โปรแกรม วิเคราะห์นโยบายไฟร์วอล

Firewall Policy Analysis Tool

โดย

นาย พิชชา เตียววิริยะกุล

นาย พัทธดนย์ ขมะณะรงค์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชา วิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมวิเคราะห์นโยบายไฟร์วอล
Firewall Policy Analysis Tool

ผู้จัดทำ

1. นาย พิชชา เตียววิริยะกุล รหัสประจำตัว 42010229
2. นาย พัทธดนย์ ขมะณะรงค์ รหัสประจำตัว 42010234

อาจารย์ที่ปรึกษา
(อาจารย์ ธนา หงษ์สุวรรณ)

อาจารย์ที่ปรึกษา
(อาจารย์ อัครเดช วัชระภูพงษ์)

โปรแกรมวิเคราะห์นโยบายไฟร์วอล

นาย พัทธดนย์ ขมะณะรงค์ 42010224
นาย พิชชา เตียววิริยะกุล 42010229
อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

ในปัจจุบันการติดต่อสื่อสารบนเครือข่ายมีความสำคัญต่อทุก ๆ องค์กรเป็นอย่างมาก แต่ละองค์กรจึงต้องการความปลอดภัยบนเครือข่ายคอมพิวเตอร์ของตนเอง จึงมีการใช้อุปกรณ์ในการเพิ่มความปลอดภัยให้กับเครือข่าย ซึ่งอุปกรณ์ที่ใช้งานกันอย่างกว้างขวางในการรักษาความปลอดภัยอย่างหนึ่ง คือ ไฟร์วอล (Firewall) ซึ่งในการตั้งค่าต่าง ๆ ให้กับไฟร์วอล นั้นเป็นเรื่องที่สลับซับซ้อน ซึ่งในบางครั้งที่ผู้ทำการกำหนดและตั้งค่าต่าง ๆ เหล่านั้นอาจจะไม่ทราบได้ว่ามีข้อบกพร่องตรงส่วนใดบ้างหรือไม่ สามารถทำงานได้ตามที่กำหนดไว้จริงหรือไม่ ซึ่งหากเกิดข้อบกพร่องอย่างใดอย่างหนึ่งขึ้นก็อาจจะทำให้เครือข่ายขององค์กรนั้นมีประสิทธิภาพในการรักษาความปลอดภัยลดลงจนถึงขั้นโดนบุกรุกจากผู้ไม่ประสงค์ดีได้ เช่น การทำ Denial of Services เป็นต้น

ลักษณะของโปรแกรมที่ออกแบบนั้น จะมีการแบ่งการทำงานออกเป็นสองส่วน ซึ่งจะติดตั้งไว้ระหว่างไฟร์วอล ส่วนแรกนั้นจะวางไว้ที่หน้าไฟร์วอล คือ ไว้ส่วนที่ส่งข้อมูลมาจากเครือข่ายภายนอกโดยผ่านไฟร์วอลเข้ามา ส่วนที่สองนั้นจะเปรียบเสมือนเครื่องที่อยู่ภายในเครือข่ายซึ่งมีไฟร์วอลเป็นตัวป้องกันอยู่ หลักการก็คือ จะเปรียบเสมือนเครื่องทั้งสองเครื่องมีการติดต่อระหว่างกันผ่านทางไฟร์วอล ซึ่งจะมีการจำลองรูปแบบการส่งข้อมูล การขอการเชื่อมต่อ เพื่อที่จะตรวจสอบดูว่าไฟร์วอลนั้นยอมให้การขอการเชื่อมต่อไปยังบริการใดๆบ้างและจะได้รู้การทำงานของไฟร์วอล

Firewall Policy Analysis Tool

Mr.Pattadon	Khamanarong	42010224
Mr.Pitcha	Tyoviriyakul	42010229
Mr.Thana	Hongsuwan	Advisor
Mr.Akkradach	Watcharapupong	Advisor

ABSTRACT

Nowadays,A network connection is important in every Organization. In each organization would like to have security in own's network. So the use of device to increase security for network. One of the most popular security device is firewall. To configure parameters in firewall become so complicate. Administrator who has to control or configure these parameters can not know whether system can run in a good performance or there are some errors in system. One error can decrease security performance of system and can be attacked from the outsider such as Denial of Services ,etc.

This program will to distribute in 2 part . A first part will setup in front of firewall. It's will be create Packet traffic in many format. And another part will setup in backyard firewall. It's will capture all packet that pass into an interface card. This program will be to create virtual connection in a network. For administrator will check performance of firewall.

กิตติกรรมประกาศ

โครงงานนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลายๆฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้โครงการนี้เสร็จลงได้ก็คือ อาจารย์ อัครเดช วัชระภูพงษ์ และ อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษาโครงการ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก ขอบพระคุณศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยี ซึ่งโครงงานโปรแกรมวิเคราะห์นโยบายไฟร์วอล ได้รับทุนอุดหนุนโครงการแข่งขันพัฒนาโปรแกรมคอมพิวเตอร์แห่งประเทศไทย ครั้งที่ 5 ประเภทโปรแกรมเพื่องานการพัฒนาด้านวิทยาศาสตร์และเทคโนโลยี และขอกราบขอบพระคุณมา ณ ที่นี้

พัทธดนย์ ขมะณะรงค์

พิชชา เตียววิริยะกุล

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญ และที่มา

เมื่อมีการติดต่อสื่อสารบนเครือข่ายมีความสำคัญมากขึ้น ความต้องการความปลอดภัยบนเครือข่ายจึงมีมากขึ้นเช่นกัน จึงมีความคิดที่จะพัฒนาเครื่องมือที่สามารถช่วยในการตรวจสอบการทำงานของ Policy Firewall เพื่อที่จะสามารถทำให้รู้ถึงการทำงานของไฟร์วอลว่าสามารถทำงานได้ตามที่ต้องการหรือไม่ สามารถป้องกันการโจมตีแบบต่างๆได้หรือไม่ และสามารถตรวจสอบความผิดปกติของข้อมูลที่ส่งผ่านมาได้หรือไม่ เพื่อที่จะสามารถป้องกันความเสียหายจากการทำงานที่ผิดพลาดของไฟร์วอลได้อย่างทันเวลา

1.2 วัตถุประสงค์

- เพื่อศึกษาการทำงานของไฟร์วอล (Firewall)
- เข้าใจการทำงานและการเขียนแอพพลิเคชั่นบนระบบปฏิบัติการลีนุกซ์
- เพื่อเข้าใจการส่งข้อมูลบนเครือข่ายโดยใช้โพรโตคอล TCP/IP
- เพื่อเข้าใจการตั้งค่าการทำงานต่างๆ ของไฟร์วอล

1.3 เป้าหมายของโครงงาน

- เพื่อสร้างเครื่องมือที่ใช้ตรวจสอบการทำงานของไฟร์วอลว่าสามารถทำงานได้ตามที่ต้องการหรือไม่
- เพื่อช่วยให้ผู้ที่ทำการติดตั้งและตั้งค่าต่างๆของไฟร์วอลนั้นสามารถตรวจสอบได้ว่ามีจุดบกพร่องในด้านใดบ้าง การตั้งค่าต่างๆในการใช้งานนั้นทำได้อย่างครบถ้วนแล้วหรือไม่ ตลอดจนตรวจสอบว่าไฟร์วอลสามารถทำงานได้ตามค่าที่ตั้งไว้ได้อย่างถูกต้องครบถ้วนหรือไม่
- เพื่อสร้างความมั่นใจให้กับผู้ใช้ไฟร์วอล ด้วยเพราะสามารถทำให้ผู้ใช้รู้จุดบกพร่องของการตั้งค่าไฟร์วอล จึงช่วยให้แก้ไขได้อย่างทันท่วงทีก่อนจะถูกโจมตีจากผู้ไม่ประสงค์ดีทั้งจากภายในและภายนอกเครือข่าย

1.4 ขอบเขตของโครงงาน

- โปรแกรมสามารถตรวจสอบและวิเคราะห์การตั้งค่ากฎสำคัญๆทั่วไปของไฟร์วอลได้ เช่น TCP/UDP Service ใดบ้างที่ไฟร์วอลควรปิดกั้น โดยไม่ยอมให้ใช้ได้ทั้งจากภายในและภายนอกเครือข่าย

- สามารถตรวจสอบการป้องกันการโจมตีบางอย่างได้ เช่น ตรวจสอบการป้องกันการถูกบุกรุกโดย Denial of Services (DoS), Port Scanning ,SYN Flooding เป็นต้น

1.5 วิธีการดำเนินงาน

1.5.1 ศึกษาทฤษฎีพื้นฐานของโพรโตคอลทีซีพี/ไอพี

1.5.2 ศึกษาทฤษฎีพื้นฐานของสเตตฟูลอินสเปกชันไฟร์วอล (Stateful Inspection Firewall)

1.5.3 ศึกษาทฤษฎีพื้นฐานของวิธีการเขียนโปรแกรมบนระบบปฏิบัติการยูนิกซ์

1.5.4 ศึกษาทฤษฎีพื้นฐานของแพ็กเก็ตสนิฟเฟอร์ (Packet Sniffer)

1.5.5 ศึกษาทฤษฎีพื้นฐานของโมดูลสเตตฟูลอินสเปกชั่นไฟร์วอล การป้องกันการโจมตีด้วยการ SYN Flood จุดบกพร่องของไฟร์วอลเวอร์ชันเก่า และรายละเอียดของโมดูลใหม่ที่นำเสนอ และศึกษาตารางสถานะ (State Table) ของสเตตฟูลอินสเปกชั่นไฟร์วอล เพื่อใช้ในการเขียนโปรแกรม

1.5.6 ศึกษาการโจมตีผ่านเครือข่ายแบบต่างๆ เพื่อใช้ในการจำลองการถูกโจมตีในแบบต่างๆ

บทที่2
โพรโตคอลทีซีพี/ไอพี

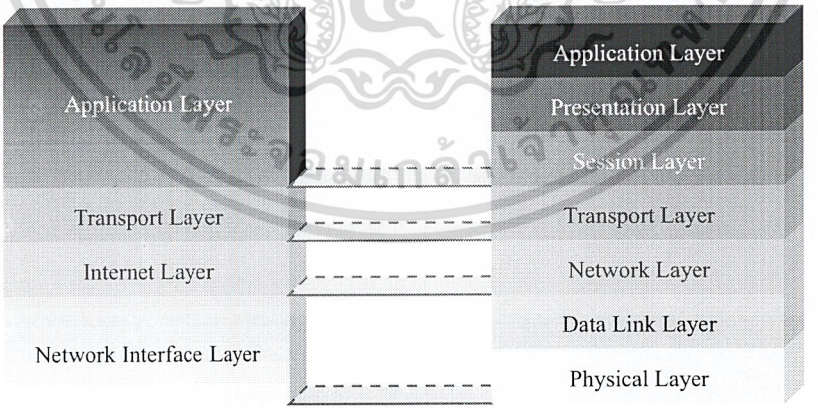
2.1 ความเป็นมาของโพรโตคอลทีซีพี/ไอพี

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

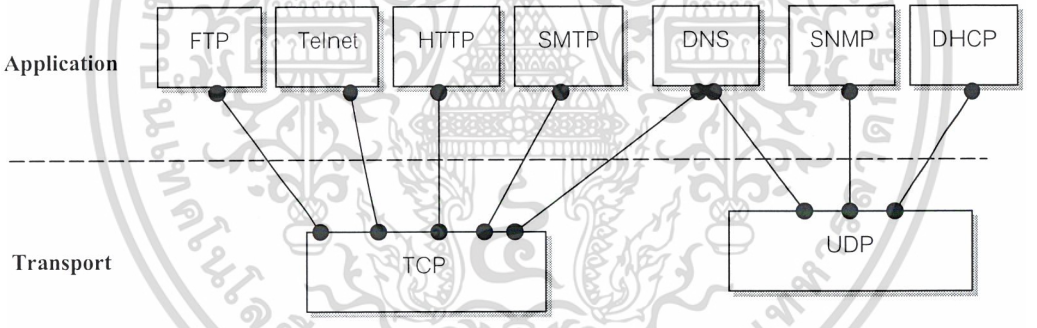


รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของทีซีพี/ไอพีกับเลเยอร์ของโอเอสไอ

ซึ่งเราจะเห็นว่าบางกลไกของโพรโตคอล ทีซีพี/ไอพี เทียบได้กับมาตรฐาน โอเอสไอ สองชั้น หรือบางกลไกก็ทำงานคาบเกี่ยวกันระหว่างบางชั้นของโมเดลอ้างอิงโอเอสไอ ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชันจนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังต่อไปนี้

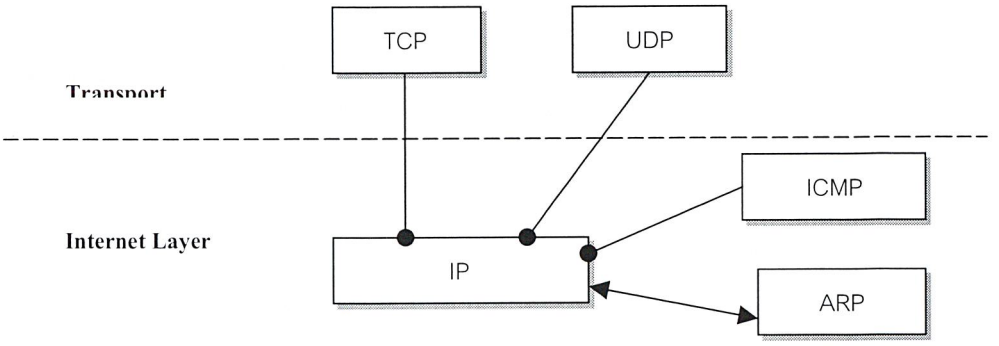
2.2.1 ชั้นแอพพลิเคชัน (Application Layer) ชั้นแอพพลิเคชันนี้รองรับการทำงานของแอพพลิเคชันต่างๆ ที่ทำงานเป็นโพรเซส อยู่ในเครื่องในเครื่องที่ให้บริการ (Sever) และเครื่องที่ขอใช้บริการ (Client) โดยจัดการเชื่อมต่อระหว่างโพรเซส หรือแอพพลิเคชันที่อยู่ต่างเครื่องกัน โดยการทำงานของแอพพลิเคชันต่างๆ มีการติดต่อกันตามแต่ละโพรโตคอลเฉพาะแล้วแต่แอพพลิเคชันที่ใช้งาน 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 โดยจุดที่เชื่อมต่อกันเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ต (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)

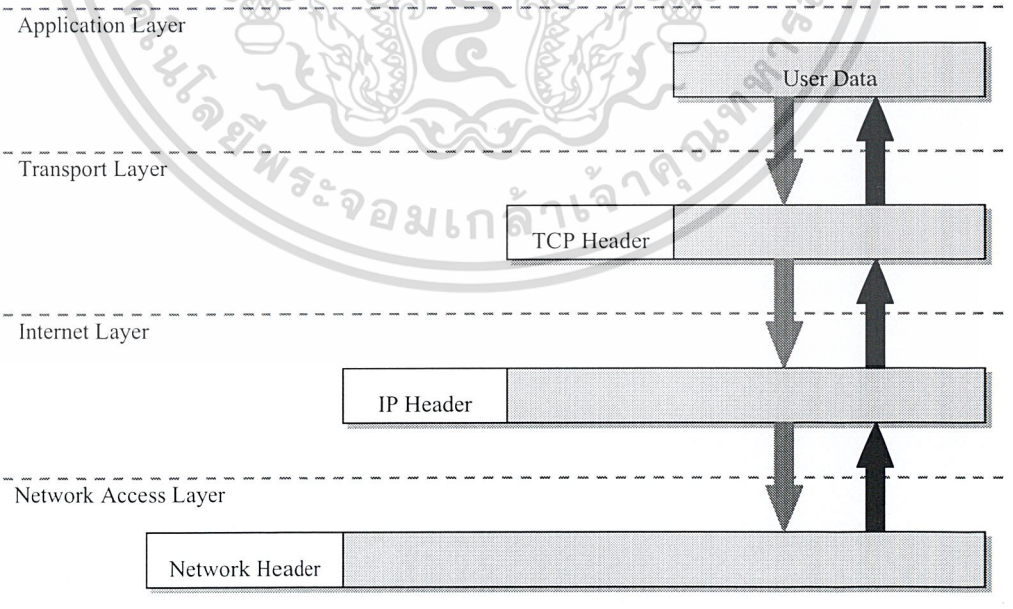


รูปที่ 2-3 โพรโตคอล TCP และ UDP อาศัยโพรโตคอล IP เพื่อส่งข้อมูลระหว่างเครือข่าย

2.2.4 ชั้นเน็ตเวิร์กอินเทอร์เฟซ (Network Interface Layer) ทำหน้าที่ในการแปลงข้อมูลให้อยู่ในรูปที่เหมาะสมกับเครือข่ายแต่ละแบบ ซึ่งแตกต่างกันออกไป และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่าย

2.3 การส่งถ่ายข้อมูลระหว่างชั้น

การส่งผ่านข้อมูลจากเครื่องที่ให้บริการไปยังเครื่องที่ข้อใช้บริการนั้น ข้อมูลจะส่งผ่านข้อมูลจากโพโตคอลที่อยู่ระดับบนสุดของเครื่องที่ให้บริการ ไปยังระดับล่างจนข้อมูลถูกแปลงให้เป็นสัญญาแล้วเดินทางผ่านเครือข่ายไปยังเครื่องขอใช้บริการ โพรโตคอลระดับล่างสุดที่เครื่องขอใช้บริการจะแปลงสัญญาที่รับมาแล้วส่งผ่านขึ้นไปยังโพรโตคอลระดับบนต่อไป



รูปที่ 2-4 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี

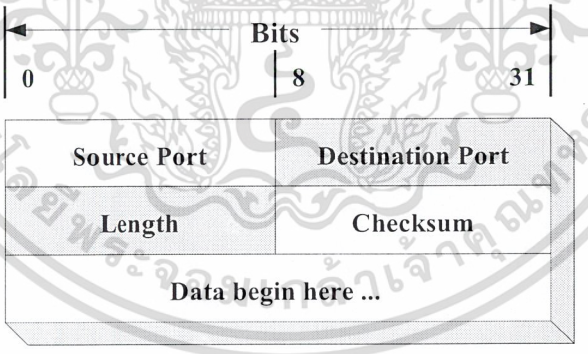
เมื่อข้อมูลผ่านแต่ละระดับชั้น โพโตคอลในแต่ละชั้นจะผนวกข่าวสารและการทำงานของโพโตคอลภายในชั้นนั้นเข้าไปเรียกว่า โพรโตคอลเฮดเดอร์ (protocol header) เข้ากับข้อมูลโพรโตคอลระดับล่างมองเฮดเดอร์ และข้อมูลเหมือนเป็นข้อมูลและเพิ่มข้อมูลของชั้นเข้าไป ข้อมูลจะถูกหุ้มเป็นชั้นๆ กระบวนการนี้เรียกว่า การเอ็นแคปซูเลต (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

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

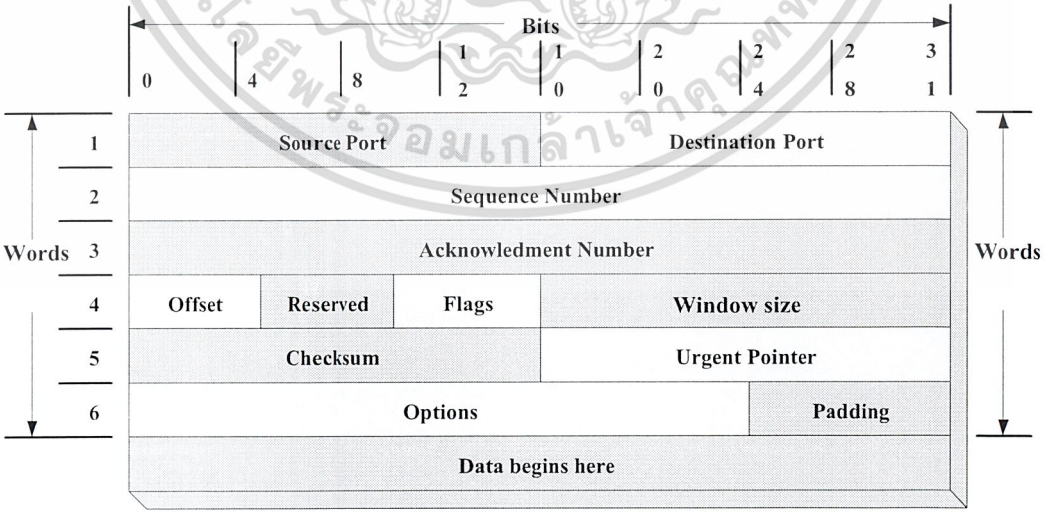
ยูดีพีเดต้าแกรมมีเฮดเดอร์ที่เรียบง่ายและมีฟิลด์จำนวนน้อย ฟิลด์ที่สำคัญมีเพียง source port และ destination port
และ checksum โดยไม่มีฟิลด์อื่นใดใช้ในการรับประกันความน่าเชื่อถือในการส่งข้อมูล

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

ส่วนประกอบของทีซีพีเฮดเดอร์

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)

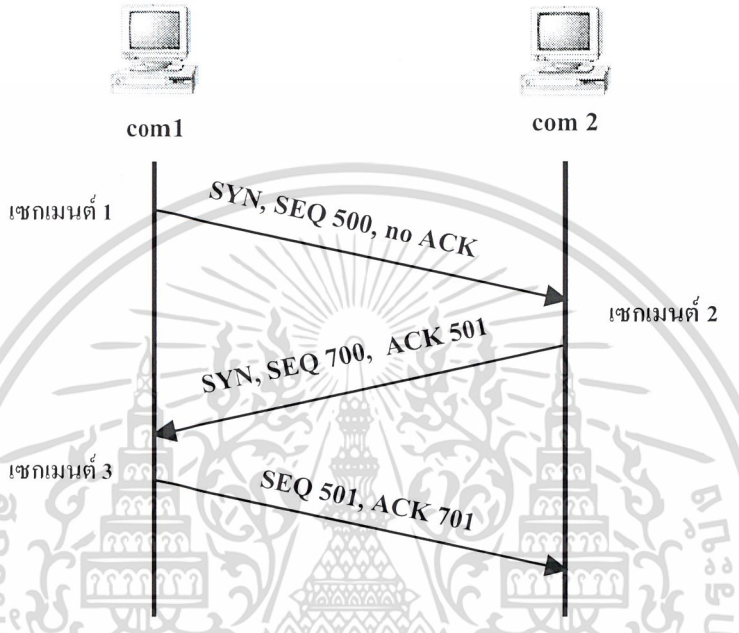
2.5.1 กลไกการทำงานของทีซีพี

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

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"

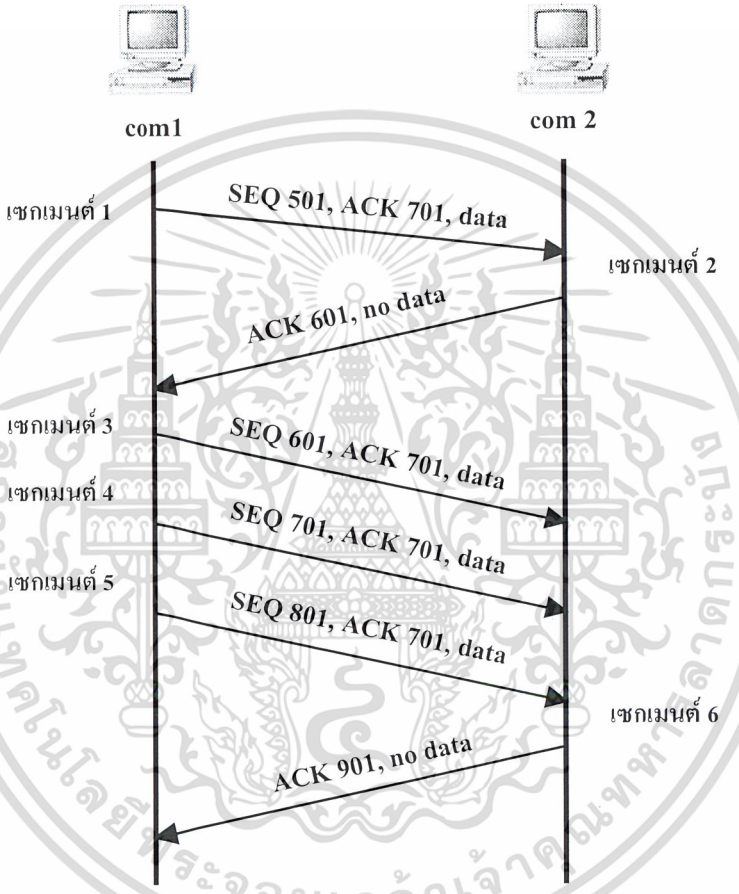
ทีซีพีของเครื่อง com2 ได้รับเซกเมนต์จาก com1 ก็จะเลือกเลขลำดับเริ่มต้นประจำตัวเช่นเดียวกันแล้วตอบกลับด้วยเซกเมนต์ 2 โดยเซ็ตแฟล็กของ SYN และ ACK ให้เป็น "1" ทั้งคู่ เพื่อแจ้งว่าได้รับเซกเมนต์1 แล้ว และเลขลำดับที่ได้รับจาก com2 บวกด้วยหนึ่ง

ทีซีพีของ com1 จะส่งเซกเมนต์ตอบรับกลับไปโดยเซ็ตแฟล็ก ACK และใช้เลขลำดับที่ได้รับจาก com2 บวกด้วยหนึ่ง

ต่อจากนั้นทีซีพีของ com1 แจ้งไปยังโพรโตคอลระดับบนว่าเชื่อมต่อแล้ว ส่วน com2 เมื่อได้รับเซกเมนต์3 ก็แจ้งขึ้นไปยังโพรโตคอลระดับบนว่าเชื่อมต่อแล้ว เมื่อสิ้นสุดนี้ การเชื่อมต่อทั้งสองด้านก็เสร็จสมบูรณ์และพร้อมรับส่งข้อมูลได้

2.5.1.2 การถ่ายโอนข้อมูล

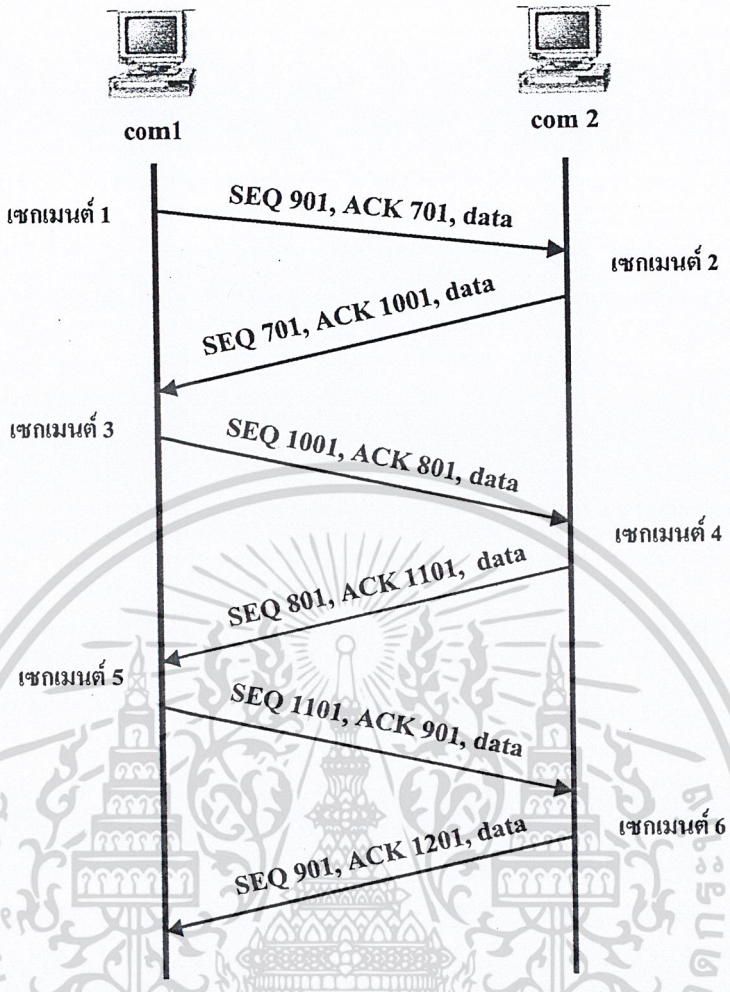
การถ่ายโอนข้อมูลเริ่มต้นหลังจากได้เชื่อมต่อเสร็จสิ้นไปแล้วรูปที่ 2-9 แสดงขั้นตอนการถ่ายโอนข้อมูลจาก com1 ไปยัง com2 ครั้งละ 100 ไบต์ จำนวน 4 ครั้ง โดยเซกเมนต์1 ที่ com1 ส่งไปยัง com2 บรรจุข้อมูลไบต์ 501 ถึง 600 และใช้เลขตอบรับ 701 เมื่อ com2 รับข้อมูลเซกเมนต์1 แล้ว ก็จะตอบกับด้วยเซกเมนต์ 1 แล้ว ก็ตอบกลับด้วยเซกเมนต์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



รูปที่ 2-9 ขั้นตอนการถ่ายโอนข้อมูล

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

ในกรณีที่ส่งข้อมูลทั้งสองทิศทางระหว่าง com1 และ com2 หลักการทั่วไปยังคงเหมือนเดิม เพียงแต่มีการตอบรับพร้อมกับส่งข้อมูลระหว่างกันดังรูปที่ 2-6



รูปที่ 2-10 การถ่ายโอนเซกเมนต์ทั้งสองทิศทาง

2.5.1.3 การยกเลิกการเชื่อมต่อ

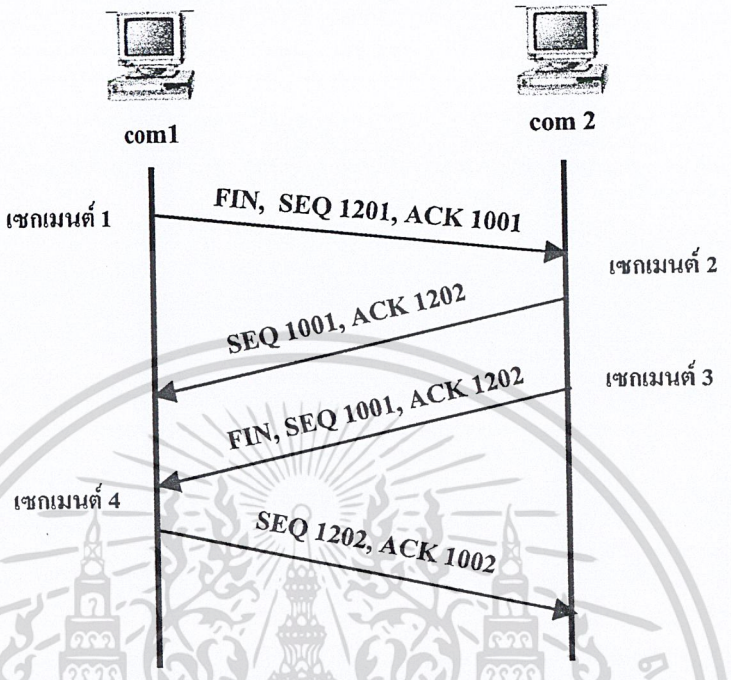
เนื่องจากการถ่ายโอนในทีซีพีเป็นแบบฟูลดูเพล็กซ์ การยกเลิกการเชื่อมต่อจึงต้องมีขั้นตอนเกิดขึ้นทั้งสองด้านดังรูปที่ ในที่นี้สมมติว่า com1 ไม่มีข้อมูลส่งอีกต่อไปจึงต้องเป็นฝ่ายขอปิดบริการเชื่อมต่อโดยโพรโตคอลประยุกต์ของ com1 จะแจ้งไปยังทีซีพีว่าส่งข้อมูลหมดแล้วถัดจากนั้นจะมีการแลกเปลี่ยนเซกเมนต์จำนวน 4 เซกเมนต์ดังนี้

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)

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 แต่ต้องมีการควบคุมปริมาณการส่งนี้ เพราะบัฟเฟอร์ฝ่ายรับย่อมมีขนาดจำกัด ฝ่ายรับต้องแจ้งให้ทางฝ่ายส่งทราบถึงขนาดข้อมูลที่รับได้โดยใช้เลขตอบรับ (ฟิลด์ Acknowledgment number) และขนาดหน้าต่าง (ฟิลด์ windows size) สองค่านี้จึงใช้เป็นตัวกำหนดขนาดการรับส่งข้อมูลแต่ละครั้ง

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

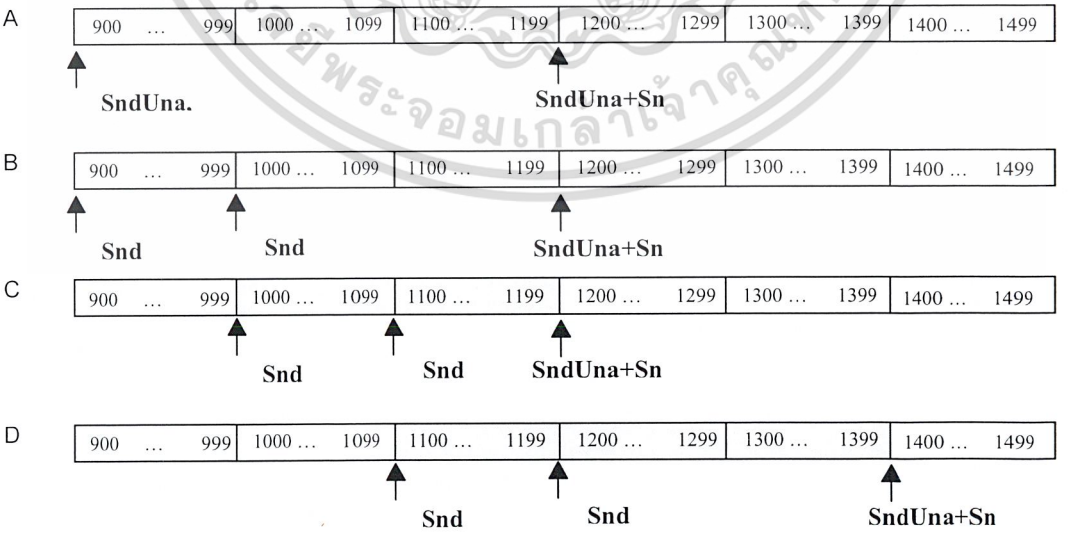
รูปที่ 2-12 ฟิลด์ในทีซีพีเฮดเดอร์ที่ใช้ควบคุมการส่งข้อมูล

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



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)

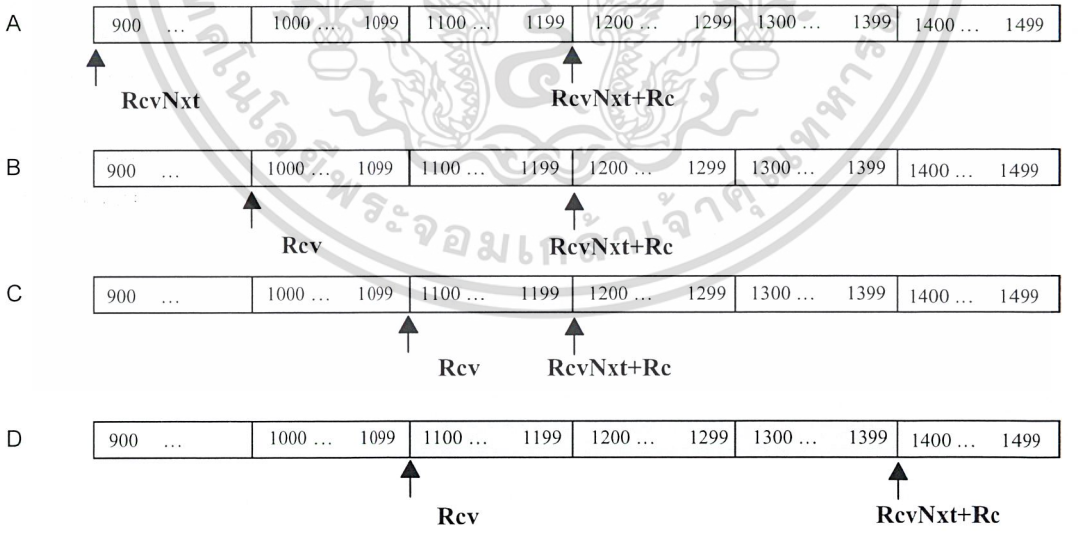
2.5.2.2 หน้าต่างฝ่ายรับ

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)

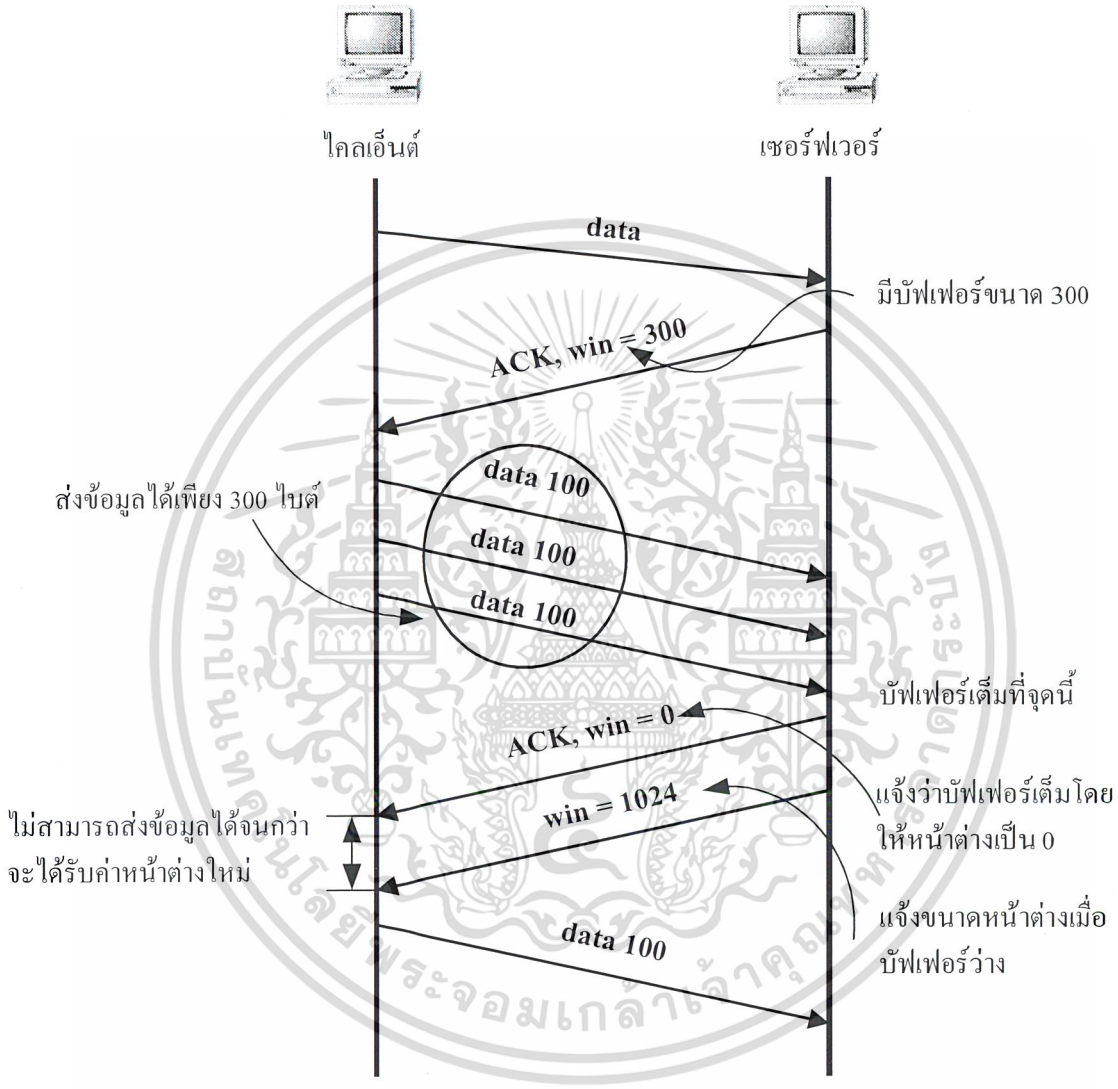
ในขั้นต่อมาหากได้รับข้อมูลอีก 100 ไบต์ ขนาดหน้าต่างก็จะลดลงเหลือ 100 ไบต์ ดังรูปที่ 2-14 (C)

ในขั้นถัดมาเมื่อโปรแกรมประยุกต์นำข้อมูลไปใช้ หน้าต่างก็เลื่อนและขยายขนาดขึ้น ดังรูป 2-14(D)



รูปที่ 2-14 การเลื่อนหน้าต่างฝ่ายรับ

ในกรณีที่บัฟเฟอร์ฝ่ายรับเต็มจนไม่สามารถรับเซกเมนต์เพิ่มได้ ทีซีพีจะส่งเซกเมนต์กำหนดขนาดหน้าต่างโดยให้ฟิลด์ window size มีค่าเท่ากับ 0 ทีซีพีอีกฝ่ายจะหยุดส่งเซกเมนต์จนกว่าจะได้รับเซกเมนต์ที่กำหนดขนาดหน้าต่างใหม่ดังรูปที่ 2-15

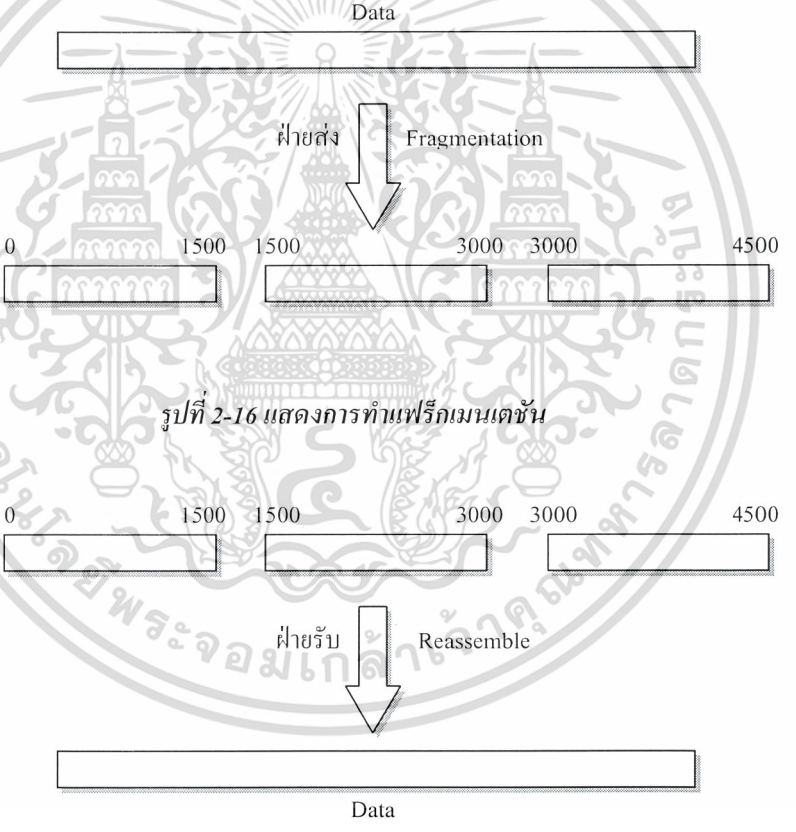


รูปที่ 2-15 การประกาศขนาดหน้าต่าง

การรอการกำหนดค่าหน้าต่างใหม่เป็นกรณีหนึ่งที่ทีซีพีต้องดำเนินการเป็นกรณีพิเศษเพราะหากเซกเมนต์ประกาศค่าหน้าต่างใหม่เกิดสูญหาย จะทำให้การสื่อสารไม่สามารถดำเนินต่อไปได้ ทีซีพีแก้ปัญหากรณีนี้โดยใช้ตัวจับเวลาเพื่อรับประกันว่าต้องส่งเซกเมนต์กำหนดค่าหน้าต่างใหม่ภายในเวลาที่กำหนด

2.6 โพรโตคอลไอพี (IP: Internet Protocol)

โพรโตคอลไอพีเป็นโพรโตคอลที่จัดการเกี่ยวกับแอดเดรสของแต่ละแพ็กเก็ต เพื่อให้ส่งแพ็กเก็ตต่างๆ ไปยังเป้าหมายได้ถูกต้อง การทำงานของไอพีเป็นเพียงการส่งข้อมูลไปยังเครื่องเป้าหมายเท่านั้น ไม่มีการส่งสัญญาณขอบริการ หรือสัญญาณให้บริการระหว่างกันเหมือนทีซีพี เรียกว่าการเชื่อมต่อแบบ Connectionless ซึ่งระบบทั้งสองตั้งสมมติฐานว่าการเชื่อมต่อระหว่างกันไม่มีความผิดพลาดเกิดขึ้นแน่

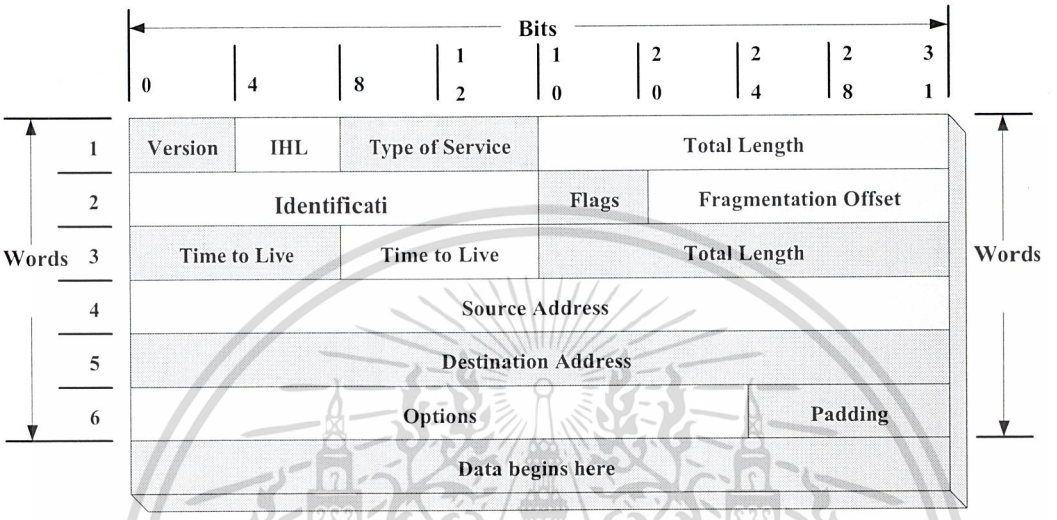


รูปที่ 2-16 แสดงการทำแฟร็กเมนเตชัน

รูปที่ 2-17 แสดงการรีแอสเซมเบิ้ล

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

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



รูปที่ 2-18 แสดงแพ็กเก็ตไอพี

ส่วนประกอบของแพ็กเก็ตไอพี

1. **version** : เป็นค่าตัวเลข 4 บิต บอกเวอร์ชันของมาตรฐานไอพีที่ใช้ โดยปกติมีค่าเป็น 4 ซึ่งหมายถึง IPv4
2. **Internet Header Length (IHL)** : เป็นตัวบอกความยาวเฮดเดอร์ของไอพี
3. **Type of Service** : เป็นส่วนที่บอกการทำงานของแพ็กเก็ตที่ส่งว่าทำหน้าที่อะไร มีทั้งหมด 8 บิต โดย

Bit 0-2 : บอกรายละเอียดการทำงานของแพ็กเก็ตนั้นๆ

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

Bit 3 : บอกถึงลักษณะของดีเลย์

0 = Normal Delay - มีดีเลย์ปกติ

1 = Low Delay - มีดีเลย์ต่ำ

Bit 4 : บอกถึงประเภทของทรูพุต

0 = Normal Throughput – มีทรูพุตปกติ

1= High Throughput – มีทรูพุตสูง

Bit 5 : บอกถึงประเภทของความน่าเชื่อถือ

0 = Normal Reliability - มีความน่าเชื่อถือพอประมาณ

1 = High Reliability – มีความน่าเชื่อถือสูง

Bit 6-7 : กันไว้ใช้ในอนาคต

4. **Total Length** : มีขนาด 16 บิต บอกถึงความยาวในดาต้าแกรมของไอพี

5. **Identification field** : เป็นตัวเลข 16 บิตเป็นค่าประจำตัวของไอพีนั้น 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

2.6.1 กลไกการทำงานของไอพี

หน้าที่หลักของโพรโตคอลไอพีคือจัดขนาดข้อมูลให้พอเหมาะและเลือกเส้นทางที่เหมาะสมเพื่อจัดส่งดาต้าแกรม ไอพีมีรูปแบบการจัดส่งดาต้าแกรมเป็นแบบ “unreliable” และ “connectionless”

ความหมายของ “unreliable” คือไอพีไม่มีกลไกรับประกันว่าดาต้าแกรมที่ส่งไปถึงปลายทางได้สำเร็จหรือไม่ หากมีความผิดปกติเกิดขึ้น ในระหว่างการส่งดาต้าแกรมไอพีทิ้งดาต้าแกรมนั้นไปแล้วรายงานสาเหตุของปัญหากลับ ไปด้วยโพรโตคอลไอซีเอ็มพี

ความหมายของ ”connectionless” ไอพีไม่มีการกำหนดเส้นทางลำเลียงระหว่างต้นทางและปลายทาง ไอพี ไม่เก็บสถานะใดๆ ของดาต้าแกรมที่ส่งออกไป ดาต้าแกรมแต่ละชิ้น จึงเป็นอิสระจากกัน โดยมีโอกาสไปถึงปลายทางโดยไม่เรียงลำดับ

2.6.2 การประมวลดาต้าแกรม

ไอพีดาต้าแกรมสร้างขึ้นโดย โฮสต้นทาง เมื่อถูกลำเลียงส่งไปนอกเครือข่ายอาจต้องเดินทางผ่านเราเตอร์หลายตัวจนกระทั่งถึงปลายทาง เราเตอร์และโฮสต์จะดำเนินการกับดาต้าแกรมโดยกรรมวิธีเฉพาะตัว

2.6.2.1 การประมวลดาต้าแกรมที่เราเตอร์

เมื่อได้รับดาต้าแกรมเราเตอร์จะตรวจสอบความถูกต้องของผลรวมตรวจสอบ จากนั้นทำการตรวจสอบเฮดเดอร์ตั้งแต่รุ่น ขนาดเฮดเดอร์ ขนาดดาต้าแกรมรวม และชนิดโพรโตคอล ว่าผิดปกติหรือไม่ จากนั้นก็ลดค่า ในฟิลด์ TTL ลงหนึ่งค่า เราเตอร์ จะกำจัดดาต้าแกรมทิ้ง ถ้าหากตรวจพบปัญหา ดังต่อไปนี้ ผลรวมตรวจสอบไม่ถูกต้อง พารามิเตอร์ในส่วนหัวที่ตรวจสอบผิดปกติหรือ ค่า TTL มีค่าเป็น 0 รวมทั้งกรณีที่บัฟเฟอร์ในเราเตอร์มีขนาดไม่เพียงพอ ในจัดการกับดาต้าแกรมนั้น

ถัดจากนั้นเราเตอร์ ทำการตรวจสอบออปชันดังนี้คือ ออปชันกำหนดเส้นทาง ชนิดบริการ และการแบ่งแฟกเมนต์ ในขั้นนี้ดาต้าแกรมอาจถูกกำจัดทิ้งอีกหากการกำหนดเส้นทางในออปชัน ไม่สามารถดำเนินการได้เพราะไม่พบเราเตอร์ที่กำหนด หรือจำเป็นต้องแฟล็กแฟรกเมนต์ไม่ได้อนุญาติให้ทำ

ขั้นถัดไปเป็นการปรับค่าในฟิลด์ต่างๆได้แก่ค่า TTL ค่าในฟิลด์ออปชัน ค่าที่จำเป็นหากมี การแฟรกเมนต์ และท้ายสุดเราเตอร์จะคำนวณผลรวมการตรวจสอบเพื่อใส่ในฟิลด์ผลรวมตรวจสอบก่อนที่จะนำส่งต่อไป

2.6.2.2 การประมวล ดาต้าแกรมที่โฮสต์ปลายทาง

เมื่อโฮสต์ปลายทางได้รับดาต้าแกรมแล้วจะเริ่มตรวจผลรวมตรวจสอบ จากนั้นทำการตรวจความถูกต้องของเฮดเดอร์ตั้งแต่ รุ่น ขนาดเฮดเดอร์ ขนาดดาต้าแกรม และชนิดของโพรโตคอลว่าผิดปกติหรือไม่ โฮสต์จะกำจัดดาต้าแกรมทิ้งหากพบว่าค่าใดค่าหนึ่งผิดปกติและแจ้งข้อผิดพลาดกลับไปด้วยไอซีเอ็มพี

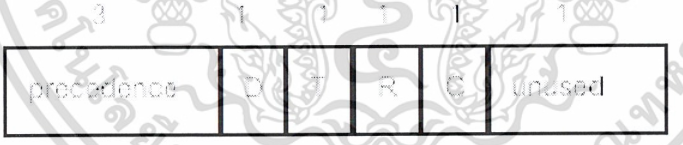
หากโฮสต์พบว่าเป็นดาต้าแกรมที่แฟรกเมนต์ โฮสต์ ต้องตรวจสอบว่าเป็นแฟรกเมนต์ที่อยู่ในชุดเดียวกันโดยตรวจฟิลด์เลขประจำตัว แอดเดรสต้นทางและปลายทาง โพรโตคอล และตรวจแฟรกเมนต์ออปเซตว่าถูกต้องหรือไม่

โฮสต์ตั้งเวลารับดาต้าแกรมให้ครบชุด ภายในเวลาในเวลาที่กำหนดตั้งแต่แฟรกเมนต์แรกเดินทางมาถึง แฟรกเมนต์ทั้งหมดถูกกำจัดทิ้งไปหากไม่สามารถรับทุกแฟรกเมนต์ในชุดนั้นภายในเวลาที่กำหนด

2.6.3 ชนิดการให้บริการ

ในรูปที่ 2-19 เป็นฟอร์แมต ของฟิลด์ 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 2-19 ฟอร์แมตของฟิลด์ TOS

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

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

บิต **T** มีค่า "1" หมายถึงต้องการเส้นทางที่มีความสามารถ ส่งผ่าน ข้อมูล ได้ในปริมาณมากๆ 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

2.6.4 การแบ่งขนาดข้อมูล

เมื่อไอพีได้รับข้อมูลจากโพรโตคอลระดับบนและเลือกอินเทอร์เฟซ ที่ส่งข้อมูลออก ก็ทำการตรวจเอ็มทียูประจำอินเทอร์เฟซ หากพบว่าดาต้าแกรมมีขนาดใหญ่เกินกว่าเอ็นทียู ไอพีจะแบ่งดาต้าแกรมออกเป็นชิ้นย่อยให้ดาต้าแกรมมีขนาดพอเหมาะกับเอ็นทียูวิธีนี้เรียกว่า แฟรกเมนต์ (fragmentation) ดาต้าแกรมที่ถูกแฟรกเมนต์จะมีเฮดเดอร์ประจำตัวเอง โดยมีข้อมูลเพิ่มเติมกำหนดลักษณะของแฟรกเมนต์ประกอบไปด้วย แต่ละชิ้นของแฟรกเมนต์เป็น ดาต้าแกรมที่สมบูรณ์ในตัวเอง เราเตอร์ระหว่างทางจะไม่รวมดาต้าแกรม (reassembly) แต่จะปล่อยให้เป็นหน้าที่ของเราเตอร์ปลายทาง การแฟรกเมนต์ดาต้าแกรมของไอพีใช้ฟิลด์ 3 ฟิลด์กำหนดข่าวสารเกี่ยวกับการแฟรกเมนต์ดังต่อไปนี้

identification ขนาด 16 บิต : หมายเลขประจำชิ้นดาต้าแกรม หากต้องมีการแฟรกเมนต์ก็ใช้เป็นค่าที่บอกว่าเป็นดาต้าแกรมชุดเดียวกัน สถานีที่สร้างดาต้าแกรมจะเพิ่มค่านี้ครั้งละหนึ่งให้แต่ละดาต้าแกรมแต่ละชุดที่ส่งออกไป

flag ขนาด 3 บิต : ใช้ควบคุมและแสดงรูปแบบแฟรกเมนต์ แบ่งออกเป็น 3 บิต

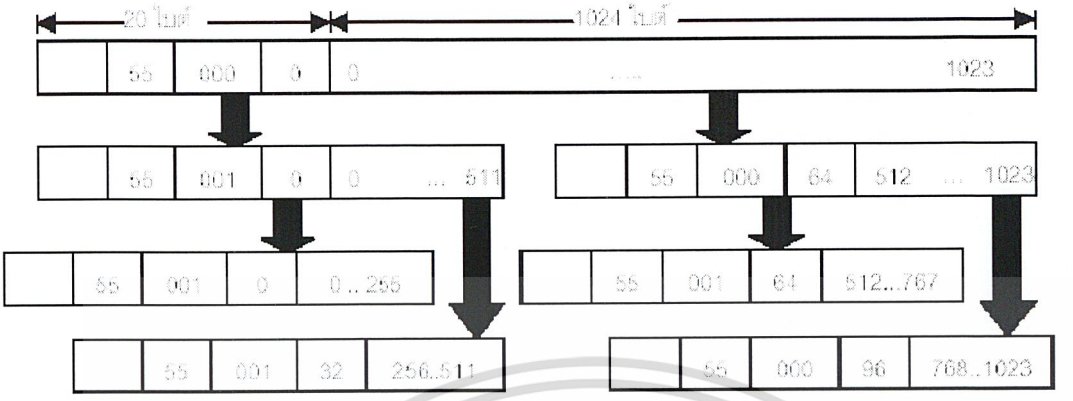
บิตแรก ไม่ได้ใช้งานและมีค่าเป็น 0 เสมอ

บิต D ใช้กำหนดว่า ให้มีแฟรกเมนต์ได้หรือไม่ หากมีค่า "0" หมายถึงให้เราเตอร์แฟรกเมนต์ได้ตามความจำเป็น หากมีค่าเป็น "1" หมายถึงบังคับไม่ให้แฟรกเมนต์ ถ้าเราเตอร์ระหว่างทางจำเป็นต้องแฟรกเมนต์แต่ถูกบังคับไม่ให้แฟรกเมนต์ด้วยบิตนี้ เราเตอร์ทำการทิ้งดาต้าแกรมนั้นไปและแจ้งความผิดพลาดกลับไปยังต้นทาง

บิต M หากเป็น "0" หมายถึง เป็นดาต้าแกรมแฟรกเมนต์สุดท้าย หากเป็น "1" หมายถึงยังมีแฟรกเมนต์ตามมาอีก

flagment offset ขนาด 13 บิต : เราเตอร์ทำการแฟรกเมนต์ข้อมูลออกเป็นบล็อกแต่ละบล็อกต้องเป็นจำนวนเท่าของ 8 ไบต์ ฟิลด์นี้ใช้บอกตำแหน่งออฟเซ็ตของข้อมูลในแต่ละแฟรกเมนต์อ้างอิงข้อมูล

2.6.4.1 การแฟรกเมนต์



รูปที่ 2-20 การแฟรกเมนต์

รูปที่ 2-20 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

เมื่อดาต้าแกรมขนาด 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

2.6.4.2 การรีแอสเซมบลี

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

ค่า total length ในแต่ละแฟรกเมนต์ของดาต้าแกรมบอกเพียงขนาดดาต้าแกรมนั้นหากแต่ละดาต้าแกรมมาถึงปลายทางโดยไม่เรียงลำดับ สถานีปลายทางไม่มีโอกาสทราบล่วงหน้าได้ว่า ดาต้าแกรมทั้งชุดมีขนาดเท่าใด ด้วยเหตุนี้สถานีปลายทางจำเป็นต้องเตรียมบัฟเฟอร์เพื่อการรีแอสเซมบลีให้เพียงพอ

2.6.4.3 ข้อคำนึงในการแฟรกเมนต์

การทำ แฟรกเมนต์ไม่ได้เกิดเฉพาะกรณีที่ต้องส่งดาต้าแกรมข้ามเครือข่ายเท่านั้น แม้แต่ในเครือข่ายเดียวกันก็ อาจจำเป็นต้องแฟรกเมนต์ดาต้าแกรมด้วยเช่นโพรโตคอลเอ็นเอฟเอส มักส่งข้อมูลครั้งละ 8,192 ไบต์ ซึ่งใหญ่กว่าเอ็นทียูในเครือข่ายแลนทั่วไป

หากสถานีปลายทางไม่สามารถรับทุกแฟรกเมนต์ได้ครบด้วยสาเหตุใดก็ตามต้องทิ้งดาต้าแกรมทั้งชุดไป ในกรณีที่ของเอ็นเอฟเอสนั้นสถานีต้นทางต้องส่งดาต้าแกรมทั้ง 8,192 ไบต์ที่ต้องทำการแฟรกเมนต์ซ้ำเช่นเดิมอีก ทำให้เพิ่มภาระในระบบเครือข่ายมากขึ้น ในเครือข่ายที่มีความผิดปกติและต้องมีการส่งแฟรกเมนต์ซ้ำจะสร้างภาระให้เครือข่ายสูงมาก

2.7 แผนภูมิสถานะทีซีพี

กระบวนการทำงานของทีซีพีซึ่งประกอบด้วยการเริ่มต้นการเชื่อมต่อ การขนถ่ายข้อมูล และการปิดการเชื่อมต่อ แต่ละขั้นตอนในกระบวนการเหล่านี้มีสถานะกำกับเพื่อกำหนดแบบแผนการทำงาน การเปลี่ยนจากสถานะหนึ่งไปสู่อีกสถานะหนึ่งขึ้นอยู่กับรูปแบบเซกเมนต์ที่ได้รับ ทีซีพีทั้งสองด้านต้องเก็บรักษาสถานะการเชื่อมโยงแต่ละส่วนไว้ตลอดเวลา

2.7.1 สถานะทีซีพี

โปรโตคอลชนิด "Connection oriented" อย่างเช่นทีซีพีมีสถานะ กำกับการทำงานแต่ละขั้นตอน สถานะของทีซีพีช่วยให้โปรโตคอลมีแบบแผนการทำงานที่ ชัดเจนและสามารถรองรับการทำงานได้หลายรูปแบบ เพื่อความเข้าใจเกี่ยวกับสถานะเหล่านี้ขอให้พิจารณาถึงแผนภูมิในรูปที่ 4-1 ประกอบคำอธิบาย แผนภูมินี้แสดงเฉพาะการเปลี่ยนแปลงสถานะตามเหตุการณ์ปกติที่เกิดขึ้นทั้งด้านทีซีพีไคลเอ็นต์และเซอร์ฟเวอร์โดยสรุปรวมไว้ในรูปเดียวกัน

2.7.1.1 การเริ่มต้นการเชื่อมต่อ

หากแยกพิจารณาสถานะทีซีพีไคลเอ็นต์ และ เซอร์ฟเวอร์ออกจากกันตามขั้นตอนการเริ่มต้นการเชื่อมต่อเพื่อจัดหมวดหมู่ให้เข้าใจง่าย สถานะของเซอร์ฟเวอร์และไคลเอ็นต์เมื่อเริ่มต้นการเชื่อมต่อ เป็นดังนี้

2.7.1.1.1 สถานะเซอร์ฟเวอร์ขั้นตอนการเริ่มต้นการเชื่อมต่อ

CLOSED สถานะจำลองที่สร้างขึ้นเพื่ออ้างอิง ในสถานะนี้ไม่มีการจัดสรรหน่วยความจำเพื่อเก็บค่าใด

LISTEN เซอร์ฟเวอร์รอคอยการขอบริการจากไคลเอ็นต์ หรือเรียกว่า เซอร์ฟเวอร์เปิดการเชื่อมต่อแบบพาสซีฟ

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

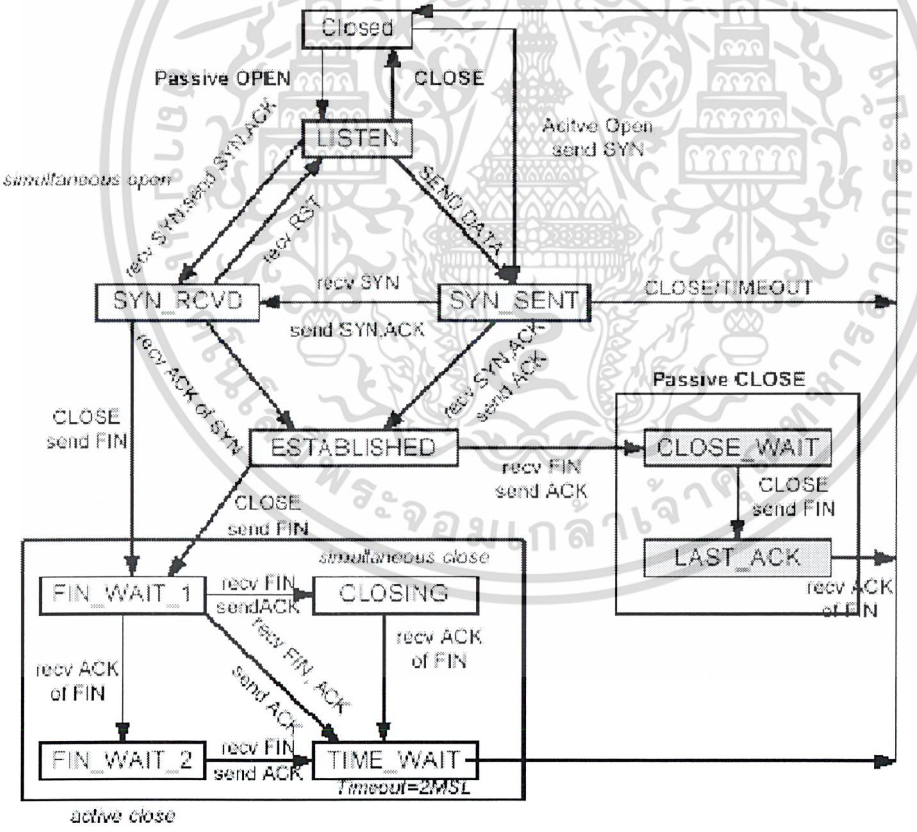
ESTABLISHED เซอร์ฟเวอร์ได้รับ ACK การสถาปนาด้านเซอร์ฟเวอร์เสร็จสมบูรณ์และพร้อมโอนถ่ายข้อมูล

2.7.1.1.2 สถานะไคลเอ็นต์ขั้นตอนการเริ่มต้นการเชื่อมต่อ

CLOSED สถานะจำลองที่สร้างขึ้นเพื่ออ้างอิง ในสถานะนี้ไม่มีการจัดสรรหน่วยความจำเพื่อเก็บค่าใด

SYN_SENT ไคลเอ็นต์ส่ง SYN และรอคอยการจับคู่เชื่อมต่อ (ตอบรับ) โดยเรียกว่าไคลเอ็นต์ขอเปิดการเชื่อมต่อแบบแอกทีฟ

ESTABLISHED ไคลเอ็นต์ได้รับ SYN/ACK จากเซอร์ฟเวอร์ และตอบกลับไปด้วย ACK การเริ่มต้นการเชื่อมต่อด้านไคลเอ็นต์เสร็จสมบูรณ์และพร้อมถ่ายโอนข้อมูล



รูปที่ 2-21 แผนภูมิสถานะทีซีพี

2.7.1.2 การปิดการเชื่อมต่อ

เมื่อเริ่มต้น การเชื่อมต่อ เสร็จสิ้นแล้ว ทั้งไคลเอ็นต์ และเซอร์ฟเวอร์ จะอยู่ ในสถานะ ESTABLISHED จนกระทั่งฝ่ายหนึ่งไม่มีข้อมูลส่งและขอปิดการเชื่อมต่อโดยส่ง เซกเมนต์ FIN การปิดการเชื่อมต่อแต่ละฝ่ายจะผ่านสถานะต่อไปนี้เป็นลำดับ

2.7.1.2.1 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

2.7.1.2.2 สถานะฝ่ายได้รับคำร้องขอปิดการเชื่อมต่อ

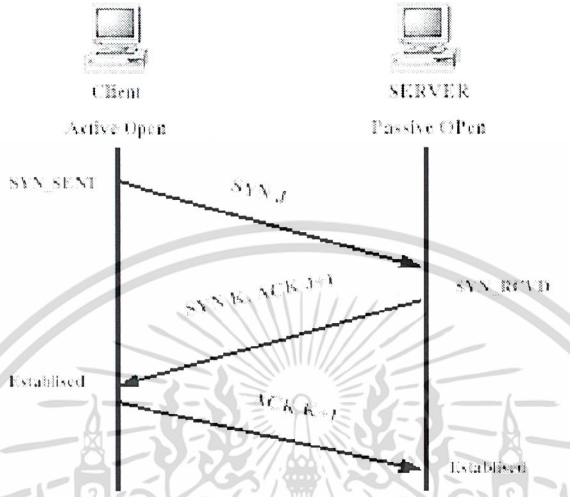
CLOSED_WAIT ได้รับ FIN ขอปิดการเชื่อมต่อ ทีซีพีส่ง ACK หากโปรแกรมประยุกต์ยังมีข้อมูลต้องนำส่งอยู่ ทีซีพีจะรอจนกว่าโปรแกรมประยุกต์สั่งปิด

LAST_ACK โปรแกรมประยุกต์สั่งปิดการส่งข้อมูล ทีซีพีส่ง FIN และคอยการตอบรับหากได้รับ ACK เมื่อใดก็จะเข้าสู่สถานะ CLOSED

CLOSED ปิดการเชื่อมต่อแล้ว ข้อมูลการเชื่อมโยงถูกกำจัดทิ้งหมด

2.7.1.3 การรับส่ง เซกเมนต์ในขั้นตอนการเริ่มต้นการเชื่อมต่อ

รูปที่ 2-22 แสดงขั้นตอนเริ่มต้นการเชื่อมต่อระหว่างไคลเอ็นต์และเซอร์ฟเวอร์คู่หนึ่ง โดยสมมติให้ไคลเอ็นต์เปิดการเชื่อมต่อแบบแอคทีฟและเซอร์เวอร์เปิดการเชื่อมต่อแบบพาสซีฟไคลเอ็นต์เริ่มต้นจากสถานะ CLOSED และเข้าสู่สถานะ SYN_SENT หลังจากส่ง เซกเมนต์ SYN เซอร์ฟเวอร์เมื่อได้ รับเซกเมนต์ SYN ก็จะตอบกลับด้วยเซกเมนต์ SYN/ACK และเปลี่ยนจากสถานะ LISTEN เข้าสู่สถานะ SYN_RCVD ไคลเอ็นต์เมื่อได้รับเซกเมนต์นี้จะส่ง เซกเมนต์ ACK กลับไปให้ เซอร์ฟเวอร์และเข้าสู่สถานะ ESTABLISHED เมื่อเซอร์ฟเวอร์ได้รั บเซกเมนต์นี้ ก็จะเข้าสู่ สถานะESTABLISHED



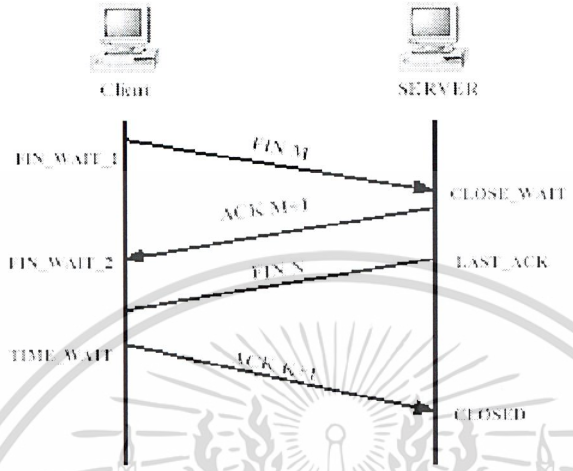
รูปที่ 2-22 สถานะการเริ่มต้นการเชื่อมทีซีพี

2.7.1.4 การรับส่ง เซกเมนต์ในขั้นตอนการปิดการเชื่อมต่อ

หากย้อนไปพิจารณารูปที่ 2-21 จะพบว่าไคลเอ็นต์และเซอร์ฟเวอร์เป็นฝ่ายเริ่มปิดการเชื่อมต่อได้ทั้งสิ้น ในรูปที่ 2-23 จะแสดงเฉพาะกรณีไคลเอ็นต์เป็นผู้ร้องขอการปิดการเชื่อมต่อดังนั้นไคลเอ็นต์จะปิดแบบแอกทีฟและเซอร์ฟเวอร์จะปิดแบบพาสซีฟ

ขั้นตอนเริ่มต้นเมื่อไคลเอ็นต์ไม่มีข้อมูลจัดส่งอีกต่อไปแล้วก็แจ้งให้ เซอร์ฟเวอร์ทรายโดยส่งเซกเมนต์ 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



รูปที่ 2-23 สถานะการปิดการเชื่อมต่อ

สถานะ TIME_WAIT เป็นสถานะรอคอยเพื่อเข้าสู่การปิดการเชื่อมต่อ ซ็อกเก็ตประจำการเชื่อมต่อนั้นไม่สามารถใช้ได้จนกว่าจะพ้นช่วงเวลานี้ไป หากมี เซกเมนต์ใดๆที่เพิ่งมาถึงในช่วงเวลานี้ (เนื่องจากอาจถ่วงเวลาอยู่) และมี ซ็อกเก็ตที่อยู่ในสถานะรอคอย เซกเมนต์นั้นจะถูกกำจัดทิ้ง ระยะเวลาในสถานะนี้มีค่าเป็นสองเท่าของค่าช่วงชีวิตนานที่สุดของเซกเมนต์จึงมักเรียกสถานะนี้ว่า “2MSL Wait state” ค่าช่วงชีวิตนานที่สุดของเซกเมนต์ที่กำหนดใน RFC 793 คือ 120 นาที ดังนั้น 2MSL จึงมีค่า 240 วินาที (ในบางระบบปฏิบัติการอาจใช้ค่า MSL เพียง 30 วินาที หรือ 1 นาที)

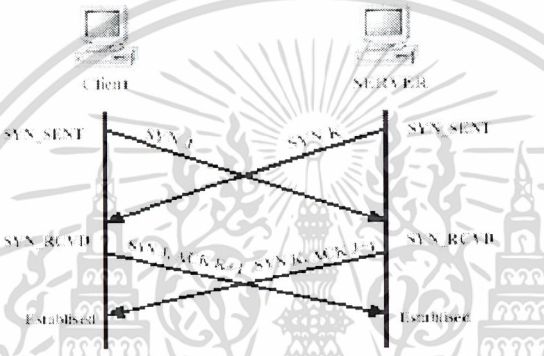
2.7.2 รูปแบบการเปิดและปิดการเชื่อมต่ออื่น

การเปิดและปิดการเชื่อมต่อที่ อธิบายก่อนหน้านี้เป็นรูปแบบที่เกิดขึ้นโดยปกติระหว่างไคลเอ็นต์และเซอร์ฟเวอร์ แต่ไคลเอ็นต์ และเซอร์ฟเวอร์อาจเข้าสู่สถานการณ์บางรูปแบบตามรูปที่ 2-21 คือ การเปิดพร้อมกัน (Simulations open) และ การปิดพร้อมกัน (Simulations close) นอกจากนี้ยังมีลักษณะการเชื่อมโยงอีก 2 แบบคือ การเปิดครึ่งฝ่าย (half-open) และ การปิดครึ่งฝ่าย (half-close)

2.7.2.1 การเปิดพร้อมกัน

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

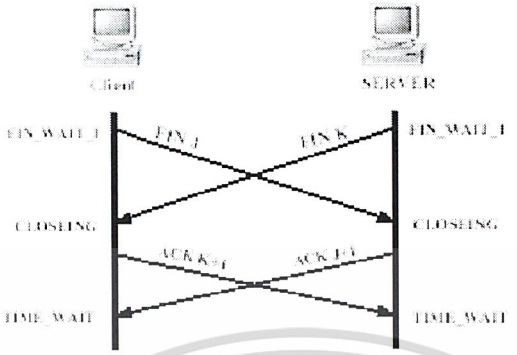
การเปิดพร้อมกันมี เซกเมนต์แลกเปลี่ยนจำนวน 4 เซกเมนต์ดังรูปที่ 2-24 โดยไม่มีการกำหนดว่าฝ่ายใดเป็นไคลเอ็นต์หรือเซอร์ฟเวอร์ทั้งสองด้านเปิดการเชื่อมต่อแบบพาสซีฟ ระหว่างกันและมีการเชื่อมโยงทางทีซีพีสองส่วน ในขณะที่การเปิดพร้อมกันจะมีการเชื่อมโยงทางทีซีพีส่วนเดียว



รูปที่ 2-24 การเปิดพร้อมกัน

2.7.2.2 การปิดพร้อมกัน

การปิดการเชื่อมต่อตามปกติเริ่มต้นจากผ่านหนึ่งซึ่งอาจเป็นไคลเอ็นต์หรือเซอร์ฟเวอร์ขอปิดการเชื่อมต่อแบบแอกทีฟโดยส่ง เซกเมนต์ FIN ไปยังอีกฝ่ายหนึ่ง แต่หากทั้งสองฝ่ายต่างขอปิดแบบแอกทีฟพร้อมกันจึงเรียกว่าการปิดพร้อมกัน



รูปที่ 2-25 การปิดพร้อมกัน

รูปที่ 2-25 แสดงการปิดพร้อมกัน เริ่มจากทั้งสองฝ่ายอยู่ในสถานะ ESTABLISHED และต่างส่งเซกเมนต์ FIN ให้อีกฝ่ายหนึ่ง และเข้าสู่สถานะ FIN_WAIT_1 เช่นกัน จำนวนเซกเมนต์ที่ใช้ในการปิดพร้อมกันยังคงเท่ากับการปิดตามปกติ แต่ลำดับการโต้ตอบเซกเมนต์จะแตกต่างกัน

2.7.2.3 การเปิดครึ่งฝ่าย (half open)

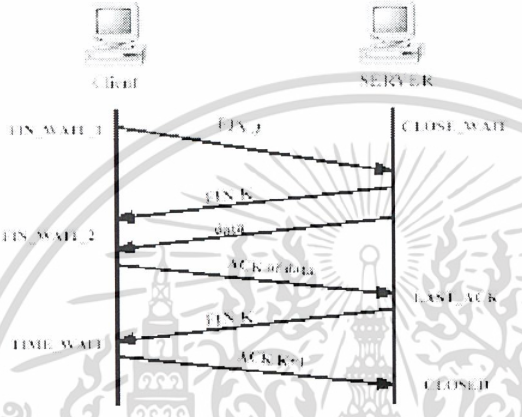
หากทีซีพีฝ่ายใดฝ่ายหนึ่งหยุดการทำงานโดยอีกฝ่ายหนึ่งไม่ทราบ ทีซีพีอีกฝ่ายที่ยังทำงานแบบเปิดครึ่งฝ่าย (half-open) โดยปกติแล้วกรณีเช่นนี้มักเกิดจากโฮสต์ฝ่ายหนึ่งเกิดปัญหากระทั่งต้องปิดเครื่องในระหว่างนั้นหากไม่มีหารส่งข้อมูลใด อีกฝ่ายหนึ่งจะยังคงอยู่ในสถานะ ESTABLISHED และรอคอยการรับส่งข้อมูลอยู่ ตัวอย่างเช่นไคลเอ็นต์ขอใช้บริการเทลเน็ตจากเซอร์ฟเวอร์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

เมื่อไคลเอ็นต์ ส่ง เซกเมนต์ไปให้เซอร์ฟเวอร์ เซกเมนต์นั้นจะไม่มี ซ็อกเก็ตใดรองรับกล่าวอีกนัยหนึ่งคือเซอร์ฟเวอร์ไม่สามารถแยกแยะได้ว่าเซกเมนต์นั้นเป็นของการเชื่อมโยงใด สิ่งที่เซอร์ฟเวอร์ดำเนินการก็คือส่งเซกเมนต์รีเซ็ต โดยเซ็ตแฟล็ก RST เพื่อแจ้งให้ไคลเอ็นต์ ยกเลิกการเชื่อมต่อนั้น

2.7.2.4 การปิดครึ่งฝ่าย (half close)

เนื่องจากการเชื่อมโยงทีซีพี เป็นแบบฟูลดูเพล็กซ์ ข้อมูลสามารถเดินทางได้ระหว่างไคลเอ็นท์และเซอร์ฟเวอร์ทั้งสองทิศทางอย่างเป็นอิสระ แต่ละฝ่ายจึงสามารถปิดตัวเองลงเมื่อไม่มีข้อมูลใดต้องส่งอีกต่อไป โดยยังสามารถรับข้อมูลจากอีกฝ่ายหนึ่งได้ การเชื่อมโยงลักษณะนี้เรียกว่า ปิดครึ่งฝ่าย (halfclose)

ตัวอย่างในรูปที่ 2-26 ไคลเอ็นต์ขอปิดการเชื่อมต่อโดยส่งเซกเมนต์ FIN และเข้าสู่สถานะ FIN_WAIT_1 เมื่อเซอร์ฟเวอร์ได้รับเซกเมนต์ FIN และตอบรับ (เซกเมนต์ ACK J+1) แล้วก็จะเข้าสู่สถานะ CLOSE_WAIT และเมื่อไคลเอ็นต์ได้รับเซกเมนต์ตอบรับก็จะเข้าสู่สถานะ FIN_WAIT_2 และเตรียมปิดการเชื่อมต่อ ในจังหวะนี้แอพลิเคชั่นด้านเซอร์ฟเวอร์อาจยังมีข้อมูลที่ต้องส่งจึงยังไม่ปิดการเชื่อมต่อ ไคลเอ็นต์ปิดการส่งแต่ยังคงต้องรับข้อมูลจากเซอร์ฟเวอร์อยู่



รูปที่ 2-26 การปิดครึ่งเดียว (half close)

ขั้นตอนถัดไปตามรูปที่ 2-26 สมมติว่าเซอร์ฟเวอร์มีข้อมูลต้องส่งเพียงเซกเมนต์เดียว (เซกเมนต์ data) เมื่อเซอร์ฟเวอร์ได้รับเซกเมนต์ตอบรับ (เซกเมนต์ ACK of data) ก็สั่งปิดการเชื่อมโยงโดยส่งเซกเมนต์ FIN และเข้าสู่สถานะ LAST_ACK ถัดจากนั้นการปิดการเชื่อมโยงก็จะดำเนินต่อไปตามปกติ

บทที่ 3
แพ็กเก็ตแคปเจอร์
Packet Capture

3.1 ความหมายแพ็กเก็ตแคปเจอร์ (Packet Capture)

คำว่า สนิฟเฟอร์ (Sniffer) นั้นเครื่องหมายการค้าซึ่งจดทะเบียนไว้โดยบริษัท Network Associates Inc. ในสหรัฐฯ เพื่อใช้ในผลิตภัณฑ์ของตนเองชื่อ Sniffer Network Analyzer ซึ่งเป็นโปรแกรมวิเคราะห์เนตเวิร์กโดยอาศัยการดักจับข้อมูลภายในเนตเวิร์ก ต่อมาคำนี้ถูกนำมาใช้เรียกอุปกรณ์ประเภทนี้ที่นำมาใช้ในการดักอ่านข้อมูลกันอย่างแพร่หลาย จนเป็นที่เข้าใจว่าสนิฟเฟอร์เป็นเครื่องมือในการดักจับข้อมูลบนเนตเวิร์ก ซึ่งถ้าหากเรียกให้ถูกต้องอุปกรณ์ประเภทนี้ควรเรียกว่า Netwire Tapping Device

แพ็กเก็ตแคปเจอร์ คืออุปกรณ์ที่ทำหน้าที่ดักจับแพ็กเก็ตที่วิ่งอยู่บนเนตเวิร์ก โดยจะทำงานร่วมกับเนตเวิร์กอินเทอร์เฟซการ์ด (Network Interface Card) ในโหมดโพรมิสคูอัส (Promiscuous-mode) โดยทำงานร่วมกันในโหมดนี้ การทำงานของเนตเวิร์กอินเทอร์เฟซการ์ดในโหมดนี้จะไม่มีการเปรียบเทียบแอดเดรสปลายทางของแพ็กเก็ตกับแอดเดรสของเนตเวิร์กอินเทอร์เฟซการ์ด นั่นคือทุกๆแพ็กเก็ตมาถึงยังเนตเวิร์กอินเทอร์เฟซการ์ด จะถูกจับแพ็กเก็ตขึ้นมาได้ทั้งหมดไม่เลือกว่าจะระบุแอดเดรสปลายทางของการ์ดเนตเวิร์กอินเทอร์เฟซอันไหน

แพ็กเก็ตแคปเจอร์ จะถูกใช้ในการเฝ้าดูการจราจรในเครือข่าย ให้มีประสิทธิภาพและแก้ไขปัญหาต่างๆ ที่เกิดจากการทำงาน ผิดพลาดของเครือข่าย และยังเป็นเครื่องมือในการศึกษา การทำงานของโพรโตคอลร่วมถึงการดูแลความปลอดภัยของระบบ

3.2 องค์ประกอบของแพ็กเก็ตแคปเจอร์

3.2.1 ฮาร์ดแวร์ (Hardware) หมายถึง อุปกรณ์อิเล็กทรอนิกส์ที่สามารถดักอ่านสัญญาณจากเนตเวิร์กเข้ามาได้ และสามารถนำสัญญาณที่ให้ส่งต่อไปประมวลผลออกมาเป็นข้อมูลทางคอมพิวเตอร์ได้ หน้าที่หลัก คือ จัดการกับการรับข้อมูลในระดับฟิสิคัล (Physical) อุปกรณ์นี้โดยทั่วไปก็คือเนตเวิร์กอแดปเตอร์

3.2.2 ไดรเวอร์ (Driver) เป็นโปรแกรมระดับล่างที่ควบคุมการดักจับข้อมูลของฮาร์ดแวร์ไปเก็บเป็นข้อมูลดิบรอการประมวลผลในลำดับถัดไป

3.2.3 บัฟเฟอร์ (Buffer) เป็นหน่วยความจำที่ใช้พักข้อมูลจากการดักมาได้ของ Driver โดยจะทำงานการจัดเก็บเพียงชั่วคราว และหมุนเวียนข้อมูลเข้ามาเสมอ กลไกการนำข้อมูลจากไดรเวอร์มาเก็บในบัฟเฟอร์ จะเป็นตัวบอกสถานะของการดักข้อมูลได้ความเร็วสูงสุดเท่าใด หากกระบวนการนำข้อมูลไปเก็บเป็นไปอย่างล่าช้า ย่อมทำให้แพ็กเก็ตแคปเจอร์ ไม่สามารถดักข้อมูลที่อยู่บนเนตเวิร์กได้ทันและต้องปล่อยข้อมูลนั้นทิ้งไป

3.2.4 ซอฟต์แวร์ (Software) เพื่อทำหน้าที่จัดการข้อมูลที่ได้เข้าโดยการประมวลผลตามวัตถุประสงค์ของการดักอ่านข้อมูลนั้น เนื่องจากข้อมูลดิบที่ดักได้นั้นจะเป็นข้อมูลในระดับต่ำ คือ เนตเวิร์ก

อินเทอร์เฟซเลเยอร์ (Network Interface Layer) ซึ่งจะมีข้อมูลที่ยังไม่ผ่านการมัลติเพล็กซ์และการจัดรูปแบบให้เข้าใจได้ ส่วนที่ได้จะเป็นข้อมูลเลขฐานสอง 0 กับ 1 จำนวนมหาศาลที่ต้องนำมาแปลความหมาย และอีกหน้าที่หนึ่งคือ ข้อมูลที่ดักได้นั้น เป็นการสื่อสารของทุกโฮสต์ที่อยู่ภายในเนตเวิร์กนั้นร่วมกันอยู่อย่างไม่เป็นระเบียบและไม่มีการแยกแยะว่าเป็นการสื่อสารเรื่องอะไร ระหว่างโฮสต์ใดกับโฮสต์ใด การที่จะแปลความหมายข้อมูลนี้ได้ก็จำเป็นต้องมีโปรแกรมสำหรับทำหน้าที่แปลความหมายของข้อมูลให้อยู่ในรูปแบบที่สามารถเข้าใจได้มากขึ้น นั่นคือการทำหน้าที่คล้ายกับดีมัลติเพล็กซ์ของข้อมูลทุกๆโฮสต์โดยไม่สนใจว่าเป็นข้อมูลของโฮสต์ใด

หลังจากข้อมูลผ่านการดีมัลติเพล็กซ์แล้วก็จะอยู่ในรูปที่สามารถเข้าใจได้ง่ายขึ้น แต่จะเข้าใจมากน้อยขึ้นอยู่กับความสามารถในการดีมัลติเพล็กซ์ของโปรแกรม หากการดีมัลติเพล็กซ์ได้จนถึงโพรโตคอลเลเยอร์ที่สูง รวมทั้งมีการแยกแยะหมวดหมู่ของการสื่อสารของแต่ละโฮสต์ก็จะเห็นความต่อเนื่องของการสื่อสารได้ดีขึ้น

ในส่วนการจัดเก็บข้อมูลดิบที่ดักมาได้ โปรแกรมแคปเจอร์ส่วนใหญ่จะทำการเก็บข้อมูลดิบที่ดักมาได้ไว้ในฐานข้อมูลเพื่อให้สามารถนำกลับมาวิเคราะห์ในภายหลังได้ เพราะในการแคปเจอร์จะต้องคอยดักจับข้อมูลอยู่ตลอดเวลา หากทำการดีมัลติเพล็กซ์ทันทีที่ได้รับข้อมูลในแบบเรียลไทม์อาจทำให้ประสิทธิภาพของแคปเจอร์ลดต่ำลง และเนื่องจากการสื่อสารข้อมูลจะต้องอาศัยความต่อเนื่องของการเชื่อมโยงกันของหลายๆแพ็กเก็ต ประกอบการดีมัลติเพล็กซ์ส่วนใหญ่จะสามารถทำได้ก็ต่อเมื่อข้อมูลได้รับมาครบถ้วนสมบูรณ์ทั้งหมดแล้วเท่านั้น

3.3 การทำงานของแพ็กเก็ตแคปเจอร์

การทำงานของแพ็กเก็ตแคปเจอร์ อาศัยหลักการของโพรโตคอลอีเทอร์เนต ที่ใช้การกระจายของข้อมูลไปยังทุกโฮสต์ที่อยู่ในเนตเวิร์กและอาศัยโฮสต์แต่ละตัวทำหน้าที่จำแนกการสื่อสารของตนเอง นั่นหมายความว่าข้อมูลทุกแพ็กเก็ตที่ใช้สื่อสารกันได้ถูกส่งไปยังทุกโฮสต์ ซึ่งจะได้รับพร้อมกันแล้วเหมือนกัน เพียงแต่การที่จะสื่อสารกันได้อย่างถูกต้องนั้นโฮสต์แต่ละตัวจะต้องมีกระบวนการที่สามารถรู้ได้ว่าข้อมูลแพ็กเก็ตใดเป็นของตนเอง และข้อมูลแพ็กเก็ตใดมิใช่ของตัวเอง ทุกๆแพ็กเก็ตที่กระจายลงบนเนตเวิร์กนั้น จะมีหมายเลข ระบุชัดเจน คือ แมคแอดเดรส (Mac Address) หรือเรียกอีกอย่างหนึ่งว่า อีเทอร์เนต (Ethernet Address) ซึ่งจะเป็นสิ่งที่บอกว่าแพ็กเก็ตมาจากฮาร์ดแวร์ใดในเนตเวิร์ก ทำให้สามารถระบุได้ว่าส่งมาจากโฮสต์ใด

แมคแอดเดรส จะเป็นหมายเลขเฉพาะฮาร์ดแวร์ทุกชนิดที่ใช้สื่อสารโดยโพรโตคอลอีเทอร์เนต และในทางทฤษฎีแล้วฮาร์ดแวร์ทุกชนิดจะไม่มีแมคแอดเดรสซ้ำกัน โดยทั่วไปแมคแอดเดรสจะกำหนดตายตัวอยู่ในรอม (Rom) ของฮาร์ดแวร์และไม่สามารถเปลี่ยนแปลงได้ด้วยซอฟต์แวร์

การใช้งานของฮาร์ดแวร์จะต้องควบคู่กับไดร์เวอร์ของฮาร์ดแวร์นั้นๆโดยปกติแล้วไดรเวอร์จะถูกกำหนดให้ปฏิบัติตามโพรโตคอลอย่างเข้มงวดคือ

- ให้รับข้อมูลที่มีแมคแอดเดรสเป็นของตนเองเท่านั้น (ห้ามอ่านข้อมูลของผู้อื่น)
- ให้ส่งข้อมูลโดยใช้แมคแอดเดรสของตนเองเท่านั้น (ห้ามปลอมเป็นผู้อื่น)

ดังนั้นหากใช้ไดรเวอร์ตามปกติที่มากับฮาร์ดแวร์แล้วเครื่องคอมพิวเตอร์ก็จะทำงานอยู่ในระเบียบเรียบร้อย และไม่สามารถวุ่นวายกับข้อมูลของผู้อื่นได้ แพ็กเก็ตแคปเจอร์จึงจำเป็นที่ต้องใช้โหมดการทำงานพิเศษที่อนุญาตให้รับข้อมูลที่แมคแอดเดรสไม่ใช่ของตนได้โหมดนั้นเรียกว่า พรอมิสคิอัสโหมด (Promiscuous Mode) ที่ทำให้ฮาร์ดแวร์สามารถรับข้อมูลดิบทั้งหมดบนเนตเวิร์กเข้ามาได้ โดยไม่สนใจว่าข้อมูลทั้งหมดเป็นของใคร ส่งให้ใคร

จากรูปที่ 3.1 ในเนตเวิร์กมีโฮสต์อยู่ 4 ตัว ต่อรวมกันเป็นเนตเวิร์กโดยใช้ฮับ (Hub) ร่วมกันและมีโฮสต์ตัวหนึ่งรันโปรแกรม ที่ทำหน้าที่เป็นโปรแกรมแคปเจอร์เพื่อดักจับข้อมูลที่เสียบต่อรวมกับฮับนั้น จากรูปแสดงให้เห็นการกระจายข้อมูลของโฮสต์ A ต้องการส่งให้โฮสต์ B ข้อมูลได้กระจายไปยังทุกโฮสต์ที่ต่ออยู่บนฮับ รวมถึงแพ็กเก็ตแคปเจอร์ด้วย

โฮสต์ B เมื่อได้รับข้อมูลก็จะตรวจแมคแอดเดรส ถ้าพบว่าเป็นข้อมูลที่ถูกส่งมาหาตนเองก็จะทำการอ่านและส่งต่อให้ระบบปฏิบัติการประมวลผล

โฮสต์ C และ โฮสต์ D โฮสต์ทั้งสองซึ่งมีเนตเวิร์กอแคปเตอร์ทำงานอยู่ในโหมดปกติ เมื่อได้ทำการตรวจสอบแล้วเห็นว่าไม่ใช่ข้อมูลของตนเอง ก็จะไม่สนใจข้อมูลนั้น

แพ็กเก็ตแคปเจอร์ สำหรับแพ็กเก็ตแคปเจอร์นั้นมีเนตเวิร์กอแคปเตอร์ที่ทำงานอยู่ในโพมิคูอัสโหมด นั่นคือไม่สนใจว่าข้อมูลที่เข้ามาเป็นของใคร แพ็กเก็ตแคปเจอร์จะรับข้อมูลทั้งหมดแล้วนำไปเก็บในบัฟเฟอร์และฐานข้อมูล เพื่อนำไปประมวลผลเพื่อหาข้อมูลที่มี ประโยชน์ภายหลัง

จากเนตเวิร์กในภาพนั้นมีแพ็กเก็ตแคปเจอร์ถูกติดตั้งในตำแหน่งดังกล่าว ก็จะทำให้ข้อมูลทั้งหมดที่โฮสต์ทั้ง 4 สื่อสารกันโดยผ่านฮับที่ใช้งานร่วมกัน ไม่ว่าข้อมูลใดจะปรากฏที่ฮับก็สามารถถูกดักอ่านได้โดยแพ็กเก็ตแคปเจอร์ทันที

3.4 ประโยชน์จากแพ็กเก็ตแคปเจอร์

3.4.1 วิเคราะห์ระบบเครือข่าย (Network Analyzer) การที่แพ็กเก็ตแคปเจอร์สามารถดักข้อมูลทั้งหมดที่อยู่บนเนตเวิร์กได้ ทำให้เราสามารถนำข้อมูลดังกล่าวมาทำการแยะมวลผลวิเคราะห์คุณสมบัติต่างๆของเนตเวิร์กได้หลายแง่มุม ไม่ว่าการกระจายการใช้งานของโพรโตคอลต่างๆ เช่น เอชทีทีพี (HTTP) , เอสเอ็มทีพี (SMTP) , เอฟทีพี (FTP) เพื่อจะได้ทราบว่ามีการใช้เนตเวิร์กเพื่อการใด มากน้อยเพียงใด จะทำให้สามารถจัดการเนตเวิร์กได้อย่างเหมาะสม หรือในด้านการใช้งานเนตเวิร์กตามช่วงเวลาต่างๆเวลาใดที่มีผู้ใช้มาก เวลาใดที่มีผู้ใช้น้อย ผู้ใช้คนใดใช้แบนด์วิดธ์ไปในทางที่เป็นประโยชน์หรือไม่

3.4.2 เครื่องมือตรวจสอบเครือข่าย (Network Debugging Tools) ในบางครั้งการหาสาเหตุความผิดปกติของแอพพลิเคชั่นต่างๆ จำเป็นที่ต้องสืบค้นลึกลงไปถึงเนื้อข้อมูลที่รับส่งกันจริงๆบนเนตเวิร์กเพื่อหาว่าเหตุใดการทำงานของแอพพลิเคชั่นจึงไม่สามารถทำงานได้ตามปกติ การได้เห็นข้อมูลที่ส่งกันจริงทำให้การวิเคราะห์และแก้ไขปัญหาสามารถดำเนินการได้อย่างรวดเร็วและตรงต่อปัญหามากขึ้น โดยเฉพาะกรณีที่มีการใช้เครื่องมือในระดับเนตเวิร์กมาเกี่ยวข้อง เช่น มีการส่งข้อมูลผ่านไฟร์วอลแล้วมีปัญหา หรือการทดสอบ ACL (Access Control List) ของเราเตอร์ เป็นต้น

3.4.3 ตรวจจับแพ็กเก็ต (Packet Monitoring) การศึกษาเทคนิคของโพรโตคอลในระดับเนตเวิร์กจำเป็นต้องเห็นข้อมูลที่สื่อสารกันอยู่จึงจะสามารถเข้าใจได้และเห็นภาพจริง แพ็กเก็ตมอนิเตอร์ริ่งจะเป็นเครื่องมือที่นำแพ็กเก็ตมาแสดงให้เห็นในรูปแบบต่างๆได้

3.4.4 ระบบตรวจจับผู้บุกรุก (Intrusion Detection System) ระบบตรวจจับผู้บุกรุกก็มีพื้นฐานมาจากการดักอ่านข้อมูล เพียงแต่เป็นการประยุกต์การวิเคราะห์รูปแบบการบุกรุกเข้ากับการดักอ่านข้อมูลบนเนตเวิร์กแบบเรียลไทม์ ทำให้สามารถทราบว่ามีกิจกรรมใดบนเนตเวิร์กที่มีแนวโน้มว่าจะเป็นการบุกรุกหรือทำอันตรายต่อผู้อื่น

บทที่ 4
สเตตฟูลไฟร์วอล
Stateful Firewall

4.1 หลักการทำงานสเตตฟูลไฟร์วอล

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

โดยทั่วไปมักจะแยกกันโดยชัดเจนระหว่างไฟร์วอลแบบ Stateful filtering และระบบ application level proxy server ในบทนี้เราจะไม่กล่าวถึงหลักการทำงานของแอพพลิเคชันพร็อกซี (Application Proxies) ซึ่งอยู่นอกขอบเขตที่ศึกษา

4.2 การแบ่งชนิดไฟร์วอล

4.2.1 แพ็กเก็ตฟิลเตอร์ริง (Packet Filtering)

ไฟร์วอลชนิดนี้จะกรองแพ็กเก็ตโดยพิจารณาแพ็กเก็ตที่เข้าออกตามกฎที่ตั้งไว้ การตัดสินใจกรองแพ็กเก็ตเหล่านี้จะยึดข้อมูลที่อยู่ในเฮดเดอร์ของแพ็กเก็ตตัวนั้นๆ เช่น แอดเดรสต้นทาง แอดเดรสปลายทาง พอรืตหรือโพรโตคอล ผลิตภัณฑ์ไฟร์วอลบางตัวที่สามารถกรองแพ็กเก็ต ยังมีความสามารถในการกรองข้อมูลในส่วนอื่นที่ไม่ใช่แพ็กเก็ตเฮดเดอร์ด้วย เมื่อแพ็กเก็ตแต่ละแพ็กเก็ตเข้ามาสู่ไฟร์วอลจะนำมาเทียบกับกฎ หากเข้ากับกฎใดกฎหนึ่งจะดูว่ากฎนั้นสั่งให้ส่งแพ็กเก็ตนั้นต่อไป หรือดร็อปแพ็เก็ตนั้นทิ้งไป และหากไม่เข้ากับกฎใดเลย ก็จะพิจารณาว่าค่าเริ่มต้นเป็นอะไร ให้ส่งหรือดร็อปแพ็กเก็ตทิ้งไป

A

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

B

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

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

ตารางที่ 4-1 แสดงตัวอย่างกฎที่ตั้งไว้สำหรับไฟร์วอลแบบแพ็กเก็ตฟิลเตอริง

ตารางข้างบนนี้ แสดงตัวอย่างของกฎที่ตั้งไว้สำหรับไฟร์วอลแบบแพ็กเก็ตฟิลเตอริง (Packet Filtering) โดยนำกฎมาใช้จะทำจากข้างบนลงมาข้างล่าง 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

การตัดสินใจในกรองแพ็กเก็ต โดยยึดข้อมูลที่อยู่ในแพ็กเก็ตเฮดเดอร์ (Packet Header) ของแพ็กเก็ตตัวนั้นๆ เช่น แอดเดรสต้นทาง (Source Address) แอดเดรสปลายทาง (Destination Address) พอร์ต (Port) โพรโตคอล (Protocol) ผลิตภัณฑ์ไฟร์วอลบางตัวที่สามารถกรองแพ็กเก็ต ยังมีความสามารถในการกรองข้อมูลในส่วนอื่นที่ไม่ใช่แพ็กเก็ตเฮดเดอร์ แต่เราจะพิจารณาส่วนนี้ในหัวข้อของ stateful inspection packet filter

4.2.2 แอพพลิเคชันพร็อกซี (Application Proxies)

แอพพลิเคชันพร็อกซีเป็นโปรแกรมแอพพลิเคชันที่รันอยู่บนระบบไฟร์วอลระหว่างสองเครือข่าย โดยเครื่องที่รันพร็อกซีไม่ต้องทำตัวเป็นเราเตอร์เมื่อโปรแกรมจากเครื่องไคลเอ็นต์เริ่มการติดต่อผ่านพร็อกซีไปยังเครื่องที่ให้บริการปลายทาง เริ่มแรกต้องทำการติดต่อกับโปรแกรมพร็อกซีเซิร์ฟเวอร์ก่อน โดยตัวไคลเอ็นต์จะทำการเจรจากับพร็อกซีให้ทำการติดต่อกับเครื่องที่ให้บริการปลายทางให้ ถ้าเป็นผลสำเร็จจะ

เกิดการเชื่อมต่อ 2 การเชื่อมต่อ คือ ระหว่างเครื่องไคลเอ็นต์กับเครื่องพร็อกซีเซิร์ฟเวอร์ และระหว่างเครื่องพร็อกซีกับเครื่องให้บริการปลายทาง เราจะเห็นได้ว่าพร็อกซีจะรับภาระของการจราจรสิงทิศทางระหว่างเครื่องไคลเอ็นต์กับเครื่องพร็อกซีเซิร์ฟเวอร์ จะทำการสร้างการติดต่อทั้งหมดและทำหน้าที่ในการตัดสินใจในการทำฟอร์เวิร์ดแพ็กเก็ตด้วย โดยพร็อกซีนี้ไม่มีส่วนเกี่ยวข้องในส่วนของการหาเส้นทาง

ทำนองเดียวกันกับแพ็กเก็ตฟิลเตอร์ริง ในแอพพลิเคชันพร็อกซีนั้นก็สามารถทำงานได้ในสองแพลตฟอร์ม คือ อุปกรณ์ที่ทำหน้าที่เป็นพร็อกซีโดยเฉพาะ กับคอมพิวเตอร์ที่ใช้ในงานทั่วไป ซึ่งโดยทั่วไปแล้วแอพพลิเคชันพร็อกซีจะช้ากว่าเราเตอร์ที่ใช้กรองแพ็กเก็ต แต่ในขณะเดียวกันแอพพลิเคชีนพร็อกซีมีความปลอดภัยสูงกว่าแพ็กเก็ตฟิลเตอร์ริง โดยที่จริงแพ็กเก็ตฟิลเตอร์ริงสามารถทนรับข้อบกพร่องจากการอิมพลีเมนต์ หรือทดแทนในส่วนของการอิมพลีเมนต์การเราต์ของระบบปฏิบัติการ แต่เมื่อความสามารถของแพ็กเก็ตฟิลเตอร์ริงถูกเพิ่มเข้าไปในการหาเส้นทาง มันก็ไม่สามารถที่จะชดเชยหรือทำให้ข้อบกพร่องในการหาเส้นทางถูกต้องได้

ผลจากการทำการกรองที่ซับซ้อนขึ้นและการตัดสินใจในการควบคุมการเข้าถึงแอพพลิเคชันพร็อกซีสามารถใช้ทรัพากรคอมพิวเตอร์ที่สำคัญ และเครื่องราคาแพงที่ใช้ในการปฏิบัติงาน ตัวอย่างเช่น ถ้าไฟร์วอลที่ทำงานบนระบบยูนิกซ์ ต้องการที่จะให้การสนับสนุนการทำงานจำนวน 200 HTTP session ที่ทำงานพร้อมๆกัน เครื่องที่ใช้ต้องมีความสามารถในการสนับสนุนงานของ 200 HTTP proxy ด้วยในแง่ของประสิทธิภาพในการรับ / ส่งข้อมูล และเมื่อเพิ่มอีก 100 FTP session , 25 SMTP session, LDAP session ส่วนหนึ่ง และ DNS transaction คุณจะต้องใช้เครื่องที่สนับสนุน 500-1000 proxy process บางพร็อกซีอาจจะอิมพลีเมนต์โดยใช้ kernel threads (เพื่อช่วยลดความต้องการของทรัพยากรลงอย่างรวดเร็ว) แต่ควมต้องการในทรัพยากรก็ยังคงมากอยู่ดี

4.2.3 สเตตฟูลอินสเปกชัน (Stateful Inspection) หรือ ไดนามิกแพ็กเก็ตฟิลเตอร์ริง (Dynamic packet filtering)

เราใช้คำว่า " สเตตฟูลอินสเปกชัน " หรือ " ไดนามิกแพ็กเก็ตฟิลเตอร์ริง " หมายถึง เซตฟังก์ชันที่ทำการกรองแพ็กเก็ตที่มีความสามารถขึ้นในเราเตอร์ โดยที่แพ็กเก็ตฟิลเตอร์ริงถูกจำกัดอยู่ในการตัดสินใจโดยข้อมูลในการตัดสินใจจะอยู่ในส่วนของเฮดเดอร์เท่านั้น และจะพิจารณาเฉพาะแพ็กเก็ตตัวนั้นๆ เท่านั้น โดยไม่ได้คำนึงถึงแพ็กเก็ตก่อนหน้านี้ ส่วนสเตตฟูลอินสเปกชันจะเอาชนะในข้อจำกัดนี้ของแพ็กเก็ตฟิลเตอร์ริงทั้งในส่วนของข้อมูลที่ใช้ในการตัดสินใจ โดยจะดูถึงข้อมูลในส่วนของเมสเซจด้วยและจะคำนึงถึงในส่วนของแพ็กเก็ตอื่นๆ ที่อยู่ก่อนหน้านี้ประกอบในการตัดสินใจ ทำให้ความสามารถในการตัดสินใจมีมากขึ้น เพราะรู้ข้อมูลมากขึ้น ซึ่งสเตตฟูลอินสเปกชันก็เหมือนกับแพ็กเก็ตฟิลเตอร์ริง คือ อิมพลีเมนต์เพิ่มลงในเราเตอร์ ดังนั้นโฮสต์ที่มีฟังก์ชันของสเตตฟูลอินสเปกชันทำงานอยู่นั้นจะต้องเป็นหรือสามารถทำงานเสมือนเราเตอร์ได้ด้วย

เจตนาพื้นฐานของสเตตฟูลอินสเปกชัน นั้นเพื่อรวมข้อดีระหว่างประสิทธิภาพในแล่ของการส่งถ่ายข้อมูลและความปลอดภัย โดยเพิ่มการทำงานลงในเราเตอร์ทำให้สเตตฟูลอินสเปกชันมีประสิทธิภาพในการส่งถ่ายข้อมูลดีกว่าพร็อกซี และยังมีการเพิ่มระดับฟังก์ชันของไฟร์วอลให้ดีกว่าแพ็กเก็ตฟิลเตอร์ริง

ธรรมดา และสเตตฟูลอินสเปกชันยังมีส่วนคล้ายกับพร็อกซี คือ เกณฑ์ของการควบคุมการเข้าถึงข้อมูลที่ซับซ้อนสามารถระบุรายละเอียดได้ และเหมือนกับแพ็กเก็ตฟิลเตอร์ริง คือ สเตตฟูลอินสปกชันได้รับการสืบทอดมาจากการอิมพลีเมนต์การหาเส้นทางที่มีคุณภาพสูง

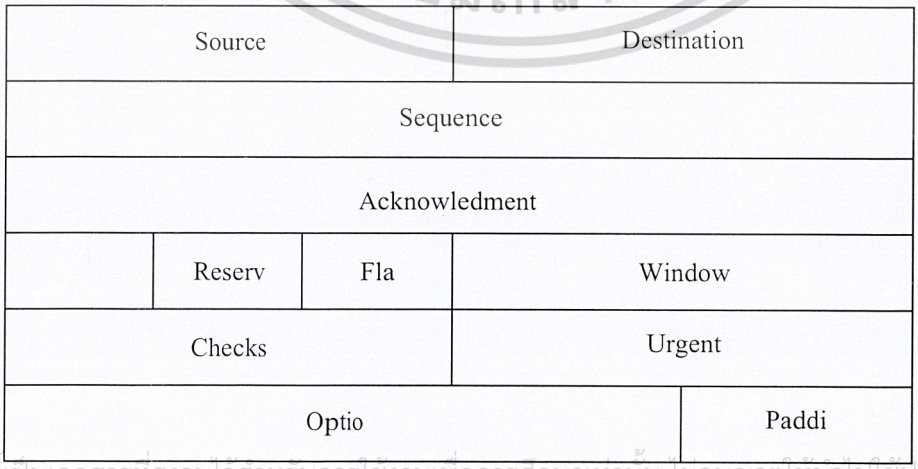
4.3 หลักการทำงานของซิมเพิลแพ็กเก็ตฟิลเตอร์ (Simple Packet Filter)

ชั้นไอพีเลเยอร์ นั้นเป็นชั้นที่ทั้งซิมเพิลแพ็กเก็ตฟิลเตอร์ (simple packet filter) และ สเตตฟูลฟิลเตอร์ (stateful filter) ใช้เป็นจุดเริ่มต้นในการพิจารณาข้อมูล สิ่งที่ซิมเพิลแพ็กเก็ตฟิลเตอร์และสเตตฟูลไฟร์วอล ทำเหมือนกันก็คือ ตรวจสอบข้อมูล header ของไอพีแพ็กเก็ตโดยสิ่งที่ถูกตรวจสอบ คือ source address, destination address, protocol , options

นโยบายของไฟร์วอลส่วนใหญ่มักจะตรวจสอบความถูกต้องของแอดเดรส (ทั้ง source address และ destination address) ด้วย ยกเว้นกรณีที่ระบุเป็น “all” และข้อมูลในส่วนของโพรโตคอลนั้นก็จะเป็นตัวบอกว่าส่วนของดาต้านั้นบรรจุข้อมูลชนิดใดอยู่ เช่น TCP, UDP หรือบางทีอาจจะเป็น IPSEC, ESP, ISAKMP กระบวนการในการตรวจสอบรูปแบบของเฮดเดอร์นั้นจะอยู่ที่เลเยอร์ชั้นบนถัดไป และข้อมูลในส่วนออปชันนั้น โดยปกติมักจะไม่มีค่าอะไร แต่อาจจะมีแฟล็ก (flag) ที่แสดงถึง ซอร์ทเราต์ติง (source routing)ได้ (source routing ใช้ในกรณีที่ผู้ส่งต้องการให้ผู้รับส่งข้อมูลผ่านเส้นทางที่ผู้ส่งได้กำหนดไว้) โดยปกติแล้วมักจะถูกใช้โดยผู้บุกรุก ดังนั้นซิมเพิลแพ็กเก็ตฟิลเตอร์ส่วนใหญ่จะดร็อปแพ็กเก็ตที่ถูกส่งมาพร้อมกับแฟล็กดังกล่าว โดยไม่มีการตรวจสอบข้อมูลอย่างอื่นเพิ่มเติม

ทีซีพีเฮดเดอร์ บรรจุข้อมูล 3 ส่วนที่สำคัญในทรานสปอร์ตเลเยอร์ (transport layer) ไว้คือ source port , destination port และ แฟล็ก โดยในส่วนของแฟล็กนั้น สามารถประกอบไปด้วย URG (urgent), ACK (acknowledgment), PSH (push), RST (reset), SYN (syncronize) และ FIN (finish) และยังมีส่วนที่สี่ที่บรรจุ sequence number ซึ่งผู้ผลิตบางค่ายนำมาใช้พิจารณาด้วย

สำหรับแฟล็กนั้นเป็นส่วนสำคัญมากในสเตตฟูลไฟร์วอล เช่น การทำ ทีซีพี ทรีเวย์แฮนด์เชค (TCP – 3 way handshake



รูปที่ 4-1 TCP header

ยูดีพี เฮดเดอร์ บรรจุข้อมูลเพียงแค่ source port และ destination port เท่านั้นเอง ไม่มีส่วนของแฟล็ก และ ซีเควนนัมเบอร์ (sequence number) แต่อย่างใด เนื่องจาก ยูดีพีเป็น connectionless protocol อยู่แล้วนั่นเอง ไม่ได้บรรจุข้อมูลพอร์ตแต่อย่างใด แต่บรรจุชนิดของ ไอซีเอ็มพี (ICMP message type) เช่น "Echo request" , "Destination Unreachable"

Source	Destination
Length	Checksum
Data begin here	

รูปที่ 4-2 UDP header

ซิมเพิลแพ็กเก็ตฟิลเตอร์ (Simple packet filter) จะพิจารณา ทีซีพีแพ็กเก็ตเพื่อตรวจสอบว่า

- แพ็กเก็ตนั้นมีแอดเดรสที่ถูกต้องหรือไม่
- แพ็กเก็ตนั้นถูกสร้างมาจาก External address จริงหรือไม่
- โพรโตคอล หรือ เซอร์วิสนั้น ผ่านการตรวจสอบแล้วหรือไม่
- ออปชั่น หรือ แฟล็กที่ส่งมานั้น ถูกต้องตามข้อกำหนดหรือไม่

ในกรณีที่การส่ง ACK แพ็กเก็ตดังกล่าวถูกส่งมาโดยเจตนาร้าย เพราะเมื่อเครื่องปลายทางได้รับ ACK มักจะส่งสัญญาณ RST กลับไป ซึ่งจะแสดงให้เห็นว่าเครื่องปลายทางนั้นยังเปิดอยู่ และเปิดให้บริการในพอร์ตดังกล่าว (destination port ที่ส่งมาพร้อมกับ ACK packet) บางครั้งยังสามารถใช้ตรวจสอบระบบปฏิบัติการของเครื่องปลายทางได้อีกด้วย (OS fingerprint) เพราะระบบปฏิบัติการในแต่ละระบบนั้นมักมีวิธีตอบกลับที่แตกต่างกันออกไป

4.4 หลักการทำงานของแพ็กเก็ตฟิลเตอร์ริงสเตตฟูลไฟร์วอล(Packet – filtering stateful firewall)

สำหรับแพ็กเก็ตฟิลเตอร์ริงสเตตฟูลไฟร์วอล แล้วก็ทำเช่นเดียวกันกับแพ็กเก็ตฟิลเตอร์ แต่จะบันทึกข้อมูลที่เกี่ยวกับการเชื่อมต่อ ที่เกิดขึ้นลงในตารางการเชื่อมต่อ ก่อนที่จะส่งแพ็กเก็ตนั้นไปยัง ไอพีสแต็ก (IP stack) ตัวตารางนี้จะมีส่วนสำหรับบันทึกข้อมูลสำหรับแต่ละการเชื่อมต่อที่ถูกต้อง โดยปกติจะเก็บข้อมูล source and destination address , protocol, port, flag แต่มีไฟร์วอลบางยี่ห้อที่เก็บข้อมูล 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 4-2 ตัวอย่าง State table

เมื่อแพ็กเก็ตฟิลเตอร์ริงไฟร์วอล ได้รับแพ็กเก็ตมันจะตรวจสอบข้อมูลกับตารางการเชื่อมต่อว่าเป็นส่วนของการเชื่อมต่อที่สร้างไว้แล้วหรือไม่ โดยจะพิจารณาจากข้อมูล source address, destination address , source port , destination port จะต้องสอดคล้องกับตารางการเชื่อมต่อ ซึ่งถ้าเป็นส่วนหนึ่งขอวการเชื่อมต่อจริงก็ไม่มีความจำเป็นใดๆที่ต้องตรวจสอบซ้ำอีก แต่ในไฟร์วอลบางยี่ห้อนั้น จะพิจารณา sequence number ของแพ็กเก็ตเพิ่มเติมด้วย เช่น CiscoPIX ในขณะที่ Checkpoint's Firewall-1 ไม่พิจารณา sequence number แต่อย่างใด และเนื่องจากมีการเก็บ history ไว้ในตารางการเชื่อมต่อ ดังนั้นสเตตฟูลไฟร์วอลสามารถดร็อผแพ็กเก็ตที่ผิดปกติได้ และถ้าแพ็กเก็ตที่ส่งมาเป็นส่วนหนึ่งของการเชื่อมต่อที่สร้างไว้แล้วก็สามารถส่งผ่านได้เลย ซึ่งทำให้ประหยัดเวลาในการตรวจสอบจากไหร์วอลใหม่อีกรอบ

ถ้าแพ็กเก็ตที่ส่งมาไม่ตรงกับการเชื่อมต่อที่สร้างไว้แล้ว และไม่ใช่ SYN แพ็กเก็ต ตัวแพ็กเก็ตนั้นก็จะถูกดร็อปทิ้งไป และแม้แต่แพ็กเก็ตที่ผสมแฟล็กแปลกๆ เช่น SYN/FIN (เป็นกระบวนการหนึ่งในการทำพอร์ตสแกนนิง) ก็จะถูกดร็อปไปเช่นเดียวกัน ทั้งนี้ไฟร์วอลส่วนใหญ่สามารถบันทึกล็อกไฟล์ได้ด้วย ซึ่งขึ้นอยู่กับการตั้งค่าของผู้ดูแลระบบเองว่าต้องการเก็บข้อมูลใด

ในกรณีที่การส่ง ACK แพ็กเก็ตดังกล่าวถูกส่งมาโดยเจตนาร้าย เพราะเมื่อเครื่องปลายทางได้รับ ACK มักจะส่งสัญญาณ RST กลับไป ซึ่งจะแสดงให้เห็นว่าเครื่องปลายทางนั้นยังเปิดอยู่ และเปิดให้บริการในพอร์ตดังกล่าว (destination port ที่ถูกส่งมาพร้อมกับ ACK packet) บางครั้งยังสามารถใช้ตรวจสอบระบบปฏิบัติการของเครื่องปลายทางได้อีกด้วย (OS fingerprint) เพราะระบบปฏิบัติการในแต่ละระบบนั้นมักมีวิธีตอบกลับที่ไม่เหมือนกัน

บทที่ 5
การโจมตีเพื่อปิดให้บริการ

Denial of Services Attack

Denial of Services เป็นการโจมตีเป้าหมายด้วยวิธีการต่างๆเพื่อมิให้เป้าหมายสามารถให้บริการได้ตามปกติ ซึ่งส่วนใหญ่จะอาศัยข้อบกพร่องของโปรโตคอลประกอบกับข้อบกพร่องของระบบปฏิบัติการที่ทำงานอยู่บนเป้าหมาย ทำให้การทำงานและการตอบสนองต่อการ DoS นั้นส่งผลกระทบอย่างรุนแรงต่อสมรรถนะการทำงาน

Anomalous Packet

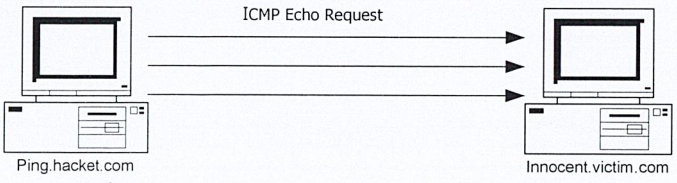
อะนอมาลัสแพ็กเก็ต(Anomalous Packet) หมายถึงแพ็กเก็ตประหลาดที่ไม่มีโอกาสเกิดขึ้นในสภาวะปกติได้โดยอย่างสิ้นเชิง แพ็กเก็ตประเภทนี้เป็นการจงใจเปลี่ยนข้อมูลสำคัญที่ใช้ควบคุมการสื่อสารข้อมูลให้ผิดปกติ ผิดธรรมชาติของการสื่อสารข้อมูลธรรมดา

ไม่ว่า IP, UDP หรือ TCP ต่างก็มีข้อกำหนดอยู่ในโปรโตคอลของตนเอง ในแต่ละโปรโตคอลย่อมมีค่าในเฮดเดอร์ที่จะใช้เป็นกลไกการควบคุมการสื่อสารข้อมูล โดยในสภาวะของการสื่อสารข้อมูลตามปกติแล้วค่าในเฮดเดอร์เหล่านี้จะมีค่าที่มีขอบเขตและคาดหมายได้ แต่อะนอมาลัสแพ็กเก็ตเหล่านี้จะเป็นแพ็กเก็ต ที่ถูกดัดแปลงด้วยเทคนิคของการควบคุมเน็ตเวิร์กเลเยอร์โดยตรงแบบไม่ผ่านโปรโตคอล ซึ่งเป็นช่องทางให้แฮกเกอร์ทั้งหลายได้ใช้โจมตีเป้าหมายให้ทำงานผิดพลาดไปเพราะต้องจัดการกับแพ็กเก็ตที่ไม่ได้ระบุอยู่ในโปรโตคอล หรือมิเช่นนั้นก็ใช้เพื่ออำพรางตนเองในการสำรวจเป้าหมาย เป็นต้น สำหรับการดัดแปลงของแต่ละโปรโตคอลสามารถแสดงได้พอสังเขปดังนี้

- IP : IP Length, Fragment Offset, Fragment Flag, TCP Option
- UDP : UDP Length, Socket (Address & Port)
- TCP : TCP Flag, Socket (Address & Port)

อะนอมาลัสแพ็กเก็ตที่นิยมนำมาใช้งานกันมากที่สุดเห็นจะเป็น TCP โดยเฉพาะการปรับเปลี่ยนแฟลกไปต่างๆนานาสารพัดเงื่อนไขเท่าที่จะสามารถเปลี่ยนได้

5.1 Ping Flood Attack



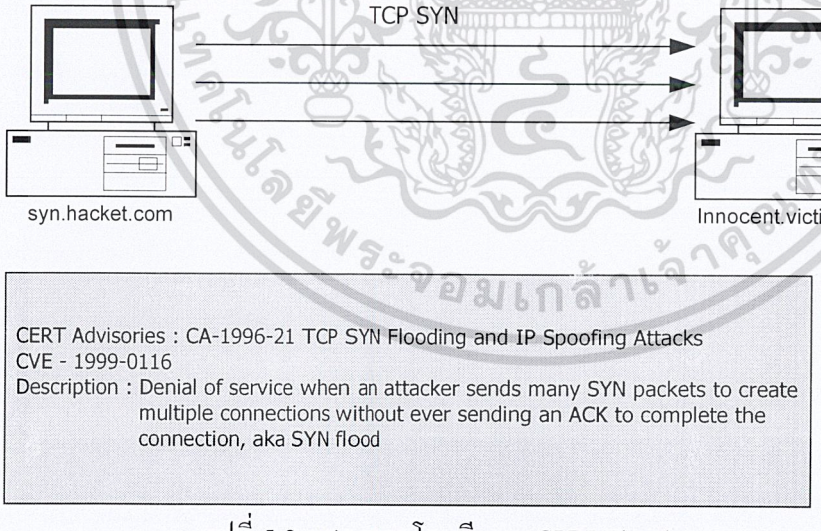
รูปที่ 5-1 แสดงการโจมตีแบบ Ping Flood Attack

Ping Flood เป็นการโจมตีที่ใช้กันในยุคแรกๆของ DoS เป็นการโจมตีที่มิได้อาศัยเทคนิคลึกลับซับซ้อนแต่อย่างใด อาศัยปริมาณแพ็กเก็ตมากๆเพียงอย่างเดียว แต่ถึงกระนั้นก็ตามจากการโจมตีวิธีนี้ก็สร้างความเสียหายได้ไม่น้อย

หลักการโจมตีของ Ping Flood คือการส่ง ICMP Echo Request (แบบเดียวกับที่ได้จากคำสั่ง Ping) ปริมาณมากๆไปยังเป้าหมายอย่างรวดเร็ว ทำให้โฮสต์ที่ถูกโจมตีจะต้องคอยตอบ ICMP Echo Reply ตลอดเวลาจนแทบจะไม่มีเวลาทำงานอื่น ความรุนแรงของการโจมตีจะมากหรือน้อยแปรผันตามความเร็วของการส่ง ICMP แพ็กเก็ตเป็นหลัก

นอกจากการสร้างความเสียหายแก่เครื่องคอมพิวเตอร์เป้าหมายแล้ว Ping Flood ยังสร้างความเสียหายให้แก่เน็ตเวิร์กที่เครื่องคอมพิวเตอร์เป้าหมายนั้นอยู่ด้วย โดยความเสียหายจะมีขอบเขตมากน้อยเพียงใดขึ้นอยู่กับลักษณะการออกแบบของเน็ตเวิร์กนั้นๆ ถึงแม้ว่าแพ็กเก็ตที่ส่งไปยังเซิร์ฟเวอร์เป้าหมายนั้นจะมีหมายเลข IP เป็นของเป้าหมายเครื่องเดียว แต่ในความเป็นจริงที่เลเยอร์ต่ำลงไป เช่น Ethernet ต่างก็ใช้งานร่วมกันกับเครื่องโฮสต์อื่นๆในเน็ตเวิร์ก การที่ข้อมูลจะเดินทางจากที่ใดและจะไปที่ใดจะต้องผ่านเน็ตเวิร์กที่ใช้งานร่วมกันนี้ ดังนั้นเมื่อแพ็กเก็ตจำนวนมากถูกส่งมายังเครื่องเป้าหมาย นอกจากจะทำให้เครื่องเป้าหมายเสียหาย แล้วเน็ตเวิร์กอันเป็นทางผ่านก็จะเต็มไปด้วยแพ็กเก็ตนี้เช่นกัน ลักษณะเช่นนี้จะเกิดมากบนเน็ตเวิร์กที่มีการใช้การสื่อสารเลเยอร์ต่ำร่วมกันเช่น 10Base-5 หรือ10Base-T ที่ใช้ Hub เป็นตัวกระจายสัญญาณ

5.2 SYN Flood Attack



รูปที่ 5-2 แสดงการโจมตีแบบ SYN Flood Attack

ถ้าจะนับว่าการโจมตีของ Ping Flood เป็นการทดลองกำลังกันในระดับ IP (ด้วย ICMP) SYN Flood ก็นับเป็นการลองกำลังในระดับ TCP สิ่งที่เป็นคุณสมบัติสำคัญของ TCP คือการเชื่อมต่อที่มีเสถียรภาพ โดยมี

ขั้นตอนการสถาปนาการเชื่อมต่อ และยุติการเชื่อมต่อ แต่ข้อดีเหล่านี้กลับถูกนำไปใช้เป็นอาวุธสำคัญในการโจมตี

ด้วยลักษณะในการเริ่มต้นเชื่อมต่อของ TCP นั้นจะเป็นการตรวจสอบซึ่งกันและกันทั้ง 2 ฝ่ายที่เรียกว่า 3-ways handshake โดยเริ่มต้นจากเครื่องที่ต้องการติดต่อ ส่งสัญญาณ SYN มายังเซิร์ฟเวอร์ หลังจากนั้นการเริ่มต้นการเชื่อมต่อตามโปรโตคอลก็จะดำเนินต่อไป

สิ่งที่ยากที่สุดในการทำงานของ TCP ก็คือการพยายามทำให้ทั้งสองฝ่ายสามารถสื่อสารกันได้อย่างถูกต้องและมีเสถียรภาพ สิ่งที่ต้องพิจารณาก็คือ แพ็กเก็ตของการเชื่อมต่อ จะเริ่มต้นด้วยการได้รับสัญญาณ SYN และจะต้องตอบ SYN ACK กลับไปให้แก่ผู้ที่ขอ จากนั้นต้องรอการตอบรับอีกครั้งหนึ่งของไคลเอนต์จึงจะจบกระบวนการ ดังนั้นเพื่อที่จะให้การเชื่อมต่อสามารถดำเนินไปได้อย่างต่อเนื่องโปรแกรมที่จัดการ 3-ways handshake ของเซิร์ฟเวอร์จะต้องจัดสรรหน่วยความจำจำนวนหนึ่งเพื่อรองรับการเชื่อมต่อแต่ละเซสชั่นจนกว่าการทำ 3-ways handshake จะสิ้นสุดลง โดยที่เซิร์ฟเวอร์เองก็ไม่มีทางรู้ได้เลยว่าไคลเอนต์จะ ACK กลับมาเพื่อจบการเชื่อมต่อนั้นของ 3-ways handshake เมื่อไร ซึ่งแน่นอนว่าในการบริหารหน่วยความจำและการสื่อสารข้อมูลจะต้องสร้างความสมดุลระหว่างเสถียรภาพของเซิร์ฟเวอร์ และประสิทธิภาพของการสื่อสาร เซิร์ฟเวอร์เองก็จะมีเวลาค่าหนึ่งที่จะรอให้ได้รับสัญญาณ ACK ตอบกลับมา หากถึงเวลาที่กำหนดแล้วไม่มีแพ็กเก็ต ACK กลับมาเซิร์ฟเวอร์จะต้องยุติการรอนั้นและคืนหน่วยความจำให้แก่ระบบปฏิบัติการ

เทคนิคที่อาศัยการ SYN นี้จะส่งผลกระทบโดยตรงกับเซิร์ฟเวอร์ หากระบบปฏิบัติการของเซิร์ฟเวอร์เป้าหมายจัดการหน่วยความจำได้ไม่มีประสิทธิภาพพอก็อาจจะหยุดทำงานทันที

จนถึงปัจจุบัน SYN Flood ก็ยังเป็นการโจมตีที่ได้ผลอยู่และหาทางป้องกันได้ยาก เนื่องจากยากที่จะจำแนกลักษณะของแพ็กเก็ตที่ใช้โจมตีกับแพ็กเก็ตที่ขอเริ่มต้นการเชื่อมต่อทั่วไป โดยเฉพาะสำหรับเซิร์ฟเวอร์ที่มีฮิตเรตสูงๆ เช่น เว็บเซิร์ฟเวอร์ และถึงแม้ว่าระบบปฏิบัติการรุ่นใหม่จะได้มีการปรับปรุงในด้านการจัดการหน่วยความจำให้ดีขึ้นแต่ก็คงจะช่วยได้เพียงเซิร์ฟเวอร์ไม่ถึงกับหยุดทำงานไปเลยเมื่อโดนโจมตีเท่านั้นแต่ก็ยากที่จะทำให้เซิร์ฟเวอร์ไม่ถึงกับหยุดทำงานไปเลยเมื่อโดนโจมตีเท่านั้น แต่ก็ยากที่จะทำให้เซิร์ฟเวอร์ยังคงรักษาความสามารถในการให้บริการได้เช่นสภาวะปกติ

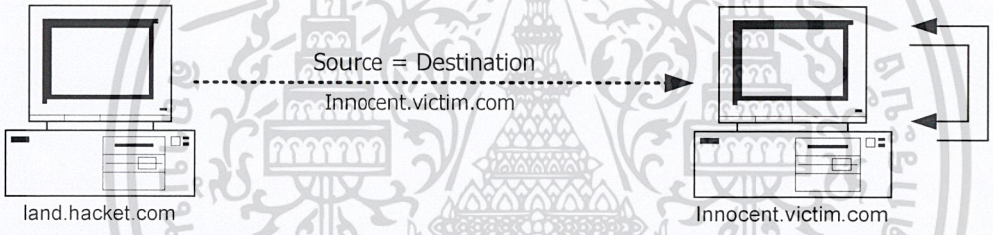
Connection on innocent.victim.com

TCP

Local Address	Remote Address	State
.	*.*	IDLE
*.ftp	*.*	LISTEN
*.smtp	*.*	LISTEN
*.http	*.*	LISTEN
*.pop	*.*	LISTEN
innocent.http	10.15.14.1.17905	SYN_RCVD
innocent.http	10.15.14.2.17905	SYN_RCVD
innocent.http	10.15.14.3.17905	SYN_RCVD
innocent.http	10.15.14.4.17905	SYN_RCVD
innocent.http	10.15.14.5.17905	SYN_RCVD
innocent.http	10.15.14.6.17905	SYN_RCVD
innocent.http	10.15.14.7.17905	SYN_RCVD
innocent.http	10.15.14.8.17905	SYN_RCVD
.	*.*	IDLE

รูปที่ 5-3 แสดงสถานะการเชื่อมต่อบน innocent.victim.com เมื่อถูกโจมตี

5.3 Land Attack



CERT Advisories : Advisory CA-1997-28 IP Denial-of-Service Attacks (Topic 1)
CVE 1999-016
Description : Land IP Denial of Service

รูปที่ 5-4 แสดงการโจมตีแบบ Land Attack

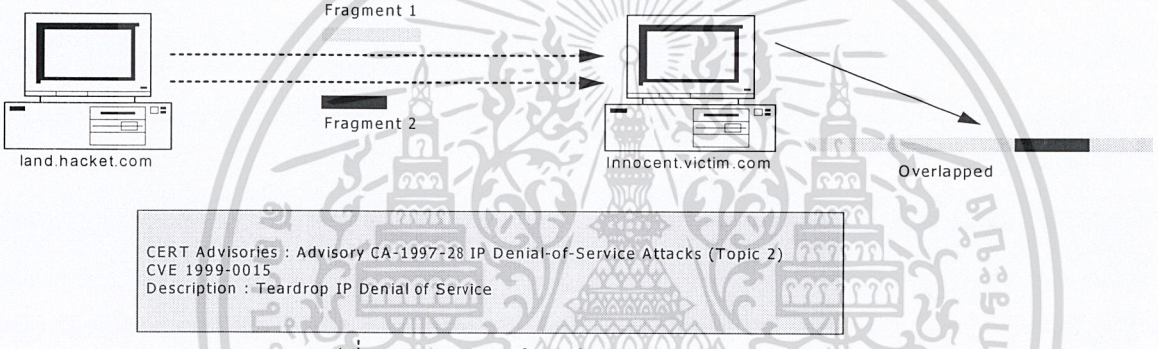
ลักษณะการโจมตี

- หมายเลข IP ต้นทางเท่ากับ IP ปลายทาง
- หมายเลขพอร์ตต้นทางเท่ากับหมายเลขพอร์ตปลายทาง
- SYN Flag ถูก Set เสมือนขอเริ่มต้นการเชื่อมต่อ
- แพ็กเก็ตจะส่งไปยัง TCP พอร์ตที่เปิดอยู่

ปกติในข้อกำหนดของ TCP เมื่อมีการส่งสัญญาณ SYN มาเพื่อการเชื่อมต่อไปยังเป้าหมาย เป้าหมายก็จะตอบรับกลับไปยังผู้ส่งด้วย SYN ACK ตามหมายเลขพอร์ตและหมายเลข IP ต้นทาง แต่สำหรับการโจมตีแบบนี้หมายเลข IP ต้นทางและปลายทางอีกทั้งหมายเลขพอร์ตต้นทางและปลายทางจะถูกตั้งให้เป็นค่าเดียวกัน

ดังนั้นการตอบกลับด้วย SYN ACK ก็จะตอบกลับไปที่ปลายทางเหมือนเดิม ซึ่งกรณีนี้ไม่มีกำหนดอยู่ในโปรโตคอลว่าควรจะทำอย่างไร โฮสต์จึงพยายามตอบสนองตามข้อกำหนดเท่าที่มีอยู่โดยการตอบกลับไปที่ IP Address และพอร์ตต้นทางที่ถูกบุกรุกมานั่นหมายถึงการตอบกลับเข้ามายังตัวเอง ซึ่งจะทำให้มีการตอบกลับไปมาของ TCP วนรอบอยู่ในตัวเองด้วยความเร็วสูง ทำให้คอมพิวเตอร์ต้องใช้ทรัพยากรที่มีอยู่ทั้งหมด ไม่ว่า CPU หน่วยความจำและอื่นๆเพื่อคอยจัดการกับ TCP ที่ตอบกลับไปมาดังกล่าวจนไม่อาจไปทำงานอื่นๆได้อีก จนดูเหมือนว่าเครื่องคอมพิวเตอร์หยุดทำงานและไม่ตอบสนองต่อการกระตุ้นใดๆแม้แต่คีย์บอร์ด ดังนั้นเหลือวิธีเดียวคือต้องรีเซ็ตเครื่องหรือปิดเครื่องจึงจะสามารถหยุดการวนรอบของ TCP ได้

5.4 Teardrop Attack



รูปที่ 5-5 แสดงการโจมตีแบบ Teardrop Attack

Teardrop เป็นการโจมตีโดยใช้ข้อบกพร่องของแฟรกเมนต์รีแอสเซมเบิล (การรวมแฟรกเมนต์แพ็กเก็ตหลายแพ็กเก็ตกลับมาเป็น IP ดาต้าแกรมเดียว) ของ IP เพื่อทำให้ระบบปฏิบัติการปลายทางของเป้าหมายทำงานผิดพลาด ไม่อยู่ในเงื่อนไขที่กำหนดไว้และหยุดทำงาน

ในการประกอบรวมแฟรกเมนต์แพ็กเก็ตกลับมาอยู่ใน 1 ดาต้าแกรมนั้น IP มีลักษณะการทำงานโดยพิจารณาจากข้อมูลของ 3 ฟิลด์คือ

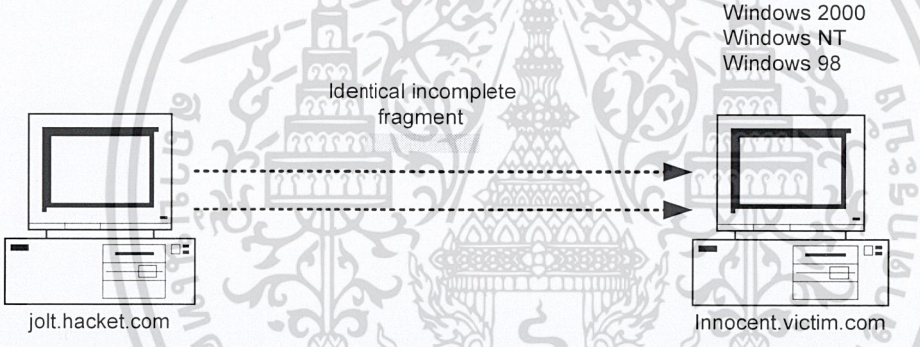
1. Data length : ขนาดความยาวของข้อมูลในแพ็กเก็ตนั้น
2. Offset : ตำแหน่งของแพ็กเก็ตที่จะนำไปประกอบรวมกลับใน IP ดาต้าแกรม
3. Flag : แฟลกซึ่งระบุว่าไม่มีแพ็กเก็ตต่อจากแพ็กเก็ตนี้อีก หมายถึงแพ็กเก็ตนี้เป็นส่วนสุดท้ายของดาต้าแกรม

โดยทั่วไปแล้ว IP แพ็กเก็ตที่ถูกแฟรกเมนต์มานั้นไม่จำเป็นจะต้องถึงปลายทางตามลำดับ ดังนั้นปลายทางผู้รับแพ็กเก็ตจึงต้องทำการรีแอสเซมเบิลโดยนำแพ็กเก็ตที่เข้ามาไปวางรอ ไว้ตรงตำแหน่งของ offset และรอจนกว่าทุกแพ็กเก็ตจะมาครบ จากนั้นจึงส่ง IP ดาต้าแกรมที่สมบูรณ์นั้นไปยังเลเยอร์ถัดขึ้นไป ด้วยลักษณะของการทำงานเช่นนี้ มีข้อบกพร่องมากมายที่สามารถนำไปใช้ในการโจมตี

หากเป็นการรับแฟรกเมนต์แพ็กเก็ตตามปกติแล้ว IP ก็จะนำแฟรกเมนต์แต่ละแพ็กเก็ตที่ได้รับมาวางรอไว้ในหน่วยความจำตามตำแหน่งที่ระบุในฟิลด์ offset ตามลำดับการมาถึงของแพ็กเก็ตจะกระทั่งทุกแฟรกเมนต์ได้มาถึงครบหมดจึงรวมกลับมาเป็นดาต้าแกรมที่สมบูรณ์ หรือหากเกินเวลาที่กำหนดแล้วแฟรกเมนต์ยังเดินทางมาไม่ครบ IP ก็จะแจ้งข้อผิดพลาดกลับไปพร้อมทั้งยกเลิกการรับดาต้าแกรมนั้น

การโจมตีของ Teardrop จะใช้การเหลื่อมกันของแพ็กเก็ตในระหว่างที่มีการรวมแฟรกเมนต์แพ็กเก็ตเข้าด้วยกัน เนื่องจากในการแฟรกเมนต์ตามปกติแล้วแพ็กเก็ตจะถูกแบ่งออกเป็นส่วนย่อย แต่สามารถนำมารวมกันใหม่ได้พอดี และตำแหน่งจะถูกต้องสอดคล้องกันเสมอ แพ็กเก็ตที่ใช้ในการโจมตีของ Teardrop จะเป็นแพ็กเก็ตที่ถูกสร้างขึ้นมาโดยเฉพาะมิได้ผ่านกลไกการแฟรกเมนต์ตามปกติของ IP โดยมีการระบุ offset ที่เหลื่อมเข้าไปในแพ็กเก็ตอื่นๆ ซึ่งจะไม่มีโอกาสเกิดขึ้นได้เลยในการทำงานปกติ ดังนั้นหากระบบปฏิบัติการไม่สามารถจัดการเงื่อนไขไม่ปกติเช่นนี้ได้ดีเพียงพอก็จะหยุดทำงานลงได้

5.5 Jolt



CVE -2000-0305
Description : Windows 95, Windows 98, Windows 2000, Windows NT 4.0, and Terminal Server systems allow a remote attacker to cause a denial of service by sending a large number of identical fragmented IP packets, aka jolt2 or the "IP Fragment Reassembly" Vulnerability.

รูปที่ 5-6 แสดงการโจมตีแบบ Jolt Attack

เป็นที่สังเกตุได้ว่าระบบปฏิบัติการของ Microsoftจะอ่อนไหวต่อแฟรกเมนต์แพ็กเก็ตเป็นพิเศษ การ DoS สำหรับ Windows ส่วนใหญ่สามารถกระทำได้โดยง่าย โดยอาศัย Fragment และ ส่งผลเสียต่อเป้าหมายได้ค่อนข้างมาก

ส่วนใหญ่เมื่อเซิร์ฟเวอร์ถูกโจมตีด้วย Fragment แล้วจะปรากฏอาการคือ

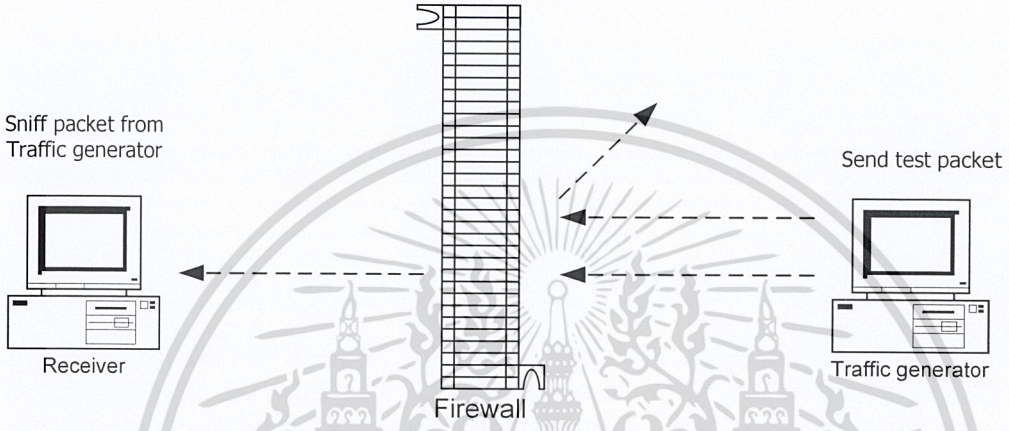
- หยุดการทำงานและแสดงหน้าจอสีฟ้า (Blue Screen) ต้องทำการ Cold boot เครื่องใหม่ หรืออาจจะต้องปิดเครื่องแล้วเปิดใหม่

- หยุดการทำงานเครื่องชั่วขณะ เซิร์ฟเวอร์ไม่ตอบสนองใดๆเหมือนกับแฮงค์ ผู้ใช้จะไม่สามารถควบคุมหรือติดต่อกับระบบปฏิบัติการได้ ส่วนใหญ่ผู้ใช้ก็ต้อง Reset และปิดเครื่องแล้วเปิดใหม่เช่นเดียวกัน
- ทำงานช้าลงไปมาก ตอบสนองต่อผู้ใช้ช้ามาก เนื่องจาก CPU จะถูกนำไปใช้ในการจัดการกับ Fragment ที่เข้ามา หากโดนโจมตีมากๆเครื่องก็จะไม่ตอบสนองอีกเลย

Jolt เป็นการโจมตีโดยอาศัยแฟรกเมนต์ทั่วไป แฮกเกอร์จะอาศัยการส่งแฟรกเมนต์แพ็กเก็ตซ้ำๆ จำนวนมากอย่างต่อเนื่องมายังเซิร์ฟเวอร์เป้าหมาย หากข้อมูลมีความเร็วและปริมาณไม่มากพอก็จะเพียงแต่ทำงานช้าลงกว่าปกติ หากแฮกเกอร์สามารถเพิ่มความเร็วในการส่งข้อมูลเครื่องก็จะหยุดทำงานไม่ตอบสนองต่อผู้ใช้อีกเลย CPU Utilization จะขึ้นสูงถึง 100% และถูกใช้ไปโดยเคอร์เนล (Kernel) ของระบบปฏิบัติการนั่นเป็นเหตุผลที่ทำให้ไม่มีการตอบสนองใดๆต่อผู้ใช้ เนื่องจากกำลังของ CPU ถูกใช้ไปในการจัดการกับ Fragment หมดนั่นเอง

บทที่ 6
โปรแกรมวิเคราะห์นโยบายไฟร์วอล

6.1 การออกแบบโปรแกรม



รูปที่ 6-1 แสดงภาพรวมการทำงานของโปรแกรมตรวจสอบไฟร์วอล

จากรูป จะเห็นว่าจะมีการทำงานออกเป็น 2 ส่วนหลักๆ คือ

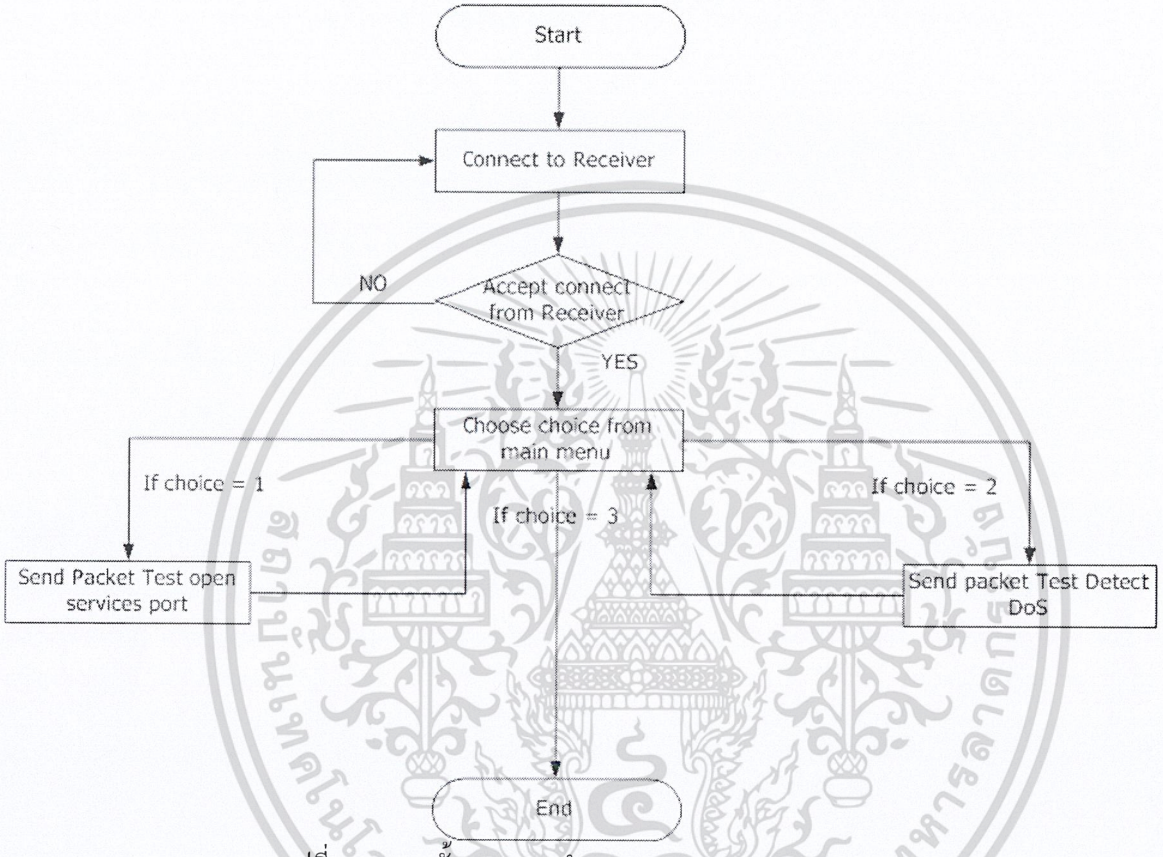
1. ใช้ในการสร้างแพ็กเก็ตชนิดต่างๆ ที่จะใช้ส่งไปเพื่อตรวจสอบการทำงานของไฟร์วอล (Traffic -Generator)

2. ใช้ดักจับแพ็กเก็ต ที่ส่งมาและนำแพ็กเก็ตที่จับได้ มาวิเคราะห์ผลการทำงานของไฟร์วอล กับสัญญาณที่ได้จากอีกฝั่ง(Receiver)

1. Traffic Generator

หลักการทำงานก็คือ จะทำการสร้างแพ็กเก็ตตามโปรโตคอลต่างๆ โดยในโปรแกรมนี้จะตรวจสอบถึงการเปิดให้บริการของไฟร์วอลว่าให้บริการโปรโตคอลใดที่พอร์ตใดบ้าง ซึ่งจะมีการตรวจสอบอยู่ 3 โปรโตคอลคือ TCP, UDP และ ICMP และนอกจากนี้ก็จะทำการจำลองการสร้างสถานะการการถูกโจมตีให้ปิดการให้บริการ (Denial of Services) ในแบบต่างๆ เพื่อที่จะใช้ในการตรวจสอบ ความสามารถในการ ป้องกันการโจมตีเพื่อ ให้ปิดให้บริการในบางชนิดเช่นแลนด์แอทแทคฯลฯ เป็นต้น

แสดงขั้นตอนการทำงานของตัว Traffic Generator

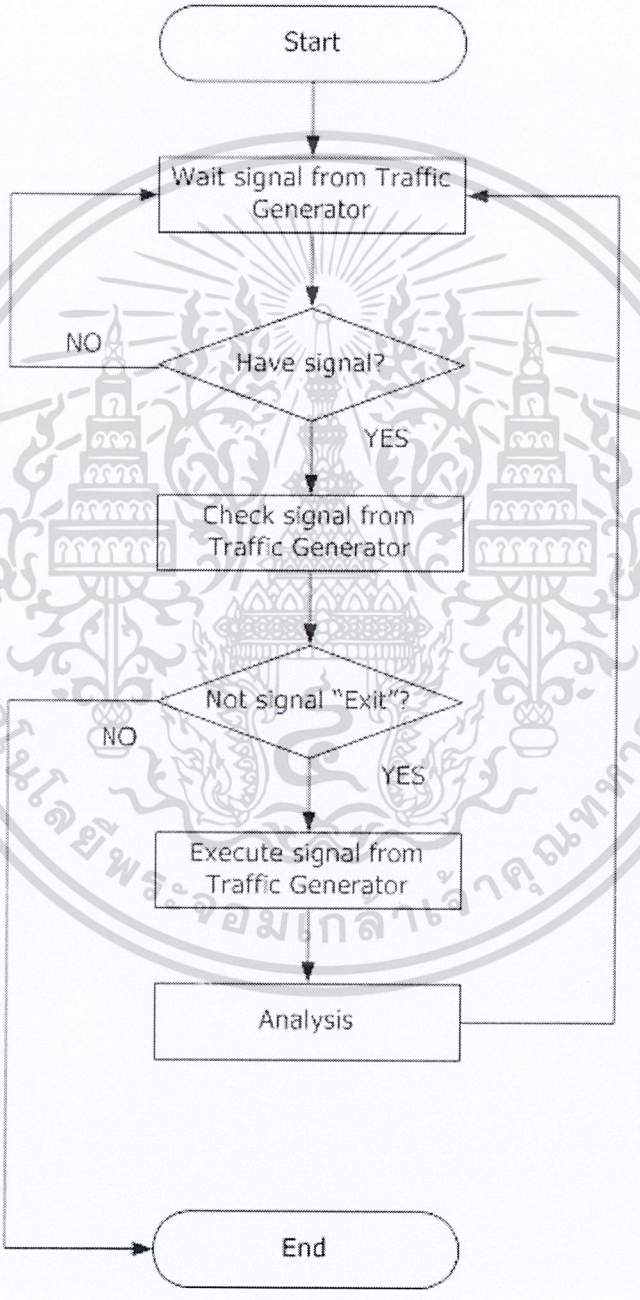


รูปที่ 6-2แสดงขั้นตอนการทำงานของ Traffic Generator

2. Receiver

หลักการทำงานก็คือจะเปิดการทำงานไว้รอรับสัญญาณต่างๆที่ส่งมาจากฝั่ง Traffic Generator และเมื่อได้รับสัญญาณก็จะตรวจสอบดูว่าได้รับสัญญาณอะไร เพื่อที่จะนำมาใช้ในการควบคุมการทำงาน ซึ่งจะทำงานจนกว่าจะได้รับสัญญาณสิ้นสุดการทำงานของโปรแกรม และเมื่อได้รับสัญญาณสิ้นสุดการทำงานของโปรแกรมเมื่อใดก็จะเริ่มทำการวิเคราะห์การทำงานของไฟร์วอลและจัดทำเป็นรีพอร์ตเพื่อให้ผู้ดูแลหรือผู้ควบคุมระบบนำไปตรวจสอบดูว่าไฟร์วอลมีจุดบกพร่องตรงไหนและสามารถทำงานได้ตามที่ต้องการหรือไม่

แสดงขั้นตอนการทำงานของตัว Receiver



รูปที่ 6-3 แสดงการทำงานของ Receiver

6.2 ขั้นตอนการศึกษา

ในขั้นแรก ได้ทำการศึกษาการทำงาน ของ การติดต่อสื่อสาร บน เครือข่าย คอมพิวเตอร์ ศึกษาการทำงานPROTOCOL TCP/IP เพื่อ เข้าใจหลักการติดต่อสื่อสารบนเครือข่าย คอมพิวเตอร์

ในขั้นที่2ได้ทำการศึกษาการทำงาน ของ FIREWALL ชนิดต่างๆ เพื่อจะได้เข้าใจหลักการทำงาน ของ FIREWALL และ การCONFIG POLICY ให้กับ FIREWALL

ในขั้นที่3 ได้ทำการศึกษา วิธีการสร้าง PACKET ข้อมูล บนพื้นฐาน PROTOCOL TCP ในรูปแบบต่างๆ พร้อมกันนั้น ก็ได้ศึกษาการดักจับ PACKET ไปพร้อมกันด้วย

ในขั้นที่4 ได้ศึกษา PACKET ที่เป็นรูปแบบ การโจมตี เพื่อให้ปิดให้บริการ หรือ Dos เพื่อนำมาใช้ในการ ทดสอบ POLICY FIREWALL ไปในตัวอีกด้วย

6.3 ขั้นตอนการดำเนินงาน

- สร้าง โปรแกรม ส่วนที่ทำหน้าที่สร้าง PACKET ให้มีรูปแบบต่างๆได้
- สร้าง ส่วน รับPACKET เพื่อทำหน้าที่รับ PACKET ที่ส่งมาจากส่วน สร้าง PACKET
- สร้างส่วน รายงานผลเพื่อรายงานผลให้ user ทราบว่า POLICY ของFIREWALL เป็นดังที่ต้องการหรือไม่
- สร้างส่วนควบคุมให้คอยควบคุมการทำงานของทั้งสองฝั่ง
- เพิ่มเติมรูปแบบต่างๆให้กับ โปรแกรมให้โปรแกรม ตรวจสอบได้หลากหลายขึ้น
- ทดสอบการทำงาน กับ FIREWALL จริง และตรวจสอบข้อผิดพลาด เพื่อแก้ไข

บทที่ 7
สรุปและวิจารณ์ ผลการทดลอง

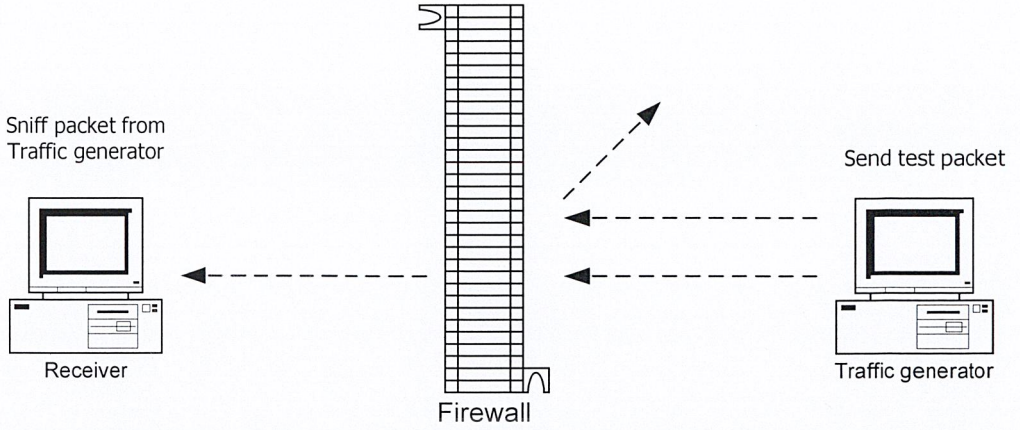
ในการออกแบบโปรแกรมนี้ ในการที่จะทำการออกแบบผู้ออกแบบได้มีการศึกษาข้อมูลต่างๆ ที่ใช้ในการออกแบบ ซึ่งทฤษฎีเหล่านี้จะช่วยในการสังเกตการทำงานของไฟร์วอลว่าไฟร์วอลควรจะมีการตอบสนองอย่างไร และจะช่วยในการออกแบบสร้างแพ็กเก็ตในรูปแบบต่างๆว่าแต่ละแพ็กเก็ตที่ส่งมานั้นโครงสร้างของแพ็กเก็ตควรเป็นไรบ้าง และควรมีฟอร์แมตอย่างไรในแต่ละรูปแบบของแพ็กเก็ต ซึ่งจะมีขั้นตอนในการศึกษาดังนี้ คือ

- ในขั้นแรก ได้ทำการศึกษาการทำงาน ของ การติดต่อสื่อสาร บน เครือข่าย คอมพิวเตอร์ ศึกษาการทำงานPROTOCOL TCP/IP เพื่อ เข้าใจหลักการติดต่อสื่อสารบนเครือข่าย คอมพิวเตอร์
- ในขั้นที่2ได้ทำการศึกษาการทำงาน ของ FIREWALL ชนิดต่างๆ เพื่อจะได้เข้าใจหลักการทำงาน ของ FIREWALL และ การCONFIG POLICY ให้กับ FIREWALL
- ในขั้นที่3 ได้ทำการศึกษา วิธีการสร้าง PACKET ข้อมูล บนพื้นฐาน PROTOCOL TCP ในรูปแบบต่างๆ พร้อมกันนั้น ก็ได้ศึกษาการดักจับ PACKET ไปพร้อมกันด้วย
- ในขั้นที่4 ได้ศึกษา PACKET ที่เป็นรูปแบบ การโจมตี เพื่อให้ปิดให้บริการ หรือ Dos เพื่อนำมาใช้ในการ ทดสอบ POLICY FIREWALL ไปในตัวอีกด้วย

ซึ่งเมื่อผู้ออกแบบได้ศึกษาข้อมูลเหล่านี้จนทำให้สามารถเข้าใจการทำงานต่างๆที่เกี่ยวข้องในการออกแบบแล้ว จึงได้มีการออกแบบโครงสร้างทั้งหมด แล้วจึงได้มีการสร้างตัวโปรแกรมชุดนี้ขึ้นมา และในการทำการสร้างตัวโปรแกรมชุดนี้ ได้มีการแบ่งทำงานเป็นขั้นตอนต่างๆ ซึ่งผู้ออกแบบมีขั้นตอนการทำงาน โดยแบ่งเป็นส่วนต่างๆ ดังนี้ คือ

- สร้าง โปรแกรม ส่วนที่ทำหน้าที่สร้าง PACKET ให้มีรูปแบบต่างๆได้
- สร้าง ส่วน รับPACKET เพื่อทำหน้าที่รับ PACKET ที่ส่งมาจากส่วน สร้าง PACKET
- สร้างส่วน รายงานผลเพื่อรายงานผลให้ user ทราบว่า POLICY ของFIREWALL เป็นดังที่ต้องการหรือไม่
- สร้างส่วนควบคุมให้คอยควบคุมการทำงานของทั้งสองฝั่ง
- เพิ่มเติมรูปแบบต่างๆให้กับ โปรแกรมให้โปรแกรม ตรวจสอบได้หลากหลายขึ้น
- ทดสอบการทำงาน กับ FIREWALL จริง และตรวจสอบข้อผิดพลาด เพื่อแก้ไข

จากสถาปัตยกรรม ของโปรแกรม ที่ได้ทำการออกแบบ นั้น สามารถทำงานได้ บนระบบปฏิบัติการ unix หรือ linux ซึ่งจากการออกแบบโปรแกรมชุดนี้ จะมีการแบ่งโปรแกรมนี้ออกเป็นสองส่วนด้วยกัน คือ ส่วนแรกเป็นตัวสร้างแพ็กเก็ตจำลองในรูปแบบต่างๆ (Traffic Generator) และส่วนที่สองเป็นตัวที่คอยดักจับแพ็กเก็ตที่ถูกส่งผ่านทางไฟร์วอลเข้ามา (Sniffer)



รูปที่ 7-1 แสดงภาพรวมการทำงานของโปรแกรมตรวจสอบไฟร์วอล

ซึ่งจากการออกแบบจะติดตั้งไว้ระหว่างไฟร์วอลส่วนแรกนั้นจะวางไว้ที่หน้าไฟร์วอล คือ ไว้ส่วนที่ส่งข้อมูลมาจากเครือข่ายภายนอกโดยผ่านไฟร์วอลเข้ามา ส่วนที่สองนั้นจะเปรียบเสมือนเครื่องที่อยู่ภายในเครือข่ายซึ่งมีไฟร์วอลเป็นตัวป้องกันอยู่ หลักการก็คือ จะเปรียบเสมือนเครื่องทั้งสองเครื่องมีการติดต่อระหว่างกันผ่านทางไฟร์วอล ซึ่งจะมีการจำลองรูปแบบการส่งข้อมูล การขอการเชื่อมต่อ เพื่อที่จะตรวจสอบดูว่าไฟร์วอลนั้นยอมให้การขอการเชื่อมต่อไปยังบริการใดๆบ้าง ซึ่งในการจำลองนั้นจะมีการจำลองหลักๆ ก็คือ

- การจำลองการขอการเชื่อมต่อโดยใช้โพรโตคอล ทีซีพี
- การจำลองการขอการเชื่อมต่อโดยใช้โพรโตคอล ยูดีพี
- การจำลองการส่งโพรโตคอล ไอซีเอ็มพี ที่บอกเหตุการณ์ในรูปแบบต่างๆ
- การจำลองการทำการโจมตีเพื่อปิดการให้บริการในแบบต่างๆ

โปรแกรมสามารถทดสอบ Policy ของ Firewall ว่าได้เปิด บริการใดให้ภายนอกเข้ามาใช้งานได้บ้าง หรือ Policy ที่ตั้งไว้ สามารถทดต่อการโจมตี Dos ในรูปแบบต่างๆได้หรือไม่ และเมื่อสามารถรู้ได้ว่าไฟร์วอลนั้นเปิดบริการที่พอร์ตใดบ้างก็จะมีการทดสอบลองโจมตีโดยผ่านพอร์ตที่เปิดให้บริการต่างๆดูเพื่อที่ทำการทดสอบว่าพอร์ตที่เปิดให้บริการนั้นเป็นจุดอ่อนที่จะทำให้เกิดการโจมตีเกิดขึ้นได้หรือไม่ อันจะทำให้ ผู้ใช้โปรแกรม สามารถ config Policy ให้กับ Firewall ได้อย่างมีประสิทธิภาพ และสามารถรู้จุดอ่อนหรือข้อผิดพลาดของไฟร์วอลและมีการแก้ไขข้อผิดพลาดได้ทันก่อนที่จะมีการนำไปใช้งานจริง

และเมื่อมีการทดสอบและนำโปรแกรมที่ออกแบบไว้นี้ไปทดลองใช้งาน ทางผู้ออกแบบได้ประสบปัญหาต่างๆในการทดสอบ และมีปัญหาบางอย่างที่ทางผู้ออกแบบไม่สามารถแก้ไขปัญหาได้ ซึ่งในขั้นตอนในการพัฒนาโปรแกรม ผู้พัฒนาประสบปัญหาดังที่จะลองยกตัวอย่างมาให้ทราบ ดังนี้

- โปรแกรมถูกแบ่งออกเป็นสองส่วน อยู่คนละฝั่งของ Firewall และทำการส่งข้อมูลระหว่างกันผ่าน Firewall ระหว่างนี้ถ้ามี ข้อมูลจากที่อื่นเข้ามาที่ เครื่องที่อยู่หลัง Firewall ซึ่งไม่ได้ส่งมาจากอีกส่วนหนึ่งของโปรแกรมอาจทำให้การตรวจสอบ Policy Firewall ผิดพลาดได้ เป็นต้น

จะทำให้ ผู้ใช้โปรแกรม สามารถ config Policy ให้กับ Firewall ได้อย่างมีประสิทธิภาพ และสามารถรู้จุดอ่อนหรือข้อผิดพลาดของไฟร์วอลและมีการแก้ไขข้อผิดพลาดได้ทันก่อนที่จะมีการนำไปใช้งานจริง

และเมื่อมีการทดสอบและนำโปรแกรมที่ออกแบบไว้นี้ไปทดลองใช้งาน ทางผู้ออกแบบได้ประสบปัญหาต่างๆในการทดสอบ และมีปัญหาบางอย่างที่ทางผู้ออกแบบไม่สามารถแก้ไขปัญหาได้ ซึ่งในขั้นตอนในการพัฒนาโปรแกรม ผู้พัฒนาประสบปัญหาดังที่จะลองยกตัวอย่างมาให้ทราบ ดังนี้

- โปรแกรมถูกแบ่งออกเป็นสองส่วน อยู่คนละฝั่งของ Firewall และทำการส่งข้อมูลระหว่างกันผ่าน Firewall ระหว่างนี้ถ้ามี ข้อมูลจากที่อื่นเข้ามาที่ เครื่องที่อยู่หลัง Firewall ซึ่งไม่ได้ส่งมาจากอีกส่วนหนึ่งของโปรแกรมอาจทำให้การตรวจสอบ Policy Firewall ผิดพลาดได้ เป็นต้น
- โปรแกรมสามารถทำงานได้กับ Gateway Firewall เท่านั้น เพราะส่วน sniffer อยู่บนเครื่องเดียวกับ Firewall และจับ ข้อมูลที่เข้ามาที่ interface เดียวกัน จะทำให้ข้อมูลที่จับได้ไม่ ผ่าน Firewall
- การโจมตีในรูปแบบใหม่ๆถูกคิดค้นขึ้นเรื่อยๆ ทั้งยังมีมากมายหลายรูปแบบจึงต้องมีการพัฒนาโปรแกรม ให้สามารถตรวจสอบจุดอ่อนต่างๆ ให้มากขึ้นไปเสมอ ฯลฯ

ดังนั้นขั้นตอนในการพัฒนาต่อไปอาจสร้างส่วนที่ทำหน้าที่ดักจับ ข้อมูลให้อยู่บนเครื่องเดียวกับ Firewall ได้โดยสามารถจับข้อมูลได้เฉพาะที่ถูกส่งผ่าน Firewall แล้วก็จะทำให้สามารถใช้โปรแกรมได้กับ Personal Firewall ที่มีการ config Policy ได้อีกด้วย นอกจากนั้นเรายังต้องควร พัฒนาโปรแกรมให้สามารถตรวจสอบให้ได้มากยิ่งขึ้นไปอีก เนื่องด้วย การโจมตี หรือจุดอ่อน ถูกค้นพบขึ้นทุกวัน จึงต้องมีการพัฒนาโปรแกรมอยู่เสมออีกด้วย

บรรณานุกรม

หนังสืออ้างอิง

[1] Joel Scambray, Stuart McClure, George Kurthz, "*Hacking Exposed*", McGraw-Hill, 2001

[2] W.Richard Stevens, "*UNIX Network Programming*", Prentice-Hell,1999

[3] Behrouz A. Forouzan. "*TCP/IP Protocol Suite*", McGraw-Hill, 2003

[4] สันติ ศรีลาศักดิ์, วรวุฒิ เที่ยงธรรม, "เจาะประเด็นงานเขียนโปรแกรมบนลีนุกซ์ ", ซีเอ็ดยูเคชั่น, 2542

เว็ปไซด์อ้างอิง

[5] **http://www.iptables.org**

[6] **http://www.thaicert.nectec.or.th**

[7] **http://www.codeguru.com**