การปรับปรุงความเร็วในการทำงานของไฟร์วอลล์

Optimizing Firewall

นายณัฐพล ชัยยศลาภ

นายดุลย์ภาส ตระการกิจวิชิต

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

การปรับปรุงความเร็วในการทำงานของไฟร์วอลล์

Optimizing Firewall

โดย

นายณัฐพล ชัยยศลาภ

นายดุลย์ภาส ตระการกิจวิชิต

อาจารย์ที่ปรึกษา

ผศ. ธนา หงษ์สุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การปรับปรุงความเร็วในการทำงานของไฟร์วอลล์

Optimizing Firewall

ผู้จัดทำ

1. นายณัฐพล ชัยยศลาภ รหัสนักศึกษา 45010235
2. นายดุลย์ภาส ตระการกิจวิชิต รหัสนักศึกษา 45010271

อาจารย์ที่ปรึกษา

(ผศ. ธนา หงษ์สุวรรณ)

การปรับปรุงความเร็วในการทำงานของไฟร์วอลล์

นายณัฐพล ชัยยศลาภ	45010235
นายดุลย์ภาส ตระการกิจวิชิต	45010271
อ. ธนา หงษ์สุวรรณ	อาจารย์ที่ปรึกษา
ปีการศึกษา 2548	

บทคัดย่อ

การรักษาความปลอดภัยในระบบเครือข่ายคอมพิวเตอร์ในปัจจุบันนั้น องค์กรทุกๆ องค์กรนั้นจะใช้ตัวไฟร์วอลล์ในการรักษาระบบความปลอดภัย ซึ่งเรียกได้ว่าระบบเครือข่ายนั้นจะขาดไฟร์วอลล์เสียไม่ได้ โดยเฉพาะไอพีเทเบิ้ลส์เป็นไฟร์วอลล์พื้นฐานที่มีอยู่ในระบบปฏิบัติการลินุกซ์ ซึ่งมีการใช้งานกันอย่างแพร่หลาย ด้วยเหตุผลนี้สิ่งสำคัญคือไฟร์วอลล์ควรจะทำงานได้อย่างมีประสิทธิภาพมากที่สุด

ไฟร์วอลล์เป็นโปรแกรมหนึ่งที่มีความสำคัญกับระบบคอมพิวเตอร์ การทำงานของไฟร์วอลล์ จะทำงานโดยใช้กฎเพื่ออนุญาตหรือไม่อนุญาตการติดต่อจากเครื่องอื่นๆ โดยจะทำงานตามกฎจากบนลงล่าง ซึ่งกลุ่มผู้วิจัยได้มีการเล็งเห็นแล้วว่าหากมีกฎจำนวนมาก จะทำให้การทำงานต่างๆ ของไฟร์วอลล์นั้นทำงานช้าลง โดยโครงงานนี้จะศึกษาวิธีการต่างๆ ที่จะปรับปรุงการทำงานของไฟร์วอลล์ให้มีการทำงานที่เร็วขึ้น ซึ่งจะเน้นส่วนที่เป็นลำดับต่างๆ ของกฎในแต่ละตารางและด้านความซ้ำซ้อนต่างๆ ของกฎ ที่จะพิจารณาถึงความสัมพันธ์ของกฎที่ซ้ำซ้อนกันว่าสามารถลบออกไปได้หรือไม่ โดยไม่ให้มีผลกระทบต่อประสิทธิภาพในการรักษาความปลอดภัยของระบบของไฟร์วอลล์

OPTIMIZING FIREWALL

Mr. Natthapol Chaiyoslarp	45010235
Mr. Dunyapad Trakarnkitvichit	45010271
Asst. Prof. Thana Hongsuwan	Advisor
Academic Year 2005	

ABSTRACT

In this time almost securitys in network system for organization use firewall to protect their data. You would say that network system can't live without a firewall. IPTables is a basic Linux's firewall that many people use it widely. From this reason, firewall should work with the most efficiently

Firewall is a program that's important for network system. It use rules to decision the packet of data that block or accept from the top to bottom rules. If firewall has many rules, it'll work slower. This project is study how to improve speed of firewall while it's working that's concentrate on relation and anomaly of rules. Moreover, it's about reordering firewall's rules to make matching packet faster. Finally, the overall semantics of the rules mustn't change and it can be improve

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำ และคำปรึกษาจาก รศ.ธนา หงษ์สุวรรณ ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอกราบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ทั้งในห้องโปรเจ็ค Network ห้องอื่นๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นายณัฐพล ชัยยศลาภ

นายดุลย์ภาส ตระการกิจวิชิต

สารบัญ

สารบัญตาราง

สารบัญภาพประกอบ

สารบัญภาพประกอบ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

การรักษาความปลอดภัยในระบบเครือข่ายคอมพิวเตอร์ในปัจจุบันนั้น องค์กรทุกๆ องค์กรนั้นจะใช้ตัวไฟร์วอลล์ในการรักษาระบบความปลอดภัย ซึ่งเรียกได้ว่าระบบเครือข่ายนั้นจะขาดไฟร์วอลล์เสียไม่ได้ โดยเฉพาะไอพีเทเบิ้ลส์เป็นไฟร์วอลล์พื้นฐานที่มีอยู่ในระบบปฏิบัติการลินุกซ์ ซึ่งมีการใช้งานกันอย่างแพร่หลาย ด้วยเหตุผลนี้สิ่งสำคัญคือไฟร์วอลล์ควรจะทำงานได้อย่างมีประสิทธิภาพมากที่สุด

ไฟร์วอลล์เป็นโปรแกรมหนึ่งที่มีความสำคัญกับระบบคอมพิวเตอร์ การทำงานของไฟร์วอลล์จะทำงานโดยใช้กฎเพื่ออนุญาตหรือไม่อนุญาตการติดต่อจากเครื่องอื่นๆ โดยจะทำงานตามกฎจากบนลงล่าง ซึ่งกลุ่มผู้วิจัยได้มีการเล็งเห็นแล้วว่าหากมีกฎจำนวนมาก จะทำให้การทำงานต่างๆ ของไฟร์วอลล์นั้นทำงานช้าลง โดยโครงงานนี้จะศึกษาวิธีการต่างๆ ที่จะปรับปรุงการทำงานของไฟร์วอลล์ให้มีการทำงานที่เร็วขึ้น ซึ่งจะเน้นส่วนที่เป็นลำดับต่างๆ ของกฎในแต่ละตารางและด้านความซ้ำซ้อนต่างๆ ของกฎ ที่จะพิจารณาถึงความสัมพันธ์ของกฎที่ซ้ำซ้อนกันว่าสามารถลบออกไปได้หรือไม่ โดยไม่ให้มีผลกระทบต่อประสิทธิภาพในการรักษาความปลอดภัยของระบบของไฟร์วอลล์

1.2 วัตถุประสงค์ของโครงงาน

ปริญญานิพนธ์ฉบับนี้มุ่งหวังเพื่อศึกษาการทำงานของไฟร์วอลล์ไอพีเทเบิ้ลส์ และการใช้ออพชั่นต่างๆ ของไอพีเทเบิ้ลส์ทั้งในการเพิ่มและลบกฎต่างๆ รวมถึงการเก็บบันทึกรายละเอียดของแพ็คเก็ตที่มีรายละเอียดตรงกับกฎในไอพีเทเบิ้ลส์ลงล็อคไฟล์ รวมถึงศึกษาความสัมพันธ์และความขัดแย้งของกฎในแต่ละแบบ เพื่อตรวจสอบไม่ให้การลบหรือจัดลำดับของกฎใหม่ตามปริมาณการใช้งานของเครือข่ายนั้นมีผลทำให้ความหมายโดยรวมของกฎทั้งหมดเปลี่ยนไป ดังนั้นในปริญญานิพนธ์นี้จึงเสนอวิธีการปรับปรุงความเร็วในการทำงานของไฟร์วอลล์ โดยใช้การตรวจจับแพ็คเก็ตโดยใช้ออพชั่น LOG ของไอพีเทเบิ้ลส์ นำมาคำนวณปริมาณการใช้งานภายในเครือข่ายเพื่อใช้เปรียบเทียบเป็น

ข้อมูลในการนำไปจัดลำดับกฎต่อไป และใช้เชลล์สคริปต์เขียนโปรแกรมที่จะกระทำการต่างๆ ในการปรับปรุงความเร็วในการทำงานของไฟร์วอลล์ทั้งหมด

1.3 ขอบเขตของโครงงาน

ในปริญญานิพนธ์ฉบับนี้ได้นำเสนอวิธีการปรับปรุงความเร็วในการทำงานของไฟร์วอลล์ (Optimizing Firewall) ทั้งการปรับปรุงโดยลบกฎที่ซ้ำซ้อนออกและจัดลำดับของกฎเรียงตามปริมาณที่ใช้งานภายในเครือข่าย ซึ่งพิจารณาจากความสัมพันธ์และความขัดแย้งของกฎในแต่ละแบบรวมถึงการจัดการกับกฎเมื่อมีความสัมพันธ์และความขัดแย้งแบบนั้นๆ รวมถึงทำการทดสอบผลที่ได้โดยใช้การตรวจจับปริมาณแพ็คเก็ตที่ผ่านภายในระบบเครือข่าย มาเปรียบเทียบว่าผลเป็นอย่างไร

1.4 วิธีการดำเนินการ

1. ศึกษาเกี่ยวกับพื้นฐานในการใช้งานระบบปฏิบัติการณ์ลินุกซ์ การใช้คำสั่งต่างๆ เบื้องต้น การติดตั้งและถอดถอนโปรแกรม
2. ศึกษาการทำงานของไฟร์วอลล์ไอพีเทเบิ้ลส์ รวมถึงคำสั่งที่ใช้ต่างๆ การเพิ่มหรือลบกฎให้เป็นไปตามที่ต้องการ
3. ศึกษาเกี่ยวกับการตรวจจับปริมาณการใช้งานของเครือข่าย โดยใช้ทั้งไลบรารี PCAP และการใช้ log-prefix ซึ่งเป็นออพชั่นของไฟร์วอลล์ไอพีเทเบิ้ลส์ และทดลองใช้การตรวจจับแต่ละแบบแล้วเลือกใช้แบบที่เหมาะสมที่สุด
4. ศึกษาความสัมพันธ์และความขัดแย้งของกฎของไฟร์วอลล์แต่ละกฎ รวมถึงการจัดการในความสัมพันธ์และความขัดแย้งแต่ละแบบในการลบกฎที่ซ้ำซ้อนหรือจัดลำดับกฎใหม่ตามปริมาณการใช้งานภายในเครือข่ายของแต่ละกฎ
5. วิเคราะห์ และออกแบบโครงสร้างของโปรแกรมในแต่ละส่วน
6. พัฒนาโปรแกรมที่ใช้ในการตรวจสอบหาความสัมพันธ์ของกฎ โปรแกรมที่ใช้ในการลบกฎที่มีความสัมพันธ์และความขัดแย้งที่บ่งบอกว่ากฎนั้นซ้ำซ้อนกันและทำการลบกฎที่ซ้ำซ้อนออก โปรแกรมที่ใช้ใส่กฎที่มีทาร์เก็ตฟังก์ชันเป็นการเก็บรายละเอียดของแพ็คเก็ตลงล็อคไฟล์เพื่อใช้ในการคำนวณปริมาณการใช้งานเครือข่ายเพื่อเปรียบเทียบ โปรแกรมคำนวณปริมาณการใช้งานในเครือข่ายออกมาเป็นตัวเลขที่สามารถเปรียบเทียบได้ โปรแกรมจัดลำดับใหม่ของกฎเรียงตามปริมาณการใช้งานภายในเครือข่ายเพื่อปรับปรุงความเร็วในการตรวจสอบแพ็คเก็ตของระบบ

7. ทดสอบการทำงานของตัวโปรแกรมใช้ปรับปรุงการทำงานของไฟร์วอลล์ ทดลองใช้เพื่อหาข้อผิดพลาดและตรวจสอบผลที่ได้โดยเปรียบเทียบกับการปรับปรุงด้วยมือ และทำการตรวจจับปริมาณการใช้งานของเครือข่ายเพื่อเปรียบเทียบก่อนและหลังการปรับปรุงการทำงานว่ามีผลเป็นอย่างไร

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้ ความเข้าใจเกี่ยวกับกระบวนการการทำงานของไฟร์วอลล์ไอพีเทเบิ้ลส์และการใช้งาน
2. ได้รับความรู้ความเข้าใจเกี่ยวกับระบบไฟร์วอลล์ในเครือข่ายและการจัดการด้านความปลอดภัยเบื้องต้น
3. ได้รับความรู้และประสบการณ์การเขียนโปรแกรมโดยใช้เชลล์สคริปต์รวมถึงการใช้คำสั่งต่างๆ บนระบบปฏิบัติการลินุกซ์
4. การทำงานของระบบเครือข่ายเร็วขึ้น ไฟร์วอลล์ไอพีเทเบิ้ลสทำการกรองแพ็คเก็ตของข้อมูลได้อย่างมีประสิทธิภาพ

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้ในโครงงาน

ในหัวข้อนี้จะกล่าวถึงทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องในการวิจัย และพื้นฐานของไฟร์วอลล์ โดยจะทำการศึกษาไอพีเทเบิ้ลส์ ซึ่งเนื้อหาในบทนี้จะกล่าวถึงโปรโตคอลทีซีพี/ไอพี ความเป็นมา การเชื่อมต่อ ทั้งในส่วนของทีซีพี ยูดีพี และไอพี

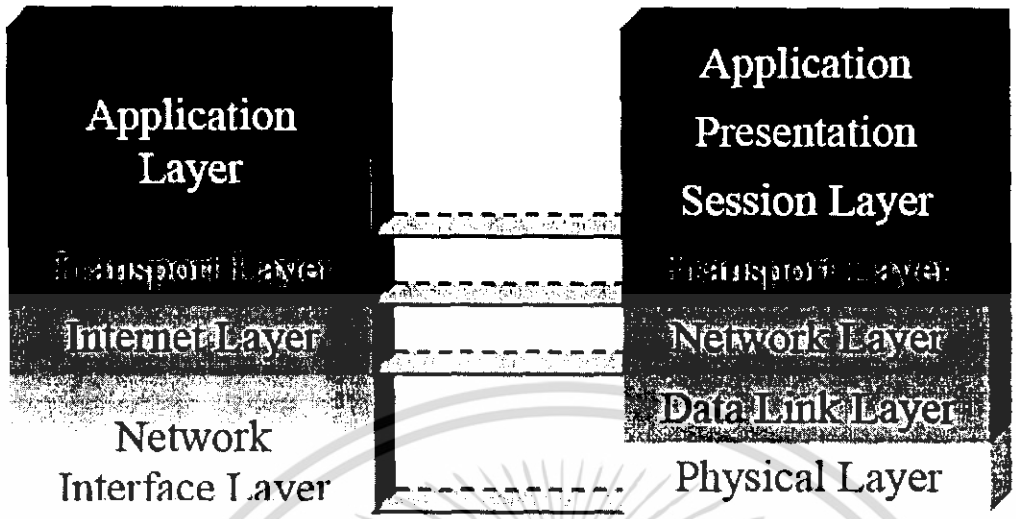
2.1 โพรโตคอลทีซีพี/ไอพี

ความเป็นมาของโพรโตคอลทีซีพี/ไอพี

ทีซีพี/ไอพี เป็นโพรโตคอลมาตรฐานที่ใช้กันอยู่ในระบบปฏิบัติการแบบยูนิกซ์ เริ่มพัฒนาโดยกระทรวงกลาโหมของสหรัฐฯ ในปี ค.ศ. 1969 เพื่อเชื่อมโยงเครื่องคอมพิวเตอร์ทางทหารของแต่ละหน่วยที่อยู่ห่างไกลกัน โดยมีจุดประสงค์ คือสร้างระบบเครือข่ายให้เครื่องคอมพิวเตอร์สามารถรับส่งข้อมูลกันได้ แม้ว่าสายส่งข้อมูลบางส่วนจะถูกทำลายเสียหายไปก็ตาม เพื่อใช้งานในยามเกิดสงคราม โดยเครือข่ายที่จัดตั้งในระยะแรกชื่อว่า Advanced Research Projects Agency Network หรือ อาร์พาเน็ต (ARPANET) ต่อมาได้พัฒนาเป็นเครือข่ายอินเทอร์เน็ต (INTERNET) โพรโตคอลนี้เหมาะสำหรับเชื่อมต่อคอมพิวเตอร์ทั้งใกล้ และไกลเข้าด้วยกัน และมีมาตรฐานรองรับทำให้ผู้ผลิตฮาร์ดแวร์ และซอฟต์แวร์ สามารถสร้างอุปกรณ์ และโปรแกรมที่จะรองรับการทำงานของโพรโตคอลนี้ ทำให้เครื่องคอมพิวเตอร์สามารถรับส่งข้อมูลกันได้ไม่ว่าจะเป็นเครื่องขนาดเล็กหรือขนาดใหญ่ หรือใช้ระบบปฏิบัติการอะไรก็ตาม ทีซีพี/ไอพี (TCP/IP) เป็นชุดโพรโตคอลที่ประกอบด้วยโพรโตคอลต่างๆ หลายโพรโตคอล แต่ละโพรโตคอลมีคุณลักษณะ และมีความสามารถในการทำงานแตกต่างกัน โดยที่ในบทนี้ได้กล่าวถึงรายละเอียดและคุณสมบัติของโพรโตคอลที่สำคัญบางโพรโตคอล

การเชื่อมต่อของโพรโตคอลทีซีพี/ไอพี (TCP/IP Linking)

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



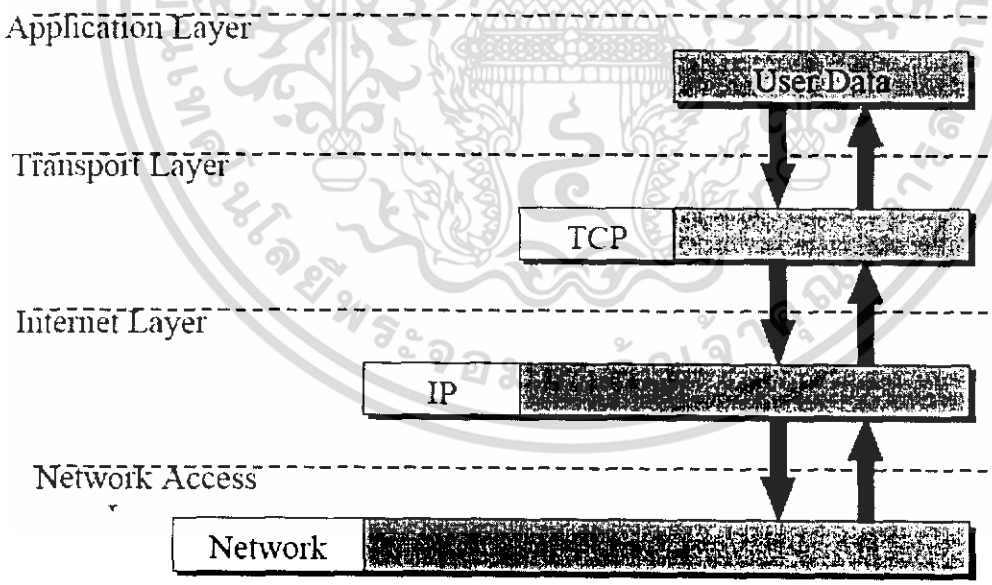
รูปที่ 2-1 แสดงการเปรียบเทียบเลเยอร์ของโอเอสไอกับเลเยอร์ของทีซีพี/ไอพี

ในแต่ละระดับชั้นของทีซีพี/ไอพีมีการทำงานที่แตกต่างกัน ตั้งแต่การติดต่อกับแอพพลิเคชัน จนกระทั่งแปลงเป็นสัญญาณส่งไปตามสายสัญญาณ ซึ่งการทำงานในแต่ละระดับชั้นของทีซีพี/ไอพี มีดังตารางที่ 2-1

ชื่อระดับชั้น	หน้าที่
1. ชั้นแอพพลิเคชัน (Application Layer)	รองรับการทำงานของแอพพลิเคชันต่างๆ ที่ทำงานเป็นโพรเซสอยู่ในเครื่องต้นทางและปลายทาง โดยจัดการเชื่อมต่อระหว่างโพรเซส หรือแอพพลิเคชันที่อยู่ต่างเครื่องกัน โดยการทำงานของแอพพลิเคชันต่างๆมีการติดต่อกันตามแต่ละโพรโตคอลเฉพาะแล้วแต่แอพพลิเคชันที่ใช้งาน ซึ่งจะขอบริการจากชั้นทรานสปอร์ตอีกทีหนึ่ง
2. ชั้นทรานสปอร์ต (Transport Layer)	สร้างการเชื่อมต่อกันระหว่างแอพพลิเคชันแบบ end-to-end โดยจุดที่เชื่อมต่อกันเพื่อรับส่งข้อมูลนี้เรียกว่า พอร์ต (port) หรือซ็อกเก็ต (Socket) ในชั้นนี้มีบริการหลักอยู่ 2 แบบ คือ Connection Oriented โดยเรียกผ่านโพรโตคอลทีซีพี (TCP: Transmission Control Protocol) และ Connectionless ซึ่งเรียกผ่านโพรโตคอลยูดีพี (UDP: User Datagram Protocol) ซึ่งกล่าวถึงในหัวข้อถัดไป

ชื่อระดับชั้น	หน้าที่
3. ชั้นอินเทอร์เน็ต (Internet Layer)	ส่งผ่านข้อมูลระหว่างเครือข่าย โดยมีโพรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆ ในอินเทอร์เน็ต คือ ไอพี (Internet Protocol: IP) ซึ่งกล่าวถึงในหัวข้อถัดไป นอกจากนี้ในชั้นนี้ยังมีโพรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ ไอซีเอ็มพี (Internet Control Message Protocol: ICMP) และเออาร์พี (Address Resolution Protocol: ARP)
4. ชั้นเน็ตเวิร์กอินเตอร์เฟส (Network Interface Layer)	แปลงข้อมูลให้อยู่ในรูปที่เหมาะสมกับเครือข่ายแต่ละแบบ ซึ่งแตกต่างกันออกไป และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่าย

ตารางที่ 2-1 การทำงานของแต่ละระดับชั้นของทีซีพี/ไอพี

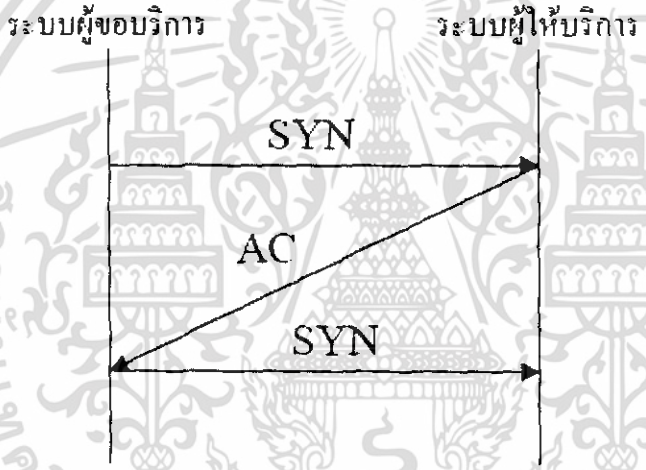


รูปที่2-2 แสดงการข้อมูลที่ส่งผ่านในโมเดลของทีซีพี/ไอพี

ในชุดโพรโตคอลทีซีพี/ไอพีนี้ มีโพรโตคอลหลัก ที่ขอกล่าวถึง 3 โพรโตคอล ได้แก่ โพรโตคอลทีซีพี โพรโตคอลยูดีพี ซึ่งทำงานในชั้นทรานสปอร์ต และโพรโตคอลไอพี ซึ่งทำงานในชั้นอินเทอร์เน็ต โดยมีรายละเอียดดังต่อไปนี้

โพรโตคอลทีซีพี (TCP: Transmission Control Protocol)

การทำงานที่สำคัญอย่างหนึ่งของโพรโตคอลทีซีพี คือ การทำ "3-Way Handshake" ซึ่งเป็นกระบวนการเริ่มต้นในการสร้างการเชื่อมต่อในชั้นทรานสปอร์ต กล่าวคือ ในการติดต่อกันระหว่างระบบในเครือข่ายต้องมีการสร้างการเชื่อมต่อไปยังระบบที่ให้บริการก่อน โดยผู้ขอบริการส่งสัญญาณ SYN เพื่อขอบริการ จากนั้นผู้ให้บริการจะส่งสัญญาณ ACK เพื่อตอบรับการเชื่อมต่อที่ร้องขอมา จึงสามารถรับส่งข้อมูลกันได้ ดังรูปที่ 2-3



รูปที่ 2-3 แสดงการทำ 3-Way Handshake

การเชื่อมต่อแบบ 3-Way Handshake นี้ เป็นการตรวจสอบความพร้อมของทั้งฝ่ายส่ง และฝ่ายรับ และการกำหนดค่าเริ่มต้นของพารามิเตอร์ต่างๆ ของทั้งสองฝ่ายให้ตรงกัน หลังจากกระบวนการทำ 3-WayHandshake สิ้นสุด ทั้งสองฝ่ายจึงสามารถรับ และส่งข้อมูลซึ่งกัน และกันได้

ดังนั้นโพรโตคอลทีซีพีจึงเป็นโพรโตคอลที่มีการรับส่งข้อมูลแบบ "Connection Oriented" ทำให้การทำงานของทีซีพีมีความน่าเชื่อถือมากขึ้น หน้าที่การทำงานของทีซีพีในการรับส่งข้อมูลมีหน้าที่หลัก 6 ข้อ คือ

1. ควบคุมการรับส่งข้อมูล (Basic Data Transfer)
2. ความน่าเชื่อถือในการรับส่งข้อมูล (Reliability)
3. ควบคุมการไหลของข้อมูล (Flow Control)
4. การทำมัลติเพล็ก (Multiplexing)

5. ควบคุมการเชื่อมต่อ (Connection)

6. ความปลอดภัยในการรับส่งข้อมูล (Security)

ส่วนประกอบของทีซีพีเฮดเดอร์

1. Source Port : เป็นหมายเลขพอร์ตของบริการที่เครื่องต้นทาง

2. Destination Port : เป็นหมายเลขพอร์ตของบริการเครื่องปลายทาง

3. Sequence Number : เป็นหมายเลขที่บอกลำดับของการรับส่งข้อมูลของเครื่องที่ต้องการขอส่งข้อมูล

4. Acknowledgement Number : เป็นหมายเลขที่บอกลำดับของการรับส่งข้อมูลที่ฝั่งรับข้อมูลปกติ

5. Data Offset : เป็นตัวบอกค่าออฟเซตของข้อมูล เพราะทีซีพีนั้นไม่มีการกำหนดความยาวแน่นอนของข้อมูล จึงต้องมีออฟเซตเป็นตัวบอก

6. Flag : เป็นบิตที่บอกชนิดของข้อมูล ได้แก่

- URG : Urgent Pointer Field Significant - แสดง Urgent Pointer
- ACK : Acknowledgement Field Significant – แสดงการ Acknowledgement
- PSH : Push Function
- RST : Reset The Connection - แสดงเมื่อรีเซ็ทการเชื่อมต่อ
- SYN : Synchronize Sequence Number - หมายเลขแพ็กเก็ตที่ส่งแบบซิงโครนัส
- FIN : No more data from sender - แสดงว่าไม่มีข้อมูลที่ส่งจากผู้ส่งแล้ว

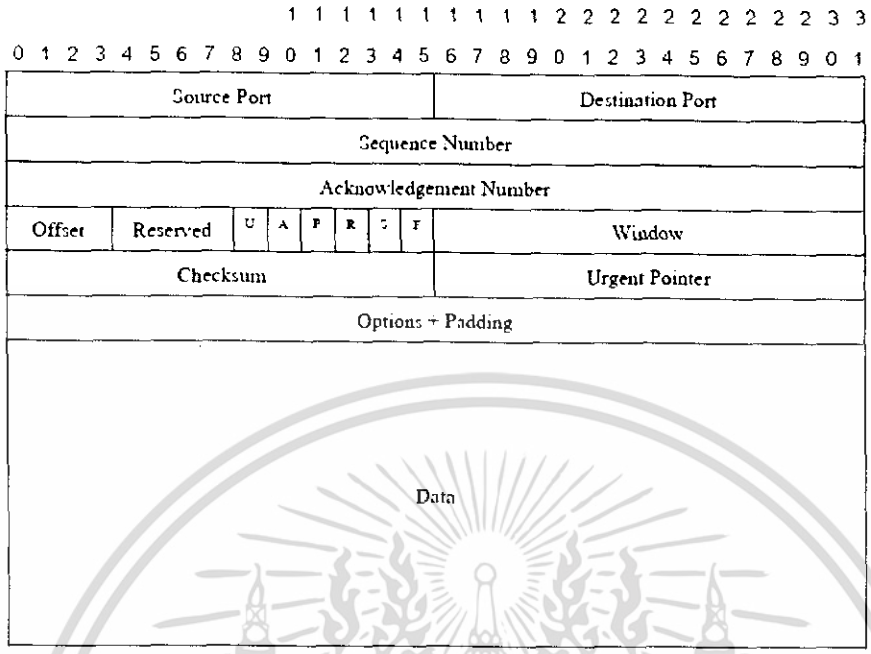
7. Window : เป็นเลขบอกจำนวนของอ็อกเต็ต (octet) ของข้อมูล จัดการในส่วนของ end-to-end flow control

8. Checksum : เป็นส่วนที่ตรวจสอบความถูกต้องของข้อมูล

9. Urgent Pointer : เป็นตัวชี้ตำแหน่งของ Urgent Data

10. Option and Padding : เป็นตัวบอกออปชันของโพรเซสที่ใช้ทีซีพี

11. Data : เนื้อข้อมูลที่ต้องการสื่อสาร มีขนาดได้ไม่ต่ำกว่า 5 32-บิตเวิร์ด (6 บิตแรกสงวนไว้และกำหนดให้เป็นศูนย์)



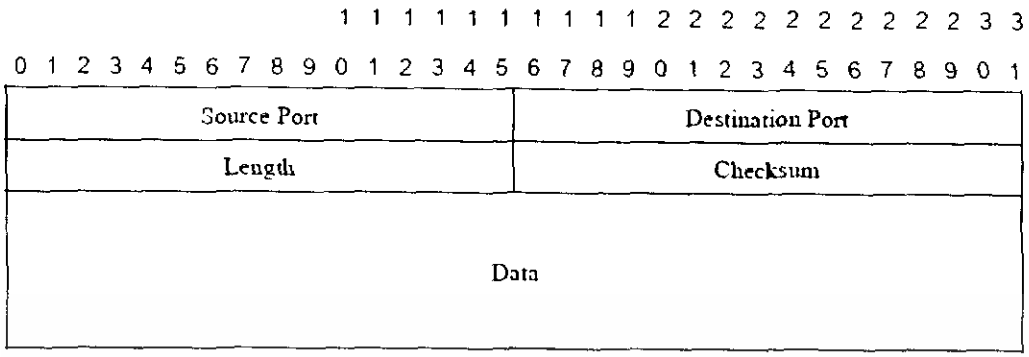
รูปที่ 2-4 แสดงแพ็กเก็ตทีซีพี

โพรโตคอลยูดีพี (UDP: User Datagram Protocol)

โพรโตคอลยูดีพีเป็นโพรโตคอลในการติดต่อสื่อสารในชั้นทรานสปอร์ต (Transport Layer) การทำงานคล้ายกับทีซีพีมาก คือจัดการเกี่ยวกับการสื่อสารระหว่างเครื่อง แต่เป็นแบบ Connectionless คือ ทั้งฝ่ายส่ง และฝ่ายรับไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน โดยไม่ต้องมีการแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโพรโตคอลทีซีพี และไม่มีการส่งสัญญาณตรวจสอบว่าข้อมูลถึงเครื่องปลายทางอย่างถูกต้องครบถ้วนในการส่งข้อมูลแต่ละครั้ง จึงไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล

ส่วนประกอบของ UDP Frame

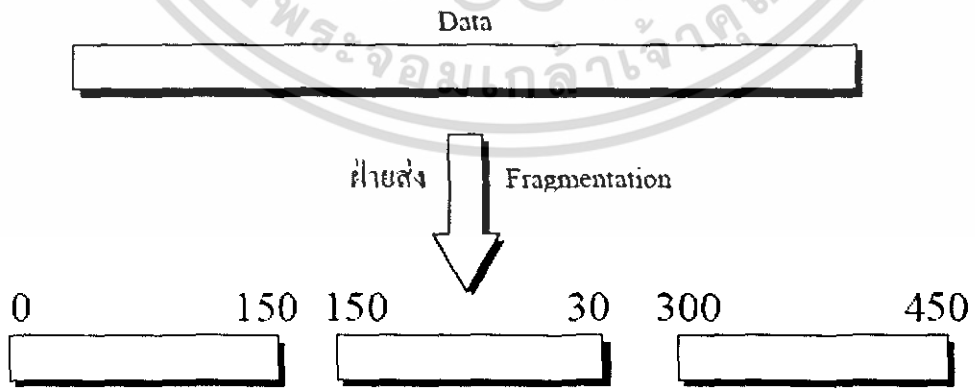
1. *Source Port* : เป็นค่าตัวเลข 16 บิต บอกพอร์ตของบริการที่เครื่องต้นทาง
2. *Destination Port* : เป็นค่าตัวเลข 16 บิต บอกพอร์ตของบริการที่เครื่องปลายทาง
3. *Length* : เป็นค่าตัวเลข 16 บิต บอกความยาวของข้อมูล
4. *Checksum* : เป็นค่าตัวเลข 16 บิต ตรวจสอบความถูกต้องของข้อมูลที่ส่ง



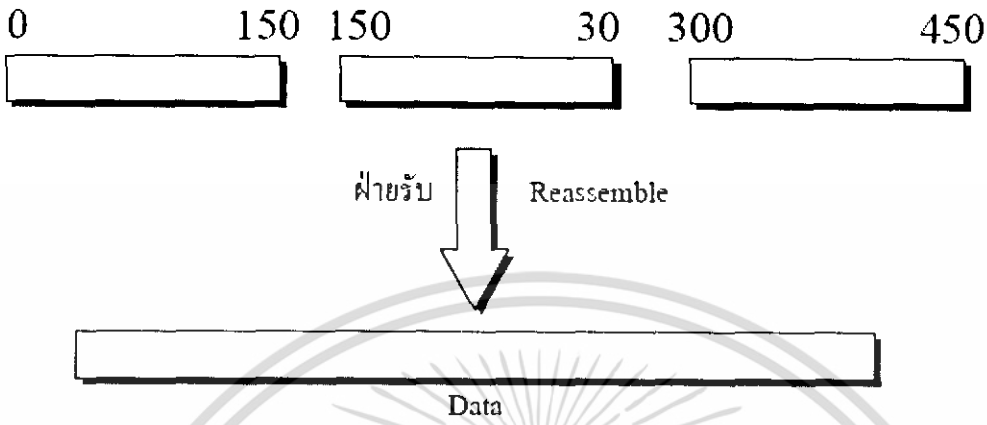
รูปที่ 2-5 แสดงแพ็กเก็ตยูดีพี

โพรโตคอลไอพี (IP: Internet Protocol)

โพรโตคอลไอพีเป็นโพรโตคอลที่จัดการเกี่ยวกับแอดเดรสของแต่ละแพ็กเก็ต เพื่อให้ส่งแพ็กเก็ตต่างๆ ไปยังเป้าหมายได้ถูกต้อง การทำงานของไอพีเป็นเพียงการส่งข้อมูลไปยังเครื่องเป้าหมายเท่านั้น ไม่มีการส่งสัญญาณขอบริการ หรือสัญญาณให้บริการระหว่างกันเหมือนทีซีพี เรียกว่าการเชื่อมต่อแบบ Connectionless ซึ่งระบบทั้งสองตั้งสมมติฐานว่าการเชื่อมต่อระหว่างกันไม่มีความผิดพลาดเกิดขึ้นแน่ เนื่องจากมาตรฐานในเครือข่ายมีหลากหลาย ขนาดของแพ็กเก็ตในแต่ละมาตรฐานจึงมีความแตกต่างกันออกไป ทำให้การส่งข้อมูลระหว่างอุปกรณ์ในเครือข่ายนั้นอาจมีการแบ่งข้อมูลออกเป็นแพ็กเก็ตย่อยๆ ในระหว่างการส่ง เรียกว่า การทำแฟร็กเมนเตชัน (Fragmentation) เช่น แพ็กเก็ตของ FDDI มีขนาด 4,500ไบต์ หากเครื่องปลายอยู่ในเครือข่าย Ethernet ซึ่งมีขนาดของแพ็กเก็ตสูงสุดเพียง 1,500 ไบต์ ดังนั้นการส่งแพ็กเก็ตไปยังเครื่องปลายทางจึงต้องมีการแบ่งเป็นแพ็กเก็ตย่อย และเมื่อแพ็กเก็ตย่อยมาถึงเครื่องเป้าหมายก็จะมารวมกันเป็นแพ็กเก็ตเดิมที่มีขนาด 4,500 ไบต์อีกครั้ง เรียกการรวมกันนี้ว่า การรีแอสเซมเบิ้ล(Reassemble) ซึ่งทำให้ได้ข้อมูลเหมือนที่ส่งมาจากเครื่องต้นทาง



รูปที่ 2-6 แสดงการทำแฟร็กเมนเตชัน



รูปที่ 2-7 แสดงการรีแอสเซมเบิ้ล

ส่วนประกอบของแพ็กเก็ตไอพี

1. *version* : เป็นค่าตัวเลข 4 บิต บอกเวอร์ชันของมาตรฐานไอพีที่ใช้ โดยปกติมีค่าเป็น 4 ซึ่งหมายถึง IPv4

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

3. *Type of Service* : เป็นส่วนที่บอกการทำงานของแพ็กเก็ตที่ส่งว่าทำหน้าที่อะไร มีทั้งหมด 8 บิต โดย

Bit 0-2 : บอกรายละเอียดการทำงานของแพ็กเก็ตนั้นๆ

111 - Network Control

110 - Internetwork Control

101 - CRITIC / ECP

100 - Flash Override

011 - Flash

010 - Immediate

001 - Priority

000 - Routine

Bit 3 : บอกถึงลักษณะของดีเลย์

0 = Normal Delay - มีดีเลย์ปกติ

1 = Low Delay - มีดีเลย์ต่ำ

Bit 4 : บอกถึงประเภทของทรูพุต

0 = Normal Throughput – มีทรูพุตปกติ

1= High Throughput – มีทรูพุตสูง

Bit 5 : บอกถึงประเภทของความน่าเชื่อถือ

0 = Normal Reliability - มีความน่าเชื่อถือพอประมาณ

1 = High Reliability - มีความน่าเชื่อถือสูง

Bit 6-7 : กันไว้ใช้ในอนาคต

4. *Total Length* : มีขนาด 16 บิต บอกถึงความยาวในดาต้าแกรมของไอพี

5. *Identification field* : เป็นตัวเลข 16 บิต เป็นค่าประจำตัวของไอพีนั้น โดยโฮสต์ที่ส่งเป็นผู้กำหนดและเพิ่มค่าขึ้นหนึ่งเมื่อมีการส่งดาต้าแกรมของไอพีใหม่ ซึ่งใช้ในการประกอบกลับ

6. *Flag* : เป็นตัวเลข 3 bit บอกลักษณะของแพ็กเก็ตว่ามีการแฟร็กเมนต์หรือไม่

Bit 0 : สงวนไว้ ปกติเป็น 0

Bit 1 : 0 = บอกว่าแพ็กเก็ตมีการแตกแพ็กเก็ตย่อย

1 = บอกว่าแพ็กเก็ตไม่มีการแตกแพ็กเก็ตย่อย

Bit 2 : 0 = บอกว่าแพ็กเก็ตนั้นเป็นแพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

1 = บอกว่าแพ็กเก็ตนั้นยังไม่ใช่แพ็กเก็ตสุดท้ายที่ได้จากการแตกแพ็กเก็ตย่อย

7. *Fragment Offset* : เป็นค่าตัวเลข 13 บิต บอกออฟเซตของแฟร็กเมนต์เมื่อเทียบในดาต้าแกรม

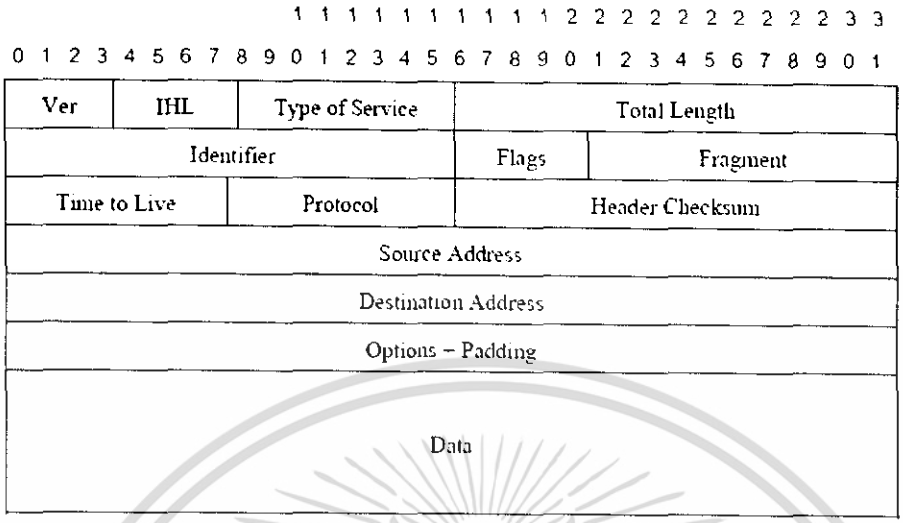
8. *Time To Live (TTL)* : เป็นตัวเลข 8 บิต บอกช่วงเวลาของแพ็กเก็ตที่ยังอยู่ในเครือข่ายได้โดยกำหนดค่าเป็นจำนวนเราท์เตอร์สูงสุดที่ดาต้าแกรมผ่านได้ ซึ่งโดยทั่วไปที่ค่าระหว่าง 32 ถึง 64 และลดค่าลงเรื่อยๆ เมื่อผ่านเราเตอร์ เพื่อเป็นการป้องกันแพ็กเก็ตล้นเครือข่าย

9. *Protocol* : เป็นตัวเลข 8 bit บอกถึงโพรโตคอลที่อยู่เหนือขึ้นไป ว่าเป็นโพรโตคอลระดับสูงกว่าประเภทใด

10. *Header Checksum* : เป็นค่าตัวเลข 32 บิต ใช้ตรวจสอบความถูกต้องของเฮดเดอร์

11. *Source Address* : เป็นค่าตัวเลข 32 บิต บอกถึงไอพีแอดเดรสของเครื่องต้นทาง

12. *Destination Address* : เป็นค่าตัวเลข 32 บิต บอกถึงไอพีแอดเดรสของเครื่องปลายทาง



รูปที่ 2-8 แสดงแพ็กเก็ตไอพี

2.2 ไฟร์วอลล์

คุณสมบัติทั่วไปของไฟร์วอลล์

ไฟร์วอลเป็นเครื่องมือรักษาความปลอดภัยที่ทำงานในเชิงป้องกัน (Protect) ซึ่งทำหน้าที่ในการควบคุมการเข้าถึงเน็ตเวิร์ค (Access Control) โดยอาศัยกฎเป็นพื้นฐาน (Rule based) สำหรับคุณสมบัติแต่ละอย่างของไฟร์วอลนั้นมีรายละเอียดดังนี้

1. การป้องกัน (Protect) : ไฟร์วอลเป็นเครื่องมือที่ใช้ทำงานในเชิงป้องกัน โดยแพ็กเก็ตที่สามารถผ่านเข้า-ออกเน็ตเวิร์คได้นั้น จะต้องเป็นแพ็กเก็ตที่ไฟร์วอลเห็นว่ามีความปลอดภัยแพ็กเก็ตที่ไฟร์วอลเห็นว่าไม่ปลอดภัยหรืออาจจะนำมาซึ่งความไม่ปลอดภัยก็จะถูกกำจัด คือทิ้งไปเสียเฉยๆไม่ส่งต่อ โดยการที่ไฟร์วอลจะตัดสินใจว่าแพ็กเก็ตใดปลอดภัยและแพ็กเก็ตใดไม่ปลอดภัยนั้นจะอยู่บนพื้นฐานของกฎที่ผู้ดูแลไฟร์วอล (Firewall Administrator) เป็นผู้กำหนดไว้ล่วงหน้า ซึ่งเงื่อนไขของกฎเหล่านี้เองทำให้ไฟร์วอลสามารถป้องกันแพ็กเก็ตที่อาจจะส่งผลร้ายไม่ให้ผ่านเข้าไปถึงเครือข่ายคอมพิวเตอร์ได้

2. ควบคุมการแอคเซส (Access Control): "แอคเซส" หมายถึงการที่โฮสต์ใดโฮสต์หนึ่งสามารถสื่อสารข้อมูลที่ต้องการไปยังโฮสต์ปลายทางได้สำเร็จ การแอคเซสในแต่ละระดับจะมีวิธีการแตกต่างกันออกไป ทำให้การควบคุมการแอคเซสสำหรับแต่ละระดับแตกต่างกันไปด้วย ไฟร์วอลจึงมีการทำงานหลายลักษณะตามวิธีที่ไฟร์วอลใช้ควบคุมการแอคเซส

3. กฎพื้นฐาน (Rule Based) : ไฟร์วอลจะควบคุมการแอคเซสโดยอาศัยการเปรียบเทียบคุณสมบัติของแพ็กเก็ตที่จะผ่านไฟร์วอลกับกฎของการแอคเซสที่ได้กำหนดไว้ หากพบว่าไม่มีกฎที่ห้ามไว้ก็จะอนุญาตให้แพ็กเก็ตนั้นผ่านไปได้ หากมีกฎที่ห้ามไว้แพ็กเก็ตนั้นก็จะถูกสกัดกั้นไว้ด้วยวิธีใดวิธีหนึ่ง

ประเภทของไฟร์วอล

ไฟร์วอลสามารถจำแนกประเภทจากลักษณะการทำงานได้ดังนี้

แพ็กเก็ตฟิลเตอริงไฟร์วอล / สกรีนิงเราเตอร์ (Packet Filtering Firewall / Screening Router)

เป็นไฟร์วอลพื้นฐานที่มีความสามารถในการควบคุมแทรฟฟิกโดยอาศัยการตรวจสอบข้อมูลที่ปรากฏอยู่ในแพ็กเก็ต ไฟร์วอลประเภทนี้อาจจะเป็นความสามารถที่เพิ่มเติมมาในเราเตอร์ โดยอาศัยโครงสร้างพื้นฐานที่เราเตอร์มีอยู่ให้ทำหน้าที่มากกว่าการเราต์ (Route คือการจัดเส้นทาง) ให้แพ็กเก็ตไปตามทิศทางที่เหมาะสมเพียงอย่างเดียว แต่จะทำการตรวจสอบเปรียบเทียบกับเงื่อนไขที่กำหนดไว้ก่อนจึงจะทำการเราต์แพ็กเก็ตออกไปก่อนที่จะรู้จักการทำงานของแพ็กเก็ตฟิลเตอริงไฟร์วอลนั้น จะขอกล่าวถึงองค์ประกอบของแพ็กเก็ตเสียก่อนแพ็กเก็ตเป็นหน่วยพื้นฐานของการรับส่งข้อมูลของเน็ตเวิร์คเลเยอร์ การรับส่งข้อมูลแต่ละครั้งของเน็ตเวิร์คเลเยอร์จะส่งข้อมูลออกไปชุดหนึ่ง โดยที่ความยาวของข้อมูลนี้จะมีค่าเท่าใดนั้นจะเป็นไปตามคุณสมบัติของเน็ตเวิร์คเลเยอร์นั้นๆ ข้อมูลแต่ชุดนั้นเรียกว่าแพ็กเก็ต สำหรับโปรโตคอล ทีซีพี/ไอพี นั้นจะใช้ ไอพีเป็นโปรโตคอลหลักในการขนส่งข้อมูลระหว่างโฮสต์โดยไอพีซึ่งอยู่ในอินเตอร์เน็ตเลเยอร์จะส่งข้อมูลลงไปยังเน็ตเวิร์คเลเยอร์ตามลำดับ โดยที่หากขนาดของดาต้าแกรม (ไอพีดาต้าแกรม) ที่จะส่งนั้นสามารถส่งไปได้โดยใช้แพ็กเก็ตเดียว ไอพีก็จะส่งดาต้าแกรมนั้นไปในทันที และแพ็กเก็ตนั้นก็คือข้อมูลของไอพี 1 ดาต้าแกรม แต่หากขนาดของไอพีดาต้าแกรมใหญ่กว่าขนาดของเน็ตเวิร์คเลเยอร์แล้วไอพีก็จะต้องทำการแบ่งส่วนหรือ แฟร็กเมนเตชัน (Fragmentation) คือกระจายดาต้าแกรมออกเป็นส่วนย่อยเสียก่อนแล้วจึงค่อยส่งลงไปที่เน็ตเวิร์คเลเยอร์ ซึ่งในกรณีนี้ ข้อมูล 1 แพ็กเก็ตจะเป็นเพียงส่วนย่อยหรือแฟร็กเมนต์หนึ่งของดาต้าแกรมเท่านั้น ดังนั้นข้อมูล 1 แพ็กเก็ตจึงไม่จำเป็นต้องเป็นข้อมูล 1 ดาต้าแกรมเสมอไป แต่อย่างไรก็ตามแพ็กเก็ตทุกแพ็กเก็ตที่ส่งมาจากไอพีจะมีข้อมูลอย่างน้อยที่สุดคือ ไอพีแอดเดรสต้นทางและปลายทางเสมอ ซึ่งจำเป็นสำหรับให้แพ็กเก็ตนั้นวิ่งออกไปจนถึงที่หมายปลายทางได้หากข้อมูล 1 แพ็กเก็ตนั้นบรรจุครบถ้วนทั้ง ไอพีดาต้าแกรมแล้ว ก็จะทำให้สามารถทราบถึงข้อมูลของโปรโตคอลเลเยอร์ที่สูงขึ้นไปด้วยว่าเป็น ไอซีเอ็มพี, ทีซีพี, ยูดีพี หรือโปรโตคอลอื่นใดที่อาศัยอยู่ใน ไอพีดาต้าแกรม นั้น แต่หากแพ็กเก็ตนั้นไม่สามารถบรรจุข้อมูลได้ครบทั้งดาต้าแกรมแล้ว ก็จะทำให้เพียงแต่ทราบว่าแพ็กเก็ตนั้นเป็นไอพีแพ็กเก็ตเท่านั้น

ข้อมูลที่สำคัญของแพ็กเก็ต

ภายในแพ็กเก็ตแต่ละแพ็กเก็ตนั้นจะประกอบด้วยข้อมูลที่สำคัญซึ่งสามารถนำมาใช้เพื่อเป็นเงื่อนไขสำหรับการควบคุมแทรฟฟิกโดยไฟร์วอลดังนี้

1. ไอพีแอดเดรส ต้นทาง : ไอพีแอดเดรสของต้นทาง เพื่อใช้ในการพิจารณาต้นทางของข้อมูลว่าอยู่ในเงื่อนไขที่อนุญาตหรือไม่

2. ไอพีแอดเดรสปลายทาง : ไอพีแอดเดรส ของปลายทางเพื่อใช้ในการพิจารณาปลายทางของข้อมูลว่าอยู่ในเงื่อนไขที่อนุญาตหรือไม่

3. โปรโตคอล : ระบุโปรโตคอลที่อาศัยอยู่ในไอพีดาต้าแกรม ที่กำลังพิจารณานี้

4. พอร์ตต้นทาง : ระบุพอร์ตต้นทางสำหรับโปรโตคอลที่ใช้พอร์ตคือ ทีซีพี และ ยูดีพี ซึ่งข้อมูลพอร์ตต้นทางนี้ส่วนใหญ่มักจะมีความสำคัญในลำดับรองลงไป และไม่ถูกนำมาใช้ควบคุมแทรฟฟิกมากนัก

5. พอร์ตปลายทาง : ระบุพอร์ตปลายทางที่แพ็กเก็ตนี้ต้องการติดต่อด้วยสำหรับโปรโตคอลที่ใช้พอร์ตเช่น ทีซีพี และ ยูดีพี

6. ข้อมูลสำคัญอื่นๆตามลักษณะของโปรโตคอลเช่น ทีซีพี แฟลก, ไอซีเอ็มพีเมสเซสจ์ เป็นต้น ข้อมูลทั้ง 6 ส่วนนี้จะมีได้อย่างครบถ้วนสมบูรณ์ก็ต่อเมื่อแพ็กเก็ตนั้นมีข้อมูลครบถ้วนทั้งหมดของไอพีดาต้าแกรมหากข้อมูลแพ็กเก็ตนั้นเป็นแฟรกเมนต์ อาจจะทำให้ข้อมูลในส่วนที่ 3 เป็นต้นไปซึ่งอยู่ในโปรโตคอลที่เลเยอร์สูงกว่าไอพีไม่สมบูรณ์ อย่างไรก็ตามไฟร์วอลส่วนใหญ่ทำการติดตั้งใช้งานในเครือข่าย ซึ่งมีขนาดของแพ็กเก็ตที่ใหญ่พอสำหรับรองรับไอพีดาต้าแกรมได้ทั้งหมด จึงมักไม่ค่อยมีปัญหาแต่อย่างใดยกเว้นมีการแฟร็กเมนต์โดยความประสงค์ของไอพีเองจากข้อมูลที่สำคัญของแพ็กเก็ตข้างต้นนี้ จะสามารถนำมาใช้เป็นเงื่อนไขสำหรับควบคุมการผ่านเข้าออกของข้อมูลได้ โดยการพิจารณาข้อมูลทั้งหมดให้เป็นไปตามกฎที่ระบุไว้ ซึ่งเรียกว่าแอคเซสรูล(Access Rules) หรือกฎของการควบคุมการผ่านเข้าออกของแพ็กเก็ต โดยทั่วไปรูปแบบของการแอคเซสรูลเบื้องต้นจะเป็นดังนี้

Source Adress	Destination Address	Protocol	Services(Dest.Port)	Action

โดยที่ข้อมูลทั้งหมดจะเป็นเสมือนตัวแปรที่จะนำมาเปรียบเทียบกับค่าที่ได้ระบุไว้ในแอคเซสรูล ที่ละค่า เงื่อนไขในการเปรียบเทียบของแต่ละตัวแปรจะเป็นตรรกะ **"และ"** ส่วนข้อมูลฟิลด์สุดท้าย หมายถึงสิ่งที่ไฟร์วอลกระทำเมื่อค่าในแพ็กเก็ตนั้นตรงกับเงื่อนไข

ตัวอย่างเช่น

Source Adress	Destination Address	Protocol	Services(Dest.Port)	Action
161.246.5.14	ANY	TCP	80	Accept

นั่นหมายถึงแอคเซสรูลนี้ได้ระบุไว้ว่าจะอนุญาตให้แพ็กเก็ตที่มีต้นทาง ไอพีแอดเดรส161.246.5.14 และปลายทางใดๆ และใช้โปรโตคอล ทีซีพี และหมายเลขพอร์ตปลายทางเท่ากับ 80 ผ่านไฟร์วอลไปได้ หากไฟร์วอลมีแอคเซสรูลนี้เพียงข้อเดียว ก็เท่ากับอนุญาตให้โฮสต์เพียงโฮสต์เดียวคือโฮสต์ที่มี ไอพีแอดเดรส 161.246.5.14 เท่านั้นที่สามารถใช้บริการเอชทีทีพี (ทีซีพี พอร์ต 80) ไปยังโฮสต์อื่นที่อยู่อีกฟากหนึ่งของไฟร์วอลได้นี่เป็นเพียงหลักการพื้นฐานในการควบคุมแทรฟฟิกของแพ็กเก็ตฟิลเตอร์ริ่งไฟร์วอลและไฟร์วอลชนิดนี้โดยทั่วไปจะเรียกว่าสกรีนิงเราเตอร์ (Screening Router) เพราะว่าเป็นการนำเอาเราเตอร์ทั่วไปที่มีความสามารถกำหนดแอคเซสรูลได้มาดัดแปลงใช้ในการควบคุมแทรฟฟิก ซึ่งการกำหนดแอคเซสรูลของแทรฟฟิกทำได้โดยพิจารณาจากข้อมูลจากข้อมูลของแต่ละแพ็กเก็ต แต่เนื่องจากเราเตอร์เป็นอุปกรณ์ที่มีพื้นฐานจากการทำงานในอินเตอร์เน็ตเลเยอร์ ทำหน้าที่เราต์แพ็กเก็ตเป็นหลักโดยพิจารณาจากไอพีแอดเดรส และจะทำการเราต์ไปทีละแพ็กเก็ต ดังนั้นจึงสามารถควบคุมแทรฟฟิกได้ดีในระดับไอพีคือดูจากไอพีแอดเดรส ทั้งต้นทางและปลายทางเท่านั้น สำหรับข้อมูลในส่วนของโปรโตคอลในเลเยอร์สูงขึ้นไป เช่น ทีซีพี, ยูดีพี, ไอซีเอ็มพี นั้น เนื่องจากเราเตอร์มีขีดจำกัดในการรับรู้ข้อมูลในเลเยอร์บนถัดขึ้นไปคือ ทรานสปอร์ตเลเยอร์จึงทำให้สามารถควบคุมแทรฟฟิกโดยระบุเงื่อนไขของโปรโตคอลในทรานสปอร์ตได้อย่างจำกัด คือจะสามารถควบคุมแทรฟฟิกได้เฉพาะเมื่อข้อมูลในทรานสปอร์ตเลเยอร์นั้นจะสามารถบรรจุได้ในแพ็กเก็ตเดียว หากมีการแฟร็กเมนต์และต้องเชื่อมโยงกันระหว่างหลายแพ็กเก็ตแล้วเราเตอร์จะไม่สามารถรับรู้การเชื่อมโยงนั้นได้

ข้อดีของแพ็กเก็ตฟิลเตอร์ริง

1. ราคาถูกเพราะเป็นคุณสมบัติที่มักมีในเราเตอร์อยู่แล้ว อาศัยเพียงการกำหนดแอคเซสรูลที่เหมาะสมเท่านั้น หากยังไม่มีไฟร์วอลลอยู่เลย ก็สามารถใช้เพื่อช่วยป้องกันเน็ตเวิร์กภายในได้ดีพอสมควรในระดับหนึ่ง

2. หากเน็ตเวิร์กภายในไม่ใหญ่มาก และมีการใช้งานอินเตอร์เน็ตอย่างจำกัด ก็สามารถใช้ทดแทนไฟร์วอลได้ทันที

3. การใช้สกรีนนิ่งเราเตอร์ควบคู่กับไฟร์วอลจะเป็นการแบ่งเบาภาระของไฟร์วอลได้มาก หากทำการกำหนดแอคเซน3626 .รูลได้อย่างสอดคล้องกันแล้ว จะทำให้มีการป้องกันที่เข้มแข็ง

4. การป้องกันบางประเภทไม่สามารถป้องกันได้โดยไฟร์วอล จะต้องทำโดยการกำหนดที่เราเตอร์เท่านั้น

ข้อเสียของแพ็กเก็ตฟิลเตอร์ริง

1. การกำหนดแอคเซสรูลทำได้ยาก ไม่มีระบบยูสเซอร์อินเตอร์เฟสเพื่อช่วยในการทำงาน ส่วนใหญ่ก็จะใช้วิธีเทลเน็ตเข้าไปยังเราเตอร์ แล้วป้อนคำสั่งในลักษณะของคอมมานด์ไลน์เข้าไปโดยตรงที่เราเตอร์ ทำให้โอกาสที่จะกำหนดผิดพลาดเนื่องจากการป้อนข้อมูลผิดรูปแบบ(Syntax) เป็นไปได้สูง

2. คำสั่งในการทำงานจะผูกติดกับยี่ห้อของเราเตอร์ ไม่มีมาตรฐานของคำสั่ง หากเปลี่ยนยี่ห้อของเราเตอร์ก็จะต้องศึกษารูปแบบของคำสั่งใหม่

3. ไม่สามารถกำหนดกฎที่ซับซ้อนได้ เนื่องจากขีดจำกัดของเราเตอร์ที่ทำงานโดยพิจารณาครั้งละแพ็กเก็ตเท่านั้น

4. มีความสามารถจำกัด เช่นไม่สามารถบันทึกล็อก (Log) ของแพ็กเก็ตที่ต้องสงสัยไว้ตรวจสอบภายหลังได้

5. เราเตอร์มีกำลังในการประมวลผลจำกัด หากเน็ตเวิร์กมีขนาดใหญ่และมีการสื่อสารข้อมูลหนาแน่น เราเตอร์จะทำงานหนักอยู่แล้ว เมื่อต้องมาทำการประมวลผลแอคเซสรูลด้วยก็อาจจะทำให้ประสิทธิภาพในการเราต์ (Route) แพ็กเก็ตต่ำลงไปมาก และการสื่อสารข้อมูลก็จะติดขัดเป็นคอขวดที่เราเตอร์

เซอร์กิตเลเวลไฟร์วอล / สเตตฟูลอินสเปกชันไฟร์วอล (Circuit-Level Firewall / StatefulInspection Firewall)

การสื่อสารโดยทั่วไปจะเป็นการสื่อสารแบบต่อเนื่อง โต้ตอบกันไปมาระหว่างผู้รับและผู้ส่งอยู่เสมอ โปรโตคอลที่อยู่ในเลเยอร์ที่สูงกว่าอินเตอร์เน็ตเลเยอร์ ไม่ว่าจะเป็นทรานสปอร์ตอย่างเช่น ทีซีพี ยูดีพี หรือเลยไปถึงแอพพลิเคชันเลเยอร์เช่น เอฟทีพี, เอชทีทีพี, เอสเอ็มทีพี ล้วนแล้วแต่จะต้องมีสถานะของการสื่อสาร (State) เสมอ สถานะนี้จะทำให้ทั้งสองฝั่งสามารถสื่อสารกันได้อย่างต่อเนื่องคือทราบว่าตอนนี้กำลังอยู่ ณ จุดใดและจะต้องส่งหรือรับข้อมูลใดเป็นลำดับต่อไป

ความแตกต่างของการพิจารณาข้อมูลแบบแพ็กเก็ตฟิลเตอร์ริงกับสเตตฟูล

อันที่จริงสองเรื่องนี้มิได้ขัดแย้งกันแต่ประการใด แพ็กเก็ตนั้นเป็นการสื่อสารที่เป็นส่วนย่อยของการสื่อสารทั้งหมด ผลของการสื่อสารข้อมูลก็คือผลรวมของการสื่อสารข้อมูลหลายๆแพ็กเก็ตนั่นเอง แต่อย่างไรก็ตามการฟิลเตอร์หรือกรอง โดยพิจารณาทีละแพ็กเก็ตของทุกแพ็กเก็ต3605 .ที่ผ่านเข้าออกนั้นอาจจะมีผลลัพธ์แตกต่างจากการฟิลเตอร์ของในแบบที่มองสถานะและภาพรวมหรือที่เรียกว่าสเตต

ฟูล (Stateful)หากเปรียบเทียบการพิจารณาข้อมูลครั้งละแพ็กเก็ตกับการพิจารณาแบบสเตตฟูลแล้ว ตัวอย่างที่น่าจะช่วยให้เข้าใจได้ง่ายขึ้นคือเปรียบเทียบการสื่อสารข้อมูลทั้งหมดเสมือนภาพยนตร์ แพ็กเก็ตก็จะหมายถึงภาพนิ่งแต่ละภาพที่นำมาต่อรวมกันแล้วเปิดดูอย่างรวดเร็ว ภาพนิ่งเหล่านั้นก็จะกลายเป็นภาพเคลื่อนไหว ดังนั้นการพิจารณาแพ็กเก็ตก็เป็นเสมือนการดูภาพนิ่งทีละภาพ โดยที่แต่ละภาพไม่มีส่วนเกี่ยวข้องกัน ดังนั้นข้อมูลที่จะรับรู้ได้ก็คือข้อมูลที่ปรากฏอยู่บนแต่ละภาพ แต่จะไม่สามารถเซ็นเซอร์เนื้อเรื่องซึ่งเป็นสิ่งที่เกิดขึ้นจากความสัมพันธ์ของภาพหลายๆภาพได้ มีโอกาสเป็นไปได้ว่ากิจกรรมบางชนิดที่หากดูเป็นภาพนิ่งแล้วจะไม่รู้สึกว่าเป็นสิ่งที่ไม่เหมาะสม แต่หากนำภาพนิ่งมาดูอย่างต่อเนื่องเป็นภาพเคลื่อนไหวแล้วก็อาจจะเป็นสิ่งที่ไม่พึงปรารถนาที่จะให้ปรากฏบนภาพยนตร์ก็เป็นได้เซอร์กิตเลเวลไฟร์วอลเป็นไฟร์วอลที่ทำงานโดยที่สามารถเข้าใจสถานะการสื่อสารทั้งกระบวนการ เพราะถือว่าการสื่อสารข้อมูลจะสมบูรณ์ได้นั้นจะต้องมีทั้งการส่งและการรับอย่าสอดคล้องสัมพันธ์กันนั่นเอง หมายถึงหากไฟร์วอลจะสามารถควบคุมการสื่อสารได้จริงก็จะต้องสามารถเข้าใจกระบวนการของการสื่อสารตั้งแต่ต้นจนจบ โดยทั่วไปเราจะเรียกไฟร์วอลแบบนี้ว่า “สเตตฟูลอินสเปกชันไฟร์วอล” (หรือเรียกย่อๆว่าสเตตฟูลไฟร์วอล) เป็นไฟร์วอลที่ทำการควบคุมแทรฟฟิกโดยใช้หลักการของแพ็กเก็ตฟิลเตอร์ริงและการกำหนดแอคเซสรูลเช่นเดียวกับสกรีนิงเราเตอร์ แต่สเตตฟูลไฟร์วอลจะมีความสามารถในการวิเคราะห์และรับรู้ความต่อเนื่องของแพ็กเก็ตในโปรโตคอลในระดับที่สูงขึ้นไปมากว่า ไม่ว่าจะเป็น ทีซีพี, เอฟทีพี, เอชทีทีพีหรือแม้กระทั่งโปรโตคอลในระดับแอพพลิเคชัน ที่จะมีวิธีการกำหนดสเตตของตนเองสเตตฟูลไฟร์วอลเป็นเครื่องมือที่ถูกออกแบบมาเพื่อทำหน้าที่ในการควบคุมแทรฟฟิกโดยเฉพาะไม่ได้เป็นการดัดแปลงการทำงานมาจากเราเตอร์จึงมีความสามารถในการควบคุมแทรฟฟิกการกำหนดแอคเซสรูล การบริหาร รวมไปถึงความยืดหยุ่นของการควบคุมแทรฟฟิก และประสิทธิภาพในการทำงานที่สูงกว่าสกรีนิงเราเตอร์เป็นอย่างมาก โดยทั่วไปหากพูดถึงไฟร์วอลจะหมายถึงไฟร์วอลประเภทนี้เองตามที่ได้กล่าวไว้ข้างต้นว่า ความแตกต่างที่สำคัญของไฟร์วอลทั้งสองชนิดนี้ในแง่ของการตรวจสอบแทรฟฟิกคือ สเตตฟูลไฟร์วอล มีความสามารถในการวิเคราะห์แทรฟฟิกที่ผ่านไปมาในโปรโตคอลที่เลเยอร์สูงขึ้นไปไม่ว่าจะเป็น ทีซีพี, ยูดีพี, ไอซีเอ็มพี ได้อย่างสมบูรณ์ต่างจาก สกรีนิงเราเตอร์ที่สามารถวิเคราะห์ได้เฉพาะเท่าที่จะมีข้อมูลใน 1 แพ็กเก็ตเท่านั้นเพราะบางครั้งแทรฟฟิกที่ผ่านไปมานั้นมีการเชื่อมโยงกันหลายแพ็กเก็ตโดยเฉพาะ ทีซีพี ซึ่งจะมีลำดับของการติดต่อสื่อสารที่สัมพันธ์กันในแต่ละแพ็กเก็ต การพิจารณาแพ็กเก็ตใดแพ็กเก็ตหนึ่งโดยไม่พิจารณาแพ็กเก็ตอื่นที่เกี่ยวข้องก็อาจจะไม่สามารถควบคุมแทรฟฟิกของ ทีซีพี ได้ นอกจากนี้ยังรวมไปถึงการที่สเตทฟูลไฟร์วอลมีความสามารถในการประกอบรวมแฟร็กเมนต์เข้าด้วยกันให้เป็นดาต้าแกรมที่สมบูรณ์ ก่อนหลังจากนั้นจึงนำดาต้าแกรมนั้นมาทำการตรวจสอบ

เปรียบเทียบกับแอคเซสรูลนอกจากการเชื่อมโยงกันของหลายแพ็กเก็ตสำหรับแพ็กเก็ตสำหรับโปรโตคอล ทีซีพี ในทรานสปอร์ตเลเยอร์แล้ว ในแอพพลิเคชันเลเยอร์ก็มีแอพพลิเคชันบางชนิดที่จะต้องอาศัยการพิจารณาแทรฟฟิกอย่างต่อเนื่องเพื่อที่จะนำมากำหนดเป็นแอคเซสรูล ยกตัวอย่างเช่น การทำงานของเอฟทีพี ซึ่งในระหว่างการทำงานของแอพพลิเคชันนั้น โฮสต์ที่เป็นไคลเอนต์จะสามารถกำหนดพอร์ตชั่วคราวขึ้นมาเป็นเซิร์ฟเวอร์พอร์ตใช้สำหรับรับ-ส่งไฟล์ได้ โดยพอร์ตเหล่านี้จะปิดลงเมื่อการรับ-ส่งข้อมูลเสร็จสิ้นสมบูรณ์ซึ่งในกรณีนี้หากไม่มีการพิจารณาแทรฟฟิกที่มีมาก่อนหน้าแล้ว ไฟร์วอลอาจจะถือได้ว่าการเปิดเซิร์ฟเวอร์พอร์ตชั่วคราวของเอฟทีพีไคลเอนท์ นั้นเป็นการเปิดให้บริการใหม่ขึ้นมาก็ได้ ดังนั้นสเตตฟูลไฟร์วอลจึงมีการทำงานที่ค่อนข้างใกล้ชิดกับแอพพลิเคชันเป็นอย่างมาก จะต้องสามารถเข้าใจคุณสมบัติของการสื่อสารข้อมูลของแต่ละแอพพลิเคชันได้ค่อนข้างดี เพราะแอพพลิเคชันที่ใช้งานอยู่ในเน็ตเวิร์กไม่ได้มีเฉพาะแอพพลิเคชันพื้นฐานเท่านั้น มีแอพพลิเคชันอื่นๆอีกมาก แต่หากแอพพลิเคชันใดมีการใช้งานอย่างแพร่หลายและเป็นที่นิยมของผู้ใช้ โดยส่วนใหญ่ผู้ผลิตจะใส่บิวต์อิน(Built-in) การควบคุมแทรฟฟิกสำเร็จรูปมาให้อยู่ในไฟร์วอลเลย

ข้อดีของสเตตฟูลไฟร์วอล

1. ใช้งานง่ายเพราะถูกออกแบบมาทำหน้าที่ของไฟร์วอลโดยเฉพาะ ตรวจสอบแก้ไขแอคเซสรูลได้ง่าย ทำให้ผู้ใช้ไม่ต้องคอยกังวลถึงคำสั่ง และรูปแบบของคำสั่ง ถึงแม้ว่าจะต่างยี่ห้อกันก็สามารถเรียนรู้ใหม่ได้อย่างรวดเร็ว

2. ประสิทธิภาพในการทำงานสูง เนื่องจากออกแบบมาทำหน้าที่ไฟร์วอลโดยเฉพาะ สามารถรองรับแอคเซสรูลที่ซับซ้อนได้ โดยที่ความสามารถในการทำงานไม่ลดลง

3. มีคุณสมบัติเพิ่มเติมให้ใช้ได้มากนอกเหนือจากการควบคุมแทรฟฟิก เช่นสามารถนำไปใช้ร่วมกับระบบการตรวจจับการบุกรุกหรือ IDS (Intrusion Detection System) เพื่อป้องกันการโจมตีได้อัตโนมัติ, สามารถบันทึกข้อมูลเอาไว้กลับมาดูภายหลังได้, สามารถใช้งานร่วมกับระบบป้องกันไวรัสได้เป็นต้น

4. การกำหนดแอคเซสรูลทำได้ง่าย เพราะไฟร์วอลมีความเข้าใจในโปรโตคอลระดับสูง ดังนั้นผู้ใช้อาจจะไม่จำเป็นต้องมีความเชี่ยวชาญในเรื่องเน็ตเวิร์กมาก ก็พอจะใช้งานไฟร์วอลได้โดยกำหนดกฎพื้นฐานของแอพพลิเคชันที่ผู้ใช้รู้จัก มากกว่าการกำหนดกฎโดยใช้ข้อมูลบนแพ็กเก็ตโดยตรง เช่นแทนที่จะต้องกำหนดแอคเซสรูลให้อนุญาต ICMP Time exceed inTransit ให้ผ่านได้ เพื่อจะใช้คำสั่ง Traceroute (ซึ่งโดยทั่วไปผู้ใช้ไม่ทราบว่าโปรแกรมใดใช้โปรโตคอลอะไร แต่จะรู้ว่าตนเองต้องการใช้โปรแกรมหรือแอพพลิเคชันอะไรบ้าง) ก็ระบุใน

ไฟร์วอลว่าอนุญาตให้คำสั่ง Traceroute ทำงานได้หลังจากนั้นไฟร์วอลจึงกำหนดเป็นแอคเซสรูลที่ระบุโปรโตคอลเอง

5. สามารถเพิ่มเติมบริการอื่นได้ เช่น เวอร์ชวลไพรเวตเน็ตเวิร์ค (Virtual Private Network),ทันแนลิง (Tunneling)

6. สามารถเพิ่มเติมความปลอดภัยโดยระบบการตรวจสอบผู้ใช้ (Authenticate) ได้

7. การสื่อสารระหว่างไฟร์วอลกับแอดมินคอนโซล (Adminstration Console : โฮสต์ที่ทำหน้าที่ในการบริหารไฟร์วอล) จะมีความปลอดภัยสูง มีการตรวจสอบสิทธิ์ของผู้ที่เป็นแอดมินรวมทั้งการสื่อสารระหว่างไฟร์วอลกับคอนโซลจะมีการรักษาความปลอดภัยที่เข้มงวด มีการเข้ารหัสเพื่อป้องกันการดักอ่านข้อมูล

ข้อเสียของสเตตฟูลไฟร์วอล

1. มีราคาแพง ถึงแม้ว่าปัจจุบันจะลดลงไปมากแล้วแต่ก็ยังแพงอยู่

2. ในกรณีที่เป็นไฟร์วอลแบบซอฟต์แวร์ที่ทำงานอยู่บนระบบปฏิบัติการทั่วไปเช่น Solaris,Windows NT, Windows 2000 ต่างก็มีความเสี่ยงที่จะถูกเจาะได้เนื่องจากปัญหาของแต่ละระบบu3611 .ฏิบัติการเอง ซึ่งจะสามารถเจาะได้ง่ายกว่าการเจาะเราเตอร์ เพราะรูรั่วของระบบปฏิบัติการมีมากกว่าของเราเตอร์

3. ในกรณีไฟร์วอลเป็นประเภทเน็ตเวิร์คแอพพลิแอนซ์ (Network Applicance) คือออกแบบทั้งซอฟต์แวร์และฮาร์ดแวร์เป็นเครื่องเดียวกันเพื่อทำหน้าที่เป็นไฟร์วอลโดยเฉพาะ ผู้ใช้จำเป็นต้องพึ่งพาผู้ผลิตค่อนข้างมาก หากมีปัญหาอาจจะไม่สามารถแก้ไขโดยการใช้อะไหล่ทดแทนจากที่อื่นได้

2.3 ไอพีเทเบิ้ลส์

โครงสร้างและการทำงานของไอพีเทเบิลส์

ลีนุกซ์สามารถใช้งานเป็นไฟร์วอลล์ได้ตั้งแต่เคอร์เนล 1.1 ซึ่งเป็นเวอร์ชันแรก โดยอลัน ค็อกซ์ (Alan Cox) ใช้ชื่อว่า ipfw (จาก BSD) ต่อมา ลีนุกซ์ 2.0 ได้ถูกพัฒนาและปรับปรุงได้เครื่องมือที่มีชื่อว่า ipfwadm โดยเครื่องมือชิ้นนี้อนุญาตให้ผู้ใช้สามารถควบคุมกฎการกรองแพ็กเก็ตได้ และต่อมา ลีนุกซ์ 2.2ก็ได้สร้างเครื่องมือตัวใหม่ชื่อไอพีเชนส์ (ipchains) ซึ่งเผยแพร่ในปี 1998 โดยรัสตี้ รุสเซล (Rusty Russel)และทีมงาน ทั้งนี้ไอพีเชนส์นี้ถือได้ว่าเป็นพัฒนาการขั้นที่สามของลีนุกซ์ไฟร์วอล จบ

จนกระทั่งในปัจจุบัน ก็มีเน็ทฟิลเตอร์และ ไอพีเทเบิลส์ซึ่งถือว่าเป็นพัฒนาการขั้นที่สี่ของ ลีนุกซ์ ไฟร์วอลเน็ทฟิลเตอร์นั้นเป็นชื่อใหม่ของโค้ดที่ทำหน้าที่เป็น แพ็กเก็ต handler (stateful inspection) ใน ลีนุกซ์ เคอร์เนล 2.4 (ที่จริงคือเวอร์ชัน 2.3.15 และเวอร์ชันต่อๆ มา) ซึ่งได้ถูกออกแบบและปรับปรุงใหม่จากเวอร์ชันก่อนหน้านี้ เป็นเรื่องที่น่ายินดีคือเน็ทฟิลเตอร์นั้นสามารถทำงานย้อนหลังร่วมกับไอพีเชนและipfwadm ได้ และคำสั่งในการเรียกใช้งานคือไอพีเทเบิลส์

ความแตกต่างระหว่างไอพีเทเบิลส์ และ ไอพีเชนส์

- ชื่อของ built-in chain (ประกอบไปด้วย INPUT, OUTPUT, FORWARD) เปลี่ยนจากตัวอักษรเล็ก (lowercase) เป็นตัวอักษรใหญ่ (uppercase)
- การใช้งานที่ต้องระบุพอร์ต ทั้ง ทีซีพี และ ยูดีพี นั้น ต้องใช้คำว่า --source-port หรือ --sport (--destination-port หรือ --dport) และต้องใช้ตามหลังจาก -p tcp หรือ -p udp
- TCP -syn flag เปลี่ยนเป็น --syn และต้องใช้ร่วมกับ -p tcp
- target จาก DENY เปลี่ยนเป็น DROP
- chain ที่ไม่มี กฎ ใดๆ เลยก็สามารถทำงานได้
- การทำ zeroing built-in chain จะทำให้ byte counter ถูกล้างค่าไปด้วย
- ชื่อของ chain ยาวสูงสุดได้ 31 ตัวอักษร
- MASQ เปลี่ยนเป็น MASQUERADE และมีรูปแบบการใช้งานเปลี่ยนไป รวมทั้ง REDIRECT ก็มีการเปลี่ยนแปลงรูปแบบใหม่

ก่อนที่ใช้งานไฟร์วอลล์ตัวใหม่นี้ได้ ให้ล็อกอินเป็น root และทดลองพิมพ์คำว่า iptables ดูว่าคำสั่งนี้มีอยู่หรือไม่ ถ้าไม่มี ต้องดาวน์โหลดจาก http://www.netfilter.org โดยให้ดาวน์โหลดเวอร์ชันล่าสุดมา ในที่นี้คือ iptables-1.2.9.tar.bz2 จากนั้นให้ขยายไฟล์และติดตั้งดังคำสั่งด้านล่าง

```
tar -jxvf iptables-1.2.9.tar.bz2
cd iptables-1.2.9
make
make install
```

ปรับแต่งเคอร์เนลหลังจากที่มีคำสั่ง iptables อยู่ในระบบแล้ว บางระบบที่มีเคอร์เนลต่ำกว่า 2.4 จำเป็นต้องคอมไพล์เคอร์เนลใหม่ เพื่อให้ใช้งานiptablesได้ ทั้งนี้โครงงานนี้ได้พัฒนาโดยใช้เคอร์เนลเวอร์ชั่น 2.4.23 ปัจจุบันซึ่งสามารถดาวน์โหลดได้จาก http://www.kernel.org

หลังจากที่ดาวน์โหลดมาแล้วให้ขยายไฟล์ไปไว้ที่ /usr/src/linux จากนั้นใช้คำสั่ง make menuconfig หรืออาจใช้ make xconfig ในกรณีที่ได้ติดตั้ง XWindows ไว้แล้ว โดยต้องมั่นใจว่าได้ enable

option ต่างๆ ด้านล่างนี้ ภายใต้ Networking options

<*> Packet socket

[*] Network packet filtering (replaces ipchains)

<*> Unix domain sockets

[*] TCP/IP networking

[*] IP: advanced router

[*] IP: policy routing

[*] IP: use netfilter MARK value as routing key

[*] IP: fast network address translation

[*] IP: use TOS value as routing key

และภายใต้เมนู "IP: Netfilter Configuration --->" ให้ enable ทุกออปชันเพื่อให้ใช้งาน netfilter ได้เต็มประสิทธิภาพ แต่มีอีกจุดหนึ่งที่ห้าม enable ภายใต้ "Networking options" คือ

[ ] IP: TCP Explicit Congestion Notification support

เพราะออปชันนี้ จะทำให้บาง แพ็กเก็ต ที่ออกจาก ลีนุกซ์ ถูกเซ็ตบิต ECN ไปด้วย ซึ่งจะทำให้ไม่สามารถสื่อสารกับ internet router บางตัวได้จากนั้นก็ให้คอมไพล์และติดตั้งเคอร์เนลโดยใช้คำสั่งดังนี้

คำสั่งที่ใช้ในการคอมไพล์เคอร์เนล

```
make dep
make clean
make bzImage
make modules
#รันคำสั่งนี้ในกรณีที่คอมไพล์เคอร์เนลแบบเลือก module ด้วย
make modules_install
#รันคำสั่งนี้ในกรณีที่คอมไพล์เคอร์เนลแบบเลือก module ด้วย
cp /usr/src/linux/arch/i386/boot/bzImage /boot/bzimage-2.4.23
#การ copy bzImage นั้น ขึ้นอยู่กับสถาปัตยกรรมของเครื่องคอมพิวเตอร์นั้นๆ ซึ่งในที่นี้ใช้ i386
cp /usr/src/linux/System.map /boot/System.map-2.4.23
ln -s /boot/System.map-2.4.23 .boot/System.map
#ถ้าได้รับ error ให้ลบไฟล์ /boot/System.map ทิ้งก่อน
```

จากนั้นให้แก้ไข /etc/lilo.conf เพื่อเพิ่ม configuration ใหม่ ตัวอย่างด้านล่างแสดงเรดแฮท (Red Hat) ลีนุกซ์ ที่มีเคอร์เนลเดิมเป็น 2.4.18 และเคอร์เนลใหม่เป็น 2.4.23

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
message=/boot/message
default=linux-2.4.23
image=/boot/vmlinuz-2.4.23
label=linux-2.4.23
read-only
root=/dev/hda1
image=/boot/vmlinuz-2.4.18
label=linux-2.4.18
read-only
root=/dev/hda1
38
```

หลังจากนั้นรันคำสั่ง lilo -v และรีบูตใหม่ ถ้าไม่มีข้อผิดพลาดใดๆ ก็จะได้เคอร์เนลของ ลีนุกซ์เวอร์ชันล่าสุดที่สนับสนุนการทำงานของ iptables อย่างเต็มที่

รูปแบบการใช้งาน iptables เบื้องต้น

iptables จะมีรูปแบบการใช้งานดังนี้คือ

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

โดย กฎ ที่เขียนขึ้นจะเป็นเป็นตัวบอกเคอร์เนลว่าให้กระทำ action อย่างไร ในกรณีที่พบ แพ็กเก็ต ตรงตามที่ระบุไว้

• [table] หมายถึง ตารางหรือ table ที่ต้องการระบุ เช่น iptables -t nat หมายถึงให้ทำงานกับ nat table

ในกรณีที่ไม่ได้ระบุตาราง iptables จะถือว่าคำสั่งดังกล่าวระบุถึง filter table โดยอัตโนมัติ

• <command> จะเป็นตัวสั่งให้ iptables ทำในสิ่งที่ต้องการ เช่น iptables -A INPUT ซึ่งหมายถึงให้สร้าง กฎ ต่อท้าย INPUT chain ใน filter table

• <match> เป็นส่วนที่ใช้ตรวจสอบว่า แพ็กเก็ต มีข้อมูลตรง (match) กับที่ระบุไว้หรือไม่ เช่น มี source ip address เป็น 1.2.3.4

• <target/jump> เป็นตัวระบุว่าเมื่อเจอ แพ็กเก็ต ที่ match ก็จะกระทำ (action) ตามที่ระบุไว้ เช่น ถ้าแพ็กเก็ต ใดมี source ip address เป็น 1.2.3.4 ให้ DROP แพ็กเก็ต นั้นทิ้งไป

Table

iptables สามารถทำงานได้กับตาราง(table) 3 ตารางหลัก สามารถระบุตารางได้โดยใช้ออปชัน -t ตามด้วยชื่อ Table คือ

1. Filter table ใช้สำหรับกรอง แพ็กเก็ต มี 3 built-in chain คือ INPUT, OUTPUT, FORWARD ซึ่งจะได้อธิบายรายละเอียดต่อไป

2. Nat table ใช้สำหรับการแปลงแอดเดรส (Network Address Translation) มี 3 built-in chain คือ PREROUTING, POSTROUTING, OUTPUT ซึ่งรายละเอียดจะได้อธิบายต่อไป

3. Mangle table เป็นตารางที่ใช้เปลี่ยนแปลงหรือแก้ไข แพ็กเก็ต เช่น เปลี่ยนค่า TTL, MARK ซึ่งปกติจะใช้ในการทำ routing ที่มีความซับซ้อนสูง มี 2 built-in chain คือ PREROUTING chain (ใช้แก้ไข แพ็กเก็ต ก่อนที่จะเข้าสู่ไฟร์วอลล์และก่อนเข้าสู่ routing decision) และ OUTPUTchain (ใช้แก้ไข แพ็กเก็ต ที่ถูกสร้างโดยไฟร์วอลล์ก่อนที่มันจะถูกส่งไปยัง routing decision)ทั้งนี้ไม่สามารถทำ network address translation หรือ masquerading ที่ table นี้ได้ และโครงงานนี้ไม่ได้ศึกษาในส่วนนี้

Command

• -A เพิ่ม กฎ ใหม่ต่อท้าย chain (Append rule) เช่น

```
# iptables -A INPUT -p ALL -i eth0 -j ACCEPT
```

• -D ลบ กฎ (Delete rule) เช่น

```
# iptables -D INPUT --dport 80 -j DROP
```

• -I เพิ่ม กฎ ใหม่ ใน chain (Insert rule) เช่น

```
# iptables -I OUTPUT -p ALL -s 127.0.0.1/32 -j ACCEPT
```

• -R แทนที่ กฎ เดิม ด้วย กฎ ใหม่ (Replace rule)

• -L แสดง กฎ ทั้งหมดใน chain (ถ้าไม่ระบุ chain จะแสดง กฎ ทั้งหมดใน filter table ทั้งสาม built-in chain) เช่น

```
# iptables -L
# iptables -L -t nat
# iptables -L INPUT
```

• -F ลบ กฎ ทั้งหมดใน chain ทิ้ง เช่น

```
# iptables -F INPUT
# iptables -F mychain
```

• -Z ใช้ reset byte counter สำหรับทุก กฎ ใน chain ที่กำหนด เช่น

```
# iptables -Z INPUT
```

• -N ใช้สร้าง chain ใหม่ เช่น

```
# iptables -N mychain
```

• -X ลบ chain ที่ไม่มี กฎ ซึ่งสามารถลบ user-defined chain ที่ไม่มี กฎ ได้ แต่ไม่สามารถลบ builtinchain ได้ เช่น

```
# iptables -X emptychain
```

• -P เปลี่ยน default policy ของ chain ค่าที่ใช้ได้คือ ACCEPT, DROP ทั้งนี้ค่านี้มีความสำคัญอย่างมากเพราะหาก แพ็กเก็ต ถูกส่งเข้ามาใน chain แล้ว และไม่ match กับ กฎ ใดๆ เลย แพ็กเก็ต นั้นก็ต้องถูกตัดสินใจโดย policy ของ chain นั้นๆ เช่น

```
# iptables -P FORWARD DROP
```

ซึ่งหาก แพ็กเก็ต ถูกส่งเข้ามายัง FORWARD chain และไม่ match กับ กฎ ใดๆ ใน FORWARD chain นี้เลย มันก็จะถูก DROP ทันที

• -E ใช้เปลี่ยนชื่อ chain ใหม่ เช่น

```
# iptables -E myoldchain mynewchain
```

การใช้ command ด้านบนนั้นสามารถใช้ร่วมกับออปชันบางอย่างได้ คือ

• -V, --verbose ใช้ร่วมกับ -L, -A, -I, -D, -R เพื่อให้แสดงจำนวน byte ที่ match กับ กฎ ออกมาด้วย (หน่วยเป็นได้ทั้ง K(x1,000),M(x1,000,000),G(x1,000,000,000)) เช่น

```
# iptables -L -v
```

• -x, --exact ใช้ร่วมกับ -L และ -v เพื่อให้แสดงจำนวน แพ็กเก็ต และจำนวนของ byte ข้อมูลที่ match โดยไม่ให้แสดงผลในหน่วยของ K,M,G เช่น

iptables -L OUTPUT -v -x

• -n, --numeric ใช้ร่วมกับ -L เพื่อสั่งให้ iptables แสดงข้อมูลไอพีแอดเดรสและ พอร์ต เป็นตัวเลขเท่านั้น เช่น

iptables -L OUTPUT -n

• --line-numbers ใช้ร่วมกับ -L เพื่อแสดงเลขบรรทัดของ กฎ ซึ่งตัวเลขที่แสดงนี้จะสามารถใช้ได้กับคำสั่ง แทรก กฎ ที่ระบุเป็นลำดับที่ของ กฎ เช่น

iptables -L --line-numbers

• --modprobe=*command* เพื่อโหลด module ที่เกี่ยวข้อง

ฟังก์ชั่นแมทช์ (Match)

การตั้งเงื่อนไขของการ match นั้นจะต้องอาศัยความเข้าใจในเรื่อง IP, ทีซีพี, ยูดีพี, และไอซีเอ็มพี มาบ้างพอสมควร จึงจะสามารถตั้งเงื่อนไขที่เหมาะสมและตรงตามความต้องการได้ ซึ่งมีรายละเอียดดังนี้

• การระบุ source, destination IP address

สามารถระบุ source ip address ของ แพ็กเก็ต โดยใช้ -s หรือ --source หรือ --src และสำหรับdestination ip address ก็ใช้ -d หรือ --destination หรือ --dst การระบุไอพีแอดเดรสนั้นสามารถทำได้ 4 แบบด้วยกัน คือ

1. ใช้ชื่อเต็มแทน เช่น localhost หรือ www.ce.kmitl.ac.th
2. ระบุไอพีแอดเดรสโดยตรง เช่น 127.0.0.1 หรือ 161.246.5.14
3. ระบุเป็น group ของไอพีแอดเดรส เช่น 161.246.5/24 ซึ่งหมายถึงไอพีแอดเดรสตั้งแต่ 161.246.5.0 – 161.246.5.255
4. หรืออาจจะใช้ 161.246.5.0/255.255.255.0 แทน 161.246.5.0/24 ได้

• การทำ Inversion

ในบางกรณีนั้นหากต้องการระบุเป็น inverse เช่น อนุญาตให้ทุกไอพียกเว้นไอพีที่ระบุไว้ ซึ่งการใช้คำสั่งดังกล่าวสามารถทำได้โดยใช้เครื่องหมาย ! นำหน้า argument ที่ต้องการ (เครื่องหมาย ! หมายถึง NOT) เช่น -p ! TCP ซึ่งจะ match กับโพรโตคอลทุกๆ ตัวที่ไม่ใช่ ทีซีพี หรือ -s ! localhost ซึ่งหมายถึง แพ็กเก็ต ที่มี source ip address อื่นๆ ยกเว้น localhost (127.0.0.1)

• การระบุโพรโตคอล

สามารถระบุโพรโตคอลที่ต้องการได้ดังนี้คือ ทีซีพี, ยูดีพี, ไอซีเอ็มพี หรือสามารถใช้ตัวเลขแทนได้ (สำหรับ *NIX อ้างอิงได้จาก /etc/protocols) และยังสามารถใช้ได้ทั้งตัวอักษรเล็กหรือใหญ่

(ใช้ได้ทั้ง tcp และ TCP) เช่น -p TCP หรือ -p ! tcp

• การระบุ interface

-i หรือ --in-interface ตามด้วยชื่อ interface ใช้เพื่อระบุ incoming interface ซึ่งหมายถึงว่า แพ็กเก็ต ที่จะ match กับ กฎ นี้ต้องเข้ามาจาก interface ที่กำหนด เช่น -i eth0 หมายความว่า ทุก แพ็กเก็ต ที่เข้ามาทาง eth0 จะ match กับ กฎ นี้ ทั้งนี้ชื่อ interface ที่สามารถใช้ได้นั้น สามารถตรวจสอบได้โดยใช้คำสั่ง ifconfigและ -o หรือ --out-interface ตามด้วยชื่อของ interface ใช้เพื่อระบุ outgoing interface ซึ่งหมายถึงว่า แพ็กเก็ต ที่จะ match กับ กฎ นี้ กำลังจะเดินทางผ่าน interface ที่ระบุไว้ เช่น -o eth1 หรือ -o !eth1

ข้อสังเกต

o สำหรับ INPUT chain นั้นไม่มี output interface ดังนั้นหากใช้ -o ร่วมกับ INPUT chain ก็จะไม่มี แพ็กเก็ต ใดที่ match กับ กฎ นี้เลย

o ทำนองเดียวกันกับ OUTPUT chain ที่ไม่มี input interface ดังนั้นหากใช้ -i ร่วมกับ OUTPUT chain ก็ไม่มีประโยชน์อันใด

o FORWARD chain มีได้ทั้ง input และ output interface

o หากระบุ interface ที่ไม่มีอยู่จริง ก็จะไม่มี แพ็กเก็ต ใดที่ match กับ กฎ นั้นเลย

o หากใช้เครื่องหมาย + ร่วมกับ interface เช่น ppp+ นั้นจะหมายถึงทุกๆ ppp interface เช่น ppp0, ppp1

• fragment packet

ในการส่งข้อมูลใน ip network นั้นเป็นเรื่องปกติที่จะเกิดการ fragment ของ แพ็กเก็ต เนื่องจากขนาดของ แพ็กเก็ต มีขนาดใหญ่เกินไปที่จะส่งไปในครั้งเดียว จำเป็นต้องมีการแบ่ง แพ็กเก็ตออกเป็นหลายๆ ชิ้นทยอยส่งไป ซึ่งเรียกกันว่าการทำ fragment โดยเครื่องปลายทางจะทำหน้าที่ประกอบ fragment packet รวมกันเป็น แพ็กเก็ต ที่สมบูรณ์ดังเดิมข้อมูลที่เป็น fragment packet นั้นจะมี header ที่สมบูรณ์แค่ แพ็กเก็ต แรกเท่านั้น ตัว แพ็กเก็ต ที่ตามมาจะมีแค่ header บางส่วนคือ ไอพีแอดเดรสเท่านั้น ไม่มีข้อมูลของโพรโตคอลแนบมาด้วย ดังนั้นการตรวจสอบข้อมูล header ของ ทีซีพี, ยูดีพี, ไอซีเอ็มพี จึงไม่สามารถทำได้ใน แพ็กเก็ต ที่สองเป็นต้นมาหากใช้ NAT บรรดา fragment packet จะถูกประกอบเข้าด้วยกันจนสมบูรณ์ก่อนที่ แพ็กเก็ต จะเข้าไปถึง packet filtering ดังนั้นจึงไม่มีความจำเป็นที่จะต้องกังวลเกี่ยวกับ fragment packetดังนั้นถ้าไม่ได้ใช้ NAT ก็ควรทำความเข้าใจไว้ว่า iptables มีกระบวนการในการทำงานกับfragment packet อย่างไร หลังจากที่ fragment packet แรกผ่านเข้ามาแล้ว iptables สามารถตรวจสอบได้ว่าจะอนุญาตให้ผ่านหรือไม่ ในขณะที่ fragment packet ที่สองและหลังจากนั้นที่

ตามมานั้น จะไม่สามารถmatch กับ กฎ ใดๆ เลย เช่น -p TCP --sport www หรือแม้แต่ -p TCP --sport ! www อย่างไรก็ตาม สามารถเขียน กฎ ให้ตรวจสอบทั้ง fragment packet ตัวที่สองและหลังจากนั้นที่ตามมาได้ด้วยการใช้ -f หรือ --fragment ทั้งนี้อาจจะเขียนในทางตรงข้ามคือไม่ต้องตรวจสอบ fragment packet ที่สองและหลังจากนั้นโดยใช้ ! -f ก็ได้ทั้งนี้โดยปกติแล้วมักจะปล่อยให้ fragment packet ผ่านไป เนื่องจากถ้าสามารถ DROP ตัวfragment packet ตัวแรกได้แล้ว มันก็ไม่สามารถถูกประกอบที่เครื่องปลายทางได้ แต่ทั้งนี้ fragment packetที่ถูกปล่อยไปดังกล่าวอาจจะทำให้เครื่องที่ได้รับ hang หรือ crash ได้ หรืออาจจะเกิดการโจมตีแบบDenial of Service โดยใช้ fragment packet ได้

• TCP extension

ถ้ามีการเรียกใช้ -p tcp ตัว TCP extension ก็จะถูกโหลดมาใช้งานโดยอัตโนมัติ โดยมีออปชั่นให้เลือกใช้งานดังนี้

o --tcp-flags *mask flags* : mask นั้นหมายถึง flag ที่ต้องการตรวจสอบ และ flag เป็นตัวที่บ่งชี้ว่า flag ใดต้องถูก set บ้าง เช่น # iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP โดย ALL นั้นหมายถึงทุกๆ flag (SYN,ACK,FIN,RST,URG,PSH) และถ้า flag SYN,ACK ถูก set พร้อมกันก็ให้ drop packet นั้นทิ้งไป นอกจากนี้ยังสามารถใช้ NONE ซึ่งหมายถึงไม่มี flag ใดถูก set ได้

o --syn เป็นสัญลักษณ์ย่อของ --tcp-flags SYN,RST,ACK SYN

o --source-port หรือ --sport สามารถใช้ได้ทั้งตัวเลขและตัวอักษร (อ้างอิงจากไฟล์ /etc/services) และระบุเป็น พอร์ต เดียว หรือช่วงของ พอร์ต ได้เช่น --sport 21:25 หมายถึง พอร์ต 21 - 25 , --sport 25: หมายถึงพอร์ตที่มากกว่าหรือเท่ากับ 25 , --sport :25 หมายถึงพอร์ตที่น้อยกว่าหรือเท่ากับ 25

o --destination-port หรือ --dport มีรูปแบบการใช้งานเช่นเดียวกันกับ --sport

o --tcp-option ใช้ตรวจสอบ TCP option ว่าตรงกับเลขที่ระบุไว้หรือไม่

• อธิบาย flag ของ ทีซีพี เพิ่มเติม

การเชื่อมต่อโดยใช้ ทีซีพี นั้น ผู้ที่เริ่มสร้าง connection จะเป็นผู้ส่ง SYN แพ็กเก็ต มายังเครื่องปลายทาง ดังนั้นหากไม่ต้องการให้ให้เครื่องใดเป็นผู้เริ่มสร้างการติดต่อก็สามารถ block ไอพีดังกล่าวได้ โดยใช้ --syn เช่น -p TCP -s x.x.x.x --syn หากยังไม่เข้าใจรูปแบบการเชื่อมต่อแบบ ทีซีพี นี้แล้ว ก็เป็นการยากที่จะสร้าง กฎ สำหรับ iptables ดังนั้นจึงขอแนะนำให้ไปศึกษาหลักการทำงานเบื้องต้นของทั้ง ทีซีพี, ยูดีพี, ไอซีเอ็มพี มาก่อน

• UDP extension

คล้ายกันกับ ทีซีพี ตัว ยูดีพี extension มีออปชันให้เลือกใช้เพียงแค่ 2 อย่างเท่านั้นคือ --source-port (--sport) และ --destination-port (--dport) โดยต้องระบุ -p udp ด้วย

• ICMP extension

โดยการระบุ -p icmp ก็สามารถใช้งาน ICMP extension ได้ โดยมีออปชันให้เลือกคือ --icmp-type เช่น --icmp-type host-unreachable (หรือใช้เลข 3 แทนได้) นอกจากนี้ยัง สามารถระบุ type/code ได้ เช่น 3/3 ซึ่งหมายถึง port unreachable

• Match Extension

เป็น netfilter package ที่อยู่ในช่วงทดลองใช้ รูปแบบการใช้งานให้ใช้ -m แล้วตามด้วย match ที่ต้องการ เช่น -m mac ทั้งนี้มีออปชันให้เลือกใช้งานดังต่อไปนี้

o mac รูปแบบการใช้งาน: -m mac หรือ --match mac ใช้ตรวจสอบ source MAC address ว่าตรงกับค่าที่ระบุไว้หรือไม่ มีประโยชน์สำหรับ PREROUTING, INPUT chain โดยมีออปชันให้ใช้งานคือ --mac-source เช่น --mac-source 00:55:81:CC:42:FF

o limit

รูปแบบการใช้งาน: -m limit หรือ --match limit ใช้เพื่อจำกัดจำนวนของการ match ที่อาจจะมากเกินไป เป็นประโยชน์สำหรับ กฎ ที่วางไว้ตอนท้ายสุดของ chain (ใช้ร่วมกันกับ DROP policy) ซึ่งส่วนใหญ่เป็น กฎ ที่ใช้เก็บข้อมูลลงล็อกไฟล์ ซึ่งถ้าผู้บุกรุกส่ง แพ็กเก็ต ที่ไม่เข้าข่าย กฎ ใดๆ ใน chain จนกระทั่งมาถึง กฎ ที่ทำหน้าที่เก็บล็อกนี้ ถ้า แพ็กเก็ต ที่เข้ามามีจำนวนมากก็อาจจะทำให้ฮาร์ดดิสก์เต็มได้ ดังนั้นจึงต้องใช้จำกัดจำนวนในการเก็บข้อมูลลงล็อก ซึ่งมีออปชันให้ใช้งานดังนี้คือ --limit ตามด้วยตัวเลข ซึ่งบอกถึงจำนวนครั้งสูงสุดของการ match กับ กฎ ที่ยินยอมต่อ 1 วินาที เช่น --limit 5/s ทั้งนี้ยังสามารถใช้หน่วยเวลาอื่นได้ เช่น /second /minute /hour /day เช่น

-m limit --limit 3/minute

--limit-burst

ตามด้วยตัวเลข แสดงถึงจำนวนมากที่สุดของ แพ็กเก็ต ที่ matchกับ กฎ นี้ ค่า default ของมันคือ 5

ตัวอย่างการใช้ --limit และ --limit-burst ร่วมกัน เช่น

iptables -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG

โดยส่วนใหญ่นิยมวาง กฎ นี้ไว้เป็น กฎ สุดท้ายใน chain โดยเฉพาะ chain ที่มี default policy เป็น DROPเพื่อเป็นตัวเก็บหลักฐานว่ามี แพ็กเก็ต ใดที่ถูกส่งมาและไม่ผ่านการตรวจสอบจาก กฎ และกำลังจะถูก DROP โดย default policy โดย กฎ ด้านบนนี้กำหนดจำนวน match สูงสุดไว้ 3 ครั้งต่อนาที

ซึ่งแสดงว่าใน 1 นาทีจะมีการบันทึกล็อกได้สูงสุด 3 ครั้งเท่านั้น และมีค่า burst เท่ากับ 3 ซึ่งอธิบายได้ว่า ถ้าสมมุติมี แพ็กเก็ต ที่ match กับ กฎ นี้ 3 ครั้งภายใน 2 วินาที และถึงแม้ว่าจะมี แพ็กเก็ต ที่ match ส่งมาอีกก็จะไม่มีการบันทึกลงล็อกแต่อย่างใด และจะต้องรอไปอีก 1 นาทีจึงจะมีการเริ่มการบันทึกล็อกใหม่อีกครั้ง ซึ่งมีประโยชน์ในกรณีที่มีคนต้องการส่ง แพ็กเก็ต เพื่อ flood log หรือทำให้ล็อกในเครื่องเต็ม ทั้งนี้นิยมใช้ร่วมกับ --log-level (อ้างอิงค่าจาก level ใน syslogd เพื่อกำหนดค่า level สำหรับ syslog) และ --log-prefix เพื่อใช้อธิบายเพิ่มเติม เช่น

iptables -A INPUT -m limit --limit 3/minute --limit-burst 3 -j LOG --log-level DEBUG --log-prefix

"Packet died: "

นอกจากนี้ยังใช้ป้องกันการโจมตีแบบ Denial of Service เช่น SYN flood ได้ด้วย เช่น

iptables -A FORWARD -p tcp --syn -m limit --limit 1/s -j ACCEPT

ใช้ป้องกันการโจมตีแบบ Ping of Death

#iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j ACCEPT

โดยปกติมักใช้วิธี drop ไอซีเอ็มพี แพ็กเก็ต ทิ้งทั้งหมด เพราะถือว่า ไอซีเอ็มพี แพ็กเก็ต เป็นข้อมูลที่มีอันตรายยิ่งและสามารถปลอมแปลงได้ง่ายหรือใช้ป้องกันการถูก scan

iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT โดยปกติไม่นิยมใช้วิธีนี้นัก เพราะมีทางเลือกที่ดีกว่า คือการตรวจสอบสอบจาก state match ว่าเป็นการเชื่อมต่อใหม่หรือไม่ (state = new) ถ้าใช่และ SYN bit ไม่ถูก set ตัว แพ็กเก็ต นั้นก็จะถูก DROP ทิ้งไป

o owner

รูปแบบการใช้งาน: -m owner หรือ --match ownerใช้ตรวจสอบลักษณะของ แพ็กเก็ต ว่าใครเป็นผู้สร้าง ซึ่งสามารถใช้ได้กับ OUTPUT chain เท่านั้น และใช้ได้กับบาง แพ็กเก็ต ที่มีเจ้าของ เช่น ไอซีเอ็มพี แพ็กเก็ต นั้นใช้ไม่ได้เพราะไม่มีเจ้าของ มีออปชันให้ใช้งานดังนี้คือ

--uid-owner *userid*

ใช้ตรวจสอบว่า แพ็กเก็ต ถูกสร้างโดย user id ที่ระบุไว้หรือไม่ (ใช้ตัวเลขแทนuserid เท่านั้น)

--gid-owner *groupid*

ใช้ตรวจสอบว่า แพ็กเก็ต ถูกสร้างโดย user ที่อยู่ใน group id ที่ระบุไว้หรือไม่(ใช้ตัวเลขแทน groupid เท่านั้น)

--pid-owner *processid*

ใช้ตรวจสอบว่า แพ็กเก็ต ถูกสร้างขึ้นจาก process ที่มี process id ตรงกับที่ระบุไว้หรือไม่

--sid-owner *sessionid*

ใช้ตรวจสอบว่า แพ็กเก็ต ถูกสร้างโดย process ที่อยู่ใน session group ที่กำหนดไว้หรือไม่

o unclean

รูปแบบการใช้งาน: -m unclean หรือ --match uncleanเป็นโมดูลที่อยู่ในระหว่างการทดลองใช้งาน นอกจากนี้ยังไม่มีออปชันสำหรับใช้งานและโปรดระมัดระวังหากจะนำไปใช้กับเครื่องที่ต้องการความปลอดภัย เนื่องจากอาจจะยังมีข้อบกพร่องของโปรแกรมอยู่โดย แพ็กเก็ต ที่เข้าข่าย unclean คือ

แพ็กเก็ต ที่มี header ของ ไอซีเอ็มพี/ทีซีพี/ยูดีพี สั้นหรือไม่สมบูรณ์

ทีซีพี, ยูดีพี แพ็กเก็ต ที่มี source หรือ destination ip address เป็นศูนย์

ทีซีพี แพ็กเก็ต ที่ใช้ flag ผสมกันแบบผิดปกติ

แพ็กเก็ต ที่ใช้ ทีซีพี option, IP option เกินความยาวที่กำหนดไว้ หรือมีความยาวของออปชันเป็นศูนย์

fragment แพ็กเก็ต ที่ไม่สมบูรณ์ ทั้งด้านความยาวและค่า offset ที่เหลื่อมซ้อนกัน เช่น Ping of Death

o multiport ใช้ร่วมกับ --sport หรือ --dport ในกรณีที่ต้องการระบุ พอร์ต จำนวนมากกว่าหนึ่ง เช่น

-m multiport -p tcp --sport 25,80,53

• The State Match

รูปแบบการใช้งาน: -m state หรือ --match state เป็นโมดูลที่ใช้ประโยชน์ได้เป็นอย่างดี มีออปชันให้ใช้งานดังนี้

o NEW

รูปแบบการใช้งาน: -m state --state new หรือ --match state --state new หมายถึง แพ็กเก็ต ที่เป็นตัวสร้าง connection ใหม่

o ESTABLISHED

รูปแบบการใช้งาน: -m state --state established หรือ --match state --state established หมายถึง แพ็กเก็ต ที่เกี่ยวข้องกันกับ connection ที่สร้างไว้แล้ว เช่น echo-reply แพ็กเก็ตหรือ แพ็กเก็ต ที่ส่งข้อมูลออกไปจาก web server เมื่อมี request web service เข้ามา

o RELATED

รูปแบบการใช้งาน: -m state --state related หรือ --match state --state relatedเป็น แพ็กเก็ต ที่เกี่ยวข้องกับ connection ที่สร้างไว้แล้ว แต่ไม่ใช่ส่วนหนึ่งส่วนใดของconnection นั้น เช่น FTP data แพ็กเก็ต (พอร์ต 20) ที่เกิดขึ้นจากการใช้คำสั่งใน FTP command (พอร์ต 21)

o INVALID

รูปแบบการใช้งาน: -m state --state invalid หรือ --match state --state invalidเป็น แพ็กเก็ต ที่ไม่เกี่ยวข้องกับส่วนอื่นเลย เช่น ไอซีเอ็มพี echo-reply ที่เกิดขึ้น โดยที่ไม่มีเครื่องได้ในระบบส่ง echo-request ออกไปเลย (กรณีเช่นนี้เกิดขึ้นได้เนื่องจากอาจจะโดนโจมตีแบบ Smurf attack)

การระบุทาร์เก็ต (target)

เมื่อมี แพ็กเก็ต ที่ match กับ กฎ แล้ว ต้องกำหนด target สำหรับ แพ็กเก็ต ไว้ด้วย โดยปกติจะใช้กัน 2target คือ DROP และ ACCEPT นอกจากนี้ยังมี target แบบอื่นได้คือ

• user-defined chain

เนื่องจาก iptables อนุญาตให้ผู้ใช้สามารถสร้าง chain ขึ้นมาได้ใหม่นอกเหนือจาก built-in chainทั้งสามตัว (INPUT, OUTPUT, FORWARD) ทั้งนี้จะต้องใช้ตัวอักษรตัวเล็กทั้งหมดสำหรับ chain ที่ผู้ใช้สร้างขึ้นเองเมื่อ แพ็กเก็ต match กับ กฎ ที่เป็น user-defined chain ตัว แพ็กเก็ต จะถูกนำไตรวจสอบใหม่โดย user-defined chain นั้นๆ และถ้าใน chain นั้นๆ ไม่มีการตัดสินใจใดๆ ตัว แพ็กเก็ต ก็สามารถย้อนกลับมายัง กฎ ถัดไปใน chain ที่เริ่มต้นเดินทางได้ (ศึกษารายละเอียดได้จากตัวอย่างด้านล่าง)

INPUT
Rule1: -p ICMP -j DROP
Rule2: -p TCP -j test
Rule3: -p UDP -j DROP

test
Rule1: -s 192.168.1.1
Rule2: -d 192.168.1.1

เช่น ถ้า ทีซีพี แพ็กเก็ต เดินทางจาก 192.168.1.1 ไปยัง 1.2.3.4 ดังนั้น แพ็กเก็ต จะเข้าสู่ INPUTchain และไม่ match กับ กฎ1 แต่ match กับ กฎ2 ซึ่งมี target เป็น test ดังนั้น แพ็กเก็ต จะเข้าสู่ test chain และ match กับ กฎ1 แต่เนื่องจาก กฎ1 ของ test ไม่ได้ระบุ target ดังนั้น แพ็กเก็ต จึงผ่านไปยัง กฎ2 ซึ่งไม่ match จากนั้น แพ็กเก็ต จึงจะเดินทางกลับไปยัง กฎ3 ของ INPUT chain อีกครั้ง ซึ่งก็ไม่ match เช่นกัน ในกรณีที่ผ่าน กฎ ทั้งหมดแล้วแต่ไม่ match หรือ match แต่ไม่มีtarget นั้น แพ็กเก็ต จะถูก DROP หรือ ACCEPT ก็ขึ้นอยู่กับ default policy ของ chain นั้นๆ ซึ่ง

สามารถตั้งค่าได้ง่ายๆ

เช่น # iptables -P INPUT DROP หรือ # iptables -P FORWARD ACCEPT

• new target

เป็น target ที่สร้างเพิ่มเติมขึ้นมาคือ

o LOG

เป็นโมดูลที่มีความสามารถในการเก็บข้อมูลลงล็อก (มี syslog facility เป็น kernel) สำหรับแพ็กเก็ต ที่ match กับ กฎ ที่ระบุ target เป็น LOG มีออปชันให้เลือกใช้งานดังนี้คือ

--log-level

เป็นการระบุ priority level ของ log ซึ่งกำหนดได้ตั้งแต่ debug, info ,notice,warning, crit, alert, emerg รายละเอียดเกี่ยวกับ syslog สามารถอ่านได้ที่

http://thaicert.nectec.or.th/paper/unix_linux/linux_syslog.php

--log-prefix

ตามด้วยชุดของตัวอักษรยาวไม่เกิน 29 ตัว ซึ่งชุดของตัวอักษรดังกล่าวจะปรากฏอยู่บนล็อกไฟล์

o REJECT

คล้ายกับ DROP เพียงแต่จะส่ง ไอซีเอ็มพี port unreachable กลับไปยังผู้ที่ส่ง แพ็กเก็ตมา (ข้อยกเว้นคือ ไอซีเอ็มพี error message ไม่ response กับ ไอซีเอ็มพี error message ด้วยกันเอง เพราะอาจจะทำให้เกิดลูปที่ไม่รู้จบ) ทั้งนี้สามารถใช้ร่วมกับ --reject-with ตามด้วย argument ที่ต้องการได้ รายละเอียดโปรดศึกษาจากคู่มือการใช้งาน iptables ที่มาพร้อมตัวโปรแกรม (#man iptables)

• special built-in target

o RETURN

กรณีที่ แพ็กเก็ต match กับ กฎ ที่มี target เป็น RETURN นั้นเสมือนกับเป็นคำสั่งให้ออกไปจาก chain ปัจจุบัน เช่น หาก match กับ กฎ ที่อยู่ใน built-in chain (INPUT,FORWARD, OUTPUT) แพ็กเก็ต ดังกล่าวจะถูกโยนไปยัง default policy ของ chain นั้นๆ และหาก แพ็กเก็ต match กับ กฎ ที่เป็น user-defined chain ตัว แพ็กเก็ต จะถูกโยนออกมา chain ก่อนหน้านั้น

o QUEUE

เป็น chain พิเศษ ใช้สำหรับส่งต่อ แพ็กเก็ต ไปยัง application ที่เขียนขึ้นมารองรับโดยเฉพาะ โดยจะต้องมี queue handler และ application เป็นส่วนประกอบที่จะทำงานร่วมกัน

การเดินทางของ แพ็กเก็ต ในระบบ

เมื่อ แพ็กเก็ต เข้ามาถึงไฟร์วอลล์ มันจะผ่านฮาร์ดแวร์เข้ามายัง Device ที่เหมาะสมในเคอร์เนล จากนั้นแพ็กเก็ต จะเดินทางไปเป็นทอดๆ ก่อนที่จะถูกส่งไปยังปลายทางที่แท้จริง เช่น แอพลิเคชันในเครื่องไฟร์วอลล์ หรือ forward ต่อไปยังเครื่องอื่น ซึ่งจะยกตัวอย่างเพื่อให้เห็นภาพอย่างชัดเจนดังนี้

Step	Table	Chain	Comment
1			ข้อมูลอยู่ในระหว่างการเดินทาง เช่น กำลังมาจากอินเทอร์เน็ต
2			ข้อมูลเข้ามายังเครื่องไฟร์วอลล์ผ่านทาง incoming interface (เช่น eth0)
3	mangle	PREROUTING	ใช้สำหรับการทำ mangling แพ็กเก็ต เท่านั้น เช่น เปลี่ยนค่า TOS ของแพ็กเก็ต ในกรณีปกติแล้วแทบไม่ได้ใช้งาน
4	nat	PREROUTING	chain นี้ใช้สำหรับทำ Destination Network Address Translation ไม่ควรสร้าง กฎ เพื่อกรอง แพ็กเก็ต ที่ chain นี้ เพราะอาจจะมีบาง แพ็กเก็ต ที่ไม่เข้าสู่ chain นี้ (มีแค่แพ็กเก็ต แรกเท่านั้นที่ผ่านเข้าสู่ chain ส่วน แพ็กเก็ตถัดไปใน connection เดียวกันนั้น จะถูกกระทำเหมือนกับที่ แพ็กเก็ตแรกได้รับ)
5			เข้าสู่ Routing decision เพื่อตัดสินใจว่า แพ็กเก็ต จะถูกส่งไปที่ใด
6	Filter	FORWARD	เนื่องจากในตัวอย่างนี้ แพ็กเก็ต จะถูกส่งไปยังเครื่องอื่นในเครือข่ายดังนั้น แพ็กเก็ต จึงต้องเข้า FORWARD chain ของ filter table ซึ่งสามารถเขียน กฎ สำหรับควบคุมการผ่านเข้าออกของ แพ็กเก็ต สำหรับforwarded แพ็กเก็ตได้ที่นี่

Step	Table	Chain	Comment
7	Nat	POSTROUTING	และก่อนที่ แพ็กเก็ต จะออกไปจากเครื่องไฟร์วอลล์ โดยส่วนใหญ่(ไม่ใช่ทั้งหมด) จะผ่าน chain นี้ ซึ่งใช้ทำ Source Network Address Translation ไม่ควรสร้าง กฎ เพื่อกรอง แพ็กเก็ต ที่ chain นี้ เพราะอาจจะมีบาง แพ็กเก็ต ที่ไม่เข้าสู่ chain นี้ (มีแค่ แพ็กเก็ต แรกเท่านั้นที่ผ่านเข้าสู่ chain ส่วนแพ็กเก็ต ถัดไปใน connection เดียวกันนั้น จะถูกกระทำเหมือนกับที่แพ็กเก็ต แรกได้รับ)
8			แพ็กเก็ต ออกไปทาง outgoing interface (เช่น eth1)
9			แพ็กเก็ต เดินทางไปสู่เป้าหมาย (เช่น ผ่านทาง LAN)

ตารางที่ 2-2 การเดินทางของฟอร์วาร์ดแพ็กเก็ต (Forwarded แพ็กเก็ต)

อย่างที่เห็นจากตาราง forwarded แพ็กเก็ตคือ แพ็กเก็ตที่มีปลายทางที่ไม่ใช่เครื่อง ไฟร์วอลล์ อาจจะเป็นแพ็กเก็ตที่มาจากอินเทอร์เน็ต หรืออาจจะเป็นแพ็กเก็ตที่มาจากเครื่องลูกในเครือข่ายที่ต้องการส่งออกไปอินเทอร์เน็ต ซึ่งไม่ว่าจะส่งในทิศทางใด แพ็กเก็ตก็จะต้องผ่าน chain ในลักษณะด้านบนนี้เสมอ

Step	Table	Chain	Comment
1			ข้อมูลอยู่ในระหว่างการเดินทาง เช่น กำลังมาจากอินเทอร์เน็ต
2			ข้อมูลเข้ามายังเครื่องไฟร์วอลล์ผ่านทาง incoming interface (เช่น eth0)
3	mangle	PREROUTING	ใช้สำหรับการทำ mangling แพ็กเก็ต เท่านั้น เช่น เปลี่ยนค่า TOS ของแพ็กเก็ต ซึ่งในกรณีปกติแล้วแทบไม่ได้ใช้งาน
4	nat	PREROUTING	chain นี้ใช้สำหรับทำ Destination Network Address Translation ไม่ควรสร้าง กฎ เพื่อกรอง แพ็กเก็ต ที่ chain นี้ เพราะอาจจะมีบาง แพ็กเก็ต ที่ไม่เข้าสู่ chain นี้ (มีแค่ แพ็กเก็ต แรกเท่านั้นที่ผ่านเข้าสู่ chain ส่วน แพ็กเก็ตถัดไปใน connection เดียวกันนั้น จะถูกกระทำเหมือนกับที่ แพ็กเก็ต แรกได้รับ

Step	Table	Chain	Comment
5			เข้าสู่ Routing decision เพื่อตัดสินใจว่า แพ็กเก็ต จะถูกส่งไปที่ใด
6	filter	INPUT	ทุก แพ็กเก็ต ที่มีเป้าหมายเป็นเครื่องไฟร์วอลล์จะต้องเข้าสู่ chain นี้เสมอไม่ว่าจะมาจาก interface ใดก็ตาม
7			Local process/application (เช่น server/client program)

ตารางที่ 2-3 การเดินทางของแพ็กเก็ตที่โฮสต์ปลายทาง (Destination localhost)

Step	Table	Chain	Comment
1			Local process/application (เช่น server/client program)
2	mangle	OUTPUT	ใช้สำหรับการทำ mangling แพ็กเก็ต เท่านั้น การกรอง แพ็กเก็ต ที่ chain นี้จะไม่มีผลใดๆ ต่อ แพ็กเก็ต
3	nat	OUTPUT	ไม่ได้ใช้งาน
4	filter	OUTPUT	ใช้สำหรับกรอง แพ็กเก็ต ที่ออกมาจาก localhost หรือเครื่องไฟร์วอลล์เอง
5			เข้าสู่ Routing decision เพื่อตัดสินใจว่า แพ็กเก็ต จะถูกส่งไปที่ใด
6	nat	POSTROUTING	และก่อนที่ แพ็กเก็ต จะออกไปจากเครื่องไฟร์วอลล์ โดยส่วนใหญ่ (ไม่ใช่ทั้งหมด) จะผ่าน chain นี้ ซึ่งใช้ทำ Source Network Address Translation ไม่ควรสร้าง กฎ เพื่อกรอง แพ็กเก็ต ที่ chain นี้ เพราะอาจจะมีบาง แพ็กเก็ต ที่ไม่เข้าสู่ chain นี้ (มีแต่ แพ็กเก็ต แรกเท่านั้นที่ผ่านเข้าสู่ chain ส่วนแพ็กเก็ตถัดไปใน connection เดียวกันนั้น จะถูกกระทำเหมือนกับที่แพ็กเก็ต แรกได้รับ)
7			แพ็กเก็ต ออกไปทาง outgoing interface (เช่น eth1)
8			Local process/application (เช่น server/client program)

ตารางที่ 2-4 การเดินทางของแพ็กเก็ตที่โฮสต์ต้นทาง (Source localhost)

จากสามตัวอย่างด้านบน คงพอจะมองเห็นภาพออกแล้วว่า แพ็กเก็ต เดินทางเข้าออกในระบบอย่างไร อย่างไรก็ตาม จะสรุปความสำคัญของแต่ละตาราง (table) อีกครั้ง ดังนี้

ฟิลเตอร์เทเบิล (Filter Table)

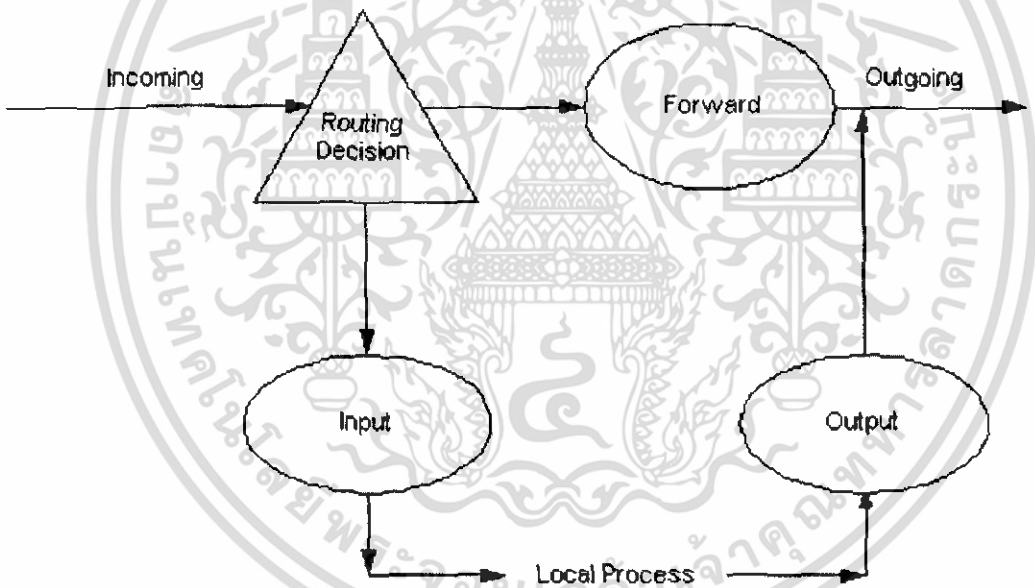
เป็นตารางที่ใช้งานมากที่สุด เป็นจุดที่ใช้ในการตรวจสอบและควบคุมการผ่านเข้าออกของแพ็กเก็ต ถ้าหากจะพิจารณาการไหลเวียนของ แพ็กเก็ต เฉพาะในส่วนของ filter table โดยไม่สนใจ table อื่นๆ นั้น ก็พอจะแสดงให้เห็นได้ดังภาพที่ 1 โดยเมื่อ แพ็กเก็ต เข้ามาในระบบ มันจะเข้าไปยัง routing decision เพื่อ

ตัดสินใจว่า แพ็กเก็ต จะถูกส่งไปที่ใด

• ในกรณีที่ แพ็กเก็ต ถูกส่งผ่านไปยังเครื่องอื่น แพ็กเก็ต นั้นจะต้องถูกตรวจสอบโดย กฎ ใน FORWARD chain

• ถ้า แพ็กเก็ต นั้น มีเป้าหมายเป็นเครื่องปัจจุบัน (เครื่องที่รัน iptables อยู่นี้ เรียกอีกอย่างว่า ลีนุกซ์ box) ตัว แพ็กเก็ต จะถูกตรวจสอบโดย กฎ ใน INPUT chain

• และในกรณีที่ แพ็กเก็ต ถูกสร้างจากเครื่องปัจจุบัน (ลีนุกซ์ box) ตัว แพ็กเก็ต จะถูกตรวจสอบจากกฎ ใน OUTPUT chain ก่อนที่จะถูกส่งออกไป



รูปที่ 2-9 แสดงให้เห็นว่า แพ็กเก็ต มีเส้นทางการเดินทางอย่างไรเมื่อเข้ามาในระบบ (filter table)

ดังภาพ iptables ประกอบไปด้วย built-in chain จำนวน 3 chain ซึ่งไม่สามารถลบได้คือ INPUT,OUTPUT, FORWARD เมื่อเครื่องคอมพิวเตอร์เริ่มทำงานในครั้งแรก ทั้งสาม chain จะมี default policy เป็น ACCEPT ซึ่งหมายความว่าอนุญาตให้ทุกอย่างผ่านเข้าออกได้หมด และสำหรับ FORWARD chain นั้น ถึงแม้จะได้กำหนดให้ policy เป็น ACCEPT แล้ว แพ็กเก็ต ก็จะยังไม่สามารถถูก

forward ไปยังจุดหมายที่ต้องการได้ ตราบใดที่ยังไม่ได้เซ็ตให้ enable IP forwarding ทั้งนี้โดย default แล้ว forward=0 สามารถกำหนดให้ enable IP forwarding (forward=1) ได้โดยใช้คำสั่ง echo "1" > /proc/sys/net/ip_forward เพื่อกำหนดให้ IP forwarding เป็น enable เพื่อให้ ลีนุกซ์ box สามารถforward ip แพ็กเก็ต ได้ ในบางครั้งนั้นการใช้คำสั่งดังกล่าวทุกครั้งอาจจะไม่สะดวก สามารถแก้ไขไฟล์ configuration ที่ /etc/sysctl.conf แล้ว set ให้ net.ipv4.ip_forward=1 เพื่อเป็นการแก้ไขแบบถาวร ในกรณีที่ต้องการให้สนับสนุนการทำงานกับ dynamic IP ด้วย เช่น PPP, SLIP, DHCP ก็สามารถทำได้โดยใช้คำสั่ง echo "1" > /proc/sys/net/ipv4/ip_dynaddr ได้เช่นเดียวกัน

2.4 กฎและความสัมพันธ์ในไฟร์วอลล์

ความสัมพันธ์ของกฎในไฟร์วอลล์

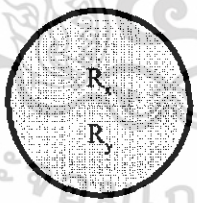
Exactly Matching ($\Re_{EM}$)

กฎ R_x จะมีความสัมพันธ์กับกฎ R_y แบบ *Exactly Matching* ($R_x \Re_{EM} R_y$) ก็ต่อเมื่อทุกๆ ฟิลด์ใน R_x มีค่าเท่ากับทุกๆ ฟิลด์ที่สัมพันธ์กันใน R_y

$\forall i: R_x[i] = R_y[i]$

โดยที่ $i \in$ {protocol, src_ip, src_port, dst_ip, dst_port}

1	TCP	140.192.37.10	any	163.122.51.*	21	accept
2	TCP	140.192.37.10	any	163.122.51.*	21	deny



รูปที่ 2-10 แสดงความสัมพันธ์ระหว่างกฎแบบ Exactly Matching

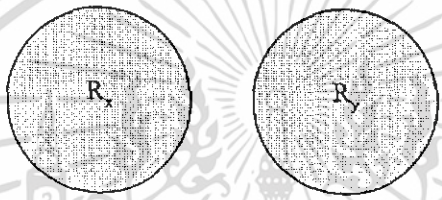
Completely Disjoint ($\Re_{CD}$)

กฎ R_x จะมีความสัมพันธ์กับกฎ R_y แบบ *Completely Disjoint* ($R_x \Re_{CD} R_y$) ก็ต่อเมื่อทุกๆ ฟิลด์ใน R_x มีค่าไม่เท่ากับ ไม่เป็นซับเซต หรือซุปเปอร์เซต กับฟิลด์ที่สัมพันธ์กันใน R_y

$\forall i: R_x[i] \not>< R_y[i]$

โดยที่ $>< \in \{\subset, \supset, =\}$ และ $i \in$ {protocol, src_ip, src_port, dst_ip, dst_port}

1	TCP	140.192.37.10	2000	163.122.51.50	80	accept
2	UDP	140.192.37.20	3000	163.122.51.60	21	accept



รูปที่ 2-11 แสดงความสัมพันธ์ระหว่างกฎแบบ Completely Disjoint

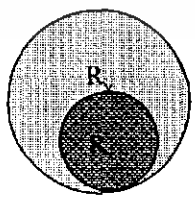
Inclusively Matching ($\Re_{IM}$)

กฎ R_x จะมีความสัมพันธ์กับกฎ R_y แบบ *Inclusively Matching* ($R_x \Re_{IM} R_y$) ก็ต่อเมื่อ ทุกๆ ฟิลด์ใน R_x มีค่าเป็นซับเซต หรือเท่ากับฟิลด์ที่สัมพันธ์กันใน R_y โดยจะเรียก R_x เป็นซับเซตแมทช์ (subset match) และ R_y เป็นซุปเปอร์เซตแมทช์ (superset match)

$\forall i: R_x[i] \subseteq R_y[i]$ และ $\exists j: R_x[j] \neq R_y[j]$

โดยที่ i, j $\in$ {protocol, src_ip, src_port, dst_ip, dst_port}

1	TCP	140.192.37.10	any	*.*.*.*	80	deny
2	TCP	140.192.37.*	any	*.*.*.*	any	accept



รูปที่ 2-12 แสดงความสัมพันธ์ระหว่างกฎแบบ Inclusively Matching

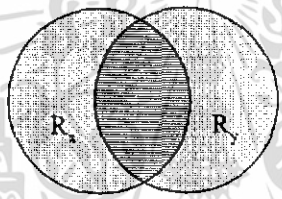
Partially Disjoint ($\Re_{PD}$)

กฎ R_x จะมีความสัมพันธ์กับกฎ R_y แบบ *Partially Disjoint* ($R_x \Re_{PD} R_y$) ก็ต่อเมื่อ ฟิลด์ใน R_x อย่างน้อยหนึ่งฟิลด์มีค่าเท่ากับ เป็นซับเซต หรือเป็นซุปเปอร์เซต กับค่าของฟิลด์ที่สัมพันธ์กันใน R_y และมีอย่างน้อยหนึ่งฟิลด์ใน R_x ที่มีค่าไม่เท่ากับ ไม่เป็นซับเซต หรือไม่เป็นซุปเปอร์เซตกับค่าของฟิลด์ที่สัมพันธ์กันใน R_y

$\exists$ i, j : $R_x[i] >< R_y[i]$ และ $R_x[i] \not>< R_y[i]$

โดยที่ $>< \in \{\subset, \supset, =\}$ และ i, j $\in$ {protocol, src_ip, src_port, dst_ip, dst_port}

1	TCP	140.192.37.10	any	*.*.*.*	80	accept
2	TCP	140.192.37.*	any	*.*.*.*	21	deny



รูปที่ 2-13 แสดงความสัมพันธ์ระหว่างกฎแบบ Partially Disjoint

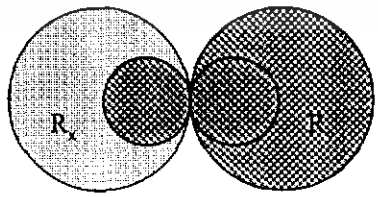
Correlation ($\Re_C$)

กฎ R_x จะมีความสัมพันธ์กับกฎ R_y แบบ *Correlation* ($R_x \Re_C R_y$) ก็ต่อเมื่อบางฟิลด์ใน R_x มีค่าเป็นซับเซต หรือเท่ากับฟิลด์ที่สัมพันธ์กันใน R_y และส่วนอื่นๆใน R_x มีค่าเป็นซุปเปอร์เซตของฟิลด์ที่สัมพันธ์กันใน R_y

$\forall$ i: $R_x[i] >< R_y[i]$ และ $\exists$ i, j: $R_x[i] \subset R_y[i]$ และ $R_x[j] \supset R_y[j]$

โดยที่ $>< \in \{\subset, \supset, =\}$ และ i, j $\in$ {protocol, src_ip, src_port, dst_ip, dst_port}

1	TCP	140.192.37.10	any	*.*.*.*	80	accept
2	TCP	*.*.*.*	any	140.192.37.*	80	deny



รูปที่ 2-14 แสดงความสัมพันธ์ระหว่างกฎแบบ Correlation

ความขัดแย้งของกฎในไฟร์วอลล์

Shadowing anomaly

กฎสองกฎจะมีความขัดแย้งแบบ Shadowing ก็ต่อเมื่อกฎก่อนหน้าที่ผ่านมามีความสัมพันธ์กับแพ็กเก็ต ซึ่งมีความสัมพันธ์กับกฎที่อยู่ถัดไปเช่นกัน

ปัญหาที่เกิดขึ้น: กฎที่อยู่ถัดไปซึ่งมีความสัมพันธ์กับกฎที่อยู่ก่อนหน้าแบบ Shadowing จะไม่ถูกใช้งาน เนื่องจากแพ็กเก็ตจะแมทช์กับกฎที่มาก่อนเสมอ

R_x[order] < R_y[order],	$R_x \Re_{EM} R_y$,	R_x[action] ≠ R_y[action]
R_x[order] < R_y[order],	$R_y \Re_{IM} R_x$,	R_x[action] ≠ R_y[action]

11	UDP	140.192.38.*	any	161.120.35.*	any	accept
12	UDP	140.192.38.*	any	161.120.35.*	any	deny

รูปที่ 2-15 แสดงตัวอย่างความขัดแย้งของกฎแบบ Shadowing

3	TCP	*.*.*.*	any	161.120.33.40	80	accept
4	TCP	140.192.37.*	any	161.120.33.40	80	deny

รูปที่ 2-16 แสดงตัวอย่างความขัดแย้งของกฎแบบ Shadowing (ต่อ)

Correlation anomaly

กฎสองกฎจะมีความขัดแย้งแบบ Correlation ก็ต่อเมื่อมีแพ็กเก็ตซึ่งแมทช์กับบางส่วนของกฎอันแรกกับกฎถัดไป และมีแพ็กเก็ตซึ่งแมทช์กับบางส่วนของกฎถัดไปกับกฎอันแรก

ปัญหาที่เกิดขึ้น: โดยส่วนมากในกรณีที่เกิดความขัดแย้งแบบ Correlation ถ้ากฎที่มีมาก่อนมีการกระทำ (action) เป็น deny จะไม่มีผลกระทบใดๆ แต่ถ้ามีการสลับที่กฎที่มีความขัดแย้งแบบ Correlation ซึ่งมีการกระทำเป็น accept ขึ้นก่อน จะทำให้นโยบายทางด้านความปลอดภัยไม่ถูกต้อง

$R_x[order] < R_y[order]$,	$R_x \Re_C R_y$,	$R_x[action] \neq R_y[action]$

1	TCP	140.192.37.20	any	*.*.*.*	80	deny
3	TCP	*.*.*.*	any	161.120.33.40	80	accept

รูปที่ 2-17 แสดงตัวอย่างความขัดแย้งของกฎแบบ Correlation

Generalization anomaly

กฎสองกฎจะมีความขัดแย้งแบบ Generalization ก็ต่อเมื่อมีแพ็กเก็ตใดๆ สามารถแมทช์กับกฎอันแรกและอันที่สองได้ทั้งหมด โดยที่การกระทำ (action) ของกฎทั้งสองแตกต่างกัน

ปัญหาที่เกิดขึ้น: ถ้ามีการสลับที่ของกฎจะทำให้นโยบายทางด้านความปลอดภัยไม่ถูกต้อง

$R_x[order] < R_y[order]$,	$R_x \Re_{IM} R_y$,	$R_x[action] \neq R_y[action]$

1	TCP	140.192.37.20	any	*.*.*.*	80	deny
2	TCP	140.192.37.*	any	*.*.*.*	80	accept

รูปที่ 2-18 แสดงตัวอย่างความขัดแย้งของกฎแบบ Generalization

Redundancy anomaly

กฎสองกฎมีความขัดแย้งแบบ Redundancy ก็ต่อเมื่อมีแพ็กเก็ตใดๆ สามารถแมทช์กับกฎที่หนึ่งและสองได้ โดยมีการกระทำ (action) ที่เหมือนกัน

ปัญหาที่เกิดขึ้น: จะทำให้กฎที่อยู่ถัดไปไม่ถูกใช้งาน และจะทำให้ไฟร์วอลล์ทำงานช้าลง

$R_x[order] < R_y[order]$,	$R_x \Re_{EM} R_y$,	$R_x[action] = R_y[action]$
$R_x[order] < R_y[order]$,	$R_x \Re_{IM} R_y$,	$R_x[action] = R_y[action]$

ในกรณีที่ $R_x[order] < R_y[order]$, $R_x \Re_{EM} R_y$, $R_x[action] = R_y[action$

x	TCP	140.192.37.20	any	*.*.*.*	80	accept
M	M	M	M	M	M	M
y	TCP	140.192.37.20	any	*.*.*.*	80	accept

ในกรณีที่ $R_x[order] < R_y[order]$, $R_x \Re_{IM} R_y$, $R_x[action] = R_y[action]$

6	TCP	140.192.37.*	any	*.*.*.*	21	accept
7	TCP	140.192.37.*	any	161.120.33.40	21	accept

9	UDP	140.192.37.*	any	161.120.33.40	53	accept
10	UDP	*.*.*.*	any	161.120.33.40	53	accept

รูปที่ 2-19 แสดงตัวอย่างความขัดแย้งของกฎแบบ Redundancy

บทที่ 3

การออกแบบและพัฒนา

การสั่งให้โปรแกรมทำงานมีการเรียกใช้ดังนี้

>./optimize <Options = {remove, log, reorder}> <Chain = {INPUT, OUTPUT, FORWARD}

ซึ่งในการสั่งให้โปรแกรมทำงานนั้น ต้องมีการกำหนดการอนุญาตการใช้งานไฟล์ด้วยคำสั่ง chmod ให้มีการยินยอมให้มีการเรียกใช้งานไฟล์

การทำงานของตัวโปรแกรมแบ่งการทำงานออกเป็น 4 ส่วน

3.1 ส่วนลบกฎที่ซ้ำซ้อน

เนื่องจากกฎที่ซ้ำซ้อนกันจะทำให้กฎที่เหมือนกันแต่อยู่ลำดับต่ำกว่าจะไม่ถูกใช้งานเลยเพราะกฎที่ถูกใช้งานจะเป็นกฎบนทุกครั้งก็จะหยุดตรวจสอบ

ขั้นตอนการทำงานของโปรแกรม

1.1 ทำการเก็บกฎทั้งหมดไว้เป็นไฟล์ชื่อ temp ที่โปรแกรมจะใช้ในการอ่านกฎ

1.2 ทำการล้างกฎของไฟร์วอลล์ในเชนนั้นๆ ทั้งหมด

1.3 ทำการอ่านรายละเอียดของกฎจากไฟล์ในข้อ 1.1

1.4 จาก 1.3 นำกฎต่อไปมาเริ่มต่อกฎเดิมในไฟล์วอลล์

1.5 เก็บกฎใหม่นี้ลงไฟล์ temp1

1.6 อ่านกฎจากไฟล์ข้อ 1.1 เรียงลำดับจากกฎด้านล่างสุดขึ้นไปยังกฎบนสุด

1.7 นำกฎที่อ่านไปเปรียบเทียบกับกฎในไฟล์ข้อ 1.5 เพื่อหาความผิดปกติของกฎว่าซ้ำซ้อนหรือไม่ แล้วปฏิบัติตามกฎข้อบังคับในการลบกฎของไฟร์วอลล์

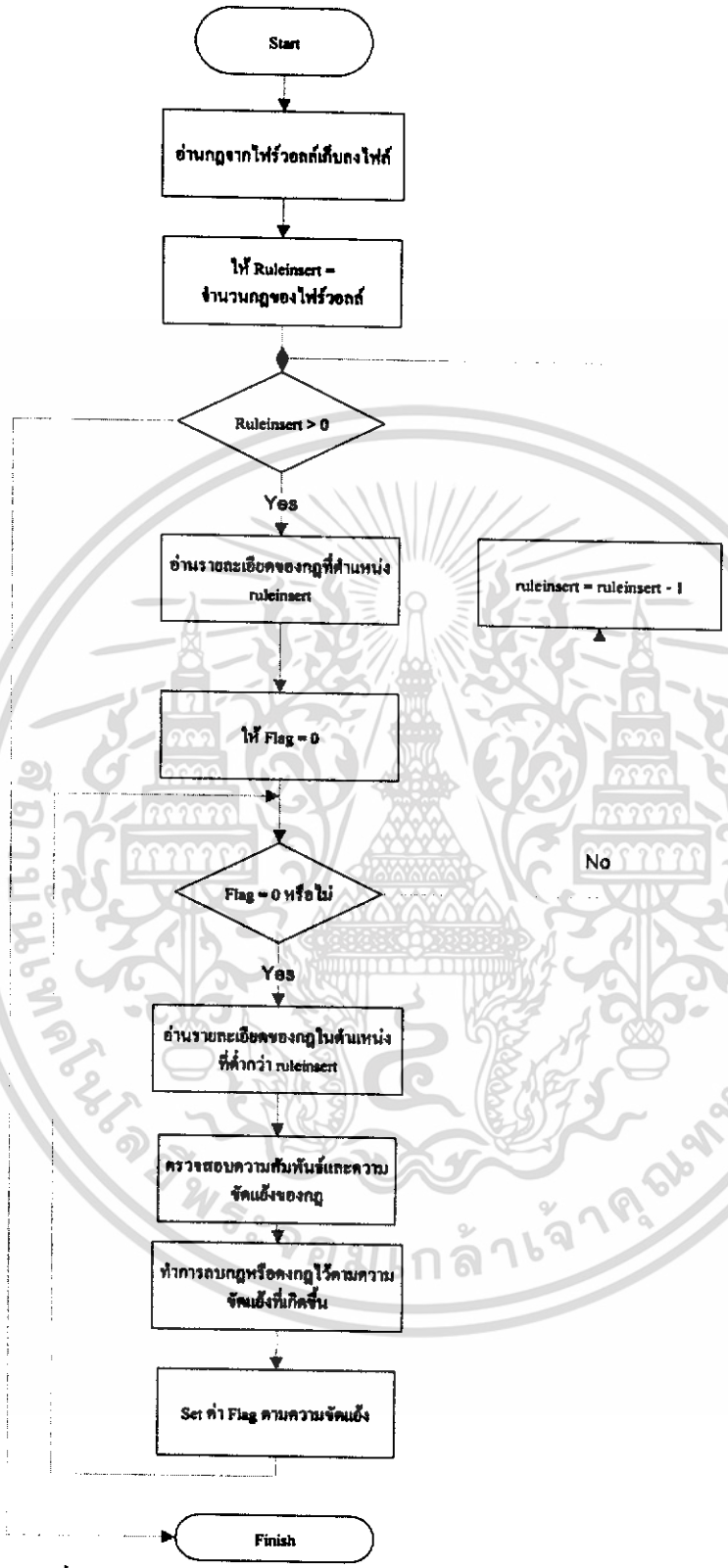
1.8 ถ้ากฎที่อ่านมานั้นไม่มีการซ้ำซ้อน จะไม่ถูกลบ ซึ่งโปรแกรมจะทำการใส่กฎลงไปแล้วทำซ้ำข้อ 1.5 มาเรื่อยๆ จนกระทั่งครบทุกกฎ

1.9 เซฟกฎของไฟร์วอลล์โดยใช้คำสั่ง iptables-save

1.10 กฎที่ถูกลบออกไปจะมีการเก็บรายงานเอาไว้ในไฟล์ removereport ซึ่งอยู่ในแฟ้มเอกสารเดียวกันกับตัวโปรแกรม

สามารถสั่งให้โปรแกรมทำงานในส่วนนี้โดยใช้คำสั่ง

>./optimize remove <chain = {INPUT, OUTPUT, FORWARD}>



รูปที่ 3-1 Flow Chart แสดงการทำงานของส่วนลบกฎที่ซ้ำซ้อน

การตรวจสอบความขัดแย้งของกฎ

ให้กฎ Rx มาก่อนกฎ Ry

จากรูปที่ 3-1 ส่วนที่ทำการตรวจสอบความขัดแย้งของกฎซึ่งจะทำการตรวจสอบความขัดแย้งระหว่าง 2 กฎ โดยสามารถแบ่งขั้นตอนการทำงานออกเป็น 3 ส่วนคือ

1. ส่วนที่ทำการเปรียบเทียบความสัมพันธ์ของแต่ละฟิลล์ในกฎของไฟล์วอลล์ ซึ่งการเปรียบเทียบฟิลล์ต่างๆในไฟร์วอลล์จะมีหลักในการเปรียบเทียบดังนี้

- ฟิลล์โพรโตคอล

ถ้าฟิลล์โพรโตคอลของ Rx มีค่าเท่ากับ ฟิลล์โพรโตคอลของ Ry ความสัมพันธ์ของฟิลล์จะมีค่าเป็น exact

ถ้าฟิลล์โพรโตคอลของ Rx มีค่าไม่เท่ากับ ฟิลล์โพรโตคอลของ Ry และฟิลล์โพรโตคอลของ Rx มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น subset

ถ้าฟิลล์โพรโตคอลของ Rx มีค่าไม่เท่ากับ ฟิลล์โพรโตคอลของ Ry และฟิลล์โพรโตคอลของ Ry มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น superset

ถ้าฟิลล์โพรโตคอลของ Rx มีค่าไม่เท่ากับ ฟิลล์โพรโตคอลของ Ry และฟิลล์โพรโตคอลของทั้ง Rx และ Ry ไม่มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น disjoint

- ฟิลล์พอร์ตต้นทางและพอร์ตปลายทาง

ถ้าฟิลล์พอร์ตต้นทางของ Rx มีค่าเท่ากับ ฟิลล์พอร์ตต้นทางของ Ry ความสัมพันธ์ของฟิลล์จะมีค่าเป็น exact

ถ้าฟิลล์พอร์ตต้นทางของ Rx มีค่าไม่เท่ากับ ฟิลล์พอร์ตต้นทางของ Ry และฟิลล์พอร์ตต้นทางของ Rx มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น subset

ถ้าฟิลล์พอร์ตต้นทางของ Rx มีค่าไม่เท่ากับ ฟิลล์พอร์ตต้นทางของ Ry และฟิลล์พอร์ตต้นทางของ Ry มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น superset

ถ้าฟิลล์พอร์ตต้นทางและพอร์ตปลายทางของ Rx มีค่าไม่เท่ากับ ฟิลล์พอร์ตต้นทางของ Ry และฟิลล์พอร์ตต้นทางของทั้ง Rx และ Ry ไม่มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น disjoint

- ฟิลด์พอร์ตปลายทาง

ถ้าฟิลล์พอร์ตปลายทางของ Rx มีค่าเท่ากับ ฟิลล์พอร์ตปลายทางของ Ry ความสัมพันธ์ของฟิลล์จะมีค่าเป็น exact

ถ้าฟิลล์พอร์ตปลายทางของ Rx มีค่าไม่เท่ากับ ฟิลล์พอร์ตปลายทางของ Ry และฟิลล์พอร์ตปลายทางของ Rx มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น subset

ถ้าฟิลล์พอร์ตปลายทางของ Rx มีค่าไม่เท่ากับ ฟิลล์พอร์ตปลายทางของ Ry และฟิลล์พอร์ตปลายทางของ Ry มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น superset

ถ้าฟิลล์พอร์ตปลายทางของ Rx มีค่าไม่เท่ากับ ฟิลล์พอร์ตปลายทางของ Ry และฟิลล์พอร์ตปลายทางของทั้ง Rx และ Ry ไม่มีค่าเป็น all ความสัมพันธ์ของฟิลล์จะมีค่าเป็น disjoint

- ฟิลล์ไอพีแอดเดรสต้นทาง

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก เท่ากับไอพีแอดเดรสต้นทาง Ry และ ค่าเน็ทมาส์กเท่ากับ 0 ความสัมพันธ์ของฟิลด์จะมีค่าเป็น exact

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก เท่ากับไอพีแอดเดรสต้นทาง Ry และ ค่าเน็ทมาส์กไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสต้นทางของ Rx กับไอพีแอดเดรสต้นทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาร์กถ้าเหมือนกันทุกบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น exact

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก เท่ากับไอพีแอดเดรสต้นทาง Ry และ ค่าเน็ทมาส์กไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสต้นทางของ Rx กับไอพีแอดเดรสต้นทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาร์กถ้ามีความแตกต่างกันระหว่างบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น disjoint

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก น้อยกว่าไอพีแอดเดรสต้นทาง Ry และ ค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Rx เท่ากับ 0 ความสัมพันธ์ของฟิลด์จะมีค่าเป็น subset

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก น้อยกว่าไอพีแอดเดรสต้นทาง Ry และค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Rx ไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสต้นทางของ Rx กับไอพีแอดเดรสต้นทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Rx ถ้าเหมือนกันทุกบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น subset

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก น้อยกว่าไอพีแอดเดรสต้นทาง Ry และค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Rx ไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสต้นทางของ Rx กับไอพีแอดเดรสต้นทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Rx ถ้ามีความแตกต่างกันระหว่างบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น disjoint

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก มากกว่าไอพีแอดเดรสต้นทาง Ry และ ค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Ry เท่ากับ 0 ความสัมพันธ์ของฟิลด์จะมีค่าเป็น superset

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก มากกว่าไอพีแอดเดรสต้นทาง Ry และค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Ry ไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสต้นทางของ Rx กับไอพีแอดเดรสต้นทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Ry ถ้าเหมือนกันทุกบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น superset

ถ้าไอพีแอดเดรสต้นทางของ Rx มีค่าเน็ทมาส์ก มากกว่าไอพีแอดเดรสต้นทาง Ry และค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Ry ไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสต้นทางของ Rx กับไอพีแอดเดรสต้นทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาส์กของไอพีแอดเดรสต้นทางของ Ry ถ้ามีความแตกต่างกันระหว่างบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น disjoint

- ฟิลด์ไอพีแอดเดรสปลายทาง

ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก เท่ากับไอพีแอดเดรสปลายทาง Ry และ ค่าเน็ทมาส์กเท่ากับ 0 ความสัมพันธ์ของฟิลด์จะมีค่าเป็น exact

ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก เท่ากับไอพีแอดเดรสปลายทาง Ry และ ค่าเน็ทมาส์กไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสปลายทางของ Rx กับไอพีแอดเดรสปลายทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาร์กถ้าเหมือนกันทุกบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น exact

ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก เท่ากับไอพีแอดเดรสปลายทาง Ry และ ค่าเน็ทมาส์กไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสปลายทาง

ของ Rx กับไอพีแอดเดรสปลายทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาร์กถ้ามีความแตกต่างกันระหว่างบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น disjoint

ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก น้อยกว่าไอพีแอดเดรสปลายทาง Ry และ ค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Rx เท่ากับ 0 ความสัมพันธ์ของฟิลด์จะมีค่าเป็น subset

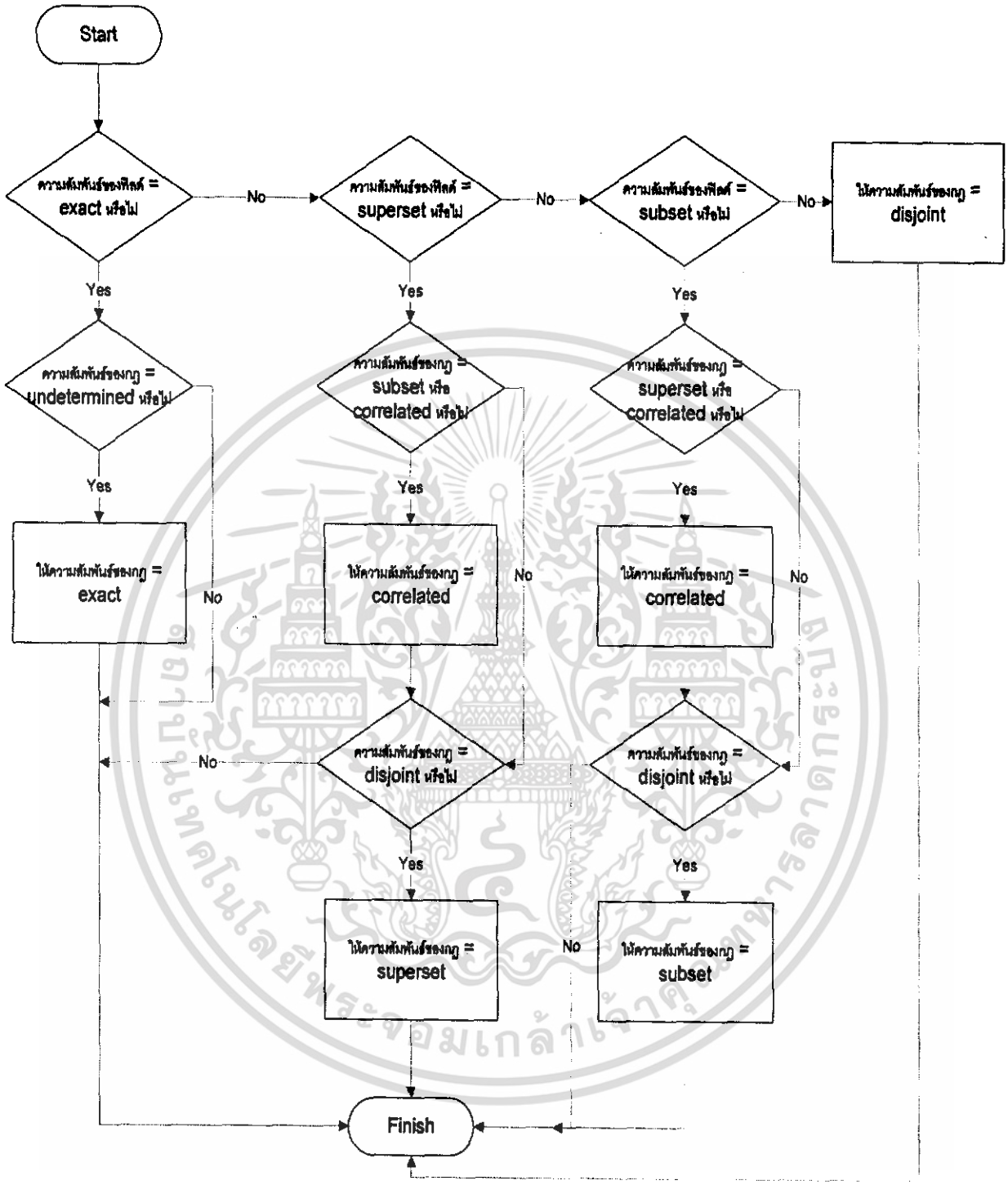
ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก น้อยกว่าไอพีแอดเดรสปลายทาง Ry และค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Rx ไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสปลายทางของ Rx กับไอพีแอดเดรสปลายทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Rx ถ้าเหมือนกันทุกบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น subset

ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก น้อยกว่าไอพีแอดเดรสปลายทาง Ry และค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Rx ไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสปลายทางของ Rx กับไอพีแอดเดรสปลายทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Rx ถ้ามีความแตกต่างกันระหว่างบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น disjoint

ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก มากกว่าไอพีแอดเดรสปลายทาง Ry และ ค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Ry เท่ากับ 0 ความสัมพันธ์ของฟิลด์จะมีค่าเป็น superset

ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก มากกว่าไอพีแอดเดรสปลายทาง Ry และค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Ry ไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสปลายทางของ Rx กับไอพีแอดเดรสปลายทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Ry ถ้าเหมือนกันทุกบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น superset

ถ้าไอพีแอดเดรสปลายทางของ Rx มีค่าเน็ทมาส์ก มากกว่าไอพีแอดเดรสปลายทาง Ry และค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Ry ไม่เท่ากับ 0 และเปรียบเทียบไอพีแอดเดรสปลายทางของ Rx กับไอพีแอดเดรสปลายทางของ Ry ทีละบิท จำนวนบิทที่เทียบเท่ากับค่าเน็ทมาส์กของไอพีแอดเดรสปลายทางของ Ry ถ้ามีความแตกต่างกันระหว่างบิท ความสัมพันธ์ของฟิลด์จะมีค่าเป็น disjoint



รูป 3-2 แสดงการขั้นตอนการหาความสัมพันธ์ระหว่างกฎ 2 กฎ

2. ส่วนที่ทำหาความสัมพันธ์ระหว่างกฎ 2 กฎของไฟล์วอลล์ ซึ่งการเปรียบเทียบกฎซึ่งจะทำการเปรียบเทียบทีละฟิลด์โดยมีขั้นตอนการเปรียบเทียบความสัมพันธ์ของกฎดังนี้

ถ้าความสัมพันธ์ของฟิลด์เป็น exact และความสัมพันธ์ของกฎเป็น undetermined จะให้ความสัมพันธ์ของกฎเป็น exact

ถ้าความสัมพันธ์ของฟิลด์เป็น exact แต่ความสัมพันธ์ของกฎเป็นค่าอื่น จะคงความสัมพันธ์ของกฎค่าเดิมไว้

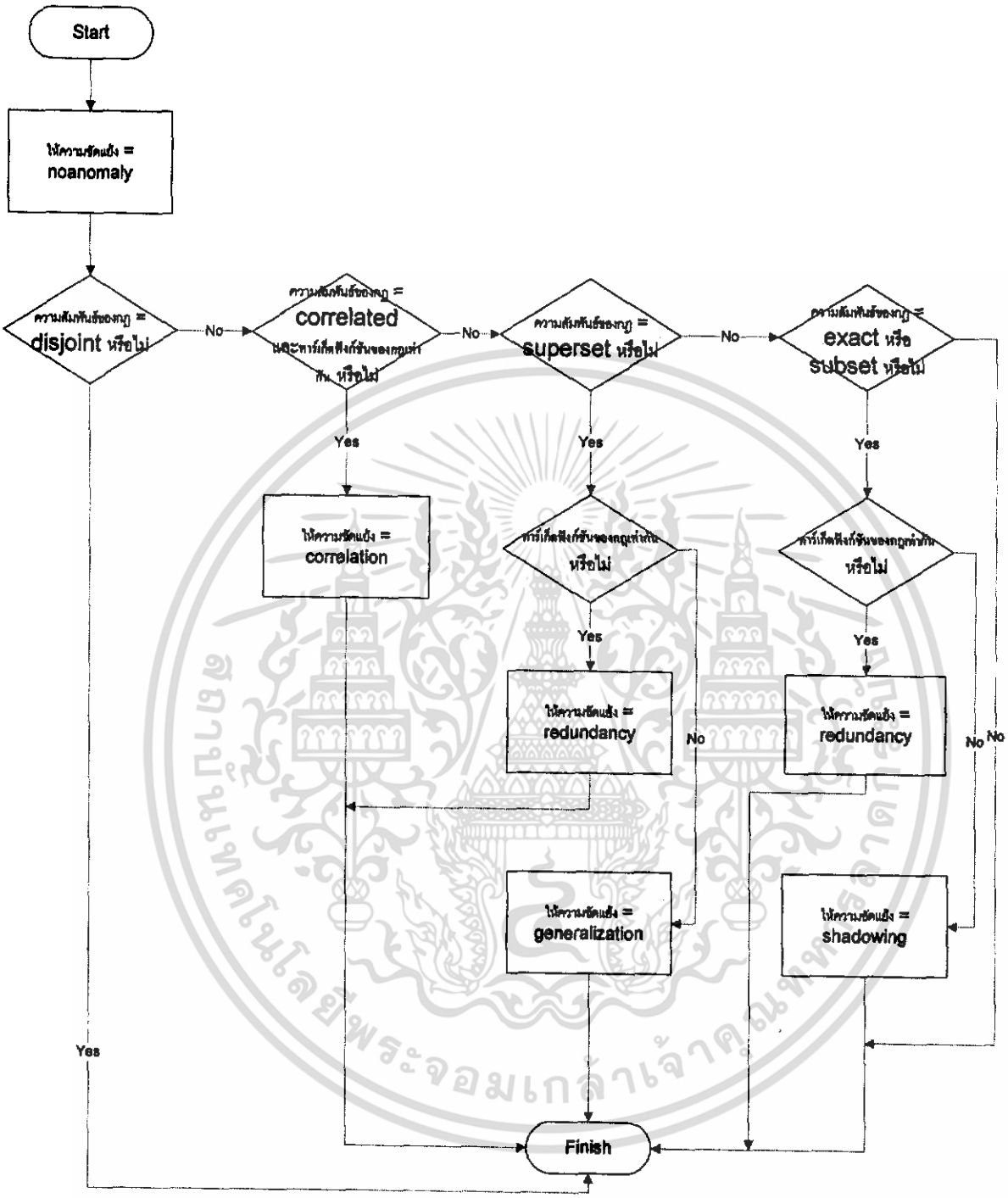
ถ้าความสัมพันธ์ของฟิลด์เป็น superset และความสัมพันธ์ของกฎเป็น subset หรือ correlated จะได้ความสัมพันธ์ของกฎเป็น correlated

ถ้าความสัมพันธ์ของฟิลด์เป็น superset และความสัมพันธ์ของกฎเป็น superset จะคงความสัมพันธ์ของกฎค่าเดิมไว้

ถ้าความสัมพันธ์ของฟิลด์เป็น subset และความสัมพันธ์ของกฎเป็น superset หรือ correlated จะได้ความสัมพันธ์ของกฎเป็น correlated

ถ้าความสัมพันธ์ของฟิลด์เป็น subset และความสัมพันธ์ของกฎเป็น subset จะคงความสัมพันธ์ของกฎค่าเดิมไว้

ถ้าความสัมพันธ์ของฟิลด์เป็น disjoint จะได้ค่าความสัมพันธ์ของกฎเป็น disjointed



รูปที่ 3-3 แสดงขั้นตอนการสรุปหาความขัดแย้งระหว่างกฎ 2 กฎ

3. ส่วนที่ทำหาความขัดแย้งระหว่างกฎ 2 กฎของไฟล์วอลล์ ซึ่งจะใช้ความสัมพันธ์ของกฎและทาร์เก็ตฟังก์ชันในการพิจารณาโดยมีขั้นตอนการเปรียบเทียบความสัมพันธ์ของกฎดังนี้

ถ้าความสัมพันธ์ของกฎเป็น correlated และทาร์เก็ตฟังก์ชันไม่เหมือนกันจะสามารถสรุปความขัดแย้งได้แบบ correlation

ถ้าความสัมพันธ์ของกฎเป็น superset และทาร์เก็ตฟังก์ชันเหมือนกันจะสามารถสรุปความขัดแย้งได้แบบ redundancy

ถ้าความสัมพันธ์ของกฎเป็น superset และทาร์เก็ตฟังก์ชันไม่เหมือนกันจะสามารถสรุปความขัดแย้งได้แบบ generalization

ถ้าความสัมพันธ์ของกฎเป็น exact หรือ subset และทาร์เก็ตฟังก์ชันเหมือนกันจะสามารถสรุปความขัดแย้งได้แบบ redundancy

ถ้าความสัมพันธ์ของกฎเป็น exact หรือ subset และทาร์เก็ตฟังก์ชันไม่เหมือนกันจะสามารถสรุปความขัดแย้งได้แบบ shadowing

กรณีอื่นๆ จะไม่เกิดความขัดแย้งของกฎใดๆ

3.2 ส่วนที่ทำการตรวจจับการใช้งานภายในเครือข่าย

ทำการตรวจจับการใช้งานภายในเครือข่าย ซึ่งจากการศึกษาในภาคเรียนที่ 1 นั้นพบว่าการใช้ล็อคไฟล์ของไอพีเทเบิ้ลส์นั้น สามารถตรวจจับการใช้งานตามรายละเอียดของกฎได้อย่างถูกต้องโดยการกำหนดรายละเอียดของกฎที่มีการเก็บการใช้งานภายในเครือข่ายสามารถกำหนดให้เหมือนกับกฎที่เราต้องการจะทราบอัตราการใช้กฎนั้นๆ ได้ แต่เนื่องจากกฎประเภทนี้จะไม่ได้ใช้ในช่วงเวลาปกติ เมื่อตรวจจับการใช้งานภายในเครือข่ายชั่วระยะเวลาหนึ่งแล้ว จะต้องมีการนำกฎประเภทนี้ออกในภายหลังด้วย ซึ่งในส่วนนี้จะมีการแบ่งการทำงานของโปรแกรมเป็นย่อยๆ อยู่ 2 ส่วนคือ

3.2.1 ขั้นตอนการทำงานของโปรแกรมใส่กฎที่บันทึกรายละเอียดของแพ็คเก็ตของข้อมูลลงล็อคไฟล์

1) ทำการเก็บกฎของไฟร์วอลล์ลงไฟล์ล็อค

2) ทำการจับจำนวนกฎในไฟล์ล็อคเพื่อนำไปใช้ในการเขียนโปรแกรมส่วนการใส่กฎที่จะบันทึกรายละเอียดของแพ็คเก็ตข้อมูล

3) ทำการใส่กฎที่บันทึกรายละเอียดของแพ็คเก็ตข้อมูลลงล็อคไฟล์โดยใช้คำสั่ง iptables –I จนกระทั่งครบทุกกฎ

4) ทำการเซฟกฎของไฟร์วอลล์โดยใช้คำสั่ง iptables-save

สามารถสั่งให้โปรแกรมทำงานในส่วนนี้โดยใช้คำสั่ง

>./optimize log <chain = {INPUT,OUTPUT,FORWARD,ALL}>

รูปที่ 3-4 Flow Chart แสดงการทำงานส่วนของโปรแกรมใส่กฎที่บันทึกรายละเอียดของแพ็คเก็ตของข้อมูลลงล็อคไฟล์

3.2.2 ขั้นตอนการทำงานของโปรแกรมลบกฎที่บันทึกรายละเอียดของแพ็คเก็ตของข้อมูลลงล๊อกไฟล์

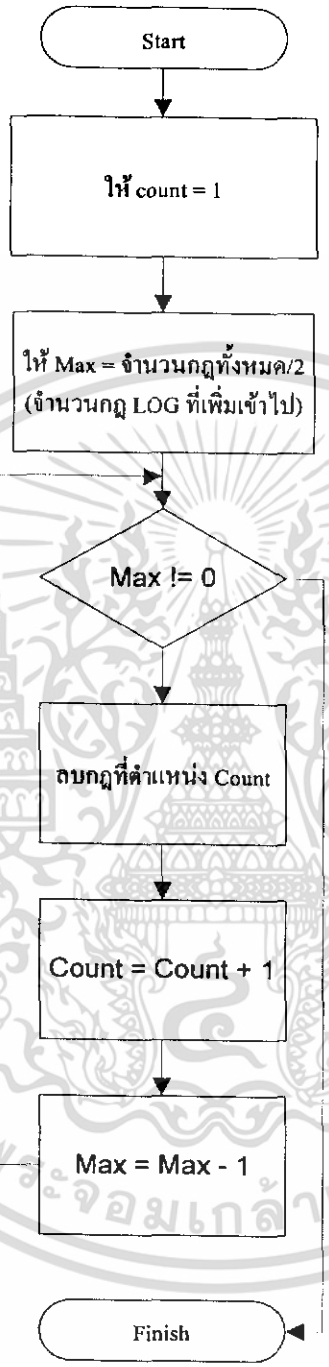
1) ทำการเก็บกฎทั้งหมดลงไฟล์ชื่อ temp เพื่อให้โปรแกรมใช้อ่านในการทำงานขั้นต่อไป

2) ทำการคำนวณจำนวนของกฎที่จะเหลือหลังจากการลบกฎที่บันทึกรายละเอียดของแพ็คเก็ตของข้อมูลลงล๊อคไฟล์ออก นั่นคือจำนวนกฎจะลดลงครึ่งหนึ่ง

3) ทำการลบกฎเรียงลำดับจากบนลงล่าง

4) ทำการเซฟกฎของไฟร์วอลล์โดยใช้คำสั่ง iptables-save

ในส่วนนำเอากฎออกนี้จะถูกเรียกใช้งานในตอนที่สั่งให้ทำการจัดลำดับกฎ



รูปที่ 3-5 Flow Chart แสดงการทำงานของโปรแกรมลบกฎที่บันทึกรายละเอียดของแพ็คเก็ตของข้อมูลลงล๊อคไฟล์

3.3 ส่วนคำนวณการใช้งานภายในเครือข่ายของแต่ละกฎจากไฟล์ล็อคที่เก็บรายละเอียดของแพ็คเก็ตข้อมูลเอาไว้

เป็นการคำนวณการใช้งานเครือข่ายโดยอ่านจากไฟล์ล็อคที่กฎที่ได้ใส่ไว้ในข้อ 2 นั้นได้บันทึกแพ็คเก็ตที่มีรายละเอียดตรงกับกฎข้อนั้นๆ โดยใช้ log-prefix ในการแบ่งแยกแต่ละแพ็คเก็ตว่ามาจากกฎข้อไหน ทำให้เราทราบถึงปริมาณการใช้งานของกฎแต่ละข้อ

ขั้นตอนการทำงานของโปรแกรม

1) ทำการอ่านปริมาณการใช้งานจากล็อคไฟล์

2) คำนวณปริมาณการใช้งานภายในเครือข่ายของแต่ละกฎแล้วทำการจัดลำดับเก็บลงไฟล์ชื่อ traffic

ในส่วนคำนวณปริมาณการใช้งานของเครือข่ายจะถูกเรียกใช้งานตอนที่สั่งให้ทำการจัดลำดับกฎใหม่

3.4 ส่วนที่ทำการจัดลำดับของกฎในแต่ละเชนเรียงตามลำดับการใช้งานภายในเครือข่ายของแต่ละกฎ

ลำดับของกฎในการตรวจสอบรายละเอียดของแพ็คเก็ตนั้น มีผลโดยรวมกับความเร็วในการทำงานมาก เนื่องจากถ้ากฎที่มีการใช้งานบ่อยๆ นั้นมีลำดับที่อยู่ด้านล่างมากๆ ในการตรวจสอบรายละเอียดของแพ็คเก็ตกับรายละเอียดของกฎนั้น จะใช้เวลาทำงานมากขึ้นยิ่งมีจำนวนแพ็คเก็ตที่ใช้งานกฎประเภทนี้มากเท่าไหร่ ก็จะใช้เวลายิ่งนานมาก

ขั้นตอนการทำงานของโปรแกรม

1) ทำการอ่านกฎเพื่อนำไปเก็บในไฟล์ชื่อ temp

2) ทำการเรียงลำดับปริมาณการใช้งาน แล้วบันทึกลงไฟล์ชื่อ traffic1 โดย2)เรียงลำดับจากน้อยไปหามาก

3) ทำการอ่านกฎทั้งหมดเพื่อเก็บลงไฟล์ชื่อ temp1

4) ทำการอ่านหมายเลขของกฎที่มีปริมาณการใช้งานในเครือข่ายมากที่สุด จากไฟล์ traffic1

5) ทำการลบกฎทั้งหมดในไฟร์วอลล์

6) ทำการอ่านกฎจากไฟล์ข้อ 1)

7) จาก 5) นำกฎที่อ่านไปต่อท้ายกฎในไฟร์วอลล์

8) ทำการอ่านหมายเลขของกฎของไฟล์ traffic1

9) ทำการอ่านกฎจากไฟล์ข้อ 3) จากลำดับบนสุดก่อน

10) ทำการตรวจสอบความสัมพันธ์ในแต่ละส่วนของรายละเอียดของแพ็คเก็ต ซึ่งจะได้ความสัมพันธ์ระหว่างกฎทั้งหมด

11) นำกฎที่ได้จากการอ่านข้อ 10) ไปตรวจหาความผิดปกติของกฎ

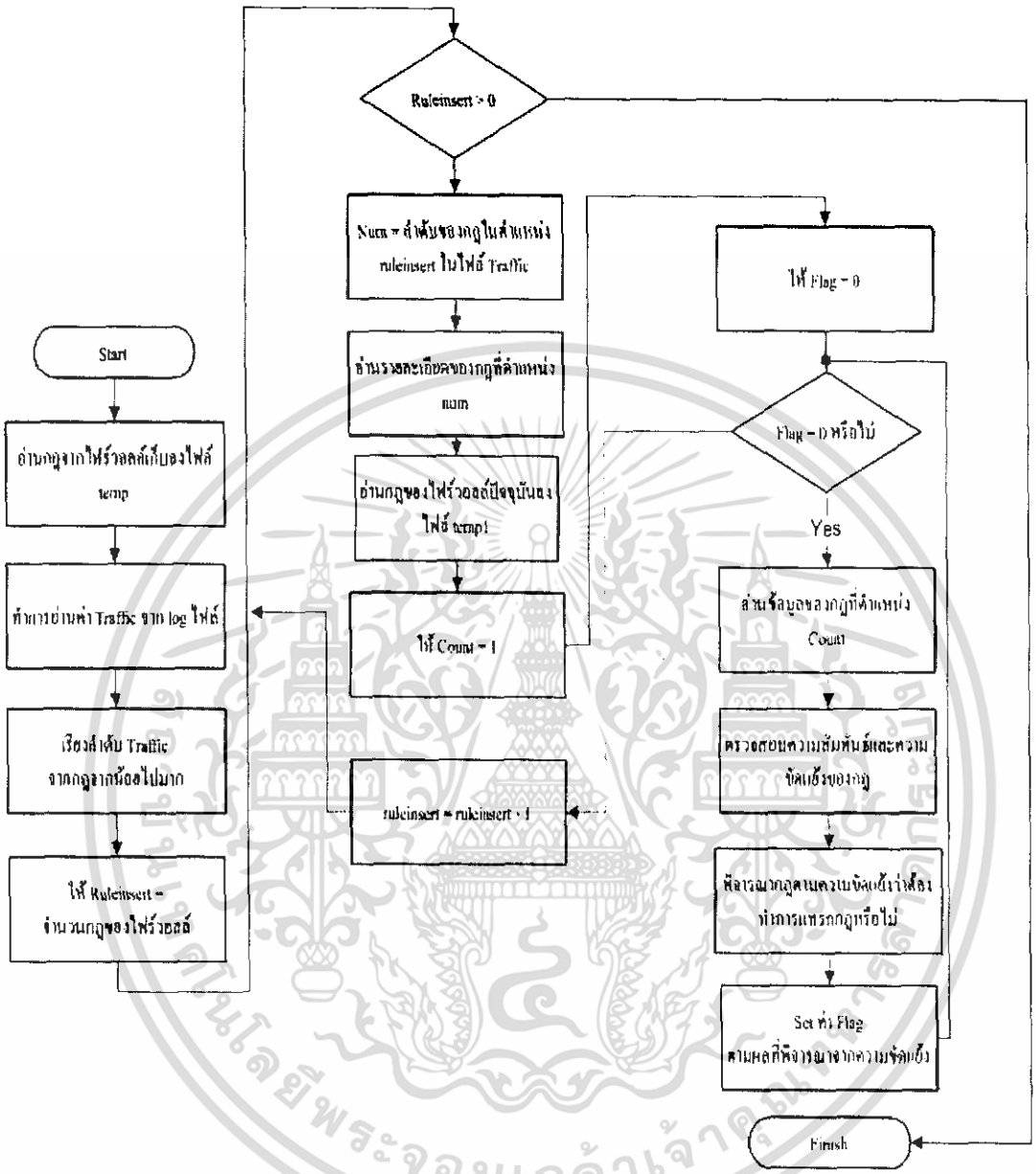
12) ถ้ามีความผิดปกติของกฎก็จะทำตามข้อบังคับในการจัดลำดับกฎ แล้วจึงทำการอ่านกฎต่อๆ ไปตามข้อ 9) ไปเรื่อยๆ

13) ทำการใส่กฎแล้วเก็บลงไฟล์ temp1

14) ทำการเซฟไฟร์วอลล์โดยใช้คำสั่ง iptables-save

สามารถสั่งให้โปรแกรมทำงานในส่วนนี้โดยใช้คำสั่ง

>./optimize reorder <chain = {INPUT,OUTPUT,FORWARD,ALL}>



รูปที่ 3-6 Flow Chart แสดงการจัดลำดับของกฎในแต่ละเซนเรียงตามลำดับการใช้งานภายในเครือข่ายของแต่ละกฎ

ในการลบกฎที่มีความซ้ำซ้อนหรือทำการจัดลำดับกฎใหม่ตามปริมาณของแพ็คเก็ตที่ผ่านกฎของไฟร์วอลล์ในระบบเครือข่ายนั้น จะมีข้อบังคับซึ่งจะพิจารณาจากความสัมพันธ์และความผิดปกติของแต่ละกฎที่ได้ทำการศึกษามาตั้งแต่ตอนภาคเรียนที่ 1 ซึ่งจะต้องทำตามเพื่อรักษาความหมายของกฎไม่ให้เปลี่ยน แพ็คเก็ตใดๆ ที่โดนบล็อกไม่ให้ผ่าน ก็ยังคงไม่ผ่านเช่นเดิม การติดต่อแบบใดที่อนุญาตให้ผ่านก็ยังอนุญาตเหมือนเดิมไม่เปลี่ยนแปลง ซึ่งมีรายละเอียดของข้อบังคับดังนี้

3.5 ข้อบังคับในการลบกฎออกจากเชนของไฟร์วอลล์

ให้กฎ Rx มาก่อนกฎ Ry

1. เมื่อกฎมีความขัดแย้งกันแบบ redundancy และมีทาร์เก็ตฟังก์ชันเหมือนกัน
   - และมีความสัมพันธ์กันแบบ exact จะสามารถลบกฎ Ry ออกได้
   - และมีความสัมพันธ์กันแบบ subset จะสามารถลบกฎ Ry ออกได้
   - และมีความสัมพันธ์กันแบบ superset จะสามารถลบกฎ Rx ออกได้
2. เมื่อกฎมีความขัดแย้งกันแบบ correlation และมีทาร์เก็ตฟังก์ชันไม่เหมือนกัน
   - และมีความสัมพันธ์กันแบบ correlate จะไม่สามารถลบกฎใดๆ ได้
3. เมื่อกฎมีความขัดแย้งกันแบบ generalization และมีทาร์เก็ตฟังก์ชันไม่เหมือนกัน
   - และมีความสัมพันธ์แบบ superset จะไม่สามารถลบกฎใดๆ ได้
4. เมื่อกฎมีความขัดแย้งกันแบบ shadowing และมีทาร์เก็ตฟังก์ชันไม่เหมือนกัน
   - และมีความสัมพันธ์กันแบบ exact จะสามารถลบกฎ Ry ออกได้
   - และมีความสัมพันธ์กันแบบ subset จะสามารถลบกฎ Ry ออกได้
5. เมื่อกฎไม่มีความขัดแย้งกัน ก็จะไม่มีการลบกฎใดๆ ออก

3.6 ข้อบังคับในการจัดลำดับของกฎตามปริมาณการใช้งานของกฎในเครือข่าย

ให้กฎ Rx มาก่อนกฎ Ry

1. เมื่อกฎไม่มีความขัดแย้งกัน จะทำการตรวจสอบกับกฎข้อถัดไปในไฟล์
2. เมื่อกฎมีความขัดแย้งกันแบบ shadowing และมีทาร์เก็ตฟังก์ชันไม่เหมือนกัน
   - และมีความสัมพันธ์กันแบบ subset จะทำการตรวจสอบกับกฎข้อถัดไป ถ้าเป็นกฎข้อสุดท้ายแล้ว ก็จะทำการนำกฎไปต่อท้ายเชน
3. เมื่อกฎมีความขัดแย้งกันแบบ generalization และมีทาร์เก็ตฟังก์ชันไม่เหมือนกัน

- และมีความสัมพันธ์กันแบบ subset จะทำการแทรกกฎที่ตำแหน่ง Ry

4. เมื่อกฎมีความขัดแย้งกันแบบ correlation และมีทาร์เก็ตฟังก์ชันไม่เหมือนกัน

- และมีความสัมพันธ์กันแบบ correlate ถ้าทาร์เก็ตฟังก์ชันมีค่า

DROP จะทำการแทรกกฎที่ตำแหน่ง Ry

ACCEPT จะทำการตรวจสอบกับกฎถัดไป ถ้าเป็นกฎข้อสุดท้ายแล้วก็จะทำการนำกฎไปต่อท้ายเซน

5. เมื่อกฎมีความขัดแย้งกันแบบ redundancy และมีทาร์เก็ตฟังก์ชันเหมือนกัน

- และมีความสัมพันธ์กันแบบ subset จะทำการตรวจสอบกับกฎข้อถัดไป ถ้าเป็นกฎข้อสุดท้ายแล้ว ก็จะทำการนำกฎไปต่อท้ายเซน

- และมีความสัมพันธ์กันแบบ superset จะทำการแทรกกฎที่ตำแหน่ง Ry

บทที่ 4

การทดลองและผลการทดลอง

บทนี้จะแสดงถึงกฎของไฟร์วอลล์ที่ใช้ในการทดลองปรับปรุงการทำงาน โดยผลที่ได้จะแสดงถึงการทำงานของตัวส่วนโปรแกรมและผลที่ได้หลังจากการปรับปรุงแล้ว โดยใช้โปรแกรมดักจับแพ็คเก็ตมาตรวจจับหาอัตราการวิ่งของแพ็คเก็ตภายในระบบเครือข่าย

4.1 ตัวอย่างกฎที่นำมาใช้ในการทดลอง

```
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            192.168.3.0/24
DROP       tcp  --  0.0.0.0/0            192.168.3.2         tcp dpt:80
DROP       udp  --  192.168.3.2          0.0.0.0/0           udp spt:63
ACCEPT     udp  --  192.168.3.0/24       192.168.5.0/24
ACCEPT     tcp  --  192.168.3.0/24       0.0.0.0/0           tcp spt:20 dpt:20
ACCEPT     tcp  --  192.168.3.0/24       0.0.0.0/0
DROP       tcp  --  192.168.1.0/24       192.168.5.0/24
ACCEPT     udp  --  192.168.5.0/24       0.0.0.0/0
DROP       tcp  --  192.168.3.0/24       0.0.0.0/0           tcp dpt:80
DROP       icmp --  192.168.5.7          192.168.3.2
ACCEPT     tcp  --  192.168.2.0/24       192.168.7.0/24
DROP       all  --  0.0.0.0/0            0.0.0.0/0
```

รูปที่ 4-1 แสดงกฎที่นำมาใช้ในการทดลอง

ในขั้นต้นตัวโปรแกรมจะทำการสร้างกฎที่เหมือนกับกฎดังกล่าวทั้ง 10 กฎข้างต้น แต่ในส่วนการกระทำของกฎนั้นจะอยู่ในลักษณะของล็อคไฟล์ นั่นคือจะทำการบันทึกรายละเอียดของแพ็คเก็ตที่มีรายละเอียดตรงกับกฎนี้ ทำให้เราสามารถทราบถึงความมากน้อยในการใช้งานของเครือข่ายของแต่ละกฎของไอพีเทเบิ้ลส์ ซึ่งข้อมูลที่ได้นั้นจะนำไปใช้ในการจัดลำดับกฎของไอพีเทเบิ้ลส์ต่อไป โดยเรียงตามลำดับการใช้งานเครือข่าย ทั้งนี้เมื่อจัดลำดับเรียบร้อยแล้วความหมายของกฎต้องไม่เปลี่ยนแปลง

4.2 ทำการปรับปรุงความเร็วในการทำงานของไฟร์วอลล์

ในขั้นแรก ตัวโปรแกรมจะทำการตัดกฎที่มีความสัมพันธ์กันแบบซ้ำซ้อนกันออกก่อน เนื่องจากกฎที่ซ้ำซ้อนกัน กฎที่อยู่ด้านล่างจะไม่ถูกใช้อย่างแน่นอน จึงทำให้ผลที่ได้ในขั้นแรกนั้นเหลือกฎดังนี้

```
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     icmp --  0.0.0.0/0            192.168.3.0/24
DROP       tcp  --  0.0.0.0/0            192.168.3.2          tcp dpt:80
DROP       udp  --  192.168.3.2          0.0.0.0/0            udp spt:63
ACCEPT     udp  --  192.168.3.0/24       192.168.5.0/24
ACCEPT     tcp  --  192.168.3.0/24       0.0.0.0/0
DROP       tcp  --  192.168.1.0/24       192.168.5.0/24
ACCEPT     udp  --  192.168.5.0/24       0.0.0.0/0
ACCEPT     tcp  --  192.168.2.0/24       192.168.7.0/24
DROP       all  --  0.0.0.0/0            0.0.0.0/0
```

รูปที่ 4-2 แสดงกฎหลังทำการลบกฎที่ซ้ำซ้อน

เมื่อทำการลบกฎแล้ว ตัวโปรแกรมจะมีการเก็บรายละเอียดของกฎที่ถูกลบออกไปลงในไฟล์ชื่อ removercport

```
*************************
FORWARD
*************************
------------------------------------------------------------
Rule target = ACCEPT protocol = tcp
source = 192.168.3.0/24 destination = 0.0.0.0/0
source port = all destination port = all
SHADOWING
Rule target = DROP protocol = tcp
source = 192.168.3.0/24 destination = 0.0.0.0/0
source port = all destination port = 80
Rule target = DROP protocol = tcp
source = 192.168.3.0/24 destination = 0.0.0.0/0 REMOVED
source port = all destination port = 80
------------------------------------------------------------
```

รูปที่ 4-3 แสดงรายละเอียดของกฎที่ถูกลบออกไปในไฟล์ removereport

หลังจากนั้นตัวโปรแกรมจะทำงานต่อในส่วนของการลำดับกฎใหม่ตามการใช้งานของแต่ละกฎในเครือข่าย เพื่อลดเวลาในการเปรียบเทียบรายละเอียดของแพ็คเก็ตกับกฎของไอพีเทเบิ้ลส์ ซึ่งได้ผลดังนี้

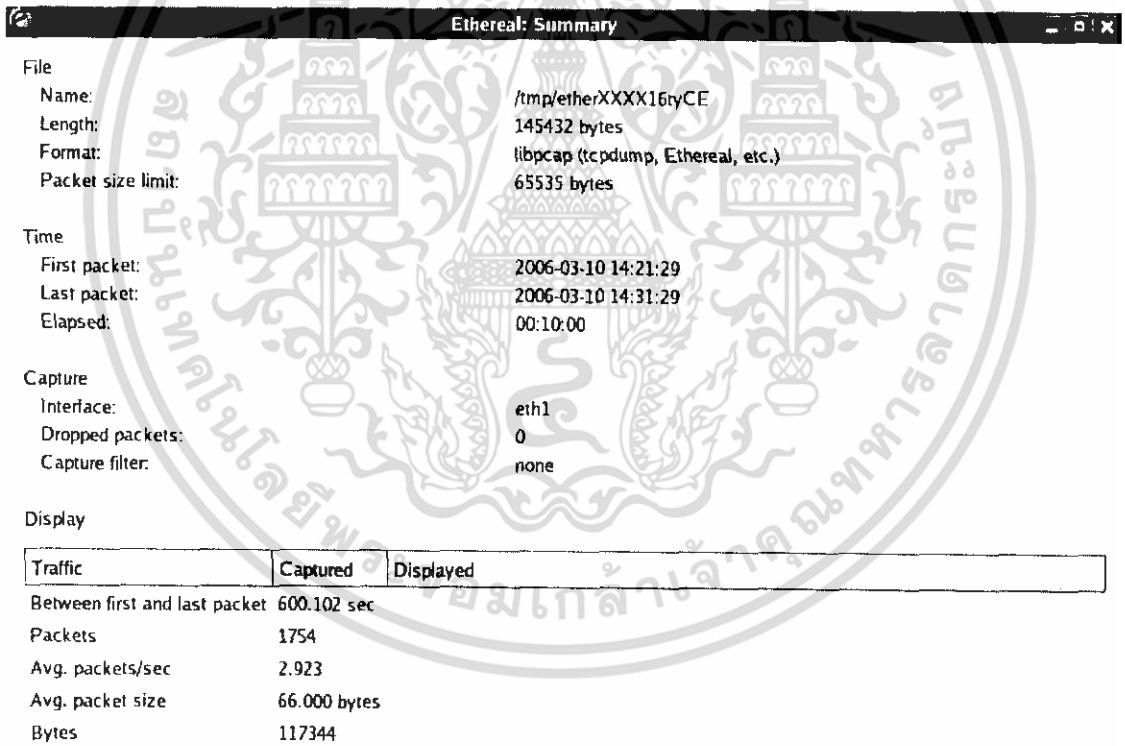
```
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.2.0/24       192.168.7.0/24
ACCEPT     udp  --  192.168.5.0/24       0.0.0.0/0
DROP       tcp  --  192.168.1.0/24       192.168.5.0/24
DROP       tcp  --  0.0.0.0/0            192.168.3.2         tcp dpt:80
ACCEPT     tcp  --  192.168.3.0/24       0.0.0.0/0
DROP       udp  --  192.168.3.2          0.0.0.0/0           udp spt:63
ACCEPT     udp  --  192.168.3.0/24       192.168.5.0/24
ACCEPT     icmp --  0.0.0.0/0            192.168.3.0/24
DROP       all  --  0.0.0.0/0            0.0.0.0/0
```

รูปที่ 4-4 แสดงกฎหลังจากทำการจัดลำดับใหม่

4.3 ทดสอบการเปลี่ยนแปลงหลังทำการปรับปรุง

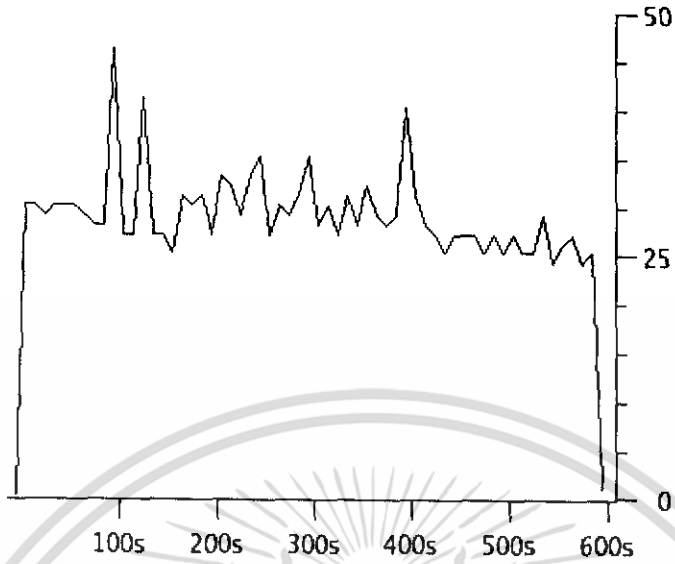
หลังจากนั้นได้ทำการใช้โปรแกรมสร้างแพ็คเก็ตส่งมายังเครื่องไฟร์วอลล์ตัวนี้เพื่อทำการตรวจจับอัตราการวิ่งของแพ็คเก็ตภายในระบบเครือข่าย

ผลการตรวจจับปริมาณการใช้งานภายในเครือข่ายของกฎตอนเริ่มแรก



รูปที่ 4-5 แสดงผลสรุปการจับแพ็คเก็ตของกฎเริ่มต้น

มีจำนวนแพ็คเก็ตที่วิ่งผ่านทั้งหมด 1754 แพ็คเก็ต เฉลี่ย 2.923 แพ็คเก็ตต่อวินาที



รูปที่ 4-6 กราฟแสดงจำนวนแพ็คเก็ตที่วิ่งผ่านในระบบของกฎเริ่มต้น

ปริมาณการใช้งานในเครือข่ายของกฎหลังทำการลบกฎที่ซ้ำซ้อน

Ethereal: Summary

File

Name:	/tmp/etherXXXXLHu6ZA
Length:	151888 bytes
Format:	libpcap (tcpdump, Ethereal, etc.)
Packet size limit:	65535 bytes

Time

First packet:	2006-03-10 15:36:01
Last packet:	2006-03-10 15:46:01
Elapsed:	00:10:00

Capture

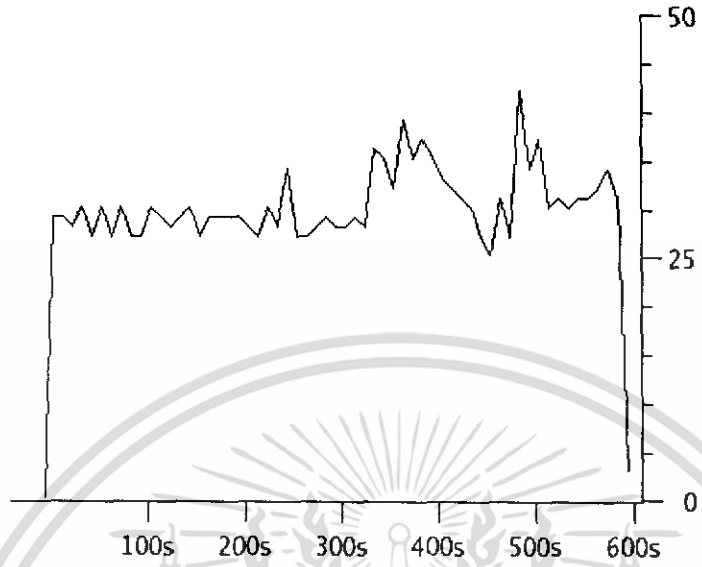
Interface:	eth1
Dropped packets:	0
Capture filter:	none

Display

Traffic	Captured	Displayed
Between first and last packet	600.642 sec	
Packets	1824	
Avg. packets/sec	3.037	
Avg. packet size	67.000 bytes	
Bytes	122680	

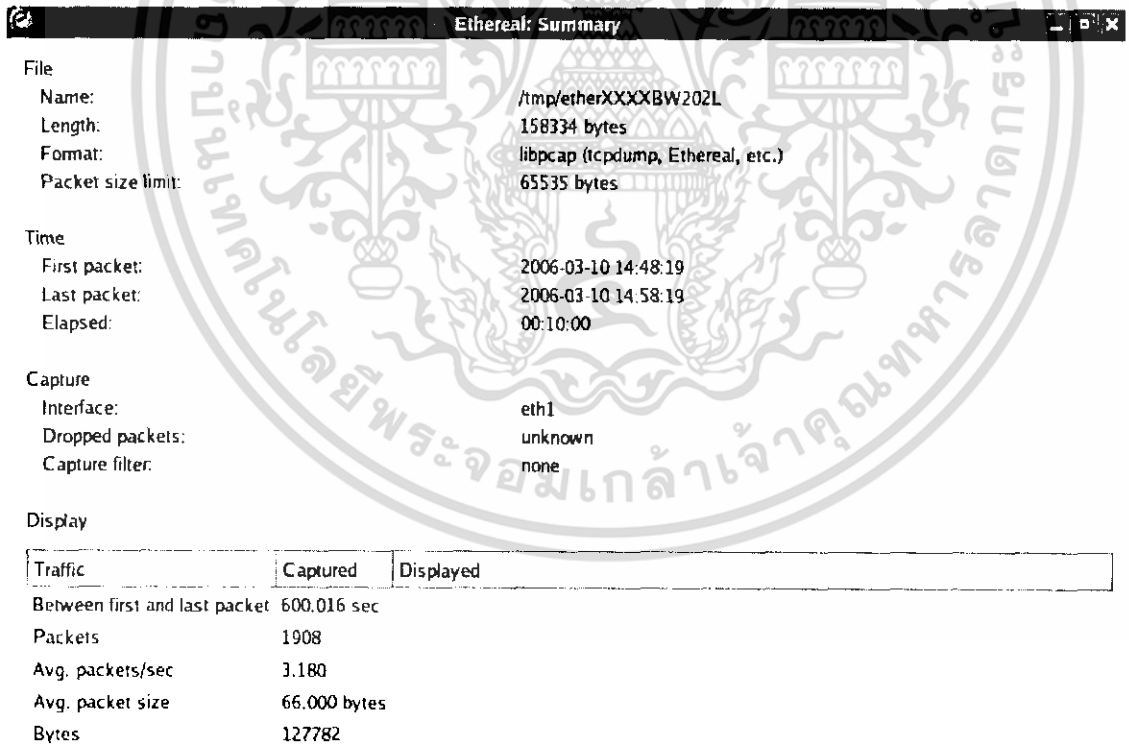
รูปที่ 4-7 แสดงผลสรุปการจับแพ็คเก็ตหลังทำการลบกฎที่ซ้ำซ้อน

มีจำนวนแพ็คเก็ตที่วิ่งผ่านทั้งหมด 1824 แพ็คเก็ต เฉลี่ย 3.037 แพ็คเก็ตต่อวินาที



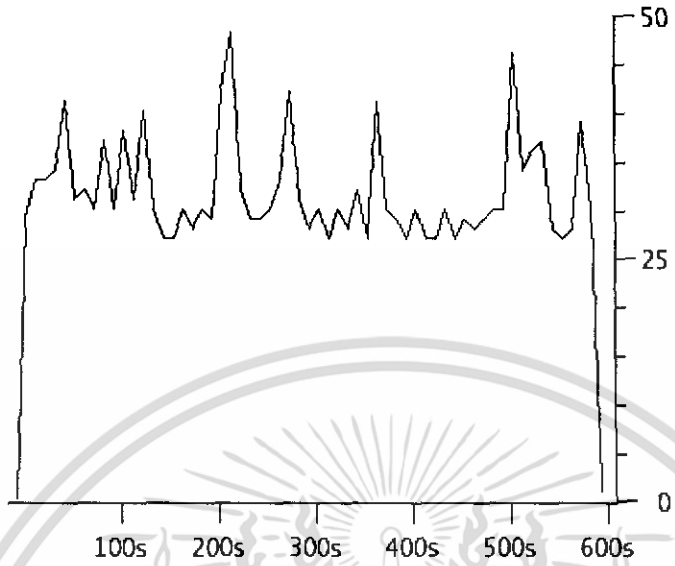
รูปที่ 4-8 กราฟแสดงจำนวนแพ็คเก็ตหลังทำการลบกฎที่ซ้ำซ้อน

ปริมาณการใช้งานในเครือข่ายของกฎหลังทำการจัดลำดับกฎใหม่



รูปที่ 4-9 แสดงผลสรุปการจับแพ็คเก็ตหลังทำการจัดลำดับของกฎใหม่

มีจำนวนแพ็คเก็ตที่วิ่งผ่านทั้งหมด 1908 แพ็คเก็ต เฉลี่ย 3.180 แพ็คเก็ตต่อวินาที



รูปที่ 4-10 กราฟแสดงจำนวนแพ็คเก็ตหลังทำการจัดลำดับของกฎใหม่

4.4 สรุปผลการทดลอง

กฎ	จำนวนแพ็คเก็ต	แพ็คเก็ตเฉลี่ย	จำนวนแพ็คเก็ตเพิ่มขึ้น	คิดเป็น
กฎเริ่มต้น	1754	2.923	-	-
ลบกฎที่ซ้ำซ้อน	1824	3.037	70	4%
จัดลำดับของกฎใหม่	1938	3.18	184	10.5%

การทดลองนี้ใช้โปรแกรมสร้างแพ็คเก็ตเป็นตัวสร้างการใช้งานภายในเครือข่ายขึ้น และใช้โปรแกรมดักจับแพ็คเก็ต ethereal ตรวจจับเป็นระยะเวลา 10 นาที พบว่าเมื่อทำการปรับปรุงการทำงานของไฟร์วอลล์ทั้งการลบกฎที่ซ้ำซ้อนและจัดลำดับของกฎใหม่นั้น ทำให้มีจำนวนแพ็คเก็ตที่ผ่านเน็ทเวิร์คอินเทอร์เฟสเพิ่มขึ้นทั้ง 2 วิธี

บทที่ 5

บทวิจารณ์และสรุป

5.1 บทสรุป

เนื่องด้วยการรักษาความปลอดภัยในระบบเครือข่ายคอมพิวเตอร์ในปัจจุบันนั้น องค์กรทุกๆ องค์กรนั้นจะใช้ตัวไฟร์วอลล์ในการรักษาระบบความปลอดภัย ซึ่งเรียกได้ว่าระบบเครือข่ายนั้นจะขาดไฟร์วอลล์เสียไม่ได้ โดยเฉพาะไอพีเทเบิ้ลส์เป็นไฟร์วอลล์พื้นฐานที่มีอยู่ในระบบปฏิบัติการลินุกซ์ ซึ่งมีการใช้งานกันอย่างแพร่หลาย ด้วยเหตุผลนี้สิ่งสำคัญคือไฟร์วอลล์ควรจะทำงานได้อย่างมีประสิทธิภาพมากที่สุด

ไฟร์วอลล์เป็นโปรแกรมหนึ่งที่มีความสำคัญกับระบบคอมพิวเตอร์ การทำงานของไฟร์วอลล์จะทำงานโดยใช้กฎเพื่ออนุญาตหรือไม่อนุญาตการติดต่อจากเครื่องอื่นๆ โดยจะทำงานตามกฎจากบนลงล่าง ซึ่งกลุ่มผู้วิจัยได้มีการเล็งเห็นแล้วว่าหากมีกฎจำนวนมาก จะทำให้การทำงานต่างๆ ของไฟร์วอลล์นั้นทำงานช้าลง โดยโครงงานนี้จะศึกษาวิธีการต่างๆ ที่จะปรับปรุงการทำงานของไฟร์วอลล์ให้มีการทำงานที่เร็วขึ้น ซึ่งจะเน้นส่วนที่เป็นลำดับต่างๆ ของกฎในแต่ละตารางและด้านความซ้ำซ้อนต่างๆ ของกฎ ที่จะพิจารณาถึงความสัมพันธ์ของกฎที่ซ้ำซ้อนกันว่าสามารถลบออกไปได้หรือไม่ โดยไม่ให้มีผลกระทบต่อความหมายโดยรวมของกฎทั้งหมด

5.2 วิจารณ์สิ่งที่ได้จากโครงงาน

ไอพีเทเบิ้ลส์เป็นสเตรทฟูลไฟร์วอลล์ซึ่งในการตรวจสอบรายละเอียดของแพ็คเก็ตกับกฎของไฟร์วอลล์นั้น จะทำการตรวจสอบกับกฎด้านบนลงมาเรื่อยๆ จนถึงด้านล่างจนกว่าจะเจอกฎที่มีรายละเอียดตรงกับแพ็คเก็ตนั้นๆ เมื่อตรงกับกฎใดนั้นจะไปทำตามทาร์เก็ตฟังก์ชันของกฎนั้น ในการปรับปรุงความเร็วในการทำงานของไฟร์วอลล์นั้น ใช้หลักการตรวจสอบรายละเอียดแพ็คเก็ตของสเตรทฟูลไฟร์วอลล์มาใช้ในการปรับปรุง นั่นคือเมื่อแพ็คเก็ตตรงกับกฎนั้น ก็จะไปทำตามทาร์เก็ตฟังก์ชันของกฎนั้นๆ และจะไม่ทำการตรวจสอบกับกฎใดๆ ด้านล่างต่อไปอีกเลย ทำให้สามารถทำการลบกฎที่มีรายละเอียดและทาร์เก็ตฟังก์ชันของกฎซ้ำซ้อนกัน เนื่องจากกฎดังกล่าวจะไม่มีการนำไปใช้อย่างแน่นอนและเนื่องมาจากลักษณะการตรวจสอบกฎนั้นเป็นไปแบบจากบนลงล่างดังนั้นลำดับของกฎจึงมี

ความสำคัญ ถ้ากฎที่มีรายละเอียดตรงกับแพ็คเก็ตของข้อมูลที่มีปริมาณมากๆ อยู่ด้านล่างๆ ในการตรวจสอบกฎจะต้องผ่านกฎที่ไม่ได้เกี่ยวข้องจำนวนหนึ่งจนกว่าจะถึงกฎนั้นๆ การจัดลำดับของกฎโดยให้กฎที่มีการใช้งานบ่อยๆ มาอยู่ด้านบน จะลดเวลาในการตรวจสอบแพ็คเก็ตกับกฎลงได้มาก ยิ่งแพ็คเก็ตของข้อมูลยิ่งมีปริมาณมากเท่าไหร่จะยิ่งลดเวลาในการทำงานได้มากขึ้นไปอีก แต่ในการเคลื่อนย้ายกฎนั้นจะต้องไม่ทำให้ความหมายโดยรวมของกฎทั้งหมดนั้นยังไม่เปลี่ยนแปลง

5.3 ปัญหาอุปสรรคในการทำโครงงาน

1. ระบบปฏิบัติการลินุกซ์ เป็นระบบปฏิบัติการที่ไม่เป็นที่คุ้นเคยมากนัก ทำให้ต้องมีการใช้ระยะเวลาสักพักหนึ่งเพื่อเรียนรู้การติดตั้งและใช้งานเบื้องต้นรวมถึงคำสั่งต่างๆ ที่ใช้บ่อยๆ
2. ไฟร์วอลล์ไอพีเทเบิ้ลไม่มีส่วนที่อำนวยความสะดวกในการติดต่อกับโปรแกรม (API) ในการจัดการกับกฎ ทำให้เกิดปัญหาในการที่จะเขียนโปรแกรมให้เพิ่มลบหรือจัดลำดับของกฎตามที่ต้องการ
3. ในส่วนการตรวจจับเครือข่าย ไม่สะดวกที่จะแยกแพ็คเก็ตของข้อมูลตามวันและเวลาที่ต้องการในส่วนการเก็บรายละเอียดของล็อกไฟล์

5.4 แนวทางการพัฒนาต่อ

1. พัฒนาในส่วนของโปรแกรมให้มีส่วนติดต่อกับผู้ใช้และแสดงกฎของไฟร์วอลล์ที่มีอยู่ในขณะนั้นและมีการรวบรวมรายงานการเปลี่ยนแปลงพร้อมเหตุผลที่ทำการเปลี่ยนแปลงกฎนั้นๆ แก่ผู้ใช้งาน
2. พัฒนาการปรับปรุงความเร็วในการทำงานของไฟร์วอลล์มากกว่า 1 ตัว ให้มีการสื่อสารกันระหว่างไฟร์วอลล์ทั้ง 2 ตัว

บรรณานุกรม

[1] E. Al-Shaer and H.Hamed, "Firewall Policy Advisor for Anomaly Detection and Rule Editing", *IEEE/IFIP Intergrated Management Conference (IM'2003)*, March 2003.

[2] E. Al-Shaer and H.Hamed, "Design and Implementation of Firewall Policy Advisor Tools", *Depaul CTI Technical Report, CTI-TR-02-006*, August 2002

[3] E. Al-Shaer and H.Hamed, "Modeling and Management of Firewall Policies", n *10th IEEE International Conference on Network Protocols ICNP*, November 2002

[4] นายพัฒนพล รัตนพงษ์พร, "การตรวจจับความขัดแย้งแบบรวดเร็วและขยายขนาดได้สำหรับตัวจัดกลุ่มแพ็คเก็ต" ,บัณฑิตศึกษา วิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง 2547

[5] Firewalling, NAT , and packet mangling for Linux :http://www.netfilter.org

[6] TCPDUMP public repository :http://www.tcpdump.org

[7] Libpcap Packet capture tutorial :http://www.cet.nau.edu/~mc8/Socket/Tutorials/section1.html

[8] Linux 2.4 Stateful Firewall : IPTABLES :http://thaicert.nectec.or.th/paper/firewall/iptables.php

[9] นพสรณ์ เกษจันทร์ , ศุภกร ชอุ่มดี "การป้องกันเว็บโดยใช้ไอพีเทเบิลส์" ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง 2546

[10]บุญลือ อยู่คง ป้องกัน Linux อย่างมืออาชีพ สำนักพิมพ์ The Knowledge Center กรุงเทพฯ

ภาคผนวก ก การติดตั้ง

เนื่องจากส่วนโค้ดของโปรแกรมจะใช้ภาษาเชลล์สคริปต์ เนื่องจากเป็นภาษาที่เรียนรู้เข้าใจง่าย และสามารถเรียกคำสั่งต่างๆ บนระบบปฏิบัติการณ์ลินุกซ์ ให้ทำงานควบคู่กันไปได้ง่าย อีกทั้งยังไม่จำเป็นต้องทำการติดตั้งหรือคอมไฟล์แพ็คเก็ตแต่อย่างใด

ตัวงานจะเป็นไฟล์ที่ทำการบีบอัดเอาไว้ชื่อ optimize.tar.gz ทำการแตกไฟล์โดยใช้คำสั่ง

```
tar -zxvf optimize.tar.gz
```

จะได้ผลออกมาเป็นโฟลเดอร์ที่มีชื่อว่า optimize ก็เป็นอันเสร็จการติดตั้งเรียบร้อย

ภาคผนวก ข การใช้งาน

ตัวโปรแกรมจะเป็นภาษาเชลล์สคริปต์ดังนั้นต้องทำการเรียกทำงานแบบเชลล์สคริปต์ นั่นคือคำสั่ง source หรือ ./ ก็ได้ตามด้วยออฟชั่น 2 ตัว

รูปแบบของคำสั่ง

./optimize <options1={remove,log,reorder}> <options2={input,output,forward}>

ในส่วนของ options1 นั้นจะเป็นการบอกกับโปรแกรมว่าให้กระทำการใด ซึ่งมีอยู่ 3 อย่างคือ

- remove ตัวโปรแกรมจะทำการตรวจสอบกฎที่ซ้ำซ้อนและลบกฎที่ซ้ำซ้อนนั้นออก
- log ตัวโปรแกรมจะทำการใส่กฎที่มีรายละเอียดเหมือนกับแต่ละกฎที่มีอยู่ในไฟร์วอลล์ไอพีเทเบิ้ลส์อยู่แล้ว แต่ทาร์เก็ตฟังก์ชันจะเป็นการเก็บรายละเอียดของแพ็คเก็ตของข้อมูลที่ผ่านกฎนั้นๆ ลงล็อคไฟล์แทน
- reorder ตัวโปรแกรมจะทำการลบกฎที่เกิดขึ้นจากออฟชั่น log ออก และคำนวณปริมาณข้อมูลภายในเครือข่ายที่ผ่านแต่ละกฎออกมาเป็นตัวเลข เพื่อใช้เปรียบเทียบกันและนำข้อมูลส่วนนี้ไปใช้ในการจัดลำดับกฎใหม่ และในส่วนของการจัดลำดับนั้น ตัวโปรแกรมจะทำการตรวจสอบความสัมพันธ์และความขัดแย้งของแต่ละกฎเพื่อไม่ให้ความหมายโดยรวมของกฎในการตรวจสอบแพ็คเก็ตนั้นเปลี่ยนแปลงไป

ในส่วนของ options2 นั้นเป็นการบอกกับโปรแกรมว่าให้กระทำการนั้นๆ กับเชนใดของไอพีเบิ้ลส์

- INPUT ตัวโปรแกรมจะกระทำการตามออฟชั่นแรกกับเชน INPUT ในไอพีเทเบิ้ลส์
- OUTPUT ตัวโปรแกรมจะกระทำการตามออฟชั่นแรกกับเชน OUTPUT ในไอพีเทเบิ้ลส์
- FORWARD ตัวโปรแกรมจะกระทำการตามออฟชั่นแรกกับเชน FORWARD ในไอพีเทเบิ้ลส์
-