การปรับปรุงการจัดการกฎไฟร์วอลล์ด้วยแนวคิดการตัดสินใจแบบโดเมนเดี่ยว

Improving the Firewall Rule Management by a Single Domain Decision Concept

อธิพงศ์ คำสีลา (Atipong Khumseela)* สมนึก พ่วงพรพิทักษ์ (Somnuk Puangpronpitag)*
และ สุชาติ คุ้มมะณี (Suchart Khummanee)*

บทคัดย่อ

การออกแบบระบบเครือข่ายให้ปลอดภัยในปัจจุบันนิยมใช้ไฟร์วอลล์เพื่อทำหน้าที่กำหนดกฎเกณฑ์การเข้าถึงการบริการต่างๆ บนระบบเครือข่าย โดยปฏิบัติตามเงื่อนไขหรือกฎของไฟร์วอลล์ที่ถูกกำหนดไว้ โดยต้องมีความสอดคล้องกับพฤติกรรมการใช้งานของแต่ละองค์กร ไฟร์วอลล์นิยมติดตั้งระหว่างเครือข่ายภายในองค์กร (Intranet) กับเครือข่ายภายนอก (Internet) ปริมาณของข้อมูลที่ไหลผ่านไฟร์วอลล์จะมีปริมาณสูง และผ่านเข้าออกตลอดเวลา ดังนั้นประสิทธิภาพการทำงานโดยรวมของไฟร์วอลล์จะขึ้นอยู่กับกฎที่ได้กำหนดไว้ หากกฎของไฟร์วอลล์ถูกกำหนดไว้ไม่เหมาะสมแล้ว จะส่งผลให้ประสิทธิภาพโดยรวมของไฟร์วอลล์ลดลง เช่น กฎเกิดความขัดแย้ง ซ้ำซ้อน หรือ ไม่ถูกประมวลผล เป็นต้น งานวิจัยนี้จึงได้ออกแบบและพัฒนาไฟร์วอลล์ต้นแบบโดยอาศัยหลักการที่เรียกว่า การตัดสินใจแบบโดเมนเดี่ยว (A Single Domain Decision: SDD) ซึ่งเป็นแนวความคิดที่สามารถขจัดปัญหาความขัดแย้งของกฎไฟร์วอลล์ได้อย่างสมบูรณ์

คำสำคัญ: การจัดการกฎของไฟร์วอลล์ ความขัดแย้งของกฎไฟร์วอลล์ การตรวจสอบกฎของไฟร์วอลล์

Abstract

Firewall is an important system to control the access of networks. The firewall performance depends significantly on firewall rule management. Bad firewall rule management (causing by firewall rule anomalies such as conflict, duplicate, redundant) can badly affect the efficiency of both firewall & network. So, in this paper, we design and develop a new prototype firewall that can completely eliminate firewall anomalies by proposing a Single Domain Decision (SDD) Firewall concept. According to the SDD, the firewall can freely swap the rules with no effect to the overall rule meaning. Experimental results have demonstrated that our proposed design can get rid of firewall rule's anomalies completely.

Keyword: Firewall rule management, Firewall rule anomaly, Firewall rule verification.

1. บทนำ

เมื่อกล่าวถึงมาตรการการรักษาความปลอดภัยบนเครือข่ายไฟล์วอลล์ เป็นระบบที่นักออกแบบระบบเครือข่ายให้ความสำคัญเป็นลำดับต้นๆ เนื่องจากไฟร์วอลล์มีคุณสมบัติที่สามารถป้องกันการเข้าถึงบริการบนระบบเครือข่ายได้เป็นอย่างดี โดยปกตินิยมติดตั้งไฟร์วอลล์ระหว่างทางเข้าออกขององค์กรกับเครือข่ายสาธารณะ สำหรับการตรวจสอบข้อมูลไฟร์วอลล์ จะทำการเปรียบเทียบข้อมูลที่ไหลผ่านตัวเองกับกฎที่ได้ถูกกำหนดไว้ หรือเรียกว่า Policy (โดยปกติจะเป็นหน้าที่ของผู้ดูแลระบบเครือข่าย) ข้อมูลทั้งหมดที่ผ่านเข้าและออกจากองค์กรจะต้องผ่าน

* สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาการสารสนเทศ มหาวิทยาลัยมหาสารคาม

ไฟร์วอลล์เสมอ ส่งผลให้ประสิทธิภาพโดยรวมของไฟร์วอลล์จะขึ้นอยู่กับการบริหารจัดการกฎที่ดี เมื่อผู้ดูแลระบบขาดความรู้ ความเข้าใจในพฤติกรรมการทำงานขององค์กรจะส่งผลให้มีโอกาสสร้างกฎที่ผิดปกติได้หลายลักษณะ เช่น การสร้างกฎขัดแย้งกันเอง กฎมากเกินความจำเป็นและไม่ถูกใช้งาน สร้างกฎที่ซับซ้อนและเข้าใจได้ยาก หรือการจัดลำดับของกฎที่ไม่เหมาะสม เป็นต้น สำหรับงานวิจัยเกี่ยวกับการบริหารจัดการกฎในปัจจุบันสามารถจำแนกได้หลายกลุ่ม ดังนี้ 1) กลุ่มนักวิจัยที่ต้องการยุบรวมกฎให้สั้น และกระชับ [1-5] 2) กลุ่มที่ต้องการลดความซับซ้อน และความขัดแย้งของกฎ [6], [8] 3) กลุ่มที่ต้องการจัดลำดับกฎตามพฤติกรรมผู้ใช้งาน [8-10] และ 4) กลุ่มที่ต้องการลดความผิดพลาดจากการสร้างกฎโดยการปรับปรุงส่วนเชื่อมต่อกับผู้ใช้ [11-12] ทั้ง 4 กลุ่มที่กล่าวมาแล้วข้างต้นมีเป้าหมายที่เหมือนกันคือต้องการให้ไฟร์วอลล์ที่พัฒนาขึ้นปราศจากความขัดแย้งน้อยที่สุด หรือไม่มีเลย แต่ผู้วิจัยของ ISAN Lab [13] ค้นพบว่าวิธีการเหล่านี้ไม่สามารถแก้ปัญหาความขัดแย้งได้อย่างสมบูรณ์ เนื่องจากไม่ได้แก้ปัญหาที่ต้นตอของสาเหตุ ซึ่งสาเหตุสำคัญของความขัดแย้งของกฎไฟร์วอลล์นั้นเกิดขึ้นจากการยอมให้กฎของไฟร์วอลล์ ตั้งแต่สองกฎขึ้นไปทับซ้อนกัน และมีการดำเนินการ (Action) ต่างกัน

งานวิจัยนี้ได้มุ่งเน้นการแก้ไขปัญหาความขัดแย้งของกฎไฟร์วอลล์ที่ต้นตอของสาเหตุ โดยอาศัยแนวความคิดที่เรียกว่า การตัดสินใจแบบขอบเขตเดี่ยว (A Single Domain Decision: SDD) ที่ถูกคิดค้นเริ่มต้น โดย Khummanee และทีมวิจัยด้านความมั่นคงของระบบเครือข่าย (ISAN) [13] และแสดงให้เห็นถึงความสามารถในการกำจัด Anomalies เนื่องจากแนวความคิดดังกล่าวสามารถขจัดปัญหากฎที่ขัดแย้งกันได้อย่างสมบูรณ์ กฎเมื่อถูกสร้างด้วยแนวคิดดังกล่าวแล้วจะสามารถอ่านได้ง่าย และลำดับของกฎที่ถูกสร้างจะไม่มีผลกระทบต่อความความเร็วของไฟร์วอลล์

2. วรรณกรรมที่เกี่ยวข้อง

2.1 การทำงานของไฟร์วอลล์

โดยหลักการทำงานพื้นฐาน ไฟร์วอลล์จะประกอบไปด้วยกฎ หรือโพลิซีที่กำหนดไว้ก่อนล่วงหน้า กฎดังกล่าวจะเป็นสิ่งที่บ่งบอกว่าข้อมูล หรือ แพ็คเกต อะไรบ้างที่สามารถไหลผ่านเข้าออกในระบบเครือข่ายได้ เมื่อข้อมูล รูปแบบทั่วๆ ไปของกฎไฟร์วอลล์สามารถเขียนได้ดังนี้

กฎที่ 1 : ⟨*เงื่อนไข*⟩ → ⟨*การตัดสินใจ*⟩
กฎที่ 2 : ⟨*เงื่อนไข*⟩ → ⟨*การตัดสินใจ*⟩
...
กฎที่ N : ⟨*เงื่อนไข*⟩ → ⟨*การตัดสินใจ*⟩

⟨เงื่อนไข⟩ คือการดำเนินการทางคณิตศาสตร์พีชคณิตบูลีนของส่วนประกอบ 5 ฟิลด์ในแต่ละกฎบนไฟร์วอลล์ คือ หมายเลขไอพีต้นทาง (Source Address: S_ip) หมายเลขไอพีปลายทาง (Destination Address: D_ip) หมายเลขพอร์ตต้นทาง (Source Port: S_port) หมายเลขพอร์ตปลายทาง (Destination Port: D_port) และโพรโทคอล (Protocol: PRO) ⟨การตัดสินใจ⟩ หรือ Action คือ เงื่อนไขที่ไฟร์วอลล์ต้องกระทำเมื่อไฟร์วอลล์ได้ทำการเปรียบเทียบกฎทั้ง 5 กับข้อมูลที่ผ่านเข้าออกตัวมัน ถ้าผลลัพธ์ในการเปรียบเทียบข้างต้นเป็นจริง ไฟร์วอลล์จะต้องเลือกทำอย่างใดอย่างหนึ่งที่กำหนดไว้ใน ⟨เงื่อนไข⟩ คือ อนุญาต หรือปฏิเสธ

2.2 ความขัดแย้งของกฎไฟร์วอลล์

ความขัดแย้งของกฎไฟร์วอลล์ (Firewall rule anomalies) จากงานวิจัยของ Al-Shaer et al [5] มีด้วยกันหลายรูปแบบ แต่มีจำนวน 4 รูปแบบที่พบปัญหามากที่สุด ดังนี้

2.2.1 Shadowing Anomaly คือ การที่กฎใดๆ ถูกบังโดยกฎอื่นอยู่ลำดับก่อนหน้า เมื่อทุกฟิลด์ของ Rule2 เป็นสมาชิกของ Rule1 ทั้งสองกฎมีผลการกระทำที่แตกต่างกัน ดังตารางที่ 1

ตารางที่ 1 *กฎของไฟร์วอลล์ที่ขัดแย้งแบบ Shadowing*

Rule	S_ip	D_ip	S_port	D_port	PRO	Action
1	192.168.1.0/24	All	All	All	TCP	Deny
2	192.168.1.10	All	All	80	TCP	Accept
3	All	All	All	All	All	Deny

2.2.2 Correlation Anomaly คือ การที่กฎใดๆ มีการเกี่ยวพันกับกฎอื่น เมื่อบางฟิลด์ของ Rule1 เป็นสมาชิกของ Rule2 และมีบางฟิลด์ของ Rule2 เป็นสมาชิกของ Rule1 ทั้งสองกฎมีผลการกระทำที่แตกต่างกัน ดังตารางที่ 2

ตารางที่ 2 *กฎของไฟร์วอลล์ที่ขัดแย้งแบบ Correlation*

Rule	S_ip	D_ip	S_port	D_port	PRO	Action
1	192.168.1.0/24	All	All	80	TCP	Deny
2	All	172.16.100	All	80	TCP	Accept
3	All	All	All	All	All	Deny

2.2.3 Generalization Anomaly คือ การที่กฎใดๆ คลุมกฎอื่น เมื่อทุกฟิลด์ของ Rule1 เป็นสมาชิกของแต่ละฟิลด์บน Rule2 ทั้งสองกฎมีผลการกระทำที่แตกต่างกัน ดังตารางที่ 3

ตารางที่ 3 *กฎของไฟร์วอลล์ที่ขัดแย้งแบบ Generalization*

Rule	S_ip	D_ip	S_port	D_port	PRO	Action
1	192.168.5.10	All	All	21	TCP	Deny
2	192.168.5.0/24	All	All	21	TCP	Accept
3	All	All	All	All	All	Deny

2.2.4 Redundancy Anomaly คือ การที่กฎใดๆ ซ้ำซ้อนกับกฎอื่น เมื่อทุกฟิลด์ของ Rule1 เป็นสมาชิกของ Rule2 ทั้งสองมีผลการกระทำที่เหมือนกัน ดังตารางที่ 4

ตารางที่ 4 *กฎของไฟร์วอลล์ที่ขัดแย้งแบบ Redundancy*

Rule	S_ip	D_ip	S_port	D_port	PRO	Action
1	192.168.5.10	All	All	21	TCP	Deny
2	192.168.5.0/24	All	All	21	TCP	Deny
3	All	All	All	All	All	Deny

ซึ่งมีหลากหลายงานวิจัย ที่พยายามขจัดปัญหาเหล่านี้ [1-15] โดยพบว่าการแก้ไขปัญหาแม้ว่าจะได้ผลที่ดี แต่ต้นเหตุของปัญหายังคงไม่ได้ถูกแก้ไขปัญหาอย่างจริงจัง ซึ่งจริงๆ แล้วต้นเหตุของปัญหาเกิดจากการยอมให้กฎใดๆ สองกฎมีผลการกระทำที่สามารถทับซ้อนกันโดยมีการตัดสินใจที่ต่างกันเกิดขึ้นได้ ซึ่งงานวิจัยนี้ได้นำเสนอการแก้ไขปัญหาอย่างสมบูรณ์

3. ที่มาและแรงจูงใจของปัญหา

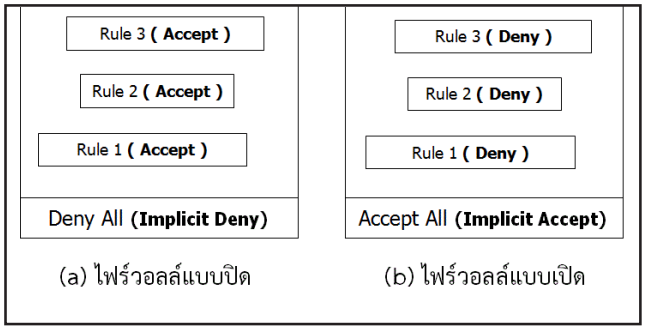
การทำงานของไฟร์วอลล์โดยทั่วไป (Traditional Firewall) จะทำการตรวจสอบกฎแบบเรียงลำดับ (Sequence) แพ็คเกตส่วนใหญ่ที่ผ่านเข้า และออกจากไฟร์วอลล์ จะตกกระทบกับกฎที่เรียกว่า การปฏิเสธโดยปริยาย (Implicit deny) ซึ่งไม่อนุญาตให้ข้อมูลผ่านไปได้ ซึ่งแสดงให้เห็นว่าแพ็คเกตส่วนใหญ่จะถูกกำจัดทิ้งไป สำหรับประสิทธิภาพของไฟร์วอลล์จะลดลงเมื่อจำนวนกฎของไฟร์วอลล์มีปริมาณมากๆ แต่แพ็คเกตไม่ตรงกับเงื่อนไขของกฎเหล่านั้นที่ถูกสร้างขึ้นเลย และมาตกกระทบกับกฎการปฏิเสธโดยปริยายทั้งหมดตลอดเวลา และการตรวจสอบแบบตามลำดับจะส่งผลให้ความเร็วในการประมวลผลลดลง นอกจากนี้ยังมีความผิดพลาดที่เกิดจากการยอมให้กฎของไฟร์วอลล์มีผลการกระทำ (Action) ที่แตกต่างกันเกิดขึ้น เป็นทั้งชนิดอนุญาตและชนิดปฏิเสธในขณะเดียวกัน ผลคือกฎบางส่วนของไฟร์วอลล์ทับซ้อนกัน และอาจจะไม่ถูกประมวลผล เป็นต้น จากเหตุผลที่กล่าวมาแล้ว งานวิจัยนี้มีแรงจูงใจที่จะแก้ปัญหาการตรวจสอบกฎแบบไม่เรียงลำดับ และขจัดกฎที่เกิดความขัดแย้งในไฟร์วอลล์ให้หมดไปโดยสมบูรณ์

4. การออกแบบและพัฒนาไฟร์วอลล์ด้วย SDD

งานวิจัยได้เสนอวิธีที่เรียกว่า การตัดสินใจขอบเขตเดี่ยว (A Single Domain Decision) [13] ซึ่งวิธีดังกล่าวจะขจัดความขัดแย้งตั้งแต่เริ่มต้นการสร้างกฎของไฟร์วอลล์ ซึ่งอธิบายในย่อหน้าถัดไป

4.1 การออกแบบและพัฒนา SDD

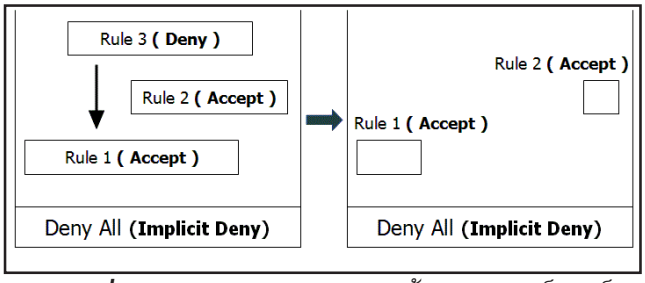
โดยพื้นฐานของไฟร์วอลล์มี 2 ประเภทคือ ระบบที่ปิดให้บริการทุกๆ บริการก่อนเสมอ เรียกว่า ไฟร์วอลล์แบบปิด จากนั้นเมื่อต้องการใช้บริการใดๆ เช่น เปิดให้บริการเว็บไซต์ จึงเปิดใช้งานเฉพาะบริการนั้นๆ เท่านั้น (ระบบนี้ค่อนข้างปลอดภัย และเป็นที่นิยมในปัจจุบัน) และระบบที่เปิดให้บริการทั้งหมดก่อนเสมอ เรียกว่า ไฟร์วอลล์แบบเปิด สำหรับการทำงานของระบบนี้ จะทำงานตรงกันข้ามกับไฟร์วอลล์แบบปิด เช่น เมื่อเกิดเหตุการณ์ที่อาจจะเสี่ยงต่อความมั่นคงในภายหลัง เช่น ไวรัส ถูกโจมตี หรือหนอนอินเทอร์เน็ต จึงเริ่มปิดบริการหรือพอร์ตเหล่านั้นในภายหลัง ดังภาพที่ 1(a) และ 1(b)



ภาพที่ 1 *การออกแบบไฟร์วอลล์การตัดสินใจเดี่ยว*

ฉะนั้นสถานะของกฎในไฟร์วอลล์จะไม่เกิดความขัดแย้งกัน ถ้าหากไฟร์วอลล์แบบปิดจะมีเงื่อนไขเฉพาะการตัดสินใจแบบอนุญาตเท่านั้น และถ้าหากไฟร์วอลล์เป็นชนิดแบบเปิดจะต้องมีเงื่อนไขเฉพาะการตัดสินใจ ปฏิเสธเท่านั้น ในกรณีที่ไฟร์วอลล์แบบปิด เมื่อต้องการเพิ่มกฎที่เป็นปฏิเสธเพิ่ม

เข้าไปในระบบ ไฟร์วอลล์จะต้องขจัดเงื่อนไขที่เป็นปฏิเสธออก และไปรวมกับกฎที่มีการตัดสินใจแบบปฏิเสธโดยปริยายโดยอัตโนมัติ ดังภาพที่ 2 เมื่อกฎที่เพิ่มเข้ามาใหม่ (กฎที่ 3) และมีเงื่อนไขการดำเนินการคือ ปฏิเสธ (Deny) ไฟร์วอลล์แบบปิดจะตัดส่วนที่คาบเกี่ยวกันออกจากกฎที่ 1 และ 2 ออก ให้เหลือไว้เฉพาะกฎที่เป็นการยอมรับอย่างเดียวเท่านั้น



ภาพที่ 2 *แนวคิดวิธีขจัดความขัดแย้งของกฎไฟร์วอลล์*

4.2 การออกแบบโครงสร้างข้อมูล

ไฟร์วอลล์ที่มีการออกแบบใหม่จะใช้โครงสร้างต้นไม้ ซึ่งจะมีประสิทธิภาพในการเข้าถึงข้อมูลได้อย่างรวดเร็ว สำหรับโครงสร้างของไฟร์วอลล์ที่เก็บข้อมูลจะถูกออกแบบให้มีความลึกเป็นค่าคงที่อยู่ที่ 5 ระดับเท่านั้น คือ ระดับที่ 1 คือ Root แทนด้วยหมายเลขไอพีต้นทาง (S_ip) ระดับที่ 2 แทนด้วยหมายเลขไอพีปลายทาง (D_ip) ระดับที่ 3 แทนด้วยหมายเลขพอร์ตต้นทาง (S_ip) ระดับที่ 4 แทนด้วยหมายเลขพอร์ตปลายทาง (D_ip) และระดับที่ 5 แทนด้วยโพรโทคอล (PRO)

การกำหนดขอบเขตของข้อมูลบนโครงสร้างต้นไม้ทวิภาคได้แบ่งการเก็บข้อมูลของแต่ละฟิลด์ เช่น S_ip เป็นช่วงของข้อมูลคือ ข้อมูลเริ่มต้น (S_ip_1) และข้อมูลสุดท้าย (S_ip_2) ซึ่งข้อมูลในแต่ละฟิลด์จะถูกเก็บเป็นเลขฐานสิบ ในฟิลด์ข้อมูลที่แทนด้วยหมายเลขไอพี (S_ip และ D_ip) จะต้องนำมาเปลี่ยนให้เป็นเลขฐานสิบตามสูตรข้างล่าง

หมายเลขไอพี = $S\_ip\ (octet4) \times 2^{24} + S\_ip\ (octet3) \times 2^{16} + S\_ip\ (octet2) \times 2^{8} + S\_ip\ (octet1) \times 2^{0}$

การเปลี่ยนหมายเลขไอพี 192.168.1.100 สามารถเปลี่ยนให้เป็นเลขฐานสิบได้ดังนี้

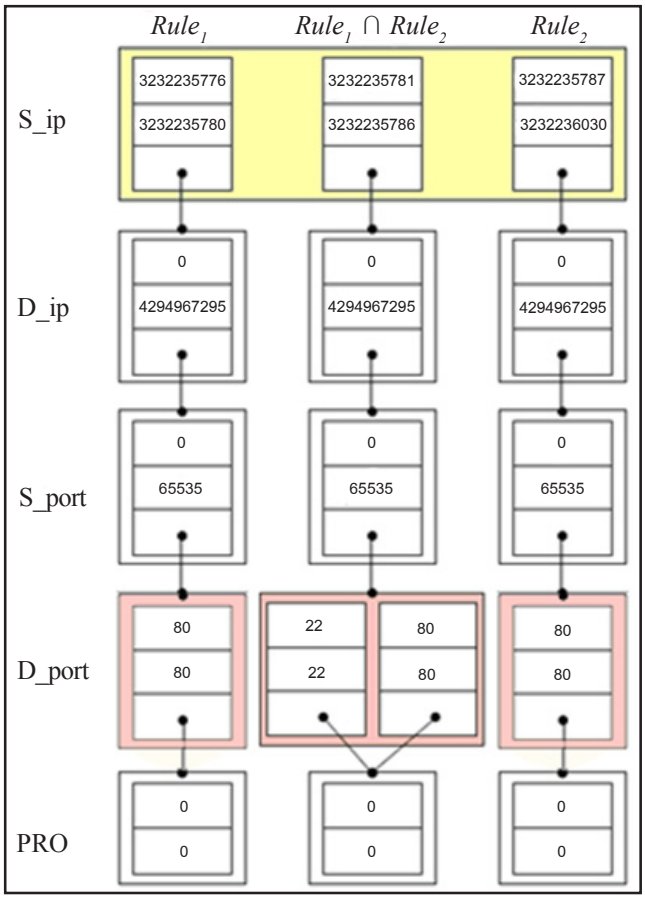
$192.168.1.100 = (192 \times 16{,}777{,}216) + (168 \times 65{,}536) + (1 \times 256) + (100 \times 1) = 3{,}232{,}235{,}876$

ในกรณีที่หมายเลขไอพีเป็นช่วงของไอพี หรือช่วงของหมายเลขพอร์ต จะคำนวณช่วงไอพีช่วงต้น และช่วงท้ายเท่านั้น เช่น หมายเลขไอพีต้นทาง 192.168.1.0/24 ช่วงของไอพีเริ่มต้นคือ 192.168.1.1 - 192.168.1.254 สามารถเปลี่ยนเป็นเลขฐานสิบได้คือ ข้อมูลเริ่มต้น S_ip (S_ip_1) = 3,232,236,031 และข้อมูลสิ้นสุด S_ip (S_ip_2) = 3,232,236,031 ผู้วิจัยสามารถกำหนดข้อมูลลงในโครงสร้างต้นไม้บนระบบไฟร์วอลล์แบบปิด แสดงดังตัวอย่างต่อไปนี้ สมมุติว่ามีกฎ 2 กฎ คือ

Rule 1: $< (S\_ip \in \{192.168.1.0/24\} \wedge (D\_ip \in all) \wedge (S\_port \in all) \wedge (D\_port \in \{80\}) \wedge PRO \in \{0\} > \rightarrow <Accept>$

Rule 2: $< (S\_ip \in \{192.168.1.4/30\} \wedge (D\_ip \in all) \wedge (S\_port \in all) \wedge (D\_port \in \{22\}) \wedge PRO \in \{0\} > \rightarrow <Accept>$

Rule 1 มีความสัมพันธ์เกี่ยวเนื่องกับ Rule 2 จะได้ว่า Rule 1 ∩ Rule 2 : $(S\_ip \in 192.168.1.4/30) \wedge (D\_ip \in all) \wedge (S\_port \in all) \wedge (D\_port \in \{80, 22\}) \wedge PRO \in \{0\} > \rightarrow <Accept>$ นั่นคือช่วงของหมายเลขไอพีต้นทาง 192.168.1.4/30 มีส่วนกระทำทั้ง Rule 1 และ Rule 2 จะทำให้กฎของไฟร์วอลล์ในโครงสร้างของต้นไม้แตกกฎเพิ่มขึ้นได้กฎใหม่ ดังภาพที่ 3



ภาพที่ 3 *ไฟร์วอลล์การตัดสินแบบของเขตเดี่ยว*

4.3 การดำเนินการบน SDD

เมื่อมีการเพิ่มหรือลบกฎของไฟร์วอลล์ ไฟร์วอลล์จะมีการตรวจสอบกฎตั้งแต่เริ่มต้นการสร้างกฎ โดยใช้อัลกอริธึมการสร้าง ลบ และแก้ไข ซึ่งอ้างอิงจาก Khummanee et al. [13] ดังนี้

Algorithm 1: *SDD inserting*

1: **SET** X = Firewall Rule /*set of fields that want to insert*/
2: **SET** *Fwr* = Insert Rule (Rule) /*any firewall rule*/
3: **LOOP 1**: for every element i in X
4: **IF** (INTERSECT (X(i), *Fwr*)) **THEN**
5: **IF** (X(i) != *Fwr*) **THEN**
6: $S1_{data1} \leftarrow \min(X(i)_{data1}, Fwr_{data1})$
7: **IF** $(X(i)_{data1} == FA_{data1})$ **THEN**
8: $S1_{data2} \leftarrow \min(X(i)_{data2}, Fwr_{data2}) - 1$
9: $S2_{data1} \leftarrow \min(X(i)_{data2}, Fwr_{data2})$
10: **ELSE THEN**
11: $S1_{data2} \leftarrow \min(\max(X(i)_{data1}, Fwr_{data1})) - 1$
12: $S2_{data1} \leftarrow \min(\max(X(i)_{data1}, Fwr_{data1}))$
13: **END**
14: $S2_{data2} \leftarrow \max(X(i)_{data2}, Fwr_{data2}))$
15: **IF** (INTERSECT (X(i),S1)) **AND** (INTERSECT (X(i),S2)) **THEN**
16: $X(i)_{data1} \leftarrow S1_{data2}$
17: Add S2 in X
18: **ELSE IF**(INTERSECT (S2, X(i))) **THEN**
19: Add S1 in X
20: **END**
21: **IF** $(S2_{data2} - S1_{data2} != 0)$ **THEN**
22: $Fwr_{data1} \leftarrow \min(X(i)_{data2}, Fwr_{data2}) + 1$
23: $Fwr_{data2} \leftarrow \max(X(i)_{data2}, Fwr_{data2})$
24: **ELSE STOP**
25: **END**
26: **END**
27: **END**
28: **IF** (*Fwr* is not Empty) **THEN**
29: ADD *Fwr* in X
30: **END**
31: SORT (X)

Algorithm 2: *INTERSECT* (set 1, set 2)

1: **IF** $((set1_{data1} >= set2_{data1})$ **AND** $(set1_{data1} <= set2_{data2}))$ **OR** $((set1_{data2} >= set2_{data1})$ **AND** $(set1_{data2} <= set2_{data2}))$ **THEN**
RETURN TRUE
2: **ELSE RETURN** FALSE
3: **END**

Algorithm 3: *SDD deleting*

1: **SET** X = Firewall Rule
2: **SET** *Fwr* = Delete Rule (Rule)
3: FA is Sub-member of X
4: **IF** (X(i)!=Fwr) **THEN**
5: **IF** $(X(i)_{data1} == Fwr_{data1})$ **THEN**
6: $X(i)_{data1} \leftarrow Fwr_{data2} + 1$
7: **ELSE IF** $(X(i)_{data1} == Fwr_{data2})$ **THEN**
8: $X(i)_{data2} \leftarrow Fwr_{data1} - 1$
9: **ELSE THEN**
10: $S1_{data2} \leftarrow$ Temp
11: $X(i)_{data2} \leftarrow Fwr_{data1} - 1$
12: $S1_{data1} \leftarrow Fwr_{data2} + 1$
13: Add S1 in X
14: **END**
15: **ELSE**
16: Remove *Fwr* from X
17: SORT (X)

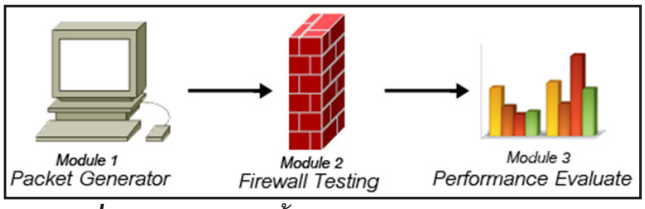
ในการเพิ่มหรือลบกฎของไฟร์วอลล์เข้าใหม่ทุกครั้งจะต้องมีการเรียงลำดับข้อมูลในแต่ละฟิลด์ที่มีการเพิ่มเข้าไป ซึ่งจะใช้อัลกอริธึม Timsort ในการเรียงข้อมูล ซึ่งมีประสิทธิภาพในการทำงานเท่ากับ O(N log N) ตามการวิเคราะห์ของ Khummanee et al [13]

5. การประเมินต้นแบบไฟร์วอลล์

5.1 การดำเนินการบน SDD

ในส่วนนี้ได้ทดสอบประสิทธิภาพของไฟร์วอลล์โดยการเปรียบเทียบระหว่างไฟร์วอลล์แบบดั้งเดิมกับไฟร์วอลล์ตัดสินใจแบบโดเมนเดี่ยว แบ่งเป็น 2 ขั้นตอนคือ ขั้นตอนแรกงานวิจัยจะประเมินการขจัดปัญหาความขัดแย้งของกฎไฟร์วอลล์

โดยใช้แนวคิด SDD และขั้นตอนที่สอง จะประเมินเวลาในการบริหารจัดการกฎ ซึ่งประกอบไปด้วย เวลาในการสร้างกฎ รวมกับเวลาในการตรวจสอบกฎ และวัดปริมาณการใช้พื้นที่หน่วยความจำที่ใช้งาน โดยการทดสอบทั้งหมด ได้เขียนซอฟแวร์จำลองประสิทธิภาพการทำงาน ซึ่งได้พัฒนาโดยใช้ Java SE 7 ทำงานบนระบบปฏิบัติการ Window 7, CPU Quad-Core 3.30 Ghz และหน่วยความจำ (RAM) 8 GB งานวิจัยนี้ได้ออกแบบขั้นตอนการเปรียบเทียบประสิทธิภาพของไฟร์วอลล์ ดังภาพที่ 4 โดยมีรายละเอียดดังต่อไปนี้



ภาพที่ 4 *การออกแบบขั้นตอนการทดสอบประสิทธิภาพ*

1) Packet Generator คือ การสุ่มข้อมูลหรือแพ็คเกตเพื่อใช้ในการตรวจสอบเข้ากับไฟร์วอลล์แบบใหม่ โดยแพ็คเกตที่ส่งไปได้จากการสุ่มข้อมูลทั้งหมด 100,000 แพ็คเกต ในส่วนประกอบของแพ็คเกตที่ส่งไปจะประกอบไปด้วย SA, DA, SP, DP และ PRO เพื่อใช้ในการวัดประสิทธิภาพการทำงานของไฟร์วอลล์

2) Firewall Testing คือ การทดสอบประสิทธิภาพของไฟร์วอลล์แบบดั้งเดิม และ SDD โดยไฟร์วอลล์จะทำการสุ่มกฎขึ้นมาตั้งแต่ 1 – 16,384 กฎ และทำการตรวจสอบกฎที่สุ่มขึ้นมาในขั้นตอน Packet Generator เพื่อประเมินประสิทธิภาพ

3) Performance Evaluate คือ การประเมินผลการทำงานของไฟร์วอลล์ ซึ่งได้แบ่งออกเป็น 2 ส่วนย่อยด้วยกัน คือ ระยะเวลาที่ใช้ในการสร้างไฟร์วอลล์ (Sorting Time + Insert Time) กับระยะเวลาที่ใช้ในการตรวจสอบข้อมูล (Verification Time) ซึ่งได้อธิบายในหัวข้อต่อไป

5.2 ผลการทดสอบและการวิจารณ์ผล

5.2.1 การขจัดปัญหาความขัดแย้งของไฟร์วอลล์ ผลจากการทดสอบในขจัดความขัดแย้งของกฎไฟร์วอลล์ ซึ่งงานวิจัยนี้ ได้ขจัดปัญหาความขัดแย้งของกฎไฟร์วอลล์ [5] ทั้ง 4 อย่าง ได้แก่ Shadowing Anomaly, Correlation Anomaly, Generalization Anomaly และ Redundancy Anomaly ได้อย่างสมบูรณ์ อย่างไรก็ตาม จำนวนกฎของไฟร์วอลล์อาจจะเพิ่มจำนวนมากกว่าไฟร์วอลล์แบบดั้งเดิม เนื่องจากเกิดการแบ่งกฎจากกฎสองกฎใดๆ ที่มีการตัดสินใจที่ขัดแย้งกันให้เหลือเพียงการตัดสินใจเดียว แต่ความเร็วในการตรวจสอบกฎยังคงไม่เปลี่ยนแปลง

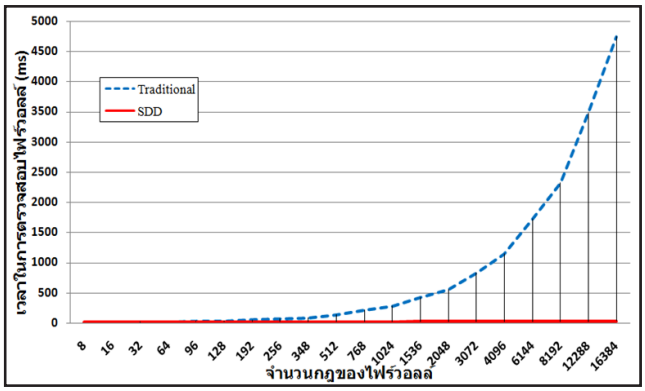
5.2.2 ประเมินการเปรียบเทียบประสิทธิภาพของไฟร์วอลล์ ในการวัดประสิทธิภาพโดยรวมของระบบโดยทั่วไป มีส่วนสำคัญ 2 ส่วนในการประเมินผลคือ เวลาที่ใช้ในการประมวลผล (Time Complexity) และจำนวนของการใช้พื้นที่ในหน่วยความจำ (Space Complexity) ซึ่ง Khummanee et al. [13] ได้วิเคราะห์การใช้ SDD ในเชิงประสิทธิภาพไว้ดังนี้

ประสิทธิภาพโดยรวมของไฟร์วอลล์ = (เวลาที่ใช้ในการประมวลผล + เวลาที่ใช้ในการตรวจสอบข้อมูล) + จำนวนของการใช้พื้นที่ในหน่วยความจำโดยแสดงวัดประสิทธิภาพดังตารางที่ 5

ตารางที่ 5 *การเปรียบเทียบประสิทธิภาพโดยการคำนวณ*

Performance Metrics	Firewall Type	
	Traditional	*SDD*
Time for building structure	C	C + O(L log N) (Timsort)
Time verifying	$O(N^2)$ Sequential search	O(M log N) Binary search
Space complexity	O(N)	O(2N-1)

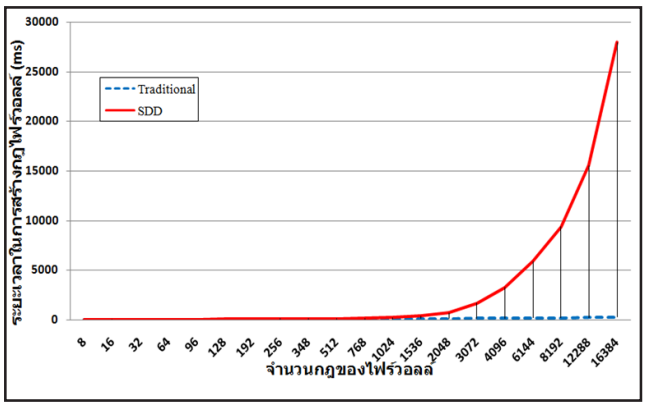
จากการทดสอบแบ่งออกเป็น 3 กลุ่มคือ 1) เปรียบเทียบเวลาที่ใช้ในการตรวจสอบระหว่างไฟร์วอลล์ทั่วไปกับ SDD ไฟร์วอลล์ 2) เปรียบเทียบเวลาที่ใช้ในการสร้างกฎของไฟร์วอลล์ และ 3) เปรียบเทียบเวลาที่ใช้หน่วยความจำ ตามลำดับ



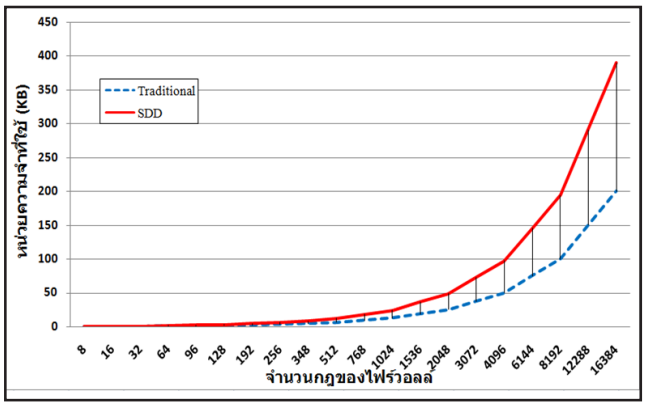
ภาพที่ 5 *เปรียบเทียบเวลาที่ใช้ในการตรวจสอบข้อมูล*

จากภาพที่ 5 ค่าเวลาเฉลี่ยที่ใช้ในตรวจสอบข้อมูลระหว่าง SDD และไฟร์วอลล์แบบดั้งเดิมซึ่งแกนนอนของกราฟประกอบไปด้วยจำนวนของกฎตั้งแต่ 1 – 16,384 และแนวตั้งแสดงเวลาในการตรวจสอบโดยเฉลี่ยตั้งแต่ 0 – 5,000 จากกราฟ สังเกตว่าช่วงจำนวนของกฎระหว่าง 8 ถึง 348 จะมีระยะเฉลี่ยการตรวจสอบกฎใกล้เคียงกัน หลังจากจำนวนกฎประมาณ 512 กฎขึ้นไป เวลาเฉลี่ยที่ใช้ในการตรวจสอบของไฟร์วอลล์แบบดั้งเดิมจะเพิ่มสูงขึ้นอย่างรวดเร็ว ในทางตรงกันข้าม SDD ใช้ระยะเวลาเฉลี่ยในการตรวจสอบเพิ่มขึ้นจากเดิมเพียงเล็กน้อย

แม้ว่าเวลาการตรวจสอบกฎของ SDD ค่อนข้างดีกว่าไฟร์วอลล์ทั่วไป แต่เวลาเฉลี่ยของการสร้างโครงสร้างของกฎไฟร์วอลล์ สำหรับ SDD จะใช้เวลาเฉลี่ยในการสร้างเพิ่มขึ้นจาก 400 เป็น 700 มิลลิวินาทีสำหรับ 2,000 กฎ ในขณะที่ไฟร์วอลล์แบบดั้งเดิมยังคงมีค่าเฉลี่ยของเวลาในการสร้างกฎเกือบคงที่ และเพิ่มขึ้นเล็กน้อย เมื่อจำนวนของกฎเพิ่มขึ้น ดังในภาพที่ 6



ภาพที่ 6 *เปรียบเทียบเวลาที่ใช้ในการสร้างกฎ*



ภาพที่ 7 *เปรียบเทียบการใช้พื้นที่ของหน่วยความจำ*

ในความเป็นจริงแล้วประสิทธิภาพของไฟร์วอลล์ที่ดีจะขึ้นอยู่กับระยะเวลาที่ใช้ในการตรวจสอบข้อมูลซึ่งจะมีความสำคัญเป็นอันดับแรก และตามปกติไฟร์วอลล์จะไม่ปรับปรุงกฎบ่อยหากไม่มีความจำเป็น สำหรับกรณีของประมาณการใช้พื้นที่หน่วยความจำระหว่างไฟร์วอลล์ทั้งสองจะไม่แตกต่างกันมาก โดยจะมีแนวโน้มการใช้หน่วยความจำที่สูงขึ้นเมื่อกฎที่สร้างมีจำนวนมากขึ้นตามลำดับ ดังภาพที่ 7

6. บทสรุป

งานวิจัยนี้ได้นำเสนอแนวความคิดในการแก้ปัญหาความขัดแย้งของกฎไฟร์วอลล์ใหม่ เรียกว่า การตัดสินใจแบบขอบเขตเดี่ยว ซึ่งแนวความคิดดังกล่าวสามารถขจัดความขัดแย้งของกฎได้อย่างสมบูรณ์ แต่กฎของไฟร์วอลล์ที่สร้างขึ้นจากแนวความคิดนี้จะส่งผลให้ไฟร์วอลล์จะมีจำนวนกฎมากกว่าไฟร์วอลล์โดยทั่วไป เพื่อแก้ปัญหาเวลาที่ใช้การตรวจสอบกฎที่เพิ่มขึ้นนี้ จึงได้ออกแบบอัลกอริทึม รวมถึงโครงสร้างข้อมูลที่เพิ่มประสิทธิภาพการทำงานให้สามารถตรวจสอบกฎได้รวดเร็วขึ้น โดยอาศัยคุณสมบัติของต้นไม้ (tree) และการค้นหาแบบทวิภาค (binary search)

7. เอกสารอ้างอิง

[1] A. Liu. “Formal Verification of Firewall Policies.” *In Proceedings of IEEE International Conference on Communications,* Beijing, China, pp. 1494-1498, May 2008.

[2] K. Golnabi, R. Min, L. Khan and E. Al-Shaer. “Analysis of Firewall Policy Rules Using Data Mining Techniques.” I*n Proceedings of IEEE/IFIP Network Operations and Management Symposium,* Vancouver, Canada, pp. 305-315, April 2006.

[3] A. Bouhoula and Z. Trabelsi. “Handling Anomalies in Distributed Firewalls.” *In Proceeding of IEEE Innovations in Information Technology,* Dubai, United Arab emirates, pp. 1-5, November 2006.

[4] R. Winding,W. Timothy and C. Michael. “System Anomaly Detection Mining Firewall Logs.” *In Proceedings of International Conference on Security and Privacy in Communication Networks and the*

Workshops, Baltimore, Maryland, USA, pp. 1-5, August 2006.

[5] E. Al-Shaer and H. Harmed. "Modeling and Management of Firewall Policies." *IEEE Transactions on Network and Service Management,* Vol. 1, Issue 1, pp. 2-10, April 2004.

[6] A. Liu and E. Torng. "Firewall Compressor : An Algorithm for Minimizing Firewall Policies." *In Proceedings of INFOCOM,* Phoenix, Arizon, USA, pp. 176-180, March 2008.

[7] M. Rezvani and R. Aryan. "Analyzing and Resolving Anomalies in Firewall Security Policies Based on Propositional Logic." *In Proceedings of International Multitopic Conference,* Islamabad, Pakistan, pp. 1-7, December 2009.

[8] H. Hamed, A. El-Atawy and E. Al-Shaer. "On Dynamic Optimization of Packet Matching in High-Speed Firewalls." *IEEE Journal on Selected Areas in Communications,* Vol. 24, Issue. 10, pp. 1817-1830, October 2006.

[9] S. Acharya, J. Wang, Z. Ge, T.F. Znati and A. Greenberg. "Traffic-Aware Firewall Optimization Strategies." *In Proceedings of International Conference on Communications,* Istanbul, Turkey, pp. 2225-2230, June 2006.

[10] E. Sayed and M. El-Alfy. "A Heuristic Approach for Firewall Policy Optimization." *IEEE/ACM Transactions on International Conference on Advanced Communication Technology,* Vol. 3, Issue. 1, Gangwon-Do, South Korea, pp. 1782-1787, February 2007.

[11] M. Ye and K. Sandrasegaran. "Teaching about Firewall Concepts using the iNetwork Simulator." *In Proceeding of International Conference on Information Technology Based Higher Education & Training,* Sydney, Australia, pp. 889-892, July 2006.

[12] J. Sun, H. Chen and C. Niu. "A New Database Firewall Based on Anomaly Detection." *In Proceedings of International Conference on Parallel and Distributed Computing, Applications and Technologies,* Wuhan, China, pp. 399-404, December 2010.