



Secure Coding -
Office of the Basic
Education Commission

สารบัญ

1. ความสำคัญของการพัฒนาแอปพลิเคชันอย่างปลอดภัย
2. ภัยจากการใช้ AI ในการ Coding
3. มาตรฐานที่ช่วยในการพัฒนาแอปพลิเคชันอย่างปลอดภัย
4. OWASP TOP 10 คืออะไร
5. ตัวอย่างการ พัฒนาแอปพลิเคชันที่ไม่ปลอดภัย, วิธีการแก้ไข
6. แนวทางที่ควรดำเนินการสำหรับการพัฒนาแอปพลิเคชันอย่างปลอดภัย
7. รูปแบบการพัฒนาแอปพลิเคชันในปัจจุบัน และเครื่องมือที่ใช้
8. สรุปภาพรวมการอบรม
9. ตอบคำถามผู้เข้าอบรม

ความสำคัญของการพัฒนาแอปพลิเคชันอย่างปลอดภัย

การพัฒนาแอปพลิเคชันอย่างปลอดภัยเป็นสิ่งสำคัญอย่างยิ่งในยุคดิจิทัลปัจจุบัน โดยมีหลายเหตุผลที่เน้นความสำคัญของการมีแนวปฏิบัติที่ปลอดภัยในการพัฒนาแอปพลิเคชัน ดังนี้



ป้องกันการโจมตีทางไซเบอร์



ปกป้องข้อมูลส่วนบุคคล



ลดค่าใช้จ่ายในระยะยาว



เพิ่มความน่าเชื่อถือและความปลอดภัยขององค์กร



การปฏิบัติตามมาตรฐานและข้อกำหนดทางกฎหมาย

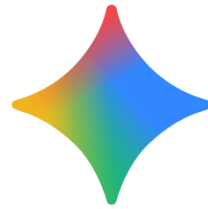


การส่งเสริมแนวปฏิบัติที่ดีในองค์กร

ภัยเงียบของการใช้ AI ช่วยในการเขียนโค้ด



ChatGPT



Gemini



Copilot



Claude

ภัยที่อาจเกิดจากการใช้ AI

การรั่วไหลของข้อมูลที่เป็นความลับ (Leaking Sensitive Data)

นักพัฒนาอาจเผลอใส่ API Keys, รหัสผ่าน หรือข้อมูลโครงสร้างพื้นฐานขององค์กรลงใน Prompt เพื่อให้ AI ช่วยแก้ปัญหา ซึ่งข้อมูลเหล่านี้อาจถูกนำไปใช้เทรนโมเดลต่อและอาจหลุดไปสู่สาธารณะได้



ภัยที่อาจเกิดจากการใช้ AI

ช่องโหว่ความปลอดภัยในโค้ด (Security Vulnerabilities)

AI มักจะแนะนำโค้ดที่ "ทำงานได้" แต่ไม่ได้ "ปลอดภัยที่สุด" เช่น โค้ดที่เสี่ยงต่อ SQL Injection, Cross-Site Scripting (XSS) หรือการไม่ทำ Input Validation



ภัยที่อาจเกิดจากการใช้ AI

การละเมิดลิขสิทธิ์ซอฟต์แวร์ (Intellectual Property & Licensing)

AI อาจนำโค้ดที่มีสัญญาอนุญาตแบบ Open Source ที่เข้มงวด เช่น ห้ามนำมาใช้ในเชิงพาณิชย์ มาเป็นคำแนะนำ หากเรานำไปใช้ในโปรเจกต์เชิงพาณิชย์โดยไม่ตรวจสอบ อาจถูกฟ้องร้องภายหลังได้



ภัยที่อาจเกิดจากการใช้ AI

การเข้าใจผิดของ AI เรื่องแพ็คเกจ (AI Package Hallucination)

AI อาจแนะนำให้ใช้ Library หรือ Package ที่ไม่มีอยู่จริง แฮกเกอร์อาจสร้าง Package ปลอมในชื่อนั้นๆ ทิ้งไว้ (Supply Chain Attack) เมื่อนักพัฒนาสั่งติดตั้งตาม AI ก็จะได้รับมัลแวร์ทันที



ภัยที่อาจเกิดจากการใช้ AI

การขาดความเข้าใจในสิ่งที่เขียน (Skill Erosion)

หากนักพัฒนา Copy-Paste โค้ดโดยไม่พยายามทำความเข้าใจ เมื่อเกิด Bug ร้ายแรงในระบบ ทำให้หาจุดผิดพลาดเกิด bug และแก้ไขได้ยาก

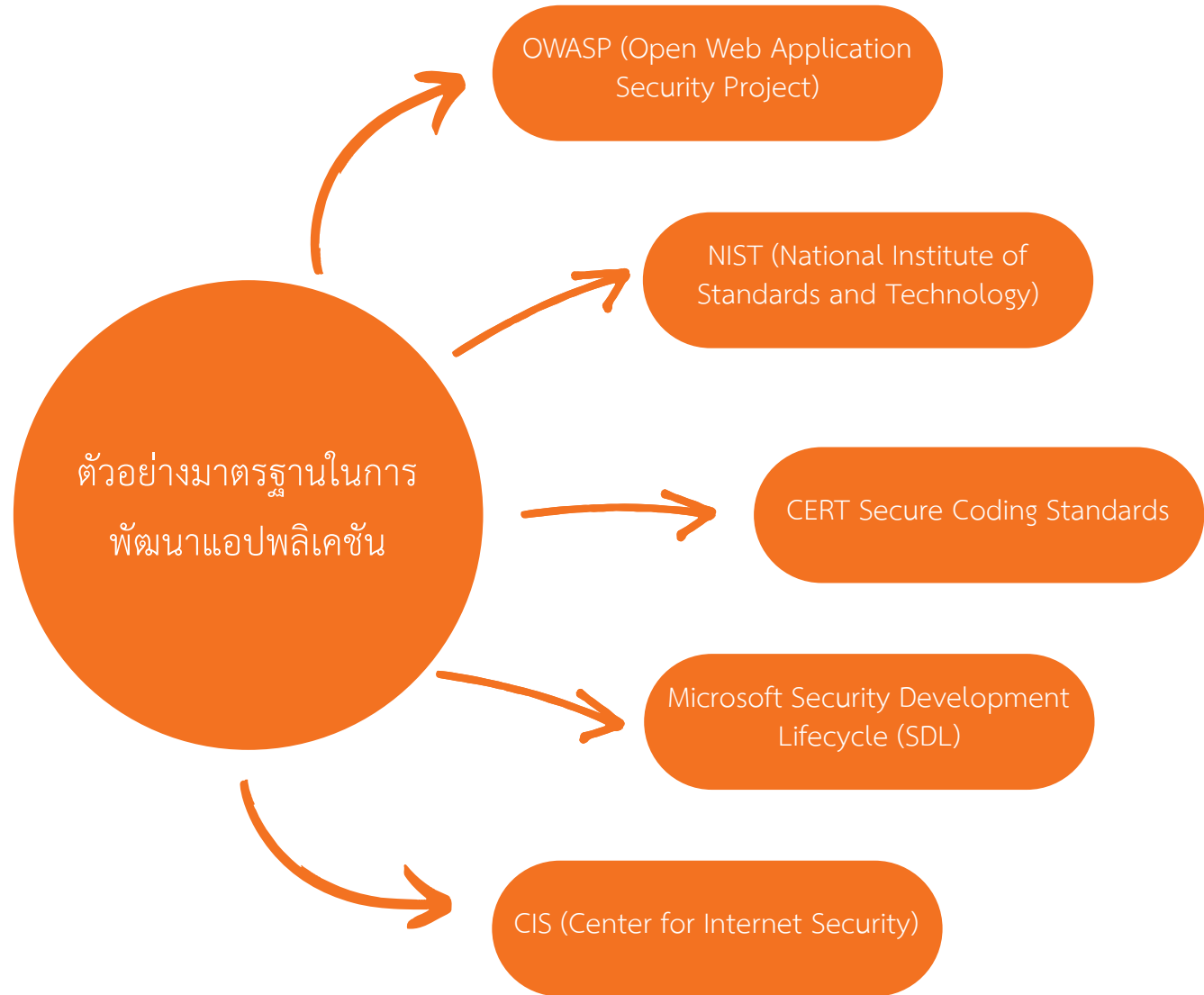


แนวทางการแก้ไข

- ทุกบรรทัดที่ AI เขียน ต้องถูก Review โดยมนุษย์ที่มีความเชี่ยวชาญเสมอ
- ใช้เครื่องมืออย่าง SAST (Static Application Security Testing) เพื่อสแกนโค้ดทุกครั้ง ก่อน Merge
- เวลาเราสั่ง AI เราควรระบุไปเลยว่า 'ให้เขียนโค้ดโดยคำนึงถึงความปลอดภัยระดับสูงสุด' หรือ 'ห้ามใช้ Library ภายนอกที่ไม่ได้มาตรฐาน'
- องค์กรต้องมี Guideline ที่ชัดเจนว่าข้อมูลไหนใส่ใน AI ได้ (Non-sensitive) และข้อมูลไหนห้ามใส่เด็ดขาด“

มาตรฐานที่ช่วยในการ พัฒนาแอปพลิเคชัน อย่างปลอดภัย

การเขียนโค้ดอย่างปลอดภัยเป็นกระบวนการ
ที่จำเป็นต้องปฏิบัติตามมาตรฐานและ
แนวทางที่ถูกต้องเพื่อป้องกันช่องโหว่และการ
โจมตีทางไซเบอร์ มีหลายมาตรฐานและ
แนวทางที่ช่วยให้การเขียนโค้ดมีความ
ปลอดภัยยิ่งขึ้น



OWASP TOP 10 (2025) คืออะไร

รายการช่องโหว่ด้านความปลอดภัยที่พบบ่อยและมีความสำคัญมากที่สุดในแอปพลิเคชันเว็บ จัดทำโดย Open Web Application Security Project (OWASP) ซึ่งเป็นองค์กรไม่แสวงหาผลกำไรที่มุ่งเน้นการปรับปรุงความปลอดภัยของซอฟต์แวร์ รายการนี้จะถูกอัปเดตเป็นระยะเพื่อสะท้อนถึงภัยคุกคามและช่องโหว่ที่พบบ่อยที่สุดในขณะนั้น

01

A1 - Broken Access Control

02

A2 - Security Misconfiguration

03

A3 - Software Supply Chain Failures

04

A4 - Cryptographic Failures

05

A5 - Injection

06

A6 - Insecure Design

07

A7 - Authentication Failures

08

A8 - Software or Data Integrity Failures

09

A9 - Security Logging & Alerting Failures

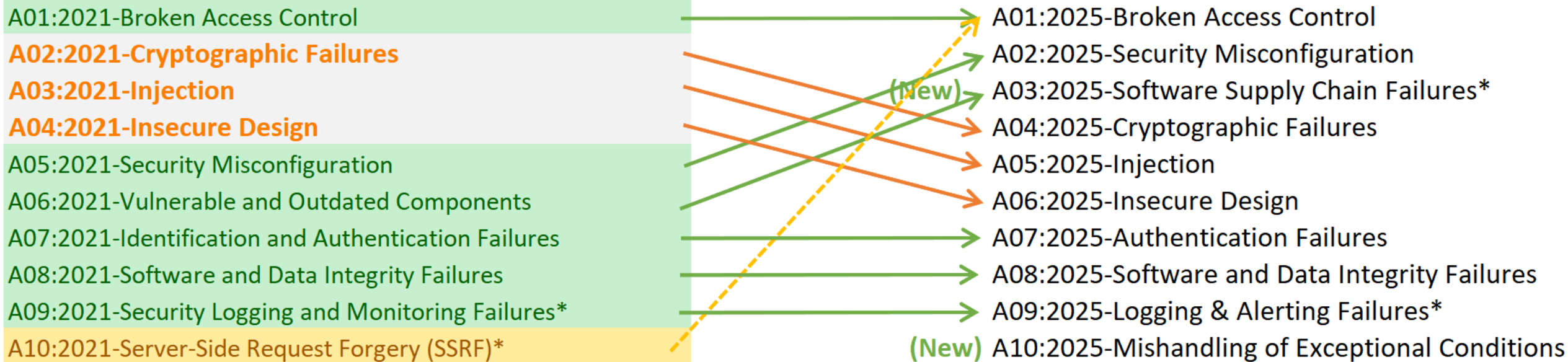
10

A10 - Mishandling of Exceptional Conditions

OWASP TOP 10 (2025) คืออะไร

2021

2025

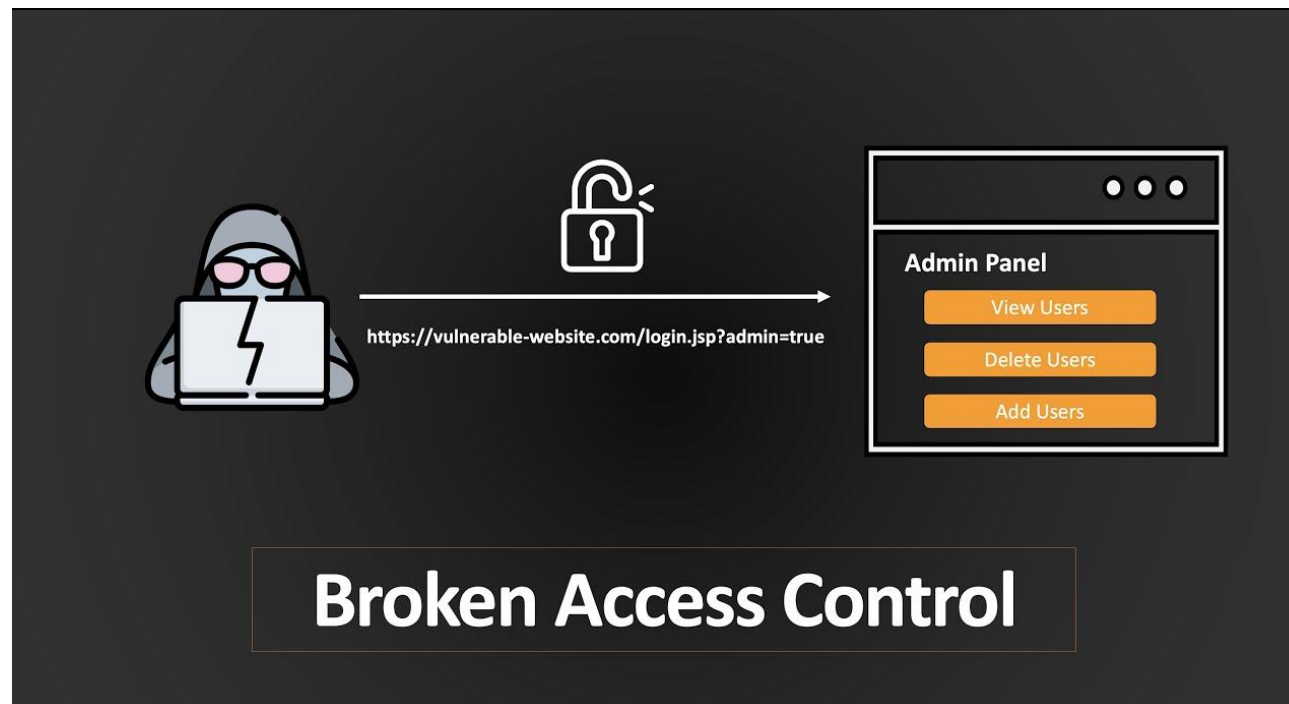


* From the Survey

* From the Survey

A1 - Broken Access Control (การควบคุมการเข้าถึงที่ไม่ถูกต้อง)

การที่ผู้ใช้สามารถเข้าถึงข้อมูลหรือฟังก์ชันที่ตนเองไม่มีสิทธิ์ ระบบควรจะมีการจำกัดสิทธิ์ (Authorization) เพื่อให้ User ทั่วไปเข้าได้เฉพาะข้อมูลของตนเอง และ Admin เท่านั้นที่เข้าหลังบ้านได้ แต่หากระบบมีช่องโหว่นี้ Hacker จะสามารถข้ามผ่านข้อจำกัดเหล่านั้นได้



ตัวอย่างการโจมตี Scenario #1:

แอปพลิเคชันใช้ข้อมูลที่ยังไม่ได้รับการตรวจสอบในการเรียกใช้ SQL เพื่อเข้าถึงข้อมูลของบัญชี



```
// นำค่าที่รับมาจากผู้ใช้ไปใส่ในคำสั่ง SQL
statement.setString(1, requestedAccountId);
// ประมวลผลและดึงข้อมูลออกมา
ResultSet accountData = statement.executeQuery();
```

ผู้โจมตีเพียงแค่แก้ไขพารามิเตอร์ 'requestedAccountId' ในเบราว์เซอร์เพื่อส่งหมายเลขบัญชีที่ต้องการ หากไม่มีการตรวจสอบอย่างถูกต้อง ผู้โจมตีสามารถเข้าถึงข้อมูลบัญชีผู้ใช้อื่นๆ ได้

<https://example.com/app/accountInfo?acct=2>

ตัวอย่างการโจมตี Scenario #1:

ผู้โจมตีเพียงแค່ทราบ URL ของแอดมิน จากนั้นใช้เบราว์เซอร์เข้าไปยัง URL เป้าหมาย
ที่ปกติแล้วต้องใช้สิทธิ์ของแอดมินสำหรับการเข้า URL นั้นโดยตรง



<https://example.com/app/getappInfo>

https://example.com/app/admin_getappInfo

หากผู้ใช้ที่ไม่ได้รับการยืนยันสามารถเข้าถึงหน้าใดหน้าหนึ่งได้ นั่นถือว่าเป็นช่องโหว่
หากผู้ใช้ที่ไม่ใช่ผู้ดูแลระบบสามารถเข้าถึงหน้าผู้ดูแลระบบได้ นี่ก็เป็นช่องโหว่เช่นกัน

WE LIKE TO SHOP



Babbage Web Spray

★★★★☆ \$25.94

[View details](#)



WTF? - The adult party game

★★★☆☆ \$86.08

[View details](#)



The Alternative Christmas Tree

★★★★★ \$79.37

[View details](#)



Six Pack Beer Belt

★★★★★ \$18.45

[View details](#)



Users

wiener - [Delete](#)

carlos - [Delete](#)

วิธีการแก้ไข และป้องกัน



ใช้หลักการ "Deny by Default" (ปฏิเสธไว้ก่อน)

ระบบควรจะ **ไม่อนุญาต** ให้ใครเข้าถึงอะไรเลย จนกว่าจะมีการระบุสิทธิ์ที่ชัดเจน



ตรวจสอบค่า parameter ที่ฝั่ง **Client** และ **Sever** ไม่ควรตรวจสอบแค่ฝั่งใดฝั่งเดียว

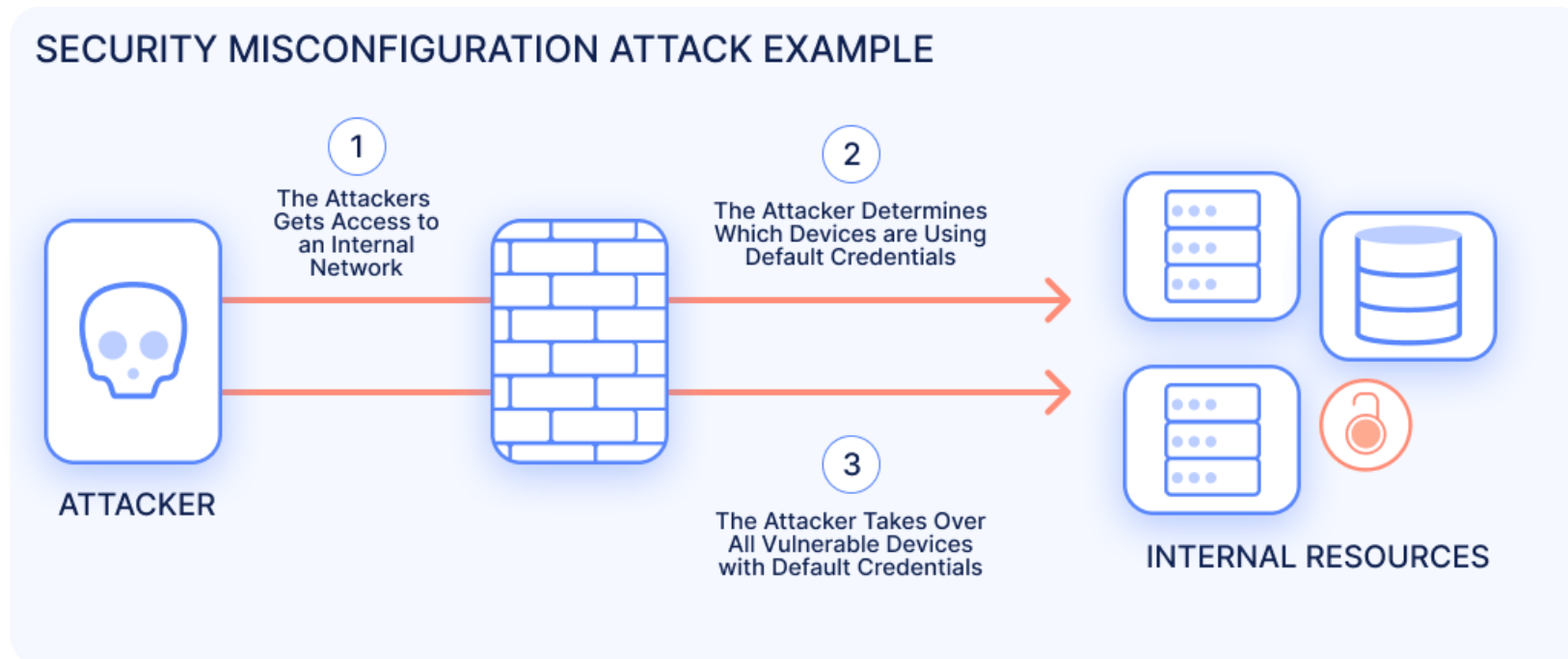


การออกแบบควรคำนึงถึงสิทธิ์ของผู้ใช้งานในแต่ละสิทธิ์ ให้ตรงตามการทำงานของแอปพลิเคชันนั้นๆ

A2 - Security Misconfiguration

(การตั้งค่าความปลอดภัยที่ผิดพลาด)

ช่องโหว่ที่เกิดจากการกำหนดค่าระบบหรือแอปพลิเคชันอย่างไม่ปลอดภัย
เช่น การใช้ค่าดั้งเดิมที่ไม่ปลอดภัย หรือการเปิดใช้งานฟังก์ชันที่ไม่จำเป็น



ตัวอย่างการโจมตี

Scenario #1:

เซิร์ฟเวอร์มาพร้อมกับแอปพลิเคชันที่ตั้งขึ้นโดยไม่ได้มีการลบแอปพลิเคชันที่ไม่จำเป็นออกจากเซิร์ฟเวอร์ แอปพลิเคชันที่ติดมาด้วยนั้น อาจช่องโหว่ด้านความปลอดภัยที่มักถูกผู้โจมตีใช้เพื่อโจมตีเซิร์ฟเวอร์ เช่น บัญชีเริ่มต้นไม่ได้ถูกเปลี่ยนแปลง ผู้โจมตีจะเข้าสู่ระบบด้วยรหัสผ่านเริ่มต้นและยึดครองความควบคุมในระบบไป

Scenario #2:

Directory Listing ถูกเปิดใช้งานบนเซิร์ฟเวอร์ ผู้โจมตีพบว่าสามารถเรียกดูรายการ Directory ได้ ทำให้ผู้โจมตีสามารถทำการดูไฟล์ต่างๆที่อยู่บนเซิร์ฟเวอร์ได้ ซึ่งอาจพบไฟล์ Source code หรือ Config ต่างๆ จากนั้นผู้โจมตีใช้ช่องโหว่ที่พบเพื่อในการโจมตีระบบต่อไป



Index of /admin/backup

www.vulnweb.com/admin/backup/ Incognito

Index of /admin/backup

Name	Last modified	Size	Description
 Parent Directory		-	
 FTP_ls.log	2020-04-27 09:20	63K	
 database_connect.php	2020-04-27 09:20	300	
 db_dump.sql	2020-04-27 09:21	96K	
 old_pass.txt	2020-04-27 09:22	6.3K	

Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.4.5 Server at 127.0.0.1 Port 80

วิธีการแก้ไข และป้องกัน



สร้างมาตรฐานการตั้งค่าระบบที่ปลอดภัย และนำมาใช้กับทุกสภาพแวดล้อม (Dev, Test(Uat), Prod) ให้เหมือนกัน



ลบซอฟต์แวร์, ตัวอย่างแอปพลิเคชัน, หน้าเว็บทดสอบ, และบัญชีผู้ใช้ที่ไม่จำเป็นออกให้หมด



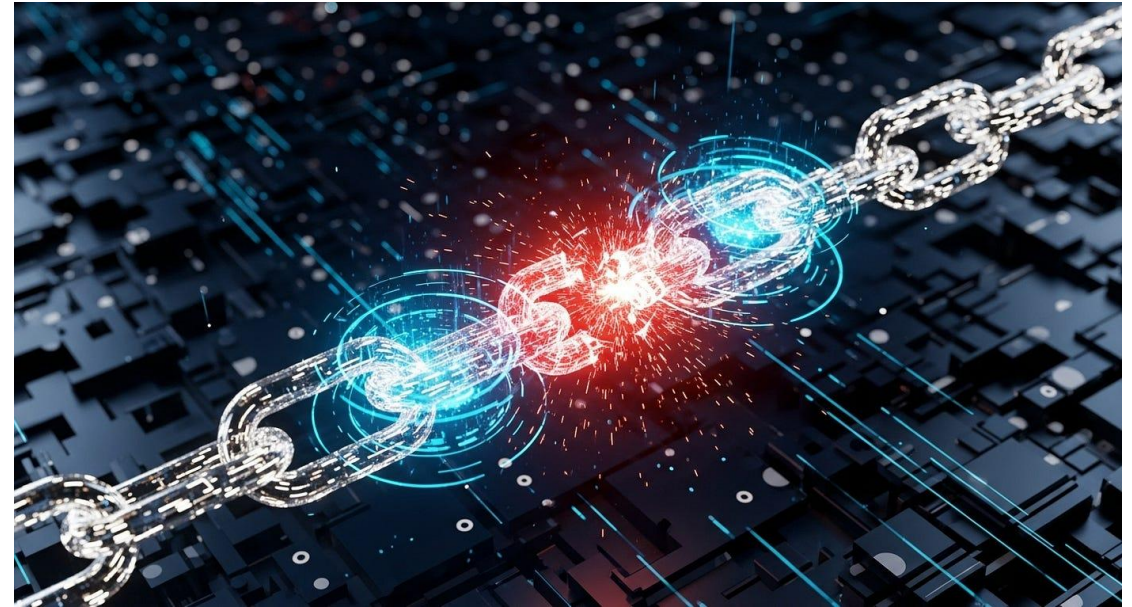
ใช้เครื่องมือสแกนช่องโหว่อัตโนมัติเพื่อตรวจสอบการตั้งค่าที่ผิดพลาดเป็นระยะ

03

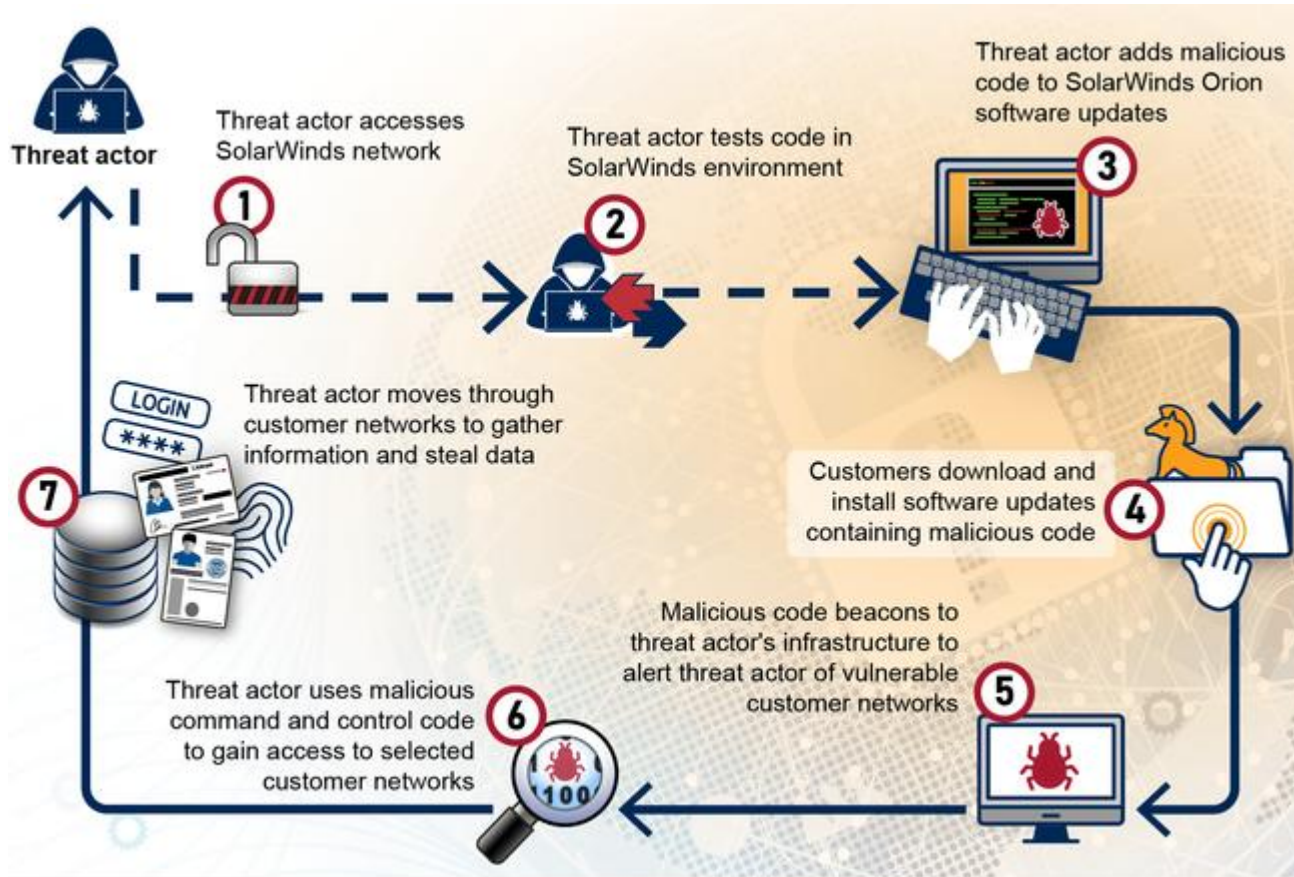
A3 - Software Supply Chain Failures

(ความล้มเหลวของซอฟต์แวร์นำมาใช้)

เป็นการโจมตีที่ไม่ได้มุ่งเป้าไปที่ซอฟต์แวร์ตัวหลักโดยตรง แต่โจมตีจาก "แหล่งที่มา" หรือ "กระบวนการสร้าง" แทน ทำให้ซอฟต์แวร์ที่ผู้ใช้ได้รับไปนั้นไม่ปลอดภัยตั้งแต่แรก เช่น การใช้ฟังก์ชันที่ล้าสมัย, การใช้ซอฟต์แวร์ third-party ที่มีช่องโหว่



ตัวอย่างการโจมตี



Vender ที่เชื่อถือถูกโจมตีด้วยมัลแวร์ เมื่อใช้ซอฟต์แวร์ของ Vender ทำให้เครื่องของคุณถูกโจมตีไปด้วย

ตัวอย่าง

- SolarWinds ในปี 2019
 - ผู้โจมตีได้เจาะเข้าสู่ระบบภายในของ SolarWinds และฝังโค้ดมัลแวร์ เข้าไปในไฟล์ไบนารีของซอฟต์แวร์
 - เมื่อ SolarWinds ปลอ่ยอัปเดตซอฟต์แวร์ ที่ถูกฝังโค้ด เมื่อลูกค้าทำการอัปเดตซอฟต์แวร์ที่ถูกฝังโค้ดมัลแวร์ ทำให้เครื่องของลูกค้าจะถูกติดตั้งมัลแวร์โดยไม่รู้ตัวSolarWinds ประมาณ 18,000 องค์กรได้รับผลกระทบ

ตัวอย่างการโจมตี

- Bybit (เดือนกุมภาพันธ์ 2025)
 - ผู้โจมตีได้มุ่งเป้าโจมตีไปที่ Safe{Wallet} ซึ่งเป็นโซลูชัน Third-party ที่ Bybit ใช้
 - โดยผู้โจมตีสามารถ bypass การตรวจสอบทำให้สามารถทำการโอนย้ายเงินจากระบบออกไปได้
 - เหตุการณ์นี้ทำให้สูญเสียเงินไปประมาณ 1.5 ล้านดอลลาร์



วิธีการแก้ไข และป้องกัน



ลบ dependencies ที่ไม่ได้ใช้, องค์กรประกอบ, ไฟล์, และเอกสารที่ไม่จำเป็นออก



ทำการตรวจสอบเวอร์ชันที่ใช้ ทั้งฝั่งไคลเอนต์และเซิร์ฟเวอร์ (เช่น เฟรมเวิร์ก, ไลบรารี) โดยใช้เครื่องมือ เช่น versions, OWASP Dependency Check, retire.js ฯลฯ



เฝ้าระวังแหล่งข้อมูลอย่างต่อเนื่อง เช่น Common Vulnerability and Exposures (CVE) และ National Vulnerability Database (NVD) สำหรับช่องโหว่ในส่วนประกอบที่คุณใช้ ใช้เครื่องมือวิเคราะห์ส่วนประกอบซอฟต์แวร์ (Software Composition Analysis)

04

A4 - Cryptographic Failures (ความผิดพลาดของการตั้งค่าการเข้ารหัส)

เป็นช่องโหว่ที่เกิดจากการใช้การเข้ารหัสที่ไม่เหมาะสม เช่น การใช้การเข้ารหัสที่ไม่ปลอดภัยหรือการจัดการกุญแจเข้ารหัส(Encryption key) ที่ไม่ดี



ตัวอย่างการโจมตี

Scenario #1:

แอปพลิเคชันไม่ได้บังคับใช้ HTTPS ทำให้เมื่อผู้ใช้ล็อกอินผ่าน Wi-Fi สาธารณะ แฮ็กเกอร์สามารถดักจับ Cookie และสวมรอยเป็นผู้ใช้คนนั้นได้ทันที



Scenario #2:

ฐานข้อมูลเก็บเลขบัตรเครดิตโดยใช้การเข้ารหัสแบบ AES-128 แต่ตัว Encryption Key กลับถูกเก็บไว้ในไฟล์ config.js บนเซิร์ฟเวอร์เดียวกัน หากแฮ็กเกอร์เข้าถึงไฟล์ได้ ข้อมูลบัตรทั้งหมดก็จะเป็นความลับอีกต่อไป

วิธีการแก้ไข และป้องกัน



ระบุว่าข้อมูลไหน "อ่อนไหว" (เช่น เลขบัตรประชาชน, ข้อมูลสุขภาพ) เพื่อรับรองว่าข้อมูลเหล่านั้นจะได้รับการป้องกันสูงสุด



บังคับใช้ HSTS (HTTP Strict Transport Security) เพื่อให้มั่นใจว่าการเชื่อมต่อทั้งหมดถูกเข้ารหัสเสมอ



จัดเก็บรหัสผ่านโดยใช้ฟังก์ชันแฮชที่แข็งแกร่งและมีการเพิ่มค่าซอลต์ พร้อมด้วย delay factor เช่น Argon2, scrypt, bcrypt หรือ PBKDF2



ปิดการใช้งาน TLS 1.0/1.1 และหันมาใช้ TLS 1.2 หรือ 1.3 เท่านั้น



อย่าใช้โปรโตคอลเก่า เช่น FTP และ SMTP สำหรับการขนส่งข้อมูลที่มีความสำคัญ

05

A5 – Injection (การแทรกคำสั่ง)

ช่องโหว่ที่เกิดจากการนำข้อมูลที่ไม่น่าเชื่อถือมาใช้ในคำสั่งเช่น SQL Injection, Command Injection ทำให้เซิร์ฟเวอร์มีการประมวลผลคำสั่งอันตรายที่ไม่ได้เป็นจุดมุ่งหมายของแอปพลิเคชัน



ตัวอย่างการโจมตี

Login

Username

Password

Log in

แอปพลิเคชันใช้ข้อมูลที่ไม่น่าเชื่อถือในการสร้างคำสั่ง SQL ที่มีช่องโหว่ดังต่อไปนี้:

```
SELECT name FROM user WHERE name='admin' AND password='xDK9&GoP1'
```

Login

Username

Password

Log in

```
SELECT name FROM user WHERE name='administrator' or 1=1--' AND passwd='xDK9&GoP1'
```

ตัวอย่างการโจมตี

My Account

Your username is: administrator

Email

Update email

ตัวอย่างการโจมตี

Scenario #2:

การโจมตีแบบ SQL Injection โดยการปรับเปลี่ยนค่า id ในเบราว์เซอร์เพื่อส่งค่า ' UNION SLEEP(10);-- นั้นสามารถทำให้เกิดการโจมตีและส่งผลกระทบต่อฐานข้อมูลและแอปพลิเคชันได้อย่างมาก ดังตัวอย่าง:



```
http://example.com/app/accountView?id=' UNION SELECT  
SLEEP(10);--
```

การโจมตีแบบ SQL Injection โดยการปรับเปลี่ยนค่า id สามารถเปลี่ยนความหมายของคำสั่ง SQL ให้ดึงข้อมูลทั้งหมดจากตารางหรือแม้กระทั่งแก้ไขหรือลบข้อมูลได้ นอกจากนี้ยังสามารถเรียกใช้ Stored Procedures ที่เป็นอันตรายได้

วิธีการแก้ไข และป้องกัน



วิธีที่แนะนำในการป้องกัน SQL Injection คือการใช้ API แทนการใช้ Parameter โดยตรง จัดหาการเชื่อมต่อแบบพารามิเตอร์ เพื่อให้มั่นใจว่าการจัดการข้อมูลเป็นไปอย่างปลอดภัยและมีประสิทธิภาพ



ใช้การตรวจสอบข้อมูล input ฝั่งเซิร์ฟเวอร์ แทนที่การตรวจสอบที่ client (web-browser)



สำหรับคำสั่งแบบไดนามิกที่เหลือยู่ ให้ทำการสร้าง whitelist อักขระพิเศษโดยใช้การสร้างชุด whitelist ที่เฉพาะเจาะจงสำหรับตัวแปรนั้นๆ



ใช้ LIMIT และการควบคุม SQL อื่น ๆ ภายในคำสั่งเพื่อป้องกันการเปิดเผยข้อมูลจำนวนมาก ในกรณีที่เกิด SQL Injection

06

A6 - Insecure Design (การออกแบบที่ไม่ปลอดภัย)

ช่องโหว่ที่เกิดจากการออกแบบแอปพลิเคชันที่ไม่คำนึงถึงความปลอดภัย เช่น ขาดการพิจารณาด้านความปลอดภัยตั้งแต่ขั้นตอนการออกแบบ ไม่ได้มีการป้องกันรูปแบบการโจมตีต่างๆ ที่อาจจะเกิดขึ้นได้กับตัวแอปพลิเคชัน



ตัวอย่างการโจมตี



Scenario #1:

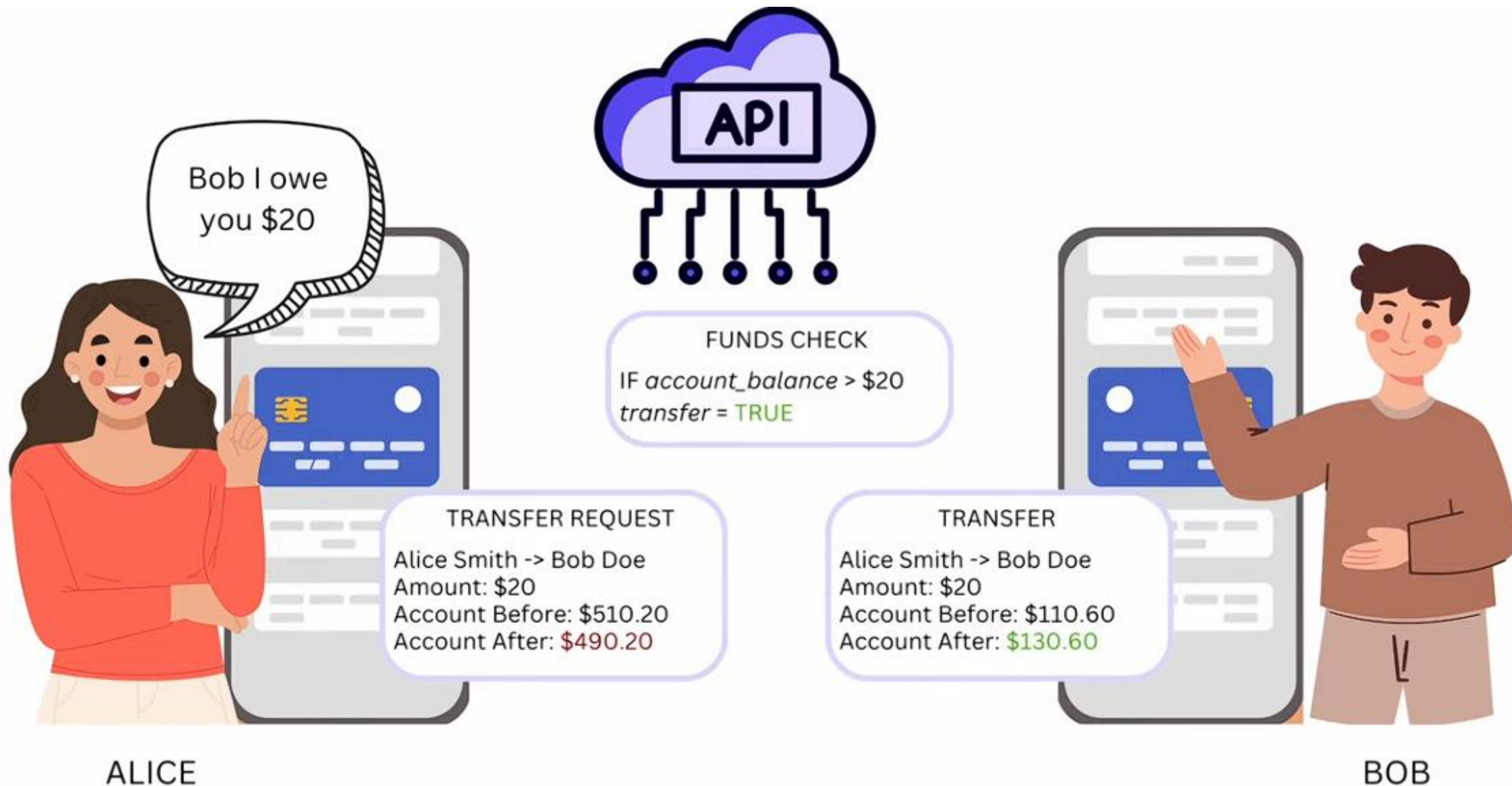
การใช้ “questions and answers” ในการยืนยันตัวตน เป็นการยืนยันตัวตนที่ไม่ปลอดภัย และไม่สามารถยืนยันความเป็นเจ้าของของบัญชีนั้นได้จริง

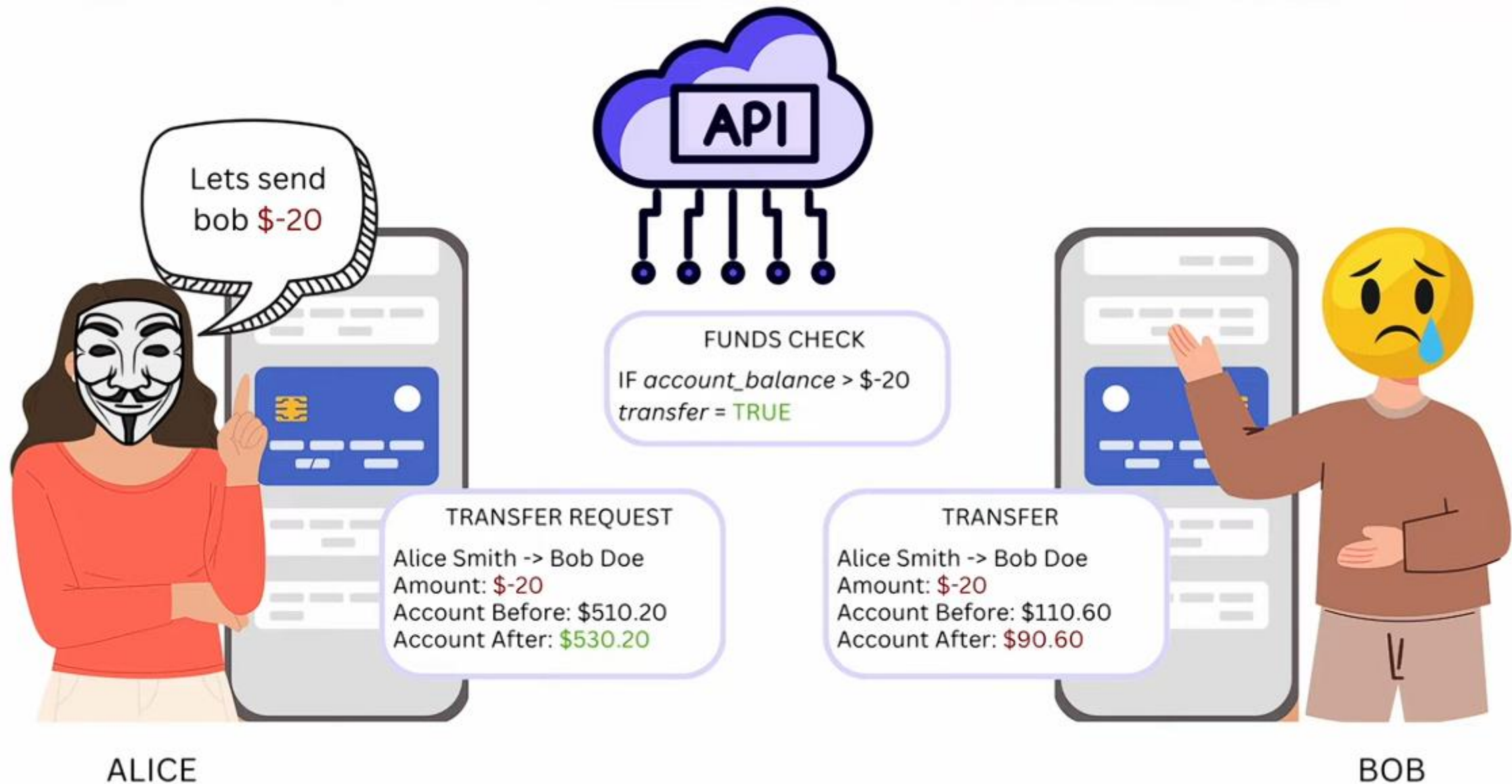
Scenario #2:

โรงพยาบาลโทรไปรื้อชิ้นมีส่วนลดสำหรับการจองแบบกลุ่มและกำหนดให้มีผู้เข้าชมสูงสุดสิบห้าคนโดยไม่ต้องมัดจำ ผู้โจมตีสามารถหาช่องโหว่จากออกแบบที่ผิดพลาดและพบว่าสามารถจองที่นั่งหกร้อยที่นั่งและทุกโรงพยาบาลในครั้งเดียวด้วยการร้องขอเพียงไม่กี่ครั้ง ซึ่งทำให้เกิดการสูญเสียรายได้อย่างมหาศาล

Scenario #3:

แอปธนาคารยอมให้ผู้ใช้ใส่จำนวนเงินที่มีค่า ตีตลาด เข้าไปได้ทำให้จากจะโอนเงินไปหาเป้าหมาย กลายเป็นไปดึงเงินจากเป้าหมายมาแทน





วิธีการแก้ไข และป้องกัน



การเชื่อมต่อด้านความปลอดภัยและการควบคุมความปลอดภัยเข้ากับ User Stories เป็นสิ่งสำคัญในการพัฒนาซอฟต์แวร์ที่ปลอดภัย



การตรวจสอบความน่าเชื่อถือในแต่ละชั้นของแอปพลิเคชัน (จาก frontend ถึง backend)



จำกัดการใช้งาน Resource โดยผู้ใช้งานหรือเซิร์ฟเวอร์ของแอปพลิเคชัน



ดำเนินการจัดตั้งการทำ secure development lifecycle กับผู้เชี่ยวชาญด้าน ความปลอดภัยด้านแอปพลิเคชันในการประเมินความปลอดภัยในการออกแบบแอปพลิเคชัน

07

A7 - Authentication Failures

(การระบุตัวตนและการตรวจสอบสิทธิ์ที่ล้มเหลว)

ช่องโหว่ที่เกิดจากการจัดการการระบุตัวตนและการตรวจสอบสิทธิ์ที่ไม่ปลอดภัย เช่น การใช้รหัสผ่านทำซ้ำบ่อยครั้ง หรือการจัดการเซสชันที่ไม่ปลอดภัย

ตัวอย่างการโจมตี



Scenario #1:

การใช้ password ที่คาดเดาง่าย ไม่มีการเปลี่ยน password เป็นประจำ หรือไม่มี policy ในการตั้ง password มักจะถูกโจมตีได้ง่าย เนื่องจากสามารถคาดเดา password ได้ง่าย

Scenario #2:

การตั้งค่าเวลาหมดอายุของเซสชันในแอปพลิเคชันไม่ถูกต้อง ผู้ที่ใช้คอมพิวเตอร์สาธารณะเพื่อเข้าถึงแอปพลิเคชัน แทนที่จะเลือก "ออกจากระบบ" ผู้ใช้จะปิดแท็บเบราว์เซอร์และเดินออกไป หากมีผู้ไม่ประสงค์ดีเข้ามาใช้คอมพิวเตอร์สาธารณะเครื่องนั้นต่อ สามารถใช้แอปพลิเคชันที่เซสชันยังไม่หมดอายุของผู้ใช้งานได้

วิธีการแก้ไข และป้องกัน



ควรใช้การตรวจสอบบัญชีแบบหลายปัจจัย (multi-factor authentication - MFA) เพื่อป้องกันการโจมตีประเภท **Brute force** และการนำข้อมูลเข้ารหัสที่ถูกขโมยไปมาใช้ในการเข้าระบบ



ห้ามแชร์ **Password** กับผู้อื่น โดยเฉพาะสำหรับผู้ดูแลระบบ



จำกัดหรือเพิ่มระยะเวลาในการพยายามเข้าสู่ระบบ ในกรณีที่เข้าระบบล้มเหลว

08

A8 - Software and Data Integrity Failures

(การล้มเหลวของความสมบูรณ์ของซอฟต์แวร์และข้อมูล)

ช่องโหว่ที่เกิดจากการขาดการตรวจสอบความสมบูรณ์ของซอฟต์แวร์และข้อมูล เช่น การอัปเดตซอฟต์แวร์จากแหล่งที่ไม่เชื่อถือ ทำให้ในโค้ดที่อัปเดตนั้น มีการฝังโค้ดอันตรายที่ทำให้ผู้ไม่ประสงค์ดีสามารถก่อความเสียหายต่อระบบได้

ตัวอย่างการโจมตี



Scenario #1:

อุปกรณ์อย่างเราเตอร์หรือกล่องทีวี มักขาดระบบตรวจสอบความปลอดภัยขณะอัปเดต (Unsigned Firmware) ทำให้ผู้บุกรุกสามารถส่งไฟล์ปลอมเข้าไปควบคุมเครื่องได้ง่าย ซึ่งปัญหานี้แก้ไขยาก เพราะต้องรออัปเดตใหญ่ในอนาคตหรือรอนกว่าคนจะเลิกใช้เครื่องรุ่นนั้นไปเอง

Scenario #2:

ผู้โจมตีเข้าถึงระบบ Build System (เช่น Jenkins หรือ GitHub Actions) และแอบแก้ไขสคริปต์ที่ใช้ในการ Compile ซอฟต์แวร์ แม้โค้ดต้นฉบับจะปลอดภัย แต่ซอฟต์แวร์ตัวสุดท้าย (Artifact) ที่ถูกส่งออกไปหาลูกค้ากลับมี Backdoor ฝังอยู่ เพราะไม่มีการตรวจสอบความถูกต้องของไฟล์ในทุกขั้นตอนของ Pipeline

วิธีการแก้ไข และป้องกัน



ใช้ digital signatures ในการยืนยันว่า ซอฟต์แวร์หรือข้อมูลมาจากแหล่งที่ถูกต้อง และไม่ได้ถูกแก้ไข



ตรวจสอบให้แน่ใจว่า libraries และ dependencies เช่น npm หรือ Maven ที่ใช้งานมาจากแหล่งที่เชื่อถือได้



ตรวจสอบให้แน่ใจว่ามีกระบวนการตรวจสอบ (Review Process) สำหรับการเปลี่ยนแปลงโค้ดและการตั้งค่า เพื่อลดโอกาสที่โค้ดหรือการตั้งค่าที่เป็นอันตรายจะถูกนำเข้าสู่ซอฟต์แวร์ของคุณ

A09 - Security Logging & Alerting Failures

(การบันทึกและการตรวจสอบความปลอดภัยที่ล้มเหลว)

ช่องโหว่ที่เกิดจากการขาดการบันทึกและการตรวจสอบกิจกรรมที่สำคัญ เช่น
การไม่มีการบันทึกเหตุการณ์ที่สำคัญ (Logging)
หรือการไม่ตรวจสอบการแจ้งเตือนความปลอดภัย (Alert)

09

ตัวอย่างการโจมตี



Scenario #1:

ระบบบันทึกเฉพาะการเข้าสู่ระบบที่สำเร็จ แต่ไม่บันทึก การเข้าสู่ระบบที่ล้มเหลว (Failed Logins ทำให้ผู้โจมตีสามารถใช้วิธี Brute Force หรือ Credential Stuffing ลองรหัสผ่านไปเรื่อยๆ ได้หลายล้านครั้งโดยที่ทีม Security ไม่เห็นความผิดปกติใน Log เลย

Scenario #2:

สิทธิ์ในการเข้าถึงไฟล์ Log ไม่ได้รับการป้องกันที่ดีพอ เมื่อแฮกเกอร์เจาะระบบได้สำเร็จ สิ่งแรกที่เขาจะทำคือ "ลบประวัติ" เพื่อปกปิดร่องรอย หากเราไม่มีการทำ Log Integrity หรือส่ง Log ไปยัง Storage ที่เขียนได้อย่างเดียว เรา将无法ทำ Digital Forensics เพื่อหาต้นตอได้เลย

วิธีการแก้ไข และป้องกัน



ตรวจสอบให้แน่ใจว่าการเข้าสู่ระบบ การเข้าถึงข้อมูล และรวมถึง Error ที่เกิดขึ้น สามารถตรวจสอบข้อมูลอินพุตฝั่งเซิร์ฟเวอร์ทั้งหมดสามารถบันทึกได้พร้อมกับบริบทของผู้ใช้ที่เพียงพอเพื่อระบุบัญชีที่น่าสงสัยหรือเป็นอันตราย และเก็บรักษาข้อมูลนี้ไว้นานพอเพื่อให้สามารถวิเคราะห์ทางนิติเวชในภายหลังได้



ตรวจสอบให้แน่ใจว่าการบันทึกข้อมูลถูกสร้างในรูปแบบที่โซลูชันการจัดการ Log สามารถนำไปใช้ได้อย่างง่ายสำหรับอุปกรณ์ที่ใช้ในการวิเคราะห์ข้อมูล



ตรวจสอบให้แน่ใจว่าข้อมูล Log ได้มีการเข้ารหัสอย่างถูกต้อง เพื่อป้องกันไม่ให้เกิดการโจมตีระบบเก็บข้อมูลหรือระบบที่ใช้ในการวิเคราะห์ข้อมูล

10

A10 - Mishandling of Exceptional Conditions (การรับมือความผิดปกติที่ผิดพลาด)

ข้อบกพร่องในการป้องกัน, ตรวจจับ, และตอบสนองต่อสถานการณ์ที่ไม่ปกติหรือไม่คาดคิดในระบบ ซึ่งอาจนำไปสู่ช่องโหว่ด้านความปลอดภัย, ระบบล่ม, หรือพฤติกรรมที่ไม่พึงประสงค์



ตัวอย่างการโจมตี



Scenario #1:

เมื่อระบบเกิด Error (เช่น Database เชื่อมต่อไม่ได้ หรือ Query ผิดพลาด) หน้าจอแสดงผลกลับ Stack Trace, ชื่อ Table, เวอร์ชันของซอฟต์แวร์ หรือแม้แต่ File Path ในเซิร์ฟเวอร์ออกมาให้ผู้ใช้งานเห็น ผู้โจมตีสามารถใช้ข้อมูลดังกล่าวในการโจมตีได้

Scenario #2:

ระบบมีการเช็คสิทธิ์ (Authorization) ก่อนเข้าถึงข้อมูล แต่เกิด Exception ในระหว่างกระบวนการเช็คสิทธิ์นั้นพอดี หากโค้ดเขียนไว้ไม่รัดกุม ระบบอาจจะข้ามขั้นตอนการเช็คไปและอนุญาตให้ผู้ใช้งานเข้าถึงข้อมูลได้เลย

Scenario #3:

แอปพลิเคชันมีการตั้งให้ดักการอัปโหลดไฟล์ แต่ไม่ปล่อยไฟล์ที่ถูกดักไว้มีมากเกินไปทำให้ทรัพยากรถูกใช้ไปจนหมด ทำให้เกิด Denial of Service - DoS

วิธีการแก้ไข และป้องกัน



ใช้ Rate Limiting, Resource Quotas เพื่อป้องกันสถานะผิดปกติ ตั้งแต่ต้นทาง



ตรวจสอบอินพุตอย่างเข้มงวด (Strict Input Validation) รวมถึงการกรองหรือการหลีกเลี่ยงอักขระที่เป็นอันตราย



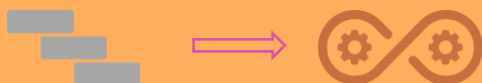
แสดงข้อความ Error แบบทั่วไปให้ผู้ใช้เห็น เช่น "เกิดข้อผิดพลาด กรุณาลองใหม่ ภายหลัง" ส่วนรายละเอียดเชิงเทคนิคให้เก็บไว้ใน Log ภายในเท่านั้น



ตรวจสอบให้แน่ใจว่าเมื่อเกิด Exception ระบบได้ทำการปิด Connection หรือคืน Memory อย่างถูกต้อง เพื่อป้องกันการโจมตีแบบ Denial of Service (DoS) จากการที่ทรัพยากรถูกใช้จนหมด

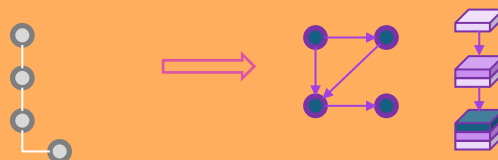
รูปแบบการพัฒนาแอปพลิเคชันได้เปลี่ยนไปแล้ว

จำนวนแอปพลิเคชันมากขึ้น จำนวนทีมผู้พัฒนา
มากขึ้น ต้องการแอปพลิเคชันเร็วขึ้น



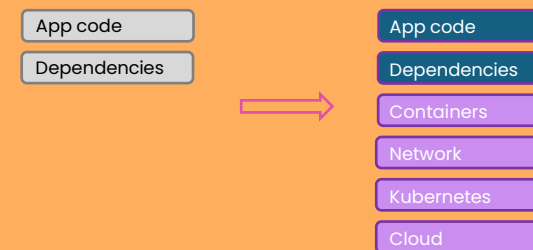
ต้องการผู้พัฒนาที่ความความเข้าใจในด้าน
ความปลอดภัยในการพัฒนาแอปพลิเคชัน

องค์ประกอบของ Software ที่นำมาใช้ในการ
พัฒนามีความซับซ้อนมากขึ้น



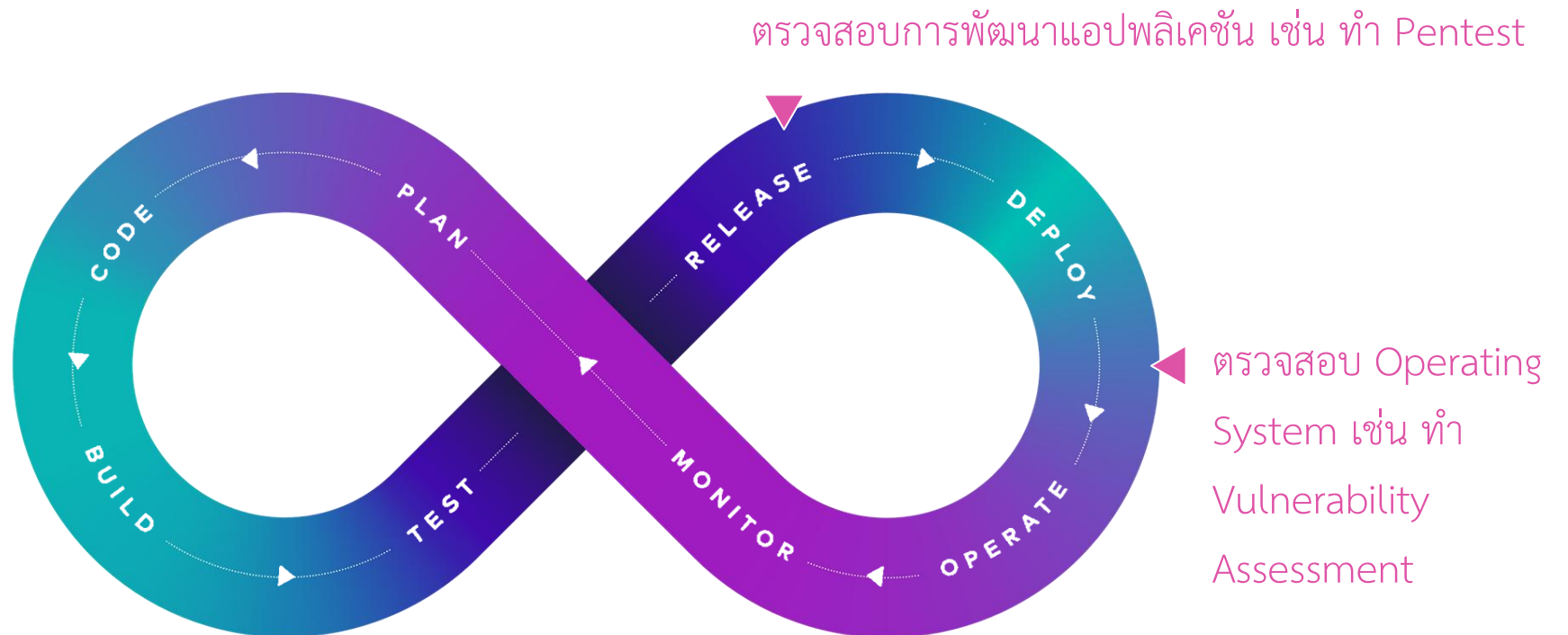
ต้องการการตรวจสอบองค์ประกอบว่า
องค์ประกอบที่นำมาใช้ นำมาใช้ถูกต้อง
หรือมีช่องโหว่หรือไม่

คลาวด์เป็นส่วนหนึ่งการพัฒนาแอปพลิเคชัน



ต้องการเข้าใจในส่วนของการพัฒนาแอป
พลิเคชันและการตั้งค่าของคลาวด์

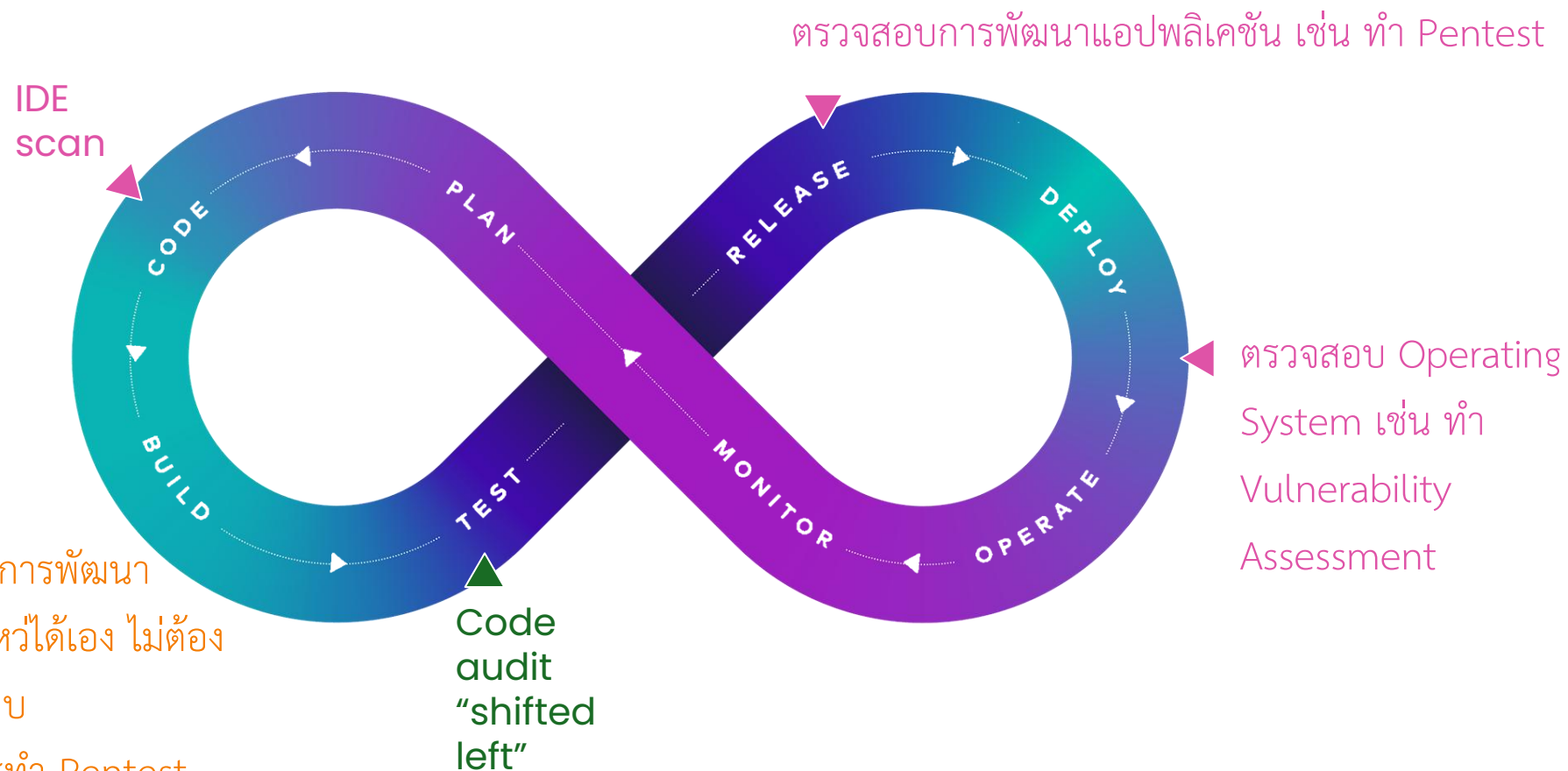
แนวทางที่ควรดำเนินการสำหรับการพัฒนาแอปพลิเคชันอย่างปลอดภัย



ข้อเสีย

- การตรวจสอบทำซ้ำเกินไป
- จำเป็นต้องมีผู้เชี่ยวชาญเข้ามาเกี่ยวข้อง
- หากพบปัญหา อาจจะทำให้การส่งงานล่าช้า

แนวทางที่ควรดำเนินการสำหรับการพัฒนาแอปพลิเคชันอย่างปลอดภัย



ข้อดี

- พบช่องโหว่ในขณะที่ยังดำเนินการพัฒนา
- ผู้พัฒนาสามารถแก้ไขช่องโหว่ได้เอง ไม่ต้องรอให้ผู้เชี่ยวชาญมาตรวจสอบ
- ช่องโหว่ที่ตรวจสอบจากการทำ Pentest, VA ลดลง ทำให้แผนการส่งมอบมีโอกาสเลื่อนได้น้อย

แนวทางที่ควรดำเนินการสำหรับการพัฒนาแอปพลิเคชันอย่างปลอดภัย

A modern approach to security is required

Traditional App Sec

Testing after development



Audit Based



Code and Infrastructure
Secured Independently



Dev-First

Continuous Testing



Fix Based



Holistic Cloud Native
Application Context



แนวทางที่ควรดำเนินการสำหรับการพัฒนาแอปพลิเคชันอย่างปลอดภัย

https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Web_Application_Penetration_Checklist_v1_1.pdf



Checklist

Category	Ref Number	Name	Objective	Notes
AppDOS	OWASP-AD-001	Application Flooding	Ensure that the application functions correctly when presented with large volumes of requests, transactions and / or network traffic.	Use various fuzzing tools to perform this test (e.g. SPIKE)
	OWASP-AD-002	Application Lockout	Ensure that the application does not allow an attacker to reset or lockout user's accounts.	

Category	Ref Number	Name	Objective	Notes
InputValidation	OWASP-IV-001	Script Injection	Ensure that any part of the application that allows input does not process scripts as part of the input.	Classic case of Cross Site Scripting but includes other scripting as well.
InputValidation. SQL	OWASP-IV-002	SQL Injection	Ensure the application will not process SQL commands from the user.	
InputValidation. OS	OWASP-IV-003	OS Command Injection	Ensure the applications will not process operating system commands from the user.	This typically includes issues such as path traversal, spawning command shells and OS functions.
InputValidation. LDAP	OWASP-IV-004	LDAP Injection	Ensure the application will not process LDAP commands from the user.	
InputValidation. XSS	OWASP-IV-005	Cross Site Scripting	Ensure that the application will not store or reflect malicious script code.	

OWASP ZAP (ZAP Proxy)

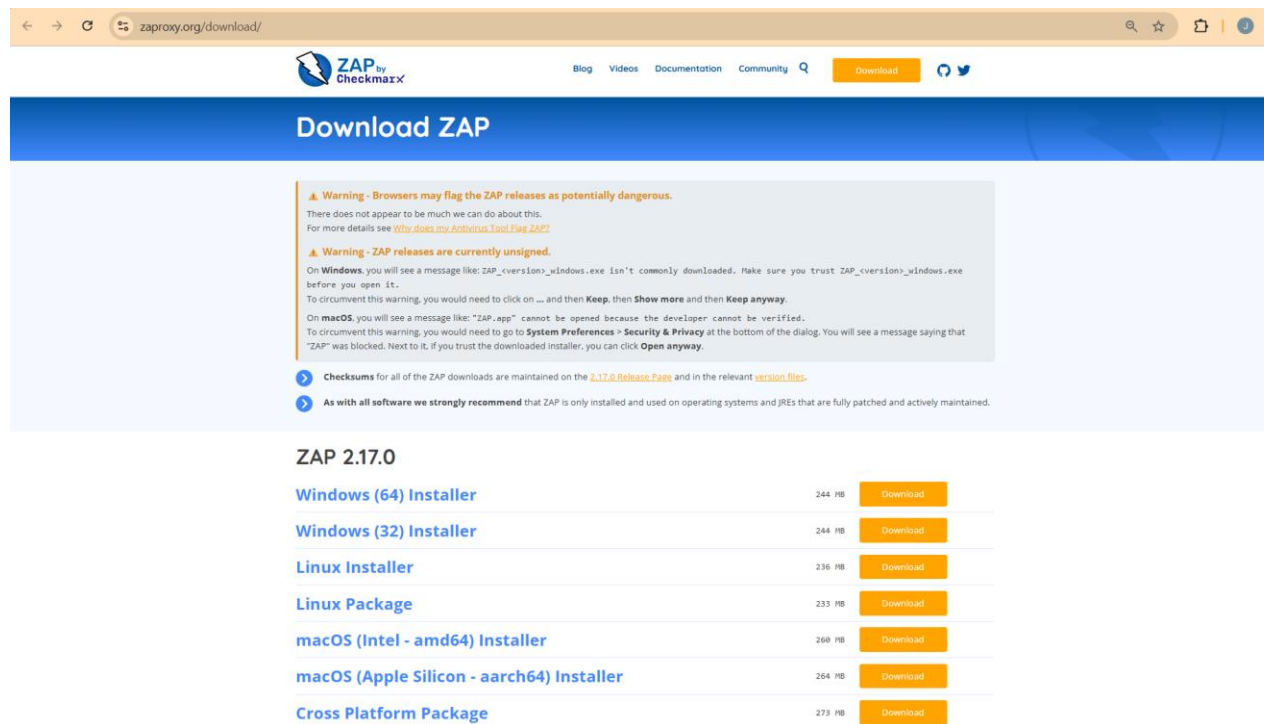
OWASP Zed Attack Proxy (ZAP) เป็นเครื่องมือ Web Proxy อีกหนึ่งตัวที่นิยมใช้กันทั่วไปสำหรับการทดสอบเจาะระบบเว็บ (Web Penetration Testing)

ZAP เป็นโปรเจกต์ Open-source ที่เปิดให้ใช้งานฟรี ริเริ่มโดย Open Web Application Security Project (OWASP) และได้รับการดูแลรักษาโดยชุมชนนักพัฒนา ซึ่งในช่วงไม่กี่ปีที่ผ่านมาเครื่องมือนี้มีการเติบโตอย่างมาก และกำลังได้รับการยอมรับในตลาดอย่างรวดเร็วในฐานะผู้นำด้านเครื่องมือ Web Proxy แบบ Open-source ในปัจจุบัน ถูกบริษัทที่ชื่อว่า Checkmarx ดำเนินการซื้อไป แต่ยังคงปล่อยให้มีการใช้ฟรีอยู่

OWASP ZAP

เราสามารถดาวน์โหลด ZAP ได้จากหน้าดาวน์โหลด (<https://www.zaproxy.org/download/>) โดยเลือกตัวติดตั้ง (Installer) ที่ตรงกับระบบปฏิบัติการที่เราใช้ และปฏิบัติตามคำแนะนำพื้นฐานเพื่อทำการติดตั้งนอกจากนี้

ZAP ยังมีให้ดาวน์โหลดในรูปแบบไฟล์ JAR ที่รองรับการทำงานข้ามแพลตฟอร์ม (Cross-platform) ซึ่งสามารถเปิดใช้งานได้ด้วยคำสั่ง `java -jar` หรือโดยการดับเบิลคลิกที่ไฟล์ดังกล่าว



The screenshot shows the OWASP ZAP download page. At the top, there's a navigation bar with links for Blog, Videos, Documentation, and Community, along with a Download button and social media icons. The main heading is "Download ZAP". Below this, there's a warning section with two alerts: one about browsers flagging ZAP releases as potentially dangerous, and another about ZAP releases being unsigned. The unsigned warning provides instructions for Windows and macOS users to bypass security warnings. Below the warnings, there's a section for checksums and a recommendation to use patched operating systems and JREs. The main content area lists download links for various platforms and architectures, each with a file size and a Download button.

Platform/Architecture	File Size	Action
ZAP 2.17.0		
Windows (64) Installer	244 MB	Download
Windows (32) Installer	244 MB	Download
Linux Installer	236 MB	Download
Linux Package	233 MB	Download
macOS (Intel - amd64) Installer	260 MB	Download
macOS (Apple Silicon - aarch64) Installer	264 MB	Download
Cross Platform Package	273 MB	Download

OWASP ZAP

ในการเริ่มต้นใช้งาน ZAP เราสามารถเปิดโปรแกรมผ่าน Terminal ด้วยคำสั่ง `zapproxy` หรือเรียกใช้จากเมนูแอปพลิเคชันได้
เมื่อ ZAP เริ่มทำงาน จะมีหน้าต่างสอบถามว่าเราต้องการสร้างโปรเจกต์ใหม่หรือใช้เป็นโปรเจกต์ชั่วคราว ในที่นี้ให้เราเลือกใช้โปรเจกต์ชั่วคราวโดยเลือก No เนื่องจากเรา
ไม่ได้กำลังทำงานกับโปรเจกต์ขนาดใหญ่ที่จำเป็นต้องบันทึกข้อมูลเก็บไว้ใช้งานต่อเนื่องหลายวัน

Do you want to persist the ZAP Session?

☐ Yes, I want to persist this session with name based on the current timestamp

☐ Yes, I want to persist this session but I want to specify the name and location

☒ No, I do not want to persist this session at this moment in time

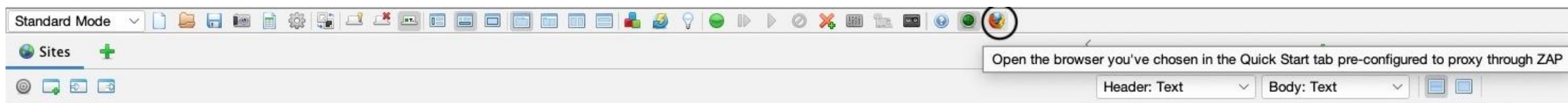
☐ Remember my choice and do not ask me again.

You can always change your decision via the Options / Database screen

OWASP ZAP

การตั้งค่า Proxy

ใน ZAP เราสามารถคลิกที่ไอคอนเบราว์เซอร์ Firefox ที่อยู่ตรงด้านขวาสุดของแถบด้านบน เพื่อเปิดเบราว์เซอร์ที่ถูกตั้งค่าไว้ล่วงหน้าขึ้นมาได้ทันที:

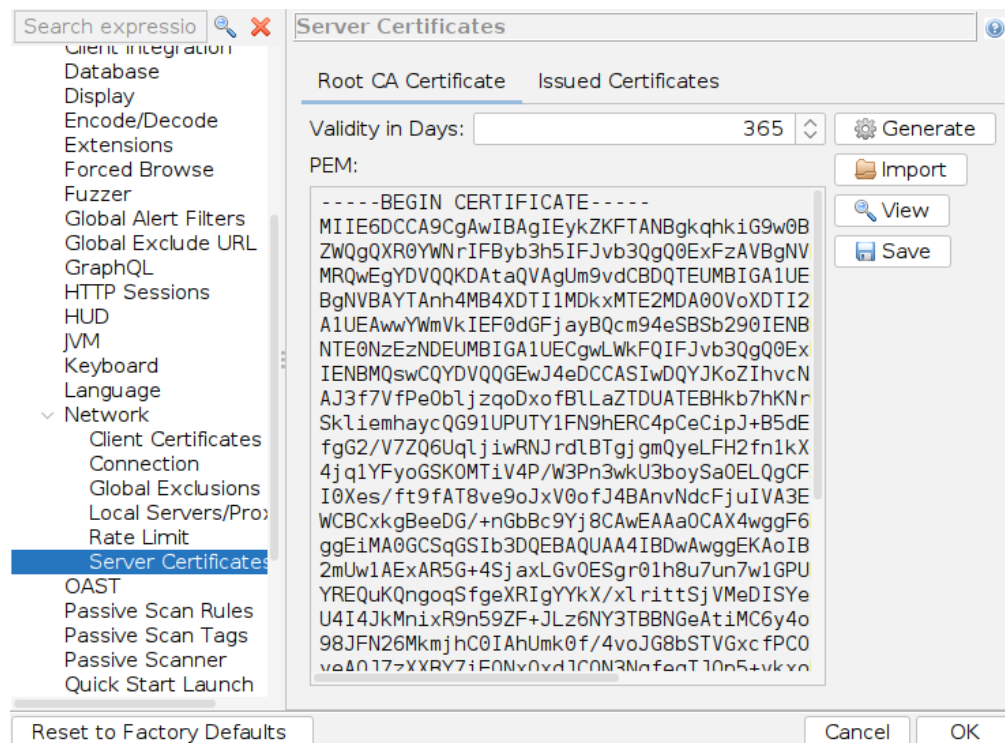


ในหลายๆ กรณี เราอาจต้องการใช้เว็บเบราว์เซอร์จริงๆ เช่น Firefox ในการทำ Pentest ซึ่งการจะใช้งาน Firefox ร่วมกับเครื่องมือ Web Proxy นั้น เราจำเป็นต้องเข้าไปตั้งค่าให้เบราว์เซอร์วิ่งผ่าน Proxy เหล่านั้นเสียก่อน เราสามารถเข้าไปกำหนดค่าเองได้ที่เมนูการตั้งค่า (Preferences) ของ Firefox เพื่อระบุ Proxy ให้ตรงกับ Listening Port ของ Web Proxy นั้นๆ โดยปกติแล้ว ZAP จะใช้พอร์ต 8080 เป็นค่าเริ่มต้น แต่เราสามารถเลือกใช้พอร์ตอื่นๆ ที่ว่างอยู่ได้เช่นกัน แต่หากเราเลือกพอร์ตที่กำลังถูกใช้งานอยู่ Proxy จะไม่สามารถเริ่มทำงานได้และจะมีข้อความแจ้งเตือนข้อผิดพลาด (Error message) ปรากฏขึ้นหมายเหตุ: หากเราต้องการเปลี่ยนพอร์ตของ Web Proxy ในโปรแกรม ZAP ไปเป็นพอร์ตอื่น เราสามารถเข้าไปตั้งค่าได้ที่เมนู (Tools > Options > Network > Local Servers/Proxies) ซึ่งไม่ว่าจะใช้พอร์ตไหน เราต้องตรวจสอบให้แน่ใจว่าการตั้งค่า Proxy ใน Firefox นั้นระบุหมายเลขพอร์ตตรงกันกับที่ตั้งไว้

OWASP ZAP

การติดตั้งใบรับรองความปลอดภัย (CA Certificate)

ใน ZAP เราสามารถคลิกที่ไอคอนเบราว์เซอร์ Firefox ที่อยู่ตรงด้านขวาสุดของแถบด้านบน เพื่อเปิดเบราว์เซอร์ที่ถูกตั้งค่าไว้ล่วงหน้าขึ้นมาใช้งานได้เลย:



OWASP ZAP

ZAP Scanner

ZAP ยังมาพร้อมกับ Web Scanner ในตัว โดย ZAP Scanner นั้นมีความสามารถในการสร้างแผนผังเว็บไซต์ (Site Maps) ด้วยการใช้ ZAP Spider และสามารถทำการสแกนได้ทั้งแบบ Passive และ Active เพื่อค้นหาช่องโหว่ประเภทต่างๆ

OWASP ZAP

Spider

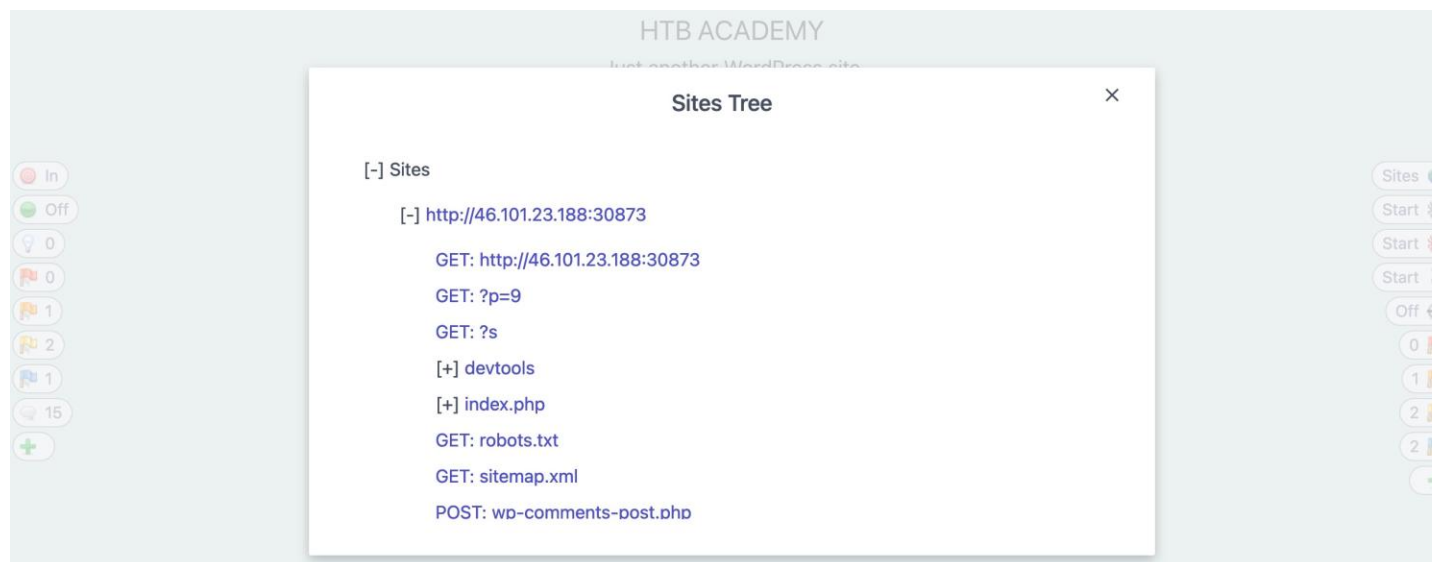
มาเริ่มกันที่ ZAP การจะเริ่มใช้งาน Spider สแกนเว็บไซต์นั้น เราสามารถทำได้โดยไปที่แท็บ History เลือก Request ที่ต้องการ แล้วคลิกขวาเลือกเมนู (Attack > Spider) อีกวิธีหนึ่งคือการใช้งานผ่าน HUD บนเบราว์เซอร์ที่ ZAP ตั้งค่ามาให้ เมื่อเราเข้าไปยังหน้าเว็บหรือเว็บไซต์ที่ต้องการสแกนแล้ว ให้คลิกที่ปุ่มที่สองบนแถบด้านขวา (Spider Start) ระบบจะแสดงหน้าต่างขึ้นมาเพื่อให้เรายืนยันการเริ่มสแกน:



หมายเหตุ: เมื่อเราคลิกที่ปุ่ม Spider โปรแกรม ZAP อาจแจ้งเตือนว่าเว็บไซต์ปัจจุบันไม่ได้อยู่ในขอบเขต (Scope) ของเรา และจะถามว่าเราต้องการเพิ่มเว็บไซต์นี้เข้าไปใน Scope โดยอัตโนมัติก่อนเริ่มสแกนหรือไม่ ซึ่งในกรณีนี้เราสามารถตอบว่า 'Yes' ได้เลย Scope คือชุดของ URL ที่ ZAP จะทำการทดสอบ หากเราเริ่มการสแกนแบบทั่วไป ซึ่งเราสามารถปรับแต่ง Scope ได้เองเพื่อกำหนดให้สแกนหลายเว็บไซต์หรือหลาย URL พร้อมกัน ลองฝึกเพิ่มเป้าหมาย (Targets) หลายๆ แห่งเข้าไปใน Scope ดู เพื่อสังเกตความแตกต่างในการทำงานของการสแกน

OWASP ZAP

ทันทีที่เราคลิกปุ่ม Start บนหน้าต่าง Pop-up ระบบจะเริ่มทำการ Spider เว็บไซต์ โดยจะค้นหาลิงก์ต่างๆ และตรวจสอบความถูกต้อง ซึ่งกระบวนการนี้จะคล้ายคลึงกับการทำงานของ Burp Crawler เราสามารถติดตามความคืบหน้าของการสแกนได้จาก HUD บริเวณปุ่ม Spider หรือผ่านหน้าจอหลัก (ZAP UI) ซึ่งปกติแล้วโปรแกรมจะสลับไปยังแท็บ Spider ให้โดยอัตโนมัติ เพื่อแสดงสถานะความคืบหน้าและ Request ต่างๆ ที่ถูกส่งออกไปเมื่อการสแกนเสร็จสิ้น เราสามารถตรวจสอบผลลัพธ์ได้ที่แท็บ Sites บนหน้าจอหลักของ ZAP หรือคลิกที่ปุ่มแรกบนแถบด้านขวา (Sites Tree) บน HUD ซึ่งจะแสดงรายการเว็บไซต์และไคเรกทอรีย่อยทั้งหมดที่ตรวจพบในรูปแบบโครงสร้างต้นไม้ (Tree-list) ที่สามารถกดขยายดูรายละเอียดได้

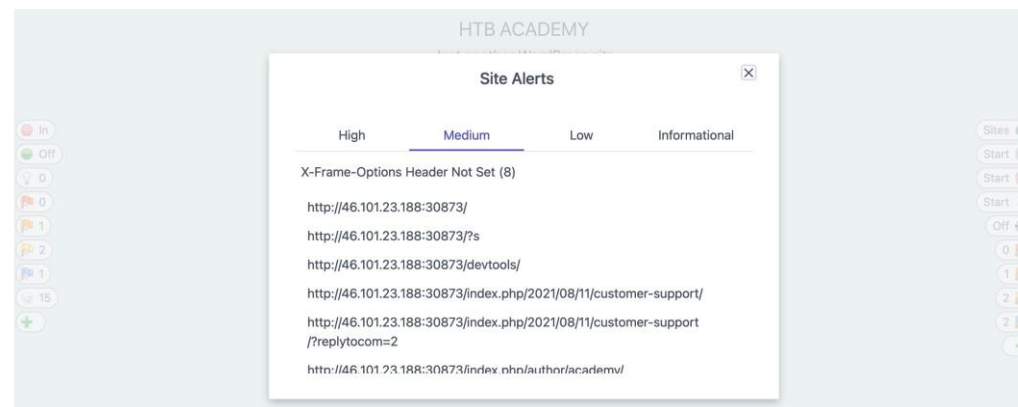


OWASP ZAP



Passive Scanner

ในขณะที่ ZAP Spider กำลังทำงานและส่ง Request ไปยัง Endpoint ต่างๆ ระบบจะทำการรัน Passive Scanner ไปพร้อมกันโดยอัตโนมัติในทุกๆ Response ที่ได้รับ เพื่อตรวจสอบหาปัญหาที่อาจซ่อนอยู่ใน Source Code เช่น การขาด Security Headers หรือช่องโหว่ประเภท DOM-based XSS ด้วยเหตุนี้ แม้ว่าเราจะได้ไม่ได้เริ่มรัน Active Scanner แต่เราอาจเห็นว่าปุ่มแจ้งเตือน (Alerts) เริ่มแสดงรายการปัญหาที่ตรวจพบขึ้นมาบ้างแล้ว โดยการแจ้งเตือนบนแถบด้านซ้ายจะแสดงปัญหาที่พบในหน้าเว็บปัจจุบันที่เรา กำลังเปิดอยู่ ส่วนแถบด้านขวาจะแสดงภาพรวมของการแจ้งเตือนทั้งหมดในเว็บแอปพลิเคชันนั้น ซึ่งรวมถึงปัญหาที่ตรวจพบบนหน้าอื่นๆ ด้วย:



OWASP ZAP



Active Scanner

เมื่อข้อมูลใน Site Tree ของเราแสดงขึ้นมาแล้ว เราสามารถคลิกที่ปุ่ม Active Scan บนแถบด้านขวาเพื่อเริ่มการสแกนแบบ Active บนหน้าเว็บทั้งหมดที่ตรวจพบ หากเรายังไม่ได้ทำการรัน Spider Scan บนเว็บแอปพลิเคชันมาก่อน ZAP จะทำการรัน Spider ให้โดยอัตโนมัติเพื่อสร้าง Site Tree สำหรับใช้เป็นเป้าหมายในการสแกน และเมื่อ Active Scan เริ่มทำงาน เราสามารถติดตามความคืบหน้าได้ในลักษณะเดียวกับตอนที่เราทำ Spider Scan



OWASP ZAP



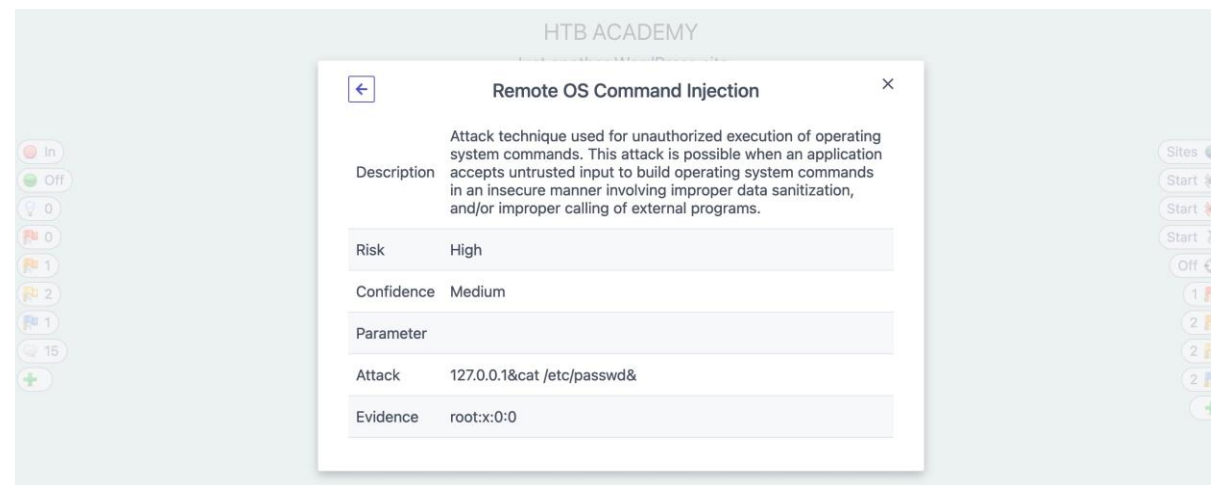
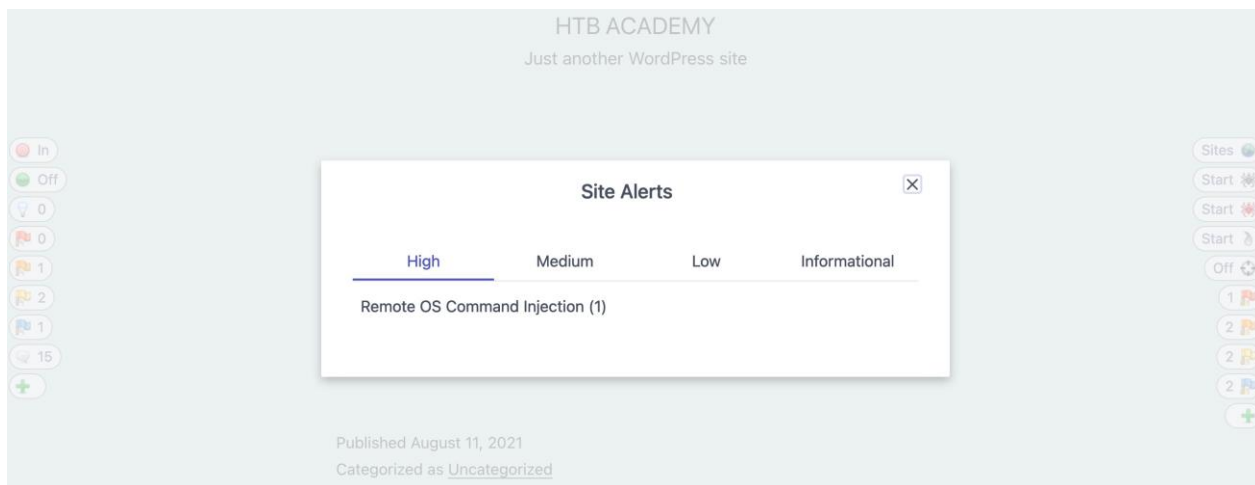
Active Scanner จะพยายามใช้รูปแบบการโจมตีที่หลากหลายกับหน้าเว็บและ HTTP Parameters ทั้งหมดที่ระบุได้ เพื่อค้นหาช่องโหว่ให้ได้มากที่สุดเท่าที่จะทำได้ ด้วยเหตุนี้การทำงานของ Active Scanner จึงใช้เวลานานกว่าจะเสร็จสิ้นในขณะที่ Active Scan กำลังทำงาน เราจะเห็นว่าปุ่มแจ้งเตือน (Alerts) เริ่มมีรายการแจ้งเตือนเพิ่มมากขึ้นเรื่อยๆ ตามจำนวนปัญหาที่ ZAP ตรวจพบ ยิ่งไปกว่านั้น เราสามารถตรวจสอบรายละเอียดเพิ่มเติมเกี่ยวกับการสแกนที่กำลังทำงานอยู่ได้ที่หน้าจอหลัก (ZAP UI) และสามารถดู Request ต่างๆ ที่ ZAP ส่งออกไปได้:

History Search Alerts Output WebSockets Spider Active Scan										
New Scan Progress: 0: http://46.101.23.188:30873 42% Current Scans:1 Num Requests:1756 New Alerts:0 Export										
Sent Messages Filtered Messages										
Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	
1,106			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		123 ms	266 bytes	2,688 bytes	
1,107			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		123 ms	266 bytes	2,688 bytes	
1,108			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		135 ms	266 bytes	2,688 bytes	
1,109			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		124 ms	266 bytes	2,688 bytes	
1,110			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		124 ms	266 bytes	2,688 bytes	
1,111			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		125 ms	266 bytes	2,688 bytes	
1,112			GET	http://46.101.23.188:30873/wp-content	301 Moved Permanently		114 ms	274 bytes	328 bytes	
1,113			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		125 ms	266 bytes	2,688 bytes	
1,114			GET	http://46.101.23.188:30873/wp-content/themes	301 Moved Permanently		113 ms	281 bytes	335 bytes	
1,115			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		126 ms	266 bytes	2,688 bytes	
1,116			GET	http://46.101.23.188:30873/wp-content/themes/twentytw...	301 Moved Permanently		112 ms	297 bytes	351 bytes	
1,117			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		122 ms	266 bytes	2,688 bytes	
1,118			GET	http://46.101.23.188:30873/wp-content/themes/twentytw...	301 Moved Permanently		112 ms	304 bytes	358 bytes	
1,119			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		123 ms	266 bytes	2,688 bytes	
1,120			GET	http://46.101.23.188:30873/wp-content/themes/twentytw...	301 Moved Permanently		113 ms	308 bytes	362 bytes	
1,121			POST	http://46.101.23.188:30873/wp-comments-post.php	200 OK		126 ms	266 bytes	2,688 bytes	
1,122			POST	http://46.101.23.188:30873/wp-content/themes/twentytw...	200 OK		137 ms	266 bytes	2,688 bytes	
1,123			GET	http://46.101.23.188:30873/wp-content/themes/twentytw...	301 Moved Permanently		113 ms	307 bytes	361 bytes	
1,124			GET	http://46.101.23.188:30873/wp-content/themes/twentytw...	200 OK		111 ms	252 bytes	2,897 bytes	
Alerts 0 2 3 Primary Proxy: localhost:8081 Current Scans 0 0 0 1 0 0 0 0										

OWASP ZAP



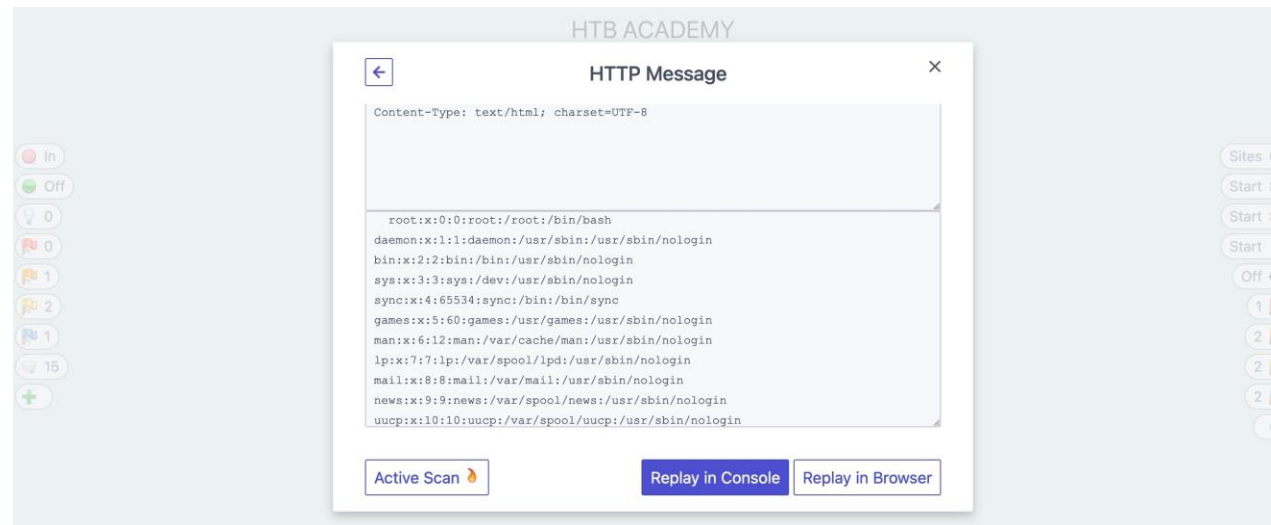
เมื่อการทำงานของ Active Scan เสร็จสิ้น เราสามารถเข้าไปดูรายการแจ้งเตือน (Alerts) เพื่อพิจารณาว่ารายการใดบ้างที่ควรต้องติดตามผลต่อแม้ว่าการแจ้งเตือนทั้งหมดควรถูกรายงานและนำมาพิจารณา แต่การแจ้งเตือนระดับสูง (High Alerts) นั้นมักจะเป็นช่องโหว่ที่นำไปสู่การถูกยึดครอง (Compromise) เว็บแอปพลิเคชัน หรือ Back-end Server ได้โดยตรงหากเราคลิกที่ปุ่ม High Alerts ระบบจะแสดงรายการแจ้งเตือนระดับความรุนแรงสูงที่ตรวจพบขึ้นมา:



OWASP ZAP



บนหน้าต่างรายละเอียดการแจ้งเตือน (Alert details window) เรายังสามารถคลิกที่ URL เพื่อดูรายละเอียดของ Request และ Response ที่ ZAP ใช้ในการตรวจสอบพบช่องโหว่นี้ และเรายังสามารถทำการส่ง Request นี้ซ้ำ (Repeat) ผ่านทาง ZAP HUD หรือ ZAP Request Editor ได้อีกด้วย:



การออกรายงาน (Reporting)

ในขั้นตอนสุดท้าย เราสามารถสร้างรายงาน (Report) ที่รวบรวมสิ่งที่ตรวจพบ (Findings) ทั้งหมดจากการสแกนรูปแบบต่างๆ ของ ZAP ได้ โดยไปที่แถบเมนูด้านบน แล้วเลือก (Report > Generate HTML Report) จากนั้นระบบจะให้เราระบุตำแหน่งที่ต้องการบันทึกไฟล์รายงานนอกจากนี้ เรายังสามารถส่งออก (Export) รายงานในรูปแบบอื่นๆ ได้อีก เช่น XML หรือ Markdown เมื่อสร้างรายงานเสร็จเรียบร้อยแล้ว เราสามารถเปิดดูไฟล์ดังกล่าวผ่านเว็บเบราว์เซอร์ใดก็ได้:

Summary of Alerts

Risk Level	Number of Alerts
High	3
Medium	5
Low	4
Informational	3
False Positives:	0

Summary of Sequences

For each step: result (Pass/Fail) - risk (of highest alert(s) for the step, if any).

Alerts

Name	Risk Level	Number of Instances
Remote OS Command Injection	High	1
SQL Injection	High	1
SQL Injection - Oracle - Time Based	High	2
Absence of Anti-CSRF Tokens	Medium	2
Content Security Policy (CSP) Header Not Set	Medium	12
Directory Browsing	Medium	9
HTTP to HTTPS Insecure Transition in Form Post	Medium	1
Missing Anti-clickjacking Header	Medium	7
Cookie No HttpOnly Flag	Low	3
Cookie without SameSite Attribute	Low	3
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	36
X-Content-Type-Options Header Missing	Low	22
Cookie Poisoning	Informational	3
Session Management Response Identified	Informational	1

OWASP ZAP



ZAP Marketplace

ZAP ยังมีฟีเจอร์ในการขยายความสามารถ (Extensibility) เป็นของตัวเองผ่านทาง Marketplace ซึ่งช่วยให้เราสามารถติดตั้ง Add-ons ประเภทต่างๆ ที่ถูกพัฒนาโดยชุมชนผู้ใช้งานได้ในการเข้าถึง Marketplace ของ ZAP ให้เราคลิกที่ปุ่ม Manage Add-ons จากนั้นเลือกไปที่แท็บ Marketplace:

The screenshot shows the OWASP ZAP interface. The 'Standard Mode' dropdown is visible at the top left. The 'Sites' tab is active on the left sidebar. The 'Manage Add-ons' window is open, showing the 'Marketplace' tab. A list of available add-ons is displayed with columns for Status, Name, Description, Update, and Selected. The 'FuzzDB Offensive' add-on is highlighted. Below the list, details for the selected add-on are shown, including its name, status, and version. At the bottom right of the window are buttons for 'Install Selected', 'More Info', and 'Close'.

Status	Name	Description	Update	Selected
Release	Core Language Files	Translations of the core language files	15.0.0	☐
Release	Custom Payloads	Ability to add, edit or remove payloads tha...	0.15.0	☐
Release	Directory List v2.3	Lists of directory names to be used with F...	4.0.0	☐
Release	Directory List v2.3 LC	Lists of lower case directory names to be ...	4.0.0	☐
Release	FuzzAI Files	FuzzAI files which can be used with the Z...	0.0.1	☐
Release	FuzzDB Files	FuzzDB files which can be used with the Z...	9.0.0	☒
Release	FuzzDB Offensive	FuzzDB web backdoors and attack files w...	5.0.0	☒
Release	Help - Chinese Simplifi...	Chinese Simplified version of the ZAP help ...	4.0.0	☐
Release	Help - Filipino	Filipino version of the ZAP help file.	4.0.0	☐
Release	Help - Indonesian	Indonesian version of the ZAP help file.	4.0.0	☐
Release	Help - Portuguese, Br...	Portuguese, Brazilian version of the ZAP h...	12.0.0	☐
Release	Help - Russian	Russian version of the ZAP help file.	3.0.0	☐
Release	Help - Spanish	Spanish version of the ZAP help file.	11.0.0	☐
Release	Help - Turkish	Turkish version of the ZAP help file.	3.0.0	☐

Name FuzzDB Offensive
Status Release
Version 5.0.0

Install Selected More Info Close

Snyk Code



← → ↺ snyk.io/product/snyk-code/ ☆ 📁 📄 J ⋮

 Platform ▾ Resources ▾ Company ▾ Pricing **Evo** New  EN Login ▾ [Sign up](#) [Book a live demo](#)

Snyk Code Overview Docs

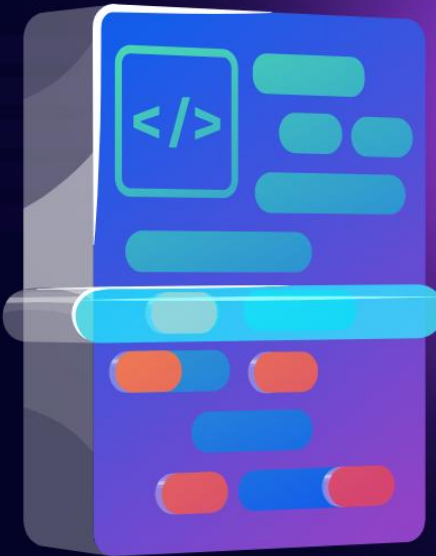
Snyk named a Leader in The Forrester Wave™: SAST, Q3 2025 →

SNYK CODE

FIND, PRIORITIZE, AND AUTO-FIX ISSUES WITH DEV-FOCUSED SAST SOLUTIONS

Find and auto-fix the most critical unsafe code up to 50x faster, with pre-validated fixes from a static application security testing tool built by and for developers.

[Book a live demo](#)

A stylized illustration of a code editor interface. It features a dark blue background with a light blue border. Inside, there's a code editor with a light blue background and a dark blue border. The editor contains a code snippet with a bug icon (a circle with a diagonal line) and a checkmark icon (a circle with a checkmark). Below the code editor, there are several colored buttons (red, blue, green) and a progress bar.

Snyk Code



app.snyk.io/org/sinority/projects?groupBy=targets&before&after&searchQuery=&sortBy=highest+severity&filters[Show]=&filters[Integrations]=&filters[CollectionIds]=

snyk sinority > Projects Add projects View import logs

TENANT
Cybersafe

GROUP
Cybersafe

ORGANIZATION
sinority

Dashboard
Projects
Integrations
Members
Settings

All projects

Add filter

Group by targets Sort by highest severity

Targets 2

Search targets

>	vedadinoft/juice-shop	9 C	41 H	46 M	6 L	...
>	vedadinoft/TestCICD	4 C	27 H	89 M	41 L	...

Ready to import another project?

Secure your entire stack with Snyk

Add projects

Snyk Code



Code Analysis

Overview History Settings

VULNERABILITY TYPES

- ☐ Server-Side Request Fo... 47
- ☐ Improper Certificate Va... 13
- ☐ Cross-site Scripting (X... 12
- ☐ Use of Hardcoded Pass... 8
- ☐ Use of Hardcoded Crede... 8
- ☐ Path Traversal 7
- ☐ Open Redirect 5
- ☐ Deserialization of Untru... 3
- ☐ Use of Password Hash ... 3
- ☐ XML External Entity (XX... 3
- ☐ Hardcoded Non-Cryptog... 3
- ☐ File Inclusion 1
- ☐ Binding to all network in... 1
- ☐ Use of Hardcoded Crypt... 1
- ☐ Use of Hardcoded Crypt... 1
- ☐ Insecure Temporary File 1
- ☐ Jinja auto-escape is set ... 1

```
66 files = {"file": ("data", data + crc32, "text/data")}
67
68 resp = requests.post(url + "/apisix/admin/migrate/import", files=files, verify=False)
```

Certificate verification is disabled by setting **verify** to **False** in **requests.post**. This may lead to Man-in-the-middle attacks.

[apisix/CVE-2021-45232/apisix_dashboard_rce.py](#)

3 steps in 1 file

Ignore

View details

H Improper Certificate Validation - SSL Verification Bypass

SNYK CODE | CWE-295

SCORE
811

```
30 print('[*] Creating a session using the provided credential...')
31 get_params = {'q': 'user/login'}
32 post_params = {'form_id': 'user_login', 'name': username, 'pass': password, 'op': 'Log in'}
33 print('[*] Finding User ID...')
34 session.post(target, params=get_params, data=post_params, verify=False, proxies=proxyConf)
```

Certificate verification is disabled by setting **verify** to **False** in **post**. This may lead to Man-in-the-middle attacks.

[drupal/CVE-2018-7602/drupa7-CVE-2018-7602.py](#)

3 steps in 1 file

Ignore

View details

1 of 122 issues

Group by none Sort by highest severity

H SQL Injection

SNYK CODE | CWE-89

SCORE
201

```
15 ctx.response.type = 'html'
16
17 let id = ctx.request.query.id || 1
18 let sql = `SELECT * FROM "user" WHERE "id"
19 const res = await client.query(sql)
```

Unsanitized input from *an HTTP parameter flows* into *query*, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

[node/CVE-2017-16082/www/app.js](#)

 Learn about this type of vulnerability and how to fix it

H SQL Injection

SNYK CODE | CWE-89

Unsanitized input from *an HTTP parameter flows* into *query*, where it is used in an SQL query. This may result in an SQL Injection vulnerability.

Data flow - 12 steps in 1 file

Find out how to remediate this issue through our [Fix analysis](#)

[Data flow](#) [Fix analysis](#)[node/CVE-2017-16082/www/app.js](#)

```
10 port = 3432
11 })
12 client.connect()
13
14 app.use(async ctx => {
15   ctx.response.type = 'html'
16
17   let id = ctx.request.query.id || 1
18   let sql = `SELECT * FROM "user" WHERE "id" = ${id}`
19   const res = await client.query(sql)
20
21   ctx.body = `<html>
22     <body>
23       <table>
24         <tr><th>id</th><td>${res.rows[0].id}</td></tr>
25         <tr><th>name</th><td>${res.rows[0].name}</td></tr>
26         <tr><th>score</th><td>${res.rows[0].score}</td></tr>
27       </table>
28     </body>
29   </html>`
30 })
31
32 app.listen(3000)
```

 Ignore

H SQL Injection

SNYK CODE | CWE-89

Data flow

Fix analysis



Details

In an SQL injection attack, the user can submit an SQL query directly to the database, gaining access without providing appropriate credentials. Attackers can then view, export, modify, and delete confidential information; change passwords and other authentication information; and possibly gain access to other systems within the network. This is one of the most commonly exploited categories of vulnerability, but can largely be avoided through good coding practices.

Best practices for prevention

- Avoid passing user-entered parameters directly to the SQL server.
- Avoid using string concatenation to build SQL queries from user-entered parameters.
- When coding, define SQL code first, then pass in parameters. Use prepared statements with parameterized queries. Examples include `SqlCommand()` in .NET and `bindParam()` in PHP.
- Use strong typing for all parameters so unexpected user data will be rejected.
- Where direct user input cannot be avoided for performance reasons, validate input against a very strict allowlist of permitted characters, avoiding special characters such as `?'<> ; - ' " \` and spaces. Use a vendor-supplied escaping routine if possible.
- Develop your application in an environment and/or using libraries that provide protection against SQL injection.
- Harden your entire environment around a least-privilege model, ideally with isolated accounts with privileges only for particular tasks.

Example fixes

andrei-tatar/HAP.js 1 / 3

```

7  - var tableName = "Temp" + req.body.id;
8  7+ var device = node.device(req.body.id);
9  8  -
10 9+ var time = Math.round(Date.now()/1000);
11 10 - if (device) {
12 11+
13 12+     device.init_temperature = true;
14 13+
15 14+     util.createProperty(device, 'temperature', req.body.
16 15+
17 16+
18 17+
19 18+
20 19+
21 20+
22 21+
23 22+
24 23+
25 24+
26 25+
27 26+
28 27+
29 28+
30 29+
31 30+
32 31+
33 32+
34 33+
35 34+
36 35+
37 36+
38 37+
39 38+
40 39+
41 40+
42 41+
43 42+
44 43+
45 44+
46 45+
47 46+
48 47+
49 48+
50 49+
51 50+
52 51+
53 52+
54 53+
55 54+
56 55+
57 56+
58 57+
59 58+
60 59+
61 60+
62 61+
63 62+
64 63+
65 64+
66 65+
67 66+
68 67+
69 68+
70 69+
71 70+
72 71+
73 72+
74 73+
75 74+
76 75+
77 76+
78 77+
79 78+
80 79+
81 80+
82 81+
83 82+
84 83+
85 84+
86 85+
87 86+
88 87+
89 88+
90 89+
91 90+
92 91+
93 92+
94 93+
95 94+
96 95+
97 96+
98 97+
99 98+
100 99+
101 100+
102 101+
103 102+
104 103+
105 104+
106 105+
107 106+
108 107+
109 108+
110 109+
111 110+
112 111+
113 112+
114 113+
115 114+
116 115+
117 116+
118 117+
119 118+
120 119+
121 120+
122 121+
123 122+
124 123+
125 124+
126 125+
127 126+
128 127+
129 128+
130 129+
131 130+
132 131+
133 132+
134 133+
135 134+
136 135+
137 136+
138 137+
139 138+
140 139+
141 140+
142 141+
143 142+
144 143+
145 144+
146 145+
147 146+
148 147+
149 148+
150 149+
151 150+
152 151+
153 152+
154 153+
155 154+
156 155+
157 156+
158 157+
159 158+
160 159+
161 160+
162 161+
163 162+
164 163+
165 164+
166 165+
167 166+
168 167+
169 168+
170 169+
171 170+
172 171+
173 172+
174 173+
175 174+
176 175+
177 176+
178 177+
179 178+
180 179+
181 180+
182 181+
183 182+
184 183+
185 184+
186 185+
187 186+
188 187+
189 188+
190 189+
191 190+
192 191+
193 192+
194 193+
195 194+
196 195+
197 196+
198 197+
199 198+
200 199+
201 200+
202 201+
203 202+
204 203+
205 204+
206 205+
207 206+
208 207+
209 208+
210 209+
211 210+
212 211+
213 212+
214 213+
215 214+
216 215+
217 216+
218 217+
219 218+
220 219+
221 220+
222 221+
223 222+
224 223+
225 224+
226 225+
227 226+
228 227+
229 228+
230 229+
231 230+
232 231+
233 232+
234 233+
235 234+
236 235+
237 236+
238 237+
239 238+
240 239+
241 240+
242 241+
243 242+
244 243+
245 244+
246 245+
247 246+
248 247+
249 248+
250 249+
251 250+
252 251+
253 252+
254 253+
255 254+
256 255+
257 256+
258 257+
259 258+
260 259+
261 260+
262 261+
263 262+
264 263+
265 264+
266 265+
267 266+
268 267+
269 268+
270 269+
271 270+
272 271+
273 272+
274 273+
275 274+
276 275+
277 276+
278 277+
279 278+
280 279+
281 280+
282 281+
283 282+
284 283+
285 284+
286 285+
287 286+
288 287+
289 288+
290 289+
291 290+
292 291+
293 292+
294 293+
295 294+
296 295+
297 296+
298 297+
299 298+
300 299+
301 300+
302 301+
303 302+
304 303+
305 304+
306 305+
307 306+
308 307+
309 308+
310 309+
311 310+
312 311+
313 312+
314 313+
315 314+
316 315+
317 316+
318 317+
319 318+
320 319+
321 320+
322 321+
323 322+
324 323+
325 324+
326 325+
327 326+
328 327+
329 328+
330 329+
331 330+
332 331+
333 332+
334 333+
335 334+
336 335+
337 336+
338 337+
339 338+
340 339+
341 340+
342 341+
343 342+
344 343+
345 344+
346 345+
347 346+
348 347+
349 348+
350 349+
351 350+
352 351+
353 352+
354 353+
355 354+
356 355+
357 356+
358 357+
359 358+
360 359+
361 360+
362 361+
363 362+
364 363+
365 364+
366 365+
367 366+
368 367+
369 368+
370 369+
371 370+
372 371+
373 372+
374 373+
375 374+
376 375+
377 376+
378 377+
379 378+
380 379+
381 380+
382 381+
383 382+
384 383+
385 384+
386 385+
387 386+
388 387+
389 388+
390 389+
391 390+
392 391+
393 392+
394 393+
395 394+
396 395+
397 396+
398 397+
399 398+
400 399+
401 400+
402 401+
403 402+
404 403+
405 404+
406 405+
407 406+
408 407+
409 408+
410 409+
411 410+
412 411+
413 412+
414 413+
415 414+
416 415+
417 416+
418 417+
419 418+
420 419+
421 420+
422 421+
423 422+
424 423+
425 424+
426 425+
427 426+
428 427+
429 428+
430 429+
431 430+
432 431+
433 432+
434 433+
435 434+
436 435+
437 436+
438 437+
439 438+
440 439+
441 440+
442 441+
443 442+
444 443+
445 444+
446 445+
447 446+
448 447+
449 448+
450 449+
451 450+
452 451+
453 452+
454 453+
455 454+
456 455+
457 456+
458 457+
459 458+
460 459+
461 460+
462 461+
463 462+
464 463+
465 464+
466 465+
467 466+
468 467+
469 468+
470 469+
471 470+
472 471+
473 472+
474 473+
475 474+
476 475+
477 476+
478 477+
479 478+
480 479+
481 480+
482 481+
483 482+
484 483+
485 484+
486 485+
487 486+
488 487+
489 488+
490 489+
491 490+
492 491+
493 492+
494 493+
495 494+
496 495+
497 496+
498 497+
499 498+
500 499+
501 500+
502 501+
503 502+
504 503+
505 504+
506 505+
507 506+
508 507+
509 508+
510 509+
511 510+
512 511+
513 512+
514 513+
515 514+
516 515+
517 516+
518 517+
519 518+
520 519+
521 520+
522 521+
523 522+
524 523+
525 524+
526 525+
527 526+
528 527+
529 528+
530 529+
531 530+
532 531+
533 532+
534 533+
535 534+
536 535+
537 536+
538 537+
539 538+
540 539+
541 540+
542 541+
543 542+
544 543+
545 544+
546 545+
547 546+
548 547+
549 548+
550 549+
551 550+
552 551+
553 552+
554 553+
555 554+
556 555+
557 556+
558 557+
559 558+
560 559+
561 560+
562 561+
563 562+
564 563+
565 564+
566 565+
567 566+
568 567+
569 568+
570 569+
571 570+
572 571+
573 572+
574 573+
575 574+
576 575+
577 576+
578 577+
579 578+
580 579+
581 580+
582 581+
583 582+
584 583+
585 584+
586 585+
587 586+
588 587+
589 588+
590 589+
591 590+
592 591+
593 592+
594 593+
595 594+
596 595+
597 596+
598 597+
599 598+
600 599+
601 600+
602 601+
603 602+
604 603+
605 604+
606 605+
607 606+
608 607+
609 608+
610 609+
611 610+
612 611+
613 612+
614 613+
615 614+
616 615+
617 616+
618 617+
619 618+
620 619+
621 620+
622 621+
623 622+
624 623+
625 624+
626 625+
627 626+
628 627+
629 628+
630 629+
631 630+
632 631+
633 632+
634 633+
635 634+
636 635+
637 636+
638 637+
639 638+
640 639+
641 640+
642 641+
643 642+
644 643+
645 644+
646 645+
647 646+
648 647+
649 648+
650 649+
651 650+
652 651+
653 652+
654 653+
655 654+
656 655+
657 656+
658 657+
659 658+
660 659+
661 660+
662 661+
663 662+
664 663+
665 664+
666 665+
667 666+
668 667+
669 668+
670 669+
671 670+
672 671+
673 672+
674 673+
675 674+
676 675+
677 676+
678 677+
679 678+
680 679+
681 680+
682 681+
683 682+
684 683+
685 684+
686 685+
687 686+
688 687+
689 688+
690 689+
691 690+
692 691+
693 692+
694 693+
695 694+
696 695+
697 696+
698 697+
699 698+
700 699+
701 700+
702 701+
703 702+
704 703+
705 704+
706 705+
707 706+
708 707+
709 708+
710 709+
711 710+
712 711+
713 712+
714 713+
715 714+
716 715+
717 716+
718 717+
719 718+
720 719+
721 720+
722 721+
723 722+
724 723+
725 724+
726 725+
727 726+
728 727+
729 728+
730 729+
731 730+
732 731+
733 732+
734 733+
735 734+
736 735+
737 736+
738 737+
739 738+
740 739+
741 740+
742 741+
743 742+
744 743+
745 744+
746 745+
747 746+
748 747+
749 748+
750 749+
751 750+
752 751+
753 752+
754 753+
755 754+
756 755+
757 756+
758 757+
759 758+
760 759+
761 760+
762 761+
763 762+
764 763+
765 764+
766 765+
767 766+
768 767+
769 768+
770 769+
771 770+
772 771+
773 772+
774 773+
775 774+
776 775+
777 776+
778 777+
779 778+
780 779+
781 780+
782 781+
783 782+
784 783+
785 784+
786 785+
787 786+
788 787+
789 788+
790 789+
791 790+
792 791+
793 792+
794 793+
795 794+
796 795+
797 796+
798 797+
799 798+
800 799+
801 800+
802 801+
803 802+
804 803+
805 804+
806 805+
807 806+
808 807+
809 808+
810 809+
811 810+
812 811+
813 812+
814 813+
815 814+
816 815+
817 816+
818 817+
819 818+
820 819+
821 820+
822 821+
823 822+
824 823+
825 824+
826 825+
827 826+
828 827+
829 828+
830 829+
831 830+
832 831+
833 832+
834 833+
835 834+
836 835+
837 836+
838 837+
839 838+
840 839+
841 840+
842 841+
843 842+
844 843+
845 844+
846 845+
847 846+
848 847+
849 848+
850 849+
851 850+
852 851+
853 852+
854 853+
855 854+
856 855+
857 856+
858 857+
859 858+
860 859+
861 860+
862 861+
863 862+
864 863+
865 864+
866 865+
867 866+
868 867+
869 868+
870 869+
871 870+
872 871+
873 872+
874 873+
875 874+
876 875+
877 876+
878 877+
879 878+
880 879+
881 880+
882 881+
883 882+
884 883+
885 884+
886 885+
887 886+
888 887+
889 888+
890 889+
891 890+
892 891+
893 892+
894 893+
895 894+
896 895+
897 896+
898 897+
899 898+
900 899+
901 900+
902 901+
903 902+
904 903+
905 904+
906 905+
907 906+
908 907+
909 908+
910 909+
911 910+
912 911+
913 912+
914 913+
915 914+
916 915+
917 916+
918 917+
919 918+
920 919+
921 920+
922 921+
923 922+
924 923+
925 924+
926 925+
927 926+
928 927+
929 928+
930 929+
931 930+
932 931+
933 932+
934 933+
935 934+
936 935+
937 936+
938 937+
939 938+
940 939+
941 940+
942 941+
943 942+
944 943+
945 944+
946 945+
947 946+
948 947+
949 948+
950 949+
951 950+
952 951+
953 952+
954 953+
955 954+
956 955+
957 956+
958 957+
959 958+
960 959+
961 960+
962 961+
963 962+
964 963+
965 964+
966 965+
967 966+
968 967+
969 968+
970 969+
971 970+
972 971+
973 972+
974 973+
975 974+
976 975+
977 976+
978 977+
979 978+
980 979+
981 980+
982 981+
983 982+
984 983+
985 984+
986 985+
987 986+
988 987+
989 988+
990 989+
991 990+
992 991+
993 992+
994 993+
995 994+
996 995+
997 996+
998 997+
999 998+
1000 999+

```

Ignore



ตอบคำถามผู้เข้าอบรม

