# การเพิ่มสมรรถนะของไฟร์วอลล์ โดยการจัดเรียงกฎใหม่

# Firewall Performance Enhancement by Rearranging Rules

สุชาติ คุ้มมะณี*

**บทคัดย่อ**

ไฟร์วอลล์มีบทบาทสำคัญเป็นอย่างมากบนระบบเครือข่าย เป็นเครื่องมือที่ช่วยในการปกป้องเครือข่ายขององค์กรจากการบุกรุกและการโจมตี ประสิทธิภาพโดยรวมของไฟร์วอลล์ขึ้นอยู่กับการตั้งกฎของผู้ดูแลระบบ และลำดับของกฎบนไฟร์วอลล์มีผลต่อประสิทธิภาพการทำงานบนไฟร์วอลล์ด้วยเช่นกัน หากกฎของไฟร์วอลล์ไม่ได้ถูกจัดเรียงอยู่ในลำดับที่เหมาะสมก็จะมีผลต่อประสิทธิภาพการทำงานบนไฟร์วอลล์ ในงานวิจัยนี้ได้สร้างสมการที่สามารถปรับลำดับกฎบนไฟร์วอลล์ เพื่อให้ไฟร์วอลล์ประมวลผลการทำงานเร็วขึ้น และพัฒนาโปรแกรมที่ช่วยวิเคราะห์และตรวจสอบประสิทธิภาพการทำงานบนไฟร์วอลล์ ซึ่งถ้ามีการปรับลำดับกฎตามสมการที่ผู้วิจัยสร้างขึ้น ทำให้ไฟร์วอลล์ใช้เวลาในการทำงานเร็วขึ้น

**คำสำคัญ:** ไฟร์วอลล์ กฎ การปรับลำดับของกฎ

**Abstract**

Firewall is a very significant Component on the network. It is a tool that helps to protect corporate networks from threats and attacks. The overall performance of the firewall rule set is based on the setting by administrator and changing the sequence of firewall rules affect the performance of the firewall. If the firewall rules are not arranged in proper order, they can affect the performance of the firewall. This project has created an equation to adjust the order of firewall rules and to rearranging the rules accordingly. So, the firewall can perform faster.

**Keyword:** Firewall, rules, rearranging rules.

## 1. บทนำ

ไฟร์วอลล์มีบทบาทสำคัญคือ มีหน้าที่ตรวจสอบและจำกัดสิทธ์การเข้าถึงระบบเครือข่าย ไฟร์วอลล์ทำงานโดยพิจารณาจากการเรียงกฎตามลำดับ เมื่อกลุ่มข้อมูล (Packet) ที่เข้ามาตกกระทบกับกฎที่จับคู่กันเหมาะสม (Match) ไฟร์วอลล์จะทำงานตามกฎนั้นทันที แต่ถ้าไม่สามารถจับคู่ได้ จะเลื่อนไปตรวจสอบในกฎต่อๆ ไปเรื่อยๆ ถ้าไม่ Match กับกฎใดเลย กลุ่มข้อมูลนั้นจะถูกโยนทิ้ง (Drop) ทันที มีงานวิจัยมากมายที่ต้องการเพิ่มประสิทธิภาพการประมวลผลของไฟร์วอลล์ให้เร็วขึ้น สามารถพิจารณาออกเป็น 2 กลุ่มใหญ่ๆ คือ ฮาร์ดแวร์ไฟร์วอลล์ [1-2] และปรับปรุงกฎที่สร้างขึ้นบนไฟร์วอลล์ [3-14] ในกลุ่มที่ 2 มีงานวิจัยที่เกี่ยวข้องกับการจัดเรียงกฎที่น่าสนใจคือ Ned Freed [15] ซึ่งทำการศึกษาพฤติกรรมของโปรแกรมประยุกต์ที่ทำงานในเครือข่ายและทำการปรับกฎตามพฤติกรรมดังกล่าว และ Weiping Wang [11] สนใจเรื่องเกี่ยวกับสถิติการใช้งานกลุ่มข้อมูลบนระบบเครือข่าย แต่ทั้งคู่เพียงแต่เสนอแนวความคิดไว้เท่านั้นยังไม่สามารถใช้งานได้จริง

งานวิจัยนี้ได้ทำการศึกษาแล้วพบว่าการปรับลำดับกฎบนไฟร์วอลล์มีผลต่อประสิทธิภาพการทำงานบนไฟร์วอลล์จริง ดังนั้นจึงพิจารณาออกแบบสมการที่สามารถปรับลำดับกฎบนไฟร์วอลล์ เรียกว่า FPERR (Firewall Performance Enhance-ment by Rearranging Rules) ทำให้ไฟร์วอลล์มีประสิทธิภาพในการประมวลผลสูงขึ้น และพัฒนาโปรแกรมที่สามารถทำงานได้กับไฟร์วอลล์จริง ตามสมการที่ได้ออกแบบไว้

* สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาการสารสนเทศ มหาวิทยาลัยมหาสารคาม

## 2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

### 2.1 การทำงานบนไฟร์วอลล์

ไฟร์วอลล์จะทำตามกลุ่มของกฎที่ได้ตั้งไว้ ซึ่งอาจจะเรียกกลุ่มของกฎนี้ว่านโยบาย (Policy) หรือรายการกฎ (Rule List) หรือแอคเซสคอนโทรลลิสต์ (Access Control List) ก็ได้ เมื่อไฟร์วอลล์ได้รับกลุ่มข้อมูล (Packet) จะเริ่มอ่านข้อมูลที่อยู่ตรงส่วนหัว (Header) ของกลุ่มข้อมูล เช่น หมายเลขที่อยู่ต้นทาง (Source IP Address) หมายเลขที่อยู่ปลายทาง (Destination IP Address) และหมายเลขพอร์ต (Port) เป็นต้น จากนั้นไฟร์วอลล์จะนำข้อมูลดังกล่าวมาตรวจสอบเทียบกับรายการกฎ โดยการเทียบกับกฎ (Rule) ที่อยู่ลำดับบนสุดก่อน ซึ่งก็คือ กฎ ลำดับที่ 1 ว่าตรงกันกับกลุ่มข้อมูลนั้นหรือไม่ ถ้าตรงกันก็จะทำตามข้อที่กำหนดในส่วนของ Action ในกฎ ถ้า Action เป็น ACCEPT ก็จะอนุญาตให้กลุ่มข้อมูลให้เดินทางต่อไป แต่ถ้าเป็น DENY ก็จะไม่อนุญาตและโยนกลุ่มข้อมูลนั้นทิ้งไป และถ้ากลุ่มข้อมูลไม่ตรงกับกฎลำดับที่ 1 ไฟร์วอลล์ก็จะทำการเทียบกับกฎลำดับถัดไป (กฎลำดับที่ 2) ทำอย่างนี้ไปเรื่อยๆ จนถึงกฎสุดท้าย ซึ่งมักจะได้รับการตั้งค่าให้ตรงกับทุกกลุ่มข้อมูล และกำหนด Action ให้เป็น DENY เสมอ เพื่อให้ไฟร์วอลล์หยุดและโยนกลุ่มข้อมูลดังกล่าวทิ้งไป

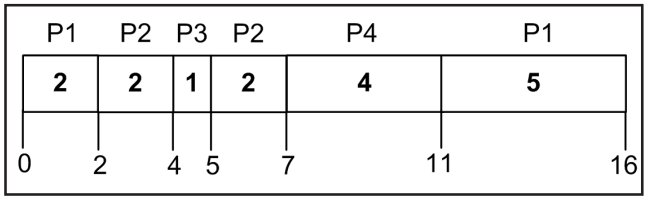
### 2.2 First-Come First-Served (FCFS) และ Shortest-Job-First (SJF) Scheduling

FCFS และ SJF [16] เป็นอัลกอริทึมที่ระบบปฏิบัติการ (Operating System) ใช้สำหรับตรวจสอบว่าโพรเซส (Process) ใดๆ ที่ต้องการประมวลผลในหน่วยประมวลผลกลาง (Central Processing Unit: CPU) ต้องใช้เวลารอคอยเพื่อจะประมวลผลสำหรับงานหนึ่งๆ (Job) ให้สำเร็จนั้นต้องใช้เวลาเท่าไหร่ ซึ่งอัลกอริทึมแบบ FCFS จะใช้เวลารอคอยมากที่สุด ส่วนอัลกอริทึมแบบ SJF จะใช้เวลารอคอยน้อยที่สุด โดยสามารถอธิบายการทำงานของอัลกอริทึมดังกล่าวได้ดังนี้

***ตารางที่ 1*** *แสดงตารางงานของโพรเซสที่ต้องการใช้ซีพียู*

| โพรเซส | เวลาที่มาถึง (นาที:วินาที) | เวลาที่ใช้ CPU (วินาที) |
|---|---|---|
| P1 | 0:0 | 7 |
| P2 | 0:2 | 4 |
| P3 | 0:4 | 1 |
| P4 | 0:5 | 4 |

จากตารางที่ 1 แสดงรายละเอียดของโพรเซสที่ต้องการใช้ซีพียูในการประมวลผล เช่น โพรเซส P1 เข้าสู่คิวอันดับที่ 1 เวลาที่มาถึงคือ 0:0 (เวลาที่เข้ามาถึงคิว) และต้องการใช้ซีพียูเพื่อจะทำงานให้เสร็จคือ 7 วินาที โพรเซส P2 มาถึงคิวเวลา 0:2 และใช้เวลาในการประมวลผล 4 วินาที โพรเซส P3 และ P4 โดยเรียงลำดับตามตาราง เมื่อนำมาเขียนเป็น Gantt Chat เพื่ออธิบายลำดับการใช้งานของ FCFS ได้ดังภาพที่ 1




***ภาพที่ 1*** *แสดงลำดับการเข้าคิวของโพรเซสแบบ FCFS*

เวลาในการรอคอย Average Waiting Time (*AVG*) หาได้จากสมการที่ 1

$$AVG = \frac{(WT_{P1} + WT_{P2} + WT_{P3} + WT_{Pn})}{n} \quad (1)$$

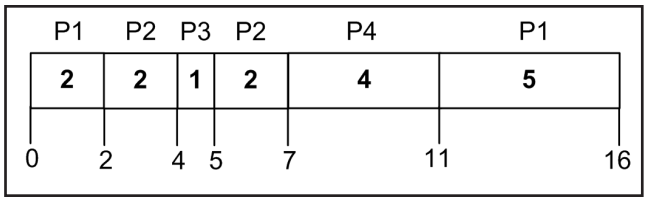
เมื่อ *AVG* คือ ผลรวมของระยะเวลารอคอย

$WT_{P1}...WT_{Pn}$ คือ ระยะเวลาการรอคอยเพื่อประมวลผลของแต่ละโพรเซส

*n* คือ จำนวนโพรเซสทั้งหมดที่ประมวลผล

จากภาพที่ 1 โพรเซส *P1* ใช้เวลารอคอยในการประมวลผล 0 วินาที เพราะว่าเข้าคิวเป็นลำดับที่ 1 โดยไม่ต้องรอโพรเซส *P2* รอคอย 5 วินาที (เวลาที่คอยให้โพรเซส *P1* ทำงานเสร็จ ลบด้วยเวลาที่มาถึงคิวเวลา 0:2 วินาที ซึ่งมีค่าเท่ากับ 7 – 2 = 5 วินาที) โพรเซสที่ 3 รอคอย 7 วินาที และโพรเซสที่ 4 รอคอย 7 วินาที เมื่อแทนค่าเวลารอคอยรวมของ FCFS ในสมการที่ 1 จะได้ผลลัพธ์คือ AVG = (0 + 5 + 7 + 7) / 4 = 4.75 วินาที

สำหรับวิธีการแบบ SJF จะให้ลำดับความสำคัญกับโพรเซสที่ใช้เวลาในการประมวลผลซีพียูน้อยที่สุดเข้ามาทำงานก่อน ซึ่งสามารถเขียนเป็น Gantt Chat ดังภาพที่ 2




***ภาพที่ 2*** *แสดงลำดับการเข้าคิวของโพรเซสแบบ SJF*

จากภาพที่ 2 เวลารอคอยของโพรเซส *P1* เท่ากับ 0 วินาที (ไม่ต้องรอคอย) และ 9 วินาที (โพรเซส *P1* ทำงานครั้งที่ 2 เมื่อ *P1* ทำงานมาถึงเวลาที่ 0:2 โพรเซส *P2* เข้ามาถึงคิว และใช้เวลาในการประมวลผลน้อยกว่า *P1* อยู่ 1 วินาที เนื่องจากเวลาดังกล่าว *P1* ทำงานไปแล้ว 2 วินาที ทำให้เวลาของ *P1* เหลือ 5 วินาที แต่ *P2* เข้ามาใหม่ใช้เวลาเพียง 4 วินาทีเท่านั้น) คำนวณได้จาก เวลาที่ *P1* ทำงานครั้งที่ 2 ลบด้วยเวลาที่ *P1* ทำงานเสร็จครั้งแรก เท่ากับ 11 – 2 = 9 วินาที โพรเซส *P2* ใช้เวลารอคอยเท่ากับ 0 วินาที เนื่องจาก *P2* มาถึงแล้วได้ทำงานทันที โพรเซส *P3* รอคอยเท่ากับ 0 วินาที และโพรเซส *P4* รอคอยเท่ากับ 2 วินาที เมื่อแทนค่าเวลารอคอยรวมของ SJF ในสมการที่ 1 จะได้ผลลัพธ์คือ

$$AVG = ((0 + 9) + 0 + 0 + 2) / 4 = 2.75 \text{ วินาที}$$

สรุปว่าถ้าระบบปฏิบัติการเลือกวิธีการประมวลผลแบบ SJF จะทำให้โพรเซสประมวลผลเร็วกว่าแบบ SCFS เท่ากับ 2 วินาที ซึ่งทฤษฎีที่ผู้วิจัยได้เสนอเพื่อปรับความเร็วของไฟร์วอลล์ (FPERR) พบว่าทำงานเหมือนกับ SJF คือ พิจารณาโพรเซสที่ใช้เวลาในการประมวลผลน้อยที่สุดและมีความถี่ในการประมวลผลมากที่สุด มาทำงานก่อนเสมอ ซึ่งจะส่งผลให้ไฟร์วอลล์ทำงานได้เร็วขึ้นจริง

## 3. ขั้นตอนการดำเนินงาน

### 3.1 ข้อกำหนดเบื้องต้นเกี่ยวกับการออกแบบ

ในงานวิจัยนี้ได้ทำการออกแบบการจัดเรียงกฎของไฟร์วอลล์ใหม่ ตามความถี่ที่ตกกระทบบนไฟร์วอลล์ แต่มีเงื่อนไขบางประการที่อาจจะส่งผลต่อวิธีการนี้ ดังนั้นงานวิจัยนี้จะตั้งอยู่บนพื้นฐานความเข้าใจดังนี้คือ

1) กฎที่ถูกจัดเรียงใหม่จะต้องปราศจากความขัดแย้งกัน โดยมีงานวิจัยที่แก้ปัญหาความขัดแย้งแล้วเช่น [3-4]

2) เวลาที่ใช้ประมวลผลในแต่ละเขตข้อมูลในกฎ (Field) เช่นไอพีต้นทาง ไอพีปลายทาง เป็นต้น มีระยะเวลาที่สั้นมาก มีหน่วยเป็นไมโครวินาทีหรือน้อยกว่านั้น ขึ้นอยู่กับสถาปัตยกรรม ของเครื่อง ฉะนั้นในการปรับกฎของงานวิจัยนี้จะไม่ปรับแบบทันทีทันใด (Real-Time)

3) กฎสุดท้ายของไฟร์วอลล์จะไม่ถูกนำมาประมวลผลเพราะเป็นกฎที่ไม่อนุญาตตลอดเวลา เมื่อนำมาคำนวณจะเป็นกฎที่ถูกตกกระทบมากที่สุด ซึ่งไม่ถูกต้อง

### 3.2 แนวความคิดในการออกแบบบการปรับกฎไฟร์วอลล์

จากภาพที่ 3 แสดงแนวคิดในการออกแบบวิธีการปรับกฎ ซึ่งสามารถอธิบายได้ดังต่อไปนี้

1) ปรับแต่งค่าไฟร์วอลล์ให้พร้อมทำงานจริง เช่น การติดตั้ง การกำหนดโซน การสร้างกฎโดยไม่ต้องคำนึงถึงลำดับก่อนหลังของกฎ เป็นต้น

2) เมื่อการเชื่อมต่อสมบูรณ์ กลุ่มข้อมูลเริ่มไหลเข้าสู่ไฟร์วอลล์ ข้อมูลมากหรือน้อยขึ้นอยู่กับปริมาณการใช้งาน

3) กลุ่มข้อมูลจะถูกประมวลผลบนไฟร์วอลล์ตามกฎที่ได้สร้างไว้ในขั้นตอนที่ 1 ซึ่งการตกกระทบกับกฎบนไฟร์วอลล์จะถูกบันทึกลงฐานข้อมูล หรือ Log file ยกเว้นกฎสุดท้ายในไฟร์วอลล์จะไม่ถูกบันทึก เนื่องจากกลุ่มข้อมูลส่วนใหญ่จะถูกโยนทิ้งเมื่อกระทบกับกฎสุดท้าย (ข้อมูลที่ถูกโยนทิ้งในกฎสุดท้ายนั้น เป็นข้อมูลที่ไม่พึงประสงค์ให้เข้าและออกในองค์กร)

4) ข้อมูลการตกกระทบจะถูกบันทึกลง Log file ไปเรื่อยๆ จนถึงระยะเวลาหนึ่งเรียกว่า จุดหยุดเพื่อพิจารณา (Break-Point) ก็จะเริ่มประมวลผลข้อมูลใน Log file ที่บันทึกไว้ (สำหรับค่าเวลาที่กำหนดใน Break-Point ผู้ดูแลระบบสามารถกำหนดได้ตามความเหมาะสม เช่น 1 วัน/สัปดาห์/เดือน/ปี หรือขึ้นกับปริมาณการใช้ข้อมูล)

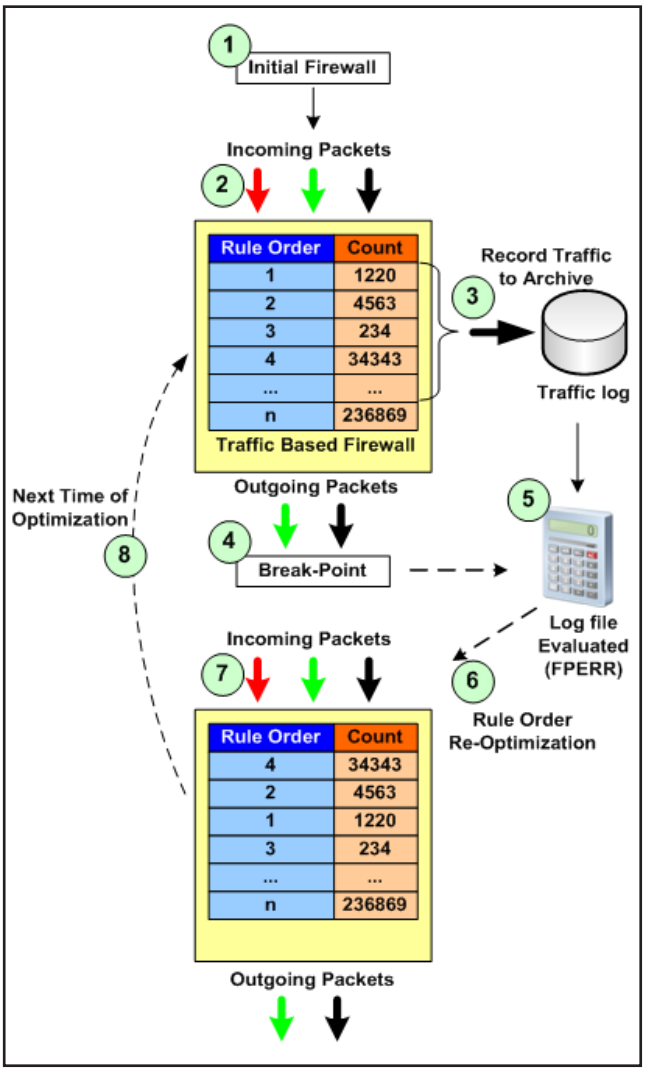
5) เป็นขั้นตอนในการประเมินผล Log file โดยใช้สมการ FPERR ซึ่งจะกล่าวในหัวข้อถัดไป ผลลัพธ์ที่ได้จากการคำนวณคือ ลำดับของกฎบนไฟร์วอลล์ ที่ถูกเรียงลำดับปริมาณการตกกระทบของกลุ่มข้อมูลจากมากไปน้อย

6) เมื่อได้ลำดับของกฎที่ถูกต้องแล้ว ขั้นตอนต่อไปคือการปรับกฎให้เรียงลำดับจากกฎที่ถูกประเมินว่าดีที่สุดขึ้นเป็นกฎลำดับที่ 1 แล้วเรียงกฎอื่นๆ ตามลำดับ (การปรับกฎจะมีผลกระทบกับไฟล์วอลล์ที่กำลังทำงานอยู่ ดังนั้นแนะนำให้ปรับในช่วงเวลาที่ไม่มีผู้ใช้งาน หรือหลัง Office hour)

7) กฎที่ได้รับการปรับใหม่จะเริ่มทำงาน ซึ่งจะกระทบกับกลุ่มข้อมูลที่เข้าและออกในระบบเครือข่ายตามปกติ

8) ในสถานะการณ์จริง บางครั้งการใช้งานโปรแกรมประยุกต์บางอย่างอาจจะเปลี่ยนแปลงไปตามช่วงเวลาที่ได้รับความนิยม หรือมีไวรัส เป็นต้น ซึ่งส่งผลให้ปริมาณการใช้ (Traffic) ในระบบขึ้นและลงไม่คงที่ ดังนั้นถ้าผู้ดูแลระบบใช้เวลา Break-Point สั้นเกินไป จะทำให้กฎถูกปรับตามแนวโน้ม

ของโปรแกรมประยุกต์ที่ใช้งานในแต่ละวัน เพื่อไม่ให้มีผลกระทบต่อการคำนวณ ผู้วิจัยแนะนำให้ตั้งเวลา Break-Point ให้เหมาะสมกับปริมาณการใช้ข้อมูลที่เข้าและออกในองค์กร หรือขึ้นอยู่กับขีดความสามารถของฮาร์ดแวร์ที่ใช้จัดเก็บ Log file เป็นต้น เมื่อถึงเวลาดังกล่าวที่ตั้งไว้ กระบวนการของการปรับกฎก็จะเริ่มต้นอีกครั้ง




***ภาพที่ 3*** *แนวความคิดในการจัดเรียงกฎใหม่*

### 3.3 การวิเคราะห์และออกแบบสมการปรับกฎ

การคำนวณเวลาที่กลุ่มข้อมูลตกกระทบต่อ 1 กฎ ในลำดับแรกจะต้องคำนวณหาค่าของเวลาที่ใช้ในการประมวลผลที่กลุ่มข้อมูลตกกระทบต่อ 1 กฎก่อน โดยแต่ละกฎที่ถูกคำนวณในไฟร์วอลล์จะใช้ทั้งหมด 5 กลุ่มข้อมูล ดังภาพที่ 4

| SIP (32bits) | SP (16bits) | DIP (32bits) | DP (16bits) | SIP (32bits) |
|---|---|---|---|---|

***ภาพที่ 4*** *กลุ่มข้อมูลที่ใช้ในการประมวลผล*

SIP = Source IP คือ หมายเลขไอพีต้นทาง มีขนาด 32 บิต เช่น 172.16.5.20

SP = Source Port คือ หมายเลขพอร์ตต้นทาง มีขนาด 16 บิต เช่น WWW = 80, DNS = 53

DIP = Destination IP คือ หมายเลขไอพีปลายทาง มีขนาด32 บิต เช่น 200.150.80.0

DP = Destination Port คือ หมายเลขพอร์ตปลายทาง มีขนาด 16 บิต เช่น FTP = 20 และ 23 หรือ SMTP = 25

PRO = Protocol คือ โพรโทคอลที่ใช้งาน มีขนาด 8 บิต เช่น TCP = 6, UDP = 17 เป็นต้น

เวลาที่ใช้ในการประมวลผลต่อ 1 กฎ สามารถคำนวณได้จากสมการที่ (2)

$$T_{R_i} = \sum(t_{SIP} + t_{SP} + t_{DIP} + t_{DP} + t_{PRO}) + t_{Action_{(A|D|R)}} \quad (2)$$

ข้อมูลกับกฎที่ $i$

$t_{SIP}$ เวลาที่ใช้ในการประมวลผลเขตข้อมูล Source IP

$t_{SP}$ เวลาที่ใช้ในการประมวลผลเขตข้อมูล Source Port

$t_{DIP}$ เวลาที่ใช้ในการประมวลผลเขตข้อมูล Destination IP

$t_{DP}$ เวลาที่ใช้ในการประมวลผลกลุ่มข้อมูล Destination Port

$t_{PRO}$ เวลาที่ใช้ในการประมวลผลกลุ่มข้อมูล Protocol

$t_{Action_{(A|D|R)}}$ เวลาที่ใช้ในการประมวลผลใน Action เช่น Accept หรือ Drop หรือ Reject

คำนวณหาจำนวนแพ็ตเก็ตที่ตกกระทบในแต่ละกฎ ในสมการที่ (3)

$$Cp_{R_i} = \sum(Packet_{R_i}) \quad (3)$$

โดยที่ $Cp_{R_i}$ คือ ผลรวมของแพ็ตเก็ตที่ตกกระทบในกฎที่ $i$

$Packet_{R_i}$ คือ กลุ่มข้อมูลที่ตกกระทบแต่ละครั้งในกฎที่ $i$

คำนวณหาผลรวมของเวลาที่ตกกระทบในกฎที่ $i$ จากสมการที่ (4)

$$Total_{\beta}(T_{R_i}) = T_{R_i} x Cp_{R_i} \quad (4)$$

โดยที่ $Total_{\beta}(T_{R_i})$ คือ ผลรวมของเวลาที่ใช้ในการประมวลผลบนกฎที่ $i$ โดยพิจารณาอยู่ในช่วงเวลาที่กำหนด คือ $\beta$ โดยมีค่าตั้งแต่ 1 ถึง Break-point สำหรับค่า Break-point หมายถึงเวลาสิ้นสุดในการเก็บบันทึก Log file สมการการปรับกฎ $FPERR_{R_i}$ ในสมการที่ (5)

$$FPERR_{R_i} = Total_{\beta}(T_{R_i}) \times Weight_{R_i} \quad (5)$$

โดย $FPERR_{R_i}$ คือ การประเมินสมรรถนะการประมวลผลของแต่ละกฎบนไฟร์วอลล์

$Weight_{R_i}$ คือ น้ำหนักของแต่ละกฎ ซึ่งค่า Weight จะใช้

สำหรับระบุความสำคัญของกลุ่มข้อมูล มีค่าอยู่ในช่วง 0.0 – 1.0 ตัวอย่างของค่าน้ำหนักที่ใช้กำหนด เช่น โปรแกรมประยุกต์ชนิดประชุมทางไกล ซึ่งทำงานแบบเรียลไทม์ และมีความสำคัญต่อผู้บริหารในองค์กรมาก แต่ทำงานไม่บ่อย จะส่งผลให้ไฟร์วอลล์ที่ปรับกฎแล้ว จะปรับให้โปรแกรมประยุกต์ดังกล่าวอยู่ในลำดับกฎท้ายๆ ซึ่งถ้าต้องการให้โปรแกรมประยุกต์ดังกล่าวทำงานในลำดับต้นๆ ของไฟร์วอลล์ ให้กำหนดค่าน้ำหนักของกฎที่ตรวจสอบโปรแกรมประยุกต์ดังกล่าวมีน้ำหนักเป็น 1 หรือเข้าใกล้เคียงค่า 1 เป็นต้น

จากสมการสิ่งที่เราต้องการคือ ต้องการให้ผลรวมของกลุ่มข้อมูลที่ตกกระทบมากๆ และผลรวมของเวลาที่ใช้ในการประมวลผลน้อยที่สุด “ทำงานมากแต่ใช้เวลาน้อย” จึงถือว่ามีประสิทธิภาพ เมื่อเข้าเงื่อนไขดังกล่าว กฎนั้นๆ จะถือว่ามีลำดับความสำคัญสูงสุด

**3.4 พิสูจน์สมการ** $FPERR_{R_i}$

ในขั้นตอนนี้จะทำการพิสูจน์สมการ $FPERR_{R_i}$ โดยยกตัวอย่างการทำงานของไฟร์วอลล์ดังต่อไปนี้ ตามตารางที่ 2

***ตารางที่ 2*** *การพิสูจน์สมการ* $FPERR_{R_i}$ *(ตัวอย่างที่ 1)*

| กฎ | $T_{R_i}$ | $Cp_{R_i}$ | $Weight_{R_i}$ | $Total_\beta(T_{R_i})$ | $FPERR_{R_i}$ |
|---|---|---|---|---|---|
| 1 | 1 | 15 | 1 | 15 | 15 |
| 2 | 1 | 8 | 1 | 8 | 8 |
| 3 | 1 | 30 | 1 | 30 | 30 |
| 4 | 1 | 55 | 1 | 55 | 55 |
| 5 | 1 | 10 | 1 | 10 | 10 |

จากตารางที่ 2 สมมุติว่าไฟร์วอลล์มีทั้งหมด 5 กฎ โดยกฎที่ 1 ใช้เวลาในการประมวลผลต่อ 1 กฎ คือ 1 วินาที กลุ่มข้อมูลที่ตกกระทบในหนึ่งหน่วยเวล Break-point) เท่ากับ n (เช่น ค่า n = 1 วันเป็นต้น) จำนวนกลุ่มข้อมูลตกกระทบทั้งหมด 15 ครั้ง และค่าน้ำหนักเท่ากับ 1 กฎที่ 2 ใช้เวลาประมวลผล 1 วินาที และตกกระทบ 8 ครั้ง ค่าน้ำหนักเท่ากับ 1 สำหรับกฎที่ 3, 4, 5 เรียงลำดับตามในตาราง ผลลัพธ์ที่ได้ในช่องของ $FPERR_{R_i}$ คือ กฎที่ 4 มีประสิทธิภาพสูงสุด (เพราะผลการคำนวณมีค่าสูงที่สุดเท่ากับ 55) รองลงมาคือ กฎที่ 3(30), 1(15), 5(10) และ 2(8) ตามลำดับ ซึ่งแสดงว่าเมื่อถึงช่วงปรับเปลี่ยนกฎใหม่ ให้จัดเรียงลำดับกฎใหม่ดังนี้ คือ 4, 3, 1, 5, 2 ตามลำดับ ลองพิจารณาตัวอย่างที่ 2 เมื่อค่า $T_{R_i}$ และ $Weight_{R_i}$ เปลี่ยนไป ดังตารางที่ 3

***ตารางที่ 3*** *ผลการคำนวณ* $FPERR_{R_i}$ *(ตัวอย่างที่ 2)*

| กฎ | $T_{R_i}$ | $Cp_{R_i}$ | $Weight_{R_i}$ | $Total_\beta(T_{R_i})$ | $FPERR_{R_i}$ |
|---|---|---|---|---|---|
| 1 | 1.0 | 15 | 0.2 | 15 | 3 |
| 2 | 1.2 | 8 | 0.3 | 9.6 | 2.88 |
| 3 | 1.3 | 30 | 0.5 | 40.5 | 20.25 |
| 4 | 1.7 | 55 | 1.0 | 93.5 | 93.5 |
| 5 | 1.9 | 10 | 0.1 | 19.5 | 1.95 |

จากตารางที่ 3 ต้องเรียงลำดับกฎไฟร์วอลล์ใหม่คือ กฎที่ 4, 3, 1, 2, 5 ตามลำดับ พิจารณาตัวอย่างที่ 3 เมื่อค่า $T_{R_i}$ และ $Weight_{R_i}$ เปลี่ยนไป ดังตารางที่ 4

***ตารางที่ 4*** *ผลการคำนวณ* $FPERR_{R_i}$ *(ตัวอย่างที่ 3)*

| กฎ | $T_{R_i}$ | $Cp_{R_i}$ | $Weight_{R_i}$ | $Total_\beta(T_{R_i})$ | $FPERR_{R_i}$ |
|---|---|---|---|---|---|
| 1 | 1.0 | 20 | 0.2 | 20 | 4 |
| 2 | 1.2 | 25 | 0.3 | 30 | 9 |
| 3 | 1.3 | 30 | 0.5 | 39 | 19.5 |
| 4 | 1.7 | 35 | 1.0 | 59.5 | 59.5 |
| 5 | 1.9 | 15 | 1.0 | 28.5 | 28.5 |

จากตารางที่ 4 ต้องเรียงลำดับกฎไฟร์วอลล์ใหม่คือ กฎที่ 4, 5, 3, 2, 1 ตามลำดับ พิจารณาตัวอย่างที่ 4 เมื่อค่า $T_{R_i}$, $Cp_{R_i}$ และ $Weight_{R_i}$ เปลี่ยนไป ดังตารางที่ 5

***ตารางที่ 5*** *ผลการคำนวณ* $FPERR_{R_i}$ *(ตัวอย่างที่ 4)*

| กฎ | $T_{R_i}$ | $Cp_{R_i}$ | $Weight_{R_i}$ | $Total_\beta(T_{R_i})$ | $FPERR_{R_i}$ |
|---|---|---|---|---|---|
| 1 | 1.1 | 10 | 0.2 | 11 | 2.2 |
| 2 | 1.2 | 10 | 0.3 | 12 | 3.6 |
| 3 | 1.3 | 10 | 0.5 | 13 | 6.5 |
| 4 | 1.3 | 10 | 0.0 | 13 | 0.0 |
| 5 | 1.4 | 10 | 0.9 | 14 | 12.6 |

จากตารางที่ 5 ต้องเรียงลำดับกฎไฟร์วอลล์ใหม่คือ กฎที่ 5, 3, 2, 1, 4 ตามลำดับ

**3.5 พิสูจน์สมการโดยเทียบกับอัลกอริทึม FCFS และ SJF**

จากการทดสอบพบว่าสมการ FPERR มีลักษณะคล้ายกับ

SJF [16] ซึ่งเป็นการยืนยันว่าเมื่อทำตามวิธีการของ SJF แล้วจะทำให้ระยะเวลารอคอยในการประมวลผลน้อยที่สุด และพบว่าไฟร์วอลล์ที่ไม่มีการปรับกฎให้เหมาะสมจะมีลักษณะการทำงานเหมือน FCFS [16] ซึ่งจะให้ผลลัพธ์ของเวลารอคอยในการประมวลผลช้าที่สุด จากตารางที่ 6 แสดงตัวอย่างตารางเวลาการทำงานของไฟร์วอลล์ เพื่อใช้ทดสอบกับ FCFS และ SJF ได้ดังนี้

***ตารางที่ 6*** *ตัวอย่างของตารางเวลาการทำงานของไฟร์วอลล์*

| กฎ | $T_{R_i}$ | $Cp_{R_i}$ | $Total_{\beta}(T_{R_i})$ |
|---|---|---|---|
| 1 | 1.1 | 10 | 11 |
| 2 | 1.2 | 10 | 12 |
| 3 | 1.3 | 10 | 13 |
| 4 | 1.3 | 10 | 13 |
| 5 | 1.4 | 10 | 14 |

จากสมการ Average Waiting Time ($AVG$) ในสมการที่ (6)

$$AVG = \frac{(T_{R_1} + T_{R_2} + T_{R_3} + ... + T_{R_n})}{n}$$

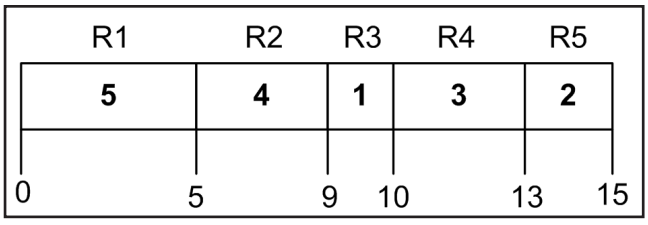
เมื่อ $AVG$ คือ ผลรวมของระยะเวลารอคอย

$T_{R_i}$ คือ ระยะเวลาการรอคอยเพื่อประมวลผลของกฎใดๆ

n คือ จำนวนกฎทั้งหมดในการประมวลผล

ทดสอบแทนค่า $AVG$ ของ FCFS =

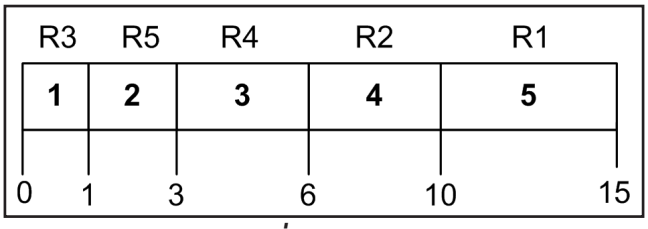
$T_{R_1} + T_{R_2} + T_{R_3} + T_{R_4} + T_{R_5} /5 = (0+5+9+10+13)/5 = 7.4$




***ภาพที่ 5*** *FCFS Scheduling*

จากภาพที่ 5 เมื่อคำนวณผลรวมของเวลารอคอยเฉลี่ยของ FCFS มีค่าเท่ากับ 7.4 เมื่อมีการเรียงกฎ คือ 1, 2, 3, 4, 5 ตามลำดับ ซึ่งไม่มีการจัดเรียงตามความเหมาะสม

ทดสอบแทนค่า $AVG$ ของ SJF = (0+1+3+6+10)/5 = 4

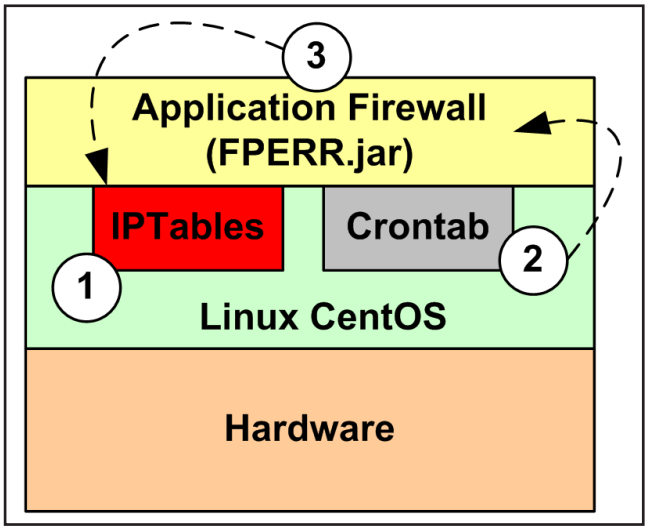



***ภาพที่ 6*** *Scheduling*

จากภาพที่ 6 เมื่อคำนวณผลรวมของเวลารอคอยเฉลี่ยของ SJF มีค่าเท่ากับ 4 เมื่อมีการเรียงกฎ คือ 3, 5, 4, 2, 1 ตามลำดับ ส่งผลให้ไฟร์วอลล์มีประสิทธิภาพสูงสุด

**3.6 พัฒนาไฟร์วอลล์จริงบนระบบปฏิบัติการลินุกซ์**

เมื่อได้สมการ FPERR แล้ว ขั้นต่อไปคือ การพัฒนาไฟร์วอลล์ต้นแบบที่ทำงานตามสมการดังกล่าว ให้สามารถทำงานได้บนระบบเครือข่ายจริง ในงานวิจัยนี้ทำการพัฒนาโปรแกรมชื่อว่า FPERR ด้วยภาษาจาวา (FPERR.jar) โดยทำงานร่วมกับ IP-Tables [17] ติดตั้งบนระบบปฏิบัติการลินุกซ์ CentOS เวอร์ชัน 5.5 บนเครื่องคอมพิวเตอร์ Pentium II processor 223 เม็กกะเฮิร์ท (MHz) หน่วยความจำหลัก 512 เม็กกะไบต์ (MB) การ์ดเน็ตเวิร์คความเร็ว 100 เม็กกะบิตต่อวินาที (Mbps) ไฟร์วอลล์แบ่งออกเป็น 2 โซนคือ Trust zone และ Untrust zone และตั้งเวลาการทำงานด้วยโปรแกรม Crontab ดังภาพที่ 7




***ภาพที่ 7*** *แสดงการทำงานของโปรแกรม FPERR*

จากภาพที่ 7 แสดงขั้นตอนการทำงานของโปรแกรม FPERR ซึ่งมีขั้นตอนดังนี้

1) ติดตั้งไฟร์วอลล์ IPTables บนระบบปฏิบัติการลินุกซ์ และปรับแต่งกฎของ IPTables ให้สามารถใช้งานได้ตามปกติ ดังภาพที่ 8 และพร้อมกับสั่งให้ไฟร์วอลล์บันทึกการทำงานลง Log file ดังภาพที่ 9

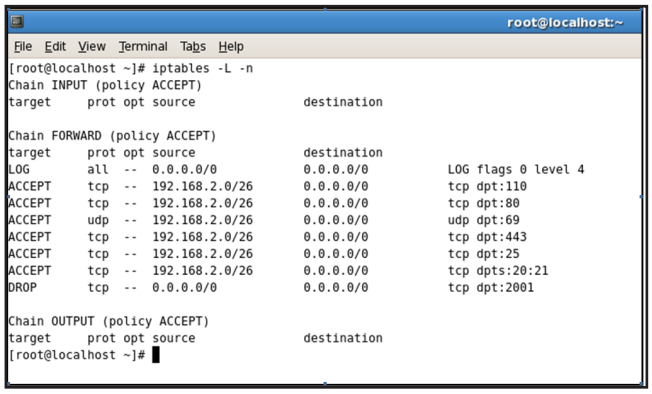
2) ตั้งค่าระยะเวลาในการปรับกฎ (Break-Point) โดยอาศัย Crontab ซึ่งเป็นโปรแกรมที่ฝังมากับลินุกซ์ เพื่อทำหน้าที่ควบคุมการทำงานของโปรแกรมอื่นๆ ให้ทำงานตามเวลาที่กำหนด ในที่นี้ให้ควบคุมการทำงานของโปรแกรม FPERR.jar ซึ่งมีคำสั่งดังภาพที่ 10 ซึ่งเป็นโปรแกรมที่ผู้วิจัยได้พัฒนา

ขึ้น เพื่อให้คำนวณและปรับกฎของไฟร์วอลล์ตามหลักการที่ถูกออกแบบไว้แล้วข้างต้น

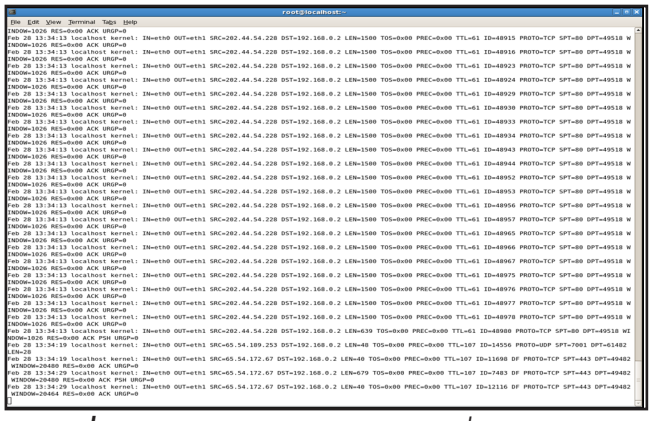
จากภาพที่ 10 กำหนดเวลาให้ Crontab สั่งให้โปรแกรม FPERR.jar ทำงานทุกๆ วันอาทิตย์ เวลาเที่ยงคืน

3) เมื่อโปรแกรม FPERR.jar ถูกประมวลผลแล้ว จะได้กฎที่ถูกจัดเรียงอย่างเหมาะสมเรียบร้อยแล้ว ขั้นต่อไปคือสั่งให้ IPables เปลี่ยนลำดับของกฎใหม่ และเริ่มต้นการทำงานของไฟร์วอลล์ใหม่อีกครั้ง ดังภาพที่ 11

จากภาพที่ 11 สังเกตุเห็นว่ากฎที่ทำงานมีประสิทธิภาพมากที่สุดจะถูกปรับลำดับไว้บนสุดของกฎใน IPTabels ในตัวอย่าง กฎที่ตรวจสอบโปรแกรมประยุกต์เว็บ (พอร์ตหมายเลข 80) จะถูกปรับอยู่ในลำดับบนสุด ซึ่งเป็นที่นิยมใช้งานมากที่สุดในทุกๆ องค์กร รองลงมาคือ โปรแกรมประยุกต์เว็บแบบปลอดภัย (พอร์ตหมายเลข 443) และอิเล็กทรอนิกส์เมล (พอร์ตหมายเลข 25) ตามลำดับ




***ภาพที่ 8*** *แสดงการทำงานของ IPTables ก่อนการปรับกฎ*



***ภาพที่ 9*** *แสดงการทำงานของ IPTables ก่อนการปรับกฎ*

```
**** 0/usr/bin/java    -jar /root/FPERR.jar
```

***ภาพที่ 10*** *แสดงการตั้งเวลาของ Crontab สั่งงานให้โปรแกรม FPERR.jar ทำงานตามเวลาที่กำหนด*

```
Chain ACCEPT)
target source destination

Chain ACCEPT)
target source destination
LOG 0.0.0.0/0 0.0.0.0/0 LOG flags

ACCEPT 192.168.2.0/26 0.0.0.0/0 tcp dpt:80
ACCEPT 192.168.2.0/26 0.0.0.0/0 tcp dpt:443
ACCEPT 192.168.2.0/26 0.0.0.0/0 tcp dpt:25
ACCEPT 192.168.2.0/26 0.0.0.0/0 tcp dpt:110
ACCEPT 192.168.2.0/26 0.0.0.0/0 tcp dpts:20:21
ACCEPT 0.0.0.0/0 0.0.0.0/0 tcp dpt:22
ACCEPT 0.0.0.0/0 0.0.0.0/0 tcp dpt:23
ACCEPT 0.0.0.0/0 0.0.0.0/0 tcp dpt:119
ACCEPT 192.168.2.0/26 0.0.0.0/0 udp dpt:69
ACCEPT 0.0.0.0/0 0.0.0.0/0 udp dpt:161
DROP 0.0.0.0/0 0.0.0.0/0 tcp dpt:2001

Chain ACCEPT)
target source destination
```

***ภาพที่ 11*** *แสดงกฎที่ถูกปรับแล้ว*

## 4. ผลการทดลอง

จากผลการทดลองโปรแกรม FPERR สามารถทำงานได้อย่างถูกต้องตรงตามทฤษฎีที่เสนอไว้ โดยสามารถปรับกฎตามประสิทธิภาพจากสูงไปต่ำ ส่งผลให้ไฟร์วอลล์ทำงานได้เร็วขึ้นจริง จากผลการทดสอบในหัวข้อ 3.4 เป็นวิธีการทดสอบด้วยคณิตศาสตร์ หัวข้อที่ 3.5 เป็นการทดสอบเทียบความเร็วกับอัลกอริทึมที่มีลักษณะการทำงานที่ใกล้เคียงกัน และหัวข้อ 3.6 เป็นการทำสอบกับการทำงานจริง โดยเปรียบเทียบความเร็วก่อนและหลังการปรับกฎ ซึ่งแสดงผลการเปรียบเทียบในตารางที่ 7

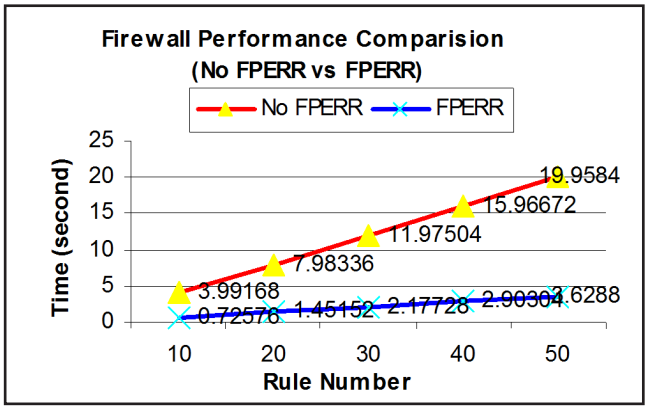
***ตารางที่ 7*** *เปรียบเทียบความเร็วในการประมวลผลก่อนและหลังการปรับกฎของไฟร์วอลล์ที่ติดตั้งและใช้งานจริง*

| การทดลองที่ | Break-point (วัน) | จำนวนกฎ | ก่อนปรับกฎ (วินาที) | หลังปรับกฎ (วินาที) |
|---|---|---|---|---|
| 1 | 7 | 10 | 3.991 | 0.725 |
| 2 | 7 | 20 | 7.983 | 1.451 |
| 3 | 7 | 30 | 11.975 | 2.177 |
| 4 | 7 | 40 | 15.966 | 2.903 |
| 5 | 7 | 50 | 19.958 | 3.628 |
| 6 | 30 | 10 | 17.1072 | 3.1104 |
| 7 | 30 | 20 | 34.2144 | 6.2208 |
| 8 | 30 | 30 | 51.3216 | 9.3312 |
| 9 | 30 | 40 | 68.4288 | 12.4416 |
| 10 | 30 | 50 | 85.536 | 15.552 |

***ตารางที่ 7*** *เปรียบเทียบความเร็วในการประมวลผลก่อนและหลังการปรับกฎของไฟร์วอลล์ที่ติดตั้งและใช้งานจริง*

| การทดลองที่ | Break-point (วัน) | จำนวนกฎ | ก่อนปรับกฎ (วินาที) | หลังปรับกฎ (วินาที) |
|---|---|---|---|---|
| 11 | 180 | 10 | 102.6432 | 18.6624 |
| 12 | 180 | 20 | 205.2864 | 37.3248 |
| 13 | 180 | 30 | 307.9296 | 55.9872 |
| 14 | 180 | 40 | 410.5728 | 74.6496 |
| 15 | 180 | 50 | 513.216 | 93.312 |
| 16 | 360 | 10 | 205.2864 | 37.3248 |
| 17 | 360 | 20 | 410.5728 | 74.6496 |
| 18 | 360 | 30 | 615.8592 | 111.9744 |
| 19 | 360 | 40 | 821.1456 | 149.2992 |
| 20 | 360 | 50 | 1,026.432 | 186.624 |

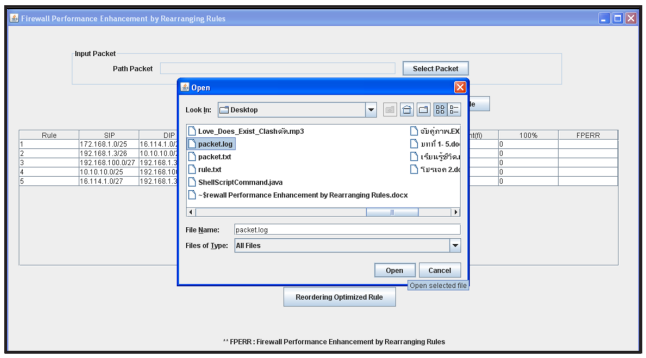
จากตารางที่ 7 เวลาที่ใช้ในการประมวลผลแต่ละกฎของไฟร์วอลล์จะแปรผันตามจำนวนฟิลด์ที่ใช้ตรวจสอบ รูปแบบการกรองข้อมูล และจำนวนของกฎ ซึ่งเวลาประเมินโดยเฉลี่ยต่อกฎประมาณ 0.66 ไมโครวินาทีต่อกฎ (กรณียังไม่ได้ปรับกฎ) และ 0.12 ไมโครวินาทีต่อกฎ (กรณีที่ปรับกฎแล้ว) สำหรับรูปแบบการทดลอง จะกำหนดค่า Break-point หรือเวลาในการทดสอบปรับกฎ โดยเริ่มจาก 168 ชั่วโมง (7 วัน) 1 เดือน 6 เดือน และ 1 ปีตามลำดับ และเพิ่มจำนวนกฎเพิ่มขึ้นโดยเริ่มตั้งแต่ 10, 20, 30, 40 และ 50 ตามลำดับ (ปกติไฟร์จะมีกฎไม่มากซึ่งควรอยู่ระหว่าง 10 – 50 กฎ) จากนั้นทำการเปรียบเทียบความเร็วก่อนและหลังการปรับกฎ ซึ่งแสดงดังตารางที่ 7 และภาพที่ 12 แสดงให้เห็นว่าหลังจากการปรับกฎที่เหมาะสมแล้วเวลาที่ใช้ในการประมวลผลจะลดลง สำหรับการประเมินผลไฟร์วอลล์ในช่วงเวลาตั้งแต่ 1 วันถึงประมาณ 30 วันนั้น สามารถทำได้ เพราะพื้นที่ในการจัดเก็บ Log file ยังเพียงพอ แต่ในการเก็บเป็นเวลา 6 เดือนหรือ 1 ปีนั้น ต้องใช้พื้นที่ในการจัดเก็บมาก ดังนั้นจากตารางที่ 7 ตั้งแต่การทดลองที่ 11 ถึง 20 จะใช้เวลาจากการประเมิน ซึ่งไม่ได้จับเวลาในการใช้งานจริง เนื่องจากติดปัญหาดังที่กล่าวมาแล้วข้างต้น




***ภาพที่ 12*** *แสดงกราฟเปรียบเทียบความเร็วในการประมวลผลก่อน (No FPERR) และหลัง (FPERR) การปรับกฎไฟร์วอลล์*

## 5. สรุปผลการทดลอง

ในงานวิจัยนี้ผู้วิจัยได้ออกแบบสมการที่สามารถปรับลำดับกฎบนไฟร์วอลล์ ทำให้ไฟร์วอลล์มีประสิทธิภาพการทำงานที่สูงขึ้น และพัฒนาโปรแกรมที่ช่วยวิเคราะห์และตรวจสอบประสิทธิภาพการทำงานของไฟร์วอลล์ ซึ่งผู้วิจัยได้พัฒนาโปรแกรมจำลองขึ้นก่อน (Firewall Simulation ในภาพที่ 13) เพื่อใช้ทดสอบทฤษฎีการสลับกฎก่อน




***ภาพที่ 13*** *Firewall Simulation*

จากนั้นได้ทำการพัฒนาโปรแกรมชื่อว่า FPERR ที่สามารถทำงานได้จริงบนระบบปฏิบัติการลินุกซ์ โดยโปรแกรมดังกล่าวสามารถทำงานได้สมบูรณ์ตรงตามทฤษฎีที่เสนอไว้และสามารถช่วยให้ประสิทธิภาพการทำงานของไฟร์วอลล์ดีขึ้นจริง

## 7. เอกสารอ้างอิง

[1] Lee, T.K.Y., S. Luk, W. Sloman, A. Lupu, E. N. Dulay, "Development framework for firewall processors," in Field-Programmable Technology, 2002. (FPT).

*Proceedings. 2002 IEEE International Conference on 16-18 Dec*. 2002.

[2] Laturnas, D.B., R.,. "Dynamic silicon firewall," *in Electrical and Computer Engineering 2005*, *Canadian Conference* on 1-4 May 2005 Saskatoon, Sask. : IEEE.

[3] Benelbahri, M.A.B., A.,. "Tuple Based Approach for Anomalies Detection within Firewall Filtering Rules," *in Computers and Communications 2007. ISCC 2007*. 12th IEEE Symposium on 1-4 July 2007 Aveiro.

[4] Al-Shaer, E.S.H., H.H. "Discovery of policy anomalies in distributed firewalls," *in INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* 7-11 March 2004.

[5] Al-Shaer, E.S.H., H.H. "Firewall Policy Advisor for anomaly discovery and rule editing," *in Integrated Network Management 2003. IFIP/IEEE Eighth International Symposium* on 24-28 March 2003.

[6] Al-Shaer, E.S.H., H.H. "Modeling and Management of Firewall Policies," *in Network and Service Management, IEEE Transactions on*. April 2004 IEEE.

[7] Bartal, Y.M., A. Nissim, K. Wool, A. "Firmato: a novel firewall management toolkit." *in Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium*, Oakland, CA, 1999.

[8] El-Alfy, E.-S.M.S., S.Z. "On Optimal Firewall Rule Ordering," *in Computer Systems and Applications 2007. AICCSA '07. IEEE/ACS International Conference* on 13-16 May 2007 Amman.

[9] Golnabi, K.M., R.K. Khan, L. Al-Shaer, E. "Analysis of Firewall Policy Rules Using Data Mining Techniques," i*n Network Operations and Management Symposium*, 2006. NOMS 2006. 10th IEEE/IFIP Vancouver, BC. 3-7 April 2006.

[10] Katic, T.P., P. "Optimization of Firewall Rules," *in Information Technology Interfaces 2007. ITI 2007. 29th International Conference* on 25-28 June 2007.

[11] Weiping Wang, et al. "Firewall Rule Ordering Based on Statistical Model," *in Computer Engineering and Technology, 2009. ICCET '09. International Conference* on 22-24 Jan. 2009 Singapore.

[12] Zhang, M.K.Y.S.C.Z. "Reducing the Size of Rule Set in a Firewall," *in Communications 2007. ICC '07. IEEE In-ternational Conference* on 24-28 June 2007 Glasgow.

[13] สุชาติ คุ้มมะณี และจุรภรณ์ ตั้งมั่นดี. "การจัดการกฎของไฟร์วอลล์แบบกึ่งอัตโนมัติ," *วารสารเทคโนโลยีสารสนเทศพระจอมเกล้าพระนครเหนือ* (Information Technology Journal) ปีที่ 5 ฉบับที่ 9 มกราคม-มิถุนายน 2552, หน้า 8-17.

[14] จุรีภรณ์ ตั้งมั่นดี ปริญญา ระดาบุตร และ สุชาติ คุ้มมะณี. "ประยุกต์ VLSM กับการตรวจสอบกฎของ ไฟล์วอลล์," *in In Proceedings of the Joint conference on Computer Science and Software Engineering* (JCSSE). Kanjanaburi, Thailand. May 2008.

[15] Freed, N., "Behavior of and Requirements for Internet Firewalls," in RFC 2979. Internet Engineering Task Force. pp. 3. October 2000.

[16] Abraham Silberschatz. *Operating System Concepts*. Seventh Edition ed. December 14, 2004: John Wiley & Sons.

[17] ภ. ด่านระหาญ. *Linux 2.4 Stateful Firewall : IPTABLES*. 3 ธันวาคม 2544, ศูนย์ประสานงานการรักษาความปลอดภัยคอมพิวเตอร์ ประเทศไทย ThaiCERT (NECTEC) Dec 2010 [online] Available Form: http://thaicert.nectec.or.th/paper/firewall/iptables.php