การค้นหาเส้นทางเชิงนโยบายโดยใช้การรวมกฎของไฟร์วอลล์

POLICY-BASED ROUTING USING FIREWALL RULES MERGING

กวิน ตันติพงศ์สกุล

KAVIN TANTIPONGSAKUL

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2553

KMITL-2010-IT-M-001-008

POLICY-BASED ROUTING USING FIREWALL RULES MERGING

KAVIN TANTIPONGSAKUL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2010

KMITL-2010-IT-M-001-008

หัวข้อวิทยานิพนธ์	การค้นหาเส้นทางเชิงนโยบายโดยใช้การรวมกฎของไฟร์วอลล์
นักศึกษา	นายกวิน ตันติพงศ์สกุล
รหัสนักศึกษา	48066435
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
พ.ศ.	2553
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.อัครินทร์ คุณกิตติ

บทคัดย่อ

ไฟร์วอลล์เป็นอุปกรณ์สำหรับป้องกันและควบคุมการเข้าออกของจราจรในระบบเครือข่าย ซึ่งนโยบายและกฎของไฟร์วอลล์เป็นองค์ประกอบที่สำคัญในการอนุญาตให้แพ็คเก็ตที่กำหนดโดยผู้ดูแลระบบเครือข่าย อย่างไรก็ตามผู้ดูแลระบบเครือข่ายนั้นไม่สามารถรู้นโยบายและกฎของไฟร์วอลล์ฝั่งปลายทางว่ามีการกำหนดนโยบายหรือกฎอะไรบ้าง จึงทำให้การควบคุมและจัดการแบนด์วิดท์นั้นไม่เกิดประสิทธิภาพ ดังนั้นจึงมีแนวคิดการส่งผ่านและการเรียนรู้ของกฎของไฟร์วอลล์โดยการจำลองการส่งผ่านกฎของไฟร์วอลล์ โดยใช้การรวมกฎของไฟร์วอลล์ในการค้นหาเส้นทางเชิงนโยบาย ในเอกสารวิทยานิพนธ์เล่มนี้จะนำเสนอการแลกเปลี่ยนและส่งผ่านกฎของไฟร์วอลล์ โดยแนวทางนี้จะเป็นการตัดสินใจและเลือกเส้นทางที่ดีที่สุดโดยใช้กฎของไฟร์วอลล์เป็นหนึ่งในตัวเลือกสำหรับการค้นหาเส้นทางสำหรับอุปกรณ์ค้นหาเส้นทาง และได้นำเสนอการพัฒนาอัลกอริธึมการค้นหาเส้นทางเชิงนโยบายเพื่อลดความสูญเสียของแบนด์วิดท์ และใช้แบนด์วิดท์ให้เกิดประสิทธิภาพได้ดีขึ้นบนอุปกรณ์ค้นหาเส้นทางในระบบเครือข่าย ดังนั้นงานวิจัยฉบับนี้จึงสามารถค้นหาเส้นทางไปยังเส้นทางที่มีการส่งผ่านของกฎไฟร์วอลล์จึงทำให้การค้นหาเส้นทางนั้นเป็นไปได้อย่างถูกต้องตามนโยบาย ส่วนแพ็คเก็ตใดที่มีการบล็อกระหว่างทางโดยกฎไฟร์วอลล์นั้นแพ็คเก็ตจะถูกทิ้งตั้งแต่ต้นทางทันที

Thesis Title	Policy-Based Routing using Firewall Rules Merging
Student	Mr.Kavin Tantipongsakul
Student ID.	48066435
Degree	Master of Science
Affiliation	Faculty of Information Technology
Program	Information Science
Year	2010
Thesis Advisor	Assist. Prof. Akharin Khunkitti

ABSTRACT

A firewall is basically a defense device to control incoming and outgoing network traffic. Policies and rules are important components of a firewall that allow packets to pass through by network administration. However, administrators do not know what the destination policies and rules are, moreover bandwidth control and management are not efficient. A concept occurred that firewalls can transfer and learn firewall rules using policy-based routing. In this thesis we propose exchanging and transferring firewall rules. This will decide and select the best path for routers over transit autonomous systems. We also propose to develop routing protocol for reducing bandwidth and optimize traffic policies on network routers. Therefore this thesis describes the selection best path for routers over allow of firewall rules. The concept will be able to appoint policy-based routing in addition blocked packets will be dropped from the source of origin.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำและคำปรึกษาจาก ผศ. อัครินทร์ คุณกิตติ ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ข้าพเจ้ารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอกราบพระคุณคณาจารย์คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณห้องวิจัยและปฏิบัติการ DiPaC และห้องวิจัยอื่นๆ คณะเทคโนโลยีสารสนเทศ ตลอดจนข้อมูล และหนังสือต่างๆ ที่ใช้ในการทำวิจัย

ขอขอบคุณ พี่ๆ เพื่อนๆ น้องๆ ในคณะเทคโนโลยีสารสนเทศ ทุกคนที่ให้คำแนะนำต่างๆ และช่วยเหลือเสมอมา

ขอขอบคุณเจ้าหน้าที่คณะเทคโนโลยีสารสนเทศทุกๆท่านๆ ที่ให้ความช่วยเหลือในการช่วยดำเนินเรื่องต่างๆ สำหรับการสอบและจัดทำรูปเล่มวิทยานิพนธ์จนแล้วเสร็จ

ขอขอบคุณบริษัท จีเน็ท เน็ทเวิร์ค โซลูชั่น จำกัด ที่คอยจัดสรรเวลาในการทำงานให้ข้าพเจ้ามีระยะเวลาในการทำวิทยานิพนธ์ฉบับนี้

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

กวิน ตันติพงศ์สกุล

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การค้นหาเส้นทางที่ดีที่สุดเป็นปัจจัยหลักในการทำงานของเราเตอร์ การเลือกเส้นทางโดยใช้นโยบายเป็นเทคนิคที่ใช้ในการเลือกเส้นทางที่ตั้งอยู่บนพื้นฐานนโยบายของผู้ควบคุมระบบ ซึ่งโดยปกติการค้นหาเส้นทางนั้นจะส่งต่อแพ็คเก็ตไปยังหมายเลขที่อยู่ปลายทาง (Destination Address) อย่างไรก็ตามในบางกรณีอาจจำเป็นต้องส่งต่อแพ็คเก็ตไปในพื้นฐานปัจจัยด้านอื่นก็ได้ เช่น ผู้ดูแลระบบเครือข่ายต้องการส่งแพ็กเก็ตไปยังหมายเลขที่อยู่ต้นทาง (Source Address) หรือ หมายเลขพอร์ต (Port Number) ซึ่งไม่ใช่ที่อยู่ปลายทางก็เป็นได้ ซึ่งการส่งต่อแพ็คเก็ตแบบนี้เรียกว่า การเลือกเส้นทางต้นทาง (Source-Based Routing) โดยต้นทางสามารถระบุเส้นทางที่แพ็คเก็ตต้องผ่านได้

ไฟร์วอลล์เป็นอุปกรณ์หรือกลุ่มของอุปกรณ์ที่ทำหน้าที่กลั่นกรองข้อมูลเข้าออกของระบบเครือข่ายโดยจะปฏิบัติตามนโยบายและกฎของการรักษาความปลอดภัยที่กำหนดในขณะที่มีการติดต่อสื่อสาร โดยกฎของการจราจรบนเครือข่ายจะถูกระบุตามหมายเลขไอพี (IP Address) การบริการ (Service) พอร์ต (Port) และ โพรโทคอล (Protocol) ซึ่งเปรียบเสมือนเป็นยามป้องกันให้กับระบบเครือข่ายในการคัดกรองการเข้าออกของข้อมูลระหว่างเครือข่ายทั้งเครือข่ายท้องถิ่น (Local Area Network) หรือเครือข่ายทางไกล (Wide Area Network) หากมีแพ็คเก็ตแปลกปลอมหรือไม่ได้รับอนุญาตให้เข้าออกในระบบ ไฟร์วอลล์ก็จะทำการทิ้ง (Drop) แพ็คเก็ตนั้นไป

จากความสำคัญของการค้นหาเส้นทางและไฟร์วอลล์ที่กล่าวมานั้น จึงได้เห็นปัญหาของการค้นหาเส้นทางตามนโยบาย (Policy-Based Routing) ที่มีการส่งแพ็คเก็ตผ่านกฎไฟร์วอลล์ ซึ่งปัญหาของการค้นหาเส้นทางโดยใช้นโยบายของผู้ดูแลระบบฝั่งต้นทางอนุญาตให้มีการส่งผ่านแพ็คเก็ตไปยังปลายทางได้ หากระหว่างทางของการส่งผ่านแพ็คเก็ตนั้นมีการบล็อก (Block) หรือทิ้ง (Drop) แพ็คเก็ตนั้นๆ ไม่ว่าจะด้วยเงื่อนไขของ หมายเลขไอพี (IP Address) การบริการ (Service) พอร์ต (Port) และ โพรโทคอล (Protocol) ก็ตาม ดังนั้นการบล็อกของกฎไฟร์วอลล์ระหว่างทางนั้นทำให้การค้นหาเส้นทางไม่มีประสิทธิภาพ และยังทำให้สิ้นเปลืองแบนด์วิดท์เนื่องจากการทิ้ง (Drop) ของแพ็คเก็ต ซึ่งปัญหาที่กล่าวมาแล้ว จึงทำให้มีแนวคิดในการค้นหาเส้นทางที่เปลี่ยนแปลงไปตามเส้นทางที่ ไม่มีการบล็อกด้วยกฎของไฟร์วอลล์ และทิ้งแพ็คเก็ตทันทีที่ต้นทางหากกฎของไฟร์วอลล์ระหว่างทางไม่ให้ส่งผ่านไปได้

ปัจจุบันมีงานวิจัยเกี่ยวกับการค้นหาเส้นทางที่ดีที่สุดอยู่มาก แต่แนวคิดในการค้นหาเส้นทางเชิงนโยบาย โดยนำกฎของไฟร์วอลล์เข้ามาเป็นหนึ่งในตัวเลือกในการค้นหาเส้นทางที่ดีที่สุดนั้นยังไม่มีงานวิจัยในลักษณะการแลกเปลี่ยนกฎของไฟร์วอลล์นี้มากนัก ดังนั้นการวิจัยฉบับนี้จึงมุ่งเน้นแนวคิดและการพัฒนาการค้นหาเส้นทางใหม่ที่จะเข้ามาแก้ไขปัญหาดังกล่าวมาแล้วข้างต้น 1.) เพื่อทำให้การค้นหาเส้นทางบนเราเตอร์สามารถลดค่าใช้จ่ายในการใช้แบนด์วิดท์ได้ ในกรณีที่เราเตอร์กลางทางหรือปลายทางมีการบล็อกแพ็คเก็ตตามกฎของไฟร์วอลล์ เราเตอร์ที่ได้รับการแลกเปลี่ยนกฎของไฟร์วอลล์แล้วจะไม่ส่งแพ็คเก็ตที่ถูกบล็อกไปยังเราเตอร์ปลายทาง 2.) นอกจากนี้การค้นหาเส้นทางในรูปแบบเชิงนโยบายได้ดียิ่งขึ้น โดยเราเตอร์จะส่งแพ็คเก็ตไปยังเส้นทางที่อนุญาตโดยกฎของไฟร์วอลล์ จึงทำให้การค้นหาเส้นทางไปยังปลายทางเป็นไปอย่างถูกต้อง

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1. เพื่อศึกษาลักษณะการทำงานของการค้นหาเส้นทางเชิงนโยบายที่มีไฟร์วอลล์เป็นอุปสรรคในการค้นหาเส้นทาง

2. เพื่อศึกษากฎและรูปแบบของไฟร์วอลล์ที่มีผลต่อการส่งต่อแพ็คเก็ตในระบบเครือข่าย

3. เพื่อนำเสนอวิธีการค้นหาเส้นทางโดยนำกฎของไฟร์วอลล์มาเป็นหนึ่งในตัวเลือกของการค้นหาเส้นทางที่ถูกต้อง สามารถส่งผ่านแพ็คเก็ตไปยังเส้นทางที่กฎของไฟร์วอลล์อนุญาตให้ผ่านและทิ้ง (Drop) แพ็คเก็ตทันทีที่ต้นทางหากกฎของไฟร์วอลล์ระหว่างทางไม่ให้ส่งผ่านไปได้

1.3 สมมติฐานของการศึกษา

โดยปกติการค้นหาเส้นทางของระบบเครือข่ายจะมี 2 ลักษณะ คือ

1.3.1 การค้นหาเส้นทางคงที่ (Static Route) การเลือกเส้นทางแบบคงที่ ผู้ดูแลระบบจะทำการสร้าง Routing Entry เข้าไปไว้ใน Routing Table เอง โดยมีรูปแบบดังนี้ เช่น ตัวอย่างการทำ Static Route บน Windows (1)

ip route <destination IP address> <Subnet Mask > < next hop>

ip route 192.168.20.0 255.255.255.0 192.168.30.1 (1)

1.3.2 การค้นหาเส้นทางที่เปลี่ยนแปลง (Dynamic Route) การเลือกเส้นทาง (Path) ที่ดีที่สุดของเราเตอร์นั้นมีหลายเงื่อนไขซึ่งอาจจะเลือก 1.) เงื่อนไขเส้นทางที่สั้นที่สุด 2.) เส้นทางที่ส่งต่อแพ็คเก็ตได้เร็วที่สุด หรือ 3.) เส้นทางตามนโยบายของผู้แลระบบ โดยปัจจัยการเลือกเส้นทางและกฎของไฟร์วอลล์นั้นอยู่ที่ผู้ดูแลระบบ

ดังนั้นข้อสมมติฐานว่าหากมีการตั้งกฎของไฟร์วอลล์ในระหว่างทางซึ่งทำการทิ้งแพ็คเก็ตในเงื่อนไขที่ตรงกัน แต่นโยบายการค้นหาเส้นทางจำเป็นต้องวิ่งผ่านเส้นทางนั้น จึงจำเป็นต้องมีการค้นหาเส้นทางโดยนำกฎของไฟร์วอลล์มาเป็นหนึ่งในตัวเลือกในการค้นหาเส้นทาง

การแก้ไขสมมติฐานข้างต้นคือ 1.) การแลกเปลี่ยนกฎของไฟร์วอลล์ระหว่างเราเตอร์หรือกลุ่มของเราเตอร์ 2.) คำนวณการค้นหาเส้นทางใหม่ซึ่งสามารถส่งแพ็คเก็ตจากต้นทางไปยังปลายทางได้สมบูรณ์ โดยจะทำการดึงกฎของไฟร์วอลล์และทำการส่งผ่านไปยังเราเตอร์หลัก 3.) กระจายคำสั่งในการค้นหาเส้นใหม่กลุ่มของเราเตอร์ทุกตัว ที่มีการเปลี่ยนแปลงกฎของไฟร์วอลล์ตลอดเวลา โดยการค้นหาเส้นทางก็จะเปลี่ยนแปลงไปตามการรวมกฎของไฟร์วอลล์

1.4 ขอบเขตงานวิจัย

งานวิทยานิพนธ์ฉบับนี้ได้ศึกษาและพัฒนาการค้นหาเส้นทางโดยสามารถใช้กฎของไฟร์วอลล์มาเป็นตัวเลือกในการค้นหาเส้นทาง สามารถรวบรวมกฎของไฟร์วอลล์จากเราเตอร์แต่ละตัวเพื่อทำการเรียงลำดับความสำคัญก่อนหลังและสามารถค้นหาเส้นทางได้ถูกต้องตามข้อสมมติฐานข้างต้น โดยการทดลองนั้นจะทำการทดลองบน PF: Packet Filter บนระบบปฏิบัติการ FreeBSD

1.5 ขั้นตอนของการศึกษา

วิทยานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความเป็นมาของงานวิจัย ความมุ่งหมายและวัตถุประสงค์ สมมติฐาน ขอบเขตของการวิจัย และขั้นตอนการศึกษา

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในการวิจัยและงานวิจัยที่เกี่ยวข้อง พื้นฐานการค้นหาเส้นทางเชิงนโยบาย พื้นฐานรูปแบบกฎของไฟร์วอลล์ และงานวิจัยที่เกี่ยวข้อง

บทที่ 3 การค้นหาเส้นทางเชิงนโยบายโดยใช้การรวมกฎของไฟร์วอลล์ การกรองและประกาศกฎของไฟร์วอลล์บนโปรแกรม PF: Packet Filter อัลกอริทึมการรวมกฎของไฟร์วอลล์ อัลกอริทึมการเรียงลำดับความสำคัญของเส้นทางที่ต้องการค้นหาเส้นทางโดยใช้ค่า Weight ของ Router_ID และ Prefix Matching ของ Subnet Mask และการค้นหาเส้นทางที่ถูกต้องหลังจากกระบวนการของการรวมกฎไฟร์วอลล์

บทที่ 4 ผลการทดลองและผลการวิเคราะห์ความถูกต้องการค้นหาเส้นทางเชิงนโยบายโดยใช้การรวมกฎของไฟร์วอลล์ โปรแกรมที่ใช้ในการทดสอบ ค่าชี้วัดผลการทดลอง พารามิเตอร์ที่ใช้ในการจำลองระบบ ข้อจำกัด ผลการทดลองของการรมกฎไฟร์วอลล์และผลความถูกต้องของการรวมกฎไฟร์วอลล์ และ วิเคราะห์ความถูกต้องของการเรียงลำดับการค้นหาเส้นทางเชิงนโยบาย

บทที่ 5 ผลสรุปการวิจัยและข้อเสนอแนะ

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

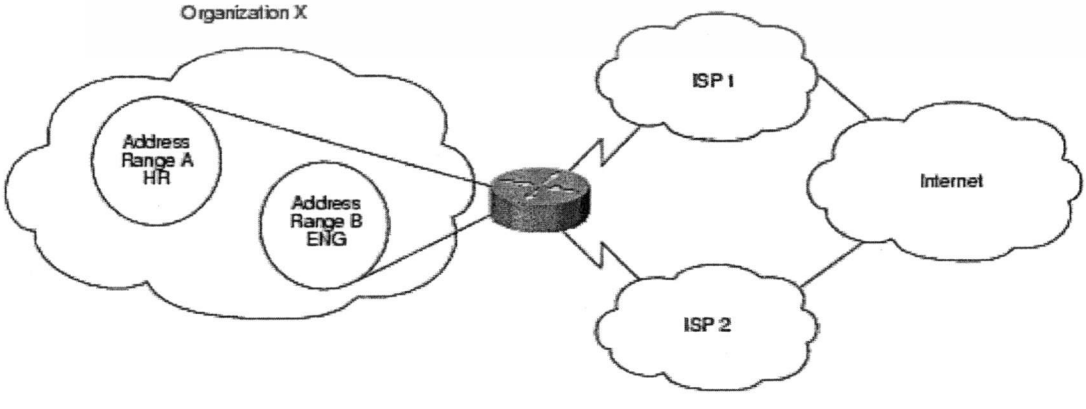
ในหัวข้อนี้จะกล่าวถึงทฤษฎีที่เกี่ยวข้องในการวิจัย ซึ่งประกอบไปด้วยการค้นหาเส้นทางเชิงนโยบาย กฎของไฟร์วอลล์ ปัญหาของการค้นหาเส้นทางที่มีกฎของไฟร์วอลล์เป็นอุปสรรคในการส่งแพ็คเก็ต และงานวิจัยที่เกี่ยวข้อง

2.1 การค้นหาเส้นทางเชิงนโยบาย (Policy-Based Routing)

ในปัจจุบันระบบเครือข่ายที่มีประสิทธิภาพสูงในองค์กรต้องการเลือกการค้นหาเส้นทางและส่งต่อแพ็คเก็ตไปยังจุดหมายปลายทางได้อย่างอิสระซึ่งถูกกำหนดไว้ด้วยนโยบายของผู้ดูแลระบบ โดยการค้นหาเส้นทางเชิงนโยบาย (Policy-Based Routing) นั้นได้เข้ามามีส่วนในการระบุเงื่อนไขในการค้นหาเส้นทางมากกว่าการค้นหาเส้นทางโดยใช้ที่อยู่ปลายทาง (Destination-Based Routing) พื้นฐานทั่วไป

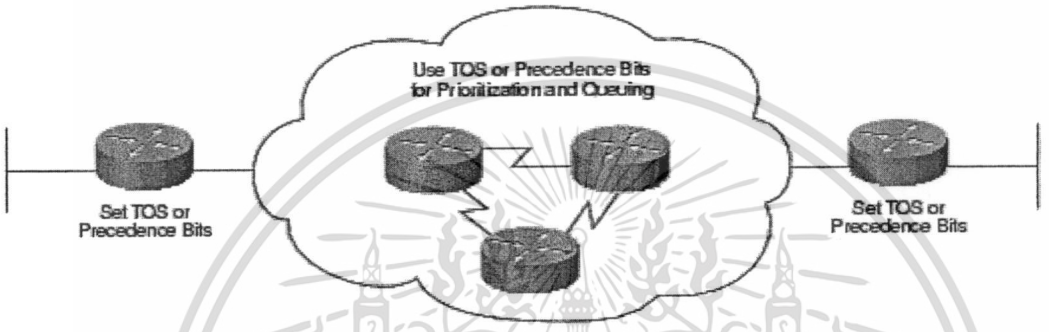
การค้นหาเส้นทางเชิงนโยบายนั้นมีกลไกในการเลือกเส้นทางที่มาจาก 1.) การค้นหาเส้นทางเชิงนโยบาย (Policy-Based Routing) 2.) รูปแบบของแพ็คเก็ต (Packet Format) หรือ 3.) เงื่อนไขอื่นๆ ซึ่งประโยชน์ของการเลือกเส้นทางเชิงนโยบายนั้นประกอบไปด้วย

2.1.1 Source-Based Transit Provider Selection ผู้ให้บริการอินเทอร์เน็ต และองค์กรต่างๆ สามารถใช้การค้นหาเส้นทางเชิงนโยบาย เพื่อส่งแพ็คเก็ตต้นทางจากกลุ่มของผู้ใช้ที่แตกต่างกันได้ โดยผ่านการเชื่อมต่อเครือข่ายอินเทอร์เน็ตที่แตกต่างกันข้ามกลุ่มของเราเตอร์ จากรูปที่ 2.1 องค์กรสามารถเลือกเส้นทางจากต้นทางได้โดยกลุ่มที่อยู่ A ไปยังเส้นทางผู้ให้บริการที่ 1 (ISP 1) และกลุ่มที่อยู่ B ไปยังเส้นทางผู้ให้บริการที่ 2 (ISP 2)



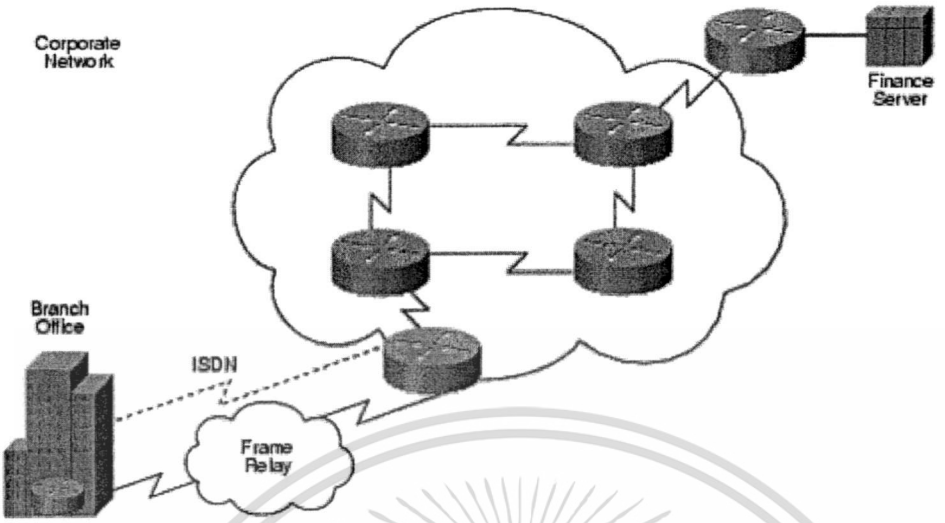
รูปที่ 2.1 การค้นหาเส้นทางโดยใช้ที่อยู่ต้นทาง

2.1.2 Quality of Services (QoS) จำแนกการจราจร (Traffic) โดยจัดการลำดับความสำคัญที่มากกว่า หรือ จัดการด้วยค่าของส่วนหัวของไอพีแพ็คเก็ต (IP Packet Header) ในแต่ละชนิดของการให้บริการ (Type of Service) ซึ่งใช้กลไกการเข้าคิว โดยดูเงื่อนไขจากการกำหนดลำดับความสำคัญก่อนและหลัง ในส่วนแกนกลางของเครือข่าย (Backbone of Network) จากรูปที่ 2.2 ผู้ดูแลระบบสามารถจำแนกชนิดของการให้บริการ และเรียกลำดับความสำคัญก่อนหลัง โดยอาจจะดูจากค่าน้ำหนัก (Weight) หรือชนิดของการให้บริการก็ได้



รูปที่ 2.2 ชนิดของการจัดลำดับสิทธิ์ก่อนหลังในการค้นหาเส้นทาง

2.1.3. Cost Saving ลดต้นทุนการใช้การเชื่อมต่อของอินเทอร์เน็ต โดยเลือกเส้นทางที่มีความรวดเร็วกว่า ราคาการเชื่อมต่อที่ถูกกว่า จากรูปที่ 2.3 ในองค์กรที่มีปริมาณการใช้แบนด์วิดท์มากขึ้นซึ่งบางงานจำเป็นต้องใช้แบนด์วิดท์ที่สูงกว่า และใช้เส้นทางที่มีราคาแพงในระยะสั้น โดยการเชื่อมต่อสำรองจะทำการสร้างการเชื่อมต่อในทันทีหากต้องการใช้การจราจรที่มีความสำคัญกว่า โดยเมื่อสิ้นสุดการใช้งานแล้วก็จะทำการปิดการเชื่อมต่อ ให้ใช้งานไปยังการเชื่อมต่อที่มีค่าต่ำกว่า

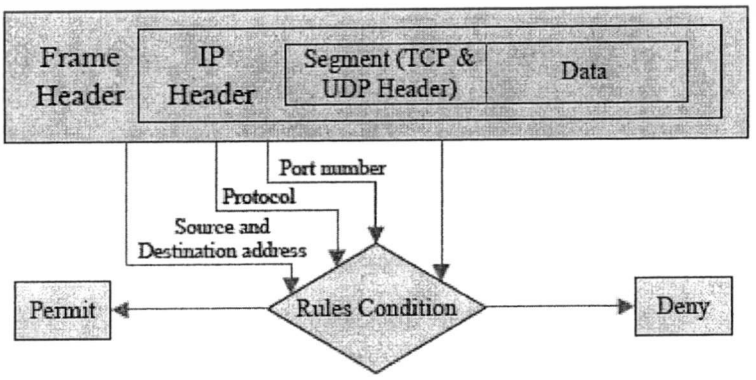


รูปที่ 2.3 การเชื่อมต่อจากการร้องขอเมื่อมีการจราจรเฉพาะกิจเกิดขึ้น

2.1.4. Load Sharing กระจายและแบ่งภาระการส่งแพ็คเก็ตไปแต่ละเส้นทาง โดยแยกแต่ละชนิดของการให้บริการที่อยู่ไอพีหรือพอร์ตตามนโยบายที่วางไว้

2.2 พื้นฐานรูปแบบของกฎไฟร์วอลล์บน PF: Packet Filter

รูปแบบการตรวจสอบกฎของไฟร์วอลล์ซึ่งแสดงดังรูปที่ 2.4 โดยส่วนหัวของเฟรม จะมี IP Header รวมถึง Segment และข้อมูล โดยจะเก็บข้อมูลหมายเลขต้นทาง (Source Address) หมายเลขปลายทาง (Destination Address) โพรโทคอล (Protocol) และหมายเลขพอร์ต (Port Number) หลังจากนั้นก็จะเข้าไปตรวจสอบเงื่อนไขว่าสามารถอนุญาตหรือไม่อนุญาตให้แพ็คเก็ตส่งผ่านได้หรือไม่



รูปที่ 2.4 การตรวจสอบกฎของไฟร์วอลล์

ตัวอย่างรูปแบบของ Firewall Rules บน PF: Packet Filter ในแต่ละคำสั่งอาจจะเป็นค่าเพียงค่าเดียว หรือเป็นค่าที่เป็นช่วงก็ได้ ดังกฎ (2)

Block out on $ext_if from 161.246.52.0/24 to 202.28.68.5 port {1024><2048}
Pass in on $ext_if from 161.246.0.0/16 to any port {80, 443}
Pass in on $int_if route-to ($exit_if 161.246.52.254) inet proto tcp from 161.246.52.0/24 (2)

กฎของไฟร์วอลล์บน PF: Packet Filter นั้นจะประมวลผลแบบเรียงลำดับ คือแต่ละแพ็คเก็ตที่วิ่งเข้ามาได้ถูกทดสอบกับกฎแรกของบรรทัด ถ้ากฎนั้นจับคู่กันแล้วผ่านก็จะส่งต่อไปยังคำสั่งบรรทัดถัดไป โดยบรรทัดสุดท้ายเป็นกฎ Implicit Deny All หากไม่มีแพ็คเก็ตจับคู่คำสั่งใดๆ เลย ซึ่งแพ็คเก็ตใดที่ไม่ได้รับอนุญาตโดยชัดเจนจะถูกปฏิเสธหากไม่มีการตั้งกฎ Allow ใดๆ โดยค่าดีฟอลท์จะเป็น Deny All หากแพ็คเก็ตใดที่ไม่ได้รับการปฏิเสธโดยชัดเจนจะถือว่าได้รับการอนุญาต Allow All ดังรูปที่ 2.5 ดังนั้นการใช้คำสั่งของของกฎไฟร์วอลล์นั้นจะเป็นตามบรรทัดทีละบรรทัด และ Subnet Mask ที่มากที่สุดต้องมาก่อน Subnet Mask หากอยู่ใน Class เดียวกันที่น้อยกว่า ซึ่งจะเข้ากับกฎ Prefix Matching ยกตัวอย่างเช่น

กฎที่เรียงลำดับได้ถูกต้อง
Block out on $ext_if from 161.246.52.0/24 to any port {1024><2048}
Pass in on $ext_if from 161.246.0.0/16 to any port {80, 443} (3)

กฎที่เรียงลำดับผิด
Pass in on $ext_if from 161.246.0.0/16 to any port {80, 443}
Block out on $ext_if from 161.246.52.0/24 to any port {1024><2048} (4)

จากตัวอย่าง (4) แสดงให้เห็นว่า หากให้หมายเลข IP 161.246.0.0/16 มาก่อน 161.246.52.0/24 นั้น หากมีแพ็คเก็ตหมายเลข 161.246.52.1 เข้ามาก็จะเข้าเงื่อนไข Pass ทันที ตั้งแต่บรรทัดแรก จึงทำให้กฎของไฟร์วอลล์ที่เรียงกันนั้นผิด ดังนั้น (3) จึงถูกต้อง เนื่องจาก จะตรวจสอบหมายเลข IP 161.246.52.0/24 ก่อน หากไม่ตรงกับเงื่อนแรกก็จะทำเงื่อนไขบรรทัดถัดไป

ดังนั้นการใช้คำสั่งกฎของไฟร์วอลล์ ผู้ดูแลระบบควรจะมีความรู้ทฤษฎีของระบบเครือข่ายเป็นอย่างดีและเข้าใจหลักการของ IP Addressing ซึ่งเป็นทฤษฎีที่สำคัญของการตั้งกฎไฟร์วอลล์ อีกทั้ง Routing เช่นกัน

รูปที่ 2.5 ACL Implicit Deny All

จากรูปที่ 2.5 แสดงถึงการตรวจสอบแพ็คเก็ตที่วิ่งเข้ามาว่าตรงกับกฎของไฟร์วอลล์หรือไม่ หากใช่ก็จะเข้า ตรวจสอบเงื่อนไขเรียงไปทีละบรรทัดจนถึงบรรทัดสุดท้าย เพื่อที่จะส่งต่อแพ็คเก็ตหรือลบแพ็คเก็ตนั้นทิ้งไป

2.3 งานวิจัยที่เกี่ยวข้อง

ปัจจุบันมีงานวิจัยเกี่ยวกับการค้นหาเส้นทางที่ดีที่สุดอยู่มากมาย แต่แนวคิดในการค้นหาเส้นทางเชิงนโยบาย โดยนำกฎของไฟร์วอลล์เข้ามาเป็นหนึ่งในตัวเลือกในการค้นหาเส้นทางที่ดีสุดนั้นยังไม่มีงานวิจัยในลักษณะการแลกเปลี่ยนกฎของไฟร์วอลล์มากนัก ซึ่งส่วนใหญ่ของงานวิจัยที่ใกล้เคียงที่สุดจะเป็นการประเมินประสิทธิภาพและพัฒนาการค้นหาเส้นทางเชิงนโยบาย โดยมีกฎของไฟร์วอลล์ที่เป็นผลกระทบต่อการค้นหาเส้นทาง งานวิจัยที่เกี่ยวข้องและใกล้เคียงกับงานวิทยานิพนธ์ฉบับนี้ได้แก่

2.3.1 งานวิจัยของ Vic Grout and John McGinn [4] เอกสารตีพิมพ์ชื่อ "Optimisation of Policy-Based Internet Routing Using Access Control Lists" งานวิจัยฉบับนี้เป็นการประเมินประสิทธิภาพของเวลาที่แปรผันของการค้นหาเส้นทางซึ่งมีกฎของไฟร์วอลล์เป็นตัวกระทบต่อการค้นหาเส้นทาง บทสรุปของงานวิจัยนี้คือ เมื่อมีกฎของไฟร์วอลล์จำนวนมากจะเป็นอุปสรรคต่อการค้นหาเส้นทาง เนื่องจากกฎของไฟร์วอลล์ปิดกั้นเส้นทาง จึงทำให้การค้นหาเส้นทางใช้เวลาเพิ่มขึ้นและสูญเสียแพ็คเก็ตไปอย่างมาก โดยประเมินจาก Latency ของการประมวลผลบน CPU

2.3.2 งานวิจัยของ S. Pozo, R. Ceballos, R.M. Gasca [8] เอกสารตีพิมพ์ชื่อ "A Heuristic Polynormal Algorithm for Local Inconsistency Diagnosis in Firewall Rule Sets" งานวิจัยฉบับนี้ได้วิเคราะห์ถึงประสิทธิภาพของการวางรูปของกฎไฟร์วอลล์และปัญหาของความสอดคล้องในการเรียงลำดับ ความสลับซับซ้อนของกฎของไฟร์วอลล์ ที่มีปัจจัยในด้านการตรวจจับความเร็วของกฎไฟร์วอลล์ โดยปัจจัยสำคัญของเอกสารนี้ต้องการบอกถึงการเรียงลำดับกฎไฟร์วอลล์ที่ถูกต้อง

2.3.3 งานวิจัยของ Priyadarsi Nanda and Andrew James Simmonds [7] เอกสารตีพิมพ์ชื่อ "Policy Based QoS Support Using BGP Routing" งานวิจัยฉบับนี้กล่าวถึงการค้นหาเส้นทางแบบเชิงนโยบาย Policy based โดยจัดลำดับความสำคัญของแพ็คเก็ต ซึ่งใช้ BGP โพรโทคอลเข้ามาทำผลการทดลองนั้น จัดการคุณภาพการค้นหาเส้นทางที่ดีตามนโยบาย ระหว่าง AS โดยใช้ Policy-Based Routing นั้นเป็นตัวสินสินใจเพิ่มเติมจาก Application ที่ใช้

สรุปงานวิจัยที่เกี่ยวข้องตามที่ได้กล่าวมานี้ ยังไม่มีการค้นหาเส้นทางโดยนำกฎของไฟร์วอลล์มาเป็นหนึ่งในตัวเลือกในการค้นหาเส้นทาง ในขณะที่งานวิจัยฉบับนี้จะสามารถตอบโจทย์ของสมมติฐานจากบทที่ 1 ได้ โดยสามารถดูรายละเอียดในบทถัดไป

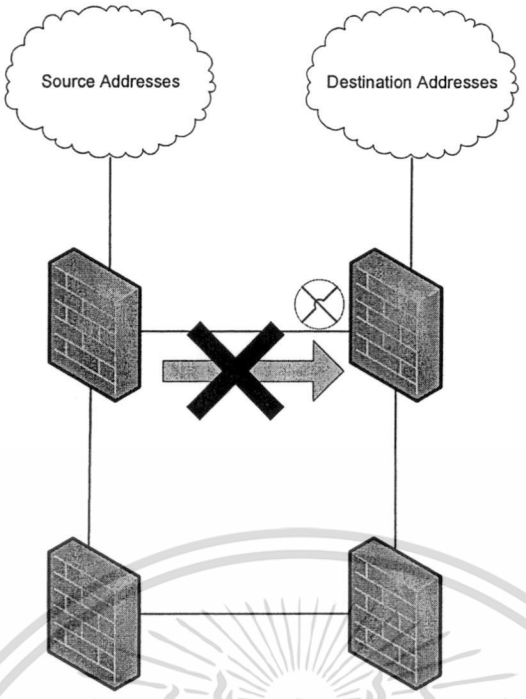
บทที่ 3
การค้นหาเส้นทางเชิงนโยบายโดยใช้การรวมกฎของไฟร์วอลล์
Policy-Based Routing Using Firewall Rules Merging

จากทฤษฎีและปัญหาที่กล่าวมาจากบทก่อนหน้านี้ การค้นหาเส้นทางเชิงนโยบายนั้นไม่สามารถเรียนรู้และแลกเปลี่ยนกฎของไฟร์วอลล์ได้ จึงได้เกิดแนวคิดในแลกเปลี่ยนและรวมกฎของไฟร์วอลล์ โดยนำกฎของไฟร์วอลล์ให้เป็นหนึ่งในตัวเลือกค้นหาเส้นทางที่ทำให้การค้นหาเส้นทางมีประสิทธิภาพมากที่สุด

โดยแนวคิดนี้สามารถใช้ได้กับไฟร์วอลล์ทุกชนิดไม่ว่าจะเป็น iptable, ipfw หรือ ACL บนอุปกรณ์เราเตอร์ ซึ่งทฤษฎีนั้นเหมือนกันแต่ต่างกันที่การพิมพ์คำสั่งในการปรับแต่งค่ากฎของไฟร์วอลล์ ในวิทยานิพนธ์ฉบับนี้เลือกใช้ Packet Filter ซึ่งเป็นไฟร์วอลล์ที่อยู่ในระบบปฏิบัติการ BSD และนอกเหนือจากนั้นไฟร์วอลล์ Packet Filter นั้นสามารถทำ Policy-Based Routing ได้อีกด้วย

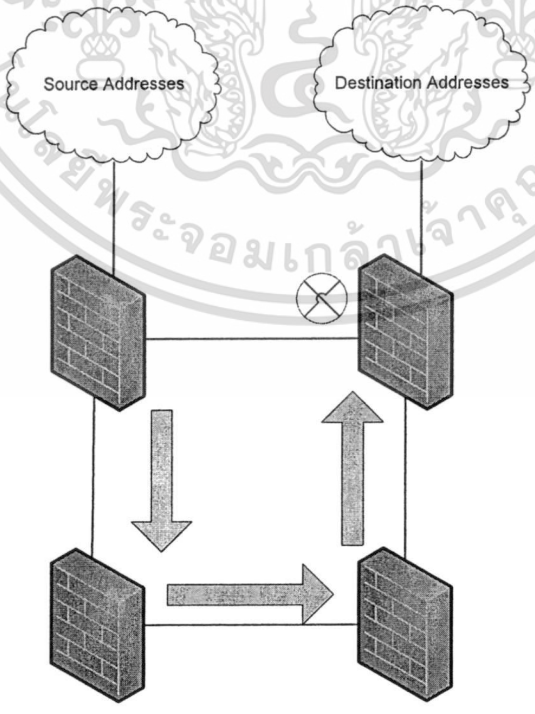
3.1 แนวทางการค้นหาเส้นทางเชิงนโยบายโดยใช้การรวมกฎของไฟร์วอลล์

จากปัญหาที่ได้กล่าวมาจากสมมติฐานบทที่ 1 ที่กล่าวไว้ว่า หากมีการตั้งกฎของไฟร์วอลล์ในระหว่างทางซึ่งทำการทิ้งแพ็คเก็ตในเงื่อนไขที่ตรงกัน แต่นโยบายการค้นหาเส้นทางจำเป็นต้องวิ่งผ่านเส้นทางนั้น แพ็คเก็ตก็จะถูกทิ้ง (Drop) ไปดังรูปที่ 3.1



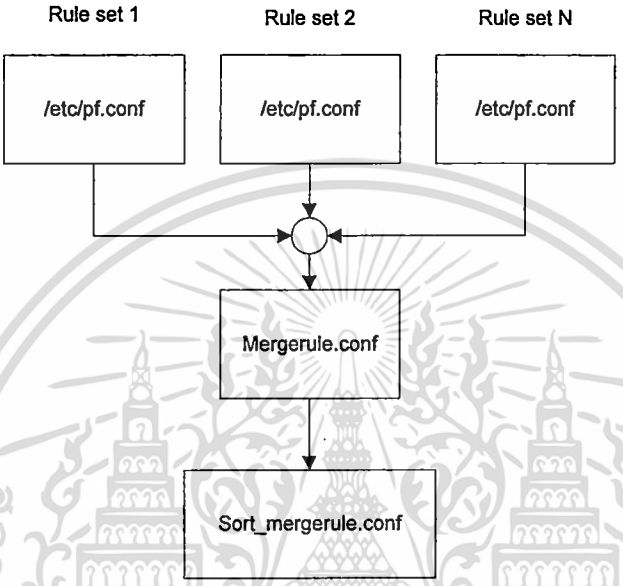
รูปที่ 3.1 รูปแบบของการส่งแพ็คเก็ตซึ่งถูกไฟร์วอลล์ทิ้ง (Drop) กลางทาง

ดังนั้นการค้นหาเส้นทางเดิมของเราเตอร์จำเป็นต้องเปลี่ยนเส้นทางเพื่อให้แพ็คเก็ตสามารถค้นหาเส้นทางได้ใหม่ตาม ทฤษฎีที่ได้เสนอไปในสมมติฐานจากบทที่ 1 โดยแสดงรูปภาพจำลองการส่งต่อแพ็คเก็ตได้ดังรูปที่ 3.2



รูปที่ 3.2 รูปแบบของการส่งแพ็คเก็ตซึ่งส่งต่อไปยังเส้นทางที่ไฟร์วอลล์อนุญาตให้ผ่าน

จากรูปที่ 3.1 และ 3.2 ซึ่งได้แสดงแบบจำลองแนวทางการค้นหาเส้นทางเชิงนโยบายโดยใช้การรวมกฎของไฟร์วอลล์นั้น ได้แสดงให้เห็นว่าสามารถทำให้การค้นหาเส้นทางไปยังเส้นทางที่ไฟร์วอลล์อนุญาตให้ผ่านได้ ในบทที่ 3 นี้จะแสดงวิธี กระบวนการ และอัลกอริทึมในโดยมีกระบวนการภาพรวมดังนี้



รูปที่ 3.3 แนวทางการรวมกฎของไฟร์วอลล์

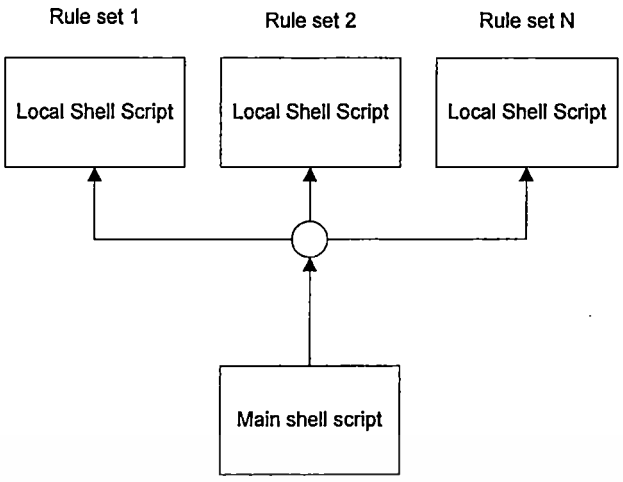
จากรูปที่ 3.3 สามารถอธิบายแนวทางการรวมกฎของไฟร์วอลล์ได้ดังนี้

3.1.1 ไฟล์ Rule Set คือ กฎของไฟร์วอลล์ที่ได้มาจากเราเตอร์แต่ละตัวจนถึงเราเตอร์ตัวที่ N

3.1.2 ทำการส่งผ่าน Rule Set ด้วย Shell Script ซึ่ง Rule Set แต่ละเครื่องจะอยู่ที่ /etc/pf.conf

3.1.3 หลังจากนั้น Rule Set จำนวน N Set จะรูปรวมกันในไฟล์ที่ชื่อว่า Mergerule.conf

3.1.4 เมื่อทำการรวมกฎของไฟร์วอลล์แล้ว กฎทุกๆ กฎที่ได้มาจาก Rule Set ทุกตัวจะถูกทำการเรียงลำดับตามค่าน้ำหนัก (Weight) ซึ่งในวิทยานิพนธ์นี้จะเรียงตาม Router_ID และเรียงลำดับตาม Prefix Matching ของ Subnet Mask



รูปที่ 3.4 การประกาศกฎของไฟร์วอลล์ และ Route-to เส้นทางใหม่ผ่านทาง Shell Script

Flowchart ของการแลกเปลี่ยนกฎของไฟร์วอลล์ระหว่างเราเตอร์โดยในงานวิจัยชิ้นจะทำการแลกเปลี่ยนกฎผ่านโพรโทคอล SSH (Secure Shell) เพื่อความปลอดภัยในการแลกเปลี่ยนกฎของข้อมูล โดยใช้คำสั่งของ Shell Script ในการ Remote Policy ดังรูปที่ 3.4

1.) จากแผนผังหลักเริ่มด้วยการ Advertise ไปยังเราเตอร์ทุกตัวในระบบเครือข่าย

2.) จากนั้นโปรแกรมก็ไปตรวจสอบกฎของไฟร์วอลล์ของทุก Rule Set

3.) หากมีกฎของไฟร์วอลล์อยู่ในเราเตอร์ตัวดังกล่าวก็จะทำการดึงกฎออกมาแล้ว

4.) ส่งสถานะพร้อมกฎกลับไปยังเราเตอร์ที่ได้ทำการ Advertise ออกมาไปเรื่อยๆ

5.) หลังจากนั้นก็จะส่งตัวแปรของกฎไฟร์วอลล์ไปยัง อัลกอริทึมการรวมไฟร์วอลล์

3.1.5 หลังจากนั้นเราเตอร์จะส่งค่าการค้นหาเส้นทาง (Route-to) ที่ได้หลังจากการเรียงลำดับกฎในของที่ 3.1.4 แล้ว ผ่านทาง Main Shell Script

3.1.6 โดย Main Shell Script จะทำการเข้าไปสั่ง Local Shell Script เพื่อให้ทำการรวมกฎของไฟร์วอลล์เดิมและค่า Route-to ของเส้นทางใหม่ด้วยกันไปยัง Rule Set N เดิมของเราเตอร์ทุกตัว

3.2 การกรองและประกาศกฎของไฟร์วอลล์บน PF: Packet Filter

รูปแบบของกฎไฟร์วอลล์บน PF: Packet Filter นั้นกฎทั้งหมดจะอยู่ใน /etc/pf.conf ซึ่งมีรูปแบบของกฎดังนี้

3.2.1 Action โดยรูปแบบคำสั่งของ Action นี้จะเป็นตัวควบคุมและกำหนดการส่งต่อแพ็คเก็ตให้ผ่านเข้าออก

3.2.1.1 คำสั่ง Pass เป็นคำสั่งที่ให้แพ็คเก็ตสามารถส่งผ่านไปได้

3.2.1.2 คำสั่ง Lock เป็นคำสั่งที่ทำให้แพ็คเก็ตที่จะวิ่งผ่านถูกทิ้งไป

3.2.2 Direction เป็นการกำหนดทิศทางของแพ็คเก็ตขาเข้า (In) และขาออก (Out)

3.2.3 Interface เป็นรูปแบบของชื่อหรือกลุ่มชื่อของ Network Interface ที่แพ็คเก็ตวิ่งผ่าน เช่น em0 , ppp หรือ fxp เป็นต้น

3.2.4 Protocol เป็นรูปแบบของโพรโทคอลเลเยอร์ที่ 4 เช่น TCP, UDP, ICMP หรือสามารถดูได้จาก /etc/protocols

3.2.5 Source Address และ Destination Address เป็นส่วนของที่อยู่หมายเลข IP ต้นทางและ IP ปลายทางโดยสามารถใส่ค่าเป็นดังนี้

3.2.5.1 ค่าที่เป็น IP Address เดี่ยว เช่น 161.246.38.141

3.2.5.2 ค่าที่เป็น Subnet Mask เช่น /16, /24 หรือ /30 เป็นต้น

3.2.5.3 ค่า Any คือค่าที่หมายถึงหมายเลข IP ใดๆ สามารถใช้คำว่า All แทนได้

3.2.6 Port เป็นส่วนของพอร์ตที่อยู่บนเลเยอร์ที่ 4 ซึ่งมาสามารถกำหนดด้วยค่าหมายเลข 1 ถึง 65535 โดยมีฟิลด์ที่จะทำการแยกตัวแปรดังนี้

ตารางที่ 3.1 รูปแบบของกฎไฟร์วอลล์บน PF: Packet Filter

Action	Interface	Protocol	Source_Addr	Dest_Addr	Port
pass , block	em0 , ppp	TCP, UDP, ICMP	IP address	IP address	0-65535

ยกตัวอย่างคำสั่งของ Packet Filter ในการกรองแพ็คเก็ต ทำการบล็อกที่ขา interface1 โพรโทคอล ICMP จาก IP ต้นทาง ไปยัง IP ปลายทางใดๆ พอร์ต 23,8080 (5) และทำการ Pass ที่ขา interface1 โพรโทคอล ICMP จาก IP ต้นทาง ไปยัง IP ปลายทางพอร์ต 23,80,443 (6)

 # block in on $interface1 proto ICMP from 161.246.1.0/24 to any port {23,8080} (5)

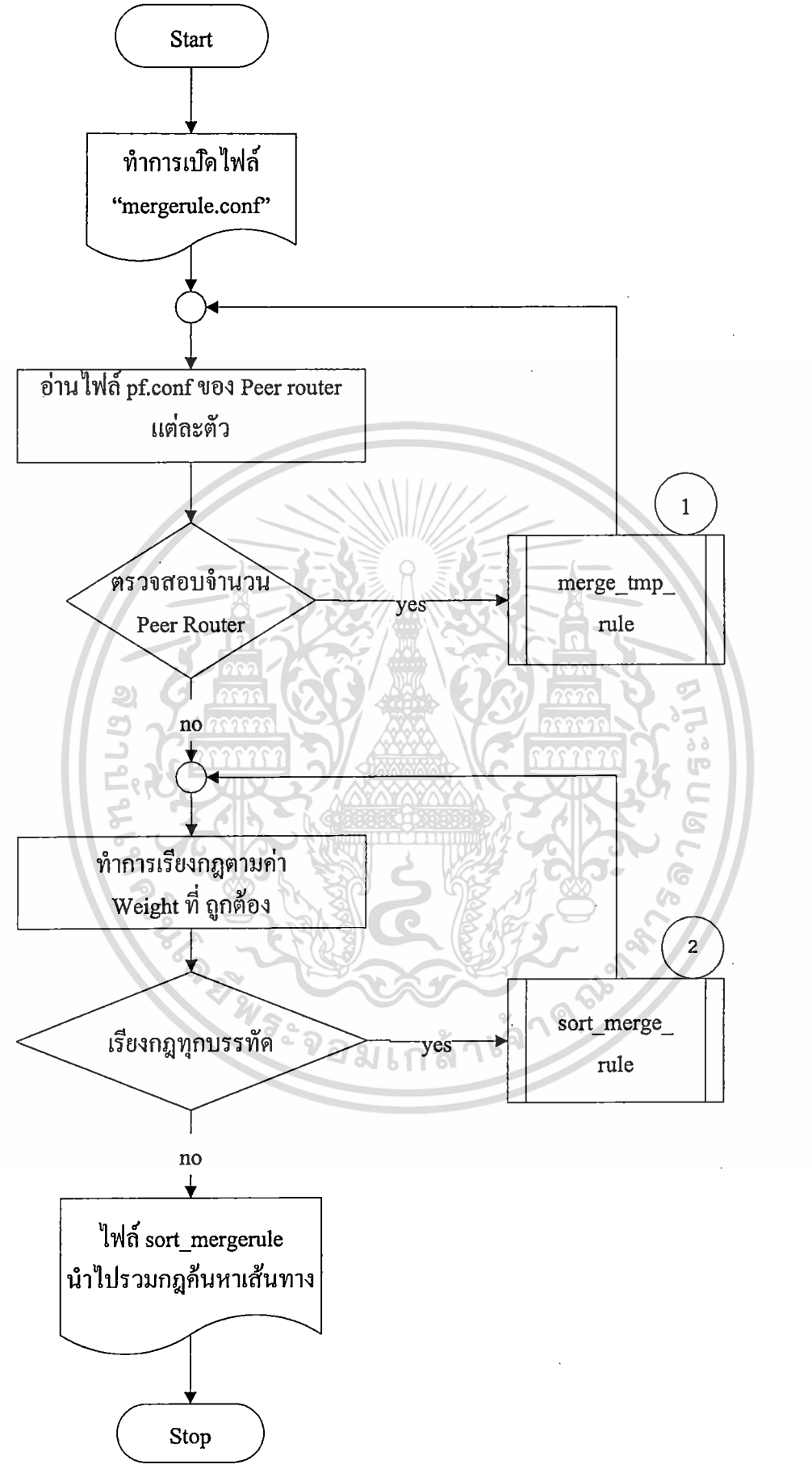
 # pass in on $interface1 proto ICMP from 161.246.0.0/16 to 202.28.68.0/24 port {23,80,443}(6)

ยกตัวอย่างคำสั่งการค้นหาเส้นทางแบบ Policy-Based Routing บน Packet Filter ทำการ Route-to ไปยัง interface1 และส่งผ่านไปยังเราเตอร์ชื่อ r3 โพรโทคอล ICMP จาก IP ต้นทางไปยัง IP ปลายทางใดๆ พอร์ต 23, 80, 443 (7) และ ทำการ Route-to ไปยัง interface 1 และส่งผ่านไปยังเราเตอร์ชื่อ r2 แต่โพรโทคอล ICMP จะถูก ทิ้ง (Drop) เนื่องจาก ! ซึ่งหมายถึง "NOT" จาก IP ต้นทางไปยัง IP ปลายทางใดๆ และทิ้ง (Drop) พอร์ต 23,80,443 เนื่องจากมีสัญลักษณ์ ! (8)

 # pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.1.1/32 to any port {23,80,443} (7)

 # pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.1.0/24 to any port !{23,8080} (8)

ดังนั้นการ Route-to จะสามารถทิ้ง (Drop) แพ็คเก็ตได้ทันทีหากมีกฎที่บล็อกส่งเข้ามา ส่วนกฎที่มี Action เป็น Pass นั้นก็จะทำการ Route-to ไปเส้นทางนั้นทันที



รูปที่ 3.5 ภาพรวมของการรวม และการเรียงลำดับของกฎไฟร์วอลล์

จากรูปที่ 3.5 จะแสดงภาพรวมของการการแลกเปลี่ยนกฎ (Exchange) การรวม (Merging) และการเรียงลำดับ (Sorting) ของไฟร์วอลล์ตามค่าน้ำหนักของการค้นหาเส้นทาง (Weight) โดย Shell Script ในการแลกเปลี่ยนกฎระหว่างเราเตอร์นั้นมีดังนี้ (9)

 # scp path source remotehost:pathDirectory

เช่น *# scp /root/.ssh/file root@10.50.35.1:/root/.ssh/file* (9)

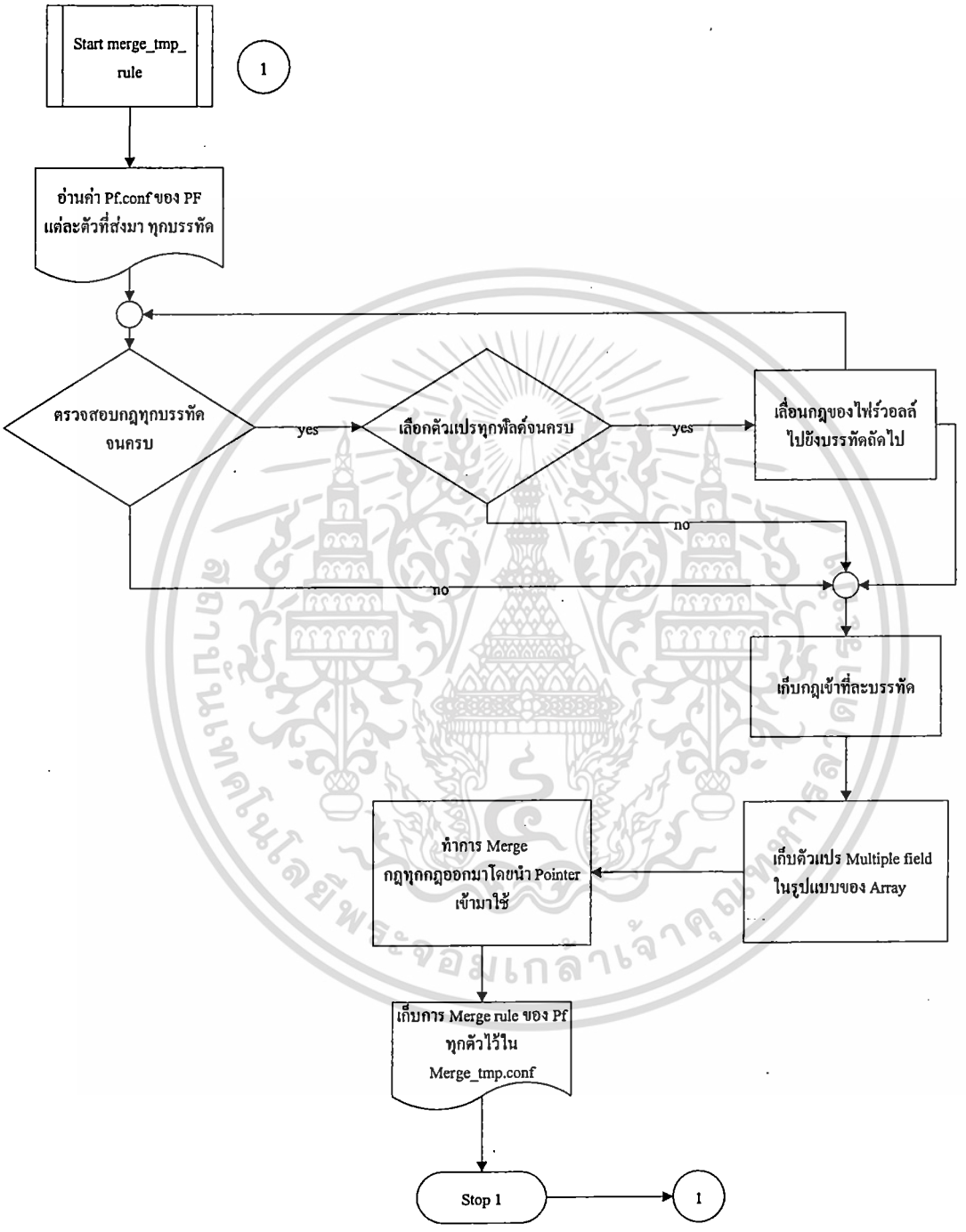
โดยมีหลักการคือ

1.) เราเตอร์ที่เป็นตัวประกาศ (Advertise) นั้นจะทำการอ่านกฎของไฟร์วอลล์ /etc/pf.conf ทุกเราเตอร์ที่อยู่ในกลุ่มเดียวกัน

2.) นำ /etc/pf.conf ซึ่งอยู่ที่ Rule Set ของเราเตอร์แต่ละตัวมาทำการเก็บไว้ใน merge_tmp.conf แล้วจึงเข้า อัลกอริทึมรวมกฎของไฟร์วอลล์และเรียงลำดับความสำคัญของการค้นหาเส้นทางโดยใช้คำสั่ง route-to บน PF:Packet Filter ซึ่งจะกล่าวในหัวข้อถัดไป

3.3 อัลกอริทึมการรวมกฎของไฟร์วอลล์

3.3.1 Flowchart ของการรวมกฎของไฟร์วอลล์



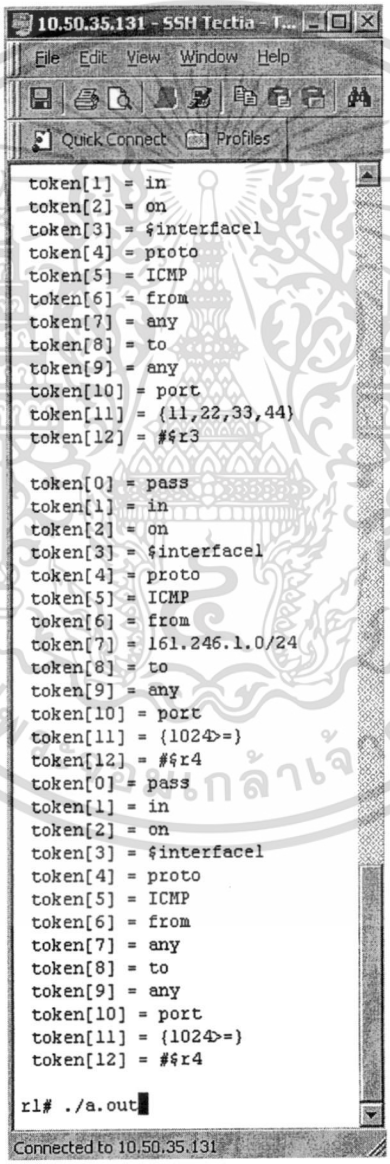
รูปที่ 3.6 Flowchart ของการรวมกฎของไฟร์วอลล์แบบ Multiple Fields

3.3.2 อธิบายกระบวนการอัลกอริทึมการรวมกฎของไฟร์วอลล์

3.3.2.1 ทำการสร้างไฟล์ชื่อ merge_tmp.conf ไว้เพื่อทำการเก็บกฎของไฟร์วอลล์ที่รับมาจากไฟร์วอลล์ที่เชื่อมต่อระหว่างกัน

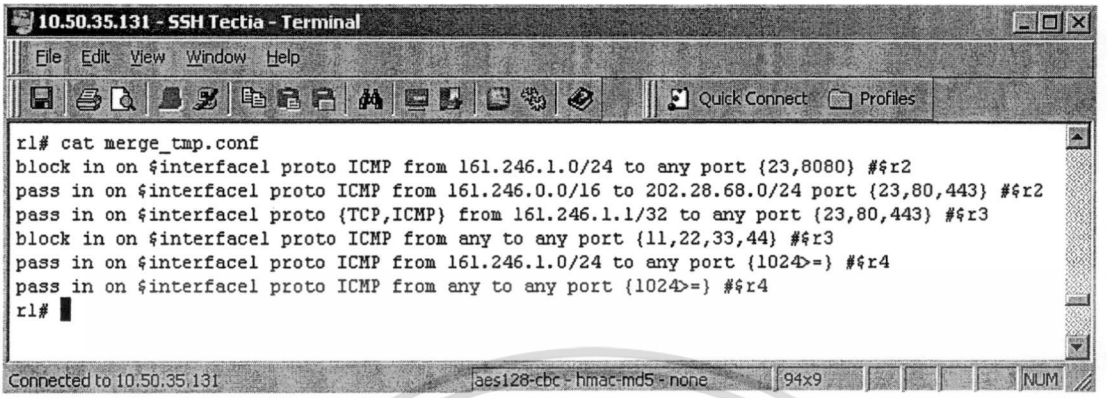
3.3.2.2 ทำการอ่านค่ากฎของไฟร์วอลล์และทำการแบ่งฟิลด์ออกเป็นส่วนๆ เพื่อนำไปเก็บให้อยู่ในรูปแบบ Multiple Field

3.3.2.3 จัดเก็บกฎที่ได้ทำการ Merge เสร็จแล้วไว้ใน merge_tmp.conf เพื่อนำไปเข้ากระบวนการ Sorting และ Weight น้ำหนักค่าของการค้นหาเส้นทาง โดยตัวอย่างระหว่างการประมวลผลดังรูปที่ 3.7 ซึ่งแสดงให้เห็นการแยกฟิลด์แต่ละฟิลด์ออกมาเพื่อประมวลผล



รูปที่ 3.7 ตัวอย่างการอ่านกฎเพื่อเก็บในรูปแบบ Struct ซึ่งเก็บใน Array

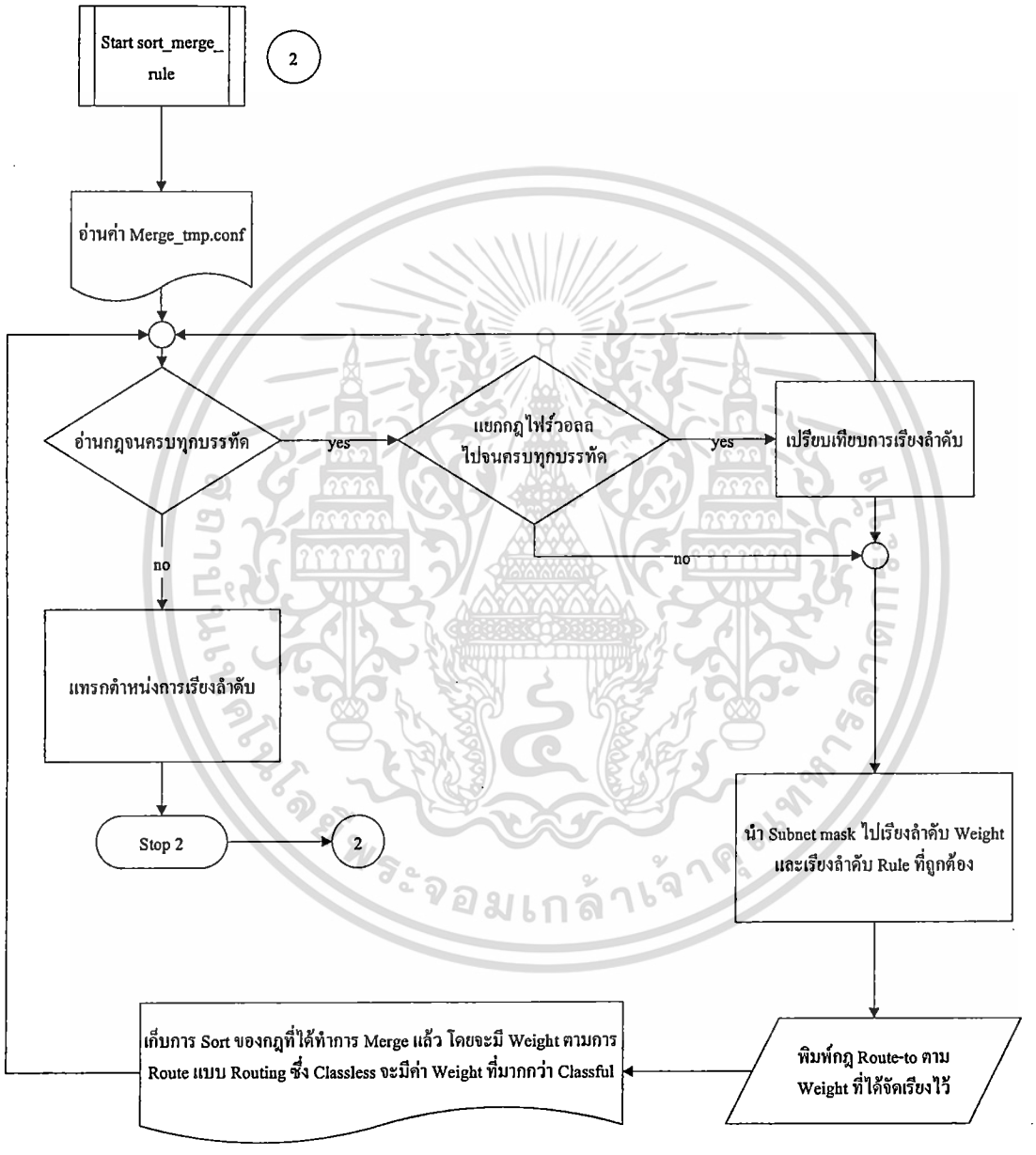
3.3.2.4 กฎที่ได้มาจะอยู่ในรูปแบบของข้อความที่เป็น Text ดังรูปที่ 3.8 โดยสามารถเห็นกฎของไฟร์วอลล์ที่ได้หลังจากการ Merge เรียบร้อยแล้ว



รูปที่ 3.8 ไฟล์ merge.tmp.conf

3.4 อัลกอริทึมการเรียงลำดับความสำคัญของเส้นทางที่ต้องการค้นหาเส้นทางโดยใช้ค่า Weight ของ Router_ID และ Prefix Matching ของ Subnet Mask

3.4.1 Flowchart การเรียงลำดับความสำคัญของเส้นทางที่ต้องการค้นหาเส้นทางโดยใช้ค่า Weight ของ Router_ID และ Prefix Matching ของ Subnet Mask



รูปที่ 3.9 Flowchart ของการเรียงลำดับความสำคัญของเส้นทางที่ต้องการค้นหาเส้นทางโดยใช้ค่า Weight ของ Router_ID และ Prefix Matching ของ Subnet Mask

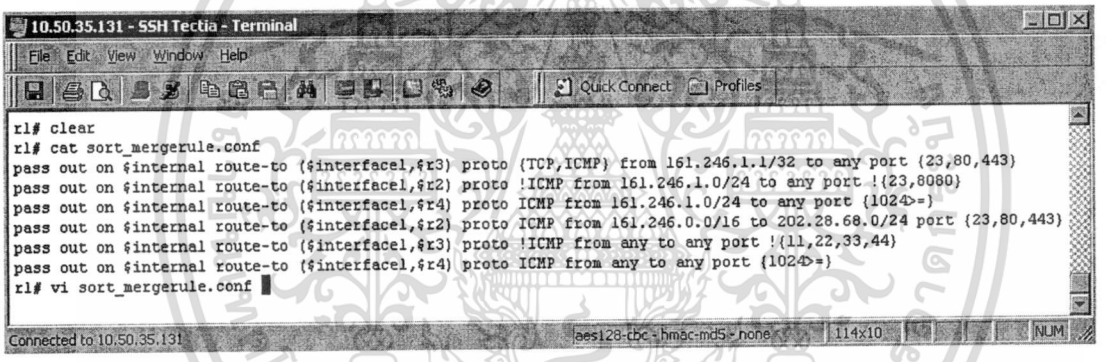
3.4.2 อธิบายกระบวนการเรียงลำดับความสำคัญของเส้นทางที่ต้องการค้นหาเส้นทางโดยใช้ค่า Weight ของ Prefix และ Subnet Mask

3.4.2.1 หลังจากกระบวนการ Merge เสร็จสิ้นแล้ว ในกระบวนการนี้จะทำการเรียงลำดับความสำคัญของการค้นหาเส้นทาง (Routing) ขั้นแรกจะทำการอ่าน merge_tmp.conf ก่อน

3.4.2.2 ทำการเรียงลำดับความสำคัญของการค้นหาเส้นทาง (Routing) โดยดูจาก Subnet Mask ซึ่ง Classless จะมี Weight ที่มากกว่า Classful คือ Subnet Mask ที่มีค่ามากที่สุดจะมีน้ำหนักมากที่สุด

3.4.2.3 การทำ Routing โดยการใช้การรวมกฎของไฟร์วอลล์นั้นจะมีเงื่อนไขหากกฎที่ได้รับมาเป็น Pass จะทำการ Route เส้นทางไปยังเส้นทางที่อนุญาตให้แพ็คเก็ตนั้นผ่านได้ หากกฎที่ได้รับมาเป็นบล็อกจะทำการทิ้ง (Drop) แพ็คเก็ตตั้งแต่ต้นทางเพื่อไม่ให้สูญเสียการส่งแพ็คเก็ตออกไป ดังรูปที่ 3.10

.

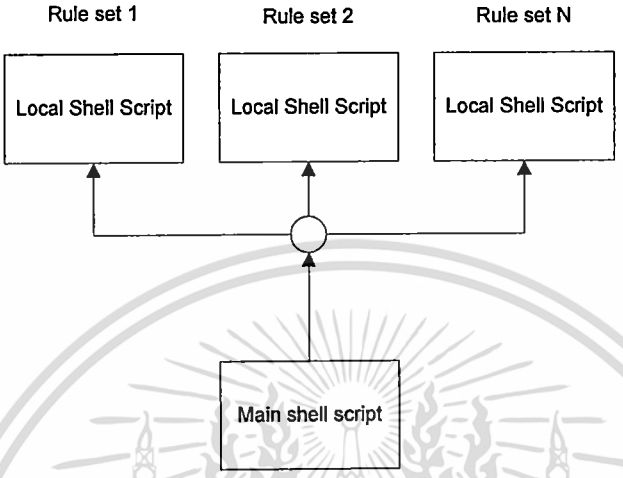


รูปที่ 3.10 ผลของการค้นหาเส้นทางแบบ Policy-Based Routing

โดยรูปแบบจะใช้คำสั่ง Route-to ในการค้นหาเส้นทางแบบ Policy-Based Routing ซึ่งจะเห็นได้ชัดเจนว่าหากกฎใดที่ถูกบล็อกก็ใส่ค่า ! เพื่อทำการทิ้ง (Drop) แพ็คเก็ตตั้งแต่ต้นทาง

3.5 การค้นหาเส้นทางหลังจากกระบวนการรวมกฎของไฟร์วอลล์

หลังจากเสร็จสิ้นการรวมกฎของไฟร์วอลล์แล้วเราเตอร์ทุกตัวที่ได้ทำการประกาศกรองกฎนั้นจะทำการส่งกฎผ่าน SSH กลับไปยังเราเตอร์ในกลุ่มใหม่แต่ละตัวในเครือข่าย ดังรูปที่ 3.11



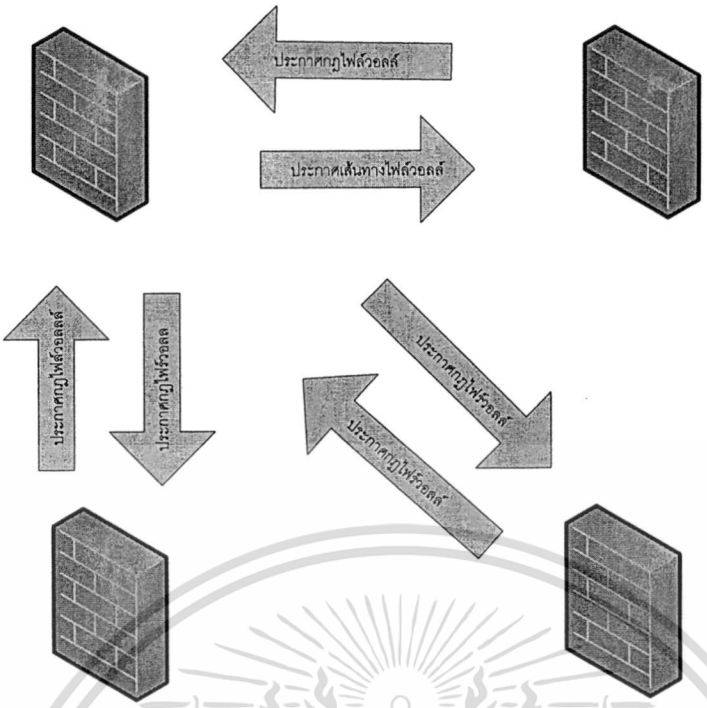
รูปที่ 3.11 รูปแบบการประกาศ Shell Script คืนกลับไปยังเราเตอร์แต่ละตัว

ในวิทยานิพนธ์ฉบับนี้ได้ทำการเปิด Authorized ให้กับเราเตอร์ทุกตัวเพื่อให้สามารถส่งผ่านโพรโทคอล SSH ได้โดยได้กำหนดสิทธิ์ในการเข้าถึงให้เป็น Root ซึ่งสามารถควบคุมเราเตอร์ในเครือข่ายได้ทั้งหมด โดยผ่าน Main Shell Script หลัก หลังจากนั้น Local Shell Script ก็จะทำงานตามสั่งของ Main Shell Script ซึ่งมีคำสั่งการส่งผ่านดังนี้ (10)

#scp path source RemoteHost:PathDirectory

เช่น *#scp /root/.ssh/file root@10.50.35.1:* (10)

ดังรูปที่ 3.11 แสดงแผนภาพการแลกเปลี่ยนกฎของไฟล์วอลล์ในเราเตอร์ทุกๆ ตัว จาก Peer ของเราเตอร์ที่ติดกันซึ่งในเราเตอร์แต่ละตัวจะมี Shell Script เพื่อทำการแลกเปลี่ยนไฟล์ /etc/pf.conf ในเราเตอร์ทุกๆ ตัวที่ทำการเชื่อมโยงกัน ซึ่งมีข้อดีหากว่าเราเตอร์ตัวใดใช้การไม่ได้ขึ้นมา เราเตอร์ตัวอื่นๆ ก็สามารถทำงานต่อไปได้



รูปที่ 3.12 การประกาศกฎของไฟร์วอลล์ระหว่างเราเตอร์

และจากนั้น Shell Script จะทำการรวมกฎที่ได้จากการรวมและเรียงลำดับเส้นทางที่ถูกต้อง กับไฟล์ /etc/pf.conf ต้นฉบับ ด้วยคำสั่ง (11)

 # cat /etc/sort_mergerule.conf >> /etc/pf.conf (11)

เมื่อเราเตอร์ทุกตัวได้รับการประกาศกฎของไฟร์วอลล์ทั้งหมดแล้วก็จะมีคำสั่ง Route-to ขึ้นมาในไฟล์ /etc/pf.conf และทำการปิดเปิด Service PF : Packet Filter ขึ้นมาใหม่ ซึ่ง Shell Script เหล่านี้ได้ถูกตั้งในเราเตอร์ที่ตัวแล้ว โดยกฎของไฟร์วอลล์ยังคงอยู่เหมือนเดิม ซึ่งเพิ่มเติมคำสั่ง Route-to เข้าไปในไฟล์ /etc/pf.conf ของเราเตอร์ทุกๆตัวในเครือข่าย ดังนั้นการแลกกฎก็จะไม่มีผลกระทบใดๆ กับกฎของไฟร์วอลล์เดิม โดยมีคำสั่ง ปิดเปิด Service ของ PF : Packet Filter (12)

#pfctl -d //มีความหมายว่าทำการปิด service ของ packet filter
#pfctl -e //มีความหมายว่าทำการเปิด service ของ packet filter (12)

บทที่ 4
ผลการทดลองและผลการวิเคราะห์ความถูกต้องของการค้นหาเส้นทางเชิงนโยบายโดยใช้การรวมกฎของไฟร์วอลล์

ในบทนี้กล่าวถึงผลการทดลองค้นหาเส้นทาง หลังจากกฎของไฟร์วอลล์ได้ติดตั้งลงไปและค่า Route-to แสดงผลในไฟล์ /etc/pf.conf และแสดงเส้นทางการค้นหาเส้นทางที่ถูกต้องในรูปแบบของโมเดลและแผนผัง

4.1 โปรแกรมที่ใช้ในการทดสอบระบบ

4.1.1 ระบบปฏิบัติการ FreeBSD 7.0 ซึ่งเป็นระบบปฏิบัติการที่เป็นโอเพนซอร์ซซึ่งเป็นระบบปฏิบัติการ Unix ที่มีการใช้งานทางด้านวิจัยอย่างกว้างขวาง

4.1.2 PF: Packet Filter เป็นไฟร์วอลล์ที่ฝังมากับระบบปฏิบัติการในตระกูล BSD ซึ่งมีความเสถียร ความรวดเร็วและนิยมใช้กันมาในระดับ Unix Administrator

4.1.3 VMware Workstation โปรแกรมจำลองระบบปฏิบัติการซึ่งสามารถนำไปใช้ได้จริงโดยดึงทรัพยากรจากเครื่องที่ติดตั้ง สามารถจำลองได้พร้อมๆกันหลายเครื่อง

4.1.4 Shell Script ซึ่งเป็น Script ที่นิยมใช้บนระบบปฏิบัติการ Unix ในงานวิจัยชิ้นนี้จะใช้ Shell Script เพื่อการประกาศกฎ รวมกฎ ส่งผ่าน และเปิดปิด Service ของ PF:Packet Filter บนระบบปฏิบัติการ Unix

4.2 ค่าชี้วัดผลการทดลอง

ในส่วนนี้เป็นการทำการเลือกค่าชี้วัดผลการทดลองได้แก่

4.2.1 ความถูกต้องของการค้นหาเส้นทาง

4.2.2 การค้นหาเส้นทางในรูปแบบโทโพโลยีที่ต่างกัน

4.3 พารามิเตอร์ที่ใช้ในการจำลองระบบ

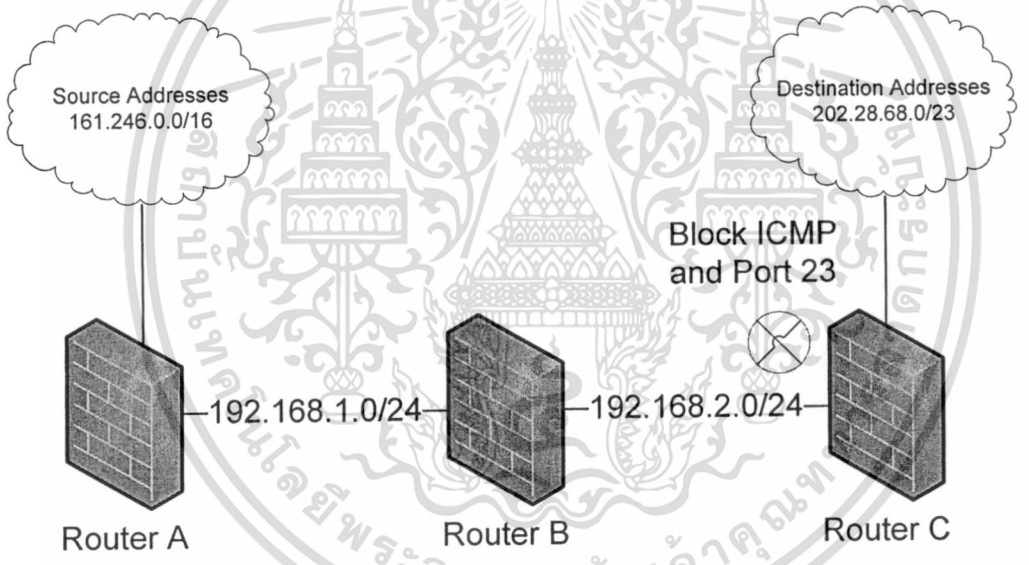
กฎของไฟร์วอลล์ในรูปแบบของ PF: Packet Filter โดยค่าที่จะเป็นตัวแปรหลักในการส่งออกหรือลบแพ็คเก็ตนั้นจะเป็นใช้กฎของ PF: Packet Filter

4.4 ข้อจำกัด

หากในโทโพโลยีทุกรูปแบบมีการปิดกั้นเส้นทางทั้งหมด จะไม่สามารถทำการส่งแพ็คเก็ตผ่านได้ ในการเปลี่ยนเส้นทางได้อัตโนมัติ ซึ่งการทดลองในบทที่ 4 นี้จะทำการวางกฎของไฟร์วอลล์ให้มีเส้นทางที่สามารถวิ่งผ่านได้ เพื่อให้การค้นหาเส้นทางสามารถ และสมมติฐานว่าการใช้คำสั่งบนไฟร์วอลล์ ถูกต้องตามหลักนโยบายและตามหลักของระบบ IP Addressing

4.5 ผลการทดลองโดยแยกโทโพโลยี

4.5.1 รูปแบบบัส (Bus) รูปที่ 4.1 แสดง Scenario การ Ping จาก 161.246.0.0 / 16 ไปยัง 202.28.68.0 ซึ่งทำการปิดกั้น ICMP และพอร์ต 23 ตรงตำแหน่ง Router C ซึ่งจะไม่สามารถ Ping และเปิด Telnet ได้



รูปที่ 4.1 รูปแบบบัส (Bus)

กฎของไฟร์วอลล์จาก Router C (13)

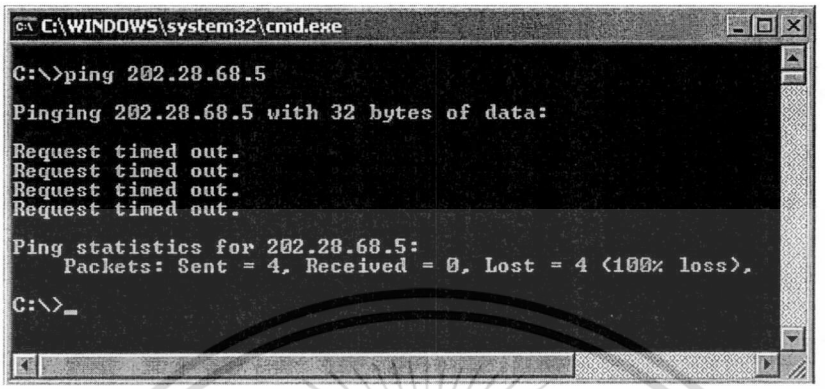
Block in on $external_interface proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23}

Pass in on $external_interfac from any to any

(13)

ผลการทดสอบในข้อ 4.5.1

1.) การ Ping จาก กลุ่มเครือข่าย 161.246.0.0/16 ไปยังกลุ่ม 202.28.68.0/23 ไม่สามารถทำได้เนื่องจากติดกฎของไฟร์วอลล์ ดังรูปที่ 4.2



รูปที่ 4.2 ผลการ Ping ไปยัง 202.28.68.5

2.) ไม่สามารถ Telnet ได้เนื่องจากติดกฎของไฟร์วอลล์คือพอร์ต 23 ดังรูปที่ 4.3

รูปที่ 4.3 ผลการ Telnet ไปยัง 202.28.69.99

สรุปผลการทดลองในหัวข้อที่ 4.5.1 นี้ไม่สามารถไปได้เนื่องจากเป็นโทโพโลยีแบบเส้นตรงไม่สามารถเปลี่ยนแปลงเส้นทางได้ ดังนั้นแพ็คเก็ตที่เป็นโพรโทคอล ICMP และพอร์ต 23 จะไม่สามารถทำการ Route ออกไปได้ตั้งแต่ Router A

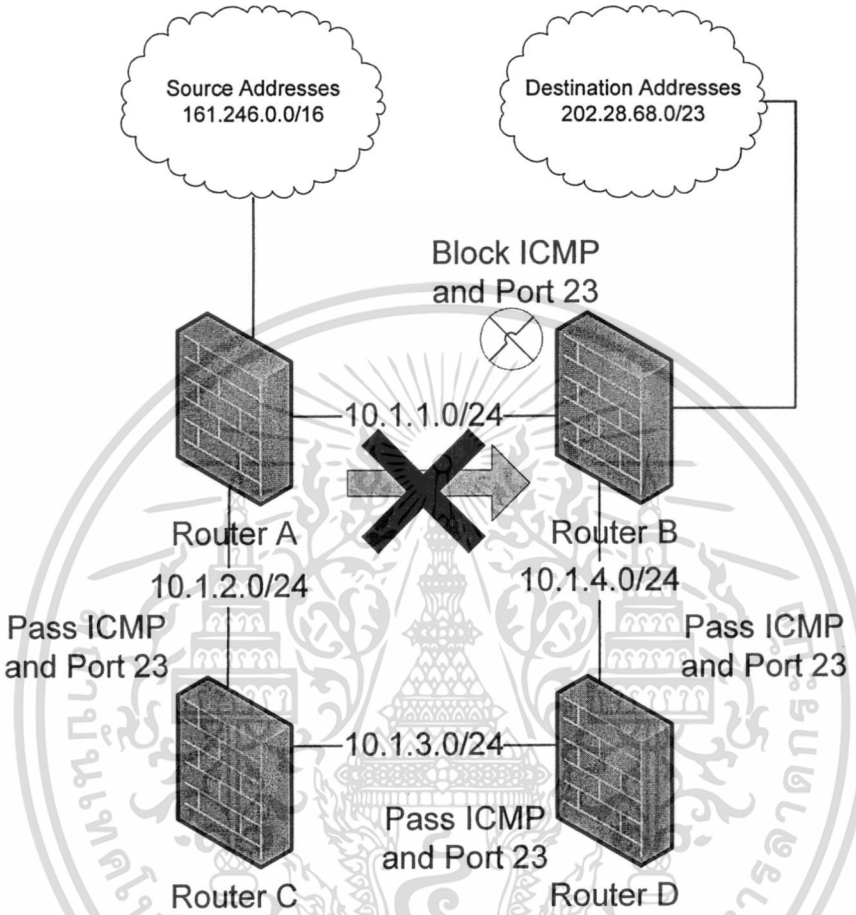
Router A ไฟล์ /etc/pf.conf (14)

Pass out on $internal_interface route-to ($external_interface 192.168.1.254) proto !ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23} (14)

Router B ไฟล์ /etc/pf.conf (15)

Pass out on $internal_interface route-to ($external_interface 192.168.2.254) proto !ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23} (15)

4.5.2 รูปแบบวงแหวน (Ring) รูปที่ 4.4 แสดง Scenario การ Ping จาก 161.246.0.0 / 16 ไปยัง 202.28.68.0 ซึ่งทำการปิดกั้น ICMP และ Port 23 ตรงตำแหน่ง Router B ซึ่งจะไม่สามารถ Ping และเปิด Telnet ได้



รูปที่ 4.4 รูปแบบวงแหวน (Ring) ที่ถูกกฎของไฟร์วอลล์บล็อก

กฎของไฟร์วอลล์จาก Router B (16)

Block in on $external_interface proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23}

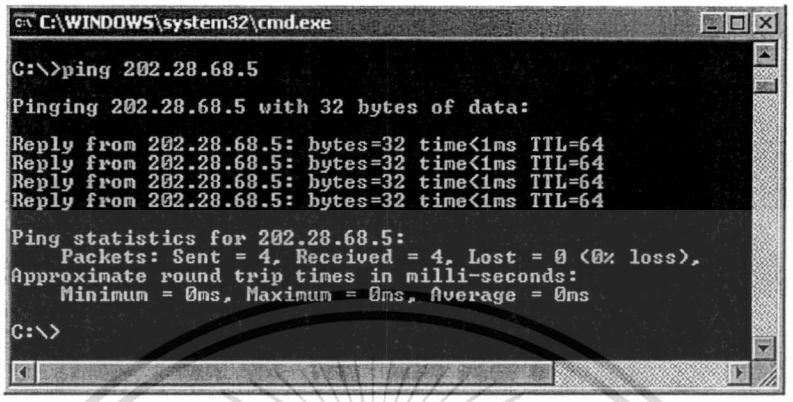
Pass in on $external_interface1 from any to any

Pass in on $external_interface2 from any to any

(16)

ผลการทดสอบในข้อ 4.5.2

1.) การ Ping จาก กลุ่มเครือข่าย 161.246.0.0/16 ไปยังกลุ่ม 202.28.68.0/23 สามารถทำได้ เนื่องจากเราเตอร์ได้เปลี่ยนเส้นทางเพื่อหลีกเลี่ยงฝั่ง Router A – Router B รูปที่ 4.5



รูปที่ 4.5 ผลการ Ping ไปยัง 202.28.68.5

2.) สามารถ Telnet ได้เนื่องจากเราเตอร์ได้เปลี่ยนเส้นทางเพื่อหลีกเลี่ยงฝั่ง Router A – Router B รูปที่ 4.6

รูปที่ 4.6 ผลการ Telnet ไปยัง 202.28.69.99

สรุปผลการทดลองในหัวข้อ 4.5.2 นั้นเราเตอร์สามารถเปลี่ยนแปลงเส้นทางได้จาก Routing Table ก่อนหน้าที่ Router A -> Router B โดยตรงนั้น ไม่สามารถทำได้ หลักจากการค้นเส้นทางได้ถูกติดตั้งลงไป Router A จึงค้นเส้นทางไปยังเส้นอื่นที่ไม่ติดกฎของไฟร์วอลล์ โดยเส้นทางที่ Router A ส่งต่อใหม่นั้นคือ Router A -> Router C -> Router D -> Router B -> โดยสามารถดูได้จากคำสั่ง Route-to ที่เพิ่มขึ้นมาใน Router A , Router C และ Router D ดังนี้

Router A ไฟล์ /etc/pf.conf (17)

Pass out on $internal_interface route-to ($external_interface2 192.168.2.254) proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23}

Pass out on $internal_interface route-to ($external_interface1 192.168.1.254) proto !ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {!23} (17)

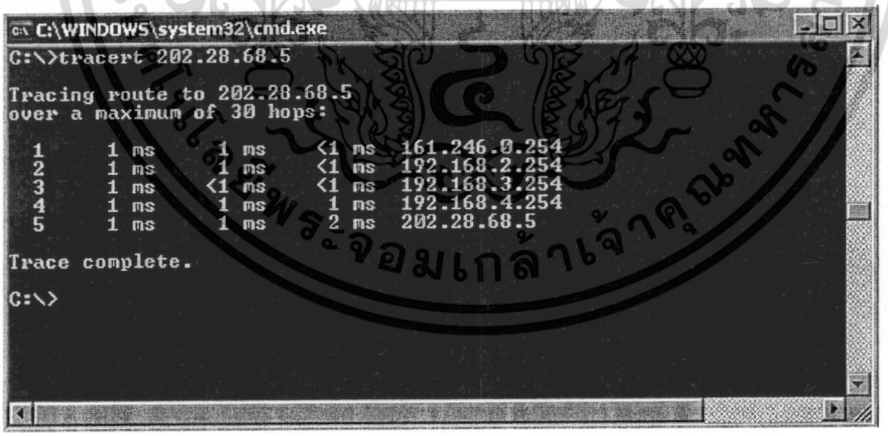
Router C ไฟล์ /etc/pf.conf (18)

Pass out on $external_interface1 route-to ($external_interface2 192.168.3.254) proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23} (18)

Router D ไฟล์ /etc/pf.conf (19)

Pass out on $external_interface1 route-to ($external_interface2 192.168.4.254) proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23} (19)

ผลการ Tracert จาก IP ต้นทางไปยังปลายทาง



รูปที่ 4.7 การ Tracert จาก 161.246.0.0/16 ไปยัง 202.28.68.0/23

ผลการทดลอง Tracert จากต้นทาง 161.246.0.0/16 ไปยัง 202.28.68.0/23 นั้นดังรูปที่ 4.7 ซึ่งโดยเรียงลำดับ ดังนี้

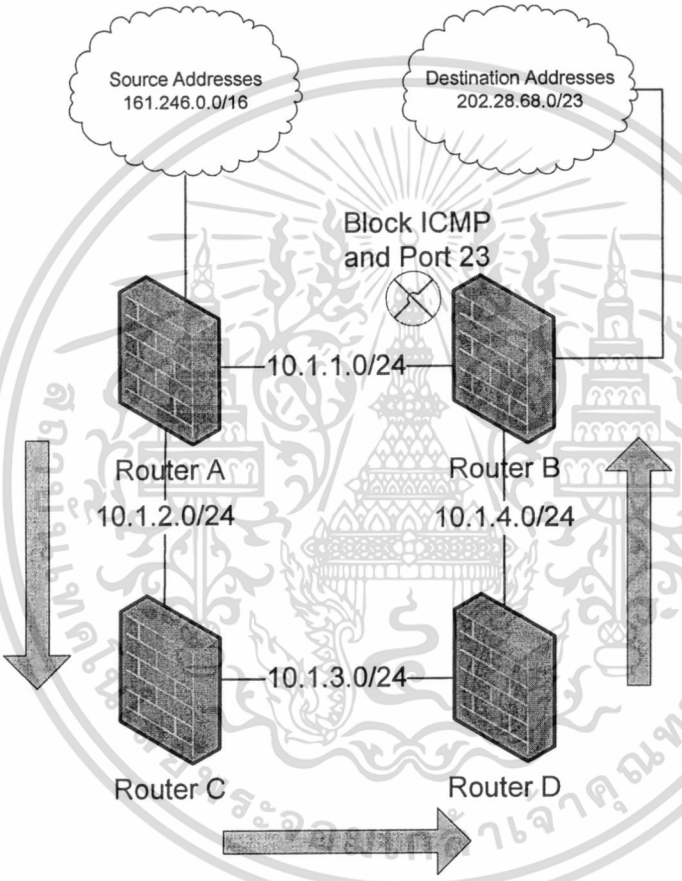
Hop ที่ 1 คือ Router A

Hop ที่ 2 คือ Router C

Hop ที่ 3 คือ Router D

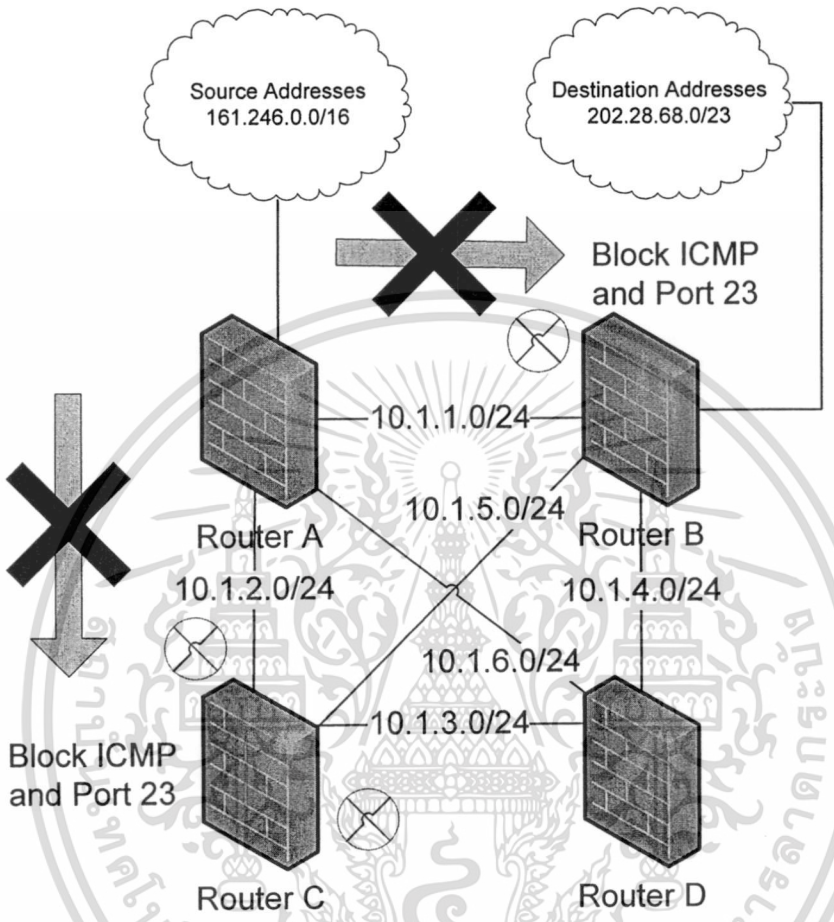
Hop ที่ 4 คือ Router B

Hop ที่ 5 คือ หมายเลข IP ปลายทางที่ทำการ Tracert ดังรูปที่ 4.8



รูปที่ 4.8 รูปแบบวงแหวน (Ring) ที่แพ็คเก็ตทำการเปลี่ยนแปลงเส้นทางไปยังเส้นทาง Pass

4.5.3 รูปแบบตาข่าย (Mesh) รูปที่ 4.9 แสดง Scenario การ Ping จาก 161.246.0.0 / 16 ไปยัง 202.28.68.0 ซึ่งทำการปิดกั้น ICMP และ Port 23 ตรงตำแหน่ง Router A -> Router B และ Router A -> Router C ซึ่งจะไม่สามารถ Ping และเปิด Telnet ได้



รูปที่ 4.9 รูปแบบตาข่าย (Mesh) ที่ถูกกฎของไฟร์วอลล์บล็อก

กฎของไฟร์วอลล์จาก Router B (20)

Block in on $external_interface proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23}

Pass in on $external_interface from any to any (20)

กฎของไฟร์วอลล์จาก Router C (21)

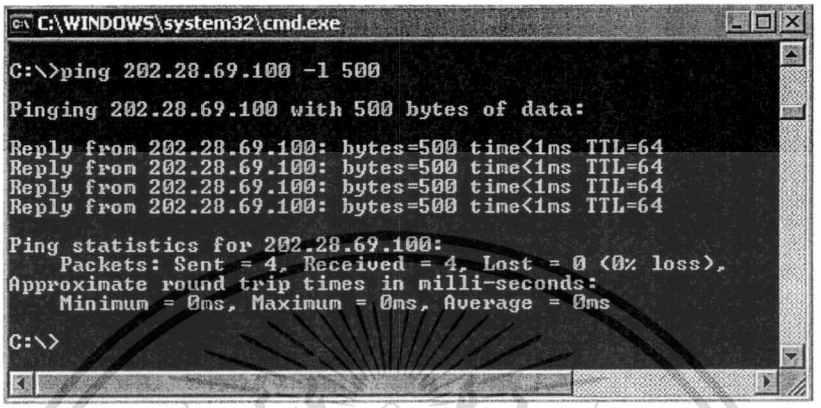
Block in on $external_interface proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23}

Block in on $external_interface2 proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23}

Pass in on $external_interface from any to any (21)

ผลการทดสอบในข้อ 4.5.3

1.) การ Ping จาก กลุ่มเครือข่าย 161.246.0.0/16 ไปยังกลุ่ม 202.28.68.0/23 สามารถทำได้ เนื่องจากเราเตอร์ได้เปลี่ยนเส้นทางเพื่อหลีกเลี่ยงฝั่ง Router A –> Router B และ Router A -> Router C รูปที่ 4.10



รูปที่ 4.10 ผลการ Ping ไปยัง 202.28.69.100 โดยใช้ขนาดของแพ็คเก็ต 500

2.) สามารถ Telnet ได้เนื่องจากเราเตอร์ได้เปลี่ยนเส้นทางเพื่อหลีกเลี่ยงฝั่ง Router A -> Router B และ Router A -> Router C รูปที่ 4.11

รูปที่ 4.11 ผลการ Telnet ไปยัง 202.28.69.99

สรุปผลการทดลองในหัวข้อ 4.5.3 นั้น Router A -> Router B และ Router A -> Router C นั้นจะทำการปิดกั้นทั้งสองเส้นทาง ซึ่งทำให้การค้นหาเส้นทางซับซ้อนมากยิ่งขึ้น แต่ก็สามารถเปลี่ยนแปลงเส้นทางได้จาก Routing Table ก่อนหน้าที่ Router A -> Router B โดยตรงนั้น ไม่สามารถทำได้ หลักจากการค้นเส้นทางได้ถูกติดตั้งลงไป Router A จึงค้นเส้นทางไปยังเส้นอื่นที่

ไม่ติดกฎของไฟร์วอลล์ โดยเส้นทางที่ Router A ส่งต่อใหม่นั้นคือ Router A -> Router D -> Router B -> โดยสามารถดูได้จากคำสั่ง Route-to ที่เพิ่มขึ้นมาใน Router A และ Router D ดังนี้

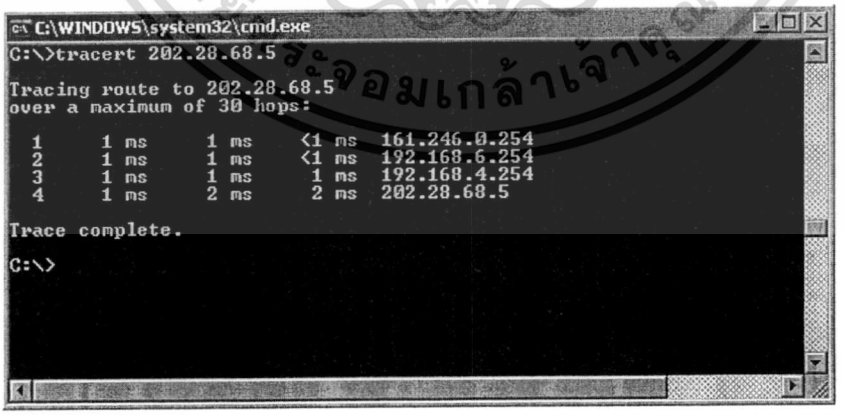
Router A ไฟล์ /etc/pf.conf (22)

Pass out on $internal_interface route-to ($external_interface1 192.168.1.254) proto !ICMP from 161.246.0.0/16 to 202.28.68.0/23 port !{23}
Pass out on $internal_interface route-to ($external_interface2 192.168.2.254) proto !ICMP from 161.246.0.0/16 to 202.28.68.0/23 port !{23}
Pass out on $internal_interface route-to ($external_interface3 192.168.6.254) proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23} (22)

Router D ไฟล์ /etc/pf.conf (23)

Pass out on $external_interface1 route-to ($external_interface1 192.168.3.254) proto !ICMP from 161.246.0.0/16 to 202.28.68.0/23 port !{23}
Pass out on $external_interface1 route-to ($external_interface2 192.168.4.254) proto ICMP from 161.246.0.0/16 to 202.28.68.0/23 port {23} (23)

ผลการ Tracert จาก IP ต้นทาง ไปยังปลายทาง



รูปที่ 4.12 การ Tracert จาก 161.246.0.0/16 ไปยัง 202.28.68.0/23

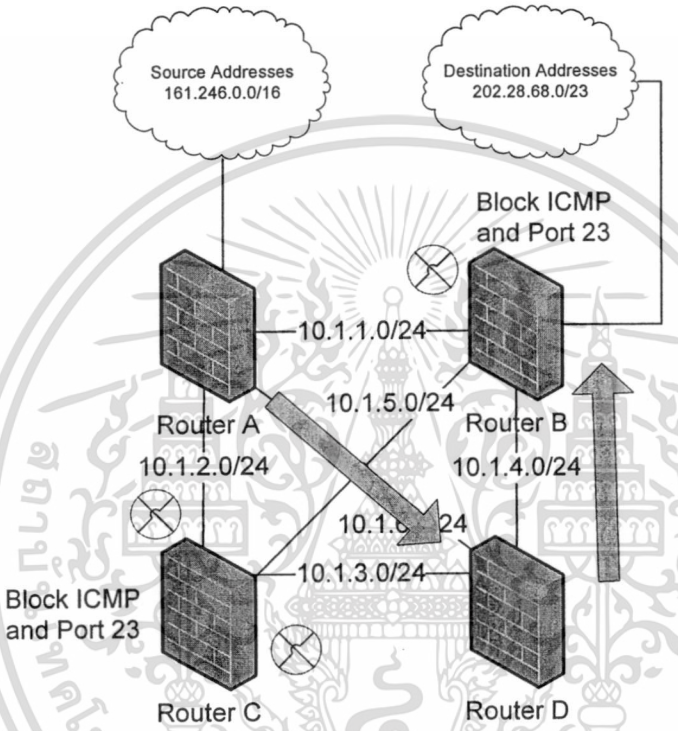
ผลการทดลอง Tracert จากต้นทาง 161.246.0.0/16 ไปยัง 202.28.68.0/23 นั้นดังรูปที่ 4.12 ซึ่งโดยเรียงลำดับ ดังนี้

Hop ที่ 1 คือ Router A

Hop ที่ 2 คือ Router D

Hop ที่ 3 คือ Router B

Hop ที่ 4 คือ หมายเลข IP ปลายทางที่ทำการ Tracert ดังรูปที่ 4.13



รูปที่ 4.13 รูปแบบตาข่าย (Mesh) ที่แพ็คเก็ตทำการเปลี่ยนแปลงเส้นทางไปยังเส้นทาง Pass

4.6 ผลการทดลองของการรวมกฎไฟร์วอลล์และผลความถูกต้องของการเรียงกฎไฟร์วอลล์

จากผลการทดลองในหัวข้อ 4.5 เป็นผลการทดลองในรูปแบบการทดสอบความถูกต้องของการนำแนวคิดการค้นหาเส้นทางโดยใช้การรวมกฎของไฟร์วอลล์ โดยนำกฎที่ได้จากการคำนวณในบทที่ 3 นั้นนำมาใช้ในสถานการณ์จริงซึ่งได้สร้าง Scenario รูปแบบต่างๆเพื่อทำการแสดงให้เห็นว่าสามารถทำงานตามสมมติฐานของบทที่ 1 ได้จริง

ดังนั้นในหัวข้อนี้จะแสดง ผลลัพธ์ของการรวมกฎไฟร์วอลล์ และการเรียงกฎของไฟร์วอลล์ด้วยจำนวนกฎที่มากและสลับซับซ้อน เพื่อแสดงให้เห็นว่ากฎของไฟร์วอลล์และคำสั่ง Route-to ได้ถูกเรียงตามกฎของ Weight ที่มี Router_ID เป็นค่าน้ำหนัก และ Prefix Matching ของ Subnet Mask ที่เรียงลำดับจากมากที่สุดคือ /32 ไปยัง Any ที่มีค่าน้อยที่สุดดังต่อไปนี้

พารามิเตอร์และผลลัพธ์ที่ได้จากการทดสอบ

4.6.1 Rule Set ของเราเตอร์แต่ละตัวเป็นจำนวน 3 ชุด โดยแต่ Rule Set จะมีกฎของไฟร์วอลล์จำนวน 30 กฎ

4.6.2 การแสดงผลการทดสอบการเรียงกฎนั้นจะแสดงให้เห็นว่ากฎนั้นถูกเรียงโดยเรียงลำดับจากมากที่สุดคือ /32 ไปยัง Any ที่มีค่าน้อยที่สุด

4.6.3 หากกฎที่ถูกเรียงนั้นมีค่า Prefix Matching เท่ากันก็จะดูจาก น้ำหนักของ Router_ID โดยในงานวิจัยชิ้นนี้จะเรียงจาก Router_ID น้อยไปมาก

ผลลัพธ์ของการรวมกฎของไฟร์วอลล์ ซึ่งรวม Rule Set จากเราเตอร์แสดงดังตารางที่ 4.1

ตารางที่ 4.1 ผลการรวมกฎของไฟร์วอลล์ merge_tmp.conf

1	pass in on $interface1 proto ICMP from 161.246.1.13/32 to 202.28.68.0/23 port {80,664,55,66} #$r1
2	block in on $interface1 proto ICMP from 161.246.9.0/24 to 202.28.69.128/25 port {134,764,44334} #$r1
3	block in on $interface1 proto ICMP from 161.246.0.0/16 to any port {1024><2048} #$r1
4	block in on $interface1 proto ICMP from 161.246.0.0/8 to 202.28.69.96/30 port {77,66,22} #$r1
5	pass in on $interface1 proto ICMP from 161.246.0.0/4 to any port {80} #$r1
6	block in on $interface1 proto ICMP from 161.246.0.0/2 to any port {8080} #$r1
7	block in on $interface1 proto ICMP from 161.246.0.0/1 to 202.28.68.224/30 port {443,4433} #$r1
8	pass in on $interface1 proto ICMP from 161.246.240.240/30 to any port {50000<=} #$r1
9	block in on $interface1 proto ICMP from 161.246.52.16/28 to 202.28.69.0/27 port {33,99,7777} #$r1
10	block in on $interface1 proto ICMP from 161.246.38.128/26 to 202.28.68.16/28 port {9999,1000<>8000} #$r1
11	pass in on $interface1 proto ICMP from 161.246.64.0/22 to any port {5050} #$r1
12	block in on $interface1 proto ICMP from 161.246.224.0/20 to any port {756,15132} #$r1
13	block in on $interface1 proto ICMP from 161.246.4.0/18 to any port {8546,65541} #$r1
14	block in on $interface1 proto ICMP from 161.246.128.0/14 to any port {3486,8454,5321,546} #$r1
15	pass in on $interface1 proto ICMP from 161.246.0.0/10 to 202.28.69.252/30 port {444,333,875} #$r1

ตารางที่ 4.1 (ต่อ) ผลการรวมกฎของไฟร์วอลล์ merge_tmp.conf

16	block in on $interface1 proto ICMP from 161.246.0.0/12 to any port {9945,2313,546} #$r1
17	block in on $interface1 proto ICMP from 161.246.0.0/19 to 202.28.68.0/24 port {1024>=} #$r1
18	block in on $interface1 proto ICMP from 161.246.0.0/8 to any port {1024<=} #$r1
19	pass in on $interface1 proto ICMP from 161.246.99.32/31 to any port {54648,12312} #$r1
20	block in on $interface1 proto ICMP from 161.246.103.64/27 to any port {6333} #$r1
21	block in on $interface1 proto ICMP from 161.246.222.128/25 to 202.28.0.0/16 port {4646,12132} #$r1
22	pass in on $interface1 proto ICMP from 161.246.250.0/23 to any port {62,4549,7445,999><1100} #$r1
23	block in on $interface1 proto ICMP from 161.246.251.240/29 to any port {10000>=} #$r1
24	pass in on $interface1 proto ICMP from 161.246.41.128/27 to any port {8888>=} #$r1
25	block in on $interface1 proto ICMP from 161.246.53.16/28 to 202.28.68.0/23 port {7538,3} #$r1
26	block in on $interface1 proto ICMP from 161.246.88.0/25 to any port {20,21,23} #$r1
27	block in on $interface1 proto ICMP from 161.246.88.128/25 to any port {5555,6666} #$r1
28	pass in on $interface1 proto ICMP from 161.246.0.0/7 to any port {1024><65530} #$r1
29	block in on $interface1 proto ICMP from 161.246.131.33/32 to 202.28.69.240/32 port {80,8080,5500,7745} #$r1
30	pass in on $interface1 proto ICMP from 161.246.9.96/29 to 202.28.69.32/29 port {8886,44433} #$r1
31	block in on $interface1 proto ICMP from 161.246.66.13/32 to 202.28.68.0/23 port {80,664,55,66} #$r2
32	block in on $interface1 proto ICMP from 161.246.99.0/24 to 202.28.69.128/25 port {134,76442,44334} #$r2
33	block in on $interface1 proto ICMP from 161.246.0.0/16 to any port {52424} #$r2
34	block in on $interface1 proto ICMP from 161.246.0.0/8 to 202.28.69.96/30 port {77,786,22} #$r2
35	pass in on $interface1 proto ICMP from 161.246.0.0/4 to any port {254} #$r2
36	block in on $interface1 proto ICMP from 161.246.0.0/2 to any port {7282} #$r2
37	block in on $interface1 proto ICMP from 161.246.0.0/1 to 202.28.68.224/30 port {443,4433} #$r2
38	block in on $interface1 proto ICMP from 161.246.224.224/30 to any port {6000<=} #$r2
39	pass in on $interface1 proto ICMP from 161.246.88.32/28 to 202.28.69.0/27 port {78,99,7777} #$r2
40	block in on $interface1 proto ICMP from 161.246.100.96/26 to 202.28.68.16/28 port {9999} #$r2
41	pass in on $interface1 proto ICMP from 161.246.128.0/22 to any port {5050} #$r2
42	block in on $interface1 proto ICMP from 161.246.64.0/20 to any port {275,15132} #$r2
43	block in on $interface1 proto ICMP from 161.246.8.0/18 to any port {8546,4245} #$r2
44	block in on $interface1 proto ICMP from 161.246.128.0/14 to any port {3486,8454,5321,546} #$r2
45	block in on $interface1 proto ICMP from 161.246.0.0/10 to 202.28.69.252/30 port {444,333,875} #$r2
46	block in on $interface1 proto ICMP from 161.246.0.0/12 to any port {88,2313,546} #$r2
47	block in on $interface1 proto ICMP from 161.246.0.0/19 to 202.28.68.0/24 port {1024>=} #$r2
48	block in on $interface1 proto ICMP from 161.246.0.0/8 to any port {1024<=} #$r2
49	pass in on $interface1 proto ICMP from 161.246.55.74/31 to any port {54648,12312} #$r2
50	block in on $interface1 proto ICMP from 161.246.205.96/27 to any port {6333} #$r2
51	block in on $interface1 proto ICMP from 161.246.238.0/25 to 202.28.0.0/16 port {4646,12132} #$r2
52	pass in on $interface1 proto ICMP from 161.246.140.0/23 to any port {62,4549,7445,999} #$r2
53	block in on $interface1 proto ICMP from 161.246.11.16/29 to any port {10000>=} #$r2
54	pass in on $interface1 proto ICMP from 161.246.76.128/27 to any port {8888>=} #$r2
55	block in on $interface1 proto ICMP from 161.246.96.16/28 to 202.28.68.0/23 port {7538,3} #$r2
56	block in on $interface1 proto ICMP from 161.246.77.0/25 to any port {20,21,23} #$r2
57	block in on $interface1 proto ICMP from 161.246.8.128/25 to any port {444,7} #$r2
58	block in on $interface1 proto ICMP from 161.246.0.0/7 to any port {253><500} #$r2

ตารางที่ 4.1 (ต่อ) ผลการรวมกฎของไฟร์วอลล์ merge_tmp.conf

59	block in on $interface1 proto ICMP from 161.246.169.69/32 to 202.28.69.240/32 port {80,8080,5500,7745} #$r2
60	block in on $interface1 proto ICMP from 161.246.19.224/29 to 202.28.69.32/29 port {8886,44433} #$r2
61	block in on $interface1 proto ICMP from 161.246.88.0/24 to 202.28.68.0/23 port {80,664,55,66} #$r3
62	block in on $interface1 proto ICMP from 161.246.55.78/32 to 202.28.69.128/25 port {134,764,44334} #$r3
63	block in on $interface1 proto ICMP from 161.246.0.0/16 to any port {1024><2048} #$r3
64	pass in on $interface1 proto ICMP from 161.246.0.0/8 to 202.28.69.96/30 port {77,66,22} #$r3
65	block in on $interface1 proto ICMP from 161.246.0.0/4 to any port {80} #$r3
66	pass in on $interface1 proto ICMP from 161.246.0.0/2 to any port {8080} #$r3
67	block in on $interface1 proto ICMP from 161.246.0.0/1 to 202.28.68.224/30 port {443,4433} #$r3
68	pass in on $interface1 proto ICMP from 161.246.240.240/30 to any port {50000<=} #$r3
69	block in on $interface1 proto ICMP from 161.246.52.16/28 to 202.28.69.0/27 port {33,99,7777} #$r3
70	pass in on $interface1 proto ICMP from 161.246.38.128/26 to 202.28.68.16/28 port {9999,1000<>8000} #$r3
71	pass in on $interface1 proto ICMP from 161.246.64.0/22 to any port {5050} #$r3
72	pass in on $interface1 proto ICMP from 161.246.224.0/20 to any port {756,15132} #$r3
73	pass in on $interface1 proto ICMP from 161.246.4.0/18 to any port {8546,65541} #$r3
74	pass in on $interface1 proto ICMP from 161.246.128.0/14 to any port {3486,8454,5321,546} #$r3
75	block in on $interface1 proto ICMP from 161.246.0.0/10 to 202.28.69.252/30 port {444,333,875} #$r3
76	block in on $interface1 proto ICMP from 161.246.0.0/12 to any port {9945,2313,546} #$r3
77	pass in on $interface1 proto ICMP from 161.246.0.0/19 to 202.28.68.0/24 port {1024>=} #$r3
78	block in on $interface1 proto ICMP from 161.246.0.0/8 to any port {1024<=} #$r3
79	block in on $interface1 proto ICMP from 161.246.99.32/31 to any port {54648,12312} #$r3
80	pass in on $interface1 proto ICMP from 161.246.103.64/27 to any port {6333} #$r3
81	pass in on $interface1 proto ICMP from 161.246.222.128/25 to 202.28.0.0/16 port {4646,12132} #$r3
82	pass in on $interface1 proto ICMP from 161.246.250.0/23 to any port {62,4549,7445,999><1100} #$r3
83	pass in on $interface1 proto ICMP from 161.246.251.240/29 to any port {10000>=} #$r3
84	block in on $interface1 proto ICMP from 161.246.41.128/27 to any port {8888>=} #$r3
85	block in on $interface1 proto ICMP from 161.246.53.16/28 to 202.28.68.0/23 port {7538,3} #$r3
86	block in on $interface1 proto ICMP from 161.246.88.0/25 to any port {20,21,23} #$r3
87	block in on $interface1 proto ICMP from 161.246.88.128/25 to any port {5555,6666} #$r3
88	pass in on $interface1 proto ICMP from 161.246.0.0/7 to any port {1024><65530} #$r3
89	block in on $interface1 proto ICMP from 161.246.131.33/32 to 202.28.69.240/32 port {80,8080,5500,7745} #$r3
90	pass in on $interface1 proto ICMP from 161.246.9.96/29 to 202.28.69.32/29 port {8886,44433} #$r3

ผลลัพธ์ของการเรียงลำดับกฎของไฟร์วอลล์ ดังตารางที่ 4.2 ซึ่งทำการเรียงกฎตามลำดับค่าน้ำหนักและ Prefix Matching ของ Subnet Mask

ตารางที่ 4.2 ผลการเรียงลำดับของการรวมกฎไฟร์วอลล์ Sort_mergerule.conf

1	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.1.13/32 to 202.28.68.0/23 port {80,664,55,66}
2	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.131.33/32 to 202.28.69.240/32 port !{80,8080,5500,7745}
3	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.66.13/32 to 202.28.68.0/23 port !{80,664,55,66}
4	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.169.69/32 to 202.28.69.240/32 port !{80,8080,5500,7745}
5	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.55.78/32 to 202.28.69.128/25 port !{134,764,44334}
6	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.131.33/32 to 202.28.69.240/32 port !{80,8080,5500,7745}
7	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.99.32/31 to any port {54648,12312}
8	pass out on $internal route-to ($interface1,$r2) proto ICMP from 161.246.55.74/31 to any port {54648,12312}
9	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.99.32/31 to any port !{54648,12312}
10	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.240.240/30 to any port {50000<=}
11	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.224.224/30 to any port !{6000<=}
12	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.240.240/30 to any port {50000<=}
13	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.251.240/29 to any port !{10000>=}
14	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.9.96/29 to 202.28.69.32/29 port {8886,44433}
15	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.11.16/29 to any port !{10000>=}
16	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.19.224/29 to 202.28.69.32/29 port !{8886,44433}
17	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.251.240/29 to any port {10000>=}
18	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.9.96/29 to 202.28.69.32/29 port {8886,44433}
19	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.52.16/28 to 202.28.69.0/27 port !{33,99,7777}
20	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.53.16/28 to 202.28.68.0/23 port !{7538,3}
21	pass out on $internal route-to ($interface1,$r2) proto ICMP from 161.246.88.32/28 to 202.28.69.0/27 port {78,99,7777}
22	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.96.16/28 to 202.28.68.0/23 port !{7538,3}
23	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.52.16/28 to 202.28.69.0/27 port !{33,99,7777}
24	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.53.16/28 to 202.28.68.0/23 port !{7538,3}
25	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.103.64/27 to any port !{6333}
26	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.41.128/27 to any port {8888>=}
27	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.205.96/27 to any port !{6333}
28	pass out on $internal route-to ($interface1,$r2) proto ICMP from 161.246.76.128/27 to any port {8888>=}
29	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.103.64/27 to any port {6333}
30	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.41.128/27 to any port !{8888>=}
31	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.38.128/26 to 202.28.68.16/28 port !{9999,1000<>8000}
32	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.100.96/26 to 202.28.68.16/28 port !{9999}
33	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.38.128/26 to 202.28.68.16/28 port {9999,1000<>8000}
34	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.222.128/25 to 202.28.0.0/16 port !{4646,12132}
35	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.88.0/25 to any port !{20,21,23}
36	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.88.128/25 to any port !{5555,6666}
37	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.238.0/25 to 202.28.0.0/16 port !{4646,12132}
38	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.77.0/25 to any port !{20,21,23}
39	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.8.128/25 to any port !{444,7}
40	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.222.128/25 to 202.28.0.0/16 port {4646,12132}
41	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.88.0/25 to any port !{20,21,23}
42	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.88.128/25 to any port !{5555,6666}
43	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.9.0/24 to 202.28.69.128/25 port !{134,764,44334}
44	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.99.0/24 to 202.28.69.128/25 port !{134,76442,44334}
45	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.88.0/24 to 202.28.68.0/23 port !{80,664,55,66}
46	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.250.0/23 to any port {62,4549,7445,999><1100}
47	pass out on $internal route-to ($interface1,$r2) proto ICMP from 161.246.140.0/23 to any port {62,4549,7445,999}

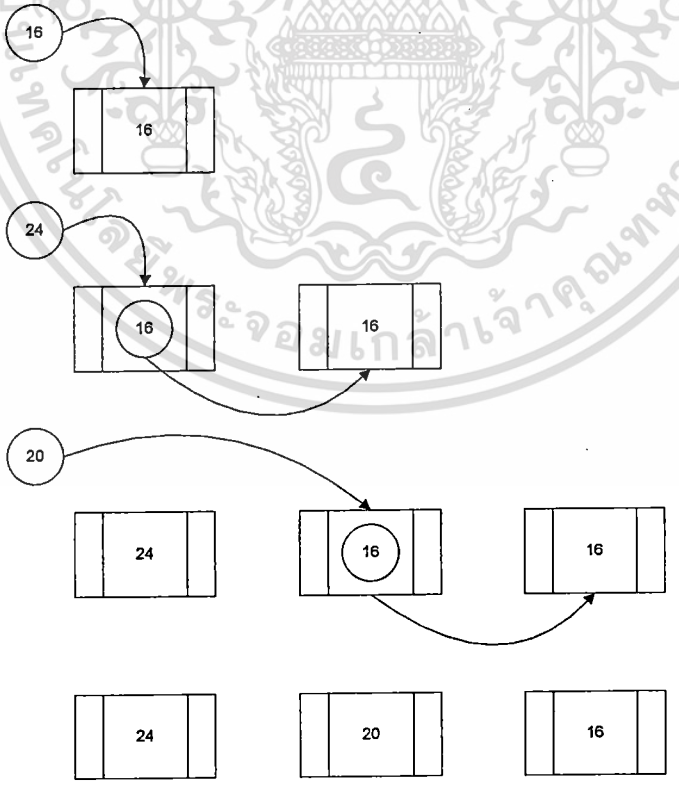
ตารางที่ 4.2 (ต่อ) ผลการเรียงลำดับของการรวมกฎไฟร์วอลล์ Sort_mergerule.conf

48	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.250.0/23 to any port {62,4549,7445,999><1100}
49	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.64.0/22 to any port {5050}
50	pass out on $internal route-to ($interface1,$r2) proto ICMP from 161.246.128.0/22 to any port {5050}
51	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.64.0/22 to any port {5050}
52	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.224.0/20 to any port !{756,15132}
53	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.64.0/20 to any port !{275,15132}
54	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.224.0/20 to any port {756,15132}
55	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.0.0/19 to 202.28.68.0/24 port !{1024>=}
56	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/19 to 202.28.68.0/24 port !{1024>=}
57	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.0.0/19 to 202.28.68.0/24 port {1024>=}
58	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.4.0/18 to any port !{8546,65541}
59	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.8.0/18 to any port !{8546,4245}
60	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.4.0/18 to any port {8546,65541}
61	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.0.0/16 to any port !{1024><2048}
62	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/16 to any port !{52424}
63	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.0.0/16 to any port !{1024><2048}
64	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.128.0/14 to any port !{3486,8454,5321,546}
65	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.128.0/14 to any port !{3486,8454,5321,546}
66	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.128.0/14 to any port {3486,8454,5321,546}
67	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.0.0/12 to any port !{9945,2313,546}
68	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/12 to any port !{88,2313,546}
69	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.0.0/12 to any port !{9945,2313,546}
70	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.0.0/10 to 202.28.69.252/30 port {444,333,875}
71	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/10 to 202.28.69.252/30 port !{444,333,875}
72	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.0.0/10 to 202.28.69.252/30 port !{444,333,875}
73	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.0.0/8 to 202.28.69.96/30 port !{77,66,22}
74	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.0.0/8 to any port !{1024<=}
75	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/8 to 202.28.69.96/30 port !{77,786,22}
76	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/8 to any port !{1024<=}
77	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.0.0/8 to 202.28.69.96/30 port {77,66,22}
78	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.0.0/8 to any port !{1024<=}
79	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.0.0/7 to any port {1024><65530}
80	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/7 to any port !{253><500}
81	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.0.0/7 to any port {1024><65530}
82	pass out on $internal route-to ($interface1,$r1) proto ICMP from 161.246.0.0/4 to any port {80}
83	pass out on $internal route-to ($interface1,$r2) proto ICMP from 161.246.0.0/4 to any port {254}
84	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.0.0/4 to any port !{80}
85	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.0.0/2 to any port !{8080}
86	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/2 to any port !{7282}
87	pass out on $internal route-to ($interface1,$r3) proto ICMP from 161.246.0.0/2 to any port {8080}
88	pass out on $internal route-to ($interface1,$r1) proto !ICMP from 161.246.0.0/1 to 202.28.68.224/30 port !{443,4433}
89	pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.0.0/1 to 202.28.68.224/30 port !{443,4433}
90	pass out on $internal route-to ($interface1,$r3) proto !ICMP from 161.246.0.0/1 to 202.28.68.224/30 port !{443,4433}

4.7 วิเคราะห์ความถูกต้องของการเรียงลำดับกฎการค้นหาเส้นทางเชิงนโยบาย

จากผลการทดลองในหัวข้อที่ 4.6 ที่ได้แสดงผลลัพธ์ของจำนวนกฎของไฟร์วอลล์ที่ได้จากการรวมกันของ Rule Set ของเราเตอร์ทุกตัว และผลการเรียงลำดับกฎการค้นหาเส้นทาง Route-to ตามค่าน้ำหนัก Router_ID และ Prefix Matching ของ Subnet Mask นั้น วิเคราะห์การเรียงลำดับได้ถูกต้องดังนี้

4.7.1 การเรียงลำดับของ Prefix Matching ของ Subnet Mask เรียงลำดับจากมากที่สุดคือ /32 จนไปถึง /1 ที่มีค่าน้อยที่สุด ในลำดับของ Source Address ซึ่งแสดงให้เห็นว่าได้มีการเรียงลำได้อย่างถูกต้องโดยการเรียงลำดับในโปรแกรมนั้นจะทำการเทียบ Subnet Mask ที่ได้มาจาก Rule Set ทีละตัวไปเรื่อยๆ หากเจอค่าที่มี Subnet Mask มากกว่าก็จะแทรกไปยัง Node นั้นแทน และเลื่อน Node ที่น้อยกว่าไปยัง Node ถัดไป ดังรูปที่ 4.14 ซึ่งจำลองการเรียงลำดับ เช่นหากมีหมายเลข 16 เข้าไปก็จะถูกเก็บลงใน Node แรก หลังจากนั้นมีหมายเลข 24 เข้ามาก็จะเทียบกับ Node แรกว่ามากกว่าหรือน้อยกว่า หากมากกว่าก็จะเลื่อน Node แรกออกไปแล้วแทรกตัวเลขที่มากกว่าลงไปแทน หลังจากนั้นมีเลข 20 เข้ามาจึงเทียบกับเลขแรกซึ่ง 20 น้อยกว่าก็จะเลื่อนไปดู Node ถัดไป หากเจอ Node ไหนที่น้อยกว่าก็จะแทรกลงไปเรื่อยๆ ตามลำดับ โดยสามารถดู Source Code ได้ที่ภาคผนวก ก.



รูปที่ 4.14 การเรียงลำดับของ Prefix Matching ของงานวิจัยนี้

4.7.2 หาก Subnet Mask มีขนาดเท่ากันแล้วเงื่อนไขต่อไปที่จะดูคือ Router_ID ซึ่งแทรกไว้ใน Rule Set แต่ละตัว เมื่อทำการเรียงลำดับการค้นหาเส้นทางใหม่ก็จะแสดงน้ำหนักนี้ไว้ที่ Next Hop ของกฎแต่ละบรรทัด ซึ่ง Router_ID ที่มีค่าน้อยกว่าจะมี Weight ที่มากกว่า Router_ID ที่มีค่ามากกว่า ซึ่งสามารถดูได้จากตารางที่ 4.2 บรรทัดที่ 2 และ 3 จะเห็นได้ชัดเจนว่า Subnet Mask มีขนาดเท่ากับ /32 เท่ากัน แต่ Router_ID ไม่เท่ากันดังนั้น Router_ID หมายเลข 1 จะมีน้ำหนักมากกว่า Router_ID หมายเลข 2

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

จากสมมติฐานข้างต้นที่กล่าวว่าหากมีการตั้งกฎของไฟร์วอลล์ในระหว่างทางซึ่งทำการทิ้งแพ็คเก็ตในเงื่อนไขที่ตรงกัน แต่นโยบายการค้นหาเส้นทางจำเป็นต้องวิ่งผ่านเส้นทางนั้น จึงจำเป็นต้องมีการค้นหาเส้นทางโดยนำกฎของไฟร์วอลล์มาเป็นหนึ่งในตัวเลือกในการค้นหาเส้นทางนั้น จึงมีแนวคิดในการค้นหาเส้นทางโดยใช้กฎของไฟร์วอลล์มาเป็นหนึ่งในตัวเลือกในการค้นหาเส้นทางนั้น จะเห็นได้อย่างชัดเจนว่าหากโทโพโลยีเป็นแบบบัส (Bus) ที่เป็นทางตรงนั้นแพ็คเก็ตไม่สามารถหลีกเลี่ยงไปยังเส้นทางอื่นได้เลย แต่หากเป็นโทโพโลยีแบบวงแหวน (Ring) หรือโทโพโลยีแบบตาข่าย (Mesh) นั้น จะมีเส้นทางในการส่งต่อแพ็คเก็ตได้มากขึ้น หากในโทโพโลยีนั้นๆ มีการปิดกั้นเส้นทางหมดทุกทางก็จะทำให้การค้นหาเส้นทางไม่ประสบผลสำเร็จ โดยผลประโยชน์จากงานวิจัยฉบับนี้ คือ สามารถค้นหาเส้นทางไปยังเส้นทางที่มีการ Pass ของกฎไฟร์วอลล์จึงทำให้การค้นหาเส้นทางนั้นเป็นไปได้อยากถูกต้องตามนโยบาย ส่วนแพ็คเก็ตใดที่มีการบล็อกระหว่างทางนั้น แพ็คเก็ตจะถูกทิ้งตั้งแต่ต้นทางเลย ซึ่งทำให้เราเตอร์ต้นทางไม่จำเป็นต้องส่งแพ็คเก็ตออกไปในกรณีที่รู้ว่าเส้นทางที่กำลังส่งไปนั้นถูกบล็อกโดยความถูกต้องของการค้นหาเส้นทางนั้นวัดที่ 1.) ค่า Prefix Matching ของ Subnet Mask และ 2.) น้ำหนัก (Weight) ของ Router_ID

ในการวิจัยนี้มีข้อจำกัดในบางส่วนคือ Rule Set ในส่วนแรกของเราเตอร์แต่ละตัวนั้นผู้ดูแลระบบเครือข่ายจำเป็นต้องติดตั้งคำสั่งได้อย่างถูกต้องและเรียงลำดับตามกฎของ IP Addressing หากมีการติดตั้งกฎของไฟร์วอลล์ที่ผิดตั้งแต่เริ่มแรก จะทำให้ผลลัพธ์ของการรวมกฎและเรียงลำดับผิดพลาดได้

5.2 ข้อเสนอแนะ

จากผลสรุปที่กล่าวไว้ว่าข้อจำกัดในบางส่วนคือ Rule Set ในส่วนแรกของเราเตอร์แต่ละตัวนั้นผู้ดูแลระบบเครือข่ายจำเป็นต้องติดตั้งคำสั่งได้อย่างถูกต้องและเรียงลำดับตามกฎของ IP Addressing หากมีการติดตั้งกฎของไฟร์วอลล์ที่ผิดตั้งแต่เริ่มแรก จะทำให้ผลลัพธ์ของการรวมกฎและเรียงลำดับผิดพลาดได้ ดังนั้นงานวิจัยฉบับต่อไปควรจะสามารถตรวจสอบความถูกต้องของการเรียงลำดับกฎของไฟร์วอลล์ได้ และจากงานวิจัยที่เกี่ยวข้องในบทที่ 2 ซึ่งงานวิจัยส่วนใหญ่จะมุ่งประเด็นไปที่ประสิทธิภาพที่จำนวนกฎของไฟร์วอลล์มากมายแปรผันกับการประมวลผลของ CPU และแบนด์วิดท์ที่สิ้นเปลืองเนื่องจากการบล็อกของแพ็คเก็ตระหว่างทาง ซึ่งในงานวิจัยฉบับ

นี้จะเน้นเรื่องความถูกต้องเป็นหลัก สำหรับงานวิจัยในอนาคตจะจัดทำให้มีการเปรียบเทียบประสิทธิภาพในการทำงานของการค้นหาโดยใช้การรวมกฎของไฟร์วอลล์ต่อไป และทำการค้นหาเส้นทางเพิ่มเติมโดยดูเงื่อนไขความซับซ้อนของ Port Routing ที่เป็น Subset และ Intersect ซึ่งกันและกัน โดยจะประเมินการเพิ่มกฎของไฟร์วอลล์เพื่อค้นหาเส้นทางแบบ Port Routing ที่ดีที่สุด

ในงานวิจัยฉบับนี้ยังมีประเด็นอื่นๆ ที่น่าสนใจอีกคือการทดลองอยู่ใน Interior ไม่ได้อยู่บน Exterior โพรโทคอล หากนำแนวคิดนี้ประยุกต์ใช้กับโพรโทคอล BGP ซึ่งเป็น Exteroir Routing Protocol ที่นิยมใช้ในการค้นหาเส้นทางระหว่าง ISP ก็สามารถใช้หลักการจากวิทยานิพนธ์ฉบับนี้ได้ ซึ่งจะเกิดผลดีกับการค้นหาเส้นทางขนาดใหญ่ที่มีจำนวนแพ็คเก็ตมหาศาล โดยมีแนวคิด ดังนี้

5.2.1 ทำการรวมกฎของไฟร์วอลล์ระหว่างเราเตอร์ BPG ดังทฤษฎีในงานวิจัยที่ได้ตีพิมพ์ของ K. Tantipongsakul and A. Khunkitti, เอกสารตีพิมพ์ชื่อ "Dynamic Policy-Based Routing using Firewall Rules" [5]

5.2.2 การส่งผ่านไฟร์วอลล์ของงานวิจัยฉบับนี้จะส่งผ่าน SSH แต่ถ้าหากมีการส่งผ่าน BGP ก็สามารถส่งตัวแปรของกฎไฟร์วอลล์ไปใน Header ของ BGP ได้เลย เนื่องจาก BGP นั้นจะทำการ Advertise กันทุกๆ 60 วินาที

5.2.3 หลังจากนั้นก็ถอดกฎของไฟร์วอลล์บน Header ของโพรโทคอล BGP เพื่อทำการรวมกฎของไฟร์วอลล์ โดยหลักการนี้จะใช้การส่ง Message บน BGP อยู่แล้ว

5.2.4 กระจายและทำการเปลี่ยนแปลง Routing Table ของ BGP ใหม่ภายใต้เงื่อนไขของกฎไฟร์วอลล์

ซึ่งโพรโทคอล BGP สามารถทำการค้นหาเส้นทางเชิงนโยบาย (Policy-Based Routing) ได้จึงมีข้อเสนอแนะในการนำแนวคิดนี้ขึ้นไปใช้ในการค้นหาเส้นทาง Exterior ที่มีจำนวนแพ็คเก็ตมหาศาล โดยจำนวนแพ็คเก็ตมหาศาลนี้สามารถถูกลดการส่งออกไปได้มากมายหากเส้นทางที่ส่งต่อนั้นเกิดการ Block และทำให้การค้นหาเส้นทางที่ถูกต้องตามนโยบายมีประสิทธิภาพมากยิ่งขึ้น

บรรณานุกรม

[1] B. R. Smith., B. and R.Garcia-Luna-Aceves. , "Efficient Policy-Based Routing without Virtual Circuits", In: The First International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks 2004.

[2] E. Quoitin. and Uhlig. S., "Modeling the routing of an Autonomous System with C-BGP, IEEE Network", Vol 19(6), November 2005.

[3] E.L. Lawler, "Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints," Ann. Discrete Maths, Vol 2, pp75-90, 1978.

[4] Grout, and V. McGinn, "Optimisation of Policy-Based Internet Routing using Access Control Lists", Proceedings of IFIP/IEEE International Symposium on Integrated Network Management, Nice, France, May 2005.

[5] K. Tantipongsakul and A. Khunkitti, "Dynamic Policy-Based Routing using Firewall Rules", Published by the IEEE Computer Society, Proceedings of UKSim 3rd European Symposium on Computer Modelling and Simulation, 25-27 November, 2009, Athens, Greece.

[6] M. Bukhatwa, and A. Patel, "Effects of Ordered Access Lists in Firewalls", Proceedings of IADIS WWW/Internet 2003, Algarve Portugal, 5th-8th November, pp257-264.

[7] Priyadarsi Nanda and Andrew James Simmonds. ,"Policy Based QoS Support Using BGP Routing"

[8] S. Pozo, R. Ceballos, R.M. Gasca. ,"A Heuristic Polynormal Algorithm for Local Inconsistency Diagnosis in Firewall Rule Sets", Journal of Networks, Vol. 4, No. 8, October 2009.

[9] T. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," Computer Commun. Rev., vol. 28, no. 4, pp. 203-214, October 1998.

[10] Y. Rekhter., T. Li. , T. Hares., "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)", http://www.faqs.org/rfcs/rfc4271.html.

ภาคผนวก ก.

Source Code ของงานวิจัย

ไฟล์ merge.c

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <arpa/inet.h>

struct node{
   int data;
   struct node *next;
}datanode;

struct rule{
        int source_mask;
        struct rule *next;
}rulenode;

int main(void){

        FILE *fpt, *fpt_w;
        char line[MAX_LEN];
        char *p, *p_buf, *tokens[MAXTOKENS];
        char *read_file ;
        char *last,*buf;
        int argc;
        int i, j, line_number = 0;
```

```
struct rule *first_node;
struct rule *last_node;
struct rule *nodetmp;
struct rule *search_node;
struct rule *current_node;
struct rule *before_node;
int status_insert = -1;
int search_tmp_integer, search_tmp_integer2;
int buf_source_mask;

char *buf_action, *buf_interface, *buf_proto, *buf_source_addr, *buf_dest_addr,
*buf_port, *buf_router_name;

first_node = (struct rule *) calloc(sizeof(rulenode),1);
last_node=first_node;

        last_node->source_mask = 99999;

/*
printf("\n ========== BEFORE ==========");

nodetmp=first_node;
while (nodetmp!=NULL) {

        printf ("\nsource_mask=%d",nodetmp->source_mask);
        printf("\n\n");
        nodetmp=nodetmp->next;
}

if ((fpt_w=fopen("mergerule.conf","w"))==NULL)
        printf("cannot open %s  for reading\n", "mergerule.conf");
```

```
for(i=0;i<3;i++){
        if(i==0){ read_file = "r2.conf";}
        if(i==1){       read_file = "r3.conf";}
        if(i==2){       read_file = "r4.conf";}

        if ((fpt=fopen(read_file,"r"))==NULL)
                printf("cannot open %s  for reading\n", read_file);

        line_number = 0;
        while (fgets(line,sizeof(line),fpt)){
                line_number++;
                fputs(line,fpt_w);
                argc = 0;
                for ((p = strtok_r(line, " \t\r\n", &last)); p; (p = strtok_r(NULL, "
\t\r\n", &last)), argc++) {
                        if (argc < MAXTOKENS - 1)
                                tokens[argc] = p;
                }
                tokens[argc] = NULL;
                printf("\n");

                buf_action = tokens[0];
                buf_interface = tokens[3];
                buf_proto = tokens[5];
                buf_source_addr = tokens[7];
                strcpy(buf,tokens[7]);
                if ((p_buf = strrchr(buf, '/')) != NULL){
                        search_tmp_integer = strtol(p_buf+1, NULL, 10);

                }else{
                        search_tmp_integer = -1;
                }
                buf_source_mask = search_tmp_integer;
```

```
buf_dest_addr = tokens[9];
buf_port = tokens[11];
buf_router_name = tokens[12];


search_node=(struct rule *) calloc(sizeof(rulenode),1);

search_node->source_mask = search_tmp_integer;
search_node->next = NULL;
int status_insert = -1;
current_node=first_node;
while (current_node!=NULL) {
if(search_node->source_mask >= current_node->source_mask){
if(search_node->source_mask == current_node->source_mask){
status_insert = 0;
break;
}
if(current_node == first_node){
search_node->next = current_node;
first_node = search_node;
}else{
before_node->next = search_node;
search_node->next = current_node;
}
status_insert = 0;
break;
}
before_node = current_node;
current_node = current_node->next;
}
if(status_insert < 0){
```

```
                        before_node->next = search_node;
                    }


            }
            fputs("\n",fpt_w);
            fclose(fpt);

    }

    fclose(fpt_w);
    free(nodetmp);
    nodetmp=first_node->next;

    if ((fpt_w=fopen("sort_mergerule.conf","w"))==NULL)
            printf("cannot open %s  for reading\n", "sort_mergerule.conf");

    while (nodetmp!=NULL) {

            if ((fpt=fopen("mergerule.conf","r"))==NULL)
                    printf("cannot open %s  for reading\n", "mergerule.conf");

            while (fgets(line,sizeof(line),fpt)){
                    line_number++;
                    argc = 0;
                    for ((p = strtok_r(line, " \t\r\n", &last)); p; (p = strtok_r(NULL, "
\t\r\n", &last)), argc++) {
                            if (argc < MAXTOKENS - 1)
                                    tokens[argc] = p;
                    }
                    tokens[argc] = NULL;
                    strcpy(buf,tokens[7]);
```

```
        if ((p_buf = strrchr(buf, '/')) != NULL){
                search_tmp_integer = strtol(p_buf+1, NULL, 10);

        }else{
                search_tmp_integer = -1;
        }
        if(search_tmp_integer == nodetmp->source_mask){

                fputs("pass out on ",fpt_w);
                fputs("$internal ",fpt_w);
                fputs("route-to ",fpt_w);
                fputs("(",fpt_w);
                fputs(tokens[3],fpt_w);
                fputs(",",fpt_w);
                fputs(tokens[12]+1,fpt_w);
                fputs(") ",fpt_w);
                fputs("proto ",fpt_w);
                if(strcmp(tokens[0],"block")==0){ fputs("!",fpt_w);}
                fputs(tokens[5],fpt_w);
                fputs(" from ",fpt_w);
                fputs(tokens[7],fpt_w);
                fputs(" to ",fpt_w);
                fputs(tokens[9],fpt_w);
                fputs(" port ",fpt_w);
                if(strcmp(tokens[0],"block")==0){ fputs("!",fpt_w);}
                fputs(tokens[11],fpt_w);
                fputs("\n",fpt_w);
        }

        printf("\n");
}
```



```
printf ("\ndest_addr=%s",nodetmp->dest_addr);
```

```
            printf ("\nport=%s",nodetmp->port);
            before_node = nodetmp;
            nodetmp=nodetmp->next;
        }
        fclose(fpt);
        fclose(fpt_w);
}
```

ตัวอย่างไฟล์ merg_tmp.conf

```
block in on $interface1 proto ICMP from 161.246.1.0/24 to any port {23,8080} #$r2
pass in on $interface1 proto ICMP from 161.246.0.0/16 to 202.28.68.0/24 port {23,80,443}
#$r2
pass in on $interface1 proto {TCP,ICMP} from 161.246.1.1/32 to any port {23,80,443} #$r3
block in on $interface1 proto ICMP from any to any port {11,22,33,44} #$r3
pass in on $interface1 proto ICMP from 161.246.1.0/24 to any port {1024>=} #$r4
pass in on $interface1 proto ICMP from any to any port {1024>=} #$r4
```

ตัวอย่างไฟล์ sort_mergerule.conf

```
pass out on $internal route-to ($interface1,$r3) proto {TCP,ICMP} from 161.246.1.1/32 to any
port {23,80,443}
pass out on $internal route-to ($interface1,$r2) proto !ICMP from 161.246.1.0/24 to any port
!{23,8080}
pass out on $internal route-to ($interface1,$r4) proto ICMP from 161.246.1.0/24 to any port
{1024>=}
pass out on $internal route-to ($interface1,$r2) proto ICMP from 161.246.0.0/16 to
202.28.68.0/24 port {23,80,443}
pass out on $internal route-to ($interface1,$r3) proto !ICMP from any to any port
!{11,22,33,44}
pass out on $internal route-to ($interface1,$r4) proto ICMP from any to any port {1024>=}
```

ตัวอย่าง ไฟล์ r2.conf

block in on $interface1 proto ICMP from 161.246.1.0/24 to any port {23,8080} #$r2

pass in on $interface1 proto ICMP from 161.246.0.0/16 to 202.28.68.0/24 port {23,80,443} #$r2

ตัวอย่างไฟล์ r3.conf

pass in on $interface1 proto {TCP,ICMP} from 161.246.1.1/32 to any port {23,80,443} #$r3

block in on $interface1 proto ICMP from any to any port {11,22,33,44} #$r3

ตัวอย่างไฟล์ r4.conf

pass in on $interface1 proto ICMP from 161.246.1.0/24 to any port {1024>=} #$r4

pass in on $interface1 proto ICMP from any to any port {1024>=} #$r4

ตัวอย่าง Shell Script ของการ Announce กฎของไฟล์วอลล์

#scp /root/.ssh/file root@192.168.2.254:/root/.ssh/sort_mergerule.conf

#scp /root/.ssh/file root@192.168.3.254:/root/.ssh/sort_mergerule.conf

#scp /root/.ssh/file root@192.168.4.254:/root/.ssh/sort_mergerule.conf

ตัวอย่าง Shell Script ของการรวมกฎใหม่ เข้ากับกฎเดิม

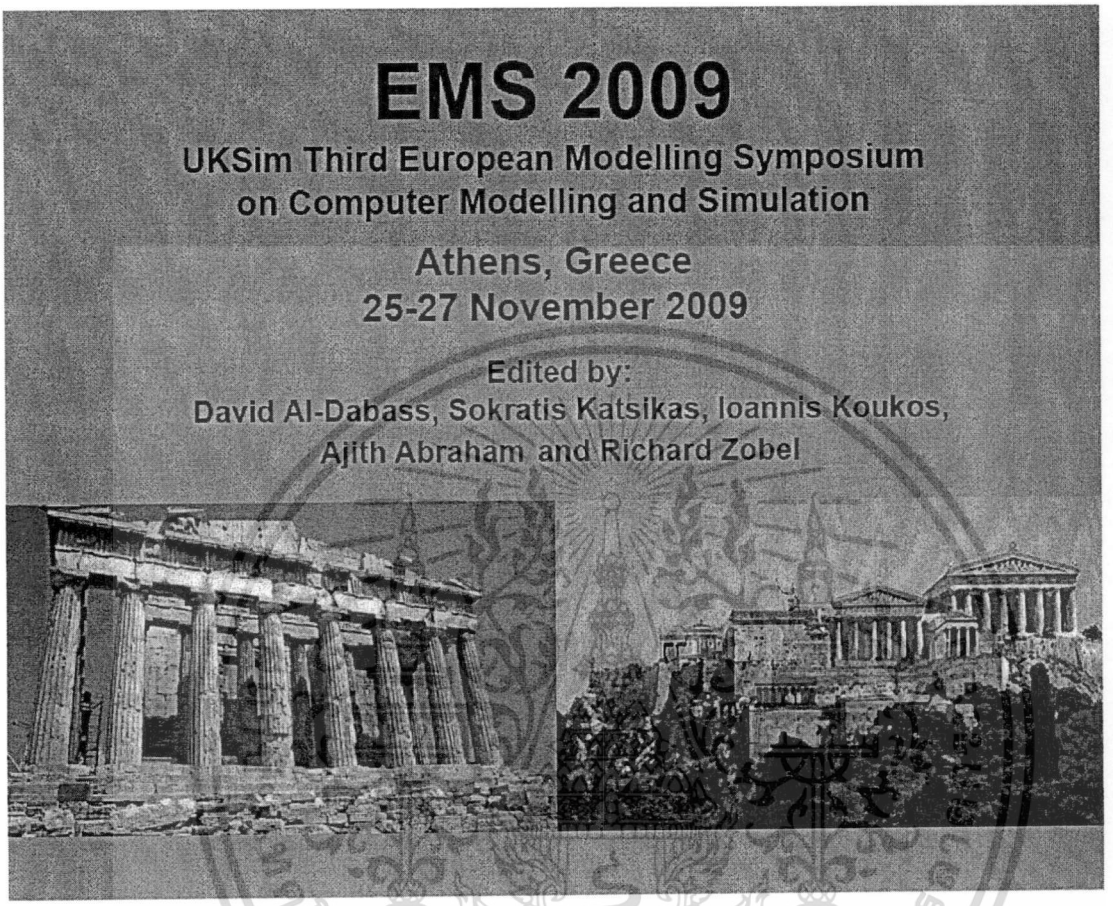
cat /etc/sort_mergerule.conf >> /etc/pf.conf

ภาคผนวก ข.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

1. K. Tantipongsakul and A. Khunkitti, **"Dynamic Policy-Based Routing using Firewall Rules"**, Published by the IEEE Computer Society, Proceedings of UKSim 3rd European Symposium on Computer Modelling and Simulation, 25-27 November, 2009, Athens, Greece.

การประชุมวิชาการและผลงานที่ได้ตีพิมพ์



2009 Third UKSim European Symposium on Computer Modeling and Simulation

Dynamic Policy-Based Routing using Firewall Rules

Kavin Tantipongsakul
Information Science, Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
E-mail: s8066435@kmitl.ac.th

Akharin Khunkitti
Information Science, Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
E-mail: akharin@it.kmitl.ac.th

***Abstract*— A firewall is basically a defense device to control incoming and outgoing network traffic. Policies and rules are important components of a firewall that allow packets to pass through by network administration. However, administrators do not know what the destination policies and rules are, and bandwidth control and management are not efficient. A concept occurred that firewalls can transfer and learn access control lists via Border Gateway Protocol (BGP). In this paper we propose transferring firewall rules using routing protocol. This will decide and select the best path for routers over transit autonomous systems. We also propose to develop routing protocol for reducing costs, for load sharing, and to utilize and optimize problems encountered in the implementation of traffic policies on network routers.**

***Keywords* - Border gateway protocol (BGP); Policy-Based Routing (PBR); Routing Information Base (RIB); Extensible RIB (XRIB); NLRI, Autonomous System (AS); Traffic Engineering, Firewall Rules; Access Control List (ACL).**

I. INTRODUCTION

A. Policy Based-Routing (PBR)

At present high performance internet networks within organizations need the freedom to implement packet forwarding and routing according to their own defined policies in a way that goes beyond traditional routing protocol concerns. Policy Based-Routing (PBR) enables the network administrator to define a routing policy other than basic destination-based routing, using the routing table. In effect, it enables the policy to override routing protocol decisions. Policy Based-Routing (PBR) includes a mechanism for selectively applying policies based on access list, packet size, or other criteria. The benefits of Policy Based-Routing (PBR) include:

- Source-Based Transit Provider Selection— Traffic originating from different sets of users through different internet connections across the policy routers.
- Quality of Service (QOS) — Differentiate traffic by setting the precedence or Type Of Service (TOS) values in the IP packet headers at the periphery of the network. Leveraging queuing mechanisms to prioritize traffic in the core or backbone of the network.
- Cost Savings—Organizations can achieve cost savings by distributing interactive and batch traffic among low-bandwidth, low-cost permanent paths and high-bandwidth, high-cost and switched paths.
- Load Sharing— Distributed traffic among multiple paths, based on the traffic characteristics.

B. Border Gateway Protocol (BGP)

BGP uses the Transmission Control Protocol (TCP) Port 179. It sends updated router table information, but only when one host has detected an exchange of routing information between gateway hosts (each with its own router) in a network of autonomous systems. BGP is often the protocol used between gateway hosts on the Autonomous System (AS). The routing table contains a list of known routers, the addresses they can reach, and a cost metric associated with the path to each router however the best available route is chosen. Each BGP router contains a Routing Information Base (RIB). The RIB contains three types of information:

- Adj-RIBs-In. The unedited routing information sent by neighboring routers.
- Loc-RIB. The actual routing information the router uses, developed from Adj-RIBs-In.
- Adj-RIBs-Out. The information the router chooses to send to neighboring routers.

C. Firewall Rules

A firewall is a secured and trusted machine that is designed to block unauthorized access while permitting authorized communications. It is a device, or set of devices, configured to permit or deny network traffic. Rules can be created for either inbound traffic or outbound traffic. The rules can be configured to specify the computers IP addresses, services, ports and protocols. It can specify which type of network adapter the rule will be applied to, such as local area network, wireless, remote access, i.e., a virtual private network connection.

Firewalls acting as packet filters are not visible as protocol end points. The firewall examines each packet and then passes the packet through to the other side unchanged, or drops the packet entirely, or handles the packet itself in some other way. See "Fig 1." below.

978-0-7695-3886-0/09 $26.00 © 2009 IEEE
DOI 10.1109/EMS.2009.33

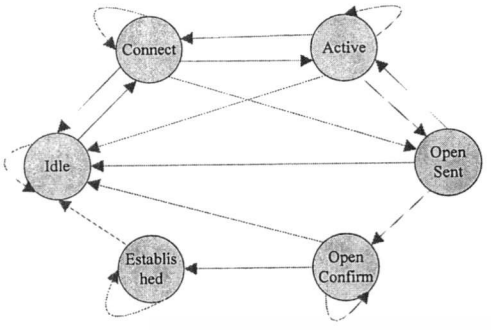


Figure 3. BGP Finite State Machine.

1) Idle: When a BGP speaking router is waiting a session it sits in the Idle state. It will not start a Idle session until a start Idle event occurs. Whenever a BGP session is shutting down because of some error, it returns to the Idle state. NOTIFICATION messages used to signal connection errors return the router to this state.

2) Connect: Once the BGP software and it's environment have been initialized, an OPEN message is sent. The router has attempted to open a TCP connection between itself and another peer.

3) Active: The router has started the first phase of initializing a TCP three-way handshake to peer. If a router fails to establish, it will drop back to Idle.

4) Open Sent: Once BGP has performed all the setup steps necessary, it sends out a TCP SYN on port 170.

5) Open Confirm: The router enters this state after the peer has sent back a TCP SYN packet indicating that the TCP session is being SYN.

6) Established: After each router has sent an ACKnowledge, one router has sent a SYN, and a TCP handshake has been completed, the router attempts to exchange BGP messages. If the router is in the OPEN CONFIRM state (the TCP Handshake is complete) and receives an UPDATE or KEEPALIVE message, the BGP session state changes to ESTABLISHED [5] "Fig. 3".

III. THE PROBLEMS AND THE COMPLEXITY OF POLICY-BASED ROUTING

In [7] a formal development is given of a *list L*, implementing a *policy Z* in *traffic T*. In case of two rules, i and j, are dependence then set $d_{ij}=1$; otherwise $d_{ij}=0$. The *latency*, $\lambda(r_i)$, of a rule r_i to be the time taken to independently process r_i, the *cumulative latency*, $\kappa(r_i(L))$, of r_i at position i in a *list L*, is the time taken to process r_i and all rules preceding it in L.

$$\kappa(r_i(L)) = \sum_{\varphi=1}^{i} \lambda(r_\varphi(L))$$

The *expected latency*, $E(L,T)$, of a *list L*, in *traffic T*, is then given by

$$E(L,T) = \sum_{i=1}^{n} h(r_i(L),T)\kappa(r_i(L)) = \sum_{i=1}^{n} h(r_i(L),T)\sum_{\varphi=1}^{i} \lambda(r_i(L))$$

For a given traffic flow, T, we require to find (or approximate) the list, L, implementing a policy, Z, that minimises $E(L,T)$. The following theorem is given.

THEOREM: SEQUENCING TO MINIMISE EXPECTED LATENCY (SMEL) is NP-complete.

PROOF: Transformation to SEQUENCING TO MINIMIZE WEIGHTED COMPLETION TIME (SMWCT)

A direct mapping from SMEL to SMWCT is achieved by setting [7].

SMEL		SMWCT
Z	to	N
r	to	t
D	to	$\preccurlyeq$ by taking $t_i \preccurlyeq t_j \Leftrightarrow (i<j) \wedge d_{ij}=1$
$\lambda(r)$	to	$l(t)$
$h(r)$	to	$w(t)$

It follows that (unless $P=NP$) guaranteed exact solutions are not reasonably to be expected for large values of n.

When the connection has established and transferred data between BGP routers then firewall will screen all packet incoming to interface. If firewall rules have blocked Port Number or Protocol.

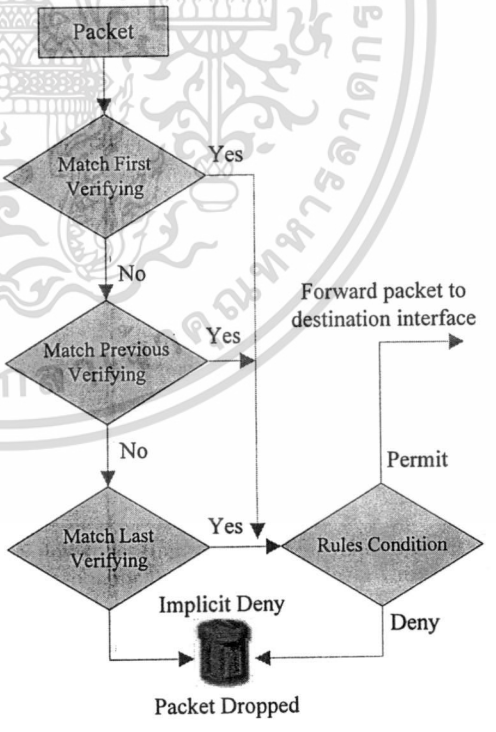


Figure 4. ACL Implicit Deny.

Routing table will route the packet to destination address. However router cannot know the destination firewall rules therefore unacceptable packets have dropped by rules. As "Fig. 4" Internet bandwidth was wasted by firewall rules if routers can learn and exchange firewall rules together and then select the best path of routing for utilizing internet bandwidth.

IV. TRANSFERRING FIREWALL RULES VIA BORDER GATEWAY PROTOCOL

A. Implementation BGP Message Header

BGP messages are sent over TCP connections. A message is processed only after it is entirely received. The maximum message size is 4096 octets. All implementations are required to support this maximum message size. The smallest message that may be sent consists of a BGP header "Fig. 5" without a data portion (19 octets).

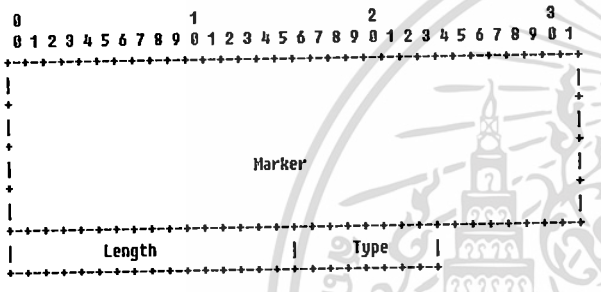


Figure 5. BGP Header.

All BGP messages [1] have the same fixed-size header, which contains a marker field indicating the total length of the message and a type field indicating the message type. Value 1=OPEN, 2=UPDATE, 3=NOTIFICATION, 4=KEEPALIVE. This paper will focus to implement the UPDATE Message "Fig. 6" type. BGP systems send update messages to exchange network reachability information. BGP systems use this information to construct a graph that describes the relationships among all known ASs. Update messages consist of the BGP header plus the following optional fields.

In Path attributes and NLRI are able to add data and variable of IP address, Prefix, port, Rules Condition (Permit and Deny) for exchanging data between AS neighbors then create dynamic routing to change routing path in routing table. In this paper call XRIB (Extensible Routing Information Base).

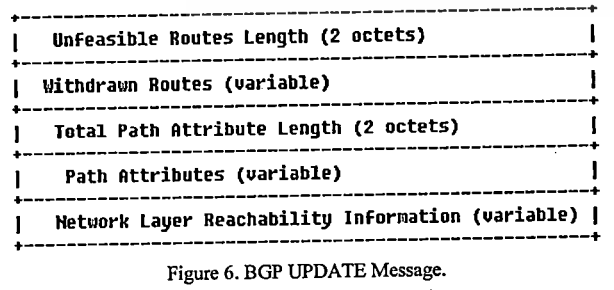


Figure 6. BGP UPDATE Message.

- Unfeasible routes length—Length of the field that lists the routes being withdrawn from service because they are no longer deemed reachable.
- Withdrawn routes—IP address prefixes for the routes being withdrawn from service "Fig. 7".

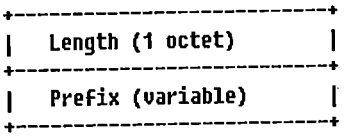


Figure 7. Withdrawn routes Message.

  - Length—The Length field indicates the length in bits of the IP address prefix. A length of zero indicates a prefix that matches all IP addresses (with prefix, itself, of zero octets).
  - Prefix—The Prefix field contains an IP address prefix, followed by the minimum number of trailing bits needed to make the end of the field fall on an octet boundary. Note that the value of trailing bits is irrelevant.
- Total path attribute length—Length of the field that lists the path attributes for a feasible route to a destination.
- Path attributes—Properties of the routes, including the path origin, the multiple exit discriminators (MED), the originating system's preference for the route, and information about aggregation, communities, confederations, and route reflection.
- Network layer reachability information (NLRI)—IP address prefixes of feasible routes being advertised in the update message.

B. Merging Firewall Rules (Access Control Lists)

ACLs are components of firewall and router that used for a variety of purposes in applying security and other policies on routers [9]. An ACL is an ordered sequence of rules, each rule matching to permit or deny any packet that incoming or outgoing. A typical rule, using Cisco IOS command [11] might be:

access-list 1 deny 202.28.68.5 0.0.0.255

access-list 1 deny 203.177.44.22 0.0.0.255

access-list 1 permit any

This example is a standard IP access list that denies the hosts 202.28.68.5/24 and 203.177.44.22/24 while permitting all other traffic. The list is applied sequentially from the top down as the router checks the packets arriving at the interface where this access list is applied, in order to check if the packets match the permit and deny rule statements.

Then parameters of firewall rules will announce ACLs via BGP. Rules announcement sends an UPDATE message to its BGP peer, the message may include variable of rules. Merging Algorithms process the list of capabilities Rules in the capabilities Path Attribute Parameter carried by the UPDATE message, XRIB (Extensible Routing Information Base) will advertise to every peer and applying updated routing table that called Dynamic Policy-Based Routing "Fig. 8".

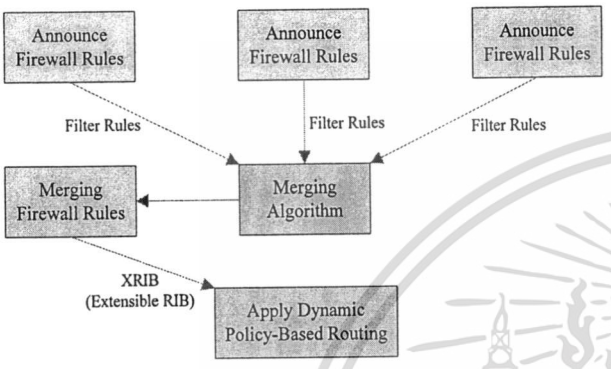


Figure 8. Merging Firewall Rules.

C. *Apply Dynamic Programming for Policy-Based Routing*

A dynamic programming technique is coded by Held and Karp (1962) and adapted in various forms to the present day (Lawler et al., 1985 and Gutin and Punnen, 2002) [8]. The generic algorithm has time complexity O(2n), space complexity O(2n) and can be adapted for SMEL as follows.

$Z = \{r_1, r_2, \ldots, r_n\}$ *(with r_n fixed)*

For $Y = \{r_1, r_2, \ldots, r_{n-1}\}$ and $r \in Y$, let $|SMEL|(Y,r)$ be the minimum expected latency of the sublist $Y \cup \{r_n\}$, *then*

$$|SMEL|(Y,r) = \{\min_{r \neq s \neq y} |SMEL|(Y-\{r\},s) + \kappa(s(Y)) \quad \kappa(r(Y)) \quad Y=\{r\} \quad Y \neq \{r\}$$

SMEL can then be calculated as $min_r\ |SMEL|(Z,r) + \kappa(r(Z))$

Because of BGP Headers contain the access control lists variable then BGP Headers are bigger than normally even though an improvement on exhaustive search, the time complexity is still exponential. However this method, on more powerful processors, provides good benchmarks for comparison.

D. *Path Selection by Access Control Lists*

After the initial exchange incremental updates by merging firewall rules are sent as the routing tables change. BGP maintains updated routing table then BGP will establish, distribute, advertise and calculate every ASs the select the updated dynamic path by using firewall rules [6].

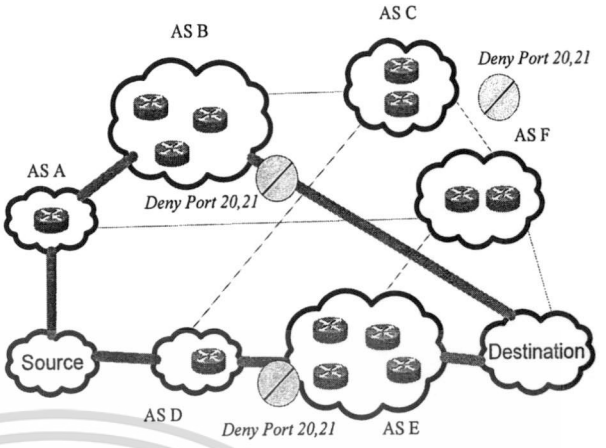


Figure 9. Common BGP Path Selection.

BGP assigns the first valid path as the current best path. BGP then compares the best path with the next path in the list, until BGP reaches the end of the list of valid paths. This policy routing provides by network administrator such as "Fig. 9".

Source > AS A > AS B > Destination or

Source > AS D > AS E > Destination

While BGP connections establish both of above path if packets are FTP port 20, 21 like the "Fig. 9". When FTP packets route to AS B or AS E. FTP packets have dropped by ACLs. In order that cost of internet bandwidth will increase and cause of bandwidth consumed.

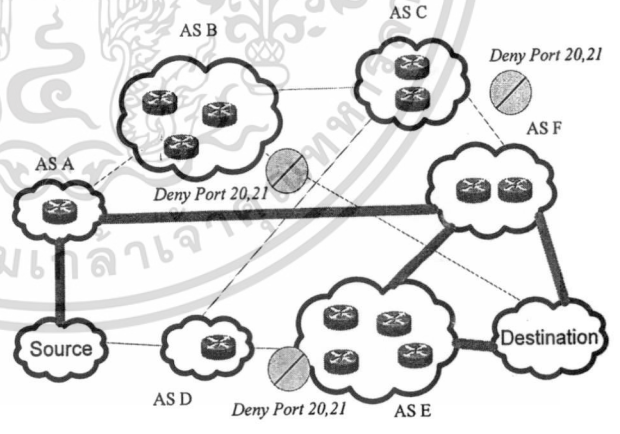


Figure 10. Path Selection by Firewall Rules

Consider the case in common BGP path from "Fig. 9", and path selection by firewall rules "Fig. 10". When comparing all the paths in route selection. Applied dynamic routing. Routing table is avoiding switching the Source to Destination. These optimizations choose firewall rules attribute to change path selection "Fig. 10".

Source > AS A > AS F > Destination or

Source > AS A > AS F > AS E > Destination

Recent "Fig. 9" and "Fig. 10" of BGP have shown a important growth in the size and dynamics routing of BGP tables. This is leading to concerns about the impact these trends in BGP have on the Internet. This paper focus on this issue for a large ISP. Large ISP networks are designed and optimized on the basis of metrics such as latency and dropped of packet switching when denying packet has dropped by firewall rule. The need to provision the network for future growth and changes in traffics. This method is typically based on calculating a traffics matrix to decide traffics engineering demands for different path of the network.

V. CONCLUSIONS

This paper approached the transferring of firewall rules via BGP and dynamic policy-based routing algorithms. They efficiently compute exchanging and classifying firewall rules, via BGP header, in the context of traffic and performance constraints. The router is able to learn and transfer access control lists that satisfy network engineer requirements. In addition, it chooses the best path between ASs for useful internet bandwidth, traffic engineer policies, and optimizes performance of internet bandwidth. Future work in this area of research will be to evaluate, optimize and compare each BGP attributes between firewall rule attributes of proposed scheme through a prototype using OpenBGPD and C-BGP. Finally, investigate further the issues covering fault management, dynamic policy-based routing, and security related to information sharing between ASs.

REFERENCES

[1] Y. Rekhter., T. Li. , T. Hares., "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)", http://www.faqs.org/rfcs/rfc4271.html.

[2] B. R. Smith., B. and R.Garcia-Luna-Aceves. , "Efficient Policy-Based Routing without Virtual Circuits", In: The First International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks 2004.

[3] T. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," Computer Commun. Rev., vol. 28, no. 4, pp. 203-214, October 1998.

[4] D. Pei., D. Massey and D.Zhang. , "Finite State Machines for BGP", UCLA CSD Technical Report.

[5] D. Brand., D.Zafiropulo., "On communicating finite-state machines", J. ACM, 30(2):323–342, 1983.

[6] Grout, and V. McGinn, "Optimisation of Policy-Based Internet Routing using Access Control Lists", Proceedings of IFIP/IEEE International Symposium on Integrated Network Management, Nice, France, May 2005.

[7] E. Quoitin. and Uhlig. S., "Modeling the routing of an Autonomous System with C-BGP, IEEE Network", Vol 19(6), November 2005 .

[8] M. Held, and M. Karp, "A Dynamic Programming Approach to Sequencing Problems", Journal of the Society of Industrial and Applied Mathematics (SIAM), Vol. 10, pp196-210.

[9] M. Bukhatwa, and A. Patel, "Effects of Ordered Access Lists in Firewalls", Proceedings of IADIS WWW/Internet 2003, Algarve Portugal, 5th-8th November, pp257-264.

[10] E.L. Lawler, "Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints," Ann. Discrete Maths, Vol 2, pp75-90, 1978.

[11] Cisco Systems., (2003) "ACL Optimizer and Hits Optimizer", USA, http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cw2000/fam_prod/acl_mgr/aclm_1_x/1_6/u_guide/ac1js.htm

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายกวิน ตันติพงศ์สกุล
วัน เดือน ปีเกิด	13 กรกฎาคม 2526
สถานที่เกิด	จังหวัดสงขลา
ประวัติการศึกษา	สารสนเทศศาสตรบัณฑิต สาขาวิชาระบบสารสนเทศเพื่อการจัดการ สำนักวิชาสารสนเทศศาสตร์ มหาวิทยาลัยวลัยลักษณ์
ประสบการณ์การทำงาน	2549 – ปัจจุบัน ตำแหน่งวิศวกรเครือข่ายและ ผู้เชี่ยวชาญงานโครงการ บริษัท จีเน็ท เน็ทเวิร์คโซลูชั่น จำกัด
ประกาศนียบัตร	- Microsoft Certified : MCSE, MCP - CompTIA Certified : Security+ - Cisco Certified : CCNA, CCDA - Juniper Certified : JNCIS-ER, JNCIA-EX, JNCIA-ES - Cyberoam Certified : CNSP