Written by Dr. Thapanapong Rukkanchanunt

14 Security 1

OUTLINE

- Web Security
- The Premise
- Type of Threats
  - SQL Injection
  - Cross-site Scripting
  - Buffer Overflow
  - Cross-site Request Forgery
  - Data Breach

Web Security

- ในความหมายของความปลอดภัยบนเว็บ เราสามารถคิดในมุมมองของผู้ใช้และผู้สร้างได้
- ในบทเรียนนี้เราจะโฟกัสเนื้อหาที่ฝั่งของผู้สร้างซึ่งจำเป็นต้องป้องกันภัยจากภายนอกในระดับหนึ่ง
- Web Security ในบริบทนี้จึงหมายถึงการสร้างมาตรการป้องกันและระบบโปรโตคอลโดยมีจุดมุ่งหมายเพื่อป้องกันเว็บไม่ให้ถูกแฮกหรือถูกใช้งานโดยไม่ได้รับอนุญาต
- เนื่องจากเว็บแอพทำงานบนพื้นฐานของอินเทอร์เน็ตจึงถูกโจมตีในหลายระดับอีกทั้งเรายังสามารถเข้าถึงข้อมูลบางส่งของเว็บแอพโดยไม่จำเป็นต้องเป็นผู้สร้างได้

The Premise

- ก่อนที่เราจะลงรายละเอียดเรื่องการรูปการถูกโจมตีและวิธีป้องกัน เราจะมาดูปัจจัยที่เราต้องคำนึงถึง แน่นอนที่ว่าแต่ละเว็บใช้วิธีการป้องกันที่แตกต่างกันไป การเรียนรู้จากตัวอย่างดังกล่าวจึงเป็นสิ่งที่จำเป็นสำหรับการสร้างแนวป้องกันของเราเอง
- มาตรฐานที่ใช้กันอย่างแพร่หลายในปัจจุบันเป็นข้อตกลงผ่านกลุ่ม OWASP (The Open Web Application Security Project) ผู้พัฒนาเว็บของบริษัท Cybersecurity ชั้นนำก็ทำตามมาตรฐานนี้ นอกจากนี้ก็ติดตามข่าวเกี่ยวกับการแฮก (Web Hacking Incident Database) อยู่เสมอ
- เพราะฉะนั้น Security ไม่ใช่สิ่งที่ทำครั้งเดียวแล้วเสร็จ จำเป็นต้องมีการอัพเดตอยู่เสมอ แต่ไม่ว่าระบบจะปลอดภัยอย่างไรแฮกเกอร์ก็สามารถหาทางแฮกได้เสมอ

SQL Injection

- SQI Injection หรือ SQi คือการที่แฮกเกอร์หาช่องโหว่ของระบบผ่านการใช้งาน Specialized Query ที่ทำหน้าที่เช่นเดียวกันกับฐานข้อมูลเรียกดูข้อมูลผ่าน Query
- โดยปกติแล้วค่า FieId ใน Query จะเป็นข้อมูลหนึ่งประเภทเช่นตัวเลขหรือข้อความ แต่ถ้าแฮกเกอร์ส่งข้อมูลพร้อมคำสั่งเพิ่มเติมด้วยก็จะทำให้การประมวลผล Query ผิดไป
- ตัวอย่าง Query ปกติ ผู้ใช้จะต้องเติมตัวเลขสำหรับการดึงข้อมูลนักเรียนรหัสนั้น

```
studentId = getRequestString("studentId");

lookupStudent  = "SELECT * FROM students WHERE studentId = " + studentId;
```

Please enter your student ID number: 117

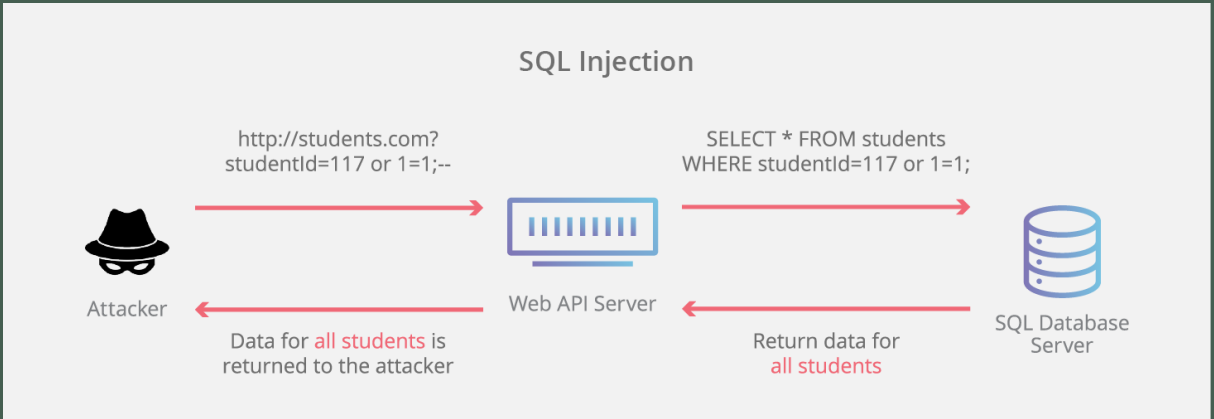
SQL Injection (2)

- ถ้าผู้ใช้กรอก 117 จะได้ผลลัพธ์ข้อความเป็น

  SELECT * FROM students WHERE studentId = 117

- แต่ถ้าแฮกเกอร์กรอก 117 OR 1=1 ผลลัพธ์จะเป็นอย่างไร

  SELECT * FROM students WHERE studentId = 117 OR 1=1

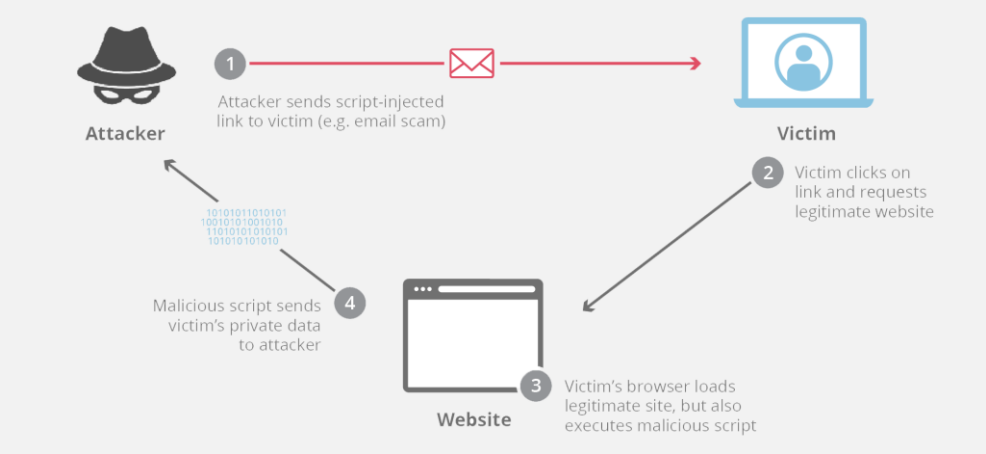


SQL Injection (3)

- ผู้ใช้ SQi ตั้งเป้าหมายไว้ที่ API ที่มีช่องโหว่ โดยการส่ง Request ที่มีคำสั่งพิเศษ
- มีโปรแกรมอัตโนมัติที่สแกนเว็บเพื่อค้นหาฟอร์มแล้วพยายามส่ง SQL Query ที่ผู้สร้างไม่ได้ป้องกันไว้
- SQi ทำได้ง่ายแต่ก็ป้องกันได้ง่ายเช่นกัน ในความเป็นจริงนั้นถ้าเราพัฒนาเว็บไม่ทันเวลาเราอาจจะละเลยเหตุการณ์นี้ได้ ผู้พัฒนาเว็บมือสมัครเล่นอาจจะข้ามการตรวจสอบตรงนี้ไปเลย
- การโจมตีด้วย SQi มักจะทิ้งร่องลอยไว้เสมอ (ฐานข้อมูลมี log การเรียกดูข้อมูล) ดังนั้นแฮกเกอร์มักจะปะปนการโจมตีนี้ในการโจมตีแบบ DDoS

Cross-Site Scripting (XSS)

- Cross-Site Scripting คือการที่แฮกเกอร์แปะโค้ดไว้ยังเว็บปกติที่เหยื่อเข้าใช้งาน
- โค้ดถูกแปะได้หลายแบบ ที่นิยมมากคือแปะต่อท้าย URL แล้วส่งไปตามอีเมล์หรือโพสลงโซเชียล
- ในทางเทคนิค XSS จะถูกเรียกว่าการโจมตีแบบ Client-Side Code Injection



Client-Side Code Injection

- Client-Side Code ในบริบทของ XSS หมายถึงโค้ดที่คุมควบการทำงานเชิงตอบโต้ของเว็บ นั่นคือ JavaScript
- ลองนึกภาพแฮกเกอร์โพสข้อความในเว็บบอร์ด แล้วในข้อความนั้นมีโค้ด <script>...</script> ถ้าเว็บบอร์ดไม่มีการตรวจสอบ Web Browser ก็จะประมวลข้อความนั้นเป็นโค้ด JavaScript ไม่ว่าใครที่เข้ามาเห็นข้อความนั้นก็จะกลายเป็นเหยื่อแบบไม่รู้ตัว
- ความเสียหายคือข้อมูลที่ถูกบันทึกไว้ใน Cookies ซึ่งสามารถเข้าถึงผ่านทาง JS ได้ แฮกเกอร์สามารถปลอมตัวเป็นเหยื่อได้ อีกทั้ง JS มีความสามารถในการส่ง HTTP Request ดังนั้นข้อมูลใน Cookies สามารถส่งไปยังแฮกเกอร์โดยตรงได้เลย

Reflected XSS

- Reflected XSS เป็นรูปแบบที่ใช้บ่อยที่สุด คือการเขียน Script ต่อท้าย URL

  http://realbank.com/index.php?user=&script>here is some bad code!</script>

- งานของแฮกเกอร์คือการสร้างแรงจูงใจให้กด URL ดังกล่าว

Persistent XSS

- Persistent XSS เกิดขึ้นกับเว็บที่อนุญาตให้ผู้ใช้โพสข้อความ ส่วนมากจะเป็นเว็บบอร์ดหรือโซเชียลมีเดีย แฮกเกอร์อาจจะสร้างโปรไฟล์แล้วแปะโค้ดในข้อความแนะนำตัว

  "Hi! My name is Dave. Nice to meet you<script>malicious code here</script>"

- เมื่อผู้ใช้คนอื่นเข้ามาดูโปรไฟล์ก็จะกลายเป็นเหยื่อโดยไม่รู้ตัว
- เหตุการณ์จริงที่เกิดขึ้นแล้วทำให้เกิดความเสียหายมากที่สุดคือคดี MySpace ในปี 2005 เมื่อ Samy Kamkar แปะ XSS Worm ไว้ในโปรไฟล์ โดยคนที่เข้ามาดูจะส่ง Friend Request ไปยัง Samy พร้อมทั้งเปลี่ยนโปรไฟล์ตัวเองเป็น XSS Worm ผ่านไป 20 ชั่วโมง มีผู้ใช้ 1 ล้านคนเป็นเหยื่อของเหตุการณ์นี้ เขาถูกจับและถูกควบคุมความประพฤติ 3 ปี ระหว่างนั้นใช้ได้แค่คอมพิวเตอร์ที่ไม่มีอินเทอร์เน็ต

Buffer Overflow

- Buffer Overflow คือความผิดปกติที่เกิดขึ้นเมื่อโปรแกรมเขียนข้อมูลลง Buffer แต่ขนาดข้อมูลเกินขนาดของ Buffer ทำให้พื้นที่หน่วยความจำที่อยู่ติดกันถูกเขียนทับ
- เป้าหมายของแฮกเกอร์คือพยายามแก้ไขหน่วยความจำของเครื่องให้สามารถควบคุมการรันของโปรแกรมได้ กล่าวคือถ้าโปรแกรมมีโครงสร้างหน่วยความจำที่เป็นระบบ แฮกเกอร์จะรู้ว่าหน่วยความจำตรงไหนที่เป็นโค้ดที่รันโปรแกรม ดังนั้นเมื่อ Buffer Overflow เกิดขึ้น โค้ดที่ต้องการรันจะสามารถไปแทนที่โค้ดส่วนนั้นได้
- ถ้าโค้ดที่ถูกแทนที่เป็น Pointer แฮกเกอร์สามารถเปลี่ยนที่อยู่ที่ Pointer ชี้ไปเป็นตำแหน่งที่มีโค้ดที่ต้องการัน

Vulnerability

- ภาษาบางภาษาไม่มีการป้องกัน Buffer Overflow เช่นภาษา C, C++ ระบบปฏิบัติการส่วนมากถูกเขียนด้วยภาษาเหล่านี้
- ภาษาใหม่ ๆ เช่น Java, PERL, C# มีกลไกที่ป้องกันการเกิด Buffer Overflow
- รูปแบบการโจมตีแบบ Buffer Overflow สามารถแยกย่อยได้หลายแบบเช่น
  - Stack Overflow เป้าหมายคือ Buffer ประเภท Stack (First-in, Last-out)
  - Heap Overflow เป้าหมายคือ Buffer ประเภท Heap (Extra Memory)
  - Integer Overflow ใช้ตัวดำเนินการทางคณิตศาสตร์กับตัวแปรประเภทจำนวนเต็ม
  - Unicode Overflow แทนที่ตัวแปรที่เก็บตัวอักษรแบบ ASCII ด้วย Unicode

Heartbleed Incident

- เวลาเราล็อกอินเพื่อใช้งานเว็บแอพ ระบบจะต้องตรวจสอบความถูกต้อง แต่ถ้าหากเว็บแอพส่งข้อมูลไปตรง ๆ แฮกเกอร์สามารถขโมยข้อมูลได้ ดังนั้นเราจึงใช้ OpenSSL เพื่อเข้ารหัสข้อมูล (แฮกเกอร์ยังขโมยได้ แต่จะอ่านข้อมูลไม่ได้)
- แต่ในระหว่างการส่งข้อมูลโดยใช้ OpenSSL นั้น มีการส่งข้อมูลอีกชุดหนึ่งในรูปแบบของ Heartbleed เพื่อตรวจสอบว่าทั้งสองฝ่ายไม่ได้ Sleep แต่ข้อมูลชุดนี้ไม่มีข้อมูลที่สำคัญเลยไม่มีการเข้ารหัส แฮกเกอร์จึงสามารถส่งโค้ดผ่านช่องทางนี้ได้
- เนื่องจาก Buffer ของช่องทางนี้ใช้พื้นที่ไม่มาก ทำให้แฮกเกอร์ Overflow ได้ง่ายโดยการส่งข้อความยาว ๆ

Cross-Site Request Forgery

- Cross-Site Request Forgery คือการหลอกผู้ใช้ให้ใช้ข้อมูลส่วนตัวในการเปลี่ยนสถานะบางอย่างเช่นโอนเงิน เปลี่ยนอีเมล์ เปลี่ยนรหัสผ่าน
- แม้ว่าการโจมตีแนวนี้จะเน้นไปที่ผู้ใช้ แต่ถ้าหากผู้ใช้เป็นแอดมินก็อาจจะทำให้ทั้งระบบถูกควบคุมได้ ทั้งเว็บแอพ API และ Service อื่น
- ขั้นตอนคร่าว ๆ คือแฮกเกอร์ปลอมแปลง Request สำหรับส่งไปยังเซิฟเวอร์ (เช่นโอนเงิน) แฮกเกอร์แปะ Request นี้ไว้กับ URL แล้วส่งให้เหยื่อทางอีเมล์ เมื่อเหยื่อคลิก URL ระบบก็จะทำการโอนเงินเพราะเหยื่อล็อกอินอยู่แล้ว

HTTP Vulnerability

- HTTP GET ดึงข้อมูลเช่นรูปภาพ แม้ว่า GET จะไม่เปลี่ยนสถานะแต่ข้อมูลที่ได้มาก็สามารถนำไปใช้งานต่อได้
- HTTP POST ใช้สำหรับเปลี่ยนสถานะโดยเฉพาะ จึงเป็นจุดเสี่ยงในการถูกโจมตี Web Browser มีมาตรการรักษาความปลอดภัยสำหรับ POST ด้วยการใช้ Same Origin Policy (SOP) และ Cross Origin Resource Sharing (CORS)
  - SOP อนุญาตให้ส่ง Request จากต้นทางเดียวกัน
  - CORS อนุญาตให้ส่ง Request บางประเภทถ้าไม่ได้มาจากต้นทางเดียวกัน
- HTTP PUT, DELETE ก็โอกาสเสี่ยงเหมือน POST แต่ก็สามารถใช้ SOP และ CORS ได้

Data Breach

- Data Breach คือการรั่วไหลของข้อมูลส่วนตัวไปยังอุปกรณ์หรือสิ่งแวดล้อมที่ไม่ถูกป้องกัน สามารถเกิดขึ้นได้โดยไม่ตั้งใจ หรือเป็นผลต่อเนื่องมาจากการโจมตี
- สาเหตุการเกิด Data Breach อาจจะมาจากการขโมยข้อมูลโดยตรงผ่าน Brute Force หรือ Man-in-the-Middle Attack
- Social Engineering คือการปลอมตัวเป็นเจ้าหน้าที่รัฐแล้วหลอกขอดูข้อมูล
- Insider Threats เป็นรั่วไหลของข้อมูลโดยคนในหรือผู้ดูแลระบบเอง
- Credential Fraud เกิดขึ้นเมื่อเหยื่อใช้ข้อมูลล็อกอินเหมือนกันในทุกบริการ

Target Incident

- ในปี 2013 ห้าง Target (เทียบเท่า Big-C, Lotus ของไทย) กลายเป็นเหยื่อของ Data Breach โดยข่าวนี้ได้รับความสนใจมาก เพราะการโจมตีซับซ้อนและหลากหลาย
- จุดเริ่มต้นมาจาก Phishing Scam ของพนักงานบริษัทแอร์ที่ Target ทำสัญญาไว้
- แอร์ทุกจุดในห้าง Target เชื่อมต่อกับคอมพิวเตอร์ที่ควบคุมระบบไฟฟ้าทั้งหมด
- แฮกเกอร์ควบคุมการทำงานของแอร์ซึ่งทำให้สามารถควบคุมระบบห้างทั้งหมดได้
- สุดท้ายแฮกเกอร์แก้ไขโค้ดของเครื่องสแกนบัตรเครดิต
- แม้ว่าเครื่องเหล่านี้จะไม่เชื่อมต่ออินเทอร์เน็ต แฮกเกอร์ก็สามารถสั่งให้ Dump ข้อมูลเป็นระยะ ๆ ไปยัง Access Point ที่แฮกเกอร์เฝ้าดูอยู่

Next Lesson

- ในบทเรียนถัดไปเราจะเรียนรู้วิธีการแก้ไขและป้องกันการโจมตีทั้งหมดที่กล่าวมา

Project Checkpoint 3

- Deadline: 31 มีนาคม 23:59
- ส่งได้สามทาง
    1. Stackblitz ใช้ชื่อ Repository เป็น รหัสนักศึกษาproject เช่น 123456789project
    2. Firebase Hosting ให้ส่ง URL มายัง thapanapong.r@cmu.ac.th
    3. Local Hosting ให้ส่งไฟล์ ZIP มายัง thapanapong.r@cmu.ac.th