ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์

ETDA Recommendation on ICT Standard

for Electronic Transactions

ขมธอ. 4 – 2559

ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ

Web Application Security Standard

เวอร์ชั่น 1.0

สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน)

กระทรวงเทคโนโลยีสารสนเทศและการสื่อสาร

ICS 35.030

ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสาร ที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์ ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัย สำหรับโปรแกรมประยุกต์บนเว็บ

ขมธอ. 4 – 2559

สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน)

อาคารเดอะ ไนน์ ทาวเวอร์ แกรนด์ พระรามเก้า (อาคารบี) ชั้น 21
เลขที่ 33/4 ถนนพระราม 9 แขวงห้วยขวาง เขตห้วยขวาง กรุงเทพมหานคร 10210
หมายเลขโทรศัพท์: 0 2123 1234 หมายเลขโทรสาร: 0 2123 1200

ประกาศโดย

สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน)

กระทรวงเทคโนโลยีสารสนเทศและการสื่อสาร

วันที่ 7 กันยายน พ.ศ. 2559

ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อการทำธุรกรรมทางอิเล็กทรอนิกส์ ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บฉบับนี้ จัดทำขึ้นเพื่อเป็นแนวทางสำหรับการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บอย่างมั่นคงปลอดภัย อันจะช่วยลดความเสี่ยงจากการถูกโจมตีผ่านทางช่องโหว่ต่าง ๆ และสามารถจัดการต่อปัญหาที่เกิดขึ้นภายใต้มาตรฐานที่ยอมรับได้ โดยพัฒนาตามแนวมาตรฐานของ

1. คู่มือ “How to Secure Your Website” ของ สำนักงานส่งเสริมเทคโนโลยีสารสนเทศประเทศญี่ปุ่น (Information-Technology Promotion Agency (IPA), Japan)
2. OWASP Testing Guide 4.0 ของ The Open Web Application Security Project

ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์ ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บฉบับนี้ จัดทำโดยสำนักมาตรฐาน ร่วมกับสำนักความมั่นคงปลอดภัย ภายใต้สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน)

สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน)
อาคารเดอะ ไนน์ ทาวเวอร์ แกรนด์ พระรามเก้า (อาคารบี) ชั้น 21 เลขที่ 33/4 ถนนพระราม 9
แขวงห้วยขวาง เขตห้วยขวาง กรุงเทพมหานคร 10310
โทรศัพท์ 0 2123 1234 โทรสาร 0 2123 1200
E-mail: estandard.center@etda.or.th, www.etda.or.th

คำนำ

ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์ ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ (Web Application Security Standard) จัดทำขึ้นโดยสำนักมาตรฐาน และสำนักความมั่นคงปลอดภัย สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน) เพื่อเป็นข้อเสนอแนะและแนวทางการรักษาความมั่นคงปลอดภัยของโปรแกรมประยุกต์บนเว็บ (Web Application) ในการดำเนินการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บอย่างมั่นคงปลอดภัยและสามารถจัดการต่อปัญหาที่เกิดขึ้นภายใต้มาตรฐานที่ยอมรับได้ เนื่องจากภัยคุกคามด้านสารสนเทศในปัจจุบันเกิดขึ้นอย่างหลากหลาย โดยเฉพาะอย่างยิ่งการอาศัยช่องโหว่ที่เกิดจากโปรแกรมประยุกต์บนเว็บ เพื่อโจมตีในลักษณะต่าง ๆ อาทิ เช่น การโจมตีเพื่อเปลี่ยนแปลงข้อมูลหน้าเว็บ (Web Defacement) การโจมตีเพื่อขโมยข้อมูลสำคัญ การโจมตีเพื่อใช้เป็นฐานในการเผยแพร่มัลแวร์ (Malware URL) หรือใช้เป็นฐานในการฉ้อโกงทางการเงินผ่านหน้าเว็บไซต์หลอกลวง (Phishing Website) ซึ่งภัยคุกคามเหล่านี้ล้วนก่อให้เกิดความเสียหายแก่ชื่อเสียงของหน่วยงานที่เป็นเจ้าของเว็บไซต์ ความน่าเชื่อถือของธุรกิจ และยังอาจเป็นอันตรายต่อความมั่นคงปลอดภัยในระดับองค์กรหรือในระดับประเทศ ดังนั้น สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน) ซึ่งให้ความสำคัญต่อการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ จึงได้จัดทำข้อเสนอแนะด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์ฉบับนี้ขึ้นเพื่อเป็นแนวทางสำหรับการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บอย่างมั่นคงปลอดภัย อันจะช่วยลดความเสี่ยงจากการถูกโจมตีผ่านทางช่องโหว่ต่าง ๆ และสามารถจัดการต่อปัญหาที่เกิดขึ้นภายใต้มาตรฐานที่ยอมรับได้

สารบัญ

สารบัญรูป

หน้า

ETDA
สพธอ
www.etda.or.th

ประกาศสำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน)

เรื่อง ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์ ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ

เพื่อเป็นแนวทางสำหรับการรักษาความมั่นคงปลอดภัยของเว็บไซต์ ที่เกี่ยวข้องกับโปรแกรมประยุกต์บนเว็บ (Web application) ลดความเสี่ยงจากการโจมตีโปรแกรมประยุกต์บนเว็บ ให้ผู้ที่เกี่ยวข้องมีวิธีการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บอย่างมั่นคงปลอดภัยและสามารถจัดการกับปัญหาที่เกิดขึ้นภายใต้มาตรฐานที่ยอมรับได้

อาศัยอำนาจตามความในมาตรา ๗(๔) แห่งพระราชกฤษฎีกาจัดตั้งสำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน) พ.ศ. ๒๕๕๔ สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน) จึงประกาศข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์ ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ เลขที่ ขมธอ. ๔ – ๒๕๕๙ ปรากฏตามท้ายประกาศฉบับนี้

ประกาศ ณ วันที่ ๗ กันยายน ๒๕๕๙

(นางสุรางคณา วายุภาพ)
ผู้อำนวยการ
สำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (องค์การมหาชน)

ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์

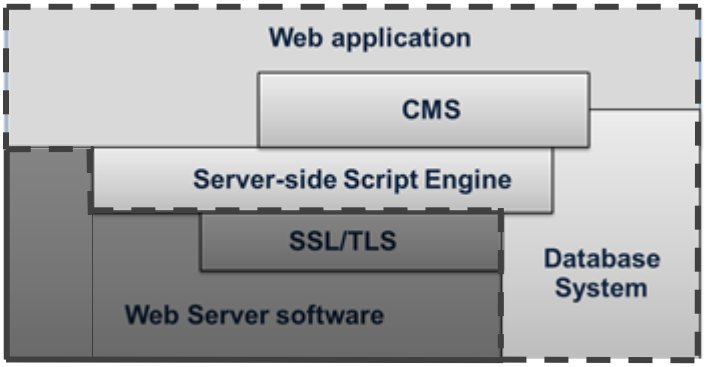
ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ

1. ขอบข่าย

ข้อเสนอแนะมาตรฐานด้านเทคโนโลยีสารสนเทศและการสื่อสารที่จำเป็นต่อธุรกรรมทางอิเล็กทรอนิกส์ ว่าด้วยมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ (Web Application Security Standard) ฉบับนี้ เป็นแนวทางสำหรับการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บเพื่อให้มีความมั่นคงปลอดภัย รวมถึงการเสนอแนะแนวทางที่เกี่ยวข้องเพื่อป้องกันการโจมตีและแก้ไขช่องโหว่ที่เกี่ยวข้อง โดยขอบเขตของข้อเสนอแนะฉบับนี้มุ่งเน้นไปที่การรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บด้วยการพัฒนา และทดสอบโปรแกรมประยุกต์บนเว็บให้มีความมั่นคงปลอดภัยจากช่องโหว่ดังต่อไปนี้

(1) SQL Injection
(2) OS Command Injection
(3) Unchecked Path Parameter หรือ Directory Traversal
(4) Improper Session Management
(5) Cross-Site Scripting
(6) Cross-Site Script Request Forgery
(7) HTTP Header Injection
(8) Mail Header Injection
(9) Lack of Authentication and authorization

อย่างไรก็ตาม มาตรฐานฉบับนี้เป็นข้อเสนอแนะที่จัดทำขึ้น เพื่อลดความเสี่ยงจากการถูกโจมตีผ่านโปรแกรมประยุกต์บนเว็บ (Web Application) อันเนื่องมาจากการที่ผู้พัฒนาอาจขาดความเข้าใจ ขาดประสบการณ์ หรือขาดความตระหนักสำหรับมาตราการด้านความมั่นคงปลอดภัย โดยไม่ได้รวมถึงประเด็นในการแก้ไขหรือรับมือเหตุการณ์โจมตีที่สำเร็จแล้ว เช่น การโจมตีเพื่อเปลี่ยนแปลงข้อมูลหน้าเว็บ (Web Defacement) การโจมตีเพื่อขโมยข้อมูลสำคัญ การโจมตีเพื่อใช้เป็นฐานในการเผยแพร่มัลแวร์ (Malware URL) หรือใช้เป็นฐานในการฉ้อโกงทางการเงินผ่านหน้าเว็บไซต์หลอกลวง (Phishing Website) ทั้งนี้ การทำให้เว็บไซต์มีความมั่นคงปลอดภัยนั้น ยังจำเป็นต้องอาศัยความเข้มแข็งในการบริหารจัดการและการดูแลเว็บไซต์ที่มากเพียงพอ การดำเนินการตามข้อเสนอแนะมาตรฐานนี้ ยังมิได้เป็นสิ่งที่รับรองว่าเว็บไซต์ มีความมั่นคงปลอดภัยโดยสิ้นเชิงจากการโจมตี หรือการบุกรุกระบบ หรือขาดความเข้มแข็งในการบริหารจัดการเว็บไซต์ในทางปฏิบัติ หรือมีภัยคุกคามในรูปแบบที่ไม่เคยเกิดขึ้นมาก่อน (Zero-Day Attack) หรือถูกโจมตี เพื่อเข้ามาในระบบโดยไม่ได้รับอนุญาตหรือโดยมิชอบ (Unauthorized Access) เป็นต้น



รูปที่ 1 ขอบเขตของมาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ (พื้นที่ภายในเส้นประ)

มาตรฐานฉบับนี้มุ่งหมายให้ผู้ที่เกี่ยวข้องมีแนวทางในการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บ ให้มีความมั่นคงปลอดภัย และลดความเสี่ยงในการถูกโจมตีด้วยช่องโหว่สำคัญ ๆ ภายใต้มาตรฐานหรือแนวปฏิบัติอันยอมรับได้ โดยมีผู้พัฒนาโปรแกรมประยุกต์บนเว็บ (Web Application Developer) ซึ่งหน้าที่ในการพัฒนา ปรับปรุงและแก้ไขส่วนประกอบของโปรแกรมประยุกต์บนเว็บให้มีความมั่นคงปลอดภัย

มาตรฐานฉบับนี้อ้างอิงแนวทางที่เกี่ยวข้องกับการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บ จากคู่มือ “How to Secure Your Website” ของ สำนักงานส่งเสริมเทคโนโลยีสารสนเทศ ประเทศญี่ปุ่น (Information-Technology Promotion Agency (IPA), Japan) [1] และ OWASP Testing Guide 4.0 ของ The Open Web Application Security Project [2]

2. บทนิยาม

คำนิยามของศัพท์ที่ใช้กับมาตรฐานฉบับนี้

2.1 เว็บเพจ (Web Page) หมายถึง เอกสารเว็บที่สร้างด้วยภาษา HTML หรือ HyperText Markup Language ซึ่งเป็นภาษามาตรฐานที่ใช้ในการสร้างเว็บเพจ [3]

2.2 เว็บไซต์ (Website) หมายถึง กลุ่มของเว็บเพจที่อยู่บนเครื่องบริการเว็บ (Web Server) ซึ่งมียูอาร์แอลกำกับอยู่ [3]

2.3 ยูอาร์แอล (Universal Resource Locator: URL) หมายถึง ตัวชี้แหล่งในอินเทอร์เน็ตซึ่งประกอบด้วยชื่อโพรโทคอลที่ใช้ในการเข้าถึงข้อมูล (เช่น http://) และ ชื่อโดเมน (เช่น www.etda.or.th) ที่กำกับเครื่องบริการเว็บ [3]

2.4 เวิลด์ไวด์เว็บ (WWW) หมายถึง กลุ่มของเว็บไซต์หรือเครื่องคอมพิวเตอร์ที่มีข้อมูลพร้อมไว้ให้ผู้ใช้บริการเรียกดูข้อมูลผ่านโพรโทคอลเอชทีทีพี (HTTP หรือ Hypertext Transfer Protocol) [3]

2.5 เครื่องบริการเว็บ (Web Server) หมายถึง เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเครื่องบริการ (Server) พร้อมกับโปรแกรมที่ให้บริการข้อมูลเว็บผ่านเครือข่ายเวิลด์ไวด์เว็บ [3]

2.6 โปรแกรมสำหรับให้บริการเว็บ (Web Sever Software) หมายถึง โปรแกรมที่ติดตั้งบนเครื่องบริการ (Server) เพื่อทำให้เครื่องบริการสามารถให้บริการข้อมูลเว็บได้ เช่น โปรแกรม Apache และ โปรแกรม Internet Informaiton Service (IIS) for Windows Server เป็นต้น

2.7 โปรแกรมค้นดูเว็บ (Web Browser) หมายถึง โปรแกรมที่ใช้เรียกข้อมูลเว็บจากเครื่องบริการเว็บผ่านเครือข่ายเวิลด์ไวด์เว็บ [3]

2.8 โปรแกรมประยุกต์บนเว็บ (Web Application) หมายถึง โปรแกรมประยุกต์ที่ถูกพัฒนาขึ้นสำหรับการเรียกใช้งานและเข้าถึงได้โดยโปรแกรมค้นดูเว็บผ่านเครือข่ายคอมพิวเตอร์ เช่น เครือข่ายอินเทอร์เน็ตหรือเครือข่ายอินทราเน็ต เป็นต้น [3]

2.9 ระบบบริหารจัดการเว็บไซต์ (Content Management System: CMS) หมายถึง โปรแกรมที่ผู้ดูแลเครื่องบริการเว็บสามารถใช้ในการดูแลบริหารจัดการเว็บไซต์ผ่านส่วนต่อประสาน (Interface) ซึ่งช่วยให้ง่ายต่อการบริหารจัดการเว็บเพจและการปรับปรุงค่าติดตั้งต่าง ๆ ที่เกี่ยวข้อง [4]

3. การนำไปใช้งาน

ในปัจจุบันหน่วยงานต่าง ๆ ใช้เว็บไซต์เป็นเครื่องมือในการเผยแพร่ประชาสัมพันธ์ข่าวสารและติดต่อกับลูกค้าเพื่อดำเนินกิจกรรมทางธุรกิจ ด้วยเหตุนี้เว็บไซต์จึงเป็นเป้าหมายหนึ่งที่ถูกคุกคามด้านสารสนเทศมากที่สุด โดยเฉพาะเว็บไซต์ที่มีชื่อเสียง หรือเว็บไซต์ที่มีช่องโหว่ มักมีความเสี่ยงที่จะตกเป็นเป้าหมายการโจมตีจากผู้ประสงค์ร้ายอยู่เสมอ โดยอาจเป็นการโจมตีเพื่อเปลี่ยนแปลงข้อมูลหน้าเว็บ (Web Defacement) การโจมตีเพื่อขโมยข้อมูลสำคัญ การโจมตีเพื่อใช้เป็นฐานในการเผยแพร่มัลแวร์ (Malware URL) หรือใช้เป็นฐานในการฉ้อโกงทางการเงินผ่านหน้าเว็บไซต์หลอกลวง (Phishing Website) ภัยคุกคามเหล่านี้ล้วนก่อให้เกิดผลกระทบต่อความน่าเชื่อถือของเว็บไซต์ การโจมตีเว็บไซต์ดังที่กล่าวมาแล้ว ล้วนอาศัยช่องโหว่ของโปรแกรมประยุกต์บนเว็บ (Web Application) เป็นส่วนมาก

แนวทาง (Guidelines) ในมาตรฐานฉบับนี้จัดทำขึ้นสำหรับการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บให้มีความมั่นคงปลอดภัย เสนอแนะแนวทางที่เกี่ยวข้องเพื่อป้องกันการโจมตีและแก้ไขช่องโหว่ที่เกี่ยวข้องกับเว็บไซต์ แบ่งออกเป็นสองกลุ่ม ซึ่งได้แก่ การพัฒนาโปรแกรมประยุกต์บนเว็บอย่างมั่นคงปลอดภัย (Web Application Security Implementation) และการทดสอบความมั่นคงปลอดภัยของโปรแกรมประยุกต์บนเว็บ (Web Application Security Testing) โดยเนื้อหาแต่ละบทจะกล่าวถึง ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats) และแนวทางเพื่อป้องกันการโจมตี ซึ่งแนวทางนั้นจะแบ่งเป็น แนวทางในการป้องกันการโจมตี (Fundamental Solutions) และแนวทางในการลดความเสียหายจากการถูกโจมตี (Mitigation Measures) ประกอบด้วยแนวทางในการพัฒนาโปรแกรมประยุกต์ให้มีความมั่นคงปลอดภัยจากช่องโหว่ดังต่อไปนี้

(1) SQL Injection
(2) OS Command Injection
(3) Unchecked Path Parameter หรือ Directory Traversal
(4) Improper Session Management
(5) Cross-Site Scripting
(6) Cross-Site Script Request Forgery
(7) HTTP Header Injection

(8) Mail Header Injection

(9) Lack of Authentication and authorization

มาตรฐานฉบับนี้เป็นแนวทางสำหรับการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ เพื่อลดความเสี่ยงจากการโจมตีเว็บไซต์ และทำให้ผู้ที่เกี่ยวข้องมีแนวทางในการพัฒนาและทดสอบโปรแกรมประยุกต์บนเว็บให้มีความมั่นคงปลอดภัย โดยผู้พัฒนาโปรแกรมประยุกต์บนเว็บ สามารถใช้ตัวอย่างของแบบประเมินเพื่อตรวจสอบความมั่นคงปลอดภัยของการพัฒนาโปรแกรมประยุกต์บนเว็บที่อยู่ในภาคผนวก ก.1 มาประยุกต์ใช้ได้กับการพัฒนาโปรแกรมประยุกต์บนเว็บทั่วไป ตามแนวทางในมาตรฐานฉบับนี้ เมื่อพบรายการที่ไม่เป็นไปตามแนวทาง สามารถใช้ภาคผนวก ก.2 แบบฟอร์มสำหรับการแก้ไขรายการที่ยังต้องปรับปรุง (จากการตรวจสอบสถานะความมั่นคงปลอดภัย) เพื่อวางแผนการปรับปรุงได้ นอกจากนี้ผู้พัฒนาโปรแกรมประยุกต์บนเว็บยังสามารถใช้มาตรฐานฉบับนี้เพื่อแสดงความสอดคล้องกับแนวทางที่เกี่ยวข้องโดย

(1) ประเมินตนเอง (Self-Assessment) และประกาศการรับรองตนเอง (Self-Declaration) ว่าได้มีการดำเนินการตามข้อเสนอแนะฉบับนี้
(2) ยืนยันถึงความสอดคล้องกับมาตรฐานจากผู้มีส่วนได้ส่วนเสียกับเว็บไซต์
(3) ยืนยันถึงการประกาศรับรองตนเองจากหน่วยงานภายนอก
(4) ขอรับการรับรอง (Certification) มาตรฐานการรักษาความมั่นคงปลอดภัยสำหรับผู้ดูแลและพัฒนาเว็บไซต์จากหน่วยตรวจสอบและรับรอง (Conformity Assessment Body)

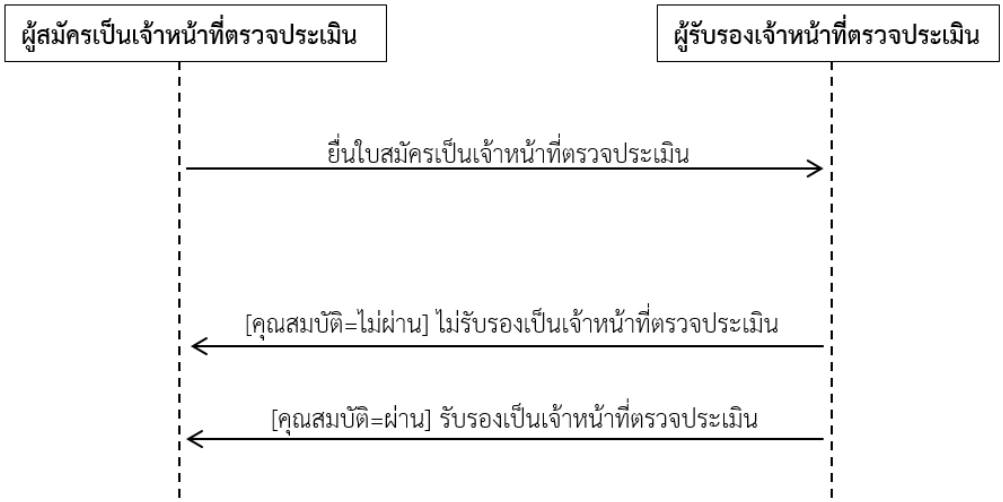
การตรวจประเมินการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บนั้น เจ้าหน้าที่ตรวจประเมินควรได้รับการรับรองคุณสมบัติ ความรู้ ความสามารถ และทักษะที่เกี่ยวข้องกับการตรวจประเมินดังกล่าว ก่อนที่จะเข้าสู่ขั้นตอนการตรวจประเมิน ดังนั้น มาตรฐานฉบับนี้ จึงเสนอแนวทางที่เกี่ยวข้อง 2 แนวทาง ได้แก่ แนวทางการรับรองเจ้าหน้าที่ตรวจประเมิน (ดังรูปที่ 2) และแนวทางที่เกี่ยวข้องกับการตรวจประเมิน (ดังรูปที่ 3)

ผู้ที่เกี่ยวข้องกับการตรวจประเมิน ได้แก่

(1) ผู้เข้ารับการตรวจประเมิน ซึ่งเป็นผู้ที่มีหน้าที่เกี่ยวข้องกับการพัฒนาโปรแกรมประยุกต์บนเว็บ
(2) เจ้าหน้าที่ตรวจประเมิน ซึ่งเป็นผู้ที่ทำหน้าที่ตรวจประเมินโปรแกรมประยุกต์บนเว็บให้เป็นไปตามแนวทางที่กำหนด
(3) ผู้รับรองเจ้าหน้าที่ตรวจประเมิน ซึ่งเป็นผู้ที่มีหน้าที่รับรองเจ้าหน้าที่ตรวจประเมิน
(4) หน่วยงานรับรองผลการตรวจประเมิน ซึ่งเป็นผู้ที่มีหน้าที่รับรองผลการตรวจประเมิน

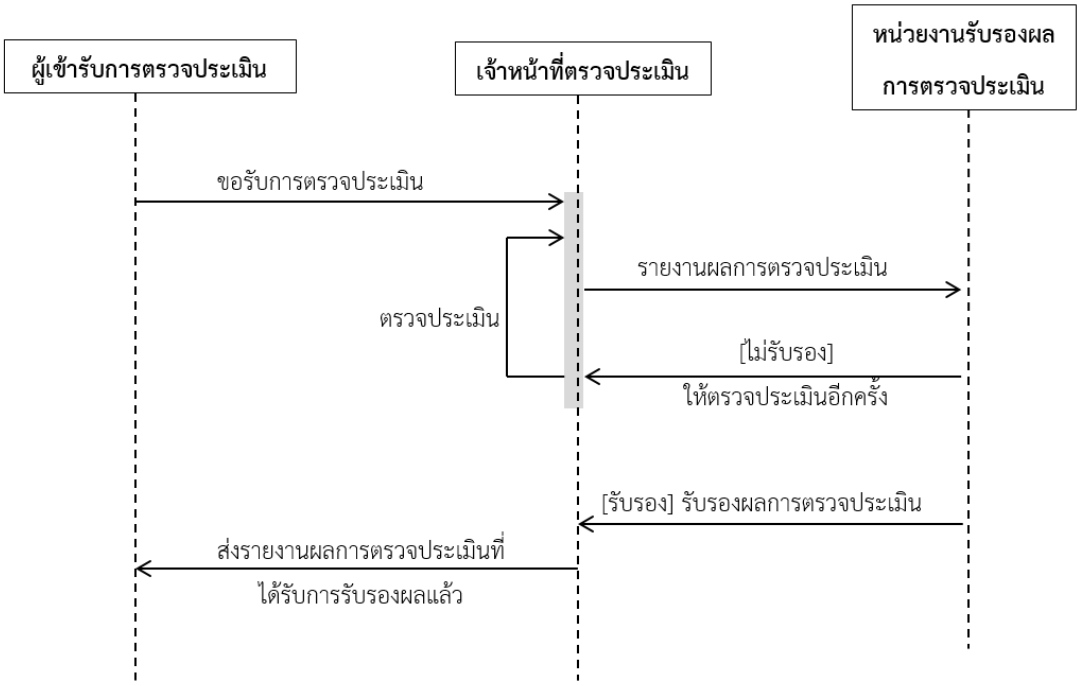
สำหรับขั้นตอนการรับรองเจ้าหน้าที่ตรวจประเมิน เริ่มจากผู้สมัครเป็นเจ้าหน้าที่ตรวจประเมินยื่นใบสมัครเป็นเจ้าหน้าที่ตรวจประเมิน ไปยังผู้รับรองเจ้าหน้าที่ตรวจประเมิน หากตรวจสอบคุณสมบัติผ่าน จะได้รับการรับรองเป็นเจ้าหน้าที่ตรวจประเมิน แต่หากตรวจสอบคุณสมบัติไม่ผ่าน จะไม่ได้รับการรับรองเป็นเจ้าหน้าที่ตรวจประเมิน (ดังรูปที่ 2)

ขั้นตอนการรับรองเจ้าหน้าที่ตรวจประเมิน



รูปที่ 2 ขั้นตอนการรับรองเจ้าหน้าที่ตรวจประเมิน

ขั้นตอนและบทบาทที่เกี่ยวข้องกับการตรวจประเมิน



รูปที่ 3 ขั้นตอนและบทบาทที่เกี่ยวข้องกับการตรวจประเมิน

ส่วนขั้นตอนและบทบาทที่เกี่ยวข้องกับการตรวจประเมิน เริ่มจากผู้เข้ารับการตรวจประเมิน ขอรับการตรวจประเมินจากเจ้าหน้าที่ตรวจประเมิน เจ้าหน้าที่ตรวจประเมินทำการตรวจประเมินและสรุปรายงานผลการตรวจส่งไปให้หน่วยงานรับรองพิจารณารับรองรายงานดังกล่าว หากพิจารณาแล้วมีความถูกต้องครบถ้วนจะรับรองผลการตรวจประเมินและส่งผลการรับรองกลับไปให้ผู้เข้ารับการตรวจประเมิน แต่หากพิจารณาแล้วไม่ถูกต้องครบถ้วนจะไม่รับรองผลการตรวจประเมิน และแจ้งให้เจ้าหน้าที่ตรวจประเมินจะทำการตรวจประเมินอีกครั้ง (ดังรูปที่ 3)

แนวทางในมาตรฐานฉบับนี้สามารถนำไปใช้ได้กับเว็บไซต์ทั่วไปที่อยู่บนเครื่องบริการเว็บส่วนตัว (Private Web Server) หรือ เว็บไซต์ที่ใช้บริการกับผู้ให้บริการเว็บโฮสติ้ง หรือ เว็บไซต์ที่ใช้บริการระบบ Cloud

4. แนวทางในการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บ

จากสถิติภัยคุกคาม ในปี พ.ศ. 2557-2558 จากศูนย์ประสานการรักษาความมั่นคงปลอดภัยระบบคอมพิวเตอร์ประเทศไทย หรือ ไทยเซิร์ต พบว่าประเภทของภัยคุกคามที่พบมากที่สุด 3 อันดับแรก ได้แก่ 1. Malicious Code 2. Fraud 3. Intrusion ซึ่งช่องโหว่ดังกล่าวเป็นช่องโหว่ที่เกิดจากโปรแกรมประยุกต์บนเว็บ เนื่องจากบ่อยครั้งที่การรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บมักจะถูกมองข้ามทั้งที่เป็นเรื่องที่สำคัญ

Open Web Application Security Project หรือ OWASP ซึ่งเป็นองค์กรจัดตั้งขึ้นเพื่อส่งเสริมและพัฒนาการรักษาความมั่นคงปลอดภัยของเว็บไซต์หรือโปรแกรมประยุกต์บนเว็บได้มีการจัดทำโครงการจัด 10 อันดับความเสี่ยงของโปรแกรมประยุกต์บนเว็บเป็นประจำทุกปี ซึ่งในปี 2013 นั้น มีการจัดอันดับความเสี่ยงที่พบสูงสุด 10 อันดับ (รายละเอียดเพิ่มเติมจากเว็บไซต์ https://www.owasp.org) ซึ่งสอดคล้องกับความเสี่ยงที่ไทยเซิร์ต ตรวจพบในเว็บไซต์ที่มีอยู่ในประเทศไทย โดยช่องโหว่หรือความเสี่ยงที่เกี่ยวกับโปรแกรมประยุกต์บนเว็บที่พบส่วนใหญ่จะเป็นเรื่องการโจมตีจากเทคนิคต่าง ๆ เช่น SQL Injection, Session Hijacking และ CSRF เป็นต้น

จากสถิติของภัยคุกคามที่เกิดกับเว็บไซต์หรือโปรแกรมประยุกต์บนเว็บดังที่ได้กล่าวไว้ข้างต้น จะเห็นได้ว่าการป้องกันหรือการลดความเสี่ยงจากภัยคุกคามที่จะเกิดกับโปรแกรมประยุกต์บนเว็บ จำเป็นต้องคำนึงถึงความมั่นคงปลอดภัย ตั้งแต่เริ่มกระบวนการพัฒนารวมถึงการหมั่นทดสอบความมั่นคงปลอดภัยของโปรแกรมประยุกต์บนเว็บควบคู่กันไป ดังนั้น มาตรฐานฉบับนี้จึงมีแนวทางในการรักษาความมั่นคงปลอดภัยสำหรับโปรแกรมประยุกต์บนเว็บแบ่งเป็น 2 ส่วน คือ การพัฒนาโปรแกรมประยุกต์บนเว็บอย่างมั่นคงปลอดภัย (Web Application Security Implementation) และการทดสอบความมั่นคงปลอดภัยของโปรแกรมประยุกต์บนเว็บ (Web Application Security Testing) ประกอบด้วยแนวทางในการพัฒนาโปรแกรมประยุกต์ให้มีความมั่นคงปลอดภัยจากช่องโหว่ ซึ่งมีรายละเอียดตามหัวข้อต่อไปนี้

4.1 SQL Injection

เว็บไซต์ส่วนใหญ่ ใช้คำสั่ง SQL ในการเชื่อมต่อกับฐานข้อมูล ผู้ประสงค์ร้ายสามารถแทรกคำสั่ง SQL ผ่านพารามิเตอร์ต่าง ๆ ของเว็บไซต์ เช่น GET POST เป็นต้น ทำให้สามารถดำเนินการใด ๆ ก็ตามกับในฐานข้อมูลได้ โดยหากการโจมตีด้วยเทคนิคนี้สำเร็จ จะทำให้ผู้ประสงค์ร้ายเข้าถึง แก้ไขเปลี่ยนแปลง หรือลบข้อมูลในฐานข้อมูลได้ หรือ สั่งคำสั่งปิดฐานข้อมูล ซึ่งเว็บไซต์ใดก็ตามที่เชื่อมต่อกับฐานข้อมูลโดยตรง หรือมีการเรียกฐานข้อมูลทุกครั้งที่เรียกหน้าเว็บเพจ ก็จะต้องระมัดระวังช่องโหว่นี้เป็นพิเศษ (ฐานข้อมูลที่ใช้กันโดยทั่วไป ได้แก่ MySQL, PostgreSQL, Oracle, Microsoft SQL Server และ DB2) ซึ่งช่องโหว่นี้เรียกว่า “SQL Injection vulnerability” และการโจมตีที่อาศัยช่องโหว่นี้เรียกว่า “SQL Injection Attack” แต่ใน

ปัจจุบันมีฐานข้อมูลประเภท NoSQL ซึ่งไม่ได้ใช้คำสั่ง SQL ในการเข้าถึงข้อมูลในฐานข้อมูล ทำให้ปลอดภัยจากการโจมตีด้วยเทคนิคนี้มากขึ้น

4.1.1 ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats)

ผู้ประสงค์ร้ายอาศัยช่องโหว่นี้ในการดำเนินการ ดังนี้

(1) เรียกดูข้อมูลลับ (Sensitive Data) ต่าง ๆ ที่เก็บในฐานข้อมูล เช่น การเปิดเผยข้อมูลส่วนตัวของผู้ใช้บริการ
(2) เพิ่ม เปลี่ยนแปลง ลบข้อมูลใด ๆ ที่เก็บในฐานข้อมูลได้ เช่น เปลี่ยนแปลงข้อมูลหน้าเว็บเพจ เปลี่ยนรหัสผ่านของผู้ใช้บริการ หรือการใช้คำสั่งปิดฐานข้อมูล
(3) หลีกเลี่ยงการยืนยันตัวตนจากระบบ Login (Login Authentication Bypass) เช่น สามารถดำเนินการคำสั่งใด ๆ ภายใต้บัญชีผู้ใช้ที่ไม่ได้รับอนุญาตได้ โดยไม่ผ่านกระบวนการยืนยัน
(4) อัปโหลดข้อมูลใด ๆ ขึ้นไปยังเว็บไซต์ หรือฐานข้อมูลได้
(5) ส่งคำสั่งโดยตรงต่อระบบปฏิบัติการได้ เช่น การสั่งให้เครื่องบริการเว็บ เปิดหรือปิด Firewall ของเครื่องบริการเว็บนั้น

4.1.2 แนวทางในการป้องกันการโจมตี

มาตรฐานฉบับนี้จึงมีแนวทางในการป้องกันปัญหาดังกล่าว ดังนี้

4.1.2.1 Fundamental Solutions : การป้องกันการโจมตี

(1) โปรแกรมประยุกต์บนเว็บต้องมีการจัดทำ Prepared Statement และ/หรือ Stored Procedure

โปรแกรมประยุกต์บนเว็บต้องมีการจัดทำ Prepared Statement และ/หรือ Stored Procedure เป็นวิธีการที่จะแยกคำสั่งในการประมวลผลและค่าที่จะนำไปประมวลผลออกจากกัน จากวิธีการดังกล่าวจะช่วยป้องกันการโจมตีด้วยวิธีการทำ SQL Injection ได้ ซึ่งสามารถศึกษารายละเอียดเพิ่มเติมเรื่อง Stored Procedure ได้ที่ https://www.owasp.org/index.php/Guide_to_SQL_Injection และ https://www.owasp.org/index.php/Avoiding_SQL_Injection#Parameterized_Stored_Procedures

(2) ไม่ควรเขียนคำสั่ง SQL โดยตรงในตัวแปร (Parameter) ที่ส่งโดยตรงไปยังโปรแกรมประยุกต์บนเว็บ

การเขียนคำสั่ง SQL โดยตรงในตัวแปร จะนำไปสู่ความเสี่ยงที่ผู้ประสงค์ร้ายจะสามารถปลอมแปลงค่าตัวแปร และกระทำการใด ๆ โดยตรงกับฐานข้อมูลได้

4.1.2.2 Mitigation Measures : การลดความเสียหายที่เกิดจากการถูกโจมตี

(1) ควบคุมการแสดงผลข้อมูล Error Message

ถ้า Error Message แสดงข้อมูลที่เกี่ยวข้องกับฐานข้อมูล เช่น ชื่อของฐานข้อมูล ตารางของฐานข้อมูล หรือคำสั่ง SQL ที่เป็นสาเหตุของการเกิดข้อผิดพลาด ทำให้ผู้ประสงค์ร้ายสามารถนำข้อมูลเหล่านี้ไปใช้ประโยชน์ในการโจมตีเว็บไซต์ในอนาคตได้

(2) กำหนดสิทธิขั้นต่ำให้บัญชีผู้ใช้ของฐานข้อมูล

หากกำหนดสิทธิของบัญชีผู้ใช้ที่สามารถเข้าถึงฐานข้อมูลมากเกินจำเป็น จะทำให้ความเสียหายที่เกิดจากการถูกโจมตีมีมากขึ้น ดังนั้นควรกำหนดสิทธิของบัญชีผู้ใช้ที่เข้าถึงฐานข้อมูลให้น้อยที่สุดที่เพียงพอกับการใช้งาน

4.1.3 การทดสอบความมั่นคงปลอดภัย

โดยปกติแล้วการพัฒนาโปรแกรมประยุกต์จะสร้างคำสั่ง SQL จาก SQL Syntax ซึ่งจะมีการรับค่าที่ได้จากโปรแกรมค้นดูเว็บ (Web Browser) มาใช้ในคำสั่ง SQL เช่น Select * from users where id=$id; ซึ่งจากตัวอย่างข้างต้น ตัวแปร $id นั้นจะรับข้อมูลจากโปรแกรมค้นดูเว็บฝั่งผู้ใช้บริการ แต่ส่วนที่เหลือเป็น static ที่เขียนโดยผู้พัฒนาโปรแกรมประยุกต์บนเว็บ

ในกรณีปกติ ผู้ใช้บริการส่ง $id=10 คำสั่ง SQL จะเป็น “Select * from users where id=10;” ระบบก็จะคืนค่าข้อมูลที่มี id=10 ออกมา ซึ่งผู้ประสงค์ร้ายสามารถเปลี่ยนแปลง SQL โดยเพิ่ม “or 1=1” ที่ประโยค where ได้ เป็น “Select * from users where id=10 or 1=1”; ซึ่งเมื่อนำคำสั่งไปประมวลผลจะเป็นจริงในทุกกรณี โปรแกรมประยุกต์ก็จะแสดงข้อมูลที่ดึงจากฐานข้อมูลออกมาโดยการทดสอบนั้น พยายามป้อนค่าทดสอบเข้าไปเพื่อค้นหาข้อผิดพลาด

เทคนิคที่ใช้ในการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่ Testing for SQL Injection (OTG-INPVAL-005) : https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)

4.2 OS Command Injection

หากผู้พัฒนาโปรแกรมประยุกต์มีการใช้คำสั่ง (OS Command) เช่น exec(), passthru(), shell_exec(), system(), popen() ร่วมกับการพัฒนาโปรแกรมประยุกต์บนเว็บ ก็จะเป็นช่วงโหว่ที่ทำให้ผู้ประสงค์ร้าย โจมตีผ่านคำสั่งระดับระบบปฏิบัติการ (OS Command) เพื่อสั่งดำเนินการใด ๆ ผ่านโปรแกรมประยุกต์บนเว็บที่มีช่องโหว่ได้ ซึ่งจะนำไปสู่การรั่วไหลของข้อมูลสำคัญ เข้าควบคุมเครื่องบริการเว็บ หรือใช้เป็นฐานเพื่อโจมตีเครื่องบริการเว็บอื่น ๆ

เว็บไซต์ใดก็ตามที่ใช้คำสั่ง (OS Command) เหล่านี้ เช่น exec(), passthru(), shell_exec(), system(), popen() ในภาษา PHP ร่วมกับการพัฒนาโปรแกรมประยุกต์ ก็จะเป็นช่วงโหว่ที่ทำให้ผู้ประสงค์ร้าย โจมตีโปรแกรมประยุกต์บนเว็บ โดยสั่ง Run คำสั่ง OS Command เพื่อสั่งดำเนินการใด ๆ กับระบบผ่านโปรแกรมประยุกต์บนเว็บได้ ซึ่งช่องโหว่นี้เรียกว่า “OS Command Injection Vulnerability” และการโจมตีที่อาศัยช่องโหว่นี้เรียกว่า “OS Command Injection Attack”

4.2.1 ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats)

ผู้ประสงค์ร้ายอาศัยช่องโหว่นี้ในการดำเนินการ ดังนี้

(1) เรียกดูข้อมูลลับ (Sensitive Data) หรือลบไฟล์ต่าง ๆ ที่เก็บในเครื่องบริการเว็บได้
(2) เข้าจัดการและควบคุมเครื่องบริการเว็บ เพื่อใช้เป็นฐานโจมตีเครื่องบริการเว็บอื่น ๆ
(3) ดาวน์โหลดและสั่งดำเนินการโปรแกรมหรือสคริปต์อันตราย

4.2.2 แนวทางในการป้องกันการโจมตี

4.2.2.1 Fundamental Solutions : การป้องกันการโจมตี

พัฒนาโปรแกรมประยุกต์บนเว็บโดยให้ปิดการใช้งานคำสั่งต่าง ๆ เพื่อป้องกันการเรียกใช้ที่ไม่พึงประสงค์ เช่น ผู้ประสงค์ร้ายอัพโหลดไฟล์ Backdoor เป็นต้น

4.2.2.2 Mitigation Measures : วิธีการลดความเสียหายที่เกิดจากการถูกโจมตี

หากจำเป็นต้องพัฒนาฟังก์ชันในโปรแกรมประยุกต์โดยใช้คำสั่ง OS Command ผู้พัฒนาโปรแกรมประยุกต์ต้องตรวจสอบ Variables ที่จะใช้กับตัวแปรของ OS Command ก่อนนำไปประมวลผลและต้องให้มั่นใจได้ว่า Execute แค่ค่านั้นเท่านั้น (ถ้าเราจะใช้ OS Command จะต้อง Clean ก่อนโดยใช้ฟังก์ชันเข้าช่วย)

4.2.3 การทดสอบความมั่นคงปลอดภัย

OS Command Injection เป็นการโจมตีที่ดำเนินการผ่านหน้าเว็บไซต์ เพื่อสั่งดำเนินการใด ๆ กับเครื่องบริการเว็บได้ เช่น ผู้ประสงค์ร้ายสามารถอัพโหลด โปรแกรมอันตรายหรือดักเอารหัสผ่านได้

วิธีการทดสอบ เบื้องต้น เช่นในภาษา PHP สามารถใช้ Semicolon (;) เพื่อระบุว่าจบส่วนที่เป็น URL และตามด้วยคำสั่ง OS Command ได้ โดยคำสั่ง ‘%3B’ คือรหัสของ Semicolon ที่เข้ารหัสเรียบร้อยแล้ว

เทคนิคที่ใช้ในการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่ Testing for Command Injection (OTG-INPVAL-013) :
https://www.owasp.org/index.php/Testing_for_Command_Injection_%28OTG-INPVAL-013%29

4.3 Unchecked Path Parameter / Directory Traversal

ผู้ประสงค์ร้ายเรียกชื่อไฟล์หรือข้อมูลที่เก็บบนเครื่องบริการเว็บได้โดยตรง ผ่าน External Parameter เช่น ใส่คำสั่ง ?file=../../secret.txt ต่อท้าย URL ของเว็บไซต์ ก็จะเป็นการระบุชื่อไฟล์ที่ต้องการพร้อม Path ไฟล์ ทำให้ผู้ประสงค์ร้าย สามารถเข้าถึงไฟล์ที่อยู่บนเครื่องบริการเว็บโดยไม่ได้รับอนุญาตได้ ซึ่งช่องโหว่นี้เรียกว่า “Directory Traversal Vulnerability” และการโจมตีในลักษณะนี้ เรียกว่า “Directory Traversal Attack”

ไม่ว่าเว็บไซต์ประเภทใดก็อาจตกเป็นเหยื่อได้ ถ้าโปรแกรมประยุกต์บนเว็บอนุญาตให้ระบุชื่อไฟล์ได้ด้วย External Parameter และถ้าเครื่องบริการเว็บของท่านเก็บข้อมูลที่เป็นความลับ เช่น ข้อมูลส่วนบุคคล ก็มีความเสี่ยงที่จะถูกผู้ประสงค์ร้ายเรียกเอาข้อมูลดังกล่าวไปได้

4.3.1 ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats)

ผู้ประสงค์ร้ายอาศัยช่องโหว่นี้ในการดูหรือเข้าไปจัดการหรือลบไฟล์ในเครื่องบริการเว็บหรือฐานข้อมูลได้

4.3.2 แนวทางในการป้องกันการโจมตี

4.3.2.1 Fundamental Solutions : การป้องกันการโจมตี

(1) ไม่อนุญาตให้ใส่ Filename เพื่อระบุถึงข้อมูลได้จาก External Parameter

เมื่อโปรแกรมประยุกต์บนเว็บอนุญาตให้ใส่ Filename เพื่อระบุถึงข้อมูลได้จาก External Parameter ผู้ประสงค์ร้ายสามารถเข้าถึงและดำเนินการใด ๆ ได้อย่างง่ายดาย และสามารถดูข้อมูลทั้งหมดที่ไม่ควรเปิดเผยแก่บุคคลภายนอก

(2) ใช้ Fixed Directory ในการจัดการระบุชื่อไฟล์

4.3.2.2 Mitigation Measures : วิธีการลดความเสียหายที่เกิดจากการถูกโจมตี

(1) กำหนด Permission การเข้าถึงไฟล์บนเครื่องบริการเว็บให้เหมาะสม

ต้องกำหนด Permission การเข้าถึงไฟล์บนเครื่องบริการเว็บให้เหมาะสม และเครื่องบริการเว็บควรป้องกันการโจมตีได้เมื่อโปรแกรมประยุกต์บนเว็บใด ๆ พยายามเรียกไฟล์ในเครื่องบริการเว็บโดยไม่ได้รับอนุญาต

(2) ตรวจสอบ Filename เช่น มีการแปล String ที่ระบุ Directory ได้

ต้องมีการแปล String ที่ระบุ Directory ได้ เช่น / -> % 2F ก่อน

โดยแนวทางการพัฒนาโปรแกรมประยุกต์บนเว็บให้มีความมั่นคงปลอดภัยจากการโจมตีด้วย Abusing Path Parameters สามารถศึกษาข้อมูลเพิ่มเติมได้ที่ https://www.owasp.org/index.php/Path_Traversal

4.3.3 การทดสอบความมั่นคงปลอดภัย

การดำเนินการต่าง ๆ ของโปรแกรมประยุกต์บนเว็บ เกี่ยวข้องกับการเรียกใช้และจัดการไฟล์ หากไม่ได้ตรวจสอบค่าที่รับจากฝั่งผู้ใช้บริการ ผู้ประสงค์ร้ายก็จะสามารถเข้าถึงไฟล์ข้อมูลที่ไม่ได้รับอนุญาตได้ โดยการกำหนดสิทธิในการเข้าถึงข้อมูลจะกำหนดสิทธิในการเข้าถึงข้อมูลบนเครื่องบริการเว็บ และสิทธิในการ แก้ไขเปลี่ยนแปลง หรือดำเนินการใด ๆ กับข้อมูล โดยจะต้องเข้าถึงได้เฉพาะข้อมูลที่ได้รับอนุญาตบนเครื่องบริการเว็บเท่านั้น

ซึ่งการออกแบบเรื่องการกำหนดสิทธิข้อมูลนี้ เพื่อป้องกันไม่ให้ผู้ประสงค์ร้ายเข้าถึงไฟล์หรือข้อมูลที่เป็นความลับ (เช่น ไฟล์ ../etc/passwd) หรือเพื่อหลีกเลี่ยงการโจมตีจากการ Run คำสั่ง OS Command

เทคนิคที่ใช้ในการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่ Directory Traversal Testing (OTG-AUTHZ-001) : https://www.owasp.org/index.php?title=Testing_Directory_traversal/file_include_%28OTG-AUTHZ-001%29

4.4 Improper Session Management

ในการเข้าถึงเว็บไซต์ใด ๆ ของผู้ใช้บริการ วิธีในการตรวจสอบและยืนยันตัวตนของผู้ใช้บริการที่หลายเว็บไซต์นิยมใช้ คือ สร้าง Session ID ขึ้นหลังจากกระบวนการยืนยันตัวตนของผู้ใช้บริการสำเร็จ โดย Session ID นี้จะถูกนำไปใช้ในการอ้างอิงและตรวจสอบสิทธิในการเข้าถึงหน้าเว็บเพจต่าง ๆ ที่ผู้ใช้บริการเข้าเยี่ยมชม

Session ID นี้จะถูกใช้ จนกว่าผู้ใช้บริการจะปิดหน้าต่างโปรแกรมค้นดูเว็บจึงจะลบ Session ID นั้นไป และจะไม่สามารถใช้ Session ID เดิมในการอ้างอิงได้อีก จากลักษณะการทำงานข้างต้น ทำให้เครื่องบริการเว็บสามารถติดตามข้อมูลทางฝั่งผู้ใช้บริการได้ตลอดตราบเท่าที่โปรแกรมค้นดูเว็บยังไม่ถูกปิด ทำให้ ผู้ประสงค์ร้ายสามารถอาศัยช่องโหว่นี้ในการโจมตีเว็บไซต์ด้วยวิธี Session Hijack ได้ [5]

หากเว็บไซต์ของท่านมีการให้บริการที่เกี่ยวกับข้อมูลที่เป็นความลับและมีความสำคัญสูง เช่น เว็บไซต์ที่ให้บริการการชำระเงินทางอิเล็กทรอนิกส์ (เช่น Internet Banking, Online Trading หรือ e-Commerce เป็นต้น) เว็บไซต์ที่มีข้อมูลส่วนตัว หรือข้อมูลที่เป็นความลับ เปิดเผยต่อบุคคลที่สามไม่ได้เด็ดขาด (เช่น Job-Hunting Website, Internal Community Website หรือ Webmails เป็นต้น) หรือเว็บไซต์ที่มีระบบการยืนยันตัวตน เพื่อเข้าใช้งานระบบ (เช่น การยืนยันตัวตนเพื่อเข้าใช้งานในฐานะผู้ดูแลระบบ หรือ เว็บไซต์ที่ให้เฉพาะสมาชิกเข้าได้เท่านั้น เป็นต้น) จะต้องระมัดระวังและป้องกันเป็นพิเศษ

4.4.1 ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats)

ผู้ประสงค์ร้ายอาศัยช่องโหว่นี้ในการในการดำเนินการ โดยใช้ประโยชน์จากการจัดการ Session ที่ไม่เหมาะสม คือสามารถ ดักขโมย Session ID ของผู้ใช้บริการ หรือนำเอา Session ID ไปใช้ในการเข้าเว็บไซต์ด้วยสิทธิของเจ้าของ Session ได้ ยกตัวอย่างเช่น

(1) เข้าถึงและดำเนินการกับบริการต่าง ๆ ที่ผู้ใช้บริการมีสิทธิ เช่น สั่งโอนเงิน (Internet Banking Service), สั่งซื้อของออนไลน์ เป็นต้น
(2) เพิ่ม แก้ไข ลบ ข้อมูลส่วนตัวต่าง ๆ ของผู้ใช้บริการ เช่น เปลี่ยนรหัสผ่าน
(3) เรียกดูข้อมูลใด ๆ ตามสิทธิของผู้ใช้บริการได้ เช่น เปิดดูข้อมูลในอีเมลส่วนตัว

4.4.2 แนวทางในการป้องกันการโจมตี

มาตรฐานฉบับนี้มีแนวทางในการป้องกันปัญหาดังกล่าว ดังนี้

4.4.2.1 Fundamental Solutions : การป้องกันการโจมตี

(1) สร้าง Session ID เป็นที่ยากต่อการคาดเดา ไม่ใช้ Algorithm ที่ง่ายเกินไป เช่น Pseudo Random Number
(2) ไม่ใช้ URL Parameter ในการเก็บ Session ID
(3) ให้มีการใช้งาน HTTPS Protocol ใช้ Secure Attribute ของ Cookies

4.4.2.2 Mitigation Measures : วิธีการลดความเสียหายที่เกิดจากการถูกโจมตี

(1) ใช้ Session ID เป็นค่าสุ่ม
(2) มีการกำหนดวันหมดอายุของ Cookies ที่เก็บ Session ID

โดย แนวทางการพัฒนาโปรแกรมประยุกต์บนเว็บให้มีความมั่นคงปลอดภัยจากการโจมตีด้วย Improper Session Management สามารถศึกษาข้อมูลเพิ่มเติมได้ที่ https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

4.4.3 การทดสอบความมั่นคงปลอดภัย

ผู้ทดสอบสามารถตรวจสอบว่า Cookies ที่ให้ผู้ใช้บริการสามารถป้องกันการโจมตีได้หลากหลายแค่ไหน ซึ่งเป้าหมายโดยรวมคือเพื่อให้ได้ Cookies ที่มีช่องโหว่ที่เกิดจากการโจมตี เช่น Session Hijacking อนุญาตให้ผู้ประสงค์ร้ายสามารถแก้ไข หรือเปลี่ยนแปลงสิทธิของผู้ใช้บริการในระบบได้ เป็นต้น

โดยปกติแล้ว รูปแบบในการโจมตีจะมีดังนี้

(1) ดักเก็บ Cookies เพื่อให้ได้ Cookies ที่เป็นไฟล์ตัวอย่าง
(2) วิเคราะห์ขั้นตอนและ Algorithm วิธีการสร้าง Cookies จากตัวอย่างที่ได้สร้างไฟล์ Cookies ปลอมขึ้นเพื่อใช้ในการโจมตี

เทคนิคการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่

(1) Testing for Bypassing Session Management Schema (OTG-SESS-001) : https://www.owasp.org/index.php/Testing_for_Session_Management_Schema_%28OTG-SESS-001%29
(2) Testing for Exposed Session Variables (OTG-SESS-004) : https://www.owasp.org/index.php/Testing_for_Exposed_Session_Variables_%28OTG-SESS-004%29

4.5 Cross-Site Scripting

Cross-Site Scripting (XSS) [6] เกิดจากช่องโหว่ของโปรแกรมประยุกต์บนเว็บที่ไม่มีการคัดกรองและตรวจสอบข้อมูลที่ได้รับจากผู้ใช้บริการว่าเป็นข้อมูลที่เชื่อถือได้หรือไม่ ทำให้ผู้ประสงค์ร้ายสามารถแทรกคำสั่งต่าง ๆ เข้าไปในเว็บเพจ เมื่อผู้ใช้บริการเรียกหน้าเว็บเพจนั้น ก็อาจจะถูกขโมยข้อมูลสำคัญไปได้ ซึ่งผู้ประสงค์ร้ายอาจจะนำไปสวมรอยและเข้าสู่ระบบไปยังเว็บไซต์เสมือนหนึ่งว่าเป็นผู้ใช้บริการตัวจริง ปัญหานี้เรียกว่า "Cross-Site Scripting Vulnerability" และการโจมตีที่ใช้ประโยชน์จากช่องโหว่นี้ เรียกว่า "Cross-Site Scripting Attack" ซึ่งอาจจะไม่เป็นอันตรายต่อเว็บไซต์ แต่จะส่งผลกระทบโดยตรงต่อความมั่นคงปลอดภัยของผู้ใช้บริการ

ทุกเว็บไซต์ควรระมัดระวังช่องโหว่ประเภทนี้ โดยเฉพาะเว็บไซต์ที่มีการจัดการ Session ID ที่เก็บในไฟล์ Cookies รวมถึงเว็บไซต์ที่ให้ผู้ใช้บริการสมัครสมาชิก และยืนยันตัวตนด้วยการ Login เพื่อเข้าสู่ระบบนั้นควรจะต้องระมัดระวังเป็นพิเศษ ซึ่งลักษณะของเว็บเพจที่อาจมีช่องโหว่ประเภทนี้ คือ

(1) มีการรับข้อมูลจากผู้ใช้บริการผ่าน Input Form เช่น หน้า Login เพื่อเข้าสู่ระบบ, หน้าลงทะเบียนสมาชิก, กล่อง Comment เป็นต้น
(2) หน้าเว็บเพจที่แสดงผลจากการค้นหาข้อมูล 4.5.1 ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats)

ผู้ประสงค์ร้ายอาศัยช่องโหว่นี้ในการดำเนินการ ดังนี้

(1) แสดงหน้าเว็บเพจปลอมหรือข้อมูลปลอม บนเว็บไซต์
(2) ดักเอาไฟล์ Cookies ที่เก็บบนโปรแกรมค้นดูเว็บ โดยถ้า Session ID นั้นเก็บอยู่ใน Cookies อาจนำไปสู่การปลอมแปลง Session ID ได้ หรือถ้าเก็บข้อมูลส่วนบุคคลใน Cookies ข้อมูลส่วนบุคคลนั้น อาจถูกเปิดเผย

(3) เปลี่ยนแปลงข้อมูลใน Cookies และทำให้โปรแกรมประยุกต์บนเว็บบันทึก Cookies ที่ถูกเปลี่ยนแปลง โดยผู้ประสงค์ร้ายอาจจะแก้ไข Session ID และบันทึกใน Cookies ผู้ใช้บริการก็จะได้ Session ID ที่ถูกเปลี่ยนแปลงแก้ไขแทน

4.5.2 แนวทางในการป้องกันการโจมตี

มาตรฐานฉบับนี้มีแนวทางในการป้องกันการโจมตีด้วยเทคนิคดังกล่าว ดังนี้

4.5.2.1 Fundamental Solutions : การป้องกันการโจมตี

(1) โปรแกรมประยุกต์บนเว็บต้องมีการทำ Output Validation ในลักษณะ Sanitization

(2) การทำ HTML Entity Encoding หรือ URL Encoding กับข้อมูลที่จะแสดงผล โดยการทำ Output Validation เปรียบเสมือนการป้องกันการแสดงผลข้อมูลที่ไม่พึงประสงค์ยังฝั่งผู้ใช้บริการ เช่น การแสดงผลข้อผิดพลาด (Error Message) ที่ในบางครั้งอาจแสดงข้อมูลที่เป็นประโยชน์ต่อผู้ประสงค์ร้าย ซึ่งข้อมูลทั้งหมดนี้ผู้ประสงค์ร้ายสามารถรวบรวมมาเป็นข้อมูลที่ใช้โจมตีเว็บไซต์ได้ง่ายมากขึ้น ตัวอย่างเช่น การใช้งานฟังก์ชัน htmlentities() ในภาษา PHP เพื่อป้องกันการโจมตีด้วยเทคนิค XSS สมมติว่าผู้ประสงค์ร้ายมีการส่งค่า <script>alert("Hacked")</script> เข้ามายังระบบผ่านตัวแปรหนึ่ง เมื่อค่าดังกล่าวถูกนำไปประมวลผลผ่านฟังก์ชัน htmlentities() ซึ่งจะมีการ Encode ค่าต่าง ๆ ให้อยู่ในรูปแบบที่โปรแกรมคนดูเว็บมองเป็นเพียงข้อความธรรมดา กรณีนี้ผลลัพธ์ที่ได้จากการ Encode ด้วยฟังก์ชันดังกล่าว จะได้ออกมาเป็น <script>alert("Hacked")</script> อย่างไรก็ตาม โปรแกรมคนดูเว็บยังสามารถแสดงผลค่าดังกล่าวได้เป็น <script>alert("Hacked")</script> ในฝั่งผู้ใช้บริการได้อยู่ แต่อยู่ในรูปแบบของข้อความ ซึ่งไม่สามารถนำมาประมวลผลในลักษณะสคริปต์ตามที่ผู้ประสงค์ร้ายต้องการได้

(3) ตรวจสอบ Input Validation ไม่ให้ใช้ HTML Tag ใด ๆ เช่นไม่ Generate Content จาก Tag <script></script>

(4) ไม่อนุญาตให้มีการเรียก Stylesheets จากเว็บไซต์ที่ไม่ได้ตรวจสอบก่อน

(5) ตั้งต่า Charset Parameter ของ HTTP Content-Type Header

(6) โปรแกรมประยุกต์บนเว็บต้องมีการตรวจสอบข้อมูลชุดคำสั่งในเว็บไซต์ ตรวจสอบข้อมูลชุดคำสั่งในเว็บไซต์ว่ากำลังรับข้อมูลที่ผิดปกติ เป็นสคริปต์ที่อันตราย หรือไม่ เช่น สคริปต์ที่มีเครื่องหมายอักขระพิเศษต่าง ๆ เช่น < > ? & # เป็นต้น โดยต้องคัดกรองเครื่องหมายเหล่านี้ก่อนที่จะนำไปประมวลผลที่เครื่องบริการเว็บ และการกรองข้อมูลที่รับเข้าจากผู้ใช้เป็นหลัก โดยข้อมูลที่รับเข้าต่าง ๆ ไม่ควรถูกนำมาใช้งานในทันที แต่ต้องมีการกรองก่อนทุกครั้ง และต้องมั่นใจได้ว่าผู้ใช้ไม่สามารถวาง Script ใด ๆ ลงในเว็บได้ ควรแปลงพวก Non-Alphanumeric Data ให้กลายเป็น HTML Character เสียก่อน เช่น เครื่องหมายน้อยกว่า "<" ควรถูกแปลงเป็น "& l t ;" เป็นต้น

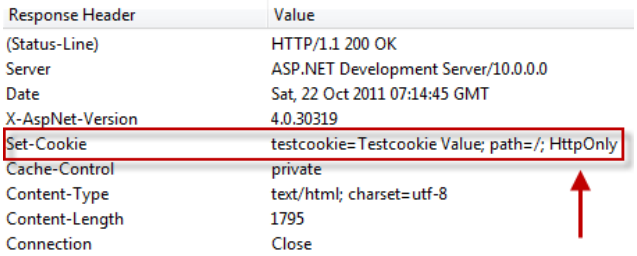
4.5.2.2 Mitigation Measures : วิธีการลดความเสียหายที่เกิดจากการถูกโจมตี

โปรแกรมประยุกต์บนเว็บต้องมีการใช้งาน HTTPOnly Cookie Flag

HTTPOnly เป็นรูปแบบการกำหนดค่าเพิ่มเติม (Flag) สำหรับป้องกันไม่ให้ฝั่งผู้ใช้บริการสามารถเข้าถึงค่า Cookie ของระบบได้ โดยทั่วไปหากระบบมีช่องโหว่ของการโจมตีด้วยเทคนิค XSS

ผู้ประสงค์ร้ายอาจส่งคำสั่งเพื่อให้โปรแกรมอ่านข้อมูล Cookie ของผู้ใช้บริการและลักลอบส่งข้อมูลออกไปยังปลายทาง รวมถึงในบางครั้งอาจสั่งให้มีการปรับเปลี่ยนค่าใน Cookie ได้ด้วย แต่อย่างไรก็ตามการใช้งาน HTTPOnly นั้นยังมีข้อจำกัดว่าสามารถใช้งานกับโปรแกรมค้นดูเว็บที่สนับสนุนเท่านั้น เช่น โปรแกรมค้นดูเว็บ Chrome ตั้งแต่เวอร์ชั่น 1.0.154 หรือ Safari ตั้งแต่เวอร์ชั่น 4 หรือ Internet Explorer ตั้งแต่เวอร์ชั่น 6sp1 เป็นต้น (ข้อมูลเพิ่มเติมจากเอกสารของ OWASP ได้ที่ https://www.owasp.org/index.php/HttpOnly#Browsers_Supporting_HttpOnly

โดยการตรวจสอบว่า เว็บไซต์ใช้ HTTPOnly หรือไม่สามารถใช้เครื่องมือ Developer Tools ของโปรแกรมค้นดูเว็บ ซึ่งหากใช้ HTTPOnly จะปรากฏข้อความดังรูป



รูปที่ 4 การแสดงผล HTTP Only

4.5.3 การทดสอบความมั่นคงปลอดภัย

จุดอ่อนด้านความมั่นคงปลอดภัยของโปรแกรมประยุกต์บนเว็บส่วนใหญ่ คือการไม่มีการตรวจสอบข้อมูลที่รับเข้ามาจากฝั่งผู้ใช้บริการก่อนที่จะนำข้อมูลนั้นไปประมวลผล ซึ่งจุดอ่อนเรื่องดังกล่าว เป็นสาเหตุของช่องโหว่สำคัญที่พบบนโปรแกรมประยุกต์บนเว็บ หลายประการ เช่น Cross-Site Scripting, SQL Injection, Interpreter Injection เป็นต้น ซึ่งในหัวข้อนี้มีเทคนิคการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่ Testing for Cross Site Scripting: https://www.owasp.org/index.php/Testing_for_Cross_site_scripting

4.6 CSRF (Cross-Site Request Forgery) [7]

Cross-Site Script Forgery (CSRF) เป็นภัยคุกคามประเภทหนึ่งที่เกิดจากการที่ผู้ประสงค์ร้ายลักลอบปลอมแปลงคำสั่งข้อมูลให้เสมือนเป็นคำสั่งจากผู้ใช้บริการระบบเพื่อให้เครื่องบริการเว็บเข้าใจว่าเป็นคำสั่งที่มาจากผู้ใช้บริการของระบบ และดำเนินการตามที่ร้องขอ เช่น ผู้ประสงค์ร้ายอาศัยช่องโหว่ของเว็บเพจ ปลอมแปลงคำสั่งข้อมูลให้เสมือนเป็นคำสั่งจากเจ้าของบัญชีจริงเพื่อติดต่อกับระบบธนาคารทางอินเทอร์เน็ต ทำให้ระบบเชื่อและเข้าใจว่าเจ้าของบัญชีต้องการทำธุรกรรมการเงินนั้น ๆ จริง ซึ่งปัญหานี้เรียกว่า “Cross-Site Request Forgery Vulnerability” และการโจมตีที่อาศัยช่องโหว่นี้เรียกว่า “Cross-Site Request Forgery Attack”

เว็บไซต์ที่ให้บริการมีความสำคัญสูงเช่น การให้บริการชำระเงินออนไลน์ หรือเว็บไซต์ที่ผู้ใช้บริการต้องผ่านกระบวนการยืนยันตัวตนด้วยการเข้าสู่ระบบก่อนจะเข้าใช้งานเว็บไซต์ เช่น การเข้าถึงหน้าเว็บเพจของผู้ดูแลระบบ เป็นต้น หรือเว็บไซต์ที่มีกระบวนการจัดการ Session ด้วยการจัดการหรือจัดเก็บ Session ในไฟล์ cookies หรือใช้เทคนิคการยืนยันตัวตนด้วย Basic Authentication (ใช้ Username และ Password) ก็ควรระมัดระวังการโจมตีประเภทนี้

4.6.1 ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats)

ผู้ประสงค์ร้ายอาศัยช่องโหว่นี้ในการ

(1) เข้าถึงและดำเนินการกับบริการต่าง ๆ ที่เข้าถึงได้เฉพาะผู้ใช้บริการที่ Login แล้ว เช่น สั่งโอนเงิน (Internet Banking Service), สั่งซื้อของออนไลน์ เป็นต้น

(2) เพิ่ม แก้ไข ลบ ข้อมูลส่วนตัวต่าง ๆ ของผู้ใช้บริการ เช่น เปลี่ยนรหัสผ่าน เป็นต้น

4.6.2 แนวทางในการป้องกันการโจมตี

มาตรฐานฉบับนี้มีแนวทางในการป้องกันการโจมตีด้วยเทคนิคดังกล่าว ดังนี้

4.6.2.1 Fundamental Solutions : การป้องกันการโจมตี

(1) ฟังก์ชันต่าง ๆ ควรดำเนินการผ่าน POST method และตรวจสอบความถูกต้องกับค่าที่ซ่อนอยู่ภายใน POST Method ก่อนจะดำเนินการต่อ

ยกตัวอย่างเช่น ขั้นตอนที่ต้องรับข้อมูลจากฝั่งผู้ใช้บริการ และต้องได้รับข้อความยืนยันจากฝั่งผู้ใช้บริการ (Comfirmation Page) ก่อนจึงจะดำเนินการ โดยให้ตั้งค่าที่เป็นความลับ ค่าหนึ่ง ให้เป็น Field ที่ซ่อนอยู่ใน POST Method เมื่อทางฝั่งผู้ใช้บริการกดส่ง Confirm จาก Confirmation Page ก็ให้แทรกค่า ๆ นี้มาด้วย (ซึ่งค่าที่เป็นความลับนี้อาจจะเป็น Session ID ที่สร้างขึ้นมาเพิ่ม ตอนที่ผู้ใช้บริการ Login) โดยเมื่อได้รับ Request ให้ตรวจสอบค่าใน Hidden Parameter ว่าเป็นค่าที่ถูกต้อง จึงจะดำเนินการต่อเท่านั้น

(2) โปรแกรมประยุกต์บนเว็บต้องมีฟังก์ชันการยืนยันตัวตนของผู้ใช้งานอีกครั้งและให้กรอก Captcha เมื่อมีการเปลี่ยนแปลงสถานะการทำงานในฟังก์ชันที่สำคัญ ๆ เช่น การชำระเงิน การเปลี่ยนรหัสผ่าน เป็นต้น

(3) โปรแกรมประยุกต์บนเว็บต้องมีการใช้งาน Unique Token และ/หรือตรวจสอบ Referrer ร่วมกับการส่งข้อมูล หรือคำสั่งผ่านแบบฟอร์ม

การสร้างข้อมูลอ้างอิง เพื่อใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่งมาประมวลผล อาจจะด้วยการสร้าง Unique Token ในแบบฟอร์ม เพื่อให้แน่ใจว่าข้อมูลในแบบฟอร์มที่จะส่งมาประมวลผลในแต่ละครั้งนั้นเป็นข้อมูลที่เกิดมาจากการที่ผู้ใช้บริการจริง ไม่ใช่โปรแกรมอัตโนมัติหรือสคริปต์ที่ใช้ในการโจมตีแต่อย่างใด ซึ่งจะเห็นตัวอย่างของวิธีการดังกล่าวได้จากการเข้าใช้งาน Online Banking ที่ในแต่ละการทำธุรกรรมจะต้องมีการยืนยันด้วยหมายเลข OTP ก่อนเสมอ เพื่อเป็นการยืนยันว่าการทำธุรกรรมนั้นเกิดจากผู้ใช้บริการจริง เมื่อตรวจสอบได้ว่ามาจาก URL ที่ถูกต้องแล้ว จึงดำเนินการต่อ

4.6.2.2 Mitigation Measures : วิธีการลดความเสียหายที่เกิดจากการถูกโจมตี

โปรแกรมประยุกต์บนเว็บต้องมีการส่งอีเมลอัตโนมัติ แจ้งผู้ใช้บริการทุกครั้ง เมื่อการดำเนินการที่สำคัญทำสำเร็จ เช่น กระบวนการสั่งโอนเงิน เป็นต้น และต้องไม่มีข้อมูลส่วนตัวของผู้ใช้บริการแทรกเข้าไปในอีเมล

โดย แนวทางการพัฒนาโปรแกรมประยุกต์บนเว็บให้มีความมั่นคงปลอดภัยจากการโจมตีด้วย CSRF สามารถศึกษาข้อมูลเพิ่มเติมได้ที่ https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

4.6.3 การทดสอบความมั่นคงปลอดภัย

CSRF เป็นการโจมตีที่สวมสิทธิของผู้ใช้บริการ ดำเนินการบางอย่างบนเว็บไซต์ ซึ่งถ้าผู้ประสงค์ร้ายสามารถโจมตีบัญชีผู้ใช้ของผู้ดูแลเครื่องบริการเว็บได้ ก็จะสามารถสั่งดำเนินการใด ๆ กับโปรแกรมประยุกต์บนเว็บ ด้วยสิทธิของผู้ดูแลเว็บไซต์ได้

เทคนิคการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่ Cross Site Request Forgery Testing (OTG-SESS-005) : https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005)

4.7 HTTP Header Injection

ผู้ประสงค์ร้ายสร้าง Response Header หรือปลอมแปลงแก้ไข Response Body ไว้ เมื่อผู้ใช้บริการเข้าเว็บไซต์ที่ผู้ประสงค์ร้ายวางกับดักไว้ จะแสดงผลหน้าเว็บเพจปลอม และอาจจะ run สคริปต์ สร้าง Cookies ปลอม และสั่งให้เก็บไว้บนโปรแกรมค้นดูเว็บของผู้ใช้บริการ ซึ่งหากขั้นตอนการสร้าง HTTP Response Header ของโปรแกรมประยุกต์บนเว็บดังกล่าวมีช่องโหว่ ผู้ประสงค์ร้ายจะสามารถเพิ่มข้อมูลใน Header หรือสามารถแก้ไขเปลี่ยนแปลงข้อมูลในส่วน Response Body ปัญหานี้เรียกว่า “HTTP Header Injection Vulnerability” และการโจมตีที่อาศัยช่องโหว่นี้ คือ “HTTP Header Injection Attack”

4.7.1 ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats)

ผู้ประสงค์ร้ายอาศัยช่องโหว่นี้ในการ

(1) กระทำการใด ๆ ที่เหมือนกับการโจมตีจากช่องโหว่ Cross-Site Scripting เช่น โปรแกรมค้นดูเว็บของผู้ใช้บริการส่ง Request ที่ไม่ได้มาจากผู้ใช้บริการโดยตรง หรือถูกบังคับให้ ฑun สคริปต์ที่แทรกอยู่บนหน้าเว็บเพจ ซึ่งสิ่งเหล่านี้เป็นภัยคุกคามประเภทเดียวกับ หัวข้อ 4.5 Cross-Site Scripting

(2) ผู้ประสงค์ร้ายสร้าง Cookies โดยไม่ได้รับอนุญาต เช่น เมื่อ HTTP Set-Cookie Header ถูกสร้างขึ้น ไฟล์ Cookies ที่ถูกสร้างโดยไม่ได้รับอนุญาต ก็จะถูกสร้างและเก็บข้อมูลในโปรแกรมค้นดูเว็บของผู้ใช้บริการด้วย

(3) Poison Web Cache HTTP Response Splitting [8] จะบังคับให้เครื่องบริการเว็บสร้าง HTTP Response หลายอันและทำให้เกิด Cache Poisioning ซึ่งจะแสดงหน้าเว็บเพจที่ผู้ประสงค์ร้ายต้องการ ด้วยการใช้ Proxy Server เก็บ Cache ของ HTTP Response ที่สร้างขึ้น และเก็บแทนที่ Cache เดิม เมื่อผู้ใช้บริการเข้าเว็บไซต์ของเหยื่อ ก็จะเห็นหน้าเว็บไซต์ที่เป็นเว็บไซต์ปลอมที่สร้างโดยผู้ประสงค์ร้าย

4.7.2 แนวทางในการป้องกันการโจมตี

4.7.2.1 Fundamental Solutions : การป้องกันการโจมตี

(1) ไม่ให้แสดงข้อมูล HTTP Header โดยตรง

(2) (ถ้ามีการใช้งาน HTTP Header API) เพิ่มการป้องกัน Unexpected Line Feeds ด้วยตนเอง ในกรณีไม่มีฟังก์ชัน Line Feed Neutralization

4.7.2.2 Mitigation Measures : วิธีลดความเสียหายที่เกิดจากการถูกโจมตี

ลบ Line Feed Characters ทั้งหมดที่ ปรากฏใน External Text Input

4.7.3 การทดสอบความมั่นคงปลอดภัย

ในส่วนนี้จะแสดงให้เห็นถึงการโจมตีที่ใช้ประโยชน์จาก HTTP โพรโทคอล โดยอาศัยจุดอ่อนของโปรแกรมประยุกต์บนเว็บร่วมด้วย ซึ่งจะวิเคราะห์จากการโจมตี HTTP Header มี 2 รูปแบบ คือ HTTP Splitting และ HTTP Smuggling

เทคนิคการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่ HTTP Header Injection Testing (OTG-INPVAL-016) : https://www.owasp.org/index.php?title=Testing_for_HTTP_Splitting/Smuggling_%28OTG-INPVAL-016%29

4.8 Mail Header Injection

โปรแกรมประยุกต์บนเว็บอาจมีฟังก์ชันในการส่งอีเมลแบบเฉพาะเจาะจงไปยังผู้ใช้บริการแต่ละราย เช่น ผู้ใช้บริการที่สั่งซื้อสินค้าของเว็บไซต์ เป็นต้น ซึ่งโดยทั่วไปแล้วอีเมลเหล่านี้จะมีเพียงผู้ดูแลเว็บ (Web Administrator) ที่สามารถเข้าถึงได้เท่านั้น โดยผู้ประสงค์ร้ายอาจจะตั้งค่า เปลี่ยนแปลง หรือเพิ่มอีเมลอื่น ๆ ได้ตามต้องการ และใช้เครื่องบริการเว็บเป็นฐานสำหรับกระจาย Spam Mail โดยช่องโหว่ดังกล่าวจะเรียกว่า “Mail Header Injection Vulnerability” และการโจมตีที่ใช้ประโยชน์จากช่องโหว่นี้ เรียกว่า “Mail Header Injection Attack”

4.8.1 ภัยคุกคามที่อาจเกิดขึ้น (Possible Threats)

ผู้ประสงค์ร้ายอาศัยช่องโหว่นี้ในการใช้โปรแกรมประยุกต์ที่ถูกโจมตีเป็นฐานส่ง Spam Mail

4.8.2 แนวทางในการป้องกันการโจมตี

4.8.2.1 Fundamental Solutions : การป้องกันการโจมตี

(1) กำหนดค่าคงที่ (Fixed Values) สำหรับองค์ประกอบของ Header เก็บค่าและแสดงผลที่รับมาจากผู้ใช้บริการในส่วนเนื้อความของอีเมล

องค์ประกอบของ Header ที่ต้องรับค่าจากผู้ใช้บริการ เช่น To, Cc, Bcc, Subject ซึ่งถ้านำค่าเหล่านี้มาใช้เป็นค่าสำหรับการส่งออกอีเมลโดยตรง ผู้ประสงค์ร้ายก็จะสามารถแทรกอีเมลอื่น ๆ แก้ไขเปลี่ยนแปลงเนื้อหาในอีเมลและส่งออกไปยังอีเมลอื่น ๆ ที่ไม่เกี่ยวข้องได้ ซึ่งแนะนำไม่ให้ใช้ External Parameter เพื่อกำหนดค่าของ อีเมล Header Element

(2) กรณีที่ไม่สามารถใช้การกำหนดค่าคงที่ (Fixed Header) ใน Header ใช้ Email-sending API ที่สามารถใช้งานร่วมกับโปรแกรมประยุกต์บนเว็บหรือภาษาที่ใช้ในการพัฒนาได้

หากไม่สามารถใช้ค่าคงที่กับ Email Header ได้ ควรใช้ Email-sending API สามารถใช้งานร่วมกับโปรแกรมประยุกต์บนเว็บหรือภาษาที่ใช้ในการพัฒนาได้แทน แต่บาง API ก็ยังอนุญาตให้เพิ่มข้อมูลใน Header ได้หลาย ๆ รายการ ซึ่งผู้พัฒนาโปรแกรมประยุกต์

บนเว็บก็จะต้องปรับปรุงเพิ่มเติม เพื่อไม่อนุญาตให้สามารถแบ่งบรรทัดใน Header ได้ เช่น การป้องกันการขึ้นบรรทัดใหม่ โดยเพิ่ม ช่องว่าง (Space) หรือ Tab หลังจากข้อมูล เพื่อเป็นสัญลักษณ์ว่าสิ้นสุดข้อมูลแล้ว และตัวอักษร หรือข้อมูลใด ๆ ที่อยู่หลังสัญลักษณ์นี้ จะถูกลบออกทั้งหมด

(3) ไม่กำหนดชื่อที่อยู่อีเมลใน HTML

ไม่ระบุชื่ออีเมลของผู้รับโดยตรงใน Hidden Parameter ที่จะส่งผ่านไปยังโปรแกรมประยุกต์บนเว็บ ผู้ประสงค์ร้ายจะสามารถดักโจมตี และเปลี่ยนค่าในอีเมลได้

4.8.2.2 Mitigation Measures : วิธีการลดความเสียหายที่เกิดจากการถูกโจมตี

ลบ Line Feed Character ทั้งหมดที่รับข้อมูลจากผู้ใช้บริการ

ลบบรรทัดของตัวอักขระ ที่รับจากผู้ใช้บริการทั้งหมดใน Input Text เนื่องจากโปรแกรมประยุกต์ บนเว็บอาจจะลบข้อมูลไม่ทั้งหมด อาจก่อให้เกิดช่องโหว่ได้

4.8.3 การทดสอบความมั่นคงปลอดภัย

ภัยคุกคามในข้อนี้จะส่งผลกับ โปรแกรมประยุกต์บนเว็บที่มีฟังก์ชันอีเมล ในการติดต่อกับ Mail Server (IMAP/SMTP) เป้าหมายของการทดสอบนี้คือ ตรวจสอบความสามารถในการส่งคำสั่งใด ๆ ก็ตามไปที่ Mail Server เพื่อดูผลลัพธ์จากการส่งข้อมูลที่ไม่ถูกต้อง

เทคนิคการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่ Mail Header Injection Testing (OTG-INPVAL-011) : https://www.owasp.org/index.php/Testing_for_IMAP/SMTP_Injection_%28OTG-INPVAL-011%29

4.9 Lack of Authentication and Authorization

ในการออกแบบเว็บไซต์ ควรจะตระหนักเรื่องความมั่นคงปลอดภัยเมื่อนำไปใช้งานจริง ซึ่งในบทนี้จะแสดงให้เห็นถึงแนวทางในการป้องกันของฟังก์ชันที่มีความสำคัญ ได้แก่ Authentication และ Authorization

4.9.1 Lack of Authentication

4.9.1.1 แนวทางในการป้องกันการโจมตี

(1) โปรแกรมประยุกต์ต้องระบุวิธีการยืนยันตัวตนของผู้ใช้บริการ (Authentication) ในกรณีที่มีการกำหนด Access Control

โดยปกติแล้ว เมื่อเว็บไซต์มีการเก็บข้อมูลที่เป็นความลับ และอนุญาตเฉพาะให้เจ้าของข้อมูลเข้าถึง และแก้ไขเปลี่ยนแปลงได้เท่านั้น ก็จะต้องมีกระบวนการในการยืนยันตัวตน แต่พบว่า บางเว็บไซต์อนุญาตให้ผู้ใช้บริการเข้าถึงข้อมูลที่เป็นความลับได้เพียงแค่ระบุอีเมลเท่านั้น ซึ่งอีเมลนั้นเป็นข้อมูลที่สามารถพบได้ทั่วไป ดังนั้นจะต้องมีวิธีการยืนยันตัวตนที่มั่นคงปลอดภัยมากขึ้น ด้วยการให้ผู้ใช้บริการระบุข้อมูลที่เป็นส่วนตัวร่วมด้วย เช่น รหัสผ่าน เป็นต้น

(2) การเก็บรหัสผ่านควรอยู่ในรูปที่มีการเข้ารหัสลับตามที่มาตรฐานด้านความมั่นคงปลอดภัยกำหนด เช่น AES หรือ Triple DES เป็นต้น และหากมีการเก็บอยู่ในรูปแบบของค่าแฮช (Hash Value)

ควรใช้ขั้นตอนวิธี (Algorithm) ตามที่มาตรฐานด้านความมั่นคงปลอดภัยกำหนดไว้ [9] เช่น SHA-224 SHA-256 SHA-384 SHA-512 เป็นต้น

4.9.1.2 Mitigation Measures : วิธีการลดความเสียหายที่เกิดจากการถูกโจมตี

รหัสผ่านที่ใช้ ต้องประกอบด้วยอักษรตัวเล็ก ตัวใหญ่ ตัวเลขและตัวอักขระพิเศษ และทั้งหมดต้องมีความยาวไม่น้อยกว่า 8 หลัก

4.9.2 Lack of Authorization

4.9.2.1 แนวทางในการป้องกันการโจมตี

โปรแกรมประยุกต์บนเว็บต้องมีกระบวนการที่ชัดเจน เพื่อให้แน่ใจว่าผู้ใช้บริการที่เข้าสู่ระบบ ไม่สามารถเข้าถึงบัญชีผู้ใช้และข้อมูลของผู้ใช้คนอื่น ๆ ได้

กระบวนการยืนยันตัวตน อนุญาตให้ผู้ใช้บริการที่เป็นเจ้าของข้อมูลสามารถเข้าถึงและแก้ไขเปลี่ยนแปลงข้อมูลได้ แต่ในกรณีที่มีผู้ใช้บริการคนอื่นเข้าสู่ระบบและใช้บริการนั้น ๆ ในเวลาเดียวกัน ผู้พัฒนาโปรแกรมประยุกต์บนเว็บจะต้องตรวจสอบให้แน่ใจว่า อนุญาตให้ผู้ใช้บริการแต่ละคนเข้าถึงได้เฉพาะข้อมูลของตนเองเท่านั้น ตัวอย่างเช่น เว็บไซต์ใช้ Session ID ในการจัดการและระบุถึงบัญชีผู้ใช้ที่เข้าสู่ระบบสำเร็จ เพื่อระบุตัวตนของผู้ใช้บริการ แต่เว็บไซต์ทั่วไปอาจจะใช้ User ID ซึ่งเป็นค่าค่าหนึ่งที่เก็บอยู่ในฐานข้อมูลในการอ้างอิง การเก็บ User ID ที่อ้างอิงจากฐานข้อมูล ใน URL หรือ POST parameter จะทำให้ผู้ประสงค์ร้ายสามารถใช้ User ID และเข้าถึงและดำเนินการใด ๆ กับฐานข้อมูลได้โดยปลอมตัวเป็นผู้ใช้บริการและเข้าถึงข้อมูลที่ไม่ได้รับอนุญาตได้

4.9.3 การทดสอบความมั่นคงปลอดภัย

ในเรื่องการรักษาความมั่นคงปลอดภัย การ Authentication และ Authorization เป็นการยืนยันตัวตนของผู้ใช้บริการที่จะเข้าติดต่อกับโปรแกรมประยุกต์บนเว็บ และเป็นกระบวนการที่อนุญาตให้เฉพาะผู้ใช้บริการที่ได้รับอนุญาต สามารถเข้าถึงข้อมูลได้เท่านั้น ตัวอย่างที่พบโดยทั่วไป คือ การ Login เข้าสู่ระบบ ซึ่งการทดสอบกระบวนการ Authentication และ Authorization นี้ก็คือการทำความเข้าใจถึงวิธีการทำงาน ขั้นตอนที่ใช้ก่อน และใช้ข้อมูลนั้นในการหาวิธีทดสอบเพื่อหลีกเลี่ยงกลไกต่าง ๆ ในการ Authentication และ Authorization

Authorization เป็นกระบวนการที่เกิดขึ้น หลังจากการ Authentication สำเร็จแล้ว ดังนั้นผู้ทดสอบจะต้องตรวจสอบ Authorization ต่อเมื่อได้รับข้อมูลที่เกี่ยวข้องอย่างถูกต้อง เช่น ชื่อบัญชีผู้ใช้และรหัสผ่านที่ถูกต้อง และบทบาทหน้าที่ พร้อมสิทธิการเข้าถึงข้อมูลต่าง ๆ ของผู้ใช้บริการคนนั้น เป็นต้น โดยผู้ทดสอบต้องตรวจสอบว่า ช่องโหว่เกี่ยวกับการกำหนดสิทธิเข้าถึงข้อมูลด้วยหรือไม่ ซึ่งเทคนิคการทดสอบโดยละเอียดสามารถดูเพิ่มเติมได้ที่

(1) Testing for Default Credentials (OTG-AUTHN-002) : https://www.owasp.org/index.php/Testing_for_default_credentials_%28OTG-AUTHN-002%29
(2) Testing for Weak Password Policy (OTG-AUTHN-007) : https://www.owasp.org/index.php?title=Testing_for_Weak_password_policy_(OTG-AUTHN-007)

(3) Testing for Bypassing Authorization Schema (OTG-AUTHZ-002) : https://www.owasp.org/index.php?title=Testing_for_Bypassing_Authorization_Schema_(OTG-AUTHZ-002)

(4) Testing for Privilege Escalation (OTG-AUTHZ-003) : https://www.owasp.org/index.php?title=Testing_for_Privilege_escalation_(OTG-AUTHZ-003)

ภาคผนวก ก. แบบประเมินสำหรับผู้พัฒนาโปรแกรมประยุกต์บนเว็บ

ก.1 แบบฟอร์มตรวจสอบสถานะความมั่นคงปลอดภัยสำหรับเว็บไซต์

ข้อที่	ประเภทช่องโหว่	ประเภทการป้องกัน	Checkbox	รายละเอียดการป้องกัน	หัวข้อที่อ้างอิงถึง
1	SQL Injection (หัวข้อที่ 4.1)	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	มีการจัดทำ Prepared Statement และ/หรือ Stored Procedure	หัวข้อ 4.1.2.1 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ไม่เขียนคำสั่ง SQL โดยตรงในตัวแปร (Parameter) ที่ส่งโดยตรงไปยังโปรแกรมประยุกต์บนเว็บ	หัวข้อ 4.1.2.1 ข้อ 2
		การลดความเสียหายที่เกิดจากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ควบคุมการแสดงผลข้อมูล Error Message	หัวข้อ 4.1.2.2 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	กำหนดสิทธิขั้นต่ำให้บัญชีผู้ใช้ของฐานข้อมูล	หัวข้อ 4.1.2.2 ข้อ 2
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน ☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	SQL Injection Testing (OTG-INPVAL-005)	หัวข้อ 4.1.3
2	OS Command Injection (หัวข้อที่ 4.2)	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	พัฒนาโปรแกรมประยุกต์บนเว็บโดยให้ปิดการใช้งานคำสั่งต่าง ๆ เพื่อป้องกันการเรียกใช้ที่ไม่พึงประสงค์	หัวข้อ 4.2.2.1
		การลดความเสียหายที่เกิดจากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	(ในการณีที่มีการเรียกใช้คำสั่ง OS Command) ต้องตรวจสอบ Variables ที่จะใช้กับตัวแปรของ OS Command ก่อนนำไปประมวลผล	หัวข้อ 4.2.2.2
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน ☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	Testing for Command Injection (OTG-INPVAL-013)	หัวข้อที่ 4.2.3
3	Unchecked Path Parameter / Directory Traversal	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ไม่อนุญาตให้ใส่ Filename เพื่อระบุถึงข้อมูลได้จาก External Parameter	หัวข้อ 4.3.2.1 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ใช้ Fixed Directory ในการจัดการระบุชื่อไฟล์	หัวข้อ 4.3.2.1 ข้อ 2
		การลดความเสียหายที่เกิด	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	กำหนด Permission การเข้าถึงไฟล์บนเครื่องบริการเว็บให้เหมาะสม	หัวข้อ 4.3.2.2 ข้อ 1

ข้อที่	ประเภทช่องโหว่	ประเภทการป้องกัน	Checkbox	รายละเอียดการป้องกัน	หัวข้อที่อ้างอิงถึง
	(หัวข้อที่ 4.3)	จากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ตรวจสอบ Filename เช่น มีการแปลง String ที่ระบุ Directory ได้ เช่น / -> % 2F	หัวข้อ 4.3.2.2 ข้อ 2
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน ☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	Testing Directory Traversal/File Include (OTG-AUTHZ-001)	หัวข้อที่ 4.3.3
4	Improper Session Management (หัวข้อที่ 4.4)	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	สร้าง Session ID เป็นที่ยากต่อการคาดเดา (ไม่ใช้ Algorithm ที่ง่ายเกินไป เช่น Pseudo Random Number)	หัวข้อ 4.4.2.1 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ไม่ใช้ URL Parameter ในการเก็บ Session ID	หัวข้อ 4.4.2.1 ข้อ 2
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	เมื่อมีการใช้งาน HTTPS Protocol ใช้ Secure Attribute ของ Cookies	หัวข้อ 4.4.2.1 ข้อ 3
		การลดความเสียหายที่เกิดจากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	กำหนดให้ Session ID เป็นค่าสุ่ม	หัวข้อ 4.4.2.2 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	กำหนดวันหมดอายุการใช้งานของ Cookies ที่เก็บ Session ID	หัวข้อ 4.4.2.2 ข้อ 2
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน ☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	1. Testing for Bypassing Session Management Schema (OTG-SESS-001) 2. Testing for Exposed Session Variables (OTG-SESS-004)	หัวข้อ 4.4.3
5	Cross-Site Scripting (หัวข้อที่ 4.5)	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ทำ Output Validation ในลักษณะ Sanitization	หัวข้อ 4.5.2.1 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ทำ HTML Entity Encoding หรือ URL Encoding กับข้อมูลที่จะแสดงผล	หัวข้อ 4.5.2.1 ข้อ 2
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ตรวจสอบ Input Validation ไม่ให้ใช้ HTML Tag ใด ๆ เช่นไม่ Generate Content จาก Tag <script></script>	หัวข้อ 4.5.2.1 ข้อ 3
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ไม่อนุญาตให้มีการเรียก Stylesheets จากเว็บไซต์ที่ไม่ได้ตรวจสอบก่อน	หัวข้อ 4.5.2.1 ข้อ 4
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ตั้งค่า Charset Parameter ของ HTTP Content-Type Header	หัวข้อ 4.5.2.1 ข้อ 5
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	โปรแกรมประยุกต์บนเว็บต้องมีการตรวจสอบข้อมูลชุดคำสั่งในเว็บไซต์	หัวข้อ 4.5.2.1 ข้อ 6

ข้อที่	ประเภทช่องโหว่	ประเภทการป้องกัน	Checkbox	รายละเอียดการป้องกัน	หัวข้อที่อ้างอิงถึง
		การลดความเสียหายที่เกิดจากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	มีการใช้งาน HTTPOnly Cookie Flag	หัวข้อ 4.5.2.2
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน ☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	Testing for Cross Site Scripting	หัวข้อที่ 4.5.3
6	Cross-Site Script Forgery (หัวข้อที่ 4.6)	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ฟังก์ชันต่าง ๆ ควรดำเนินการผ่าน POST Method และตรวจสอบความถูกต้องกับค่าที่ซ่อนอยู่ภายใน POST Method ก่อนจะดำเนินการต่อ	หัวข้อ 4.6.2.1 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	มีฟังก์ชันการยืนยันตัวตนของผู้ใช้งานอีกครั้งและให้กรอก Captcha เมื่อมีการเปลี่ยนแปลงสถานะการทำงานในฟังก์ชันที่สำคัญ ๆ	หัวข้อ 4.6.2.1 ข้อ 2
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ต้องมีการใช้งาน Unique Token และ/หรือ ตรวจสอบ Referrer ร่วมกับการส่งข้อมูล หรือคำสั่งผ่านแบบฟอร์ม	หัวข้อ 4.6.2.1 ข้อ 3
		การลดความเสียหายที่เกิดจากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	มีการส่งอีเมลอัตโนมัติ แจ้งผู้ใช้บริการทุกครั้งเมื่อการดำเนินการที่สำคัญทำสำเร็จ	หัวข้อ 4.6.2.2
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน ☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	Testing for Cross-Site Request Forgery (CSRF) (OTG-SESS-005)	หัวข้อที่ 4.6.3
7	HTTP Header Injection (หัวข้อที่ 4.7)	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ไม่ให้แสดงข้อมูล HTTP Header โดยตรง	หัวข้อ 4.7.2.1 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	(ถ้ามีการใช้งาน HTTP Header API) เพิ่มการป้องกัน Unexpected Line Feeds ด้วยตนเอง (กรณีไม่มีฟังก์ชัน Line Feed Neutralization)	หัวข้อ 4.7.2.1 ข้อ 2
		การลดความเสียหายที่เกิดจากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ลบ Line Feed Characters ทั้งหมดที่ ปรากฏใน External Text Input	หัวข้อ 4.7.2.2

ข้อที่	ประเภทช่องโหว่	ประเภทการป้องกัน	Checkbox	รายละเอียดการป้องกัน	หัวข้อที่อ้างอิงถึง
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน ☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	HTTP Header Injection Testing (OTG-INPVAL-016)	หัวข้อที่ 4.7.3
8	Mail Header Injection (หัวข้อที่ 4.8)	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	กำหนดค่าคงที่ (Fixed Values) สำหรับองค์ประกอบของ Header เก็บค่าและแสดงผลที่รับมาจากผู้ใช้บริการในส่วนเนื้อความของอีเมล	หัวข้อ 4.8.2.1 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	(กรณีที่ไม่สามารถใช้การกำหนดค่าคงที่ (Fixed Header) ใน Header) ใช้ Email-sending API ที่สามารถใช้งานร่วมกับโปรแกรมประยุกต์บนเว็บหรือภาษาที่ใช้ในการพัฒนาได้	หัวข้อ 4.8.2.1 ข้อ 2
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ไม่กำหนดชื่อที่อยู่อีเมลใน HTML	หัวข้อ 4.8.2.1 ข้อ 3
		การลดความเสียหายที่เกิดจากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ลบ Input Line Feed Character ทั้งหมดที่รับข้อมูลจากผู้ใช้บริการ	หัวข้อ 4.8.2.2
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน ☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	Mail Header Injection Testing (OTG-INPVAL-011)	หัวข้อที่ 4.8.3
9	Lack of Authentication and Authorization (หัวข้อที่ 4.9)	การป้องกันการโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	ระบุวิธีการยืนยันตัวตนของผู้ใช้บริการ (Authentication) ในกรณีที่มีการกำหนด Access Control	หัวข้อ 4.9.1.1 ข้อ 1
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	เก็บรหัสผ่านอยู่ในรูปที่มีการเข้ารหัสลับตามที่มาตรฐานด้านความมั่นคงปลอดภัยกำหนด	หัวข้อ 4.9.1.1 ข้อ 2
			☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	มีกระบวนการที่ชัดเจน เพื่อให้แน่ใจว่าผู้ใช้บริการที่เข้าสู่ระบบ ไม่สามารถเข้าถึงบัญชีผู้ใช้และข้อมูลของผู้ใช้คนอื่น ๆ ได้	หัวข้อ 4.9.2.1
		การลดความเสียหายที่เกิดจากการถูกโจมตี	☐ ยอมรับได้ ☐ ยังต้องปรับปรุง	รหัสผ่านที่ใช้ ต้องประกอบด้วยอักษรตัวเล็ก ตัวใหญ่ ตัวเลขและตัวอักขระพิเศษ และทั้งหมดต้องมีความยาวไม่น้อยกว่า 8 หลัก	หัวข้อ 4.9.1.2
		การทดสอบ	☐ ทดสอบผ่าน ☐ ทดสอบไม่ผ่าน	1. Testing for Default Credentials (OTG-AUTHN-002)	หัวข้อที่ 4.9.3

ข้อที่	ประเภทช่องโหว่	ประเภทการป้องกัน	Checkbox	รายละเอียดการป้องกัน	หัวข้อที่อ้างอิงถึง
			☐ ยังไม่สามารถทดสอบได้ด้วยตนเอง	2. Testing for Weak Password Policy (OTG-AUTHN-007) 3. Testing for Bypassing Authorization Schema (OTG-AUTHZ-002) 4. Testing for Privilege Escalation (OTG-AUTHZ-003)	

ก.2 แบบฟอร์มสำหรับการแก้ไขรายการที่ยังต้องปรับปรุง (จากการตรวจสอบสถานะความมั่นคงปลอดภัย)

ตัวอย่างของแบบฟอร์มสำหรับการแก้ไขรายการที่ยังต้องปรับปรุงซึ่งเกิดจากการตรวจสอบสถานะความมั่นคงปลอดภัยของเว็บไซต์ตามแนวทางในมาตรฐานฉบับนี้ และเมื่อพบรายการที่ไม่เป็นไปตามแนวทางที่กำหนดก็ให้ระบุรายการแก้ไขลงในแบบฟอร์มพร้อมกับกำหนดระยะเวลาในการแก้ไขเพื่อนำเสนอต่อผู้ที่เกี่ยวข้องต่อไป

วันที่ตรวจสอบสถานะ			เว็บไซต์				
โดยหน่วยงาน							
ลำดับที่	วันที่รายงาน	คำอธิบายรายการที่ยังต้องปรับปรุง	สาเหตุ	การแก้ไขชั่วคราว	สิ่งที่ต้องแก้ไข		
					รายการแก้ไข	รับผิดชอบโดย	วันที่แล้วเสร็จ

อภิธานศัพท์

SQL (Structured Query Language) [10]

หมายถึง ภาษาที่ใช้ในการเขียนคำสั่งดำเนินการกับฐานข้อมูล (Database) เพื่อให้ฐานข้อมูลดำเนินการทีละคำสั่ง เพื่อให้คอมพิวเตอร์ไปเลือกหาข้อมูลที่ต้องการมาแสดง

NoSQL (Not Only SQL) [11]

หมายถึง เทคโนโลยีฐานข้อมูลที่ออกแบบมาให้สามารถจัดการข้อมูลที่มีกับขนาดใหญ่ และมีความยืดหยุ่นในการขยายขนาดโครงสร้างของข้อมูลได้ โดยไม่ต้องกำหนดโครงสร้างของตารางในการจัดเก็บไว้ก่อนเหมือนกับฐานข้อมูลเชิงสัมพันธ์ (Relational Database)

GET method [12]

หมายถึง วิธีการในการเรียกดูข้อมูลจาก Web Server โดยการระบุข้อมูลที่ต้องการเรียกดูผ่าน Request-URL

POST method [12]

หมายถึง วิธีการในการส่งข้อมูลให้กับ Web Server โดยเป็นการส่งข้อมูลโดยตรงจาก HTML Form และมีการเข้ารหัสข้อมูล ก่อนส่งไปยัง Web Server

Stored Procedure [13]

หลายครั้งในการพัฒนาโปรแกรมประยุกต์บนเว็บมีการ Query ข้อมูลที่ซับซ้อนมากเกินกว่า 1 คำสั่ง ดังนั้นฐานข้อมูลส่วนใหญ่จึงอนุญาตให้ผู้พัฒนาสร้างโปรแกรมย่อยซึ่งสามารถดำเนินการชุดคำสั่ง SQL ได้ และถูกจัดเก็บไว้ในฐานข้อมูล โปรแกรมย่อยดังกล่าวนี้เรียกว่า Stored Procedure โดยโปรแกรมประยุกต์บนเว็บสามารถ Query ข้อมูลที่ซับซ้อนนี้ได้ผ่าน Stored Procedure แทนการเขียนชุดคำสั่ง SQL ที่ซับซ้อนลงในโค้ดของโปรแกรมประยุกต์บนเว็บ

SMTP (Simple Mail Transfer Protocol) [14]

หมายถึง โปรโตคอลในการส่งอีเมลในเครือข่ายอินเทอร์เน็ต

มัลแวร์ (Malware : Malicious Software) [14]

หมายถึง ซอฟต์แวร์ไม่พึงประสงค์ เป็นโปรแกรมที่ถูกเขียนขึ้นมาเพื่อทำอันตรายกับระบบ เช่น ทำให้คอมพิวเตอร์ทำงานผิดปกติ ขโมย หรือทำลายข้อมูล หรือเปิดช่องทางให้ผู้ไม่หวังดีเข้ามาควบคุมเครื่องคอมพิวเตอร์ เป็นต้น

ในการแบ่งประเภทของมัลแวร์ โดยปกติจะแบ่งตามพฤติกรรมการทำงาน ตัวอย่างเช่น
Virus - แพร่กระจายตัวเองไปยังเครื่องอื่น ๆ ผ่านไฟล์
Worm - แพร่กระจายตัวเองไปยังเครื่องอื่น ๆ ผ่านระบบเครือข่าย (เช่น อีเมล หรือระบบแชร์ไฟล์)

Trojan - หลอกว่าเป็นโปรแกรมที่ปลอดภัยแล้วให้ผู้ใช้หลงเชื่อนำไปติดตั้ง
Backdoor - เปิดช่องทางให้ผู้ไม่หวังดีเข้ามาควบคุมเครื่อง
Rootkit - เปิดช่องทางให้ผู้ไม่หวังดีเข้ามาควบคุมเครื่อง พร้อมได้สิทธิ์ของผู้ดูแลระบบ
Spyware - แอบดูพฤติกรรมการใช้งานของผู้ใช้ และอาจขโมยข้อมูลส่วนตัวด้วย

Phishing [14]

หมายถึง เทคนิคการหลอกลวงโดยใช้อีเมลหรือหน้าเว็บไซต์ปลอมเพื่อให้ได้มาซึ่งข้อมูล เช่น ชื่อผู้ใช้ รหัสผ่าน หรือข้อมูลส่วนบุคคลอื่น ๆ เพื่อนำข้อมูลที่ได้ไปใช้ในการเข้าถึงระบบโดยไม่ได้รับอนุญาต หรือสร้างความเสียหายในด้านอื่น ๆ เช่น ด้านการเงิน เป็นต้น ในบทความนี้จะเน้นในเรื่องของ Phishing ที่มีจุดมุ่งหมายเพื่อหลอกลวงทางการเงิน เนื่องจากจะทำให้ผู้อ่านมองเห็นผลกระทบได้ง่าย

Session [15]

ใน HTTP โปรโตคอลนั้น เครื่อง Server จะส่ง Respond ตามที่ Client ส่ง Request เข้ามาโดยไม่คำนึงถึงลำดับหรือความสัมพันธ์ของ Request ก่อนหน้า เนื่องจาก HTTP Server ไม่มีการบันทึกสถานะของ Client Request จึงจำเป็นต้องมีกลไกที่ใช้ในการจัดการแลกเปลี่ยนข้อมูล และบันทึกสถานะในการส่ง Request และ Response ระหว่าง Server และ Client

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [16]

หมายถึง กลไกที่คอมพิวเตอร์แม่ข่ายมีกลไกในการถามผู้ใช้งานด้วยสร้างการทดสอบขึ้นมา และผู้ใช้จำเป็นต้องตอบให้ถูกต้องเพื่อให้สามารถเข้าสู่ระบบได้ ทั้งนี้ คอมพิวเตอร์เองนั้นไม่สามารถแก้ปัญหาที่ตัวมันเองสร้างขึ้นได้ สามารถตรวจสอบได้แค่ว่าข้อมูลที่ผู้ใช้งานกรอกเข้ามาถูกหรือผิดเท่านั้น ระบบ CAPTCHA โดยทั่วไปจะให้ผู้ใช้ตอบคำถามด้วยการกดแป้นตัวอักษรตามที่ปรากฏในรูปภาพที่บิดเบี้ยว บางครั้งอาจมีการเพิ่มจุด แถบสี หรือเส้นบิดงอ ลงในรูปภาพนั้น เพื่อวัตถุประสงค์ในการหลีกเลี่ยงการตรวจจับของโปรแกรมประเภทโอซีอาร์

บรรณานุกรม

[1] J. Information-Technology Promotion Agency, How to Secure Your Website, IPA.

[2] OWASP, OWASP Testing Guide 4.0.

[3] สำนักงานราชบัณฑิตยสภา, “ศัพท์บัญญัติราชบัณฑิตยสถาน,” [ออนไลน์]. Available: http://rirs3.royin.go.th/coinages/webcoinage.php.

[4] “คลังศัพท์ไทย,” สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ, [ออนไลน์]. Available: http://thaiglossary.org/.

[5] OWASP, “Top10-2013-A2-Broken Authentication and Session Management,” 2013. [ออนไลน์]. Available: https://www.owasp.org.

[6] OWASP, “Cross-site Scripting,” [ออนไลน์]. Available: https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).

[7] OWASP, “Cross-Site Request Forgery (CSRF),” 2015. [ออนไลน์]. Available: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF).

[8] OWASP, “HTTP Response Splitting,” 2013. [ออนไลน์]. Available: https://www.owasp.org/index.php/HTTP_Response_Splitting.

[9] E. Barker และ A. Roginsky, “NIST Special Publication 800-131A,” National Institute of Standards and Technology, NIST, U.S. Department of Commerce , 2011.

[10] W3Schools, “w3schools,” [ออนไลน์]. Available: http://www.w3schools.com/sql/sql_intro.asp.

[11] “mongoDB,” [ออนไลน์]. Available: https://www.mongodb.com/nosql-explained.

[12] “Tutorials Point,” [ออนไลน์]. Available: http://www.tutorialspoint.com/php/php_get_post.htm.

[13] Microsoft, “Microsoft Developer Network,” [ออนไลน์]. Available: https://msdn.microsoft.com/en-us/library/ms190782.aspx.

[14] ThaiCERT ETDA, [ออนไลน์]. Available: https://www.thaicert.or.th.

[15] W3Schools, “W3Schools,” [ออนไลน์]. Available: http://www.w3schools.com/php/php_sessions.asp.

[16] Carnegie Mellon University, “CAPTCHA,” [ออนไลน์]. Available: http://www.captcha.net/.