

การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข

นางสาวณัฐธิดา ลูกเสือ

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต
สาขาวิชาการบริหารเครือข่ายดิจิทัลและความมั่นคงปลอดภัยสารสนเทศ
บัณฑิตวิทยาลัย มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2568
ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข

นางสาวณัฐธิดา ลูกเสือ

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตร
วิทยาศาสตรมหาบัณฑิต
สาขาวิชาการบริหารเครือข่ายดิจิทัลและความมั่นคงปลอดภัยสารสนเทศ
บัณฑิตวิทยาลัย มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2568
ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ



ใบรับรองการค้นคว้าอิสระ

บัณฑิตวิทยาลัย มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เรื่อง การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข

โดย นางสาวณัฐธิดา ลูกเสือ

ได้รับอนุมัติให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาการบริหารเครือข่ายดิจิทัลและความมั่นคงปลอดภัยสารสนเทศ

คณบดีคณะเทคโนโลยีสารสนเทศและ

.....

นวัตกรรมดิจิทัล

(ผู้ช่วยศาสตราจารย์ ดร.สุนันฑา สดสี)

คณะกรรมการสอบการค้นคว้าอิสระ

.....

ประธาน

(รองศาสตราจารย์ ดร.สุมิตรา นวลมีศรี)

.....

อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร.สุนันฑา สดสี)

.....

กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.พงศ์ศรัณย์ บุญโญปกรณ์)

ชื่อ : นางสาวณัฐธิดา ลูกเสือ
ชื่อการค้นคว้าอิสระ : การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข
สาขาวิชา : การบริหารเครือข่ายดิจิทัลและความมั่นคงปลอดภัยสารสนเทศ
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
อาจารย์ที่ปรึกษาการค้นคว้าอิสระหลัก : ผู้ช่วยศาสตราจารย์ ดร.สุนันฑา สดสี
ปีการศึกษา : 2568

บทคัดย่อ

ในยุคดิจิทัลปัจจุบัน เว็บแอปพลิเคชันได้กลายเป็นหัวใจสำคัญในการขับเคลื่อนธุรกิจ การให้บริการ และการสื่อสารในทุกภาคส่วน อย่างไรก็ตาม การเติบโตนี้มาพร้อมกับความเสี่ยงด้านความปลอดภัยทางไซเบอร์ที่เพิ่มขึ้นอย่างทวีคูณ เว็บแอปพลิเคชันมักตกเป็นเป้าหมายหลักของการโจมตี เนื่องจากเป็นช่องทางตรงที่นำไปสู่ข้อมูลสำคัญขององค์กรและข้อมูลส่วนบุคคลของผู้ใช้งาน โดยจากสถิติในช่วงครึ่งแรกของปี 2024 พบว่าประเทศไทยมีความพยายามในการโจมตีทางเว็บเฉลี่ยสูงถึง 5,811 ครั้งต่อวัน และมีแนวโน้มการโจมตีแบบเจาะจงที่เว็บไซต์ (Hacked Website) เพิ่มสูงขึ้นอย่างต่อเนื่อง การโจมตีที่ประสบความสำเร็จอาจนำมาซึ่งความเสียหายร้ายแรง ทั้งในด้านการเงิน ชื่อเสียง และความน่าเชื่อถือขององค์กร โดยในปี 2025 ค่าใช้จ่ายเฉลี่ยทั่วโลกจากเหตุการณ์ข้อมูลรั่วไหลสูงถึง 4.88 ล้านดอลลาร์สหรัฐ ซึ่งเพิ่มขึ้น 10% จากปีก่อนหน้า

เพื่อรับมือกับความท้าทายดังกล่าว เครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชัน ได้กลายเป็นเครื่องมือพื้นฐานที่องค์กรและนักพัฒนาใช้ในการค้นหาจุดอ่อนและความเสี่ยงเชิงรุกก่อนที่ผู้ไม่หวังดีจะค้นพบ อย่างไรก็ตาม ในปัจจุบันมีเครื่องมือให้เลือกใช้งานหลากหลาย ทั้งแบบ Open-source ที่ไม่มีค่าใช้จ่าย และแบบ Commercial ที่มีฟังก์ชันการทำงานขั้นสูง ซึ่งแต่ละเครื่องมือก็มีจุดแข็ง จุดอ่อน และประสิทธิภาพในการตรวจจับช่องโหว่ที่แตกต่างกันไป และในการเลือกใช้เครื่องมือที่ไม่เหมาะสมอาจนำไปสู่ปัญหาสำคัญสองประการที่อาจเกิดขึ้นได้ คือ ผลบวกลวง (False Positives) การที่เครื่องมือแจ้งเตือนว่ามีช่องโหว่ทั้งที่ไม่มีอยู่จริง ทำให้ทีมพัฒนาต้องเสียเวลาและทรัพยากรไปกับการตรวจสอบและแก้ไขในสิ่งที่ไม่ใช่ปัญหา และผลลบลวง (False Negatives) การที่เครื่องมือ "ไม่สามารถ" ตรวจจับช่องโหว่ที่มีอยู่จริง ซึ่งเป็นสถานการณ์ที่อันตรายที่สุด เพราะทำให้องค์กร

เข้าใจผิดว่าระบบของตนปลอดภัย และปล่อยให้ช่องโหว่ร้ายแรงนั้นคงอยู่ต่อไป ยิ่งไปกว่านั้น วงจรการจัดการความปลอดภัยไม่ได้สิ้นสุดลงเพียงแค่การค้นพบช่องโหว่ แต่ยังต้องครอบคลุมไปถึงกระบวนการยืนยันผลการแก้ไข (Remediation Validation) ซึ่งเป็นขั้นตอนที่สำคัญอย่างยิ่งแต่กลับถูกมองข้ามไปบ่อยครั้ง การแก้ไขช่องโหว่โดยปราศจากการตรวจสอบและยืนยันผลที่ชัดเจนอาจทำให้ช่องโหว่เดิมยังคงอยู่ หรืออาจก่อให้เกิดปัญหาใหม่โดยไม่คาดคิด

ดังนั้น การศึกษาและเปรียบเทียบประสิทธิภาพของเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชัน ควบคู่ไปกับการกำหนดกระบวนการยืนยันผลการแก้ไขที่ชัดเจนและเชื่อถือได้ จึงมีความสำคัญอย่างยิ่ง การค้นคว้าอิสระนี้จึงมีวัตถุประสงค์เพื่อวิเคราะห์และเปรียบเทียบประสิทธิภาพของเครื่องมือสแกนโดยจะยกตัวอย่างเครื่องมือแสกนในกลุ่ม Open-source พร้อมทั้งนำเสนอแนวทางการแก้ไขและกระบวนการตรวจสอบที่เป็นระบบ ผลลัพธ์จากการศึกษานี้จะเป็นประโยชน์ต่อองค์กรและนักพัฒนาในการตัดสินใจเลือกใช้เครื่องมือที่เหมาะสมกับบริบทของตนเอง และสามารถสร้างกระบวนการจัดการช่องโหว่ที่ครบวงจรและมีประสิทธิภาพ เพื่อยกระดับความปลอดภัยของเว็บแอปพลิเคชันให้สูงขึ้นอย่างยั่งยืน

(มีจำนวนทั้งสิ้น 89 หน้า)

คำสำคัญ : ความมั่นคงปลอดภัยเว็บแอปพลิเคชัน, การสแกนช่องโหว่, การจัดการช่องโหว่

_____ อาจารย์ที่ปรึกษาการค้นคว้าอิสระหลัก

Name : Miss Natthida Looksua

Independent Study Title : Performance comparison of web application vulnerability scanning tools and patch validation processes

Major Field : Digital Network and Information Security Management
King Mongkut's University of Technology North Bangkok

Independent Study Advisor : Assistant Professor Dr. Sunantha Sodsee

Academic Year : 2025

ABSTRACT

In the current digital era, web applications have become the fundamental core driving business operations, service delivery, and communication across all sectors. However, this rapid growth is accompanied by exponentially increasing cybersecurity risks. Web applications are frequently primary targets for cyberattacks as they serve as direct gateways to sensitive organizational assets and users' personal data. Statistics from the first half of 2024 reveal that Thailand faced an average of 5,811 web attack attempts per day, with a continuous upward trend in targeted website defacements and breaches. A successful attack can result in catastrophic consequences, including financial loss, reputational damage, and a decline in organizational credibility. By 2025, the average global cost of a data breach reached 4.88 million USD, a 10% increase from the previous year.

To counter these challenges, Web Application Vulnerability Scanners have become essential tools for organizations and developers to proactively identify weaknesses and risks before they are exploited by malicious actors. Currently, a wide range of tools is available, from free Open-source options to advanced Commercial solutions, each possessing distinct strengths, weaknesses, and detection efficiencies. Selecting an inappropriate tool can lead to two critical issues: False Positives,

where a tool alerts to non-existent vulnerabilities, causing developers to waste time and resources on non-issues; and False Negatives, where a tool fails to detect actual vulnerabilities. The latter represents the most dangerous scenario, as it fosters a false sense of security while leaving critical risks unaddressed. Furthermore, the vulnerability management lifecycle does not conclude with mere discovery; it must encompass Remediation Validation, a critical yet often overlooked phase. Addressing vulnerabilities without a clear and reliable verification process may leave the original flaws intact or inadvertently introduce new complications. Therefore, studying and comparing the efficiency of vulnerability scanners, alongside establishing a systematic verification process, is of paramount importance.

This Independent Study aims to analyze and compare the performance of various scanners, specifically focusing on prominent Open-source tools. It also presents systematic remediation guidelines and verification processes. The findings of this research will serve as a valuable resource for organizations and developers in selecting the tools best suited to their contexts, enabling them to establish a comprehensive and effective vulnerability management process to enhance web application security sustainably.

(Total 89 Pages)

Keywords: Web Application Security, Vulnerability Scanning, Vulnerability Management

_____ Advisor

กิตติกรรมประกาศ

การค้นคว้าอิสระฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาอย่างยิ่งจาก ผู้ช่วยศาสตราจารย์ ดร. สุนันฑา สดสี อาจารย์ที่ปรึกษาการค้นคว้าอิสระ ที่ได้ให้ความเมตตา ให้คำปรึกษา แนะนำแนวทางที่ถูกต้อง และช่วยตรวจทานแก้ไขข้อบกพร่องต่าง ๆ ด้วยความเอาใจใส่เสมอมา ตลอดจนการชี้แนะในส่วนของการประเมินผลงานวิจัยเพื่อให้มีความสมบูรณ์เชิงวิชาการ ผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอกราบขอบพระคุณคณาจารย์ในสาขาวิชาการบริหารเครือข่ายดิจิทัลและความมั่นคงปลอดภัยสารสนเทศทุกท่านที่ได้ประสิทธิประสาทวิชาความรู้ ตลอดจนคำแนะนำอันเป็นประโยชน์ที่เป็นรากฐานสำคัญในการทำวิจัยชิ้นนี้

ขอขอบพระคุณผู้เชี่ยวชาญและผู้ใช้งานทั้ง 4 ท่าน ที่ได้สละเวลาอันมีค่าในการตรวจสอบคุณภาพและประเมินประสิทธิภาพของ คู่มือแนวทางการแก้ไขช่องโหว่ ตลอดจนให้ข้อเสนอแนะในเชิงเทคนิคที่ทรงคุณค่า ซึ่งช่วยให้งานวิจัยชิ้นนี้มีความสมบูรณ์และสามารถนำไปประยุกต์ใช้งานได้จริงในระดับปฏิบัติการ

สุดท้ายนี้ ผู้วิจัยขอขอบพระคุณครอบครัว เพื่อนร่วมงาน และเพื่อนร่วมการศึกษาทุกท่าน ที่เป็นกำลังใจและให้ความช่วยเหลือสนับสนุนในทุกด้านเสมอมา

ณัฐธิดา ลูกเสือ

สารบัญ

หน้า

กิตติกรรมประกาศ ฉ

สารบัญ ช

สารบัญตาราง ฌ

สารบัญรูปภาพ ญ

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญของปัญหา 1

1.2 วัตถุประสงค์ของการวิจัย 2

1.3 ขอบเขตของการวิจัย 2

1.4 วิธีการดำเนินการวิจัย 3

1.5 ประโยชน์ที่คาดว่าจะได้รับ 4

บทที่ 2 แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง 6

2.1 แนวคิดและทฤษฎีเกี่ยวกับความมั่นคงปลอดภัยของแอพพลิเคชั่น 6

2.2 แนวคิดเกี่ยวกับเทคโนโลยีการตรวจหาช่องโหว่เว็บแอปพลิเคชัน 7

2.3 วงจรการบริหารจัดการช่องโหว่ (Vulnerability Management Lifecycle) 8

2.4 งานวิจัยที่เกี่ยวข้อง 10

บทที่ 3 วิธีการดำเนินการวิจัย 17

3.1 กรอบแนวคิดการวิจัย 17

3.2 ขั้นตอนการทำวิจัย 18

สารบัญ (ต่อ)

หน้า

บทที่ 4 ผลการวิจัย 33

4.1 ผลการเปรียบเทียบและคัดเลือกเครื่องมือสแกนช่องโหว่ 34

4.2 ผลการตรวจพบช่องโหว่สถานะฐาน (Initial Baseline Results) 42

4.3 ผลการวิเคราะห์สาเหตุและแนวทางการแก้ไข (Remediation Analysis) 46

4.4 ผลการตรวจสอบยืนยันการแก้ไข (Validation and Re-scan Results) 46

4.5 สรุปผลสัมฤทธิ์ของการดำเนินงานวิจัย 52

บทที่ 5 สรุป อภิปรายผล และข้อเสนอแนะ 55

5.1 สรุปผลการวิจัย 55

5.2 การอภิปรายผล 56

5.3 ข้อจำกัดของการวิจัย 56

5.4 ข้อเสนอแนะ 57

บรรณานุกรม 59

ภาคผนวก ก คู่มือแนวทางการแก้ไขช่องโหว่ 61

ภาคผนวก ข เอกสารหลักฐานที่เกี่ยวข้อง 87

ประวัติผู้เขียน 89

สารบัญตาราง

ตารางที่		หน้า
2-1	งานวิจัยที่เกี่ยวข้อง	13
3-1	เป้าหมายหลักเพื่อพิสูจน์ประสิทธิภาพของเครื่องมือสแกนช่องโหว่	20
3-2	ขอบเขตองค์ประกอบของคู่มือการแก้ไขช่องโหว่	27
3-3	แบบประเมินคุณภาพคู่มือแนวทางการแก้ไขช่องโหว่	31
4-1	ผลการเปรียบเทียบเครื่องมือแสกน	33
4-2	การเปรียบเทียบเชิงปริมาณและระยะเวลา	39
4-3	การวิเคราะห์ความครอบคลุมตามมาตรฐาน OWASP Top 10	39
4-4	แสดงสถิติจำนวนช่องโหว่สถานะฐานจำแนกตามระดับความรุนแรง	43
4-5	รายละเอียดช่องโหว่ที่ตรวจพบจากรีพอร์ตของ OWASP ZAP	43
4-6	รายละเอียดช่องโหว่ที่ตรวจพบเทียบกับ Owasp Top 10 2021	44
4-7	ตารางเปรียบเทียบจำนวนช่องโหว่ก่อนและหลังการปรับปรุงระบบ	47
4-8	ผลคะแนนเฉลี่ย	51

สารบัญรูปภาพ

ภาพที่		หน้า
2-1	CIA Triad	6
2-2	2021 OWASP Top 10 list	7
2-3	Vulnerability Assessment	8
2-4	Vulnerability Management Lifecycle	9
3-1	ขั้นตอนและกระบวนการดำเนินงาน (Process Diagram)	19
3-2	OWASP ZAP version 2.17.0	24
3-3	Arachni version 1.6.1.3 -Web UI version 0.6.1.1	25
3-4	Burp Suite Version Community Edition 2025.11.6	25
4-1	รูปที่ระบุถึงความล้าสมัยของ Arachni	35
4-2	Owasp ZAP รองรับ Automated Scan	36
4-3	Arachni รองรับ Automated Scan	36
4-4	Burp Suite ไม่รองรับ Automated Scan	37
4-5	ภาพผลการแสกนจาก Arachni	38
4-6	ภาพผลการแสดงจาก Owasp Zap	38
4-7	ตัวอย่าง Report ในรูปแบบ PDF จาก Owasp ZAP	41
4-8	ภาพ Report จาก Owasp ZAP	42
4-9	จำนวนผู้ประเมิณ 4 ท่านจากตำแหน่งงานต่าง ๆ	48
4-10	ผลการประเมิณข้อที่ 1-3	49
4-11	ผลการประเมิณข้อที่ 4-6	50

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในยุคดิจิทัลปัจจุบัน เว็บแอปพลิเคชันได้กลายเป็นหัวใจสำคัญในการขับเคลื่อนธุรกิจ การให้บริการ และการสื่อสารในทุกภาคส่วน อย่างไรก็ตาม การเติบโตนี้มาพร้อมกับความเสี่ยงด้านความปลอดภัยทางไซเบอร์ที่เพิ่มขึ้นอย่างทวีคูณ เว็บแอปพลิเคชันมักตกเป็นเป้าหมายหลักของการโจมตี เนื่องจากเป็นช่องทางตรงที่นำไปสู่ข้อมูลสำคัญขององค์กรและข้อมูลส่วนบุคคลของผู้ใช้งาน โดยจากสถิติในช่วงครึ่งแรกของปี 2024 พบว่าประเทศไทยมีความพยายามในการโจมตีทางเว็บเฉลี่ยสูงถึง 5,811 ครั้งต่อวัน และมีแนวโน้มการโจมตีแบบเจาะจงที่เว็บไซต์ (Hacked Website) เพิ่มสูงขึ้นอย่างต่อเนื่อง (Fortinet, 2024) [1] การโจมตีที่ประสบความสำเร็จอาจนำมาซึ่งความเสียหายร้ายแรง ทั้งในด้านการเงิน ชื่อเสียง และความน่าเชื่อถือขององค์กร โดยในปี 2025 ค่าใช้จ่ายเฉลี่ยทั่วโลกจากเหตุการณ์ข้อมูลรั่วไหลสูงถึง 4.88 ล้านดอลลาร์สหรัฐ ซึ่งเพิ่มขึ้น 10% จากปีก่อนหน้า (IBM, 25025) [2]

เพื่อรับมือกับความท้าทายดังกล่าว "เครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชัน" ได้กลายเป็นเครื่องมือพื้นฐานที่องค์กรและนักพัฒนาใช้ในการค้นหาจุดอ่อนและความเสี่ยงเชิงรุกก่อนที่ผู้ไม่หวังดีจะค้นพบ อย่างไรก็ตาม ในปัจจุบันมีเครื่องมือให้เลือกใช้งานหลากหลาย ทั้งแบบ Open-source ที่ไม่มีค่าใช้จ่าย และแบบ Commercial ที่มีฟังก์ชันการทำงานขั้นสูง ซึ่งแต่ละเครื่องมือก็มีจุดแข็ง จุดอ่อน และประสิทธิภาพในการตรวจจับช่องโหว่ที่แตกต่างกันไป และในการเลือกใช้เครื่องมือที่ไม่เหมาะสมอาจนำไปสู่ปัญหาสำคัญสองประการที่อาจเกิดขึ้นได้ คือ ผลบวกลวง (False Positives) การที่เครื่องมือแจ้งเตือนว่ามีช่องโหว่ทั้งที่ไม่มีอยู่จริง ทำให้ทีมพัฒนาต้องเสียเวลาและทรัพยากรไปกับการตรวจสอบและแก้ไขในสิ่งที่ไม่ใช่ปัญหา และผลลบลวง (False Negatives) การที่เครื่องมือ "ไม่สามารถ" ตรวจจับช่องโหว่ที่มีอยู่จริง ซึ่งเป็นสถานการณ์ที่อันตรายที่สุด เพราะทำให้องค์กรเข้าใจผิดว่าระบบของตนปลอดภัย และปล่อยให้ช่องโหว่ร้ายแรงนั้นคงอยู่ต่อไป

ยิ่งไปกว่านั้น วงจรการจัดการความปลอดภัยไม่ได้สิ้นสุดลงเพียงแค่การค้นพบช่องโหว่ แต่ยังต้องครอบคลุมไปถึง "กระบวนการยืนยันผลการแก้ไข" (Remediation Validation) ซึ่งเป็นขั้นตอนที่

สำคัญอย่างยิ่งแต่กลับถูกมองข้ามไปบ่อยครั้ง การแก้ไขช่องโหว่โดยปราศจากการตรวจสอบและยืนยันผลที่ชัดเจน อาจทำให้ช่องโหว่เดิมยังคงอยู่ หรืออาจก่อให้เกิดปัญหาใหม่โดยไม่คาดคิด

ดังนั้น การศึกษาและเปรียบเทียบประสิทธิภาพของเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันควบคู่ไปกับการกำหนดกระบวนการยืนยันผลการแก้ไขที่ชัดเจนและเชื่อถือได้ จึงมีความสำคัญอย่างยิ่ง การค้นคว้าอิสระนี้จึงมีวัตถุประสงค์เพื่อวิเคราะห์และเปรียบเทียบประสิทธิภาพของเครื่องมือสแกนโดยจะยกตัวอย่างเครื่องมือแสกนในกลุ่ม Open-source พร้อมทั้งนำเสนอแนวทางการแก้ไขและกระบวนการตรวจสอบที่เป็นระบบ ผลลัพธ์จากการศึกษานี้จะเป็นประโยชน์ต่อองค์กรและนักพัฒนาในการตัดสินใจเลือกใช้เครื่องมือที่เหมาะสมกับบริบทของตนเอง และสามารถสร้างกระบวนการจัดการช่องโหว่ที่ครบวงจรและมีประสิทธิภาพ เพื่อยกระดับความปลอดภัยของเว็บแอปพลิเคชันให้สูงขึ้นอย่างยั่งยืน

1.2 วัตถุประสงค์ของการวิจัย

โครงการค้นคว้าอิสระนี้มีวัตถุประสงค์หลัก 3 ประการ ดังนี้

1.2.1 เพื่อเปรียบเทียบประสิทธิภาพของเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชัน ทำการประเมินและเปรียบเทียบเครื่องมือที่เลือกมา (OWASP ZAP, Arachni, Burp Suite)

1.2.2 เพื่อนำเสนอแนวทางการแก้ไขและกระบวนการยืนยันผลสำหรับช่องโหว่ที่ตรวจพบ วิเคราะห์หาสาเหตุและนำเสนอขั้นตอนการแก้ไขอย่างเป็นระบบ

1.2.3 เพื่อจัดทำข้อเสนอแนะในการเลือกใช้เครื่องมือและปรับปรุงกระบวนการพัฒนาเว็บแอปพลิเคชันให้ปลอดภัย สรุปผลการวิจัยทั้งหมดเพื่อให้ข้อเสนอแนะที่เป็นประโยชน์แก่นักพัฒนาหรือผู้ที่เกี่ยวข้อง

1.3 ขอบเขตของการวิจัย

1.3.1 ขอบเขตด้านเครื่องมือ (Tools Scope)

การวิจัยนี้จะทำการศึกษาและเปรียบเทียบประสิทธิภาพของเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันประเภท Open-Source จำนวน 3 ตัว ได้แก่ OWASP ZAP, Arachni และ Burp Suite โดยพิจารณาจากการเลือกใช้งานขององค์กร ในมิติต่าง ๆ เช่น ความครอบคลุมในการตรวจจับช่องโหว่, ความแม่นยำและคุณภาพของรายงานผล เพื่อวิเคราะห์จุดแข็งและจุดอ่อนของแต่ละเครื่องมือ

1.3.2 ขอบเขตด้านสภาพแวดล้อม (Environment Scope)

1.3.2.1 การทดสอบทั้งหมดจะดำเนินการใน สภาพแวดล้อมที่จัดเตรียมขึ้นสำหรับการทดสอบ เท่านั้น เพื่อควบคุมตัวแปรและป้องกันผลกระทบต่อระบบที่ใช้งานจริง

1.3.2.2 เว็บแอปพลิเคชันที่ใช้ในการทดสอบจะเป็นเว็บแอปพลิเคชันที่มีช่องโหว่ตามที่กำหนดไว้ เพื่อใช้เป็นเกณฑ์มาตรฐาน (Benchmark) ในการวัดประสิทธิภาพของเครื่องมือ

1.3.3 ขอบเขตด้านช่องโหว่ (Vulnerability Scope)

1.3.3.1 การวิจัยจะมุ่งเน้นการตรวจจับช่องโหว่ที่พบบ่อยและมีความเสี่ยงสูงตามมาตรฐานสากล เช่น OWASP Top 10[3] เป็นต้น

1.3.3.2 จะมีการปรับลดช่องโหว่ทั้งหมดเท่าที่จะสามารถปรับปรุงได้ โดยวัดผลการปรับลดเป็นเปอร์เซ็นต์เพื่อดูผลที่ได้รับ

1.3.4 ขอบเขตด้านกระบวนการ (Process Scope)

1.3.4.1 การวิจัยจะครอบคลุมตั้งแต่ขั้นตอนการวางแผน, การสแกนและรวบรวมข้อมูล, การวิเคราะห์ผล, การลงมือแก้ไข, และการตรวจสอบผลการแก้ไข

1.3.4.2 การยืนยันผลการแก้ไขจะใช้วิธีการ สแกนซ้ำ (Verification Scan) ควบคู่ไปกับการตรวจสอบด้วยตนเอง (Manual Verification) เพื่อให้มั่นใจว่าช่องโหว่ถูกปิดอย่างสมบูรณ์

1.4 วิธีการดำเนินการวิจัย

1.4.1 กำหนดวัตถุประสงค์และขอบเขต

1.4.1.1 ตั้งเป้าหมายการวิจัยและคำถามที่ต้องการคำตอบเลือกเว็บแอปพลิเคชันเป้าหมายและขออนุญาตในการแสกน

1.4.2 เตรียมสภาพแวดล้อมและเครื่องมือ

1.4.2.1 ติดตั้งและตั้งค่า OWASP ZAP, Arachni, Burp Suite เตรียมสภาพแวดล้อมทดสอบ (Test Environment)

1.4.3 สแกนหาช่องโหว่ และเปรียบเทียบเครื่องมือ

1.4.3.1 ดำเนินการสแกนเว็บแอปพลิเคชันด้วยเครื่องมือทั้ง 3 ตัวอย่างเป็นระบบ เพื่อเปรียบเทียบ performance ของแต่ละโปรแกรมที่ยกมา

1.4.3.2 Export รายงานผลการสแกนจากแต่ละเครื่องมือ เพื่อ สร้างตาราง Matrix เปรียบเทียบช่องโหว่ที่พบ ว่าแต่ละโปรแกรมเจอช่องโหว่ไหนที่ต่างหรือเหมือนกันบ้างอย่างไร

1.4.3.3 วิเคราะห์หาจุดร่วมและจุดต่างของผลลัพธ์ ประเมินประสิทธิภาพเครื่องมือ สรุปจุดแข็ง, จุดอ่อน, และความเหมาะสมในการใช้งานของแต่ละเครื่องมือ เพื่อเลือก 1 เครื่องมือมาใช้ในการแสกนเพื่อแก้ไขผลการแสกนช่องโหว่ต่อไป

1.4.4 ดำเนินการแก้ไขช่องโหว่ (Remediation)

1.4.4.1 วิเคราะห์สาเหตุเชิงลึกของช่องโหว่ทั้งหมดและเสนอแนวทางแก้ไข

1.4.4.2 นำเสนอแนวทางการแก้ไขให้ทีมที่รับผิดชอบ เพื่อทำการปรับปรุงโค้ดหรือการตั้งค่าเพื่อปิดช่องโหว่ที่ค้นพบต่อไป

1.4.5 สแกนซ้ำเพื่อยืนยันผล (Verification)

1.4.5.1 เมื่อได้รับแจ้งรายงานการปิดช่องโหว่ที่เสนอแนะไปเรียบร้อยแล้ว ใช้เครื่องมือที่เลือกมาทำการสแกนซ้ำเพื่อพิสูจน์ว่าการแก้ไขได้ผล ช่องโหว่ถูกแก้ไขไปประมาณกี่เปอร์เซ็นต์ และตรวจสอบว่าไม่มีผลกระทบข้างเคียงใด ๆ (Regression Test) ต่อระบบ

1.4.6 จัดทำรายงานผลและข้อเสนอแนะ เขียนรายงานการวิจัยฉบับสมบูรณ์ ให้ข้อเสนอแนะสำหรับการพัฒนาเว็บแอปพลิเคชันที่ปลอดภัย

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้ข้อมูลเปรียบเทียบประสิทธิภาพเชิงลึกของเครื่องมือ Open-Source ทำให้ทราบถึงจุดแข็ง, จุดอ่อน, และข้อจำกัดของเครื่องมือสแกนช่องโหว่แต่ละตัวที่ยกมาในการค้นคว้านี้ (OWASP ZAP, Arachni, Burp Suite) ในสถานการณ์จริง ช่วยให้สามารถเลือกใช้เครื่องมือที่เหมาะสมกับประเภทของงานและบริบทขององค์กรได้ดียิ่งขึ้น

1.5.2 ได้แนวทางปฏิบัติที่เป็นรูปธรรมสำหรับการจัดการช่องโหว่ สร้างกระบวนการที่ชัดเจนและครบวงจร ตั้งแต่การตรวจจับช่องโหว่, การวิเคราะห์, การแก้ไข, ไปจนถึงการตรวจสอบยืนยันผล ซึ่งสามารถนำไปปรับใช้เป็นแนวทางมาตรฐานในการทำงานได้

1.5.3 ช่วยลดความเสี่ยงด้านความปลอดภัยทางไซเบอร์ ผลลัพธ์จากการวิจัยจะช่วยให้นักพัฒนาและองค์กรสามารถค้นหาและปิดช่องโหว่ของเว็บแอปพลิเคชันได้อย่างมีประสิทธิภาพมากขึ้น ซึ่งเป็นการช่วยลดความเสี่ยงจากการถูกโจมตีและป้องกันความเสียหายที่อาจเกิดขึ้นกับข้อมูลและชื่อเสียงขององค์กร

1.5.4 เพิ่มประสิทธิภาพและลดเวลาในการทำงาน การมีข้อมูลเปรียบเทียบที่ชัดเจนช่วยลดเวลาในการตัดสินใจเลือกเครื่องมือ และการมีกระบวนการยืนยันผลที่เป็นระบบจะช่วยลดการทำงานซ้ำซ้อนในการตรวจสอบช่องโหว่ที่อาจแก้ไขไม่สมบูรณ์

1.5.5 เป็นองค์ความรู้สำหรับนักพัฒนาและผู้ที่สนใจ ผลการวิจัยสามารถใช้เป็นเอกสารอ้างอิงและเป็นแนวทางสำหรับนักพัฒนา, นักศึกษา, หรือผู้ที่สนใจด้านความปลอดภัยของเว็บแอปพลิเคชันในการเริ่มต้นศึกษาและนำไปประยุกต์ใช้ในการพัฒนาระบบที่ปลอดภัย (Secure Coding) ต่อไป

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทที่ 2
แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

ในการศึกษาเรื่อง การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข นั้นผู้ศึกษาได้รวบรวมหลักการ แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง เพื่อใช้เป็นฐานความรู้และกรอบแนวคิดในการวิจัย โดยแบ่งเนื้อหาออกเป็นประเด็นสำคัญดังนี้

2.1 แนวคิดและทฤษฎีเกี่ยวกับความมั่นคงปลอดภัยของแอพพลิเคชั่น

ในสภาวการณ์ปัจจุบันที่การดำเนินธุรกรรมทางอิเล็กทรอนิกส์ขยายตัวอย่างรวดเร็ว เว็บแอปพลิเคชันกลายเป็นเป้าหมายหลักของการโจมตีทางไซเบอร์ เนื่องจากเป็นส่วนที่ต้องเปิดให้บุคคลภายนอกเข้าถึงได้ผ่านโครงข่ายอินเทอร์เน็ต แนวคิดด้านความมั่นคงปลอดภัยที่สำคัญประกอบด้วย

2.1.1 หลักการพื้นฐาน CIA Triad การรักษาความมั่นคงปลอดภัยสารสนเทศมุ่งเน้นที่การคุ้มครององค์ประกอบ 3 ประการ แสดงดังภาพที่ 2-1



ภาพที่ 2-1 CIA Triad

2.1.1.1 ความลับ (Confidentiality) การรับรองว่าข้อมูลจะถูกเข้าถึงได้เฉพาะผู้ที่มีสิทธิ์เท่านั้นความ

2.1.1.2 ถูกต้องครบถ้วน (Integrity) การรักษาความถูกต้องของข้อมูลไม่ให้ถูกแก้ไขหรือเปลี่ยนแปลงโดยมิชอบ

2.1.1.3 ความพร้อมใช้งาน (Availability) การทำให้ระบบและข้อมูลพร้อมใช้งานเมื่อมีความต้องการอย่างสม่ำเสมอ

2.1.2 มาตรฐาน OWASP Top 10 Open Web Application Security Project (OWASP)

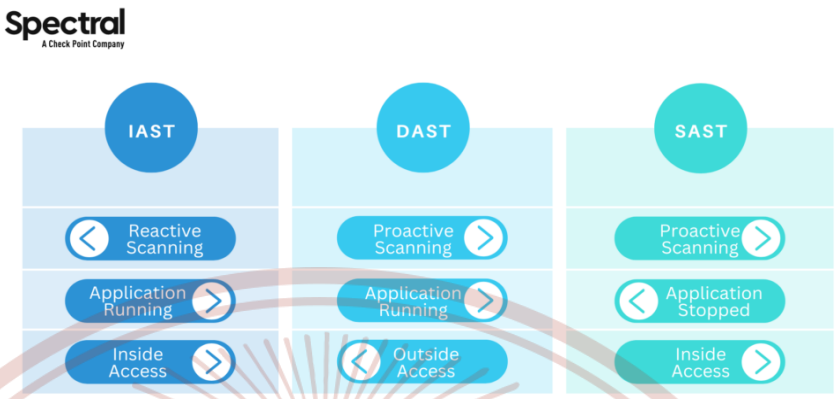
NT cyfence จากบริษัท โทรคมนาคมแห่งชาติ จำกัด (มหาชน) ได้ให้ข้อมูลเกี่ยวกับ OWAS Top Ten ไว้สามารถสรุป ได้ดังนี้ OWAS Top Ten คือโครงการของ OWASP ที่จัดอันดับความเสี่ยงทางด้านความมั่นคงปลอดภัยของเว็บแอปพลิเคชันไว้ 10 อันดับ โดยทุกๆ 4 ปี จะมีการจัดอันดับช่องโหว่ใหม่เพื่อให้องค์กรทั่วโลกนำข้อมูลไปใช้เพื่อปกป้องเว็บแอปพลิเคชันขององค์กร สำหรับ OWASP Top Ten จะแสดงดังภาพที่ 2-2



ภาพที่ 2-2 2021 OWASP Top 10 list

2.2 แนวคิดเกี่ยวกับเทคโนโลยีการตรวจหาช่องโหว่เว็บแอปพลิเคชัน

เทคโนโลยีที่ใช้ในการค้นหาและวิเคราะห์ช่องโหว่ (Vulnerability Assessment) สามารถจำแนกตามลักษณะการทดสอบได้เป็น 3 ประเภทหลัก แสดงดังภาพที่ 2-3



ภาพที่ 2-3 Vulnerability Assessment

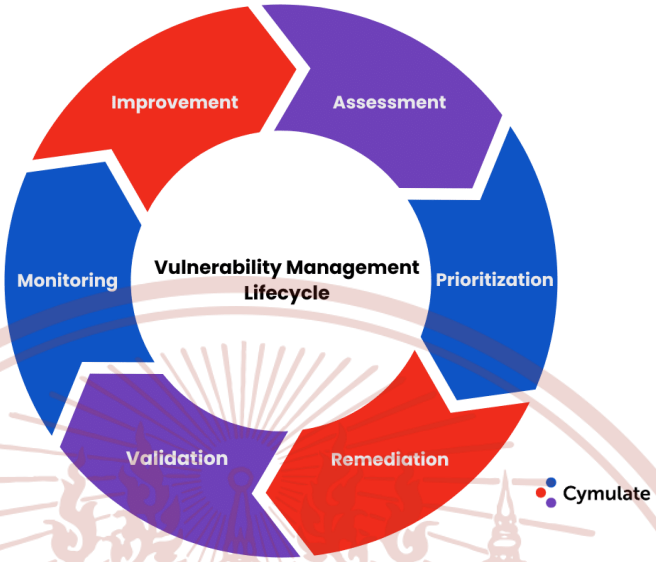
2.2.1 Static Application Security Testing (SAST) เป็นการตรวจสอบช่องโหว่ในลักษณะ "กล่องขาว" (White-box Testing) โดยการวิเคราะห์ซอร์สโค้ด (Source Code) หรือไบนารีของแอปพลิเคชันโดยไม่มีการรันระบบ ช่วยให้ตรวจพบข้อผิดพลาดในตรรกะการเขียนโปรแกรมได้ตั้งแต่ระยะเริ่มต้นของการพัฒนา

2.2.2 Dynamic Application Security Testing (DAST) เป็นการตรวจสอบในลักษณะ "กล่องดำ" (Black-box Testing) โดยเครื่องมือจะจำลองบทบาทเป็นผู้โจมตีเพื่อส่งชุดคำสั่งทดสอบ (Payload) เข้าไปยังแอปพลิเคชันที่กำลังทำงานอยู่ (Runtime) ผ่านอินเทอร์เฟซของเว็บ เพื่อสังเกตการตอบสนองและระบุช่องโหว่ที่เกิดขึ้นจริงในสภาวะการใช้งานจริง

2.2.3 Interactive Application Security Testing (IAST) เป็นเทคโนโลยีแบบผสมผสาน (Hybrid) โดยการติดตั้งโปรแกรมตรวจวัด (Agent) ไว้ภายในแอปพลิเคชัน เพื่อวิเคราะห์การทำงานภายในร่วมกับการทดสอบแบบ Dynamic ซึ่งช่วยเพิ่มความแม่นยำและลดอัตราการแจ้งเตือนที่ผิดพลาด

2.3 วงจรการบริหารจัดการช่องโหว่ (Vulnerability Management Lifecycle)

วงจรการบริหารจัดการช่องโหว่ประกอบด้วย 6 ขั้นตอนหลัก ซึ่งเป็นกรอบการดำเนินงานเชิงระบบที่ใช้ในการระบุ ประเมิน และจัดการความเสี่ยง โดยในการศึกษานี้ได้นำมาประยุกต์ใช้เพื่อกำหนดขั้นตอนการทดสอบเปรียบเทียบเครื่องมือ แสดงดังภาพที่ 2-4



ภาพที่ 2-4 Vulnerability Management Lifecycle

2.3.1 ขั้นตอนที่ 1 การสำรวจและจัดทำบัญชีสินทรัพย์ (Asset Discovery and Inventory Management) การดำเนินงานเริ่มต้นด้วยการกำหนดขอบเขตของระบบเป้าหมาย (Scope) โดยในการศึกษานี้คือการระบุและวิเคราะห์โครงสร้างของเว็บแอปพลิเคชันที่ใช้เป็นตัวทดสอบ (เช่น เว็บแอปพลิเคชันจำลองที่มีช่องโหว่มาตรฐาน) เพื่อให้เครื่องมือสแกนทั้ง 3 ชนิด ได้แก่ OWASP ZAP, Arachni และ Burp Suite ทำการตรวจสอบบนฐานข้อมูลสินทรัพย์และเส้นทางเชื่อมโยง (Endpoints) ที่เท่าเทียมกัน

2.3.2 ขั้นตอนที่ 2 การประเมินช่องโหว่แบบอัตโนมัติ (Automated Vulnerability Assessment) เป็นขั้นตอนการใช้ความสามารถของเครื่องมือสแกนอัตโนมัติในการตรวจหาจุดอ่อน โดยจะทำการเปรียบเทียบความสามารถในการตรวจจับ (Detection Capabilities) ของเครื่องมือแต่ละประเภท ข้อมูลที่ได้ในขั้นตอนนี้คือรายการช่องโหว่ที่ตรวจพบในสถานะเริ่มต้น (Initial Scan) ซึ่งจะถูกนำไปใช้เป็นข้อมูลดิบในการวิเคราะห์ประสิทธิภาพเชิงเปรียบเทียบต่อไป

2.3.3 ขั้นตอนที่ 3 การจัดลำดับความสำคัญตามระดับความเสี่ยง (Risk-Based Prioritization) ข้อมูลช่องโหว่ที่ได้รับจากเครื่องมือจะถูกนำมาประเมินและจัดลำดับความสำคัญตามบริบทของความเสี่ยง (Contextual Intelligence) โดยใช้เกณฑ์มาตรฐาน เช่น CVSS ในขั้นตอนนี้จะมีการวิเคราะห์ความแม่นยำของเครื่องมือแต่ละชนิด เพื่อจำแนกความแตกต่างระหว่างช่องโหว่ที่เกิดขึ้นจริงและข้อมูลที่แจ้งเตือนผิดพลาด (False Positive)

2.3.4 ขั้นตอนที่ 4 การดำเนินการแก้ไขและตอบสนอง (Orchestrated Remediation and Response) ภายหลังจากการวิเคราะห์และจัดลำดับความสำคัญ จะเข้าสู่กระบวนการแก้ไขช่องโหว่ (Remediation) โดยการปรับปรุงซอร์สโค้ด (Source Code Patching) หรือการปรับตั้งค่าระบบตามคำแนะนำของเครื่องมือ ขั้นตอนนี้เป็นส่วนสำคัญในการเตรียมสภาพแวดล้อมเพื่อเข้าสู่กระบวนการทดสอบประสิทธิภาพของการยืนยันผลในลำดับถัดไป

2.3.5 ขั้นตอนที่ 5 การยืนยันผลและการเฝ้าระวังอย่างต่อเนื่อง (Continuous Validation and Monitoring) ขั้นตอนนี้เป็นส่วนสำคัญของการศึกษาเพื่อประเมินประสิทธิภาพใน "กระบวนการยืนยันผลการแก้ไข" โดยการใช้เครื่องมือทั้ง 3 ชนิดทำการสแกนซ้ำ (Re-scanning) เพื่อตรวจสอบความถูกต้องของการแก้ไข ผลลัพธ์ที่ได้จะแสดงให้เห็นถึงความสามารถของเครื่องมือแต่ละตัวในการระบุสถานะของช่องโหว่ว่าได้รับการแก้ไขอย่างสมบูรณ์แล้วหรือไม่

2.3.6 ขั้นตอนที่ 6 การรายงานผลและการปรับปรุงอย่างต่อเนื่อง (Comprehensive Reporting and Continuous Improvement) ขั้นตอนสุดท้ายคือการรวบรวมข้อมูลทั้งหมดมาวิเคราะห์ผลเชิงสถิติ เพื่อจัดทำรายงานเปรียบเทียบประสิทธิภาพเชิงลึก โดยมุ่งเน้นที่ดัชนีชี้วัดหลัก เช่น อัตราความแม่นยำ ระยะเวลาในการสแกน และประสิทธิภาพในการยืนยันผลการแก้ไข เพื่อสรุปเป็นข้อเสนอแนะในการเลือกใช้เครื่องมือให้เหมาะสมกับบริบทขององค์กรและการพัฒนาความมั่นคงปลอดภัยอย่างยั่งยืน

2.4 งานวิจัยที่เกี่ยวข้อง

2.4.1 Belay, T. E., Gupta, S., & Kumar, A. (2025) [4] ได้ทำการศึกษาในหัวข้อ "Systematic Evaluation and Model Based Selection of Web Vulnerability Scanners" โดยนำเสนอแนวทางการประเมินดังนี้ นำเครื่องมือสแกน (WVS) หลายตัวมาทดสอบกับเว็บแอปพลิเคชันที่จงใจสร้างให้มีช่องโหว่ โดยใช้กรอบการประเมินแบบถ่วงน้ำหนัก (Weighted KPI) ที่มีความยืดหยุ่น สามารถปรับเปลี่ยนค่าน้ำหนักความสำคัญและจัดลำดับตามความต้องการของผู้ใช้งานองค์กรได้จริง

อย่างไรก็ตาม งานวิจัยดังกล่าวยังมีข้อจำกัดตรงที่ผลลัพธ์จำกัดอยู่แคในกลุ่มเครื่องมือและชุดทดสอบที่เลือกใช้ ซึ่งอาจทำให้ผลคะแนนเอนเอียง และการทดสอบบนเว็บแอปพลิเคชันจำลองอาจไม่ครอบคลุมสถานการณ์การนำไปใช้งานในโลกความเป็นจริง

2.4.2 Escarrone, T., Rosa, R. L., Kreutz, D., et al. (2025) [5] ได้ทำการศึกษาในหัวข้อ " Sob a Lupa Corporativa Brasileira: Avaliação da Cobertura de Scanners de

Vulnerabilidades" โดยนำเสนอแนวทางการประเมินดังนี้ ทำการสแกนเปรียบเทียบด้วยเครื่องมือ 10 ตัวบนเว็บเป้าหมายที่เป็นมาตรฐาน (OWASP Juice Shop) และเว็บที่ใช้ JavaScript เป็นหลัก โดยใช้วิธีการศึกษากรณีตัวอย่าง (Case Study) จากสภาพแวดล้อมจริงในองค์กร และใช้คนตรวจสอบผลซ้ำเพื่อให้ผลลัพธ์มีความน่าเชื่อถือสูง

อย่างไรก็ตาม งานวิจัยดังกล่าวยังมีข้อจำกัดเนื่องจากเป็นการศึกษาเน้นกรณีขององค์กรเดียว และทดสอบบนเว็บแอปพลิเคชันแค่บางประเภท ทำให้ผลลัพธ์อาจไม่สามารถนำไปอ้างอิงกับองค์กรอื่นที่มีบริบททางวัฒนธรรมและโครงสร้างที่แตกต่างกันได้ทั้งหมดทั่วโลก

2.4.3 Potti, U.-S., Huang, H.-S., Chen, H.-T., et al. (2025) [6] ได้ทำการศึกษาในหัวข้อ "Security Testing Framework for Web Applications: Benchmarking ZAP V2.12.0 and V2.13.0" โดยนำเสนอแนวทางการประเมินดังนี้ นำโปรแกรม OWASP ZAP เวอร์ชัน 2.12.0 และ 2.13.0 มาสแกนเปรียบเทียบกันโดยตรงบนชุดทดสอบมาตรฐาน OWASP Benchmark ในสภาพแวดล้อมที่ถูกควบคุม ซึ่งช่วยแสดงให้เห็นถึงการเปลี่ยนแปลงประสิทธิภาพของเครื่องมือในแต่ละเวอร์ชันอย่างน่าเชื่อถือ

อย่างไรก็ตาม งานวิจัยดังกล่าวยังมีข้อจำกัดเนื่องจากศึกษาวิจัยแค่โปรแกรม OWASP ZAP เพียงตัวเดียวและใช้ชุดทดสอบเดียว จึงไม่สามารถนำผลไปอ้างอิงหรืออธิบายภาพรวมของเครื่องมือสแกนยี่ห้ออื่นทั้งหมดในตลาดได้

2.4.4 Jha, T. K. (2024) [7] ได้ทำการศึกษาในหัวข้อ "Modern Web Application Penetration Testing Tools: A Comparative Analysis" โดยนำเสนอแนวทางการประเมินดังนี้ นำเครื่องมือสแกนทั้งแบบเสียเงินและโอเพนซอร์สจำนวน 8 ตัวมาทดสอบบนแพลตฟอร์ม WebGoat และ DVWA โดยประเมินผลผ่านตัวชี้วัดที่หลากหลาย เช่น ค่าความแม่นยำ (Precision), ความครอบคลุม (Recall) และดัชนี Youden ตามเกณฑ์ของ OWASP ทำให้การวิเคราะห์ทางสถิติมีความรอบด้านและหนักแน่น

อย่างไรก็ตาม งานวิจัยดังกล่าวยังมีข้อจำกัดตรงที่การทดสอบจำกัดอยู่บนเว็บแอปพลิเคชันเพียง 2 ตัวอย่างและขอบเขตประเภทช่องโหว่มีจำกัด ซึ่งอาจมีข้อกังวลว่าผลลัพธ์ที่ได้จะไม่เหมือนกับการนำไปใช้งานกับเว็บแอปพลิเคชันในโลกความเป็นจริง

2.4.5 Kondraciuk, A., Bartos, A., & Pańczyk, B. (2022) [8] ได้ทำการศึกษาในหัวข้อ "Comparative Analysis of the Effectiveness of OWASP ZAP, Burp Suite, Nikto and

Skipfish" โดยนำเสนอแนวทางการประเมินดังนี้ ใช้วิธีเปรียบเทียบผลลัพธ์ "ก่อนและหลัง" การนำระบบเสริมประสิทธิภาพ JARVIS มาใช้งานร่วมกับเครื่องมือสแกน 5 ตัว บนเว็บแอปพลิเคชันที่ไม่ปลอดภัย 7 ตัว ซึ่งพิสูจน์ได้ว่าช่วยให้ค้นหาช่องโหว่เจอได้มากขึ้นอย่างเห็นได้ชัดและสามารถนำไปประยุกต์ใช้กับการสแกนแบบคู่ขนานได้จริง

อย่างไรก็ตาม งานวิจัยดังกล่าวยังมีข้อจำกัดที่การประเมินจำกัดอยู่แค่เว็บที่จงใจสร้างให้มีช่องโหว่ ยังไม่สามารถรับมือกับความหลากหลายของเว็บแอปพลิเคชันจริงได้อย่างเต็มที่ รวมถึงยังมีปัญหาการตรวจพบช่องโหว่เดียวกันซ้ำซ้อนและยากต่อการตรวจจับข้อผิดพลาดของตรรกะในโปรแกรม

2.4.6 Mburano, B., & Si, W. (2018) [9] ได้ทำการศึกษาในหัวข้อ "Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark" โดยนำเสนอแนวทางการประเมินดังนี้ ใช้การทดสอบแบบจำลองสถานการณ์โดยนำเสนอกรอบการทดสอบเปรียบเทียบที่พัฒนาขึ้นมาใหม่และครอบคลุมกว่าเดิม โดยนำเกณฑ์การประเมินที่ทันสมัยมาใช้ร่วมกับระบบการให้คะแนนเพื่อจัดอันดับเปรียบเทียบเครื่องมืออย่างเป็นระบบระเบียบและนำไปใช้ตัดสินใจได้จริง

อย่างไรก็ตาม งานวิจัยดังกล่าวยังมีข้อจำกัดเนื่องจากอันดับเครื่องมือจะเปลี่ยนไปตามค่าน้ำหนักคะแนนที่ตั้งไว้ การตั้งค่าตัวแปรในการจำลองอาจทำให้ผลลัพธ์เอนเอียง และระบบการให้คะแนนดังกล่าวยังผ่านการตรวจสอบจากการใช้งานจริงค่อนข้างน้อย

2.4.7 Alsaleh, M., Alomar, N., Alshreef, M., et al. (2017) [10] ได้ทำการศึกษาในหัวข้อ " Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners" โดยนำเสนอแนวทางการประเมินดังนี้ ใช้วิธีการทบทวนวรรณกรรมอย่างเป็นระบบ (Systematic Literature Review - SLR) วิเคราะห์งานวิจัย 90 ฉบับ เพื่อรวบรวมข้อมูลประเภทเครื่องมือสแกน ตัวชี้วัดประสิทธิภาพ (เช่น ความแม่นยำ) และสร้างแผนภาพรวมที่ชี้ให้เห็นถึงช่องว่างและความเอนเอียงในงานวิจัยที่ผ่านมา

อย่างไรก็ตาม งานวิจัยดังกล่าวยังมีข้อจำกัดเนื่องจากงานวิจัยพื้นฐานในเรื่องนี้ยังมีจำนวนไม่มากและแต่ละงานใช้วิธีการทดสอบที่แตกต่างกันเกินไป ทำให้ไม่สามารถนำข้อมูลมาวิเคราะห์ทางสถิติเพื่อหาข้อสรุปเพียงหนึ่งเดียวได้อย่างชัดเจน (Meta-analysis) และพบว่าส่วนใหญ่มักเน้นทดสอบแค่ช่องโหว่ที่รู้จักกันดี เช่น SQLi และ XSS โดยจะสรุปได้ดังตารางที่ 2-1 นี้

ตารางที่ 2-1 งานวิจัยที่เกี่ยวข้อง

ปี	เรื่อง	วิธีการ	จุดแข็ง	ข้อจำกัด
2025	Systematic Evaluation and Model Based Selection of Web Vulnerability Scanners	นำเครื่องมือสแกน WVS หลายๆตัวมาทดสอบกับเว็บแอปที่จงใจสร้างให้มีช่องโหว่,นำกรอบการประเมินแบบถ่วงน้ำหนัก มาใช้งาน ใช้ระบบให้คะแนนที่จัดลำดับความสำคัญตามความต้องการของผู้ใช้	มีกรอบการคัดเลือกที่ยืดหยุ่น เปลี่ยนค่าน้ำหนักความสำคัญได้ มีการทดสอบเปรียบเทียบและพิสูจน์ผลกับเครื่องมือสแกนหลายตัวจริงๆ ใช้เกณฑ์การคัดเลือกที่ตอบโจทย์การใช้งานจริงสำหรับองค์กร	ผลการวิจัยจำกัดอยู่แค่ในกลุ่มเครื่องมือที่นำมาทดสอบ ซึ่งมีผลต่ออันดับคะแนนและทำให้ผลลัพธ์เอนเอียงได้ กรอบการทำงานนี้ถูกพิสูจน์บนเว็บแอปที่มีช่องโหว่แค่บางอย่าง ไม่ครอบคลุมการนำไปใช้งานจริง
2025	Sob a Lupa Corporativa Brasileira: Avaliação da Cobertura de Scanners de Vulnerabilidades	สแกนเปรียบเทียบด้วยเครื่องมือ 10 ตัวที่แตกต่างกันทดสอบบน OWASP Juice Shop และเว็บที่ใช้ JavaScript เยอะ ใช้คนตรวจสอบผลการสแกนเพื่อยืนยันความถูกต้อง ใช้วิธีการศึกษากรณีตัวอย่าง จากสภาพแวดล้อมการทำงานจริงในองค์กร	ทดสอบเครื่องมือสแกน 10 ตัว เห็นภาพรวมที่กว้าง ใช้คนตรวจสอบผลซ้ำทำให้ผลลัพธ์มีความน่าเชื่อถือสูงสำหรับนำไปใช้งานจริง เน้นเว็บJavaScript ซึ่งเป็นเทคโนโลยีที่ใช้กันแพร่หลายในปัจจุบัน และได้ข้อมูลเชิงลึกจากสภาพแวดล้อมการใช้งานจริงในองค์กร	การศึกษาเน้นกรณีขององค์กรเดียว ผลลัพธ์ไม่สามารถนำไปอ้างอิงองค์กรอื่นได้ ทดสอบบนเว็บแอปแค่บางประเภทเท่านั้น บริบทวัฒนธรรมและโครงสร้างองค์กรอาจแตกต่างกันไปในแต่ละที่ ทำให้ผลลัพธ์อาจใช้ไม่ได้กับทุกแห่งทั่วโลก

ตาราง 2-1 (ต่อ)

ปี	เรื่อง	วิธีการ	จุดแข็ง	ข้อจำกัด
2025	Security Testing Framework for Web Applications: Benchmarking ZAP V2.12.0 and V2.13.0	นำโปรแกรม ZAP เวอร์ชัน 2.12.0 และ 2.13.0 มาสแกนเปรียบเทียบกันบนชุดทดสอบมาตรฐานเดียวกัน เพื่อเปรียบเทียบจากผลลัพธ์ที่ได้จากการทดลองจริงและนำข้อมูลผลการทำงานมาวิเคราะห์เพื่อวัดประสิทธิภาพ	เป็นการเปรียบเทียบประสิทธิภาพระหว่างเวอร์ชันเก่าและใหม่โดยตรงโดยใช้ OWASP Benchmark ซึ่งเป็นมาตรฐานที่น่าเชื่อถือ แสดงให้เห็นว่าประสิทธิภาพของเครื่องมือสแกนเปลี่ยนแปลงไปได้ในแต่ละเวอร์ชัน	วิจัยแค่โปรแกรม OWASP ZAP ตัวเดียวและยังใช้ชุดทดสอบ แค่ชุดเดียวในการประเมิน ทำให้ผลลัพธ์ที่ได้อาจไม่สามารถใช้อธิบายภาพรวมของเครื่องมือสแกนทั้งหมดในตลาดได้ และไม่สามารถไปอ้างอิงกับเครื่องมือสแกนยี่ห้ออื่นได้
2024	Modern Web Application Penetration Testing Tools: A Comparative Analysis	นำเครื่องมือสแกน 8 ตัวมาเปรียบเทียบจากการทดลองจริง ทำการทดสอบบนแพลตฟอร์ม WebGoat และ DVWA และมีการประเมินผลโดยใช้ค่าความแม่นยำ, ความครอบคลุมและดัชนี Youden โดยใช้วิธีการประเมินที่อิงตามเกณฑ์ของ OWASP	มีทั้งเครื่องมือแบบเสียเงินและโอเพนซอร์สทำให้เห็นภาพการเปรียบเทียบที่กว้างเพราะใช้ตัวชี้วัดหลายตัว ทำให้การประเมินผลมีความน่าเชื่อถือและรอบด้าน ทดสอบบนเว็บแอปพลิเคชันที่เป็นที่ยอมรับในวงการ และมีวิธีการวิเคราะห์ทางสถิติที่ครอบคลุม	การเปรียบเทียบสมรรถนะจำกัดอยู่แค่แอปพลิเคชันที่มีการทดสอบอยู่แค่บนเว็บแอปพลิเคชัน 2 ตัวอย่างเท่านั้น เครื่องมือแต่ละตัวมีประสิทธิภาพในการตรวจจับช่องโหว่แตกต่างกันมากเกินไปและขอบเขตของประเภทช่องโหว่ที่นำมาทดสอบมีจำกัด

ตาราง 2-1 (ต่อ)

ปี	เรื่อง	วิธีการ	จุดแข็ง	ข้อจำกัด
2022	Comparative Analysis of the Effectiveness of OWASP ZAP, Burp Suite, Nikto and Skipfish	นำระบบJARVIS มาใช้งานนำไปทดสอบกับเครื่องมือสแกน 5 ตัว บนเว็บแอปพลิเคชันที่ไม่ปลอดภัย 7 ตัวใช้วิธีเปรียบเทียบผลลัพธ์ "ก่อนและหลัง" การนำระบบมาใช้ ทำการทดลองโดยเปรียบเทียบผลระหว่างการสแกนแบบปกติ กับแบบที่ได้รับการเสริมประสิทธิภาพแล้ว	วิธีนี้ช่วยให้หาช่องโหว่เจอได้เยอะขึ้นอย่างเห็นได้ชัด ผลการทดลองยืนยันว่าดีขึ้นกับทุกแอปที่นำไปทดสอบ สามารถนำไปประยุกต์ใช้กับการสแกนแบบคู่ขนานด้วยเครื่องมือหลายตัวได้ และเป็นวิธีเสริมประสิทธิภาพที่จับต้องได้สามารถนำไปใช้งานได้จริง	การประเมินจำกัดอยู่แค่เว็บที่จงใจสร้างให้มีช่องโหว่ ไม่สามารถรับมือกับความหลากหลายของเว็บที่ใช้งานจริงได้อย่างเต็มที่ ยังมีประเด็นเรื่องที่เครื่องมือหลายตัวอาจตรวจเจอช่องโหว่เดียวกันซ้ำๆ กัน โปรแกรมยังคงเป็นสิ่งที่ตรวจจับได้ยากสำหรับเคสนี้
2018	Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark	ใช้การจำลองสถานการณ์นำเครื่องมือหลายๆ ตัวมาทดสอบ นำกรอบการทดสอบที่พัฒนาขึ้นมาใหม่มาใช้งานจริง มีการใช้ระบบการให้คะแนนเพื่อจัดอันดับเครื่องมือและขั้นตอนการประเมินเครื่องมืออย่างเป็นระบบ	เสนอกรอบการเปรียบเทียบที่ขยายขอบเขตครอบคลุมกว่าเดิม มีการนำเกณฑ์การประเมินใหม่ๆ ที่ทันสมัยเข้ามาใช้ด้วย ให้ผลการจัดอันดับเครื่องมือที่ชัดเจนและนำไปใช้ตัดสินใจได้จริง มีแนวทางในการเปรียบเทียบเครื่องมือที่เป็นระบบระเบียบ	อันดับขึ้นอยู่กับน้ำหนักคะแนนเปลี่ยนตามการให้ความสำคัญของหัวข้อผลลัพธ์อาจแตกต่างไปตามเงื่อนไขหากนำไปใช้ในกรณีที่มีข้อจำกัดที่ต่างออกไป ผลลัพธ์ก็อาจไม่เหมือนเดิมระบบการให้คะแนนนี้ยังผ่านการใช้งานจริงค่อนข้างน้อย

ตาราง 2-1 (ต่อ)

ปี	เรื่อง	วิธีการ	จุดแข็ง	ข้อจำกัด
2017	Performance-Based Comparative Assessment of Open-Source Web Vulnerability Scanners	ใช้วิธีการทบทวนวรรณกรรมอย่างเป็นระบบ (Systematic Literature Review - SLR) วิเคราะห์งานวิจัยทั้งหมด 90 ฉบับที่เกี่ยวข้องกับการประเมินเครื่องมือสแกน ดึงข้อมูลเกี่ยวกับประเภทของเครื่องมือสแกนและขอบเขตการตรวจจับช่องโหว่จากงานวิจัยเหล่านั้น วิเคราะห์เปรียบเทียบตัวชี้วัดประสิทธิภาพ (เช่น ความแม่นยำ) ที่แต่ละงานวิจัยรายงานไว้	ชี้ให้เห็นช่องว่างและอคติในงานวิจัย ทำให้รู้ว่างานวิจัยที่ผ่านมามีจุดอ่อนหรือความเอนเอียงตรงไหนบ้าง สร้างแผนภาพรวมของเครื่องมือสแกน รวบรวมข้อมูลเครื่องมือสแกนทั้งหมดที่เคยถูกประเมินผลเป็นตัวเลขมาไว้ในที่เดียว เป็นการรวบรวมข้อมูลจากงานวิจัยจำนวนมาก และทำให้เห็นว่าหัวข้อไหนที่ควรมีการทำวิจัยเพิ่มในอนาคต	งานวิจัยตั้งต้นมีน้อย ชี้ให้เห็นว่างานวิจัยพื้นฐานในเรื่องนี้ยังมีจำนวนไม่มาก และแต่ละงานวิจัยก็ใช้วิธีที่แตกต่างกันเกินไป เพราะในแต่ละงานวิจัยมีวิธีการทดสอบไม่เหมือนกัน จึงเป็นการยากที่จะนำผลมาเปรียบเทียบกันโดยตรง จึงทำให้ไม่สามารถนำข้อมูลทั้งหมดมาวิเคราะห์ทางสถิติเพื่อหาข้อสรุปที่ชัดเจนเพียงหนึ่งเดียวได้ และพบว่างานวิจัยส่วนใหญ่มักจะเน้นทดสอบแค่ช่องโหว่ที่รู้จักกันดี

บทที่ 3
วิธีการดำเนินการวิจัย

การค้นคว้าอิสระเรื่อง การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข เป็นการวิจัยเชิงทดลองและแก้ปัญหา ที่มุ่งเน้นการประเมินสมรรถนะของเครื่องมือที่เหมาะสมกับบริบทและข้อจำกัดขององค์กร พร้อมทั้งพัฒนากระบวนการยืนยันความถูกต้องของการจัดการช่องโหว่หรือการเสนอแนะการแก้ไข เพื่อยกระดับความมั่นคงปลอดภัยของเว็บแอปพลิเคชันให้มีประสิทธิภาพ โดยผู้วิจัยได้กำหนดวิธีการดำเนินการวิจัยตามลำดับขั้นตอน ดังต่อไปนี้

3.1 กรอบแนวคิดการวิจัย

กรอบแนวคิดการวิจัยนี้ได้รับการพัฒนาขึ้นโดยบูรณาการองค์ความรู้จากมาตรฐานความมั่นคงปลอดภัยสากล วงจรการบริหารจัดการช่องโหว่ และขั้นตอนการทดลองเชิงประยุกต์ โดยแบ่งออกเป็น 3 ส่วนหลัก ดังนี้

3.1.1 ปัจจัยนำเข้า (Input) ประกอบด้วย

3.1.1.1 มาตรฐานความปลอดภัยสากล ได้แก่ OWASP Top 10 (2021) ที่ใช้เป็นเกณฑ์ในการจำแนกประเภทช่องโหว่ และ NIST SP 800-115 (Technical Guide to Information Security Testing and Assessment) [11] ที่ใช้เป็นแนวทางหลักในการกำหนดวิธีการทดสอบความปลอดภัย

3.1.1.2 เครื่องมือสแกนช่องโหว่ ได้แก่ OWASP ZAP, Arachni และ Burp Suite Community Edition

3.1.1.3 เกณฑ์การวัดผลประสิทธิภาพ ได้แก่ และตัวชี้วัดเปอร์เซ็นต์การลดลงของช่องโหว่ (Vulnerability Reduction Rate)

3.1.1.4 สภาพแวดล้อมการทดลอง ได้แก่ เว็บแอปพลิเคชันเป้าหมายภายใต้บริบทขององค์กรและรายการช่องโหว่ตั้งต้นที่ตรวจพบ

3.1.2 กระบวนการ (Process) ประกอบด้วย

3.1.2.1 การทดลองสแกนเพื่อประเมินประสิทธิภาพเชิงเปรียบเทียบของเครื่องมือที่เลือกมาทั้ง 3 เครื่องมือ

3.1.2.2 การคัดเลือกเครื่องมือที่มีความเหมาะสมกับบริบทขององค์กรมากที่สุด

3.1.2.3 การวิเคราะห์สาเหตุเชิงลึก (Root Cause Analysis) และการจับคู่ทฤษฎีการแก้ไขให้ตรงกับประเภทของช่องโหว่

3.1.2.4 การจัดทำคู่มือและดำเนินการแก้ไขช่องโหว่ในระดับซอร์สโค้ดและระดับการตั้งค่าในตัวระบบ

3.1.2.5 การสแกนซ้ำเพื่อตรวจสอบและยืนยันผลการแก้ไข (Remediation Validation)

3.1.3 ผลลัพธ์ (Output) ประกอบด้วย

3.1.3.1 รายงานผลการเปรียบเทียบประสิทธิภาพของเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชัน

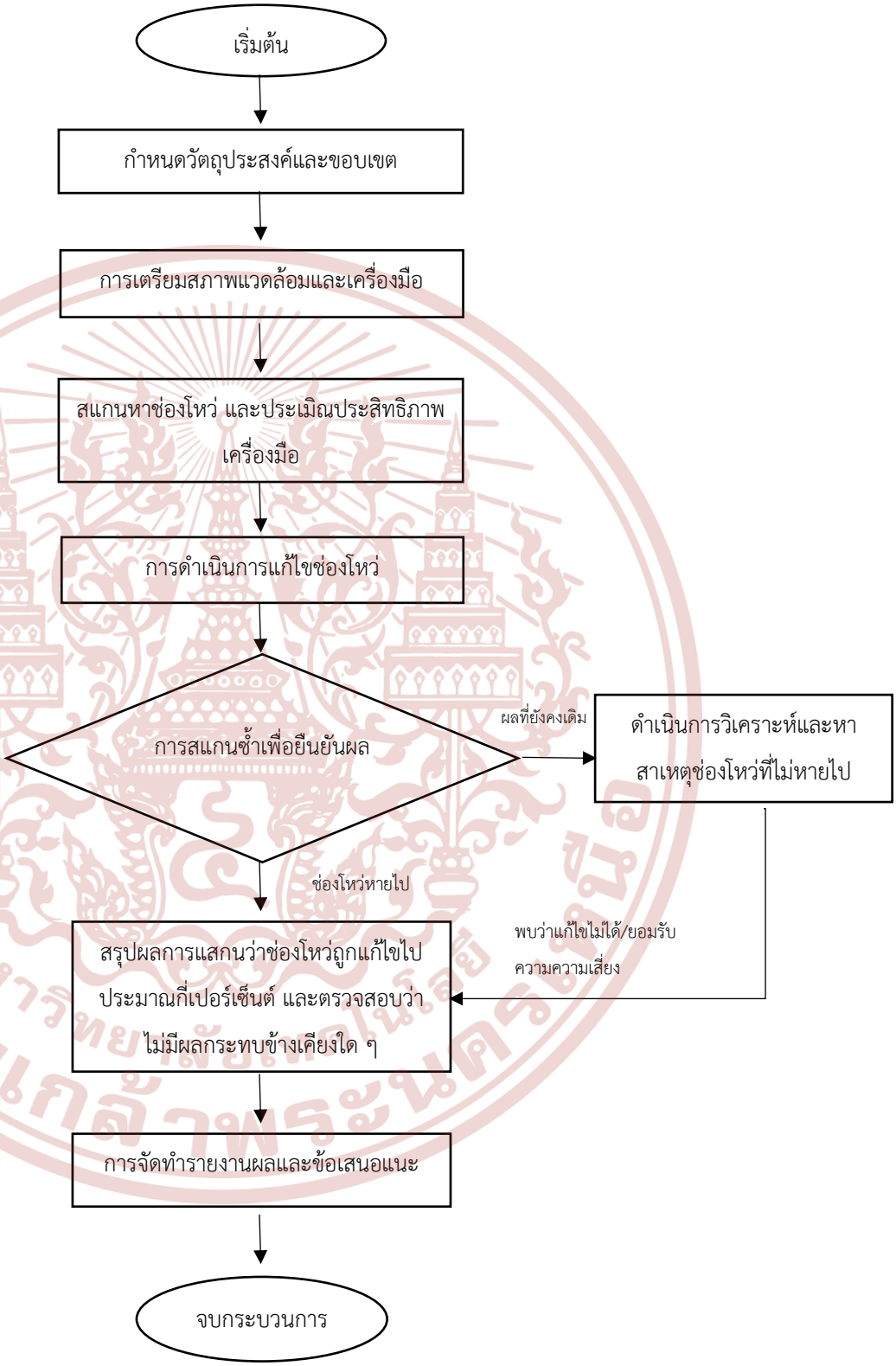
3.1.3.2 คู่มือปฏิบัติงานเพื่อการแก้ไขช่องโหว่ (Remediation Manual) สำหรับนักพัฒนาและผู้ดูแลระบบ

3.1.3.3 สรุปผลดัชนีชี้วัดความสำเร็จในรูปแบบเปอร์เซ็นต์การลดลงของช่องโหว่ภายหลังการดำเนินการแก้ไข

3.1.3.4 ข้อเสนอแนะเชิงนโยบายและแนวทางปฏิบัติในการยกระดับความมั่นคงปลอดภัยของเว็บแอปพลิเคชันอย่างยั่งยืน

3.2 ขั้นตอนการทำวิจัย

ผู้วิจัยได้กำหนดขั้นตอนการดำเนินงานวิจัยออกเป็น 6 ขั้นตอนหลัก เพื่อให้ครอบคลุมทั้งการประเมินประสิทธิภาพเครื่องมือและการยืนยันผลการแก้ไขช่องโหว่ตามมาตรฐานสากล ดังภาพ 3-1 นี้



ภาพที่ 3-1 ขั้นตอนและกระบวนการดำเนินงาน

(Process Diagram)

3.2.1 ขั้นตอนที่ 1 การกำหนดวัตถุประสงค์และขอบเขต (Define Objectives and Scope) ในขั้นตอนนี้ ผู้วิจัยมุ่งเน้นการวางรากฐานการดำเนินงานให้เป็นระบบเพื่อให้การทดลองมีกรอบที่ชัดเจน โดยมีรายละเอียดการดำเนินงานดังนี้

3.2.1.1 กำหนดเป้าหมายการวิจัยและคำถามวิจัยที่ต้องการคำตอบอย่างชัดเจน เพื่อให้การทดลองเป็นไปตามจุดมุ่งหมาย

3.2.1.1.1 ผู้วิจัยกำหนดเป้าหมายหลักเพื่อพิสูจน์ประสิทธิภาพของเครื่องมือสแกนช่องโหว่ทั้ง 3 ชนิด โดยได้กำหนดเกณฑ์ในการวิเคราะห์เปรียบเทียบเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชัน โดยแบ่งออกเป็น 4 มิติหลัก ครอบคลุมทั้งหมด 9 หัวข้อ เพื่อให้ได้ผลการวิเคราะห์ที่รอบด้านและสอดคล้องกับวัตถุประสงค์ของการวิจัย ดังตารางที่ 3-1

ตารางที่ 3-1 เป้าหมายหลักเพื่อพิสูจน์ประสิทธิภาพของเครื่องมือสแกนช่องโหว่

คุณสมบัติการทำงาน (Features)	OWASP ZAP	Arachni	Burp Suite (Community)
1. ด้านการบริหารจัดการและลิขสิทธิ์			
1.1. ใบอนุญาต (License)			
1.2. สถานะการอัปเดต (Maintenance)			
2. ด้านความสามารถทางเทคนิค			
2.1. การสแกนอัตโนมัติ (Automated Scan)			
2.2. การทดสอบแบบแมนนวล (Manual Tools)			
2.3. การรองรับ Modern Web (SPA/AJAX)			
3. ด้านประสิทธิภาพเชิงลึก			
3.1. ความครอบคลุมของประเภทช่องโหว่ (Vulnerability Coverage Analysis)			
3.2. การขยายขีดความสามารถ (Extensibility & Add-ons)			

ตารางที่ 3-1 (ต่อ)

4. ด้านการนำไปใช้งานและการบูรณาการ (Integration & Reporting)			
4.1. การเชื่อมต่อ CI/CD (DevSecOps)			
4.2. ความสามารถในการนำข้อมูลรายงานออกมาใช้งาน (Reporting)			
ผลรวม			

โดยมีเกณฑ์การกำหนดความหมายของสัญลักษณ์ดังนี้

✅ = ครบถ้วน ,ดีเยี่ยม ,รองรับ มีค่า 1 คะแนน

⚠️ = จำกัด, ไม่เสถียร มีค่า 0.5 คะแนน

❌ = ไม่มี, หยุดพัฒนา, ไม่รองรับ มีค่า 0 คะแนน

3.2.1.1.1.1 ด้านการบริหารจัดการและลิขสิทธิ์ (Management & Sustainability) เน้นความคุ้มค่า ความต่อเนื่อง และความเป็นไปได้ในการนำไปใช้งานจริงในระยะยาวขององค์กร ประกอบได้ด้วย ใบอนุญาต (License) พิจารณาถึงรูปแบบการถือครองลิขสิทธิ์และค่าใช้จ่ายในการใช้งาน โดยมุ่งเน้นที่ความคุ้มค่าและความสามารถในการเข้าถึงฟังก์ชันสำคัญในเวอร์ชันที่ไม่มีค่าใช้จ่าย (Open-Source หรือ Free Edition) เพื่อให้สอดคล้องกับข้อจำกัดด้านงบประมาณของหน่วยงาน และสถานะการอัปเดต (Maintenance) ประเมินความต่อเนื่องในการพัฒนาซอฟต์แวร์และการอัปเดตฐานข้อมูลช่องโหว่ (Vulnerability Signatures / CVEs) จากผู้พัฒนา เนื่องจากเครื่องมือที่มีการปรับปรุงอย่างสม่ำเสมอจะมีความสามารถในการตรวจจับภัยคุกคามรูปแบบใหม่ได้อย่างมีประสิทธิภาพมากกว่าเครื่องมือที่หยุดการพัฒนาไปแล้ว

3.2.1.1.1.2 ด้านความสามารถทางเทคนิค (Technical Capabilities) เน้นขีดความสามารถของเครื่องมือในการเข้าถึงและตรวจสอบระบบในเชิงลึก ประกอบไปด้วยประการแรก การสแกนอัตโนมัติ (Automated Scan) ประเมินความสามารถในการทำงานแบบอัตโนมัติ (Dynamic Application Security Testing - DAST) ตั้งแต่ขั้นตอนการสำรวจโครงสร้างเว็บไซต์ (Crawling) ไปจนถึงการส่ง Payload เพื่อทดสอบโจมตีโดยไม่ต้องพึ่งพาการสั่งการจากมนุษย์ในทุกขั้นตอน ประการที่สองการทดสอบแบบแมนนวล (Manual Tools) พิจารณาเครื่องมือ

เสริมที่ช่วยให้นักทดสอบความปลอดภัยสามารถดำเนินการตรวจสอบเชิงลึกได้ด้วยตนเอง เช่น Intercepting Proxy, Repeater และ Fuzzer เพื่อยืนยันช่องโหว่ในจุดที่ระบบอัตโนมัติไม่สามารถเข้าถึงได้ และสุดท้ายการรองรับ Modern Web (SPA/AJAX) ประเมินประสิทธิภาพในการวิเคราะห์เว็บแอปพลิเคชันสมัยใหม่ที่มีการทำงานซับซ้อน เช่น Single Page Application (SPA) หรือระบบที่ใช้ AJAX และ JavaScript อย่างหนัก ซึ่งเครื่องมือต้องมีความสามารถในการประมวลผลคำสั่งฝั่งไคลเอนต์ (Client-side Rendering) ได้อย่างถูกต้อง

3.2.1.1.1.3 ด้านประสิทธิภาพเชิงลึก (Performance & Quality) เน้นคุณภาพและความน่าเชื่อถือของผลลัพธ์ที่ได้จากการสแกน ประกอบไปด้วยการประเมิน คุณภาพของผลลัพธ์ที่ได้จากการสแกน โดยพิจารณาความแม่นยำเพื่อลดการรายงานที่ผิดพลาด (Accuracy) และความสามารถในการขยายขีดความสามารถ (Extensibility) ผ่านส่วนเสริมต่าง ๆ เพื่อให้เครื่องมือมีประสิทธิภาพสูงกว่าค่ามาตรฐานทั่วไป โดยประเมิณจาก ความครอบคลุมของประเภทช่องโหว่ (Vulnerability Coverage Analysis). การเปรียบเทียบเชิงปริมาณและระยะเวลา และ การวิเคราะห์ความครอบคลุมตามมาตรฐาน OWASP Top 10 และการขยายขีดความสามารถ (Extensibility & Add-ons) พิจารณาความยืดหยุ่นในการเพิ่มขีดความสามารถผ่านส่วนเสริม (Plugins) หรือการเขียนสคริปต์เพิ่มเติมจากชุมชนนักพัฒนา (Community Support) เพื่อให้เครื่องมือสามารถปรับตัวตามสภาพแวดล้อมที่เฉพาะเจาะจงได้

3.2.1.1.1.4 ด้านการนำไปใช้งานและการบูรณาการ (Integration & Reporting) เน้นการนำผลลัพธ์ไปส่งต่อให้ทีมพัฒนาเพื่อทำการแก้ไขอย่างเป็นรูปธรรม ประกอบไปด้วย การเชื่อมต่อ CI/CD (DevSecOps) ประเมินความสามารถในการบูรณาการเข้ากับวงจรการพัฒนาซอฟต์แวร์ (SDLC) ผ่านระบบอัตโนมัติ เช่น การรองรับ REST API หรือการใช้งานผ่าน Command Line เพื่อผลักดันแนวคิด DevSecOps ให้เกิดขึ้นจริงในองค์กรและความสามารถในการนำข้อมูลรายงานออกมาใช้งาน (Reporting) (เกณฑ์ตัดสินสำคัญ) พิจารณาความหลากหลายและความละเอียดของรูปแบบไฟล์รายงาน (เช่น HTML, XML, JSON) ซึ่งมีความจำเป็นอย่างยิ่งต่อการนำข้อมูลไปวิเคราะห์หาสาเหตุเชิงลึก (Root Cause Analysis) และการจัดทำคู่มือแก้ไขช่องโหว่ในขั้นตอนถัดไป

3.2.1.2 ดำเนินการเลือกเว็บแอปพลิเคชันเป้าหมายที่จะใช้ในการทดสอบ โดยคำนึงถึงความเหมาะสมของเทคโนโลยีที่ใช้พัฒนา (เช่น เว็บที่ใช้ JavaScript เป็นหลัก)

3.2.1.2.1 ผู้วิจัยเลือกใช้ระบบ UAT (User Acceptance Testing) ของหน่วยงาน (URL: https://devxxxxxx.xx.xxxxxxx.ac.th/) เป็นกรณีศึกษา

3.2.1.2.2 การคัดเลือกพิจารณาจากเทคโนโลยีที่ใช้พัฒนา (เช่น PHP, Apache) และโครงสร้างแบบ Modern Web ที่มีการประมวลผล JavaScript เพื่อให้เหมาะสมกับการทดสอบสมรรถนะของเครื่องมือสแกนช่องโหว่ยุคปัจจุบันที่ต้องรับมือกับความซับซ้อนของคำสั่งฝั่งไคลเอนต์และเอพีไอ (API)

3.2.1.3 ดำเนินการขออนุญาตอย่างเป็นทางการในการสแกนและทดสอบระบบ เพื่อให้ถูกต้องตามหลักจริยธรรมการวิจัยและความปลอดภัยของข้อมูล

3.2.1.3.1 ผู้วิจัยดำเนินการจัดทำหนังสือขออนุมัติถึงผู้ดูแลระบบหรือหน่วยงานที่รับผิดชอบ เพื่อกำหนดข้อตกลงในการทดสอบ (Rules of Engagement) ให้ชัดเจน

3.2.1.3.2 มีการระบุ ขอบเขตและประเภทของการสแกนที่จะไม่ส่งผลกระทบต่อความพร้อมใช้งานของระบบ (Availability) ทั้งนี้เพื่อให้การวิจัยเป็นไปตามหลักจริยธรรมของนักทดสอบความปลอดภัยสารสนเทศและกฎหมายที่เกี่ยวข้อง

3.2.2 ขั้นตอนที่ 2 การเตรียมสภาพแวดล้อมและเครื่องมือ (Environment and Tool Preparation) ในขั้นตอนนี้ ผู้วิจัยดำเนินการจัดเตรียมโครงสร้างพื้นฐานทางเทคนิคและปรับแต่งค่าการทำงานของเครื่องมือให้พร้อมสำหรับการทดสอบ โดยแบ่งการดำเนินงานออกเป็น 3 ส่วนหลัก ดังนี้

3.2.2.1 การเตรียมระบบปฏิบัติการและเครือข่ายสำหรับการทดลอง

3.3.2.1.1 ผู้วิจัยจัดเตรียมเครื่องคอมพิวเตอร์สำหรับการทดสอบ (Attack Machine) โดยใช้ระบบปฏิบัติการที่รองรับการทำงานของเครื่องมือสแกนทั้ง 3 ชนิดได้อย่างเสถียร

Edition: Windows 11 Enterprise version 25H2

Version: Windows Feature Experience Pack 1000.26100.291.0

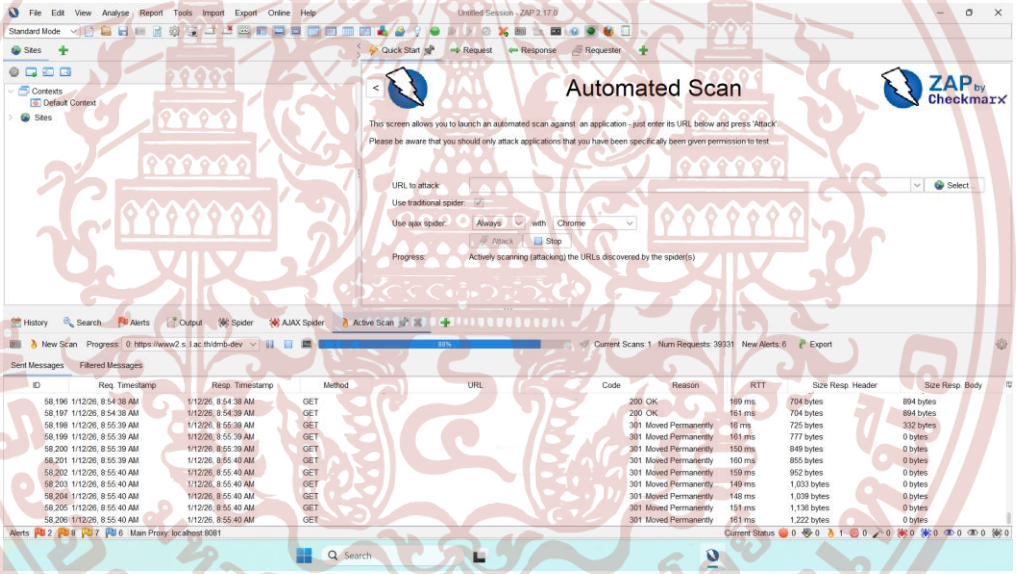
Processor: 12th Gen Intel(R) Core (TM) i5-1235U (1.30 GHz)

Installed RAM: 16.0 GB

3.2.2.1.2 ดำเนินการตรวจสอบการเชื่อมต่อเครือข่ายระหว่างเครื่องทดสอบและระบบเป้าหมาย (Target System) คือ เพื่อให้มั่นใจว่าไม่มีอุปกรณ์ป้องกันเครือข่าย เช่น Firewall หรือระบบป้องกันการบุกรุก (IPS) มาปิดกั้นการส่งข้อมูลของเครื่องมือสแกน ซึ่งอาจส่งผลให้ผลการทดลองคลาดเคลื่อน

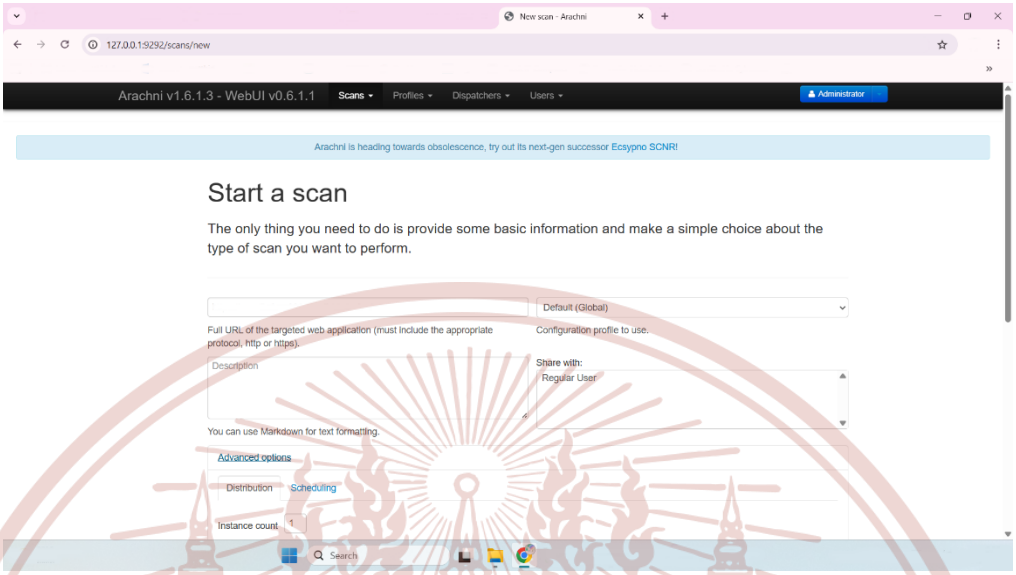
3.2.2.2 การติดตั้งและตั้งค่าเครื่องมือสแกนช่องโหว่

3.2.2.2.1 OWASP ZAP ทำการติดตั้งเวอร์ชันล่าสุด (นับจากช่วงเวลาในการทดสอบ)และตั้งค่าการทำงานในส่วนของ Standard Scan Policy รวมถึงการกำหนดค่า Context เพื่อให้เครื่องมือเข้าใจขอบเขตของเว็บแอปพลิเคชันเป้าหมาย แสดงผลดังภาพที่ 3-2



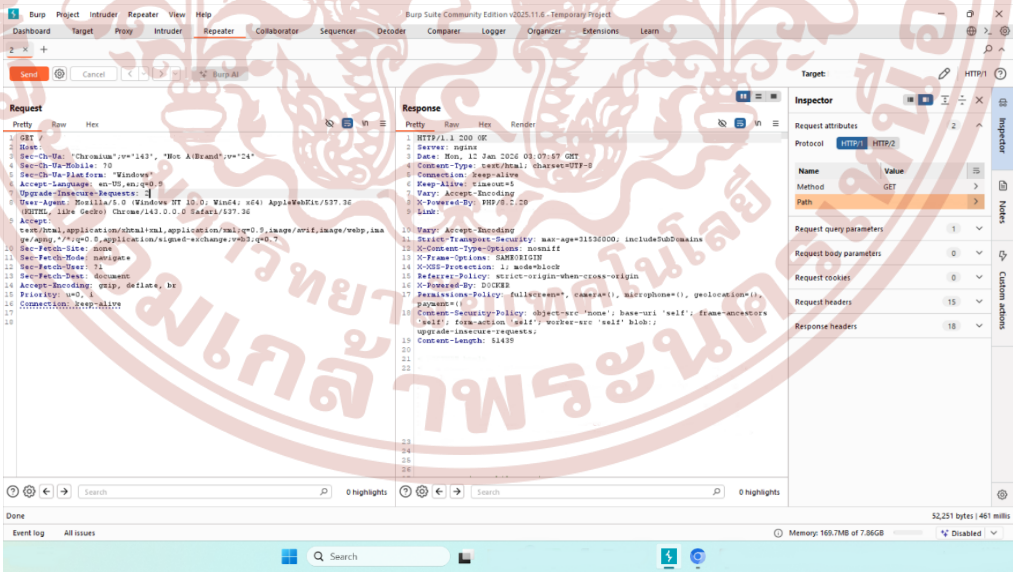
ภาพที่ 3-2 OWASP ZAP version 2.17.0

3.2.2.2.2 Arachni ดำเนินการติดตั้งและตรวจสอบความพร้อมของระบบ โดยกำหนดค่าโปรไฟล์การสแกนให้ครอบคลุมประเภทช่องโหว่พื้นฐานตามมาตรฐาน แสดงผลดังภาพที่ 3-3



ภาพที่ 3-3 Arachni version 1.6.1.3 -Web UI version 0.6.1.1

3.2.2.2.3 Burp Suite Community Edition ติดตั้งและตั้งค่า เพื่อเตรียมพร้อมสำหรับการดักจับข้อมูล รวมถึงการตั้งค่าเบราว์เซอร์ภายใน สำหรับการเข้าถึงหน้าเว็บที่ต้องการตรวจสอบ แสดงผลดังภาพที่ 3-3



ภาพที่ 3-4 Burp Suite Version Community Edition 2025.11.6

3.2.2.3 การตรวจสอบสภาพแวดล้อมก่อนการทดลอง (Pre-test Environment Validation) ผู้วิจัยดำเนินการตรวจสอบสถานะของเว็บแอปพลิเคชันเป้าหมายเพื่อให้มั่นใจว่าสภาพแวดล้อมในการทดลองมีความเสถียรและพร้อมสำหรับการเก็บข้อมูล โดยมีรายละเอียดดังนี้

3.2.2.3.1 ทำการตรวจสอบว่าเว็บแอปพลิเคชันเป้าหมาย สามารถเข้าถึงได้ตามปกติผ่านเครือข่าย และทดสอบฟังก์ชันการทำงานพื้นฐานที่ผู้ใช้ทั่วไปเข้าถึงได้ เช่น หน้าการแสดงผล และหน้าฟอร์มรับข้อมูล เพื่อยืนยันว่าไม่มีข้อผิดพลาดของระบบ (Runtime Error) ที่จะส่งผลต่อการทำงานของเครื่องมือสแกน

3.2.2.3.2 ดำเนินการสำรวจและบันทึกข้อมูลทางเทคนิคที่ตรวจพบจากภายนอก เช่น ประเภทของ Web Server และเทคโนโลยีที่ใช้ (Fingerprinting) เพื่อใช้เป็นข้อมูลประกอบการวิเคราะห์ เพื่อใช้เป็นจุดอ้างอิงในการเปรียบเทียบเชิงปริมาณภายหลังจากดำเนินการแก้ไขตามข้อเสนอแนะในขั้นตอนถัดไป

3.2.3 ขั้นตอนที่ 3 การสแกนหาช่องโหว่ และการเปรียบเทียบเครื่องมือ (Vulnerability Scanning and Tool Comparison) โดยอ้างอิงเป้าหมายจากตารางที่ 3-2

3.2.3.1 ผู้วิจัยทำการสั่งรันเครื่องมือสแกนช่องโหว่ทั้ง 3 ชนิด (OWASP ZAP, Arachni และ Burp Suite) เข้ากับระบบเป้าหมายทีละตัวภายใต้เงื่อนไขเดียวกัน คือกำหนดรูปแบบการสแกนเป็นแบบ "สแกนเชิงรุกแบบอัตโนมัติ (Automated Active Scan)"

3.2.3.2 ดำเนินการคัดเลือกเครื่องมือที่ให้ผลลัพธ์และมีความเหมาะสมกับบริบทขององค์กรมากที่สุดดีที่สุดเพียง 1 เครื่องมือ (กำหนดเกณฑ์ตามเป้าหมายจากหัวข้อที่ 3.2.1.1.1.) เพื่อนำมาใช้เป็นเครื่องมือหลักในการสแกนเพื่อยืนยันผลการแก้ไขในขั้นตอนถัดไป

3.2.4 ขั้นตอนที่ 4 การดำเนินการแก้ไขช่องโหว่ (Vulnerability Remediation)

3.2.4.1 นำเครื่องมือที่ดีที่สุดจากหัวข้อ 3.3.3.3. มาดำเนินการแสกนช่องโหว่ซ้ำอีกครั้งเพื่อดึงรีพอร์ตมาใช้ในการแก้ไขช่องโหว่ รูปแบบ Report HTML หรือ PDF

3.2.4.2 นำรายงานผลการสแกนจากเครื่องมือที่ได้รับการคัดเลือกมาวิเคราะห์หาสาเหตุเชิงลึก (Root Cause Analysis) ของช่องโหว่ทั้งหมดที่พบ แบ่งเป็นเกิดจากการเขียนซอร์สโค้ดที่หละหลวม (Code Level) หรือเกิดจากการตั้งค่าเซิร์ฟเวอร์ที่ผิดพลาด (Configuration Level)

3.2.4.2.1 สาเหตุในระดับซอร์สโค้ด (Code Level) มักพบในกลุ่มช่องโหว่ระดับสูง (High) และ ปานกลาง (Medium) ยกตัวอย่างเช่น การขาดการตรวจสอบข้อมูลนำเข้า (Input Validation) เว็บแอปพลิเคชันยอมรับค่าจากผู้ใช้โดยตรงโดยไม่มีการกรองคำสั่งอันตราย ทำให้เสี่ยงต่อการถูกโจมตีแบบ Injection หรือการจัดการเซสชันและสิทธิ์ที่ไม่รัดกุม การเขียนคำสั่งตรวจสอบตัวตนไม่ครอบคลุม ทำให้เครื่องมือสแกนสามารถเข้าถึงส่วนที่ควรมีการป้องกันได้ เป็นต้น

3.2.4.2.2 สาเหตุในระดับการตั้งค่าระบบและเซิร์ฟเวอร์ (Configuration Level) มักพบในกลุ่มช่องโหว่ระดับ ต่ำ (Low) และ แจ้งเพื่อทราบ (Informational) ซึ่งมีสาเหตุสำคัญส่วนมากจากการขาดหายไปของ Security Headers เซิร์ฟเวอร์ไม่ได้ถูกตั้งค่าให้ส่ง HTTP Response Headers ที่สำคัญ เช่น X-Content-Type-Options หรือ Content-Security-Policy เพื่อป้องกันการโจมตีทางเว็บเบราว์เซอร์ รวมไปถึงการเปิดเผยข้อมูลทางเทคนิค การตั้งค่าเริ่มต้นที่ยอมให้ระบบส่งข้อมูลเวอร์ชันของซอฟต์แวร์หรือรายละเอียดข้อผิดพลาด (Error Message) ออกมา ซึ่งผู้โจมตีสามารถนำไปใช้เป็นข้อมูลประกอบการโจมตีได้ เป็นต้น

3.2.4.3 เสนอแนวทางการแก้ไขเชิงเทคนิคที่สอดคล้องกับประเภทช่องโหว่ และนำเสนอต่อทีมงานที่รับผิดชอบเพื่อดำเนินการปรับปรุงซอร์สโค้ด (Source Code) หรือปรับปรุงการตั้งค่าระบบ (Configuration) ให้รัดกุมโดยจะมีการจัดทำคู่มือการแก้ไขช่องโหว่ที่เกิดขึ้น โดยจะมีขอบเขตองค์ประกอบ ดังตารางที่ 3-2

ตารางที่ 3-2 ขอบเขตองค์ประกอบของคู่มือการแก้ไขช่องโหว่

ชื่อของช่องโหว่	
ระดับความเสี่ยง (Severity)	
พารามิเตอร์ที่พบปัญหา	
ประเภทช่องโหว่	
คำอธิบายโดยสังเขป	
วิธีการแก้ไขช่องโหว่	
ตัวอย่างการแก้ไขช่องโหว่	

3.2.4.3.1 ชื่อของช่องโหว่ (Vulnerability Name) ระบุชื่อเรียกมาตรฐานของช่องโหว่ที่ตรวจพบ โดยอ้างอิงตามมาตรฐานสากล เช่น OWASP Top 10 หรือ CWE (Common Weakness Enumeration) ประโยชน์ เพื่อให้ผู้ที่เกี่ยวข้อง (นักพัฒนาและผู้ดูแลระบบ) มีความเข้าใจตรงกัน และสามารถสืบค้นข้อมูลเพิ่มเติมจากฐานข้อมูลสากลได้ทันที

3.2.4.3.2 ระดับความเสี่ยง (Severity) การระบุความรุนแรงของปัญหา โดยแบ่งเป็นระดับ (เช่น High, Medium, Low, Information) ซึ่งพิจารณาจากโอกาสในการเกิดเหตุ (Likelihood) และผลกระทบต่อระบบ (Impact) ประโยชน์ เพื่อช่วยในการจัดลำดับความสำคัญ (Prioritization) ว่าควรแก้ไขจุดใดก่อน เพื่อลดความเสี่ยงที่รุนแรงที่สุดขององค์กรภายใต้เวลาที่จำกัด

3.2.4.3.3 พารามิเตอร์ที่พบปัญหา (Affected Parameters) คือการระบุตำแหน่งที่แน่ชัดที่ช่องโหว่ปรากฏ เช่น URL ของหน้าเว็บ, พารามิเตอร์ (Parameter), หรือส่วนหัวของคำร้องขอ (HTTP Headers ประโยชน์ เพื่อช่วยให้นักพัฒนาสามารถชี้เป้าไปยังบรรทัดของซอร์สโค้ด (Source Code) หรือการตั้งค่าที่ผิดพลาดได้อย่างแม่นยำ

3.2.4.3.4 คำอธิบายโดยสังเขป อธิบายลักษณะการทำงานของช่องโหว่ในเชิงเทคนิค และวิธีการที่ผู้ไม่หวังดีอาจใช้ในการโจมตี (Exploit Scenario) รวมถึงผลกระทบที่อาจเกิดขึ้นกับข้อมูล (CIA Triad: Confidentiality, Integrity, Availability) ประโยชน์ เพื่อให้ทีมงานเข้าใจถึง "ความอันตราย" และบริบทของช่องโหว่นั้น ๆ ในสภาพแวดล้อมของแอปพลิเคชัน

3.2.4.3.5 วิธีการแก้ไขช่องโหว่ ระบุหลักการหรือทฤษฎีในการป้องกันที่ถูกต้องตามมาตรฐาน Secure Coding เช่น การทำ Input Validation, การใช้ Parameterized Queries หรือการตั้งค่า Security Headers ประโยชน์ เพื่อเป็นการให้ "ทางแก้เชิงทฤษฎี" ที่นักพัฒนาสามารถนำไปประยุกต์ใช้กับส่วนอื่นๆ ของระบบที่มีลักษณะใกล้เคียงกันได้

3.2.4.3.6 ตัวอย่างการแก้ไขช่องโหว่ ระบุตัว Code ที่ต้องแก้แบบชัดเจน เพื่อให้นักพัฒนาเห็นภาพชัดเจนว่าต้องเปลี่ยนโค้ดอย่างไร ช่วยลดความผิดพลาดและระยะเวลาในการแก้ไขระบบลงอย่างมีนัยสำคัญ

3.2.5. ขั้นตอนที่ 5 การสแกนซ้ำเพื่อยืนยันผล (Re-scanning and Validation)

3.2.5.1 ภายหลังจากการได้รับการยืนยันว่าการปิดช่องโหว่เสร็จสิ้น ผู้วิจัยจะใช้เครื่องมือตัวเดิมที่ได้รับการคัดเลือกมาทำการสแกนซ้ำ (Re-scanning)

3.2.5.2 ดำเนินการเปรียบเทียบผลลัพธ์แบบ "ก่อนและหลัง" (Before-and-After Analysis) เพื่อพิสูจน์ประสิทธิภาพของการแก้ไข และคำนวณสัดส่วนหรือเปอร์เซ็นต์ของช่องโหว่ที่ได้รับการแก้ไขสำเร็จโดย ผู้วิจัยกำหนดให้ใช้ "เปอร์เซ็นต์การลดลงของช่องโหว่" เป็นตัวชี้วัดหลัก ซึ่งมีสมการที่ใช้ในการประเมินผลดังนี้

เปอร์เซ็นต์การลดลง = [(จำนวนช่องโหว่ก่อนแก้ไข - จำนวนช่องโหว่หลังแก้ไข) / จำนวนช่องโหว่ก่อนแก้ไข] × 100 (3-1)

โดยเกณฑ์การประเมิน จะนำข้อมูลตัวเลขที่ได้จากขั้นตอนที่ 3.2.4.1 และ 3.2.5.1 มาเข้าสู่สมการนี้ โดยหากค่าที่คำนวณได้มีเปอร์เซ็นต์สูง จะเป็นการยืนยันประสิทธิภาพของคู่มือและกระบวนการแก้ไขช่องโหว่ที่ผู้วิจัยจัดทำขึ้น

3.2.6 ขั้นตอนที่ 6 การจัดทำรายงานผลและข้อเสนอแนะ (Reporting and Recommendations) ในขั้นตอนนี้ผู้วิจัยจะดำเนินการรวบรวมและวิเคราะห์ผลการดำเนินงานทั้งหมด เพื่อสรุปเป็นองค์ความรู้และแนวทางปฏิบัติ โดยมีรายละเอียดดังนี้

3.2.6.1 สรุปผลการวิจัยทั้งหมดและจัดทำรายงานฉบับสมบูรณ์ที่แสดงถึงประสิทธิภาพของเครื่องมือและกระบวนการจัดการช่องโหว่

3.2.6.1.1 เป็นการสังเคราะห์ข้อมูลเชิงประจักษ์ (Empirical Data) จากการเปรียบเทียบประสิทธิภาพของเครื่องมือสแกนช่องโหว่ทั้งในมิติด้านความแม่นยำ (Accuracy) ความคุ้มค่า (License) และความสามารถในการนำรายงานไปใช้งานต่อ

3.2.6.1.2 รายงานจะแสดงถึงระดับความสำเร็จของการลดช่องโหว่ (Vulnerability Reduction) โดยเปรียบเทียบสถานะก่อนและหลังการใช้คู่มือการแก้ไขช่องโหว่ (Remediation Manual) เพื่อพิสูจน์ให้เห็นถึงประสิทธิภาพของกระบวนการจัดการช่องโหว่ที่ได้พัฒนาขึ้นในบริบทขององค์กรกรณีศึกษา

3.2.6.2 ให้ข้อเสนอแนะเชิงนโยบายและเทคนิคสำหรับการพัฒนาเว็บแอปพลิเคชันให้ปลอดภัย (Secure Development) เพื่อยกระดับความมั่นคงปลอดภัยขององค์กรอย่างยั่งยืน

3.2.6.2.1 ข้อเสนอแนะเชิงนโยบาย (Policy Recommendations) ผู้วิจัยนำเสนอแนวทางการกำหนดมาตรฐานความมั่นคงปลอดภัยในระดับองค์กร เช่น การกำหนดให้

การสแกนช่องโหว่เป็นส่วนหนึ่งของวงจรการพัฒนาซอฟต์แวร์ (SDLC) และการจัดตั้งกรอบเวลาในการบริหารจัดการช่องโหว่ (Vulnerability Management Lifecycle) ที่ชัดเจนตามระดับความรุนแรง

3.2.6.2.2 ข้อเสนอแนะเชิงเทคนิค (Technical Recommendations) มุ่งเน้นไปที่การสร้างวัฒนธรรมความปลอดภัยในการเขียนโปรแกรม (Secure Coding Practices) โดยเสนอแนะให้มีการบูรณาการเครื่องมือที่ได้รับการคัดเลือกเข้ากับระบบการทำงานอัตโนมัติ (CI/CD Pipeline) เพื่อให้สามารถตรวจพบความเสี่ยงได้ตั้งแต่ระยะเริ่มต้น และการใช้คู่มือการแก้ไขช่องโหว่ที่ผู้วิจัยสร้างขึ้นเป็นฐานข้อมูลความรู้ (Knowledge Base) สำหรับนักพัฒนาเพื่อป้องกันการเกิดช่องโหว่ซ้ำในอนาคต

3.2.6.3 การประเมินคุณภาพและความเหมาะสมของงานวิจัยโดยผู้เชี่ยวชาญหรือผู้ใช้งาน (Expert and User Evaluation) ผู้วิจัยดำเนินการสร้างความน่าเชื่อถือให้กับผลลัพธ์ของงานวิจัยผ่านกระบวนการตรวจสอบและประเมินผลจากมุมมองของบุคคลภายนอกที่มีส่วนเกี่ยวข้อง เพื่อให้มั่นใจว่าองค์ความรู้ที่สร้างขึ้นสามารถนำไปใช้ปฏิบัติงานได้จริง เพื่อให้ทราบถึงประสิทธิภาพของคู่มือที่ผู้วิจัยสร้างขึ้น ผู้วิจัยได้สร้างแบบประเมินความพึงพอใจของผู้เชี่ยวชาญ/ผู้ใช้งาน จำนวน 4 ท่าน มาทำการวิเคราะห์หาค่าเฉลี่ย โดยมีรายละเอียดดังต่อไปนี้

3.2.6.3.1 ความชัดเจนของเนื้อหา (Content Clarity) วัดความสามารถในการสื่อสารข้อมูลทางเทคนิคที่ซับซ้อนให้เข้าใจง่าย โดยเน้นว่าผู้อ่าน (นักพัฒนา) สามารถเข้าใจถึง "ต้นเหตุ (Root Cause)" และ "ผลกระทบ (Impact)" ของช่องโหว่นั้น ๆ ได้อย่างชัดเจน โดยไม่ต้องอาศัยการตีความเพิ่มเติม

3.2.6.3.2 ลำดับขั้นตอนการปฏิบัติ (Practical Steps) วัดความเป็นเหตุเป็นผลและความต่อเนื่องของกระบวนการแก้ไข (Workflow) โดยพิจารณาว่าคู่มือมีการจัดลำดับขั้นตอนแบบ Step-by-Step ที่ชัดเจน ไม่ข้ามขั้นตอนสำคัญ และสามารถนำไปปฏิบัติจริงในสภาพแวดล้อมการทำงานได้โดยไม่ติดขัด

3.2.6.3.3 ความถูกต้องเชิงเทคนิค (Technical Accuracy) วัดความเชื่อถือได้ของเนื้อหา โดยมุ่งเน้นที่ "ตัวอย่างรหัสต้นทาง (Source Code)" และ "การตั้งค่าระบบ (Configuration)" ว่ามีความสอดคล้องกับหลักการ Secure Coding มาตรฐานสากล และสามารถแก้ไขช่องโหว่ได้จริงโดยไม่ส่งผลกระทบเชิงลบต่อการทำงานปกติของแอปพลิเคชัน

3.2.6.3.4 ประสิทธิภาพด้านเวลา (Efficiency & Time-saving) วัดความคุ้มค่าและความสะดวกในการใช้งาน โดยประเมินว่าคู่มือนี้ช่วยลดภาระและระยะเวลาในการค้นคว้าวิธีการแก้ไขจากแหล่งข้อมูลภายนอก (เช่น Stack Overflow หรือ Google) ได้มากน้อยเพียงใด เพื่อช่วยให้นักพัฒนาสามารถ Patch ช่องโหว่ได้รวดเร็วยิ่งขึ้น

3.2.6.3.5 รูปแบบและการนำเสนอ (Presentation & Usability) วัดคุณภาพด้านการออกแบบและการจัดวางองค์ประกอบ (Information Architecture) เช่น การใช้ภาพประกอบที่ตรงประเด็น การจัดกลุ่มเนื้อหาตามรหัสมาตรฐาน (WASC/OWASP) และความง่ายในการค้นหาหัวข้อที่ต้องการ (Searchability) ภายในเล่มคู่มือ

3.2.6.3.6 ประโยชน์และความเหมาะสมต่อองค์กร (Organizational Value) วัดความเป็นไปได้ในการนำไปใช้งานในระดับนโยบาย โดยประเมินว่าคู่มือนี้มีคุณภาพเพียงพอที่จะนำไปใช้เป็น "มาตรฐานการปฏิบัติงาน (Standard Operating Procedure: SOP)" ขององค์กรในระยะยาว เพื่อป้องกันการเกิดช่องโหว่ซ้ำในอนาคต โดยใช้การส่งแบบประเมินให้ทางผู้เกี่ยวข้องตามรูปแบบคำถามดังตารางที่ 3-3

คำชี้แจง โปรดทำเครื่องหมาย ☑ ลงในช่องที่ตรงกับความคิดเห็นของท่านมากที่สุด (5 = มากที่สุด, 4 = มาก, 3 = ปานกลาง, 2 = น้อย, 1 = ควรปรับปรุง)

ตารางที่ 3-3 แบบประเมินคุณภาพคู่มือแนวทางการแก้ไขช่องโหว่

ข้อที่	รายการประเมิน	5	4	3	2	1
1	ความชัดเจนของเนื้อหา เนื้อหาอธิบายสาเหตุและผลกระทบของช่องโหว่ได้เข้าใจง่าย					
2	ขั้นตอนการปฏิบัติ ลำดับขั้นตอนการแก้ไข (Step-by-Step) มีความละเอียดและทำตามได้จริง					
3	ความถูกต้องของวิธีแก้ ตัวอย่างโค้ด (Code Example) หรือการตั้งค่า มีความถูกต้องตามหลักการ					

ตารางที่ 3-3 (ต่อ)

4	การประหยัดเวลา คู่มือนี้ช่วยลดเวลาในการค้นหาวิธีแก้ไขช่องโหว่ด้วยตนเองได้จริง					
5	รูปแบบและการนำเสนอ การจัดวางเนื้อหา ภาพประกอบ และรหัสมาตรฐาน (WASC/OWASP) ช่วยให้ค้นหาข้อมูลง่าย					
6	ประโยชน์ในภาพรวม คู่มือนี้มีความเหมาะสมที่จะนำไปใช้เป็นมาตรฐานการปฏิบัติงานในองค์กร					

เพื่อให้ทราบถึงประสิทธิภาพของคู่มือที่ผู้วิจัยสร้างขึ้น จึงกำหนดให้นำผลคะแนนจากแบบประเมินความพึงพอใจของผู้เชี่ยวชาญ/ผู้ใช้งาน จำนวน 4 ท่าน มาทำการวิเคราะห์หาค่าเฉลี่ยต่อ โดยใช้ ค่าเฉลี่ยเลขคณิตเพื่อหาค่ากลางของระดับความคิดเห็นในแต่ละข้อ โดยคำนวณจากผลรวมของคะแนนทั้งหมดหารด้วยจำนวนผู้ประเมิน ตามสูตร

$$\bar{x} = \frac{\sum x}{n} \quad (3\text{-}2)$$

คำอธิบายสัญลักษณ์

- $\bar{x}$ คือ ค่าเฉลี่ยเลขคณิต
- $\sum x$ คือ ผลรวมของคะแนนทั้งหมดจากผู้ประเมิน
- n คือ จำนวนผู้ประเมินทั้งหมด

และมีเกณฑ์การแปลผลคะแนนดังนี้

- 4.50 – 5.00 เหมาะสม/พึงพอใจในระดับ มากที่สุด
- 3.50 – 4.49 เหมาะสม/พึงพอใจในระดับ มาก
- 2.50 – 3.49 เหมาะสม/พึงพอใจในระดับ ปานกลาง
- 1.50 – 2.49 เหมาะสม/พึงพอใจในระดับ น้อย
- 1.00 – 1.49 เหมาะสม/พึงพอใจในระดับ ควรปรับปรุง

บทที่ 4

ผลการวิจัย

ในการดำเนินงานวิจัยเรื่อง การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข ผู้วิจัยได้ดำเนินการทดสอบและรวบรวมข้อมูลตามระเบียบวิธีวิจัยที่กำหนดไว้ โดยในบทนี้จะนำเสนอผลการวิเคราะห์ข้อมูลเชิงประจักษ์ เพื่อตอบวัตถุประสงค์ในการคัดเลือกเครื่องมือที่เหมาะสมที่สุดและการพิสูจน์ประสิทธิผลของกระบวนการแก้ไขช่องโหว่ โดยมีรายละเอียดตามหัวข้อดังนี้

ตารางที่ 4-1 ผลการเปรียบเทียบเครื่องมือแสกน

คุณสมบัติการทำงาน (Features)	OWASP	Arachni	Burp Suite
1. ด้านการบริหารจัดการและลิขสิทธิ์			
1.1. ใบอนุญาต (License)	☑	☑	⚠
1.2. สถานะการอัปเดต (Maintenance)	☑	✖	☑
2. ด้านความสามารถทางเทคนิค			
2.1. การสแกนอัตโนมัติ (Automated Scan)	☑	☑	✖
2.2. การทดสอบแมนนวล (Manual Tools)	☑	✖	☑
2.3. การรองรับ Modern Web (SPA/AJAX)	☑	⚠	⚠
3. ด้านประสิทธิภาพเชิง			
3.1. ความครอบคลุมของประเภทช่องโหว่	☑	⚠	⚠
3.2. การขยายขีดความสามารถ (Extensibility & Add-ons)	☑	✖	☑
4. ด้านการนำไปใช้งานและการบูรณาการ			
4.1. การเชื่อมต่อ CI/CD (DevSecOps)	☑	⚠	✖
4.2. ความสามารถในการนำข้อมูลรายงานออกมาใช้งาน (Reporting)	☑	⚠	✖
ผลรวม	9	4	4.5

4.1 ผลการเปรียบเทียบและคัดเลือกเครื่องมือสแกนช่องโหว่

หลังจากการได้ข้อสรุปสำหรับโปรแกรมทั้งสามตัวที่ยกมาข้างต้นแล้ว ผู้วิจัยได้ค้นพบว่า Arachni ได้ถูกประกาศให้เป็นซอฟต์แวร์ที่ ล้าสมัย (Obsolete) และเข้าสู่ช่วงหยุดการพัฒนาอย่างเป็นทางการแล้วผู้พัฒนาหลัก Ecsypno LLC ได้พัฒนาโปรเจกต์ใหม่ที่ชื่อว่า Codename SCNR (หรือเรียกสั้นๆ ว่า SCNR) เพื่อมาทดแทน Arachni โดยตรงแต่ยังไม่อยู่ในขั้นตอนการเปิดให้ใช้บริการในปัจจุบัน และในส่วนของ Burp Suite ก็มีการเปลี่ยนแปลงจาก Free Edition เป็น Community Edition และผู้วิจัยได้ทราบภายหลังว่า ไม่มีระบบ Automate scan เนื่องจากโปรแกรมนี้ ถูกใช้ทางฝั่งนักพัฒนาโปรแกรม ซึ่งมีความสามารถในการ Manual scan เองได้ แต่เนื่องจากผู้วิจัยต้องการหาจุดกึ่งกลางของผลการเทียบเครื่องมือเหล่านี้ ผู้วิจัยจึงได้ทำการวิเคราะห์เชิงเปรียบเทียบ โดยอาศัยเกณฑ์การประเมินประสิทธิภาพ 9 ด้านที่ได้กำหนดไว้ในบทที่ 3 โดยการเปรียบเทียบนี้มุ่งเน้นการคัดเลือกเครื่องมือที่มีความสมดุลระหว่างความแม่นยำทางเทคนิคและความคุ้มค่าภายใต้ข้อจำกัดขององค์กร เพื่อให้ได้เครื่องมือหลักที่เป็นมาตรฐาน (Primary Tool) สำหรับการนำไปปฏิบัติงานจริงในขั้นตอนถัดไป ผลการเปรียบเทียบคุณสมบัติการทำงาน (Features) ปรากฏดังตารางที่ 4-1 นี้

✅ = ครบถ้วน ,ดีเยี่ยม ,รองรับ มีค่า 1 คะแนน

⚠️ = จำกัด, ไม่เสถียร มีค่า 0.5 คะแนน

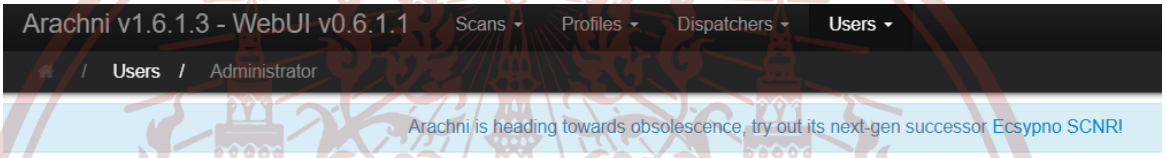
❌ = ไม่มี, หยุดพัฒนา, ไม่รองรับ มีค่า 0 คะแนน

4.1.1 ด้านการบริหารจัดการและลิขสิทธิ์ (Management & Sustainability)

เป็นการประเมินในมุมมองของ "ความคุ้มค่าและความยั่งยืน" ในการใช้งานจริงภายในองค์กร โดยพิจารณาจากข้อตกลงการใช้งาน (License) ที่ไม่สร้างภาระด้านงบประมาณ และความต่อเนื่องในการพัฒนาซอฟต์แวร์ (Maintenance) เพื่อให้มั่นใจว่าเครื่องมือจะสามารถรับมือกับภัยคุกคามในอนาคตได้

4.1.1.1 ใบอนุญาต (License) ผลการเปรียบเทียบพบว่า OWASP ZAP และ Arachni มีความได้เปรียบสูงสุดเนื่องจากเป็นซอฟต์แวร์แบบ Open-Source ที่ไม่มีค่าใช้จ่ายลิขสิทธิ์และเปิดให้ใช้งานทุกฟังก์ชันได้อย่างอิสระ เหมาะสำหรับการใช้งานในองค์กรที่ต้องการควบคุมงบประมาณ ในขณะที่ Burp Suite Community Edition ใช้โมเดลแบบ Freemium ซึ่งมีการจำกัดฟังก์ชันสำคัญหลายประการไว้สำหรับผู้ที่ชำระเงินในรุ่น Professional เท่านั้น ทำให้เกิดข้อจำกัดในการเข้าถึงเทคโนโลยีขั้นสูงในเวอร์ชั่นนี้

4.1.1.2 สถานะการอัปเดต (Maintenance) ผลการเปรียบเทียบพบว่า OWASP ZAP มีสถานะการอัปเดตที่โดดเด่นที่สุดด้วยการสนับสนุนจากชุมชนนักพัฒนาทั่วโลกอย่างต่อเนื่อง เช่นเดียวกับ Burp Suite ที่มีการปรับปรุงซอฟต์แวร์สม่ำเสมอจากทีมพัฒนา PortSwigger ทำให้ทั้งสองเครื่องมือมีฐานข้อมูลช่องโหว่ (CVEs) ที่ทันสมัย แตกต่างจาก Arachni ที่ปัจจุบันอยู่ในสถานะหยุดการพัฒนา (Discontinued) ส่งผลให้ประสิทธิภาพในการตรวจจับภัยคุกคามรูปแบบใหม่ลดลงอย่างมีนัยสำคัญและมีความเสี่ยงต่อความน่าเชื่อถือของข้อมูลดังภาพที่ 4-1 ที่ระบุถึงความล้าสมัยของ Arachni

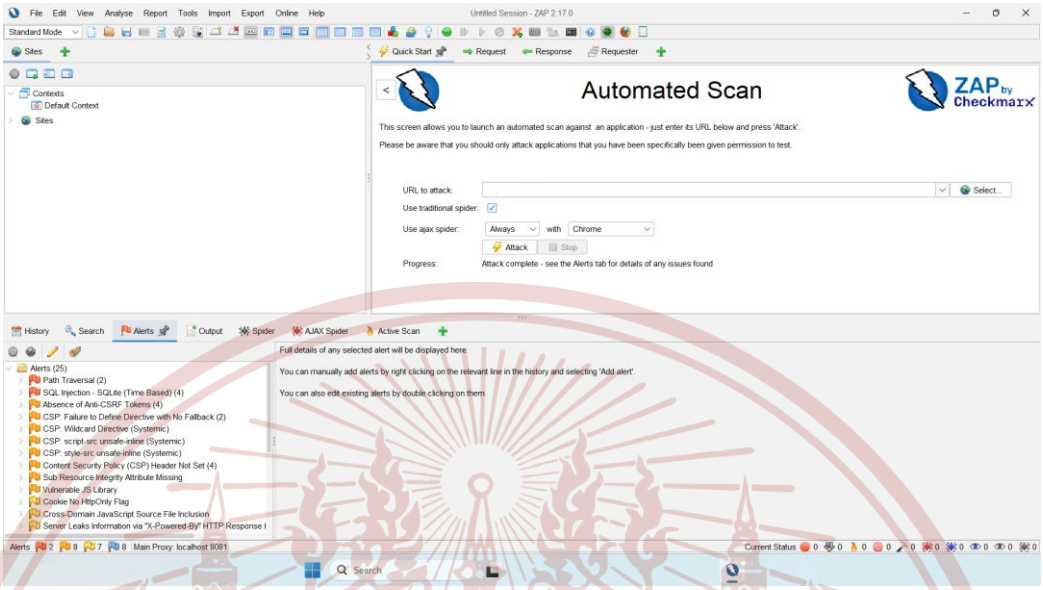


ภาพที่ 4-1 รูปที่ระบุถึงความล้าสมัยของ Arachni

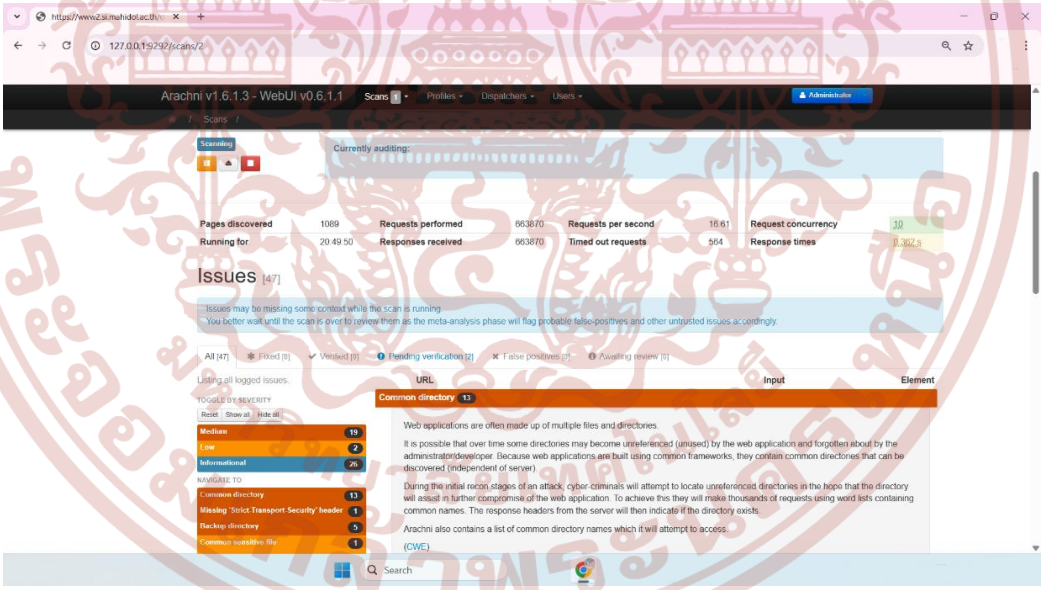
4.1.2 ด้านความสามารถทางเทคนิค (Technical Capabilities)

เป็นการประเมิน "ศักยภาพพื้นฐาน" ของเครื่องมือในการเข้าถึงและตรวจสอบเว็บแอปพลิเคชัน โดยเน้นที่กระบวนการทำงานอัตโนมัติ (Automated Scan) ความยืดหยุ่นในการปรับแต่งด้วยมือ (Manual Tools) และความสามารถในการเข้าใจโครงสร้างเว็บไซต์สมัยใหม่ที่ซับซ้อน (Modern Web Support)

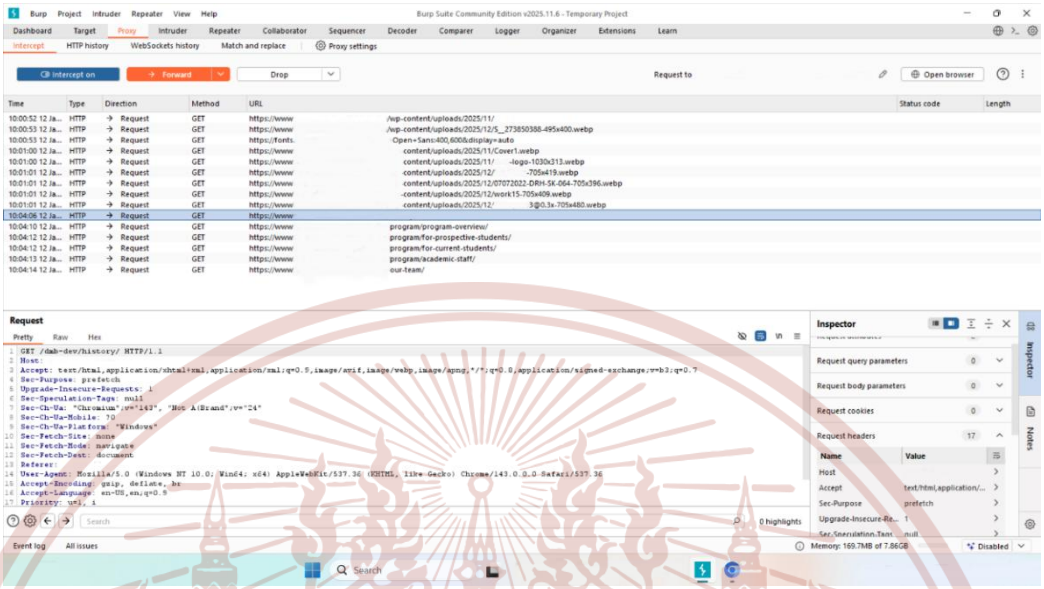
4.1.2.1 การสแกนอัตโนมัติ (Automated Scan) ผลการเปรียบเทียบพบว่า OWASP ZAP และ Arachni สามารถดำเนินการสแกนอัตโนมัติได้สมบูรณ์ครอบคลุมทั้งขั้นตอนการสำรวจโครงสร้าง (Crawling) และการโจมตีเชิงรุก (Active Scan) โดยไม่ต้องอาศัยการสั่งการจากมนุษย์ในทุกขั้นตอน ซึ่งถือเป็นจุดอ่อนที่สำคัญที่สุดของ Burp Suite Community Edition เนื่องจากทางผู้พัฒนาได้ปิดฟังก์ชันการสแกนอัตโนมัติในรุ่นฟรีไว้อย่างชัดเจน ทำให้ไม่สามารถตอบโจทย์การทำงานที่เน้นประสิทธิภาพด้านเวลาและความเร็วได้ ดังภาพที่ 4-2 ภาพที่ 4-3 และภาพที่ 4-4



ภาพที่ 4-2 Owasp ZAP รองรับ Automated Scan



ภาพที่ 4-3 Arachni รองรับ Automated Scan



ภาพที่ 4-4 Burp Suite ไม่รองรับ Automated Scan

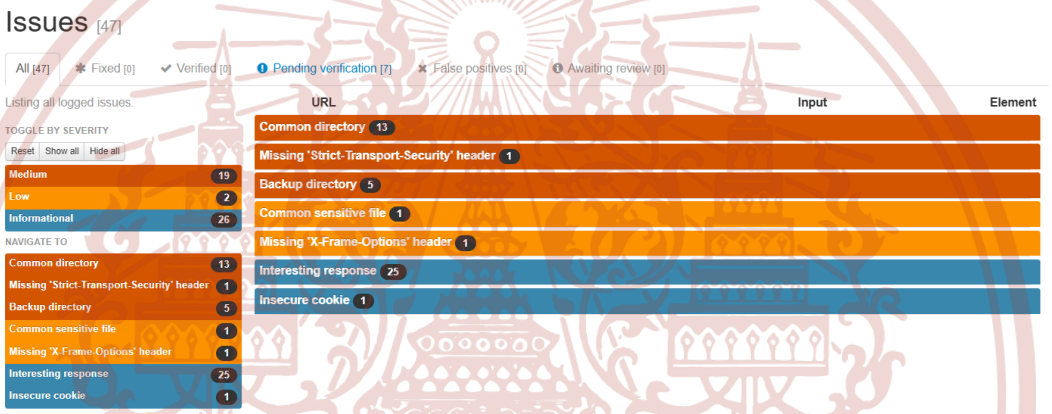
4.1.2.2 การทดสอบแบบแมนนวล (Manual Tools) ผลการเปรียบเทียบพบว่า Burp Suite ยังคงเป็นผู้นำที่เหนือกว่าเครื่องมืออื่นในด้านนี้ ด้วยชุดเครื่องมือทดสอบด้วยมือที่มีความคล่องตัวสูงและเป็นที่ยอมรับในระดับสากล รองลงมาคือ OWASP ZAP ที่มีเครื่องมือเสริมพื้นฐานเพียงพอต่อการตรวจสอบเบื้องต้น ส่วน Arachni แทบไม่มีเครื่องมือสนับสนุนการทดสอบด้วยมือที่ชัดเจน เนื่องจากถูกออกแบบมาเพื่อเน้นการทำงานผ่านบรรทัดคำสั่ง (Command Line) และระบบอัตโนมัติเป็นหลัก

4.1.2.3. การรองรับ Modern Web (SPA/AJAX) ผลการเปรียบเทียบพบว่า OWASP ZAP มีประสิทธิภาพสูงสุดในมิตินี้ด้วยฟังก์ชัน AJAX Spider ที่ช่วยให้สามารถตรวจสอบหน้าเว็บที่ประมวลผลด้วย JavaScript ได้อย่างทั่วถึง แตกต่างจาก Arachni ที่มีข้อจำกัดสูงเนื่องจาก Engine ในการประมวลผล JavaScript ล้าสมัยทำให้การสแกนเว็บสมัยใหม่มักเกิดความผิดพลาด ส่วน Burp Suite (รุ่นฟรี) แม้ตัว Browser Engine จะรองรับได้ดี แต่การขาดระบบสแกนอัตโนมัติทำให้การตรวจสอบเว็บที่มี AJAX จำนวนมากต้องใช้แรงงานคนและใช้เวลานานเกินกว่าที่จะปฏิบัติได้จริง

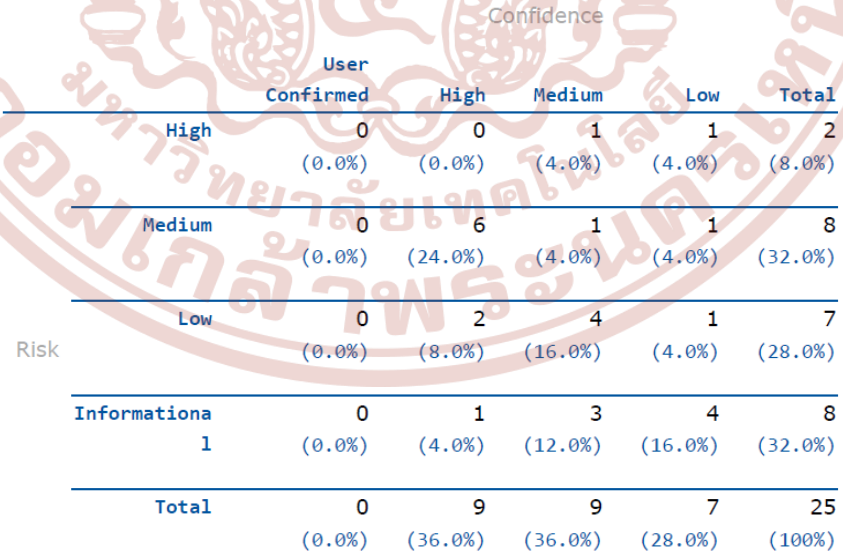
4.1.3 ด้านประสิทธิภาพเชิงลึก (Performance & Quality)

เป็นการประเมิน "คุณภาพของผลลัพธ์" ที่ได้จากการสแกน โดยพิจารณาความแม่นยำเพื่อลดการรายงานที่ผิดพลาด (Accuracy) และความสามารถในการขยายขีดความสามารถ (Extensibility) ผ่านส่วนเสริมต่างๆ เพื่อให้เครื่องมือมีประสิทธิภาพสูงกว่าค่ามาตรฐานทั่วไป

4.1.3.1 ความครอบคลุมของประเภทช่องโหว่ (Vulnerability Coverage Analysis) การเปรียบเทียบความสอดคล้องของผลการแสกน ในส่วนนี้ ผู้วิจัยนำผลลัพธ์จากการสแกนของเครื่องมือทั้ง 2 ชนิดมาวิเคราะห์เปรียบเทียบในมิติของปริมาณ ขอบเขตประเภทช่องโหว่ และระยะเวลาที่ใช้ในการประมวลผล เพื่อประเมินความคุ้มค่าในการใช้งานจริง เนื่องจาก Burp Suite ไม่สามารถ Auto scan และ ออกรีพอร์ตเพื่อนำมาวิเคราะห์ได้ โดยจะต้องอาศัยความชำนาญจากผู้เชี่ยวชาญเฉพาะเท่านั้น ในส่วนนี้จึงได้เปรียบเทียบแค่ระหว่าง OWASP ZAP และ Arachni ดังภาพที่ 4-5 และ 4-6



ภาพที่ 4-5 ภาพผลการแสกนจาก Arachni



Risk	User Confirmed	High	Medium	Low	Total
High	0 (0.0%)	0 (0.0%)	1 (4.0%)	1 (4.0%)	2 (8.0%)
Medium	0 (0.0%)	6 (24.0%)	1 (4.0%)	1 (4.0%)	8 (32.0%)
Low	0 (0.0%)	2 (8.0%)	4 (16.0%)	1 (4.0%)	7 (28.0%)
Informational	0 (0.0%)	1 (4.0%)	3 (12.0%)	4 (16.0%)	8 (32.0%)
Total	0 (0.0%)	9 (36.0%)	9 (36.0%)	7 (28.0%)	25 (100%)

Confidence

ภาพที่ 4-6 ภาพผลการแสดงจาก Owasp Zap

4.1.3.1.1 การเปรียบเทียบเชิงปริมาณและระยะเวลา (Quantitative & Time Analysis) จากการทดสอบสแกนระบบเป้าหมายเดียวกันภายใต้เงื่อนไขมาตรฐาน พบความแตกต่างด้านประสิทธิภาพดังตารางที่ 4-2

ตารางที่ 4-2 การเปรียบเทียบเชิงปริมาณและระยะเวลา

ตัวชี้วัดประสิทธิภาพ	OWASP ZAP	Arachni
จำนวนรายการแจ้งเตือนทั้งหมด (Total Alerts)	25 รายการ	47 รายการ
ระยะเวลาที่ใช้ในการสแกน (Scan Duration)	1 ชั่วโมง	<1 วัน (24 ชั่วโมง)
อัตราการตรวจพบต่อชั่วโมง (Alerts per Hour)	25	1.95

4.1.3.1.2 การวิเคราะห์ความครอบคลุมตามมาตรฐาน OWASP Top 10 (Coverage Mapping) ผู้วิจัยได้นำรายการช่องโหว่ที่ตรวจพบมาจัดหมวดหมู่ตามมาตรฐานสากล เพื่อดูศักยภาพในการตรวจพบช่องโหว่ที่มีความสำคัญ ดังตารางที่ 4-3

ตารางที่ 4-3 การวิเคราะห์ความครอบคลุมตามมาตรฐาน OWASP Top 10

หมวดหมู่ (OWASP TOP 10)	OWASP ZAP	Arachni
A01: Broken Access Control	☑ (Path Traversal, CSRF)	☑ (Insecure Cookie)
A03: Injection	☑ (SQL Injection - Time Based)	✗ (not found)
A05: Security Misconfiguration	☑ (CSP, HSTS, Headers)	☑ (HSTS, X-Frame)
A06: Vulnerable Components	☑ (Vulnerable JS Library)	✗ (not found)
Reconnaissance (Information Leak)	☑ (Version Disclosure)	☑ (Directory Brute-force)

4.1.3.1.3 จากการเปรียบเทียบข้อมูลข้างต้น ผู้วิจัยมีข้อสังเกตที่สำคัญต่อประสิทธิภาพของเครื่องมือทั้งสองดังนี้ ความลึกเทียบกับความกว้าง (Depth vs. Breadth) แม้ Arachni จะพบจำนวนรายการรวมสูงถึง 47 รายการ แต่เมื่อพิจารณาในรายละเอียดพบว่าส่วนใหญ่เป็นช่องโหว่ในกลุ่มการสำรวจเส้นทาง (Common Directory/Backup) และการตอบสนองทั่วไป (Interesting Response) ซึ่งเป็นการเน้นการเก็บข้อมูลเชิงรุก (Reconnaissance) ในขณะที่ OWASP ZAP แม้จะพบเพียง 25 รายการ แต่กลับตรวจพบช่องโหว่ระดับวิกฤตที่มีความซับซ้อนสูงกว่า เช่น SQL Injection (Time Based) และ Path Traversal ซึ่งส่งผลกระทบต่อระบบได้รุนแรงกว่า ทางด้าน ประสิทธิภาพด้านเวลา (Time Efficiency) ความแตกต่างด้านเวลามีความชัดเจนอย่างมาก โดย OWASP ZAP ใช้เวลาเพียง 1 ชั่วโมงในการเข้าถึงช่องโหว่สำคัญ ในขณะที่ Arachni ใช้เวลาสูงถึง 1 วัน เนื่องจากการเสียเวลาไปกับกระบวนการคาดเดาชื่อไดเรกทอรี (Directory Brute-forcing) ซึ่งในบริบทขององค์กรที่ต้องการความรวดเร็วในการตรวจสอบ ZAP จึงมีความคุ้มค่ามากกว่าอย่างนัยสำคัญ สุดท้ายคุณภาพของข้อมูลเพื่อการแก้ไข (Actionability) รายการแจ้งเตือนของ OWASP ZAP มีความหลากหลายและครอบคลุมหมวดหมู่ OWASP Top 10 มากกว่า (เช่น พบทั้ง Injection และ Vulnerable Library) ทำให้ผู้วิจัยมีข้อมูลที่เพียงพอในการนำไปจัดทำ "คู่มือการแก้ไขช่องโหว่" ที่ครอบคลุมหลายมิติ มากกว่าการเน้นไปที่การปิดไดเรกทอรีเพียงอย่างเดียวเหมือนผลลัพธ์จาก Arachni

4.1.3.2 การขยายขีดความสามารถ (Extensibility & Add-ons) ผลการเปรียบเทียบพบว่า OWASP ZAP และ Burp Suite มีระบบ Marketplace (Add-ons / BApp Store) ที่แข็งแกร่งและมีชุมชนสนับสนุนการพัฒนาปลั๊กอินอย่างสม่ำเสมอ ช่วยให้ผู้ใช้สามารถเพิ่มขีดความสามารถเฉพาะทางได้ทันที ซึ่งแตกต่างอย่างสิ้นเชิงกับ Arachni ที่มีโครงสร้างซอฟต์แวร์แบบปิดกว่าและไม่มีส่วนเสริมใหม่ๆ รองรับ ทำให้ขาดความยืดหยุ่นในการปรับแต่งเพื่อรับมือกับสภาพแวดล้อมที่เฉพาะเจาะจง

4.1.4. ด้านการนำไปใช้งานและการบูรณาการ (Integration & Reporting)

เป็นการประเมิน "ความสะดวกในการนำข้อมูลไปใช้ต่อ" โดยเน้นที่ความสามารถในการเชื่อมต่อกับระบบการพัฒนาซอฟต์แวร์แบบอัตโนมัติ (DevSecOps) และคุณภาพของรายงาน (Reporting) ที่ต้องมีความละเอียดและหลากหลายรูปแบบ เพื่อให้ทีมพัฒนาสามารถนำไปใช้เป็นคู่มือในการแก้ไขช่องโหว่ได้อย่างมีประสิทธิภาพ

4.1.4.1 การเชื่อมต่อ CI/CD (DevSecOps) ผลการเปรียบเทียบพบว่า OWASP ZAP ได้รับการออกแบบมาเพื่อรองรับ API-driven และ Command Line อย่างสมบูรณ์ ทำให้การบูรณาการเข้ากับกระบวนการ DevSecOps ทำได้ง่ายและไม่มีค่าใช้จ่ายลิขสิทธิ์เพิ่มเติม แตกต่างจาก Burp Suite (รุ่นฟรี) ที่ถูกจำกัดไม่ให้เชื่อมต่อกับระบบอัตโนมัติภายนอกได้หากไม่ขยับไปใช้รุ่น Enterprise ส่วน Arachni แม้จะรองรับ CLI แต่ด้วยสถานะการหยุดพัฒนาทำให้ความมั่นคงและความเข้ากันได้กับระบบ Pipeline สมัยใหม่มีจำกัด

4.1.4.2 ความสามารถในการนำข้อมูลรายงานออกมาใช้งาน (Reporting) (ปัจจัยตัดสินสำคัญ) ผลการเปรียบเทียบพบว่า OWASP ZAP เป็นเครื่องมือเดียวที่ให้เสรีภาพในการส่งออกรายงานได้หลากหลายรูปแบบ (HTML, XML, JSON) และมีความละเอียดสูง ซึ่งเป็นปัจจัยตัดสินสำคัญที่ช่วยให้ผู้วิจัยนำข้อมูลไปวิเคราะห์ต่อได้ทันที ดังภาพที่ 4-7 ในขณะที่ Burp Suite (รุ่นฟรี) ปิดฟังก์ชันการส่งออกรายงานไว้อย่างชัดเจนทำให้เกิดอุปสรรคในการรวบรวมข้อมูล และ Arachni แม้จะรองรับการออกรายงานแต่ความเสถียรของรูปแบบรายงานมักพบปัญหาไฟล์แสดงผลผิดพลาดบ่อยครั้ง



ภาพที่ 4-7 ตัวอย่าง Report ในรูปแบบ PDF จาก Owasp ZAP

โดยสรุปจากการวิเคราะห์เชิงเปรียบเทียบในทั้ง 9 หัวข้อ ผู้วิจัยพบว่า OWASP ZAP มีคะแนนประเมินอยู่ในเกณฑ์ 'ดีเยี่ยม' ในทุกมิติของเกณฑ์ที่ตั้งไว้ และสามารถตอบโจทย์การวิจัยภายใต้ข้อจำกัดขององค์กรได้ครบถ้วนที่สุด โดยเฉพาะด้านความแม่นยำและการนำข้อมูลรายงานไป

ใช้งานต่อ ดังนั้น ผู้วิจัยจึงตัดสินใจคัดเลือก OWASP ZAP เป็นเครื่องมือหลักในการดำเนินงานวิจัยในขั้นตอนถัดไป

4.2 ผลการตรวจพบช่องโหว่สถานะฐาน (Initial Baseline Results)

จากดำเนินการคัดเลือกเครื่องมือที่เหมาะสมที่สุดซึ่งได้แก่ Owasp ZAP นั้นผู้วิจัยได้นำเครื่องมือดังกล่าวมาดำเนินการสแกนเว็บแอปพลิเคชันเป้าหมายใหม่ (https://devxxxxxx.xx.xxxxxxx.xx.xx/xxx_xx) อย่างละเอียด เพื่อกำหนดค่าดัชนีชี้วัดพื้นฐาน (Security Baseline) ผลการวิจัยในส่วนนี้จะสะท้อนถึงสถานะความปลอดภัยในปัจจุบันของระบบก่อนได้รับการปรับปรุง โดยจำแนกตามระดับความรุนแรงของช่องโหว่และพารามิเตอร์ที่ได้รับผลกระทบ ซึ่งข้อมูลในส่วนนี้จะถูกใช้เป็นจุดอ้างอิงสำคัญในการประเมินความสำเร็จของกระบวนการแก้ไขช่องโหว่ในลำดับถัดไป

4.2.1. Report ผลการแสกน ที่สถิติจำนวนช่องโหว่จากการแสกนก่อนการแก้ไข (Vulnerability Statistics Summary) ดังภาพที่ 4-8

Confidence

Risk	User Confirmed	High	Medium	Low	Total
High	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (5.9%)	1 (5.9%)
Medium	0 (0.0%)	2 (11.8%)	3 (17.6%)	0 (0.0%)	5 (29.4%)
Low	0 (0.0%)	2 (11.8%)	4 (23.5%)	0 (0.0%)	6 (35.3%)
Informational	0 (0.0%)	0 (0.0%)	4 (23.5%)	1 (5.9%)	5 (29.4%)
Total	0 (0.0%)	4 (23.5%)	11 (64.7%)	2 (11.8%)	17 (100%)

ภาพที่ 4-8 ภาพ Report จาก Owasp ZAP

4.2.2. การวิเคราะห์สถานะความปลอดภัยเบื้องต้น (Baseline Security Insight)

4.2.2.1. สรุปจำนวนช่องโหว่ที่ตรวจพบดังตารางที่ 4-4

ตารางที่ 4-4 แสดงสถิติจำนวนช่องโหว่สถานะฐานจำแนกตามระดับความรุนแรง

ระดับความเสี่ยง	จำนวน	คำอธิบายโดยสังเขป
สูง (High / Critical)	1	ช่องโหว่ที่สามารถถูกใช้โจมตีเพื่อเข้าควบคุมระบบ ขโมยข้อมูล หรือทำให้ระบบหยุดทำงานได้ ควรแก้ไขเร่งด่วนที่สุด
ปานกลาง (Medium)	5	ช่องโหว่ที่อาจนำไปสู่การรั่วไหลของข้อมูลหรือถูกใช้เป็นทางผ่านในการโจมตีอื่นๆ ควรวางแผนแก้ไขในระยะเวลาที่กำหนด
ต่ำ (Low)	6	ช่องโหว่เชิงเทคนิคหรือการตั้งค่าที่ไม่รัดกุม ควรแก้ไขเพื่อยกระดับความปลอดภัยเชิงลึก (Defense-in-Depth)
แจ้งเพื่อทราบ (Informational)	5	ข้อสังเกตด้านสถาปัตยกรรมหรือพฤติกรรมของระบบ ไม่ใช่ช่องโหว่โดยตรง แต่ใช้ประกอบการประเมินภาพรวมได้
รวมทั้งหมด (Total)	17	

4.2.2.2. รายละเอียดช่องโหว่ที่ตรวจพบดังตารางที่ 4-5 และ ตารางที่ 4-6

ตารางที่ 4-5 รายละเอียดช่องโหว่ที่ตรวจพบจากรีพอร์ตของ OWASP ZAP

ลำดับ	รายการแจ้งเตือนช่องโหว่	Risk	Confidence	แท็กอ้างอิง	
				WASC ID	CWE ID
1	SQL Injection	High	Low	19	89
2	Application Error Disclosure	Medium	High	13	550
3	Buffer Overflow	Medium	High	7	120
4	Content Security Policy (CSP) Header Not Set	Medium	High	15	693
5	Sub Resource Integrity Attribute Missing	Medium	High	15	345
6	Vulnerable JS Library	Medium	High	-	1395
7	Cookie Without Secure Flag	Low	High	13	614
8	HTTPS Content Available via HTTP	Low	High	4	311

ตารางที่ 4-5 (ต่อ)

9	Server Leaks Information via "X-Powered-By" HTTP ResponseHeader Field(s)	Low	Medium	13	497
10	Server Leaks Version Information via "Server" HTTP ResponseHeader Field	Low	Medium	13	497
11	Strict-Transport-Security Header Not Set	Low	Medium	15	319
12	X-Content-Type-Options Header Missing	Low	Medium	15	693
13	Information Disclosure - Suspicious Comments	Informational	Medium	13	615
14	Modern Web Application	Informational	Medium	-	-
15	Session Management Response Identified	Informational	Medium	-	-
16	User Agent Fuzzer	Informational	Medium	-	-
17	User Controllable HTML Element Attribute (Potential XSS)	Informational	Medium	20	20

ตารางที่ 4-6 รายละเอียดช่องโหว่ที่ตรวจพบเทียบกับ Owasp Top 10 2021

ลำดับ	รายการแจ้งเตือนช่องโหว่	แท็กอ้างอิง	
		Owasp Top 10 2021	
1	SQL Injection	A03	Injection
2	Application Error Disclosure	A05	Security Misconfiguration
3	Buffer Overflow	A03	Injection
4	Content Security Policy (CSP) Header Not Set	A05	Security Misconfiguration

ตารางที่ 4-6 (ต่อ)

5	Sub Resource Integrity Attribute Missing	A05	Security Misconfiguration
6	Vulnerable JS Library	A06	Vulnerable and Outdated Components
7	Cookie Without Secure Flag	A05	Security Misconfiguration
8	HTTPS Content Available via HTTP	A02	Cryptographic Failures
9	Server Leaks Information via "X-Powered-By" HTTP ResponseHeader Field(s)	A05	Security Misconfiguration
10	Server Leaks Version Information via "Server" HTTP ResponseHeader Field	A05	Security Misconfiguration
11	Strict-Transport-Security Header Not Set	A05	Security Misconfiguration
12	X-Content-Type-Options Header Missing	A05	Security Misconfiguration
13	Information Disclosure - Suspicious Comments	A05	Security Misconfiguration
14	Modern Web Application	-	-
15	Session Management Response Identified	A07	Identification and Authentication Failures
16	User Agent Fuzzer	-	-
17	User Controllable HTML Element Attribute (Potential XSS)	A03	Injection

จากการสรุปสถิติข้างต้น ผู้วิจัยมีข้อสังเกตที่สำคัญเกี่ยวกับความมั่นคงปลอดภัยของเว็บแอปพลิเคชันเป้าหมายในสภาวะเริ่มต้น ดังนี้

4.2.2.1 การเผชิญกับภัยคุกคามระดับวิกฤต แม้จะตรวจพบช่องโหว่ระดับสูง (High) เพียง 1 รายการ คือ SQL Injection แต่ถือเป็นความเสี่ยงที่มีนัยสำคัญสูงสุด เนื่องจากเป็นช่องทางที่

ผู้ไม่หวังดีสามารถใช้เจาะเข้าถึงฐานข้อมูลเพื่อขโมยหรือเปลี่ยนแปลงข้อมูลได้โดยตรง เมื่อรวมกับช่องโหว่ระดับปานกลาง (Medium) อีก 5 รายการ เช่น Buffer Overflow และ Vulnerable JS Library สะท้อนให้เห็นว่าระบบมีจุดอ่อนในระดับโครงสร้างซอฟต์แวร์ที่อาจถูกใช้โจมตีเพื่อเข้าควบคุมระบบได้

4.2.2.2 การขาดการกำกับดูแลด้านการตั้งค่าความปลอดภัย (Security Misconfiguration) พบช่องโหว่ในระดับปานกลางและระดับต่ำรวมกันถึง 11 รายการ ซึ่งส่วนใหญ่เกี่ยวข้องกับการขาดนโยบายความปลอดภัยของระบบเครือข่ายและเว็บเซิร์ฟเวอร์ (เช่น CSP, HSTS, และ Security Headers ต่างๆ) ข้อมูลนี้ระบุชัดเจนว่าระบบยังไม่ได้ผ่านกระบวนการปรับแต่งค่าความปลอดภัย (Hardening) ตามมาตรฐานสากล ซึ่งส่งผลให้ผู้โจมตีสามารถรวบรวมข้อมูลของระบบ (Information Leakage) เพื่อนำไปใช้เตรียมการโจมตีในระดับที่สูงขึ้นได้ง่าย

4.2.2.3 การกำหนดดัชนีชี้วัดเพื่อการพัฒนา (Baseline for Improvement) ตัวเลขช่องโหว่รวมทั้งสิ้น 17 รายการนี้ จะถูกใช้เป็น "ค่าดัชนีชี้วัดพื้นฐาน (Baseline Metric)" เพื่อนำไปเปรียบเทียบกับผลการสแกนภายหลังการปรับปรุงในหัวข้อ 4.4 โดยผู้วิจัยจะมุ่งเน้นการแก้ไขช่องโหว่ในกลุ่ม High และ Medium เป็นลำดับแรก เพื่อให้บรรลุเป้าหมายในการลดอัตราความเสี่ยง (Reduction Rate) ให้ได้ตามเกณฑ์ที่กำหนดไว้ในวัตถุประสงค์ของการวิจัย

4.3 ผลการวิเคราะห์สาเหตุและแนวทางการแก้ไข (Remediation Analysis)

การวิจัยในขั้นตอนถัดมามุ่งเน้นการเปลี่ยนข้อมูลเชิงเทคนิคจากรายงานการสแกน ให้กลายเป็นแนวทางปฏิบัติที่เป็นรูปธรรม ผู้วิจัยได้ดำเนินการวิเคราะห์สาเหตุเชิงลึก ของแต่ละช่องโหว่ และจัดทำเป็นคู่มือแนวทางการแก้ไขช่องโหว่แสดงดัง ภาคผนวก ก โดยครอบคลุมทั้งหลักการทางทฤษฎีและตัวอย่างการปรับปรุงซอร์สโค้ด โดยมีเป้าหมายเพื่อสื่อสารองค์ความรู้ด้านความปลอดภัยไปยังทีมพัฒนาซอฟต์แวร์ได้อย่างแม่นยำและลดข้อผิดพลาดในการแก้ไขระบบ โดยอ้างอิงรายละเอียดตามตารางที่ 3-2 ขอบเขตองค์ประกอบของรายงานการแก้ไขช่องโหว่

4.4 ผลการตรวจสอบยืนยันการแก้ไข (Validation and Re-scan Results)

ผู้วิจัยได้ดำเนินการสแกนซ้ำภายหลังการปรับปรุงระบบ (Re-scanning) เพื่อประเมินประสิทธิภาพของคู่มือแนวทางการแก้ไข โดยมีรายละเอียดการเปรียบเทียบผลลัพธ์ดังนี้

4.4.1 การเปรียบเทียบจำนวนช่องโหว่ก่อนและหลังการปรับปรุงระบบ

ผลการสแกนพบว่าช่องโหว่ที่มีระดับความรุนแรงสูงและปานกลางถูกกำจัดออกไปทั้งหมด ในขณะที่ช่องโหว่ระดับต่ำและแจ้งเพื่อทราบยังคงเหลืออยู่บางส่วนเนื่องจากข้อจำกัดด้านโครงสร้างพื้นฐาน ดังแสดงในตารางต่อไปนี้

ตารางที่ 4-7 ตารางเปรียบเทียบจำนวนช่องโหว่ก่อนและหลังการปรับปรุงระบบ

ระดับความเสี่ยง (Severity)	จำนวนก่อนแก้ไข (Baseline)	จำนวนหลังแก้ไข (Post-Remediation)	ผลลัพธ์ (Result)
สูง (High / Critical)	1	0	ลดลง 100%
ปานกลาง (Medium)	5	0	ลดลง 100%
ต่ำ (Low)	6	5	คงเดิม
แจ้งเพื่อทราบ (Informational)	5	4	ลดลง 20%
รวมทั้งหมด (Total)	17	9	ลดลง 8 รายการ

4.4.2 การคำนวณอัตราการลดลงของช่องโหว่ (Vulnerability Reduction Rate)

ผู้วิจัยประเมินความสำเร็จเชิงปริมาณโดยใช้สมการการการคำนวณเปอร์เซ็นต์การลดลงดังที่กำหนดไว้ข้างต้น ได้ผลลัพธ์เท่ากับ มีการลดลงอยู่ที่ 47.06% ข้อสังเกต แม้อัตราการลดลงในภาพรวมจะอยู่ที่ 47.06% แต่หากพิจารณาเฉพาะช่องโหว่ที่มีผลกระทบต่อระบบอย่างรุนแรง (High & Medium) พบว่ามีอัตราการลดลงถึง 100% ซึ่งถือว่าบรรลุเป้าหมายหลักในการป้องกันการโจมตีที่สำคัญ

4.4.3 การวิเคราะห์สาเหตุของช่องโหว่คงเหลือ

ผู้วิจัยได้จำแนกสาเหตุที่ช่องโหว่จำนวน 9 รายการยังคงเหลืออยู่ภายหลังการสแกนซ้ำเพื่อเป็นข้อมูลสำหรับการบริหารจัดการความเสี่ยงขององค์กรในอนาคต ดังนี้

4.4.3.1 กลุ่มช่องโหว่ระดับต่ำ (Low Risk) - รวม 5 รายการ

4.4.3.1.1 รายการที่พบ Cookie Without Secure Flag, Strict-Transport-Security Header Not Set, X-Content-Type-Options Missing, HTTPS Content Available via HTTP, Server Version Information Leak.

4.4.3.1.2. เหตุผลทางเทคนิค ช่องโหว่กลุ่มนี้เกี่ยวข้องกับการตั้งค่าในระดับโครงสร้างพื้นฐานและเว็บเซิร์ฟเวอร์ เช่น การตั้งค่าในไฟล์ nginx.conf หรือ httpd.conf ซึ่งอยู่นอกเหนือขอบเขตการแก้ไขในระดับซอร์สโค้ด ที่ผู้วิจัยได้รับสิทธิ์ในการดำเนินการ

4.4.3.2 กลุ่มการแจ้งเพื่อทราบ (Informational) - รวม 4 รายการ

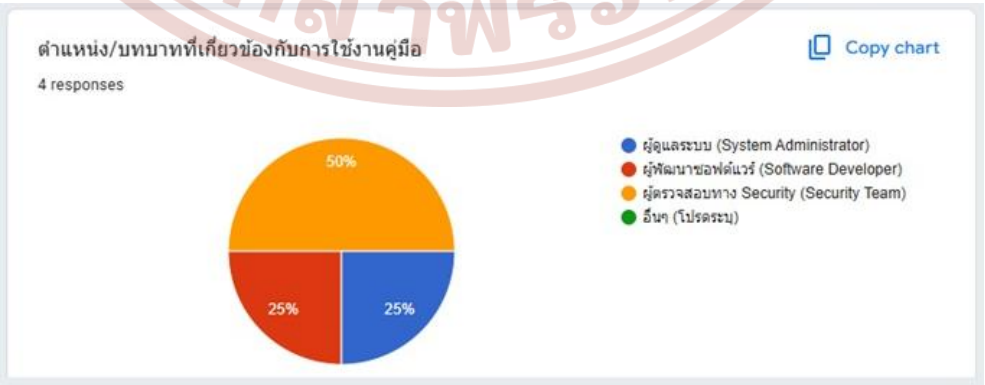
4.4.3.2.1 รายการที่พบ Information Disclosure - Suspicious Comments, Modern Web Application, Session Management Response, User Agent Fuzzer.

4.4.3.2.2 เหตุผลทางเทคนิค เป็นข้อมูลเชิงพฤติกรรมของระบบที่องค์กรตัดสินใจยอมรับความเสี่ยง (Risk Acceptance) เนื่องจากเป็นคุณลักษณะที่จำเป็นในการทำงานของเว็บแอปพลิเคชันสมัยใหม่ และไม่ได้ส่งผลกระทบโดยตรงต่อความมั่นคงปลอดภัยของข้อมูล

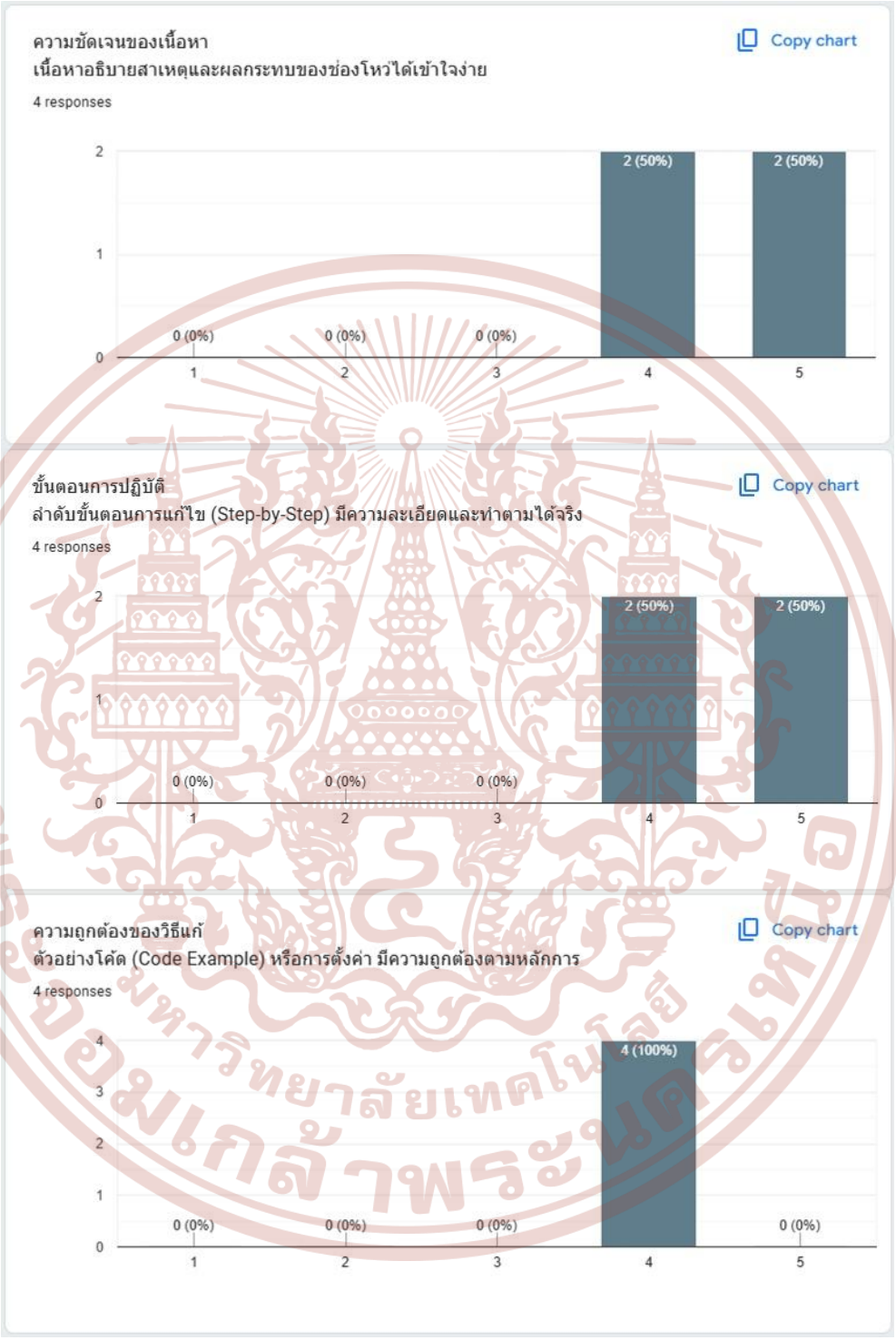
ผลการตรวจสอบยืนยันการแก้ไขนี้แสดงให้เห็นว่า คู่มือแนวทางการแก้ไขที่ผู้วิจัยจัดทำขึ้นมีประสิทธิภาพสูงในการกำจัดความเสี่ยงในระดับวิกฤตและระดับปานกลางได้ครบถ้วน สำหรับช่องโหว่ที่คงเหลือนั้นเป็นความเสี่ยงในระดับที่ยอมรับได้ และต้องอาศัยการประสานงานร่วมกับหน่วยงานที่ดูแลโครงสร้างพื้นฐาน เพื่อยกระดับความปลอดภัยในลำดับต่อไป

4.4.4 ผลการประเมินคุณภาพและความเหมาะสมของคู่มือแนวทางการแก้ไขช่องโหว่

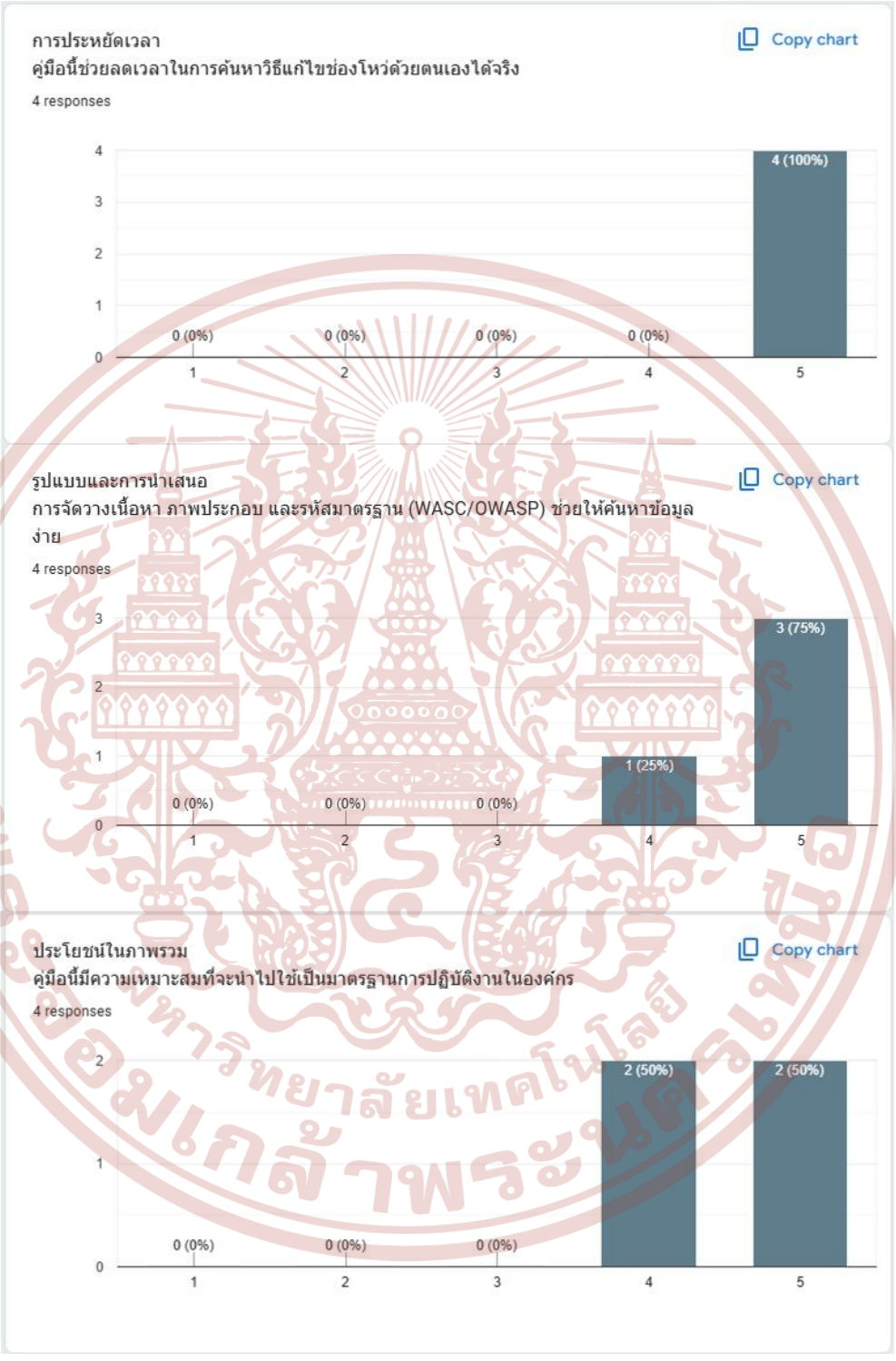
เพื่อสร้างความน่าเชื่อถือให้กับผลลัพธ์ของงานวิจัยผ่านกระบวนการตรวจสอบและประเมินผลจากมุมมองของบุคคลที่มีส่วนเกี่ยวข้อง เพื่อให้มั่นใจว่าสามารถนำไปใช้ปฏิบัติงานได้จริง และทราบถึงประสิทธิภาพของคู่มือที่สร้างขึ้น ผู้วิจัยได้นำผลคะแนนจากแบบประเมินความพึงพอใจของผู้เชี่ยวชาญ/ผู้ใช้งาน จำนวน 4 ท่าน มาทำการวิเคราะห์หาค่าเฉลี่ย ดังนี้



ภาพที่ 4-9 จำนวนผู้ประเมิณ 4 ท่านจากตำแหน่งงานต่าง ๆ



ภาพที่ 4-10 ผลการประเมิณข้อที่ 1-3



ภาพที่ 4-11 ผลการประเมิณข้อที่ 4-6

จากการนำคู่มือไปให้ผู้เชี่ยวชาญและผู้ใช้งานจำนวน 4 ท่านประเมินคุณภาพ ผู้วิจัยได้นำผลคะแนนมาคำนวณหาค่าเฉลี่ยที่กำหนดไว้ เพื่อสรุปความพึงพอใจในภาพรวม ผลได้ดังนี้

ตารางที่ 4-8 ผลคะแนนเฉลี่ย

ข้อที่	รายการประเมิน	ค่าเฉลี่ย ($\bar{x}$)	ระดับความคิดเห็น
1	ความชัดเจนของเนื้อหา	4.5	มากที่สุด
2	ลำดับขั้นตอนการปฏิบัติ (Step-by-Step)	4.5	มากที่สุด
3	ความถูกต้องของตัวอย่างโค้ดและวิธีแก้	4	มาก
4	ความสามารถในการลดเวลาการทำงาน	5	มากที่สุด
5	รูปแบบการนำเสนอและการค้นหาข้อมูล	4.75	มากที่สุด
6	ประโยชน์ในภาพรวมต่อองค์กร	4.5	มากที่สุด
	รวมเฉลี่ย	4.54	มากที่สุด

4.4.4.1 ความชัดเจนของเนื้อหา ($\bar{x}$ = 4.50) อยู่ในระดับ มากที่สุด แสดงให้เห็นว่าการเรียบเรียงเนื้อหาในคู่มือมีการใช้ภาษาที่สื่อสารได้อย่างตรงไปตรงมา สามารถอธิบายต้นเหตุของปัญหา (Root Cause) และผลกระทบที่อาจเกิดขึ้นจากช่องโหว่ได้อย่างชัดเจน ช่วยให้ผู้ใช้งานที่มีพื้นฐานความรู้ด้านไอทีที่แตกต่างกันสามารถทำความเข้าใจเนื้อหาได้โดยง่าย

4.4.4.2 ลำดับขั้นตอนการปฏิบัติ ($\bar{x}$ = 4.50) อยู่ในระดับ มากที่สุด สะท้อนถึงคุณภาพของการจัดลำดับขั้นตอนการทำงาน (Workflow) ที่มีความต่อเนื่องและเป็นเหตุเป็นผล ผู้วิจัยมีการจัดทำขั้นตอนแบบทีละขั้น (Step-by-Step) ซึ่งช่วยลดความสับสนในการปฏิบัติงาน และทำให้ผู้อ่านสามารถดำเนินการแก้ไขตามคู่มือได้อย่างเป็นระบบ

4.4.4.3. ความถูกต้องของตัวอย่างโค้ดและวิธีแก้ ($\bar{x}$ = 4.00) อยู่ในระดับ มาก แม้จะมีค่าเฉลี่ยต่ำสุดเมื่อเทียบกับข้ออื่น แต่ยังคงอยู่ในเกณฑ์ระดับที่ดีมาก ซึ่งผลการประเมินนี้ระบุว่าตัวอย่างรหัสต้นทาง (Source Code) และการตั้งค่าระบบที่ผู้วิจัยนำเสนอ มีความถูกต้องตาม

หลักการ Secure Coding อย่างไรก็ตาม ผู้ประเมินบางท่านมีความเห็นว่าตัวอย่างโค้ดบางส่วนมีความเฉพาะเจาะจงกับโครงสร้างระบบที่ใช้ในการทดลอง ซึ่งอาจต้องมีการปรับแก้เล็กน้อยหากนำไปใช้กับภาษาโปรแกรมอื่น

4.4.4.4. ความสามารถในการลดเวลาการทำงาน ($\bar{x}$ = 5.00) อยู่ในระดับ มากที่สุด และเป็นรายการที่ได้รับคะแนนเต็มจากผู้ประเมินทุกคน ยืนยันว่าคู่มือฉบับนี้เป็นเครื่องมือที่ช่วยเพิ่มประสิทธิภาพในการทำงานได้อย่างมีนัยสำคัญ โดยช่วยให้นักพัฒนาและเจ้าหน้าที่ไอทีลดเวลาในการค้นหาวิธีการแก้ไขจากแหล่งข้อมูลภายนอก และสามารถจัดการกับช่องโหว่ได้ทันที จากแนวทางที่ผู้วิจัยสรุปไว้ให้

4.4.4.5. รูปแบบการนำเสนอและการค้นหาข้อมูล ($\bar{x}$ = 4.75) อยู่ในระดับ มากที่สุด แสดงว่าการออกแบบรูปเล่มคู่มือมีการจัดวางองค์ประกอบที่ดี มีการใช้ภาพประกอบจากซอฟต์แวร์จริงประกอบกับการอ้างอิงรหัสมาตรฐาน OWASP/WASC ที่ชัดเจน ทำให้ผู้ใช้งานสามารถสืบค้นหัวข้อที่ต้องการได้อย่างสะดวกและรวดเร็ว

4.4.4.6. ประโยชน์ในภาพรวมต่อองค์กร ($\bar{x}$ = 4.50) อยู่ในระดับ มากที่สุด ชี้ให้เห็นว่าคู่มือนี้มีคุณค่าเพียงพอที่จะนำไปประยุกต์ใช้เป็นมาตรฐานการปฏิบัติงาน (SOP) ภายในองค์กร เพื่อยกระดับความมั่นคงปลอดภัยของระบบในระยะยาว และใช้เป็นฐานข้อมูลความรู้ (Knowledge Base) สำหรับฝึกอบรมบุคลากรในทีมพัฒนาซอฟต์แวร์ต่อไป

บทสรุปภาพรวม

ในภาพรวม ผลการประเมินคุณภาพคู่มือแนวทางการแก้ไขช่องโหว่มีค่าเฉลี่ยรวมเท่ากับ 4.54 ซึ่งจัดอยู่ในระดับ "มากที่สุด" สถิตินี้เป็นเครื่องยืนยันว่างานค้นคว้าอิสระฉบับนี้ ประสบความสำเร็จในการสร้างองค์ความรู้ที่ใช้งานได้จริง (Practical Knowledge) โดยคู่มือที่พัฒนาขึ้นไม่เพียงแต่มีความถูกต้องเชิงเทคนิคเท่านั้น แต่ยังมีความโดดเด่นในด้านการเพิ่มประสิทธิภาพการทำงาน (Efficiency) และความง่ายในการนำไปปฏิบัติ ซึ่งเป็นหัวใจสำคัญในการจัดการช่องโหว่ของเว็บแอปพลิเคชันในสภาวะการทำงานจริง

4.5 สรุปผลสัมฤทธิ์ของการดำเนินงานวิจัย

จากการดำเนินงานวิจัยเรื่องการเปรียบเทียบประสิทธิภาพเครื่องมือและการพัฒนาคู่มือการแก้ไขช่องโหว่ ผู้วิจัยสามารถสรุปผลสัมฤทธิ์ตามวัตถุประสงค์ที่ตั้งไว้ใน 3 มิติหลัก ดังนี้

4.5.1 ความสำเร็จด้านประสิทธิภาพในการคัดเลือกเครื่องมือ (Efficiency & Tool Selection) ผู้วิจัยพิสูจน์ให้เห็นว่า OWASP ZAP เป็นเครื่องมือที่มีความเหมาะสมที่สุดสำหรับการใช้งานในองค์กรกรณีศึกษา โดยมีดัชนีชี้วัดความสำเร็จที่ชัดเจน คือ

4.5.1.1 ความคุ้มค่าด้านเวลา (Time Efficiency) สามารถลดระยะเวลาการสแกนลงได้มหาศาลจาก 1 วัน (Arachni) เหลือเพียง 1 ชั่วโมง (OWASP ZAP) โดยที่ยังสามารถรักษาความลึกในการตรวจพบช่องโหว่ระดับวิกฤตไว้ได้ โดยโดยไม่นับ Burp Suite ที่ระยะเวลาขึ้นอยู่กับความสามารถของผู้แสกน

4.5.1.2 ความแม่นยำและความลึก (Detection Depth) ตรวจพบช่องโหว่ที่มีความรุนแรงสูง เช่น SQL Injection และ Path Traversal ได้อย่างแม่นยำ ในขณะที่เครื่องมืออื่นเน้นเพียงการสำรวจไดเรกทอรีพื้นฐานเท่านั้น โดยโดยไม่นับ Burp Suite ที่ตกการคัดเลือกสำหรับการไม่มี Function Automated ไป

4.5.2 ความสำเร็จด้านการบริหารจัดการความเสี่ยง (Risk Mitigation Success) ผลการตรวจสอบยืนยันภายหลังการแก้ไข (Post-Remediation) แสดงให้เห็นถึงการยกระดับความมั่นคงปลอดภัยของระบบอย่างเป็นรูปธรรม

4.5.2.1 การกำจัดความเสี่ยงวิกฤต สามารถปิดช่องโหว่ระดับ สูง (High) และ ปานกลาง (Medium) ได้อย่างสมบูรณ์ คิดเป็น 100% ของรายการที่ตรวจพบในสถานะฐาน

4.5.2.2 การจัดการความเสี่ยงคงเหลือ แม้จะมีช่องโหว่คงเหลือ 9 รายการ (ระดับ Low และ Informational) แต่ผลการวิจัยชี้ให้เห็นชัดเจนว่าเป็นความเสี่ยงในระดับที่ยอมรับได้ (Acceptable Risk) และเกี่ยวข้องกับการตั้งค่าระดับโครงสร้างพื้นฐาน ซึ่งได้ถูกนำเสนอเป็นแนวทางในการพัฒนาลำดับต่อไป

4.5.3 ความสำเร็จด้านการสร้างองค์ความรู้และมาตรฐานการปฏิบัติงาน (Process Standardization) ผลผลิตที่มีค่าที่สุดของการวิจัยนี้คือ คู่มือการแก้ไขช่องโหว่ (Remediation Manual) ที่ผู้วิจัยสร้างขึ้น

4.5.3.1 การเปลี่ยนข้อมูลเทคนิคเป็นแนวทางปฏิบัติ สามารถเปลี่ยนรายงานสแกนที่ซับซ้อนให้กลายเป็นขั้นตอนการแก้ไขที่นักพัฒนา (Developer) นำไปใช้ได้จริง

4.5.3.2 ความยั่งยืน คู่มือนี้ไม่ได้เป็นเพียงผลการทดลอง แต่เป็นฐานข้อมูลความรู้ (Knowledge Base) ที่องค์กรสามารถนำไปใช้เป็นมาตรฐานในการเขียนโค้ดที่ปลอดภัย (Secure Coding) เพื่อป้องกันการเกิดช่องโหว่ซ้ำในอนาคตได้

4.5.3.3 การยืนยันคุณภาพและความน่าเชื่อถือจากมุมมองผู้ใช้งาน (External Validation and Reliability) คุณภาพของคู่มือได้รับการพิสูจน์ผ่านกระบวนการประเมินจากผู้เชี่ยวชาญและผู้ใช้งานจริงจำนวน 4 ท่าน ซึ่งให้คะแนนความพึงพอใจเฉลี่ยสูงถึง 4.54 (ระดับมากที่สุด) โดยเฉพาะอย่างยิ่งในด้านความสามารถในการลดระยะเวลาการทำงานที่ได้รับคะแนนเต็ม 5.00 ผลลัพธ์นี้เป็นเครื่องยืนยันเชิงประจักษ์ว่า คู่มือที่ผู้วิจัยพัฒนาขึ้นมีความสมบูรณ์ทั้งในด้านความถูกต้องเชิงเทคนิคและความสะดวกในการนำไปปฏิบัติจริง ส่งผลให้องค์กรสามารถมั่นใจได้ว่ามาตรฐานการปฏิบัติงานนี้จะช่วยลดความเสี่ยงและเพิ่มประสิทธิภาพในการจัดการช่องโหว่ได้

บทที่ 5

สรุป อภิปรายผล และข้อเสนอแนะ

การค้นคว้าอิสระเรื่อง การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข มีวัตถุประสงค์เพื่อคัดเลือกเครื่องมือที่เหมาะสมที่สุดสำหรับองค์กรกรณีศึกษา และเพื่อทดสอบประสิทธิผลของคู่มือการแก้ไขช่องโหว่ในการลดความเสี่ยงทางไซเบอร์ ผลการดำเนินงานทั้งหมดสามารถสรุป อภิปรายผล และให้ข้อเสนอแนะได้ดังนี้

5.1 สรุปผลการวิจัย

จากการดำเนินงานวิจัยที่ผ่านมา สามารถสรุปผลสัมฤทธิ์ตามวัตถุประสงค์หลักได้ 2 ประเด็น ดังนี้

5.1.1 ผลการเปรียบเทียบประสิทธิภาพเครื่องมือ จากการทดสอบเปรียบเทียบเครื่องมือสแกนช่องโหว่ทางเทคนิค 3 ชนิด พบว่า OWASP ZAP มีความเหมาะสมสูงสุดในการนำมาใช้งานเป็นเครื่องมือหลัก โดยมีประสิทธิภาพด้านเวลาที่โดดเด่นอย่างยิ่ง ซึ่งใช้เวลาในการประมวลผลเพียง 1 ชั่วโมง เมื่อเทียบกับ Arachni ที่ใช้เวลากว่า 24 ชั่วโมง และ Burp Suite ที่ต้องอิงจากความสามารถของผู้ใช้งานในการ manual scan นอกจากนี้ ในด้านคุณภาพของข้อมูล OWASP ZAP สามารถตรวจพบช่องโหว่ในระดับวิกฤต (Critical/High) เช่น SQL Injection (Time Based) ได้อย่างแม่นยำ ในขณะที่ Arachni เน้นการตรวจพบเพียงข้อมูลพื้นฐานของไดเรกทอรีเพราะได้ยกเลิกการใช้งานไปแล้ว

5.1.2 ผลสัมฤทธิ์ของกระบวนการจัดการและแก้ไขช่องโหว่ กระบวนการยืนยันผลการแก้ไข (Validation) พิสูจน์ให้เห็นว่า "คู่มือการแก้ไขช่องโหว่" ที่ผู้วิจัยสร้างขึ้นมีประสิทธิผลในการลดความเสี่ยงอย่างเป็นรูปธรรม โดยสามารถ กำจัดช่องโหว่ระดับสูง (High) และปานกลาง (Medium) ได้ครบถ้วน 100% ส่งผลให้จำนวนประเภทช่องโหว่รวมลดลงจาก 17 รายการ เหลือเพียง 9 รายการ ซึ่งช่องโหว่ที่ยังคงเหลืออยู่นั้นล้วนเป็นช่องโหว่ระดับต่ำ (Low) ที่เกี่ยวข้องกับการตั้งค่าเว็บเซิร์ฟเวอร์และโครงสร้างพื้นฐานเท่านั้น

5.2 การอภิปรายผล

ผู้วิจัยได้วิเคราะห์และอภิปรายผลการดำเนินงานในประเด็นที่สำคัญทางเทคนิค ดังนี้

5.2.1 อิทธิพลของสถานะซอฟต์แวร์ต่อความแม่นยำ สาเหตุที่ประสิทธิภาพของ Arachni ด้อยกว่า OWASP ZAP อย่างชัดเจน ทั้งในด้านระยะเวลาและความแม่นยำ เกิดจากสถานะของซอฟต์แวร์ที่หยุดการพัฒนา (Discontinued) ส่งผลให้ Engine ในการวิเคราะห์ Payload ล้าสมัย ไม่สามารถรับมือกับโครงสร้างเว็บแอปพลิเคชันสมัยใหม่ได้ดีเท่าที่ควร ในขณะที่ OWASP ZAP มีการอัปเดตสม่ำเสมอจากชุมชนนักพัฒนาทั่วโลก ทำให้มีความสามารถในการทำ Crawling และการส่ง Payload ที่ซับซ้อนได้รวดเร็วและตรงจุดมากกว่า ยิ่งไม่ต้องกล่าวถึง Burp Suite ที่ต้องอาศัยความสามารถของผู้ใช้งานเท่านั้น

5.2.2 ประสิทธิผลของคู่มือการแก้ไขต่อการลดช่องโหว่ประยุกต์ (Application Layer) ความสำเร็จในการปิดช่องโหว่ระดับ High และ Medium ได้ทั้งหมด สะท้อนให้เห็นว่า "ช่องว่างระหว่างเครื่องมือกับนักพัฒนา" สามารถปิดได้ด้วยการจัดทำคู่มือที่มีตัวอย่าง Secure Coding ที่ชัดเจน การที่นักพัฒนาเห็นตัวอย่างเปรียบเทียบระหว่างโค้ดที่มีปัญหาและโค้ดที่ได้รับการแก้ไข (Remediation Example) ช่วยให้กระบวนการ Patching เกิดขึ้นได้อย่างรวดเร็วและถูกต้องตั้งแต่วันแรกที่เริ่มแก้ไข

5.2.3 การวิเคราะห์ความเสี่ยงคงเหลือ (Residual Risk) สำหรับการคงเหลือของช่องโหว่ 9 รายการ (เช่น Cookie Without Secure Flag, HSTS, และ X-Content-Type-Options) ผู้วิจัยอภิปรายว่านี่คือภาพสะท้อนของ Shared Responsibility Model ในงาน Cyber Security เนื่องจากช่องโหว่เหล่านี้ต้องอาศัยการแก้ไขที่ไฟล์กำหนดค่า (Configuration Files) ของ Web Server (เช่น Nginx หรือ Apache) หรือการจัดการที่ใบรับรอง SSL ซึ่งโดยปกติจะเป็นหน้าที่ของทีม System Administrator หรือ Infrastructure ทีม ไม่ใช่นักพัฒนาซอฟต์แวร์เพียงลำพัง

5.3 ข้อจำกัดของการวิจัย

เพื่อให้ผู้ที่สนใจนำงานวิจัยไปศึกษาต่อเข้าใจขอบเขต ผู้วิจัยขอระบุข้อจำกัดที่พบดังนี้

5.3.1 ขอบเขตของสิทธิ์ในการเข้าถึง (Administrative Constraints) ผู้วิจัยได้รับสิทธิ์ในระดับการพัฒนาแอปพลิเคชันเท่านั้น ทำให้ไม่สามารถทดลองแก้ไขช่องโหว่ที่เกี่ยวข้องกับการตั้งค่า

ระบบปฏิบัติการหรือเซิร์ฟเวอร์โดยตรงได้ ส่งผลให้ผลการสแกนรอบสุดท้ายยังคงมีรายการแจ้งเตือนในระดับโครงสร้างพื้นฐานปรากฏอยู่

5.3.2 มิติของการทดสอบ (Black Box vs. White Box) งานวิจัยนี้มุ่งเน้นการทดสอบจากมุมมองภายนอก (DAST) ซึ่งมีความแม่นยำในการหาจุดอ่อนที่ถูกโจมตีได้จริงจากอินเทอร์เน็ต แต่อาจไม่ครอบคลุมช่องโหว่เชิงตรรกะ (Business Logic) หรือจุดอ่อนลึกๆ ในซอร์สโค้ดที่การสแกนแบบ Black Box เข้าไม่ถึง

5.4 ข้อเสนอแนะ

5.4.1 ข้อเสนอแนะเพื่อการนำไปใช้งาน (Practical Recommendations)

5.4.1.1 การปรับปรุงกระบวนการพัฒนา (Process Integration) ในปัจจุบันองค์กรของผู้วิจัย ได้ใช้งาน OWASP ZAP เข้ามาเป็นส่วนหนึ่งของระบบตรวจสอบอัตโนมัติ (Automated Security Testing) ในช่วงก่อนการขึ้นระบบจริง (UAT Phase) เพื่อให้เกิดการสแกน Baseline เป็นระยะ อยู่แล้ว แต่เพื่อการค้นหาตัวเลือกใหม่ ๆจึงมีการจัดทำค้นคว้าอิสระนี้ขึ้น แต่ผลประกฏว่า OWASP ZAP ยังคงเป็นตัวเลือกที่ดีที่สุดในขณะนี้ ดังนั้นการใช้คู่มือที่ผู้วิจัยสร้างขึ้นเป็นฐานข้อมูลความรู้ (Knowledge Base) สำหรับนักพัฒนารุ่นใหม่ เพื่อลดการซ้ำซ้อนของการปิดช่องโหว่จึงเห็นควรเป็น Process ที่ดี ที่ควรมีคู่กันไป

5.4.1.2 การยกระดับความปลอดภัยในระดับโครงสร้าง (Hardening) ควรนำรายงานผลการวิจัยในส่วนที่ยังคงเหลือ (9 รายการ) ไปดำเนินการปรับปรุง Security Headers และการตั้งค่า SSL/TLS ที่เว็บเซิร์ฟเวอร์ ให้มากที่สุดเท่าที่จะทำได้ เพื่อปิดช่องว่างความเสี่ยงระดับต่ำให้ครบถ้วนตามแนวทาง Defense-in-Depth ต่อไป

5.4.2 ข้อเสนอแนะสำหรับการศึกษาในอนาคต (Future Research) เพื่อให้เกิดการต่อยอดองค์ความรู้จากการวิจัยนี้ ผู้วิจัยขอเสนอแนะประเด็นสำหรับการศึกษาในครั้งต่อไป ดังนี้

5.4.2.1 การศึกษาเปรียบเทียบระหว่าง DAST และ IAST (Interactive Application Security Testing)เนื่องจากการวิจัยครั้งนี้เน้นการทดสอบแบบภายนอก (DAST) ซึ่งมีข้อจำกัดในการมองเห็นลอจิกภายในซอร์สโค้ด การศึกษาในอนาคตควรนำเทคโนโลยี IAST ซึ่งเป็นการติดตั้ง Agent ไว้ในตัวแอปพลิเคชันมาทดสอบร่วมด้วย เพื่อเปรียบเทียบว่าความแม่นยำในการตรวจพบช่องโหว่จะ

เพิ่มขึ้นเพียงใด และสามารถลดอัตราการแจ้งเตือนที่ผิดพลาด (False Positive) ได้ดีกว่าเครื่องมือแบบ DAST เพียงอย่างเดียวหรือไม่

5.4.2.2 การวิจัยการตั้งค่าความมั่นคงปลอดภัยในระดับโครงสร้างพื้นฐานแบบอัตโนมัติ (Automated Infrastructure Hardening) สืบเนื่องจากผลการวิจัยที่พบว่าช่องโหว่ระดับต่ำ (Low) ยังคงเหลืออยู่ 9 รายการเนื่องจากเป็นข้อจำกัดในระดับเว็บเซิร์ฟเวอร์ การศึกษาในอนาคตควรขยายขอบเขตไปสู่การวิจัยเครื่องมือประเภท Infrastructure as Code (IaC) Scanning เพื่อตรวจสอบการตั้งค่าไฟล์ Configuration (เช่น Nginx, Apache หรือ Dockerfile) โดยอัตโนมัติก่อนการขึ้นระบบ ซึ่งจะช่วยปิดช่องว่างความเสี่ยงในส่วนที่นักพัฒนาซอฟต์แวร์ไม่สามารถเข้าถึงได้

5.4.2.3 การพัฒนาระบบตรวจสอบความปลอดภัยแบบต่อเนื่อง (Continuous Security Monitoring) ในสภาพแวดล้อม DevSecOps จากการที่ผลการวิจัยยืนยันว่า OWASP ZAP สามารถประมวลผลได้อย่างรวดเร็ว (1 ชั่วโมง) การศึกษาครั้งต่อไปควรทดลองบูรณาการเครื่องมือนี้เข้ากับระบบ CI/CD Pipeline อย่างเต็มรูปแบบ เพื่อศึกษาผลกระทบต่อความเร็วในการปล่อยซอฟต์แวร์ (Velocity) และประสิทธิภาพในการตรวจพบช่องโหว่ทันทีที่มีการแก้ไขโค้ด (Real-time Detection) แทนการสแกนเป็นรอบๆ แบบเดิม

5.4.2.4 การทดสอบความแม่นยำของเครื่องมือกับสถาปัตยกรรมแบบ Microservices และ API เว็บแอปพลิเคชันเป้าหมายในการวิจัยนี้มีลักษณะเฉพาะรูปแบบหนึ่ง การศึกษาในอนาคตควรนำเครื่องมือที่คัดเลือกได้ (OWASP ZAP) ไปทดสอบประสิทธิภาพกับสถาปัตยกรรมสมัยใหม่อื่นๆ เช่น RESTful API, GraphQL หรือระบบที่ทำงานบน Container (Docker/Kubernetes) เพื่อประเมินว่าเครื่องมือแบบ Open-source ยังคงรักษามาตรฐานความแม่นยำและความครอบคลุมได้ดีเพียงใดในสภาพแวดล้อมที่ซับซ้อนขึ้น

บรรณานุกรม

[1] Fortinet, (2024). 2024 Global Threat Landscape Report from FortiGuard Labs [Online]. Available: https://www.fortinet.com/resources/reports/threat-landscape-report

[2] IBM, (2025). Cost of a Data Breach Report 2025. IBM. [Online]. Available: https://www.ibm.com/reports/data-breach

[3] “Top 10 Web Application Security Risks” Open Worldwide Application Security Project Blog. Accessed: Sept, 2025. [Online]. Available: https://owasp.org/www-project-top-ten/

[4] Belay, T. E., Gupta, S., & Kumar, A. (2025). Systematic Evaluation And Model Based Selection Of Web Vulnerability Scanners: Toward A Prior Guided Assessment Framework. International Journal of Environmental Sciences.

[5] Escarrone, T., Rosa, R. L., Kreutz, D., et al. (2025). Sob a Lupa Corporativa Brasileira: Avaliação da Cobertura de Scanners de Vulnerabilidades em Aplicações Atuais.

[6] Potti, U.-S., Huang, H.-S., Chen, H.-T., et al. (2025). Security Testing Framework for Web Applications: Benchmarking ZAP V2.12.0 and V2.13.0 by OWASP as an example.

[7] Jha, T. K. (2024). Modern Web Application Penetration Testing Tools: A Comparative Analysis. Deleted Journal.

[8] Kondraciuk, A., Bartos, A., & Pańczyk, B. (2022). Comparative analysis of the effectiveness of OWASP ZAP, Burp Suite, Nikto and Skipfish in testing the security of web applications. Journal of Computer Sciences Institute.

[9] Mburano, B., & Si, W. (2018). Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark. 2018 IEEE International Conference on Systems Engineering.

[10] Alsaleh, M., Alomar, N., Alshreef, M., et al. (2017). Performance-Based Comparative Assessment of Open-Source Web Vulnerability Scanners. Security and Communication Networks.

[11] NIST SP 800-115 (2008) Technical Guide to Information Security Testing and Assessment. NIST. [Online]. Available: https://www.ibm.com/reports/data-breach



ภาคผนวก ก

คู่มือแนวทางการแก้ไขช่องโหว่

คู่มือแนวทางการแก้ไขช่องโหว่

ลำดับที่	1		
ชื่อของช่องโหว่	SQL Injection	**ระดับความเสี่ยง (Severity)**	Risk=High, Confidence=Low
พารามิเตอร์ที่พบปัญหา	year (ผ่านเมธอด GET)	**ประเภทช่องโหว่**	SQL Injection (SQLi) ในไฟล์ summary.php
คำอธิบายโดยสังเขป	แอปพลิเคชันขาดกลไกการตรวจสอบและคัดกรองข้อมูลนำเข้า (Input Validation and Sanitization) ที่เพียงพอในพารามิเตอร์ year ระบบอนุญาตให้ส่งอักขระพิเศษที่เกี่ยวข้องกับไวยากรณ์ของฐานข้อมูล เช่น %27 ซึ่งเป็นการเข้ารหัส URL ของเครื่องหมายอัญประกาศเดี่ยว (Single Quote: ') เข้าไปประมวลผลร่วมกับคำสั่ง SQL หลังบ้านได้		
วิธีการแก้ไขช่องโหว่	ใช้ Prepared Statements (Parameterized Queries) แทนที่จะนำค่าจากผู้ใช้ไปต่อสตริง (String Concatenation) เข้ากับคำสั่ง SQL ตรงๆ ให้เปลี่ยนมาใช้การทำ Parameterized Query ซึ่งจะแยกคำสั่ง SQL กับข้อมูลออกจากกัน ทำให้ฐานข้อมูลมองว่า Input เป็นแค่ "ข้อมูล" ไม่ใช่ "คำสั่ง"		
ตัวอย่างการแก้ไขช่องโหว่	1.แก้ไขด้วย PDO ใน PHP ใช้ Prepared Statement $year = $_GET['year']; เตรียมคำสั่ง SQL โดยใช้ placeholder (?) หรือ named parameter (:year) $stmt = $pdo->prepare('SELECT * FROM summary_data WHERE report_year = :year'); ผูกค่าและประมวลผล (Execute) $stmt->execute(['year' => $year]); $data = $stmt->fetchAll(); 2. บังคับประเภทข้อมูล ในกรณีนี้ พารามิเตอร์ year ควรจะเป็นตัวเลขจำนวนเต็มเสมอ เราสามารถกรองและบังคับให้ค่าที่รับมาเป็นตัวเลขเท่านั้นก่อนนำไปใช้งานได้โดยใช้ filter_input เพื่อตรวจสอบว่าเป็นตัวเลขจริงๆ $year = filter_input(INPUT_GET, 'year', FILTER_VALIDATE_INT); หรือใช้การแปลงเป็น Integer โดยตรง $year = intval($_GET['year']); if ($year === false || $year === 0) {ข้อมูลไม่ถูกต้อง ให้หยุดการทำงานหรือกำหนดค่า Default die("Invalid year parameter.");} จากนั้นค่อยนำ $year ไป query ต่อ (ควรใช้คู่กับวิธีที่ 1) 3. ตรวจสอบพารามิเตอร์อื่นที่เกี่ยวข้อง ใน URL มีพารามิเตอร์ &q= อยู่ด้วย แม้จะเน้นที่ year แต่ก็ควรนำวิธี Prepared Statements ไปใช้ครอบคลุมกับตัวแปร q และ Input อื่นๆ ทั้งหมดที่รับมาจากผู้ใช้ด้วย		

ลำดับที่	2		
ชื่อของช่องโหว่	Content Security Policy (CSP) Header Not Set	**ระดับความเสี่ยง (Severity)**	Risk=Medium, Confidence=High
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การขาดหายไปของส่วนหัว CSP
คำอธิบายโดยสังเขป	ไม่มีการกำหนดค่า HTTP Header ที่ชื่อว่า Content-Security-Policy (CSP) ส่งกลับมายังเว็บเบราว์เซอร์ฝั่งไคลเอนต์ นโยบาย CSP เป็นมาตรฐานและกลไกการรักษาความปลอดภัยเชิงลึกที่ช่วยให้ผู้ดูแลระบบสามารถกำหนดขอบเขต (Whitelist) ของแหล่งที่มาที่เชื่อถือได้ สำหรับการโหลดทรัพยากรต่างๆ		
วิธีการแก้ไขช่องโหว่	ควรจะต้องกำหนดค่า CSP พื้นฐาน Content-Security-Policy		
ตัวอย่างการแก้ไขช่องโหว่	การกำหนดค่า CSP พื้นฐานคือ Content-Security-Policy: default-src 'self'; ระดับ Web Server (Apache - ในไฟล์ .htaccess หรือ httpd.conf): <IfModule mod_headers.c> Header set Content-Security-Policy "default-src 'self';" </IfModule> ระดับแอปพลิเคชัน <?php // ควรใส่ไว้ที่ส่วนหัวสุดของไฟล์ก่อนที่จะมีการแสดงผล HTML ใด ๆ header("Content-Security-Policy: default-src 'self';"); ?> ข้อควรระวังในการนำไปใช้ default-src 'self' เป็นเพียงตัวอย่างเริ่มต้น หากแอปพลิเคชันมีการเรียกใช้ทรัพยากรจากภายนอกที่จำเป็น เช่น Google Analytics, CDN สำหรับโหลดไลบรารี CSS/JS หรือ API ภายนอก ทีมพัฒนาจำเป็นต้องปรับแต่งนโยบายให้สอดคล้องกับการใช้งานจริงเพื่อป้องกันไม่ให้การทำงานของระบบผิดเพี้ยนไปจากการบล็อกทรัพยากรที่ถูกต้อง		
ลำดับที่	3		
ชื่อของช่องโหว่	Sub Resource Integrity Attribute Missing	**ระดับความเสี่ยง (Severity)**	Risk=Medium, Confidence=High
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การขาดหายไปของแอตทริบิวต์ตรวจสอบความสมบูรณ์ของทรัพยากร
คำอธิบายโดยสังเขป	มีการเรียกใช้งานทรัพยากรจากภายนอก (Third-party Resources) เช่น ไฟล์สคริปต์ (JavaScript) หรือไฟล์สไตล์ชีต (CSS) จาก Content Delivery Network (CDN) โดย		

	ไม่ได้กำหนดแอตทริบิวต์ integrity ในแท็ก <script> หรือ <link> Subresource Integrity (SRI)		
วิธีการแก้ไขช่องโหว่	ตรวจสอบ <script> และ <link> ทั้งหมดที่มีการดึงไฟล์มาจากโดเมนภายนอก และดำเนินการเพิ่มแอตทริบิวต์ integrity พร้อมกับ crossorigin="anonymous" เพื่อให้เบราว์เซอร์ทำการตรวจสอบความถูกต้องของไฟล์ก่อนใช้งาน โดยสามารถสร้างค่า Hash (รองรับ SHA-256, SHA-384 หรือ SHA-512) ได้จากเครื่องมือออนไลน์ หรือสร้างผ่าน Command Line		
ตัวอย่างการแก้ไขช่องโหว่	ตัวอย่างชุดคำสั่งสำหรับการแก้ไขใน HTML <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"> </script><link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/ bootstrap.min.css"><script src=https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js integrity="sha512894YE6QWD5I59HgZOGReFYm4dnWc1Qt5NtvYSaNcOP+ u1T9qYdvdihz0PPSiiqn/+/3e7Jo4EaG7TubfWGUrMQ==" crossorigin= "anonymous" referrerpolicy="no-referrer"> <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/ bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOn IxN0hoP+VmmDGMN5t9UJ0Z"crossorigin="anonymous"> ข้อควรระวัง: หากมีการอัปเดตเวอร์ชันของไลบรารีในอนาคต จะต้องทำการสร้างและอัปเดตค่า integrity hash ให้ตรงกับไฟล์เวอร์ชันใหม่ด้วยเสมอ		
ลำดับที่	4		
ชื่อของช่องโหว่	Application Error Disclosure	**ระดับความเสี่ยง (Severity)**	Risk=Medium, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การเปิดเผยข้อผิดพลาดของแอปพลิเคชัน
คำอธิบายโดยสังเขป	แอปพลิเคชันมีการตั้งค่าการแสดงผลข้อผิดพลาดที่ไม่รัดกุม เมื่อเกิดข้อผิดพลาดในการประมวลผล (เช่น ไม่พบไฟล์) เซิร์ฟเวอร์ได้ส่งหน้าเพจที่แสดงรายละเอียดของข้อผิดพลาดทางเทคนิคกลับมาให้ผู้ใช้งานทั่วไปเห็นโดยตรง		

วิธีการแก้ไขช่องโหว่	ปิดการแสดงผลข้อผิดพลาด (Disable Error Display) ผู้พัฒนาหรือผู้ดูแลระบบเซิร์ฟเวอร์ควรปรับการตั้งค่าของภาษาโปรแกรมมิ่งหรือเฟรมเวิร์กเพื่อซ่อนข้อผิดพลาด (Error Details) จากผู้ใช้ปลายทาง		
ตัวอย่างการแก้ไขช่องโหว่	ตัวอย่างการตั้งค่าในไฟล์ php.ini (สำหรับ PHP ทั่วไป) Ini, TOML display_errors = Off display_startup_errors = Off ตัวอย่างการตั้งค่าในไฟล์ .env (หากระบบใช้เฟรมเวิร์กอย่าง Laravel) Code snippet APP_ENV=production APP_DEBUG=false สร้างหน้าข้อผิดพลาดแบบกำหนดเอง ระบบควรดักจับข้อผิดพลาดที่เกิดขึ้น และแสดงผลหน้าเว็บเพจแบบทั่วไป (Generic Error Message) ที่เป็นมิตรกับผู้ใช้แทน เช่น หน้า "500 Internal Server Error - ขออภัย ระบบขัดข้องชั่วคราว" โดยไม่เปิดเผยข้อมูลเชิงลึกใดๆ การบันทึกข้อผิดพลาดลงในError Logging เพื่อไม่ให้กระทบต่อการทำงานของทีมพัฒนาในการแก้ไขบั๊ก ควรตั้งค่าให้บันทึกรายละเอียดข้อผิดพลาดทั้งหมดลงในไฟล์ Log บนเซิร์ฟเวอร์ ซึ่งบุคคลภายนอกไม่สามารถเข้าถึงได้ ตัวอย่างการตั้งค่าในไฟล์ php.ini Ini, TOML log_errors = On		
ลำดับที่	5		
ชื่อของช่องโหว่	Buffer Overflow	**ระดับความเสี่ยง (Severity)**	Risk=Medium, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	ภาวะหน่วยความจำล้นเกิน
คำอธิบายโดยสังเขป	ช่องโหว่ที่เกิดขึ้นเมื่อแอปพลิเคชันรับข้อมูลนำเข้า (Input) ที่มีขนาดใหญ่เกินกว่าที่พื้นที่หน่วยความจำ (Buffer) ที่จัดสรรไว้จะรองรับได้ ส่งผลให้ข้อมูลส่วนเกินล้นไปทับซ้อนข้อมูลในหน่วยความจำตำแหน่งที่อยู่ติดกัน		
วิธีการแก้ไขช่องโหว่	ข้อสังเกตพิเศษสำหรับการกรณีนี้ เนื่องด้วยแอปพลิเคชันใน URL นี้พัฒนาด้วยภาษา PHP ซึ่งมีระบบจัดการหน่วยความจำอัตโนมัติ การเกิด Buffer Overflow ในระดับแอปพลิเคชันโค้ด โดยตรงจึงเป็นไปได้ยาก การที่เครื่องมือสแกน (Scanner) แจ้งเตือนช่องโหว่นี้ มักเกิดจาก 2 กรณีหลัก 1. False Positive (ผลบวกปลวง) Scanner อาจส่ง Payload ที่เป็นตัวอักษรขนาดยาวมากเข้าไประบบ (เช่น ผ่าน Form Login แบบ POST) แล้วระบบตอบกลับด้วย HTTP 500 Error, เกิดการประมวลผลช้าลงอย่างมาก, หรือไม่มีการตอบสนอง (Timeout) ทำ		

	ให้ Scanner "ตีความ" ไปเองว่าเกิดการ Crash จาก Buffer Overflow หรือ Denial of Service (DoS) 2. ช่องโหว่ในระดับ Core/Extension อาจเกิดช่องโหว่ Buffer Overflow จริงๆ แต่ไม่ได้เกิดที่โค้ด PHP ของนักพัฒนา ทว่าเกิดที่ตัวแปลภาษา (PHP Engine), Web Server (Apache/Nginx), หรือ Module/Extension ของบุคคลที่สามที่เขียนด้วยภาษา C/C++ ซึ่งจัดการข้อมูลที่ส่งเข้ามาได้ไม่ดีพอ		
ตัวอย่างการแก้ไขช่องโหว่	1. การตรวจสอบยืนยันความถูกต้องก่อนทำการแก้ไขโค้ด ผู้ดูแลระบบควรตรวจสอบ Server Logs (เช่น Apache/Nginx Error Logs หรือ PHP FPM Logs) ในช่วงเวลาที่ทำการสแกน เพื่อดูว่าระบบเกิด Fatal Error จากหน่วยความจำเต็ม (Memory Exhausted), เกิด Segmentation Fault (Segfault), หรือเป็นเพียงข้อผิดพลาดของแอปพลิเคชันทั่วไป เพื่อยืนยันว่าเป็นช่องโหว่จริงหรือไม่ 2. การจำกัดขนาดข้อมูลนำเข้า ไม่ว่าจะเป็นช่องโหว่จริงหรือไม่ แนวทางปฏิบัติที่ดีที่สุด (Best Practice) คือการจำกัดความยาวสูงสุดของข้อมูลที่รับเข้ามาจากผู้ใช้ทั้งหมด (เช่น ชื่อผู้ใช้ รหัสผ่าน หรือพารามิเตอร์อื่นๆ) ทั้งในฝั่ง Client-side (HTML maxlength) และ Server-side (การตรวจสอบความยาวสตริงใน PHP) ก่อนนำไปประมวลผล 3. การอัปเดตระบบปฏิบัติการและซอฟต์แวร์ ควรตรวจสอบให้แน่ใจว่า Web Server (Apache/Nginx), เวอร์ชันของ PHP, และไลบรารีต่างๆ ที่ติดตั้งบนเซิร์ฟเวอร์ ได้รับการอัปเดตเป็นเวอร์ชันล่าสุดเพื่ออุดช่องโหว่ด้านความปลอดภัยที่รู้จัก (CVEs) 4. การติดตั้ง Web Application Firewall (WAF) การใช้งาน WAF สามารถช่วยกรองและบล็อก HTTP Requests ที่มีขนาดใหญ่ผิดปกติ หรือมีลักษณะที่บ่งชี้ถึงความพยายามในการทำ Buffer Overflow ได้ก่อนที่จะเข้าถึงตัวแอปพลิเคชัน		
ลำดับที่	6		
ชื่อของช่องโหว่	Vulnerable JS Library	**ระดับความเสี่ยง (Severity)**	Risk=Medium, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การใช้งานไลบรารี JavaScript ที่มีช่องโหว่
คำอธิบายโดยสังเขป	แอปพลิเคชันมีการเรียกใช้งานไลบรารีฝั่งไคลเอนต์ (Client-side Library) คือ bootstrap.bundle.min.js ซึ่งเป็นไลบรารีของเฟรมเวิร์ก Bootstrap ในเวอร์ชันที่ล้าสมัย (Outdated Version) และมีรายงานช่องโหว่ด้านความปลอดภัยที่ถูกเปิดเผยต่อสาธารณะ (Known Vulnerabilities หรือ CVEs)		

ข้อเสนอและแนวทางการแก้ไขช่องโหว่	1. ตรวจสอบและอัปเดตเวอร์ชันของไลบรารี (Update and Patch) ผู้พัฒนาควรตรวจสอบเวอร์ชันปัจจุบันของ Bootstrap ที่ระบบกำลังใช้งานอยู่ และดำเนินการอัปเกรดให้เป็นเวอร์ชันล่าสุดที่มีความเสถียร หรืออย่างน้อยที่สุดคือเวอร์ชั่นที่มีการแพตช์แก้ไขช่องโหว่ความปลอดภัยเป้าหมายแล้ว 2. การทดสอบระบบหลังการอัปเดต เนื่องจากการอัปเกรดเวอร์ชันของไลบรารี UI อาจส่งผลกระทบโดยตรงต่อการจัด Layout ฟังก์ชันการทำงาน หรือพฤติกรรมของหน้าเว็บเพจ ทีมพัฒนาจำเป็นต้องทำการทดสอบระบบ อย่างละเอียดเพื่อให้มั่นใจว่าแอปพลิเคชันยังคงแสดงผลและใช้งานได้ตามปกติหลังจากการอัปเดต 3. การจัดการซอฟต์แวร์ส่วนประกอบ (Software Composition Analysis - SCA) ควรมีกระบวนการติดตาม ตรวจสอบ และทำรายการ ของไลบรารีภายนอก (Third-party Libraries) ทั้งหมดที่นำมาใช้ในโปรเจกต์อย่างสม่ำเสมอ เพื่อให้สามารถรับมือและอัปเดตแพตช์รักษาความปลอดภัยได้ทันท่วงทีเมื่อมีช่องโหว่ใหม่ถูกค้นพบในอนาคต ** ในส่วนเฉพาะที่ไม่สามารถแก้ไขได้ แนะนำให้ hide path ไม่ให้สามารถเข้าถึงโดยตรงจากหน้า .js ได้		
ลำดับที่	7		
ชื่อของช่องโหว่	Server Leaks Version Information via "Server" HTTP ResponseHeader Field	**ระดับความเสี่ยง (Severity)**	Risk=Low, Confidence=High
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การเปิดเผยข้อมูลเวอร์ชันของเว็บเซิร์ฟเวอร์
คำอธิบายโดยสังเขป	เมื่อเว็บเซิร์ฟเวอร์ตอบกลับการร้องขอ (HTTP Response) กลับไปยังเว็บเบราว์เซอร์หรือเครื่องมือสแกน โดยปกติจะมีการแนบส่วนหัว (HTTP Header) ที่ชื่อว่า Server มาด้วย ซึ่งจากการตั้งค่าเริ่มต้นของเว็บเซิร์ฟเวอร์ส่วนใหญ่ (เช่น Apache, Nginx, IIS) ค่าของ Header นี้มักจะระบุชนิดและหมายเลขเวอร์ชันของซอฟต์แวร์เว็บเซิร์ฟเวอร์ที่กำลังทำงานอยู่ รวมถึงอาจมีข้อมูลระบบปฏิบัติการแฝงมาด้วย (ตัวอย่างเช่น Server: Apache/2.4.41 (Ubuntu))		
วิธีการแก้ไขช่องโหว่	ผู้ดูแลระบบเซิร์ฟเวอร์ ควรปรับแต่งการตั้งค่าของ Web Server เพื่อซ่อนหมายเลขเวอร์ชัน และจำกัดการแสดงผลใน Header Server ให้เหลือน้อยที่สุด หรือแสดงเพียงชื่อซอฟต์แวร์โดยไม่มีเวอร์ชันกำกับ		

ตัวอย่างการแก้ไขช่องโหว่	1. สำหรับ Apache Web Server เข้าไปแก้ไขไฟล์คอนฟิกูเรชัน (เช่น httpd.conf หรือ apache2.conf) และเพิ่มหรือแก้ไขคำสั่งดังนี้ ServerTokens Prod ServerSignature Off (ผลลัพธ์ จะแสดงเพียง Server: Apache โดยไม่มีหมายเลขเวอร์ชันและชื่อ OS) 2. สำหรับ Nginx Web Server เข้าไปแก้ไขไฟล์ nginx.conf (มักจะอยู่ในบล็อก http { ... }) และเพิ่มคำสั่งดังนี้ server_tokens off; (ผลลัพธ์ จะแสดงเพียง Server: nginx โดยไม่มีหมายเลขเวอร์ชัน) 3. สำหรับ Microsoft IIS สามารถใช้เครื่องมือ URL Rewrite Module ใน IIS Manager เพื่อสร้างRule ลบหรือแก้ไข HTTP Response Header ที่ชื่อว่า Server ออกไปทั้งหมด		
ลำดับที่	8		
ชื่อของช่องโหว่	Strict-Transport-Security Header Not Set	**ระดับความเสี่ยง (Severity)**	Risk=Low, Confidence=High
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การขาดหายไปของส่วนหัว HTTP Strict Transport Security
คำอธิบายโดยสังเขป	HTTP Response Header ของเว็บเซิร์ฟเวอร์ ไม่มีการส่งส่วนหัว Strict-Transport-Security (HSTS) กลับมายังเว็บเบราว์เซอร์		
วิธีการแก้ไขช่องโหว่	ผู้ดูแลระบบควรตั้งค่าเว็บเซิร์ฟเวอร์ให้ส่ง Header Strict-Transport-Security กลับมาในทุกๆ HTTP Response ที่เป็น HTTPS โดยควรกำหนดระยะเวลา ให้นานเพียงพอ (เช่น 1 ปี หรือ 31536000 วินาที) และครอบคลุมโดเมนย่อยด้วย		
ตัวอย่างการแก้ไขช่องโหว่	1. สำหรับ Apache เพิ่มคำสั่งนี้ในไฟล์คอนฟิกูเรชัน (เช่น .htaccess) <IfModule mod_headers.c>Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"</IfModule> 2. สำหรับ Nginx เพิ่มคำสั่งนี้ในไฟล์ nginx.conf ภายใน server { ... } ที่จัดการ HTTPS add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always; 3. สำหรับ Microsoft IIS สามารถเพิ่ม Header ได้ผ่านทางเมนู "HTTP Response Headers" ใน IIS Manager โดยกำหนดชื่อเป็น Strict-Transport-Security และใส่ค่าเป็น max-age=31536000; includeSubDomains		

	ข้อควรระวัง: ควรทดสอบการทำงานของระบบและ Subdomain ทั้งหมดให้แน่ใจว่ารองรับ HTTPS อย่างสมบูรณ์แล้ว ก่อนที่จะเปิดใช้งาน includeSubDomains		
ลำดับที่	9		
ชื่อของช่องโหว่	HTTPS Content Available via HTTP	**ระดับความเสี่ยง (Severity)**	Risk=Low, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	เนื้อหาที่ควรเข้ารหัสสามารถเข้าถึงได้ผ่าน
คำอธิบายโดยสังเขป	เว็บเซิร์ฟเวอร์อนุญาตให้ผู้ใช้งานดึงข้อมูลหรือไฟล์ทรัพยากรต่างๆ ของเว็บไซต์ ผ่านทางโปรโตคอล HTTP แบบปกติที่ไม่มีการเข้ารหัสข้อมูลได้ แม้ว่าระบบจะรองรับโปรโตคอล HTTPS ที่มีความปลอดภัยแล้วก็ตาม เกิดขึ้นเนื่องจากเว็บเซิร์ฟเวอร์ขาดการตั้งค่าเพื่อบังคับเปลี่ยนเส้นทาง จาก HTTP ไปยัง HTTPS		
วิธีการแก้ไขช่องโหว่	เพื่อแก้ไขปัญหานี้และยกระดับความปลอดภัย ผู้ดูแลระบบควรตั้งค่าเว็บเซิร์ฟเวอร์ให้บังคับการใช้งาน HTTPS เท่านั้น โดยเปลี่ยนเส้นทาง (Redirect) คำขอ HTTP ทั้งหมดไปยัง HTTPS ด้วยรหัสสถานะ 301 Moved Permanently		
ตัวอย่างการแก้ไขช่องโหว่	1. สำหรับ Apache Web Server เพิ่มคำสั่งด้านล่างนี้ในไฟล์ .htaccess หรือไฟล์ Virtual Host RewriteEngine On RewriteCond %{HTTPS} off RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301] 2. สำหรับ Nginx Web Server: เพิ่มบล็อกเซิร์ฟเวอร์เพื่อรับคำขอพอร์ต 80 และทำการ Redirect server {listen 80; server_name devsiapp2.si.mahidol.ac.th; return 301 https://$host$request_uri;} 3. สำหรับ Microsoft IIS สามารถใช้เครื่องมือ URL Rewrite Module เพื่อสร้างกฎ (Rule) ตรวจสอบเงื่อนไข {HTTPS} ว่ามีค่าเป็น ^OFF$ หรือไม่ และกำหนด Action เป็น Redirect ไปยัง https://{HTTP_HOST}/{R:1} (ข้อแนะนำเพิ่มเติม การแก้ไขช่องโหว่นี้ควรทำควบคู่ไปกับการเปิดใช้งาน HTTP Strict Transport Security (HSTS)เพื่อให้การป้องกันมีความสมบูรณ์แบบมากที่สุด)		
ลำดับที่	10		

ชื่อของช่องโหว่	Cookie Without Secure Flag	**ระดับความเสี่ยง (Severity)**	Risk=Low, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การตั้งค่าคุกกี้โดยไม่มีแฟล็ก Secure
คำอธิบายโดยสังเขป	เว็บแอปพลิเคชันมีการสร้างคุกกี้เพื่อใช้งานในระบบ (เช่น Session ID สำหรับระบุตัวตน) แต่ไม่มีการกำหนดแอตทริบิวต์ Secure ให้กับคุกกี้ดังกล่าว แอตทริบิวต์ Secure เป็นคำสั่งที่ใช้บอกเว็บเบราว์เซอร์ว่า "ให้ส่งคุกกี้นี้กลับมายังเซิร์ฟเวอร์ผ่านช่องทางที่เข้ารหัส (HTTPS) เท่านั้น" หากไม่มีการกำหนดแฟล็กนี้ เบราว์เซอร์จะยอมส่งคุกกี้พ่วงไปกับคำขอที่เป็น HTTP ปกติด้วยในกรณีที่ผู้ใช้งานเผลอเข้าใช้งานเว็บไซต์ผ่านลิงก์ที่ไม่เข้ารหัส		
วิธีการแก้ไขช่องโหว่	ผู้พัฒนาควรปรับปรุงโค้ดหรือการตั้งค่าของเซิร์ฟเวอร์เพื่อให้มั่นใจว่าคุกกี้ที่สำคัญ (โดยเฉพาะ Session Cookie) มีการแนบแฟล็ก Secure เสมอ		
ตัวอย่างการแก้ไขช่องโหว่	1. การตั้งค่าในระดับ Application (ตัวอย่างภาษา PHP) ควรกำหนดพารามิเตอร์ของ Session Cookie ก่อนที่จะเรียกใช้คำสั่ง session_start() session_set_cookie_params(['secure' => true, // บังคับส่งคุกกี้ผ่าน HTTPS เท่านั้น 'httponly' => true, // แนะนำให้ตั้งค่าเพื่อป้องกัน XSS 'samesite' => 'Lax' // แนะนำให้ตั้งค่าเพื่อป้องกัน CSRF]); session_start(); 2. การตั้งค่าในระดับตัวแปลภาษา (ไฟล์ php.ini) สามารถกำหนดค่าเริ่มต้นให้กับทุก Session ในระบบได้โดยปรับค่าใน php.ini Ini, TOML session.cookie_secure = 1 3. การตั้งค่าในระดับ Web Server (Apache / Nginx) หากไม่สามารถแก้ที่โค้ดได้ สามารถตั้งค่าที่เซิร์ฟเวอร์ให้ทำการเพิ่มแฟล็กนี้เข้าไปใน Header Set-Cookie ขาออกได้ เช่นใน Apache สามารถใช้ Header edit Set-Cookie ^(.*)$ $1;Secure ข้อแนะนำ: การแก้ไขช่องโหว่นี้จะทำงานได้อย่างสมบูรณ์ก็ต่อเมื่อเว็บไซต์มีการบังคับใช้ HTTPS (Force HTTP to HTTPS) และ HSTS อย่างถูกต้องครบถ้วนแล้ว		
ลำดับที่	11		
ชื่อของช่องโหว่	Server Leaks Information via "X-Powered-By" HTTP ResponseHeader Field(s)	**ระดับความเสี่ยง (Severity)**	Risk=Low, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การเปิดเผยข้อมูลเทคโนโลยีเบื้องหลังผ่านส่วนหัว HTTP "X-Powered-By"

คำอธิบายโดยสังเขป	การตอบกลับของเว็บเซิร์ฟเวอร์มีการส่งส่วนหัวที่ชื่อว่า X-Powered-By กลับมายังฝั่งไคลเอนต์ โดยทั่วไปแล้ว เฟรมเวิร์กหรือภาษาโปรแกรมมิ่งฝั่งเซิร์ฟเวอร์ (เช่น PHP, ASP.NET, Express.js) มักจะมีการตั้งค่าเริ่มต้นให้แนบ Header นี้มาด้วยโดยอัตโนมัติ เพื่อบ่งบอกว่าเว็บไซต์หรือแอปพลิเคชันนั้นๆ ถูกสร้างและขับเคลื่อนด้วยเทคโนโลยีใด และบ่อยครั้งมักจะมีการระบุหมายเลขเวอร์ชันของภาษานั้นๆ ติดมาด้วยอย่างชัดเจน		
วิธีการแก้ไขช่องโหว่	เพื่อปฏิบัติตามหลักการรักษาความปลอดภัยแบบซ่อนเร้น (Security by Obscurity) และลดการให้ข้อมูลแก่ผู้โจมตีโดยไม่จำเป็น ขอเสนอให้ผู้ดูแลระบบดำเนินการตั้งค่าเพื่อซ่อนหรือลบ Header X-Powered-By ออกจากการตอบกลับของเซิร์ฟเวอร์		
ตัวอย่างการแก้ไขช่องโหว่	1. การตั้งค่าในระดับตัวแปลภาษา (สำหรับ PHP) เข้าไปแก้ไขไฟล์คอนฟิกูเรชัน php.ini โดยค้นหาบรรทัด expose_php และปรับค่าเป็น Off Ini, TOML expose_php = Off 2. การตั้งค่าในระดับ Web Server (Apache) สามารถใช้โมดูล mod_headers เพื่อลบ Header ขาออก โดยเพิ่มคำสั่งนี้ในไฟล์ .htaccess หรือไฟล์คอนฟิกูเรชันหลัก <IfModule mod_headers.c> Header unset X-Powered-By </IfModule> 3. การตั้งค่าในระดับ Web Server (Nginx) หากใช้ Nginx ร่วมกับ PHP-FPM สามารถตั้งค่าไม่ให้ส่งต่อ Header นี้ได้ในบล็อก location ~ \.php$ { ... }: fastcgi_hide_header X-Powered-By; (หรือหากเป็น Proxy ทั่วไป สามารถใช้ proxy_hide_header X-Powered-By;) 4. สำหรับ Microsoft IIS สามารถเข้าไปที่หน้า IIS Manager เลือกเมนู "HTTP Response Headers" ค้นหาตัวแปรที่ชื่อ X-Powered-By (มักจะมีค่าเป็น ASP.NET) และทำการลบ (Remove) ออกจากระบบได้โดยตรง		
ลำดับที่	12		
ชื่อของช่องโหว่	X-Content-Type-Options Header Missing	**ระดับความเสี่ยง (Severity)**	Risk=Low, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การขาดหายไปของส่วนหัว HTTP X-Content-Type-Options
คำอธิบายโดยสังเขป	จากการตรวจสอบการตอบกลับของเว็บเซิร์ฟเวอร์เมื่อมีการร้องขอไฟล์ทรัพยากร พบว่าไม่มีการส่ง Header ที่ชื่อว่า X-Content-Type-Options กลับมาด้วย เว็บเบราว์เซอร์ส่วนใหญ่มีฟีเจอร์ที่เรียกว่า MIME Sniffing ซึ่งเบราว์เซอร์จะพยายามวิเคราะห์และเดาประเภทของไฟล์จาก "เนื้อหาภายใน" แทนที่จะเชื่อถือค่าที่เซิร์ฟเวอร์แจ้งมาเสมอไป การขาดส่วนหัว X-Content-Type-Options: nosniff จะเป็นการอนุญาตให้เบราว์เซอร์ทำการเดาประเภทไฟล์ได้อย่างอิสระ		

วิธีการแก้ไขช่องโหว่	ผู้ดูแลระบบควรตั้งค่าเว็บเซิร์ฟเวอร์ให้ทำการแนบ HTTP Header X-Content-Type-Options โดยกำหนดค่าเป็น nosniff ในทุกๆ การตอบกลับของเซิร์ฟเวอร์ เพื่อบังคับให้เบราว์เซอร์ยึดถือประเภทไฟล์ตามที่เซิร์ฟเวอร์กำหนดไว้อย่างเคร่งครัด		
ตัวอย่างการแก้ไขช่องโหว่	1. การตั้งค่าในระดับ Web Server (Apache): เพิ่มคำสั่งด้านล่างนี้ในไฟล์ .htaccess หรือไฟล์คอนฟิกูเรชันของ Apache (เช่น httpd.conf) <IfModule mod_headers.c> Header always set X-Content-Type-Options "nosniff" </IfModule> 2. การตั้งค่าในระดับ Web Server (Nginx) เพิ่มคำสั่งนี้ในไฟล์ nginx.conf ภายในบล็อก server { ... } หรือ http { ... } add_header X-Content-Type-Options "nosniff" always; 3. สำหรับ Microsoft IIS สามารถเข้าไปที่หน้า IIS Manager เลือกเมนู "HTTP Response Headers" และเพิ่มตัวแปรชื่อ X-Content-Type-Options โดยกำหนดค่าเป็น nosniff		
ลำดับที่	13		
ชื่อของช่องโหว่	Information Disclosure - Suspicious Comments	**ระดับความเสี่ยง (Severity)**	Risk=Informational, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การเปิดเผยข้อมูลผ่านคอมเมนต์ที่น่าสงสัย
คำอธิบายโดยสังเขป	ในโค้ดฝั่งหน้าบ้าน (HTML, JavaScript, หรือ CSS) มีการทิ้งข้อความคอมเมนต์ของนักพัฒนาเอาไว้ ข้อความเหล่านั้นอาจมีคีย์เวิร์ดที่เข้าข่ายน่าสงสัย หากไม่ได้ลบออกก่อนนำระบบขึ้นใช้งานจริง ข้อความเหล่านี้จะถูกส่งมายังเว็บเบราว์เซอร์ และสามารถอ่านได้ง่ายๆ เพียงแค่กด "View Page Source" หรือ "Inspect Element"		
ข้อเสนอและแนวทางการแก้ไขช่องโหว่	1. การลบคำอธิบายที่ไม่จำเป็นออก ทีมพัฒนาควรตรวจสอบและลบคอมเมนต์ที่เกี่ยวข้องกับข้อมูลสำคัญ โครงสร้างระบบ รหัสผ่าน หรือข้อความระบุถึงช่องโหว่ที่รอการแก้ไข ออกจากโค้ดฝั่ง Client-side ทั้งหมดก่อนนำขึ้นสู่สภาพแวดล้อมจริง 2. การใช้กระบวนการ Minification แนะนำให้นำกระบวนการ Minify มาใช้ในขั้นตอนการสร้างระบบ เพื่อบีบอัดไฟล์ JavaScript และ CSS ซึ่งกระบวนการนี้จะทำการลบคอมเมนต์และช่องว่างทั้งหมดออกโดยอัตโนมัติ 3. การจัดการคอมเมนต์ในฝั่งเซิร์ฟเวอร์ (Server-side Comments) หากจำเป็นต้องมีคอมเมนต์เพื่ออธิบายโค้ด ควรเขียนคอมเมนต์ในภาษาฝั่งเซิร์ฟเวอร์ (เช่น // หรือ /* */ ใน PHP) เนื่องจากคอมเมนต์เหล่านี้จะถูกประมวลผลที่ฝั่งเซิร์ฟเวอร์ และไม่ถูกส่งต่อมาให้ผู้ใช้งานฝั่งเบราว์เซอร์เห็น		

ลำดับที่	14		
ชื่อของช่องโหว่	Modern Web Application	**ระดับความเสี่ยง (Severity)**	Risk=Informational, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	ตรวจพบสถาปัตยกรรมเว็บแอปพลิเคชันสมัยใหม่
คำอธิบายโดยสังเขป	เป็นแค่ข้อความจากเครื่องมือสแกนช่องโหว่ที่แจ้งให้ผู้ตรวจสอบทราบว่า ระบบที่กำลังสแกนอยู่นี้มีโครงสร้างแบบ "แอปพลิเคชันสมัยใหม่" ซึ่งมักจะหมายถึงการเป็น Single Page Application (SPA) หรือมีการใช้งานเทคโนโลยีฝั่งไคลเอนต์ เช่น JavaScript, AJAX หรือเฟรมเวิร์กอย่าง React, Angular, หรือ Vue.js อย่างหนักหน่วงในการประมวลผลและสร้างหน้าเว็บเพจ สาเหตุที่ Scanner ต้องแจ้งเตือนเรื่องนี้ เป็นเพราะสถาปัตยกรรมแบบนี้ทำให้เครื่องมือสแกนแบบดั้งเดิม ที่อ่านเฉพาะซอร์สโค้ด HTML เริ่มต้น ไม่สามารถค้นหาหน้าเว็บและจุดเชื่อมต่อ ทั้งหมดได้ Scanner รุ่นใหม่จึงต้องปรับกลไกไปใช้เบราว์เซอร์จำลอง (Headless Browser) เพื่อรันสคริปต์และจำลองพฤติกรรมผู้ใช้ในการวิเคราะห์โครงสร้างระบบ แม้จะไม่ใช่ช่องโหว่โดยตรง แต่การที่แอปพลิเคชันเป็น Modern Web App จะทำให้ Attack Surface เปลี่ยนรูปแบบไปจากเดิม		
ข้อเสนอและแนวทางการแก้ไขช่องโหว่	เนื่องจากไม่ใช่ช่องโหว่ ทีมพัฒนาจึงไม่ต้องทำการแก้ไขซอร์สโค้ดเพื่อปิดการแจ้งเตือนนี้ แต่มีข้อแนะนำเชิงนโยบายดังนี้ 1.ต้องตระหนักว่าการตรวจสอบข้อมูลนำเข้า (Input Validation) และการควบคุมสิทธิ์ (Access Control) จะต้องถูกบังคับใช้ที่เซิร์ฟเวอร์ (Server-side API) อย่างเข้มงวดเสมอ ห้ามพึ่งพาการตรวจสอบที่ฝั่งไคลเอนต์ (JavaScript) เพียงอย่างเดียว เพราะผู้โจมตีสามารถดัดแปลงหรือข้ามขั้นตอนบนเบราว์เซอร์ได้ 2. สำหรับผู้ตรวจสอบระบบ ในการประเมินและเปรียบเทียบเครื่องมือสแกนช่องโหว่ (Vulnerability Scanner Comparison) ความสามารถในการรองรับ Modern Web Application ถือเป็นหนึ่งในเกณฑ์ชี้วัดประสิทธิภาพที่สำคัญที่สุด ควรประเมินว่า Scanner แต่ละยี่ห้อสามารถรวบรวม (Crawl) และทดสอบ API Endpoints ที่ซ่อนอยู่หลัง JavaScript ได้ครบถ้วนและแม่นยำมากน้อยเพียงใด		
ลำดับที่	15		
ชื่อของช่องโหว่	Session Management Response Identified	**ระดับความเสี่ยง (Severity)**	Risk=Informational, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	ตรวจพบการตอบกลับที่เกี่ยวกับการจัดการเซสชัน

คำอธิบายโดยสังเขป	การแจ้งเตือนนี้เกิดขึ้นเมื่อเครื่องมือสแกนช่องโหว่ตรวจพบว่า เว็บเซิร์ฟเวอร์มีการส่ง HTTP Header ที่เกี่ยวข้องกับการสร้างหรือจัดการ Session กลับมาให้เบราว์เซอร์ ซึ่งโดยปกติจะมาในรูปแบบของคำสั่ง Set-Cookie เพื่อกำหนดค่า Session ID ระบบที่เป็น Web Application ส่วนใหญ่จำเป็นต้องมีกลไกนี้เพื่อจดจำสถานะของผู้ใช้งาน เช่น การจดจำว่าผู้ใช้ล็อกอินเข้าสู่ระบบแล้ว หรือจดจำข้อมูลชั่วคราวระหว่างการใช้งาน Scanner จึงทำการไฮไลต์จุดนี้ไว้เพื่อให้ผู้ประเมินระบบทราบว่า "นี่คือจุดเริ่มต้นของการจัดการเซสชัน" แม้ข้อความนี้จะไม่ใช่ช่องโหว่ แต่เป็นจุดที่บอกผู้โจมตี (หรือผู้ประเมินความปลอดภัย) ว่าระบบมีการใช้เซสชันตรงนี้ หากการจัดการเซสชันทำได้ไม่รัดกุมพอ จะนำไปสู่ช่องโหว่อื่นๆ ได้		
ข้อเสนอและแนวทางการแก้ไขช่องโหว่	เช่นเดียวกับการแจ้งเตือน Informational อื่นๆ ระบบไม่ต้องแก้ไขโค้ดเพื่อปิดการแจ้งเตือนนี้ แต่มีข้อแนะนำดังนี้ 1. การตั้งค่าความปลอดภัยของคุกกี้ (Secure Cookie Attributes) ตรวจสอบให้แน่ใจว่าคุกกี้ที่ใช้จัดการเซสชันทั้งหมดมีการเปิดใช้งานแอตทริบิวต์ Secure, HttpOnly, และ SameSite อย่างครบถ้วน เพื่อป้องกันการถูกขโมยผ่าน XSS หรือดักจับผ่านเครือข่ายที่ไม่ปลอดภัย 2. สำหรับการประเมินเครื่องมือสแกนช่องโหว่ (Scanner Evaluation) ในการจัดทำโครงงานเปรียบเทียบ Web Application Vulnerability Scanner ความสามารถในการวิเคราะห์ Session Management ถือเป็นตัวชี้วัดความฉลาดของ Scanner เพราะ Scanner ที่ดีควรจะสามารถล็อกอินเข้าสู่ระบบ เก็บค่า Session นี้ไว้ และนำไปใช้จำลองการโจมตี (Authenticated Scanning) ในหน้าเว็บส่วนลึกๆ ต่อไปได้		
ลำดับที่	16		
ชื่อของช่องโหว่	User Agent Fuzzer	**ระดับความเสี่ยง (Severity)**	Risk=Informational, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	การทดสอบช่องโหว่ด้วยการดัดแปลงค่า User-Agent
คำอธิบายโดยสังเขป	ในการส่ง HTTP Request เบราว์เซอร์จะมีการแนบส่วนหัวที่เรียกว่า User-Agent ไปด้วยเสมอ เพื่อบอกว่าผู้ใช้กำลังเข้าเว็บด้วยเบราว์เซอร์อะไรและระบบปฏิบัติการใด การที่เครื่องมือสแกนแจ้งเตือนหัวข้อนี้ หมายความว่าตัวเครื่องมือได้ทำการโจมตีแบบ Fuzzing เข้าไปในฟิลด์ User-Agent และพบว่าเซิร์ฟเวอร์มีการตอบสนองที่เปลี่ยนแปลงไปจากปกติ (เช่น ขนาดหน้าเว็บเปลี่ยนไป, โค้ด HTTP Status เปลี่ยน, หรือเกิด Error) แม้ว่าในเบื้องต้นอาจเป็นเพียงแค่ระบบพยายามปรับหน้าเว็บให้เข้ากับอุปกรณ์ หรือ API มีการตรวจสอบประเภทไคลเอนต์ แต่ในมุมมองด้านความปลอดภัย การเปิดรับค่า User-Agent โดยไม่ระมัดระวังอาจนำไปสู่ความเสี่ยงได้เช่นกัน		

ข้อเสนอและแนวทางการแก้ไขช่องโหว่	เช่นเดียวกับการแจ้งเตือน Informational อื่นๆ ระบบไม่ต้องแก้ไขโค้ดเพื่อปิดการแจ้งเตือนนี้ แต่มีข้อแนะนำดังนี้ 1. การตรวจสอบตรรกะของระบบ (Business Logic Review) ตรวจสอบว่า API ดังกล่าวมีความจำเป็นต้องเปลี่ยนพฤติกรรมการทำงานตามค่า User-Agent หรือไม่ หากไม่มีความจำเป็น ควรตั้งค่าให้ API ตอบสนองด้วยมาตรฐานเดียวกันทั้งหมด 2. การทำความสะอาดข้อมูลนำเข้า ต้องพึงระลึกเสมอว่า User-Agent คือข้อมูลนำเข้าที่ผู้ใช้สามารถควบคุมและแก้ไขได้ร้อยเปอร์เซ็นต์ ดังนั้น หากต้องนำค่านี้ไปประมวลผล บันทึกลง Log หรือแสดงผล จะต้องผ่านกระบวนการตรวจสอบและกรองอักขระพิเศษ (Validation & Encoding) เสมอ ห้ามเชื่อถือค่านี้โดยเด็ดขาด		
ลำดับที่	17		
ชื่อของช่องโหว่	User Controllable HTML Element Attribute (Potential XSS)	**ระดับความเสี่ยง (Severity)**	Risk=Informational, Confidence=Low
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด POST	**ประเภทช่องโหว่**	การควบคุมแอตทริบิวต์ HTML โดยผู้ใช้ / ความเสี่ยงต่อการเกิด XSS
คำอธิบายโดยสังเขป	แอปพลิเคชันมีการรับ Input จากผู้ใช้งานผ่านพารามิเตอร์ต่างๆ แล้วนำข้อมูลเหล่านั้นไปแสดงผลซ้ำ หรือแทรกลงในแอตทริบิวต์ของแท็ก HTML บนหน้าเว็บเพจ (ตัวอย่างเช่น <input type="text" name="username" value="[ข้อมูลจากผู้ใช้]">) โดยขาดกระบวนการคัดกรอง หรือการเข้ารหัสอักขระพิเศษที่รัดกุมเพียงพอ ช่องโหว่นี้อนุญาตให้ผู้ไม่ประสงค์ดีสามารถแทรกแซงโครงสร้าง HTML ได้โดยการส่งอักขระพิเศษ เช่น เครื่องหมายฟันหนู (") หรืออัญประกาศเดี่ยว (') เพื่อปิดแอตทริบิวต์เดิมของระบบ จากนั้นจึงทำการแทรกเหตุการณ์ (Event Handlers เช่น onmouseover หรือ onerror) หรือโค้ดสคริปต์ JavaScript อันตรายเข้าไปทำงานบนหน้าเว็บเบราว์เซอร์ของผู้ใช้งานคนอื่นๆ ได้		

ข้อเสนอและแนวทางการแก้ไขช่องโหว่	เพื่อปิดกั้นช่องโหว่ XSS อย่างมีประสิทธิภาพ ขอเสนอแนวทางปฏิบัติ ดังนี้ 1. การเข้ารหัสข้อมูลก่อนแสดงผล กฎเหล็กในการป้องกันช่องโหว่นี้คือ ก่อนนำข้อมูลที่รับจากผู้ใช้ไปแสดงผลในแอตทริบิวต์ HTML ระบบจะต้องทำการแปลงอักขระพิเศษทั้งหมด (เช่น <, >, ", ', &) ให้เป็นรูปแบบ HTML Entities เสมอ ตัวอย่างการแก้ไขในภาษา PHP ควรใช้ฟังก์ชัน htmlspecialchars() พร้อมกับกำหนดแฟล็ก ENT_QUOTES เพื่อให้ระบบแปลงทั้งเครื่องหมายฟันหนูและอัญประกาศเดี่ยว $safe_value = htmlspecialchars($_POST['user_input'], ENT_QUOTES, 'UTF-8'); echo '<input type="text" name="username" value="' . $safe_value . '">'; 2. การตรวจสอบความถูกต้องของข้อมูลนำเข้า แอปพลิเคชันควรใช้กลไกการตรวจสอบข้อมูลแบบ Whitelist โดยอนุญาตเฉพาะรูปแบบตัวอักษรที่ถูกต้องตามบริบทเท่านั้น (เช่น ชื่อผู้ใช้ควรมีแค่ตัวอักษรและตัวเลข) หากพบอักขระที่ไม่เกี่ยวข้อง ระบบควรปฏิเสธการประมวลผลทันที 3. การบังคับใช้นโยบายความปลอดภัยเชิงลึก ทีมพัฒนาควรดำเนินการตั้งค่า HTTP Header Content-Security-Policy (สอดคล้องกับช่องโหว่ CSP ที่พบก่อนหน้านี้) เพื่อเป็นเกราะป้องกันชั้นที่สอง โดยจำกัดไม่ให้เบราว์เซอร์ทำการรันสคริปต์แบบ Inline (script-src ไม่ควรมี 'unsafe-inline') ซึ่งจะช่วยลดทอนผลกระทบจากช่องโหว่ XSS ได้อย่างมีนัยสำคัญ

แนวทางการแก้ไขช่องโหว่ความมั่นคงปลอดภัยที่พบได้บ่อย

แนวทางการแก้ไขช่องโหว่ความมั่นคงปลอดภัยเพิ่มเติมจำนวน 10 รายการ ที่มักตรวจพบในระบบขององค์กร (รวมถึงเคยตรวจพบแล้ว) เพื่อใช้เป็นฐานข้อมูลความรู้ และมาตรฐานการปฏิบัติงานเพิ่มเติม สำหรับนักพัฒนาในการป้องกันและรับมือกับภัยคุกคามเชิงรุก

ลำดับที่	1		
ชื่อของช่องโหว่	Directory Traversal (Path Traversal)	**ระดับความเสี่ยง (Severity)**	Risk=High, Confidence=Low
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	File Inclusion
คำอธิบายโดยสังเขป	ระบบอนุญาตให้ผู้ใช้ระบุชื่อไฟล์ที่จะดาวน์โหลดได้โดยตรง ทำให้ผู้โจมตีใส่ค่า ../../../../etc/passwd เพื่อไปอ่านไฟล์ลับของระบบปฏิบัติการได้		
วิธีการแก้ไขช่องโหว่	ห้ามใช้ชื่อไฟล์จากผู้ใช้โดยตรง, ใช้ฟังก์ชัน basename() และตรวจสอบว่าไฟล์นั้นอยู่ในโฟลเดอร์ที่กำหนดเท่านั้น (White-list)		

ตัวอย่างการแก้ไขช่องโหว่	ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท PHP
	--- อันตราย (Vulnerable) ---
	$path = "/var/www/uploads/" . $_GET['file_name'];
	readfile($path);
	--- ปลอดภัย (Secure) ---
	$user_input = $_GET['file_name'];
	1. ดึงเฉพาะชื่อไฟล์ออกมา ป้องกันการใช้ ../ หรือ /
	$safe_file = basename($user_input);
	$base_path = "/var/www/uploads/";
	$full_path = $base_path . $safe_file;
	2. ตรวจสอบว่าไฟล์มีอยู่จริงในโฟลเดอร์ที่กำหนดเท่านั้น
	if (file_exists($full_path) && is_file($full_path)) {
	readfile($full_path);
	} else {
	die("Error: ไม่พบไฟล์ที่ระบุ");
	}		

ลำดับที่	2		

ชื่อของช่องโหว่	CSP: Failure to Define Directive with No Fallback	**ระดับความเสี่ยง (Severity)**	Risk=Medium, Confidence=High
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด GET	**ประเภทช่องโหว่**	Security Misconfiguration ในระดับ Web Server
คำอธิบายโดยสังเขป	การไม่กำหนด default-src หรือ Directive พื้นฐานใน CSP ทำให้เบราว์เซอร์ไม่มีนโยบายสำรอง (Fallback) ส่งผลให้นักโจมตีสามารถโหลดสคริปต์หรือเนื้อหาอันตรายจากภายนอกได้ (XSS, Data Exfiltration)		
วิธีการแก้ไขช่องโหว่	กำหนดนโยบาย default-src 'self' เพื่อจำกัดสิทธิ์พื้นฐานให้โหลดได้เฉพาะจากโดเมนตัวเอง และระบุเฉพาะแหล่งที่มาที่เชื่อถือได้เท่านั้น		

ตัวอย่างการแก้ไขช่องโหว่	ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท PHP
	/* --- อันตราย (Vulnerable) --- */
	// ไม่มีการกำหนด CSP หรือกำหนดไม่ครอบคลุม
	header("Content-Security-Policy: script-src 'self' https://trusted.com");
	/* --- ปลอดภัย (Secure) --- */
	// กำหนด default-src เป็น 'self' เพื่อป้องกันทุก Directive ที่ไม่ได้ระบุ
	header("Content-Security-Policy: default-src 'self'; script-src 'self' https://trusted.com; object-src 'none';");		

ลำดับที่	3		
ชื่อของช่องโหว่	Spring4Shell	**ระดับความเสี่ยง (Severity)**	Risk=High, Confidence=Medium
พารามิเตอร์ที่พบปัญหา	ผ่านเมธอด POST	**ประเภทช่องโหว่**	Remote Code Execution (RCE)
คำอธิบายโดยสังเขป	ช่องโหว่นี้เกิดจากการที่ระบบอนุญาตให้ผู้โจมตีส่งพารามิเตอร์พิเศษเพื่อเข้าไปแก้ไขคุณสมบัติของ Class Loader ผ่านการทำ Data Binding ของ Framework ส่งผลให้ผู้โจมตีสามารถสั่งเขียนไฟล์ JSP อันตราย (Web Shell) ลงบนเซิร์ฟเวอร์และรันคำสั่งควบคุมระบบจากระยะไกลได้		
วิธีการแก้ไขช่องโหว่	วิธีที่ดีที่สุดคืออัปเกรด Spring Framework เป็นเวอร์ชัน 5.3.18 หรือ 5.2.20 ขึ้นไป และอัปเกรด Spring Boot เป็นเวอร์ชัน 2.6.6 หรือ 2.5.12 ขึ้นไป แต่หากยังไม่สามารถอัปเกรดได้ ให้ตั้งค่าการทำความสะอาดข้อมูล (Data Sanitization) โดยใช้ WebDataBinder เพื่อจำกัดฟิลด์ที่ไม่ได้รับอนุญาต (Disallow-list)		

<table>
<tr><td>ตัวอย่างการแก้ไขช่องโหว่</td><td colspan="3">ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท Java
/* --- อันตราย (Vulnerable) --- */
// โค้ดที่ไม่มีการจำกัดฟิลด์ในการทำ Data Binding ทำให้ ClassLoader ถูกแก้ไขได้
@PostMapping("/user")
public String update(User user) {
   return "success"; }
/* --- ปลอดภัย (Secure) --- */
// การใช้ @InitBinder เพื่อกำหนด Disallow-list ป้องกันการเข้าถึง ClassLoader
@ControllerAdvice
public class SecurityAdvice {
   @InitBinder
   public void setAllowedFields(WebDataBinder dataBinder) {
     // รายการฟิลด์ที่ต้องห้ามเพื่อป้องกัน Spring4Shell
     String[] denylist = new String[]{"class.*", "Class.*", "*.class.*", "*.Class.*"};
     dataBinder.setDisallowedFields(denylist);
   }
}</td></tr>
<tr><td>ลำดับที่</td><td colspan="3">4</td></tr>
<tr><td>ชื่อของช่องโหว่</td><td>Hidden File Found</td><td>ระดับความเสี่ยง (Severity)</td><td>Risk=Medium, Confidence=High</td></tr>
<tr><td>พารามิเตอร์ที่พบปัญหา</td><td>ผ่านเมธอด GET</td><td>ประเภทช่องโหว่</td><td>Security Misconfiguration</td></tr>
<tr><td>คำอธิบายโดยสังเขป</td><td colspan="3">เป็นช่องโหว่ที่เกิดจากการปล่อยให้Configuration Files, Backup Files, หรือไฟล์ Development Metadata สามารถเข้าถึงได้ผ่านอินเทอร์เน็ตโดยตรง ไฟล์เหล่านี้มักบรรจุข้อมูลอ่อนไหว เช่น Database Credentials, API Keys, หรือโครงสร้างซอร์สโค้ด ซึ่งช่วยให้ผู้โจมตีสามารถขยายผลการโจมตีไปยังส่วนอื่นๆ ของระบบได้ง่ายขึ้น</td></tr>
<tr><td>วิธีการแก้ไขช่องโหว่</td><td colspan="3">ลบไฟล์ที่ไม่จำเป็น ตรวจสอบและลบไฟล์สำรอง (.bak, .old, .zip) หรือไฟล์ทดสอบออกจาก Production Environment ทันที กำหนดค่าการเข้าถึงไฟล์ในระดับ Server Config (เช่น .htaccess หรือ nginx.conf) เพื่อปฏิเสธการเข้าถึงไฟล์ที่ขึ้นต้นด้วยจุด (.) หรือไฟล์นามสกุลที่สุ่มเสี่ยง และควบคุมการนำไฟล์ขึ้นระบบผ่าน Git โดยระบุไฟล์ที่ไม่ต้องการให้เผยแพร่ใน .gitignore</td></tr>
</table>

<table>
<tr><td>ตัวอย่างการแก้ไขช่องโหว่</td><td colspan="3">ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท Nginx
/* --- อันตราย (Vulnerable) --- */
// เว็บเซิร์ฟเวอร์อนุญาตให้เข้าถึงทุกไฟล์ในโฟลเดอร์ที่เปิดใช้งาน
location / {
    try_files $uri $uri/ =404;
}
/* --- ปลอดภัย (Secure) --- */
// การตั้งค่าใน Nginx เพื่อบล็อกการเข้าถึงไฟล์ .env และโฟลเดอร์ .git
location ~ /\.(env|git) {
    deny all;
    return 404;
}
// ปฏิเสธการเข้าถึงไฟล์สำรองและไฟล์ที่ใช้ทดสอบระบบ
location ~ \.(bak|config|sql|phpinfo|old|save)$ {
    deny all;
    access_log off;
    log_not_found off;
}</td></tr>
<tr><td>ลำดับที่</td><td colspan="3">5</td></tr>
<tr><td>ชื่อของช่องโหว่</td><td>Missing Anti-clickjacking Header</td><td>ระดับความเสี่ยง (Severity)</td><td>Risk=Medium, Confidence=Medium</td></tr>
<tr><td>พารามิเตอร์ที่พบปัญหา</td><td>ผ่านเมธอด GET)</td><td>ประเภทช่องโหว่</td><td>Response Headers</td></tr>
<tr><td>คำอธิบายโดยสังเขป</td><td colspan="3">เป็นช่องโหว่ที่เกิดขึ้นเมื่อเว็บแอปพลิเคชันไม่ได้ส่งคำสั่งใน Header เพื่อระบุว่าเบราว์เซอร์ควรอนุญาตให้เว็บไซต์นี้ถูกนำไปแสดงผลใน <iframe>, <frame> หรือ <object> หรือไม่ หากไม่ได้ป้องกันไว้ ผู้โจมตีสามารถสร้างเว็บไซต์ปลอมแล้วดึงหน้าเว็บจริงไปซ้อนทับด้วยเลเยอร์โปร่งใส (Transparent Layer) เพื่อล่อลวงให้ผู้ใช้คลิกปุ่มในหน้าเว็บจริงโดยไม่รู้ตัว</td></tr>
<tr><td>วิธีการแก้ไขช่องโหว่</td><td colspan="3">ส่ง HTTP Header X-Frame-Options: SAMEORIGIN เพื่ออนุญาตให้แสดงผลภายใน iframe ได้เฉพาะในโดเมนเดียวกันเท่านั้น หรือใช้ Content Security Policy (CSP) โดยระบุ Directive frame-ancestors 'self' ซึ่งมีความยืดหยุ่นและเป็นมาตรฐานใหม่ที่ปลอดภัยกว่า</td></tr>
</table>

<table>
<tr><td>ตัวอย่างการแก้ไขช่องโหว่</td><td colspan="3">ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท PHP
/* --- อันตราย (Vulnerable) --- */
// ไม่มีการส่ง Header ป้องกัน ทำให้เว็บถูกนำไปใส่ใน iframe ของใครก็ได้
echo "<h1>Welcome to Transaction Page</h1>";
/* --- ปลอดภัย (Secure) --- */
// วิธีที่ 1: ใช้ X-Frame-Options (รองรับเบราว์เซอร์เก่า)
header("X-Frame-Options: SAMEORIGIN");
// วิธีที่ 2: ใช้ CSP frame-ancestors (มาตรฐานปัจจุบัน)
header("Content-Security-Policy: frame-ancestors 'self';");
echo "<h1>Welcome to Transaction Page (Protected)</h1>";</td></tr>
<tr><td>ลำดับที่</td><td colspan="3">6</td></tr>
<tr><td>ชื่อของช่องโหว่</td><td>Cross-Domain Misconfiguration</td><td>ระดับความเสี่ยง (Severity)</td><td>Risk=Medium, Confidence=Medium</td></tr>
<tr><td>พารามิเตอร์ที่พบปัญหา</td><td>ผ่านเมธอด GET</td><td>ประเภทช่องโหว่</td><td>Security Misconfiguration</td></tr>
<tr><td>คำอธิบายโดยสังเขป</td><td colspan="3">เกิดจากการที่เซิร์ฟเวอร์ถูกตั้งค่าให้นโยบายการแชร์ทรัพยากรข้ามโดเมน (CORS) ผ่อนปรนเกินไป เช่น การใช้เครื่องหมายดอกจัน (*) เพื่ออนุญาตทุกโดเมน หรือการสะท้อนค่าโดเมนจาก Header Origin ของผู้ใช้กลับไปโดยไม่มีการตรวจสอบ ทำให้เว็บไซต์อันตรายสามารถเขียนสคริปต์มาดึงข้อมูลสำคัญ ของผู้ใช้ที่ล็อกอินอยู่ในระบบได้</td></tr>
<tr><td>วิธีการแก้ไขช่องโหว่</td><td colspan="3">กำหนดรายการอนุญาต (Whitelist) ระบุเฉพาะโดเมนที่เชื่อถือได้เท่านั้นใน Header Access-Control-Allow-Origin แทนการใช้ * และหลีกเลี่ยงการใช้ Credentials ร่วมกับ Wildcard หากจำเป็นต้องส่งคุกกี้ (Access-Control-Allow-Credentials: true) ห้ามใช้ค่า * ในส่วนของ Origin โดยเด็ดขาด
ตรวจสอบค่า Origin หากมีหลายโดเมนที่ต้องอนุญาต ให้ใช้การตรวจสอบในฝั่งโปรแกรมเมอร์เทียบกับรายการโดเมนที่อนุญาต (Whitelist) ก่อนส่ง Header กลับไป</td></tr>
</table>

<table>
<tr><td>ตัวอย่างการแก้ไขช่องโหว่</td><td colspan="3">ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท PHP
/* --- อันตราย (Vulnerable) --- */
// อนุญาตทุกโดเมน (Wildcard) เสี่ยงต่อการถูกขโมยข้อมูลผ่านเว็บอื่น
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Credentials: true");
/* --- ปลอดภัย (Secure) --- */
1. กำหนดรายการโดเมนที่อนุญาต
$allowed_origins = ['https://www.mywebsite.com',
'https://api.mywebsite.com'];
$origin = $_SERVER['HTTP_ORIGIN'] ?? '';
2. ตรวจสอบว่า Origin ที่เรียกมาอยู่ในรายการที่อนุญาตหรือไม่
if (in_array($origin, $allowed_origins)) {
header("Access-Control-Allow-Origin: " . $origin);
header("Access-Control-Allow-Credentials: true");
} else {
// ถ้าไม่อยู่ในรายการ ให้ปฏิเสธหรือส่ง Origin พื้นฐานกลับไป
header("Access-Control-Allow-Origin: https://www.mywebsite.com");
}</td></tr>
<tr><td>ลำดับที่</td><td colspan="3">7</td></tr>
<tr><td>ชื่อของช่องโหว่</td><td>PII Disclosure</td><td>ระดับความเสี่ยง (Severity)</td><td>Risk=High, Confidence=High</td></tr>
<tr><td>พารามิเตอร์ที่พบปัญหา</td><td>ผ่านเมธอด GET</td><td>ประเภทช่องโหว่</td><td>Sensitive Data Exposure</td></tr>
<tr><td>คำอธิบายโดยสังเขป</td><td colspan="3">เป็นช่องโหว่ที่เกิดจากการที่แอปพลิเคชันส่งข้อมูลส่วนบุคคลที่ระบุตัวตนได้ (เช่น เลขประจำตัวประชาชน, ข้อมูลทางการเงิน, ที่อยู่, หรือเบอร์โทรศัพท์) ออกมาให้ผู้ใช้เข้าถึงได้โดยไม่จำเป็น หรือมีการส่งข้อมูลออกมามากเกินความจำเป็นในการแสดงผล เช่น API ส่งข้อมูลผู้ใช้มาทั้งแถวในฐานข้อมูลแทนที่จะส่งแค่ชื่อและนามสกุล ข้อมูลเหล่านี้อาจถูกผู้โจมตีนำไปใช้สวมรอย หรือนำไปสู่การละเมิดความเป็นส่วนตัวอย่างร้ายแรง</td></tr>
<tr><td>วิธีการแก้ไขช่องโหว่</td><td colspan="3">ออกแบบ API ให้ส่งกลับเฉพาะฟิลด์ข้อมูลที่ต้องใช้งานจริงในหน้านั้นๆ เท่านั้น แสดงข้อมูลเพียงบางส่วนสำหรับข้อมูลอ่อนไหว เช่น แสดงเลขบัตรประชาชนเฉพาะ 4 ตัวท้าย หรือการตรวจสอบทุกครั้งว่าผู้ที่เรียกขอข้อมูลมีสิทธิ์เข้าถึงข้อมูลของบุคคลนั้นๆ จริงหรือไม่</td></tr>
</table>

<table>
<tr><td>ตัวอย่างการแก้ไขช่องโหว่</td><td colspan="3">ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท PHP
/* --- อันตราย (Vulnerable) --- */
// ดึงข้อมูลทั้งหมดจากตารางและส่งออกไปผ่าน API โดยไม่มีการคัดกรอง
$user = $db->query("SELECT * FROM users WHERE id = " . $_GET['id'])-
>fetch();
echo json_encode($user); // ข้อมูลรั่วไหล เช่น password_hash, ssn, phone
/* --- ปลอดภัย (Secure) --- */
1. เลือกเฉพาะฟิลด์ที่จำเป็นต้องใช้แสดงผลจริง (ป้องกัน Over-fetching)
$stmt = $db->prepare("SELECT username, email, phone FROM users
WHERE id = ?");
$stmt->execute([$_GET['id']]);
$user = $stmt->fetch();
2. ทำ Data Masking สำหรับข้อมูลเบอร์โทรศัพท์ก่อนแสดงผล
if ($user) {
$user['phone'] = substr($user['phone'], 0, 3) . '-XXXX-' .
substr($user['phone'], -3);
}
echo json_encode($user);</td></tr>
<tr><td>ลำดับที่</td><td colspan="3">8</td></tr>
<tr><td>ชื่อของช่องโหว่</td><td>Cloud Metadata Potentially Exposed</td><td>ระดับความเสี่ยง (Severity)</td><td>Risk=High, Confidence=Medium</td></tr>
<tr><td>พารามิเตอร์ที่พบปัญหา</td><td>ผ่านเมธอด GET</td><td>ประเภทช่องโหว่</td><td>Information Disclosure</td></tr>
<tr><td>คำอธิบายโดยสังเขป</td><td colspan="3">Cloud Providers จะมีบริการ Metadata Service ที่ไอพีเฉพาะ เพื่อให้เซิร์ฟเวอร์ดึงข้อมูลคอนฟิกของตัวเองได้ หากแอปพลิเคชันมีช่องโหว่ที่ยอมให้ดึงข้อมูลจากไอพีนี้ออกมาแสดงผลได้ แฮกเกอร์จะสามารถขโมยข้อมูลสำคัญ เช่น IAM Role Credentials (Access Keys/Secret Keys), ข้อมูลเครือข่ายภายใน หรือรหัสผ่านชั่วคราว ซึ่งนำไปสู่การเข้ายึดครองระบบคลาวด์ทั้งหมดได้</td></tr>
<tr><td>วิธีการแก้ไขช่องโหว่</td><td colspan="3">ใช้ IMDSv2 (สำหรับ AWS) บังคับใช้ระบบ Session-oriented (Token-based) เพื่อป้องกันการดึงข้อมูลผ่าน SSRF แบบธรรมดา
ตั้งค่า Network Firewall/WAF บล็อกการเชื่อมต่อขาออก (Outbound) จากแอปพลิเคชันไปยังไอพี 169.254.169.254 หากไม่มีความจำเป็นต้องใช้งาน
จำกัดสิทธิ์ (Least Privilege) กำหนดสิทธิ์ให้ IAM Role ของเซิร์ฟเวอร์มีสิทธิ์เฉพาะเท่าที่จำเป็นที่สุด เพื่อลดผลกระทบหากคีย์หลุด</td></tr>
</table>

<table>
<tr><td>ตัวอย่างการแก้ไขช่องโหว่</td><td colspan="3">ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท PHP
/* --- อันตราย (Vulnerable) --- */
// แอปพลิเคชันรับ URL จากผู้ใช้ไปดึงข้อมูลโดยไม่มีการตรวจสอบไอพีปลายทาง
$url = $_GET['url'];
$data = file_get_contents($url); // ผู้โจมตีใส่ http://169.254.169.254/latest/meta-data/ เพื่อขโมยคีย์
echo $data;
/* --- ปลอดภัย (Secure) --- */
1. ตรวจสอบและบล็อกไอพีภายใน (Internal/Metadata IP)
$url = $_GET['url'];
$parsed_url = parse_url($url);
$host = $parsed_url['host'];
รายชื่อไอพีที่ต้องห้าม (Deny-list)
$forbidden_ips = ['169.254.169.254', '127.0.0.1', 'localhost'];
if (in_array($host, $forbidden_ips)) {
die("Error: ไม่อนุญาตให้เข้าถึงที่อยู่ภายในระบบ");
}
2. แนะนำให้ใช้การเรียกข้อมูลผ่าน Proxy ที่มีการทำ Filter อย่างเข้มงวดแทนการใช้ file_get_contents โดยตรง</td></tr>
<tr><td>ลำดับที่</td><td colspan="3">9</td></tr>
<tr><td>ชื่อของช่องโหว่</td><td>Server-Side Template Injection</td><td>ระดับความเสี่ยง (Severity)</td><td>Risk=High, Confidence=High</td></tr>
<tr><td>พารามิเตอร์ที่พบปัญหา</td><td>ผ่านเมธอด POST</td><td>ประเภทช่องโหว่</td><td>Application Logic</td></tr>
<tr><td>คำอธิบายโดยสังเขป</td><td colspan="3">เกิดจากการที่แอปพลิเคชันรับข้อมูลจากผู้ใช้แล้วนำไปรวมกับสตริงของเทมเพลต (Template String) โดยตรง แทนที่จะส่งเป็นข้อมูล (Data) ทำให้ผู้โจมตีสามารถส่งสัญลักษณ์พิเศษของเทมเพลตนั้น ๆ (เช่น {{ ... }} หรือ ${ ... }) เพื่อสั่งให้เซิร์ฟเวอร์คำนวณหรือรันคำสั่งภายในระบบปฏิบัติการ (RCE) ได้</td></tr>
<tr><td>วิธีการแก้ไขช่องโหว่</td><td colspan="3">ห้ามต่อสตริงในเทมเพลตโดยส่งข้อมูลผ่านตัวแปร (Variables) หรือพารามิเตอร์ที่เทมเพลตเตรียมไว้ให้เท่านั้น
ใช้ Sandbox ถ้าหากจำเป็นต้องให้ผู้ใช้แก้ไขเทมเพลตเอง ให้รันในสภาพแวดล้อมที่จำกัดสิทธิ์เท่านั้น</td></tr>
</table>

<table>
<tr><td></td><td colspan="3">Input Validation ตรวจสอบและกรองอักขระพิเศษที่เกี่ยวข้องกับคำสั่งเทมเพลตออกก่อนประมวลผล</td></tr>
<tr><td>ตัวอย่างการแก้ไขช่องโหว่</td><td colspan="3">ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท Pytron
/* --- อันตราย (Vulnerable) --- */
# นำ input จากผู้ใช้มาสร้างเป็น String ของเทมเพลตโดยตรง
from flask import render_template_string, request
@app.route('/hello')
def hello():
name = request.args.get('name', 'Guest')
# อันตราย: ผู้โจมตีใส่ {{ 7*7 }} จะได้ผลลัพธ์เป็น 49 หรือใส่คำสั่งรันระบบได้
template = f"<h1>Hello {name}!</h1>"
return render_template_string(template)
/* --- ปลอดภัย (Secure) --- */
ส่งข้อมูลผ่านตัวแปรของเทมเพลต (Context Variables)
@app.route('/hello')
def hello():
name = request.args.get('name', 'Guest')
# ปลอดภัย: Jinja2 จะมอง name เป็นแค่ข้อความธรรมดา ไม่ประมวลผลเป็นคำสั่ง
return render_template_string("<h1>Hello {{ name }}!</h1>",
name=name)</td></tr>
<tr><td>ลำดับที่</td><td colspan="3">10</td></tr>
<tr><td>ชื่อของช่องโหว่</td><td>Format String Error</td><td>ระดับความเสี่ยง (Severity)</td><td>Risk=Medium, Confidence=Medium</td></tr>
<tr><td>พารามิเตอร์ที่พบปัญหา</td><td>ผ่านเมธอด POST</td><td>ประเภทช่องโหว่</td><td>Input Validation</td></tr>
<tr><td>คำอธิบายโดยสังเขป</td><td colspan="3">เกิดจากการที่แอปพลิเคชันรับข้อมูลจากผู้ใช้แล้วนำไปใช้เป็น "ตัวกำหนดรูปแบบ" (Format String) โดยตรง แทนที่จะใช้เป็นเพียง "ข้อมูล" (Data) ทำให้ผู้โจมตีสามารถใส่สัญลักษณ์พิเศษ เช่น %x เพื่ออ่านข้อมูลจาก Stack หรือ %n เพื่อเขียนทับค่าในหน่วยความจำ ส่งผลให้ข้อมูลรั่วไหลหรือสามารถควบคุมการทำงานของโปรแกรมได้</td></tr>
<tr><td>วิธีการแก้ไขช่องโหว่</td><td colspan="3">ห้ามใช้ Input เป็น Format String ให้กำหนด Format String เป็นค่าคงที่ (Static String) เสมอ และส่งข้อมูลจากผู้ใช้เป็นพารามิเตอร์แยกต่างหาก
ใช้ฟังก์ชันที่ปลอดภัย เลือกใช้ฟังก์ชันที่มีการตรวจสอบขอบเขตของข้อมูล (Boundary Checking)</td></tr>
</table>

	Input Filtering ต้องตรวจสอบและกรองอักขระพิเศษที่เป็นสัญลักษณ์ของการจัดรูปแบบข้อความออกก่อนนำไปประมวลผล

ตัวอย่างการแก้ไขช่องโหว่	ตัวอย่างการแก้ไขในไฟล์ระบบ ประเภท ภาษา C เนื่องจากช่องโหว่นี้มักเกิดขึ้นใน Format เก่า
	/* --- อันตราย (Vulnerable) --- */
	// รับค่าจากผู้ใช้แล้วนำไปพิมพ์โดยตรงผ่าน printf
	void print_user_input(char *user_input) {
	// อันตราย: หาก user_input คือ "%x %x %x" จะเป็นการดึงข้อมูลจาก Stack ออกมาโชว์
	printf(user_input);
	}
	/* --- ปลอดภัย (Secure) --- */
	// กำหนดรูปแบบเป็นข้อความธรรมดา (%s) แล้วส่ง input เป็นพารามิเตอร์
	void print_user_input(char *user_input) {
	// ปลอดภัย: %s จะทำให้โปรแกรมมองทุกอย่างใน user_input เป็นเพียงตัวอักษรปกติ
	printf("%s", user_input);
	}

ภาคผนวก ข

เอกสารหลักฐานที่เกี่ยวข้อง



ที่ อว ๗๑๐๗/ ๐๒๔

คณะเทคโนโลยีสารสนเทศและนวัตกรรมดิจิทัล
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
๑๕๑๘ ถนนประชาราษฎร์ ๑ แขวงวงศ์สว่าง
เขตบางซื่อ กทม. ๑๐๘๐๐

๑๒ มกราคม ๒๕๖๙

เรื่อง ขอความอนุเคราะห์เก็บรวบรวมข้อมูลเพื่อประกอบการทำการค้นคว้าอิสระ

เรียน รองคณบดีฝ่ายสารสนเทศ

เนื่องด้วย นางสาวณัฐธิดา ลูกเสือ รหัสประจำตัว ๖๗-๐๗๐๓๑๘-๕๗๑๘-๒ นักศึกษาระดับบัณฑิตศึกษา ภาควิชาการบริหารเครือข่ายดิจิทัลและความมั่นคงปลอดภัยสารสนเทศ คณะเทคโนโลยีสารสนเทศและนวัตกรรมดิจิทัล มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ กำลังดำเนินการจัดทำการค้นคว้าอิสระ เรื่อง “การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข (Performance comparison of web application vulnerability scanning tools and patch validation processes)” โดยมี ผู้ช่วยศาสตราจารย์ ดร.สุนันฑา สดสี เป็นประธานคณะกรรมการที่ปรึกษาการค้นคว้าอิสระ ในการนี้ นักศึกษามีความประสงค์ขอความอนุเคราะห์ในการเก็บรวบรวมข้อมูลที่เกี่ยวข้องกับเรื่องดังกล่าว เพื่อประกอบการทำการค้นคว้าอิสระต่อไป

จึงเรียนมาเพื่อโปรดพิจารณาให้ความอนุเคราะห์ จักขอบพระคุณยิ่ง

ขอแสดงความนับถือ

(ผู้ช่วยศาสตราจารย์ ดร.ศักดิ์ชาย ตั้งวรรณวิทย์)
รองคณบดีฝ่ายวิชาการและวิจัย
ปฏิบัติการแทนคณบดีคณะเทคโนโลยีสารสนเทศและนวัตกรรมดิจิทัล

ภาควิชาการบริหารเครือข่ายดิจิทัลและความมั่นคงปลอดภัยสารสนเทศ
โทร. ๐ ๒๕๕๕ ๒๗๑๗
โทรสาร ๐ ๒๕๘๗ ๕๘๘๐
www.itd.kmutnb.ac.th

ประวัติผู้เขียน

ชื่อ	นางสาวณัฐธิดา ลูกเสือ
ชื่อการค้นคว้าอิสระ	การเปรียบเทียบประสิทธิภาพเครื่องมือสแกนช่องโหว่เว็บแอปพลิเคชันและกระบวนการยืนยันผลการแก้ไข
สาขาวิชา	การบริหารเครือข่ายดิจิทัลและความมั่นคงปลอดภัยสารสนเทศ
ประวัติ	สำเร็จการศึกษาระดับปริญญาตรี วิทยาศาสตรบัณฑิต คณะวิทยาศาสตร์ สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏสวนสุนันทา ปีการศึกษา 2558 ประสบการณ์ทำงาน ปฏิบัติงานในตำแหน่ง นักวิชาการคอมพิวเตอร์ ทีม Security สังกัดงานเทคนิคและปฏิบัติการ ฝ่ายสารสนเทศ คณะแพทย์ศาสตร์ศิริราชพยาบาล มหาวิทยาลัยมหิดล พ.ศ. 2567 - ปัจจุบัน ปฏิบัติงานในตำแหน่ง เจ้าหน้าที่ระบบคอมพิวเตอร์ ฝ่ายสารสนเทศ สถาบันอาหาร สังกัดกระทรวงอุตสาหกรรม พ.ศ. 2560 – 2566

พระจอมเกล้าพระนครเหนือ มหาวิทยาลัยเทคโนโลยี