สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

โปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล

STATEFUL PACKET CAPTURE

T046169

นายเกรียงศักดิ์ เครื่องนันตา

นายอนันต์ สุวรรณทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

วพ. ก 7682 ส 544

เลขหมู่.....
เลขทะเบียน 46169
วัน, เดือน, ปี 20 มี.ค. 2546

b.....11287810.....
i.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล
STATEFUL PACKET CAPTURE

โดย
นายเกรียงศักดิ์ เครื่องนันตา
นายอนันต์ สุวรรณทา

อาจารย์ที่ปรึกษา
อาจารย์ ธนา หงษ์สุวรรณ
อาจารย์ อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล

STATEFUL PACKET CAPTURE

ผู้จัดทำ

1. นาย เกรียงศักดิ์ เครื่องนันตา รหัสประจำตัว 42015295
2. นาย อนันต์ สุวรรณทา รหัสประจำตัว 42015330

_____ อาจารย์ที่ปรึกษา

(อาจารย์ ธนา หงษ์สุวรรณ)

_____ อาจารย์ที่ปรึกษา

(อาจารย์ อัครเดช วัชระภูพงษ์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล

นายเกรียงศักดิ์ เครื่องนันตา
นายอนันต์ สุวรรณทา
อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา

ปีการศึกษา 2544

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เป็นอีกแนวทางหนึ่งในการทดลองสร้างโปรแกรมเพื่อเฝ้าดูพฤติกรรมของผู้ใช้บนเครือข่าย โดยตัวโปรแกรมจะทำการดักจับข้อมูลบนเครือข่ายที่ผู้ดูแลระบบต้องการทราบมาเก็บไว้เป็นไฟล์ โดยในแต่ละไฟล์จะแยกเป็นแต่ละเซสชันเพื่อความสะดวกในการนำไปวิเคราะห์พฤติกรรมของผู้ใช้ในภายหลัง

เนื่องจากโปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล มีความสัมพันธ์กับโปรแกรมไฟวอลท์ ดังนั้นในวิทยานิพนธ์ฉบับนี้จึงได้ประยุกต์เอาความสามารถทางด้านการตรวจสอบสเตตที่มีอยู่ในโปรแกรมไฟวอลท์ มาใช้กับโปรแกรมที่จะสร้างขึ้น ซึ่งทำให้มีความสามารถในการติดตามสเตตของแต่ละแพ็กเก็ตได้ดีขึ้น และสามารถป้องกันการโจมตีแบบ Syn Flooding และ DOS Attacks ได้ โดยเรียกโปรแกรมนี้ว่า "โปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล" Stateful Packet Capture

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Stateful Packet Capture

Kriangsak Kruengnanta

Anan Suwantha

Assoc. Prof. Thana Hongsuwan Advisor

ABSTRACT

This thesis is related to programme, planned to keep a close watch over the action of internet users.This programme would catch up data on the networks which supporters want to know and save in their files.Individual file would be devided to each session for analysis user manners easily later.

Thanks to " Stateful Packet Capture " is linked to "Firewall Programme" , this thesis applied the state inspection ability in "Firewall Programme" to employ with this programme .Hence the programme is able to detect state in each packege properly. The proposed architecture protects against SYN Flooding and firewall saturation DoS attacks. This programme is named "Stateful Packet Capture" .

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลสองท่านแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้ปริญญานิพนธ์นี้เสร็จลงได้ ก็คือ อาจารย์ ธนา หงษ์สุวรรณ และ อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

เกรียงศักดิ์ เครื่องนันตา
อนันต์ สุวรรณทา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้าที่

บทคัดย่อภาษาไทย I
บทคัดย่อภาษาอังกฤษ II
กิตติกรรมประกาศ III
สารบัญ IV
สารบัญภาพประกอบ VIII
สารบัญตาราง XI
บทที่ 1 บทนำ 1
1.1 ความสำคัญและที่มา 1
1.2 วัตถุประสงค์ของงานวิจัย 1
1.3 ขอบเขตของงานวิจัย 1
1.4 วิธีการดำเนินงาน 1
บทที่ 2 โพรโตคอลทีซีพี/ไอพี 3
2.1 ความเป็นมาของโพรโตคอลทีซีพี/ไอพี 3
2.2 โครงสร้างโพรโตคอล ทีซีพี/ไอพี 3
2.2.1 ชั้นแอพพลิเคชัน 4
2.2.2 ชั้นทรานสปอร์ต 4
2.2.3 ชั้นอินเทอร์เน็ต 4
2.2.4 ชั้นเน็ตเวิร์กอินเทอร์เฟซ 5
2.3 การส่งถ่ายข้อมูลระหว่างชั้น 5
2.4 โพรโตคอลยูดีพี 6
2.5 โพรโตคอลทีซีพี 7
2.5.1 กลไกการทำงานของทีซีพี 8
2.5.1.1 การเริ่มต้นการเชื่อมต่อ 8
2.5.1.2 การถ่ายโอนข้อมูล 10
2.5.1.3 การยกเลิกการเชื่อมต่อ 11
2.5.2 สไลดิงวินโดว์ 12
2.5.2.1 หน้าต่างฝ่ายส่ง 13
2.5.2.2 หน้าต่างฝ่ายรับ 14
2.6 โพรโตคอลไอพี 16

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้าที่

บทที่ 3 ไอพี 19

3.1 โพรโตคอลไอพี 19

3.2 การประมวลดาต้าแกรม 19

3.2.1 การประมวลดาต้าแกรมที่เราเตอร์ 19

3.2.2 การประมวณดาต้าแกรมที่โฮสต์ปลายทาง 20

3.3 ชนิดการให้บริการ 20

3.4 การแบ่งขนาดข้อมูล 21

3.4.1 การแฟรกเมนต์ 22

3.4.2 การรีแอสเซมบลี 22

3.4.3 ข้อคำนึงในการแฟรกเมนต์ 23

บทที่ 4 แผนภูมิสถานะทีซีพี 24

4.1 สถานะทีซีพี 24

4.1.1 การเริ่มต้นการเชื่อมต่อ 24

4.1.1.1 สถานะเซอร์ฟเวอร์ขั้นตอนการเริ่มต้นการเชื่อมต่อ 24

4.1.1.2 สถานะไคลเอ็นต์ขั้นตอนการเริ่มต้นการเชื่อมต่อ 25

4.1.2 การปิดการเชื่อมต่อ 26

4.1.2.1 สถานะฝ่ายที่ขอปิดการเชื่อมต่อ 26

4.1.2.2 สถานะฝ่ายได้รับคำร้องขอปิดการเชื่อมต่อ 26

4.1.3 การรับส่งเซกเมนต์ในขั้นตอนการเริ่มต้นการเชื่อมต่อ 26

4.1.4 การรับส่งเซกเมนต์ในขั้นตอนการปิดการเชื่อมต่อ 27

4.2 รูปแบบการเปิดและปิดการเชื่อมต่ออื่น 28

4.2.1 การเปิดพร้อมกัน 29

4.2.2 การปิดพร้อมกัน 29

4.2.3 การเปิดครึ่งฝ่าย 30

4.2.4 การปิดครึ่งฝ่าย 30

บทที่ 5 แพ็กเก็ตแคปเจอร์ 32

5.1 ความหมายแพ็กเก็ตแคปเจอร์ 32

5.2 องค์ประกอบของแพ็กเก็ตแคปเจอร์ 32

5.2.1 ฮาร์ดแวร์ 32

5.2.2 ไดรเวอร์ 32

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้าที่

5.2.3 บัฟเฟอร์ 32
5.2.4 ซอฟต์แวร์ 33
5.3 การทำงานของแพ็กเก็ตแคปเจอร์ 33
5.4 ประโยชน์จากแพ็กเก็ตแคปเจอร์ 35
5.4.1 วิเคราะห์ระบบเครือข่าย 35
5.4.2 เครื่องมือตรวจสอบเครือข่าย 35
5.4.3 ตรวจจับแพ็กเก็ต 35
5.4.4 ระบบตรวจจับผู้บุกรุก 36
บทที่ 6 สเตตฟูลไฟร์วอล 37
6.1 หลักการทำงาน 37
6.2 การแบ่งชนิดของไฟร์วอล 37
6.2.1 เพ็กเก็ตฟิลเตอร์ริง 37
6.2.2 แอพพลิเคชันพร็อกซี 39
6.2.3 สเตตฟูลอินสเป็กชัน 39
6.3 หลักการทำงานของ simple packet filter 40
6.4 หลักการทำงานของ packet-filtering firewall 42
6.5 FW-1 สเตตฟูลอินสเป็กชัน 43
6.5.1 หลักการทำงาน 43
6.5.2 สเตตฟูลอินสเป็กชัน 44
6.5.3 การทำงานของ Inspection Module 44
6.5.4 วิเคราะห์ Inspection Module 45
บทที่ 7 การโจมตีเครือข่าย 48
7.1 การโจมตีโดย TCP SYN Flooding 48
7.1.1 การทำงานของ TCP SYN Flooding 48
7.1.2 คิวของการรอรับการขอติดต่อ 49
7.1.3 ผลกระทบจาก TCP SYN Flooding 51
7.2. การตรวจสอบการเปิดพอร์ต 51
7.2.1 การทำงานของ Scan Port 51
7.3 การปลอมหมายเลขไอพี 52
บทที่ 8 การออกแบบโปรแกรมวิเคราะห์แพ็กเก็ตแบบสเตตฟูล 53
8.1 การออกแบบโปรแกรม 53

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

หน้าที่

8.2 โครงสร้างข้อมูลของโปรแกรม 57
8.2.1 ความหมายของตัวแปรแต่ละตัว 57
8.3 ฟังก์ชันต่างๆ ของโปรแกรม 58
8.3.1 ฟังก์ชันหลัก 58
8.3.2 ฟังก์ชันอื่นๆ 58
8.4 ข้อจำกัดของโปรแกรม 62
บทที่ 9 สรุปและวิจารณ์ 63
9.1 ปัญหาและอุปสรรค 63
9.2 คุณลักษณะของโปรแกรมที่ได้พัฒนา 63
9.3 แนวทางในการพัฒนางานวิจัยเพิ่มเติม 63
บรรณานุกรม 64

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ

หน้าที่

รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของทีซีพี/ไอพีกับเลเยอร์ของโอเอสไอ 3
รูปที่ 2-2 โพรโตคอลต่างที่เรียกใช้ทรานสปอร์ตเลเยอร์ 4
รูปที่ 2-3 โพรโตคอล TCP และ UDP อาศัยโพรโตคอล IP เพื่อส่งข้อมูลระหว่างเครือข่าย 5
รูปที่ 2-4 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี 5
รูปที่ 2-5 ยูดีพีดาต้าแกรม 6
รูปที่ 2-6 ทีซีพีเซกเมนต์ 7
รูปที่ 2-7 แสดงแพ็กเก็ตทีซีพี 7
รูปที่ 2-8 การเริ่มต้นการเชื่อมต่อด้วยทีซีพี 9
รูปที่ 2-9 ขั้นตอนการถ่ายโอนข้อมูล 10
รูปที่ 2-10 การถ่ายโอนเซกเมนต์ทั้งสองทิศทาง 11
รูปที่ 2-11 การปิดการเชื่อมต่อ 12
รูปที่ 2-12 ฟิลด์ในทีซีพีเฮดเดอร์ที่ใช้ควบคุมการส่งข้อมูล 13
รูปที่ 2-13 การเลื่อนหน้าต่างฝ่ายส่ง 13
รูปที่ 2-14 การเลื่อนหน้าต่างฝ่ายรับ 14
รูปที่ 2-15 การประกาศขนาดหน้าต่าง 15
รูปที่ 2-16 แสดงการทำแฟร็กเมนเตชัน 16
รูปที่ 2-17 แสดงการรีแอสเซมเบิ้ล 16
รูปที่ 2-18 แสดงแพ็กเก็ตไอพี 17
รูปที่ 3-1 ฟอร์แมตของดาต้าแกรม 19
รูปที่ 3-2 ฟอร์แมตของฟิลด์ TOS 20
รูปที่ 3-3 การแฟรกเมนต์ 22
รูปที่ 4-1 แผนภูมิสถานะทีซีพี 25
รูปที่ 4-2 สถานะการเริ่มต้นการเชื่อมทีซีพี 27
รูปที่ 4-3 สถานะการปิดการเชื่อมต่อ 28
รูปที่ 4-4 การเปิดพร้อมกัน 29
รูปที่ 4-5 การปิดพร้อมกัน 30
รูปที่ 4-6 การปิดครึ่งเดียว (half close) 31
รูปที่ 5-1 การทำงานของแคปเจอร์ 34
รูปที่ 6-1 TCP header 41
รูปที่ 6-2 UDP header 41
รูปที่ 6-3 Firewall Packet Inspection Logic 43

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพประกอบ(ต่อ)

หน้าที่

รูปที่ 6-4 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN packet (A) 45
รูปที่ 6-5 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN packet (B) 45
รูปที่ 6-6 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN และ SYN/ACK packet (A) 45
รูปที่ 6-7 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN และ SYN/ACK packet (B) 46
รูปที่ 6-8 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN/ACK packet (A) 46
รูปที่ 6-9 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN/ACK packet (A) 47
รูปที่ 7-1 แสดงการจัดการคิวของทีซีพี และแอปพลิเคชั่น แยกกัน 50
รูปที่ 7-2 แสดงการ Scan Port แบบ Half Scan 51
รูปที่ 7-3 แสดงการ Scan Port แบบ Full Scan 51
รูปที่ 8-1 ไฟร์วอลล์กั้นระหว่างอินเตอร์เน็ตกับเน็ตเวิร์กภายใน 53
รูปที่ 8-2 ความแตกต่างระหว่างไฟร์วอลท์และโปรแกรมดักจับข้อมูล 54
รูปที่ 8-3 โฟลว์ชาร์ตของโปรแกรมวิเคราะห์แพ็กเก็ตแบบสเตตฟูล 55
รูปที่ 8-4 โฟลว์ชาร์ตแสดงการทำงานเมื่อ Syn แพ็กเก็ตเข้ามา 55
รูปที่ 8-5 โฟลว์ชาร์ตแสดงการทำงานเมื่อ Ack แพ็กเก็ตเข้ามา 56
รูปที่ 8-6 โฟลว์ชาร์ตแสดงการทำงานเมื่อ Data แพ็กเก็ตเข้ามา 56

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

หน้าที่

ตารางที่ 6-1 ตัวอย่างของกฎที่ตั้งไว้สำหรับไฟร์วอลล์แบบ Packet Filtering 38
ตารางที่ 6-2 ตัวอย่าง State table 42
ตารางที่ 7-1 แสดงขนาดของ backlog คิว 50
ตารางที่ 8-1 ตัวอย่างของ State Table ที่ใช้ภายในโปรแกรม 57

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันพัฒนาการทางด้านการติดต่อสื่อสารนับว่าได้ก้าวหน้าไปมาก มีการนำคอมพิวเตอร์มาใช้ในการติดต่อสื่อสารผ่านทางเครือข่ายเพิ่มมากขึ้นอย่างรวดเร็ว ด้วยความสะดวก ประหยัด และ รวดเร็วนี้เป็นเหตุผลที่ทำให้หลายๆคนมีชีวิตที่ขึ้นอยู่กับคอมพิวเตอร์มากขึ้น ยกตัวอย่างเช่น การรับส่งอีเมล์ การแชท การซื้อขายบนอินเทอร์เน็ต ฯลฯ

ด้วยความสามารถในการสื่อสารผ่านเครือข่ายนั้น ทำให้เกิดธุรกิจต่างๆขึ้นมากมายที่ใช้ความสามารถของการติดต่อสื่อสารผ่านทางเครือข่าย แต่จะรู้ได้อย่างไรว่าลูกค้าที่เข้ามาใช้บริการที่เปิดไว้ให้นั้นไม่ได้พยายามขโมยข้อมูลหรือพยายามโจมตีเครื่องเซิร์ฟเวอร์อยู่ จากที่กล่าวมาเป็นการยากที่จะป้องกันเนื่องจากว่าไม่สามารถเฝ้าดูพฤติกรรมของผู้ใช้ทุกๆ คนในเวลาเดียวกันได้ ฉะนั้นผู้ดูแลระบบจึงจำเป็นต้องมีสิ่งที่ช่วยเก็บข้อมูลของผู้ใช้แต่ละคนไว้ เพื่อนำไปวิเคราะห์ดูพฤติกรรมของผู้ใช้ในภายหลัง

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 ศึกษาโพรโตคอลทีซีพี/ไอพี เพื่อความเข้าใจในการรับส่งข้อมูลบนเครือข่าย

1.2.2 ศึกษาการใช้งานและการเขียนโปรแกรมบนระบบปฏิบัติการยูนิกซ์ (Unix)

1.2.3 ศึกษาโปรแกรมดักจับข้อมูลที่มีอยู่ในปัจจุบันและนำมาประยุกต์ใช้

1.2.4 เพื่อเป็นประโยชน์สำหรับผู้ดูแลระบบในการเฝ้าดูพฤติกรรมของผู้ใช้แต่ละคน

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้จะสร้างโปรแกรมดักจับข้อมูลขึ้นมา โดยอาศัยหลักการของสเตตฟูลในการวิเคราะห์แพ็กเก็ตที่เข้ามา โดยได้นำความสามารถของสเตตฟูลอินสเปกชันไฟวอลท์ (Stateful Inspection Firewall) มาประยุกต์ใช้ในการตรวจจับข้อมูล

แต่งานวิจัยนี้จะไม่สนใจในแง่ของการวิเคราะห์พฤติกรรมของผู้ใช้แต่อย่างใด สิ่งที่ทำได้คือการดักจับข้อมูลตามที่ผู้ดูและระบบได้กำหนดเงื่อนไขเอาไว้ สิ่งที่เป็นผลลัพธ์ของโปรแกรมวิเคราะห์แพ็กเก็ตแบบสเตตฟูลนี้จะเป็นประโยชน์กับการนำไปใช้ในการวิเคราะห์พฤติกรรมของผู้ใช้ต่อไป

นอกจากนั้นในโครงงานนี้ยังถือว่าเป็นโครงการที่ทดลองสร้าง เพื่อศึกษาความเป็นไปได้ในการใช้งาน ดังนั้นจึงมีข้อจำกัดของข้อมูลบางอย่าง แต่ถึงอย่างไรก็ยังเพียงพอต่อการทดสอบอย่างแน่นอน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลัก ๆ อยู่ 3 เรื่องด้วยกัน คือ โพรโตคอลทีซีพี/ไอพี สเตตฟูลอินสเปกชันไฟวอลท์ (Stateful Inspection Firewall) และวิธีการเขียนโปรแกรมบนระบบปฏิบัติการยูนิกซ์ ซึ่งรายละเอียดของโพรโตคอลทีซีพี/ไอพีอยู่ในบทที่ 2 ,3 และ 4 ประกอบไปด้วยความรู้พื้นฐานของไอพีโพรโตคอล ทีซีพีโพรโตคอล โพรโตคอล การสถาปนาการเชื่อมต่อ การยกเลิกการเชื่อมต่อ สถานะต่างๆของโพรโตคตอลทีซีพี

ในบทที่ 5 แพ็กเก็ตสนิฟเฟอร์ (Packet Sniffer) อธิบายถึงหลักการทำงานของการดักอ่านข้อมูลและการนำไปใช้ บทที่ 6 จะเป็นโมดูลของสเตตฟูลอินสเปกชันไฟวอลท์ จะกล่าวถึงการป้องกันการถูกโจมตีด้วยการ SYN Flood จุดบกพร่องของไฟวอลท์เวอร์ชันเก่า และรายละเอียดของโมดูลใหม่ที่นำเสนอ และอธิบายให้เข้าใจถึงตารางสถานะ(State Table) ของสเตตฟูลอินสเปกชันไฟวอลท์ เพื่อใช้ในการเขียนโปรแกรม ส่วนในบทที่ 7 กล่าวถึงการโจมตีเครือข่ายและหลักการทำงานของ SYN Flooding

จากนั้นก็จะเริ่มเข้าสู่ขั้นตอนของการออกแบบโปรแกรม โดยบทที่ 8 กล่าวถึงองค์ประกอบโดยรวมของระบบที่พัฒนาขึ้นมาทั้งหมด และบทที่ 9 ซึ่งเป็นบทสุดท้ายเป็นการสรุปการทำงาน ผลที่ได้รับจากงานวิจัยชิ้นนี้ และแนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่2
โพรโตคอลทีซีพี/ไอพี

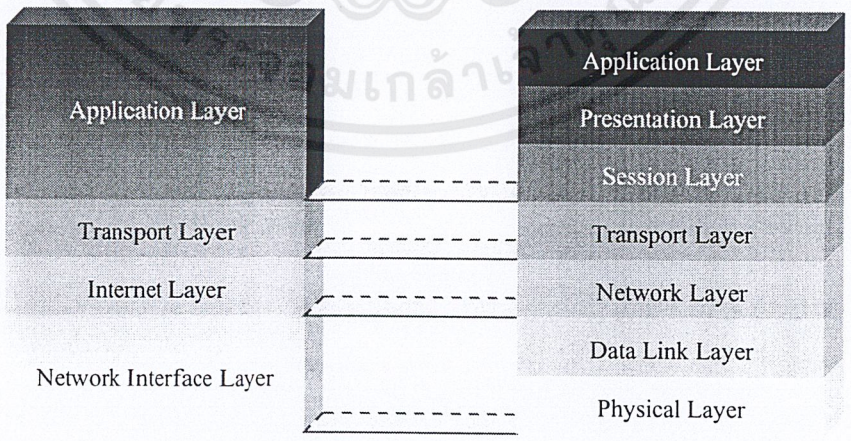
2.1 ความเป็นมาของโพรโตคอลทีซีพี/ไอพี

ทีซีพี/ไอพี เป็นมาตรฐานการรับส่งข้อมูลที่ใช้กันอย่างแพร่หลายในปัจจุบัน เริ่มพัฒนาขึ้นโดยกระทรวงกลาโหมของสหรัฐ เริ่มทำการทดลองในปี ค.ศ. 1969 เพื่อเชื่อมโยงคอมพิวเตอร์ทางทหารของแต่ละหน่วยเข้าด้วยกัน ซึ่งเป็นคอมพิวเตอร์ต่างชนิดกันให้สามารถติดต่อรับส่งข้อมูลกันได้ โครงการนี้มีชื่อว่า Advanced Research Project Agency Network หรือ ARPANET

ต่อมาได้มีพัฒนาเป็นเครือข่ายอินเทอร์เน็ต โพรโตคอลทีซี/ไอพีสามารถเชื่อมต่อระหว่างเครือข่ายอื่นเข้าด้วยกัน การเชื่อมต่อระหว่างเครือข่ายสามารถทำงานได้กับสารสื่อสารและอุปกรณ์ฮาร์ดแวร์ที่แตกต่างกันได้หรือใช้ระบบปฏิบัติการที่แตกต่างกันก็ตาม และยังสามารถสื่อสารถึงกันได้แม้ว่าอุปกรณ์เครือข่ายบางจุดหยุดทำงานหรือเส้นทางถูกตัดขาด โพรโตคอลทีซีพี/ไอพีประกอบด้วยชุดโพรโตคอลต่างๆ แต่ละโพรโตคอลมีความสามรถในการทำงานที่แตกต่างกัน ภายในบทนี้เราจะกล่าวถึงรายละเอียดและความสำคัญบางโพรโตคอล

2.2 โครงสร้างโพรโตคอล ทีซีพี/ไอพี

โพรโตคอล ทีซีพี/ไอพี การจัดลักษณะกลไกการการทำงานเป็นชั้นหรือเลเยอร์ (Layer) เรียงต่อกัน สามารถเปรียบเทียบกับโมเดลอ้างอิงโอเอสไอ (Open System Interconnection Reference Model: OSI) ตามมาตรฐานไอเอส (International Organization for Standardization: ISO) ดังรูปที่



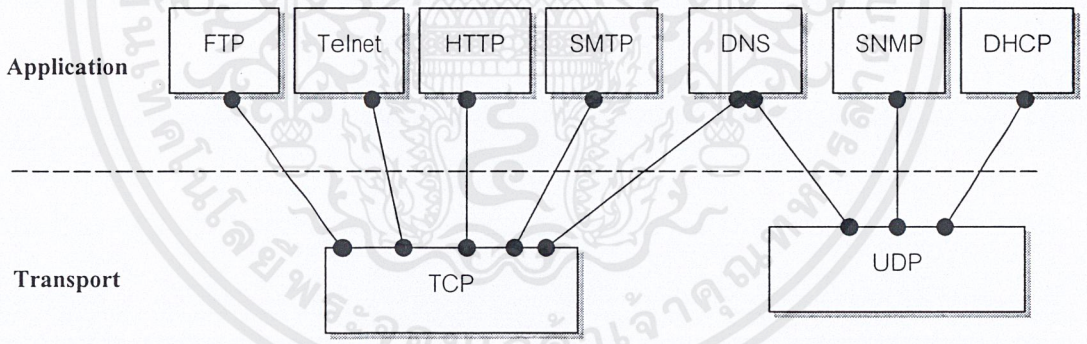
รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของทีซีพี/ไอพีกับเลเยอร์ของโอเอสไอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งเราจะเห็นว่าบางกลไกของโพรโตคอล ทีซีพี/ไอพี เทียบได้กับมาตรฐาน โอเอสไอ สองชั้นหรือบางกลไกก็ทำงานคาบเกี่ยวกันระหว่างบางชั้นของโมเดลอ้างอิงโอเอสไอ ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชันจนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังต่อไปนี้

2.2.1 ชั้นแอพพลิเคชัน (Application Layer) ชั้นแอพพลิเคชันนี้รองรับการทำงานของแอพพลิเคชันต่างๆ ที่ทำงานเป็นโพรเซส อยู่ในเครื่องในเครื่องที่ให้บริการ (Sever) และเครื่องที่ขอใช้บริการ (Client) โดยจัดการเชื่อมต่อระหว่างโพรเซส หรือแอพพลิเคชันที่อยู่ต่างเครื่องกัน โดยการทำงานของแอพพลิเคชันต่างๆ มีการติดต่อกันตามแต่ละโพรโตคอลเฉพาะแล้วแต่แอพพลิเคชันที่ใช้งาน ซึ่งจะขอบริการจากชั้นทรานสปอร์ตอีกทีหนึ่ง โพรโตคอลหลักๆ ที่ทำงานชั้นแอพพลิเคชันได้แก่ FTP (File transfer Protocol), Telnet, HTTP (HyperText Transfer Protocol), SMTP (Simple Mail Transfer protocol)

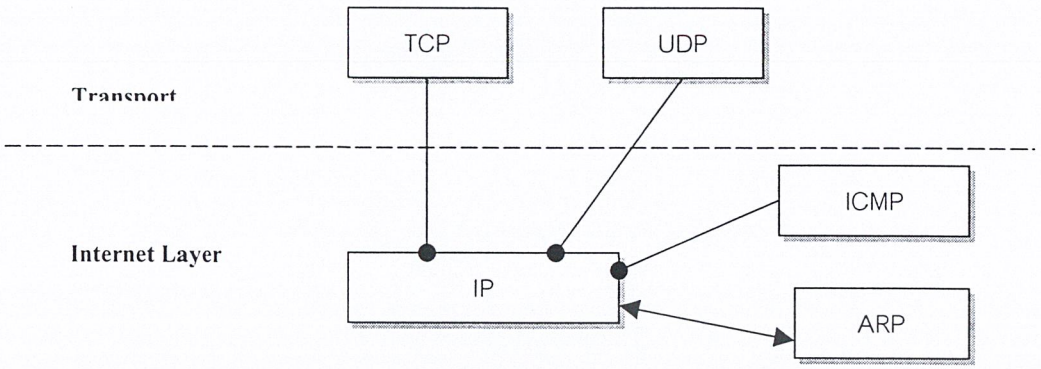
2.2.2 ชั้นทรานสปอร์ต (Transport Layer) มีหน้าที่จัดการต่อจากชั้นแอพพลิเคชัน ทำหน้าที่สร้างการเชื่อมต่อระหว่างแอพพลิเคชันแบบ end-to-end โดยจุดที่เชื่อมต่อกันเพื่อรับส่งข้อมูลนี้เรียกว่าพอร์ต (port) หรือซ็อกเก็ต (Socket) ในชั้นนี้มีบริการหลักอยู่ 2 แบบ คือ Connection Oriented โดยเรียกผ่าน โพรโตคอลทีซีพี (TCP: Transmission Control Protocol) และ Connectionless ซึ่งเรียกผ่านโพรโตคอลยูดีพี (UDP: User Datagram Protocol)



รูปที่ 2-2 โพรโตคอลต่างที่เรียกใช้ทรานสปอร์ตเลเยอร์

2.2.3 ชั้นอินเทอร์เน็ต (Internet Layer) ชั้นนี้มีหน้านี้ส่งผ่านข้อมูลระหว่างเครือข่าย โดยมีโพรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆ ในอินเทอร์เน็ต คือ ไอพี (Internet Protocol: IP) นอกจากนี้ในชั้นนี้ยังมีโพรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ ไอซีเอ็มพี (Internet Control Message Protocol: ICMP) และเออาร์พี (Address Resolution Protocol: ARP)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

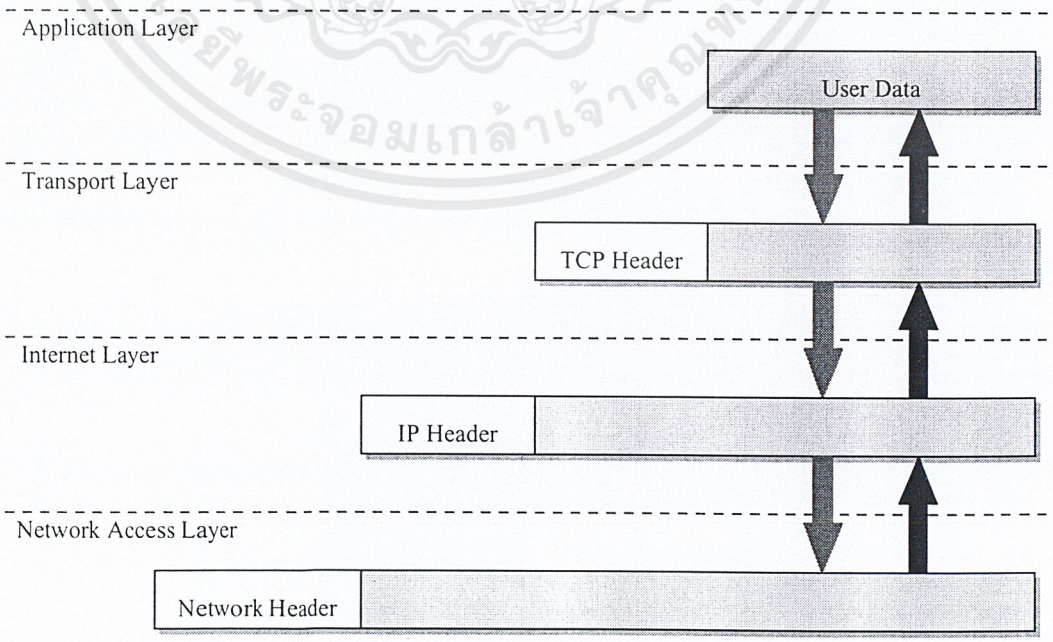


รูปที่ 2-3 โพรโตคอล TCP และ UDP อาศัยโพรโตคอล IP เพื่อส่งข้อมูลระหว่างเครือข่าย

2.2.4 ชั้นเน็ตเวิร์กอินเทอร์เฟซ (Network Interface Layer) ทำหน้าที่ในการแปลงข้อมูลให้อยู่ในรูปที่เหมาะสมกับเครือข่ายแต่ละแบบ ซึ่งแตกต่างกันออกไป และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่าย

2.3 การส่งถ่ายข้อมูลระหว่างชั้น

การส่งผ่านข้อมูลจากเครื่องที่ให้บริการไปยังเครื่องที่ข้อใช้บริการนั้น ข้อมูลจะส่งผ่านข้อมูลจากโพโตคอลที่อยู่ระดับบนสุดของเครื่องที่ให้บริการไปยังระดับล่างจนข้อมูลถูกแปลงให้เป็นสัญญาแล้วเดินทางผ่านเครือข่ายไปยังเครื่องขอใช้บริการ โพรโตคอลระดับล่างสุดที่เครื่องขอใช้บริการจะแปลงสัญญาที่รับมาแล้วส่งผ่านขึ้นไปยังโพร โตคอลระดับบนต่อไป



รูปที่ 2-4 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

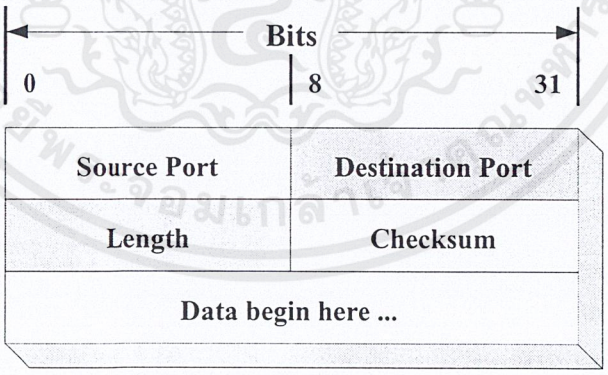
เมื่อข้อมูลผ่านแต่ละระดับชั้น โพโตคอลในแต่ละชั้นจะผนวกข่าวสารและการทำงานของโพโตคอลภายในชั้นนั้นเข้าไปเรียกว่า โพรโตคอลเฮดเดอร์ (protocol header) เข้ากับข้อมูลโพรโตคอลระดับล่างมองเฮดเดอร์ และข้อมูลเหมือนเป็นข้อมูลและเพิ่มข้อมูลของชั้นเข้าไป ข้อมูลจะถูกหุ้มเป็นชั้นๆ กระบวนการนี้เรียกว่า การเอ็นแคปซูเลต (encapsulation) และเมื่อเครื่องที่ข้อใช้บริการได้รับแพ็กเก็ตก็จะดำเนินการส่งไปตามลำดับชั้น โพรโตคอลประจำชั้นถอดเฮดเดอร์ออกและส่งส่วนที่เหลือไปยังชั้นถัดไป เฮดเดอร์จะถูกถอดออกเหลือเฉพาะข้อมูลเมื่อถึงชั้นบนสุด กระบวนการนี้เรียกว่า การดีแคปซูเลต (decapsulation) ดั้งรูป

ในโพรโตคอลทีซีพี/ไอพีนี้ มีโพรโตคอลหลักที่ขอกล่าวถึง 3 โพรโตคอล ได้แก่ โพรโตคอลทีซีพี โพรโตคอลยูดีพี ซึ่งทำงานในชั้นทรานสปอร์ต และโพรโตคอลไอพี ทำงานในชั้นอินเทอร์เน็ต โดยมีรายละเอียดดังต่อไปนี้

2.4 โพรโตคอลยูดีพี (UDP :User Datagram Protocol)

โพรโตคอลที่ให้บริการในระดับเป็นโพรโตคอลแบบ Connectionless คือไม่มีการทำเฮนด์เช็กกับเครื่องปลายทาง และไม่มีการตรวจสอบความถูกต้องของข้อมูลที่รับมา ทำให้มีความเร็วในการทำงานสูง(ในเลเยอร์นี้) เหมาะกับงานที่ส่งข้อมูลขนาดเล็กและส่งบ่อย ๆ

หากมีปัญหาเกิดขึ้นกับยูดีพีเดต้าแกรม เช่นเดต้าแกรมสูญหาย หรือผิดลำดับหรือมีเดต้าแกรมซ้ำกัน ยูดีพีไม่จัดการกับปัญหาเหล่านี้เนื่องจากไม่มีกลไกที่รับประกันความถูกต้องของเดต้าแกรม โพรโตคอลประยุกต์ที่ใช้ยูดีพีต้องแก้ปัญหานี้เอง



รูปที่ 2-5 ยูดีพีเดต้าแกรม

<u>ส่วนประกอบของยูดีพีเดต้าแกรม</u>

1. **Source Port** ขนาด 16 บิต : เป็นหมายเลขพอร์ตของบริการที่เครื่องต้นทาง
2. **Destination Port** ขนาด 16 บิต : เป็นหมายเลขพอร์ตของบริการเครื่องปลายทาง
3. **Length** ขนาด 16 บิต : บอกความยาวของเดต้าแกรม (ทั้งเฮดเดอร์และข้อมูล) เป็นจำนวนไบต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. **Checksum** ขนาด 16 บิต : เป็นส่วนที่ตรวจสอบความถูกต้องของข้อมูล

ยูดีพีเดต้าแกรมมีเฮดเดอร์ที่เรียบง่ายและมีฟิลด์จำนวนน้อย ฟิลด์ที่สำคัญมีเพียง source port และ destination port

และ checksum โดยไม่มีฟิลด์อื่นใดใช้ในการรับประกันความน่าเชื่อถือในการส่งข้อมูล

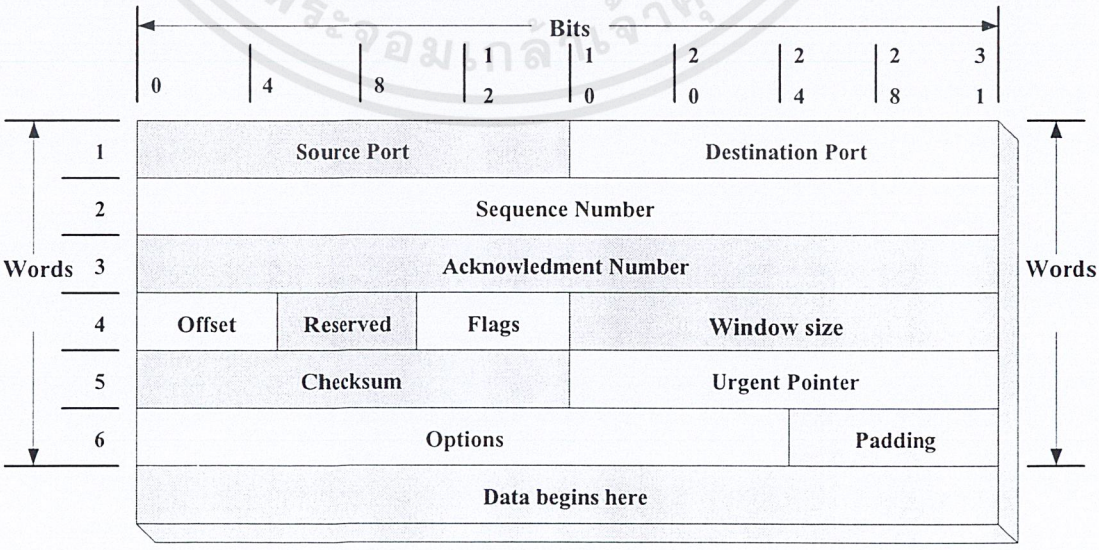
2.5 โพรโตคอลทีซีพี (TCP: Transmission Control Protocol)

ทีซีพีเป็นโพรโตคอลแบบ connection – Oriented ที่ให้บริการงานการรับส่งข้อมูลโดยรับประกันการเชื่อถือในการลำเลียงข้อมูล ทีซีพีรับประกันความเชื่อถือโดยทำหน้าที่ตรวจเซกเมนต์ที่ผิดปกติและจัดส่งเซกเมนต์ซ้ำใหม่ รวมทั้งการจัดลำดับให้ถูกต้องก่อนส่งต่อไปยังโปรแกรมประยุกต์ระดับบน เฮดเดอร์และข้อมูลของทีซีพี เรียกว่า เซกเมนต์ (segment)



รูปที่ 2-6 ทีซีพีเซกเมนต์

การส่งข้อมูลของทีซีพีนั้นส่งทีละเซกเมนต์โดยเครื่องปลายทางส่งสัญญาตอบรับมายังเครื่องต้นทางทุกเซกเมนต์ที่ได้รับและตรวจสอบแล้วว่าไม่พบข้อผิดพลาด ถ้าหากเครื่องต้นทางไม่ได้รับสัญญาตอบกับมาสำหรับเซกเมนต์ที่ส่งไป สันนิฐานว่าเซกเมนต์นั้นมีปัญหาขึ้นและจะส่งเซกเมนต์นั้นไปอีกครั้ง วิธีการนี้เราเรียกว่า Positive Acknowledgment with Re-transmission (PAR)



รูปที่ 2-7 แสดงแพ็กเก็ตทีซีพี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

<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. **Flags** : เป็นบิตที่บอกชนิดของข้อมูล ได้แก่
   - URG : Urgent Pointer Field Significant - แสดง Urgent Pointer
   - ACK : Acknowledgement Field S7ignificant – แสดงการ Acknowledgement
   - PSH : Push Function
   - RST : Reset The Connection – แสดงเมื่อรีเซ็ตการเชื่อมต่อ
   - SYN : Synchronize Sequence Number - หมายเลขแพ็กเก็ตที่ส่งแบบซิงโครนัส
   - FIN : No more data from sender - แสดงว่าไม่มีข้อมูลที่ส่งจากผู้ส่งแล้ว
7. **Window size** : เป็นเลขบอกจำนวนของอ็อกเต็ต (octet) ของข้อมูล จัดการในส่วนของ 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 บิตแรกสงวนไว้และกำหนดให้เป็นศูนย์)

2.5.1 กลไกการทำงานของทีซีพี

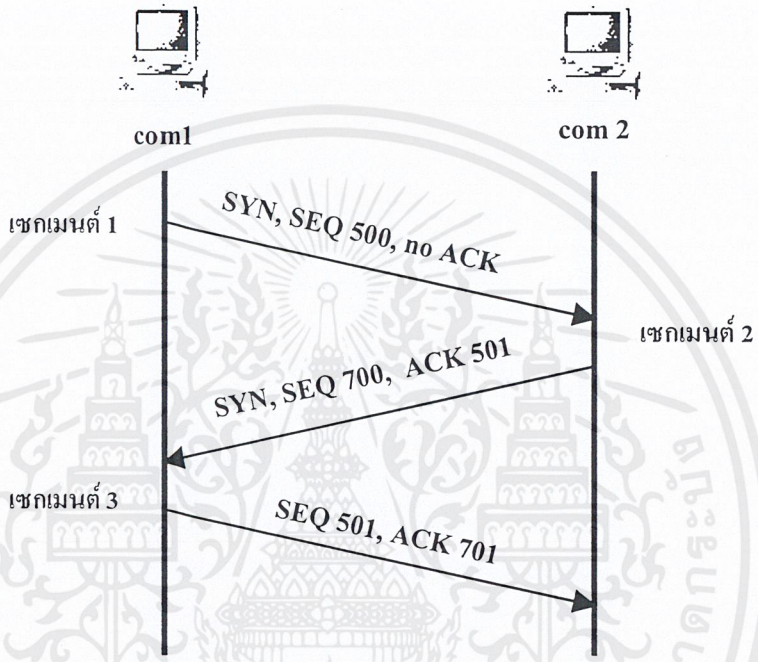
ทีซีพีมีกลไกในการลำเลียงข้อมูลระหว่างต้นทางและปลายทางหลายส่วน ในที่นี้จะกล่าวเฉพาะกลไกที่สำคัญ 3 ประการได้แก่ กระบวนการเริ่มต้นในการเชื่อมต่อ กระบวนการถ่ายโอนข้อมูล และกระบวนการยกเลิกการเชื่อมต่อ

2.5.1.1 การเริ่มต้นการเชื่อมต่อ

การทำ "3-way Handshake" ซึ่งเป็นกระบวนการเริ่มต้นในการสร้างการเชื่อมต่อในชั้นทรานสปอร์ต กล่าวคือ ในการติดต่อกันระหว่างระบบในเครือข่ายต้องมีการสร้างการเชื่อมต่อไปยังระบบที่ให้บริการก่อน โดยผู้ขอบริการส่งสัญญาณ SYN เพื่อขอบริการ จากนั้นผู้ให้บริการจะส่งสัญญาณ ACK เพื่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตอบรับการเชื่อมต่อที่ร้องขอมา จึงสามารถรับส่งข้อมูลกันได้ ตัวอย่างการเริ่มต้นการเชื่อมต่อระหว่างเครื่อง com1 และ com2 โดย com1 เป็นฝ่านขอบริการจาก com2 โดยโพโตคอลประยุกต์ของด้าน com1 จะเป็นไคลเอ็นต์ และโพรโตคอลประยุกต์ทางด้าน com2 เป็นเซิร์ฟเวอร์ กระบวนการทำ "3-way Handshake " มีรายละเอียดดังนี้



รูปที่ 2-8 การเริ่มต้นการเชื่อมต่อด้วยทีซีพี

ทีซีพีของ com1 เลือกเลขลำดับเริ่มต้นแล้วส่งเซกเมนต์ที่บรรจุเลขลำดับนี้ไปยัง com2 พร้อมทั้งเซ็ตแฟล็กของ SYN ให้เป็น "1" สังเกตว่าเซกเมนต์ 1 แฟล็ก ACK จะมีค่าเป็น "0"

ทีซีพีของเครื่อง com2 ไดรับเซกเมนต์จาก com1 ก็จะเลือกเลขลำดับเริ่มต้นประจำตัวเช่นเดียวกันแล้วตอบกลับด้วยเซกเมนต์ 2 โดยเซ็ตแฟล็กของ SYN และ ACK ให้เป็น "1" ทั้งคู่ เพื่อแจ้งว่าได้รับเซกเมนต์1 แล้ว และเลขลำดับที่ได้รับจาก com2 บวกด้วยหนึ่ง

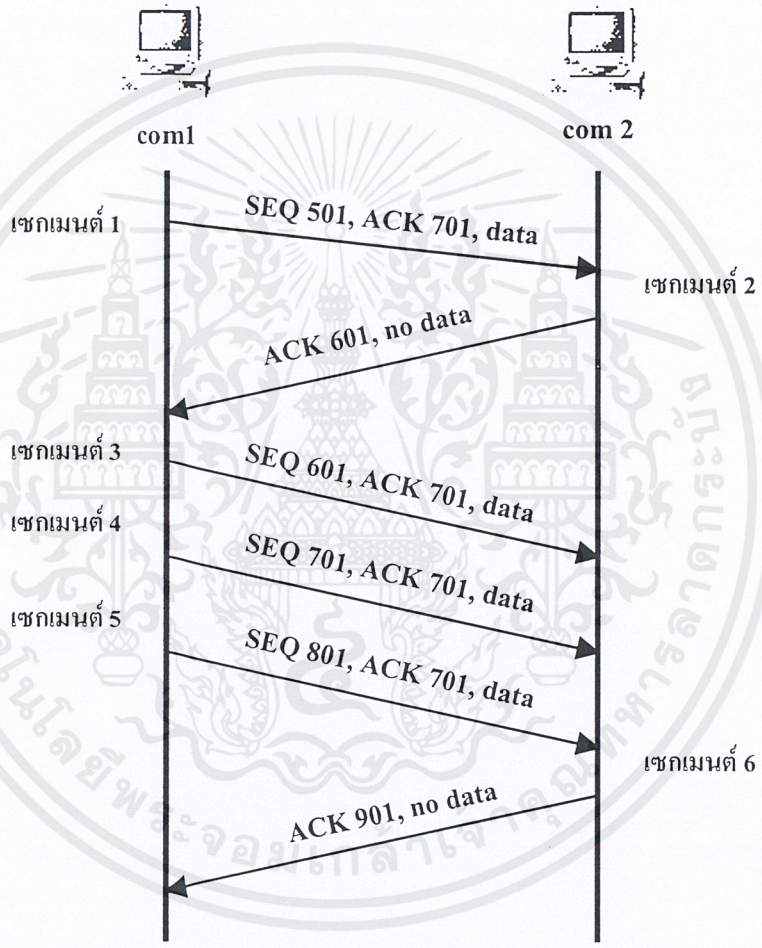
ทีซีพีของ com1 จะส่งเซกเมนต์ตอบรับกลับไปโดยเซ็ตแฟล็ก ACK และใช้เลขลำดับที่ได้รับจาก com2 บวกด้วยหนึ่ง

ต่อจากนั้นทีซีพีของ com1 แจ้งไปยังโพรโตคอลระดับบนว่าเชื่อมต่อแล้ว ส่วน com2 เมื่อได้รับเซกเมนต์3 ก็แจ้งขึ้นไปยังโพรโตคอลระดับบนว่าเชื่อมต่อแล้ว เมื่อสิ้นสุดนี้ การเชื่อมต่อทั้งสองด้านก็เสร็จสมบูรณ์และพร้อมรับส่งข้อมูลได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.1.2 การถ่ายโอนข้อมูล

การถ่ายโอนข้อมูลเริ่มต้นหลังจากได้เชื่อมต่อเสร็จสิ้นไปแล้วรูปที่ 2-9 แสดงขั้นตอนการถ่ายโอนข้อมูลจาก com1 ไปยัง com2 ครั้งละ 100 ไบต์ จำนวน 4 ครั้ง โดยเซกเมนต์1 ที่ com1 ส่งไปยัง com2 บรรจุข้อมูลไบต์ 501 ถึง 600 และใช้เลขตอบรับ 701 เมื่อ com2 รับข้อมูลเซกเมนต์1 แล้ว ก็จะตอบกับด้วยเซกเมนต์ 1 แล้ว ก็ตอบกลับด้วยเซกเมนต์ 2 พร้อมเลขตอบรับกำหนดข้อมูลที่คาดว่าจะได้รับต่อไปคือ 601

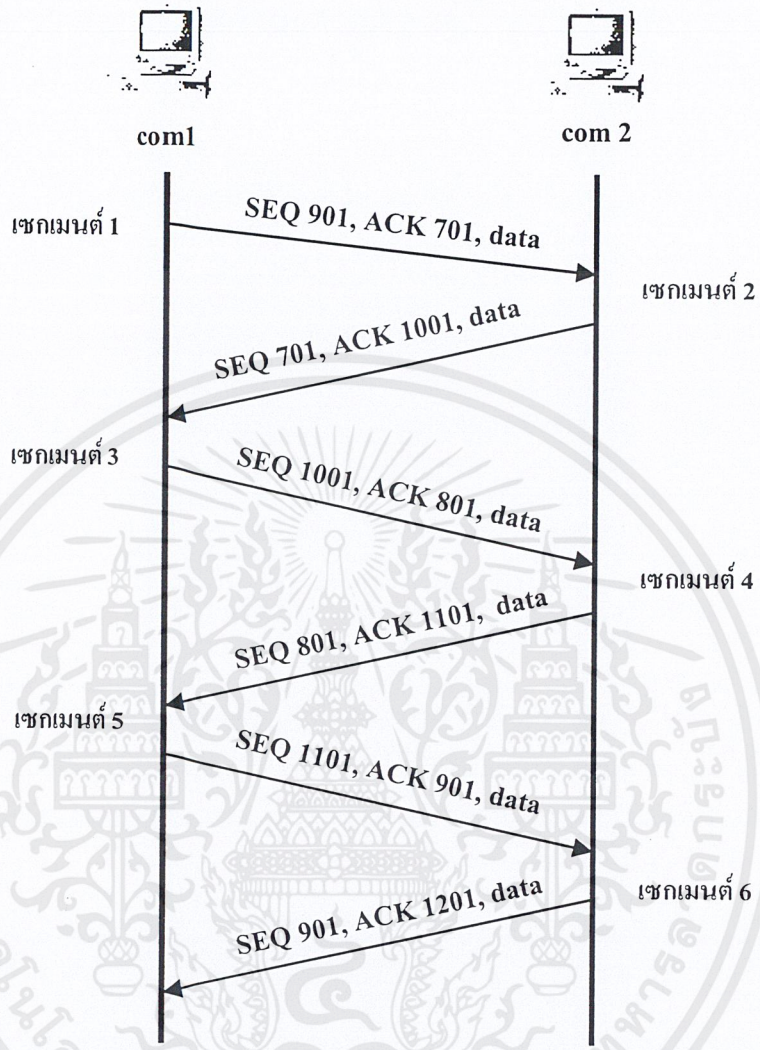


รูปที่ 2-9 ขั้นตอนการถ่ายโอนข้อมูล

com1 สามารถส่งเซกเมนต์ไปอย่างต่อเนื่องได้ ดังรูป 2-8 มีการส่งไป 3 เซกเมนต์ คือ 601, 701 และ 801 ทาง com2 สามารถตอบกลับในคราวเดี่ยวกันได้คือเซกเมต์ 6

ในกรณีที่ส่งข้อมูลทั้งสองทิศทางระหว่าง com1 และ com2 หลักการทั่วไปยังคงเหมือนเดิม เพียงแต่มีการตอบรับพร้อมกับส่งข้อมูลระหว่างกันดังรูปที่ 2-6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2-10 การถ่ายโอนเซกเมนต์ทั้งสองทิศทาง

2.5.1.3 การยกเลิกการเชื่อมต่อ

เนื่องจากการถ่ายโอนในทีซีพีเป็นแบบฟูลดูเพล็กซ์ การยกเลิกการเชื่อมต่อจึงต้องมีขั้นตอนเกิดขึ้นทั้งสองด้านดังรูปที่ ในที่นี้สมมติว่า com1 ไม่มีข้อมูลส่งอีกต่อไปจึงต้องเป็นฝ่ายขอปิดบริการเชื่อมต่อโดยโพรโตคอลประยุกต์ของ com1 จะแจ้งไปยังทีซีพีว่าส่งข้อมูลหมดแล้วถัดจากนั้นจะมีการแลกเปลี่ยนเซกเมนต์จำนวน 4 เซกเมนต์ดังนี้

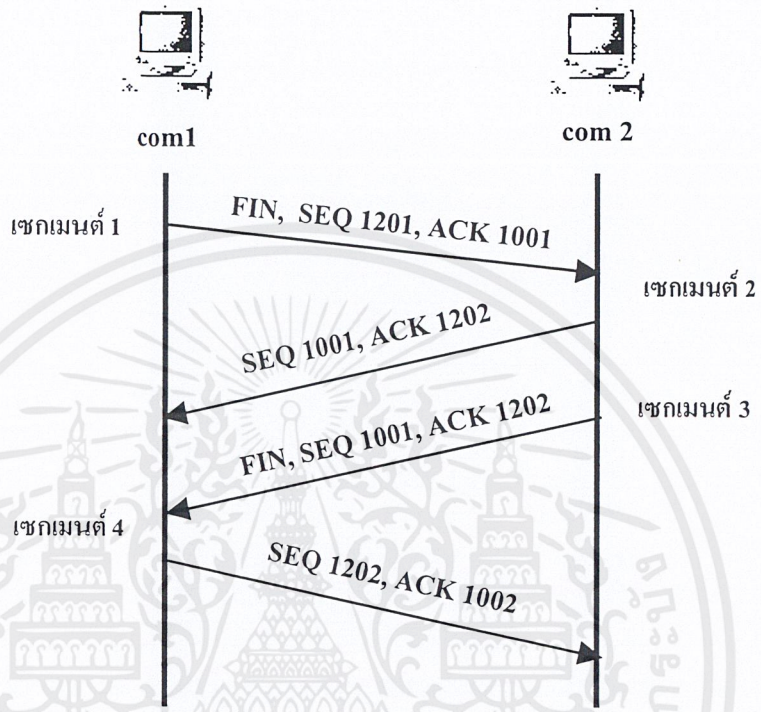
1. ทีซีพีของ com1 ส่งเซกเมนต์ที่มีเลขตอบรับและเลขลำดับตามปกติ แต่เซ็ตแฟล็ก FIN เป็น "1"

2. เมื่อทีพีซีพีของ com2 ได้รับเซกเมนต์ที่มีการเซ็ต FIN จาก com1 จะส่งเซกเมนต์ตอบรับคือเซกเมนต์ 2 และแจ้งไปยังโปรแกรมประยุกต์ com1 ขอปิดการเชื่อมต่อโปรแกรมประยุกต์ com2 แจ้งกลับมายังทีซีพีว่าขอปิดการเชื่อมต่อได้

3. ทีซีพี com2 ส่งเซกเมนต์ FIN ไปยัง com1 (เซกเมนต์ 3)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อ com1 ได้รับเซกเมนต์ FIN จะตอบรับกลับไป (เซกเมนต์ 4) และแจ้งไปยังโปรแกรมประยุกต์ว่าการเชื่อมโยงปิดลงแล้ว



รูปที่ 2-11 การปิดการเชื่อมต่อ

2.5.2 การเลื่อนหน้าต่าง (Sliding window)

ในสภาวะการถ่ายโอนข้อมูลระหว่างไคลเอ็นต์ com1 และ เซอร์ฟเวอร์ com2 หาก com1 ใช้หลักการตอบรับเป็นรายเซกเมนต์คือทุกครั้งที่ส่งเซกเมนต์ต้องรอการตอบจาก com2 ก่อนส่งเซกเมนต์ถัดไปได้ ช่วงที่ com1 รอการตอบรับนับไม่สามารถนำส่งข้อมูลได้และเท่ากับเป็นการใช้สายสื่อสารแบบฮาล์ฟดูเพล็กซ์ซึ่งทำให้เกิดการสูญเสียแบนด์วิดธ์ไปครึ่งหนึ่ง เนื่องจากต้องรอให้อีกด้านส่งข้อมูลเสร็จสิ้นก่อน

ทีซีพีแก้ปัญหานี้โดยอาศัยการทำงานแบบฟูลดูเพล็กซ์ ฝ่ายส่งสามารถส่งเซ็กเมนต์ออกได้หลายเซ็กเมนต์โดยที่ไม่ต้องรอการตอบรับทุกๆ เซกเมนต์ที่ส่งออกไป แต่ต้องมีการควบคุมปริมาณการส่งนี้เพราะบัฟเฟอร์ฝ่ายรับย่อมมีขนาดจำกัด ฝ่ายรับต้องแจ้งให้ทางฝ่ายส่งทราบถึงขนาดข้อมูลที่รับได้โดยใช้เลขตอบรับ (ฟิลด์ Acknowledgment number) และขนาดหน้าต่าง (ฟิลด์ windows size) สองค่านี้จึงใช้เป็นตัวกำหนดขนาดการรับส่งข้อมูลแต่ละครั้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Source Port			Destination Port
Sequence Number			
Acknowledgment Number : 1000			
Offset	Reserved	Flags	Window size : 1024
Checksum			Urgent Pointer
Options			Padding
Data begins here			

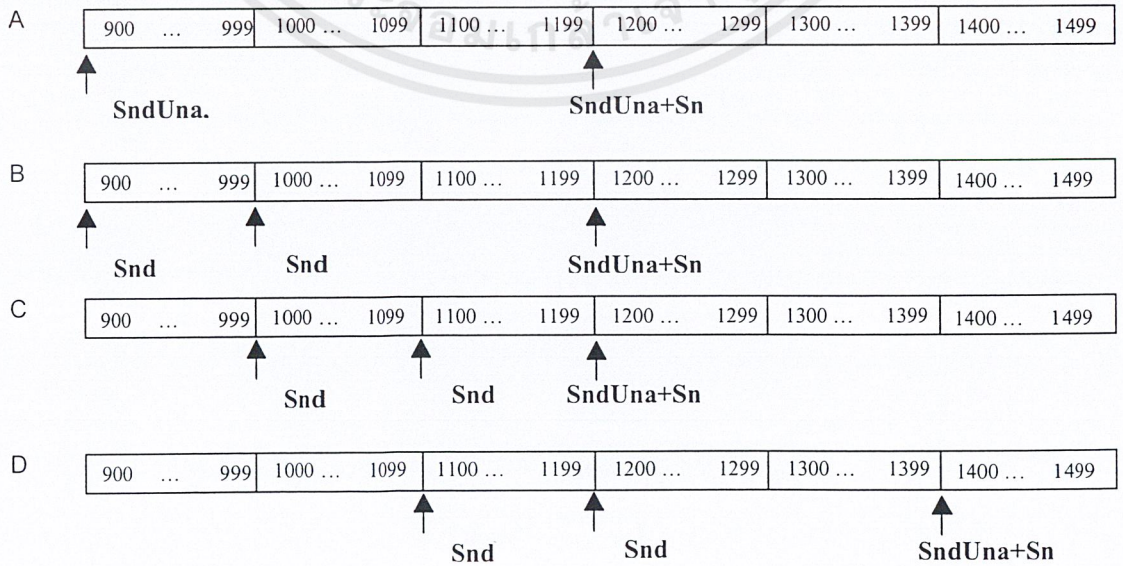
รูปที่ 2-12 ฟิลด์ในทีซีพีเฮดเดอร์ที่ใช้ควบคุมการส่งข้อมูล

จากรูปที่ 2-12 แสดงตัวอย่างทีซีพีเฮดเดอร์ซึ่งประกาศหน้าต่างขนาด 1024 ไบต์ ค่า Acknowledgment จะแจ้งว่าได้รับข้อมูลตั้งแต่ต้นถึงลำดับ 999 ครบถ้วนแล้ว และพร้อมที่จะรับข้อมูลไม่เกินเลขลำดับ 2023 (คำนวณ 1000-1+1024)

2.5.2.1 หน้าต่างฝ่ายส่ง

ทีซีพีควบคุมกระแสข้อมูลที่ฝ่ายส่งโดยใช้ค่า 3 ค่าคือ ตัวแปรชี้ตำแหน่งไบต์ที่ตอบรับ(SndUna), ตัวแปรชี้ตำแหน่งไบร์ที่ต้องส่งครั้งถัดไป (SndNxt) และ ตัวแปรกำหนดหน้าต่างที่ส่งได้(SndWnd) ค่าทั้ง 3 จะแบ่งบัฟเฟอร์ของฝ่ายส่งออกเป็น 4 ส่วน

เพื่อให้เห็นการทำงานของหน้าต่างฝ่ายรับ ขอให้พริจารณารูปที่ 2-13 โดยให้ในสภาวะเริ่มต้นทางฝ่ายส่งมีขนาดหน้าต่างเท่ากับ 300 ไบต์ และหมายเลขลำดับเริ่มต้นที่ 900 ดังรูป2-13 (A)



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2-13 การเลื่อนหน้าต่างฝ่ายส่ง

เมื่อส่งข้อมูลออกไป 100 ไบต์แรก SndNxt จะเลื่อนำไปทางขวา 100 ไบต์ดังรูป 2-13 (B) ความจำเป็นของตัวแปรชี้ตำแหน่งไบต์ที่ได้รับการตอบรับล่าสุดก็เพื่อใช้เก็บตำแหน่งของข้อมูลที่ส่งไปแล้วแต่ยังไม่ได้รับการตอบรับเนื่องจากอาจต้องส่งซ้ำใหม่

ขั้นถัดไปหากมีข้อมูลส่งอีก 100 ไบต์ SndNxt จะขยับไปอยู่ที่ตำแหน่ง 1100 และเมื่อมีการตอบรับข้อมูล 100 ไบต์แรกกลับมา SndUna จะเลื่อนไปที่ตำแหน่ง 1000 ตามรูปที่ 2-13 (C)

ในขั้นตอนถัดไปหากมีการตอบรับกลับมาพร้อมกับการกำหนดขนาดหน้าต่าง 300 ไบต์หน้าต่างจะเป็นดังรูปที่ 2-13 (D)

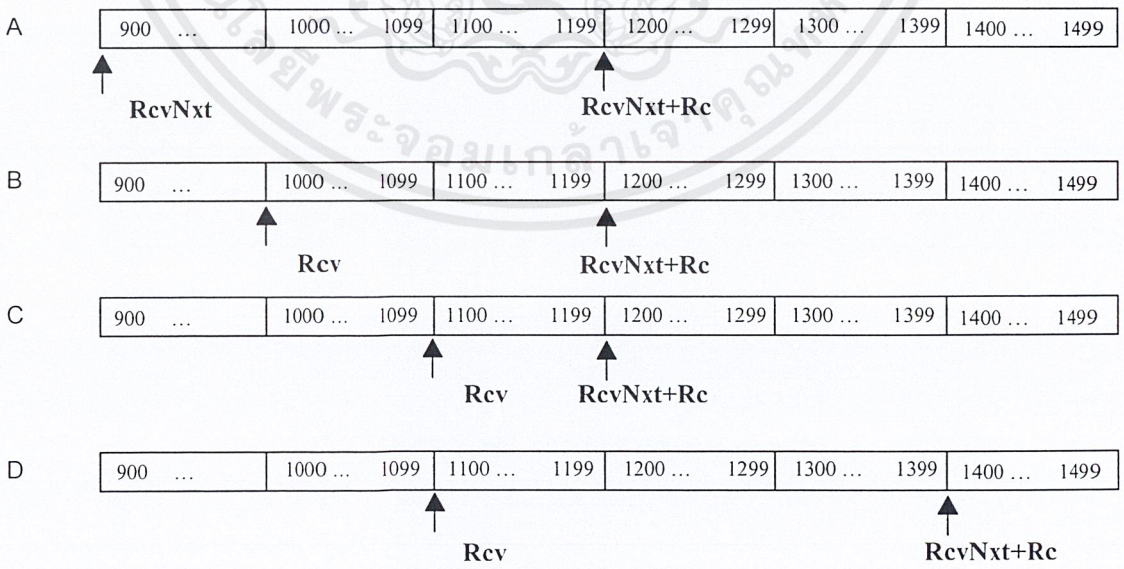
2.5.2.2 หน้าต่างฝ่ายรับ

ทีซีพีฝ่ายรับจัดการกับบัฟเฟอร์โดยให้มี ตัวแปรบอกขนาดหน้าต่าง (RcvWnd) และตัวแปรชี้ตำแหน่งข้อมูลถัดไปที่คาดว่าจะได้รับ (RcvNxt) ตัวแปรทั้งสองนี้จะแบ่งบัฟเฟอร์ออกเป็น 3 ส่วน

หากพิจารณาการเลื่อนหน้าต่างของฝ่ายรับโดยกำหนดให้มีขนาดหน้าต่างเริ่มต้นที่ 300 ไบต์ ดังรูปที่ 2-14 (A) เมื่อข้อมูล 100 ไบต์ แรกเดินทางมาถึง หน้าต่างข้อมูลที่รับได้จะลดลงเหลือ 200 ไบต์รูปที่ 2-14 (B)

ในขั้นต่อมาหากได้รับข้อมูลอีก 100 ไบต์ ขนาดหน้าต่างก็จะลดลงเหลือ 100 ไบต์ ดังรูปที่ 2-14 (C)

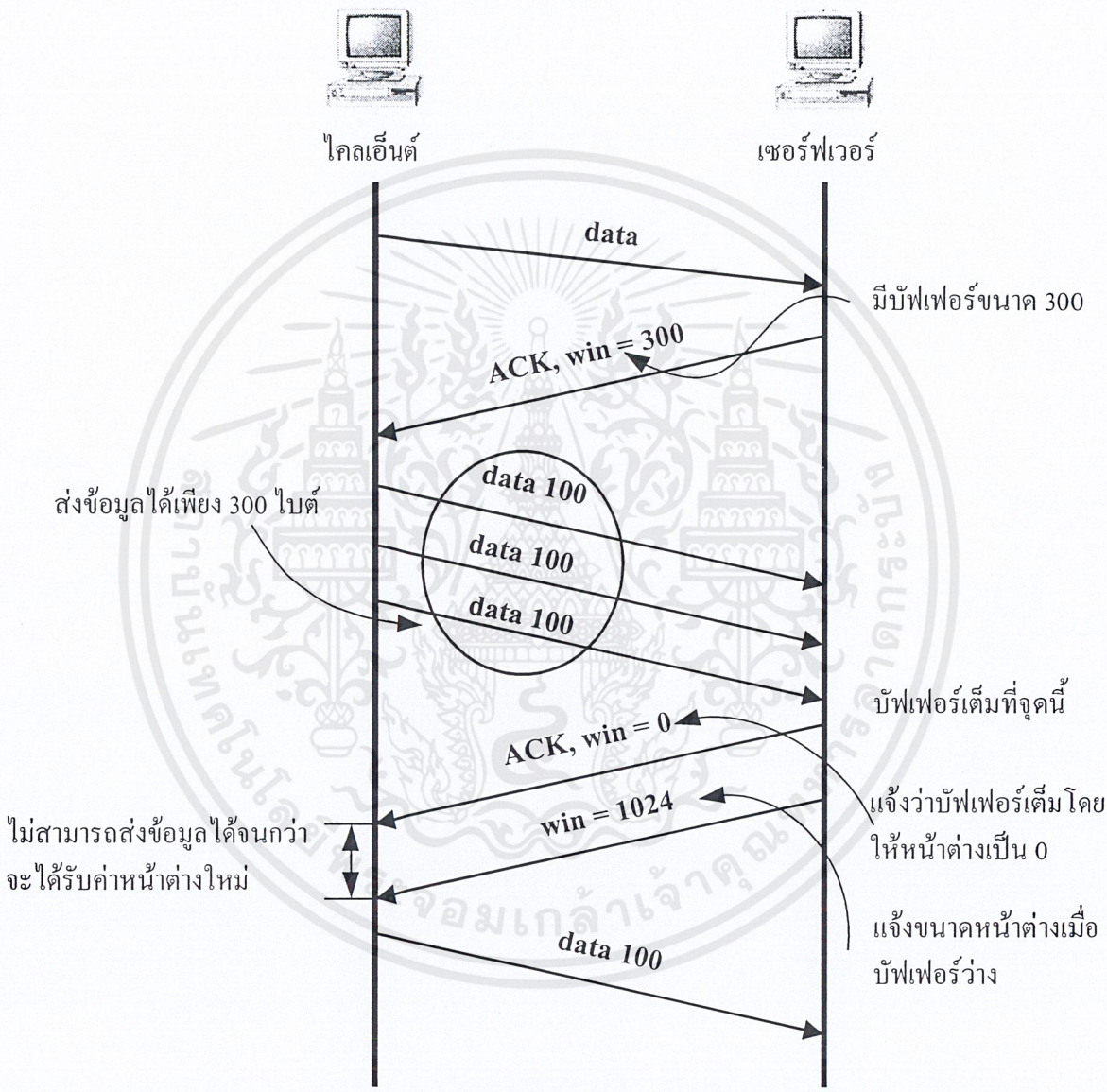
ในขั้นถัดมาเมื่อโปรแกรมประยุกต์นำข้อมูลไปใช้ หน้าต่างก็เลื่อนและขยายขนาดขึ้น ดังรูป 2-14(D)



รูปที่ 2-14 การเลื่อนหน้าต่างฝ่ายรับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในกรณีที่บัฟเฟอร์ฝ่ายรับเต็มจนไม่สามารถรับเซกเมนต์เพิ่มได้ ทีซีพีจะส่งเซกเมนต์กำหนดขนาดหน้าต่างโดยให้ฟิลด์ window size มีค่าเท่ากับ 0 ทีซีพีอีกฝ่ายจะหยุดส่งเซกเมนต์จนกว่าจะได้รับเซกเมนต์ที่กำหนดขนาดหน้าต่างใหม่ดังรูปที่ 2-15



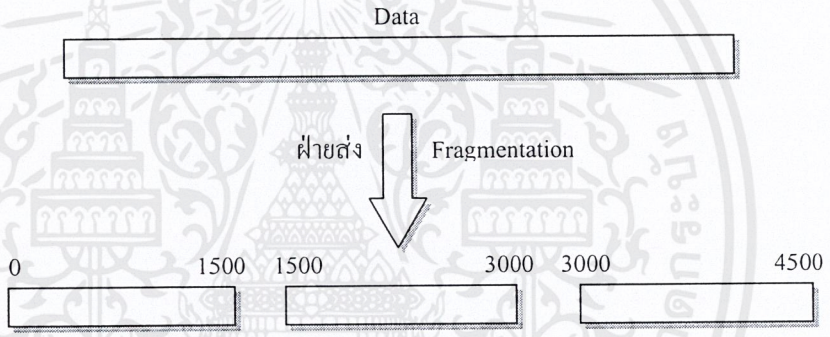
รูปที่ 2-15 การประกาศขนาดหน้าต่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

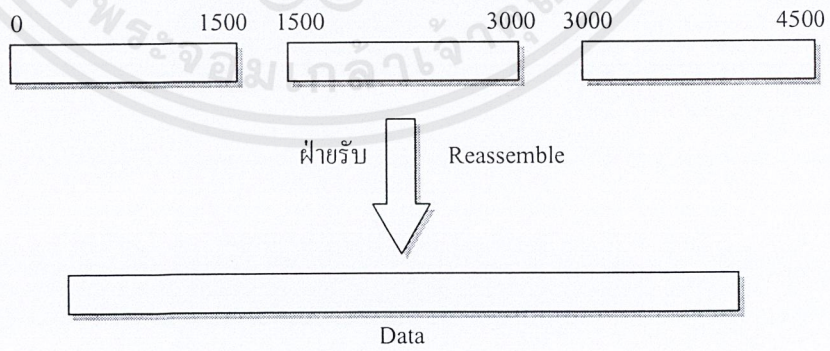
การรอการกำหนดค่าหน้าต่างใหม่เป็นกรณีหนึ่งที่ทีซีพีต้องดำเนินการเป็นกรณีพิเศษเพราะหากเซกเมนต์ประกาศค่าหน้าต่างใหม่เกิดสูญหาย จะทำให้การสื่อสารไม่สามารถดำเนินต่อไปได้ ทีซีพีแก้ปัญหากรณีนี้โดยใช้ตัวจับเวลาเพื่อรับประกันว่าต้องส่งเซกเมนต์กำหนดค่าหน้าต่างใหม่ภายในเวลาที่กำหนด

2.6 โพรโตคอลไอพี (IP: Internet Protocol)

โพรโตคอลไอพีเป็นโพรโตคอลที่จัดการเกี่ยวกับแอดเดรสของแต่ละแพ็กเก็ต เพื่อให้ส่งแพ็กเก็ตต่างๆ ไปยังเป้าหมายได้ถูกต้อง การทำงานของไอพีเป็นเพียงการส่งข้อมูลไปยังเครื่องเป้าหมายเท่านั้น ไม่มีการส่งสัญญาณขอบริการ หรือสัญญาณให้บริการระหว่างกันเหมือนทีซีพี เรียกว่าการเชื่อมต่อแบบ Connectionless ซึ่งระบบทั้งสองตั้งสมมติฐานว่าการเชื่อมต่อระหว่างกันไม่มีความผิดพลาดเกิดขึ้นแน่



รูปที่ 2-16 แสดงการทำแฟร็กเมนเตชัน



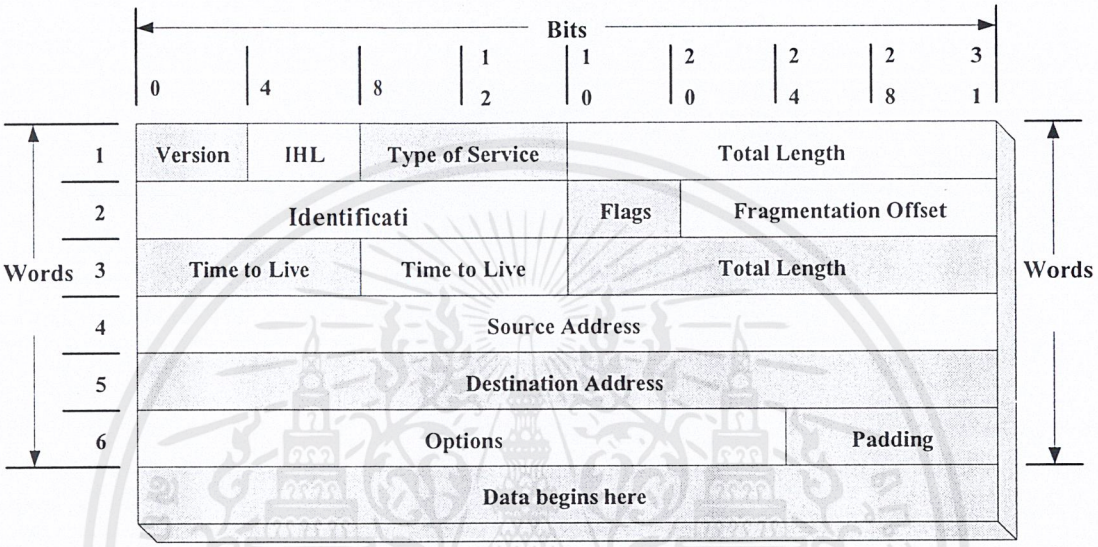
รูปที่ 2-17 แสดงการรีแอสเซมเบิ้ล

เนื่องจากมาตรฐานในเครือข่ายมีหลากหลาย ขนาดของแพ็กเก็ตในแต่ละมาตรฐานจึงมีความแตกต่างกันออกไป ทำให้การส่งข้อมูลระหว่างอุปกรณ์ในเครือข่ายนั้นอาจมีการแบ่งข้อมูลออกเป็นแพ็กเก็ตย่อยๆ ในระหว่างการส่ง เรียกว่า การทำแฟร็กเมนเตชัน (Fragmentation) เช่น แพ็กเก็ตของ FDDI มีขนาด 4,500

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ไบต์ หากเครื่องปลายอยู่ในเครือข่าย Ethernet ซึ่งมีขนาดของแพ็กเก็ตสูงสุดเพียง 1,500 ไบต์ ดังนั้นการส่งแพ็กเก็ตไปยังเครื่องปลายทางจึงต้องมีการแบ่งเป็นแพ็กเก็ตย่อย และเมื่อแพ็กเก็ตย่อยมาถึงเครื่องเป้าหมายก็จะมารวมกันเป็นแพ็กเก็ตเดิมที่มีขนาด 4,500 ไบต์อีกครั้ง เรียกการรวมกันนี้ว่า การรีแอสเซมเบิ้ล (Reassemble) ซึ่งทำให้ได้ข้อมูลเหมือนที่ส่งมาจากเครื่องต้นทาง



รูปที่ 2-18 แสดงแพ็กเก็ตไอพี

<u>ส่วนประกอบของแพ็กเก็ตไอพี</u>

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

0 = Normal Delay - มีดีเลย์ปกติ

1 = Low Delay - มีดีเลย์ต่ำ

Bit 4 : บอกถึงประเภทของทรูพุต

0 = Normal Throughput – มีทรูพุตปกติ

1= High Throughput – มีทรูพุตสูง

เอกสารนี้เป็นเอกสารที่สงวนสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

46169

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 บอกออปเซตของแฟร็กเมนต์เมื่อเทียบในดาต้าแกรม

8. **Time To Live (TTL)** : เป็นตัวเลข 8 บิต บอกช่วงเวลาของแพ็กเก็ตที่ยังอยู่ในเครือข่ายได้ โดยกำหนดค่าเป็นจำนวนเราท์เตอร์สูงสุดที่ดาต้าแกรมผ่านได้ 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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ไอพี

3.1 โพรโตคอลไอพี

หน้าที่หลักของโพรโตคอลไอพีคือจัดขนาดข้อมูลให้พอเหมาะและเลือกเส้นทางที่เหมาะสมเพื่อจัดส่งดาต้าแกรม ไอพีมีรูปแบบการจัดส่งดาต้าแกรมเป็นแบบ “unreliable” และ “connectionless”



รูปที่ 3-1 ฟอร์แมตของดาต้าแกรม

ความหมายของ “unreliable” คือไอพีไม่มีกลไกรับประกันว่าดาต้าแกรมที่ส่งไปถึงปลายทางได้สำเร็จหรือไม่ หากมีความผิดปกติเกิดขึ้นในระหว่างการส่งดาต้าแกรมไอพีทิ้งดาต้าแกรมนั้นไปแล้วรายงานสาเหตุของปัญหากลับ ไปด้วยโพรโตคอลไอซีเอ็มพี

ความหมายของ ”connectionless” ไอพีไม่มีการกำหนดเส้นทางลำเลียงระหว่างต้นทางและปลายทาง ไอพีไม่เก็บสถานะใดๆของดาต้าแกรมที่ส่งออกไป ดาต้าแกรมแต่ละชิ้นจึงเป็นอิสระจากกัน โดยมีโอกาสไปถึงปลายทางโดยไม่เรียงลำดับ

3.2 การประมวลดาต้าแกรม

ไอพีดาต้าแกรมสร้างขึ้นโดยโฮสต์ต้นทาง เมื่อถูกลำเลียงส่งไปนอกเครือข่ายอาจต้องเดินทางผ่านเราเตอร์หลายตัวจนกระทั่งถึงปลายทาง เราเตอร์และโฮสต์จะดำเนินการกับดาต้าแกรมโดยกรรมวิธีเฉพาะตัว

3.2.1 การประมวลดาต้าแกรมที่เราเตอร์

เมื่อได้รับดาต้าแกรมเราเตอร์จะตรวจสอบความถูกต้องขอผลรวมตรวจสอบ จากนั้นทำการตรวจสอบเฮดเดอร์ตั้งแต่รุ่น ขนาดเฮดเดอร์ ขนาดดาต้าแกรมรวม และชนิดโพรโตคอลว่าผิดปกติหรือไม่ จากนั้นก็ลดค่าในฟิลด์ TTL ลงหนึ่งค่า เราเตอร์จะกำจัดดาต้าแกรมทิ้งถ้าหากตรวจพบปัญหาดังต่อไปนี้ ผลรวมตรวจสอบไม่ถูกต้อง พารามิเตอร์ในส่วนหัวที่ตรวจสอบผิดปกติ หรือ ค่า TTL มีค่าเป็น 0 รวมทั้งกรณีที่บัฟเฟอร์ในเราเตอร์มีขนาดไม่เพียงพอในจัดการกับดาต้าแกรมนั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถัดจากนั้นเราเตอร์ทำการตรวจสอบออปชันดังนี้คือ ออปชันกำหนดเส้นทาง ชนิดบริการ และการแบ่งแฟกเมนต์ ในขั้นนี้ดาต้าแกรมอาจถูกกำจัดทิ้งอีกหากการกำหนดเส้นทางในออปชัน ไม่สามารถดำเนินการได้เพราะไม่พบเราเตอร์ที่กำหนด หรือจำเป็นต้องแฟล็กแฟรกเมนต์ไม่ได้อนุญาติให้ทำ

ขั้นถัดไปเป็นการปรับค่าในฟิลด์ต่างๆได้แก่ค่า TTL ค่าในฟิลด์ออปชัน ค่าที่จำเป็นหากมีการแฟรกเมนต์ และท้ายสุดเราเตอร์จะคำนวณผลรวมการตรวจสอบเพื่อใส่ในฟิลด์ผลรวมตรวจสอบก่อนที่จะนำส่งต่อไป

3.2.2 การประมวณดาต้าแกรมที่โฮสต์ปลายทาง

เมื่อโฮสต์ปลายทางได้รับดาต้าแกรมแล้วเริ่มตรวจผลรวมตรวจสอบ จากนั้นทำการตรวจความถูกต้องของเฮดเดอร์ตั้งแต่ รุ่น ขนาดเฮดเดอร์ ขนาดดาต้าแกรม และชนิดของโพรโตคอลว่าผิดปกติหรือไม่ โฮสต์จะกำจัดดาต้าแกรมทิ้งหากพบว่าค่าใดค่าหนึ่งผิดปกติและแจ้งข้อผิดพลาดกลับไปด้วยไอซีเอ็มพี

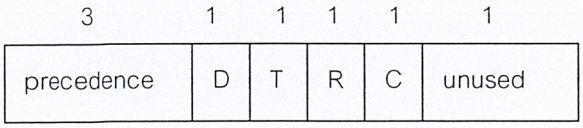
หากโฮสต์พบว่าเป็นดาต้าแกรมที่แฟรกเมนต์ โฮสต์ต้องตรวจสอบว่าเป็นแฟรกเมนต์ที่อยู่ในชุดเดียวกันโดยตรวจฟิลด์เลขประจำตัว แอดเดรสต้นทางและปลายทาง โพรโตคอล และตรวจแฟรกเมนต์ออปเซตว่าถูกต้องหรือไม่

โฮสต์ตั้งเวลารับดาต้าแกรมให้ครบชุดภายในเวลาในเวลาที่กำหนดตั้งแต่แฟรกเมนต์แรกเดินทางมาถึง แฟรกเมนต์ทั้งหมดถูกกำจัดทิ้งไปหากไม่สามารถรับทุกแฟรกเมนต์ในชุดนั้นภายในเวลาที่กำหนด

3.3 ชนิดการให้บริการ

ในรูปที่ 3-2 เป็นฟอร์แมตของฟิลด์ TOS ซึ่งแบ่งออกเป็นสามส่วน ส่วนแรกคือ precedence ขนาด 3 บิต, ฟิลด์กำหนดคุณภาพการให้บริการ 4 บิต และฟิลด์สุดท้ายขนาด 1 บิตซึ่งไม่ได้ใช้งานและมีค่าเป็น 0

precedence ขนาด 3 บิตใช้กำหนดลำดับความสำคัญของดาต้าแกรมซึ่งมีได้ 8 ระดับจาก 0 (ต่ำสุด) ถึง 7 (สูงสุด)
เราเตอร์ตรวจค่านี้เพื่อให้บริการกับดาต้าแกรมที่มีลำดับความสำคัญสูงกว่าก่อน แต่ในปัจจุบันฟิลด์นี้มักไม่ได้ใช้งาน เราเตอร์ส่วนใหญ่ไม่จัดลำดับค่าความสำคัญของดาต้าแกรม



รูปที่ 3-2 ฟอร์แมตของฟิลด์ TOS

สี่บิตถัดจาก precedence ใช้กำหนดคุณภาพการให้บริการกับดาต้าแกรมตามชนิดของแอพลิเคชัน แต่ละบิตมีความหมายเฉพาะคือ D (minimize delay), T (maximize throughput), R (maximize reliability) และ C (minimize monetary cost) ดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บิต D มีค่า "1" หมายถึงต้องการเส้นทางที่มีเวลาหน่วง (delay time) ต่ำสุดเท่าที่เราเตอร์หาให้ได้

บิต T มีค่า "1" หมายถึงต้องการเส้นทางที่มีความสามารถส่งผ่านข้อมูลได้ในปริมาณมากๆ ในหนึ่งหน่วยเวลา เราเตอร์ต้องจัดเส้นทางที่ตรงกับความต้องการนี้

บิต R มีค่า "1" หมายถึงต้องการเส้นทางที่มีความเชื่อถือได้สูง หรือเส้นทางที่โอกาสทำให้สูญเสียดาต้าแกรมน้อย

บิต C มีค่า "1" หมายถึงต้องการเส้นทางที่มีค่าพารามิเตอร์ของ "cost" ต่ำ

ทั้ง 4 บิตถ้าใดมีค่า "1" แล้วบิตที่เหลือจะมีค่าเป็น "0" เพราะเราเตอร์ให้บริการหลายแบบพร้อมกันไม่ได้ และหากทั้ง 4 บิตมีค่าเป็น "0" หมายถึงไม่มีบริการพิเศษใด

3.4 การแบ่งขนาดข้อมูล

เมื่อไอพีได้รับข้อมูลจากโพรโตคอลระดับบนและเลือกอินเทอร์เฟซที่ส่งข้อมูลออก ก็ทำการตรวจเอ็มทียูประจำอินเทอร์เฟซ หากพบว่าดาต้าแกรมมีขนาดใหญ่เกินกว่าเอ็นทียู ไอพีจะแบ่งดาต้าแกรมออกเป็นชิ้นย่อยให้ดาต้าแกรมมีขนาดพอเหมาะกับเอ็นทียูวิธีนี้เรียกว่า แฟรกเมนต์ (fragmentation)

ดาต้าแกรมที่ถูกแฟรกเมนต์จะมีเฮดเดอร์ประจำตัวเอง โดยมีข้อมูลเพิ่มเติมกำหนดลักษณะของแฟรกเมนต์ประกอบไปด้วย แต่ละชิ้นของแฟรกเมนต์เป็นดาต้าแกรมที่สมบูรณ์ในตัวเอง เราเตอร์ระหว่างทางจะไม่รวมดาต้าแกรม (reassembly) แต่จะปล่อยให้เป็นหน้าที่ของเราเตอร์ปลายทาง การแฟรกเมนต์ดาต้าแกรมของไอพีใช้ฟิลด์ 3 ฟิลด์กำหนดข่าวสารเกี่ยวกับการแฟรกเมนต์ดังต่อไปนี้

identification ขนาด 16 บิต : หมายเลขประจำชิ้นดาต้าแกรม หากต้องมีการแฟรกเมนต์ก็ใช้เป็นค่าที่บอกว่าเป็นดาต้าแกรมชุดเดียวกัน สถานีที่สร้างดาต้าแกรมจะเพิ่มค่านี้ครั้งละหนึ่งให้แต่ละดาต้าแกรมแต่ละชุดที่ส่งออกไป

flag ขนาด 3 บิต : ใช้ควบคุมและแสดงรูปแบบแฟรกเมนต์ แบ่งออกเป็น 3 บิต

บิตแรก ไม่ได้ใช้งานและมีค่าเป็น 0 เสมอ

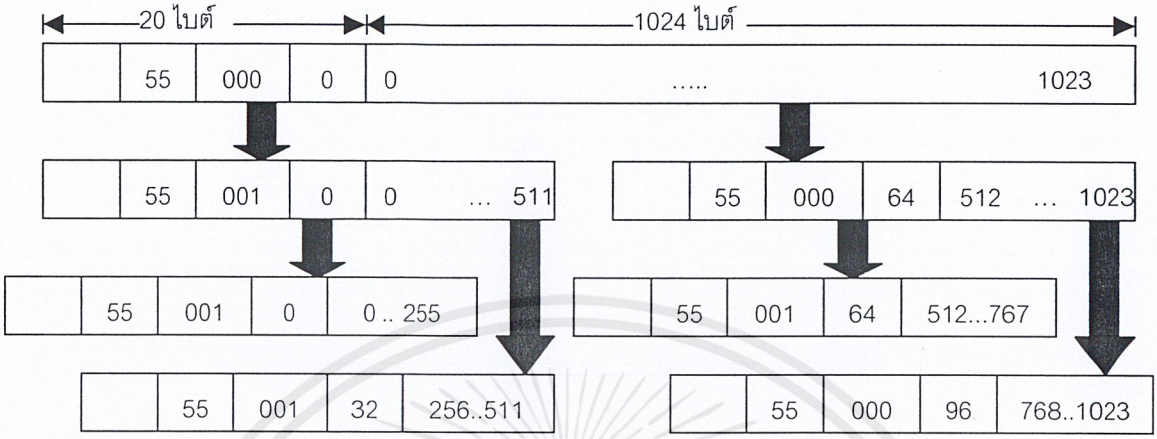
บิต D ใช้กำหนดว่าให้มีแฟรกเมนต์ได้หรือไม่ หากมีค่า "0" หมายถึงให้เราเตอร์แฟรกเมนต์ได้ตามความจำเป็น หากมีค่าเป็น "1" หมายถึงบังคับไม่ให้แฟรกเมนต์ ถ้าเราเตอร์ระหว่างทางจำเป็นต้องแฟรกเมนต์แต่ถูกบังคับไม่ให้แฟรกเมนต์ด้วยบิตนี้ เราเตอร์ทำการทิ้งดาต้าแกรมนั้นไปและแจ้งความผิดพลาดกลับไปยังต้นทาง

บิต M หากเป็น "0" หมายถึงเป็นดาต้าแกรมแฟรกเมนต์สุดท้าย หากเป็น "1" หมายถึงยังมีแฟรกเมนต์ตามมาอีก

flagment offset ขนาด 13 บิต : เราเตอร์ทำการแฟรกเมนต์ข้อมูลออกเป็นบล็อกแต่ละบล็อกต้องเป็นจำนวนเท่าของ 8 ไบต์ ฟิลด์นี้ใช้บอกตำแหน่งออฟเซ็ตของข้อมูลในแต่ละแฟรกเมนต์อ้างอิงข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.1 การแฟรกเมนต์



รูปที่ 3-3 การแฟรกเมนต์

รูปที่ 3-3 สาธิตการแฟรกเมนต์ดาต้าแกรมโดยสมมติให้ดาต้าแกรมขนาด 1044 ไบต์ ซึ่งประกอบด้วยเฮดเดอร์ 20 ไบต์ และข้อมูล 1024 ไบต์ ดาต้าแกรมนี้เดินผ่าน 2 เครือข่ายคือ N1 และ N2 ซึ่งมีเอ็มทียูเท่ากับ 532 และ 276 ตามลำดับ ค่าที่ใช้นี้เป็นเพียงค่าสมมติเพื่อให้เข้าใจการทำงานโดยง่าย

กำหนดให้ไม่มีการใช้ออปชัน ไอพีเฮดเดอร์จะมีขนาดคงที่ 20 ไบต์ เครือข่าย N1 มีเอ็นทียู 532 ไบต์จะแฟรกเมนต์ดาต้าแกรมหากข้อมูลมีขนาดเกิน 512 ไบต์ และเครือข่าย N2 มีเอ็นทียู 276 ไบต์ จะแฟรกเมนต์ดาต้าแกรมหากข้อมูลมีขนาดเกิน 256 ไบต์

เมื่อเดาต้าแกรมขนาด 1044 ไบต์ ถูกส่งผ่านเราเตอร์ R1 ไปยัง N1 เราเตอร์ R1 ทำการแบ่งดาต้าแกรมออกเป็น 2 แฟรกเมนต์ๆ ละ 532 ไบต์ คือไอพีเฮดเดอร์ 20 ไบต์ และข้อมูล 512 ไบต์ ในฟิลด์ identification มีค่าเท่ากับ 55 โดยที่บิต M ของแฟรกเมนต์แรกมีค่าเป็น “1” เพื่อบอกว่ามีแฟรกเมนต์อื่นตามมา และบิต M แฟรกเมนต์ที่สองมีค่าเป็น “0” เพื่อบอกว่าเป็นแฟรกเมนต์สุดท้าย ส่วนค่าออปเซ็ตของแฟรกเมนต์แรกมีค่าเป็น 0 และออปเซ็ตของแฟรกเมนต์ที่สองมีค่าเป็น 64 (มาจาก 512/8) เมื่อแต่ละแฟรกเมนต์ผ่าน R2 เข้าสู่เครือข่าย N2 แต่ละแฟรกเมนต์ถูกแบ่งออกเป็นอีก 2 แฟรกเมนต์ทำให้มีแฟรกเมนต์หลังจากผ่าน N2 เป็นจำนวน 4 แฟรกเมนต์ด้วยวิธีเดียวกัน

3.4.2 การรีแอสเซมบลี

สถานีปลายทางทำหน้าที่ รีแอสเซมบลี (reassembly) หรือรวมแต่ละแฟรกเมนต์เข้าด้วยกัน แฟรกเมนต์ที่อยู่ในดาต้าแกรมชุดเดียวกันต้องมีค่าต่อไปนี้ตรงกัน คือ identification, แอดเดรสต้นทาง, แอดเดรสปลายทาง และชนิดโพรโตคอล

ค่า total length ในแต่ละแฟรกเมนต์ของดาต้าแกรมบอกเพียงขนาดดาต้าแกรมนั้นหากแต่ละดาต้าแกรมมาถึงปลายทางโดยไม่เรียงลำดับ สถานีปลายทางไม่มีโอกาสทราบล่วงหน้าได้ว่าดาต้าแกรมทั้งชุดมีขนาดเท่าใด ด้วยเหตุนี้สถานีปลายทางจำเป็นต้องเตรียมบัฟเฟอร์เพื่อการรีแอสเซมบลีให้เพียงพอ

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้ทำซ้ำ ดัดแปลง เผยแพร่ หรือนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งไม่อนุญาตให้นำไฟล์นี้ไปเผยแพร่ต่อ ไม่ว่าจะเป็นการเผยแพร่ฟรีหรือเก็บค่าใช้จ่าย

3.4.3 ข้อคำนึงในการแฟรกเมนต์

การทำแฟรกเมนต์ไม่ได้เกิดเฉพาะกรณีที่ต้องส่งดาต้าแกรมข้ามเครือข่ายเท่านั้น แม้แต่ในเครือข่ายเดียวกันก็อาจจำเป็นต้องแฟรกเมนต์ดาต้าแกรมด้วยเช่น โพรโตคอลเอ็นเอฟเอสมักส่งข้อมูลครั้งละ 8,192 ไบต์ ซึ่งใหญ่กว่าเอ็นทียูในเครือข่ายแลนทั่วไป

หากสถานีปลายทางไม่สามารถรับทุกแฟรกเมนต์ได้ครบด้วยสาเหตุใดก็ตามต้องทิ้งดาต้าแกรมทั้งชุดไป ในกรณีที่ของเอ็นเอฟเอสนั้นสถานีต้นทางต้องส่งดาต้าแกรมทั้ง 8,192 ไบต์ที่ต้องทำการแฟรกเมนต์ซ้ำเช่นเดิมอีก ทำให้เพิ่มภาระในระบบเครือข่ายมากขึ้น ในเครือข่ายที่มีความผิดปกติและต้องมีการส่งแฟรกเมนต์ซ้ำจะสร้างภาระให้เครือข่ายสูงมาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
แผนภูมิสถานะทีซีพี

กระบวนการทำงานของทีซีพีซึ่งประกอบด้วยการเริ่มต้นการเชื่อมต่อ การขนถ่ายข้อมูล และการปิดการเชื่อมต่อ แต่ละขั้นตอนในกระบวนการเหล่านี้มีสถานะกำกับเพื่อกำหนดแบบแผนการทำงาน การเปลี่ยนจากสถานะหนึ่งไปสู่อีกสถานะหนึ่งขึ้นอยู่กับรูปแบบเซกเมนต์ที่ได้รับ ทีซีพีทั้งสองด้านต้องเก็บรักษาสถานะการเชื่อมโยงแต่ละส่วนไว้ตลอดเวลา

4.1 สถานะทีซีพี

โปรโตคอลชนิด "Connection oriented" อย่างเช่นทีซีพีมีสถานะกำกับการทำงานแต่ละขั้นตอน สถานะของทีซีพีช่วยให้โปรโตคอลมีแบบแผนการทำงานที่ชัดเจนและสามารถรองรับการทำงานได้หลายรูปแบบ เพื่อความเข้าใจเกี่ยวกับสถานะเหล่านี้ขอให้พิจารณาถึงแผนภูมิในรูปที่ 4-1 ประกอบคำอธิบาย แผนภูมินี้แสดงเฉพาะการเปลี่ยนแปลงสถานะตามเหตุการณ์ปกติที่เกิดขึ้นทั้งด้านทีซีพีไคลเอ็นต์และเซอร์ฟเวอร์โดยสรุปรวมไว้ในรูปเดียวกัน

4.1.1 การเริ่มต้นการเชื่อมต่อ

หากแยกพิจารณาสถานะทีซีพีไคลเอ็นต์และเซอร์ฟเวอร์ออกจากกันตามขั้นตอนการเริ่มต้นการเชื่อมต่อเพื่อจัดหมวดหมู่ให้เข้าใจง่าย สถานะของเซอร์ฟเวอร์และไคลเอ็นต์เมื่อเริ่มต้นการเชื่อมต่อ เป็นดังนี้

4.1.1.1 สถานะเซอร์ฟเวอร์ขั้นตอนการเริ่มต้นการเชื่อมต่อ

CLOSED สถานะจำลองที่สร้างขึ้นเพื่ออ้างอิง ในสถานะนี้ไม่มีการจัดสรรหน่วยความจำเพื่อเก็บค่าใด

LISTEN เซอร์ฟเวอร์รอคอยการขอบริการจากไคลเอ็นต์ หรือเรียกว่า เซอร์ฟเวอร์เปิดการเชื่อมต่อแบบพาสซีฟ

SYN_RECEIVED เซอร์ฟเวอร์ได้รับ SYN และส่ง SYN/ACK และรอคำตอบจากไคลเอ็นต์(รอ ACK)

ESTABLISHED เซอร์ฟเวอร์ได้รับ ACK การสถาปนาด้านเซอร์ฟเวอร์เสร็จสมบูรณ์และพร้อมโอนถ่ายข้อมูล

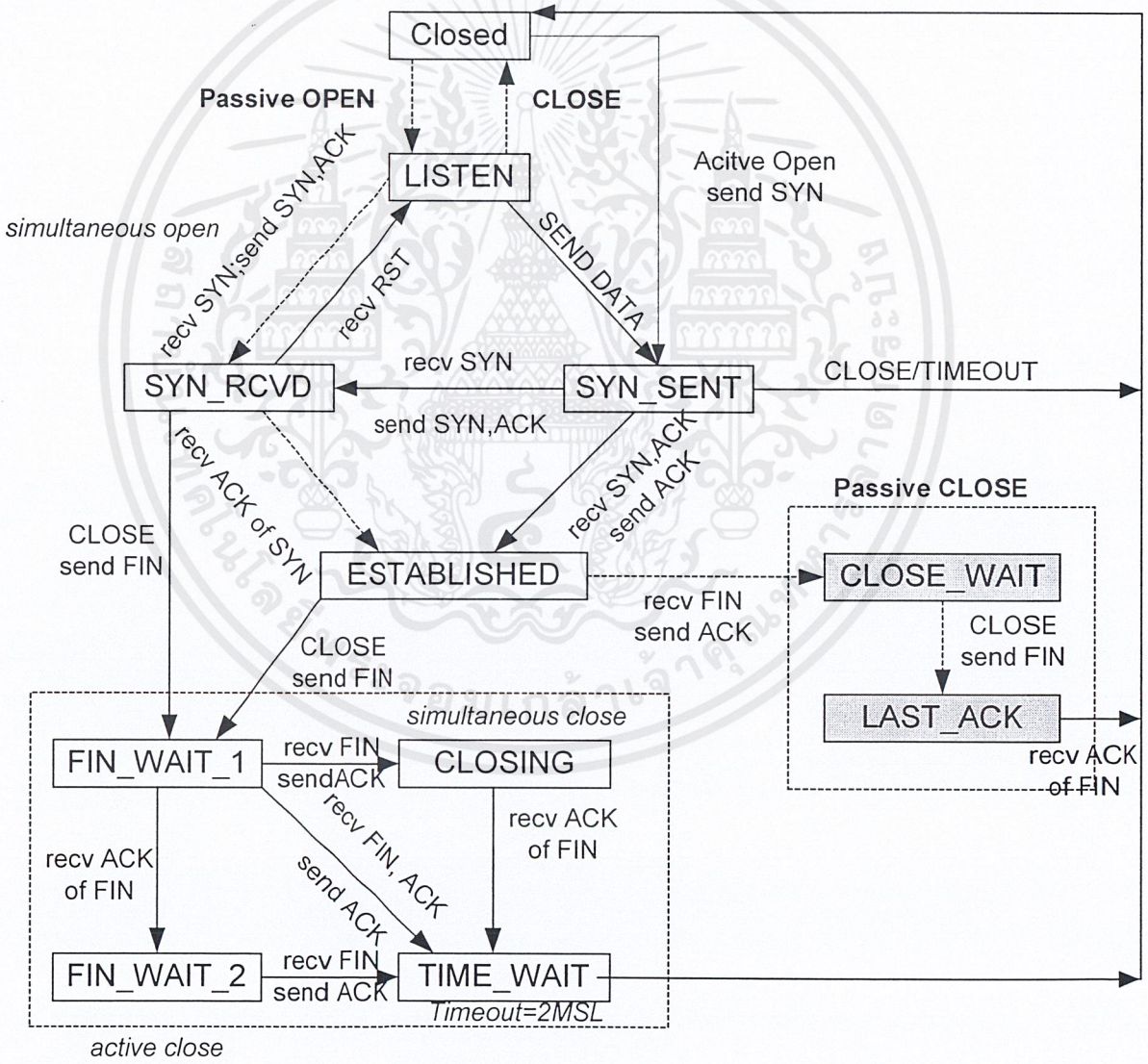
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.1.2 สถานะไคลเอ็นต์ขั้นตอนการเริ่มต้นการเชื่อมต่อ

CLOSED สถานะจำลองที่สร้างขึ้นเพื่ออ้างอิง ในสถานะนี้ไม่มีการจัดสรรหน่วยความจำเพื่อเก็บค่าใด

SYN_SENT ไคลเอ็นต์ส่ง SYN และรอคอยการจับคู่เชื่อมต่อ (ตอบรับ) โดยเรียกว่าไคลเอ็นต์ขอเปิดการเชื่อมต่อแบบแอกทีฟ

ESTABLISHED ไคลเอ็นต์ได้รับ SYN/ACK จากเซอร์ฟเวอร์ และตอบกลับไปด้วย ACK การเริ่มต้นการเชื่อมต่อด้านไคลเอ็นต์เสร็จสมบูรณ์และพร้อมถ่ายโอนข้อมูล



รูปที่ 4-1 แผนภูมิสถานะทีซีพี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.1.2 การปิดการเชื่อมต่อ

เมื่อเริ่มต้นการเชื่อมต่อเสร็จสิ้นแล้ว ทั้งไคลเอ็นต์และเซอร์ฟเวอร์จะอยู่ในสถานะ ESTABLISHED จนกระทั่งฝ่ายหนึ่งไม่มีข้อมูลส่งและขอปิดการเชื่อมต่อโดยส่งเซกเมนต์ FIN การปิดการเชื่อมต่อแต่ละฝ่ายจะผ่านสถานะต่อไปนี้เป็นลำดับ

4.1.2.1 สถานะฝ่ายที่ขอปิดการเชื่อมต่อ

FIN_WAIT_1 ส่ง FIN และเข้าสู่สถานะรอคอยเพื่อรอการตอบยืนยันว่าได้รับคำสั่งปิดการเชื่อมต่อ ในขั้นนี้อาจยังมีข้อมูลส่งมาจากอีกฝ่ายหนึ่งได้

FIN_WAIT_2 ฝ่ายขอปิดได้รับ ACK ยืนยันว่าอีกฝ่ายได้รับเซกเมนต์ขอปิดแล้ว (แต่อีกฝ่ายอาจไม่พร้อมที่จะปิดเพราะยังมีข้อมูลที่ต้องส่งอยู่)

CLOSING ฝ่ายขอปิดได้รับเซกเมนต์ FIN และจะส่ง ACK ตอบกลับ

TIME_WAIT สถานะรอคอยเพื่อรอสู่การปิดการเชื่อมต่อ

CLOSED ปิดการเชื่อมต่อแล้ว ข้อมูลการเชื่อมต่อถูกกำจัดทิ้งหมด

4.1.2.2 สถานะฝ่ายได้รับคำร้องขอปิดการเชื่อมต่อ

CLOSED_WAIT ได้รับ FIN ขอปิดการเชื่อมต่อ ทีซีพีส่ง ACK หากโปรแกรมประยุกต์ยังมีข้อมูลต้องนำส่งอยู่ ทีซีพีจะรอจนกว่าโปรแกรมประยุกต์สั่งปิด

LAST_ACK โปรแกรมประยุกต์สั่งปิดการส่งข้อมูล ทีซีพีส่ง FIN และคอยการตอบรับหากได้รับ ACK เมื่อใดก็จะเข้าสูสถานะ CLOSED

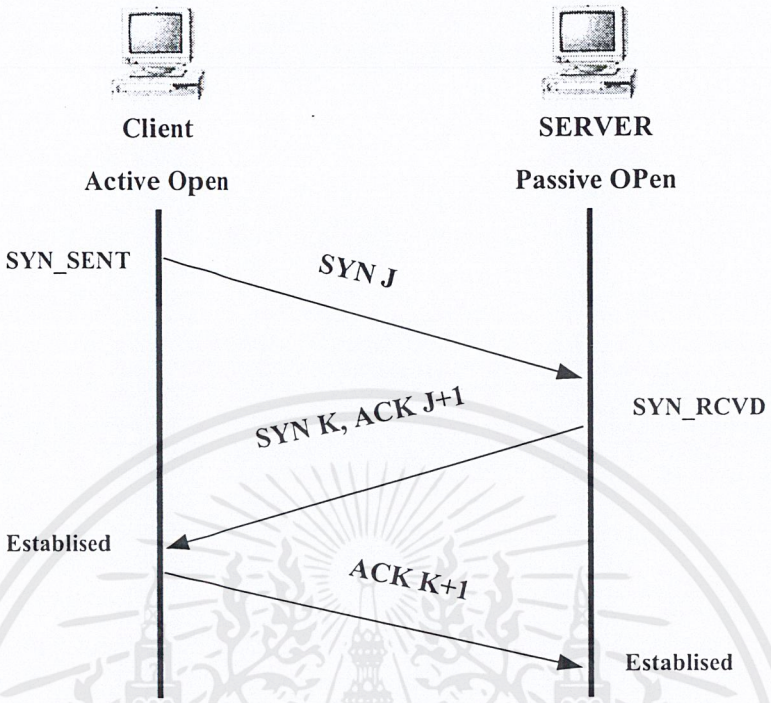
CLOSED ปิดการเชื่อมต่อแล้ว ข้อมูลการเชื่อมโยงถูกกำจัดทิ้งหมด

4.1.3 การรับส่งเซกเมนต์ในขั้นตอนการเริ่มต้นการเชื่อมต่อ

รูปที่ 4-2 แสดงขั้นตอนเริ่มต้นการเชื่อมต่อระหว่างไคลเอ็นต์และเซอร์ฟเวอร์คู่หนึ่ง โดยสมมติให้ไคลเอ็นต์เปิดการเชื่อมต่อแบบแอคทีฟและเซอร์เวอร์เปิดการเชื่อมต่อแบบพาสซีฟ

ไคลเอ็นต์เริ่มต้นจากสถานะ CLOSED และเข้าสู่สถานะ SYN_SENT หลังจากส่งเซกเมนต์ SYN เซอร์ฟเวอร์เมื่อได้รับเซกเมนต์ SYN ก็จะตอบกลับด้วยเซกเมนต์ SYN/ACK และเปลี่ยนจากสถานะ LISTEN เข้าสู่สถานะ SYN_RCVD ไคลเอ็นต์เมื่อได้รับเซกเมนต์นี้จะส่งเซกเมนต์ ACK กลับไปให้เซอร์ฟเวอร์และเข้าสู่สถานะ ESTABLISHED เมื่อเซอร์ฟเวอร์ได้รับเซกเมนต์นี้ก็จะเข้าสู่สถานะ ESTABLISHED

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-2 สถานะการเริ่มต้นการเชื่อมทีซีพี

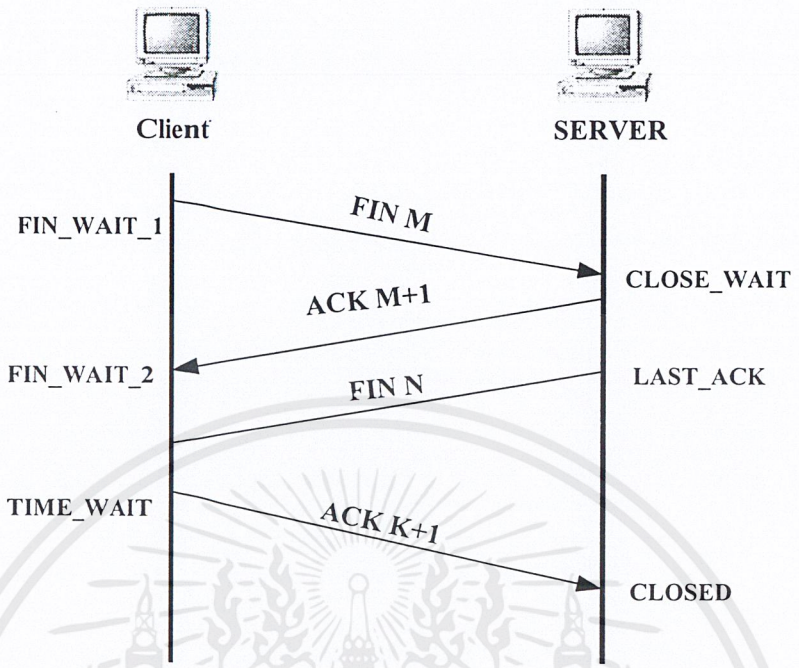
4.1.4 การรับส่งเซกเมนต์ในขั้นตอนการปิดการเชื่อมต่อ

หากย้อนไปพิจารณารูปที่ 4-1 จะพบว่าไคลเอ็นต์และเซอร์ฟเวอร์เป็นฝ่ายเริ่มปิดการเชื่อมต่อได้ทั้งสิ้น ในรูปที่ 4-3 จะแสดงเฉพาะกรณีไคลเอ็นต์เป็นผู้ร้องขอการปิดการเชื่อมต่อดังนั้นไคลเอ็นต์จะปิดแบบแอกทีฟและเซอร์ฟเวอร์จะปิดแบบพาสซีฟ

ขั้นตอนเริ่มต้นเมื่อไคลเอ็นต์ไม่มีข้อมูลจัดส่งอีกต่อไปแล้วก็แจ้งให้เซอร์ฟเวอร์ทรายโดยส่งเซกเมนต์ FIN และเข้าสู่สถานะ FIN_WAIT_1 เมื่อเซอร์ฟเวอร์ได้รับก็จะตอบกลับด้วยเซกเมนต์ ACK และเข้าสู่สถานะ CLOSE_WAIT เมื่อไคลเอ็นต์ได้รับเซกเมนต์นี้จะเปลี่ยนจากสถานะ FIN_WAIT_1 ไปสู่ FIN_WAIT_2

ในจังหวะนี้เซอร์ฟเวอร์อาจยังมีข้อมูลที่นำส่งไปยังไคลเอ็นต์ได้อยู่ เมื่อเซอร์ฟเวอร์ส่งข้อมูลหมดสิ้นแล้วก็ส่งเซกเมนต์ FIN แจ้งให้ไคลเอ็นต์ทราบและเปลี่ยนสถานะไปเป็น LAST_ACK ด้านไคลเอ็นต์เมื่อได้รับเซกเมนต์ FIN ก็รับทราบว่าเซอร์ฟเวอร์พร้อมปิดตัวเองแล้ว ไคลเอ็นต์ตอบกลับด้วยเซกเมนต์ ACK และเข้าสู่สถานะ TIME_WAIT เพื่อปิดตัวเองภายในเวลา 240 วินาที ในขณะที่เซอร์ฟเวอร์ปิดการเชื่อมต่อโดยเข้าสู่สถานะ CLOSED เมื่อได้รับเซกเมนต์ ACK นี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-3 สถานะการปิดการเชื่อมต่อ

สถานะ TIME_WAIT เป็นสถานะรอคอยเพื่อเข้าสู่การปิดการเชื่อมต่อ ซ็อกเก็ตประจำการเชื่อมต่อนั้นไม่สามารถใช้ได้จนกว่าจะพ้นช่วงเวลานี้ไป หากมีเซกเมนต์ใดๆที่เพิ่งมาถึงในช่วงเวลานี้ (เนื่องจากอาจถ่วงเวลาอยู่) และมีซ็อกเก็ตที่อยู่ในสถานะรอคอย เซกเมนต์นั้นจะถูกกำจัดทิ้ง ระยะเวลาในสถานะนี้มีค่าเป็นสองเท่าของค่าช่วงชีวิตนานที่สุดของเซกเมนต์จึงมักเรียกสถานะนี้ว่า "2MSL Wait state" ค่าช่วงชีวิตนานที่สุดของเซกเมนต์ที่กำหนดใน RFC 793 คือ 120 นาที ดังนั้น 2MSL จึงมีค่า 240 วินาที (ในบางระบบปฏิบัติการอาจใช้ค่า MSL เพียง 30 วินาที หรือ 1 นาที)

4.2 รูปแบบการเปิดและปิดการเชื่อมต่ออื่น

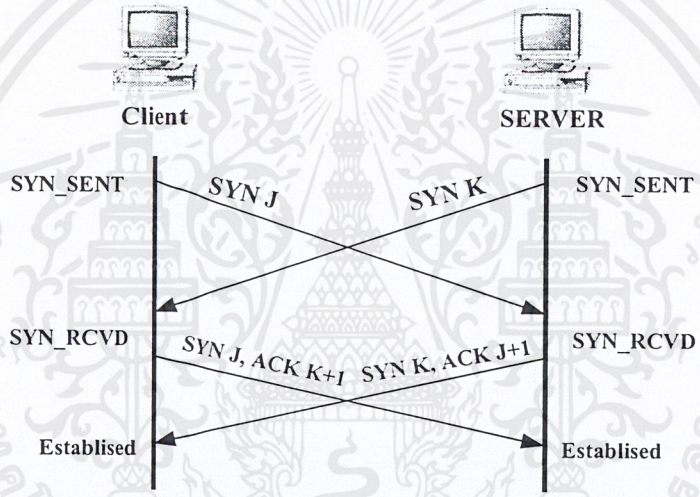
การเปิดและปิดการเชื่อมต่อที่อธิบายก่อนหน้านี้เป็นรูปแบบที่เกิดขึ้นโดยปกติระหว่างไคลเอ็นต์และเซอร์ฟเวอร์ แต่ไคลเอ็นต์และเซอร์ฟเวอร์อาจเข้าสู่สถานการณ์บางรูปแบบตามรูปที่ 4-1 คือ การเปิดพร้อมกัน (Simulations open) และ การปิดพร้อมกัน (Simulations close) นอกจากนี้ยังมีลักษณะการเชื่อมโยงอีก 2 แบบคือ การเปิดครึ่งฝ่าย (half-open) และ การปิดครึ่งฝ่าย (half-close)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.1 การเปิดพร้อมกัน

การเปิดพร้อมกันเกิดขึ้นในกรณีที่สถานีคู่หนึ่งขอเปิดการเชื่อมต่อแบบแอกทีฟพร้อมกันในขณะเวลาเดียวกัน แต่ละฝ่ายส่งเซกเมนต์ SYN โดยกำหนดพอร์ตต้นทางและปลายทางที่สมนัยกัน ตัวอย่างเช่น สถานี A มีพอร์ตต้นทางเท่ากับ 1000 ขอเปิดไปยังพอร์ต 2000 ของสถานี B ในขณะเดียวกันที่สถานี B ใช้พอร์ตต้นทาง 2000 ขอเปิดพอร์ต 1000 ของสถานี A

การเปิดพร้อมกันมีเซกเมนต์แลกเปลี่ยนจำนวน 4 เซกเมนต์ดังรูปที่ 4-4 โดยไม่มีการกำหนดว่าฝ่ายใดเป็นไคลเอ็นต์หรือเซอร์ฟเวอร์ทั้งสองด้านเปิดการเชื่อมต่อแบบพาสซีฟระหว่างกันและมีการเชื่อมโยงทางทีซีพีสองส่วน ในขณะที่การเปิดพร้อมกันจะมีการเชื่อมโยงทางทีซีพีส่วนเดียว

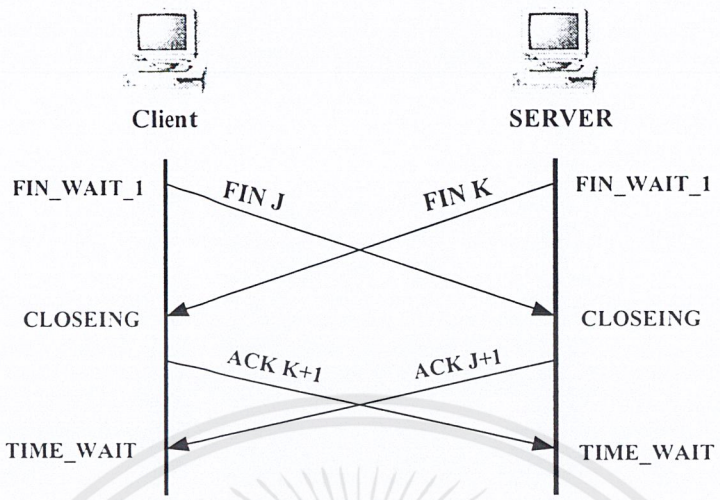


รูปที่ 4-4 การเปิดพร้อมกัน

4.2.2 การปิดพร้อมกัน

การปิดการเชื่อมต่อตามปกติเริ่มต้นจากฝ่านหนึ่งซึ่งอาจเป็นไคลเอ็นต์หรือเซอร์ฟเวอร์ขอปิดการเชื่อมต่อแบบแอกทีฟโดยส่งเซกเมนต์ FIN ไปยังอีกฝ่ายหนึ่ง แต่หากทั้งสองฝ่ายต่างขอปิดแบบแอกทีฟพร้อมกันก็จเรียกว่าการปิดพร้อมกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4-5 การปิดพร้อมกัน

รูปที่ 4-5 แสดงการปิดพร้อมกัน เริ่มจากทั้งสองฝ่ายอยู่ในสถานะ ESTABLISHED และต่างส่งเซกเมนต์ FIN ให้อีกฝ่ายหนึ่ง และเข้าสู่สถานะ FIN_WAIT_1 เช่นกัน จำนวนเซกเมนต์ที่ใช้ในการปิดพร้อมกันยังคงเท่ากับการปิดตามปกติ แต่ลำดับการโต้ตอบเซกเมนต์จะแตกต่างกัน

4.2.3 การเปิดครึ่งฝ่าย (half open)

หากทีซีพีฝ่ายใดฝ่ายหนึ่งหยุดการทำงานโดยอีกฝ่ายหนึ่งไม่ทราบ ทีซีพีอีกฝ่ายที่ยังทำงานแบบเปิดครึ่งฝ่าย (half-open) โดยปกติแล้วกรณีเช่นนี้มักเกิดจากโฮสต์ฝ่ายหนึ่งเกิดปัญหากระทั่งต้องปิดเครื่องในระหว่างนั้นหากไม่มีหารส่งข้อมูลใด อีกฝ่ายหนึ่งจะยังคงอยู่ในสถานะ ESTABLISHED และรอคอยการรับส่งข้อมูลอยู่ ตัวอย่างเช่นไคลเอ็นต์ขอใช้บริการเทลเน็ตจากเซอร์ฟเวอร์ หากเซอร์ฟเวอร์หยุดการทำงานและต้องรีบูตเครื่องใหม่ สถานภาพที่เชื่อมโยงกับไคลเอ็นต์จะสูญหายไปหมดจากการรีบูตโดยไคลเอ็นต์ไม่ทราบ ไคลเอ็นต์จึงเปิดการทำงานเพียงครึ่งเดียว

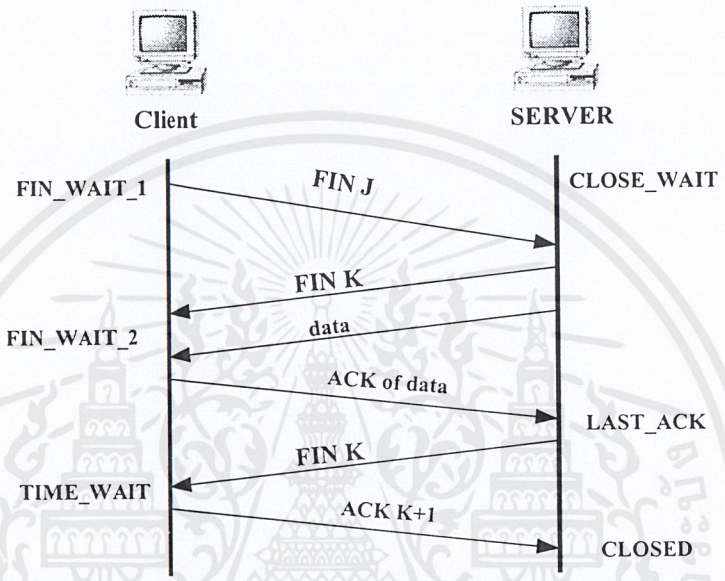
เมื่อไคลเอ็นต์ส่งเซกเมนต์ไปให้เซอร์ฟเวอร์ เซกเมนต์นั้นจะไม่มีซ็อกเก็ตใดรองรับกล่าวอีกนัยหนึ่งคือเซอร์ฟเวอร์ไม่สามารถแยกแยะได้ว่าเซกเมนต์นั้นเป็นของการเชื่อมโยงใด สิ่งที่เซอร์ฟเวอร์ดำเนินการก็คือส่งเซกเมนต์รีเซ็ต โดยเซ็ตแฟล็ก RST เพื่อแจ้งให้ไคลเอ็นต์ยกเลิกการเชื่อมต่อนั้น

4.2.4 การปิดครึ่งฝ่าย (half close)

เนื่องจากการเชื่อมโยงทีซีพีเป็นแบบฟูลดูเพล็กซ์ ข้อมูลสามารถเดินทางได้ระหว่างไคลเอ็นท์และเซอร์ฟเวอร์ทั้งสองทิศทางอย่างเป็นอิสระ แต่ละฝ่ายจึงสามารถปิดตัวเองลงเมื่อไม่มีข้อมูลใดต้องส่งอีกต่อไป โดยยังสามารถรับข้อมูลจากอีกฝ่ายหนึ่งได้ การเชื่อมโยงลักษณะนี้เรียกว่า ปิดครึ่งฝ่าย (half-close)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่างในรูปที่ 4-6 ไคลเอ็นต์ขอปิดการเชื่อมต่อโดยส่งเซกเมนต์ FIN และเข้าสู่สถานะ FIN_WAIT_1 เมื่อเซอร์ฟเวอร์ได้รับเซกเมนต์ FIN และตอบรับ (เซกเมนต์ ACK J+1) แล้วก็จะเข้าสู่สถานะ CLOSE_WAIT และเมื่อไคลเอ็นต์ได้รับเซกเมนต์ตอบรับก็จะเข้าสู่สถานะ FIN_WAIT_2 และเตรียมปิดการเชื่อมต่อ ในจังหวะนี้แอพลิเคชั่นด้านเซอร์ฟเวอร์อาจยังมีข้อมูลที่ต้องส่งจึงยังไม่ปิดการเชื่อมต่อ ไคลเอ็นต์ปิดการส่งแต่ยังคงต้องรับข้อมูลจากเซอร์ฟเวอร์อยู่



รูปที่ 4-6 การปิดครึ่งเดียว (half close)

ขั้นตอนถัดไปตามรูปที่ 4-6 สมมติว่าเซอร์ฟเวอร์มีข้อมูลต้องส่งเพียงเซกเมนต์เดียว (เซกเมนต์ data) เมื่อเซอร์ฟเวอร์ได้รับเซกเมนต์ตอบรับ (เซกเมนต์ ACK of data) ก็สั่งปิดการเชื่อมโยงโดยส่งเซกเมนต์ FIN และเข้าสู่สถานะ LAST_ACK ถัดจากนั้นการปิดการเชื่อมโยงก็จะดำเนินต่อไปตามปกติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
แพ็กเก็ตแคปเจอร์
Packet Capture

5.1 ความหมายแพ็กเก็ตแคปเจอร์ (Packet Sniffer)

คำว่า สนิฟเฟอร์ (Sniffer) นั้นเครื่องหมายการค้าซึ่งจดทะเบียนไว้โดยบริษัท Network Associates Inc. ในสหรัฐฯ เพื่อใช้ในผลิตภัณฑ์ของตนเองชื่อ Sniffer Network Analyzer ซึ่งเป็นโปรแกรมวิเคราะห์เน็ตเวิร์กโดยอาศัยการดักจับข้อมูลภายในเน็ตเวิร์กทั้งหมดมาวิเคราะห์การใช้งานระบบเน็ตเวิร์ก ต่อมาคำนี้ถูกนำมาใช้เรียกอุปกรณ์ประเภทนี้ที่นำมาใช้ในการดักอ่านข้อมูลกันอย่างแพร่หลาย จนเป็นที่เข้าใจว่าสนิฟเฟอร์เป็นเครื่องมือในการดักจับข้อมูลบนเน็ตเวิร์ก ซึ่งถ้าหากเรียกให้ถูกต้องอุปกรณ์ประเภทนี้ควรเรียกว่า Net wire Tapping Device

แพ็กเก็ตแคปเจอร์ คืออุปกรณ์ที่ทำหน้าที่ดักจับแพ็กเก็ตที่วิ่งอยู่บนเน็ตเวิร์ก โดยจะทำงานร่วมกับเน็ตเวิร์กอินเทอร์เฟซการ์ด (Network Interface card) ในโหมดโพรมิสคูอัส (Promiscuous-mode) โดยทำงานร่วมกันในโหมดนี้ การทำงานของเน็ตเวิร์กอินเทอร์เฟซการ์ดในโหมดนี้จะไม่มีการเปรียบเทียบแอดเดรสปลายทางของแพ็กเก็ตกับแอดเดรสของเน้ตเวิร์กอินเทอร์เฟซการ์ด นั่นคือทุกๆแพ็กเก็ตมาถึงยังเน็ตเวิร์กอินเทอร์เฟซการ์ดจะถูกจับแพ็กเก็ตขึ้นมาได้ทั้งหมดไม่เลือกว่าจะระบุแอดเดรสปลายทางของการ์ดเน็ตเวิร์กอินเทอร์เฟซอันไหน

แพ็กเก็ตแคปเจอร์จะถูกใช้ในการเฝ้าดูการจารจรในเครือข่ายให้มีประสิทธิภาพและแก้ไขปัญญาต่างที่เกิดจากการทำงานผิดพลาดของเครือข่ายและยังเป็นเครื่องมือในการศึกษาการทำงานของโพรโตคอลร่วมถึงการดูแลความปลอดภัยของระบบ

5.2 องค์ประกอบของแพ็กเก็ตแคปเจอร์

5.2.1 ฮาร์ดแวร์ (Hardware) หมายถึงอุปกรณ์อิเล็กทรอนิคส์ที่สามารถดักอ่านสัญญาณจากเน็ตเวิร์กเขามาได้ และสามารถนำสัญญาณที่ให้ส่งต่อไปประมาลผลออกมาเป็นข้อมูลทางคอมพิวเตอร์ได้ หน้าที่หลักคือจัดการกับการรับข้อมูลในระดับฟิสิคัล (Physical) อุปกรณ์นี้โดยทั่วไปก็คือเน็ตเวิร์กอะแดปเตอร์

5.2.2 ไดรเวอร์ (Diver) เป็นโปรแกรมระดับล่างที่ควบคุมการดักจับข้อมูลของฮาร์ดแวร์ไปเก็บเป็นข้อมูลดิบรอการประมวลผลในลำดับถัดไป

5.2.3 บัฟเฟอร์ (Buffer) เป็นหน่วยความจำที่ใช้พักข้อมูลจากการดักมาได้ของ Diver โดยจะทำงานการจัดเก็บเพียงชั่วคราว และหมุนเวียนข้อมูลใหม่เข้ามาเสมอ กลไกลการนำข้อมูลจากไดรเวอร์มาเก็บในบัฟ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เฟอร์ จะเป็นตัวบอกสมรรถนะของการดักข้อมูลได้ความเร็วสูงสุดเท่าใด หากกระบวนการนำข้อมูลไปเก็บเป็นไปอย่างล่าช้า ย่อมทำให้แพ็กเก็ตแคปเจอร์ไม่สามารถดักข้อมูลที่อยู่บนเน็ตเวิร์กได้ทันและต้องปล่อยข้อมูลนั้นทิ้งไป

5.2.4 ซอฟต์แวร์ (Software) เพื่อทำหน้าที่จัดการข้อมูลที่ได้เข้าโดยการประมวลผลตามวัตถุประสงค์ของการดักอ่านข้อมูลนั้น เนื่องจากข้อมูลดิบที่ดักได้นั้นจะเป็นข้อมูลในระดับต่ำ คือ เน็ตเวิร์กอินเทอร์เฟซเลเยอร์(Network Interface Layer) ซึ่งจะมีข้อมูลที่ยังไม่ผ่านการมัลติเพล็กซ์และจัดรูปแบบให้เข้าใจได้ ส่งที่ได้จะเป็นข้อมูลเลขฐานสอง 0 กับ 1 จำนวนมหาศาลที่ต้องนำมาแลความหมาย และอีกหน้าที่หนึ่งคือข้อมูลที่ดักได้นั้นเป็นการสื่อสารของทุกโฮสต์ที่อยู่ภายในเน็ตเวิร์กนั้นร่วมกันอยู่ อย่างไม่เป็นระเบียบและไม่มีการแยกแยะว่าเป็นการสื่อสารเรื่องอะไร ระหว่างโฮสต์ใดกับโฮสต์ใด การที่จะแปลความหมายของข้อมูลนี้ได้ก็จำเป็นต้องมีโปรแกรมสำหรับทำหน้าที่แปลความหมายของข้อมูลให้อยู่ในรู้แบบที่สามารถเข้าใจได้มากขึ้น นั่นคือทำหน้าที่คล้ายกับดีมัลติเพล็กซ์ของโพรโตคอลปกติ แต่จะเป็นการดีมัลติเพล็กซ์ของข้อมูลทุกๆโฮสต์โดยไม่สนใจว่าเป็นข้อมูลของโฮสต์ใด

หลังจากข้อมูลผ่านการดีมัลติเพล็กซ์แล้วก็จะอยู่ในรูปที่สามารถเข้าใจได้ง่ายขึ้น แต่จะเข้าใจมากน้อยขึ้นอยู่กับความสามารถในการดีมัลติเพล็กซ์ของโปรแกรม หากสามารถดีมัลติเพล็กซ์ได้จนถึงโพรโตคอลเลเยอร์ที่สูง รวมทั้งหากมีการแยกแยะหมวดหมู่ของการสื่อสารของแต่ละโฮสต์ก็จะเห็นความต่อเนื่องของการสื่อสารได้ดีขึ้น

ในส่วนของการจัดเก็บข้อมูลดิบที่ดักมาได้โปรแกรมแคปเจอร์ส่วนใหญ่จะทำการเก็บข้อมูลดิบที่ดักมาได้ไว้ในฐานข้อมูลเพื่อให้สามารถนำกลับมาวิเคราะห์ในภายหลังได้ เพราะในการของแคปเจอร์จะต้องคอยดักจับข้อมูลอยู่ตลอดเวลา หากทำการดีมัลติเพล็กซ์ทันทีที่ได้รับข้อมูลในแบบเรียลไทม์อาจทำให้ประสิทธิภาพของแคปเจอร์ลดต่ำลง และเนื่องจากการสื่อสารข้อมูลต้องอาศัยความต่อเนื่องของการเชื่อมโยงกันของหลายๆแพ็กเก็ต ประกอบการดีมัลติเพล็กซ์ส่วนใหญ่จะสามารถทำได้ก็ต่อเมื่อข้อมูลได้รับมาครบถ้วนสมบูรณ์ทั้งหมดแล้วเท่านั้น

5.3 การทำงานของแพ็กเก็ตแคปเจอร์

การทำงานของแพ็กเก็ตแคปเจอร์อาศัยหลักการของโพรโตคอลอีเทอร์เน็ตที่ใช้การกระจายของข้อมูลไปยังทุกโฮสต์ที่อยู่ในเน็ตเวิร์กและอาศัยโฮสต์แต่ละตัวทำหน้าที่จำแนกการสื่อสารของตนเอง นั่นหมายความว่าข้อมูลข้อมูลทุกแพ็กเก็ตที่ใช้สื่อสารกันได้ถูกส่งไปยังทุกโฮสต์ ซึ่งจะได้รับพร้อมกันแล้วเหมือนกัน เพียงแต่การที่จะสื่อสารกันได้อย่างถูกต้องนั้นโฮสต์แต่ละตัวจะต้องมีกระบวนการที่สามารถรู้ได้ว่าข้อมูลแพ็กเก็ตใดเป็นของตนเอง และข้อมูลแพ็กเก็ตใดมิใช่ของตนเอง ทุกๆ แพ็กเก็ตที่กระจายลงบนเน็ตเวิร์กนั้นจะมีหมายเลขระบุชัดเจนคือ แม็คแอดเดรส (MAC Address) หรือเรียกอีกอย่างหนึ่งว่า อีเทอร์เน็ต (Ethernet Address) ซึ่งจะเป็นสิ่งที่บอกว่าแพ็กเกตมาจากฮาร์ดแวร์ใดในเน็ตเวิร์ก ทำให้สามารถระบุได้ว่าส่งมาจากโฮสต์ใด

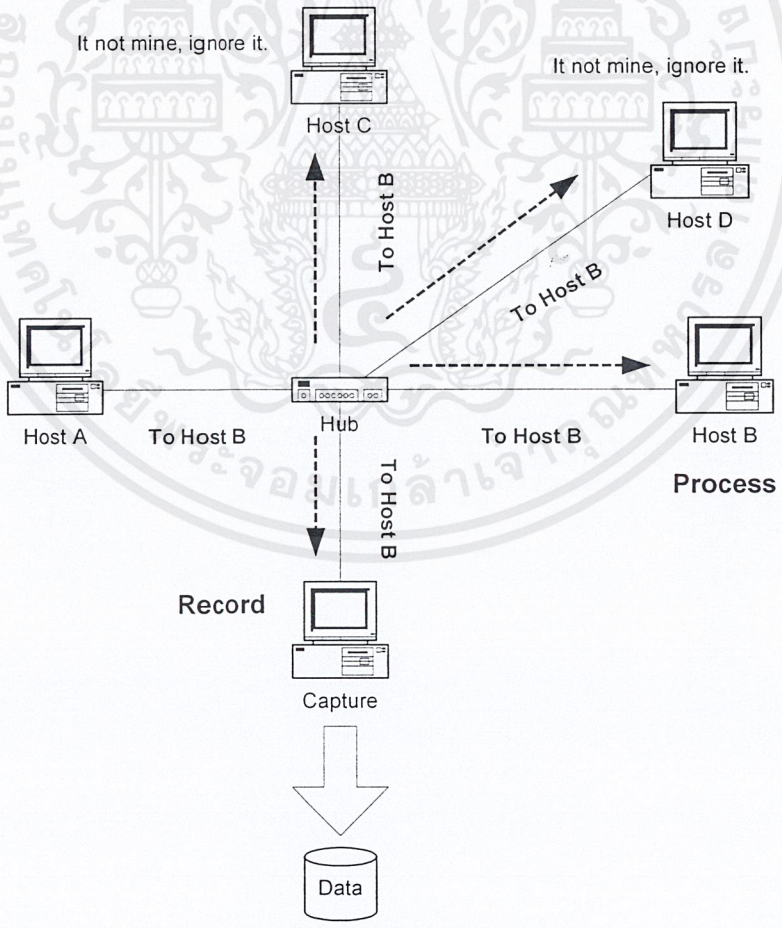
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แม็คแอดเดรสจะเป็นหมายเลขเฉพาะฮาร์ดแวร์ทุกชนิดที่ใช้สื่อสารโดยโพรโตคอลอีเธอร์เน็ต และในทางทฤษฎีแล้วฮาร์ดแวร์ทุกชนิดจะไม่มีแม็คแอดเดรสซ้ำกัน โดยทั่วไปแม็คแอดเดรสจะกำหนดตายตัวอยู่ในรอม (ROM) ของฮาร์ดแวร์และไม่สามารถเปลี่ยนแปลงได้ด้วยซอฟต์แวร์

การใช้งานของฮาร์ดแวร์จะต้องควบคู่กับไดรเวอร์ของฮาร์ดแวร์นั้นๆ โดยปกติแล้วไดรเวอร์จะถูกกำหนดให้ปฏิบัติตามโพรโตคอลอย่างเข้มงวดคือ

- ให้รับข้อมูลที่มีแม็คแอดเดรสเป็นของตนเองเท่านั้น (ห้ามอ่านข้อมูลของผู้อื่น)
- ให้ส่งข้อมูลโดยใช้แม็คแอดเดรสของตนเองเท่านั้น (ห้ามปลอมเป็นผู้อื่น)

ดังนั้นหากใช้ไดรเวอร์ตามปกติที่มากับฮาร์ดแวร์แล้วเครื่องคอมพิวเตอร์ก็จะทำงานอยู่ในระเบียบเรียบร้อย และไม่สามารถวุ่นวายกับข้อมูลของผู้อื่นได้ แพ็กเก็ตแคปเจอร์จึงจำเป็นที่ต้องใช้โหมดการทำงานพิเศษที่อนุญาตให้รับข้อมูลที่แม็คแอดเดรสไม่ใช้ของตนได้โหมดนั้นเรียกว่า พรอมิสคิอัสโหมด (Promiscuous Mode) ที่ทำให้ฮาร์ดแวร์สามารถรับข้อมูลดิบทั้งหมดบนเน็ตเวิร์กเข้ามาได้ โดยไม่สนใจว่าข้อมูลทั้งหมดเป็นของใคร ส่งให้ใคร



รูปที่ 5-1 การทำงานของแพ็กเก็ตแคปเจอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 5-1 ในเน็ตเวิร์กมีโฮสต์อยู่ 4 ตัวต่อรวมกันเป็นเน็ตเวิร์กโดยใช้ฮับ (Hub)ร่วมกัน และมีโฮสต์ตัวหนึ่งรันโปรแกรมที่ทำหน้าที่เป็นโปรแกรมแคปเจอร์เพื่อดักจับข้อมูลเสียบต่อร่วมเข้ากับฮับนั้น จากรูปแสดงให้เห็นการกระจายของข้อมูลที่โฮสต์ A ต้องการส่งให้โฮสต์ B ข้อมูลได้กระจายไปยังทุกโฮสต์ที่ต่ออยู่บนฮับ รวมถึงแพ็กเก็ตแคปเจอร์ด้วย

โฮสต์ B เมื่อได้รับข้อมูลก็จะตรวจแม็คแอดเดรสถ้าพบว่าเป็นข้อมูลที่ถูกส่งมาหาตนเองก็จะทำการอ่านและส่งต่อให้ระบบปฏิบัติการประมวลผล

โฮสต์ C และ *โฮสต์ D* โฮสต์ทั้งสองซึ่งมีเน็ตเวิร์กอะแคปเตอร์ทำงานอยู่ในโหมดปกติ เมื่อได้ทำการตรวจสอบแล้วเห็นว่าไม่ใช่ข้อมูลของตนเอง ก็จะไม่สนใจข้อมูลนั้น

แพ็กเก็ตแคปเจอร์ สำหรับแพ็กเก็ตแคปเจอร์นั้นมีเน็ตเวิร์กอะแคปเตอร์ที่ทำงานอยู่ในโพมิสคูอัสโหมดนั่นคือไม่สนใจว่าข้อมูลที่เข้ามาเป็นของใคร แพ็กเก็ตแคปเจอร์จะรับข้อมูลทั้งหมดแล้วนำไปเก็บในบัฟเฟอร์และฐานข้อมูล เพื่อนำไปประมวลผลเพื่อหาข้อมูลที่มีประโยชน์ภายหลัง

จากเน็ตเวิร์กในภาพนั้นมีแพ็กเก็ตแคปเจอร์ถูกติดตั้งในตำแหน่งดังกล่าวก็จะทำให้ข้อมูลทั้งหมดที่โฮสต์ทั้ง 4สื่อสารกันโดยผ่านฮับที่ใช้งานร่วมกัน ไม่ว่าข้อมูลใดจะปรากฏที่ฮับก็สามารถถูกดักอ่านได้โดยแพ็กเก็ตแคปเจอร์ทันที

5.4 ประโยชน์จากแพ็กเก็ตแคปเจอร์

5.4.1 วิเคราะห์ระบบเครือข่าย (Network Analyzer) การที่แพ็กเก็ตแคปเจอร์สามารถดักข้อมูลทั้งหมดที่อยู่บนเน็ตเวิร์กได้ ทำให้เราสามารถนำข้อมูลดังกล่าวมาทำการประมวลผลวิเคราะห์คุณสมบัติต่างๆของเน็ตเวิร์กได้หลายแง่มุม ไม่ว่าการกระจายการใช้งานเน็ตเวิร์กของโพรโตคอลต่างๆ เช่น เอชทีทีพี (HTTP), เอสเอ็มทีพี (SMTP), เอฟทีพี (FTP) เพื่อจะได้ทราบว่ามีการใช้เน็ตเวิร์กเพื่อการใด มากน้อยเพียงใด จะทำให้สามารถจัดการเน็ตเวิร์กได้อย่างเหมาะสม หรือในด้านการใช้งานเน็ตเวิร์กตามช่วงเวลาต่างๆเวลาใดที่มีผู้ใช้มาก เวลาใดที่มีผู้ใช้น้อย ผู้ใช้คนใดใช้แบนด์วิดธ์ไปในทางที่เป็นประโยชน์หรือไม่

5.4.2 เครื่องมือตรวจสอบเครือข่าย (Network Debugging Tools) ในบางครั้งการหาสาเหตุความผิดปกติของแอพพลิเคชั่นต่างๆ จำเป็นที่ต้องสืบค้นลึกลงไปถึงเนื้อข้อมูลที่รับส่งกันจริงๆบนเน็ตเวิร์กเพื่อหาว่าเหตุใดการทำงานของแอพพลิเคชั่นจึงไม่สามารถทำงานได้ตามปกติ การได้เห็นข้อมูลที่ส่งกันจริงทำให้การวิเคราะห์และแก้ไขปัญหาสามารถดำเนินการอย่างรวดเร็วและตรงต่อปัญหาได้มากขึ้น โดยเฉพาะกรณีที่มีการใช้เครื่องมือในระดับเน็ตเวิร์กมาเกี่ยวข้องเช่น มีการส่งข้อมูลผ่านไฟร์วอลล์แล้วปัญหา หรือการทดสอบ ALC (Access Control List) ของเราเตอร์ เป็นต้น

5.4.3 ตรวจจับแพ็กเก็ต (Packet Monitoring) การศึกษาเทคนิคของโพรโตคอลในระดับเน็ตเวิร์กจำเป็นต้องเห็นข้อมูลที่สื่อสารกันอยู่จึงจะสามารถเข้าใจได้และเห็นภาพจริง แพ็กเก็ตมอนิเตอร์ริ่งจะเป็นเครื่องมือที่นำแพ็กเก็ตมาเสดงเห็นในรูปแบบต่างๆได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.4.4 ระบบตรวจจับผู้บุกรุก (Intrusion Detection System) ระบบตรวจจับผู้บุกรุกก็มีพื้นฐานมาจากการดักอ่านข้อมูล เพียงแต่เป็นการประยุกต์การวิเคราะห์รูปแบบการบุกรุกเข้ากับการดักอ่านข้อมูลบนเน็ตเวิร์กแบบเรียลไทม์ ทำให้สามารถทราบว่ามีกิจกรรมใดบนเน็ตเวิร์กที่มีแนวโน้มว่าจะเป็นการบุกรุกหรือทำอันตรายต่อผู้อื่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6
สเตตฟูลไฟร์วอล
Stateful Firewall

6.1 หลักการทำงานสเตตฟูลไฟร์วอล

การทำงานของสเตตฟูลอินสเป็คชั่น จะมีการติดตามสถานะ(State) การทำงานของการเชื่อมต่อแบบ TCP ซึ่งเป็นผลให้สามารถดูรูปแบบการทำงานได้ทั้งกระบวนการ ไม่ได้ดูเพียงข้อมูลในแต่ละแพ็คเก็ต โดยสามารถดูลักษณะการเชื่อมต่อ การโต้ตอบของแต่ละโพรโตคอลที่มีลักษณะที่แตกต่างกัน โดยสามารถแยกแยะโพรโตคอลที่ถูกต้องกับโพรโตคอลที่ไม่ถูกต้องออกจากกันได้

โดยทั่วไปมักจะแยกกันโดยชัดเจนระหว่างไฟร์วอลล์แบบ statefull filtering และแบบ application-level proxy server ในบทนี้เราจะไม่กล่าวถึงหลักการทำงานของ แอพพลิเคชันพร็อกซี (Application Proxies) ซึ่งอยู่นอกขอบเขตที่ศึกษา

6.2 การแบ่งชนิดของไฟร์วอล

6.2.1 พ็กเก็ตฟิลเตอร์ริง (Packet filtering)

ไฟล์วอลล์ชนิดนี้จะกรองแพ็กเก็ตโดยพิจารณาแพ็กเก็ตที่เข้าออกตามกฎที่ตั้งไว้ การตัดสินใจกรองแพ็กเก็ตเหล่านี้จะยึดข้อมูลที่อยู่ในเฮดเดอร์ของแพ็กเก็ตตัวนั้นๆ เช่น แอดเดรสต้นทาง แอดเดรสปลายทาง พอร์ตหรือโพรโตคอล ผลิตภัณฑ์ไฟร์วอลล์บางตัวที่สามารถกรองแพ็กเก็ต ยังมีความสามารถในการกรองข้อมูลในส่วนอื่นที่ไม่ใช่แพ็กเก็ตเฮดเดอร์ด้วย เมื่อแพ็กเก็ตแต่ละแพ็กเก็ตเข้ามาสู่ไฟร์วอลล์ จะนำมาเทียบกับกฎ หากเข้ากับกฎใดกฎหนึ่ง จะดูว่ากฎนั้นสั่งให้ส่งแพ็กเก็ตนั้นต่อไป หรือดร็อปแพ็กเก็ตนั้นทิ้งไป และหากไม่เข้ากับกฎใดเลย ก็จะพิจารณาว่าค่าเริ่มต้นเป็นอะไรให้ส่ง หรือดร็อปแพ็กเก็ตทิ้งไป

A

Action	Out host	Port	Their host	port	Comment
Block	*	*	SPIGOT	*	We don't trust these people
Allow	OUR_GW	25	*	*	Connection to our port

B

Action	Out host	Port	Their host	port	Comment
Block	*	*	*	*	Default

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

C

Action	Out host	Port	Their host	port	Comment
Allow	*	*	*	25	Connection to their SMTP port

D

Action	Out host	Port	Their host	port	Comment
Allow	{our host}	*	*	25	Our packets to their SMTP port
Allow	*	25	*	*	Their replies

E

Action	Out host	Port	Their host	port	Comment
Allow	{our host}	*	*	*	Our outgoing calls
Block	*	*	SPIGOT	*	Replies to our calls
Allow	OUR_GW	25	*	*	Traffic to nonservers

ตารางที่6-1ตัวอย่างของกฎที่ตั้งไว้สำหรับไฟร์วอลล์แบบ Packet Filtering

ตารางข้างบนนี้ แสดงตัวอย่างของกฎที่ตั้งไว้สำหรับไฟร์วอลล์แบบแพ็กเก็ตฟิลเตอริง (Packet Filtering) โดยการนำกฎมาใช้จะทำจากข้างบนลงมาข้างล่าง โดยในตาราง A นั้นมีการสร้างบล็อกเครื่องที่ชื่อ SPIGOT โดยจะปล่อย SMTP แต่จะต้องไปยังเครื่อง OUP-GW เท่านั้น ในตาราง B เป็นค่า Default ซึ่งจะถูกพิจารณาสุดท้าย ตาราง C จะยอมให้ข้อมูลที่มีพอร์ตปลายทางเป็น SMTP ผ่านไปได้ ซึ่งกรณีนี้ยังมีจุดอ่อน คือ สามารถโจมตีผ่านพอร์ต 25 ได้ ตาราง D จะยอมให้เฉพาะเครื่องที่อยู่ในลิสต์ที่มี SYN เท่านั้น ซึ่งจะทำให้ไม่สามารถโจมตีที่พอร์ต 25 ได้ ตาราง E จะยอมให้เฉพาะเครื่องที่อยู่ในลิสต์เท่านั้นที่ออกได้ และจะรับแพ็กเก็ตที่มี SYN หรือมีพอร์ตมากกว่า 1024 เท่านั้น

การตัดสินใจในการกรองแพ็กเก็ต โดยยึดข้อมูลที่อยู่ในแพ็กเก็ตเฮดเดอร์ (packet header) ของแพ็กเก็ตตัวนั้นๆ เช่น แอดเดรสต้นทาง (Source Address) แอดเดรสปลายทาง (Destination Address) พอร์ต (Port) โพรโตคอล (Protocol) ผลิตภัณฑ์ไฟร์วอลล์บางตัวที่สามารถกรองแพ็กเก็ต ยังมีความสามารถในการกรองข้อมูลในส่วนอื่นที่ไม่ใช่แพ็กเก็ตเฮดเดอร์ แต่เราจะพิจารณาส่วนนี้ในหัวข้อของ stateful inspection packer filter

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.2.2 แอพพลิเคชันพร็อกซี (Application Proxies)

แอพพลิเคชันพร็อกซีเป็นโปรแกรมแอพพลิเคชันที่รันอยู่บนระบบไฟร์วอลระหว่างสองเครือข่าย โดยเครื่องที่รันพร็อกซีไม่ต้องทำตัวเป็นเราเตอร์ เมื่อโปรแกรมจากเครื่องไคลเอ็นต์เริ่มการติดต่อผ่านพร็อกซีไปยังเครื่องที่ให้บริการปลายทาง เริ่มแรกต้องทำการติดต่อกับโปรแกรมพร็อกซีเซิร์ฟเวอร์ก่อน โดยตัวไคลเอ็นต์จะทำการเจรจากับพร็อกซีให้ทำการติดต่อกับเครื่องที่ให้บริการปลายทางให้ ถ้าเป็นผลสำเร็จจะเกิดการเชื่อมต่อ 2 การเชื่อมต่อคือระหว่างเครื่องไคลเอ็นต์กับเครื่องพร็อกซีเซิร์ฟเวอร์ และระหว่างเครื่องพร็อกซีกับเครื่องให้บริการปลายทาง เราจะเห็นได้ว่าพร็อกซีจะรับภาระของการจราจรสองทิศทางระหว่างเครื่องไคลเอ็นต์กับเครื่องพร็อกซีเซิร์ฟเวอร์ จะทำการสร้างการติดต่อทั้งหมดและทำหน้าที่ในการตัดสินใจในการทำฟอร์เวิร์ดแพ็กเก็ตด้วย โดยพร็อกซีนี้ไม่มีส่วนเกี่ยวข้องในส่วนของการหาเส้นทาง

ทำนองเดียวกับแพ็กเก็ตฟิลเตอริง ในแอพพลิเคชันพร็อกซีนั้นก็สามารถทำงานได้ในสองแพลตฟอร์ม คืออุปกรณ์ที่ทำหน้าที่เป็นพร็อกซีโดยเฉพาะ กับคอมพิวเตอร์ที่ใช้ในงานทั่วไป ซึ่งโดยทั่วไปแล้วแอพพลิเคชันพร็อกซีจะช้ากว่าเราเตอร์ที่ใช้กรองแพ็กเก็ต แต่ในขณะเดียวกันแอพพลิเคชันพร็อกซีมีความปลอดภัยสูงกว่าแพ็กเก็ตฟิลเตอร์ริง โดยที่จริงแพ็กเก็ตฟิลเตอริงสามารถทนรับการข้อบกพร่องจากการอิมพลีเมนต์หรือทดแทนในส่วนของการอิมพลีเมนต์การเราต์ของระบบปฏิบัติการ แต่เมื่อความสามารถของแพ็กเก็ตฟิลเตอริงถูกเพิ่มเข้าไปในการหาเส้นทาง มันก็ไม่สามารถที่จะชดเชยหรือทำให้ข้อบกพร่องในการหาเส้นทางถูกต้องได้

ผลจากการทำการกรองที่ซับซ้อนขึ้นและการตัดสินใจในการควบคุมการเข้าถึงแอพพลิเคชันพร็อกซีสามารถใช้ทรัพยากรคอมพิวเตอร์ที่สำคัญ และเครื่องราคาแพงที่ใช้ในการปฏิบัติงาน ตัวอย่างเช่น ถ้าไฟร์วอลที่ทำงานบนระบบยูนิกซ์ ต้องการที่จะให้สนับสนุนการทำงานจำนวน 200 HTTP session ที่ทำงานพร้อมๆ กัน เครื่องที่ใช้ต้องมีความสามารถในการสนับสนุนงานของ 200 HTTP proxy ด้วยเหตุผลในแง่ของประสิทธิภาพในการส่ง/รับข้อมูล และเมื่อเพิ่มอีก 100 FTP session, 25 SMTP session , LDAP session ส่วนหนึ่ง และ DNS transaction คุณจะต้องใช้เครื่องที่สนับสนุน 500-1000 proxy process บางพร็อกซีอาจจะอิมพลีเมนต์โดยใช้ kernel threads (เพื่อช่วยลดความต้องการของทรัพยากรลงอย่างรวดเร็ว) แต่ความต้องการในทรัพยากรก็ยังคงมากอยู่ดี

6.2.3 สเตตฟูลอินสเป็กชัน (Stateful Inspection) หรือ ไดนามิกแพ็กเก็ตฟิลเตอริง (Dynamic packet filtering)

เราใช้คำว่า “สเตตฟูลอินสเป็กชัน” หรือ “ไดนามิกแพ็กเก็ตฟิลเตอริง” หมายถึง เซตของฟังก์ชันที่ทำการกรองแพ็กเก็ตที่มีความสามารถมากขึ้นในเราเตอร์ โดยที่แพ็กเก็ตฟิลเตอริงถูกจำกัดอยู่ในการตัดสินใจโดยข้อมูลในการตัดสินใจจะอยู่ในส่วนของเฮดเดอร์เท่านั้น และจะพิจารณาเฉพาะแพ็กเก็ตตัวนั้นๆ เท่านั้น โดยไม่ได้คำนึงถึงแพ็กเก็ตก่อนหน้านี้ ส่วนสเตตฟูลอินสเป็กชันจะเอาชนะในข้อจำกัดนี้ของแพ็กเก็ตฟิลเตอริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทั้งในส่วนของข้อมูลที่ใช้ในการตัดสินใจ โดยจะดูถึงข้อมูลในส่วนของเมสเซจด้วยและจะคำนึงถึงในส่วนของแพ็กเก็ตอื่นๆ ที่อยู่ก่อนหน้านี้ประกอบในการตัดสินใจ ทำให้ความสามารถในการตัดสินใจมากขึ้น เพราะรู้ข้อมูลมากขึ้น ซึ่งสเตตฟูลอินสเป็กชันก็เหมือนกับแพ็กเก็ตฟิลเตอริง คือถูกอิมพลีเมนต์เพิ่มลงในเราเตอร์ ดังนั้นโฮสต์ที่มีฟังก์ชันของสเตตฟูลอินสเปคชันทำงานอยู่นั้นจะต้องเป็นหรือสามารถทำงานเสมือนเป็นเราเตอร์ได้ด้วย

เจตนาพื้นฐานของสเตตฟูลอินสเป็กชันนั้นเพื่อรวมข้อดีระหว่างประสิทธิภาพในแง่ของการส่งถ่ายข้อมูลและความปลอดภัย โดยการเพิ่มการทำงานลงในเราเตอร์ทำให้สเตตฟูลอินสเปคชันมีประสิทธิภาพในการส่งถ่ายข้อมูลดีกว่าพร็อกซี และยังมีการเพิ่มระดับของฟังก์ชันของไฟร์วอลให้ดีกว่าแพ็กเก็ตฟิลเตอริงธรรมดา และสเตตฟูลอินสเป็กชันยังมีส่วนคล้ายกับพร็อกซีคือ เกณฑ์ของการควบคุมการเข้าถึงข้อมูลที่ซับซ้อนสามารถจะระบุรายละเอียดได้ และเหมือนกับแพ็กเก็ตฟิลเตอริง คือสเตตฟูลอินสเป็กชันได้รับการสืบทอดมาจากการอิมพลีเมนต์การหาเส้นทางที่มีคุณภาพสูง

6.3 หลักการทำงานของ Simple Packet Filter

ชั้น IP layer นั้นเป็นชั้นที่ทั้งซิมเพิลแพ็กเก็ตฟิลเตอร์(simple packet filter) และ สเตตฟูลฟิลเตอร์ (stateful filter) ใช้เป็นจุดเริ่มต้นในการพิจารณาข้อมูล สิ่งที่ซิมเพิลแพ็กเก็ตฟิลเตอร์และสเตตฟูลไฟร์วอล ทำเหมือนกันก็คือ ตรวจสอบข้อมูล header ของไอพีแพ็กเก็ตโดยสิ่งที่ถูกตรวจสอบคือ source address ,destination address ,protocol ,options

นโยบายของไฟร์วอลล์ส่วนใหญ่มักจะตรวจสอบความถูกต้องของแอดเดรส (ทั้ง source address และ destination address) ด้วย ยกเว้นกรณีที่ระบุเป็น "all" และข้อมูลในส่วนของโพรโตคอลนั้นก็จะเป็นตัวบอกว่าส่วนของดาต้านั้นบรรจุข้อมูลชนิดใดอยู่ เช่น TCP, UDP หรือบางทีอาจจะเป็น IPSEC, ESP, ISAKMP กระบวนการในการตรวจสอบรูปแบบของ header นั้นจะอยู่ที่ layer ชั้นบนถัดไป และข้อมูลในส่วนออปชันนั้น โดยปกติมักจะไม่มีค่าอะไร แต่อาจจะมี flag ที่แสดงถึง source routing ได้ (source routing ใช้ในกรณีที่ผู้ส่งต้องการให้ผู้รับส่งข้อมูลผ่านเส้นทางที่ผู้ส่งได้กำหนดไว้) โดยปกติแล้วมักจะถูกใช้โดยผู้บุกรุก ดังนั้นซิมเพิลแพ็กเก็ตฟิลเตอร์ส่วนใหญ่จะดร็อปแพ็กเก็ตที่ถูกส่งมาพร้อมกับ flag ดังกล่าว โดยไม่มีการตรวจสอบข้อมูลอย่างอื่นเพิ่มเติม

TCP header บรรจุข้อมูล 3 ส่วนที่สำคัญใน transport layer ไว้คือ source port , destination port และ flags โดยในส่วนของ flag นั้น สามารถประกอบไปด้วย URG(urgent), ACK(acknowledgment), PSH(push), RST(reset), SYN(syncronize) และ FIN(finish) และยังมีส่วนที่สี่ที่บรรจุ sequence number ซึ่งผู้ผลิตบางค่ายนำมาใช้พิจารณาด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับ flag นั้นเป็นส่วนที่สำคัญมากในสเตตฟูลไฟร์วอลเช่น การทำ TCP 3-way handshake

Source			Destination
Sequence			
Acknowledment			
Offs	Reserv	Fla	Window
Checks			Urgent
Optio			Paddi

รูปที่ 6-1 TCP header

UDP header บรรจุข้อมูลเพียงแค่ source port และ destination port เท่านั้นเอง ไม่มีส่วนของ flag และ sequence number แต่อย่างใด เนื่องจาก UDP เป็น connectionless protocol อยู่แล้วนั่นเอง และใน ICMP header นั้น ไม่ได้บรรจุข้อมูล port แต่อย่างใด แต่บรรจุชนิดของ ICMP (ICMP message type) เช่น "Echo request" , "Destination unreachable"

Source	Destination
Length	Checksu
Data begin here	

รูปที่ 6-2 UDP header

Simple packet filter จะพิจารณา TCP แพ็กเก็ตเพื่อตรวจสอบว่า

- แพ็กเก็ตนั้น มี address ที่ถูกต้องหรือไม่
- แพ็กเก็ตนั้น ถูกสร้างมาจาก external address จริงหรือไม่
- protocol หรือ service นั้น ผ่านการตรวจสอบแล้วหรือไม่
- option หรือ flag ที่ส่งมานั้น ถูกต้องตามข้อกำหนดหรือไม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในกรณีที่การส่ง ACK แพ็กเก็ตดังกล่าวถูกส่งมาโดยเจตนาร้าย เพราะเมื่อเครื่องปลายทางได้รับ ACK มักจะส่งสัญญาณ RST กลับไป ซึ่งจะแสดงให้เห็นว่าเครื่องปลายทางนั้นยังเปิดอยู่ และเปิดให้บริการใน port ดังกล่าว(destination port ที่ส่งมาพร้อมกับ ACK packet) บางครั้งยังสามารถใช้ตรวจสอบระบบปฏิบัติการของเครื่องปลายทางได้อีกด้วย (OS fingerprint) เพราะระบบปฏิบัติการในแต่ละระบบนั้นมักจะมีวิธี response ที่แตกต่างกันออกไป

6.4 หลักการทำงานของ packet-filtering firewall (stateful)

สำหรับ packet-filtering firewall (stateful) แล้วก็ทำเช่นเดียวกันกับแพ็กเก็ตฟิลเตอร์ แต่จะบันทึกข้อมูลเกี่ยวกับการเชื่อมต่อ ที่เกิดขึ้นลงในตารางการเชื่อมต่อ ก่อนที่จะส่งแพ็กเก็ตนั้นไปยัง IP stack ตัวตารางนี้จะมีส่วนสำหรับบันทึกข้อมูลสำหรับแต่ละการเชื่อมต่อที่ถูกต้อง โดยปกติจะเก็บข้อมูล source and destination address, protocol, port, flag แต่มีไฟร์วอลล์บางยี่ห้อที่เก็บข้อมูล sequence number เพิ่มด้วย เพื่อใช้ในการตรวจสอบแพ็กเก็ตที่กำลังจะเข้ามาและป้องกันการทำ session hijacking

Src_IP	Src_Prt	Dst_IP	Dst_Prt	IP_prot	Type	Flags
191.168.7.131	10003	207.229.143.8	25	6	16385	02ffff00
191.168.7.131	10003	207.229.143.8	24	6	16385	02ffff00

ตารางที่ 6-2 ตัวอย่าง State table

เมื่อ packet-filtering firewall ได้รับแพ็กเก็ตมันจะตรวจสอบข้อมูลกับตารางการเชื่อมต่อว่าเป็นส่วนของการเชื่อมต่อที่สร้างไว้แล้วหรือไม่ โดยจะพิจารณาจากข้อมูล source adderss, destination address, source port, destnation port จะต้องสอดคล้องกับตารางการเชื่อมต่อ ซึ่งถ้าเป็นส่วนหนึ่งของการเชื่อมต่อจริงก็ไม่มีความจำเป็นใดๆ ที่ต้องตรวจสอบซ้ำอีก แต่ในไฟร์วอลล์บางยี่ห้อนั้นจะพิจารณา sequence number ของแพ็กเก็ตเพิ่มเติมด้วย เช่น CiscoPIX ในขณะที่ Checkpoint's Firewall-1 ไม่พิจารณา sequence number แต่อย่างใด และเนื่องจากมีการเก็บ history ไว้ในตารางการเชื่อมต่อ ดังนั้นสเตตฟูลไฟร์วอลสามารถดร็อปแพ็กเก็ตที่ผิดปกติได้ และถ้าแพ็กเก็ตที่ส่งมาเป็นส่วนหนึ่งของการเชื่อมต่อที่สร้างไว้แล้วก็สามารถส่งผ่านได้เลย ซึ่งทำให้ประหยัดเวลาในการตรวจสอบจากไฟร์วอลใหม่อีกรอบ

ถ้าแพ็กเก็ตที่ส่งมาไม่ตรงกับการเชื่อมต่อที่สร้างไว้แล้ว และไม่ใช่ SYN แพ็กเก็ตตัว แพ็กเก็ตนั้นๆ ก็จะถูกดร็อปทิ้งไป และแม้แต่แพ็กเก็ตที่ผสม flag แปลกๆ เช่น SYN/FIN (เป็นกระบวนการหนึ่งในการทำ port scanning) ก็จะถูกดร็อปทิ้งไปเช่นเดียวกัน ทั้งนี้ไฟร์วอลล์ส่วนใหญ่สามารถบันทึกล็อกไฟล์ได้ด้วย ซึ่งขึ้นอยู่กับการตั้งค่าของผู้ดูแลระบบเองว่าต้องการเก็บข้อมูลใด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

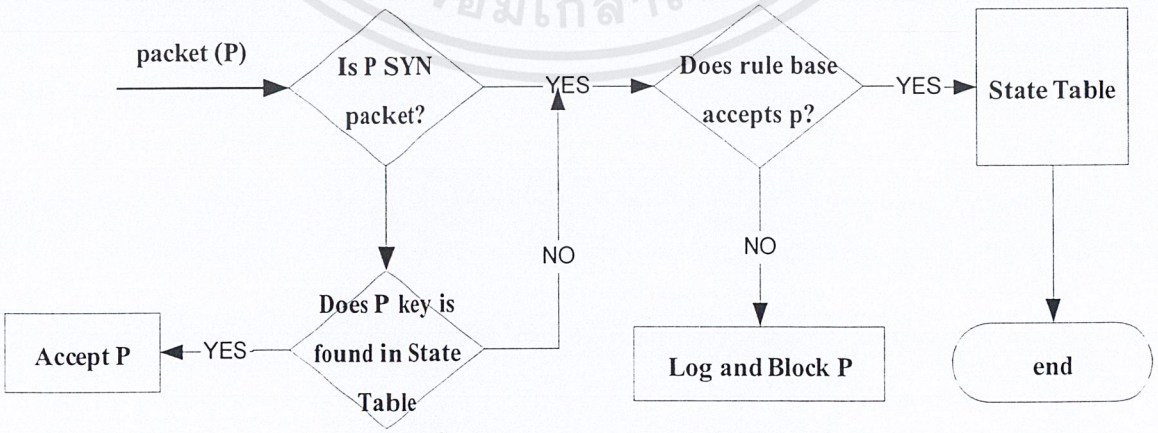
ในกรณีที่ packet-filtering firewall ได้รับ UDP แพ็กเก็ตเนื่องจากไม่มีกระบวนการ 3-way handshake เพราะมันเป็น stateless protocol ไม่มี sequence number แต่มันก็ยังมี source port และ destination port ซึ่งจะถูกสร้างขึ้นใน ตารางการเชื่อมต่อซึ่งรวมเอา source address , destination address มาไว้ด้วยกัน ทั้งนี้ เนื่องจาก UDP ไม่มี FIN หรือ RST ซึ่งใช้สำหรับยกเลิกการเชื่อมต่อเหมือนกับ TCP ดังนั้นมันจึงต้องมี timeout เพื่อลบข้อมูลออกจากตารางสถานะ

ในการลบข้อมูลออกในตารางการเชื่อมต่อโดยเฉพาะสำหรับ TCP connection ซึ่งอาจจะถูกโจมตีโดยการส่ง SYN แพ็กเก็ต(เรียกอีกอย่างว่า SYN flood) จำนวนมากเข้ามายังไฟร์วอลล์ ซึ่งก่อให้เกิด Denial of Service ได้ เพราะมันทำให้ ตารางการเชื่อมต่อเต็ม ปัญหานี้สามารถแก้ไขได้โดยการตั้ง timeout ของแต่ละการเชื่อมต่อไว้

6.5 FW-1 สเตตฟูลอินสเป็คชั่น

6.5.1 หลักการทำงาน

การทำงานที่เป็นสเตตฟูลอินสเป็คชั่นคือเมื่อไฟร์วอลได้รับ SYN แพ็กเก็ตที่เป็น แพ็กเก็ตเริ่มต้นของการเชื่อมต่อแบบ TCP/IP แล้ว SYN แพ็กเก็ตนั้นจะถูกนำไปตรวจสอบกับกฎของไฟร์วอลเหมือนกับ router คือSYN แพ็กเก็ตนี้จะถูกเปรียบเทียบกับกฎแบบเรียงลำดับ คือเริ่มตั้งแต่กฎข้อที่ 0 ไปเรื่อยๆ ถ้าไม่มีการยอมรับจากการตรวจสอบกับกฎทุกๆ ข้อแล้ว แพ็กเก็ตนั้นจะถูกปฏิเสธแล้วการเชื่อมต่อก็จะถูกทิ้ง แพ็กเก็ตนั้นถูกยอมรับ session นั้นจะถูกเก็บเข้าตารางการเชื่อมต่อของไฟร์วอลซึ่งอยู่ใน Kernel memory ทุกๆ แพ็กเก็ตที่ตามมาที่ไม่มี SYN จะถูกเปรียบเทียบกับตารางสเตตฟูลอินสเป็คชั่น ถ้ามี session นั้นอยู่ในตารางและ แพ็กเก็ตนั้นเป็นส่วนของ session นั้น แพ็กเก็ตนั้นจะถูกยอมรับด้วย ถ้าไม่ได้เป็นส่วนหนึ่งของ session จะถูกทิ้ง นี้เป็นวิธีการที่ไม่ต้องเปรียบเทียบทุกๆ single แพ็กเก็ตที่เขามากับกฎของไฟร์วอล แค่ SYN แพ็กเก็ตที่เป็นตัวเริ่มต้นการเชื่อมต่อเท่านั้นที่นำมาเปรียบเทียบกับกฎ TCP



รูปที่ 6-3 Firewall Packet Inspection Logic

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้ามีการเริ่มต้น session ด้วย ACK แพ็กเก็ตแล้ว ไฟร์วอลจะนำ ACK แพ็กเก็ตไปเปรียบเทียบตารางสถานะ ใน Kernel memory แต่จะไม่มี session นี้ในตารางสถานะเพราะว่าไม่มีเคยมี SYN แพ็กเก็ต แต่ ACK แพ็กเก็ตนั้นจะถูกยอมรับแล้วก็จะถูกเก็บไว้ในตารางสถานะเมื่อมี แพ็กเก็ตที่ไม่เป็นส่วนหนึ่งของตารางสถานะเข้ามา แพ็กเก็ตนั้นจะถูกนำไปตรวจกับกฎ โดยไม่คำนึงว่ามันเป็น SYN,ACK หรือ แพ็กเก็ตอะไรก็ตาม ถ้า session นั้นถูกต้องตามกฎแล้ว session นั้นจะถูกเก็บไว้ในตารางสถานะหลังจากนั้นทุกๆ แพ็กเก็ตตามมาของ session นี้จะถูกเปรียบกับตารางสถานะและถูกยอมรับ เพราะมี session นี้อยู่ในตารางสถานะแล้วจึงไม่ต้องนำไปเปรียบเทียบกับกฎ ข้างล่างแสดงตัวอย่างของตารางสถานะ

6.5.2 Stateful Inspection

- Filtering on the ACK bit

ความสามารถในการฟิลเตอร์ ACK bit นั้นสำคัญมากเพราะ ACK bit แสดงความแตกต่างระหว่างการเข้ามาและออกไปเชื่อมต่อ แพ็กเก็ตแรกที่เข้ามาเชื่อมต่อนั้นแสดงถึง แพ็กเก็ตที่เริ่มต้นการเชื่อมต่อ ต้องมีการทำให้ ACK bit นั้นว่างเปล่า และ แพ็กเก็ตอื่นๆในการเชื่อมต่อจะไม่สนใจทิศทางซึ่งมีการกำหนด ACK bit เรียบร้อยแล้ว

- Filering on SYN bit

เทคนิคที่ช่วยการป้องกัน denial of service attacks (DOS) คือการตรวจสอบการเชื่อมต่อและใช้ตารางสถานะเพื่อเก็บ แพ็กเก็ตที่ถูกต้องตามกฎที่พิจารณาว่าเป็น SYN แพ็กเก็ตหรือไม่ โดยการฟิลเตอร์ SYN bit โดยการตรวจสอบ SYN แพ็กเก็ต กับกฎ นี้เป็นเทคนิคที่ช่วยป้องกันการโจมตีจาก DOS

6.5.3 การทำงานของ Inspection Module

1. เมื่อ ไฟร์วอลได้รับ SYN แพ็กเก็ตมันจะทำการตรวจสอบกับกฎพื้นฐานของ ไฟร์วอล ถ้า แพ็กเก็ตไม่ถูกต้อง key จะถูก log และ แพ็กเก็ตนั้นจะหยุด แต่ถ้าถูกต้องตามกฎจะนำ key ไปค้นหามใน pending list ถ้าไม่เจอจะทำการหยุดและปฏิเสธ แพ็กเก็ตนั้น แต่ถ้าพบก็จะทำการเก็บค่า key เข้าไว้ใน connection pending list

พร้อมกับทำการเริ่มต้นนับเวลา และส่งแพ็กเก็ตไปยังปลายทาง

2. เมื่อ ไฟร์วอลได้รบ SYN/ACK แพ็กเก็ตมันจะตรวจสอบ key ว่าตรงกับ key ของ SYN แพ็กเก็ตใน pending list หรือไม่ ถ้าไม่เจอ key ของ แพ็กเก็ตจะถูก log และถูกหยุด แต่ถ้าเจอจะทำการหาในตารางสถานะอีกว่า key ถูกบันทึกหรือยัง ถ้ายังก็จะทำการบันทึกค่า key ลงในตารางสถานะพร้อมกับเริ่มต้นนับเวลา และส่งแพ็กเก็ตไปยังปลายทาง แต่ key ถูกบันทึกแล้วจะทำการ log และหยุด แพ็กเก็ตนั้น

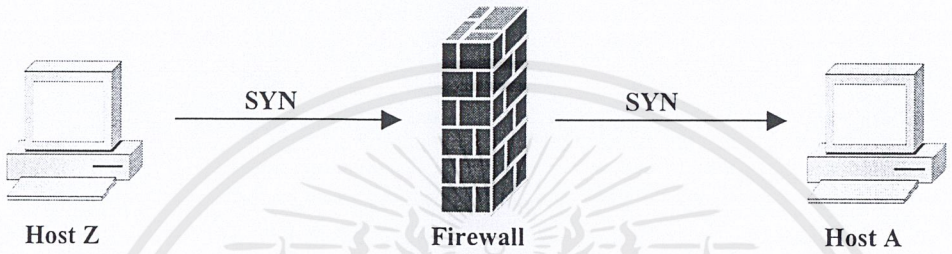
3.เมื่อ ไฟร์วอลได้รับ ACK แพ็กเก็ตแล้วจะตรวจสอบ key กับตารางสถานะถ้าเจอจะทำการส่ง แพ็กเก็ตต่อไปยังปลายทางและ key ที่อยู่ใน pending list จะถูกลบ แต่ถ้าไม่เจอ แพ็กเก็ตจะถูก log และหยุด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.5.4 วิเคราะห์ Inspection Mudule

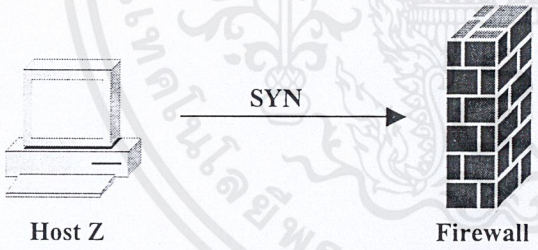
1. การใช้ฟิลเตอร์ SYN แพ็กเก็ตป้องกันการโจมตีด้วย SYN Flooding มีดั้งนี้

-ไฟร์วอลรับและยอมให้ SYN แพ็กเก็ตผ่านไปยัง host A และบันทึก key ลง ใน connection pending list



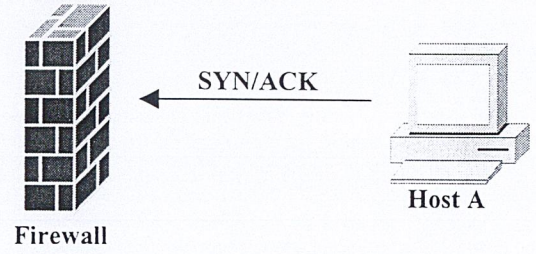
รูปที่ 6-4 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN packet (A)

- ถ้าไฟร์วอลรับ SYN แพ็กเก็ตต่อมาจาก Z ไป A ก่อนการเริ่มต้นการเชื่อมต่อ ไฟร์วอลจะทำการ block แพ็กเก็ตนี้ แต่ค่า key ยังจะคงถูกบันทึกลงใน pending list



รูปที่ 6-5 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN packet (B)

2..ใช้ฟิลเตอร์ SYNและ SYN/ACK แพ็กเก็ตป้องกันการโจมตี DOS



รูปที่ 6-6 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN และ SYN/ACK packet (A)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

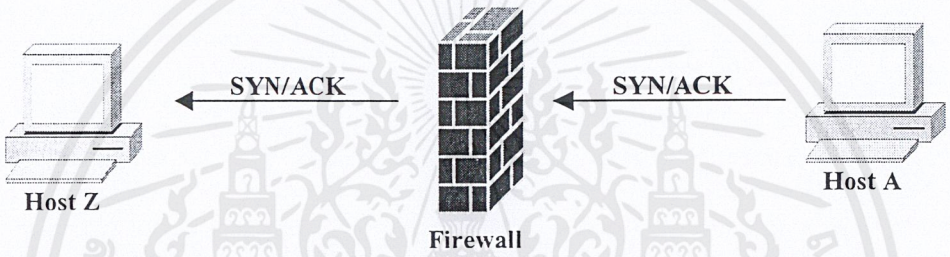
- ไฟร์วอลจะกั้นการตอบด้วย SYN/ACK แพ็กเก็ตของ A ไปยัง Z และ ดูว่า key นั้นตรงกับ SYN แพ็กเก็ตที่ส่งโดย Z ใน pending list หรือไม่

- ถ้าไม่เจอใน pending list จะถูก block และ log

- ถ้าเจอ key ของ ทำการคัดลอก key แล้วตรวจสอบกับตาราง state

- ถ้าตรวจเจอ key ในตาราง state, แพ็กเก็ตนั้นจะถูก block และ log

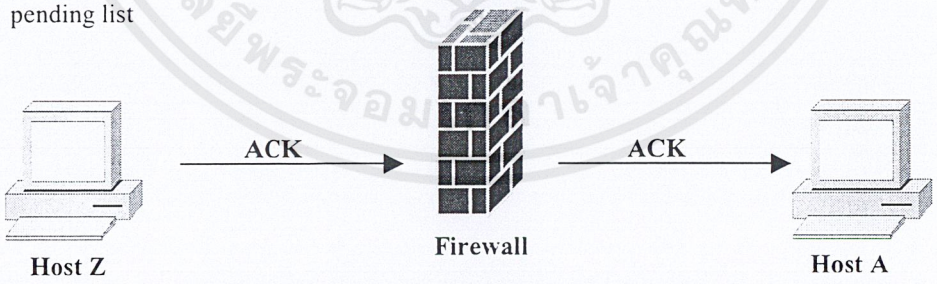
- ถ้าตรวจไม่พบ key จะทำการบันทึกเข้าไปในตารางสถานะและส่งไปยัง Z



รูปที่ 6-7 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN และ SYN/ACK packet (B)

3. ใช้ฟิลเตอร์ ACK แพ็กเก็ตป้องกันการโจมตีจาก DOS

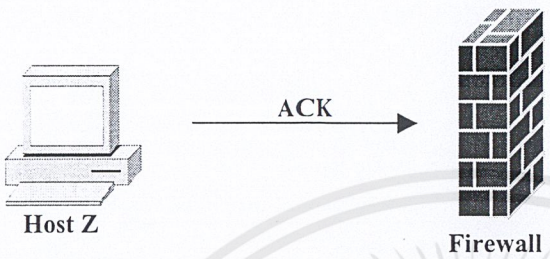
- ไฟร์วอลจะกั้นการตอบบด้วย ACK แพ็กเก็ตของ Z ไปยัง A ถ้าการเชื่อมต่อของ Z ถูกต้อง ไฟร์วอลจะหา key ที่ตรงกันในตารางสถานะและเมื่อพบจะส่ง แพ็กเก็ตไปยัง A แลลบ key ใน pending list



รูปที่ 6-8 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN/ACK packet (A)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ถ้าไม่เจอใน ตารางการเชื่อมต่อแสดงว่า ACK ไม่ใช้ส่วนหนึ่งของการเชื่อมต่อ ไฟร์วอลจะทำการblock และ log



รูปที่ 6-9 รูปแสดงการป้องกัน DoS โดยฟิลเตอร์ SYN/ACK packet (A)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7
การโจมตีเครือข่าย

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

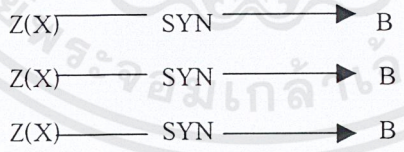
7.1 การโจมตีโดย TCP SYN Flooding

TCP SYN Flooding เป็นการโจมตีที่เกิดจากข้อบกพร่องของโพรโตคอลทีซีพี/ไอพี โดยจะถือเอาความไม่สมบูรณ์ของ three-way handshake เป็นหลัก ดังมีรายละเอียดและขั้นตอนการทำงานดังนี้

7.1.1 การทำงานของ TCP SYN Flooding

-ขั้นตอนที่ 1

โฮสต์ (Host) ที่จะทำการโจมตีจะทำการปลอมหมายเลขไอพีต้นทางเป็นโฮสต์ที่ในขณะนั้นปิดบริการอยู่ไม่สามารถตอบรับแพ็กเกจที่ระบุส่งถึงโฮสต์นั้นได้ เพื่อทำการส่งแพ็กเกจที่มีการกำหนดแฟล็ก SYN ดังรูป



Z(x) หมายถึง โฮสต์ Z ทำการปลอมหมายเลขไอพีต้นทางเป็นโฮสต์ X แล้วทำการส่งแพ็กเกจที่มีการกำหนดแฟล็ก SYN ไปยังโฮสต์เป้าหมาย คือ โฮสต์ B

-ขั้นตอนที่ 2

โฮสต์ B เมื่อได้รับการร้องขอการติดต่อ (ได้รับแพ็กเกจแฟล็ก SYN) ถ้าเป็นบริการที่โฮสต์นั้นให้บริการอยู่นั่นคือมีการเปิดพอร์ตเพื่อรอรับการขอติดต่อ ก็จะทำสร้างแพ็กเกจที่ทำการตอบกลับไปด้วย โดยจะมีการกำหนดแฟล็กเป็น SYN, ACK ไปยังโฮสต์ X เพราะหมายเลขไอพีที่ได้รับจากแพ็กเกจแฟล็ก SYN เป็นไอพีของโฮสต์ X แต่ถ้าโฮสต์ B ไม่เปิดบริการบริการนั้นก็จะส่งแพ็กเกจที่กำหนดแฟล็กเป็น RST ไปยังโฮสต์ X เพื่อยกเลิกการขอติดต่อทำให้การโจมตีไม่มีผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

X ◄——— SYN ,ACK ——— B

X ◄——— SYN ,ACK ——— B

ในขั้นตอนนี้ชั้นทีซีพีจะสร้างแพ็กเกจที่มีแฟล็กเป็น SYN,ACK เพื่อตอบรับแพ็กเกจ SYN เมื่อแพ็กเกจถูกผ่านลงไปยังชั้นของไอพีในตอนแรกจะมีการส่งแพ็กเกจออกไปจริงๆ แต่เนื่องจากโฮสต์ X ไม่เปิดบริการหรือไม่สามารถตอบกลับได้ เราท์เตอร์ที่อยู่ใกล้จะทราบว่าโฮสต์ X ไม่เปิดและเราท์เตอร์จะทำการรายงานข้อผิดพลาดกับมาที่โฮสต์ B ว่าไม่สามารถส่งแพ็กเกจถึงโฮสต์ X ได้ ในชั้นของไอพีจะไม่ทำการส่งแพ็กเกจออกไปอีกแต่ในชั้นของทีซีพีซึ่งมีคิวที่จัดการขอติดต่อที่ยังไม่สมบูรณ์อยู่ (ยังไม่ครบ three-way handshake) ไม่ทราบว่าโฮสต์ X ส่งแพ็กเกจไปไม่ถึง จะคิดว่าเกิดข้อผิดพลาดขึ้น (Error) ก็จะทำการสร้างแพ็กเกจใหม่แล้วส่งลงมายังชั้นของไอพีใหม่ ซึ่งชั้นของไอพีก็จะไม่ทำการส่งแพ็กเกจออกไปเพราะทราบแล้วว่าแพ็กเกจไปไม่ถึงยังโฮสต์นั้น

-ขั้นตอนที่ 3

ปัญหาจะเกิดเมื่อคิวที่ใช้รับการขอติดต่อ (ไม่ครบ three-way handshake) เต็มทีซีพีจะไม่รับการขอติดต่อเข้ามาใหม่ เพราะว่าคิวเต็มทีซีพีก็จะพยายามทำกระบวนการขอติดต่อ (three-way -handshake) ให้สมบูรณ์ นั่นคือรอ ACK จากโฮสต์ X ซึ่งไม่มีวันจะส่งมาถึงโฮสต์ B จะทำการรอจนกว่าจะหมดเวลาที่ตั้งไว้ (time out) แล้วโฮสต์ B จะส่งแพ็กเกจที่กำหนดแฟล็ก RST ออกไปเพื่อยกเลิกการติดต่อนั้น

: B รอ ACK

: B รอ ACK

◄——— RST ——— : B (time out)

เมื่อ time out คิวก็จะว่างลง 1 คิวเพื่อรอรับขอการติดต่อใหม่แพ็กเกจชุดใหม่ก็จะโจมตีเข้ามาอีกทำให้คิวเต็ม

7.1.2 คิวของการรอรับการขอติดต่อ

1. ในชั้นของทีซีพีแต่ละจุดที่มีการให้บริการจะมีการรอรับการขอติดต่อ (Listening) โดยใช้คิวขนาดคงที่รอรับการขอติดต่อในชั้นของทีซีพี

2. ในชั้นของแอปพลิเคชั่นจะกำหนดความยาวของ คิว (backlog) เพื่อรอรับและจัดการการขอการติดต่อซึ่งมีค่าแตกต่างกันไปตามแอปพลิเคชั่น

3. เมื่อการร้องขอการติดต่อเข้ามา (------- SYN ----->) ในชั้นของทีซีพีจะดูค่าของ Listening ถ้าค่ามากกว่า 0 ก็จะยอมให้กระทำกระบวนการติดต่อ (three way handshake) โดยทั่วไปค่านี้จะมีความสัมพันธ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กับค่า Backlog และจะลดค่าลงหนึ่งเมื่อมีการย่อมรับ การขอติดต่อได้ถูกสร้างขึ้นแล้วก็จะเพิ่มค่าขึ้นหนึ่ง แล้วขอ การ listening ว่าจะยอมรับการติดต่อ หรือไม่

- ค่าของ backlog ถูกกำหนดโดยแอปพลิเคชั่นเพื่อให้ค่า มากสุด ของคิวที่ยอมให้ ติดต่อที่จุดนั้น ๆ

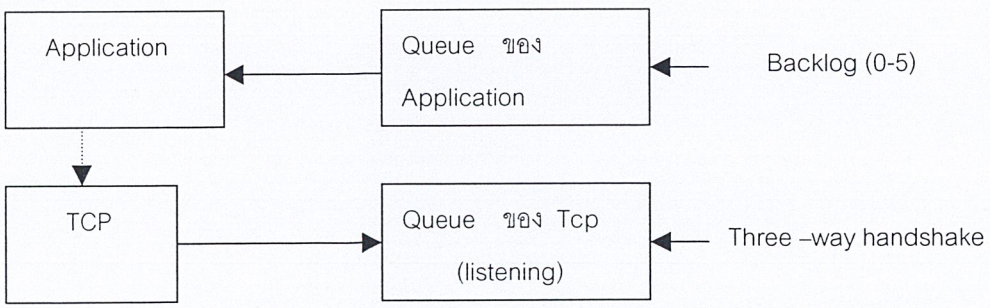
Backlog value	Max # of queued connection BSD	Solaris 2.2
0	1	0
1	2	1
2	4	2
3	5	3
4	7	4
5	8	5

ตารางที่ 7-1 แสดงขนาดของ backlog คิว

4. ค่าของ Backlog จะกำหนด คิวของการขอติดต่อเฉพาะจุดของ (ทีซีพี → แอปพลิเคชั่น) นั้นเท่านั้น ไม่ใช่คิวของระบบ

5. ทีซีพีจะจัดการส่ง SYN, ACK และจะจัดการรอจนกว่าการติดต่อจะถูกจัดตั้งขึ้น แอปพลิเคชั่นที่เซิร์ฟเวอร์จะมองไม่เห็นการขอติดต่อใหม่นี้จนกว่าจะได้รับ ACK ของกระบวนการทำ three-way handshake ถ้าไม่มีคิวสำหรับ listening การขอติดต่อที่เข้ามาใหม่ทีซีพีจะไม่สนใจ SYN ที่เข้ามาและจะไม่ส่งอะไรตอบกลับ (รวมทั้ง RST)

คิวของทีซีพีและ แอปพลิเคชั่น ที่ใช้รับการติดต่อจะแยกกันโดยคิวของทีซีพีตัวระบบโอเอสจะเป็นตัวควบคุมและจัดการ



รูปที่ 7-1 แสดงการจัดการคิวของทีซีพี และแอปพลิเคชั่น แยกกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กับค่า Backlog และจะลดค่าลงหนึ่งเมื่อมีการย่อมรับ การขอติดต่อได้ถูกสร้างขึ้นแล้วก็จะเพิ่มค่าขึ้นหนึ่ง แล้วขอ การ listening ว่าจะยอมรับการติดต่อ หรือไม่

- ค่าของ backlog ถูกกำหนดโดยแอปพลิเคชั่นเพื่อให้ค่า มากสุด ของคิวที่ยอมให้ ติดต่อที่จุดนั้น ๆ

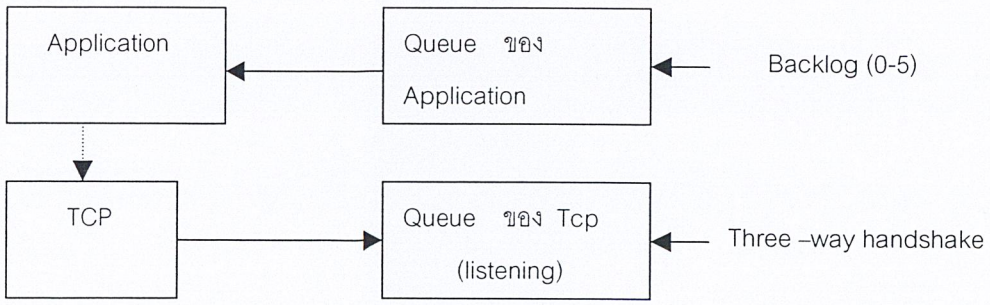
Backlog value	Max # of queued connection BSD	Solaris 2.2
0	1	0
1	2	1
2	4	2
3	5	3
4	7	4
5	8	5

ตารางที่ 7-1 แสดงขนาดของ backlog คิว

4. ค่าของ Backlog จะกำหนด คิวของการขอติดต่อเฉพาะจุดของ (ทีซีพี ⟶ แอปพลิเคชั่น) นั้นเท่านั้น ไม่ใช่คิวของระบบ

5. ทีซีพีจะจัดการส่ง SYN, ACK และจะจัดการรอจนกว่าการติดต่อจะถูกจัดตั้งขึ้น แอปพลิเคชั่นที่เซิร์ฟเวอร์จะมองไม่เห็นการขอติดต่อใหม่นี้จนกว่าจะได้รับ ACK ของกระบวนการทำ three-way handshake ถ้าไม่มีคิวสำหรับ listening การขอติดต่อที่เข้ามาใหม่ทีซีพีจะไม่สนใจ SYN ที่เข้ามาและจะไม่ส่งอะไรตอบกลับ (รวมทั้ง RST)

คิวของทีซีพีและ แอปพลิเคชั่น ที่ใช้รับการติดต่อจะแยกกันโดยคิวของทีซีพีตัวระบบโอเอสจะเป็นตัวควบคุมและจัดการ



รูปที่ 7-1 แสดงการจัดการคิวของทีซีพี และแอปพลิเคชั่น แยกกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7.1.3 ผลกระทบจาก TCP SYN Flooding

เป้าหมายของการโจมตีด้วย TCP SYN Flooding นั้นจะทำให้โฮสต์และพอร์ตหรือบริการที่โดนโจมตีจะไม่สามารถตอบรับการร้องขอการติดต่อที่เข้ามาที่พอร์ตหรือบริการนั้นได้แสมือนว่าโฮสต์ที่ถูกโจมตีไม่ได้เปิดให้บริการนั้น

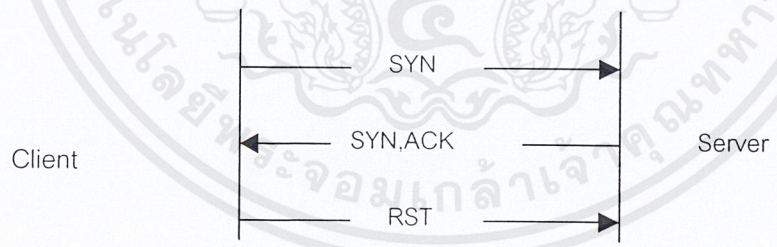
7.2 การตรวจสอบการเปิดพอร์ต (Scan Port)

การสแกนพอร์ตเป็นการหาข้อมูลของโฮสต์ที่ถูกสแกนว่าเปิดให้บริการบริการอะไรบ้าง ยกตัวอย่างเช่น WWW, ftp, E-mail เพื่อที่ผู้ที่ทำการสแกนจะได้ทราบข้อมูลของโฮสต์นั้น โดยข้อมูลที่ได้ไปนี้อาจจะทำให้เกิดผลเสียหายกับโฮสต์ที่ถูกสแกนเช่น อาจมีการสั่งโจมตีบริการที่ให้บริการอยู่ ซึ่งอาจจะทำให้บริการนั้นไม่สามารถให้บริการได้และยิ่งกว่านั้นการตรวจสอบหาที่มานั้นทำได้ยาก เช่น TCP SYN Flooding หรืออาจจะทำการบุกรุกเข้ามาในระบบโดยอาศัยข้อมูลที่ได้จากการ สแกน มาเป็นส่วนช่วย ซึ่งไม่เป็นผลดีกับโฮสต์นั้นๆ

7.2.1 การทำงานของ Scan Port

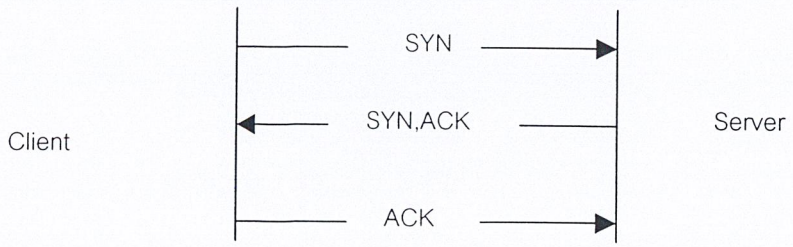
การสแกนพอร์ตแบ่งตามการทำงานออกเป็น 2 ชนิดคือ Full Scan และ Half Scan ซึ่งทั้ง 2 แบบมีความแตกต่างกันในแพ็กเกจที่ 3 นั่นคือ ACK ในแบบ Full Scan และ RST ในแบบ Half Scan ซึ่งจะเป็นตัวบอกว่ามีการติดต่อกันเกิดขึ้นหรือไม่ ดังรูป

-Half Scan



รูปที่ 7-2 แสดงการ Scan Port แบบ Half Scan

-Full Scan



รูปที่ 7-3 แสดงการ Scan Port แบบ Full Scan

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การสแกนพอร์ตมีลักษณะเหมือนกับการขอติดต่อขอเข้าใช้บริการต่างๆ ของโฮสต์ที่ให้บริการ โดยที่ การขอใช้บริการเหล่านั้นต้องการทราบว่าโฮสต์นั้นเปิดให้บริการอะไรบ้าง โดยจะทำการร้องขอการติดต่อมาที่บริการต่างๆ จนทราบว่าโฮสต์นั้นให้บริการอะไรบ้าง

การสแกนพอร์ตแบบ Full Scan จะเป็นมีการติดต่อกันเกิดขึ้นจริง ในบางโฮสต์อาจจะมีการบันทึกการติดต่อที่เกิดขึ้น (Logfile System) ทำให้การสแกนพอร์ตแบบ Full Scan สามารถตรวจได้ง่าย ส่วนแบบ Half Scan จะไม่มีการติดต่อเกิดขึ้นจริง โฮสต์ส่วนมากจะไม่ทำการบันทึกข้อมูลลงล็อกไฟล์ของระบบ

7.3 การปลอมไอพี (IP-Spoofing)

ในการโจมตีเครือข่ายส่วนมากมักจะมีการปลอมหมายเลขไอพีเป็นส่วนประกอบเสมอ ยกตัวอย่างการปลอมไอพีใน TCP SYN Flooding การปลอมไอพีจะทำให้โฮสต์ที่ถูกโจมตีไม่รู้ว่าเป็นการส่งแพ็กเกจมาจากที่ไหน หรือทำให้เข้าใจผิดเพื่อใช้ในการผ่านไฟล์วอล์ล

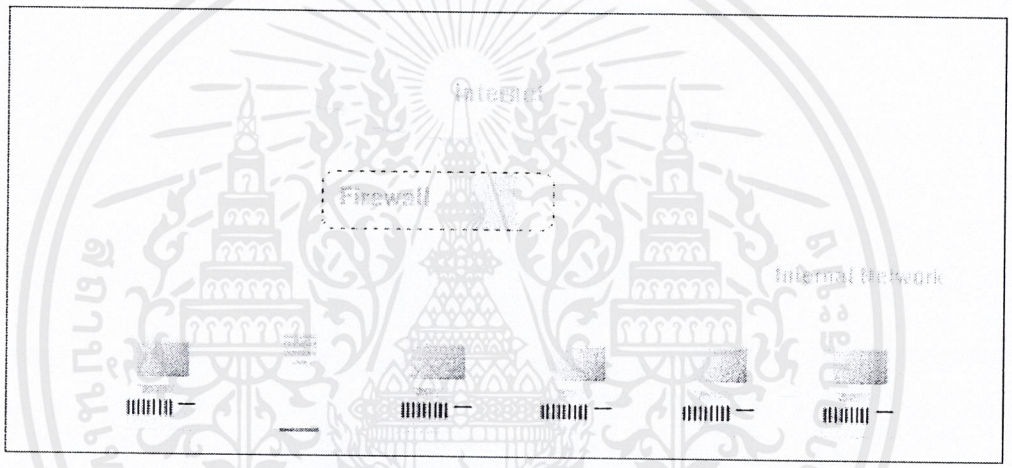
การปลอมไอพีคือ การที่แพ็กเกจที่ออกจากโฮสต์ที่ถูกกำหนดหมายเลขไอพีหนึ่งเอาไว้ เช่น เป็น 161.246.6.220 โปรแกรมหรือเครื่องมือบางอย่างสามารถจะทำการสร้างแพ็กเกจขึ้นมาโดยยอมให้ทำการกำหนดค่าของหมายเลขไอพีได้ ซึ่งค่าที่กำหนดนี้อาจจะเป็นค่าที่ถูกต้องหรืออาจจะไม่ถูกต้องก็ได้ เช่น เครื่องหมายเลข ไอพี 161.246.6.220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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 8
การออกแบบโปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล

8.1 การออกแบบโปรแกรม

ในการออกแบบโปรแกรมจะนำความสามารถของไฟร์วอลล์วัน (FW – 1) มาประยุกต์ โดยที่ในการทำงานของไฟร์วอลล์นั้นไฟร์วอลล์จะเป็นตัวกั้นกลางระหว่างโฮสต์ภายในและโฮสต์ภายนอก



รูปที่ 8-1 ไฟร์วอลล์กั้นระหว่างอินเตอร์เน็ตกับเน็ตเวิร์กภายใน

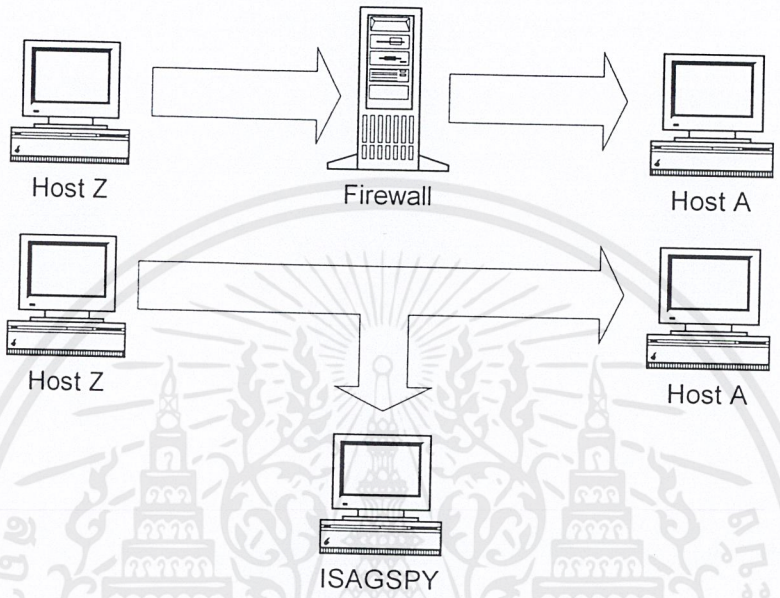
ในความหมายทางด้านการก่อสร้างแล้ว ไฟร์วอลล์ จะหมายถึง กำแพงที่เอาไว้ป้องกันไฟไม่ให้ลุกลามไปยังส่วนอื่นๆ ส่วนทางด้านคอมพิวเตอร์นั้นก็จะมีความหมายคล้ายๆ กันก็คือ เป็นระบบที่เอาไว้ป้องกันอันตรายจากอินเตอร์เน็ตหรือเน็ตเวิร์กภายนอกนั่นเอง

ไฟร์วอลล์ เป็นคอมโพเน็นต์หรือกลุ่มของคอมโพเน็นต์ที่ทำหน้าที่ในการควบคุมการเข้าถึงระหว่างเน็ตเวิร์กภายนอกหรือเน็ตเวิร์กที่เราคิดว่าไม่ปลอดภัย กับเน็ตเวิร์กภายในหรือเน็ตเวิร์กที่เราต้องการจะป้องกัน โดยที่คอมโพเน็นต์นั้นอาจจะเป็นเราเตอร์ คอมพิวเตอร์ หรือเน็ตเวิร์ก ประกอบกันก็ได้ ขึ้นอยู่กับวิธีการหรือ Firewall Architecture ที่ใช้

ในการดักจับข้อมูลนั้นไม่จำเป็นต้องเป็นตัวที่อยู่กั้นกลางระหว่างโฮสต์ที่กำลังติดต่อสื่อสารกันอยู่ เพียงแค่เป็นโฮสต์ที่อยู่บนโลคอลเน็ตเวิร์ก (local network)เดียวกับโฮสต์ที่ต้องการดักจับข้อมูล เนื่องจาก Ethernet ถูกสร้างขึ้นมาบนกฎของการใช้ร่วมกัน นั่นคือคอมพิวเตอร์ทุกเครื่องบนโลคอลเน็ตเวิร์ก จะใช้ Ethernet wire เดียวกัน ฮาร์ดแวร์ของ Ethernet ถูกสร้างมาด้วย “ตัวกรอง” ที่จะไม่น่าสนใจ traffic ที่เป็นของ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

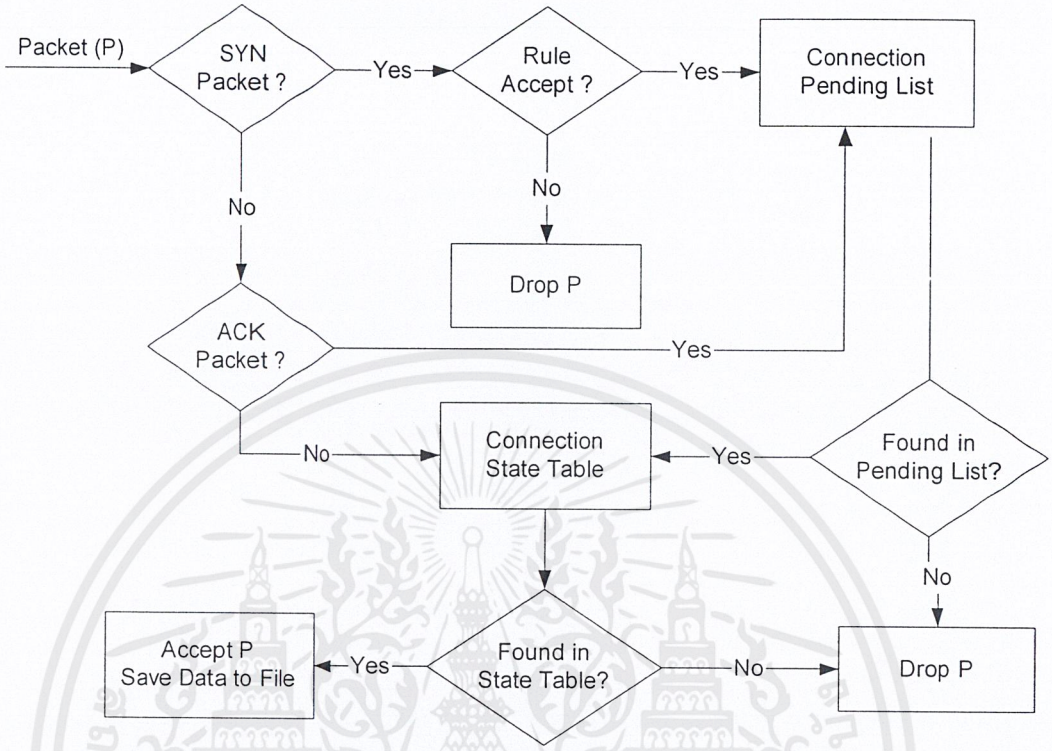
คนอื่น ซึ่งทำได้โดยจะไม่สนใจเฟรมที่มี MAC address ไม่ตรงกับของตนเอง แต่โปรแกรมดักจับข้อมูลจะตัดตัวกรองนี้ออกไป ทำให้ฮาร์ดแวร์อยู่ใน “โหมดสำส่อน” (promiscuous mode)



รูปที่ 8-2 ความแตกต่างระหว่างไฟร์วอลท์และโปรแกรมดักจับข้อมูล

ดังนั้นจะเห็นได้ว่าสามารถใช้หลักการในการตรวจสอบสถานะแพ็กเก็ตของไฟร์วอลท์มาใช้ในการสร้างโปรแกรมดักจับข้อมูลได้ แต่มีข้อจำกัดที่ว่าโปรแกรมดักจับข้อมูลจะต้องทำงานในโลคอลเน็ตเวิร์กเท่านั้นไม่สามารถดักจับข้อมูลบนเครือข่ายอินเตอร์เน็ทได้

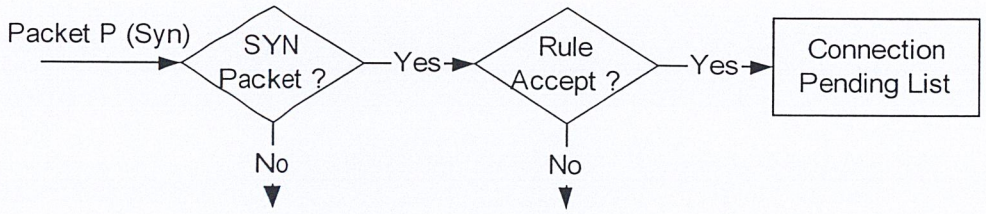
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 8-3 โฟลว์ชาร์ต ของโปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล

จากทฤษฎีในบทที่ 3 เราสามารถนำมาเขียนให้อยู่ในรูป โฟลว์ชาร์ตได้ดังรูปที่ 8-3 และสามรถแบ่งการทำงานได้ออกเป็นดังนี้โดยที่ Packet (P) จะเป็นแพ็กเก็ตที่ได้รับเข้ามา

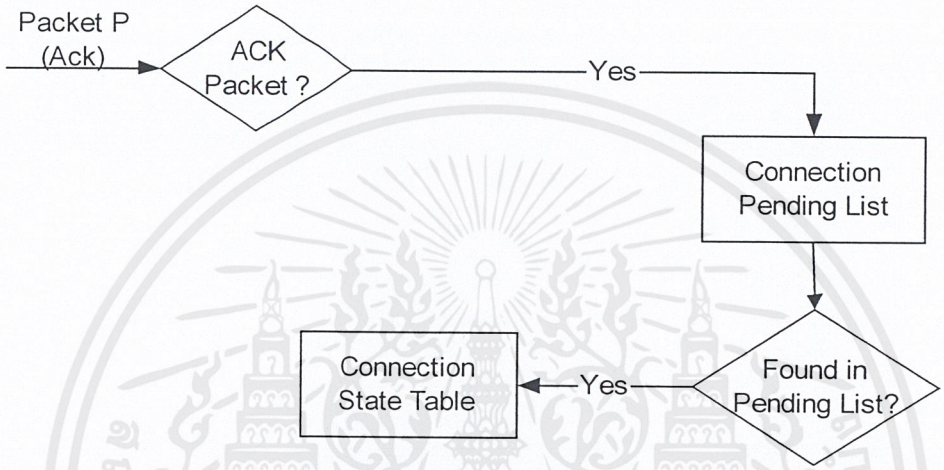
1. เมื่อแพ็กเก็ต P เข้ามาจะทำการตรวจสอบว่าเป็น SYN แพ็กเก็ตหรือไม่ ถ้าใช่จะนำแพ็กเก็ตนั้นไปตรวจสอบกับกฎ (rule) ถ้าผ่านกฎจะนำหมายเลขแอดเดรสปลายทาง หมายเลขแอดเดรสต้นทาง หมายเลขพอร์ตปลายทาง หมายเลขพอร์ตต้นทาง และ ชนิดของโปรโตคอล โดยค่าทั้งหมดนี้จะเรียกว่าคีย์ (keys) ไปเก็บไว้ใน Pending List ดังรูป



รูปที่ 8-4 โฟลว์ชาร์ตแสดงการทำงานเมื่อ Syn แพ็กเก็ตเข้ามา

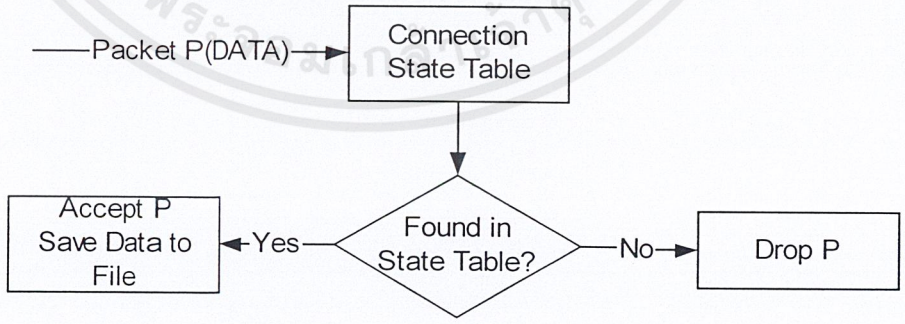
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. ถ้าแพ็กเก็ตที่รับเข้ามาเป็น ACK แพ็กเก็ต จะนำค่าคีย์ของแพ็กเก็ตนั้นไปเทียบกับคีย์ใน Pending List ว่าตรงกันหรือไม่ ถ้าไม่ตรงจะละเลยไม่สนใจแพ็กเก็ตนั้น แต่ถ้าค่าคีย์ตรงกับใน Pending List จะนำค่าคีย์นั้นไปเก็บไว้ใน State table แล้วลบค่าคีย์นั้นใน Pending List ทิ้งไป



รูปที่ 8-5 โฟลว์ชาร์ตแสดงการทำงานเมื่อ Ack แพ็กเก็ตเข้ามา

3. เมื่อแพ็กเก็ตที่เป็นข้อมูลเข้ามา จะนำค่าคีย์ของแพ็กเก็ตนั้นไปเทียบกับค่าคีย์ใน State Table ถ้าตรงกันจะทำการจัดเก็บข้อมูลภายในแพ็กเก็ต ถ้าคีย์ไม่ตรงจะละเลยไม่สนใจแพ็กเก็ตนั้น



รูปที่ 8-6 โฟลว์ชาร์ตแสดงการทำงานเมื่อ Data แพ็กเก็ตเข้ามา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการเขียนโปรแกรมจะใช้ไลบารีพิเศษมาช่วยในการเขียนโปรแกรมคือไลบารี libnids ซึ่งช่วยในขั้นตอนการจัดเก็บและเปรียบเทียบค่าคีย์ใน Pending List โดยที่จะต้องไลบารี libpcap และ libnet ด้วยจึงจะสามารถใช้งานไลบารี libnids ได้

8.2 โครงสร้างข้อมูลของโปรแกรม

ในโปรแกรมได้ทำการสร้างโครงสร้างข้อมูล (Data Structure) ขึ้นมาเพื่อใช้เก็บข้อมูลของ state table โดยให้ชื่อโครงสร้างข้อมูลนี้ว่า struct table มีรูปแบบดังนี้

```
struct table   {
        char  *Saddr;
        char  *Daddr;
        int  Sport;
        int  Dport;
        char  ProType[12];
        char  FileName[80];
        int  flag;
};
```

8.2.1 ความหมายของตัวแปรแต่ละตัวมีดังนี้

Saddr	หมายเลขแอดเดรสต้นทาง
Daddr	หมายเลขแอดเดรสปลายทาง
Sport	หมายเลขพอร์ตต้นทาง
Dport	หมายเลขพอร์ตปลายทาง
ProType	ชนิดของโพรโตคอล
FileName	เก็บชื่อของไฟล์ที่ทำการเก็บข้อมูลของแต่ละเซสชัน
flag	เป็นเฟล็กที่ใช้บอกสถานะว่า state table ช่องนี้ว่างอยู่หรือไม่

Flag	Saddr	Daddr	Sport	Dport	ProType	FileName

ตารางที่ 8-1 ตัวอย่างของ State Table ที่ใช้ภายในโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

8.3 ฟังก์ชันต่างๆ ของโปรแกรม

8.3.1 ฟังก์ชันหลัก (main function)

- **int main (int argc, char *argv)**

จุดมุ่งหมาย : เป็นฟังก์ชันหลักในการทำงานของโปรแกรม ประกอบไปด้วย

(1) กำหนดค่าเริ่มต้นให้กับตัวแปรต่างๆ

(2) รอรับค่าอาร์กิวเมนต์ต่างที่ผู้ใช้ ได้กำหนดเข้ามาตามออปชันที่มี

(3) รันโปรแกรมในลักษณะเดมอนโพรเซส

(4) รอรับแพ็กเก็ตจนกระทั่งจบโปรแกรม

ขั้นตอนการทำงาน :

(1) กำหนดค่าเริ่มต้นให้กับตารางสเตต

(2) ถ้าค่าอาร์กิวเมนต์ที่ผู้ใช้กำหนดไม่ถูกต้อง แจ้งให้ผู้ใช้ทราบ

(3) ถ้าไม่ได้กำหนดออปชันดีบั๊กไว้จะรันโปรแกรมให้ลักษณะเดมอนโพรเซส

(4) วนลูปเพื่อรอรับแพ็กเก็ต ถ้าไม่มีแพ็กเก็ตเข้ามาโปรแกรมจะพัก (sleep)จนกระทั่งมีแพ็กเก็ตใหม่เข้ามาจึงทำงานใหม่อีกครั้ง

8.3.2 ฟังก์ชันอื่นๆ

- **int process_command (int argc, char **argv)**

จุดมุ่งหมาย : ตรวจสอบค่าอาร์กิวเมนต์ที่ผู้ใช้กำหนดและกำหนดพารามิเตอร์ให้กับโปรแกรมตามค่าอาร์กิวเมนต์ที่ได้รับ

อากิวเมนต์ :

- argc, argv เป็นค่าของอาร์กิวเมนต์ที่ได้รับ

ขั้นตอนการทำงาน :

(1) รับค่าอาร์กิวเมนต์

(2) นำค่าอาร์กิวเมนต์มาตรวจสอบและแยกแยะตามออปชัน

(3) กำหนดค่าพารามิเตอร์ตามค่าอาร์กิวเมนต์ที่ได้รับ

(4) ถ้าค่าอาร์กิวเมนต์ที่ได้รับไม่ถูกต้องจะคืนค่า 1 กลับไป

ผลลัพธ์ :

คืนค่า 1 กลับไปถ้าอาร์กิวเมนต์ที่ได้รับผิด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **void tcp_callback (struct tcp_stream *a_tcp, void ** this_time_not_needed)**

จุดมุ่งหมาย : เป็นฟังก์ชันในไลบารี libnids เพื่อใช้ในการอ่านแพ็กเก็ตและตัดสินว่าจะรับแพ็กเก็ตนั้นมาเก็บไว้หรือไม่

อาก๎วเมนต์ :

- a_tcp เป็นค่าที่รับแพ็กเก็ตเข้ามาโดยเป็น sturcture ของ tcp_stream ภายในจะเก็บค่าพารามิเตอร์ต่างๆ ของแพ็กเก็ตเอาไว้
- this_time_not_needed เป็นค่าพารามิเตอร์ที่ส่งเข้ามาเพื่อกำหนดการทำงาน

ขั้นตอนการทำงาน :

(1) เมื่อได้รับแพ็กเก็ตค่าพารามิเตอร์ a_tcp->nids_state จะเก็บสถานะของแพ็กเก็ตนั้น

(2) ตรวจสอบสถานะของแพ็กเก็ต

(3) ถ้าแพ็กเก็ตนั้นอยู่ในสถานะ NIDS_JUST_EST จะถูกทำการจัดเก็บค่าพารามิเตอร์ต่างๆ ของแพ็กเก็ต (ค่าคีย์)เอาไว้ในตารางสเตต โดยตรวจสอบว่ามีค่าคีย์นั้นถูกเก็บไว้ก่อนหน้านี้หรือไม่ด้วย

(4) ถ้าแพ็กเก็ตนั้นอยู่ในสถานะ DATA จะนำค่าคีย์ของแพ็กเก็ตนั้นมาตรวจสอบกับตารางสเตต ถ้ามีคีย์ที่ตรงกันข้อมูลของแพ็กเก็ตนั้นจะถูกจัดเก็บลงไฟล์

(5) ถ้าแพ็กเก็ตอยู่ในสถานะอื่นๆ ค่าคีย์ที่ถูกเก็บอยู่ในตารางสเตตจะถูกทิ้งไป

- **int getaddr (char *cp)**

จุดมุ่งหมาย : จัดเก็บหมายเลขไอพีแอดเดรสที่ผู้ใช้ต้องการตรวจสอบไว้ในตารางแอดแดรส

อาก๎วเมนต์ :

- cp ค่าอาร์กิวเมนต์ที่ผู้ใช้ป้อนเข้ามา

ขั้นตอนการทำงาน :

(1) นำอาร์กิวเมนต์ที่ได้รับมาแยกออกเป็นส่วน

(2) ตรวจสอบว่าอาร์กิวเมนต์ที่ได้รับมาถูกต้องหรือไม่

(3) จัดเก็บหมายเลขแอดเดรสทั้งหมดไว้ในตารางแอดเดรสเพื่อใช้ในการตรวจสอบ

ผลลัพธ์ :

- จะคืนค่า 1 เมื่อผู้ใช้ใส่ค่าอาร์กิวเมนต์ไม่ถูกต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **int getport (char *cp)**

จุดมุ่งหมาย : จัดเก็บหมายเลขพอร์ตที่ผู้ใช้ต้องการตรวจสอบไว้ในตารางพอร์ต

อาก๊วเมนต์ :

- cp ค่าอาร์กิวเมนต์ที่ผู้ใช้ป้อนเข้ามา

ขั้นตอนการทำงาน :

(1) นำอาร์กิวเมนต์ที่ได้รับมาแยกออกเป็นส่วน

(2) ตรวจสอบว่าอาร์กิวเมนต์ที่ได้รับมาถูกต้องหรือไม่

(3) จัดเก็บหมายเลขพอร์ตทั้งหมดไว้ในตารางพอร์ตเพื่อใช้ในการตรวจสอบ

ผลลัพธ์ :

- จะคืนค่า 1 เมื่อผู้ใช้ใส่ค่าอาร์กิวเมนต์ไม่ถูกต้อง

- **int check_addr (struct tcp_stream *a_tcp)**

จุดมุ่งหมาย : ตรวจสอบหมายเลขแอดเดรสของแพ็กเก็ตกับตารางแอดเดรส

อาก๊วเมนต์ :

- a_tcp เป็นค่าที่รับแพ็กเก็ตเข้ามาโดยเป็น sturcture ของ tcp_stream ภายในจะเก็บค่าพารามิเตอร์ต่างๆ ของแพ็กเก็ตเอาไว้

ขั้นตอนการทำงาน :

(1) นำหมายเลขแอดเดรสของแพ็กเก็ตที่ได้รับมาตรวจสอบกับหมายเลขแอดเดรสในตารางแอดเดรส

ผลลัพธ์:

- จะคืนค่า 1 ถ้าหมายเลขแอดเดรสตรงกับในค่าตารางแอดเดรส

- **int check_port (struct tcp_stream *a_tcp)**

จุดมุ่งหมาย : ตรวจสอบหมายเลขพอร์ตของแพ็กเก็ตกับตารางพอร์ต

อาก๊วเมนต์ :

- a_tcp เป็นค่าที่รับแพ็กเก็ตเข้ามาโดยเป็น sturcture ของ tcp_stream ภายในจะเก็บค่าพารามิเตอร์ต่างๆ ของแพ็กเก็ตเอาไว้

ขั้นตอนการทำงาน :

(1) นำหมายเลขพอร์ตของแพ็กเก็ตที่ได้รับมาตรวจสอบกับหมายเลขพอร์ตในตารางพอร์ต

ผลลัพธ์ :

- จะคืนค่า 1 ถ้าหมายเลขพอร์ตตรงกับค่าในตารางพอร์ต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **int ignore_tcp_packet (struct tcp_stream *a_tcp)**

จุดมุ่งหมาย : ตรวจสอบแพ็กเก็ตที่เข้ามาว่าผู้ใช้สนใจเฝ้าดูและเก็บข้อมูลหรือไม่

อากิวเมนต์ :

- a_tcp เป็นค่าที่รับแพ็กเก็ตเข้ามาโดยเป็น sturcture ของ tcp_stream ภายในจะเก็บค่าพารามิเตอร์ต่างๆ ของแพ็กเก็ตเอาไว้

ขั้นตอนการทำงาน :

(1) ตรวจสอบออปชั่นว่าผู้ใช้ได้กำหนดค่าใดไว้บ้าง

(2) ตรวจสอบค่าที่ผู้ใช้ได้กำหนดไว้กับตารางแอดเดรสและตารางพอร์ต

ผลลัพธ์ :

- จะคืนค่า 0 ถ้าเป็นแพ็กเก็ตที่ผู้ใช้สนใจ

- **char *adres1 (struct tuple4 addr)**

จุดมุ่งหมาย : เก็บหมายเลขไอพีแอดเดรสของแอดเดรสต้นทาง

อากิวเมนต์ :

- addr เป็นตัวแปรชนิด struct truple 4 ของ libnids ซึ่งเก็บหมายเลขพอร์ตและหมายเลขแอดเดรสของ ไคลเอ็นต์และเซิร์ฟเวอร์เอาไว้

ขั้นตอนการทำงาน :

(1) เก็บหมายเลขไอพีแอดเดรสต้นไว้ในบัฟเฟอร์

ผลลัพธ์ :

- คืนค่าออกมาเป็นหมายเลขไอพีแอดเดรสต้นทาง

- **char *adres1 (struct tuple4 addr)**

จุดมุ่งหมาย : เก็บหมายเลขไอพีแอดเดรสของแอดเดรสปลายทาง

อากิวเมนต์ :

- addr เป็นตัวแปรชนิด struct truple 4 ของ libnids ซึ่งเก็บหมายเลขพอร์ตและหมาย2เลขแอดเดรสของ ไคลเอ็นต์และเซิร์ฟเวอร์เอาไว้

ขั้นตอนการทำงาน :

(1) เก็บหมายเลขไอพีแอดเดรสปลายทางไว้ในบัฟเฟอร์

ผลลัพธ์ :

- คืนค่าออกมาเป็นหมายเลขไอพีแอดเดรสปลายทาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- **char *formatDate (time_t timeValue)**

จุดมุ่งหมาย : บันทึกเวลาในขณะที่แพ็กเก็ตเข้า

ผลลัพธ์ : ค่าเวลาในขณะที่แพ็กเก็ตเข้ามา

8.4 ข้อจำกัดของโปรแกรม

โปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูลสร้างขึ้นโดยมีข้อจำกัดของการใช้งานดังนี้

(1) ผู้ที่มีสิทธิ์เป็นรูท (root) เท่านั้นที่จะสามารถใช้งานโปรแกรมได้

(2) ในกรณีที่เครือข่ายนั้นมีแพ็กเก็ตในปริมาณมาก อาจทำให้เปลืองพื้นที่ในการเก็บข้อมูล

(3) สามารถตรวจจับได้เฉพาะชั้นทีซีพีเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 9
สรุปและวิจารณ์

ในการทำงานของปริญญานิพนธ์นี้ สามารถที่จะสร้างซอฟต์แวร์ที่สามารถดักจับข้อมูลภายในระบบเน็ตเวิร์กพร้อมทั้งเก็บบันทึกข้อมูล เพื่อนำไปวิเคราะห์การทำงานของระบบ การตรวจจับผู้บุกรุก หรือการศึกษาเทคนิคการทำงานของโพรโตคอล ข้อมูลที่ดักจับได้จะถูกบันทึกลงในไฟล์ ซึ่งสะดวกต่อการนำไปวิเคราะห์หรือนำไปใช้งานได้ภายหลัง

ในการดูแลระบบเครือข่ายจะมีการเปลี่ยนแปลงตลอดเวลาการที่จะดูแลระบบให้ปลอดภัยและสามารถใช้งานได้อย่างมีประสิทธิภาพนั้น ผู้ดูแลจะต้องทราบสิ่งใหม่ที่เกี่ยวข้องตลอดเวลา ทั้งทางเทคโนโลยี กระบวนการบุกรุกที่มีการค้นพบและคิดขึ้นใหม่ ดังนั้นการตรวจสอบและดูแลการใช้งานระบบของผู้ใช้ที่ใช้งานอยู่ภายในระบบจึงจำเป็น เพื่อนำไปศึกษาและแก้ปัญหาที่เกิดขึ้นภายในระบบ

9.1 ปัญหาและอุปสรรค

ในการพัฒนาโปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูลนี้ มีปัญหาและอุปสรรคในการพัฒนาอยู่หลายประการ ได้แก่

(1) โปรแกรมที่พัฒนาขึ้นมานี้สามารถทำงานได้ในระบบปฏิบัติการยูนิกซ์หรือลีนุกซ์เท่านั้น ในการพัฒนาได้ทำการพัฒนาบนระบบปฏิบัติการลีนุกซ์ Slackware 7.1 หากมีการนำเอาโปรแกรมไปใช้กับระบบปฏิบัติการอื่นๆ อาจเกิดการผิดพลาดได้

(2) ระบบปฏิบัติการจะต้องทำการติดตั้งไลบารี libpcap, libnet, libnids ก่อนจึงจะสามารถติดตั้งโปรแกรมได้ เนื่องจากในโปรแกรมจำเป็นต้องใช้ไลบารีเหล่านี้เพื่อตรวจจับแพ็กเก็ต

(3) ระบบจะต้องมีพื้นที่เพียงพอสำหรับการจัดเก็บล็อกไฟล์ที่ได้จากการวิเคราะห์สถานะแล้ว ความต้องการพื้นที่ในการจัดเก็บนั้นแตกต่างกันตามความมากน้อยของแพ็กเก็ตที่เข้ามา

(4) ยังไม่สามารถวิเคราะห์พฤติกรรมของผู้ใช้ได้ ทำได้เพียงวิเคราะห์สถานะในการขอทำการเชื่อมต่อ เพื่อเป็นประโยชน์ให้กับแอพพลิเคชันอื่นๆ ที่ต้องการข้อมูลของผู้ใช้ที่ได้เข้ามาขอใช้บริการ

9.2 คุณลักษณะของโปรแกรมที่ได้พัฒนา

(1) การใช้งานเป็นแบบ command line นั่นคือป้อนคำสั่งที่พรอมต์เพื่อสั่งรันโปรแกรมได้เลย

(2) สามารถติดตามสถานะในระดับชั้นทีซีพีได้

(3) ผลลัพธ์ที่ได้จากโปรแกรมจะถูกจัดเก็บให้อยู่ในรูปของเท็กซ์ไฟล์ (text file) และแบ่งแยกการจัดเก็บตามรูปแบบของโพรโตคอลนั้นๆ

(4) โปรแกรมจะถูกรันในลักษณะของเดมอนโพรเซส (background process) หรือหากไม่ต้องการให้รันเป็นเดมอนก็สามารถทำได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(5) ในกรณีที่ไม่ต้องการให้รันในแบบเดมอนโพรเซส เมื่อมีการขอทำการเชื่อมต่อกับเซอร์วิสที่เราทำการดักจับแพ็กเก็ต จะมีการแสดงหมายเลขไอพีแอดเดรสและโพรโตคอลที่เข้ามาขอทำการเชื่อมต่อ รวมทั้งชื่อไฟล์ที่จัดเก็บข้อมูลของเซสชันนั้นด้วย

(6) สามารถตรวจสอบเฉพาะข้อมูลที่ถูกส่งจากไคลเอ็นต์และข้อมูลที่ถูกส่งออกจากเซิร์ฟเวอร์อย่างใดอย่างหนึ่งหรือทั้งสองอย่างได้ เพื่อความสะดวกในการวิเคราะห์พฤติกรรมของผู้ใช้

9.3 แนวทางในการพัฒนางานวิจัยเพิ่มเติม

(1) สร้างกฎ (rule) เพิ่มเติมโดยอาจทำให้รับกฎได้จากไฟล์

(2) พัฒนาให้สามารถดักจับข้อมูลได้ทุกโพรโตคอล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

[1] Douglas E. Comer, David L. Stevens, "*Internetworking With TCP/IP*", Second Edition, Prentice Hall, 1996, pp. 43-54, 363-401

[2] Andrew S. Tanenbaum, "*Computer Networks*", Third Edition, Prentice Hall, 1996, pp. 413-448, 521-543

[3] Stevens W. Richard, "*Unix Network Programming*", Original U.S. Edition, Prentice Hall India, 1999, pp. 1-86

[4] สุวัฒน์ ปุณณชัยยะ, ต้น ตันฑ์สุทธิวงศ์, สุพจน์ ปุณณชัยชนะ, "*เปิดโลกของ TCP/IP และโพรโตคอลของอินเทอร์เน็ต*", โปรวิชั่น, 2543

[5] สุรศักดิ์ สงวนพงษ์, "*สถาปัตยกรรมและโปรโตคอลทีซีพี/ไอพี*", ซีเอ็ดยูเคชั่น, 2543

[6] สันติ ศรีลาศักดิ์, วรวุฒิ เที่ยงธรรม, "*เจาะประเด็นงานเขียนโปรแกรมบนลีนุกซ์*", ออฟเซ็ท เพรส, 2542

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก
การติดตั้งและการใช้งานโปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล

การติดตั้งโปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล

การติดตั้งทำตามขั้นตอนดังนี้

```
root@linux-server# tar xzvf isagspy.tar.gz
root@linux-server# cd isagspy
root@linux-server# make
root@linux-server# make install
```

การใช้งานโปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูล

โปรแกรมดักจับแพ็กเก็ตแบบสเตตฟูลรันบนระบบปฏิบัติการยูนิกส์ (unix) และผู้ที่มีสิทธิเป็นรูท (root) เท่านั้นที่จะสามารถสั่งรันโปรแกรมได้ โดยหลังจากคอมไพล์และติดตั้งโปรแกรมแล้วสามารถใช้งานโปรแกรมได้โดยอาจใช้ร่วมกับออปชัน (options) ต่างๆดังนี้

-d	แสดงการเชื่อมต่อออกทางหน้าจอ
-i	ดักจับข้อมูลเฉพาะแพ็กเก็ตที่เข้ามา
-o	ดักจับข้อมูลเฉพาะแพ็กเก็ตที่ออกไป
-a <Address>	ดักจับข้อมูลเฉพาะหมายเลขไอพีแอดเดรสที่เรากำหนดเท่านั้น
-f <file>	จัดเก็บข้อมูลที่ได้ลงใน file
-c <count>	ออกจากโปรแกรมหลังจากรับแพ็กเก็ตครบตามที่ระบุไว้
-p <port>	ดักจับข้อมูลเฉพาะ หมายเลขพอร์ตปลายทางที่เรากำหนดเท่านั้น

ตัวอย่างของคำสั่งที่ใช้ในการรันโปรแกรม

```
root@linux-server# isagspy
```

- เป็นการดักจับข้อมูลทั้งหมดภายในเครือข่ายโดยรันอยู่ในลักษณะเดมอนโพรเซส

```
root@linux-server# isagspy –d
```

- ดักจับข้อมูลภายในเครือข่ายทั้งหมดและแสดงการเชื่อมต่อทางหน้าจอ

```
root@linux-server# isagspy –p '1-23,80'
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ดักจับข้อมูลภายในเครือข่ายโดยสนใจเฉพาะที่มีหมายเลยพอร์ตตั้งแต่ 1 ถึง 23 และหมายเลขพอร์ต 80

```
root@linux-server#   isagspy –S ‘161.246.6.*,203.24.5.3’
```

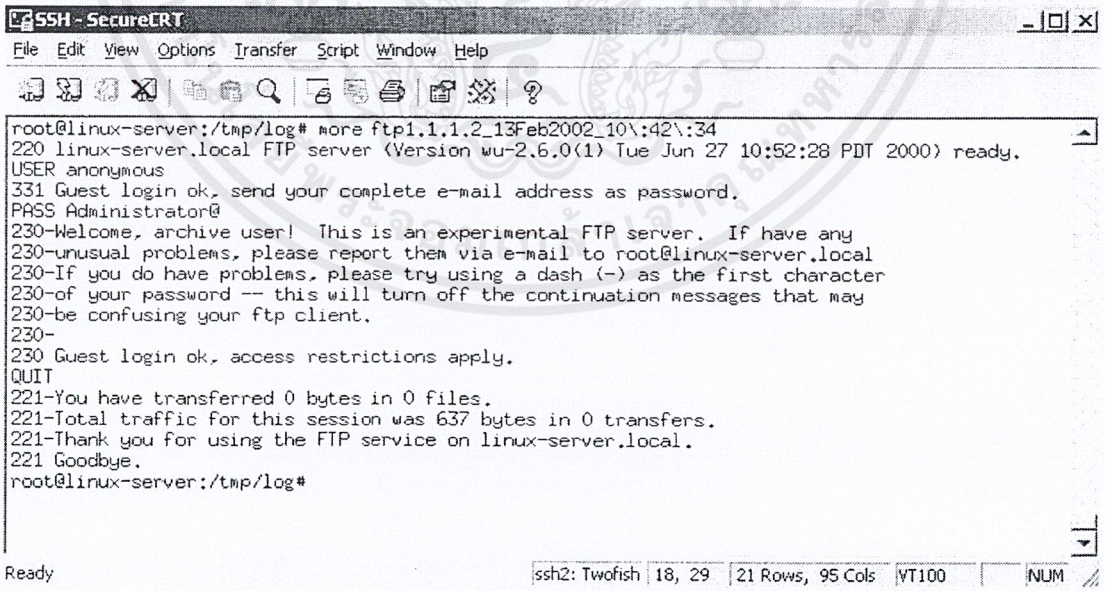
- ดักจับข้อมูลภายในเครือข่ายโดยสนใจเฉพาะคลาสซีของไอพี 161.246.6 และหมายเลขไอพี 203.24.5.3

```
root@linux-server#   isagspy –p telnet,ssh  –S 161.246.6.142  -f  test
```

- ดักจับข้อมูลภายในเครือข่ายโดยสนใจเฉพาะหมายเลขไอพี 161.246.6.142 และหมายเลขพอร์ตของบริการเทลเน็ตกับซีเคียวเชล (หมายเลขพอร์ต 23, 22) โดยจัดเก็บลงไฟล์ชื่อ test

ตัวอย่างผลการรันโปรแกรม

เมื่อรันโปรแกรมแล้วหากไม่ได้ใช้ออปชัน –f โปรแกรมจะทำการจัดเก็บล็อกไฟล์ไปไว้ที่ไดเร็กทอรี /tmp/log โดยชื่อไฟล์มีรูปแบบดังนี้ service sourceaddress date month year hour:min:second



รูปที่ 1 ตัวอย่างล็อกไฟล์ที่ได้จากการดักจับข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้