ฝกอร/61
21 กรกฎาคม 2560

เลขที่.....
วันที่ 25 ก.ค. 2560
เวลา 09.52 น.

กผส. เลขที่ 1209
เลขที่ 562 วันที่ 26/7/60
วันที่ 21 ก.ค. 2560 เวลา 08:45
เวลา 08.55 น.

รายงานสรุปการฝึกอบรม/สัมมนาภายนอก ประจำปี 2560

เรียน รผบ. ผ่าน ผชก. (นายสุชินฯ) ผอ.ฝทบ. รผก. (CIO) ผชก. (นายสาโรจน์ฯ) ผอ.ฝทท. ผอ.กผส. หน.มพ.

ฝทบ. เลขที่ 4063 วันที่ 31 ก.ค. 60 เวลา 13:0

ส่วนที่ 1 ข้อมูลทั่วไป (สำหรับผู้เข้ารับการฝึกอบรม)

1. ข้าพเจ้า นางสาวธนากานต์ ส่งสุข ตำแหน่ง พนักงานบริหารระบบคอมพิวเตอร์ 4 และนางสาวอมลรดา ด้วงแพง ตำแหน่ง พนักงานบริหารระบบคอมพิวเตอร์ 4 สังกัด แผนก ความมั่นคงปลอดภัยและพัฒนาสารสนเทศ กอง แผนงานและพัฒนาระบบสารสนเทศ ฝ่าย เทคโนโลยีสารสนเทศ ได้รับอนุมัติให้เข้ารับการฝึกอบรม/สัมมนา หลักสูตร Web Application Security จัดโดย บริษัท มายด์เทอร่า จำกัด ระหว่างวันที่ 22 – 23 มิถุนายน 2560 สถานที่จัด อาคาร BizTown ลาดพร้าว ค่าลงทะเบียนอบรม/สัมมนา ☑ เสียค่าใช้จ่าย 27,000 บาท ☐ ไม่เสียค่าใช้จ่าย

กกร.
เลขที่ 1797
วันที่ 31 ก.ค. 2560
เวลา 14.40 น.

2. ข้าพเจ้าขอรายงานสรุปการฝึกอบรม/สัมมนา ดังนี้

2.1 สรุปรายละเอียดเนื้อหาของหลักสูตร

สรุปเนื้อหาของหลักสูตร Web Application Security ได้ดังนี้

2.1.1 OWASP คืออะไร

Open Web Application Security Project (OWASP) เป็นองค์กรไม่แสวงหาผลกำไร (Non-profit organization) ถูกจัดตั้งขึ้นที่ประเทศสหรัฐอเมริกาเมื่อวันที่ 21 เมษายน 2004 โดยมีจุดประสงค์เพื่อเป็นองค์กรสากลที่เป็นศูนย์รวมในการร่วมมือจากนักพัฒนาเว็บแอปพลิเคชันทั่วโลกในการสร้างเว็บแอปพลิเคชันให้มีความปลอดภัย โดย OWASP ได้รับการสนับสนุนจากบริษัท IT ชั้นนำทั่วโลกในการจัดสัมมนาและการจัดอบรมเกี่ยวกับความปลอดภัยเว็บแอปพลิเคชัน อีกทั้งยังมีเว็บไซต์ที่ใช้ในการเก็บรวบรวมและเผยแพร่ความรู้เกี่ยวกับช่องโหว่ที่พบได้บ่อยและวิธีการป้องกัน

2.1.2 OWASP TOP 10 2013

ในปี 2010 OWASP ได้ทำการจัดอันดับช่องโหว่ที่มีความรุนแรงและพบเจอได้บ่อยในเว็บแอปพลิเคชัน 10 อันดับขึ้นมาเป็นครั้งแรก และได้รับกระแสการตอบรับอย่างดีจากนักพัฒนาเว็บแอปพลิเคชันทั่วโลก โดย OWASP TOP 10 2010 ได้รับการยึดถือเป็นมาตรฐานการตรวจสอบช่องโหว่ของเว็บแอปพลิเคชันในหลายองค์กรทั่วโลก และในปี 2013 OWASP ได้เผยแพร่เอกสาร OWASP TOP 10 2013 ที่อธิบายรายละเอียดช่องโหว่ที่พบได้บ่อยและมีความรุนแรง 10 อันดับแรก โดยมีรายละเอียด ดังนี้

เพื่อลงนาม

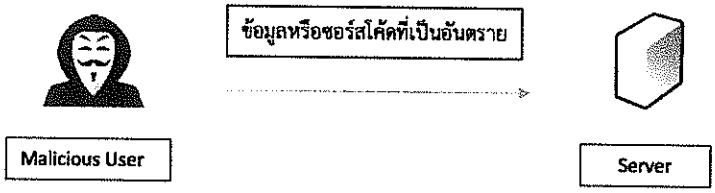
1) Injection

จุดเริ่มต้นของ Injection

ในสมัยยุคที่มีการกำเนิด Internet และ World Wide Web เว็บไซต์ถูกออกแบบให้เป็นแบบ static นั่นคือสามารถแสดงผลข้อมูลแต่ไม่สามารถรับข้อมูลจากผู้ใช้งานได้ เว็บไซต์โดยส่วนมากถูกสร้างขึ้นเพื่อใช้ในการแชร์ข้อมูลให้กับผู้ใช้งานอินเทอร์เน็ตเพียงเท่านั้น เมื่ออินเทอร์เน็ตและเว็บเทคโนโลยีได้รับความนิยมและพัฒนามากขึ้น ทำให้เกิดสิ่งที่เรียกว่า Dynamic Web Application หรือเว็บแอปพลิเคชันที่สามารถมีปฏิสัมพันธ์กับผู้ใช้งานได้ เช่น การรับข้อมูลจากผู้ใช้งาน การแสดงผลข้อมูลที่เปลี่ยนแปลงไปตามลักษณะของการใช้งานและผู้ใช้งาน การพัฒนาการของ Dynamic Web Application นับเป็นก้าวสำคัญของเว็บเทคโนโลยีเนื่องจากเว็บแอปพลิเคชันต้องมีการรับข้อมูลและคำสั่งจากผู้ใช้งาน ซึ่งช่องทางที่เว็บแอปพลิเคชันรับข้อมูลจากผู้ใช้งานก่อให้เกิดช่องทางในการโจมตีชนิด Injection ขึ้นมาในภายหลัง

Injection คือ

Injection คือ ช่องโหว่ที่ผู้บุกรุกสามารถแทรกซอร์สโค้ด หรือข้อมูลที่เป็นอันตรายไปให้เว็บแอปพลิเคชันประมวลผลหรือจัดเก็บ เพื่อหวังผลในการ โจรกรรม ลบ แก้ไข หรือ หยุดการทำงานของเว็บแอปพลิเคชัน โดยการโจมตีประเภทนี้ถูกจัดเป็นการโจมตีที่สามารถส่งผลกระทบร้ายแรงต่อเว็บแอปพลิเคชัน เซิร์ฟเวอร์ และฐานข้อมูลได้



ซึ่งการโจมตีประเภท Injection ที่พบมากเป็นอันดับต้น ๆ จนถูกจัดให้เป็นช่องโหว่อันดับ 1 ใน OWASP TOP 10 ทั้งในปี 2010 และ 2013 ก็คือ SQL Injection (SQLi)

รู้จักกับ SQL

ในปัจจุบัน “ฐานข้อมูล” นับว่าเป็นส่วนประกอบสำคัญในเว็บแอปพลิเคชัน เนื่องด้วยฐานข้อมูลเป็นแหล่งที่ใช้ในการเก็บข้อมูลของเว็บแอปพลิเคชัน ไม่ว่าจะเป็นข้อมูลผู้ใช้งาน ข้อมูลกระดานสนทนา หรือข้อมูลอื่น ๆ ที่แตกต่างกันไปตามชนิดของเว็บ เรียกได้ว่าการทำงานระหว่างเว็บแอปพลิเคชันและฐานข้อมูลนั้นมีความสัมพันธ์กัน โดยปกติเมื่อเว็บแอปพลิเคชันต้องการเรียกดู แก้ไข เพิ่ม หรือลบข้อมูลในฐานข้อมูลก็จะส่งสิ่งที่เรียกว่า SQL Statement หรือเรียกอีกชื่อ SQL query ไปให้กับฐานข้อมูล ดังนั้นเราสามารถเรียกได้ว่า SQL เป็นภาษาที่ใช้ในการสั่งการฐานข้อมูล

<u>สาเหตุของ SQL Injection</u>

สาเหตุหลักของช่องโหว่ประเภท SQL Injection นั้นเกิดจากการที่เว็บแอปพลิเคชันนำข้อมูลที่ได้รับมาจากผู้ใช้งานมาสร้าง SQL query โดยไม่ได้ทำการตรวจสอบก่อนว่าข้อมูลนั้นมีความถูกต้องเหมาะสมหรือไม่ ดังนั้นวิธีการป้องกัน SQL Injection ต้องเริ่มต้นจากการตรวจสอบข้อมูลที่ได้รับมาจากผู้ใช้งาน

<u>แนวทางในการป้องกัน SQL Injection</u>

วิธีการป้องกัน SQL Injection นั้นประกอบไปด้วย 3 วิธี ผู้พัฒนาระบบจะเลือกใช้วิธีใดวิธีหนึ่งหรือหลายวิธีในการป้องกันก็ได้ เพื่อความปลอดภัยที่มากขึ้นของเว็บแอปพลิเคชัน โดยวิธีป้องกัน SQL Injection นั้นประกอบไปด้วย

- Prepared Statements หรืออีกชื่อหนึ่งคือ Parameterized Query เป็นหนึ่งในวิธีการที่ได้รับความนิยมอย่างมากในการป้องกัน SQL Injection เนื่องด้วยความง่ายในการใช้งาน และเกือบทุกภาษาที่ใช้ในการเขียนเว็บแอปพลิเคชันในปัจจุบันมี library ให้นักพัฒนาสามารถเรียกใช้งานได้อย่างสะดวก

<u>ตัวอย่าง PHP prepared statements</u>

```
?php
        $mysqli = new mysqli("localhost", "username", "password",
"database_name");
        $stmt = $mysqli->prepare("SELECT * FROM users WHERE username =
? AND password = ?");

        $stmt->bind_param("ss", $username,$password);    // [illegible]

        $username = $_POST['user_username'];
        $password = $_POST['user_password'];

        $stmt->execute();

        if($stmt->fetch()){
                echo "Login Success";
        }else{
                echo "Login Fail";
        }

?>
```

จะเห็นได้ว่าการใช้ Prepared Statements เป็นการระบุให้กับเว็บแอปพลิเคชันรู้ก่อนว่า ข้อมูลที่ได้รับจากผู้ใช้งานนั้นเป็น string ทั้งหมดและไม่ให้นำข้อมูลที่เป็น string เหล่านี้ไปประมวลผลในรูปแบบของ SQL Statement ดังนั้นโปรแกรมนี้จึงไม่มีช่องโหว่ประเภท SQL Injection ซึ่งไม่เพียงแต่ภาษา PHP เท่านั้นที่รองรับการทำ Prepared Statements ภาษาอื่น ๆ ที่ใช้ในการเขียนเว็บเช่น .NET หรือ Java ก็มี Prepared Statements ให้เรียกใช้งาน

- Stored Procedure เป็นอีกหนึ่งวิธีการที่ใช้ในการป้องกัน SQL Injection โดย Stored Procedure จะเก็บ SQL query ไว้ที่ฐานข้อมูลและให้เว็บแอปพลิเคชันส่ง parameter มาให้ฐานข้อมูลเรียก SQL query ที่เก็บไว้มาทำงาน แทนที่เว็บแอปพลิเคชันจะเป็นผู้สร้าง SQL query แล้วส่งไปให้กับฐานข้อมูลแต่ Stored Procedure เป็นวิธีการที่ไม่ค่อยได้รับความนิยมเท่ากับ Prepared Statements เนื่องจากต้องมีการสร้าง SQL query เก็บไว้ในฐานข้อมูลก่อนเรียกใช้ ทำให้การแก้ไข SQL query มีความยุ่งยากมากกว่า อย่างไรก็ตามฐานข้อมูลส่วนใหญ่ (MySQL, Oracle, DB2) ต่างก็รองรับการทำงานของ Stored Procedure

- Escaping User Input เป็นวิธีการที่ใช้ในการป้องกัน SQL Injection ด้วยการตรวจสอบข้อมูลที่ได้รับมาจากผู้ใช้งาน แก้ไข หรือ ลบ ข้อมูลที่อาจจะเป็นอันตรายออกก่อนที่จะนำไปสร้างเป็น SQL query หากเราลองสังเกตรูปแบบการโจมตีด้วย SQL Injection อักขระพิเศษ เช่น ‘ (single quote) หรือ ” (double quote) คือตัวการสำคัญในการพยายามดัดแปลง SQL query (นอกจากนี้ยังมี # — และ keyword อื่น ๆ เช่น AND OR) ดังนั้นหากเราสามารถที่จะกรองเครื่องหมายเหล่านี้ออกไปก่อนที่จะนำข้อมูลจากผู้ใช้งานไปสร้างเป็น SQL query ก็จะช่วยป้องกัน SQL Injection ได้มากขึ้น อย่างไรก็ตาม Escaping User Input นั้นไม่มีประสิทธิภาพเมื่อเทียบกับ Prepared Statements และ Stored Procedure เนื่องจากข้อมูลที่ได้รับมาจากผู้ใช้งานนั้นมีรูปแบบที่หลากหลายและมีการเปลี่ยนแปลงพัฒนาไปตามเวอร์ชันของฐานข้อมูล ทำให้การกรองข้อมูลผู้ใช้งานอาจจะไม่ครบถ้วนถูกต้องสมบูรณ์ ดังนั้น Escaping User Input ควรถูกเลือกเป็นตัวเลือกสุดท้ายในการป้องกัน SQL Injection และหากจำเป็นต้องใช้ Escaping User Input ก็ควรเลือกใช้ Library ที่น่าเชื่อถือ เช่น ESAPI จาก OWASP เป็นต้น

<u>ตัวอย่างการตรวจสอบเว็บแอปพลิเคชันที่มีช่องโหว่ SQL Injection</u>

- ทดสอบการทำ Authentication หรือการ Login เข้าระบบ โดยการยืนยันตัวตนด้วยการทำเงื่อนไขใน SQL Statement (Where Clause) ให้เป็นจริงเสมอ ทดลองใส่ **’ or ‘1’ = ‘1** ในช่องกรอก Username และ Password โดย SQL Statement จะเป็นดังนี้ SELECT * FROM account WHERE username = ‘’ **or ‘1’ = ‘1’** ซึ่งอักษรสีแดงคือสิ่งที่ได้ทดลองป้อนเข้าไป โดย ‘1’ = ‘1’ คือเงื่อนไขที่เป็นจริง อ้างอิงตารางตรรกะ OR

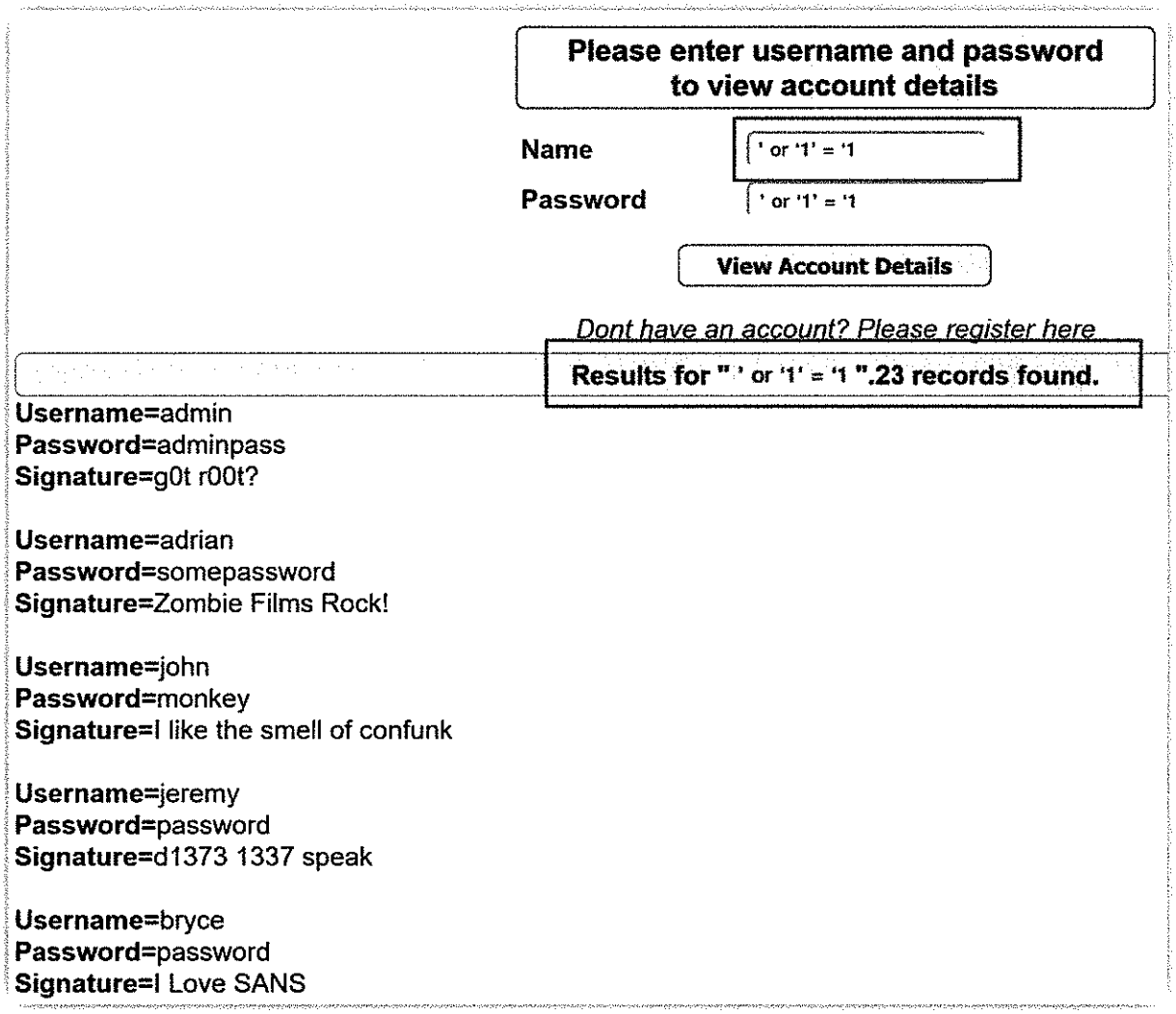
T or T = True

T or F = True

F or T = True

F or F = False

จะเห็นว่า เพียงมีตัวแปรด้านใดด้านหนึ่งเป็นจริงก็จะทำให้เงื่อนไขเป็นจริง ซึ่งในส่วนที่ของ '1' = '1' เป็นจริง จึงทำให้อีกส่วนที่เหลือเป็นอะไรก็ได้ ในที่นี้ให้เป็น ''



เมื่อหลีกเลี่ยงการยืนยันตัวตนได้ จะมีการแสดงข้อมูลชื่อผู้ใช้งาน และรหัสผ่านทั้งหมดในระบบ ดังภาพ

2) Broken Authentication and Session Management

ช่องโหว่ Broken Authentication and Session Management (BASM) เป็นช่องโหว่ประเภทที่ถูกพบเจอได้บ่อยจากเว็บแอปพลิเคชันที่มีการทำ User Authentication (user login)

<u>Authentication คือ</u>

Authentication คือ กระบวนการที่เว็บแอปพลิเคชันใช้ในการยืนยันตัวบุคคลก่อนเข้าใช้งาน ซึ่งปัจจุบันเว็บแอปพลิเคชันส่วนใหญ่เลือกใช้ Username และ Password ในการยืนยันตัวบุคคลเพื่อเข้าใช้งานโปรแกรม ตัวอย่างเช่น ผู้ใช้งานต้องทำการยืนยันตัวบุคคลโดยใส่ Username และ Password ก่อนเข้าอ่าน email ในกล่องข้อความ เป็นต้น

Session management คือ

Session management เป็นการจัดการกับสถานะของผู้ใช้งาน เนื่องจาก HTTP Protocol เป็น Stateless Protocol การแยกแยะว่า HTTP Request แต่ละอันมาจากผู้ใช้งานคนไหนจึงใช้ SESSION ID เป็นหลัก การจัดการกับ SESSION ของผู้ใช้งานจึงเป็นสิ่งสำคัญเนื่องจาก SESSION ID ควรถูกเก็บเป็นความลับและรู้กันเฉพาะเจ้าของ SESSION และ เว็บแอปพลิเคชันเท่านั้น

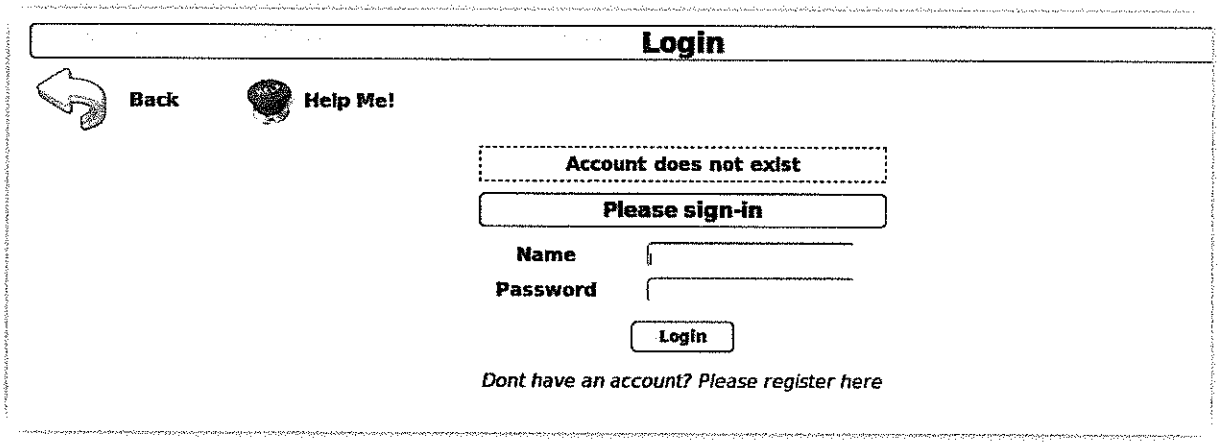
Broken Authentication and Session Management (BASM) คือ

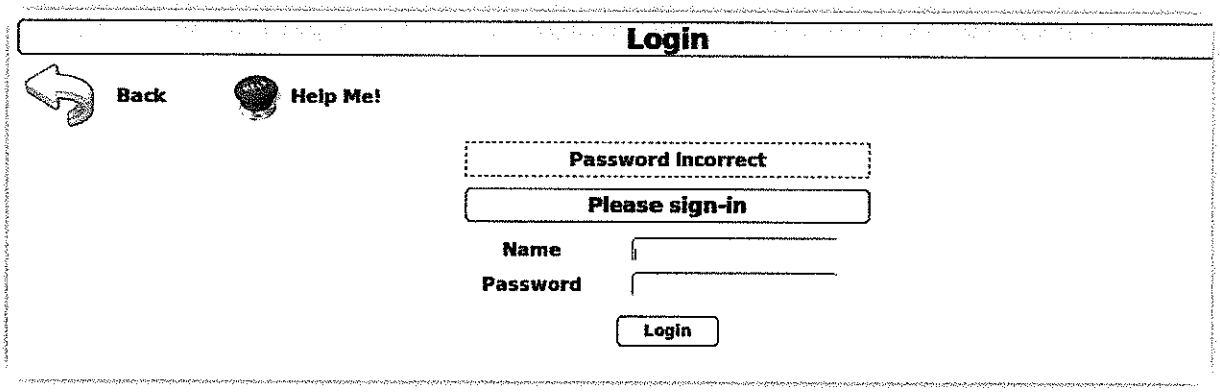
BASM เป็นช่องโหว่ที่เกี่ยวกับการจัดการยืนยันตัวบุคคลและการจัดการ SESSION ที่ผิดพลาด ซึ่งสามารถพบได้บ่อย แม้ว่าการยืนยันตัวบุคคลจะเป็นสิ่งที่สามารถพบได้ในเกือบทุกเว็บไซต์ การพัฒนาระบบที่มีระบบการยืนยันตัวบุคคลที่มีความปลอดภัยนั้นไม่ง่าย ช่องโหว่ที่ถูกจัดอยู่ในหมวดนี้ ได้แก่

- ช่องโหว่ที่อนุญาตให้ผู้ใช้งานสามารถปลอมเป็นผู้ใช้งานคนอื่นได้ ตัวอย่างเช่น หากเว็บแอปพลิเคชันปล่อยให้ผู้ใช้งานสามารถใช้ Username เดียวกันได้มีโอกาสเกิดช่องโหว่ประเภทนี้
- เว็บแอปพลิเคชันไม่ได้จำกัดจำนวนครั้งในการ Login ที่ผิดพลาด หลายเว็บแอปพลิเคชันอนุญาตให้ผู้ใช้งานทำการ Login fail กี่ครั้งก็ได้ ซึ่งเป็นภัยร้ายแรงต่อระบบอย่างมาก เนื่องจากผู้ไม่หวังดีสามารถใช้เทคนิคเครื่องมือในการทำ Brute force เพื่อเดาสุ่ม Password ได้
- เว็บแอปพลิเคชันเก็บ Password ในฐานข้อมูลเป็น plain-text การเก็บ Password เป็น plain-text นั้นเป็นอันตรายต่อผู้ใช้งานอย่างมากเนื่องจากหากข้อมูลในฐานข้อมูลมีการรั่วไหล ผู้ที่รู้ Password ก็จะสามารถใช้งาน User แอปพลิเคชัน ที่ Password รั่วไหลได้ทันที ดังนั้น Password ควรถูกเก็บอยู่ในรูปของ Hash
- เว็บไซต์ที่ไม่ใช่ HTTPS เปิดโอกาสให้ผู้ไม่หวังดีดักจับ SESSION ID ของผู้ใช้งาน อย่างที่ทราบกันว่า HTTP เป็น Protocol ที่ไม่ปลอดภัยเนื่องจากไม่มีการเข้ารหัสข้อมูล ดังนั้น ใครก็ตามที่อยู่ในเส้นทางที่ผู้ใช้งานส่งไปหาเว็บเซิร์ฟเวอร์ก็สามารถดักจับ SESSION ID ได้ Google ได้ตระหนักถึงความไม่ปลอดภัยของ HTTP และมีการออกมาตรการกระตุ้นให้ผู้พัฒนาเว็บไซต์เปลี่ยนมาใช้ HTTPS โดยการแสดงผลเว็บไซต์ที่ใช้ HTTPS ก่อนในผลการค้นหา
- การทำงานของ Forget Password ไม่ปลอดภัย Forget Password เป็นอีกช่องทางที่ผู้ไม่หวังดีสามารถใช้ในการสืบค้น Password ของเป้าหมายได้ ในบางแอปพลิเคชัน Forget Password ถามคำถามเกี่ยวกับผู้ใช้งานด้วยคำถามง่าย ๆ เช่น เกิดวันที่เท่าไร เกิดเมืองอะไร หากตอบถูกก็จะเปิดเผย Password ให้กับผู้ตอบคำถาม ซึ่งข้อมูลเหล่านี้ไม่เป็นความลับและนำไปสู่การโจรกรรม password ได้
- ช่องโหว่ SESSION Fixation ที่ผู้ไม่หวังดีหลอกให้เป้าหมายใช้ SESSION ID ของตนในการ Login เพื่อที่จะได้สิทธิในการเข้าใช้งานของผู้ใช้งานคนนั้น

ตัวอย่างเว็บแอปพลิเคชันที่มีช่องโหว่ Broken Authentication and Session Management

- ทดลองใส่ Username และ Password แล้วกด Login จะสังเกตเห็นว่าระบบมีการแจ้งเตือนมาว่า “Account does not exist” ซึ่งแปลว่า ไม่มี Username ที่กรอกลงไปในระบบ หรือ “Password Incorrect” ซึ่งแปลว่า มี Username นี้ในระบบ แต่ Password ไม่ถูกต้อง





การโจมตีรูปแบบนี้เรียกว่า Username enumeration โดยเทคนิคนี้จะช่วยให้สามารถเก็บรวบรวม Username ที่มีในระบบฐานข้อมูลได้จำนวนหนึ่ง ซึ่งจะช่วยให้การ Brute force รวดเร็วยิ่งขึ้น

3) Cross Site Scripting (XXS)

Cross-site Scripting (XSS) เป็นช่องโหว่ที่เป็นอันตรายต่อผู้ใช้งานเว็บแอปพลิเคชัน เพราะเป็นช่องทางให้ผู้ไม่หวังดีสามารถขโมยข้อมูลที่เป็นความลับด้วยการใส่ JavaScript ลงไปในเว็บแอปพลิเคชัน

สาเหตุของ Cross-site Scripting (XSS)

การโจมตีและช่องโหว่ประเภท Cross-site Scripting (XSS) มี 3 ประเภท ได้แก่ Reflected, Stored, และ DOM-based XSS ซึ่งเกิดจากการที่ผู้พัฒนาเว็บแอปพลิเคชันไม่ได้ทำการ

ตรวจสอบข้อมูลที่ได้รับมาจากผู้ใช้งานก่อนนำไปแสดงผลให้กับผู้ใช้งานคนอื่น ดังนั้น หากต้องการป้องกันการเกิดช่องโหว่ประเภทนี้ ต้องเริ่มต้นที่การตรวจสอบและคัดกรองข้อมูลที่ได้รับมาจากผู้ใช้งาน

<u>แนวทางในการป้องกัน XSS</u>

- วิธีการป้องกันช่องโหว่ประเภท XSS ที่ดีที่สุดคือการไม่นำข้อมูลที่ได้รับมาจากผู้ใช้งานมาแสดงผลใน tag ที่เปิดโอกาสให้เรียกใช้งาน JavaScript

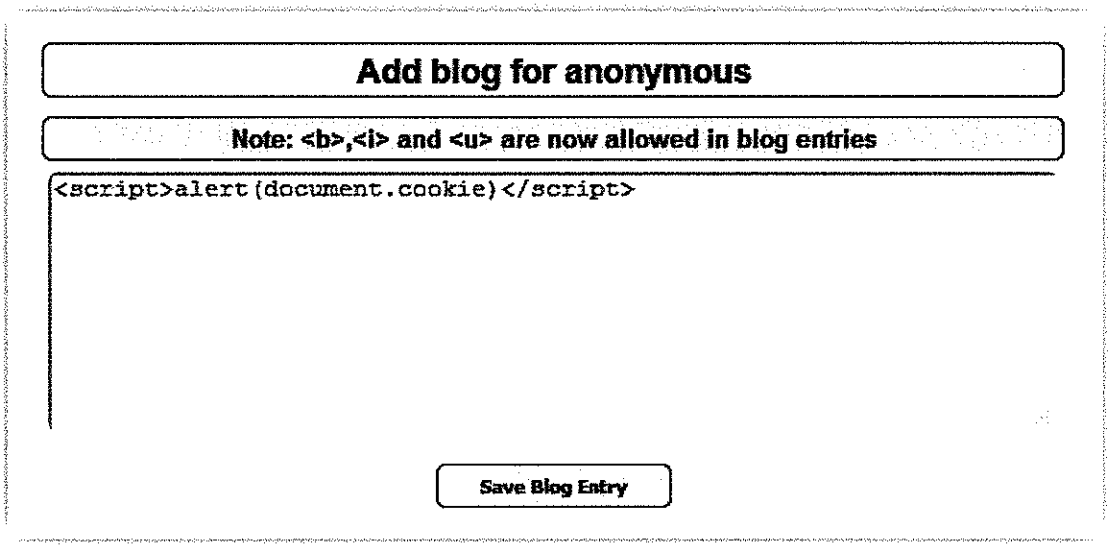
```
<script>...NEVER PUT UNTRUSTED DATA HERE...</script>   in script tag

<!--...NEVER PUT UNTRUSTED DATA HERE...-->  in comment tag

<div ...NEVER PUT UNTRUSTED DATA HERE...=test /> as attribute

<NEVER PUT UNTRUSTED DATA HERE... href="test" />  as html tag

<style>...NEVER PUT UNTRUSTED DATA HERE...</style> in style tag
```

- บางเว็บแอปพลิเคชันการไม่แสดงผลข้อมูลที่ได้รับมาจากผู้ใช้งานนั้นเป็นไปไม่ได้ (ตัวอย่างเช่น web board หรือ forum จำเป็นต้องนำข้อมูลที่ได้รับมาจากผู้ใช้งานมาแสดงผล) ดังนั้น การป้องกัน XSS มีเพียงวิธีเดียวคือการคัดกรองข้อมูลก่อนนำไปแสดงผล เพื่อให้มั่นใจว่าข้อมูลที่ได้รับมาไม่มี JavaScript ฝังอยู่ เราเรียกวิธีการนี้ว่า Data Sanitizing (การตัด keyword หรือ tag ออก) และ HTML Escaping (การ encode อักขระพิเศษให้อยู่ใน format ของ HTML encode)

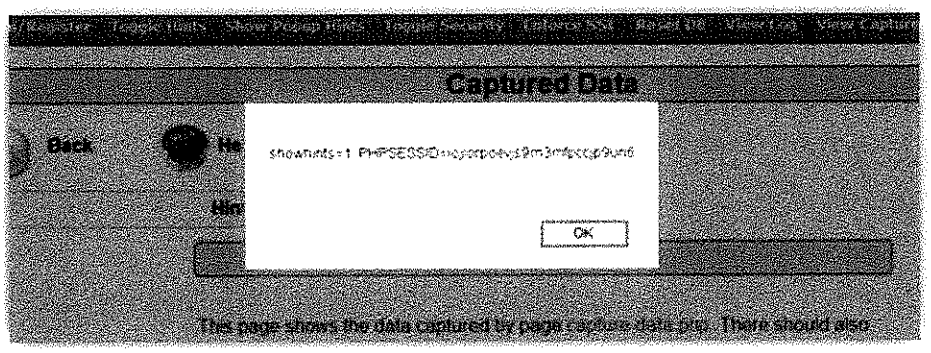
- หากต้องการใส่ข้อมูลลงใน Tag ต้องทำ JavaScript, Attribute, และ CSS Escaping เพื่อให้แน่ใจว่าข้อมูลที่ได้รับจากผู้ใช้งานจะไม่สามารถเรียกใช้งาน JavaScript ได้

<u>ตัวอย่างเว็บแอปพลิเคชันที่มีช่องโหว่ Cross Site Scripting</u>

ทดสอบช่องโหว่ Cross Site Scripting เพื่อดู Cookie ที่ช่องกรอกข้อความ โดยทดลองใส่สคริปต์ ดังนี้ <script>alert(document.cookie)</script> ซึ่งเป็นสคริปต์ที่ใช้เรียกดู Session ID ที่ใช้งานอยู่ในขณะนั้น



ผลลัพธ์ที่ได้คือ Session ID ที่ใช้งานในขณะนั้น จึงถือได้ว่าเว็บแอปพลิเคชันดังกล่าวมีช่องโหว่ Cross Site Scripting



4) Insecure Direct Object References (IDOR)

Insecure Direct Object References (IDOR) เป็นช่องโหว่ที่เกิดจากความผิดพลาดในการทำ Authorization (การตรวจสอบสิทธิในการเข้าถึง) นั่นก็คือ หากเว็บแอปพลิเคชันไหนปล่อยให้ผู้ใช้งานสามารถเข้าถึงข้อมูล หรือ ไฟล์ต่าง ๆ ที่ไม่มีสิทธิในการเข้าถึงได้โดยตรง ก็เรียกได้ว่ามีช่องโหว่ประเภท IDOR อยู่

<u>แนวทางในการป้องกัน Insecure Direct Object Reference</u>

IDOR เกิดขึ้นจากความผิดพลาดในการทำ Authorization ดังนั้นวิธีการป้องกัน IDOR ที่ดีที่สุดคือการตรวจสอบขั้นตอนการทำ Authorization โดยเฉพาะการเข้าถึงข้อมูลที่เป็นความลับ การตรวจสอบการทำ Authorization อาจจะไม่มีแนวทางในการปฏิบัติที่ชัดเจน แต่มีขั้นตอนในการป้องกันเบื้องต้นดังนี้

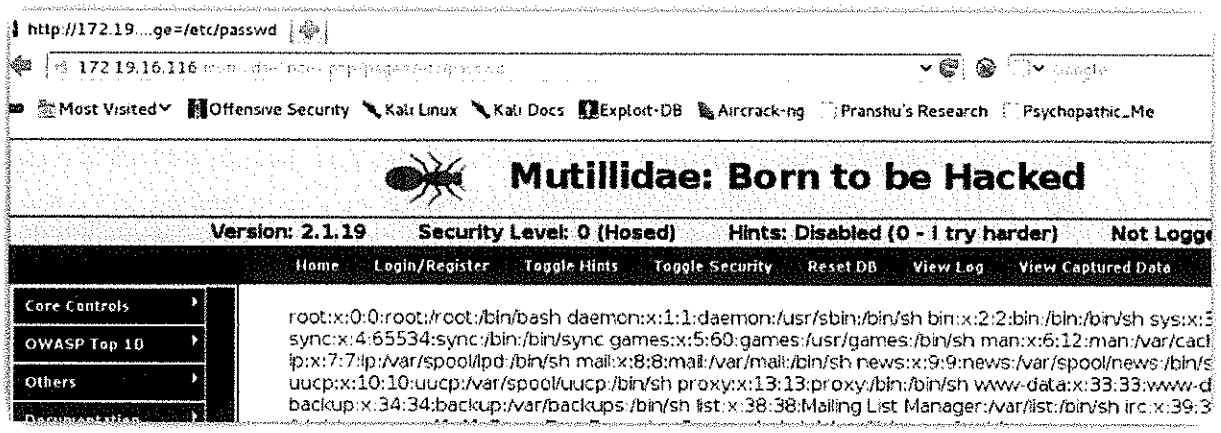
- ตรวจสอบการทำ Authorization ใน Web Page ที่มีการรับข้อมูลจากผู้ใช้งาน โดยทำการตรวจสอบข้อมูลประเภท User ID หรือ Object ID ที่สามารถแก้ไขโดยการตรวจสอบนี้ไม่เฉพาะแค่การรับข้อมูลจาก URL นักพัฒนาเว็บแอปพลิเคชันต้องตรวจสอบข้อมูลที่ได้รับมาชนิด POST ด้วย
- Disable directory listing โดยปกติ Web Application Software เช่น Apache จะมีฟังก์ชัน Directory listing นั่นคือ การผู้ใช้งานทำการเข้าถึง Path ที่เป็น Folder ไฟล์ที่อยู่ใน Folder จะถูกแสดงออกมาทั้งหมด ฟังก์ชันนี้ควรถูกปิดเนื่องจากเปิดโอกาสให้ผู้ไม่หวังดีสามารถเห็นรายชื่อของไฟล์ใน Folder ซึ่งนำไปสู่การเข้าถึงไฟล์โดยไม่ได้รับอนุญาต
- ทำให้ User ID หรือ Object ID ยากแก่การเดาสุ่ม การทำชื่อไฟล์หรือ User ID ให้ยากแก่การเดาสุ่มนั้น จะช่วยป้องกันช่องโหว่ประเภท IDOR ได้ระดับนึง เนื่องจากผู้ไม่หวังดีไม่อาจจะคาดเดาชื่อไฟล์หรือ user id ได้ถูกต้องได้ง่ายนัก อย่างไรก็ตาม วิธีนี้เป็นเพียงแค่ลดความเสี่ยงลง แต่ไม่ได้แก้ไขปัญหาที่ต้นเหตุ

สรุปช่องโหว่ Insecure Direct Object Reference

ช่องโหว่ประเภท Insecure Direct Object Reference เป็นช่องโหว่ที่ปล่อยให้ผู้ไม่หวังดีสามารถเข้าถึงหรือกระทำการกับข้อมูลผ่านทางการแก้ไข Parameter URL หรือ Parameter ในการส่งข้อมูลประเภท POST ซึ่งสามารถทำให้ข้อมูลผู้ใช้งานรั่วไหล หรือในกรณีรุนแรงส่งผลให้ข้อมูลเสียหาย การป้องกันช่องโหว่ประเภทนี้ การทำ Authorization เป็นส่วนสำคัญ เพราะหากมีการตรวจสอบสิทธิในการเข้าถึงหรือการกระทำกับข้อมูลอย่างถูกต้องแล้ว ช่องโหว่ประเภท IDOR ก็จะไม่มีอยู่ในเว็บแอปพลิเคชัน

ตัวอย่างการทดสอบเว็บแอปพลิเคชันที่มีช่องโหว่ Insecure Direct Object References

- ทดสอบด้วยการเรียกไฟล์ที่อยู่ในเครื่องแม่ข่ายด้วยการใส่ /ect/passwd ไปที่พารามิเตอร์ หากไฟล์นั้นอยู่ในเครื่องแม่ข่าย ที่หน้าเว็บเพจก็จะแสดงข้อมูลที่อยู่ในไฟล์นั้น ๆ ออกมา เช่น ไฟล์ที่เรียก คือ /ect/passwd ผลลัพธ์คือ ไฟล์ที่ใช้เก็บรายละเอียดของแต่ละบัญชีที่ถูกใช้งานอยู่บนเครื่องแม่ข่ายเครื่องนั้น



5) Security Misconfiguration

Security Misconfiguration เป็นช่องโหว่ที่เกิดจากการตั้งค่าที่ผิดพลาดของระบบ เนื่องจากเว็บแอปพลิเคชันประกอบไปด้วยหลายส่วนประกอบ ทุกส่วนต้องได้รับการดูแลและตั้งค่าอย่างถูกต้องมิฉะนั้นก็จะกลายเป็นช่องโหว่ที่นำความเสียหายมาสู่ระบบทั้งระบบ

สาเหตุที่ทำให้เกิดช่องโหว่ Security Misconfiguration

- Web application ช่องโหว่สามารถเกิดขึ้นได้จากการตั้งค่าที่ผิดพลาดของเว็บแอปพลิเคชัน เช่น การกำหนดสิทธิในการเข้าถึงแต่ละส่วนอย่างผิดพลาด ช่องโหว่ประเภทนี้ไม่ได้รวมถึงการทำ authorization ที่ผิดพลาด แต่หมายถึงการตั้งค่าสิทธิ หรือ การตั้งค่าอื่น ๆ ที่ผิดพลาด นอกจากนี้หากเว็บแอปพลิเคชันมีการใช้ Content Management System (Joomla, WordPress, Dupal) เป็นโครงสร้างและไม่ได้ทำการ update security patch ให้กับ CMS เหล่านี้ก็ถูกจัดให้เป็นช่อง

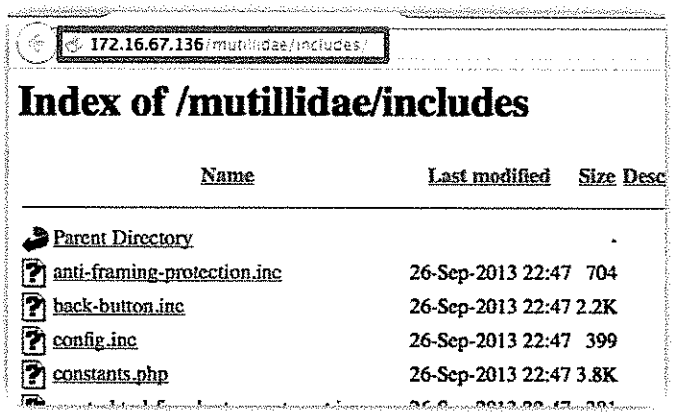
โหว่ในประเภท Security Misconfiguration โดยผลกระทบของช่องโหว่ก็แตกต่างกันไป เช่น การตั้งค่าที่ผิดพลาดอาจจะนำไปสู่การเปิดเผยข้อมูลความลับให้กับผู้ใช้งานที่ไม่สมควร การอนุญาตให้ผู้ใช้งานลบข้อมูลสำคัญบางอย่างของระบบเป็นต้น

- Web application framework เช่น Hibernate, CakePHP, หรือ Django ได้รับความนิยมในการนำมาช่วยในการพัฒนาเว็บแอปพลิเคชันมากขึ้น เนื่องจากช่วยให้ผู้พัฒนาสามารถพัฒนาเว็บแอปพลิเคชันได้รวดเร็วและเป็นระเบียบขึ้น อย่างไรก็ตาม เนื่องจาก Web application framework ก็เป็น software ตัวหนึ่งซึ่งสามารถมีช่องโหว่ได้เช่นกันซึ่งหากไม่ได้รับการ Update security patch เป็นเวลานานก็สามารถเป็นช่องทางในการโจมตีสำหรับผู้ไม่หวังดีได้

- Web server software เช่น Apache, WebSphere, NodeJS เป็นส่วนประกอบของการทำงานของเว็บแอปพลิเคชัน หน้าที่ของ Web server software คือการรับ HTTP request มาประมวลผล ดังนั้น เรียกได้ว่า Web server software เป็นส่วนประกอบที่มีการรับข้อมูลจากผู้ใช้งานโดยตรง Configuration การทำงานของ Interpreter และ Compiler ที่ใช้ในเว็บแอปพลิเคชัน รวมถึงการตั้งค่าอื่น ๆ เกี่ยวกับการจัดการ HTTP request และ HTTP response ที่ Web server

ตัวอย่างการทดสอบเว็บแอปพลิเคชันที่มีช่องโหว่ Security Misconfiguration

ทดสอบโดยการเข้าถึงไดเรคทอรีที่สำคัญ ๆ เช่น Classes Includes เป็นต้น ผลลัพธ์ที่ได้ คือ จะสามารถเข้าถึงข้อมูลที่อยู่ภายในไดเรคทอรีนั้น ๆ ได้



6) Sensitive Data Exposure

Sensitive Data Exposure เป็นช่องโหว่ที่เกิดขึ้นกับข้อมูลเป็นหลัก ช่องโหว่ประเภทนี้เกิดได้จากการที่เว็บแอปพลิเคชันมีช่องโหว่ที่อนุญาตให้ผู้ไม่หวังดีเข้าถึงข้อมูลที่เป็นความลับ ซึ่งเกิดได้จากหลายกรณี เช่น การใช้ Encryption Algorithm ที่ไม่แข็งแรง ไม่เข้ารหัสข้อมูลขณะส่ง (ไม่ใช้ HTTPS) และ การเก็บข้อมูล Backup ไม่ปลอดภัย เป็นต้น

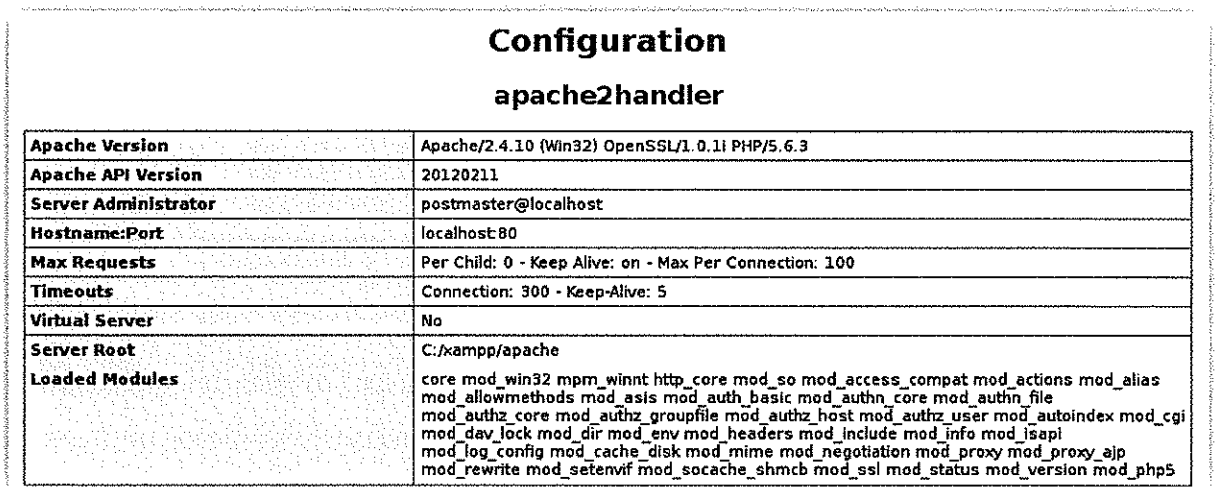
Sensitive Data Exposure (SDE) คือ

Sensitive Data Exposure ถ้าแปลตรงตัวก็คือการเปิดเผยข้อมูลที่มีความอ่อนไหว ซึ่งหมายถึงข้อมูลที่อาจจะส่งผลกระทบต่อผู้ใช้งาน หรือเว็บแอปพลิเคชันหากข้อมูลเกิดการรั่วไหล

ไปสู่บุคคลไม่พึงประสงค์ ตัวอย่างข้อมูลที่มีความอ่อนไหวได้แก่ ข้อมูลบัตรเครดิต ข้อมูล User แอปพลิเคชัน ข้อมูลผู้ป่วย ข้อมูลคดี ข้อมูลยอดขาย ข้อมูลลูกค้า เป็นต้น จะเห็นได้ว่าข้อมูลที่อ่อนไหวเป็นข้อมูลที่สามารถสร้างความเสียหายต่อเจ้าของเมื่อข้อมูลตกไปอยู่ในมือของผู้ไม่ประสงค์ดี เว็บแอปพลิเคชันในปัจจุบัน เช่น เว็บไซต์ธนาคาร เว็บไซต์ขายของออนไลน์ เว็บไซต์ของโรงพยาบาล อาจมีข้อมูลสำคัญเหล่านี้อยู่ หากข้อมูลเหล่านี้ไม่ได้รับการปกป้องด้วยเทคนิคต่าง ๆ อย่างถูกวิธี ก็เรียกได้ว่าเว็บแอปพลิเคชันนั้นมีช่องโหว่ประเภท Sensitive Data Exposure

ตัวอย่าง Sensitive Data Exposure

- เว็บแอปพลิเคชันที่มีการจัดการ User แอปพลิเคชัน ที่ไม่ได้ใช้ HTTPS เรียกได้ว่ามีช่องโหว่ประเภท SDE อยู่ทั้งนั้น เนื่องจากข้อมูล SESSION ID ของผู้ใช้งานถูกส่งผ่านอินเทอร์เน็ตด้วย plain-text ทำให้ใครสามารถดักจับข้อมูล SESSION ID ของผู้ใช้งานได้
- เว็บแอปพลิเคชันที่ไม่ได้มีการทำ Hashing password อย่างถูกวิธี มีโอกาสเปิดเผยข้อมูล password ของผู้ใช้งานให้กับบุคคลที่ไม่สมควรได้รู้
- เว็บแอปพลิเคชันที่มีการเก็บข้อมูลสำคัญ เช่น Credit card แต่ไม่ได้เข้ารหัส
- เว็บแอปพลิเคชันที่ผู้พัฒนาระบบไม่ได้ทำการปิดบังหรือลบรายละเอียดของไฟล์ที่เรียกใช้งานฟังก์ชัน phpinfo() โดยจะบอกรายละเอียดต่าง ๆ เช่น Apache Version, Service Admin, Hostname หรือ Port ดังภาพด้านล่าง



Configuration

apache2handler

Apache Version	Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.6.3
Apache API Version	20120211
Server Administrator	postmaster@localhost
Hostname:Port	localhost:80
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	No
Server Root	C:/xampp/apache
Loaded Modules	core mod_win32 mpm_winnt http_core mod_so mod_access_compat mod_actions mod_alias mod_allowmethods mod_asis mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_groupfile mod_authz_host mod_authz_user mod_autoindex mod_cgi mod_dav_lock mod_dir mod_env mod_headers mod_include mod_info mod_isapi mod_log_config mod_cache_disk mod_mime mod_negotiation mod_proxy mod_proxy_ajp mod_rewrite mod_setenvif mod_socache_shmcb mod_ssl mod_status mod_version mod_php5

แนวทางในการป้องกัน Sensitive Data Exposure

- เข้ารหัสข้อมูลที่เป็นความลับเสมอ ทั้งข้อมูลที่เก็บอยู่ในฐานข้อมูลและข้อมูลที่ถูกส่งผ่านเครือข่ายคอมพิวเตอร์ ด้วย Strong encryption algorithm

- ใช้ HTTPS แทน HTTP เนื่องจาก HTTPS จะช่วยปกป้องข้อมูลไม่ให้ถูกดักจับระหว่างการส่งผ่านเครือข่ายอินเทอร์เน็ต

- ไม่เก็บข้อมูลสำคัญไว้บนเครื่องเซิร์ฟเวอร์ (ถ้าเป็นไปได้) ข้อมูลบางอย่าง เช่น Credit card ไม่สมควรเก็บไว้ที่เซิร์ฟเวอร์ หากเป็นไปได้ให้ผู้ใช้งานกรอกข้อมูลใหม่ทุกครั้งจะเป็นการช่วยปกป้องข้อมูลความลับของผู้ใช้งานได้ดีกว่า

- ให้สิทธิเท่าที่จำเป็นแก่ผู้ใช้งาน โดยข้อมูลไหนที่เป็นความลับต้องถูกจำกัดให้เข้าถึงได้เฉพาะผู้ที่เกี่ยวข้อง

- แบ่งประเภทของข้อมูลอย่างชัดเจน เช่น ข้อมูล ลับ ลับมาก ลับสุดยอด เป็นต้น เพื่อให้การจัดการข้อมูลเหล่านี้เป็นไปอย่างมีประสิทธิภาพและรัดกุมยิ่งขึ้น

7) Missing Function Level Access Control

Missing Function Level Access Control (MFLAC) เป็นช่องโหว่ที่เกิดจากความผิดพลาดในการทำ Authorization ความผิดพลาดในการตรวจสอบฟังก์ชัน หรือสิทธิต่าง ๆ ในการเชื่อมต่อเข้าสู่ระบบ ทำให้ผู้บุกรุกสามารถทำการ เชื่อมต่อเข้าสู่ระบบ โดยการข้ามขั้นตอนการพิสูจน์ตัวตนได้ เช่น มีการตรวจสอบการ login แต่ไม่ได้ตรวจสอบว่ามีสิทธิในการเข้าใช้งานในส่วนผู้ดูแลระบบ ทำให้ผู้ใช้งานทั่วไปสามารถเข้าไปใช้งานในส่วนของผู้ดูแลระบบได้ เป็นต้น

<u>ความแตกต่างระหว่าง Authentication และ Authorization</u>

Authentication คือ การพิสูนจ์ตัวตน เช่น การ Login เข้าเว็บแอปพลิเคชันด้วย Username และ Password เป็นต้น

Authorization คือ การตรวจสอบสิทธิ เช่น user1 ได้ทำการ Login เข้าเว็บแอปพลิเคชันแล้ว ต้องการเข้าไปแก้ไขข้อมูลส่วนตัวของ user1 ก็สามารถทำได้ แต่หาก user1 ต้องการแก้ข้อมูลของ User2 เว็บแอปพลิเคชันต้องทำการตรวจสอบสิทธิของ user1 ว่าไม่ใช่เจ้าของ user2 แอปพลิเคชันจึงไม่อนุญาตให้ทำการแก้ไขข้อมูล เป็นต้น

จะเห็นได้ว่าการทำ Authentication และ Authorization นั้นแตกต่างกัน การทำ Authentication เป็นเพียงแค่การตรวจสอบว่าคน ๆ นี้คือใคร แต่การทำ Authorization คือการตรวจสอบว่าคน ๆ นี้มีสิทธิกระทำการอย่างที่ร้องขอหรือไม่

<u>สาเหตุของช่องโหว่ MFLAC</u>

- ไม่ได้มีการตรวจสอบประเภทของผู้ใช้งานก่อนอนุญาตให้ใช้งาน ผู้ใช้งานสามารถเข้าใช้งานฟังก์ชันได้ผ่านการเปลี่ยน URL

- ตรวจสอบใช้งานเฉพาะหน้าแสดงผล แต่ไม่ได้ตรวจสอบผู้ใช้งานเมื่อ Submit ข้อมูล ตัวอย่างเช่น user1 อาจจะไม่สามารถเข้าหน้าแก้ไขข้อมูลของผู้ใช้งานคนอื่นได้ แต่ user1 สามารถทำการ submit ข้อมูลแล้วเปลี่ยน user id เพื่อแก้ไขข้อมูลของ user2 ได้

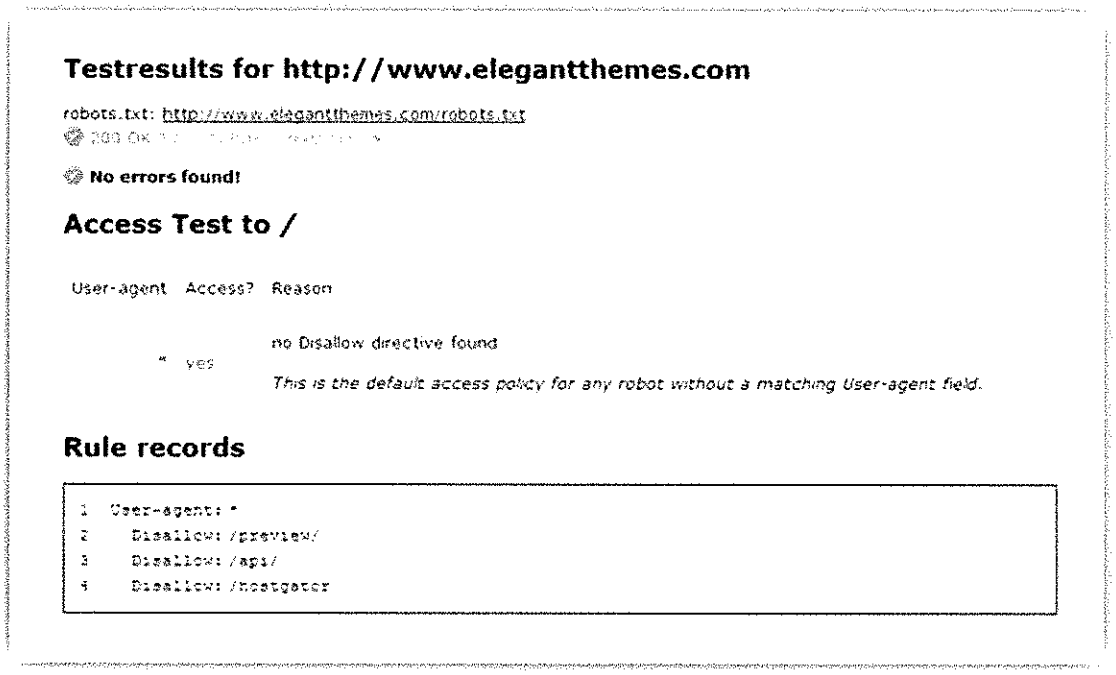
● การเก็บ user role ไว้ใน cookie ช่องโหว่ที่เกิดจากความผิดพลาดประเภทนี้สามารถพบเจอได้บ่อยโดยเฉพาะนักพัฒนาระบบที่ไม่มีความชำนาญ เนื่องจาก cookie เป็นข้อมูลที่สามารถแก้ไขได้โดยผู้ใช้งาน ดังนั้น ผู้ใช้งานอาจจะเปลี่ยน role ที่เก็บไว้ใน cookie ได้เพื่อยกระดับประเภทของผู้ใช้งาน เพื่อป้องกันเหตุการณ์นี้ เซิร์ฟเวอร์ไม่ควรจะเรียกใช้ user role ที่เก็บอยู่ใน cookie แต่ทำการดึง user role มาจากฐานข้อมูล

<u>แนวทางการป้องกัน MFLAC</u>

MFLAC เป็นช่องโหว่ที่เกิดขึ้นจากความผิดพลาดในการอนุญาตการใช้งานให้กับผู้ใช้งานที่ไม่ถูกประเภท วิธีการแก้ไขที่ดีที่สุดคือการ List ฟังก์ชันทั้งหมดที่มีอยู่ในระบบ แล้วทำการตรวจสอบความถูกต้องของ Authorization Process

<u>ตัวอย่างการทดสอบเว็บแอปพลิเคชันที่มีช่องโหว่ Missing Function Level Access Control</u>

● ทดลองเรียกไฟล์ Robots.txt ผ่าน URL ซึ่งถ้าเครื่องแม่ข่ายมีไฟล์ Robots.txt เว็บแอปพลิเคชันจะแสดงข้อมูลที่อยู่ในไฟล์นี้ออกมา



Testresults for http://www.elegantthemes.com

robots.txt: http://www.elegantthemes.com/robots.txt

200 OK

No errors found!

Access Test to /

User-agent	Access?	Reason
*	yes	no Disallow directive found *This is the default access policy for any robot without a matching User-agent field.*

Rule records

```
1  User-agent: *
2    Disallow: /preview/
3    Disallow: /api/
4    Disallow: /hostgator
```

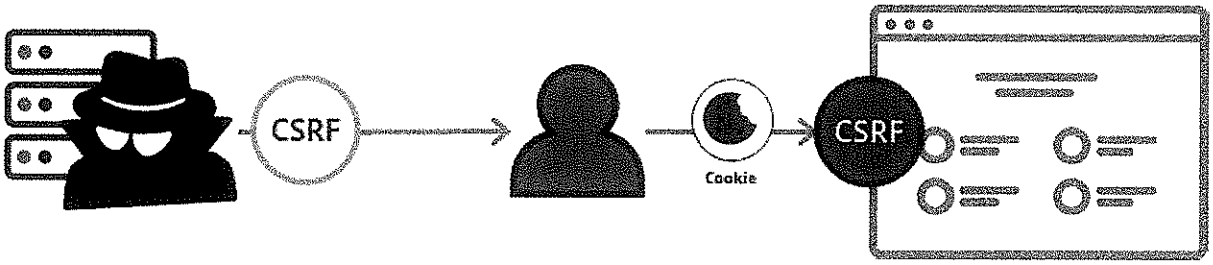
8) Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) เป็นช่องโหว่ที่เกิดขึ้นจากการที่ผู้ไม่หวังดีบังคับให้เว็บเบราว์เซอร์ของเหยื่อส่ง HTTP request ที่ปรับเปลี่ยนแก้ไข รวมถึง session cookie และ ข้อมูลเกี่ยวกับการระบุตัวตน (Authentication) ไปยังเว็บแอปพลิเคชันที่มีช่องโหว่ เพื่อทำกิจกรรม

บางอย่าง เช่น การโอนเงิน, ทำลายข้อมูล, ลบการตั้งค่าความปลอดภัย เป็นต้น ที่แอปพลิเคชันคิดว่ามาจากผู้ใช้ที่ถูกต้องได้ (สวมรอยสิทธิการใช้งานของผู้อื่น)

การทำงานของ CSRF

CSRF เป็นช่องโหว่ที่ผู้ไม่หวังดีส่ง HTML หรือ JavaScript ให้เว็บเบราเซอร์ของเหยื่อส่ง HTTP request เพื่อไปกระทำการบางอย่างที่อาจจะเป็นอันตรายต่อผู้ใช้งาน



แนวทางการในการป้องกัน CSRF

CSRF มีสาเหตุหลักมาจากการที่เว็บแอปพลิเคชันไม่ได้ทำการตรวจสอบว่า request ที่ถูกส่งมาจากผู้ใช้งานนั้นถูกส่งมาจากผู้ใช้งานจริงโดยตั้งใจหรือไม่ ดังนั้น แนวทางในการป้องกัน CSRF ประกอบไปด้วย 3 วิธีหลัก ๆ ซึ่งแต่ละวิธีก็จะมีข้อดีข้อเสียแตกต่างกันไปดังนี้

- Synchronizer Token Pattern (STP) หลักการทำงานของ STP คือ เมื่อผู้ใช้งานทำการ Login (เริ่ม SESSION ใหม่) ให้ทำการสร้าง Random String ขึ้นมา 1 ชุดทุกครั้ง และเมื่อผู้ใช้งานจะทำการส่งคำสั่งสำคัญ เช่น การโอนเงิน ให้ทำการใส่ Random String ที่ผูกกับ SESSION มาด้วย จะทำให้ Attacker ไม่สามารถโจมตีผู้ใช้งานด้วย CSRF ได้เนื่องจาก Attacker ไม่ทราบ Random String ของผู้ใช้งาน

- HTTP referrer header ซึ่ง HTTP referrer เป็น Field ใน HTTP request ที่ใช้ในการบอกว่า Request ที่ถูกส่งมานั้นมาจาก URL อะไร การตรวจสอบ HTTP referrer จะสามารถป้องกัน CSRF ได้เนื่องจาก หาก Attacker ทำการ Redirect เว็บเบราเซอร์โดยให้ HTTP referrer เป็น URL แทน ดังนั้นเว็บแอปพลิเคชันไม่ทำตามคำขอ Attacker

- Re-authentication & CAPTCHA

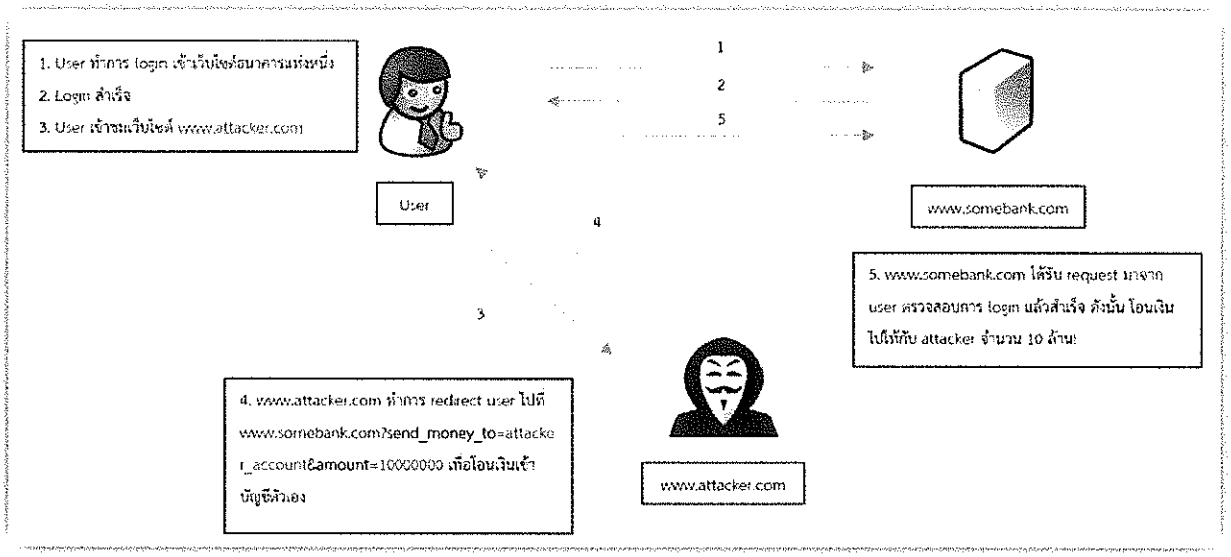
Re-authentication เป็นกระบวนการที่ให้ผู้ใช้งานทำการใส่ password อีกครั้งเพื่อยืนยันการส่งคำสั่งที่สำคัญ เช่น การโอนเงิน การทำ Re-Authentication สามารถป้องกัน CSRF ได้ผลดีมาก เนื่องจากผู้ที่ส่งคำสั่งจำเป็นต้องรู้ password

CAPTCHA เป็นเครื่องมือที่ถูกนำมาใช้มากในการตรวจสอบบุคคล โดยส่วนมากผู้ใช้งานจะเจอกับ CAPTCHA เมื่อทำการสมัครสมาชิกเว็บไซต์ต่าง ๆ จุดประสงค์ของ CAPTCHA นั้นถูกสร้างขึ้นมาเพื่อแยกแยะระหว่าง BOT และมนุษย์ อย่างไรก็ตามการใช้ CAPTCHA ในการยืนยันการส่งคำสั่งนับได้ว่าเป็นอีกวิธีที่สามารถป้องกัน CSRF ได้อย่างดี เนื่องจากการทำ Redirect จะไม่สามารถใช้งานได้

ตัวอย่างการโจมตีด้วย CSRF

ขณะที่เหยื่อล็อกอินเข้าใช้เว็บไซต์ของธนาคารแล้ว และเซสชันยังคงค้างอยู่ เหยื่อเผลอกดลิงค์บางอย่างจากแฮกเกอร์ที่อาจแฝงมากับอีเมล Phishing หรือสคริปต์บนเว็บไซต์ต่าง ๆ ส่งผลให้เหยื่อกระทำธุรกรรมตามที่แฮกเกอร์ต้องการ เช่น โอนเงินจากบัญชีของตนไปให้แฮกเกอร์ โดยที่เหยื่อไม่รู้ตัว และเว็บไซต์ของธนาคารก็นึกว่าเป็นการทำธุรกรรมปกติที่มาจากตัวเหยื่อเอง เรียกได้ว่าแฮกเกอร์ใช้ประโยชน์จากข้อมูลล็อกอินและเซสชันที่ค้างอยู่ของเหยื่อในการปลอมตัวเป็นเหยื่อเพื่อทำธุรกรรมตามที่ตนต้องการได้นั่นเอง

การโจมตีแบบนี้ตรวจจับได้ยากเนื่องจากเป็นการกระทำธุรกรรมปกติในนามของเหยื่อเอง ทั้งข้อมูลการพิสูจน์ตัวตนและ IP ต่างถูกนำมาใช้เพื่อยืนยันว่าเป็นเจ้าของบัญชีทำธุรกรรมจริง ส่งผลให้ทางธนาคารต้องใช้เวลานานในการเก็บข้อมูลและสืบสวน ก่อนที่จะดำเนินเรื่องย้อนหลังการทำธุรกรรม หรือทำให้ธุรกรรมเป็นโมฆะไป



9) Using Know Vulnerable Components

Using Know Vulnerable Components (UCKV) เป็นช่องโหว่ที่เกิดขึ้นจากตัวระบบหรือแอปพลิเคชัน ที่มีการใช้งานไลบรารี่และส่วนประกอบ หรือคำสั่งที่ไม่มีความปลอดภัยอยู่ในระบบ จึงทำให้ระบบมีความเสี่ยงในการถูกโจมตี

สาเหตุหลักของ Using Components with Known Vulnerability

โดยปกติเมื่อเริ่มต้นการพัฒนาเว็บแอปพลิเคชัน นักพัฒนาโปรแกรมควรที่จะใช้ส่วนประกอบทุกส่วนเวอร์ชันล่าสุด อย่างไรก็ตามมีหลายครั้งที่เราไม่สามารถใช้ซอร์ฟแวร์เวอร์ชันล่าสุดได้ ด้วยเหตุผลต่าง ๆ เช่น ความเข้ากันได้กับซอร์ฟแวร์และฮาร์ดแวร์ (Software and Hardware

compatibility) ทำให้นักพัฒนาต้องเลือกใช้เวอร์ชันของซอร์ฟแวร์ที่ต่ำกว่าเวอร์ชันปัจจุบัน ไม่เพียงเท่านั้น การพัฒนาซอร์ฟแวร์และเว็บแอปพลิเคชันในบางครั้งใช้เวลาเป็นเวลาหลายเดือนจนถึงหลายปี ช่องโหว่ใหม่ ๆ ก็อาจจะถูกค้นพบได้ในระหว่างกันพัฒนา และสุดท้าย แม้ว่าหลังจากเสร็จสิ้นการพัฒนาซอร์ฟแวร์แล้ว ซอร์ฟแวร์ต่าง ๆ เช่น Framework และ Library ยังคงต้องได้รับการดูแลและคอยหมั่น Update Security Patch อยู่เสมอ (Security Maintenance) การจัดการ Security Patch เป็นเรื่องที่มีความซับซ้อนและละเอียดอ่อน เนื่องจากการ Update Version ของส่วนประกอบ (Component) อาจจะก่อให้เกิดปัญหาเรื่องความเข้ากันได้กับตัวเว็บแอปพลิเคชัน (Compatibility Problem) ด้วยเหตุดังกล่าวทำให้ UCKV เป็นช่องโหว่ที่พบได้บ่อยที่สุดของเว็บแอปพลิเคชัน

<u>แนวทางการป้องกัน Using Components with Known Vulnerability</u>

การป้องกันช่องโหว่ประเภทนี้สามารถทำได้โดยการหมั่นตรวจสอบช่องโหว่ที่เกิดขึ้นใหม่ ๆ ในแต่ละวัน แนวทางการจัดการกับช่องโหว่ประเภทนี้ดีที่คือ ให้ทำการ List ส่วนประกอบของเว็บแอปพลิเคชันทั้งหมดพร้อมทั้งเวอร์ชัน จากนั้นให้คอยเฝ้าตรวจสอบ CVE ใหม่ ๆ ที่ออกมาว่ามีช่องโหว่ใดที่อาจจะส่งผลต่อความปลอดภัยของเว็บแอปพลิเคชัน

10) Unvalidated Redirects and Forwards

Unvalidated Redirects and Forwards คือ ช่องโหว่ที่เกิดขึ้นจากระบบขาดการป้องกันการรับค่าอินพุตอย่างเหมาะสม ทำให้ผู้บุกรุกสามารถโจมตีด้วยเทคนิค Phishing (การสร้างเว็บไซต์หลอกลวง) เพื่อหลอกลวงเหยื่อให้ป้อนข้อมูลสำคัญของระบบ เช่น ชื่อผู้ใช้งานและรหัสผ่าน, ข้อมูลบัตรเครดิต, ข้อมูลส่วนบุคคล เป็นต้น

<u>สาเหตุและแนวทางในการป้องกัน Unvalidated Redirects and Forwards (URF)</u>

URF เป็นช่องโหว่ที่เกิดจากการที่เว็ปแอปพลิเคชันนั้นไม่ได้ตรวจสอบข้อมูลที่รับมาจากผู้ใช้งานก่อนทำการ Redirect ดังนั้น วิธีการป้องกันที่ดีที่สุดในการป้องกันช่องโหว่ประเภทนี้ประกอบไปด้วย

- พยายามไม่ redirect ไปเว็บไซต์อื่นหากเป็นไปได้
- ไม่เปิดโอกาสให้ผู้ใช้งานใส่ URL มาเพื่อ redirect ผ่านเว็บไซต์
- ตรวจสอบ URL ที่รับเข้ามาโดยการสร้าง white list ของเว็บไซต์ที่น่าเชื่อถือก่อนทำการ redirect
- แจ้งเตือนให้ผู้ใช้งานได้รับรู้ว่ากำลังจะถูก redirect ไปยังเว็บไซต์อื่น ก่อนทำการ redirect

ตัวอย่างการทดสอบเว็บแอปพลิเคชันที่มีช่องโหว่ Unvalidated Redirects and Forwards

- ใส่ลิงค์ที่ต้องการให้ Redirect ไปยังหน้าเว็บไซต์นั้น ๆ ที่ตัวแปร returnURL ดังภาพด้านล่าง แต่การที่จะให้เหยื่อกดลิงค์นี้ต้องอาศัยเทคนิค Social Engineer

2.2 ข้อเสนอแนะในการนำความรู้ที่ได้รับมาประยุกต์ใช้กับองค์กร

รฟม. มีการพัฒนาเว็บแอปพลิเคชันภายในองค์กรและภายนอกองค์กร ดังนั้น การเข้ารับฝึกอบรมในหลักสูตร Web Application Security สามารถนำความรู้ในการฝึกอบรมมาใช้ในการตรวจสอบหาช่องโหว่เว็บแอปพลิเคชันขององค์กร เพื่อประเมินความเสี่ยง และดำเนินแก้ไขช่องโหว่ที่พบ ทำให้เว็บแอปพลิชันขององค์กรมีความมั่นคงปลอดภัย และป้องกันการถูกโจมตีจากผู้ไม่ประสงค์ดีได้

2.3 ความคิดเห็นเกี่ยวกับการฝึกอบรม/สัมมนา

2.3.1 หลักสูตรที่ฝึกอบรม/สัมมนาครั้งนี้ช่วงเพิ่มพูนความรู้ของท่านเพียงใด

☑ มาก ☐ ปานกลาง ☐ น้อย

2.3.2 ท่านคิดว่าการฝึกอบรม/สัมมนาครั้งนี้มีประโยชน์กับตัวท่านและองค์กรเพียงใด

☑ มาก ☐ ปานกลาง ☐ น้อย

ระบุเหตุผล (ตอบได้มากกว่า 1 ข้อ)

☑ เนื้อหาเกี่ยวข้องโดยตรงและสามารถนำไปใช้กับการปฏิบัติงานได้อย่างดี
☐ เนื้อหาไม่เกี่ยวข้องกับการปฏิบัติงาน
☑ เป็นความรู้เสริม และมีประโยชน์ในการปฏิบัติงาน
☐ ได้แลกเปลี่ยนประสบการณ์กับบุคคลนอกองค์การ
☑ วิทยากรมีความรู้ ความสามารถ และประสบการณ์ ในการบรรยายเป็นอย่างดี
☐ เนื้อหาการอบรมไม่ตรงกับหัวข้อการบรรยาย
☐ อื่น ๆ .....

3. วิทยากรที่ให้ความรู้ในหลักสูตรนี้ ได้แก่

	ชื่อ-สกุล	จากสถาบัน/หน่วยงาน	ระดับความสามารถของวิทยากร		
3.1	Khajornchol Puwarang	Mindterra	☑ มาก	☐ ปานกลาง	☐ น้อย
3.2	.....	.....	☐ มาก	☐ ปานกลาง	☐ น้อย

4. ข้อเสนอแนะในการส่งพนักงานเข้ารับการฝึกอบรม/สัมมนาตามหลักสูตรนี้ในครั้งต่อไป

การเข้ารับฝึกอบรมในหลักสูตร Web Application Security มีประโยชน์ต่อการพัฒนาเว็บแอปพลิเคชันขององค์กรให้ปลอดภัยจากภัยคุกคามจากผู้ไม่ประสงค์ดี จึงมีความเหมาะสมที่บุคลากรด้านเทคโนโลยีสารสนเทศจะเข้ารับการฝึกอบรมหลักสูตรนี้ในครั้งต่อไป

ลงชื่อ ..... ผู้เข้าอบรม
(น.ส. .....)
ตำแหน่ง พนักงานปฏิบัติการระบบเครือข่าย 4
วันที่ 21/7/60

(น.ส. ยมลรดา ก้าวแผง)
21/7/60