ระบบตรวจจับผู้บุกรุกเครือข่ายบนยูนิกซ์

Network Intrusion Detection System (NIDS) for Unix

นายอภิชน ไวท์ยางกูร

นางสาวอังสนา วงศ์รัตนวิจิตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ระบบตรวจจับผู้บุกรุกเครือข่ายบนยูนิกซ์

Network Intrusion Detection System (NIDS) for Unix

โดย

นายอภิชน ไวท์ยางกูร

นางสาวอังสนา วงศ์รัตนวิจิตร

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมศาสตร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบตรวจจับผู้บุกรุกเครือข่ายบนยูนิกซ์

Network Intrusion Detection System (NIDS) for Unix

ผู้จัดทำ

1. นายอภิชน ไวท์ยางกูร รหัสประจำตัว 41014513
2. นางสาวอังสนา วงศ์รัตนวิจิตร รหัสประจำตัว 41014533

_____ อาจารย์ที่ปรึกษา

(อาจารย์ธนา หงษ์สุวรรณ)

_____ อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

ระบบตรวจจับผู้บุกรุกเครือข่ายบนยูนิกซ์

นาย อภิชน ไวท์ยางกูร 41014513
นางสาว อังสนา วงศ์รัตนวิจิตร 41014533
อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ประเด็นด้านความปลอดภัยนับเป็นหัวข้อสำคัญที่ต้องคำนึงถึงในการใช้ระบบคอมพิวเตอร์ในการดำเนินงานต่างๆในปัจจุบัน ผู้บุกรุกสามารถเข้ามาก่อความเสียหายให้ระบบได้หลายรูปแบบ ทั้งการเข้าถึงหรือเปลี่ยนแปลงข้อมูลที่เป็นส่วนตัวไม่ต้องการเปิดเผย หรือให้เปลี่ยนแปลงโดยบุคคลอื่น หรือการทำให้ระบบไม่สามารถทำงานหรือให้บริการได้ต่อไป ซึ่งการที่จะทำการเหล่านี้ได้ ผู้บุกรุกก็ต้องทำการศึกษาเบื้องต้นเกี่ยวกับระบบเป้าหมายเสียก่อน เพื่อที่จะรู้จุดอ่อนที่จะสามารถโจมตีได้

วิทยานิพนธ์ฉบับนี้ ศึกษาและพัฒนาระบบตรวจสอบผู้บุกรุกเครือข่ายขึ้นบนฐานของโพรโตคอลทีซีพี/ไอพี ซึ่งทำงานบนระบบปฏิบัติการยูนิกซ์ โดยได้ออกแบบเพื่อง่ายต่อการใช้งานและสามารถแก้ไขให้เข้ากับเครือข่ายเฉพาะได้ โดยที่มีความสามารถที่จะตรวจจับการกระทำต่างๆของผู้บุกรุกดังที่กล่าวถึงเบื้องต้น ซึ่งก็คือสามารถตรวจจับการโจมตีเพื่อให้ปิดบริการ การโจมตีในจุดอ่อนของแอพพลิเคชัน รวมถึงการการสำรวจระบบเบื้องต้นของผู้บุกรุก โดยจะทำงานโดยการตรวจสอบทราฟฟิกของเครือข่ายที่เครื่องทำงานอยู่ และวิเคราะห์ข้อมูลที่ได้ซึ่งจะดูทั้งจากชั้นเน็ตเวิร์ก ชั้นทรานสปอร์ต และชั้นแอพลิเคชัน เพื่อหาพฤติกรรมที่เข้าข่ายว่าเป็นการกระทำของผู้บุกรุก และแสดงผลรายงานให้ผู้ดูแลระบบทราบเพื่อเป็นข้อมูลในการจัดการปรับปรุงระบบ และหาทางป้องกันรักษาความปลอดภัยของระบบที่ดูแลต่อไป

Network Intrusion Detection System (NIDS) for Unix

Mr. Apichon Witayungkurn
Miss. Angsana Wongratanavijit
Mr. Thana Hongsuwan Advisor
Mr. Akkradach Watcharapupong Advisor

ABSTACT

Nowadays, security becomes the important issue that must be considered very carefully when use computer technology. The intruders could cause damage to system by many means for example changing important data or getting the secret information from the company or disable your system from providing service to customers.

This thesis concerns with the study and development of Network Intrusion Detection System based on TCP/IP stack of Linux Operating System. This system designed in a way that is easy to use and easy to adapt in different environments. It can detect the Denial of Service attacks (Dos) and the attacking on the bug of specific applications. Also, the system can identify the reconnaissance behavior of the intruder that are done when starting to attack the new network system. It analyzes network traffic to determine the suspect behavior of packets that could be an intrusion on network layer, transport layer and application layer. When Network Intrusion Detection System detect the intrusion, it alerts to administrator who use this information to investigate the intrusion and may re-configure the security or system to improve security.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้ด้วยดี เนื่องจากได้รับการแนะนำ สนับสนุน และให้คำปรึกษาเป็นอย่างดีจาก อาจารย์ธนา หงษ์สุวรรณ และอาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง รวมทั้งอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ให้การอบรมสั่งสอนวิชาความรู้แก่คณะผู้จัดทำมาโดยตลอด

และขอขอบพระคุณเป็นอย่างสูงสำหรับบุคคลที่สำคัญที่สุดที่ทำให้คณะผู้จัดทำมีวันนี้ คือ บิดามารดา ผู้เป็นที่เคารพรักยิ่งของคณะผู้จัดทำ ซึ่งท่านให้การอบรมสั่งสอน เลี้ยงดู และให้โอกาสในการศึกษาอย่างเต็มที่ จึงขอกราบขอบพระคุณมา ณ ที่นี้

สุดท้ายนี้ขอขอบพระคุณผู้ดูแลระบบคอมพิวเตอร์ภาควิชาวิศวกรรมศาสตร์ และสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่อำนวยความสะดวกในการใช้งานเครือข่าย และขอขอบคุณเพื่อนๆ ที่ให้ข้อคิดเป็น และเป็นกำลังใจให้เสมอมา

คณะผู้จัดทำ

สารบัญ

หน้าที่

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพประกอบ

หน้าที่

สารบัญภาพประกอบ (ต่อ)

หน้าที่

สารบัญตาราง

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันเราคงจะไม่สามารถปฏิเสธได้ถึงความสำคัญของคอมพิวเตอร์ที่เข้ามาเกี่ยวข้องกับชีวิตประจำวันในรูปแบบต่างๆ เช่น การซื้อขายสินค้าผ่านอินเทอร์เน็ต การท่องเว็บไซต์ การติดต่องาน ทำธุรกิจ และเมื่อเทคโนโลยีในด้านนี้เจริญขึ้น จากที่เป็นเพียงคอมพิวเตอร์เดี่ยวๆ เครื่องเดียวทำงาน ก็กลายเป็นโครงข่ายในการติดต่อสื่อสารขนาดใหญ่ จนเป็นอีกความจำเป็นหนึ่งในการทำงานซึ่งจะขาดไปไม่ได้ และเมื่อเป็นระบบที่มีความสำคัญมาก และมีการใช้อย่างแพร่หลาย ก็ทำให้เกิดผู้บุกรุกระบบขึ้นในรูปแบบต่างๆ เช่น บุกรุกเข้ามาเพื่อขโมยความลับทางธุรกิจ ขโมยข้อมูลลูกค้า หรือทำให้ระบบไม่สามารถให้บริการได้ ทำให้ช่องโหว่ของระบบถูกค้นพบมากขึ้น และพัฒนาการในการบุกรุกก็เติบโตไปพร้อมกับพัฒนาการของเทคโนโลยี เพื่อเป็นการป้องกัน และ ตรวจสอบการกระทำที่อาจก่อให้เกิดผลเสียหายต่อระบบของผู้ถูกบุกรุก จึงได้มีการออกแบบแนวคิด และสร้างเครื่องมือต่างๆขึ้นมากมาย ในหลายรูปแบบเพื่อดูแลและป้องกันระบบ หนึ่งในนั้นก็คือ ระบบป้องกันผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ (Network Intrusion Detection System หรือ NIDS) เป็นระบบที่จะตรวจสอบการบุกรุกทางเครือข่ายที่เกิดขึ้น โดยทำการตรวจสอบเน็ตเวิร์กทราฟฟิก เมื่อเกิดการโจมตีก็แจ้งเตือนไปยังผู้ดูแลระบบเพื่อนำไปปรับปรุงระบบ และยังเก็บข้อมูลไว้เพื่อใช้ในการตรวจสอบภายหลัง ซึ่งระบบนี้มีความสามารถในการตรวจจับที่หลากหลายได้หลายรูปแบบ แต่ในปริญญานิพนธ์นี้จะมุ่งเน้นการทำงานของ ระบบป้องกันผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ ไปในการตรวจสอบการโจมตีแบบการโจมตีเพื่อให้ปิดบริการ (Denial of Services หรือ DoS) การตรวจสอบการสำรวจระบบ และการโจมตีระบบในชั้นแอพพลิเคชัน

ปริญญานิพนธ์นี้จึงมุ่งเน้นการศึกษาประเภทต่างๆ ของการโจมตีแบบต่างๆ โดยจัดแบ่งประเภทของการโจมตี รวมถึงการศึกษาแนวทางตรวจสอบการโจมตีแต่ละประเภท เพื่อพัฒนาระบบตรวจจับบนระบบปฏิบัติการลีนุกซ์ให้สามารถตรวจจับการโจมตี ดังกล่าวได้อย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์ที่จัดทำขึ้นนี้ จัดทำภายใต้วัตถุประสงค์หลัก 4 ประการ ได้แก่

1) เพื่อศึกษารายละเอียดของการโจมตีแบบต่างๆ
2) เพื่อแบ่งประเภทของการโจมตีเพื่อให้ปิดบริการสำหรับสแต็กทีซีพี/ไอพี
3) เพื่อศึกษาแนวทางการตรวจสอบ การโจมตีเพื่อให้ปิดบริการสำหรับสแต็ก ทีซีพี/ไอพีประเภทต่างๆ
4) เพื่อศึกษารูปแบบการสำรวจระบบ และหาแนวทางการตรวจสอบการสำรวจระบบประเภทต่างๆ
5) เพื่อศึกษาแนวทางการตรวจสอบการโจมตีในชั้นแอพพลิเคชัน
6) เพื่อสร้างระบบต้นแบบในการตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์

1.3 ขอบเขตของปริญญานิพนธ์

ขอบเขตการทำงานของปริญญานิพนธ์นี้ ได้แก่

1) ศึกษาและ จัดแบ่งประเภทของการการ โจมตีประเภทต่างๆ ทั้งในชั้นเน็ตเวิร์ก ชั้นทรานสปอร์ต และชั้นแอพพลิเคชัน และศึกษาแนวทางในการตรวจสอบการโจมตีเหล่านี้
2) ออกแบบ และพัฒนาระบบต้นแบบการตรวจจับการโจมตีบนเครือข่าย แต่ละประเภทบนระบบยูนิกซ์โดยเขียนให้มีโครงสร้างที่แก้ไขเพิ่มเติมในอนาคตได้ง่าย (เขียนโดยใช้แนวคิดของการเขียนโปรแกรมแบบเชิงวัตถุ)
3) ระบบที่สร้างขึ้นต้องสามารถตรวจจับ แจ้งเตือน และเก็บข้อมูลที่เกิดการโจมตีได้อย่างถูกต้อง

1.4 ขั้นตอนการดำเนินงาน

1) ศึกษารายละเอียดเกี่ยวกับทีซีพี/ไอพีเบื้องต้น
2) ศึกษาเกี่ยวกับระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์
3) ศึกษารายละเอียดและการทำงานของการโจมตีแบบต่างๆ
4) จัดแบ่งประเภทของการโจมตี
5) กำหนดวิธีการตรวจจับการโจมตีแต่ละประเภท
6) ออกแบบโครงสร้างของระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์
7) ศึกษาการเขียนโปรแกรมผ่านเครือข่าย
8) ออกแบบขั้นตอนการทำงานของระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์
9) พัฒนาระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์
10) ทดสอบและปรับปรุงระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์

บทที่ 2
โพรโตคอลทีซีพี/ไอพี

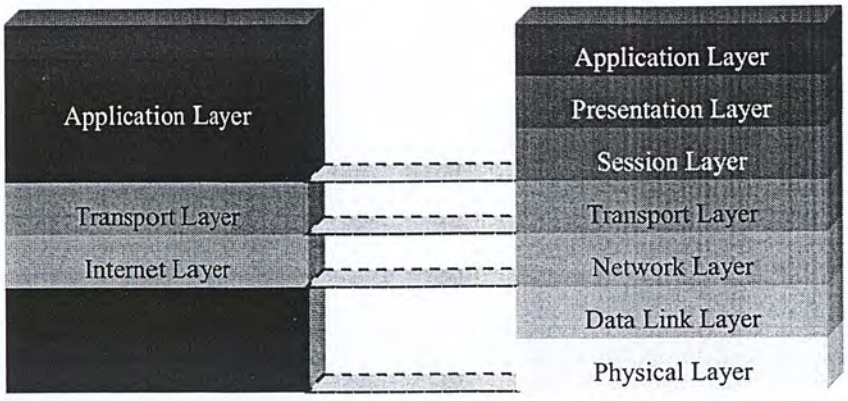
2.1 ความเป็นมาของโพรโตคอลทีซีพี/ไอพี

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

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

2.2 การเชื่อมต่อของโพรโตคอลทีซีพี/ไอพี (TCP/IP Linking)

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

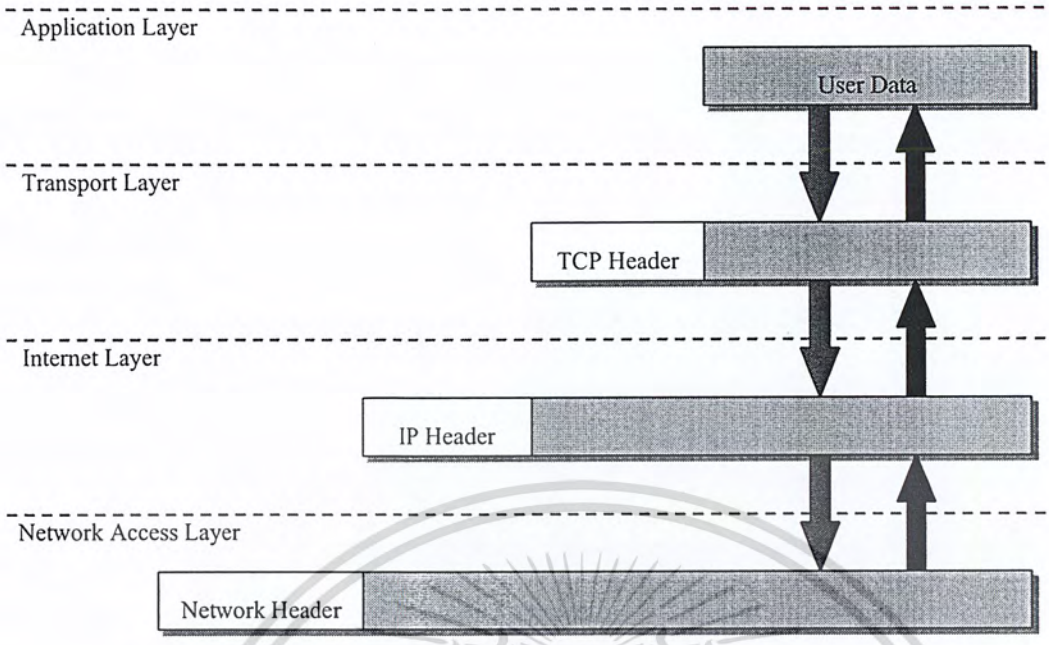


รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอพี

ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชันจนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังตารางที่ 2-1

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

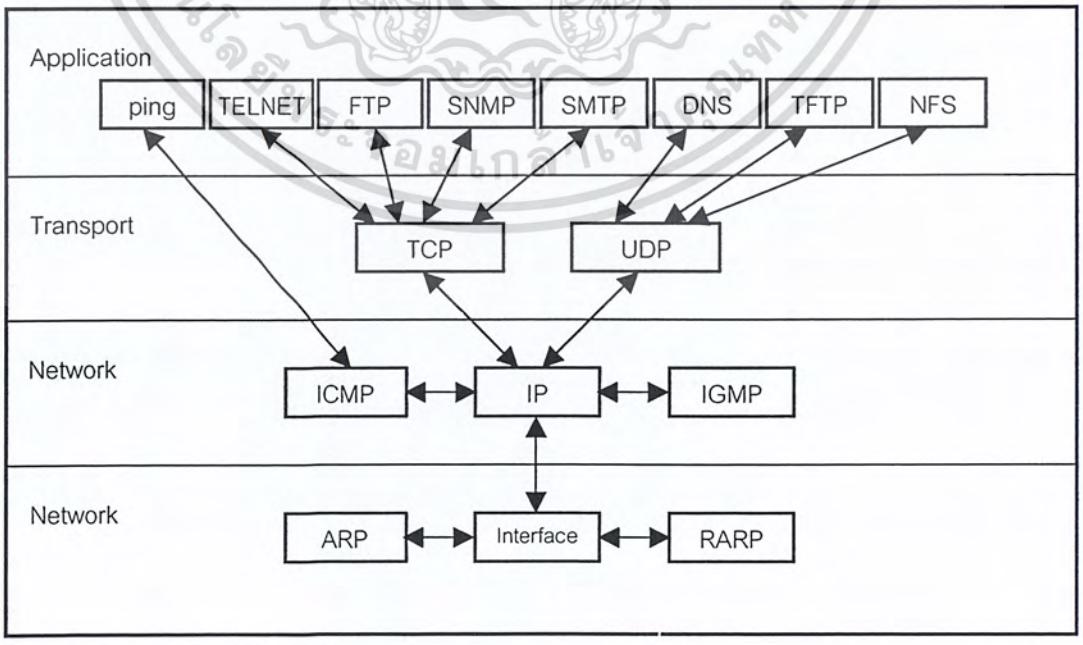
ตารางที่ 2-1 การทำงานของแต่ละระดับชั้นของทีซีพี/ไอพี



รูปที่ 2-2 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี

2.3 โพรโตคอลสแต็ก

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



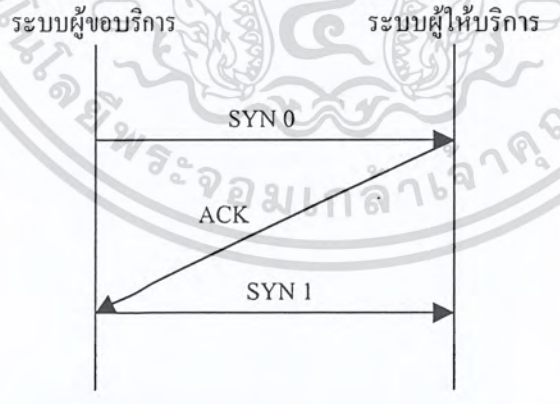
รูปที่ 2-3 โพรโตคอลสแต็กของทีซีพี/ไอพี

ไอพีซึ่งอยู่ในระดับชั้นเน็ตเวิร์กตามรูป เป็นแกนสำคัญของโพรโตคอลสแต็ก เนื่องจากทั้ง ทีซีพี และ ยูดีพี ต้องใช้ไอพีเพื่อเลือกเส้นทางส่งแพ็กเก็ต ในระดับชั้นเน็ตเวิร์กยังมีไอซีเอ็มพีสนับสนุนการทำงานของไอพีเพื่อรายงานข้อผิดปกติที่เกิดขึ้นเนื่องจากการส่งแพ็กเก็ต และมีไอจีเอ็มพีดูแลการจัดกลุ่มโฮสต์ในเครือข่ายมัลติคาสต์ ระดับชั้นทรานสปอร์ตมี 2 โพรโตคอลที่สำคัญ คือ ทีซีพีและยูดีพี แอพลิเคชันจะเลือกใช้ทีซีพีหรือยูดีพีตามลักษณะงาน โพรโตคอลระดับล่างถัดจากไอพีได้แก่โพรโตคอลระดับเน็ตเวิร์กอินเตอร์เฟสซึ่งกำหนดการทำงานตามเทคโนโลยีเครือข่ายที่ใช้งาน ในระดับชั้นนี้มี โพรโตคอลในชุดของ ทีซีพี/ไอพี ทำหน้าที่สนับสนุนการทำงานอยู่สอง โพรโตคอล คือ เออาร์พี และ อาร์เออาร์พี ทั้งสองโพรโตคอลทำหน้าที่แปลงค่าระหว่างแอดเดรสไอพี กับ ฮาร์ดแวร์แอดเดรส

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

2.4 โพรโตคอลทีซีพี (TCP: Transmission Control Protocol)

การทำงานที่สำคัญอย่างหนึ่งของโพรโตคอลทีซีพี คือ การทำ "3-way Handshake" ซึ่งเป็นกระบวนการเริ่มต้นในการสร้างการเชื่อมต่อในชั้นทรานสปอร์ต กล่าวคือ ในการติดต่อกันระหว่างระบบในเครือข่ายต้องมีการสร้างการเชื่อมต่อไปยังระบบที่ให้บริการก่อน โดยผู้ขอบริการส่งสัญญาณ SYN เพื่อขอบริการ จากนั้นผู้ให้บริการจะส่งสัญญาณ ACK เพื่อตอบรับการเชื่อมต่อที่ร้องขอมาจึงสามารถรับส่งข้อมูลกันได้ ดังรูปที่ 2-4



รูปที่ 2-4 แสดงการทำ 3-way Handshake

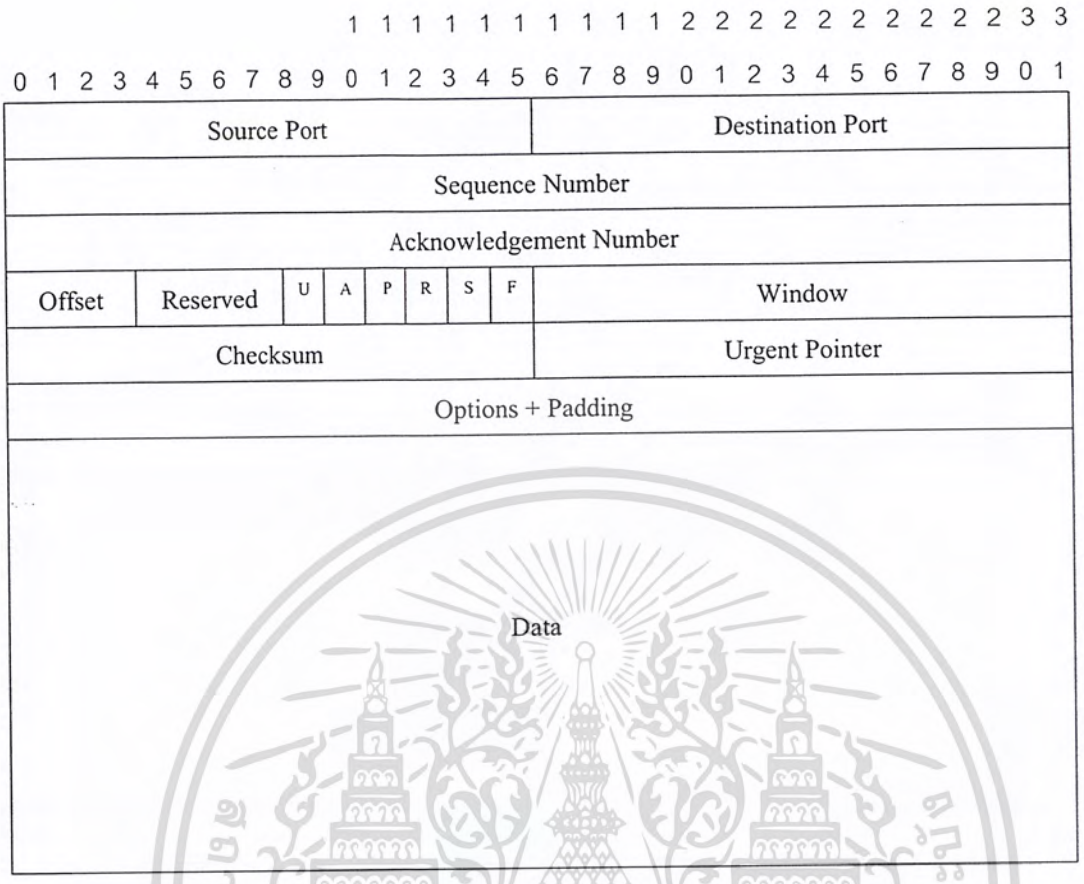
การเชื่อมต่อแบบ 3-way handshake นี้ เป็นการตรวจสอบความพร้อมของทั้งฝ่ายส่งและฝ่ายรับ และการกำหนดค่าเริ่มต้นของพารามิเตอร์ต่างๆ ของทั้งสองฝ่ายให้ตรงกัน หลังจากกระบวนการทำ 3-way handshake สิ้นสุด ทั้งสองฝ่ายจึงสามารถรับและส่งข้อมูลซึ่งกันและกันได้

ดังนั้นโพรโตคอลทีซีพีจึงเป็นโพรโตคอลที่มีการรับส่งข้อมูลแบบ "Connection Oriented" ทำให้การทำงานของทีซีพีมีความน่าเชื่อถือมากขึ้น หน้าที่การทำงานของทีซีพีในการรับส่งข้อมูลมีหน้าที่หลัก 6 ข้อคือ

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

<u>ส่วนประกอบของทีซีพีเฮดเดอร์</u>

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



รูปที่ 2-5 แสดงแพ็กเก็ตทีซีพี

2.5 โพรโตคอลยูดีพี (UDP: User Datagram Protocol)

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

<u>ส่วนประกอบของ UDP Frame</u>

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

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

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

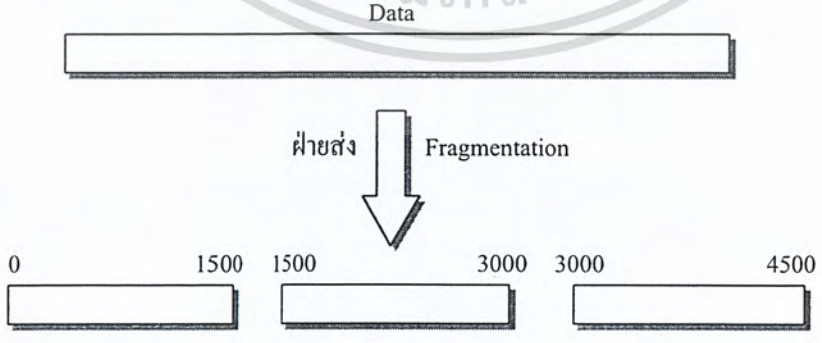
Source Port	Destination Port
Length	Checksum
Data	

รูปที่ 2-6 แสดงแพ็กเก็ตยูดีพี

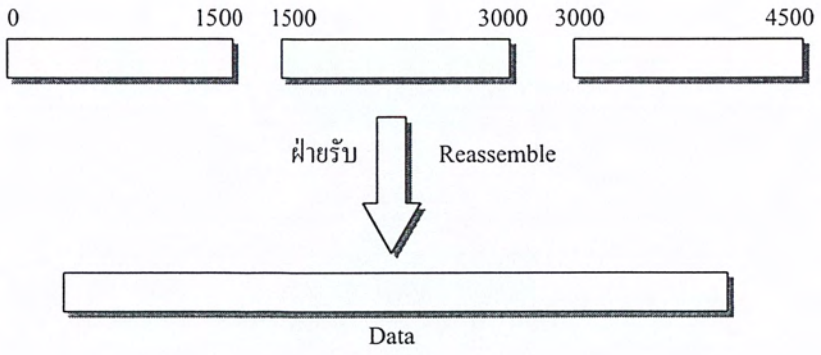
2.6 โพรโตคอลไอพี (IP: Internet Protocol)

โพรโตคอลไอพีเป็นโพรโตคอลที่จัดการเกี่ยวกับแอดเดรสของแต่ละแพ็กเก็ต เพื่อให้ส่งแพ็กเก็ตต่างๆ ไปยังเป้าหมายได้ถูกต้อง การทำงานของไอพีเป็นเพียงการส่งข้อมูลไปยังเครื่องเป้าหมายเท่านั้น ไม่มีการส่งสัญญาณขอบริการ หรือสัญญาณให้บริการระหว่างกันเหมือนทีซีพี เรียกว่าการเชื่อมต่อแบบคอนเนกชันเลส ซึ่งระบบทั้งสองตั้งสมมติฐานว่า การเชื่อมต่อระหว่างกันไม่มีความผิดพลาดเกิดขึ้นแน่

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 การรีแอสเซม-เบิ้ล (Reassemble) ซึ่งทำให้ได้ข้อมูลเหมือนที่ส่งมาจากเครื่องต้นทาง



รูปที่ 2-7 แสดงการทำแฟร็กเมนเตชัน



รูปที่ 2-8 แสดงการรีแอสเซมเบิ้ล

ส่วนประกอบของแพ็กเก็ตไอพี

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

   Bit 0-2 : บอกรายละเอียดการทำงานของแพ็กเก็ตนั้นๆ

   111 - Network Control

   110 - Internetwork Control

   101 - CRITIC / ECP

   100 - Flash Override

   011 - Flash

   010 - Immediate

   001 - Priority

   000 - Routine

   Bit 3 : บอกถึงลักษณะของดีเลย์

   0 = Normal Delay - มีดีเลย์ปกติ

   1 = Low Delay - มีดีเลย์ต่ำ

   Bit 4 : บอกถึงประเภทของทรูพุต

   0 = Normal Throughput – มีทรูพุตปกติ

   1= High Throughput – มีทรูพุตสูง

Bit 5 : บอกถึงประเภทของความน่าเชื่อถือ

0 = Normal Reliability - มีความน่าเชื่อถือพอประมาณ

1 = High Reliability - มีความน่าเชื่อถือสูง

Bit 6-7 : กันไว้ใช้ในอนาคต

4. *Total Length* : มีขนาด 16 บิต บอกถึงความยาวในดาต้าแกรมของไอพี
5. *Identification field* : เป็นตัวเลข 16 บิต เป็นค่าประจำตัวของไอพีนั้น โดยโฮสต์ที่ส่งเป็นผู้กำหนด และเพิ่มค่าขึ้นหนึ่งเมื่อมีการส่งดาต้าแกรมของไอพีใหม่ ซึ่งใช้ในการประกอบกลับ
6. *Flag* : เป็นตัวเลข 3 bit บอกลักษณะของแพ็กเก็ตว่ามีการแฟร็กเมนต์หรือไม่

   Bit 0 : สงวนไว้ ปกติเป็น 0

   Bit 1 : 0 = บอกว่าแพ็กเก็ตมีการแตกแพ็กเก็ตย่อย

   1 = บอกว่าแพ็กเก็ตไม่มีการแตกแพ็กเก็ตย่อย

   Bit 2 : 0 = บอกว่าแพ็กเก็ตนั้นเป็นแพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

   1 = บอกว่าแพ็กเก็ตนั้นยังไม่ใช่แพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย
7. *Fragment Offset* : เป็นค่าตัวเลข 13 บิต 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

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

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

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

รูปที่ 2-9 แสดงแพ็กเก็ตไอพี

2.7 โพรโตคอลเออาร์พี(ARP:Address Resolution Protocol)

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

Header	ARP/RARP datagram	FCS

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

รูปที่ 2-10 เออาร์พีดาทาแกรม

ส่วนประกอบของเออาร์พีดาทาแกรม

1. *Hardware 16 บิต* : กำหนดชนิดของฮาร์ดแวร์เครือข่ายที่เออาร์พีทำงานอยู่ ค่าใช้งานมีตัวอย่างดังต่อไปนี้

   1 อีเทอร์เน็ต
   4 โทเค็นริง
   5 เคออส(chaos)
   6 เครือข่าย IEEE 802
   7 อาร์คเน็ต
   12 โลคัลทอล์ค

2. *protocol 16 บิต* : ชนิดของโพรโตคอลที่ร้องขอใช้เออาร์พี

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

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

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

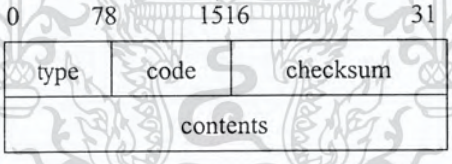
   ARP request (ค่าเท่ากับ 1)
   ARP reply (ค่าเท่ากับ 2)
   RARP request (ค่าเท่ากับ 3)
   RARP reply (ค่าเท่ากับ 4)

6. *Address* : ฟิลด์แอดเดรสเรียงลำดับจากฮาร์ดแวร์และเน็ตเวิร์กแอดเดรสของสถานีที่ร้องขอตามด้วยฮาร์ดแวร์และเน็ตเวิร์กแอดเดรสของสถานีที่ตอบรับ

2.8 โพรโตคอล ไอซีเอ็มพี (ICMP : Internet Control Message Protocol)

หน้าที่หลักของ โพรโตคอล ไอซีเอ็มพี คือการแจ้งหรือแสดงข้อความจากระบบ เพื่อบอกให้ผู้ใช้ทราบว่าเกิดอะไรขึ้นในการส่งผ่านข้อมูลนั้น ซึ่งปัญหาส่วนมากที่พบ คือส่งไปไม่ได้ หรือปลายทางรับข้อมูลไม่ได้ เป็นต้น นอกจากนี้โพรโตคอลไอซีเอ็มพียังถูกเรียกใช้งานจากเครื่องเซิร์ฟเวอร์ และเราเตอร์อีกด้วย เพื่อแลกเปลี่ยนข้อมูลที่ใช้ควบคุม ส่วนรูปแบบการทำงานของโพรโตคอลไอซีเอ็มพีนั้นจะทำงานควบคู่กับโพรโตคอลไอพีในระดับเดียวกัน และข้อความต่างๆที่แจ้งให้ทราบจะถูกผนึกอยู่ภายในข้อมูลของไอพี(ไอพีดาทาแกรม)อีกทีหนึ่ง ข้อความที่โพรโตคอลไอซีเอ็มพีส่งนั้น แบ่งออกได้ 2 แบบ คือ ICMP error message หรือข้อความแจ้งข้อผิดพลาด และ ICMP query หรือข้อความเรียกขอข้อมูลเพิ่ม ตัวอย่างกลไกการทำงานของโพรโตคอลไอซีเอ็มพี เช่น เมื่อมีการส่งผ่านข้อมูลจากผู้ใช้ไปยังปลายทางที่ไม่ถูกต้อง หรือขณะนั้นเครื่องปลายทางเกิดปัญหาจนไม่สามารถรับข้อมูลได้ที่เราเตอร์จะส่งข้อความแจ้งเป็นไอซีเอ็มพีเมสเซจ (ICMP message) ที่ชื่อ destination unreachable ให้กับผู้ส่งข้อมูล นอกจากนี้ตัว ข้อมูลที่แจ้งข้อความก็จะมีส่วนของข้อมูลไอพีดาทาแกรมที่เกิดปัญหาด้วย ดังนั้นเมื่อผู้ส่งข้อมูลได้รับข้อความแจ้งแล้วก็จะได้ทราบว่าจุดที่เกิดปัญหานั้นอยู่ที่ใด

ดังนั้นโพรโตคอลไอซีเอ็มพี จึงกลายมาเป็นเครื่องมืออย่างหนึ่งในการช่วยทดสอบเครือข่าย เช่น คำสั่ง ping ที่เรามักใช้ทดสอบว่าเครื่องเซิร์ฟเวอร์ที่ให้บริการหรืออุปกรณ์ที่ต่ออยู่ในเครือข่ายอินเทอร์เน็ตนั้นยังทำงานเป็นปกติหรือไม่ แล้วคำสั่ง ping มีการเรียกใช้งานโพรโตคอล ไอซีเอ็มพี แจ้งเป็นข้อความให้ทราบอีกต่อหนึ่ง



รูปที่ 2-11 ฟอร์แมตของ ไอซีเอ็มพี

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

2.9 ข้อบกพร่องของทีซีพีไอพี

2.9.1 ขาดกลไกด้านความปลอดภัย

สำหรับทีซีพีไอพีแล้วหากมีการกระตุ้นที่ถูกต้องตามโพรโตคอลแล้วจะต้องตอบรับเสมอหรือถ้าก็คือหากถูกถามจะต้องตอบเสมอ โพรโตคอลจะกำหนดไว้ชัดเจนว่าหากมีการกระตุ้นที่ถูกต้องตามโพรโตคอลจะต้องตอบรับ โฮสต์ผู้รับจะทำหน้าที่ตรวจสอบความถูกต้องเพียงว่ามันถูกต้องตามโพรโตคอลหรือไม่เท่านั้น ไม่ได้กำหนดในเรื่องอื่นเช่น ปริมาณมากเกินไปหรือไม่ ต่อเนื่องหรือไม่ ไม่มีกลไกในการตรวจสอบผู้ถาม และการจัดการกับการถามที่ผิดปกติ เช่นในกรณีปกติ หากโฮสต์ของเราได้รับ ICMP Echo Request มาโฮสต์ของเราไม่สามารถตรวจสอบได้ว่าผู้ส่งมีสิทธิ์ได้รับคำตอบหรือไม่ หรือผู้ส่งถามซ้ำมากเกินไปแล้ว แต่สิ่งที่ทำได้ก็คือตอบกลับไปด้วย ICMP Echo Reply เท่านั้น และต้องตอบกลับไปทุกกรณี ทั้งนี้เพราะกลไกด้ายนความปลอดภัยไม่ได้มีอยู่ในโพรโตคอล

ลักษณะนี้ต่างกับในระบบปฏิบัติการและแอพพลิเคชั่นต่างๆที่มีกลไกด้านความปลอดภัยโดยจะอนุญาตให้ผู้มีสิทธิ์ในระบบเท่านั้นที่จะสามารถใช้ทรัพยากรของระบบได้ แต่ตราบใดที่โพรโตคอลยังคงต้องทำงานอยู่บนทีซีพีไอพีและอาศัยทีซีพีไอพี ในการสื่อสารข้อมูล ช่องว่างของทีซีพีไอพีก็ยังคงอยู่และไม่สามารถแก้ไขได้ด้วยการรักาความปลอดภัยในระดับที่สูงขึ้น

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

2.9.2 การตอบรับเป็นสิ่งที่คาดหมายได้

แน่นอนที่สุดว่าการตอบรับจะต้องเป็นสิ่งที่สามารถคาดหมายได้ เพราะการตอบรับใดๆจะต้องเป็นตามข้อกำหนดในโพรโตคอลซึ่งเผยแพร่แก่สาธารณชนอยู่แล้ว หากการตอบสนองคาดหมายไม่ได้การสื่อสารสองทางก็คงไม่เกิดขึ้น ผู้บุกรุกจึงอาศัยการวิเคราะห์การตอบรับของเป้าหมายในสถานการณ์ต่างๆมาใช้ให้เป็นประโยชน์

ถึงแม้ผู้ที่จะใช้ประโยชน์จากการตอบรับในลักษณะนี้ได้ต้องมีความเข้าใจในโพรโตคอลเป็นอย่างดีและค่อนข้างละเอียด เพราะต้องทราบว่าการกระตุ้นแต่ละชนิดจะได้รับคำตอบจากเป้าหมายอย่างไร การตอบรับแต่ละแบบมีความหมายอย่างไร แต่มิได้เกินความสามารถของผู้บุกรุกทั้งหลายในการได้มาซึ่งข้อมูลที่สำคัญของเป้าหมาย จะพบได้ว่าการกระตุ้นและวิเคราะห์เป้าหมายเทคนิคต่างๆจะเป็นขั้นตอนที่สำคัญของผู้บุกรุกส่วนใหญ่จำเป็นต้องใช้ เพราะเป็นช่องทางที่ไม่มีการรักษาความปลอดภัย ในปัจจุบันเครื่องมือต่างๆที่ใช้ในการบุกรุกเจาะระบบได้พัฒนาไปมาก มีผู้ผลิตโปรแกรมที่ทำหน้าที่กระตุ้นและวิเคราะห์เป้าหมายแพร่หลายอยู่ทั่วไป ช่วยให้การเจาะระบบและสำรวจเป้าหมายยุ่งยากน้อยลง

2.9.3 การตอบรับกำหนดไว้ไม่ครอบคลุมทุกเงื่อนไข

ด้วยเป้าหมายของโพรโตคอลในเบื้องต้นคือ ความถูกต้อง ความมีเสถียรภาพและประสิทธิภาพของการสื่อสาร ดังนั้นเงื่อนไขข้อกำหนดต่างๆ ที่ระบุไว้นั้นก็เป็นไปเพื่อรับใช้วัตถุประสงค์ดังกล่าว จุดสำคัญอยู่ที่หากมีการสื่อสารที่ถูกต้องตามโพรโตคอล คอมพิวเตอร์ทั้ง 2 ฝั่งจะต้องอำนวยความสะดวกให้ทำการการสื่อสารด้วยความถูกต้อง มีเสถียรภาพและประสิทธิภาพสูงสุด แต่โพรโตคอลกลับละเลยใน

ส่วนนี้ หากมีการสื่อสารที่ไม่ถูกต้องตามโพรโตคอลแล้วจะจัดการต่อไปอย่างไร อาจจะมีกำหนดไว้ในจุดที่สำคัญ แต่ก็ยังคงเหลือเงื่อนไขอื่นๆ อีกมากมายที่โพรโตคอลไม่ได้ระบุ ตัวอย่างเช่น ทีซีพีกำหนดให้ส่ง SYN ในตอนสร้างการเชื่อมต่อและส่ง FIN ในตอนยกเลิกการติดต่อ ถามว่าหากมีการส่งทีซีพีที่มีทั้ง SYN และ FIN พร้อมกันจะเกิดอะไรขึ้น ผู้รับจะตอบรับกลับไปอย่างไร

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

บทที่ 3
การโจมตีของผู้บุกรุก

3.1 ขั้นตอนหลักของการบุกรุกเข้าสู่ระบบ

ขั้นที่ 1 : การตรวจสอบระบบจากภายนอก ผู้บุกรุกจะพยายามปลอมแปลงตัวเองเพื่อไม่ให้รู้ว่าตัวเองคือใครและมีเจตนาอะไร โดยการปรากฏตัวเป็นผู้ใช้ทั่วไป ซึ่งในขั้นนี้เราไม่สามารถตรวจพบได้ ผู้บุกรุกจะทำการใช้คำสั่ง 'whois' เพื่อค้นหาข้อมูลของเครือข่ายหลังจากนั้นผู้บุกรุกจะเข้าดูตาราง DNS ของเครือข่ายโดยใช้คำสั่ง 'nslookup' หรือ 'dig' เพื่อหาชื่อของคอมพิวเตอร์ในเครือข่าย

ขั้นที่ 2 : การตรวจสอบระบบจากภายใน ผู้บุกรุกใช้วิธีการบุกรุกต่างๆเพื่อจะค้นหาข้อมูลของเครือข่าย โดยอาจเข้าเว็บเพจของระบบและค้นหา สคริปต์ซีจีไอ (CGI scripts) หรือ อาจใช้วิธีการ 'ping' กวาดไปทั่วระบบเพื่อดูว่ามีเครื่องไหนที่ทำงานอยู่ หรือ อาจจะทำการใช้การสแกน ทีซีพี/ยูดีพี(TCP/UDP scan) บนเครื่องเป้าหมาย เพื่อดูว่ามีบริการอะไรที่เปิดรับอยู่ ณ จุดนี้สิ่งที่ผู้บุกรุกทำเป็นพฤติกรรมที่ปกติที่เกิดขึ้นบนเครือข่ายและยังไม่มีสิ่งใดที่จะระบุได้ว่าเป็นการบุกรุก แต่ว่าระบบตรวจสอบผู้บุกรุกทางเครือข่ายจะสามารถบอกได้ว่ามีบางคนกำลังตรวจสอบระบบของเราอยู่ แต่ว่ายังไม่ได้ทำอะไร

ขั้นที่ 3 : การเจาะระบบ ผู้บุกรุกจะเริ่มเจาะระบบผ่านทางช่องโหว่ของระบบ โดยผู้บุกรุกอาจจะพยายามส่ง สคริปต์ซีจีไอ ที่มี คำสั่งเชลล์ (shell) ในฟิลด์อินพุต (input fields) หรือ อาจพยายามส่งข้อมูลจำนวนมากเพื่อทำให้เกิดบัฟเฟอร์โอเวอร์รัน (buffer-overrun) หรือ อาจจะทำการหาล็อกอินที่ง่ายที่จะเดาพาสเวิร์ด เมื่อได้แอคเคาน์แล้วก็จะพยายามที่จะได้เป็น root/admin หรือ หาทางเข้ามาจากช่องโหว่ของโปรแกรมต่างที่ให้บริการของบนเครื่องเป้าหมาย

ขั้นที่ 4 : สร้างช่องทางลับและลบหลักฐาน ณ จุดนี้ผู้บุกรุกจะสร้างช่องทางลับในระบบ โดยเจาะเข้าเครื่องคอมพิวเตอร์ในระบบนั้น เป้าหมายหลักเพื่อลบหลักฐานของการบุกรุกและทำให้แน่ใจว่าสามารถที่จะกลับเข้ามาอีกได้ โดยผู้บุกรุกจะติดตั้ง 'toolkits' เพื่อให้สามารถเข้าระบบได้ หรือ ส่งม้าโทรจันไปฝังตัวไว้เพื่อสร้างพาสเวิร์ดแบ็กดอร์ (backdoor password) หรือ สร้างแอคเคาน์ขึ้นใหม่เลย ทำให้สามารถเข้ามาอีกเมื่อไหร่ก็ได้

ขั้นที่ 5 : แสวงหาผลประโยชน์ ผู้บุกรุกจะแสวงหาผลประโยชน์จากสิทธิ์ที่ตนได้รับโดยอาจขโมยข้อมูลลับ เช่น รหัสบัตรเครดิต หรือ ทำให้ระบบทำงานผิดพลาด หรืออาจใช้ระบบนี้ เพื่อเป็นทางผ่านไปยังระบบอื่น หรือโจมตีระบบอื่นเพื่อไม่ให้รู้ตัวตนที่แท้จริง ของผู้บุกรุก

3.2 รูปแบบการบุกรุก

3.2.1 การสำรวจระบบ (Reconnaissance) เป็นการสำรวจเป้าหมายขั้นต้นของผู้ที่จะทำการบุกรุกเพื่อที่จะนำข้อมูลนั้นมาวิเคราะห์เพื่อหาจุดอ่อนที่จะโจมตีของระบบ โดยทั่วไปจะไม่ส่งผลกระทบต่อระบบ เพราะไม่ใช่การโจมตี แต่เป็นกระบวนการเริ่มต้นของการโจมตี การสำรวจระบบนั้นทำได้หลายระดับ เพื่อวัตถุ

ประสงค์และเทคนิคที่แตกต่างกัน ซึ่งหากเรารู้และตรวจผลของการกระทำได้แต่เนิ่นๆ ย่อมทำให้สามารถเพิ่มความระมัดระวังและเตรียมพร้อมก่อนการถูกโจมตีจริง อีกนัยหนึ่งคือถ้าเราทราบว่าระบบของเรามีจุดที่สามารถถูกสำรวจได้ง่าย ก็จะได้ป้องกันส่วนนั้นได้ โดยการสำรวจนั้นสามารถแบ่งได้เป็น 3 แบบ คือ

1. *การสำรวจเน็ตเวิร์ก* เพื่อตรวจสอบเน็ตเวิร์กเป้าหมาย ผลที่ได้จะบอกถึงจำนวนโฮสต์ที่มีอยู่ในเน็ตเวิร์ก และ ลักษณะการใช้งานของโฮสต์
2. *การสำรวจโฮสต์* เพื่อวิเคราะห์รายละเอียดในแต่ละโฮสต์ที่คาดว่าจะเจาะเข้าไปได้โดยง่าย ว่ามีพฤติกรรมการใช้งานเป็นอย่างไร ลักษณะจะประกอบด้วยการสแกนที่สำคัญดังนี้คือ การสแกนพอร์ต การสแกนข้อมูลระบุแอพพลิเคชั่น การสแกนข้อมูลระบบปฏิบัติการ
3. *การสำรวจเพื่อหาแอพพลิเคชั่นเฉพาะ* ได้แก่ การสำรวจหาแอพพลิเคชั่นเฉพาะเจาะจงที่ทราบจุดอ่อนอยู่แล้ว หรือ การหาโปรแกรมโทรจัน หรือแบ็กดอร์ต่างๆ

3.2.2 การใช้ประโยชน์จากระบบ (Exploits) ผู้บุกรุกใช้ประโยชน์จากคุณลักษณะที่ซ่อนอยู่ หรือ ข้อผิดพลาดต่างๆ (bugs) เพื่อได้การเข้าถึงระบบ ตัวอย่างของลักษณะนี้ได้แก่ บัฟเฟอร์โอเวอร์โฟล (Buffer Overflow) การส่งคำสั่งที่มีลักษณะบุกรุก (Unexpected Combination) และ อินพุตที่ผิดปกติ (Unhandled Input) ซึ่งจะได้กล่าวถึงรายละเอียดต่อไป

3.2.3 การโจมตีเพื่อให้ปิดบริการ (Denial of Services : DoS) หมายถึง การกระทำใดๆ ที่ทำให้ระบบเป้าหมายไม่สามารถให้บริการบางอย่างได้ หรือไม่สามารถให้บริการต่อไปได้อีก โดยทั่วไปโจมตีที่พอร์ตของทีซีพี/ไอพี ซึ่งเชื่อมต่อกับบริการ (Services) ที่รองรับพอร์ตนั้นๆ ดังนั้นการโจมตีพอร์ตจึงเท่ากับการโจมตีบริการของระบบนั่นเอง ซึ่งการโจมตีแบบนี้ถือเป็นความเสียหายที่รุนแรงมากในระบบที่ต้องให้บริการ ข้อมูลที่รวดเร็ว

3.3 รายละเอียดการโจมตีแบบต่างๆ

การโจมตีแบบต่างๆ แบ่งประเภทตามชั้นของการสื่อสาร

3.3.1 ประเภทอยู่ในชั้นเน็ตเวิร์ก

3.3.1.1 การส่งแพ็กเก็ตจำนวนมาก (Amount of Packets Sending)

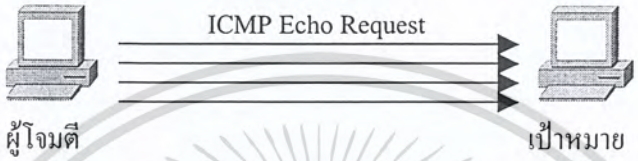
การโจมตีแบบนี้เป็นการส่งแพ็กเก็ตปริมาณมากเข้าไปยังระบบเป้าหมาย อาจทำให้ระบบเป้าหมายไม่สามารถให้บริการบางอย่าง หรือไม่สามารถทำงานต่อไปได้ ซึ่งแพ็กเก็ตที่ส่งออกไปนี้สามารถแบ่งออกได้เป็น

(1) แพ็กเก็ตข้อมูล (Data Packets)

การโจมตีวิธีนี้ทำได้โดยการส่งแพ็กเก็ตข้อมูลปริมาณมาก เมื่อข้อมูลเข้ามาสู่เครื่องเป้าหมายก็เก็บไว้ในบัฟเฟอร์ก่อนนำมาประมวลผลอีกครั้ง ดังนั้นหากส่งแพ็กเก็ตเข้ามาเป็นปริมาณมาก อาจทำให้บัฟเฟอร์ของเครื่องเป้าหมายไม่เพียงพอที่จะรองรับแพ็กเก็ตเหล่านั้นได้ทั้งหมด อาจทำให้เครื่องเป้าหมาย

ให้บริการได้ช้าลง หรือต้องหยุดการให้บริการไปเลย ตัวอย่างการโจมตีประเภทนี้เช่น Ping Flood Attack เป็นต้น

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



รูปที่ 3-1 แสดงการโจมตีด้วย Ping Flood Attack

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

14:49:43.217137 62.51.12.23 > 10.1.1.10 : icmp: echo request
14:49:43.217175 10.1.1.10 > 62.51.12.23 : icmp: echo reply
14:49:43.217195 62.51.12.23 > 10.1.1.10 : icmp: echo request
14:49:43.217219 10.1.1.10 > 62.51.12.23 : icmp: echo reply
14:49:43.217245 96.141.106.124 > 10.1.1.10 : icmp: echo request
14:49:43.217279 10.1.1.10 > 96.141.10.124 : icmp: echo reply
14:49:43.219017 172.19.251.18 > 10.1.1.10 : icmp: net 162.75.127.79 unreachable
14:49:43.237136 75.126.62.65 > 10.1.1.10 : icmp: echo request
14:49:43.237169 10.1.1.10 > 75.126.62.65 : icmp: echo reply
14:49:43.237193 75.126.62.65 > 10.1.1.10 : icmp: echo request
14:49:43.237216 10.1.1.10 > 75.126.62.65 : icmp: echo reply
14:49:43.237240 218.155.179.58 > 10.1.1.10 : icmp: echo request
14:49:43.237272 10.1.1.10 > 218.155.17.58 : icmp: echo reply

รูปที่3-2 รูปแบบแพ็กเก็ตที่เกิดขึ้นจากการโจมตีจาก Ping Flood Attack

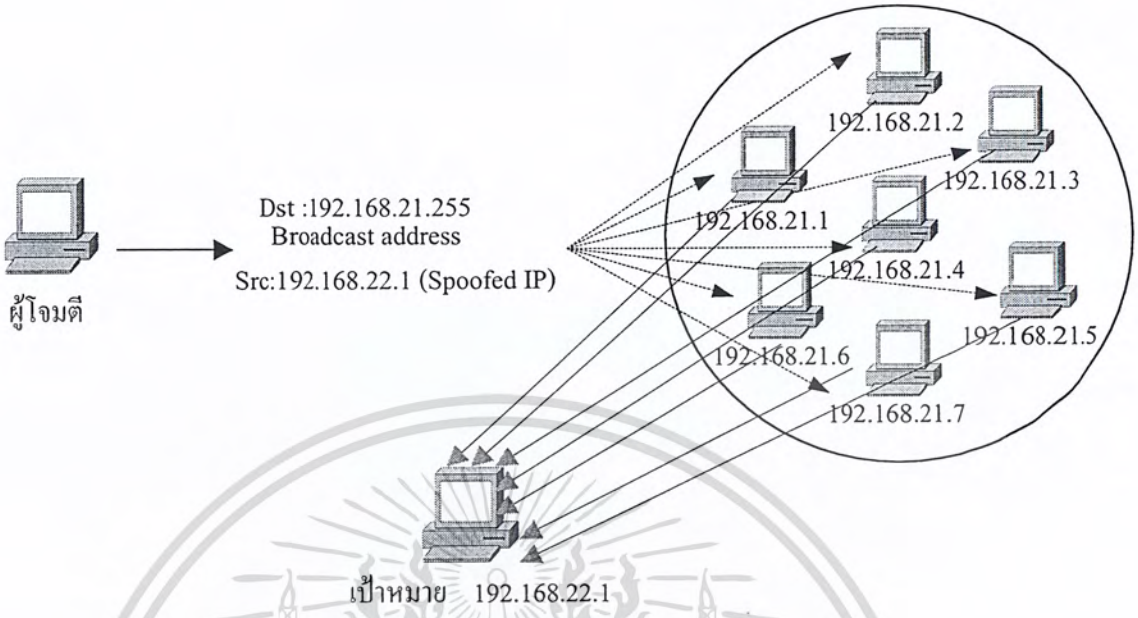
นอกจาก Ping Flood Attack จะสร้างความเสียหายแก่เครื่องเป้าหมายแล้วยังสร้างความเสียหายแก่ระบบเครือข่ายของเครื่องเป้าหมายด้วย เพราะการโจมตีวิธีนี้จะสร้างแพ็กเก็ตเป็นจำนวนมากขึ้นในเครือข่ายที่เครื่องเป้าหมายตั้งอยู่ ทำให้ระบบเครือข่ายเกิดความคับคั่งของข้อมูล(Congestion) อาจส่งผลให้เครือข่ายเป็นอัมพาตได้

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

อีกตัวอย่างหนึ่งของการโจมตีลักษณะนี้ ซึ่งมีลักษณะพิเศษขึ้นอีกเพื่อเพิ่มความสามารถในการโจมตี คือ สเมิร์ฟ (smurf)

สเมิร์ฟเป็นการโจมตีที่มีรูปแบบที่ให้ผลกระทบที่ขยายตัวออกไปในวงกว้าง (amplication effect) เป็นการปรับปรุงเทคนิคการส่งแพ็กเก็ตจำนวนมากให้ฉลาดกว่าเดิม โดยการขยายการโจมตีทำให้แฮกเกอร์มีเครื่องทุ่นแรง และเปิดโอกาสให้ผู้ที่มีแบนด์วิดธ์ต่ำสามารถทำการ flood เป้าหมายได้รุนแรงจากคุณสมบัติของการบรอดคาสต์ การส่งแพ็กเก็ตใดไปยังแอดเดรสบรอดคาสต์จะทำให้ทุกโฮสต์ในเน็ตเวิร์กได้รับแพ็กเก็ตนั้นอย่างทั่วถึง ดังนั้นหากมีโฮสต์ใดที่ส่ง ICMP Echo Request มายัง บรอดคาสต์ก็จะทำให้ทุกๆ โฮสต์ทั้งหมดที่อยู่ในเน็ตเวิร์กนั้นได้รับ ICMP Echo พร้อมกัน และด้วยข้อกำหนดของ ICMP เมื่อโฮสต์ได้รับ ICMP Echo Request จะต้องตอบกลับด้วย ICMP Echo Reply กลับไปยังผู้ส่งเสมอ ซึ่งเป็นไปได้ว่าหากมีการส่ง ICMP Echo ไปยังบรอดคาสต์เพียงครั้งเดียวก็จะได้รับ ICMP Echo Reply ตอบกลับเท่ากับจำนวนเครื่องที่อยู่ในเน็ตเวิร์กนั้นเลย ปรากฏการณ์นี้เรียกว่าการขยายสัญญาณ (Amplification) โดยการบรอดคาสต์ อัตราการขยายก็จะขึ้นอยู่กับปริมาณโฮสต์ที่อยู่ในเน็ตเวิร์กขณะนั้น วิธีการโจมตีนี้จะทำได้ต้องอาศัยเทคนิคการปลอม IP Address ด้วย โดยให้ IP Address ต้นทางของ ICMP Echo Request ที่ส่งไปนั้นเป็น IP Address ของเป้าหมาย

การโจมตีชนิดนี้ เป้าหมายอาจไม่จำเป็นต้องเป็นโฮสต์ใดโฮสต์หนึ่ง โดยการดัดแปลงการโจมตีให้เกิดผลเสียหายต่อเน็ตเวิร์กได้โดยการทำให้เน็ตเวิร์กท่วมไปด้วยแพ็กเก็ต เพียงแฮกเกอร์ส่งแพ็กเก็ตที่ใช้การโจมตีอย่างต่อเนื่อง และอัตราสูงก็จะทำให้เน็ตเวิร์กนั้นเต็มไปด้วยแพ็กเก็ตของ ICMP Echo Reply ซึ่งทำให้แพ็กเก็ตอื่น สำหรับใช้งานตามปกติไม่สามารถออกไปได้

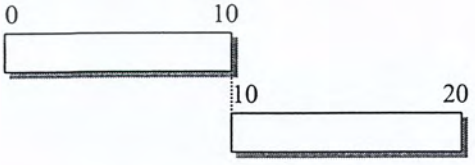


รูปที่ 3-3 แสดง การโจมตีด้วย Smurf

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

3.3.1.2 ความผิดปกติของแฟร็กเมนต์ (Abnormal Fragmentation)

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



รูปที่ 3-4 แสดงการรีแอสเซมบลีแบบปกติ

(1) การส่งแพ็กเก็ตที่มีลำดับผิดปกติ (Abnormal Sequences of Packets Sending)

ปกติการส่งแพ็กเก็ตมักเรียงตามลำดับกันไป หากไม่เรียงลำดับก็ต้องรอจนกว่าแพ็กเก็ตก่อนหน้านี้มาถึง เพื่อเรียงลำดับแพ็กเก็ตที่เครื่องรับ แต่การโจมตีแบบนี้กลับส่งเฉพาะแพ็กเก็ตสุดท้าย เพื่อให้ระบบเป้าหมายรอแพ็กเก็ตก่อนหน้า และส่งไปเป็นปริมาณมากๆ เพื่อให้ระบบเป้าหมายไม่สามารถให้บริการอย่างอื่นได้

รูปที่ 3-5 แสดงแพ็กเก็ตสุดท้ายที่ต้องรอแพ็กเก็ตก่อนหน้า

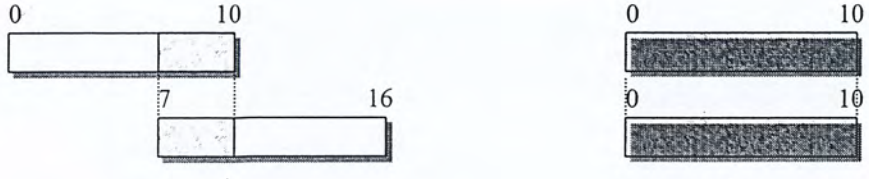
โดยปกติแล้วการโจมตีในรูปแบบนี้ผู้โจมตีจะแก้ไขข้อมูลในฟิลด์แสดงลำดับของแพ็กเก็ต (Fragment Offset) ของแพ็กเก็ตไอพี ซึ่งเป็นส่วนที่แสดงลำดับของข้อมูลหลังจากกระบวนการแฟร็กเมนต์เตชันโดยแก้ให้ส่งแพ็กเก็ตสุดท้ายหรือแพ็กเก็ตหลังๆ เพียงแพ็กเก็ตเดียวเลย ทำให้ระบบเป้าหมายต้องรอแพ็กเก็ตก่อนหน้านี้

(2) การส่งแพ็กเก็ตที่มีขนาดเหลื่อมกัน (Overlapped Packets' Size Sending)

ปกติแพ็กเก็ตที่ส่งมาต้องนำมาต่อกันที่ระบบเป้าหมายได้พอดี แต่การโจมตีแบบนี้เป็นการส่งแพ็กเก็ตที่มีขนาดเหลื่อมกัน หรือซ้อนทับกัน ทำให้ข้อมูลเมื่อมาต่อกันแล้วเกิดความผิดพลาด หรือไม่สามารถเชื่อมต่อกันได้

โดยปกติแล้วการโจมตีแบบนี้ ผู้บุกรุกสามารถแก้ไขข้อมูลได้ 2 แห่งใหญ่ๆ ได้แก่

- การแก้ไขข้อมูลที่ฟิลด์แสดงลำดับของแพ็กเก็ต (Fragment Offset) ของแพ็กเก็ตไอพี หลังจากกระบวนการรีแอสเซมเบิ้ล ซึ่งทำให้ลำดับในการส่งมีความผิดพลาด และอาจเกิดการเหลื่อมล้ำของแพ็กเก็ต กระบวนการรีแอสเซมเบิ้ลอาจเกิดปัญหาได้
- การแก้ไขฟิลด์แสดงความยาวของ (Total Length) ของแพ็กเก็ตไอพี หลังจากกระบวนการรีแอสเซมเบิ้ล ขนาดของแพ็กเก็ตที่มาต่อไม่พอดีกัน ทำให้ไม่สามารถรวมแพ็กเก็ตได้ หรือหากรวมได้ ข้อมูลที่ได้ก็ไม่ถูกต้อง



รูปที่ 3-6 แสดงการรีแอสเซมบลีแบบแพ็กเก็ตมีขนาดเหลื่อมกัน

(3) การส่งแพ็กเก็ตแบบวนลูป (Looping)

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

ตัวอย่างของการโจมตีแบบนี้ได้แก่ LAND ซึ่งเป็นโปรแกรมโจมตีที่มีลักษณะดังนี้ คือ

- หมายหมายแอดเดรสต้นทาง และแอดเดรสปลายทางเป็นค่าเดียวกัน คือ เป็นแอดเดรสของเครื่องเป้าหมาย
- หมายเลขพอร์ตต้นทางเท่ากับหมายเลขพอร์ตปลายทาง
- SYN Flag ถูกตั้งเสมือนขอเริ่มต้นการเชื่อมต่อ

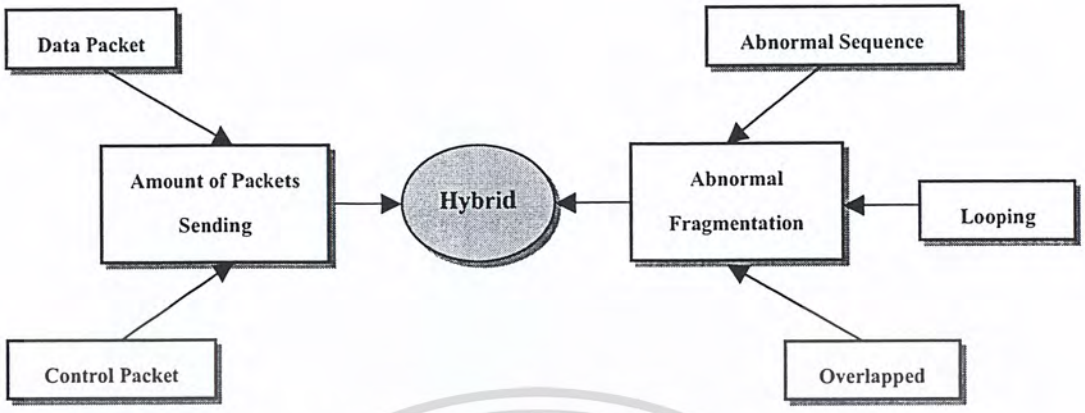
โดยปกติเมื่อมีการส่งสัญญาณ SYN มาก็ต้องมีการตอบกลับไปด้วยสัญญาณ SYN ACK ดังนั้นในที่นี้การตอบกลับจะตอบไปที่เครื่องเดิม ซึ่งในกรณีนี้ไม่มีกำหนดอยู่ในโพรโตคอลว่าควรทำอย่างไร โฮสต์จึงพยายามตอบสนองตามข้อกำหนดเท่าที่มีอยู่โดยการตอบกลับไปที่ไอพีแอดเดรส และพอร์ตต้นทางที่ถูกบุกรุกมา นั่นหมายถึงการตอบกลับเข้ามายังตัวเอง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

แพ็กเก็ตประเภทนี้เกิดจากการปลอมไอพีแอดเดรส ซึ่งถ้ามีการจัดการป้องกันการปลอมที่ดีพอก็สามารถป้องกันได้ แต่การป้องกันการปลอมนั้นสามารถบังคับใช้ได้ผลกับการปลอมข้าม เน็ตเวิร์กเท่านั้น ถ้าเป็นการปลอมในแชร์โดเมน(share domain)เดียวกันจะไม่สามารถทำได้ ซึ่งในปัจจุบันมีการปรับปรุง ทีซีพีสแต็กให้รัดกุมขึ้นจนการโจมตีแบบนี้ไม่เป็นผลกับคอมพิวเตอร์ที่ใช้ระบบปฏิบัติการที่ได้รับการแก้ไขแล้ว

รูปที่ 3-7 รูปแสดงการโจมตีด้วย Land Attack

3.3.1.3 แบบผสม (Hybrid)

คือ การโจมตีที่อาศัยวิธีการผสมกันระหว่างสามแบบแรกที่ได้กล่าวมาแล้ว ซึ่งทำให้เกิดผลเสียหายต่อระบบมากยิ่งขึ้น



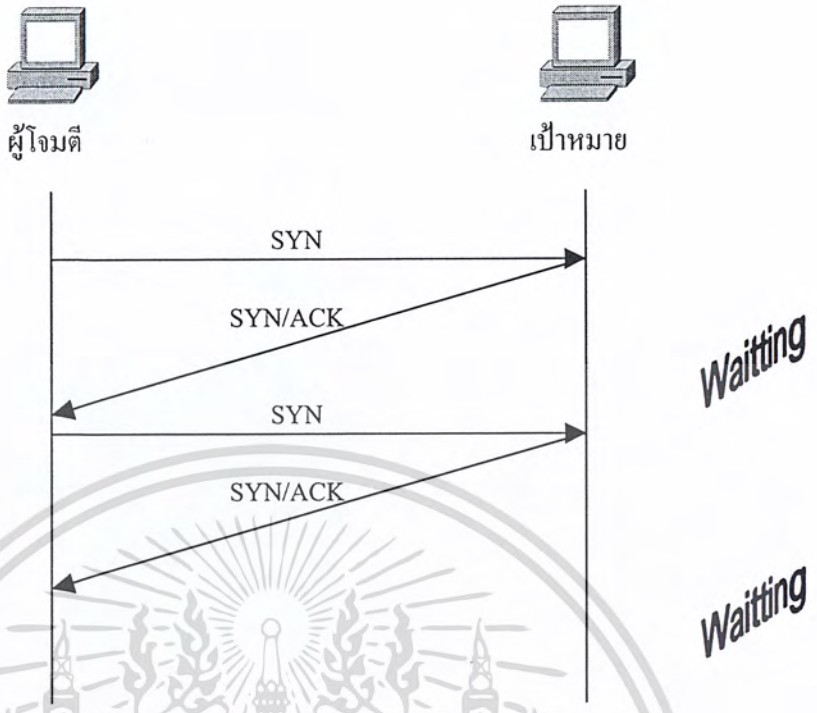
รูปที่ 3-8 แสดงแผนภูมิแสดงประเภทของการโจมตีเพื่อให้ปิดบริการสำหรับสแต็กทีซีพี/ไอพี

3.3.2 ประเภทอยู่ในชั้นทรานสปอร์ต หรือชั้นอินเทอร์เน็ต

3.3.2.1 การส่งแพ็กเก็ตสำหรับควบคุม (Control Packets)

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

ตัวอย่างของการโจมตีแบบนี้ ได้แก่ การทำ SYN Flooding ปกติการเชื่อมต่อแบบ 3-way handshake เป็นไปตามลักษณะที่ได้อธิบายไปแล้ว แต่ในการโจมตีลักษณะนี้ใช้วิธีทำให้การทำ 3-way handshake ไม่สมบูรณ์ กล่าวคือ เครื่องขอบริการส่งสัญญาณ SYN ไป แต่เมื่อได้รับสัญญาณ ACK จากเครื่องที่ให้บริการแล้ว ไม่ส่งสัญญาณ SYN ตอบกลับไป ทำให้เครื่องที่ให้บริการต้องเปิดการเชื่อมต่อรอการตอบกลับ ดังรูปที่ 3-9 ซึ่งการเปิดการเชื่อมต่อรอเอาไว้นี้ต้องใช้ทรัพยากรของระบบส่วนหนึ่ง และหากมีการส่งสัญญาณในลักษณะนี้มากๆ และทรัพยากรของระบบมีไม่เพียงพอ อาจทำให้ระบบไม่สามารถให้บริการอย่างอื่น หรือให้บริการกับผู้ร้องขอรายอื่นได้



รูปที่ 3-9 แสดงการส่งแพ็กเก็ตแบบ SYN Flood

สำหรับการโจมตีแบบ SYN Flood จะทำให้เกิดแพ็กเก็ตในระบบเครือข่ายในลักษณะดังรูปที่ 3-10

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

รูปที่ 3-10 แพ็กเก็ตที่เกิดขึ้นจากการโจมตีแบบ SYN Flood

ในปัจจุบันนี้การโจมตีแบบ SYN Flood ยังเป็นการโจมตีที่ได้ผลและหาทางป้องกันได้ยาก เนื่องจากยากที่จะแยกลักษณะของแพ็กเก็ตที่ใช้ในการโจมตีกับแพ็กเก็ตที่ขอเริ่มต้นเชื่อมต่อทั่วไป นอกจากนี้ไฟร์วอลล์หรือเราเตอร์ทั่วไปยังไม่สามารถป้องกันการโจมตีประเภทนี้ได้อย่างสมบูรณ์ หนทางที่เป็นไปได้คือการใช้ระบบตรวจจับผู้บุกรุกทางระบบเครือข่ายทำการตรวจจับการโจมตี เพื่อนำข้อมูลจากการโจมตีกลับไปตั้งค่าอุปกรณ์เราเตอร์หรือไฟร์วอลล์เพื่อป้องกันการโจมตีมายังเซิร์ฟเวอร์

3.3.2.2 การสำรวจระบบ

เป็นการสำรวจเน็ตเวิร์กโดยละเอียดเพื่อให้ได้มาซึ่งข้อมูลของทรัพยากรทุกๆอย่างที่มีอยู่บนเน็ตเวิร์กนั้น ไม่ว่าจะเป็นจำนวนโฮสต์ ลักษณะการเชื่อมต่อกันของโฮสต์ อุปกรณ์อื่นๆ ที่ไม่ใช่คอมพิวเตอร์ เส้นทางการเดินทางของแพ็กเก็ต ช่องทางการสื่อสารกับเน็ตเวิร์กภายนอก การได้มาซึ่งข้อมูลรายละเอียดเหล่านี้ยิ่งมากก็ยิ่งเป็นประโยชน์สำหรับผู้บุกรุก

3.3.2.2.1 Ping Sweep

การสำรวจวิธีนี้เป็นพื้นฐานเบื้องต้นปกติไม่ว่าจะเป็นผู้บริหารเครือข่ายระบบเองหรือผู้บุกรุกก็ใช้ การพิจารณาจากแพ็กเก็ตที่ปรากฏขึ้นในเน็ตเวิร์กเพียงอย่างเดียวนั้นยากที่จะแยกแยะได้ว่าเป็นการกระทำที่มุ่งร้ายหรือไม่ แต่อย่างน้อยที่สุดก็เป็นสัญญาณเตือนเบื้องต้นในการทำการตรวจสอบต่อไปว่าว่าผู้กระทำมีจุดมุ่งหมายอย่างไร วิธีการสำรวจนี้ใช้การส่ง ICMP Echo Request ไปยังโฮสต์ทุกตัวในเน็ตเวิร์ก เพื่อสำรวจดูว่าในเน็ตเวิร์กเป้าหมายนั้น มีโฮสต์ใดที่เปิดใช้งานอยู่บ้าง เมื่อโฮสต์ใดก็ตามได้รับ ICMP Echo Request เข้ามาก็จะต้องตอบกลับไปด้วย ICMP Echo Reply การตอบกลับมานี่เองจะเป็นสิ่งยืนยันได้ว่าโฮสต์นั้นเปิดใช้งานอยู่ การจะสังเกตว่า ICMP Echo Request แพ็กเก็ตนั้นต้องสงสัยว่าจะเป็นการสำรวจเน็ตเวิร์กหรือไม่ มิอาจพิจารณาได้จากแพ็กเก็ตเดียว จำเป็นต้องพิจารณาจากรูปแบบและความต่อเนื่องของหลายแพ็กเก็ต โดยจุดที่จะสามารถระบุได้ว่ามีความเป็นไปได้สูงคือ

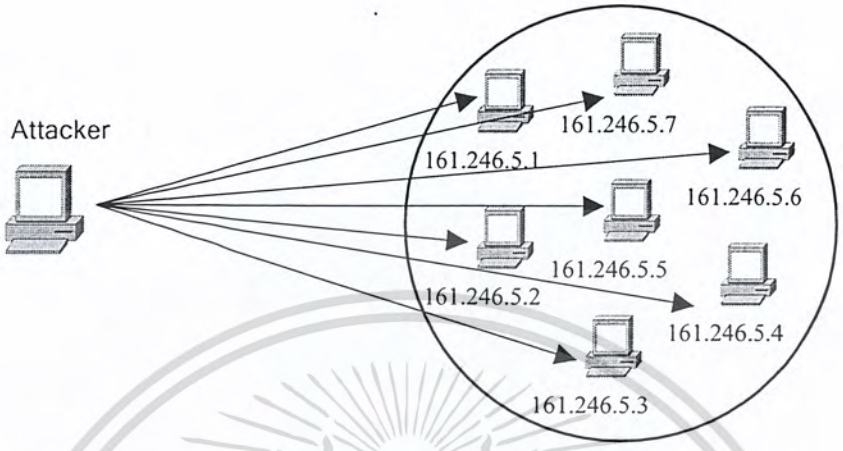
1. แพ็กเก็ตนั้นมาจากที่เดียวกันและส่งไปยังโฮสต์ปลายทางหลายๆโฮสต์ ซึ่งถ้าไอพีแอดเดรสมาจากภายนอกเน็ตเวิร์ก ก็มีความเป็นไปได้สูงที่จะเป็นการสแกนที่มุ่งร้าย เพราะบุคคลภายนอกไม่ควรสำรวจเน็ตเวิร์กผู้อื่นโดยไม่มีหน้าที่ และโดยพลการ
2. ช่วงเวลาระหว่างแพ็กเก็ตมีค่าน้อยมาก ปัจจัยในส่วนของเวลาระหว่างแพ็กเก็ตนั้นจะบอกได้ว่าแพ็กเก็คเหล่านั้นถูกส่งมาจากคำสั่ง Ping ปกติหรือถูกส่งมาจากเครื่องมือที่ใช้สำหรับสแกนโดยเฉพาะ หากระยะเวลาระหว่างแพ็กเก็ตนั้นมีค่าน้อยกว่า 0.5 วินาที ให้สันนิษฐานได้ว่าเป็นแพ็กเก็ตที่มาจากเครื่องมือแน่นอน

3.3.2.2.2 การสแกนพอร์ต

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

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

เพื่อการสแกนพอร์ตนั้นล้วนเป็นการดัดแปลงข้อกำหนดในโพรโตคอลมาใช้งานทั้งสิ้น อาจจะมีบางส่วนที่ใช้ช่องว่างที่ไม่มีกำหนดไว้ในโพรโตคอลเพื่อทำให้ได้ผลลัพธ์มาในที่สุด ดังจะอธิบายแต่ละวิธีต่อจากนี้



รูปที่ 3-11 แสดงการสแกนของผู้บุกรุก

TCP SYN Scan

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

FIN Scan

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

SYN/FIN Scan

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

Null Scan

วิธีนี้จะไม่ใช้แฟล็กใดๆในการสแกนเลย โดยส่งแพ็กเก็ตที่ไม่มีแฟล็กใดที่ถูกเซตไว้เลยไปยังเป้าหมาย เป็นการเซตแฟล็กทุกค่าให้เป็น 0 หมด ซึ่งแพ็กเก็ตลักษณะนี้จะไม่มีอยู่ในโพรโตคอล โดยทั่วไปการตอบสนองแพ็กเก็ตที่ไม่ได้อยู่ในโพรโตคอล จะมีการตอบรับที่ต่างกันออกไปตามแต่ประเภทของระบบปฏิบัติการ ดังนั้นนอกจากการใช้แพ็กเก็ตเหล่านี้เพื่อการสแกนพอร์ตแล้วยังสามารถนำแพ็กเก็ตเหล่านี้ไปใช้ในการตรวจสอบระบบปฏิบัติการของเป้าหมายได้อีกด้วย โดยการส่งแพ็กเก็ตที่มีแฟล็กซึ่งไม่อยู่ในข้อกำหนด การส่งแพ็กเก็ตลักษณะนี้ หากพอร์ตของเครื่องเป้าหมายปิดอยู่ การตอบรับจะเป็นการส่ง RST กลับไป

Xmas Scan

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

UDP Scan

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

3.3.3 ประเภทอยู่ในชั้นแอพพลิเคชัน

ส่วนใหญ่เกิดจากการใช้จุดอ่อนหรือข้อผิดพลาดของแอพพลิเคชันที่เครื่องเป้าหมายใช้อยู่ในการโจมตีเครื่องเป้าหมายเอง ไม่ว่าจะเป็นจุดอ่อนของระบบปฏิบัติการ หรือข้อผิดพลาดของซอฟต์แวร์ก็ตาม โดยสามารถแบ่งออกได้เป็นประเภทดังนี้

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

การส่งคำสั่งที่มีลักษณะบุกรุก (Unexpected Combination) เนื่องจากการที่โปรแกรมมักมีลักษณะเป็นการประกอบด้วยโค้ดหลายๆเลเยอร์ ซึ่งรวมถึงเลเยอร์ล่างสุดซึ่งก็คือ ระบบปฏิบัติการ ผู้บุกรุกจะอาศัยจุดนี้ส่งอินพุตที่ไม่มีความหมายสำหรับเลเยอร์หนึ่ง แต่มีความหมายกับอีกเลเยอร์หนึ่งเข้าไปในโปรแกรม ตัวอย่างที่เห็นได้ชัดได้แก่ ภาษา Perl ซึ่งมักจะส่งอินพุตต่อไปให้โปรแกรมอื่นเพื่อดำเนินการ

ต่อ ซึ่งถ้ามีการส่ง "| **mail < /etc/passwd** "จะทำให้เกิดการส่งไฟล์พาสเวิร์ดไปทางเมล์ได้ เนื่องจาก Perl จะให้ระบบปฏิบัติการเรียกโปรแกรมขึ้นมาจากอินพุตที่ส่ง โดยที่ระบบปฏิบัติการเองจะเห็น | pipe เป็นการเรียกโปรแกรมเมล์ด้วย

อินพุตที่ผิดปกติ (Unhandled Input) โปรแกรมส่วนมากเขียนขึ้นให้สามารถจัดการกับอินพุตที่ถูกต้องได้ โดยโปรแกรมเมอร์มักจะละเลยในส่วนของอินพุตอื่นๆที่ผิด ซึ่งจะเป็นช่องให้ผู้บุกรุกสามารถใช้ประโยชน์ได้

ตัวอย่างของการบุกรุกในชั้นแอพพลิเคชัน

- สคริปต์ซีจีไอ (CGI Script) : ซีจีไอเป็นโปรแกรมที่รู้กันถึงความไม่ปลอดภัยของมัน เป็นทางผ่านในการเข้าไปโจมตีจุดอ่อนในระบบส่วนอื่น โดยการส่งผ่านอินพุตที่ผิดรูปแบบเข้าไปเพื่อจุดประสงค์ในการไปสั่งการโดยตรงกับระบบ เช่น อาจให้เมล์ส่งไฟล์ของพาสเวิร์ดไปให้กับผู้บุกรุกได้ สคริปต์ซีจีไอที่เป็นที่รู้จักกันดี เช่น Phf.cgi , text counter , guestbook , webdist.cgi, files.pl เป็นต้น ซึ่งถ้าพบว่ามีการพยายามใช้หนึ่งในสคริปต์ซีจีไอเหล่านี้ หรือทั้งหมดโดยที่เราไม่ได้เป็นผู้ใช้เอง ก็อาจเป็นได้ว่ามีความพยายามบุกรุกเกิดขึ้นแล้ว
- การโจมตีเว็บเซิร์ฟเวอร์ (Web server attack) เว็บเซิร์ฟเวอร์จำนวนมากที่เขียนขึ้นเอง ซึ่งรวมถึง IIS 1.0 และ Netware 2.x มีช่องโหว่อยู่ซึ่งชื่อไฟล์สามารถรวมชุดของ "../" ในชื่อพาธเพื่อที่จะย้ายไปที่อื่นในระบบไฟล์ ทำให้สามารถเข้าถึงไฟล์ใดก็ได้ที่ต้องการ และอีกหนึ่งช่องว่างที่พบบ่อยก็คือบัฟเฟอร์โอเวอร์โฟลในฟิลด์ที่ต้องการ หรือในฟิลด์ HTTPอื่น
- Sendmail เป็นโพรเซสที่รันอยู่ในแบ็กกราวนด์ ช่องโหว่ส่วนใหญ่ของมันมักมีสาเหตุจากปัญหาของรีโมตบัฟเฟอร์โอเวอร์โฟลว์และการตรวจสอบอินพุตที่ไม่ดีพอ หนึ่งในช่องโหว่ยอดนิยมได้แก่ ช่องโหว่เรื่องการส่งไปป์(pipe) ของ Sendmail ซึ่งอยู่ในเวอร์ชัน 4.1 ซึ่งจะทำให้สามารถส่งไปป์คำสั่งที่ต้องการเอ็กซิคิวต์เข้าไปยังโพรเซส Sendmail ได้โดยตรง คำสั่งใดๆที่อยู่ต่อจากคำสั่ง DATA จะถูกเอ็กซีคิวต์(execute)โดย Sendmail ภายใต้สิทธิพิเศษของ bin
- การเจาะระบบทาง IIS 5 มีปัญหาเรื่อง Translate:f โดยผู้บุกรุกส่งอินพุตที่คาดไม่ถึงไปยังเซิร์ฟเวอร์ เพื่อให้มันส่งไฟล์บางไฟล์ซึ่งปกติไม่ควรเปิดเผย โดยใช้โพรโตคอลทีซีพี โดยการส่ง HTTP Get Request ที่ไม่เหมาะสมไปยังเว็บเซิร์ฟเวอร์เพื่อให้เว็บเซิร์ฟเวอร์เอ็กซิคิวต์สคริปต์ไฟล์บางสคริปต์ให้ แนวคิดของ request ดังกล่าวคือ การส่งเฮดเดอร์ (HTTP Header) พิเศษที่มี Translate:f อยู่ส่วนท้ายของไฟล์ และมีเครื่องหมาย "\" ต่อท้ายเข้าไปใน URL ที่ขอร้องไป โดยการส่งเท็กซ์ไฟล์ที่ประกอบด้วยสตริงข้อความต่างๆเข้าไปป์ ต่อเข้าไปยังเซิร์ฟเวอร์เป้าหมายก็จะได้ข้อมูลที่จริงๆแล้วไม่ต้องการเปิดเผยไปได้
- Server Side Include (SSIs) เป็นกลไกที่ทำให้เว็บเซิร์ฟเวอร์ให้บริการได้แบบโต้ตอบ ในเวลาจริง โดยที่ไม่ต้องอาศัยโปรแกรม บ่อยครั้งที่นักพัฒนาเว็บมักใช้มันเพื่อการเรียนรู้วันเวลาของระบบอย่างรวดเร็วหรือเพื่อเอ็กซิคิวต์คำสั่งที่เครื่องเว็บเซิร์ฟเวอร์และประเมินดูเอาต์พุตสำหรับการตัดสินใจในการควบคุมโฟลว์ของโปรแกรม ฟีเจอร์ต่างๆของ SSI (เรียกว่า tags)จำนวนมากได้แก่ echo,

include,fsize,flastmode,exec,config,odbc,email,if,goto,label, และ break tags ที่มีประโยชน์ที่สุดกับผู้บุกรุกก็คือ include,exec และ mail เทคนิคการโจมตีได้ถูกสร้างขึ้นโดยการเพิ่มโค้ดที่ใช้ tags ของ SSI เข้าไปในฟิลด์เว็บเพจที่จะต้องถูกประเมินค่าโดยเว็บเซิร์ฟเวอร์ ทำให้ผู้บุกรุกเอ็กซิคิวต์คำสั่งที่ต้องการบนเว็บเซิร์ฟเวอร์ได้และได้รับการเข้าถึงตัวเซิร์ฟเวอร์โดยตรง ตัวอย่างเช่น โดยการเพิ่ม SSI tag เข้าไปในฟิลด์ first name หรือ last name เมื่อมีการสร้างแอคเคานต์ เว็บเซิร์ฟเวอร์จะประเมินนิพจน์นั้นและพยายามที่จะรันมัน SSI tag ต่อไปนี้จะส่งกลับ xterm มาให้ผู้บุกรุก

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

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

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

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

3.4 โปรแกรมที่ใช้โจมตีเพื่อให้ปิดบริการ

โปรแกรมที่ใช้ในการโจมตีเพื่อให้ปิดบริการนี้เกิดขึ้นมากมาย และนับวันจะเพิ่มรูปแบบมากขึ้นเรื่อยๆ แต่โปรแกรมในปัจจุบันยังคงมีรูปแบบการโจมตีไม่มากไปกว่าที่ได้กล่าวมาแล้ว ซึ่งโปรแกรมต่างๆ ที่ได้ศึกษามีดังตารางที่ 3-1

ชื่อโปรแกรม	ประเภท	โพรโตคอลที่ใช้	ลักษณะการโจมตี	ระบบที่มีปัญหา
1. syncflood	ส่งแพ็กเก็ตสำหรับการควบคุมปริมาณมาก	TCP	เป็นการส่งสัญญาณ SYN ไปขอเปิดการเชื่อมต่อ แล้วเมื่อได้รับ ACK ก็ไม่ส่งสัญญาณ SYN กลับไป ดังที่ได้อธิบายมาแล้ว	- Windows 95
2. oshare	แฟร็กเมนเตชันผิดปกติ (แพ็กเก็ตเหลื่อมล้ำกัน)	IP	เป็นการส่งแพ็กเก็ตที่เหมือนกันมาที่เครื่องเป้าหมาย ทำให้แพ็กเก็ตที่ส่งมาซ้อนทับกัน	- Windows 95/98 - Windows NT 4 + Service Pack 5 ลงมา
3. land	แฟร็กเมนเตชันผิดปกติ (ส่งแพ็กเก็ตแบบวนลูป)	IP	เป็นการส่งแพ็กเก็ตที่มีแอดเดรสต้นทางและปลายทางเป็นค่าเดียวกัน ทำให้เกิดการส่งแบบวนลูป	- Windows 95
4. smurf	แพ็กเก็ตปริมาณมาก	ICMP	เป็นการส่ง ICMP echo (ping) traffic จำนวนมากโดยการ Broadcast ไปยังเครื่องเป้าหมาย	- Windows 95
5. opentear	แฟร็กเมนเตชันผิดปกติ (ต้องรอแพ็กเก็ตก่อนหน้า)	UDP	เป็นการสร้างแพ็กเก็ตสุดท้ายมา เพื่อหลอกให้ระบบเป้าหมายรอแพ็กเก็ตก่อนหน้านี้	- Windows 95
6. pimp	แฟร็กเมนเตชันผิดปกติ (รีแอสเซมเบิ้ลแล้วมีปัญหา)	IP, IGMP	เป็นการสร้างแพ็กเก็ตที่เมื่อรีแอสเซมเบิ้ลแล้วได้ข้อมูลที่ไม่มีความหมาย เกิดเป็นขยะในระบบ	- Windows 95

ตารางที่ 3-1 ตัวอย่างโปรแกรมที่โจมตีเพื่อให้ปิดบริการ

ชื่อโปรแกรม	ประเภท	โพรโตคอล	ลักษณะการโจมตี	ระบบที่มีปัญหา
7. targa	แฟร็กเมนเตชัน (แพ็กเก็ตเหลื่อมล้ำกัน)	IP	เป็นการส่งแพ็กเก็ตที่เหมือนกันมาที่เครื่องเป้าหมาย ทำให้แพ็กเก็ตที่ส่งมาซ้อนทับกัน	- Windows 95/98 - Windows NT 4 + Service Pack 5 ลงมา
8. Gin	แพ็กเก็ตสำหรับควบคุมปริมาณมาก	ICMP	เป็นการส่ง ICMP Echo ปริมาณมากไปยังเครื่องเป้าหมาย	- Windows 95/98 - Modem
9. raped	แพ็กเก็ตสำหรับควบคุมปริมาณมาก	TCP	เป็นการส่งสัญญา SYN ปริมาณมากไปยังเครื่องเป้าหมาย	- Windows 95/98
10. stream	แพ็กเก็ตปริมาณมาก	TCP	เป็นการส่งแพ็กเก็ตทีซีพีปริมาณมากไปยังเครื่องเป้าหมาย	- Windows 95/98 - Modem
11. moya	แพ็กเก็ตสำหรับควบคุมปริมาณมาก	ICMP	เป็นการส่ง ICMP Echo (ping) ปริมาณมากไปยังเครื่องเป้าหมาย	- Windows 98
12. Kox	แฟร็กเมนเตชันผิดปกติ (แพ็กเก็ตเหลื่อมล้ำกัน)	IGMP	ส่งแพ็กเก็ตที่เหลื่อมกันไปยังเครื่องเป้าหมาย ทำให้เครื่องเป้าหมายไม่สามารถประกอบแพ็กเก็ตเหล่านั้นได้	- Windows 98/98SE - Windows 2000 build 2000
13. Sesquipe - daliam	แฟร็กเมนเตชันผิดปกติ	UDP	ส่งแพ็กเก็ตยูดีพีที่มีแฟร็กเมนเตชันผิดปกติไปยังเครื่องเป้าหมาย	- Windows 98/98SE
14. smurf	แพ็กเก็ตสำหรับควบคุมปริมาณมาก	ICMP	ส่งแพ็กเก็ต ICMP echo (ping) ไปยังเครื่องเป้าหมายเป็นปริมาณมาก	- Windows 95/98
15. WinArp	อยู่ในชั้นแอปพลิเคชัน	ARP	เป็นการส่งแพ็กเก็ตไปยังเครื่องปลายทาง โดยเครื่องปลายทางต้องกดปุ่ม “OK” สำหรับทุกแพ็กเก็ตที่เข้ามา	- Windows 95/98 - Windows NT 4.0 ลงมา

ตารางที่ 3-1 ตัวอย่างโปรแกรมที่โจมตีเพื่อให้ปิดบริการ (ต่อ)

ชื่อโปรแกรม	ประเภท	โพรโตคอลที่ใช้	ลักษณะการโจมตี	ระบบที่มีปัญหา
16. winnuke	อยู่ในชั้นแอปพลิเคชัน	NetBIOS (Port 139)	เป็นการส่ง OOB (Out Of Band) data ไปยังพอร์ต 139 (NetBIOS) ทำให้เกิดหน้าจอฟ้า ขาดการเชื่อมต่อกับอินเตอร์เน็ต และทำให้เกิดปัญหาในการติดต่อบนเครือข่าย ซึ่งจะเกิดกับ	- WFWG 3.11 - Win95 - WinNT 3.51 - WintNT 4.0 + Service Pack 4 ลงมา

ตารางที่ 3-1 ตัวอย่างโปรแกรมที่โจมตีเพื่อให้ปิดบริการ (ต่อ)

ซึ่งเห็นได้ว่าโปรแกรมที่ใช้โจมตีเพื่อให้ปิดบริการสำหรับโพรโตคอลสแต็กทีซีพี/ไอพี มีมากมาย ในที่นี้หลายโปรแกรมเป็นโปรแกรมเก่า จึงไม่สามารถทำลายระบบปฏิบัติการใหม่ๆ ได้ หรือไม่สามารถโจมตีระบบที่ได้อัปเกรดแล้วได้

นอกจากโปรแกรมเหล่านี้แล้ว ยังมีโปรแกรมอื่นๆ ที่ทำให้เกิดการปิดบริการมากมาย และนอกเหนือจากโปรแกรมเหล่านี้ยังมีโปรแกรมที่ทำการโจมตีในชั้นแอพพลิเคชันอีก เช่น coke, winARP, nuke เป็นต้น

บทที่ 4
ระบบตรวจจับผู้บุกรุกเครือข่าย

4.1 แนวความคิดพื้นฐานของระบบตรวจจับผู้บุกรุก

ข้อมูลที่เรามองเห็นได้โดยผ่านแอพพลิเคชั่นจะเป็นข้อมูลที่รับส่งกันตามปกติถูกต้องตามโพรโตคอลทุกประการ เพราะหากมีส่วนใดส่วนหนึ่งของข้อมูลนั้นเกิดผิดพลาด ไม่เป็นไปตามโพรโตคอล ไม่ว่าจะเป็นที่ชั้นใด ข้อมูลนั้นก็จะถูกดร็อปทิ้งไป หรือข้อมูลบางอย่างที่ถึงแม้ว่าจะเป็นข้อมูลที่ถูกต้องตามปกติ แต่เป็นกลไกการรับส่งกันเองของโพรโตคอลเลเยอร์ล่างเพื่อให้การสื่อสารสมบูรณ์ ก็จะไม่ถูกส่งขึ้นมาให้ผู้ใช้ได้รับรู้ ซึ่งการทำงานลักษณะดังกล่าวในมุมมองของผู้ใช้งานแล้วน่าจะถูกต้อง เพราะข้อมูลไม่เกี่ยวข้องก็ไม่น่าจะต้องส่งมาให้ผู้ใช้ได้พบเห็นแต่อย่างไร

ดังนั้นสิ่งที่ควรทำความเข้าใจเบื้องต้นก็คือ กิจกรรมต่างๆบนเน็ตเวิร์กที่ผู้ใช้เห็นและรับรู้นั้นเป็นเพียงส่วนที่ถูกกำหนดในแอพพลิเคชั่นว่าให้นำมาแสดงต่อผู้ใช้เท่านั้น สิ่งอื่นๆ ที่ผู้ใช้ไม่ได้เห็น มิได้หมายความว่าไม่มีกิจกรรมใดเกิดขึ้น ยังมีอะไรอีกมากมายที่เกิดขึ้นบนเน็ตเวิร์กหรือแม้กระทั่งบนเครื่องคอมพิวเตอร์ของเราโดยที่เราเองไม่รู้ตัวถึงแม้ว่าจะนั่งอยู่หน้าเครื่องตลอดเวลาก็ตาม อีกประการหนึ่งคือการสื่อสารของคอมพิวเตอร์นั้นมีระบบรักษาความปลอดภัยต่ำมาก โดยเฉพาะในระดับเน็ตเวิร์กเลเยอร์ หากใช้งานตามดีฟอลต์โดยไม่ได้มีการปรับแต่งเป็นพิเศษแล้วแทบจะไม่สามารถป้องกันตัวเองได้จากการติดต่อจากผู้อื่นเลย คือใครอยากจะส่งข้อมูลมาหาเครื่องเราก็สามารถทำได้ทันทีโดยที่เราหลีกเลี่ยงไม่ได้ สิ่งที่เราทำได้ก็คือเพียงแค่เลือกว่าจะนำข้อมูลนั้นไปใช้งานหรือไม่ หากไม่ใช่สิ่งที่ต้องการก็ดร็อปทิ้งไป หากว่าใช่สิ่งที่ต้องการก็นำไปใช้งาน แต่อย่างน้อยที่สุดก็ต้องรับเข้ามาก่อนเสมอ แทนที่จะสามารถเลือกรับเฉพาะข้อมูลที่ต้องการเท่านั้น ซึ่งแค่จุดอ่อนนี้เพียงจุดเดียวก็สามารถนำไปใช้ในการโจมตีเพื่อให้ปิดบริการ เครื่องคอมพิวเตอร์ทั่วไปได้ทันที

หากเปรียบเทียบระบบหรือคอมพิวเตอร์ของเราเสมือนบ้าน ก็จะเป็นบ้านที่ไม่มีประตู ทุกคนสามารถเข้าออกได้อย่างเสรี ระบบปฏิบัติการและแอพพลิเคชั่นในเครื่องของเราเป็นเจ้าของบ้านและข้อมูลที่สื่อสารกันไปมาบนเน็ตเวิร์กก็จะเป็นเหมือนคนเดินถนนทั่วไป เมื่อบ้านไม่มีประตูใครที่อยู่ข้างนอกอยากเข้ามาในบ้านก็เดินเข้ามาได้ตามปกติ เจ้าของบ้านนั้นมีหน้าที่เดินมาสอบถามทุกคนที่เข้ามาเพื่อให้ทราบว่าเป็นคนที่ต้องการติดต่อด้วยหรือไม่ หากไม่ใช่คนที่ติดต่อด้วยก็บอกให้เขากลับออกไป หากใช่ก็จะเชิญเข้ามาสนทนาด้วย เช่นหากแอพพลิเคชั่นของเรามีเฉพาะเมล์เซิร์ฟเวอร์ เจ้าของบ้านก็จะยินดีต้อนรับเฉพาะบุรุษไปรษณีย์เท่านั้น หากใครที่ไม่ใช่ก็จะไม่สนทนาด้วย ไม่ว่าจะเป็นคนดีหรือไม่ดี หรือเป็นผู้ที่ทำหน้าที่อื่นที่อาจจะเข้ามาผิดบ้านก็ตาม อย่างไรก็ตามเจ้าของบ้านหลังนี้ไม่สามารถห้ามไม่ให้คนอื่นเดินเข้ามาได้หรือแม้กระทั่งไล่คนที่ไม่ต้องการออกไปก็ทำไม่ได้ ไม่ว่าใครจะเข้ามาในบ้านเจ้าของบ้านก็ต้องคอยออกมาสอบถามทุกครั้งไป ถึงแม้ว่าจะเป็นคนเดิมๆ ที่พยายามจะเข้ามาซ้ำแล้วซ้ำอีกตลอดทั้งวัน

โดยส่วนใหญ่แอพพลิเคชั่นที่ให้บริการจะถูกออกแบบมาเพื่อให้บริการที่ดีที่สุดโดยไม่ได้ระวังอะไร เปรียบเสมือนคนที่มองโลกในแง่ดี ใครเข้ามาที่บ้านก็พยายามบริการอย่างดีที่สุด ดังนั้นหากใครจะ

กลั่นแกล้งเจ้าของบ้านก็จะทำได้ไม่ยากเย็น เช่นส่งคนเข้าไปในบ้านทีเดียวพร้อมกันหลายๆคนจนเจ้าของบ้านไม่มีเวลาไปทำงานอื่น (เทียบได้กับ Ping Flood) , ส่งคนเข้ามาในบ้านแต่พอเจ้าของบ้านถามอะไรก็ไม่ยอมตอบปล่อยให้เจ้าของบ้านรอเก้อ (SYN Flood) , ส่งคนหลายๆแบบมาหาเจ้าของบ้านเพื่อสืบว่าเจ้าของบ้านยินดีต้อนรับคนประเภทไหน (Port Scanning) เป็นต้น เมื่อแอพพลิเคชั่นไม่ได้ถูกออกแบบมาให้ระมัดระวังเรื่องความปลอดภัย หากถูกก่อกวนมากๆจนไม่สามารถมือได้ก็จะหยุดทำงานในที่สุด

หากเปรียบระบบเป็นเสมือนบ้านแล้ว IDS จะเป็นเสมือนยามรักษาการณ์ทำหน้าที่เป็นผู้ช่วยเจ้าของบ้าน เนื่องจากเจ้าของบ้านจะชำนาญเฉพาะเรื่องการบริการเท่านั้น กล่าวคือ ระบบตรวจจับผู้บุกรุก (Intrusion Detection System) จะช่วยเสริมจุดด้อยส่วนนี้ให้แข็งแรงมากยิ่งขึ้น โดยทำตัวเป็นยามที่ชำนาญในการวิเคราะห์คนผ่านเข้าออกโดยดูจากลักษณะของคนเหล่านั้น และรู้จักพฤติกรรมของอันธพาลหรือพวกก่อกวนเป็นอย่างดี หากใครมีพฤติกรรมต้องสงสัยก็จะรีบรายงานให้เจ้าของบ้านรู้ทันทีเมื่อได้รับรายงานแล้วจะดำเนินการอย่างไรต่อไปนั้นก็ต้องพิจารณาอีกที แต่อย่างน้อยที่สุดก็เป็นการป้องกันภัยล่วงหน้าสามารถรับรู้ถึงการพยายามบุกรุกหรือก่อกวนในทันทีที่เหตุการณ์เกิดขึ้น นับว่าระบบตรวจจับผู้ บุกรุกเป็นเครื่องมือสำคัญอย่างยิ่งที่จะรับมือกับการบุกรุกเข้ามา ของผู้ไม่หวังดี

4.2 ระบบตรวจจับผู้บุกรุก (IDS - Intrusion Detection System)

เป็นระบบจัดการความปลอดภัยสำหรับคอมพิวเตอร์ และ เครือข่าย ที่พยายามจะตรวจหา และเตือนภัยเมื่อมีความพยายามในการบุกรุกเข้ามาในระบบ หรือ เครือข่าย เป็นอีกเครื่องมือหนึ่งที่ใช้กันอย่างมาก และมีความสำคัญอย่างยิ่งในปัจจุบัน ถึงแม้ว่าเครือข่ายอาจมีการป้องกันการบุกรุกอยู่แล้วโดยใช้ไฟร์วอลล์ (Firewall) อย่างไรก็ตามไฟร์วอลล์ก็ยังไม่ใช่เครื่องมือที่จะป้องกันการบุกรุกได้โดยอัตโนมัติ จะต้องอาศัยผู้ที่บริหารกำหนดกฎให้เหมาะสมกับการใช้งาน และแม้จะมีกฎที่ดีแล้ว แต่ก็อาจไม่สามารถป้องกันการบุกรุกได้ การบริหารไฟร์วอลล์ที่ดีก็ควรจะมีการตรวจสอบย้อนหลัง และทดสอบการเจาะระบบเพื่อเป็นการทดสอบระบบอีกครั้ง ซึ่งตรงจุดนี้ ระบบตรวจจับผู้บุกรุกจะช่วยได้มาก เพื่อตรวจสอบแพ็กเก็ตต่างๆที่ผ่านเข้ามา ถ้าตรวจพบการบุกรุก ผู้บริหารก็สามารถนำข้อมูลที่ได้ไปปรับปรุงกฎให้รัดกุมยิ่งขึ้น

ระบบตรวจจับผู้บุกรุก แบ่งเป็น 2 ประเภท คือ

1.ระบบตรวจจับผู้บุกรุกในโฮสต์ (Host-based Intrusion Detection System)

2.ระบบตรวจจับผู้บุกรุกเครือข่าย (Network Intrusion Detection System)

โดยระบบที่จะศึกษาและทำขึ้นมาในที่นี้คือ ระบบตรวจจับผู้บุกรุกเครือข่าย

4.3 ความหมายของการตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์

ระบบตรวจจับผู้บุกรุกเครือข่าย (Network Intrusion Detection System หรือ NIDS) เป็นแขนงหนึ่งของระบบตรวจจับผู้บุกรุก (Intrusion Detection System หรือ IDS) โดยเน้นไปทางการตรวจจับทางเครือข่ายคอมพิวเตอร์เป็นหลักซึ่งก็หมายถึงว่าการตรวจสอบนั้นจะครอบคลุมทั้งเน็ตเวิร์ก

โดยระบบนี้จะทำการตรวจสอบแพ็กเก็ตต่างๆ บนเครือข่ายเพื่อดูว่ามีข้อมูลที่ผิดปกติ หรือว่ามีพฤติกรรมที่น่าสังสัยหรือไม่ ซึ่งก็คือการพยายามค้นหาแฮ็กเกอร์ที่กำลังพยายามเข้ามาในระบบ หรือ ปิด

การให้บริการของระบบ (a denial of service attack) โดยการนำแพ็กเก็ตต่างๆ ที่เข้ามาสู่ระบบ แล้วนำมาวิเคราะห์เปรียบเทียบกับกฎต่างๆ ที่ระบุไว้ว่าเป็นการโจมตีหรือการบุกรุก รวมถึงนโยบายขององค์กรก็นำมาพิจารณาด้วย เพื่อตรวจสอบว่ามีสิ่งผิดปกติเกิดขึ้นกับระบบหรือไม่ ตัวอย่างที่มักจะเกิดขึ้นคือ การส่งแพ็กเก็ตที่เป็นการร้องขอการเชื่อมต่อ (TCP connection requests (SYN)) สู่พอร์ต(port) ต่างๆ บนเครื่องเป้าหมาย หรือส่งแพ็กเก็ตจำนวนมากจนเครื่องเป้าหมายรับไม่ไหวทำให้ระบบต้องหยุดตัวลง โดยทั่วไปแล้วระบบตรวจจับผู้บุกรุกเครือข่ายนี้จะถูกติดตั้งบนเครื่องเดียวแต่ตรวจสอบและวิเคราะห์แพ็กเก็ตทั้งระบบเครือข่าย โดยจะต้องทำการติดตั้งบนระบบที่ใช้ฮับ (hub) ในการเชื่อมต่อ เพื่อที่จะสามารถรับข้อมูลทั้งหมดในช่องทางการสื่อสารได้ ถ้าเราติดตั้งระบบนี้บนสวิตช์ เราจะรับข้อมูลได้เฉพาะของเครื่องเราเท่านั้น ซึ่งก็จะทำให้ไม่ได้ประสิทธิภาพ

4.4 หลักการทำงานพื้นฐานของระบบตรวจจับผู้บุกเครือข่าย

4.4.1 การดักจับแพ็กเก็ตจากเครือข่าย (Packet Sniffer)

ระบบตรวจจับผู้บุกรุกเครือข่าย ทำงานโดยการใช้แหล่งข้อมูลจากเครือข่าย เป็นการดักจับแพ็กเก็ตที่ผ่านมาในเครือข่ายที่อยู่ในแชร์โดเมน(share domain)เดียวกัน และนำข้อมูลแพ็กเก็ตมาวิเคราะห์ และเมื่อตรวจพบลักษณะที่ตรงกับข้อมูลที่จัดว่าเป็นการบุกรุกอยู่ก็จัดการตามที่ตั้งไว้ต่อไปซึ่งอาจจะเป็นการเก็บข้อมูลลงล็อกไฟล์(log file) หรือการแสดงข้อความเตือนผู้ดูแลระบบ ซึ่งในส่วนของการที่ได้ข้อมูลมานั้นจะใช้หลักการของเครื่องมือที่ชื่อว่า แพ็กเก็ตสนิฟเฟอร์(Packet Sniffer หรือ Network Wire Tapping Device)

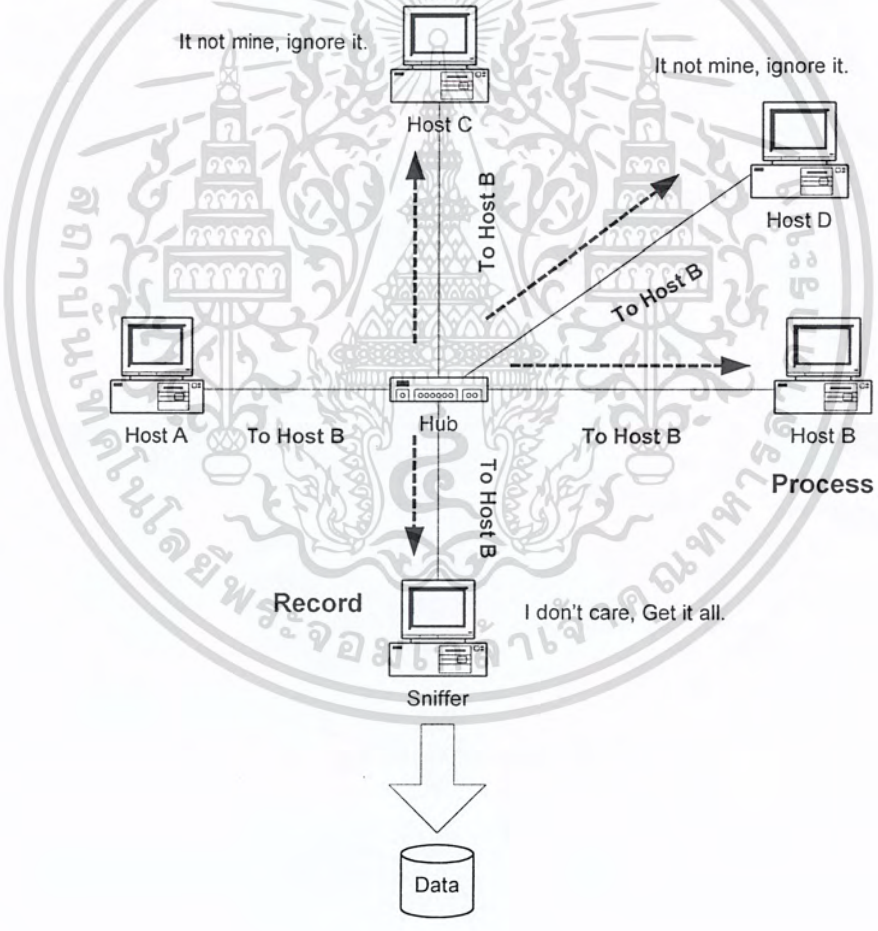
การที่สนิฟเฟอร์สามารถดักอ่านข้อมูลที่อยู่บนเน็ตเวิร์กได้นั้นมีสาเหตุที่สำคัญคือ ด้วยลักษณะของโพรโตคอลอีเทอร์เน็ตที่ใช้หลักการกระจายของข้อมูลไปยังทุกโฮสต์ที่อยู่ในเน็ตเวิร์กและอาศัยโฮสต์แต่ละตัวทำหน้าที่จะแนกการสื่อสารของตนเอง นั่นหมายความว่าข้อมูลทุกแพ็กเก็ตที่ใช้สื่อสารกันนั้นได้ถูกส่งไปยังโฮสต์ทุกตัว ซึ่งจะได้รับพร้อมกันและเหมือนกัน เพียงแต่การที่จะสื่อสารกันได้อย่างถูกต้องนั้นโฮสต์แต่ละตัวจะต้องมีกระบวนการที่สามารถรู้ได้ว่าข้อมูลแพ็กเก็ตใดเป็นของตัวเอง และข้อมูลแพ็กเก็ตใดมิใช่ของตนเอง ทุกๆแพ็กเก็ตที่กระจายลงบนเน็ตเวิร์กนั้นจะมีหมายเลขระบุชัดเจนคือ MAC Address หรือ เรียกอีกอย่างหนึ่งว่าอีเทอร์เน็ตแอดเดรส(Ethernet Address) ซึ่งเป็นสิ่งที่บอกว่าแพ็กเก็ตมาจากฮาร์ดแวร์ใดในเน็ตเวิร์ก ทำให้สามารถระบุได้ว่าแพ็กเก็ตนั้นส่งมาจากโฮสต์ใด และต้องการส่งให้โฮสต์ใด

MAC Address จะเป็นหมายเลขเฉพาะตามฮาร์ดแวร์ทุกชนิดที่ใช้การสื่อสารโดยโพรโตคอลอีเทอร์เน็ต และในทางทฤษฎีแล้วฮาร์ดแวร์ทุกชนิดจะไม่มี MAC Address ที่ซ้ำกัน โดยทั่วไป MAC Address จะถูกกำหนดตายตัวอยู่ใน Rom ของฮาร์ดแวร์และไม่สามารถเปลี่ยนแปลงได้โดยซอฟต์แวร์ แต่เนื่องจาก การใช้งานของฮาร์ดแวร์นั้นต้องควบคู่ไปกับไดรเวอร์ของฮาร์ดแวร์นั้นๆ ซึ่งโดยปกติแล้วไดรเวอร์จะถูกกำหนดให้ปฏิบัติตามโพรโตคอลอย่างเข้มงวดคือ

- ให้รับข้อมูลที่มี MAC Address เป็นของตนเองเท่านั้น (ห้ามอ่านข้อมูลผู้อื่น)
- ให้ส่งข้อมูลโดยใช้ MAC Address ของตนเองเท่านั้น (ห้ามปลอมเป็นผู้อื่น)

หากใช้ไดรเวอร์ตามปกติที่มากับฮาร์ดแวร์แล้วเครื่องคอมพิวเตอร์ก็จะทำงานอยู่ในรูปแบบปกติ และไม่สามารถวุ่นวายกับข้อมูลอื่นได้ แต่อย่างไรก็ตามไดรเวอร์ก็เป็นเพียงโปรแกรมประเภทหนึ่งเท่านั้นที่ทำหน้าที่จัดการกับการสื่อสารข้อมูลในระดับต่ำของฮาร์ดแวร์กับระบบปฏิบัติการ ดังนั้นถ้าเขียนไดรเวอร์ขึ้นมาใหม่ให้ไม่มีข้อจำกัดทางโพรโตคอล โดยละคุณสมบัติที่ให้รับข้อมูลเฉพาะที่มี MAC Address เป็นของตนเองเท่านั้นสนิฟเฟอร์ก็จะสามารถทำงานได้ โหมดการทำงานที่อนุญาตฮาร์ดแวร์รับข้อมูลของผู้อื่นเข้ามาได้โดยไม่มีการปิดกั้นเรียกว่า *โพรมิสคูอัสโหมด (Promiscuous Mode)* เป็นโหมดที่ทำให้ฮาร์ดแวร์อ่านข้อมูลดิบทั้งหมดบนเน็ตเวิร์กเข้ามาในเครื่องคอมพิวเตอร์ของตนเองได้โดยไม่สนใจว่าจะเป็นของใคร ส่งให้ใคร และเป็นการละเมิดข้อบังคับของโพรโตคอลหรือไม่

และเนื่องจาก ระบบตรวจจับผู้บุกรุกทางเครือข่ายทำงานโดยอาศัยหลักการนี้ ดังนั้นจึงทำให้เกิดข้อจำกัดในการใช้งาน คือ จะไม่สามารถตรวจจับนอกแชร์โดเมนของตัวเองได้ ดังนั้นถ้าเครือข่ายเป็นเครือข่ายที่ใช้สวิตช์ ระบบตรวจจับผู้บุกรุกจะไม่สามารถทำงานได้



รูป 4-1 รูปภาพจำลองการทำงานของสนิฟเฟอร์

4.4.2 ลักษณะของซิกเนเจอร์ที่ใช้ในการตรวจสอบ

การวิเคราะห์แพ็กเก็ตนั้น เราจะสนใจแพ็กเก็ตที่มีลักษณะซิกเนเจอร์ตรงกับที่มีข้อมูลอยู่ ซิกเนเจอร์แบ่งออกได้เป็น 3 ประเภท ดังนี้ คือ

1. สตริงซิกเนเจอร์ (string signatures) จะสนใจส่วนของข้อมูลในแพ็กเก็ต โดยหาส่วนของสตริงที่อาจบ่งถึงว่าเป็นการบุกรุกได้ ตัวอย่างของสตริงซิกเนเจอร์สำหรับระบบยูนิกซ์ เช่น "cat"++">/.rhosts" ซึ่งถ้าเจอในแพ็กเก็ตใด ก็อาจสรุปได้ว่าเป็นแพ็กเก็ตของการโจมตี

2. พอร์ตซิกเนเจอร์ (port signatures) ตรวจดูการเชื่อมต่อที่ต่อไปยังพอร์ตที่นิยมใช้ในการโจมตี ตัวอย่างของพอร์ตเหล่านี้ ได้แก่ telnet (ทีซีพี พอร์ต 23) , FTP (ทีซีพี พอร์ต 21/20) ,SUNRPC (ทีซีพี/ยูดีพี พอร์ต 111), และ IMAP (ทีซีพี พอร์ต 143) ซึ่งถ้าพอร์ตเหล่านี้ไม่ได้ถูกใช้โดยไซต์นั้นแล้ว แพ็กเก็ตที่เข้ามายังพอร์ตเหล่านี้ก็จะจัดว่าเป็นที่น่าสงสัยว่าอาจเป็นการบุกรุก

3. เฮดเดอร์ซิกเนเจอร์ (header signature) ตรวจสอบส่วนหัวของแพ็กเก็ตว่ามีส่วนประกอบที่ผิดปกติ ไม่สมเหตุสมผลหรือไม่ ตัวอย่างที่รู้จักกันดีที่สุดก็คือ Winnuke หรืออีกตัวอย่างก็คือ แพ็กเก็ตที่มีทั้งแฟล็กSYN และ FIN ตั้งไว้ซึ่งหมายความว่าผู้ส่งต้องการที่จะเริ่มและหยุดการติดต่อในเวลาเดียวกัน

4.5 ประโยชน์ของระบบตรวจจับผู้บุกรุกที่เป็นแบบทางเครือข่าย

ระบบตรวจจับผู้บุกรุกทางเครือข่ายมีหลายจุดแข็งซึ่งตัวตรวจจับผู้บุกรุกที่เป็นแบบโฮสต์เบสไม่สามารถทำได้โดยลำพัง ได้แก่การที่สามารถดักจับแพ็กเก็ตแบบเรียลไทม์ และวิเคราะห์มันได้ ในหัวข้อต่อไปนี้จะเป็นจุดแข็งที่แสดงให้ถึงความจำเป็นของการมีระบบตรวจจับผู้บุกรุกเป็นส่วนหนึ่งระบบรักษาความปลอดภัย

1. ค่าของการดูแลจัดการ ระบบรักษาความปลอดภัยทางเครือข่าย ให้การนำมาใช้กับระบบที่ต้องการเป็นไปได้ง่าย โดยสามารถติดตั้งเป็นจุดๆไปได้ และใช้ได้กับเครื่องเป้าหมายที่กว้าง ซอฟต์แวร์ที่ติดตั้งไม่จำเป็นต้องติดในทุกเครื่องเหมือนกับแบบโฮสต์เบส การที่จุดติดตั้งการตรวจจับมีจำนวนน้อย ทำให้ค่าการดูแลและจัดการเป็นไปอย่างมีประสิทธิภาพมากขึ้น

2. การวิเคราะห์แพ็กเก็ต ระบบตรวจจับผู้บุกรุกทางเครือข่าย จะตรวจสอบในส่วนหัวของแพ็กเก็ตเพื่อหาสัญญาณของการบุกรุก หรือการกระทำที่เป็นที่น่าสงสัย ซึ่งการโจมตีเพื่อปิดบริการในปัจจุบันหลายตัวจะถูกตรวจพบได้โดยการดูที่ส่วนหัวของมันเมื่อแพ็กเก็ตผ่านมาในเครือข่าย ตัวอย่างเช่น การโจมตีแบบ Land จะเป็นแพ็กเก็ตที่ถูกปลอมขึ้นมาให้มีไอพีแอดเดรสต้นทางและแอดเดรสปลายทางเหมือนกัน ซึ่งการโจมตีประเภทนี้จะถูกตรวจพบได้โดยง่าย เมื่อใช้ระบบตรวจจับผู้บุกรุกทางเครือข่ายทำงานแบบเรียลไทม์ ทั้งนี้การโจมตีที่ใช้แพ็กเก็ตแฟร็กเมนต์ เช่น Teardrop ก็สามารถถูกตรวจพบได้ในชั้นการวิเคราะห์แพ็กเก็ตเช่นกัน ซึ่งระบบตรวจจับผู้บุกรุกแบบโฮสต์เบสจะไม่สามารถตรวจพบการโจมตีประเภทเหล่านี้ได้ และนอกจากการตรวจสอบที่ส่วนหัวของแพ็กเก็ตแล้ว ระบบตรวจสอบผู้บุกรุกทางเครือข่ายยังสามารถตรวจสอบในส่วนเนื้อข้อมูลในแพ็กเก็ตเพื่อที่จะหาคำสั่งเฉพาะหรือรูปแบบโครงสร้างบางชนิดที่ใช้ในการโจมตี ซึ่งคำสั่งเหล่านี้จะเป็นตัวชี้บอกถึงว่าเป็นการโจมตี ไม่ว่าการโจมตีนั้นจะสำเร็จหรือไม่ก็ตาม ตัวอย่างเช่น ผู้บุกรุกทำการลองตรวจสอบการมีของโปรแกรม Back Orifice ในระบบที่ไม่

ถูกบุกรุกโดย Back Orifice ซึ่งการกระทำนี้จะไม่มีผลกระทบต่อระบบนี้ แต่เราจะสามารถรู้ได้ถึงความพยายามที่จะบุกรุก ซึ่งถ้าเป็นแบบโฮสต์เบสจะไม่สามารถตรวจสอบแบบข้อมูลในแพ็กเก็ตได้

3. การลบร่องรอย ระบบตรวจจับผู้บุกรุกคอมพิวเตอร์เครือข่ายใช้การตรวจจับแบบเรียลไทม์ และเมื่อตรวจจับได้แล้ว ผู้บุกรุกจะไม่สามารถลบหลักฐานนี้ทิ้งได้ สิ่งที่ตรวจจับได้ไม่ใช่เพียงแค่การโจมตีเท่านั้น แต่ยังมีข้อมูลอื่นที่อาจนำต่อไปได้ถึงผู้บุกรุกด้วย ปัญหาหนึ่งของระบบตรวจจับผู้บุกรุกทางคอมพิวเตอร์แบบโฮสต์เบสที่มักจะพบก็คือ ผู้บุกรุกเข้าใจและมีความรู้ในเรื่องล็อกไฟล์เป็นอย่างดี และมันก็มักจะเป็นที่แรกที่ผู้บุกรุกจะไปลบรอยของตัวเอง และเอาออก หรือทำลายข้อมูลส่วนนั้น

4. การตรวจจับและการตอบสนองแบบเรียลไทม์ ระบบตรวจสอบผู้บุกรุกทางเครือข่ายตรวจจับความผิดปกติ หรือการบุกรุกที่เกิดขึ้นอย่างทันต่อเหตุการณ์และรายงานในทันที ตัวอย่างเช่น ถ้าตรวจสอบพบว่าการโจมตีเพื่อให้ปิดบริการเกิดขึ้นโดยเกิดบนโพรโตคอลทีซีพี อาจจะมีการสั่งให้ระบบส่ง ทีซีพี รีเซตในทันทีเพื่อหยุดยั้งการโจมตีไว้ก่อนที่จะทำให้เกิดความเสียหายแก่ระบบมากขึ้น ในหลายๆ สถานการณ์ด้วยระบบแบบโฮสต์เบส การรับทราบถึงเหตุการณ์ที่เกิดจะเกิดขึ้นในภายหลังและบางทีอาจไม่มีการเตือนถึงเลยเนื่องมาจากที่ระบบได้เสียหายไปก่อนแล้ว ในการที่มีการเตือนแบบเรียลไทม์นั้นจะทำให้สามารถตอบสนองต่อเหตุการณ์ที่เกิดขึ้นได้อย่างทันท่วงที หรือถ้าไม่ต้องการก็สามารถรวบรวมข้อมูลไว้เพื่อวิเคราะห์ต่อในภายหลังได้

5. การตรวจจับเจตนาที่มุ่งร้าย ระบบจะมีประโยชน์มากในการต้องการตรวจจับถึงเจตนาของการกระทำ ถ้านำระบบตรวจจับผู้บุกรุกทางเครือข่ายไปไว้ก่อนไฟร์วอลล์ เราจะสามารถรู้ได้ถึงความพยายามในการที่จะโจมตีของผู้บุกรุกได้ แม้ว่าแพ็กเก็ตที่ต้องการโจมตีนั้นจะถูกปฏิเสธโดยไฟร์วอลล์ก็ตาม ถ้าเป็นระบบแบบโฮสต์เบสจะไม่มีโอกาสตรวจพบความพยายามที่จะโจมตีที่โดนปฏิเสธออกไปแล้วนี้เลย เนื่องจากมันไม่ได้ถูกกระทำจริงบนโฮสต์ แต่มันก็ถือว่าเป็นสิ่งจำเป็นที่ต้องรู้ถึงความถี่และชนิดของการโจมตีที่กระทำบนเครือข่ายของเรา

6. การทำให้สมบูรณ์ยิ่งขึ้นและการช่วยตรวจสอบ ระบบตรวจสอบผู้บุกรุกจะเป็นองค์ประกอบที่ทำให้ส่วนอื่นๆที่ใช้ในมาตรการรักษาความปลอดภัยอยู่แล้วสมบูรณ์ยิ่งขึ้น ตัวอย่างเช่นในการใช้การเข้ารหัสข้อมูล แม้ว่าระบบตรวจสอบผู้บุกรุกบนเครือข่ายจะไม่สามารถอ่านข้อมูลที่เข้ารหัสได้ แต่มันจะสามารถตรวจสอบได้ว่า ข้อมูลในเครือข่ายอันไหนที่ไม่ได้ถูกเข้ารหัสไว้ ส่วนในกรณีของไฟร์วอลล์ ระบบตรวจสอบผู้บุกรุกบนเครือข่ายจะช่วยในการตรวจสอบว่ามันได้ทำหน้าที่ในการป้องกันแพ็กเก็ตที่ควรจะปฏิเสธได้ถูกต้อง ครบถ้วนหรือยัง

7. การไม่ขึ้นอยู่กับระบบปฏิบัติการใดๆ ระบบตรวจสอบผู้บุกรุกบนเครือข่ายไม่ขึ้นอยู่กับระบบปฏิบัติการของโฮสต์ที่ต้องการตรวจสอบความผิดปกติ เหมือนกับวิธีของแบบโฮสต์เบสซึ่ง ข้อมูลในล็อกของระบบแบบโฮสต์เบสจะได้มาได้ต้องขึ้นกับการทำงานของระบบปฏิบัติงานที่ถูกต้อง

บทที่ 5
การออกแบบและสร้างระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์

5.1 ขอบเขตของระบบต้นแบบการตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ที่สร้างขึ้น

ระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ที่สร้างขึ้น มุ่งเน้นการศึกษา ออกแบบ และพัฒนาระบบต้นแบบของการตรวจจับผู้บุกรุกทางเครือข่าย ซึ่งสามารถจะตรวจสอบได้ทุกเครื่องที่มีเน็ตเวิร์กแอดเดรสเดียวกันและติดตั้งอยู่บนฮับ โดยจะสามารถตรวจจับการโจมตีเพื่อให้ปิดบริการสำหรับโพรโตคอลสแต็กทีซีพี/ไอพีเป็นหลัก ซึ่งเป็นหนึ่งในประเภทของการผู้บุกรุกตามที่ได้กล่าวมาแล้ว ซึ่งมีแนวโน้มเพิ่มขึ้นทุกวัน และ ตรวจจับการสแกนระบบ ซึ่งจะเป็นการเตือนว่าอาจจะมีการบุกรุกเข้ามาในเครือข่าย รวมถึงการตรวจจับผู้บุกรุกที่ใช้จุดอ่อนของชั้นแอพพลิเคชันที่ทำงานอยู่ในระบบเพื่อทำการโจมตีด้วย โดยเมื่อทำการตรวจสอบพบก็จะส่งผลการวิเคราะห์การโจมตีออกทางหน้าจอ หรือทางเว็บไซต์ หรือ เก็บไว้ในล็อกไฟล์ อีกทั้งยังเก็บข้อมูลเหล่านี้ไว้ “SysLog“ ซึ่งเป็นล็อกไฟล์ของระบบปฏิบัติการ

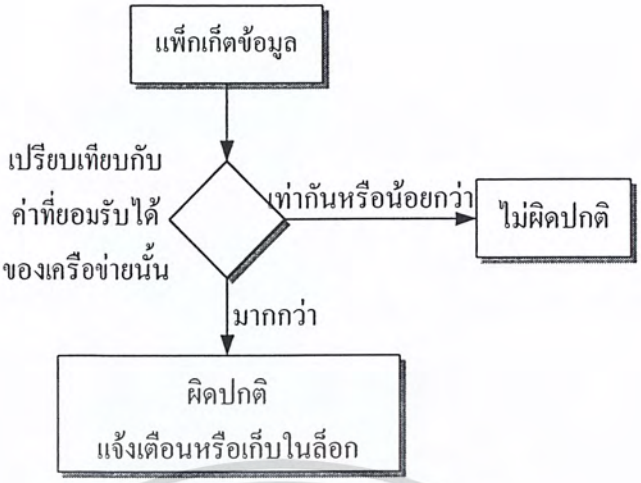
5.2 วิธีการตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์

ระบบตรวจจับผู้บุกรุกที่สร้างขึ้นนี้มีวิธีการตรวจจับผู้บุกรุก แบ่งออกตามชั้นโพรโตคอลสแต็กทีซีพี/ไอพี ได้ดังนี้

5.2.1 ประเภทอยู่ในชั้นเน็ตเวิร์ก

5.2.1.1 การส่งแพ็กเก็ตปริมาณมาก (Flooding)

การตรวจจับการโจมตีประเภทนี้ เราต้องวิเคราะห์แพ็กเก็ตที่เข้ามาในเน็ตเวิร์ก โดยดูที่แอดเดรสปลายทางเป็นหลัก คือเมื่อมีแพ็กเก็ตเข้ามาเราก็จะทำการนำค่าของแอดเดรสปลายทางมาดูว่ามีข้อมูลเก็บอยู่หรือยัง ถ้ายังเราก็จะสร้างข้อมูลของมันขึ้นมา ส่วนถ้าเกิดว่ามีข้อมูลอยู่แล้วเราก็จะทำการเพิ่มค่าของตัวนับ ซึ่งหมายถึงจำนวนจำนวนแพ็กเก็ตที่เข้ามาในเครื่อง ในการเพิ่มค่าตัวนับนั้นต้องทำการตรวจสอบค่าของช่วงเวลาระหว่างแพ็กเก็ตก่อนว่ามีค่าอยู่ในช่วงที่กำหนดหรือไม่ เพื่อความไม่ผิดพลาดในการวิเคราะห์ ถ้าเกิดว่าเกินช่วงเวลาเราก็จะทำการลบข้อมูลนั้นทิ้ง หลังจากที่ทำการเพิ่มค่าของตัวนับแล้ว ก็นำข้อมูลที่เก็บไว้เหล่านั้นมาเปรียบเทียบกับค่ามาตรฐานของเน็ตเวิร์กที่ระบบนี้ติดตั้งอยู่ หากค่าที่นับได้มากกว่าค่าที่ยอมรับได้ ก็ให้แจ้งเตือนแก่ผู้ดูแลระบบ ซึ่งการทำงานดังกล่าวมานี้ เป็นไปดังรูปที่ 5-1

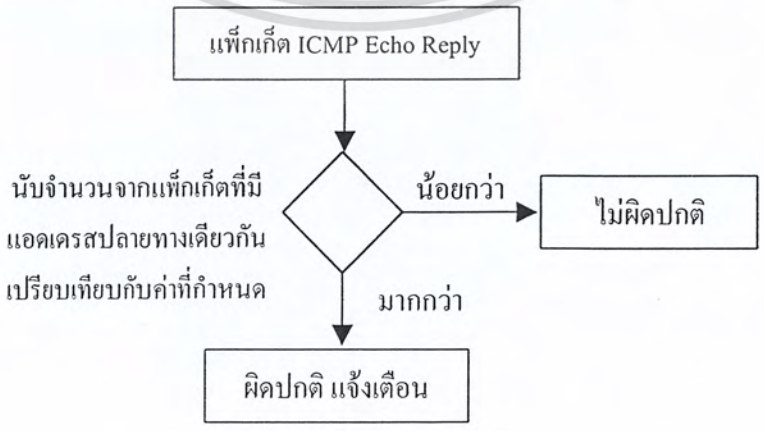


รูปที่ 5-1 แสดงการตรวจสอบการส่งแพ็กเก็ตปริมาณมาก

ความยากของการวิเคราะห์แบบนี้อยู่ที่การหาค่าที่ระบบยอมรับได้ เพราะขึ้นอยู่กับปัจจัยหลายประการ เช่น อัตราเร็วของเครือข่าย อัตราเร็วของหน่วยประมวลผลเครื่อง ปริมาณหน่วยความจำในเครื่อง เป็นต้น การหาค่าที่ระบบยอมรับได้นี้ สามารถทำได้โดยการเปิดการเชื่อมต่อกับระบบที่วิเคราะห์ จากนั้นหาจำนวนแพ็กเก็ตที่เข้ามาในระบบในลักษณะการใช้งานปกติของแต่ละช่วงเวลา จากนั้นนำค่าสูงสุดที่ได้มาเป็นค่าที่ระบบยอมรับได้ โดยระบบที่วิเคราะห์มีค่าที่ยอมรับได้ประมาณ 20,000 – 30,000 แพ็กเก็ตต่อนาที

5.2.1.2 การโจมตีแบบ สเมิร์ฟ (Smurf Attack)

การโจมตีแบบนี้จะใช้หลักการเดียวกับการส่งแพ็กเก็ตปริมาณมาก แต่จะต่างกันที่การ โจมตีแบบนี้จะใช้หลักการของ บรอดแคสต์แอดเดรส ทำการตรวจสอบการ โจมตีแบบนี้ทำได้โดยการตรวจสอบ แพ็กเก็ต ICMP Echo Reply ที่จะถูกส่งมายังเครื่องปลายทางเครื่องเดียวกัน และแอดเดรสของผู้ส่งอยู่ เน็ตเวิร์กแอดเดรสเดียวกัน หลักการที่เหลือจะเหมือนกับของการส่งแพ็กเก็ตปริมาณมาก



รูปที่ 5-2 แสดงการตรวจสอบการโจมตีแบบสเมิร์ฟ

5.2.1.3 ความผิดปกติของแฟร็กเมนต์

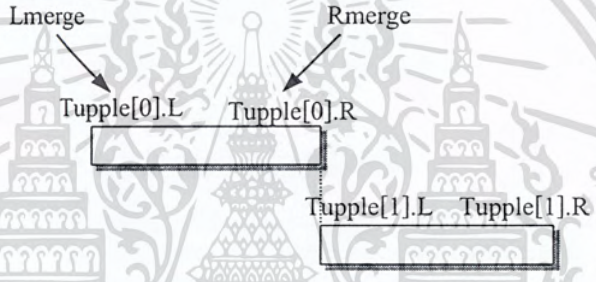
การตรวจสอบความผิดปกติของแฟร็กเมนต์มีขั้นตอนค่อนข้างซับซ้อน ซึ่งแยกอธิบายตามประเภทของความผิดปกติได้ ดังต่อไปนี้

(1) การส่งแพ็กเก็ตที่มีลำดับผิดปกติ และแพ็กเก็ตที่มีขนาดเหลื่อมล้ำกัน

การวิเคราะห์ความผิดปกติของแพ็กเก็ตในลักษณะนี้ ต้องวิเคราะห์หลังกระบวนการรีแอสเซมเบิ้ลไปแล้ว ดังนั้นจึงนำบัฟเฟอร์เข้ามาช่วยในการเก็บข้อมูล เพื่อนำมาวิเคราะห์ ดังนี้

- Fragment Buffer

คือ บัฟเฟอร์ที่เก็บข้อมูลในการวิเคราะห์ ซึ่งเก็บข้อมูลของแพ็กเก็ตไอพี และข้อมูลที่จำเป็นอื่นๆ ไว้ ได้แก่ หมายเลขแพ็กเก็ต (ID), จำนวนแพ็กเก็ตที่มีหมายเลขแพ็กเก็ตเดียวกัน, แอดเดรสปลายทาง, ขอบซ้ายและขอบขวาของแพ็กเก็ต (Lmerge และ Rmerge) ซึ่งชี้โดยตัวแปร tupple ดังรูปที่ 4-2 จำนวน tupple และแฟล็กแสดงค่าของแพ็กเก็ต



รูปที่ 5-3 แสดงการเก็บข้อมูลของตัวแปร tupple

การเก็บข้อมูลใน Fragment Buffer มีตัวแปรต่างๆ ที่จัดเก็บดังตารางที่ 4-1

ID	count	IP	flag	Lmerge	Rmerge	tupple	Tupple_count	merge

ตารางที่ 5-1 แสดงโครงสร้างการเก็บข้อมูลของ Fragment Buffer

- Overlap Buffer

คือ บัฟเฟอร์ที่เก็บข้อมูล เมื่อตรวจพบว่าเกิดการเหลื่อมล้ำของแพ็กเก็ต

- Gap Frame Buffer

คือ บัฟเฟอร์ที่เก็บข้อมูล เมื่อตรวจพบว่าเกิดการประกอบเฟรมไม่ได้ในลักษณะมีช่องว่างระหว่างแพ็กเก็ต

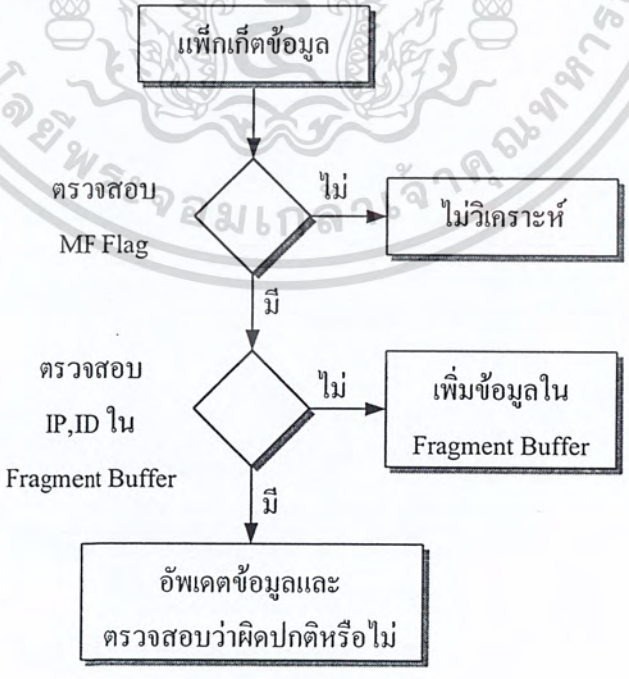
ข้อมูลที่เก็บไว้ของ Overlap Buffer และ Gap Frame Buffer มีโครงสร้างการเก็บข้อมูลที่มีส่วนประกอบเหมือนกัน ดังตารางที่ 4-2

ID	Count	IP	Start_min	Start_sec	End_min	End_sec	Etherhdr*

ตารางที่ 5-2 แสดงโครงสร้างการเก็บข้อมูลของ Overlap Buffer และ Gap Frame Buffer

ในการวิเคราะห์ใช้บัฟเฟอร์ทั้งสามนี้ร่วมกัน โดยเก็บข้อมูลแพ็กเก็ตที่เข้ามาทั้งหมดลงใน Fragment Buffer และหากแพ็กเก็ตที่ส่งมาสามารถรวมกันได้ก็รวมกันเป็นแพ็กเก็ตเดียวที่ต่อเนื่องกัน โดยดูจากขอบซ้ายและขอบขวา เช่น จากรูปที่ 4-2 หาก Tupple[0].R = Tupple[1].L แสดงว่าแพ็กเก็ตทั้งสองนี้สามารถเชื่อมต่อกันได้ ให้รวมเป็นแพ็กเก็ตเดียวกัน โดยแพ็กเก็ตใหม่มี Lmerge = Tupple[0].L และ Rmerge = Tupple[1].R

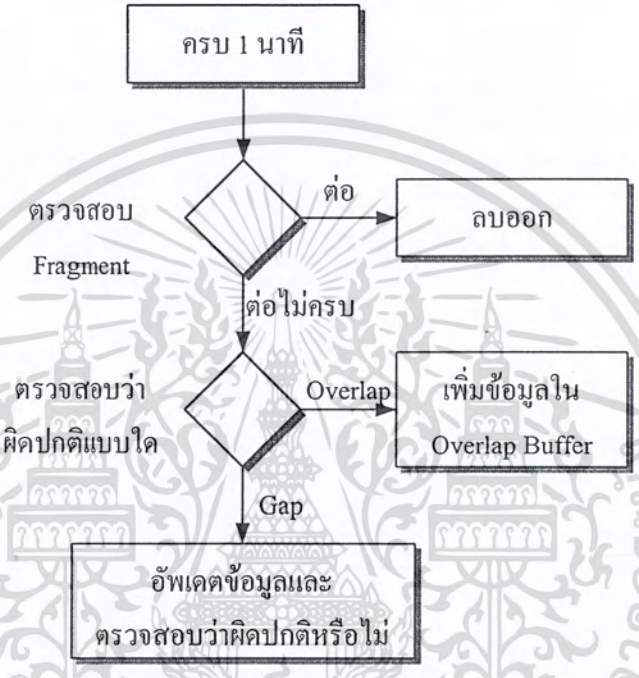
แต่หากรวมกันแล้วเกิดความผิดปกติ ให้แจ้งมายัง Overlap Buffer หรือ Gap Frame Buffer แล้วแต่ชนิดความผิดปกติที่เกิดขึ้น



รูปที่ 5-4 แสดงการเก็บข้อมูลลง Fragment Buffer

แต่หากไม่มีความผิดปกติขึ้น เมื่อครบ 1 นาที โปรแกรมตรวจสอบจาก Fragment Buffer ว่าหากมีแพ็กเก็ตใดยังไม่ได้ประกอบ หรือประกอบไม่ครบ ก็ให้เก็บไว้ใน Overlap Buffer หรือ Gap Frame Buffer เช่นเดียวกัน และเมื่อครบ 1 นาที ข้อมูลใน Overlap Buffer และ Gap Buffer นี้ จะออกมาที่หน้าจอเพื่อแจ้งให้ผู้ดูแลระบบทราบ หรือเก็บไว้ในล็อกไฟล์ เพื่อบันทึกความผิดปกติที่เกิดขึ้นไว้

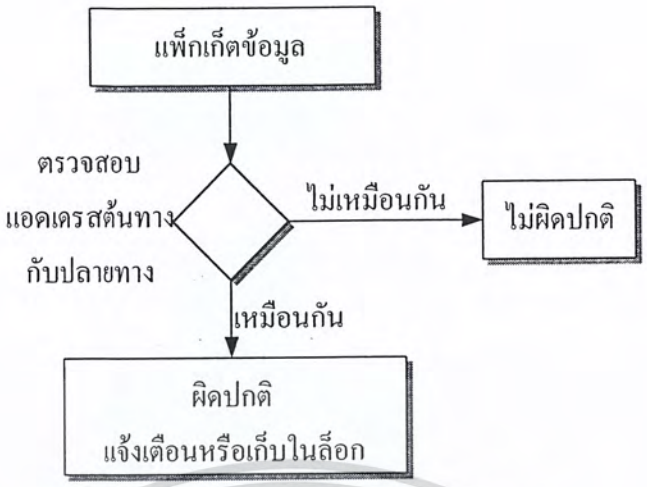
แต่หากไม่มีความผิดปกติใดๆ เกิดขึ้นเลย และแพ็กเก็ตเหล่านั้นสามารถประกอบเป็นเฟรมได้อย่างถูกต้อง ให้ลบเฟรมเหล่านั้นออกจากบัฟเฟอร์ทันที เพื่อไม่ให้สิ้นเปลืองเนื้อที่ในการจัดเก็บ



รูปที่ 5-5 แสดงการตรวจสอบความผิดปกติในการทำแฟร็กเมนเตชัน

5.2.1.4 การส่งแพ็กเก็ตแบบวนลูป (Land Attack)

สามารถทำได้โดยการเปรียบเทียบค่าแอดเดรสต้นทาง และแอดเดรสปลายทางของแพ็กเก็ต หากไอพี เป็นค่าเดียวกันแสดงว่ามีความผิดปกติเกิดขึ้น เพราะทำให้เกิดการส่งในลักษณะวนลูป ซึ่งขั้นตอนการตรวจสอบเป็นไปตามรูปที่ 5-6



รูปที่ 5-6 แสดงการตรวจสอบแพ็กเก็ตที่ส่งแบบวนลูป

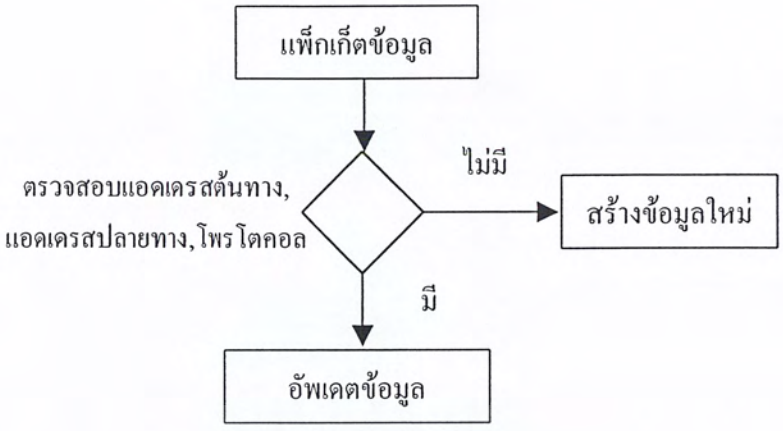
5.2.1.5 แบบผสม

การวิเคราะห์แพ็กเก็ตประเภทนี้ให้นำวิธีการวิเคราะห์ที่กล่าวข้างต้นมาใช้ร่วมกัน เนื่องจากเกิดจากวิธีการที่ผสมผสานกันระหว่างวิธีต่างๆ ที่ได้กล่าวมาแล้ว ซึ่งสามารถแยกวิเคราะห์ออกเป็นแต่ละแบบหรือวิเคราะห์รวมกันก็ได้

5.2.2 ประเภทอยู่ในชั้นทรานสปอร์ต หรือชั้นอินเทอร์เน็ต

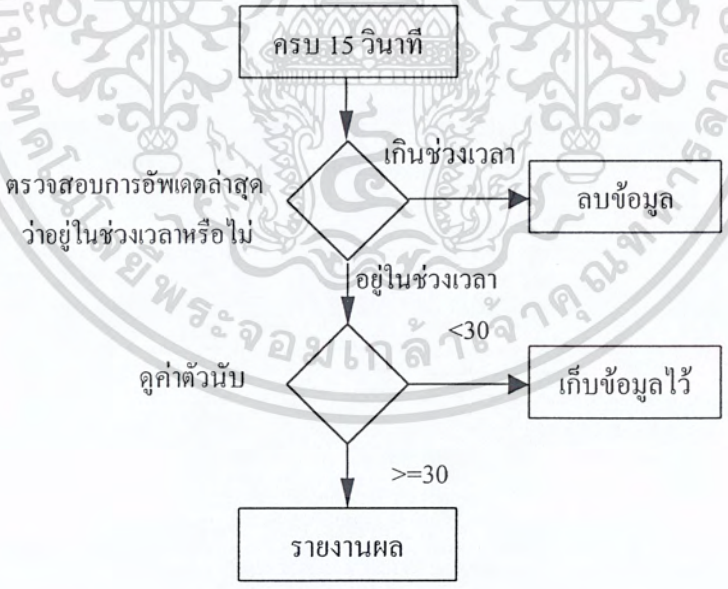
5.2.2.1 สแกนพอร์ต

การตรวจจับการบุกรุกแบบนี้ทำได้โดยการเก็บข้อมูลของแพ็กเก็ตที่เข้ามาในระบบ โดยเมื่อแพ็กเก็ตเข้าระบบก็จะตรวจสอบดูว่ามีข้อมูลของแพ็กเก็ตนี้อยู่ในฐานข้อมูลหรือไม่ถ้าไม่มีก็จะทำการเก็บเก็บไอพีแอดเดรส (IP address) ของเครื่องต้นทาง เครื่องปลายทาง พอร์ตปลายทางและโพรโตคอลที่ใช้ แต่ว่าถ้ามีข้อมูลอยู่แล้วก็จะทำการอัพเดตข้อมูล โดยทำการเพิ่มค่าของตัวนับและเริ่มนับเวลาใหม่ ในการนับเวลานั้น ถ้าเกิดว่าเวลาเกินจากค่าที่กำหนดก็จะทำการเริ่มค่าตัวนับใหม่ด้วย แล้วเมื่อเวลาผ่านไปช่วงเวลาหนึ่งก็จะมาตรวจสอบข้อมูลที่เราเก็บเอาไว้ว่า เครื่องไหนบ้างที่มีลักษณะเข้าข่ายของการถูกสแกนพอร์ต คือ ถ้าแพ็กเก็ตที่ส่งมายังเครื่องปลายทางส่งมาจากเครื่องต้นทางเดียวกัน แต่ว่ามีพอร์ตปลายทางที่ต่างกัน โดยอาจเป็นช่วงของพอร์ต เช่น 1-1024 อีกทั้งยังตรวจสอบอีกว่าช่วงเวลาระหว่างแต่ละแพ็กเก็ตนั้นมีค่าอยู่ในที่กำหนดไว้หรือไม่ โดยรูปแบบของแพ็กเก็ตเหล่านี้ต้องอยู่ในรูปแบบ ยูดีพี หรือทีซีพี ซึ่งถ้าเป็นแบบทีซีพี จะต้องมาตรวจสอบแฟล็กของแพ็กเก็ตเหล่านั้นว่าเป็นรูปใด เพื่อนำมาระบุรูปแบบของการโจมตีของการสแกนพอร์ต เช่น FIN, SYN, NULL , XMAS



รูปที่ 5-7 รูปภาพแสดงขั้นตอนการเก็บข้อมูลของการตรวจสอบสแกนพอร์ต

ทุกๆ 15 วินาที จะทำการตรวจสอบในแต่ละข้อมูลที่เก็บว่ามีการอัพเดตค่าภายในเวลาที่กำหนดหรือไม่ ถ้าไม่อยู่ในช่วงเวลาที่กำหนดก็จะทำการลบข้อมูลของแพ็กเก็ตนั้นออก เพื่อเป็นการไม่สิ้นเปลืองทรัพยากรเนื่องจากไม่มีความจำเป็นต่อการวิเคราะห์แล้ว แต่ถ้ามีการอัพเดตในช่วงเวลาก็จะทำการตรวจสอบต่อว่าค่าของตัวนับนับไปถึงค่าที่กำหนดหรือไม่ ถ้านับไม่ถึงก็จะไม่ทำอะไรต่อ แต่ถ้านับถึงค่าก็จะทำการสร้างตัวรายงานแล้วส่งต่อไปให้ส่วนของการออกรายงานเพื่อนำรายงานนั้นไปแสดงต่อไป

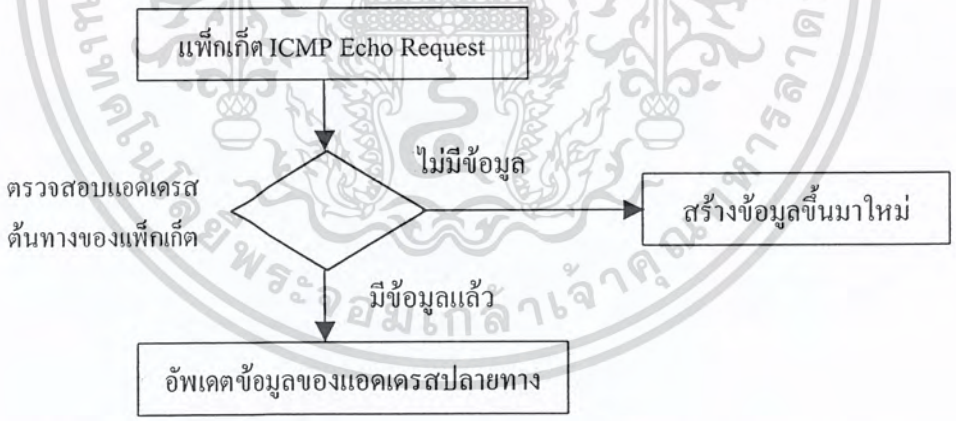


รูปที่ 5-8 รูปภาพแสดงขั้นตอนนำข้อมูลมาวิเคราะห์ว่าเกิดการสแกนพอร์ตหรือไม่

หมายเหตุ : ค่าของช่วงเวลาระหว่างแพ็กเก็ต และ ค่าของตัวนับ ซึ่งก็หมายถึงจำนวนพอร์ตที่สแกนนั้นได้มาจากการทดลองใช้โปรแกรมที่ใช้ในการสแกนแบบต่างๆ แล้วดูว่าลักษณะการทำงานเป็นอย่างไร แล้วนำผลที่ได้มาใช้ในตัวระบบตรวจจับผู้บุกรุก

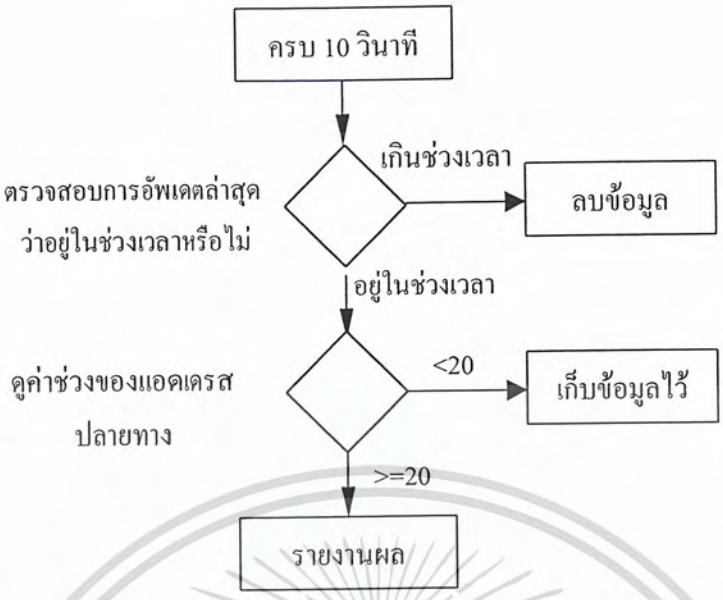
5.2.2.2 Ping Sweep

การตรวจจับการโจมตีแบบนี้ทำได้โดยการตรวจสอบแพ็กเก็ตที่เข้ามาในเครือข่ายว่ามีการส่ง ICMP Echo Request จากเครื่องต้นทางเครื่องเดียวกัน ไปยังเครื่องปลายทางหลายๆ หรือไม่ ซึ่งการที่มีการส่ง ICMP Echo Request มานี้เพื่อทดสอบดูว่าเครื่องปลายทางนั้นทำงานอยู่หรือไม่ เพราะถ้าเครื่องเป้าหมายทำงานอยู่มันจะตอบกลับไปด้วย ICMP Echo Reply ดังนั้นเมื่อมีแพ็กเก็ตลักษณะนี้เข้ามาเราจะทำการเก็บข้อมูลไว้ โดยจะเก็บข้อมูลของแอดเดรสต้นทาง และแอดเดรสปลายทางซึ่งจะมีลักษณะเป็นช่วงเช่น 161.246.5.1-161.246.5.222 ถ้าเกิดว่าข้อมูลมีอยู่แล้วก็จะทำการอัพเดตข้อมูลของแอดเดรสปลายทาง แต่ถ้ายังไม่มีข้อมูลก็จะทำการสร้างข้อมูลขึ้นมาใหม่ ในการตรวจสอบแต่ละครั้งจะต้องตรวจสอบช่วงเวลาระหว่างแพ็กเก็ตด้วยว่ามีค่ามากเกินที่กำหนดหรือไม่ เพราะถ้าเกินเราจะไม่สนใจข้อมูลนี้แล้ว เนื่องจากว่ารูปแบบของแพ็กเก็ตนี้อาจเกิดจากโปรแกรม Ping ซึ่งเป็นโปรแกรมที่ใช้กันอย่างกว้างขวางเพื่อตรวจสอบว่าโฮสต์นั้นยังทำงานอยู่หรือไม่ ดังนั้นถ้าเราไม่สนใจเรื่องของช่วงเวลาจะทำให้การวิเคราะห์ข้อมูลผิดพลาดได้



รูปที่ 5-9 รูปภาพแสดงขั้นตอนการเก็บข้อมูลของการตรวจสอบ Ping Sweep

ทุกๆ 10 วินาที จะทำการตรวจสอบในแต่ละข้อมูลที่เก็บว่ามีการอัพเดตค่าภายในเวลาที่กำหนดหรือไม่ ถ้าไม่อยู่ในช่วงเวลาที่กำหนดก็จะทำการลบข้อมูลของแพ็กเก็ตนั้นออก เพื่อเป็นการไม่สิ้นเปลืองทรัพยากรเนื่องจากไม่มีความจำเป็นต่อการวิเคราะห์แล้ว แต่ถ้ามีการอัพเดตในช่วงเวลาก็จะทำการ ตรวจสอบต่อว่า ช่วงของแอดเดรสปลายทางนั้นมีค่ามากถึงค่าที่กำหนดหรือไม่ ถ้าไม่ถึงก็จะไม่ทำอะไรต่อ แต่ถ้านับถึงค่าก็จะทำการสร้างตัวรายงานแล้วส่งต่อไปให้ส่วนของการออกรายงานเพื่อนำรายงานนั้นไปแสดงต่อไป



รูปที่ 5-10 รูปภาพแสดงขั้นตอนนำข้อมูลมาวิเคราะห์ว่าเกิด Ping Sweep หรือไม่

5.2.3 ประเภทอยู่ในชั้นแอพพลิเคชัน

การโจมตีประเภทนี้จะเป็นการโจมตีที่เฉพาะเจาะจงไปที่ตัวแอพพลิเคชัน คือจะอาศัยจุดอ่อนหรือ บั๊ก(Bug)ของแอพพลิเคชัน เพื่อให้เกิดการทำงานผิดพลาด โดยการส่งข้อมูลไปที่แอพพลิเคชันเหล่านั้นเพื่อให้แอพพลิเคชันจัดการข้อมูลผิดพลาด ดังนั้นการจะตรวจสอบการโจมตีแบบนี้จะต้องทำการดูที่ข้อมูลของแพ็กเก็ต ว่ามีแพตเทิร์นของการโจมตีหรือไม่ การที่เราจะตรวจสอบได้นั้นเราต้องมีข้อมูลของการโจมตีแต่ละชนิด โดยข้อมูลเหล่านี้สามารถหาได้จากเว็บไซต์ที่รายงานเกี่ยวกับเรื่องระบบรักษาความปลอดภัย เช่น CERT โดยเมื่อระบบเริ่มทำงานจะต้องทำการอ่านแพตเทิร์นที่เก็บไว้ในไฟล์ขึ้นมา ซึ่งรูปแบบที่เก็บไว้ในไฟล์เก็บเป็นรูปแบบของ rule base ซึ่งมีลักษณะดังนี้

name transport_protocol src_port dest_port payload_pattern

name คือ ชื่อของการโจมตี **transport_protocol** คือ ชนิดของแพ็กเก็ต ได้แก่ TCP UDP ICMP

src_port คือ พอร์ตต้นทาง **dest_port** คือ พอร์ตปลายทาง

payload_pattern คือ รูปแบบของซิกเนเจอร์ของการโจมตี โดยที่ ในลักษณะของ payload_pattern มีดังนี้

"..." : เป็นซิกเนเจอร์ที่ case sensitive '...' : เป็นซิกเนเจอรที่เป็น case insensitive

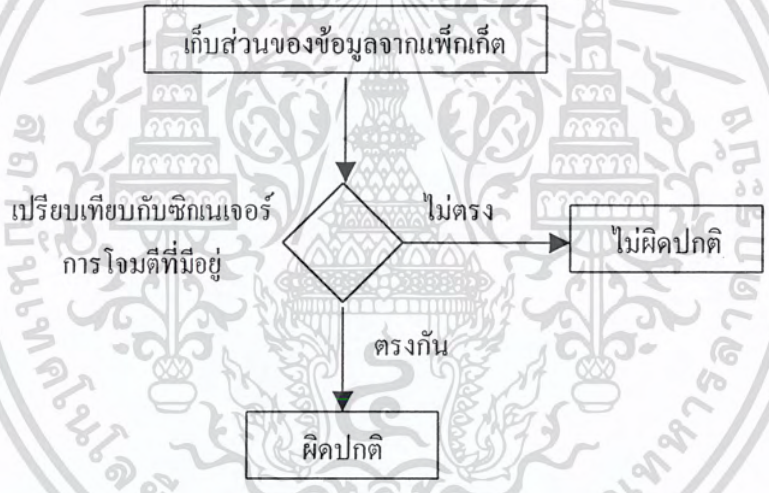
|...| ระหว่าง "" or " : เป็นซิกเนเจอร์ที่เป็น binary

หมายเหตุ : ในกรณีที่ transport_protocol เป็นแบบ ICMP ในส่วนของ src_port dest_port จะเป็นค่าของ type และ code ตามลำดับ

ตัวอย่างของซิกเนเจอร์

IIS-iisadmpwd	tcp * 80 'iisadmpwd/aexp3.htr'
FTP-exploit1	tcp * 21 "|50 57 44 0A 2F 69|"
DOS-TFN	icmp 0 0 "|73 68 65 6C 6C 20 62 6F 75 6E 64 20 74 6F 20 70 6F 72 74|"

โดยเอามาเก็บโดยใช้โครงสร้างของ Tree เมื่อมีแพ็กเก็ตเข้ามาก็จะเอาก็นำส่วนข้อมูลของแพ็กเก็ตนั้นมาเปรียบเทียบกับแพตเทิร์นที่เรามีอยู่ โดยในการเปรียบเทียบแพตเทิร์นนั้นจะใช้อัลกอริทึมโบเยอร์ แอนด์มัวร์ **(Boyer-Moore Algorithm)** ซึ่งเป็นอัลกอลิทึมที่ใช้ใช้ในการเปรียบเทียบสตริงซึ่งจะมีความเร็วมากกว่าการเปรียบเทียบแบบธรรมดา โดยใช้การเปรียบเทียบจากตัวสุดท้ายของแพตเทิร์นก่อนถ้าไม่ตรงก็จะข้ามไปเป็นจำนวนเท่ากับขนาดของแพตเทิร์น ซึ่งจะเห็นว่ามันสามารถลดจำนวนการเปรียบเทียบลงได้มาก

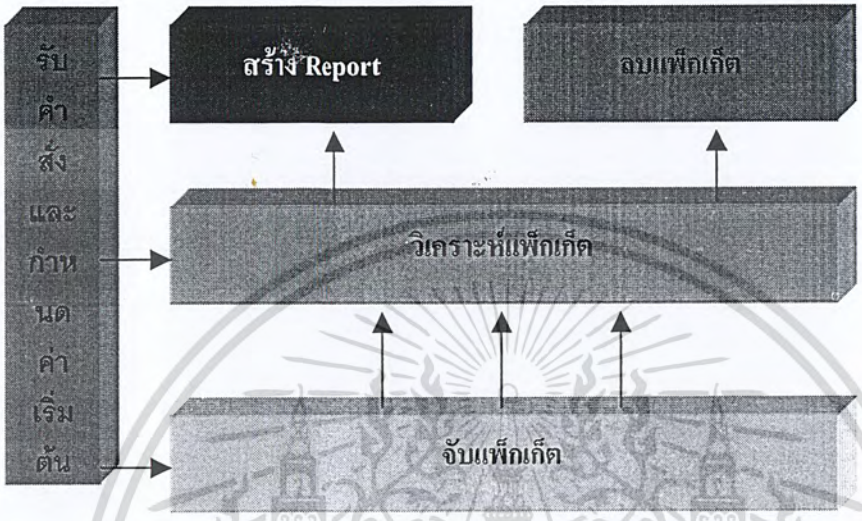


รูป 5-11 แสดงการตรวจสอบแพ็กเก็ตบนระดับชั้นแอพพลิเคชัน

หมายเหตุ : การวิเคราะห์แบบนี้นั้นจะมีผลทำให้ประสิทธิภาพของระบบลดลงเนื่องการต้องใช้เวลาในการเปรียบเทียบข้อมูลกับแพตเทิร์นที่เรามี ซึ่งถ้าเรามีแพตเทิร์นมากเกินก็จะต้องเสียเวลามากในการวิเคราะห์แต่ละแพ็กเก็ต ดังนั้นแพตเทิร์นต่างๆที่เราจะเอาเข้ามาใส่ก็ควรจะดูว่ามีความจำเป็นต่อระบบหรือไม่ เพื่อให้ระบบยังคงมีความมีประสิทธิภาพ

5.3 การทำงานของระบบตรวจจับผู้บุกรุก

ระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ พัฒนาขึ้นโดยใช้ ภาษา C++ สำหรับยูนิกซ์ ลักษณะการทำงานของระบบจะเป็นแบบ มัลติเธรดซึ่งก็คือแต่ละส่วนจะทำงานแยกจากกัน ไม่ขึ้นต่อกัน ทำให้การวิเคราะห์ข้อมูลทำได้เร็วขึ้น โดยจะมีการทำงานดังนี้



รูป 5-12 รูปแบบการทำงานของระบบ

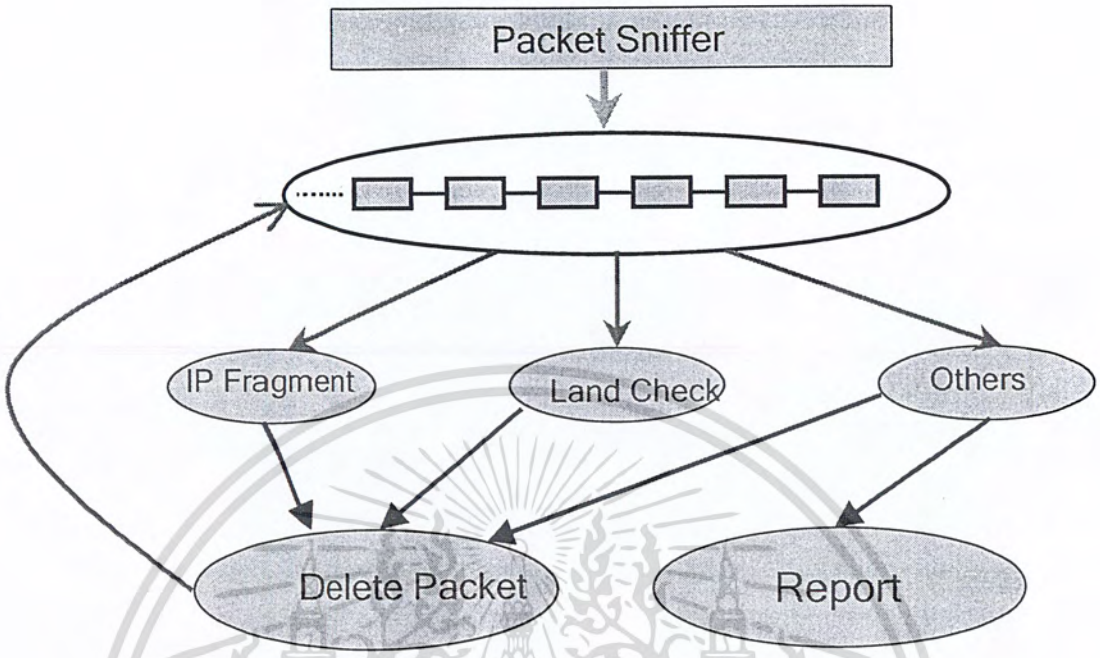
ส่วนที่ 1 : รับคำสั่งและกำหนดค่าเริ่มต้น รับค่าแฟล็กต่างๆจากผู้ใช้ เพื่อมากำหนดค่าเริ่มต้นให้กับโปรแกรม

ส่วนที่ 2 : ทำการจับแพ็กเก็ต จะจับแพ็กเก็ตแล้วนำแพ็กเก็ตไปแปลงให้อยู่ในรูปแบบโครงสร้างที่ออกแบบไว้แล้วนำไปเก็บไว้ใน ลิงค์ลิสต์

ส่วนที่ 3 : วิเคราะห์แพ็กเก็ต แบ่งเป็นเธรดหลายๆ เธรดตามชนิดของการโจมตี เช่น Ipdefragment, Land, TCP scan port โดยแต่ละเธรดจะหยิบแพ็กเก็ตจาก ลิงค์ลิสต์รวม เพื่อมาวิเคราะห์ ซึ่งส่วนนี้ทำงานแยกกันหมด

ส่วนที่ 4 : ลบแพ็กเก็ต เมื่อส่วนที่ 2 ทำการวิเคราะห์แพ็กเก็ต เรียบร้อยทุก thread แล้วส่งสัญญาณบอกว่าแพ็กเก็ตนี้วิเคราะห์เสร็จแล้ว เมื่อได้รับสัญญาณ ก็ลบแพ็กเก็ตนั้นออกจาก ลิงค์ลิสต์เพื่อไม่ให้เปลืองหน่วยความจำ

ส่วนที่ 5 : สร้างรายงาน เมื่อวิเคราะห์แพ็กเก็ตแล้วตรวจพบการโจมตี จะมีสัญญาณมาบอกให้ส่วนนี้ทำการสร้างรายงานออกมา โดยอาจจะ แสดงผลออกทางหน้าจอ หรือ เก็บลงล็อกไฟล์ หรือ สร้างเป็นเว็บเพจ หรือเก็บลงล็อกไฟล์ของระบบ



รูปที่ 5-13 แสดงการทำงานของระบบ

5.3.1 ส่วนการกำหนดค่าเริ่มต้นให้ระบบ

ในส่วนนี้จะเป็นส่วนการจัดการกำหนดค่าของตัวแปรต่างๆที่จะเป็นต่อระบบ การกำหนดค่าเริ่มต้นนี้จะมี 2 ส่วนคือ

1. ใส่เป็น Argument เวลาเริ่มทำงานโปรแกรม
2. อ่านค่าจาก Configuration File ซึ่งก็คือไฟล์ IsagNids.conf

รายละเอียดของส่วนที่เก็บข้อมูล Configuration คือ

```
typedef struct _config
    {
        char *pcap_filter;  // filter สำหรับกำหนดแพ็กเก็ตที่เข้ามาในระบบ
        int report_screen; // ตัวเลือกให้แสดงค่าออกหน้าจอ
        int report_html;  // ตัวเลือกให้แสดงค่าออกเว็บเพจ
        char  *www_directory; // ที่อยู่ที่จะเก็บเว็บเพจ
        int report_syslog;  // ตัวเลือกให้แสดงค่าใน SysLog
        int report_logfile;  // ตัวเลือกมให้แสดงค่าลง Logไฟล์
        char *log_dir; // ที่เก็บของ Log ไฟล์
```

```
        char *log_file; // ชื่อของล็อกไฟล์ที่เราจะเก็บข้อมูล
        char *signature_file;  // ชื่อไฟล์ที่เก็บข้อมูลของแพตเทิร์น
        int scan_ttl; // กำหนดช่วงเวลาระหว่างแพ็กเก็ต
        int scan_max_count;  // ค่าสูงสุดที่ยอมรับได้ของการสแกน
        int flood_ttl; // กำหนดช่วงเวลาระหว่างแพ็กเก็ต
        int flood_max_count; // ค่าสูงสุดที่ยอมรับได้ของการ Flood
        int verbose; // ตั้งค่าให้แสดงค่าขั้นตอนต่างๆ ออกทางหน้าจอ
        int background; // ตั้งให้โปรแกรมทำงานเป็น background  mode
        char *device;  ชื่อของ เน็ตเวิร์กการ์ด
    } config;
```

เมื่ออ่านข้อมูลมาแล้วก็จะเอาไปเก็บไว้ในสร้างนี้ เพื่อให้ส่วนอื่นๆมานำไปใช้เมื่อต้องการ การทำงานในส่วนนี้ถูกจัดการโดยคลาส Manager ซึ่งมีรายละเอียดดังนี้

```
class manage
{
    private:
        config con;                    // ใช้เพื่อเก็บข้อมูลของ Configuration
    public:
        void check_config();          // อ่านข้อมูลจากไฟล์  Configuration
        void usage(char *argv0);   // แสดงส่วนช่วยเหลือการใช้งาน
         void  check_command(int argc,char **argv);
         config *get_config() { return &con;}  // ส่งค่าของ Configuration
};
```

- Void check_config()

  จุดมุ่งหมาย : เป็นเมธอดที่ใช้ในการตรวจสอบ ไฟล์ Configuration แล้วนำค่าไปเก็บไว้ในโครงสร้างของ Configuration

  ขั้นตอนการทำงาน :

  1. อ่านข้อมูล จากไฟล์ IsagNids.conf
  2. กำหนดค่าให้กับ โครงสร้างของ Configuration

- Void check_command(int argc,char **argv)

  จุดมุ่งหมาย : เป็นฟังก์ชันในการอ่านข้อมูลของ Argument ที่รับเข้ามาจากตอนเริ่มต้นการทำงาน

  ขั้นตอนการทำงาน :

1. อ่านข้อมูลจาก Argv
2. กำหนดค่าให้กับ โครงสร้างของ Configuration

5.3.2 ส่วนการออกรายงาน

ในส่วนนี้จะรับหน้าที่ในการสร้างรายงานทั้งหมดของระบบ โดยในระบบนี้สามารถสร้างรายงานผลการวิเคราะห์ออกได้ 4 ทาง คือ

1. สร้างรายงานออกทางหน้าจอ
2. สร้างรายงานออกทางเว็บเพจในรูปแบบของ CGI
3. สร้างรายงานเก็บไว้ในล็อกไฟล์ที่เรากำหนด
4. สร้างรายงานเก็บไว้ใน SysLog ซึ่งเป็นล็อกไฟล์ของระบบปฏิบัติการ

สำหรับการเลือกว่าต้องการออกรายงานแบบใดนั้นจะสามารถกำหนดได้ในไฟล์ Configuration

เมื่อเริ่มทำงานส่วนนี้จะเป็นเธรดหนึ่งซึ่งจะยังไม่มีการทำงานใดๆทั้งสิ้น โดยจะหยุดรอจนกระทั่งมีส่วนการวิเคราะห์ส่งรายงานมาให้ หลังจากนั้นส่วนนี้จะทำหน้าที่เอารายงานนั้นมาแสดงผลตามที่ได้กำหนดไว้ สำหรับในกรณีที่มีการส่งรายงานมาให้ส่วนนี้หลายๆ อันพร้อมกันนั้น จะไม่เกิดปัญหาเพราะเมื่อมีรายงานเข้ามาจะถูกเก็บเป็นลิสต์ของข้อมูล แล้วค่อยๆ เอามาจัดการต่อไป ซึ่งส่วนนี้มีการจัดการในเรื่องของการ Synchonization ด้วย โดยใช้หลักการของ เซมาฟอร์(Semaphore) การทำงานทั้งหมดนี้จะเกี่ยวข้องกับ 3 คลาสด้วยกันคือ คลาส Report คลาส Html และ คลาส Syslog ซึ่งมีรายละเอียดดังนี้

โครงสร้างของรายงาน

```
struct report_node {
        struct in_addr srcip;    // แอดเดรสของผู้ส่ง
        struct in_addr dstip;    // แอดเดรสของผู้รับ
        u_int srcport;   // พอร์ตต้นทาง
        u_int dstport;   // พอร์ตปลายทาง
        char name[50];  // ชื่อของคลาสที่วิเคราะห์
        char description[150]; // รายละเอียดของคลาสที่วิเคราะห์
        char type[50]; // รูปแบบการโจมตี
        char detail[200]; // รายละเอียดของการโจมตี
        int Shour;   // เก็บเวลาที่เริ่มรับข้อมูล
        int Smin;   // เก็บเวลาที่เริ่มรับข้อมูล
        int Ssec;  // เก็บเวลาที่เริ่มรับข้อมูล
        int Ehour; // เก็บเวลาที่เลิกรับข้อมูล
        int Emin; // เก็บเวลาที่เลิกรับข้อมูล
        int Esec; // เก็บเวลาที่เลิกรับข้อมูล
```

```
        int count; // จำนวนครั้งการโจมตี
        int report_type; // ชนิดของรายงาน ซึ่งมา 5 รูปแบบ
};
```

class report

```
{
    private:
        report_node *current_report_queue;  // Pointer ชี้ตำแหน่งของReportปัจจุบัน
        report_node *first_report_queue;  // Pointer ชี้ตำแหน่งของReport แรก
        sem_t sem_report; // เซมาฟอร์ จัดการเรื่อง Synchronization
        sem_t sem_lockx;  // เซมาฟอร์ จัดการเรื่อง Synchronization
        int count; // จำนวนรายงานในคิว
        slog gen_syslog;
        htmlreport *gen_html;
        config *con;
        int report_screen;  // Flag กำหนดให้ออกรายงานหรือไม่
        int report_syslog;  // Flag กำหนดให้ออกรายงานหรือไม่
        int report_logfile;  // Flag กำหนดให้ออกรายงานหรือไม่
        int report_html;   // Flag กำหนดให้ออกรายงานหรือไม่
    public:
        void gen_report_to_screen();  // สร้างรายงานออกหน้าจอ
        void gen_report_to_file();  //สร้างรายงานออกไฟล์
        void gen_report_to_syslog(); // สร้างรายงานลงใน SysLog
        void gen_report_to_html(report_node *new_report); // สร้างรายงานในเว็บ
        int add_report(report_node *new_report);
        int check_report_in_queue();
        int gen_report_in_queue();
};
```

- Void add_report(report_node *new_report)

จุดมุ่งหมาย : เป็นเมธอดที่ทำหน้าที่นำรายงานที่ส่วนการวิเคราะห์ส่งมา มาเก็บลงในคิว ซึ่งเมธอดนี้จะถูกเรียกใช้แบบ Callback จากส่วนการวิเคราะห์ต่างๆ

ขั้นตอนการทำงาน :

1. รับรายงานจากส่วนการวิเคราะห์

2. นำรายงานนั้นไปเก็บลงในคิว

- Void gen_report_in_queue()

  จุดมุ่งหมาย : เป็นเมธอดที่ทำหน้าที่นำรายงานที่อยู่ในคิวมาแสดงผล

  ขั้นตอนการทำงาน :

  1. รอจนกระทั่งมีข้อมูลในคิว
  2. อ่านรายงานจากคิว
  3. ตรวจสอบFlag ว่าให้แสดงออกที่ใดบ้าง
  4. แสดงรายงาน
  5. กลับไปทำข้อ 1

```
class htmlreport {
        private:
                char htmldir[PATH_MAX] ;  // ที่อยู่รวมของเว็บ
                char htmldoc[PATH_MAX]; // ที่อยู่ของเว็บ
                char htmldocdir[PATH_MAX]; //ที่อยู่ของเว็บแบ่งตามหน้า
                char latest[PATH_MAX];  // Link ของเว็บหน้าล่าสุด
                long pages;    // จำนวนหน้าปัจจุบัน
                config *con;  // เก็บข้อมูล Configuration
        public:
                htmlreport(config *conf);  // Constructor
                void output_plugin_infos(FILE *fd,report_node *newreport);
                void output_report_infos(FILE *fd, report_node *newreport);
                void output_host_infos(FILE *fd,report_node *newreport);
                void create_detailled_report(report_node *newreport);
                FILE *setup_htmldoc(void);
                void update_host_index(const char *link,report_node *newreport);
};
```

- Void create_detailled_report(report_node *newreport);

  จุดมุ่งหมาย : เป็นเมธอดที่ทำหน้าที่สร้างรายละเอียดของรายงานการวิเคราะห์ลงในเว็บเพจ โดยตัวมันจะไปเรียกเมธอด output_plugin_infos , output_report_infos และ output_host_infos เพื่อทำการสร้างรายงานในแต่ละส่วน แล้วก็จะเรียกเมธอด update_host_index เพื่อสร้าง Link จากหน้าหลัก

ขั้นตอนการทำงาน :

1. สร้างไฟล์สำหรับเว็บเพจ
2. เขียนรายงานในส่วนของคลาสที่วิเคราะห์
3. เขียนรายงานในส่วนของรายละเอียดการโจมตี
4. เขียนรายงานในส่วนของโฮสต์ที่โจมตีและถูกโจมตี
5. สร้าง Link ของเพจ จากหน้าหลัก

- FILE *setup_htmldoc(void);

จุดมุ่งหมาย : เป็นเมธอดที่ทำหน้าที่สร้างข้อมูลหลักของเว็บเพจทั้งหมด ซึ่งก็คือสร้างไดเรกทอรี (Directory) สร้าง Link รวมทั้งสร้างไฟล์เริ่มต้น

ขั้นตอนการทำงาน :

1. ตรวจสอบว่า ไดเรกทอรี มีอยู่หรือไม่ ถ้าไม่มีก็สร้าง
2. ในกรณีที่ไม่มีเลยก็จะสร้างหน้าเพจเริ่มต้น

```
class slog {
        private:
            char buff[256];  // ข้อมูลที่ต้องการจะเขียนลงในล็อกไฟล์
        public:
            slog(); //  Constructor
            int slog_init(void);  // กำหนดค่าเริ่มต้น
            void slog_close(void);  // ปิด SysLog ของระบบ
            void slog_dump (char *std);  // เขียนข้อมูลลงใน SysLog
    };
```

5.3.3 ส่วนของการลบข้อมูล แพ็กเก็ต

ในส่วนนี้จะรับหน้าที่ในการลบข้อมูลของแพ็กเก็ตออกจากระบบหลังจากที่แพ็กเก็ตเหล่านั้นถูกวิเคราะห์เสร็จเรียบร้อยแล้ว ซึ่งส่วนนี้ก็จะทำงานเป็นเธรดหนึ่ง โดยเมื่อเริ่มทำงานส่วนนี้จะรอสัญญานให้ลบข้อมูลจากส่วนของการวิเคราห์ข้อมูล ซึ่งใช้หลักการของ เซมาฟอร์ คือ เมื่อเริ่มเธรดจะถูก Block โดย เซมาฟอร์ แล้วเมื่อส่วนของการวิเคราะห์ตรวจสอบข้อมูลเสร็จก็จะส่งสัญญานให้เพิ่มค่าของ เซมาฟอร์ เมื่อทุกการวิเคราะห์ส่งมาครบก็จะ ผ่านเข้ามาลบข้อมูลแพ็กเก็ตนั้นได้ ซึ่งจากหลักการนี้ทำให้ไม่เกิดปัญหาในเรื่องของการที่ข้อมูลถูกลบออกก่อนที่จะวิเคราะห์เสร็จ

ขั้นตอนการทำงาน :

1. เข้าสู่ลูปไม่จำกัด
2. ถูก Block โดย เซมาฟอร์

3. ทำการลบข้อมูลของแพ็กเก็ตอันบนสุด
4. วนกลับไปทำข้อ 2

หมายเหตุ : ในการลบแพ็กเก็ตข้อมูลนั้นเราจะทำการลบข้อมูลของอันที่อยู่แรกสุดของ ลิงค์ลิสต์ แล้วค่อยๆลบถัดเข้าไปเรื่อยๆ

5.3.4 ส่วนของการดักจับแพ็กเก็ตข้อมูล

ในส่วนนี้จะรับหน้าที่ในการดักจับข้อมูลจากเน็ตเวิร์กการ์ดแล้วนำมาจัดให้อยู่ในรูปแบบที่ง่ายต่อการวิเคราะห์ เพื่อที่ส่วนวิเคราะห์จะได้นำไปตรวจสอบต่อไป ก่อนการดักจับแพ็กเก็ตข้อมูลนั้นเราต้องตั้งค่าของเน็ตเวิร์กการ์ดให้รับแพ็กเก็ตข้อมูลจากทุกแอดเดรส โดยเราจะเรียกโหมดการทำงานแบบนี้ว่า โพรมิสคิวอัส (Promiscuous Mode) สำหรับในส่วนที่ดึงข้อมูลขึ้นมาจากเน็ตเวิร์กการ์ดนั้น จะใช้ไลบรารี Pcap ซึ่งจะจัดการในการจับแพ็กเก็ตข้อมูล รวมทั้งกรองแพ็กเก็ตต่างๆ ที่ไม่จำเป็นต่อการวิเคราะห์ออกไป อีกทั้งส่วนนี้ยังเป็นส่วนการทำงานหลักที่เป็นผู้เริ่มต้นการทำงานของเธรดอื่นๆ

ขั้นตอนการทำงาน :

1. หาเน็ตเวิร์กการ์ดของระบบ
2. ตั้งค่าของเน็กเวิร์กการ์ดให้อยู่ในโหมด โพรมิสคิวอัส
3. ตั้งค่าของ Filter ที่จะใช้ในการกรองแพ็กเก็ต
4. ตรวจสอบค่าของการวิเคราะห์แบบต่างๆ
5. เริ่มการทำงานของเธรดต่างๆ
6. เริ่มการดึงข้อมูลจากเน็ตเวิร์กการ์ด
7. จัดรูปแบบของแพ็กเก็ต แล้วนำไปเก็บไว้ใน ลิงค์ลิสต์
8. แสดงข้อมูลของแพ็กเก็ตที่รับเข้า ถ้าถูกตั้งค่าให้แสดง
9. กลับไปรับข้อมูลแพ็กเก็ตต่อไป

หมายเหตุ : ในการดักจับแพ็กเก็ตนั้นเราสามารถจัดการได้เองโดยการใช้ การเปิด Socket แล้วดึงแพ็กเก็ตข้อมูลขึ้นมาเองก็ได้ แต่เหตุผลที่ใช้ ไลบรารี Pcapเนื่องจากต้องการให้ระบบมีประสิทธิภาพมากขึ้น และสามารถรองรับการทำงานในระบบที่มีการสื่อสารข้อมูลมากๆ เพราะไลบรารีตัวนี้ จะทำการจับและกรองแพ็กเก็ตข้อมูลที่ระดับล่างเลย ซึ่งต่างจากของการเปิด Socket ที่เราต้องอ่านข้อมูลขึ้นมาก่อนแล้วค่อยเลือก ไลบรารีตัวนี้เป็นของ Berkeley University ซึ่งแพร่หลายและเป็นที่นิยมอย่างมากในเรื่องของการทำแพ็กเก็ต สนิฟเฟอร์

สำหรับคลาสที่จัดการส่วนนี้คือ คลาส Sniff ซึ่งมันก็จะไปเรียกคลาส Node มาใช้อีกทีเพื่อสร้างตัวข้อมูล ซึ่งมีรายละเอียดดังนี้

```
Class Sniff {
        Private:
                config *con;  // เก็บค่า Configuration
                node  *rootnode; // ตัวชี้ตำแหน่งแพ็กเก็ตล่าสุด
```

```
        node *firstnode; // ตัวชี้แพ็กเก็ตแรกสุด
        Land *land_check;  // อินสแตนท์ของคลาส Land
        IPdefragment *fragment_check;  // อินสแตนท์ของคลาส IPdefragment
        flood *flood_check;  // อินสแตนท์ของคลาส  flood
          scan *scan_check;  // อินสแตนท์ของคลาส  scan
        signature *signature_check;  // อินสแตนท์ของคลาส signature
        manage *manager;  // อินสแตนท์ของคลาส manage
        report *report_thread;  // อินสแตนท์ของคลาส report
    Public:
        int  lookup_lan_card(); // หาเน็ตเวิร์กการ์ดจากระบบ
        int ispromisc_mode(int);
      static  void  ether_filter(u_char*user,const  struct  pcap_pkthdr  *h,  const
                                                          u_char *p);
        void ip_filter(const u_char *p, int length);
      static  void* signature_thread(void* arg); // เธรดควบคุมการวิเคราะห์
      static  void* delete_packet(void* arg); // เธรดควบคุมการวิเคราะห์
      static  void* Land_thread(void* arg);  // เธรดควบคุมการวิเคราะห์
      static  void* IPdefragmented_thread(void* arg);  // เธรดควบคุมการวิเคราะห์
      static  void* Bomb_thread(void* arg);  // เธรดควบคุมการวิเคราะห์
      static  void* scan_port_thread(void* arg);  // เธรดควบคุมการวิเคราะห์
      static  void* Report_thread(void* arg);  // เธรดควบคุมการวิเคราะห์
          int      sniffer();  // ส่วนหลักในการจับข้อมูล
        void    exit_prog(int signo);
};
```

- Static Void* xxxxxx_thread(void* arg)

จุดมุ่งหมาย : เป็นฟังก์ชันในการสร้างthread ของการวิเคราะห์รูปแบบต่างๆ

ขั้นตอนการทำงาน :

1. กำหนดค่าต่างๆ
2. เข้าสู่อินฟินิตลูป
3. สร้าง ออบเจ็กต์ของคลาส ของการวิเคราะห์นั้นๆ
4. รอสัญญาณของเซมาฟอร์จากเธรดอื่นๆ
5. เรียกฟังก์ชันเพื่อตรวจสอบแพ็กเก็ต

หมายเหตุ xxxxxx : เป็นชื่อของการวิเคราะห์แต่ละแบบ เช่น Land, Ipdefragment

- Void ip_filter(const u_char *p,int length)

จุดมุ่งหมาย : เป็นฟังก์ชันในการรับ ไอพีแพ็กเก็ตและแปลงให้อยู่ในรูปแบบที่กำหนดแล้ว นำไปใส่ลงลิงค์ลิสต์

ขั้นตอนการทำงาน :

1. ปิดการรับสัญญาณทุกชนิดเพื่อกันการขัดจังหวะ
2. จักเก็บแพ็กเก็ตลงไปรูปแบบใหม่และนำไปใส่ลงลิงค์ลิสต์
3. ส่งสัญญาณเซมาฟอร์เพื่อบอกเธรดอื่นว่า มีแพ็กเก็ตมาแล้ว

- Void ether_filter(u_char*user, const struct pcap_pkthdr *h, const u_char *p)

จุดมุ่งหมาย : เป็นฟังก์ชันในการรับ อีเทอร์เน็ตแพ็กเก็ตแล้วเลือกเฉพาะ ไอพีแพ็กเก็ต

ขั้นตอนการทำงาน :

1. รับ อีเทอร์เน็ตแพ็กเก็ต
2. เลือกเฉพาะ ไอพีแพ็กเก็ต
3. เรียกฟังก์ชัน ip_filter

- int ispromisc_mode (int flag)

จุดมุ่งหมาย : เป็นฟังก์ชันในการเปลี่ยนโหมดของการ์ดแลน

ขั้นตอนการทำงาน :

1. ตรวจสอบค่าแฟล็กที่รับเข้ามา
2. เปลี่ยนโหมดของการ์ดแลน โดยถ้าค่าที่รับมามีค่าเป็น
   1 : ตั้งค่าการ์ดเป็น โพรมิสคิวอัส
   2 : ตั้งค่ากลับเหมือนเดิม

- void exit_prog(int signo)

จุดมุ่งหมาย : มีจุดมุ่งหมาย 3 ประการ คือ

1. เปลี่ยนเน็ตเวิร์กการ์ดให้เป็นโหมดปกติ
2. คืนหน่วยความจำให้ระบบ
3. แสดงสถิติต่างๆ

ขั้นตอนการทำงาน :

1. คืนหน่วยความจำ
2. เรียกฟังก์ชัน ispromisc_mode(2) เพื่อเปลี่ยนโหมดของการ์ดแลน ให้กลับสู่ปกติ
3. แสดงจำนวนแพ็กเก็ตที่รับได้และที่รับไม่ได้

4. สั่ง exit(0) ซึ่งเป็นการออกจากโปรแกรมแบบปกติ (Normal Exit)

- void Sniffer ()

จุดมุ่งหมาย : เก็บข้อมูลในเครือข่าย

ขั้นตอนการทำงาน :

1. อ่านค่าจาก Configuration
2. เรียกฟังก์ชัน lookup_lan_card() เพื่อหาการ์ดแลน(เน็ตเวิร์กการ์ด)
3. สร้างเธรดต่างๆที่จำเป็น
4. จัดการดักจับ Signal เพื่อใช้ออกจากโปรแกรม
5. เรียกใช้ไลบรารีของ pcap
6. เริ่มต้นจับข้อมูล

```
class node {
        public:
            node *next,*last;  //pointer to next node
            u_int8_t  eth_dhost[ETH_ALEN],eth_shost[ETH_ALEN];
            struct in_addr srcip,dstip;     // src and dst ip address
            u_short ip_len;    // บอกขนาดของ ไอพีแพ็กเก็ต
            u_short ip_id;      //  identification
            u_short ip_off;     // หมายเลย Offset
            u_int8_t ip_p;      // Protocal
            u_int16_t eth_type; // ชนิดของอีเทอร์เน็ต แพ็กเก็ต
            int hour; // เวลาที่รับแพ็กเก็ต
            int min;  // เวลาที่รับแพ็กเก็ต
            int sec;  // เวลาที่รับแพ็กเก็ต
            struct ip *this_iphdr;  // Header ของ IP
            struct tcphdr *this_tcphdr; // Header ของ TCP
            struct udphdr *this_udphdr;  // Header ของ UDP
            struct icmphdr *this_icmphdr;  // Header ของ ICMP
            u_int srcport,dstport;  // src and dst port  // หมายเลขพอร์ต
            u_char *data;  // ข้อมูลในส่วนของ Payload
            void print_frame();
        };
```

- void node::PrintFrame ()

  จุดมุ่งหมาย : แสดงข้อมูลของแพ็กเก็ตที่เก็บได้ให้อยู่ในรูป

  [Time] [Source HW Address] → [Destination Address]

  [PROTO] [Protocol Description]

  [Source IP Address] → [Destination IP Address] [Protocol]

  [Packet ID] [Fragment Offset] [Length][DATA in Hex]

  และนำค่าต่างๆ เหล่านี้มาแสดง

5.3.5 การวิเคราะห์ข้อมูล (Analyze Data)

ในส่วนนี้โปรแกรมจะนำข้อมูลของแพ็กเก็ตที่เก็บไว้มาวิเคราะห์ว่าเกิดความผิดปกติหรือไม่ และเกิดความผิดปกติในรูปแบบใด โดยแบ่งเป็นคลาสที่ใช้ในการวิเคราะห์ออกเป็น 5 คลาส เพื่อใช้ในการวิเคราะห์ความผิดปกติที่แตกต่างกัน 4 ประเภท

1. คลาส Land เป็นคลาสสำหรับตรวจสอบการโจมตีแบบมีการส่งแบบวนลูป
2. คลาส IPdefragmentment เป็นคลาสสำหรับวิเคราะห์แพ็กเก็ตที่มีความผิดปกติของแฟร็กเมนต์
3. คลาส flood เป็นคลาสสำหรับตรวจสอบการส่งแพ็กเก็ตมาโจมตีปริมาณมาก
4. คลาส Scan เป็นคลาสมสำหรับตรวจสอบการสำรวจระบบประเภทต่างๆ
5. คลาส Signature เป็นคลาสสำหรับตรวจแพตเทิร์นของข้อมูลที่เข้ามา

ซึ่งแต่ละส่วนนี้จะทำงานแยกจากกันโดยสิ้นเชิงเนื่องจากทำงานในรูปแบบของเธรด นอกจากนี้แล้วยังมีคลาสที่ทำหน้าที่จัดการเรื่องการเก็บข้อมูลของโฮสต์ต่างๆ อีก ซึ่งก็คือ คลาส Hostdb โดยแต่ละคลาสมีรายละเอียดดังนี้

```
class Land {
        private:
                int Lmaxindex;
                int Lcount;
                struct LandData land[STD_BUF]; // ที่เก็บข้อมูล
                report_node report_land;  // รายงานการโจมตี
                config *con;  // Configuration
        public:
                void LAND(node *data);
                void ShowLandResult(report *reportor);
};
```

โครงสร้างการเก็บข้อมูล

```
struct LandData {
        in_addr host;  // แอดเดรสของโฮสต์ที่ถูกโจมตี
        int count; // จำนวนครั้งการโจมตี
        int start_hour;  // เวลาเริ่มต้น
        int start_min;  // เวลาเริ่มต้น
        int start_sec;  // เวลาเริ่มต้น
        int end_hour; // เวลาล่าสุด
        int end_min; // เวลาล่าสุด
        int end_sec; // เวลาล่าสุด
};
```

- void LAND (node *data)

จุดมุ่งหมาย : ตรวจสอบการโจมตีที่มีการส่งแพ็กเก็ตที่ทำให้เกิดการวนลูป (Land Attack)

ขั้นตอนการทำงาน :

1. สร้างลูปให้อ่านข้อมูลจนกว่าจะจบไฟล์ โดยภายในลูปให้ตรวจสอบแอดเดรสต้นทางและแอดเดรสปลายทางว่าเป็นค่าเดียวกันหรือไม่ หากเป็นค่าเดียวกันให้เก็บค่าไว้ในบัฟเฟอร์
2. เมื่อครบ 1 นาทีก็นำค่าที่เก็บไว้มาสร้างเป็นรายงานแล้วส่งรายงานไปให้ส่วนออกรายงานจัดการต่อ

void ShowLandResult(sem_t *sem_report,report *reporter)

จุดมุ่งหมาย : แสดงผลการโจมตีที่มีการส่งแพ็กเก็ตที่ทำให้เกิดการวนลูป (Land Attack)

ขั้นตอนการทำงาน :

1. เมื่อครบ 1 นาทีก็นำค่าที่เก็บไว้มาสร้างเป็นรายงานแล้วส่งรายงานไปให้ส่วนออกรายงานจัดการต่อ

```
class IPdefragment {
        private:
                config *con;
                int overflow;    // overflow flag 0 = overflow else 1;
                report_node report_overlap;
                report_node report_gap;
                struct OverlapData overlap[STD_BUF];
```

```
        struct OverlapData gap[STD_BUF];
        int fragcount;
        int gapcount;
        node *data;
public:
        void Initialize();              // Initialize Buffer for Use;
        void FoundInBuffer();  // Do all job if same-id Frame recieved;
        void NotFoundInBuffer();  // Do all job if new-id Frame recieved;
        void FixAndReduce();       // Reassemble and Reduce the tupple;
        void AddNewTupple();     // Add new tupple to Buffer;
        void showbuffer();
        void NormalCheck(int);     // Checking normal reassemble
        void AbnormalCheck(int);  // Checking Abnormal reassemble
        void AddOverlapID(int);   // AddID for OverlapFrame in Buffer
        void ShowResult(report *reportor);    // Show fragment Result
        void GapDetect(int i);    // Logging gap in Reassemble method
        void RealTimeFragChk(node *data1);      // For Checking Fragment
                                                    in Realtime
};
```

โครงสร้างการเก็บข้อมูล

```
struct OverlapData {
        in_addr ip;  // แอดเดรสของโฮสต์ที่ถูกโจมตี
        u_long count;  // จำนวนครั้งการโจมตี
        int id;  //หมายเลข Id ของแพ็กเก็ต
        int start_hour;  // เวลาเริ่มต้น
        int start_min;  // เวลาเริ่มต้น
        int start_sec;   // เวลาเริ่มต้น
        int end_hour;  //เวลาล่าสุด
        int end_min;  //เวลาล่าสุด
        int end_sec;  //เวลาล่าสุด
        };
```

- void RealTimeFragChk ( )

จุดมุ่งหมาย : ทำการวิเคราะห์ระบบจากบัฟเฟอร์ ซึ่งสามารถทำงานได้แบบตามเวลาจริง สามารถตรวจสอบการโจมตีแบบแพ็กเก็ตเหลื่อมกัน ซ้อนทับกัน หรือประกอบกันไม่ได้

ขั้นตอนการทำงาน :

1. ตรวจสอบว่างแพ็กเก็ตนั้นมีการแฟร็กเมนเตชันหรือไม่ ถ้าไม่มีการแฟร็กเมนเตชันก็ไม่นำแพ็กเก็ตนั้นมาวิเคราะห์
2. ตรวจสอบค่าหมายเลขแพ็กเก็ต (ID) ของแพ็กเก็ตนั้นว่ามีอยู่ใน Fragment Buffer หรือไม่ถ้ามี ให้เรียกฟังก์ชัน FoundInBuffer ( ) แต่ถ้าไม่มี ให้เรียกฟังก์ชัน NotFoundBuffer ( )

- void Initialize ( )

จุดมุ่งหมาย :ตั้งค่าเริ่มต้นของตัวแปรต่างๆ ที่จำเป็นต้องใช้

- void NotFoundInBuffer ( )

จุดมุ่งหมาย :ทำงานทุกๆ ขั้นตอน ในกรณีที่มีแพ็กเก็ตที่มีหมายเลขแพ็กเก็ตไม่ซ้ำกับข้อมูลใน Fragment Buffer

ขั้นตอนการทำงาน :

1. เพิ่มข้อมูลของแพ็กเก็ตที่รับเข้ามาใหม่ใน Fragment Buffer ซึ่งมี Lmerge, Rmerge, IP, ID, tupple count, count, tupple และ flag
2. ถ้าข้อมูลเต็มบัฟเฟอร์ (มีแพ็กเก็ตเข้ามามากกว่าที่ได้จองบัฟเฟอร์เอาไว้) แสดงว่ามีแนวโน้มการเกิด flooding จึงมีการเรียกใช้ฟังก์ชัน CheckFlood ( ) ถ้าบัฟเฟอร์เต็ม

- void FoundInBuffer ( )

จุดมุ่งหมาย : ทำงานทุกๆ ขั้นตอนในกรณีที่มีแพ็กเก็ตที่มีหมายเลขแพ็กเก็ตซ้ำกับข้อมูลใน Fragment Buffer

ขั้นตอนการทำงาน :

1. ทำงานอัพเดตค่าแฟล็กของข้อมูลที่มีหมายเลขแพ็กเก็ตเดียวกัน
2. หลังจากที่อัพเดตแฟล็กแล้ว จึงตรวจสอบแฟล็กว่าสามารถรีแอสเซมเบิ้ลได้หรือไม่
3. กรณีที่แพ็กเก็ตใหม่ที่รับเข้ามาสามารถรีแอสเซมเบิ้ลได้ ให้ทำการรีแอสเซมเบิ้ล โดยเรียกฟังก์ชัน FixAndReduce ( ) ถ้าทำไม่ได้ก็ให้

เพิ่มข้อมูล tupple ลงไปในบัฟเฟอร์ โดยเรียกฟังก์ชัน AddNewTupple ( ) มาทำงาน

4. ในกรณีที่เรียกใช้ฟังก์ชัน FixAndReduce จนค่า flag = 3 และ tupple_count = 1 แสดงว่าแพ็กเก็ตที่ได้รับมาสมบูรณ์ ไม่ผิดปกติ ให้ลบข้อมูลในส่วนแพ็กเก็ตเหล่านั้นออกทั้งหมด

- void NormalCheck (int)

จุดมุ่งหมาย : ตรวจสอบว่าแพ็กเก็ตที่รับเข้ามาสามารถรีแอสเซมเบิ้ลได้หรือไม่

การทำงาน :

1. ตรวจสอบของซ้ายของแพ็กเก็ตที่รับเข้ามาว่าเท่ากับขอบขวาของแพ็กเก็ตที่รับเข้ามาแล้วหรือไม่
2. ตรวจสอบของขวาของแพ็กเก็ตที่รับเข้ามาว่าเท่ากับขอบซ้ายของแพ็กเก็ตที่รับเข้ามาแล้วหรือไม่

- void AbnormalCheck ( )

จุดมุ่งหมาย : ตรวจสอบว่าแพ็กเก็ตที่รับเข้ามา มีการรีแอสเซมเบิ้ลที่ผิดปกติหรือไม่ ซึ่งสามารถตรวจสอบได้ 3 แบบ คือ

1. แพ็กเก็ตใหม่มีการซ้อนทับกับส่วนหน้าของแพ็กเก็ตเดิม
2. แพ็กเก็ตใหม่มีการซ้อนทับกับส่วนหลังของแพ็กเก็ตเดิม
3. แพ็กเก็ตใหม่มีการซ้อนทับกับแพ็กเก็ตเดิมทั้งแพ็กเก็ต

การทำงาน : สร้างเงื่อนไขตรวจสอบขอบซ้าย ขวาของแพ็กเก็ตใหม่ว่า มีการเหลื่อมหรือซ้อนทับกับแพ็กเก็ตเดิมหรือไม่ ถ้ามีการเหลื่อมหรือซ้อนทับกัน ก็บันทึกข้อมูลความผิดปกติไว้ใน Overlap Buffer

- void AddOverlapID ( )

จุดมุ่งหมาย : บันทึกหรืออัพเดตข้อมูลการเกิดการเหลื่อมล้ำกันของแพ็กเก็ต (Overlap)

การทำงาน : ค้นหาข้อมูลหมายเลขแพ็กเก็ตว่ามีอยู่ใน Overlap Buffer หรือไม่

1. ถ้าไม่มี ให้เพิ่มข้อมูลการโจมตี ซึ่งได้แก่ ID, Destination IP, count และเวลาของการเริ่มโจมตี
2. ถ้ามีอยู่แล้ว ให้อัพเดตข้อมูล count และเวลาของการโจมตี

- void ShowResult (report *reporter)

จุดมุ่งหมาย : แสดงผลข้อมูลการโจมตีที่เกิดจากแฟร็กเมนเตชัน

การทำงาน : ถ้าพบรายงานการโจมตี ให้ทำการสร้างรายงานแล้วส่งไปให้ส่วนการออกรายงานจัดการต่อ

- void AddNewTupple ( )

จุดมุ่งหมาย :เพิ่มข้อมูลของ tupple ลงใน Fragment Buffer

ขั้นตอนการทำงาน :

1. ตรวจสอบใน Fragment Buffer ที่มีหมายเลขแพ็กเก็ตเดียวกับแพ็กเก็ตใหม่ ว่าจำนวน tupple มากกว่าที่กำหนดหรือไม่
2. ถ้ามีจำนวน tupple มากเกินค่าที่กำหนด แสดงว่าแพ็กเก็ตไม่สามารถประกอบแพ็กเก็ตได้ เนื่องจากเกิดช่องว่างระหว่างแพ็กเก็ต จึงบันทึกความผิดปกตินี้ลงใน Gap Frame Buffer
3. เพิ่มข้อมูล tupple ใหม่ลงไปใน Fragment Buffer โดยใช้ค่า lFragOff และ rFragOff ของ tupple

- void GapDetect ( )

จุดมุ่งหมาย :เก็บค่าของข้อมูลในกรณีที่ไม่สามารถประกอบแพ็กเก็ตได้ เพราะมีช่องว่างระหว่างแพ็กเก็ต

การทำงาน : ตรวจสอบใน Gap Frame Buffer ว่ามีข้อมูลที่มีหมายเลขแพ็กเก็ตเดียวกับหมายเลขแพ็กเก็ตใหม่หรือไม่

1. ถ้าไม่มี ให้เพิ่มข้อมูลการโจมตี ซึ่งได้แก่ ID, Destination IP, count และเวลาของการเริ่มโจมตี
2. ถ้ามี ให้ทำการอัพเดตข้อมูล count และเวลาของการ โจมตี

- void FixAndReduce ( )

จุดมุ่งหมาย :เป็นกระบวนการรีแอสเซมเบิ้ล tupple ที่สามารถประกอบกันได้ เพื่อจำลองกระบวนการรีแอสเซมเบิ้ลของแพ็กเก็ต

การทำงาน :

ตรวจสอบค่า Lmerge กับ Rmerge ว่าค่าไหนมากกว่ากัน

1. ถ้าค่า Lmerge < Rmerge ให้นำ tupple ที่ Lmerge ต่อด้วย tupple ที่ Rmerge แล้วลดค่า tupple_count
2. ถ้าค่า Rmerge < Lmerge ให้นำ tupple ที่ Rmerge ต่อด้วย tupple ที่ Lmerge แล้วลดค่า tupple_count

```
class flood {
        private:
                unsigned int cnx_ttl; // ช่วงเวลาระหว่างแพ็กเก็ต
```

```
        unsigned int max_cnx_count ; // ค่าสูงสุดที่ยอมรับได้
        u_long all_pps; // จำนวนแพ็กเก็ตต่อวินาที
        float all_kbps; // ขนาดของข้อมูลที่รับมาต่อวินาที
        hostdb *flood_host;  // ที่เก็บรายละเอียดโฮสต์
        report_node report_flood; // รายงานการโจมตี
        report *reportors; // แอดเดรสของตัวสร้างรายงาน
        config *con;  // Configuration
public:
        void do_report_if_needed(hostdb_t *h,cnxInfo_t_flood *cnx,report
                                        *reportor);
        void expire_cnx();
        cnxInfo_t_flood *new_cnx(node *data, const char *kind);
        void modify_cnx(hostdb_t *h,node *data,cnxInfo_t_flood *cnx);
        void update_hdb_entry(node *data, const char *kind, hostdb_t *h);
        void create_hdb_entry(node *data, const char *kind);
        void generic_packet(node *data, const char *kind);
        void tcp_packet(node *data);  // แปลงข้อมูลให้เหมาะสม
        void udp_packet(node *data);  // แปลงข้อมูลให้เหมาะสม
        void icmp_packet(node *data);  // แปลงข้อมูลให้เหมาะสม
        void flood_run(node *data,report *reportor);  // เริ่มการทำงาน
};
```

โครงสร้างการเก็บข้อมูล

```
typedef struct _cnxInfo_flood
        {
        unsigned int cnx_count;  //จำนวนครั้งการโจมตี
        unsigned int port; // พอร์ตที่ถูกโจมตี
        struct in_addr firstip,lastip;  // IP แอดเดรส
        char *kind; // ชนิดของการโจมตี
        long first_cnxtime; // เวลาเริ่มต้น (จำนวนวินาที)
        long last_cnxtime;  // เวลาล่าสุด (จำนวนวินาที)
        int Shour;  // เวลาเริ่มต้น
        int Smin;  // เวลาเริ่มต้น
        int Ssec;  // เวลาเริ่มต้น
```

```
    int Ehour;  // เวลาล่าสุด
    int Emin;  // เวลาล่าสุด
    int Esec;  // เวลาล่าสุด
    u_long pps;  // จำนวนแพ็กเก็ตต่อวินาที
    float kbps; // ขนาดของข้อมูลที่รับเข้ามาต่อวินาที
} cnxInfo_t_flood;
```

- Void do_report_if_needed(hostdb_t *h,cnxInfo_t_flood *cnx,report *reportor)

  จุดมุ่งหมาย : สร้างรายงานถ้าเกิดว่าตัวนับมีค่ามากกว่าค่าที่ยอมรับได้

  ขั้นตอนการทำงาน :

  1. เช็คค่าของตัวนับกับค่าที่ยอมรับได้
  2. ถ้ามากกว่าก็สร้างรายงาน
  3. ส่งรายงานให้ส่วนออกรายงาน

- Void expire_cnx()

  จุดมุ่งหมาย : ตรวจสอบข้อมูลที่เก็บอยู่ในฐานข้อมูลว่าเกินช่วงเวลาหรือยัง ถ้าเกินแล้วก็ทำการลบข้อมูลนั้นทิ้ง โดยส่วนนี้จะทำงานทุกๆ 10 วินาที

  ขั้นตอนการทำงาน :

  1. ตรวจสอบว่าครบเวลาหรือยัง ถ้าครบก็เริ่มทำการตรวจสอบข้อมูล
  2. นำข้อมูลจากฐานข้อมูลขึ้นมาตรวจสอบทุกอันโดยเริ่มจากอันแรก
  3. ตรวจสอบว่าข้อมูลนั้นเกินช่วงเวลาหรือยัง ถ้าเกินก็ลบทิ้ง

- CnxInfo_t_flood *new_cnx(node *data, const char *kind)

  จุดมุ่งหมาย : สร้างข้อมูลของการวิเคราะห์ขึ้นมาใหม่

  ขั้นตอนการทำงาน :

  1. สร้างที่เก็บข้อมูลการวิเคราะห์
  2. นำข้อมูลที่จำเป็นจาก data ไปใส่ในที่เก็บที่สร้างขึ้น

- Void modify_cnx(hostdb_t *h,node *data,cnxInfo_t_flood *cnx)

  จุดมุ่งหมาย : อัพเดตค่าของตัวนับในข้อมูลที่มีอยู่แล้ว

  ขั้นตอนการทำงาน :

  1. กำหนดเวลาระหว่างแพ็กเก็ตใหม่
  2. เพิ่มค่าตัวนับการโจมตี
  3. คำนวณค่าแพ็กเก็ตที่รับเข้ามาต่อวินาที

4. คำนวณค่าข้อมูลที่รับเข้ามาต่อวินาที
5. ตรวจสอบว่าค่าตัวนับมากกว่าที่ยอมรับได้หรือไม่
6. ถ้ามากกว่าก็ทำการสร้างรายงานและลบข้อมูลนั้นทิ้ง

- Void generic_packet(node *data, const char *kind)

จุดมุ่งหมาย : ตรวจสอบว่ามีข้อมูลหรือยัง ถ้ายังก็จะสร้างขึ้นมาใหม่ แต่ถ้ามีแล้วก็จะไปอัพเดตค่าของข้อมูล

ขั้นตอนการทำงาน :

1. ตรวจสอบว่ามีข้อมูลหรือไม่
2. ถ้าไม่มี ก็สร้างข้อมูลขึ้นมาใหม่
3. ถ้ามีก็อัพเดตข้อมูล

```
class scan {    private:
                    unsigned int cnx_ttl; // ช่วงเวลาระหว่างแพ็กเก็ต
                    unsigned int max_cnx_count ; // ค่าสูงสุดที่ยอมรับได้
                    hostdb *scan_host; // ที่เก็บข้อมูลของโฮสต์
                    report_node report_scan; // รายงานการวิเคราะห์
                    config *con; // Configuration
                public:
                    void do_report_if_needed(hostdb_t *h,cnxInfo_t *cnx,report
                                                *reportor);
                    void expire_cnx(report *reportor);
                    char *kind_cnxInfo(struct tcphdr *tcp);
                    cnxInfo_t *new_cnx(node *data, const char *kind);
                    void modify_cnx(node *data,cnxInfo_t *cnx);
                    void update_hdb_entry(node *data, const char *kind, hostdb_t *h);
                    void create_hdb_entry(node *data, const char *kind);
                    void generic_packet(node *data, const char *kind);
                    void tcp_packet(node *data); // แปลงข้อมูลให้เหมาะสม
                    void udp_packet(node *data); // แปลงข้อมูลให้เหมาะสม
                    void icmp_packet(node *data); // แปลงข้อมูลให้เหมาะสม
                    void scandetect_run(node *data,report *reportor); // เริ่มการทำงาน
                };
```

โครงสร้างการเก็บข้อมูล

```
typedef struct _cnxInfo
    {
        unsigned int cnx_count;  // จำนวนครั้งการสแกน
        unsigned int first_port;  // พอร์ตน้อยสุดที่สแกน
        unsigned int last_port;  // พอร์ตสูงสุดที่สแกน
        struct in_addr firstip,lastip;  //  ไอพีแอดเดรส
        char *kind;  // ชนิดของการสแกน
        long first_cnxtime; // เวลาเริ่มต้น (จำนวนวินาที)
        long last_cnxtime;  // เวลาล่าสุด (จำนวนวินาที)
        int Shour; // เวลาเริ่มต้น
        int Smin; // เวลาเริ่มต้น
        int Ssec; // เวลาเริ่มต้น
        int Ehour;  // เวลาล่าสุด
        int Emin;  // เวลาล่าสุด
        int Esec;  // เวลาล่าสุด
    } cnxInfo_t;
```

- Void do_report_if_needed(hostdb_t *h,cnxInfo_t_flood *cnx,report *reportor)

  จุดมุ่งหมาย : สร้างรายงานถ้าเกิดว่าตัวนับมีค่ามากกว่าค่าที่ยอมรับได้

  ขั้นตอนการทำงาน :

  1. เช็คค่าของตัวนับกับค่าที่ยอมรับได้
  2. ถ้ามากกว่าก็สร้างรายงาน
  3. ส่งรายงานให้ส่วนออกรายงาน

- Void expire_cnx()

  จุดมุ่งหมาย : ตรวจสอบข้อมูลที่เก็บอยู่ในฐานข้อมูลว่าเกินช่วงเวลาหรือยัง ถ้าเกินแล้วก็ทำการลบข้อมูลนั้นทิ้ง โดยส่วนนี้จะทำงานทุกๆ 10 วินาที

  ขั้นตอนการทำงาน :

  1. ตรวจสอบว่าครบเวลาหรือยัง ถ้าครบก็เริ่มทำการตรวจสอบข้อมูล
  2. นำข้อมูลจากฐานข้อมูลขึ้นมาตรวจสอบทุกอันโดยเริ่มจากอันแรก
  3. ตรวจสอบว่าข้อมูลนั้นเกินช่วงเวลาหรือยัง
  4. ถ้าเกิน ให้เรียกเมธอด do_report_if_needed
  5. ถ้าไม่เกินก็ปล่อยไว้

- CnxInfo_t_flood *new_cnx(node *data, const char *kind)

  จุดมุ่งหมาย : สร้างข้อมูลของการวิเคราะห์ขึ้นมาใหม่

  ขั้นตอนการทำงาน :

  1. สร้างที่เก็บข้อมูลการวิเคราะห์
  2. นำข้อมูลที่จำเป็นจาก data ไปใส่ในที่เก็บที่สร้างขึ้น

- Char *kind_cnxInfo(struct tcphdr *tcp)

  จุดมุ่งหมาย : ตรวจสอบรูปแบบของการสแกน แล้วคืนค่าเป็นชนิดของการสแกน

  ขั้นตอนการทำงาน :

  1. อ่านแพ็กเก็ตข้อมูล ทีซีพี
  2. อ่าน Flag ต่าง
  3. นำค่า Flag ที่อ่านมาเทียบกับค่ารูปแบบต่างๆ
  4. คืนค่าชนิดของการสแกน

- Void modify_cnx(hostdb_t *h,node *data,cnxInfo_t_flood *cnx)

  จุดมุ่งหมาย : อัพเดตค่าของตัวนับในข้อมูลที่มีอยู่แล้ว

  ขั้นตอนการทำงาน :

  1. กำหนดเวลาระหว่างแพ็กเก็ตใหม่
  2. เพิ่มค่าตัวนับการสแกน

- Void generic_packet(node *data, const char *kind)

  จุดมุ่งหมาย : ตรวจสอบว่ามีข้อมูลหรือยัง ถ้ายังก็จะสร้างขึ้นมาใหม่ แต่ถ้ามีแล้วก็จะไป อัพเดตค่าของข้อมูล

  ขั้นตอนการทำงาน :

  1. ตรวจสอบว่ามีข้อมูลหรือไม่
  2. ถ้าไม่มี ก็สร้างข้อมูลขึ้นมาใหม่
  3. ถ้ามีก็อัพเดตข้อมูล

```
class hostdb {
        private:
                hostdb_t *host_hash[HASH_SIZE];  // ตาราง Hash Table
                hostdb_t *first_host; //  ตัวชี้โฮสต์ตัวแรก
                hostdb_t *last_host; // ตัวชี้โฮสต์ตัวสุดท้าย
        public:
```

```
        hostdb_t *hostdb_search(node *data);
        hostdb_t *hostdb_new(node *data);
        void hostdb_del(hostdb_t *h, const unsigned int pid);
        hostdb_t  *get_host_hash(int i) {return host_hash[i];}
};
```

โครงสร้างการเก็บข้อมูล

```
typedef struct _hostdb {
        struct ip *ip; // IP แอดเดรส
        u_int srcport,dstport;  // src and dst port
        int key_cache;  // Hash key
        unsigned long *pdata;  // ที่เก็บตัวชี้ของข้อมูลการวิเคราะห์
        struct _hostdb *prev, *next; // ตัวชี้โฮสต์ในตาราง
        struct _hostdb *hprev, *hnext;  // ตัวชี้โฮสต์ระหว่างข้อมูลที่มี
} hostdb_t;
```

- hostdb_t *hostdb_search(node *data)

  จุดมุ่งหมาย : หาข้อมูลของโฮสต์ใน Hash Table ถ้ามีก็คืนค่าชี้ไปที่ข้อมูล ถ้าไม่พบก็คืนค่า NULL

  ขั้นตอนการทำงาน :

  1. ตรวจสอบว่ามีข้อมูลของโฮสต์นั้นหรือไม่ โดยใช้แอดเดรสต้นทาง แอดเดรสปลายทางและโพรโตคอล มาสร้างเป็น Key ในการเข้าถึงข้อมูล
  2. ถ้ามี คืนค่าตัวชี้ที่ข้อมูลของโฮสต์นั้น
  3. ถ้าไม่พบ คืนค่า NULL

- hostdb_t *hostdb_new(node *data)

  จุดมุ่งหมาย : สร้างข้อมูลของโฮสต์ขึ้นมาใหม่ โดยใช้ key ในการเข้าถึงข้อมูลเป็นแอดเดรสต้นทาง แอดเดรสปลายทาง และโพรโตคอล รวมกัน

  ขั้นตอนการทำงาน :

  1. สร้างข้อมูลของโฮสต์
  2. คืนค่าตัวชี้ข้อมูลของโฮสต์ที่สร้างขึ้น

- void hostdb_del(hostdb_t *h, const unsigned int pid)

จุดมุ่งหมาย : ลบข้อมูลของโฮสต์ที่ชี้โดย h ออกจาก Hash Table

ขั้นตอนการทำงาน :

1. ลบข้อมูลโฮสต์
2. คืนหน่วยความจำ
3. จัดเรียงค่าตัวชี้ใหม่ (next, prev)

```
class signature {
        private:
            s_node *tcp_tree;  // ตัวชี้ tree สำหรับเก็บแพตเทิร์นทีซีพี
            s_node *udp_tree;  // ตัวชี้ tree สำหรับเก็บแพตเทิร์นยูดีพี
            s_node *icmp_tree;  // ตัวชี้ tree สำหรับเก็บแพตเทิร์น ICMP
            report_node report_signature; // ที่เก็บรายงาน
            report *reportors; // ตัวชี้ report ใช้เรียก Callback
            config *con; // Configuration
        public:
            c_node *new_c_node(s_node *service,  u_int sig_id, u_char
                                *pattern, u_int pattern_len, char * name,
                                u_char pattern_mode );
            s_node *new_s_node( u_long src_number,u_long dest_number );
            s_node * get_s_node( char *protocol, u_long src_number, u_long
                                dst_number ); //ดึงค่าตัวชี้ของ s_node
            c_node *detect_signature( node *data,u_char * capital, s_node
                                **s_node_ptr,  c_node **c_node_ptr );
            int init_signature(char *sig_file);
            int make_pattern( u_char *pattern, u_int patternsize, char
                                *sig_reader, char *endof_sig, u_char
                                pattern_mode );
            int make_bin_pattern( u_char *sig_writer, char *bin_reader, char
                                *endof_bin, u_char pattern_mode );
            int dump_c_tree(s_node * s_tree ); // แสดงรายละเอียดของแพ็ต
                                เทิร์นที่เก็บ

            void dump_s_tree(s_node * s_tree );  // แสดงรายละเอียดของ
```

```
                                                    Service
        void free_s_tree( s_node * s_tree);  // คืนหน่วยความจำของ s_tree
        void free_c_tree( c_node * c_tree); // คืนหน่วยความจำของ c_tree
        int boyer_moore_match ( u_char * data, int data_length, u_char *
                                        pattern, int pattern_length, u_char
                                        *delta1, u_char *delta2 );
        u_char *boyer_moore_delta1( u_char *pattern, u_char
                                        pattern_length); //จาก boyer&moore
        u_char *boyer_moore_delta2( u_char *pattern, u_char
                                        pattern_length);//จาก boyer&moore
        void signature_run(node *data,report *reportor);  // เริ่มการทำงาน
        void read_payload(node *data );  // อ่านข้อมูลจาก data
        int min(int value1, int value2);  // หาค่าน้อยสุด
        int max(int value1, int value2); // หาค่ามากสุด
};
```

โครงสร้างการเก็บข้อมูล

```
typedef struct content_node{
        struct content_node *next;  // เก็บตัวชี้แพตเทิร์นต่อไป
        void * s_node; // ตัวชี้ s_node
        u_char *pattern;  // แพตเทิร์นของการโจมตี
        int pattern_len;  // ความยาวของแพตเทิร์น
        u_char *delta1;  // ค่าที่ใช้เปรียบเทียบของ boyer_moore
        u_char *delta2;  // ค่าที่ใช้เปรียบเทียบของ boyer_moore
          char * name;  // ชื่อของการโจมตี
        u_char pattern_mode;   // รูปแบบของแพตเทิร์น
} c_node;

typedef struct service_node{
        u_long src_number; // พอร์ตต้นทาง
        u_long dest_number; // พอร์ตปลายทาง
        c_node *c_tree;  // ตัวชี้แพตเทิร์นการโจมตี
        struct service_node *next;  // ตัวชี้ Service ตัวต่อไป
} s_node;
```

- c_node *new_c_node(s_node *service, u_int sig_id, u_char *pattern, u_int pattern_len, char* name, u_char pattern_mode)

จุดมุ่งหมาย : สร้างที่เก็บข้อมูลของแพตเทิร์น โดยรับข้อมูลของ Service แพตเทิร์น ชื่อการโจมตีและรูปแบบของแพตเทิร์น

ขั้นตอนการทำงาน :

1. สร้างหน่วยความจำของ c_node
2. เก็บค่าของแพตเทิร์นและ ชื่อการโจมตี
3. คืนค่าตัวชี้ของข้อมูล

- s_node *new_s_node(u_long src_number,u_long dest_number)

จุดมุ่งหมาย : สร้างที่เก็บข้อมูลของ Service โดยรับข้อมูลของพอร์ตมากเป็นตัวสร้างข้อมูล ซึ่งตัว Service นี้จะทำหน้าที่แบ่งแยกแพตเทิร์นว่าต้องทำการตรวจสอบที่พอร์ตไหนบ้าง

ขั้นตอนการทำงาน :

1. สร้างหน่วยความจำของ s_node
2. เก็บค่าของพอร์ตต้นทางและปลายทาง
3. คืนค่าตัวชี้ของข้อมูล

- int init_signature(char *sig_file)

จุดมุ่งหมาย : สร้าง tree ของแพตเทิร์นต่างจากไฟล์ข้อมูลของการโจมตี โดยแบ่งแยกได้ 3 tree คือ TCP , UDP, ICMP และแต่ละ tree จะเก็บข้อมูลการโจมตีเฉพาะที่เป็นของตัวเอง

ขั้นตอนการทำงาน :

1. อ่านไฟล์ข้อมูลการโจมตี
2. ตรวจสอบว่ามี Service หรือยัง ถ้าไม่มีก็สร้างขึ้นมาใหม่ โดยเรียกเมธอด new_s_node
3. เรียกเมธอด make_pattern เพื่อจัดข้อมูลแพตเทิร์น
4. สร้าง c_node เพื่อเก็บแพตเทิร์นการโจทตี โดยเรียกเมธอด new_c_node แล้วก็เก็บข้อมูล
5. จัดเรียง tree ใหม่
6. วนกลับไปอ่านข้อมูลการโจมตีอันต่อไป

- c_node *detect_signature(node *data,u_char * capital, s_node **s_node_ptr, c_node **c_node_ptr)

จุดมุ่งหมาย : เปรียบเทียบข้อมูลที่รับเข้ามากับแพตเทิร์นของการโจมตีที่มีอยู่ โดยถ้าพบจะคืนค่าข้อมูลของการโจมตีนั้น แต่ถ้าไม่พบจะคืนค่า NULL

ขั้นตอนการทำงาน :

1. ค้นหา Service จาก พอร์ตต้นทางและปลายทางเพื่อเข้าถึงแพตเทิร์น
2. ถ้าพบก็เริ่มทำการเปรียบเทียบ ถ้าไม่พบก็คืนค่า NULL
3. เรียกเมธอด boyer_moore_match เพื่อเปรียบเทียบ ถ้าตรงจะคืนค่า c_node
4. ถ้าไม่ตรง ก็เปรียบเทียบกับแพตเทิร์นอันถัดไป
5. เมื่อหมดแล้ว ไม่ตรงเลย คืนค่า NULL

- int boyer_moore_match (u_char * data, int data_length, u_char * pattern, pattern, int pattern_length, u_char *delta1, u_char *delta2)

จุดมุ่งหมาย : เปรียบเทียบข้อมูลที่รับเข้ามากับแพตเทิร์นของการโจมตีที่มีอยู่ โดยถ้าพบจะคืนค่าข้อมูลของการโจมตีนั้น แต่ถ้าไม่พบจะคืนค่า NULL ในการเปรียบเทียบนั้นจะต้องใช้ค่าของ delta1 และ delta2 ซึ่งได้ถูกคำนวณมาจากแพตเทิร์นของการโจมตี

ขั้นตอนการทำงาน :

1. เริ่มเปรียบเทียบที่ตัวอักษรอันที่เท่ากับขนาดของแพตเทิร์น
2. ถ้าตัวอักษรนั้นไม่มีอยู่ในแพตเทิร์น ก็ให้ข้ามไปจำนวนเท่ากับความยาวของแพตเทิร์นแล้วเริ่มเปรียบเทียบใหม่
3. แต่ถ้ามีอยู่ในแพตเทิร์นก็จะ เปรียบเทียบต่อว่าตรงกับตำแหน่งหรือไม่ ถ้าตรงก็ย้อนกลับมา 1 ตัวอักษรเพื่อเทียบเทียบต่อ ทำไปเรื่อยๆ จนครบแพตเทิร์น ก็สรุปว่าพบแพตเทิร์นที่ต้องการ
4. ถ้ามีแต่ไม่ตรง ก็ให้ Shift ไปทางขวาเท่ากับค่า max(d1,d2)

   d1 = delta1[c] -(pattern_length-j) ;

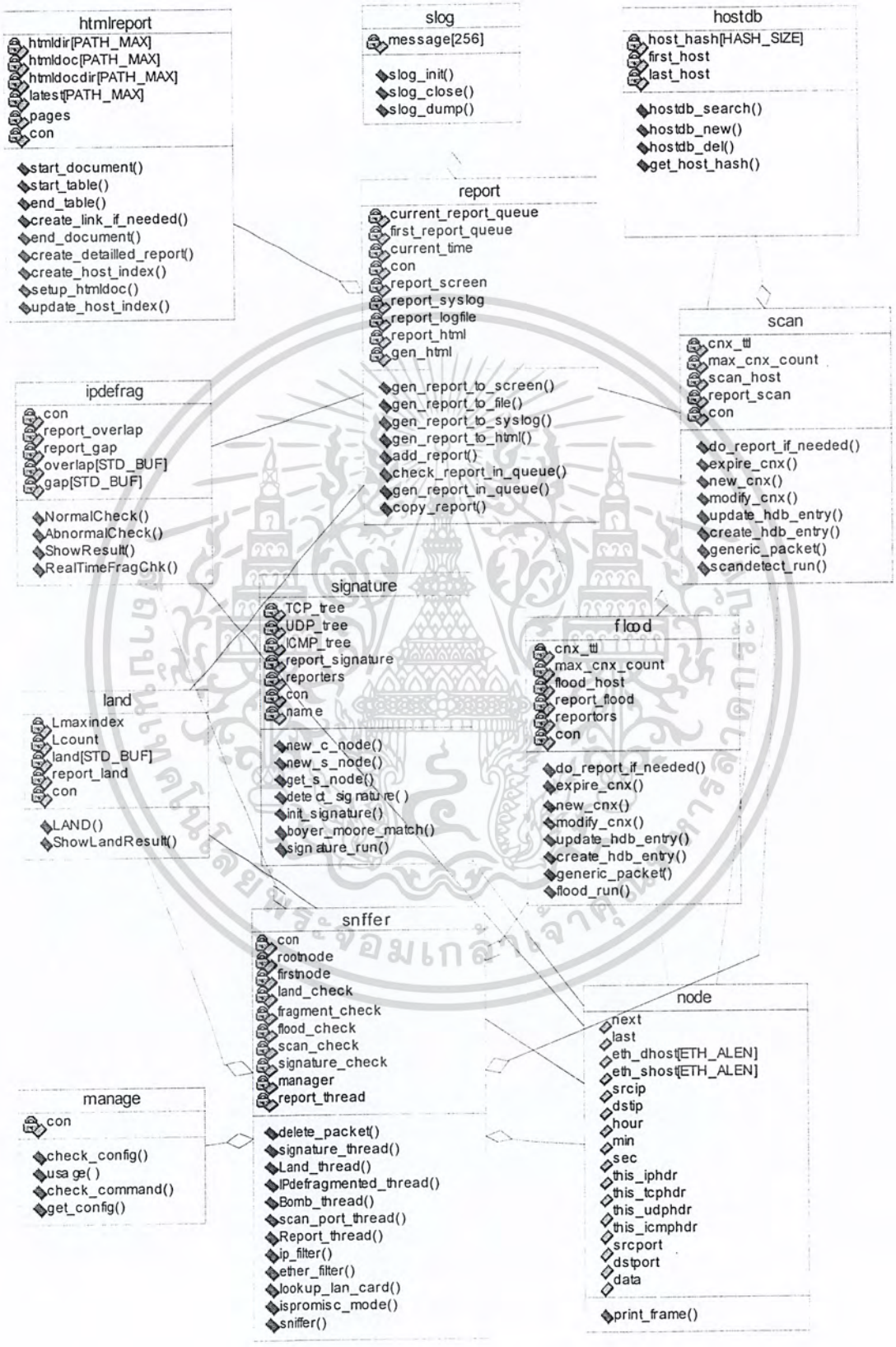
   d2 = delta2[j] ;

   j : ตัวชี้ตำแหน่งของแพตเทิร์นว่าเป็นตัวที่เท่าไร

   c : ตัวอักษรที่อยู่ในแพตเทิร์น

   pattern_length : ความยาวของแพตเทิร์น
5. เริ่มเปรียบเทียบใหม่
6. ทำไปจนหมดข้อมูล

5.4 คลาสไดอะแกรมของระบบ



รูปที่ 5-14 คลาสไดอะแกรมของระบบ

5.5 คุณสมบัติของระบบ

ระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ที่สร้างขึ้น มีความสามารถในการทำงานดังต่อไปนี้

1. เก็บข้อมูลของแพ็กเก็ต และสามารถแสดงข้อมูลของแพ็กเก็ตผ่านหน้าจอได้
2. ตรวจสอบการสแกนพอร์ตชนิดต่างๆได้ เช่น FIN, NULL, XMAS
3. ตรวจสอบการ Ping Sweep ได้
4. วิเคราะห์แพ็กเก็ตที่เกิดความผิดปกติแบบมีปริมาณมากเกินไปได้
4. ตรวจสอบการโจมตีแบบสเมิร์ฟ ได้
5. วิเคราะห์แพ็กเก็ตที่เกิดความผิดปกติแบบมีการทำแฟร็กเมนเตชันที่ผิดปกติได้
6. วิเคราะห์แพ็กเก็ตที่เกิดความผิดปกติแบบมีการส่งแพ็กเก็ตแบบวนลูปได้
7. ตรวจสอบการโจมตีในระดับแอพพลิเคชันได้
8. สามารถเพิ่มรูปแบบการโจมตีในชั้นแอพพลิเคชันได้โดยเป็นลักษณะของ rule base
9. วิเคราะห์แบบตามเวลาจริงก็ได้
10. เลือกวิเคราะห์เฉพาะเครื่องที่ต้องการได้
11. เลือกวิเคราะห์เฉพาะชนิดแพ็กเก็ตที่ต้องการได้
12. แสดงผลการวิเคราะห์ ได้ทั้งทางหน้าจอและเก็บลงล็อกไฟล์
13 เก็บผลการวิเคราะห์ลง Syslog
14. แสดงผลการวิเคราะห์ ทางเว็บได้
15. มีไฟล์คอนฟิกูเรชันเพื่อสามารถปรับแต่งค่าของโปรแกรมได้ง่าย
16. การทำงานเป็นแบบมัลติเธรด

5.6 ข้อจำกัดของระบบ

ระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์สร้างขึ้นโดยมีข้อจำกัดของการใช้งาน ดังต่อไปนี้

(1) ระบบนี้สร้างขึ้นบนระบบปฏิบัติการลีนุกซ์ซึ่งอาจแตกต่างกัน หากนำไปใช้งานในระบบอื่น

(2) ผู้ที่มีสิทธิ์รูท (root) เท่านั้นที่จะสามารถใช้งานโปรแกรมนี้ได้

(3) ระบบนี้สามารถตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ได้เฉพาะเครื่องปลายทางที่อยู่ในแชร์โดเมน (Share Domain) เดียวกันเท่านั้น

(4) ระบบนี้จะต้องติดตั้งอยู่บนเครือข่ายที่ใช้ฮับจึงจะใช้ประโยชน์ได้อย่างเต็มที่

บทที่ 6
การใช้งานระบบ

6.1 ฟังก์ชันการใช้งานของระบบ

ในการทำงานของระบบตรวจจับผู้บุกรุกทางเครือข่ายนี้ การตั้งค่าเพื่อใช้งานนั้น ต้องกระทำ 2 ขั้นตอน ได้แก่ การใส่ค่าแฟล็กเมื่อทำการเรียกโปรแกรม และ การตั้งค่าในไฟล์คอนฟิกูเรชันไฟล์

6.1.1 การตั้งเมื่อเริ่มการทำงานของโปรแกรม

6.1.1.1 การเรียกใช้งาน

ระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ที่พัฒนาขึ้นมานี้ สามารถเรียกใช้ได้โดยการเรียก **IsagNids** โดยสามารถเลือกฟังก์ชันใช้งานเองได้ ซึ่งมีฟังก์ชันการทำงานเป็นแฟล็กดังนี้

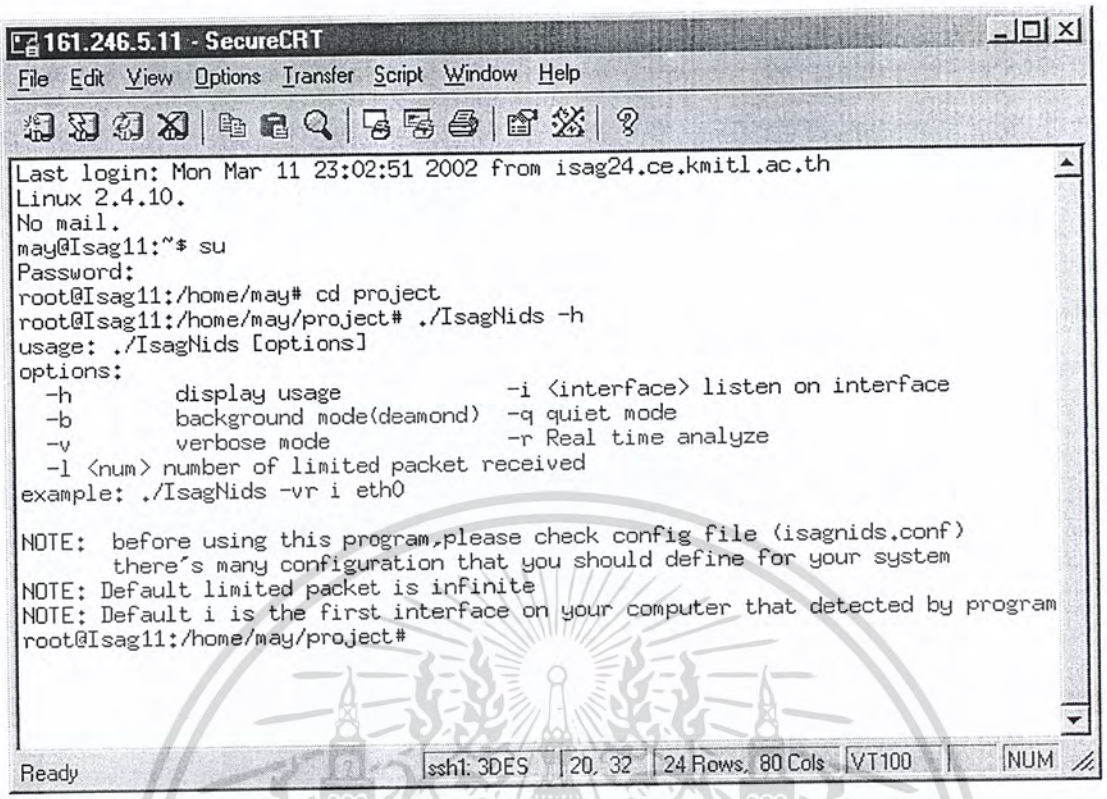
- h	แสดงส่วนช่วยเหลือ
- v	พิมพ์ข้อมูลของแพ็กเก็ตที่เก็บได้ออกหน้าจอ
- r	ตรวจสอบการโจมตีแบบตามเวลาจริง
- b	ให้ทำงานในโหมด daemond
- q	ไม่แสดงผลใดๆออกหน้าจอ
- i <interface>	เลือกการ์ดแลนที่จะใช้เป็นตัวจับแพ็กเก็ต
- l <num>	จำกัดจำนวนของแพ็กเก็ตที่รับเข้ามาวิเคราะห์

6.1.1.2 การเลือกใช้ฟังก์ชันการทำงาน

การทำงานของระบบแบ่งออกได้เป็น 3 ส่วนใหญ่ๆ คือ การแสดงส่วนช่วยเหลือ และ การเก็บข้อมูลของแพ็กเก็ตที่เข้าสู่ระบบ และการวิเคราะห์แพ็กเก็ตเหล่านั้น ซึ่งในการเลือกใช้ฟังก์ชันของระบบในการทำงานแต่ละส่วนนั้น ผู้ใช้สามารถเลือกใช้ได้แล้วแต่ความเหมาะสมของสภาพแวดล้อม และลักษณะของการใช้งาน ซึ่งมีข้อแนะนำในการใช้งานดังต่อไปนี้

6.1.1.2.1 การแสดงส่วนช่วยเหลือ

ในขั้นตอนนี้เป็นการแสดงส่วนช่วยเหลือของโปรแกรมเพื่อบอกผู้ใช้ถึงวิธีการเรียกโปรแกรม ว่ามีแฟล็กอะไรให้ใช้บ้าง เรียกโดย IsagNids –h โดยจะมีลักษณะดังรูป ที่ 6-1

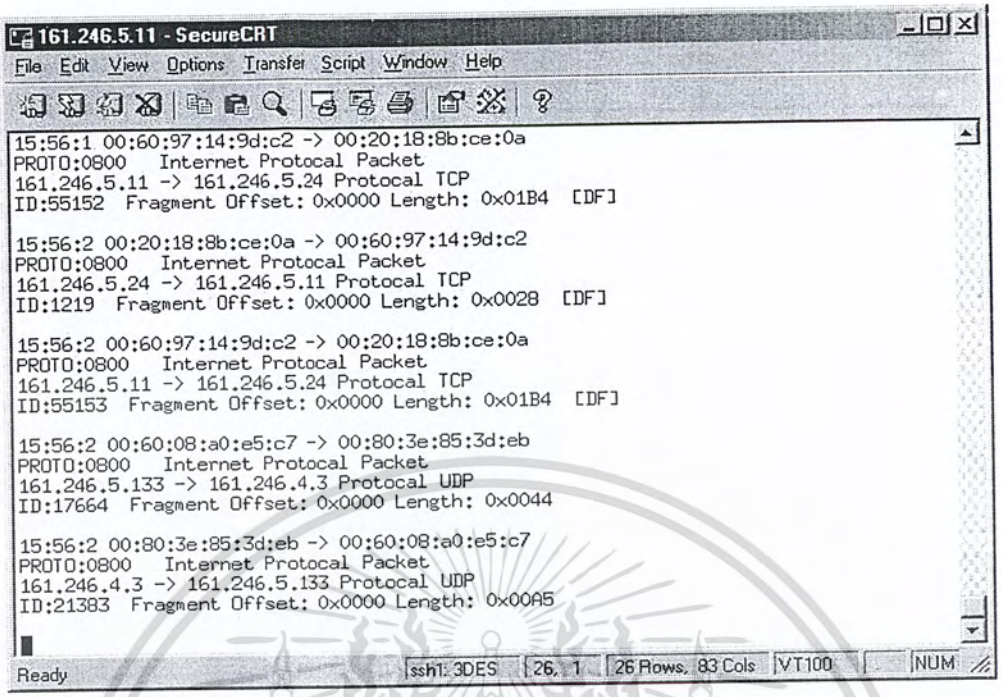


รูปที่ 6-1 แสดงการแสดงส่วนช่วยเหลือของโปรแกรม

6.1.1.2.2 การเก็บข้อมูล

ขั้นตอนนี้เป็นการเก็บข้อมูลของแพ็กเก็ตที่เข้าสู่ระบบ โดยผู้ใช้อาจเลือกให้แสดงข้อมูลของแพ็กเก็ตเหล่านั้นบนหน้าจอก็ได้ โดยกรณีที่ต้องการดูข้อมูลแพ็กเก็ต ผู้ใช้สามารถเลือกใช้ฟังก์ชัน **IsagNids –v** ซึ่งเมื่อสั่งคำสั่งดังกล่าว ระบบจะแสดงข้อมูลของแพ็กเก็ตที่เข้าสู่ระบบผ่านหน้าจอเทอร์มินอล โดยแสดงข้อมูลของเฮดเดอร์ของแพ็กเก็ตทั้งของทีซีพี/ไอพี และอีเทอร์เน็ตบางส่วน รวมทั้งวันและเวลาที่แพ็กเก็ตเหล่านั้นเข้ามาด้วย (ทำงานคล้ายกับเป็นตัวสนิฟเฟอร์) ดังรูปที่ 6-2

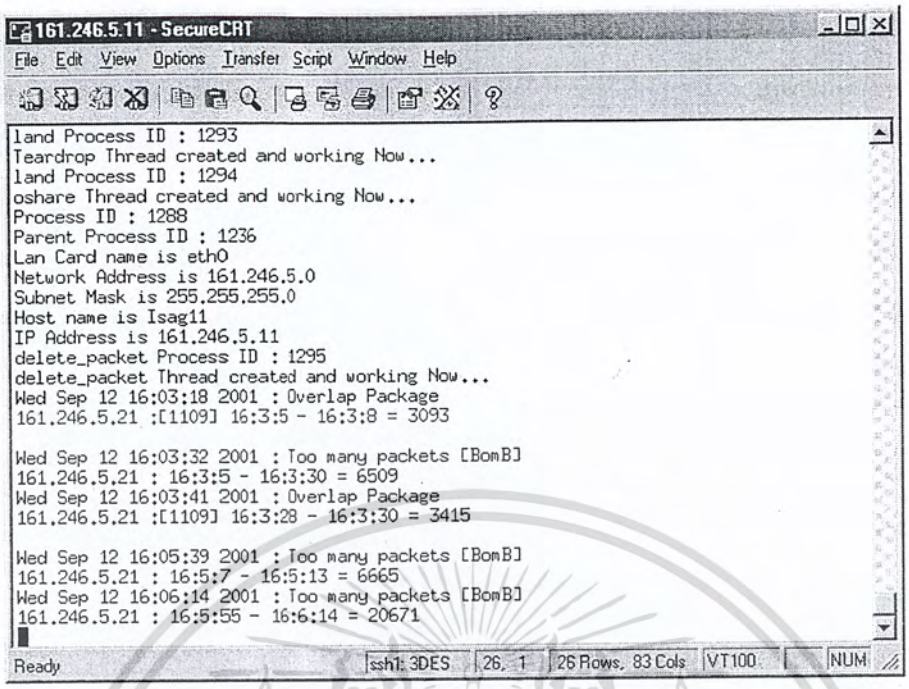
การดูข้อมูลของแพ็กเก็ตในลักษณะนี้มีข้อดีที่ได้รู้ว่าในเครือข่ายของเรามีแพ็กเก็ตอะไรผ่านเข้าออกบ้าง เพื่อเก็บข้อมูลเอาไว้วิเคราะห์ต่อไป



รูปที่ 6-2 แสดงการแสดงผลการเก็บข้อมูลแพ็กเก็ตผ่านหน้าจอเทอร์มินอล

6.1.1.2.3 การวิเคราะห์ข้อมูล

ในขั้นตอนของการวิเคราะห์ข้อมูลนั้น จะเป็นการวิเคราะห์แบบตามเวลาจริง คือ การนำข้อมูลของแพ็กเก็ตที่เข้ามาขณะนั้น มาวิเคราะห์เลย โดยเรียกใช้คำสั่ง **IsagNids –r** ทำให้ได้ข้อมูลวิเคราะห์การโจมตีตามเวลาจริงๆ โดยจะแสดงผลออกมาผ่านหน้าจอเทอร์มินอล และผ่านทางเว็บโดยจะแสดงผลทั้งเครื่องเป้าหมายที่ถูกโจมตี วันและช่วงเวลาที่ถูกโจมตี จำนวนแพ็กเก็ตที่โจมตีเข้ามา และชนิดของการโจมตีด้วย ดังรูปที่ 6-3 (ส่วนการแสดงผลทางเว็บเพจ จะกล่าวถึงในภายหลัง)



รูปที่ 6-3 แสดงหน้าจอการแสดงผลเมื่อเกิดการโจมตี

6.1.1.2.4 แฟล็กการใช้งานอื่นๆ

- b เป็นการทำงานในแบ็กกราวนด์ (Daemon)

-q ถ้าใช้จะทำให้ไม่มีการแสดงผลทางหน้าจอเกิดขึ้น จะมีข้อความเฉพาะช่วงเริ่มต้นโปรแกรม แต่หลังจากนั้น แม้ว่าจะเป็นการสั่งให้ วิเคราะห์ตามเวลาจริง หรือเป็นการให้เก็บข้อมูลแพ็กเก็ตที่เข้ามาก็จะไม่มีการแสดงค่าใดๆ ในหน้าจอใช้งาน

ตัวอย่างเช่น isagnids –rq จะเป็นการให้โปรแกรมทำงานวิเคราะห์ตามเวลาจริง แต่จะไม่มีการแสดงผลการวิเคราะห์พบการโจมตีที่ได้ขึ้นหน้าจอ

-i เป็นการเลือกเน็ตเวิร์กการ์ด(การ์ดแลน)ที่ใช้ในการตรวจจับ เช่น isagnids –r i eth0 โดยที่ถ้าไม่ตั้งค่าให้ โปรแกรมจะใช้ตัวที่ตรวจหาพบเป็นตัวแรก

-l เป็นการกำหนดจำนวนแพ็กเก็ตสูงสุดที่โปรแกรมจะรับ ถ้าไม่มีการตั้งค่าให้ ค่าที่ใช้จะเป็นอินฟินิต

6.1.2 การตั้งค่าในไฟล์คอนฟิกูเรชัน

ไฟล์ชื่อ isagnids.conf เป็นไฟล์คอนฟิกูเรชันที่ใช้กำหนดค่าในการทำงานให้แก่ระบบ โดยมีรายละเอียดดังนี้

pcap_filter เป็นการกำหนดแพ็กเก็ตที่จะรับเข้ามา โดยถ้านอกเหนือจากที่กำหนดก็จะไม่พิจารณา *ตัวอย่าง* pcap_filter = "tcp dst port 21" เลือกรับแต่แพ็กเก็ตทีซีพีที่มีพอร์ตปลายทาง คือ 21

report_screen เป็นการกำหนดว่าจะให้แสดงการรายงานผลที่หน้าจอที่ทำงานอยู่หรือไม่ report_screen = 1; คือให้แสดง และถ้า report_screen = 0; คือไม่แสดง

report_html เป็นการกำหนดว่าจะให้แสดงผลการวิเคราะห์ในรูปแบบเว็บเพจหรือไม่ โดยที่ report_html = 1; คือแสดง และ report_html = 0; คือไม่แสดง

www_directory เป็นไดเรกทอรี ที่ใช้เก็บไฟล์แสดงผลข้อมูลทางเว็บเพจ ตัวอย่าง www_directory = "/www/htdocs/nids";

report_syslog เป็นการกำหนดว่าจะให้เก็บผลรายงานลง syslog system API หรือไม่ โดยที่ ถ้า report_syslog = 1; คือเก็บ และ report_syslog = 0; ไม่เก็บ

report_logfile เป็นการกำหนดว่าจะให้เก็บผลรายงานลงในล็อกไฟล์ที่กำหนดหรือไม่ โดยที่ถ้า report_logfile = 1; คือเก็บ และถ้า report_logfile = 0; คือไม่เก็บ

log_dir เป็นการกำหนดไดเรกทอรีในการเก็บล็อกไฟล์ที่จะเก็บ ตามที่กำหนดว่าจะเก็บไว้ใน report_logfile *ตัวอย่าง* log_dir = "/var/log/isagnids.log";

log_file เป็นการกำหนดชื่อไฟล์ที่ต้องการให้เป็นล็อกไฟล์ที่จะเก็บข้อมูลการวิเคราะห์ *ตัวอย่าง* log_file = "isagnids.log";

signature_file เป็นการกำหนดไฟล์ที่ใช้เก็บข้อมูลซิกเนเจอร์ที่ใช้ในการวิเคราะห์การโจมตี *ตัวอย่าง* signature_file = "/home/may/project/signature.txt";

scan_ttl เป็นการกำหนดค่าช่วงเวลาที่ใช้ในการวิเคราะห์ว่าเป็นการสแกนพอร์ตหรือไม่ โดยเป็นผลต่างของระยะเวลาระหว่างการส่งแพ็กเก็ต *ตัวอย่าง* scan_ttl = 15;

scan_max_count เป็นการกำหนดจำนวนแพ็กเก็ตที่เข้ามา โดยถ้ามีพฤติกรรมที่สอดคล้องมากกว่าค่านี้ก็จะสรุปว่าเป็นการสแกนพอร์ต *ตัวอย่าง* scan_max_count = 30;

flood_ttl เป็นการกำหนดค่าระยะเวลาที่ต่างระหว่าง 2 แพ็กเก็ตที่เข้ามา ที่จะถือว่าเป็นการโจมตีแบบ flood *ตัวอย่าง* flood_ttl = 6;

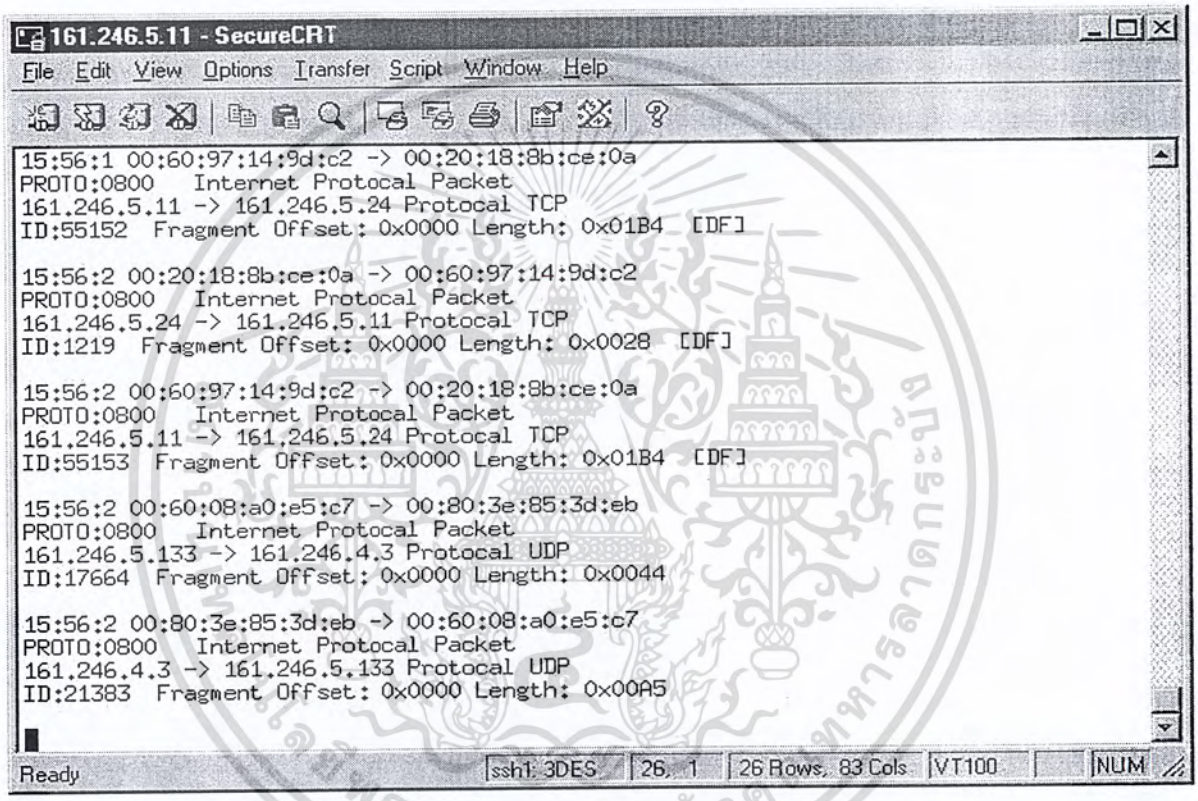
flood_max_count เป็นการกำหนดค่าที่ใช้ในการวิเคราะห์การ flood โดยเป็นค่าที่ยอมรับได้ของจำนวนแพ็กเก็ตที่เข้ามา โดยถ้าเกินกว่านี้จะถือว่าเป็นการโจมตี *ตัวอย่าง* flood_max_count = 300;

6.2 ตัวอย่างหน้าจอการทำงานของโปรแกรม

6.2.1 การเก็บข้อมูล

ในการเก็บข้อมูลแพ็กเก็ตนั้น ระบบจะบันทึกข้อมูลต่างๆ ดังต่อไปนี้

- เวลาที่แพ็กเก็ตนั้นเข้ามา
- แอดเดรสต้นทาง (Source Address)
- แอดเดรสปลายทาง (Destination Address)
- โพรโตคอลที่ใช้
- หมายเลขประจำแพ็กเก็ต (ID)
- แฟร็กเมนต์ออปเซต (Fragment Offset)
- ความยาวของแพ็กเก็ต (Length)



รูปที่ 6-4 แสดงการแสดงผลข้อมูลของแพ็กเก็ตที่เข้าสู่ระบบ

6.2.2 การวิเคราะห์ข้อมูล

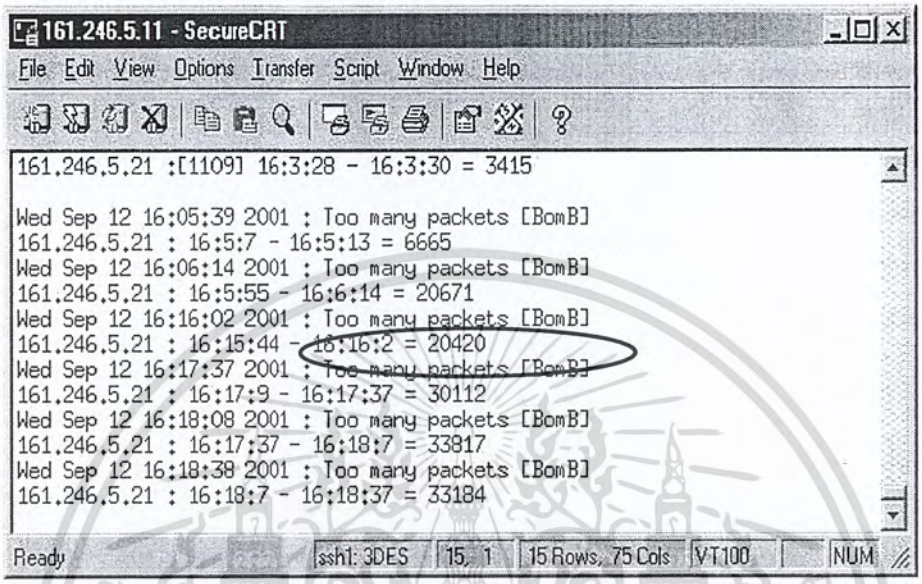
ในการวิเคราะห์ข้อมูลนั้น หากพบว่าแพ็กเก็ตเกิดความผิดปกติ คือวิเคราะห์แล้วสรุปว่าเป็นการบุกรุก ก็จะแจ้งเตือนออกทางหน้าจอ(ในกรณีที่ไม่ได้อยู่ในโหมดของการให้ไม่แสดงผล) แล้วอาจเก็บไว้เป็นล็อกไฟล์ หรือ แสดงผ่านทางเว็บเพจ ซึ่งทั้งนี้ขึ้นอยู่กับการตั้งค่าในไฟล์คอนฟิกูเรชัน ซึ่งมีข้อมูลที่แสดงและจัดเก็บ ดังต่อไปนี้

- วัน เดือน ปี ที่แพ็กเก็ตเหล่านั้นเข้ามา
- ชนิดของการโจมตี
- แอดเดรสปลายทาง (Destination Address)

- ช่วงเวลาที่แพ็กเก็ตเหล่านั้นเข้ามา
- จำนวนแพ็กเก็ตที่เข้ามา

ซึ่งผลการวิเคราะห์ถ้าเกิดความผิดปกติ จะมีการแสดงเตือน ดังตัวอย่างต่อไปนี้

(1) การส่งแพ็กเก็ตปริมาณมาก

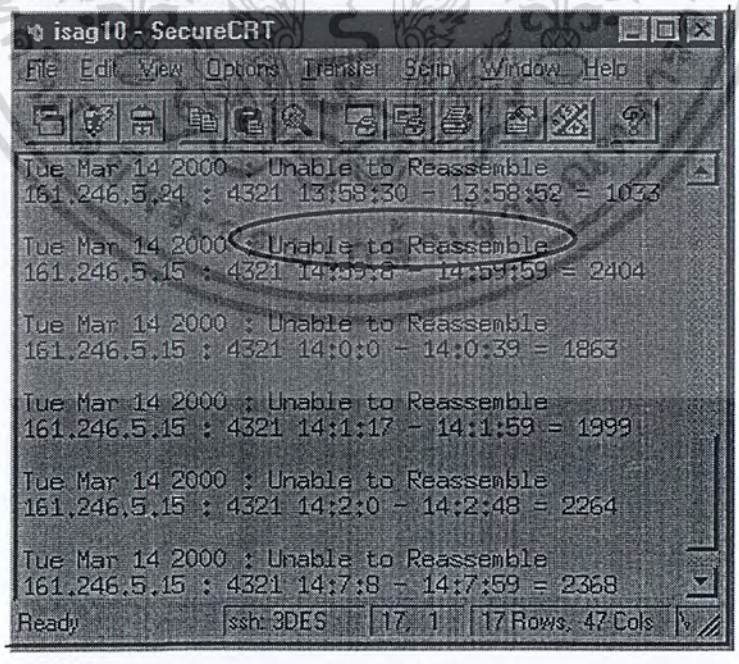


รูปที่ 6-5 แสดงการแจ้งเตือนเมื่อตรวจพบการโจมตีแบบแพ็กเก็ตปริมาณมาก

(2) แฟร็กเมนเตชันผิดปกติ

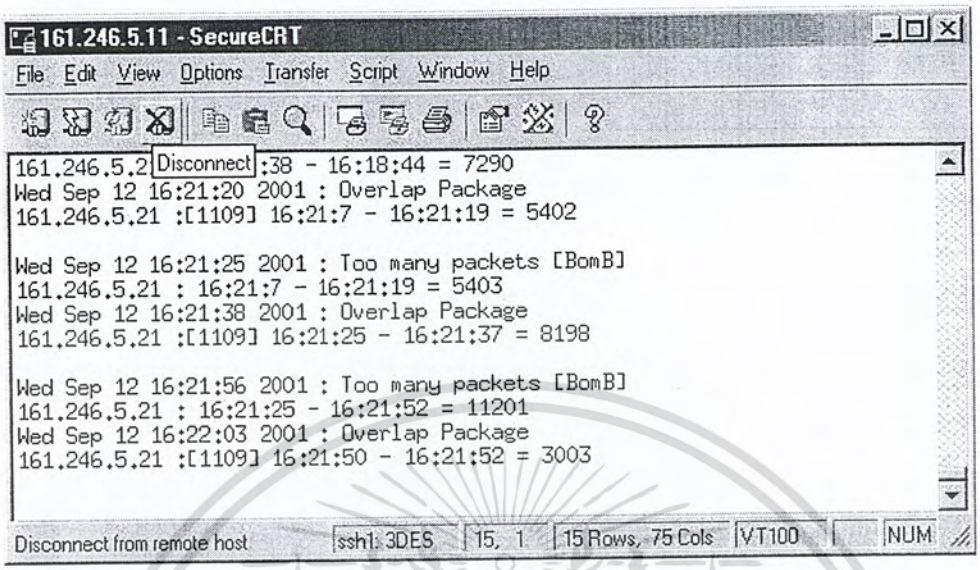
ซึ่งแบ่งออกได้ดังต่อไปนี้

- การส่งแพ็กเก็ตที่มีลำดับผิดปกติ



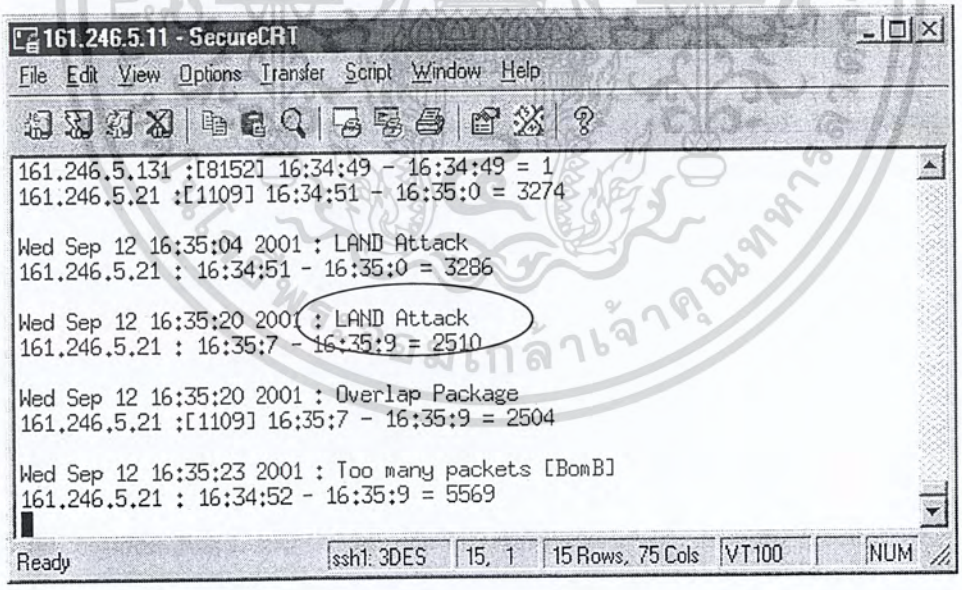
รูปที่ 6-6 แสดงการแจ้งเตือนเมื่อไม่สามารถประกอบแพ็กเก็ตได้

- การส่งแพ็กเก็ตที่มีขนาดเหลื่อมล้ำกัน



รูปที่ 6-7 แสดงการแจ้งเตือนเมื่อมีการเหลื่อมล้ำกันของแพ็กเก็ต

- การส่งแพ็กเก็ตแบบวนลูป



รูปที่ 6-8 แสดงการแจ้งเตือนเมื่อมีการส่งแพ็กเก็ตแบบวนลูป

6.3 การแสดงผลผ่านทางเว็บ

นอกจากการแสดงผลทางหน้าจอ และเก็บทางล็อกไฟล์ที่กำหนดแล้ว ระบบยังสามารถแสดงผลผ่านทางเว็บไซต์ได้ โดยจะแสดงผลของการวิเคราะห์ที่สรุปว่าเป็นการโจมตี เพื่อเป็นการแจ้งเตือนต่อผู้ดูแลระบบ

ในหน้าจอการแสดงผล จะแสดงข้อมูลดังนี้คือ

Date : วันที่ และ เวลาที่เกิดการโจมตี

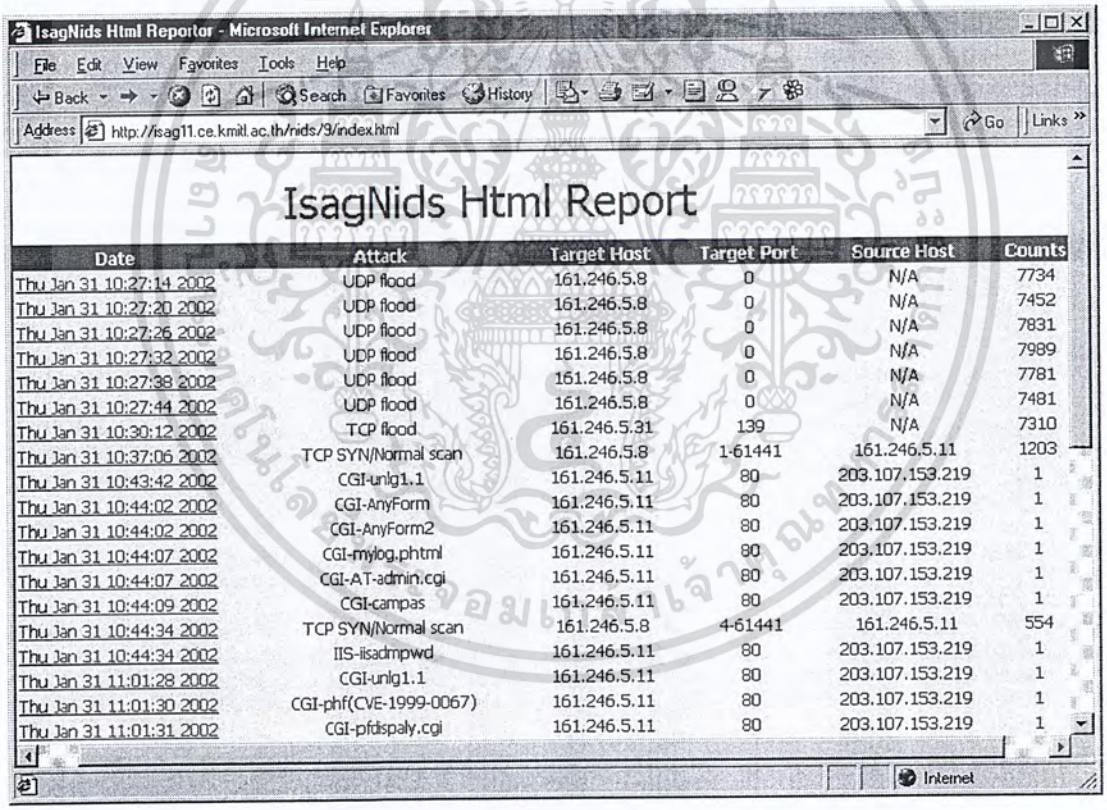
Attack : ประเภทของการโจมตีที่เกิดขึ้น

Target Host : ไอพีของเครื่องคอมพิวเตอร์ที่เป็นเป้าหมายในการโจมตี

Target Port : หมายเลขพอร์ตที่เป็นเป้าหมาย ถ้าเป็นการสแกน จะได้ค่าเป็นช่วงของพอร์ต

Source Host : ไอพีของเครื่องคอมพิวเตอร์ต้นทางที่ส่งแพ็กเก็ต

Count : จำนวนแพ็กเก็ตการโจมตี



รูปที่6-9 เว็บแสดงผลการแจ้งเตือนเมื่อมีการโจมตีจากข้อมูลที่วิเคราะห์ได้

โดยในคอลัมน์ Date ที่แสดงค่าวันที่และเวลา จะเป็นลิงก์เชื่อมต่อไปยังรายละเอียดเพิ่มเติมของการโจมตีนั้นๆ พร้อมคำอธิบายเกี่ยวกับประเภทของการโจมตีแบบสั้นๆ

IsagNids Html Report

Detection Class Information

Name	Overlap Package Detection Class
Description	This Class is used to detect IPdefragment attack

Report Information

Quick Description	Overlap Package
Date	Tue Jan 29 21:41:32 2002
Time	21:41:19 - 21:41:31
Frame Id	1109
Received	11657 times
Detail	someone send fragment packet that maybe have same fragment or overlap fragment. This could be issued in order to make you to handle this packet in an unpredictable or as a DoS attack.

Host Information

Target Host	161.246.5.17
Target Port	22
Source Host	currently N/A
Source Port	currently N/A

รูปที่6-10 แสดงรายละเอียดของการโจมตี

ตัวอย่างแสดงการรายงานการตรวจพบการโจมตีแบบต่างๆ ทางเว็บไซต์

Thu Jan 31 13:31:58 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1547
Thu Jan 31 13:34:50 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1547
Thu Jan 31 13:36:18 2002	TCP XMAS scan	161.246.5.15	1-512	161.246.5.11	511
Thu Jan 31 13:36:26 2002	TCP SYN/Normal scan	161.246.5.8	1-1023	161.246.5.11	906
Thu Jan 31 13:37:37 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1513
Thu Jan 31 13:38:17 2002	TCP SYN/Normal scan	161.246.5.8	1-1023	161.246.5.11	697
Thu Jan 31 13:39:37 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1547
Thu Jan 31 13:41:13 2002	TCP SYN/Normal scan	161.246.5.8	1-1023	161.246.5.11	832
Thu Jan 31 13:42:41 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1548
Thu Jan 31 13:43:53 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1548
Thu Jan 31 13:45:38 2002	TCP SYN/Normal scan	161.246.5.8	1-1022	161.246.5.11	872
Thu Jan 31 13:48:42 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1547
Thu Jan 31 13:49:30 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1548
Thu Jan 31 13:51:54 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1548
Thu Jan 31 13:53:30 2002	TCP SYN/Normal scan	161.246.5.8	2-1023	161.246.5.11	821
Thu Jan 31 14:06:55 2002	Ping flood	161.246.5.20	0	N/A	6253
Thu Jan 31 14:07:02 2002	Overlap Package	161.246.5.20	48274	N/A	9261

รูปที่ 6-11 แสดงการตรวจพบการสแกนพอร์ต

Thu Jan 31 13:51:54 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1548
Thu Jan 31 13:53:30 2002	TCP SYN/Normal scan	161.246.5.8	2-1023	161.246.5.11	821
Thu Jan 31 14:06:55 2002	Ping flood	161.246.5.20	0	N/A	6253
Thu Jan 31 14:07:02 2002	Overlap Package	161.246.5.20	48274	N/A	9261
Thu Jan 31 14:07:34 2002	Ping Sweep	161.246.5.20 - 161.246.5.33	N/A	161.246.5.11	9268
Thu Jan 31 14:07:34 2002	TCP SYN/Normal scan	161.246.5.20	1-65301	161.246.5.11	1112
Thu Jan 31 14:08:30 2002	Ping Sweep	161.246.5.0 - 161.246.5.205	N/A	161.246.5.11	59
Thu Jan 31 14:08:54 2002	TCP SYN/Normal scan	161.246.5.8	1-65301	161.246.5.11	1548
Thu Jan 31 14:10:30 2002	TCP SYN/Normal scan	161.246.5.8	1-1023	161.246.5.11	852

รูป 6-12 แสดงการตรวจพบแพ็กเก็ตที่โอเวอร์แลป และ การ Ping Sweep

Thu Jan 31 07:08:19 2002	TCP flood	161.246.5.8	139	N/A	6000
Thu Jan 31 07:09:19 2002	TCP flood	161.246.5.8	139	N/A	7211
Thu Jan 31 07:10:34 2002	TCP flood	161.246.5.8	139	N/A	7197
Thu Jan 31 07:12:18 2002	TCP flood	161.246.5.8	139	N/A	6925
Thu Jan 31 07:14:00 2002	TCP flood	161.246.5.33	139	N/A	6811
Thu Jan 31 07:14:06 2002	TCP flood	161.246.5.33	139	N/A	7586
Thu Jan 31 07:14:12 2002	TCP flood	161.246.5.33	139	N/A	7285
Thu Jan 31 07:14:18 2002	TCP flood	161.246.5.33	139	N/A	7385
Thu Jan 31 10:25:43 2002	Ping flood	161.246.5.8	0	N/A	6000
Thu Jan 31 10:25:49 2002	Ping flood	161.246.5.8	0	N/A	6000
Thu Jan 31 10:25:50 2002	Overlap Package	161.246.5.8	22	N/A	12797
Thu Jan 31 10:25:55 2002	Ping flood	161.246.5.8	0	N/A	6851

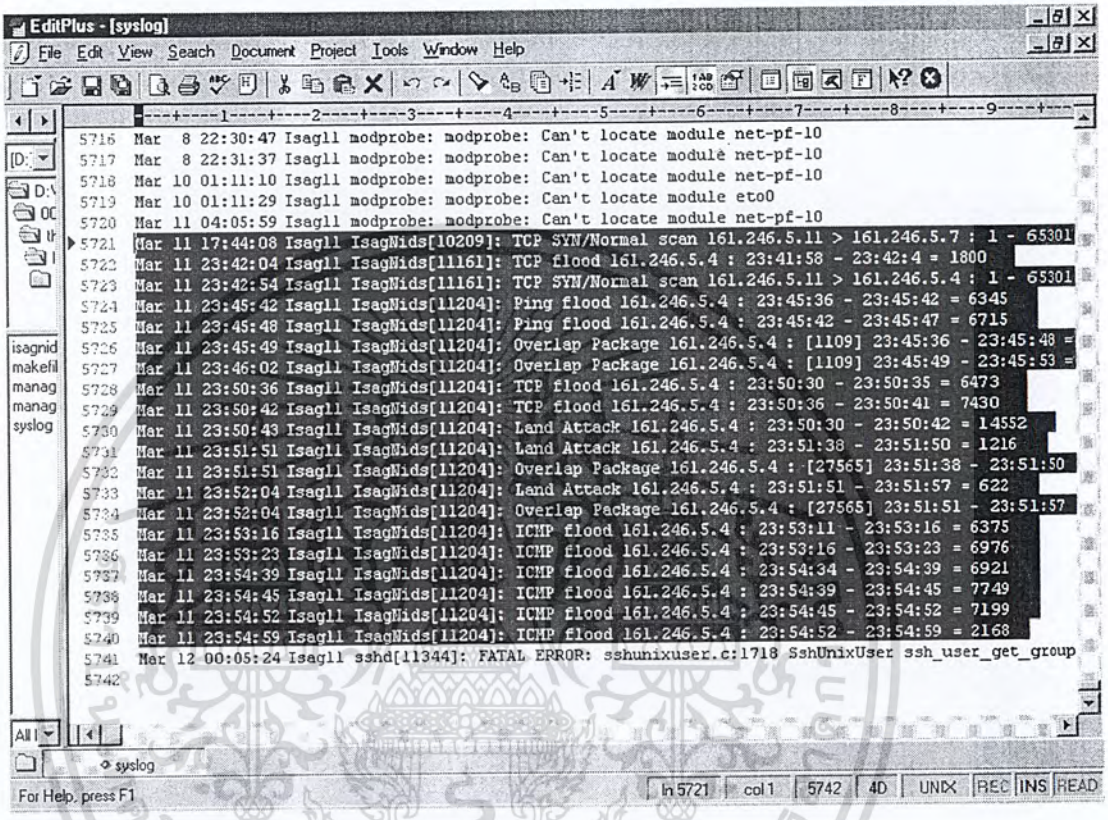
รูป 6-13 แสดงการตรวจพบการโจมตีด้วยแพ็กเก็ตจำนวนมาก

Thu Jan 31 11:01:31 2002	CGI-pfdispaly.cgi	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:32 2002	CGI-aglimpse	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:32 2002	CGI-AnyForm	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:32 2002	CGI-AnyForm2	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:36 2002	CGI-formmail	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:37 2002	CGI-formmail	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:38 2002	CGI-mlog.phtml	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:48 2002	CGI-mylog.phtml	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:48 2002	CGI-args.bat	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:49 2002	CGI-AT-admin.cgi	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:49 2002	CGI-bnbform.cgi	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:50 2002	CGI-campas	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:01:51 2002	IIS-carbo.dll	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:02:06 2002	IIS-iisadmpwd	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:02:15 2002	CGI-classifieds.cgi	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:02:24 2002	CGI-environ.cgi	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:02:25 2002	CGI-faxsurvey	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:02:25 2002	CGI-filemail.pl	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:02:29 2002	CGI-files.pl	161.246.5.11	80	203.107.153.219
Thu Jan 31 11:02:33 2002	CGI-GuestBook(CVE-1999-0237)	161.246.5.11	80	203.107.153.219

รูป 6-14 แสดงการตรวจพบการโจมตีบนชั้น แอพพลิเคชัน

6.4 การเก็บผลการวิเคราะห์ลง Syslog

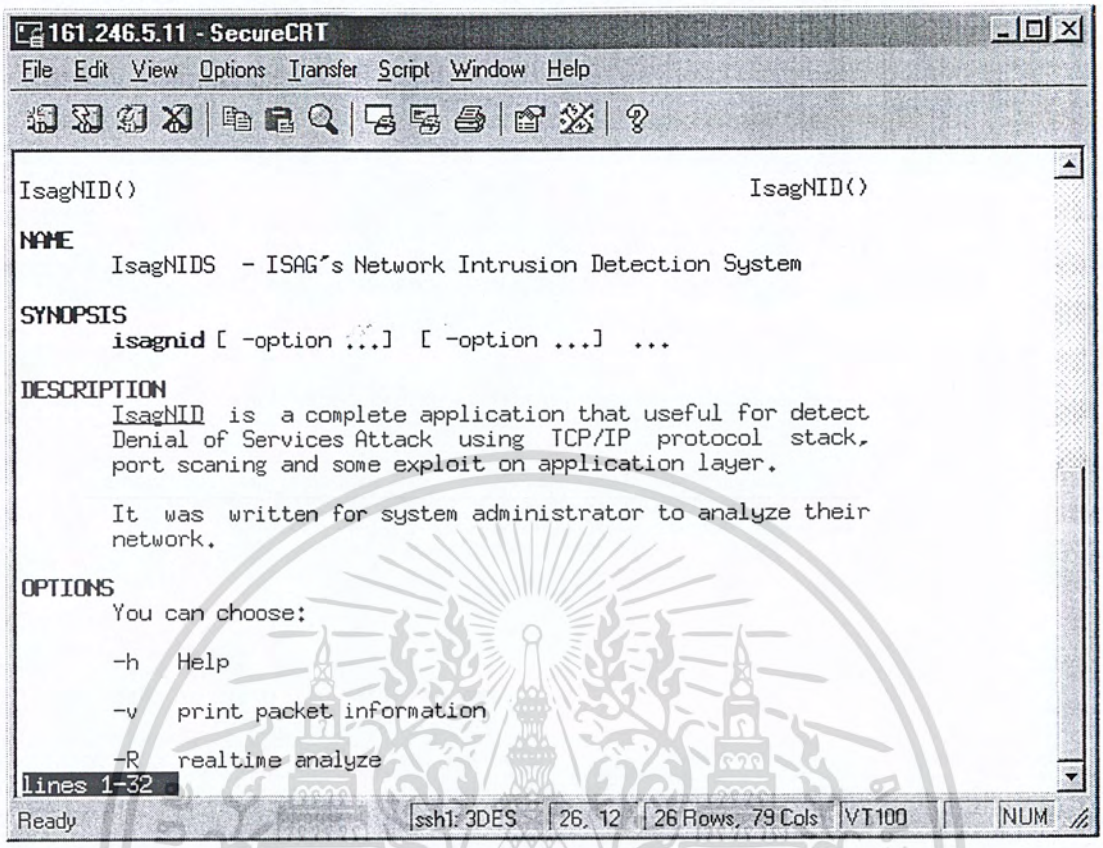
เมื่อมีการบุกรุกเกิดขึ้น และในไฟล์คอนฟิกูเรชันได้มีการสั่งให้ทำการเก็บลงในไฟล์ syslog ด้วย ในไฟล์จะมีการแสดงผลดังนี้



รูปที่ 6-15 แสดงข้อมูลการวิเคราะห์พบการบุกรุกที่เก็บในไฟล์ syslog

6.5 ส่วนแสดงความช่วยเหลือสำหรับระบบ

ได้มีการจัดทำการช่วยเหลือสำหรับระบบนี้ไว้ โดยสามารถเรียกดูได้จาก คำสั่ง man(manual) โดยเรียก **man IsagNids**



รูปที่ 6-16 แสดงการเรียกคำสั่ง man IsagNids

บทที่ 7
สรุปและวิจารณ์

7.1 ปัญหาและอุปสรรค

ในการพัฒนาระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์นี้ มีปัญหาและอุปสรรคในการพัฒนาหลายประการ ได้แก่

(1) ยังไม่สามารถตรวจสอบแพ็กเก็ตการโจมตีที่มีการแฟรกเมนต์ที่เฮดเดอร์ของแพ็กเก็ตได้

(2) ปัญหาเรื่องการติดต่อกันระหว่างเธรดต้องมีการซิงโครไนซ์ที่ดี มิฉะนั้นจะเกิดปัญหาได้

(3) ความไม่เข้ากันระหว่างภาษา C และ C++ เนื่องจากระบบพัฒนาโดยใช้ภาษา C++ แต่จำเป็นต้องใช้ไลบรารีบางตัวที่ต้องใช้ภาษาC ทำให้เกิดปัญหาในการทำงานได้ เช่น ใช้เธรดของ C และเมื่อจะมาใช้ IOStream ในเธรดจะใช้ไม่ได้ ทำให้เธรดหยุดการทำงาน

(4) การตรวจสอบข้อมูลในชั้นแอพพลิเคชันด้วยทำให้การทำงานของโปรแกรมช้า จำเป็นต้องมีอัลกอริทึมในการจัดการข้อมูล และค้นหาที่ดีพอ

(5) ความไม่สมบูรณ์ของไลบรารีในลีนุกซ์ เช่น บางฟังก์ชันระบุว่าสามารถทำงานอย่างหนึ่งได้ แต่เมื่อได้ลองใช้จริงแล้วทำไม่ได้

7.2 แนวทางการวิจัยและพัฒนาต่อ

เนื่องจากระบบตรวจจับผู้บุกรุกทางเครือข่ายคอมพิวเตอร์ที่พัฒนาขึ้นมานี้ สามารถตรวจจับผู้บุกรุกได้แค่ระดับที่ทราบว่ามีการโจมตีระบบ โดยการส่งแพ็กเก็ตในชั้นอินเทอร์เน็ตและทรานสปอร์ต และเป็นระบบที่สร้างขึ้นเพื่อใช้งานบนระบบปฏิบัติการลีนุกซ์เท่านั้น ดังนั้นจึงสามารถนำไปพัฒนาต่อได้โดย

(1) สร้างระบบป้องกันขึ้นด้วย กล่าวคือ เมื่อมีการแจ้งเตือนสามารถกันไม่ได้รับแพ็กเก็ตที่มาจากไอพีต้นทาง (Source IP) หรือแม็คแอดเดรสต้นทาง (Source MAC) นั้นได้ โดยให้ไปตั้งค่าที่ไฟร์วอลล์ (Firewall) หรือสวิตช์ชิ่ง (Switching) ให้โดยอัตโนมัติ

(2) ทำให้สามารถตรวจสอบได้ว่าผู้บุกรุกเป็นใคร โดยเมื่อมีความผิดปกติเกิดขึ้นให้ Trace กลับไปยังเครื่องที่โจมตีมาได้

(3) เพิ่มความสามารถในการตรวจจับแพ็กเก็ตการโจมตีที่ผ่านการดีแฟรกเมนต์มาได้

(4) เพิ่มความสามารถในการตรวจจับให้มากขึ้นได้ โดยการเพิ่มขอบเขตการตรวจจับให้กว้างขึ้น (ทำเป็นแบบกระจาย (Distributed))

(5) เพิ่มรูปแบบของตัวซิกเนเจอร์ให้มีความซับซ้อนยิ่งขึ้นเพื่อความละเอียดในการตรวจจับ (เทียบเคียงกับ snort rule base)

บรรณานุกรม

หนังสืออ้างอิง

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

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

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

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

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

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

[7] เอกสิทธิ์ วิริยจารี, *"ปิดทางแฮกเกอร์"*, ซีเอ็ดยูเคชั่น, 2544

เว็บไซต์อ้างอิง

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

[2] http://www.robertgraham.com/pubs/network-intrusion-detection.html

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