การป้องกันการโจมตีแบบครอสไซต์สคริปติ้ง

วรากรณ์ สุภัคนิกร[*]
ดร.ชัยพร เขมะภาตะพันธ์[**]

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์ที่จะนำเสนอแนวทางวิธีที่จะป้องกันการโจมตีครอสไซต์สคริปติ้ง เพื่อลดความเสี่ยงที่ผู้ใช้งานจะถูกโจมตี โดยแบ่งแยกวิธีป้องกันการโจมตีเป็น 2 รูปแบบ ได้แก่ แบบที่ 1 ทำการป้องกันด้วยการกรองข้อมูลเพื่อดักจับและลบรหัสโปรแกรมอันตรายที่อาจเป็น Cross-Site Scripting (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

1. บทนำ

ในปัจจุบันทุกหน่วยงานมักจะมีเว็บไซต์เป็นของตนเองซึ่งจะมีการเปิดให้บุคคลภายนอกสามารถเข้ามาเยี่ยมชมเว็บไซต์ได้โดยเปิดพอร์ท TCP 80 (http) หรือ TCP 443 (https) 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) เป็นวิธีการยอดนิยมที่ผู้โจมตี (hacker) รู้จักกันดี ซึ่งเป็นเทคนิคการฝังโค้ดเข้าไปกับหน้าเว็บเพจที่มีช่องโหว่เมื่อผู้ใช้โหลดหน้าเว็บเพจไป ค่าที่สำคัญบางอย่าง เช่น cookie, username หรือ password เป็นต้น ก็อาจจะถูกขโมยไปได้ ช่องโหว่ที่สามารถทำให้เกิดการโจมตี

[*] นักศึกษาหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์

[**] ที่ปรึกษาสารนิพนธ์

Cross-site Scripting (XSS) ได้ก็เนื่องมาจาก web application รับข้อมูลเข้ามาจากผู้ใช้ ซึ่งโปรแกรมเมอร์ที่พัฒนา web application ดังกล่าวไม่ได้ทำการตรวจสอบความปลอดภัยของข้อมูลนั้นก่อนทำการบันทึกเข้าสู่ฐานข้อมูล หรือไม่มีการตรวจสอบการแก้ไขข้อมูลในฐานข้อมูล

บทความฉบับนี้จึงมุ่งเน้นเพื่อเสนอรูปแบบวิธีในการโจมตี Cross-site Scripting (XSS) และวิธีการป้องกันการโจมตี Cross-site Scripting (XSS)

2. แนวคิดทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ครอสไซต์สคริปติ้ง (Cross-site scripting)

ครอสไซต์สคริปติ้ง (Cross-site scripting : XSS) เป็นการโจมตีแบบระบบประยุกต์ (Application) ประเภท malicious injection โดยอาศัยหลักการที่คอมพิวเตอร์ผู้ใช้งาน (Client computer) สามารถส่งข้อความอะไรก็ได้ไปยังเครื่องบริการ (Server) แล้วเครื่องบริการนำข้อความที่ได้รับมาไปประมวลผล

ภาพที่ 1 รูปแบบการโจมตีครอสไซต์สคริปติ้งแบบถาวร

โดยมีประเภทของครอสไซต์สคริปติ้ง ดังนี้

1.ครอสไซต์สคริปติ้งแบบถาวร (Stored Cross-Site Scripting) เป็นเทคนิคการโจมตี (Persistent) โดยอาศัยประโยชน์จากเว็บบอร์ด เว็บบล็อก (Web blog) หรือสมุดเยี่ยมชม (Guestbook) ที่โดยปกติเว็บระบบประยุกต์ (Web application) กลุ่มนี้จะให้ผู้ใช้งานสามารถกรอกเนื้อหา (Content) ได้ด้วยตนเองหรือจะนำข้อมูลที่รับเข้าไปเก็บและใช้งานโดยที่ไม่ได้ทำการตรวจสอบความปลอดภัยก่อน ผู้โจมตีก็จะสามารถใส่บทคำสั่งเข้าไปได้โดยตรงเลย

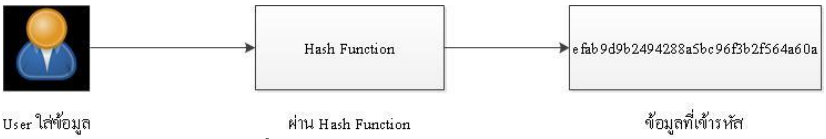
2. ครอสไซต์สคริปติ้งแบบชั่วคราว (Reflected Cross-Site Scripting) เป็นเทคนิคการโจมตีแบบชั่วคราว (Non-persistent) โดยส่วนใหญ่จะอยู่ในรูปแบบของการเชื่อมโยง (Link) เมื่อเหยื่อ (Victim) ทำการคลิกที่การเชื่อมโยงแล้วบทคำสั่งจะทำการโจมตี ปกติจะอยู่ในเว็บไซต์ที่มีการรับข้อมูลที่รับเข้าจากผู้ใช้งานมาแสดงผลบนเบราว์เซอร์

3. ครอสไซต์สคริปติ้งแบบชนิด 0 (DOM-based Cross-Site Scripting) เป็นการโจมตีโดยอาศัยคุณสมบัติของ Document Object Model (DOM) มาช่วยในการทำการโจมตี โดยนำเข้าข้อมูลที่

รับเข้ามาแปลงให้เป็นคำสั่ง เช่น การที่เครื่องบริการไปอ่านข้อมูลจาก RSS feeds มาแล้วทำการแสดงผลไปยังเบราว์เซอร์โดยที่ไม่ทำการตรวจสอบก่อน

2.2 ฟังก์ชันแฮช (Hash Function)

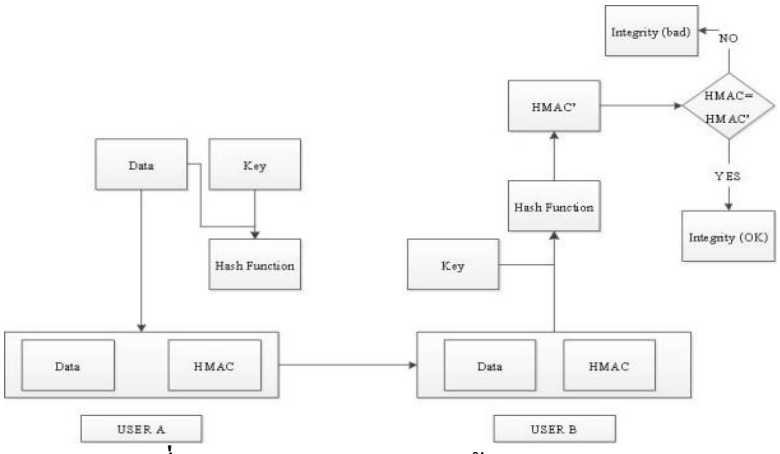
แฮชฟังก์ชั่นเป็นการย่อยข้อมูล หรือ เป็นการแมพ (map) ค่าไบนารีของข้อมูลที่มีขนาด ไม่คงที่ให้เป็นค่าไบนารีของข้อมูลที่มีขนาดคงที่สามารถใช้ประโยชน์จาการทำแฮชได้ เช่น กรณีการเก็บรหัสของเว็ปไซต์ผู้ให้บริการต่าง ๆที่มีการล็อกอินเพื่อเข้าสู่ระบบจะมีการเก็บรหัสที่มีการผ่านฟังก์ชั่นแฮชเก็บลงใน ฐานข้อมูล เมื่อมีผู้ใช้ ทำการล็อกอินเข้าสู่ระบบ รหัสผ่านจะถูกนำมาผ่านฟังก์ชั่นแฮชแล้วนำไปเปรียบเทียบกับในฐานข้อมูลถ้าตรงกันแสดงว่า ผู้ใช้ คนนั้นสามารถเข้าใช้ระบบได้จริง



ภาพที่ 2 การเข้ารหัสด้วยฟังก์ชันแฮช

2.3 Hash Message Authentication Code (HMAC)

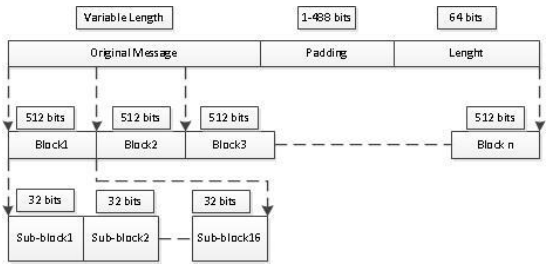
แนวคิดของ HMAC คือ ต้องการยืนยันข้อความ (Message Authentication) ที่ถูกส่ง ผ่านระหว่างเครือข่าย ว่าข้อความที่ส่งไปนั้นมีการถูกเปลี่ยนแปลงแก้ไขหรือไม่



ภาพที่ 3 กระบวนการตรวจสอบข้อมูลของ HMAC

2.4 กระบวนการทำงาน MD5

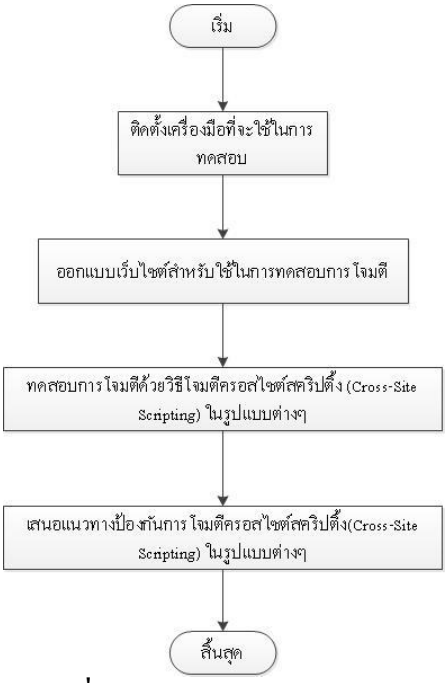
การทำงานของอัลกอริทึม MD5 จะมีการแบ่งข้อความต้นฉบับขนาดใด ๆ ออกเป็นกลุ่มบิตหลาย กลุ่มบิต ที่มีขนาด 512 บิต ตามลำดับ แต่ละกลุ่มบิตนี้จะถูกแบ่งย่อยออกเป็น 16 กลุ่มย่อย(Sub-block) กลุ่มบิตย่อยละ 32 บิต และเมื่อผ่านการดำเนินการตามที่กำหนดไว้ในอัลกอริทึม MD5 จนครบแล้วจะให้ผลลัพธ์เป็นเซตของกลุ่มบิตย่อยขนาด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



ภาพที่ 4 การทำงานของอัลกอริทึม MD5

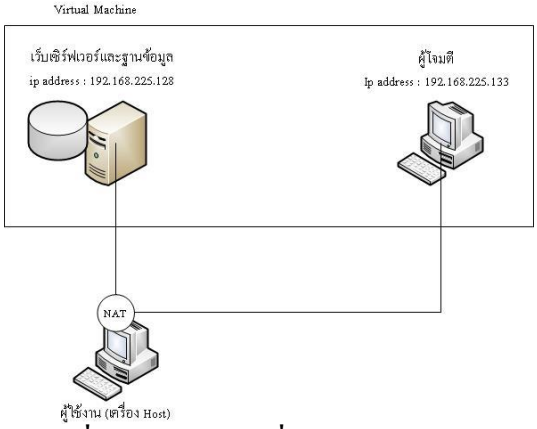
3. วิธีการดำเนินงาน

บทความนี้จัดทำเพื่อแสดงรูปแบบการโจมตี ครอสไซต์สคริปติ้ง (Cross-Site Scripting) โดยออกแบบระบบเพื่อใช้ในการทดสอบการโจมตี และเสนอวิธีการป้องกันการโจมตี ครอสไซต์สคริปติ้ง (Cross-Site Scripting)ในฝั่งของเครื่องแม่ข่าย (Server) โดยมีภาพรวมในการดำเนินงาน ดังนี้



ภาพที่ 5 ภาพรวมในการดำเนินงาน

3.1 รูปแบบการเชื่อมต่อและติดตั้งเครื่องมือที่ใช้ในการทดสอบโดยรวม



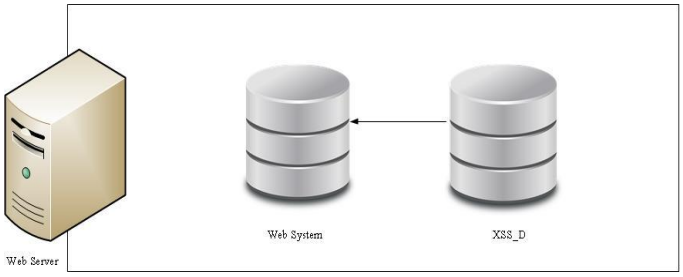
ภาพที่ 6 รูปแบบการเชื่อมต่อระบบโดยรวม

3.2 การออกแบบฐานข้อมูล

3.2.1 ฐานข้อมูลของเว็บเซิร์ฟเวอร์จะแบ่งออกเป็น 2 ฐานข้อมูล

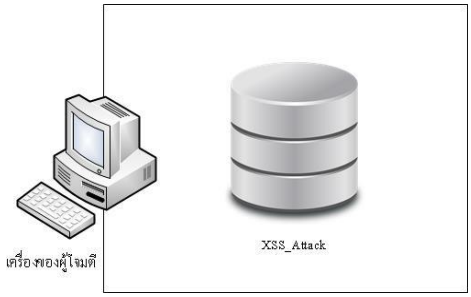
3.2.1.1 Web System เป็นฐานข้อมูลสำหรับเก็บข้อมูลต่าง ๆ ของเว็บไซต์

3.2.1.2 XSS_D เป็นฐานข้อมูลไว้เก็บข้อมูล hash เพื่อไว้ใช้สำหรับตรวจสอบความถูกต้องของไฟล์และข้อมูลในฐานข้อมูล



ภาพที่ 7 ฐานข้อมูลเว็บเซิร์ฟเวอร์

3.2.2 ฐานข้อมูลของผู้โจมตี จะประกอบด้วยฐานข้อมูล XSS_Attack ที่ใช้สำหรับบันทึกค่า cookie ที่ขโมยได้จากผู้ใช้งานอื่น ๆ



ภาพที่ 8 ฐานข้อมูลของเครื่องผู้โจมตี

3.3 ทดสอบการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) และแนวทางการป้องกัน

3.3.1 วิธีที่ 1 การโจมตีโดยฝังข้อมูลที่เป็นการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) ผ่านทางหน้าเว็บไซต์

ทดสอบโดยการเข้าใช้งานเว็บไซต์ด้วยสิทธิที่เป็น user ทำการบันทึกข้อมูลคำสั่ง (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>
```

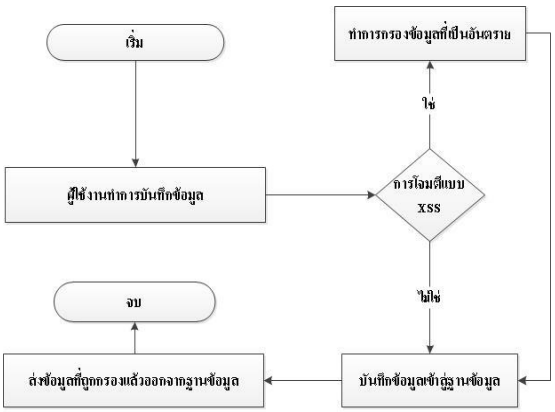
ที่เลือกใช้คำสั่งนี้ในการทดสอบเนื่องจากการโจมตีโดยฝังข้อมูลผ่านเว็บบอร์ด การที่จะเข้าไปดูเนื้อหาในแต่ละกระทู้ผู้ใช้งานต้องทำการคลิกลิงก์ที่ชื่อหัวข้อกระทู้ที่ต้องการ ทำให้ผู้ใช้งานเข้าใช้งานตามปกติโดยไม่ทราบว่ากำลังโดนโจมตีอยู่ คำสั่งนี้เมื่อมีผู้ใช้งานเข้ามาใช้งานเว็บบอร์ด ก็จะแสดงชื่อหัวข้อกระทู้ “Click_Here”

ภาพที่ 9 การบันทึกข้อมูลคำสั่ง (Code) ที่เป็นอันตรายเข้าไปที่ชื่อหัวข้อกระทู้ของเว็บบอร์ด

ซึ่งถ้าผู้ใช้งานคนอื่นทำการคลิก สมมติให้ผู้ใช้งานที่ได้รับสิทธิ admin เข้ามาคลิกที่ลิงก์ดังกล่าว คำสั่งนี้จะไปสั่งการทำงาน ให้ไปรันคำสั่งในไฟล์ hack.php ในเครื่องของผู้โจมตี ซึ่งจะทำการส่งค่า cookie ของผู้ที่ใช้งานดังกล่าวไปบันทึกไว้ในฐานข้อมูลของผู้โจมตี ก็จะสามารถนำค่า cookie ที่ได้ไปตั้งที่เว็บเบราว์เซอร์แล้วทำการรีเฟรชหน้าก็จะได้สิทธิ admin มาใช้

<u>แนวทางการป้องกันการโจมตีครอสไซต์สคริปติ้ง ผ่านทางหน้าเว็บไซต์</u>

ป้องกันโดยวิธีการกรองข้อมูลที่คาดว่าจะเป็นคำสั่งการโจมตีครอสไซต์สคริปติ้ง ซึ่งเสนอรูปแบบวิธีการกรองข้อมูลทั้งหมด 6 วิธี ซึ่งสามารถเลือกไปใช้ได้ตามความเหมาะสม ดังนี้



ภาพที่ 12 การกรองข้อมูลที่เป็นการโจมตีครอสไซต์สคริปติ้ง

1. การกรองสัญลักษณ์ที่เป็นอันตรายโดยทำการเปลี่ยนสัญลักษณ์ดังกล่าวเป็นเลขฐาน 16

ซึ่งเขียนโปรแกรมสำหรับการป้องกันโดยการกรองสัญลักษณ์ที่คาดว่าจะเป็นอันตรายและเปลี่ยนให้เป็นเลขฐาน 16 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 13 การกรองสัญลักษณ์และเปลี่ยนให้เป็นเลขฐาน 16

2. การกรองสัญลักษณ์ที่เป็นอันตรายโดยทำการเปลี่ยนสัญลักษณ์ดังกล่าวเป็น HTML NUMBER

ซึ่งเขียนโปรแกรมสำหรับการป้องกันโดยการกรองสัญลักษณ์ที่คาดว่าจะเป็นอันตรายและเปลี่ยนให้กลายเป็น HTML NUMBER 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 14 การกรองสัญลักษณ์และเปลี่ยนให้เป็น HTML NUMBER

3. การกรองสัญลักษณ์ที่เป็นอันตรายโดยทำการเปลี่ยนสัญลักษณ์ดังกล่าวเป็น HTML NAME

ซึ่งเขียนโปรแกรมสำหรับการป้องกันโดยการกรองสัญลักษณ์ที่คาดว่าจะเป็นอันตรายและเปลี่ยนให้กลายเป็น HTML NAME แล้วจึงนำไปบันทึกเข้าสู่ฐานข้อมูล

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

ภาพที่ 15 การกรองสัญลักษณ์และเปลี่ยนให้เป็น HTML ENTITY

4. การกรองข้อมูลที่โจมตีด้วยการเข้ารหัสแบบ base_64

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

```php
<?php
function filter_base64($txt)
{
    $pattern_fileter_base64 = '/.*base64.*/i';
    return preg_replace($pattern_fileter_base64,'You_cannot_hack',$txt);
}
?>
```

ภาพที่ 16 การกรองข้อมูลที่โจมตีด้วยการเข้ารหัส 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 อื่น ๆที่สามารถใช้ได้ จึงต้องทำการกรอง TAG ที่คาดว่าจะเป็นอันตราย โดยนำข้อมูลที่ป้อนเข้ามานำมาเข้าฟังก์ชัน 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 17 การกรองข้อมูลที่เป็น TAG อันตราย

[^>] คือ ตัวอักษรทุกตัวที่ไม่ใช่ >

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

\s คือ ช่องว่าง

i คือ จะใช้อักษรตัวใหญ่หรือตัวเล็กก็ให้ความหมายเดียวกัน

6. การกรองข้อมูลคำสั่งที่เป็นอันตราย

การกรองข้อมูลคำสั่งที่เป็นอันตราย โดยนำข้อมูลที่ป้อนเข้ามา นำมาเข้าฟังก์ชัน 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 18 การกรองข้อมูลคำสั่งที่เป็นอันตราย

จากการป้องกันโดยกรองข้อมูลที่เป็นอันตราย จะพบว่า สามารถป้องกันการโจมตีการบันทึกข้อมูลคำสั่งที่เป็นอันตรายจากผู้โจมตีโดยการกรองข้อมูลได้ ดังนี้

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

ภาพที่ 19 การกรองข้อมูลที่เป็นอันตรายก่อนบันทึกเข้าสู่ฐานข้อมูล

3.3.2 วิธีที่ 2 การโจมตีโดยบันทึกคำสั่งการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting)ไว้ในไฟล์หรือฐานข้อมูลโดยตรง

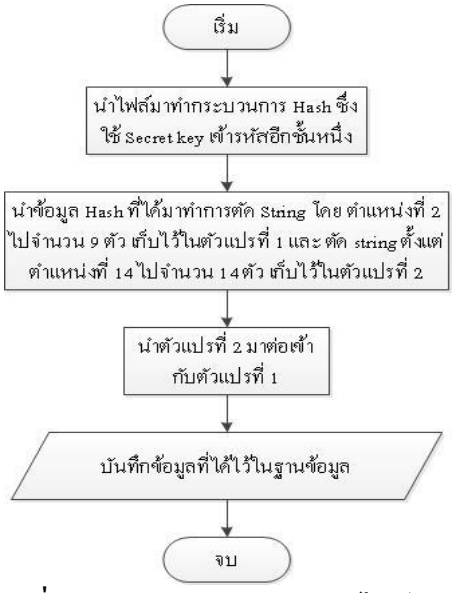
3.3.2.1 การโจมตีโดยฝังคำสั่งการโจมตีครอสไซต์สคริปติ้งไว้ในไฟล์

```
<?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 20 การโจมตีโดยฝังคำสั่งการโจมตีครอสไซต์สคริปติ้งไว้ในไฟล์

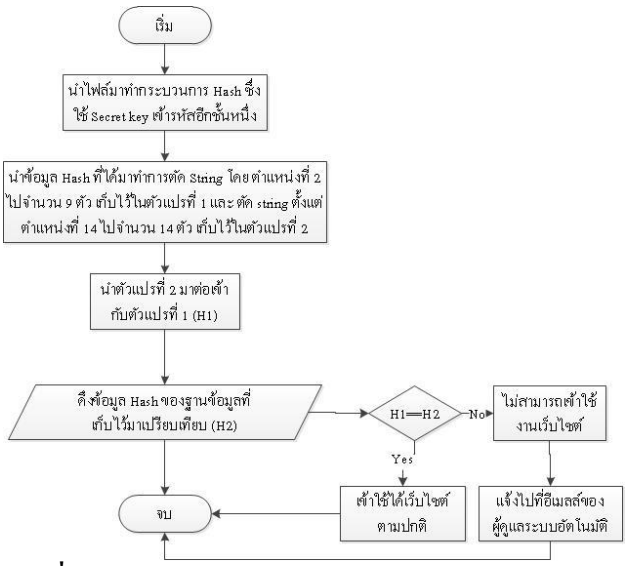
แนวทางการป้องกันการโจมตีครอสไซต์สคริปติ้งโดยฝังข้อมูลที่เป็นอันตรายไว้ในไฟล์ที่เว็บเซิร์ฟเวอร์

ทำกระบวนการ hash ไฟล์แล้วเก็บข้อมูลดังกล่าวไว้ที่ฐานข้อมูลที่แยกออกจาก Web System เพื่อเพิ่มความปลอดภัยจากการโจมตีมากขึ้น



ภาพที่ 21 กระบวนการการ hash ไฟล์ข้อมูล

เมื่อผู้ใช้งานเข้าใช้งานก็ให้ทำกระบวนการ hash ดังกล่าวอีก 1 รอบและนำข้อมูลที่ได้มาเปรียบเทียบกับ hash ที่เก็บไว้ในฐานข้อมูล

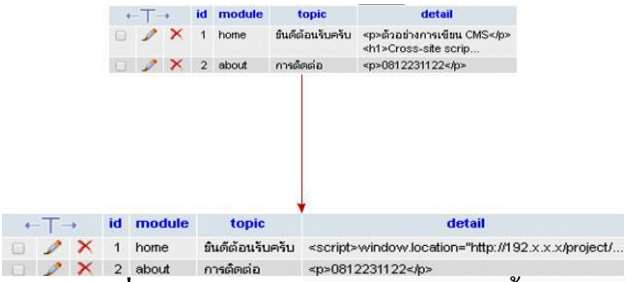


ภาพที่ 22 กระบวนการตรวจสอบการแก้ไขข้อมูลในไฟล์

หากตรวจสอบพบว่าข้อมูลที่ทำการ hash ไม่ตรงกับข้อมูล hash ที่เก็บไว้ในฐานข้อมูล ระบบจะไม่อนุญาตให้ผู้ใช้งานเข้าใช้งาน และจะส่งอีเมล์ไปแจ้งเตือนผู้ดูแลระบบว่ามีการแก้ไขข้อมูลในไฟล์

3.3.2.2 การโจมตีโดยโดยฝังคำสั่งการโจมตีครอสไซต์สคริปติ้ง (Cross-Site Scripting) ในฐานข้อมูล

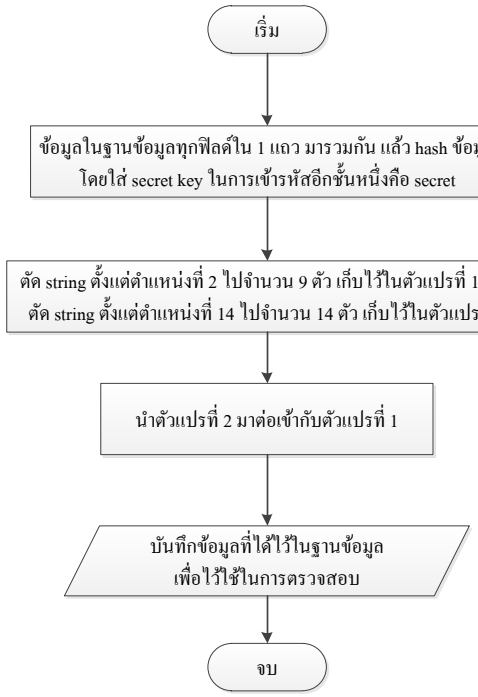
เนื่องจากปัจจุบันมีการใช้ระบบการจัดการเนื้อหาของเว็บไซต์(Content Management System : CMS) ทำให้มีการเก็บข้อมูลเนื้อหาต่าง ๆ ของหน้าเว็บไซต์ไว้ในฐานข้อมูลเมื่อถูกโจมตีโดยเข้ามาแก้ไขในฐานข้อมูลจะทำให้ผู้ใช้งานที่เข้าใช้งานเว็บไซต์ดังกล่าวโดนโจมตีไปด้วย



ภาพที่ 23 การเปลี่ยนแปลงข้อมูลระบบการจัดการเนื้อหาของเว็บไซต์

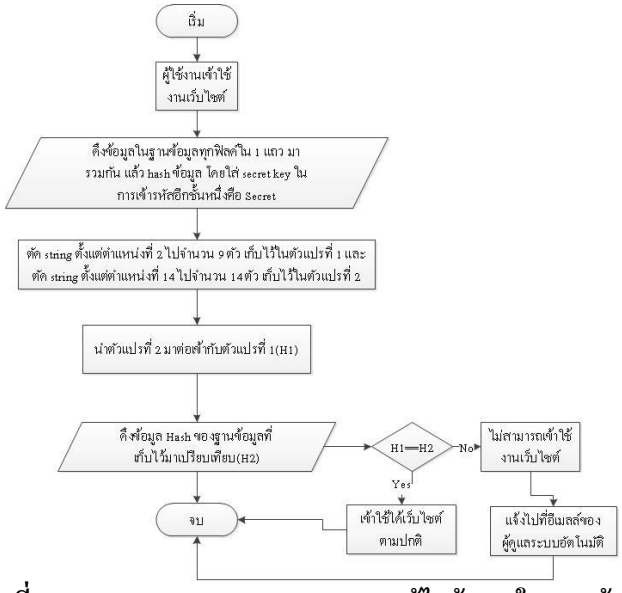
ทำกระบวนการ hash ข้อมูลในฐานข้อมูลแล้วเก็บข้อมูลดังกล่าวไว้ที่ฐานข้อมูลที่แยกออกจาก Web System เพื่อเพิ่มความปลอดภัยจากการโจมตีมากขึ้น

<u>แนวทางการป้องกันการฝังคำสั่งที่เป็นอันตรายไว้ในฐานข้อมูลโดยตรง</u>



ภาพที่ 24 กระบวนการการ hash ข้อมูลในฐานข้อมูล

เมื่อผู้ใช้งานเข้าใช้งานก็ให้ทำกระบวนการ hash ดังกล่าวอีก 1 รอบและนำข้อมูลที่ได้มาเปรียบเทียบกับ hash ที่เก็บไว้ในฐานข้อมูล



ภาพที่ 25 กระบวนการตรวจสอบการแก้ไขข้อมูลในฐานข้อมูล

เมื่อมีผู้เข้าใช้งานทำการตรวจสอบกลับโดยใช้วิธีที่เข้ารหัสจากในฐานข้อมูล หากไม่ตรงกันให้แจ้งเตือนไม่ให้เข้าใช้งาน และแจ้งไปในอีเมล์ของผู้ดูแลระบบ

<u>ผลการทดลองเปรียบเทียบประสิทธิภาพ</u>

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

ภาพที่ 26 ตารางเปรียบเทียบการโจมตี Cross-site Scripting แล้วได้ผลสำเร็จ

4. การเปรียบเทียบกับงานวิจัยอื่น ๆ

จากการทดสอบการป้องกันการโจมตีจะพบว่า งานวิจัยนี้สามารถที่จะป้องกันการโจมตีครอสไซต์สคริปติ้งแบบถาวร (Stored Cross Site Scripting) ได้ เนื่องจากสามารถที่จะป้องกันการบันทึกข้อมูลต่าง ๆที่เป็นอันตรายเข้าสู่ฐานข้อมูลโดยการกรองข้อมูล และ ใช้วิธีการ hash ในการตรวจสอบความถูกต้องของข้อมูลไฟล์และฐานข้อมูลว่าไม่ได้ถูกเปลี่ยนแปลงแก้ไข ทำให้เกิดความปลอดภัยในการใช้งาน ซึ่งไม่ได้ป้องกันในส่วนของ Non-Persistent และ DOM จากการเปรียบเทียบการป้องกันกับงานวิจัยอื่น ๆ จะได้ดังนี้

ตารางเปรียบเทียบกับงานวิจัยอื่นๆ			
งานวิจัย	Non-Persistent	Stored	DOM
งานวิจัยนี้	✖	✓	✖
การตรวจจับปัญหาครอสไซต์สคริปติ้งโดยใช้เว็บพร็อกซี่[6]	✓	✖	✖
Enhanced Browser Defense for Reflected Cross-Site Scripting [11]	✓	✖	✖
Automated removal of cross site scripting vulnerabilities in web applications [1]	✓	✓	✖

ภาพที่ 27 การเปรียบเทียบกับงานวิจัยอื่น ๆ

5. สรุปผลการศึกษาวิจัย

งานวิจัยนี้ได้เสนอวิธีป้องกันการโจมตีครอสไซต์สคริปติ้ง 2 รูปแบบ

1. วิธีป้องกันด้วยการกรองข้อมูลที่บันทึกผ่านหน้าเว็บไซต์ก่อนเข้าฐานข้อมูล สามารถป้องกันการทำงานของคำสั่งที่เป็นการโจมตีครอสไซต์สคริปติ้ง

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

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.

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

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.

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

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

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

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

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

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

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