การทดสอบเจาะระบบและป้องกันการโจมตีเว็บแอปพลิเคชันด้วยคาลีลินุกซ์

พิชชา ภัทรเนรมิต

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตร์มหาบัณฑิต
บัณฑิตศึกษา สาขาเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีไทย-ญี่ปุ่น
ปีการศึกษา 2568

PENETRATION TESTING AND PROTECTION AGAINST WEB APPLICATION ATTACKS USING KALI LINUX

Picha Patraneramit

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Information Technology

Graduate Studies

Thai-Nichi Institute of Technology

Academic Year 2025

หัวข้อวิทยานิพนธ์	การทดสอบเจาะระบบและป้องกันการโจมตีเว็บแอปพลิเคชันด้วยคาลีลินุกซ์
โดย	พิชชา ภัทรเนรมิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ดร.ณปภัช วิชัยดิษฐ์

บัณฑิตศึกษา สถาบันเทคโนโลยีไทย-ญี่ปุ่น อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็น ส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....รองอธิการบดีฝ่ายวิชาการ
(รองศาสตราจารย์ ดร.วรากร ศรีเชวงทรัพย์)
วันที่.....เดือน.....พ.ศ.....

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.อรรณพ หมั่นสกุล)

.....กรรมการ
(ดร.ประมุข บุญเสี่ยง)

.....กรรมการ
(ดร.ศรายุทธ นนท์ศิริ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์
(ดร.ณปภัช วิชัยดิษฐ์)

พิชชา ภัทรเนรมิต : การทดสอบเจาะระบบและป้องกันการโจมตีเว็บแอปพลิเคชันด้วยคาลีลินุกซ์. อาจารย์ที่ปรึกษา : ดร.ณปภัช วิชัยดิษฐ์, 59 หน้า.

ปัจจุบันการใช้เว็บแอปพลิเคชันมีการขยายตัวอย่างรวดเร็ว และถูกนำมาใช้ในหลากหลายกระบวนการทำงานขององค์กร โดยเฉพาะในภาคการแพทย์ที่ต้องบริหารจัดการข้อมูลผู้ป่วยซึ่งมีความละเอียดอ่อนและมีความเสี่ยงสูงต่อการถูกโจมตีทางไซเบอร์ การศึกษาพบว่าเว็บแอปพลิเคชันกว่าร้อยละ 70 มีการใช้ฐานข้อมูล SQL เป็นโครงสร้างหลักในการจัดเก็บข้อมูล ซึ่งเพิ่มโอกาสที่ข้อมูลผู้ป่วยจำนวนมากอาจถูกโจรกรรม เผยแพร่ เรียกร้องค่าไถ่ หรือถูกนำไปใช้ในทางที่ผิด หากองค์กรไม่สามารถป้องกันช่องโหว่ได้อย่างมีประสิทธิภาพ ย่อมส่งผลให้เกิดความเสียหายทั้งด้านการเงิน ความน่าเชื่อถือ และชื่อเสียงขององค์กร ดังนั้นการประเมินความปลอดภัยและการจัดการช่องโหว่ของเว็บแอปพลิเคชันจึงเป็นสิ่งจำเป็นอย่างยิ่ง

ผู้วิจัยเล็งเห็นถึงความสำคัญของข้อมูลผู้ป่วย จึงเป็นที่มาของการศึกษาการทดสอบเจาะระบบและป้องกันการโจมตีเว็บแอปพลิเคชันด้วยคาลีลินุกซ์ (Kali Linux) เพื่อศึกษาวิธีการเจาะระบบรูปแบบต่างๆ ของโรงพยาบาลในประไทย วิธีการตรวจสอบการหาช่องโหว่ การโจมตีทางไซเบอร์โดยใช้ระบบปฏิบัติการคาลีลินุกซ์ และศึกษาแนวทางวิธีป้องกันการถูกโจรกรรมข้อมูลของเว็บแอปพลิเคชัน โดยใช้เครื่องมือ OWASP ZAP และ Nikto ดำเนินการสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาลกลุ่มตัวอย่าง 12 แห่ง เพื่อจำลองและสร้างสภาพแวดล้อมช่องโหว่ให้ใกล้เคียงกับโรงพยาบาลจริง จากนั้นทำการออกแบบเว็บแอปพลิเคชันต้นแบบ สแกนหาช่องโหว่และวิเคราะห์ความเสี่ยง ทดสอบโจมตีช่องโหว่และหาแนวทางป้องกัน

จากผลการศึกษาพบว่า การสแกนหาช่องโหว่ของเว็บแอปพลิเคชันโรงพยาบาลจำนวน 12 แห่ง พบว่าโรงพยาบาลของรัฐมีจำนวนช่องโหว่มากที่สุด โดยสามารถพัฒนาเว็บแอปพลิเคชันต้นแบบของโรงพยาบาลให้มีสภาพแวดล้อมช่องโหว่ที่ใกล้เคียงกับระบบจริง พร้อมทั้งดำเนินการสแกนช่องโหว่ การจัดลำดับความเสี่ยง และการแก้ไขปิดช่องโหว่ได้สำเร็จลุล่วง ทั้งนี้ ในอนาคตควรขยายขอบเขตการศึกษาไปยังระบบสารสนเทศประเภทอื่นของโรงพยาบาล เช่น แอปพลิเคชันบนอุปกรณ์เคลื่อนที่ (Mobile Application) เพื่อใช้เป็นข้อมูลประกอบการตัดสินใจสำหรับหน่วยงานภาครัฐองค์กรด้านการแพทย์ และหน่วยงานที่เกี่ยวข้องต่อไป

บัณฑิตศึกษา ลายมือชื่อนักศึกษา.....

สาขาวิชา เทคโนโลยีสารสนเทศ ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา 2568

PICHA PATRANERAMIT : PENETRATION TESTING AND PROTECTION AGAINST WEB APPLICATION ATTACKS USING KALI LINUX. ADVISOR : DR.NAPAPHAT VICHAIDIS, 59 PP.

Web applications have rapidly expanded and become vital tools for organizational operations, particularly in the healthcare sector where sensitive patient data is highly susceptible to cyberattacks. Since more than 70% of web applications utilize SQL databases, large volumes of patient records are at risk of theft, disclosure, ransom, or misuse, potentially resulting in financial losses and reputational damage if vulnerabilities remain unaddressed.

This research focuses on penetration testing and prevention of cyberattacks on hospital web applications using Kali Linux. The study investigates different penetration testing methods applicable to hospitals in Thailand, techniques for vulnerability detection, penetration testing, and prevention strategies. Vulverability scanning tools such as OWASP ZAP and Nikto were employed to scan for vulnerabilities across web applications from 12 sample hospitals. A prototype hospital web application was then developed to replicate real-world vulnerability environments. The prototype underwent vulnerability scanning, risk assessment, penetration testing, and the implementation of mitigation measures.

The results revealed that state hospitals exhibited the highest number of vulnerabilities compared to other hospital types. Moreover, the prototype web application was successfully enhanced to closely reflect actual hospital environments, while vulnerability scanning, risk assessment, and remediation were effectively completed. Future research should extend to other hospital information systems, such as mobile applications, to provide valuable insights for government agencies, medical organizations, and related stakeholders in making informed cyber security decisions.

Graduate Studies Student's Signature.....

Field of Study Information Technology Advisor's Signature.....

Academic Year 2025

กิตติกรรมประกาศ

ในงานวิจัยฉบับนี้ สำเร็จลุล่วงได้อย่างสมบูรณ์ด้วยความกรุณาอย่างยิ่งจาก ดร.ณปภัช วิชัยดิษฐ์ ที่ได้กรุณาสละเวลาอันมีค่าในการเป็นอาจารย์ที่ปรึกษาเพื่อให้งานวิจัยฉบับนี้ดำเนินการสำเร็จลุล่วงจนสมบูรณ์

ข้าพเจ้าขอกราบขอบพระคุณ รองศาสตราจารย์.ดร.อรรณพ หมั่นสกุล ที่ได้กรุณาให้คำปรึกษาและข้อเสนอแนะที่เป็นประโยชน์ต่อการดำเนินงานวิจัย จนทำให้งานวิจัยฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

นอกจากนี้ ข้าพเจ้าขอขอบคุณ ครอบครัวและบุคคลใกล้ชิด ที่เป็นกำลังใจและกำลังสนับสนุนที่สำคัญทำให้ข้าพเจ้ามีกำลังใจและสามารถดำเนินการวิจัยจนเสร็จสมบูรณ์

ข้าพเจ้าขอขอบคุณ สถาบันเทคโนโลยีไทย-ญี่ปุ่น ที่ได้มอบโอกาสในการศึกษา ถ่ายทอดองค์ความรู้ และให้คำแนะนำอันทรงคุณค่า ตลอดระยะเวลาการศึกษาในครั้งนี้

ท้ายที่สุด ข้าพเจ้ามีความหวังเป็นอย่างยิ่งว่า งานวิจัยฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจศึกษาในด้าน การรักษาความมั่นคงปลอดภัยไซเบอร์ และสามารถนำผลงานวิจัยไปประยุกต์ ต่อยอดหรือใช้ในการดำเนินงานที่เกี่ยวข้องได้อย่างเหมาะสมต่อไปในอนาคต

พิชชา ภัทรเนรมิต

สารบัญ

หน้า

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญรูป

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันมีการใช้เว็บแอปพลิเคชัน (Web Application) มีการขยายตัวอย่างรวดเร็ว โดยเว็บแอปพลิเคชัน มีหลายประเภท เช่น เว็บแอปพลิเคชันสำหรับสำรองข้อมูล, เว็บแอปพลิเคชันสำหรับสร้างเอกสารออนไลน์, เว็บแอปพลิเคชันสำหรับปรับแต่งรูปภาพ, เว็บแอปพลิเคชันสำหรับแปลภาษา และเว็บแอปพลิเคชันสำหรับเล่นเกม เป็นต้น และได้กลายเป็นเครื่องมือสำคัญในการสนับสนุนกระบวนการทำงานในองค์กรทุกประเภท ทั้งในภาคธุรกิจ การแพทย์ การศึกษา และการบริการทั่วไป ด้วยความสะดวกสบายในการเข้าถึงผ่านอินเทอร์เน็ต เว็บแอปพลิเคชันช่วยเพิ่มประสิทธิภาพในการทำงาน เช่น การจัดการข้อมูล การประมวลผลแบบเรียลไทม์ และการเข้าถึงข้อมูลจากทุกที่ทุกเวลา อย่างไรก็ตาม การเพิ่มขึ้นของการใช้เว็บแอปพลิเคชันยังนำไปสู่ความเสี่ยงทางด้านความปลอดภัยทางไซเบอร์ที่สูงขึ้นตามมา

องค์กรต่าง ๆ โดยเฉพาะในภาคการแพทย์ มักจะใช้เว็บแอปพลิเคชันในการบริหารจัดการข้อมูลที่มีความละเอียดอ่อน เช่น ข้อมูลผู้ป่วย ซึ่งมีความเสี่ยงต่อการถูกโจมตีทางไซเบอร์ (Cyber Attack) จากการศึกษาพบว่าเว็บแอปพลิเคชันจะประกอบด้วยฐานข้อมูล SQL สำหรับจัดเก็บข้อมูลมากถึง 70% [1] ซึ่งส่งผลให้ฐานข้อมูลผู้ป่วยจำนวนมากเสี่ยงถูกโจรกรรม จำหน่าย เผยแพร่ รวมถึงการเรียกร้องค่าไถ่ องค์กรที่ไม่สามารถป้องกันช่องโหว่ในเว็บแอปพลิเคชันของตน อาจส่งผลให้เกิดการสูญเสียทางการเงิน ชื่อเสียง และความน่าเชื่อถือ การโจมตีลักษณะนี้สามารถทำให้ข้อมูลของผู้ใช้หลุดออกสู่สาธารณะหรือถูกนำไปใช้ในทางที่ผิด ตัวอย่างสำคัญในประเทศไทย จากข้อมูลของสำนักงานไซเบอร์ กระทรวงดีอีเอส ปี 2564 พบว่ามีข้อมูลบัญชีรายชื่อผู้ป่วยกว่า 16 ล้าน Records ของกระทรวงสาธารณสุขถูกโจรกรรมนำไปวางขายบนเว็บ Raidforums เหตุการณ์นี้ไม่เพียงแต่แสดงถึงช่องโหว่ที่มีอยู่ในระบบของหน่วยงานรัฐ แต่ยังเป็นการตอกย้ำถึงความจำเป็นในการปกป้องข้อมูลในเว็บแอปพลิเคชันอย่างมีประสิทธิภาพ

จากความเป็นมาและปัญหาดังที่กล่าวข้างต้น ผู้วิจัยเห็นถึงความสำคัญของข้อมูลผู้ป่วย อีกทั้งการทดสอบเจาะระบบ (Penetration Testing) นับเป็นขั้นตอนสำคัญในการป้องกันปัญหานี้ ที่ทุกเว็บไซต์หรือทุกแอปพลิเคชันต้องดำเนินการเพื่อมั่นใจได้ว่าจะสามารถปกป้องข้อมูลและความเป็นส่วนตัวของผู้ใช้งานได้ [2] โดยเฉพาะการใช้เครื่องมือที่ทันสมัยและมีความสามารถสูงอย่าง Kali Linux ซึ่งเป็นระบบปฏิบัติการที่ถูกออกแบบมาโดยเฉพาะสำหรับการทดสอบเจาะระบบขั้นสูง ด้วยเครื่องมือที่หลากหลายสำหรับการตรวจสอบช่องโหว่ การทดสอบดังกล่าวสามารถช่วยให้องค์กรมี

ความมั่นใจว่าระบบของตนจะสามารถปกป้องข้อมูลสำคัญและความเป็นส่วนตัวของผู้ใช้งานได้อย่างมีประสิทธิภาพ ทางผู้วิจัยจึงมีวัตถุประสงค์เพื่อศึกษาและทำความเข้าใจวิธีการเจาะระบบรูปแบบต่างๆ ของโรงพยาบาลในประเทศไทย ศึกษาแนวทางและวิธีการตรวจสอบการหาช่องโหว่ (Vulnerabilities) การโจมตีทางไซเบอร์ โดยใช้ระบบปฏิบัติการคาลีลินุกซ์ (Kali Linux) และศึกษาแนวทางวิธีป้องกันการถูกโจรกรรมข้อมูลของเว็บแอปพลิเคชัน

อย่างไรก็ตาม งานวิจัยนี้ผู้วิจัยได้กำหนดขอบเขตในการศึกษาและวิจัย โดยเลือกความเสี่ยงในด้านความปลอดภัยทางไซเบอร์ที่สำคัญที่สุดสำหรับเว็บแอปพลิเคชันจาก OWASP TOP 10 ปี 2021 เท่านั้น

1.2 วัตถุประสงค์ของการศึกษา

1.2.1 ศึกษาและทำความเข้าใจวิธีการเจาะระบบรูปแบบต่างๆของโรงพยาบาลในประไทย

1.2.2 ศึกษาแนวทางและวิธีในการป้องกันการถูกโจรกรรมข้อมูลของเว็บแอปพลิชั่น

1.2.3 ศึกษาแนวทางและวิเคราะห์วิธีการตรวจสอบช่องโหว่การโจมตีทางไซเบอร์โดยใช้ระบบปฏิบัติการคาลีลินุกซ์ (Kali Linux)

1.2.4 ศึกษาความเสี่ยงและวิธีการป้องกันจากช่องโหว่ทางไซเบอร์ที่สำคัญที่สุด ตาม OWASP TOP 10 ปี 2021

1.2.5 เสริมสร้างความรู้ความเข้าใจในด้านการป้องกันและรับมือกับภัยคุกคามทางไซเบอร์ในระดับองค์กร

1.3 ขอบเขตในการศึกษา

1.3.1 การเลือกวิเคราะห์เฉพาะช่องโหว่จาก OWASP TOP 10 ปี 2021

การศึกษาครั้งนี้มุ่งเน้นการวิเคราะห์และทดสอบช่องโหว่ทางด้านความปลอดภัยของเว็บ แอปพลิเคชัน โดยวิเคราะห์และทดสอบช่องโหว่จาก OWASP TOP 10 ปี 2021 ซึ่งเป็นมาตรฐานสากลที่ได้รับการยอมรับ

1.3.2 ประเภทของเว็บแอปพลิเคชันที่ศึกษา

การศึกษานี้จำกัดขอบเขตเฉพาะการทดสอบและวิเคราะห์เว็บแอปพลิเคชัน (Web Application) ที่ใช้งานในโรงพยาบาลของประเทศไทย โดยไม่รวมถึงการทดสอบแอปพลิเคชันประเภทอื่น ๆ เช่น แอปพลิเคชันมือถือ (Mobile Application) หรือระบบที่ไม่เกี่ยวข้องกับเว็บ การศึกษานี้จะมุ่งเน้นไปที่ระบบเว็บแอปพลิเคชันที่จัดการข้อมูลที่มีความอ่อนไหว เช่น ข้อมูลผู้ป่วย ข้อมูลการนัดหมาย และข้อมูลการจัดการทางการแพทย์

1.3.3 การใช้เครื่องมือและระบบปฏิบัติการสำหรับการทดสอบเจาะระบบ

ในการทดสอบครั้งนี้จะใช้ระบบปฏิบัติการ Kali Linux ซึ่งเป็นระบบปฏิบัติการที่มีเครื่องมือการทดสอบความปลอดภัยทางไซเบอร์และการหาช่องโหว่อย่างครบวงจร โดยในการทดสอบจะใช้เครื่องมือที่มาพร้อมกับ Kali Linux เช่น Nmap, Metasploit, และ SQLmap ซึ่งเป็นเครื่องมือที่มีประสิทธิภาพในการตรวจจับช่องโหว่และทดสอบความปลอดภัยของระบบ

1.3.4 การศึกษาเฉพาะโรงพยาบาลในประเทศไทย

ขอบเขตของการศึกษาครั้งนี้มุ่งเน้นการวิเคราะห์และทดสอบเว็บแอปพลิเคชันที่ถูกใช้งานในโรงพยาบาลในประเทศไทย โดยเลือกโรงพยาบาลที่มีการใช้ระบบเว็บแอปพลิเคชันในการจัดการข้อมูลและบริการทางการแพทย์เป็นหลัก ขอบเขตนี้ไม่รวมถึงโรงพยาบาลในประเทศอื่นหรือโรงพยาบาลที่ใช้ระบบอื่นที่ไม่ใช่เว็บแอปพลิเคชัน เนื่องจากข้อจำกัดในด้านข้อมูลและความแตกต่างของกฎหมายความปลอดภัยทางไซเบอร์ในแต่ละประเทศ

1.3.5 การทดสอบในสภาวะแวดล้อมที่ควบคุม

การทดสอบเจาะระบบในครั้งนี้จะทำภายใต้สภาวะแวดล้อมที่ถูกควบคุมและจำลองขึ้น (Controlled Environment) เพื่อไม่ให้เกิดผลกระทบต่อข้อมูลหรือระบบที่ใช้งานจริง การทดสอบจะจำกัดเพียงการวิเคราะห์ช่องโหว่เพื่อการศึกษาเท่านั้น โดยจะไม่มีการแก้ไขหรือปรับปรุงระบบที่ใช้งานจริง และจะไม่ส่งผลกระทบต่อการทำงานของระบบในปัจจุบัน หรือทำให้เกิดความเสี่ยงใด ๆ ต่อข้อมูลของผู้ใช้งานจริง

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 เพิ่มความรู้และความเข้าใจเกี่ยวกับการโจมตีทางไซเบอร์ที่เกี่ยวข้องกับเว็บแอปพลิเคชัน

1.4.2 แนวทางป้องกันการโจมตีทางไซเบอร์ที่สามารถนำไปใช้ได้ในทางปฏิบัติ

1.4.3 การพัฒนาขีดความสามารถในการตรวจสอบและจัดการช่องโหว่ในเว็บแอปพลิเคชัน

1.4.4 การป้องกันข้อมูลสำคัญและเพิ่มความมั่นใจในความปลอดภัยของระบบดิจิทัลในโรงพยาบาล

1.4.5 การนำความรู้ไปต่อยอดในการพัฒนาเครื่องมือและเทคโนโลยีด้านความปลอดภัยทางไซเบอร์

1.5 นิยามศัพท์เฉพาะ

1.5.1 เว็บแอปพลิเคชัน (Web Application) คือ ซอฟต์แวร์โปรแกรมที่ทำงานบนเว็บเซิร์ฟเวอร์และสามารถเข้าถึงได้ทางอินเทอร์เน็ตไคลเอนต์ต่างๆ เช่น เบราว์เซอร์ หรือ แอปพลิเคชันมือถือ [3]

1.5.2 OWASP TOP 10 คือ Open Web Application Security Project (OWASP) เป็นองค์กรที่ไม่แสวงหากำไรก่อตั้งขึ้นในปี 2001 เพื่อช่วยเหลือให้ความรู้เจ้าของเว็บไซต์และผู้เชี่ยวชาญทาง ด้านความปลอดภัยจากการโจมตีทางไซเบอร์ และมีการจัด10อันดับช่องโหว่ที่เป็นปัญหาด้านความปลอดภัยของเว็บแอปพลิเคชัน [2] [4]

1.5.3 Kali Linux คือ เป็นระบบปฏิบัติการ Linux ที่ใช้ Debian เป็นหลัก โดยเน้นที่การทดสอบเจาะระบบขั้นสูงและการแฮกอย่างมีจริยธรรม โดยมีเครื่องมือหลายร้อยรายการที่มุ่งเป้าไปที่งานด้านความปลอดภัยของข้อมูล [5]

บทที่ 2
แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

ในการวิจัยครั้งนี้ ผู้วิจัยได้ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้อง และได้นำเสนอตามหัวข้อต่อไปนี้

2.1 การโจมตีทางไซเบอร์ (Cyber Attack)

2.2 OWASP (Open Worldwide Application Security Project)

2.3 การทดสอบเจาะระบบ (Penetration Testing)

2.1 การโจมตีทางไซเบอร์ (Cyber Attack)

2.1.1 การโจมตีทางไซเบอร์ (Cyber Attack) คืออะไร

การโจมตีทางไซเบอร์ (Cyber Attack) คือการกระทำที่มุ่งเน้นโจมตีระบบคอมพิวเตอร์หรือเครือข่ายด้วยเจตนาไม่พึงประสงค์โดยผู้โจมตีอาจพยายามเข้าถึงข้อมูลสำคัญโดยไม่ได้รับอนุญาต ทั้งนี้เพื่อขโมย แก้ไข หรือทำลายข้อมูลที่เก็บรักษาอยู่ในระบบ เป้าหมายของการโจมตีทางไซเบอร์อาจเป็นไปได้ทั้งหน่วยงานของรัฐ องค์กรธุรกิจ หรือบุคคลทั่วไป โดยกระบวนการโจมตีนี้สามารถดำเนินการได้โดยบุคคลหรือกลุ่มบุคคลที่มีความรู้ความสามารถทางเทคโนโลยี

รูปแบบการโจมตีทางไซเบอร์มักจะใช้ซอฟต์แวร์ที่มีเจตนาร้าย (Mallicious Code) เช่น ไวรัส, มัลแวร์ หรือ แรนซัมแวร์ เพื่อแทรกซึมเข้าไปในระบบเป้าหมาย และจากนั้นเปลี่ยนแปลงหรือลบข้อมูลรวมถึงการขัดขวางการทำงานปกติของระบบ ทั้งนี้การโจมตีบางประเภทอาจสร้างผลกระทบอย่างรุนแรงที่ไม่เพียงแต่จะทำให้ระบบเสียหายหรือข้อมูลสูญหาย แต่อาจนำไปสู่การกระทำผิดทางอาชญากรรม เช่น การขโมยข้อมูลส่วนตัว การขโมยข้อมูลทางการเงิน หรือการใช้ข้อมูลเหล่านั้นเพื่อการขู่กรรโชก

การโจมตีทางไซเบอร์ยังมีการพัฒนาไปอย่างรวดเร็วทั้งในด้านเทคนิคและวิธีการโจมตี ผู้โจมตีอาจใช้วิธีการทางสังคมวิศวกรรม (Social Engineering) เพื่อหลอกลวงเหยื่อให้เปิดเผยข้อมูลส่วนตัว หรือใช้วิธีการโจมตีทางเทคนิค เช่น Distributed Denial of Service (DDoS) ที่ทำให้บริการของระบบหยุดทำงาน ทั้งนี้ ความเสี่ยงจากการโจมตีทางไซเบอร์ทำให้หน่วยงานต่าง ๆ จำเป็นต้องมีการป้องกันระบบที่เข้มงวดขึ้น และใช้มาตการความปลอดภัยทางไซเบอร์ที่ทันสมัย

2.1.2 ประเภทของการโจมตีทางไซเบอร์

การโจมตีทางไซเบอร์มีหลายประเภทที่อาจมุ่งเป้าไปที่ระบบคอมพิวเตอร์ เครือข่าย และข้อมูลสำคัญ โดยประเภทหลัก ๆ ของการโจมตีทางไซเบอร์มีดังนี้ [6]

1) มัลแวร์ (Malware) มัลแวร์คือซอฟต์แวร์ที่มีอันตรายซึ่งถูกออกแบบมาโดยมีวัตถุประสงค์เพื่อการโจรกรรมข้อมูลหรือสร้างความเสียหายให้กับระบบคอมพิวเตอร์โดยจะถูกติดตั้งลงในคอมพิวเตอร์ของผู้ใช้โดยไม่ได้รับความยินยอม มัลแวร์ส่วนมากจะแพร่กระจายผ่านทางอีเมลหรือการใช้สื่อบันทึกข้อมูลร่วมกัน เช่น แฟรชไดรฟ์, การดาวน์โหลดไฟล์, การติดตั้งหรือการดาวน์โหลดโปรแกรมต่าง ๆ จากอินเทอร์เน็ต และการคลิกลิงค์ที่ไม่รู้จักแหล่งที่มา โดยในปัจจุบันการโจมตีด้วยมัลแวร์มักจะมุ่งเป้าไปที่ข้อมูลทางธุรกิจหรือการเงินมากกว่าข้อมูลส่วนบุคคลของผู้ใช้งาน

2) วิศวกรรมทางสังคม (Social Engineering) การโจมตีที่ใช้วิธีการหลอกลวงทางจิตวิทยาเพื่อชักจูงหรือหลอกให้ผู้ใช้เปิดเผยข้อมูลส่วนตัวหรือข้อมูลที่มีความสำคัญ เช่น ชื่อบัญชีผู้ใช้ (Username), รหัสผ่าน, เลขบัตรเครดิต เป็นต้น โดยอาศัยทักษะการสื่อสารของผู้โจมตี ซึ่งสามารถดำเนินการผ่านหลายช่องทาง ไม่ว่าจะเป็นทางโทรศัพท์หรือทางอีเมล หรือแม้แต่การสนทนาผ่านข้อความ เช่น การสร้างหรือดัดแปลง URL เว็บไซต์ให้เหมือนกับของจริงเพื่อหลอกให้ผู้ใช้เปิดเผยข้อมูลส่วนตัว หรือ สร้างสถานการณ์ในลักษณะที่เร่งด่วน เพื่อให้ผู้ใช้รีบตัดสินใจในทันที เป็นต้น

3) Denial-of-Service (DoS) เป็นการโจมตีทางไซเบอร์ที่มุ่งเน้นให้ระบบไม่สามารถให้บริการตามปกติได้ โดยผู้โจมตีจะทำการส่งคำขอหรือปริมาณข้อมูลจำนวนมากไปยังเป้าหมาย เช่น เซิร์ฟเวอร์, เว็บไซต์ หรือ แอปพลิเคชัน จนทำให้ทรัพยากรของระบบ เช่น CPU, หน่วยความจำ หรือแบนด์วิดท์ ถูกใช้งานเกินกว่าความสามารถที่ระบบจะรองรับได้ ส่งผลให้ผู้ใช้งานไม่สามารถเข้าถึงบริการได้ตามปกติ

4) การหลอกลวง (Phising) คือ การโจมตีทางไซเบอร์โดยอาศัยการส่งอีเมลหรือข้อความหลอกลวงซึ่งถูกสร้างขึ้นให้ดูเหมือนมาจากแหล่งที่เชื่อถือได้โดยมีวัตถุประสงค์หลักเพื่อหลอกลวงให้ผู้รับเปิดเผยข้อมูลส่วนบุคคลหรือข้อมูลประจำตัวเช่นชื่อผู้ใช้งาน (Username), รหัสผ่าน, ข้อมูลบัตรเครดิต หรือข้อมูลทางการเงินอื่น ๆ การโจมตีรูปแบบ Phishing เป็นรูปแบบหนึ่งของ Social Engineering ซึ่งผสมผสานเทคนิคทางจิตวิทยากับการหลอกลวงทางเทคนิค โดยมักอยู่ในรูปแบบของอีเมลที่ประกอบด้วยลิงค์หรือไฟล์แนบที่เป็นอันตรายลิงค์เหล่านี้อาจนำผู้ใช้ไปยังเว็บไซต์ปลอมที่ออกแบบมาให้คล้ายคลึงกับเว็บไซต์จริงเพื่อหลอกให้ผู้ใช้ป้อนข้อมูลสำคัญของตน หรือในบางกรณี ลิงค์อาจนำไปสู่การดาวน์โหลดซอฟต์แวร์ที่เป็นมัลแวร์เข้าสู่ระบบของผู้ใช้โดยไม่รู้ตัว

5) การโจมตีแบบสปูฟฟิง (Spoofing Attack) คือ เทคนิคการโจมตีทางไซเบอร์ที่ผู้โจมตีปลอมแปลงตนเองให้ดูเหมือนเป็นแหล่งข้อมูลที่เชื่อถือได้ โดยมีจุดประสงค์เพื่อหลอกลวงเหยื่อให้เชื่อถือและเปิดช่องทางให้ผู้โจมตีสามารถเข้าถึงระบบหรืออุปกรณ์ของเหยื่อได้ การปลอมแปลงใน

ลักษณะนี้ช่วยให้ผู้โจมตีสามารถทำกิจกรรมที่เป็นอันตราย เช่น ขโมยข้อมูลที่สำคัญ, ขู่กรรโชกเงิน หรือทำการติดตั้งซอฟต์แวร์ที่เป็นอันตรายหรือมัลแวร์ลงในอุปกรณ์เป้าหมายได้

6) การโจมตีแบบอิงตามตัวตน (Identity-Based Attacks) คือการโจมตีทางไซเบอร์ที่มีความซับซ้อนและตรวจจับได้ยากอย่างยิ่ง เนื่องจากผู้โจมตีใช้ข้อมูลประจำตัวของผู้ใช้งานที่ถูกต้องที่ได้มาจากการแฮกหรือการหลอกลวง เมื่อข้อมูลประจำตัวที่แท้จริงของผู้ใช้งานถูกขโมย ผู้โจมตีสามารถปลอมแปลงเป็นผู้ใช้นั้นและเข้าถึงระบบหรือข้อมูลสำคัญได้โดยไม่ถูกตรวจพบ ความยากในการตรวจสอบพฤติกรรมของผู้ใช้งานที่ถูกปลอมแปลงนี้เกิดจากความคล้ายคลึงระหว่างพฤติกรรมการใช้งานที่ปกติของผู้ใช้จริงและการกระทำของผู้โจมตี ทำให้ระบบรักษาความปลอดภัยทั่วไปที่อาศัยการตรวจจับพฤติกรรมผิดปกติไม่สามารถแยกแยะได้อย่างแม่นยำ

7) การโจมตีด้วยการแทรกโค้ด (Code Injection Attacks) คือการโจมตีทางไซเบอร์ที่ผู้โจมตีแทรกโค้ดที่เป็นอันตรายเข้าไปในระบบหรือเครือข่ายที่มีช่องโหว่โดยมีจุดประสงค์เพื่อเปลี่ยนแปลงการทำงานของระบบหรือเข้าถึงข้อมูลที่เป็นความลับ การโจมตีประเภทนี้มีหลายรูปแบบโดยตัวอย่างที่พบได้บ่อย เช่น

- การโจมตีด้วยการแทรกโค้ด SQL (Code Injection Attacks) คือการโจมตีที่ใช้ช่องโหว่ของระบบแอปพลิเคชันที่มีการเชื่อมต่อกับฐานข้อมูลเพื่อแทรกคำสั่ง SQL ที่เป็นอันตรายเข้าไปในระบบ โดยคำสั่ง SQL เหล่านี้สามารถใช้ในการดึงข้อมูลที่สำคัญจากฐานข้อมูล แก้ไขข้อมูล หรือลบข้อมูลออกจากระบบ ฐานข้อมูลที่ไม่ได้รับการป้องการหรือมีช่องโหว่สามารถตกเป็นเป้าหมายหลักของการโจมตีลักษณะนี้
- การโจมตีด้วยการแทรกโค้ด Cross-Site Scripting (XSS) เป็นการโจมตีที่ผู้โจมตีแทรกโค้ดที่เป็นอันตรายเข้าไปในเว็บไซต์ โดยโค้ดเหล่านี้จะถูกเรียกใช้เป็นสคริปต์ที่เป็นอันตรายในเว็บเบราว์เซอร์ของผู้ใช้ซึ่งทำให้ผู้โจมตีสามารถขโมยข้อมูลสำคัญหรือสวมรอยเป็นผู้ใช้นั้นๆ

8) การโจมตีในห่วงโซ่อุปทาน (Supply Chain Attacks) คือการโจมตีทางไซเบอร์รูปแบบหนึ่งที่มุ่งเป้าไปยังผู้ให้บริการหรือผู้จัดหาซอฟต์แวร์ที่มีบทบาทสำคัญในกระบวนการทำงานขององค์กร โดยผู้โจมตีจะอาศัยช่องโหว่หรือการแทรกซึมเข้าไปในซอฟต์แวร์ของผู้จัดหา เพื่อกระจายโค้ดที่เป็นอันตรายเข้าสู่ระบบของผู้ใช้ที่เกี่ยวข้อง โดย Supply Chain Attacks จะเกิดขึ้นเมื่อผู้โจมตีแทรกโค้ดที่เป็นอันตรายลงในซอฟต์แวร์ที่ได้รับการจัดหาจากภายนอก ซึ่งเมื่อซอฟต์แวร์ที่ได้รับผลกระทบถูกนำไปใช้งานในระบบ โค้ดที่เป็นอันตรายจะแพร่กระจายไปยังผู้ใช้ทั้งหมดที่พึ่งพาซอฟต์แวร์นี้ ส่งผลให้เกิดความเสี่ยงต่อข้อมูลและการปฏิบัติงานขององค์กรในวงกว้าง

9) ภัยคุกคามจากบุคคลภายใน (Insider Threats) ภัยคุกคามจากบุคคลภายในองค์กรเป็นความเสี่ยงที่มักถูกมองข้าม หากการจัดการด้านความปลอดภัยของข้อมูลมุ่งเน้นเพียงการตรวจสอบและป้องกันภัยคุกคามจากบุคคลภายนอก องค์กรอาจยังคงเปิดโอกาสให้เกิดการโจมตีได้ บุคคลภายในที่เป็นภัยคุกคามสามารถเป็นได้ทั้งพนักงานปัจจุบันหรืออดีตพนักงาน ซึ่งมีสิทธิ์เข้าถึงเครือข่ายขององค์กร ข้อมูลที่สำคัญ และทรัพย์สินทางปัญญา อีกทั้งยังมีความรู้เกี่ยวกับกระบวนการทางธุรกิจ นโยบายของบริษัท หรือข้อมูลอื่น ๆ ที่สามารถช่วยให้การโจมตีเกิดขึ้นได้

10) การโจมตีแบบ DNS Tunneling เป็นการโจมตีทางไซเบอร์ที่ใช้ช่องทางการสื่อสารผ่านโปรโตคอล DNS (Domain Name System) ในการส่งข้อมูลหรือโค้ดที่เป็นอันตรายผ่านเครือข่าย ซึ่งเป็นวิธีการที่ผู้โจมตีใช้ในการหลบเลี่ยงมาตรการความปลอดภัยแบบดั้งเดิมของระบบเครือข่ายเมื่อระบบถูกติดมัลแวร์ ผู้โจมตีสามารถใช้ช่องทาง DNS ในการดำเนินการควบคุมและสั่งการจากระยะไกล (Command-and-Control) เพื่อแพร่กระจายมัลแวร์หรือทำการดึงข้อมูลสำคัญออกจากระบบ เช่น ข้อมูลทรัพย์สินทางปัญญา หรือข้อมูลส่วนบุคคล โดยผู้โจมตีจะเข้ารหัสข้อมูลเหล่านี้เป็นบิตเล็ก ๆ แล้วส่งผ่านคำร้องและการตอบสนอง DNS ทำให้กระบวนการนี้ยากต่อการตรวจจับ

11) การโจมตีในระบบ Internet of Things (IoT) เป็นการโจมตีทางไซเบอร์ที่มุ่งเป้าไปยังอุปกรณ์ IoT หรือเครือข่ายที่เชื่อมต่อกับอุปกรณ์เหล่านี้ เมื่ออุปกรณ์ IoT ถูกโจมตีสำเร็จ ผู้โจมตีสามารถเข้าควบคุมอุปกรณ์ ขโมยข้อมูล หรือทำให้เกิดการรวมกลุ่มของอุปกรณ์ที่ติดเชื้อเพื่อสร้างบอตเน็ต (Botnet) เพื่อใช้ในการโจมตีแบบปฏิเสธการให้บริการ (DoS) หรือการโจมตีแบบปฏิเสธการให้บริการแบบกระจาย (DDoS)

12) การโจมตีที่ขับเคลื่อนด้วยปัญญาประดิษฐ์ (AI-Powered Attacks) คือการโจมตีทางไซเบอร์ที่ใช้เทคโนโลยีปัญญาประดิษฐ์และการเรียนรู้ของเครื่อง (Machine Learning: ML) เพื่อประโยชน์ในการเจาะเข้าถึงระบบเครือข่ายหรือขโมยข้อมูลสำคัญ ผู้โจมตีสามารถใช้ AI ในการตรวจหาช่องโหว่ที่ซับซ้อนหรือไม่สามารถตรวจพบได้โดยง่าย รวมถึงสร้างมัลแวร์ที่ปรับเปลี่ยนตนเองได้เพื่อหลบเลี่ยงการตรวจจับจากระบบความปลอดภัยดั้งเดิม

2.2 OWASP (Open Worldwide Application Security Project)

2.2.1 OWASP คืออะไร

OWASP (Open Worldwide Application Security Project) เป็นมูลนิธิไม่แสวงหาผลกำไรที่มุ่งมั่นพัฒนาความปลอดภัยของซอฟต์แวร์ โดยมีโครงการโอเพนซอร์สที่ขับเคลื่อนโดยชุมชนทั่วโลกมากกว่า 250 ชุมชน และมีสมาชิกหลายหมื่นคน องค์กรนี้ให้เครื่องมือ เอกสาร และมาตรฐานฟรี

แก่ทุกคนเพื่อพัฒนาความปลอดภัยในการพัฒนาแอปพลิเคชัน โดย OWASP เริ่มในวันที่ 1 ธันวาคม 2001 และได้จดทะเบียนเป็นองค์กรไม่แสวงหากำไรในสหรัฐฯ ในวันที่ 21 เมษายน 2004

2.2.2 OWASP Top 10 คืออะไร

OWASP Top 10 เป็นเอกสารที่สร้างความตระหนักให้กับนักพัฒนาและการรักษาความปลอดภัยเว็บแอปพลิเคชัน ซึ่งแสดงถึงความเห็นพ้องจากผู้เชี่ยวชาญทั่วโลกเกี่ยวกับความเสี่ยงด้านความปลอดภัยที่สำคัญที่สุดของเว็บแอปพลิเคชัน เป็นที่ยอมรับในวงกว้างว่าเป็นขั้นตอนแรกสำหรับการเขียนโค้ดที่ปลอดภัยยิ่งขึ้น

ในปัจจุบัน OWASP Top 10 ประจำปี 2021 ได้ระบุหมวดหมู่ความเสี่ยงที่สำคัญต่อความปลอดภัยของเว็บแอปพลิเคชัน โดยมีการจัดลำดับความเสี่ยงที่พบได้บ่อยและมีผลกระทบสูงดังนี้

1) A01:2021 - Broken Access Control คือความเสี่ยงที่เกิดจากการควบคุมสิทธิ์ในการเข้าถึงที่ไม่สมบูรณ์ ทำให้ผู้โจมตีสามารถเข้าถึงฟังก์ชันและข้อมูลที่ควรถูกจำกัดได้ เป็นหนึ่งในความเสี่ยงที่พบบ่อยที่สุด

2) A02:2021 - Cryptographic Failures เป็นความล้มเหลวในการใช้วิธีการเข้ารหัสที่เหมาะสม อาจส่งผลให้ข้อมูลที่อ่อนไหวถูกเปิดเผยหรือระบบถูกโจมตี ความเสี่ยงนี้เคยถูกจัดอยู่ในหมวด "Sensitive Data Exposure" แต่ในปี 2021 มีการเน้นที่สาเหตุของปัญหาจากการเข้ารหัสเป็นหลัก

3) A03:2021 - Injection การโจมตีที่เกิดจากการฝังโค้ดที่ไม่พึงประสงค์ในแอปพลิเคชันเช่น SQL หรือ Cross-Site Scripting (XSS) โดยการโจมตีลักษณะนี้ยังคงมีอยู่มาก

4) A04:2021 - Insecure Design คือความเสี่ยงจากข้อบกพร่องในการออกแบบที่ไม่ปลอดภัย เช่น การขาดการใช้ Threat Modeling หรือรูปแบบการออกแบบที่ปลอดภัย ส่งผลให้แอปพลิเคชันอาจเสี่ยงต่อการถูกโจมตีตั้งแต่ขั้นตอนการพัฒนา

5) A05:2021 - Security Misconfiguration เกิดจากการตั้งค่าความปลอดภัยที่ไม่ถูกต้องหรือขาดการควบคุมความปลอดภัยที่เพียงพอ

6) A06:2021 - Vulnerable and Outdated Components เป็นความเสี่ยงที่เกิดจากการใช้ส่วนประกอบหรือไลบรารีที่ล้าสมัย หรือมีช่องโหว่ที่รู้จักกันอยู่แล้ว ส่งผลให้แอปพลิเคชันสามารถถูกโจมตีได้จากช่องโหว่เหล่านั้น

7) A07:2021 - Identification and Authentication Failures คือปัญหาการระบุตัวตนและการตรวจสอบสิทธิ์ที่ไม่สมบูรณ์ ซึ่งแม้จะยังคงเป็นปัญหาหลัก แต่มีการลดลงจากการใช้เฟรมเวิร์กมาตรฐานที่เพิ่มมากขึ้น

8) A08:2021 - Software and Data Integrity Failures ความเสี่ยงที่เกิดจากการอัพเดตซอฟต์แวร์ การจัดการข้อมูลสำคัญ และ CI/CD Pipelines โดยไม่มีการตรวจสอบความสมบูรณ์ ทำให้มีช่องทางในการโจมตีได้

9) A09:2021 - Security Logging and Monitoring Failures ความเสี่ยงจากการขาดการบันทึกและติดตามเหตุการณ์ความปลอดภัยที่เพียงพอ ส่งผลให้การแจ้งเตือนหรือการตรวจสอบเหตุการณ์ในภายหลังทำได้ยาก

10) A10:2021 - Server-Side Request Forgery (SSRF) เกิดจากการส่งคำร้องขอจากเซิร์ฟเวอร์ไปยังแหล่งภายนอกโดยไม่ได้ตั้งใจ เมื่อเว็บแอปพลิเคชันดึงข้อมูลจากแหล่งข้อมูลระยะไกลโดยไม่ทำการตรวจสอบ URL ที่ผู้ใช้ส่งเข้ามา ส่งผลให้ผู้โจมตีสามารถใช้ประโยชน์จากช่องโหว่ดังกล่าวเพื่อบังคับแอปพลิเคชันส่งคำร้องขอไปยังปลายทางที่ไม่คาดคิด แม้จะมีการป้องกันด้วยไฟร์วอลล์, VPN, หรือ Access Control List (ACL) ก็ตาม

2.3 การทดสอบเจาะระบบ (Penetration Testing)

2.3.1 การทดสอบเจาะระบบ (Penetration Testing) คืออะไร

การทดสอบเจาะระบบ (Penetration Testing) เป็นกระบวนการทดสอบความปลอดภัยของระบบคอมพิวเตอร์โดยการจำลองการโจมตีจากบุคคลภายนอกหรือภายในที่ไม่มีสิทธิ์เข้าถึงข้อมูล วัตถุประสงค์หลักคือเพื่อระบุช่องโหว่ที่อาจถูกผู้ไม่หวังดีใช้ประโยชน์ในการเข้าถึงระบบโดยไม่ได้รับอนุญาต การทดสอบนี้จะช่วยตรวจสอบทั้งด้านฮาร์ดแวร์ ซอฟต์แวร์ และกระบวนการทำงานที่เกี่ยวข้อง เพื่อประเมินประสิทธิภาพของมาตรการป้องกันและกำหนดแนวทางในการปรับปรุงมาตรการรักษาความปลอดภัยให้แข็งแกร่งมากยิ่งขึ้น [7]

กระบวนการทดสอบเจาะระบบครอบคลุมตั้งแต่การรวบรวมข้อมูล การประเมินความเสี่ยง การระบุช่องโหว่ต่างๆ ที่อาจเกิดขึ้น เช่น การกำหนดค่าระบบที่ไม่ถูกต้อง ข้อบกพร่องในซอฟต์แวร์และฮาร์ดแวร์ และการวิเคราะห์แนวทางปฏิบัติด้านความปลอดภัยของบุคลากรที่เกี่ยวข้อง ซึ่งสามารถช่วยให้ทีมรักษาความปลอดภัยรับรู้ถึงช่องโหว่ที่อาจถูกโจมตีได้ และหาแนวทางในการป้องกันล่วงหน้า

ทั้งนี้ การทดสอบเจาะระบบแตกต่างจากการประเมินความปลอดภัยทั่วไป เพราะเน้นการจำลองการโจมตีจริงที่อาจเกิดขึ้นในสถานการณ์จริง นอกจากนี้ยังสามารถดำเนินการได้ทั้งในส่วนของระบบภายใน (เช่น อุปกรณ์ IoT เครือข่าย และแอปพลิเคชัน) และระบบภายนอก (เช่น เว็บแอปพลิเคชัน หรือระบบที่เปิดให้เข้าถึงผ่านเครือข่ายสาธารณะ) โดยใช้เครื่องมืออัตโนมัติหรือการโจมตีด้วยวิธีการแบบแมนนวล เพื่อวัดระดับความปลอดภัยของระบบ

2.4 CWE (Common Weakness Enumeration)

2.4.1 CWE (Common Weakness Enumeration) คืออะไร

CWE (Common Weakness Enumeration) คือรายการมาตรฐานที่ใช้ในการรวบรวมและจัดหมวดหมู่จุดอ่อน (Weaknesses) ของซอฟต์แวร์ เฟิร์มแวร์ ฮาร์ดแวร์ และองค์ประกอบของระบบ ซึ่งภายใต้เงื่อนไขบางประการอาจนำไปสู่การเกิดช่องโหว่ด้านความปลอดภัยได้ รายการนี้ได้รับการพัฒนาและดูแลโดย MITRE Corporation และอัปเดตเป็นประจำหลายครั้งต่อปีเพื่อสะท้อนข้อมูลใหม่ ๆ

CWE นำเสนอทั้งในรูปแบบ “รายการจุดอ่อน” (CWE List) และ “โครงสร้างการจัดหมวดหมู่” (Classification Taxonomy) เพื่อให้สามารถอธิบายและสื่อสารถึงจุดอ่อนต่าง ๆ ได้อย่างเป็นระบบ โดยมีการจัดทำมุมมอง (Views) เพื่อรองรับการใช้งานในหลากหลายโดเมน เช่น การพัฒนาซอฟต์แวร์ การออกแบบฮาร์ดแวร์ และงานวิจัยเชิงความมั่นคงปลอดภัย

นอกจากนี้ CWE ยังถูกใช้งานอย่างแพร่หลายทั่วโลกโดยนักพัฒนา นักวิชาการ ผู้ให้บริการด้านความปลอดภัย ตลอดจนหน่วยงานภาครัฐ (government agencies) เพื่อเป็นมาตรฐานกลางในการอธิบายจุดอ่อนของระบบ โดยเปิดให้ใช้งานได้โดยไม่มีค่าใช้จ่าย เพื่อสนับสนุนการยกระดับความมั่นคงปลอดภัยของซอฟต์แวร์และฮาร์ดแวร์ในระดับสากล

2.5 CVSS (Common Vulnerability Scoring System)

2.5.1 CVSS คืออะไร

CVSS (Common Vulnerability Scoring System) คือระบบมาตรฐานสากลที่ใช้ในการให้คะแนนและวัดระดับความรุนแรงของช่องโหว่ด้านความมั่นคงปลอดภัย โดยมีวัตถุประสงค์เพื่อช่วยให้องค์กรสามารถจัดลำดับความสำคัญในการแก้ไขช่องโหว่ได้อย่างเป็นระบบและสามารถเปรียบเทียบข้ามระบบหรือผลิตภัณฑ์ได้อย่างเท่าเทียม

CVSS ได้รับการดูแลและพัฒนาโดย FIRST (Forum of Incident Response and Security Teams) ซึ่งเป็นองค์กรชั้นนำระดับโลกด้านการตอบสนองต่อเหตุการณ์ความมั่นคงปลอดภัยไซเบอร์ มีสมาชิกมากกว่า 800 ทีมทั่วโลก ครอบคลุมทั้งหน่วยงานรัฐบาล ภาคเอกชน และสถาบันการศึกษา FIRST มีบทบาทสำคัญในการส่งเสริมความร่วมมือ การแลกเปลี่ยนข้อมูล และการประสานงานในการป้องกันและตอบสนองต่อเหตุการณ์ด้านความมั่นคงปลอดภัย ทั้งในเชิงปฏิกิริยา (reactive) และเชิงรุก (proactive)

ดังนั้น CVSS จึงเป็นระบบการให้คะแนนที่ได้รับการยอมรับในระดับสากล ซึ่งมีรากฐานจากองค์กรผู้เชี่ยวชาญด้านการตอบสนองเหตุการณ์ความปลอดภัยไซเบอร์ ทำให้สามารถใช้เป็นมาตรฐานกลางในการประเมินและสื่อสารระดับความรุนแรงของช่องโหว่ได้อย่างกว้างขวางทั่วโลก

2.6 ราชกิจจานุเบกษา (Royal Thai Government Gazette)

2.6.1 ราชกิจจานุเบกษา (Royal Thai Government Gazette) คืออะไร

ราชกิจจานุเบกษา (Royal Thai Government Gazette) คือ หนังสือทางการของรัฐบาลไทย ที่จัดพิมพ์และเผยแพร่โดยสำนักงานราชกิจจานุเบกษา สำนักเลขาธิการคณะรัฐมนตรี ทำหน้าที่เป็นช่องทางในการประกาศกฎหมาย ข้อบังคับ คำสั่ง และประกาศต่าง ๆ ของหน่วยงานรัฐ เพื่อให้มีผลบังคับใช้โดยสมบูรณ์และเป็นที่รับรู้ในทางราชการและสาธารณะ

2.7 งานวิจัยที่เกี่ยวของ

K. Nagendran et al. [1] ได้กล่าวถึงการศึกษาและประเมินกระบวนการทดสอบการเจาะระบบ (Penetration Testing) ที่มุ่งเน้นไปที่เว็บแอปพลิเคชัน โดยเฉพาะการป้องกันและตรวจจับการโจมตีจากช่องโหว่ที่พบบ่อยในแอปพลิเคชันเว็บ ซึ่งมีรายละเอียดขั้นตอนวิธีการวิจัยดังนี้ 1) การวางแผนการทดสอบ (Planning the Test) นักวิจัยเริ่มต้นด้วยการกำหนดขอบเขตของการทดสอบ โดยรวมถึงการกำหนดเป้าหมายที่ชัดเจนและประเภทของการโจมตีที่ต้องการทดสอบ เพื่อเตรียมเครื่องมือและเทคนิคที่เหมาะสมต่อไป 2) การเก็บข้อมูล (Information Gathering) ขั้นตอนนี้เกี่ยวข้องกับการรวบรวมข้อมูลที่จำเป็นเกี่ยวกับเว็บแอปพลิเคชัน เช่น โครงสร้างของระบบ ฟังก์ชันการทำงาน และกลไกการป้องกัน เพื่อให้เข้าใจบริบทของการทดสอบอย่างชัดเจน 3) การวิเคราะห์ช่องโหว่ (Vulnerability Analysis) นักวิจัยทำการวิเคราะห์ช่องโหว่ที่มีอยู่ในเว็บแอปพลิเคชัน โดยใช้เครื่องมือต่าง ๆ เพื่อตรวจสอบจุดอ่อน เช่น SQL Injection, Cross-Site Scripting (XSS) และช่องโหว่ที่เกี่ยวข้องกับการจัดการข้อมูล 4) การโจมตี (Exploitation) เมื่อนักวิจัยพบช่องโหว่ที่เป็นไปได้ ได้มีการดำเนินการโจมตีโดยใช้เทคนิคต่าง ๆ เพื่อแสดงให้เห็นว่าช่องโหว่นั้นสามารถถูกใช้เพื่อเข้าถึงข้อมูลหรือระบบที่มีความสำคัญได้อย่างไร 5) การรายงาน (Reporting) สุดท้าย นักวิจัยจัดทำรายงานเกี่ยวกับผลการทดสอบ รวมถึงรายละเอียดเกี่ยวกับช่องโหว่ที่พบและแนวทางในการแก้ไขหรือป้องกันปัญหาดังกล่าว เพื่อให้ผู้พัฒนาเว็บแอปพลิเคชันสามารถดำเนินการปรับปรุงความปลอดภัยได้อย่างเหมาะสม ผลลัพธ์จากการทดสอบที่ดำเนินการ นักวิจัยพบว่ามีช่องโหว่ในเว็บแอปพลิเคชันจำนวนมาก โดยเฉพาะในด้านการป้องกัน SQL Injection และ XSS ซึ่งบ่งชี้ว่าหลายระบบยังคงมีความเสี่ยงที่จะถูกโจมตีด้วยเทคนิคเหล่านี้ นอกจากนี้ ผลการวิจัยยังชี้ให้เห็นถึงความจำเป็นในการดำเนินการป้องกันที่เข้มงวดมากขึ้น เช่น การใช้มาตรการป้องกันที่เหมาะสม รวมถึงการฝึกอบรมและการตรวจสอบระบบอย่างสม่ำเสมอ งานวิจัยนี้ไม่เพียงแต่เน้นความสำคัญของการทดสอบเจาะระบบในเว็บแอปพลิเคชัน แต่ยังเสนอแนะถึงแนวทางที่ควรนำไปใช้เพื่อเพิ่มความปลอดภัยและความแข็งแกร่งของระบบในอนาคต

S. Yadav and P. Singh. [2] ได้กล่าวการวิจัยเพื่อวิเคราะห์ความเสี่ยงและช่องโหว่ที่มีอยู่ในเว็บแอปพลิเคชัน รวมถึงการเสนอวิธีการป้องกันที่มีประสิทธิภาพ โดยวัตถุประสงค์ของการวิจัยเพื่อชี้ให้เห็นถึงช่องโหว่ในเว็บแอปพลิเคชัน และเสนอแนวทางในการป้องกันเพื่อช่วยให้ผู้อ่านมีความเข้าใจในด้านความปลอดภัยของเว็บแอปพลิเคชัน ผู้วิจัยได้ทำการทบทวนเอกสารที่เกี่ยวข้องเพื่อเข้าใจช่องโหว่ที่เกิดขึ้นในเว็บแอปพลิเคชัน โดยมีการอ้างอิงถึง OWASP Top 10 ซึ่งเป็นแนวทางที่สำคัญในการประเมินความปลอดภัยของเว็บแอปพลิเคชัน โดยช่องโหว่ที่ถูกวิเคราะห์ได้แก่ Injection รวมถึง SQL Injection ที่สามารถเข้าถึงฐานข้อมูลได้, Cross Site Scripting (XSS), การจัดการการเข้าถึงที่ไม่ปลอดภัย (Broken Authentication), และการเปิดเผยข้อมูลที่สำคัญ (Sensitive Data Exposure) โดยวิธีการทดสอบการวิจัยผู้วิจัยได้กำหนดขั้นตอนในการทดสอบ โดยเริ่มจากการวางแผนและการเก็บข้อมูล เพื่อวิเคราะห์ความเสี่ยงที่เกิดขึ้นจากช่องโหว่ที่ค้นพบและ ใช้เครื่องมือในการทดสอบ เช่น SQLMap สำหรับการทดสอบ SQL Injection โดยการส่งคำสั่ง SQL ที่เป็นอันตรายเพื่อค้นหาช่องโหว่ในฐานข้อมูล Burp Suite เพื่อทดสอบช่องโหว่ทั่วไป โดยการดักจับและวิเคราะห์คำขอ HTTP ผลลัพธ์การศึกษานี้ได้รวมการศึกษาร่วมระหว่างการทดสอบการเจาะระบบและการทดสอบแอปพลิเคชัน เพื่อให้สามารถเข้าใจภาพรวมของช่องโหว่ที่มีอยู่ในเว็บแอปพลิเคชันได้อย่างชัดเจน

A. Anas et al. [3] ได้วิเคราะห์และนำเสนอการป้องกันและตรวจจับการโจมตีประเภท "Broken Access Control" (BAC) นักวิจัยได้ใช้วิธีการวิจัยที่ครอบคลุมการวิเคราะห์เชิงลึกของช่องโหว่ที่เกิดจากการควบคุมการเข้าถึงในเว็บแอปพลิเคชัน โดยได้รวบรวมข้อมูลจากหลายแหล่งเพื่อศึกษาเทคนิคที่มีอยู่ในการตรวจจับและป้องกันการโจมตี โดยแบ่งเทคนิคต่างๆ ออกเป็นหมวดหมู่ตามการทำงาน โดยนำเสนอข้อจำกัดของเทคนิคปัจจุบัน เช่น การพึ่งพาการรวบรวมข้อมูลด้วยมนุษย์หรือบอท การพึ่งพาเทคโนโลยีหรือแพลตฟอร์มที่เฉพาะเจาะจง และความต้องการในการเข้าถึงโค้ดต้นทางของแอปพลิเคชัน อีกทั้ง ยังชี้ให้เห็นว่าหลายเทคนิคไม่สามารถตรวจจับการโจมตีแบบ Insecure Direct Object References (IDOR) ได้ ซึ่งเป็นช่องโหว่ที่สำคัญในการควบคุมการเข้าถึงของแอปพลิเคชัน การวิเคราะห์เทคนิคต่าง ๆ โดยอ้างอิงคุณลักษณะ เช่น False Positives (FP), Persistency Aware (PA), Workflows Aware (WA) และ Manual Intervention Required (MIR) แสดงให้เห็นว่า การตรวจจับ BAC ในปัจจุบันยังขาดประสิทธิภาพในหลายด้าน โดยเฉพาะการพึ่งพาภาษาพัฒนาและแพลตฟอร์มที่แตกต่างกัน การเข้าถึงโค้ดต้นทางของแอปพลิเคชัน และการจัดการข้อมูล NoSQL การวิจัยได้สรุปว่า แนวทางที่มีอยู่ในปัจจุบันยังไม่ครอบคลุมและมีประสิทธิภาพเพียงพอในการป้องกันการโจมตีที่เกิดจากการควบคุมการเข้าถึงที่บกพร่อง ซึ่งเป็นปัญหาที่สำคัญมาก

A. Alanda et al. [4] ได้กล่าวถึงการทดสอบการเจาะระบบเว็บแอปพลิเคชันโดยใช้การโจมตีแบบ SQL Injection มุ่งเน้นการตรวจสอบความปลอดภัยของเว็บแอปพลิเคชันผ่านการเจาะช่องโหว่ SQL Injection ซึ่งถือเป็นหนึ่งในความเสี่ยงที่พบได้บ่อยในระบบเว็บแอปพลิเคชัน การวิจัยนี้

ใช้วิธีการทดสอบแบบ Black-box Penetration Testing โดยนักวิจัยได้เลือกเว็บไซต์แบบสุ่มจำนวน 10 แห่ง ได้แก่ เว็บไซต์ภาครัฐ โรงเรียน และเว็บไซต์เชิงพาณิชย์ เพื่อทำการประเมินความสามารถในการป้องกันช่องโหว่ SQL Injection โดยการดำเนินการทดสอบเจาะระบบ (Penetration Testing) ประกอบด้วยขั้นตอนที่ชัดเจน ซึ่งอิงตามมาตรฐานการทดสอบเจาะระบบ (Penetration Testing Execution Standard: PTES) โดยขั้นตอนเหล่านี้รวมถึงการเก็บข้อมูล การวิเคราะห์ช่องโหว่ และการรายงานผล ดังนี้ 1) Pre-engagement Interactions ขั้นตอนแรกที่นักทดสอบเจาะระบบจะหารือกับลูกค้าเกี่ยวกับขอบเขตและวัตถุประสงค์ของการทดสอบ รวมถึงการอธิบายรายละเอียดของกิจกรรมที่ดำเนินการตั้งแต่เริ่มต้นจนถึงการจัดทำรายงาน 2) Intelligence Gathering นักทดสอบจะเก็บข้อมูลที่สำคัญจากเป้าหมาย ซึ่งจะนำมาใช้ในระหว่างการทดสอบ โดยจะพยายามระบุกลไกการป้องกันที่มีอยู่ในระบบผ่านการตรวจสอบอย่างละเอียด 3) Threat Modeling ใช้ข้อมูลที่ได้จากการเก็บข้อมูลเพื่อกำหนดวิธีการโจมตีที่มีประสิทธิภาพที่สุด โดยผลลัพธ์จากการสร้างโมเดลนี้จะช่วยในการวางแผนการโจมตีระบบ 4) Vulnerability Analysis การรวมข้อมูลจากขั้นตอนก่อนหน้านี้เพื่อวิเคราะห์และทำความเข้าใจว่าการโจมตีใดบ้างที่สามารถทำได้อย่างมีประสิทธิภาพ 5) Exploitation ตรวจสอบว่าระบบสามารถถูกโจมตีได้หรือไม่ โดยการดำเนินการโจมตีที่อาจทำให้ระบบเสียหาย และเข้าใจจุดที่ควรโจมตี 6) Post Exploitation การเก็บข้อมูลเพิ่มเติมเกี่ยวกับระบบที่ถูกโจมตีสำเร็จเพื่อเข้าถึงภายในระบบได้มากขึ้น 7) Reporting สร้างรายงานเกี่ยวกับกิจกรรมที่ดำเนินการตลอดการทดสอบเจาะระบบ ซึ่งผลลัพธ์ขึ้นอยู่กับความเชี่ยวชาญของนักวิจัยด้านความปลอดภัย ผลลัพธ์ที่ได้รับจากการทดสอบเว็บไซต์จำนวน 10 แห่ง พบว่า 80% ของเว็บไซต์มีความเสี่ยงต่อการโจมตีด้วย SQL Injection ซึ่งชี้ให้เห็นถึงความอ่อนแอของระบบความปลอดภัยในปัจจุบัน นอกจากนี้ งานวิจัยยังแนะนำให้มีการศึกษาเพิ่มเติมเกี่ยวกับเทคนิคในการป้องกันและแก้ไขปัญหาช่องโหว่นี้อย่างมีประสิทธิภาพเพื่อเสริมสร้างมาตรการป้องกันที่ดีขึ้นในการรับมือกับการโจมตีประเภทนี้

P. Cisar and R. Pinter. [5] ได้กล่าวถึงการสำรวจวิธีการและเทคนิคในการเจาะระบบและประเมินความปลอดภัยของเครือข่าย โดยเน้นการใช้เครื่องมือ Kali Linux เพื่อวิเคราะห์ช่องโหว่และการป้องกันการโจมตีที่อาจเกิดขึ้นในระบบต่าง ๆ การศึกษาแบ่งออกเป็นสองประเภทหลัก คือ การโจมตีในฝั่งไคลเอนต์และการโจมตีในฝั่งเซิร์ฟเวอร์วิธีการวิจัยและขั้นตอนการทดสอบดังนี้ 1) Packet Sniffing ใช้เครื่องมือ Aircrack-ng และ airodump-ng เพื่อดักจับข้อมูลจากเครือข่ายโดยเปิดโหมดมอนิเตอร์และวิเคราะห์ข้อมูลที่ดักจับได้ 2) Deauthentication Attack ส่งแพ็กเกจยกเลิกการพิสูจน์ตัวตนเพื่อตัดการเชื่อมต่อของอุปกรณ์ลูกค้า 3) MITM Attack ใช้ ARP Spoofing เพื่อดักฟังข้อมูลระหว่างอุปกรณ์และควบคุมการรับส่งข้อมูล 4) Honeypot Creation สร้าง honeypot เพื่อดึงดูดและตรวจสอบพฤติกรรมของผู้โจมตี 5) Website Hacking ใช้เครื่องมือเช่น Burp Suite เพื่อตรวจสอบช่องโหว่ในเว็บไซต์และทำการโจมตี 6) Best Kali Tools สำรวจและทดสอบเครื่องมือที่

มีใน Kali Linux เพื่อพัฒนาและปรับปรุงระบบความปลอดภัย โดยการโจมตีในฝั่งไคลเอนต์ต้องการการมีส่วนร่วมจากผู้ใช้ เช่น การเปิดไฟล์หรือลิงก์ โดยมีการใช้เทคนิคทางจิตวิทยา (Social Engineering) ในการรวบรวมข้อมูลเกี่ยวกับเป้าหมายและการโจมตีในฝั่งเซิร์ฟเวอร์ไม่ต้องการการมีส่วนร่วมของผู้ใช้ เพียงแค่ทราบที่อยู่ IP ของเป้าหมาย และมักเกิดขึ้นได้ง่ายเมื่อเป้าหมายอยู่ในเครือข่ายเดียวกัน ข้อสรุปงานวิจัยนี้เน้นความสำคัญของการทดสอบและวิเคราะห์ช่องโหว่ในระบบและเครือข่าย เพื่อให้ผู้ดูแลระบบสามารถพัฒนาวิธีการป้องกันที่มีประสิทธิภาพและใช้เครื่องมือจาก Kali Linux ในการดำเนินการโจมตีและปรับปรุงความปลอดภัยในอนาคต การศึกษาเหล่านี้จึงช่วยให้เข้าใจถึงช่องโหว่ต่าง ๆ ที่อาจเกิดขึ้นในระบบ ซึ่งจะเป็นประโยชน์ในการวางแผนและดำเนินการป้องกันที่มีประสิทธิภาพมากขึ้น ผลลัพธ์จากการวิจัย Kali Linux เป็นระบบปฏิบัติการที่มีเครื่องมือที่มีประสิทธิภาพมากมายรวมอยู่ในที่เดียว ซึ่งถูกออกแบบมาเพื่อการโจมตีที่หลากหลายประเภท โดยมีข้อดีที่ชัดเจนคือการมีเครื่องมือแฮกมากมายรวมอยู่ในที่เดียว ทำให้การประเมินช่องโหว่และการทดสอบความปลอดภัยง่ายขึ้น โดยรวมแล้ว Kali Linux เป็นเครื่องมือสำคัญในการวิเคราะห์ช่องโหว่และความปลอดภัย โดยเข้าถึงและใช้งานได้ง่ายสำหรับผู้ที่ต้องการพัฒนาความรู้ความสามารถด้านความปลอดภัยไซเบอร์

A. Bacudio et al. [7] ได้กล่าวถึงการเสนอแนวทางการทดสอบการเจาะระบบ (Penetration Testing) ซึ่งเป็นกระบวนการสำคัญในการประเมินความปลอดภัยของระบบคอมพิวเตอร์ โดยมีจุดประสงค์หลักในการระบุช่องโหว่และความเสี่ยงที่อาจเกิดขึ้นในระบบต่าง ๆ โดยงานวิจัยนี้ผู้วิจัยใช้วิธีการดำเนินการดังนี้ 1) การเตรียมการ (Preparation) การกำหนดเป้าหมายและขอบเขตของการทดสอบ 2) การรวบรวมข้อมูล (Information Gathering) การใช้เทคนิคการสแกนและสำรวจเพื่อเก็บข้อมูลที่เกี่ยวข้อง 3) การวิเคราะห์ช่องโหว่ (Vulnerability Analysis) การประเมินความเสี่ยงและช่องโหว่ที่พบในระบบ 4) การทดลองโจมตี (Exploitation) การทดสอบช่องโหว่ที่พบเพื่อประเมินความเสี่ยงและผลกระทบ 5) การรายงาน (Reporting) การสรุปผลและให้คำแนะนำในการปรับปรุงระบบความปลอดภัย ผลลัพธ์จากการวิจัยพบว่าการทดสอบการเจาะระบบเป็นเครื่องมือที่มีประสิทธิภาพในการระบุช่องโหว่และช่วยให้องค์กรสามารถปรับปรุงความปลอดภัยของระบบได้อย่างมีประสิทธิภาพผลลัพธ์จากการทดสอบช่วยให้องค์กรสามารถกำหนดกลยุทธ์ความปลอดภัยที่เหมาะสม รวมถึงการเลือกทีมทดสอบที่มีความเชี่ยวชาญการดำเนินการทดสอบควรเป็นกระบวนการที่ต่อเนื่อง เนื่องจากภัยคุกคามและเทคโนโลยีมีการเปลี่ยนแปลงอยู่เสมอ

D. Rohmaniah et al. [8] ได้กล่าวถึงการเพิ่มความปลอดภัยของเว็บไซต์โดยใช้วิธีการ Vulnerability Assessment and Penetration Testing (VAPT) อ้างอิงตามมาตรฐาน OWASP Top Ten โดยมีขั้นตอนการวิจัยหลัก 4 ขั้น ได้แก่ 1) การเก็บข้อมูล (Information Gathering) 2) การสแกนหาช่องโหว่ (Vulnerability Scanning) 3) การทดสอบเจาะระบบ (Exploitation) และ 4)

การจัดทำรายงาน (Reporting) ผลการทดสอบพบช่องโหว่หลายประเภท เช่น Clickjacking, Improper HTTP to HTTPS Redirection, Directory Listing และ Sensitive Information Disclosure ทั้งนี้ ผู้วิจัยได้วิเคราะห์ระดับความรุนแรงของช่องโหว่เหล่านี้ด้วย CWE และให้คะแนนด้วย CVSS เพื่อสะท้อนระดับความเสี่ยงตามสภาพแวดล้อมจริง โดยพบว่าช่องโหว่บางประเภทมีความรุนแรงสูงถึงระดับ *critical* เช่น กรณีการเปิดเผยข้อมูลผู้ใช้งาน (Sensitive Information Disclosure) นอกจากนี้ งานวิจัยยังเสนอแนวทางแก้ไขที่ชัดเจน อาทิ การใช้ HTTP Security Headers (X-Frame-Options, Content Security Policy) การลบไฟล์การตั้งค่าที่ไม่ควรเปิดเผย การอัปเดต library หรือ dependencies ที่มีช่องโหว่ และการกำหนดค่า redirect ไปยัง HTTPS อย่างถูกต้อง อีกทั้งได้เน้นย้ำถึงความสำคัญของการทำ VAPT อย่างต่อเนื่องเพื่อช่วยให้องค์กรสามารถลดความเสี่ยงและปกป้องข้อมูลผู้ใช้ได้อย่างมีประสิทธิภาพ

M. Syarifudin et al. [9] ได้กล่าวถึงการวิเคราะห์และแก้ไขช่องโหว่ด้านความปลอดภัยของเว็บไซต์โดยใช้วิธีการ Penetration Testing อ้างอิงตามมาตรฐาน OWASP Top Ten (2021) โดยมีกระบวนการสำคัญ 6 ขั้นตอน ได้แก่ 1) การกำหนดขอบเขต (Scope) 2) การเก็บข้อมูล (Reconnaissance) 3) การวิเคราะห์ช่องโหว่ (Vulnerability Analysis/Scanning) 4) การทดสอบเจาะระบบ (Exploitation) 5) การรายงานผล (Reporting) และ 6) การแก้ไขปัญหา (Mitigation) ผลการทดสอบพบช่องโหว่ในหลายด้าน เช่น Security Misconfiguration, Vulnerable and Outdated Components และ Identification and Authentication Failures โดยตรวจพบปัญหา เช่น การไม่ได้ตั้งค่า Content Security Policy (CSP), การไม่มี X-Frame-Options, การเปิดเผยข้อมูลผ่าน X-Powered-By Header, พอร์ต MySQL ที่เปิดโดยไม่จำเป็น, การไม่มี Anti-CSRF Tokens, และการไม่ใช้ Subresource Integrity (SRI) ในการเรียก external resources ผลการวิจัยเสนอแนวทางแก้ไข เช่น การเพิ่ม HTTP Security Headers (CSP, X-Frame-Options, Referrer-Policy, Permission-Policy) การปิดการเปิดเผยข้อมูลของเซิร์ฟเวอร์ การจำกัดการเข้าถึงพอร์ตด้วย firewall และ rate-limiting การเปิดใช้งาน CSRF Protection ใน framework และการตรวจสอบความปลอดภัยของ external resources ด้วย SRI รวมถึงการอัปเดต dependencies ให้ทันสมัย โดยสรุป งานวิจัยย้ำถึงความสำคัญของการกำหนดนโยบายความปลอดภัยที่เข้มงวดและการทำการทดสอบเจาะระบบอย่างต่อเนื่อง เพื่อช่วยลดความเสี่ยงจากการโจมตีและเพิ่มความมั่นคงปลอดภัยของเว็บไซต์

บทที่3
วิธีการดำเนินงานวิจัย

จากการศึกษางานวิจัย แนวคิด และทฤษฎีที่เกี่ยวข้อง ผู้วิจัยได้นำความรู้มาประยุกต์ใช้ในการทดสอบเจาะระบบและป้องกันการโจมตีเว็บแอปพลิเคชันด้วยระบบปฏิบัติการคาลีลินุกซ์ โดยมีขั้นตอนการดำเนินงาน ดังนี้

3.1 รวบรวมข้อมูลเบื้องต้น

3.2 การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของกลุ่มตัวอย่าง (Vulnerability Scanning) และวิเคราะห์ความเสี่ยงของช่องโหว่ด้วย OWASP ZAP และ Nikto

3.3 การออกแบบและพัฒนาเว็บแอปพลิเคชันต้นแบบ (Web Application Development)

3.4 การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ และวิเคราะห์ความเสี่ยงของช่องโหว่ด้วย OWASP ZAP และ Nikto

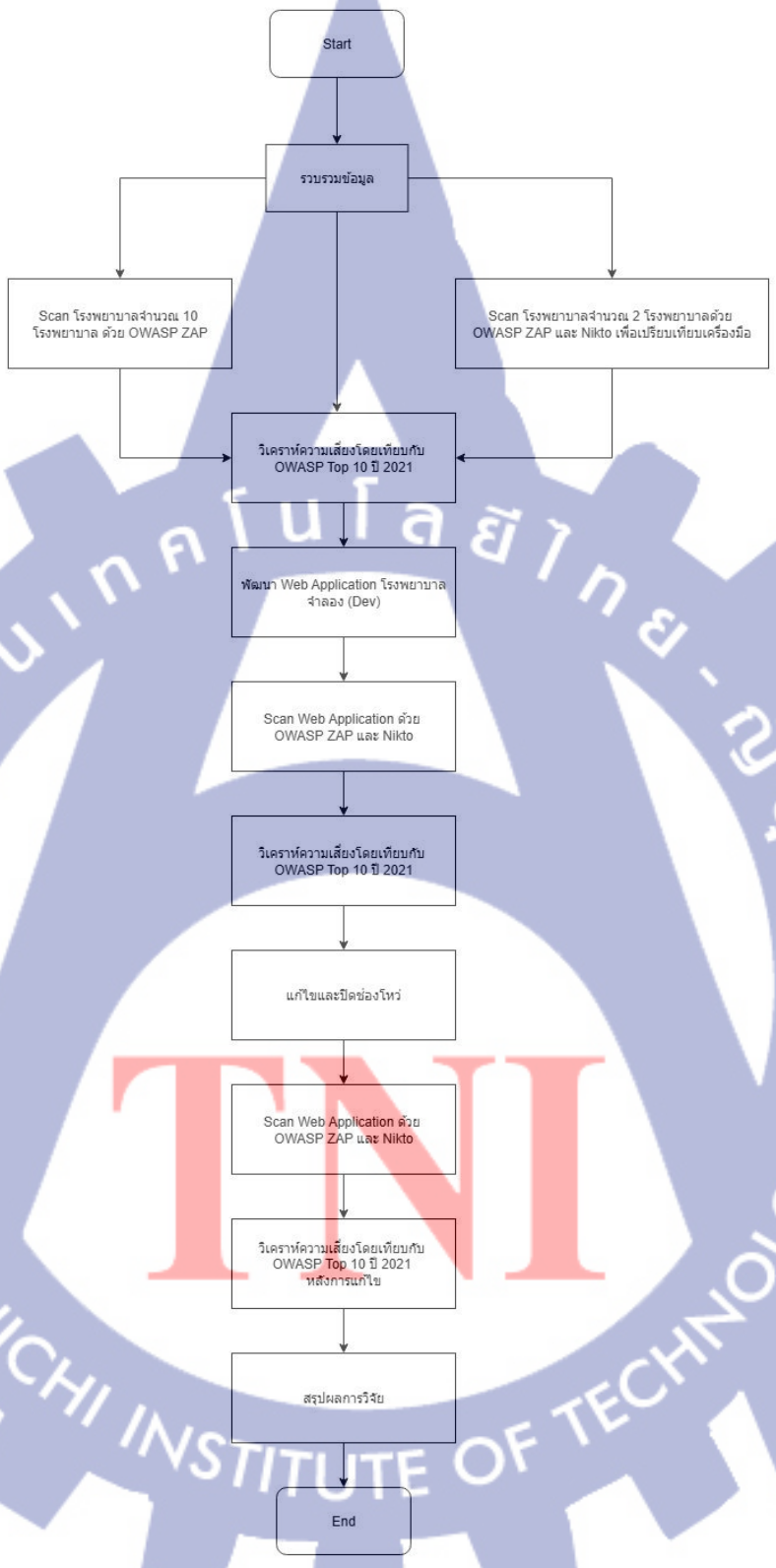
3.5 การทดสอบการโจมตีช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ (Penetration Testing)

3.6 การวิเคราะห์ความเสี่ยงของช่องโหว่ ด้วย CVSS V4.0

3.7 การปรับปรุงและพัฒนาแนวทางป้องกันการโจมตีช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ

3.8 การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ (Vulnerability Scanning)

3.9 สรุปผล



รูปที่ 3.1 ขั้นตอนการดำเนินงานวิจัย

3.1 รวบรวมข้อมูลเบื้องต้น

ในการศึกษา ผู้วิจัยพิจารณาคัดเลือกกลุ่มตัวอย่างโดยใช้เกณฑ์กำหนด เพื่อให้โรงพยาบาลที่ได้รับเลือกสามารถเป็นตัวแทนที่เหมาะสมของกลุ่มโรงพยาบาลที่ต้องการศึกษา เกณฑ์การคัดเลือกประกอบด้วย

1. ความหลากหลายของประเภทโรงพยาบาล

เลือกทั้งโรงพยาบาลขนาดใหญ่ที่มีชื่อเสียงในเขตเมือง และโรงพยาบาลภูมิภาค เพื่อให้ได้ผลการวิจัยที่สะท้อนถึงความแตกต่างของระบบสารสนเทศด้านการแพทย์ในแต่ละบริบท

2. ความพร้อมด้านบริการออนไลน์ (Web Application Availability)

โรงพยาบาลที่คัดเลือกต้องมีเว็บไซต์ที่เปิดให้สาธารณะเข้าถึงได้และมีฟังก์ชันที่ผู้ใช้งานสามารถโต้ตอบได้จริง เช่น ระบบนัดหมายแพทย์ออนไลน์ หรือการลงทะเบียนผู้ป่วย เพื่อให้สามารถทำการทดสอบด้วยเครื่องมือ OWASP ZAP และ Nikto ได้อย่างเป็นรูปธรรม

3. ความสำคัญทางสังคมและเศรษฐกิจ (Relevance & Impact)

เลือกโรงพยาบาลที่มีชื่อเสียงและมีผู้ใช้บริการจำนวนมาก รวมถึงโรงพยาบาลในภูมิภาค เพื่อสะท้อนให้เห็นถึงระดับความเสี่ยงด้านความปลอดภัยที่อาจส่งผลกระทบต่อผู้ใช้ในวงกว้าง

4. ข้อจำกัดทางจริยธรรม (Ethical Considerations)

เลือกเฉพาะเว็บไซต์ที่เปิดสาธารณะ ไม่ทำการเจาะระบบหลังบ้านหรือข้อมูลส่วนบุคคลของผู้ป่วย เพื่อให้อยู่ในขอบเขตจริยธรรมของการวิจัยด้านความปลอดภัยไซเบอร์

5. ความเป็นไปได้เชิงปฏิบัติ (Practicality)

เลือกจำนวนโรงพยาบาลที่สามารถเข้าถึงได้จริงและอยู่ในขอบเขตเวลาของการวิจัย

จากเกณฑ์การคัดเลือกดังกล่าว ผู้วิจัยได้ทำการรวบรวมข้อมูลและพิจารณาเลือกตัวแทนโรงพยาบาล ทั้งหมด 12 แห่ง เพื่อใช้ในการทดสอบสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาลด้วยเครื่องมือ OWASP ZAP และ Nikto โรงพยาบาลที่ได้รับการคัดเลือกนั้นครอบคลุมทั้งโรงพยาบาลของรัฐและโรงพยาบาลเอกชน เพื่อให้การทดลองสามารถดำเนินการได้ครอบคลุมและสามารถวิเคราะห์ผลได้อย่างมีประสิทธิภาพ

โดยชื่อโรงพยาบาลที่เข้าร่วมการศึกษาได้ถูกเข้ารหัสเป็นตัวอักษร A ถึง L แทนการใช้ชื่อจริงเพื่อคงไว้ซึ่งความเป็นส่วนตัวของหน่วยงาน แสดงผลดังตารางที่ 3.1

ตารางที่ 3.1 ตารางการแยกประเภทโรงพยาบาล

อักษรย่อ	ประเภทโรงพยาบาล	อักษรย่อ	ประเภทโรงพยาบาล
A	เอกชน	G	เอกชน
B	เอกชน	H	เอกชน
C	เอกชน	I	รัฐบาล
D	รัฐบาล	J	เอกชน
E	รัฐบาล	K	รัฐบาล
F	เอกชน	L	รัฐบาล

3.2 การสแกนหาช่องโหว่บนเว็บแอปพลิเคชัน (Vulnerability Scanning) และวิเคราะห์ความเสี่ยง (Risk Rating) ของช่องโหว่ ด้วย OWASP ZAP

เมื่อได้ข้อมูลตัวแทนโรงพยาบาลทั้งสิ้น 12 แห่งแล้ว ลำดับถัดมาจึงทำการสแกนหาช่องโหว่บนเว็บแอปพลิเคชัน โดยใช้เครื่องมือ OWASP ZAP และ Nikto แบ่งเป็น 2 กลุ่ม ดังนี้

3.2.1 สแกนหาช่องโหว่ด้วยเครื่องมือ OWASP ZAP จำนวน 10 โรงพยาบาล

วัตถุประสงค์: เพื่อสแกนหาช่องโหว่บนเว็บแอพลิเคชันโรงพยาบาล รวบรวมข้อมูลช่องโหว่เพื่อใช้เป็นข้อมูลสำหรับการสร้างเว็บแอพลิเคชันโรงพยาบาลต้นแบบ

กลุ่มเป้าหมาย: โรงพยาบาล A, B, C, D, E, F, G, H, I และ J

3.2.2 สแกนหาช่องโหว่ด้วยเครื่องมือ OWASP ZAP และ Nikto จำนวน 2 โรงพยาบาล

วัตถุประสงค์: เพื่อเปรียบเทียบประสิทธิภาพของเครื่องมือ

กลุ่มเป้าหมาย: โรงพยาบาล K และ L

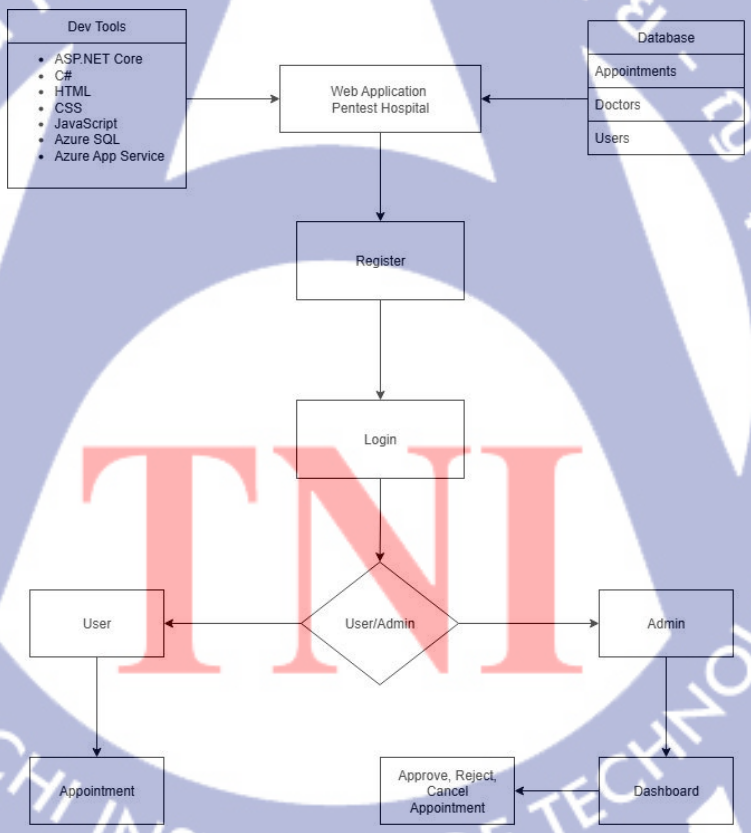
ผู้วิจัยได้ทำการวิเคราะห์ความเสี่ยงของช่องโหว่ที่ตรวจพบ โดยใช้ข้อมูลจากผลการสแกนของเครื่องมือ OWASP ZAP ในกลุ่มโรงพยาบาลจำนวน 10 แห่ง และใช้ข้อมูลจากทั้ง OWASP ZAP และ Nikto ในกลุ่มโรงพยาบาลจำนวน 2 แห่ง รวมเป็นโรงพยาบาลทั้งหมด 12 แห่ง จากนั้นจึงทำการประเมินระดับความรุนแรง (Risk Rating) ของช่องโหว่ที่ตรวจพบ โดยแบ่งออกเป็น 3 ระดับ ได้แก่ สูง (High), กลาง (Medium) และต่ำ (Low) การกำหนดระดับความรุนแรงดังกล่าวมีวัตถุประสงค์เพื่อสะท้อนถึงความร้ายแรงและผลกระทบที่อาจเกิดขึ้นต่อระบบ หากช่องโหว่นั้นถูกนำไปใช้โจมตีจริง โดยระดับสูง (High) หมายถึงช่องโหว่ที่อาจสร้างความเสียหายรุนแรงต่อความลับ ความถูกต้อง และความพร้อมใช้งานของระบบ ซึ่งจำเป็นต้องได้รับการแก้ไขอย่างเร่งด่วน ส่วนระดับกลาง (Medium) หมายถึงช่องโหว่ที่แม้ไม่ก่อให้เกิดความเสียหายรุนแรงทันที แต่ยังมีโอกาสส่งผลกระทบ

ต่อระบบหากถูกละเลย ขณะที่ระดับต่ำ (Low) หมายถึงช่องโหว่ที่มีความเสี่ยงน้อยหรือผลกระทบจำกัด แต่ก็ยังควรถูกติดตามและแก้ไขเพื่อป้องกันความเสี่ยงสะสมในระยะยาว

ดังนั้น การจัดลำดับความรุนแรงของช่องโหว่จึงมีบทบาทสำคัญต่อการวางแผนด้านความมั่นคงปลอดภัยของเว็บแอปพลิเคชันโรงพยาบาล เพราะช่วยให้สามารถจัดลำดับความสำคัญในการแก้ไขได้อย่างเหมาะสม ตลอดจนใช้เป็นแนวทางในการเปรียบเทียบและประเมินประสิทธิภาพของเครื่องมือสแกนช่องโหว่ที่นำมาใช้ในการวิจัยครั้งนี้

3.3 ออกแบบและพัฒนาเว็บแอปพลิเคชัน (Web Application Development)

ในการวิจัยนี้ผู้วิจัยได้ทำการศึกษาเว็บแอปพลิเคชันของโรงพยาบาลและออกแบบโครงสร้าง รวมถึงระบบของเว็บแอปพลิเคชันโรงพยาบาลในประเทศไทยเพื่อจำลองและสร้างสภาพแวดล้อมในให้ใกล้เคียงกับโรงพยาบาลจริงในประเทศไทย เพื่อไม่ให้เกิดผลกระทบต่อระบบหรือข้อมูลที่ถูกใช้งานจริง



รูปที่ 3.2 โครงสร้างเว็บแอปพลิเคชัน

ผู้วิจัยได้ดำเนินการพัฒนาเว็บแอปพลิเคชันสำหรับการจำลองระบบโรงพยาบาล หรือเรียกว่าเว็บแอปพลิเคชันต้นแบบ โดยใช้เทคโนโลยีและเครื่องมือหลัก ดังนี้

ตารางที่ 3.2 เทคโนโลยีในการพัฒนาเว็บแอปพลิเคชันต้นแบบ

Web Framework	ASP.NET Core Razor Pages
Programming Language	C#,HTML,CSS,JavaScript
Database	Azure SQL Database
Server	Azure App Service

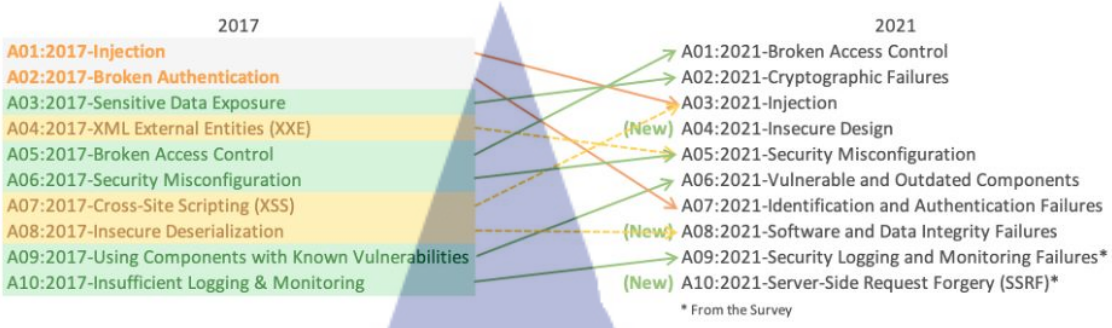
3.4 การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ และวิเคราะห์ความเสี่ยงของช่องโหว่ด้วย OWASP ZAP และ Nikto

ดำเนินการสแกนหาช่องโหว่บนเว็บแอพพลิเคชันของโรงพยาบาลต้นแบบ โดยใช้เครื่องมือภายในระบบปฏิบัติการ Kali Linux ได้แก่ OWASP ZAP และ Nikto เพื่อค้นหาช่องโหว่ที่อาจเกิดขึ้นในเว็บแอปพลิเคชัน ผลการตรวจสอบช่องโหว่ที่ได้จากทั้งสองเครื่องมือนำมาประมวลผลเพื่อเปรียบเทียบความสอดคล้องและความแตกต่างของรายการช่องโหว่ โดยใช้เกณฑ์การนับดังนี้

3.4.1 หากช่องโหว่ใดถูกตรวจพบโดยทั้ง OWASP ZAP และ Nikto จะนับเป็น หนึ่งช่องโหว่ที่ตรงกัน (matched)

3.4.2 หากช่องโหว่ใดถูกตรวจพบโดยเพียงเครื่องมือใดเครื่องมือหนึ่ง จะนับเป็น หนึ่งช่องโหว่เพิ่มเติม (unique)

เพื่อให้การสรุปผลครอบคลุมทั้งรายการที่ซ้อนทับกันและรายการที่แตกต่างกันระหว่างเครื่องมือทั้งสอง เมื่อได้ชุดข้อมูลช่องโหว่ที่ผ่านการรวมและนับแล้ว ผู้วิจัยจึงนำช่องโหว่ทั้งหมดมาจัดกลุ่ม ตามหมวดหมู่ของ OWASP Top 10:2021 [10] เพื่อแสดงการกระจายตัวของช่องโหว่ในแต่ละประเภท ตลอดจนวิเคราะห์แนวโน้มความเสี่ยงที่พบบ่อยในเว็บแอปพลิเคชันโรงพยาบาลต้นแบบ และใช้ผลการวิเคราะห์นี้เป็นฐานในการประเมินความรุนแรง และกำหนดลำดับความสำคัญในการแก้ไขต่อไป



รูปที่ 3.3 OWASP TOP 10 ปี 2021 [10]

3.5 การทดสอบการโจมตีช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ (Penetration Testing)

เมื่อทราบจำนวนช่องโหว่และความเสี่ยงของแต่ละช่องโหว่แล้ว จึงดำเนินการทดสอบการโจมตีช่องโหว่บนเว็บแอปพลิเคชัน ในขั้นตอนนี้ ผู้วิจัยมุ่งเน้นที่การยืนยันเชิงปฏิบัติ (proof-of-concept) โดยมีวัตถุประสงค์ของการทดสอบ ดังนี้

3.5.1 ดำเนินการเจาะระบบเพื่อประเมินว่าช่องโหว่ที่ตรวจพบโดย OWASP ZAP และ Nikto สามารถถูกโจมตีได้จริง (True Positive) หรือเป็นช่องโหว่ที่ไม่ก่อให้เกิดความเสี่ยงในเชิงปฏิบัติ (False Positive)

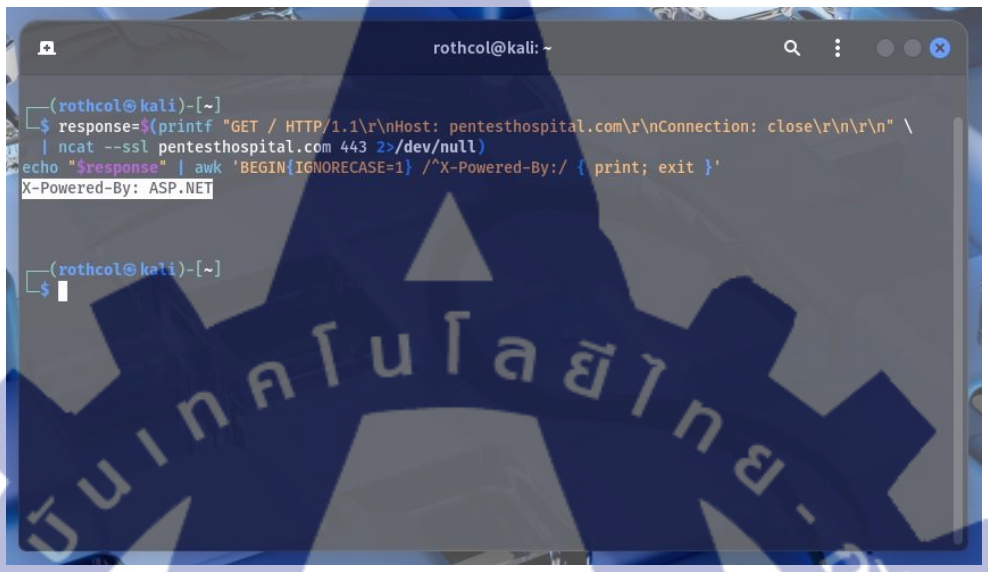
3.5.2 เพื่อให้คะแนน CVSS v4.0 โดยเป็นการประเมินระดับความเสี่ยงเชิงปฏิบัติการโดยให้คะแนนจากผลการทดสอบจริง เพื่อสะท้อนความรุนแรงที่เกิดขึ้นได้จริงในสภาพแวดล้อมเป้าหมาย

3.5.3 จัดลำดับความสำคัญของการแก้ไขตามผลกระทบจริงที่พิสูจน์ได้

โดยการศึกษาวิจัยนี้ จะขอยกตัวอย่างการทดสอบเจาะระบบ 1 ช่องโหว่ เพื่อแสดงแนวทางกระบวนการคิดวิเคราะห์อย่างครบถ้วน โดยมีหลักการพิจารณาเลือกทดสอบการโจมตีช่องโหว่ตามระดับความเสี่ยง High, Medium และ Low ตามลำดับ ผู้วิจัยได้พิจารณาเลือกช่องโหว่ Server Info Leak: X-Powered-By ซึ่งเป็นช่องโหว่ความรุนแรงระดับ Medium ซึ่งเป็นหนึ่งในช่องโหว่ระดับสูงสุดของเว็บแอปพลิเคชันต้นแบบดังนี้

- Server Info Leak: X-Powered-By คือ ช่องโหว่ A05:2021 Security Misconfiguration โดยเป็นการเปิดเผยข้อมูลผ่าน ผ่าน HTTP Response Header ที่ชื่อ X-Powered-By โดยปกติ Header นี้จะแสดงเทคโนโลยีที่ใช้ เช่น ASP.NET หรือ PHP เป็นต้น
- ผู้วิจัยใช้เครื่องมือ ncat (เป็นส่วนหนึ่งของ Nmap) ใน Kali Linux เพื่อส่งคำขอ HTTP/HTTPS แบบดิบ (raw request) และอ่าน HTTP response โดยตรง การทดสอบมุ่งตรวจ header ที่ใช้บอกเทคโนโลยี ซึ่งหากมีการตอบกลับ

จะถือเป็นหลักฐานการรั่วไหลของข้อมูลเทคโนโลยีที่ระบบใช้ ซึ่งอาจช่วยผู้โจมตีค้นหาเวอร์ชันและช่องโหว่ที่สอดคล้องได้ ซึ่งจากผลการทดสอบโดยใช้ ncat ได้ผลลัพธ์ว่ามีการรั่วไหลของข้อมูลเทคโนโลยีที่ระบบใช้จริง ดังภาพที่ 3.4



รูปที่ 3.4 ตัวอย่างช่องโหว่ Server Info Leak: X-Powered-By

โดยสำหรับการศึกษาเพิ่มเติมในเชิงทฤษฎีเกี่ยวกับรูปแบบของโค้ดที่ก่อให้เกิดช่องโหว่และแนวทางการแก้ไข สามารถอ้างอิงได้จากฐานข้อมูลเว็บ Common Weakness Enumeration (CWE) [11] และสำหรับขั้นตอนการทดสอบเจาะระบบ (Penetration testing procedures) แนะนำให้ศึกษาจาก OWASP Web Security Testing Guide (WSTG) [12]

3.6 การวิเคราะห์ความเสี่ยงของช่องโหว่ด้วย OWASP ZAP และ CVSS V4.0

หลังจากดำเนินการทดสอบเจาะระบบในแต่ละช่องโหว่แล้ว เพื่อตรวจสอบและยืนยันสมมติฐานว่าสามารถตรวจพบช่องโหว่เหล่านี้ได้จริงบนเว็บแอปพลิเคชันต้นแบบ ผู้วิจัยได้ทำการประเมินระดับความรุนแรงของช่องโหว่แต่ละรายการโดยใช้เกณฑ์ CVSS v4.0 ซึ่งอ้างอิงจากผลการทดสอบเชิงปฏิบัติ (proof-of-concept) ที่ได้จากการโจมตีจริง จากนั้นจึงทำการให้คะแนนและจัดลำดับความรุนแรงของช่องโหว่จากมากไปหาน้อย เพื่อใช้เป็นแนวทางในการกำหนดลำดับความสำคัญของการแก้ไขและปรับปรุงความปลอดภัยของระบบให้มีประสิทธิภาพสูงสุด

3.7 การปรับปรุงและพัฒนาแนวทางป้องกันการโจมตีช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ

ผู้วิจัยได้กำหนดแนวทางในการปรับปรุงแก้ไขช่องโหว่โดยอ้างอิงจัดระดับความรุนแรงตามเกณฑ์ CVSS v4.0 โดยดำเนินการจากช่องโหว่ที่มีความรุนแรงสูงสุดไปยังช่องโหว่ที่มีความรุนแรงต่ำกว่า ทั้งนี้ ได้มีการแมปช่องโหว่แต่ละรายการเข้ากับ CWE (Common Weakness Enumeration) [12] เพื่อระบุรหัสที่เกี่ยวข้อง และศึกษาวิธีการแก้ไขที่แนะนำจากฐานข้อมูล CWE จากนั้นจึงดำเนินการปรับปรุงและแก้ไขช่องโหว่ตามลำดับความสำคัญที่ได้จัดไว้ เพื่อเสริมสร้างความมั่นคงปลอดภัยของเว็บแอปพลิเคชันต้นแบบอย่างเป็นระบบ ทั้งนี้ แนวทางการปรับปรุงดังกล่าวยังมุ่งลดความเสี่ยงที่อาจส่งผลต่อการปกป้องข้อมูลสำคัญของระบบ และป้องกันผลกระทบที่อาจเกิดจากการจัดการกลไกด้านความปลอดภัยของข้อมูลที่ไม่เหมาะสม [13] ในการทดสอบการปรับปรุงแก้ไขช่องโหว่ขอยกตัวอย่าง 1 ช่องโหว่เพื่อแสดงกระบวนการปรับปรุงแก้ไขช่องโหว่ คือช่องโหว่ Server Info Leak: X-Powered-By ที่ผู้วิจัยได้ยกตัวอย่างการทดสอบเจาะระบบไปก่อนหน้านี้ ซึ่งเป็นช่องโหว่ความรุนแรงระดับ Medium ซึ่งเป็นหนึ่งในช่องโหว่ระดับสูงสุดของเว็บแอปพลิเคชันต้นแบบ ดังนี้

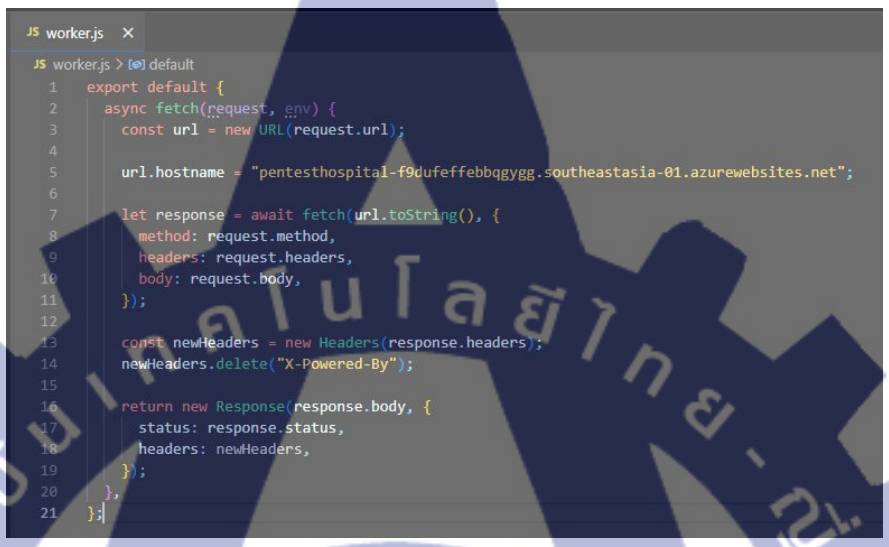
การแก้ไขช่องโหว่ Server Info Leak: X-Powered-By

ช่องโหว่ Server Leaks Information via X-Powered-By HTTP Response Header จัดอยู่ใน CWE-497: Exposure of Sensitive System Information เนื่องจากมีการเปิดเผยข้อมูลเทคโนโลยีภายใน (เช่น ASP.NET, PHP, Express) ผ่าน HTTP Header ไปยังผู้ใช้ภายนอก ซึ่งสอดคล้องกับคำอธิบาย CWE-497 ที่ระบุว่า “ระบบไม่ควรเปิดเผยรายละเอียดภายในให้ผู้ใช้เห็น” แนวทางแก้ไขคือการปิดการส่งค่า X-Powered-By ออกจาก response และเก็บรายละเอียดเทคโนโลยีไว้เฉพาะใน log ภายในเท่านั้น

เนื่องจากงานวิจัยนี้ทำการ Deploy ระบบบน Azure App Service ซึ่งสถาปัตยกรรมของบริการดังกล่าวมี Azure Frontend (Application Request Routing: ARR + IIS layer) คั่นอยู่ด้านหน้าตัว Web Application อีกชั้นหนึ่ง โดยค่า HTTP Response Header บางส่วน เช่น X-Powered-By, Server, และ ARRAffinity ถูกเพิ่มเข้ามาจาก Azure Frontend โดยตรงการแก้ไขผ่านไฟล์ web.config หรือการปรับแต่งภายในระดับ Application ไม่สามารถลบหรือแก้ไข Header เหล่านี้ได้ เนื่องจาก Header ถูก Inject หลังจากที่ Application ส่ง Response กลับมาแล้ว อีกทั้งข้อจำกัดของผู้วิจัยซึ่งใช้ App Service Free Plan ไม่สามารถเข้าถึงหรือตั้งค่า Configuration ของ Azure Frontend ได้โดยตรง

ดังนั้น จึงเลือกใช้แนวทางการ เปลี่ยนเส้นทางการรับส่งข้อมูลผ่าน Cloudflare โดยชี้ Custom Domain ที่ได้ทำการจด Domain และนำ ไปที่ Cloudflare และใช้ Cloudflare Worker ทำหน้าที่เป็น Reverse Proxy บน Edge Server ทำให้สามารถดักจับ Response ก่อนส่งถึงผู้ใช้งาน

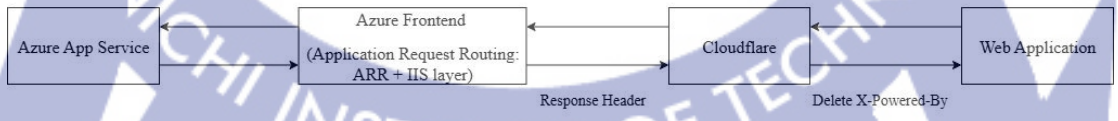
แล้วทำการลบหรือแก้ไข Header ที่ไม่ต้องการได้ เช่น X-Powered-By และ Server วิธีนี้ช่วยแก้ปัญหาการรั่วไหลของข้อมูล (Information Disclosure) ได้ โดยไม่ต้องเข้าไปแก้ไขการตั้งค่าของ Azure โดยตรง ซึ่งสามารถทำได้โดยการเขียน Cloudflare Worker Script ซึ่งใช้ภาษา JavaScript ในการเขียน ดังภาพที่ 3.5



รูปที่ 3.5 ตัวอย่าง Cloudflare Worker Script

โดย Logic การทำงานดังนี้

1. ดัก Request ที่เข้ามาที่ Cloudflare
2. Forward request ต่อไปยัง Azure App Service
3. ดึง response กลับมา
4. ลบ header X-Powered-By ออกจาก response
5. ส่ง response ใหม่ กลับไปให้ Browser



รูปที่ 3.6 Diagram การแก้ไขช่องโหว่ Server Info Leak: X-Powered-By

ผลลัพธ์การแก้ไขช่องโหว่ Server Info Leak: X-Powered-By

หลังจากดำเนินการแก้ไขตามข้อมูลข้างต้นแล้วผู้วิจัยจึงได้ทำการลองทดสอบเจาะระบบใช้เครื่องมือ ncat ใน Kali Linux อีกครั้งเพื่อตรวจสอบว่าช่องโหว่อยู่คงอยู่หรือไม่ ผลลัพธ์คือช่องโหว่ Server Info Leak: X-Powered-By ไม่ถูกแสดงขึ้นมาบนหน้า Terminal จึงสรุปได้ว่า ช่องโหว่ได้ถูกแก้ไขเรียบร้อยแล้ว ดังภาพ 3.7

รูปที่ 3.7 ตัวอย่างผลลัพธ์หลังการแก้ไขช่องโหว่ Server Info Leak: X-Powered-By

3.8 การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ

หลังจากดำเนินการแก้ไขช่องโหว่ตามมาตรการที่วางไว้แล้วผู้วิจัยได้ทำการสแกนซ้ำบนเว็บแอปพลิเคชันต้นแบบโดยใช้เครื่องมือเดียวกับการสแกนครั้งแรก เพื่อยืนยันว่าช่องโหว่ที่เคยตรวจพบได้รับการแก้ไขแล้วหรือไม่ โดยผลการสแกนซ้ำถูกนำมาเปรียบเทียบกับรายงานก่อนการแก้ไขเพื่อพิจารณาว่าช่องโหว่ดังกล่าวถูกปิด ลดความเสี่ยง หรือยังคงปรากฏอยู่ ซึ่งขั้นตอนนี้ช่วยยืนยันประสิทธิภาพของการแก้ไขและใช้เป็นหลักฐานว่าการปรับปรุงด้านความมั่นคงปลอดภัยได้ผลตามที่ตั้งเป้าไว้

3.9 สรุปผล

ในขั้นตอนสุดท้ายของบทนี้ ผู้วิจัยจะดำเนินการแก้ไขช่องโหว่ที่ตรวจพบ โดยอ้างอิงลำดับความรุนแรงตามเกณฑ์ CVSS v4.0 และแนวทางการแก้ไขจากฐานข้อมูล CWE จากนั้นจะทำการสแกนซ้ำบนเว็บแอปพลิเคชันต้นแบบเพื่อยืนยันว่าช่องโหว่ที่ได้รับการแก้ไขถูกปิดหรือไม่

นอกจากนี้ ผู้วิจัยยังวางแผนที่จะทำการเปรียบเทียบผลการทำงานของเครื่องมือ OWASP ZAP และ Nikto เพื่อประเมินความสามารถในการตรวจหาช่องโหว่ ความครอบคลุมของการสแกน และความเหมาะสมในการนำมาใช้เป็นเครื่องมือทดสอบเชิงวิจัย โดยจะนำรายละเอียดเชิงลึกและผลการเปรียบเทียบไปนำเสนอในบทที่ 4 ต่อไป

บทที่ 4
ผลการวิจัย

การศึกษาวิจัยเรื่องการทดสอบเจาะระบบและป้องกันการโจมตีเว็บแอปพลิเคชันด้วยคาลีลินุกซ์ มีวัตถุประสงค์เพื่อศึกษาและทำความเข้าใจวิธีการเจาะระบบรูปแบบต่างๆ ของเว็บแอปพลิชั่นโรงพยาบาลในประไทย ศึกษาแนวทางและวิธีในการป้องกันการถูกโจรกรรมข้อมูลบนเว็บแอปพลิชั่น วิเคราะห์วิธีการตรวจสอบช่องโหว่การโจมตีทางไซเบอร์โดยใช้ระบบปฏิบัติการคาลีลินุกซ์ และศึกษาความเสี่ยงและวิธีการป้องกันจากช่องโหว่ทางไซเบอร์ตาม OWASP TOP 10 ปี 2021 โดยมีผลการศึกษา ดังนี้

4.1 ข้อมูลของกลุ่มตัวอย่าง

การรายงานผลการวิจัยครั้งนี้ ชื่อโรงพยาบาลที่เข้าร่วมการศึกษาได้ถูกเข้ารหัสเป็นตัวอักษร A ถึง L แทนการใช้ชื่อจริง เพื่อคงไว้ซึ่งความเป็นส่วนตัวของหน่วยงาน โดยแบ่งเป็น โรงพยาบาลเอกชนจำนวน 7 โรงพยาบาล และ โรงพยาบาลรัฐบาลจำนวน 5 โรงพยาบาล หลังจากทำการ Scan ด้วย OWASP ZAP และ Nikto ได้ผลลัพธ์ดังนี้

ตารางที่ 4.1 ผลจำนวนช่องโหว่ที่ตรวจพบบนเว็บแอปพลิเคชันของโรงพยาบาลเอกชน

Hospital	Number of Valnerability
โรงพยาบาล A	19
โรงพยาบาล B	23
โรงพยาบาล C	13
โรงพยาบาล F	15
โรงพยาบาล G	15
โรงพยาบาล H	24
โรงพยาบาล J	14
Total	123

ตารางที่ 4.2 ผลจำนวนช่องโหว่ที่ตรวจพบบนเว็บแอปพลิเคชันของโรงพยาบาลรัฐ

Hospital	Number of Valnerability
โรงพยาบาล D	14
โรงพยาบาล E	18
โรงพยาบาล I	30
โรงพยาบาล K	17
โรงพยาบาล L	14
Total	93

จากการเปรียบเทียบจำนวนช่องโหว่ของโรงพยาบาลเอกชนและโรงพยาบาลรัฐ พบว่า โรงพยาบาลที่พบช่องโหว่มากที่สุด คือ โรงพยาบาล I จำนวน 30 ช่องโหว่ ซึ่งเป็นโรงพยาบาลของรัฐ แต่อย่างไรก็ตาม เมื่อพิจารณาผลรวมจำนวนช่องโหว่ทั้งหมดพบว่า โรงพยาบาลเอกชนมีผลรวมจำนวนช่องโหว่มากกว่าโรงพยาบาลรัฐ ซึ่งอาจมีสาเหตุจากการที่กลุ่มตัวอย่างโรงพยาบาลเอกชนมีจำนวนมากกว่ากลุ่มโรงพยาบาลรัฐ

4.2 ผลการการสแกนหาช่องโหว่บนเว็บแอปพลิเคชัน (Vulnerability Scanning) ของโรงพยาบาลกลุ่มตัวอย่าง เทียบกับ OWASP TOP 10 ปี 2021 และการวิเคราะห์ความเสี่ยงของช่องโหว่ (Risk Rating) ด้วย OWASP ZAP และ Nikto

การสแกนหาช่องโหว่บนเว็บแอปพลิเคชัน โดยใช้เครื่องมือ OWASP ZAP และ Nikto แบ่งเป็น 2 กลุ่ม ดังนี้

4.2.1 สแกนหาช่องโหว่ด้วยเครื่องมือ OWASP ZAP จำนวน 10 โรงพยาบาล ผลลัพธ์แสดงดังตารางที่ 4.3 - 4.22

เมื่อได้ผลการสแกนช่องโหว่ของโรงพยาบาลแต่ละโรงพยาบาลแล้ว ได้นำข้อมูลดังกล่าวมาทำการจัดหมวดหมู่ (Mapping) โดยอ้างอิงตามรายการ 10 อันดับ อันดับความเสี่ยงด้านความมั่นคงปลอดภัยที่พบบ่อยที่สุดในเว็บแอปพลิเคชัน (OWASP TOP 10) เพื่อวิเคราะห์จำนวนช่องโหว่ย่อย (Valnerability) ในแต่ละหมวดหมู่อย่างเป็นระบบ

4.2.1 สแกนหาช่องโหว่ด้วยเครื่องมือ OWASP ZAP

ตารางที่ 4.3 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล A

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Absence of Anti-CSRF Tokens	Medium
		Cookie with SameSite Attribute None	Low
		Cookie without SameSite Attribute	Low
		Information Disclosure - Debug Error Messages	Low
		Timestamp Disclosure - Unix	Low
A04:2021	Insecure Design	PII Disclosure	High
		Big Redirect Detected (Potential Sensitive Information Leak)	Low
A05:2021	Security Misconfiguration	CSP: Wildcard Directive	Medium
		Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Application Error Disclosure	Low
		Cookie No HttpOnly Flag	Low
		Secure Pages Include Mixed Content	Low
		Strict-Transport-Security Header Not Set	Low
		X-Content-Type-Options Header Missing	Low
A06:2021	Vulnerable and Outdated Components	CSP: script-src unsafe-eval	Medium
		CSP: script-src unsafe-inline	Medium
		CSP: style-src unsafe-inline	Medium
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.4 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	5
A04:2021	Insecure Design	2
A05:2021	Security Misconfiguration	8
A06:2021	Vulnerable and Outdated Components	3
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.5 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล B

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Absence of Anti-CSRF Tokens	Medium
		Cookie with SameSite Attribute None	Low
		Cookie without SameSite Attribute	Low
		Information Disclosure - Debug Error Messages	Low
		Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low
		Timestamp Disclosure - Unix	Low
A04:2021	Insecure Design	PII Disclosure	High
		Big Redirect Detected (Potential Sensitive Information Leak)	Low
		Old Asp.Net Version in Use	Low
A05:2021	Security Misconfiguration	Application Error Disclosure	Medium
		CSP: Wildcard Directive	Medium
		Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Cookie No HttpOnly Flag	Low
		Cookie Without Secure Flag	Low
		Server Leaks Version Information via "Server" HTTP Response Header Field	Low
		Strict-Transport-Security Header Not Set	Low
		X-AspNet-Version Response Header	Low
		X-Content-Type-Options Header Missing	Low
A06:2021	Vulnerable and Outdated Components	Vulnerable JS Library	High
		CSP: script-src unsafe-inline	Medium
		CSP: style-src unsafe-inline	Medium
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.6 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	6
A04:2021	Insecure Design	3
A05:2021	Security Misconfiguration	10
A06:2021	Vulnerable and Outdated Components	3
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.7 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล C

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Absence of Anti-CSRF Tokens	Medium
		Cookie without SameSite Attribute	Low
		Timestamp Disclosure - Unix	Low
A04:2021	Insecure Design	PII Disclosure	High
A05:2021	Security Misconfiguration	Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Missing Anti-clickjacking Header	Medium
		Application Error Disclosure	Low
		Cookie No HttpOnly Flag	Low
		Cookie Without Secure Flag	Low
		Strict-Transport-Security Header Not Set	Low
		X-Content-Type-Options Header Missing	Low
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.8 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	3
A04:2021	Insecure Design	1
A05:2021	Security Misconfiguration	8
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.9 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล D

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Cookie with SameSite Attribute None	Low
		Cookie without SameSite Attribute	Low
		Timestamp Disclosure - Unix	Low
A04:2021	Insecure Design	PII Disclosure	High
A05:2021	Security Misconfiguration	Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Missing Anti-clickjacking Header	Medium
		Application Error Disclosure	Low
		Cookie Without Secure Flag	Low
		Strict-Transport-Security Header Not Set	Low
		X-Content-Type-Options Header Missing	Low
A06:2021	Vulnerable and Outdated Components	Vulnerable JS Library	High
		Vulnerable JS Library	Medium
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.10 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	3
A04:2021	Insecure Design	1
A05:2021	Security Misconfiguration	7
A06:2021	Vulnerable and Outdated Components	2
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.11 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล E

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Cookie without SameSite Attribute	Low
		Information Disclosure - Debug Error Messages	Low
		Private IP Disclosure	Low
		Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low
		Timestamp Disclosure - Unix	Low
A04:2021	Insecure Design	Big Redirect Detected (Potential Sensitive Information Leak)	Low
A05:2021	Security Misconfiguration	Cloud Metadata Potentially Exposed	High
		Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Missing Anti-clickjacking Header	Medium
		Application Error Disclosure	Low
		Cookie No HttpOnly Flag	Low
		Cookie Without Secure Flag	Low
		Server Leaks Version Information via "Server" HTTP Response Header Field	Low
		Strict-Transport-Security Header Not Set	Low
		X-Content-Type-Options Header Missing	Low
A06:2021	Vulnerable and Outdated Components	Vulnerable JS Library	Medium
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.12 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	5
A04:2021	Insecure Design	1
A05:2021	Security Misconfiguration	10
A06:2021	Vulnerable and Outdated Components	1
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.13 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล F

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Cookie with SameSite Attribute None	Low
		Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low
		Timestamp Disclosure - Unix	Low
A05:2021	Security Misconfiguration	Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Hidden File Found	Medium
		Missing Anti-clickjacking Header	Medium
		Application Error Disclosure	Low
		Server Leaks Version Information via "Server" HTTP Response Header Field	Low
		Strict-Transport-Security Header Not Set	Low
		Strict-Transport-Security Multiple Header Entries (Non-compliant with Spec)	Low
		X-Content-Type-Options Header Missing	Low
A06:2021	Vulnerable and Outdated Components	Vulnerable JS Library	High
		Vulnerable JS Library	Medium
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.14 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	3
A05:2021	Security Misconfiguration	9
A06:2021	Vulnerable and Outdated Components	2
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.15 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล G

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Cookie without SameSite Attribute	Low
		Timestamp Disclosure - Unix	Low
A05:2021	Security Misconfiguration	CSP: Wildcard Directive	Medium
		Content Security Policy (CSP) Header Not Set	Medium
		Application Error Disclosure	Low
		CSP: Notices	Low
		Cookie No HttpOnly Flag	Low
		Cookie Without Secure Flag	Low
		Server Leaks Version Information via "Server" HTTP Response Header Field	Low
		Strict-Transport-Security Header Not Set	Low
		X-Content-Type-Options Header Missing	Low
A06:2021	Vulnerable and Outdated Components	CSP: script-src unsafe-inline	Medium
		CSP: style-src unsafe-inline	Medium
		Vulnerable JS Library	Medium
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.16 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	2
A05:2021	Security Misconfiguration	9
A06:2021	Vulnerable and Outdated Components	3
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.17 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล H

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Absence of Anti-CSRF Tokens	Medium
		Cookie with SameSite Attribute None	Low
		Cookie without SameSite Attribute	Low
		Information Disclosure - Debug Error Messages	Low
		Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low
		Timestamp Disclosure - Unix	Low
A04:2021	Insecure Design	PII Disclosure	High
A05:2021	Security Misconfiguration	Application Error Disclosure	Medium
		CSP: Wildcard Directive	Medium
		Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Missing Anti-clickjacking Header	Medium
		Multiple X-Frame-Options Header Entries	Medium
		Application Error Disclosure	Low
		Cookie Without Secure Flag	Low
		Secure Pages Include Mixed Content	Low
		Server Leaks Version Information via "Server" HTTP Response Header Field	Low
		Strict-Transport-Security Header Not Set	Low
		X-AspNet-Version Response Header	Low
		X-Content-Type-Options Header Missing	Low
A06:2021	Vulnerable and Outdated Components	Vulnerable JS Library	High
		CSP: style-src unsafe-inline	Medium
		Vulnerable JS Library	Medium
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.18 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Valnerability
A01:2021	Broken Access Control	6
A04:2021	Insecure Design	1
A05:2021	Security Misconfiguration	13
A06:2021	Vulnerable and Outdated Components	3
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.19 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล I

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Absence of Anti-CSRF Tokens	Medium
		Cookie without SameSite Attribute	Low
		Information Disclosure - Debug Error Messages	Low
		Private IP Disclosure	Low
		Server Leaks Information via "X-Powered-By" HTTP Response Header Field	Low
		Timestamp Disclosure - Unix	Low
A02:2021	Cryptographic Failures	HTTPS to HTTP Insecure Transition in Form Post	Medium
A04:2021	Insecure Design	PII Disclosure	High
		Old Asp.Net Version in Use	Low
A05:2021	Security Misconfiguration	Cloud Metadata Potentially Exposed	High
		CSP: Wildcard Directive	Medium
		CSP: style-src unsafe-hashes	Medium
		Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Hidden File Found	Medium
		Missing Anti-clickjacking Header	Medium
		Secure Pages Include Mixed Content (Including Scripts)	Medium
		Application Error Disclosure	Low
		Cookie No HttpOnly Flag	Low
		Cookie Without Secure Flag	Low
		Secure Pages Include Mixed Content	Low

ตารางที่ 4.19 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล I (ต่อ)

OWASP TOP10 Matching		Vulnerability	Risk Rating
		Server Leaks Version Information via "Server" HTTP Response Header	Low
		Strict-Transport-Security Header Not Set	Low
		X-AspNet-Version Response Header	Low
		X-Content-Type-Options Header Missing	Low
A06:2021	Vulnerable and Outdated Components	Vulnerable JS Library	High
		CSP: script-src unsafe-inline	Medium
		CSP: style-src unsafe-inline	Medium
		Vulnerable JS Library	Medium
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.20 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	6
A02:2021	Cryptographic Failures	1
A04:2021	Insecure Design	2
A05:2021	Security Misconfiguration	16
A06:2021	Vulnerable and Outdated Components	4
A08:2021	Software and Data Integrity Failures	1

ตารางที่ 4.21 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล J

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Absence of Anti-CSRF Tokens	Medium
		Cookie with SameSite Attribute None	Low
		Information Disclosure - Debug Error Messages	Low
		Timestamp Disclosure - Unix	Low
A04:2021	Insecure Design	PII Disclosure	High
		Big Redirect Detected (Potential Sensitive Information Leak)	Low

ตารางที่ 4.21 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล J (ต่อ)

OWASP TOP10 Matching		Vulnerability	Risk Rating
A05:2021	Security Misconfiguration	Content Security Policy (CSP) Header Not Set	Medium
		Missing Anti-clickjacking Header	Medium
		Application Error Disclosure	Low
		Cookie No HttpOnly Flag	Low
		Cookie Without Secure Flag	Low
		Strict-Transport-Security Header Not Set	Low
		X-Content-Type-Options Header Missing	Low
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ตารางที่ 4.22 แสดงสรุปจำนวนช่องโหว่ย่อย (Vulnerability) ตามการจัดหมวดหมู่ OWASP Top10

OWASP TOP 10		Number of Vulnerability
A01:2021	Broken Access Control	4
A04:2021	Insecure Design	2
A05:2021	Security Misconfiguration	7
A08:2021	Software and Data Integrity Failures	1

4.2.2 สแกนหาช่องโหว่ด้วยเครื่องมือ OWASP ZAP และ Nikto

ผู้วิจัยต้องการเปรียบเทียบเครื่องมือระหว่าง OWASP ZAP และ Nikto เพิ่มเติม แต่เนื่องด้วยข้อจำกัดทางด้านเวลาของผู้วิจัย จึงได้ทำการสแกนหาช่องโหว่เพิ่ม 2 โรงพยาบาล ได้แก่ โรงพยาบาล K และ L เพื่อเปรียบเทียบเครื่องมือ ได้ผลสรุปดังตารางที่ 4.23-4.24

ผลลัพธ์จากการสแกนหาช่องโหว่ด้วย OWASP ZAP

ตารางที่ 4.23 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล K

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Absence of Anti-CSRF Tokens	Medium
		Cookie without SameSite Attribute	Low
A05:2021	Security Misconfiguration	Content Security Policy (CSP) Header Not Set	Medium
		Missing Anti-clickjacking Header	Medium

ตารางที่ 4.23 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันของโรงพยาบาล K (ต่อ)

OWASP TOP10 Matching		Vulnerability	Risk Rating
		Cookie No HttpOnly Flag	Low
		Cookie Without Secure Flag	Low
		Strict-Transport-Security Header Not Set	Low
		X-Content-Type-Options Header Missing	Low
		Private IP Disclosure	Low
		Server Leaks Information via "X-Powered-By"	Low
		Timestamp Disclosure - Unix	Low
A06:2021	Vulnerable and Outdated Components	Vulnerable JS Library	High
		Vulnerable JS Library	Medium
		Vulnerable JS Library	Low
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ผลลัพธ์จากการสแกนหาช่องโหว่ด้วย Nikto

- Missing Anti-clickjacking Header
- Strict-Transport-Security Header Not Set
- X-Content-Type-Options Header Missing
- Cookies Missing Secure Flag
- Cookies Missing HttpOnly Flag
- X-Powered-By Header Exposed
- Uncommon/Informational Headers
- Drupal/WordPress Link Header Disclosure

ตารางที่ 4.24 ผลลัพธ์จากการสแกนหาช่องโหว่ด้วย OWASP ZAP ของโรงพยาบาล L

OWASP TOP10 Matching		Vulnerability	Risk Rating
A01:2021	Broken Access Control	Cookie with SameSite Attribute None	Low
		Cookie without SameSite Attribute	Low
		Timestamp Disclosure - Unix	Low
		Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low

ตารางที่ 4.24 ผลลัพธ์จากการสแกนหาช่องโหว่ด้วย OWASP ZAP ของโรงพยาบาล L (ต่อ)

OWASP TOP10 Matching		Vulnerability	Risk Rating
A05:2021	Security Misconfiguration	Content Security Policy (CSP) Header Not Set	Medium
		Cross-Domain Misconfiguration	Medium
		Missing Anti-clickjacking Header	Medium
		Cookie No HttpOnly Flag	Low
		Cookie Without Secure Flag	Low
		Server Leaks Version Information via "Server" HTTP Response Header Field	Low
		Strict-Transport-Security Header Not Set	Low
		X-Content-Type-Options Header Missing	Low
A08:2021	Software and Data Integrity Failures	Cross-Domain JavaScript Source File Inclusion	Low

ผลลัพธ์จากการสแกนหาช่องโหว่ด้วย Nikto

- Missing Anti-clickjacking Header (X-Frame-Options)
- Strict-Transport-Security (HSTS) Header Not Set
- X-Content-Type-Options Header Missing
- Cookies Missing Secure Flag
- Cookies Missing HttpOnly Flag
- X-Powered-By Header Exposed
- Server Version Information via "Server" Header
- Uncommon/Informational Headers

รวมช่องโหว่ที่พบทั้งหมดจาก 12 โรงพยาบาล ที่ได้จากการสแกนหาช่องโหว่ด้วย OWASP ZAP และ Nikto รวมทั้งหมดได้ 216 ช่องโหว่ ดังตารางที่ 4.25

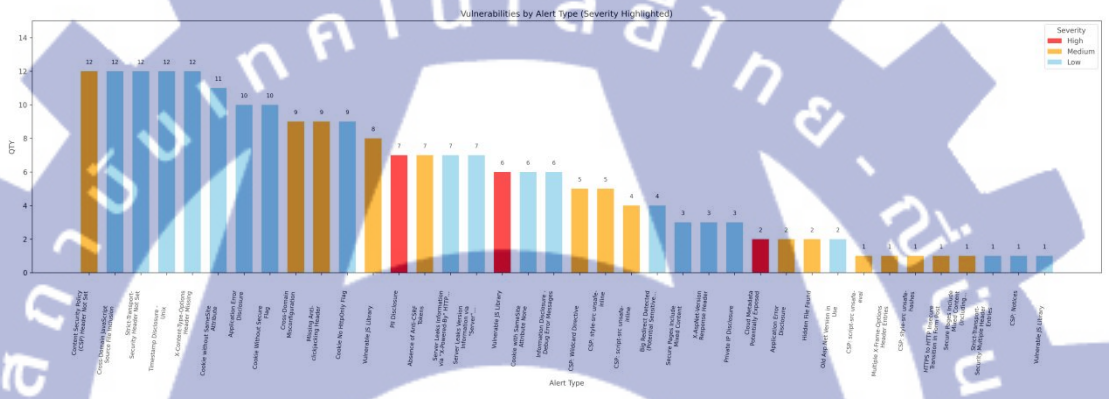
ตารางที่ 4.25 ผลลัพธ์รวมช่องโหว่ 12 โรงพยาบาล

OWASP TOP 10		Number of Valnerability
A01:2021	Broken Access Control	**49**
A02:2021	Cryptographic Failures	**1**

ตารางที่ 4.25 ผลลัพธ์รวมช่องโหว่ 12 โรงพยาบาล (ต่อ)

OWASP TOP 10		Number of Valnerability
A04:2021	Insecure Design	13
A05:2021	Security Misconfiguration	117
A06:2021	Vulnerable and Outdated Components	24
A08:2021	Software and Data Integrity Failures	12
Total		216

แสดงผลในรูปแบบกราฟแผนภูมิสรุปช่องโหว่ย่อยระดับความรุนแรง และ จำนวนดังนี้



รูปที่ 4.1 ช่องโหว่และระดับความรุนแรงทั้งหมด 12 โรงพยาบาล

รวมระดับความรุนแรงที่พบจากการทดสอบเจาะระบบด้วย OWASP ZAP และ Nikto ทั้งหมดจาก 12 โรงพยาบาล

ตารางที่ 4.26 ระดับความรุนแรงทั้งหมด 12 โรงพยาบาล

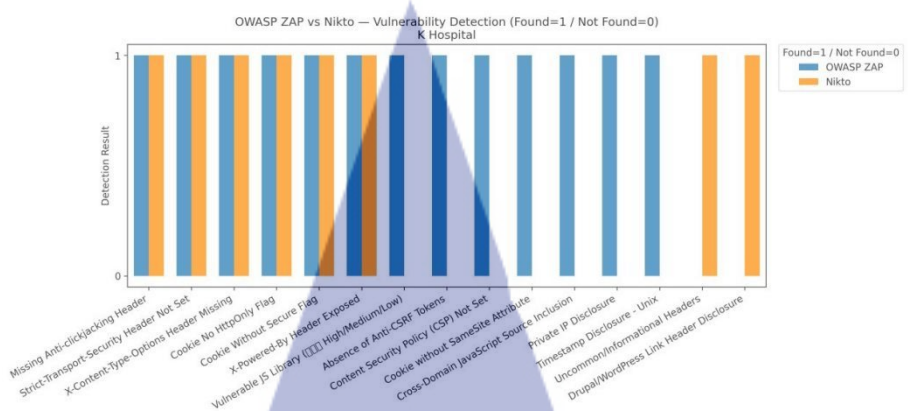
Severity	Number of Valnerability
High	14
Medium	57
Low	145
Summary	216

4.3 ผลลัพธ์การเปรียบเทียบเครื่องมือที่ใช้ในการสแกนหาช่องโหว่ระหว่าง OWASP ZAP และ Nikto

ตารางที่ 4.27 ผลลัพธ์การเปรียบเทียบเครื่องมือโดย OWASP ZAP และ Nikto ของ โรงพยาบาล K

Alert Tags	OWASP ZAP	Nikto
Missing Anti-clickjacking Header	พบ	พบ
Strict-Transport-Security Header Not Set	พบ	พบ
X-Content-Type-Options Header Missing	พบ	พบ
Cookie No HttpOnly Flag	พบ	พบ
Cookie Without Secure Flag	พบ	พบ
X-Powered-By Header Exposed	พบ	พบ
Vulnerable JS Library (รวม High/Medium/Low)	พบ	ไม่พบ
Absence of Anti-CSRF Tokens	พบ	ไม่พบ
Content Security Policy (CSP) Not Set	พบ	ไม่พบ
Cookie without SameSite Attribute	พบ	ไม่พบ
Cross-Domain JavaScript Source Inclusion	พบ	ไม่พบ
Private IP Disclosure	พบ	ไม่พบ
Timestamp Disclosure - Unix	พบ	ไม่พบ
Uncommon/Informational Headers	ไม่พบ	พบ
Drupal/WordPress Link Header Disclosure	ไม่พบ	พบ

ผลลัพธ์การเปรียบเทียบเครื่องมือโดย OWASP ZAP และ Nikto ของ โรงพยาบาล K สรุปเป็น Graph ได้ดังนี้

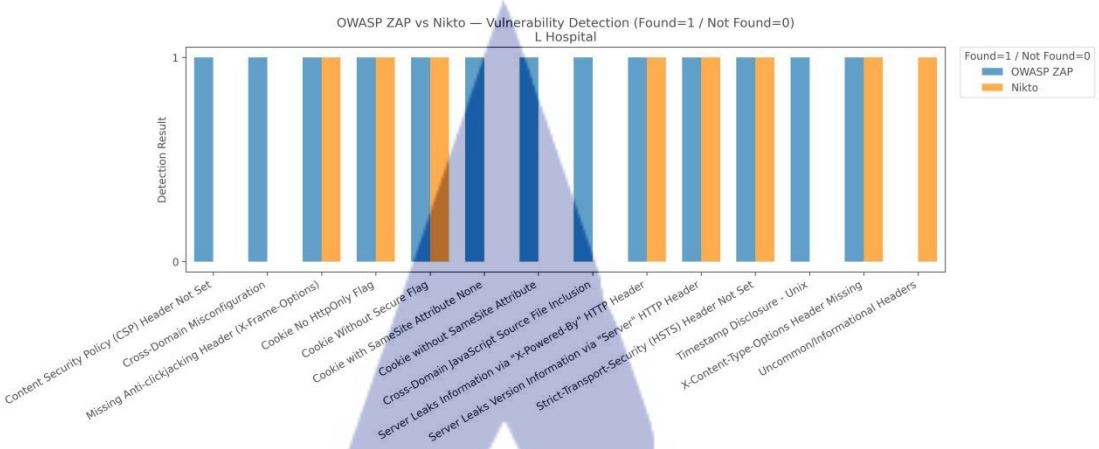


รูปที่ 4.2 ผลลัพธ์เปรียบเทียบการสแกนหาช่องโหว่ โดย OWASP ZAP และ Nikto โรงพยาบาล K

ตารางที่ 4.28 ผลลัพธ์การเปรียบเทียบเครื่องมือโดย OWASP ZAP และ Nikto ของ โรงพยาบาล L

Vulnerability	OWASP ZAP	Nikto
Content Security Policy (CSP) Header Not Set	พบ	ไม่พบ
Cross-Domain Misconfiguration	พบ	ไม่พบ
Missing Anti-clickjacking Header (X-Frame-Options)	พบ	พบ
Cookie No HttpOnly Flag	พบ	พบ
Cookie Without Secure Flag	พบ	พบ
Cookie with SameSite Attribute None	พบ	ไม่พบ
Cookie without SameSite Attribute	พบ	ไม่พบ
Cross-Domain JavaScript Source File Inclusion	พบ	ไม่พบ
Server Leaks Information via "X-Powered-By" HTTP Header	พบ	พบ
Server Leaks Version Information via "Server" HTTP Header	พบ	พบ
Strict-Transport-Security (HSTS) Header Not Set	พบ	พบ
Timestamp Disclosure - Unix	พบ	ไม่พบ
X-Content-Type-Options Header Missing	พบ	พบ
Uncommon/Informational Headers	ไม่พบ	พบ

ผลลัพธ์การเปรียบเทียบเครื่องมือโดย OWASP ZAP และ Nikto ของ โรงพยาบาล L สรุปเป็น Graph ได้ดังนี้



รูปที่ 4.3 ผลลัพธ์เปรียบเทียบการสแกนหาช่องโหว่ โดย OWASP ZAP และ Nikto โรงพยาบาล L

ผลลัพธ์ช่องโหว่ทั้งหมดที่ตรวจพบจากการสแกนเว็บแอปพลิเคชันโรงพยาบาลต้นแบบมาจัดกลุ่มตามหมวดหมู่ของ OWASP TOP 10 2021 และ เปรียบเทียบเครื่องมือ OWASP ZAP และ Nikto ดังตามตารางที่ 4.29 ตรวจพบช่องโหว่ทั้งหมด 14 ช่องโหว่

ตารางที่ 4.29 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันโรงพยาบาลต้นแบบ

OWASP TOP10	Vulnerability	OWASP ZAP	Nikto
A05:2021 Security Misconfiguration	Content Security Policy (CSP) Missing	/	/
	Anti-clickjacking (X-Frame-Options)	/	/
	X-Content-Type-Options Missing	/	/
	Cookie Without Secure Flag	/	/
	Server Info Leak: X-Powered-By	/	/
	Server Version Leak	/	x
	Permissions-Policy Missing	x	/
	Referrer-Policy Missing	x	/
A06:2021 Vulnerable Components	Vulnerable JS Library (jquery-validation)	/	x
A04:2021 Insecure Design	Big Redirect Detected	/	x

ตารางที่ 4.29 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันโรงพยาบาลต้นแบบ (ต่อ)

OWASP TOP10	Vulnerability	OWASP ZAP	Nikto
A01:2021 Broken Access Control	Cookie with SameSite=None	/	x
	Cookie without SameSite	/	x
A02:2021 Cryptographic Failures	Hostname Mismatch	x	/
	BREACH (deflate compression)	x	/

สามารถสรุปผลลัพธ์การเปรียบเทียบเครื่องมือที่ใช้ในการสแกนหาช่องโหว่ระหว่าง OWASP ZAP และ Nikto ได้ดังนี้

4.3.1 ความครอบคลุมในการตรวจจับช่องโหว่

- OWASP ZAP ตรวจพบช่องโหว่ได้กว้างกว่า โดยครอบคลุมทั้งด้านการตั้งค่าความปลอดภัย (Security Misconfiguration), ช่องโหว่การออกแบบที่ไม่ปลอดภัย (Insecure Design) และการควบคุมสิทธิ์ (Broken Access Control) เช่น Cookie with SameSite Attribute Missing/None, Big Redirect Detected และ Vulnerable JS Library เป็นต้น
- Nikto เน้นการตรวจสอบการกำหนดค่าของเว็บเซิร์ฟเวอร์และการตอบสนองของ HTTP headers เช่น Server Version Leak, Permissions-Policy Missing และ Referrer-Policy Missing ซึ่งเป็นรายการบางอย่างที่ ZAP ไม่สามารถตรวจพบได้

ดังนั้น OWASP ZAP จะเหมาะสมสำหรับการทดสอบช่องโหว่เชิงลึกในระดับแอปพลิเคชัน ขณะที่ Nikto เหมาะสำหรับการตรวจสอบพื้นฐานของ Web Server และการตั้งค่าที่ไม่ปลอดภัย การใช้เครื่องมือทั้งสองร่วมกันจึงช่วยให้การวิจัยสามารถครอบคลุมการประเมินความเสี่ยงได้ทั้งในระดับ Application Layer และ Server Layer อย่างครบถ้วน

4.4 ผลลัพธ์การสแกนหาช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ ด้วย OWASP ZAP และ Nikto และผลการวิเคราะห์ความเสี่ยงของช่องโหว่ (Risk Rating) ด้วย OWASP ZAP และ CVSS V4.0

ตารางที่ 4.30 ผลลัพธ์การวิเคราะห์ความเสี่ยงด้วย OWASP ZAP / Nikto และ CVSS V4.0

Exploitation			
OWASP TOP10	Vulnerability	Severity Rating	
		OWASP ZAP/Nikto	CVSS v4.0
A01:2021 Broken Access Control	CWE-1275: Cookie with SameSite=None	Low	1.3 (Low)
	CWE-1275: Cookie without SameSite	Low	2.1 (Low)
A02:2021 Cryptographic Failures	Hostname Mismatch	Unmatch	Unmatch
	BREACH (deflate compression)	Unmatch	Unmatch
A04:2021 Insecure Design	CWE-201: Big Redirect Detected	Low	1.3 (Low)
A05:2021 Security Misconfiguration	CWE-693: Content Security Policy (CSP) Missing	Medium	1.3 (Low)
	CWE-1021: Anti-clickjacking (X-Frame-Options)	Medium	1.3 (Low)
	CWE-693: X-Content-Type-Options Missing	Low	1.3 (Low)
	CWE-497 Server Info Leak: X-Powered-By	Low	5.5 (Medium)
	CWE--497 Server Version Leak	Low	5.5 (Medium)
	CWE-693 Permissions-Policy Missing	Low	5.5 (Medium)
	CWE-200 Referrer-Policy Missing	Low	2.1 (Low)
	Cookie Without Secure Flag	Unmatch	Unmatch

ตารางที่ 4.30 ผลลัพธ์การวิเคราะห์ความเสี่ยงด้วย OWASP ZAP / Nikto และ CVSS V4.0 (ต่อ)

Exploitation			
OWASP TOP10	Vulnerability	Severity Rating	
		OWASP ZAP/Nikto	CVSS v4.0
A06:2021 Vulnerable Components	CWE-1395 Vulnerable JS Library (jquery-validation)	Medium	1.3 (Low)

จากนั้นจึงทำการจัดเรียงลำดับความเสี่ยงช่องโหว่ตามระดับความรุนแรง สูง (High), กลาง (Medium) และต่ำ (Low) เพื่อใช้ในการพิจารณาเลือกแก้ไขช่องโหว่ต่อไป ดังตารางที่ 4.31

ตารางที่ 4.31 ผลลัพธ์การจัดเรียงลำดับความรุนแรงเพื่อใช้ในการพิจารณาเลือกแก้ไขช่องโหว่

Exploitation		
OWASP TOP10	Vulnerability	Severity Rating
A05:2021 Security Misconfiguration	CWE-497 Server Info Leak: X-Powered-By	5.5 (Medium)
A05:2021 Security Misconfiguration	CWE-497 Server Version Leak	5.5 (Medium)
A05:2021 Security Misconfiguration	CWE-693 Permissions-Policy Missing	5.5 (Medium)
A01:2021 Broken Access Control	CWE-1275: Cookie without SameSite	2.1 (Low)
A05:2021 Security Misconfiguration	CWE-200 Referrer-Policy Missing	2.1 (Low)
A01:2021 Broken Access Control	CWE-1275: Cookie with SameSite=None	1.3 (Low)
A04:2021 Insecure Design	CWE-201: Big Redirect Detected	1.3 (Low)
A05:2021 Security Misconfiguration	CWE-693: Content Security Policy (CSP) Missing	1.3 (Low)
A05:2021 Security Misconfiguration	CWE-1021: Anti-clickjacking (X-Frame-Options)	1.3 (Low)

ตารางที่ 4.31 ผลลัพธ์การจัดเรียงลำดับความรุนแรงเพื่อใช้ในการพิจารณาเลือกแก้ไขช่องโหว่ (ต่อ)

Exploitation		
OWASP TOP10	Vulnerability	Severity Rating
A05:2021 Security Misconfiguration	CWE-693: X-Content-Type-Options Missing	1.3 (Low)
A06:2021 Vulnerable Components	CWE-1395 Vulnerable JS Library (jquery-validation)	1.3 (Low)

4.5 ผลลัพธ์การปรับปรุงแก้ไขเพื่อลดหรือปิดช่องโหว่และแนวทางป้องกันการโจมตีช่องโหว่บนเว็บแอปพลิเคชันต้นแบบ

ผู้วิจัยได้ดำเนินการปิดช่องโหว่ปรับปรุงแก้ไขเพื่อลดหรือปิดช่องโหว่ตามแนวทาง CWE และได้ทำการสแกนซ้ำ บนเว็บแอปพลิเคชันต้นแบบโดยใช้เครื่องมือเดียวกับการสแกนครั้งแรก เพื่อยืนยันว่าช่องโหว่ที่เคยตรวจพบได้รับการแก้ไขแล้ว โดยผลลัพธ์การทดสอบการปรับปรุงแก้ไขช่องโหว่ขอยกตัวอย่างการแก้ไข 1 ช่องโหว่ เพื่อแสดงกระบวนการคิดวิเคราะห์และปรับปรุงแก้ไขช่องโหว่

ผู้วิจัยได้ยกตัวอย่างการทดสอบเจาะระบบดังข้อ 3.5 พิจารณาเลือกแก้ไขช่องโหว่ Server Info Leak: X-Powered-By ซึ่งเป็นช่องโหว่ความรุนแรงระดับ Medium และเป็นหนึ่งในช่องโหว่ระดับสูงสุดของเว็บแอปพลิเคชันต้นแบบ จากรูปที่ 4.4 ผลลัพธ์ก่อนแก้ไขช่องโหว่ จากผลรายงานการสแกนพบว่ายังมีช่องโหว่ Server Info Leak: X-Powered-By อยู่ แต่หลังจากได้ทำการแก้ไขแล้ว จากรูปที่ 4.5 ผลลัพธ์หลังแก้ไขจะเห็นได้ว่า Server Info Leak: X-Powered-By ได้หายไปจากผลรายงานการสแกน จึงสรุปผลการวิจัยได้ว่าช่องโหว่ได้ถูกแก้ไขจริง

Alert type	Risk	Count
Content Security Policy (CSP) Header Not Set	Medium	15 (88.2%)
Missing Anti-clickjacking Header	Medium	15 (88.2%)
Vulnerable JS Library	Medium	1 (5.9%)
Big Redirect Detected (Potential Sensitive Information Leak)	Low	1 (5.9%)
Cookie Without Secure Flag	Low	2 (11.8%)
Cookie with SameSite Attribute None	Low	2 (11.8%)
Cookie without SameSite Attribute	Low	2 (11.8%)
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	51 (300.0%)
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	51 (300.0%)
X-Content-Type-Options Header Missing	Low	29 (170.6%)

รูปที่ 4.4 ผลลัพธ์การสแกนก่อนแก้ไขช่องโหว่

Alert type	Risk	Count
Content Security Policy (CSP) Header Not Set	Medium	6 (40.0%)
Missing Anti-clickjacking Header	Medium	5 (33.3%)
Vulnerable JS Library	Medium	1 (6.7%)
Application Error Disclosure	Low	1 (6.7%)
Cookie Without Secure Flag	Low	3 (20.0%)
Cookie with SameSite Attribute None	Low	21 (140.0%)
Cookie without SameSite Attribute	Low	21 (140.0%)
Strict-Transport-Security Header Not Set	Low	2 (13.3%)
X-Content-Type-Options Header Missing	Low	19 (126.7%)

รูปที่ 4.5 ผลลัพธ์การสแกนหลังแก้ไขช่องโหว่

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษาช่องโหว่ในด้านความมั่นคงปลอดภัยของเว็บแอปพลิเคชันโรงพยาบาล โดยใช้เครื่องมือ OWASP ZAP และ Nikto ในการสแกนหาช่องโหว่และวิเคราะห์ความเสี่ยง ผลการศึกษาและวิเคราะห์ และ เปรียบเทียบเครื่องมือ OWASP ZAP และ Nikto สามารถสรุปได้ดังนี้

5.1.1 การสแกนเว็บแอปพลิเคชันโรงพยาบาลจำนวน 12 แห่ง พบว่าโรงพยาบาลที่ตรวจพบช่องโหว่มากที่สุดคือโรงพยาบาลของรัฐบาล อย่างไรก็ตาม เมื่อพิจารณาผลรวมของช่องโหว่ทั้งหมด พบว่า กลุ่มโรงพยาบาลเอกชนมีจำนวนช่องโหว่รวมมากกว่ากลุ่มโรงพยาบาลรัฐ ซึ่งอาจมีสาเหตุจากการที่กลุ่มตัวอย่างโรงพยาบาลเอกชนมีจำนวนมากกว่าโรงพยาบาลรัฐ

5.1.2 ช่องโหว่ที่ได้ทำการตรวจพบส่วนใหญ่จัดอยู่ในหมวด Security Misconfiguration (A05:2021) และ Broken Access Control (A01:2021) ตามเกณฑ์ของ OWASP Top 10:2021 ซึ่งสอดคล้องในรายงานของ OWASP ว่าความเสี่ยงดังกล่าวเป็นปัญหาที่พบมากในเว็บแอปพลิเคชัน

5.1.3 เมื่อเปรียบเทียบเครื่องมือ พบว่า OWASP ZAP ตรวจพบช่องโหว่เชิงตรรกะของเว็บแอปพลิเคชันได้ละเอียดกว่าในขณะที่เครื่องมือ Nikto นั้นตรวจพบช่องโหว่ด้านการตั้งค่าเว็บเซิร์ฟเวอร์ (Server Misconfiguration) ได้ชัดเจนกว่า การใช้เครื่องมือทั้งสองร่วมกันจึงช่วยเพิ่มความครอบคลุมของการวิเคราะห์

5.1.4 ผลการประเมินความเสี่ยง (Risk Rating) ถูกแบ่งออกเป็น สูง (High), ปานกลาง (Medium) และต่ำ (Low) ช่วยให้สามารถจัดลำดับความสำคัญของการแก้ไขช่องโหว่ได้อย่างมีระบบ ซึ่งสอดคล้องกับประกาศในราชกิจจานุเบกษา เรื่อง มาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับเว็บไซต์ ประกาศใช้เมื่อวันที่ 16 กันยายน พ.ศ.2568 ข้อ 6. ข้อกำหนดการดำเนินการและการรักษาความปลอดภัยสำหรับเว็บไซต์ระบุว่า หน่วยงานจะต้องมีการจัดการทรัพย์สิน (Asset Management) การประเมินความเสี่ยง (Risk Assessment) การประเมินช่องโหว่และการทดสอบเจาะระบบ (Vulnerability Assessment and Penetration Tesing) ให้เป็นไปตามประมวลแนวทางปฏิบัติและกรอบมาตรฐานด้านการรักษาความมั่นคงปลอดภัยไซเบอร์

5.2 ปัญหาและอุปสรรคในการวิจัย

5.2.1 ข้อจำกัดด้านของเวลาการดำเนินงานวิจัยนั้นอยู่ภายใต้กรอบเวลาที่จำกัด เนื่องจากลักษณะของงานวิจัยต้องใช้เวลาค่อนข้างมาก ส่งผลให้เกิดความยากลำบากในการดำเนินการและการจัดการแผนงานให้เป็นไปตามที่กำหนด

5.2.2 ผลลัพธ์จากเครื่องมือ OWASP ZAP และ Nikto มีความแตกต่างกันบางส่วน จำเป็นต้องใช้การวิเคราะห์เพิ่มเติมในการปรับผลลัพธ์ให้สามารถนำมาเปรียบเทียบและรวมเป็นข้อมูลเดียวกัน

5.2.3 ข้อจำกัดด้านงบประมาณทำให้ไม่สามารถจัดเตรียมสภาพแวดล้อมในด้านเซิร์ฟเวอร์ (Server Environment) ได้อย่างสมบูรณ์ ส่งผลให้การตั้งค่า (Configuration) และการทดสอบบางประเด็นไม่สามารถทำได้เต็มรูปแบบ

5.3 ข้อเสนอแนะงานวิจัย

5.3.1 สำหรับโรงพยาบาล ควรมีการตรวจสอบช่องโหว่เป็นประจำ โดยใช้มากกว่าหนึ่งเครื่องมือเพื่อให้ได้ผลการวิเคราะห์ที่รอบด้านรวมถึงควรปรับปรุงในด้านการตั้งค่าความปลอดภัยของเว็บเซิร์ฟเวอร์ให้สอดคล้องกับมาตรฐานที่ระบุใน ราชกิจจานุเบกษา [14]

5.3.2 สำหรับผู้วิจัยในอนาคต ควรขยายการศึกษาไปยังระบบสารสนเทศประเภทอื่นของโรงพยาบาล เช่น Mobile Application

5.3.3 ควรพัฒนา Framework การประเมินความเสี่ยงเฉพาะสำหรับโรงพยาบาลไทย ที่ผสานเกณฑ์จาก OWASP Top 10:2021 เข้ากับมาตรฐานความมั่นคงปลอดภัยที่ประกาศในราชกิจจานุเบกษา เพื่อเป็นแนวทางที่เหมาะสมและปฏิบัติได้จริงในระดับองค์กร


สถาบันเทคโนโลยีไทย-ญี่ปุ่น
THAI - NICHI INSTITUTE OF TECHNOLOGY
TNI

บรรณานุกรม

บรรณานุกรม

[1] K. Nagendran et al., "Web application penetration testing," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 10, pp. 1029 - 1035, August 2019.

[2] S. Yadav and P. Singh, "Web application and penetration testing," *Journal of Informatics Electrical and Electronics Engineering*, vol. 1, no. 3, pp. 1 - 11, December 2021.

[3] A. Anas et al., "Survey on detecting and preventing web application broken access control attacks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 14, no. 1, pp. 772 - 781, February 2024.

[4] A. Alanda et al., "Web application penetration testing using SQL injection attack," *International Journal on Informatics Visualization*, vol. 5, no 3, pp. 320 - 326, September 2021.

[5] P. Cisar and R. Pinter, "Some ethical hacking possibilities in Kali Linux environment," *Journal of Applied Technical and Educational Sciences (jATES)*, vol. 9, no. 4, pp. 129 - 149, December 2019.

[6] K. Baker, "*12 Most Common Types of Cyberattack,*" [Online]. Available: https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/common-cyberattacks/. [Accessed: May 13, 2024].

[7] A. Bacudio et al., "An overview of penetration testing," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 3, no. 6, pp. 19 - 38, November 2011.

[8] D. Rohmaniah et al., "Enhancing website security using vulnerability assessment and penetration testing (VAPT) based on OWASP top ten," *Journal of Applied Informatics and Computing (JAIC)*, vol. 9, no. 2, pp. 404 - 411, April 2025.

[9] M. Syarifudin at el., "Web security vulnerability analysis and mitigation," *Journal of Artificial Intelligence and Engineering Applications*, vol. 4, no. 3, pp. 1830 - 1834, June 2025.

[10] OWASP Foundation, Inc., “*Top 10 Web Application Security Risks*,” [Online]. Available: https://owasp.org/www-project-top-ten/. [Accessed: September 10, 2021].

[11] The MITRE Corporation, “*Common Weakness Enumeration (CWE),*” [Online]. Available: https://cwe.mitre.org/. [Accessed: September 16, 2025].

[12] OWASP Foundation, Inc., “*OWASP Web Security Testing Guide*,” [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/. [Accessed: October 28, 2025].

[13] OWASP Foundation, Inc., “*A02:2021 – Cryptographic Failures,*” [Online]. Available: https://owasp.org/Top10/A02_2021-Cryptographic_Failures. [Accessed: September 10, 2025].

[14] ราชกิจจานุเบกษา, “ *ประกาศคณะกรรมการการรักษาความมั่นคงปลอดภัยไซเบอร์แห่งชาติ เรื่อง มาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับเว็บไซต์ พ.ศ. 2565*,” เล่ม 142 ตอนพิเศษ 305 ง, 16 กันยายน 2565. [Online]. Available:https://drive.ncsa.or.th/s/JLWCNF6ppaRjYmG?dir=/&editing=false&openfile=true. [Accessed: 17 กันยายน 2568].