ฟเวอร์ระยะไกล(remote file server) POP และ IMAP เกี่ยวข้องกับการรับอีเมล์และไม่ยุ่งเกี่ยวกับ SMTP ซึ่งเป็นโพรโตคอลสำหรับส่งอีเมล์ข้ามอินเทอร์เน็ต

7. NNTP ย่อมาจาก Network News Transfer Protocol ใช้พอร์ตหมายเลข 119 NNTP คือโพรโตคอลที่ใช้โดยคอมพิวเตอร์ทั้งเซิร์ฟเวอร์และไคลเอ็นต์สำหรับจัดการข้อความ (notes) ที่ทิ้งไว้บนกลุ่มข่าว Usenet(Usenet newsgroups) NNTP ได้มาแทนที่โพรโตคอล Usenet ดั้งเดิมคือ UUCP(UNIX-to-UNIX Copy Protocol) เซิร์ฟเวอร์ NNTP จะจัดการเครือข่ายของกลุ่มข่าว Usenet ที่ถูกรวบรวมและรวมเซิร์ฟเวอร์เข้าไว้ที่ผู้ให้บริการอินเทอร์เน็ตของเรา ไคลเอ็นต์ NNTP อาจจะถูกรวมเป็นส่วนหนึ่งของ Netscape, Internet Explorer, Opera หรือเว็บบราวเซอร์อื่น ๆ หรือเราอาจจะใช้โปรแกรมแยกต่างหากที่เรียกว่า newsreader ก็ได้

8.SNMPย่อมาจาก Simple Network Management Protocol ใช้พอร์ตหมายเลข 161 SNMP คือโพรโตคอลที่ใช้บริหารการจัดการเครือข่ายและการมอนิเตอร์อุปกรณ์ในเครือข่ายและฟังก์ชันของอุปกรณ์เหล่านั้น ซึ่งไม่ได้จำกัดอยู่เฉพาะเครือข่ายที่ใช้ทีซีพี/ไอพี

9. IRC ย่อมาจาก Internet Relay Chat (IRC) ใช้พอร์ตหมายเลข 194 IRC คือระบบสำหรับ chat ที่เกี่ยวข้องกับชุดของกฎ ข้อตกลงและซอฟต์แวร์ประเภทไคลเอ็นต์/เซิร์ฟเวอร์ ในเว็บมีไซต์เฉพาะเช่นเมืองแห่งการคุย(Talk City) หรือเครือข่าย IRC และช่วยให้เราดาวน์โหลดไคลเอ็นต์ IRC มายังเครื่องคอมพิวเตอร์ของเรา เราสามารถเริ่มการคุยในกลุ่ม(เรียกว่าแชนแนล) ใดก็ได้ที่มีอยู่ ซึ่งมีโพรโตคอลสำหรับค้นหากลุ่มการคุยที่มีอยู่และสมาชิกของกลุ่มนั้น ๆ ด้วย ผู้ที่เข้าไปร่วมคุยในกลุ่มการคุยใดก็ตามจะใช้ชื่อเล่นซึ่งใช้ได้เฉพาะครั้งนั้น ๆ (เราไม่สามารถเป็นเจ้าของชื่อเล่นนั้นได้คืออาจมีคนใช้ซ้ำกับเราได้)

บทที่ 4
การโจมตีเครือข่าย

การโจมตีเครือข่ายหรือการบุกรุกเข้ามาในเครือข่าย มีพื้นฐานและหลักการทำงานคล้ายๆ กัน นั่นคือ อาจเกิดจากความบกพร่องของระบบปฏิบัติการ, ระบบการทำงานของเครือข่าย เช่น โพรโตคอลที่ใช้ในการติดต่อสื่อสาร ตัวอย่างเช่น ทีซีพี, ยูดีพี หรืออาจเกิดจากส่วนต่างๆ ที่เกี่ยวข้องรวมถึงฮาร์ดแวร์และอื่นๆ ซึ่งถูกนำมาใช้เป็นการเครื่องมือในการโจมตีเครือข่ายได้ ดังได้ยกเป็นตัวอย่างและศึกษาดังนี้

4.1 การโจมตีเพื่อให้ปิดบริการ (Denial of Services: DoS)

หมายถึง การกระทำใดๆ ที่ทำให้ระบบ เป้าหมายไม่สามารถให้บริการบางอย่างได้ หรือไม่สามารถให้บริการต่อไปได้อีก โดยทั่วไปโจมตีที่พอร์ตของทีซีพี/ไอพี ซึ่งเชื่อมต่อกับบริการ (Services) ที่รองรับพอร์ตนั้นๆ ดังนั้นการโจมตีพอร์ตจึงเท่ากับการโจมตีบริการของระบบนั่นเอง ซึ่งการโจมตีแบบนี้ถือเป็นความเสียหายที่รุนแรงมากในระบบที่ต้องให้บริการ ข้อมูลที่รวดเร็ว

4.2 รายละเอียดการโจมตีแบบต่างๆ

การโจมตีแบบต่างๆ แบ่งประเภทตามชั้นของการสื่อสารดังนี้

4.2.1 ประเภทอยู่ในชั้นเน็ตเวิร์ก

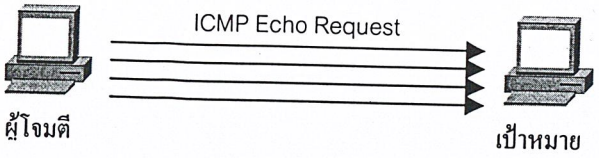
4.2.1.1 การส่งแพ็กเก็ตจำนวนมาก (Amount of Packets Sending)

การโจมตีแบบนี้เป็นการส่งแพ็กเก็ตปริมาณมากเข้าไปยังระบบเป้าหมาย อาจทำให้ระบบ เป้าหมายไม่สามารถให้บริการบางอย่าง หรือไม่สามารถทำงานต่อไปได้ ซึ่งแพ็กเก็ตที่ส่งออกไปนี้สามารถแบ่งออกได้เป็น

(1) แพ็กเก็ตข้อมูล (Data Packets)

การโจมตีวิธีนี้ทำได้โดยการส่งแพ็กเก็ตข้อมูลปริมาณมาก เมื่อข้อมูลเข้ามาสู่เครื่องเป้าหมายก็เก็บไว้ในบัฟเฟอร์ก่อนนำมาประมวลผลอีกครั้ง ดังนั้นหากส่งแพ็กเก็ตเข้ามาเป็นปริมาณมาก อาจทำให้บัฟเฟอร์ของเครื่องเป้าหมายไม่เพียงพอที่จะรองรับแพ็กเก็ตเหล่านั้นได้ทั้งหมด อาจทำให้เครื่องเป้าหมายให้บริการได้ช้าลง หรือต้องหยุดการให้บริการไปเลย ตัวอย่างการโจมตีประเภทนี้เช่น Ping Flood Attack เป็นต้น

Ping Flood เป็นการโจมตีในยุคแรกๆ ของ DoS หลักการคือส่ง ICMP Echo Request (รูปแบบเดียวกับคำสั่ง Ping) ไปยังเป้าหมายมากๆ ในระยะเวลาติดต่อกัน ทำให้เป้าหมายต้องคอยตอบ ICMP Echo Reply ตลอดเวลาจนไม่สามารถให้บริการอย่างอื่นได้ ความรุนแรงของการโจมตีขึ้นอยู่กับปริมาณแพ็กเก็ตที่โจมตีไปยังเครื่องเป้าหมาย หากเครื่องที่ทำการโจมตีมีประสิทธิภาพสูงและเครือข่ายมีแบนด์วิดธ์มาก อาจส่งผลทำให้เครื่องเป้าหมายหยุดการทำงานลงได้



รูปที่ 4-1 แสดงการโจมตีด้วย Ping Flood Attack

ข้อสังเกตสำหรับการโจมตีประเภทนี้คือ ปรากฏแพ็กเก็ต ICMP Echo Request และ ICMP Echo Reply ปริมาณมหาศาลซึ่งมีการรับส่งกันระหว่างเครื่องเป้าหมายที่ถูกโจมตีกับเครื่องอื่นๆ ซึ่งอาจมีหรือไม่มีในอินเทอร์เน็ตก็ได้ เนื่องจากกระบวนการสำคัญอย่างหนึ่งของการโจมตีลักษณะนี้คือ ผู้โจมตีต้องปลอมหมายเลขไอพี (IP Spoofing) เสมอ เพื่อป้องกันไม่ให้แพ็กเก็ต ICMP Echo Reply ถูกส่งกลับมายังเครื่องตัวเอง อันจะทำให้ผู้โจมตีได้รับผลจากการโจมตีด้วย และการปลอมไอพียังเป็นหลักประกันได้ว่าไม่สามารถติดตามได้ว่าผู้ใดเป็นผู้โจมตี รูปแบบแพ็กเก็ตที่เกิดขึ้นจากการโจมตีมีลักษณะดังนี้

```
14:49:43.217137 62.51.12.23          > 10.1.1.10      : icmp: echo request
14:49:43.217175 10.1.1.10          > 62.51.12.23      : icmp: echo reply
14:49:43.217195 62.51.12.23          > 10.1.1.10      : icmp: echo request
14:49:43.217219 10.1.1.10          > 62.51.12.23      : icmp: echo reply
14:49:43.217245 96.141.106.124     > 10.1.1.10      : icmp: echo request
14:49:43.217279 10.1.1.10        > 96.141.10.124      : icmp: echo reply
14:49:43.219017172.19.251.18 > 10.1.1.10             :icmp:net 162.75.127.79
unreachable
14:49:43.237136 75.126.62.65         > 10.1.1.10      : icmp: echo request
14:49:43.237169 10.1.1.10      > 75.126.62.65         : icmp: echo reply
14:49:43.237193 75.126.62.65         > 10.1.1.10      : icmp: echo request
14:49:43.237216 10.1.1.10      > 75.126.62.65         : icmp: echo reply
14:49:43.237240 218.155.179.58       > 10.1.1.10      : icmp: echo request
14:49:43.237272 10.1.1.10      > 218.155.17.58        : icmp: echo reply
```

รูปที่4-2รูปแบบแพ็กเก็ตที่เกิดขึ้นจากการโจมตีจาก Ping Flood Attack

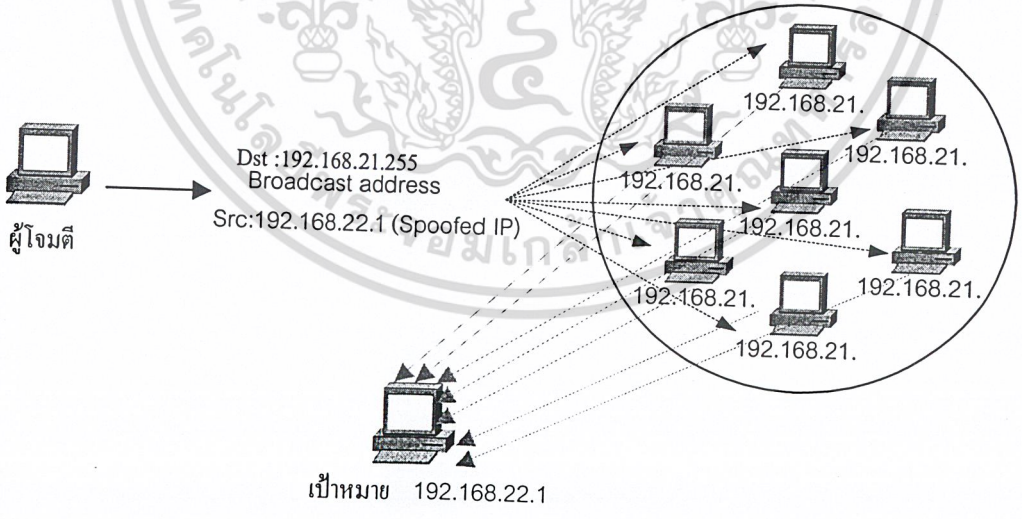
นอกจาก Ping Flood Attack จะสร้างความเสียหายแก่เครื่องเป้าหมายแล้วยังสร้างความเสียหายแก่ระบบเครือข่ายของเครื่องเป้าหมายด้วย เพราะการโจมตีวิธีนี้จะสร้างแพ็กเก็ตเป็นจำนวนมากขึ้นในเครือข่ายที่เครื่องเป้าหมายตั้งอยู่ ทำให้ระบบเครือข่ายเกิดความคับคั่งของข้อมูล (Congestion) อาจส่งผลให้เครือข่ายเป็นอัมพาตได้

การป้องกันการการโจมตีลักษณะนี้ทำได้โดยการกำหนดที่อุปกรณ์เราเตอร์หรือไฟร์วอลล์ โดยกำหนดไม่ให้แพ็กเก็ต ICMP Echo เข้ามายังเซิร์ฟเวอร์ แต่อย่างไรก็ตามแพ็กเก็ต ICMP Echo ได้ถูกใช้ในโปรแกรม Ping หากมีการปิดกั้นแพ็กเก็ต ICMP Echo จะทำให้ไม่สามารถตรวจสอบสถานะของเซิร์ฟเวอร์ได้ แนวทางที่ควรปฏิบัติคือกำหนดแบนด์วิดธ์ของแพ็กเก็ต ICMP Echo ให้เหมาะสมในอุปกรณ์เราเตอร์ โดยให้เพียงพอต่อการใช้งานสำหรับการตรวจสอบสถานะของระบบ แต่ไม่มากจนทำให้เกิดการโจมตีได้

อีกตัวอย่างหนึ่งของการโจมตีลักษณะนี้ ซึ่งมีลักษณะพิเศษขึ้นอีกเพื่อเพิ่มความสามารถในการโจมตี คือ สเมิร์ฟ (smurf)

Smurf เป็นการโจมตีที่มีรูปแบบที่ให้ผลกระทบที่ขยายตัวออกไปในวงกว้าง (Amplification effect) เป็นการปรับปรุงเทคนิคการส่งแพ็กเก็ตจำนวนมากให้ฉลาดกว่าเดิม โดยการขยายการโจมตีทำให้แฮกเกอร์มีเครื่องทุ่นแรง และเปิดโอกาสให้ผู้ที่มีแบนด์วิดธ์ต่ำสามารถทำการ flood เป้าหมายได้รุนแรง จากคุณสมบัติของการบรอดคาสต์ การส่งแพ็กเก็ตใดไปยังแอดเดรสบรอดคาสต์จะทำให้ทุกโฮสต์ในเน็ตเวิร์กได้รับแพ็กเก็ตนั้นอย่างทั่วถึง ดังนั้นหากมีโฮสต์ใดที่ส่ง ICMP Echo Request มายัง บรอดคาสต์ ก็จะทำให้ทุกๆ โฮสต์ทั้งหมดที่อยู่ในเน็ตเวิร์กนั้นได้รับ ICMP Echo พร้อมกัน และด้วยข้อกำหนดของ ICMP เมื่อโฮสต์ได้รับ ICMP Echo Request จะต้องตอบกลับด้วย ICMP Echo Reply กลับไปยังผู้ส่งเสมอ ซึ่งเป็นไปได้ว่าหากมีการส่ง ICMP Echo ไปยังบรอดคาสต์เพียงครั้งเดียวก็จะได้รับ ICMP Echo Reply ตอบกลับเท่ากับจำนวนเครื่องที่อยู่ในเน็ตเวิร์กนั้นเลย ปรากฏการณ์นี้เรียกว่าการขยายสัญญาณ (Amplification) โดยการบรอดคาสต์ อัตราการขยายก็จะขึ้นอยู่กับปริมาณโฮสต์ที่อยู่ในเน็ตเวิร์กขณะนั้น วิธีการโจมตีนี้จะทำได้ต้องอาศัยเทคนิคการปลอม IP Address ด้วย โดยให้ IP Address ต้นทางของ ICMP Echo Request ที่ส่งไปนั้นเป็น IP Address ของเป้าหมาย

การโจมตีชนิดนี้ เป้าหมายอาจไม่จำเป็นต้องเป็นโฮสต์ใดโฮสต์หนึ่ง โดยการดัดแปลงการโจมตีให้เกิดผลเสียหายต่อเน็ตเวิร์กได้โดยการทำให้เน็ตเวิร์กท่วมไปด้วยแพ็กเก็ต เพียงแฮกเกอร์ส่งแพ็กเก็ตที่ใช้การโจมตีอย่างต่อเนื่อง และอัตราสูงก็จะทำให้เน็ตเวิร์กนั้นเต็มไปด้วยแพ็กเก็ตของ ICMP cho Reply ซึ่งทำให้แพ็กเก็ตอื่น สำหรับใช้งานตามปกติไม่สามารถออกไปได้

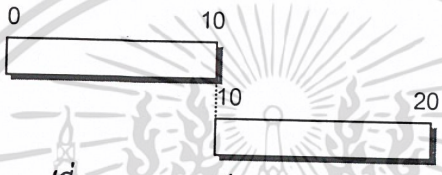


รูปที่ 4-3 แสดง การโจมตีด้วย Smurf

(2) แพ็กเก็ตสำหรับการควบคุม (Control Packets) ซึ่งจะเป็นแพ็กเก็ตที่กำหนดค่าแฟล็กควบคุมต่างๆ เช่น Syn Fin Rst เป็นต้น โดยการกำหนดค่าจะไม่เป็นตามมาตรฐานของ ทีซีพี/ไอพี โดยค่าเหล่านี้จะถูกกำหนดที่แพ็กเก็ตของ ทีซีพี ซึ่งจะอธิบายถึงรายละเอียดในหัวข้อ การโจมตีในชั้นทรานสปอร์ต

4.2.1.2 ความผิดปกติของแฟร็กเมนต์ (Abnormal Fragmentation)

การโจมตีวิธีนี้อาศัยหลักการแฟร็กเมนเตชันและรีแอสเซมเบิ้ลที่กล่าวไว้ข้างต้น(ในบทที่2) โดยทำให้แพ็กเก็ตนั้นต้องมีการรีแอสเซมเบิ้ล (กำหนดค่า MF flag = 0) ซึ่งปกติการรีแอสเซมเบิ้ลแพ็กเก็ตทั้งหมดต้องสามารถเชื่อมต่อกันได้สนิท ดังรูปที่ 3-4 แต่แพ็กเก็ตที่ผู้บุกรุกส่งไปมีการแก้ไขข้อมูลในบางฟิลด์ ทำให้เกิดความผิดปกติในกระบวนการรีแอสเซมเบิ้ล ซึ่งการโจมตีในลักษณะนี้ แบ่งได้ดังต่อไปนี้



รูปที่ 4-4 แสดงการรีแอสเซมบลีแบบปกติ

(1) การส่งแพ็กเก็ตที่มีลำดับผิดปกติ (Abnormal Sequences of Packets Sending)

ปกติการส่งแพ็กเก็ตมักเรียงตามลำดับกันไป หากไม่เรียงลำดับก็ต้องรอจนกว่าแพ็กเก็ตก่อนหน้านี้มาถึง เพื่อเรียงลำดับแพ็กเก็ตที่เครื่องรับ แต่การโจมตีแบบนี้กลับส่งเฉพาะแพ็กเก็ตสุดท้าย เพื่อให้ระบบเป้าหมายรอแพ็กเก็ตก่อนหน้า และส่งไปเป็นปริมาณมากๆ เพื่อให้ระบบเป้าหมายไม่สามารถให้บริการอย่างอื่นได้

รูปที่ 4-5 แสดงแพ็กเก็ตสุดท้ายที่ต้องรอแพ็กเก็ตก่อนหน้า

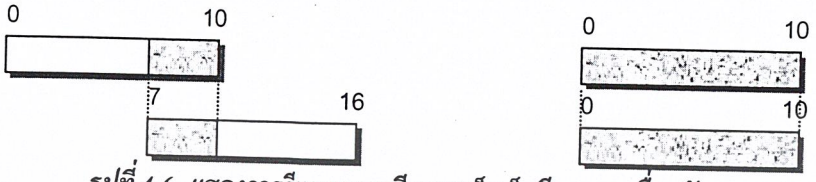
โดยปกติแล้วการโจมตีในรูปแบบนี้ผู้โจมตีจะแก้ไขข้อมูลในฟิลด์แสดงลำดับของแพ็กเก็ต (Fragment Offset) ของแพ็กเก็ตไอพี ซึ่งเป็นส่วนที่แสดงลำดับของข้อมูลหลังจากกระบวนการแฟร็กเมนต์เตชันโดยแก้ให้ส่งแพ็กเก็ตสุดท้ายหรือแพ็กเก็ตหลังๆ เพียงแพ็กเก็ตเดียวเลย ทำให้ระบบเป้าหมายต้องรอแพ็กเก็ตก่อนหน้านี้

(2) การส่งแพ็กเก็ตที่มีขนาดเหลื่อมกัน (Overlapped Packets' Size Sending)

ปกติแพ็กเก็ตที่ส่งมาต้องนำมาต่อกันที่ระบบเป้าหมายได้พอดี แต่การโจมตีแบบนี้เป็นการส่งแพ็กเก็ตที่มีขนาดเหลื่อมกัน หรือซ้อนทับกัน ทำให้ข้อมูลเมื่อมาต่อกันแล้วเกิดความผิดพลาด หรือไม่สามารถเชื่อมต่อกันได้โดยปกติแล้วการโจมตีแบบนี้ ผู้บุกรุกสามารถแก้ไขข้อมูลได้ 2 แห่งใหญ่ๆ ได้แก่

- การแก้ไขข้อมูลที่ฟิลด์แสดงลำดับของแพ็กเก็ต (Fragment Offset) ของแพ็กเก็ตไอพี หลังจากกระบวนการรีแอสเซมเบิ้ล ซึ่งทำให้ลำดับในการส่งมีความผิดพลาด และอาจเกิดการเหลื่อมล้ำของแพ็กเก็ต กระบวนการรีแอสเซมเบิ้ลอาจเกิดปัญหาได้

- การแก้ไขฟิลด์แสดงความยาวของ (Total Length) ของแพ็กเก็ตไอพี หลังจากกระบวนการรีแอสเซมเบิ้ล ขนาดของแพ็กเก็ตที่มาต่อไม่พอดีกัน ทำให้ไม่สามารถรวมแพ็กเก็ตได้ หรือหากรวมได้ ข้อมูลที่ได้ก็ไม่ถูกต้อง



รูปที่ 4-6 แสดงการรีแอสเซมบลีแบบแพ็กเก็ตมีขนาดเหลื่อมกัน

(3) การส่งแพ็กเก็ตแบบวนลูป (Looping)

คือ การส่งโดยกำหนดค่าแอดเดรสต้นทาง (Source Address) และแอดเดรสปลายทาง (Destination Address) ให้เหมือนกันทำให้เกิดการรับส่งวนไปวนมาอยู่ที่เครื่องเป้าหมายเอง

ตัวอย่างของการโจมตีแบบนี้ได้แก่ LAND ซึ่งเป็นโปรแกรมโจมตีที่มีลักษณะดังนี้ คือ

- หมายหมายแอดเดรสต้นทาง และแอดเดรสปลายทางเป็นค่าเดียวกัน คือ เป็นแอดเดรสของเครื่องเป้าหมาย
- หมายเลขพอร์ตต้นทางเท่ากับหมายเลขพอร์ตปลายทาง
- SYN Flag ถูกตั้งเสมือนขอเริ่มต้นการเชื่อมต่อ

โดยปกติเมื่อมีการส่งสัญญาณ SYN มาก็ต้องมีการตอบกลับไปด้วยสัญญาณ SYN ACK ดังนั้นในที่นี้การตอบกลับจะตอบไปที่เครื่องเดิม ซึ่งในกรณีนี้ไม่มีกำหนดอยู่ในโพรโตคอลว่าควรทำอย่างไร โฮสต์จึงพยายามตอบสนองตามข้อกำหนดเท่าที่มีอยู่โดยการตอบกลับไปที่ไอพีแอดเดรส และพอร์ตต้นทางที่ถูกบุกรุกมา นั่นหมายถึงการตอบกลับเข้ามายังตัวเอง ซึ่งจะทำให้มีการตอบกลับไปมาของทีซีพีวนรอบอยู่ในตัวเองด้วยความเร็วสูง ทำให้คอมพิวเตอร์ต้องใช้ทรัพยากรที่มีอยู่ทั้งหมดเพื่อคอยจัดการกับทีซีพีที่ตอบกลับไปกลับมาจนไม่สามารถทำงานอื่นได้อีก ทำให้ต้องรีเซตเครื่องใหม่เพื่อหยุดการวนรอบของมัน

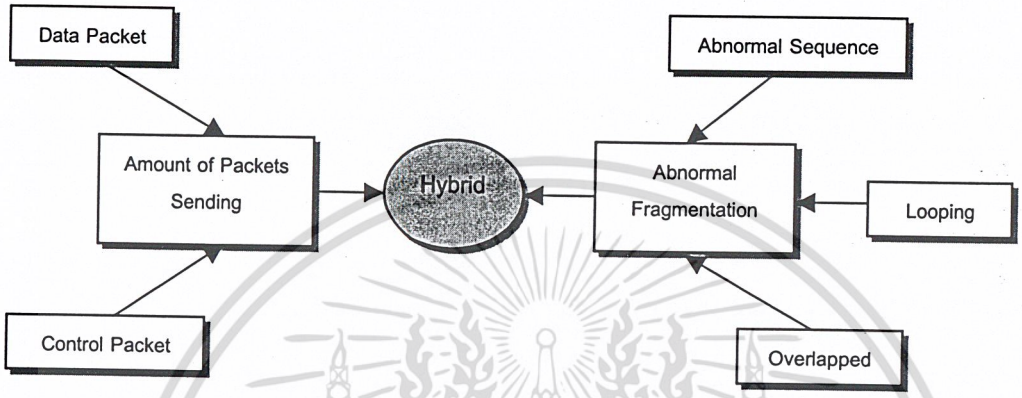
แพ็กเก็ตประเภทนี้เกิดจากการปลอมไอพีแอดเดรส ซึ่งถ้ามีการจัดการป้องกันการปลอมที่ดีพอก็สามารถป้องกันได้ แต่การป้องกันการปลอมนั้นสามารถบังคับใช้ได้ผลกับการปลอมข้าม เน็ตเวิร์กเท่านั้น ถ้าเป็นการปลอมในแชร์โดเมน(share domain)เดียวกันจะไม่สามารถทำได้ ซึ่งในปัจจุบันมี

รูปที่4-7 รูปแสดงการโจมตีด้วย Land Attack

การปรับปรุง ทีซีพีสแต็กให้รัดกุมขึ้นจนการโจมตีแบบนี้ไม่เป็นผลกับคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการที่ ได้รับการแก้ไขแล้ว

4.2.1.3 แบบผสม (Hybrid)

คือ การโจมตีที่อาศัยวิธีการผสมกันระหว่างสามแบบแรกที่ได้กล่าวมาแล้ว ซึ่งทำให้เกิดผล เสียหายต่อระบบมากยิ่งขึ้น



รูปที่ 4-8 *แสดงแผนภูมิแสดงประเภทของการโจมตีเพื่อให้ปิดบริการสำหรับสแต็กทีซีพี/ไอพี*

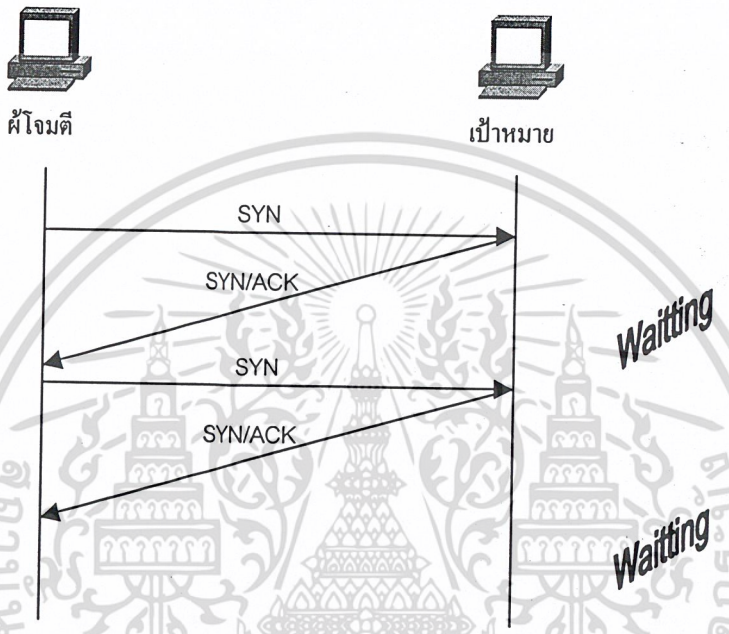
4.2.2ประเภทอยู่ในชั้นทรานสปอร์ต หรือชั้นอินเทอร์เน็ต

4.2.2.1 การส่งแพ็กเก็ตสำหรับควบคุม (Control Packets)

ตัวอย่างของการโจมตีแบบนี้ ได้แก่ การทำ SYN Flooding ซึ่งปกติการเชื่อมต่อแบบ 3-way handshake เป็นไปตามลักษณะที่ได้อธิบายในหัวข้อ 2.4 แต่ในการโจมตีลักษณะนี้ใช้วิธีทำให้การทำ 3-way handshake ไม่สมบูรณ์ กล่าวคือ เครื่องที่ขอบริการส่งสัญญาณ SYN ไป แต่เมื่อได้รับสัญญาณ ACK จากเครื่องที่ให้บริการแล้ว ไม่ส่งสัญญาณ SYN ตอบกลับไป ทำให้เครื่องที่ให้บริการต้องเปิดการเชื่อมต่อรอการตอบกลับ ดังรูปที่ 3-9 ซึ่งการเปิดการเชื่อมต่อรอเอาไว้นี้ต้องใช้ทรัพยากรของระบบส่วนหนึ่ง และหากมีการส่งสัญญาณในลักษณะนี้มากๆ และทรัพยากรของระบบมีไม่เพียงพอ อาจทำให้ระบบไม่สามารถให้บริการอย่างอื่น หรือให้บริการกับผู้ร้องขอรายอื่นได้

ตัวอย่างของการโจมตีแบบนี้ ได้แก่ การทำ SYN Flooding ปกติการเชื่อมต่อแบบ 3-way handshake เป็นไปตามลักษณะที่ได้อธิบายไปแล้ว แต่ในการโจมตีลักษณะนี้ใช้วิธีทำให้การทำ 3-way handshake ไม่สมบูรณ์ กล่าวคือ เครื่องขอบริการส่งสัญญาณ SYN ไป แต่เมื่อได้รับสัญญาณ ACK จาก

เครื่องที่ให้บริการแล้ว ไม่ส่งสัญญาณ SYN ตอบกลับไป ทำให้เครื่องที่ให้บริการต้องเปิดการเชื่อมต่อรอการตอบกลับ ดังรูปที่ 3-9 ซึ่งการเปิดการเชื่อมต่อรอเอาไว้นี้ต้องใช้ทรัพยากรของระบบส่วนหนึ่ง และหากมีการส่งสัญญาณในลักษณะนี้มากๆ และทรัพยากรของระบบมีไม่เพียงพอ อาจทำให้ระบบไม่สามารถให้บริการอย่างอื่น หรือให้บริการกับผู้ร้องขอรายอื่นได้



รูปที่ 4-9 แสดงการส่งแพ็กเก็ตแบบ SYN Flood

สำหรับการโจมตีแบบ SYN Flood **จะทำให้เกิดแพ็กเก็ตในระบบเครือข่ายในลักษณะดังรูปที่ 3-10**

10:09:43.137 10.0.0.1 >isag11.ce.kmitl.ac.th.80: S 399259715:399259715(0)
10:09:43.139 10.0.0.2 > isag11.ce.kmitl.ac.th.80: S 399259715:399259715(0)
10:09:43.143 10.0.0.3 > isag11.ce.kmitl.ac.th.80: S 399259715:399259715(0)
10:09:43.149 10.0.0.4 > isag11.ce.kmitl.ac.th.80: S 399259715:399259715(0)
10:09:43.158 10.0.0.5 > isag11.ce.kmitl.ac.th.80: S 399259715:399259715(0)
10:09:43.165 10.0.0.6 > isag11.ce.kmitl.ac.th.80: S 399259715:399259715(0)
10:09:43.177 10.0.0.7 > isag11.ce.kmitl.ac.th.80: S 399259715:399259715(0)
10:09:43.185 10.0.0.8 > isag11.ce.kmitl.ac.th.80: S 399259715:399259715(0)

รูปที่ 4-10 แพ็กเก็ตที่เกิดขึ้นจากการโจมตีแบบ SYN Flood

ในปัจจุบันนี้การโจมตีแบบ SYN Flood ยังเป็นการโจมตีที่ได้ผลและหาทางป้องกันได้ยาก เนื่องจากยากที่จะแยกลักษณะของแพ็กเก็ตที่ใช้ในการโจมตีกับแพ็กเก็ตที่ขอเริ่มต้นเชื่อมต่อทั่วไป นอกจากนี้ไฟร์วอลล์หรือเราเตอร์ทั่วไปยังไม่สามารถป้องกันการโจมตีประเภทนี้ได้อย่างสมบูรณ์ หนทางที่เป็นไปได้คือการใช้ระบบตรวจจับผู้บุกรุกทางระบบเครือข่ายทำการตรวจจับการโจมตี เพื่อนำข้อมูลจากการโจมตีกลับไปตั้งค่าอุปกรณ์เราเตอร์หรือไฟร์วอลล์เพื่อป้องกันการโจมตีมายังเซิร์ฟเวอร์

4.2.2.2 การสำรวจระบบ

เป็นการสำรวจเน็ตเวิร์กโดยละเอียดเพื่อให้ได้มาซึ่งข้อมูลของทรัพยากรทุกๆอย่างที่มีอยู่บน เน็ตเวิร์กนั้น ไม่ว่าจะเป็นจำนวนโฮสต์ ลักษณะการเชื่อมต่อกันของโฮสต์ อุปกรณ์อื่นๆ ที่ไม่ใช่คอมพิวเตอร์ เส้นทางการเดินทางของแพ็กเก็ต ช่องทางการสื่อสารกับเน็ตเวิร์กภายนอก การได้มาซึ่งข้อมูลรายละเอียดเหล่านี้ยิ่งมากก็ยิ่งเป็นประโยชน์สำหรับผู้บุกรุก

4.2.2.2.1 Ping Sweep

การสำรวจวิธีนี้เป็นพื้นฐานเบื้องต้นปกติไม่ว่าจะเป็นผู้บริหารเครือข่ายระบบเองหรือผู้บุกรุกก็ใช้การพิจารณาจากแพ็กเก็ตที่ปรากฏขึ้นในเน็ตเวิร์กเพียงอย่างเดียวนั้นยากที่จะแยกแยะได้ว่าเป็นการกระทำที่มุ่งร้ายหรือไม่ แต่อย่างน้อยที่สุดก็เป็นสัญญาณเตือนเบื้องต้นในการทำการตรวจสอบต่อไปว่าว่าผู้กระทำมีจุดมุ่งหมายอย่างไร วิธีการสำรวจนี้ใช้การส่ง ICMP Echo Request ไปยังโฮสต์ทุกตัวใน เน็ตเวิร์ก เพื่อสำรวจดูว่าในเน็ตเวิร์กเป้าหมายนั้น มีโฮสต์ใดที่เปิดใช้งานอยู่บ้าง เมื่อโฮสต์ใดก็ตามได้รับ ICMP Echo Request เข้ามาก็จะต้องตอบกลับไปด้วย ICMP Echo Reply การตอบกลับมานี่เองจะเป็นสิ่งยืนยันได้ว่าโฮสต์นั้นเปิดใช้งานอยู่ การจะสังเกตว่า ICMP Echo Request แพ็กเก็ตนั้นต้องสงสัยว่าจะเป็นการสำรวจเน็ตเวิร์กหรือไม่ มิอาจพิจารณาได้จากแพ็กเก็ตเดียว จำเป็นต้องพิจารณาจากรูปแบบและความต่อเนื่องของหลายแพ็กเก็ต โดยจุดที่จะสามารถระบุได้ว่ามีความเป็นไปได้สูงคือ

1. แพ็กเก็ตนั้นมาจากที่เดียวกันและส่งไปยังโฮสต์ปลายทางหลายๆโฮสต์ ซึ่งถ้าไอพีแอดเดรส มาจากภายนอกเน็ตเวิร์ก ก็มีความเป็นไปได้สูงที่จะเป็นการสแกนที่มุ่งร้าย เพราะบุคคลภายนอกไม่ควรสำรวจเน็ตเวิร์กผู้อื่นโดยไม่มีหน้าที่ และโดยพลการ

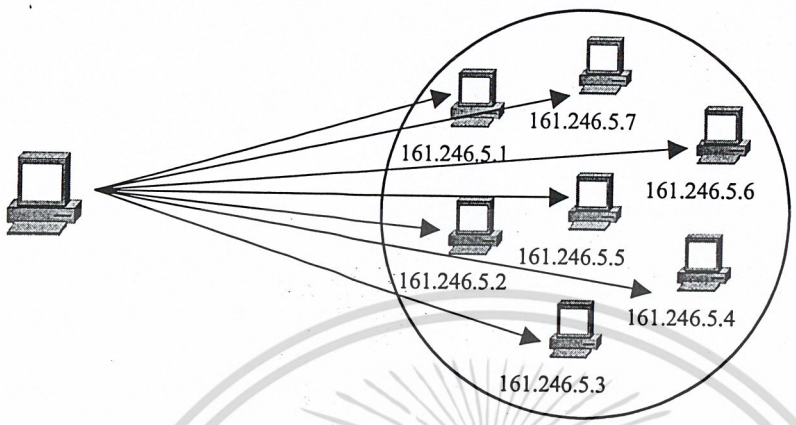
2. ช่วงเวลาระหว่างแพ็กเก็ตมีค่าน้อยมาก ปัจจัยในส่วนของเวลาระหว่างแพ็กเก็ตนั้นจะบอกได้ว่าแพ็กเก็ตเหล่านั้นถูกส่งมาจากคำสั่ง Ping ปกติหรือถูกส่งมาจากเครื่องมือที่ใช้สำหรับสแกนโดยเฉพาะ หากระยะเวลาระหว่างแพ็กเก็ตนั้นมีค่าน้อยกว่า 0.5 วินาที ให้สันนิษฐานได้ว่าเป็นแพ็กเก็ตที่มาจากเครื่องมือแน่นอน

4.2.2.2.2 การสแกนพอร์ต

จากความสำคัญของพอร์ตที่ทีซีพีใช้หมายเลขพอร์ตเพื่อระบุว่าข้อมูลที่ส่งเข้ามานั้นเป็นของแอพพลิเคชั่นใด และแอพพลิเคชั่นต่างๆ ก็จะเลือกใช้พอร์ตหมายเลขต่างๆ กัน เช่น FTP ใช้พอร์ตหมายเลข 21, SMTP ใช้พอร์ตหมายเลข 25 เป็นต้น เมื่อแอพพลิเคชั่นเลือกพอร์ตใดมาใช้งานแล้วก็มีหน้าที่คอยดูว่ามีการติดต่อมาที่พอร์ตของตนหรือไม่ หากมีก็ทำการตอบรับกลับไป ด้วยความสำคัญของพอร์ตนี้เองพอร์ตจึงเป็นเป้าหมายของผู้บุกรุก เพื่อที่จะรู้ได้ว่ามีแอพพลิเคชั่นใดบ้างที่ทำงานอยู่บนโฮสต์ โดยปกติทั่วไปแล้วแอพพลิเคชั่นแต่ละชนิดที่เปิดให้บริการอยู่จะใช้หมายเลขพอร์ตที่ตายตัวและรู้จักกันโดยทั่วไป ดังนั้นเมื่อทำการสแกนแล้วก็จะนำผลมาเปรียบเทียบกับมาตรฐาน

การสแกนพอร์ต เป็นการสำรวจแต่ละโฮสต์ โดยมีขอบเขตเฉพาะโฮสต์เพียงตัวเดียว เป็นการส่งสัญญาณไปสอบถามยังทุกๆพอร์ตที่มีอยู่บนโฮสต์ ทั้ง ทีซีพี และ ยูดีพี เพื่อตรวจสอบว่ามีการเปิดให้บริการอะไรบ้างบนโฮสต์นั้น ซึ่งนั้นหมายถึงว่ามีแอพพลิเคชันประเภทใดอยู่บ้าง เทคนิคต่างๆ ที่นำมาใช้

เพื่อการสแกนพอร์ตนั้นล้วนเป็นการดัดแปลงข้อกำหนดในโพรโตคอลมาใช้งานทั้งสิ้น อาจจะมีบางส่วนที่ใช้ช่องว่างที่ไม่มีกำหนดไว้ในโพรโตคอลเพื่อทำให้ได้ผลลัพธ์มาในที่สุด ดังจะอธิบายแต่ละวิธีต่อจากนี้



รูปที่ 4-11 แสดงการสแกนของผู้บุกรุก

TCP SYN Scan

วิธีนี้ผู้สแกนจะทำการส่ง SYN แพ็กเก็ต (เซตค่าแฟล็ก SYN ไว้เป็น 1) เพื่อทำการติดต่อโดยตรงกับเป้าหมายโดยไม่ผ่านระบบปฏิบัติการ และรอผลการตอบรับของเป้าหมายกลับมา ซึ่งหากเป้าหมายทำงานอยู่ก็จะตอบกลับมาด้วย SYN ACK (เป็นแพ็กเก็ตที่ เซตค่าแฟล็ก SYN และ ACK ไว้เป็น1) หรือหากไม่มีแอพพลิเคชันทำงานอยู่จะตอบกลับมาด้วย RST การสแกนแบบนี้หากตรวจสอบบนโฮสต์ เป้าหมายจะพบว่ามีการขอเชื่อมต่อเข้ามา แต่ไม่สามารถเปิดการติดต่อได้สำเร็จ เทคนิคนี้บางครั้งถูกเรียกว่า half-open scanning คือไม่สามารถทำ 3-way handshake ได้ จึงไม่มีการเชื่อมต่อใดๆเกิดขึ้นระหว่างเครื่องผู้สแกน กับเครื่องที่ถูกสแกน

FIN Scan

เป็นการส่ง FIN แพ็กเก็ตไปยังเป้าหมาย โดยที่เครื่องเป้าหมายก็จะยังตอบแพ็กเก็ตนั้นกลับไป แม้จะไม่มีการสื่อสารใดๆมาก่อนก็ตาม ซึ่งโดยปกติแล้วแพ็กเก็ตที่เซตค่า FIN เป็น 1 จะเป็นแพ็กเก็ตที่ใช้ในการตอบกลับ และการตอบกลับของเครื่องเป้าหมายสำหรับพอร์ตที่เปิดไว้ และพอร์ตที่ไม่ได้เปิดให้บริการก็ไม่เหมือนกัน หากเป็นพอร์ตที่เปิดอยู่ก็จะตอบด้วย FIN ACK กลับไป และหากเป็นพอร์ตที่ไม่ได้เปิดก็จะตอบด้วย RST ACK

SYN/FIN Scan

วิธีนี้จะใช้ ทีซีพี Flag ทั้ง SYN และ FIN พร้อมกัน ซึ่งปกติเป็นแฟล็กที่ไม่มีกำหนดไว้ในโพรโตคอล และจะไม่พบแฟล็กเช่นนี้ในการสื่อสารตามปกติเป็นอันขาด เพราะโดยปกติแล้ว SYN Flag จะใช้เมื่อเริ่มการติดต่อ ส่วน FIN จะใช้เมื่อต้องการยุติการติดต่อ การตอบรับของโฮสต์แต่ละประเภทในกรณีที่ทำงานอยู่นั้นอาจจะแตกต่างกันไป เช่นเป็น SYN ACK หรือ FIN ACK อย่างใดอย่างหนึ่ง ส่วนการตอบรับในกรณีที่พอร์ตปิดจะตอบเหมือนกันคือ RST

Null Scan

วิธีนี้จะไม่ใช้แฟล็กใดๆในการสแกนเลย โดยส่งแพ็กเก็ตที่ไม่มีแฟล็กใดที่ถูกเซตไว้เลยไปยังเป้าหมาย เป็นการเซตแฟล็กทุกค่าให้เป็น 0 หมด ซึ่งแพ็กเก็ตลักษณะนี้จะไม่มีอยู่ในโพรโตคอล โดยทั่วไป

การตอบสนองแพ็กเก็ตที่ไม่ได้อยู่ในโพรโตคอล จะมีการตอบรับที่ต่างกันออกไปตามแต่ประเภทของระบบปฏิบัติการ ดังนั้นนอกจากการใช้แพ็กเก็ตเหล่านี้เพื่อการสแกนพอร์ตแล้วยังสามารถนำแพ็กเก็ตเหล่านี้ไปใช้ในการตรวจสอบระบบปฏิบัติการของเป้าหมายได้อีกด้วย โดยการส่งแพ็กเก็ตที่มีแฟล็กซึ่งไม่อยู่ในข้อกำหนด การส่งแพ็กเก็ตลักษณะนี้ หากพอร์ตของเครื่องเป้าหมายปิดอยู่ การตอบรับจะเป็นการส่ง RST กลับไป

Xmas Scan

จะเป็นการส่งแพ็กเก็ตทีซีพีที่เซตแฟล็ก FIN, Push, URGENT ไปยังพอร์ตเป้าหมายที่เครื่องปลายทาง ซึ่งมักไม่เป็นที่สนใจในการตรวจสอบเท่ากับ SYN-ACK-RST เครื่องปลายทางจะส่งแพ็กเก็ตทีซีพี RST ของพอร์ตที่ปิดอยู่กลับมาให้

UDP Scan

จะส่งแพ็กเก็ตของโพรโตคอลยูดีพี ไปยังพอร์ตเป้าหมาย แต่เนื่องจากยูดีพีมีการจัดการที่แตกต่างจากทีซีพีโดยโพรโตคอลยูดีพี เป็นโพรโตคอลลักษณะคอนเน็กชันเลส(connectionless) ดังนั้นผลลัพธ์ของการสแกนเมื่อพอร์ตเปิดอยู่นั้นจะไม่สามารถคาดการณ์ได้ ขึ้นอยู่กับแต่ละแอพพลิเคชัน และไม่มีมาตรฐานที่เหมือนกันแต่อย่างใด ดังนั้นการสแกนยูดีพี จึงต้องดูผลลัพธ์จาก ICMP เป็นหลัก หากพอร์ตไม่เปิดให้บริการ จะมี ICMP Message ว่า UDP Port Unreachable กลับมา และหากพอร์ตเปิดให้บริการ อาจมีการตอบรับหรือไม่ และอย่างไร จะขึ้นอยู่กับการทำงานของแอพพลิเคชันที่เปิดพอร์ตนั้น แต่ที่แน่ๆคือจะไม่มี ICMP Message กลับมา

4.2.3 ประเภทอยู่ในชั้นแอพพลิเคชัน

ส่วนใหญ่เกิดจากการใช้จุดอ่อนหรือข้อผิดพลาดของแอพพลิเคชันที่เครื่องเป้าหมายใช้อยู่ในการโจมตีเครื่องเป้าหมายเอง ไม่ว่าจะเป็นจุดอ่อนของระบบปฏิบัติการ หรือข้อผิดพลาดของซอฟต์แวร์ก็ตาม โดยสามารถแบ่งออกได้เป็นประเภทดังนี้

บัฟเฟอร์โอเวอร์โฟล (Buffer Overflow) เป็นการใส่ข้อมูลอินพุตที่มีขนาดใหญ่เกินกว่าที่โปรแกรมกำหนดไว้ให้โปรแกรม ทำให้เกิดความผิดพลาดในการทำงานได้ ตัวอย่างเช่น กำหนดรับ ชื่อผู้ใช้(User name) ได้ 256 ตัว ซึ่งโดยปกติไม่น่าจะเกิดกรณีที่ใส่ได้ถึงจำนวน แต่ในกรณีของการต้องการบุกรุก อาจใส่ไปถึง 300 ตัว ซึ่งอาจทำให้โปรแกรมทำงานผิดพลาด ซึ่งก็คือเป็นไปตามความต้องการของผู้บุกรุกนั่นเอง

การส่งคำสั่งที่มีลักษณะบุกรุก (Unexpected Combination) เนื่องจากการที่โปรแกรมมักมีลักษณะเป็นการประกอบด้วยโค้ดหลายๆเลเยอร์ ซึ่งรวมถึงเลเยอร์ล่างสุดซึ่งก็คือ ระบบปฏิบัติการ ผู้บุกรุกจะอาศัยจุดนี้ส่งอินพุตที่ไม่มีความหมายสำหรับเลเยอร์หนึ่ง แต่มีความหมายกับอีกเลเยอร์หนึ่งเข้าไปในโปรแกรม ตัวอย่างที่เห็นได้ชัดได้แก่ ภาษา Perl ซึ่งมักจะส่งอินพุตต่อไปให้โปรแกรมอื่นเพื่อดำเนินการต่อ ซึ่งถ้ามีการส่ง "| mail < /etc/passwd "จะทำให้เกิดการส่งไฟล์พาสเวิร์ดไปทางเมล์ได้ เนื่องจาก Perl จะให้ระบบปฏิบัติการเรียกโปรแกรมขึ้นมาจากอินพุตที่ส่ง โดยที่ระบบปฏิบัติการเองจะเห็น | pipe เป็นการเรียกโปรแกรมเมล์ด้วย

อินพุตที่ผิดปกติ (Unhandled Input) โปรแกรมส่วนมากเขียนขึ้นให้สามารถจัดการกับอินพุตที่ถูกต้องได้ โดยโปรแกรมเมอร์มักจะละเลยในส่วนของอินพุตอื่นๆที่ผิด ซึ่งจะเป็นช่องให้ผู้บุกรุกสามารถใช้ประโยชน์ได้

ตัวอย่างของการบุกรุกในชั้นแอพพลิเคชัน

- สคริปต์ซีจีไอ (CGI Script) : ซีจีไอเป็นโปรแกรมที่รู้กันถึงความไม่ปลอดภัยของมันเป็นทางผ่านในการเข้าไปโจมตีจุดอ่อนในระบบส่วนอื่น โดยการส่งผ่านอินพุตที่ผิดรูปแบบเข้าไปเพื่อจุดประสงค์ในการไปสั่งการโดยตรงกับระบบ เช่น อาจให้เมล์ส่งไฟล์ของพาสเวิร์ดไปให้กับผู้บุกรุกได้ สคริปต์ซีจีไอที่เป็นที่รู้จักกันดี เช่น Phf.cgi , text counter , guestbook , webdist.cgi, files.pl เป็นต้น ซึ่งถ้าพบว่ามีการพยายามใช้หนึ่งในสคริปต์ซีจีไอเหล่านี้ หรือทั้งหมดโดยที่เราไม่ได้เป็นผู้ใช้เอง ก็อาจเป็นได้ว่ามีความพยายามบุกรุกเกิดขึ้นแล้ว

- การโจมตีเว็บเซิร์ฟเวอร์ (Web server attack) เว็บเซิร์ฟเวอร์จำนวนมากที่เขียนขึ้นเอง ซึ่งรวมถึง IIS 1.0 และ Netware 2.x มีช่องโหว่อยู่ซึ่งชื่อไฟล์สามารถรวมชุดของ "../" ในชื่อพาธเพื่อที่จะย้ายไปที่อื่นในระบบไฟล์ ทำให้สามารถเข้าถึงไฟล์ใดก็ได้ที่ต้องการ และอีกหนึ่งช่องว่างที่พบบ่อยก็คือบัฟเฟอร์โอเวอร์โฟลในฟิลด์ที่ต้องการ หรือในฟิลด์ HTTPอื่น

- Sendmail เป็นโพรเซสที่รันอยู่ในแบ็กกราวนด์ ช่องโหว่ส่วนใหญ่ของมันมักมีสาเหตุจากปัญหาของรีโมตบัฟเฟอร์โอเวอร์โฟลว์และการตรวจสอบอินพุตที่ไม่ดีพอ หนึ่งในช่องโหว่ยอดนิยมได้แก่ช่องโหว่เรื่องการส่งไปป์ (pipe) ของ Sendmail ซึ่งอยู่ในเวอร์ชัน 4.1 ซึ่งจะทำให้สามารถส่งไปป์คำสั่งที่ต้องการเอ็กซิคิวต์เข้าไปยังโพรเซส Sendmail ได้โดยตรง คำสั่งใดๆที่อยู่ต่อจากคำสั่ง DATA จะถูกเอ็กซีคิวต์ (execute) โดย Sendmail ภายใต้สิทธิพิเศษของ bin

- การเจาะระบบทาง IIS 5 มีปัญหาเรื่อง Translate:f โดยผู้บุกรุกส่งอินพุตที่คาดไม่ถึงไปยังเซิร์ฟเวอร์ เพื่อให้มันส่งไฟล์บางไฟล์ซึ่งปกติไม่ควรเปิดเผย โดยใช้โพรโตคอลทีซีพี โดยการส่ง HTTP Get Request ที่ไม่เหมาะสมไปยังเว็บเซิร์ฟเวอร์เพื่อให้เว็บเซิร์ฟเวอร์เอ็กซิคิวต์สคริปต์ไฟล์บางสคริปต์ให้ แนวคิดของ request ดังกล่าวคือ การส่งเฮดเดอร์ (HTTP Header) พิเศษที่มี Translate:f อยู่ส่วนท้ายของไฟล์ และมีเครื่องหมาย "\" ต่อท้ายเข้าไปใน URL ที่ขอร้องไป โดยการส่งเท็กซ์ไฟล์ที่ประกอบด้วยสตริงข้อความต่างๆเข้าไปป์ ต่อเข้าไปยังเซิร์ฟเวอร์เป้าหมายก็จะได้ข้อมูลที่จริงๆแล้วไม่ต้องการเปิดเผยไปได้

- Server Side Include (SSIs) เป็นกลไกที่ทำให้เว็บเซิร์ฟเวอร์ให้บริการได้แบบโต้ตอบ ในเวลาจริง โดยที่ไม่ต้องอาศัยโปรแกรม บ่อยครั้งที่นักพัฒนาเว็บมักใช้มันเพื่อการเรียนรู้วันเวลาของระบบอย่างรวดเร็วหรือเพื่อเอ็กซิคิวต์คำสั่งที่เครื่องเว็บเซิร์ฟเวอร์และประเมินดูเอาต์พุตสำหรับการตัดสินใจในการควบคุมโฟลว์ของโปรแกรม ฟีเจอร์ต่างๆของ SSI (เรียกว่า tags)จำนวนมากได้แก่ echo,include,fsize,flastmode,exec,config,odbc,email,if,goto,label, และ break tags ที่มีประโยชน์ที่สุดกับผู้บุกรุกก็คือ include, exec และ mail เทคนิคการโจมตีได้ถูกสร้างขึ้นโดยการเพิ่มโค้ดที่ใช้ tags ของ SSI เข้าไปในฟิลด์เว็บเพจที่จะต้องถูกประเมินค่าโดยเว็บเซิร์ฟเวอร์ ทำให้ผู้บุกรุกเอ็ก

ซิคิวต์คำสั่งที่ต้องการบนเว็บเซิร์ฟเวอร์ได้และได้รับการเข้าถึงตัวเซิร์ฟเวอร์โดยตรง ตัวอย่างเช่น โดยการเพิ่ม SSI tag เข้าไปในฟิลด์ first name หรือ last name เมื่อมีการสร้างแอคเคานต์ เว็บเซิร์ฟเวอร์จะประเมินนิพจน์นั้นและพยายามที่จะรันมัน SSI tag ต่อไปนี้จะส่งกลับ xterm มาให้ผู้บุกรุก

<!-#exec cmd="/usr/X11R6/bin/xterm –display attacker:0 &"->

- ช่องโหว่ของ Irix CGI โดยบนระบบ Irix นั้น Outbox Environment Subsystem ของมันได้รวบรวมเอาโปรแกรมจำนวนมากที่มีจุดอ่อนต่อการโจมตีด้วยการตรวจสอบอินพุตที่ไม่เหมาะสม webdist.cgi ซึ่งเป็นตัวจัดการเอ็นจิน (handler) และ wrap scripts ที่ให้มากับ Irix 5.x และ 6.x จะอนุญาตให้ผู้บุกรุกส่งโลคอลคอมมานด์ (local command) ไปยังสคริปต์และทำให้มันถูกเอ็กซิคิวต์ที่เครื่องเว็บเซิร์ฟเวอร์ได้ URL ต่อไปนี้สามารถให้เพื่อวิวดูไฟล์ passwd ของยูนิกซ์ได้ (ถ้าเว็บเซิร์ฟเวอร์นั้นมีสิทธิเพียงพอ)

http://192.168.51.101/cgi-bin/handler/something;cat<tab>/etc/passedl?data=Download<tab>HTTP/1.0

- NetWare Perl เป็นช่องโหว่ดั้งเดิมที่ถ้าไม่ได้ใช้เวอร์ชันก่อนหน้าของเน็ตแวร์ 4.x หรือ IntraNetware อยู่ คุณก็จะไม่ได้รับผลกระทบจากช่องโหว่นี้ ซึ่งช่องโหว่นี้จะทำให้ผู้บุกรุกสามารถเอ็กซิคิวต์เพิร์ลสคริปต์จากที่ไหนๆก็ได้ รวมทั้งโฮมไดเรกทอรีของผู้ใช้ หรือไดเรกทอรีทั่วๆไปอย่างเช่น login และ mail อันตรายที่เกิดก็คือ ผู้บุกรุกสามารถสร้างเพิร์ลสคริปต์เพื่อนำเอาข้อมูลในไฟล์สำคัญต่างๆ ขึ้นมาโชว์ในเว็บเบราเซอร์ได้ ตัวอย่างเช่น ไฟล์ autoexec.ncf หรือ ldremote.ncf ซึ่งเป็นไฟล์ที่เก็บรหัสผ่านที่ต้องระบุก่อนจะได้รับอนุญาตให้รัน rconsole เพื่อจัดการระบบจากระยะไกล

4.3 โปรแกรมที่ใช้โจมตีเพื่อให้ปิดบริการ

โปรแกรมที่ใช้ในการโจมตีเพื่อให้ปิดบริการนี้เกิดขึ้นมากมาย และนับวันจะเพิ่มรูปแบบมากขึ้นเรื่อยๆ แต่โปรแกรมในปัจจุบันยังคงมีรูปแบบการโจมตีไม่มากไปกว่าที่ได้กล่าวมาแล้ว ซึ่งโปรแกรมต่างๆที่ได้ศึกษามีดังตารางที่ 4-1

ชื่อโปรแกรม	ประเภท	โพรโตคอลที่ใช้	ลักษณะการโจมตี	ระบบที่มีปัญหา
1. syncflood	ส่งแพ็กเก็ตสำหรับการควบคุมปริมาณมาก	TCP	เป็นการส่งสัญญาณ SYN ไปขอเปิดการเชื่อมต่อ แล้วเมื่อได้รับ ACK ก็ไม่ส่งสัญญาณ SYN กลับไป ดังที่ได้อธิบายมาแล้ว	- Windows 95
2. oshare	แฟร็กเมนเตชันผิดปกติ (แพ็กเก็ตเหลื่อมล้ำกัน)	IP	เป็นการส่งแพ็กเก็ตที่เหมือนกันมาที่เครื่องเป้าหมาย ทำให้แพ็กเก็ตที่ส่งมาซ้อนทับกัน	- Windows 95/98 - Windows NT 4 + Service Pack 5 ลงมา
3. land	แฟร็กเมนเตชันผิดปกติ	IP	เป็นการส่งแพ็กเก็ตที่มีแอดเดรสต้นทางและปลายทางเป็นค่าเดียว	- Windows 95

	(ส่งแพ็กเก็ตแบบวนลูป)		กัน ทำให้เกิดการส่งแบบวนลูป	
4. smurf	แพ็กเก็ตปริมาณมาก	ICMP	เป็นการส่ง ICMP echo (ping) traffic จำนวนมาก โดยการ Broadcast ไปยังเครื่องเป้าหมาย	- Windows 95
5. opentear	แฟร็กเมนเตชันผิดปกติ (ต้องรอแพ็กเก็ตก่อนหน้า)	UDP	เป็นการสร้างแพ็กเก็ตสุดท้ายมาเพื่อหลอกให้ระบบเป้าหมายรอแพ็กเก็ตก่อนหน้านี้	- Windows 95
6. pimp	แฟร็กเมนเตชันผิดปกติ (รีแอสเซมเบิ้ลแล้วมีปัญหา)	IP, IGMP	เป็นการสร้างแพ็กเก็ตที่เมื่อรีแอสเซมเบิ้ลแล้วได้ข้อมูลที่ไม่มีความหมาย เกิดเป็นขยะในระบบ	- Windows 95
7. targa	แฟร็กเมนเตชัน (แพ็กเก็ตเหลื่อมล้ำกัน)	IP	เป็นการส่งแพ็กเก็ตที่เหมือนกันมาที่เครื่องเป้าหมาย ทำให้แพ็กเก็ตที่ส่งมาซ้อนทับกัน	- Windows 95/98 - Windows NT 4 + Service Pack 5 ลงมา
8. Gin	แพ็กเก็ตสำหรับควบคุมปริมาณมาก	ICMP	เป็นการส่ง ICMP Echo ปริมาณมากไปยังเครื่องเป้าหมาย	- Windows 95/98 - Modem
9. raped	แพ็กเก็ตสำหรับควบคุมปริมาณมาก	TCP	เป็นการส่งสัญญา SYN ปริมาณมากไปยังเครื่องเป้าหมาย	- Windows 95/98
10. stream	แพ็กเก็ตปริมาณมาก	TCP	เป็นการส่งแพ็กเก็ตทีซีพี ปริมาณมากไปยังเครื่องเป้าหมาย	- Windows 95/98 - Modem
11. moya	แพ็กเก็ตสำหรับควบคุมปริมาณมาก	ICMP	เป็นการส่ง ICMP Echo (ping) ปริมาณมากไปยังเครื่องเป้าหมาย	- Windows 98
12. Kox	แฟร็กเมนเตชันผิดปกติ (แพ็กเก็ตเหลื่อมล้ำกัน)	IGMP	ส่งแพ็กเก็ตที่เหลื่อมกันไปยังเครื่องเป้าหมาย ทำให้เครื่องเป้าหมายไม่สามารถประกอบแพ็กเก็ตเหล่านั้นได้	- Windows 98/98SE - Windows 2000 build 2000
13. Sesquipe-daliam	แฟร็กเมนเตชันผิดปกติ	UDP	ส่งแพ็กเก็ตยูดีพี ที่มีแฟร็กเมนเตชันผิดปกติไปยังเครื่องเป้าหมาย	- Windows 98/98SE

14. smurf	แพ็กเก็ตสำหรับควบคุมปริมาณมาก	ICMP	ส่งแพ็กเก็ต ICMP echo (ping) ไปยังเครื่องเป้าหมายเป็นปริมาณมาก	- Windows 95/98
15. WinArp	อยู่ในชั้นแอปพลิเคชัน	ARP	เป็นการส่งแพ็กเก็ตไปยังเครื่องปลายทาง โดยเครื่องปลายทางต้องกดปุ่ม "OK" สำหรับทุกแพ็กเก็ตที่เข้ามา	- Windows 95/98 - Windows NT 4.0 ลงมา
16. winnuke	อยู่ในชั้นแอปพลิเคชัน	NetBIOS (Port 139)	เป็นการส่ง OOB (Out Of Band) data ไปยังพอร์ต 139 (NetBIOS) ทำให้เกิดหน้าจอฟ้าขาดการเชื่อมต่อกับอินเตอร์เน็ต และทำให้เกิดปัญหาในการติดต่อบนเครือข่าย ซึ่งจะเกิดกับ	- WFWG 3.11 - Win95 - WinNT 3.51 - WintNT 4.0 + Service Pack 4 ลงมา

ตารางที่ 4-1 ตัวอย่างโปรแกรมที่โจมตีเพื่อให้ปิดบริการ

ซึ่งเห็นได้ว่าโปรแกรมที่ใช้โจมตีเพื่อให้ปิดบริการสำหรับโพรโตคอลสแต็กทีซีพี/ไอพี มีมากมาย ในที่นี้หลายโปรแกรมเป็นโปรแกรมเก่า จึงไม่สามารถทำลายระบบปฏิบัติการใหม่ๆ ได้ หรือไม่สามารถโจมตีระบบที่ได้อัปเกรดแล้วได้

นอกจากโปรแกรมเหล่านี้แล้ว ยังมีโปรแกรมอื่นๆ ที่ทำให้เกิดการปิดบริการมากมาย และนอกเหนือจากโปรแกรมเหล่านี้ยังมีโปรแกรมที่ทำการโจมตีในชั้นแอพพลิเคชันอีก เช่น Coke, winARP, nuke เป็นต้น

4.4 IP-Spoofing

ในการโจมตีเครือข่ายส่วนมากมักจะมีการปลอมหมายเลขไอพีเป็นส่วนประกอบเสมอ ยกตัวอย่างการปลอมไอพีใน TCP SYN Flooding การปลอมไอพีจะทำให้โฮสต์ที่ถูกโจมตีไม่รู้ว่าเป็นการส่งแพ็กเกจมาจากที่ไหน หรือทำให้เข้าใจผิดเพื่อใช้ในการผ่านไฟล์วอล์ล

การปลอมไอพีคือ การที่แพ็กเกจที่ออกจากโฮสต์ที่ถูกกำหนดหมายเลขไอพีหนึ่งเอาไว้ เช่น เป็น 161.246.6.220 โปรแกรมหรือเครื่องมือบางอย่างสามารถจะทำการสร้างแพ็กเกจขึ้นมาโดยยอมให้ทำการกำหนดค่าของหมายเลขไอพีได้ ซึ่งค่าที่กำหนดนี้อาจจะเป็นค่าที่ถูกต้องหรืออาจจะไม่ถูกต้องก็ได้ เช่น เครื่องหมายเลข ไอพี 161.246.6.220 ทำการกำหนดหมายเลขไอพีต้นทางเป็นไอพี 161.246.6.215 ก็จะทำให้ทางด้านโฮสต์ที่ได้รับแพ็กเกจเข้าใจผิดว่าเป็นแพ็กเกจที่มาจากหมายเลขไอพี 161.246.6.215 ซึ่งจริงๆ แล้วเป็นแพ็กเกจที่ส่งออกมาจากเครื่องไอพีหมายเลข 161.246.6.220

บทที่ 5
Tcpdump

5.1 tcpdump คืออะไร?

Tcpdump คือ โปรแกรมที่ทำหน้าที่จับแพ็กเกจที่วิ่งอยู่ในเครือข่าย โดยโปรแกรม tcpdump จะทำงานร่วมกับเน็ตเวิร์กอินเตอร์เฟสการ์ดในโหมด Promiscuous – mode บนเครื่องที่ได้รับการติดตั้งโปรแกรม tcpdump โดยการทำงานร่วมกันในโหมดนี้ แพ็กเกจทั้งหมดในเครือข่ายจะสามารถถูกโปรแกรม tcpdump จับขึ้นมาได้ เพราะการทำงานของเน็ตเวิร์กอินเตอร์เฟสการ์ดในโหมดนี้จะไม่มีการเปรียบเทียบแอดเดรสปลายทางของข้อมูลกับแอดเดรสของเน็ตเวิร์คอินเตอร์เฟสการ์ด นั่นก็คือทุกๆ แพ็กเกจที่มาถึงยังเน็ตเวิร์คอินเตอร์เฟส
การ์ดจะถูกโปรแกรม Tcpdump จับขึ้นมาได้ทั้งหมดไม่เลือกว่าจะระบุแอดเดรสปลายทางของเน็ตเวิร์กอินเตอร์-เฟสการ์ดไหน ข้อนี้ถือเป็นข้อดีของโปรแกรม tcpdump เพราะว่าสามารถทำให้ดูข้อมูลของแพ็กเกจได้ทั้งเครือข่าย

โปรแกรม Tcpdump จะถูกใช้ในการเฝ้าดูการจราจรในเครือข่ายเพื่อการบริหารเครือข่ายให้มีประสิทธิภาพและแก้ปัญหาต่างๆ ที่เกิดจากการทำงานผิดพลาดของเครือข่ายและยังเป็นเครื่องมือที่ดีในการศึกษาการทำงานของโพรโตคอลรวมถึงการดูแลความปลอดภัยของระบบเครือข่าย

Tcpdump จะทำการพิมพ์ของส่วนหัวของแพ็กเกจที่วิ่งในเครือข่ายออกมาในรูปของเท็กซ์ (text) ทำให้สามารถใช้โปรแกรมที่สามารถประมวลผลข้อมูลเท็กซ์ มาวิเคราะห์แพ็กเกจได้ แต่การรันโปรแกรม tcpdump จะมีข้อกำหนดสิทธิ์ในการรันแตกต่างกันในแต่ละระบบดังนี้

- ภายใต้ระบบ SunOS ที่มี nit หรือ bpf การรัน Tcpdump จะต้องการมีสิทธิ์ การอ่าน (read access) /dev/nit หรือ /dev/bpf
- ภายใต้ระบบ Solaris ที่มี dlpi การรัน Tcpdump จะต้องใช้สิทธิ์การอ่านอุปกรณ์ network pseudo device เช่น /dev/le
- ภายใต้ HP UX ที่มี dlpi การรัน Tcpdump จะต้องใช้สิทธิ์รูท (root)
- ภายใต้ IRIX ที่มี snoop การรัน Tcpdump จะต้องใช้สิทธิ์รูท
- ภายใต้ระบบ BSD จะต้องมีสิทธิ์ในการอ่าน /dev/bpf

5.2 การใช้งานโปรแกรม tcpdump

Tcpdump สามารถจับแพ็กเกจที่วิ่งผ่านสายในเครือข่าย โดยจุดมุ่งหมายของการสร้างโปรแกรมต้องการที่จะหลีกเลี่ยงการนำโปรแกรมไปใช้ในทางที่ผิด เช่น การเก็บพาสเวิร์ด (password) หรือการกระทำเพื่อการเจาะระบบ (Hack) ต่างๆ ผลลัพธ์ของโปรแกรมจึงออกมาเป็นเลขฐานสิบหก (Hex) ในส่วนที่เป็นข้อมูลทั้งหมดของแพ็กเกจ ยกเว้นส่วนที่เป็นส่วนหัวของแพ็กเกจที่จะแสดงในรูปเท็กซ์

5.2.1 OPTION ของโปรแกรม tcpdump

ใช้กำหนดรูปแบบของการทำงานและการแสดงผลดังอธิบายย่อๆ ดังนี้

-a แปลงแอดเดรสไปเป็นชื่อ

-c ออกจากโปรแกรมหลังจากจับแพ็กเกจได้ครบตามจำนวนที่กำหนดไว้

-e พิมพ์แอดเดรสของเน็ตเวิร์คอินเตอร์เฟสการ์ด

-F ใช้เพื่อนำเอาไฟล์ที่เก็บ expression มาเป็นอินพุตของ command line

-i กำหนดเน็ตเวิร์คอินเตอร์เฟสการ์ดที่จะทำการจับแพ็กเกจในกรณีที่มีจำนวนเน็ตเวิร์คอินเตอร์เฟสการ์ดมากกว่าหนึ่งการ์ด ตามปกติถ้าไม่กำหนด tcpdump จะเลือกเอาหมายเลขที่ต่ำที่สุด

-l ทำให้ผลลัพธ์ออกทาง Stdout เช่น ในการจับแพ็กเกจลงไปเก็บไว้ในไฟล์จะไม่สามารถมองเห็นผลลัพธ์ทางหน้าจอได้ จะต้องใช้ -l ร่วมด้วยในกรณีต้องการให้ออกที่หน้าจอ

-n ไม่ต้องแปลงแอดเดรสไปเป็นชื่อ

-N ไม่พิมพ์ domain name ของชื่อโฮสต์ เช่น ถ้า -N ถูกกำหนดชื่อโฮสต์“nic.ddn.mil” จะพิมพ์เพียง “nic”

-p ไม่ต้องการการทำงานในโหมด promiscuous mode

-q พิมพ์ผลลัพธ์โดยเร็ว, พิมพ์ผลลัพธ์ออกมาสั้นๆ

-r อ่านแพ็กเกจจากไฟล์ (ที่สร้างจาก -w) และสามารถอ่านจาก Standard input โดยกำหนดชื่อไฟล์เป็น “-“

-s กำหนดความยาวขนาดของแพ็กเกจที่ต้องการจะจับ

-t ไม่ต้องพิมพ์ timestamp ในแต่ละ แพ็กเกจ

-tt พิมพ์ timestamp ที่ยังไม่ได้จัดฟอร์แมตในแต่ละ แพ็กเกจ

-v เพิ่มรายละเอียดของผลลัพธ์ที่ออกมา เช่น time to live และ type of service

-vv เพิ่มรายละเอียดของผลลัพธ์ โดยจะมากกว่า -v

-w เขียนแพ็กเกจที่จับได้ไฟล์ แทนที่จะออกหน้าจอ โดยจะสามารถพิมพ์ออกมาโดยใช้ -r ในภายหลัง และสามารถเขียนลง Standard output ถ้ากำหนดให้ชื่อไฟล์เป็น “-“

-x พิมพ์แต่ละแพ็กเกจทั้งหมดออกมาในรูปแบบเลขฐานสิบหก โดยจะรวมเอาส่วนของข้อมูลด้วย

5.2.2 EXPRESSION ของโปรแกรม tcpdump

ใช้เพื่อเลือกแพ็กเกจที่จะทำการจับ โดยถ้าไม่กำหนด tcpdump จะทำการจับแพ็กเกจทุกๆ แพ็กเกจแต่ถ้าหากมีการกำหนด expression ก็จะจับเฉพาะแพ็กเกจที่เป็นจริง ตาม expression เท่านั้น โดยจะประกอบด้วย expressionn ต่าง ๆ ดังนี้

type ใช้เพื่อกำหนดชื่อหรือหมายเลขของสิ่งที่ต้องการอ้างอิงถึง

- host : กำหนดชื่อโฮสต์เช่น "host foo"

- net : กำหนดชื่อเครือข่ายเช่น "net 182.3"

- port : กำหนดชื่อหรือหมายเลขของพอร์ตเช่น " port 20 "

ถ้าหากไม่มีการกำหนด type tcpdump จะกำหนดให้เป็น host

Dir ใช้เพื่อกำหนดทิศทางของแพ็กเกจที่ต้องการจับโดยสามารถใช้ and/or ร่วมด้วย

- src : กำหนดต้นทาง type ของแพ็กเกจที่ต้องการ เช่น "src foo"
- dst : กำหนดปลายทาง type ของแพ็กเกจที่ต้องการ เช่น "dst net 128.3"
- src or dst : กำหนดต้นทางหรือปลายทาง type ของแพ็กเกจที่ต้องการ เช่น " src or dst port ftp - data"
-src and dst : กำหนดต้นทางและปลายทาง type ของแพ็กเกจที่ต้องการ เช่น "src and dst net128.3"

ถ้าหากไม่มีการกำหนด dir tcpdump จะกำหนดให้เป็น src or dst

proto ใช้เพื่อกำหนดโพรโตคอลเฉพาะลงไปในของแพ็กเกจที่ต้องการจะจับโดยมีคำที่สามารถใช้ได้ดังนี้

- ether
- fddi
- ip
- arp
- rarp
- decnet
- lat
- sca
- moprc
- mopdi
- tcp
- udp

ถ้าหากไม่มีการกำหนดโพรโตคอล tcpdump จะจับทุก ๆ โพรโตคอล

นอกจากนี้ ยังมี expression พิเศษอีก คือ

- Gateway
- Broadcast
- Less
- Greater
- Arithmetic expression >,<,>=,<=,=,!= และค่าคงที่

รวมทั้งการทำงานไบนารี [+,-,*,/,\,[]

- การเข้าถึงข้อมูลภายในแพ็กเกจโดยมีรูปแบบดังนี้

proto[expr : size]

proto : ชนิดของโพรโตคอล

expr : ตำแหน่งของข้อมูลภายในแพ็กเกจนับเป็นไบต์

Size : จำนวนข้อมูลที่ต้องการมีหน่วยเป็นไบต์

นอกจากนี้ ยังสามารถใช้

( ) กลุ่มของ expression

" ! " or 'not'

"&&" or 'and'

"|| ' or 'or'

โดย '!' or 'not' จะมีความสำคัญ (priority) สูงสุด '&&' '|' มีสำคัญเท่ากันเท่ากัน และมีลำดับเท่ากันและมีสำคัญจากซ้ายไปขวา

ตัวอย่างการใช้ Expression

host ใช้เพื่อกำหนดโฮสต์ของแพ็กเกจที่ต้องการจะจับ

#tcpdump host diamond.ce.kmitl.ac.th จับแพ็กเกจที่เข้าหรือออกจากโฮสต์ diamond.ce.kmitl.ac.th

net ใช้เพื่อกำหนดหมายเลขของเครือข่ายของแพ็กเกจที่ต้องการจับ

#tcpdump net 161.246 จับแพ็กเกจที่มาจากเครือข่ายหมายเลข 161.246 ที่ผ่านมาทางโฮสต์ที่รันโปรแกรม tcpdump

port ใช้เพื่อกำหนดพอร์ตของแพ็กเกจที่ต้องการจับ

#tcpdump port 20 จับเพ็กเกจที่ผ่านเข้าออกทางพอร์ต 20

src,dst ใช้เพื่อกำหนดทิศทางของแพ็กเกจที่ต้องการจับ

#tcpdump src diamond.ce.kmitl.ac.th จับแพ็กเกจที่ต้นทางมาจากโฮสต์ diamond.ce.kmitl.ac.th

#tcpdump dst net 161.246 จับแพ็กเกจที่ปลายทางเป็นเครือข่ายหมายเลข 161.246

proto [expr : size] ใช้เพื่อต้องการเข้าถึงข้อมูลในแพ็กเกจ โดยที่

proto = ether,fddi,ip,arp,rarp,tcp,udp,icmp อย่างใดอย่างหนึ่ง

expr = ไบต์ออฟเซตของข้อมูลในแพ็กเกจ

size = ขนาดของข้อที่ต้องการ มีหน่วยเป็นไบต์

#tcpdump 'ether[0] & 1 != 0' จับแพ็กเกจชนิดอีเธอร์เน็ต (Ethernet) โดยที่ไบต์แรก (expr = 0) ขนาด 1 ไบต์ (ถ้า size ไม่มี tcpdump จะกำหนดให้เป็น 1 ไบต์) and กับ 1 แล้วไม่เท่ากับ 0

#tcpdump 'ip[6:2] & 0x1fff = 0' จับแพ็กเกจชนิดไอพีโดยที่ไบต์ที่ 6 (expr = 6) ขนาด 2 ไบต์ (size = 2) and กับ 0x1fff แล้วไม่เท่ากับ 0

5.3 ตัวอย่างการใช้งานและผลลัพธ์ของโปรแกรม tcpdump

ผลลัพธ์ของโปแกรม tcpdump จะขึ้นอยู่กับโพรโตคอล การกำหนดออปชัน (option) และ experssion ทำให้สามารถกำหนดขอบเขตของข้อมูลที่จะจับและรูปแบบของผลลัพท์ที่นำไปใช้ต่อไป ตัวอย่างข้างล่างนี้เป็นตัวอย่างการใช้คำสั่งและผลของโปรแกรม

```
#tcpdump –e                    ← คำสั่ง
00:43:27.481297 0:80:ad:6:a6:79 0:a0:24:b3:54:e0 ip 114: media02.ce.kmitl.ac.th.1467
> ← บรรทัดที่ 1
as2-13.qualitynet.net.10516: P 30339803:30339863(60) ack 6363488 win 8350 (DF)
  ← บรรทัดที่ 2
00:43:27.481297 2:60:8c:6a:19:7d 0:a0:24:b3:54:e0 ip 88: Zintoo.kmitl.ac.th.1078 >
← บรรทัดที่ 3
```

Chaokhun.kmitl.ac.th.domain: 63163+ (46) ← บรรทัดที่ 4

00:43:39.821297 0:a0:24:b3:54:e0 Broadcast arp 60: ← บรรทัดที่ 5

arp who-has digit06.ce.kmitl.ac.th tell compnet.ce.kmitl.ac.th ← บรรทัดที่ 6

00:43:44.661297 0:20:18:61:82:cf 0:20:18:62:38:aa 0028 60:
a1f60600.00:20:18:61:82:cf.4068 > 19990503.00:00:00:00:00:01.451: ipx-ncp 10

จากข้างบนผลของคำสั่ง tcpdump จะทำการแปลงแพ็กเกจให้ออกมาอยู่ในรูปเท็กซ์ บรรทัดที่ 1 และบรรทัดที่ 2 จะเป็น 1 แพ็กเกจ บรรทัดที่ 3,4 เป็นอีก 1 แพ็กเกจและบรรทัดที่ 5,6 เป็นอีก 1 แพ็กเกจ ในแต่ละชุดของเท็กซ์จะถูกแบ่งออกเป็นส่วนๆ เพื่อแสดงข้อมูลของแต่ละส่วน โดยจะประกอบด้วยเวลา (Time Stamp) ข้อมูลในส่วนหัวของชั้นลิงค์เลเยอร์, ข้อมูลในส่วนหัวของชั้นเน็ตเวิร์คเลเยอร์ และข้อมูลในส่วนหัวของชั้นทรานสปอร์ตเลเยอร์ดังนี้

00:43:27.481297 ← Time Stamp
0:80:ad:6:a6:79 0:a0:24:b3:54:e0 ip 114: ← Link Layer
media02.ce.kmitl.ac.th.1467 > as2-13.qualitynet.net.10516:← Network Layer
P 30339803:30339863(60) ack 6363488 win 8350 (DF)← Transport Layer

การตีความหมายของข้อมูลก็จะแบ่งอธิบายแยกตามแต่ละส่วนดังนี้

5.3.1 เวลา (Time Stamp)

จะเป็นเวลาที่เน็ตเวิร์คอินเตอร์เฟสการ์ดได้รับแพ็กเกจ โดยจะมีรูปแบบอยู่ 2 แบบ คือแบบที่ 1 เวลาที่มีการจัดรูปแบบแล้วโดยจะมีรูปแบบเป็น [ชั่วโมง:นาที:วินาที] และแบบที่ 2 เวลาที่ยังไม่ได้จัดรูปแบบ โดยจะมีรูปแบบเป็น [วินาที] ซึ่งเป็นวินาทีที่เริ่มนับจากวันที่ 1 เดือนมกราคม ค.ศ. 1970 และต้องกำหนดด้วย

ออปชั่น –tt

00:43:27.481297 ← แบบที่ 1

911482869.041368 ← แบบที่ 2

5.3.2 ลิงค์เลเยอร์ (Link Layer)

ในขั้นนี้จะมีโพรโตคอลคือ อีเธอร์เน็ต, FDDI และ SLIP การที่จะให้โปรแกรม tcpdump แสดงรายละเอียดของข้อมูลในขั้นนี้ออกมาจะต้องใช้ –e ผลลัพธ์ของโปแกรมจะมีความแตกต่างกันเล็กน้อยยกตัวอย่างเช่น บนอีเธอร์เน็ต กับ FDDI ผลลัพท์ของแพ็กเกจที่วิ่งบน FDDI จะมีการพิมพ์ “frame control” ออกมาด้วยในขณะที่บนอีเธอร์เน็ตจะไม่มี ตัวอย่างต่อไปนี้เป็นแพ็กเกจที่วิ่งบนเครือข่ายอีเธอร์เน็ต

0:80:ad:6:a6:79 0:a0:24:b3:54:e0 ip 114:มีความหมายดังนี้

0:80:ad:6:a6:79 ← Ethernet Source Address

แสดงแอดเดรสต้นทางของเน็ตเวิร์คอินเตอร์เฟสการ์ด

0:a0:24:b3:54:e0 ← Ethernet Destination Address

แสดงแอดเดรสปลายทางของเน็ตเวิร์คอินเตอร์เฟสการ์ด

ip ← Fram Type

แสดงชนิดของเฟรมนี้คือไอพีซึ่งก็คือโพรโตคอลที่อยู่ในชั้นบนของอีเธอร์เน็ตเฟรมจะประกอบด้วย

ไอพี 0x0800

เออาร์พี 0x0806

อาร์เออาร์พี 0x8305

แอปเปิ้ลทอร์ค 0x809B

แอปเปิ้ลทอร์ค เออาร์พี 0x80F3

เน็ตแวร์ ไอพีเอ็กซ์/เอสพีเอ็กซ์ 0x8137

114: ← Frame Length

แสดงขนาดความยาวของอีเธอร์เน็ตเฟรมซึ่งมีหน่วยเป็นไบต์

5.3.3 เน็ตเวิร์คเลเยอร์ (Network Layer)

ในส่วนนี้จะเป็นข้อมูลในส่วนของโพรโตคอลที่ถูกหุ้มโดยอีเธอร์เน็ตเฟรมชั้นหนึ่งซึ่งเป็นชั้นของเน็ตเวิร์คเลเยอร์จะประกอบไปด้วยโพรโตคอล ไอพี (IP), เออาร์พี (ARP), อาร์เออาร์พี(RARP)

00:45:43.441297 2:60:8c:6a:19:7d 0:a0:24:b3:54:e0 ip 86:← บรรทัดที่ 1

Zintoo.kmitl.ac.th.1108 > Chaokhun.kmitl.ac.th.domain: 63192+ (44) ← บรรทัดที่ 2

00:43:51.921297 0:a0:24:b3:54:e0 0:80:48:ed:74:b0 arp 60: ← บรรทัดที่ 3

arp who-has technician02.ce.kmitl.ac.th (0:80:48:ed:74:b0) tell compnet.ce.kmitl.ac.th ← บรรทัดที่ 4

00:43:51.921297 0:80:48:ed:74:b0 0:a0:24:b3:54:e0 arp 60: ← บรรทัดที่ 5

arp reply technician02.ce.kmitl.ac.th is-at 0:80:48:ed:74:b0 ← บรรทัดที่ 6

จากตัวอย่างข้างบนความหมายของข้อมูลส่วนของเน็ตเวิร์คจะมีความแตกต่างกันตามชนิดของโพรโตคอลซึ่งได้ยกเอาแพ็กเกจของไอพี (บรรทัดที่ 1,2) กับเออาร์พี (บรรทัดที่ 3-6) มาเป็นตัวอย่าง

Zintoo.kmitl.ac.th.1108 > Chaokhun.kmitl.ac.th.domain: ← IP Header

มีความหมายดังนี้

Zintoo.kmitl.ac.th. ← IP Source Address

แสดงแอดเดรสต้นทางของไอพีแพ็กเกจโดยเป็นชื่อหรือหมายเลขไอพี

1108 ← Source Port Number

แสดงหมายเลขพอร์ตต้นทางของไอพีแพ็กเกจโดยเป็นชื่อบริการหรือหมายพอร์ต

> ← Direction

แสดงทิศทางต้นทางกับปลายทางของไอพีแพ็กเกจ

Chaokhun.kmitl.ac.th. ← IP Destination Address

แสดงแอดเดรสปลายทางของไอพีแพ็กเกจโดยเป็นชื่อหรือหมายเลขไอพี

domain: ← Destination Port Number

แสดงหมายเลขพอร์ตปลายทางของไอพีแพ็กเกจโดยเป็นชื่อบริการหรือหมายพอร์ต

arp who-has technician02.ce.kmitl.ac.th (0:80:48:ed:74:b0) tell compnet.ce.kmitl.ac.th ←ARP Request

มีความหมายดังนี้

arp who-has

แสดงชนิดของแพ็กเกจที่ทำการร้องขอเออาร์พี

technician02.ce.kmitl.ac.th (0:80:48:ed:74:b0)

แสดงชื่อหรือหมายเลขไอพีของเครื่องที่ต้องการทราบหมายเลขเน็ตเวิร์กอินเตอร์เฟสการ์ด ในกรณีที่มีข้อมูลอยู่ในหน่วยความจำของเออาร์พี (ARP Cache) จะมีวงเล็บหมายเลขของเน็ตเวิร์คอินเตอร์เฟสการ์ดด้วย

tell compnet.ce.kmitl.ac.th

แสดงชื่อหรือหมายเลขไอพีของเครื่องที่ทำการร้องขอเออาร์พี เพื่อให้เครื่องที่ทราบทำการตอบกลับ

arp reply technician02.ce.kmitl.ac.th is-at 0:80:48:ed:74:b0 ← ARP Reply

มีความหมายดังนี้

arp reply

แสดงชนิดของแพ็กเกจที่ทำการตอบรับการร้องขอเออาร์พี

technician02.ce.kmitl.ac.th is-at 0:80:48:ed:74:b0

แสดงชื่อหรือหมายเลขไอพีและหมายเลขเน็ตเวิร์คอินเตอร์เฟสการ์ด

5.3.4 ทรานสปอร์ตเลเยอร์ (Transport Layer)

ในชั้นนี้จะเป็นข้อมูลของโพรโตคอลถูกไอพีดาต้าแกรมหุ้มอยู่ ซึ่งประกอบด้วย ทีซีพี และ ยูดีพี

00:43:27.791297 0:a0:24:b3:54:e0 0:80:48:ed:74:b0 ip 60: Chaokhun.kmitl.ac.th.3128 > technician02.ce.kmitl.ac.th.1208: S 1637511044:1637511044(0) ack 123689 win 8760 <mss 1460> (DF)

00:43:22.611297 2:60:8c:6a:19:7d 0:a0:24:b3:54:e0 ip 86: Zintoo.kmitl.ac.th.1077 > Chaokhun.kmitl.ac.th.domain: 63162+ (44)

จากตัวอย่างข้างบนจะประกอบด้วย 2 แพ็กเกจ คือ ทีซีพีแพ็กเกจและยูดีพีแพ็กเกจ

S 1637511044:1637511044(0) ack 123689 win 8760 <mss 1460> (DF) ← TCP Header

มีความหมายดังนี้

S ← Flag

แสดงแฟล็กซึ่งใช้ในการติดต่อในส่วนของทีซีพีซึ่งประกอบด้วย S (SYN), F (FIN), R (RST), P (PSH), . (No Flag Set)

1637511044:1637511044(0) ← Sequence Number

แสดง หมายเลข Sequence Number ของแพ็กเกจและจำนวนของข้อมูล (ที่มีหน่วยเป็นไบต์) ที่อยู่ภายในวงเล็บที่เกิดจากผลต่างหมายเลข Sequence Number ตัวหน้ากับตัวหลังโคลอน

ack 123689 ← Acknowledgment Number

แสดง หมายเลข Acknowledgment Number ของแพ็กเกจ ซึ่งแฟล็ก ACK จะถูกแยกออกมาต่างหากไม่รวมอยู่ในส่วนของแฟล็ก S, F, R ,P และ .

win 8760 ← Window Size

แสดง ขนาดของวินโดว์ที่ใช้สำหรับรับข้อมูลที่ทางฝั่งรับสามารถจะรับได้

<mss 1460> ← Maximum Segment Size(MSS)

แสดง ค่าขนาดของทีซีพีเซ็กเมนต์ (TCP header + TCP data) ที่ทางด้านผู้รับไม่ต้องการรับข้อมูลที่มีขนาดใหญ่เกินนี้

(DF) ← Don't Fragment

แสดง การอนุญาตให้มีการแบ่งแพ็กเกจ (Fragment) หรือไม่โดยที่หากมี (DF) จะถือว่าแพ็กเกจนี้ไม่ยอมให้มีการแบ่งแพ็กเกจย่อยออกไปอีก แต่ถ้าหากไม่มีแพ็กเกจอาจถูกแบ่งออกไปอีกเพื่อประโยชน์ในการส่งแพ็กเกจ

สำหรับในยูดีพีจะมีความแตกต่างกันตามบริการที่ยูดีพีให้บริการซึ่งแต่ละบริการจะมีความหมายเฉพาะแยกตามบริการดังตัวอย่าง

rwho packet:

actinide.who > broadcast.who: udp 84

Name server requests :

h2opolo.1538 > helios.domain: 3+ A? ucbvax.berkeley.edu. (37)

Name server responses :

helios.domain > h2opolo.1537: 2 NXDomain* 0/1/0 (97)

ในส่วนนี้จะไม่ขออธิบายเพราะไม่ได้ใช้ในโครงงาน นอกจากนี้ tcpdump ยังสามารถแสดงข้อมูลทั้งแพ็กเกจในรูปของแพ็กเกจเลขฐานสิบหก ในกรณีที่ใช้ออปชัน –x

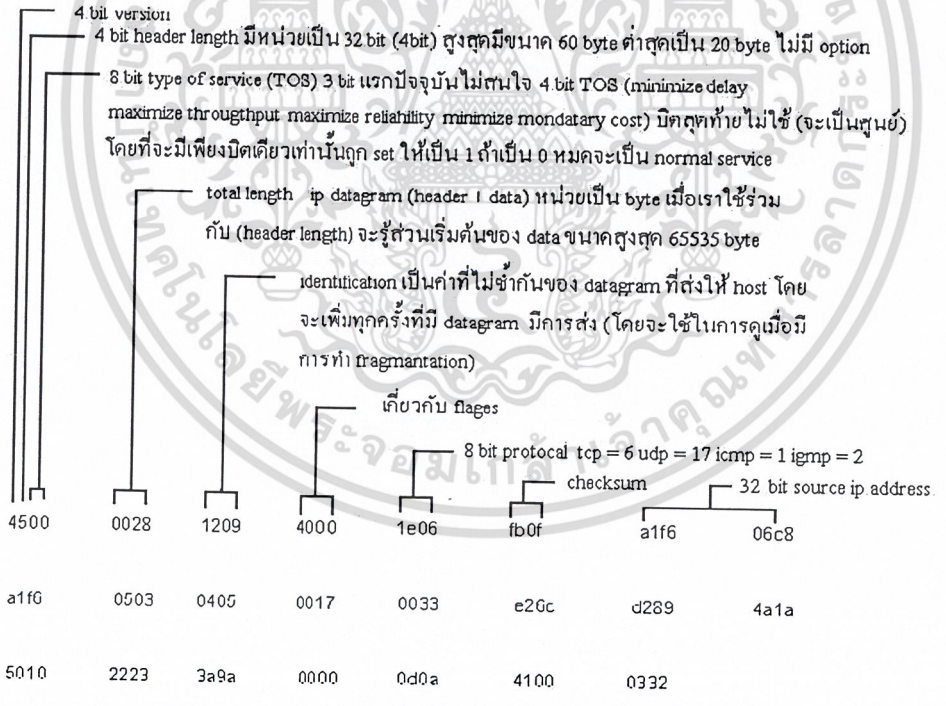
secure3:/home/staff/sintoo# tcpdump –x ← คำสั่ง

tcpdump: listening on eth0

15:44:42.683233 office0.ce.kmitl.ac.th.1029 > secure3.ce.kmitl.ac.th.telnet: . ack 3532212762 win 8739 (DF)

4500 0028 1209 4000 1e06 fb0f a1f6 06c8
a1f6 0503 0405 0017 0033 e26c d289 4a1a
5010 2223 3a9a 0000 0d0a 4100 0332

ข้อมูลในส่วนของเลขฐานสิบหกจะเป็นไอพีแพ็กเกจมีความหมายดังนี้



รูปที่5-1 แสดงความหมายของแพ็กเกจที่ได้จากโปรแกรม tcpdump

บทที่ 6
หลักการทำงานย่อยของโปรแกรม

6.1 หลักการสร้างแพ็คเก็ต (Packet)

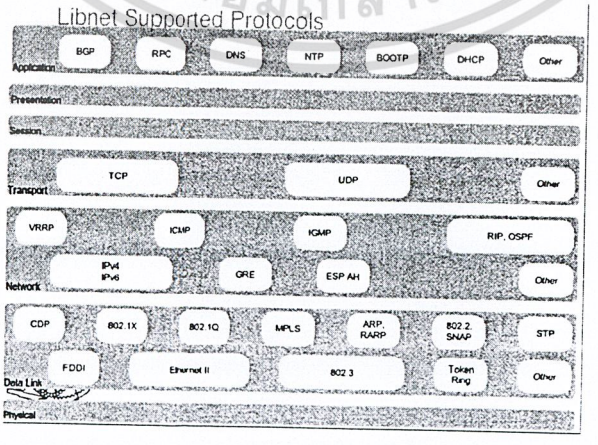
6.1.1 ความหมายและหลักการทำงานของลิปเน็ต (Libnet)

ลิปเน็ตเป็นไลบรารี่ (Library) ของภาษาซีที่เตรียมเอาไว้เพื่อใช้ในการเชื่อมต่อระดับสูง (high-level) เพื่อส่งผ่านแพ็กเก็ตจากตำแหน่งต้นทางปยังอีกฝั่งปลายทาง ซึ่งลิปเน็ตมีตัวอย่างเอกสารอธิบายด้วยไว้อย่างชัดเจนเกี่ยวกับเอพีไอ (API) ที่ใช้ได้อย่างง่ายดายและเรียกใช้ในโปรแกมได้อย่างเหมาะสมซึ่งลิปเน็ตถูกสร้างขึ้นนี้ประกอบด้วยเหตุผล 2 ประการ

- เพื่อสร้างการเชื่อมต่ออย่างง่ายสำหรับโปรแกรมเมอร์ที่ไม่สนใจความยุ่งยากของการทำเน็ตเวิร์กโปรแกรมมิ่ง(Network programming) ระดับต่ำ(low-level) และให้ความสนใจกับการเขียนโปรแกรมของพวกเขาและ การแก้ไขปัญหาต่างมากกว่า
- เพื่อลดความไม่พอใจของเน็ตเวิร์กโปรแกรมเมอร์หลายๆคน ที่ขาดประสบการ์ณที่เป็นมาตรฐาน

ลิปเน็ตเป็นเครื่องมือที่มีประโยชน์ในการเขียนแอฟพลิเคชั่น(Applications) ที่เกี่ยวข้องกับการรักษาความปลอดภัย, เครื่องมือและโมดูล(modules) ต่างๆ ซึ่งการสร้างโปรแกรมที่ทำงานไม่ประสงค์ดีหลายอันก็พัฒนาโปรแกรมจากการใช้ libnet และรวมถึงเครื่องมือที่เกี่ยวกับการรักษาความปลอดภัยหลายโปรแกรมโดยตัวโปรแกรมของลิปเน็ตมีประโยชน์พอสมควรคือสามารถสร้างและส่งแพ็กเก็ตลงไปบน เน็ตเวิร์กได้แต่อย่างไรก็ตามตัวลิปเน็ตก็ไม่สามารถทำในส่วนของการดักจับ(Capture packet) ได้ซึ่งสามารถหาไลบรารี่ลิปพีแคป(Libpcap) มาช่วยทำในส่วนของการดักจับแพ็กเก็ตได้ เนื้อหาในบทนี้ครอบคลุมถึง หัวข้อการจัดการกับไลบรารี่ให้เกิดประโยชน์

ปัจจุบันตัวไลบรารี่ลิปเน็ตเป็นไลบรารี่ของภาษาซีที่ออกแบบมาให้เล็กและง่ายต่อการใช้ โดยจุดประสงค์หลักของตัวลิปเน็ตเองคือการสร้างแพ็คเก็ตปัจจุบันมีโปรดตคอลที่สนับสนุน ดังรูป



รูปที่ 6-1 แสดงโปรโตคอลที่ลิปเน็ตสนับสนุน

6.1.2 ขั้นตอนการดำเนินการ

เพื่อสร้างและส่งแพ็กเก็ตออกไป จะต้องมีคำสั่งพื้นฐานดังนี้

6.1.2.1 สร้างคอนเท็กซ์ (Context) ของตัวแพ็คเก็ต

เป็นการเลือกเน็ตเวิร์กการ์ด (Network card) และระดับการควบคุมแพ็คเก็ตตามที่ต้องการว่าต้องการควบคุมตั้งแต่ลิงค์เลเยอร์ (Link layer) หรือไอพีเลเยอร์ (IP Layer) ขึ้นไปโดยฟังก์ชันที่ใช้ คือ

```
libnet_t* libnet_init( int          // injection_type
                       char *       // device
                       char *       // err_buf   )
```

Injction_type เป็นตัวแปลที่บอกว่าต้องการควบแพ็คเก็ตในระดับไหน

- LIBNET_RAW4, LIBNET_RAW6 เป็นการสร้างแพ็คเก็ต ที่จะควบคุมตั้งแต่ไอพีเลเยอร์ขึ้นไป

- LIBNET_LINK, LIBNET_LINK6 เป็นการสร้างแพ็คเก็ต ที่จะควบคุมตั้งแต่ลิงค์เลเยอร์ขึ้นไป

ดีไวซ์(Device) เป็นการบอกว่าจะใช้เน็ตเวิร์กการ์ดอันไหนในการส่งแพ็คเก็ต โดยปกติค่าจะเป็น eth0 หรือไอพีแอดเดรส (IP address) ของเครื่องนั้น

Err_buf เป็นการแจ้งความผิดปกติ (Error) ถ้าเกิดการผิดพลาดระหว่างการสร้างจะต้องมีการเขียนข้อความแจ้งความผิดปกติ (Error Message) จะเขียนไว้ที่นี่

และตัวฟังก์ชัน Libnet_init ถ้าเกิดข้อผิดพลาดก็จะเขียนข้อความแจ้งความผิดปกติไว้ที่ err_buf และคืนค่านูล (NULL) ถ้าไม่มีการผิดพลาดจะคืนค่าพอย์เตอร์ (pointer) ที่ชี้ไปยังคอนเท็กซ์

6.1.2.2 สร้างแต่ละโพรโตคอลบล็อก (Protocol block)

ในการสร้างแต่ละแพ็คเก็ตต้องมีการสร้างโปรโตคอลของแต่ละชั้นในการสร้างแพ็คเก็ตเรียกว่าโพรโตคอลบล็อกเช่น ถ้าต้องการสร้างแพ็คเก็ตดีเอ็นเอสคิวรี่ (DNS query) ก็ต้องสร้างโพรโตคอลบล็อกของชั้นแอฟพลิเคชั่นคือ ดีเอนเอสและต้องการส่งแบบยูดีพี (UDP) ต้องสร้างโพรโตคอลบล็อกของยูดีพีเป็นต้น

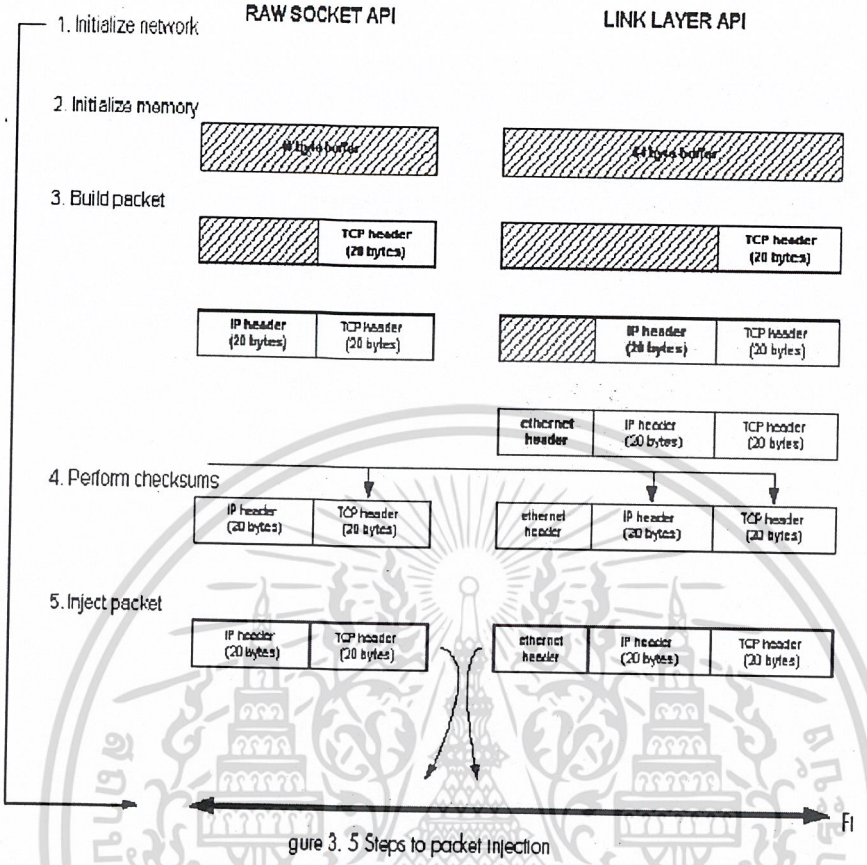
6.1.2.3 เป็นขั้นตอนในการส่งแพ็คเก็ต

ถ้าในการส่งต้องมีการทำเช็คซัม(Checksum) ซึ่งค่าเช็คซัมจะถูกคำนวณโดยอัตโนมัติเมื่อสร้างโพรโตคอลบล็อกสร็จ ในการส่งแพ็คเก็ตใช้ฟังก์ชัน

```
int libnet_write( libnet_t* )
```

6.1.2.4 เป็นขั้นตอนในการลบข้อมูลทั้งหมดที่สร้างไว้ออกจากหน่วยความจำ โดยการเรียกฟังก์ชัน

```
void libnet_destroy( libnet_t* )
```



รูปที่ 6-2 แสดงการขั้นตอนการสร้างและส่ง Packet ลงบนเครือข่าย

6.2 การทำซ้ำทราฟฟิกบนเครือข่าย

6.2.1 ความหมายของการทำซ้ำแพ็กเก็ตลงบนเครือข่าย (Replay packet)

หมายถึง การนำไฟล์ที่ดักจับแพ็คเก็ตมาได้จะเป็นไฟล์นามสกุล *.pcap จากระบบเน็ตเวิร์ก โดยวิธีการหรือโปรแกรมต่างๆ และนำมาแก้ไข หรือส่งแพ็คเก็ตออกไปตามไฟล์ที่ดักจับมาได้ ก่อนที่จะส่งออกไปลงบนเครือข่าย

6.2.2 องค์ประกอบของการทำซ้ำทราฟฟิกบนเครือข่าย

ในส่วนของการทำซ้ำทราฟฟิกบนเครือข่ายต้องมีไลบรารี่ที่เกี่ยวข้องคือลิปเน็ตและลิปพีแค็ปเพื่อทำให้การจัดการทำงานของการทำซ้ำทราฟฟิกบนเครือข่าย

6.2.3 การทำงานของการทำซ้ำทราฟฟิกบนเครือข่าย

การทำงานของการทำซ้ำทราฟฟิกบนเครือข่าย อาศัยไลบรารี่ลิปพีแค็ปในการอ่านข้อมูลจากจากไฟล์ที่ดักจับมาได้ และใช้ไลบรารี่ลิปเน็ตในการสร้างแพ็คเก็ตออกไปบนเครือข่าย

ในส่วนของการใช้ไลบรารี่ลิปพีแค็ปฟังก์ชันหลักที่ใช้ คือ

1) ฟังก์ชันในการเปิดอ่านไฟล์ที่ดักจับมาได้

pcapnav_t* pcapnav_open_offline (const char *filename);

ฟังก์ชันนี้เป็นการเปิดไฟล์ด้วยชื่อของไฟล์ และจัดการเกี่ยวกับไฟล์ ส่งค่ากลับมาเป็นการจัดการเกี่ยวกับข้อมูล(Handle for trace) ถ้าสำเร็จ หรือนูลถ้าเกิดข้อผิดพลาด เมื่อคุณทำงานเกี่ยวกับไฟล์ที่เปิดออกมานี้เสร็จแล้วก็ควรปิดการใช้งานด้วยฟังก์ชัน pcapnav_close()

2) ฟังก์ชันในการอ่านแพ็คเก็ตในไฟล์ทีละแพ็คเก็ต

int pcap_next_ex(pcap_t *p, struct pcap_pkthdr **pkt_header,const u_char **pkt_data)

ฟังก์ชันนี้เป็นการอ่านแพ็คเก็ตต่อไป ค่าที่จะส่งกลับมาเป็น u_char pointer ของข้อมูลในแพ็คเก็ตนั้น ถ้าค่าที่ส่งกลับมาเป็น null แสดงว่าเกิดข้อผิดพลาดในการอ่านข้อมูลแพ็คเก็ตนั้นๆ

ในส่วนของการใช้ไลบรารี่ลิปเน็ตฟังก์ชันหลักที่ใช้ คือ

1) ฟังก์ชันในการสร้าง

libnet_init(LIBNET_LINK_ADV, intf, ebuf)

ฟังก์ชันในการจองหน่วยความจำและเลือกชนิดในการสร้างแพ็คเก็ตว่าจะต้องการในระดับ เน็ตเวิร์กหรือระดับดาต้าสิงค์ที่เน็ตเวิร์กอันไหน ถ้าเกิดข้อผิดพลาดก็จะส่งข้อความเกี่ยวกับข้อผิดพลาดมาที่ ebuf

2) ฟังก์ชันในการส่งแพ็คเก็ตออกไปบนเครือข่าย

libnet_adv_write_link(l, pktdata, pkthdr.caplen)

ฟังก์ชันนการสร้างแพ็คเก็ตลงไปบนเครือข่ายโดยพารามิเตอร์(Parameter) ที่ส่งไปเป็นข้อมูลของแพ็คเก็ตและความยาวของแพ็คเก็ตที่จะสร้างออกไป ถ้าสามารถสร้างแพ็คเก็ตออกไปได้ฟังก์ชันนี้จะส่งค่ากลับมาเป็น 1 แต่ถ้าเกิดข้อผิดพลาดฟังก์ชันนี้จะส่งค่ากลับมาเป็น -1

6.2.4 ประโยชน์ของการทำซ้ำทราฟฟิกบนเครือข่าย

6.2.4.1 ในการตรวจสอบเครือข่าย (Network Debugging Tools) ในการตรวจสอบแอพพิเคชั่น หรือไฟล์วอล (firewall) และการทำการตรวจสอบเหมือนเดิมหลายๆครั้งเพื่อทำการเปรียบเทียบผลที่ได้ออกมา เพื่อวัดประสิทธภาพของเครือข่ายหรือวิเคราะห์ปัญหาที่เกิดขึ้น สามารถทำให้แก้ก่อนการเปิดให้บริการได้

6.2.4.2 เพื่อใช้ในการศึกษาการทำงานของโปรโตคอลในระดับชั้นต่างๆ ในระบบเครือข่าย ศึกษาลักษณะการโจมตีแบบต่างๆ ในระบบเครือข่าย

6.3 หลักการดักจับแพ็กเก็ต

6.3.1 ความหมายของแพ็คเก็ตแคปเจอร์ (Packet sniffer)

คำว่า สนิฟเฟอร์ (Sniffer) นั้นเครื่องหมายการค้าซึ่งจดทะเบียนไว้โดยบริษัท Network Associates Inc. ในสหรัฐฯเพื่อใช้ในผลิตภัณฑ์ของตนเองชื่อ Sniffer Network Analyzer ซึ่งเป็นโปรแกรมวิเคราะห์เน็ตเวิร์กโดยอาศัยการดักจับข้อมูลภายในเน็ตเวิร์กทั้งหมดมาวิเคราะห์การใช้งานระบบเน็ตเวิร์ก ต่อมาคำนี้ถูกนำมาใช้เรียกอุปกรณ์ประเภทนี้ที่นำมาในการดักอ่านข้อมูลกันอย่างแพร่หลาย จนเป็นที่เข้าใจว่าสนิฟเฟอร์เป็นเครื่องมือในการดักจับข้อมูลบนเน็ตเวิร์ก ซึ่งถ้าหากเรียกให้ถูกต้องอุปกรณ์ประเภทนี้ควรเรียกว่า Net wire Tapping Device

แพ็กเก็ตแคปเจอร์ คืออุปกรณ์ที่ทำหน้าที่ดักจับแพ็กเก็ตที่วิ่งอยู่บนเน็ตเวิร์ก โดยจะทำงานร่วมกับเน็ตเวิร์กอินเทอร์เฟซการ์ด (Network Interface card) ในโหมดโพรมิสคิวอัส (Promiscuous-mode)

โดยการทำงานร่วมกับโหมดนี้ การทำงานของเน็ตเวิร์กอินเทอร์เฟซการ์ดในโหมดนี้จะไม่มีการเปรียบเทียบแอดเดรสปลายทางของแพ็กเก็ตกับแอดเดรสของเน็ตเวิร์กอินเทอร์เฟซการ์ด นั่นคือทุกๆแพ็กเก็ตมาถึงยังเน็ตเวิร์กอินเทอร์เฟซการ์ดจะถูกจับแพ็กเก็ตขึ้นมาได้ทั้งหมดไม่เลือกว่าจะระบุแอดเดรสปลายทางของการ์ดเน็ตเวิร์กอินเทอร์เฟซอันไหน

แพ็กเก็ตแคปเจอร์จะถูกใช้ในการเฝ้าดูการจารจรในเครือข่ายให้มีประสิทธิภาพและแก้ไขปัญหาต่างๆที่เกิดจากการทำงานผิดพลาดของเครือข่ายและยังเป็นเครื่องมือในการศึกษาการทำงานของโพรโตคอลรวมถึงการดูแลความปลอดภัยของระบบ

6.3.2 องค์ประกอบของแพ็กเก็ตแคปเจอร์

6.3.2.1 ฮาร์ดแวร์ (Hardware) หมายถึงอุปกรณ์อิเล็กทรอนิคส์ที่สามารถดักอ่านสัญญาณจากเน็ตเวิร์กเข้ามาได้ และสามารถนำสัญญาณที่ให้ส่งต่อไปประมวลผลออกมาเป็นข้อมูลทางคอมพิวเตอร์ได้ หน้าที่หลักคือจัดการกับการรับข้อมูลในระดับฟิสิกคอล (Physical) อุปกรณ์นี้โดยทั่วไปคือเน็ตเวิร์กอะแดปเตอร์

6.3.2.2 ไดรเวอร์(Driver) เป็นโปรแกรมระดับล่างที่ควบคุมการดักจับข้อมูลของฮาร์ดแวร์ไปเก็บเป็นข้อมูลดิบรอการประมาวลผลในลำดับถัดไป

6.3.2.3 บัฟเฟอร์(Buffer) เป็นหน่วยความจำที่ใช้พักข้อมูลจากการดักจับมาได้ของ driver โดยจะทำงานการจัดเก็บเพียงชั่วคราว และหมุนเวียนข้อมูลใหม่เข้ามาแสมอ กลไกลการนำข้อมูลขากไดรเวอร์มาเก็บในบัฟเฟอร์ จะเป็นตัวบอกสมรรถนะของการดักข้อมูลได้ความเร็วสูงสุดเท่าใด หากกระบวนการนำข้อมูลไปเก็บเป็นไปอย่างล่าช้า ย่อมทำให้แพ็กเก็ตแคปเจอร์ไม่สามารถดักจับข้อมูลที่อยู่บนเน็ตเวิร์กได้ทันและต้องปล่อยข้อมูลนั้นทิ้งไป

6.3.2.4 ซอฟต์แวร์ (Software) เพื่อทำหน้าที่จัดการข้อมูลที่ได้เข้าโดยการประมวลผลตามวัตถุประสงค์ของการดักอ่านข้อมูลนั้น เนื่องจากข้อมูลดิบที่ดักได้นั้นจะเป็นข้อมูลในระดับต่ำคือ เน็ตเวิร์กอินเทอร์เฟซเลเยอร์ (Network Interface Layer) ซึ่งจะมีข้อมูลที่ยังไม่ผ่านการมัลติเพล็กซ์และจัดรูปแบบให้เข้าใจได้ ส่วนที่ได้จะเป็นข้อมูลเลขฐานสอง 0 กับ 1 จำนวนมหาศาลที่ต้องนำมาหาความหมายและอีกหน้าที่หนึ่งคือข้อมูลที่ดักได้นันเป็นการสื่อสารของทุกโฮสต์ที่อยู่ภายในเน็ตเวิร์กนั้นร่วมกันอยู่ อย่างไม่เป็นระเบียบและไม่มีการแยกแยะว่าเป็นการสื่อสารเรื่องอะไร ระหว่างโฮสต์ใดกับโฮสต์ใด การที่จะแปลความหมายของข้อมูลนี้ได้ก็จำเป็นต้องมีโปรแกรมสำหรับทำหน้าที่แปลความหมายของข้อมูลให้อยู่ในรูปแบบที่สามารถเข้าใจได้มากขึ้นนั่นคือการทำหน้าที่คล้ายกับดีมัลติเพล็กซ์ของโพรโตคอลปกติ แต่จะเป็นการดีมัลติเพล็กซ์ของข้อมูลทุกๆโฮสต์โดยไม่สนใจว่าเป็นข้อมูลของโฮสต์ใด

หลังจากข้อมูลผ่านการดีมัลติเพล็กซ์แล้วก็จะอยู่ในรูปที่สามารถเข้าใจได้ง่ายขึ้น แต่จะเข้าใจมากน้อยขึ้นอยู่กับความสามารถในการดีมัลติเพล็กซ์ของโปรแกรม หากสามารถดีมัลติพล็กซ์ได้จนถึงโพรโตคอลเลเยอร์ที่สูงรวมทั้งหากมีการแยกแยะหมวดหมู่ของการสื่อสารของแต่ละโฮสต์ก็จะเห็นความต่อเนื่องของการสื่อสารได้ดีขึ้น

ในส่วนของการจัดเก็บข้อมูลดิบที่ดักมาได้โปรแกรมแคปเจอร์ส่วนใหญ่จะทำการเก็บข้อมูลดิบที่ดักมาได้ไว้ในฐานข้อมูลเพื่อให้สามารถนำกลับมาวิเคราห์ในภายหลังได้ เพราะในการแคปเจอร์จะต้องคอยดักจับข้อมูลอยู่ตลอดเวลา หากทำการดีมัลติเพล็กซ์ทันที่ที่ได้รับข้อมูลในแบบเรียลไทม์อาจทำให้ประ

สิทธิภาพของแคปเจอร์ลดต่ำลง และเนื่องจากการสื่อสารข้อมูลต้องอาศัยความต่อเนื่องของการเชื่อมโยงกันของหลายๆแพ็กเก็ต ประกอบการดีมัลติเพล็กซ์ส่วนใหญ่จะสามารถทำได้ก็ต่อเมื่อข้อมูลได้รับมาครบถ้วนสมบูรณ์ทั้งหมดแล้วเท่านั้น

6.3.3 การทำงานของแพ็กเก็ตแคปเจอร์

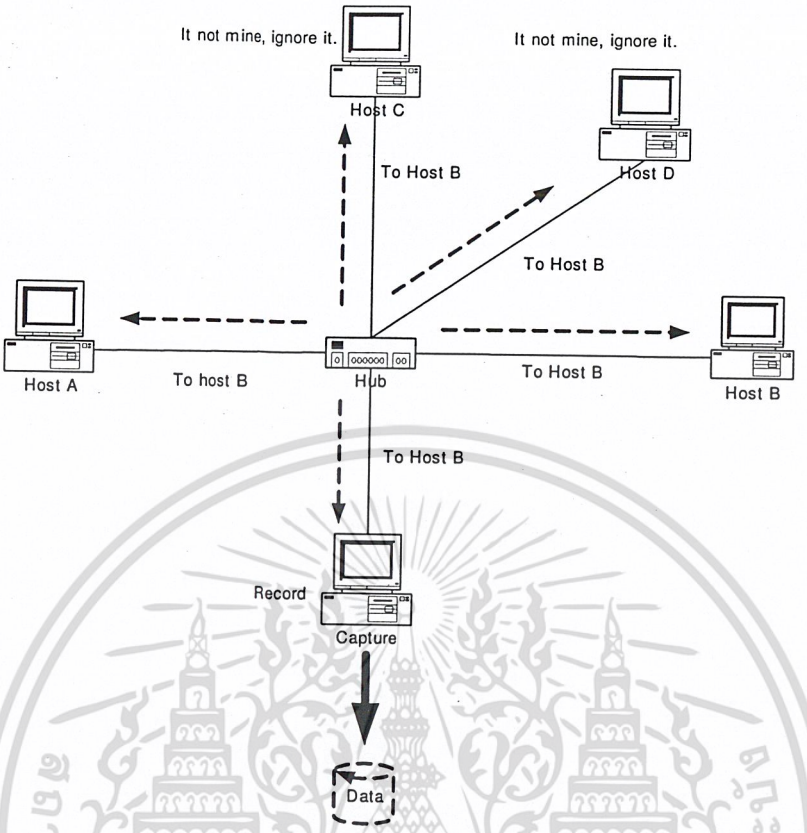
การทำงานของแพ็กเก็ตแคปเจอร์อาศัยหลักการของโพรโตคอลอีเทอร์เน็ตที่ใช้การกระจายของข้อมูลไปยังทุกโฮสต์ที่อยู่ในเน็ตเวิร์กและอาศัยโฮสต์แต่ละตัวทำหน้าที่จำแนกการสื่อสารของตนเอง นั่นหมายความว่าข้อมูลทุกแพ็กเก็ตที่ใช้สื่อสารกันได้ถูกส่งไปยังทุกโฮสต์ ซึ่งจะได้รับพร้อมกันแล้วเหมือนกัน เพียงแต่การที่จะสื่อสารกันได้อย่างถูกต้องนั้นโฮสต์แต่ละตัวจะต้องมีการบวนการที่สามารถรู้ได้ว่าข้อมูลแพ็กเก็ตใดเป็นของตนเอง และข้อมูลแพ็กเก็ตใดมิใช่ของตนเอง ทุกๆแพ็กเก็ตที่กระจายลงบนเน็ตเวิร์กนั้นจะมีหมายเลขระบุชัดเจตคือ แม็คแอดเดรส (MAC Address) หรือเรียกอีกอย่างหนึ่งว่าอีเทอร์เน็ต (Ethernet Address) ซึ่งจะเป็นสิ่งที่บอกว่าแพ็กเก็ตมาจากฮาร์ดแวร์ใดในเน็ตเวิร์ก ทำให้สามารถระบุได้ว่าส่งมาจากโฮสต์ใด

แม็คแอดเดรสจะเป็นหมายเลขเฉพาะฮาร์ดแวร์ทุกชนิดที่ใช้สื่อสารโดยโพรโตคอลอีเธอร์เน็ต และในทางทฤษฎีแล้วฮาร์ดแวร์ทุกชนิดจะไม่มีแม็คแอดเดรสซ้ำกัน โดยทั่วไปแม็คแอดเดรสจะกำหนดตายตัวอยู่ในรอม(Rom) ของฮาร์ดแวร์และไม่สามารถเปลี่ยนแปลงได้ด้วยซอฟต์แวร์

การใช้งานของฮาร์ดแวร์จะต้องควบคู่กับไดรเวอร์ของฮาร์ดแวร์นั้นๆโดยปกติแล้วไดรเวอร์จะถูกกำหนดให้ปฏิบัติตามโพรโตคอลอย่างเข้มงวดคือ

- ให้รับข้อมูลที่มีแม็คแอดเดรสเป็นของตนเองเท่านั้น (ห้ามอ่านข้อมูลของผู้อื่น)
- ให้ส่งข้อมูลโดยใช้แม็คแอดเดรสของตอนเองเท่านั้น (ห้ามปลอมเป็นผู้อื่น)

ดังนั้นหากใช้ไดรเวอร์ตามปกติที่มากับฮาร์ดแวร์แล้วเครื่องคอมพิวเตอร์ก็จะทำงานอยู่ในระเบียบเรียบร้อย และไม่สามารถวุ่นวายกับข้อมูลของผู้อื่นได้ แพ็กเก็ตแคปเจอร์จึงจำเป็นที่ต้องใช้โหมดการทำงานพิเศษที่อนุญาตให้รับข้อมูลที่แม็คแอดเดรสไม่ใช้ของตนได้โหมดนั้นเรียกว่า โพรมิสคิวอัสโหมด (Promiscuous Mode) ที่ทำให้ฮาร์ดแวร์สามารถรับข้อมูลดิบทั้งหมดบนเน็ตเวิร์กเข้ามาได้ โดยไม่สนใจว่าข้อมูลทั้งหมดเป็นของใคร ส่งให้ใคร



รูปที่ 6-3การทำงานของแพ็กเก็ตแคปเจอร์

จากรูปที่ 6-3 ในเน็ตเวิร์กมีโฮสต์อยู่ 4 ตัวต่อรวมกันเป็นเน็ตเวิร์กโดยใช้ฮับ (Hub) ร่วมกัน และมีโฮสต์ตัวหนึ่งรันโปรแกรมที่ทำหน้าที่เป็นโปรแกรมแคปเจอร์เพื่อดักจับข้อมูลเสียบต่อร่วมเข้ากับฮับนั้น จากรูปแสดงให้เห็นการกระจายของข้อมูลที่โฮสต์ A ต้องการส่งให้โฮสต์ B ข้อมูลได้กระจายไปยังทุกโฮสต์ที่ต่ออยู่บนฮับ รวมถึงแพ็กเก็ตแคปเจอร์ด้วย

โฮสต์ B เมื่อได้รับข้อมูลก็จะตรวจแม็คแอดเดรสถ้าพบว่าเป็นข้อมูลที่ถูกส่งมาหาตนเองก็ทำการอ่านและส่งต่อให้ระบบปฏิบัติการประมวลผล

โฮสต์ C และโฮสต์ D โฮสต์ทั้งสองซึ่งมีเน็ตเวิร์กอะแดปเตอร์ทำงานอยู่ในโหมดปกติ เมื่อได้ทำการตรวจสอบแล้วว่าไม่ใช่ข้อมูลของตนเอง ก็จะไม่สนใจข้อมูลนั้น

แพ็กเก็ตแคปเจอร์ สำหรับแพ็กเก็ตแคปเจอร์นั้นมีเน็ตเวิร์กแอแดปเตอร์ที่ทำงานอยู่ในโพรมิสคิวอัสโหมดนั่นคือไม่สนใจว่าข้อมูลที่เข้ามาเป็นของใคร แพ็กเก็ตแคปเจอร์จะรับข้อมูลทั้งหมดแล้วนำไปเก็บไว้ในบัฟเฟอร์และฐานข้อมูลเพื่อนำไปประมวลผลเพื่อหาข้อมูลที่มีประโยชน์ภายหลัง

จากเน็ตเวิร์กในภาพนั้นมีแพ็กเก็ตแคปเจอร์ถูกติดตั้งในตำแหน่งดังกล่าวก็จะทำให้ข้อมูลทั้งหมดที่โฮสต์ทั้ง 4 สื่อสารกันโดยผ่านฮับที่ใช้งานร่วมกัน ไม่ว่าข้อมูลใดจะปรากฏที่ฮับก็สามารถถูกดักอ่านได้โดยแพ็กเก็ตแคปเจอร์ทันที

6.3.4 ประโยชน์จากแพ็กเก็ตแคปเจอร์

6.3.4.1 วิเคราะห์ระบบเครือข่าย (Network Analyzer)

การที่แพ็กเก็ตแคปเจอร์สามารถดักจับข้อมูลทั้งหมดที่อยู่บนเน็ตเวิร์กได้ ทำให้เราสามารถนำข้อมูลดังกล่าวมาทำการประมวลผลวิเคราะห์คุณสมบัติต่างๆของเน็ตเวิร์กได้หลายแง่มุม ไม่ว่าการกระจายการใช้งานเน็ตเวิร์กของโพรโตคอลต่างๆเช่น เอชทีทีพี(HTTP), เอสเอ็มทีพี(SMTP), เอฟทีพี(FTP) เพื่อจะได้ทราบว่ามีการใช้เน็ตเวิร์กเพื่อการใด มากน้อยเพียงใด จะทำให้สามารถจัดการเน็ตเวิร์กได้อย่างเหมาะสมหรือในด้านการใช้งานเน็ตเวิร์กตามช่วงเวลาดังกล่าว หรือในด้านการใช้งานเน็ตเวิร์กตามช่วงเวลาต่างๆเวลาใดที่มีผู้ใช้มากเวลาใดที่มีผู้ใช้น้อย ผู้ใช้คนใดใช้แบนด์วิดธ์ไปในทางที่เป็นประโยชน์หรือไม่

6.3.4.2 เครื่องมือตรวจสอบเครือข่าย (Network Debugging Tools)

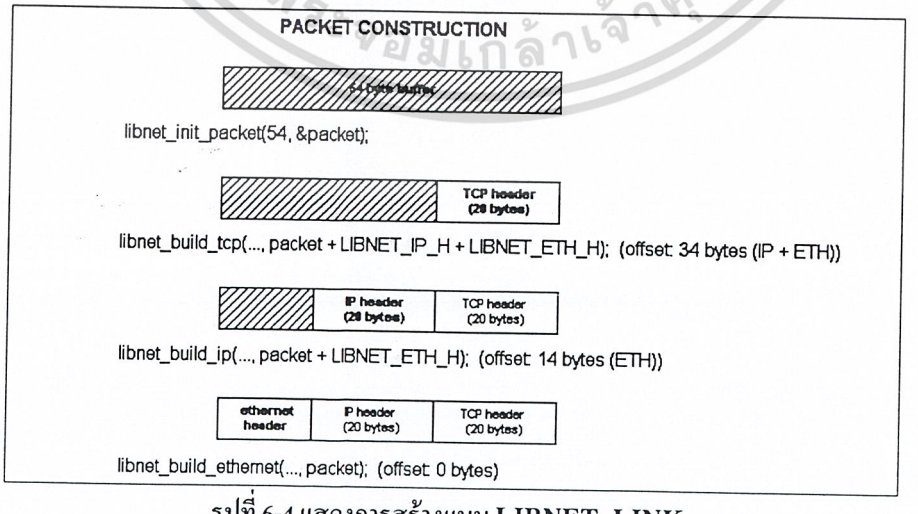
ในบางครั้งการหาสาเหตุความผิดปกติของแอพพลิเคชั่นต่างๆ จำเป็นที่ต้องสืบค้นลึกลงไปถึงเนื้อข้อมูลที่รับส่งกันจริงๆบนเน็ตเวิร์กเพื่อหาว่าเหตุใดการทำงานของแอพพลิเคชั่นจึงไม่สามาถทำงานได้ตามปกติ การได้เห็นข้อมูลที่ส่งกันจริงทำให้การวิเคราะห์และแก้ไขปัญหาสามาถดำเนินการอย่างรวดเร็วและตรงต่อหัญหาได้มากขึ้น โดยเฉพาะกรณีที่มีการใช้เครื่องมือในระดับเน็ตเวิร์กมาเกี่ยวข้องเช่น มีการส่งข้อมูลผ่านไฟร์วอลล์แล้วปัญหา หรือการทดสอบ ALC (Access Control List) ของเราเตอร์เป็นต้น

6.3.4.3 ตรวจจับแพ็กเก็ต (Packet Monitoring)

การศึกษาเทคนิคของโพรโตคอลในระดับเน็ตเวิร์กจำเป็นต้องเห็นข้อมูลที่สื่อสารกันอยู่จึงจะสามารถเข้าใจได้และเห็นภาพจริง แพ็กเก็ตมอนิเตอร์ริ่งจะเป็นเครื่องมือที่นำแพ็กเก็ตมาแสดงเห็นในรูปแบบต่างๆได้

6.3.4.4 ระบบตรวจจับผู้บุกรุก (Intrusion Detection System)

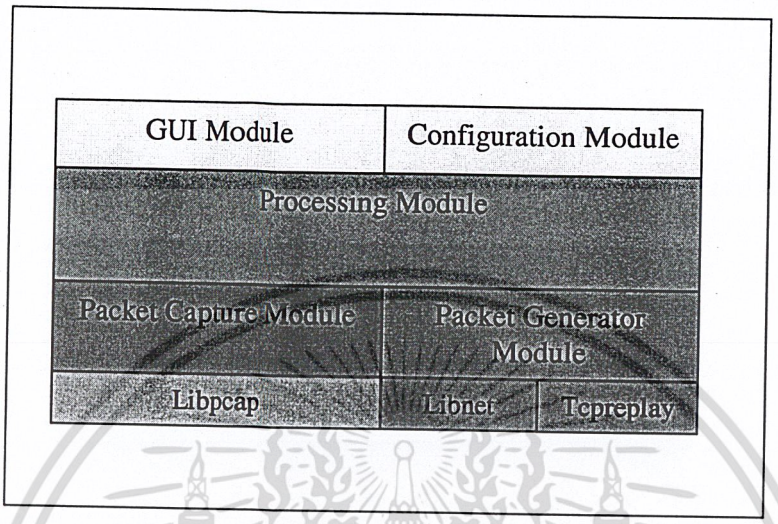
ระบบตรวจจับผู้บบุกรุกก็มีพื้นฐานมาจาการดักอ่านข้อมูลเพียงแต่เป็นการประยุกต์การวิเคราะห์รูปแบบการบุกรุกเข้ากับการดักอ่านข้อมูลบนเน็ตเวิร์กแบบเรียลไทม์ ทำให้สามารถทราบว่ามีกิจกรรมใดบนเน็ตเยิร์กที่มีแนวโน้มว่าจะเป็นการบุกรุกหรือทำอันตรายต่อผู้อื่น



รูปที่ 6-4 แสดงการสร้างแบบ LIBNET_LINK

6.4 การออกแบบโปรแกรม

โครงสร้าง Software (Design)



GUI Module : เป็นส่วนของโปรแกรมที่รับผิดชอบเกี่ยวกับ graphic user interface เพื่อสร้างความสะดวกในการติดต่อโดยตรงกับ ผู้ใช้

Configuration Module : เป็นส่วนของโปรแกรมที่รับผิดชอบการอ่านconfiguration file แล้วทำการสร้างแพ็กเก็ตตามลำดับใน configuration fileซึ่งผู้ใช้งานสามารถสร้าง configuration file ได้เอง

Packet Capture Module: เป็นส่วนของโปรแกรมที่รับผิดชอบการดักจับ packet ในเครือข่าย โดยใช้ไลบรารีชื่อ libpcap ช่วยในการดักจับข้อมูล

Packet Generator Module: เป็นส่วนของโปรแกรมที่รับผิดชอบการสร้าง packet อย่างที่ต้องการลงไปในระบบตามรูปแบบที่กำหนดจากConfiguration file สามารถสั่ง generate packet ได้ซ้ำๆ ตามต้องการ โดยใช้ไลบรารีชื่อ libnet และ Tcpreplay ช่วยในการสร้าง Packet Processing

Processing Module : เป็นส่วนของโปรแกรมที่ทำหน้าที่ประมวลผลลำดับของการทำงานที่สร้างจาก Configuration Module หรือจาก GUI แล้วสร้าง กระบวนการต่างๆ ตามลำดับนั้นโดยการสร้าง Packet จะส่งให้Packet Generator Module ทำงาน และในส่วนของการเก็บข้อมูลในเครือข่ายจะส่งให้ Packet Capture Module ทำงาน

GUI Module : เป็นส่วนของโปรแกรมที่รับผิดชอบเกี่ยวกับ graphic user interface เพื่อสร้างความสะดวกในการติดต่อโดยตรงกับ ผู้ใช้

Configuration Module : เป็นส่วนของโปรแกรมที่รับผิดชอบการอ่านconfiguration file แล้วทำการสร้างแพ็กเก็ตตามลำดับใน configuration fileซึ่งผู้ใช้งานสามารถสร้าง configuration file ได้เอง

Packet Capture Module:	เป็นส่วนของโปรแกรมที่รับผิดชอบการดักจับ packet ในเครือข่าย โดยใช้ไลบรารีชื่อ libpcap ช่วยในการดักจับข้อมูล
Packet Generator Module:	เป็นส่วนของโปรแกรมที่รับผิดชอบการสร้าง packet อย่างที่ต้องการลงไปในระบบตามรูปแบบที่กำหนดจากConfiguration file สามารถสั่ง generate packet ได้ซ้ำๆ ตามต้องการ โดยใช้ไลบรารีชื่อ libnet และ Tcpreplay ช่วยในการสร้าง Packet Processing
Processing Module :	เป็นส่วนของโปรแกรมที่ทำหน้าที่ประมวลผลลำดับของการทำงานที่สร้างจาก Configuration Module หรือจาก GUI แล้วสร้าง กระบวนการต่างๆ ตามลำดับนั้น โดยการสร้าง Packet จะส่งให้Packet Generator Module ทำงาน และในส่วนของการเก็บข้อมูลในเครือข่ายจะส่งให้ Packet Capture Module ทำงาน

บทที่ 7

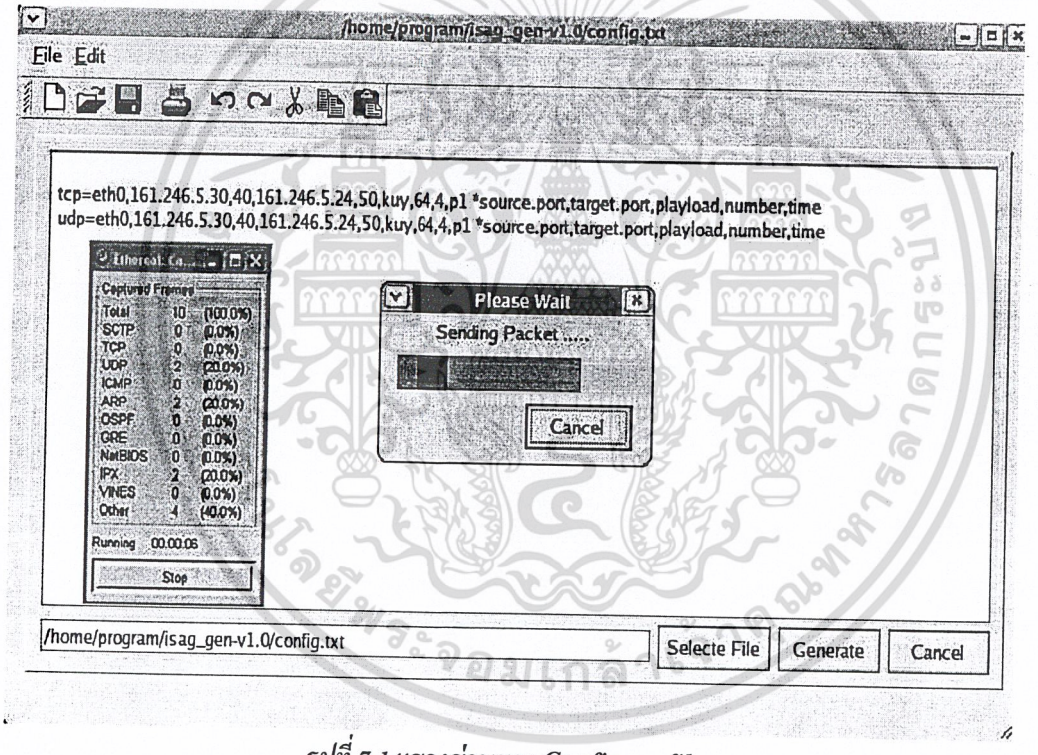
การทำงานของโปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่าย

<u>การทำงานของโปรแกรม</u>

โปรแกรมแบ่งออกเป็น 3 ส่วน คือ ส่วนของ Generate packet, ส่วนของ Replay packet และส่วนของ Capture packet

7.1 **Generate Packet** ได้โดยใช้ library libnet ในการช่วยสร้าง packet เช่น ICMP, TCP, UDP, etc. แบ่งการทำงานออกเป็น 3 Mode ดังนี้

7.1.1 Mode Configuration file สามารถ ลงกำหนดรูปแบบ packet ไฟล์แล้วโปรแกรมจะนำ file ไปทำการ generate packet หรือ replay packet ให้ตามที่เซ็ตไว้



รูปที่ 7-1 แสดงส่วนของ Configure file

จากตัวอย่าง Configure file ในบรรทัดแรกเป็นการบอกว่า ต้องการสร้างแพ็คเก็ต TCP โดยต้องกำหนดตัวพารามิเตอร์ดังนี้

Tcp=eth0,161.246.5.30,40,161.246.5.24,50,kuy,64,4,p1

Eth0 – อินเทอร์เฟซที่ต้องการส่ง

161.246.5.30.1 - เป็น source IP

40 - เป็น source Port

161.246.5.24.1 - เป็น destination IP

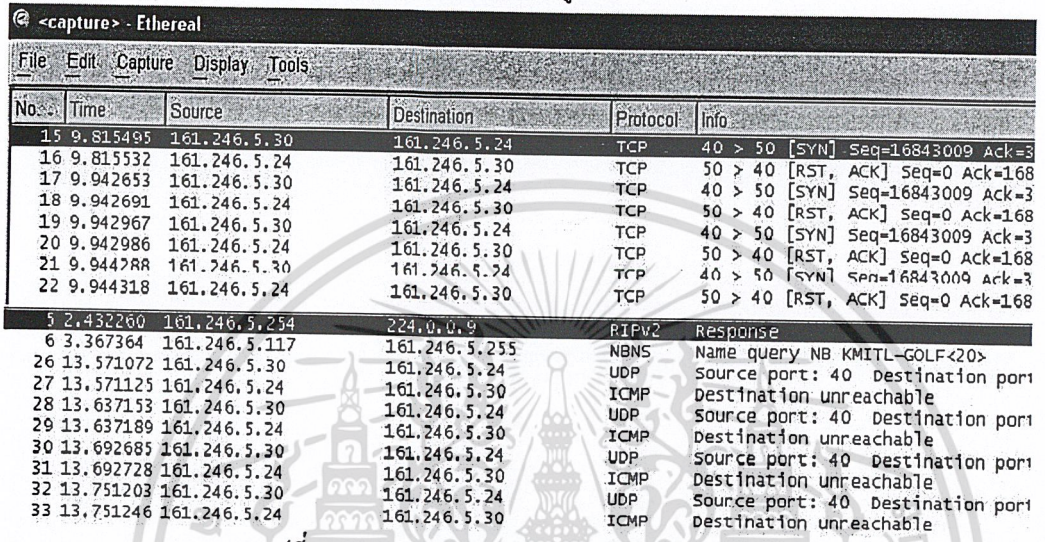
50 - เป็น destination Port

64 - เป็นจำนวน payload

4 - เป็นจำนวนที่ต้องการจะส่ง

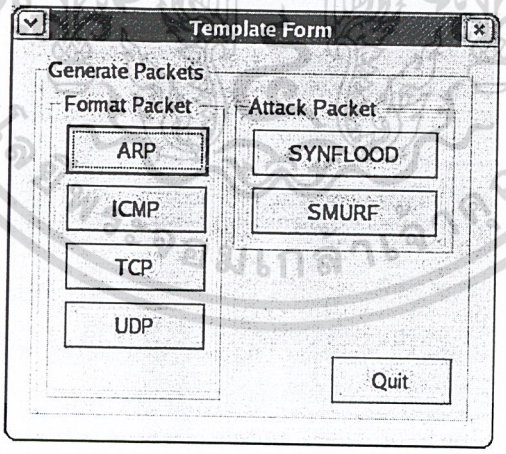
P1 - เป็นเวลาที่ต้องการส่ง

ส่วนอีกบรรทัดเป็นจำลองการส่งแพ็กเก็ตแบบ UDP ซึ่งรายละเอียดในการส่งจะคล้ายกับใน TCP และเมื่อลองใช้โปรแกรมอีเทอร์เรียลจับผลจะแสดงผลที่ได้ดังรูปที่ 7.2



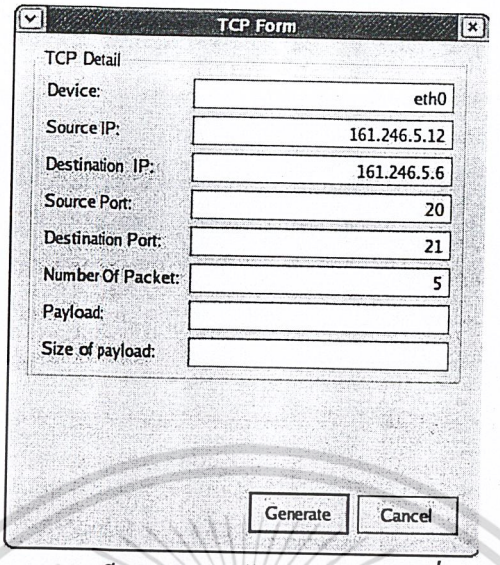
รูปที่ 7-2 แสดงผลของการสร้างจาก Configure file

7.1.2 Mode Template Packet มีรูปแบบ packet ทั้งรูปแบบปกติและรูปแบบโจมตีให้เลือก เมื่อเลือกรูปแบบที่ต้องการได้แล้ว โปรแกรมจะสร้าง Packet ให้ตามที่เซ็ตไว้



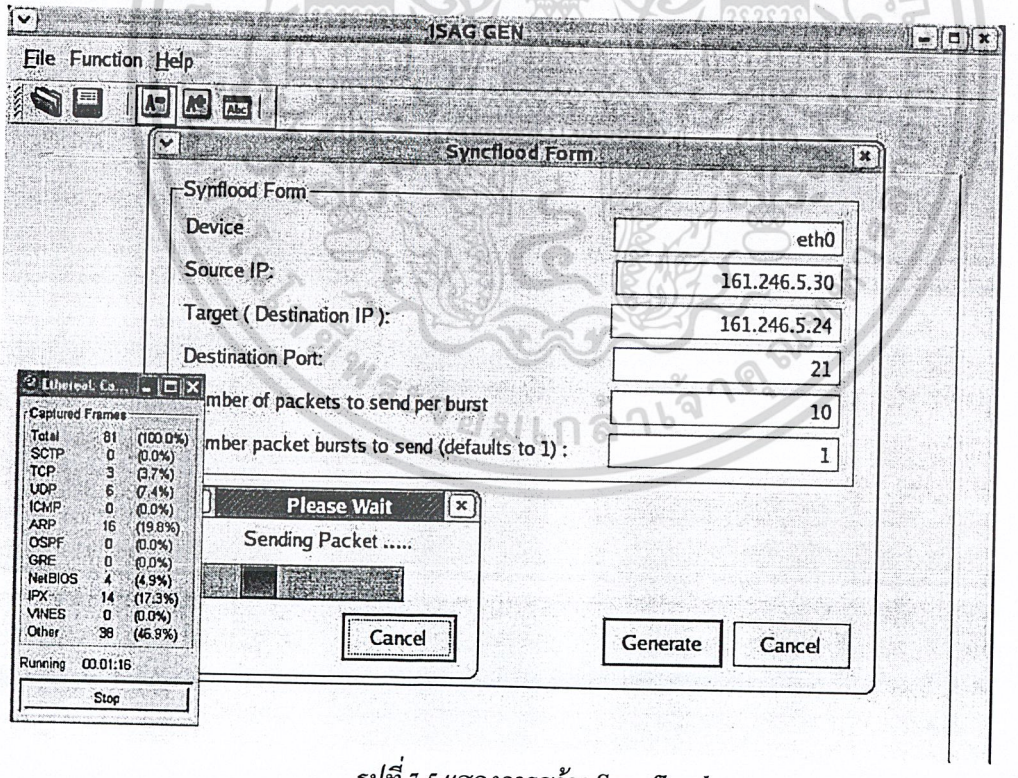
รูปที่ 7-3 ตัวอย่างการฟอร์แมตของแพ็คเก็ตที่ต้องการสร้าง

7.1.3 Mode Editable Packet คือสามารถแก้ไขรูปแบบ packet ตามที่ต้องการได้แต่ต้อง อยู่ภายใต้ขอบเขตที่กำหนดไว้



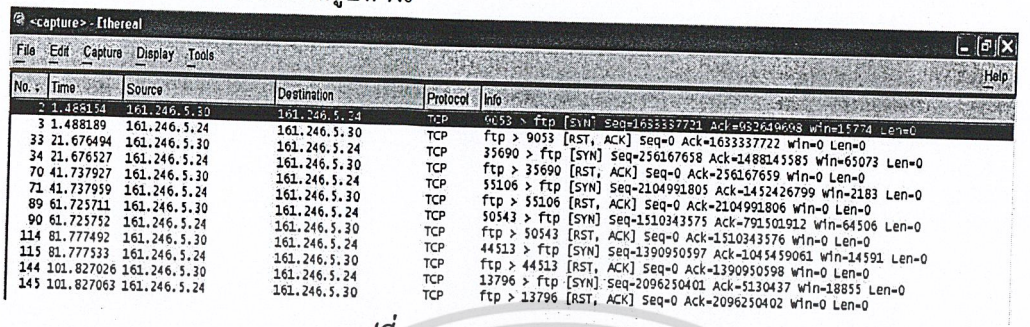
รูปที่ 7-4 แสดงรายระเอียดของการสร้าง Packet TCP ที่สามารถแก้ไขได้

ในรูปที่ 7.5 จะเป็นตัวอย่างการสร้าง packet โจมตีแบบ syncflood ซึ่งเป็นฟังก์ชั่นหนึ่งที่สามารเรียกใช้ได้ใน template mode และยังสามารถกำหนดรายละเอียดเองได้ดังรูปที่ 7.7 จะเป็นตัวอย่างการสร้างแพ็กเก็ตแบบปกติชนิด ICMP

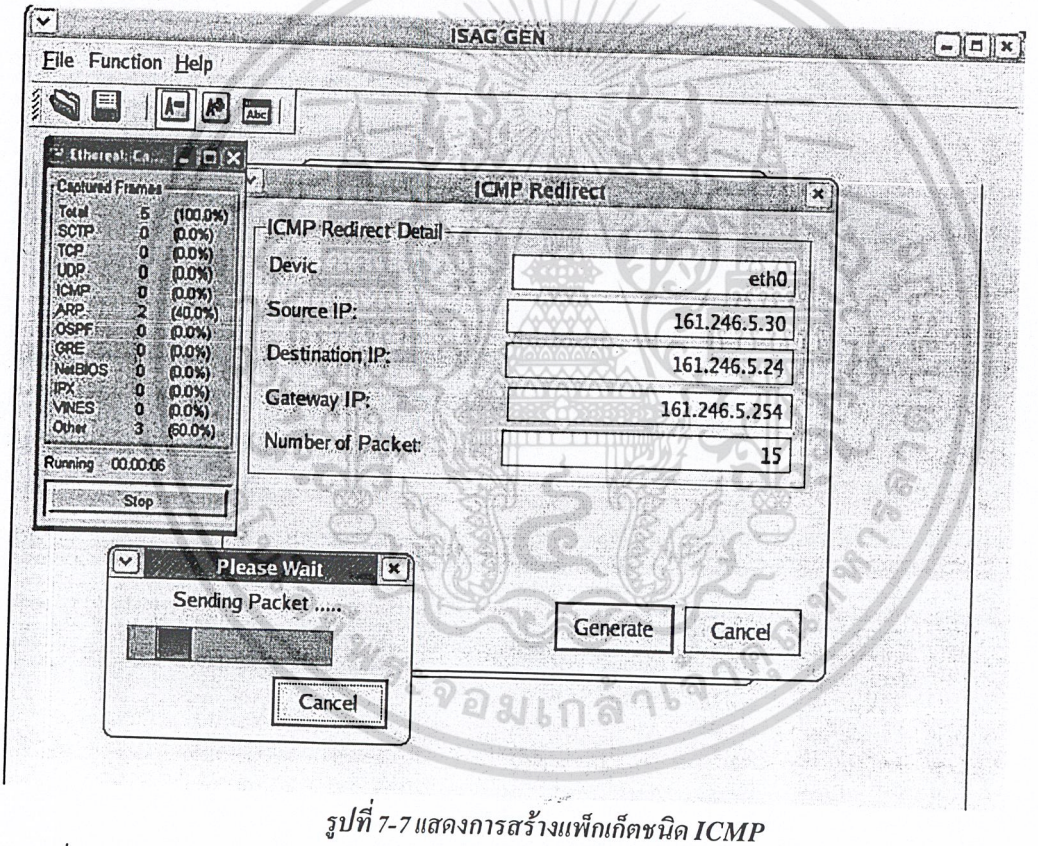


รูปที่ 7-5 แสดงการสร้าง Syncflood

ดังรูปที่ 7.5 จะแสดงผลของการสร้างแพ็กเก็ตโจมตีแบบ syncflood โดยระบุให้มี source IP เป็น 161.246.5.30 และ destination IP เป็น 161.246.5.24 ขณะที่ทำการส่งได้ลองดักจับแพ็กเก็ตที่ส่งด้วยซึ่งผลของการส่งจะได้ดังตัวอย่างในรูปที่ 7.6

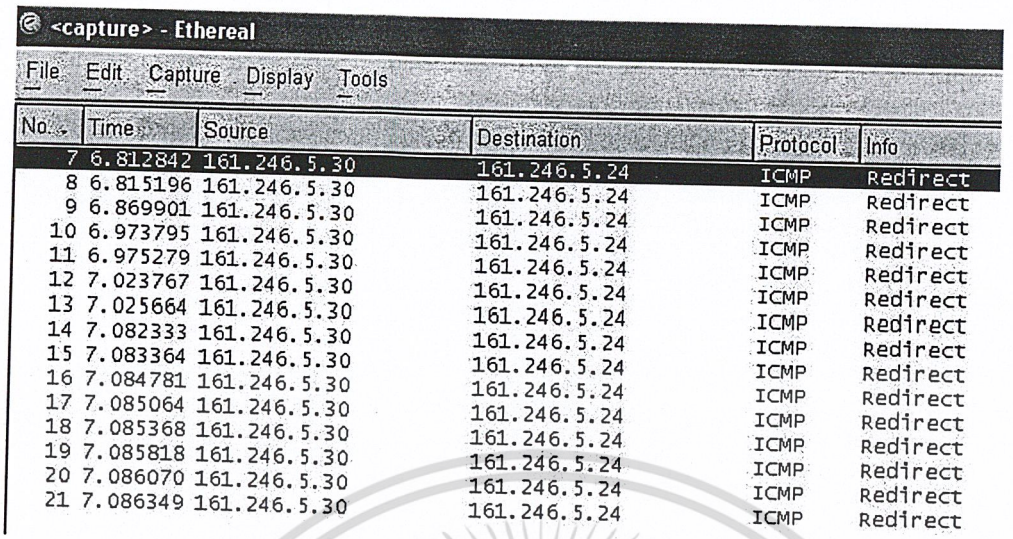


รูปที่ 7-6 แสดงผลของการสร้าง Syncflood



รูปที่ 7-7 แสดงการสร้างแพ็กเก็ตชนิด ICMP

จากรูปที่ 7.7 จะสามารถกำหนดรายละเอียดในการส่ง packet ได้โดยจากรูปกำหนดให้มีการส่งไปที่ interface eth0, source IP เป็น 161.246.5.30, และdestination IP เป็น 161.246.5.24, gateway IP เป็นๆ 161.246.5.254 และสุดท้ายสามารถกำหนดจำนวนของ packet ที่ใช้ทำการส่งได้อีกด้วย ซึ่งผลของการทำงานจะอยู่ในรูปที่ 7.8



รูปที่ 7-8 แสดงผลของการสร้างแพ็กเก็ตชนิด ICMP

7.2 Replay Packet โดยอาศัย library tcpreplay ในการช่วย replay packet จากไฟล์ที่ดักจับมาได้ เพื่อสร้าง environment ปิด และสามารถกำหนด option ในการ replay ได้ตามที่ต้องการ ซึ่งรายละเอียดในการทำฟังก์ชั่น รีเพลย์แพ็กเก็ตเป็นดังนี้

(1) เลือกกำหนด **Pattern** ได้เพื่อระบุว่าต้องการ replay ตามแบบใดมีฟังก์ชั่นให้เลือกดังนี้ Source, Destination, SourceIP&DestinationIP, SourceIP||DestinationIP, ListofPacketNumbers, Protocol

(2) การกำหนด **IP** เพื่อใช้กับ บาง pattern เช่นถ้าเลือก pattern เป็น "Source IP" หมายความว่าให้ replay packet ที่มี source IP เหมือนหรือไม่เหมือนกับ IP ที่กำหนด

(3) การกำหนด **List Packet Number** จะถูกกำหนดในกรณีที่เลือก pattern เป็นแบบ "ListofPacketNumbers" ซึ่งสามารถเลือกลำดับของ packet ได้จากตารางด้านล่างสุดหรือกำหนดลำดับเอง

(4) การเลือกชนิดของ **Protocols** จะต้องเลือกฟังก์ชั่นนี้ก็ต่อเมื่อเลือก pattern การ replay เป็นแบบ "Protocol"

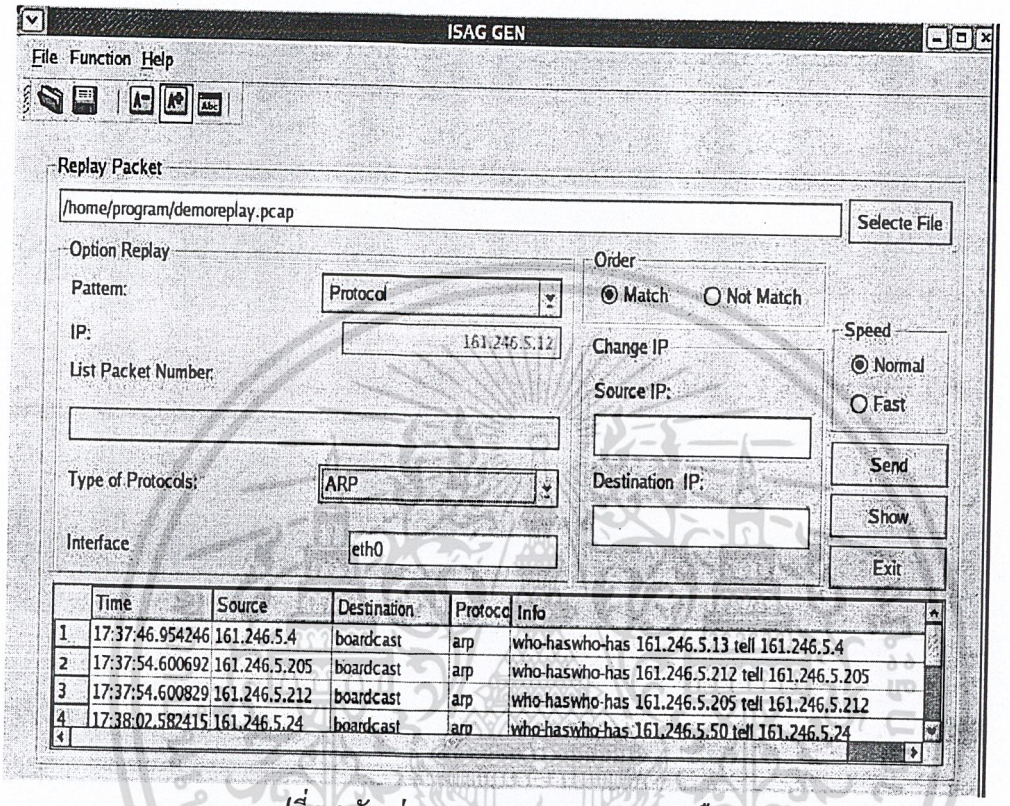
(5) การกำหนด ว่าต้องการให้ replay โดยให้ **Match** หรือ **Not Match** กับ pattern ที่เลือกไว้ก่อนหน้านี้

(6) การกำหนด **Source IP** กับ **Destination IP** ที่ต้องการจะแลกเปลี่ยนกัน ถ้ามีการกำหนดทั้ง Source IP และ Destination IP พร้อมกันจึงจะทำฟังก์ชั่น exchange IP เช่นกำหนด source IP เป็น 161.246.5.12 และกำหนด destination IP เป็น 161.246.5.6 โปรแกรมจะทำการ replay packet โดยเปลี่ยนไอพีต้นทางเป็นไอพีปลายทางตามที่กำหนด

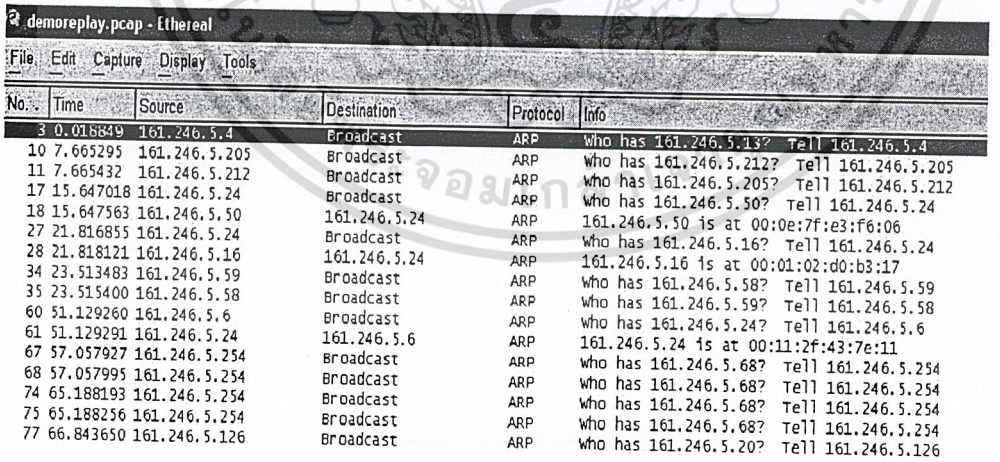
(7) การกำหนด **Speed** ของการ replay สามารถเลือกได้ 2 ระดับ คือ แบบ Normal กับแบบ Fast

(8) แสดงตารางรายการของ packet ที่ถูก replay ออกไปตามที่กำหนดไว้

ซึ่งการทำ replay packet สามารถเลือก option ที่จะทำการซ้ำได้หลายฟังก์ชั่นตัวอย่างการทำซ้ำแพ็กเก็ตในรูปที่ 7.9 จะแสดงการทำซ้ำแพ็กเก็ตโดยกำหนดให้ส่งเฉพาะโปรโตคอล arp จากไฟล์ที่ดักจับมาได้ และผลของการทำงานจะอยู่ในรูปที่ 7.10

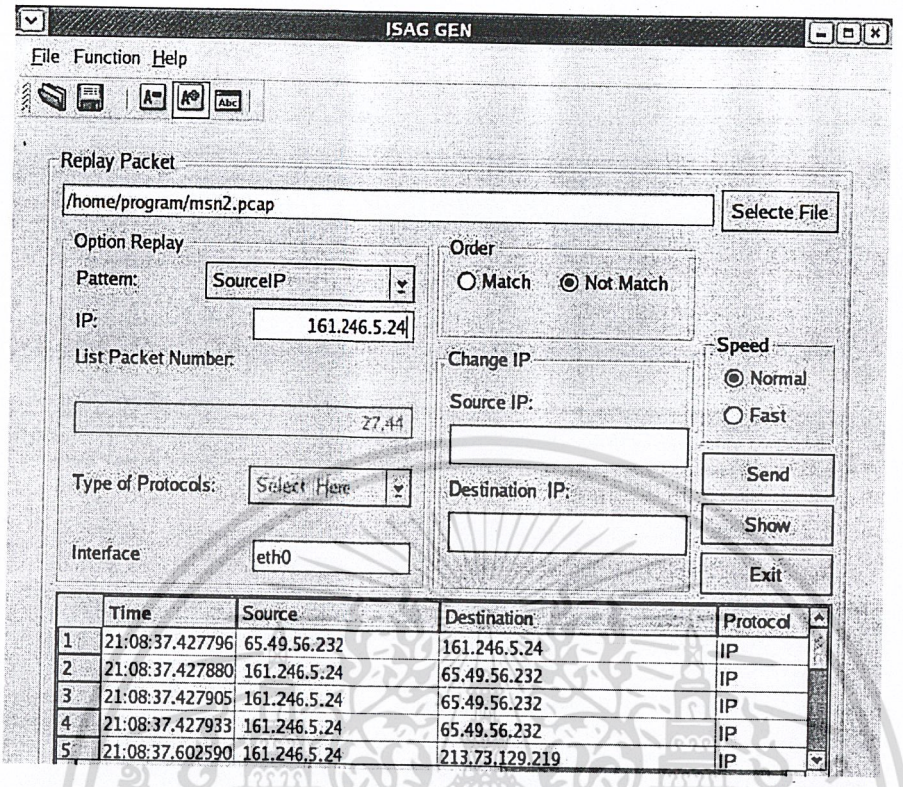


รูปที่ 7-9 ตัวอย่าง การ Replay packet แบบเลือก match

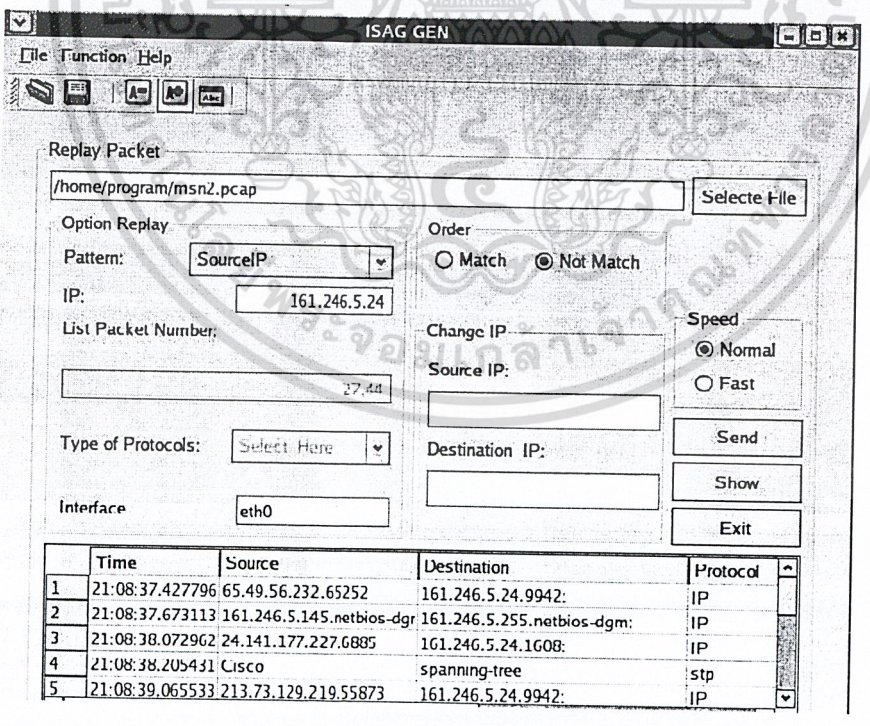


รูปที่ 7-10 ผลของการทำ Replay packet แบบเลือก match

รูปที่ 7.11 จะเป็นตัวอย่างการแสดงการทำซ้ำแพ็กเก็ตแบบเลือกไม่ match กับไอพีที่กำหนดและผลที่ได้จะแสดงในรูปที่ 7.12



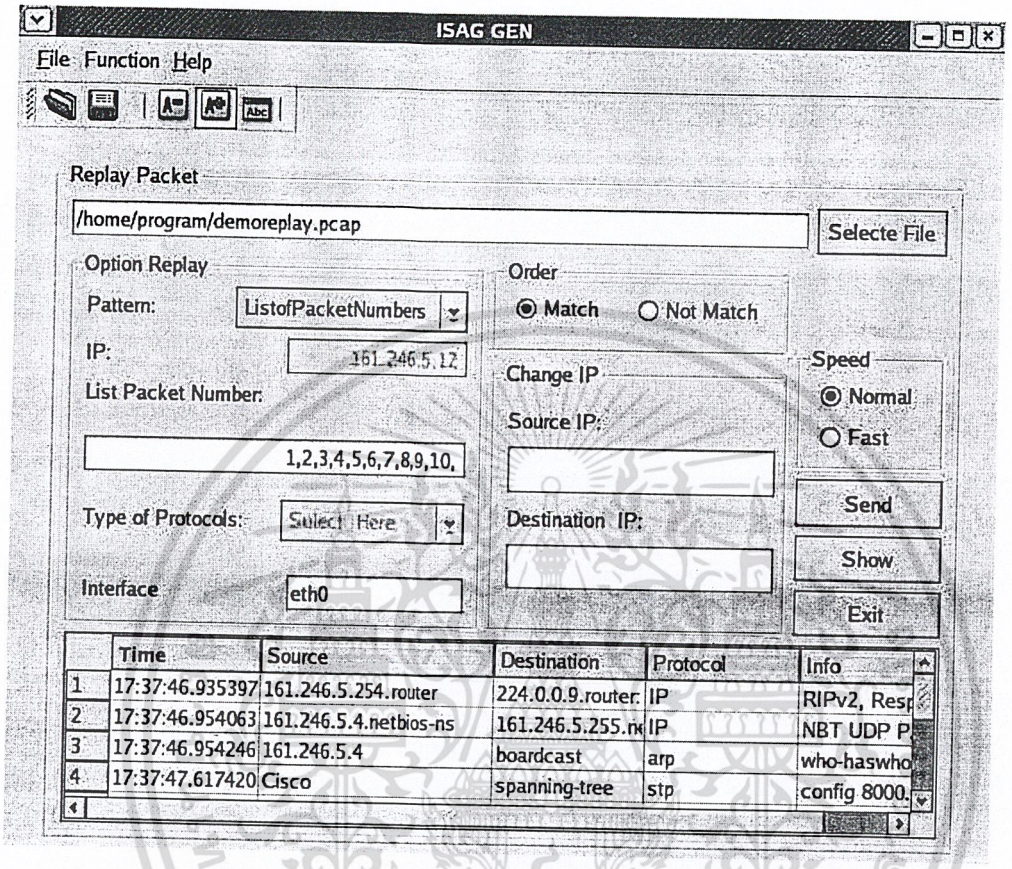
รูปที่ 7-11 ตัวอย่างฟังก์ชั่นของ การ Replay packet แบบเลือกไม่ match



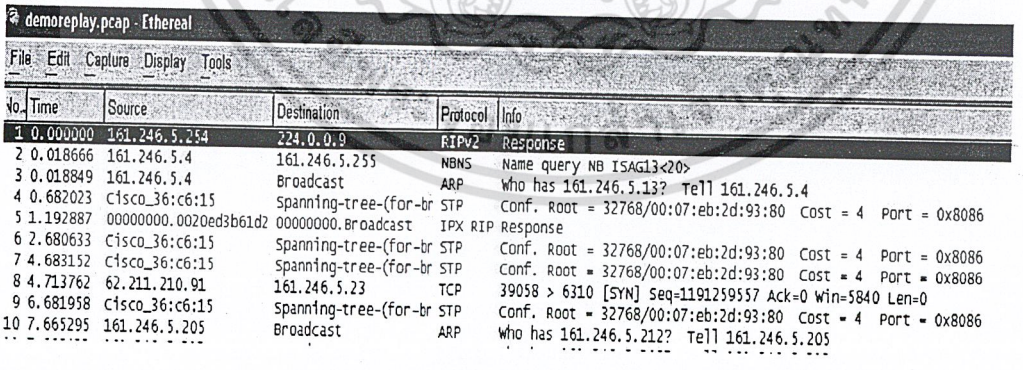
รูปที่ 7-12 แสดงผลของการทำ Replay packet แบบเลือกไม่ match

อีกตัวอย่างหนึ่งของการทำซ้ำแพ็กเก็ตคือสามารถเลือกจาก list ที่อยู่ในตารางแสดงผลด้านล่างของโปรแกรมโดยเลือกเฉพาะลำดับแพ็กเก็ตที่ต้องการส่งซ้ำได้ดังรูปที่ 7.13 ในที่นี้จะเลือก 10 ลำดับแรกในตา

รางเพื่อทำการส่งผลที่ได้สามารถส่งไปได้เฉพาะ 10 ลำดับแรกในตารางด้านล่างโปรแกรมซึ่งผลจากการทำจะแสดงในรูปที่ 7.14

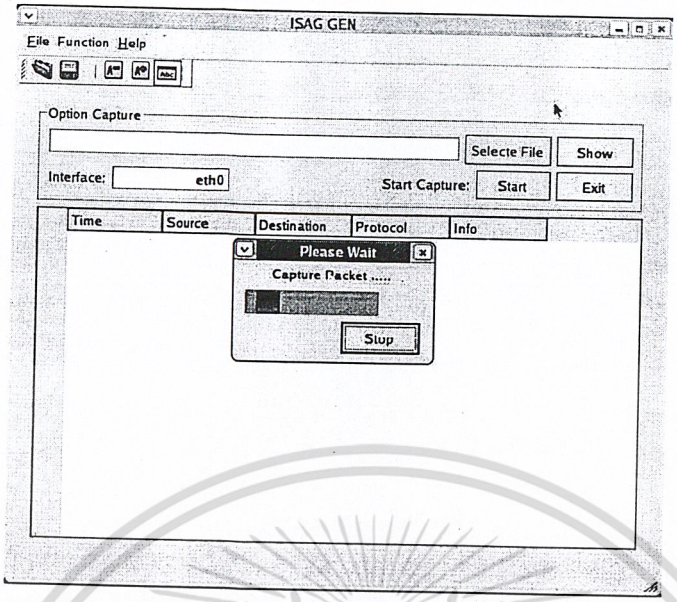


รูปที่ 7-13 แสดงการทำ Replay packet แบบเลือกใน list

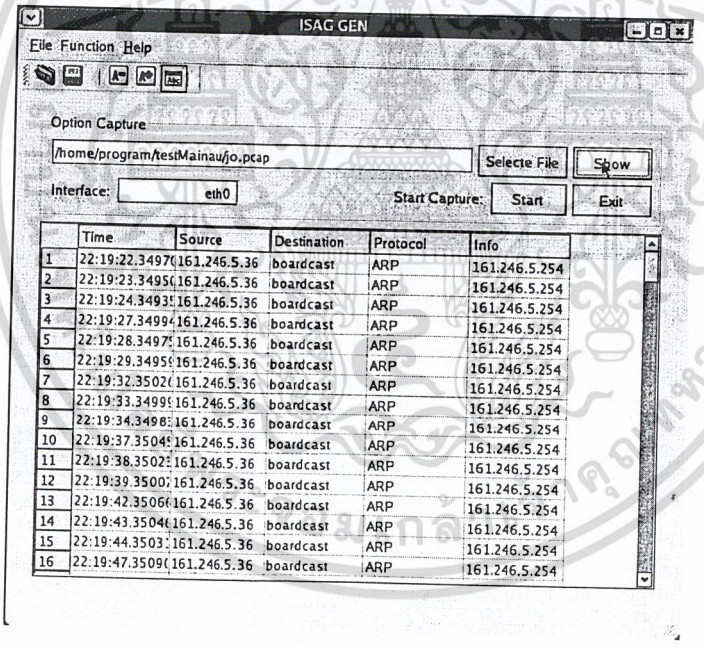


รูปที่ 7-14 แสดงผลของการทำ Replay packet แบบเลือกใน list

7.3 Capture Packet โดยอาศัย library libpcap ในการทำฟังก์ชั่น capture ซึ่งสามารถเลือกกำหนด interface ที่ต้องการจะ capture ได้และกำหนดชื่อไฟล์ที่ต้องการได้ เมื่อcapture packet มาแล้วต้องการจะแสดงรายการที่ต้องการก็สามารถเลือกที่จะให้แสดงสิ่งที่ดักจับมาได้



รูปที่ 7-15 แสดงการใช้ฟังก์ชั่นในการดักจับแพ็คเก็ต



	Time	Source	Destination	Protocol	Info
1	22:19:22.3497(	161.246.5.36	boardcast	ARP	161.246.5.254
2	22:19:23.3495(	161.246.5.36	boardcast	ARP	161.246.5.254
3	22:19:24.3493!	161.246.5.36	boardcast	ARP	161.246.5.254
4	22:19:27.3499(	161.246.5.36	boardcast	ARP	161.246.5.254
5	22:19:28.3497!	161.246.5.36	boardcast	ARP	161.246.5.254
6	22:19:29.3495(	161.246.5.36	boardcast	ARP	161.246.5.254
7	22:19:32.3502(	161.246.5.36	boardcast	ARP	161.246.5.254
8	22:19:33.3499(	161.246.5.36	boardcast	ARP	161.246.5.254
9	22:19:34.3498:	161.246.5.36	boardcast	ARP	161.246.5.254
10	22:19:37.3504!	161.246.5.36	boardcast	ARP	161.246.5.254
11	22:19:38.3502!	161.246.5.36	boardcast	ARP	161.246.5.254
12	22:19:39.3500;	161.246.5.36	boardcast	ARP	161.246.5.254
13	22:19:42.3506(	161.246.5.36	boardcast	ARP	161.246.5.254
14	22:19:43.3504(	161.246.5.36	boardcast	ARP	161.246.5.254
15	22:19:44.3503;	161.246.5.36	boardcast	ARP	161.246.5.254
16	22:19:47.3509(	161.246.5.36	boardcast	ARP	161.246.5.254

รูปที่ 7-16 แสดงผลของการดักจับแพ็คเก็ต

บทที่ 8
สรุปและวิจารณ์

8.1 สรุปผล

โปรแกรมสร้างและทำซ้ำทราฟฟิกบนเครือข่ายที่ถูกพัฒนาขึ้นนี้มีความสามารถในการทำงานดังที่ตั้งใจไว้ดังต่อไปนี้

1. การสร้างแพ็กเก็ตสามารถเลือกรูปแบบการสร้างแพ็กเก็ตได้หลากหลายรูปแบบดังต่อไปนี้ TCP, UDP, ICMP, ARP และที่เป็นในรูปแบบโจมตีดังนี้แบบ Syncflood และ แบบ Smurf ซึ่งแต่ละรูปแบบก็สามารถดัดแปลงแก้ไขข้อมูลของแต่ละแพ็กเก็ตได้ตามต้องการเพื่อเป็นการเพิ่มความหลากหลายของทราฟฟิกที่ใช้ในการทดสอบ

2. การทำซ้ำทราฟฟิกสามารถทำซ้ำได้จากไฟล์ที่ดักจับมา แล้วยังสามารถนำเอาไฟล์มาแก้ไขดัดแปลงเพิ่มเติมก่อนที่จะทำซ้ำลงบนเครือข่าย ซึ่งผลจากการทดสอบสามารถทำได้หลายครั้งแล้วนำผลที่ได้จากการทดลองมาเปรียบเทียบและหาสาเหตุความผิดปกติหรือเพื่อปรับปรุงประสิทธิภาพของระบบโดยรวมให้มีศักยภาพในการรองรับการทำงานได้มากยิ่งขึ้น

3. ส่วนของการดักจับแพ็กเก็ตสามารถทำการดักจับแพ็กเก็ตแล้วนำมาบันทึกลงไฟล์ที่ต้องการแล้วนำเอาไฟล์ที่ได้ไปทดสอบ ทำซ้ำใหม่เพื่อวิเคราะห์และหาสาเหตุความผิดปกติเพื่อปรับปรุงระบบให้ดีขึ้น

4. ในการทำงานทั่วไปๆ ยูสเซอร์สามารถเรียกใช้การทำงานโปรแกรมได้จากการเขียนคำสั่งต่างๆ ลงไฟล์ ที่เรียกว่า configuration file ได้ด้วย

ซึ่งการทำงานโดยรวมของโปรแกรมถือว่าทำงานได้มีประสิทธิภาพพอสมควรซึ่งถ้าต้องการความหลากหลายของรูปแบบแพ็กเก็ตก็สามารถเพิ่มเติมส่วนนั้นๆเข้าไปได้

8.2 ปัญหาและอุปสรรค

8.1.1 ปัญหาเกี่ยวกับการสร้างแพ็คเก็ต (Packet) ให้ได้ตามเวลาที่กำหนด (Real time) เนื่องมาจากหลายสาเหตุเช่น ในขณะที่ทำการเจนเนอเรตแพ็กเก็ต (Generate Packet) ออกไปสภาวะของเครือข่ายขณะนั้นเกิดความคับคั่งของแพ็กเก็ตอยู่แล้วทำให้ไม่สามารถส่งแพ็กเก็ตออกไปบน เครือข่ายได้ตามเวลาต้องการทำให้เกิดความล่าช้าหรือเกิดข้อผิดพลาดขึ้นได้

8.1.2. ปัญหาในขั้นตอนการเขียนโปรแกรมเนื่องจากโปรแกรมที่ใช้ในการทำงานเขียนขึ้นโดยใช้ภาษาซีแต่โปรแกรมคิวที (QT Programming) ที่ใช้ในการเขียนกราฟฟิกยูสเซอร์อินเทอร์เฟซ (Graphic User Interface) ถูกเขียนขึ้นด้วยภาษาซีพลัสๆเมื่อนำมาลิงค์กันเกิดปัญหาการส่งค่าระหว่างภาษาซีกับภาษาซีพลัสๆ จึงแก้ปัญหาโดยการส่งคำสั่งไปแล้วสั่งให้เอาท์พุทที่ได้ให้ไปเขียนลงบนไฟล์ก่อนแล้วค่อยนำไฟล์ไปแสดงผลซึ่งอาจทำให้การทำงานเกิดความล่าช้ากว่าปกติ

8.3 แนวทางการวิจัยและพัฒนาต่อ

เนื่องจากโปรแกรมได้ถูกพัฒนาให้มีความสามารถในการสร้างแพ็กเก็ตได้หลายรูปแบบและมีการกำหนดฟังก์ชั่นในการทำซ้ำแพ็กเก็ตตามต้องการอีกทั้งยังสามารถดักจับแพ็กเก็ตเพื่อนำมาบันทึกลงไฟล์ได้ แล้ว แต่ยังมีส่วนที่น่าจะนำมาเพิ่มเติมเพื่อเพิ่มศักยภาพของโปรแกรมให้มีความสามารถหลากหลายขึ้นเป็นต้นว่า

8.2.1 ในการสร้างทราฟฟิกบนเครือข่ายสามารถเพิ่มรูปแบบของการเจนเนอเรตแพ็กเก็ตได้อีกรวมทั้งฟังก์ชั่นของรูปแบบการทำซ้ำแพ็กเก็ต

8.2.2 เพิ่มความสามารถในการปรับ sequence number และ port ของแพ็กเก็ตก่อนการทำรีเพลย์แพ็กเก็ตได้

ภาคผนวก ก

Function in Libnet which used in this project

Packet Construction

Packets are constructed modularly. For each protocol layer, there should be a corresponding call to a **libnet_build function**. Depending on your end goal, different things may happen here. For the above IP-layer example, calls to libnet_build_ip() and libnet_build_tcp() will be made. For the link-layer example, an additional call to libnet_build_ethernet() will be made. It is important to note that the ordering of the packet constructor function calls is not significant, it is only necessary that the correct memory locations be passed to these functions. The functions need to build the packet headers inside the buffer as they would appear on the wire and be demultiplexed by the recipient, and this can be done in any order

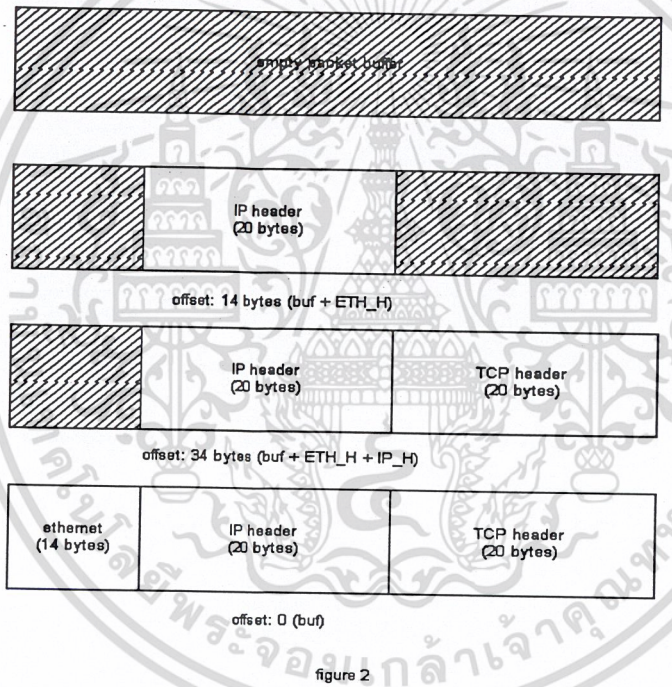


figure 2

libnet_build_ethernet() would be passed the beginning of the buffer with an offset of 0 (as it needs to build an ethernet header at the front of the packet). libnet_build_ip() would get the buffer at a 14

u_long libnet_name_resolve(u_char *ip, u_short use_name);

return value upon success	network-byte ordered IP address
return value upon failure	-1
re-entrant	yes
arguments	1 - human readable IP address or FQDN 2 - use_name flag

libnet_name_resolve() takes a NULL terminated ASCII string representation of an IP address (dots and decimals or, if the usename flag is LIBNET_RESOLVE, a canonical hostname) and converts it into a network-ordered (big-endian) unsigned long value.

int libnet_init_packet(u_short packet_size, u_char **buf);

return value upon success	1
return value upon failure	-1
re-entrant	yes
arguments	1 - desired packet size 2 - address of a u_char pointer

libnet_init_packet() creates memory for a packet (it doesn't so much create memory as it requests it from the underlying operating system via malloc()). Upon success the memory is zero-filled. The function accepts two arguments, the packet size and the address of the pointer to the packet. The packet size parameter may be 0, in which case the library will attempt to guess a packet size for you. Passing in the pointer to a pointer (passing by address) is necessary as we are allocating memory locally. If we instead passed in just a pointer (passing by value) the allocated memory would be lost.

This function is a good example of interface hiding. This function is essentially a malloc() wrapper. By using this function the details of what's really happening are abstracted so that you, the programmer, can worry about your task at hand.

int libnet_write_ip(int socket, u_char *packet, int packet_size);

return value upon success	packet size
return value upon failure	-1
re-entrant	yes
arguments	1 - socket 2 - pointer to the packet 3 - packet size
arguments	1 - maximum size of pseudo-random number desired

libnet_write_ip() writes an IP packet to the network. The first argument is the socket created with a previous call to libnet_open_raw_sock, the second is a pointer to a buffer containing a complete IP datagram, and the third argument is the total packet size. The function returns the number of bytes written upon success or -1 on error (with errno containing the reason).

void libnet_destroy_packet(u_char **buf);

return value upon success	NA
return value upon failure	NA
re-entrant	yes
arguments	1 - address of a u_char pointer

libnet_destroy_packet() is the free() analog to libnet_init_packet. It destroys the packet referenced by 'buf'. In reality, it is of course a simple free() wrapper. It frees the heap memory and points `buf` to NULL to dispel the dangling pointer. The function does make the assertion that `buf` is not NULL. A pointer to a pointer is passed to maintain interface consistency.

u_long libnet_get_prand(int modulus);

libnet_get_prand() generates a psuedo-random number. The range of the returned number is controlled by the function's only argument:

VALUE	DESCSRIPTION
LIBNET_PR2	0 - 1
LIBNET_PR8	0 - 255
LIBNET_PR16	0 - 32767
LIBNET_PRu16	0 - 65535
LIBNET_PR32	0 - 2147483647
LIBNET_PRu32	0 - 4294967295

The function does not fail.

int libnet_seed_prand();

return value upon success	1
return value upon failure	-1
re-entrant	yes
arguments	NA

libnet_seed_prand() seeds the pseudo-random number generator. The function is basically a wrapper to srandom. It makes a call to gettimeofday to get entropy. It can return -1 if the call to gettimeofday fails (check errno). It otherwise returns 1.

ภาคผนวก ข

Function in Libpcap which used in this project.

pcap_open_offline () is called to open a ``savefile" for reading. fname specifies the name of the file to open. The file has the same format as those used by tcpdump(1) and tcpslice(1). The name "-" in a synonym for stdin. errbuf is used to return error text and is only set when pcap_open_offline() fails and returns NULL.

pcap_loop () is similar to pcap_dispatch() except it keeps reading packets until cnt packets are processed or an error occurs. It does not return when live read timeouts occur. Rather, specifying a non-zero read timeout to pcap_open_live() and then calling pcap_dispatch() allows the reception and processing of any packets that arrive when the timeout occurs. A negative cnt causes pcap_loop() to loop forever (or at least until an error occurs). -1 is returned on an error; 0 is returned if cnt is exhausted; -2 is returned if the loop terminated due to a call to pcap_breakloop() before any packets were processed. If your application uses pcap_breakloop(), make sure that you explicitly check for -1 and -2, rather than just checking for a return value < 0.

pcap_dump() outputs a packet to the ``savefile" opened with pcap_dump_open(). Note that its calling arguments are suitable for use with pcap_dispatch() or pcap_loop(). If called directly, the user parameter is of type pcap_dumper_t as returned by pcap_dump_open().

pcap_dump_file() returns the standard I/O stream of the ``savefile" opened by pcap_dump_open().

pcap_open_live() is used to obtain a packet capture descriptor to look at packets on the network. device is a string that specifies the network device to open; on Linux systems with 2.2 or later kernels, a device argument of "any" or NULL can be used to capture packets from all interfaces. snaplen specifies the maximum number of bytes to capture. If this value is less than the size of a packet that is captured, only the first snaplen bytes of that packet will be captured and provided as packet data. A value of 65535 should be sufficient, on most if not all networks, to capture all the data available from the packet. promisc specifies if the interface is to be put into promiscuous mode. (Note that even if this parameter is false, the interface could well be in promiscuous mode for some other reason.) For now, this doesn't work on the "any" device; if an argument of "any" or NULL is supplied, the promisc flag is ignored. to_ms specifies the read timeout in milliseconds. The read timeout is used to arrange that the read not necessarily return immediately when a packet is seen, but that it wait for some amount of time to allow more packets to arrive and to read multiple packets from the OS kernel in one operation. Not all platforms support a read timeout; on platforms that don't, the read timeout is ignored. A zero value for to_ms, on platforms that support a read timeout, will cause a read to wait forever to allow enough packets to arrive, with no timeout. errbuf is used to return error or warning text. It will be set to error text when pcap_open_live() fails and returns NULL. errbuf may also be set to warning text when pcap_open_live() succeds; to detect this case the caller should store a zero-length string in errbuf before calling pcap_open_live() and display the warning to the user if errbuf is no longer a zero-length string.

pcap_freealldevs() is used to free a list allocated by pcap_findalldevs().

pcap_findalldevs() constructs a list of network devices that can be opened with pcap_open_live(). (Note that there may be network devices that cannot be opened with

pcap_open_live() by the process calling pcap_findalldevs(), because, for example, that process might not have sufficient privileges to open them for capturing; if so, those devices will not appear on the list.) alldevsp is set to point to the first element of the list; each element of the list is of type pcap_if_t, and has the following members:

pcap_dump() outputs a packet to the ``savefile'' opened with pcap_dump_open(). Note that its calling arguments are suitable for use with pcap_dispatch() or pcap_loop(). If called directly, the user parameter is of type pcap_dumper_t as returned by pcap_dump_open().

ภาคผนวก ค

Function in Libpcapnav which used in this project.

libpcapnav is a libpcap wrapper library that allows navigation to arbitrary locations in a tcpdump trace file between reads. The API is intentionally much like that of the pcap library. You can navigate in trace files both in time and space: you can jump to a packet which is at appr. 2/3 of the trace, or you can jump as closely as possible to a packet with a given timestamp, and then read packets from there. In addition, the API provides convenience functions for manipulating timeval structures.

Like libpcap, this library handles things through an opaque handle struct. For trace file navigation and reading packets, this handle is enough. If you need to apply BPF filters or write packets to disk, you can access the familiar pcap handle that is used internally

pcapnav_goto_offset ()

```
pcapnav_result_t pcapnav_goto_offset    (pcapnav_t *pn,
                                         off_t offset,
                                         pcapnav_cmp_t boundary);
```

The function tries to jump as closely as possible to a valid offset in the file near offset. The difference to pcapnav_set_offset() is that the latter simply modifies the file stream position, trusting that you know what you're doing. An offset of 0 is the first packet in the trace (i.e., the trace file header is not included). Using &boundary, you can limit the result to packets below, beyond, or anywhere around the requested offset.

pn : pcapnav handle.
offset : position to jump to.
boundary : where around the offset to jump to.
Returns : success state.

pcapnav_pcap ()

```
pcap_t*  pcapnav_pcap          (pcapnav_t *pn);
```

Use this function for interaction with libpcap. It returns the pcap handler for the given pcapnav handler. Do *not* mess with this unless you have to -- do not close the pcap handler etc behind pcapnav's back ...

pn : pcapnav handler.
Returns : pcap handler, or NULL on invalid input.

บรรณานุกรม

หนังสืออ้างอิง

[1] Mark A. Miller, P.E., "Troubleshooting TCP/IP Analyzing the Protocols of the Internet", M&T Books, 1993, pp. 121-215

[2] William Stallings, "Data and Computer Communications", 5th Edition, Prentice Hall, 1997, pp. 497-526, 585-619

[3] Neil Matthew, Richard Stones, "Beginning Linux Programming", Wrox Press, 1996

[4] Lowell Jay Arthur, Ted Burns, "UNIX Shell Programming", 4th Edition, John Wiley & Sons, 1997

[5] เรืองไกร รังสิพล, "เจาะระบบ TCP/IP จุดอ่อนของโปรโตคอล และวิธีป้องกัน", โปรวิชั่น, 2544

[6] สุวัฒน์ ปุณณชัยยะ, ต้น ตันฑ์สุทธิวงศ์, สุพจน์ ปุณณชัยชนะ, "เปิดโลกของ TCP/IP และโปรโตคอลของอินเตอร์เน็ต", โปรวิชั่น, 2543

[7] LBNL, Network Research Group, "libpcap", http://ee.lbl.gov

[8] Stevens, W. Richard, "UNIX Network Programming, vol. I, 2nd ed.", Prentice Hall PTR, 1998

[9] Hanson, David R., "C Interfaces and Implementations", Addison-Wesley, 1997

เว็บไซต์อ้างอิง

[1] http://www.securityfocus.com

[2] http://doc.trolltech.com/

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

[4] http://www.ethreal.com/

[5] http://tcpreplay.sourceforge.net/

[6] http://winpcap.mirror.ethereal.com/301a/docs/group__wpcapfunc.html

[7] http://www.packetfactory.net/libnet

[8] libnet projects: http://www.packetfactory.net/

[9] libnet homepage: http://www.packetfactory.net/libnet

[10] libnet mailing list: http://www.packetfactory.net/libnet/mailing-list.html