บทความวิจัย (Research Article)

การออพติไมซ์พารามิเตอร์ระบบป้องกันการบุกรุก Suricata สำหรับเครือข่ายความเร็วสูงแบบมัลติกิกะบิต

Parameter Optimization for Suricata Intrusion Prevention Systems supporting Multi-Gigabit High Speed Network

วราฤทธิ์ หงษ์กำเนิด, พีรวัฒน์ วัฒนพงศ์ และ สุรศักดิ์ สงวนพงษ์*

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
มหาวิทยาลัยเกษตรศาสตร์
* ผู้รับผิดชอบบทความ
surasak.s@ku.ac.th

Received: 15 Jan 2021
Revised: 26 May 2021
Accepted: 31 Dec 2021

บทคัดย่อ

บทความนี้นำเสนอการศึกษาและปรับตั้งพารามิเตอร์ของ Suricata ซึ่งเป็นระบบป้องกันการบุกรุก (Intrusion Prevention Systsm: IPS) ที่ใช้แพร่หลาย การวางแผนใช้งานไอพีเอสอย่างเช่น Suricata จำเป็นต้องคำนึงถึงการทำงานแบบหลายเธรดสำหรับหน่วยประมวลผลแบบหลายแกน (Multi-Thread/Multi-Cores) ภายใต้เครื่องแม่ข่ายแบบ COTS (Commodity-Of -The-Shelf) เพื่อให้ได้สมรรถนะการทำงานในเครือข่ายความเร็วสูงระดับหลายกิกะบิต เช่นในงานวิจัยนี้เน้นกรณีเครือข่ายระดับ 10 กิกะบิต งานวิจัยชิ้นนี้ทดลองและศึกษาเพื่อหาตัวแปรที่เหมาะสมสำหรับปรับแต่ง Suricata และเปรียบเทียบวิธีการรับแพ็กเก็ตจากอินเทอร์เฟซเครือข่าย 2 วิธีหลักคือ AF_PACKET และ NFQ โดยให้ทำงานแบบกระจายตัวในหลายแกนประมวลผลพร้อมกัน รวมทั้งศึกษาการจัดวางเธรดการประมวลผลเพื่อหารูปแบบที่ส่งผลให้ได้สมรรถนะการทำงานสูง ผลการศึกษาพบว่าวิธี AF_PACKET ให้สมรรถนะที่สูงกว่า NFQ นอกจากนี้ยังพบว่าควรจัดวางเธรดที่ใช้ทำงานของ Suricata โดยเลี่ยงการข้ามหน่วยประมวลผลเพื่อให้ได้สมรรถนะที่ดีกว่า

คำสำคัญ: IPS, Suricata, AF_PACKET, NFQ

Abstract

This paper proposes a study and measurement of one of the most well-known Intrusion Protection System called Suricata under real-time traffic protection of an enterprise network. The challenge problem in IPS deployment is to assure the performance in multigigabit environment. In this experiment, the testbed is tested under 10 Gbps network. Major goals are to find the combination of Suricata parameters to optimize the overall performance. The experiments cover both AF_PACKET and NFQ packet capture technique. The results show that AF_PACKET yields better performance over NFQ. Moreover, Suricata worker thread should be placed on the same CPU so that no communications overhead will not affect the overall performance.

Keywords: IPS, Suricata, AF_PACKET, NFQ

1. บทนำ

ระบบป้องกันการบุกรุก (Intrusion Prevention System: IPS) หรือไอพีเอส เป็นระบบที่พัฒนาขึ้นเพื่อใช้ตรวจจับและป้องกันการโจมตีระบบคอมพิวเตอร์หรืออุปกรณ์เครือข่าย รูปแบบของไอพีเอสอาจเป็นซอฟต์แวร์ที่ติดตั้งในเครื่องแม่ข่าย หรืออาจเป็นแอพลายแอนซ์ที่สร้างมาสำเร็จรูป ข้อมูลจราจรที่ผ่านเข้าออกเครือข่ายจะถูกส่งผ่านไอพีเอสเพื่อตรวจจับความผิดปกติแบบเรียลไทม์ โดยการเทียบลักษณะเฉพาะของแพ็กเก็ตที่ผ่านเข้ามากับกฎที่ตั้งไว้ โดยปกติหากไอพีเอสตรวจพบรูปแบบแพ็กเก็ตที่ตรงกับกฎที่ตั้งไว้ก็จะกำจัดแพ็กเก็ตนั้นไม่ให้ผ่านเข้าออกเครือข่าย หรือผู้ดูแลระบบสามารตั้งค่าดำเนินการอื่นใดกับแพ็กเก็ตนั้นตามที่ต้องการได้ ในปัจจุบันมีซอฟต์แวร์ไอพีเอสแบบโอเพนซอร์สอยู่หลากหลาย และที่นิยมใช้แพร่หลายอย่างมากคือ Snort และ Suricata

Suricata เป็นซอฟต์แวร์ไอพีเอสสมรรถนะสูงที่ออกแบบมาให้งานเซิร์ฟเวอร์ COTS (Commodity-Of-The-Shelf) ที่สามารถหาซื้อได้ทั่วไป โดยสามารถทำงานแบบมัลติเธรด (Multi Thread) ร่วมกับหน่วยประมวลผลแบบหลายแกน (Multi Core) และหลายหน่วยประมวลผล (Multi Processor) ได้อย่างมีประสิทธิภาพ ภายใต้สถาปัตยกรรม NUMA (Non Uniform Memory Access) [1] ที่อนุญาตให้ใช้หน่วยความจำร่วมกันแบบข้ามแกนหน่วยประมวลผลได้ ข้อนี้นับเป็นจุดเด่นของ Suricata ที่เหนือกว่า Snort ซึ่งพัฒนาขึ้นมาก่อน แต่หากปรับแต่งการทำงานแบบมัลติเทรดและพารามิเตอร์อื่นไม่เหมาะสมกับสภาพระบบที่มีอยู่ก็จะส่งผลให้เกิดแพ็กเก็ตสูญหาย (Loss) มากกว่าที่เกิดใน Snort

บทความนี้จะนำเสนอผลการศึกษาการปรับแต่งพารามิเตอร์ของ Suricata เพื่อหาผู้ดูแลระบบสามารถประเมินหาค่าที่เหมาะสมที่สนับสนุนให้ Suricata ทำงานได้อย่างมีสมรรถนะสูงสอดคล้องกับทรัพยากรของเครื่องแม่ข่ายที่มีอยู่ นอกจากนี้จะนำเสนอผลการวัดสมรรถนะเมื่อเลือกใช้เทคนิคการรับแพ็กเก็ตระหว่าง AF_PACKET และ NFQ โดยวัดจากอัตราการใช้ซีพียู อัตราเร็วการส่งออกแพ็กเก็ต และการสูญเสียแพ็กเก็ตที่เกิดขึ้นจากทั้งสองวิธีเปรียบเทียบกัน

หัวข้อถัดไปจะกล่าวถึงหลักการทำงานของ Suricata และงานวิจัยที่เกี่ยวข้อง จากนั้นจะอธิบายถึงการออกแบบและผลการทดสอบระบบที่วัดจากการวัดด้วยข้อมูลจราจรจริงที่สำเนาบันทึกไว้ และหัวข้อสุดท้ายเป็นการสรุปผล

2. แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

Suricata ทำงานในลักษณะมัลติเธรดโดยจัดแบ่งประเภทเธรดในระบบไว้ 4 แบบ คือ

- Packet acquisition: ทำหน้าที่ในการรับแพ็กเก็ตจากอินเทอร์เฟซเครือข่ายเพื่อนำมาประมวลผล มีวิธีการรับแพ็กเก็ตเมื่อใช้เป็นไอพีเอส หลายแบบได้แก่ AF_PACKET, NFQ, netmap และ ipfw
- Decode and stream application: ทำหน้าที่ถอดข้อมูลแพ็กเก็ต รวมแพ็กเก็ตเข้าด้วยกัน และอื่น ๆ
- Detection: ทำหน้าที่ตรวจสอบข้อมูลในแพ็กเก็ตว่าตรงกับกฎที่ตั้งไว้
- Outputs: ทำหน้าที่แจ้งเตือนให้ผู้ดูแลระบบรับทราบ

Suricata มีพารามิเตอร์ที่สามารถปรับแต่งเพื่อให้แต่ละเธรดทำงานได้อย่างมีสมรรถนะตามสภาพแวดล้อม หนึ่งในพารามิเตอร์สำคัญคือการปรับแต่งให้ใช้ประโยชน์จาก NUMA ได้ โดยช่วยให้หน่วยประมวลผลแต่ละตัวเสมือนมีหน่วยความจำแยกเป็นของตนเอง ขณะที่สภาพจริงนั้นมีจะเพียงหน่วยความจำส่วนกลางที่เชื่อมต่อกับหน่วยประมวลผลผ่านบัส

กลไกของ NUMA ช่วยลดอัตราการใช้แบนด์วิดท์ และร่นเวลาเข้าถึงหน่วยความจำ เมื่อเข้าถึงหน่วยความจำที่ต่อกับหน่วยประมวลผลในโหนดเดียวกัน (Local memory) และยังสามารถเข้าถึงหน่วยความจำในโหนดอื่นได้ (Remote memory) แต่มีเวลาในการเข้าถึงนานกว่า

งานวิจัยใน [2] ได้ทดสอบและเปรียบเทียบด้านสมรรถนะและความแม่นยำระหว่าง Snort และ Suricata โดยสร้างการโจมตีจำลองด้วย Metasploit แล้วทดสอบกับข้อมูลอัตรา 3.1 Mbps ภายใต้ข้อกำหนดว่าจะต้องมีอัตราการสูญหายของแพ็กเก็ตไม่เกิน 2% ได้ข้อสรุปว่า Suricata มีความแม่นยำมากกว่า Snort และสมรรถนะที่ดีกว่าเมื่อประมวลผลแบบหลายแกน อีกงานวิจัยหนึ่ง [3] ได้นำเสนอผลคล้ายคลึงกันโดยใช้ Pytbull เป็นเฟรมเวิร์คทดสอบความแม่นยำ รวมทั้งตัวแปรที่ใช้เปรียบเทียบ

สมรรถนะและการใช้งานหน่วยความจำด้วย งานวิจัย [4] เสนอข้อเปรียบเทียบกับ Suricata กับ Snort ด้านสมรรถนะและความแม่นยำบนระบบปฏิบัติการวินโดวส์และลินุกซ์ พบว่า Suricata มีอัตราการใช้งานทรัพยากรสูงกว่า แต่มีอัตราการสูญหายของแพ็กเก็ตต่ำกว่า Snort ทั้งสองสามารถตรวจจับภัยคุกคามได้ไม่ต่างกัน แต่ในงานวิจัย [5] ทดสอบกับการโจมตีด้วย DOS และ DDOS Snort 3.0 สามารถตรวจจับการโจมตีได้ดีกว่า Suricata แลกกับอัตราการใช้หน่วยประมวลผลที่สูงกว่า

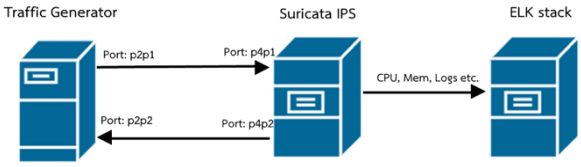
งานวิจัย [6] เน้นการทดสอบสมรรถนะ Suricata เพียงอย่างเดียว จุดประสงค์เพื่อวิเคราะห์หาพารามิเตอร์ที่ส่งผลต่อสมรรถนะโดยปรับแต่งเครื่องแม่ข่าย ให้สามารถรับข้อมูลจราจร HTTP/HTTPS ได้ถึง 34 Gbps โดยใช้ AF_PACKET ด้วยการ์ด Intel XL710 ขณะที่งานวิจัย [7] ปรับแต่ง Suricata โดยทดสอบกับข้อมูลจราจรจากไฟล์ pcap แล้วเปรียบเทียบเวลาที่ใช้การอ่าน พบว่าการเปรียบเทียบสายอักขระแบบ Aho-Corasick ซึ่งเป็นขั้นตอนวิธีพื้นฐานของ Suricata มีสมรรถนะไม่สูงมากนักเมื่อเทียบกับอัลกอริทึม Wu-Manber

งานวิจัย [8] และ [9] อธิบายลักษณะการทำงานของระบบป้องกันการบุกรุกและลักษณะเด่น ลักษณะด้อยของระบบป้องกันการบุกรุกแต่ละประเภท รวมถึงลักษณะเด่นและด้อย และทดสอบความแม่นยำในการตรวจจับภัยคุกคาม

การเพิ่มสมรรถนะของ Suricata อีกรูปแบบหนึ่งคือการจัดการคิวข้อมูลจราจร ซึ่งพบได้ในงานวิจัย [10] ซึ่งอธิบายการทำ Load balancing โดยใช้ RSS queue ของการ์ดเครือข่ายเพื่อกระจายข้อมูลแพ็กเก็ตที่ได้รับ ทั้งแบบ AF_PACKET ซึ่งใช้ raw socket และอีกวิธีคือ NFQ ใช้กฎของไฟร์วอลล์ในการส่งแพ็กเก็ต ผ่านสายการทำงานของ iptables ก่อนเข้าคิว แล้วส่งต่อให้ Suricata ประมวลผลซึ่งจะช่วยให้สมรรถนะการทำงานสูงขึ้น โดยสรุปแล้วพบว่าในสภาพแวดล้อมที่ใช้ใช้หน่วยประมวลผลกลางหลายแกน Suricata สามารถประมวลผลให้ได้สมรรถนะที่สูงกว่า การประเมินพารามิเตอร์สำหรับ Suricata ที่มีอย่างหลากหลายเป็นสิ่งที่งานวิจัยนี้จะนำเสนอเพิ่มเติมจากงานวิจัยก่อนหน้าที่ไม่ได้นำเสนอไว้ โดยแสดงข้อมูลเชิงเปรียบเทียบเมื่อปรับเปลี่ยนพารามิเตอร์เพื่อช่วยให้เห็นความแตกต่างจากการปรับแต่งพารามิเตอร์แต่ละค่าอย่างชัดเจน

3. วิธีการดำเนินงาน

การดำเนินงานแยกหัวข้อศึกษาออกเป็น 4 กลุ่ม รวม 6 การทดลองย่อยด้วยตัวแปรที่แตกต่างกัน รวมถึงเปรียบเทียบสมรรถนะการรับแพ็กเก็ตทั้ง 2 วิธีของระบบป้องกันการบุกรุก คือ AF_PACKET และ NFQ แพตฟอร์มการทดลองได้ติดตั้ง Suricata 5.0.2 ทำงานในภาวะ IPS ดังในรูปที่ 1 เครื่องแม่ข่ายนี้ทำงานติดตั้งระบบปฏิบัติการ Centos 7 เคอร์เนล 4.4.228-2.el7.elrepo.x86_64 ด้วย Intel Xeon E5-2643v3 จำนวน 2 ตัว แต่ละตัวมี 6 แกน (12 เธรด) มีหน่วยความจำขนาด 64GB ฮาร์ดดิสก์ขนาด 500GB และการ์ดเครือข่าย Intel 82599ES รองรับอัตราเร็ว 10Gbps โดยสามารถปรับแต่ง RSS queue และขนาดบัฟเฟอร์ได้



รูปที่ 1 แผนผังการเชื่อมต่อระหว่างเครื่องเซิร์ฟเวอร์

การทดลองแบ่งออกเป็น 4 หมวดหลัก ๆ ได้แก่

- **การทดลองที่ 1 และ 2** ทดลองเกี่ยวกับเธรดจัดการการทำงาน (เธรดจัดการเรื่องการไหลของข้อมูลจราจร เซสชัน เช่น ตำแหน่งและจำนวนของเธรดจัดการการทำงาน)
- **การทดลองที่ 3** ทดลองเกี่ยวกับการเลือกใช้งานเธรดทำงาน (เธรดสำหรับรับแพ็กเก็ต ตรวจสอบ และการแจ้งเตือน)
- **การทดลองที่ 4** ทดลองเกี่ยวกับกฎของ Suricata เพื่อเปรียบเทียบกฎที่คัดลอกซ้ำ และกฎที่เลือกจาก emerging-threat
- **การทดลองที่ 5 และ 6** ทดลองเกี่ยวกับอัตราเร็วข้อมูลจราจรที่รับได้

กฎที่ใช้ในการทดลองที่ 1-3 เลือกมาจาก emerging-threat [11] ซึ่งเป็นเซ็ตของกฎพื้นฐานที่นิยมใช้กับ Suricata และเปิดให้ดาวน์โหลดมาใช้งานได้ วิธีการทดลองจะเลือกกฎมาเพียง 1 กฎ แล้วคัดลอกซ้ำโดยเปลี่ยนเพียงเลขไอดีของกฎเพื่อให้ในแต่ละกฎมีความซับซ้อนเท่ากัน แต่กฎในการทดลองที่ต้องการ

วัดอัตราเร็วของข้อมูลจราจรที่รับได้ (การทดลองที่ 5 และ 6) ใช้ emerging-threat ทั้งชุดซึ่งประกอบด้วยกลุ่มกฎหลายด้าน เช่น การตรวจจับมัลแวร์ การตรวจจับช่องโหว่ การโจมตีแบบ DoS และอื่น ๆ รวมทั้งสิ้น 11037 กฎ

การทดลองที่ 1–6 เสนอข้อเปรียบเทียบระหว่างการรับแพ็กเก็ตแบบ AF_PACKET และ NFQ แต่มีการทดลองที่ทดลองเฉพาะวิธีการรับแพ็กเก็ตแบบใดแบบหนึ่ง คือการทดลองเพื่อหาอัตราเร็วแพ็กเก็ตที่รับได้ของ AF_PACKET และ NFQ

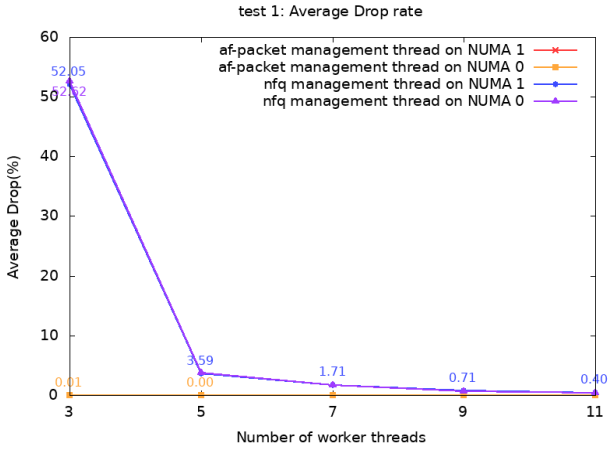
แต่ละการทดลองย่อยจะใช้ tcpreplay สร้างข้อมูลจราจรจากเครื่อง Traffic generator (ดังรูปที่ 1) tcpreplay จะนำข้อมูลจราจรในฟอร์แมต pcap ที่บันทึกไว้จากการใช้งานจริงในมหาวิทยาลัยเกษตรศาสตร์ ไฟล์ pcap มีขนาด 189 GB เมื่อ tcpreplay ในเครื่อง traffic generator สร้างข้อมูลจราจรจากไฟล์ pcap นี้และส่งผ่านอินเทอร์เฟซ 10GbE จะใช้เวลาประมาณ 6 นาที แต่ละการทดลองจะทดสอบ 7 ครั้ง (ได้ทดสอบซ้ำมากกว่านี้ พบว่าไม่มีความแตกต่างจึงเลือกจำนวนครั้งค่านี้) และหาค่าเฉลี่ยของอัตราการใช้งานหน่วยประมวลผล และอัตราการสูญหายของแพ็กเก็ตเพื่อใช้เปรียบเทียบสมรรถนะข้อมูลอัตราการใช้งานหน่วยประมวลผลวัดได้จากการติดตั้ง Telegraf [12] เพื่อส่งข้อมูลเก็บไว้ที่เครื่องที่ติดตั้ง ELK stack [13]

4. ผลการวิจัย

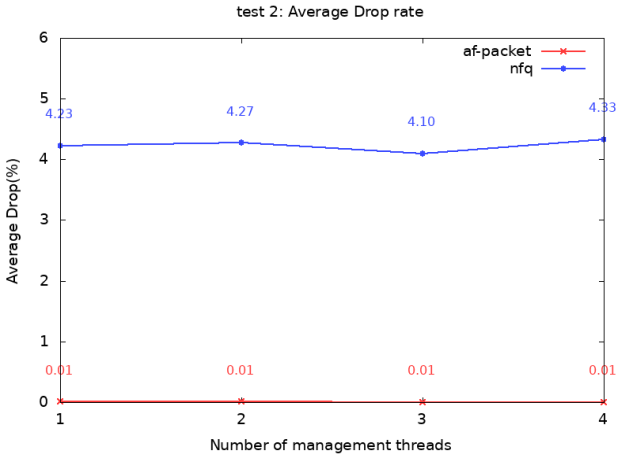
การทดลองที่ 1 เปรียบเทียบเมื่อให้เธรดจัดการการทำงาน วางตัวใน NUMA 0 และ NUMA 1 โดยเธรดทำงานบน NUMA 1 ซึ่งเป็น NUMA เดียวกับการ์ดเครือข่าย ทดลองด้วยอัตราเร็วของข้อมูลจราจรขนาด 4 Gbps ผลการทดลองในรูปที่ 2 พบว่าอัตราการสูญเสียของแพ็กเก็ตไม่ได้แตกต่างกันอย่างมีนัยยะสำคัญ ทั้งใน AF_PACKET และ NFQ ทำให้สามารถสรุปได้ว่า เธรดจัดการการทำงานไม่จำเป็นต้องอยู่ NUMA เดียวกับเธรดทำงาน

การทดลองที่ 2 เปรียบเทียบเมื่อให้จำนวนเธรดจัดการการทำงานเพิ่มขึ้น ขณะที่จำนวนเธรดทำงานมี 8 เธรด ทดลองด้วยอัตราเร็วของข้อมูลจราจรขนาด 4 Gbps ผลการทดลองดังรูปที่ 3 แสดงค่าอัตราการสูญเสียแพ็กเก็ต เมื่อให้เธรดจัดการการทำงานเพิ่มขึ้นไม่ได้แตกต่างกันอย่างมีนัยยะสำคัญ ได้ข้อสรุปว่าจำนวนของเธรดจัดการการทำงานไม่ได้ส่งผลต่อสมรรถนะ

การทดลองที่ 3 เปรียบเทียบเมื่อให้เธรดทำงาน อยู่ใน NUMA 0 และ NUMA 1 (NUMA ที่ต่ออยู่กับการ์ดเครือข่าย) ใช้ 1 เธรดจัดการการทำงาน ทดลองด้วยอัตราเร็ว 4 Gbps ผลการทดลองในรูปที่ 4 แสดงว่าตำแหน่งของเธรดทำงาน มีผลต่อสมรรถนะค่อนข้างมาก โดยเฉพาะอย่างยิ่งการรับแพ็กเก็ตแบบ NFQ ดังนั้นจึงสรุปได้ว่าเธรดทำงานควรทำงานอยู่ใน NUMA เดียวกับการ์ดเครือข่ายเพื่อให้อัตราการสูญเสียแพ็กเก็ตที่ต่ำกว่า



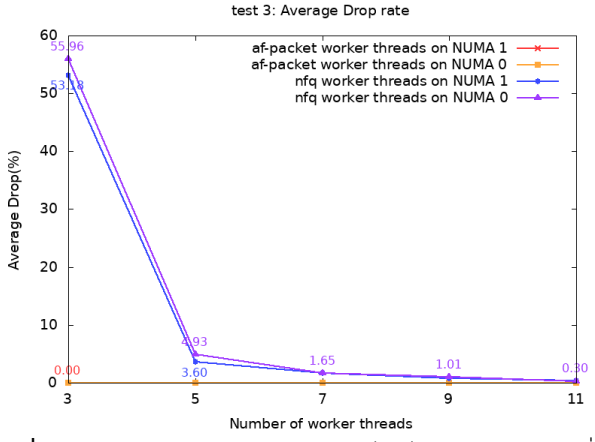
รูปที่ 2 กราฟอัตราการสูญเสียของแพ็กเก็ตจากการทดลองที่ 1



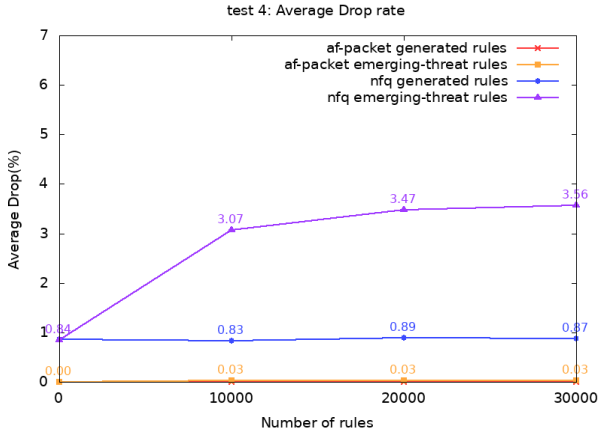
รูปที่ 3 กราฟอัตราการสูญเสียของแพ็กเก็ตจากการทดลองที่ 2

การทดลองที่ 4 เปรียบเทียบสมรรถนะเมื่อใช้กฎแบบคัดลอกซ้ำ กับกฎที่นำมาจาก emerging-threat โดยทดลองด้วยอัตราเร็ว 4 Gbps และเธรดทำงาน จำนวน 12 เธรดใน NUMA เดียวกัน ผลดังในรูปที่ 5 กฎที่ใช้จาก emerging-threat โดยแต่ละกฎไม่ซ้ำกัน มีอัตราการสูญหายของแพ็กเก็ตสูงกว่ากฎที่เลือก

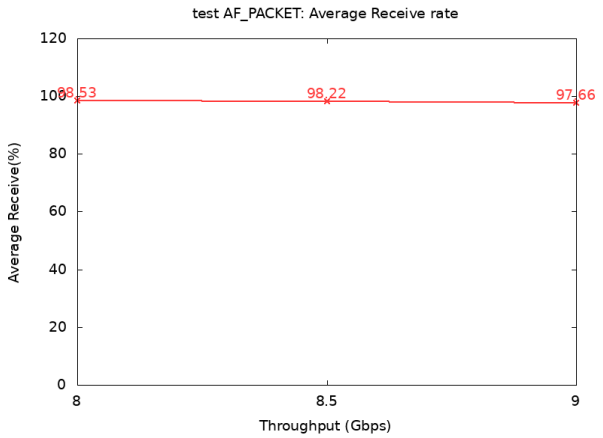
มาจาก emerging-threat แบบคัดลอกซ้ำอยู่ค่อนข้างมาก ซึ่งหมายถึงว่าลักษณะของกฎจะมีผลต่อสมรรถนะของ Suricata



รูปที่ 4 กราฟผลอัตราการสูญเสียของแพ็กเก็ตจากการทดลองที่ 3



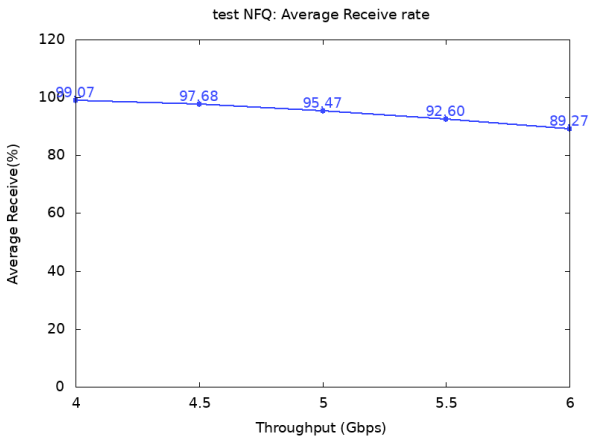
รูปที่ 5 กราฟผลอัตราการสูญเสียของแพ็กเก็ตจากการทดลองที่ 4



รูปที่ 6 กราฟอัตราที่รับได้ของแพ็กเก็ต จากการทดลองหาอัตราเร็วของข้อมูลจราจร ที่รับได้ของ AF_PACKET

การทดลองที่ 5 เป็นการวัดอัตราเร็วข้อมูลจราจรที่รับได้กรณีใช้ AF_PACKET เมื่อให้เธรดจัดการการทำงาน ทำงานบนแกนที่ 0 และเธรดทำงาน ทำงานบนแกนทั้งหมดใน NUMA 1 (12 แกน) ผลดังรูปที่ 6 สรุปได้ว่า AF_PACKET สามารถรับอัตราเร็วได้ 8.5 Gbps โดยรับแพ็กเก็ตได้ 98.22% หรือมีอัตราการสูญหายของแพ็กเก็ตต่ำกว่า 2% ดังนั้นการใช้ Suricata ในอินเทอร์เฟซ 10 GbE จึงทำงานได้ด้วยประสิทธิภาพในระดับดี แต่ยังคงมีอัตราสูญเสียแพ็กเก็ตอยู่ หากหน่วยประมวลผมีความเร็วสูงขึ้น อัตราเร็วข้อมูลอาจทำได้ใกล้ 10 Gbps และคาดว่ามีอัตราสูญเสียที่ต่ำกว่านี้

การทดลองที่ 6 เป็นการวัดอัตราเร็วข้อมูลจราจรที่รับได้ของ NFQ โดยกำหนดให้รับแพ็กเก็ตได้ไม่ต่ำกว่า 98% เพื่อประเมินว่าในอัตรานี้ NFQ สามารถรองรับความเร็วได้เท่าไร การจัดวางยังคงให้เธรดจัดการการทำงาน อยู่ในแกน 0 และเธรดทำงาน อยู่ในแกนทั้งหมดใน NUMA 1 จำนวน 12 แกน ดังรูปที่ 7 ผลที่ได้คือ NFQ สามารถรับอัตราเร็วของข้อมูลจราจรได้เพียง 4 Gbps โดยรับแพ็กเก็ตได้ 99.07% สูงกว่าข้อกำหนดเล็กน้อย จากผลการทดลองนี้จะเห็นได้อย่างชัดเจนว่า AF_PACKET มีสมรรถนะกว่า NFQ กว่า 2 เท่าตัว



รูปที่ 7 กราฟอัตราที่รับได้ของแพ็กเก็ตจากการทดลองหาอัตราเร็วของข้อมูลจราจรที่รับได้ของ NFQ

5. สรุปผลการวิจัย

จากการทดลองเพื่อหาค่าพารามิเตอร์ที่เหมาะสมของ Suricata เพื่อให้ทำงานได้อย่างมีประสิทธิภาพ ผลการทดลองที่ 1 ทดลอง

เกี่ยวกับเธรดที่ใช้จัดการการทำงาน สรุปได้ว่าตำแหน่งและจำนวนของเธรดจัดการการทำงานไม่ส่งผลต่อสมรรถนะของ Suricata เนื่องจากอัตราการสูญหายของแพ็กเก็ตก็ไม่ได้แตกต่างกันอย่างมีนัยสำคัญ ตัวแปรที่ส่งผลต่อสมรรถนะของ Suricata อย่างมากคือตำแหน่งและจำนวนของเธรดทำงาน ที่ได้จากการทดลองที่ 3 โดยเลือกใช้เธรดทำงานซึ่งควรอยู่ใน NUMA เดียวกับการ์ดเครือข่าย หากกำหนดให้ตำแหน่งของเธรดทำงานอยู่ต่าง NUMA กับการ์ดเครือข่าย มักทำให้เกิดการติดต่อข้าม NUMA และเกิดคอขวดซึ่งส่งผลให้สมรรถนะโดยรวมลดลง

นอกจากนี้จากผลการทดลองที่ 4 รูปแบบและจำนวนกฎที่ใช้ก็มีผลค่อนข้างมาก รวมไปถึงวิธีการรับแพ็กเก็ต ผลการทดลองพบว่า AF_PACKET สามารถรองรับอัตราเร็วได้สูงกว่า NFQ ประมาณ 2 เท่า เนื่องจากกลไกของ AF_PACKET ใช้หลักการเก็บในบัฟเฟอร์แบบวงแหวนซึ่งเชื่อมเข้ากับหน่วยความจำ จากนั้น Suricata จึงดึงแพ็กเก็ตจากบัฟเฟอร์ไปประมวลผล ต่างจาก NFQ ที่มีกลไกผ่านสายการทำงานของ iptables ก่อนแล้วจึงเข้าคิวให้ Suricata ดึงไปประมวลผล ทำให้ NFQ มีสมรรถนะด้อยกว่า AF_PACKET

6. กิตติกรรมประกาศ

ขอขอบคุณคุณพีระพงษ์ ทองภูเบศร์ และคุณสุรชัย จิตพินิจยล ที่ให้คำแนะนำเพิ่มเติม และให้ความรู้เรื่องการใช้งานเครื่องมือทดสอบ รวมไปถึงสำนักบริการคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ที่อนุเคราะห์ข้อมูลและการใช้เครือข่ายทดสอบระบบ

7. เอกสารอ้างอิง

[1] M. Chiang, S. Tu, W. Su and C. Lin, "Enhancing Inter-Node Process Migration for Load Balancing on Linux-Based NUMA Multicore Systems," In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 394-399, 2018.

[2] D. Jonathan and B. Burns, “A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines,” In *the 5th International Conference on Digital Society*, pp. 187-192, 2011.

[3] Q. Hu,S. Yu and M. R. Asghar, “Analysing performance issues of open-source intrusion detection systems in high-speed networks,” In *J. of Information Security and Applications*, vol.51, 2020.

[4] B. Brumen and J. Legvart, "Performance analysis of two open source intrusion detection systems," In the *39th International Convention on Information and Communication Technology*, pp. 1387-1392, 2016.

[5] D. Fadhilah and M. I. Marzuki, "Performance Analysis of IDS Snort and IDS Suricata with Many-Core Processor in Virtual Machines Against Dos/DDoS Attacks," In *the 2nd International Conference on Broadband Communications*, pp. 157-162, 2020.

[6] K. Jakimoski and N. V Singhai, “Improvement of Hardware Firewall’s Data Rates by Optimizing Suricata Performances,” In the *27th Telecommunica-tions forum TELFOR 2019*, pp. 1-4, 2019.

[7] W. Messer, Performance Testing Suricata: The Effect of Configuration Variables on Offline Suricata Performance, 2011. Available: https://redmine.openinfosecfoundation.org/attachments/download/706/Messer-Practicum-Final-v4.pdf [Accessed on: 13 May 2020].

[8] A. K. Saxena, S. Sinha and P. Shukla, "General study of intrusion detection system and survey of agent-based intrusion detection system," In *International Conference on Computing*, pp. 471-421, 2017.

[9] A. Garg and P. Maheshwari, "Performance analysis of Snort-based Intrusion Detection System," In *the 3rd International Conference on Advanced Computing and Communication Systems*, pp. 1-5, 2016.

[10] E. Leblond and G. Longo. "Suricata IDPS and its interaction with Linux kernel," In *netdev 1.1*, pp. 1-4, 2016.

[11] Proofpoint, Proofpoint Emerging Threats Rules, 2021. Available: https://rules.emergingthreats.net/open/suricata-5.0/rules/ [Accessed on: 14 January 2021].

[12] Telegraf 1.18 documentation, 2021. Available: https://docs.influxdata.com/telegraf/v1.18/ [Accessed on: 20 May 2021].

[13] Elastic, What is the ELK Stack? 2021. Available: https://www.elastic.co/what-is/elk-stack [Accessed on: 20 May 2021].