การวิเคราะห์ปัญหาและพัฒนาวิธีแก้ปัญหาสำหรับการทำงานผิดปกติของเอชเอสทีเอสและการจู่โจมโดยการเปลือยเอสเอสแอลต่อความมั่นคงของเว็บ

วิทยานิพนธ์

ของ

ภารเดช คะเชนรัมย์

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

มิถุนายน 2564

การวิเคราะห์ปัญหาและพัฒนาวิธีแก้ปัญหาสำหรับการทำงานผิดปกติของเอชเอสทีเอสและการจู่โจมโดยการเปลือยเอสเอสแอลต่อความมั่นคงของเว็บ

วิทยานิพนธ์

ของ

ภารเดช คะเชนรัมย์

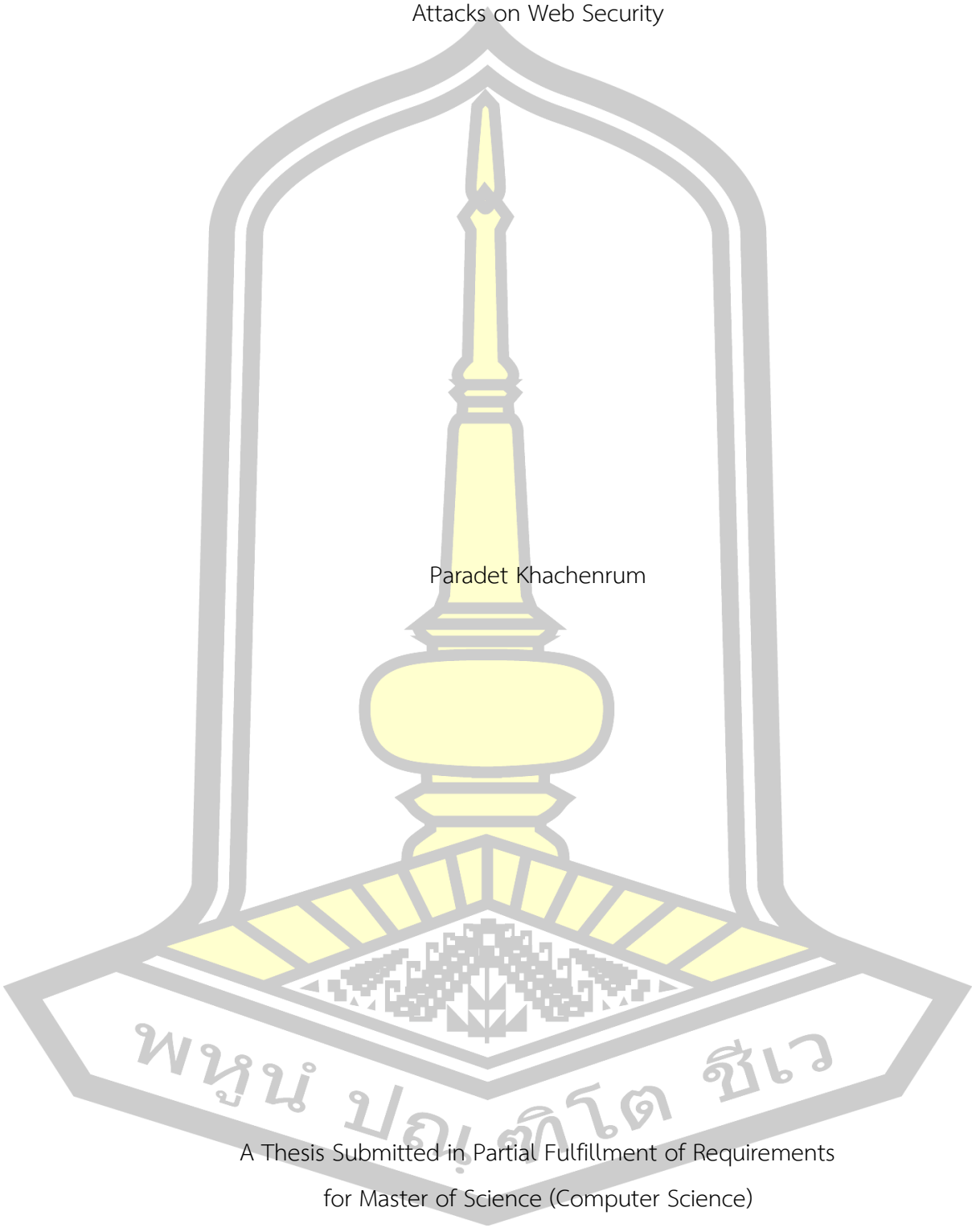
เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

มิถุนายน 2564

Problem Analysis and Solution Development for HSTS malfunction and SSL Stripping Attacks on Web Security

Paradet Khachenrum



A Thesis Submitted in Partial Fulfillment of Requirements

for Master of Science (Computer Science)

June 2021

คณะกรรมการสอบวิทยานิพนธ์ ได้พิจารณาวิทยานิพนธ์ของนายภารเดช คะเชนรัมย์ แล้วเห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ผศ. ดร. วรารัตน์ สงฆ์แป้น)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผศ. ดร. สมนึก พ่วงพรพิทักษ์)

.....กรรมการ
(ผศ. ดร. ฉัตรเกล้า เจริญผล)

.....กรรมการ
(ผศ. ดร. สุชาติ คุ้มมะณี)

มหาวิทยาลัยอนุมัติให้รับวิทยานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

.....
(ผศ. ศศิธร แก้วมั่น)
คณบดีคณะวิทยาการสารสนเทศ

.....
(รศ. ดร. กริสน์ ชัยมูล)
คณบดีบัณฑิตวิทยาลัย

ชื่อเรื่อง การวิเคราะห์ปัญหาและพัฒนาวิธีแก้ปัญหาสำหรับการทำงานผิดปกติของเอชเอสทีเอสและการจู่โจมโดยการเปลือยเอสเอสแอลต่อความมั่นคงของเว็บ

ผู้วิจัย ภารเดช คะเชนรัมย์

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร. สมนึก พ่วงพรพิทักษ์

ปริญญา วิทยาศาสตรมหาบัณฑิต **สาขาวิชา** วิทยาการคอมพิวเตอร์

มหาวิทยาลัย มหาวิทยาลัยมหาสารคาม **ปีที่พิมพ์** 2564

บทคัดย่อ

การโจมตีด้วยการเปลือยเอสเอสแอลเป็นหนึ่งในเทคนิคยอดนิยมเพื่อหลบหลีกจากโพรโทคคอลเอสเอสแอล ทำให้เว็บไซต์ไม่มีการสื่อสารผ่านเอชทีทีพีเอส ดังนั้นกลไกเอชเอสทีเอสจึงได้ถูกนำเสนอเพื่อแก้ปัญหาดังกล่าว แต่อย่างไรก็ตาม หลายการศึกษาเมื่อไม่นานมานี้ ได้รายงานว่าระบบธนาคารออนไลน์และเว็บอีคอมเมิร์ซหลายแห่ง ถูกโจมตีอย่างได้ผลด้วยการเปลือยเอสเอสแอล แม้จะมีการตั้งค่าเอชเอสทีเอสแล้วก็ตาม ดังนั้นวิทยานิพนธ์นี้ จึงทำการตรวจสอบและวิเคราะห์หาเหตุผลเบื้องหลังการทำงานล้มเหลวของเอชเอสทีเอส และการกลับมาโจมตีได้ใหม่ของการเปลือยเอสเอสแอล เพื่อวิเคราะห์ปัญหา ได้มีการทดลองบนเครือข่ายเพื่อการทดสอบ ต่อเว็บธนาคารออนไลน์ของไทย 11 แห่ง ระบบเว็บอีคอมเมิร์ซ 4 เว็บ เว็บระบบทะเบียนของมหาวิทยาลัยในไทย 11 เว็บ และเว็บอาสาสมัครอีก 2 เว็บ นอกจากนี้ ยังมีการวิเคราะห์เฮดเดอร์ของเอชทีทีพีที่ตอบกลับมา และวิเคราะห์สคริปต์ที่แฮกเกอร์ใช้ในการโจมตี ในที่สุดสาเหตุของปัญหาก็ได้รับการวิเคราะห์และแนวทางในแก้ปัญหาจากการตั้งค่าได้ถูกเสนอแนะ ยิ่งไปกว่านี้ วิทยานิพนธ์นี้ยังได้พัฒนาแนวทางแก้ปัญหาเพิ่มเติมเพื่อเป็นชั้นที่สองจากเอสเอสแอลเพื่อป้องกันการดักจับข้อมูล แนวทางการแก้ปัญหานี้ถูกออกแบบจากแนวคิด รหัสผ่านที่ถูกแฮชกับซอลท์ และรหัสผ่านใช้ครั้งเดียวผ่านโทรศัพท์มือถือ ต้นแบบของแนวทางแก้ปัญหาดังกล่าวได้ถูกพัฒนา และได้ถูกวิเคราะห์สมรรถภาพและความมั่นคง เราได้พบว่าแนวทางการป้องกันการแอบดักจับข้อมูลที่เสนอมีประโยชน์และมีต้นทุนทางสมรรถภาพเพียงเล็กน้อย

คำสำคัญ : เอชทีทีพีเอส, กลไกเอชเอสทีเอส, การโจมตีด้วยการเปลือยเอสเอสแอล, ความมั่นคงเว็บ

TITLE Problem Analysis and Solution Development for HSTS malfunction and SSL Stripping Attacks on Web Security

AUTHOR Paradet Khachenrum

ADVISORS Assistant Professor Somnuk Puangpronpitag , Ph.D.

DEGREE Master of Science **MAJOR** Computer Science

UNIVERSITY Mahasarakham University **YEAR** 2021

ABSTRACT

SSL (Secure Socket Layer) stripping attack was one of the most popular techniques to bypass the SSL protocol, causing the website to do not communicate via HTTPS. So, an HSTS (HTTP Strict Transport Security) mechanism had been proposed to solve this problem. However, several recent studies have reported that some online banking systems and e-commerce websites could be effectively attacked by the SSL stripping technique again, even with an HSTS setting. Hence, this thesis investigates and analyzes the reasons behind the malfunction of HSTS and the return of SSL stripping attacks. To analyze the problem, testbed experiments have been done on 11 Thai online banking websites, 4 e-commerce websites, 11 university registration system websites, 2 volunteer websites. Furthermore, HTTP response headers and the hacker scripts hackers have been analyzed. The causes of the problems have finally been analyzed, and the setting solutions are suggested. In addition, this thesis has also developed an extra solution as a second security layer of SSL to protect against sniffing. The solution is designed on the concepts of a salted hash password and mobile OTP (One Time Password). The prototype of the solution has been developed. Security and performance analysis has been done. We have found that the sniffing-protection solution is useful and causes very little performance overhead.

Keyword : HTTPS, HTTP Strict Transport Security, SSL Stripping Attack, Web Security

พหูน ปณฺฑิโต ชีเว

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ เป็นผลจากการศึกษาระดับปริญญาโท สาขาวิทยาการคอมพิวเตอร์ โดยได้รับคำชี้แนะในการศึกษาและแนวทางในการดำเนินการวิจัยจากอาจารย์ ผู้ช่วยศาสตราจารย์ ดร.สมนึก พ่วงพรพิทักษ์ ประธานกรรมการคุมวิทยานิพนธ์

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.สมนึก พ่วงพรพิทักษ์ ที่เป็นผู้ผลักดันให้วิทยานิพนธ์นี้เสร็จสมบูรณ์

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.วรารัตน์ สงฆ์แป้น ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.ฉัตรเกล้า เจริญผล และผู้ช่วยศาสตราจารย์ ดร.สุชาติ คุ้มมะณี กรรมการสอบวิทยานิพนธ์ที่เป็นผู้ให้คำชี้แนะเพื่อปรับปรุงข้อบกพร่องของวิทยานิพนธ์ฉบับนี้

ขอขอบคุณการสนับสนุนการวิจัย จากกรมสอบสวนคดีพิเศษ กระทรวงยุติธรรม โดยเฉพาะอย่างยิ่งคุณเอกชัย พ่วงพรพิทักษ์ ขอขอบคุณคณะวิทยาการสารสนเทศ ที่ให้ทุนสนับสนุนงานวิจัยกับทีมที่ปรึกษาที่มีส่วนหนึ่งของวิทยานิพนธ์นี้เกี่ยวข้อง

ขอขอบพระคุณบิดา มารดา และสมาชิกในครอบครัวทุกคน ที่สั่งสอนและให้กำลังใจเสมอมา

ภารเดช คะเชนรัมย์

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพประกอบ

หน้า

บทที่ 1

บทนำ

1.1 หลักการและเหตุผล

การรักษาความมั่นคงของเว็บไซต์มีประเด็นที่ต้องให้ความสำคัญ คือ ข้อมูลที่มีการสื่อสารระหว่างผู้ใช้งานกับผู้ให้บริการ (Client Browser กับ Web Server) จำเป็นต้องมีการเข้ารหัสในการสื่อสาร ซึ่งเทคโนโลยีที่มีการใช้กันได้แก่ HTTPS (Hypertext Transfer Protocol Secure) [1] ร่วมกับ TLS (Transport Layer Security) [2] เพื่อปกป้องข้อมูลระหว่างการสื่อสาร โดยอาศัยเทคนิคโครงสร้างพื้นฐานกุญแจสาธารณะที่เรียกว่า Public Key Infrastructure (PKI) [3] โครงสร้างดังกล่าวจะมีกระบวนการออกใบรับรอง (Certificate) ของเซิร์ฟเวอร์สำหรับการพิสูจน์ตัวจริง (Authentication) ซึ่งใบรับรองดังกล่าวจะถูกรับรองโดยหน่วยงานที่เรียกว่า Certificate Authority (CA) เพื่อรับประกันว่าเว็บเซิร์ฟเวอร์ของผู้ให้บริการนั้นเป็นตัวจริง ไม่ได้ถูกปลอมแปลง ซึ่งมีกระบวนการเข้ารหัสแบบสมมาตร (Symmetric) และอสมมาตร (Asymmetric) ทำให้การสื่อสารผ่านโพรโทคอล HTTPS อยู่ในรูปแบบ Cipher Text มีความมั่นคงระหว่างการสื่อสารเพื่อป้องกันปัญหาการถูกดักจับข้อมูล

แต่อย่างไรก็ตาม HTTPS ที่ใช้รักษาความมั่นคงในเว็บไซต์ก็ยังถูกโจมตีได้ด้วยวิธีการต่าง ๆ ซึ่งมีเทคนิคที่สำคัญในการโจมตีได้แก่การเปลือยเอสเอสแอล (SSL Stripping Attack) ถูกเสนอโดย Moxie Marlinspike และคณะ [4] ในปี ค.ศ 2009 ทำให้ระบบธนาคารออนไลน์ (Internet Banking) และระบบการค้าอิเล็กทรอนิกส์ (E-Commerce) หรือในหลายเว็บไซต์ถูกโจมตีและตกเป็นคดีที่สำคัญในทั่วโลกรวมถึงประเทศไทย ผลลัพธ์จากการถูกโจมตีด้วยเทคนิค SSL Stripping Attack ส่งผลกระทบต่อการทำงานของโพรโทคอล HTTPS ซึ่งจะถูกเปลี่ยนการทำงานเป็น HTTP ทำให้ระบบเว็บไซต์ธนาคารออนไลน์ถูกแฮกเกอร์โจมตีได้ SSL Stripping Attack จึงถือเป็นภัยคุกคามร้ายแรงที่สร้างปัญหาให้กับระบบเว็บไซต์มายาวนาน มีหลายงานวิจัยที่พยายามเข้ามาวิเคราะห์แก้ไขปัญหาดังกล่าว เช่น SSLock [5], HProxy [6], HTTPSLock [7], ISAN-HTTPS Enforcer [8],Click2Enforce [9], HTTP Strict Transport Security (HSTS) [10] กระทั่งปัจจุบันมาตรฐานที่ถูกเลือกมาแก้ไขปัญหาดังกล่าวก็คือกลไก HSTS และยังเป็นหนึ่งในมาตรฐานของ Internet Engineering Task Force (IETF) ตามเอกสาร RFC 6797 ที่ถูกเสนอให้ใช้ป้องกันการถูกโจมตีด้วยวิธี SSL Stripping Attack และยังรองรับการทำงานในทุกเว็บเบราว์เซอร์ โดยหน้าที่ผู้ดูแลเว็บเซิร์ฟเวอร์ต้องทำการ Configuration ให้กลไก HSTS ทำงาน ซึ่งเว็บไซต์ทั่วโลกรวมถึงในประเทศไทยหลายแห่งมีการดำเนินการปรับใช้กลไกดังกล่าวเป็นที่เรียบร้อยแล้ว โดยการทำงานของกลไก HSTS จะบังคับให้

เว็บเบราว์เซอร์ที่กำลังทำงานอยู่บนเว็บไซต์ต้องสื่อสารผ่าน HTTPS เท่านั้น แม้ผู้ใช้จะไม่ระบุว่าต้องการใช้ HTTPS ก็ตามจึงทำให้เว็บไซต์ที่ต้องการความมั่นคงสูงอย่าง เช่น ธนาคารออนไลน์ เว็บไซต์การค้าอิเล็กทรอนิกส์ สามารถป้องกันการถูกโจมตีได้ ยกเว้นแต่ว่ามีบางเว็บไซต์ที่ไม่มีการดำเนินการปรับใช้กลไก HSTS เท่านั้น

จากที่กล่าวมาข้างต้นจะพบว่าปัญหาการโจมตีแบบ SSL Stripping Attack เหมือนจะสิ้นสุดลงแล้ว กระทั่งเมื่อเดือน ตุลาคม พ.ศ 2562 ISAN Lab [11] ได้ทำการวิเคราะห์ปัญหา Web Security ในประเทศไทย สำรวจเว็บไซต์ที่ให้บริการธนาคารออนไลน์ รวมถึงระบบเซอร์วิสที่สำคัญต่าง ๆ ของประเทศไทย ค้นพบว่าระบบเว็บไซต์ธนาคารออนไลน์ที่เคยได้รับการป้องกัน ล้วนแล้วแต่ถูกโจมตีได้อีกครั้งเกือบทั้งสิ้น ในการทดสอบครั้งนี้ใช้การจู่โจมทั้งสคริปต์การโจมตีแบบใหม่ของแฮกเกอร์และสูตรการโจมตีแบบเก่าของ Moxie Marlinspike พบว่าถึงแม้มีการ Configuration ปรับใช้กลไกของ HSTS ตามสูตรที่รู้กัน HSTS กลับไม่ประสบผลสำเร็จในการป้องกัน ทำให้การโจมตีด้วย SSL Stripping Attack ที่ไม่สามารถโจมตีได้มานานหลายปีกลับมาเป็นภัยคุกคามต่อความมั่นคงของระบบเว็บไซต์ใหม่อีกครั้ง

ดังนั้นวิทยานิพนธ์นี้ จึงเสนอที่จะทำการวิเคราะห์ปัญหาความผิดปกติของกลไก HSTS ซึ่งเป็นกลไกที่สำคัญของการทำงานร่วมกับโพรโทคอล HTTPS ในการรักษาความมั่นคงของเว็บไซต์ และเพื่อเข้าใจสาเหตุที่เทคนิค SSL Stripping Attack กลับมาโจมตีได้ใหม่อีกครั้ง โดยข้อมูลที่สรุปได้นี้ จะนำไปสู่การพัฒนาแนวทางการแก้ไขปัญหาและโปรแกรมต้นแบบ (prototype) เพื่อป้องกันการถูกโจมตีดังกล่าวให้ยั่งยืน

1.2 วัตถุประสงค์ของการวิจัย

1) เพื่อวิเคราะห์ปัญหาการทำงานที่ผิดปกติของกลไก HSTS และการกลับมาจู่โจมได้ใหม่ของเทคนิคการเปลือยเอสเอสแอลต่อระบบความมั่นคงเว็บไซต์

2) เพื่อเสนอกลไกชั้นที่ 2 ป้องกันการถูกดักจับข้อมูล (Data Sniff) กรณี SSL/TLS ถูกทำลาย

3) เพื่อประเมินประสิทธิผลแนวทางการแก้ปัญหาการถูกดักจับข้อมูลจากเทคนิคการโจมตีด้วยการเปลือยเอสเอสแอล

1.3 ความสำคัญของการวิจัย

ความสำคัญของงานวิจัย แบ่งออกเป็น 3 ส่วน คือ

1) องค์ความรู้ใหม่ทางด้านการวิเคราะห์ปัญหาการทำงานผิดปกติของกลไก HSTS จากการศึกษาพบว่ายังไม่มีงานวิจัยใดวิเคราะห์ปัญหาด้านนี้มาก่อน ซึ่งผลการสำรวจจากกลุ่ม Isan Lab [11] ในเดือน ตุลาคม ปี พ.ศ. 2562 ชี้ให้เห็นว่าการ Configuration ปรับใช้ HSTS เพื่อรักษาการทำงานของโพรโทคอล HTTPS ไม่ประสบผลสำเร็จในการป้องกันจากการถูกโจมตีด้วยเทคนิค SSL Stripping Attack ซึ่งปกติตามสูตรที่รู้กัน HSTS สามารถป้องกันการโจมตีดังกล่าวได้ ทั้งนี้ เพื่อให้เข้าใจถึงปัญหาการเปลี่ยนแปลงของกลไก HSTS และหาแนวทางการแก้ไขจากการถูกโจมตีด้วยเทคนิค SSL Stripping Attack ซึ่งทำให้เกิดช่องโหว่จากการถูกจู่โจมคือผู้ประสงค์ร้ายสามารถดักจับข้อมูล (Data Sniff) ของเหยื่อที่เป็นเป้าหมายได้อย่างง่ายได้ งานวิจัยนี้จึงทำการวิเคราะห์ปัญหาความมั่นคงของกลไก HSTS อย่างละเอียด เพื่อให้เข้าใจถึงปัญหาและการเปลี่ยนแปลง ซึ่งยังไม่มีงานวิจัยอื่นทำมาก่อน

2) องค์ความรู้เกี่ยวกับแนวทางการแก้ไขปัญหาป้องกันการถูกดักจับข้อมูล (Data Sniff) เป็นที่ทราบกันดีว่าโพรโทคอล HTTPS ถูกสร้างขึ้นมีวัตถุประสงค์หลักที่สำคัญในส่วนหนึ่ง คือ ป้องกันการถูกดักจับข้อมูล เว็บไซต์ทุกหน่วยงานอาศัยความมั่นคงของโพรโทคอล HTTPS เป็นมาตรฐานหลักในการรักษาความปลอดภัยระหว่างการสื่อสาร จากการศึกษางานวิจัยที่เกี่ยวข้องจาก Isan Lab [11] พบว่า HTTPS มักถูก Bypassing ด้วยเทคนิคการโจมตีแบบ SSL Sniffing Attack และ SSL Stripping Attack แม้แต่ล่าสุดมาตรฐานการป้องกันด้วย HSTS ก็ยังมีปัญหาในการรักษาความปลอดภัย งานวิจัยนี้จึงเสนอวิธีการแก้ปัญหาที่ยั่งยืน โดยสร้างมาตรการรักษาความปลอดภัยชั้นที่ 2 ซ้อนทับ HTTPS อีกชั้น เพื่อป้องกันการถูกดักจับข้อมูล (Data Sniff) ถึงแม้ว่าโพรโทคอลมาตรฐานจะถูกทำลาย แต่ก็ยังได้รับการป้องกันจากกลไกชั้นที่ 2 รักษาข้อมูลให้อยู่ในรูปค่า Hash

3) การนำไปใช้ประโยชน์ งานวิจัยนี้จึงเสนอวิธีการป้องกันปัญหาการโจมตีเว็บไซต์ที่ทำงานบน HTTPS และเพื่อป้องกันการถูกดักจับข้อมูล สามารถนำมาเป็นองค์ความรู้เพื่อปรับใช้ได้จริง ในระบบที่ต้องการความมั่นคงสูงอย่างเช่น ระบบธนาคารออนไลน์ ระบบการค้าอิเล็กทรอนิกส์ หรือเว็บไซต์ต่าง ๆ เพื่อให้บริการเว็บไซต์มีความมั่งคงปลอดภัย นอกจากนี้ยังสามารถช่วยป้องกันการก่ออาชญากรรมทางคอมพิวเตอร์ ซึ่งจะส่งผลดีด้านเศรษฐกิจ สังคม เกิดประโยชน์ต่อการพัฒนาประเทศในต่อไป

1.4 ขอบเขตของการวิจัย

1) วิเคราะห์ปัญหาการทำงานที่ผิดปกติของกลไก HSTS (HTTP Strict Trans port Security) ที่ใช้รักษาความมั่นคงโพรโทคอล HTTPS ในระบบเว็บไซต์

2) ประเมินปัญหาการป้องกันเว็บไซต์จากการโจมตี HTTPS โดยวิธีการ SSL Stripping Attack ในกลุ่มตัวอย่างระบบที่ให้บริการ Internet Banking, E-Commerce และระบบทะเบียนมหาวิทยาลัยในประเทศไทย

3) เสนอแนวทางการแก้ไขปัญหาจากการถูกโจมตีดักจับข้อมูล (Data Sniff) ด้วยเทคนิคการโจมตีแบบ SSL Stripping Attack

1.5 นิยามศัพท์เฉพาะ

1) Hyper Text Transfer Protocol (HTTP) คือ โพรโทคอลในระดับชั้นแอปพลิเคชันของชุด โพรโทคอล Transmission Control Protocol/Internet Protocol (TCP/IP) ซึ่งกำหนดรูปแบบการร้องขอข้อมูลของไคลเอนท์ ในรูปแบบ HTTP Request ผ่านทางโปรแกรมเว็บเบราว์เซอร์ไปยังเซิร์ฟเวอร์ และกำหนดรูปแบบการถ่ายโอนไฟล์จากทางด้านเซิร์ฟเวอร์ไปยังไคลเอนท์ ซึ่งเมื่อเซิร์ฟเวอร์ได้รับการร้องขอ ก็จะทำการค้นหาไฟล์ที่ถูกระบุใน Uniform Resource Locator (URL) ซึ่งเป็นที่อยู่ของไฟล์หรือเว็บไซต์บนอินเทอร์เน็ต ถ้าพบก็จะทำการตอบกลับ HTTP Response พร้อมกับส่งไฟล์ดังกล่าวไปให้กับเว็บเบราว์เซอร์เพื่อแสดงผลที่ฝั่งของไคลเอนท์

2) การโจมตีแบบแทรกกลางการสื่อสาร (Man in The Middle Attack: MITM) เป็นรูปแบบการโจมตีที่ผู้โจมตีเข้าแทรกกลางการสื่อสารระหว่างคอมพิวเตอร์สองเครื่อง โดยทำการดักจับข้อมูลที่รับและส่ง ในระหว่างการสื่อสาร ซึ่งข้อมูลที่เป็นเป้าหมายการดักจับได้แก่ ชื่อผู้ใช้ (Username), รหัสผ่าน (Password) ที่ใช้ในการตรวจสอบสิทธิ์เข้าใช้งานระบบ ข้อมูลบัตรเครดิต เป็นต้น

3) HTTP Strict Transport Security (HSTS) คือ กลไกที่บังคับใช้ HTTPS โดยกำหนดให้เว็บเบราว์เซอร์ที่กำลังทำงานอยู่บนเว็บไซต์ต้องสื่อสารผ่าน HTTPS เท่านั้น โดยฝั่งเซิร์ฟเวอร์ (Server) จะไม่สื่อสารผ่าน HTTP รูปแบบการขอ Request ระหว่าง Client กับ Web Server จะเป็นลักษณะสื่อสารผ่าน HTTPS เท่านั้น

4) เกณฑ์วิธีป้องกันชั้นซ็อกเก็ต (Secure Socket Layer Protocol: SSL) เป็นโพรโทคอลที่ถูกพัฒนาโดย Netscape Communications เพื่อใช้ในโพรโทคอล HTTP โดยโพรโทคอล SSL จะทำงานระหว่าง Application Protocol และ TCP เพื่อใช้เข้ารหัสของข้อมูลและการพิสูจน์ตัวตนในการสื่อสารระหว่างเซิร์ฟเวอร์และไคลเอนท์ ซึ่งทำให้การสื่อสารผ่านเว็บมีความปลอดภัยขึ้น เป็นเทคโนโลยีที่ถูกพัฒนาขึ้น เพื่อสนับสนุนการค้าอิเล็กทรอนิกส์ (E-Commerce) ผ่านหน้าเว็บ

5) การโจมตีโดยการเปลือยเอสเอสแอล (SSL Stripping Attack) คือ การโจมตีที่อาศัยวิธีโจมตีแบบแทรกกลางการสื่อสาร ร่วมกับวิธีการโจมตีเว็บไซต์ที่ทำงานบน HTTPS ซึ่งเมื่อเหยื่อถูกโจมตี โปรแกรมเว็บเบราว์เซอร์จะใช้โพรโทคอล HTTP ทำให้ไม่มีความปลอดภัยในการสื่อสาร

6) เว็บเบราว์เซอร์ คือ ซอฟต์แวร์โปรแกรมสำหรับใช้ท่องเว็บหรือใช้ดูข้อมูลสารสนเทศในเว็บไซต์ต่าง ๆ เปรียบเสมือนเครื่องมือในการสื่อสารกับเครือข่ายคอมพิวเตอร์ขนาดใหญ่ที่เรียกว่า World Wide Web

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

เนื้อหาในบทนี้กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้องเพื่อทำการศึกษาโพรโทคอลที่มีการนำมาใช้ในการสื่อสารสำหรับระบบเว็บไซต์ที่ต้องการความมั่นคง ในการป้องกันการโจมตีแบบแทรกกลางการสื่อสาร (Man In The Middle Attack) การโจมตีด้วยวิธีเปลือยเอสเอสแอล (SSL Stripping Attack) ซึ่งจะกล่าวถึงลักษณะการทำงานและปัญหาในการใช้งานตลอดจนเครื่องมือที่ใช้ในการโจมตีแบบแทรกกลางการสื่อสารที่นิยมใช้ในปัจจุบัน

2.1 วิธีป้องกันชั้นซ็อกเก็ต Secure Socket Layer

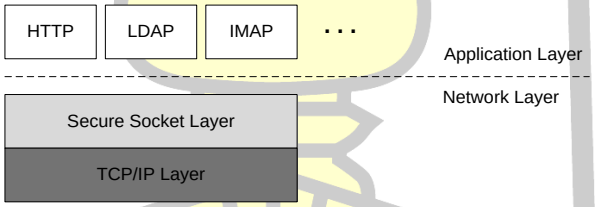
วิธีป้องกันชั้นซ็อกเก็ต (Secure Socket Layer: SSL) [12] สร้างขึ้นเพื่อรักษาความมั่นคงปลอดภัยของข้อมูลที่ส่งผ่านระบบเครือข่าย ถูกพัฒนาขึ้นโดยบริษัทเน็ตสเคป (Netscape Communications) ในปี ค.ศ. 1994 ในช่วงที่การค้าอิเล็กทรอนิกส์เป็นที่นิยมถูกสร้างขึ้นเพื่อเป็นโพรโทคอลที่ให้บริการสำหรับการเข้ารหัสของข้อมูลและการพิสูจน์ตัวตนในการสื่อสารระหว่างเซิร์ฟเวอร์และไคลเอนท์ ซึ่งจะทำให้การสื่อสารผ่านระบบเว็บไซต์มีความมั่นคงมากยิ่งขึ้น โดยปกติแล้วข้อมูลที่ส่งไปมาระหว่างเซิร์ฟเวอร์และไคลเอนท์ จะไม่มีการเข้ารหัสข้อมูลทำให้ข้อมูลอยู่ในรูปแบบของ Clear Text ข้อมูลที่ส่งผ่านระบบเครือข่ายอินเตอร์เน็ตจึงสามารถถูกดักจับได้โดยง่าย

จนกระทั่งเมื่อปี ค.ศ. 1997 SSL เวอร์ชัน 3.0 ได้ถูกทำการทดสอบความมั่นคงพบว่าไม่มีความปลอดภัยในการใช้งาน จึงนำไปสู่การพัฒนา Transport Layer Security (TLS) Version 1.0 ให้เกิดเป็นมาตรฐานกลางของโพรโทคอลบนอินเทอร์เน็ต ที่ถูกกำหนดให้เป็นมาตรฐานอย่างเป็นทางการ ตามเอกสาร RFC 2246 [12] ในปี ค.ศ. 1999 โดยเป็นมาตรฐานการสื่อสารของ Internet Engineering Task Force (IETF) ซึ่งโพรโทคอล TLS นั้นจะมีความแตกต่างจาก SSL ไม่มากหนัก โดยส่วนที่แตกต่างกันก็คือ Version, Cipher Suite, Alert Protocol, Handshake Protocol และ Record Protocol เป็นต้น

ต่อมา TLS เวอร์ชัน 1.0 จำเป็นที่จะต้องยกเลิกโพรโตคอลที่ล้าสมัย มีการพัฒนาเพื่อให้เกิดความมั่งคงในการใช้งานผ่านโพรโทคอลบนอินเทอร์เน็ตอย่างต่อเนื่อง จึงเกิดการพัฒนา TLS เวอร์ชัน 1.2 ขึ้นตามเอกสาร RFC 5246 [2] ในปี ค.ศ. 2008 นั้น TLS เวอร์ชัน 1.2 จะมีรูปแบบการทำงานของระบบเทียบเท่ากับ TLS เวอร์ชัน 1.0 โดยปัจจุบันเพื่อให้ตอบสนองต่อเทคโนโลยีที่ต้องการความมั่งคงในการใช้งาน TLS เวอร์ชัน 1.3 เวอร์ชันใหม่ล่าสุดที่เพิ่งประกาศใช้งาน ตามเอกสาร RFC 8446 [13] ในปี ค.ศ. 2018 ถูกออกแบบมาเพื่อป้องกันการดักจับข้อมูลที่ส่งผ่านระบบเครือข่ายให้

สามารถทำได้ยากขึ้น และเพิ่มประสิทธิภาพในการเข้ารหัสข้อมูลให้มีความปลอดภัย ด้วยการลดขั้นตอนที่เกิดในกระบวนการโดยรวมให้น้อยลง รวมถึงบังคับเลิกใช้ Encryption Algorithm ตัวเก่าหลายตัวที่เคยสนับสนุนใน TLS เวอร์ชัน 1.2 ที่มีช่องโหว่ต่าง ๆ มากมาย

SSL และ TLS เป็นโพรโทคอลที่ทำงานในระดับของ Transport Layer ดังนั้นในการนำใช้งานจึงไม่ได้ถูกจำกัดเพียงแค่เฉพาะการใช้งานกับโปรแกรมประยุกต์เว็บ (Web Application) เท่านั้น แต่โพรโทคอลอื่นในระดับของ Application Layer เช่น IMAP, POP3, LDAP, SSH และ FTP ก็สามารถใช้โพรโทคอลทั้งสองนี้ในการในการสื่อสารระหว่างกันเพื่อความปลอดภัยได้เช่นเดียวกัน ดังแสดงในภาพที่ 2.1 ในงานวิจัยนี้ได้ศึกษาถึงการนำ SSL/TLS มาประยุกต์ใช้กับโปรแกรมประยุกต์เว็บ ซึ่งปกติใช้โพรโทคอล HTTP ในการสื่อสารข้อมูลระหว่างกัน โดยข้อมูลที่รับส่งนี้ก็จะอยู่ในรูปแบบของข้อความปกติ (Clear Text) ที่มีความเสี่ยงต่อการถูกโจมตีด้วยการดักจับข้อมูล จึงมีการเสนอให้นำ SSL/TLS มาประยุกต์ใช้ร่วมกับ HTTP สร้างเป็นการสื่อสารรูปแบบใหม่ขึ้นเรียกว่า HTTP Over TLS (HTTPS) [1] ซึ่งเกิดจากแนวความคิดในการสร้างช่องทางการสื่อสารข้อมูลบนเว็บไซต์ที่มีการเข้ารหัสเพื่อความปลอดภัยในระหว่างการใช้งานเว็บไซต์ ซึ่งบนโพรโทคอล HTTPS นี้ เว็บเบราว์เซอร์จะแสดง URL ของเว็บไซต์เป็น https://www.example.com แทน ซึ่งเดิมในโพรโทคอล HTTP ที่เคยใช้งานนั้น URL ดังกล่าว คือ http://www.example.com



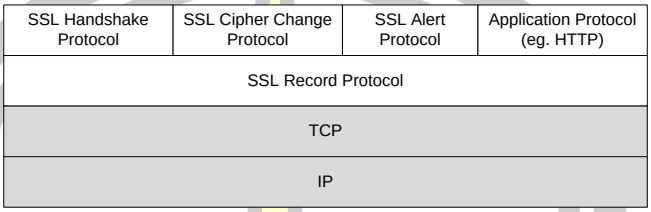
ภาพที่ 2.1 SSL/TLS ทำงานระหว่าง Application Protocol และ TCP/IP
ที่มา: [14]

2.1.1 การทำงานของ Secure Socket Layer

มาตรฐานโพรโทคอล TLS เวอร์ชัน 1.2 ตามเอกสาร RFC 5246 [2] นั้น ได้กล่าวถึงการทำงานของ SSL ซึ่งประกอบไปด้วยการทำงานร่วมกันของโพรโทคอลกลุ่มต่าง ๆ ที่ถูกแบ่งการทำงานออกเป็น 2 ชั้น ประกอบด้วย

1) SSL Record Protocol ซึ่งเป็นโพรโทคอลที่ทำหน้าที่ในการรักษาความมั่นคงของข้อมูล (Security) และคงความสมบูรณ์ของข้อมูล (Integrity) เอาไว้

2) SSL Handshake Protocol, SSL Cipher Change Protocol และ SSL Alert Protocol โดยทั้ง 3 โพรโทคอลที่กล่าวมานี้จะถูกออกแบบมาเพื่อใช้ในการเชื่อมต่อโดยเฉพาะ ดังแสดงในภาพที่ 2.2



ภาพที่ 2.2 โครงสร้างการทำงานของ SSL/TLS Protocol

ที่มา: [15]

การทำงานของ SSL/TLS จะเริ่มต้นจากการเจรจาระหว่างกัน เพื่อตกลงเกี่ยวกับอัลกอริทึมและคีย์ที่จะถูกนำมาใช้สำหรับเข้ารหัสข้อมูล รวมถึงขั้นตอนของการพิสูจน์ทราบตัวตน (Authentication) ของผู้ใช้เมื่อทำการตกลงซึ่งกันและกันได้แล้วไคลเอนท์และเซิร์ฟเวอร์ก็จะเริ่มทำการสื่อสารกัน โดยข้อมูลที่รับส่งระหว่างกันนั้นจะถูกเข้ารหัสด้วยเซสชั่นคีย์ที่ตกลงกันไว้ตั้งแต่ต้น ซึ่งตัวอย่างของรูปแบบการทำงานก็คือในขั้นตอนการสื่อสารข้อมูลของเว็บไซต์ที่ทำงานบน HTTPS ก่อนการเริ่มต้นสื่อสารข้อมูลระหว่างไคลเอนท์กับเว็บเซิร์ฟเวอร์นั้น ขั้นตอนแรกเมื่อผู้ใช้ต้องการเรียกใช้งานเว็บไซต์ใดใดก็ตาม โดยปกติจะใช้เว็บเบราว์เซอร์โดยมีการระบุชื่อเว็บไซต์ที่ต้องการ ตัวอย่างเช่น example.com หรือ www. example.com จากนั้นเว็บเบราว์เซอร์จะทำการประมวลผลแล้วเริ่มต้นการสื่อสารโดยส่งการร้องขอไปยังเว็บเซิร์ฟเวอร์บน HTTP ตาม URL ที่ระบุเอาไว้และเมื่อเว็บเซิร์ฟเวอร์ได้รับคำขอดังกล่าวแล้วก็จะประมวลผลการร้องขอนั้น แล้วส่งข้อมูลเพื่อตอบกลับมายังฝั่งไคล์เอนท์บนโพรโทคอล HTTPS พร้อมกับส่งใบรับรองของเว็บไซต์มาด้วย จากนั้นระบบจึงเริ่มขั้นตอนของ SSL Handshake ในการสื่อสารข้อมูล โดยเว็บเบราว์เซอร์จะทำหน้าที่ในการประมวลผลเพื่อพิสูจน์ใบรับรองของเว็บไซต์ โดยการใช้กุญแจสาธารณะของ Certificate Authority (CA) ที่ถูกติดตั้งเอาไว้บนเว็บเบราว์เซอร์มาทำการตรวจสอบความถูกต้อง และขั้นตอนในการตกลงกุญแจที่ถูกนำมาใช้ในกระบวนการเข้ารหัส ถอดรหัสข้อมูล

2.1.2 วัตถุประสงค์ของ Secure Socket Layer

1) พิสูจน์ตัวตน (Authentication) ของทั้งไคลเอนท์และเซิร์ฟเวอร์ โดยอาศัยหลักการทำงานของ Public Key Encryption (PKE) [16] มาใช้ในการพิสูจน์ใบประกาศอิเล็กทรอนิกส์

(Digital Certificate) และลายเซ็นดิจิทัล (Digital Signature) ของทั้งไคลเอนท์และเซิร์ฟเวอร์ ซึ่ง SSL Certificates จะออกโดยหน่วยงานที่มีความน่าเชื่อถือ (Certificate Authority: CA)

2) คงความสมบูรณ์ของข้อมูล (Integrity) ในระหว่างการสื่อสาร โดยมีการรักษาความคงสภาพข้อมูลจากแหล่งที่มา และไม่ได้ถูกแก้ไขโดยผู้ที่ไม่ได้รับอนุญาต

3) รักษาความลับของข้อมูล (Confidentiality) ระหว่างการสื่อสารเป็นการทำให้ข้อมูลสามารถเข้าถึงหรือเปิดเผยได้เฉพาะผู้ที่ได้รับอนุญาตเท่านั้น โดยข้อมูลที่ถูกรับส่งระหว่างไคลเอนท์และเซิร์ฟเวอร์จะถูกนำมาทำการเข้ารหัส ซึ่งจะอาศัยการทำงานของการเข้ารหัสร่วมกันสองแบบ คือ Public Key Encryption (PKE) และ Secret Key Encryption (SKE) [17]

2.2 Public Key Infrastructure

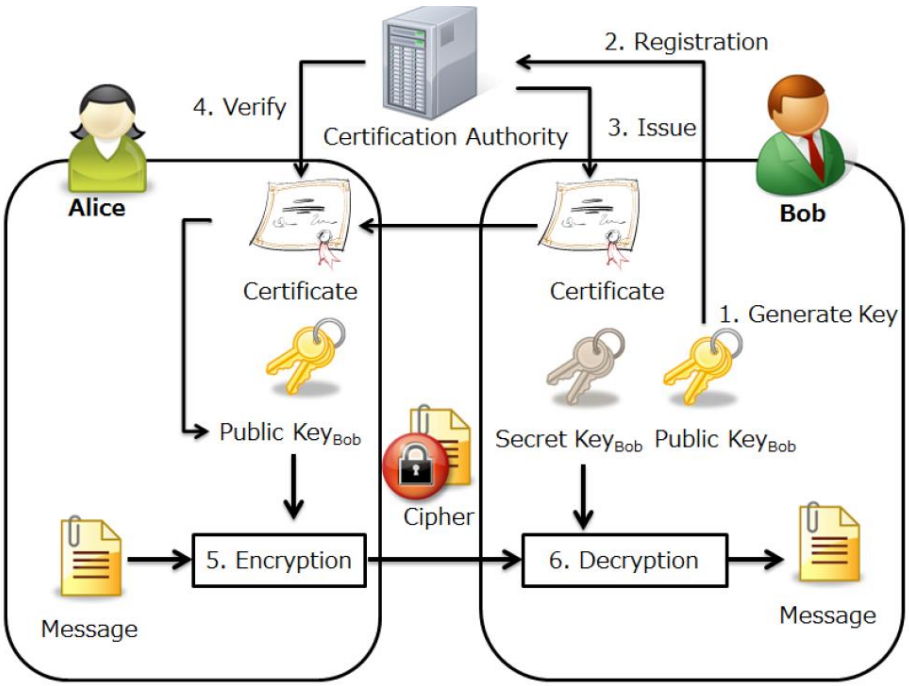
Public Key Infrastructure (PKI) หรือ เทคโนโลยีโครงสร้างพื้นฐานกุญแจสาธารณะ เกิดจากความต้องการในการเข้ารหัสข้อมูลแบบกุญแจสาธารณะ (Public Key Encryption) เนื่องจากการเข้ารหัสนี้ ผู้รับจะต้องแจกจ่ายกุญแจสาธารณะ (Public Key) ให้แก่ผู้ส่ง เพื่อใช้ในการเข้ารหัส และเมื่อผู้รับได้ข้อมูลที่เข้ารหัสก็จะทำการถอดรหัสด้วยกุญแจส่วนตัว (Private Key) แต่การที่ผู้ส่งจะรู้ได้ว่า Public Key ของผู้รับนั้นเป็นของจริงหรือไม่ ต้องอาศัยกระบวนการในการพิสูจน์ความเป็นตัวจริงกุญแจสาธารณะ ที่เรียกว่า เทคโนโลยีโครงสร้างพื้นฐานกุญแจสาธารณะ หรือ PKI โดยจะประกอบไปด้วย 4 องค์ประกอบ ดังนี้

1) End Entity หรือผู้ใช้บริการ ที่ต้องการขอใช้บริการใบรับรองดิจิทัล (Digital Certificate) ซึ่งจะต้องทำการร้องขอการลงทะเบียนไปยัง Registration Authority (RA)

2) Registration Authority หรือ RA หน่วยงานที่รับหน้าที่ลงทะเบียนใบรับรอง ให้แก่ผู้ใช้ทำหน้าที่ยืนยันความถูกต้องของข้อมูลผู้ใช้ ที่ร้องขอการลงทะเบียน นอกจากนั้นยังทำหน้าที่ในการเพิกถอนใบรับรอง หรือต่ออายุใบรับรอง โดย RA อาจจะเป็นหน่วยงานเดียวกับผู้ให้การรับรอง หรือ Certificate Authority (CA) ได้

3) Certificate Authority หรือ CA เป็นหน่วยงานที่ให้การพิสูจน์ความเป็นตัวจริง ว่าใบรับรองดังกล่าว เป็นของผู้ใช้ตัวจริง โดย CA จะต้องมีการลงลายเซ็นดิจิทัล (Digital Signature) ในใบรับรอง (Certificate) ของผู้ใช้งานด้วย Private Key ของ CA

4) Digital Certificate หรือ ใบรับรองดิจิทัล หรืออาจจะเรียกว่า Certificate ก็ได้ เป็นเอกสารดิจิทัล ที่ประกอบไปด้วย ข้อมูลพื้นฐานทั่วไปของผู้ใช้ (เจ้าของ Public Key นั้น) ข้อมูล Public Key ของผู้ใช้ และข้อมูล Digital Signature ที่เกิดจากการรับรองของ CA โดยมีการทำงานของ PKI ดังภาพที่ 2.1



ภาพที่ 2.1 โครงสร้างพื้นฐานกุญแจสาธารณะ

ที่มา: [18]

จากภาพที่ 2.1 เริ่มต้น (1) ผู้ใช้งาน (Bob) จะทาการสร้างกุญแจคู่ (Generate Key Pair) ประกอบด้วย Public Key และ Private Key โดยที่ Private Key นั้นจะเก็บไว้เป็นความลับ ส่วน Public Key จะทำการนำเข้าสู่กระบวนการร้องขอใบรับรอง (Certificate Signing Request : CSR) (2) การลงทะเบียนกับ RA หรือโดยทั่วไปคือ CA ซึ่งก็จะออกใบรับรอง (Certificate) ในขั้นตอนที่ (3) ให้กับผู้ใช้เพื่อนำไปแจกจ่ายแก่ผู้ใช้อื่นๆ เช่น Alice นำมาเข้ารหัสสำหรับส่งข้อมูลไปยังผู้รับ (Bob) (4) เมื่อ Alice (ผู้ส่ง) ได้รับใบรับรอง ก็จะทำการตรวจสอบใบรับรองว่าเป็นของ Bob จริงหรือไม่ และหากพิสูจน์แล้วว่าข้อมูลทุกอย่างเป็นจริง ก็จะนาเอา Public Key ของ Bob ที่อยู่ในใบรับรอง มาใช้สำหรับกระ บวนการเข้ารหัสข้อมูล ในขั้นตอนที่ (5) เพื่อส่งข้อมูลที่เข้ารหัสนี้ไปยังผู้รับ (Bob) และ (6) เมื่อ Bob ซึ่งเป็นผู้รับ ได้รับข้อมูลที่เข้ารหัสมาจาก Alice ก็จะทำการถอดรหัสด้วยกุญแจส่วนตัว (Private Key) ซึ่งก็จะได้ข้อความที่ถอดรหัสแล้ว นำไปใช้งานต่อไป ด้วยการบวนการต่างๆ เหล่านี้ ทำให้การติดต่อสื่อสารระหว่างผู้ส่งและผู้รับ มีความมั่นคง สามารถพิสูจน์ความเป็นตัวจริงและปกปิดข้อมูลเป็นความลับได้

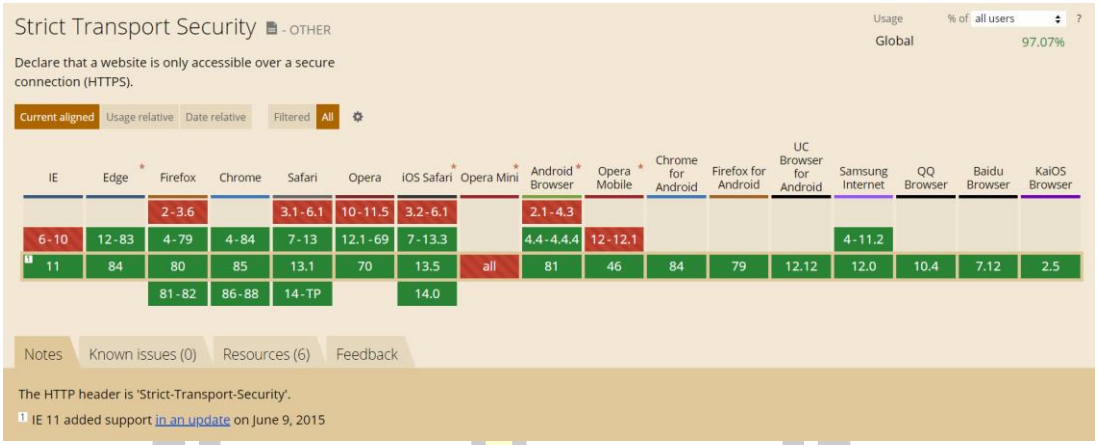
2.3 ปัญหาความมั่นคงของ PKI

แม้ว่า PKI จะมีกระบวนการสำหรับพิสูจน์ความเป็นตัวจริง และกระบวนการเข้ารหัสข้อมูล เพื่อให้การรับรองแก่ผู้ใช้ว่า เครื่องแม่ข่ายเป็นตัวจริง ไม่ได้ถูกปลอมแปลง ทำให้การสื่อสารของระบบ Internet Banking มีความมั่นคง อย่างไรก็ตาม PKI ยังคงมีปัญหาความมั่นคง ดังที่ได้เกิดขึ้นมาแล้วในอดีต และปัจจุบัน คือ ปัญหาความมั่นคงที่เกิดจาก C

A ถูกจู่โจม เพื่อทำให้การออกใบรับรองไม่ถูกต้อง เป็นผลให้เกิดการพิสูจน์ความเป็นตัวจริงผิดพลาด เช่น กรณีการถูกจู่โจมของบริษัท Comodo [19] และ DigiNotar [20] ในปี 2011 ซึ่งเป็นบริษัทออกใบรับรอง ด้วยวิธีการจู่โจม CA นี้ แม้จะถูกจู่โจมแค่บริษัท หรือหน่วยงานเดียว แต่ก็เกิดความเสียหายไปทั่วทั้งระบบได้ หรือกล่าวได้ว่า มีโอกาสที่จะเกิดการจู่โจมได้จากทุก CA และผลการจู่โจมนั้น ส่งผลกระทบต่อระบบพิสูจน์ความเป็นตัวจริงทั้งระบบ

2.4 กลไก HTTP Strict Transport Security

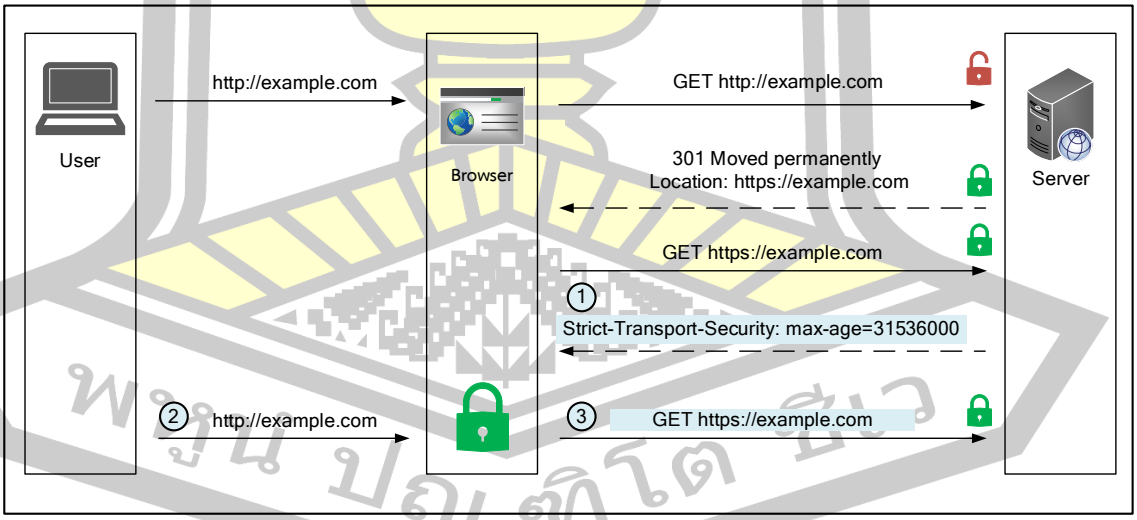
HTTP Strict Transport Security (HSTS) [10] เป็นกลไกมาตรฐานการสื่อสารของ Internet Engineering Task Force (IETF) ตามเอกสาร RFC 6797 ในปี ค.ศ. 2012 ที่ถูกสร้างขึ้นเพื่อรักษาความมั่นคงเว็บไซต์ที่ทำงานผ่านเว็บเบราว์เซอร์ (Web Browser) โดยมีจุดประสงค์หลักคือป้องกันการถูกโจมตีด้วยวิธีแทรกกลางการสื่อสาร (Man In The Middle) [4] รูปแบบการทำงานโดยเซิร์ฟเวอร์ (Web Server) จะมีการตอบกลับ (Response) ในส่วนของ HTTP Header เมื่อเว็บเบราว์เซอร์ตรวจสอบพบ HTTP Header ชื่อ Strict-Transport-Security: max-age=31536000; include SubDomains เว็บเซิร์ฟเวอร์จะบังคับให้สื่อสารผ่านช่องทางการเข้ารหัส HTTPS เท่านั้น และเว็บเบราว์เซอร์จะปฏิเสธการเชื่อมต่อ HTTP ทั้งหมด ทำให้ HSTS มีประสิทธิภาพในการป้องกันการถูกโจมตีด้วยด้วยวิธีแทรกกลางการสื่อสาร เช่น SSL Stripping Attack ซึ่งเป็นเทคนิคที่นิยมในการโจมตี และปัจจุบันกลไก HSTS รองรับการทำงาน Browser หลักทุกตัว [21] ดังแสดงในภาพที่ 2.3



ภาพที่ 2.3 เบราว์เซอร์ที่รองรับการทำงานกลไก HSTS

2.4.1 การทำงานในเว็บเซิร์ฟเวอร์ HTTP Strict Transport Security

โดยทั่วไปเมื่อไคลเอนท์ (Client) ป้อน URL ในเว็บเบราว์เซอร์ (Web Browser) ตัวอย่าง เช่น www.example.com ในกรณีเช่นนี้เว็บเบราว์เซอร์จะอนุมานว่าไคลเอนท์ (Client) ต้องการที่จะสื่อสารผ่านโพรโทคอล HTTP ในขั้นตอนนี้เอง HSTS จะทำงานโดยบังคับให้เว็บเซิร์ฟเวอร์ทำการเปลี่ยนเส้นทาง (301 Redirect) ชี้ไปยังโพรโทคอล HTTPS ลักษณะการทำงานดังแสดงในภาพที่ 2.4



ภาพที่ 2.4 การทำงาน HTTP Strict Transport Security

ที่มา: [22]

การใช้งานในเว็บเซิร์ฟเวอร์ HTTP Strict Transport Security (HSTS) เพื่อให้เว็บเบราว์เซอร์ (Web Browser) ทำงานโดยบอกให้เว็บเบราว์เซอร์ที่กำลังสื่อสารผ่านเว็บไซต์ต้องเชื่อมต่อผ่าน HTTPS เท่านั้น สามารถทำได้โดยเพิ่มชุดคำสั่ง HTTP Header ในเว็บเซิร์ฟเวอร์ ตัวอย่างที่เป็นมาตรฐาน Strict-Transport-Security: max-age=31536000;includeSubDomains ซึ่งมีลักษณะการทำงานแต่ละพารามิเตอร์ดังนี้

1. Strict-Transport-Security คือ ส่วนหัว (Header) ที่สำคัญสำหรับบอกเว็บเบราว์เซอร์เพื่อให้เข้าใจว่าเว็บไซต์ต้องการเรียกใช้ HSTS

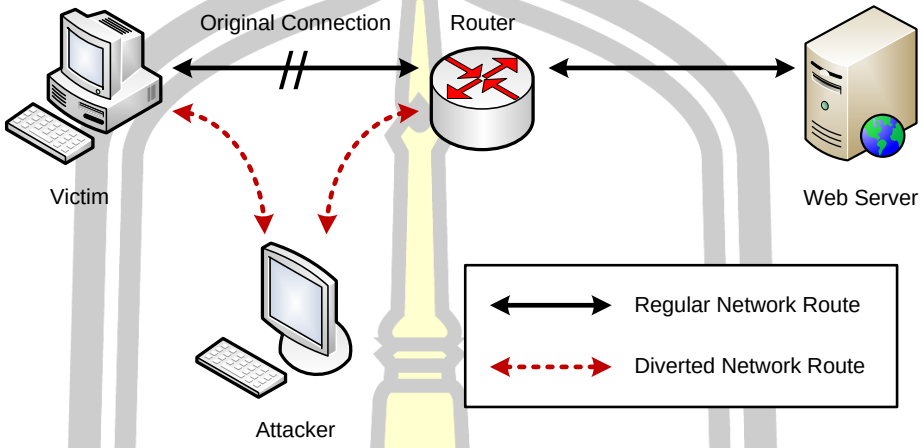
2. max-age คือ อายุของการบังคับใช้ HSTS เช่น ตั้งค่า max-age=31536000 มีหน่วยเป็นวินาที แสดงว่าเซิร์ฟเวอร์ (Server) จะไม่สื่อสารผ่าน HTTP บนเว็บเบราว์เซอร์ (Web Browser) เลยเป็นเวลา 1 ปี

3. includeSubDomains กรณีสื่อสารผ่าน Sub Domains ของเว็บไซต์ก็ต้องสื่อสารผ่าน HSTS เช่นเดียวกัน

2.5 การโจมตีแบบแทรกกลางการสื่อสาร

การโจมตีแบบแทรกกลางการสื่อสาร (Man in The Middle Attack) [23, 24] หมายถึงการที่มีผู้ไม่ประสงค์ดีเข้ามาทำการแทรกกลางในการสนทนาระหว่างคอมพิวเตอร์สองเครื่อง โดยทำหน้าที่เป็นตัวกลางในการรับส่งข้อมูลของคู่สนทนา โดยที่คู่สนทนาไม่สามารถทราบได้ว่ามีผู้ไม่ประสงค์ดีทำหน้าที่เป็นผู้รับและส่งข้อมูลต่อให้กับเครื่องคอมพิวเตอร์ที่เป็นคู่สนทนาของตนอยู่ ดังแสดงในภาพที่ 2.5 ทำให้ผู้ไม่ประสงค์ดีสามารถใช้รูปแบบการโจมตีในลักษณะนี้ กระทำการดักจับ (Sniffing) หรือเปลี่ยนแปลงข้อมูลที่ทั้ง 2 ฝ่ายสื่อสารกันอยู่ได้ ซึ่งการโจมตีในรูปแบบนี้ถูกนำมาประยุกต์ใช้กับการสื่อสารต่าง ๆ ในระบบคอมพิวเตอร์ ตัวอย่างเช่น การโจมตีแบบ MITM ในระบบเครือข่ายไร้สาย Wi-Fi ทำให้ผู้ไม่ประสงค์ดีสามารถแทรกแซงการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์และอุปกรณ์ Wi-Fi Access Point เพื่ออ่าน ปลอมแปลง หรือแก้ไขข้อมูลที่รับส่งระหว่างคอมพิวเตอร์ทั้ง 2 เครื่องนั้นได้ ซึ่งการเข้ารหัสลับข้อมูลในการสื่อสารเพียงอย่างเดียวไม่สามารถป้องกันการโจมตีในรูปแบบนี้ได้เสมอไป ถ้าหากผู้รับและส่งสารไม่ได้มีกลไกใด ๆ ที่นำมาใช้ในการยืนยันความถูกต้องของเครื่องคอมพิวเตอร์ที่เป็นคู่สนทนา การโจมตีด้วยวิธี MITM ดังกล่าวก็จะสามารถใช้โจมตีการสื่อสารของระบบได้โดยง่าย เนื่องจากรูปแบบและมาตรฐานในการสื่อสารข้อมูลต่าง ๆ บนระบบเครือข่ายอินเทอร์เน็ตไม่ได้ถูกออกแบบมาให้มีการรักษาความมั่นคงปลอดภัยของข้อมูล เช่น การสื่อสารข้อมูลผ่านทาง Hyper Text Transfer Protocol (HTTP) เพื่อเรียกดูข้อมูลหรือใช้บริการเว็บไซต์ต่าง ๆ ซึ่งส่วนใหญ่มักจะไม่มีการเข้ารหัสในการป้องกันข้อมูล จึงทำให้ผู้ไม่ประสงค์ดีสามารถใช้โปรแกรมสำหรับดักจับข้อมูลในเครือข่าย เช่น Driftnet [25], Dsniff [26], Bettercap [27],

Ettercap [28], Wireshark [29] และ TCPDump [30] โจมตีเพื่อทำการดักจับแพ็กเก็ตหรือเฟรมข้อมูลที่ถูกส่งผ่านทางระบบเครือข่ายได้



ภาพที่ 2.5 การโจมตีแบบแทรกกลางการสื่อสาร

ที่มา: [31]

ในงานวิจัยนี้ได้นำเครื่องมือซึ่งได้รับความนิยมที่มีการอัพเดตต่อเนื่อง คือ โปรแกร Kali linux, Bettercap และ Ettercap มาใช้เป็นเครื่องมือในการทดสอบการโจมตีแบบแทรกกลางการสื่อสาร ซึ่งการใช้เครื่องมือดังกล่าวข้างต้นมีรูปแบบการใช้งานดังต่อไปนี้

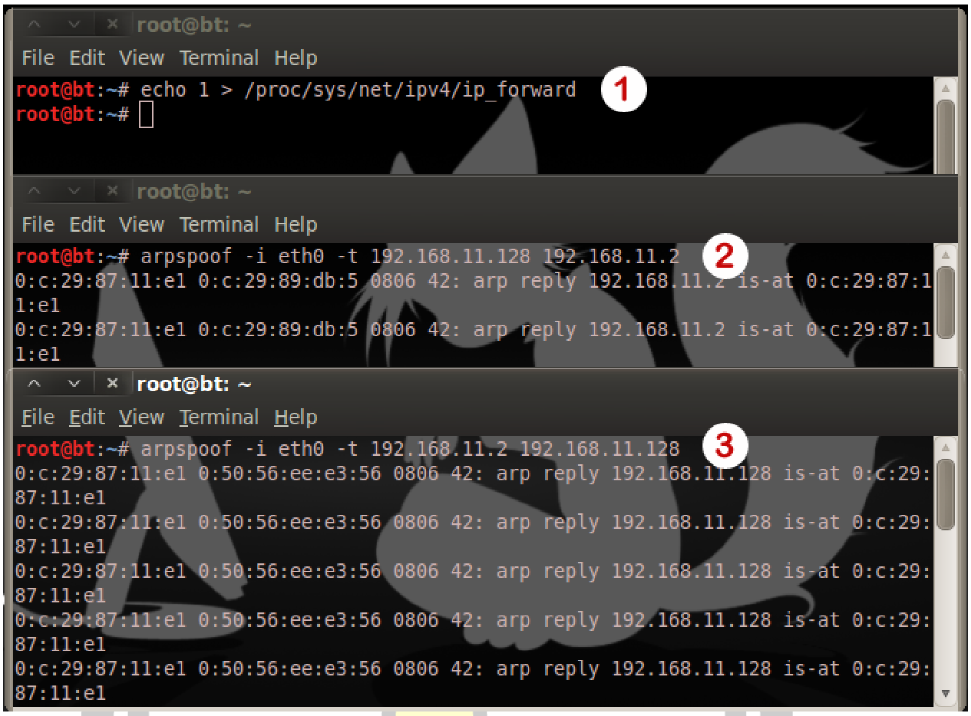
2.5.1 การโจมตีแบบแทรกกลางการสื่อสารด้วย Kali linux

การโจมตีแบบแทรกกลางการสื่อสารด้วย Kali linux นั้น คำสั่งในการใช้งานจะเป็นในลักษณะแบบคอมมานไลน์ (Command line) เป็นหลัก มีขั้นตอนในการโจมตีดังต่อไปนี้

1) ใช้คำสั่ง echo 1>/proc/sys/net/ipv4/ip_forward เพื่อกำหนดให้ระบบทำการส่งต่อข้อมูลของเหยื่อที่ผ่านเข้ามายังเครื่องผู้โจมตีส่งต่อไปที่เว็บเซิร์ฟเวอร์

2) ใช้คำสั่ง arpspoof -t [ไอพีของเครื่องเหยื่อ] [ไอพีเกตเวย์ของเครื่องเหยื่อ] เพื่อทำการหลอกเกตเวย์ว่าเครื่องผู้โจมตีนั้นเป็นเครื่องเหยื่อ

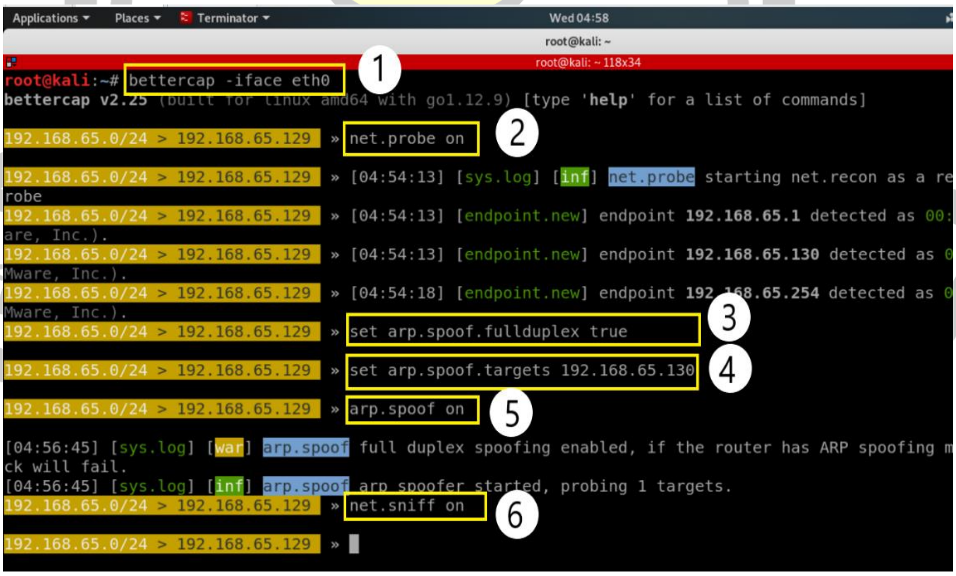
3) ใช้คำสั่ง arpspoof -t [ไอพีเกตเวย์ของเครื่องเหยื่อ] [ไอพีของเครื่องเหยื่อ] เพื่อหลอกเครื่องเหยื่อว่าเครื่องผู้โจมตีเป็นเกตเวย์ ดังภาพที่ 2.6



ภาพที่ 2.6 การโจมตีแบบแทรกกลางการสื่อสารด้วย Backtrack

2.5.2 การโจมตีแบบแทรกกลางการสื่อสารด้วย Bettercap: ARP Spoofing

การโจมตีแบบแทรกกลางการสื่อสารด้วย Bettercap: ARP Spoofing นั้น คำสั่งในการใช้งานจะเป็นในลักษณะแบบคอมมานไลน์ (Command line) เป็นหลัก มีขั้นตอนในการโจมตีดังภาพที่ 2.7



ภาพที่ 2.7 การโจมตีแบบแทรกกลางการสื่อสารด้วย Bettercap: ARP Spoofing

1) ใช้คำสั่ง bettercap –iface eth0 เพื่อทำการเลือกอุปกรณ์อินเตอร์เฟซ

```
# bettercap -iface eth0
```

2) ใช้คำสั่ง net.probe on เพื่อทำการค้นหาโฮสต์เครือข่ายเดียวกันโดยพื้นฐานจะส่งเป็นแพ็คเก็ต UDP แบบสุ่มไปยังทุก IP ที่เป็นไปได้ในซับเน็ต สามารถเรียกดูผลลัพธ์โดยใช้คำสั่ง net.show จะแสดงผลแบบเรียลไทม์

```
# net.probe on
```

3) ใช้คำสั่ง **set arp.spoof.fullduplex true** เพื่อทำการหลอกเกตเวย์ว่าเครื่องผู้โจมตีนั้นเป็นเครื่องเหยื่อ และหลอกเครื่องเหยื่อว่าเครื่องผู้โจมตีเป็นเกตเวย์

```
# net.probe on
```

4) ใช้คำสั่ง **set arp.spoof.targets 192.168.65.130** เพื่อทำการหนด IP เป้าหมายที่จะโจมตี

```
# set arp.spoof.targets 192.168.65.130
```

5) ใช้คำสั่ง **arp.spoof on** เพื่อทำการ start ARP spoofer เริ่มกระบวนการปลอมแปลง

```
# arp.spoof on
```

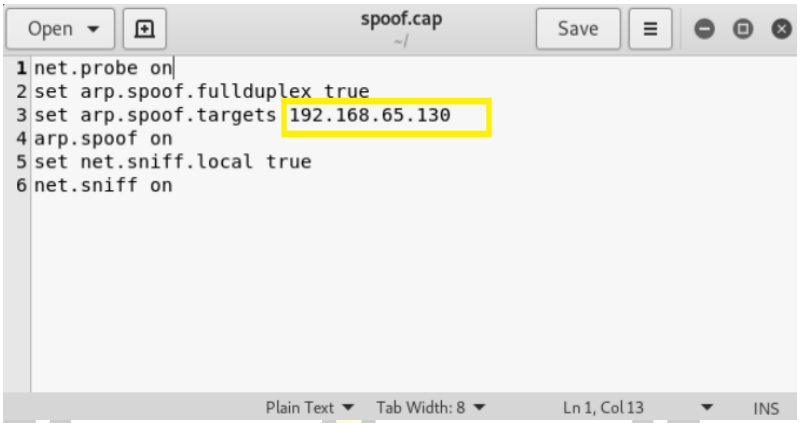
6) ใช้คำสั่ง net.sniff on เพื่อทำการดักจับข้อมูลในเครือข่ายอินเทอร์เน็ต

```
# net.sniff on
```

2.5.3 การโจมตีแบบแทรกกลางการสื่อสารด้วย Bettercap: Bypassing HTTPS

การโจมตีแบบแทรกกลางการสื่อสารด้วย Bettercap: Bypassing HTTPS จะอยู่ในรูปแบบของการลดระดับโพรโทคอล HTTPS ให้เป็น HTTP คำสั่งในการใช้งานจะเป็นในลักษณะแบบคอมมานไลน์ (Command line) เป็นหลัก มีขั้นตอนในการโจมตีดังต่อไปนี้

1) ให้ทำการสร้าง script โดยการเขียนคำสั่งเพื่อทำการแทรกกลางการสื่อสารโดยตรง สามารถใช้ Text Editor ใน Kali linux ได้ ดังแสดงในภาพที่ 2.8

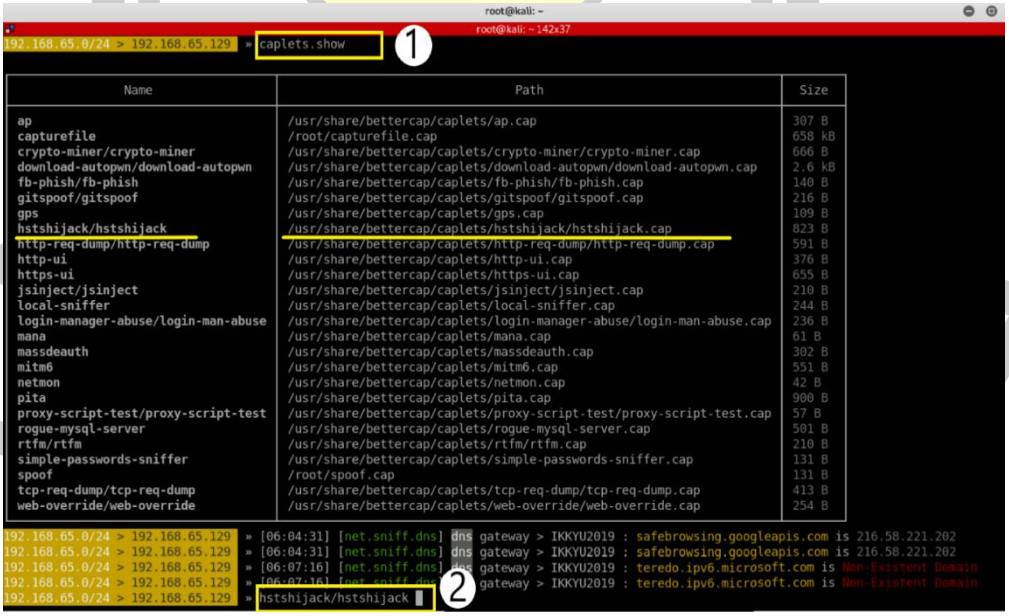


ภาพที่ 2.8 สร้าง script เพื่อทำการแทรกกลางการสื่อสาร

2) ให้ใช้คำสั่ง **bettercap -iface eth0 -caplet /root/spoof.cap** เพื่อทำการ Run Script ที่ได้ทำการบันทึกไว้ก่อนหน้านี้

```
# bettercap -iface eth0 -caplet /root/spoof.cap
```

3) ให้ใช้คำสั่ง caplets.show เพื่อเช็คความพร้อมในการทำงาน หลังจากนั้นให้ใช้คำสั่ง hstshijack/hstshijack เพื่อทำการแทรกกลางการสื่อสารและทำการ Bypass HTTPS ดังแสดงในภาพที่ 2.9

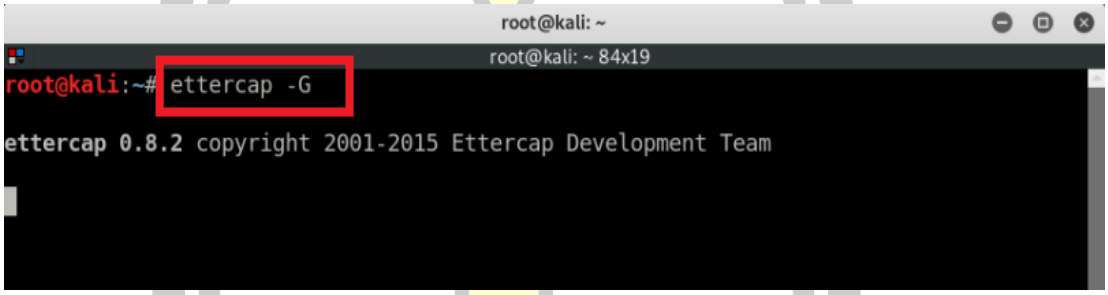


ภาพที่ 2.9 คำสั่งในการ Bypass HTTPS

2.5.4 การโจมตีแบบแทรกกลางการสื่อสารด้วย Ettercap

การโจมตีแบบแทรกกลางการสื่อสารด้วย Ettercap จะเป็นการใช้งานคำสั่งจะอยู่บนรูปแบบของ GUI ซึ่งมีขั้นตอนในการโจมตีดังต่อไปนี้

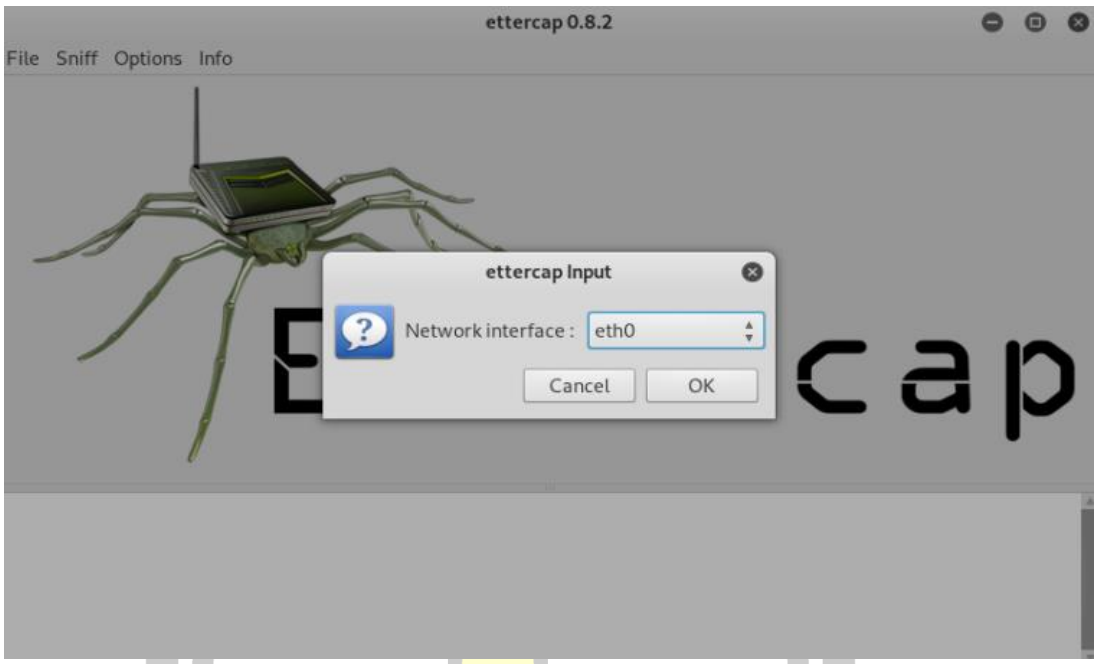
1) ใช้คำสั่ง ettercap –G เพื่อเรียกใช้งานโปรแกรม Ettercap ในโหมดของ GUI ดังแสดงในภาพที่ 2.10



ภาพที่ 2.10 คำสั่งในการเรียกใช้ Ettercap ในโหมด GUI

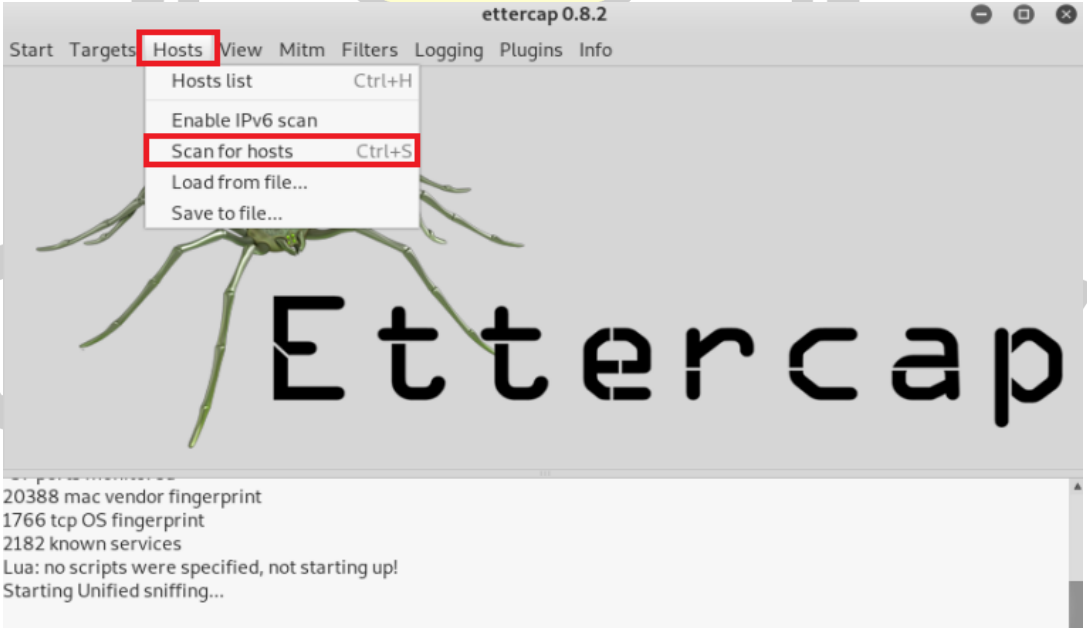
2) คลิกเมนู Sniff ตามด้วยคลิก Unified Sniffing หรือกดปุ่ม Shift+U เพื่อกำหนดการ์ดเน็ตเวิร์กให้กับโปแกรม ซึ่งปกติจะถูกกำหนดให้เป็น “eth0” ดังแสดงในภาพที่ 2.11 และภาพที่ 2.12

ภาพที่ 2.11 การกำหนดโหมดของการดักจับข้อมูล

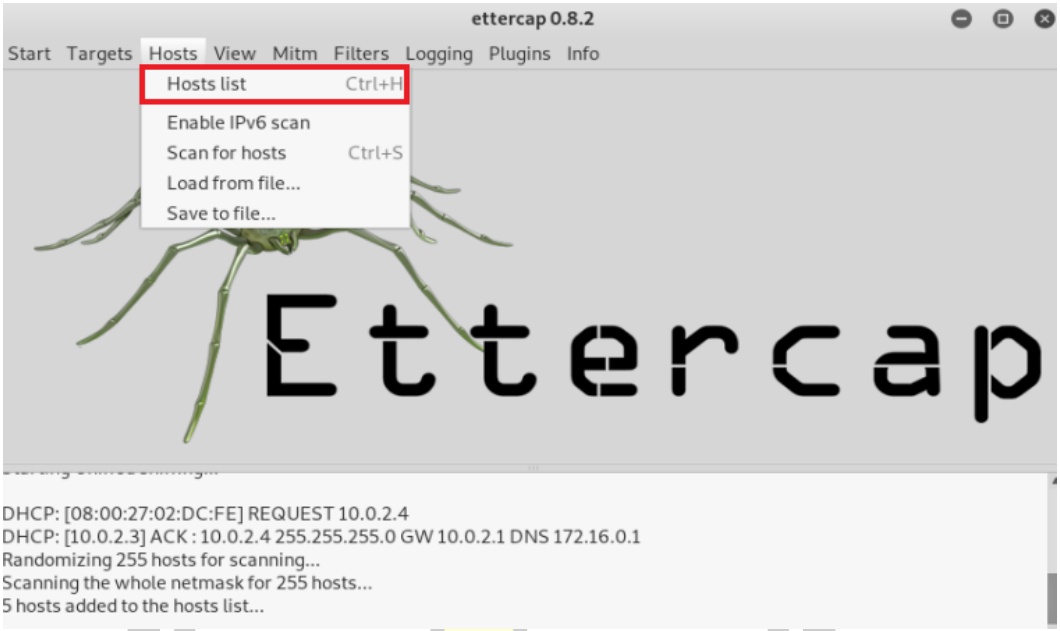


ภาพที่ 2.12 การกำหนดการ์ดเน็ตเวิร์กให้กับโปรแกรม Ettercap

3) คลิกเมนู Hosts ตามด้วยคลิกที่ Scan for Hosts หรือกดปุ่ม Ctrl+S เพื่อค้นหาเครื่องเหยื่อ จากนั้นคลิกเลือก Hosts List เพื่อแสดงไอพีของเกตเวย์และเครื่องเหยื่อ ดังแสดงในภาพที่ 2.13 และภาพที่ 2.14

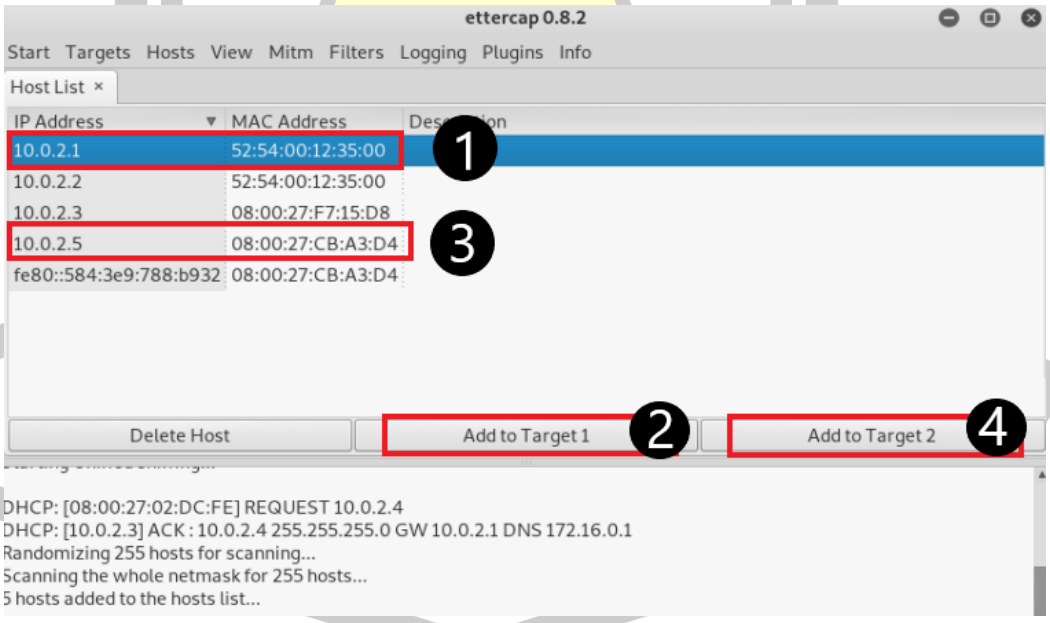


ภาพที่ 2.13 ค้นหาเป้าหมายที่จะทำการโจมตี



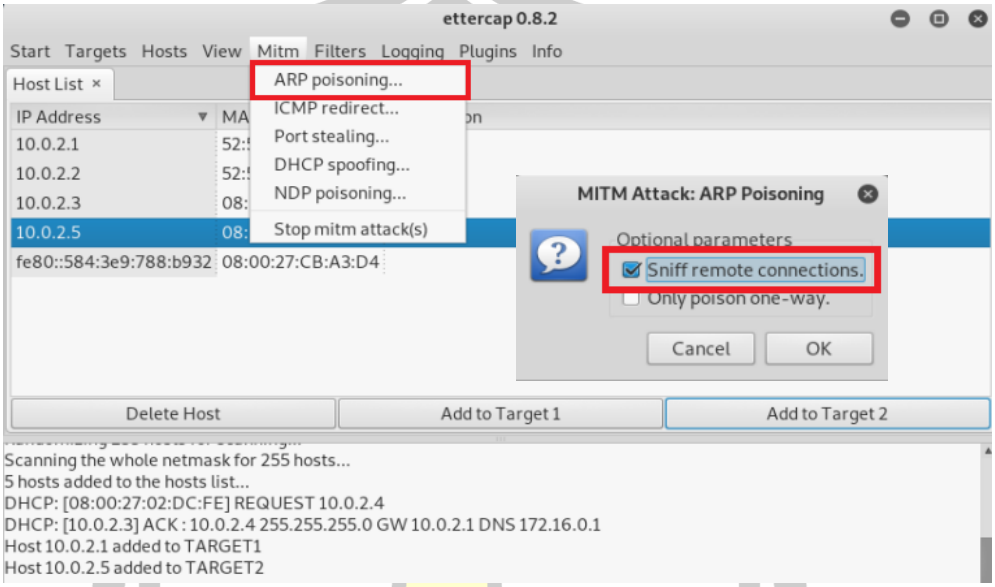
ภาพที่ 2.14 กำหนดให้ Ettercap แสดงผลลัพธ์ของการค้นหา

4) เลือกไอพีของเกตเวย์ แล้วคลิกปุ่ม Add to Target 1 จากนั้นเลือกไอพีของเครื่องเหยื่อ แล้วคลิกปุ่ม Add to Target 2 ดังภาพที่ 2.15



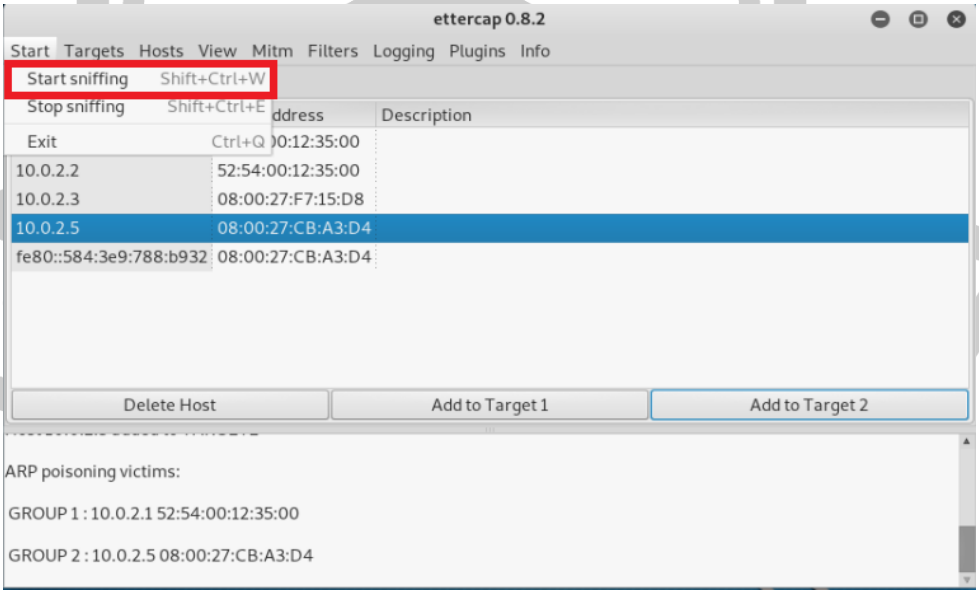
ภาพที่ 2.15 กำหนดไอพีของเกตเวย์และไอพีของเป้าหมายในการโจมตี

5) เริ่มการโจมตีแทรกกลางการสื่อสารระหว่างเครื่องเหยื่อกับเกตเวย์ โดยคลิกที่เมนู Mitm ตามด้วย Arp poisoning แล้วคลิกเลือก Sniff remote connections ดังภาพที่ 2.16



ภาพที่ 2.16 กำหนดการโจมตีแบบแทรกกลางการสื่อสาร

6) คลิกเลือกเมนู Start ตามด้วยคลิก Start sniffing เพื่อโจมตีเครื่องเหยื่อ ดังภาพที่ 2.17

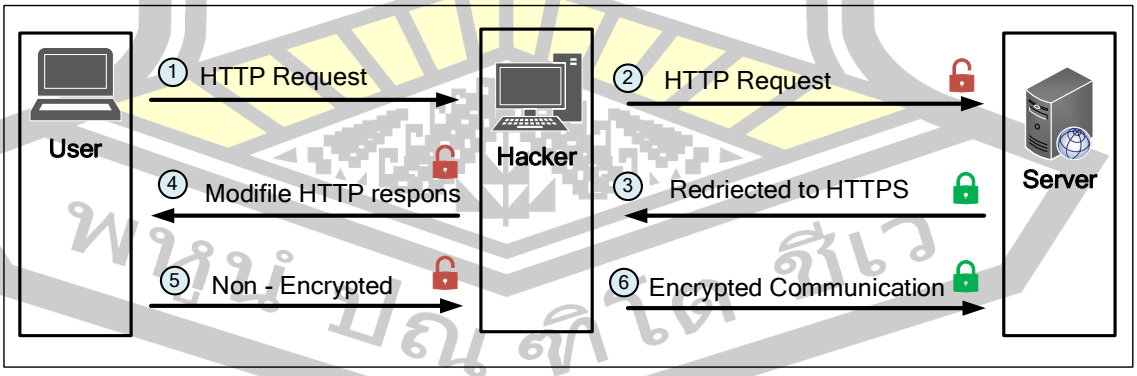


ภาพที่ 2.17 เริ่มการโจมตีแบบแทรกกลางการสื่อสาร

2.6 การโจมตี SSL Stripping Attack

ในปี ค.ศ. 2009 Marlinspike [4] ได้เสนอวิธีการโจมตี SSL ด้วยวิธี SSL Stripping Attack ในงาน Blackhat Conference 2009 โดยใช้ SSL Strip ซึ่งเป็นเครื่องมือที่ถูกติดตั้งใน Backtrack Linux ซึ่งเป็น Linux-base Penetration Testing ที่ถูกกลุ่มผู้เชี่ยวทางด้านระบบเครือข่ายทั่วโลกนำไปใช้งานในงานทางด้านการทดสอบเจาะและประเมินความปลอดภัยของระบบเครือข่าย ซึ่งรวมถึงการถูกนำไปใช้โดยผู้ที่ไม่ประสงค์ดีเช่นเดียวกัน

การเปลือยเอสเอสแอล หรือ SSL Stripping Attack มีรูปแบบการโจมตีโดยอาศัยวิธีโจมตีแบบแทรกกลางการสื่อสารร่วมกับวิธีการโจมตีแบบ SSL Stripping Attack ควบคู่กัน โดยการโจมตีเว็บไซต์ที่มีการกำหนดช่องทางการสื่อสารผ่านโพรโทคอล HTTPS เมื่อเครื่องเหยื่อถูกแฮกเกอร์โจมตีเว็บเบราว์เซอร์จะถูกบังคับเปลี่ยนการทำงานที่โพรโทคอล HTTP ทำให้ข้อมูลไม่ได้รับการเข้ารหัสการสื่อสาร โดยข้อมูลต่าง ๆ ที่เครื่องเหยื่อส่งไปที่เว็บเซิร์ฟเวอร์จะถูกส่งผ่านไปยังเครื่องแฮกเกอร์ก่อน ซึ่งแฮกเกอร์สามารถดักจับข้อมูลของเหยื่อได้อย่างง่ายดาย เนื่องจากข้อมูลที่สื่อสารผ่าน HTTP อยู่ในรูปของ Clear Text ที่สามารถอ่านเข้าใจได้ หลังจากนั้นการทำงานต่อไปของ SSL Stripping Attack จะทำหน้าที่นำข้อมูลของเหยื่อมาเข้ารหัสด้วย HTTPS แล้วส่งต่อไปที่เว็บเซิร์ฟเวอร์ ด้วยเหตุนี้ผลของการโจมตีที่เว็บเบราว์เซอร์ของเครื่องเหยื่อจึงไม่สามารถตรวจสอบหรือแสดงข้อความแจ้งเตือนความผิดพลาดได้ เนื่องจากเครื่องของเหยื่อสามารถสื่อสารกับเว็บเซิร์ฟเวอร์ได้ตามปกติ เพียงแต่เป็นการสื่อสารที่ถูกบังคับให้อยู่บนโพรโทคอล HTTP แทนที่จะเป็นโพรโทคอล HTTPS ที่มีการทำงานอย่างปลอดภัย ดังแสดงภาพที่ 2.18



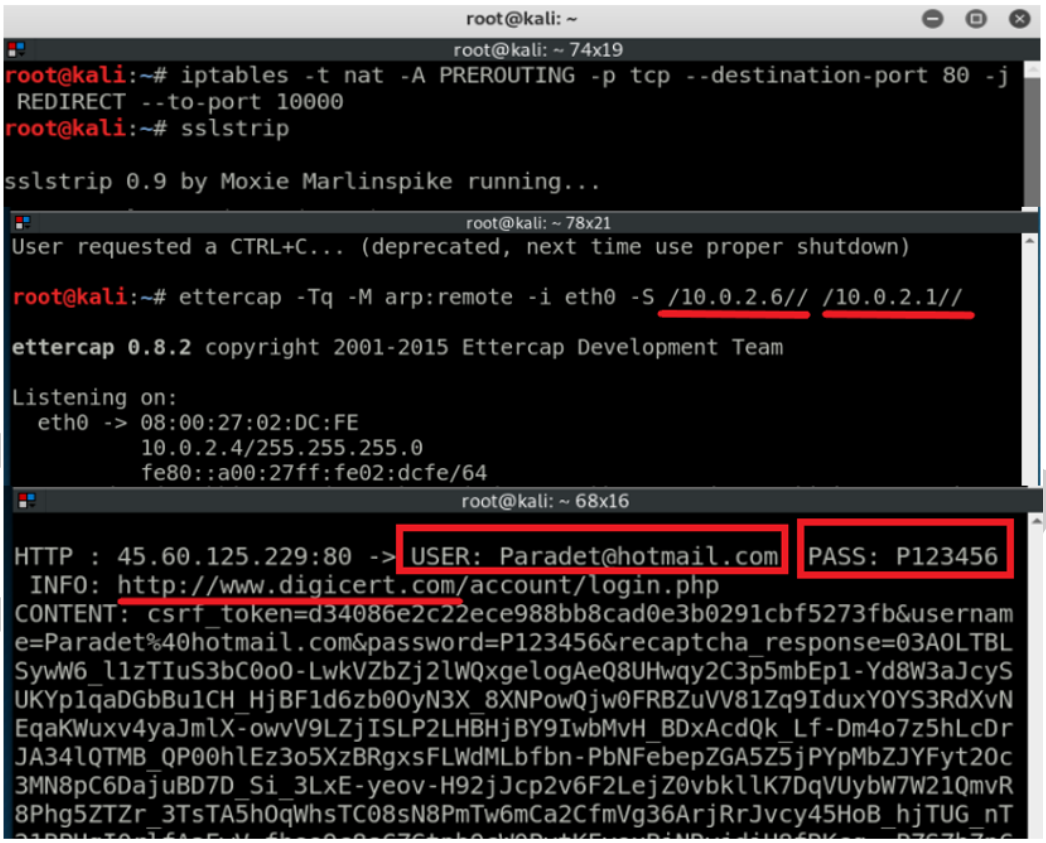
ภาพที่ 2.18 รูปแบบการโจมตี SSL Stripping Attack

ในงานวิจัยนี้จะทำการสาธิตการโจมตีระบบเว็บไซต์ที่ใช้งาน SSL/TLS ด้วยเครื่องมือ SSL Strip ที่ถูกติดตั้งอยู่บนระบบปฏิบัติการ Kali Linux เวอร์ชัน 2019.3 ดังภาพที่ 2.19 ซึ่งมีขั้นตอนดังต่อไปนี้

1) ใช้คำสั่ง iptables –t nat –A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 10000 เพื่อการกำหนดให้ข้อมูลที่เข้ามายังเครื่องผู้โจมตีทาง Port หมายเลข 80 ให้ส่งต่อไปที่ Port หมายเลข 10000

2) ใช้คำสั่ง sslstrip เพื่อโจมตีเว็บไซต์ที่ใช้งาน SSL/TLS เมื่อเครื่องของเหยื่อเข้าใช้งานเว็บไซต์ แล้วตามปกติบนเว็บเบราว์เซอร์จะแสดงโพรโทคอล HTTPS แต่เมื่อถูกโจมตีด้วยวิธี SSL Stripping Attack นี้แล้ว ผลของการโจมตีจะส่งผลให้บนเว็บเบราว์เซอร์ถูกกำหนดให้ใช้งานโพรโทคอล HTTP แทน

3) ใช้คำสั่ง Ettercap -Tq -M arp:remote -I eth0 -S /10.0.2.6// /10.0.2.1// เพื่อแสดงข้อมูลของเครื่องเหยื่อที่สามารถดักจับได้ เช่น ชื่อบัญชีผู้ใช้และรหัสผ่านที่ใช้ในการตรวจสอบสิทธิ์ในการเข้าใช้งานระบบ ดังภาพที่ 2.19



ภาพที่ 2.19 การโจมตี SSL/TLS ด้วยวิธี SSL Stripping Attack

2.7 Hashcat

Hashcat [32] เป็นโปรแกรม Open Source Software ถูกสร้างขึ้นเพื่อใช้ในการถอดรหัส Password Cracking โดยสามารถใช้ถอดค่า Hash Algorithm ได้หลากหลาย เช่น MD5, SHA1, SHA256, HMAC, WPA, JWT รวมถึง BitCoin, Ethereum และยัง Support ทั้ง CPU และ GPU โดยสามารถเลือกประเภทการโจมตีในการถอดรหัส Password Cracking ที่ได้รับความนิยมได้ เช่น 1) Dictionary Attack เป็นการสุ่มเดา Password จากไฟล์ที่มีการรวบรวมคำศัพท์ต่างๆ ที่พบอยู่ใน Dictionary ซึ่งจัดว่าเป็นวิธีการที่ถูกนำมาใช้ในการถอดรหัสผ่านมากที่สุด 2) Brute Force Attack เป็นการเดา password ทุกความเป็นไปได้ของตัวอักษรในแต่ละหลัก เช่น รหัส ATM มีจำนวน 4 หลัก แต่ละหลักสามารถตั้งค่าตัวเลข 0-9 ดังนั้น โปรแกรมจะทำการไล่ตัวเลขจาก 0000 ไปจนถึง 9999 หมื่นวิธีจนได้ password ที่ถูกต้องเป็นต้น

2.8 งานวิจัยที่เกี่ยวข้อง

Hodges และคณะ [10] ได้เสนอ HTTP Strict Transport Security (HSTS) ได้กำหนดให้เป็นกลไกมาตรฐานการสื่อสาร IETF ตามเอกสาร RFC 6797 เพื่อแก้ปัญหาการถูกโจมตี HTTPS เช่น การโจมตีด้วยวิธี SSL Stripping Attack การทำงานของ HSTS ในเว็บเบราว์เซอร์ (Web Browser) ล่าสุดมีการรองรับการทำงานสำหรับเว็บเบราว์เซอร์หลักทุกตัว เช่น Google Chrome, Mozilla Firefox, Safari, Operra, IE เป็นต้น รูปแบบการทำงานโดยเซิร์ฟเวอร์ (Server) จะมีการตอบกลับ (Response) ในส่วนของ HTTP Header เมื่อเว็บเบราว์เซอร์ตรวจสอบพบ HTTP Header ชื่อ Strict-Transport-Security: max-age=31536000; include SubDomains ซึ่งเป็นส่วนหัวที่สำคัญสำหรับบอกเว็บเบราว์เซอร์เพื่อให้เข้าใจว่าเว็บไซต์ต้องการเรียกใช้ HSTS และยังมีส่วนที่เป็นค่าเวลาในการกำหนดการเชื่อมต่อ HSTS เพื่อบังคับใช้โพรโทคอล HTTPS โดยเว็บเบราว์เซอร์จะทำการบังคับใช้โพรโทคอล HTTPS ตามระยะเวลาที่กำหนดไว้ใน HTTP Header นอกจากนี้ยังมีการเรียกใช้งาน HSTS Preload ลักษณะการทำงานคือจะมีการติดตั้ง HSTS ไว้ที่เว็บเบราว์เซอร์ทำให้เว็บไซต์ที่อยู่ในรายการ HSTS Preload มีประสิทธิภาพในการป้องกันการถูกโจมตี เช่น Google, Paypal, Twitter, facebook, Simple, Linode, Stripe, Lastpass เป็นต้น การทำงาน HSTS enforced on specific names คือการดูแลแบบพิเศษรวมถึงชื่อโดเมนของเว็บไซต์ทั้งหมดตัวอย่างของ facebook ดังแสดงในภาพที่ 2.20

```
{
  "name": "facebook.com", "policy": "custom",
  "mode": "force-https", "pins": "facebook", "include_subdomains_for_pinning": true
},
{ "name": "www.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "m.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "tablet.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "secure.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "pixel.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "apps.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "upload.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "developers.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "touch.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "mbasic.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "code.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "t.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "mtouch.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "business.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "research.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{
  "name": "messenger.com", "policy": "custom",
  "mode": "force-https", "pins": "facebook", "include_subdomains_for_pinning": true
},
{ "name": "www.messenger.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
```


ภาพที่ 2.20 HSTS enforced on specific names

ที่มา: [33]

ปัจจุบันทาง Google ในโครงการ chromium ที่เป็นองค์กรดูแล HSTS Preload ได้เปิดให้เว็บไซต์ทั่วไปลงทะเบียนใช้งาน HSTS Preload ที่ต้องการความมั่นคงในการสื่อสารอินเทอร์เน็ต การบังคับใช้โพรโทรคอล HTTPS สำหรับการสื่อสารผ่านเว็บเบราว์เซอร์จากการเช็ครายชื่อที่มีการลงทะเบียนใช้งานทั้งหมด เมื่อวันที่ 4 พ.ย 2562 จากไฟล์ transport_security_state_static.json จากทั่วโลกที่มีการลงทะเบียนใช้งาน HSTS Preload พบว่ามีเว็บไซต์ทั้งหมดประมาณ 88,803 รายชื่อโดเมน และพบหน่วยงานไทยที่มีการลงทะเบียน HSTS Preload ดังแสดงใน
ตารางที่ 2.1

ตารางที่ 2.1 ชื่อเว็บไซต์ที่ถูก HSTS Preload .or.th และ .ac.th

No	สกุลเว็บไซต์ .or.th และ .ac.th
1	www.etda.or.th
2	san.ac.th

ตารางที่ 2.2 ชื่อเว็บไซต์ที่ถูก HSTS Preload .co.th

No	สกุลเว็บไซต์ .co.th
1	insightera.co.th
2	ginja.co.th
3	infura.co.th
4	cipher.co.th
5	dotbrick.co.th
6	odoo.co.th
7	officeprint.co.th
8	officeprint.co.th
9	extreme.co.th
10	nexthop.co.th
11	dotsiam.co.th
12	saleduck.co.th
13	vsoy.co.th
14	begintravel.co.th
15	thaihong.co.th
16	jaisiam.co.th

จากข้อมูลข้างต้นถือว่าน้อยมากสำหรับเว็บไซต์ที่ต้องการความมั่นคงสูง หลังจากนั้นยังได้ทำการเช็คความมั่นคงหน่วยงานไทยที่ต้องการความมั่นคงสูง เช่น ธนาคารออนไลน์ (Internet Banking) ที่มีการเปิดใช้งาน HSTS โดยเลือกธนาคารออนไลน์รายหลักของไทยจำนวน 11 เว็บไซต์ โดยใช้เทคนิคการโจมตี SSL Strip พบว่า สามารถป้องกันการโจมตีได้เพียง 1 เว็บไซต์เท่านั้น และอีก 10 เว็บไซต์ธนาคารออนไลน์พบว่าล้มเหลวในการป้องการการถูกโจมตี HTTPS จากการทดสอบพบว่าเว็บไซต์ธนาคารออนไลน์ที่รอดจากโจมตีนั้น ได้ตั้งค่าตามมาตราฐาน HSTS ที่ถูกต้องและมีการลงทะเบียนเปิดใช้งาน HSTS Preload ตามนโยบายของ Google ซึ่งเป็นวิธีที่ดีที่สุดในปัจจุบัน

Fairweather และคณะ [34] ได้นำเสนอ CAT + S ที่ถูกพัฒนาระบบด้วยภาษา JavaScript โดยทำการพัฒนาระบบขึ้นเป็น Browser Extension ที่ทำงานอยู่บน Google Chrome เพื่อป้องกันการโจมตีแบบแทรกกลางการสื่อสารและการโจมตีด้วย SSL Stripping Attack โดยมีการ

พัฒนาให้ระบบสามารถตรวจสอบความไม่ปลอดภัยของเว็บไซต์อัตโนมัติ โดยใช้ JavaScript ในการเข้าถึงองค์ประกอบของเว็บไซต์ผ่าน HTML สร้างอินสแตนซ์ LinkMonitor เพื่อจัดเก็บรูปแบบ HTML ทั้งหมดที่ใช้ฟังก์ชัน getElementsBy TagName กระบวนการนี้จะทำซ้ำทุก ๆ 100 วินาที เป็นการตรวจสอบแบบ Dynamic เพื่อหาการเปลี่ยนแปลงของโพรโทคอลที่ผิดปกติของเว็บไซต์ ในการตรวจสอบว่าเว็บไซต์ที่กำลังใช้งานอยู่นั้นทำงานอยู่บนโพรโทคอล HTTPS หรือไม่ และหากพบว่ามีการสื่อสารผ่านโพรโทคอล HTTP ระบบ CAT + S จะเปลี่ยนการสื่อสารเป็น HTTPS อัตโนมัติโดยที่ผู้ใช้ไม่ต้องดำเนินการใด ๆ ทั้งสิ้น และยังสามารถรักษาความปลอดภัยแบบฟอร์มที่มีการสื่อสารผ่าน HTTP ในเว็บไซต์ให้คงเดิมให้อีกด้วย ในการนำมาใช้จริงระบบ CAT + S พบว่ายังเป็นขอเสนอที่ใช้ป้องกันการถูกโจมตีด้วย SSL Stripping Attack เท่านั้น และจากการวิเคราะห์ระบบ CAT + S ยังพบว่ามีข้อจำกัดเรื่องของเวลาในการโหลดข้อมูลมาเช็คเพื่อตรวจสอบความผิดปกติของโพรโทคอล HTTPS ในระบบเว็บไซต์ หากถูกโจมตีด้วย SSL Stripping Attack จะทำให้ระบบมีการตรวจสอบวนลูปในการคืนค่าระหว่างโพรโทคอล HTTP ไปยังโพรโทคอล HTTPS ซึ่งจะส่งผลให้ระบบเว็บไซต์ทำงานต่อไม่ได้จึงเกิดเหตุการณ์ที่เรียกว่า DoS (Denial-of-Service)

Selvi [35] ได้เสนอวิธีการโจมตี HSTS ใน Blackhat Conference โดยมีการวิเคราะห์ข้อดีและข้อเสียของ HSTS แล้วเสนอวิธีการโจมตีโดยอาศัยการโจมตีที่ Network Time Protocol (NTP) ของเครื่องเหยื่อเพื่อเปลี่ยนแปลงเวลาบนเครื่องเหยื่อให้เพิ่มมากขึ้นทาให้การทำงานของ HSTS มองว่าค่าใน Parameter ที่ชื่อ max-age หมดอายุ ดังนั้นเมื่อ Web Browser เปรียบเทียบเวลาที่ระบุใน max-age กับเวลาปัจจุบันบนเครื่องเหยื่อแล้วได้ผลว่าเวลาใน max-age หมดอายุการประมวลผลของ Web Browser ก็จะไม่ทำตามเงื่อนไขของ HSTS แล้วผู้โจมตีสามารถใช้วิธีโจมตีแบบ SSL Strip ได้สำเร็จ โดยในงานวิจัยได้พัฒนาเครื่องมือชื่อ Delorean แล้วทดสอบโจมตีทั้งในระบบปฏิบัติการ Ubuntu Linux, Fedora Linux, Mac OS X Lion, Mac OS X Mavericks และ Microsoft Windows และกับ Web Browser ได้แก่ Safari, Firefox และ Google Chrome

Fung และคณะ [5] ได้เสนอ SSLock โดยเป็นข้อเสนอวิธีการบังคับใช้โพรโทคอล SSL กับเว็บไซต์ โดยการพิจารณาการแบ่ง Domain Name สำหรับการใช้งาน SSL โดยในการใช้งานนั้นผู้พัฒนาเว็บไซต์ต้องกำหนดรูปแบบการทำงานของระบบเว็บไซต์ตามกระบวนการที่ SSLock เสนอเพื่อตอบกลับ HTTP Header ที่ชื่อ SSLock-Candidates ซึ่งทำหน้าที่จัดเก็บค่าของ Domain Name เช่น gmail.com พร้อมด้วย Javascript ที่ใช้อ่านค่า HTTP Header จากนั้นในการทำงานของเว็บเบราว์เซอร์ของไคลเอนต์จะทำการเปลี่ยนแปลงชื่อ Domain Name เสียใหม่และทำการส่งคำขอไปที่เว็บเซิร์ฟเวอร์ด้วย Domain Name ที่มีความมั่นคงปลอดภัยไปบนโพรโทคอล HTTPS เช่น https://secure.gmail.com แต่วิธีการป้องกันการโจมตีของ SSLock นี้ยังมีข้อเสียในด้านของการ

พัฒนาระบบให้เป็นมาตรฐาน ซึ่ง SSLock สามารถทำได้เพียงการตรวจสอบการโจมตีเท่านั้น แต่ไม่สามารถป้องกันการโจมตีด้วยวิธี SSL Stripping Attack ได้

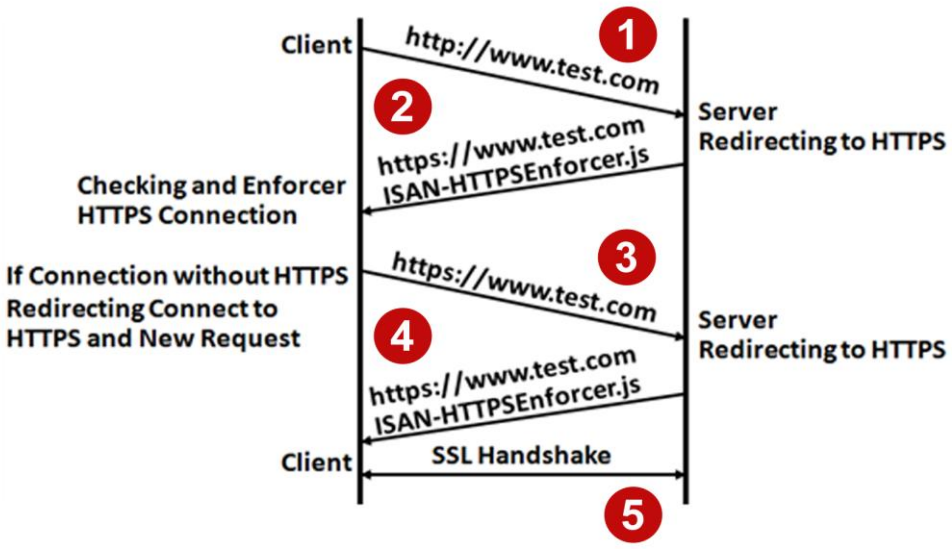
Fung และคณะ [7] ได้เสนอ HTTPSLock โดยเป็นข้อเสนอวิธีการบังคับใช้งาน HTTPS สำหรับเว็บไซต์ที่มีการใช้งาน Certificate ใบรับรองที่ถูกต้อง โดยอาศัย JavaScript ในการตรวจสอบ Certificate หากเว็บไซต์ที่กำลังใช้งานถูกตรวจสอบแล้วพบว่ามีสถานะเป็น ใบรับรองที่ไม่ถูกต้อง ซึ่งอาจเกิดจากการถูกโจมตี SSL/TLS ด้วยวิธี SSL Sniff การทำงานของ HTTPSLock จะไม่อนุญาตให้มีการใช้งานเว็บไซต์ดังกล่าว และสำหรับเว็บไซต์ที่มีการใช้งาน ใบรับรองที่ไม่ถูกต้อง แต่เมื่อเข้าใช้งานเว็บไซต์โดยที่การแสดงผลของโพรโทคอลใน URL ที่ช่อง Address Bar บนเว็บเบราว์เซอร์เป็น HTTP ซึ่งแทนที่จะเป็น HTTPS การทำงานของ HTTPSLock ก็จะแสดงผลของการตรวจสอบว่า ผู้ใช้กำลังใช้งานเว็บไซต์ที่ไม่ถูกต้องและมีความเสี่ยง ซึ่งยังพบว่าวิธีการทำงานของระบบ HTTPSLock ยังมีข้อเสียอยู่ เพราะสามารถตรวจสอบการโจมตี HTTPS ได้เท่านั้น ซึ่งแจ้งเตือนการโจมตีด้วย SSL Stripping Attack โดยเฉพาะรวมถึงปัญหาการใช้งาน เนื่องมาจากระบบรองรับการใช้งาน 70% ของเว็บเบราว์เซอร์ทั้งหมด ดังนั้นเมื่อถูกโจมตี จะส่งผลกระทบให้ผู้ใช้ไม่สามารถเข้าใช้งานเว็บไซต์ได้ตามปกติ

Puangpronpitag และ Sriwiboon [36] ได้เสนอ ISAN-HTTPS Enforcer ถูกพัฒนาขึ้นโดยใช้ JavaScript ซึ่งเป็นระบบต้นแบบที่สามารถทำงานได้กับทุกโปรแกรมเว็บเบราว์เซอร์ที่รองรับการใช้งาน JavaScript โดยไม่ต้องทำการปรับเปลี่ยนหรือติดตั้ง Plug-in เพิ่ม ขั้นตอนการทำการทำงานประกอบด้วย

1) เมื่อคำขอของผู้ใช้ถูกส่งไปยังเว็บไซต์ โดยทั่วไปจะไม่ได้ระบุ “https” ใน URL ในแถบของ Address Bar ตัวอย่างเช่น test.com หรือ www.test.com

2) ทางด้านฝั่งเว็บเซิร์ฟเวอร์ หากมีการร้องขอหน้าเว็บที่เป็นความลับ หรือต้องการความมั่นคงปลอดภัย เว็บเซิร์ฟเวอร์ก็จะเปลี่ยนเส้นทางการเชื่อมต่อจาก HTTP เป็นการเชื่อมต่อกับ HTTPS แทน หลังจากที่เว็บเซิร์ฟเวอร์ตอบสนองต่อข้อความที่ถูกร้องขอมา เช่น ข้อมูลของหน้าเว็บและ JavaScript ให้กับผู้ใช้ การสื่อสารระหว่างไคลเอนต์และเว็บเซิร์ฟเวอร์นั้นจะใช้เชื่อมต่อ HTTPS

3) ในกรณีหน้าเว็บถูกโหลดเสร็จสิ้นแล้ว และมีการสื่อสารอยู่บน HTTP ปกติ ISAN-HTTPS Enforcer ที่ทำงานทางด้านไคลเอนต์จะทำการตรวจสอบ URL ถ้ามีข้อมูลอยู่ในรายชื่อของการบังคับใช้ โพรโทคอล HTTPS แล้ว ISAN-HTTPS Enforcer จะทำการเปลี่ยนเส้นทางการเชื่อมต่อไปเป็น HTTPS ซึ่งอัลกอริทึมแสดงดังภาพที่ 2.21



ภาพที่ 2.21 แผนภาพการทำงานของ ISAN-HTTPS Enforcer

ที่มา: [8]

สมนึก พ่วงพรพิทักษ์ และอภิรักษ์ ทูลธรรม [37] ได้ทำการประเมินวิธีแก้ไขปัญหาการโจมตีด้วยการเปลือยเอสเอสแอล โดยคัดเลือกระบบป้องกันที่มีการเสนอขึ้นเพื่อทำการทดสอบ พบว่า ISAN-HTTPS Enforcer, HSTS และ SSLock มีความสามารถในการบังคับให้เว็บเบราว์เซอร์กลับมาใช้โพรโทคอล HTTPS อีกครั้ง เมื่อถูกโจมตีด้วยวิธี SSL Stripping Attack ในขณะที่ HProxy, HTTPSLock และ EV-SSL มีความสามารถในการตรวจจับการโจมตีแล้วแจ้งเตือนให้กับผู้ใช้ทราบ แต่ไม่ได้ป้องกันการโจมตี ในด้านของการใช้งานนั้น SSLock, HTTPSLock และ HSTS สามารถรองรับได้เฉพาะบางเว็บเบราว์เซอร์เท่านั้น ซึ่งในผลการทดลองได้แสดงให้เห็นว่า ISAN-HTTPS Enforcer สามารถใช้งานได้กับ Platform ที่หลากหลายของโปรแกรมเว็บเบราว์เซอร์และระบบปฏิบัติการ ในแง่ของการเป็นมิตรต่อผู้ใช้ ISAN-HTTPS Enforcer มีมากกว่า HSTS แม้แต่การพิมพ์ที่ “https://” ในช่องของ Address Bar ก็ยังให้ผลของ KLM ที่ดีกว่า แต่อย่างไรก็ตามวิธีที่นำเสนอนี้มี Overhead ในแง่ของการตอบสนองต่อเวลา (Response Time) อยู่บ้างเล็กน้อย ซึ่งสาเหตุมาจากเวลาที่ถูกใช้ไปในการประมวลผลของ JavaScript แต่ข้อเสียที่สำคัญที่สุดของงานวิจัยนี้ก็คือ หาก SSL Stripping Attack สามารถปลดการใช้งานโพรโทคอล HTTPS ออกได้ การปลด JavaScript ที่ใช้ในการป้องกันนี้ก็สามารถทำได้เช่นเดียวกัน โดยการแก้ไขโปรแกรม SSLStrip ซึ่งเป็น Python Script ให้ทำการลบ <script type="text/javascript" src="ISAN-HTTPSEnforcer.js"></script> ออก เมื่อ Tag ที่เรียกใช้งาน JavaScript ที่ใช้ป้องกันถูกปลดออกไปก็จะไม่มีระบบป้องกันที่มาทำหน้าที่ในการบังคับให้มีการใช้โพรโทคอล HTTPS อีก การจู่โจมก็จะสามารถกระทำได้

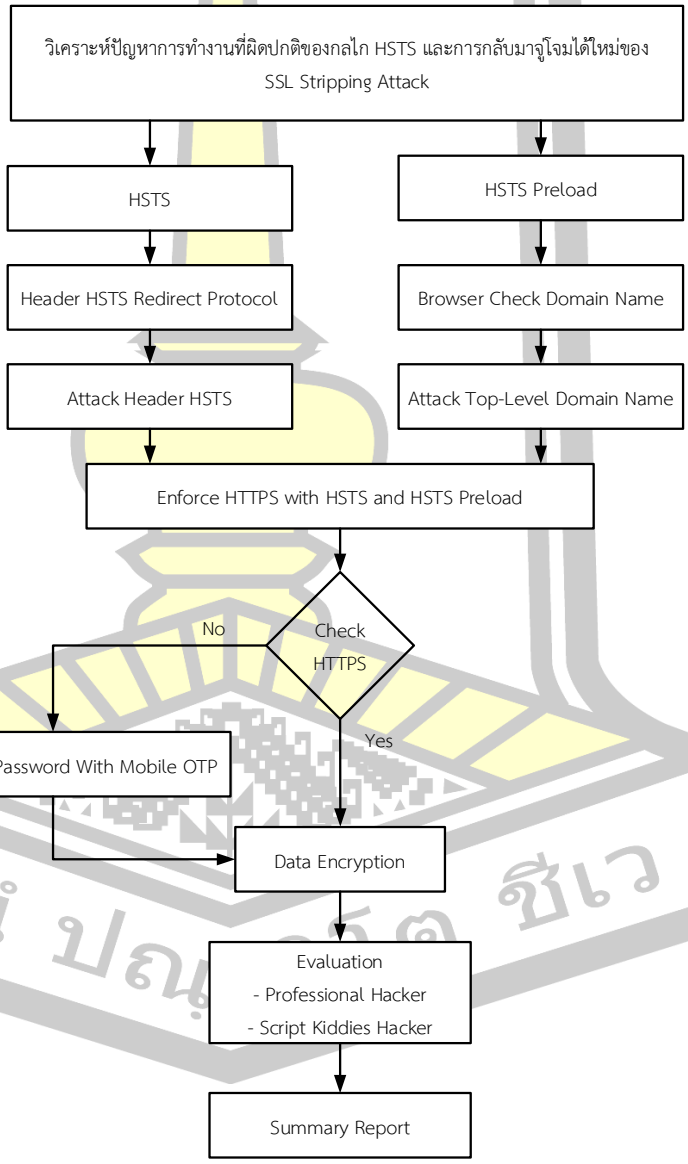
ณัฐวุฒิ ศรีวิบูลย์ และสมนึก พ่วงพรพิทักษ์ [38] ได้เสนอการแก้ไขปัญหาการโจมตีเว็บไซต์ที่ ทำงานบน HTTPS ด้วยวิธีการโจมตีแบบ SSL Stripping Attack โดยใช้การสังเกต EV-SSL และโพรโท- คอล HTTPS โดยทำการจัดเก็บรายชื่อเว็บไซต์ที่มีการกำหนดให้เรียกใช้โพรโทคอล HTTPS ไว้ใน เครื่องมือ Bookmark ของเว็บเบราว์เซอร์ เพื่อแก้ไขปัญหาของวิธีการป้องกันการโจมตี SSL ของ งานวิจัยที่ถูกนำเสนอก่อนหน้านี้ ที่ไม่สามารถป้องกันการโจมตี SSL ด้วยวิธี SSL Stripping Attack ได้ อย่างมีประสิทธิภาพ รวมถึงปัญหาความไม่สะดวกในการเรียกใช้งานของผู้ใช้ ปัญหาเรื่องความสามารถ ในการปรับใช้กับระบบเว็บไซต์เดิมที่มีอยู่ปัจจุบัน และปัญหาที่ระบบไม่รองรับการทำงานกับทุก โปรแกรมเว็บเบราว์เซอร์ ซึ่งจากผลการทดสอบของงานวิจัยแสดงให้เห็นว่า วิธีที่นำเสนอสามารถทำงาน ได้อย่างมีประสิทธิภาพ ผู้ใช้ได้รับความสะดวกในการใช้งานมากกว่าวิธีแก้ไขปัญหาแบบเดิม โดยทำการ วัดจากเวลาทั้งหมดในการทำกิจกรรมบนเว็บเบราว์เซอร์ ด้วยวิธี Keystroke-level Model (KLM) โดยวิธีการที่นำเสนอนี้สามารถรองรับการทำงานได้บนทุกเว็บเบราว์เซอร์ และไม่ต้องเปลี่ยนแปลงวิธีการทำงานของเว็บเซิร์ฟเวอร์แต่อย่างใด ซึ่งผลการทดสอบระบบป้องกันการโจมตี HTTPS ได้

แต่พบว่าการใช้งานวิธีดังกล่าวนี้ยังมีข้อเสียคือ ความยุ่งยากของผู้ใช้ ซึ่งจะต้องทำการบันทึก URL ที่ขึ้นต้นด้วย “https://” ของเว็บไซต์ที่ต้องการเรียกใช้ลงใน Bookmark ของโปรแกรมเว็บ เบราว์เซอร์เสมอทุกครั้งเมื่อมีการใช้งานเว็บไซต์ใหม่ ที่ยังไม่ได้ถูกบันทึกลงใน Bookmark List และใน กรณีที่ผู้ใช้ ใช้โปรแกรมเว็บเบราว์เซอร์หลายโปรแกรม ผู้ใช้ก็จะต้องทำการเพิ่มข้อมูลของ URL ลงใน Bookmark ของทุกโปรแกรมด้วย ทำให้สูญเสียเวลาไปโดยไม่จำเป็น

บทที่ 3

วิธีดำเนินการวิจัย

ในบทนี้จะกล่าวถึงกระบวนการวิเคราะห์ปัญหาการทำงานที่ผิดปกติของระบบ HSTS การกลับมาถูกโจมตีได้ใหม่ด้วยการเปลือยเอสเอสแอล (SSL Stripping Attack) กระบวนการทดสอบโจมตีและเสนอกลไกป้องกันการถูกดักจับข้อมูลเป็นกลไกชั้นที่ 2 ป้องกันกรณี HTTPS ถูกทำลาย ตลอดจนเครื่องมือและสภาพแวดล้อมที่ใช้ในการทดลอง



ภาพที่ 3.1 วิธีดำเนินการวิจัย

จาก**Error! Reference source not found.** เป็นวิธีดำเนินงานวิจัยเพื่อให้บรรลุตามวัตถุประสงค์สามารถออกแบบภาพรวมการดำเนินการวิจัยได้แก่ 1) การวิเคราะห์ปัญหาการทำงานที่ผิดปกติของกลไก HSTS และการกลับมาจู่โจมได้ใหม่ของการเปลือยเอสเอสแอล 2) การออกแบบพัฒนาระบบป้องกันการถูกดักจับข้อมูลกรณี HTTPS ถูกทำลาย 3) การทดสอบและประเมินประสิทธิผลของระบบป้องกัน ซึ่งรายละเอียดจะได้กล่าวต่อไปในรายงาน

3.1 การวิเคราะห์ตรวจสอบ HTTP Response Header และเว็บไซต์ที่ใช้ในการทดลอง SSL Stripping Attack

จากการที่ SSL Stripping Attack เป็นเทคนิคที่ถูกใช้งานอย่างแพร่หลายในการโจมตีระบบ Online Banking, ระบบ E-Commerce และ Web Applications อื่น ๆ ปัจจุบันการแก้ปัญหาที่ดีที่สุดและเป็นที่ยอมรับก็คือกลไก HSTS ซึ่งเว็บไซต์ส่วนใหญ่ที่ต้องการความมั่นคงสูงหันมาปรับใช้กันอย่างแพร่หลาย แต่กลับยังพบว่าระบบธนาคารออนไลน์ก็ยังมีผู้ไม่ประสงค์ดีโจมตีดักจับข้อมูล บัญชีผู้ใช้ และรหัสผ่าน ให้เห็นอยู่ในปัจจุบัน ดังนั้นงานวิจัยนี้จึงได้นำเสนอการประเมินปัญหาของ SSL Stripping Attack บนระบบเว็บไซต์ของ Online Banking, ระบบ E-Commerce, ระบบทะเบียนมหาวิทยาลัย ทำการทดลองบน Test-bed พร้อมทั้งเว็บเบราว์เซอร์ที่หลากหลายประกอบด้วย Google Chrome, Edge, Mozilla Firefox, Internet Explorer และ Safari ซึ่งผลที่ได้นี้จะสามารถนำมาวิเคราะห์ช่องโหว่ที่เกิดขึ้นและนำมาเป็นแนวทางในการแก้ไขปัญหาของ SSL Stripping Attack ได้ โดยมีรายละเอียดดังต่อไปนี้

1) กลุ่มตัวอย่างเว็บไซต์ที่ใช้งาน SSL/TLS ของระบบเว็บไซต์จำนวน 27 เว็บ ซึ่งประกอบด้วยระบบที่ให้บริการธนาคารออนไลน์ในไทย จำนวน 11 เว็บไซต์ โดยเลือกจากผู้ให้บริการในปัจจุบันซึ่งเป็นที่นิยมและจากฐานข้อมูล Internet Banking ของธนาคารแห่งประเทศไทย [39] ระบบที่ให้บริการ E-commerce จำนวน 5 เว็บไซต์ โดยการเลือกระบบของต่างประเทศ และในประเทศไทยที่ได้รับความนิยม ระบบทะเบียนมหาวิทยาลัย จำนวน 11 เว็บไซต์ โดยการเลือกกลุ่มมหาวิทยาลัยชั้นนำของประเทศไทยมาทำการทดสอบ

2) เพื่อให้เข้าใจปัญหาการทำงานของกลไก HSTS อย่างแท้จริง ในการทดลองนี้จึงได้ทำการตรวจสอบและวิเคราะห์ HTTP Response Header ของเว็บไซต์กลุ่มตัวอย่างทั้ง 28 เว็บไซต์ว่ามีการตั้งค่าปรับใช้กลไก HSTS หรือไม่อย่างไร

3) ทดลองโจมตีเว็บไซต์จากกลุ่มตัวอย่าง 27 เว็บไซต์ โดยใช้วิธี SSL Stripping Attack ในการโจมตี ซึ่งประเมินปัญหาโดยการใช้ระบบปฏิบัติการ Kali linux ในการโจมตีแบบแทรกกลางการสื่อสารและใช้ Wireshark ในการโจมตีเพื่อดักจับรหัสผ่านของเครื่องเป้าหมายและทำการทดสอบบน

เว็บเบราว์เซอร์จำนวน 5 โปรแกรมประกอบด้วยโปรแกรม Google Chrome, Edge, Mozilla Firefox, Internet Explorer และ Safari

4) สรุปผลการทดลองโจมตี SSL Stripping Attack

ตารางที่ 3.1 รายชื่อเว็บไซต์ระบบให้บริการธนาคารออนไลน์ในไทย

ลำดับ	เว็บไซต์*	URL
1	A	https://**********.com
2	B	https://**********.com
3	C	https://**********.com
4	D	https://**********.com
5	E	https://**********.com
6	F	https://**********.com
7	G	https://**********.com
8	H	https://**********.com
9	I	https://**********.com
10	J	https://**********.com
11	K	https://**********.com

** เพื่อสงวนชื่อเว็บไซต์ธนาคารออนไลน์ในไทย จึงใช้อักษรย่อแทน*

ตารางที่ 3.2 รายชื่อเว็บไซต์ระบบผู้ให้บริการ E-commerce

ลำดับ	เว็บไซต์*	URL
1	L	https://**********.com
2	M	https://**********.com
3	N	https://**********.com
4	O	https://**********.com
5	P	https://**********.com

** เพื่อสงวนชื่อเว็บไซต์ E-commerce จึงใช้อักษรย่อแทน*

ตารางที่ 3.3 รายชื่อเว็บไซต์ที่ให้บริการระบบทะเบียนมหาวิทยาลัย

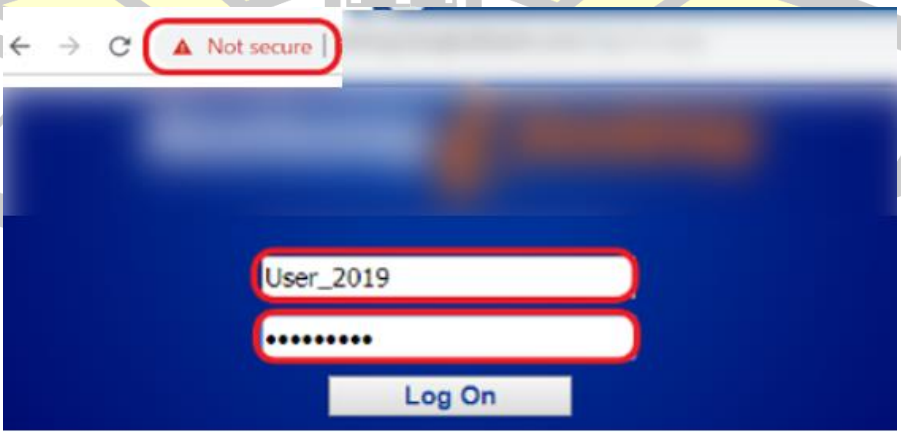
ลำดับ	เว็บไซต์	URL
1	Q	https://reg**********.ac.th
2	R	https://reg**********.ac.th
3	S	https://reg**********.ac.th
4	T	https://reg**********.ac.th
5	U	https://reg**********.ac.th
6	V	https://reg**********.ac.th
7	W	https://reg**********.ac.th
8	X	https://reg**********.ac.th
9	Y	https://reg**********.ac.th
10	Z	https://reg**********.ac.th
11	VI	https://reg**********.ac.th

** เพื่อสงวนชื่อเว็บไซต์ระบบทะเบียนมหาวิทยาลัย จึงใช้อักษรย่อแทน*

3.2 เกณฑ์การประเมินจากเทคนิคการโจมตี SSL Stripping Attack

เกณฑ์การประเมินที่ถูกใช้เพื่อพิจารณาผลลัพธ์ที่เกิดจากการโจมตีด้วยเทคนิค SSL Stripping Attack มีเกณฑ์การประเมินดังนี้

1) SSL Stripping Attack สามารถโจมตีได้ หมายถึง เว็บไซต์ที่นำมาทดสอบถูกทำลายระบบป้องกัน SSL/TLS ออกได้ และเปลี่ยนไปใช้โพรโทคอล HTTP ในการสื่อสารแทน ดังแสดงภาพที่ 3.2

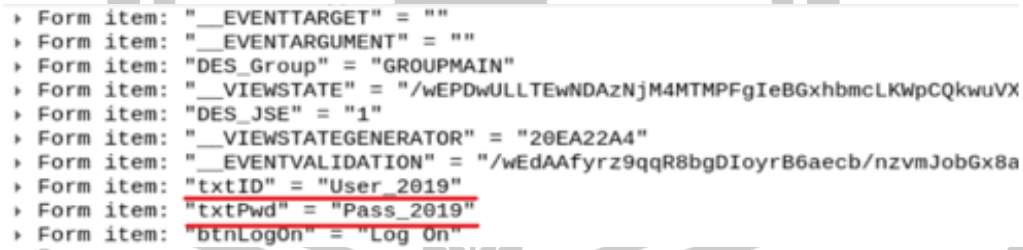


ภาพที่ 3.2 SSL Stripping Attack สามารถโจมตีได้

2) SSL Stripping Attack ไม่สามารถโจมตีได้ หมายถึง เว็บไซต์ที่นำมาทดสอบไม่ถูกทำลายระบบป้องกัน SSL/TLS ออก และยังคงใช้โพรโทคอล HTTPS ในการสื่อสาร ดังแสดงภาพที่ 3.3

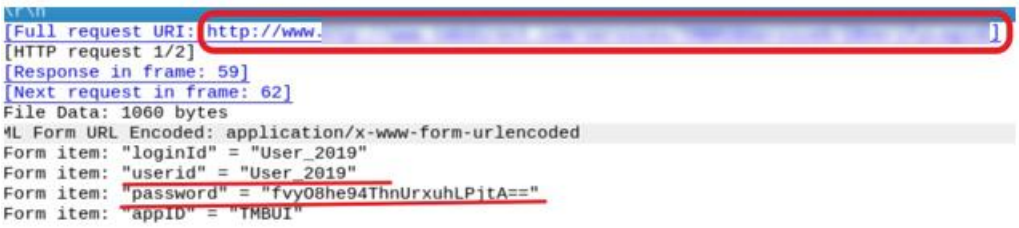
ภาพที่ 3.3 SSL Stripping Attack ไม่สามารถโจมตีได้

3) Data Sniffing Attack สามารถโจมตีได้ หมายถึง หลังจากถูกโจมตีด้วย SSL Stripping Attack แล้ว สามารถดักจับข้อมูลของชื่อผู้ใช้และรหัสผ่านได้ ดังแสดงภาพที่ 3.4



ภาพที่ 3.4 Data Sniffing Attack สามารถโจมตีได้

4) Data Sniffing Attack ไม่สามารถโจมตีได้ หมายถึง หลังทำการโจมตีแล้วไม่สามารถดักจับข้อมูลของชื่อผู้ใช้หรือดักจับรหัสผ่านแบบ Clear Text ได้ ดังแสดงภาพที่ 3.5



ภาพที่ 3.5 Data Sniffing Attack ไม่สามารถโจมตีได้

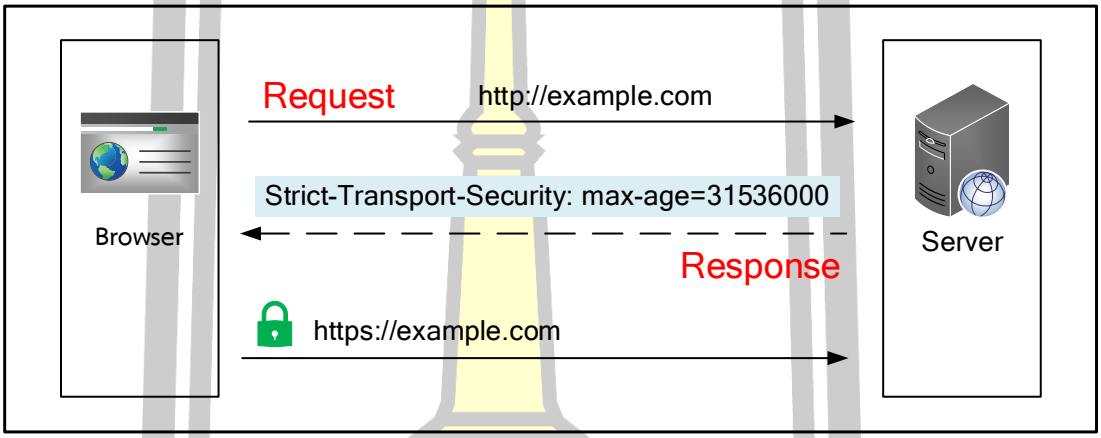
3.3 การวิเคราะห์ปัญหากลไก HSTS และการกลับมาจู่โจมได้ใหม่ของ SSL Stripping Attack

ในวงการวิจัยเป็นที่ทราบกันดีว่า HSTS ได้เข้ามาแก้ปัญหาการถูกโจมตีด้วย SSL Striping Attack และระบบ HSTS ยังเป็นหนึ่งในมาตรฐานของ Internet Engineering Task Force (IETF) ตามเอกสาร RFC 6797 ทำให้ทุกเบราว์เซอร์ทุกตัวรองรับการทำงานระบบ HSTS โดยมีกลไกการทำงาน 2 รูปแบบด้วยกันคือ 1) HSTS Directive 2) HSTS Preload ซึ่งระบบ HSTS จะทำหน้าที่เป็นกลไกส่วนเสริมของโพรโทคอล HTTPS ที่เปิดให้เว็บเซิร์ฟเวอร์ "บังคับ" ให้เว็บเบราว์เซอร์เชื่อมต่อผ่าน HTTPS เท่านั้น แม้ผู้ใช้จะไม่ระบุว่าต้องการใช้ HTTPS ก็ตาม ทำให้ผู้ใช้งานเว็บไซต์ระหว่าง Web Browser กับ Web Server ไม่มีการเชื่อมต่อแบบ HTTP ที่มีปัญหาเรื่องการถูกดักจับข้อมูลแบบ Clear Text สำหรับเว็บไซต์ที่ต้องการความมั่นคงสูงอย่างเช่น ระบบธนาคารออนไลน์ (Internet Banking), ระบบการค้าอิเล็กทรอนิกส์ (E-commerce) จึงมีการปรับใช้ระบบป้องกัน HSTS อย่างแพร่หลายเพื่อรักษาความมั่นคง Web Application ในงานวิจัยนี้จึงเกิดคำถามว่าวิธีการแก้ไขปัญหา HTTPS จากการถูกโจมตีด้วย SSL Striping Attack ที่มีการปรับใช้กลไก HSTS แต่เพียงอย่างเดียวนั้นมีประสิทธิภาพในการป้องกันการโจมตี จริงหรือไม่ ซึ่งจากการทดลอง ISAN Lab ค้นพบว่าระบบเว็บไซต์ธนาคารออนไลน์ที่เคยได้รับการป้องกันจาก HSTS ล้วนแล้วแต่ถูกโจมตีได้อีกครั้งเกือบทั้งสิ้นด้วยเทคนิค SSL Striping Attack และแฮกเกอร์ที่มีความเชี่ยวชาญก็สามารถปรับปรุงโค้ดเพื่อต่อยอดความสามารถในการโจมตีได้ ดังนั้นในงานวิจัยนี้จึงมีแนวคิดทดสอบกลไก HSTS เพื่อวิเคราะห์การทำงานที่ผิดปกติของกลไก HSTS ซึ่งเป็นส่วนเสริมที่สำคัญในการบังคับการสื่อสารผ่านโพรโทคอล HTTPS ระหว่าง Web Browser กับ Web Server โดยมีรายละเอียดการวิเคราะห์ดังต่อไปนี้

3.3.1 แนวคิดการทดลอง HSTS Directive

การทำงานของ HSTS Directive มีลักษณะการทำงานโดยเว็บเซิร์ฟเวอร์จะมีการตอบกลับ (Response) ในส่วนของ HTTP Header ชื่อ Strict-Transport-Security: max-age=3153 6000; include SubDomains เมื่อเว็บเบราว์เซอร์ตรวจสอบพบ HTTP Header

ดังกล่าวเว็บเซิร์ฟเวอร์ก็จะบังคับให้เว็บเบราว์เซอร์เชื่อมต่อผ่านโพรโทคอล HTTPS ทำให้การสื่อสารมีความมั่นคงปลอดภัย จากที่กล่าวมาข้างต้นจะพบว่าชุดคำสั่ง Header HSTS ถูกเพิ่มและตั้งค่าเก็บไว้ที่เว็บเซิร์ฟเวอร์ การทำงานที่เบราว์เซอร์จะถูกประมวลผลทุกครั้งที่ไคลเอนต์เพื่อตรวจสอบการบังคับใช้โพรโทคอล HTTPS ดังนั้นการเรียกใช้งานกลไก HSTS Directive จึงเป็นช่องโหว่ที่ใช้ในการโจมตีได้ด้วยเทคนิค SSL Stripping Attack ซึ่งลักษณะการทำงานของ HSTS Directive ดังแสดงภาพที่ 3.6

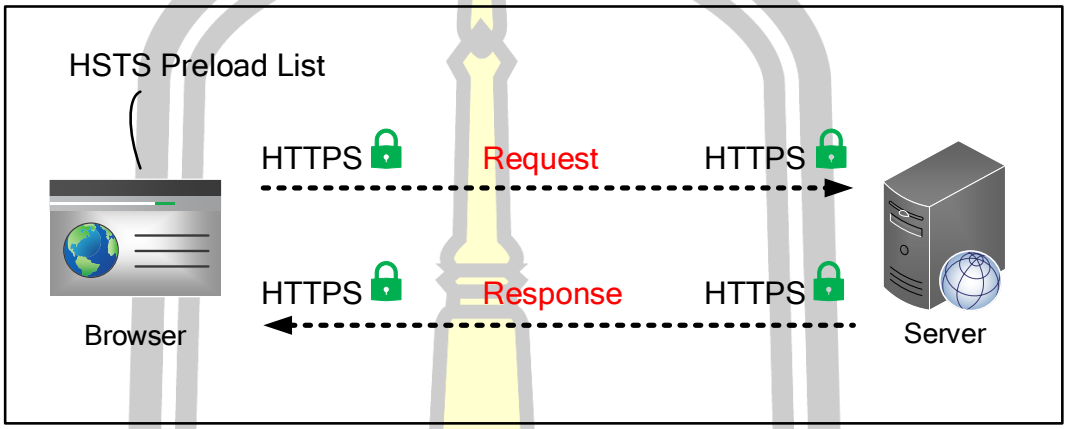


ภาพที่ 3.6 การทำงาน HSTS Directive

3.3.2 แนวคิดการทดลอง HSTS Preload

HSTS Preload เข้ามาแก้ปัญหาการสื่อสารผ่านโพรโทคอล HTTP ที่ไม่ปลอดภัย โดยการทำงานของ HSTS Preload จะบังคับให้เชื่อมต่อผ่านโพรโทคอล HTTPS ตั้งแต่เริ่มต้นการสื่อสารทำให้เว็บเบราว์เซอร์กับเว็บเซิร์ฟเวอร์มีความมั่นคงปลอดภัยจากการถูกโจมตีดักจับข้อมูล ซึ่งการเปิดใช้งาน HSTS Preload นั้น ให้เพิ่มชุดคำสั่ง Strict-Transport-Security: max-age=31536 000;includeSubDomains; preload (อธิบายโค้ดแต่ละส่วน 1.max-age ระยะเวลาของการบังคับใช้โพรโทคอล HSTS ที่กำหนดให้สื่อสารผ่านโพรโทคอล HTTPS นานเท่าใด โดยมีหน่วยเป็นวินาที 2.includeSubDomains กรณีสื่อสารผ่าน subdomains ของเว็บไซต์ก็ต้องสื่อสารผ่าน HSTS เช่นเดียวกัน 3.preload คือ domain ที่ถูกติดตั้งอยู่ในรายการ HSTS preload บน browser กำหนดให้เชื่อมต่อ HTTPS จากเว็บเบราว์เซอร์ไปที่เซิร์ฟเวอร์ตั้งแต่เริ่มต้นการสื่อสาร) ให้นำชุดคำสั่งดังกล่าวลงใน HTTP Header จากนั้นนำ Domain Name ลงทะเบียนเปิดใช้งานที่เว็บไซต์ https:// hstspreload.org ในโครงการของ Chromium ซึ่งข้อมูล Domain Name ดังกล่าวจะถูกอัพโหลดไปยังทุกเบราว์เซอร์ที่รองรับการทำงานของกลไก HSTS จากที่กล่าวมาข้างต้น

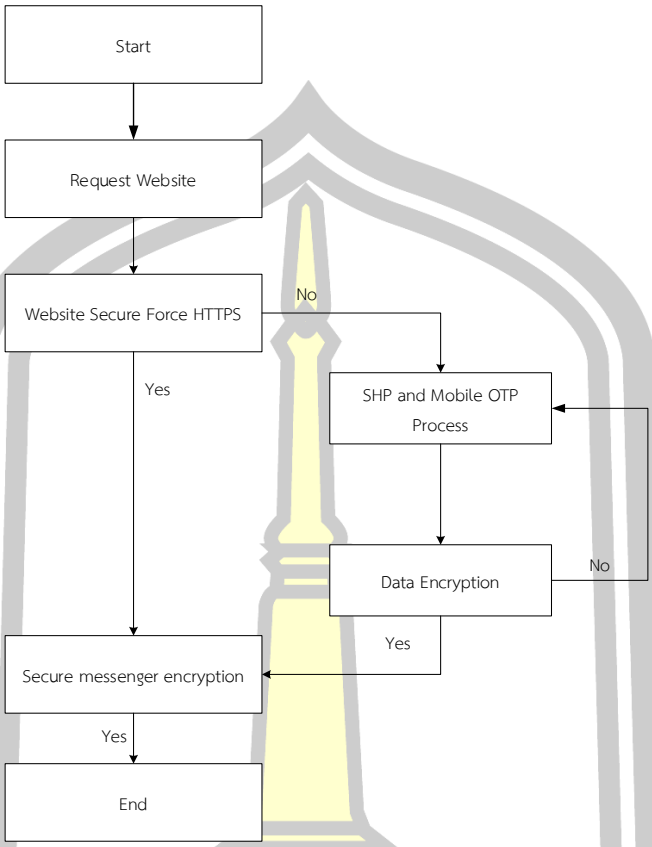
จะพบว่าสุดท้ายแล้วการบังคับเชื่อมต่อโพรโทคอล HTTPS จะอยู่ที่ Domain Name ในฐานข้อมูลของเว็บเบราว์เซอร์ที่เรียกว่า HSTS Preload List ดังนั้นการเรียกใช้งาน Domain Name จึงเป็นช่องโหว่ที่ใช้ในการโจมตีได้ในระดับ Top-Level Domain (TLD) ซึ่งมีลักษณะการทำงานดังแสดงภาพที่ 3.7



ภาพที่ 3.7 การทำงาน HSTS Preload

3.4 เสนอแนวทางป้องกันการถูกดักจับข้อมูลที่ถูกโจมตีด้วยวิธี SSL Stripping Attack

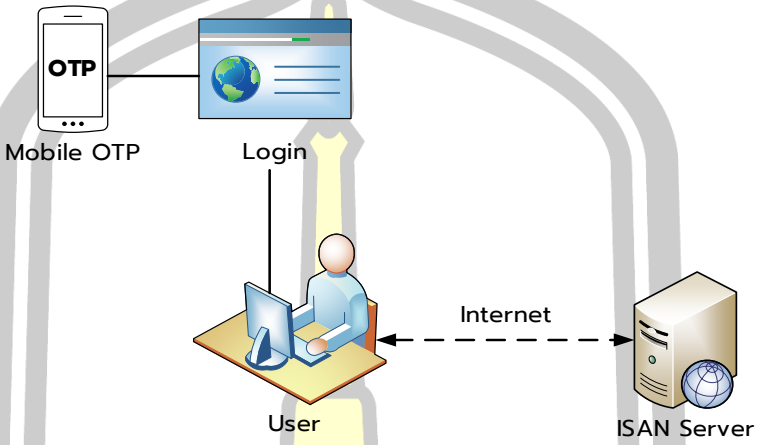
การแก้ปัญหาการถูกดักจับข้อมูลจากเทคนิคการโจมตีด้วย SSL Stripping Attack มีแนวคิดในการออกแบบ isan banking ดังแสดงโครงสร้างการทำงานในภาพที่ 3.8 โดยเริ่มจากการเรียกใช้งานเว็บไซต์ที่ตำแหน่ง (Start Point) ในสภาพแวดล้อมที่ใช้ในการทดสอบ จากนั้นเมื่อเว็บไซต์ถูกโจมตีด้วย SSL Stripping Attack จะมีกระบวนการทำงานคือ เมื่อถูกโจมตีจากเทคนิคดังกล่าวเว็บไซต์มีการบังคับใช้ HTTPS ตลอดกระบวนการใช้งานใช่หรือไม่ หากมีการใช้งาน HTTPS ถือว่าข้อมูลมีการเข้ารหัสการสื่อสารก็จะสิ้นสุดการทำงาน แต่หากว่าไม่ปลอดภัยก็จะส่งข้อมูลเริ่มกระบวนการป้องกันชั้นที่ 2 โดยนำข้อมูลเข้ารหัส Salted Hash Password (SHP) ร่วมกับ Mobile OTP จากนั้นเมื่อตรวจสอบข้อมูลมีการเข้ารหัสก็เป็นจุดสิ้นสุดการทำงาน (End Point) ใช่หรือไม่ ถ้าพบว่าไม่ใช่จุดสิ้นสุดการทำงานระบบจะวนกลับไปเริ่มขบวนการทำงานตั้งแต่การเข้ารหัส SHP ใหม่อีกครั้ง



ภาพที่ 3.8 แสดงโครงสร้างการทำงานของระบบป้องกันชั้นที่ 2 ป้องกันการถูกดักจับข้อมูล

จากเทคนิคการโจมตีด้วยวิธี SSL Stripping Attack มีลักษณะบังคับเปลี่ยนแปลงการทำงานของโพรโทคอลในการสื่อสารจาก HTTPS เป็น HTTP ซึ่งทำให้ไม่ปลอดภัยระหว่างการส่งข้อมูลของระบบเว็บไซต์ที่กำลังสื่อสารกับเว็บเซิร์ฟเวอร์ จึงมีแนวคิดออกแบบและพัฒนาระบบป้องกันชั้นที่ 2 ซ้อนทับ HTTPS อีกชั้น โดยมีเหตุผลที่ว่าเมื่อเหยื่อถูกจู่โจมจากเทคนิค SSL Stripping Attack โพรโทคอลเดิมที่เป็น HTTPS จะถูกบังคับให้สื่อสารผ่าน HTTP ที่ไม่ปลอดภัย จึงเกิดช่องโหว่ที่ผู้โจมตีสามารถดักจับข้อมูลของเหยื่อได้ เนื่องจากข้อมูลอยู่ในรูป Clear Text แต่หากมีการปรับใช้ระบบป้องกันในชั้นที่ 2 คือ Salted Hash Password สร้างเกราะป้องกันซ้อนทับ HTTPS พร้อมกับเสริมความมั่นคงด้วยโมบาย OTP เป็นตัวสร้างรหัสที่ใช้เพียงครั้งเดียวในการเข้าสู่ระบบ ถึงแม้ผู้จู่โจมสามารถทะลุผ่านชั้นมาตรฐานไปได้ แต่ก็ยังมีด่านป้องกันชั้นที่ 2 ค่อยป้องกันข้อมูลอีกชั้น ทั้งนี้ ได้เลือกใช้ TOTP (Time-Based One Time Password) ที่เป็นมาตรฐาน RFC 6238 มาเป็นฐานในการพัฒนาระบบ OTP เพราะปัจจุบันเครื่องสมาร์ทโฟนส่วนใหญ่ไม่มีปัญหาเวลาที่ไม่ตรงกับเวลามาตรฐานสากล ซึ่งการทำงานของสมาร์ทโฟนจะถูกตั้งค่าตามเวลาของ Network ISP ในการ Sync เวลาสากล จึงทำให้ไม่มีข้อผิดพลาดเรื่องเวลาในสมาร์ทโฟนที่ใช้ในปัจจุบัน ซึ่งมีภาพรวมและ

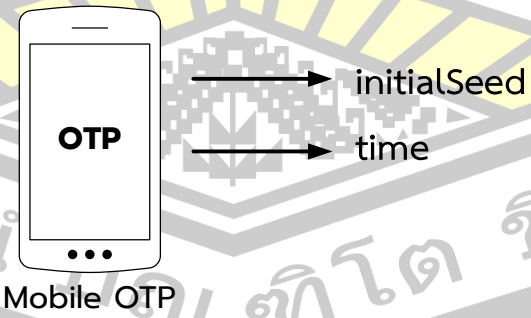
ส่วนประกอบมาตรการสร้างมั่นคงให้กับระบบเว็บไซต์ดังแสดงError! Reference source not found.



ภาพที่ 3.9 ภาพรวมและส่วนประกอบมาตรการสร้างมั่นคงให้กับระบบเว็บไซต์

จากError! Reference source not found. การสร้างมาตรการความมั่นคงชั้นที่ 2 ให้กับระบบเว็บไซต์ จะอาศัยระบบ Mobile OTP บนสมาร์ทโฟนเป็นตัวช่วยเพื่อสร้าง OTP ที่ใช้สำหรับการเข้าสู่ระบบของเว็บไซต์ โดย OTP ที่ถูกสร้างขึ้นจะเปลี่ยนไปตามช่วงเวลา NTP (Network Time Protocol) ที่เปลี่ยนไปและทางฝั่งของ ISAN Server ก็จะมีการสร้างค่า OTP ที่ตรงกันขึ้นเพื่อยืนยันความถูกต้องในการเข้าสู่ระบบ

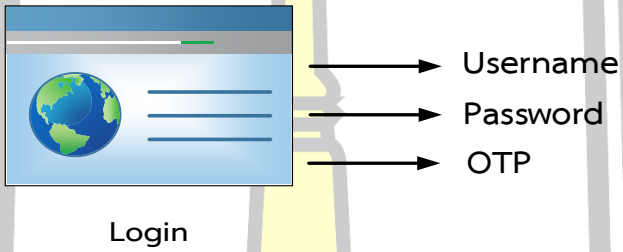
3.4.1 กระบวนการสร้าง OTP ของ Mobile OTP



ภาพที่ 3.10 กระบวนการสร้าง OTP ของ Mobile OTP

จากภาพที่ 3.10 การสร้าง OTP ในฝั่งของ Mobile OTP จะมีกระบวนรับค่าจาก initialSeed และ Time โดยที่ initialSeed จะเป็นค่า Secret Key ที่ไม่ซ้ำกัน (อาจเป็นข้อมูลความลับจากกระบวนการยืนยันตัวตนก่อนเข้าใช้งานระบบเว็บไซต์) จากนั้นนำข้อมูลไป Hash ด้วย SHA 256 และในส่วน Time เป็นค่าที่ได้จาก Network Time Protocol ที่ดึงข้อมูลมาจากสมาร์ทโฟน ซึ่งจะนำค่าที่ได้ทั้ง 2 มาเข้ากระบวนการสร้าง OTP เพื่อนำมาใช้ร่วมกับการเข้าสู่ระบบเว็บไซต์ ซึ่ง OTP จะแสดงค่าใหม่ในทุก ๆ 30 หรือ 60 วินาที

3.4.1 กระบวนการเข้าสู่ระบบ



ภาพที่ 3.11 กระบวนการเข้าสู่ระบบ

จาก**Error! Reference source not found.** การเข้าสู่ระบบเว็บไซต์จะใช้ทั้ง 3 ค่า คือ Username, Password และ OTP โดยขั้นตอนแรกให้ยืนยัน Username เมื่อกรอกและยืนยัน ระบบจะนำข้อมูลต้นฉบับมาทำการ Hash ด้วย SHA 256 ซ้ำ 2 รอบ กระบวนการนี้จะเกิดขึ้นที่ฝั่งของ Client ทำการส่งค่า Hash ไปตรวจสอบที่ฝั่ง Server ในรูปแบบ Client Side Script บนภาษา JavaScript เพื่อป้องกันการดักจับข้อมูลระหว่างการสื่อสาร หน้าถัดไปให้กรอก Password และ OTP ในส่วน Password จะนำไป Hash ด้วย SHA 256 ซ้ำ 2 รอบและนำมาต่อด้วยค่า Salt (ใช้เป็นค่า OTP) แล้ว Hash ซ้ำอีก 1 รอบ เสร็จกระบวนการนี้จะได้ค่า OTP พร้อมนำไปใช้งานยืนยันความถูกต้องเพื่อเข้าสู่ระบบเว็บไซต์

3.4.1 เครื่องมือที่ใช้ในการพัฒนา

เครื่องมือที่ถูกนำมาใช้ในการพัฒนาระบบแก้ปัญหาป้องกันการถูกดักจับข้อมูลที่ถูกโจมตีด้วยวิธี SSL Stripping Attack

1) PHP เป็นภาษาประเภท Script Language ที่ทำงานแบบ Server Side Script กระบวนการทำงานจะทำงานแบบโปรแกรมแปลคำสั่ง interpreter คือแปลภาษาทุกครั้งที่มีคนเรียกสคริปต์ ข้อดีคือ ไม่ต้องนำไปประมวลผลใหม่ (Compiler) เมื่อจะนำโปรแกรมไปใช้งาน ภาษา PHP จัดอยู่ในประเภท การเขียนโปรแกรมบนเว็บ (Web-based Programming) เพราะจะเก็บโค้ดคำสั่งหรือสคริปต์ทั้งหมดที่เขียนขึ้นมาไว้บนเครื่องเซิร์ฟเวอร์ที่เดียว (Web Server) และให้ผู้ใช้งาน

(Client) เรียกใช้งานโปรแกรมผ่านเว็บเบราเซอร์ต่างๆ เช่น Internet Explorer, Mozilla Firefox, Google Chrome, Opera, Safari ฯลฯ เพื่อนำข้อมูลมาแสดงผลที่หน้าจอของผู้ใช้แต่ละคนนั่นเอง

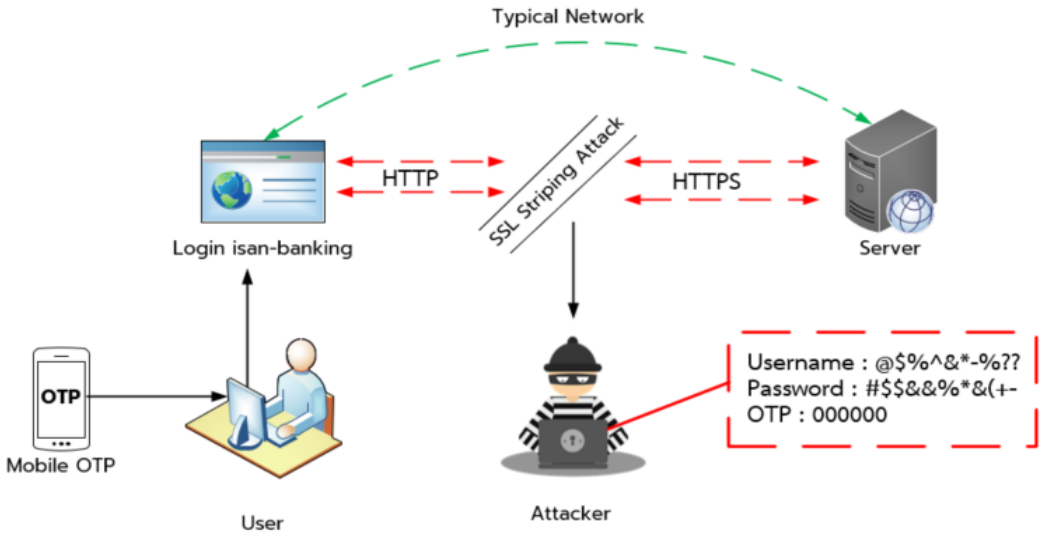
2) jQuery คือ JavaScript Library ยอดนิยมที่โปรแกรมเมอร์นำมาใช้พัฒนาเว็บไซต์ สามารถรองรับการทำงานทุกเบราเซอร์ โดย Library นี้มีวัตถุประสงค์ที่จะเอามาแก้ปัญหาการใช้งาน JavaScript ที่มีความยุ่งยากในการใช้งานให้สามารถใช้งานได้ง่ายขึ้น เช่น การรวบคำสั่งหลาย ๆ บรรทัดของ JavaScript ที่ซับซ้อนมาเป็นการใช้งานผ่าน jQuery เพียงแค่ 1 บรรทัด หรือการเรียกใช้งานคำสั่งประเภท Ajax, DOM ให้ใช้งานได้ง่ายขึ้นเป็นต้น บริษัทใหญ่ ๆ หลายบริษัททั่วโลกนำ JQuery ไปใช้งานในเว็บของเขา เช่น Google, Microsoft, IBM ทำให้ JQuery ถูกใช้งานได้อย่างกว้างขวาง

3) CryptoJS เป็นไลบรารีการเข้ารหัสของ JavaScript ที่มีไว้เพื่อเรียกใช้อัลกอริทึมต่าง ๆ ประกอบด้วยไซเฟอร์ต่อไปนี้: AES-128, AES-192, AES-256, DES, Triple DES, Rabbit, RC4, RC4Drop และแฮช: MD5, RIPEMD-160, SHA-1, SHA-256, SHA-512, SHA -3 กับ 224, 256, 384 หรือ 512 บิต

4) Bootstrap คือ Frontend Framework ที่รวม HTML, CSS และ JS เข้าด้วยกันสำหรับพัฒนา Web ที่รองรับทุก Smart Device หรือ เรียกว่า Responsive Web หรือ Mobile Firs

3.5 วิธีการทดลองความมั่นคงของ isan banking

ในการทดสอบความมั่นคงของระบบ isan-Banking ทำการทดลองบนเครือข่าย (Test-bed) ซึ่งใกล้เคียงกับระบบ i-Banking ของธนาคารโดยทั่วไปดังแสดงภาพที่ 3.12



ภาพที่ 3.12 จำลองเครือข่ายที่ใช้ทดสอบ isan-banking

จากภาพที่ 3.12 ผู้ใช้งาน (User) เข้าใช้งานหน้าเข้าสู่ระบบของ isan-banking ผ่าน Web Browser โดยทางผู้ใช้งานจะกรอกข้อมูลเพื่อยืนยันความถูกต้องซึ่งได้แก่ Username, Password และ OTP เพื่อส่งไปยังเครื่องเซิร์ฟเวอร์ (Server) ในระหว่างที่มีการสื่อการข้อมูลผู้โจมตี (Attacker) จะใช้เทคนิค SSL Stripping Attack และค่อยดักจับ (Sniffing) ข้อมูลระหว่างเครื่องผู้ใช้งานและเครื่องเซิร์ฟเวอร์ ด้วยการใช้เทคนิคการโจมตีแบบแทรกกลางการสื่อสาร Man-in-the-Middle Attack

3.6 เครื่องมือที่ใช้ในการทดลอง

การทดลองในงานวิจัยนี้มีการเลือกใช้เครื่องมือต่าง ๆ ซึ่งประกอบไปด้วยโปรแกรมที่ใช้เพื่อวัตถุประสงค์ในการแทรกกลางการสื่อสาร โปรแกรมที่ใช้ในการดักจับข้อมูล โปรแกรมที่ใช้ในการโจมตี SSL ด้วยวิธี SSL Stripping Attack และโปรแกรมเว็บเบราว์เซอร์ เครื่องมือที่ใช้ในการทดสอบได้แก่

1) เครื่องมือที่ใช้ในการโจมตี

เครื่องมือที่ใช้ในการโจมตีเหล่านี้จะถูกนำมาใช้งานร่วมกันเพื่อทำการแทรกกลางการสื่อสารระบบเว็บไซต์ที่ทำงานบน HTTPS จากนั้นก็จะทำการโจมตีด้วยวิธี SSL Stripping Attack แล้วดักจับข้อมูลที่สำคัญของผู้ใช้ออกมา เครื่องเหยื่อใช้ CPU Intel i5 RAM 8GB โดยมี MS Windows 10 เป็นระบบปฏิบัติการ เครื่องของผู้โจมตีใช้ CPU Intel i5 RAM 8GB โดยใช้ Kali Linux 2020.1 เป็นระบบปฏิบัติการประกอบไปด้วยการใช้งานคำสั่ง SSL Strip, Ettercap, Bettercap และ wireshark ซึ่งถูกติดตั้งอยู่บนระบบปฏิบัติการ Kali Linux

2) เว็บเบราว์เซอร์

ในการทดสอบโจมตีเว็บไซต์ที่ทำงานบน HTTPS นั้น เพื่อให้เห็นถึงความแตกต่างในการแสดงผลบนแต่ละโปรแกรมเว็บเบราว์เซอร์ รวมถึงประสิทธิภาพในการป้องกันการโจมตี SSL/TLS ในงานวิจัยนี้จึงได้เลือกใช้เว็บเบราว์เซอร์จำนวน 5 โปรแกรม โดยการเลือกโปรแกรมที่นำมาทำการทดสอบนี้ใช้การอ้างอิงข้อมูลจากส่วนแบ่งทางการตลาดของเว็บเบราว์เซอร์ซึ่งเป็นที่นิยมจำนวน 5 อันดับ มีรายละเอียดดังแสดงใน**Error! Reference source not found.** และ**Error! Reference source not found.** โดยประกอบไปด้วย

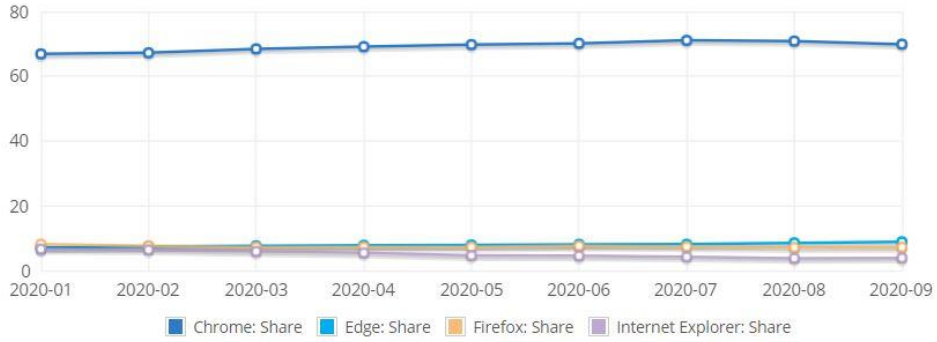
(1) Google Chrome [40] ในงานวิจัยนี้เลือกใช้ Google Chrome เวอร์ชัน 79 ซึ่งมีการใช้งานอย่างแพร่หลายเป็นอันดับหนึ่ง สามารถติดตั้งได้ทั้งบนระบบปฏิบัติการ Windows, Mac OSX, และ Linux

(2) Microsoft Edge [41] เป็นเว็บเบราว์เซอร์ที่รองรับการทำงานบนระบบปฏิบัติการ Windows, Mac OSX, และ Linux โดยในงานวิจัยนี้เลือกใช้ Microsoft Edge เวอร์ชัน 85 ในการทดสอบ

(3) Mozilla Firefox [42] ในงานวิจัยนี้ใช้ Mozilla Firefox เวอร์ชัน 72 ซึ่งการใช้งานสามารถนำมาติดตั้งได้ทั้งบนระบบปฏิบัติการ Windows, Mac OSX, และ Linux

(4) Internet Explorer [43] รองรับการทำงานบนระบบปฏิบัติการ Windows ซึ่งในงานวิจัยนี้ได้เลือกใช้ Internet Explorer เวอร์ชัน 11 มาใช้ในการทดสอบ

(5) Safari รองรับการทำงานบนระบบปฏิบัติการ Windows, Mac OSX และ IOS โดยในงานวิจัยนี้เลือกใช้ Apple Safari [44] เวอร์ชัน 13 มาทำการทดสอบ



Show 10 entries Search:

Browser	Share
Chrome	69.43%
Edge	7.92%
Firefox	7.38%
Internet Explorer	4.98%
Safari	3.73%

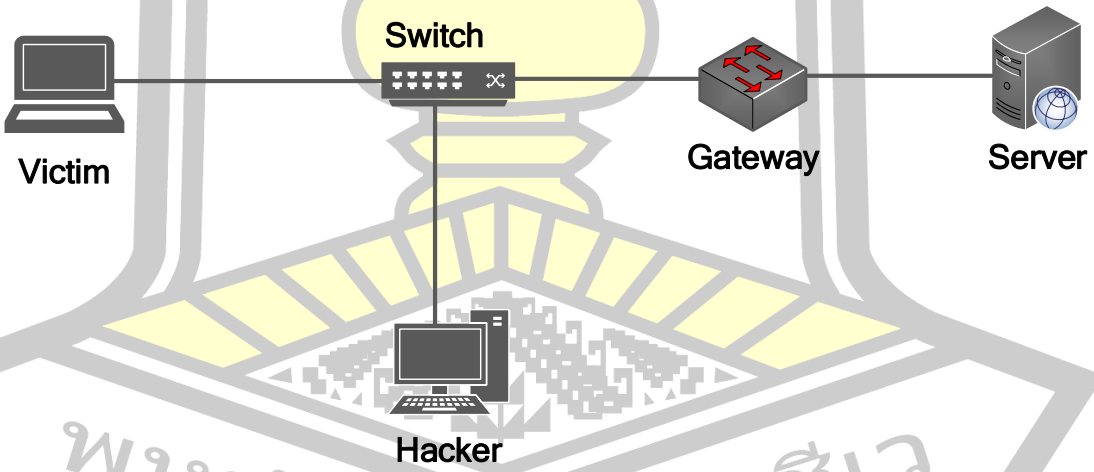
ภาพที่ 3.13 กราฟแสดงส่วนแบ่งทางการตลาดของเว็บเบราว์เซอร์ สำรวจเมื่อ ค.ศ. 2020
ที่มา: [45]

ตารางที่ 3.4 แสดงส่วนแบ่งทางการตลาดของเว็บเบราว์เซอร์ใน ปี ค.ศ. 2020

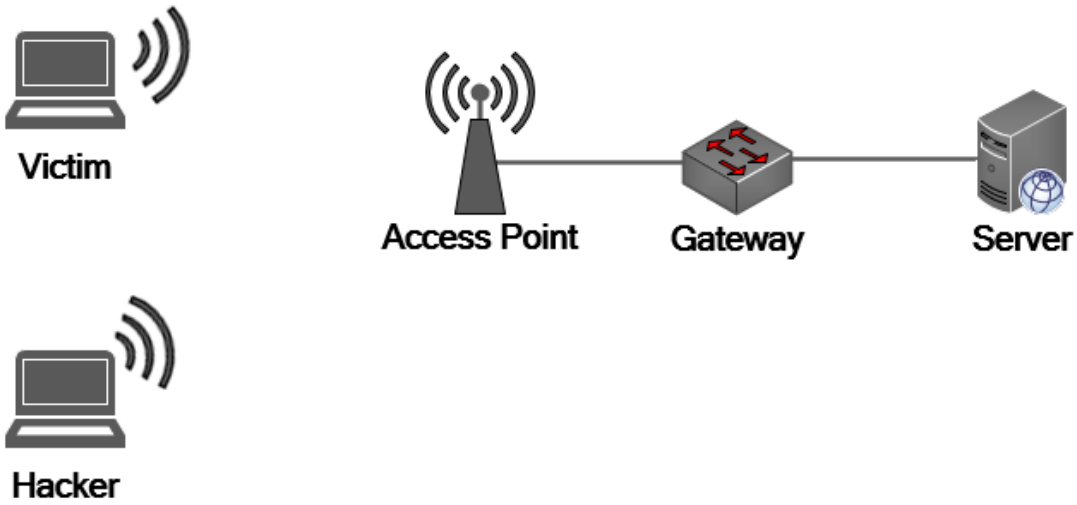
Month	Chrome	Edge	Firefox	Internet Explorer	Safari
September 2020	69.94%	8.84%	7.19%	3.88%	3.57%
August 2020	70.89%	8.52%	7.11%	3.79%	3.53%
July 2020	71.11%	8.09%	7.36%	4.23%	3.36%
June 2020	70.19%	8.07%	7.58%	4.53%	3.56%
May 2020	69.81%	7.86%	7.23%	4.61%	3.90%
April 2020	69.18%	7.76%	7.25%	5.45%	3.94%

3.7 สภาพแวดล้อมที่ใช้ในการทดลอง

สภาพแวดล้อมที่ถูกกำหนดเป็น Test-bed เพื่อใช้ในการทดสอบโจมตี แบ่งออกเป็น 2 รูปแบบ ประกอบด้วย Wire Network ใช้สำหรับทดสอบการโจมตีบนแพลตฟอร์มประเภท PC Desktop ดัง**Error! Reference source not found.** และ Wireless Network ใช้สำหรับทดสอบการโจมตีแบบเชื่อมต่อไร้สาย ดัง**Error! Reference source not found.**



ภาพที่ 3.14 ระบบ Wire Network ที่ใช้ทดสอบการโจมตี



ภาพที่ 3.15 ระบบ Wireless Network ที่ใช้ทดสอบการโจมตี

3.8 ข้อจรรยาบรรณในการวิจัย

การทดลองโจมตีเว็บไซต์ด้วยเทคนิค SSL Stripping Attack นี้ ไม่ได้มีวัตถุประสงค์ในการเจาะเข้าไปในระบบผู้ให้บริการของเว็บไซต์ที่ใช้ในการทดสอบหรือก่อให้เกิดผลกระทบต่อการทำงานของระบบผู้ให้บริการแต่อย่างใด หากแต่เป็นการทดสอบโจมตีเพื่อทำการดักจับข้อมูลในระหว่างการสื่อสารเท่านั้น การทดลองจึงไม่ได้ใช้ชื่อผู้ใช้หรือรหัสผ่านจริง

ทั้งนี้ การทดลองกระทำภายใต้การกำกับของนิติกรจากส่วนคดีเทคโนโลยีสารสนเทศ กรมสอบสวนคดีพิเศษ เพื่อไม่ให้ผิดจรรยาบรรณและกฎหมาย

บทที่ 4

ผลการวิจัย

ในบทนี้จะกล่าวถึงผลการทดลอง SSL Stripping Attack จากกลุ่มเว็บไซต์ตัวอย่างที่นำมาทดสอบ และผลการวิเคราะห์ปัญหากลไก HSTS อย่างละเอียดเพื่อให้เข้าใจสาเหตุของการกลับมาโจมตีได้ใหม่ของการเปลือยเอสเอสแอลต่อความมั่นคงของเว็บไซต์ โดยมีรายละเอียดดังนี้

4.1 ผลการวิเคราะห์ตรวจสอบ HTTP Response Header

การวิเคราะห์ตรวจสอบการตั้งค่ากลไก HSTS ในกลุ่มเว็บไซต์ที่ใช้ในการทดลองจำนวน 27 เว็บไซต์ ประกอบด้วยเว็บไซต์ที่ให้บริการธนาคารออนไลน์ในไทย จำนวน 11 เว็บ, เว็บไซต์ที่ให้บริการ E-commerce จำนวน 5 เว็บ, ระบบทะเบียนมหาวิทยาลัย จำนวน 11 เว็บไซต์ ซึ่งสามารถตรวจสอบการตั้งค่า Header HSTS โดยการ Request เว็บไซต์ผ่าน Web Browser จากนั่น Inspect Network แล้วเลือกดูข้อมูลในหัวข้อ Response Headers ก็จะพบ Header HSTS ดังแสดง**Error! Reference source not found.**

ภาพที่ 4.1 ผลการตรวจสอบการตั้งค่ากลไก HSTS

4.1.1 ผลการตรวจสอบการตั้งค่ากลไก HSTS ของเว็บไซต์ธนาคารออนไลน์ในประเทศไทย

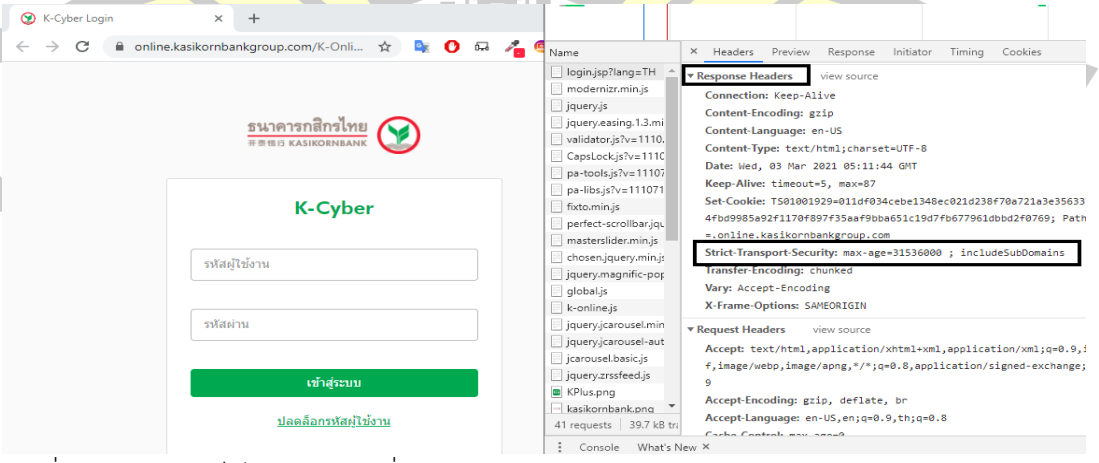
ผลการตรวจสอบ HTTP Header HSTS ของเว็บไซต์ที่ให้บริการธนาคารออนไลน์ในประเทศไทย จำนวน 11 เว็บ พบว่ามีการปรับแต่งค่าตามค่ามาตรฐาน 4 ธนาคาร (ค่าที่แนะนำโดย Google คือ Strict-Transport-Security: max-age=31536000; includeSubDomains) และมี 3 ธนาคาร ที่มีการตั้งค่า Max-Age Configuration ไม่เหมาะสม (ค่าที่แนะนำโดย Google คือ

31536000 ขึ้นไป) และไม่พบการ Configure HSTS จำนวน 4 ธนาคาร ซึ่งธนาคารออนไลน์ดังกล่าวน่าจะโดนโจมตีด้วย SSL Stripping Attack ได้ และมีเพียง 1 ธนาคาร ที่มีการตั้งค่ากลไก HSTS แบบ Preload ซึ่งน่าจะเป็นค่าที่เหมาะสมที่สุดในการป้องกัน SSL Stripping Attack ดังแสดง**Error! Reference source not found.**

ตารางที่ 4.1 ผลการตรวจสอบการตั้งค่ากลไก HSTS ของเว็บไซต์ธนาคารออนไลน์ในประเทศไทย

ระบบธนาคารออนไลน์ในไทย*	HSTS			
	Header	IncludeSubdomains	Max-Age	Preload
A	Yes	Yes	31536000	No
B	Yes	Yes	31536000	Yes
C	Yes	Yes	31536000	No
D	Yes	Yes	31536000	No
E	Yes	Yes	12051306	No
F	Yes	Yes	15552000	No
G	Yes	No	No	No
H	ไม่พบการ Configure HSTS			
I	ไม่พบการ Configure HSTS			
J	ไม่พบการ Configure HSTS			
K	ไม่พบการ Configure HSTS			

** เพื่อสงวนชื่อเว็บไซต์ธนาคารออนไลน์ในประเทศไทย จึงใช้อักษรย่อแทน*



ภาพที่ 4.2 ตัวอย่างเว็บไซต์ธนาคารที่ Configuration Max-Age HSTS เหมาะสม

4.1.2 ผลการตรวจสอบการตั้งค่ากลไก HSTS ของเว็บไซต์ E-commerce

ผลการตรวจสอบ HTTP Header HSTS ของเว็บไซต์ที่ให้บริการระบบ E-commerce จำนวน 5 เว็บ พบว่ามีการ Configuration ตามค่ามาตรฐาน 4 เว็บ และมีเพียง 1 เว็บ ที่มีการตั้งค่ากลไก HSTS แบบ Preload ซึ่งน่าจะเป็นค่าที่เหมาะสมที่สุดดังแสดง**Error! Reference source not found.**

ตารางที่ 4.2 ผลการตรวจสอบการตั้งค่ากลไก HSTS ของเว็บไซต์ E-commerce

เว็บไซต์ E-commerce*	HSTS			
	Header	IncludeSubdomains	Max-Age	Preload
L	Yes	Yes	47474747	Yes
M	Yes	Yes	31536000	No
N	Yes	Yes	31536000	No
O	Yes	Yes	31536000	No
P	Yes	Yes	31536000	No

** เพื่อสงวนชื่อเว็บไซต์ E-commerce จึงใช้อักษรย่อแทน*

4.1.3 ผลการตรวจสอบการตั้งค่ากลไก HSTS ของเว็บไซต์ระบบทะเบียนมหาวิทยาลัย

ผลการตรวจสอบ HTTP Header HSTS ของเว็บไซต์ที่ให้บริการระบบทะเบียนมหาวิทยาลัย จำนวน 11 เว็บ พบว่ามีการ Configuration ตามค่ามาตรฐาน 3 เว็บไซต์ และไม่พบการ Configure HSTS จำนวน 8 เว็บ ซึ่งน่าจะโดนโจมตีด้วย SSL Stripping Attack ได้ดังแสดง **Error! Reference source not found.**

ตารางที่ 4.3 ผลการตรวจสอบการตั้งค่ากลไก HSTS ของเว็บไซต์ระบบทะเบียนมหาวิทยาลัย

เว็บไซต์ระบบทะเบียนมหาวิทยาลัย*	HSTS			
	Header	IncludeSubdomains	Max-Age	Preload
Q	Yes	Yes	31536000	No
R	Yes	Yes	31536000	No
S	Yes	Yes	31536000	No
T	ไม่พบการ Configure HSTS			
U	ไม่พบการ Configure HSTS			

พหุน ปณ ฑิโต ชีเว

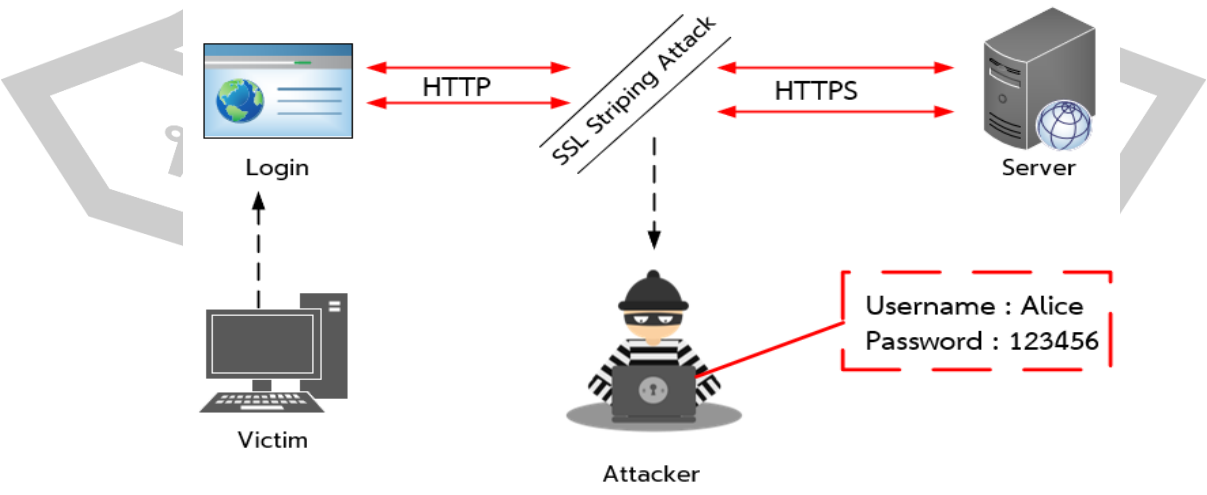
(ต่อ) ตารางที่ 4.3 ผลการตรวจสอบการตั้งค่ากลไก HSTS ของเว็บไซต์ระบบทะเบียนมหาวิทยาลัย

เว็บไซต์ระบบทะเบียนมหาวิทยาลัย*	HSTS			
	Header	IncludeSubdomains	Max-Age	Preload
V	ไม่พบการ Configure HSTS			
W	ไม่พบการ Configure HSTS			
X	ไม่พบการ Configure HSTS			
Y	ไม่พบการ Configure HSTS			
Z	ไม่พบการ Configure HSTS			
VI	ไม่พบการ Configure HSTS			

** เพื่อสงวนชื่อเว็บไซต์ระบบทะเบียนมหาวิทยาลัย จึงใช้อักษรย่อแทน*

4.2 ผลการทดลองโจมตี SSL Stripping Attack

ผลการทดลองโจมตีเว็บไซต์ด้วย SSL Stripping Attack จากกลุ่มตัวอย่างจำนวน 27 เว็บไซต์ เพื่อให้ทราบผลการโจมตีว่าเป็นไปตามความคาดหวังหลังอ่านค่า HSTS Response Header หรือไม่ จึงทำการทดลอง Strip และ Sniff ในหน้าเข้าสู่ระบบ (Login) บนเว็บเบราว์เซอร์ Google Chrome, Safari, Internet Explorer, Mozilla Firefox และ Opera ในการทดสอบพบว่า หากการโจมตีด้วยเทคนิค SSL Stripping Attack สามารถทำการทำลายระบบป้องกัน SSL/TLS ได้สำเร็จ โพรโทคอลเดิมที่เป็น HTTPS จะถูกบังคับให้ใช้เป็น HTTP แทน ทำให้สามารถดักจับข้อมูล (Data Sniff) ของชื่อผู้ใช้และรหัสผ่านออกมาได้ ซึ่งมีรูปแบบการโจมตีดัง**Error! Reference source not found.**



ภาพที่ 4.3 รูปแบบการโจมตีด้วย SSL Stripping Attack

4.2.1 ผลการโจมตีเว็บไซต์ธนาคารออนไลน์ในประเทศไทยด้วยวิธี SSL Stripping Attack

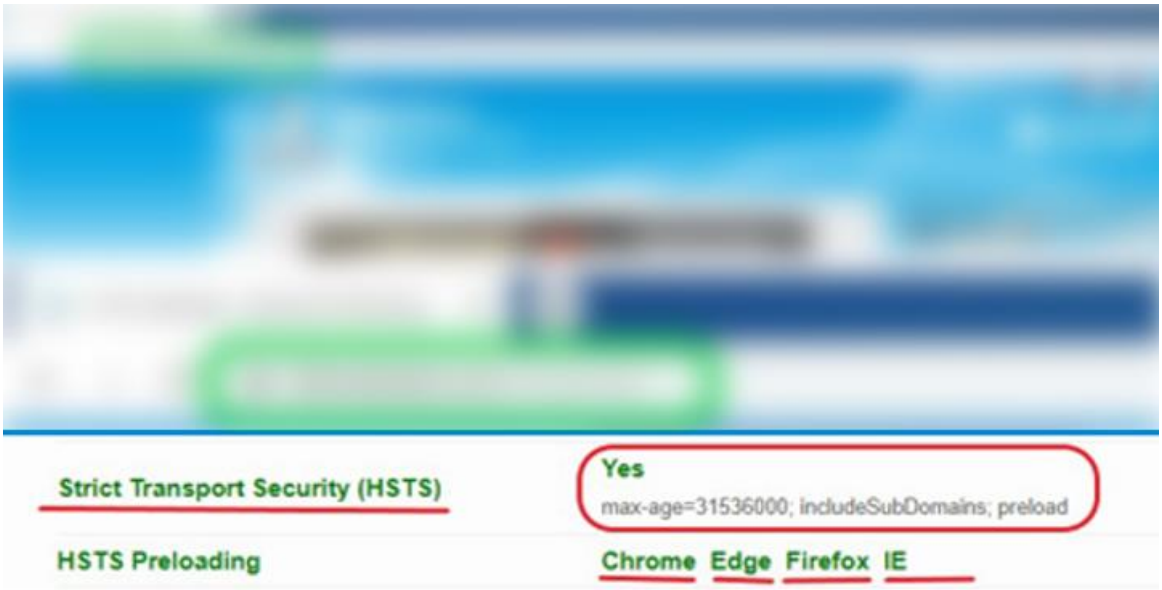
จากการทดลองโจมตีเว็บไซต์ธนาคารออนไลน์ในประเทศไทย จำนวน 11 เว็บ ด้วยวิธี SSL Stripping Attack ได้ผลลัพธ์ในการทดลองและรายละเอียดดังนี้ จาก**Error! Reference source not found.** จะเห็นได้ว่า มีเพียงธนาคาร B ที่รอดจากการถูกโจมตีด้วย SSL Strip และ Sniff ดักจับข้อมูล จากกรณีศึกษาพบเป็นธนาคาร 1 เดียวที่มีการตั้งค่า HSTS แบบ Preload ดังแสดง**Error! Reference source not found.** และในผลการทดลองมี 10 ใน 11 ธนาคาร ที่สามารถทำลายระบบป้องกันปลด HTTPS ไปเป็น HTTP ได้ แม้มีการตั้งค่า HSTS Config ด้วยค่า Max-age ที่เหมาะสมเมื่อดูจาก HTTP Response Header โดยใน 10 ธนาคาร ที่ถูก Strip มี 9 ธนาคารที่ถูก Sniff ทำให้สามารถดักจับข้อมูลรหัสผ่านของเหยื่อได้ดังแสดง**Error! Reference source not found.** ทั้งนี้ มี 1 ธนาคาร E ที่ถึงแม้สามารถโจมตี SSL Strip ได้ แต่กลับไม่สามารถนำข้อมูลไปใช้ประโยชน์ได้เนื่องจากข้อมูลมีการเข้ารหัส Salted-hash password (SHP) ดังแสดง **Error! Reference source not found.**

ตารางที่ 4.4 ผลการโจมตีเว็บไซต์ธนาคารออนไลน์ในประเทศไทยด้วยวิธี SSL Stripping Attack

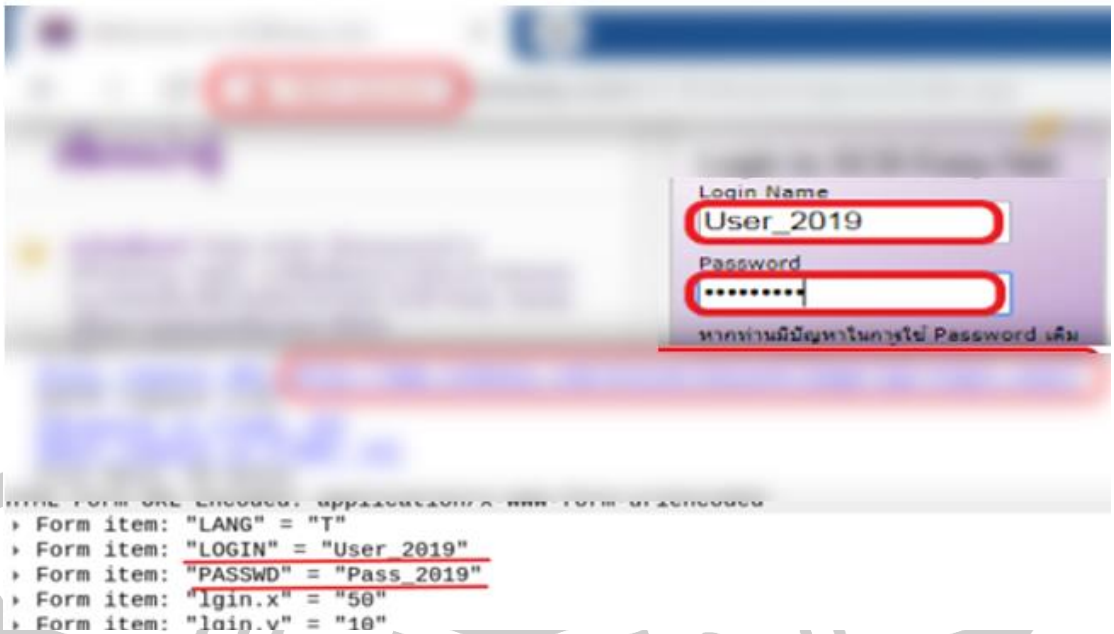
เว็บไซต์*	HSTS		SSL Strip Attack	
	Max-Age	Preload	SSL Strip	Data Sniff
A	31536000	No	✓	✓
B	31536000	Yes	×	×
C	31536000	No	✓	✓
D	31536000	No	✓	✓
E	12051306	No	✓	×
F	15552000	No	✓	✓
G	No	No	✓	✓
H	ไม่พบการ Configure HSTS		✓	✓
I	ไม่พบการ Configure HSTS		✓	✓
J	ไม่พบการ Configure HSTS		✓	✓
K	ไม่พบการ Configure HSTS		✓	✓

** เพื่อสงวนชื่อเว็บไซต์ธนาคารออนไลน์ในไทย จึงใช้อักษรย่อแทน*

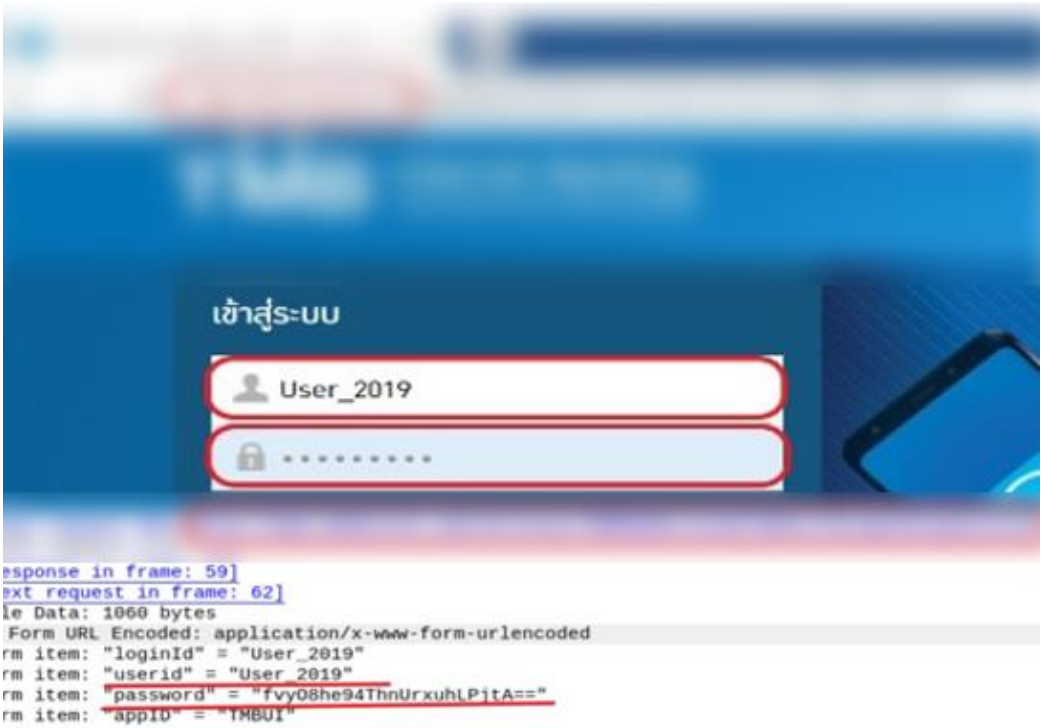
✓ : สามารถโจมตีได้ × : ไม่สามารถโจมตีได้



ภาพที่ 4.4 ผลการทดลองเว็บธนาคารที่มีการตั้งค่า HSTS แบบ Preload



ภาพที่ 4.5 ผลการทดลองเว็บธนาคารที่ถูก SSL Strip และ Sniff



ภาพที่ 4.6 ผลการทดลองเว็บธนาคารที่มีการปรับใช้ Salted-hash password

ตารางที่ 4.5 สรุปผลการโจมตีเว็บไซต์ธนาคารออนไลน์ในประเทศไทย

Web Browser	SSL Strip Attack		
	โจมตีได้	โจมตีไม่ได้	ดักจับข้อมูลไม่ได้
Google Chrome	10	1	2
Safari	10	1	2
Internet Explorer	10	1	2
Mozilla Firefox	10	1	2
Opera	10	1	2

4.2.2 ผลการโจมตีเว็บไซต์ E-commerce ด้วยวิธี SSL Stripping Attack

จากการทดลองโจมตีเว็บไซต์ E-commerce จำนวน 5 เว็บ ด้วยวิธี SSL Stripping Attack ได้ผลลัพธ์ในการทดลองและรายละเอียดดังนี้ จาก**Error! Reference source not found.** จะเห็นได้ว่า มีเพียง L ที่รอดจากการถูกโจมตี SSL Strip และ Sniff ดักจับข้อมูล เนื่องจากมีการตั้งค่า HSTS แบบ Preload ดังแสดง**Error! Reference source not found.** และใน 4 E-

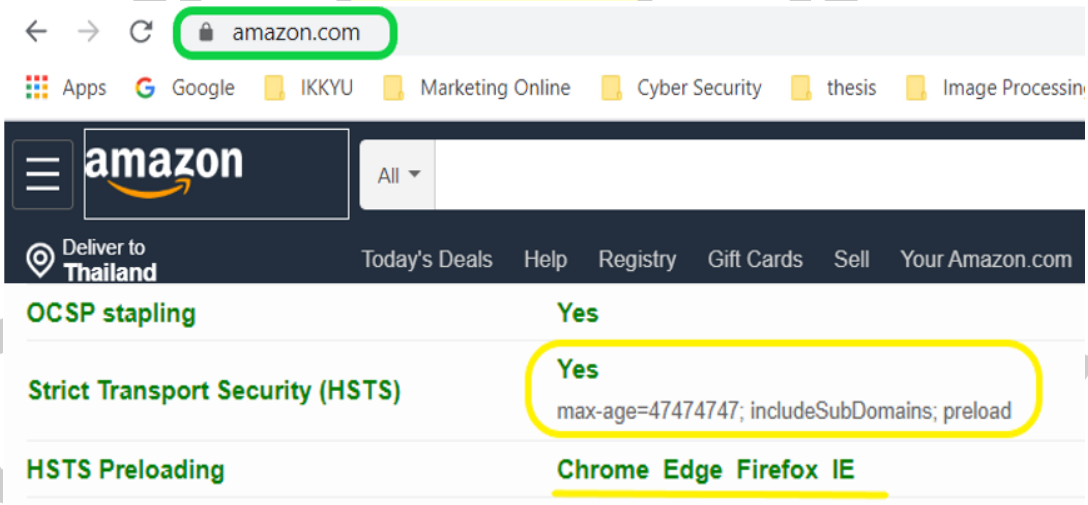
commerce Web ผลการทดลองสามารถทำลายระบบป้องกันปลด HTTPS ไปเป็น HTTP ได้ แม้มีการตั้งค่า HSTS Config ด้วยค่า Max-age ที่เหมาะสมเมื่อดูจาก HTTP Response Header แต่กลับสามารถโจมตี Strip และ Sniff ดักจับข้อมูลรหัสผ่านของเหยื่อได้ดังแสดง**Error! Reference source not found.**

ตารางที่ 4.6 ผลการโจมตีเว็บไซต์ E-commerce ด้วยวิธี SSL Stripping Attack

เว็บไซต์*	HSTS		SSL Strip Attack	
	Max-Age	Preload	SSL Strip	Data Sniff
L	47474747	Yes	×	×
M	31536000	No	✓	✓
N	31536000	No	✓	✓
O	31536000	No	✓	✓
P	31536000	No	✓	✓

** เพื่อสงวนชื่อเว็บไซต์ E-commerce จึงใช้อักษรย่อแทน*

✓ : สามารถโจมตีได้ × : ไม่สามารถโจมตีได้



ภาพที่ 4.7 ผลการทดลองเว็บ E-commerce ที่มีการตั้งค่า HSTS แบบ Preload

ภาพที่ 4.8 ผลการทดลองเว็บ E-commerce ที่ถูก SSL Strip และ Sniff

ตารางที่ 4.7 สรุปผลการโจมตีเว็บไซต์ธนาคารออนไลน์ในประเทศไทย

Web Browser	SSL Strip Attack		
	โจมตีได้	โจมตีไม่ได้	ดักจับข้อมูลไม่ได้
Google Chrome	4	1	0
Safari	4	1	0
Internet Explorer	4	1	0
Mozilla Firefox	4	1	0
Opera	4	1	0

4.2.3 ผลการโจมตีเว็บไซต์ระบบทะเบียนมหาวิทยาลัยด้วยวิธี SSL Stripping Attack

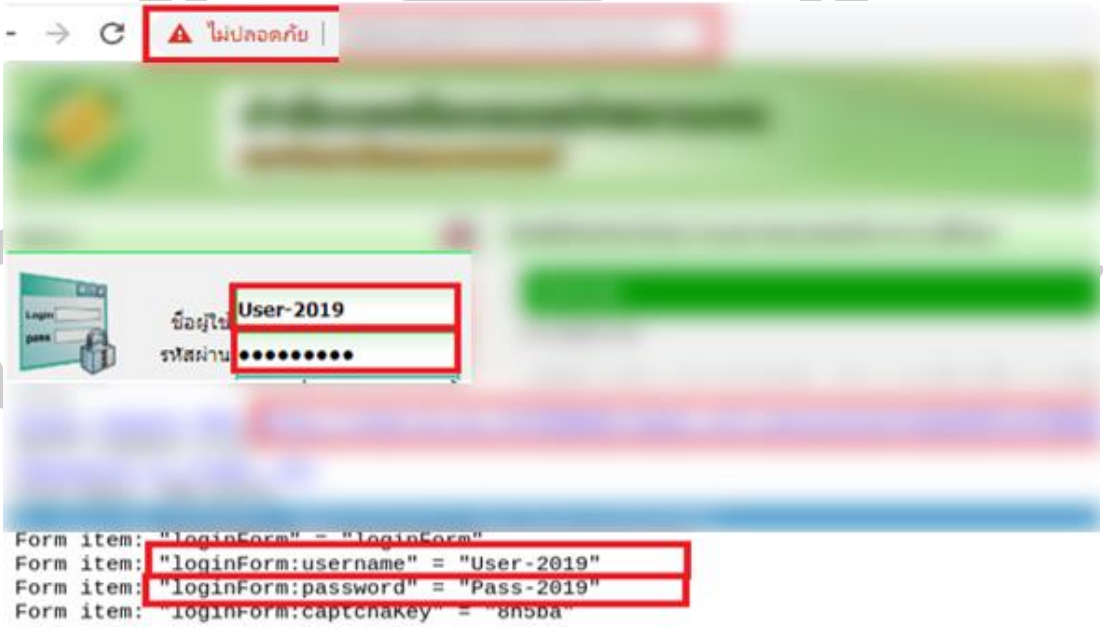
จากการทดลองโจมตีเว็บไซต์ระบบทะเบียนมหาวิทยาลัย จำนวน 11 เว็บ ด้วยวิธี SSL Stripping Attack ได้ผลลัพธ์ในการทดลองและรายละเอียดดังนี้ จาก**Error! Reference source not found.** จะเห็นได้ว่าผลการทดลองทั้ง 11 เว็บ สามารถทำลายระบบป้องกันปลด HTTPS ไปเป็น HTTP ได้ และพบ 3 เว็บที่มีการตั้งค่า HSTS Config ด้วยค่า Max-age ที่เหมาะสม เมื่อดูจาก HTTP Response Header แต่ก็ไม่ได้รับการป้องกันจากการโจมตีด้วย SSL Stripping Attack แต่อย่างใดตัวอย่างดังแสดงภาพที่ 4.9

ตารางที่ 4.8 ผลการโจมตีเว็บไซต์ระบบทะเบียนมหาวิทยาลัยด้วยวิธี SSL Stripping Attack

เว็บไซต์*	HSTS		SSL Strip Attack	
	Max-Age	Preload	SSL Strip	Data Sniff
Q	31536000	No	✓	✓
R	31536000	Yes	✓	✓
S	31536000	No	✓	✓
T	ไม่พบการ Configure HSTS		✓	✓
U	ไม่พบการ Configure HSTS		✓	✓
V	ไม่พบการ Configure HSTS		✓	✓
W	ไม่พบการ Configure HSTS		✓	✓
X	ไม่พบการ Configure HSTS		✓	✓
Y	ไม่พบการ Configure HSTS		✓	✓
Z	ไม่พบการ Configure HSTS		✓	✓
VI	ไม่พบการ Configure HSTS		✓	✓

* เพื่อสงวนชื่อเว็บไซต์ระบบทะเบียนมหาวิทยาลัย จึงใช้อักษรย่อแทน

✓ : สามารถโจมตีได้ ✕ : ไม่สามารถโจมตีได้



ภาพที่ 4.9 ผลการทดลองเว็บระบบทะเบียนมหาวิทยาลัย ที่ถูก SSL Strip และ Sniff

ตารางที่ 4.9 สรุปผลการโจมตีเว็บไซต์ระบบทะเบียนมหาวิทยาลัย

Web Browser	SSL Strip Attack		
	โจมตีได้	โจมตีไม่ได้	ดักจับข้อมูลไม่ได้
Google Chrome	11	0	0
Safari	11	0	0
Internet Explorer	11	0	0
Mozilla Firefox	11	0	0
Opera	11	0	0

4.2.4 ผลสรุปการโจมตีเว็บไซต์กลุ่มตัวอย่างทั้ง 27 เว็บไซต์

ตารางที่ 4.10 สรุปผลการโจมตีเว็บไซต์กลุ่มตัวอย่างทั้ง 27 เว็บไซต์

เว็บไซต์	SSL Strip Attack		
	โจมตีได้	โจมตีไม่ได้	ดักจับข้อมูลไม่ได้
เว็บไซต์ธนาคารออนไลน์ในประเทศไทย 11 เว็บ	10	1	2
เว็บไซต์ E-commerce 5 เว็บ	4	1	1
เว็บไซต์ระบบทะเบียนมหาวิทยาลัย 11 เว็บ	11	0	0

4.2.5 จาก4.2.4 ผลสรุปการโจมตีเว็บไซต์กลุ่มตัวอย่างทั้ง 27 เว็บไซต์

ตารางที่ 4.10 ผลการทดลองโจมตีเว็บไซต์กลุ่มตัวอย่างของระบบที่ให้บริการ Online Banking, E-commerce, ระบบทะเบียนมหาวิทยาลัย จำนวน 27 เว็บไซต์ พบว่า ทั้งสคริปต์การโจมตีแบบใหม่ของแฮกเกอร์ที่ใช้ Bettercap Script และสูตรการโจมตีแบบเก่าของ Moxie Marlinspike ก็ให้ผลการโจมตีเหมือนกัน ซึ่งหากสามารถทำการจู่โจมด้วย SSL Stripping Attack ได้สำเร็จ นั่นคือ การเปลี่ยนการใช้งานทางด้านเครื่องเหยื่อจากการใช้โพรโทคอล HTTPS เป็น HTTP ได้แล้วนั้น ผู้โจมตีจะสามารถดักจับข้อมูลสำคัญต่าง ๆ ของเหยื่อได้ไม่ว่าจะเป็น ชื่อผู้ใช้ รหัสผ่าน หรือข้อมูลอื่น ๆ ที่ถูกส่งไปยัง Server แต่ข้อสังเกตก็คือ หากบนเว็บไซต์ที่ถูกโจมตีใช้ SSL/TLS อย่างเดียวในการป้องกันระบบ จะทำให้ผู้โจมตีสามารถดักจับข้อมูลออกมาได้ในรูปของ Clear Text แต่ใน

กรณีของบางเว็บไซต์ หากมีการใช้ทั้ง SSL/TLS และการเข้ารหัส Hash Password ร่วมด้วยจะทำให้ผลของการโจมตีของ Hacker จะได้ข้อมูลออกมาในลักษณะของ Cipher Text ที่ไม่สามารถนำไปใช้งานได้ ซึ่งสามารถสรุปเพิ่มเติมได้ดังนี้

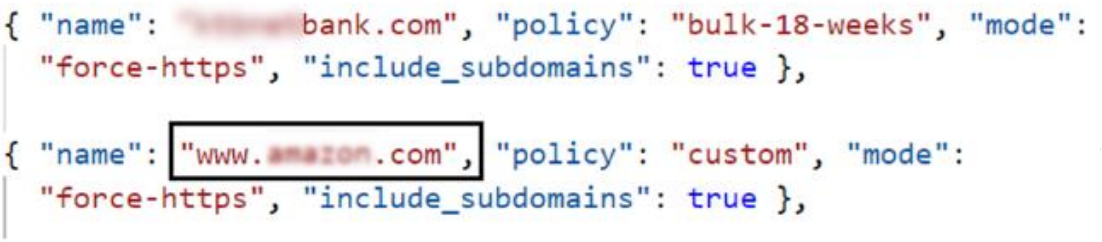
1) การโจมตี HTTPS ด้วยวิธีการ SSL Stripping Attack พบว่า ถึงแม้จะมีการใช้งาน SSL/TLS ในรูปแบบตลอดขบวนการ หรือ เฉพาะหน้า Login ยังเกิดปัญหาการโจมตีจากการจู่โจมด้วยเทคนิค SSL Stripping Attack ได้

2) การโจมตีด้วยเทคนิค SSL Stripping Attack ทั้งการโจมตีด้วย Bettercap ที่เป็นรูปแบบใหม่ และการโจมตีแบบเก่าของ Moxie Marlinspike พบว่าถึงแม้มีการ Configuration ปรับใช้กลไกของ HSTS ตามสูตรที่รู้กัน HSTS กลับไม่ประสบผลสำเร็จในการป้องกัน

3) การโจมตีด้วยเทคนิค SSL Stripping Attack สามารถทำการโจมตีได้บนทุกแพลตฟอร์มของฮาร์ดแวร์ ระบบปฏิบัติการ และเว็บเบราว์เซอร์

4.2.6 ผลการวิเคราะห์ HSTS Preload List

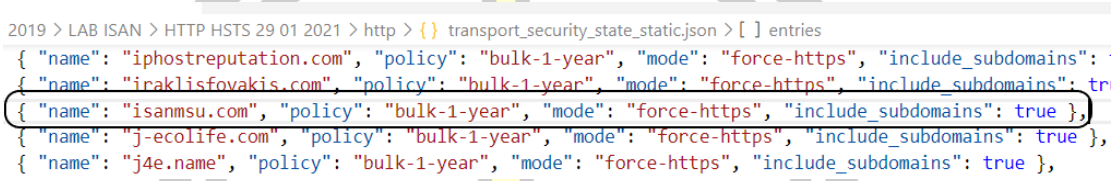
จากการศึกษาเว็บไซต์ที่มีการตั้งค่ากลไก HSTS Preload List พบว่า จะถูกบรรจุไว้ใน List ที่ https://chromiu m.googlesource.com/chromium/src/net/+/master/http/transport_security_state_static.json และเมื่อเว็บเบราว์เซอร์มีการ Update จะมีการดึงเอา List นี้ไปเก็บไว้ในเว็บเบราว์เซอร์ ผลการทดลองเว็บไซต์ที่มีการปรับใช้ HSTS แบบ Preload ที่ไม่สามารถโจมตีได้ มีธนาคารออนไลน์ B และเว็บไซต์ E-commerce L 2 เว็บไซต์ดังกล่าวล้วนถูกลิสต์ไว้ใน transport_security_state_static .json ดังแสดงภาพที่ 4.10



ภาพที่ 4.10 HSTS Preload List ในเว็บกลุ่มตัวอย่าง

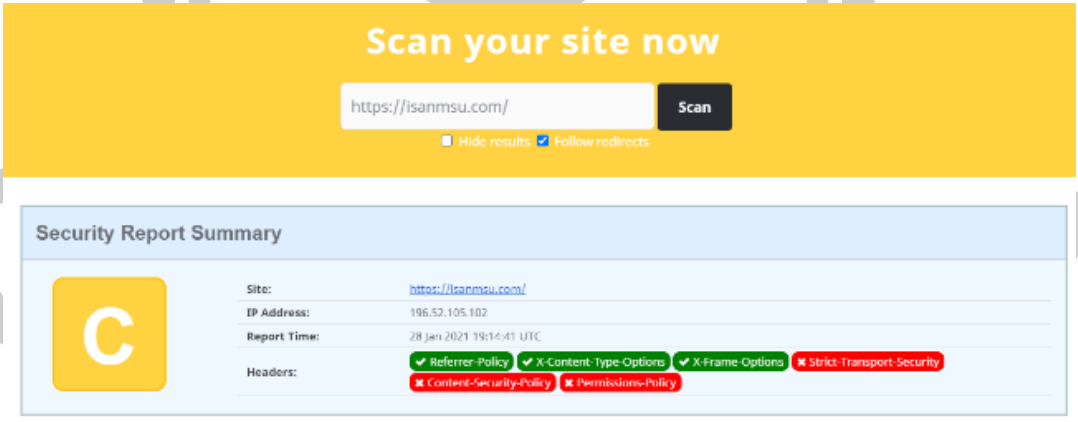
เพื่อให้เห็นผลการทดลองในมิติอื่น ๆ จึงนำเว็บไซต์อาสาสมัคร isanmsu.com มาทำการทดสอบ โดยเริ่มจากการ Inspect Network เพื่อเช็คค่า HTTP Response Header ได้ผลคือไม่พบการตั้งค่า HSTS หลังจากนั้นทำการโจมตีด้วย SSL Stripping Attack ผลเป็นที่น่าแปลกใจคือไม่สามารถ SSL Strip HTTPS ของเว็บไซต์อาสาสมัครได้ ดังนั้นจึงทำการตรวจสอบ HSTS Preload

List ดังแสดงภาพที่ 4.11 ทำให้รู้ว่าเว็บไซต์ isanmsu.com มีการปรับใช้ความมั่นคง HSTS แบบ Preload ซึ่งก่อนหน้านี้ที่เช็คค่า Header ไม่พบการตั้งค่า HSTS เพราะว่าหลังลงทะเบียนเสร็จแล้ว เจ้าของเว็บไซต์ได้ลบค่า Header HSTS ออก ทำให้รู้ว่าถึงลบกาตั้งค่า HSTS หลังจากลงทะเบียน Preload สำเร็จแล้วก็ไม่ส่งผลกระทบในการป้องกันแต่อย่างใด



ภาพที่ 4.11 HSTS Preload ของเว็บไซต์ isanmsu.com

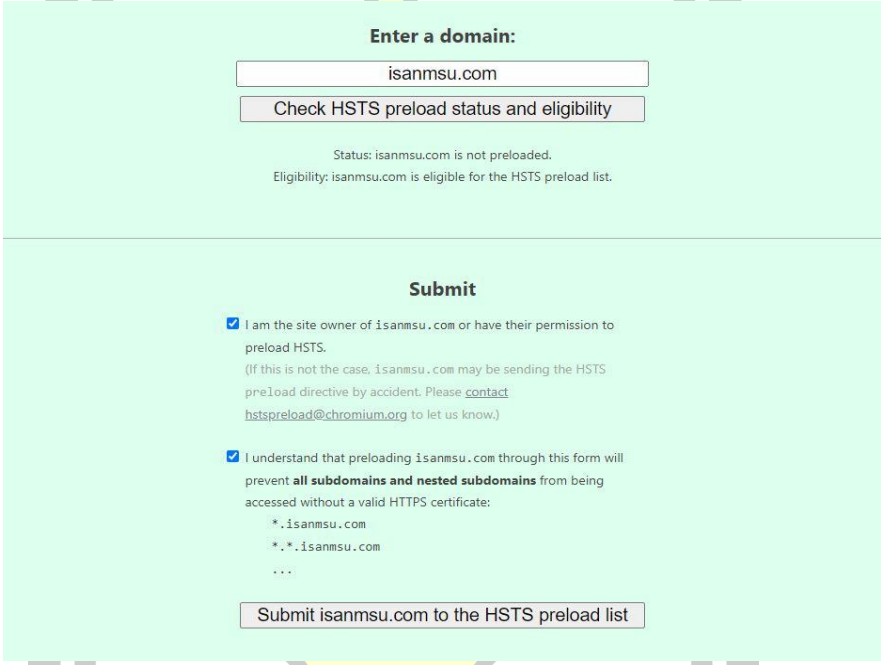
จากการทดสอบเว็บไซต์ที่จัดอันดับให้คะแนนความปลอดภัยของเว็บไซต์หลายบริการ เช่น https://securityheaders.com, www.serpworx.com และ ssllabs.com พบว่ามีข้อผิดพลาดในการประเมินผล HSTS ดังตัวอย่างแสดงภาพที่ 4.12 ที่ให้คะแนนความปลอดภัยไม่ผ่านในด้านการป้องกัน SSL Stripping Attack เพียงเพราะเช็คจากค่า HTTP Response Header เพียงอย่างเดียว ทั้งที่เว็บไซต์ isanmsu.com มีความปลอดภัยจากการถูกโจมตีดังกล่าว เพราะทำการตั้งค่า HSTS แบบ Preload และถูกเก็บไว้ใน Preload List เรียบร้อยแล้ว การค้นพบนี้เป็นองค์ความรู้สำคัญที่ควรนำไปปรับวิธีให้คะแนนความปลอดภัยเว็บไซต์ใหม่



ภาพที่ 4.12 ผลการ Scan Website isanmsu.com

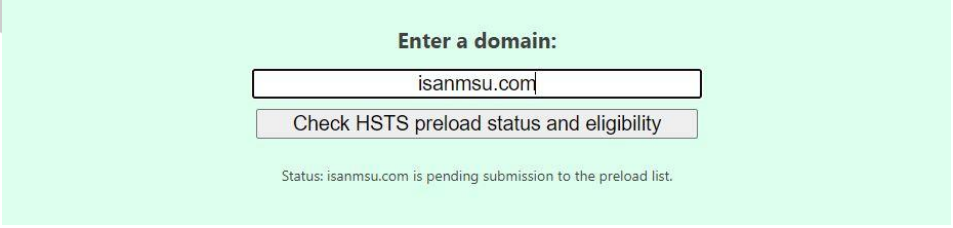
4.2.7 ผลการแนะนำการตั้งค่า HSTS แบบ Preload

HSTS Preload ได้รับการดูแลในโครงการ Chromium ที่ถูกพัฒนาโดย Google มีเป้าหมายเพื่อสร้างกลไกต่าง ๆ ให้มีความมั่นคงปลอดภัยและมีเสถียรภาพในการใช้งาน ผู้ดูแลระบบตั้งค่า HTTP Header ให้เป็น Strict-Transport-Security: maxage=31536000;includeSub Domains; preload และนำโดเมนเนม (Domain Name) เข้าตรวจสอบและลงทะเบียนใช้งานได้ที่เว็บไซต์ hstspreload.org ดังแสดงภาพที่ 4.13



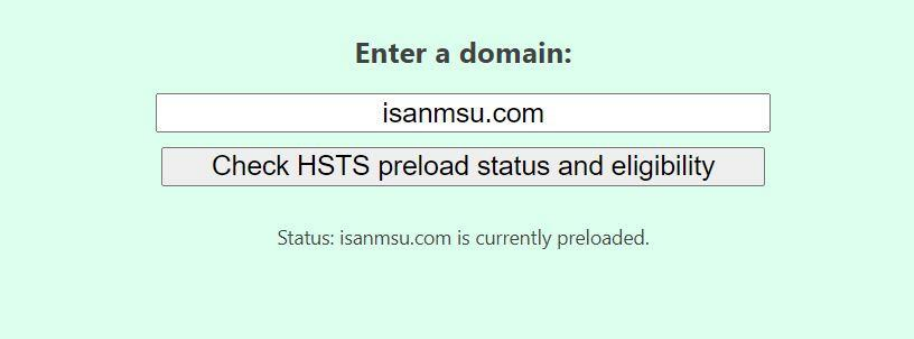
ภาพที่ 4.13 ตรวจสอบโดเมนเพื่อลงทะเบียน HSTS Preload

หลังจากนำโดเมนเนมลงทะเบียนสำเร็จระบบจะแสดงสถานะบอกให้รู้ว่ามีการส่งไปยัง Preload List เพื่อเตรียมความพร้อมในการอัพโหลดเข้าฐานข้อมูล HSTS Preload ในเว็บเบราว์เซอร์ดังแสดงภาพที่ 4.14



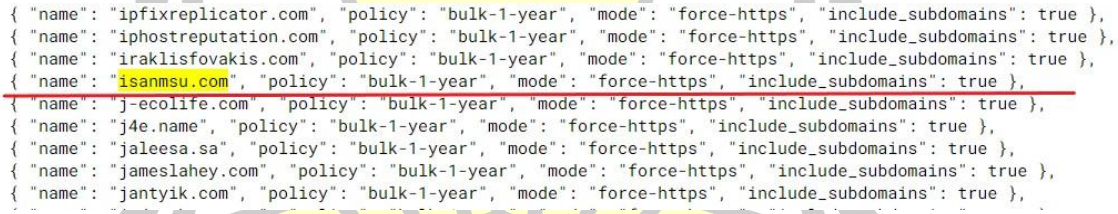
ภาพที่ 4.14 สถานะการส่งคำร้อง HSTS Preload ลงทะเบียนสำเร็จ

การลงทะเบียน HSTS Preload ที่สามารถป้องกันการถูกโจมตีได้ ต้องรอให้ Google ในโครงการ Chromium นำข้อมูลโดเมนอัพโหลดเข้าฐานข้อมูลเพื่อบรรจุลงในเบราว์เซอร์ (Browser) จึงจะสามารถทำงานได้อย่างสมบูรณ์ โดยสามารถเข้าเช็คสถานะดังกล่าวได้ที่ hstspreload.org ดังแสดงภาพที่ 4.15



ภาพที่ 4.15 สถานะการทำงาน Preload HSTS

HSTS Preload ที่ได้รับการ Submission จะมีการฝั่งอยู่ที่เว็บเบราว์เซอร์สามารถเช็คข้อมูลได้ที่ Chromium HSTS Preload List: https://cs.chromium.org/chromium/src/net/http/transport_security_state_static.json



ภาพที่ 4.16 HSTS Preload List

4.3 ผลการวิเคราะห์ปัญหากลไก HSTS และการกลับมาจู่โจมได้ใหม่ของ SSL Stripping Attack

เพื่อให้เข้าใจผลการทดลองในเชิงลึกยิ่งขึ้น จึงได้ทำการศึกษาเครื่องมือที่ใช้ในการโจมตีด้วย SSL Stripping Attack ทั้ง Ettercap ที่เป็นแบบเดิมของ Marlinspike และแบบใหม่ของผู้โจมตีที่ใช้ Bettercap จากการศึกษา Script แบบใหม่ของ Bettercap พบการโจมตีที่เรียกว่า HSTS Hijack ที่ทำให้ทราบสาเหตุการล้มเหลวของกลไก HSTS ซึ่งจะได้แสดงผลการทดลองดังนี้

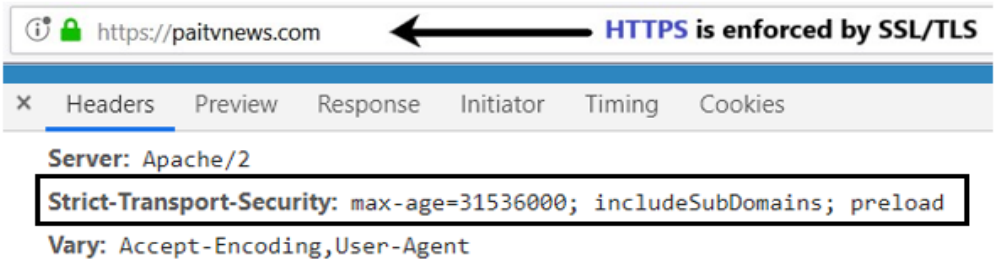
4.3.1 ผลการทดลอง HSTS Directive

การทดลองครั้งนี้เลือกเว็บไซต์อาสาสมัคร paitvnews.com ที่มีการปรับใช้ HTTPS ร่วมกับ HSTS ในการป้องกัน SSL Stripping Attack ในการทดลองได้ศึกษาการทำงานของกลไก HSTS ซึ่งมีรูปแบบการ Request – Response ที่ทำงานในฝั่ง Server ผ่าน HTTP Header การโจมตีจึงทำการแก้ไขโค้ดภาษา javascript ของ Bettercap ชื่อไฟล์ hstshijack.js ดังแสดงภาพที่ 4.17 เพิ่มเข้าไปในไฟล์ที่ใช้ในการติดต่อไปยังเซิร์ฟเวอร์ของ Bettercap ที่ติดตั้งใน Kali Linux 2020.1

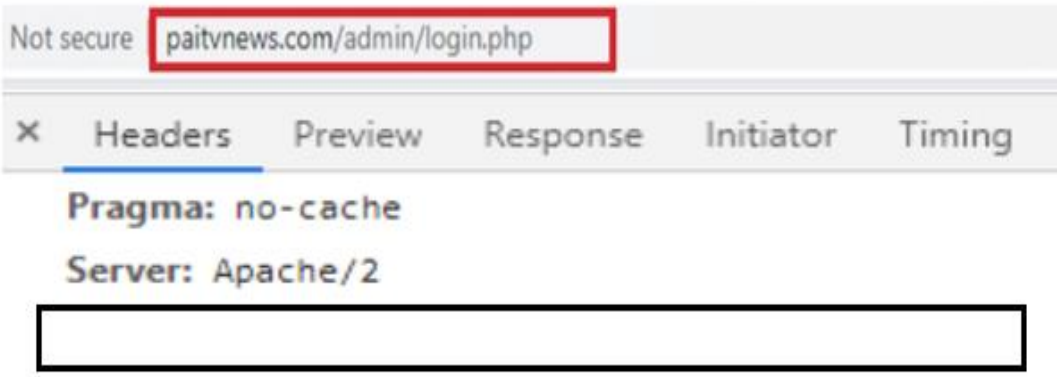
```
function onRequest(req, res) {
    res.Status      = 200;
    res.ContentType = "text/html";
    res.Body        = readFile("caplets/www/index.html");
    headers         = res.Headers.split("\r\n")
    for (var i = 0; i < headers.length; i++) {
        header_name = headers[i].replace(/:.*/, "")
        res.RemoveHeader(header_name);
    }
    res.SetHeader("Connection", "close");
}
```

ภาพที่ 4.17 รูปแบบโค้ด javascript ใน Bettercap ที่ใช้ในการโจมตี HSTS

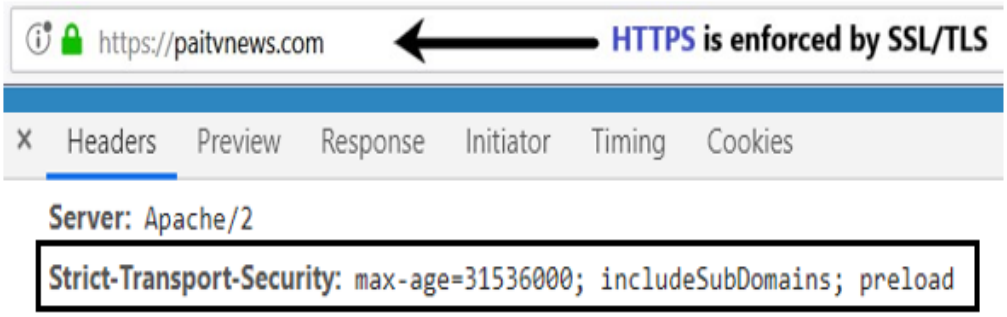
เพื่อให้เข้าใจผลการโจมตีของ Module Bettercap จึงทำการแก้ไข Script ที่ใช้ในการโจมตี โดยลบ HSTS Header ที่ตั้งค่ามาจากฝั่ง Server ดังแสดง**Error! Reference source not found.** – ภาพที่ 4.19และทำการโจมตีเปลี่ยนค่า Header HSTS เข้าไปใหม่โดยที่ไม่เคยมีการตั้งค่ามาก่อนดังแสดงภาพที่ 4.20 – **Error! Reference source not found.** ซึ่งสรุปได้ว่าการโจมตีดังกล่าวสามารถเพิ่มและลบการตั้งค่ากลไก HSTS ได้อย่างง่ายดาย โดยไม่มีการแจ้งเตือนความผิดปกติใด ๆ จาก Web Browser



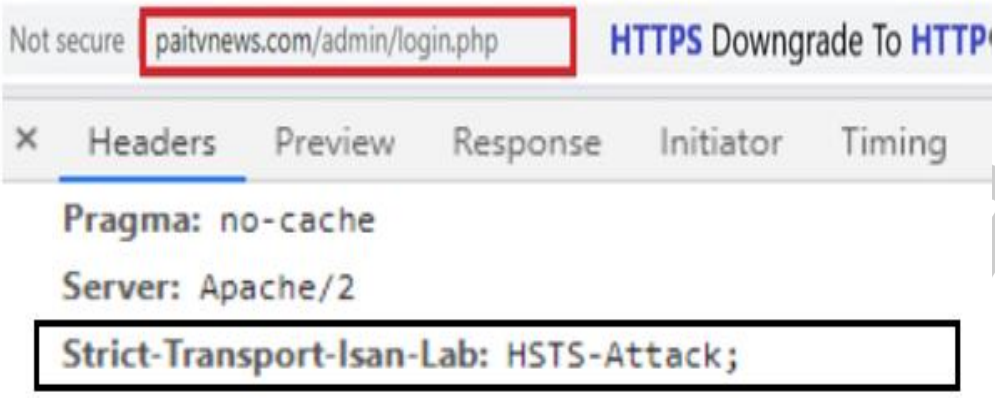
ภาพที่ 4.18 ก่อนถูกโจมตียังเห็นค่า Header HSTS ปกติ



ภาพที่ 4.19 หลังถูกโจมตีค่า Header HSTS จะถูกปลดออก



ภาพที่ 4.20 ก่อนถูกโจมตี inject HSTS header



ภาพที่ 4.21 หลังจากถูกโจมตี inject HSTS header

4.3.1 ผลการทดลอง HSTS Preload

การทดลองโจมตีได้เลือกเว็บไซต์ facebook.com จากการศึกษาพบว่าการปรับใช้ HSTS แบบ Preload ดูมีความมั่นคงปลอดภัยกว่า HSTS แบบธรรมดา เนื่องจากมีการบังคับเชื่อมต่อ HTTPS ตั้งแต่เริ่มต้นการสื่อสาร แต่การที่จะบอกให้เว็บเบราว์เซอร์เชื่อมต่อ HTTPS ตั้งแต่เริ่มต้นการสื่อสารได้ก็จำเป็นต้องมีข้อมูลเว็บไซต์เหล่านั้นอยู่ในฐานข้อมูล Web Browser เสียก่อน เหตุนี้เองจึงมีแนวคิดทำการทดสอบโดยใช้เทคนิคการการโจมตีที่เรียกว่า Homograph Attack โดยอาศัยความสามารถ Bettercap ใน Kali Linux 2020.1 ซึ่งมีรูปแบบโค้ดคำสั่งที่ใช้โจมตีดังแสดงภาพที่ 4.22

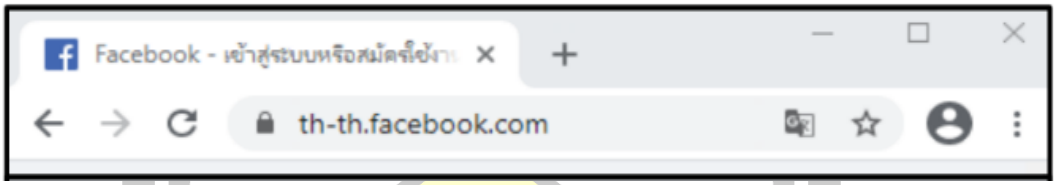
```
set hstshijack.targets          facebook.com, *.facebook.com
set hstshijack.replacements     facebook.corn, *.facebook.corn

set http.proxy.script  ********************************

set dns.spoof.domains           facebook.corn, *.facebook.corn
http.proxy  on
dns.spoof   on
```

ภาพที่ 4.22 รูปแบบคำสั่งของเทคนิค Homograph Attack ที่ใช้ในการโจมตี HSTS Preload

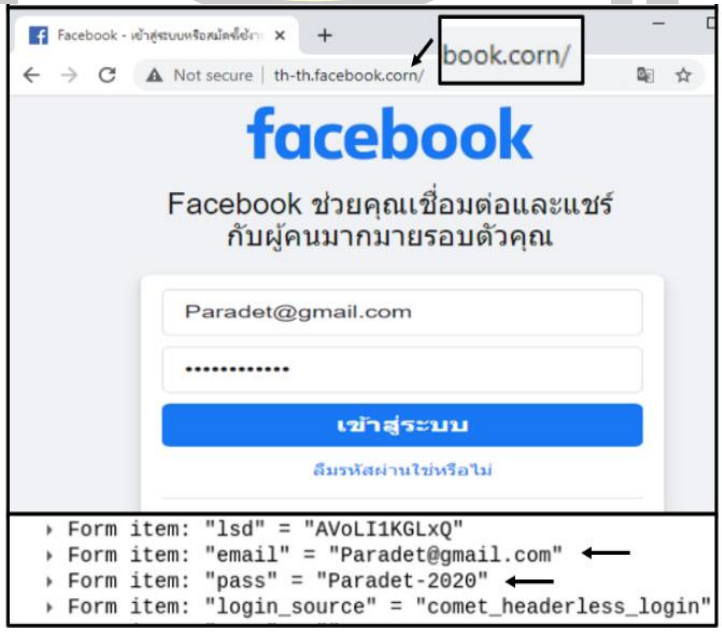
เพื่อให้ทราบผลการโจมตี facebook.com จึงทำการการเช็ค HSTS Preload List ในฐานข้อมูลของไฟล์ transport_security_state_static.json พบว่า facebook.com มีการปรับใช้ HSTS Preload ดังแสดง**Error! Reference source not found.** เพื่อการป้องกันการถูกโจมตี SSL Stripping Attack จากการจู่โจมด้วยเทคนิค Homograph Attack ได้ทำการปลอมแปลงโดเมน (DNS Spoof) ในระดับการโจมตี Top-Level Domain (TLD) แปลงจาก .com เป็น .corn ผลการโจมตีพบว่า สามารถปลดระบบป้องกันออกได้อย่างสมบูรณ์ และทำการดักจับข้อมูลของเหยื่อได้อย่างง่ายดายดังแสดงภาพที่ 4.24 - ภาพที่ 4.25

```
{
  "name": "facebook.com", "policy": "custom",
  "mode": "force-https", "pins": "facebook", "include_subdomains_for_pinning": true
},
{ "name": "www.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "m.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "tablet.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "secure.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "pixel.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "apps.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "upload.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "developers.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "touch.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "mbasic.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "code.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "t.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "mtouch.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "business.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{ "name": "research.facebook.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
{
  "name": "messenger.com", "policy": "custom",
  "mode": "force-https", "pins": "facebook", "include_subdomains_for_pinning": true
},
{ "name": "www.messenger.com", "policy": "custom", "mode": "force-https", "include_subdomains": true, "pins": "facebook" },
```


ภาพที่ 4.23 facebook.com ที่อยู่ใน HSTS Preload List



ภาพที่ 4.24 ก่อนถูกโจมตียังมีการบังคับใช้ HTTPS และ TLD ยังเป็น .com อยู่



ภาพที่ 4.25 หลังจากถูกโจมตีการบังคับใช้ HTTPS จะถูกปลดออก และ TLD จะเป็น .corn

4.4 ผลการออกแบบป้องกันการถูกดักจับข้อมูลที่ถูกโจมตีด้วยวิธี SSL Stripping Attack

จากแนวคิดที่ต้องการรักษาข้อมูลในระหว่างการสื่อสารเว็บไซต์ จึงทำการปรับใช้ Salted Hash Password (SHP) เพื่อเสริมสร้างเกราะป้องกันซ้อนทับ SSL/TLS อีกชั้น หากถูกโจมตีแบบ SSL Striping Attack หรือโพรโทคอลมาตรฐานทำงานผิดพลาด ไม่ได้รับการป้องกัน แต่หากมีการเข้ารหัส Message Encryption ที่ดี ถึงแม้ว่าถูกดักจับข้อมูลระหว่างการสื่อสารก็อยู่ในรูป Cipher Text การออกแบบระบบครั้งนี้ ได้พัฒนาเว็บไซต์ต้นแบบคือ isan-banking ทำงานร่วมกับ Mobile OTP ซึ่งจะได้แสดงผลลัพธ์ในการทดลองดังนี้

4.4.1 ผลการทดลอง isan-banking ร่วมกับ Mobile OTP

1) การทำงานของ Mobile OTP

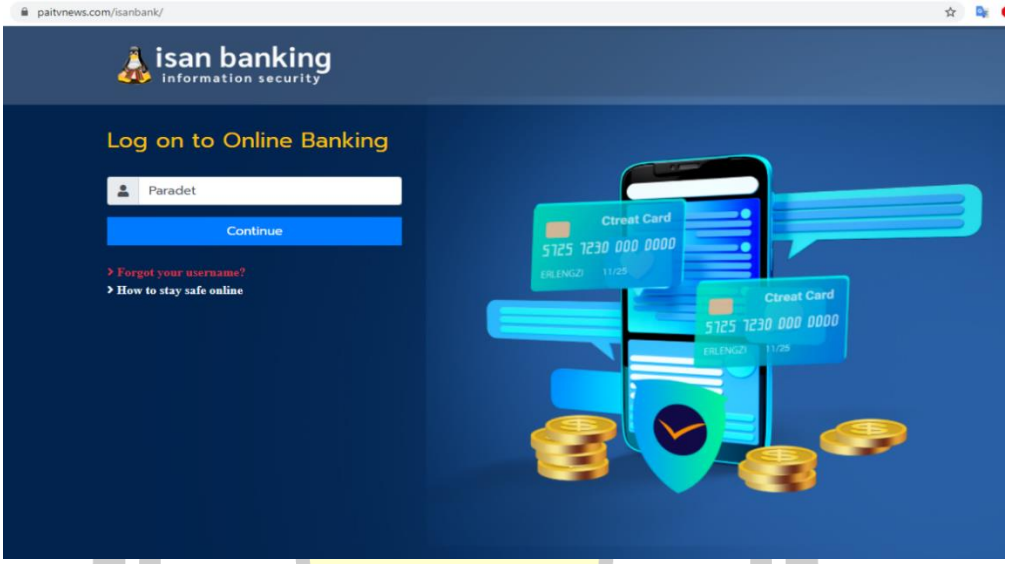
การแสดงรหัส OTP ของทางฝั่ง Mobile OTP เมื่อผู้ใช้งานต้องการเข้าสู่ระบบเพื่อใช้งานเว็บไซต์ isan-banking ต้องใช้รหัส OTP ที่ได้จาก Mobile OTP ในการยืนยันเข้าสู่ระบบ ซึ่งรหัส OTP จะถูกสร้างขึ้นใหม่ในทุก ๆ 30–60 วินาทีแล้วแต่การตั้งค่าดังแสดง**Error! Reference source not found.**

ภาพที่ 4.26 รหัส OTP จาก Mobile-OTP

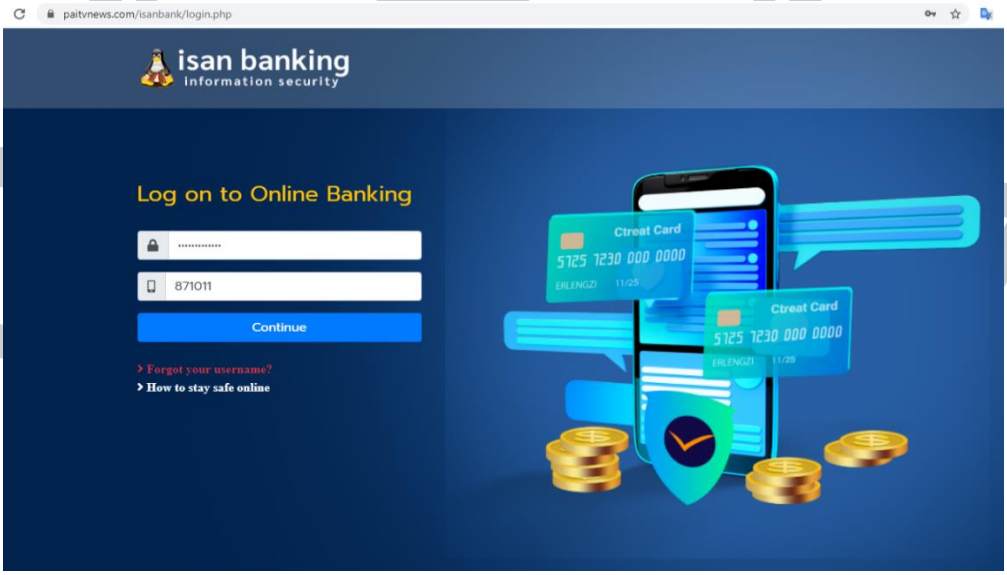
พหูน ปณ ฑิโต ชีเว

2) หน้า Login เข้าสู่ระบบ isan-banking

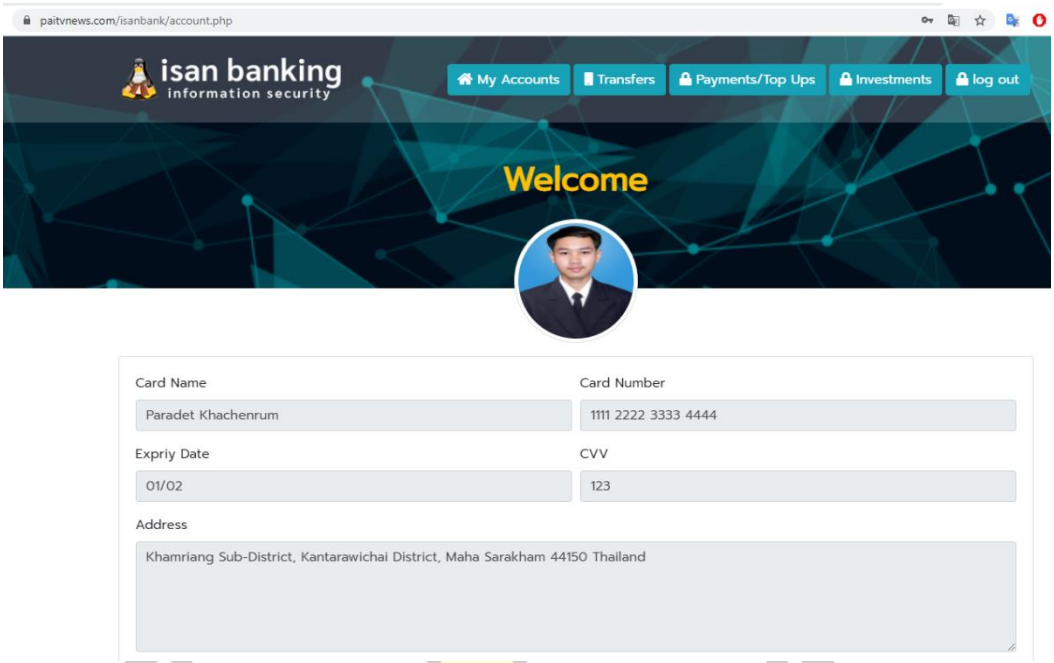
เริ่มแรกระบบจะให้ผู้ใช้งานกรอก Username เพื่อพิสูจน์ตัวตนดังแสดง**Error! Reference source not found.** ถ้าตรวจสอบแล้วถูกต้องก็จะเข้าสู่หน้าถนัดไป เพื่อทำการ Login ในหน้า Password และ OTP หลังจากทำการยืนยันข้อมูลดังแสดงภาพที่ 4.28 หากข้อมูลถูกต้องทั้งหมดก็จะสามารถเข้าสู่ระบบ isan-banking ได้สำเร็จดังแสดงภาพที่ 4.29



ภาพที่ 4.27 หน้า Login เพื่อยืนยัน Username



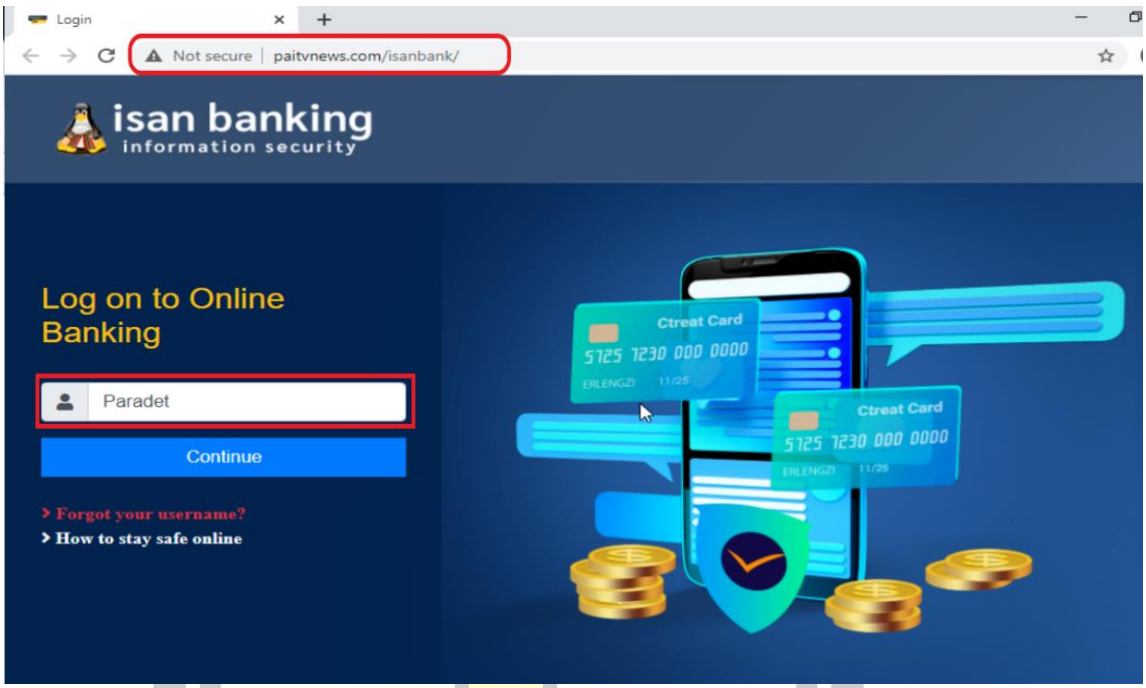
ภาพที่ 4.28 หน้า Login เพื่อยืนยัน Password กับ OTP



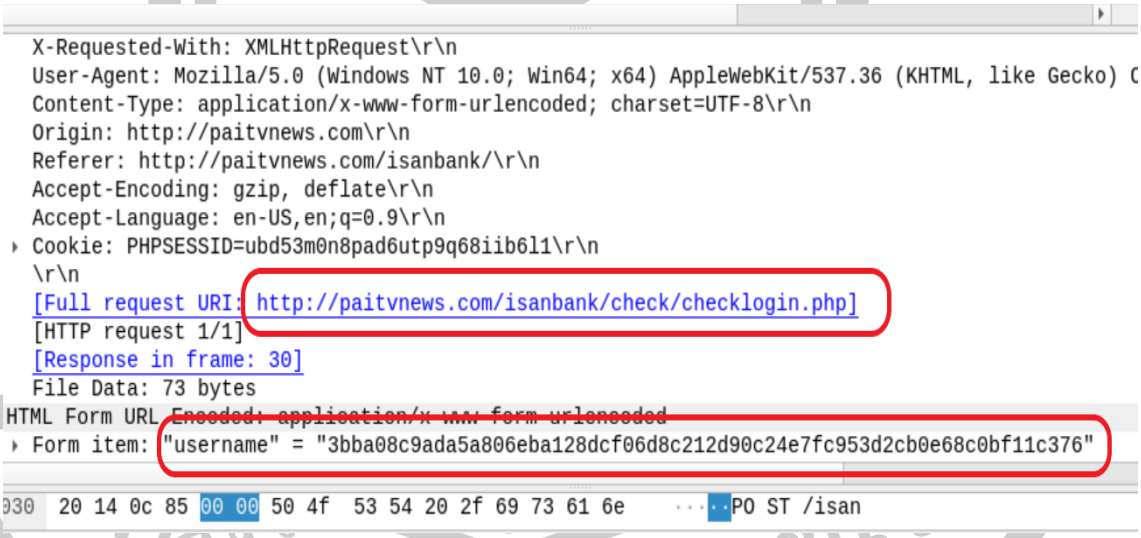
ภาพที่ 4.29 ผลการ Login เข้าสู่ระบบสำเร็จ เว็บ isan-banking

4.4.2 ผลการทดสอบความมั่นคง isan-banking ด้วยวิธี SSL Stripping Attack

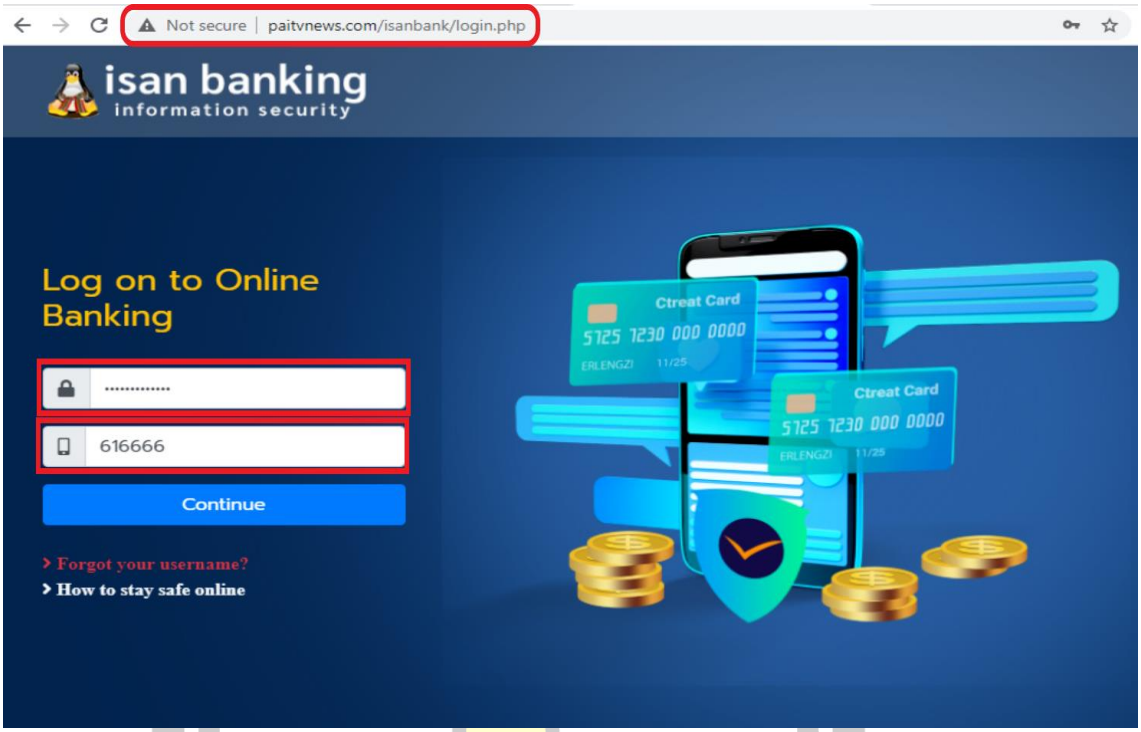
การทดสอบ isan-banking โจมตีด้วยเทคนิค SSL Stripping Attack พบว่าเมื่อสามารถทำลายโพรโทคอล SSL/TLS เปลี่ยนการสื่อสารจาก HTTPS เป็น HTTP ได้ จะส่งผลให้สามารถดักสกัดข้อมูล Username, Password และ OTP ระหว่างการสื่อสารบนระบบเครือข่ายได้ ดังแสดงภาพที่ 4.30 ภาพที่ 4.34 แต่จากผลการทดลองดังกล่าวก็ยังไม่สามารถหาประโยชน์จากข้อมูลที่ดักจับได้ เนื่องจากระบบที่พัฒนาขึ้นมีการนำ Username, Password เข้ากระบวนการ Hash ด้วยภาษา Javascript ที่มีการประมวลผลฝั่งไคลเอนต์ Client Side Script ก่อนที่จะถูกส่งผ่านไปยังเครื่อง Server ทำให้ถึงแม้มีการดักจับข้อมูล แต่ข้อมูลก็จะอยู่ในรูปแบบ Hash ที่ไม่สามารถนำมาคำนวณย้อนกลับเพื่อหาข้อมูลที่แท้จริงได้และในส่วน OTP ถึงแม้จะอยู่ในรูป Clear text แต่ค่า OTP ก็จะมีอายุใช้งานเพียง 30-60 วินาที และจะสร้าง OTP ใหม่ เพื่อนำมาใช้ใช้งานใหม่ไปเรื่อย ๆ ซึ่งเป็นการยากที่จะนำค่า OTP ที่ดักจับได้ในขณะนั้นไปใช้งานเพื่อยืนยันการเข้าสู่ระบบ



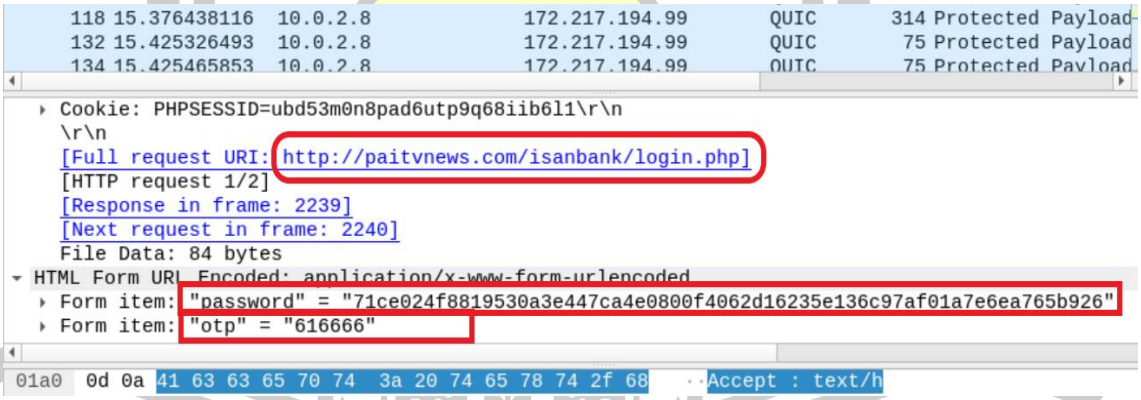
ภาพที่ 4.30 ผลหน้า Login User เมื่อถูกโจมตี SSL Strip การบังคับใช้ HTTPS จะถูกปลดออก



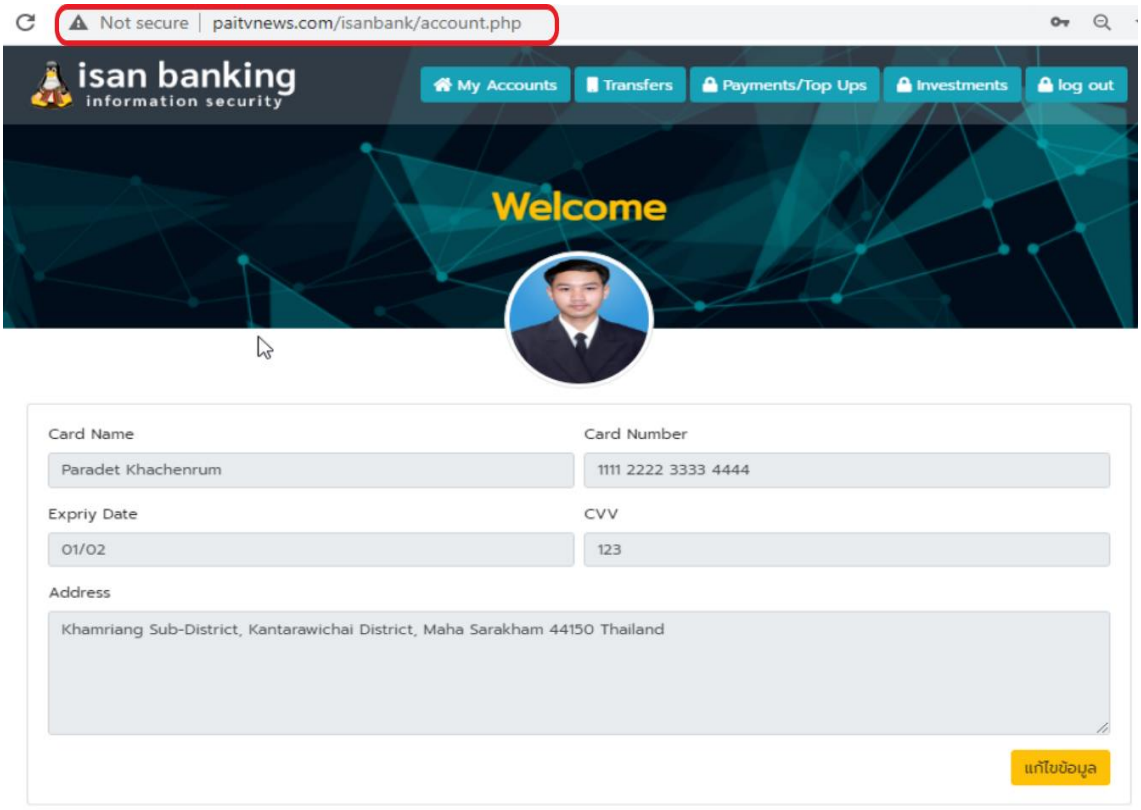
ภาพที่ 4.31 ผลลัพธ์การ Strip และ Sniff หน้า Login Username



ภาพที่ 4.32 ผลหน้า Login Password และ OTP โจมตีด้วย SSL Strip



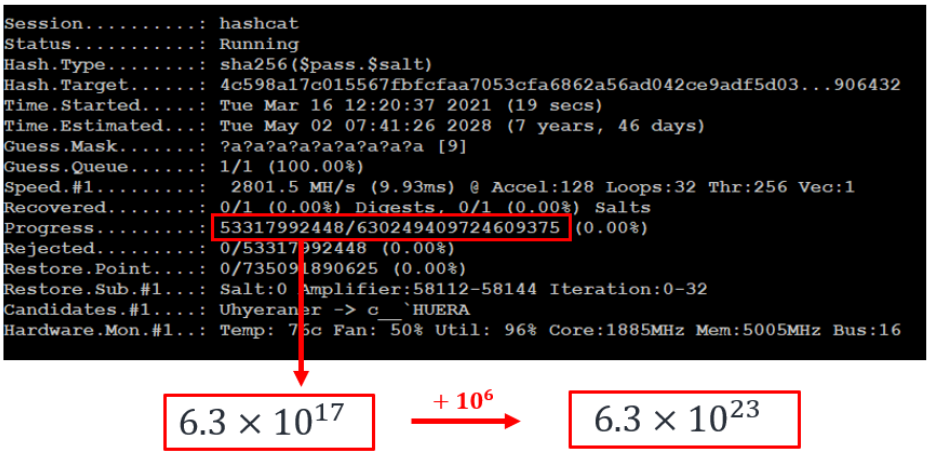
ภาพที่ 4.33 ผลลัพธ์การ Strip และ Sniff หน้า Login Password และ OTP



ภาพที่ 4.34 ผลการ Login เข้าสู่ระบบสำเร็จ

4.4.3 การทดสอบ Brute Force Attack ระบบ isanbanking

จากการทดสอบ Brute Force Attack ระบบ isan banking เพื่อถอดค่า Hash ต้องทำการบวกค่า Salt เท่ากับ 10^6 และบวกเข้ากับค่า Hash เดิมจะได้เท่ากับ 10^{23} ดังแสดงภาพที่ 4.35 อุปกรณ์ที่ใช้ในการทดลอง Windows 10, CPU Core i5 2320, Ram 4 GB, NVIDIA GeForce GTX 1080 Ti ความเร็วในการถอดรหัสที่ 2801.5 MH/s และใช้โปรแกรม Hashcat ในการ Brute Force Attack จากการคำนวณเพื่อหาระยะเวลาที่ใช้ในการถอดค่า Hash พบว่าต้องใช้เวลาโดยประมาณ 7,130,884 ปี ดังแสดงการคำนวณภาพที่ 4.36



ภาพที่ 4.35 ค่า Hash + Salt ของระบบ isan banking

วิธีการคำนวณให้นำค่า Hash and Salt decryption process ÷ Speed of decryption of GTX1080 × SEC × MIN × Hour × Year แทนค่าตามสูตรดังแสดงภาพที่ 4.36

$$\frac{6.3 \times 10^{23}}{2801500000 \times 60 \times 60 \times 24 \times 365} = 7{,}130{,}883.08041$$

ภาพที่ 4.36 ผลการถอดค่า hash ระบบ isan banking ที่ใช้การ์ดจอ NVIDIA GTX 1080 Ti

จากการศึกษา GPU Farms ของ University of Hong Kong [46] ที่ใช้การ์ดจอ NIVIDA GeForce RTX 2080 Ti ถึง 100 เท่า ซึ่งแต่ละตัวมีความเร็วในการถอดรหัสที่ 5519.1 MH/s นำมาคิดเป็น 100 เท่าจะได้ 5519.1 x 100 = 551,910 MH/s จากการคำนวณตามสเปคดังกล่าว โดยเอาค่า Hash ที่ทำการทดลองก่อนหน้านี้มาเป็นตัวตั้ง จะได้เท่ากับ 6.3×10^{23} นำมาหารกับความเร็วของอุปกรณ์เพื่อหาเวลาที่ใช้ในการ Brute Force Attack ดังตัวอย่างที่แสดงนี้

$$\frac{6.3 \times 10^{23}}{551{,}910{,}000{,}000 \times 60 \times 60 \times 24 \times 365} = 36196.4245517$$

ผลปรากฏว่า NIVIDA GeForce RTX 2080 Ti ที่ 100 เท่า จากการคำนวณต้องใช้เวลาในการถอดค่า Hash ประมาณ 36,197 ปี ผลการเปรียบเทียบนี้ชี้ให้เห็นว่าผู้โจมตีที่ใช้อุปกรณ์

ดังกล่าวไม่สามารถถอดรหัสได้ทันภายในเวลา 30-60 วินาที ของค่า OTP ที่มีการเปลี่ยนใหม่ไปเรื่อย ๆ

ในการทดสอบเพื่อให้เห็นผลแตกต่างยิ่งขึ้นจึงนำสเปกซุปเปอร์คอมพิวเตอร์จาก IBM [47] คือ IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR InfiniBand มี 2.28 ล้านคอร์ และความเร็ว 122.3 Petaflop แทนค่าตามสูตรจะได้ดังนี้

$$\frac{6.3 \times 10^{23}}{122.3 \times 10^{15} \times 60 \times 60 \times 24} = 59.6211501772$$

จากผลการคำนวณของซุปเปอร์คอมพิวเตอร์ของ IBM ในการ Brute Force Attack เพื่อถอดค่า hash จะใช้เวลาโดยประมาณ 60 วัน ซึ่งก็ยังไม่สามารถถอดค่า Hash ได้ทันภายในเวลา 30-60 วินาที ของค่า OTP ที่มีการเปลี่ยนใหม่ไปเรื่อย ๆ จึงทำให้การใช้งาน Mobile OTP มีความมั่นคง

บทที่ 5

สรุป

5.1 สรุปผล

การโจมตีด้วยเทคนิค SSL Stripping Attack เป็นภัยคุกคามร้ายแรงและตกเป็นคดีที่สำคัญในทั่วโลกรวมถึงประเทศไทย ระบบธนาคารออนไลน์ (Internet Banking) ระบบการค้าอิเล็กทรอนิกส์ (E-Commerce) หรือในหลายเว็บไซต์ถูกโจมตีดักจับข้อมูลที่สำคัญของผู้ใช้ SSL Stripping Attack จึงถือเป็นภัยคุกคามที่สร้างปัญหาให้กับระบบเว็บไซต์มายาวนาน งานวิจัยนี้ได้ทำการวิเคราะห์ปัญหาการทำงานที่ผิดปกติของกลไก HSTS เพื่อประเมินหาช่องโหว่การกลับมาโจมตีได้ใหม่ของ SSL Stripping Attack และเสนอแนวทางในการป้องกันเว็บไซต์จากการถูกโจมตี HTTPS จากผลการวิจัยสามารถสรุปได้ดังนี้

5.2 ผลที่ได้จากการวิจัย

5.2.1 องค์ความรู้ใหม่ที่ได้

1) ผลการตรวจสอบการใช้งานกลไก HSTS ที่เป็นเทคโนโลยีที่ใช้ในการป้องกัน SSL Stripping Attack พบว่า เว็บไซต์ธนาคารออนไลน์ในประเทศไทย, เว็บไซต์ E-commerce และเว็บไซต์ระบบทะเบียนมหาวิทยาลัย หลายแห่งยังมีการปรับใช้กลไก HSTS ไม่ถูกต้อง เนื่องจากทำการตั้งค่า HSTS แบบ None Preload ที่เว็บเซิร์ฟเวอร์แบบเดิมที่ไม่ได้รับการสนับสนุนแล้ว และบางเว็บไซต์แย่ถึงขั้นไม่มีการ Configuration กลไก HSTS เลย ซึ่งจากผลการทดลอง 27 เว็บไซต์ พบมีเพียง 1 เว็บไซต์ธนาคารออนไลน์ในประเทศไทย และ 1 เว็บไซต์ E-commerce ที่มีการ Preload HSTS อย่างถูกต้อง

2) การทำงานที่ Web Browser ในปัจจุบัน ไม่สนับสนุนการตั้งค่า HSTS ที่รับค่าจาก HTTP Response Header ที่ส่งผ่านอินเทอร์เน็ตอีกแล้ว เนื่องจากการโจมตีด้วย HSTS Hijacking สามารถทำลายการตั้งค่าของกลไก HSTS ออกได้

3) การปรับใช้กลไก HSTS ที่เหมาะสมเพื่อช่วยป้องกัน SSL Stripping Attack ต้องทำการตั้งค่า HSTS แบบ Preload เท่านั้น ซึ่งการตั้งค่า HSTS Configuration ที่ Web Server ไม่มีประโยชน์อีกต่อไป เพราะฝ่าย Hacker สามารถปลดค่า Header HSTS ออกได้

4) เว็บไซต์ที่มีการตั้งค่า HSTS แบบ Preload จากผลการทดสอบถึงแม้สามารถป้องกันการโจมตีด้วย SSL Stripping Attack ได้ แต่หากผู้โจมตีในระดับ Professional Hacker ที่มี

ความเชี่ยวชาญ ทำการจู่โจมด้วยเทคนิค Homograph Attack ก็ย่อมที่จะสามารถปลดค่ากลไก HSTS แบบ Preload ออกได้

5) ระบบตรวจสอบความปลอดภัยของเว็บไซต์ที่ Scan Website เพื่อเช็คว่าป้องกัน SSL Stripping Attack ได้หรือไม่ ใช้วิธีตรวจสอบแค่ HTTP Response Header HSTS จะไม่สามารถเชื่อถือได้ ต้องทำการตรวจสอบในฐานข้อมูล HSTS Preload List เท่านั้น จึงจะได้ผลที่ถูกต้อง

5.2.2 แนวทางการแก้ไขปัญหา

ในงานวิจัยนี้ได้พัฒนาระบบต้นแบบเว็บไซต์ isanbanking ที่รักษาความปลอดภัยของข้อมูลด้วย Salted Hash Password ร่วมกับ Mobile OTP เพื่อช่วยป้องกันการถูกดักจับข้อมูลระหว่างการสื่อสารในอินเทอร์เน็ต จากการทดสอบโจมตีด้วยวิธี SSL Striping Attack ที่มีลักษณะการทำงานคือจะบังคับเปลี่ยนแปลงโพรโทคอลในการสื่อสารจาก HTTPS เป็น HTTP ซึ่งทำให้ไม่ปลอดภัยระหว่างการสื่อสาร ผลการทดลองพบว่าเว็บไซต์ต้นแบบที่พัฒนาขึ้นสามารถป้องกันการโจมตีดังกล่าวได้ โดยให้ผลลัพธ์คือแฮกเกอร์ที่ทำการโจมตีดักจับข้อมูลจะได้ค่า Hash ที่ไม่สามารถนำไปใช้ประโยชน์ได้ จึงทำให้การสื่อสารของระบบเว็บไซต์ isanbanking มีความมั่นคงปลอดภัยจากการถูกโจมตีด้วยวิธี SSL Striping Attack

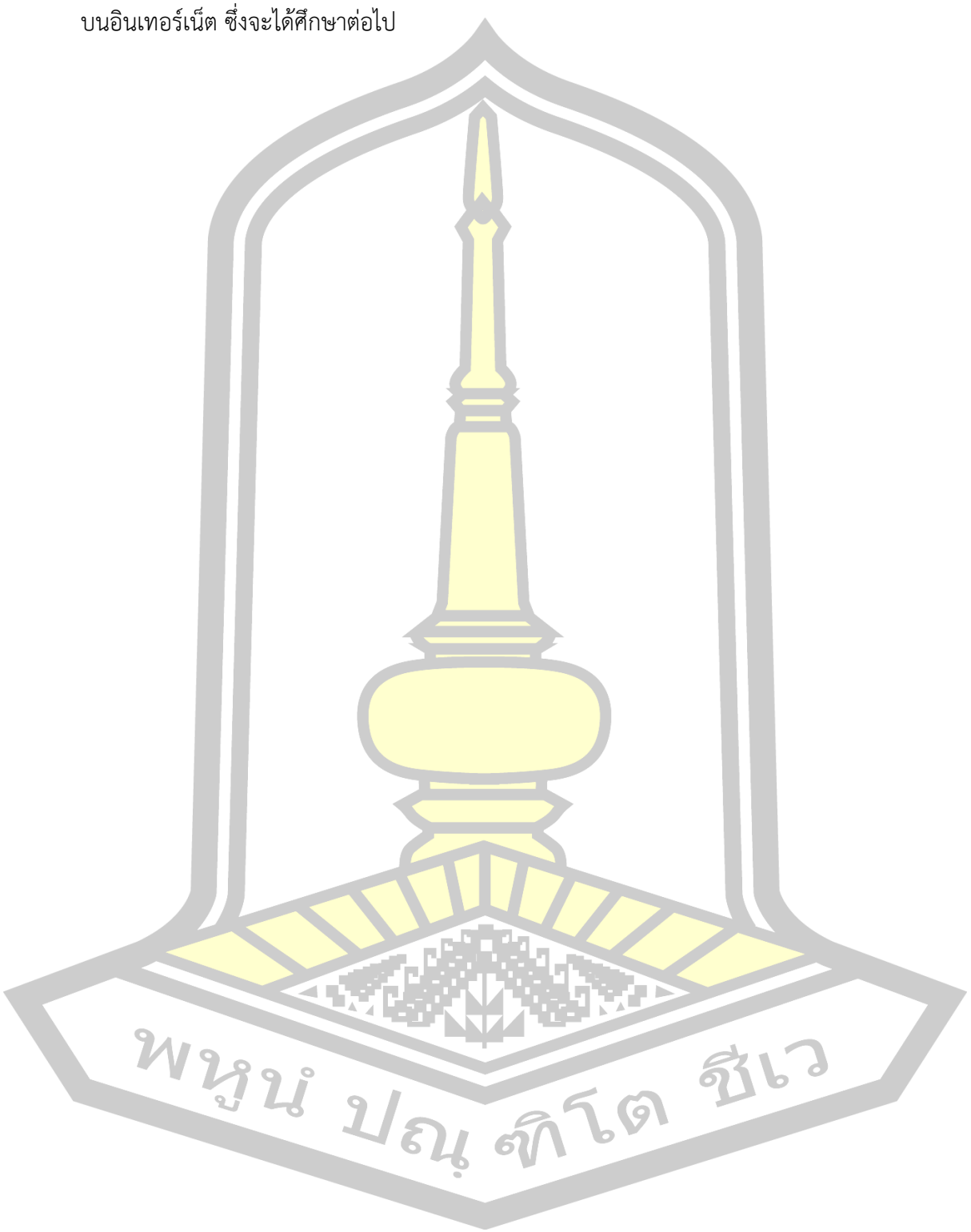
5.2.3 การนำไปใช้ประโยชน์จริง

งานวิจัยนี้ได้ร่วมมือกับหน่วยงานภาครัฐจากกรมสอบสวนคดีพิเศษ ในการแก้ปัญหาป้องกันการก่ออาชญากรรมทางเทคโนโลยีเกี่ยวกับเทคนิคการเปลือยเอสเอสแอล และได้ถูกนำไปปรับใช้จริงสำหรับเว็บที่ต้องการความมั่นคงสูง ในการป้องกันการถูกโจมตีจากเทคนิค SSL Stripping Attack ซึ่งได้เปิดเผยการ Configuration ตั้งค่ากลไก HSTS แบบ Preload ที่สามารถป้องกันการถูกโจมตีดังกล่าวได้ และได้พัฒนาระบบการป้องกันต้นแบบ (Prototype) ในการรักษาข้อมูลระหว่างการสื่อโดยการปรับใช้ Salted Hash Password (SHP) ร่วมกับ Mobile OTP กรณีที่ HTTPS ถูกทำลายหรือไม่ได้รับการป้องกันจากกลไกมาตรฐาน แต่หากมีการปรับใช้กลไกดังกล่าว ถึงแม้ผู้โจมตีดักจับข้อมูลได้ ก็จะอยู่ในรูปแบบค่า Hash ที่ไม่สามารถนำไปใช้ประโยชน์ได้

5.3 แนวทางการวิจัยต่อในอนาคต

แม้ว่างานวิจัยนี้จะประสบผลสำเร็จในการแก้ปัญหาป้องกันการถูกดักจับข้อมูลที่ถูกโจมตีด้วยวิธี SSL Stripping Attack แต่ก็ยังมีบางประเด็นถึงแม้จะไม่ได้อยู่ในขอบเขตของวิทยานิพนธ์นี้ แต่ก็ยังสามารถดำเนินการวิจัยต่อได้ในอนาคตในเรื่องเกี่ยวกับมาตรฐานความมั่นคงต่าง ๆ ไม่ว่าจะเป็น PKI, TLS, HPKP ล้วนแล้วแต่มีปัญหาด้านความมั่นคงทั้งสิ้น หรือแม้กระทั่ง CA ก็ยังเกิดปัญหา

ถูกจู่โจมหรือสร้างปัญหาขึ้นเสียเอง (Trust Turn Evil) ส่งผลทำให้ไม่มีความปลอดภัยในการสื่อสารบนอินเทอร์เน็ต ซึ่งจะได้ศึกษาต่อไป



บรรณานุกรม

[1] Rescorla E. HTTP Over TLS. IETF, RFC 2818, May 2000.

[2] Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2. IETF, RFC 5246, August 2008.

[3] Adams C, Lloyd S. "Understanding PKI: concepts, standards, and deployment considerations". Addison Wesley Professional. 2003; pp. 11-15.

[4] Marlinspike M. New Tricks For Defeating SSL In Practice. [online]. 2009 [cited 14 September 2019]; https://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf.

[5] Fung A, Cheung K. SSLock: sustaining the trust on entities brought by SSL. Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security; 13-16 April 2010; Beijing, China. pp. 204-213.

[6] Nikiforakis N, Younan Y, Joosen W. HProxy: client-side detection of SSL stripping attacks. Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment; 8-9 July 2010; Bonn, Germany. pp. 200-218.

[7] Fung A, Cheung K. HTTPSLock: Enforcing HTTPS in Unmodified Browsers with Cached Javascript. Proceedings of the 4th Network and System Security; 1-3 September 2010; Melbourne, Australia. pp. 269-274.

[8] Puangpronpitag S, Sriwiboon N. Proceedings of 4th International Conference on Computational Intelligence, Communication Systems and Networks. 24-27 July 2012.

[9] Tooltham A, Puangpronpitag S. Click2Enforce: a Browser Extension to Protect against SSL Stripping Attacks. 2, July - December 2013.

[10] Hodges J, Jackson C, Barth A. HTTP Strict Transport Security (HSTS). IETF, RFC 6797, November 2012.

[11] Puangpronpitag S. Information Security and Advanced Networks (ISAN). Faculty of Informatics, Mahasarakham University. [online]. 2019 [https://isanmsu.com.

[12] Dierks T, Allen C. The TLS Protocol Version 1.0. IETF, RFC 2246, January 1999.

[13] Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. IETF, RFC 8446, August 2018.

[14] Introduction to SSL. [online]. September 1998 [cited 14 September 2019]; http://docs.sun.com/source/816-6156-10/contents.htm.

[15] SSL/TLS Strong Encryption: An Introduction. [online]. 2012 [cited 20 September 2019]; http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html.

[16] Standard Specifications For Public-Key Cryptography. [online]. October 2008 [cited 17 September 2019]; https://standards.ieee.org/standard/1363-2000.html.

[17] Secret Key Encryption. [online]. 2013 [cited 25 September 2019]; https://pynacl.readthedocs.io/en/stable/secret/.

[18] PKI System. [ออนไลน์].สืบค้นเมื่อ 16 กันยายน 2562]; http://www.cipher.risk.tsukuba.ac.jp/wordpress/wp-content/uploads/2012/02/pki_e.png.

[19] Keizer G. Solo Iranian hacker takes credit for Comodo certificate attack. [ออนไลน์].สืบค้นเมื่อ 15 กันยายน 2562]; https://www.computerworld.com/article/2507258/solo-iranian-hacker-takes-credit-for-comodo-certificate-attack.html.

[20] DigiNotar SSL certificate hack amounts to cyberwar. [ออนไลน์].สืบค้นเมื่อ 15 กันยายน 2562]; https://www.theguardian.com/technology/2011/sep/05/diginotar-certificate-hack-cyberwar.

[21] Browser Support HSTS. [online]. 1 June 2020 [cited 14 June 2020]; https://caniuse.com/#search=hsts.

[22] HTTP Strict Transport Security. [online]. 2019 [cited 17 September 2019]; https://www.acunetix.com/wp-content/uploads/2019/05/hsts.png.

[23] Burkholder P. SSL Man-in-the-Middle Attacks. [online]. 2002 [cited 25 September 2019]; https://www.sans.org/reading-room/whitepapers/threats/ssl-man-in-the-middle-attacks-480.

[24] Burkholder P. Man-in-the-middle attack. [online]. 2019 [cited 25 September 2019]; https://en.wikipedia.org/wiki/Man-in-the-middle_attack.

[25] Driftnet. [computer program]. Version 1.1.5. doko@debian.org; 2015.

[26] Dsniff. [computer program]. Version 2.3. monkey.org; December 2000.

[27] Bettercap. [computer program]. Version 2.26.1. bettercap.org; 2019.

[28] Ettercap. [computer program]. Version 0.8.3. Alberto Ornaghi; July 2019.

[29] Wireshark. [computer program]. Version 3.0.6. Wireshark Foundation; April 2016.

[30] TCPDump. [computer program]. Version 4.9.2.5072. Microolap Technologies LTD; June 2019.

[31] สมนึก พ่วงพรพิทักษ์, ณัฐวุฒิ ศรีวิบูลย์, อภิรักษ์ ทูลธรรม. การประเมินปัญหาและพัฒนาวิธีแก้ไขปัญหาการโจมตีระบบเว็บไซต์ที่ให้บริการธนาคารทางอินเทอร์เน็ต. ได้รับการสนับสนุนการวิจัยงบประมาณแผ่นดิน: มหาวิทยาลัยมหาสารคาม; 2557.

[32] Steube J, Gristina G. Hashcat. [online]. 2020 [cited 14 February 2021]; https://hashcat.net/hashcat/.

[33] transport_security_state_static. [online]. 2019 [cited 4 November 2019]; https://cs.chromium.org/chromium/src/net/http/transport_security_state_static.json.

[34] Fairweather D, Mozer H, Rinehart S, Shin D. An enhanced approach to preventing the SSLstripping attack. 2015 International Conference on Information and Communication Technology Convergence (ICTC); Jeju 2015; pp. 339-343.

[35] Selvi J. Bypassing HTTP Strict Transport Security. Proceedings of Black Hat Europe 2014; 14-17 October 2014;

[36] Puangpronpitag S, Masusai N. Sample of Conference Proceedings. Proceedings of the 6th International Conference on Electrical Engineering/ Electronics, Computer, Telecommunications, and Information Technology (ECTI-CON); 6-9 May 2009; Pattaya, Chonburi, Thailand. pp. 910-913.

[37] สมนึก พ่วงพรพิทักษ์, อภิรักษ์ ทูลธรรม. การประเมินวิธีแก้ไขปัญหาการโจมตีด้วยการเปลี่ยนเอสเอสแอล. วารสารเทคโนโลยีสารสนเทศ มกราคม - มิถุนายน 2557; 10[1]: 37-47.

[38] Sriwiboon N, Puangpronpitag S. Detection & Protection Mechanisms Against SSL Strip Attacks. Proceedings of 9th International Joint Conference on Computer Science and Software Engineering; 30 May-1 June 2012; Bangkok, Thailand. pp. 25-30.

[39] Bank of Thailand. Internet Banking. [online].Cited 18 November 2019]; https://www.bot.or.th/Thai/Pages/default.aspx.

[40] Google Chrome. [computer program]. Version 79. Google Incorporation; 2020.

[41] Microsoft Edge. [computer program]. Version 85. Microsoft; 2020.

[42] Mozilla Firefox. [computer program]. Version 72. mozilla.org; 2020.

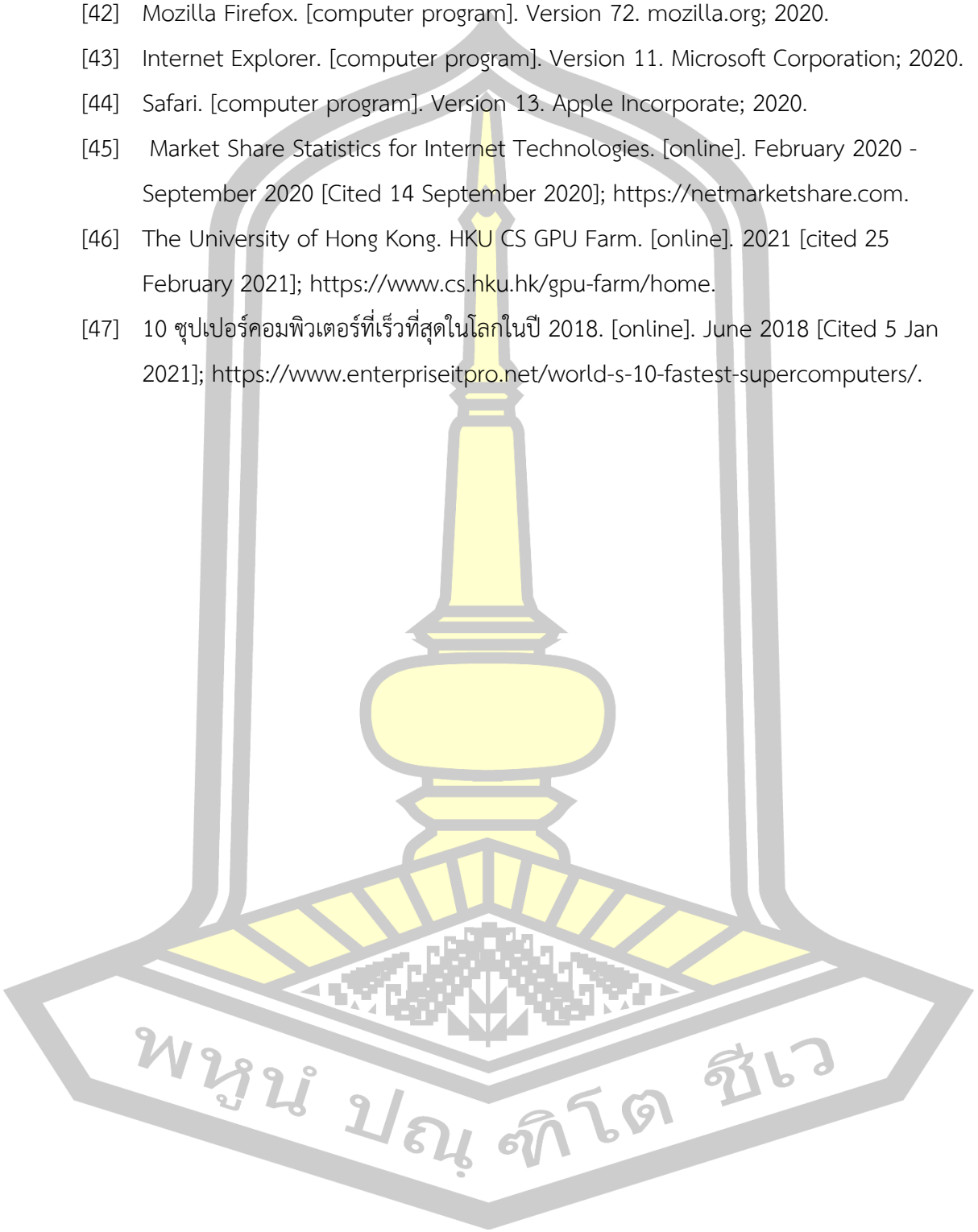
[43] Internet Explorer. [computer program]. Version 11. Microsoft Corporation; 2020.

[44] Safari. [computer program]. Version 13. Apple Incorporate; 2020.

[45] Market Share Statistics for Internet Technologies. [online]. February 2020 - September 2020 [Cited 14 September 2020]; https://netmarketshare.com.

[46] The University of Hong Kong. HKU CS GPU Farm. [online]. 2021 [cited 25 February 2021]; https://www.cs.hku.hk/gpu-farm/home.

[47] 10 ซุปเปอร์คอมพิวเตอร์ที่เร็วที่สุดในโลกในปี 2018. [online]. June 2018 [Cited 5 Jan 2021]; https://www.enterpriseitpro.net/world-s-10-fastest-supercomputers/.



ประวัติผู้เขียน

ชื่อ	นายภารเดช คะเชนรัมย์
วันเกิด	วันที่ 12 พฤศจิกายน พ.ศ. 2539
สถานที่เกิด	อำเภอขุขันธ์ จังหวัดศรีสะเกษ
สถานที่อยู่ปัจจุบัน	77 หมู่ 17 ต.ปราสาท อ.บ้านด่าน จ.บุรีรัมย์ 31000
ตำแหน่งหน้าที่การงาน	ผู้อำนวยการสถานีวิทยุ
สถานที่ทำงานปัจจุบัน	สถานีวิทยุบุรีรัมย์เรดิโอ คลื่นความถี่ FM 103 MHz อ.บ้านด่าน จ.บุรีรัมย์
ประวัติการศึกษา	พ.ศ. 2552 ประถมศึกษาปีที่ 6 โรงเรียนวัดบ้านโคกเหล็ก จังหวัดบุรีรัมย์ พ.ศ. 2555 มัธยมศึกษาปีที่ 3 โรงเรียนอิสาณโกศลศึกษา จังหวัดบุรีรัมย์ พ.ศ 2558 มัธยมศึกษาปีที่ 6 โรงเรียนห้วยราชพิทยาคม จังหวัดบุรีรัมย์ พ.ศ. 2562 ปริญญาวิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยราชภัฏบุรีรัมย์ พ.ศ. 2564 ปริญญาวิทยาศาสตรมหาบัณฑิต (วท.ม) สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยมหาสารคาม
ทุนวิจัย	พ.ศ. 2564 ทุนสนับสนุนการวิจัยจากกรมสอบสวนคดีพิเศษ กระทรวงยุติธรรม
ผลงานวิจัย	พ.ศ. 2564 การวิเคราะห์ปัญหาการทำงานผิดพลาดของกลไกเอชเอสทีเอสและการจู่โจมด้วยการเปลือยเอสเอสแอล