การพัฒนาซอฟต์แวร์เครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์

Software Development of Firewall rules Analysis and Reduction Tool

โดย

นายเกริกศักดิ์ ลิขิตสุภิณ

รหัส 44067027

อาจารย์ที่ปรึกษา

อาจารย์ อัครินทร์ คุณกิตติ

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2545

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	การพัฒนาซอฟต์แวร์เครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์
นักศึกษา	นายเกริกศักดิ์ ลิขิตสุภิณ
อาจารย์ที่ปรึกษา	อาจารย์อัครินทร์ คุณกิตติ
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2545

บทคัดย่อ

ไฟร์วอลล์เป็นอุปกรณ์ที่ใช้รักษาความปลอดภัยให้กับระบบเครือข่ายภายในองค์กร โดยไฟร์วอลล์จะทำหน้าที่ในการควบคุมการเข้า-ออกของข้อมูลระหว่างระบบเครือข่ายภายในองค์กรกับระบบเครือข่ายภายนอกองค์กร หรืออินเทอร์เน็ต โปรแกรมไฟร์วอลล์นั้นมีอยู่มากมาย แต่ในโครงการนี้จะมุ่งความสนใจไปที่ IP Firewall และ IP Filter ซึ่งมีการทำงานแบบการกรองแพ็คเก็ต คือ เมื่อมีแพ็คเก็ตผ่านไฟร์วอลล์ ไฟร์วอลล์ก็จะตรวจข้อมูลในส่วนหัวของแพ็คเก็ต กับกฎที่มีอยู่ในไฟร์วอลล์ โดยกฎนั้นจะถูกกำหนดโดยผู้ดูแลระบบเครือข่าย ซึ่งถ้ากฎของไฟร์วอลล์มีความซับซ้อนมากจะทำให้ผู้ดูแลระบบดูแลและตรวจสอบยาก จึงได้มีแนวความคิดที่จะลดจำนวนของกฎต่างๆ โดยเงื่อนไขในการลดจำนวนกฎจะพิจารณาจาก 1) กฎที่สามารถรวมได้โดยใช้หลักการ subnet 2) กฎที่ไม่อาจไปถึงได้ 3) กฎที่มีการขัดแย้งกัน โดยการตรวจสอบนั้นจะตรวจสอบจากหมายเลขไอพีต้นทาง-ปลายทาง และพอร์ต โดยโปรแกรมจะมีการทำงานอยู่ 3 ส่วนด้วยกัน คือ 1) ส่วนของการแปลงกฎของ IP firewall หรือ IP filter ให้เป็นรูปแบบกลาง 2) ส่วนของการวิเคราะห์และลดรูปกฎ 3) ส่วนของการแปลงรูปแบบกลางให้เป็นของ IP firewall หรือ IP filter ในส่วนของการวิเคราะห์ และลดรูปกฎจะตรวจสอบว่ากฎสามารถรวมให้อยู่ในรูปแบบการรวม subnet ได้หรือไม่ ถ้าได้ก็จะรวมกฎ และตรวจสอบว่ามีกฎ IP address ที่เป็น subset กันหรือไม่ ถ้าเป็นมีก็จะตรวจสอบ action ว่ามีความขัดแย้งกันหรือไม่ ถ้าใช่จะถือว่าเป็นกฎที่ขัดแย้งกัน ถ้าไม่จะเป็นกฎที่ไปไม่ถึง และลบกฎนั้นทิ้ง ผลลัพธ์ที่ได้คือ สามารถลดรูปกฎตามเงื่อนไขที่ต้องการได้ จึงทำให้ประสิทธิภาพการทำงานของไฟร์วอลล์เพิ่มขึ้น และผู้ดูแลระบบสามารถดูแลตรวจสอบกฎของไฟร์วอลล์ได้ง่าย

Title	Software Development of Firewall rules Analysis and Reduction Tool
Student	Mr. Kreksak Likitsupin
Advisor	Mr. Akharin Khunkitti
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2002

Abstract

Firewall is a networked device for security in the organization. Firewall is used to control the packet of data passed across network between the organizations and internet. This project proposes IP firewall and IP filter which are packet filtering. Packet filtering is comparing data in the header of packet with rules in firewall when packets are arrived. Firewall rules are defined and managed by network administrator. The management is complicated when the firewall rules are very complex. The key idea is to consider these condition for reducing the firewall rules 1) subnetted rules 2) unreachable rules 3) conflict rules. The analysis and reduction check source IP address, destination IP address and port. This program consists of three functions which are 1) Function of converting IP firewall or IP filter rules to program rules 2) Function of analysis and reduction rules 3) Function of converting program rules to IP firewall or IP filter rules. Analysis and reduction rules function is checking the action of both rules. If the actions are conflict, it is conflict rule. Otherwise the rule is unreachable rule which be removed. The experimental results demonstrate that the proposed program yields easy to manage firewall rules and improves the performance of firewall.

กิจกรรมประกาศ

ผู้จัดทำโครงงานขอขอบพระคุณ อาจารย์อัครินทร์ คุณกิตติ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน ที่ช่วยให้คำแนะนำในการจัดทำโครงงาน และให้คำปรึกษาด้านวิชาการที่เป็นประโยชน์ในการทำโครงงาน รวมทั้งให้ความช่วยเหลือในการแก้ไขเอกสาร เรียบเรียงเอกสาร ซึ่งช่วยให้โครงงานนี้สำเร็จลงได้

ขอขอบคุณ คณาจารย์ทุกๆ ท่านในคณะเทคโนโลยีสารสนเทศที่ได้ให้ความรู้ซึ่งเป็นประโยชน์อย่างยิ่งแก่กระผม

ขอบคุณเพื่อนๆ ทุกคนให้ความช่วยเหลือ

และสุดท้ายขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ที่ให้กำเนิด ให้การศึกษา และเป็นกำลังใจตลอดมา

เกริกศักดิ์ ลิขิตสุภิณ

กุมภาพันธ์ 2546

สารบัญ

หน้า

บทคัดย่อภาษาไทย..... I

บทคัดย่อภาษาอังกฤษ..... II

กิตติกรรมประกาศ..... III

สารบัญ..... IV

สารบัญตาราง..... VI

สารบัญภาพ..... VII

บทที่

1. บทนำ 1
   - 1.1 ความสำคัญและที่มา 1
   - 1.2 เป้าหมายของการพัฒนาระบบงาน 1
   - 1.3 วัตถุประสงค์ของการพัฒนาระบบงาน 2
   - 1.4 ขอบเขตของการพัฒนาโปรแกรม 2
   - 1.5 ประโยชน์ที่คาดว่าจะได้รับ 2
   - 1.6 ขั้นตอนการพัฒนาระบบงาน 3
   - 1.7 รายละเอียดของแต่ละบท 3
2. IP Firewall และ IP Filter..... 4
   - 2.1 ไฟร์วอลล์ และหลักการทำงานของไฟร์วอลล์..... 4
   - 2.2 หลักการทำงานของ IP Firewall และ IP Filter..... 7
   - 2.3 รูปแบบ โครงสร้างกฎของ IP Firewall และ IP Filter..... 8
   - 2.4 ทฤษฎีที่เกี่ยวข้องกับการทำ subnet..... 36
3. การออกแบบระบบงาน..... 38
   - 3.1 ลักษณะการทำงานของโปรแกรม..... 38
   - 3.2 โครงสร้างกลางของกฎภายในโปรแกรม และการ map จากกฎของ IP Firewall และ IP Filter..... 41

3.3 อัลกอริทึมที่ใช้ในการวิเคราห์ และลดรูปกฎของไฟร์วอลล์ 55
4. การพัฒนาระบบงาน และผลการทดลอง 67
4.1 ภาษา และเครื่องมือที่ใช้ในการพัฒนาระบบงาน 67
4.2 ผลการทดลอง 67
5. สรุปผลการพัฒนาโปรแกรม 76
5.1 สรุปผลการพัฒนาโปรแกรม 76
5.2 ข้อเสนอแนะ 76
บรรณานุกรม 78
ภาคผนวก ก. การติดตั้งและใช้งานโปรแกรมเครื่องมือสำหรับวิเคราะห์ และลดจำนวนกฎของไฟร์วอลล์ 80

สารบัญตาราง

ตารางที่ หน้า

2.1 การ map กฎของ IP Firewall ให้เข้ากับโครงสร้าง 21

2.2 operands ที่สามารถใช้กับหมายเลขพอร์ต 26

2.3 การกำหนดพอร์ตเป็นช่วง 26

2.4 การ map กฎของ IP Filter ให้เข้ากับโครงสร้าง 33

3.1 กลุ่มของ Action 41

3.2 การ map กฎของ IP Firewall และ IP Filter ให้เข้ากับโครงสร้างกลาง 52

สารบัญภาพ

ภาพที่		หน้า
2.1	การติดตั้งไฟร์วอลล์ระหว่างระบบเครือข่ายภายในองค์กรกับอินเทอร์เน็ต	5
2.2	ผังขั้นตอนในการกรองแพ็คเก็ต	6
2.3	ผังการทำงานของ IP Firewall	7
2.4	ผังการทำงานของ IP Filter	10
2.5	ส่วนประกอบหมายเลขไอพี	36
2.6	Netmask	36
2.7	การทำ subnet	37
3.1	Context Diagram	39
3.2	Dataflow Diagram Level 1	40
3.3	Dataflow Diagram Level 2 Process 2	40
ก.1	ตัวอย่างการทำงานของโปรแกรม	81

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันอินเทอร์เน็ตได้รับความนิยมกันอย่างแพร่หลาย เพราะอินเทอร์เน็ตนั้นมีการบริการอยู่มากมาย แต่เนื่องจากอินเทอร์เน็ตนั้นมีการเชื่อมต่อจากระบบเครือข่ายจำนวนมาก จึงทำให้มีความปลอดภัยต่ำ เพื่อที่จะรักษาความปลอดภัยไว้จึงต้องมีการนำระบบรักษาความปลอดภัยมาใช้ ไฟล์วอลล์จัดเป็นระบบรักษาความที่สำคัญต่อองค์กร โดยทำหน้าที่ในการควบคุมการเข้า-ออกของข้อมูลระหว่างระบบเครือข่ายภายในองค์กร กับระบบเครือข่ายภายนอกองค์กรหรืออินเทอร์เน็ต

ในปัจจุบันมีไฟร์วอลล์จำนวนมากมายหลายชนิด โดยงานวิจัยนี้จะทำการศึกษาเกี่ยวกับ IP Firewall และ IP Filter ซึ่งจัดเป็นไฟร์วอลล์ที่น่าสนใจ ซึ่ง IP Firewall และ IP Filter จะมีการทำงานแบบการกรองแพ็คเก็ต คือ เมื่อมีแพ็คเก็ตผ่านไฟร์วอลล์ ไฟร์วอลล์ก็จะตรวจข้อมูลในส่วนหัวของแพ็คเก็ต กับกฎที่มีอยู่ในไฟร์วอลล์ ถ้ากฎอนุญาตให้แพ็คเก็ตนั้นผ่านไปได้แพ็คเก็ตก็สามารถผ่านไฟร์วอลล์ได้ แต่ถ้าไม่อนุญาตแพ็คเก็ตนั้นก็ไม่สามารถผ่านไฟร์วอลล์ได้ โดยกฎนั้นจะถูกกำหนดโดยผู้ดูแลระบบเครือข่าย ซึ่งจะสร้างตามนโยบายความปลอดภัยขององค์กร ถ้านโยบายความปลอดภัยนั้นมีความซับซ้อนมากจะทำให้มีจำนวนกฎจำนวนมาก ซึ่งอาจจะทำให้ประสิทธิภาพการทำงานของไฟร์วอลล์ลดลง และทำให้ผู้ดูแลระบบดูแลและตรวจสอบกฎของไฟร์วอลล์ได้ยาก จึงนำมาซึ่งแนวคิดที่จะลดจำนวนของกฎต่างๆ โดยการลดจำนวนกฎนั้นจะพิจารณาจากเงื่อนไข ดังนี้ 1) กฎที่เป็น subnet กัน 2) กฎที่ไม่สามารถไปถึง 3) กฎที่มีความขัดแย้งกัน ซึ่งในการพิจารณานั้นจะพิจารณาจากหมายเลขไอพีต้นทาง-ปลายทาง และพอร์ต

1.2 วัตถุประสงค์ของการพัฒนาระบบงาน

- เพื่อศึกษาการทำงานของไฟร์วอลล์ รูปแบบกฎของ IP Firewall และ IP Filter และทฤษฎีที่เกี่ยวข้อง
- เพื่อนำหลักการดังกล่าว มาใช้ในการพัฒนาซอฟต์แวร์เครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์ เพื่อเพิ่มประสิทธิภาพการทำงานของไฟร์วอลล์ และทำให้ผู้ดูแลระบบสามารถดูแลและตรวจสอบกฎของไฟร์วอลล์ได้ง่ายยิ่งขึ้น

1.3 เป้าหมายของการพัฒนาระบบงาน

โปรแกรมเครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์ จะรับอินพุตจากแฟ้มข้อมูล ซึ่งแฟ้มข้อมูลนั้นจะเก็บกฎของ IP Firewall หรือ IP Filter เอาไว้ แล้วนำกฎต่างๆที่อยู่ในแฟ้มข้อมูลไปวิเคราะห์ และลดรูป เมื่อเสร็จแล้วจะได้ผลลัพธ์เป็นรายการของกฎที่ได้มีการลดรูปและหมายเหตุของการลดรูป นอกจากนี้ยังสามารถให้ผลลัพธ์ออกมาเป็นแฟ้มข้อมูลที่เก็บกฎของไฟร์วอลล์ที่ได้มีการลดรูปไว้แล้วได้อีกด้วย

ซึ่งเงื่อนไขในการวิเคราะห์ และลดรูปจะมี ดังนี้

- ลดรูปกฎโดยใช้หลักของการรวม subnet
- ลดรูปกฎโดยจะตรวจสอบว่าเป็นกฎที่ไม่สามารถไปถึง หรือกฎที่ขัดแย้งกัน หรือไม่ถ้าเป็นจะลบกฎนั้นทิ้ง
- การลดรูปจะพิจารณาจากหมายเลขไอพีต้นทาง-ปลายทาง และพอร์ต

1.4 ขอบเขตของการพัฒนาโปรแกรม

โปรแกรมเครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์ จะมีขอบเขตต่างๆ ดังนี้

- โปรแกรมสามารถลดรูปกฎได้ทั้ง IP Firewall และ IP Filter
- โปรแกรมได้พัฒนาเพื่อใช้งานกับ IP Firewall และ IP Filter version ที่อยู่บนระบบปฏิบัติการ FreeBSD 4.7
- โปรแกรมสามารถรองรับโปรโตคอล IP version 4 เท่านั้น ไม่รองรับโปรโตคอล IP version 6
- โปรแกรมสามารถลดรูปกฎได้จากหมายเลข IP ต้นทาง-ปลายทาง และพอร์ต
- โปรแกรมสามารถลดรูปกฎที่อยู่ในเงื่อนไข ดังนี้ 1) กฎที่เป็น subnet กัน 2) กฎที่ไม่สามารถไปถึง 3) กฎที่ขัดแย้งกัน
- โปรแกรมสามารถรายงานผลเพื่อบอกว่ากฎข้อไหนเป็นกฎที่เป็น subnet กัน กฎที่ไม่สามารถไปถึง และกฎที่ขัดแย้งกันได้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ช่วยเพิ่มประสิทธิภาพการทำงานของไฟร์วอลล์ และทำให้ผู้ดูแลระบบสามารถดูแลและตรวจสอบกฎของไฟร์วอลล์ได้ง่ายยิ่งขึ้น นอกจากนี้ยังสามารถนำโปรแกรมไปประยุกต์ใช้กับไฟร์วอลล์ตัวอื่นๆได้โดยปรับเปลี่ยนในส่วนของการแปลงกฎ

1.6 ขั้นตอนในการพัฒนาระบบงาน

- ศึกษาการหลักทำงานของไฟร์วอลล์
- วิเคราะห์ถึงปัญหาที่เกิดขึ้นเมื่อกฎของไฟร์วอลล์มีความซับซ้อน
- หาแนวทางในการแก้ปัญหา และเงื่อนไขที่จะใช้ในการลดรูปกฎของไฟร์วอลล์ได้
- ศึกษาโครงสร้างกฎของ IP Firewall และ IP Filter
- ทำการวิเคราะห์ และออกแบบโปรแกรมเครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์
- ออกแบบกฎของไฟร์วอลล์ที่เป็นรูปแบบกลาง
- พัฒนาโปรแกรมเครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์
- ทดสอบการใช้งาน และปรับปรุงแก้ไขโปรแกรมที่พัฒนาแล้ว
- สรุปผลการทดสอบจากการใช้งานที่เกิดขึ้น
- จัดทำเอกสารประกอบโครงงาน

1.7 รายละเอียดของแต่ละบท

- บทที่ 2 : รวบรวมทฤษฎีต่างๆ ที่เกี่ยวข้องกับไฟร์วอลล์ โดยจะกล่าวถึงหลักการทำงานของไฟร์วอลล์ รูปแบบโครงสร้างของกฎของ IP Firewall และ IP Filter และการ map กฎเข้ากับโครงสร้าง ศึกษาข้อมูลและเงื่อนไขที่ใช้ในการลดรูปกฎของไฟร์วอลล์
- บทที่ 3 : เป็นการออกแบบลักษณะการทำงานของโปรแกรม และออกแบบโครงสร้างของกฎที่เป็นรูปแบบกลาง พร้อมทั้งวิธีการ map กฎของ IP Firewall และ IP Filter ให้อยู่ในโครงสร้างที่เป็นรูปแบบกลาง
- บทที่ 4 : การพัฒนาระบบงาน จะเป็นการกล่าวถึง ภาษา และเครื่องมือที่ใช้ในการพัฒนาโปรแกรม
- บทที่ 5 : ผลการทดลอง
- บทที่ 6 : สรุปผลการพัฒนาโปรแกรม และข้อเสนอแนะ

บทที่ 2

IP Firewall และ IP Filter

2.1 ไฟร์วอลล์ และหลักการทำงานของไฟร์วอลล์

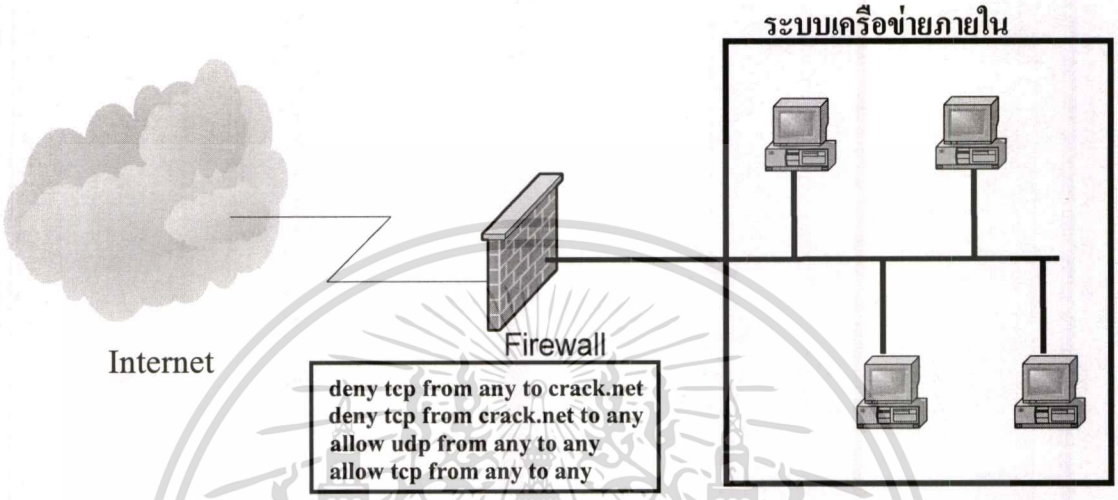
ไฟร์วอลล์ เป็นคอมพิวเตอร์หรืออุปกรณ์ทำหน้าที่ในการควบคุมการเข้า-ออกของข้อมูลระหว่างระบบเครือข่ายภายในองค์กร กับระบบเครือข่ายภายนอกองค์กรหรืออินเทอร์เน็ต ไฟร์วอลล์มีจุดประสงค์เพื่อป้องกันผู้บุกรุกที่มีจุดประสงค์ร้ายจากระบบเครือข่ายภายนอกองค์กรไม่ให้เข้ามาภายในระบบเครือข่ายขององค์กร จึงสามารถป้องกันความเสียหายจากผู้บุกรกที่เจาะเข้ามาในระบบเครือข่ายภายในองค์กรได้ เราสามารถจำแนกไฟร์วอลล์ออกได้เป็น 3 ประเภท คือ 1) ไฟร์วอลล์แบบการกรองแพ็คเก็ต 2) Application Proxy 3) Stateful Multilayer Inspection Firewall

IP Firewall และ IP Filter จะมีการทำงานแบบการกรองแพ็คเก็ต และยังสามารถทำงานแบบ Stateful Multilayer Inspection Firewall ได้อีกด้วย ซึ่งในเอกสารนี้จะอธิบายถึงลักษณะการทำงานของไฟร์วอลล์แบบการกรองแพ็คเก็ต และแบบ Stateful Multilayer Inspection Firewall เท่านั้นซึ่งไฟร์วอลล์ทั้ง 2 ชนิดจะมีการทำงาน ดังนี้

2.1.1 การกรองแพ็คเก็ต (Packet Filtering)

การกรองแพ็คเก็ตจะเป็นการอนุญาตหรือปฏิเสธแพ็คเก็ตที่เดินทางผ่านโดยตรวจสอบจากส่วนหัวของแพ็คเก็ต แล้วนำข้อมูลมาเปรียบเทียบกับกฎ (Rules) ที่ได้กำหนดไว้ เพื่อตรวจสอบว่าจะอนุญาตให้แพ็คเก็ตที่มีลักษณะใดผ่านได้บ้าง และแพ็คเก็ตลักษณะใดควรจะทิ้ง (Drop)ไป ซึ่งการพิจารณาส่วนหัวของแพ็คเก็ตจะดำเนินการในชั้น Network layer และ Transport layer ของ OSI Reference Model หรือชั้น IP ของ TCP/IP โดยที่ชั้น Network layer จะเป็นการตรวจสอบหมายเลข IP ต้นทาง หมายเลข IP ปลายทาง และชนิดของโปรโตคอล เช่น TCP ,UPD ,ICMP ส่วนในชั้นของ Transport layer จะตรวจสอบหมายเลขพอร์ตต้นทาง หมายเลขพอร์ตปลายทาง Flag (ซึ่งจะมีเฉพาะในส่วนหัวของแพ็คเก็ต TCP) และชนิดของ ICMP message (ในแพ็คเก็ต ICMP) โดยหมายเลขพอร์ตของชั้น Transport layer จะสามารถบอกถึงโปรแกรมประยุกต์ หรือการบริการที่แพ็คเก็ตนั้นต้องการติดต่อ เช่น หมายเลขพอร์ต 80 หมายถึง HTTP (Hyper Text Transfer Protocol) หมายเลขพอร์ต 21 หมายถึง FTP (File Transfer Protocol) หมายเลขพอร์ต 23 หมายถึง Telnet (Terminal Emulator) เป็นต้น เมื่อการกรองแพ็คเก็ตพิจารณาส่วนหัวของแพ็คเก็ตก็จะสามารถกำหนดกฎที่เหมาะกับการควบคุมแพ็คเก็ตที่เข้ามาได้ เช่น อาจจะห้ามแพ็คเก็ตที่มีต้นทางมาจาก

crack.cracker.net เข้ามายังระบบเครือข่ายขององค์กร routerโดยทั่วไปจะมีความสามารถในการทำการกรองแพ็คเก็ตอยู่แล้ว ซึ่งเราอาจเรียกว่า router เหล่านี้ว่า Screening Router



รูปที่ 2.1 การติดตั้งไฟร์วอลล์ระหว่างระบบเครือข่ายภายในองค์กรกับอินเทอร์เน็ต

ขั้นตอนในการกรองแพ็คเก็ตเป็นดังนี้

1) เงื่อนไขในการกรองแพ็คเก็ตสำหรับ router จะถูกเก็บไว้ ซึ่งเรียกเงื่อนไขเหล่านี้ว่า กฎของการกรองแพ็คเก็ต (Packet Filter Rules)

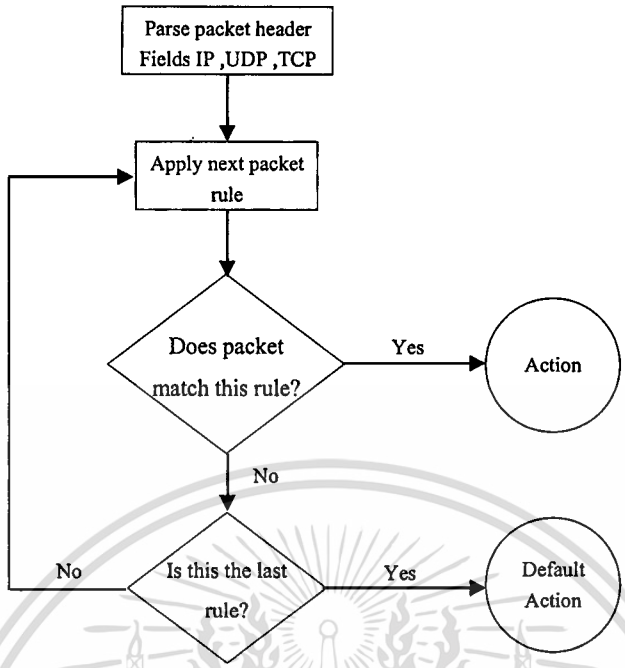
2) เมื่อแพ็คเก็ตมาถึงพอร์ตของ router ก็จะถูกตรวจสอบส่วนหัวโดยการตรวจสอบจะตรวจสอบส่วนหัวของ IP TCP และ UDP เท่านั้น

3) กฎของการกรองที่ได้จัดเก็บไว้จะมีการระบุถึงลำดับ ทำให้ตรวจสอบแต่ละแพ็คเก็ตเป็นลำดับตามกฎ

4) ถ้ากฎเป็นการไม่อนุญาตให้แพ็คเก็จผ่านออกไป หรือไม่อนุญาตให้รับแพ็คเก็ตเข้ามา ดังนั้นจะถือว่าแพ็คเก็จนี้ไม่ได้รับอนุญาต

5) ถ้ากฎเป็นการอนุญาตให้สามารถส่งผ่านแพ็คเก็ตออกไป หรืออนุญาตให้รับแพ็คเก็ตเข้ามาได้ ดังนั้นจะถือว่าแพ็คเก็ตนั้นได้รับอนุญาต

6) ถ้าแพ็คเก็ตไม่เข้าคู่กับกฎข้อใดเลยแพ็คเก็ตนี้จะได้รับอนุญาต หรือไม่ได้รับอนุญาตจะขึ้นอยู่กับ default rule ซึ่งจะเป็นกฎข้อสุดท้ายว่าจะอนุญาตหรือไม่ แต่โดยปกติจะไม่อนุญาต



รูปที่ 2.2 ผังขั้นตอนในการกรองแพ็คเก็ต

2.1.2 Stateful Multilayer Inspection Firewall

โดยปกติไฟร์วอลล์แบบการกรองแพ็คเก็ต (Stateless) จะควบคุมการเข้าออกของแพ็คเก็ตโดยพิจารณาข้อมูลจากส่วนหัวของแต่ละแพ็คเก็ตนำมาเปรียบเทียบกับกฎที่มีอยู่ ดังนั้นการกรองแพ็คเก็ตจึงไม่สามารถทราบได้ว่าแพ็คเก็ตนี้อยู่ส่วนใดของการเชื่อมต่อส่วนการทำงานของไฟร์วอลล์แบบ Stateful จะนำเอาส่วนข้อมูลของแพ็คเก็ต (message content) และข้อมูลที่ได้จากแพ็คเก็ตก่อนหน้านี้ที่ได้บันทึกเอาไว้ นำมาพิจารณาด้วย จึงทำให้สามารถระบุได้ว่าแพ็คเก็ตใดเป็นแพ็คเก็ตที่ติดต่อเข้ามาใหม่ หรือเป็นส่วนหนึ่งของการเชื่อมต่อที่มีอยู่แล้ว เช่น ถ้าไฟร์วอลล์สามารถจดจำสถานะได้ มันจะต้องจำสถานะของการเชื่อมต่อขาออกที่ไคลเอนต์ของ FTP เปิดไปหาเซิร์ฟเวอร์ FTP ไว้ก่อน ดังนั้นการเชื่อมต่อขาเข้าที่เปิดเข้ามาหาเครื่องไคลเอนต์ของ FTP ภายในจะต้องเป็นการเชื่อมต่อที่สอดคล้องหรือเข้าคู่กับการเชื่อมต่อที่ไคลเอนต์เคยเปิดออกไปหา แต่ถ้าหากการเชื่อมต่อจากภายนอกที่เข้ามาไม่มีคู่ของมันอยู่ ไฟร์วอลล์อาจจะไม่ยินยอมให้มีการติดต่อเข้ามา รวมทั้งความสามารถในการนำ fragment มารวมกัน (reassemble) แล้วจึงนำมาตรวจสอบข้อมูลกับกฎ ซึ่งโดยปกติการตรวจสอบการบุกรุกหรือร่องรอยของการกระทำใดๆนั้น router และไฟร์วอลล์ต้องอาศัยการพิจารณาจาก datagram ที่สมบูรณ์ แต่การรวม fragment ให้เป็น IP datagram นั้นจะมี

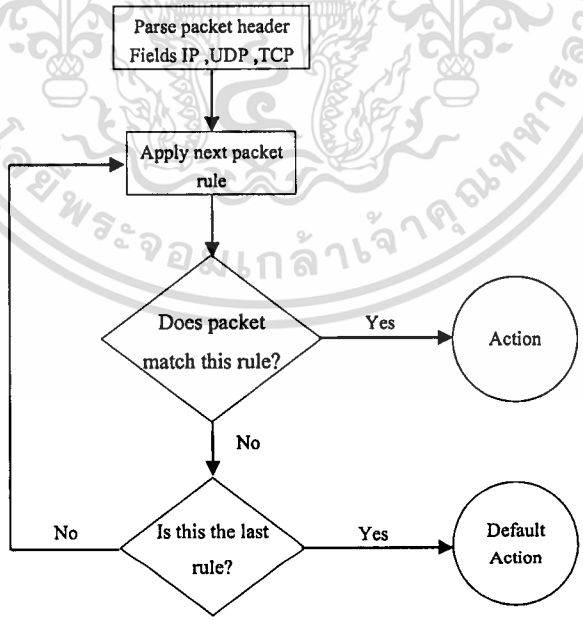
การทำที่โฮสต์ปลายทางเท่านั้นทำให้ router และไฟร์วอลล์ไม่สามารถตรวจสอบข้อมูลที่สมบูรณ์ได้

ไฟร์วอลล์แบบ stateful นั้นสามารถทำ reassemble ได้ทำให้สามารถตรวจสอบข้อมูลภายในที่สมบูรณ์ได้ แต่ในการทำ reassemble นั้นต้องใช้ทรัพยากรจำนวนมาก โดยเฉพาะหน่วยความจำ และการประมวลผล เนื่องจากไฟร์วอลล์จะต้องทำ reassemble ทุก datagram ที่มีการ fragment

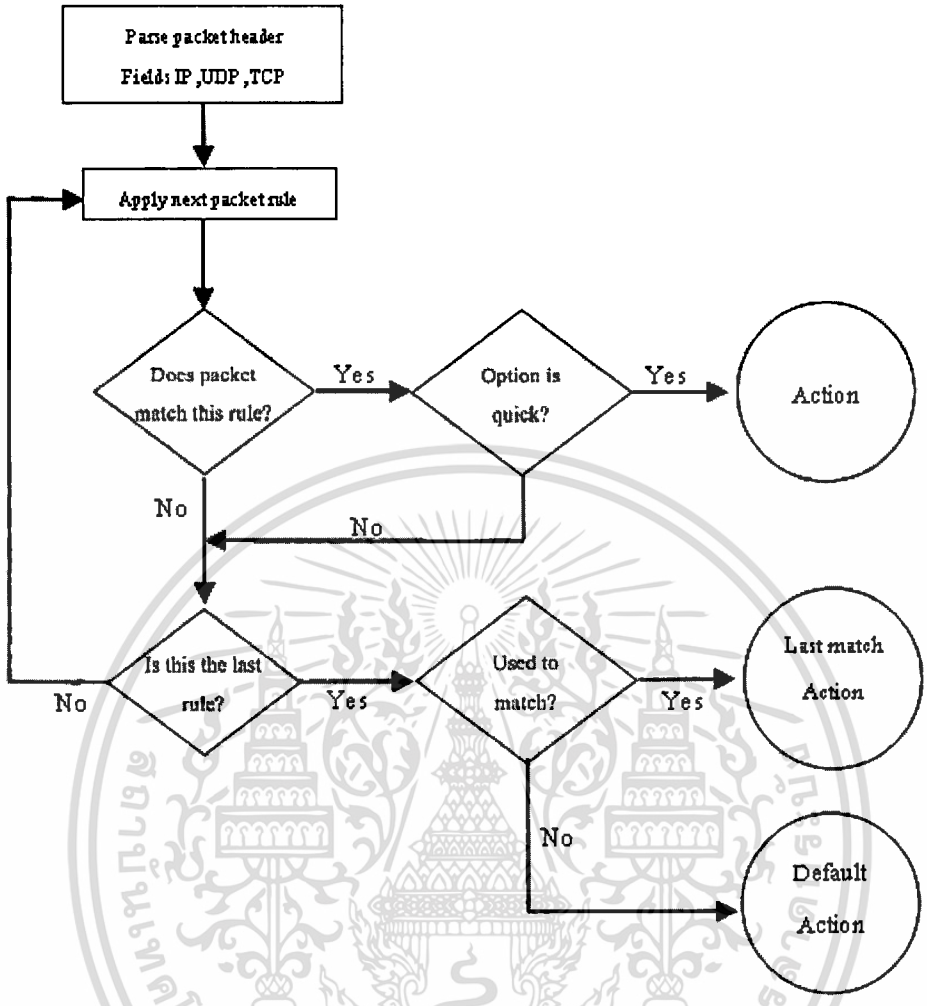
2.2 หลักการทำงานของ IP Firewall และ IP Filter

IP Firewall และ IP Filter จะมีการทำงานแบบการกรองแพ็คเก็ต คือ เมื่อมีแพ็คเก็ตผ่านไฟร์วอลล์ ไฟร์วอลล์ก็จะตรวจข้อมูลในส่วนหัวของแพ็คเก็ต กับกฎที่มีอยู่ในไฟร์วอลล์ ถ้ากฎอนุญาตให้แพ็คเก็ตนั้นผ่านไปได้แพ็คเก็ตก็สามารถผ่านไฟร์วอลล์ได้ แต่ถ้าไม่อนุญาตแพ็คเก็ตนั้นก็ไม่สามารถผ่านไฟร์วอลล์ได้

IP Firewall และ IP Filter จะมีส่วนแต่ต่างกันตรงการตรวจสอบข้อมูลแพ็คเก็ตกับกฎที่มีอยู่นั้น ถ้ามีการเข้าคู่กับกฎ IP Firewall จะดำเนินการทำตาม action ที่ระบุในกฎข้อนั้นเลย ส่วน IP Filter จะดำเนินการตรวจสอบกับกฎจนครบทุกข้อ และจะดำเนินการกับแพ็คเก็ตตามที่ระบุไว้ที่กฎข้อสุดท้ายที่มีการเข้าคู่ แต่ถ้ามีการระบุ option quick ไว้ IP Filter จะดำเนินการตาม action ทันทีเหมือนกับ IP Firewall ซึ่งการทำงานของ IP Firewall และ IP Filter จะแสดงดังแผนผังต่อไปนี้



รูปที่ 2.3 ผังการทำงานของ IP Firewall



รูปที่ 2.4 ผังการทำงานของ IP Filter

2.3 รูปแบบ โครงสร้างกฎของ IP Firewall และ IP Filter

การที่จะนำกฎของ IP Firewall และ IP Filter มาวิเคราะห์ และลดรูปจะต้องมีการศึกษาเกี่ยวกับรูปแบบ และโครงสร้างกฎของ IP Firewall และ IP Filter เพื่อใช้เป็นแนวทางในการออกแบบโครงสร้างกฎของโปรแกรมให้สามารถรองรับได้ทั้ง IP Firewall และ IP Filter นอกจากนี้ยังจะต้องมีการศึกษาถึงวิธีที่จะแปลงกฎที่อยู่ในลักษณะของคำสั่งในอยู่ในโครงสร้างที่เป็นภาษาซี เพื่อที่จะใช้ในการเขียนโปรแกรมในการวิเคราะห์ และลดรูปกฎ

2.3.1 รูปแบบกฎของ IP Firewall

รูปแบบกฎของ IP Firewall มีเป็นดังนี้

[**prob** *match_proba*bility] *action* [**log** [**logamount** *number*]]
proto **from** *src* **to** *dst* [*int*erface-spec] [*option*s]

prob *match_proability*

เข้าคู่ด้วยความน่าจะเป็นที่กำหนด (เป็นทศนิยมระหว่าง 0 ถึง 1)

action เป็นการกำหนดถึงผลของการกรอง มีดังนี้

allow

อนุญาตให้แพ็คเก็ตผ่านได้ อาจจะใช้คำว่า **pass** , **permit** และ **accept** แทน

Deny

เป็นการทิ้งแพ็คเก็ตโดยไม่มีการส่งแพ็คเก็ต ICMP ไปบอกต้นทางของแพ็คเก็ตที่ถูกทิ้ง อาจจะใช้คำว่า **drop** แทน

reject

(ไม่นิยมใช้ในปัจจุบัน) เป็นการทิ้งแพ็คเก็ต และส่งแพ็คเก็ต ICMP host หรือ port unreachable ไปยังต้นทางของแพ็คเก็ตที่ถูกทิ้ง

unreach *code*

ทิ้งแพ็คเก็ตที่เข้าคู่กับกฎ และส่ง ICMP unreachable ไปบอกโดยมีรหัส *code* ซึ่ง *code* มีค่าเป็น 0 ถึง 255 หรือใช้คำว่า **net , host , protocol , port , needfrag , srcfail , net-unknow , host-unknow , isolated , net-prohib , host-prohib , tosnet , toshost , filter-prohib , host-precedence** หรือ **precedence-cutoff**

reset

จะใช้เฉพาะแพ็คเก็ต TCP เท่านั้น จะเป็นการทิ้งแพ็คเก็ตที่เข้าคู่กับกฎ และส่ง TCP reset (RST)

count

เป็นการปรับค่าตัวนับแพ็คเก็ต แต่จะไม่มีการอนุญาตหรือปฏิเสธแพ็คเก็ตในกฎข้อนี้ ให้ไปยังตรวจสอบยังกฎถัดไป

check-state

จะตรวจสอบแพ็คเก็ตเทียบกับ dynamic ruleset ถ้ามีการเข้าคู่ก็จะหยุด ถ้าไม่ก็จะตรวจสอบที่กฎถัดไป ถ้ากฎไม่ได้ระบบ **check-state** จะมีการตรวจสอบ dynamic ruleset ที่กฎแรกที่มีคำว่า **keep-state**

divert *port*

ส่งแพ็คเก็ตที่เข้าคู่กับกฎนี้ไปยัง divert (4) socket ที่พอร์ต *port* และหยุดการค้นหา

tee *port*

คัดลอกแพ็คเก็ตที่ข้าคู่กับกฎนี้ แล้วส่งไปยัง divert (4) socket ที่พอร์ต *port* โดยจะหยุดการค้นหา และแพ็ตเก็ตต้นฉบับจะได้รับการยอมรับ

fwd *ipaddr*[,*port*]

แพ็คเก็ตที่เข้าคู่กับกฎจะมีการส่งต่อไปที่ *ipaddr* โดยจะกำหนดหมายเลข IP โดยใช้ dotted quad หรือชื่อโฮสต์

pipe *pipe_nr*

จะมีการผ่านแพ็คเก็ตไปยัง dummynet(4) "*pipe*" และการค้นหาจะสิ้นสุดลง

queue *queue_nr*

จะมีการผ่านแพ็คเก็ตไปยัง dummynet(4) "*queue*"

skipto *number*

จะมีการข้ามกฎที่มีหมายเลขน้อยกว่า *number* การค้นหาจะเริ่มที่กฎหมายเลข *number* หรือสูงกว่านั้น

log [logamount *number*]

ถ้า kernel มีการ compiled ด้วย option IPFIREWALL_VERBOSE เมื่อแพ็คเก็ตเข้าคู่กับกฎด้วย **log** keyword ข้อมูลจะถูกจัดเก็บ syslogd(8) ด้วย LOG_SECURITY หมายเหตุ โดยปกติข้อมูลจะต่อท้ายที่ไฟล์ /var/log/security ถ้า kernel compiled ด้วย option IPFIREWALL_VERBOSE_LIMIT โดยปกติการเก็บข้อมูลจะสิ้นสุดลงเมื่อมีจำนวนแพ็คเก็ตเกินจำนวนที่ระบุไว้ใน *net.inet.ip.fw.verbose_limit* แต่ถ้ามีการ **logamount** *number* จะมีการจำกัดจำนวนแพ็คเก็ตโดยใช้ *number*

proto

IP protocol จะระบุโดยใช้หมายเลข หรือชื่อ (ซึ่งอาจจะเป็น **ip** หรือ **all** หมายถึง ตรงกับทุกๆแพ็คเก็ต IP , **icmp** ตรงกับแพ็คเก็ต ICMP , tcp ตรงกับแพ็คเก็ต TCP , udp ตรงกับแพ็คเก็ต UDP อาจจะดูใน */etc/protocols*)

src และ *dst* :

any | me | [not] <*address/mask*> [*ports*]

ถ้าระบุเป็น **any** กฎจะเข้าคู่กับทุกหมายเลข IP

ถ้าระบุเป็น **me** กฎจะเข้าคู่กับทุกหมายเลข IP ที่ระบุบน interface ในระบบ

<*address/mask*> จะมีการระบุโดย

ipno เป็นหมายเลข IP ในรูปแบบ 1.2.3.4

ipno/bits เป็นหมายเลข IP กับความยาวของ mask ในรูปแบบ 1.2.3.4/24 ในกรณีนี้จะเข้าคู่กับทุกหมายเลข IP จาก 1.2.3.0 ถึง 1.2.3.255

ipno:mask เป็นหมายเลข IP กับความยาวของ mask ในรูปแบบ 1.2.3.4:255.255.240.0 ในกรณีนี้จะเข้าคู่กับทุกหมายเลข IP จาก 1.2.0.0 ถึง 1.2.15.255

ถ้าต้องการให้มีการเข้าคู่กับ address อื่นที่ไม่ได้การระบุไว้จะใช้คำว่า **not** แต่จะไม่มีผลกับหมายเลขพอร์ต

ถ้าใช้โปรโตคอล TCP หรือ UDP จะมีการตัวเลือก *ports* ซึ่งจะมีการระบุดังนี้

{*port*|*port-port*|*port:mask*}[,*port*[,...]]

เครื่องหมาย - เป็นการระบุพอร์ตเป็นช่วง

เครื่องหมาย : เป็นการระบุพอร์ต และ mask

อาจจะมีการใช้ชื่อของการบริการ (จาก /etc/services) แทนหมายเลขพอร์ต

แพ็คเก็ตที่มีการ fragmented จะไม่เข้าคู่กับกฎที่มีการระบุหมายเลขพอร์ต

interface-spec

จะมีการระบุ ดังต่อไปนี้

in จะเข้าคู่กับแพ็คเก็ตที่เดินทางเข้า

out จะเข้าคู่กับแพ็คเก็ตที่เดินทางออก

via *ifX* แพ็คเก็ตต้องเดินทางมาจาก interface *ifX*

via *if** แพ็คเก็ตต้องเดินทางมาจาก interface *ifX* โดยที่ *X* เป็นตัวเลขใดก็ได้

via any แพ็คเก็ตจะเดินทางจาก interface ไหนก็ได้

via *ipno* แพ็คเก็ตจะเดินทางจาก interface ที่มี IP address *ipno*

ถ้าใช้ keyword **via** จะมีการตรวจสอบ interface ทุกครั้ง ถ้าใช้ **recv** หรือ **xmit** แทน **via** ก็จะตรวจสอบเฉพาะ interface ที่รับเข้า และส่งออก ถ้ามีการระบุทั้งสองอย่าง ก็จะเข้าคู่ทั้ง interface ที่รับเข้า และส่งออก ตัวอย่างเช่น

ipfw add 100 deny ip from any to any out recv ed0 xmit ed1

ถ้าใช้ **xmit** จะต้องใช้ **out** ด้วย

options สามารถระบุได้ ดังนี้

keep-state

เป็นการบอกให้มีการเก็บข้อมูลของสถานะไว้ ซึ่งจะทำให้ไฟร์วอลล์ทำงานแบบ stateful

limit ***{src-addr src-port dst-addr dst-port}*** *N*

ไฟร์วอลล์จะอนุญาตเฉพาะการเชื่อมต่อจำนวน *N* ที่ระบุไว้ในพารามิเตอร์ ซึ่งพารามิเตอร์จะระบุได้หลายตัว

bridged

เข้าคู่เฉพาะแพ็คเก็ต bridged ใช้กับ multicast หรือ broadcast traffic ถ้าไม่เช่นนั้นแพ็คเก็ตจะเดินทางผ่านไฟร์วอลล์ 2 ครั้ง

frag

เข้าคู่ถ้าแพ็คเก็ตมีการ fragment และไม่ใช้ fragment แรกของ datagram **frag** ไม่สามารถใช้ร่วมกับ **tcpflags** หรือมีการระบุพอร์ต

ipoptions *spec*

จะเข้าคู่ถ้าส่วนหัวของ IP ตรงกับที่ระบุไว้ในส่วนของ *spec* ซึ่ง *spec* จะมีได้หลายตัวโดยจะขั้นด้วยเครื่องหมายจุลภาค (,) IP options ที่รองรับจะมีดังนี้ **ssrr** (strict source route), **lsrr** (loose source route), **rr** (record packet route) และ ts (time stamp)

tcpoptions *spec*

จะเข้าคู่ถ้าส่วนหัวของ TCP ตรงกับที่ระบุไว้ในส่วนของ *spec* ซึ่ง *spec* จะมีได้หลายตัวโดยจะขั้นด้วยเครื่องหมายจุลภาค (,) TCP options ที่รองรับจะมีดังนี้ **mss** (maximum seqment size), **window** (tcp window advertisement), **sack** (selective ack) , **ts** (rfc1323 time stamp) และ **cc** (rfc1644 t/tcp connection count)

established

จะเข้าคู่ถ้าแพ็คเก็ตเป็นส่วนของ TCP ที่ได้มีการสร้างการเชื่อมต่อไว้แล้ว (บิต RST หรือ ACK มีค่าเป็น 1)

setup

จะเข้าคู่ถ้าแพ็คเก็ต TCP ได้มีการพยายามสร้างการเชื่อมต่อ (บิต SYN มีค่าเป็น 1 แต่บิต ACK มีค่าเป็น 0)

tcpflags *spec*

จะเข้าคู่ถ้าส่วนหัวของ TCP ตรงกับที่ระบุไว้ในส่วนของ *flags* ซึ่ง *flags* จะมีได้หลายตัว โดยจะขั้นด้วยเครื่องหมายจุลภาค (,) flags ที่รองรับจะมีดังนี้ **fin, syn, rst, psh, ack** และ **urg**

icmptypes *types*

จะเข้าคู่กับชนิดของ ICMP ที่กำหนดในรายการของ *types* ซึ่งอาจจะระบุเป็นช่วง หรือแต่ละชนิดแล้วขั้นด้วยเครื่องหมายจุลภาค (,) โดยปกติแล้วชนิดของ ICMP จะเป็นดังนี้ 0 echo reply (ping reply), 3 destination unreachable , 4 source quench , 5 redirect (ping request) , 8 echo request , 9 router advertisement , 10 router solicitation , 11 time exceeded (จะถูกใช้ในการแสดงว่าค่า TTL นั้นสิ้นสุด ซึ่งจะใช้กับ traceroute) , 12 IP header bad , 13 timestamp request , 14 timestamp reply , 15 information request , 16 information reply , 17 address mask request และ 18 address mask reply

uid *user*

เข้าคู่กับแพ็คเก็ต TCP หรือ UDP ที่ส่งหรือรับสำหรับ *user* โดยที่ *user* อาจจะเป็นชื่อ หรือหมายเลข

gid *group*

เข้าคู่กับแพ็คเก็ต TCP หรือ UDP ที่ส่งหรือรับสำหรับ *group* โดยที่ *group* อาจจะเป็นชื่อหรือหมายเลข

2.3.2 โครงสร้างกฎของ IP Firewall

โครงสร้างกฎของ IP Firewall จะเป็นตัวที่ใช้ในการเก็บข้อมูลของกฎ ซึ่งจะมีการจัดเก็บอยู่ในรูปแบบของภาษาซี โดยจะมีโครงสร้างดังนี้

```
#ifndef _IP_FW_H
#define _IP_FW_H
#include <sys/queue.h>
/*
 * เป็นโครงสร้างแบบ union เพื่อใช้ในการระบุ interface ซึ่งจะสามารถระบุโดยชื่อ หรือ
 * หมายเลขไอพี flags IP_FW_F_IIFNAME และ IP_FW_F_OIFNAME จะใช้กับโครงสร้างนี้
 * ถ้า interface unit มีค่าเป็น -1 หมายถึงสามารถเข้าคู่ได้กับทุก interface
```

```
 * หมายเลยไอพีที่เป็น 0.0.0.0 หมายถึงสามารถเข้าคู่ได้กับทุก interface
 *
 * จะมีการเปรียบเทียบ interfaces ที่ส่งเข้าและรับออก ก็ต่อเมื่อ
 * bit (IP_FW_F_IIFACE or  IP_FW_F_OIFACE) ถูก set ไว้
 */
union ip_fw_if {
        struct in_addr  fu_via_ip;      /* เป็นการระบุโดยหมายเลขไอพี */
        struct {                        /* เป็นการระบุโดยชื่อ interface */
#define FW_IFNLEN       10
                char    name[FW_IFNLEN];
                short   unit;           /* -1 หมายถึงเข้าคู่กับทุก interface */
        } fu_via_if;
};
/*
* fw_src, fw_dst, fw_smsk, fw_dmsk จะเก็บในรูปแบบของ network byte order
 * fw_flg and fw_n*p จะเก็บในรูปแบบ host byte order
 * หมายเลขพอร์ตจะเก็บในรูปแบบ HOST byte order
 */
struct ip_fw {
        u_int32_t       fw_flg;         /* Flags */
        struct in_addr  fw_src;         /* หมายเลขไอพีต้นทาง */
        struct in_addr  fw_dst;         /* หมายเลขไอพีปลายทาง*/
        struct in_addr  fw_smsk;        /* Mask ของหมายเลขไอพีต้นทาง */
        struct in_addr  fw_dmsk;        /* Mask ของหมายเลขไอพีปลายทาง */
        u_short         fw_number;      /* หมายเลขกฎ */
        u_char          fw_prot;        /* โปรโตคอลไอพี */
#if 1
        u_char          fw_nports;      /* จำนวนของพอร์ตต้นทางและปลายทางที่อยู่ในอาเรย์ */
#define IP_FW_GETNSRCP(rule)                    ((rule)->fw_nports & 0x0f)
```

```
#define IP_FW_SETNSRCP(rule, n)     do {                              \
                                      (rule)->fw_nports &= ~0x0f;     \
                                      (rule)->fw_nports |= (n);       \
                                    } while (0)
#define IP_FW_GETNDSTP(rule)              ((rule)->fw_nports >> 4)
#define IP_FW_SETNDSTP(rule, n)     do {                              \
                                      (rule)->fw_nports &= ~0xf0;     \
                                      (rule)->fw_nports |= (n) << 4;\
                                    } while (0)
#define IP_FW_HAVEPORTS(rule)             ((rule)->fw_nports != 0)
#else
        u_char          __pad[1];
        u_int           _nsrcp;
        u_int           _ndstp;
#define IP_FW_GETNSRCP(rule)              (rule)->_nsrcp
#define IP_FW_SETNSRCP(rule,n)            (rule)->_nsrcp = n
#define IP_FW_GETNDSTP(rule)              (rule)->_ndstp
#define IP_FW_SETNDSTP(rule,n)            (rule)->_ndstp = n
#define IP_FW_HAVEPORTS(rule)             ((rule)->_ndstp + (rule)->_nsrcp != 0)
#endif
#define IP_FW_MAX_PORTS   10              /* จำนวนพอร์ตสูงสุด */
  union {
        u_short         fw_pts[IP_FW_MAX_PORTS];  /* หมายเลขพอร์ตต่างๆที่นำมาเข้าคู่ */
#define IP_FW_ICMPTYPES_MAX     128
#define IP_FW_ICMPTYPES_DIM     (IP_FW_ICMPTYPES_MAX / (sizeof(unsigned) * 8))
        unsigned        fw_icmptypes[IP_FW_ICMPTYPES_DIM];      /*ชนิดของICMP */
  } fw_uar;
        u_int           fw_ipflg;       /* IP flags */
        u_char          fw_ipopt;       /* IP options set */
        u_char          fw_ipnopt;      /* IP options unset */
```

```
        u_char          fw_tcpopt;      /* TCP options set */
        u_char          fw_tcpnopt;     /* TCP options unset */
        u_char          fw_tcpf;        /* TCP flags set/unset */
        u_char          fw_tcpnf;       /* TCP flags set/unset */

        union ip_fw_if  fw_in_if;       /* interfaces ที่แพ็คเก็ตเข้ามา*/
        union ip_fw_if  fw_out_if;      /* interfaces ที่แพ็คเก็ตออกไป*/
union {
        u_short         fu_divert_port; /* ใช้กับ Divert/tee port (options IPDIVERT) */
        u_short         fu_pipe_nr;     /* ใช้กับ queue number (option DUMMYNET) */

        u_short         fu_skipto_rule; /* ใช้กับคำสั่ง SKIPTO เพื่อไปยังกฎตามหมายเลขที่
                                         * ระบุ
                                         */
        u_short         fu_reject_code; /* ระบุ REJECT response code */
        struct sockaddr_in fu_fwd_ip;
} fw_un;
        /*
        * จำนวนของพอร์ตต้นทาง และพอร์ตปลายทางที่อยู่ในอาเรย์ (พอร์ตปลายทางจะอยู่หลัง
        * พอร์ตต้นทาง จำนวนสูงสุดของทั้งพอร์ตต้นทางและปลายทางเป็น 10
         * ถ้ามีจำนวนเป็น 0 หมายถึงเข้าคู่กับทุกพอร์ต
         */
        uid_t           fw_uid;                 /* หมายเลขผู้ใช้ที่ต้องการเข้าคู่ */
        gid_t           fw_gid;                 /* หมายเลขกลุ่มที่ต้องการเข้าคู่ */
        int             fw_logamount;           /* ปริมาณที่ต้องการ log ไว้ */
        u_int64_t       fw_loghighest;          /* จำนวนแพ็คเก็ตสูงสุดที่ log ไว้ */

        long            dont_match_prob;        /* 1.0 จะหมายถึงไม่มีการเข้าคู่ได้เลย */
        u_char          dyn_type;               /* ชนิดของ dynamic rule */
```

```
#define DYN_KEEP_STATE    0       /* ชนิดสำหรับกฎที่มีคำสั่ง keep-state */
#define DYN_LIMIT         1       /* ชนิดสำหรับกฎที่มีการจำกัดการเชื่อมต่อ */
#define DYN_LIMIT_PARENT  2       /* เป็น parent entry สำหรับกฎที่มีการจำกัดการเชื่อมต่อ
                                   */
        /*
         * 2 fields ต่อไปจะใช้ในการจำกัดจำนวนการเชื่อมต่อ
         * ซึ่งจะขึ้นอยู่กับ src, srcport, dst, dstport.
         */
        u_char          limit_mask;     /* mask ของ limit rule  */
#define DYN_SRC_ADDR    0x1
#define DYN_SRC_PORT    0x2
#define DYN_DST_ADDR    0x4
#define DYN_DST_PORT    0x8

        u_short         conn_limit;     /* จำนวนของการเชื่อมต่อที่ต้องการจำกัด */
};
#define fw_divert_port  fw_un.fu_divert_port
#define fw_skipto_rule  fw_un.fu_skipto_rule
#define fw_reject_code  fw_un.fu_reject_code
#define fw_pipe_nr      fw_un.fu_pipe_nr
#define fw_fwd_ip       fw_un.fu_fwd_ip

/*
 * ค่าต่างๆที่ใช้กับ "flags" field
 */
#define IP_FW_F_COMMAND 0x000000ff      /* Mask for type of chain entry: */
#define IP_FW_F_DENY    0x00000000      /* เป็นกฎที่มีการ deny */
#define IP_FW_F_REJECT  0x00000001      /* Deny และส่งแพ็คเก็ตตอบกลับ */
#define IP_FW_F_ACCEPT  0x00000002      /* เป็นกฎที่มีการ accept */
#define IP_FW_F_COUNT   0x00000003      /* เป็นกฎที่มีการ count */
```

```
#define IP_FW_F_DIVERT      0x00000004   /* เป็นกฎที่มีการเปลี่ยนทิศทางแพ็คเก็ต */
#define IP_FW_F_TEE         0x00000005   /* เป็นกฎที่มีการคัดลอกแพ็คเก็ต */
#define IP_FW_F_SKIPTO      0x00000006   /* เป็นการข้ามไปยังกฎอื่น */
#define IP_FW_F_FWD         0x00000007   /* เปลี่ยนที่อยู่ที่ต้องการส่งต่อ */
#define IP_FW_F_PIPE        0x00000008   /* เป็นกฎ dummynet */
#define IP_FW_F_QUEUE       0x00000009   /* เป็นกฎ dummynet queue */

#define IP_FW_F_IN          0x00000100   /* ตรวจสอบแพ็คเก็ตที่เข้ามา */
#define IP_FW_F_OUT         0x00000200   /* ตรวจสอบแพ็คเก็ตที่ออกไป */
#define IP_FW_F_IIFACE      0x00000400   /* ตรวจสอบ interface ขาเข้า */
#define IP_FW_F_OIFACE      0x00000800   /* ตรวจสอบ interface ขาออก */
#define IP_FW_F_PRN         0x00001000   /* พิมพ์กฎนี้เมื่อมีการเข้าคู่ */
#define IP_FW_F_SRNG        0x00002000   /* พอร์ตต้นทางสองพอร์ตแรกเป็นขอบเขต
                                          * ต่ำสุดและสูงสุด (เก็บอยู่ในรูปแบบ host byte
                                          * order)
                                          */
#define IP_FW_F_DRNG        0x00004000   /* พอร์ตปลายทางสองพอร์ตแรกเป็นขอบเขต
                                          * ต่ำสุดและสูงสุด (เก็บอยู่ในรูปแบบ host byte
                                          * order)
                                          */
#define IP_FW_F_FRAG        0x00008000   /* Fragment */
#define IP_FW_F_IIFNAME     0x00010000   /* interface เข้ามีระบุโดยชื่อและยูนิต */
#define IP_FW_F_OIFNAME     0x00020000   /* interface ออกมีระบุโดยชื่อและยูนิต */
#define IP_FW_F_INVSRC      0x00040000   /* ตรวจสอบต้นทางที่ตรงกันข้าม */
#define IP_FW_F_INVDST      0x00080000   /* ตรวจสอบปลายทางที่ตรงกันข้าม */
#define IP_FW_F_ICMPBIT     0x00100000   /* ชนิดของ ICMP */
#define IP_FW_F_UID         0x00200000   /* กรองโดย uid */
#define IP_FW_F_GID         0x00400000   /* กรองโดย gid */
#define IP_FW_F_RND_MATCH   0x00800000   /* มีการระบุความน่าจะเป็นในการเข้าคู่กับกฎ
                                          */
```

```
#define IP_FW_F_SMSK      0x01000000    /* src-port + mask */
#define IP_FW_F_DMSK      0x02000000    /* dst-port + mask */
#define IP_FW_BRIDGED     0x04000000    /* จะมีการเข้าคู่กับ bridged packets */
#define IP_FW_F_KEEP_S    0x08000000    /* เก็บสถานะ */
#define IP_FW_F_CHECK_S   0x10000000    /* ตรวจสอบสถานะ */
#define IP_FW_F_SME       0x20000000    /* ต้นทาง = me */
#define IP_FW_F_DME       0x40000000    /* ปลายทาง = me */

#define IP_FW_F_MASK      0x7FFFFFFF    /* ทุก flag ที่เป็นไปได้ */

/*
 * Flags สำหรับ 'fw_ipflg' เพื่อใช้เปรียบเทียบค่าของไอพี และโปรโตคอล
 * ไม่ได้มีการใช้ทั้งหมด ปัจจุบันใช้แค่ IP_FW_IF_*MSK ( IP_FW_IF_TCPEST )
 */
#define IP_FW_IF_TCPOPT   0x00000001    /* tcp options */
#define IP_FW_IF_TCPFLG   0x00000002    /* tcp flags */
#define IP_FW_IF_TCPSEQ   0x00000004    /* ลำดับของ tcp */
#define IP_FW_IF_TCPACK   0x00000008    /* หมายเลข tcp acknowledgement */
#define IP_FW_IF_TCPWIN   0x00000010    /* ขนาดของ tcp window */
#define IP_FW_IF_TCPEST   0x00000020    /* established TCP connection */
#define IP_FW_IF_TCPMSK   0x00000020    /* ทุกค่าของ tcp */

#define IP_FW_IF_IPOPT    0x00000100    /* ip options */
#define IP_FW_IF_IPLEN    0x00000200    /* ip length */
#define IP_FW_IF_IPID     0x00000400    /* ip identification */
#define IP_FW_IF_IPTOS    0x00000800    /* ip type of service */
#define IP_FW_IF_IPTTL    0x00001000    /* ip time to live */
#define IP_FW_IF_IPVER    0x00002000    /* ip version */
#define IP_FW_IF_IPMSK    0x00000000    /* ทุกค่าของ ip */
#define IP_FW_IF_MSK      0x00000020    /* ทุกค่าที่เป็นไปได้ของ flag */
```

```
/*
 * ความหมายของรหัสตอบกลับการ REJECT
 * ค่าที่น้อยกว่า 256 จะตรงกับ ICMP unreachable codes.
 */
#define IP_FW_REJECT_RST   0x0100 /* TCP packets: send RST */
/*
 * ความหมายของ IP option
 */
#define IP_FW_IPOPT_LSRR        0x01
#define IP_FW_IPOPT_SSRR        0x02
#define IP_FW_IPOPT_RR          0x04
#define IP_FW_IPOPT_TS          0x08
/*
 * ความหมายของ TCP option
 */
#define IP_FW_TCPOPT_MSS        0x01
#define IP_FW_TCPOPT_WINDOW     0x02
#define IP_FW_TCPOPT_SACK       0x04
#define IP_FW_TCPOPT_TS         0x08
#define IP_FW_TCPOPT_CC         0x10
/*
 * ความหมายของ TCP flags.
 */
#define IP_FW_TCPF_FIN          TH_FIN
#define IP_FW_TCPF_SYN          TH_SYN
#define IP_FW_TCPF_RST          TH_RST
#define IP_FW_TCPF_PSH          TH_PUSH
#define IP_FW_TCPF_ACK          TH_ACK
#define IP_FW_TCPF_URG          TH_URG
#endif /* _IP_FW_H */
```

2.3.3 การ map กฎของ IP Firewall ให้เข้ากับโครงสร้าง

รูปแบบกฎของ IP Firewall นั้นจะสามารถนำมา map ให้อยู่ในโครงสร้างที่เป็นภาษาซีเพื่อนำมาใช้ในการเขียนโปรแกรมควบคุมการทำงานของไฟร์วอลล์ ซึ่งจะทำได้ดังนี้

ตารางที่ 2.1 การ map กฎของ IP Firewall ให้เข้ากับโครงสร้าง

รูปแบบกฎ	โครงสร้างของกฎ
Probability	
prob *match_proability*	dont_match_prob = 1-*match_probility*;
Action	
allow ,pass ,permit ,accept	fw_flg |= IP_FW_F_ACCEPT;
deny ,drop	fw_flg |= IP_FW_F_DENY;
reject	fw_flg |= IP_FW_F_REJECT; fw_reject_code = ICMP_UNREACH_HOST;
unreach *code*	fw_flg |= IP_FW_F_REJECT; fw_reject_code = *code;*
reset	fw_flg |= IP_FW_F_REJECT; fw_reject_code = IP_FW_REJECT_RST
count	fw_flg |= IP_FW_F_COUNT;
check-state	fw_flg |= IP_FW_F_CHECK_S;
divert *port*	fw_flg |= IP_FW_F_DIVERT; fw_divert_port = *port*
tee *port*	fw_flg |= IP_FW_F_TEE; fw_divert_port = *port*
fwd *ipaddr[,port]*	fw_flg |= IP_FW_F_FWD; fw_fwd_ip = *ipaddr[,port]*
pipe *pipe_nr*	fw_flg |= IP_FW_F_PIPE; fw_divert_port = *pipe_nr*
queue *queue_nr*	fw_flg |= IP_FW_F_QUEUE; fw_divert_port = *pipe_nr*

skipto *number*	fw_flg |= IP_FW_F_SKIPTO; fw_skipto_rule = *number*
Log	
log [logamount *number*]	fw_flg |= IP_FW_F_PRN; fw_logamount = *number*
Protocol	
proto	fw_prot = *proto*
src **and** *dst*	
src	fw_src fw_smsk
dst	fw_dst fw_dmsk
ports	fw_pts[IP_FW_MAX_PORTS] u_char fw_nports;
interface-spec	fw_in_if; fw_out_if;
Options	
keep-state	fw_flg |= IP_FW_F_KEEP_S; dyn_type = DYN_KEEP_STATE;
limit {src-addr | src-port | dst-addr | **dst-port}** *N*	fw_flg |= IP_FW_F_KEEP_S; dyn_type = DYN_LIMIT; limit_mask; (DYN_SRC_ADDR,DYN_SRC_PORT, DYN_DST_ADDR,DYN_DST,PORT conn_limit = *N*;
bridged	fw_flg |= IP_FW_BRIDGED;
frag	fw_flg |= IP_FW_F_FRAG;
ipoptions *spec*	fw_ipopt; fw_ipnopt; (IP_FW_IPOPT_SSRR,IP_FW_IPOPT_LSRR,

	IP_FW_IPOPT_RR,IP_FW_IPOPT_TS)
tcpoptions *spec*	fw_tcpopt; fw_tcpnopt; (IP_FW_TCPOPT_MSS,IP_FW_TCPOPT_WINDOW ,IP_FW_TCPOPT_SACK ,IP_FW_TCPOPT_TS, IP_FW_TCPOPT_CC)
established	fw_ipflg |= IP_FW_IF_TCPEST;
setup	fw_tcpf |= IP_FW_TCPF_SYN; fw_tcpnf |= IP_FW_TCPF_ACK;
tcpflags *spec*	fw_tcpf; fw_tcpnf; (IP_FW_TCPF_SYN, IP_FW_TCPF_FIN, IP_FW_TCPF_ACK, IP_FW_TCPF_PSH, IP_FW_TCPF_RST , IP_FW_TCPF_URG)
icmptypes *types*	fw_uar.fw_icmptypes; fw_flg |= IP_FW_F_ICMPBIT;
uid *user*	fw_flg |= IP_FW_F_UID; fw_uid = *user*
gid *group*	fw_flg |= IP_FW_F_GID; fw_gid = *group*

2.3.4 รูปแบบกฎของ IP Filter

รูปแบบกฎของ IP Filter มีเป็นดังนี้

action in-out [*options*] [*tos*] [*ttl*]
[*proto*] [*ip*] [*group*]

action เป็นการกำหนดถึงผลของการกรอง มีดังนี้

pass

อนุญาตให้แพ็คเก็ตผ่านได้

block [return-icmp[return-code] | "return-rst"]

เป็นการทิ้งแพ็คเก็ต ซึ่งอาจจะมีการระบุได้ว่าจะมีการตอบกลับไปยังต้นทางหรือไม่ อย่างไร เช่น อาจจะตอบกับด้วย ICMP จะใช้ keyword ว่า return-icmp [*return-code*] และนอกจากนี้ยังมี keyword ต่างๆ return-rst จะเป็นการส่ง TCP reset ตอบกลับไป

count

จะเป็นการนับจำนวนแพ็คเก็ต แต่จะไม่นับแพ็คเก็ตที่ได้รับอนุญาต

log [body] [first] [or-block] [level *loglevel*]

เป็น keyword ที่ใช้บอกว่าให้เก็บข้อมูลเอาไว้

skip *number*

ข้ามกฎไปตามจำนวนที่ระบุ

auth

อนุญาตให้มีการ authentication โดยโปรแกรมของผู้ใช้ที่ทำงานอยู่

preauth

เป็นการบอกให้แพ็คเก็ตในกลุ่มนี้ไปตรวจสอบที่ pre-authentication เพื่อใช้ในการจัดกลุ่ม

call [now] *function-name*

เป็นการเรียกใช้ฟังก์ชันของ kernel ที่กำหนดไว้

in-out เป็นการกำหนดถึงทิศทางที่แพ็คเก็ตเดินทางผ่าน network interface มีดังนี้

in

จะเข้าคู่กับแพ็คเก็ตที่เดินทางเข้า

out

จะเข้าคู่กับแพ็คเก็ตที่เดินทางออก

options มีได้ดังต่อไปนี้

log [**body**] [**first**] [**or-block**] [**level** *loglevel*]

เป็น keyword ที่ใช้บอกว่าให้เก็บข้อมูลเอาไว้

quick

เป็น keyword ที่ใช้บอกให้ทำ action โดยไม่ต้องไปตรวจสอบกฎข้อถัดไป

on *interface-name* [**dup-to** *interface-name*[:ipaddr]] [**fastroute** | **to** *interface-name*]

on จะเป็นการระบุถึง interface **dup-to** จะเป็นการบอกให้สำเนาแพ็คเก็ตและส่งไปยัง interface ที่กำหนด **fastroute** จะเป็นการส่งแพ็คเก็ตอย่างรวดเร็ว **to** จะเป็นการส่งแพ็คเก็ตไปยัง

interface ที่กำหนดอย่างรวดเร็วซึ่งจะคล้ายกับ fastroute ชื่อ *interface* อาจมีดังนี้ lo (loopback), xl (3com ethernet) เป็นต้น

tos *number*

เป็นการระบุ Type of Service

ttl *number*

เป็นการระบุ Time-To-Live

proto *protocol*

เป็นการกำหนดถึงโปรโตคอล ชื่อโปรโตคอลเป็นชื่อที่มีอยู่ในไฟล์ /etc/protocols อาจมีดังนี้ tcp, udp, icmp หรืออาจจะใช้เป็นหมายเลขโปรโตคอลก็ได้ เราอาจจะระบุหลายโปรโตคอลโดยใช้เครื่องหมาย / ขั้นได้ เช่น tcp/upd

ip จะเป็นการบอกถึงหมายเลข IP ของต้นทาง และปลายทางมีรูปแบบดังนี้

srcdst [*flags*] [**with** *withopt*] [*icmp*] [*keep*]

srcdst สามารถระบุเป็น

all หมายถึงเข้าคู่กับทุกหมายเลข IP

from [!] *object* **to** [!] *object*

object จะมีรูปแบบ ดังนี้

addr [*port*]

addr อาจระบุเป็น **any** ซึ่งจะหมายถึงทุกๆ address หรือ host-name/mask หรือ host-name **mast** maskหรือ IP address หรือ host-name ก็ได้ในที่นี้ host-name อาจหมายถึง หมายเลข IP หรือชื่อโฮสต์ หรือ any ก็ได้

port จะใช้งานได้กับโปรโตคอล TCP และ UDP ซึ่งสามารถใช้ได้ทั้งหมายเลขพอร์ตและชื่อของการบริการที่มาจากไฟล์ /etc/services

operands ที่สามารถใช้กับหมายเลขพอร์ตมีดังนี้

ตารางที่ 2.2 operands ที่สามารถใช้กับหมายเลขพอร์ต

Operand	**Alias**	**Parameter Result**	
<	lt	port#	เป็นจริงถ้าพอร์ตมีค่าน้อยกว่าค่าที่กำหนด
>	gt	port#	เป็นจริงถ้าพอร์ตมีค่ามากกว่าค่าที่กำหนด
=	eq	port#	เป็นจริงถ้าพอร์ตมีค่าเท่ากับค่าที่กำหนด
!=	ne	port#	เป็นจริงถ้าพอร์ตมีค่าไม่เท่ากับค่าที่กำหนด
<=	le	port#	เป็นจริงถ้าพอร์ตมีค่าน้อยกว่าหรือเท่ากับค่าที่กำหนด
=>	ge	port#	เป็นจริงถ้าพอร์ตมีค่ามากกว่าหรือเท่ากับค่าที่กำหนด

หมายเลขพอร์ตสามารถกำหนดเป็นช่วงได้โดยใช้

ตารางที่ 2.3 การกำหนดพอร์ตเป็นช่วง

	Operand		
port1#	<>	port2#	เป็นจริงถ้าพอร์ตมีค่าน้อยกว่า port1 หรือมากกว่า port2
port1#	><	port2#	เป็นจริงถ้าพอร์ตมีค่ามากกว่า port1 และน้อยกว่า port2

flags

บอกถึง TCP flags เช่น F(FIN) , S(SYN), R(RST) , P(PUSH) , A(ACK) , U(URG) , SA (SYN-ACK)

withopt

จะนำหน้าด้วยคำว่า **with** หรือ **and** โดย *withopt* จะเป็น **[not|no]** *opttype* [*withopt*]

opttype อาจจะเป็น **ipopts** หรือ **short** หรือ **frag** หรือ **opt** *ipopts*

ipopts มีค่าเป็น nop , rr , zsu , ssrr , ts ,tr เป็นต้น

icmp-type *type* **หรือ code** *code*

เป็นการระบุชนิดของโปรโตคอล ICMP โดย *type* อาจจุระบุเป็นชื่อหรือหมายเลขก็ได้ เช่น 0 echo reply (ping reply), 3 destination unreachable, 5 redirect (ping request) และ 11 time exceeded (จะถูกใช้ในการแสดงว่าค่า TTL นั้นสิ้นสุด ซึ่งจะใช้กับ traceroute)

keep state

เป็นการบอกให้มีการเก็บข้อมูลของสถานะไว้ ซึ่งจะทำให้ไฟร์วอลล์ทำงานแบบ stateful

keep frags

เป็นการบอกให้ไฟร์วอลล์สังเกตและเก็บข้อมูลของเส้นทางของแพ็คเก็ตที่ถูก fragment *group* สามารถระบุเป็น **head** *number* **หรือ group** *number* **head** จะหมายถึงกลุ่มใหม่ **group** จะหมายถึงกฎควรอยู่ในกลุ่มไหน

2.3.5 โครงสร้างกฎของ IP Filter

โครงสร้างกฎของ IP Fiilter จะเป็นตัวที่ใช้ในการเก็บข้อมูลของกฎ ซึ่งจะมีการจัดเก็บอยู่ในรูปแบบของภาษาซีเช่นกัน โดยจะมีโครงสร้างดังนี้

```
#ifndef __IP_FIL_H__
#define __IP_FIL_H__

typedef struct  fr_ip  {
        u_32_t  fi_v:4;         /* IP version */
        u_32_t  fi_fl:4;        /* packet flags */
        u_32_t  fi_tos:8;       /* IP packet TOS */
        u_32_t  fi_ttl:8;       /* IP packet TTL */
        u_32_t  fi_p:8;         /* โปรโตคอลของ IP packet */
        union   i6addr fi_src;  /* หมายเลขต้นทางของแพ็คเก็ต */
        union   i6addr fi_dst;  /* หมายเลขปลายทางของแพ็คเก็ต*/
        u_32_t  fi_optmsk;      /* bitmask ของ IP options */
        u_short fi_secmsk;      /* bitmask ของ IP security options */
        u_short fi_auth;        /* รหัส authentication จาก IP sec. options */
} fr_ip_t;
```

```
#define FI_OPTIONS    (FF_OPTIONS >> 24)
#define FI_TCPUDP     (FF_TCPUDP >> 24)    /* TCP/UCP หมายถึงการเปรียบเทียบ */
#define FI_FRAG       (FF_FRAG >> 24)
#define FI_SHORT      (FF_SHORT >> 24)
#define FI_CMP        (FI_OPTIONS|FI_TCPUDP|FI_SHORT)

#define fi_saddr fi_src.in4.s_addr
#define fi_daddrfi_dst.in4.s_addr
/*
 * ใช้ในการบอกว่าพอร์ตมีการใช้ wildcard
 */
#define FI_W_SPORT          0x00000100
#define FI_W_DPORT          0x00000200
#define FI_WILDP      (FI_W_SPORT|FI_W_DPORT)
#define FI_W_SADDR          0x00000400
#define FI_W_DADDR          0x00000800
#define FI_WILDA      (FI_W_SADDR|FI_W_DADDR)

typedef struct    frdest    {
        void      *fd_ifp;
        union     i6addr    fd_ip6;
        char      fd_ifname[LIFNAMSIZ];
} frdest_t;

#define fd_ip     fd_ip6.in4
typedef struct    frpcmp    {
        int               frp_cmp;      /* ข้อมูลสำหรับการเปรียบเทียบพอร์ต */
        u_short           frp_port;     /* พอร์ตสำหรับ <> and >< */
        u_short           frp_top;      /* พอร์ต top สำหรับ <> and >< */
} frpcmp_t;
```

```
typedef struct    frtuc    {
        u_char          ftu_tcpfm;      /* tcp flags mask */
        u_char          ftu_tcpf;       /* tcp flags */
        frpcmp_t        ftu_src;
        frpcmp_t        ftu_dst;
} frtuc_t;

#define ftu_scmp        ftu_src.frp_cmp
#define ftu_dcmp        ftu_dst.frp_cmp
#define ftu_sport       ftu_src.frp_port
#define ftu_dport       ftu_dst.frp_port
#define ftu_stop        ftu_src.frp_top
#define ftu_dtop        ftu_dst.frp_top

typedef struct    frentry {
        struct    fr_ip    fr_ip;
        struct    fr_ip    fr_mip; /* mask */

        u_short fr_icmpm;       /* ข้อมูลสำหรับแพ็คเก็ต ICMP (mask) */
        u_short fr_icmp;

        u_int    fr_age[2];     /* อายุสำหรับสถานะ */
        frtuc_t  fr_tuc;
        u_32_t  fr_group;       /* กลุ่มที่กฎเป็นสมาชิกอยู่ */
        u_32_t  fr_grhead;      /* หมายเลขกลุ่มที่กฎนี้เริ่มต้น */
        u_32_t  fr_flags;       /* flags && options (ดูข้างล่าง) */
        u_int    fr_skip;       /* จำนวนของกฎที่ต้องข้ามไป */
        u_int    fr_loglevel;   /* syslog log facility + priority */
        int      (*fr_func) __P((int, ip_t *, fr_info_t *));  /* เรียกใช้ฟังก์ชันนี้ */
        u_char  fr_icode;       /* รหัส ICMP ที่ตอบกลับ */
```

```
        char    fr_ifnames[4][LIFNAMSIZ];
        struct  frdest  fr_tif;  /* "to" interface */
        struct  frdest  fr_dif;  /* คัดลอกแพ็คเก็ตไปยัง interfaces */
} frentry_t;

#define fr_v            fr_ip.fi_v
#define fr_proto        fr_ip.fi_p
#define fr_ttl          fr_ip.fi_ttl
#define fr_tos          fr_ip.fi_tos
#define fr_tcpfm        fr_tuc.ftu_tcpfm
#define fr_tcpf         fr_tuc.ftu_tcpf
#define fr_scmp         fr_tuc.ftu_scmp
#define fr_dcmp         fr_tuc.ftu_dcmp
#define fr_dport        fr_tuc.ftu_dport
#define fr_sport        fr_tuc.ftu_sport
#define fr_stop         fr_tuc.ftu_stop
#define fr_dtop         fr_tuc.ftu_dtop
#define fr_dst          fr_ip.fi_dst.in4
#define fr_src          fr_ip.fi_src.in4
#define fr_dmsk         fr_mip.fi_dst.in4
#define fr_smsk         fr_mip.fi_src.in4
#define fr_ifname       fr_ifnames[0]
#define fr_oifname      fr_ifnames[2]
#define fr_ifa          fr_ifas[0]
#define fr_oifa         fr_ifas[2]

#define FR_CMPSIZ   (sizeof(struct frentry) - offsetof(frentry_t, fr_ip))
/*
 * fr_flags
 */
```

```
#define FR_BLOCK        0x00001     /* ไม่อนุญาตให้แพ็คเก็ตผ่าน */
#define FR_PASS         0x00002     /* อนุญาตให้แพ็คเก็ตผ่าน */
#define FR_OUTQUE       0x00004     /* แพ็คเก็ตที่เดินทางออก */
#define FR_INQUE        0x00008     /* แพ็คเก็ตที่เดินทางเข้า */
#define FR_LOG          0x00010     /* Log */
#define FR_LOGB         0x00011     /* Log-fail */
#define FR_LOGP         0x00012     /* Log-pass */
#define FR_NOTSRCIP     0x00020     /* ไม่ใช่ src IP ที่กำหนด */
#define FR_NOTDSTIP     0x00040     /* ไม่ใช่ dst IP ที่กำหนด */
#define FR_RETRST       0x00080     /* ส่งแพ็คเก็ต TCP RST - reset connection
                                     * กลับไป
                                     */
#define FR_RETICMP      0x00100     /* ส่งแพ็คเก็ต ICMP unreachable */
#define FR_FAKEICMP     0x00180     /* ส่งแพ็คเก็ต ICMP unreachable with fake
                                     * source
                                     */
#define FR_NOMATCH      0x00200     /* ไม่มีการเข้าคู่เกิดขึ้น */
#define FR_ACCOUNT      0x00400     /* นับไบต์ของแพ็คเก็ต */
#define FR_KEEPFRAG     0x00800     /* เก็บข้อมูลการ fragment */
#define FR_KEEPSTATE    0x01000     /* เก็บข้อมูลของสถานะเชื่อมต่อ */
#define FR_INACTIVE     0x02000
#define FR_QUICK        0x04000     /* เข้าและหยุดการตรวจสอบ */
#define FR_FASTROUTE    0x08000     /* ส่งผ่าน normal routing */
#define FR_CALLNOW      0x10000     /* เรียกใช้ฟังก์ชัน (fr_func) ถ้ามีการเข้าคู่ */
#define FR_DUP          0x20000     /* คัดลอกแพ็คเก็ต */
#define FR_LOGORBLOCK   0x40000     /* ไม่ให้แพ็คเก็ตผ่านถ้าไม่สามารถ log ได้/
#define FR_LOGBODY      0x80000     /* Log the body */
#define FR_LOGFIRST     0x100000    /* Log ไบต์แรกถ้าสถานะยังเหมือนเดิม */
#define FR_AUTH         0x200000    /* ใช้ authentication */
#define FR_PREAUTH      0x400000    /* ร้องขอการทำ preauthentication */
```

```
#define FR_DONTCACHE       0x800000       /* ไม่มีการเก็บผลลัพธ์ */
#define FR_LOGMASK (FR_LOG|FR_LOGP|FR_LOGB)
#define FR_RETMASK (FR_RETICMP|FR_RETRST|FR_FAKEICMP)
/*
 * ความหมายที่กำหนดสำหรับ FI_* และถูกที่จัดเก็บใน fr_flags
 */
#define FF_OPTIONS   0x01000000
#define FF_TCPUDP    0x02000000
#define FF_FRAG      0x04000000
#define FF_SHORT     0x08000000
/*
 * flags สำหรับ SIOCGETFF และ SIOCSETFF, และ fr_flags
 */
#define FF_LOGPASS          0x10000000
#define FF_LOGBLOCK         0x20000000
#define FF_LOGNOMATCH       0x40000000
#define FF_LOGGING  (FF_LOGPASS|FF_LOGBLOCK|FF_LOGNOMATCH)

#define FR_NONE 0
#define FR_EQUAL 1
#define FR_NEQUAL 2
#define FR_LESST 3
#define FR_GREATERT 4
#define FR_LESSTE 5
#define  FR_GREATERTE 6
#define FR_OUTRANGE 7
#define FR_INRANGE 8
#ifndef  ICMP_UNREACH_FILTER
# define ICMP_UNREACH_FILTER      13
#endif
```

```
#ifndef  IPF_LOGGING
# define IPF_LOGGING 0
#endif
#ifndef  IPF_DEFAULT_PASS
# define IPF_DEFAULT_PASS   FR_PASS
#endif

#define  IPMINLEN(i, h)((i)->ip_len >= ((i)->ip_hl * 4 + sizeof(struct h)))
#define  IPLLOGSIZE    8192

#define  IPF_OPTCOPY 0x07ff00          /* bit mask ของ copied options */

#endif   /* __IP_FIL_H__ */
```

2.3.6 การ map กฎของ IP Filter ให้เข้ากับโครงสร้าง

รูปแบบกฎของ IP Filter นั้นจะสามารถนำมา map ให้อยู่ในโครงสร้างที่เป็นภาษาซีเพื่อนำมาใช้ในการเขียนโปรแกรมควบคุมการทำงานของไฟร์วอลล์ ซึ่งจะทำได้ดังนี้

ตารางที่ 2.4 การ map กฎของ IP Filter ให้เข้ากับโครงสร้าง

รูปแบบกฎ	โครงสร้างกฎ
Action	
pass	fr_flag |= FR_PASS;
Block	
block	fr_flags |= FR_BLOCK;
block return-icmp-as-dest *return-code*	fr_flags |= FR_BLOCK; fr_flags |= FR_FAKEICMP;
block return-icmp *return-code*	fr_flags |= FR_BLOCK; fr_flags |= FR_ RETICMP; fr_icode = *return-code*
block return-reset	fr_flags |= FR_BLOCK;

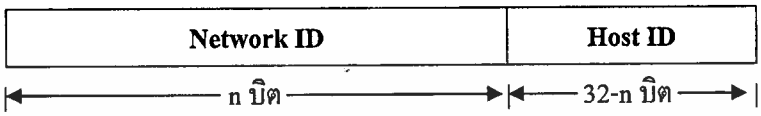
	fr_flags |= FR_RETRST'
count	fr_flags |= FR_ACCOUNT;
skip *number*	fr_skip = *number*
auth	fr_flags |= FR_AUTH;
preauth	fr_flags |= FR_ PREAUTH;
call *function-name*	fr_flags |= FR_ FR_CALLNOW; fr_func = *function-name*
log	
log	fr_flags |= FR_LOG;
body	fr_flags |= FR_LOG; fr_flags |= FR_LOGBODY;
first	fr_flags |= FR_LOG; fr_flags |= FR_LOGFIRST;
or-block	fr_flags |= FR_LOG; fr_flags |= FR_ LOGORBLOCK;
level *loglevel*	fr_flags |= FR_LOG; fr_loglevel = *loglevel*
in-out	
in	fr_flags |= FR_INQUE;
out	fr_flags |= FR_OUTQUE;
options	
log	ให้ดูข้างบน
quick	fr_flags |= FR_QUICK;
on *interface-name*	fr_ifname
dup-to interface-name[":"ipaddr]	fr_dif = interface-name[":"ipaddr];
fastroute	fr_flags |= FR_FASTROUTE;
to interface-name	fr_tif = interface-name;
tos *number*	fr_tos = *number*; fr_mip.fi_tos = 0xff;

ttl *number*	fr_ttl = *number*; fr_mip.fi_ttl = 0xff;
proto *protocol*	
tcp/udp	fr_ip.fi_fl |= FI_TCPUDP; fr_mip.fi_fl |= FI_TCPUDP;
udp | tcp | icmp | *number*	fr_proto = udp | tcp | icmp | number
IP	
srcdst	
src	fr_flags |= FR_NOTSRCIP; (ถ้ามี ! ข้างหน้า) fr_src fr_smsk สำหรับ port ใช้ fr_scmp ,fr_sport ,fr_stop
dst	fr_flags |= FR_NOTDSTIP; (ถ้ามี ! ข้างหน้า) fr_dst fr_dmsk สำหรับ port ใช้ fr_dcmp ,fr_dport ,fr_dtop
flags	fr_tcpf = tcpflags; fr_tcpfm = tcpflagmsk;
with option	fr_ip.fi_fl |= oflags; fr_ip.fi_optmsk |= opts; fr_ip.fi_secmsk |= secmsk;
icmp-type	fr_proto != IPPROTO_ICMP fr_icmp = htons(fil.fr_icmp); fr_icmpm = htons(fil.fr_icmpm);
keep state	fr_flags |= FR_KEEPSTATE;
state-age	fr_age[0]; fr_age[1];
keep frags	fr_flags |= FR_KEEPFRAG;
head *number*	fr_grhead = number;
group *number*	fr_group = number;

2.4 ทฤษฎีที่เกี่ยวข้องกับการทำ subnet

2.4.1 หมายเลขไอพี

หมายเลขไอพีจะมีการเก็บข้อมูลเป็น 32 บิต โดยหมายเลยไอพีนั้นจะประกอบด้วย 2 ส่วน ด้วยกันคือ Network ID และ Host ID โดยถ้า Network ID มีจำนวน n บิต Host ID ก็จะมีจำนวน 32-n บิต ซึ่งจะมีลักษณะดังรูป



รูปที่ 2.5 ส่วนประกอบหมายเลขไอพี

เครือข่ายที่มี Network ID จำนวน n บิต อาจจะใช้แทนว่าเครือข่าย /n ตัวอย่างเช่น คณะเทศโนโลยีสารสนเทศมีหมายเลยไอพีตั้งแต่ 203.154.86.1 ถึง 203.154.86.255 จะใช้แทนว่า 203.154.86.0/23

2.4.2 Netmask

Netmask จะมีบิตในส่วนของ Network ID เป็น 1 ทั้งหมด Netmask จะใช้โดยโฮสต์และ router เพื่อใช้ในการระบุ Network ID

	Network ID	Host ID
IP Address :		
Netmask :	11111111111111111111111	00000000

รูปที่ 2.6 Netmask

ปกติ netmask จะเขียนให้อยู่ในรูปแบบของ dotted decimal notation (255.255.255.0) หรือ hexadecimal notation (0xFFFFFF00) ตัวอย่างเช่น ระบบเครือข่ายคลาส B (มี host ID 16 บิต) จะมี netmask เป็น 255.255.0.0

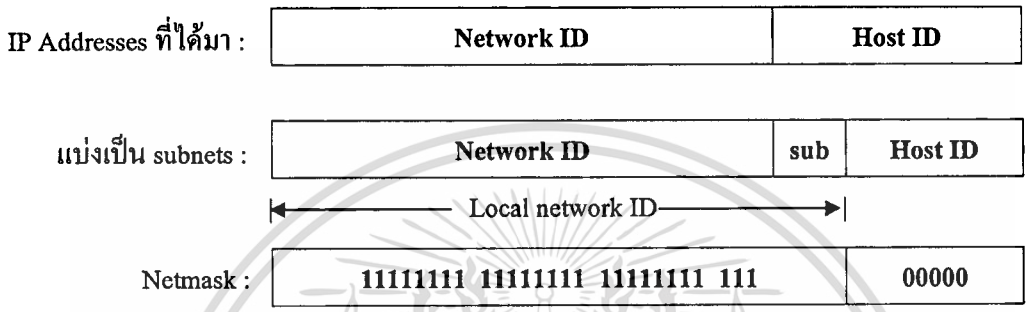
Netmask จะใช้ในการระบุถึง Network Address และ Host ID โดย

Network Address = (IP & Netmask)

Host ID = (IP & ~Netmask)

2.4.3 การทำ Subnet

Network address นั้นสามารถแบ่งเป็นระบบเครือข่ายย่อยๆได้ ซึ่งจะเรียกว่า subnets ตัวอย่างเช่น 203.154.24.0/24 เป็นเครือข่ายที่มี network address คลาส C ซึ่งปกติจะรับรองโฮสต์ได้ 253 โฮสต์จะสามารถแบ่งเป็น เครือข่ายที่รับโฮสต์ 125 โฮสต์ได้ 2 เครือข่าย คือ 203.154.24.0/25 และ 203.154.24.128/25 โดยทั้งสองเครือข่ายจะมี Network ID เป็น 25 บิต



รูปที่ 2.7 การทำ subnet

จากที่กล่าวมาข้างต้นจะเห็นได้ว่า ไฟร์วอลล์เป็นอุปกรณ์ที่ใช้ในการควบคุมการเข้า-ออกของข้อมูลระหว่างระบบเครือข่ายภายในองค์กร กับระบบเครือข่ายภายนอกองค์กร หรืออินเทอร์เน็ต ซึ่งจุดประสงค์ของไฟร์วอลล์นั้นเป็นไปเพื่อป้องกันการบุกรุกที่มีจุดประสงค์ร้ายจากระบบเครือข่ายภายนอกองค์กร อันจะก่อให้เกิดความเสียหายต่อระบบเครือข่ายภายในองค์กรได้

IP Firewall และ IP Filter จะมีการทำงานในได้ 2 รูปแบบด้วยกัน คือ แบบการกรองแพ็คเก็ต และ Stateful Multilayer Inspection Firewall โดยการทำงานแบบการกรองแพ็คเก็ตจะทำงานเมื่อมีแพ็คเก็ตผ่านไฟร์วอลล์ ไฟร์วอลล์ก็จะตรวจข้อมูลในส่วนหัวของแพ็คเก็ต กับกฎที่มีอยู่ในไฟร์วอลล์ ถ้ากฎอนุญาตให้แพ็คเก็ตนั้นผ่านไปได้แพ็คเก็ตก็สามารถผ่านไฟร์วอลล์ได้ แต่ถ้าไม่อนุญาตแพ็คเก็ตนั้นก็ไม่สามารถผ่านไฟร์วอลล์ได้ ส่วนการทำงานแบบ Stateful Multilayer Inspection Firewall จะทำงานคล้ายกับไฟร์วอลล์แบบการกรองแพ็คเก็ตแต่ได้มีเพิ่มความสามารถในการจดจำสถานะของการเชื่อมต่อ

โครงสร้างกฎของ IP Firewall และ IP Filter จะเป็นตัวที่ใช้ในการเก็บข้อมูลของกฎ ซึ่งจะอยู่ในรูปแบบของโครงสร้างแบบ struct ของภาษาซี ซึ่งในข้างต้นได้มีศึกษาถึงรูปแบบของกฎ โครงสร้างของกฎ และการ map จากกฎให้อยู่ในโครงสร้างของ IP Firewall และ IP Filter เพื่อใช้เป็นแนวทางในการออกแบบโครงสร้างกฎแบบกลางเพื่อใช้ในการพัฒนาโปรแกรมเครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์ ซึ่งจะกล่าวในบทถัดไป

นอกจากนี้ยังทำการศึกษาทฤษฎีที่เกี่ยวข้องกับการทำ subnet เพื่อใช้ในการวิเคราะห์ และลดรูปกฎ ซึ่งจะกล่าวในบทถัดไปเช่นกัน

บทที่ 3

การออกแบบระบบงาน

เนื้อหาในบทนี้จะกล่าวถึงหลักการทำงานของโปรแกรมเครื่องมือสำหรับวิเคราะห์ และลดจำนวนกฎของไฟร์วอลล์ รูปแบบกลางที่ได้ออกแบบไว้ วิธีการ map จากกฎของ IP Firewall และ IP Filter ให้อยู่ในรูปแบบกลาง รวมถึงอัลกอริทึมที่ใช้ในการวิเคราะห์ และลดรูปกฎของไฟร์วอลล์

3.1 ลักษณะการทำงานของโปรแกรม

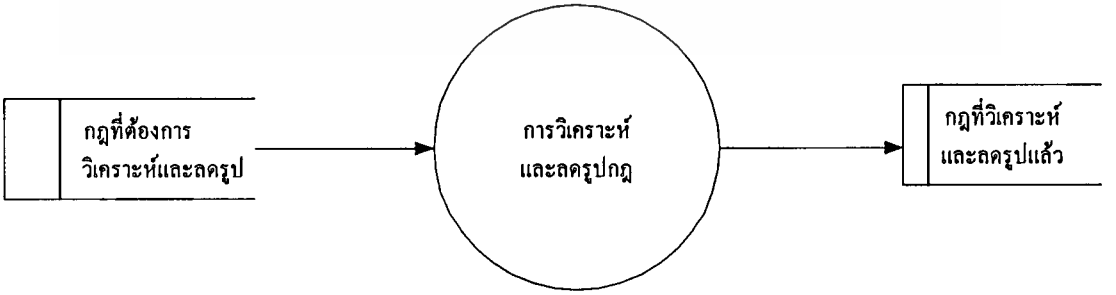
การทำงานของโปรแกรมจะเป็นการนำกฎของ Firewall ที่มีอยู่ในแฟ้มข้อมูลมาวิเคราะห์และลดรูป ซึ่งโปรแกรมจะอยู่ในรูปแบบของคำสั่ง โดยมีบอกพารามิเตอร์เป็นชื่อแฟ้มข้อมูลของกฎที่ต้องการวิเคราะห์ ชื่อแฟ้มข้อมูลของผลลัพธ์ (-o output-file) และชนิดของกฎของไฟร์วอลล์ว่าเป็น IP firewall (ไม่ต้องมี option) หรือ IP filter (-f)

ipfw_analysis [-f] [-o output-file] input-file

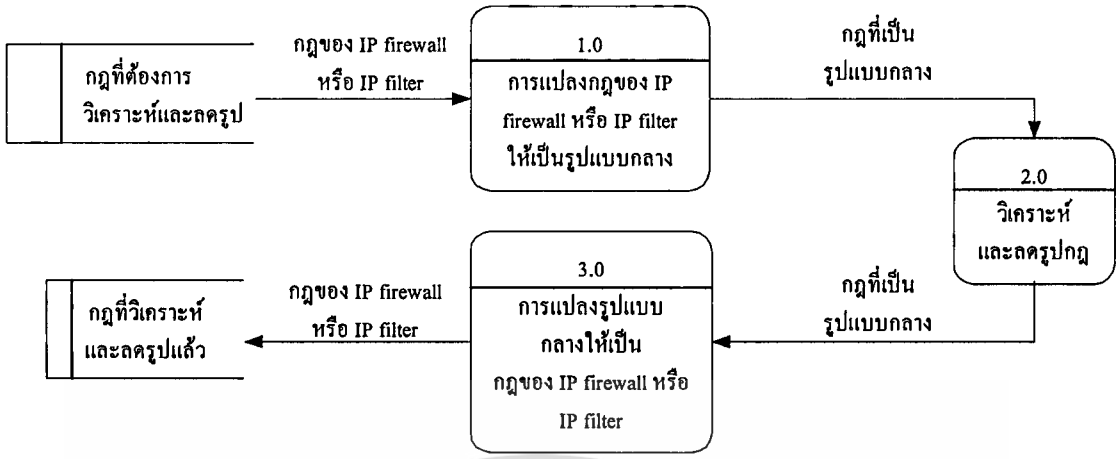
3.1.1 การทำงานของโปรแกรม

การทำงานของโปรแกรมจะออกแบ่งเป็น 3 ส่วนใหญ่ๆ ด้วยกันคือ

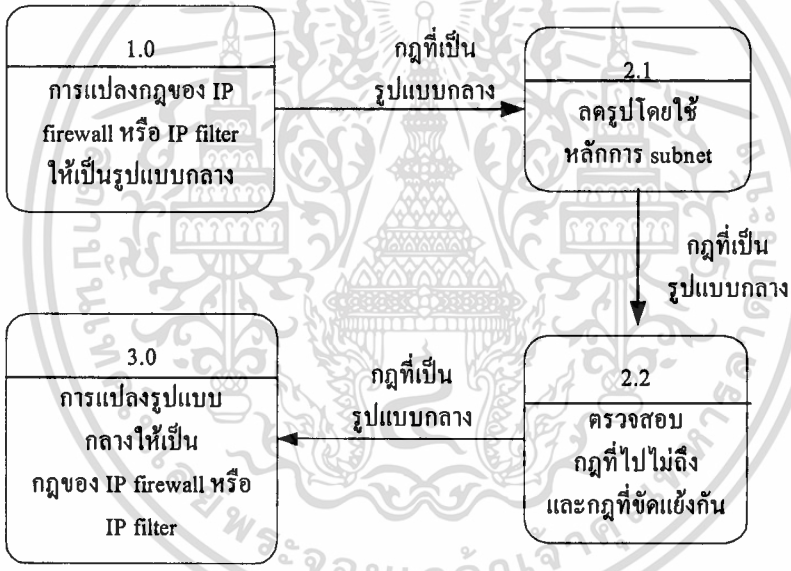
1) ส่วนที่แปลงกฎของ IP Firewall และ IP Filter ให้อยู่ในรูปแบบกลาง
2) ส่วนที่ทำการวิเคราะห์ และลดรูปกฎของไฟร์วอลล์
3) ส่วนที่แปลงกฎที่เป็นรูปแบบกลางให้เป็นกฎของ IP Firewall และ IP Filter



รูปที่ 3.1 Context Diagram



รูปที่ 3.2 Dataflow Diagram Level 1



รูปที่ 3.3 Dataflow Diagram Level 2 Process 2

ในส่วนของการวิเคราะห์ และลดรูปกฎ จะดำเนินการลดรูปโดยพิจารณาจาก หมายเลข IP ต้นทาง , หมายเลข IP ปลายทาง , หมายเลขพอร์ตต้นทาง , หมายเลขพอร์ตปลายทาง และชนิดของโปรโตคอล โดยจะมีการลดรูปต่างๆ ดังนี้

1) กฎที่สามารถรวมได้

ตรงส่วนนี้จะเป็นการตรวจสอบว่ามีหมายเลข IP ของต้นทาง หรือหมายเลข IP ปลายทางของกฎ 2 กฎว่าสามารถรวมกันเป็นกฎเดียวได้หรือไม่ ถ้าได้ก็จะนำมารวมกันโดยใช้หลักการของ

การรวม subnet ตัวอย่าง เช่น 203.157.12.0/25 กับ 203.157.12.128/25 สามารถรวมเป็น 203.157.12.0/24 ได้ โดยการรวมนั้นจะทำได้ก็ต่อเมื่อข้อมูลอื่นๆนั้นเหมือนกัน

2) กฎที่ไม่อาจไปถึงได้ และกฎที่มีการขัดแย้งกัน

โดยวิธีการ คือ จะต้องมีการจัดกลุ่มของ action เพื่อใช้เปรียบเทียบว่าเข้ากลุ่มเดียวกัน หรือขัดแย้งกัน โดยจะมีการแบ่งกลุ่มของ action ได้ดังนี้

ตารางที่ 3.1 กลุ่มของ Action

กลุ่มของ Action	**Action ต่างๆ**
unblock	allow , permit , accept , pass
block	block , deny , drop
redirect	divert *port* , fwd *ipaddr*[,*port*] , pipe *pipe_nr*
statistic	count , log
continue matching	check-state , skip *number* , auth , preauth
other	call *function-name*

เมื่อจัดกลุ่มของ Action แล้วจะได้ว่ามีส่วนของกลุ่มที่อนุญาต และกลุ่มที่ไม่อนุญาตจะมีการขัดแย้งกัน นอกจากนั้นจะไม่มีการขัดแย้งกัน การตรวจสอบดูว่ามีกฎใดที่มีหมายเลข IP ที่เป็น subset กันหรือไม่ (ข้อมูลในส่วนอื่นต้องเหมือนกัน) ถ้ามีก็จะตรวจสอบ action ว่ามีความขัดแย้งกันหรือไม่ ถ้า action ไม่มีความขัดแย้งกันก็จะบอกได้ว่ากฎที่เป็น subset เป็นกฎที่ไม่อาจไปถึงได้ และกฎนั้นก็จะถูกลบออกไป แต่ถ้า action มีความขัดแย้งกันก็จะบอกได้ว่ากฎมีความขัดแย้งกัน และกฎที่เป็น subset ก็จะถูกลบออกไปเช่นกัน

3.2 โครงสร้างกลางของกฎภายในโปรแกรม และการ map จากกฎของ IP Firewall และ IP Filter

3.2.1 โครงสร้างกลางของกฎภายในโปรแกรม

ก่อนที่จะมีการนำกฎของ IP Firewall และ IP Filter มาวิเคราะห์และลดรูป จะต้องมีการแปลงกฎของ IP Firewall และ IP Filter ให้อยู่ในโครงสร้างที่เป็นภาษาซี เพื่อที่จะนำไปใช้ในการเขียนโปรแกรมเพื่อวิเคราะห์ และลดรูปกฎ ซึ่งจะต้องมีการออกแบบโครงสร้างที่เป็นรูปแบบกลางไว้ โดยที่โครงสร้างกลางที่ออกแบบไว้จะต้องมีรูปแบบที่ครอบคลุมทั้งกฎของ IP Firewall และ IP Filter ซึ่งในที่นี้จะมีการยึดถือโครงสร้างกฎของ IP Firewall เป็นหลัก เนื่องจากโครงสร้างกฎของ IP Firewall มีความครอบคลุมโครงสร้างกฎของ IP Filter โครงสร้างกลางที่ออกแบบไว้จะเป็นดังนี้

```
#include <sys/queue.h>

#ifndef _IP_FW_H
#define _IP_FW_H

union ip_fw_if {
        struct in_addr          fu_via_ip;              /* เป็นการระบุโดยหมายเลขไอพี */
        struct {                                        /* เป็นการระบุโดยชื่อ interface */
#define FW_IFNLEN   10                                  /* need room ! was IFNAMSIZ */
                char    name[FW_IFNLEN];
                short   unit;                           /* -1 หมายถึงเข้าคู่กับทุก interface */
        } fu_via_if;
};
/*
 * โครงสร้างกลาง
 */
struct ip_fw {
        u_char          valid;          /* ใช้บอกว่ากฎข้อนี้ถูกลบแล้วหรือไม่ */
        u_int64_t       fw_flg;         /* Flags */
        u_char          action_group;   /* กลุ่มของ Action A = Allow , B = Block , L = Log ,
                                         * R = Redirect, C = Continue
                                         */
        u_char          reduce_code;    /* สาเหตุของการลดรูป C = Conflict rule ,
                                         * R = Unreachabled rule, S = Subneted rule
                                         */
        u_short         relate_rule;    /* หมายเลขกฎที่กฎข้อนี้ขัดแย้ง */
        u_char          quick;          /* เข้าคู่ และหยุดการเปรียบเทียบ */
        struct in_addr  fw_src;         /* หมายเลขไอพีต้นทาง */
        struct in_addr  fw_dst;         /* หมายเลขไอพีปลายทาง*/
        struct in_addr  fw_smsk;        /* Mask ของหมายเลขไอพีต้นทาง */
```

```
        struct in_addr  fw_dmsk;        /* Mask ของหมายเลขไอพีปลายทาง*/
        u_short         fw_number;      /* หมายเลขกฎ */
        u_char          fw_prot;        /* โปรโตคอลไอพี */
#if 1
        u_char          fw_nports;      /*จำนวนของพอร์ตต้นทางและปลายทางที่อยู่ในอาเรย์*/
#define IP_FW_GETNSRCP(rule)            ((rule)->fw_nports & 0x0f)
#define IP_FW_SETNSRCP(rule, n)         do {                            \
                                          (rule)->fw_nports &= ~0x0f;   \
                                          (rule)->fw_nports |= (n);     \
                                        } while (0)
#define IP_FW_GETNDSTP(rule)            ((rule)->fw_nports >> 4)
#define IP_FW_SETNDSTP(rule, n)         do {                            \
                                          (rule)->fw_nports &= ~0xf0;   \
                                          (rule)->fw_nports |= (n) << 4;\
                                        } while (0)
#define IP_FW_HAVEPORTS(rule)           ((rule)->fw_nports != 0)
#else
        u_char          __pad[1];
        u_int           _nsrcp;
        u_int           _ndstp;
#define IP_FW_GETNSRCP(rule)            (rule)->_nsrcp
#define IP_FW_SETNSRCP(rule,n)          (rule)->_nsrcp = n
#define IP_FW_GETNDSTP(rule)            (rule)->_ndstp
#define IP_FW_SETNDSTP(rule,n)          (rule)->_ndstp = n
#define IP_FW_HAVEPORTS(rule)           ((rule)->_ndstp + (rule)->_nsrcp != 0)
#endif
#define IP_FW_MAX_PORTS   10            /* จำนวนพอร์ตสูงสุด */
  union {
        u_short         fw_pts[IP_FW_MAX_PORTS];  /* หมายเลขพอร์ตต่างๆที่นำมาเข้าคู่ */
#define IP_FW_ICMPTYPES_MAX     128
```

```
#define IP_FW_ICMPTYPES_DIM     (IP_FW_ICMPTYPES_MAX / (sizeof(unsigned) * 8))
        unsigned        fw_icmptypes[IP_FW_ICMPTYPES_DIM]; /*ICMP types bitmap*/
  } fw_uar;
        /*
         * สำหรับการเปรียบเทียบ และระบุพอร์ตช่วง
         */
        u_char          fr_scmp;        /* ข้อมูลสำหรับการเปรียบเทียบพอร์ตต้นทาง */
        u_char          fr_dcmp;        /* ข้อมูลสำหรับการเปรียบเทียบพอร์ตปลายทาง */
        u_short         fr_sport;       /* พอร์ตสำหรับการเปรียบเทียบพอร์ตต้นทาง */
        u_short         fr_dport;       /* พอร์ตสำหรับการเปรียบเทียบพอร์ตปลายทาง */
        u_short         fr_stop;        /* พอร์ต top สำหรับการเปรียบเทียบพอร์ตต้นทาง */
        u_short         fr_dtop;        /* พอร์ต top สำหรับการเปรียบเทียบพอร์ตปลายทาง */

        u_int           fw_ipflg;       /* IP flags word */
        u_int           fw_mipflg;      /* Mask IP flags word */

        u_char          fw_ipopt;       /* IP options set */
        u_char          fw_ipnopt;      /* IP options unset */
        u_char          fw_tcpopt;      /* TCP options set */
        u_char          fw_tcpnopt;     /* TCP options unset */

        u_char          fw_tcpf;        /* TCP flags set/unset */
        u_char          fw_tcpnf;       /* TCP flags set/unset สำหรับ IP Firewall
                                         * และใช้เป็น mask ของ tcpf ใน IP Filter
                                         */

        u_short         fr_icmpm;       /* ข้อมูลสำหรับ ICMP packets (mask) */
        u_short         fr_icmp;        /* ข้อมูลสำหรับ ICMP packet */

        union ip_fw_if  fw_in_if;       /* interfaces ที่แพ็คเก็ตเข้ามา*/
```

```
        union ip_fw_if  fw_out_if;      /* interfaces ที่แพ็คเก็ตออกไป*/
        union ip_fw_if  fw_other_if[2]; /* interfaces อื่นๆ */
        union ip_fw_if  fw_to_if;       /* interfaces สำหรับ to-interface */
        union ip_fw_if  fw_dup_if;      /* interfaces ที่ต้องการคัดลอกแพ็คเก็ตส่งไป */

    union {
        u_short         fu_divert_port; /* ใช้กับ Divert/tee port (options IPDIVERT) */
        u_short         fu_pipe_nr;     /* ใช้กับ queue number (option DUMMYNET) */
        u_short         fu_skipto_rule; /* ใช้กับคำสั่ง SKIPTO เพื่อไปยังกฎตามหมายเลขที่
                                         * ระบุ
                                         */
        u_short         fu_reject_code; /* ระบุ REJECT response code */
        struct  sockaddr_in     fu_fwd_ip;      /* ระบุ REJECT response code */
    } fw_un;
        /*
        * จำนวนของพอร์ตต้นทาง และพอร์ตปลายทางที่อยู่ในอาเรย์ (พอร์ตปลายทางจะอยู่หลัง
        * พอร์ตต้นทาง จำนวนสูงสุดของทั้งพอร์ตต้นทางและปลายทางเป็น 10
         * ถ้ามีจำนวนเป็น 0 หมายถึงเข้าคู่กับทุกพอร์ต
         */

        uid_t           fw_uid;                 /* หมายเลขผู้ใช้ที่ต้องการเข้าคู่ */
        gid_t           fw_gid;                 /* หมายเลขกลุ่มที่ต้องการเข้าคู่ */
        u_int32_t       fr_group;               /* หมายเลขกลุ่ม (IP Filter) */
        u_int32_t       fr_grhead;              /* หมายเลขกลุ่มที่กฎข้อนี้เริ่มต้น (IP Filter) */
        int             fw_logamount;           /* ปริมาณที่ต้องการ log ไว้ */
        u_int64_t       fw_loghighest;          /* จำนวนแพ็คเก็ตสูงสุดที่จะ log ไว้ */
        u_int           fr_loglevel;            /* log facility + priority */

        long            dont_match_prob;        /* 1.0 จะหมายถึงไม่มีการเข้าคู่ได้เลย */
        u_char          dyn_type;               /* ชนิดของ dynamic rule */
```

```
#define DYN_KEEP_STATE    0       /* ชนิดสำหรับกฎที่มีคำสั่ง keep-state */
#define DYN_LIMIT         1       /* ชนิดสำหรับกฎที่มีการจำกัดการเชื่อมต่อ */
#define DYN_LIMIT_PARENT  2       /*เป็น parent entry สำหรับกฎที่มีการจำกัดการเชื่อมต่อ
                                   */

        /*
         * 2 fields ต่อไปจะใช้ในการจำกัดจำนวนการเชื่อมต่อ
         * ซึ่งจะขึ้นอยู่กับ src, srcport, dst, dstport.
         */
        u_char          limit_mask;     /* mask ของ limit rule  */

#define DYN_SRC_ADDR            0x1
#define DYN_SRC_PORT            0x2
#define DYN_DST_ADDR            0x4
#define DYN_DST_PORT            0x8

        u_short         conn_limit;             /* จำนวนของการเชื่อมต่อที่ต้องการจำกัด */

        u_int32_t       fr_tos:8;               /* จำนวนของการเชื่อมต่อที่ต้องการจำกัด */
        u_int32_t       fr_ttl:8;               /* IP packet TTL */
        u_int32_t       fr_fl:8;                /* packet flags */
        u_int32_t       fr_mtos:8;              /* Mark สำหรับ IP packet TOS */
        u_int32_t       fr_mttl:8;              /* Mark สำหรับ IP packet TTL */
        u_int32_t       fr_mfl:8;               /* Mark สำหรับ packet flags */
        u_int32_t       fr_optmsk;              /* bitmask ที่ประกอบไปด้วย IP options */
        u_int32_t       fr_moptmsk;             /* mask สำหรับ fr_optmsk */
        u_short         fr_secmsk;              /* bitmask ที่ประกอบไปด้วย IP security
                                                 * options
                                                 */
        u_short         fr_msecmsk;             /* mask สำหรับ fr_secmsk */
```

```
        u_int           fr_age[2];              /* อายุของสถานะ */
};
#define fw_divert_port          fw_un.fu_divert_port
#define fw_skipto_rule          fw_un.fu_skipto_rule
#define fw_reject_code          fw_un.fu_reject_code
#define fw_pipe_nr              fw_un.fu_pipe_nr
#define fw_fwd_ip               fw_un.fu_fwd_ip
/*
 * ค่าต่างๆที่ใช้กับ "flags" field
 */
#define IP_FW_F_COMMAND 0x0000000000000FFF   /* Mask for type of chain entry: */
#define IP_FW_F_DENY    0x0000000000000001   /* เป็นกฎที่มีการ deny */
#define IP_FW_F_REJECT  0x0000000000000002   /* Deny และส่งแพ็คเก็ตตอบกลับ */
#define IP_FW_F_ACCEPT  0x0000000000000004   /* เป็นกฎที่มีการ accept */
#define IP_FW_F_COUNT   0x0000000000000008   /* เป็นกฎที่มีการ count */
#define IP_FW_F_DIVERT  0x0000000000000010   /* เป็นกฎที่มีการเปลี่ยนทิศทาง
                                              * แพ็คเก็ต
                                              */
#define IP_FW_F_TEE     0x0000000000000020   /* เป็นกฎที่มีการคัดลอกแพ็คเก็ต */
#define IP_FW_F_SKIPTO  0x0000000000000040   /* เป็นการข้ามไปยังกฎอื่น */
#define IP_FW_F_FWD     0x0000000000000080   /* ส่งต่อไปยังตำแหน่งที่ระบุ */
#define IP_FW_F_PIPE    0x0000000000000100   /* เป็นกฎ dummynet */
#define IP_FW_F_QUEUE   0x0000000000000200   /* เป็นกฎ dummynet queue */
#define IP_FW_F_IN      0x0000000000001000   /* ตรวจสอบแพ็คเก็ตที่เข้ามา */
#define IP_FW_F_OUT     0x0000000000002000   /* ตรวจสอบแพ็คเก็ตที่ออกไป */
#define IP_FW_F_IIFACE  0x0000000000004000   /* ตรวจสอบ interface ขาเข้า */
#define IP_FW_F_OIFACE  0x0000000000008000   /* ตรวจสอบ interface ขาออก */
#define IP_FW_F_LOG     0x0000000000010000   /* Log เมื่อมีการเข้าคู่ */
#define IP_FW_F_LOGORBLOCK      0x0000000000020000   /* ไม่ให้แพ็คเก็ตผ่านถ้าไม่
                                                      * สามารถ log ได้
```

```
                                                    */
#define IP_FW_F_LOGBODY   0x0000000000040000   /* Log the body */
#define IP_FW_F_LOGFIRST  0x0000000000080000   /* Log ไบต์แรกถ้าสถานะยัง
                                                * เหมือนเดิม
                                                */
#define IP_FW_F_IIFNAME   0x0000000000100000   /*interface เข้ามีระบุโดยชื่อและยูนิต*/
#define IP_FW_F_OIFNAME   0x0000000000200000   /* interface ออกมีระบุโดยชื่อ
                                                * และยูนิต */
#define IP_FW_F_INVSRC    0x0000000000400000   /* ตรวจสอบต้นทางที่ตรงกันข้าม */
#define IP_FW_F_INVDST    0x0000000000800000   /* ตรวจสอบปลายทางที่ตรงกันข้าม */
#define IP_FW_F_ICMPBIT   0x0000000001000000   /* ชนิดของ ICMP */
#define IP_FW_F_FAKEICMP  0x0000000002000000   /* ส่งแพ็คเก็ต ICMP unreachable
                                                * with fake source
                                                */
#define IP_FW_F_AUTH      0x0000000004000000   /* ใช้ authentication */
#define IP_FW_F_PREAUTH   0x0000000008000000   /* ร้องขอการทำ preauthentication */
#define IP_FW_F_UID       0x0000000010000000   /* กรองโดย uid */
#define IP_FW_F_GID       0x0000000020000000   /* กรองโดย gid */
#define IP_FW_F_KEEP_S    0x0000000040000000   /* เก็บสถานะ */
#define IP_FW_F_KEEP_F    0x0000000080000000   /* เก็บข้อมูลการ fragment */
#define IP_FW_F_CHECK_S   0x0000000100000000   /* ตรวจสอบสถานะ */
#define IP_FW_F_SME       0x0000000200000000   /* ต้นทาง = me */
#define IP_FW_F_DME       0x0000000400000000   /* ปลายทาง = me */
#define IP_FW_F_RND_MATCH       0x0000000800000000   /* ใช้ความน่าจะเป็นในการ
                                                      * เข้าคู่
                                                      */
#define IP_FW_F_FRAG      0x0000001000000000   /* Fragment */
#define IP_FW_F_FASTROUTE       0x0000002000000000   /* ส่งผ่าน normal routing */
#define IP_FW_F_CALLNOW   0x0000004000000000   /* เรียกใช้ฟังก์ชัน (fr_func) ถ้ามีการ
                                                * เข้าคู่ */
```

```
#define IP_FW_BRIDGED       0x0000008000000000   /* จะมีการเข้าคู่กับ bridged packets */
#define IP_FW_F_NOMATCH     0x0000010000000000
#define FR_LOGB             0x0000020000000000   /* Log Block */
#define FR_LOGP             0x0000040000000000   /* Log Pass */
#define FR_RETRST           0x0000080000000000   /* Return reset */

#define FR_LOGMASK          (IP_FW_F_LOG | FR_LOGP | FR_LOGB)
#define IP_FW_F_MASK        0xFFFFFFFFFFFFFFFF          /* ทุก flag ที่เป็นไปได้ */
/*
 * ความหมายของ FI_* และข้อมูลที่เก็บใน fw_flg
 */
#define FF_OPTIONS          0x01000000
#define FF_TCPUDP           0x02000000
#define FF_FRAG             0x04000000
#define FF_SHORT            0x08000000

#define FI_OPTIONS          (FF_OPTIONS >> 24)
#define FI_TCPUDP           (FF_TCPUDP >> 24)    /* TCP/UCP */
#define FI_FRAG             (FF_FRAG >> 24)
#define FI_SHORT            (FF_SHORT >> 24)
#define FI_CMP              (FI_OPTIONS|FI_TCPUDP|FI_SHORT)

/*
 * Flags สำหรับ 'fw_ipflg' เพื่อใช้เปรียบเทียบค่าของไอพี และโปรโตคอล
 * ไม่ได้มีการใช้ทั้งหมด ปัจจุบันใช้แค่ IP_FW_IF_*MSK ( IP_FW_IF_TCPEST )
 */
#define IP_FW_IF_TCPOPT     0x00000001    /* tcp options */
#define IP_FW_IF_TCPFLG     0x00000002    /* tcp flags */
#define IP_FW_IF_TCPSEQ     0x00000004    /* ลำดับของ tcp */
#define IP_FW_IF_TCPACK     0x00000008    /* หมายเลข tcp acknowledgement */
```

```
#define IP_FW_IF_TCPWIN     0x00000010    /* ขนาดของ tcp window */
#define IP_FW_IF_TCPEST     0x00000020    /* established TCP connection */
#define IP_FW_IF_TCPMSK     0x00000020    /* ทุกค่าของ tcp */

#define IP_FW_IF_IPOPT      0x00000100    /* ip options */
#define IP_FW_IF_IPLEN      0x00000200    /* ip length */
#define IP_FW_IF_IPID       0x00000400    /* ip identification */
#define IP_FW_IF_IPTOS      0x00000800    /* ip type of service */
#define IP_FW_IF_IPTTL      0x00001000    /* ip time to live */
#define IP_FW_IF_IPVER      0x00002000    /* ip version */
#define IP_FW_IF_IPMSK      0x00000000    /* ทุกค่าของ ip */
/*
 * ความหมายของรหัสตอบกลับการ REJECT
 * ค่าที่น้อยกว่า 256 จะตรงกับ ICMP unreachable codes.
 */
#define IP_FW_REJECT_RST    0x0100 /* TCP packets: send RST */
/*
 * ความหมายของ IP option
 */
#define IP_FW_IPOPT_LSRR    0x01
#define IP_FW_IPOPT_SSRR    0x02
#define IP_FW_IPOPT_RR      0x04
#define IP_FW_IPOPT_TS      0x08
/*
 * สำหรับการเปรียบเทียบพอร์ต และการระบุพอร์ตเป็นช่วง
 */
#define FW_NONE             0
#define FW_EQUAL            1
#define FW_NEQUAL           2
#define FW_LESST            3
```

```
#define FW_GREATERT        4
#define FW_LESSTE          5
#define FW_GREATERTE       6
#define FW_OUTRANGE        7
#define FW_INRANGE         8
#define FW_MASK            9
/*
 * ความหมายของ TCP option
 */
#define IP_FW_TCPOPT_MSS          0x01
#define IP_FW_TCPOPT_WINDOW       0x02
#define IP_FW_TCPOPT_SACK         0x04
#define IP_FW_TCPOPT_TS           0x08
#define IP_FW_TCPOPT_CC           0x10
/*
 * ความหมายของ TCP flags.
 */
#define IP_FW_TCPF_FIN            TH_FIN
#define IP_FW_TCPF_SYN            TH_SYN
#define IP_FW_TCPF_RST            TH_RST
#define IP_FW_TCPF_PSH            TH_PUSH
#define IP_FW_TCPF_ACK            TH_ACK
#define IP_FW_TCPF_URG            TH_URG

#ifdef _KERNEL
#define IP_FW_PORT_DYNT_FLAG      0x10000
#define IP_FW_PORT_TEE_FLAG       0x20000
#define IP_FW_PORT_DENY_FLAG      0x40000
#endif /* _KERNEL */
#endif /* _IP_FW_H */
```

3.2.2 การ map จากกฎของ IP Firewall และ IP Filter ให้อยู่ในโครงสร้างกลาง

ตารางที่ 3.2 การ map กฎของ IP Firewall และ IP Filter ให้เข้ากับโครงสร้างกลาง

IP Firewall	IP Filter	Internal Format
Prob *match_proability*		dont_match_prob = 1-*match_probility*;
allow , pass , permit , accept	**pass**	fw_flg |= IP_FW_F_ACCEPT;
deny ,drop	**block**	fw_flg |= IP_FW_F_DENY;
reject		fw_flg |= IP_FW_F_REJECT; fw_reject_code = ICMP_UNREACH_HOST;
unreach *code*	**block return-icmp** *return-code*	fw_flg |= IP_FW_F_REJECT; fw_reject_code = *code;*
	block return-icmp-as-dest *return-code*	fw_flg | = IP_FW_F_REJECT; fw_flg |= IP_FW_F_FAKEICMP; fw_reject_code = *code;*
reset	**block return-reset**	fw_flg |= IP_FW_F_REJECT; fw_reject_code = IP_FW_REJECT_RST;
count	**count**	fw_flg |= IP_FW_F_COUNT;
check-state		fw_flg |= IP_FW_F_CHECK_S;
divert *port*		fw_flg |= IP_FW_F_DIVERT; fw_divert_port = *port*;
tee *port*		fw_flg |= IP_FW_F_TEE; fw_divert_port = *port*;
fwd *ipaddr[,port]*		fw_flg |= IP_FW_F_FWD; fw_fwd_ip = *ipaddr[,port]*;
pipe *pipe_nr*		fw_flg |= IP_FW_F_PIPE; fw_divert_port = *pipe_nr*;
queue *queue_nr*		fw_flg |= IP_FW_F_QUEUE;

		fw_divert_port = *pipe_nr*;
skipto *number*	**skip** *number*	fw_flg |= IP_FW_F_SKIPTO; fw_skipto_rule = *number*;
	auth	fw_flg |= IP_FW_F_AUTH;
	preauth	fw_flg |= IP_FW_F_PREAUTH FR;
	call *function-name*	fw_flg |= IP_FW_F_CALLNOW; fw_func = *function-name*
Log		
log [logamount *number*]		fw_flg |= IP_FW_F_LOG; fw_logamount = *number*;
	body	fw_flg |= IP_FW_F_LOG; fw_flg |= FR_LOGBODY;
	first	fw_flg |= IP_FW_F_LOG; fw_flg |= IP_FW_F _LOGFIRST;
	or-block	fw_flg |= IP_FW_F_LOG; fwflg |= IP_FW_F _ LOGORBLOCK;
	level *loglevel*	fw_flg |= IP_FW_F _LOG; fw_loglevel = *loglevel*;
Protocol		
proto	**tcp/udp** **udp | tcp | icmp |** *proto*	fw_prot = *proto*; fw_flg |= FW_TCPUDP ; (สำหรับ tcp/udp)
in-out		
	in	fw_flg |= IP_FW_F_IN;
	out	fw_flg |= IP_FW_F_OUT;
src **and** *dst*		
src	*src*	fw_src; fw_smsk;
dst	*dst*	fw_dst;

		fw_dmsk;
ports	*ports*	fw_pts[IP_FW_MAX_PORTS]; fw_nports; สำหรับ port ใช้ fr_scmp ,fr_sport ,fr_stop (สำหรับพอร์ตต้นทาง) สำหรับ port ใช้ fr_dcmp ,fr_dport ,fr_dtop (สำหรับพอร์ตปลายทาง)
interface-spec	**on** *interface-name* **to** interface-name	fw_in_if; fw_out_if;
options		
	quick	quick = 1;
keep-state	**keep state**	fw_flg |= IP_FW_F_KEEP_S; dyn_type = DYN_KEEP_STATE;
	keep frags	fw_flg |= IP_FW_F_KEEP_F;
limit {**src-addr** | **src-port** | **dst-addr** | **dst-port**} *N*		fw_flg |= IP_FW_F_KEEP_S; dyn_type = DYN_LIMIT; limit_mask; (DYN_SRC_ADDR,DYN_SRC_PORT, DYN_DST_ADDR,DYN_DST,PORT conn_limit = *N*;
bridged		fw_flg |= IP_FW_BRIDGED;
frag		w_flg |= IP_FW_F_FRAG;
ipoptions *spec*		u_char fw_ipopt; u_char fw_ipnopt; (IP_FW_IPOPT_SSRR,IP_FW_IPOPT_LSRR, IP_FW_IPOPT_RR,IP_FW_IPOPT_TS)
tcpoptions *spec*		u_char fw_tcpopt; u_char fw_tcpnopt; (IP_FW_TCPOPT_MSS,IP_FW_TCPOPT_WINDOW ,IP_FW_TCPOPT_SACK ,IP_FW_TCPOPT_TS,

		IP_FW_TCPOPT_CC)
established		fw_ipflg |= IP_FW_IF_TCPEST;
setup		fw_tcpf |= IP_FW_TCPF_SYN; fw_tcpnf |= IP_FW_TCPF_ACK;
tcpflags *spec*	**flags**	fw_tcpf; fw_tcpnf; (IP_FW_TCPF_SYN, IP_FW_TCPF_FIN, IP_FW_TCPF_ACK, IP_FW_TCPF_PSH, IP_FW_TCPF_RST , IP_FW_TCPF_URG)
icmptypes *types*	**icmp-type**	fw_uar.fw_icmptypes; fw_flg |= IP_FW_F_ICMPBIT;
uid *user*		fw_flg |= IP_FW_F_UID; fw_uid = *user*;
gid *group*	**group** *number*	fw_flg |= IP_FW_F_GID; fw_gid = *group*;
	head *number*	fw_grhead = number;
	dup-to interface- name[":"ipaddr]	fw_flg |= IP_FW_F_DUP; fw_dup_if;
	fastroute	fw_flg |= IP_FW_F_FASTROUTE;
	tos *number*	fw_tos = *number*;
	ttl *number*	fw_ttl = *number*;
	with option	fw_fl |= oflags; fw_optmsk |= opts; fw_secmsk |= secmsk;

3.3 อัลกอริทึมที่ใช้ในการวิเคราห์ และลดรูปกฎของไฟร์วอลล์

การลดรูปโดยใช้หลักการของการตรวจสอบกฎที่เป็นสมาชิกกัน การลดรูปแบบนี้จะต้องมีการตรวจสอบว่าหมายเลขไอพีหนึ่งเป็นสมาชิกของอีกหมายเลขไอพีหนึ่งหรือไม่ เมื่อการตรวจสอบได้แล้ว จะต้องมีการตรวจสอบถึงกลุ่มของ action ลำดับก่อนหลัง และ options ว่าเป็น quick หรือไม่

เงื่อนไขที่จะสามารถลดรูปกฎได้มี 2 เงื่อนไข ดังนี้

เงื่อนไขที่ 1 หมายเลขไอพีของกฎที่ 2 เป็นสมาชิกของหมายเลขไอพีของกฎข้อที่ 1 และ กฎข้อที่ 1 มี options เป็น quick จึงสามารถลบกฎข้อที่ 2 ได้

กฎข้อที่ 1 A.B.C.D (มี option quick)

กฎข้อที่ 2 a .b .c .d

โดยที่ a .b .c .d เป็นสมาชิกของ A.B.C.D

เงื่อนไขที่ 2 หมายเลขไอพีของกฎที่ 1 เป็นสมาชิกของหมายเลขไอพีของกฎข้อที่ 2 แต่ options ของกฎทั้งสองกฎไม่เป็น quick จึงสามารถลบกฎข้อที่ 1 ได้

กฎข้อที่ 1 a .b .c .d (ไม่มี option quick)

กฎข้อที่ 2 A.B.C.D

โดยที่ a .b .c .d เป็นสมาชิกของ A.B.C.D

3.3.1 ฟังก์ชันที่ใช้ในการวิเคราะห์ และลดรูปกฎที่ไม่อาจไปถึง และกฎที่ขัดแย้งกันมี ดังต่อไปนี้

ฟังก์ชันใช้ในการตรวจสอบว่า Address1 เป็นสมาชิกของ Address2 หรือไม่

Input คือ IP address ตัวแรก กับ Netmask ของตัวแรก และ
IP address ตัวที่สอง กับ Netmask ของตัวที่สอง

Output คือ true จะหมายถึง IP ตัวแรกเป็นสมาชิกของ IP ตัวที่สอง
false จะหมายถึง IP ตัวแรกไม่เป็นสมาชิกของ IP ตัวที่สอง

```
boolean  IsMember (Addr1 , Mask1 , Addr2 , Mask2)
{
        if ( ( (Addr1&Mask1)&Mask2 ) = =  (Addr2&Mask2) )
                return true;
        else
                return false;
} //end function
```

ฟังก์ชันใช้ในการตรวจสอบว่า Address1 กับ Address2 เหมือนกันหรือไม่

Input คือ IP address ตัวแรก กับ Netmask ของตัวแรก และ
IP address ตัวที่สอง กับ Netmask ของตัวที่สอง

Output คือ true จะหมายถึง IP ตัวแรกและ IP ตัวที่สองเท่ากัน
false จะหมายถึง IP ตัวแรกและ IP ตัวที่สองเท่ากัน

```
boolean  IsEqual (Addr1 , Mask1 , Addr2 , Mask2)
{
        if ( IsMember (Addr1 , Mask1 , Addr2 , Mask2)
          && IsMember (Addr2 , Mask2 , Addr1 , Mask1) )
                return true;
        else
                return false;
}//end function
```

ฟังก์ชันใช้ในการตรวจสอบ fields อื่นๆนอกจาก address และ action เหมือนกันหรือไม่

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

Output คือ true จะหมายถึง fields อื่นเท่ากัน
false จะหมายถึง fields อื่นไม่เท่ากัน

```
boolean  IsSameField (Rule1 , Rule2)
{
        if ( ( (Rule1.fw_flg | IP_FW_F_COMMAND) ^
        (Rule2.fw_flg | IP_FW_F_COMMAND) ) == !IP_FW_F_MASK){
                if(
                (Rule1.dont_match_prob == Rule1.dont_match_prob)
                /* Action zone */
                && (Rule1.fw_divert_port  == Rule2.fw_divert_port)
                && (Rule1.fw_skipto_Rule  == Rule2.fw_skipto_Rule)
                && (Rule1.fw_reject_code  == Rule2.fw_reject_code)
                && (Rule1.fw_fwd_ip  == Rule2.fw_fwd_ip)
                && (Rule1.fw_pipe_nr  == Rule2.fw_pipe_nr)
```

```
&& (Rule1.fw_logamount  == Rule2.fw_logamount)
&& (Rule1.fw_loghighest  == Rule2.fw_loghighest)
&& (Rule1.fw_prot == Rule2.fw_prot)
&& (Rule1.fw_uid == Rule2.fw_uid)
&& (Rule1.fw_gid == Rule2.fw_gid)
&& ( (Rule1.dyn_type == Rule2.dyn_type)
        && (Rule1.conn_limit == Rule2.conn_limit)
        && (Rule1.limit_mask == Rule2.limit_mask) )
&& (Rule1.fw_in_if == Rule2.fw_in_if)
&& (Rule1.fw_out_if == Rule2.fw_out_if)
&& ((Rule1.fw_ipopt == Rule2.fw_ipopt) &&
        (Rule1.fw_ipnopt == Rule2.fw_ipnopt) )
&& ( (Rule1.fw_tcpf == Rule2.fw_tcpf) &&
        (Rule1.fw_tcpnf == Rule2.fw_tcpnf) )
&& ( (Rule1.fw_tcpopt == Rule2.fw_tcpopt) &&
        (Rule1.fw_tcpnopt == Rule2.fw_tcpnopt) )
&& (Rule1.fw_ipflg == Rule2.fw_ipflg)
&& (Rule1.fw_mipflg == Rule2.fw_mipflg)
/* Additional for ipfilter */
&& (Rule1.fr_loglevel == Rule2.fr_loglevel)
&& (Rule1.fr_icmp == Rule2.fr_icmp)
&& (Rule1.fr_icmpm == Rule2.fr_icmpm)
&& (Rule1.fw_dup_if == Rule2.fw_dup_if)
&& (Rule1.fw_to_if == Rule2.fw_to_if)
&& (Rule1.fr_tos == Rule2.fr_tos)
&& (Rule1.fr_ttl == Rule2.fr_ttl)
&& (Rule1.fr_fl == Rule2.fr_fl)
&& (Rule1.fr_group == Rule2.fr_group)
&& (Rule1.fr_grhead == Rule2.fr_grhead)
&& (Rule1.fr_optmsk == Rule2.fr_optmsk)
```

```
                && (Rule1.fr_secmsk == Rule2.fr_secmsk) ){
                        if(Rule1.fw_prot == IPPROTO_ICMP){
                                if(Rule1.fw_uar.fw_icmptypes ==
                                Rule2.fw_uar.fw_icmptypes)
                                        return true;
                                else
                                        return false;
                        }//end if protocol is ICMP
                        else
                                return true;
                }
                else
                        return false;
        }//end if flag is same
        else
                /* if flag is not same */
                return false;
}//end function
```

ฟังก์ชันใช้ในการระบุว่ากฎข้อนี้ได้ถูกลบออกไป

Input คือ กฎของไฟร์วอลล์

```
void reduce (Rule) {
        Rule.valid = 0;
        reduced = 1;
}
```

ฟังก์ชันใช้ในการหา Netmask รวมของ Address1 และ Address2

Input คือ IP address ตัวแรก , IP address ตัวที่สอง , Netmask ของทั้ง address

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

Output คือ Netmask ใหม่ของ address ทั้งสองตัว ถ้าไม่สามารถหาได้หรือ Address1 กับ Address2 เป็น address เดียวกัน จะคืนค่า Netmask เป็น 255.255.255.255

```
Address summarized (Address1, Address, Netmask)
{
        if( (Address1 & Netmask) = = (Address2  & Netmask) )
                /* If it same address return 255.255.255.255 */
                return  255.255.255.255;
        temp  = ( (Address1 & Netmask) ^ (Address2  & Netmask) );
        Newmask = ( (~temp) & Netmask);
        /* Check new mask is mask */
        if(is_mask(Newmask))
                return Newmask;
        else
                /* If it not mask return 255.255.255.255 */
                return 255.255.255.255;
}//end function summarized
```

ฟังก์ชันใช้ในการลดรูป subnetted rules โดยจะตรวจสอบจากหมายเลขไอพีต้นทาง

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

```
void reduceSrcSubnetted (Rule1 , Rule2)
{
        /* Reduce by subnet */
        if(Rule1.fw_smsk == Rule2.fw_smsk){
                new_mask = summarized (Rule1.fw_src, Rule2.fw_src, Rule1.fw_smsk);
                if( (new_mask != "255.255.255.255")
                && IsEqual (Rule1.fw_dst, Rule1.fw_dmsk,Rule2.fw_dst,Rule2.fw_dmsk)
                && IsSamePort(Rule1, Rule2)&& IsSameAction(Rule1, Rule2)
                && IsSameField (Rule1, Rule2) ){
                        Rule1.fw_src = Rule1.fw_src & new_mask;
```

```
                Rule1.fw_smsk = new_mask;
                reduce (Rule2);
            }
            else if( (new_mask != "255.255.255.255")
            && IsEqual (Rule2.fw_dst, Rule2.fw_dmsk,Rule1.fw_dst,Rule1.fw_dmsk)
            && IsSamePort(Rule2, Rule1)&& IsSameAction(Rule2, Rule1)
            && IsSameField (Rule2, Rule1) ){
                Rule2.fw_src = Rule2.fw_src & new_mask;
                Rule2.fw_smsk = new_mask;
                reduce (Rule1);
            }
        }
}//end function
```

ฟังก์ชันใช้ในการลดรูป subnetted rules โดยจะตรวจสอบจากหมายเลขไอพีปลายทาง

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

```
void reduceDstSubnetted (Rule1 , Rule2)
{
        /* Reduce by subnet */
        if(Rule1.fw_dmsk == Rule2.fw_dmsk){
            new_mask = summarized (Rule1.fw_dst, Rule2.fw_dst, Rule1.fw_dmsk);
            if( (new_mask != "255.255.255.255")
            && IsEqual (Rule1.fw_src, Rule1.fw_smsk,Rule2.fw_src,Rule2.fw_smsk)
            && IsSamePort(Rule1, Rule2)&& IsSameAction(Rule1, Rule2)
            && IsSameField (Rule1, Rule2) ){
                Rule1.fw_dst = Rule1.fw_dst & new_mask;
                Rule1.fw_dmsk = new_mask;
                reduce (Rule2);
            }
            else if( (new_mask != "255.255.255.255")
```

```
            && IsEqual (Rule2.fw_src, Rule2.fw_smsk,Rule1.fw_src,Rule1.fw_smsk)
            && IsSamePort(Rule2, Rule1)&& IsSameAction(Rule2, Rule1)
            && IsSameField (Rule2, Rule1) ){
                Rule2.fw_dst = Rule2.fw_dst & new_mask;
                Rule2.fw_dmsk = new_mask;
                reduce (Rule1);
            }
        }
}//end function
```

ฟังก์ชันใช้ในการลดรูป subnetted rules

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

```
void reduceSubnettedRule (Rule1 , Rule2)
{
reduced = 1;
        while ( reduced != 0 ){
            reduced = 0;
            for(i = 0 ; i < Number of Rules; i++){
                if(Rule[i].valid == 1){
                    for(j = i+1 ; j < Number of Rules; j++){
                        if(i != j && Rule[i].valid = = 1 && Rule[j].valid == 1)
                            reduceSrcSubnetted(Rule[i],Rule[j]);
                        if(i != j && Rule[i].valid = = 1 && Rule[j].valid == 1)
                            reduceDstSubnetted (Rule[i],Rule[j])
                    }//end for j
                }//end if rule[i] is valid
            }//end for i
        }//end while
}//end function
```

ฟังก์ชันใช้ในการลดรูปกฎแบบ unreachable rules โดยจะตรวจสอบจากหมายเลขไอพีต้นทาง

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

```
void reduceSrcUnreach (Rule1 , Rule2)
{
        if ( IsMember(Rule1.fw_src, Rule1.fw_smsk, Rule2.fw_src,Rule2. fw_smsk)
        && IsEqual(Rule1.fw_dst, Rule1.fw_dmsk, Rule2.fw_dst, Rule2. fw_dmsk)
        && IsSameField (Rule1 , Rule2) && IsPortMember (Rule1, Rule2) ){
                if ( (Rule1.quick = = true)&&(!IsConflict(Rule1.action,Rule2.action) )
                        reduce (Rule2);
        }
        else if ( IsMember(Rule2.fw_src, Rule2.fw_smsk, Rule1.fw_src,Rule1. fw_smsk)
        && IsEqual(Rule2.fw_dst, Rule2.fw_dmsk, Rule1.fw_dst, Rule1. fw_dmsk)
        && IsSameField (Rule2 , Rule1) && IsPortMember (Rule2, Rule1) ){
                if ( (Rule1.quick = = false)&&(!IsConflict(Rule1.action,Rule2.action) )
                        reduce (Rule1);
        }
}//end function
```

ฟังก์ชันใช้ในการลดรูปกฎแบบ unreachable rules โดยจะตรวจสอบจากหมายเลขไอพีปลายทาง

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

```
void reduceDstUnreach (Rule1 , Rule2)
{
        if ( IsMember(Rule1.fw_dst, Rule1.fw_dmsk, Rule2.fw_dst,Rule2. fw_dmsk)
        && IsEqual(Rule1.fw_src, Rule1.fw_smsk, Rule2.fw_src, Rule2. fw_smsk)
        && IsSameField (Rule1 , Rule2) && IsPortMember (Rule1, Rule2) ){
                if ( (Rule1.quick = = true)&&(!IsConflict(Rule1.action,Rule2.action) )
                        reduce (Rule2);
        }
        else if ( IsMember(Rule2.fw_dst, Rule2.fw_dmsk, Rule1.fw_dst,Rule1. fw_msk)
        && IsEqual(Rule2.fw_src, Rule2.fw_smsk, Rule1.fw_src, Rule1. fw_smsk)
```

```
        && IsSameField (Rule2 , Rule1) && IsPortMember (Rule2, Rule1) ){
                if ( (Rule1.quick = = false)&&(!IsConflict(Rule1.action,Rule2.action) )
                        reduce (Rule1);
        }
}//end function
```

ฟังก์ชันใช้ในการลดรูปกฎแบบ conflict rules โดยจะตรวจสอบจากหมายเลขไอพีต้นทาง

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

```
void reduceSrcConflict (Rule1 , Rule2)
{
        if ( IsMember(Rule1.fw_src, Rule1.fw_smsk, Rule2.fw_src,Rule2. fw_smsk)
        && IsEqual(Rule1.fw_dst, Rule1.fw_dmsk, Rule2.fw_dst, Rule2. fw_dmsk)
        && IsSameField (Rule1 , Rule2) && IsPortMember (Rule1, Rule2) ){
                if ( (Rule1.quick = = true)&&(IsConflict(Rule1.action,Rule2.action) )
                        reduce (Rule2);
        }
        else if ( IsMember(Rule2.fw_src, Rule2.fw_smsk, Rule1.fw_src,Rule1. fw_smsk)
        && IsEqual(Rule2.fw_dst, Rule2.fw_dmsk, Rule1.fw_dst, Rule1. fw_dmsk)
        && IsSameField (Rule2 , Rule1) && IsPortMember (Rule2, Rule1) ){
                if ( (Rule1.quick = = false)&&(IsConflict(Rule1.action,Rule2.action) )
                        reduce (Rule1);
        }
}//end function
```

ฟังก์ชันใช้ในการลดรูปกฎแบบ unreachable rules โดยจะตรวจสอบจากหมายเลขไอพีปลายทาง

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

```
void reduceDstConflict (Rule1 , Rule2)
{
        if ( IsMember(Rule1.fw_dst, Rule1.fw_dmsk, Rule2.fw_dst,Rule2. fw_dmsk)
        && IsEqual(Rule1.fw_src, Rule1.fw_smsk, Rule2.fw_src, Rule2. fw_smsk)
```

```
        && IsSameField (Rule1 , Rule2) && IsPortMember (Rule1, Rule2) ){
                if ( (Rule1.quick = = true)&&(IsConflict(Rule1.action,Rule2.action) )
                        reduce (Rule2);
        }
        else if ( IsMember(Rule2.fw_dst, Rule2.fw_dmsk, Rule1.fw_dst,Rule1. fw_msk)
        && IsEqual(Rule2.fw_src, Rule2.fw_smsk, Rule1.fw_src, Rule1. fw_smsk)
        && IsSameField (Rule2 , Rule1) && IsPortMember (Rule2, Rule1) ){
                if ( (Rule1.quick = = false)&&(IsConflict(Rule1.action,Rule2.action) )
                        reduce (Rule1);
        }
}//end function
```

ฟังก์ชันใช้ในการลดรูปกฎ โดยจะมีการวนลูปเป็นลักษณะ bubble

Input คือ กฎของไฟร์วอลล์ตัวแรก และกฎของไฟร์วอลล์ ตัวที่สอง

```
void reduceUnreachConflict ( Set of Firewall's Rules R)
{
reduced = 1;
        while ( reduced != 0 ){
                reduced = 0;
                for(i = 0 ; i < Number of Rules; i++){
                        if(Rule[i].valid == 1){
                                for(j = i+1 ; j < Number of Rules; j++){
                                        if(i != j && Rule[i].valid = = 1 && Rule[j].valid == 1){
                                                reduceSrcUnreach(Rule[i],Rule[j]);
                                                reduceSrcConflict(Rule[i],Rule[j]);
                                        }
                                        if(i != j && Rule[i].valid = = 1 && Rule[j].valid == 1){
                                                reduceDstUnreach(Rule[i],Rule[j])
                                                reduceDstConflict(Rule[i],Rule[j]);
                                        }
```

```
                    }//end for j
                }//end if rule[i] is valid
            }//end for i
        }//end while
}// end function
```

ฟังก์ชันการทำงานหลักของโปรแกรม

```
int main(int argc, char *argv[])
{
        FILE = open_file(file_name);
        while(!End Of File(FILE)){
                line = getline(FILE);
                temp_rule = convert_rule_to_internal(line);
                fw_rule = add_to_array (temp_rule);
        }
        reduceSubnettedRule(fw_rule);
        reduceUnreachConflict(fw_rule);
        close(FILE);
}
```

บทที่ 4

การพัฒนาระบบงานและผลการทดลอง

4.1 ภาษา และเครื่องมือที่ใช้ในการพัฒนาระบบงาน

เนื้อหาในส่วนนี้จะกล่าวถึงภาษา และเครื่องมือที่จะใช้ในการพัฒนาระบบงาน ซึ่งมีดังต่อไปนี้

- ระบบปฏิบัติการ FreeBSD 4.7 พร้อมทั้งโปรแกรม IP Firewall และ IP Filter
- ภาษาซี โปรแกรมเครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์ใช้ภาษาซีในการพัฒนา
- gcc compiler : ใช้ในการ compiler โค้ดของโปรแกรม
- Text editor : ใช้ในการเขียนโค้ดของโปรแกรม
- Microsoft Word : ใช้ในการทำเอกสารประกอบโครงงาน
- Microsoft Visio : ใช้ในการทำ Dataflow Diagram
- Adobe Photoshop : ใช้ในการทำรูปภาพประกอบของเอกสารประกอบโครงงาน

4.2 การพัฒนาระบบ

การพัฒนาระบบจะใช้ภาษาซีในการพัฒนา โดยการเก็บกฎของไฟร์วอลล์จะเป็นแบบ static memory allocation คือ จะมีการใช้ array ในการจัดเก็บกฎของไฟร์วอลล์ โดยจะมีการประกาศ array ไว้จำนวน 65,536 element ซึ่งเป็นจำนวนกฎที่มากที่สุดของไฟร์วอลล์

4.3 ผลการทดลอง

เนื้อหาในส่วนนี้จะกล่าวถึงการทดสอบการทำงานของโปรแกรมเครื่องมือสำหรับวิเคราะห์และลดจำนวนกฎของไฟร์วอลล์ โดยการทดสอบระบบงานของโครงงานนี้จะทำบนระบบปฏิบัติการ FreeBSD 4.7 โดยจะข้อมูลที่จะนำมาทดสอบจะแสดงให้เห็นถึงการลดรูปในรูปแบบต่างๆ ดังต่อไปนี้

4.3.1 การลดรูปจากกฎที่เป็น subnet

ตัวอย่างที่ 1 ข้อมูลจะเป็นกฎที่เป็น subnet กันจำนวน 8 กฎโดยแต่ละกฎจะมี mask จำนวน 24 บิต ผลลัพธ์ก็คือ จะเหลือกฎเพียง 1 กฎซึ่งจะมี mask จำนวน 21 บิต

IP Firewall

Input	deny tcp from 203.154.0.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.1.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.2.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.3.0/24 to 203.154.12.2/16 keep-state deny tcp from 203.154.4.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.5.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.6.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.7.0/24 to 203.154.12.2/23 keep-state
output	deny tcp from 203.154.1.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.2.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.3.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.4.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.5.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.6.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.7.0/24 to 203.154.12.0/23 keep-state is subneted rule. ---------- Reduced rules ---------- deny tcp from 203.154.0.0/21 to 203.154.12.0/23 keep-state

IP Filter

Input	block in proto tcp from 203.154.0.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.1.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.2.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.3.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.4.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.5.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.6.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.7.0/24 to 203.154.12.2/23
output	block in proto tcp from 203.154.1.0/24 to 203.154.12.0/23 is subneted rule. block in proto tcp from 203.154.2.0/24 to 203.154.12.0/23 is subneted rule. block in proto tcp from 203.154.3.0/24 to 203.154.12.0/23 is subneted rule. block in proto tcp from 203.154.4.0/24 to 203.154.12.0/23 is subneted rule. block in proto tcp from 203.154.5.0/24 to 203.154.12.0/23 is subneted rule. block in proto tcp from 203.154.6.0/24 to 203.154.12.0/23 is subneted rule. block in proto tcp from 203.154.7.0/24 to 203.154.12.0/23 is subneted rule. ---------- Reduced rules ---------- block in proto tcp from 203.154.0.0/21 to 203.154.12.0/23

ตัวอย่างที่ 2 ข้อมูลจะเป็นกฎจำนวน 7 กฎ โดยแต่ละกฎจะมี mask จำนวน 24 บิต ซึ่งจะคล้ายกับตัวอย่างที่ 1 แต่จะมีการนำกฎที่มีหมายเลขไอพี 203.154.3.0/24 ออก ผลลัพธ์ก็คือ จะสามารถลดรูปได้โดยรวม subnet ให้เหลือ 3 subnet

IP Firewall

Input	deny tcp from 203.154.0.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.1.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.2.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.4.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.5.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.6.0/24 to 203.154.12.2/23 keep-state deny tcp from 203.154.7.0/24 to 203.154.12.2/23 keep-state
output	deny tcp from 203.154.1.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.5.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.6.0/24 to 203.154.12.0/23 keep-state is subneted rule. deny tcp from 203.154.7.0/24 to 203.154.12.0/23 keep-state is subneted rule. ---------- Reduced rules ---------- deny tcp from 203.154.0.0/23 to 203.154.12.0/23 keep-state deny tcp from 203.154.2.0/24 to 203.154.12.0/23 keep-state deny tcp from 203.154.4.0/22 to 203.154.12.0/23 keep-state

IP Filter

Input	block in proto tcp from 203.154.0.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.1.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.2.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.4.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.5.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.6.0/24 to 203.154.12.2/23 block in proto tcp from 203.154.7.0/24 to 203.154.12.2/23
output	block in proto tcp from 203.154.1.0/24 to 203.154.12.0/23

	is subneted rule. block in proto tcp from 203.154.5.0/24 to 203.154.12.0/23 is subneted rule. block in proto tcp from 203.154.6.0/24 to 203.154.12.0/23 is subneted rule. block in proto tcp from 203.154.7.0/24 to 203.154.12.0/23 is subneted rule. ---------- Reduced rules ---------- block in proto tcp from 203.154.0.0/23 to 203.154.12.0/23 block in proto tcp from 203.154.2.0/24 to 203.154.12.0/23 block in proto tcp from 203.154.4.0/22 to 203.154.12.0/23

4.3.2 การลดรูปจากกฎที่ไม่สามารถไปถึง

ตัวอย่างที่ 3 เป็นกฎของ IP Firewall จะเห็นว่าหมายเลขไอพี และพอร์ตของกฎข้อที่ 4 จะเป็นสมาชิกของกฎข้อที่ 1 ซึ่งกฎของ IP Firewall เปรียบเสมือนว่ามี option เป็น quick ทั้งหมดจะเป็นการเข้าเงื่อนไขที่ 1 และเมื่อตรวจสอบ action ของทั้งสองปรากฏว่ามีความสอดคล้องกัน ดังนั้นจะได้ผลว่ากฎข้อที่4 จะเป็นกฎไม่สามารถไปถึง จึงลบกฎข้อนี้ออกไป

IP Firewall

Input	deny udp from 161.56.24.52/16 10-100 to 161.246.49.50/32 allow udp from any to any ipoptions ssrr allow icmp from any to any icmptypes 3,9 deny udp from 161.56.24.52/24 30,40 to 161.246.49.50/32
output	deny udp from 161.56.24.0/24 30,40 to 161.246.49.50 is unreachabled rule. ---------- Reduced rules ---------- deny udp from 161.56.0.0/16 10-100 to any allow udp from any to any ipopt ssrr allow icmp from any to any icmptype 3,9

ตัวอย่างที่ 4 เป็นกฎของ IP Filter จะเห็นว่าหมายเลขไอพี และพอร์ตของกฎข้อที่ 4 จะเป็นสมาชิกของกฎข้อที่ 3 ซึ่งกฎข้อที่ 3 มี option เป็น quick จะเป็นการเข้าเงื่อนไขที่ 1 และเมื่อตรวจสอบ action ของทั้งสองปรากฏว่ามีความสอดคล้องกัน ดังนั้นจะได้ผลว่ากฎข้อที่ 4 จะเป็นกฎไม่สามารถไปถึง จึงลบกฎข้อนี้ออกไป

IP Filter

Input	pass in proto tcp from any to any block in proto icmp from any to any block in quick proto udp from 161.56.24.52/16 port > 5000 to any block in proto udp from 161.56.24.52/24 port = 2000 to 161.246.49.50/32
output	block in proto udp from 161.56.24.0/24 to 161.246.49.50/32 is unreachabled rule. ---------- Reduced rules ---------- pass in proto tcp from any to any block in proto icmp from any to any block in proto udp from 161.56.0.0/16 to any

ตัวอย่างที่ 5 เป็นกฎของ IP Filter จะเห็นว่าหมายเลขไอพีของกฎข้อที่ 3 จะเป็นสมาชิกของกฎข้อที่ 4 ซึ่งกฎทั้งสองข้อนี้ไม่มี option ที่เป็น quick จะเป็นการเข้าเงื่อนไขที่ 2 และเมื่อตรวจสอบ action ของทั้งสองปรากฏว่ามีความสอดคล้องกัน ดังนั้นจะได้ผลว่ากฎข้อที่ 3 จะเป็นกฎไม่สามารถไปถึง จึงลบกฎข้อนี้ออกไป

IP Filter

Input	pass in proto tcp from any to any block in proto icmp from any to any block in proto udp from 161.56.24.52/24 to 161.246.49.50/32 block in proto udp from 161.56.24.52/16 to any
output	block in proto udp from 161.56.24.0/24 to 161.246.49.50/32 is unreachabled rule. ---------- Reduced rules ----------

	pass in proto tcp from any to any block in proto icmp from any to any block in proto udp from 161.56.0.0/16 to any

ตัวอย่างที่ 6 เป็นกฎของ IP Filter จะเห็นว่าหมายเลขไอพีของกฎข้อที่ 3 จะเป็นสมาชิกของกฎข้อที่ 4 ซึ่งกฎข้อที่ 3 มี option เป็น quick ซึ่งไม่มีการเข้ากับเงื่อนไขข้อใด จึงไม่สามารถลดรูปกฎได้

IP Filter

Input	pass in proto tcp from any to any block in proto icmp from any to any block in quick proto udp from 161.56.24.52/24 to 161.246.49.50/32 block in proto udp from 161.56.24.52/16 to any
output	---------- Reduced rules ---------- pass in proto tcp from any to any block in proto icmp from any to any block in quick proto udp from 161.56.24.52/24 to 161.246.49.50/32 block in proto udp from 161.56.24.52/16 to any

4.3.3 การลดรูปจากกฎที่ขัดแย้งกัน

ตัวอย่างที่ 7 เป็นกฎของ IP Firewall จะเห็นว่าหมายเลขไอพีของกฎข้อที่ 4 จะเป็นสมาชิกของกฎข้อที่ 1 ซึ่งกฎของ IP Firewall เปรียบเสมือนว่ามี option เป็น quick ทั้งหมดจะเป็นการเข้าเงื่อนไขที่ 1 และเมื่อตรวจสอบ action ของทั้งสองปรากฏว่ามีความขัดแย้งกัน ดังนั้นจะได้ผลว่ากฎข้อที่4 ขัดแย้งกับกฎข้อที่ 1 จึงลบกฎข้อที่ 4 ออกไป

IP Firewall

Input	deny udp from 161.56.24.52/16 to 161.246.49.50/32 allow udp from any to any ipoptions ssrr allow icmp from any to any icmptypes 3,9 deny udp from 161.56.24.52/24 to 161.246.49.50/32
output	deny udp from 161.56.24.0/24 to 161.246.49.50

	is conflict with rule deny udp from 161.56.24.52/16 to 161.246.49.50/32 ---------- Reduced rules ---------- deny udp from 161.56.0.0/16 to any allow udp from any to any ipopt ssrr allow icmp from any to any icmptype 3,9

ตัวอย่างที่ 8 เป็นกฎของ IP Filter จะเห็นว่าหมายเลขไอพีของกฎข้อที่ 4 จะเป็นสมาชิกของกฎข้อที่ 3 ซึ่งกฎข้อที่ 3 มี option เป็น quick จะเป็นการเข้าเงื่อนไขที่ 1 และเมื่อตรวจสอบ action ของทั้งสองปรากฏว่ามีความขัดแย้งกัน ดังนั้นจะได้ผลว่ากฎข้อที่ 4 ขัดแย้งกับกฎข้อที่ 3 จึงลบกฎข้อที่ 4 ออกไป

IP Filter

Input	pass in proto tcp from any to any block in proto icmp from any to any block in quick proto udp from 161.56.24.52/16 to any block in proto udp from 161.56.24.52/24 to 161.246.49.50/32
output	block in proto udp from 161.56.24.0/24 to 161.246.49.50/32 is conflict with rule block in quick proto udp from 161.56.24.52/16 to any ---------- Reduced rules ---------- pass in proto tcp from any to any block in proto icmp from any to any block in proto udp from 161.56.0.0/16 to any

ตัวอย่างที่ 9 เป็นกฎของ IP Filter จะเห็นว่าหมายเลขไอพีของกฎข้อที่ 3 จะเป็นสมาชิกของกฎข้อที่ 4 ซึ่งกฎทั้งสองข้อนี้ไม่มี option ที่เป็น quick จะเป็นการเข้าเงื่อนไขที่ 2 และเมื่อตรวจสอบ action ของทั้งสองปรากฏว่ามีความขัดแย้งกัน ดังนั้นจะได้ผลว่ากฎข้อที่ 3 ขัดแย้งกับกฎข้อที่ 4 จึงลบกฎข้อที่ 3 ออกไป

IP Filter

Input	pass in proto tcp from any to any block in proto icmp from any to any block in proto udp from 161.56.24.52/24 to 161.246.49.50/32 block in proto udp from 161.56.24.52/16 to any
output	block in proto udp from 161.56.24.0/24 to 161.246.49.50/32 is conflict with rule block in proto udp from 161.56.24.52/16 to any ---------- Reduced rules ---------- pass in proto tcp from any to any block in proto icmp from any to any block in proto udp from 161.56.0.0/16 to any

ตัวอย่างที่ 10 เป็นกฎของ IP Filter จะเห็นว่าหมายเลขไอพีของกฎข้อที่ 3 จะเป็นสมาชิกของกฎข้อที่ 4 ซึ่งกฎข้อที่ 3 มี option เป็น quick ซึ่งไม่มีการเข้ากับเงื่อนไขข้อใด จึงไม่สามารถลดรูปกฎได้

IP Filter

Input	pass in proto tcp from any to any block in proto icmp from any to any block in quick proto udp from 161.56.24.52/24 to 161.246.49.50/32 block in proto udp from 161.56.24.52/16 to any
output	---------- Reduced rules ---------- pass in proto tcp from any to any block in proto icmp from any to any block in quick proto udp from 161.56.24.52/24 to 161.246.49.50/32 block in proto udp from 161.56.24.52/16 to any

บทที่ 5
สรุปผลการพัฒนาโปรแกรม

เนื้อหาในบทนี้จะสรุปผลของการพัฒนาโปรแกรมเครื่องมือสำหรับวิเคราะห์ และลดจำนวนกฎของไฟร์วอลล์ พร้อมทั้งข้อเสนอแนะสำหรับผู้ที่สนใจจะนำโปรแกรมไปทำการพัฒนาต่อไป

5.1 สรุปผลการพัฒนาโปรแกรม

ไฟร์วอลล์เป็นอุปกรณ์ที่ใช้ในการควบคุมการเข้า-ออกของข้อมูลระหว่างระบบเครือข่ายภายในองค์กร กับระบบเครือข่ายภายนอกองค์กร หรืออินเทอร์เน็ต ซึ่งจุดประสงค์ของไฟร์วอลล์นั้นเป็นไปเพื่อป้องกันการบุกรุกที่มีจุดประสงค์ร้ายจากระบบเครือข่ายภายนอกองค์กร อันจะก่อให้เกิดความเสียหายต่อระบบเครือข่ายภายในองค์กรได้ โปรแกรมไฟร์วอลล์นั้นมีอยู่มากมาย แต่ในโครงงานนี้จะมุ่งความสนในไปที่ IP Firewall และ IP Filter ซึ่งการทำงานของไฟร์วอลล์จะใช้กฎเป็นตัวควบคุมข้อมูลที่เข้า-ออก ถ้ากฎมีความซับซ้อนมากจะทำให้ประสิทธิในการทำงานต่ำ และผู้ดูแลระบบจะดูแลตรวจสอบยาก จึงได้มีแนวความคิดที่จะลดจำนวนของกฎต่างๆ โดยเงื่อนไขในการลดจำนวนกฎจะพิจารณาจาก 1) กฎที่สามารถรวมได้ 2) กฎที่ไม่สามารถไปถึง และกฎที่ขัดแย้งกัน

ผลที่ได้จากการพัฒนาโปรแกรมพบว่าสามารถลดรูปกฎของ IP Firewall และ IP Filter ได้ตามที่ต้องการ โดยจะสามารถลดรูปกฎที่อยู่ในรูปแบบของ subnet กฎที่ไม่สามารถไปถึง กฎที่ขัดแย้งกัน และมีการแสดงรายงานออกมาว่ากฎข้อใดถูกลดรูปในกรณีไหน มีความขัดแย้งกับกฎข้อใดบ้าง นอกจากนี้ยังสามารถระบุให้มีการจัดเก็บกฎที่ผ่านการลดรูปแล้วในแฟ้มข้อมูลที่กำหนดได้ ซึ่งเมื่อมีการนำกฎที่ผ่านการลดรูปไปใช้งานจะทำให้การทำงานของไฟร์วอลล์มีประสิทธิภาพมากขึ้น และผู้ดูแลระบบสามารถดูแลตรวจสอบกฎของไฟร์วอลล์ได้ง่าย

5.2 ข้อเสนอแนะ

โปรแกรมเครื่องมือสำหรับวิเคราะห์ และลดจำนวนกฎของไฟร์วอลล์ สามารถนำไปพัฒนาต่อให้สมบูรณ์ และเกิดประโยชน์มากขึ้นได้ โดยเพิ่มเติมความสามารถของการทำงานดังนี้

1) ทำให้โปรแกรมสามารถแปลงกฎระหว่างกฎของ IP Firewall และกฎของ IP Filter ได้

2) พัฒนาให้สามารถใช้งานได้กับโปรแกรมไฟร์วอลล์ชนิดอื่นๆได้ เช่น IP table เป็นต้น โดยอาจจะทำได้โดยเพิ่มเติมในส่วนของการแปลงกฎให้อยู่ในโครงสร้างแบบกลาง และการแปลงโครงสร้างแบบกลางให้อยู่ในรูปของกฎทั่วไป

3) ขยายความสามารถในการลดรูปโดยอาจจะมีการพิจารณาจะส่วนประกอบอื่นๆ นอกเหนือจาก หมายเลข IP ต้นทาง-ปลายทาง และพอร์ต

4) ทำให้โปรแกรมนั้นสามารถรองรับโปรโตคอล IP version 6

บรรณานุกรม

Chris Hare and Karanjit Siyan.1996. **Internet firewalls and Network Security**. 2nd Edition. Newriders.

Daniel Boulet. et al. 1997. IPFIREWALL(4). FreeBSD Kernel Interfaces Manual,

D. Brent Chapman and Elizabeth D. Zwicky. 1995. Building Internet Firewalls. O'Reilly & Associates, Inc.

James P. Anderson. et al. 1997. "Firewalls : An Expert Roundtable". IEEE Magazine,

Michael R. Lyu and Lorrien K. Y. Lau. 2000. "Firewall security : Policies , Testing and Performance Evaluation". IEEE Magazine.

Robert Zalenski. 2002. "Firewall technologies". IEEE Potentials.

Steven M. Bellovin and William R. Cheswick. 1994. "Network Firewalls". IEEE Communications Magazine.

Ugen J. S. Antsilevich. et al . 2001. IPFW(8), FreeBSD System Manager's Manual.

IPF(4), FreeBSD's Manual.

IPF(5), FreeBSD's Manual.

ภาคผนวก

ภาคผนวก ก
การติดตั้งและใช้งานโปรแกรมเครื่องมือสำหรับวิเคราะห์ และลดจำนวนกฎของไฟร์วอลล์

การจะใช้งานโปรแกรมเครื่องมือสำหรับวิเคราะห์ และลดจำนวนกฎของไฟร์วอลล์นี้ จำเป็นต้องมีการติดตั้งโปรแกรม ซึ่งมีขั้นตอนต่างๆ ดังนี้

1) แตกที่ถูกบีบอัดโดยใช้คำสั่ง tar xvfz ตัวอย่างเช่น

```
$ tar xvfz  ipfw_analysis.tar.gz
```

2) จากนั้นก็เข้าไปในไดเร็กเทอร์รีของตัวโปรแกรมที่เราทำการแตกออกมา ตัวอย่างเช่น

```
$ cd ipfw_analysis
```

3) ทำการคอมไฟล์โปรแกรมด้วยคำสั่ง make

```
$ make
```

4) ทำให้สามารถเรียกใช้คำสั่งจากตำแหน่งใดก็ได้ โดยไปเพิ่มตำแหน่งที่อยู่ของโปรแกรมไว้ในตัวแปร $PATH

5) สามารถใช้งานโปรแกรมได้โดยใช้คำสั่ง ipfw_analysis

โปรแกรมเครื่องมือสำหรับวิเคราะห์ และลดจำนวนกฎของไฟร์วอลล์ จะอยู่ในรูปแบบคำสั่ง ซึ่งมีรูปแบบดังนี้

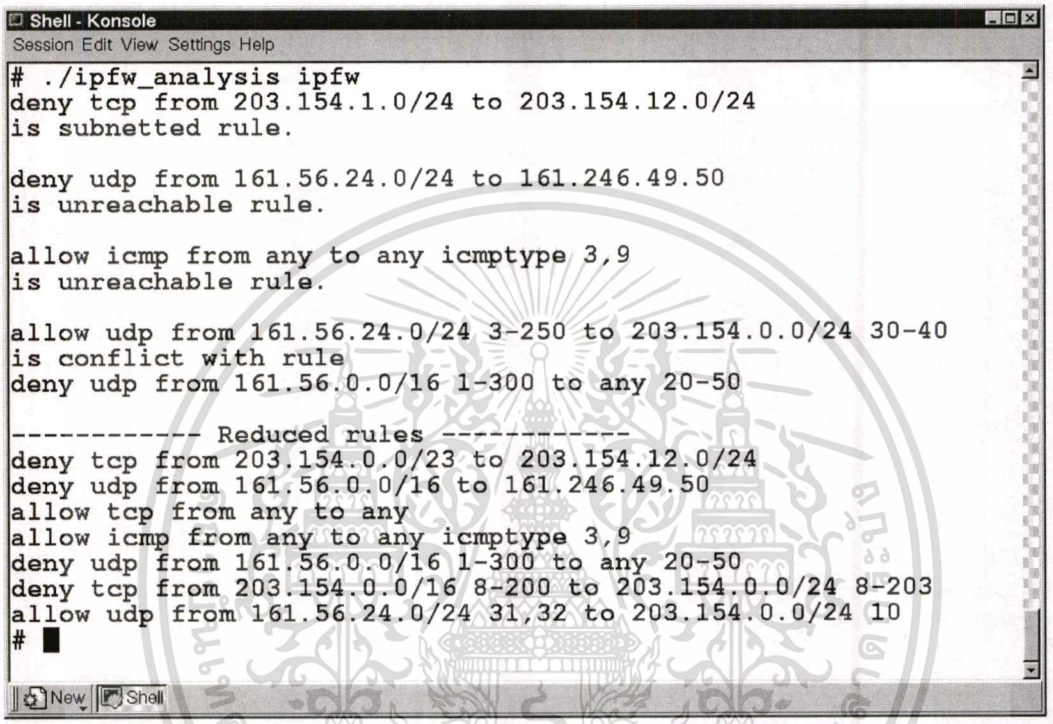
```
$ ipfw_analysis [-nf] [-o output-file] input-file
```

[-n] หมายถึง ไม่ต้องมีการแสดงผลลัพธ์ออกทางหน้าจอ

[-f] หมายถึง กฎที่เป็น input เป็นกฎของ IP Filter ถ้าไม่มีการระบุ option นี้จะหมายถึงว่ากฎที่เป็น input เป็นกฎของ IP Firewall

[-o output-file] เป็นการระบุให้จัดเก็บกฎที่มีการลดรูปแล้วลงในแฟ้มข้อมูลที่กำหนด

ยกตัวอย่างเช่นคำสั่ง " $ ipfw_analysis ipfw " จะเป็นการสั่งให้โปรแกรมเครื่องมือสำหรับวิเคราะห์ และลดจำนวนกฎของไฟร์วอลล์ลดรูปกฎของ IP Firewall ในแฟ้มข้อมูลที่มีชื่อว่า "ipfw" โดยผลลัพธ์จะแสดงออกทางหน้าจอ ซึ่งแบ่งเป็น 2 ส่วนคือ 1) ส่วนของการรายงานว่ากฎข้อใดเป็นกฎประเภทใด 2) เป็นการแสดงถึงกฎที่ได้มีการวิเคราะห์และลดรูปแล้ว ดังรูป



รูปที่ ก.1 ตัวอย่างการทำงานของโปรแกรม

ประวัติผู้เขียนโครงงาน

ชื่อผู้จัดทำโครงงาน นายเกริกศักดิ์ ลิขิตสุภิณ

วันเดือนปีเกิด 16 กรกฎาคม 2522

สถานที่เกิด กรุงเทพมหานคร

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนศึกษาพัฒนา	กรุงเทพมหานคร
มัธยมศึกษาตอนต้น	โรงเรียนพรตพิทยพยัต	กรุงเทพมหานคร
มัธยมศึกษาตอนปลาย	โรงเรียนพรตพิทยพยัต	กรุงเทพมหานคร
อุดมศึกษา	มหาวิทยาลัยบูรพา	ชลบุรี