ระบบควบคุมไฟร์วอลผ่านระบบตรวจจับการบุกรุก

Firewall control system with intrusion detection system

โดย

วีระวัฒน์ พัฒโน

รหัส 45066094

อาจารย์ที่ปรึกษา

ผศ. อัครินทร์ คุณกิตติ

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2547

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	ระบบควบคุมไฟร์วอลผ่านระบบตรวจจับการบุกรุก
นักศึกษา	นายวีระวัฒน์ พัฒโน
อาจารย์ที่ปรึกษา	ผศ. อัครินทร์ คุณกิตติ
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2547

บทคัดย่อ

ระบบเครือข่ายอินเตอร์เน็ตในปัจจุบันนั้น จำเป็นต้องมีการป้องกันภัยคุกคามจากการโจมตีที่มีอยู่หลายรูปแบบ และเกิดขึ้นอยู่ตลอดเวลา ดังนั้นการรักษาความปลอดภัยให้กับระบบเครือข่ายนั้นจะต้องมีการตรวจดูข้อมูลบนเครือข่าย หากมีการตรวจพบว่าเป็นการโจมตีจากผู้บุกรุกก็จะต้องป้องกันการบุกรุกนั้นทันที ด้วยเหตุดังกล่าวจึงได้ทำการพัฒนาระบบควบคุมไฟร์วอลผ่านระบบตรวจจับการบุกรุกขึ้น โดยเป็นการพัฒนาบนระบบปฏิบัติการลินุกซ์ และทำงานอยู่บนเครื่องเดียวกับระบบตรวจจับการบุกรุกบนเครือข่าย Snort โดย Snort จะบันทึกข้อมูลของการบุกรุกที่ตรวจพบลงไปที่ไฟล์ alert ระบบควบคุมไฟร์วอลที่พัฒนาขึ้นจะอ่านข้อมูลจากไฟล์นี้ เพื่อนำเอาข้อมูลการบุกรุกมาสร้างเป็นสคริปต์คำสั่ง แล้วจึงใช้สคริปต์นี้สั่งให้ไฟร์วอลป้องกันการบุกรุกนั้นทันที และยังให้ผู้ใช้ปรับแต่งค่าในการทำงานได้เอง เช่น ตั้งระดับในการป้องกัน เปลี่ยนรูปแบบของสคริปต์คำสั่งไฟร์วอล กำหนดเวลาในการยกเลิกการป้องกันที่เคยสั่งไว้ ผลที่ได้จากการพัฒนาระบบนั้นเป็นไปตามที่ต้องการคือสามารถที่จะอ่านข้อมูลการบุกรุกจาก alert ได้อย่างถูกต้อง และสามารถสั่งไฟร์วอลให้ป้องกันได้ทันที ซึ่งไฟร์วอลที่ใช้ได้แก่ iptables, ipchains และipfw เป็นต้น ด้วยระบบที่พัฒนาขึ้นนี้จะเป็นการเพิ่มประสิทธิภาพในการรักษาความปลอดภัยบนเครือข่าย

Title	Firewall control system with intrusion detection system
Student	Mr. Weerawat Pattano
Advisor	Asst. Prof. Akharin Khunkitti
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2004

ABSTRACT

Since all Internet network systems require protection against various attacks, effective intrusion detection systems must be implemented in order to provide security for the network systems. This project is a development of firewall control system base on Linux operating system. The system runs on a computer which has intrusion detection systems called "Snort". Snort detects the intrusions and then records their information into a log file named alert. Then, the firewall control system will read the intrusion information from the alert file to get intrusion information. The information is used to build firewall scripts for sending commands to the firewall to block such intrusions. The program allow user to configure variables such as blocking level, firewall script pattern and release expiration time. The program can detect intrusions from alert file and then run firewall script immediately to protect network. This program can work with iptables, ipchains and ipfw. As conclusion, this system is an effective automatic protection for networks on the Internet.

กิตติกรรมประกาศ

ในการทำโครงการพัฒนาระบบงานครั้งนี้จะไม่สามารถสำเร็จลุล่วงได้ถ้าหากขาดบุคคลท่านหนึ่งผู้ที่ให้ความช่วยเหลือในการทำโครงการ ดังนั้นผู้จัดทำโครงการขอขอบพระคุณท่านผู้นี้ ซึ่งก็คือ ผศ. อัครินทร์ คุณกิตติ อาจารย์ที่ปรึกษา แม้ที่ผ่านมาผู้จัดทำโครงการขาดความต่อเนื่องในการทำงาน แต่อาจารย์ก็ได้ให้โอกาสในการทำงานนี้เสมอมา และชี้แนะสิ่งที่เป็นประโยชน์หลายอย่าง รวมทั้งเอาใจใส่กับโครงงานที่ได้ทำขึ้นเป็นอย่างดี

สุดท้ายนี้ขอกราบขอบพระคุณบิดา มารดา ผู้ที่ให้การสนับสนุนอย่างเต็มที่ตลอดมา และขอบคุณพระเจ้าผู้ทรงประทานกำลัง สติปัญญาในการทำงาน และเป็นแสงสว่างนำทางในชีวิต

วีระวัฒน์ พัฒโน

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญภาพ

หน้า

ภาพที่

สารบัญภาพ (ต่อ)

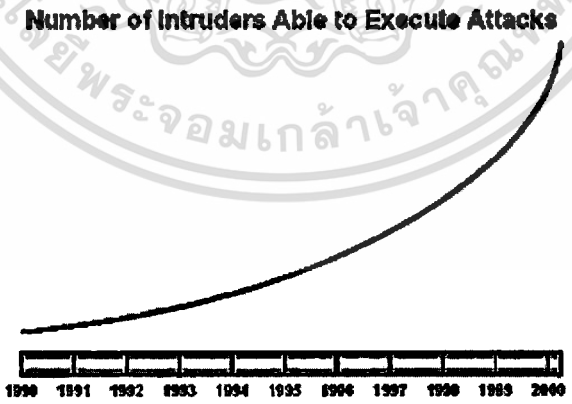
หน้า

ภาพที่

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากการที่มีจำนวนผู้ใช้อินเตอร์เน็ตเพิ่มมากขึ้นในแต่ละวัน อีกทั้งยังมีบรรดานักเขียนโปรแกรมอยู่ไม่น้อยที่สร้างโปรแกรมที่ช่วยในการเจาะระบบเครือข่ายคอมพิวเตอร์ออกมา และผู้ใช้อินเตอร์เน็ตก็สามารถหาได้จากอินเตอร์เน็ต อีกทั้งโปรแกรมเหล่านี้ก็ทำขึ้นมาให้ง่ายต่อการใช้งานมากขึ้น ซึ่งเป็นการเปิดโอกาสให้เกิดการเปลี่ยนแปลงจากการเป็นผู้ใช้อินเตอร์เน็ตทั่วไปกลายเป็นผู้บุกรุก (Intruder) ที่สามารถสร้างความเสียหายต่อระบบเครือข่ายคอมพิวเตอร์ได้ จากข้อมูลของทางหน่วยงาน CERT/CC ซึ่งเป็นหน่วยงานที่ทำงานเกี่ยวด้านความปลอดภัยของในระบบอินเตอร์เน็ตแห่งหนึ่ง ในแต่ละวันก็จะมีรายงานเหตุการณ์ที่มีการละเมิดความปลอดภัยคอมพิวเตอร์ เช่น ในปี ค.ศ.2000 มีจำนวน 21,756 ครั้ง , ปีค.ศ.2001 มีจำนวน 52,658 ครั้ง และช่วงไตรมาส 1-3 ของ ปี ค.ศ.2002 มีจำนวนถึง 73,359 ครั้งเลยทีเดียว และนอกจากนี้ได้มีศึกษาถึง การเพิ่มจำนวนของผู้บุกรุกในแต่ละปี ซึ่งแสดงเป็นกราฟไว้ ในรูปที่ 1.1



รูปที่ 1.1 การเพิ่มจำนวนของผู้บุกรุกในแต่ละปี

การที่จะทำให้ระบบเครือข่ายคอมพิวเตอร์มีความพร้อมที่จะสามารถเข้าใช้งานได้อย่างดีตลอดเวลานั้น นอกจากจะต้องดูแลอุปกรณ์ทั้งฮาร์ดแวร์ และซอฟท์แวร์ภายในเครือข่ายให้ดีแล้ว

เราจำเป็นต้องมีระบบป้องกันเครือข่ายจากการโจมตีจากผู้ไม่ประสงค์ดีทั้งจากภายในและภายนอก โดยเฉพาะอย่างยิ่งเครือข่ายคอมพิวเตอร์ที่เชื่อมต่อกับอินเตอร์เน็ตจึงเป็นเหตุให้ระบบการรักษาความปลอดภัยของเครือข่ายคอมพิวเตอร์นั้นมีการพัฒนาอย่างต่อเนื่อง ในขณะเดียวกันวิธีการโจมตี รวมทั้งจำนวนคนที่มีความสามารถในการโจมตีเครือข่ายก็มีมากขึ้นเช่นกัน ดังนั้นความถี่ในการที่เครือข่ายที่ใดที่หนึ่งจะถูกโจมตีนั้นเกิดบ่อยขึ้นเรื่อยๆ ส่วนวิธีการในการโจมตีระบบนั้นก็มีหลายรูปแบบและมีการพัฒนารูปแบบการโจมตีให้มีมากขึ้นด้วยดังนั้นผู้ดูแลระบบเครือข่ายจึงมีบทบาทสำคัญในการดูและจัดการให้เกิดความปลอดภัยกับระบบเครือข่ายคอมพิวเตอร์ที่รับผิดชอบ และดูจะเป็นภาระหนักในการที่จะทำให้ระบบมีความปลอดภัย

ที่ผ่านมาจึงได้มีการคิดค้นเครื่องมือที่จะช่วยในการป้องกันระบบเครือข่าย และหนึ่งในนั้นคือ ระบบตรวจจับการบุกรุกบนเครือข่าย (Network-based Intrusion Detection System) ซึ่งเป็นระบบที่ทำการตรวจจับ Packet ที่วิ่งบนเครือข่าย เพื่อเอามาวิเคราะห์ว่ามีการพยายามที่จะโจมตีเครือข่ายหรือไม่ ซึ่งเมื่อพบการโจมตีการจะแจ้งเตือนในผู้ดูแลระบบรู้เพื่อป้องกันต่อไป และโปรแกรมที่นิยมใช้กัน คือ Snort ซึ่งเป็นระบบตรวจจับการบุกรุกบนเครือข่ายที่มีความสามารถในการตรวจจับการบุกรุกได้เป็นอย่างดี แต่ยังไม่สามารถทำการป้องกันการบุกรุกนั้นได้โดยอัตโนมัติ ซึ่งเครื่องมือที่ใช้การป้องกันนั้นคือ ไฟร์วอล (Firewall) ซึ่งเป็นระบบที่ใช้ในการควบคุมการเข้าถึงระหว่างเครือข่ายสองเครือข่าย เพื่อป้องกันเครือข่ายของเราจากการโจมตีของผู้ไม่ประสงค์ดี ซึ่งไฟร์วอลนั้นอาจเป็นได้ทั้ง เราเตอร์ คอมพิวเตอร์หนึ่งเครื่อง หรือหลายๆ เครื่อง ประกอบกัน โดยการควบคุมนั้นผู้ดูแลระบบจะต้องกำหนดเอง ทั้งนี้จะเห็นว่าถ้าหาก Snort มีการตรวจพบการโจมตีซึ่งอาจจะมาจากหลายๆ ที่ จะทำให้ผู้ดูแลระบบจะต้องทำงานหนักในการที่คอยสั่งไฟร์วอลให้ป้องกันในแต่ละครั้ง และอาจจะช้าเกินไปในการป้องกัน เพราะผู้บุกรุกอาจเข้าถึงระบบและสามารถสร้างความเสียหายให้เกิดขึ้นกับระบบได้

ดังนั้นจึงได้ทำการพัฒนาระบบที่ทำให้มีการป้องกันการบุกรุกได้ทันทีที่ Snort ตรวจพบการโจมตีโดยไม่ต้องอาศัยผู้ดูแลระบบคอยสั่งไฟร์วอลทุกครั้ง โดยระบบจะตรวจสอบการแจ้งเตือนการบุกรุกจาก Snort ผ่านไฟล์ alert.log ซึ่งเป็นไฟล์ที่บันทึกข้อมูลการบุกรุก แล้วจึงสั่งไฟร์วอลให้ป้องกันการบุกรุกทันที และไฟร์วอลตัวหลักที่ใช้นั้นคือ IPTABLES ซึ่งเป็นไฟร์วอลที่อยู่บนลินุกซ์ ในเคอร์เนลเวอร์ชั่นที่ 2.4 นอกจากนี้ระบบสามารถใช้งานกับไฟร์วอลได้หลายชนิดอื่นอีก ซึ่งการสั่งไฟร์วอลแต่ละแบบโดยขึ้นอยู่กับรูปแบบสคริปต์คำสั่งของไฟร์วอลตัวนั้น

1.2 ขอบเขตของโครงการพัฒนาระบบงาน

ในโครงการพัฒนาระบบจะเป็นการพัฒนาโปรแกรมบนระบบปฏิบัติการลินุกซ์ โดยใช้ภาษาC++ ในการเขียนโปรแกรม ซึ่งระบบที่พัฒนาขึ้นจะทำงานร่วมกับ Snort โดยใช้การอ่านไฟล์ alert ซึ่งเป็น Log ของ snort เพื่อหาข้อมูลของการบุกรุก ส่วนไฟร์วอลหลักที่ใช้ในโครงการนี้คือ IPtables โดยระบบนี้จะสามารถสั่งไฟร์วอลให้ป้องกันการบุกรุกที่พบโดยผ่านสคริปต์ของไฟล์วอล ซึ่งระบบที่จะพัฒนาขึ้นมานั้นเป็นโปรแกรมที่ทำขึ้นแยกต่างหาก ซึ่งจะไม่มีการปรับแต่งซอร์สโค้ดของ SNORT และ IPtables

1.3 ขั้นตอนการดำเนินงาน

ในโครงการนี้ได้แบ่งขั้นตอนในการศึกษา และพัฒนาโปรแกรมดังนี้

ขั้นตอนที่ 1 ศึกษาการทำงานของ Snort หลักการในการตรวจจับการบุกรุก การแจ้งเตือนลง Log file รูปแบบการบันทึก และในส่วนการทำงานของคำสั่งที่ใช้ควบคุมไฟร์วอล IPTABLES

ขั้นตอนที่ 2 ศึกษาการเขียนโปรแกรมด้วยภาษา C++ และศึกษาเครื่องมือที่ใช้ในการพัฒนาโปรแกรม

ขั้นตอนที่ 3 วิเคราะห์และออกแบบระบบ ตามขอบเขตของโครงการ

ขั้นตอนที่ 4 พัฒนาตัวโปรแกรม และทดสอบปรับปรุงให้ตรงกับความต้องการ

ขั้นตอนที่ 5 สรุปผลการใช้งานระบบ

ขั้นตอนที่ 6 ปรับปรุงเพิ่มเติมความสามารถอื่น รวมทั้งในการใช้กับไฟร์วอลแบบอื่นนอกจาก IPTABLES

1.4 ประโยชน์ที่คาดว่าจะได้รับ

- เข้าใจหลักการทำงานของโปรแกรมตรวจจับการบุกรุก Snort และไฟร์วอล และสามารถประยุกต์ใช้งานในการรักษาความปลอดภัยในเครือข่ายได้

- เข้าใจในการเขียนโปรแกรมด้วยภาษา C++ บนระบบลินุกซ์ และสามารถนำความรู้ไปใช้พัฒนาโปรแกรมอื่นๆ ได้

- ระบบที่สร้างขึ้นสามารถช่วยให้เกิดการป้องกันการบุกรุกบนเครือข่ายโดยอัตโนมัติ

บทที่ 2
ระบบตรวจจับการบุกรุกและไฟร์วอล

2.1 รูปแบบการโจมตีระบบเครือข่าย

การโจมตีระบบเครือข่ายนั้นมีหลายรูปแบบ ก่อนอื่นเราจะต้องรู้ว่าผู้บุกรุกจะสามารถใช้วิธีใดบ้างที่จะโจมตีระบบของเรา เพื่อที่เราเองจะสามารถเตรียมการป้องกันได้อย่างมีประสิทธิภาพ และครอบคลุมการโจมตีทุกรูปแบบที่จะมา และวิธีการที่โจมตีนั้นมีมากมาย และมีการคิดค้นใหม่ได้อยู่เรื่อย ๆ แต่ที่จะยกตัวอย่างนั้น ได้แก่

1) DoS (Denial of service) เป็นการโจมตีไปเครื่องเป้าหมาย โดยใช้เครื่องที่ถูก compromise แล้ว จากที่อื่นๆ โจมตีไปเครื่องเป้าหมายพร้อมๆ กัน ทำให้เกิดความคับคั่งในเครือข่าย และเครื่องที่ถูกโจมตีไม่สามารถทำงานตามปกติได้
2) Web server attacks โจมตีผ่านทาง web server เช่น พยายาม execute file พวก CGI หรือ script ที่เตรียมไว้บน Web server เพื่อจะเข้าควบคุมเครื่องนั้น
3) Web browser attacks การโจมตีจะแฝงมากับ script ต่างๆ บนหน้าเว็บ เช่น เมื่อเปิดเว็บที่มี Script เหล่านี้ เครื่องก็จะรัน Script ซึ่งอาจเป็น Script ที่ส่งผลเสียต่อเครื่องที่เปิด
4) Software Bug มีผู้ไม่ประสงค์ดีพยายามใช้ช่องโหว่ของ software ในการโจมตี ซึ่งส่วนมาก Bug จะถูกแก้ไขเมื่อ พบว่ามีการโจมตีโดยอาศัยช่องทางนี้ไปแล้ว
5) Access การพยายาม log in เป็น Root ของเครื่อง การcrack password การพยายาม เข้าถึง File ที่เก็บ บัญชีผู้ใช้เครื่อง
6) IP spoofing การปลอม IP Address เข้ามาโจมตี เพื่อให้สามารถเข้าสู่ระบบที่ให้บาง IP address เข้าได้ และทำให้หาแหล่งที่โจมตีไม่ได้
7) Buffer Overflows พยายามส่งข้อมูลให้เกิด Buffer Overflows ซึ่งจะลดความสามารถในการป้องกัน ทำให้ง่ายขึ้นในการโจมตีของเครื่องนั้น
8) Ping sweeps เป็นพยายามหาเครื่องที่ online เพื่อที่จะได้ทำการโจมตี โดยวิธีการ ping คือ ส่ง ICMP Packet แล้วให้เครื่องตอบกลับมา
9) Scan port การค้นหา port ที่เปิดทิ้งไว้ที่เครื่องของเหยื่อเปิด เพื่อจะอาศัย port นั้นเข้าไปโจมตี

10) Account scans การค้นหา บัญชีผู้ใช้ที่มีสิทธิ์ในการเข้าสู่ระบบ อาจจะใช้การดักข้อมูลบนเครือข่ายแล้วพยายามหาชื่อบัญชีผู้ใช้จากข้อมูลเหล่านั้น

11) Ping of death เป็นการ ping ไปยังเครื่องเป้าหมาย เพื่อให้เครื่องเป้าหมายตอบกลับมาอย่างต่อเนื่องและเป็นจำนวนมาก ซึ่งจะทำให้เครื่องช้าลง และเกิดความคับคั่งในเครือข่ายด้วย

12) SYN flood การส่ง TCP SYN Packet จำนวนมากอย่างรวดเร็ว ไปที่เครื่องเป้าหมาย เพื่อให้ตอบกลับมา หวังผลลดความสามารถการทำงานของเครื่องในขณะนั้น หรือ ใช้ UDP flood ก็ได้

2.2 ระบบตรวจจับผู้บุกรุกบนเครือข่าย

ระบบตรวจจับผู้บุกรุก หรือ Intrusion Detection System เป็นระบบตรวจจับผู้บุกรุกชนิดหนึ่ง ซึ่งระบบตรวจจับผู้บุกรุกมีอยู่หลักๆ 3 ชนิดดังนี้

- ระบบตรวจจับผู้บุกรุกบนเครื่องแม่ข่าย (Host-based Intrusion Detection System) เป็นระบบที่คอยตรวจจับการบุกรุกในเครื่องคอมพิวเตอร์ที่มีระบบนี้อยู่เท่านั้น โดยตรวจสอบดูการทำงาน วิเคราะห์การใช้งานภายในเครื่อง เช่น การ access file , CPU utilization, วิเคราะห์ Log file, การ Log in เข้ามาใช้งาน เป็นต้น
- ระบบตรวจจับผู้บุกรุกบนโปรแกรม (Application-based Intrusion Detection System) เป็นระบบที่คอยตรวจจับการบุกรุกในระดับโปรแกรม โดยเฝ้าดูการทำงานเฉพาะโปรแกรมบางโปรแกรม เช่น ตรวจ Log file ของโปรแกรมนั้น
- ระบบตรวจจับผู้บุกรุกบนเครือข่าย (Network-based Intrusion Detection System) เป็นระบบที่ตรวจจับการบุกรุกบนเครือข่ายที่ติดตั้งระบบนี้ ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อต่อไป

2.2.1 การทำงานระบบตรวจจับผู้บุกรุกบนเครือข่าย

ระบบตรวจจับผู้บุกรุกบนเครือข่ายเป็นระบบที่ทำการตรวจจับ Packet ที่วิ่งบนเครือข่าย เพื่อหาว่ามีการพยายามที่จะโจมตีเครือข่ายหรือไม่ โดยใช้ Network Adapter ที่อยู่ในสถานะ Promiscuous mode (คือโหมดที่ Network Adapter จะดักเอา packet ทุกอย่างที่ปรากฏอยู่) และทำการวิเคราะห์ Traffic บนเครือข่ายในขณะนั้น วิธีการที่ใช้ในการตรวจสอบดูว่ามีการบุกรุกหรือความพยายามที่จะการบุกรุก คือ

- ตรวจสอบ Pattern คือ วิเคราะห์ข้อมูลในลักษณะที่เป็นเครื่องบ่งชี้ (Signature) ที่อยู่ใน packet ที่วิ่งบนเครือข่าย และเปรียบเทียบกับข้อมูลที่เก็บไว้ ซึ่งอาจอยู่ในฐานข้อมูลว่า Signature

นั้นตรงกับรูปแบบใดรูปแบบหนึ่ง ของการโจมตีระบบหรือไม่ การเปรียบเทียบส่วนมากมีอยู่ 3 แบบด้วยกันคือ

1) String Signature คือ ดูในส่วน ที่เป็นข้อความว่าน่าจะเป็นการโจมตีหรือไม่ เช่น สำหรับการโจมตีระบบ UNIX ถ้ามีคำว่า "cat"+ + " > /.rhosts" ก็จะรู้ว่ามีพยายามโจมตีเกิดขึ้น

2) Port Signature คือ ดูว่ามีการติดต่อกับ เบอร์ PORT ที่ไม่ได้เปิดไว้หรือไม่ เช่น SUNRPC(TCP/UDP 111)

3) Header Signature คือ ดูในส่วน Header ของ packet ที่น่าสงสัย เช่น มี ทั้ง SYN และ FIN flag มาในครั้งเดียวกัน

- ตรวจสอบจากสถิติ คือ วิเคราะห์จากสถิติการใช้งานของเครือข่ายที่น่าสงสัยและผิดปกติและที่ได้บันทึกไว้ เช่น พบว่ามีการพยายาม login หลายครั้งติดต่อกันอย่างต่อเนื่อง

2.2.2 สิ่งที่จะทำเมื่อตรวจพบการบุกรุก

เมื่อระบบตรวจจับผู้บุกรุกบนเครือข่ายตรวจพบว่ามีการบุกรุกเข้ามาภายในเครือข่าย ระบบก็จะมีการทำอย่างหนึ่งอย่างใดเพื่อเป็นการป้องกันระบบ ขึ้นอยู่กับความสามารถของระบบตรวจจับผู้บุกรุกนั้น และการที่ผู้ดูแลระบบตั้งค่าให้ทำอะไรในแต่ละกรณี และสิ่งทำได้แก่

- สั่ง firewall ให้ Block IP address ของ ผู้บุกรุก ไม่ให้เข้ามาในระบบได้ต่อไป แต่ผู้บุกรุกก็อาจจะเปลี่ยนไปใช้ IP address อื่นๆ อีก

- ส่งเสียงเตือนให้ได้ยิน เช่น "You are under attack!" เพื่อให้ผู้ดูแลระบบทำการป้องกันทันในเวลานั้น เพราะอาจจะไม่ได้ เฝ้าดู อยู่ตลอดเวลา

- ส่ง SNMP Trap datagram ไปยังโปรแกรมจัดการเครือข่าย เช่น HP Open View, Tivoli

- บันทึกไว้ที่ Log ของระบบ เพื่อให้ผู้ดูแลระบบทำการวิเคราะห์ ในภายหลัง หรือรันโปรแกรมให้วิเคราะห์ Log อยู่ตลอดเวลาเพื่อหาการโจมตี หรือหาช่องโหว่อัตโนมัติ

- ส่ง email แจ้งไปยัง ผู้ดูแลระบบให้ทราบว่ามีการตรวจพบการโจมตีเกิดขึ้น

- ส่ง Page หรือ SMS ให้ผู้ดูแลระบบซึ่งทราบเร็วกว่า email ทำให้ทำการป้องกันได้เร็วกว่า

- เก็บ Log ข้อมูลของผู้บุกรุก เช่น timestamp, IP address, IP address และ port ของผู้ถูกโจมตี, ข้อมูลเกี่ยวกับprotocol ที่ใช้ เพื่อใช้ในวิเคราะห์หาทางป้องกันต่อไป

- สั่งให้โปรแกรมบางโปรแกรมที่ตั้งไว้ทำงาน เช่น สั่งไฟร์วอลให้ทำการป้องกัน หรือให้โปรแกรมส่งข้อมูลการบุกรุกไปยังระบบตรวจจับผู้บุกรุกอื่นๆ เพื่อทำการป้องกันล่วงหน้า

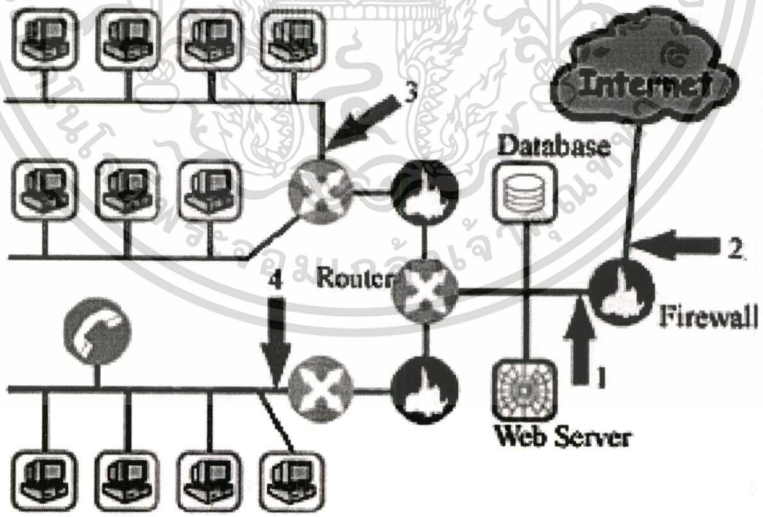
- ส่ง TCP FIN Packet เพื่อยกเลิกการติดต่อ Session ที่ต้องสงสัยว่าเป็นการบุกรุกทันที ในกรณีที่เป็นโปรโตคอล TCP

2.2.3 การติดตั้งระบบตรวจจับผู้บุกรุกบนเครือข่าย

การติดตั้งระบบตรวจจับผู้บุกรุก ในจุดที่ต่างกันบนเครือข่าย ก็จะส่งผลถึงการตรวจจับการบุกรุกที่ไม่เหมือน เพราะแต่ละจุดมีจำนวนการเข้าออกและลักษณะของข้อมูลไม่เหมือนกัน ดังรูปที่ 2.1 ซึ่งมี 4 ตำแหน่งหลักที่ต่างกันดังนี้

ตำแหน่งที่ 1 อยู่หลัง Firewall ที่ต่อกับ Internet

- เห็นการเจาะเข้ามาที่ระบบจากอินเตอร์เน็ตที่ผ่านแนวป้องกันด่านแรกมาได้
- มีข้อมูลมากที่บริเวณนี้ แต่ผ่านการกรองจาก firewall ระดับหนึ่งแล้ว
- เห็นการเจาะที่เข้ามาที่ Web Server, Database ที่อยู่หลัง Firewall
- สามารถตรวจจับ Traffic จาก Server ของเราที่ถูก Compromise ไปแล้ว



รูปที่ 2.1 การวางระบบตรวจจับผู้บุกรุกในตำแหน่งต่างๆ 4 ตำแหน่ง

ตำแหน่งที่ 2 อยู่หน้า Firewall ต่อกับ Internet

- ตรงนี้จะมี Traffic เป็นจำนวนมากที่ผ่านเข้ามาจากอินเตอร์เน็ต สามารถเห็นจำนวน และรูปแบบของ Traffic ที่มาที่เครือข่ายของเราได้ทั้งหมด

ตำแหน่งที่ 3 อยู่ที่ backbone ของ network ภายใน

- มี Traffic ภายในจำนวนมาก
- สามารถตรวจสอบ การโจมตีจากคนข้างใน

ตำแหน่งที่ 4 อยู่ที่ subnet ย่อย

- สามารถเจาะจงการตรวจจับภายในระบบย่อย ซึ่งใช้ทรัพยากรในการทำงานไม่มากนัก

2.3 โปรแกรมตรวจจับการบุกรุก Snort

Snort เป็นโปรแกรมที่ใช้ในการตรวจจับการบุกรุกบนเครือข่าย ซึ่งเป็นแบบเปิดเผยซอร์สโค้ด (Open source) และสามารถติดตั้งได้ในหลายแพลทฟอร์ม โปรแกรม Snort นั้นสามารถทำการวิเคราะห์ข้อมูลที่รับส่งบนเครือข่ายแบบ Real time ,บันทึก Packet, วิเคราะห์โปรโตคอล, ตรวจสอบข้อมูลใน Packet และค้นหาในส่วนที่ต้องการ, ตรวจจับการโจมตีในรูปแบบต่างๆ ได้ เช่น buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts และอื่นๆ เมื่อ Snort ตรวจพบการโจมตีก็สามารถทำการแจ้งเตือนได้ทันที

การตรวจจับ Packet ของ snort นั้นจะใช้ไลบารี่ libpcap ซึ่งเป็นไลบารี่เดียวกับที่โปรแกรม tcpdump ใช้ ซึ่งปกติจะต้องวางเครื่องที่จะใช้ snort ให้อยู่ในตำแหน่งสามารถรับเอา Packet ที่อยู่บนในเครือข่าย เช่น อยู่ระหว่าง Router กับ ระบบเครือข่ายภายใน ซึ่งจะได้ข้อมูลที่ผ่านเข้าออก Router หรือต่อเครื่องไว้กับ Hub และ เปิดโหมดของเน็ตเวิร์คเป็น แบบ Promiscous mode ซึ่งจะรับเอาทุก Packet ที่เข้ามา หรือถ้าเป็น Switch ก็ควรเป็น switch ที่สามารถ Mirror จากพอร์ตดักที่ต้องการมาที่เครื่อง Snort ได้

2.3.1 โหมดการทำงานของ Snort

การทำงานของ Snort นั้นจะมีอยู่ด้วยกัน 3 โหมด คือ

1. **Sniffer Mode** – เป็นการดัก Packet บนเครือข่าย แล้วนำมาแสดงบนหน้าจอว่ามีข้อมูลอะไรบ้างที่วิ่งอยู่บนเครือข่าย

2. **Packet Logger Mode** – เป็นการบันทึกข้อมูลจากการดัก Packet ที่วิ่งบนเครือข่าย เก็บลงบนดิสก์ไว้

3. **Network Intrusion System Mode** – เป็นการวิเคราะห์ข้อมูลบนเครือข่ายเพื่อตรวจหาว่าเป็นลักษณะ Packet ที่เข้ากับรูปแบบใดรูปแบบหนึ่งของการโจมตี ซึ่งจะใช้กฎ (Rule) ในการตัดสินใจว่าการโจมตีหรือไม่ ซึ่งโดยปกติกฎจะถูกเก็บไว้ในไฟล์ snort.conf ซึ่งกฎนั้นมีผลโดยตรงต่อผลการทำงานของ Snort เพราะถ้าไม่มีกฎที่ตรงกับรูปแบบการโจมตีแบบหนึ่งแบบใด Snort ก็จะไม่สามารถตรวจจับการโจมตีแบบนั้นได้ หรือมีกฎที่กว้างเกินไปก็จะทำให้เกิดการเข้าใจ Packet ที่ปกติว่าเป็นการโจมตี ซึ่งจะทำให้เกิดการแจ้งเตือนที่ผิดพลาด (False alert)

2.3.2 ลักษณะของกฎใน Snort

เพื่อจะให้สามารถเข้าใจการทำงานของ Snort มากขึ้น เราจำเป็นที่จะต้องรู้จักลักษณะของกฎที่ Snort จะใช้ในการตรวจจับการบุกรุก ซึ่งจะถูกเขียนลงในไฟล์ snort.conf และเราสามารถปรับปรุงแก้ไขกฎของเราเองได้ ตัวอย่างของกฎจะเป็นดังรูปที่ 2.2

```
var MY_NET [192.168.1.0/24,10.1.1.0/24]

alert tcp any any -> $MY_NET any (flags: S; msg: "SYN \
    packet";)

alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86\
    a5|"; msg: "mountd access";)
```

รูปที่ 2.2 ตัวอย่างของกฎในไฟล์ snort.conf

นี่เป็นตัวอย่างของกฎซึ่งถ้ากฎมีมากกว่า 1 บรรทัดก็ให้ใช้เครื่องหมาย \ (Back slash) ในการเชื่อมบรรทัดต่อไป และเราสามารถกำหนดตัวแปรเพื่อให้สะดวกในการปรับแต่งกฎได้ ซึ่งจะอยู่ในรูปแบบของ var <name> <value> สำหรับในส่วนของกฎจะแบ่งออกเป็น 2 ส่วน คือ Rule Header และ Rule Option

- **Rule Header**

รูปแบบของ Rule Header คือ

<Action> <Protocol> <IP addr/Net mask> <Port> <Direction> < IP addr/Net mask > <Port>

- **Action** เป็นส่วนที่บอกว่าเมื่อพบ Packet ที่ตรงกับ Rule นี้จะให้ทำอะไร ซึ่งได้แก่

1. alert – ทำการแจ้งเตือนว่าพบการบุกรุกซึ่งจะมีหลายวิธีในการแจ้ง และบันทึก Packet นั้นไว้

2. log - บันทึก Packet นั้นอย่างเดียว

3. pass – ไม่สนใจ Packet นั้น ปล่อยให้ผ่านไป ไม่มีการบันทึก

4. activate – ทำการแจ้งเตือน แล้วทำ Dynamic rule ที่กำหนด

5. dynamic – จะไม่ทำอะไร จนกว่าจะได้มีการActivate โดย activate rule ซึ่งทำงานเหมือนกับ Log rule

- **Protocol** เป็นการบอกประเภทของ Protocol ซึ่ง Snort นั้นสามารถวิเคราะห์ได้เพียง 4 Protocol คือ TCP, UDP, ICMP และ IP ส่วนในอนาคตที่จะมีการเพิ่มเติม ได้แก่ ARP, IGRP, GRE, OSPF, RIP, IPX และอื่นๆ

- **IP address / Net mask** คือค่า IP address และ Net mask ของ เครื่องต้นทาง และปลายทางที่ติดต่อ เช่น 192.168.1.0/24 หรือ any ก็จะหมายถึง IP address อะไรก็ได้

- **Port** คือ เบอร์ Port ของเครื่องต้นทาง และปลายทางที่ใช้ในการติดต่อ เช่น 25 (คือ SMTP), 111 ดังตัวอย่างข้างบน หรือ any ก็จะหมายถึง Port อะไรก็ได้

- **Direction** คือส่วนที่บอกทิศทางการของการติดต่อ ได้แก่ -> คือ พิจารณาเพียงทิศทางเดียว ฝั่งซ้ายของเครื่องหมายคือ ต้นทาง ส่วนฝั่งขวา คือ ปลายทาง และ <> คือพิจารณาทั้งสองทิศทาง

● **Rule Option**

ส่วนของ Rule Option เป็นส่วนสำคัญที่ช่วยเพิ่มความสามารถในกฎ ซึ่งจะอยู่ในรูปแบบ

($Keyword_1$: $Argument_1$; $Keyword_2$: $Argument_2$; …. $Keyword_n$: $Argument_n$;)

ซึ่ง Keyword ที่ Snort ใช้ ตัวอย่างเช่น

1) msg: "<message text>";

ระบุข้อความที่จะถูกแสดงใน ตอนแจ้งเตือน(ALERT) และ ตอนบันทึก Packet

2) logto: "<filename>";

บันทึก Packet ไปยังไฟล์ที่ระบุ แทนที่จะเป็นบันทึกรวมกับlog อื่นๆ บนไฟล์มาตรฐาน

3) ttl: "<number>";

ตรวจสอบค่า time-to-liveของ Packet ว่าตรงกับที่ระบุหรือไม่

4) tos: "<number>";

ตรวจสอบค่าใน TOS field ของ Packet ว่าตรงกับที่ระบุหรือไม่

5) id: "<number>";

ตรวจสอบค่าใน fragment ID field ของ IP header ว่าตรงกับที่ระบุหรือไม่ ซึ่งโปรแกรมของแฮกเกอร์บางโปรแกรมระบุค่านี้ไว้ เช่น 31337 เป็นต้น

6) ipopts: <option>;

ดูค่าใน IP option field ว่าตรงกับที่ระบุไว้หรือไม่ ซึ่งใน 1 กฎจะระบุค่า IP option ได้ 1 ค่าเท่านั้น

7) fragbits: <bit values>;

ตรวจ Fragment และ Reserved bit ซึ่งมี bit value 3 ค่า ได้แก่ R แทน Reserved Bit (RB), M แทน More Fragments (MF) bit และ D แทน Don't Fragment (DF) bit

8) dsize: [>|<] <number> [<> <number>];

ระบุขนาด Payload ของ Packet เช่น dsize:100<>500 คือขนาดระหว่าง 100 ถึง 500 Bytes

9) flags: <flag values>[,mask value];

ตรวจค่า TCP Flag ซึ่งมี ค่า Flag 9 ค่าที่ใช้ได้ใน Snort ได้แก่

F = FIN (LSB in TCP Flags byte),

S = SYN

R = RST

P = PSH

A = ACK

U = URG

2 = Reserved bit 2

1 = Reserved bit 1 (MSB in TCP Flags byte)

0 = No TCP Flags Set

10) seq: <number>;

ดูค่า TCP Sequence Number ซึ่งจะระบุเห็นค่าคงที่ ดังนั้นปกติจะไม่ค่อยได้ใช้เพราะค่า Sequence Number จะเปลี่ยนอยู่ตลอด

11) ack: <number>;

ตรวจค่าใน acknowledgement field (ACK) ว่าตรงกับที่ระบุหรือไม่

12) itype: <number>;

ตรวจดูค่าใน ICMP type field ซึ่งค่าในส่วนนี้มักใช้กับการโจมตีแบบ Denial of Service และ flooding attack

13) icode: <number>;

ตรวจดูค่าใน ICMP code field ซึ่งค่าใช้ประโยชน์ได้เหมือนกับ itype

14) icmp_id: <number>;

ดูค่าหมายเลข ICMP ID ของ ICMP ECHO Packet

15) icmp_seq:<number>;

ดูค่า ICMP Sequence number ของ ICMP ECHO Packet

16) content:[!] "<content string>";

เป็นการค้นหาในส่วนของ Packet Payload ว่ามีข้อมูลตรงกับที่ระบุไว้หรือไม่ ซึ่งในส่วน content string สามารถระบุค่าที่เป็น Text รวม กับ Binary ได้ โดยค่าที่เป็น Binary จะปิดหัวท้ายด้วยเครื่องหมาย "|" และการใช้เครื่องหมาย "!" อยู่หลัง ":" หมายถึงการค้นหาว่าไม่มีข้อความนี้ในส่วน Payload ซึ่งจะแสดงตัวอย่างทั้ง 2 แบบ ดังรูปที่ 2.3

```
alert tcp any any -> 192.168.1.0/24 143 (content: "|90C8 \
    C0FF FFFF|/bin/sh"; msg: "IMAP buffer overflow!";)
alert tcp any any -> 192.168.1.0/24 21 (content: !"GET"; \
    depth: 3; nocase; dsize: >100; msg: "Long Non-Get \
    FTP command!";)
```

รูปที่ 2.3 กฎที่ใช้ content ค้นหาคำแบบผสมระหว่าง Text กับ Binary

```
# NOT ALLOW SITE
"www.xxx.com"
"adults"
"hard core"
#...
```

รูปที่ 2.4 ตัวอย่างไฟล์ที่ใช้กับ content-list

17) content-list: "<file_name>";

เป็นการค้นหาในส่วนของ Packet Payload โดยใช้ข้อมูลในไฟล์ซึ่งบรรจุคำที่ใช้ในการค้นหา ตัวอย่างดังรูปที่ 2.4

18) offset: <number>;

เป็น Keyword ที่ใช้ร่วมกับ content ในการบอกตำแหน่งถัดจากจุดเริ่มต้นของ Payload ในการค้นหาคำ มีประโยชน์สำหรับ CGI scan ซึ่งจะไม่ค้นใน 4 Bytes แรก ดังรูปที่ 2.5

19) depth:<number>;

เป็น Keyword ที่ใช้ร่วมกับ content บอกถึงจำนวน byte ที่จะค้นหาใน Payload ซึ่งทำให้ไม่ต้องเสียเวลาในการทำ Pattern matching ทั้งหมดถ้าเรารู้ขอบเขตของข้อมูลที่ต้องการ เช่น การค้นหาคำ "cgi-bin/phf" ใน Web Packet เราแค่ค้นหาในส่วนแรกไม่เกิน 20 bytes ไม่ต้องค้นหาทั้งหมด ซึ่งจะเห็นในรูปที่ 2.5

```
alert tcp any any -> 192.168.1.0/24 80 (content: "cgi-\
bin/phf"; offset: 3; depth: 22; msg: "CGI-PHF access";)
```

รูปที่ 2.5 การใช้ Keyword offset และ depth ร่วมกับ content

20) session: [printable|all];

ใช้ในการบันทึกข้อมูลจาก TCP session เมื่อต้องการรู้ว่ามีการพิมพ์ข้อความหรือเห็นอะไรบ้างในการใช้งาน เช่น telnet , rlogin, ftp หรือ web ซึ่งเลือกได้ว่าจะเก็บเฉพาะข้อมูลที่แสดงผลออกมาได้ (printable) หรือข้อมูลทั้งหมด(all)

21) sid: <snort rules id>;

เป็นการกำหนด หมายเลข IDของกฎ snort เพื่อช่วยในการอ้างอิงกับ Plug-in ที่จะมาใช้งานกฎ หรือใช้อ้างหมายเลข sid กับข้อความที่จะถูกแจ้งเตือนในไฟล์ sid-msg.map

22) classtype:<class name>;

ใช้ในการกำหนด Class ให้กับกฎ ซึ่งแต่ละ Class ก็จะมีค่า Default Priority ที่ถูกแบ่งไว้อยู่แล้ว

23) priority rule severity identifier

ค่า Priority เป็นการบอกถึงระดับสำคัญของกฎซึ่งค่าน้อยยิ่งสำคัญมาก และค่านี้สามารถไปแทนที่ค่า Default Priority ที่มาจาก Classtype ของกฎได้

นอกจากนี้ยังมี Keyword อื่นๆ อีกมากที่ช่วยในการทำให้การใช้งานกฎใน snort มีความยืดหยุ่นและเพิ่มประสิทธิภาพในการตรวจจับการบุกรุก

2.3.3 การแจ้งเตือน (Alert) เมื่อเจอการบุกรุก

เราสามารถกำหนดรูปแบบการแจ้งเตือนของ snort เมื่อพบว่ามีการบุกรุกในเครือข่าย และในส่วนเอาท์พุทโมดูลก็จะทำงานเพื่อรับรับการ Alert หรือ Log โดยปกติจะอยู่ที่ /var/log/snort และรูปแบบของเอาท์พุทเราสามารถกำหนดไว้ในไฟล์ของกฎได้ ซึ่งมีดังต่อไปนี้

1) Alert_syslog

ระบบจะส่ง Alert ไปยัง syslog ของระบบ ซึ่งสามารถที่จะใช้ โปรแกรม swatch หรือ logcheck เป็นตัวที่จะส่งอี-เมล์ให้เราได้

2) Alert_fast

Alert message แต่ละอันจะถูกบันทึกใน 1 บรรทัด ซึ่งจะให้ความรวดเร็วกว่า Alert_full

3) Alert_full

Alert message จะเก็บรายละเอียดของ Packet Header ทั้งหมด

```
[**] [1:483:2] ICMP PING CyberKit 2.2 Windows [**]
[Classification: Misc activity] [Priority: 3]
09/08-17:11:34.592121 203.145.29.77 -> 203.145.7.134
ICMP TTL:124 TOS:0x0 ID:8564 IpLen:20 DgmLen:92
Type:8  Code:0  ID:512  Seq:32775  ECHO
[Xref => http://www.whitehats.com/info/IDS154]
```

รูปที่ 2.6 ตัวอย่างการแจ้งเตือนในไฟล์ alert

4) Alert_smb

ระบบจะส่ง Alert message ไปยัง WinPopUp ผ่านทาง โปรโตคอล SMB ซึ่งในเครื่องที่ได้รับจะ popup ข้อความขึ้นมาให้ทันที

5) Alert_unixsock

จะส่ง Alert message ไปยัง unix socket ซึ่งสามารถรันโปรแกรมอื่นให้ที่รอรับอยู่ที่ socket นี้ได้ทันที แต่ยังอยู่ในช่วงทดลองอยู่

6) Log_tcpdump

จะเก็บบันทึก Packet บนเครือข่ายในรูปแบบของ tcpdump ซึ่งสามารถจะให้โปรแกรมที่รู้จักไฟล์รูปแบบ tcpdump (ซึ่งมีอยู่มาก) ไปทำตรวจสอบต่อไป

7) XML

ให้ XML plug-in เก็บ log ในรูปแบบ SNML (simple network markup language หรือ snort markup language) ซึ่งสามารถส่ง Report ไปยัง Database หรือ snort ตัวอื่นได้ ซึ่งยังอยู่ในช่วงที่กำลังพัฒนาอยู่

8) Database

สามารถบันทึก log ไปยังฐานข้อมูล SQL ได้ซึ่งที่สามารถใช้ได้ตอนนี้ก็ มีMySQL, PostgreSQL, Oracle และunixODBC-compliant database

9) CSV

CSV Plugin จะส่ง alert message ในอยู่รูปแบบที่ง่ายต่อการอิมพอร์ทไปยัง ฐานข้อมูล

10) Unified

Unified output plugin ถูกออกแบบให้มีความเร็วมากที่สุดในการให้บันทึก log ของ snort ซึ่งจะบันทึกไว้ 2 ส่วนคือ ไฟล์Alert จะเก็บรายละเอียดของเหตุการณ์ใน snort ส่วนไฟล์ Log จะข้อมูลเกี่ยวกับ Packet ที่เกี่ยวกับเหตุการณ์ และทั้ง 2 ไฟล์ก็เก็บในรูปแบบของไบนารี่ไฟล์ และต่อไปจะเป็นมาตรฐานสำหรับ snort ด้วย

11) SNMP Trap

SNMP trap output module จะส่ง alert ไปยัง network management station (NMS) ซึ่งจะทำให้ snort ทำงานร่วมกับโปรแกรมอื่นๆ ในมาตรฐานเดียวกันได้

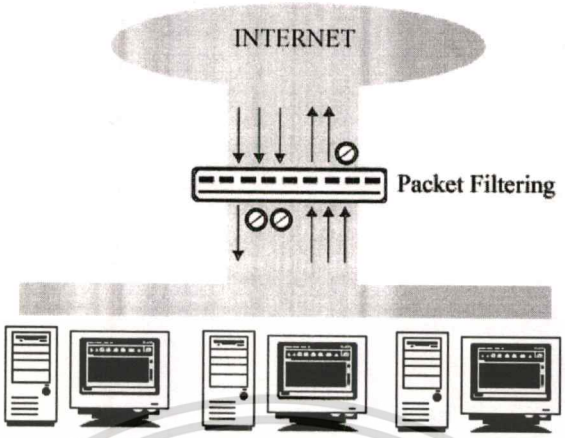
2.4 ไฟร์วอล (Firewall)

ไฟร์วอลเป็นระบบที่ใช้ในการควบคุมการเข้าถึงระหว่างเครือข่ายสองเครือข่าย เพื่อป้องกันเครือข่ายของเราจากการโจมตีของผู้ไม่ประสงค์ดี ซึ่งไฟร์วอลนั้นอาจเป็นได้ทั้ง เราเตอร์ คอมพิวเตอร์หนึ่งเครื่อง หรือหลายๆ เครื่อง ประกอบกัน เป็นต้น

2.4.1 ประเภทการทำงานของไฟร์วอล

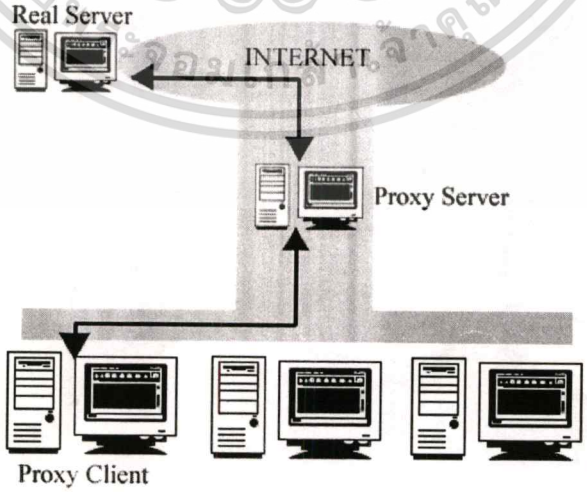
ไฟร์วอลสามารถแบ่งได้ตามวิธีการทำงาน ซึ่งมี 3 แบบ คือ

1) Packet filtering คือ เราเตอร์ หรือคอมพิวเตอร์ที่ทำหน้าที่เป็นเราเตอร์ ที่ทำหน้าที่ Route ข้อมูลระหว่างเครือข่าย โดยจะพิจารณาจากข้อมูลส่วนที่อยู่ใน header ของ Packet ที่ผ่านเข้ามา เทียบกับกฎที่กำหนดไว้ แล้วตัดสินว่าจะยอมให้ Packet นั้นผ่านไปได้หรือทิ้ง Packet นั้นไป ดังรูปที่ 2.7 ซึ่งสิ่งที่ Packet Filtering จะตรวจสอบ ได้แก่ IP address และ Port ของต้นทางและปลายทาง, ชนิดโปรโตคอล, Flag, ชนิด ICMP Message เป็นต้น ซึ่งการทำงานในลักษณะนี้จะทำงานได้เร็ว ไม่ขึ้นกับแอพพลิเคชั่น และยังสามารถรองรับการขยายตัวที่เพิ่มขึ้นได้



รูปที่ 2.7 Router ที่ทำหน้าที่ Packet Filtering

2) Proxy service เป็นแอพพลิเคชันโปรแกรมที่บนไฟร์วอลที่ตั้งอยู่ระหว่างเครือข่าย 2 เครือข่าย ทำหน้าที่ควบคุมการเชื่อมต่อระหว่างเครือข่าย ภายในและภายนอก ซึ่งจะช่วยเพิ่มความปลอดภัยได้มากเนื่องจากมีการตรวจสอบข้อมูลถึงในระดับของแอพพลิเคชันเลเยอร์ (Application Layer) เมื่อเครื่องในเครือข่ายภายในต้องการใช้เชื่อมต่อไปยังภายนอก ก็จะร้องขอไปที่ Proxy ก่อน แล้ว Proxy จะติดต่อไปยังเครื่องปลายทาง ซึ่งจะมี 2 การเชื่อมต่อ คือ เครื่องภายในกับ Proxy และ Proxy กับเครื่องปลายทาง โดยที่ Proxy จะทำหน้าที่รับข้อมูลและส่งต่อข้อมูล ทั้งนี้ ทำหน้าที่ในการตัดสินใจว่าจะให้มีการเชื่อมต่อหรือส่ง Packet ให้หรือไม่ ซึ่งการทำงานจะแสดงไว้ดังรูปที่ 2.8



รูปที่ 2.8 Proxy Server ที่ทำหน้าที่เชื่อมต่อระหว่างเครือข่ายภายในกับภายนอก

Proxy นั้นมีข้อดีที่สามารถตรวจสอบข้อมูลในระดับแอพพลิเคชั่นได้ แต่การทำงานก็จะช้ากว่า Packet Filtering

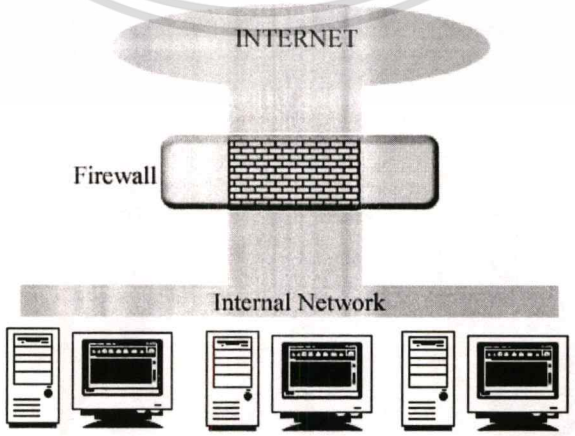
3) Stateful Inspection เป็นเทคโนโลยีที่เพิ่มเข้าไปใน Packet Filtering โดย Packet Filtering แบบทั่วไป (Stateless Inspection) จะตรวจสอบเฉพาะในส่วนของ Header ของ Packet แล้วเทียบกับกฎ ซึ่งจะไม่สามารถทราบว่า Packet นั้นเป็น Packet ใหม่ หรือเป็น Packet ที่ต่อเนื่องกับ Packet ก่อนหน้านี้ ส่วน Stateful Inspection แทนที่จะดูข้อมูลจาก Header เพียงอย่างเดียว ก็ได้เอาส่วนข้อมูลของ Packet (message content) และของPacket ก่อนหน้านี้ที่ได้ทำการบันทึกเอาไว้นำมาพิจารณาด้วย จึงทำให้สามารถรู้ได้ว่า Packet ใดเป็นการเชื่อมต่อเข้ามาใหม่ หรือว่าเป็นส่วนหนึ่งของการเชื่อมต่อก่อนหน้านี้ ตัวอย่างไฟร์วอลที่เป็นแบบ stateful inspection ได้แก่ Check Point Firewall-1 , Cisco Secure Pix Firewall, SunScreen Secure Net และNetFilter ของ Linux (iptables ใน Linux kernel 2.3 ขึ้นไป)

2.4.2 ลักษณะการติดตั้งไฟร์วอลบนเครือข่าย (Firewall Architecture)

ในส่วนการติดตั้งไฟร์วอลนั้น เป็นเรื่องของการจัดวางไฟร์วอลในแบบต่างๆ ซึ่งจะมีผลต่อความสามารถในการรักษาความปลอดภัยบนเครือข่าย ซึ่งมี 4 แบบดังนี้

1) Single Box Architecture

รูปแบบที่ง่ายที่สุด คือมีไฟร์วอลเพียงตัวเดียวติดตั้งอยู่ระหว่างเครือข่ายภายในกับเครือข่ายภายในภายนอก ทำให้ดูแลได้ง่ายเพราะมีเพียงจุดเดียวที่ควบคุมการเข้าออก ดังรูปที่ 2.9 แต่ข้อเสียคือ ทำให้มีความเสี่ยงสูง หากมีไฟร์วอลตัวนี้เกิดทำงานผิดพลาดก็จะมีผลกระทบทั้งเครือข่ายได้



รูปที่ 2.9 Single Box Architecture

ไฟร์วอลที่ใช้ในการติดตั้งแบบนี้นั้นมีอยู่ 3 แบบ

- Screening Router

ใช้เราเตอร์ทำ Packet Filtering วิธีนี้แม้จะให้ไฟร์วอลที่มีความเร็วสูง แต่ไม่ยืดหยุ่นในการปรับแต่งมากนัก เหมาะกับ เครือข่ายที่มีการป้องในระดับโฮสต์ดีแล้ว

- Dual-Homed Host

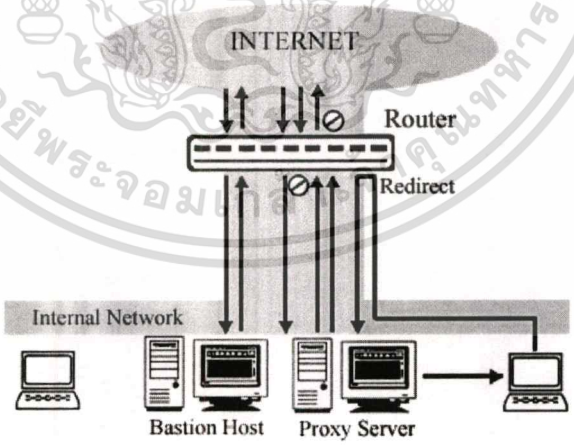
ใช้คอมพิวเตอร์ที่มีอย่างน้อย 2 เน็ตเวิร์กอินเตอร์เฟส บริการเป็น Proxy ให้กับเครื่องภายในเครือข่าย เหมาะสำหรับเครือข่ายที่มีการใช้งานอินเตอร์เน็ตน้อย และไม่ได้มีข้อมูลสำคัญๆ

- Multi-purposed Firewall Box

ทำหน้าที่ได้หลายอย่าง ทั้ง Packet Filtering, Proxy อยู่ในเครื่องเครื่องเดียวกัน

2) Screened Host Architecture

มีเครื่องซึ่งให้บริการ Proxy แต่เครื่องนั้นจะอยู่ภายในเครือข่ายที่ ไม่ต่ออยู่กับเครือข่ายภายนอก (ดังนั้นไม่จำเป็นจะต้องมี 2 เน็ตเวิร์คอินเตอร์เฟส) และจะมี เราเตอร์ ที่ทำหน้าที่บังคับให้เครื่องภายในเครือข่ายต้องติดต่อผ่าน Proxy เท่านั้น ไม่ยอมให้ติดต่อภายนอกโดยตรง และก็ให้ภายนอกเข้าถึงได้เฉพาะ Bastion host (คือเครื่องที่มีความเสี่ยงสูงต่อการถูกโจมตี มักจะเป็นเครื่องที่เปิดให้บริการกับอินเตอร์เน็ต เช่น Web Server, Mail Server) ซึ่งจะต้องมีป้องกันเป็นอย่างดี

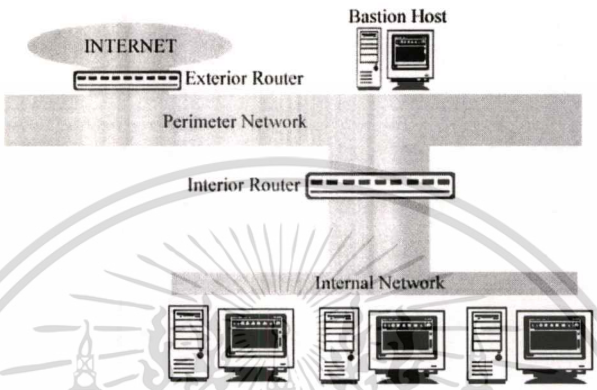


รูปที่ 2.10 Screened Host Architecture

3) Multi Layer Architecture

การวางไฟร์วอลแบบหลายชั้น ซึ่งจะเกิดขึ้นจากหลายๆส่วนทำหน้าที่ประกอบกันในการป้องกันเป็นระบบ ซึ่งจะเพิ่มความปลอดภัยได้มาก เพราะลดความเสี่ยงจากความผิดพลาดจากไฟร์

วอลล์ที่มีเพียงตัวเดียว แต่ถ้ามีหลายชั้น ระบบก็ยังคงมีชั้นอื่นๆ ในการป้องกันอันตราย และแต่ละชั้นนั้นมีการใช้เทคโนโลยีที่แตกต่างกัน ป้องกันช่องโหว่ที่อาจมีในเทคโนโลยีชนิดใดชนิดหนึ่ง โดยทั่วไปจะเป็นการต่อเรียงกันโดยมี Perimeter Network (หรือ DMZ Network) อยู่ตรงกลาง เรียกว่า Screened Subnet Architecture ดังรูปที่ 2.11



รูปที่ 2.11 Screened Subnet Architecture

จากรูปที่ 2.11 เราจะเห็นได้ว่า Screened Subnet Architecture มีการเพิ่ม Perimeter Network เข้าไปกั้นระหว่างอินเตอร์เน็ตกับเครือข่ายภายใน ทำให้มีความปลอดภัยมากกว่าต่อกันโดยตรงเพราะ จะต้องผ่านเราเตอร์ 2 ตัวในการเข้าถึงเครือข่ายภายใน ซึ่งมีส่วนประกอบดังนี้

- **Perimeter Network** เป็นเครือข่ายที่กั้นระหว่างเครือข่ายภายใน กับเครือข่ายภายนอก
- **Bastion Host** จะตั้งอยู่บน Perimeter Network ทำหน้าที่ให้บริการ Proxy และให้บริการต่างๆ กับผู้ใช้บนอินเตอร์เน็ต ดังนั้นมีความเสี่ยงต่อการโจมตีสูง จึงต้องมีการดูแลด้านความปลอดภัยเป็นอย่างดี
- **Interior Router** ตั้งอยู่ระหว่าง Perimeter Network กับเครือข่ายภายใน ทำหน้าที่ Packet Filtering ป้องกันเครือข่ายภายในจาก Perimeter Network
- **Exterior Router** นั้นอยู่ระหว่างเครือข่ายภายนอกกับ Perimeter Network ต้องมีการป้องกันโดยเฉพาะอย่างยิ่ง Packet จากภายนอกที่ปลอม IP Address โดยอ้างว่ามาจากเครือข่ายภายใน

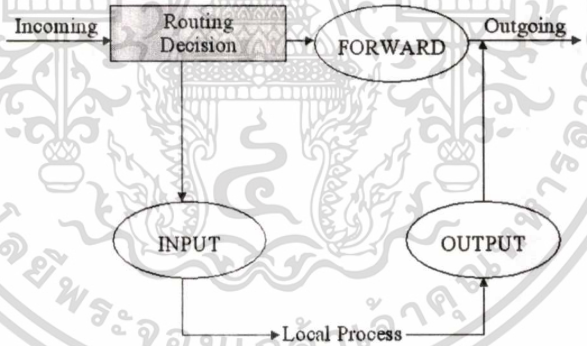
2.5 ไฟร์วอล IPTABLES

ไฟร์วอลที่ใช้ในโครงการนี้คือ IPTABLES ซึ่งเป็นไฟร์วอลที่อยู่ในระบบปฏิบัติการลินุกซ์ เคอร์เนลเวอร์ชั่น 2.4 โดยเป็นไฟล์วอลประเภท Stateful Inspection คือแทนที่จะดูข้อมูลจาก Header เพียงอย่างเดียว ก็ได้เอาส่วนข้อมูลของ Packet (message content) และของPacket ก่อนหน้านี้ที่ได้ทำการบันทึกเอาไว้ นำมาพิจารณาด้วย จึงทำให้สามารถรู้ได้ว่า Packet ใดเป็นการเชื่อมต่อเข้ามาใหม่ หรือว่าเป็นส่วนหนึ่งของการเชื่อมต่อก่อนหน้านี้

2.5.1 การทำงานของ IPTABLES

ไฟร์วอล iptables นั้น มีความสามารถหลักๆ เหมือนกับไฟร์วอลทั่วไปคือ กรอง packet ที่ผ่านเข้าออกระหว่างเครือข่ายซึ่งปกติจะต้องวางขั้นไว้ระหว่าง 2 เครือข่าย iptables สามารถทำงานได้กับ 3 ส่วนหรือตาราง 3 ตารางหลัก คือ

1) Filter table ใช้สำหรับกรอง packet มี 3 chain คือ INPUT, OUTPUT, FORWARD ซึ่งจะใช้เป็นส่วนหลักในการป้องกันระบบเครือข่าย และการทำงานของ Filter table เป็นดังรูปที่ 2.12



รูปที่ 2.12 การเดินทางของ packet ใน filter table

หากจะพิจารณาการเดินทางของ packet เฉพาะในส่วนของ filter table ดังรูปที่ 2.12 โดยเมื่อ packet เข้ามาในระบบ จะเข้าไปยัง routing decision เพื่อตัดสินใจว่า packet จะถูกส่งไปที่ใด

- ในกรณีที่ packet ถูกส่งผ่านไปยังเครื่องอื่น packet นั้นจะต้องถูกตรวจสอบโดย rule ใน FORWARD chain
- ถ้า packet นั้น มีเป้าหมายเป็นเครื่องนี้ (เครื่องที่รัน iptables อยู่นี้) ตัว packet จะถูกตรวจสอบโดย rule ใน INPUT chain

- และในกรณีที่ packet ถูกสร้างจากเครื่องนี้ ตัว packet จะถูกตรวจสอบจาก rule ใน OUTPUT chain ก่อนที่จะถูกส่งออกไป

2) Nat table ใช้สำหรับการแปลงแอดเดรส (Network Address Translation) มี 3 chain คือ PREROUTING, POSTROUTING, OUTPUT

3) Mangle table เป็นตารางที่ใช้เปลี่ยนแปลงหรือแก้ไข packet เช่น เปลี่ยนค่า TTL, MARK ซึ่งปกติจะใช้ในการทำ routing ที่มีความซับซ้อนสูง มี 2 แบบ คือ PREROUTING chain (ใช้แก้ไข packet ก่อนที่จะเข้าสู่ไฟร์วอลและก่อนเข้าสู่ routing decision) และ OUTPUT chain (ใช้แก้ไข packet ที่ถูกสร้างโดยไฟร์วอลล์ก่อนที่มันจะถูกส่งไปยัง routing decision) แต่เป็นส่วนที่ไม่ค่อยนำไปใช้งาน

ซึ่งในการป้องกันเครือข่ายจากการบุกรุกนั้น จะใช้ filter table ส่วนของ INPUT และ FORWARD ระบุ IP address ที่ไม่ต้องการให้ส่ง packet เข้ามายังเครือข่าย

2.5.2 รูปแบบการใช้งาน iptables

เราสามารถสั่ง ไฟร์วอล iptables ในการป้องกันเครือข่ายได้ โดยการทำงานนั้นอยู่ในรูปแบบของกฎ ซึ่งเราสามารถสั่งที่ Command shell บนลินุกซ์ ในการเพิ่มหรือลบกฎ โดยใช้รูปแบบคำสั่งดังนี้คือ

```
# iptables [table] <command> <match> <target/jump>
```

โดยกฎที่เขียนขึ้นจะเป็นเป็นตัวบอกไฟร์วอลว่าให้ทำอะไร ในกรณีที่พบ Packet ตรงตามกฎที่ระบุไว้

- **[table]** หมายถึง ตาราง หรือ table ที่ต้องการระบุ เช่น iptables -t nat หมายถึงให้ทำงานกับ nat table

  ในกรณีที่ไม่ได้ระบุตาราง iptables จะถือว่าคำสั่งดังกล่าวระบุถึง filter table โดยอัตโนมัติ
- **<command>** จะเป็นตัวสั่งให้ iptables ทำในสิ่งที่ต้องการ เช่น iptables -A INPUT ซึ่งหมายถึงให้สร้างกฎต่อท้ายใน INPUT chain
- **<match>** เป็นส่วนที่ใช้ตรวจสอบว่า packet มีข้อมูลตรงกับที่ระบุไว้หรือไม่ เช่น มี source ip address เป็น 202.145.4.140
- **<target/jump>** เป็นตัวระบุว่าเมื่อเจอ packet ที่ match ก็จะกระทำ (action) ตามที่ระบุไว้ เช่น ถ้า packet ใดมี source ip address เป็น 10.10.10.1 ให้ DROP packet นั้นทิ้งไป

สำหรับระบบควบคุมไฟร์วอลที่จะทำขึ้นนั้น จะใช้คำสั่งหลักในการป้องกันเพียงอย่างเดียว คือป้องกัน packet จาก ip address ที่ตรวจพบว่าเป็นการบุกรุก เช่น ระบบตรวจจับการบุกรุกพบการบุกรุกที่มาจาก ip address 203.113.84.10 ไปที่ 202.144.2.1 ระบบจะสั่งไฟร์วอลดังนี้คือ

iptables –A FORWARD –s 203.113.84.10 –d 202.144.2.1 –j DROP

คือทิ้ง packet ที่มี IP address ต้นทางคือ 203.113.84.10 และปลายทางคือ 202.144.2.1 ทุกโปรโตคอล

จากหลักการทำงานของไฟร์วอลจะเห็นว่าเราสามารถควบคุมการทำงานไฟร์วอลด้วยกฎและคำสั่งในการปรับเปลี่ยนกฎมีรูปแบบที่ชัดเจน ทำให้สามารถประยุกต์ใช้ในการทำงานของระบบควบคุมไฟร์วอลที่จะพัฒนาขึ้นได้

บทที่ 3

การออกแบบระบบ

3.1 ความต้องการของระบบ

ความต้องการของระบบควบคุมไฟร์วอลผ่านระบบตรวจจับการบุกรุกนั้นที่จะพัฒนาขึ้น มีดังต่อไปนี้

- เป็นระบบที่สามารถทำงานบนระบบปฏิบัติการลินุกซ์ ซึ่งโปรแกรมที่เขียนขึ้นใช้ภาษา c++ ตัวคอมไพล์ใช้ของ GNU ซึ่งมีอยู่บนระบบลินุกซ์ทั่วไป เช่น Redhat , Slackware, Mandrake เป็นต้น
- ระบบนี้จะทำการวิเคราะห์หาข้อมูลการบุกรุกจากไฟล์ alert ของ Snort โดยข้อมูลที่ต้องการได้แก่ โปรโตคอล, IP address และ Port ของต้นทางกับปลายทาง ซึ่งรูปแบบของไฟล์ alert ที่ระบบสนับสนุนได้แก่ แบบ full และ แบบ fast ซึ่งผู้ใช้สามารถเลือกได้
- เมื่อระบบพบการบุกรุกสามารถสั่งไฟร์วอล IPtables ให้ป้องกันทันทีผ่านสคริปต์ของไฟร์วอล ซึ่งรูปแบบของสคริปต์สามารถที่จะกำหนดหรือเปลี่ยนแปลงได้เองจากผู้ใช้
- ในการสั่งไฟร์วอลสามารถเลือกระดับการป้องกันได้ 2 ระดับ โดยผู้ใช้สามารถเลือกเอง ได้แก่
  - ระดับ IP address คือ ป้องกัน packet ที่มี IP address ของต้นทางกับปลายทางที่ตรงกับข้อมูลที่พบจากไฟล์ alert โดยไม่สนใจว่าเป็น Port ไหน
  - ระดับ Port คือ ป้องกัน packet ที่มี IP address และ Port ของต้นทางกับปลายทาง ที่ตรงกับข้อมูลที่พบจากไฟล์ alert
- สำหรับการบุกรุกที่สั่งให้ไฟร์วอลป้องกันไปแล้วนั้น สามารถกำหนดเวลาให้มีการยกเลิกการป้องกันนั้นได้จากผู้ใช้
- ระบบจะทำงานเป็นโปรเซสที่อยู่เบื้องหลัง และทำงานตลอดเวลาจนกว่าจะสั่งปิดโปรเซส

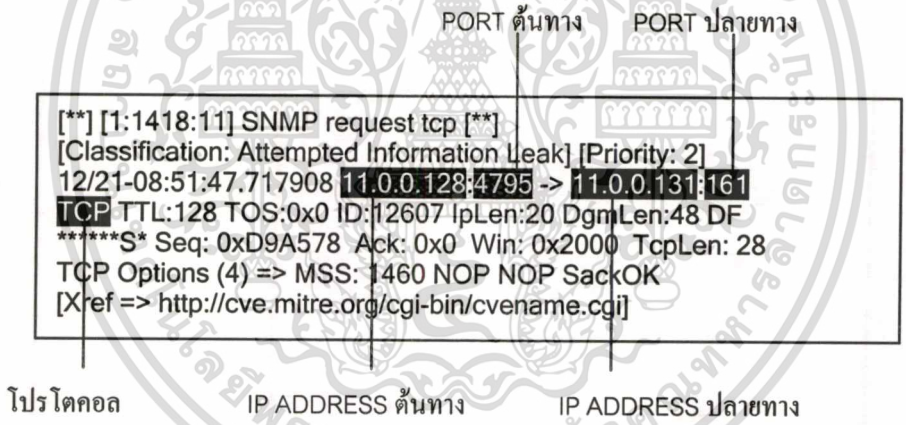
3.2 การออกแบบการทำงานของระบบ

หลังจากได้ศึกษาวิเคราะห์ในการทำงานของ Snort และ ไฟร์วอล IPTABLES แล้ว จึงได้ทำการออกแบบโครงสร้างการทำงานของระบบให้แบ่งเป็นสองส่วนหลักๆคือ

1) ส่วนตรวจสอบการแจ้งเตือนจาก SNORT

โดยส่วนนี้จะทำการอ่านไฟล์ alert ของ SNORT ซึ่งเป็นเท็กซ์ไฟล์ที่เก็บข้อมูลของการบุกรุกที่ตรวจพบ ระบบจะอ่านข้อมูลของจากไฟล์นี้ แล้วดึงเอาส่วนของ IP ADDRESS PROTOCOL PORT ของผู้บุกรุก และเครื่องเป้าหมาย ซึ่งการที่จะอ่านข้อมูลจากไฟล์ alert ได้จะต้องศึกษารูปแบบของการบันทึก โดยปกติผู้ใช้ Snort สามารถระบุรูปแบบของไฟล์ได้เอง ซึ่งมี 2 รูปแบบดังนี้

- **แบบ full** เป็นแบบที่ใช้เป็นมาตรฐาน หากไม่ได้ระบุเป็นรูปแบบอื่นก็จะใช้แบบ full ซึ่งตัวอย่างเป็นดังรูปที่ 3.1



รูปที่ 3.1 ตัวอย่างการแจ้งเตือนในไฟล์ alert แบบ full

ซึ่งการบุกรุก 1 ครั้งจะมีข้อมูลถูกบันทึกเป็นรูปแบบ ดังรูปที่ 3.2

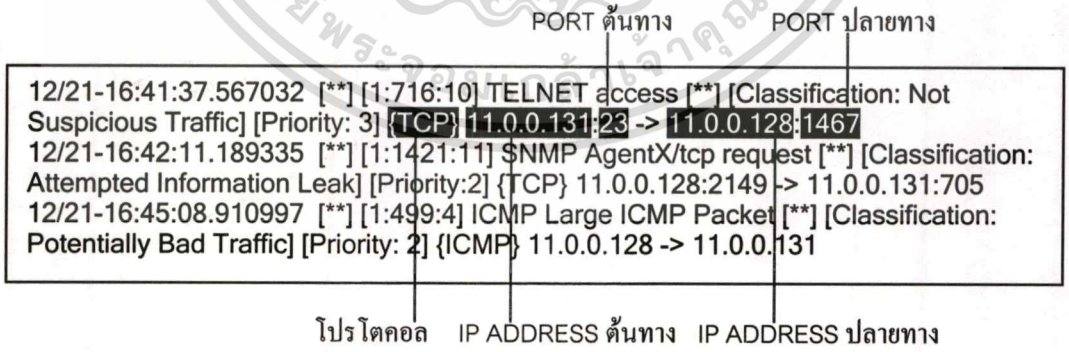
```
<Type>
<Class>
<Timestamp> <Src IP>:<Src Port> -> <Des IP>:<Des Port>
<Protocol> <Packet Header>
<Protocol Option>
<Reference>
```

รูปที่ 3.2 รูปแบบการบันทึกไฟล์ alert แบบ full

ตารางที่ 3.1 ความหมายของข้อมูลในไฟล์ alert แบบ full

ชื่อ	ความหมาย	ประเภทของข้อมูล
Type	ชื่อประเภทการบุกรุก	ข้อความ
Class	หมวดหมู่ของการบุกรุก	ข้อความ
Timestamp	เวลาที่ตรวจพบ	วันที่ เวลา
Src IP	IP address ต้นทาง	ตัวเลข IP address
Src Port	Port ต้นทาง	ตัวเลข
Des IP	IP address ปลายทาง	ตัวเลข IP address
Des Port	Port ปลายทาง	ตัวเลข IP
Protocol	โปรโตคอลที่ใช้	ข้อความ
Packet Header	ค่า optionของ IP packet	ข้อความ
Protocol Option	ค่า optionของ โปรโตคอล	ข้อความ
Reference	แหล่งข้อมูลอ้างอิง อธิบายรายละเอียดของการบุกรุกที่ตรวจพบ	ข้อความ

- **แบบ fast** เป็นการบันทึกข้อมูลหลักๆ เท่านั้น ดังนั้นจะบันทึกแค่บรรทัดเดียวในแต่ละครั้ง ซึ่งเวลาใช้งาน Snort ต้องใช้อาร์กรูเมนต์ –A fast ซึ่งรูปแบบเป็นดังรูปที่ 3.3



รูปที่ 3.3 ตัวอย่างการแจ้งเตือนในไฟล์ alert แบบ fast

ซึ่งการบุกรุก 1 ครั้งจะมีข้อมูลถูกบันทึกเป็นรูปแบบ ดังรูปที่ 3.4

```
<Timestamp> <Type> <Class> <Protocol> <Src IP>:<Src Port> -> <Des IP>:<Des Port>
```

รูปที่ 3.4 รูปแบบการบันทึกไฟล์ alert แบบ fast

ตารางที่ 3.2 ความหมายของข้อมูลในไฟล์ alert แบบ fast

ชื่อ	ความหมาย	ประเภทของข้อมูล
Timestamp	เวลาที่ตรวจพบ	วันที่ เวลา
Type	ชื่อประเภทการบุกรุก	ข้อความ
Class	หมวดหมู่ของการบุกรุก	ข้อความ
Protocol	โปรโตคอลที่ใช้	ข้อความ
Src IP	IP address ต้นทาง	ตัวเลข IP address
Src Port	Port ต้นทาง	ตัวเลข
Des IP	IP address ปลายทาง	ตัวเลข IP address
Des Port	Port ปลายทาง	ตัวเลข IP

ซึ่งเมื่อทราบรูปแบบการบันทึกข้อมูลของไฟล์ alert แล้วก็สามารถให้ระบบที่พัฒนาขึ้นอ่านข้อมูลในส่วนที่ต้องการได้

2) ส่วนของการสั่งไฟร์วอล

เป็นส่วนที่ทำการสั่งไฟร์วอลให้ป้องกัน packet จาก IP Address ของผู้บุกรุก และสามารถทำการยกเลิกการป้องกัน ที่เคยสั่งไปแล้วไว้ได้ เมื่อถึงเวลากำหนด และรูปแบบในการสั่งไฟร์วอลให้ทำงานเป็นดังนี้

- การป้องกันในระดับ IP คือป้องกัน packet ที่มี IP address ต้นทางกับปลายทางตรงกัน

- ทำการป้องกัน (Block)

```
iptables –A INPUT  -p <protocol> -s <ipaddress> -d <ipaddress>  -j DROP
```

- ยกเลิกการป้องกัน (Unblock)

```
iptables –D INPUT  -p <protocol> -s <ipaddress> -d <ipaddress> -j DROP
```

- การป้องกันในระดับ Port คือป้องกัน packet ที่มี IP address และ Port ต้นทางกับปลายทางตรงกัน

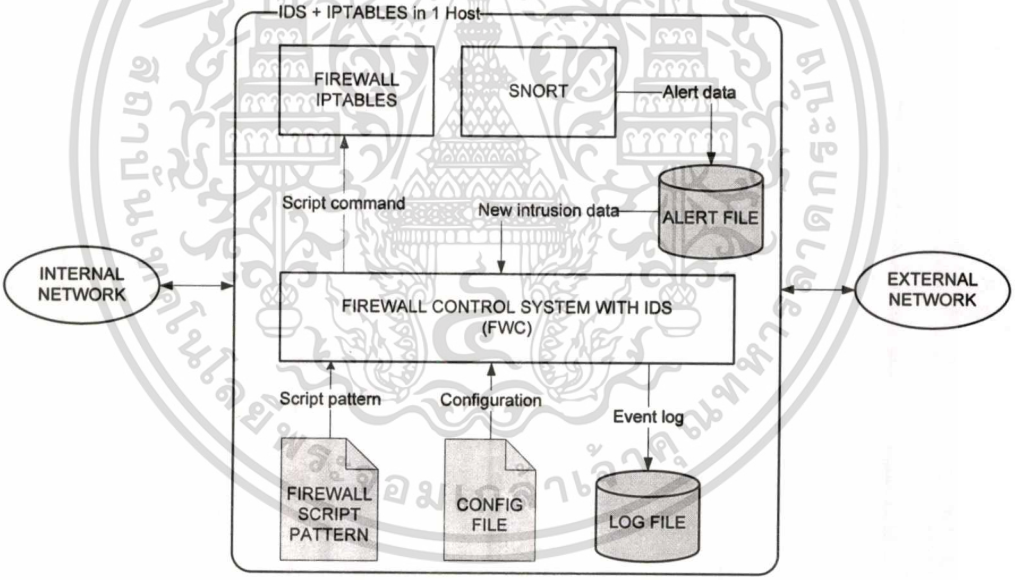
- ทำการป้องกัน (Block)

iptables –A INPUT -p <protocol> -s <ipaddress> --sport <port> -d <ipaddress> --dport <port> -j DROP

- ยกเลิกการป้องกัน (Unblock)

iptables –D INPUT -p <protocol> -s <ipaddress> --sport <port> -d <ipaddress> --dport <port> -j DROP

3) การทำงานของโปรแกรมร่วมกันกับ Snort และ IPTables



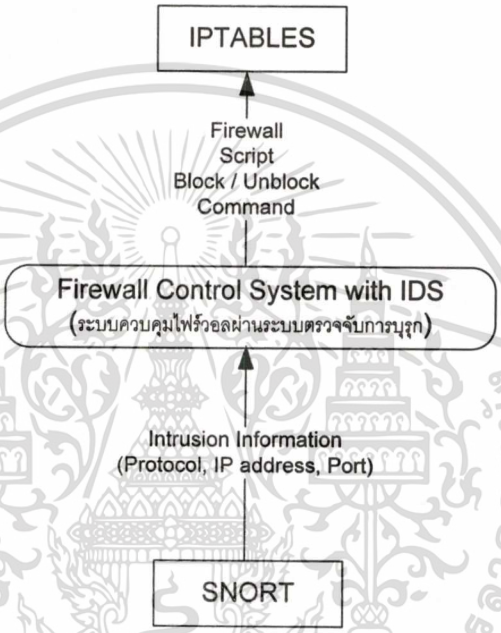
รูปที่ 3.5 ระบบเครือข่ายที่ SNORT อยู่บนเครื่องเดียวกับไฟร์วอล

จากรูปจะเห็นว่าระบบที่ทำขึ้นจะลง Snort และ IPTables ไว้ที่เครื่องเดียวกัน ซึ่งเป็นไฟร์วอลกั้นระหว่างเครือข่าย เมื่อโปรแกรมเริ่มทำงานก็จะอ่านไฟล์ Configuration และอ่านไฟล์รูปแบบของสคริปต์ไฟร์วอล โดยเมื่อ Snort อยู่ซึ่งบนเครื่องเดียวกับไฟร์วอล IPtables มีการตรวจจับการบุกรุกได้ ก็จะบันทึกข้อมูลการบุกรุกไว้ที่ไฟล์ alert ระบบควบคุมไฟร์วอลก็จะทำการอ่านไฟล์ alert เป็นช่วงๆ ถ้าพบการบุกรุกใหม่ก็จะอ่านข้อมูลเฉพาะ protocol, IP Address, Port ทั้ง

ต้นทางและปลายทาง เมื่อได้ข้อมูลดังกล่าว ก็รันสคริปต์สั่งไฟร์วอลให้ป้องกัน packet ที่มี protocol, IP address, Port ปลายทางและต้นทางที่ตรงกันกับที่ตรวจพบ

3.2.1 Data Flow Diagram

- Context Diagram



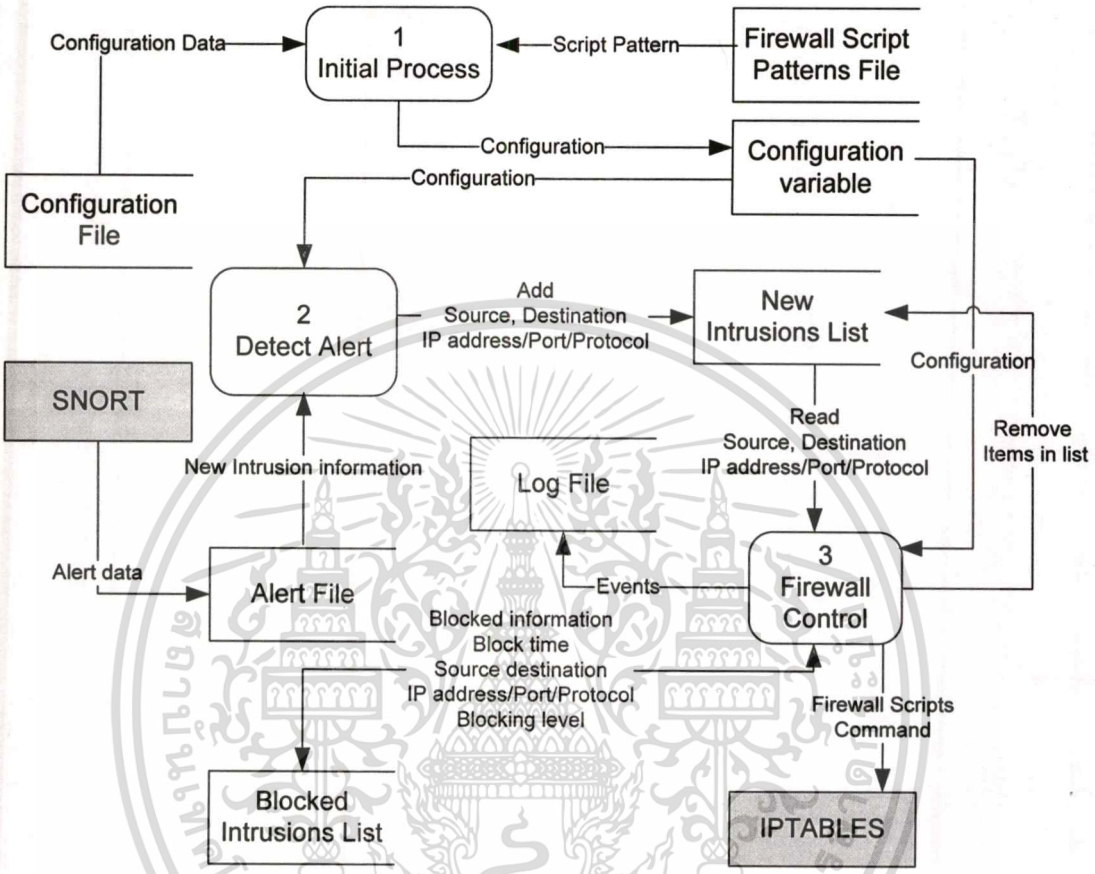
รูปที่ 3.6 Context diagram ของระบบ

จากรูปที่ 3.6 แสดงถึง Context diagram ของระบบ จะเห็นว่า ส่วนต่างๆจะทำงานดังนี้

Firewall Control System จะทำงานร่วมกับ 2 อินเตอร์เฟส คือ

- Firewall IPtables – โดยระบบจะทำการสั่งไฟร์วอลผ่านสคริปต์คำสั่ง คือ ทำการ Block และ ยกเลิกการ Block
- Snort – ระบบจะอ่านข้อมูลการบุกรุกที่เข้ามาในเครือข่ายจากโปรแกรม Snort ซึ่งข้อมูลที่ต้องการได้แก่ Protocol, IP address, Port ทั้งของต้นทางและปลายทาง

- Data Flow Diagram Level 1: Firewall Control System with IDS



รูปที่ 3.7 Data Flow Diagram Level 1

จากรูปที่ 3.7 แสดงถึง Data Flow Diagram Level 1 จะเห็นว่า ส่วนต่างๆจะทำงานดังนี้

- Initial Process (1)

เป็นส่วนที่กำหนดค่าเริ่มต้นของระบบ โดยการอ่านจากไฟล์ Configuration และกำหนดรูปแบบการสั่งไฟร์วอลจะอ่านจากไฟล์ Firewall Script Patterns และเก็บไว้เป็นตัวแปรแบบ Global ให้ process อื่นเข้ามาอ่านได้

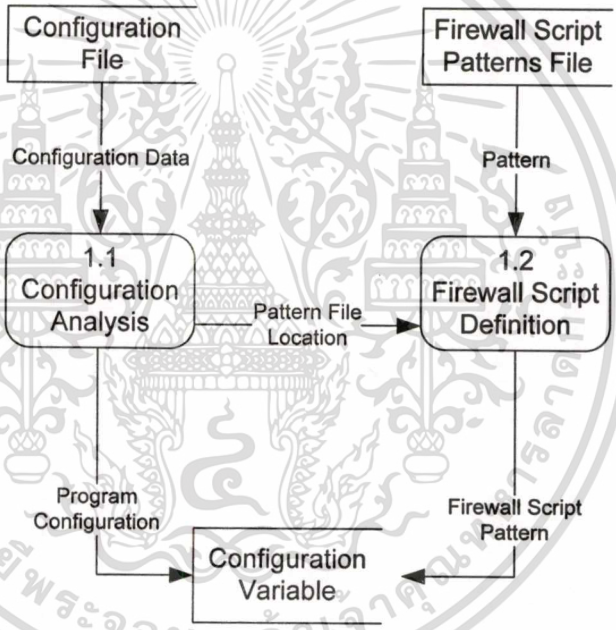
- Intrusion Check (2)

ทำหน้าที่ตรวจสอบว่ามีการบุกรุกหรือไม่ โดยการวิเคราะห์จากไฟล์ alert ที่บันทึกโดยโปรแกรม Snort และดูว่าเป็นการบุกรุกใหม่ แล้วเก็บข้อมูลการบุกรุกไว้ที่รายการ Intrusion list

- Firewall Control (3)

ทำหน้าที่อ่านข้อมูลรายการการบุกรุกจาก Intrusion list ได้แก่ IP address, Port, Protocol ทั้งต้นทางและปลายทาง แล้วสั่งไฟร์วอลให้ป้องกันการบุกรุก จากนั้น ลบรายการนั้นออกจาก Intrusion list และจะทำการบันทึกการป้องกันที่สั่งไฟร์วอลไปแล้วไว้ที่ Blocked List และคอยตรวจดูใน Blocked List ว่ามีรายการของสคริปต์ไฟร์วอลที่ถึงเวลาปลดออกหรือไม่ ถ้ามีก็ทำการสั่งไฟร์วอลให้ปลดออก ทั้งการ Block และ การปลด Block จะบันทึกลงไฟล์ log ไว้ด้วย

- Data Flow Diagram Level 2 Process 1



รูปที่ 3.8 Data Flow Diagram Level 2 Process 1

จากรูปที่ 3.8 จะเห็นว่า ส่วนต่างๆจะทำงานดังนี้

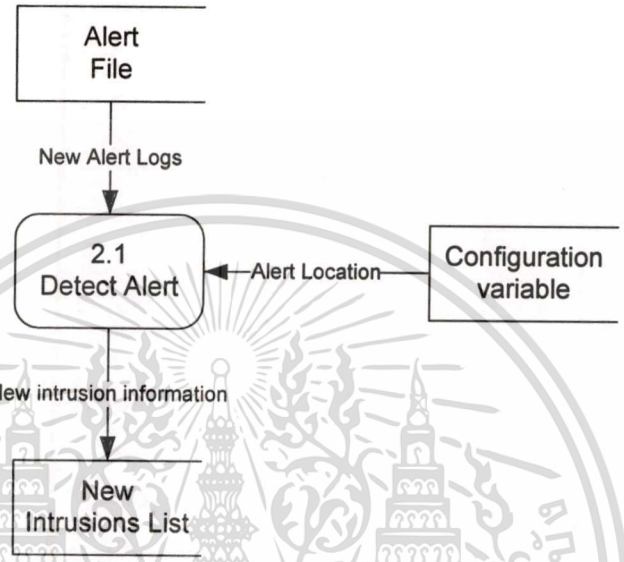
- Configuration Analysis (1.1)

เป็นส่วนที่อ่านจากไฟล์ Configuration แล้ววิเคราะห์ค่าเริ่มต้นของระบบ และบอกตำแหน่งและชื่อของไฟล์ Firewall Script Patterns เก็บค่าเริ่มต้นทั้งหมดให้กับระบบ

- Firewall Script Definition (1.2)

เป็นส่วนที่อ่านค่าจากไฟล์ Firewall Script Patterns เพื่อกำหนดรูปแบบของสคริปต์ในการสั่งไฟร์วอล เก็บไว้เป็นตัวแปรเริ่มต้นให้กับระบบ

- Data Flow Diagram Level 2 Process 2



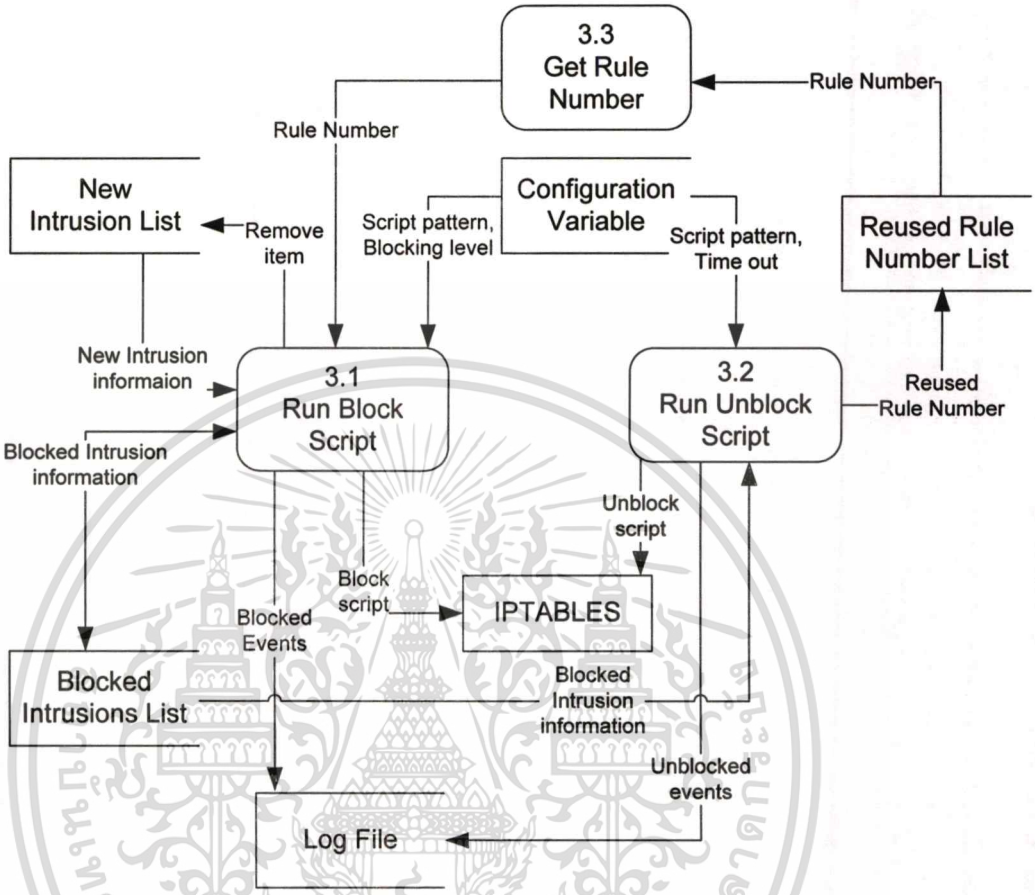
รูปที่ 3.9 Data Flow Diagram Level 2 Process 2

จากรูปที่ 3.9 ส่วนต่างๆจะทำงานดังนี้

- Detect Alert (2.1)

อ่านค่าตำแหน่งของไฟล์ alert จาก Configuration variable แล้วเข้าไปอ่านข้อมูลการบุกรุกใหม่จากไฟล์ Alert จากนั้นตรวจหาข้อมูลการบุกรุก ได้แก่ IP address, Port, Protocol แล้วส่งไปที่รายการ New Intrusion list

- **Data Flow Diagram Level 2 Process 3**



รูปที่ 3.10 Data Flow Diagram Level 2 Process 3

จากรูปที่ 3.10 ส่วนต่างๆจะทำงานดังนี้

- **Run Block Script (3.1)**

อ่านข้อมูลการบุกรุกใหม่จาก New Intrusion List แล้วดูว่ามีใน Blocked intrusion list ว่าถ้าไม่มีแสดงว่ายังไม่เคยสั่ง Block การบุกรุกนี้ จากนั้นจึงสั่ง IPtables ผ่านสคริปต์ ซึ่งรูปแบบนั้นได้มาจาก Configuration Variable เมื่อสั่งไฟล์วอลแล้วจึงลบข้อมูลออกจาก New Intrusion List แล้วบันทึกข้อมูลการสั่ง Block ไว้ที่ ไฟล์ log ถ้ามีการใช้ Rule Number ด้วยก็จะเอาหมายเลขจาก Get Rule Number

- **Run Unblock Script (3.2)**

อ่านข้อมูลการบุกรุกที่เคยสั่ง Block จาก Blocked Intrusion List แล้วดูว่าเวลาตอนที่สั่งจนถึงปัจจุบันเกินหรือเท่ากับเวลา time out ที่กำหนดไว้หรือไม่ ถ้าเกิน ก็สั่งปลด Block ที่ IPtables โดยผ่านสคริปต์ ซึ่งรูปแบบนั้นได้มาจาก Configuration Variable เมื่อสั่ง

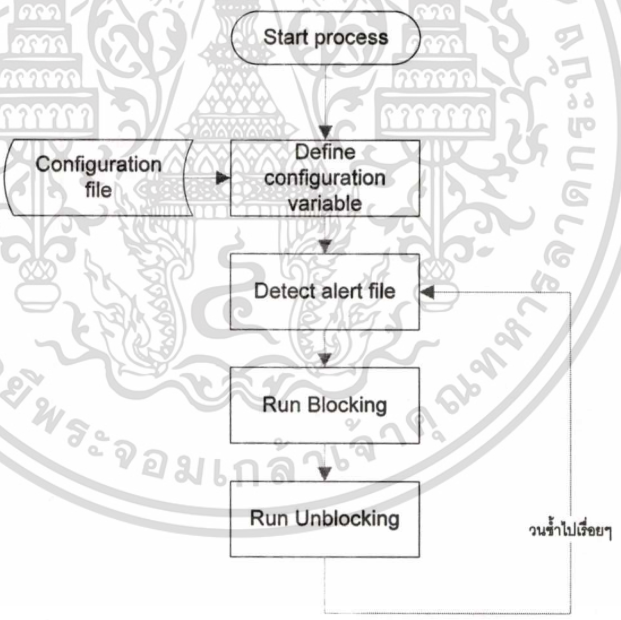
ไฟล์วอลแล้วจึงลบข้อมูลออกจาก Blocked Intrusion List แล้วบันทึกข้อมูลการสั่งปลด Block ไว้ที่ ไฟล์ log ถ้ามีการใช้ Rule Number ด้วยก็จะเอาหมายเลข Rule Number ที่ปลด Block แล้วเพิ่มเข้าไปที่รายการ Reused Rule Number List

- Get Rule Number (3.3)

ทำหน้าที่ส่งหมายเลขกฎ (Rule Number) ไปให้ส่วนของ Run Block Script เพื่อไว้สร้างสคริปต์ไฟร์วอล โดยหมายเลขนั้นจะนั้นมาจากรายการ Reused Rule Number List ถ้าไม่มีก็สร้างเป็นเลขใหม่ที่ยังไม่เคยถูกใช้

3.2.2 Flow Chart ของ Process หลัก

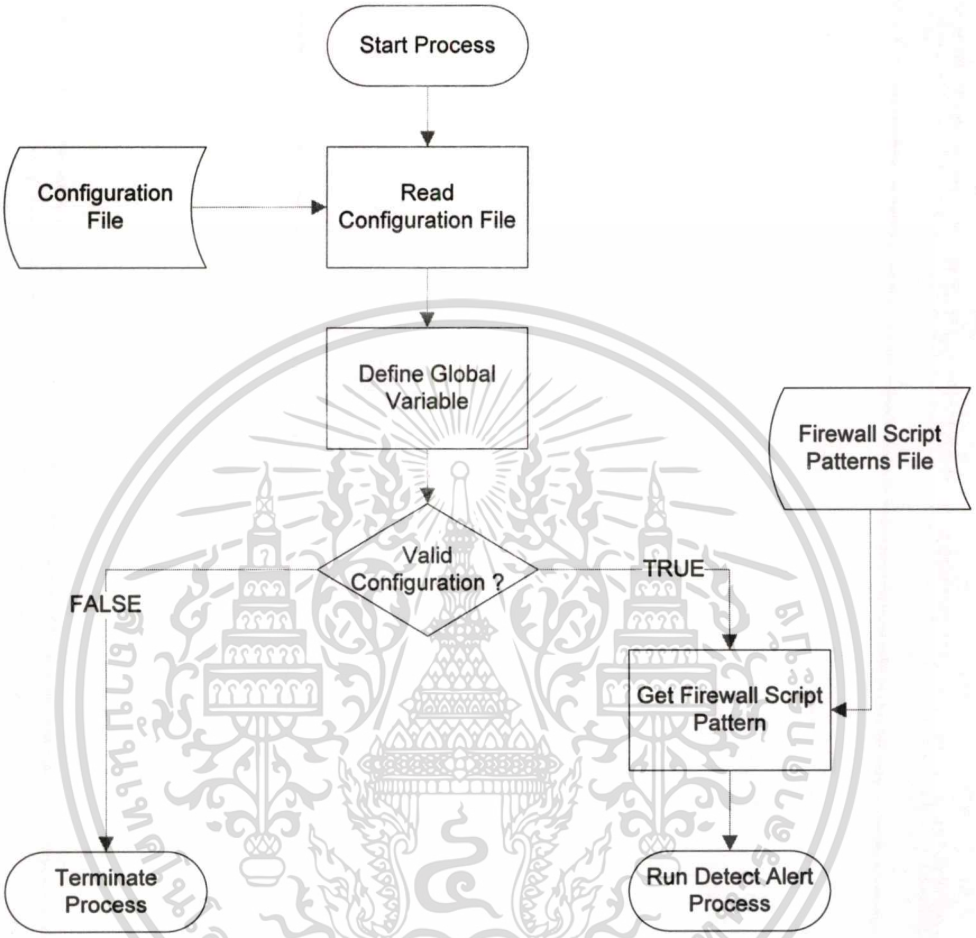
- Main Flow Chart



รูปที่ 3.11 Flow Chart ของโปรแกรมทั้งหมด

จากรูปที่ 3.11 จะเห็นว่าเมื่อเริ่มต้นโปรแกรมทำการกำหนดค่าตัวแปรเริ่มต้นของโปรแกรม (Define Configuration variable) โดยอ่านจาก Configuration file แล้วจึงการวนซ้ำไปเรื่อยๆ โดยเริ่มจากหาการบุกรุก คือ Detect alert file ต่อมาจึงมาที่ Run blocking เพื่อสั่งไฟร์วอลป้องกันการบุกรุกใหม่ที่เข้า ต่อมา Run unblocking จะหาว่าการบุกรุกไหนที่ถึงเวลาปลด ก็ทำการปลดการป้องกันการบุกรุกนั้น แล้วก็กลับมาที่ Detect alert file ทำการหาการบุกรุกใหม่ต่อไป

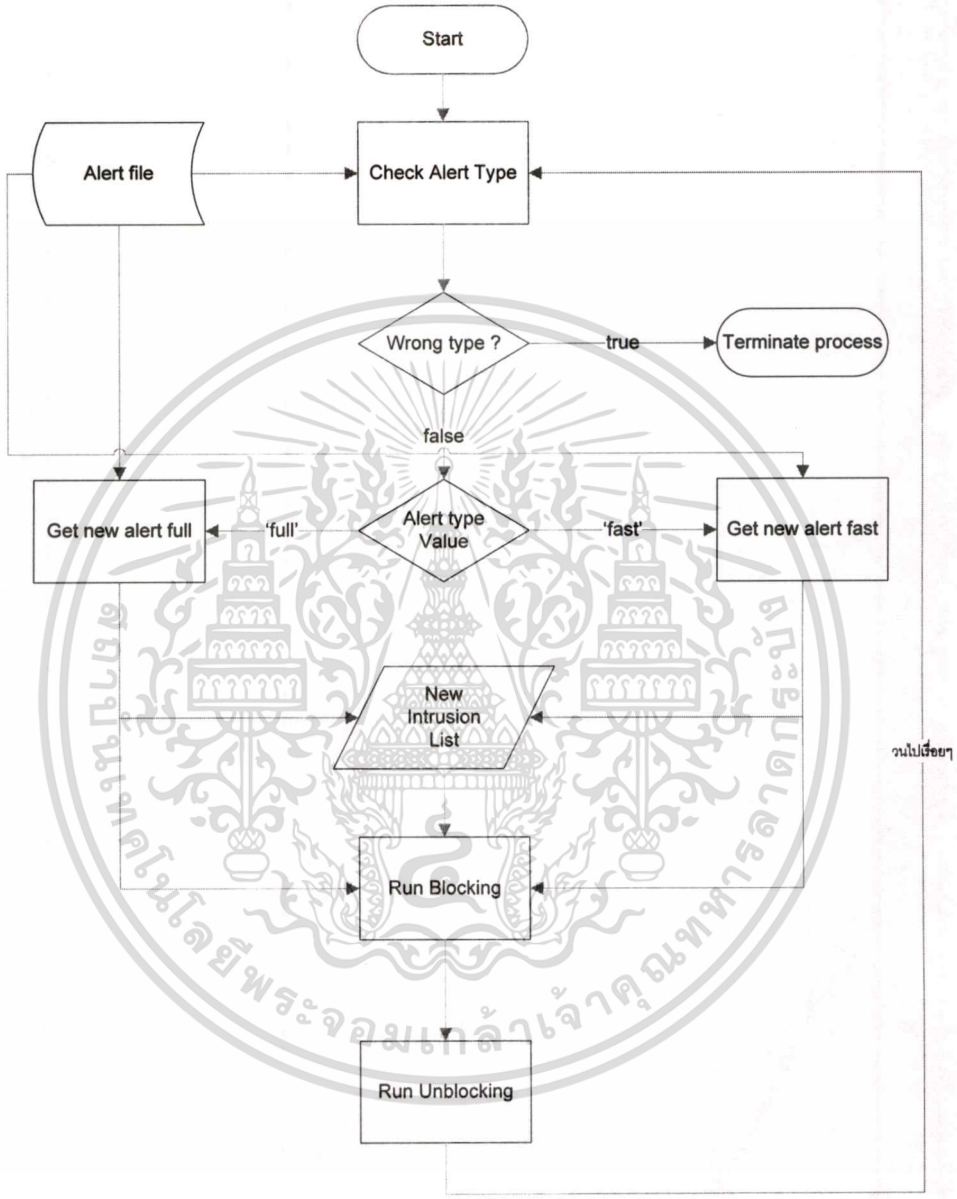
- Define Configuration Variable Flow Chart



รูปที่ 3.12 Flow Chart ของส่วน Define Configuration

จากรูปที่ 3.12 จะเห็นว่าเริ่มต้นก็จะทำการอ่านไฟล์ Configuration ซึ่งเก็บค่าตัวแปรที่กำหนดจากผู้ใช้ แล้วทำการกำหนดค่าเริ่มต้นของระบบที่ Define Initial Variable ถ้าค่าที่กำหนดถูกต้องอยู่ขอบเขตที่โปรแกรมยอมรับ ก็จะไปที่ Get Firewall Script Patern ซึ่งจะทำการอ่านรูปแบบของสคริปต์ไฟร์วอลมาเก็บไว้ จากนั้นจึงจะไปที่ Run Detect Alert ต่อไป

- Detect Alert Flow Chart



รูปที่ 3.13 Flow Chart ของส่วน Detect Alert

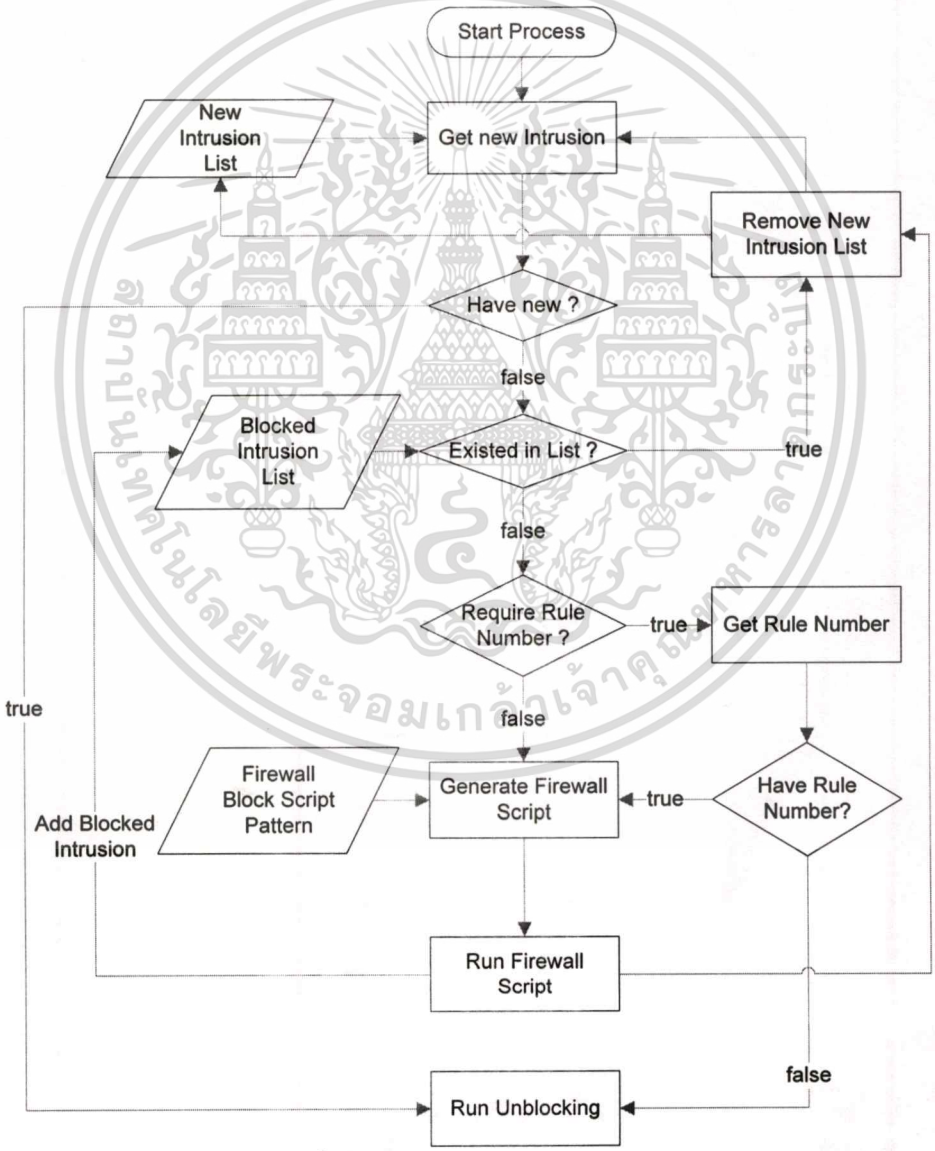
จากรูปที่ 3.13 Detect Alert จะเริ่มต้นทำการอ่านไฟล์ alert โดยอ่านเฉพาะข้อมูลที่เข้ามาใหม่ แล้วตรวจสอบรูปแบบของไฟล์ alert ว่าเป็นแบบ full หรือ fast แต่ถ้าเป็นรูปแบบที่ไม่รู้จักก็จะออกจากโปรแกรม

ถ้ารูปแบบของไฟล์ alert เป็นแบบ full ก็จะไปทำงานที่ Get new alert full แต่ถ้าเป็นแบบ fast ก็จะทำงานที่ Get new alert fast ซึ่งกระบวนการทั้งสองก็เป็นการเอาข้อมูลการบุกรุกจากไฟล์

alert ซึ่งจะได้ค่า IP address และ Port ของต้นทางกับปลายทาง รวมทั้ง Protocol ด้วย ไปบันทึกข้อมูลในรายการNew Intrusion List

จากนั้นจึงทำกระบวนการ Run Blocking เพื่อสั่งไฟล์วอลให้ Block การบุกรุกที่ตรวจพบ โดยอ่านข้อมูลจาก New Intrusion List เสร็จแล้วจึงไปที่กระบวนการ Run Unblocking ซึ่งทำการตรวจดูการ Block ที่ผ่านมาว่ารายการใดถึงเวลาที่จะปลดออก โดยดูจากค่า time out ที่กำหนดที่ Configuration file แต่ถ้าไม่ได้กำหนดก็จะไม่มีการปลด Block

- Run Blocking Flow Chart



รูปที่ 3.14 Flow Chart ของส่วน Run Blocking

จากรูปที่ 3.14 กระบวนการ Run Blocking เริ่มด้วยการอ่านรายการการบุกรุกที่เข้ามาใหม่ค่าใน New Intrusion List ถ้าไม่มีก็ไปที่ Run Unblocking แต่ถ้ามีเข้าใหม่ก็ดูว่าข้อมูลตรงกับใน Blocked Intrusion List หรือไม่

ถ้าตรงก็แสดงว่ามีการ Block การบุกรุกนี้ไว้อยู่แล้ว ก็ไปที่ส่วน Remove new intrusion list เพื่อลบรายการนี้ แล้วไปเอาข้อมูลการบุกรุกใหม่ต่อไปที่ Get new intrusion แต่ถ้ายังไม่ได้ Block ก็ไปทำงานที่ส่วนที่บอกว่าในการรันสคริปต์นั้นต้องการใช้หมายเลขกฎ(Rule number) หรือไม่ ถ้าไม่ต้องการก็ไปที่ส่วน Generate firewall script แต่ถ้าต้องการก็ไปเอาค่า Rule Number จาก Get rule number ถ้าได้หมายเลขค่อยไปที่ Generate firewall script แต่ถ้าไม่ได้หมายเลขแสดงว่ามีการใช้หมดแล้ว ให้ไปที่ทำงานที่ Run unblocking

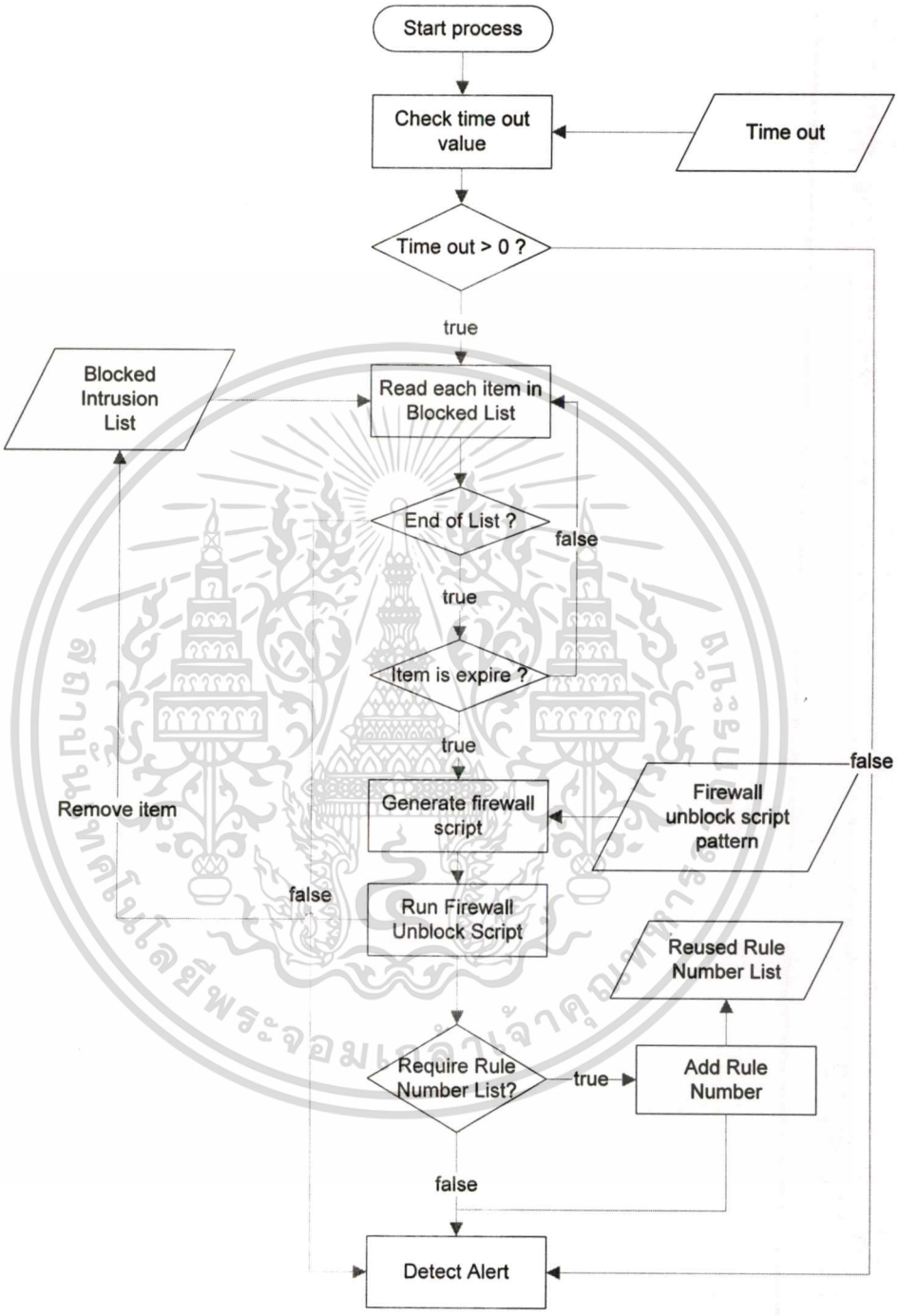
ส่วน Generate firewall script จะทำการสร้างสคริปต์ของไฟร์วอล โดยเอาข้อมูลการบุกรุก คือ Protocol, IP Address, Port ที่ได้จาก New Intrusion List และ Rule Number มาแทนที่ตามรูปแบบของสคริปต์ไฟร์วอล จาก Firewall block script pattern

เมื่อได้สคริปต์ของไฟร์วอลแล้วก็ส่งสคริปต์นี้ไปทำงานที่ Run firewall script ตรงนี้จะเป็นการสั่งไฟร์วอลตามสคริปต์ที่ได้รับ จากนั้นก็เพิ่มข้อมูลการสั่งไปไว้ที่ Blocked Intrusion List เพื่อเก็บรายการที่เคยสั่งไปแล้ว จากนั้นก็ไปที่ส่วน Remove new intrusion list เพื่อลบรายการนี้จาก New intrusion list แล้วไปเอาข้อมูลการบุกรุกใหม่ต่อไปที่ Get new intrusion

- Run Unblocking Flow Chart

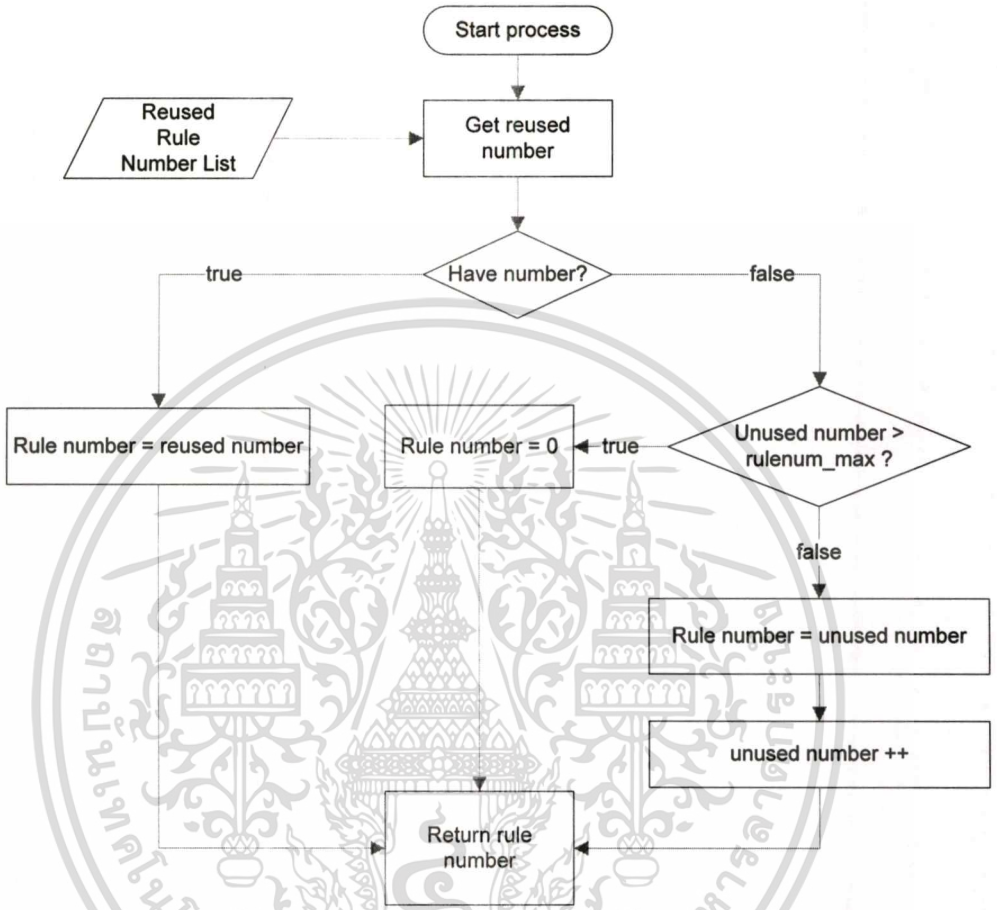
จากรูปที่ 3.15 กระบวนการ Run Unlocking เริ่มด้วยการตรวจว่ามีการตั้งค่า time out มากกว่า 0 หรือไม่ ถ้าไม่ก็กลับไปทำงานที่ Detect Alert ถ้ามีก็อ่านรายการที่เคยสั่ง Block ที่ Read item in blocked list โดยจะอ่านทีละรายการจนกว่าจะหมดรายการ ถ้ารายการหมดก็ไปที่ Detect Alert แต่ถ้ายังมีรายการของการสั่ง Block ที่ผ่านมาก็ให้ตรวจดูว่ารายการนั้นถึงเวลาที่ต้องปลดหรือยัง โดยดูว่ารายการสั่งมานานเกิน time out หรือไม่ถ้าไม่เกิน ก็ไปอ่านรายการต่อไป

ถ้ามีรายการที่เกิน time out ก็ทำการสร้างสคริปต์ที่ Generate firewall script โดยใช้ firewall unblock script pattern เป็นรูปแบบในการสั่งปลด Block รวมกับข้อมูลที่เคยสั่งไป เช่น Protocol, IP Address, Port, Rule number เมื่อสร้างสคริปต์เสร็จก็ส่งไปยัง Run firewall unblock script ให้สั่งไฟร์วอลให้ปลด Block ของรายการนั้น แล้วลบรายการนั้นออกจาก Blocked Intrusion List จากนั้นดูว่ามีการใช้ Rule Number หรือไม่ ถ้ามีก็ไปที่ส่วน Add Rule Number เพิ่มหมายเลขไปที่รายการ Reused rule number list แล้วไปทำงานที่ Detect alert เพื่อตรวจหาการบุกรุกใหม่ต่อไป



รูปที่ 3.15 Flow Chart ของส่วน Run Unblocking

- Get Rule Number



รูปที่ 3.16 Flow Chart ของส่วน Get Rule number

จากรูปที่ 3.16 กระบวนการ Get Rule Number จะตรวจดูในรายการ Reused rule number list ว่ามีหมายเลขถ้ามีก็ดึงค่าออกมาแล้วให้ rule number เท่ากับค่านั้น ถ้าไม่มีก็ดูว่าตอนนี้ unused number มากกว่า rulenum_max(ซึ่งเป็นค่าจากไฟล์ Firewall script pattern) หรือไม่ ถ้ามากกว่าก็ให้ rule number เท่ากับ 0 แต่ถ้าไม่มากกว่าก็ให้ rule number เท่ากับ unused number แล้วให้ unused number เพิ่มค่าเข้าไปอีก 1

สุดท้ายจะส่งค่า rule number ตามที่ได้กำหนดไว้ ซึ่งถ้าเป็น 0 ก็หมายความว่าหมายเลขกฎถูกใช้หมดแล้ว

3.2.3 โครงสร้างของข้อมูลที่เกี่ยวข้องกับระบบ

สำหรับระบบนี้อาศัยการอ่านข้อมูลและทำงานกับไฟล์ข้อความเป็นหลัก ดังนั้นจึงต้องเข้ารูปแบบโครงสร้างของไฟล์เหล่านั้น และโครงสร้างของสคริปต์ที่เกี่ยวข้องดังนี้

1) ไฟล์ Alert

โครงสร้างของไฟล์ alert มี 2 แบบซึ่งระบบที่ทำขึ้นสามารถอ่านได้ทั้ง 2 แบบ คือ แบบ full และแบบ fast (โดยโครงสร้างของไฟล์ได้อธิบายไว้ที่หัวข้อ 3.2 ในเรื่องของส่วนตรวจสอบการแจ้งเตือนจาก SNORT)

โดยการอ่านไฟล์ alert นั้นก่อนอื่นต้องตรวจสอบว่าเป็นรูปแบบใด จึงค่อยดึงเอาข้อมูลหลักที่ต้องการ คือ <Protocol> < IP ADDRESS ต้นทาง> < PORT ต้นทาง> < IP ADDRESS ปลายทาง> < PORT ปลายทาง>

2) ไฟล์ Configuration ของระบบ

เนื่องจากระบบต้องการให้มีการกำหนดค่าเริ่มต้นของระบบก่อนทำงานได้ จึงมีไฟล์ Configuration เพื่อให้ผู้ใช้ได้กำหนดตัวแปรต่างๆ ให้เกิดการใช้งานเหมาะสม ซึ่งรูปแบบไฟล์จะเป็นไฟล์ข้อความ ซึ่งเก็บตัวแปรกับค่า โดยมีรูปแบบตามตารางที่ 3.3 และรูปที่ 3.17

โดยรูปแบบคือ "ชื่อตัวแปร = ค่าของตัวแปร" ซึ่งตัวแปรมีดังนี้

ตารางที่ 3.3 ตารางตัวแปรในไฟล์ Configuration

ชื่อตัวแปร	ความหมาย	ค่าที่สามารถกำหนดได้
alert_file	ที่อยู่ของไฟล์ alert ของ snort	ข้อความ
alert_type	รูปแบบของไฟล์ alert	full หรือ fast
firewall_script_file	ที่อยู่ของไฟล์ firewall script pattern	ข้อความ
time_out	เวลาที่กำหนดให้ปลด Block การป้องกัน โดยหน่วยเป็นวินาที	ตัวเลขค่าบวก 0 คือ ไม่จำกัดเวลา
block_level	ระดับของการป้องกัน IP หรือ PORT	ip หรือ port
log_file	ที่อยู่ของไฟล์ log ของระบบ	ข้อความ
interval_time	เวลาที่รอระหว่างรอบการอ่านไฟล์ Alert	ตัวเลข หน่วยเป็น microsecs ต้องเป็นเลขมากกว่าหนึ่ง

```
# This is a configuration file of Firewall Control Program

# alert_file is location of snort alert file, default is /var/log/snort/alert

alert_file = /var/log/snort/alert

# alert_type is type of format in alert file, by default is 'full'
# Now support type 'full' or 'fast'

alert_type = full

# firewall_script_file is file of firewall pattern

firewall_script_file = /etc/fwc/iptables.pattern

# time_out is used for unblocking intrusion (unit is seconds)
# use '0' for no limit time out

time_out = 60

# block level is option for block
# if block ip & port use value 'port'
# if block only ip use value 'ip'

block_level = port

# log_file is location of program log file.

log_file = /var/log/fwc.log

# interval_time is wait time in each loop which read new data from alert file.
# unit is microseconds (1 second = 1,000,000 microseconds)
# default is 500000 (0.5 second)

interval_time = 500000
```

รูปที่ 3.17 ไฟล์ Configuration

3) ไฟล์ Firewall script pattern

เป็นไฟล์ข้อความที่เก็บรูปแบบของการสั่งงานไฟร์วอลในการ Block และปลด Block โดยให้ระบุรูปแบบของคำสั่ง และเนื่องจากไฟร์วอลบางชนิดต้องใช้หมายเลขของกฎกำกับอยู่ในสคริปต์ด้วยเช่น ipfw ของ freebsd แต่บางชนิดก็ไม่ต้อง เช่น iptables, ipchains ดังนั้นเพื่อให้เกิดความยืดหยุ่นในการใช้งานกับไฟร์วอลได้หลายประเภท จึงต้องมีการกำหนดว่าสคริปต์ไฟร์วอลที่จะใช้นั้นให้มีหมายเลขกฎด้วยหรือไม่

โดยรูปแบบคือ "ชื่อตัวแปร = ค่าของตัวแปร" ซึ่งมีตัวแปรดังนี้

ตารางที่ 3.4 ตารางตัวแปรในไฟล์ Firewall script pattern

ชื่อตัวแปร	ความหมาย	ประเภทของข้อมูล
block_ip_level	สคริปต์สั่ง Block ในระดับ IP คือไม่ต้องระบุ Port	ข้อความ
unblock_ip_level	สคริปต์สั่งปลด Block ในระดับ IP คือไม่ต้องระบุ Port	ข้อความ
block_port_level	สคริปต์สั่ง Block ในระดับ Port คือต้องระบุ Port ด้วย	ข้อความ
unblock_port_level	สคริปต์สั่งปลด Block ในระดับ Port คือต้องระบุ Port ด้วย	ข้อความ
require_rulenum	ต้องการใช้ rule number ในสคริปต์หรือไม่	1 คือ ใช้ 0 คือ ไม่ใช้
rulenum_min	ค่าที่น้อยที่สุดที่ใช้เป็น rule number ได้	ตัวเลข มากกว่า 1
rulenum_max	ค่าที่มากที่สุดที่ใช้เป็น rule number ได้	ตัวเลข มากกว่า 1

นอกจากนี้ในแต่ละสคริปต์ยังต้องมีตัวแปรของสคริปต์ซึ่งจะอยู่รูปแบบ “<.....>” เป็นตัวบอกว่า เมื่อโปรแกรมจะทำการรันสคริปต์ ตัวแปรของสคริปต์จะถูกแทนที่ด้วยค่าอะไร ดังตารางที่ 3.5

ตารางที่ 3.5 ตารางตัวแปรของสคริปต์

ชื่อตัวแปร	ค่าที่เข้ามาแทนที่เมื่อรันสคริปต์	ประเภทของข้อมูล
<PROTO>	ชื่อ Protocol ได้แก่ ICMP TCP UDP	ข้อความ
<SRC_IP>	IP address ของต้นทาง	หมายเลข IP address
<SRC_PORT>	Port ของต้นทาง	ตัวเลข
<DES_IP>	IP address ของปลายทาง	หมายเลข IP address
<DES_PORT>	Port ของปลายทาง	ตัวเลข
<RULE_NUM>	Rule number ที่ได้จากโปรแกรม	ตัวเลข

สำหรับไฟร์วอล IPtables ใช้สคริปต์ดังนี้

block_ip_level = /usr/sbin/iptables -A INPUT -p <PROTO> -s <SRC_IP> -d <DES_IP> -j DROP

unblock_ip_level = /usr/sbin/iptables -D INPUT -p <PROTO> -s <SRC_IP> -d <DES_IP> -j DROP

block_port_level = /usr/sbin/iptables -A INPUT -p <PROTO> -s <SRC_IP> --sport <SRC_PORT> -d <DES_IP> --dport <DES_PORT> -j DROP

unblock_port_level = /usr/sbin/iptables -D INPUT -p <PROTO> -s <SRC_IP> --sport <SRC_PORT> -d <DES_IP> --dport <DES_PORT> -j DROP

ยกตัวอย่างการทำงาน โดยถ้าเลือกระดับการป้องกันที่ระดับ port เมื่อโปรแกรมตรวจพบว่ามีการบุกรุกเข้ามาใหม่โดยมีข้อมูลดังนี้ คือ

Protocol คือ TCP

IP address ของต้นทาง คือ 203.141.8.120

Port ของต้นทาง คือ 3141

IP address ของปลายทาง คือ 161.12.80.11

Port ของปลายทาง คือ 80

สคริปต์ที่จะใช้สำหรับ IPtables ในการป้องกัน เอาจากตัวแปร block_port_level ซึ่งรูปแบบดังนี้

/usr/sbin/iptables -A INPUT -p <PROTO> -s <SRC_IP> --sport <SRC_PORT> -d <DES_IP> --dport <DES_PORT> -j DROP

และเมื่อถูกแทนที่ด้วยข้อมูลข้างต้น ก็จะเป็นสคริปต์ที่นำไปใช้ได้ทันทีดังนี้

/usr/sbin/iptables -A INPUT -p TCP -s 203.141.8.120--sport 3141 -d 161.12.80.11 --dport 80 -j DROP

และถ้าสคริปต์ไฟร์วอลที่ต้องมี <RULE_NUM> ก็จะถูกแทนที่ด้วยตัวเลข โดยโปรแกรมจะเลือกตัวเลขมาให้ ตั้งแต่ค่า rulenum_min ถึง rulenum_max

```
# This is a firewall script pattern file

#<PROTO>       is protocol
#<SRC_IP>      is source ip address
#<DES_IP>      is destination ip address
#<SRC_PORT>    is source port
#<DES_PORT>    is destination port
#<RULE_NUM>    is id number of firewall rule (required if use rule number)

block_ip_level = /usr/sbin/iptables -I INPUT -p <PROTO> -s <SRC_IP> -d
<DES_IP> -j DROP

unblock_ip_level = /usr/sbin/iptables -D INPUT -p <PROTO> -s <SRC_IP> -d
<DES_IP> -j DROP

block_port_level = /usr/sbin/iptables -I INPUT -p <PROTO> -s <SRC_IP> --
sport <SRC_PORT> -d <DES_IP> --dport <DES_PORT> -j DROP

unblock_port_level = /usr/sbin/iptables -D INPUT -p <PROTO> -s <SRC_IP>
--sport <SRC_PORT> -d <DES_IP> --dport <DES_PORT> -j DROP

# require_rulenum
# 0 it's mean to don't use rule number in script
# 1 it's mean to use rule number inscript

require_rulenum = 0

# Rule numbers are identify number of rule which are used by this program.
# rulenum_min is minimum number.
# rulenum_max is maximum number.
# range (1 - 50000 or firewall maximum rule number)

rulenum_min = 100
rulenum_max = 50000
```

รูปที่ 3.18 ตัวอย่างไฟล์ Firewall script pattern สำหรับไฟร์วอล IPtables

4) ไฟล์บันทึกการทำงาน (Log File)

เมื่อระบบมีการสั่งไฟร์วอลแต่ละครั้งจะให้มีการบันทึกลงไฟล์ ซึ่งจะให้บันทึกที่ไหนก็ระบุตัวแปรในไฟล์ Configuration โดยสิ่งที่จะบันทึกได้แก่ เวลาที่รันคำสั่ง และคำสั่งที่ใช้ โดยมีโครงสร้างใน 1 บรรทัดดังนี้

[<Run Script Time>] <Action> : <Firewall Script>

ซึ่งสามารถดูรูปแบบได้จากตารางที่ 3.6 และตัวอย่างไฟล์ log ในรูปที่ 3.19

ตารางที่ 3.6 ตารางรูปแบบข้อมูลใน Log file

ชื่อตัวแปร	ความหมาย	ประเภทของข้อมูล
Run Script Time	เวลาที่รันสคริปต์	ข้อความ (ปี-เดือน-วัน ชั่วโมง:นาที:วินาที)
Action	เป็นการBlock หรือUnblock	ข้อความ(BLOCK / UNBLOCK)
Firewall Script	สคริปต์ที่สั่งไฟร์วอล	ข้อความ
<DES_IP>	IP address ของปลายทาง	หมายเลข IP address
<DES_PORT>	Port ของปลายทาง	ตัวเลข
<RULE_NUM>	Rule number ที่ได้จากโปรแกรม	ตัวเลข

```
**START FIREWALL CONTROL WITH INTRUSION SYSTEM** @ [2004-12-21 16:47:07]
[2004-12-21 16:47:15] BLOCK   :/usr/sbin/iptables -I INPUT -p ICMP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-21 16:47:15] BLOCK   :/usr/sbin/iptables -I INPUT -p ICMP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-21 16:47:20] BLOCK   :/usr/sbin/iptables -I INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-21 16:47:27] BLOCK   :/usr/sbin/iptables -I INPUT -p TCP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-21 16:48:55] UNBLOCK :/usr/sbin/iptables -D INPUT -p ICMP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-21 16:48:55] UNBLOCK :/usr/sbin/iptables -D INPUT -p ICMP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-21 16:49:00] UNBLOCK :/usr/sbin/iptables -D INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-21 16:49:07] UNBLOCK :/usr/sbin/iptables -D INPUT -p TCP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-21 16:49:12] BLOCK   :/usr/sbin/iptables -I INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
**START FIREWALL CONTROL WITH INTRUSION SYSTEM** @ [2004-12-22 05:49:16]
[2004-12-22 05:54:05] BLOCK   :/usr/sbin/iptables -A INPUT -p TCP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-22 05:54:07] BLOCK   :/usr/sbin/iptables -A INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-22 05:54:25] BLOCK   :/usr/sbin/iptables -A INPUT -p ICMP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-22 05:54:25] BLOCK   :/usr/sbin/iptables -A INPUT -p ICMP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-22 05:55:45] UNBLOCK :/usr/sbin/iptables -D INPUT -p TCP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-22 05:55:47] UNBLOCK :/usr/sbin/iptables -D INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-22 05:56:05] UNBLOCK :/usr/sbin/iptables -D INPUT -p ICMP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-22 05:56:05] UNBLOCK :/usr/sbin/iptables -D INPUT -p ICMP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-22 06:17:54] BLOCK   :/usr/sbin/iptables -A INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:19:34] UNBLOCK :/usr/sbin/iptables -D INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:34:12] BLOCK   :/usr/sbin/iptables -A INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:35:52] UNBLOCK :/usr/sbin/iptables -D INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:55:14] BLOCK   :/usr/sbin/iptables -A INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:56:54] UNBLOCK :/usr/sbin/iptables -D INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
```

รูปที่ 3.19 ตัวอย่างไฟล์บันทึกการทำงาน (Log File)

5) ตัวแปร New intrusions list

เป็นตัวแปรแบบ List ใน C++ ซึ่งเก็บรายการข้อมูลการบุกรุก โดยในแต่ละเรคคอร์ดจะเก็บข้อมูลการบุกรุกที่ได้จากไฟล์ alert จะประกอบด้วยฟิลด์ ดังนี้

ตารางที่ 3.6 ตารางรูปแบบการเก็บข้อมูลในตัวแปร New intrusions list

ชื่อฟิลด์	ความหมาย	ประเภทของข้อมูล
src_ipaddr	หมายเลข IP Address ของต้นทาง	string
src_port	หมายเลข Port ของต้นทาง	string
des_ipaddr	หมายเลข IP Address ของปลายทาง	string
des_port	หมายเลข Port ของปลายทาง	string
protocol	โปรโตคอล	string

6) ตัวแปร Blocked intrusions list

เป็นตัวแปรแบบ List ใน C++ โดยในแต่ละเรคคอร์ดจะเก็บข้อมูลการบุกรุกที่ได้สั่งไฟร์วอลไปแล้วจะประกอบด้วยฟิลด์ ดังนี้

ตารางที่ 3.7 ตารางรูปแบบการเก็บข้อมูลในตัวแปร Blocked intrusions list

ชื่อ fields	ความหมาย	ประเภทของข้อมูล
src_ipaddr	หมายเลข IP Address ของต้นทาง	string
src_port	หมายเลข Port ของต้นทาง	string
des_ipaddr	หมายเลข IP Address ของปลายทาง	string
des_port	หมายเลข Port ของปลายทาง	string
protocol	โปรโตคอล	string
blocktime	เวลาที่สั่งสคริปต์ไฟร์วอล	time_t (C++)
blockinglevel	ระดับในการสั่งไฟร์วอล ip หรือ port	string
blockscript	สคริปต์ที่ใช้สั่งไฟร์วอล	string
strrulenum	หมายเลข Rule Number ที่ใช้ในสคริปต์นี้	string

6) ตัวแปร Reused rule number list

เป็นตัวแปรแบบ List ใน C++ ในแต่ละรายการจะเก็บ 1 ค่า คือ ตัวเลข rule number ที่เคยถูกใช้แล้วที่สามารถนำไปใช้ได้ ซึ่งเป็นประเภทข้อมูลแบบ unsigned long integer

จากการออกแบบการทำงานของระบบที่ผ่าน ทำให้สามารถเข้าใจการระบบบอย่างชัดเจนทั้งกระบวนการทำงาน ข้อมูลที่เกี่ยวข้อง ซึ่งสิ่งเหล่านี้จะนำไปใช้ในการเขียนโปรแกรมให้ระบบที่ได้ออกแบบไว้ให้สามารถทำงานได้จริง ซึ่งการพัฒนาระบบจะอยู่ในบทต่อไป

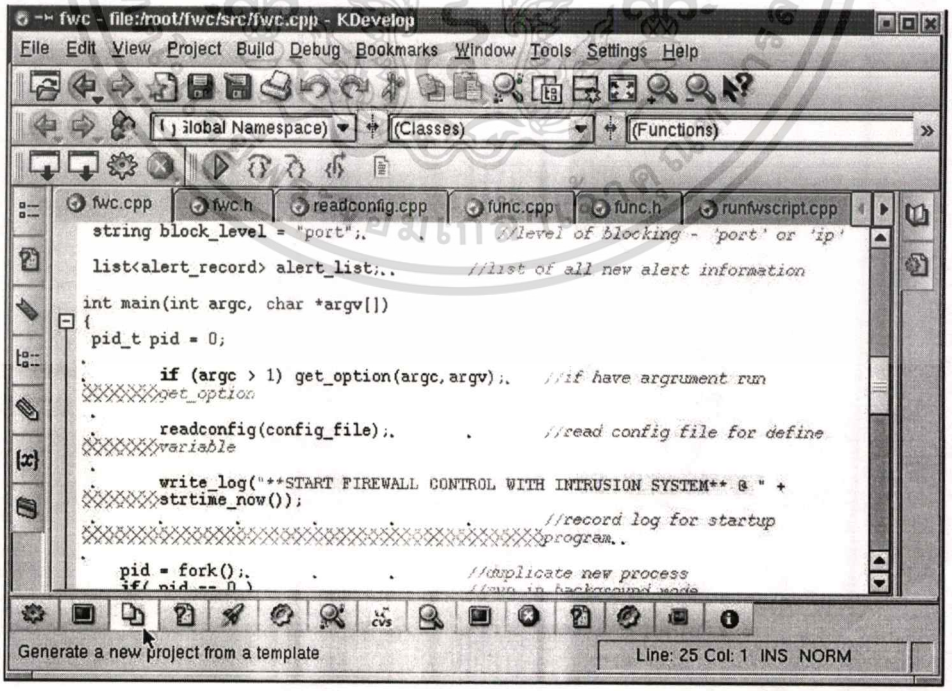
บทที่ 4
การพัฒนาระบบ

ระบบที่จะพัฒนาขึ้นนั้นมีจุดมุ่งหมายที่ทำงานบนระบบปฏิบัติการลินุกซ์ ซึ่งในครั้งนี้ได้ใช้ Slackware เวอร์ชั่น 10 เป็นระบบปฏิบัติการในการพัฒนาโปรแกรม ซึ่งรันด้วยเคอร์เนลเวอร์ชั่น 2.4.26 ส่วนภาษาที่ใช้ในการเขียนโปรแกรมนั้นคือ C++ ซึ่งระบบนี้จะเขียนโปรแกรมในลักษณะเชิงโครงสร้าง และใช้ตัวคอมไพลเลอร์ของ GNU C++ Compiler เวอร์ชั่น 3.3.4

4.1 เครื่องมือที่ใช้ในการพัฒนา

1) โปรแกรม KDevelop เวอร์ชั่น 3.0

โปรแกรม KDevelop เป็นโปรแกรมสำหรับการพัฒนาระบบซึ่งสนับสนุนหลายภาษา ได้แก่ Ada ,C/C++,Fortran,PHP,Python,Java,Pascal,Ruby เป็นต้น ซึ่งในการพัฒนาด้วยภาษา C++ นั้น โปรแกรม KDevelop จะใช้ GNU C++ Compiler ในการคอมไพล์โปรแกรม นอกจากนี้สามารถสนับสนุนเขียนโปรแกรมบน X window ได้



รูปที่ 4.1 หน้าจอของโปรแกรม KDevelop

2) โปรแกรม Snort เวอร์ชั่น 2.2.0

เป็นโปรแกรมตรวจจับการบุกรุกบนเครือข่าย ใช้ในการบันทึกการบุกรุกลงไฟล์ alert เพื่อให้โปรแกรมที่พัฒนาขึ้นอ่านไฟล์นั้น ส่วนรายละเอียดอื่นๆ ได้อธิบายไว้แล้วในบทที่ 2

3) ไฟร์วอล IPtables เวอร์ชั่น 1.2.10

เป็นไฟร์วอลบนลินุกซ์ ซึ่งในการใช้งานของระบบจะใช้ในส่วนของ packet filter เป็นหลัก เพื่อป้องกันการบุกรุก โดยทั้ง Snort ,IPtables และโปรแกรมที่พัฒนาขึ้นจะทำงานอยู่บนเครื่องเดียวกัน ส่วนรายละเอียดอื่นๆ ของ IPtables ได้อธิบายไว้แล้วในบทที่ 2

4) โปรแกรมที่ใช้สร้างการบุกรุกบนเครือข่าย

ในการพัฒนาโปรแกรมจำเป็นต้องให้มีการบุกรุกเกิดขึ้น เพื่อจะได้เขียนโปรแกรมได้อย่างถูกต้อง โดยโปรแกรมเหล่านี้จะส่ง packet ที่มีรูปแบบการบุกรุก เช่น Ping Flood , Scan port เป็นต้น โดยที่ Snort จะพบการบุกรุกแล้วบันทึกลงไฟล์ alert ซึ่งโปรแกรมที่ใช้สร้างการบุกรุกจะอยู่ที่เครื่องอื่น

4.2 ขั้นตอนในการพัฒนาระบบ

หลังจากที่ออกแบบระบบแล้ว ก็เริ่มทำการเขียนโปรแกรม แต่เนื่องจากไม่มีประสบการณ์ในการเขียนภาษา C และ C++ อย่างจริงจังมาก่อน จึงต้องทำการศึกษาการใช้ภาษา C++ ได้แก่ ไวยากรณ์ของภาษา ลักษณะการเขียน การเรียกใช้ไลบรารี่ต่างๆ บนระบบ UNIX การเข้าใจพื้นฐานของ process บนระบบ UNIX และได้ตั้งชื่อโปรแกรมที่จะเขียนขึ้นย่อๆ ว่า fwc มาจาก firewall control

ในการเขียนโปรแกรมนั้นได้เขียนฟังก์ชั่นหลักๆ ดังนี้

- ฟังก์ชั่นในการเตรียมค่าเริ่มต้นของโปรแกรม

หน้าที่หลักคืออ่านค่าจากไฟล์ Configuration โดยรับข้อมูลเป็นสตริงที่ละบรรทัด หาบรรทัดที่มีตัวแปร แล้วแยกส่วนของตัวแปรกับค่าของตัวแปรออกจากกัน ก็จะทราบว่าตัวแปรแต่ละตัวถูกกำหนดค่าอะไรไว้ แล้วจึงกำหนดเป็นค่าเริ่มต้นของระบบในการทำงานฟังก์ชั่นอื่นต่อไป

- ฟังก์ชั่นตรวจสอบไฟล์ alert

หน้าที่หลักคืออ่านไฟล์ alert โดยในตอนเริ่มแรกนั้นจะชี้ตัวอ่านไปที่ตำแหน่งสุดท้ายของไฟล์แล้ววนลูปเข้าตรวจดูเรื่อยๆ ว่าขนาดไฟล์เพิ่มขึ้นหรือไม่ ถ้าเพิ่มก็อ่านค่าที่เกิดขึ้นใหม่แล้วหาว่ามีรายการการบุกรุกอะไรบ้าง แล้วเก็บลงในโครงสร้างข้อมูลแบบ List ซึ่งโครงสร้างนี้อยู่ใน Standard Template Library ของ C++ โดยที่ List จะเก็บข้อมูลรายการการบุกรุกใหม่ที่พบในขณะนั้นไว้

- ฟังก์ชั่นสั่งไฟร์วอลให้ Block

หน้าที่หลักคือเข้าไปดึงรายการจากใน List รายการการบุกรุกที่เกิดจากฟังก์ชั่นตรวจสอบไฟล์ alert ในที่นี่รายการที่ถูกดึงมาแล้วจะไม่อยู่ใน List อีก และรายการที่ถูกดึงมานั้นจะใช้เป็นข้อมูลในการสั่งไฟร์วอลให้ Block การบุกรุก โดยการสร้างสคริปต์ไฟร์วอลขึ้นจากรูปแบบของสคริปต์บวกกับข้อมูล Protocol, IP address, Port ยกตัวอย่างเช่น

เมื่อตรวจพบรายการการบุกรุกเข้ามาใหม่ มีข้อมูลดังนี้ คือ

Protocol คือ UDP

IP address ของต้นทาง คือ 201.10.18.120

Port ของต้นทาง คือ 3021

IP address ของปลายทาง คือ 65.34.20.14

Port ของปลายทาง คือ 53

และสคริปต์ที่จะใช้สำหรับ IPtables ในการป้องกัน มีรูปแบบดังนี้

/usr/sbin/iptables -A INPUT -p <PROTO> -s <SRC_IP> --sport <SRC_PORT> -d <DES_IP> --dport <DES_PORT> -j DROP

จากนั้นถูกแทนที่ด้วยข้อมูลข้างต้น ก็จะเป็นสคริปต์ที่นำไปใช้ได้ทันที

/usr/sbin/iptables -A INPUT -p TCP -s 201.10.18.120--sport 3021 -d 65.34.20.14 --dport 53 -j DROP

ถ้ามีการใช้ Rule Number ก็แทน <RULE_NUM> ในสคริปต์ ด้วยหมายเลขที่ได้จากฟังก์ชั่นกำหนดหมายเลขกฎ ถ้าหมายเลขที่ส่งมาให้เป็น 0 แสดงว่าหมายเลขหมด การบุกรุกนี้ต้องรอจนกว่าจะมีการปลด Block คืนหมายเลข จึงจะได้เลขที่มาสร้างเป็นสคริปต์ต่อไป

เมื่อได้สคริปต์แล้วก็สั่งไฟร์วอลโดยใช้ฟังก์ชั่นที่มีอยู่แล้วคือ system(สคริปต์) เมื่อสั่งไปแล้วก็เก็บรายการนี้ไว้ใน List ของการบุกรุกที่ถูก Block เอาไว้แล้ว รวมทั้งหมายเลข Rule Number เพื่อไว้อ้างถึงในการปลด Block ต่อไป

- ฟังก์ชั่นสั่งไฟร์วอลให้ปลด Block

หน้าที่หลักคือเข้าไปตรวจดูในรายการใน List ของการบุกรุกที่ถูก Block แล้ว โดยดูว่าแต่ละรายการถูกสั่ง Block เป็นเวลานานเท่าไหร่ถ้าเกินค่า time out ที่ตั้งไว้ก็สร้างสคริปต์ไฟร์วอลขึ้นเช่นเดียวกับฟังก์ชั่นสั่งไฟร์วอลให้ Block โดยใช้ข้อมูลจาก Protocol, IP address, Port หรือ Rule Number ที่ได้จาก List ของการบุกรุกที่ถูก Block แล้ว เมื่อได้สคริปต์แล้วก็สั่งไฟร์วอลให้ปลด Block ถ้ามีการใช้ Rule Number ก็จะเอาหมายเลขกฎที่ปลด Block แล้วไปไว้ที่ List รายการหมายเลขที่ใช้ได้

โดยฟังก์ชั่นตรวจสอบไฟล์ alert ฟังก์ชั่นสั่งไฟร์วอลให้ Block และฟังก์ชั่นสั่งไฟร์วอลให้ปลด Block ทำจะงานวนลูปเรียงตามลำดับ โดยวนซ้ำอยู่เรื่อยๆ

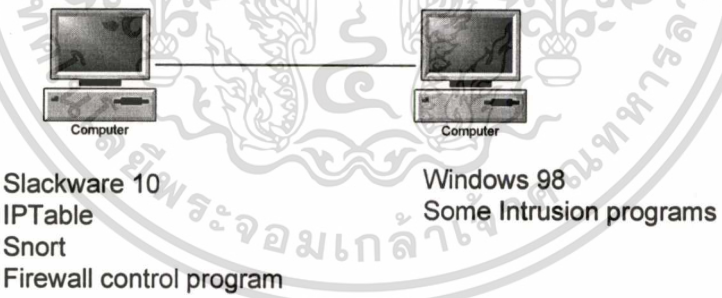
และเพื่อให้โปรแกรมนั้นทำงานอยู่เบื้องหลัง (Background) หรือเป็นแบบ daemon ก็ใช้คำสั่ง fork() แตก Child process มา 1 อัน แล้วค่อย exit หรือ ออกจาก Parent process ซึ่ง Child process ก็จะทำงานเหมือนกับ Parent process เดิม

- ฟังก์ชั้นกำหนดหมายเลขกฎ (Rule Number)

หน้าที่คือส่งค่าหมายเลขกฎที่สามารถใช้ได้ ค่าอยู่ในช่วง rulenum_min ถึง rulenum_max ที่กำหนดไว้จากไฟล์ Firewall script pattern ฟังก์ชั่นเริ่มต้นโดยการดึงค่าจากใน List รายการหมายเลขที่ใช้ได้ก่อน ถ้าไม่มีก็เอาค่าที่น้อยที่สุดที่ยังไม่ถูกใช้ ซึ่งเมื่อโปรแกรมเริ่มทำงานค่านี้จะเท่ากับ rulenum_min เมื่อถูกใช้แต่ละครั้งค่านี้ก็จะเพิ่มทีละหนึ่งจนถึงค่า rulenum_max แล้วถ้าเมื่อถึงค่า rulenum_max และไม่มีเลขในList รายการหมายเลขที่ใช้ได้ แสดงว่าเลขถูกใช้หมดแล้ว ก็จะส่งค่า 0 ไปให้

4.3 การทดสอบการทำงานระบบ

เนื่องจากการทดสอบนั้นต้องใช้เครื่อง อย่างน้อย 2 เครื่อง คือเครื่องหลักที่มีโปรแกรมอยู่กับเครื่องที่ทำการบุกรุก



รูปที่ 4.2 เครื่องคอมพิวเตอร์ในการทดสอบ

เครื่องที่ใช้ทดสอบมีรายละเอียดดังนี้

- CPU Intel Pentium M 1.3
- Ram 256 MB
- OS Linux Slackware 10 kernel 2.4.26
- Snort กฎที่ใช้เป็นกฎมาตรฐานไม่ได้ปรับแต่งใดๆ
- ไฟร์วอล IPTables

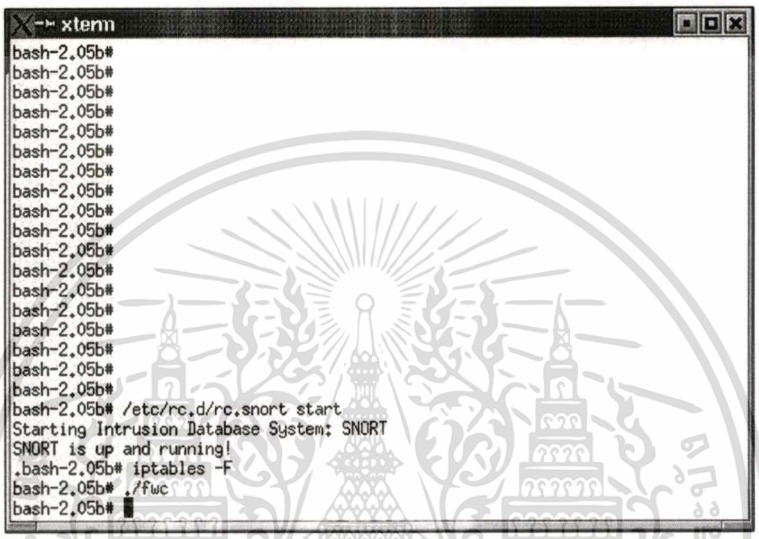
4.3.1 ขั้นตอนการทดสอบ

ทดสอบการใช้งานปกติดังนี้

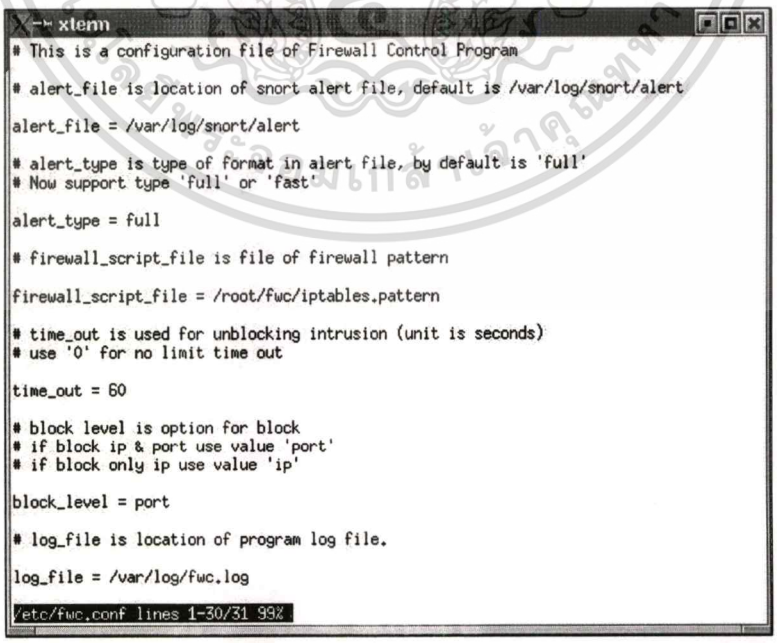
1) เปิดโปรแกรม snort โดยเลือกรูปแบบ alert เป็นแบบ full และ fast โดยดูทีละกรณี

2) ลบ rule ของ IPTables ให้หมด โดยสั่ง iptables –F

3) เปิดโปรแกรม fwc โดยเลือกระดับการป้องกันเป็น ip และ port โดยดูทีละกรณี



รูปที่ 4.3 ตัวอย่างการเริ่มทดสอบระบบ

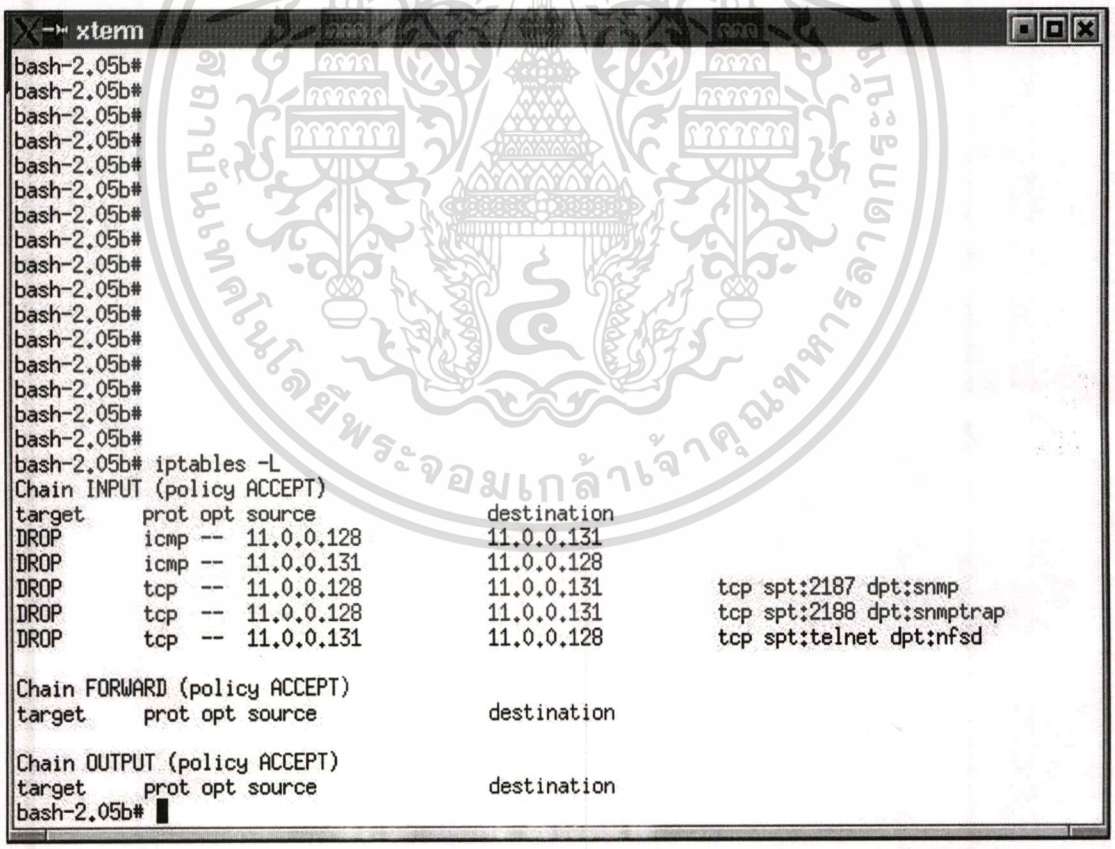


รูปที่ 4.4 ตัวอย่างไฟล์ Configuration ที่ใช้ทดสอบระบบ

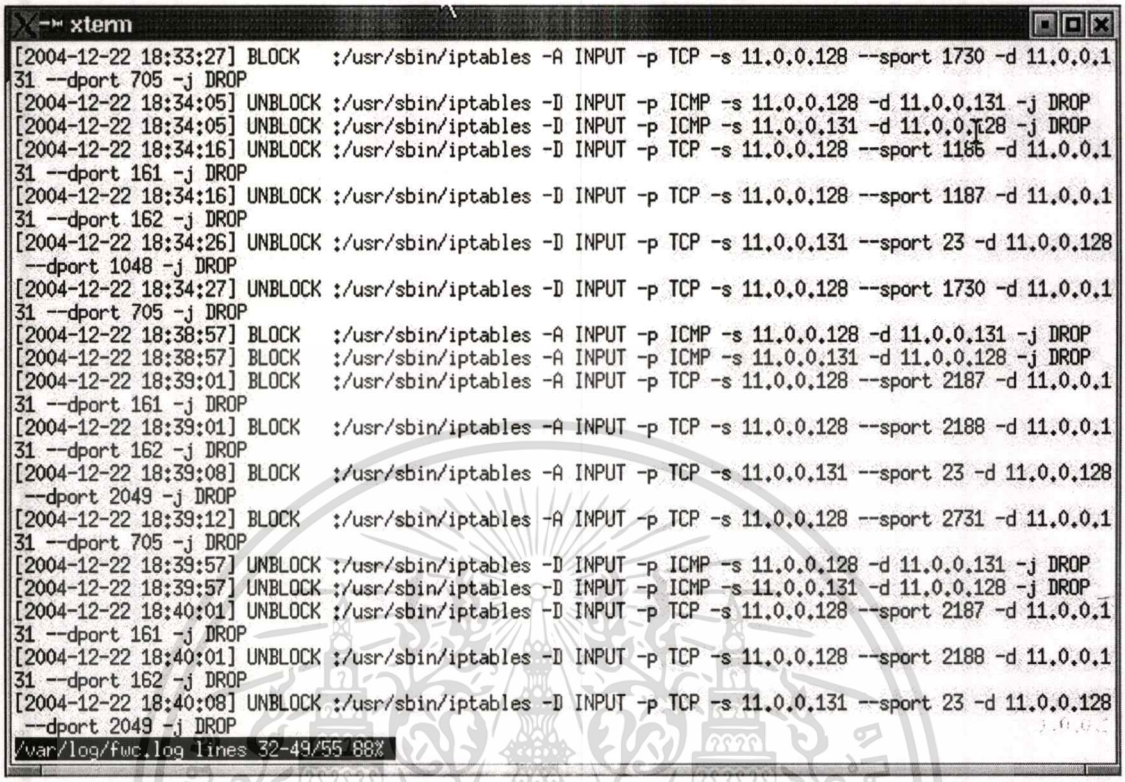
4.3.2 ผลการทดสอบ

โปรแกรมสามารถใช้งานได้ตามที่ต้องการคือ

- สามารถสนับสนุนไฟล์ alert ทั้งแบบ full และ fast
- สามารถทำงานได้ทั้งระดับการป้องกันเป็น ip และ port
- สามารถสั่งไฟร์วอลให้ป้องกันได้ ดูรูปที่ 4.5
- ในการปลด block สามารถทำได้ตามเวลาที่กำหนดไว้เช่นกัน ดูรูปที่ 4.6 แต่ในกรณีที่เครื่องทำงานหนักเวลาก็จะอาจจะคลาดเคลื่อนไปเล็กน้อย เนื่องจากเสียเวลาในการประมวลผลในแต่ละรอบการทำงาน
- ในกรณีที่มีการใช้ Rule Number ก็สามารถใช้งานได้ คือสามารถเพิ่มหมายเลขกฎในสคริปต์ในการสั่ง Block และปลด Block ได้อย่างถูกต้อง ตามช่วงหมายเลขที่กำหนดไว้



รูปที่ 4.5 กฎของไฟล์วอลที่เพิ่มเข้ามาในขณะทดสอบระบบ

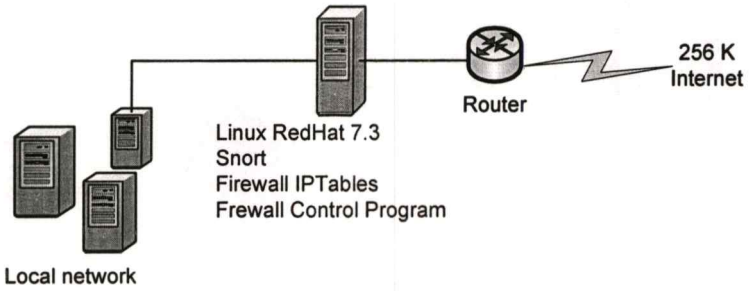


รูปที่ 4.6 ล็อกไฟล์ของโปรแกรมที่บันทึกการ block และปลด block

สรุปผลการทดสอบเป็นไปตามความต้องการของระบบเบื้องต้นที่ตั้งไว้ ทั้งนี้การทดสอบที่ทำนั้นมีการสร้างการบุกรุกเพียงเล็กน้อย และยังมีรูปแบบไม่กี่ประเภท แต่ในการใช้งานจริงย่อมมีข้อมูลเข้าออกในเครือข่ายมากกว่านี้ และมีรูปแบบการบุกรุกแบบอื่นๆ อีกมากนอกเหนือจากที่ได้ทดสอบ ซึ่งจะได้ดูในการทดลองใช้งานจริง

4.4 การทดลองใช้งานจริง

ในการทดลองใช้งานจริงนั้น เพื่อดูการทำงานของโปรแกรมว่าสามารถทำงานในสภาวะที่มีการใช้เครือข่ายเป็นจำนวนมากได้หรือไม่ และดูว่าโปรแกรมสามารถรองรับการบุกรุกที่เป็นแบบอื่นๆ นอกเหนือจากการใช้โปรแกรมสร้างการบุกรุก ซึ่งในการทดลองนั้นได้ลงโปรแกรมไว้ที่เครื่องที่มีไฟร์วอล IPTables และ Snort โดยการเชื่อมต่อเป็นดังนี้



รูปที่ 4.7 การทดลองใช้งานจริง

ซึ่งผลที่ได้จากการทดลองใช้งานจริง สามารถตอบสนองการทำงานได้คืออ่านไฟล์ alert จาก Snort และสามารถสั่งไฟร์วอล IPtables ได้ทันที ทั้งการป้องกันระดับ ip และ port ซึ่งโดยรวมสามารถทำงานได้ตามต้องการ แต่ระหว่างการทดลองใช้งานจริงนั้นได้พบปัญหาคือ ไฟล์ alert ที่เป็นแบบ full นั้นจะมีข้อมูลการบุกรุกบางประเภทที่มีรูปแบบการบันทึกที่ไม่ได้เป็นไปตามที่ได้คิดออกแบบไว้ในตอนแรกเล็กน้อย จึงทำให้เกิดการผิดพลาดในการอ่าน ซึ่งหลังจากที่พบปัญหานี้ก็ได้นำไปปรับปรุงแก้ไขโปรแกรมให้มีความยืดหยุ่นในการรองรับรูปแบบดังกล่าวด้วย

บทที่ 5
บทสรุป

5.1 ผลจากการพัฒนาระบบ

ผลจากการพัฒนาระบบควบคุมไฟร์วอลผ่านระบบตรวจจับการบุกรุกนั้น สามารถที่จะทำให้ระบบใช้งานได้ตามความต้องการที่ตั้งไว้ ซึ่งระบบจะประกอบด้วย 2 ส่วนคือส่วนที่หาการบุกรุกใหม่ ก็ทำได้โดยเฝ้าดูการเปลี่ยนแปลงของไฟล์ alert และสามารถนำข้อมูลมาแยกแยะส่วนที่ต้องการได้ นอกจากนี้ยังสามารถตรวจสอบรูปแบบของไฟล์ alert ได้โดยอัตโนมัติ ว่าเป็นแบบ full หรือ fast และอีกส่วนคือการสั่งไฟร์วอล ซึ่งตอนนี้สามารถใช้ภายในเครื่องเดียวกันได้เท่านั้น และการยกเลิกการป้องกันโดยอัตโนมัติก็สามารถทำงานได้ตามเวลาที่กำหนดไว้

5.1 ประโยชน์ที่ได้รับ

ระบบที่พัฒนาขึ้นสามารถที่จะนำไปใช้ในการป้องกันการบุกรุกได้อย่างอัตโนมัติ โดยเมื่อมีการบุกรุกเข้ามาที่เครือข่ายก็สามารถสั่งให้ไฟร์วอลป้องกันทันที ซึ่งรวดเร็วกว่าการใช้คนมาสั่งไฟร์วอลทีหลัง และลดความเสียหายที่เกิดขึ้นจากการโจมตีได้ ทั้งนี้ผู้ใช้สามารถระดับในการป้องกันได้ให้เหมาะกับเครือข่ายที่ใช้งานอยู่ แต่ข้อควรระวังในกรณีที่ระบบตรวจจับการบุกรุกแจ้งเตือนผิดพลาดคือเข้าใจผิดว่าการใช้งานปกติเป็นการบุกรุก ซึ่งระบบนี้ก็จะป้องกันทันทีซึ่งจะกลายเป็นการขัดขวางการใช้งานนั้นไป ดังนั้นในการใช้งานจริงควรจะปรับแต่งตัวระบบตรวจจับการบุกรุกให้เหมาะสมกับการใช้งานของเครือข่าย เพื่อจะเกิดการแจ้งเตือนผิดพลาดให้น้อยที่สุด

5.3 แนวทางในการพัฒนาต่อ

เนื่องจากตอนนี้ระบบใช้ได้โดยอยู่เครื่องเดียวกับไฟร์วอลเท่านั้น ซึ่งในความเป็นจริงไฟร์วอลมักจะอยู่แยกต่างหาก ดังนั้นสิ่งที่จะปรับปรุงคือให้สามารถสั่งไฟร์วอลที่เครื่องอื่นได้ และควรจะสนับสนุนในการสั่งไฟร์วอลอื่นๆ นอกจาก IPtables เช่น ไฟร์วอลบน FreeBSD หรือระบบอื่น แม้ตอนนี้ผู้ใช้สามารถแก้ไขรูปแบบสคริปต์ไฟร์วอลได้ แต่ตัวโปรแกรมอาจจะไม่สามารถ

วิธีการใช้งาน

พิมพ์คำสั่งดังนี้

/usr/local/fwc หรือ pathที่ได้ติดตั้งไว้

ซึ่งโดยปกติ fwc.conf อยู่ที่ /etc/fwc/

หรือระบุ fwc.conf ที่อื่น ก็ใช้คำสั่ง ระบุพารามิเตอร์ –c

/usr/local/fwc –c <location of fwc.conf >

นอกจากนี้ผู้ใช้สามารถเปลี่ยนรูปแบบสคริปต์ไฟร์วอลได้ที่ไฟล์ /etc/fwc/iptables.pattern ซึ่งมีรูปแบบดังนี้

```
# This is a firewall script pattern file

#<PROTO>       is protocol
#<SRC_IP>      is source ip address
#<DES_IP>      is destination ip address
#<SRC_PORT>    is source port
#<DES_PORT>    is destination port
#<RULE_NUM>    is id number of firewall rule (required if use rule number)

block_ip_level = /usr/sbin/iptables -I INPUT -p <PROTO> -s <SRC_IP> -d
<DES_IP> -j DROP

unblock_ip_level = /usr/sbin/iptables -D INPUT -p <PROTO> -s <SRC_IP> -d
<DES_IP> -j DROP

block_port_level = /usr/sbin/iptables -I INPUT -p <PROTO> -s <SRC_IP> --
sport <SRC_PORT> -d <DES_IP> --dport <DES_PORT> -j DROP

unblock_port_level = /usr/sbin/iptables -D INPUT -p <PROTO> -s <SRC_IP>
--sport <SRC_PORT> -d <DES_IP> --dport <DES_PORT> -j DROP

# require_rulenum
# 0 it's mean to don't use rule number in script
# 1 it's mean to use rule number inscript

require_rulenum = 0

# Rule numbers are identify number of rule which are used by this program.
# rulenum_min is minimum number.
# rulenum_max is maximum number.
# range (1 - 50000 or firewall maximum rule number)

rulenum_min = 100
rulenum_max = 500
```

ไฟล์ iptables.pattern

คอมไพล์ได้ในระบบอื่น ซึ่งอาจต้องปรับปรุงในส่วนนี้ และควรเพิ่มความสามารถในการยกเว้น IP address หรือ Sub net ที่จะไม่ต้อง Block ได้ ช่วยเพิ่มความยืดหยุ่นช่วยให้ใช้งานได้มีประสิทธิภาพมากขึ้น

บรรณานุกรม

ภูวดล ด่านระหาญ. 2544. **Linux 2.4 Stateful Firewall : IPTABLES.** [Online]. Available : http://thaicert.nectec.or.th/paper/firewall/ iptables.php

D. Brent Chapman and Elizabeth D. Zwicky. 1996. **Firewall Design.** [Online]. Available :http://sunsite.nstu.nsk.su/sunworldonline/swol-01-1996/swol-01-firewall.html

Martin Roesch and Chris Green. 2003. **Snort Users ManualSnort Release: 2.0.1.** [Online]. Available: http://www.snort.org/docs /SnortUsersManual-2.0.1.pdf

Rebecca Bace and Peter Mell. n.p. **Intrusion Detection Systems.** [Online]. Available : http://www.snort.org/docs/nist-ids.pdf

Rusty Russell. 2001. **Linux 2.4 Packet Filtering HOWTO.** [Online]. Available : http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/index.html

ภาคผนวก

ภาคผนวก ก
การติดตั้งระบบ

ความต้องการของระบบ

สำหรับเครื่องที่สามารถติดตั้งได้นั้นมีรายละเอียดดังนี้

- ระบบปฏิบัติการลินุกซ์ หรือ FreeBSD
- มีไฟร์วอล IPtables IPChains หรือ IPfw
- มีโปรแกรม Snort

การติดตั้ง

สำหรับการติดตั้งนั้น ต้องติดตั้งบนเครื่องเดียวกับ Snort และ firewall ซึ่งไฟล์ที่ใช้ติดตั้งจะอยู่ในรูปแบบบีบอัดไว้แล้ว ไฟล์คือ fwc.tar.gz

- ขยายไฟล์ของโปรแกรม

tar –zxvf fwc.tar.gz

cd fwc

- คอมไพล์โปรแกรม และติดตั้งโดยทำคำสั่ง

./configure –prefix = < path ที่จะติดตั้งตัว fwc ค่า default เป็น /usr/local/bin>

make

make install

เท่านี้ตัวไบนารีไฟล์ fwc จะอยู่ที่ path ที่ตั้งไว้

- คัดลอกไฟล์ fwc.conf และ iptables.pattern ซึ่งอยู่ใน etc ไปไว้ที่ /etc หรือที่ไหนก็ได้

cd etc

cp –r fwc /etc/

ก็จะได้ไฟล์ fwc.conf และ iptables.pattern อยู่ใน /etc/fwc/

หรือจะไว้ที่อื่นก็ได้ แต่เวลาสั่งงานต้องระบุตำแหน่งของไฟล์ fwc.conf ด้วย ส่วน ตำแหน่งของไฟล์ iptables.pattern จะอยู่ในไฟล์ fwc.conf อยู่แล้วสามารถแก้ไขได้

ภาคผนวก ข

การใช้งานระบบ

การใช้งาน

เงื่อนไขในการใช้งานได้ต้องมีไฟล์ fwc.conf และ iptables.pattern และควรตั้งค่าตัวแปรในไฟล์ ดังกล่าวให้ตรงกับการใช้งาน ซึ่งมีรูปแบบดังนี้

```
# This is a configuration file of Firewall Control Program

# alert_file is location of snort alert file, default is /var/log/snort/alert

alert_file = /var/log/snort/alert

# alert_type is type of format in alert file, by default is 'full'
# Now support type 'full' or 'fast'

alert_type = full

# firewall_script_file is file of firewall pattern

firewall_script_file = /etc/fwc/iptables.pattern

# time_out is used for unblocking intrusion (unit is seconds)
# use '0' for no limit time out

time_out = 60

# block level is option for block
# if block ip & port use value 'port'
# if block only ip use value 'ip'

block_level = port

# log_file is location of program log file.

log_file = /var/log/fwc.log

# interval_time is wait time in each loop which read new data from alert file.
# unit is microseconds (1 second = 1,000,000 microseconds)
# default is 500000 (0.5 second)

interval_time = 500000
```

ไฟล์ fwc.conf

ผู้ใช้สามารถตรวจสอบ log file ของโปรแกรม เพื่อดูการสั่งไฟร์วอลทั้งที่เป็นการ Block และการปลด Block ซึ่งปกติอยู่ที่ /var/log/fwc.log

```
**START FIREWALL CONTROL WITH INTRUSION SYSTEM** @ [2004-12-21 16:47:07]
[2004-12-21 16:47:15] BLOCK   :/usr/sbin/iptables -I INPUT -p ICMP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-21 16:47:15] BLOCK   :/usr/sbin/iptables -I INPUT -p ICMP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-21 16:47:20] BLOCK   :/usr/sbin/iptables -I INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-21 16:47:27] BLOCK   :/usr/sbin/iptables -I INPUT -p TCP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-21 16:48:55] UNBLOCK :/usr/sbin/iptables -D INPUT -p ICMP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-21 16:48:55] UNBLOCK :/usr/sbin/iptables -D INPUT -p ICMP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-21 16:49:00] UNBLOCK :/usr/sbin/iptables -D INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-21 16:49:07] UNBLOCK :/usr/sbin/iptables -D INPUT -p TCP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-21 16:49:12] BLOCK   :/usr/sbin/iptables -I INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
**START FIREWALL CONTROL WITH INTRUSION SYSTEM** @ [2004-12-22 05:49:16]
[2004-12-22 05:54:05] BLOCK   :/usr/sbin/iptables -A INPUT -p TCP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-22 05:54:07] BLOCK   :/usr/sbin/iptables -A INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-22 05:54:25] BLOCK   :/usr/sbin/iptables -A INPUT -p ICMP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-22 05:54:25] BLOCK   :/usr/sbin/iptables -A INPUT -p ICMP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-22 05:55:45] UNBLOCK :/usr/sbin/iptables -D INPUT -p TCP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-22 05:55:47] UNBLOCK :/usr/sbin/iptables -D INPUT -p TCP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-22 05:56:05] UNBLOCK :/usr/sbin/iptables -D INPUT -p ICMP -s 11.0.0.128 -d 11.0.0.131 -j DROP
[2004-12-22 05:56:05] UNBLOCK :/usr/sbin/iptables -D INPUT -p ICMP -s 11.0.0.131 -d 11.0.0.128 -j DROP
[2004-12-22 06:17:54] BLOCK   :/usr/sbin/iptables -A INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:19:34] UNBLOCK :/usr/sbin/iptables -D INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:34:12] BLOCK   :/usr/sbin/iptables -A INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:35:52] UNBLOCK :/usr/sbin/iptables -D INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:55:14] BLOCK   :/usr/sbin/iptables -A INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
[2004-12-22 06:56:54] UNBLOCK :/usr/sbin/iptables -D INPUT -p UDP -s 11.0.0.1 -d 239.255.255.250 -j DROP
```

ไฟล์ fwc.log

ประวัติผู้เขียน

ชื่อ–สกุล	นายวีระวัฒน์ พัฒโน
วัน-เดือน-ปี เกิด	12 เมษายน 2520
ที่อยู่	2 ซ.8/1 เพชรเกษม ต.หาดใหญ่ อ.หาดใหญ่ จ.สงขลา 90110
อีเมล์	pweerawat@yahoo.com
ประวัติการศึกษา	- ชั้นประถมศึกษา โรงเรียนแสงทองวิทยา - ชั้นมัธยมศึกษา โรงเรียนหาดใหญ่วิทยาลัย - ปริญญาตรี วิศวกรรมศาสตร์บัณฑิต ภาควิชาคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ที่ทำงานปัจจุบัน	บริษัท ซัคเซส อินฟอร์เมชั่น ซิสเต็ม จำกัด