การพัฒนาโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

Development of TCP Port Scanning Detection Program

โดย

นายชูศักดิ์ บุญญศิริวัฒน์

รหัส 42067095

อาจารย์ที่ปรึกษา

อ. อัครินทร์ คุณกิตติ

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2543

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การพัฒนาโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต
นักศึกษา	นายชูศักดิ์ บุญญศิริวัฒน์
อาจารย์ที่ปรึกษา	อ.อัครินทร์ คุณกิตติ
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2543

บทคัดย่อ

ความปลอดภัยนับเป็นปัจจัยสำคัญอย่างหนึ่งในการใช้คอมพิวเตอร์ที่มีการเชื่อมต่อเป็นเครือข่ายอย่างในปัจจุบันซึ่งมีผู้ต้องการโจมตีเครื่องคอมพิวเตอร์ผ่านทางเครือข่ายมากขึ้น การกวาดดูทีซีพีพอร์ตเป็นวิธีหนึ่งที่ผู้บุกรุกมักใช้หาเซอร์วิสที่เปิดให้บริการในแต่ละโฮสต์และพยายามที่จะบุกรุกเข้าโฮสต์ที่เป็นเหยื่อต่อไป ดังนั้นโครงการพัฒนาระบบงานนี้จึงออกแบบและพัฒนาโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตขึ้นบนระบบปฏิบัติการลีนุกซ์ สำหรับตรวจจับการบุกรุกสถานีงานในเครือข่ายคอมพิวเตอร์ โดยอาศัยการวิเคราะห์รูปแบบแพ็กเก็ตที่รับมาจากเครือข่าย ผลที่ได้จากการวิเคราะห์จะนำมาเปรียบเทียบกับรูปแบบของการบุกรุกเครื่องคอมพิวเตอร์ที่มักตรวจพบกันได้โดยทั่วไป หลังจากนั้นจึงรายงานสถานะให้ผู้ดูแลระบบทราบเพื่อเป็นข้อมูลสำคัญสำหรับใช้ในการป้องกันและรักษาความปลอดภัยในระบบต่อไป

Title	Development of TCP Port Scanning Detection Program
Student	Mr. Choosak Bunyasiriwat
Advisor	Mr. Akharin Khunkitti
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2000

Abstract

Security is one of the most important topics when using computer on the network because there are many attacks via computer network nowadays. TCP Port Scanning is one of attacks that some intruders always use for finding opened services in each hosts and try to intrude that victim hosts later. Thus, this project provides a development of TCP Port Scanning Detection Program on Linux operating system for host-based intrusion detection in computer network. Packet flows to the host are fed to the capture engine, then the results are analyzed based on the characteristics of TCP Port Scanning. If TCP Port Scanning attacks can be detected, it will send mail to system administrator specified in program's configuration file in order to notify system administrator find the best solution to protect that system's security.

กิตติกรรมประกาศ

โครงการพัฒนาระบบงานนี้สำเร็จได้ด้วยดี เนื่องจากได้รับคำแนะนำ คำตักเตือน ทั้งหลายจากอาจารย์อัครินทร์ คุณกิตติ อาจารย์ที่ปรึกษาโครงการพัฒนาระบบงาน ซึ่งขอขอบพระคุณเป็นอย่างสูง รวมทั้งอาจารย์ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ให้การอบรมสั่งสอนวิชาความรู้แก่ข้าพเจ้ามาโดยตลอด

ขอขอบคุณ อุรุพงษ์ กัลยาสิริ, วิภา จรัสมานะโชติ, น้องเหลิม และน้องอาร์ท ที่คอยให้ความช่วยเหลือและคำแนะนำต่างๆในการทำงานตลอดเวลา

ที่สำคัญและขาดมิได้ คือ กำลังใจที่ได้รับจากบุคคลต่อไปนี้เสมอมา ได้แก่ มารดา พี่ชาย น้องสาว สาลี่ และบุคคลรอบข้างที่คอยเป็นห่วง

ชูศักดิ์ บุญญศิริวัฒน์

5 มีนาคม 2544

สารบัญ

หน้า

สารบัญภาพ

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ปัจจุบันเมื่อเครือข่ายคอมพิวเตอร์เข้ามามีบทบาทในชีวิตประจำวันมากขึ้น ความต้องการในการนำคอมพิวเตอร์ไปใช้ในการติดต่อสื่อสารหรือเชื่อมต่อเพื่อให้บริการแก่กันก็ย่อมมากขึ้นตามไปด้วย ในขณะเดียวกันก็ย่อมต้องมีผู้ไม่ประสงค์ดีต้องการที่จะบุกรุกเข้าระบบโดยผ่านบริการ(services)ที่เปิดอยู่

นับวันผู้บุกรุกทางเครือข่ายคอมพิวเตอร์จะมีจำนวนมากขึ้นและมีรูปแบบของการโจมตีที่หลากหลายขึ้น การโจมตีรูปแบบหนึ่งที่ผู้บุกรุกมักใช้กัน คือ การกวาดดูทีซีพีพอร์ต(TCP Port Scanning) ซึ่งเป็นการหาเซอร์วิสที่เปิดให้บริการในแต่ละโฮสต์ เมื่อผู้บุกรุกรู้ว่าเซอร์วิสที่เปิดบริการของโฮสต์นั้นๆ ก็จะพยายามบุกรุกเข้าระบบต่อไป

ดังนั้นจึงต้องมีการตรวจจับการโจมตีจากผู้บุกรุกเหล่านี้ จึงเกิดแนวคิดของการทำโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต(TCP Port Scanning Detection Program)ขึ้น เพื่อตรวจสอบการบุกรุกทางเครือข่ายคอมพิวเตอร์ที่เกิดขึ้นรวมทั้งแจ้งเตือนไปยังผู้ดูแลระบบและเก็บข้อมูลไว้ในล็อกไฟล์ เพื่อใช้ในการตรวจสอบได้ภายหลัง

โครงการพัฒนาระบบงานนี้จึงมุ่งเน้นการศึกษารูปแบบการโจมตีโดยการกวาดดูทีซีพีพอร์ต เพื่อพัฒนาโปรแกรมตรวจจับบนระบบปฏิบัติการลีนุกซ์ให้สามารถตรวจจับการโจมตีดังกล่าวได้อย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของโครงการพัฒนาระบบงาน

1.2.1 เพื่อศึกษารายละเอียดและการทำงานของการโจมตีโดยการกวาดดูทีซีพีพอร์ตสำหรับโปรโตคอลทีซีพี/ไอพี

1.2.2 เพื่อศึกษาแนวทางการตรวจจับการโจมตีโดยการกวาดดูทีซีพีพอร์ต

1.2.3 เพื่อสร้างโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

1.3 ขอบเขตของโครงการพัฒนาระบบงาน

1.3.1 ในงานวิจัยนี้ได้พัฒนาโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตสำหรับระบบปฏิบัติการ RedHat 7.0 เขียนด้วยภาษาซี(C Language) ถ้านำไปใช้กับระบบปฏิบัติการอื่นอาจมีปัญหาได้

1.3.2 เครื่องที่สามารถตรวจจับการกวาดดูทีซีพีพอร์ตต้องมีโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตทำงานอยู่(host-based intrusion detection program)

1.3.3 สามารถกำหนดค่าบางอย่างเกี่ยวกับ โปรแกรมได้จากแฟ้มคอนฟิกูเรชัน (configuration file) เพื่อให้เหมาะสมกับสภาพแวดล้อมที่โปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตทำงานอยู่

1.3.4 เมื่อโปรแกรมตรวจพบว่ามีการกวาดดูทีซีพีพอร์ตเกิดขึ้น จะทำการเก็บลงแฟ้มล็อก (log file) และส่งเมล์แจ้งไปยังผู้ดูแลระบบด้วย

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษารายละเอียดเกี่ยวกับโปรโตคอลทีซีพี/ไอพีเบื้องต้น

1.4.2 ศึกษาเกี่ยวกับระบบตรวจจับการบุกรุกทางเครือข่ายคอมพิวเตอร์

1.4.3 ศึกษารายละเอียดและการทำงานของการโจมตีโดยการกวาดดูทีซีพีพอร์ตสำหรับโปรโตคอลทีซีพี/ไอพี

1.4.4 กำหนดรูปแบบของการโจมตีโดยการกวาดดูทีซีพีพอร์ต

1.4.5 ออกแบบโครงสร้างของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

1.4.6 ศึกษาการเขียนโปรแกรมผ่านทางเครือข่าย

1.4.7 ออกแบบขั้นตอนการทำงานของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

1.4.8 พัฒนาโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

1.4.9 ทดสอบและปรับปรุงโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 เข้าใจการทำงานของโปรโตคอลทีซีพี/ไอพีได้ดีขึ้น

1.5.2 เข้าใจการทำงานของระบบตรวจจับการบุกรุกทางเครือข่ายคอมพิวเตอร์

1.5.3 เข้าใจการทำงานของการกวาดดูทีซีพีพอร์ต

1.5.4 สามารถนำความรู้ที่ได้จากการศึกษามาสร้างโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตได้จริง

บทที่ 2
ทฤษฎีและหลักการเบื้องต้นที่เกี่ยวกับการกวาดดูทีซีพีพอร์ต

2.1 โปรโตคอลทีซีพี/ไอพี

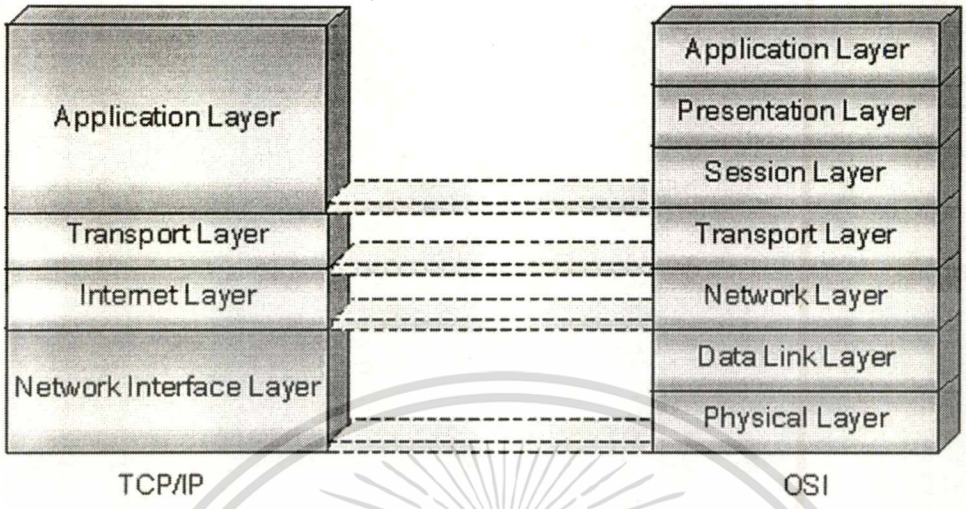
2.1.1 ความเป็นมาของโปรโตคอลทีซีพี/ไอพี

เป็นโปรโตคอลมาตรฐานที่ใช้กันอยู่ในระบบปฏิบัติการยูนิกซ์ เริ่มพัฒนาโดยกระทรวงกลาโหมของสหรัฐใน ค.ศ. 1969 เพื่อเชื่อมต่อเครื่องคอมพิวเตอร์หลายชนิดที่อยู่ห่างไกลกัน เครือข่ายที่จัดตั้งในระยะแรกชื่อว่า อาร์พาเน็ต(ARPANET)

ต่อมาได้พัฒนาเป็นเครือข่ายอินเตอร์เน็ต โปรโตคอลนี้เหมาะสำหรับเชื่อมต่อคอมพิวเตอร์ทั้งใกล้และไกลเข้าด้วยกัน และมีมาตรฐานรองรับทำให้ผู้ผลิตฮาร์ดแวร์และซอฟท์แวร์สามารถสร้างอุปกรณ์และโปรแกรมที่จะรองรับการทำงานของโปรโตคอลนี้ ทำให้เครื่องคอมพิวเตอร์สามารถรับส่งข้อมูลกันได้ไม่ว่าจะเป็นเครื่องขนาดเล็กหรือขนาดใหญ่ หรือใช้ระบบปฏิบัติการอะไรก็ตาม ทีซีพี/ไอพี(TCP/IP) เป็นชุดโปรโตคอลที่ประกอบด้วยโปรโตคอลต่างๆหลายโปรโตคอล แต่ละโปรโตคอลมีคุณลักษณะและมีความสามารถในการทำงานแตกต่างกัน โดยที่ในบทนี้ได้กล่าวถึงรายละเอียดและคุณสมบัติของโปรโตคอลที่สำคัญบางโปรโตคอล ได้แก่ โปรโตคอลทีซีพี(TCP Protocol) และ โปรโตคอลไอพี(IP Protocol)

2.1.2 การเชื่อมต่อของโปรโตคอลทีซีพี/ไอพี(TCP/IP Linking)

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



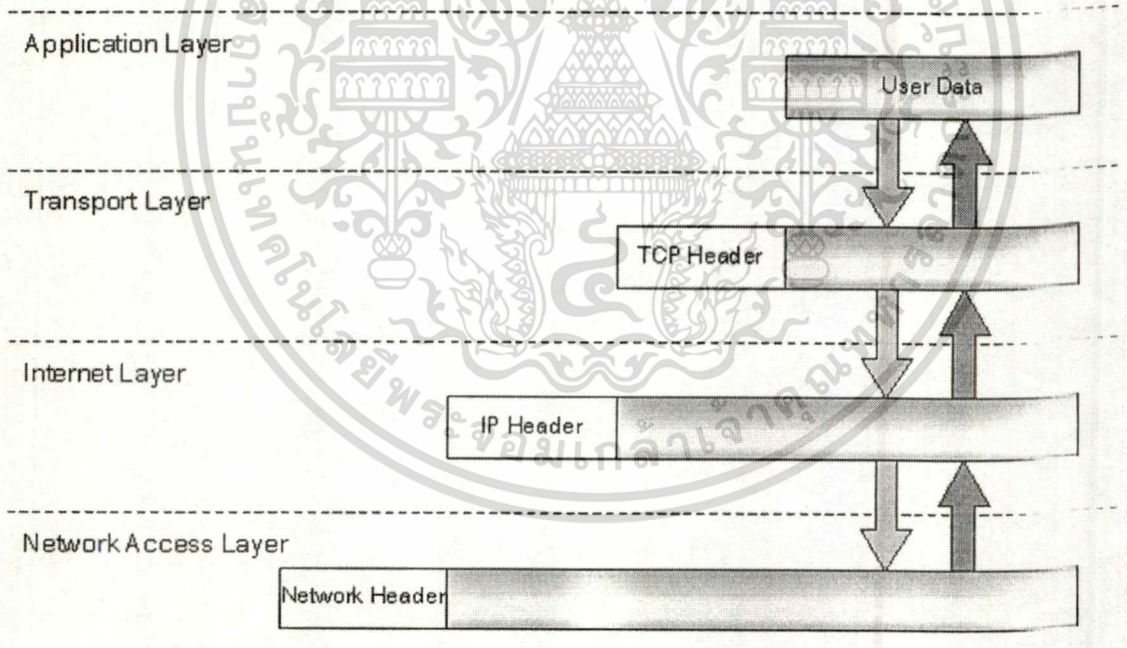
รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของทีซีพี/ไอพีกับเลเยอร์ของโอเอสไอ

ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชั่นจนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังตารางที่ 2-1

ชื่อระดับชั้น	หน้าที่
1. ชั้นแอพพลิเคชั่น (Application Layer)	ชั้นนี้รองรับการทำงานของแอพพลิเคชั่นต่างๆ ที่ทำงานเป็นโปรเซสอยู่ในเครื่องต้นทางและปลายทาง โดยจัดการเชื่อมต่อระหว่างโปรเซสหรือแอพพลิเคชั่นที่อยู่ต่างเครื่องกัน โดยการทำงานของแอพพลิเคชั่นต่างๆมีการติดต่อกันตามแต่ละโปรโตคอลเฉพาะแล้วแต่แอพพลิเคชั่นที่ใช้งาน ซึ่งจะถูกขอบริการจากชั้นทรานสปอร์ตอีกทีหนึ่ง
2. ชั้นทรานสปอร์ต (Transport Layer)	ชั้นนี้มีการสร้างการเชื่อมต่อกันระหว่างแอพพลิเคชั่นแบบ end-to-end โดยจุดที่เชื่อมต่อกันเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ต(port) หรือซ็อกเก็ต (socket) ในชั้นนี้มีบริการหลักอยู่ 2 แบบ คือ Connection-Oriented โดยเรียกผ่านโปรโตคอลทีซีพี(TCP: Transmission Control Protocol) และ Connectionless ซึ่งเรียกผ่านโปรโตคอลยูดีพี(UDP: User Datagram Protocol) ซึ่งจะกล่าวในหัวข้อถัดไป

3. ชั้นอินเตอร์เน็ต (Internet Layer)	ชั้นนี้มีหน้าที่ส่งผ่านข้อมูลระหว่างเครือข่าย โดยมีโปรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆ ในอินเตอร์เน็ต คือ โปรโตคอลไอพี(IP: Internet Protocol) ซึ่งกล่าวถึงในหัวข้อถัดไป นอกจากนี้ในชั้นนี้ยังมีโปรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ โปรโตคอลไอซีเอ็มพี(ICMP: Internet Control Message Protocol) และโปรโตคอลเออาร์พี(ARP: Address Resolution Protocol)
4. ชั้นเน็ตเวิร์กอินเตอร์เฟส (Network Interface Layer)	ชั้นนี้มีหน้าที่ในการแปลงข้อมูลให้อยู่ในรูปที่เหมาะสมกับเครือข่ายแต่ละแบบ ซึ่งแตกต่างกันออกไป และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่าย

ตารางที่ 2-1 การทำงานของแต่ละระดับชั้นของทีซีพี/ไอพี



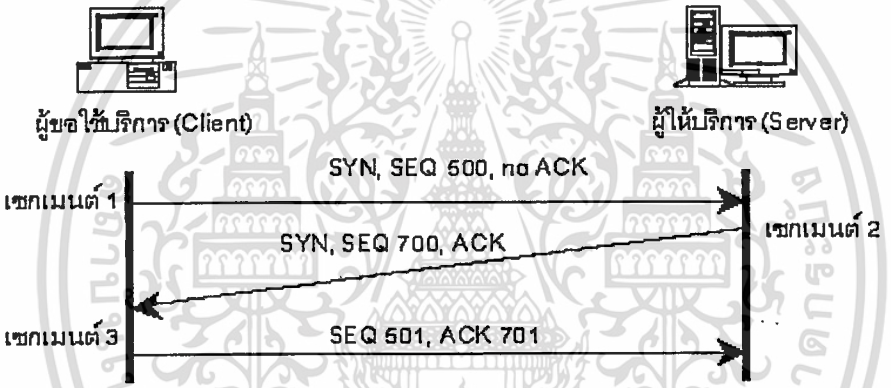
รูปที่ 2-2 แสดงการส่งผ่านข้อมูลในโมเดลของทีซีพี/ไอพี

ในชุดโปรโตคอลทีซีพี/ไอพีนี้ มีโปรโตคอลหลักที่ขอกล่าวถึง 3 โปรโตคอล ได้แก่ โปรโตคอลทีซีพี โปรโตคอลยูดีพี ซึ่งทำงานในชั้นทรานสปอร์ต และโปรโตคอลไอพี ซึ่งทำงานในชั้นอินเตอร์เน็ต โดยมีรายละเอียดดังต่อไปนี้

2.1.3 โปรโตคอลทีซีพี (Transmission Control Protocol)

การทำงานที่สำคัญอย่างหนึ่งของโปรโตคอลทีซีพี คือ การทำ "three-way handshake" ซึ่งเป็นกระบวนการเริ่มต้นในการสร้างการเชื่อมต่อในชั้นทรานสปอร์ต กล่าวคือ ทีซีพี(TCP) ในการติดต่อกันระหว่างระบบในเครือข่ายต้องมีการสร้างการเชื่อมต่อไปยังระบบที่ให้บริการก่อน ประกอบด้วย 3 ขั้นตอน ดังต่อไปนี้

1. ผู้ขอบริการส่งสัญญาณ SYN เพื่อขอบริการ
2. จากนั้นผู้ให้บริการจะส่งสัญญาณ SYN&ACK เพื่อตอบรับการเชื่อมต่อที่ร้องขอมา
3. ผู้ใช้บริการตอบรับการเชื่อมต่อโดยส่ง ACK หลังจากนั้นการเชื่อมต่อจึงถือว่าเสร็จสมบูรณ์และสามารถรับส่งข้อมูลกันได้ ดังรูปที่ 2-3(ข้อมูลที่ใช้ติดต่อในทีซีพี เรียกว่า **เซกเมนต์(Segment) หรือแพ็กเก็ต(Packet)** ประกอบด้วยเฮดเดอร์และข้อมูลของทีซีพี)



รูปที่ 2- 3แสดงการทำ three-way handshake สำหรับสร้างการเชื่อมต่อของ TCP

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

ดังนั้นโปรโตคอลทีซีพีจึงเป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบ "Connection-Oriented" ทำให้การทำงานของทีซีพีมีความน่าเชื่อถือมากขึ้น หน้าที่การทำงานของทีซีพีในการรับส่งข้อมูลมีหน้าที่หลัก 6 ข้อ คือ

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 ขนาด 16 บิต*: เป็นหมายเลขพอร์ตของบริการที่เครื่องต้นทาง

2. *Destination Port ขนาด 16 บิต*: เป็นหมายเลขพอร์ตของบริการที่เครื่องปลายทาง

3. *Sequence Number ขนาด 32 บิต*: ทีซีพีใช้ **เลขลำดับ** เป็นตัวนับจำนวนไบต์ที่ส่งทุกครั้งที่สถาปนาการเชื่อมโยงทีซีพีจะเลือกเลขลำดับเริ่มต้นสำหรับชี้ตำแหน่งข้อมูลไบต์แรกที่จะจัดส่ง หมายเลขเริ่มต้นไม่จำเป็นต้องเริ่มด้วย 1 แต่อาจเริ่มด้วยค่าใดๆก็ได้ ข้อมูลในเซกเมนต์ถัดไปจะมีเลขลำดับที่สัมพัทธ์เลขลำดับในเซกเมนต์ก่อนหน้า

4. *Acknowledgement Number ขนาด 32 บิต*: ค่ากำหนด **เลขตอบรับ** ซึ่งใช้ตอบกลับไปว่าได้รับข้อมูลแล้ว เลขตอบรับจะมีค่าเท่ากับเลขลำดับประจำเซกเมนต์บวกด้วยจำนวนไบต์ข้อมูลและบวกด้วยหนึ่ง เช่น เซกเมนต์หนึ่งมีเลขลำดับเท่ากับ 21 และมีข้อมูล 20 ไบต์ เลขตอบรับที่ต้องส่งกลับไปจะเท่ากับ 21+20+1=42 ซึ่งแจ้งว่าได้รับข้อมูลตั้งแต่ต้นถึงไบต์ที่ 41 และคาดว่าไบต์ถัดไปคือไบต์ที่ 42

5. *Offset ขนาด 4 บิต*: เป็นตัวบอกค่าออฟเซตของข้อมูล เพราะทีซีพีนั้นไม่มีการกำหนดความยาวที่แน่นอนของข้อมูล จึงต้องมีออฟเซตเป็นตัวบอก

6. *Reserved(RSV)ขนาด 6 บิต*: สำรองไว้ใช้ในอนาคต

7. *Code ประกอบด้วย 6 ฟิล์ดย่อย แต่ละฟิล์ดย่อยมีขนาด 1 บิต* : ทำหน้าที่เป็น flag บอกชนิดของข้อมูล ได้แก่

- URG: Urgent Pointer Field Significant – แสดง Urgent Pointer ถ้าบิตนี้เป็น '1' แสดงว่าบรรจุตำแหน่งข้อมูลที่ต้องรีบดำเนินการเร่งด่วนก่อน

- ACK: Acknowledgement Field Significant – แสดงการ Acknowledgement ถ้าบิตนี้เป็น '1' แสดงว่าเป็นเซกเมนต์ตอบรับ โดยตอบอ้างอิงเลขลำดับตามที่กำหนดในฟิล์ด Acknowledgement number

- PSH: Push Function ถ้าบิตนี้เป็น '1' แสดงว่าทันทีที่สถานีปลายทางได้รับเซกเมนต์ต้องรีบส่งข้อมูลไปยังโปรโตคอลประยุกต์ทันทีโดยไม่ต้องรอให้บัฟเฟอร์เต็ม

- RST: Reset the Connection – แสดงเมื่อรีเซ็ตการเชื่อมต่อ ถ้าบิตนี้เป็น '1' แสดงถึงการยกเลิกการเชื่อมต่อนี้ เนื่องจากอาจมีความผิดปกติเกิดขึ้นระหว่างคู่สถานีที่กำลังติดต่อกันอยู่ หากจำเป็นต้องส่งข้อมูลระหว่างกันอีกก็ต้องเริ่มต้นสถาปนาการเชื่อมต่อใหม่

-SYN: Synchronize Sequence Number ถ้าบิตนี้เป็น '1' แสดงถึงขอเริ่มต้นสถาปนาการเชื่อมต่อและเมื่อการสถาปนาเสร็จสิ้น บิตนี้จะถูกกำหนดให้เป็น '0' หลังจากนั้นจึงสามารถส่งผ่านข้อมูลระหว่างกันได้

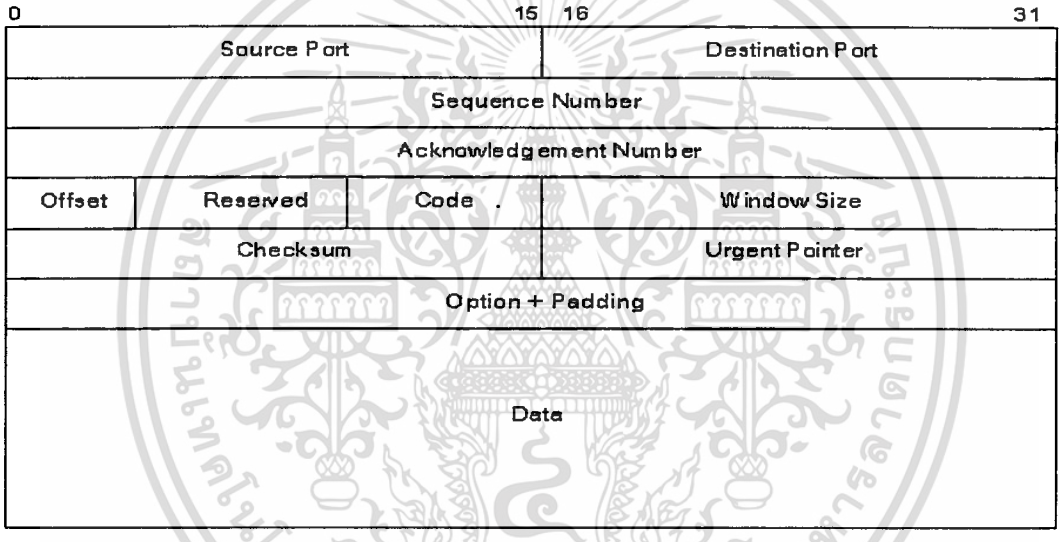
-FIN: Finish – แสดงว่าไม่มีข้อมูลที่ส่งจากผู้ส่งแล้ว จึงขอจบการเชื่อมต่อ

8. *Window Size ขนาด 16 บิต*: สถานีปลายทางใช้ฟิลด์นี้แจ้งขนาดบัฟเฟอร์ที่มีอยู่(หน่วยเป็นไบต์) สถานีที่ติดต่อด้วยต้องไม่ส่งข้อมูลเกินค่านี้

9. *Checksum ขนาด 16 บิต*: ผลรวมตรวจสอบความถูกต้องของเซกเมนต์โดยคำนวณทั้งเฮดเดอร์และข้อมูล

10. *Urgent Pointer ขนาด 16 บิต*: พอยเตอร์ชี้ตำแหน่งไบต์ข้อมูลที่ต้องดำเนินการเร่งด่วนที่ต้องการให้โปรแกรมประยุกต์ดำเนินการทันที ค่าที่บรรจุในฟิลด์นี้จะมีความหมายก็ต่อเมื่อแฟล็ก URG ถูกเซ็ตเป็น '1'

11.*Option and Padding*: เป็นตัวบอกออปชันของโปรเซสที่ใช้ทีซีพี



รูปที่ 2-4 แสดงแพ็กเก็ตทีซีพี

2.1.4 โปรโตคอลยูดีพี(UDP: User Datagram Protocol)

โปรโตคอลยูดีพีเป็นโปรโตคอลในการติดต่อสื่อสารในชั้นทรานสปอร์ต(Transport Layer) การทำงานคล้ายกับทีซีพีมาก คือ จัดการเกี่ยวกับการสื่อสารระหว่างเครื่อง แต่เป็นแบบ Connectionless คือ ทั้งฝ่ายส่งและฝ่ายรับไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน โดยไม่ต้องมีการแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโปรโตคอลทีซีพี และไม่มีการส่งสัญญาณตรวจสอบว่าข้อมูลถึงเครื่องปลายทางอย่างถูกต้องครบถ้วนในการส่งข้อมูลแต่ละครั้ง จึงไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล

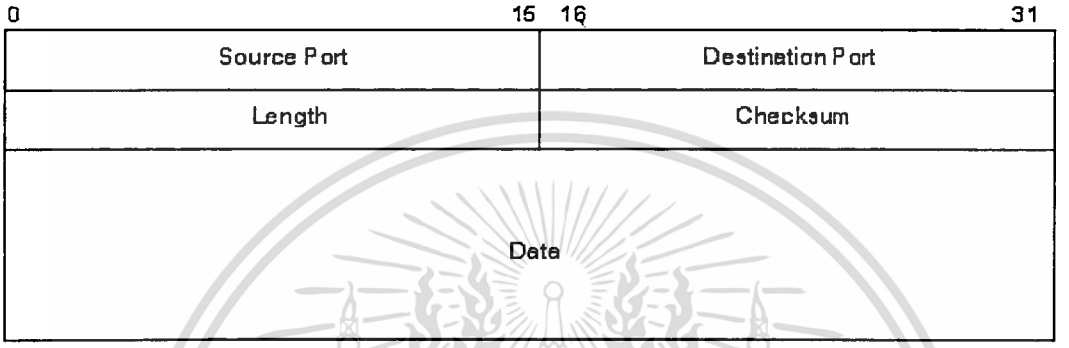
ส่วนประกอบของยูดีพีเฮดเดอร์

1. *Source Port ขนาด 16 บิต*: เป็นหมายเลขพอร์ตของบริการที่เครื่องต้นทาง

2. *Destination Port ขนาด 16 บิต*: เป็นหมายเลขพอร์ตของบริการที่เครื่องปลายทาง

3. *Length ขนาด 16 บิต*: บอกความยาวของข้อมูล

4. *Checksum ขนาด 16 บิต*: ตรวจสอบความถูกต้องของข้อมูลที่ส่ง



รูปที่ 2-5 แสดงแพ็กเก็ตยูดีพี

2.1.5 โปรโตคอลไอพี(IP: Internet Protocol)

โปรโตคอลไอพี มีหน้าที่จัดขนาดของข้อมูลให้เหมาะสมและเลือกเส้นทางที่เหมาะสมเพื่อจัดส่ง**เดทาแกรม(datagram)** ซึ่งประกอบด้วยเฮดเดอร์และข้อมูลของไอพี ไอพีมีรูปแบบการจัดส่งเดทาแกรมเป็นแบบ "*unreliable*" คือ ไอพีไม่มีกลไกรับประกันว่าเดทาแกรมที่ส่งจะไปถึงปลายทางได้สำเร็จ ไอพีให้บริการลำเลียงเดทาแกรมอย่างดีที่สุด หากมีความผิดปกติใดเกิดขึ้นระหว่างการนำส่งเดทาแกรม เช่น บัฟเฟอร์ของเราเตอร์ระหว่างทางเต็มจนไม่สามารถรับเดทาแกรมได้ สิ่งที่ไอพีดำเนินการกับเดทาแกรมคือทิ้งเดทาแกรมนั้นไป แล้วรายงานสาเหตุของปัญหากลับไปด้วยโปรโตคอลไอซีเอ็มพี และ "*connectionless*" คือ ไอพีไม่สถาปนาการเชื่อมต่อเพื่อกำหนดเส้นทางลำเลียงระหว่างต้นทางและปลายทาง ไอพีไม่เก็บสถานะใดๆของเดทาแกรมที่ส่งออกไป เดทาแกรมแต่ละชิ้นจึงเป็นอิสระจากกัน และมีโอกาสไปถึงปลายทางโดยไม่เรียงลำดับ

ส่วนประกอบของไอพีเฮดเดอร์

1. *Version ขนาด 4 บิต*: บอกเวอร์ชั่นของมาตรฐานไอพีที่ใช้ โดยปกติมีค่าเป็น 4 ซึ่งหมายถึง IPv4

2. *Internet Header Length(IHL):* เป็นตัวบอกความยาวเฮดเดอร์ของไอพี

3. *Type of Service ขนาด 8 บิต:* เป็นส่วนที่บอกการทำงานของแพ็กเก็ตที่ส่งว่าทำหน้าที่อะไร

บิตที่ 0-2: บอกรายละเอียดการทำงานของแพ็กเก็ตนั้นๆ

111-Network Control

110-Internetwork Control

101-CRITIC/ECP

100-Flash Override

011-Flash

010-Immediate

001-Priority

000-Routine

บิตที่ 3: บอกถึงลักษณะของดีเลย์

0 = Normal Delay – มีดีเลย์ปกติ

1 = Low Delay – มีดีเลย์ต่ำ

บิตที่ 4: บอกถึงประเภทของทรูพุต

0 = Normal Throughput – มีทรูพุตปกติ

1 = High Throughput – มีทรูพุตสูง

บิตที่ 5: บอกถึงประเภทของความน่าเชื่อถือ

0 = Normal Reliability – มีความน่าเชื่อถือพอประมาณ

1 = High Reliability – มีความน่าเชื่อถือสูง

บิตที่ 6-7: กันไว้ใช้ในอนาคต

4. *Total Length ขนาด 16 บิต:* บอกถึงความยาวในดาตาแกรมของไอพี

5. *Identification field ขนาด 16 บิต:* เป็นค่าประจำตัวของไอพีนั้น โดยโฮสต์ที่ส่งเป็นผู้กำหนดและเพิ่มค่าขึ้นหนึ่งเมื่อมีการส่งดาตาแกรมของไอพีใหม่ซึ่งใช้ในการประกอบกลับ

6. *Flag ขนาด 3 บิต:* เป็นตัวเลขบอกลักษณะของแพ็กเก็ตว่ามีการแฟร็กเมนต์หรือไม่

บิตที่ 0: สงวนไว้ ปกติเป็น 0

บิตที่ 1: 0 = บอกว่าแพ็กเก็ตมีการแตกแพ็กเก็ตย่อย

1 = บอกว่าแพ็กเก็ตไม่มีการแตกแพ็กเก็ตย่อย

บิตที่ 2: 0 = บอกว่าแพ็กเก็ตนั้นเป็นแพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

1 = บอกว่าแพ็กเก็ตนั้นยังไม่ใช่แพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

7. *Fragment Offset ขนาด 13 บิต:* บอกออปเซตของแฟร็กเมนต์เมื่อเทียบในดาตาแกรม

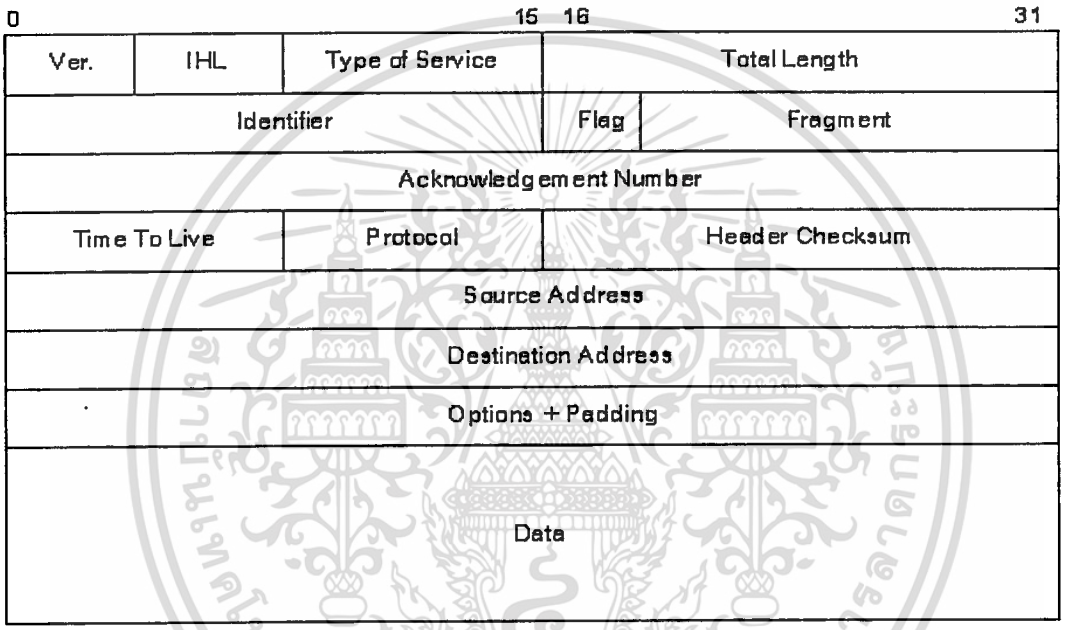
8. *Time To Live(TTL) ขนาด 8 บิต:* บอกช่วงเวลาของแพ็กเก็ตที่ยังอยู่ในเครือข่ายได้ โดยกำหนดค่าเป็นจำนวนเราเตอร์สูงสุดที่ดาตาแกรมผ่านได้ ซึ่งโดยทั่วไปมีค่าระหว่าง 32 ถึง 64 และลดค่าลงเรื่อยๆ เมื่อผ่านเราเตอร์ เพื่อเป็นการป้องกันแพ็กเก็ตล้นเครือข่าย

9. *Protocol ขนาด 8 บิต:* เป็นค่าที่บอกประเภทโปรโตคอลที่อยู่ระดับสูงขึ้นไป

10.*Header Checksum ขนาด 32 บิต:* ใช้ตรวจสอบความถูกต้องของเฮดเดอร์

11. *Source Address ขนาด 32 บิต:* บอกถึงหมายเลขไอพีของเครื่องต้นทาง

12. *Destination Address ขนาด 32 บิต:* บอกถึงหมายเลขไอพีของเครื่องปลายทาง



รูปที่ 2-6 แสดงแพ็กเก็ตไอพี

2.2 ระบบตรวจจับการบุกรุกทางเครือข่าย

2.2.1 ความหมายของระบบตรวจจับการบุกรุกทางเครือข่าย

การบุกรุก(intrusion) คือ การกระทำการใดๆที่จะทำอันตรายต่อความปลอดภัยของทรัพยากรของระบบคอมพิวเตอร์หรือระบบเครือข่ายคอมพิวเตอร์

การตรวจจับการบุกรุก(intrusion detection) คือ เทคนิคที่จะทำให้ความปลอดภัยของระบบแข็งแกร่งขึ้นและช่วยเพิ่มความทนทานต่อการโจมตีจากทั้งภายในและภายนอกระบบ

ระบบตรวจจับการบุกรุก(intrusion detection system) คือ ระบบที่ทำหน้าที่ติดตามดูการทำงานที่เกิดขึ้นบนระบบคอมพิวเตอร์หรือระบบเครือข่ายคอมพิวเตอร์ เพื่อค้นหาร่องรอยที่บอกว่า

มีผู้กำลังพยายามบุกรุกระบบหรือค้นหาการกระทำที่เกินขอบเขตสิทธิ์ของผู้ใช้ระบบ ซึ่งบางระบบเมื่อพบว่าเกิดการบุกรุกก็จะทำการตอบโต้หรือตัดสิทธิ์ผู้ที่บุกรุกระบบหรือแจ้งไปยังผู้ดูแลระบบให้ทราบว่าเกิดการบุกรุกขึ้น

2.2.2 การแบ่งแยกประเภทของระบบตรวจจับการบุกรุก

การแบ่งแยกประเภทของระบบตรวจจับการบุกรุกสามารถแบ่งได้หลายแบบจากคุณลักษณะทางด้านหน้าที่และการดำเนินงาน

2.2.2.1 แบ่งโดยวิธีการตรวจจับ(Detection Approach) แบ่งออกเป็น 2 แบบ

2.2.2.1.1 การตรวจจับการบุกรุกโดยเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก(Misuse Detection)

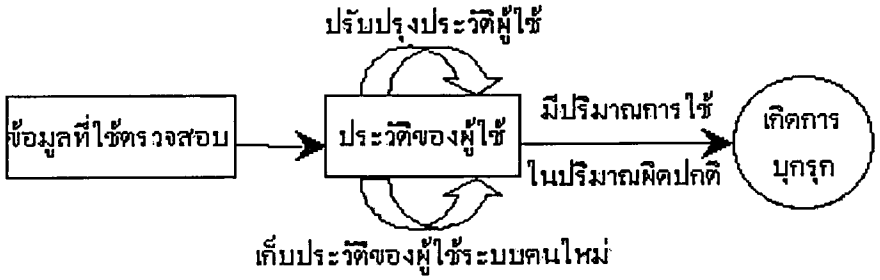
การตรวจจับแบบนี้จะดูจากรูปแบบการบุกรุกที่เคยเกิดขึ้นแล้ว โดยสามารถเปรียบเทียบได้จากรูปแบบที่แน่นอนหรือลำดับของเหตุการณ์ที่เกิดขึ้นหรือข้อมูลที่รู้จัก(signature of intrusion)

รูปที่ 2-7 แสดงการทำงานของการตรวจจับการบุกรุกโดยเปรียบเทียบพฤติกรรมผู้ใช้กับรูปแบบการบุกรุกที่รู้จัก

จากรูปที่ 2-7 ข้อมูลที่ใช้ตรวจสอบ(audit data) ถูกนำมาเปรียบเทียบกับกฎ(rules) ที่มีอยู่ ซึ่งมีการนำข้อมูลทางด้านเวลาเข้ามาพิจารณาด้วย การตรวจจับการบุกรุกโดยวิธีนี้สามารถมีการแก้ไขกฎหรือเพิ่มเติมกฎได้ ในระบบที่เป็นปัญญาประดิษฐ์(artificial intelligent) ระบบอาจทำการแก้ไขกฎหรือเพิ่มเติมกฎได้โดยอัตโนมัติ

2.2.2.1.2 การตรวจจับการบุกรุกโดยเปรียบเทียบพฤติกรรมผู้ใช้ที่ผิดปกติไปจากรูปแบบการบุกรุกที่รู้จัก(Anomaly Detection)

การตรวจจับแบบนี้จะดูจากการใช้งานทรัพยากรระบบที่ผิดปกติโดยมีสมมติฐานที่ว่า การกระทำการใดๆที่เป็นการบุกรุกจะต้องมีการใช้งานทรัพยากรระบบอย่างผิดปกติ หมายความว่า ในการตรวจจับวิธีนี้ต้องมีการเก็บพฤติกรรมปกติของผู้ใช้ระบบไว้ เพื่อเปรียบเทียบว่าพฤติกรรมใดเบี่ยงเบนไปจากพฤติกรรมปกติจะเป็นพฤติกรรมที่ผิดปกติ โดยการเปรียบเทียบจะใช้หลักทางสถิติศาสตร์เข้าช่วย



รูปที่ 2-8 แสดงการทำงานของการตรวจจับการบุกรุกโดยเปรียบเทียบพฤติกรรมผู้ใช้ที่ผิดปกติไปจากรูปแบบการบุกรุกที่รู้จัก

จากรูปที่ 2-8 ข้อมูลที่ใช้ตรวจสอบ(audit data)ซึ่งเก็บบันทึกโดยระบบจะถูกนำมาปรับปรุงไฟล์ประวัติของผู้ใช้(user profile) พร้อมกันนั้นจะนำข้อมูลนี้มาเปรียบเทียบกับสถิติการใช้งานของผู้ใช้แต่ละคน หากพบว่ามีค่าผิดไปจากค่าปกติ ก็ระบุว่าอยู่ในสถานะที่มีการบุกรุกเกิดขึ้น และถ้ามีการตรวจสอบโดยใช้ข้อมูลใหม่ๆก็สามารถเพิ่มในไฟล์ประวัติได้

2.2.2.2 แบ่งโดยปฏิกิริยาต่อการบุกรุกที่เกิดขึ้น(Reaction on intrusion) แบ่งออกเป็น 2 แบบ

2.2.2.2.1 แบบตอบโต้(Active) คือ เมื่อมีการบุกรุกเกิดขึ้น ระบบตรวจจับการบุกรุกจะกระทำการบางอย่างเพื่อตอบโต้การบุกรุกที่เกิดขึ้น เช่น การปิดช่องโหว่ที่ถูกบุกรุก การปิดบริการเซอร์วิสที่ถูกบุกรุก หรือ การเก็บล็อกรายละเอียดเกี่ยวกับผู้บุกรุก เป็นต้น

2.2.2.2.2 แบบไม่ตอบโต้(Passive) คือ เมื่อมีการบุกรุกเกิดขึ้น ระบบตรวจจับการบุกรุกจะมีสัญญาณเตือนหรือแจ้งให้ผู้ดูแลระบบทราบ

2.2.2.3 แบ่งโดยเป้าหมายของการตรวจจับการบุกรุก(Targets) แบ่งออกเป็น 3 แบบ

2.2.2.3.1 เป้าหมายเป็นแอพพลิเคชัน(Application-based)

ระบบตรวจจับการุกรุกแบบนี้จะเก็บรวบรวมข้อมูลและตรวจจับการบุกรุกในระดับของแอพพลิเคชัน ยกตัวอย่างเช่น เว็บเซิร์ฟเวอร์หรืออีคอมเมิร์ชเซิร์ฟเวอร์ เป็นต้น

2.2.2.3.2 เป้าหมายเป็นโฮสต์(Host-based)

ระบบตรวจจับการบุกรุกแบบนี้(บางครั้งเรียกว่า "เอเจนต์(agent)" หรือ "เซ็นเซอร์(sensor)") จะเก็บรวบรวมและวิเคราะห์ข้อมูลของกิจกรรมบนโฮสต์ที่ระบุในระบบ

2.2.2.3.3 เป้าหมายเป็นเครือข่าย(Network-based)

ระบบตรวจจับการบุกรุกแบบนี้จะทำงานในระดับเครือข่าย โดยเก็บรวบรวมและวิเคราะห์ข้อมูลของการสื่อสารข้อมูลในเครือข่าย

2.2.2.4 แบ่งโดยเวลาที่ใช้ในการวิเคราะห์(Analysis timing) แบ่งออกเป็น 2 แบบ

2.2.2.4.1 แบบเรียลไทม์(Real-time)

การประมวลผลในการตรวจจับการบุกรุกสามารถทำงานได้อย่างต่อเนื่อง

2.2.2.4.2 แบบเป็นระยะๆ(Periodically)

การประมวลผลในการตรวจจับการบุกรุกจะทำงานตามช่วงเวลาที่กำหนดไว้

2.2.2.5 แบ่งโดยสถาปัตยกรรม(Architecture) แบ่งออกเป็น 2 แบบ

2.2.2.5.1 แบบรวมศูนย์(Centralized)

การออกแบบระบบตรวจจับการบุกรุกแบบดั้งเดิมมักจะเป็นแบบรวมศูนย์ คือ สร้างโมดูลเพียงหนึ่งก้อน ทำหน้าที่ทุกอย่างเลย

2.2.2.5.2 แบบกระจาย(Distributed)

การออกแบบระบบตรวจจับการบุกรุกแบบกระจายจะประกอบด้วย หลายเอนติตี้(entities)ซึ่งถูกวางไว้ทั่วทั้งระบบ โดยแต่ละเอนติตี้จะทำงานได้ด้วยตัวเองอย่างอิสระต่อกัน วิธีหนึ่งที่ใช้จะสร้าง *ตัวแทนอิสระ(autonomous agents)* ซึ่งแต่ละตัวจะกระจายหน้าที่การทำงานและเป็นอิสระต่อกัน

2.2.3 ประเภทของการโจมตี

ทุกวันนี้แฮกเกอร์ใช้การโจมตีหลายประเภทโดยการโจมตีมีตั้งแต่แบบง่ายๆไปจนถึงการโจมตีที่มีความสลับซับซ้อนมาก แบ่งได้เป็น 4 ประเภท ดังตารางที่ 2-2

ประเภท	รายละเอียด
หนึ่งต่อหนึ่ง	ผู้โจมตีจะใช้เครื่องหนึ่งเครื่องโจมตีเครื่องเป้าหมายหนึ่งเครื่อง ตัวอย่างเช่น การโจมตีโดยอาศัยช่องโหว่ของโปรแกรม sendmail
หนึ่งต่อหลาย	ผู้โจมตีจะใช้เครื่องหนึ่งเครื่องโจมตีเครื่องเป้าหมายหลายเครื่อง ตัวอย่างเช่น การโจมตีเพื่อให้ระบบหยุดบริการ(denial of service attacks)
หลายต่อหนึ่ง	ผู้โจมตีจะแบ่งการโจมตีให้กับเครื่องภายนอกหลายเครื่องเพื่อที่จะทำการโจมตีเครื่องที่เป็นเหยื่อเพียงเครื่องเดียว ประเภทนี้ยากต่อการตรวจจับ เพราะว่าการเชื่อมต่อหลายอันที่เกิดขึ้นจากเครื่องต้นทางหลายเครื่องมองดูจะไร้เดียงสากว่าการเชื่อมต่อหลายอันที่เกิดจากเครื่องต้นทางเพียงเครื่องเดียว ตัวอย่างเช่น syn flood โดยใช้การหลอกไอพี(ip spoofing) เพื่อให้เซอร์วิสหยุดบริการ
หลายต่อหลาย	ผู้โจมตีหลายคนทำงานร่วมกันโดยแบ่งงานของการตรวจสอบและโจมตีเหยื่อหลายเครื่อง ประเภทนี้คล้ายประเภทหลายต่อหนึ่งโดยเพิ่มความซับซ้อนเพราะว่ามีเครื่องเป้าหมายหลายเครื่อง วิธีนี้ยากมากในการตรวจจับ ตัวอย่างเช่น การโจมตี

	โดยใช้โปรแกรม smurf เป็นการโจมตีจากเครื่องต้นทางหลายเครื่อง

ตารางที่ 2-2 แสดงประเภทของการโจมตี

2.2.4 คุณสมบัติที่ควรมีของระบบตรวจจับการบุกรุก

2.2.4.1 สามารถทำงานได้อย่างต่อเนื่อง(run continually)โดยปราศจากการดูแลโดยมนุษย์ ทั้งนี้ระบบต้องมีความเชื่อถือได้พอที่จะอนุญาตให้ระบบตรวจจับการบุกรุกทำงานในโหมดเบื้องหลัง(background)ของระบบเพื่อคอยสังเกตพฤติกรรม อย่างไรก็ตามมันไม่ควรจะเป็นกล่องดำ(black box) หมายความว่า การทำงานภายในของระบบตรวจจับการบุกรุกสามารถถูกตรวจสอบได้จากภายนอก

2.2.4.2 ทนทานต่อความผิดพลาด(fault tolerant)ในกรณีที่ระบบล่ม(system crash) เช่น อุบัติเหตุหรือสาเหตุจากกิจกรรมที่มีเจตนาร้าย เมื่อระบบเริ่มทำงานใหม่ ระบบตรวจจับการบุกรุกควรจะสามารถกู้สถานะเดิมก่อนระบบล่มกลับมาได้และทำงานต่อไปโดยไม่มีผลกระทบอะไร

2.2.4.3 ทนต่อการเปลี่ยนแปลงเวอร์ชัน(resist subversion) ระบบตรวจจับการบุกรุกควรจะสามารถตรวจดูตัวเองและตรวจสอบได้ ถ้ามีการเปลี่ยนแปลงโดยผู้โจมตี

2.2.4.4 ควรจะสร้างค่าโสหุ้ย(overhead)บนระบบให้น้อยที่สุด และไม่กระทบต่อการทำงานปกติ

2.2.4.5 สามารถสังเกตพฤติกรรมที่เบี่ยงเบนไปจากพฤติกรรมปกติ

2.2.4.6 กลไกการรับมือการบุกรุกควรจะปรับเปลี่ยนให้เหมาะสมกับรูปแบบต่างๆโดยง่าย เนื่องจากทุกระบบจะมีรูปแบบการใช้งาน(usage pattern)ที่แตกต่างกัน

2.2.4.7 สามารถรับมือต่อพฤติกรรมระบบที่เปลี่ยนไปตลอดเวลา ถ้ามีแอพพลิเคชันใหม่เพิ่มเข้ามา สภาพแวดล้อมของระบบเปลี่ยนไป ระบบตรวจจับการบุกรุกควรจะปรับเปลี่ยนตามด้วย

2.2.4.8 สามารถขยายขนาดเพื่อครอบคลุมเครือข่ายขนาดใหญ่ที่มีหลายโฮสต์ได้ โดยที่ยังคงตรวจจับการบุกรุกได้อย่างแม่นยำและใช้เวลาพอเหมาะ

2.2.4.9 ถ้าส่วนประกอบบางส่วนของระบบตรวจจับการบุกรุกหยุดทำงานด้วยเหตุผลใดก็ตาม ส่วนประกอบอื่นที่เหลือควรจะมีผลกระทบน้อยที่สุดเท่าที่จะเป็นไปได้

2.2.4.10 สามารถปรับเปลี่ยนคอนฟิกกูเรชันของระบบตรวจจับการบุกรุกได้(dynamically reconfiguration) โดยไม่ต้องหยุดการทำงานของระบบตรวจจับการบุกรุก

2.3 การกวาดดูทีซีพีพอร์ต(TCP Port Scanning)

2.3.1 การกวาดดูเบื้องต้น(Introduction to Scanning)

การกวาดดู คือ การตรวจดูระบบที่เปิดอยู่และเข้าถึงได้ผ่านอินทราเน็ตหรืออินเตอร์เน็ต และดูว่าบริการ(services)ไหนที่เปิดให้ใช้ประโยชน์ได้บ้าง โดยใช้เทคนิคต่างๆ ได้แก่ ping sweeps หรือ port scanning เป็นต้น ถึงแม้ว่าในการกวาดดูทีซีพีพอร์ตนั้นไม่ได้เป็นการบุกรุกโฮสต์นั้นโดยตรง แต่จะสามารถให้ข้อมูลแก่ผู้บุกรุกเพื่อนำไปใช้ร่วมกับเทคนิคในการบุกรุกอื่นๆได้ เพราะบริการต่างๆในโปรโตคอลทีซีพี/ไอพีมักจะมีจุดอ่อนที่ผู้บุกรุกจะสามารถเจาะเข้ามายังระบบได้

การกวาดดูสามารถเปรียบเทียบเหมือนกับขโมยที่กำลังตรวจสอบว่ามีประตูและหน้าต่างของบ้านที่ต้องการจะบุกรุกเข้าไปว่ามีกี่บาน

ทุกวันนี้จำนวนของเครื่องมือกวาดจับ(scanner)อัตโนมัติเพิ่มขึ้นอยู่เรื่อยๆ เป็นผลให้มีการโจมตีริเริ่มขึ้นอย่างสำเร็จ เพราะฉะนั้นเพื่อที่จะเตรียมตัวให้ดี จำเป็นที่จะต้องเข้าใจเครื่องมือที่ใช้กวาดดูและวิธีการที่เครื่องนั้นๆใช้ในการกวาดดู

2.3.2 ประเภทของการกวาดดู(Scanning Types)

2.3.2.1 การกวาดดูโดยใช้ปิง(Ping Sweeps) เป็นวิธีที่ใช้ตรวจดูว่าเครื่องเป้าหมายเปิดอยู่หรือปิดอยู่ แบ่งออกเป็น 5 วิธี

2.3.2.1.1 ICMP Sweeps(ICMP ECHO requests)

การส่งแพ็กเก็ต ICMP ECHO request(ICMP type 8) ไปยังเครื่องเป้าหมาย เพื่อคอย ICMP ECHO reply(ICMP type 0) ถ้ามีตอบกลับมาแสดงว่าเครื่องเป้าหมายเปิดอยู่ ถ้าไม่มีตอบกลับแสดงว่าเครื่องเป้าหมายปิดอยู่

2.3.2.1.2 Broadcast ICMP

การส่งแพ็กเก็ต ICMP ECHO request ไปยังหมายเลขเครือข่าย(network address)หรือหมายเลขบรอดคาสต์(broadcast address)จะทำให้ได้ข้อมูลทั้งหมดที่ต้องการใช้สำหรับวาดเครือข่ายเป้าหมายได้โดยง่าย เนื่องจากการส่งบรอดคาสต์ไปยังโฮสต์ที่เปิดอยู่ในเครือข่ายเป้าหมาย โฮสต์เหล่านั้นจะตอบกลับไปยังเครื่องของผู้บุกรุก ทำให้ผู้บุกรุกรู้ได้ว่ามีเครื่องใดในระบบบ้าง เช่น ping 255.255.255.255 เป็นต้น

2.3.2.1.3 Non-ECHO ICMP

การป้องกันการส่งแพ็กเก็ต ICMP ECHO request ไม่เพียงพอต่อการรวบรวมข้อมูลต่างๆเกี่ยวกับระบบของผู้บุกรุก เนื่องจากผู้บุกรุกสามารถใช้ Non-ECHO ICMP เช่น การส่งแพ็กเก็ต ICMP type 13(TIMESTAMP) การส่งแพ็กเก็ต ICMP type 17(ADDRESS MASK REQUEST)

ICMP timestamp request และ reply อนุญาตให้ระบบหนึ่งถามเวลาปัจจุบัน(current time) ไปยังอีกเครื่องหนึ่งได้ ถ้ามีการตอบกลับแสดงว่าเครื่องนั้นเปิดอยู่

ICMP address mask request และ reply ถูกใช้กับระบบที่ไม่มีดิสก์ แต่ต้องการที่จะได้รับ subnet mask เวลาบูตเครื่อง เพราะฉะนั้นจึงสามารถใช้เครื่องมือที่ใช้ icmppush และ icmpquery เพื่อทำการกวาดดูชนิดนี้ได้ ทั้งนี้ไฟร์วอลล์(firewalls)ส่วนมากมักจะป้องกันเพียงการส่งแพ็กเก็ต ICMP ECHO จึงเป็นอีกช่องทางที่ผู้บุกรุกใช้การส่ง Non-ECHO ICMP เพื่อดูว่าระบบใดเปิดอยู่

2.3.2.1.4 TCP Sweeps

แทนที่จะส่ง ICMP ECHO request ก็ส่งแพ็กเก็ต TCP ACK หรือ TCP SYN(ขึ้นอยู่กับผู้ส่งว่ามีสิทธิ์รูท(root access)หรือไม่)ไปยังเครือข่ายเป้าหมาย โดยหมายเลขพอร์ตสามารถเลือกพอร์ตใดก็ได้ตามที่ต้องการ โดยปกติมักใช้พอร์ตหมายเลข 21, 22, 23, 25, 80 เนื่องจากไฟร์วอลล์มักอนุญาตให้ใช้บริการ(services)เหล่านี้ ถ้ามีการตอบรับแสดงว่าเครื่องนั้นเปิดอยู่

2.3.2.1.5 UDP Sweeps(หรือเรียกกัน "UDP Scanning")

การส่งยูดีพีดาตาแกรม(UDP datagram) ไปยังยูดีพีพอร์ตที่ต้องการจะตรวจสอบบนระบบเป้าหมาย โดยไม่มีข้อความ ICMP PORT UNREACHABLE ตอบกลับมา แสดงว่าพอร์ตเปิดอยู่

UDP Scanning ไม่น่าเชื่อถือด้วยเหตุผลต่อไปนี้

- เราเตอร์สามารถทำลายแพ็กเก็ตยูดีพีถ้ามีการส่งในอินเตอร์เน็ต
- หลายยูดีพีเซอร์วิสไม่ตอบกลับเมื่อมีการไต่ถามอย่างถูกต้อง
- ไฟร์วอลล์ปกติจะทำลายแพ็กเก็ตยูดีพี(ยกเว้น DNS)
- UDP Sweep เชื่อว่ายูดีพีพอร์ตที่ปิดอยู่จะตอบข้อความ ICMP PORT UNREACHABLE กลับมา

2.3.2.2 การกวาดดูพอร์ต(Port Scanning) เป็นวิธีที่ใช้ตรวจดูว่าเครื่องเป้าหมายมีเปิดเซอร์วิส(services) หรือพอร์ต(ports) อะไรบ้าง

เซอร์วิสที่ถูกตรวจพบว่ากำลังรอการเชื่อมต่อ(listening) อาจจะกลายเป็นจุดอ่อนของระบบ

ถ้า

- การกำหนดเซอร์วิสที่เปิดให้ใช้งานผิด เนื่องจากอาจมีบางเซอร์วิสที่ไม่ได้ใช้งานแต่เปิดไว้ เป็นการเพิ่มช่องทางให้ผู้บุกรุกเข้าระบบได้มากขึ้น
- เวอร์ชันของซอฟท์แวร์ของเซอร์วิสที่เปิดอยู่มีช่องโหว่ทางด้านความปลอดภัย

ซึ่งจุดอ่อนเหล่านี้สามารถนำไปสู่การใช้งานโดยไม่ได้รับสิทธิ์(unprevileged access)จากผู้บุกรุกได้

2.3.2.2.1 ประเภทของการกวาดดูพอร์ต(Port Scanning Types)

2.3.2.2.1.1 TCP connect() scan

วิธีนี้จะอาศัยพื้นฐานกลไกการเชื่อมต่อแบบทีซีพีเพื่อเปิดการเชื่อมต่อไปยังพอร์ตที่สนใจบนเครื่องเป้าหมาย ดังขั้นตอนต่อไปนี้

1) ส่งแพ็กเก็ต SYN ไปยังพอร์ตที่สนใจ

2) เครื่องผู้ส่งรอแพ็กเก็ตที่ตอบกลับจากเครื่องเป้าหมายว่าเป็นชนิดไหน

- ถ้าได้รับแพ็กเก็ต SYN/ACK หมายความว่า พอร์ตอยู่ในสถานะ LISTENING
- ถ้าได้รับแพ็กเก็ต RST/ACK หมายความว่า พอร์ตไม่ได้อยู่ในสถานะ LISTENING และการเชื่อมต่อจะถูกปิดลง(RESET)

3) การทำ three-way handshake จะเสร็จสมบูรณ์(ถ้าได้รับแพ็กเก็ต SYN/ACK) โดยการส่งแพ็กเก็ต ACK ไปยังระบบเป้าหมาย

4) การเชื่อมต่อจะถูกยุติหลังจากกระบวนการเชื่อมต่อแบบเต็มเสร็จสมบูรณ์

หมายเหตุ บางระบบปฏิบัติการอนุญาตให้เรียกใช้ connect() system call เพื่อเปิดการเชื่อมต่อไปยังพอร์ตที่สนใจบนเครื่องหนึ่ง โดยไม่ต้องมีสิทธิ์พิเศษใดๆ ผู้ใช้ทั่วไปเรียกใช้ connect() ได้ วิธีนี้จึงทำได้เร็ว แต่การกวาดดูประเภทนี้สามารถถูกตรวจพบได้โดยง่าย โดยดูจากล็อกของระบบเป้าหมายซึ่งจะแสดงจำนวนของการเชื่อมต่อและข้อความผิดพลาด(error messages) ทันทีหลังจากมีการเชื่อมต่อ

2.3.2.2.1.2 TCP SYN scan(half open scanning)

วิธีนี้แตกต่างจาก TCP connect() scan เพราะว่าไม่ได้เปิดการเชื่อมต่อทีซีพีแบบเต็ม

1) ส่งแพ็กเก็ต SYN ไปยังพอร์ตที่สนใจ

2) เครื่องผู้ส่งรอแพ็กเก็ตที่ตอบกลับจากเครื่องเป้าหมาย

- ถ้าได้รับแพ็กเก็ต SYN/ACK หมายความว่า พอร์ตอยู่ในสถานะ LISTENING
- ถ้าได้รับแพ็กเก็ต RST/ACK หมายความว่า พอร์ตไม่ได้อยู่ในสถานะ LISTENING

3) ถ้าได้รับแพ็กเก็ต SYN/ACK จะยุติการขอเชื่อมต่อโดยส่งแพ็กเก็ต RST กลับไป

<u>หมายเหตุ</u> การกวาดดูประเภทนี้น้อยเครื่องที่สามารถล็อกการกระทำนี้ได้ แต่ทั้งนี้การสร้างแพ็กเก็ตวิธีนี้ต้องมีสิทธิ์รูท(root privileges)

2.3.2.2.1.3 Stealth Scanning

ในบางครั้งไฟร์วอลล์และตัวกรองแพ็กเก็ตจะจับตาดูที่เซกเมนต์ SYN ที่ไปยังพอร์ตต่างๆที่กำหนด และโปรแกรมเช่น Synlogger และ Courtney สามารถที่จะตรวจจับการกวาดดูพอร์ตผ่านแฟล็ก SYN ได้ รูปแบบการทำ Stealth Scanning คือ การกวาดดูพอร์ตโดยใช่แฟล็กอื่นที่ไม่ใช่ SYN ส่งไปยังพอร์ตต่างๆ เช่น

TCP FIN scan วิธีนี้จะส่งแพ็กเก็ต FIN ไปยังพอร์ตที่สนใจบนเครื่องเป้าหมายและรอสัญญาณตอบกลับ ซึ่งถ้าพอร์ตปิดอยู่จะได้รับแพ็กเก็ต RST/ACK ตอบกลับมา แต่ถ้าพอร์ตเปิดอยู่จะไม่ได้รับอะไรตอบกลับ

Xmas Tree(Christmas Tree) วิธีนี้จะส่งแพ็กเก็ตทีซีพีโดยกำหนด Flag บางตัว ได้แก่ FIN URG และ PSH ซึ่งถ้าพอร์ตปิดอยู่จะได้รับแพ็กเก็ต RST/ACK ตอบกลับมา แต่ถ้าพอร์ตเปิดอยู่จะไม่ได้รับอะไรตอบกลับ

NULL Scanning วิธีนี้จะส่งแพ็กเก็ตทีซีพีโดยปิด Flag ทุกตัว ซึ่งถ้าพอร์ตปิดอยู่จะได้รับแพ็กเก็ต RST/ACK ตอบกลับมา แต่ถ้าพอร์ตเปิดอยู่จะไม่ได้รับอะไรตอบกลับ

2.3.2.2.1.4 TCP reverse ident scanning

เป็นการกวาดดูพอร์ตผ่านทางโปรโตคอลไอเด็น(ident)เนื่องจากโปรโตคอลไอเด็นจะอนุญาตให้มีการเปิดเผยชื่อผู้ใช้ที่เป็นเจ้าของโปรเซส(process)ต่างๆที่เชื่อมต่อผ่านทางโปรโตคอลทีซีพี โดยที่ไม่ต้องเริ่มต้นเชื่อมต่อบริการนั้นเอง

2.3.2.2.1.5 FTP bounce attack scanning

เป็นการกวาดดูพอร์ตผ่านทางเอฟทีพีเซอร์ฟเวอร์(FTP Server) วัตถุประสงค์ก็เพื่อซ่อนตัวอยู่หลังเอฟทีพีเซอร์ฟเวอร์ ทำให้ผู้ถูกบุกรุกไม่สามารถทราบได้ว่าผู้บุกรุกมาจากที่ไหน

2.3.2.2.2 เทคนิคของการกวาดดูพอร์ต(Port Scanning Techniques)

2.3.2.2.2.1 การกวาดดูพอร์ตแบบสุ่ม(Random Port Scan)

ระบบตรวจจับการบุกรุกโดยมากที่ขายกันและไฟร์วอลล์จะค้นหาความพยายามที่จะเชื่อมต่อแบบเรียงลำดับ(sequential connection attempts) เมื่อรูปแบบตรงตามกฎการกวาดดูพอร์ตก็ถูกรายงาน

การสุ่มลำดับของพอร์ตที่ถามอาจจะป้องกันการตรวจจับจากระบบตรวจจับการบุกรุกได้

2.3.2.2.2.2 การกวาดดูอย่างช้าๆ(Slow Scan)

ระบบตรวจจับการบุกรุกสามารถตรวจจับการกวาดดูพอร์ตได้โดยวิเคราะห์ปริมาณแพ็กเก็ตในเครือข่ายภายในช่วงเวลาหนึ่ง(เรียกว่า "site detection threshold") ถ้ามีมากถึงค่าดังกล่าวก็จะตรวจพบว่ามีการกวาดดูพอร์ตเกิดขึ้น แฮกเกอร์บางคนจึงอดทนอย่างมากใช้เครื่องกวาดดูเครือข่าย(network scanner) โดยขยายเวลาที่ใช้ในการกวาดดูให้นานขึ้น ตัวอย่างเช่น อัตราการกวาดดู(scan rate)ช้าเท่ากับ 2 แพ็กเก็ตต่อวันต่อเครื่องเป้าหมาย ถ้าผู้โจมตีสามารถเดา detection threshold ของเครื่องเป้าหมายได้ ก็สามารถลดโอกาสของการตรวจจับให้น้อยที่สุด หรือไม่ถูกตรวจจับเลย

2.3.2.2.2.3 การกวาดดูแบบแบ่งเป็นส่วนย่อย(Fragmentation Scanning)

ทุกไอพีแพ็กเก็ตที่ใช้ส่งข้อมูลสามารถถูกแบ่งเป็นส่วนย่อย(Fragment) ดังนั้นบางอุปกรณ์ที่กรองแพ็กเก็ตและระบบตรวจจับการบุกรุกอาจจะรวมแพ็กเก็ตกลับมาผิด ทำให้ไม่สามารถตรวจจับการกวาดดูพอร์ตได้ ทั้งนี้ขึ้นอยู่กับแต่ละเครือข่ายว่ามีประสิทธิภาพสูงพอที่จะรองรับแพ็กเก็ตที่แบ่งเป็นส่วนย่อยได้แค่ไหน

2.3.2.2.2.4 นกต่อ(Decoy)

เครื่องมือกวาดดูเครือข่ายบางตัวรวมลักษณะนกต่อหรือการหลอกหมายเลขไอพีก่อนทำการกวาดดูพอร์ตจึงทำให้การตรวจจับยากขึ้น

2.3.2.2.2.5 การกวาดดูแบบประสานกัน(Coordinated Scans)

การกวาดดูพอร์ตบนเครื่องเป้าหมายจากเครื่องผู้บุกรุกเพียงเครื่องเดียวจะใช้เวลาค่อนข้างมากจึงทำให้ระบบตรวจจับการบุกรุกตรวจจับได้ก่อน การกวาดดูอย่างช้าๆบางครั้งก็สามารถถูกตรวจจับได้เหมือนกัน

การกวาดดูแบบประสานกันเป็นการร่วมมือกันของเครื่องผู้บุกรุกหลายเครื่องช่วยกันกวาดดูพอร์ตต่างๆบนเครื่องเป้าหมาย โดยแบ่งกันกวาดดูในช่วงเวลาที่แตกต่างกัน ดังนั้นวิธีนี้จึงเกือบจะเป็นไม่ได้เลยที่จะตรวจจับการกวาดดูพบ

2.3.3 ตัวอย่างของโปรแกรมกวาดดูทีซีพีพอร์ต

โปรแกรมกวาดดูพอร์ตในปัจจุบันมีอยู่มากมายหลายโปรแกรม บางโปรแกรมก็พัฒนาขึ้นเพื่อใช้กวาดดูพอร์ตโดยเฉพาะ แต่บางโปรแกรมก็สามารถทำงานอย่างอื่นได้ด้วย ซึ่งในที่นี้จะขอกล่าวถึงเฉพาะส่วนที่ใช้ในการกวาดดูพอร์ตทีซีพีเท่านั้น

NMAP(Network MAPper) เป็นโปรแกรมสำหรับการสำรวจเครือข่าย ซึ่งถูกออกแบบมาเพื่อให้ผู้ดูแลระบบและผู้ที่อยากรู้อยากเห็นใช้ในการกวาดดูเครือข่ายขนาดใหญ่เพื่อดูว่าโฮสต์เครื่องไหนเปิดอยู่และเซอร์วิสไหนที่เปิดให้บริการบ้าง NMAP เป็นโปรแกรมกวาดดูพอร์ตที่นิยมกันมากเพราะว่ามีความสามารถในการกวาดดูพอร์ตได้หลายรูปแบบ คือ

- TCP connect() scanning
- TCP SYN(half open) scanning
- TCP FIN, Xmas, or NULL(stealth) scanning
- TCP ftp proxy(bounce attack) scanning
- SYN/FIN scanning using IP fragments(bypasses some packet filters)
- TCP ACK and Window scanning
- UDP raw ICMP port unreachable scanning
- ICMP scanning(ping-sweep)
- TCP Ping scanning
- Direct(non portmapper) RPC scanning
- Remote OS Identification by TCP/IP Fingerprinting
- Reverse-ident scanning

นอกจากนี้ NMAP ยังสนับสนุนลักษณะการทำงานอย่างมีประสิทธิภาพ(performance)และความน่าเชื่อถือ(reliability) ได้แก่ การคำนวณเวลาหน่วงได้เอง(dynamic delay time calculations) การส่งแพ็กเก็ตกรณีส่งไม่ถึงและหมดเวลา(packet timeout and retransmission) การกวาดดูแบบขนาน(parallel port scanning) การตรวจดูโฮสต์ที่ปิดอยู่ผ่านการปิงแบบขนาน(detection of down hosts via parallel pings) อีกทั้งยังสามารถระบุพอร์ตและเครื่องเป้าหมายได้เป็นหนึ่งเดียวหรือเป็นช่วงได้ด้วย

รูปแบบของคำสั่ง NMAP

nmap [Scan Type(s)] [Options] <host or net#1 ... [#N]>

รูปแบบการกวาดดู(Scan Types) ที่ขอกล่าวถึง เนื่องจากโปรแกรมที่พัฒนาขึ้นจะตรวจจับการกวาดดูทีซีพีพอร์ตต่อไปนี้ ได้แก่

-sT แทน TCP connect() scanning (การกวาดดูรูปแบบนี้ไม่ต้องมี root privileges)
-sS แทน TCP SYN scanning (การกวาดดูรูปแบบนี้ต้องมี root privileges)
-sF แทน Stealth FIN scanning (การกวาดดูรูปแบบนี้ต้องมี root privileges)
-sX แทน Stealth Xmas scanning (การกวาดดูรูปแบบนี้ต้องมี root privileges)
-sN แทน Stealth Null scanning (การกวาดดูรูปแบบนี้ต้องมี root privileges)

ทางเลือก(Options) ที่ขอกล่าวถึง เนื่องจากมักใช้บ่อย ได้แก่

-p <port ranges> แทน การระบุพอร์ตที่ต้องการกวาดดูเช่น –p 23 หมายถึงต้องการกวาดดูพอร์ต 23 เพียงพอร์ตเดียว –p 20-30,139,60000- หมายถึงต้องการกวาดดูพอร์ต

ระหว่าง 20 ถึง 30 และพอร์ต 139 และพอร์ตที่มากกว่า 60000 ขึ้นไป เป็นต้น โดยปกติค่าเริ่มต้นจะกวาดดูพอร์ตระหว่าง 1 ถึง 1024

ตัวอย่างการใช้คำสั่ง

nmap -v target.example.com

ตัวอย่างนี้จะเป็นการกวาดดูทุกทีซีพีพอร์ตบนเครื่อง target.example.com โดย –v หมายถึง การแสดงขั้นตอนการทำงานของโปรแกรมด้วย

nmap -sX -p 22,53 128.210.*.1-127

ตัวอย่างนี้จะเป็นการกวาดดูแบบ Stealth Xmas scanning บนทุกเครื่องของเครือข่ายย่อยของคลาส B ของ 128.210 โดยดูเครื่องที่ 1 ถึง 127 และสนใจกวาดดูพอร์ต 22(sshd) และ 53(dns)

บทที่ 3

การออกแบบและหลักการทำงานของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

3.1 โปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต(TCP Port Scanning Detection Program:TCPPSD)

รูปแบบของการบุกรุกระบบคอมพิวเตอร์ที่นำมาใช้ในการวิเคราะห์แพ็กเก็ตเพื่อตรวจสอบการบุกรุกมีอยู่มากมายหลายรูปแบบ ในโครงการพัฒนาระบบงานนี้จะพัฒนาโปรแกรมตรวจจับการบุกรุกโดยจะเปรียบเทียบผลการวิเคราะห์แพ็กเก็ตกับรูปแบบของการบุกรุกที่มักตรวจพบกันได้โดยทั่วไป คือ การกวาดดูทีซีพีพอร์ต ซึ่งเป็นรูปแบบการบุกรุกโดยผ่านโปรโตคอลทีซีพี/ไอพีเพื่อตรวจสอบการให้บริการพื้นฐานของชุดโปรโตคอลทีซีพี/ไอพี ซึ่งบริการพื้นฐานเหล่านี้มักจะมีจุดอ่อนที่ผู้บุกรุกสามารถใช้ในการบุกรุกเข้ามาเพื่อทำอันตรายระบบ และจะรายงานสถานะที่ได้จากการวิเคราะห์แพ็กเก็ตให้ผู้ดูแลระบบทราบ โดยซอฟท์แวร์ที่พัฒนาขึ้นจะสามารถตรวจจับการบุกรุกสถานีงานด้วยวิธีการกวาดดูทีซีพีพอร์ตซึ่งสามารถปรับเปลี่ยนค่าที่ใช้ในการตัดสินใจให้เหมาะสมกับสภาพแวดล้อมของระบบเครือข่ายคอมพิวเตอร์ได้โดยแก้ไขแฟ้มคอนฟิกูเรชัน ซอฟท์แวร์จะอยู่ในรูปของโปรแกรมภาษาซีและทำงานบนระบบปฏิบัติการลีนุกซ์(RedHat 7.0)

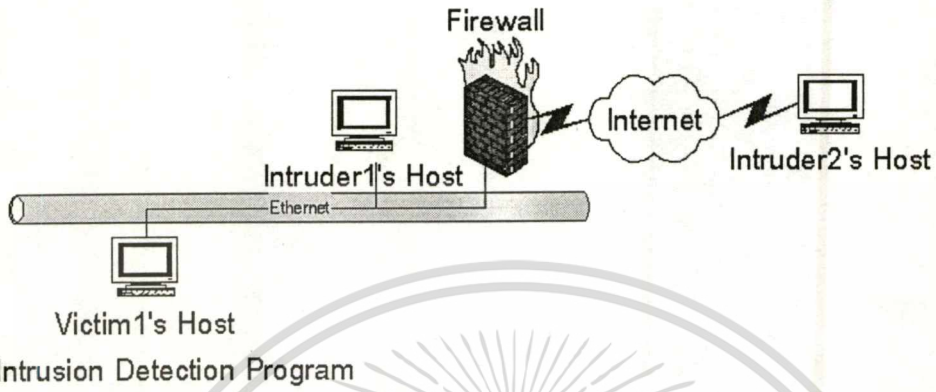
3.2 ประโยชน์ของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

คือ สามารถตรวจจับการกวาดดูทีซีพีพอร์ตได้ เป็นการบอกให้ผู้ดูแลระบบรู้ตัวว่าโฮสต์ภายในเครือข่ายกำลังถูกบุกรุกและหามาตรฐานป้องกันได้ก่อนที่ผู้บุกรุกจะทำอันตรายระบบต่อไป ซึ่งผู้บุกรุกมักใช้การกวาดดูทีซีพีพอร์ตเพื่อวาดเครือข่ายเป้าหมายว่ามีการโครงสร้างเป็นอย่างไร เมื่อได้ข้อมูลเกี่ยวกับโฮสต์และเซอร์วิสที่เปิดให้บริการ ผู้บุกรุกก็จะทำการหาช่องโหว่จากซอฟท์แวร์ของระบบเพื่อบุกรุกเข้าระบบต่อไป

3.3 ภาพรวมของระบบที่จะทำ

จากรูปที่ 3-1 เมื่อมีผู้บุกรุกทั้งจากภายในระบบอินทราเน็ต(Intruder1's Host) และระบบอินเตอร์เน็ต(Intruder2's Host) ทำการกวาดดูเซอร์วิสที่เปิดให้บริการของเครื่องที่เป็นเหยื่อ(Victim1's Host) แต่เนื่องจากเครื่องที่เป็นเหยื่อมีโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตคอยตรวจจับอยู่ โดยดูจากแพ็กเก็ตที่วิ่งเข้ามาในเครื่องและวิเคราะห์ดูถ้าเข้าข่ายการกวาดดูทีซีพี

พอร์ตก็จะส่งเมล์แจ้งไปยังผู้ดูแลระบบ(Victim1's System Administrator) ว่ามีการกวาดดูทีซีพีพอร์ตเกิดขึ้นให้หาทางป้องกันอันตรายที่อาจจะเพิ่มขึ้นในอนาคตกับเครื่องที่เป็นเหยื่อ



รูปที่ 3-1 แสดงภาพรวมของระบบที่จะทำ

3.4 สถาปัตยกรรมของระบบ(System Architecture)

3.4.1 สมมติฐานในการตรวจจับการกวาดดูทีซีพีพอร์ต

3.4.1.1 ประเภทของการโจมตี(Attack types) ได้แก่ TCP connect, TCP half-connect, Stealth FIN , Xmas Tree และ NULL Scanning

3.4.1.2 พฤติกรรมการโจมตี(Attack behavior) ได้แก่ ณ เวลาเดียวกัน มีเพียงการโจมตีเดียวซึ่งมุ่งโจมตีเครื่องเป้าหมายเพียงเครื่องเดียว

3.4.1.3 ตำแหน่งของโปรแกรมตรวจจับ(System location) ได้แก่ โปรแกรมตรวจจับจะป้องกันสถานีงานหนึ่ง(host-based) และสถานีงานนั้นต้องมีโปรแกรมตรวจจับทำงานอยู่ในเครื่องที่เป็นเป้าหมายของการโจมตี

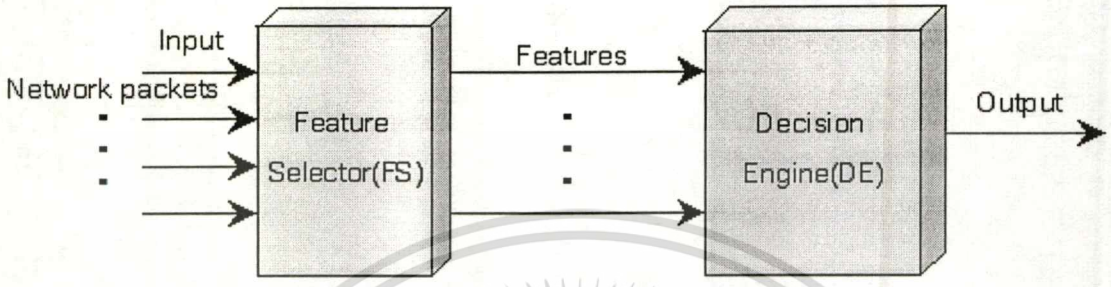
3.4.1.4 ข้อมูลที่ใช้วิเคราะห์ ได้แก่ เครื่องที่มีโปรแกรมตรวจจับอยู่จะรับแพ็กเก็ตจากเครือข่ายเพื่อทำการวิเคราะห์ชุดของแพ็กเก็ตที่เข้าข่ายกวาดดูทีซีพีพอร์ต

3.4.1.5 สภาพแวดล้อมของเครือข่าย ได้แก่ อีเทอร์เน็ตที่ใช้ความเร็วร่วมกัน 10 Mbps(Share-based 10Mbps Ethernet)

3.4.2 โครงสร้างของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

โปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตใช้เทคนิคแผนผังสถานะโดยใช้กฎ(rule-based state diagram) มีการกำหนดค่าขอบเขต(threshold)ที่เหมาะสม แล้วพัฒนาเป็นส่วนหนึ่งของระบบตรวจจับการบุกรุกทางเครือข่าย

จุดสำคัญของโปรแกรมคือพิจารณาถึงลักษณะเด่น(features)ซึ่งสกัดมาจากแพ็กเก็ตในเครือข่าย โปรแกรมมีการใช้ค่าขอบเขตช่วยในการสรุป โครงสร้างของโปรแกรมประกอบด้วย 2 ส่วน คือ (1) Feature Selector(FS) และ (2) Decision Engine(DE) ดังแสดงในรูป 3-2



รูปที่ 3-2 แสดงโครงสร้างของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

3.4.2.1 ผู้เลือกลักษณะเด่น(Feature Selector หรือ FS)

หน้าที่หลักของ FS คือ สกัดลักษณะเด่นที่สำคัญ(features) และข้อมูลอื่นจากแพ็กเก็ต ลักษณะเด่นที่สกัด เช่น หมายเลขไอพีต้นทาง TCP flags หมายเลขพอร์ตปลายทาง เวลาระหว่างแพ็กเก็ต(packet time interval) และจำนวนแพ็กเก็ตที่ได้รับ

รูปแบบของลักษณะเด่นที่กำหนดไว้ก่อน(Predefined Feature หรือ K) ประกอบด้วย 5 ส่วนดังต่อไปนี้

$K = (F, S, P, \Delta T, N)$

แต่ละส่วนมีความหมายต่อไปนี้

1) F = { URG, ACK, PSH, RST, SYN, FIN } เป็นชุดของแฟล็ก(Flags)ที่เป็นลักษณะสำคัญบนเฮดเดอร์ทีซีพีที่แพ็กเก็ตผิดปกติ

2) S คือ หมายเลขไอพีต้นทางของแพ็กเก็ต

3) P คือ หมายเลขพอร์ตของปลายทาง

4) ΔT คือ ช่วงเวลาของสองแพ็กเก็ตที่ตามกันมา

5) N คือ จำนวนของแพ็กเก็ตที่สงสัย

3.4.2.2 เครื่องกลตัดสินใจ(Decision Engine หรือ DE)

หน้าที่หลักของ DE คือ ส่วนวิเคราะห์ของระบบ โดยจะวิเคราะห์เหตุการณ์กับชุดของกฎการตรวจจับ

*รูปแบบที่บ่งบอกถึงการบุกรุก(intrusive patterns)*ต้องประกอบด้วยหลายเงื่อนไข

1) ถ้า F เป็น subset ของ { URG, ACK, PSH, RST, SYN, FIN }

2) แต่ละแพ็กเก็ตมาจากหมายเลขไอพีต้นทางเดียวกัน

3) ถูกส่งไปยังพอร์ตบนเครื่องปลายทางที่ต่างกัน

4) ΔT ต้องน้อยกว่า $\propto$ ซึ่งค่าปัจจุบันของ $\propto$ คือ 1 วินาที(ได้มาจากเครื่องมือการกวาดดูหลายอันที่มีคนรู้จักดี) ซึ่งค่านี้พบว่าไม่มีสองแพ็กเก็ตที่ตามกันมาสร้างการเชื่อมต่อต่างพอร์ตแต่มาจากหมายเลขไอพีต้นทางเดียวกันภายใน 1 วินาทีได้ แต่สามารถกำหนดค่า $\propto$ เป็นค่าอื่นได้ตามความเหมาะสม

5) N จำนวนของแพ็กเก็ตที่สงสัยต้องน้อยกว่า β ในที่นี้จะใช้ $\beta = 20$(ได้มาจากจำนวนพอร์ตที่ใช้หาประโยชน์บ่อยในเครื่องมือการกวาดดูส่วนมาก) แต่สามารถกำหนดค่า β เป็นค่าอื่นได้ตามความเหมาะสม

หมายเหตุ ทั้ง ΔT และ N อาจจะปรับได้ ขึ้นอยู่กับสภาพแวดล้อมทางเครือข่าย

จากเงื่อนไขด้านบนสามารถแทนด้วย*ชุดของกฎ(a set of rules)*ดังต่อไปนี้

START: IF flag is a subset of set F THEN

IF S are same THEN

IF P are different THEN

IF $\Delta T < \propto$ THEN

IF N = β THEN

Intrusion alerts

ELSE goto START

ELSE stop detection

ELSE stop detection

ELSE goto START

ELSE stop detection

โดยกำหนด 13 *สถานะ(states)*สำหรับการดำเนินการตรวจจับ ได้แก่

1) CLOSED: สถานะว่าง ไม่มีการตรวจจับ

2) LISTEN: สถานะเริ่มต้นของปฏิบัติการตรวจจับ

3) FLAG_RCVD: รับ F ของ 5-tuple K

4) SAME_SRC: แพ็กเก็ตมีหมายเลขไอพีต้นทางเดียวกัน

5) DIFF_SRC: แพ็กเก็ตมีหมายเลขไอพีต้นทางต่างกัน

6) SAME_PORT: แพ็กเก็ตมีหมายเลขพอร์ตปลายทางเดียวกัน

7) DIFF_PORT: แพ็กเก็ตมีหมายเลขพอร์ตปลายทางต่างกัน

8) TIME_LESS: ช่วงเวลาน้อยกว่า $\propto$

9) WAIT: สถานะคอยและยกเว้น

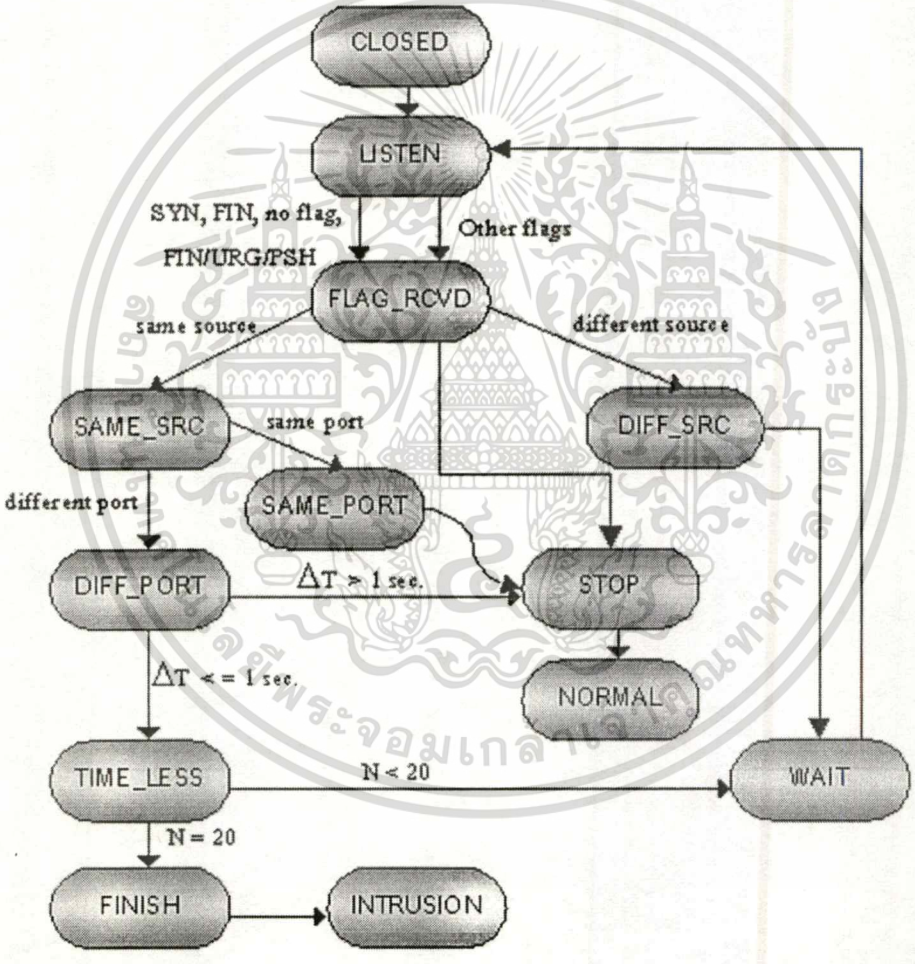
10) FINISH: การตรวจสอบการบุกรุกทั้งหมดถูกทำแล้ว

11) INTRUSION: บ่งชี้ว่าเกิดการบุกรุก เป็นการตัดสินใจครั้งสุดท้าย

12) STOP: การตรวจสอบการบุกรุกถูกหยุด

13) NORMAL: พฤติกรรมปกติ

จากกฎ(rules)และสถานะ(states)ข้างต้น นำมาสร้างเป็นแผนผังสถานะการตรวจจับ(detection state diagram) ดังแสดงในรูปที่ 3-3



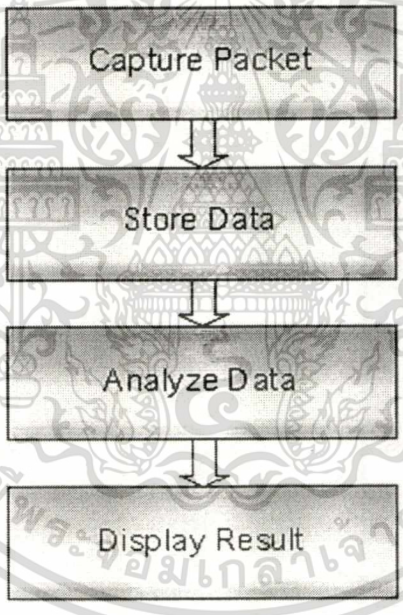
รูปที่ 3-3 แสดงแผนผังสถานะการตรวจจับ(detection state diagram)

3.5 โครงสร้างการทำงานของระบบ

แนวคิดในการออกแบบโปรแกรมตรวจจับการบุกรุกสถานีงานนั้นมีมากมายหลายรูปแบบ ทั้งการวิเคราะห์โดยนำข้อมูลที่ได้จากแฟ้มล็อกของเครื่องคอมพิวเตอร์ที่เก็บข้อมูลเกี่ยวกับแพ็กเก็ต

ที่เข้ามายังเครื่องมาวิเคราะห์ ซึ่งรูปแบบนี้จะสามารถวิเคราะห์ข้อมูลได้อย่างถูกต้องเพราะข้อมูลที่ได้มาจะได้มาครบถ้วนแต่วิธีนี้จะไม่สามารถรายงานผลได้อย่างรวดเร็วทำให้อาจจะป้องกันอันตรายที่อาจจะเกิดขึ้นตามมาไม่ทัน หรือการออกแบบให้การดักจับแพ็กเก็ตไปพร้อมกับการวิเคราะห์ข้อมูลก็ได้ ในการพัฒนาโปรแกรมนี้จะใช้วิธีดักจับแพ็กเก็ตไปพร้อมกับการวิเคราะห์ซึ่งจะทำให้การรายงานผลเป็นไปอย่างรวดเร็วและเหมาะที่จะพัฒนาเพื่อให้มีความสามารถในการตรวจจับการบุกรุกรูปแบบอื่นซึ่งอาจจะต้องกระทำการบางอย่างก่อนที่การบุกรุกจะเสร็จสมบูรณ์เพื่อไม่ให้เกิดความเสียหายแก่ระบบเครือข่ายคอมพิวเตอร์

โปรแกรมนี้พัฒนาโดยใช้ภาษาซีบนระบบปฏิบัติการยูนิกซ์ ซึ่งเป็นระบบปฏิบัติการที่เหมาะสมกับการพัฒนาซอฟท์แวร์ที่เกี่ยวข้องกับระบบเครือข่ายคอมพิวเตอร์ ข้อจำกัดต่างๆมีน้อยกว่าระบบวินโดว์ รูปที่ 3-4 แสดงถึงโครงสร้างการทำงานของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตที่พัฒนาขึ้น



รูปที่ 3-4 แสดงโครงสร้างการทำงานของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

โปรแกรมทำงานโดยเริ่มจากการเปิดแฟ้มคอนฟิกูเรชั่น เพื่ออ่านเงื่อนไขที่ผู้ใช้กำหนดเพิ่มขึ้นมา เงื่อนไขเหล่านี้จะเป็นการกำหนดหมายเลขไอพีและพอร์ตที่เราไว้ใจ และกำหนดพอร์ตที่เราต้องการจะเฝ้ามองเป็นพิเศษเมื่อมีผู้เชื่อมต่อเข้ามา รวมไปถึงชื่อแอคเคาท์(account)ที่ต้องการให้รายงานผลผ่านทางอิเล็กทรอนิคเมล์ หลังจากนั้นก็จะทำการเปิดซ็อคเก็ตเพื่อจับแพ็กเก็ตที่เข้ามายังเครื่องผ่านโปรโตคอลทีซีพีและเก็บข้อมูลที่สำคัญที่ต้องใช้ในการวิเคราะห์ไว้ตามโครงสร้างที่ออกแบบไว้ หลังจากนั้นก็นำข้อมูลจากแพ็กเก็ตที่เก็บไว้มาวิเคราะห์ตามเงื่อนไขต่างๆ ที่ได้กำหนดขึ้น

และเมื่อโปรแกรมวิเคราะห์แพ็กเก็ตแล้วพบว่าเป็นการบุกรุกโดยวิธีการกวาดดูทีซีพีพอร์ต โปรแกรมก็จะรายงานผลจากการวิเคราะห์ให้ผู้ดูแลระบบทราบทั้งผ่านทางหน้าจอและผ่านทางอิเล็คทรอนิคเมล์ รวมไปถึงการเก็บผลการวิเคราะห์ไว้ในรูปของแฟ้มล็อกเพื่อให้ผู้ดูแลระบบสามารถกลับมาตรวจสอบเหตุการณ์ที่เกิดขึ้นได้ในภายหลัง

3.6 การจับแพ็กเก็ตข้อมูล(Capture Packet)

การจับแพ็กเก็ตข้อมูลมีขั้นตอนการทำงานตามลำดับ คือ เปิดซ็อคเก็ต อ่านข้อมูลจากอินเตอร์เฟสและอ่านเวลาการจับแพ็กเก็ต

3.6.1 เปิดซ็อคเก็ตรับทุกแพ็กเก็ต โดยเรียกใช้ฟังก์ชัน socket() ดังรูปที่ 3-5

```
socket(AF_INET,SOCK_RAW,IPPROTO_TCP)
```

รูปที่ 3-5 แสดงการเปิดซ็อคเก็ตเพื่อรับแพ็กเก็ตข้อมูล

3.6.2 อ่านข้อมูลจากอินเตอร์เฟส โดยเรียกใช้ฟังก์ชัน read() ดังรูปที่ 3-6

```
read(fd, buffer, SIZE_BUFFER)
```

รูปที่ 3-6 แสดงการอ่านข้อมูลจากอินเตอร์เฟสมาเก็บในบัฟเฟอร์

3.6.3 อ่านเวลาที่จับแพ็กเก็ต โดยการเรียกใช้ฟังก์ชัน time() ดังรูปที่ 3-7

```
time(0)
```

รูปที่ 3-7 แสดงการอ่านเวลาที่จับแพ็กเก็ต

3.7 การจัดเก็บข้อมูล(Store Data)

แพ็กเก็ตจะถูกจัดเก็บในบัฟเฟอร์ โครงสร้างสำหรับการจัดเก็บมีลักษณะดังรูปที่ 3-8

```
struct host{
        unsigned int src_addr;
        time_t firsttime;
        time_t lasttime;
        int lastindex;
        unsigned short  src_port[MAXSCAN];
        unsigned short  dst_port[MAXSCAN];
        int count;
        u_int8_t flags[MAXSCAN];
        int diffflagcount;
        int sameportcount;
        int timeintervalcount;
        int portmatch;
};
```

รูปที่ 3-8 แสดงโครงสร้างสำหรับจัดเก็บข้อมูลแพ็กเก็ตจากแต่ละ src_addr

โครงสร้างสำหรับจัดเก็บข้อมูลแพ็กเก็ตจากแต่ละหมายเลขเครื่องต้นทาง(src_addr) อธิบายได้ดังนี้

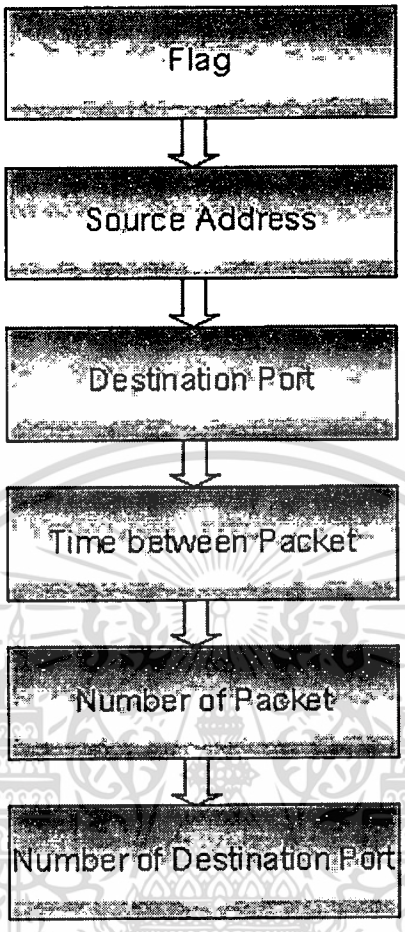
- หมายเลขเครื่องต้นทาง ใช้แบบข้อมูลเป็นเลขจำนวนเต็มบวกขนาด 32 บิต
- เวลาที่ได้รับแพ็กเก็ตแรกจากเครื่องต้นทาง ใช้แบบข้อมูลเป็น time_t
- เวลาที่ได้รับแพ็กเก็ตล่าสุดจากเครื่องต้นทาง ใช้แบบข้อมูลเป็น time_t
- ตำแหน่งล่าสุดของหมายเลขไอพีต้นทางนั้น
- หมายเลขพอร์ตเครื่องต้นทาง ใช้แบบข้อมูลเป็นเลขจำนวนเต็มบวกขนาด 16 บิต
- หมายเลขพอร์ตเครื่องปลายทาง ใช้แบบข้อมูลเป็นเลขจำนวนเต็มบวกขนาด 16 บิต
- จำนวนของแพ็กเก็ตที่สงสัยจากเครื่องต้นทางหนึ่ง ใช้แบบข้อมูลเป็นเลขจำนวนเต็มขนาด 32 บิต
- แฟล็กที่ทีซีพีแพ็กเก็ตกำหนด ใช้แบบข้อมูลเป็นเลขจำนวนเต็มบวกขนาด 8 บิต

- จำนวนครั้งที่แฟล็กของสองแพ็กเก็ตที่ติดกันมีแฟล็กต่างกัน ใช้แบบข้อมูลเป็นเลขจำนวนเต็มขนาด 32 บิต
- จำนวนครั้งที่พอร์ตของสองแพ็กเก็ตที่ติดกันมีพอร์ตเหมือนกัน ใช้แบบข้อมูลเป็นเลขจำนวนเต็มขนาด 32 บิต
- จำนวนครั้งที่เวลาของสองแพ็กเก็ตที่ติดกันมีค่ามากกว่าค่าเวลาที่กำหนด(TIMEINTERVAL) ใช้แบบข้อมูลเป็นเลขจำนวนเต็มขนาด 32 บิต
- จำนวนของแพ็กเก็ตที่มีพอร์ตปลายทางตรงกับพอร์ตที่เฝ้าดูเป็นพิเศษจากเครื่องต้นทางหนึ่ง ใช้แบบข้อมูลเป็นเลขจำนวนเต็มขนาด 32 บิต

ในระหว่างการจับแพ็กเก็ต โปรแกรมจะต้องใช้เวลาในการจัดเก็บและการทำงานอย่างอื่นให้น้อยที่สุดเพื่อจะได้มีเวลาพอสำหรับการจับแพ็กเก็ตต่อไป การจัดเก็บลงดิสก์จะใช้เวลามากกว่าการจัดเก็บลงหน่วยความจำมาก ในโปรแกรมนี้จึงจัดเก็บข้อมูลลงเฉพาะในหน่วยความจำเพื่อใช้เวลาน้อยที่สุด ดังนั้นโปรแกรมจะทำงานได้ถูกต้องและรวดเร็วหากระบบมีหน่วยความจำหลักมากเพียงพอ

3.8 การวิเคราะห์ข้อมูล(Analyze Data)

การวิเคราะห์จะเปรียบเทียบข้อมูลจากแพ็กเก็ตที่เก็บไว้กับเงื่อนไขที่กำหนดไว้ ซึ่งมีลำดับในการวิเคราะห์ดังรูปที่ 3-9



รูปที่ 3-9 แสดงลำดับการวิเคราะห์เงื่อนไขที่เข้าข่ายการกวาดดูทีซีพีพอร์ต

จากรูปที่ 3-9 สามารถอธิบายค่าของเงื่อนไขต่างๆที่ใช้ในการพิจารณาเพื่อตัดสินใจว่าระบบเครือข่ายคอมพิวเตอร์ถูกบุกรุกหรือไม่ ซึ่งมีรายละเอียดดังต่อไปนี้

- แฟล็ก (Flag)

  ถ้าแฟล็ก ack ถูกเซ็ตเป็น 1 ไว้แสดงว่าแพ็กเก็ตนี้อาจจะเป็นการตอบรับการเชื่อมต่อและโดยปกติแล้วถ้าเป็นแพ็กเก็ตที่ถูกสร้างขึ้นมาโดยไม่ได้เป็นการตอบรับแฟล็ก ack จะถูกเซ็ตเป็น 0 ดังนั้นถ้าแฟล็ก ack ถูกเซ็ตเป็น 1 โปรแกรมนี้จะไม่เก็บข้อมูลของแพ็กเก็ตเหล่านี้ไว้ โดยจะพิจารณาแฟล็กอื่นๆที่เกี่ยวข้องต่อไปด้วยดังตารางที่ 3-1 ได้แก่

Scan Type	Flag of TCP Header					
	URG	ACK	PSH	RST	SYN	FIN
TCP connect() or TCP SYN	0	0	0	0	1	0
Stealth FIN	0	0	0	0	0	1
Stealth Xmas	1	0	1	0	0	1
Stealth NULL	0	0	0	0	0	0

ตารางที่ 3-1 แสดงความสัมพันธ์ระหว่าง Scan Type และ Flag ของ TCP Header

- ถ้าแฟล็ก SYN ถูกเซ็ตเป็น 1 แสดงว่าอาจจะเป็นการกวาดดูแบบ TCP connect() หรือ TCP SYN ก็ได้(flags=0x02)

- ถ้าแฟล็ก FIN ถูกเซ็ตเป็น 1 แสดงว่าเป็นการกวาดดูแบบ Stealth FIN (flags=0x01)

- ถ้าแฟล็ก URG, PSH และ FIN ถูกเซ็ตเป็น 1 แสดงว่าเป็นการกวาดดูแบบ Stealth Xmas(flags=0x29)

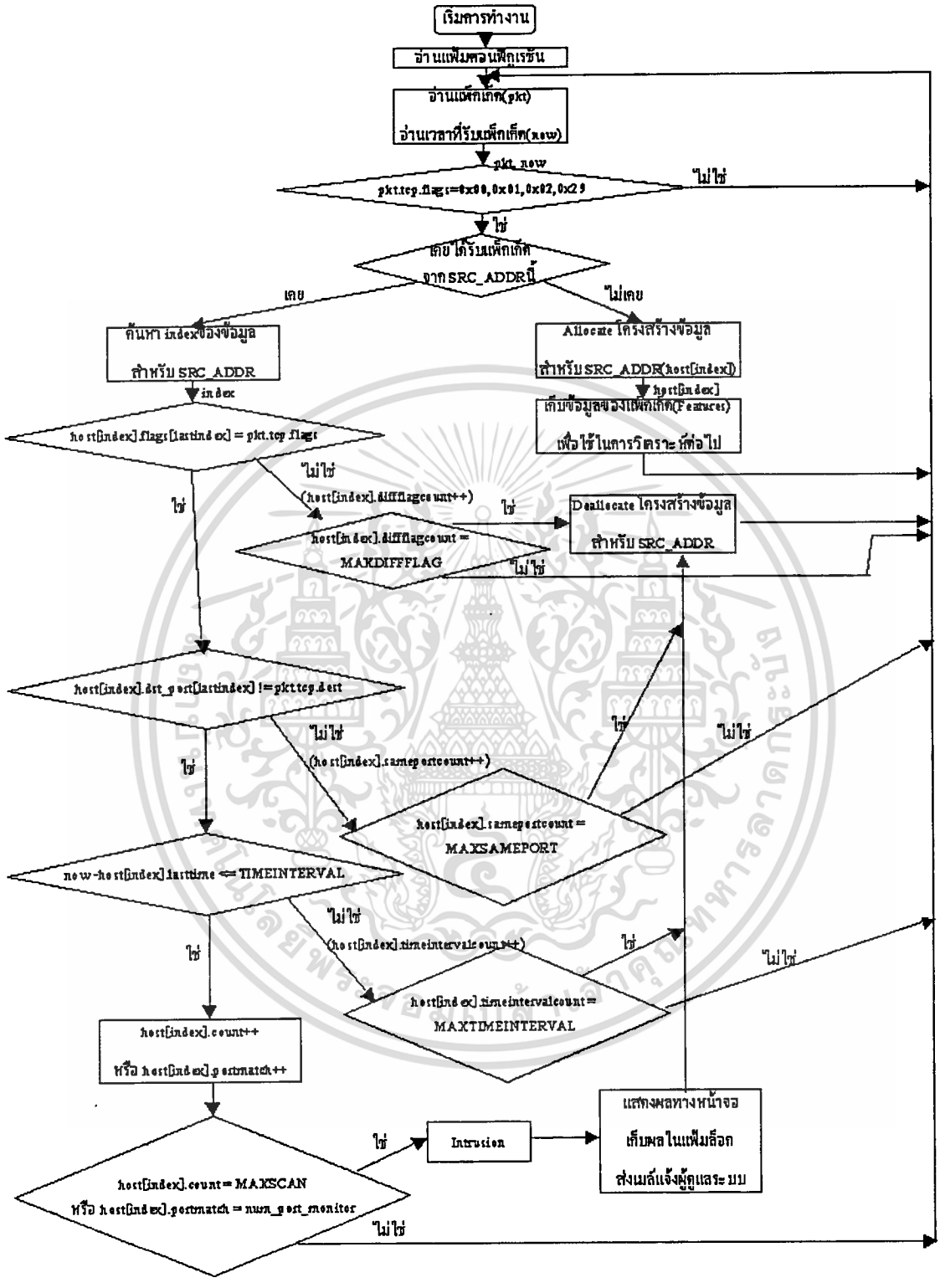
- ถ้าทุกแฟล็กถูกเซ็ตเป็น 0 หมด แสดงว่าเป็นการกวาดดูแบบ Stealth NULL (flags=0x00)

- หมายเลขไอพีของเครื่องต้นทาง(Source Address)
  ในการตรวจสอบเงื่อนไขนี้จะตรวจสอบว่าหมายเลขไอพีต้นทางนี้เคยส่งแพ็กเก็ตเข้ามาแล้วหรือไม่ และ เป็นหมายเลขไอพีที่ไว้วางใจหรือไม่ ถ้าไม่เคยมีการส่งแพ็กเก็ตเข้ามาก็จะเริ่มเก็บข้อมูลที่มาจากหมายเลขไอพีนี้ แต่ถ้าเคยส่งแพ็กเก็ตเข้ามาแล้วก็จะเพิ่มค่าจำนวนครั้งใจในการสถาปนาการเชื่อมต่อซึ่งต้องใช้เพื่อการตัดสินใจในการวิเคราะห์
- หมายเลขพอร์ตของเครื่องปลายทาง(Destination Port)
  ในการกำหนดส่วนนี้เป็นส่วนที่ทำงานร่วมกับแฟ้มคอนฟิกูเรชันที่เพิ่มเติมขึ้นมา เพื่อการเฝ้าดูพอร์ตบางพอร์ตเป็นกรณีพิเศษ เพราะโปรแกรมกวาดดูพอร์ตบางโปรแกรมก็ไม่ได้กวาดดูพอร์ตทุกพอร์ตที่มี แต่จะเลือกกวาดดูเฉพาะพอร์ตที่ให้บริการที่สำคัญเท่านั้น(well-known port) เช่น เทลเน็ต เอฟทีพี เป็นต้น
- ระยะเวลาระหว่างแพ็กเก็ตที่ติดกันซึ่งมาจากหมายเลขไอพีของเครื่องต้นทางเดียวกัน(Time between Packet)

ในโปรแกรมได้กำหนดค่านี้ไว้ 1 วินาที ซึ่งผู้ใช้สามารถเปลี่ยนแปลงค่านี้ได้ตามความเหมาะสมกับสภาพของเครือข่าย ซึ่งถ้ากำหนดค่านี้ไว้น้อยเกินไปจะทำให้อาจพลาดการตรวจจับได้ แต่ถ้ามากเกินไปก็อาจจะทำให้ได้รับแจ้งว่ามีการบุกรุกจากโฮสต์ที่ไม่ได้บุกรุก

- จำนวนครั้งในการสถาปนาการเชื่อมต่อ(Number of Packet)
  ในโปรแกรมได้กำหนดค่านี้ไว้ 20 แพ็กเก็ต เงื่อนไขนี้จะคล้ายกับระยะเวลาระหว่างแพ็กเก็ตที่ติดกัน คือ ผู้ใช้สามารถเปลี่ยนค่าได้ตามสภาพของเครือข่าย แต่ควรจะเป็นค่าที่ไม่มากและไม่น้อยจนเกินไป
- จำนวนพอร์ตของเครื่องปลายทางที่ถูกเชื่อมต่อซึ่งตรงกับหมายเลขพอร์ตที่กำหนดไว้ในแฟ้มคอนฟิกูเรชัน(Number of Destination Port)
  แพ็กเก็ตที่เข้ามายังพอร์ตที่กำหนดเหล่านี้จะถูกเฝ้ามองเป็นกรณีพิเศษ เพราะจะเป็นพอร์ตที่มีความน่าสนใจเป็นพิเศษ

3.9 รายละเอียดของส่วนวิเคราะห์ตรวจจับการกวาดดูทีซีพีพอร์ต(Flowchart of TCPPSD)



รูปที่ 3-10 แสดงรายละเอียดของส่วนวิเคราะห์ตรวจจับการกวาดดูทีซีพีพอร์ต

จากรูปที่ 3-10 อธิบายเป็นขั้นตอนได้ดังนี้

- เริ่มการทำงานของโปรแกรม TCPPSD
- อ่านค่าที่กำหนดจากแฟ้มคอนฟิกูเรชัน(TCPPSD.CONF) ได้แก่

หมายเลขไอพีของโฮสต์ที่ไว้ใจ(TRUSTEDHOST)

อีเมล์ของผู้ดูแลระบบ(MAILTO)

ระยะเวลาของสองแพ็กเก็ตที่ตามกันมา(TIMEINTERVAL)

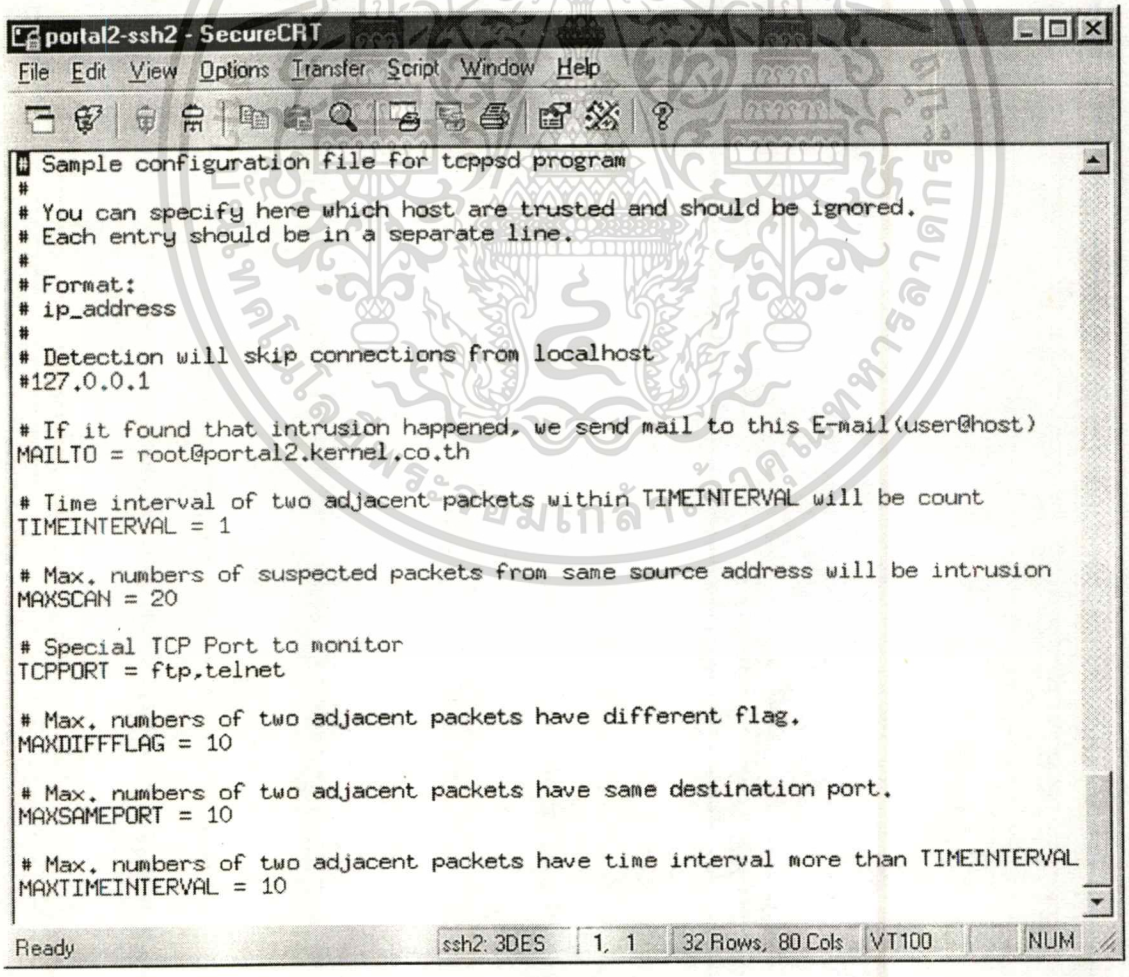
ค่ามากสุดของจำนวนแพ็กเก็ตที่สงสัย(MAXSCAN)

พอร์ตที่ต้องการให้เฝ้าดูเป็นพิเศษ(TCPPORT)

ค่ามากสุดที่สองแพ็กเก็ตที่ตามกันมาจะมีทีซีพีแฟล็กต่างกันได้(MAXDIFFFLAG)

ค่ามากสุดที่สองแพ็กเก็ตที่ตามกันมาจะมีพอร์ตปลายทางเหมือนกันได้(MAXSAMEPORT)

ค่ามากสุดที่สองแพ็กเก็ตที่ตามกันมาจะมีระยะเวลาเกิน TIMEINTERVAL ได้(MAXTIMEINTERVAL)



รูปที่ 3-11 แสดงตัวอย่างของแฟ้มคอนฟิกูเรชันของโปรแกรม TCPPSD

- รับแพ็กเก็ตจากเครือข่ายที่ส่งมาถึงโฮสต์ที่มีโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตทำงานอยู่

- อ่านเวลาที่ได้รับแพ็กเก็ต

- กรองเฉพาะแพ็กเก็ตที่มีทีซีพีแฟล็กตรงตามรูปแบบที่สนใจตรวจจับได้แก่ 0x00(NULL Scan) 0x01(Fin Scan) 0x02(TCP Connect() or TCP SYN Scan) และ 0x29(Xmas Scan)

- ตรวจดูว่าเคยได้รับแพ็กเก็ตจากหมายเลขไอพีต้นทางนี้แล้วหรือยัง

- ถ้ายังไม่เคยได้รับจากหมายเลขไอพีต้นทางนี้ก็ทำการกำหนดโครงสร้างในหน่วยความจำเพื่อเก็บข้อมูลที่สำคัญ(Features)ไว้วิเคราะห์ต่อไป

- วนกลับขึ้นไปรับแพ็กเก็ตจากเครือข่ายอีก

- ถ้าเคยได้รับแพ็กเก็ตจากหมายเลขไอพีต้นทางนี้แล้ว ก็ทำการค้นหา index ของโครงสร้างนั้นจากหน่วยความจำ

- ทำการเปรียบเทียบเวลาของแฟล็กของแพ็กเก็ตที่แล้วกับแพ็กเก็ตปัจจุบันว่าตรงกันหรือไม่

- ถ้าตรงก็ทำการเปรียบเทียบต่อ โดยดูพอร์ตปลายทางของแพ็กเก็ตที่แล้วกับแพ็กเก็ตปัจจุบันว่าต่างกันหรือไม่

- ถ้าต่างกันก็ทำการเปรียบเทียบต่อ โดยดูระยะเวลาของแพ็กเก็ตที่แล้วกับแพ็กเก็ตปัจจุบันว่าตามกันมาอยู่ภายในช่วงเวลา TIMEINTERVAL หรือไม่

- ถ้าอยู่ในช่วงเวลาที่กำหนดอีก ก็นับแพ็กเก็ตนั้นเป็นแพ็กเก็ตที่ต้องสงสัย(count)

- ถ้าจำนวนแพ็กเก็ตที่ต้องสงสัยเท่ากับ MAXSCAN ก็จะทำการแสดงผลลัพธ์ผ่านทางหน้าจอ เก็บลงแฟ้มล็อก และส่งเมล์เพื่อแจ้งให้ผู้ดูแลระบบทราบจะได้หาทางป้องกันอันตรายที่อาจเกิดขึ้นกับระบบต่อไป

- นอกจากนี้ถ้ามีการกำหนดพอร์ตที่ให้เฝ้าดูเป็นพิเศษแล้ว ถ้าแพ็กเก็ตมีพอร์ตปลายทางตรงตามที่ระบุไว้ในพอร์ตเฝ้าดูเป็นพิเศษ ก็สามารถนับจำนวนแพ็กเก็ตที่สงสัยได้เหมือนกัน(portmatch)

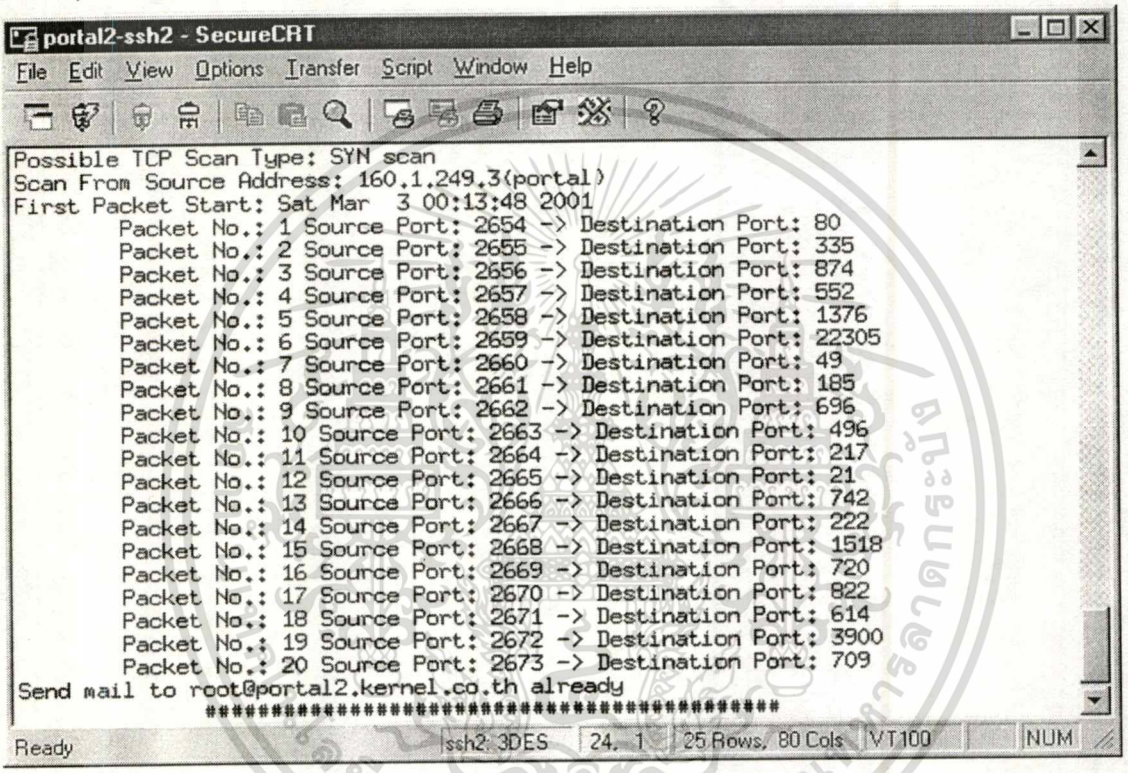
- ถ้าจำนวนแพ็กเก็ตที่ต้องสงสัยมีค่าเท่ากับค่าที่กำหนดไว้ ก็จะทำการแจ้งเตือนหรือถ้าจำนวนแพ็กเก็ตที่ไม่สงสัยมีค่าเท่ากับ MAXDIFFFLAG, MAXSAMEPORT, MAXTIMEINTERVAL แล้วก็จะทำการคืนค่า(deallocate)ของโฮสต์นั้นให้กับหน่วยความจำ เพื่อเก็บข้อมูลเกี่ยวกับโฮสต์อื่นที่อาจจะทำการกวาดดูทีซีพีพอร์ตต่อไป

- จากนั้นก็จะวนกลับไปรับแพ็กเก็ตต่อไป

- การทำงานของโปรแกรมจะยกเลิกก็ต่อเมื่อผู้ดูแลระบบทำการยกเลิก(Ctrl+C)การทำงานของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

3.10 การรายงานผลการวิเคราะห์(Display Result)

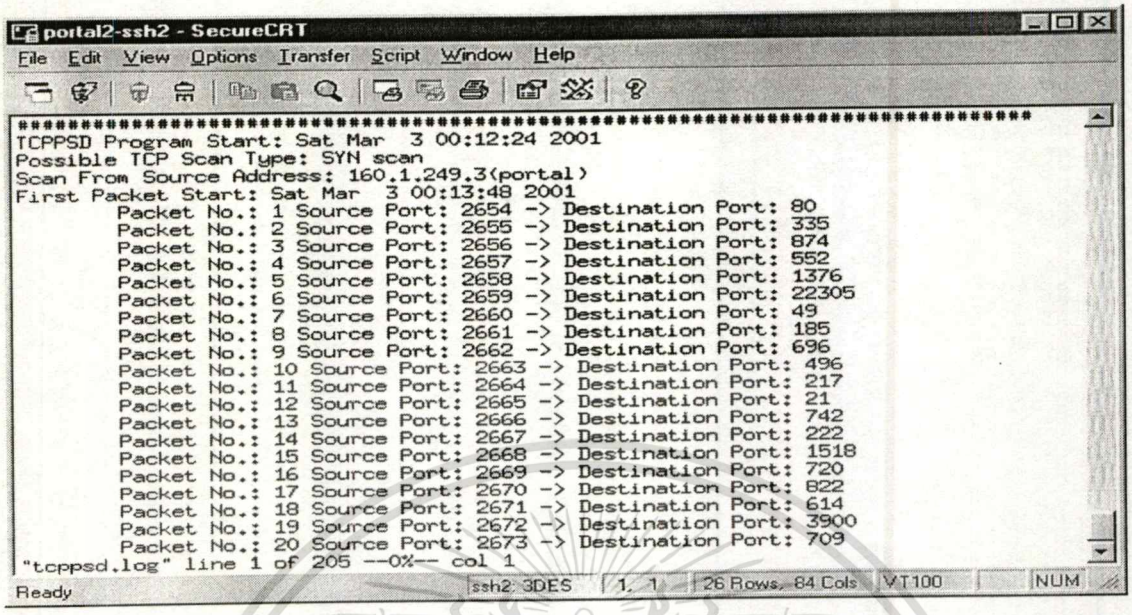
เมื่อโปรแกรมวิเคราะห์แพ็กเก็ตแล้วพบว่าเป็นการบุกรุกโดยวิธีการกวาดดูทีซีพีพอร์ต โปรแกรมจะรายงานผลการวิเคราะห์ให้ผู้ดูแลระบบทราบ โดยจะแสดงผลทางหน้าจอพร้อมทั้งเก็บบันทึกไว้ในรูปของแฟ้มล็อก นอกจากนี้ยังสามารถกำหนดให้โปรแกรมรายงานผลผ่านทางอิเล็คทรอนิคเมล์ได้ รูปแบบของการรายงานผลการวิเคราะห์ที่จะแสดงให้ผู้ดูแลระบบทราบจะเป็นดังรูปที่ 3-12, 3-13 และ 3-14 ตามลำดับ



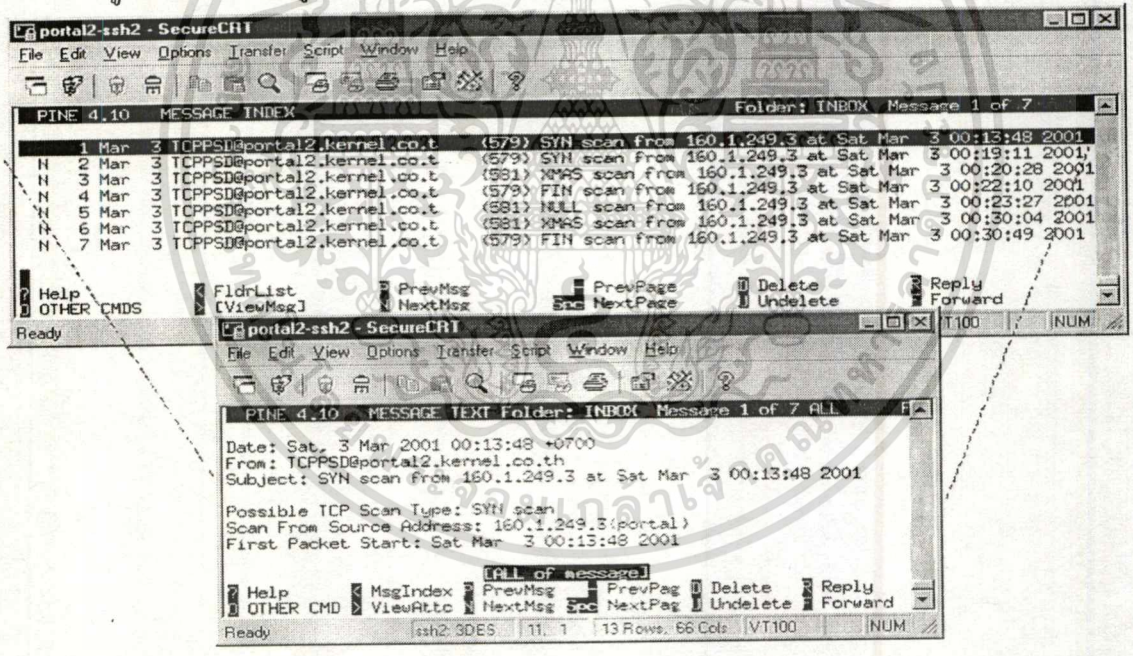
รูปที่ 3-12 แสดงรูปแบบของการรายงานผลการวิเคราะห์ผ่านทางหน้าจอ

จากรูปแบบการรายงานผลการวิเคราะห์จะเห็นว่าจะมีการแจ้งให้ผู้ดูแลระบบทราบถึงแฟล็กที่ใช้ในการกวาดดูทีซีพีพอร์ต ชื่อเครื่องที่กวาดดูทีซีพีพอร์ต เวลาที่เริ่มกวาดดูทีซีพีพอร์ต และ พอร์ตต้นทางกับพอร์ตปลายทางในช่วงที่เก็บข้อมูลไว้

ในส่วนของการรายงานผลโดยเก็บบันทึกในแฟ้มล็อกนั้นจะมีรูปแบบเหมือนกับการรายงานผลผ่านทางหน้าจอเช่นกัน แต่จะเพิ่มการบันทึกเวลาเริ่มต้นในการรันโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตไว้ด้วย



รูปที่ 3-13 แสดงรูปแบบของการรายงานผลการวิเคราะห์ผ่านทาง TCPPSD.LOG



รูปที่ 3-14 แสดงรูปแบบของการรายงานผลการวิเคราะห์ผ่านทางอิเล็คทรอนิคเมล์

บทที่ 4

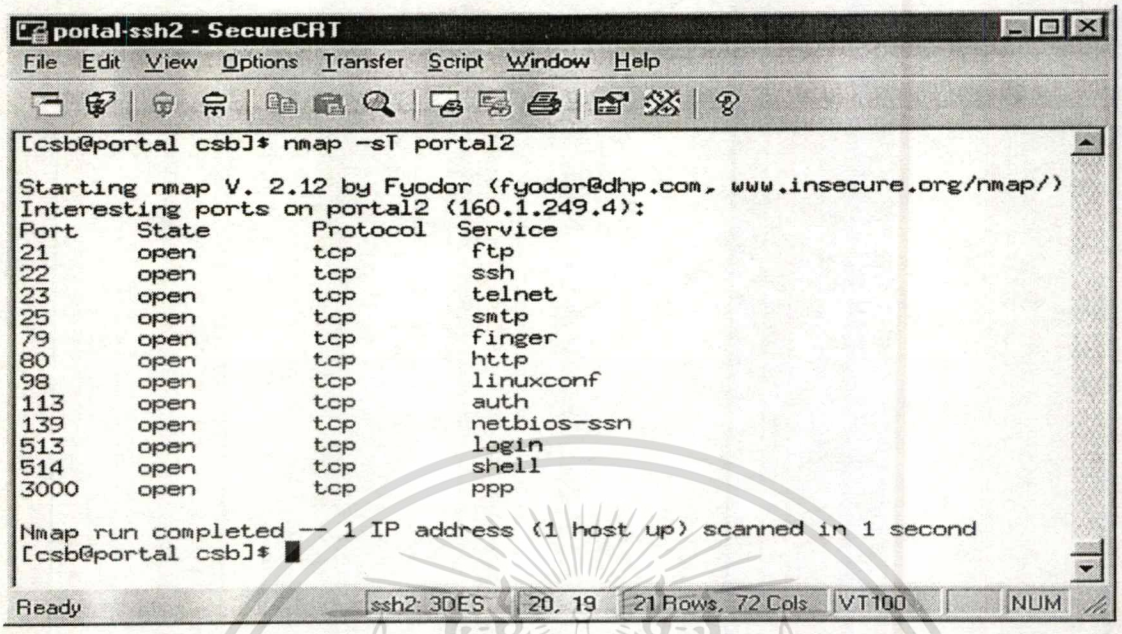
การทดสอบผลการทำงานของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

โปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตจะสามารถทำงานได้ถูกต้องเพียงใดขึ้นอยู่กับปัจจัยต่างๆ เช่น การกำหนดค่าต่างๆ ที่ใช้ในการตัดสินใจให้เหมาะสม กับสภาพแวดล้อมของเครือข่าย หน่วยความจำหลักของเครื่องคอมพิวเตอร์ที่รันโปรแกรม เป็นต้น โดยในการทดสอบโปรแกรมได้ทดสอบในสภาวะที่เครื่องไม่ได้มีการให้บริการอื่นๆ คือ เป็นเครื่องที่ทำงานเฉพาะโปรแกรมตรวจจับการบุกรุกแบบการกวาดดูทีซีพีพอร์ตและค่าที่กำหนดไว้ได้กำหนดให้วิเคราะห์ว่าเมื่อมีแพ็กเก็ตจากเครื่องหนึ่งที่เข้ามายังเครื่องที่รันโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตเป็นจำนวน 20 แพ็กเก็ต โดยแต่ละแพ็กเก็ตมีระยะห่างกันไม่เกิน 1 วินาที จะถือว่าเป็นการบุกรุกด้วยวิธีการกวาดดูทีซีพีพอร์ต

หลังจากที่ได้รันโปรแกรม tcppsd(TCP Port Scanning Detection Program) และทดสอบด้วยโปรแกรมกวาดดูทีซีพีพอร์ต NMAP พบว่าโปรแกรมสามารถตรวจจับการบุกรุกผ่านการกวาดดูทีซีพีพอร์ตได้ดังต่อไปนี้

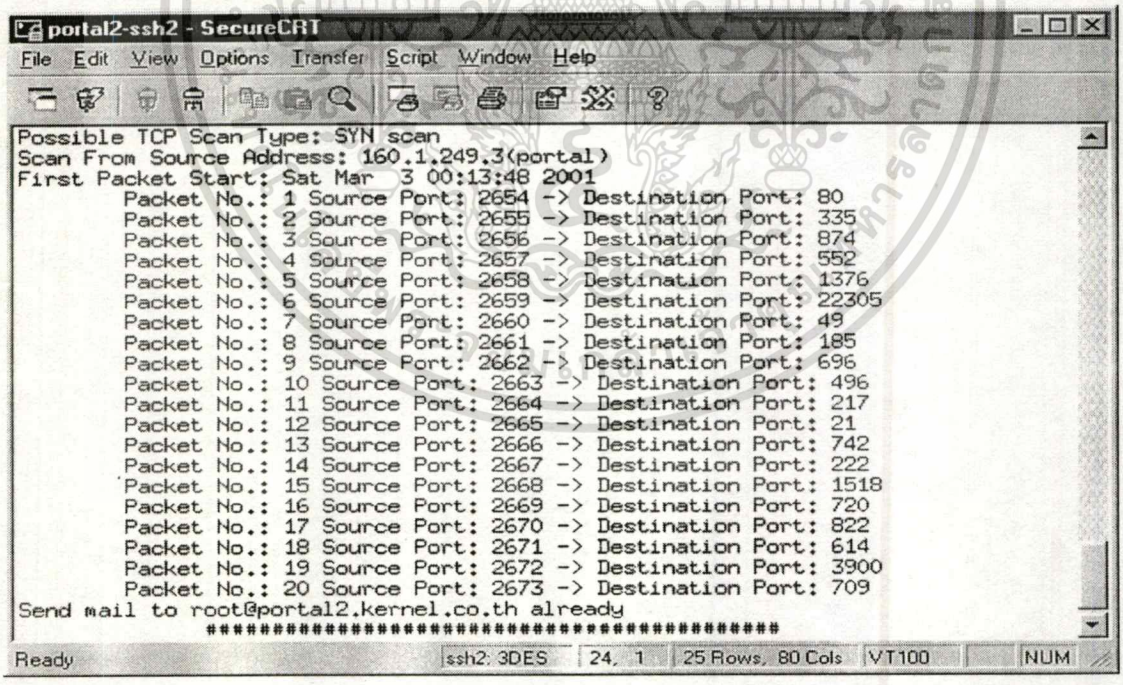
4.1 ตรวจจับการกวาดดูแบบ TCP connect()

4.1.1 การกวาดดูแบบ TCP connect() โดยใช้โปรแกรม NMAP ดังรูปที่ 4-1



รูปที่ 4-1 แสดงการกวาดดูแบบ TCP connect() โดยใช้ NMAP(ไม่ต้องเป็น root ก็รันคำสั่งได้)

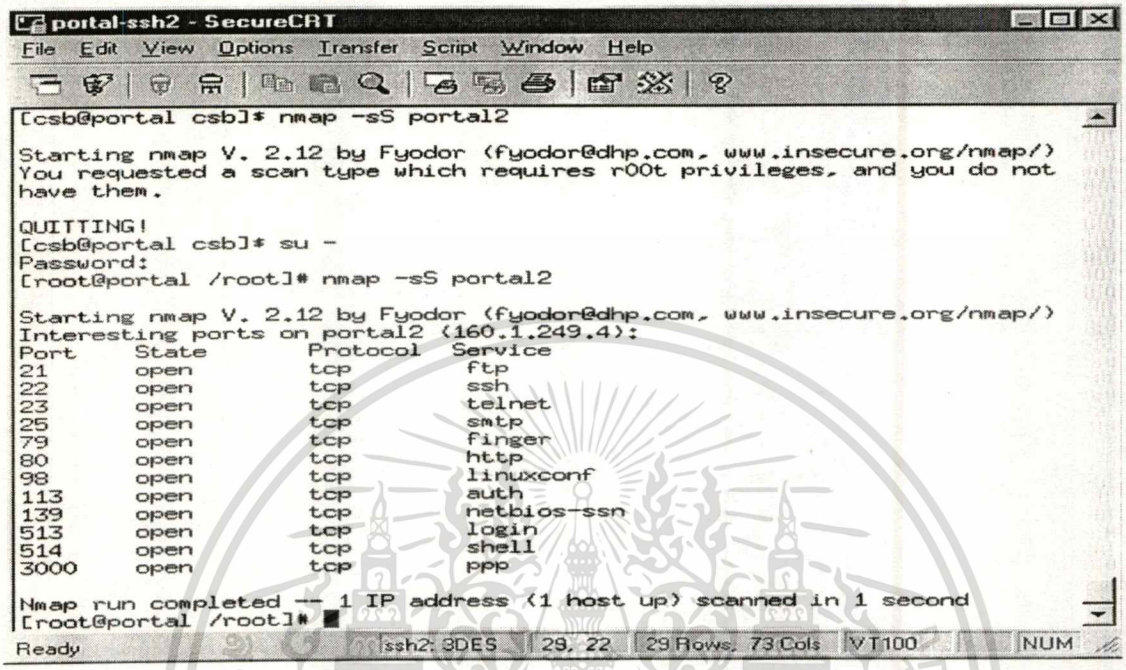
4.1.2 การตรวจจับการกวาดดูแบบ TCP connect() ดังรูปที่ 4-2



รูปที่ 4-2 แสดงการตรวจจับการกวาดดูแบบ TCP connect()

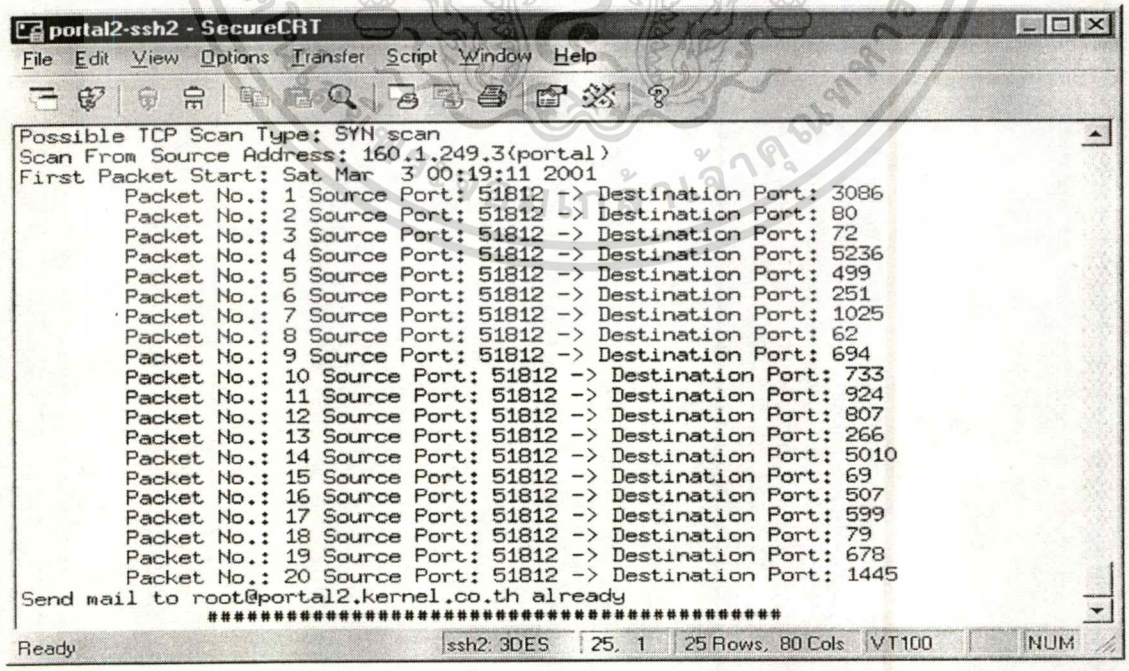
4.2 ตรวจจับการกวาดดูแบบ TCP SYN

4.2.1 การกวาดดูแบบ TCP SYN โดยใช้โปรแกรม NMAP ดังรูปที่ 4-3



รูปที่ 4-3 แสดงการกวาดดูแบบ TCP SYN(ต้องเป็น root ก่อนจึงจะรันคำสั่งได้)

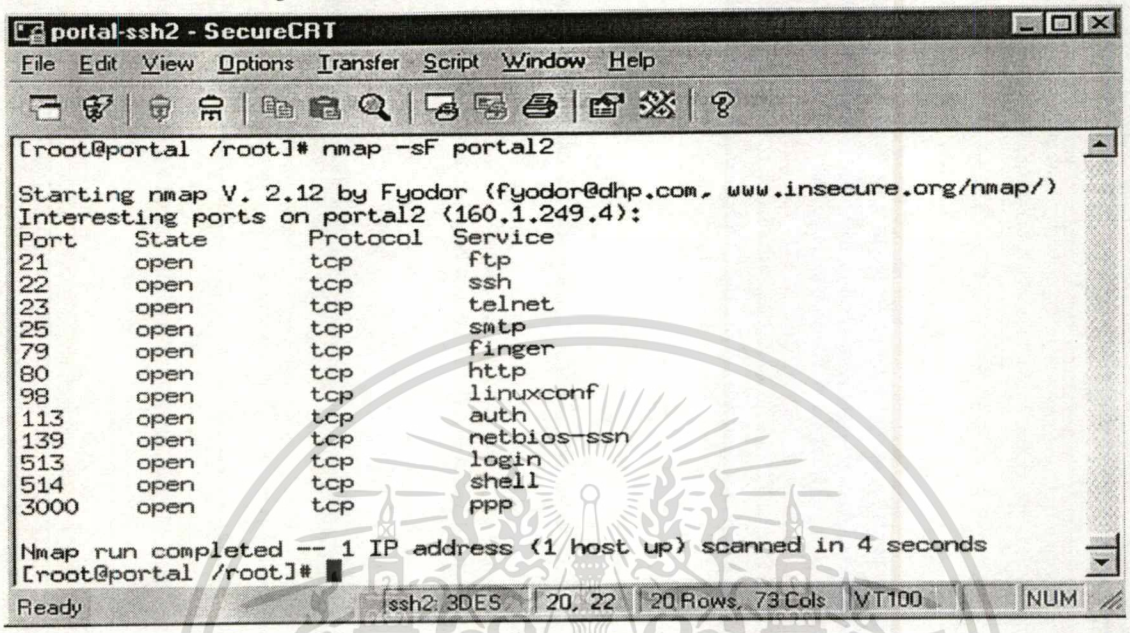
4.2.2 การตรวจจับการกวาดดูแบบ TCP SYN ดังรูปที่ 4-4



รูปที่ 4-4 แสดงการตรวจจับการกวาดดูแบบ TCP SYN

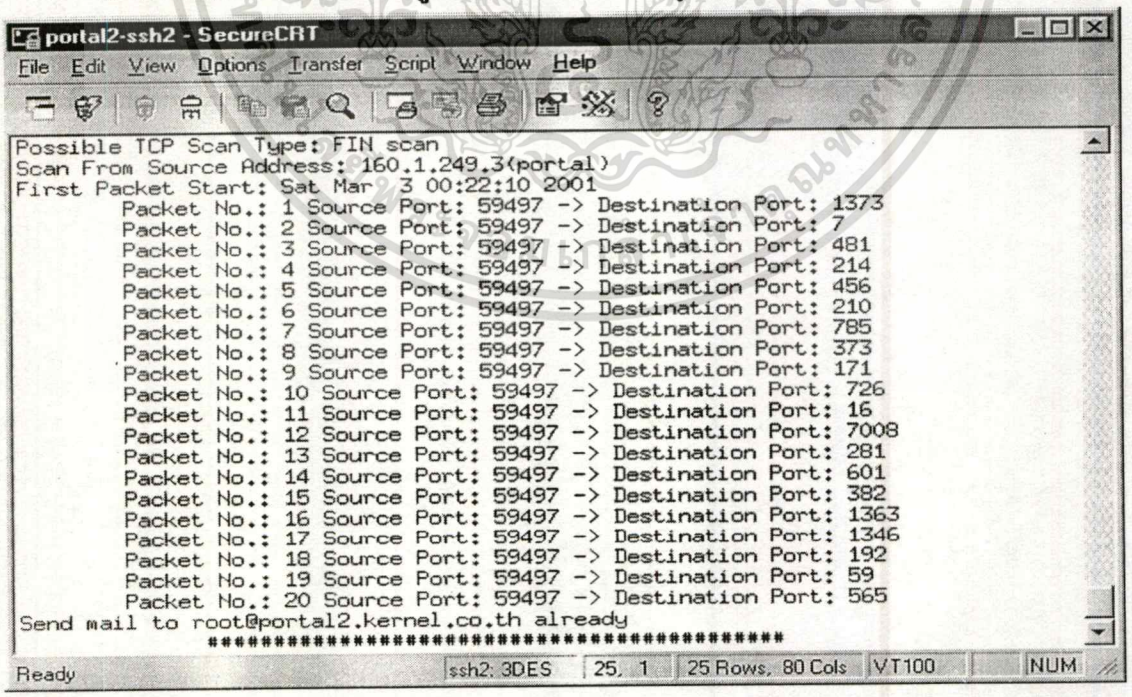
4.3 ตรวจจับการกวาดดูแบบ Stealth FIN

4.3.1 การกวาดดูแบบ Stealth FIN โดยใช้โปรแกรม NMAP ดังรูปที่ 4-5



รูปที่ 4-5 แสดงการกวาดดูแบบ Stealth FIN(ต้องเป็น root ก่อนจึงจะรันคำสั่งได้)

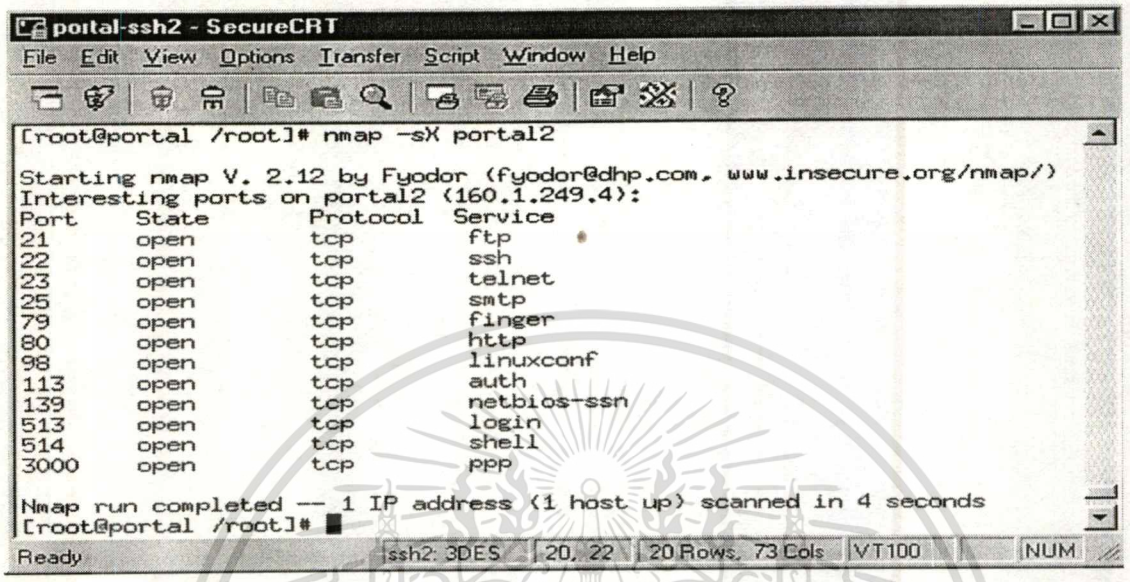
4.3.2 การตรวจจับการกวาดดูแบบ Stealth FIN ดังรูปที่ 4-6



รูปที่ 4-6 แสดงการตรวจจับการกวาดดูแบบ Stealth FIN

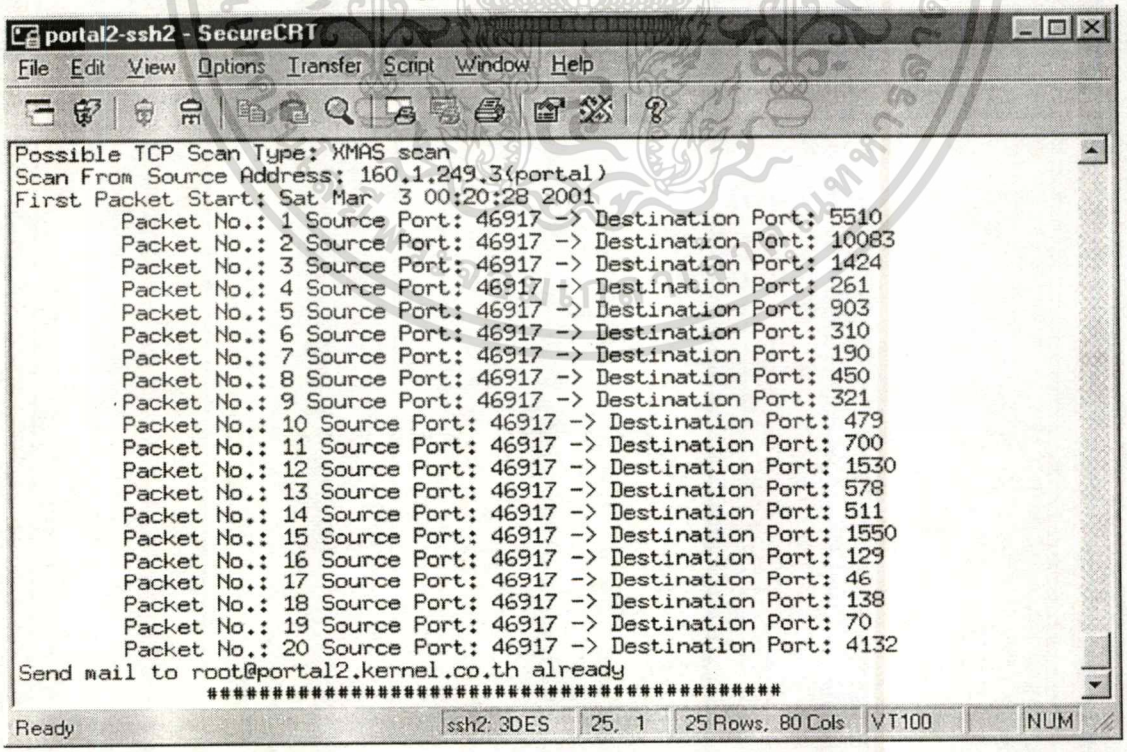
4.4 ตรวจจับการกวาดดูแบบ Stealth Xmas

4.4.1 การกวาดดูแบบ Stealth Xmas โดยใช้โปรแกรม NMAP ดังรูปที่ 4-7



รูปที่ 4-7 แสดงการกวาดดูแบบ Stealth Xmas(ต้องเป็น root ก่อนจึงจะรันคำสั่งได้)

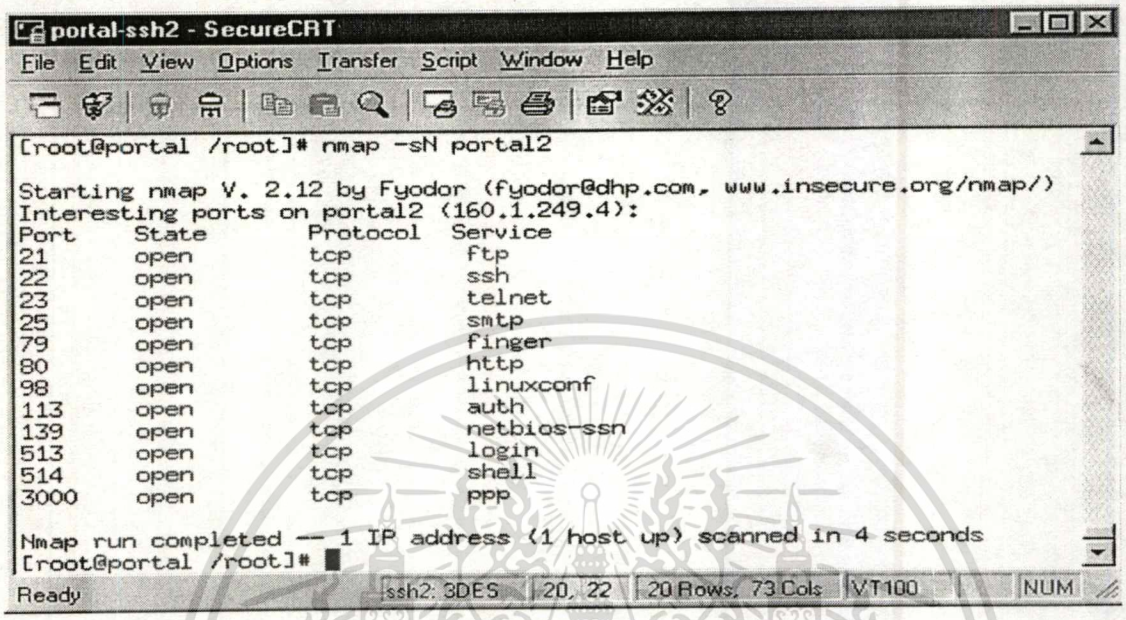
4.4.2 การตรวจจับการกวาดดูแบบ Stealth Xmas ดังรูปที่ 4-8



รูปที่ 4-8 แสดงการตรวจจับการกวาดดูแบบ Stealth Xmas

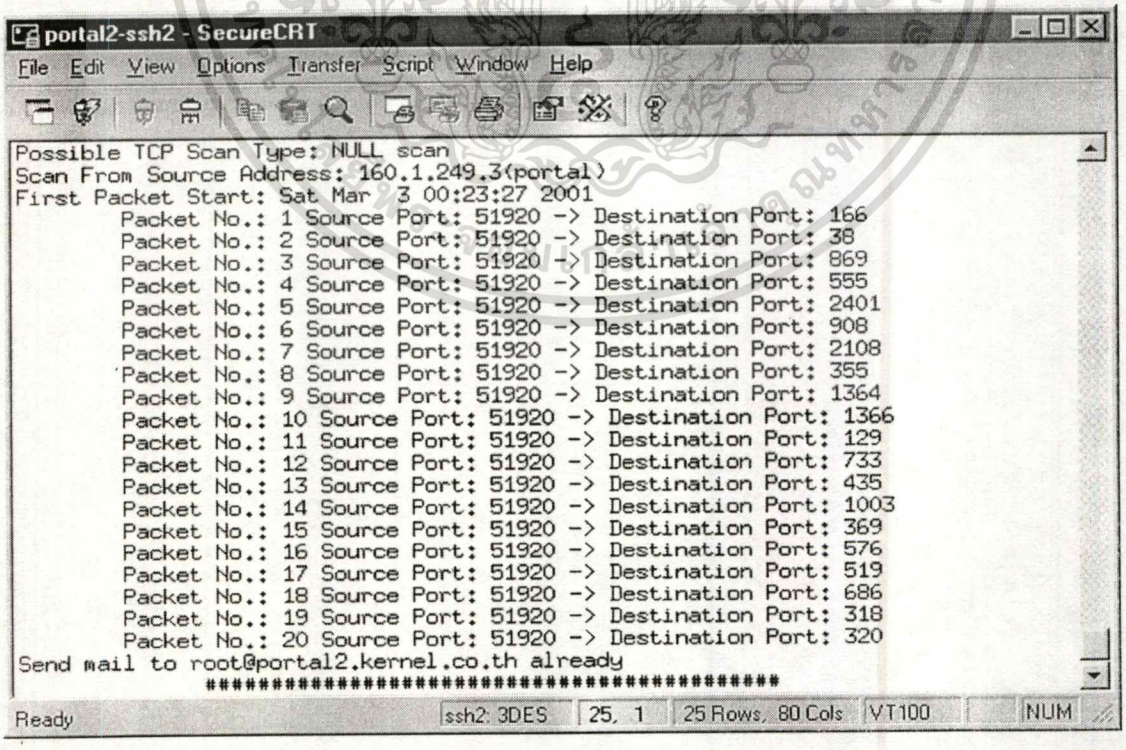
4.5 ตรวจจับการกวาดดูแบบ Stealth NULL

4.5.1 การกวาดดูแบบ Stealth NULL โดยใช้โปรแกรม NMAP ดังรูปที่ 4-9



รูปที่ 4-9 แสดงการกวาดดูแบบ Stealth NULL(ต้องเป็น root ก่อนจึงจะรันคำสั่งได้)

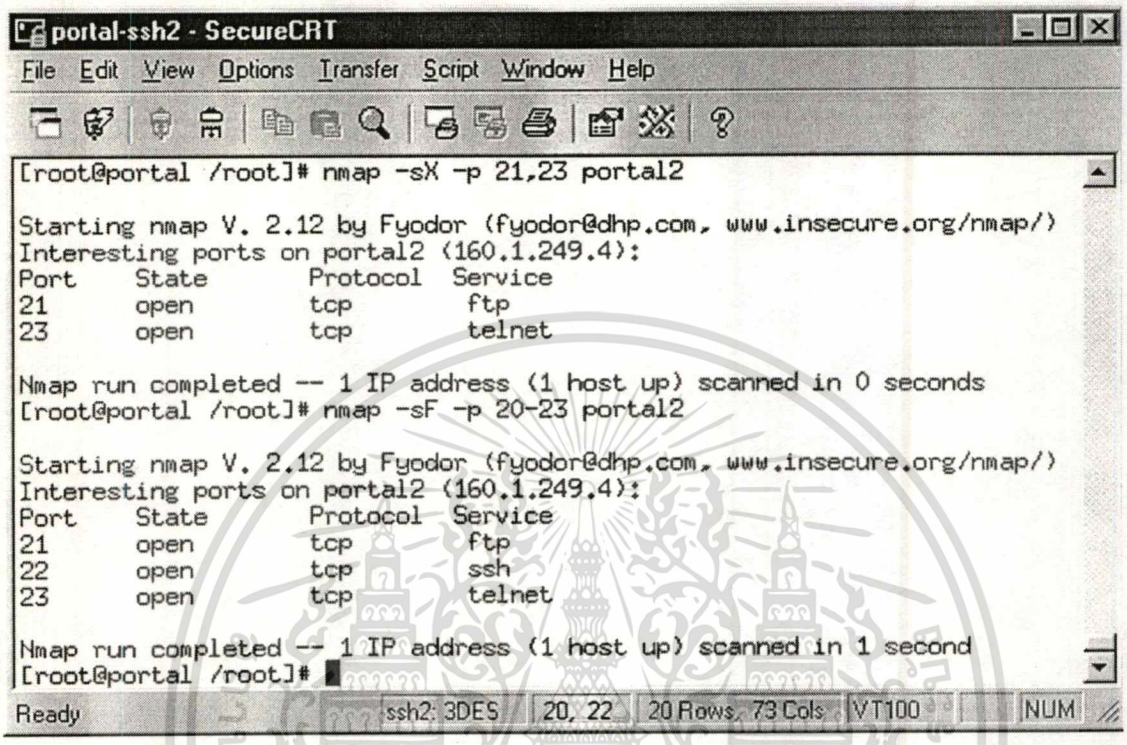
4.5.2 การตรวจจับการกวาดดูแบบ Stealth NULL ดังรูปที่ 4-10



รูปที่ 4-10 แสดงการตรวจจับการกวาดดูแบบ Stealth NULL

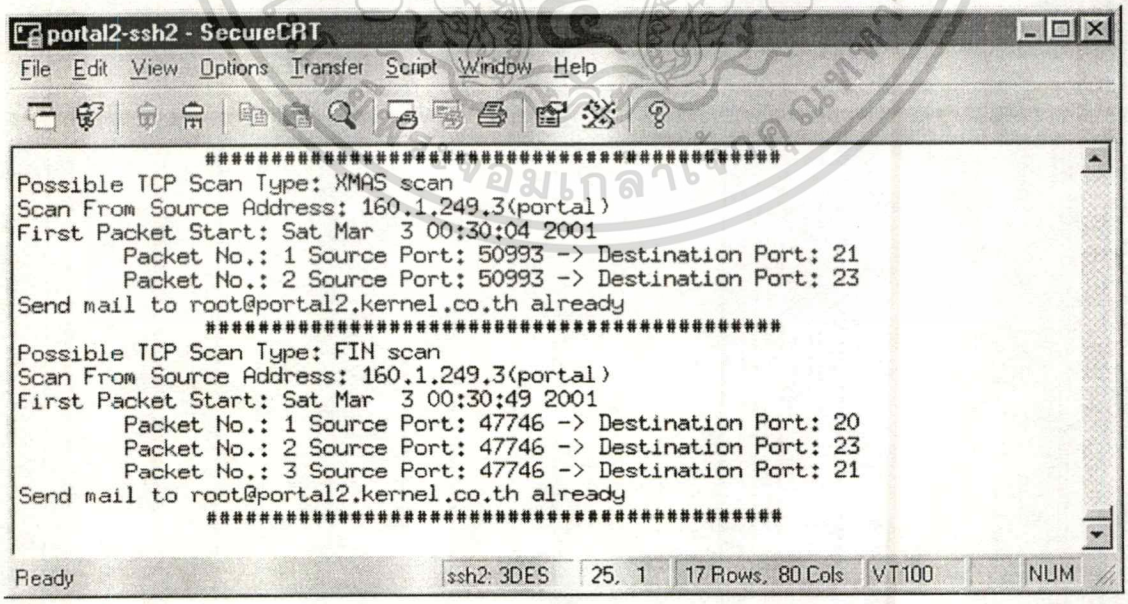
4.6 ตรวจจับการกวาดดูแบบที่เลือกกวาดดูบางพอร์ต

4.6.1 การกวาดดูแบบที่เลือกกวาดดูบางพอร์ต โดยใช้โปรแกรม NMAP ดังรูปที่ 4-11



รูปที่ 4-11 แสดงการกวาดดูแบบที่เลือกกวาดดูบางพอร์ต

4.6.2 การตรวจจับการกวาดดูแบบที่เลือกกวาดดูบางพอร์ต ดังรูปที่ 4-12



รูปที่ 4-12 แสดงการตรวจจับกวาดดูแบบที่เลือกกวาดดูบางพอร์ต

4.7 ข้อจำกัดของโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ต

โปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตสร้างขึ้นโดยมีข้อจำกัดของการใช้งานโปรแกรมดังต่อไปนี้

4.7.1 โปรแกรมนี้สร้างขึ้นบนระบบปฏิบัติการลีนุกซ์ RedHat 7.0 ซึ่งอาจแตกต่างกันหากนำไปใช้บนระบบอื่น

4.7.2 ผู้ที่มีสิทธิ์รูท(root) เท่านั้นที่จะสามารถใช้งานโปรแกรมนี้ได้

4.7.3 โปรแกรมนี้สามารถตรวจจับการกวาดดูทีซีพีพอร์ตได้เฉพาะเครื่องที่รันโปรแกรมนี้อยู่เท่านั้น

4.7.4 .ในกรณีที่เครือข่ายนั้นแพ็กเก็ตปริมาณมาก อาจทำให้บัฟเฟอร์ที่ใช้เก็บข้อมูลของแพ็กเก็ตเต็มก่อนนำมาวิเคราะห์ได้

บทที่ 5
บทสรุปและแนวทางการพัฒนาต่อ

5.1 ประโยชน์ที่ได้รับจากโครงการพัฒนาระบบงาน

5.1.1 สามารถตรวจจับการบุกรุกระบบโดยการกวาดดูทีซีพีพอร์ตในแบบต่างๆ ได้แก่

- TCP connect() scanning
- TCP SYN scanning
- Stealth FIN scanning
- Stealth Xmas scanning
- Stealth NULL scanning

ทำให้ได้รู้ว่าเกิดการบุกรุกจากเครื่องใด เพื่อแจ้งให้ผู้ดูแลระบบทราบและหามาตรการป้องกันอันตรายที่อาจเกิดขึ้นได้ในอนาคตได้อย่างทันท่วงที

5.1.2 ให้ความสะดวกแก่ผู้ดูแลระบบเครือข่าย เนื่องจากไม่ต้องคอยตรวจสอบแฟ้มล็อกเพื่อดูพฤติกรรมที่ผิดปกติซึ่งอาจเกิดจากการบุกรุกด้วยวิธีการกวาดดูทีซีพีพอร์ต

5.2 ปัญหาและอุปสรรค

ในการพัฒนาโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตนี้ มีปัญหาและอุปสรรคในการพัฒนาหลายประการ ได้แก่

5.2.1 โปรแกรมที่ใช้ในการกวาดดูพอร์ตมีมากมาย โดยใช้เทคนิควิธีการกวาดดูพอร์ตที่มีความสลับซับซ้อนมาก การศึกษาการทำงานของโปรแกรมเหล่านั้นและการนำมาจัดแบ่งประเภททำได้ยากและใช้เวลานาน

5.2.2 ยังไม่สามารถตรวจสอบการโจมตีในชั้นแอพพลิเคชันได้

5.3 แนวทางการพัฒนาต่อ

เนื่องจากโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตที่พัฒนาขึ้นมานี้ สามารถตรวจจับผู้บุกรุกได้แค่ระดับที่ทราบว่ามีการโจมตีระบบ โดยการส่งแพ็กเก็ตในชั้นอินเตอร์เน็ตและทรานสปอร์ตและโปรแกรมที่สร้างขึ้นเพื่อใช้งานบนระบบปฏิบัติการลีนุกซ์เท่านั้น การพัฒนาโปรแกรมในโครงการพัฒนาระบบงานนี้เป็นการสร้างโปรแกรมตรวจจับการกวาดดูทีซีพีพอร์ตเฉพาะบนสถานี

งานเพียงรูปแบบเดียว คือ การกวาดดูทีซีพีพอร์ต ซึ่งเป็นรูปแบบที่พบได้บ่อยและมีการทำงานที่ง่าย ไม่สลับซับซ้อน แต่ในการบุกรุกระบบเครือข่ายคอมพิวเตอร์นั้นยังมีรูปแบบการบุกรุกอื่นๆอีกมากมาย และหลักการจริงๆของการกวาดดูพอร์ต นั้นคือ การหาจุดอ่อนเพื่อที่จะนำข้อมูลที่ได้ไปเป็นแนวทางในการบุกรุกโดยใช้รูปแบบของการบุกรุกที่ยุ่งยากและซับซ้อนมากขึ้น ดังนั้นแนวทางการพัฒนาต่อสามารถทำได้หลายรูปแบบ เช่น

(1) การเพิ่มเติมรูปแบบการบุกรุกอื่นๆ เพื่อให้ระบบเครือข่ายมีความปลอดภัยมากยิ่งขึ้น

(2) อาจจะพัฒนาต่อให้สามารถตรวจจับการบุกรุกได้ทั้งเครือข่าย(Network-based Intrusion Detection) แต่ไม่ว่าจะพัฒนาไปในรูปแบบใดก็หมายถึงการทำงานที่เพิ่มมากขึ้นและสลับซับซ้อนมากยิ่งขึ้น ดังนั้นสิ่งที่จะต้องคำนึงอยู่เสมอ คือ โปรแกรมตรวจจับการบุกรุกจะต้องสามารถตรวจจับการบุกรุกที่เข้ามายังเครื่องตัวเองได้ และต้องมีประสิทธิภาพในการทำงานได้อย่างถูกต้องรวดเร็ว ไม่มีจุดอ่อนที่ผู้บุกรุกจะสามารถใช้เพื่อบุกรุกเข้ามาได้ เพราะโปรแกรมตรวจจับการบุกรุกที่มีจุดอ่อนจะเป็นเครื่องมือที่สำคัญสำหรับผู้บุกรุกที่จะใช้ในการบุกรุกเครื่องคอมพิวเตอร์เครื่องอื่นๆต่อไป

(3) สร้างระบบป้องกันขึ้นด้วย กล่าวคือ เมื่อมีการแจ้งเตือนก็สามารถกันไม่ให้รับแพ็กเก็ตที่มาจากหมายเลขไอพีต้นทาง(Source IP) หรือ แม็คแอดเดรสต้นทาง(MAC Address)นั้นได้ โดยให้ไปตั้งค่าที่ไฟร์วอลล์(Firewall) หรือ สวิตช์ชิ่ง(Switching) ให้โดยอัตโนมัติ

(4) ทำให้สามารถตรวจสอบได้ว่าผู้บุกรุกเป็นใคร โดยเมื่อมีความผิดปกติเกิดขึ้นให้ Trace กลับไปยังเครื่องที่โจมตีมาได้

(5) เพิ่มความละเอียดของการวิเคราะห์ไปถึงชั้นเน็ตเวิร์กอินเตอร์เฟส คือ มีการนำแม็คแอดเดรสมาพิจารณาด้วย

(6) เพิ่มความสามารถในการตรวจจับให้มากขึ้นได้ โดยการเพิ่มขอบเขตของการตรวจจับให้กว้างขึ้น (ทำเป็น Distributed)

(7) สร้างให้ระบบสามารถทำงานบนแพล็ตฟอร์มอื่นได้ เช่น Windows 9x, Windows NT, Windows 2000 เป็นต้น

นอกจากนี้ยังสามารถพัฒนาโปรแกรมให้มีส่วนติดต่อกับผู้ใช้ให้เป็นแบบกราฟฟิก เพื่อเพิ่มความสะดวกในการใช้งานให้กับผู้ดูแลระบบ

บรรณานุกรม

ชูศักดิ์ บุญญศิริวัฒน์. 2543. "*ระบบตรวจจับการบุกรุกแบบกระจาย*", สัมมนา 2, 2 ต.ค. , หน้า 2-5

สุรศักดิ์ สงวนพงษ์. 2543. "*สถาปัตยกรรมและโปรโตคอลทีซีพี/ไอพี*": ซีเอ็ดยูเคชั่น.

David A. Curry. 1991. "*C on the UNIX System*": O'Reilly&Associates,Inc., February

Fyodor. 1997. "*The Art of Port Scanning*", http://www.insecure.org/nmap/nmap_doc.html, September 6

Fyodor. 2000. "*Nmap Free Security Scanner*", http://www.insecure.org/nmap/index.html, January

Kanlayasiri U., Sanguanpong S.,and Jaratmanachot W. 2000. "*A Rule-based Approach for Port Scanning Detection*": Electrical Engineering Conference Thailand

Ofir Arkin. 1999. "*Network Scanning Techniques*", http://www.sys-security.com/archive/paper/Network_Scanning_Techniques.pdf, November

Richard W. Stevens. 2000. "*Volume1 Networking APIs: Sockets and XTI*", UNIX Network Programming second edition: Addison Wesley Longman Singapore Pte Ltd.

ภาคผนวก

A-1. ซอร์สโค้ด(Source Code) ส่วนเป็นแฟ้มเฮดเดอร์(Header File)

```
/*************************************************************************
 * Written by    : Choosak Bunyasiriwat
 * File          : globaldef.h
 * Description   : global definition of tcppsd program
 * Last Modify   : Mar 4,2001
 *************************************************************************/
#ifndef _GLOBALDEF_H
#define _GLOBALDEF_H 1

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/utsname.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <time.h>

/*
 * TCPPSD's Configuration File
 */
#define CONFIGFILENAME "tcppsd.conf"
```

```
/*
 * TCPPSD's Log File
 */
#define LOGFILENAME "tcppsd.log"
char * TRUSTEDHOST;
/*
 * If it found that intrusion happened, we send mail to this E-mail:<user@host>
 * (you can change its value in TCPPSD's Configuration)
 */
char * MAILTO;
/*
 * Time interval of two adjacent packets within TIME_INTERVAL will be count
 * Default TIMEINTERVAL is 1 second.(you can change its value in TCPPSD's Configuration)
 */
int TIMEINTERVAL;
/*
 * Max. numbers of suspected packets from same source address will be intrusion
 * Default MAXSCAN is 20 (you can change its value in TCPPSD's Configuration)
 */
int MAXSCAN;
/*
* Max. numbers of two adjacent packets have different flag.
* Default MAXDIFFFLAG is 10(you can change its value in TCPPSD's Configuration)
*/
int MAXDIFFFLAG;
/*
* Max. numbers of two adjacent packets have same destination port.
* Default MAXSAMEPORT is 10(you can change its value in TCPPSD's Configuration)
*/
int MAXSAMEPORT;
/*
```

```
* Max. numbers of two adjacent packets have time interval more than TIMEINTERVAL
* Default MAXTIMEINTERVAL is 10(you can change its value in TCPPSD's Configuration)
*/
int MAXTIMEINTERVAL;

// Other global constant and variable
/* SMTP Port is 25 */
#define MAILPORT 25

extern int errno;
FILE * LOGFILE;
char * fromhost;
int mail;
char * scantype;
char * src_addr;
char * firstscan;

#endif /* globaldef.h */

/*****************************************************************************
 * Written by    : Choosak Bunyasiriwat
 * File          : tcppsd.h
 * Description   : header file of tcppsd.c
 * Last Modify   : Mar 4,2001
 *****************************************************************************/
#ifndef _TCPPSD_H
#define _TCPPSD_H 1
/* number of index in host structure */
int num_host_index;

extern int allocate(int *p, unsigned int addr);
```

```
extern void deallocate(int index);
extern int isPortMatch(unsigned short port);

#endif /* tcppsd.h */

/*****************************************************************************
 * Written by    : Choosak Bunyasiriwat
 * File          : readconfig.h
 * Description   : header file of readconfig.c
 * Last Modify   : Mar 4,2001
 *****************************************************************************/
#ifndef _READCONFIG_H
#define _READCONFIG_H 1
#define MAXPORT 255

char ** ignore_hosts;

int num_ignore_host;
int *monitor_ports;
int num_monitor_port;

void readconfig();
int chk_ignore(int addr);
int readportconf(char *buff,int *ports);
void add_port(char *temp,int num,int *ports);
int readhostconf(char *buff,char **hosts);

#endif /* readconfig.h */

/*****************************************************************************
 * Written by    : Choosak Bunyasiriwat
```

```
 * File              : display.h
 * Description    : header file of display.c
 * Last Modify   : Mar 4,2001
 ********************************************************************************/
#ifndef _DISPLAY_H
#define _DISPLAY_H 1

#include "packet.h"
extern char * hostlookup(int i);
extern void display(struct host hosts);
extern int send_mail(char *message);

#endif /* display.h */

/*******************************************************************************

 * Written by      : Choosak Bunyasiriwat

 * File         : packet.h
 * Description  : header file for packet structure definition
 * Last Modify   : Mar 4,2001
 ********************************************************************************/

#ifndef _PACKET_H
#define _PACKET_H 1

#include <netinet/ip.h>
#include <netinet/tcp.h>
```

```
/* max number of host to check in the same time */
#define MAX_HOST 50

/*
 * structure of incoming packet
 */
struct packet{
        struct iphdr ip;
        struct tcphdr tcp;
} pkt;

/*
 * structure of host
 */
struct host{
        unsigned int src_addr;
        time_t firsttime;
        time_t lasttime;
        int lastindex;
        unsigned short src_port[255];
        unsigned short dst_port[255];
        int count;
        u_int8_t flags[255];
        int diffflagcount;
        int sameportcount;
        int timeintervalcount;
        int portmatch;
};

struct host host[MAX_HOST];
```

```
u_int8_t currentflags;

#endif /* packet.h */

/*****************************************************************************
 * Written by     : Choosak Bunyasiriwat
 * File           : ignore.h
 * Description    : header file for ignore.c
 * Last Modify    : Mar 4,2001
 *****************************************************************************/

#ifndef _IGNORE_H
#define _IGNORE_H 1

/* max number of ignore to keep */

#define MAX_IGNORE 50

/* number of seconds to ignore host */
#define I_SEC 5

/*
 * define for return
 */
 #define NO_IGNORE 0
 #define IGNORE 1

 #define MAX_HOST_NAME 50
```

```
/*
 * structure for host that ignore keep
 */
struct ignore {
 unsigned int host;
 time_t starttime;
} ignore_host[MAX_IGNORE];

extern void add_ignore(int addr);
extern void del_ignore(int index);
extern int if_ignore(int host, int port);

#endif /* ignore.h */
```

A-2. ซอร์สโค้ด(Source Code) ส่วนที่เป็นแฟ้มหลัก(C File)

```
/*
 *
 *              TCP Port Scanning Detection Program
 *
 *              Information Technology Project
 *
 *              Author : Choosak Bunyasiriwat
 *                       Information Science(IS7.2)
 *                       Information Technology Faculty
 *                       King Mongkut's Institute of Technology Ladkrabang
 *
 */
/*****************************************************************************
 * Written by   : Choosak Bunyasiriwat
 * File         : tcppsd.c
 * Description  : detect about intrusion
```

```
* Last Modify  : Mar 4,2001
* Contains of  : int allocate(int *p, unsigned int addr)
*              : void deallocate(int index)
*              : int isPortMatch(unsigned short port)
*              : int main()
**************************************************************************/
#include "globaldef.h"
#include "tcppsd.h"
#include "display.h"
#include "ignore.h"
#include "readconfig.h"

/**************************************************************************
 * Function     : allocate(int *p, unsigned int addr)
 * Argument     : 'p' is
 *              : 'addr' is host address want to allocate in array
 * Description  : add IP address from host array
 * Return       : (int) array number of IP address
 **************************************************************************/
int allocate(int *p, unsigned int addr)
{
  int i;
  time_t now = time(0);

  for (i=0; i<MAX_HOST; i++) {
        if (host[i].src_addr == addr) {
              *p = 1;
              return i;
        }
  }
```

```
 for (i=0; i<MAX_HOST; i++) {
     if (host[i].src_addr == 0) {
                 *p = 0;
                 return i;
     }
 }

 for (i=0; i<MAX_HOST; i++) {
        if ((now - host[i].lasttime > (TIMEINTERVAL)) && (host[i].firsttime != 0)) {
              deallocate(i);
                 *p = 0;
                 return i;
        }
 }
}

/*****************************************************************************
 * Function    : deallocate(int index)
 * Argument    : 'index' is array number of IP address
 * Description : remove IP address from host array
 * Return      : (void) nothing
 *****************************************************************************/
void deallocate(int index)
{
  int i;

/* set all field to 0,which means this host array isn't in use yet */
  host[index].src_addr = 0;
  host[index].firsttime = 0;
  host[index].lasttime = 0;
  host[index].count = 0;
```

```
  host[index].lastindex = 0;
  host[index].diffflagcount = 0;
  host[index].sameportcount = 0;
  host[index].timeintervalcount = 0;
  host[index].portmatch = 0;

  for (i = 0;i < MAXSCAN;i++) {
    host[index].src_port[i] = 0;
    host[index].dst_port[i] = 0;
    host[index].flags[i] = 0;
  }
}

/*****************************************************************************
 * Function    : isPortMatch(unsigned short port)
 * Argument    : 'port' is destination port
 * Description : Check port that match in special port
 * Return      : (int) return '1'=match or '0'=not match
 *****************************************************************************/
int isPortMatch(unsigned short port) {
        int port_count;

        if (num_monitor_port != 0) {
          for (port_count = 0;port_count < num_monitor_port;port_count++) {
                if (port == monitor_ports[port_count]) {
                        return 1;
                        break;
          }
        }
          }
        return 0;
```

```
}

/*****************************************************************************
 * Function    : main()
 * Argument    : nothing
 * Description : top-level control function of program
 * Return      : (int) system return
 *****************************************************************************/
int main()
{
 int sockfd;
 int index;
 int lastindex;
 int has;
 time_t now;

/* Only root to run this program */
 if((geteuid()) && (getuid()) != 0) {
  printf("You need to be root to run this program.\n");
  exit(1);
 }

 /* Open socket to read raw packet */
 if ((sockfd = socket(AF_INET, SOCK_RAW, IPPROTO_TCP)) < 0) {
  printf("Can't open socket\n");
  exit(1);
 }

 printf("#############################################################################
 ########\n");
 /* read config file for monitoring port and ignore hosts */
```

```
  readconfig();

/* for log file */
  if ((LOGFILE = fopen(LOGFILENAME,"a")) == NULL) {
    printf ("Can't write log file\n");
  }
  else {
    now = time(0);

fprintf(LOGFILE,"##################################################################
##################\n");
    fprintf (LOGFILE,"TCPPSD Program Start: %s",ctime(&now));
    fclose(LOGFILE);
  }

  printf("Started TCP Port Scanning Detection Program(TCPPSD)...\n");

/* main loop for detection */
  while(1) {
    read(sockfd, (struct packet*) &pkt, sizeof(pkt));
    now = time(0);

    currentflags = 0;
    currentflags = pkt.tcp.urg;
    currentflags = (currentflags << 1) | pkt.tcp.ack;
    currentflags = (currentflags << 1) | pkt.tcp.psh;
    currentflags = (currentflags << 1) | pkt.tcp.rst;
    currentflags = (currentflags << 1) | pkt.tcp.syn;
    currentflags = (currentflags << 1) | pkt.tcp.fin;

  /* check flag for look connections */
```

```
  if (currentflags == 0x00 || currentflags == 0x01 || currentflags == 0x02 || currentflags == 0x29)
{
    if (if_ignore(pkt.ip.saddr, pkt.tcp.dest)) {
     continue;
    }

    index = allocate(&has, pkt.ip.saddr);

/* initial struct packet for new address */
    if (!has) {
     host[index].src_addr = pkt.ip.saddr;
     host[index].firsttime = now;
     host[index].lasttime = now;
         host[index].lastindex = 0;
     host[index].src_port[0] = pkt.tcp.source;
     host[index].dst_port[0] = pkt.tcp.dest;
     host[index].count = 1;
     host[index].flags[0] = currentflags;
         host[index].portmatch = 0;
         /* check special port */
         if (isPortMatch(pkt.tcp.dest))
                 host[index].portmatch++;
     continue;
    }

.//printf("%d:%d:",ntohs(host[index].dst_port[host[index].lastindex]),ntohs(pkt.tcp.dest));
    if (host[index].flags[host[index].lastindex] == currentflags) {
         if (host[index].dst_port[host[index].lastindex] != pkt.tcp.dest) {
           if ((now - host[index].lasttime) <= TIMEINTERVAL) {
                 host[index].count++;
                 host[index].lastindex++;
```

```
                lastindex = host[index].lastindex;
                host[index].src_port[lastindex] = pkt.tcp.source;
                host[index].dst_port[lastindex] = pkt.tcp.dest;
                host[index].flags[lastindex] = currentflags;
                host[index].lasttime = now;

                /* check for MAXSCAN */
                if (host[index].count == MAXSCAN) {
                display(host[index]);
                add_ignore(host[index].src_addr);
                deallocate(index);
                }

                /* check special port */
                if (isPortMatch(pkt.tcp.dest))
                        host[index].portmatch++;

                /* check number of portmatch specified in Config */
//printf("portmatch=%d",host[index].portmatch);
        if (host[index].portmatch == num_monitor_port) {
                display(host[index]);
                add_ignore(host[index].src_addr);
                deallocate(index);
                }
          }
          else {
                host[index].timeintervalcount++;
                if (host[index].timeintervalcount == MAXTIMEINTERVAL)
                        deallocate(index);
          }
        }
```

```
        else {
                host[index].sameportcount++;
                if (host[index].sameportcount == MAXSAMEPORT)
                deallocate(index);
        }
    }
    else {
                host[index].diffflagcount++;
                if (host[index].diffflagcount == MAXDIFFFLAG)
                deallocate(index);
    }
   } /* end if (pkt.tcp.ack == 0) */
  } /* end while(1) */
} /* end main() */

/*
 *
 *              TCP Port Scanning Detection Program
 *
 *               Information Technology Project
 *
 *               Author : Choosak Bunyasiriwat
 *               Information Science(IS7.2)
 *              Information Technology Faculty
 *          King Mongkut's Institute of Technology Ladkrabang
 *
 */
/*****************************************************************************
 * Written by   : Choosak Bunyasiriwat
 * File         : readconfig.c
 * Description  : read about hosts and ports from config file
```

```
* Last Modify  : Mar 4,2001
* Contains of  : void readconfig()
*              : int chk_ignore(int addr)
*              : int readportconf(char *buff, int *ports)
*              : void add_port(char *temp,int num,int *ports)
*              : int readhostconf(char *buff, char **hosts)
*****************************************************************************/
#include "globaldef.h"
#include "ignore.h"
#include "readconfig.h"

void getSysInfo(char * infoname)
{
    struct utsname sysInfo;

    if (uname(&sysInfo) != -1) {
          if (strcmp(infoname,"sysname") == 0) {
        puts(sysInfo.sysname);
          }
          else if (strcmp(infoname,"nodename") == 0) {
        //puts(sysInfo.nodename);
            fromhost = strdup(sysInfo.nodename);
          }
          else if (strcmp(infoname,"release") == 0) {
        puts(sysInfo.release);
          }
          else if (strcmp(infoname,"version") == 0) {
        puts(sysInfo.version);
          }
          else if (strcmp(infoname,"machine") == 0) {
        puts(sysInfo.machine);
```

```
        }
    }
    else {
      perror("uname() error");
    }
}

char *ConfigTrim(char *v)
{
        char *tail;

        if (!v) return NULL;
        for (;(*v != 0) && (*v == ' ' || *v == '\t');v++);
        tail = &v[strlen(v)-1];
        for (;(tail >= v) && (*tail == ' ' || *tail == '\t');tail--)
                *tail = 0;
        return v;
}

char **LoadConfig(char *fname)
{
        FILE *fp;
        char str[1024];
        char *name,*value;
        char **config;
        int cnt;

        if (!fname) return NULL;
        fp = fopen(fname,"r");
        if (!fp) return NULL;
        config = NULL; cnt = 0;
```

```
        while (1) {
                if (!fgets(str,sizeof(str),fp)) break;
                if (!strtok(str,"\r\n")) continue; /* Check for blank line */
                if (str[0] == '#')      continue; /* Check for comment line */
                name = ConfigTrim(strtok(str,"="));
                if (!name) continue;
                value = ConfigTrim(strtok(NULL,""));
                if (!value) continue;
                if (cnt > 0)
                        config = (char**)realloc(config,sizeof(char*)*((cnt+1)*2));
                else
                        config = (char**)malloc(sizeof(char*)*2);
                config[cnt*2] = strcpy((char*)malloc(strlen(name)),name);
                config[(cnt*2)+1] = strcpy((char*)malloc(strlen(value)),value);
                cnt++;
        }
        if (config) {
                config = (char**)realloc(config,sizeof(char*)*((cnt*2)+1));
                config[cnt*2] = NULL;
        }
        return config;
}

char **ConfigClear(char **config)
{
        int i;

        if (!config) return NULL;
        for (i = 0;config[i];i++)
                free(config[i]);
        free(config);
```

```
        return NULL;
}

char *ConfigGetStringValue(char **config,char *name)
{
        int i;

        if (!config || !name) return NULL;
        for (i = 0;config[i];i+=2)
                if (!strcasecmp(config[i],name))
                        return config[i+1];
        return NULL;
}

long ConfigGetLongValue(char **config,char *name)
{
        char *value = ConfigGetStringValue(config,name);

        if (value) return atol(value);
        return 0;
}

/*****************************************************************************
 * Function    : readconfig()
 * Argument    : nothing
 * Description : read config file for get config about hosts and ports
 * Return      : (void) nothing
 *****************************************************************************/
void readconfig()
{
  char ** Config;
```

```
char * allport = "";
int i=0;

// Initialize CONFIG's value
MAILTO = "";
TIMEINTERVAL = 0;
MAXSCAN = 0;

mail = 0;

Config = LoadConfig(CONFIGFILENAME);
TRUSTEDHOST = ConfigGetStringValue(Config,"TRUSTEDHOST");
MAILTO = ConfigGetStringValue(Config,"MAILTO");
TIMEINTERVAL = ConfigGetLongValue(Config,"TIMEINTERVAL");
MAXSCAN = ConfigGetLongValue(Config,"MAXSCAN");
allport = ConfigGetStringValue(Config, "TCPPORT");
MAXDIFFFLAG = ConfigGetLongValue(Config, "MAXDIFFFLAG");
MAXSAMEPORT = ConfigGetLongValue(Config, "MAXSAMEPORT");
MAXTIMEINTERVAL = ConfigGetLongValue(Config, "MAXTIMEINTERVAL");

if (TRUSTEDHOST != NULL) {
        printf("TRUSTEDHOST=%s\n",TRUSTEDHOST);
    ignore_hosts = (char **) malloc(MAX_IGNORE);
        num_ignore_host = readhostconf(TRUSTEDHOST,ignore_hosts);
} else
        num_ignore_host = i;
if (MAILTO != NULL) {
        printf("MAILTO=%s\n",MAILTO);
        getSysInfo("nodename");
    mail = 1;
}
```

```
if (TIMEINTERVAL != 0) {
        printf("TIMEINTERVAL=%d\n",TIMEINTERVAL);
}
if (MAXSCAN != 0) {
        printf("MAXSCAN=%d\n",MAXSCAN);
}
if (allport != NULL) {
    monitor_ports = (int *) malloc(MAXPORT);
    num_monitor_port = readportconf(allport,monitor_ports);
}
if (MAXDIFFFLAG != 0) {
        printf("MAXDIFFFLAG=%d\n",MAXDIFFFLAG);
}
if (MAXSAMEPORT != 0) {
        printf("MAXSAMEPORT=%d\n",MAXSAMEPORT);
}
if (MAXTIMEINTERVAL != 0) {
        printf("MAXTIMEINTERVAL=%d\n",MAXTIMEINTERVAL);
}
}

/*****************************************************************************
 * Function    : chk_ignore(int addr)
 * Argument    : 'addr' is host address want to check
 * Description : check ignore hosts from config file
 * Return      : (int) ignore or not
 *****************************************************************************/
int chk_ignore(int addr)
{
  int i;
  struct in_addr ia;
```

```
 if (num_ignore_host != 0) {
  for (i = 0;i < num_ignore_host;i++) {
   ia.s_addr = inet_addr(ignore_hosts[i]);
   if (ia.s_addr == addr) {
printf("\tpacket from TRUSTEDHOST(%s)\n",ignore_hosts[i]);
      return IGNORE;
   }
  }
 }
 return NO_IGNORE;
}

/*****************************************************************************
 * Function    : readportconf(char *buff, int *ports)
 * Argument    : 'buff' is string that have all of port name
 *             : 'ports' is name of port array
 * Description : get all port
 * Return      : (int) number of all port
 *****************************************************************************/
int readportconf(char *buff, int *ports)
{
 char *temp;
 int allp = 0;

 if ((temp = strtok(buff, ",")) != NULL) {
  add_port(temp,0,ports);

  for(allp = 1;(temp=strtok(NULL,",")) != NULL;allp++){
   add_port(temp,allp,ports);
  }
 }
```

```
 return allp;
}

/*****************************************************************************
 * Function    : add_port(char *temp,int num,int *ports)
 * Argument    : 'temp' is port name want to add into port array
 *             : 'num' is index point to port array
 *                   : 'ports' is name of port array
 * Description : add port into list
 * Return      : (void) nothing
 *****************************************************************************/
void add_port(char *temp,int num,int *ports)
{
 struct servent *serv;
 int p;

 if ((temp[0] > '0') && (temp[0] < '9')) {
  p = htons(atoi(temp));
  if ((serv = getservbyport(p, "tcp")) != NULL) {
   ports[num] = serv->s_port;
  }
  else {
   printf ("port %s : please check in config file\n", temp);
   exit (1);
  }
 }
 else {
  if ((serv = getservbyname(temp, "tcp")) != NULL) {
   ports[num] = serv->s_port;
   printf("%s = %d\n",temp,ntohs(ports[num]));
  }
```

```
  else {
    printf ("port %s : please check in config file\n", temp);
    exit (1);
  }
 }
}

/*****************************************************************************
 * Function    : readhostconf(char *buff, int *hosts)
 * Argument    : 'buff' is string that have all of host name
 *             : 'hosts' is name of host array
 * Description : get all host
 * Return      : (int) number of all host
 *****************************************************************************/
int readhostconf(char *buff, char **hosts)
{
 char *temp;
 int allh = 0;

 if ((temp = strtok(buff, ",")) != NULL) {
  hosts[allh] = strdup(temp);
  for(allh = 1;(temp=strtok(NULL,",")) != NULL;allh++){
        hosts[allh] = strdup(temp);
  }
 }
 return allh;
}

/*
 *
 *
```

```
*
*                Information Technology Project
*
*                Author : Choosak Bunyasiriwat
*                 Information Science(IS7.2)
*                Information Technology Faculty
*          King Mongkut's Institute of Technology Ladkrabang
*
*/
/*****************************************************************************
 * Written by   : Choosak Bunyasiriwat
 * File         : display.c
 * Description  : show about intrusion
 * Last Modify  : Mar 4,2001
 * Contains of  : char *hostlookup(int i)
 *              : void display(struct host hosts)
 *              : int send_mail(char *message)
 *****************************************************************************/
#include "globaldef.h"
#include "display.h"


/*****************************************************************************
 * Function     : hostlookup(int i)
 * Argument     : 'i' is source address
 * Description  : look for host name
 * Return       : (char) name of host
 *****************************************************************************/
char *hostlookup(int i)
{
  static char hostname[256];
```

```
 struct in_addr ia;
 struct hostent *he;

 ia.s_addr = i;
 if (!(he = gethostbyaddr((char *)&ia, sizeof ia, AF_INET))) {
  strncpy(hostname,inet_ntoa(ia),sizeof hostname);
 }
 else {
  strncpy(hostname,he->h_name,sizeof hostname);
 }
 return hostname;
}

/*************************************************************************
 * Function    : display(struct host hosts)
 * Argument    : 'hosts' is description of host
 * Description : print message to monitor, write log file and send mail
 * Return      : (void) nothing
 *************************************************************************/
void display(struct host hosts)
{
 int i, src_port, dst_port;
 char * scanfromip;
 char * scanfromname;
 char disp[256];
 char *firsttime;
 struct in_addr ia;

 ia.s_addr = hosts.src_addr;
 scanfromip = inet_ntoa(ia);
 scanfromname = hostlookup(hosts.src_addr);
```

```
firsttime = ctime(&hosts.firsttime);

switch (hosts.flags[0]) {
  case 0x00 :
      sprintf (disp,"Possible TCP Scan Type: NULL scan\nScan From Source Address:
%s(%s)\nFirst Packet Start: %s", scanfromip, scanfromname, firsttime);
          scantype = "NULL scan";
          src_addr = scanfromip;
          firstscan = firsttime;
      break;

  case 0x01 :
      sprintf (disp,"Possible TCP Scan Type: FIN scan\nScan From Source Address:
%s(%s)\nFirst Packet Start: %s", scanfromip, scanfromname, firsttime);
          scantype = "FIN scan";
          src_addr = scanfromip;
          firstscan = firsttime;
      break;

  case 0x02 :
      sprintf (disp,"Possible TCP Scan Type: SYN scan\nScan From Source Address:
%s(%s)\nFirst Packet Start: %s", scanfromip, scanfromname, firsttime);
          scantype = "SYN scan";
          src_addr = scanfromip;
          firstscan = firsttime;
      break;

  case 0x29 :
      sprintf (disp,"Possible TCP Scan Type: XMAS scan\nScan From Source Address:
%s(%s)\nFirst Packet Start: %s", scanfromip, scanfromname, firsttime);
          scantype = "XMAS scan";
```

```
            src_addr = scanfromip;
            firstscan = firsttime;
        break;

   default   :
        sprintf (disp,"Possible TCP Scan Type: UNKNOWN scan\nScan From Source Address:
%s(%s)\nFirst Packet Start: %s", scanfromip, scanfromname, firsttime);
            scantype = "UNKNOWN scan";
            src_addr = scanfromip;
            firstscan = firsttime;
        break;

  }

  /* for log file */
  if ((LOGFILE = fopen(LOGFILENAME,"a")) != NULL) {
   fprintf (LOGFILE, "%s", disp);
  }
  else
   printf ("Can't open log file\n");

  // Display detection via Monitor
  printf ("%s",disp);

  // printf ("IGNORE: %s for 5 seconds\n", scanfromip);

  for (i = 0;i < MAXSCAN;i++) {
   src_port = ntohs(hosts.src_port[i]);
   dst_port = ntohs(hosts.dst_port[i]);

   if ((src_port != 0) && (dst_port != 0)) {
```

```
    printf("\tPacket No.: %d Source Port: %d -> Destination Port: %d\n", i+1, src_port, dst_port);
    fprintf(LOGFILE,"\tPacket No.: %d Source Port: %d -> Destination Port: %d\n", i+1,
src_port, dst_port);
   }
  }

  if (mail) {
   if ((send_mail(disp)) < 0)
    printf ("Can't send mail\n");
   else {
    printf("Send mail to %s already\n",MAILTO);
    printf("          ##########################################\n");
    fprintf(LOGFILE,"          ##########################################\n");
   }
  }

  fclose (LOGFILE);
}

int readline(fd,ptr,maxlen)
register int fd;
register char *ptr;
register int maxlen;
{
        int n,rc;
        char c;

        for (n=1;n<maxlen;n++) {
                if ((rc = read(fd,&c,1)) == 1) {
                        *ptr++ = c;
                        if (c == '\n')
```

```
                        break;
                } else if (rc == 0) {
                        if (n == 1)
                                return 0;
                        else
                                break;
                } else
                        return -1;
        }

        *ptr = 0;
        return n;
}

int mail_read_until(int sockfp,char *head,char *err)
{
        char tmp[4096];

        do {
                if (!readline(sockfp,tmp,sizeof(tmp))) return 0;
                if (err && !strncmp(tmp,err,strlen(err))) return 0;
        } while (strncmp(tmp,head,strlen(head)));
        return 1;
}

/*****************************************************************************
 * Function     : send_mail(char *message)
 * Argument     : 'message' is description of intrusion
 * Description  : send mail to system administrator
 * Return       : (int) return 0 if success or return -1 if fail
 *****************************************************************************/
```

```
int send_mail(char *message)
{
 int sockfd,i;
 struct sockaddr_in serv_addr;
 register struct hostent *hostptr;
 char combuf[256];
 char *comm[] = {"HELO ", fromhost, "MAIL FROM: TCPPSD@", fromhost,
          "RCPT TO:", MAILTO, "DATA", "Subject:", "QUIT"
         };
 time_t now;


if ((hostptr = gethostbyname(fromhost)) == NULL) {
  printf ("Error gethostbyname error for host: %s",fromhost);
  return -1;
 }

 bzero((char *) &serv_addr,sizeof(serv_addr));
 serv_addr.sin_family = AF_INET;
 bcopy( hostptr->h_addr,  &serv_addr.sin_addr, hostptr->h_length);
 serv_addr.sin_port = htons( MAILPORT );

 if ((sockfd = socket(AF_INET,SOCK_STREAM,0)) < 0) {
  printf ("Error can't open stream socket");
  return -1;
 }

 if (connect(sockfd,(struct sockaddr *) &serv_addr.sizeof(serv_addr)) < 0) {
  printf ("Error can't connect to server");
  return -1;
 }
```

```
  // send HELO command to SMTP Server
  i=0;
  sprintf (combuf, "%s%s\r\n", comm[i], comm[i+1]);
//printf("helo->%s",combuf);
  if (write(sockfd, combuf, strlen(combuf)) < 0) {
     close(sockfd);
     printf ("error:can't send HELO command\n");
     return -1;
  }
  mail_read_until(sockfd,"250",NULL);
  // send MAIL command to SMTP Server
  i+=2;
  sprintf (combuf, "%s%s\r\n", comm[i], comm[i+1]);
//printf("mail->%s",combuf);
  if (write(sockfd, combuf, strlen(combuf)) < 0) {
     close(sockfd);
     printf ("error:can't send MAIL command\n");
     return -1;
  }
  if (!mail_read_until(sockfd,"25","55")) { close(sockfd); return 0;}
  // send RCPT command to SMTP Server
  i+=2;
  sprintf (combuf, "%s%s\r\n", comm[i], comm[i+1]);
//printf("rcpt->%s",combuf);
  if (write(sockfd, combuf, strlen(combuf)) < 0) {
     close(sockfd);
     printf ("error:can't send RCPT command\n");
     return -1;
  }
  if (!mail_read_until(sockfd,"25","55")) { close(sockfd); return 0;}
  // send DATA command to SMTP Server
```

```
  i+=2;
  sprintf (combuf, "%s\r\n", comm[i]);
//printf("data->%s",combuf);
  if (write(sockfd, combuf, strlen(combuf)) < 0) {
    close(sockfd);
    printf ("error:can't send DATA command\n");
    return -1;
  }
  mail_read_until(sockfd,"354",NULL);
  // send Subject: message
  i+=1;
  now = time(0);
  sprintf (combuf, "%s %s from %s at %s\r\n", comm[i], scantype, src_addr, firstscan);
//printf("subject->%s",combuf);
  if (write(sockfd, combuf, strlen(combuf)) < 0) {
    close(sockfd);
    printf ("error:can't send Subject: message\n");
    return -1;
  }
  // send message in body(details about TCP Scan Port)
//printf("message->%s",message);
  if (write(sockfd, message, strlen(message)) < 0) {
    printf ("error:can't send message in body\n");
    return -1;
  }
  // send \r\n.\r\n for end body
  if (write(sockfd, "\r\n.\r\n", 5) < 0) {
    printf ("error:can't send \\r\\n.\\r\\n for end body\n");
    return -1;
  }
  mail_read_until(sockfd,"250",NULL);
```

```
  // send QUIT command to SMTP Server
  i+=1;
  sprintf (combuf, "%s\r\n", comm[i]);
//printf("quit->%s",combuf);
  if (write(sockfd, combuf, strlen(combuf)) < 0) {
    close(sockfd);
    printf ("error:can't send QUIT command\n");
    return -1;
  }
  mail_read_until(sockfd,"221",NULL);
  close(sockfd);
  return 0;
}

/*
 *
 *              TCP Port Scanning Detection Program
 *
 *               Information Technology Project
 *
 *               Author : Choosak Bunyasiriwat
 *               Information Science(IS7.2)
 *          Information Technology Faculty
 *          King Mongkut's Institute of Technology Ladkrabang
 *
 */
/*****************************************************************************
 * Written by   : Choosak Bunyasiriwat
 * File         : ignore.c
 * Description  : manage about host that ignore
 * Last Modify  : Mar 4,2001
```

```
 ignore_host[index].starttime = time(0);
}

/*****************************************************************************
 * Function    : del_ignore(int index)
 * Argument    : 'index' is index of array that point to ignore host
 * Description : delete ignore host from array
 * Return      : (void) nothing
 *****************************************************************************/
void del_ignore(int index)
{

 ignore_host[index].host = 0;
 ignore_host[index].starttime = 0;
}

/*****************************************************************************
 * Function    : if_ignore(int addr)
 * Argument    : 'addr' is host address want to check
 * Description : check this address that is ignore host
 * Return      : (int) ignore or not
 *****************************************************************************/
int if_ignore(int addr, int port)
{
 int i;
 time_t now = time(0);

/*
 * check ignore host form config file
 */
 if (chk_ignore (addr) == IGNORE) {
```

```
  return IGNORE;
 }

/*
 * check ignore host that just attack
 */
 for (i = 0;i < MAX_IGNORE;i++ ) {
  if (ignore_host[i].host == addr) {
   if ((now - ignore_host[i].starttime) <= I_SEC)
    return IGNORE;
   else {
    del_ignore(i);
    return NO_IGNORE;
   }
  }
 }

 return NO_IGNORE;
}
```

A-3. ตัวอย่างของแฟ้มคอนฟิกูเรชัน(Configuration File)

```
# Sample configuration file for tcppsd program
#
# You can specify here which host are trusted and should be ignored.
# Each entry should be in a separate comma.
#
# Format:
# TRUSTEDHOST = ip_address[,ip_address]
# Example: TRUSTEDHOST = 127.0.0.1
# Detection will skip connections from localhost(127.0.0.1)
TRUSTEDHOST = 127.0.0.1
```

If it found that intrusion happened, we send mail to this E-mail(user@host)

MAILTO = root@firewall.hitech.com

Time interval of two adjacent packets within TIMEINTERVAL will be count

TIMEINTERVAL = 1

Max. numbers of suspected packets from same source address will be intrusion

MAXSCAN = 20

Special TCP Port to monitor

TCPPORT = ftp,telnet

Max. numbers of two adjacent packets have different flag.

MAXDIFFFLAG = 10

Max. numbers of two adjacent packets have same destination port.

MAXSAMEPORT = 10

Max. numbers of two adjacent packets have time interval more than TIMEINTERVAL

MAXTIMEINTERVAL = 10

ประวัติผู้เขียน

ชื่อ นาย ชูศักดิ์ บุญญศิริวัฒน์

วันเกิด 19 สิงหาคม พ.ศ. 2518

สถานที่เกิด อำเภอ หัวหิน จังหวัด ประจวบคีรีขันธ์

ประวัติการศึกษา

จบการศึกษาระดับปริญญาตรี วิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยธรรมศาสตร์ ปีการศึกษา 2539

ประวัติการทำงาน

ชื่อบริษัท บริษัท เดอร์เนล คอมพิวเตอร์ แอนด์ คอมมิวนิเคชัน จำกัด

ตำแหน่ง System Engineer

ระยะเวลาทำงาน พ.ศ. 2540 - ปัจจุบัน