

Logfile-Based SQL Injection Detection and Prevention System for PHP Web Applications

Chanakit Mitsongkore^{1,*}

^{1,*}Program in Digital Content Design, Faculty of Humanities and Social Sciences, Kanchanaburi Rajabhat University, Kanchanaburi 71190, Thailand; Email: chanakit2004@kru.ac.th

**Corresponding author:* Chanakit Mitsongkore (chanakit2004@kru.ac.th)

Received: 19 February 2026; **Revised:** 30 March 2026;

Accepted: 2 April 2026; **Published:** 30 April 2026

Abstract

This research aims to 1) analyze real-world SQL Injection attack patterns from system Logfile, 2) design and develop an automated SQL Injection detection and prevention system using rule-based techniques, and 3) evaluate the system's performance. Methodology The study was conducted in three phases 1) collecting and analyzing 4,649 historical log records to establish detection rules 2) validating the algorithm with a new dataset of 4,312 records and 3) evaluating performance using TP, FP, FN, and TN metrics through a real-world deployment on the Cultural Map Thailand information system over a period of 830 days. The findings demonstrate that the system achieved up to 100% Accuracy and F1-Score across all testing phases under controlled conditions. During the practical implementation, the system successfully detected a total of 25,660 attack attempts. Furthermore, the results indicate a significant decrease in the average daily attack rate post-deployment, illustrating the system's deterrent effect and its capability to effectively identify complex evasion techniques. This approach significantly enhances the long-term security posture of web-based information systems.

Keywords: SQL Injection Detection, Logfile Analysis, Weight-based Scoring, Web Application Security



Copyright © 2026 Journal of Computer and Creative Technology. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC-BY-NC-ND).

Citation:

Mitsongkore, C. (2026). Logfile-Based SQL Injection Detection and Prevention System for PHP Web Applications. *Journal of Computer and Creative Technology*, 4(1), e3560. <https://doi.org/10.65205/jcct.2026.e3560>

ระบบตรวจจับและป้องกันการโจมตีด้วยการแทรกคำสั่งเอสคิวแอลจากข้อมูล ไฟล์บันทึกเหตุการณ์ สำหรับเว็บแอปพลิเคชันที่พัฒนาด้วยภาษาพีเอชพี

ชนกฤต มิตรสงเคราะห์^{1,*}

^{1,*}สาขาวิชาการออกแบบดิจิทัลคอนเทนต์ คณะมนุษยศาสตร์และสังคมศาสตร์ มหาวิทยาลัยราชภัฏกาญจนบุรี จังหวัด
กาญจนบุรี 71190 ประเทศไทย; Email: chanakit2004@kru.ac.th

*ผู้ประพันธ์บทความ: ชนกฤต มิตรสงเคราะห์ (chanakit2004@kru.ac.th)

รับบทความ: 19 กุมภาพันธ์ 2569; แก้ไขบทความ: 30 มีนาคม 2569;
ตอบรับบทความ: 2 เมษายน 2569; เผยแพร่บทความ: 30 เมษายน 2569

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อ 1) เพื่อศึกษารูปแบบการโจมตี SQL Injection ที่เกิดขึ้นจริงจากข้อมูล Logfile ของระบบ 2) เพื่อออกแบบและพัฒนาระบบตรวจจับและป้องกัน SQL Injection แบบอัตโนมัติด้วยเทคนิค Rule-based 3) เพื่อประเมินประสิทธิภาพของระบบ วิธีดำเนินการวิจัยแบ่งออกเป็น 3 ระยะ เริ่มจาก 1) การรวบรวมและวิเคราะห์ข้อมูลจาก Logfile จริง ในอดีตจำนวน 4,649 รายการ เพื่อสร้างกฎการตรวจจับ 2) การทดสอบความถูกต้องกับชุดข้อมูลใหม่จำนวน 4,312 รายการ และ 3) ประเมินประสิทธิภาพของระบบด้วยตัวชี้วัด TP, FP, FN และ TN โดยประเมินผลจากการใช้งานจริงบนระบบสารสนเทศด้านวัฒนธรรมและแผนที่วัฒนธรรม เป็นเวลา 830 วัน ผลการวิจัย พบว่าระบบมีความแม่นยำ (Accuracy) และค่า F1-Score สูงถึงร้อยละ 100 ในทุกระยะการทดสอบ โดยในระยะใช้งานจริงสามารถตรวจจับความพยายามในการโจมตีได้ทั้งสิ้น 25,660 ครั้ง นอกจากนี้ ผลการวิจัยชี้ให้เห็นว่าอัตราการโจมตีเฉลี่ยต่อวันลดลงอย่างมีนัยสำคัญภายหลังการติดตั้งระบบ ซึ่งแสดงถึงประสิทธิภาพในการยับยั้งการโจมตี และความสามารถในการตรวจจับเทคนิคการหลบเลี่ยง ได้อย่างมีประสิทธิภาพ ช่วยยกระดับความมั่นคงปลอดภัยให้กับระบบสารสนเทศในระยะยาว

คำสำคัญ: การตรวจจับการโจมตีด้วยการแทรกคำสั่งเอสคิวแอล, การวิเคราะห์ไฟล์บันทึกเหตุการณ์, การให้คะแนนน้ำหนัก, ความมั่นคงปลอดภัยเว็บแอปพลิเคชัน



Copyright © 2026 Journal of Computer and Creative Technology. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC-BY-NC-ND).

การอ้างอิง:

ชนกฤต มิตรสงเคราะห์. (2569). ระบบตรวจจับและป้องกันการโจมตีด้วยการแทรกคำสั่งเอสคิวแอลจากข้อมูลไฟล์บันทึกเหตุการณ์ สำหรับเว็บแอปพลิเคชันที่พัฒนาด้วยภาษาพีเอชพี. วารสารคอมพิวเตอร์และเทคโนโลยีสร้างสรรค์, 4(1), e3560.
<https://doi.org/10.65205/jcct.2026.e3560>

1. บทนำ (Introduction)

ในปัจจุบันระบบสารสนเทศมีการเผยแพร่ผ่านอินเทอร์เน็ต เว็บแอปพลิเคชันได้กลายเป็นโครงสร้างพื้นฐานสำคัญ เช่น ระบบสารสนเทศด้านวัฒนธรรมและแผนที่วัฒนธรรม (culturalmapthailand.info) ที่พัฒนาด้วยภาษา PHP และ MySQL เพื่อเผยแพร่ข้อมูลงานวิจัยสู่สาธารณะ อย่างไรก็ตาม การให้บริการผ่านเครือข่ายสาธารณะทำให้ระบบพบกับภัยคุกคามทางไซเบอร์อย่างต่อเนื่อง โดยเฉพาะการโจมตีแบบ SQL Injection ซึ่งสอดคล้องกับรายงานสถิติของสำนักงานคณะกรรมการการรักษาความมั่นคงปลอดภัยไซเบอร์แห่งชาติ (Securities and Exchange Commission, 2023) ที่ระบุว่า การโจมตีเว็บแอปพลิเคชันยังคงเป็นปัญหาหลักของหน่วยงานในประเทศไทย

จากรายงาน OWASP Top 10 (OWASP, 2021) ยืนยันว่าช่องโหว่ประเภท Injection และความบกพร่องของระบบบันทึกเหตุการณ์ (Security Logging) เป็นความเสี่ยงสำคัญที่อาจทำให้องค์กรไม่สามารถตรวจจับการโจมตีได้ทันทั่วทั้งงานวิจัยในช่วงปี พ.ศ. 2563 ถึง 2566 แสดงให้เห็นว่าผู้โจมตีมักใช้เทคนิคหลบเลี่ยง (Evasion Attacks) เพื่อเจาะผ่านระบบป้องกันด่านหน้าอย่าง Web Application Firewall (WAF) ทำให้การป้องกันเพียงชั้นเดียวไม่เพียงพอ และจำเป็นต้องมีแนวทางป้องกันเชิงลึกที่มีประสิทธิภาพมากขึ้น

จากการตรวจสอบ Logfile ของระบบ culturalmapthailand.info ระหว่างวันที่ 25 มีนาคม ถึง 24 เมษายน 2566 พบข้อมูลบันทึก 91,027 รายการ และตรวจพบความพยายามโจมตีแบบ SQL Injection มากกว่า 4,000 ครั้ง สะท้อนว่าภัยคุกคามนี้เกิดขึ้นจริงไม่ใช่แค่เพียงทฤษฎี (OWASP, 2021) แม้ระบบจะมีการเก็บ Logfile แต่การบันทึกข้อมูลทุกรายการโดยไม่คัดกรอง จะสร้างภาระต่อพื้นที่จัดเก็บข้อมูลและยากต่อการวิเคราะห์เชิงลึก

ดังนั้น งานวิจัยนี้จึงมุ่งพัฒนา กลไกตรวจจับและป้องกันการโจมตีแบบ SQL Injection โดยใช้แนวทาง Rule-based Detection เพื่อสกัดกั้นคำร้องขอที่ผิดปกติผ่านการกำหนดรูปแบบคำสั่ง (Patterns) ซึ่งเป็นวิธีการที่เหมาะสมสำหรับระบบที่มีทรัพยากรจำกัด ช่วยเพิ่มประสิทธิภาพให้ระบบบันทึกเหตุการณ์สามารถนำไปใช้วิเคราะห์และตอบสนองต่อภัยคุกคามได้อย่างเป็นรูปธรรม

2. วัตถุประสงค์งานวิจัย (Research Objectives)

- 2.1 เพื่อศึกษารูปแบบการโจมตี SQL Injection ที่เกิดขึ้นจริงจากข้อมูล Logfile ของระบบ
- 2.2 เพื่อออกแบบและพัฒนาระบบตรวจจับและป้องกัน SQL Injection แบบอัตโนมัติด้วยเทคนิค Rule-based
- 2.3 เพื่อประเมินประสิทธิภาพของระบบด้วยตัวชี้วัด TP, FP, FN และ TN

3. ขอบเขตของการวิจัย (Research Scopes)

งานวิจัยนี้มุ่งเน้นการพัฒนาระบบตรวจจับการโจมตี SQL Injection ในระดับเว็บแอปพลิเคชัน โดยใช้ภาษา PHP เวอร์ชัน 5.6 ขึ้นไป และฐานข้อมูล MySQL เพื่อให้สามารถใช้งานร่วมกับระบบ culturalmapthailand.info ระบบจะทำการตรวจสอบคำร้องขอจากผู้ใช้ก่อนเข้าสู่กระบวนการประมวลผลหลัก การณที่ตรวจพบรูปแบบที่เข้าข่ายการโจมตี ระบบจะยุติการแสดงผลของหน้าที่ถูกเรียกใช้งาน และดำเนินการตามกลไกที่กำหนดไว้

การวิจัยนี้ไม่ครอบคลุมการประยุกต์ใช้เทคนิค Machine Learning หรือการวิเคราะห์เชิงพฤติกรรมขั้นสูง แต่เน้นการพัฒนาแนวทางที่สามารถนำไปใช้งานได้จริงในระบบที่มีทรัพยากรจำกัด และสามารถประยุกต์ใช้กับเว็บแอปพลิเคชันทั่วไปได้

4. การทบทวนวรรณกรรมและทฤษฎีที่เกี่ยวข้อง (Literature Review)

4.1 ภัยคุกคามด้าน SQL Injection และระบบบันทึกเหตุการณ์ (OWASP)

เว็บแอปพลิเคชันเป็นโครงสร้างพื้นฐานสำคัญที่เสี่ยงต่อภัยคุกคามไซเบอร์ โดยรายงาน OWASP Top 10 (OWASP, 2021) จัดให้ Injection เป็นความเสี่ยงอันดับต้น ซึ่งเกิดจากการที่ระบบไม่สามารถแยก “ข้อมูล” ออกจาก “คำสั่ง” ทำให้ผู้โจมตีสามารถแทรกคำสั่ง SQL เพื่อเข้าถึงหรือแก้ไขฐานข้อมูลได้ นอกจากนี้ยังเน้นย้ำถึง Security Logging and Monitoring Failures ว่าการขาดระบบบันทึกและวิเคราะห์เหตุการณ์ที่เหมาะสม จะส่งผลให้องค์กรไม่สามารถตรวจพบหรือรับมือกับการโจมตีได้ทันทั่วทั้ง

4.2 ลักษณะการโจมตีและความท้าทายในการป้องกัน

การโจมตีแบบ SQL Injection มักกระทำผ่าน URL Parameters หรือ Form Inputs หากขาดการตรวจสอบที่รัดกุม แม้จะมีการใช้ Web Application Firewall (WAF) แต่การศึกษาพบว่าผู้โจมตีสามารถใช้เทคนิคหลบเลี่ยง (Evasion Attacks) เช่น การเข้ารหัสคำสั่ง เพื่อเจาะผ่านระบบตรวจจับแบบ Signature-based ได้ (Kaur et al., 2023) แนวทางการป้องกันที่มีประสิทธิภาพจึงต้องอาศัยการปรับแต่งกฎเกณฑ์ให้เหมาะสมกับบริบทของแอปพลิเคชัน (Mutedi & Tjahjono, 2022) สอดคล้องกับงานวิจัยนี้ที่มุ่งเน้นการตรวจจับจากข้อมูล Logfile โดยใช้กลไกการให้น้ำหนักคะแนน (Scoring Mechanism) เพื่อลดข้อผิดพลาดและเพิ่มความยืดหยุ่น

4.3 แนวทางการตรวจจับด้วย Machine Learning

ช่วงปี 2020-2023 มีการใช้ Machine Learning (ML) เพื่อเพิ่มประสิทธิภาพการตรวจจับ เช่น การใช้ Naïve Bayes ร่วมกับเทคนิค SMOTE เพื่อแก้ปัญหาชุดข้อมูลไม่สมดุล (Arnap & Kusirini, 2022) และการเปรียบเทียบอัลกอริทึมประเภท Support Vector Machine (SVM) หรือ Random Forest ซึ่งให้ค่าความแม่นยำสูง (Alghawazi et al., 2022) นอกจากนี้ Fu et al. (2023) พบว่า การวิเคราะห์พฤติกรรม การรับส่งข้อมูลทั้งขาเข้าและขาออกช่วยให้แยกแยะเจตนาแฝงของผู้โจมตีได้ดีขึ้น อย่างไรก็ตาม วิธีการกลุ่ม Machine Learning (ML) ยังมีข้อจำกัดด้านการใช้ทรัพยากรระบบและความซับซ้อนในการฝึกโมเดลอย่างต่อเนื่อง

Pan et al. (2019) นำเสนอแนวทางการตรวจจับการโจมตีเว็บแอปพลิเคชันด้วย End-to-End Deep Learning โดยใช้เครื่องมือที่เรียกว่า Robust Software Modeling Tool (RSMT) ซึ่งทำการบันทึก Call Graph ของโปรแกรมขณะทำงานจริง แล้วนำมาฝึก Stacked Denoising Autoencoder เพื่อเรียนรู้รูปแบบการทำงานปกติ เมื่อค่าร้องขอใดให้ค่า Reconstruction Error สูงเกินเกณฑ์ที่กำหนด ระบบจะจัดว่าเป็นการโจมตี วิธีการนี้สามารถตรวจจับได้ทั้ง SQL Injection, XSS และ Object Deserialization โดยไม่ต้องกำหนด Feature ด้วยมือ ซึ่งแนวทางดังกล่าวต้องการข้อมูลฝึกสอนจำนวนมากและมีความซับซ้อนในการติดตั้งสูง รวมถึงต้องการการเข้าถึง Runtime ของแอปพลิเคชันในระดับ JVM ซึ่งไม่เหมาะสมสำหรับระบบที่มีทรัพยากรจำกัดอย่างระบบ PHP/MySQL ที่ใช้ในงานวิจัยนี้

Das et al. (2019) นำเสนอวิธีตรวจจับ SQL Injection โดยใช้ Edit-distance เปรียบเทียบโครงสร้างของ Dynamic SQL Query กับ Profile ที่เรียนรู้ในช่วง Training phase ซึ่งสามารถตรวจจับได้ทั้งการโจมตีแบบ Injected-in และ Injected-out บน PHP/MySQL โดยไม่พบ False Positive ในชุดข้อมูลจำลอง ซึ่งงานวิจัยดังกล่าวใช้ชุดข้อมูลที่สร้างขึ้นเองในสภาพแวดล้อมจำลอง ไม่ใช่ข้อมูลจากระบบที่ให้บริการจริง

นอกจากนี้ Muhammad & Ghafory (2022) ได้เปรียบเทียบอัลกอริทึม Machine Learning หลายประเภท ได้แก่ Logistic Regression (LRN), Sequential Minimal Optimization (SMO), Instance-Based Learner (IBK), Naive Bayes (NB), J48 (Decision Tree) และอื่นๆ สำหรับการตรวจจับ SQL Injection โดยเฉพาะ โดยใช้วิธีประเมินแบบ 10-Fold Cross Validation ผลพบว่า SMO ให้ความแม่นยำสูงสุดที่ร้อยละ 98.78 อย่างไรก็ตาม งานวิจัยนี้ยังใช้ชุดข้อมูลจำลองและไม่ได้ประเมินในสภาวะการใช้งานจริงระยะยาว

4.4 การเปรียบเทียบงานวิจัยที่เกี่ยวข้องกับงานวิจัยนี้

จากการทบทวนวรรณกรรมในหัวข้อ 4.1-4.3 พบว่า งานวิจัยที่ผ่านมาสามารถแบ่งออกได้เป็น 3 กลุ่มหลักตามแนวทางที่ใช้ ได้แก่ กลุ่มที่ใช้ Machine Learning กลุ่มที่ใช้ Deep Learning และกลุ่มที่ใช้ Rule-based หรือ Signature-based Detection โดยแต่ละกลุ่มมีจุดแข็งและข้อจำกัดที่แตกต่างกัน ซึ่งส่งผลต่อความเหมาะสมในการนำไปใช้กับระบบที่มีทรัพยากรจำกัด

1. กลุ่มที่ 1 Machine Learning-based Detection

งานวิจัยในกลุ่มนี้ ได้แก่ Arnap & Kusirini(2022), Alghawazi et al. (2022), Muhammad & Ghafory (2022) และ Kaur et al. (2023) ซึ่งแสดงให้เห็นว่า ML สามารถให้ความแม่นยำสูงในการตรวจจับ SQL Injection ได้ แต่เมื่อวิเคราะห์เชิงลึกพบว่างานวิจัยทั้งหมดในกลุ่มนี้ดำเนินการบนชุดข้อมูล Benchmark หรือชุดข้อมูลจำลอง ไม่ใช่ข้อมูลจาก Logfile ของระบบที่เปิดให้บริการจริง นอกจากนี้ยังมีข้อจำกัดเหมือนกัน คือต้องการทรัพยากรระบบสูงในการฝึกโมเดล และเมื่อรูปแบบการโจมตีเปลี่ยนแปลงจะต้องดำเนินการ Retrain โมเดลใหม่ ซึ่งไม่เหมาะสมกับระบบสารสนเทศขนาดเล็กที่มีทรัพยากรจำกัด อีกทั้งงานวิจัยเหล่านี้ยังไม่ได้รายงานค่า False Positive ในสภาวะการใช้งานจริงระยะยาว

2. กลุ่มที่ 2 Deep Learning-based Detection

Pan et al. (2019) และ Fu et al. (2023) เป็นตัวแทนของแนวทาง Deep Learning ที่ก้าวหน้ากว่า โดย Pan et al. (2019) ใช้ Stacked Denoising Autoencoder วิเคราะห์ Call Graph ของโปรแกรม ซึ่งมีข้อได้เปรียบที่ไม่ต้องกำหนด Feature ด้วยมือและครอบคลุมการโจมตีหลายประเภท ส่วน Fu et al. (2023) วิเคราะห์ Traffic ทั้งขาเข้าและขาออก พร้อมกัน ช่วยระบุเจตนาของผู้โจมตีได้แม่นยำขึ้น ซึ่งทั้งสองงานมีข้อจำกัดที่สำคัญคือต้องการโครงสร้างพื้นฐานที่ซับซ้อน โดย Pan et al. (2019) ต้องเข้าถึง JVM Runtime ซึ่งไม่รองรับ PHP/MySQL และ Fu et al. (2023) ต้องการการเข้าถึง Outbound Traffic ที่ไม่สามารถทำได้ในทุกสภาพแวดล้อม

Das et al. (2019) ได้นำเสนอเทคนิคการเปรียบเทียบโครงสร้างคำสั่ง (Edit-distance) ซึ่งมีความโดดเด่นในการ ลดอัตราการแจ้งเตือนผิดพลาด (False Positive) จนเป็นศูนย์ได้ในสภาพแวดล้อมจำลอง ซึ่งงานวิจัยดังกล่าวยังคงมีข้อจำกัดเนื่องจากใช้ชุดข้อมูลที่สร้างขึ้นเอง (Synthetic Dataset) ไม่ใช่ข้อมูลจาก Logfile ของระบบที่เปิดให้บริการจริง ซึ่งอาจไม่สะท้อนความซับซ้อนของพฤติกรรมผู้ใช้ในสถานการณ์จริง

3. กลุ่มที่ 3 Rule-based และ WAF-based Detection

Mutedi & Tjahjono (2022) ทบทวนการใช้ WAF ซึ่งเป็นแนวทางที่ใกล้เคียงกับงานวิจัยนี้มากที่สุด ผลการทบทวนพบว่า WAF ที่ใช้กฎทั่วไปสามารถถูกเลี่ยงด้วยเทคนิคการหลบเลี่ยง (Evasion) ได้ ซึ่งเป็นจุดอ่อนที่งานวิจัยนี้มุ่งแก้ไขด้วยการออกแบบกฎตรวจจับจาก Logfile จริงของระบบ และเสริมด้วยกลไกการให้คะแนน (Scoring Mechanism) ที่พิจารณาบริบทของข้อมูลแทนการตรวจสอบเพียงคำสำคัญ (Keyword) เพื่อให้ระบบมีความยืดหยุ่นและสามารถรับมือกับเทคนิคการหลบเลี่ยงได้ดีกว่า WAF ทั่วไป

4.5 ช่องว่างของงานวิจัยที่ผ่านมา (Research Gap)

เมื่อเปรียบเทียบงานวิจัยทั้งหมด พบว่ายังไม่มีงานวิจัยใดที่ครอบคลุมองค์ประกอบครบทั้ง 4 ด้านพร้อมกัน ได้แก่ (1) ใช้ข้อมูล Logfile จากระบบจริงที่เปิดให้บริการ (2) ประเมินผลในสภาวะการใช้งานจริงระยะยาว (3) ความสามารถในการควบคุมและลดอัตราการแจ้งเตือนผิดพลาด (False Positive) ให้เหลือน้อยที่สุดหรือเป็นศูนย์ภายใต้เงื่อนไขของระบบงานจริง และ (4) เหมาะสมกับระบบที่มีทรัพยากรจำกัด

งานวิจัยนี้จึงมุ่งปิดช่องว่างดังกล่าวด้วยการพัฒนาระบบ Rule-based Detection ที่ผสมผสาน Whitelist Filtering และ Scoring Mechanism โดยประเมินผลบนระบบ culturalmapthailand.info

4.6 บทบาทของ Logging และช่องว่างของงานวิจัย

การบันทึกเหตุการณ์ (Logging) ที่ดีต้องมาจากสภาพแวดล้อมการใช้งานจริงมากกว่าชุดข้อมูลจำลอง เพื่อให้สะท้อนรูปแบบการโจมตีที่เกิดขึ้นจริง (Fu et al, 2023) จากการทบทวนวรรณกรรมพบว่า งานวิจัยส่วนใหญ่เน้นการเพิ่มความแม่นยำของแบบจำลอง ML (Arnap & Kusriani, 2022; Alghawazi et al., 2022) หรือการประเมินประสิทธิภาพของเครื่องมือสำเร็จรูป (Kaur et al., 2023; Mutedi & Tjahjono, 2022) แต่ยังมีงานวิจัยจำนวนน้อยที่พัฒนากฎตรวจจับ (Rule-based) จาก Logfile ของระบบจริงที่เปิดให้บริการ

4.7 ทิศทางของงานวิจัยนี้

งานวิจัยนี้จึงมุ่งใช้ข้อมูล Logfile จากระบบจริงเพื่อออกแบบกฎตรวจจับ SQL Injection ที่สอดคล้องกับพฤติกรรมการโจมตี โดยประเมินผลด้วยตัวชี้วัดมาตรฐาน ได้แก่ True Positive (TP), False Positive (FP), False Negative (FN) และ True Negative (TN) เพื่อพัฒนากลไกที่มีความสมดุลระหว่างความแม่นยำและความเรียบง่าย เหมาะสำหรับระบบที่มีข้อจำกัดด้านทรัพยากร

4.8 สมมติฐานการวิจัย (Research Hypothesis)

กลไกการให้คะแนนน้ำหนักความเสี่ยง (Scoring Mechanism) ร่วมกับ Whitelist Filtering สามารถแยกคำร้องขอปกติออกจากคำสั่งโจมตีได้ โดยไม่เกิดการแจ้งเตือนผิดพลาด (False Positive) ในสภาวะการใช้งานจริงโดยแบ่งการดำเนินงานออกเป็น 2 ระยะ ดังนี้

ระยะที่ 1 วิเคราะห์รูปแบบการโจมตีจาก Logfile ที่เก็บไว้ในอดีต เพื่อสร้างกฎตรวจจับ และทดสอบกับ Logfile ในช่วงเวลาถัดไปที่ระบบไม่เคยเห็นมาก่อน

ระยะที่ 2 นำระบบที่ผ่านการทดสอบไปติดตั้งและใช้งานจริง เพื่อประเมินประสิทธิภาพในสภาวะการใช้งานจริง
ระยะยาว

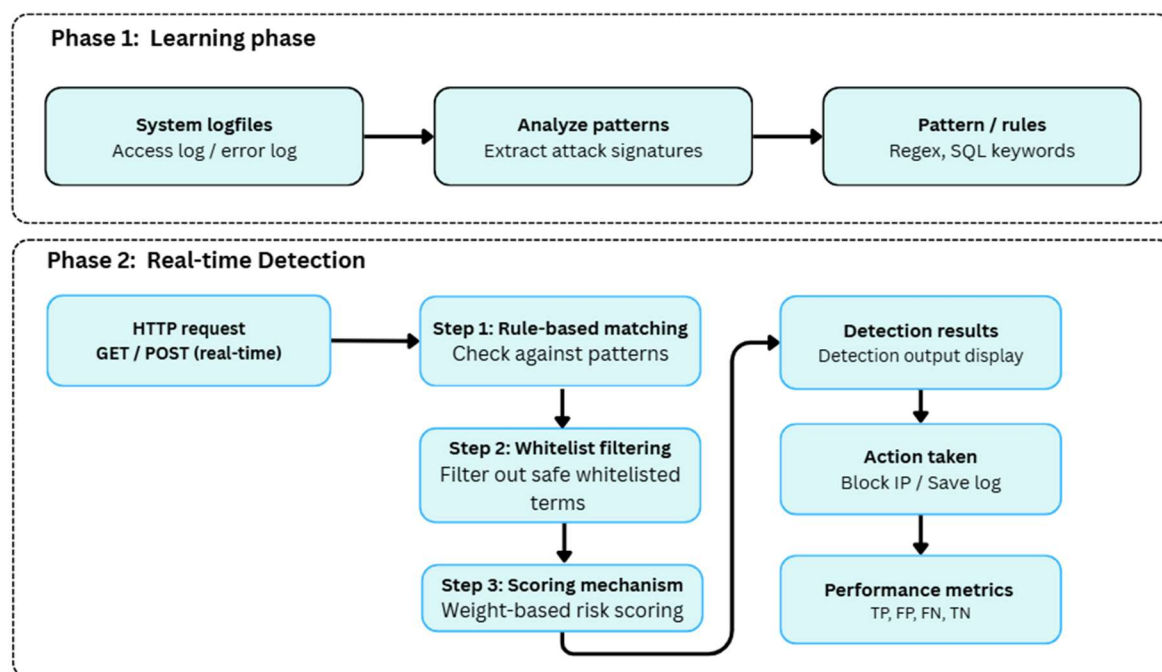


Figure 1. Two-Phase Framework for SQL Injection Detection and Prevention System

5. วิธีดำเนินการวิจัย (Methodology)

งานวิจัยนี้ดำเนินการในลักษณะการพัฒนาระบบ (System Development Research) ควบคู่กับการทดลองเชิงประจักษ์ (Empirical Evaluation) โดยอาศัยข้อมูลการเข้าใช้งานจริงของระบบสารสนเทศด้านวัฒนธรรมและแผนที่วัฒนธรรม (culturalmapthailand.info) เป็นฐานในการวิเคราะห์ ออกแบบ และทดสอบระบบตรวจจับและป้องกันการโจมตีแบบ SQL Injection กระบวนการวิจัยแบ่งออกเป็น 4 ขั้นตอนหลัก ดังนี้

5.1 การรวบรวมและวิเคราะห์ข้อมูลจาก Logfile จริง

1) แหล่งข้อมูลและชุดข้อมูลที่ใช้ในการวิจัย (Dataset) มาจากระบบบันทึกเหตุการณ์ (Logfile) ที่พัฒนาขึ้นเองภายในระบบ culturalmapthailand.info เพื่อเก็บข้อมูลคำร้องขอ (HTTP Requests) ที่เข้ามายังเว็บแอปพลิเคชัน ข้อมูลถูกเก็บในช่วงระหว่างวันที่ 25 มีนาคม พ.ศ. 2566 ถึง 24 เมษายน พ.ศ. 2566 รวมระยะเวลา 1 เดือน (31 วัน) มีจำนวนข้อมูลทั้งสิ้น 91,027 รายการ โดยในจำนวนดังกล่าวพบคำร้องขอที่เข้าข่าย SQL Injection มากกว่า 4,000 รายการ

2) โครงสร้างข้อมูลถูกจัดเก็บในฐานข้อมูล MySQL ตาราง Logfileview โดยมีฟิลด์หลัก ได้แก่

1. LogId = รหัสลำดับข้อมูล
2. ConPage = ค่าพารามิเตอร์ URL หรือ Query String ที่ถูกเรียกใช้งาน
3. is_attack = ค่าป้ายกำกับ (1 = SQL Injection, 0 = ปกติ)

3) การกำหนดป้ายกำกับ (Labeling) การกำหนดค่า is_attack ดำเนินการโดย

1. วิเคราะห์โครงสร้าง ชุดคำสั่งอันตราย (Payload) ที่มีลักษณะคำสั่ง SQL แทรกอยู่
2. ตรวจสอบรูปแบบที่สอดคล้องกับเทคนิค SQL Injection ที่เป็นที่ยอมรับและพบจริงในระบบ

3. ทดสอบซ้ำกับฐานข้อมูลจำลองเพื่อยืนยันความแม่นยำ โดยรูปแบบที่ใช้เป็นเกณฑ์ในการ Labeling สอดคล้องกับมาตรฐาน OWASP SQL Injection Prevention Cheat Sheet และเทคนิคที่ได้รับการยืนยันในวรรณกรรม ได้แก่ UNION-based, Boolean-based, Time-based และ Error-based Injection ซึ่งเป็นเกณฑ์เดียวกับที่ Alghawazi et al.

(2022) และ Fu et al. (2023) ใช้ในงานวิจัยของตน ทำให้การกำหนดป้ายกำกับในงานวิจัยนี้มีความสอดคล้องกับมาตรฐานที่ยอมรับในระดับสากล

ConPage	LiDate 1
/?	2023-04-23 22:15:29
param=-1+UNION+SELECT+GROUP_CONCAT(table_name)+F...	2023-04-23 22:18:34
/index-content-detail.php?Title=if(now())=sysdate()...	2023-04-23 22:18:52
/index-content-detail.php?Title=0"XOR(if(now())=sys...	2023-04-23 22:19:09
/index-content-detail.php?Title=if(now())=sysdate()...	2023-04-23 22:19:31
/index-content-detail.php?Title=0"XOR(if(now())=sys...	2023-04-23 22:22:18
/index-cul.php?content=cookie-detail"%26%26sleep(2...	2023-04-23 22:22:42
/index-cul.php?content=cookie-detail"%7C%7Csleep(2...	2023-04-23 22:22:44
/index-content-detail.php?Title=if(now())=sysdate()...	2023-04-23 22:23:52
/index-content-detail.php?Title=if(now())=sysdate()...	2023-04-23 22:25:04
/index-cul.php?content=if(now())=sysdate()%2Csleep(...	2023-04-23 22:25:10
/index-cul.php?content=register_insert"%26%26sleep...	2023-04-23 22:25:31
/index-content-detail.php?Title=0"XOR(if(now())=sys...	

Figure 2. SQL Injection Attack Samples in Logfileview

5.2 การออกแบบกลไกตรวจจับแบบ Rule-based

1. จากการวิเคราะห์ข้อมูลจริงในช่วงที่ 1 ระยะเวลา 1 เดือน (31 วัน) พบรูปแบบการโจมตีที่หลากหลาย จึงออกแบบกฎตรวจจับ (Detection Rules) โดยอาศัย ชุดลำดับอักขระพิเศษที่ใช้กำหนดรูปแบบ (Pattern) สำหรับการค้นหา จับคู่ (Regular Expression: Regex) เป็นกลไกหลักระบบถูกออกแบบในลักษณะ Rule-based ประกอบด้วย

1) Whitelist Filtering ใช้เป็นกลไกคัดกรองเบื้องต้นที่ทำงานร่วมกับไฟล์ .htaccess เพื่ออนุญาตเฉพาะพารามิเตอร์ที่จำเป็นต่อระบบ culturalmapthailand.info ให้เข้าถึงฐานข้อมูลได้โดยตรงในขณะที่ไฟล์ .htaccess จะทำหน้าที่เป็นด่านหน้าในการจำกัดสิทธิ์การเข้าถึงไฟล์สำคัญและป้องกันการรันสคริปต์แปลกปลอมในโดเมนที่มิได้รับอนุญาต การทำงานประสานกันเช่นนี้ช่วยลดภาระการประมวลผลของ Rule-based Detection และเสริมความปลอดภัยแบบเชิงลึก (Defense in Depth) ให้กับเว็บแอปพลิเคชันอย่างมีประสิทธิภาพ

2) Pattern Matching Detection เพื่อใช้ในการตรวจจับโดยใช้กฎ (rule-based) ผ่าน Regular Expression (Regex) เพื่อค้นหารูปแบบที่เข้าข่าย SQL Injection เช่น UNION SELECT, OR 1=1, SLEEP(), sysdate(), XOR, information_schema|table_name เป็นต้น ทั้งนี้งานวิจัยนี้ได้ค้นหาและศึกษาข้อมูลรูปแบบ คำสั่ง SQL Injection อื่นๆ เพิ่มเติมนอกเหนือจากที่พบจาก Logfile ที่ถูกเก็บไว้ในระบบ culturalmapthailand.info เพื่อเพิ่มการป้องกันให้ครอบคลุมมากที่สุด

3) Scoring Mechanism เป็นกลไกในการให้ คะแนนความเสี่ยง แทนการตัดสินจากกฎเดียว โดยการสะสมคะแนนจากรูปแบบการโจมตีที่ตรวจพบในชุดคำสั่งอันตราย (Payload) เดียวกัน แต่ละรูปแบบ (Pattern) จะถูกกำหนดค่าน้ำหนักตามระดับความเสี่ยง เมื่อผลรวมคะแนนค่าที่จำกัด (Threshold) ที่กำหนดไว้ ระบบจะจัดประเภทคำร้องขอเป็นการโจมตี วิธีการดังกล่าวช่วยลดการแจ้งเตือนผิดพลาดจาก Keyword เดียว และเพิ่มความแม่นยำของระบบโดยพิจารณาบริบทโดยรวมของชุดคำสั่งอันตราย (Payload)

2. ประเภทการโจมตีที่ใช้เป็นกรอบการออกแบบกฎถูกจัดกลุ่มตามรูปแบบการโจมตีที่พบจริงและกำหนด น้ำหนักของรูปแบบแต่ละรูปแบบ (Patterns) ดังนี้

Table 1. High-Risk Group (High-Risk Structural Injection)

Pattern	Weight	Reasoning / Logic
TIME_BASED	6	A blind injection technique that confirms vulnerabilities via time delays demonstrates control over the execution flow.
STACKED_QUERY	6	Using the character to execute stacked commands constitutes full query manipulation.
UNION_BASED	5	Ability to directly retrieve data from other tables, posing a significant data exfiltration risk.
BOOLEAN_BASED	4	Employs logical conditions (Boolean-based) to bypass authentication.

จาก Table 1. เหตุผลเชิงหลักการ รูปแบบ (Pattern) กลุ่มนี้สะท้อนถึง “โครงสร้างคำสั่ง SQL ที่สมบูรณ์” และสามารถเปลี่ยนแปลงผลลัพธ์ของ Query ได้โดยตรง จึงกำหนดน้ำหนักสูง

Table 2. Medium-Risk Group (Reconnaissance/ Enumeration)

Pattern	Weight	Reasoning / Logic
SCHEMA_ENUM	3	Used to explore database structures (information_schema)
ERROR_BASED	4	Relies on error messages to reveal system information

จาก Table 2. เหตุผลเชิงหลักการ รูปแบบ (Pattern) เป็นขั้นตอนการ Reconnaissance ก่อนการโจมตีจริง มีความเสี่ยงแต่ยังไม่ใช้การแก้ไข Query โดยตรง

Table 3. Low-Risk Group (Contextual Indicators)

Pattern	Weight	Reasoning / Logic
COMMENT_BYPASS	2	Using comments to truncate the query
ENCODED	2	Encoding that may be used for detection evasion
SQL_KEYWORDS	1	Standard standalone keyword; may result from normal input

จาก Table 3. เหตุผลเชิงหลักการ รูปแบบ (Pattern) กลุ่มนี้ไม่เพียงพอในการยืนยันการโจมตีด้วยตัวเอง จึงกำหนดน้ำหนักต่ำ และต้องเกิดร่วมกับรูปแบบ (Pattern) อื่นจึงจะผ่านค่าที่จำกัด (Threshold)

การจัดกลุ่มดังกล่าวช่วยให้สามารถวิเคราะห์สถิติของแต่ละประเภทได้ในขั้นตอนประเมินผล โดยในงานนี้ตั้งค่าน้ำหนัก (Weight) ไว้ที่ 5 ขึ้นไปจะถือว่าเป็นการโจมตี

4. ชุดข้อมูลสำหรับการทดสอบระยะที่ 2 ภายหลังจากการออกแบบและพัฒนาระบบตรวจจับในระยะที่ 1 ดำเนินการทดสอบระบบกับข้อมูลใหม่ที่ไม่ได้ใช้ในการออกแบบกฎ (Unseen Dataset) เพื่อประเมินความสามารถของระบบในการทำงานกับข้อมูลจริงในช่วงเวลาถัดไป โดยใช้ข้อมูล Logfile ที่เก็บในช่วงระหว่างวันที่ 25 เมษายน พ.ศ. 2566 ถึง 24 พฤษภาคม พ.ศ. 2566 รวมระยะเวลา 30 วัน ชุดข้อมูลดังกล่าวมีจำนวนทั้งสิ้น 88,925 รายการ และถูกใช้เป็นชุดข้อมูลสำหรับการประเมินประสิทธิภาพของระบบในสภาวะแวดล้อมจริง (Real-World Validation) โดยใช้ตัวชี้วัดเดียวกับระยะที่ 1 ได้แก่ TP, FP, FN และ TN เพื่อให้สามารถเปรียบเทียบผลลัพธ์ระหว่างช่วงเวลาได้อย่างเป็นระบบ

5.3 การพัฒนาระบบป้องกันเชิงรุก

ระบบถูกติดตั้งในระดับ Application Layer ของเว็บแอปพลิเคชัน PHP โดยทำงานก่อนการประมวลผลคำสั่ง SQL โดยมีขั้นตอนการทำงานของระบบ ดัง Figure 3.

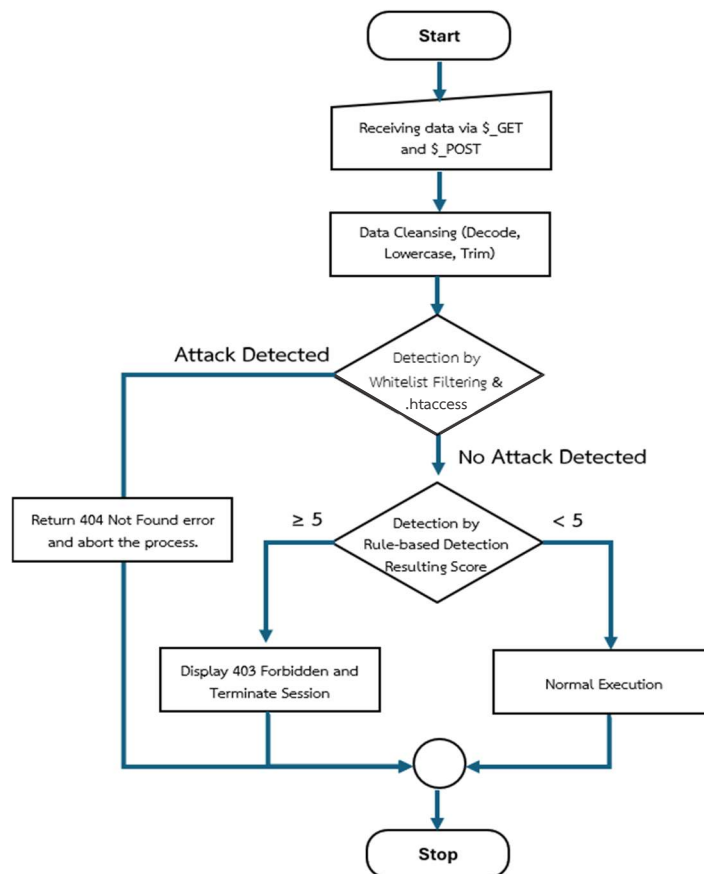


Figure 3. System Process Flowchart

จาก Figure 3. เป็นแนวคิดด้านการจัดการ Logfile เนื่องจากในช่วงการเก็บข้อมูลพบ SQL Injection จำนวนมาก หากบันทึก ชุดคำสั่งอันตราย (Payload) ที่เป็นอันตรายทั้งหมดอาจทำให้ฐานข้อมูลเติบโตอย่างรวดเร็ว งานวิจัยนี้จึงออกแบบระบบให้ทำงานดังนี้ 1. บล็อกการโจมตีทันที 2. ไม่บันทึกข้อมูลที่เป็นอันตรายเพิ่มเติม 3. ลดปัญหาข้อมูลมีปริมาณมากเกินไป (Logfile Pollution) ในระยะยาว และ 4. แนวทางดังกล่าวเน้นการป้องกันเชิงรุก (Proactive Protection) มากกว่าการเก็บข้อมูลเชิงย้อนหลัง

ทั้งนี้งานวิจัยนี้ได้กำหนดระยะเวลาการดำเนินการวิจัยไว้ 3 ระยะ ดัง Figure 4. โดย ระยะที่ 3 ยังคงทำการบันทึก Logfile ที่พบการโจมตีลงในฐานข้อมูล

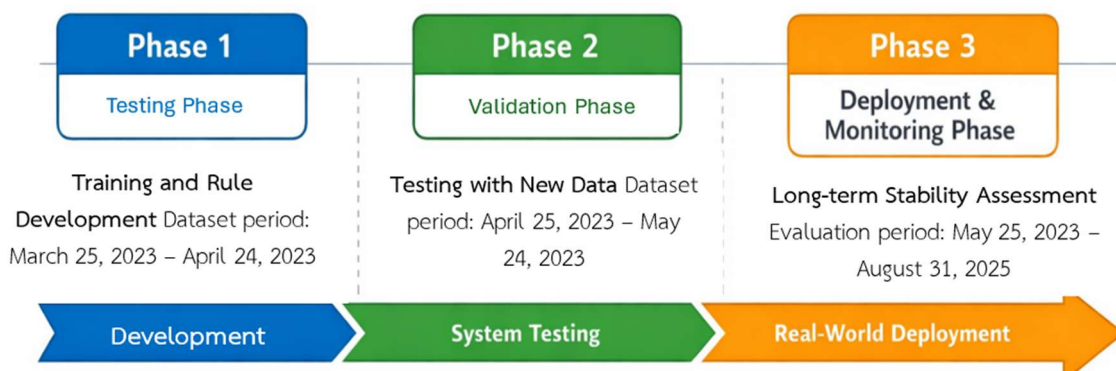


Figure 4. Dataset Timeline for Performance Evaluation

5.4 การประเมินประสิทธิภาพของระบบ

การประเมินประสิทธิภาพของระบบตรวจจับ SQL Injection ถูกออกแบบเป็น 3 ระยะ เพื่อสะท้อนวงจรการพัฒนาเชิงระบบตั้งแต่การออกแบบกลไก การตรวจสอบความถูกต้อง จนถึงการนำไปใช้งานจริง ดังนี้

ระยะที่ 1 Testing Phase ใช้ชุดข้อมูลจาก Logfile ระหว่างวันที่ 25 มีนาคม 2566 ถึง 24 เมษายน 2566 จำนวน 91,027 รายการ ระยะนี้ใช้สำหรับการออกแบบและปรับแต่งกฎ (Rule Refinement) การกำหนดค่าน้ำหนัก (Weight Assignment) ของแต่ละรูปแบบการโจมตี และการกำหนดค่าที่จำกัด (Threshold) ที่เหมาะสม ภายใต้ชุดข้อมูลที่ผ่านการทำ Manual Labeling แล้ว

ระยะที่ 2 Validation Phase ใช้ชุดข้อมูลจาก Logfile คนละช่วงเวลา โดยใช้ Logfile ระหว่างวันที่ 25 เมษายน 2566 ถึง 24 พฤษภาคม 2566 จำนวน 88,925 รายการ ระยะนี้ใช้เพื่อประเมินความสามารถในการทั่วไป ของโมเดล โดยไม่มีการปรับแก้กฎเพิ่มเติมเพื่อทดสอบความสามารถของระบบ

ระยะที่ 3 Deployment & Monitoring Phase นำระบบไปสอบและติดตั้งใช้งานจริงใน ตั้งแต่ระหว่างวันที่ 25 พฤษภาคม 2566 ถึง 31 สิงหาคม 2568 รวม 830 วัน โดยยังคงบันทึก Logfile ทั้งหมดเพื่อใช้ประเมินผลเชิงสถิติ ก่อนปิดการบันทึก Logfile หลังจากสิ้นสุดการทดสอบของระยะที่ 3 สำหรับระยะนี้จะใช้วิเคราะห์ประสิทธิภาพของระบบ ภายใต้สภาวะการใช้งานจริงในระยะยาว

ในการวัดประสิทธิภาพของระบบตรวจจับการโจมตี งานวิจัยนี้ได้ใช้ตัวชี้วัดมาตรฐาน 4 ค่า อ้างอิงตามแนวทางของ Lu et al. (2023) โดยการจำแนกผลการตรวจจับออกเป็น 4 รูปแบบ ดังนี้

1. True Positive (TP) ระบบตรวจจับว่าเป็นการโจมตี และข้อมูลนั้นเป็นการโจมตีจริง
2. False Positive (FP) ระบบตรวจจับว่าเป็นการโจมตี แต่ข้อมูลนั้นเป็นคำร้องขอปกติ
3. False Negative (FN) ระบบไม่ตรวจพบการโจมตี ทั้งที่ข้อมูลนั้นเป็นการโจมตีจริง
4. True Negative (TN) ระบบตรวจว่าไม่ใช่การโจมตี และข้อมูลนั้นเป็นคำร้องขอปกติจริง

โดยเปรียบเทียบผลการตรวจจับของระบบกับข้อมูลที่กำหนดสถานะจริง Logfile จากนั้นคำนวณค่าที่เกี่ยวข้อง ดังนี้

1. Accuracy (ความแม่นยำโดยรวม) คือ สัดส่วนของการทำนายที่ถูกต้องทั้งหมด เมื่อเทียบกับข้อมูลทั้งหมด คำนวณจากสูตรที่ (1)

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

หมายถึง ระบบตรวจจับถูกทั้งกรณีที่เป็นการโจมตีจริง และกรณีที่ไม่ใช่การโจมตี ใช้ดู “ภาพรวม” ของระบบทั้งหมด

2. Precision (ความแม่นยำของการแจ้งเตือน) คือ สัดส่วนของสิ่งที่ระบบแจ้งว่าเป็นการโจมตี แล้ว “เป็นการโจมตีจริง” คำนวณจากสูตรที่ (2)

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

ถ้า Precision สูง แสดงว่า ระบบแทบไม่แจ้งเตือนผิดพลาด (False Positive ต่ำ) ใช้ดูว่า “เวลาเตือนแล้ว เชื่อถือได้แค่ไหน”

3. Recall (ความสามารถในการตรวจจับ) คือ สัดส่วนของการโจมตีจริงที่ระบบสามารถตรวจจับได้ คำนวณจากสูตรที่ (3)

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

ถ้า Recall สูง แสดงว่า ระบบไม่พลาดการโจมตี (False Negative ต่ำ) ใช้ดูว่า “ระบบจับภัยคุกคามได้ครบหรือไม่”

4. F1-Score คือค่าชี้วัดที่ใช้ประเมินประสิทธิภาพของโมเดลจำแนกประเภท (classification) โดยเป็นค่าเฉลี่ยแบบ *Harmonic Mean* ระหว่าง Precision และ Recall เพื่อสะท้อนความสมดุลของทั้งสองค่าในตัวเลขนเดียว คำนวณจากสูตรที่ (4) และ (5)

นิยามเชิงคณิตศาสตร์

$$F1 = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (4)$$

หรือเขียนในรูป TP, FP, FN ได้เป็น

$$F1 = \frac{2TP}{2TP + FP + FN} \quad (5)$$

ถ้าผลการทดลองให้ค่า F1-Score เท่ากับ 100% สะท้อนให้เห็นว่าระบบสามารถตรวจจับการโจมตีได้ครบถ้วนโดยไม่เกิดการแจ้งเตือนผิดพลาดในชุดข้อมูลที่ทดสอบ

6. ผลการวิจัย (Results)

6.1 ผลการศึกษารูปแบบการโจมตี SQL Injection จากข้อมูล Logfile จริง

งานวิจัยนี้ได้ดำเนินการเก็บรวบรวมข้อมูลคำร้องขอในสภาพแวดล้อมจริงในระยะเวลาที่ 1 (25 มีนาคม 2566-24 เมษายน 2566) เป็นเวลา 31 วัน โดยยังไม่เปิดใช้งานกลไกการยับยั้ง เพื่อประเมินสถานการณ์ภัยคุกคามที่เกิดขึ้นจริง พบคำร้องขอทั้งหมด 91,027 รายการ โดยตรวจพบความพยายามโจมตีแบบ SQL Injection จำนวน 4,649 ครั้ง คิดเป็นร้อยละ 4.84 ของคำร้องขอทั้งหมด

Table 4. Detailed Classification of Attack Patterns (Phase 1)

Attack Type	Frequency	Percentage
UNION_BASED	6	0.13
BOOLEAN_BASED	1	0.02
TIME_BASED	4,641	99.83
STACKED_QUERY	4	0.09
SCHEMA_ENUM	4,076	87.67
COMMENT_BYPASS	0	0.00
ERROR_BASED	0	0.00
ENCODED	4,251	91.44
SQL_KEYWORDS	4,582	98.56

จากข้อมูลการจำแนกใน Table 4. พบว่าพฤติกรรมโจมตีส่วนใหญ่อยู่ในกลุ่ม TIME_BASED (ร้อยละ 99.83) และมีการใช้เทคนิค ENCODED (ร้อยละ 91.44) ร่วมด้วยอย่างมีนัยสำคัญ ข้อมูลนี้บ่งชี้ว่าผู้โจมตีมุ่งเน้นการใช้วิธีการตรวจสอบโครงสร้างฐานข้อมูลแบบหน่วงเวลาเพื่อหลบเลี่ยงการตรวจจับขั้นพื้นฐาน และมีการพรางคำสั่ง (Encoding) เพื่อสำรวจช่องโหว่และพยายามดึงข้อมูลโครงสร้างระบบ (SCHEMA_ENUM) ซึ่งข้อมูลเหล่านี้ถูกนำไปใช้เป็นเกณฑ์ในการปรับปรุงค่าที่จำกัด (Threshold) ของระบบให้มีความเข้มงวดสอดคล้องกับพฤติกรรมจริงในขั้นตอนถัดไป

6.2 ผลออกแบบและพัฒนาระบบตรวจจับและป้องกัน SQL Injection แบบอัตโนมัติด้วยเทคนิค Rule-based

ข้อมูลการโจมตีจริงที่รวบรวมได้ในระยะที่ 1 ถูกนำมาใช้เป็นฐานในการวิเคราะห์พฤติกรรมและจำแนกรูปแบบคำสั่งอันตราย เพื่อสร้างกฎการตรวจจับ (Rule-based) และกำหนดน้ำหนักคะแนนความเสี่ยง (Scoring) ให้สอดคล้องกับลักษณะการโจมตีที่พบจริงในระบบ culturalmapthailand.info จากนั้นได้ดำเนินการทดสอบระบบในสภาวะป้องกันเชิงรุก

ในระยะเวลาที่ 2 (25 เมษายน 2566-24 พฤษภาคม 2566) เป็นเวลา 30 วัน พบคำร้องขอรวมทั้งสิ้น 88,925 รายการ โดยระบบที่ออกแบบและพัฒนาสามารถตรวจพบความพยายามโจมตีจำนวน 4,312 ครั้ง

Table 5. Detailed Classification of Attack Patterns (Phase 2)

Attack Type	Frequency	Percentage
UNION_BASED	0	0.00
BOOLEAN_BASED	0	0.00
TIME_BASED	4312	100.00
STACKED_QUERY	0	0.00
SCHEMA_ENUM	440	10.20
COMMENT_BYPASS	0	0.00
ERROR_BASED	0	0.00
ENCODED	440	10.20
SQL_KEYWORDS	4312	100.00

เมื่อเปรียบเทียบความแตกต่างของพฤติกรรมการโจมตีระหว่างระยะที่ 1 และระยะที่ 2 สามารถสรุปข้อสังเกตสำคัญได้ดังนี้

1. ความเปลี่ยนแปลงของรูปแบบการโจมตี รูปแบบการโจมตีที่มีความซับซ้อนในเชิงโครงสร้าง เช่น UNION_BASED, BOOLEAN_BASED และ STACKED_QUERY ลดลงจนไม่พบในฐานข้อมูล ร้อยละ 0 ซึ่งสะท้อนถึงความสามารถของระบบในการจำแนกและตอบสนองต่อภัยคุกคามในระดับโครงสร้างได้อย่างเด็ดขาด
2. การลดลงของความพยายามสืบค้นฐานข้อมูล ปริมาณพฤติกรรมในกลุ่ม SCHEMA_ENUM ลดลงอย่างมีนัยสำคัญจาก 4,076 ครั้งในระยะแรก เหลือเพียง 440 ครั้ง บ่งบอกถึงประสิทธิภาพของกฎการตรวจจับที่สามารถหยุดยั้งขั้นตอนการสำรวจช่องโหว่ ของผู้โจมตีได้อย่างรวดเร็ว
3. การวิเคราะห์ค่าเฉลี่ยรายวัน ค่าเฉลี่ยการโจมตีรายวันลดลงจาก 149.97 ครั้ง เหลือ 143.73 ครั้งต่อวัน แม้ปริมาณคำร้องขอรวมจะมีความใกล้เคียงกัน แต่การหายไปของเทคนิคการโจมตีบางประเภทแสดงให้เห็นถึงผลเชิงยับยั้ง ที่เกิดจากประสิทธิภาพของกลไกการตรวจจับที่ทำงานสอดคล้องกับพฤติกรรมที่พบจริงของระบบ

เพื่อให้มั่นใจในเสถียรภาพและความยั่งยืนของระบบ งานวิจัยนี้ได้ติดตามผลการใช้งานในสภาวะป้องกันเชิงรุกจริงอย่างต่อเนื่องในระยะที่ 3 (25 พฤษภาคม 2566-31 สิงหาคม 2568) เป็นเวลา 830 วัน โดยยังทำการจัดเก็บข้อมูลบันทึกเหตุการณ์ต่าง ๆ ลงใน Logfile เพื่อใช้ข้อมูลมาวิเคราะห์ในระยะยาว ในช่วงเวลาดังกล่าวมีคำร้องขอเข้าสู่ระบบรวมทั้งสิ้น 2,041,487 รายการ และระบบสามารถตรวจพบการโจมตีได้จำนวน 25,660 ครั้ง

Table 6. Detailed Classification of Attack Patterns (Phase 3)

Attack Type	Frequency	Percentage
UNION_BASED	5,800	22.60
BOOLEAN_BASED	8,034	31.31
TIME_BASED	7,589	29.58
STACKED_QUERY	1	0.00
SCHEMA_ENUM	1,908	7.44
COMMENT_BYPASS	23,685	92.30
ERROR_BASED	-	0.00
ENCODED	4,230	16.48
SQL_KEYWORDS	12,427	48.43

จาก Table 6. พบว่าพฤติกรรมการโจมตีในระยะยาวมีความซับซ้อนขึ้น โดยมีการใช้เทคนิค COMMENT_BYPASS ร้อยละ 92.30 เพื่อพยายามหลบเลี่ยงกฎการตรวจจับพื้นฐาน แต่ระบบยังคงสามารถจำแนกและยับยั้งได้ครอบคลุมทุกลักษณะที่ปรากฏ สะท้อนให้เห็นว่าการตอบสนองอย่างสม่ำเสมอของระบบส่งผลเชิงพฤติกรรม ต่อผู้โจมตี ทำให้ความพยายามโจมตีในรูปแบบเดิมลดน้อยลงในที่สุด

Table 7. Temporal Comparison of Results across Phase 1 to Phase 3

Phase	Duration (Days)	Log Count	Attacks	Avg. Attacks/Day
1	31	91,027	4,649	149.97
2	30	88,925	4,312	143.73
3	830	2,041,487	25,660	30.92

จาก Table 7. ข้อมูลเชิงสถิติแสดงให้เห็นแนวโน้มการลดลงของความถี่การโจมตีอย่างชัดเจน โดยเฉพาะในระยะที่ 3 ซึ่งค่าเฉลี่ยการโจมตีต่อวัน ลดลงมากกว่าร้อยละ 79.38 เมื่อเทียบกับระยะที่ 1 ผลลัพธ์นี้สนับสนุนสมมติฐานว่า กลไก Rule-based Detection ร่วมกับระบบตอบสนองอัตโนมัติ ไม่ได้ทำหน้าที่เพียงแค่การตรวจจับที่แม่นยำเท่านั้น แต่ยังสามารถสร้าง ผลเชิงยับยั้งในระยะยาว ได้จริงในสภาพแวดล้อมการใช้งานจริง เนื่องจากผู้โจมตีจะยุติความพยายามเมื่อพบว่าระบบมีมาตรการป้องกันที่สามารถสกัดกั้นการโจมตีได้อย่างมีประสิทธิภาพและสม่ำเสมอ

6.3 ผลการประเมินประสิทธิภาพของระบบด้วยตัวชี้วัด TP, FP, FN และ TN

งานวิจัยนี้ได้ประเมินประสิทธิภาพของระบบตลอดทั้ง 3 ระยะ โดยใช้ตัวชี้วัดมาตรฐาน ได้แก่ Accuracy, Precision, Recall และ F1-Score ซึ่งคำนวณจากค่า TP, FP, FN และ TN ดังแสดงใน Table 8.

Table 8. Summary of Performance Metrics across All Phases.

Duration	Total number of logs	TP	FP	FN	TN	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Average attacks per day
Phase 1: 25 Mar - 24 Apr 2023 (31 days)	91,027	4,649	0	0	86,378	100	100	100	100	149.97
Phase 2: 25 Apr - 24 May 2023 (30 days)	88,925	4,312	0	0	84,613	100	100	100	100	143.73
Phase 3: 25 May 2023 - 31 Aug 2025 (830 days)	2,041,487	25,660	0	0	2,015,827	100	100	100	100	30.92

จาก Table 8. ผลการประเมินแสดงให้เห็นว่าระบบสามารถรักษาประสิทธิภาพได้ในระดับสมบูรณ์ตลอดทั้ง 3 ระยะ โดยมีรายละเอียดดังนี้

ด้านความถูกต้องของการตรวจจับ ค่า True Positive ในแต่ละระยะสะท้อนจำนวนการโจมตีที่ระบบตรวจจับได้จริง ได้แก่ 4,649 ครั้งในระยะที่ 1, 4,312 ครั้งในระยะที่ 2 และ 25,660 ครั้งในระยะที่ 3 โดยในทุกระยะพบว่าค่า False Positive และ False Negative มีค่าเท่ากับ 0 หมายความว่าระบบไม่เคยแจ้งเตือนผิดพลาดต่อคำร้องขอปกติ และไม่มีการโจมตีใดหลุดรอดจากการตรวจจับเลย

ด้านค่าตัวชี้วัด ค่า Accuracy, Precision, Recall และ F1-Score มีค่าเท่ากับร้อยละ 100 ในทุกระยะ ซึ่งสะท้อนให้เห็นว่ากฎการตรวจจับที่ออกแบบขึ้นมีความแม่นยำ แม้จะผ่านการทดสอบในสภาวะแวดล้อมและระยะเวลาที่แตกต่างกันอย่างมีนัยสำคัญ โดยเฉพาะในระยะที่ 3 ซึ่งครอบคลุมระยะเวลากว่า 830 วัน และมีปริมาณคำร้องขอรวมสูงถึง 2,041,487 รายการ

นอกจากนี้ผลการทดลองยังชี้ให้เห็นว่าระบบที่พัฒนาขึ้นส่งผลกระทบต่อการบริหารจัดการทรัพยากรสารสนเทศและความยั่งยืนของระบบในระยะยาว โดยเฉพาะในด้านการลดภาระการจัดเก็บข้อมูล ในระหว่างการทดลองระยะที่ 1 ถึง 3 ระบบทำหน้าที่บันทึกทุกเหตุการณ์ เพื่อประเมินประสิทธิภาพและยืนยันความแม่นยำ เมื่อหลังจากสิ้นสุดการทดลองระยะที่ 3 ซึ่งระบบได้พิสูจน์ค่าความแม่นยำที่ร้อยละ 100 อย่างสม่ำเสมอ งานวิจัยนี้จึงได้กำหนดแนวทางการจัดการข้อมูลใหม่ โดยยกเลิกการบันทึก Log สำหรับคำร้องขอที่ถูกจำแนกเป็นการโจมตี หรือมีค่าคะแนนความเสี่ยง (Risk Score) ตั้งแต่ 5 ขึ้นไป การดำเนินการนี้ส่งผลให้ระบบไม่ต้องแบกรับภาระการจัดเก็บข้อมูลขยะที่เกิดจากการโจมตีซึ่งอาจมีปริมาณมากกว่า 10,000 รายการต่อปี (อ้างอิงจากฐานข้อมูลคำร้องขอในระยะที่ 3) ช่วยป้องกันปัญหฐานข้อมูลขยายตัวเกินความจำเป็นในอนาคต

7. สรุปผลการวิจัย (Conclusion)

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษารูปแบบการโจมตี SQL Injection จาก Logfile จริง ออกแบบและพัฒนาระบบตรวจจับแบบ Rule-based และประเมินประสิทธิภาพด้วยตัวชี้วัด TP, FP, FN และ TN ผลการวิจัยยืนยันว่าบรรทัดฐานของระบบทั้ง 3 ข้อได้อย่างครบถ้วน

ระบบที่พัฒนาขึ้นใช้แนวทาง Hybrid Rule-based Detection ที่ผสมผสาน Whitelist Filtering, Pattern Matching และ Scoring Mechanism เข้าด้วยกัน โดยออกแบบกฎตรวจจับจากข้อมูล Logfile จริงของระบบ culturalmapthailand.info ซึ่งพัฒนาด้วย PHP และ MySQL ผลการทดสอบทั้ง 3 ระยะในสภาวะแวดล้อมจริงพบว่าระบบสามารถรักษาค่า Accuracy, Precision, Recall และ F1-Score ที่ร้อยละ 100 ได้อย่างต่อเนื่องตลอด 830 วัน โดยไม่เกิด False Positive แม้แต่ครั้งเดียว และสามารถตรวจจับการโจมตีในสภาวะการใช้งานจริงได้รวม 25,660 ครั้ง

ผลลัพธ์ดังกล่าวแสดงให้เห็นว่าแนวทาง Rule-based ที่ออกแบบจากข้อมูลจริงและเสริมด้วย Scoring Mechanism เป็นทางเลือกที่มีประสิทธิภาพสูงสำหรับระบบที่มีทรัพยากรจำกัด และสามารถนำไปประยุกต์ใช้กับเว็บแอปพลิเคชัน PHP/MySQL ทั่วไปได้โดยไม่ต้องพึ่งพา Machine Learning

8. ข้อจำกัดของการวิจัย (Research Limitations)

ผลการวิจัยที่ให้ค่า Accuracy, Precision และ Recall ร้อยละ 100 ในทุกระยะการทดลองนั้น มีข้อจำกัดที่ควรพิจารณา คือ รูปแบบการโจมตี (Patterns) และค่าน้ำหนัก (Weights) ถูกออกแบบและปรับแต่งโดยอ้างอิงจากข้อมูล Logfile ที่เกิดขึ้นจริงในระบบเฉพาะของระบบ culturalmapthailand.info ซึ่งข้อมูลปกติ มีรูปแบบที่ค่อนข้างคงที่ จึงทำให้กลไก Whitelist Filtering และ Rule-based ตรวจจับได้แม่นยำสูง อย่างไรก็ตาม ผลลัพธ์นี้อาจเปลี่ยนแปลงได้หากนำไปใช้กับระบบที่มีความซับซ้อนของพารามิเตอร์มากกว่านี้ หรือเผชิญกับการโจมตีรูปแบบใหม่ที่ยังไม่เคยปรากฏในชุดข้อมูล

9. อภิปรายผลการวิจัย (Discussion)

จากการพัฒนาระบบตรวจจับและป้องกันการโจมตีแบบ SQL Injection โดยใช้กลไก Whitelist Filtering การให้ร่วมกับน้ำหนักคะแนน Scoring Mechanism จากข้อมูลใน Logfile ผลการทดลองแสดงให้เห็นถึงประสิทธิภาพที่น่าพึงพอใจ ซึ่งสามารถนำมาอภิปรายผลในประเด็นสำคัญได้ดังนี้

1. ผลลัพธ์ที่ได้ร้อยละ 100 ในทุกตัวชี้วัดไม่ได้เกิดจากปัจจัยใดปัจจัยเดียว แต่เป็นผลรวมของการออกแบบระบบที่ทำงานประสานกัน 3 ระดับ

ระดับแรกคือ คุณภาพของข้อมูลต้นทาง กฎตรวจจับและค่าน้ำหนักทั้งหมดถูกออกแบบจาก Logfile จริงของระบบ culturalmapthailand.info ซึ่งต่างจากงานของ Arnep & Kusirini (2022) และ Alghawazi et al. (2022) ที่ใช้

ชุดข้อมูลจำลอง การที่กฎหมายจากข้อมูลจริงของระบบนั้นเองทำให้ครอบคลุม Edge Case ที่จำเพาะเจาะจงกับพฤติกรรม Traffic ของระบบ ซึ่ง Benchmark Dataset ไม่สามารถจำลองได้

ระดับที่สองคือ สถาปัตยกรรมแบบลำดับชั้น การเริ่มต้นด้วย Whitelist Filtering ทำให้คำร้องขอปกติที่รู้จักผ่านไปได้ทันทีโดยไม่เข้าสู่กระบวนการตรวจจับ สอดคล้องกับแนวคิดของ Hofesh (2022) ที่ระบุว่า Allowlisting เป็นวิธีที่ดีที่สุดในการจัด False Positive ส่วน Pattern Matching ที่ทำงานในขั้นถัดมาจึงรับมือเฉพาะคำร้องขอที่ยังไม่ถูกระบุว่าปลอดภัย ลดโอกาสที่จะเกิดการตรวจจับผิดพลาดลงได้อย่างมีนัยสำคัญ

ระดับที่สามคือ การ Calibrate ค่า Threshold จากข้อมูลจริง เกณฑ์ 5 คะแนนถูกกำหนดจากการวิเคราะห์ Distribution ของคะแนนในระยะที่ 1 ทำให้ระบบไม่ตัดสินใจจาก Keyword เดียว แต่พิจารณาจากบริบทโดยรวมของ Payload ซึ่งเป็นเหตุผลที่ทำให้ระบบสามารถรับมือกับเทคนิค Evasion ที่ผู้โจมตีใช้หลบเลี่ยงการตรวจจับแบบ Signature-based ได้ตามที่ Kaur et al. (2023) และ Mutedi & Tjahjono (2022) ชี้ให้เห็นว่าเป็นจุดอ่อนสำคัญของระบบตรวจจับทั่วไป

2. ความเหมาะสมของ Scoring Mechanism ในระบบที่มีทรัพยากรจำกัด แม้ว่าการวิจัยส่วนใหญ่จะมุ่งเน้นการใช้ Machine Learning เช่น งานของ Lu et al. (2023) แต่ผลการทดลองนี้พิสูจน์ว่า Scoring Mechanism เป็นทางเลือกที่เหมาะสมกว่าสำหรับระบบที่ต้องการความเร็ว (Real-time Response) เนื่องจากใช้ทรัพยากรต่ำแต่ยังสามารถจำแนกภัยคุกคามที่มีการหลบเลี่ยง ได้อย่างแม่นยำผ่านการคำนวณน้ำหนักคะแนนความเสี่ยงสะสม

3. นัยสำคัญของการลดลงของจำนวนการโจมตี การโจมตีที่ลดลงอย่างต่อเนื่องจนเหลือเพียง 30.92 ครั้งต่อวันในระยะที่ 3 สะท้อนถึงประสิทธิภาพของกลไกการยับยั้ง เมื่อระบบสามารถสกัดกั้นคำร้องขอที่เป็นอันตรายได้ทันที ทำให้ผู้โจมตีไม่สามารถดำเนินขั้นตอนการโจมตีต่อไปได้ตามทฤษฎีของ Fu et al. (2023) นอกจากนี้ยังช่วยให้การบริหารจัดการข้อมูลบันทึกเหตุการณ์ มีความยั่งยืนมากขึ้น โดยลดปริมาณการจัดเก็บข้อมูลลงใน Logfile ที่ไม่จำเป็นลงได้ร้อยละ 4 - 5 ต่อเดือน

4. ข้อจำกัดและแนวทางการพัฒนาในอนาคต แม้ผลลัพธ์จะเป็นที่น่าพอใจ แต่การพึ่งพากฎแบบคงที่ (Static Rules) ยังมีข้อจำกัดในการรับมือกับภัยคุกคามรูปแบบใหม่ ๆ สอดคล้องกับข้อเสนอแนะของ Alghawazi et al., (2022) ที่ชี้ให้เห็นความสำคัญของการเรียนรู้ของเครื่อง ดังนั้น การบูรณาการระหว่างกฎเชิงตรรกะร่วมกับเทคนิคการเรียนรู้เชิงลึก (Deep Learning) จึงเป็นแนวทางสำคัญในการยกระดับการป้องกันให้ครอบคลุมการโจมตีที่ซับซ้อนยิ่งขึ้นในอนาคต

10. ข้อเสนอแนะงานวิจัย (Recommendation)

จากการผ่านการใช้งานจริงกว่า 2 ปี มีข้อเสนอแนะที่เป็นประโยชน์ต่อการนำระบบไปใช้งานต่อ และแนวทางสำหรับงานวิจัยในอนาคต ดังนี้

1. ข้อเสนอแนะในการนำไปใช้งาน การปรับค่าเกณฑ์คะแนนขีดจำกัด (Threshold) แม้ผลการศึกษาระบุว่าเกณฑ์การตัดสินใจที่ 5 คะแนน ให้ความแม่นยำสูงสุดสำหรับระบบ culturalmapthailand.info แต่หากจะนำระบบนี้ไปใช้กับแอปพลิเคชันอื่น ผู้ดูแลระบบจำเป็นต้องทำการทดสอบและปรับค่าคะแนน ให้สอดคล้องกับพฤติกรรมข้อมูลของระบบนั้นๆ ก่อน เนื่องจากรูปแบบคำสั่ง SQL ปกติของบางระบบอาจมีความซับซ้อนจนไปกระตุ้นกฎการตรวจจับ ซึ่งอาจนำไปสู่การปฏิเสธคำร้องขอที่ถูกต้องได้

2. การป้องกันเชิงลึก ระบบที่พัฒนาขึ้นนี้ทำหน้าที่เป็นปราการด่านหน้าที่ดี อย่างไรก็ตาม เพื่อให้เกิดความมั่นคงปลอดภัยสูงสุด ควรใช้ระบบนี้ควบคู่ไปกับการเขียนโปรแกรมที่ปลอดภัย เช่น การใช้ Prepared Statements หรือ Parameterized Queries เพื่อสร้างเกราะป้องกันหลายชั้นที่สามารถรับมือกับภัยคุกคามได้ทุกระดับ

3. การต่อยอดด้านการวิเคราะห์ความยั่งยืน จากการที่ระบบเริ่มยกเลิกการบันทึกข้อมูลลงใน Logfile ของการโจมตีที่มี Score สูง ($Score \geq 5$) เพื่อลดภาระฐานข้อมูล ในอนาคตอาจมีการพัฒนาโมดูลสรุปผลทางสถิติแยกต่างหาก เพื่อให้ง่ายต่อการเห็นแนวโน้มของภัยคุกคามได้โดยไม่ต้องเก็บข้อมูลชุดคำสั่งอันตราย (Payload) ขนาดใหญ่ไว้ในฐานข้อมูลหลัก ช่วยให้การบริหารจัดการข้อมูลมีความยั่งยืนและปลอดภัยไปพร้อมกัน

4. การทดสอบข้ามระบบ (Cross-system Validation) งานวิจัยนี้พัฒนาและทดสอบระบบบนแอปพลิเคชัน culturalmapthailand.info ซึ่งมีลักษณะการใช้งานและรูปแบบ HTTP request ที่จำเพาะเจาะจง เพื่อยืนยัน

ความสามารถในการนำไปประยุกต์ใช้ในวงกว้าง งานวิจัยในอนาคตควรทดสอบระบบบนแอปพลิเคชัน PHP/MySQL ประเภทอื่น เช่น เว็บที่เป็นระบบ e-Commerce ระบบจัดการเนื้อหา (CMS) หรือระบบสารสนเทศองค์กรอื่น ๆ เพื่อประเมินว่าค่า Threshold ที่ 5 คะแนน และชุดกฎ Regex ที่ออกแบบไว้ยังคงให้ค่า False Positive เป็นศูนย์ในบริบทที่แตกต่างหรือไม่ รวมถึงศึกษาว่าจำเป็นต้องปรับ Whitelist ให้เหมาะสมกับแต่ละระบบมากน้อยเพียงใด

5. การเปรียบเทียบกับมาตรฐานสากล แม้ระบบจะแสดงประสิทธิภาพสูงสุดในสภาพแวดล้อมจริง แต่เพื่อให้สามารถเปรียบเทียบกับงานวิจัยอื่นในระดับสากลได้อย่างเป็นกลาง งานวิจัยในอนาคตควรทดสอบระบบกับชุดข้อมูลมาตรฐาน เช่น CSIC 2010 HTTP Dataset หรือ Kaggle SQL Injection Dataset ซึ่งเป็นชุดข้อมูลที่นักวิจัยด้าน Web Security ใช้อ้างอิงร่วมกันอย่างแพร่หลาย การทดสอบดังกล่าวจะช่วยให้สามารถระบุได้ว่า Scoring Mechanism ที่ออกแบบจาก Logfile จริงมีประสิทธิภาพเทียบเท่าหรือเหนือกว่าวิธีการที่ได้รับการทดสอบบน Benchmark แล้วอย่างไร และเสริมความน่าเชื่อถือของงานวิจัยให้มากขึ้น

11. การเปิดเผยการใช้ปัญญาประดิษฐ์เชิงสร้างสรรค์ (Declaration of Generative AI)

งานวิจัยนี้ได้นำเครื่องมือปัญญาประดิษฐ์มาช่วยเป็นต้นแบบในการเตรียมต้นฉบับเพื่อให้เนื้อหา มีความสมบูรณ์ยิ่งขึ้น โดยใช้ ChatGPT, Gemini และ Claude.ai เพื่อช่วยตรวจสอบการสะกดคำผิด และปรับปรุงการเรียบเรียงประโยคให้เป็นภาษาวิชาการ รวมถึงใช้ในการตรวจสอบความสอดคล้องของเนื้อหา ระหว่างบท เพื่อให้การอธิบายขั้นตอนการตรวจจับ SQL Injection และการวิเคราะห์ Logfile มีความต่อเนื่องและเข้าใจง่าย นอกจากนี้ใช้ AI เป็นเครื่องมือช่วยในการแปลบทความเป็นภาษาอังกฤษและขัดเกลาทางไวยากรณ์ให้ถูกต้องตามหลักสากลและขัดเกลาทางไวยากรณ์ให้ถูกต้องตามหลักสากล รวมถึงใช้ Gemini ในการช่วยจัดรูปแบบกราฟิกและแผนภูมิแสดงระยะเวลา (Timeline)

การยืนยันความถูกต้อง ขอรับรองว่าเนื้อหาหลัก ข้อมูลสถิติจากระบบ Cultural Map Thailand ผลการทดลองทั้งหมดแต่ละระยะ และการวิเคราะห์ผลทั้งหมด เป็นผลมาจากการดำเนินงานของผู้วิจัยโดยตรง AI เป็นเพียงเครื่องมือช่วยในการจัดทำเอกสารและตรวจสอบภาษาเท่านั้น ผู้วิจัยได้ตรวจสอบและแก้ไขเนื้อหาทุกส่วนด้วยตนเองเพื่อให้เป็นเนื้อหาต้นฉบับและมีความถูกต้องทางวิชาการอย่างครบถ้วน โดยผู้วิจัยขอรับผิดชอบต่อเนื้อหาทั้งหมดในงานวิจัยนี้แต่เพียงผู้เดียว

12. การแสดงบทบาทผู้เขียน (CRediT Author Statement)

Chanakit Mitsongkore: Conceptualization, Methodology, Formal Analysis, Investigation, Software, Validation, Writing-Original Draft, Writing-Review & Editing.

The author has read and approved the final manuscript.

13. เอกสารอ้างอิง (References)

- Alghawazi, M., Alghazzawi, D., & Alarifi, S. (2022). Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review. *Journal of Cybersecurity and Privacy*, 2(4), 764–777. <https://doi.org/10.3390/jcp2040039>
- Arnab, A., & Kusrini. (2024). Enhancing SQL Injection Attack Detection Using Naïve Bayes and SMOTE Method on Imbalanced Datasets. *Journal of Artificial Intelligence and Engineering Applications*, 4(1), 74–81. <https://doi.org/10.59934/jaiea.v4i1.559>
- Das, D., Sharma, U., & Bhattacharyya, D. K. (2019). Defeating SQL Injection Attack in Authentication Security: An Experimental Study. *International Journal of Information Security*, 18(1), 1–22. <https://doi.org/10.1007/s10207-017-0393-x>
- Fu, H., Guo, C., Jiang, C., Ping, Y., & Lv, X. (2023). SDSIOT: An SQL Injection Attack Detection and Stage Identification Method Based on Outbound Traffic. *Electronics*, 12(11), 2472. <https://doi.org/10.3390/electronics12112472>

- Hofesh, B. (2022). *SQL Injection Attack: How It Works, Examples and Prevention*. Bright Security.
<https://brightsec.com/blog/sql-injection-attack>
- Kaur, J., Garg, U., & Bathla, G. (2023). Detection of Cross-Site Scripting (XSS) Attacks Using Machine Learning Techniques: A Review. *Artificial Intelligence Review*, 56(11), 12725-12769. <https://doi.org/10.1007/s10462-023-10433-3>
- Lu, D., Fei, J., & Liu, L. (2023). A Semantic Learning-Based SQL Injection Attack Detection Technology. *Electronics*, 12(6), 1344. <https://doi.org/10.3390/electronics12061344>
- Muhammad, T., & Ghafory, H. (2022). SQL Injection Attack Detection Using Machine Learning Algorithm. *Mesopotamian Journal of CyberSecurity*, 2022, 5-17. <https://doi.org/10.58496/MJCS/2022/002>
- Mutedi, A., & Tjahjono, B. (2022). Systematic Literature Review: Preventing SQL Injection Attacks Using Tools OWASP CSR Web Application Firewall. *Jurnal Informatika Universitas Pamulang*, 7(1), 151-156.
- OWASP. (2021). *Introduction: Welcome to the OWASP Top 10-2021*. https://owasp.org/Top10/2021/A00_2021_Introduction
- Pan, Y., Sun, F., Teng, Z., White, J., Schmidt, D. C., Staples, J., & Krause, L. (2019). Detecting Web Attacks with End-to-End Deep Learning. *Journal of Internet Services and Applications*, 10(1), 16.
<https://doi.org/10.1186/s13174-019-0115-x>
- Securities and Exchange Commission. (2023). *Cyber Attack Trends 2023, Based on Data from the National Cyber Security Agency*. <https://www.sec.or.th/th/pages/cyberresilience-statistics-2566.aspx> (In Thai)