กรอบงานการควบคุมการเข้าถึงบนฐานบทบาทสำหรับตัวจัดการความมั่นคงไมโครเซอร์วิส

นายจิตติภัทร ผสมทรัพย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

Role-Based Access Control Framework for Microservice Security Manager

Mr. Chittipat Pasomsup

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computer Science
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2020

หัวข้อวิทยานิพนธ์	กรอบงานการควบคุมการเข้าถึงบนฐานบทบาทสำหรับตัวจัดการความมั่นคงไมโครเซอร์วิส
โดย	นายจิตติภัทร ผสมทรัพย์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์)

..... กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต)

จิตติภัทร ผสมทรัพย์ : กรอบงานการควบคุมการเข้าถึงบนฐานบทบาทสำหรับตัวจัดการความมั่นคงไมโครเซอร์วิส. (Role-Based Access Control Framework for Microservice Security Manager) อ.ที่ปรึกษาหลัก : รศ. ดร.ญาใจ ลิ่มปิยะกรณ์

ปัจจุบัน แพลตฟอร์มไมโครเซอร์วิสได้รับความนิยมเพื่อเปลี่ยนผ่านไปสู่ระบบไม่รวมศูนย์ที่มีความเร็วและกินพื้นที่น้อย อย่างไรก็ตาม จำนวนบริการที่เพิ่มมากขึ้นส่งผลให้เกิดความท้าทายในการบำรุงรักษาความมั่นคงของการควบคุมการเข้าถึง พื้นผิวการโจมตีที่มากขึ้นสามารถนำมาซึ่งความเสี่ยงในด้านความมั่นคงปลอดภัยและความเป็นส่วนตัวเข้าสู่ข้อมูลที่สำคัญและละเอียดอ่อน ดังนั้น ในงานนี้จึงได้นำเสนอแนวทางห่วงโซ่ของโดเมนที่เชื่อถือได้เพื่อแก้ไขปัญหาดังกล่าว แบบจำลองที่ขยายเพิ่มจากโมเดลการควบคุมการเข้าถึงตามบทบาทได้ถูกพัฒนาขึ้นเพื่อจัดการกับภัยคุกคามจากการเข้าถึงข้อมูลสารสนเทศที่สำคัญโดยไม่ได้รับอนุญาต รวมทั้งจัดการการตรวจสอบตัวตนในทุกสภาพแวดล้อมในโซลูชันคอนเทนเนอร์แอปพลิเคชัน ตัวจัดการความมั่นคงได้ถูกออกแบบบนพื้นฐานแบบจำลองอาร์แบ็กเพื่อพิสูจน์ตัวตน อนุญาต และระบุการควบคุมการเข้าถึงของผู้ใช้ผ่านเอพีไอเกตเวย์ ระบบต้นแบบที่ผสานรวมกับเซิร์ฟเวอร์การตรวจสอบสิทธิ์ได้ถูกพัฒนาขึ้นเพื่อการศึกษาเชิงประจักษ์ ผลลัพธ์รายงานว่าแนวทางดังกล่าวช่วยให้เข้าถึงข้อมูลสารสนเทศได้เร็วขึ้นและยืดหยุ่นมากขึ้น นอกจากนี้ ยังช่วยปรับปรุงเวลาในการตอบสนองต่อเหตุการณ์ให้ดีขึ้น

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2563 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6270034221 : MAJOR COMPUTER SCIENCE

KEYWORD: Role-Based Access Control, Identity Management, Microservice, IT Security

Chittipat Pasomsup : Role-Based Access Control Framework for Microservice Security Manager. Advisor: Assoc. Prof. Yachai Limpiyakorn, Ph.D.

For transitioning to a decentralized system, a microservices platform has become popular in today software development due to its lightweight mechanisms. However, increasing the number of services results in a challenge to maintain the security of access control. The more attack surfaces can bring security and privacy risk via sensitive data. Therefore, a chain of trust domains was introduced to solve this problem in this work. The extended Role-Based Access Control model for microservice security managers is implemented for managing threats of unauthorized access to sensitive information and identity verification across all environments in application container solutions. A RBAC-based security manager is designed to authenticate, authorize, and identify user's access control via API-Gateway. A prototype system integrated with authentication server is implemented for empirical study. The results report that the approach provides faster and more flexible access to information in addition to improving incident response time.

Field of Study: Computer Science Student's Signature .....

Academic Year: 2020 Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความอนุเคราะห์อย่างดียิ่งของรองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งท่านได้สละเวลาให้ความรู้ ให้คำปรึกษาตรวจสอบ ให้คำแนะนำแนวทางการวิจัย และสนับสนุนจนทำให้การวิจัยในครั้งนี้สำเร็จออกมาด้วยดี ข้าพเจ้าจึงขอกราบระลึกถึงพระคุณของรองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ ไว้ ณ ที่นี้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ ประธานกรรมการสอบวิทยานิพนธ์ และ อาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต กรรมการสอบวิทยานิพนธ์ ที่กรุณาเสียสละเวลา ให้คำแนะนำติชม และตรวจสอบวิทยานิพนธ์ฉบับนี้

ท้ายที่สุด ผู้จัดทำวิทยานิพนธ์ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ คุณภรรยา และครอบครัวอันเป็นที่รักสำหรับกำลังใจที่มีค่ายิ่ง รวมถึงขอขอบพระคุณผู้บังคับบัญชาในสายงาน เพื่อนร่วมงาน และมิตรสหายคอยติดตามให้กำลังใจ ให้การสนับสนุนและความช่วยเหลือในด้านต่างๆ ตลอดการทำงานงานนี้ และขอบพระคุณท่านที่มิได้กล่าวถึงไว้ ณ ที่นี้ที่มีส่วนช่วยให้วิทยานิพนธ์ของข้าพเจ้าสำเร็จไปได้ด้วยดี

จิตติภัทร ผสมทรัพย์

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ปัจจุบันการพัฒนาด้านเทคโนโลยีการสื่อสารภายในองค์กรเป็นไปอย่างรวดเร็วด้วยการเพิ่มขึ้นของอุปกรณ์การใช้งาน ทำให้มีข้อมูลที่อยู่ในระบบเครือข่ายเป็นจำนวนมาก ทั้งนี้ข้อมูลที่ถูกจัดเก็บมีหลากหลายประเภท ทั้งข้อมูลที่เปิดเผยได้ และข้อมูลที่เปิดเผยไม่ได้ ซึ่งการควบคุมการเข้าถึงทรัพยากรเป็นเรื่องสำคัญ และมักจะเป็นปัญหาที่ยุ่งยาก โดยเฉพาะอย่างยิ่งสำหรับระบบที่มีขนาดใหญ่ และมีความซับซ้อนสูง ซึ่งมักมีระบบจัดเก็บข้อมูลแบบรวมศูนย์ เป็นสถาปัตยกรรมแบบเดิมที่เรียกว่า สถาปัตยกรรมแบบโมโนลิทิก (Monolithic Architecture) แต่มีการเสนอแนวทางใหม่ คือ สถาปัตยกรรมไมโครเซอร์วิส (Microservice Architecture) ที่มีความสามารถในการขยาย (Scalability) และลดความซับซ้อน โดยแบ่งตัวบริการ (Service) เป็นส่วนย่อย ๆ ที่ทำให้ง่ายต่อการจัดการ เพิ่มความรวดเร็วในการพัฒนางาน และความต่อเนื่องของการส่งมอบงาน [1] แต่ปัญหาที่ตามมาคือ การควบคุม จัดการความมั่นคงของตัวบริการที่มากนั่นเอง ดังนั้นจึงได้มีการศึกษาการควบคุมการเข้าถึงแบบต่าง ๆ ขึ้น หนึ่งในนั้นที่เป็นที่นิยม คือ การควบคุมการเข้าถึงแบบบนฐานบทบาท หรือ Role-Based Access Control (RBAC) ที่เป็นตัวกำหนดสิทธิ์ผู้ใช้งานตามหน้าที่รับผิดที่ถูกมอบหมายไว้ โดยสามารถจัดการยืนยันตัวบุคคล และตรวจสอบสิทธิ์การเข้าถึงข้อมูลที่มีความอ่อนไหวประเภทต่าง ๆ ได้ดี เพื่อที่จะป้องกันและควบคุมผู้ที่ไม่เกี่ยวข้องให้เข้าถึงข้อมูลโดยไม่จำเป็นได้

วิทยานิพนธ์นี้นำเสนอกรอบงานการควบคุมการเข้าถึงบนฐานบทบาทที่อ้างตามโมเดล RBAC (ANSI, 2004) [2] สำหรับตัวจัดการความมั่นคงไมโครเซอร์วิสเพื่อช่วยเพิ่มความมั่นคงในการจัดการเอกลักษณ์และสิทธิ์การใช้งานในระบบทั้งหมด ซึ่งจะช่วยให้การเข้าถึงข้อมูลมีความยืดหยุ่นมากขึ้น และยังสามารถตอบสนองต่อเหตุการณ์ได้รวดเร็วมากขึ้นด้วย โดยมีการศึกษาระบบยืนยันตัวตนแบบรวมศูนย์โดยใช้ RBAC เป็นต้นแบบมาแล้ว เช่น MTAS [3] ที่ใช้การอนุญาตบริการแบบห่วงโซ่การเรียกใช้แบบประสานงาน แต่ทางผู้วิจัยเล็งเห็นว่า ปัจจุบันระบบแบบไมโครเซอร์วิสมีข้อได้เปรียบในการดำเนินงานที่มากกว่า จึงมีแนวคิดที่จะนำข้อได้เปรียบดังกล่าวมาต่อยอดตัวจัดการความมั่นคง (Security Manager) ตามสถาปัตยกรรมไมโครเซอร์วิสที่ใช้แพร่หลายในขณะนี้

1.2 วัตถุประสงค์ของงานวิจัย

เพื่อสร้างตัวจัดการความมั่นคงสำหรับการควบคุมการเข้าถึงข้อมูลและทรัพยากร บนแพลตฟอร์มไมโครเซอร์วิสได้อย่างถูกต้องและตอบสนองต่อเหตุการณ์ได้รวดเร็วมากขึ้น

1.3 ขอบเขตการวิจัย

1. แอปพลิเคชันก่อนที่จะนำมาใช้งานกับระบบจัดการความปลอดภัยจะมีการยืนยันตัวบุคคลด้วยชื่อผู้ใช้และรหัสผ่าน มีการกำหนดสิทธิเข้าใช้งานด้วย Rule Based Policy
2. การทำงานของระบบทั้งหมดจะทำงานอยู่ภายใต้สภาพแวดล้อมที่สมบูรณ์ โดยไม่มีความผิดพลาดของสภาพแวดล้อมเกิดขึ้น
3. ใช้สำหรับการจัดการเอกลักษณ์ผู้ใช้งาน ยืนยันตัวบุคคลและควบคุมการเข้าถึงข้อมูลเพื่อตรวจสอบสิทธิตามบทบาทที่มี

1.4 ขั้นตอนการวิจัย

1. ศึกษาค้นคว้าทฤษฎีและงานวิจัยที่เกี่ยวข้องกับกระบวนการและโมเดลการตรวจสอบการเข้าถึงบนฐานบทบาท (RBAC) มาเป็นตัวกำหนด เพื่อสร้างโมเดลสำหรับตัวจัดการความมั่นคงไมโครเซอร์วิส
2. การออกแบบโมเดลการตรวจสอบการเข้าถึงบนฐานบทบาทสำหรับตัวจัดการความมั่นคงไมโครเซอร์วิสโดยอ้างอิงโมเดลของ RBAC ตามมาตรฐาน (ANSI INCITS)
3. พัฒนาระบบตามโมเดลข้อ 2 โดยการ Implement โมเดลดังกล่าว ด้วยการใช้ Node.js ผสานกับ XACML 3.0 บนแอปพลิเคชันคอนเทนเนอร์ Docker
4. ตรวจสอบความถูกต้องของระบบที่พัฒนา โดยตั้งกรณีทดสอบที่ครอบคลุมการใช้งานทั้งหมดและแสดงการใช้งานตัวอย่าง 2 กรณี กรณีปกติในการตรวจสอบสิทธิ และกรณีของการบริการแบบประสานงานของแต่ละโมโครเซอร์วิส
5. วิเคราะห์ในเชิงการเปรียบเทียบ เรื่องประสิทธิภาพการทำงานของระบบทั้งหมด เวลาการตอบสนอง ค่าภาระการรองรับการโหลด และอภิปรายผลที่ได้
6. สรุปผลและประเมินผลการดำเนินงาน
7. เรียบเรียงและจัดทำบทความวิชาการ

8. เรียบเรียงและจัดทำเล่มวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้โมเดลที่รองรับการจัดการเอกลักษณ์ผู้ใช้งานบนแพลตฟอร์มไมโครเซอร์วิสในสภาพแวดล้อมที่ต่างกัน
2. ได้โมเดลที่รองรับการกำหนดสิทธิ์ควบคุมและการตรวจสอบการเข้าถึงบนฐานบทบาทของผู้ใช้งานที่ถูกกำหนดขึ้นตามนโนบายที่มี

1.6 ลำดับการจัดเรียงเนื้อหาในวิทยานิพนธ์

วิทยานิพนธ์นี้มีทั้งหมด 6 บท ดังต่อไปนี้ บทที่ 1 บทนำ ความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของการทำวิจัย ขอบเขตการทำวิจัย ประโยชน์ที่คาดว่าจะได้รับและผลงานตีพิมพ์ บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง บทที่ 3 วิธีการดำเนินการวิจัย บทที่ 4 การออกแบบและพัฒนาระบบตามแนวทางการวิจัยที่นำเสนอ บทที่ 5 การทดสอบและวิเคราะห์ผลการทดลอง และบทที่ 6 สรุปผลการวิจัย ข้อเสนอแนะและแนวทางสำหรับอนาคต

1.7 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์ได้รับการตีพิมพ์บทความวิชาการ C. Pasomsup and Y. Limpiyakorn, " HT-RBAC: A Design of Role-based Access Control Model for Microservice Security Manager" ในรายงานประชุมวิชาการนานาชาติสืบเนื่องจาก ACM 2021 2nd International Conference on Information System and System Management (ISSM 2021), Aug 12-14, 2021, Guizhou, China, pp. 1-5.

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1. โครงสร้างกุญแจสาธารณะ (Public Key Infrastructure)

โครงสร้างกุญแจสาธารณะ (Public Key Infrastructure–PKI) เป็นสิ่งที่ช่วยให้ผู้ใช้ที่อยู่ในเครือข่ายที่ไม่มีความปลอดภัย เช่น อินเทอร์เน็ตให้สามารถสร้างความปลอดภัยในการใช้งานเครือข่ายที่อาจจะมีการส่งข้อมูลหรือแลกเปลี่ยนข้อมูลทางด้านการเงินขึ้นมาได้ โดยผ่านการใช้การเข้ารหัสแบบที่มีการใช้คู่กุญแจแบบสาธารณะ และแบบส่วนตัว ที่ได้รับมาจากผู้มีอำนาจที่ไว้วางใจ (Trusted Authority) โครงสร้างกุญแจสาธารณะสามารถนำมาใช้ในการสร้างลายเซ็นอิเล็กทรอนิกส์ (Digital Signature) ที่สามารถใช้เพื่อระบุตัวตนของบุคคลหรือองค์กรได้ โดยโครงสร้างกุญแจสาธารณะมีคุณสมบัติที่ใช้ในการสร้างความปลอดภัยดังนี้

1. การยืนยันตัวบุคคล (Authentication)
2. การรักษาความลับของข้อมูล (Confidential)
3. การรักษาความครบถ้วนถูกต้องของข้อมูล (Integrity)
4. การไม่สามารถปฏิเสธความรับผิดชอบได้ (Non-Repudiation)

โครงสร้างกุญแจสาธารณะประกอบด้วยส่วนต่างๆ ดังนี้

1. Certification Authority (CA) ทำหน้าที่ในการออกใบรับรองอิเล็กทรอนิกส์ และตรวจสอบใบรับรองอิเล็กทรอนิกส์ โดยใบรับรองอิเล็กทรอนิกส์จะมีข้อมูลในส่วนของกุญแจสาธารณะและข้อมูลเกี่ยวกับกุญแจสาธารณะ

2. Registration Authority (RA) ทำหน้าที่เป็นผู้ตรวจสอบคำขอใบรับรองอิเล็กทรอนิกส์ก่อนที่จะให้ CA ทำการออกใบรับรองอิเล็กทรอนิกส์

3. บริการไดเรกทอรี ทำหน้าที่ในการเก็บใบรับรองอิเล็กทรอนิกส์และรายชื่อใบรับรองอิเล็กทรอนิกส์ที่ถูกยกเลิก

4. ระบบบริหารจัดการใบรับรองอิเล็กทรอนิกส์

ส่วนหลักการทำงานของการเข้ารหัส/ถอดรหัสโดยใช้กุญแจสาธารณะและกุญแจส่วนตัวคือการสร้างคู่กุญแจสาธารณะและกุญแจส่วนตัวโดยใช้อัลกอริทึมอย่างเช่น RSA, DSA หรือ Elliptic Curve เป็นต้น โดยถ้าใช้กุญแจดอกใดดอกหนึ่งเข้ารหัสไว้ ก็จำเป็นจะต้องใช้กุญแจที่เป็นคู่กันอีกดอกหนึ่งถอดรหัสเท่านั้น ซึ่งโดยปกติแล้วกุญแจสาธารณะนั้นจะเปิดเผยให้กับผู้อื่นได้ทราบอยู่แล้วซึ่งจะ

อยู่ในรูปของใบรับรองอิเล็กทรอนิกส์ ที่จะเก็บอยู่ในบริการไดเรกทอรี แต่กุญแจส่วนตัวจะอยู่ที่เจ้าของเพียงผู้เดียวเท่านั้น ความสัมพันธ์สรุปดังแสดงในตารางที่ 1 ซึ่งอธิบายว่าเหตุการณ์ใด ควรจะใช้กุญแจของใคร และประเภทใด ในการเข้ารหัสหรือถอดรหัส

ตารางที่ 1 ความสัมพันธ์ของเหตุการณ์และประเภทกุญแจที่ใช้ในการเข้ารหัสหรือถอดรหัส

เหตุการณ์	**ร้องขอกุญแจ**	**ประเภทของกุญแจ**
เมื่อต้องการจะส่งข้อความที่เข้ารหัสไว้	ผู้รับ	กุญแจสาธารณะ
เมื่อต้องการจะส่งข้อความที่มีลายเซ็น	ผู้ส่ง	กุญแจส่วนตัว
เมื่อต้องการจะถอดรหัสของข้อความที่เข้ารหัสไว้	ผู้รับ	กุญแจส่วนตัว
เมื่อต้องการจะถอดรหัสของข้อความที่มีลายเซ็น (ใช้ในการยืนยันตัวบุคคลของผู้ส่งได้)	ผู้ส่ง	กุญแจสาธารณะ

2.1.2. การกำหนดสิทธิเข้าใช้งาน (Authorization)

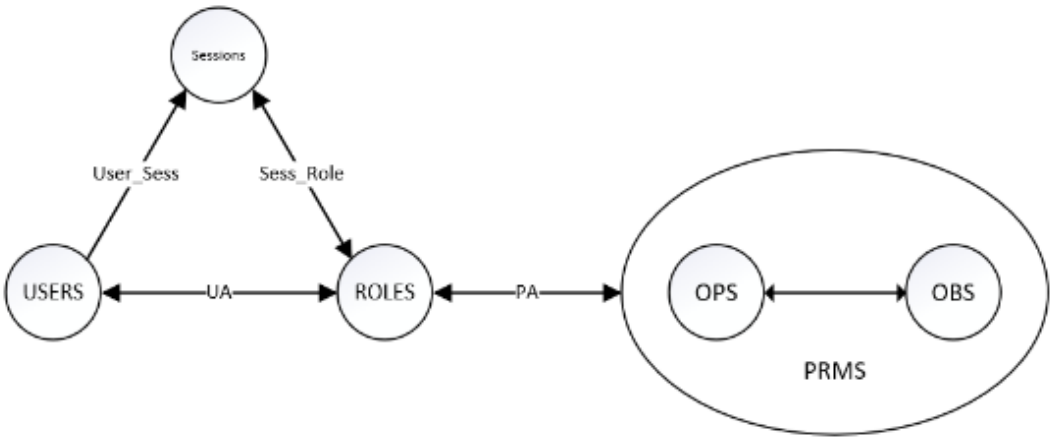
การกำหนดสิทธิเข้าใช้งานคือกระบวนการในการกำหนดสิทธิเข้าใช้งานทรัพยากรที่อาจจะอยู่ในรูปแบบไฟล์ แฟ้มข้อมูล หรือแอปพลิเคชัน เป็นต้น ซึ่งสามารถแบ่งได้เป็น 3 ประเภทหลักๆ ได้แก่

1. ประเภทแบ่งตามการตัดสินใจ หรือ Discretionary Access Control (DAC) คือการควบคุมการเข้าถึงโดยเจ้าของทรัพยากรสามารถกำหนดสิทธิให้ผู้อื่นหรือกลุ่มต่างๆได้เอง ตัวอย่างเช่น ระบบไฟล์ของ UNIX ที่ทุก ๆ ไฟล์ในระบบจะมีเจ้าของ และเจ้าของจะสามารถกำหนดสิทธิให้กับผู้ใช้คนอื่นๆได้ นอกจากนี้ผู้ใช้ใน DAC ยังสามารถมอบหมายการควบคุมของไฟล์ให้ผู้ใช้คนอื่นได้อีกด้วย และ DAC ยังมีการใช้กันอย่างกว้างขวางในการบังคับนโยบายการเข้าถึงในระบบปฏิบัติการต่างๆ

2. ประเภทแบ่งตามการรับมอบอำนาจ หรือ Mandatory Access Control (MAC) คือ การกำหนดสิทธิในลักษณะของการติดป้ายบอกระดับความปลอดภัยของทรัพยากรไว้ โดยผู้ใช้จะสามารถอ่านข้อมูลของทรัพยากรได้ก็ต่อเมื่อผู้ใช้มีระดับความปลอดภัยมากกว่าหรือเท่ากับระดับความปลอดภัยของทรัพยากร และผู้ใช้จะสามารถเขียนข้อมูลของทรัพยากรได้ก็ต่อเมื่อผู้ใช้มีระดับความปลอดภัยน้อยกว่าหรือเท่ากับระดับความปลอดภัยของทรัพยากร ซึ่ง MAC จะมีการใช้ในระบบที่เกี่ยวข้องกับข้อมูลที่มีความสำคัญเป็นอย่างมาก เช่นในวงการทหาร เป็นต้น

3. ประเภทแบ่งตามบทบาทหน้าที่ หรือ Role Based Access Control (RBAC) จะมีส่วนประกอบ 3 ส่วนคือ ผู้ใช้ (User) บทบาท (Role) และสิทธิ (Permission) ซึ่ง RBAC ไม่ได้กำหนดสิทธิให้กับผู้ใช้โดยตรง แต่จะกำหนดสิทธิให้กับบทบาทว่าสามารถมีสิทธิใดบ้าง โดยผู้ใช้จะถูกกำหนดบทบาทให้อย่างน้อย 1 บทบาท หรือมากกว่า โดยบทบาทสามารถสืบทอดได้ (inherited) และให้ส่วนของการกำหนดสิทธิสามารถทำการเปลี่ยนแปลงได้ง่าย โดยจะส่งผลให้กับผู้ใช้ทุกคนภายใต้บทบาทที่ได้ทำการเปลี่ยนแปลง ลักษณะการทำงานของ Role Based Access Control ดังรูปที่ 1

เมื่อพิจารณาถึงความเหมาะสมของประเภทการกำหนดสิทธิเข้าใช้งาน ผู้วิจัยมีความเห็นว่างานวิจัยนี้มีความเหมาะสมกับการกำหนดสิทธิเข้าใช้งานประเภทแบ่งตามบทบาทหน้าที่ เนื่องจากสามารถบริหารจัดการในเรื่องของการกำหนดสิทธิให้ผู้ใช้ได้ง่ายกว่าประเภทอื่น จึงเหมาะกับระบบที่สนับสนุนผู้ใช้จำนวนมาก อีกทั้งยังเป็นการควบคุมการเข้าถึง (Access Control) ที่แพร่หลายและเป็นที่ยอมรับในวงกว้าง โดยมาตรฐานที่ใช้อ้างอิงในงานวิจัยนี้จากหน่วยงาน American National Standard for Information Technology (ANSI) ซึ่งได้อธิบายองค์ประกอบ และความสัมพันธ์ต่าง ๆ ของโมเดล RBAC รวมทั้งมีการกำหนดลำดับชั้นของบทบาท และเพิ่มข้อจำกัดการใช้งานเพื่อไม่ให้โมเดลเกิดความขัดแย้งกันเองในการกำหนดบทบาทแต่ละแบบ และเมื่อสภาพแวดล้อมของการใช้งานมีความซับซ้อนและความหลากหลายของทรัพยากรที่ต้องจัดการมากขึ้น ทำให้มีการจำแนกโมเดลออกเป็น 3 องค์ประกอบหลัก คือ ส่วนหลัก (Core) ลำดับชั้น (Hierarchy) และเงื่อนไขบังคับ (Constraint)

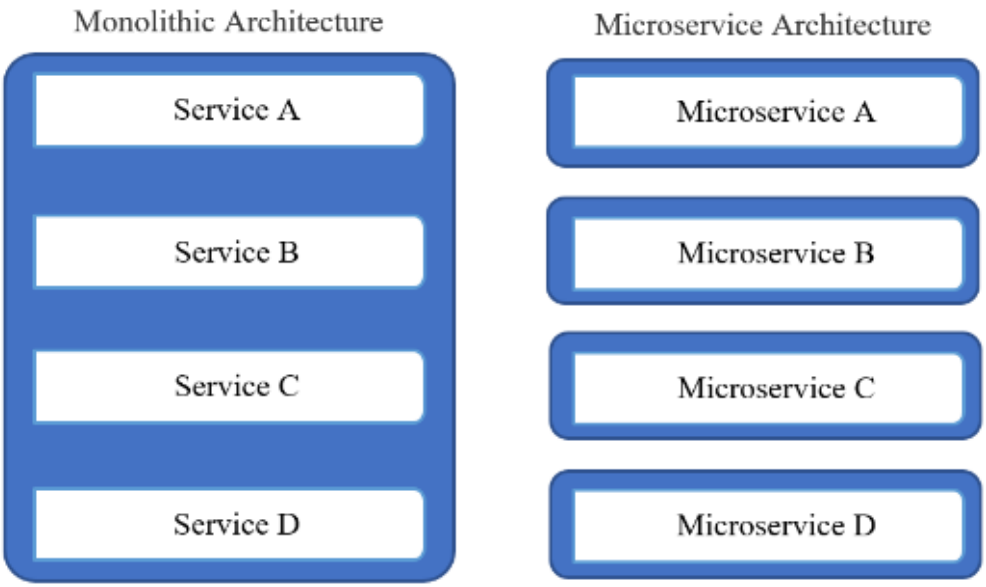


รูปที่ 1 ลักษณะการทำงานของ Role Based Access Control [2]

2.1.3. สถาปัตยกรรมไมโครเซอร์วิส (Microservice Architecture)

คำจำกัดความของสถาปัตยกรรมไมโครเซอร์วิส คือ กระบวนการแยกบริการ (service) ที่มีอยู่ในสถาปัตยกรรมแบบโมโนลิทิกเดิม ออกเป็นชุดของบริการอิสระซึ่งเป็นโครงสร้างแบบแยกส่วน

เมื่อเปรียบเทียบกับการออกแบบเดิมแล้ว การเปลี่ยนแปลงหรือปรับปรุงแอปพลิเคชันจำเป็นต้องสร้างและแก้ไขกับระบบทั้งหมดใหม่ ด้วยเหตุนี้การบำรุงรักษาจึงทำได้ยากและไม่มีประสิทธิภาพ ทำให้สถาปัตยกรรมไมโครเซอร์วิสถูกเสนอเข้ามาแทนที่ ซึ่งเป็นระบบแบบ Service Oriented Architecture (SOA) สื่อสารผ่านโปรโตคอล HTTP และรูปแบบ JSON ผ่าน SSL / TLS โดยเข้ารหัสข้อมูลเพื่อความมั่นคงมากขึ้น ซึ่งแต่ละเซอร์วิสจะมี API เอาไว้ให้เรียกใช้งาน โดยที่ REST API จะถูกนำมาใช้สำหรับติดต่อระหว่างบริการ และมี HTTP Method เป็นชุดคำสั่งสำหรับการดำเนินการ นอกจากนี้ยังช่วยในการจัดส่งที่รวดเร็วและมีขนาดข้อมูลที่น้อยลง ยิ่งไปกว่านั้นการผสมผสานระหว่างการพัฒนาที่คล่องตัว (Agile) และความสามารถในการปรับขนาดที่เพิ่มความเร็วการติดต่อที่สูงขึ้น ทำให้การพัฒนาและปรับปรุงแอปพริเคชันให้เร็วขึ้น ส่งผลโดยตรงต่อการส่งมอบแอปพลิเคชันและบริการที่ดีขึ้น โดยความแตกต่างของสถาปัตยกรรมทั้ง 2 แบบ ดังแสดงในรูปที่ 2

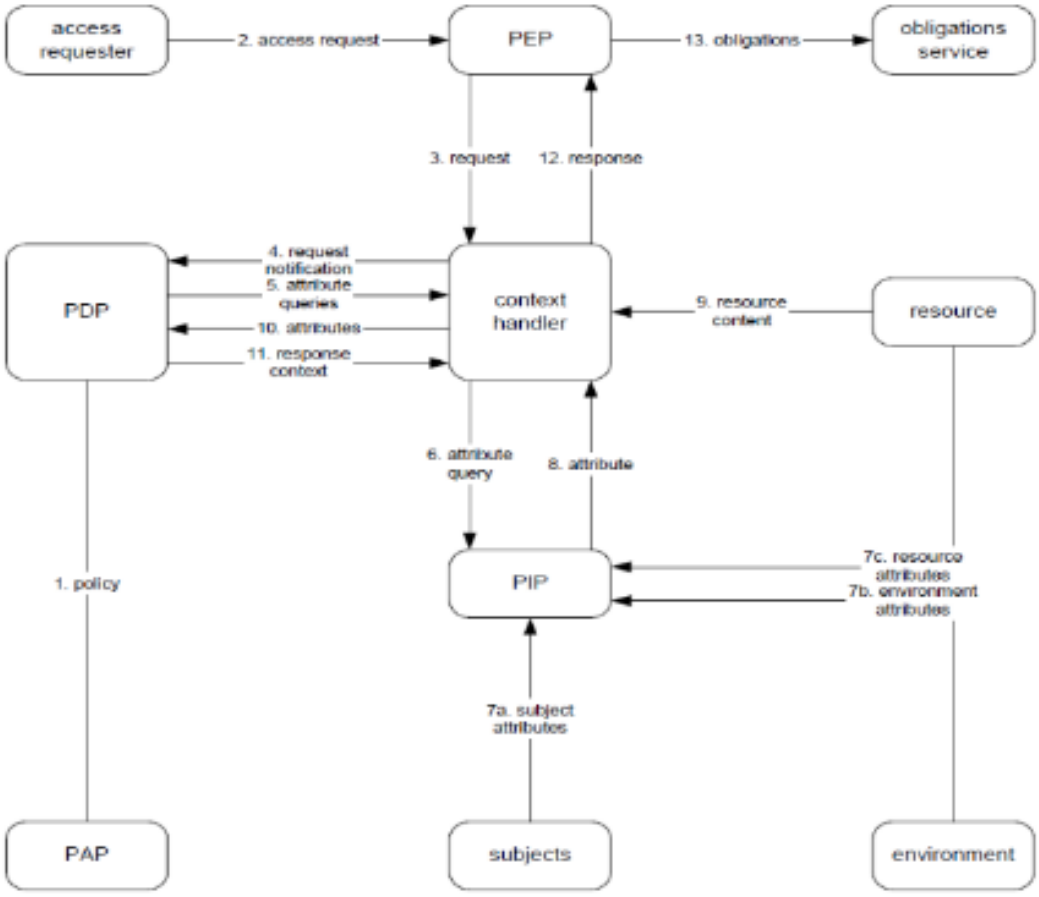


รูปที่ 2 ความแตกต่างของสถาปัตยกรรมแบบโมโนลิทิกและไมโครเซอร์วิส [4]

2.1.4. eXtensisible Access Control Markup Language (XACML)

XACML [5] เป็นมาตรฐานที่กำหนดขึ้นเพื่อใช้ในการกำหนดนโยบายในการควบคุมการเข้าถึงด้วยการใช้ XML เป็นภาษาพื้นฐานมาประกอบกันขึ้นเรียกว่า นโยบาย (Policy) เพื่อใช้ในการกำหนด Attribute-Based Access Control (ABAC) ซึ่งสามารถนำมาใช้ใน Role-Based Access Control (RBAC) ได้ด้วย ทำนองเดียวกัน ได้ถูกนำมาใช้งานเป็นกระบวนการทำงานประกอบขึ้นตามรูปที่ 3 โดยมีองค์ประกอบหลัก ได้แก่ ผู้ร้องขอในการตรวจสอบสิทธิ์การเข้าถึง ส่วนนโยบายที่ไว้บังคับเพื่อให้มีการตรวจสอบสิทธิ์การเข้าถึง และกระบวนการการตัดสินใจสำหรับการร้องขอ เป็นต้น และมี

โครงสร้างหลักที่ประกอบด้วย คือ Rule Policy และ Policy Set ในขั้นตอนการทำงานของ XACML สามารถสรุปเป็นขั้นตอนได้ดังนี้ คือ เมื่อมีการร้องขอเพื่อตรวจสอบการเข้าถึง จะถูกส่งไปให้ตัดสินใจจากนโยบายที่ถูกสร้างไว้ จากนั้นจึงส่งผลของการตัดสินใจไปให้กับผู้ร้องขอ ซึ่งผลลัพธ์ ได้แก่ มีสิทธิ์ ไม่มีสิทธิ์ ไม่สามารถตัดสินใจได้ เป็นต้น



รูปที่ 3 แผงผังการไหลผ่านของข้อมูลในการทำงานของ XACML [5]

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยและบทความวิชาการที่เกี่ยวข้องกับวิทยานิพนธ์ฉบับนี้ ประกอบด้วยงานวิจัยที่เกี่ยวกับการสร้างโมเดลการควบคุมการเข้าถึงบนฐานบทบาท งานวิจัยที่เกี่ยวกับการยืนยันตัวบุคคล งานวิจัยที่เกี่ยวกับการประยุกต์ใช้ระบบหลายตัวแทน และงานวิจัยที่เกี่ยวกับการประยุกต์ใช้ XACML

2.2.1. RBAC+: Dynamic Access Control for RBAC-administered web-based Databases [6]

คณะผู้วิจัยนำเสนอโมเดล RBAC+ ที่มีความสามารถแบบไดนามิก เป็นโมเดลควบคุมการเข้าถึงบนฐานบทบาทเพื่อบังคับใช้การควบคุมการเข้าถึงแบบละเอียด โดยอธิบายด้วย Z notation

ทำให้เห็นขั้นตอนการดำเนินงานที่ชัดเจน สามารถใช้กับแอปพลิเคชันฐานข้อมูลบนเว็บ ลดปัญหาการควบคุมการเข้าถึงที่ไม่เกี่ยวข้องกับข้อมูลและตัวตนของผู้ใช้ได้ โดยสามารถกำหนดบทบาทตามลักษณะแอปพลิเคชัน โปรไฟล์ของแอปพลิเคชัน และเซสชันแอปพลิเคชันย่อยได้ดี อีกทั้งยังช่วยเพิ่มความสามารถในการตรวจจับอันตราย ธุรกรรม (Transaction) ที่สาเหตุสำคัญที่ทำลายฐานข้อมูลได้ ทำให้การโจมตีทางธุรกรรมที่เป็นอันตรายสามารถถูกตรวจพบและยกเลิกได้ทันเวลา นับเป็นตัวช่วยในระบบรักษาความมั่นคงทางไซเบอร์ได้ และได้ยกตัวอย่างการใช้งานกับระบบ DBMS จากเว็บการจัดการคอร์สออนไลน์ ที่ช่วยบันทึกทั้งชื่อคอสทั้งหมด รวมถึงฐานข้อมูลที่เกี่ยงข้อง ที่ช่วยลดความซับซ้อน และความเสี่ยงการเข้าถึงข้อมูลที่ไม่เกี่ยวข้อง ซึ่งระบุผ่านโปรไฟล์ของแอปพลิเคชันนั่นเอง

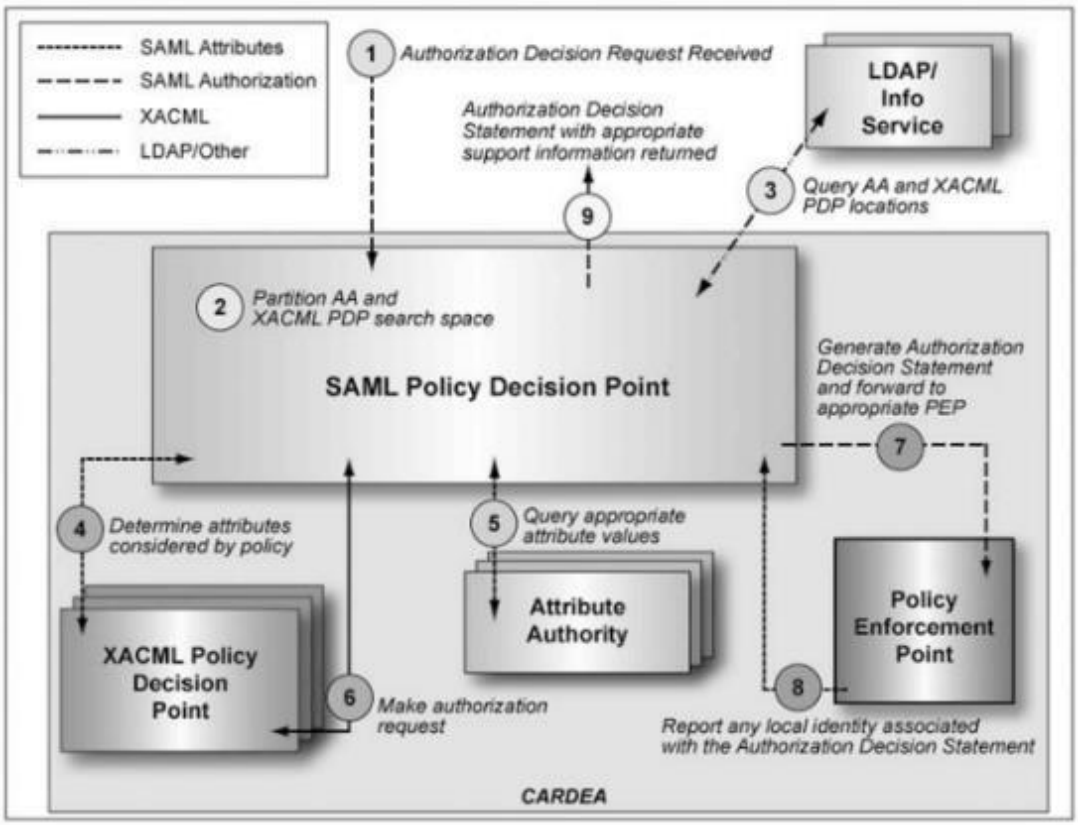
2.2.2. A development of multi-SSO authentication and RBAC model in the distributed systems [7]

งานวิจัยนี้ได้เสนอการออกแบบและการพัฒนาของการตรวจสอบสิทธิ์สองปัจจัย หรือที่เรียกว่าระบบ Single Sign-On (SSO) และ โมเดล RBAC ชนิดไดนามิก ของระบบการอนุญาตในหลายแอปพลิเคชันและหลายโดเมน ซึ่งจะใช้การรับรองความถูกต้องและการอนุญาตบนใบรับรองกุญแจสาธารณะแบบ X.509 และโครงสร้างพื้นฐานการจัดการสิทธิ์ หรือ Privilege Management Infrastructure (PMI) ในแบบจำลองนี้ใช้การรองรับการแลกเปลี่ยนพิสูจน์ตัวตนและการอนุญาตด้วย Security Assertion Markup Language (SAML) เพื่อที่จะได้สามารถลงชื่อเข้าใช้งานเพียงครั้งเดียว (SSO) เพื่อให้ง่ายต่อจัดการและขยายขนาดได้มากขึ้น นับเป็นระบบการจัดการแบบกระจายตัวที่ให้ความมั่นคงที่ดีและมีความยืดหยุ่น โดยผ่านการใช้งานระบบหลายตัวแทน โดยเพิ่มการตรวจสอบและทวนสอบก่อนการเรียกใช้งานแอปพริเคชันแบบอัตโนมัติ ผ่านความน่าเชื่อถือของใบรับรอง หรือ Certificate Trust Lists (CTLs) และใช้กรณีทดสอบสำหรับการยืนยันความน่าเชื่อถือของระบบตัวอย่างผ่านการใช้งานธุรกรรมการเงิน อีกทั้งยังเสนอว่าควรจะต่อยอดโดยการเสริมความทนทาน (Robustness) ของระบบหลายโดเมน และความน่าเชื่อถือ (Reliability) ที่จำเป็นต้องได้รับการทดสอบภายใต้ลูกค้าจำนวนมาก โดยจะสามารถจัดการความซับซ้อนของโดเมน PKI ที่แตกต่างกันของแอปพลิเคชันต่าง ๆ และสถาปัตยกรรมอื่น ๆ ได้มากขึ้น และควรมีการประเมินและทวนสอบเพิ่มเติมในส่วนประสิทธิภาพการทำงานและการใช้ทรัพยากรของระบบ

2.2.3. Cardea: Dynamic access control in distributed systems [8]

งานวิจัยนี้ได้มีการนำเสนอระบบกำหนดการเข้าถึงแบบกระจาย (Distributed Authorization System) โดยใช้ชื่อโครงการนี้ว่า Cardea ซึ่งเป็นสวนหนึ่งของ NASA Information Power Grid ที่ซึ่งได้มีการนำ SAML และ XACML มาใช้ในการควบคุมการเข้าถึงการทำงานของระบบแบบกระจายเพื่อต้องการสร้างความปลอดภัยและตรวจสอบการทำงานของสิ่งแวดล้อมแบบกระจายสำหรับการเข้า

ใช้งานทรัพยากรข้ามโดเมนอย่างยืดหยุ่น Cardea ถูกพัฒนาด้วยภาษา Java และส่วนประกอบของระบบที่ประกอบด้วย SAML Policy Decision Point, Attribute Authorities, Policy Enforcement Points, Information Service/LDAP, XACML context handler, XACML Policy Administration Points และ XACML Policy Decision Point ดังแสดงในรูปที่ 4

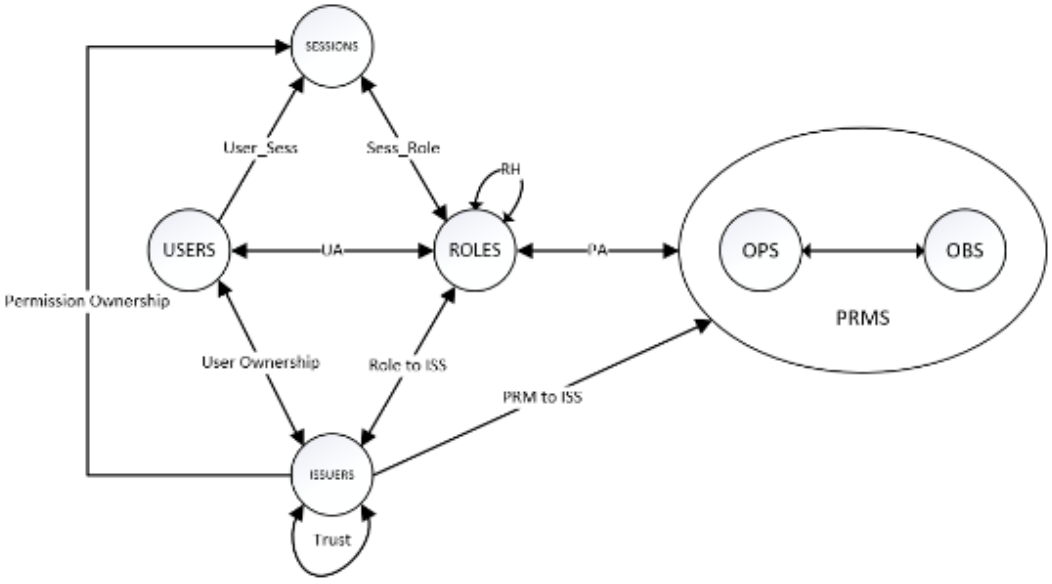


รูปที่ 4 โครงสร้างการทำงานของ Cardea [8]

บทที่ 3
แนวคิดและวิธีการวิจัย

3.1 การออกแบบโมเดลสำหรับตัวจัดการความมั่นคงไมโครเซอร์วิสอ้างอิง RBAC

งานวิจัยนี้ได้พัฒนาตัวจัดการความมั่นคง (Security Manager) บนสถาปัตยกรรมไมโครเซอร์วิส โดยออกแบบการควบคุมการเข้าถึงบนฐานบทบาทที่อ้างอิงโมเดลของ RBAC ตามมาตรฐาน (ANSI, INCITS) ซึ่งเป็นตัวกำหนดคุณสมบัติสำหรับจัดการเซสชันการตรวจสอบสิทธิ์ซึ่งจะเป็นตัวออกสิทธิ์ (Issuer) แก่ผู้ใช้บนฐานบทบาทของตน ดังนั้น ตัวจัดการความมั่นคงจึงถูกนำมาใช้เพื่อการจัดการข้อมูลเอกลักษณ์ของผู้ใช้งาน โดยตัวออกสิทธิ์จะดำเนินการควบคุมระบบการอนุญาตบนหลายบริการ ดังรูปที่ 5 ความสัมพันธ์ที่เพิ่มเข้ามาใน RBAC คือ Issuer Trust ($IT \subseteq I \times I, \forall i_n$) ผู้ที่ใช้บริการเหล่านี้จะสร้างอินเทอร์เฟซส่วนตัว เพื่อดึงข้อมูลส่วนบุคคลและการดำเนินการของผู้ใช้งานแต่ละรายจะถูกแยกออกจากกันอย่างชัดเจน นอกจากนี้ ความสัมพันธ์ IT ยังเป็นแบบ Role Hierarchy (RH) ซึ่งแสดงถึงห่วงโซ่แห่งความไว้วางใจของแต่ละบริการที่สัมพันธ์กับผู้ใช้นั้น และการอนุญาต (PRMS) จาก IT เพื่อให้สามารถเข้าถึงข้อมูลผ่านอินเทอร์เฟซการอนุญาต ประกอบด้วย 3 ส่วน: 1) สิทธิ์การดำเนินการ (Rights) 2) ตัวดำเนินงาน (Operation) และ 3) ข้อมูลที่ได้รับสิทธิ์ (Grant Objects) การให้สิทธิ์ผู้ใช้หนึ่งรายจะขึ้นอยู่กับตัวออกสิทธิ์เพียงรายเดียว และความเป็นเจ้าของแต่ละรายจะเกี่ยวข้องกับตัวออกสิทธิ์เพียงรายเดียวเท่านั้น จากนั้นจะดำเนินงานสร้างเซสชันให้ผู้ได้รับสิทธิ์ ซึ่งกิจกรรมต่างๆของการเข้าถึงของผู้ใช้จะถูกจัดเก็บ และจำกัดโดยการเป็นเจ้าของสิทธิ์



รูปที่ 5 โมเดล RBAC สำหรับตัวจัดการความมั่นคงไมโครเซอร์วิส

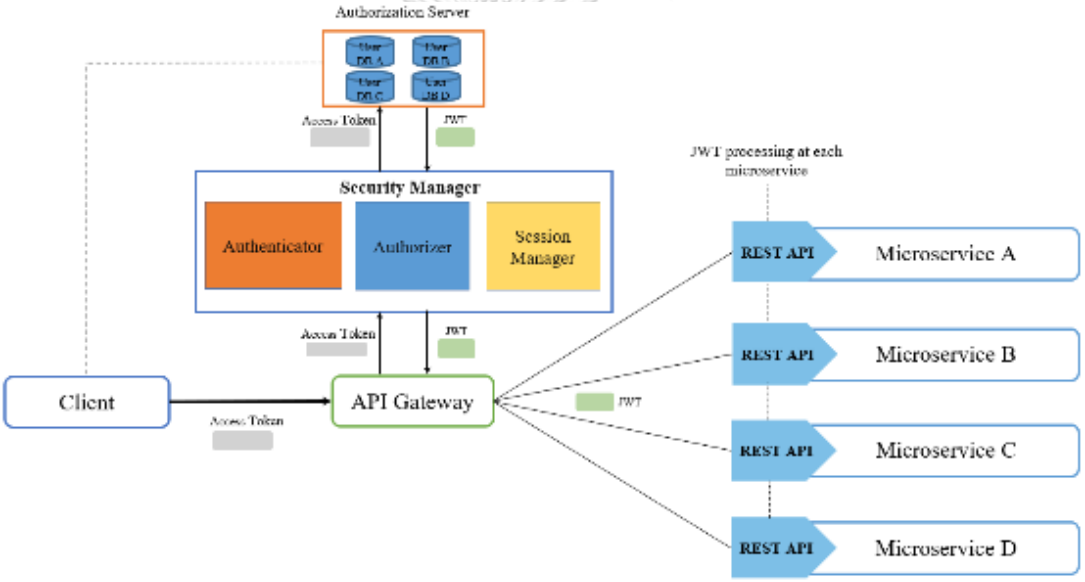
รายละเอียดความสัมพันธ์ของบริบทต่าง ๆ สามารถอธิบายด้วยเซตทางคณิตศาสตร์ที่ถูกนิยาม เช่น

- *Role to ISS* $\subseteq$ *ROLES x ISSUERS* คือ ความสัมพันธ์ของบทบาทและตัวออกสิทธิ์ เป็นความสัมพันธ์แบบกลุ่มต่อกลุ่ม (many-to-many relationships)
- *ISSUERES_roles* : *ISSUERES* $\longrightarrow$ *ROLES* คือ ความสัมพันธ์ของตัวออกสิทธิ์เป็นไปยังบทบาท
- *ISSUERES_roles* (*iss*) = *{r* $\in$ *ROLES|(r, iss)* $\in$ *Role to ISS }* คือ ความสัมพันธ์ของตัวออกสิทธิ์เป็นไปยังบทบาทซึ่งจะเกิดขึ้นแบบหนึ่งต่อหนึ่งเมื่อบทบาท r ที่ผูกกับตัวออกสิทธิ์ iss นั้น
- *Permission Ownership : SESSIONS* $\longrightarrow$ $2^{Permission\ Ownership}$ คือ ความสัมพันธ์ของเจ้าของสิทธิ์ที่ได้รับอนุญาต ซึ่งจะเกิดขึ้นตามจำนวนเซสชันที่เป็นคู่ของตัวออกสิทธิ์และเซสชันนั้น
- *PRMS* = $2^{(OPS \times OBJ)}$ คือ จำนวนของการอนุญาต (Permission) ตามจำนวนของคู่ความสัมพันธ์ระหว่างเซตของตัวดำเนินการ (Operations) และ ข้อมูลที่ถูกเข้าถึง (Objects)
- *PRM to ISS* $\subseteq$ *PRMS x ISSUERES*, คือ ความสัมพันธ์ของการอนุญาตและตัวออกสิทธิ์เป็นความสัมพันธ์แบบกลุ่มต่อกลุ่ม
- *ISSUERES_perms* : *ISSUERES* $\longrightarrow$ 2^{PRMS} คือ ความสัมพันธ์ของการอนุญาตที่ได้รับจากตัวออกสิทธิ์ ซึ่งจะเกิดขึ้นตามจำนวนการอนุญาต ที่เป็นคู่ของการอนุญาตและตัวออกสิทธิ์นั้น
- *ISSUERES_perms* (*iss*) = *{p* $\in$ *PRMS|* (*p, iss*) $\in$ *PRM to ISS}* คือ ความสัมพันธ์ของตัวออกสิทธิ์ไปยังการอนุญาตซึ่งจะเกิดขึ้นแบบหนึ่งต่อหนึ่งเมื่อการอนุญาต p ที่ผูกกับตัวออกสิทธิ์ iss นั้น

3.2 การพัฒนาระบบตัวจัดการความมั่นคงไมโครเซอร์วิสด้วยการควบคุมการเข้าถึงบนฐานบทบาทจากโมเดลที่ได้กำหนดขึ้น

การพัฒนาระบบตัวจัดการความมั่นคงไมโครเซอร์วิสด้วยการจัดการเอกลักษณ์จากโมเดล RBAC ที่ได้กำหนดขึ้น ในการพัฒนาระบบนี้ให้เป็นสถาปัตยกรรมไมโครเซอร์วิสจะถูกทำบนแอปพลิเคชันคอนเทนเนอร์ที่เรียกว่า Docker ซึ่งจะใช้งาน แอปพริเคชันย่อยตามเซอร์วิสนั้นโดยไม่ขึ้นกับระบบปฏิบัติการ ซึ่งตัวคอนเทนเนอร์จะจัดการเตรียมทรัพยากรที่จำเป็นต่อระบบโดยไม่จำเป็นต้องลงระบบปฏิบัติการเสมือน (Virtual Machine) ทำให้ลดขนาดข้อมูลได้เป็นจำนวนมาก อีกทั้งยังสามารถขยายขนาดและเพิ่มเติม อินสแตนซ์ (Instance) หรือระบบย่อยของแต่ละบริการได้เลยโดยไม่ขึ้นกับ

Instance อื่น ทั้งนี้ในโมเดลที่นำเสนอนั้นสามารถแบ่งการทำงานออกเป็น 2 ใหญ่ คือ ส่วนของ Services และ ส่วนของ Security Manager ซึ่งจะเชื่อมต่อกันด้วย API Gateway ที่ทำหน้าที่ติดต่อแต่ละเซอร์วิสผ่าน REST / HTTP โดยจะส่งข้อมูลแบบ JWT (JSON Web Token) [9] ที่มีการเข้ารหัสไว้ ซึ่งในขั้นตอนสำคัญที่เกี่ยวข้องในการใช้การรักษาความมั่นคงของไมโครเซอร์วิสนั้น จะทำการยืนยันตัวตนด้วย Access token ของผู้ใช้งานไปที่ Authenticator และ Authorizer จะส่งสิทธิ์การเข้าถึงข้อมูลของผู้ใช้งานนั้น โดยสร้างเซสชันผ่าน Session Manager ที่ควบคุมการเชื่อมต่อของผู้ใช้งานในระบบที่ร้องขอ ซึ่งแสดงได้ตามรูปที่ 5 โดยจะติดตั้งด้วย Node.js ที่เป็นการประมวลผล (runtime) ของภาษา JavaScript โดยจะเชื่อมโยงนโนบายของกฎการเข้างานตามบทบาทด้วย XACML 3.0 เพื่อแสดงการทำงานของระบบทั้งหมด



รูปที่ 6 โครงสร้างระบบที่ใช้ในการทดสอบ

3.3 การทดสอบความถูกต้องของโมเดล

การตรวจสอบความถูกต้องของโมเดลที่พัฒนา จะทำโดยการตั้งกรณีทดสอบที่ครอบคลุมการใช้งานทั้งหมดและแสดงการใช้งานตัวอย่าง 2 กรณี คือ กรณีปกติในการตรวจสอบสิทธิ และกรณีของการบริการแบบประสานงานของการบริการแบบประสานงานของแต่ละโมโครเซอร์วิส จากนั้นทำการวิเคราะห์ในเชิงการเปรียบเทียบ เรื่องประสิทธิภาพการทำงานที่มากขึ้นด้วยหรือไม่ เนื่องจากตัวโมเดลจะต้องเป็นตัวกลางที่รองรับการร้องขอของผู้ตรวจสอบสิทธิ์เป็นจำนวนมาก เมื่อถูกใช้งานบนคลาวน์ โดยใช้การวัดในเรื่องของเวลาการตอบสนอง ค่าปริมาณงานในการรองรับการโหลด และแผนภูมิของระยะเวลาการตอบสนองทั้งหมด ทั้งนี้จะอ้างอิงจากการทำงานจริงโดยใช้เหตุการณ์การโจมตีสมมติเพื่อดูกรณีทดสอบว่าเป็นไปตามที่ได้ออกแบบไว้ และมีความถูกต้องสมบูรณ์ดี

บทที่ 4
การออกแบบและพัฒนาระบบ

4.1 สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา

4.1.1. สภาพแวดล้อม

1. ระบบปฏิบัติการ Microsoft Windows 10 64-bit
2. หน่วยประมวลผลอินเทล คอร์ ไอ 7 2.90 กิกะเฮิร์ต (CPU Intel Core i7 2.90Ghz)
3. หน่วยความจำ 16 กิกะไบต์ (RAM 16 GB)

4.1.2. เครื่องมือที่ใช้ในการพัฒนา

1. Microsoft Visual Studio Code
2. MongoDB Atlas Database
3. Docker Desktop

4.2 การออกแบบสถาปัตยกรรม

งานวิจัยนี้มุ่งเน้นการนำเสนอระบบการเข้าใช้งานผ่านตัวจัดการความมั่นคงของระบบไมโครเซอร์วิส โดยจะนำการรวม Local Policy จากการเข้าใช้งานผ่านระบบโมโนลิทิกแบบเดิม นำมารวมกันเป็น Group Policy เพื่อให้เหมาะกับการเข้าใช้งานแบบไมโครเซอร์วิสที่มี API Gateway เป็นตัวกลางในการจัดการเอกลักษณ์ การระบุตัวตน การพิสูจน์ตัวตน และการควบคุมสิทธิการใช้งานตามลำดับ ประกอบด้วย 3 ขั้นตอนหลัก คือ 1) การสร้างระบบสำหรับทดสอบ 2) การจับคู่และการรวมนโยบาย และ 3) การระบุตัวตน การพิสูจน์ตัวตน และการควบคุมสิทธิ์การใช้งานในระบบไมโครเซอร์วิส

4.2.1. ส่วนประกอบในการทำงานของระบบทั้งหมด

การพัฒนาระบบที่ใช้ตัวจัดการความมั่นคงบนสถาปัตยกรรมไมโครเซอร์วิส จะมีโครงสร้างการทำงานของระบบที่ประกอบไปด้วยการจำลองระบบไมโครเซอร์วิสสมมติที่มีจากใช้งาน REST API ในการติดต่อกับผู้ใช้ โดยตัวจัดการความมั่นคงจะเป็นผู้ตรวจสอบ ยืนยัน และใช้สิทธิผู้ใช้ของระบบย่อยทั้งหมด

4.2.2. การจับคู่และการรวมนโยบาย (Policy Mapping and Integration)

ในส่วนของการทำ Policy Mapping and Integration นั้นจะเป็นการช่วยให้แต่ละระบบที่มีการกำหนดสิทธิที่แตกต่างกันให้สามารถทำงานร่วมกันได้ ด้วยการนำเสนอวิธีการ Mapping การ

กำหนดสิทธิรูปแบบต่างๆ โดยจะแปลงนโยบายเดิม (Local Policy) จากระบบแบบโมโนลิทิกเดิมที่มีอยู่ในรูปแบบ Ruled-Based และ XML-Based ของแต่ละ Local Policy ซึ่งตัวอย่างการกำหนดความสัมพันธ์ระหว่างชื่อผู้ใช้และบทบาทแบบ Rule-Based แสดงในภาคผนวก ก และ ตัวอย่างการกำหนดสิทธิเข้าใช้งานแบบ Rule-Based แสดงในภาคผนวก ข ในส่วนของนโยบายแบบ XML-Based ตัวอย่างตัวอย่างการกำหนดความสัมพันธ์ระหว่างชื่อผู้ใช้และบทบาทแสดงในภาคผนวก ค และ ตัวอย่างการกำหนดสิทธิเข้าใช้งานแบบ XML-Based แสดงอยู่ในภาคผนวก ง ตามลำดับซึ่งจะถูกแปลงให้มาอยู่ในรูปแบบของ XACML โดย XACML ถือได้ว่าเป็นภาษากลางที่เป็นมาตรฐานในการกำหนดสิทธิได้ หลังจากทำการ Mapping แล้วจะมีการเก็บ Policy ไว้ใช้งานในครั้งต่อไป ซึ่งเป็นข้อดีในการทำ Policy Administration and Coordination และการกำหนด Authorization Model ในรูปแบบ RBAC ที่มีการ Integrate Policy เข้าด้วยกัน เป็น Group Policy ของแต่ละเซอร์วิส ซึ่งสามารถสนับสนุนการทำงานในกรณีที่มีจำนวน User มากๆ ในแต่ละ Domain ได้ดีกว่าแต่ละอันซึ่งเป็น RBAC อยู่แล้ว นอกจากนี้ อัลกอลิทึมที่นำเสนอจะครอบคลุมถึงการ Mapping ที่สอดคล้องกับ XACML Syntax อย่างครบถ้วน

ในการ Mapping XML Policy นั้น ลักษณะของ XML Policy จะเป็นเซตของกฎต่างๆ ที่ใช้ในการกำหนดสิทธิการเข้าใช้งานให้แต่ละบทบาท โดยจะมีส่วนประกอบ 2 ส่วนคือ Condition และ Element Privilege ซึ่งจะมี Action ตามสิทธิของแต่ละบทบาท โดยส่วนของ Condition จะมีอีก 3 Elements คือ Role, ConstraintType และ ConstrainiValue ซึ่ง ConstraintType ประกอบด้วย 3 ประเภทด้วยกัน ได้แก่ Time Constraint, Location Constraint และ Event Constraint มีไว้สำหรับกำหนดเงื่อนไขในการเข้าใช้งานระบบ นอกจากนี้ จะกำหนดให้แต่ละกฎในนโยบายแบบ XML จะต้องไม่มีการขัดแย้งกัน และมีการจัดเก็บนโยบายแบบ XML ไว้เพียงที่เดียวกันทั้งหมด โดยวิธีในการ Mapping จาก XML Policy ไปเป็น XACML Policy นั้น จะมี 2 ขั้นตอนหลักด้วยกัน ได้แก่

1. การแยกส่วนประกอบของ XML Policy (Decompose XML Policy Element) คือ ทำการแยกส่วนประกอบย่อยของ XML Policy ประกอบด้วย Role, Privilege, ConstraintType และ ConstraintValue
2. การจับคู่ XML Element กับ XACML Tag (Map XML elements into XACML Tag) โดยสิ่งที่ต้องการจากขั้นตอนนี้คือ XACML Policy ซึ่งประกอบด้วยส่วนของ XACML Rule List โดยมีวิธีการคือ นำส่วนของ Role มาจับคู่กับแท็ก Subject ของ XACML และนำส่วนของ Privilege มาจับคู่กับแท็ก Action ของ XACML ส่วน Constraint จะนำ ConstraintType มาแยกประเภทก่อนแล้วจึงนำค่า ConstraintValue มาจับคู่กับแท็กที่จะกำหนด Constraint ประเภทต่างๆ ประกอบด้วย Time Constraint, Location

Constraint หรือ Event Constraint เพื่อให้ได้ XACML Rule List ที่จะนำมารวมกันเป็น XACML Policy

ตัวอย่างเงื่อนไขเกี่ยวกับการกำหนดสิทธิ < Action >

สมมติให้ a เป็น Privilege ของ Group Policy ซึ่งมีค่าเท่ากับ read เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังรูปที่ 7

```
<Actions>
    <Action>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml: 1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                read
            </AttributeValue>
            <ActionAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml: 1.0:action:action-id"
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
        </Action>
    </Actions>
```

รูปที่ 7 ตัวอย่างเงื่อนไขเกี่ยวกับการกำหนดสิทธิ < Action >

ตัวอย่างเงื่อนไขเกี่ยวกับการกำหนดบทบาท < Subject >

สมมติให้ role เป็น Role ของ Group Policy ซึ่งมีค่าเท่ากับ admin เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังรูปที่ 8


```
<Subjects>
    <Subject>
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
                admin
            </AttributeValue>
            <SubjectAttributeDesignator AttributeId="role"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
    </Subject>
</Subjects>
```

รูปที่ 8 ตัวอย่างเงื่อนไขเกี่ยวกับกากำหนดบทบาท < Subject >

ตัวอย่างเงื่อนไขเกี่ยวกับเวลา < TimeConstraint >

สมมติให้ Group Policy มีคำของ ConstraintType เป็น time และค่าของ ConstrainiValue เป็น 0900 - 1700 เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังรูปที่ 9

```
<Condition FunctionId="um:oasis:names:tc:xacml:1.0:function:and">
    <Apply FunctionId="urn:oasis:names:tc:xacml: 1.0:function:time-greater-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
            <EnvironmentAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
                DataType="http://www.w3.org/2001/XMLSchema#time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">
            09:00:00
        </AttributeValue>
    </Apply>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
            <EnvironmentAttributeDesignator
                AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
                DataType="http://www.w3.org/2001/XMLSchema#time">
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">
            17:00:00
        </AttributeValue>
    </Apply>
</Condition>
```

รูปที่ 9 ตัวอย่างเงื่อนไขเกี่ยวกับเวลา < TimeConstraint >

ตัวอย่างเงื่อนไขเกี่ยวกับสถานที่ < LocationConstraint >

สมมติให้ Group Policy มีค่าของ ConstraintType เป็น location และค่าของ ConstrainiValue เป็น 192. 168.1.1 เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังรูปที่ 10


```
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
        <SubjectAttributeDesignator AttributeId="location"
            DataType="http://www.w3.org/2001/xMLSchema#string"/>
    </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            192.168.1.1
        </AttributeValue>
</Condition>
```

รูปที่ 10 ตัวอย่างเงื่อนไขเกี่ยวกับสถานที่ < LocationConstraint >

ตัวอย่างเงื่อนไขเกี่ยวกับเหตุการณ์ < EventConstraint >

สมมติให้ Group Policy มีค่าของ ConstraintType เป็น Event และค่าของ ConstrainiValue, เป็น Quantity < 100 เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าแป็นดังรูปที่ 11

```
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
        <SubjectAttributeDesignator AttributeId="Quantity"
            DataType="http://www.w3.org/2001/XMLSchema#integer"/>
    </Apply>
        <AttributeValue DataType="http:/www.w3.org/2001/xMLSchema#integer">
            100
        </AttributeValue>
</Condition>
```

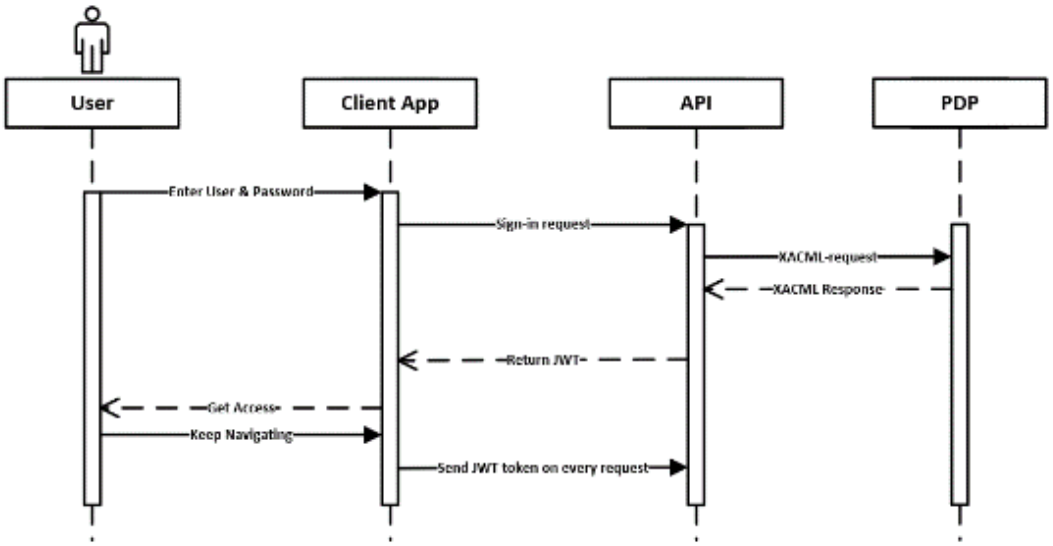
รูปที่ 11 ตัวอย่างเงื่อนไขเกี่ยวกับเหตุการณ์ < EventConstraint >

4.2.3. การระบุตัวตน การพิสูจน์ตัวตน และการควบคุมสิทธิ์การใช้งานของตัวจัดการความมั่นคงไมโครเซอร์วิส

สำหรับไมโครเซอร์วิส การระบุตัวตน การพิสูจน์ตัวตน และการควบคุมสิทธิ์การใช้งาน มักนิยมใช้โทเคนการเข้าถึงที่ถูกเข้ารหัสแบบ Jason Web Token (JWT) ผ่านช่องทาง SSL/TLS ที่ส่งผ่านเพื่อตรวจสอบความถูกต้องของข้อมูลประจำตัวผู้ใช้ และให้สิทธิ์เซสชันผู้ใช้ตามลำดับ JWT [9] เป็นมาตรฐานแบบเปิด (RFC 7519) ซึ่งจะใช้งาน Application Programming Interface (API) ในการติดต่อสื่อสารกันระหว่างฟังก์ชันของเซอร์วิส โดยผู้ที่เข้ามาเชื่อมต่อ ไม่จำเป็นจะต้องเข้าใจถึงกลไกการทำงานและความซับซ้อนที่ซ่อนอยู่ข้างใน โดยมีการรวมศูนย์การจัดการ API ทั้งหมดเข้าสู่ API Gateway ที่ทำหน้าที่จัดการความมั่นคงปลอดภัยและการเข้าถึงเซอร์วิสแต่ละตัว ซึ่งใช้การส่งข้อมูลระหว่างบริการอย่างปลอดภัย โครงสร้างของตัวโทเคน JWT ประกอบด้วย Header, Payload และ Signature อัลกอริทึมการเข้ารหัสที่ใช้ตามปกติคือ HMAC SHA256 และเข้ารหัสในรูปแบบ Base64Url ดังแสดงในรูปที่ 12

รูปที่ 12 การใช้งาน JWT ที่มีการเข้ารหัสและถอดรหัสสำหรับการพิสูจน์ตัวตน (jwt.io debugger).

สำหรับขั้นตอนในการเข้าใช้งานของผู้ใช้จะเริ่มจากการกรอกชื่อผู้ใช้และรหัสผ่านเพื่อเข้าใช้แอปพลิเคชันที่ต้องการ จากนั้น จึงส่งคำร้องขอเข้าใช้งานผ่านตัว API ในรูปแบบ JWT โดยผ่าน API Gateway ไปที่ตัวจัดการความมั่นคงที่เก็บนโยบาย XACML การเข้าถึงไว้ โดย PDP จะทำการตรวจสอบระบุตัวตนของผู้ใช้ และตรวจสอบสิทธิการเข้าถึงบนฐานบทบาท และส่งการตอบกลับแบบ XACML กลับไปผ่านช่องทางเดิม ในรูปแบบ JWT เช่นเดิม จนในที่สุด แอปพลิเคชันนั้นจะให้สิทธิตามที่ผู้ใช้งานร้องขอ และเมื่อมีการใช้งานต่อเนื่อง ก็จะส่งข้อมูลการใช้งานผ่านโทเคน JWT ต่อไป แสดงดังรูปที่ 13



รูปที่ 13 ขั้นตอนในการเข้าใช้งานระบบไมโครเซอร์วิส

เมื่อมีการส่งคำร้องจาก API Gateway มาที่ตัวจัดการความมั่นคงซึ่งจะทำการยืนยันข้อมูลกับฐานข้อมูลผู้ใช้เพื่อตรวจสอบว่าผู้ใช้นั้นมีบทบาทใดและมีสิทธิเข้าใช้งานระบบใดบ้าง และในระบบนั้นมีสิทธิทำอะไรได้บ้าง โดยจะทำการตรวจสอบผ่านฟังก์ชัน Authorization.js แสดงดังตัวอย่างโปรแกรมตามรูปที่ 14 (https://github.com/CodingGangsta/express-rbac.git)

ทั้งนี้ เมื่อมีการออกจากระบบ (logout) ตัวจัดการความมั่นคงจะส่งคำสั่งเพื่อนำเซสชันที่ไม่มีการใช้งานออกจากระบบข้อมูลของผู้เข้าใช้งาน ณ ขณะนั้น เพื่อไม่ให้มีการขโมยช่องทางการใช้งานของผู้ไม่หวังดี ตัวอย่างโปรแกรมดังกล่าวแสดงดังรูปที่ 15 ซึ่งจะสามารถช่วยตรวจสอบการเข้าใช้งานที่ผิดบทบาท หรือ ไม่มีข้อมูล object ที่ระบุได้ และยังช่วยเรื่องการเข้าถึงข้อมูลที่ไม่ตรงตามสิทธิ์ได้อีกทางหนึ่ง และเมื่อมีการเรียกใช้ลำดับชั้นของบทบาทตามที่กำหนดไว้ จะมีการเรียกแพคเกจ accesscontrol (https://www.npmjs.com/package/accesscontrol) ซึ่งจะเป็นตัวช่วยจัดการกำหนดสิทธิ์ย่อยระหว่างชั้นแต่ละชั้นว่าต่างกันอย่างไร เช่น ผู้จัดการของผู้ดูแลระบบ จะมีสิทธิ์การเข้าถึงตามบทบาทผู้ดูแลระบบในทุกระบบ เป็นต้น

Git > rbac > JS authorize.js > ...

```
1   module.exports = Authorization;
2
3   var Middlewares = require('./framework/middlewares');
4   var middlewares = new Middlewares();
5   var Functions = require('./framework/functions');
6   var functions = new Functions();
7
8   function Authorization() {
9       this._options = {};
10      this._loadCallback = null;
11      this._functions = null;
12      this._middlewares = null;
13  }
14
15  Authorization.prototype.authorize = function (options, loadCallback) {
16      if (!loadCallback) {
17          loadCallback = options;
18          options = {};
19      }
20      if (typeof loadCallback !== 'function') {
21          throw new Error("Authorization: Load callback should be a function");
22      }
23      this._options = options;
24      this._loadCallback = loadCallback;
25      this._functions = functions.register();
26      this._middlewares = middlewares.register(this);
27      return this._middlewares.initialize();
28  };
29
30  Authorization.prototype.isInRole = function (role) {
31      return this._middlewares.isInRole(role);
32  }
33
34  Authorization.prototype.isInAnyRole = function (role) {
35      return this._middlewares.isInAnyRole(role);
36  }
37
38  Authorization.prototype.hasPermission = function (permission) {
39      return this._middlewares.hasPermission(permission);
40  }
41
42  Authorization.prototype.hasAnyPermission = function (permission) {
43      return this._middlewares.hasAnyPermission(permission);
44  }
```

รูปที่ 14 ตัวอย่างโปรแกรม Authorization.js ที่ทำการตรวจสอบสิทธิผู้ใช้

removePermission.js

```
RBAC.prototype._removePermission = function (objects, roles, actionOwnership) {
    var _this = this;
    objects = utils_1.utils.toStringArray(objects);
    if (objects.length === 0 || !utils_1.utils.isFilledStringArray(objects)) {
        throw new core_1.AccessControlError(
            "Invalid object(s): " + JSON.stringify(objects));
    }
    if (roles !== undefined) {
        roles = utils_1.utils.toStringArray(roles);
        if (roles.length === 0 || !utils_1.utils.isFilledStringArray(roles)) {
            throw new core_1.AccessControlError
                ("Invalid role(s): " + JSON.stringify
                    (roles));
        }
    }
    utils_1.utils.eachRoleObject(this._grants, function (role, object, permissions) {
        if (objects.indexOf(object) >= 0
            && (!roles || roles.indexOf(role) >= 0)) {
            if (actionOwnership) {
                var ao = utils_1.utils.normalizeActionOwnership({ action: actionOwnership }, true);
                delete _this._grants[role][object][ao];
            }
            else {
                delete _this._grants[role][object];
            }
        }
    });
};
```

รูปที่ 15 ตัวอย่างโปรแกรมที่ใช้สำหรับการนำเซสชันที่ไม่มีการใช้งานออกจากระบบ

บทที่ 5
การทดสอบและวิเคราะห์ผล

5.1 วัตถุประสงค์ของการทดสอบ

เพื่อตรวจสอบความถูกต้องของระบบที่พัฒนา โดยตั้งกรณีทดสอบที่ครอบคลุมการใช้งานเมื่อมีการรวมนโยบายการเข้าใช้งานของแต่ละระบบย่อยมาเป็นการเข้าใช้งานที่ใช้นโยบายแบบกลุ่มและแสดงการใช้งานตัวอย่าง 2 กรณี กล่าวคือ กรณีปกติในการตรวจสอบสิทธิตามนโยบายเดี่ยว และกรณีการตรวจสอบสิทธิตามนโยบายแบบกลุ่มซึ่งสามารถจัดการเอกลักษณ์ผู้ใช้งานของแต่ละโมโครเซอร์วิสได้ โดยจะทำการทดสอบประสิทธิภาพการทำงานของระบบ และทดสอบความปลอดภัยของการเข้าใช้ผ่านตัวอย่างการการโจมตีทางไซเบอร์

5.2 การทดสอบระบบ

ในส่วนนี้แสดงถึงผลการตรวจสอบความถูกต้องในการทำงานของระบบ โดยแบ่งผลการตรวจสอบความถูกต้องออกเป็น 2 ส่วน ดังต่อไปนี้

5.2.1. ผลการตรวจสอบความถูกต้องในส่วนของการยืนยันตัวบุคคล

ในส่วนของการยืนยันตัวบุคคลในงานวิจัยนี้เป็นการยืนยันตัวบุคคลทั้งแบบนโยบายเดี่ยวและนโยบายกลุ่ม ซึ่งในการยืนยันตัวบุคคล 2 แบบนี้จะต้องถูกต้องทั้งหมด ตารางที่ 2 สรุปผลลัพธ์ตัวอย่างในการตรวจสอบการยืนยันตัวบุคคลดังนี้

ตารางที่ 2 ผลการตรวจสอบความถูกต้องในส่วนของการยืนยันตัวบุคคล

ชนิดของเซอร์วิส	การตรวจสอบสิทธิตามนโยบายเดี่ยว	กรณีการตรวจสอบสิทธิตามนโยบายกลุ่ม	ผลที่คาดหวัง
Service A	ผ่าน	ผ่าน	ถูกต้องเหมือนกัน
Service B	ผ่าน	ผ่าน	ถูกต้องเหมือนกัน
Service C	ผ่าน	ผ่าน	ถูกต้องเหมือนกัน
Service D	ผ่าน	ผ่าน	ถูกต้องเหมือนกัน

5.2.2. ผลการตรวจสอบความถูกต้องในส่วนของการกำหนดสิทธิเข้าใช้งาน

ในแต่ละบทบาทของผู้ใช้จะมีการกำหนดเงื่อนไขในการเข้าใช้งานไว้ โดยสิทธิตามบทบาททั้งหมด แสดงตามตารางที่ 3 และเมื่อได้ทำการรวมเงื่อนไขทั้งหมดตามค่านโยบายแบบเดี่ยวที่มี เป็น

นโยบายกลุ่มแล้ว ดังแสดงตามตารางที่ 4 ซึ่งผลลัพธ์ของตัวอย่างในการตรวจสอบการกำหนดสิทธิ์เข้าใช้งานจะถูกทดสอบตามสิทธิพื้นฐาน 9 สิทธิ์ ประกอบด้วย

1. canAddUser
2. canUpdateUser
3. canDeleteUser
4. canEditContent
5. canRemoveContent
6. canAuditPeople
7. canViewContent
8. canAddContent
9. canViewReports

โดยแต่ละบทบาทจะถูกกำหนดด้วยเงื่อนไขบังคับที่ต่างกัน ทั้งนี้ ได้ทำการทดสอบ 36 กรณีทดสอบ (4 บทบาท x 9 สิทธิ์) ตัวอย่างดังแสดงในตารางที่ 4

ตารางที่ 3 สิทธิตามบทบาททั้งหมดของผู้ใช้งานในระบบ

บทบาท	**สิทธิ**
Admin	canAddUser canUpdateUser canDeleteUser canViewReports
Developer	canEditContent canRemoveContent canViewReports
Auditor	canAuditPeople canUpdateUser canRemoveContent canViewReports
User	canViewContent canAddContent

ตารางที่ 4 ตัวอย่างผลการตรวจสอบความถูกต้องในส่วนของการกำหนดสิทธิ์เข้าใช้งาน

กรณีทดสอบ	บทบาท	สิทธิ	รูปแบบเงื่อนไข	ค่าของเงื่อนไข	ค่าความจริงของเงื่อนไข	ผลลัพธ์จากนโยบายเดี่ยว	ผลลัพธ์จากนโยบายกลุ่ม	ผลการเปรียบเทียบผลลัพธ์
TC01	Admin	canAddUser	Time	0800-1900	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.1	จริง			
TC02	Admin	canUpdateUser	Time	0800-1900	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.1	จริง			
TC03	Admin	canDeleteUser	Time	0800-1900	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.1	จริง			
TC04	Admin	canEditContent	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.1	เท็จ			
TC05	Admin	canRemoveContent	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.1	เท็จ			
TC06	Admin	canAuditPeople	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.1	เท็จ			
TC07	Admin	canViewContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC08	Admin	canAddContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC09	Admin	canViewReports	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.1	จริง			
TC10	Developer	canAddUser	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.2	เท็จ			
TC11	Developer	canUpdateUser	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.2	เท็จ			
TC12	Developer	canDeleteUser	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.2	เท็จ			
TC13	Developer	canEditContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC14	Developer	canRemoveContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			

กรณีทดสอบ	บทบาท	สิทธิ	รูปแบบเงื่อนไข	ค่าของเงื่อนไข	ค่าความจริงของเงื่อนไข	ผลลัพธ์จากนโยบายเดี่ยว	ผลลัพธ์จากนโยบายกลุ่ม	ผลการเปรียบเทียบผลลัพธ์
TC15	Developer	canAuditPeople	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.2	เท็จ			
TC16	Developer	canViewContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC17	Developer	canAddContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC18	Developer	canViewReports	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.2	จริง			
TC19	Auditor	canAddUser	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.3	เท็จ			
TC20	Auditor	canUpdateUser	Time	0800-1900	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.3	จริง			
TC21	Auditor	canDeleteUser	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.3	เท็จ			
TC22	Auditor	canEditContent	Time	0800-1900	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	192.168.10.3	เท็จ			
TC23	Auditor	canRemoveContent	Time	0800-1900	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.3	จริง			
TC24	Auditor	canAuditPeople	Time	0800-1900	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.3	จริง			
TC25	Auditor	canViewContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC26	Auditor	canAddContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC27	Auditor	canViewReports	Time	0800-1900	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	192.168.10.3	จริง			
TC28	User	canAddUser	Time	-	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	-	เท็จ			
TC29	User	canUpdateUser	Time	-	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	-	เท็จ			

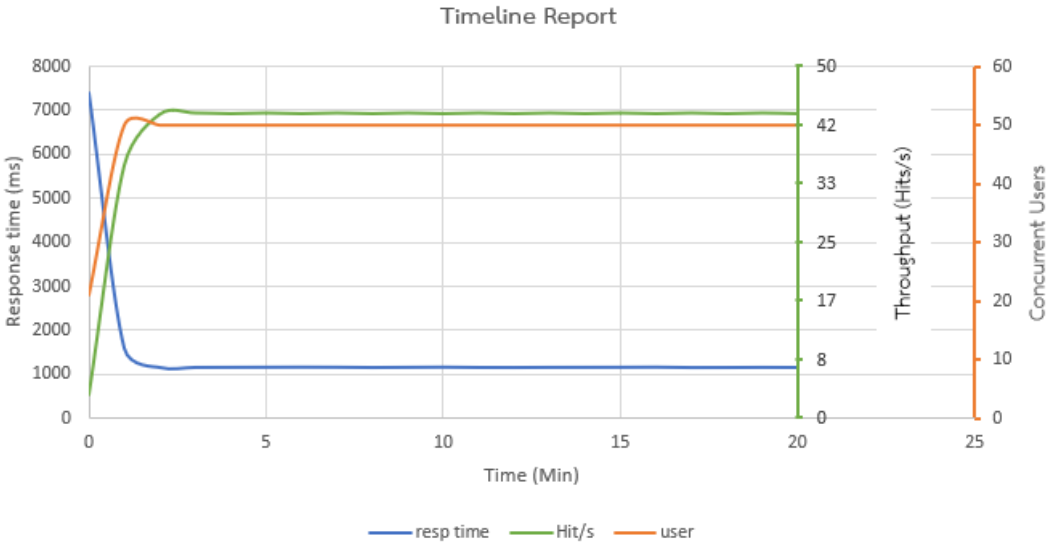
กรณีทดสอบ	บทบาท	สิทธิ	รูปแบบเงื่อนไข	ค่าของเงื่อนไข	ค่าความจริงของเงื่อนไข	ผลลัพธ์จากนโยบายเดี่ยว	ผลลัพธ์จากนโยบายกลุ่ม	ผลการเปรียบเทียบผลลัพธ์
TC30	User	canDeleteUser	Time	-	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	-	เท็จ			
TC31	User	canEditContent	Time	-	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	-	เท็จ			
TC32	User	canRemoveContent	Time	-	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	-	เท็จ			
TC33	User	canAuditPeople	Time	-	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	-	เท็จ			
TC34	User	canViewContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC35	User	canAddContent	Time	-	จริง	อนุญาต	อนุญาต	ตรงกัน
			Location	-	จริง			
TC36	User	canViewReports	Time	-	เท็จ	ไม่อนุญาต	ไม่อนุญาต	ตรงกัน
			Location	-	เท็จ			

5.2.3. ผลการทดสอบประสิทธิภาพของระบบ

จากการทดสอบประสิทธิภาพของระบบไมโครเซอร์วิสถูกทดสอบบนเว็บไซต์ blazemeter.com โดยใช้ค่าตัววัดประกอบด้วย เวลาการตอบสนอง (Response Time) ค่าปริมาณงานในการรองรับการโหลด (Throughput) และอัตราการส่งข้อมูล โดยสมมติให้มีผู้เข้าใช้งานทั้งหมด 50 คน ภายในเวลา 20 นาที ข้อมูลผลการทดสอบที่ได้แสดงดังตารางที่ 5 และกราฟการทดสอบประสิทธิภาพตามเวลาที่เกิดขึ้น แสดงตามรูปที่ 16 เห็นได้ว่า ระบบที่พัฒนามีค่าเวลาตอบสนองที่สูงในช่วงเวลาเริ่มแรกและตอบสนองไวขึ้นตามลำดับ โดยให้ค่าการรองรับโหลดได้ทั้งหมดตามข้อดีของสถาปัตยกรรมไมโครเซอร์วิส และมีอัตราการส่งข้อมูลที่ต่ำ เฉลี่ยที่เพียง 60.23 KBytes/s เท่านั้น นับว่ามีประสิทธิภาพเพียงพอการใช้งานจริงได้

ตารางที่ 5 ผลการทดสอบประสิทธิภาพของระบบไมโครเซอร์วิส

ค่าทดสอบ	ระบบไมโครเซอร์วิส
จำนวนการเรียกใช้งาน (samples)	48200
เวลาการตอบสนองเฉลี่ย avgResponseTime(ms)	1214.58
ค่าปริมาณงานเฉลี่ย avgThroughput (Hits/s)	40.23
เวลาการตอบสนองต่ำสุด minResponseTime (ms)	1189
เวลาการตอบสนองสูงสุด maxResponseTime (ms)	8407
อัตราการส่งข้อมูลเฉลี่ย avgBytes (KBytes/s)	60.23
จำนวนผู้ใช้งาน concurrency	50



รูปที่ 16 กราฟการทดสอบประสิทธิภาพของระบบ

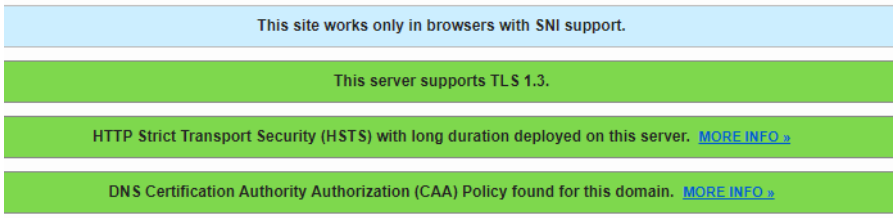
5.2.4. ผลการทดสอบความมั่นคงปลอดภัยของระบบ

สำหรับการทดสอบความมั่นคงปลอดภัยของระบบ ได้ทำการตรวจสอบ 2 แบบ:

1. การทดสอบ SSL Server test ซึ่งเป็นรายงานผลจาก Qualys SSL Labs ประกอบด้วยการทดสอบย่อย คือ การตรวจสอบใบรับรอง CA ความปลอดภัยในการแลกเปลี่ยนกุญแจระหว่างกัน อัลกอริทึมในการเข้ารหัส และ โปรโตคอลที่ปลอดภัยในการเรียกใช้งาน ตามตารางที่ 6 พบว่า ผลการทดสอบจากทั้ง 2 ระบบ ให้ค่าใกล้เคียงกันมาก แตกต่างกันที่ความหลากหลายของโปรโตคอลที่รองรับเท่านั้น โดยจากรูปที่ 17 เห็นได้ว่า ระบบไมโครเซอร์วิสที่พัฒนารองรับ TLS 1.3 จึงทำให้ได้รับการประเมินที่สูงกว่านั่นเอง

ตารางที่ 6 รายงานผลจาก Qualys SSL Labs

ชนิดระบบ	รายงานผลจาก Qualys SSL Labs
ระบบโมโนลิทิก	
ระบบไมโครเซอร์วิส	



รูปที่ 17 รายละเอียดของรายงานผลจาก Qualys SSL Labs บนระบบไมโครเซอร์วิส

2. การทดสอบการป้องกันการโจมตีทางไซเบอร์ด้วยโปรแกรม OWASP ZAP (https://www.zaproxy.org/) โดยทำการทดสอบการโจมตีชนิด Cross-Site Scripting (XSS) ซึ่งเป็นการฝังสคริปต์เพื่อโจมตีจากช่องโหว่ที่มีบนหน้าเว็บไซต์ และ การโจมตี SQL injection รวมถึงการโจมตีแบบ Buffer Overflow ผลการทดสอบที่ได้คือสามารถป้องกันการโจมตีได้ทั้งระบบโมโนลิทิกและไมโครเซอร์วิสทั้งคู่ ดังแสดงในตารางที่ 6 ซึ่งค่า Request ของแต่ละการโจมตีจะขึ้นกับรายละเอียดการแสดงผลของหน้า webpage นั้นว่ามี element ที่ตรงกับการแสกน ซึ่งจะถูกตรวจสอบตามที่โปรแกรมระบุไว้

ตารางที่ 7 ตัวอย่างผลการโจมตีทางไซเบอร์ด้วยโปรแกรม OWASP ZAP

Analyser	Monolithic Architecture			Microservice Architecture		
	Reqs	Alerts	Status	Reqs	Alerts	Status
Path Traversal	78	0	Completed	30	0	Completed
Remote File Inclusion	50	0	Completed	20	0	Completed
Source Code Disclosure - /WEB-INF folder	8	0	Completed	6	0	Completed
External Redirect	45	0	Completed	18	0	Completed
Server Side Include	20	0	Completed	8	0	Completed
Cross Site Scripting (Reflected)	20	0	Completed	8	0	Completed
Cross Site Scripting (Persistent) - Prime	5	0	Completed	2	0	Completed
Cross Site Scripting (Persistent) - Spider	31	0	Completed	12	0	Completed
SQL Injection	130	0	Completed	44	0	Completed
Server Side Code Injection	40	0	Completed	16	0	Completed
Remote OS Command Injection	160	0	Completed	64	0	Completed
Directory Browsing	31	0	Completed	12	0	Completed
Buffer Overflow	5	0	Completed	2	0	Completed
CRLF Injection	35	0	Completed	14	0	Completed
Parameter Tampering	35	0	Completed	14	0	Completed
ELMAH Information Leak	1	0	Completed	1	0	Completed
.htaccess Information Leak	8	0	Completed	3	0	Completed
Cross Site Scripting (DOM Based)	0	0	Completed	120	0	Completed

5.3 อภิปรายผลการวิจัย

จากการตรวจสอบความถูกต้องในส่วนของการยืนยันตัวบุคคลและการตรวจสอบความถูกต้องในส่วนของการกำหนดสิทธิเข้าใช้งาน นับได้ว่าการรวมนโยบายแบบเดี่ยวไปเป็นแบบกลุ่มมีความถูกต้องสมบูรณ์ ครบถ้วน โดยการทดสอบประสิทธิภาพของระบบแสดงในเห็นถึงความสะดวกรวดเร็วในการทำงานของสถาปัตยกรรมแบบไมโครเซอร์วิส ทั้งนี้หากพิจารณาเปรียบเทียบกับระบบตรวจสอบอัตลักษณ์เดิมบนไมโครเซอร์วิสที่มี เช่น Access Lists Control (ACL) ซึ่งจะถูกพบการใช้งานในองค์กรขนาดใหญ่ ที่จะอ้างอิงการเข้าใช้งานตามการตัดสินใจ (DAC) โดยจะมีข้อจำกัดในการดูแลรักษาและปรับปรุงข้อมูลสิทธิมากกว่า เมื่อเทียบกับระบบ RBAC ที่พัฒนา อีกทั้งระบบนี้ยังมีข้อดีที่สามารถลดปัญหาการทับซ้อนกันของสิทธิผู้ใช้งานในกรณีมีการเข้าใช้งานข้ามแอปพลิเคชันได้ และจากการทดสอบความมั่นคงปลอดภัยของระบบ นับว่าสามารถป้องกันการโจมตีทั่วไปได้ หากแต่มีข้อควรสังเกตคือ เมื่อเวลาผ่านไปอาจมี zero-day exploits เกิดขึ้นได้ ดังนั้นผู้ดูแลระบบต้องหมั่นอัปเดตเวอร์ชันของซอฟต์แวร์เพื่อลดความเสี่ยงจากช่องโหว่ประเภทนี้

บทที่ 6
สรุปผลการวิจัย

6.1 สรุปผลการวิจัย

จากการพัฒนากรอบงานการควบคุมการเข้าถึงบนฐานบทบาทสำหรับตัวจัดการความมั่นคงไมโครเซอร์วิส ที่นำข้อดีของ RBAC ซึ่งง่ายต่อการควบคุมผู้ใช้งาน และแบ่งแยกส่วนของผู้ใช้ บทบาท และสิทธิ์ออกมาอย่างชัดเจน ทำให้เป็นที่นิยมในการใช้งานในปัจจุบันเป็นอย่างสูง อีกทั้งการออกแบบโมเดลที่ดีทำให้ลดความทับซ้อน หรือผิดพลาดการการกำหนดสิทธิ์ที่ผิดพลาดได้ ซึ่งเมื่อนำโมเดลที่ออกแบบไว้มาพัฒนาระบบช่วยให้ผู้พัฒนาทราบถึงขอบเขตการทำงานอย่างชัดเจน นอกจากนี้ การใช้รูปแบบของนโยบาย eXtensible Access Control Markup Language (XACML) เพื่อทำการกำหนดสิทธิการเข้าใช้งานแบบกลุ่มช่วยการรวมกันของนโยบายเดี่ยวของแต่ละเซอร์วิสเข้ามาเป็นกลุ่มเดียวกันได้อย่างราบรื่น และยังมีความยืดหยุ่นในการใช้งานข้ามแพลตฟอร์มอีกด้วย สำหรับการประเมินผลด้านความถูกต้องของการทำงานของระบบและประสิทธิภาพในการทำงาน พบว่าระบบที่พัฒนาสามารถทำงานได้อย่างถูกต้องทุกขั้นตอน และมีข้อได้เปรียบในเรื่องประสิทธิภาพการทำงานที่รวดเร็วตามโครงสร้างสถาปัตยกรรมไมโครเซอร์วิส รวมทั้งแนวโน้มทางด้านการขยายตัว (Scalability) ของระบบที่ดีกว่าระบบเดิมอย่างเห็นได้ชัด

6.2 ข้อจำกัดในงานวิจัย

เนื่องจากระบบที่พัฒนาขึ้นนี้ เป็นเพียงระบบที่สร้างขึ้นเพื่อทดลองการทำงานเท่านั้น เพื่อตอบปัญหาการทำงานที่ซับซ้อนและไม่ครอบคลุมการใช้งานจริง ทำให้จำนวนแอปพลิเคชัน จำนวนผู้ใช้ และจำนวนของเงื่อนไขในการใช้งานแอปพลิเคชันต่างๆ ที่นำมาใช้ทดลองจึงยังมีจำนวนไม่มากตามปัญหาที่พบเจอจริง ดังนั้น จึงควรเพิ่มจำนวนแอปพลิเคชัน จำนวนผู้ใช้ และจำนวนของเงื่อนไขในการใช้งานแอปพลิเคชันให้มีจำนวนเพิ่มมากขึ้น เพื่อที่สามารถทดสอบในเรื่องของประสิทธิภาพในการทำงานได้สมจริงมากขึ้น และเพิ่มความน่าเชื่อถือที่สูงขึ้น กอปรกับข้อจำกัดด้านทรัพยากรในระบบนี้ที่ถูกพัฒนาบนคอมพิวเตอร์หนึ่งเครื่องที่ใช้การจำลองซอฟต์แวร์ของระบบงานขึ้นมา ทำให้ประสิทธิภาพการทำงานในระบบจริงอาจคลาดเคลื่อนได้ โดยมีข้อสังเกตคือ เมื่อมีการใช้งานที่มากขึ้น ตัวจัดการความมั่นคงทำงานช้าลงเพราะต้องรองรับการทำงานของทั้งระบบ ซึ่งอาจต้องมีการเพิ่มเซอร์วิสย่อยเพื่อเป็นการกระจายโหลด เพื่อลดปัญหาคอขวดที่เกิดขึ้น

6.3 แนวทางการวิจัยในอนาคต

ควรนำโปรโตคอล gRPC มาเปรียบเทียบกับ REST APIs.เดิมที่มี เพื่อเปรียบเทียบประสิทธิภาพทั้งด้านการทำงานและความความมั่นคงปลอดภัยต่อระบบ โดยอาจใช้การเข้ารหัสข้อมูลขั้นสูงขึ้นเพื่อเพิ่มความน่าเชื่อถือและความมั่นใจต่อซอฟต์แวร์ที่พัฒนาได้มากขึ้น และขยายขนาดระบบเพื่อทดสอบประสิทธิภาพให้แม่นยำมากยิ่งขึ้น

บรรณานุกรม

1. Balalaie, A., A. Heydarnoori, and P. Jamshidi, *Microservices architecture enables devops: Migration to a cloud-native architecture.* Ieee Software, 2016. **33**(3): p. 42-52.
2. Institute, A.N.S., *Role Based Access Control*, in *for information technology*. 2004.
3. Tang, B., R. Sandhu, and Q. Li, *Multi-tenancy authorization models for collaborative cloud services.* Concurrency and Computation: Practice and Experience, 2015. **27**(11): p. 2851-2868.
4. Indrasiri, K. *Microservices in Practice*. 2015 [cited 2021 Mar]; Available from: http://kasunpanorama.blogspot.com/2015/11/microservices-in-practice.html.
5. Rissanen, E., *XACMLv3. 0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0.* OASIS Standard, 2014.
6. Bouchahda, A., et al. *Rbac+: Dynamic access control for rbac-administered web-based databases*. in *2010 Fourth International Conference on Emerging Security Information, Systems and Technologies*. 2010. IEEE.
7. Fugkeaw, S., P. Manpanpanich, and S. Juntapremjitt. *A development of multi-SSO authentication and RBAC model in the distributed systems*. in *2007 2nd International Conference on Digital Information Management*. 2007. IEEE.
8. Lepro, R., *Cardea: Dynamic access control in distributed systems.* System, 2004. **13**(13): p. 4-2.
9. Jones, M., J. Bradley, and N. Sakimura, *RFC7519–JSON Web Token (JWT). IETF. May 2015*.

ภาคผนวก

ภาคผนวก ก
ตัวอย่างการกำหนดความสัมพันธ์ระหว่างชื่อผู้ใช้และบทบาทแบบ Rule Based

Rule.1.User=user1

Rule.1.Role=**Admin**1

Rule.2.User=user2

Rule.2.Role=**Admin**2

Rule.3.User=user3

Rule.3.Role=Developer1

Rule.4.User=user4

Rule.4.Role=Developer2

Rule.5.User=user5

Rule.5.Role=Auditor1

Rule.6.User=user6

Rule.6.Role=Auditor2

Rule.7.User=user7

Rule.7.Role=User1

Rule.8.User=user8

Rule.8.Role= User2

ภาคผนวก ข

ตัวอย่างการกำหนดสิทธิเข้าใช้งานแบบ Rule Based

Rule.1.Role=Admin1	Rule.6.Role=Admin2
Rule.1.Permission=canAddUser	Rule.6.Permission=canDeleteUser
Rule.1.ConstraintType=location	Rule.6.ConstraintType=time
Rule.1.ConstraintValue=192.168.10.1	Rule.6.ConstraintValue=0800-1900
Rule.2.Role=Admin1	Rule.7.Role=Admin2
Rule.2.Permission=canAddUser	Rule.7.Permission=canViewReports
Rule.2.ConstraintType=time	Rule.7.ConstraintType=time
Rule.2.ConstraintValue=0800-1900	Rule.7.ConstraintValue=0800-1900
Rule.3.Role=Admin1	Rule.8.Role=Admin2
Rule.3.Permission=canUpdateUser	Rule.8.Permission=canViewReports
Rule.3.ConstraintType=location	Rule.8.ConstraintType=time
Rule.3.ConstraintValue=192.168.10.1	Rule.8.ConstraintValue=0800-1900
Rule.4.Role=Admin1	Rule.9.Role=User1
Rule.4.Permission=canUpdateUser	Rule.9.Permission=canViewContent
Rule.4.ConstraintType=time	Rule.10.Role=User1
Rule.4.ConstraintValue=0800-1900	Rule.10.Permission=CanAddContent
Rule.5.Role=Admin2	Rule.11.Role=User2
Rule.5.Permission=canDeleteUser	Rule.11.Permission=canViewContent
Rule.5.ConstraintType=location	Rule.12.Role=User2
Rule.5.ConstraintValue=192.168.10.1	Rule.12.Permission=CanAddContent

ภาคผนวก ค

ตัวอย่างการกำหนดความสัมพันธ์ระหว่างชื่อผู้ใช้และบทบาทแบบ XML

```
<Policy>
    <Rule id="1">
        <User>
            user1
        </User>
        <Role>
            Administrator1
        </Role>
    </Rule>
    <Rule id="2">
        <User>
            user2
        </User>
        <Role>
            Administrator2
        </Role>
    </Rule>
    <Rule id="3">
        <User>
            user3
        </User>
        <Role>
            Developer1
        </Role>
        <Rule id="4">
            <User>
                user2
            </User>
            <Role>
                Developer2
            </Role>
        </Rule>
        <Rule id="5">
            <User>
                user5
            </User>
            <Role>
                Auditor1
            </Role>
        </Rule>
        <Rule id="6">
            <User>
                user6
            </User>
```

```
            <Role>
                Auditor2
            </Role>
        </Rule>
        <Rule id="7">
            <User>
                user7
            </User>
            <Role>
                User1
            </Role>
        </Rule>
        <Rule id="8">
            <User>
                user8
            </User>
            <Role>
                User2
            </Role>
        </Rule>
    </Rule>
</Policy>
```

ภาคผนวก ง

ตัวอย่างการกำหนดสิทธิเข้าใช้งานแบบ XML

```
<Policy>
    <Rule id="1">
        <Role>Administrator1</Role>
        <Permission>canDeleteUser</Permission>
        <ConstraintType>location</ConstraintType>
        <ConstraintValue>192.168.100.1</ConstraintValue>
    </Rule>
    <Rule id="2">
        <Role>Developer1</Role>
        <Permission>canEditContent</Permission>
        <ConstraintType>time</ConstraintType>
        <ConstraintValue>0800-1900</ConstraintValue>
    </Rule>
    <Rule id="3">
        <Role>Auditor1</Role>
        <Permission>canAuditPeople</Permission>
        <ConstraintType>location</ConstraintType>
        <ConstraintValue>192.168.100.1</ConstraintValue>
    </Rule>
    <Rule id="4">
        <Role>User1</Role>
        <Permission>canViewContent</Permission>
        <ConstraintType></ConstraintType>
        <ConstraintValue></ConstraintValue>
    </Rule>
</Policy>
```

ประวัติผู้เขียน

ชื่อ-สกุล	จิตติภัทร ผสมทรัพย์
วัน เดือน ปี เกิด	4 ตุลาคม 2530
สถานที่เกิด	พิษณุโลก
วุฒิการศึกษา	วิทยาศาสตรบัณฑิต
ที่อยู่ปัจจุบัน	
ผลงานตีพิมพ์	C. Pasomsup and Y. Limpiyakorn, " HT-RBAC: A Design of Role-based Access Control Model for Microservice Security Manager" ในรายงานประชุมวิชาการนานาชาติสืบเนื่องจาก ACM 2021 2nd International Conference on Information System and System Management (ISSM 2021), Aug 12-14, 2021, Guizhou, China, pp.1-5.