วิธีป้องกันการโจมตีจากคนคั่นกลางของการเข้ารหัสลับ

ENCRYPTION MAN-IN-THE-MIDDLE ATTACK AVOIDANCE

ชนาธิป แก้วประเสริฐศรี
รณวิทย์ ทิพย์โภชนา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2559

วิธีป้องกันการโจมตีจากคนคั่นกลางของการเข้ารหัสลับ

ENCRYPTION MAN-IN-THE-MIDDLE ATTACK AVOIDANCE

ชนาธิป แก้วประเสริฐศรี
รณวิทย์ ทิพย์โภชนา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2559

ปริญญานิพนธ์ปีการศึกษา 2559

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง วิธีป้องกันการโจมตีจากคนคั่นกลางของการเข้ารหัสลับ

ENCRYPTION MAN-IN-THE-MIDDLE ATTACK AVOIDANCE

ผู้จัดทำ

1. นายชนาธิป แก้วประเสริฐศรี รหัสนักศึกษา 56010251

2. นายรณวิทย์ ทิพย์โภชนา รหัสนักศึกษา 56010996

_____อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ อัครเดช วัชระภูพงษ์)

วิธีป้องกันการโจมตีจากคนคั่นกลางของการเข้ารหัสลับ

นายชนาธิป แก้วประเสริฐศรี 56010251
นายรณวิทย์ ทิพย์โภชนา 56010996
ผศ.อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2559

บทคัดย่อ

เนื่องจากการใช้งานอินเทอร์เน็ตในชีวิตประจำวัน มีการส่งข้อมูลที่สำคัญซึ่งเป็นความลับอยู่บ่อยครั้ง เช่น การทำธุรกรรมทางอิเล็กทรอนิกส์, พาณิชยกรรมอิเล็กทรอนิกส์, สื่อสังคมต่าง ๆ เป็นต้น จึงเป็นสาเหตุทำให้ผู้ที่ไม่ประสงค์ดี อาศัยช่องทางต่าง ๆ ที่ไม่ปลอดภัย เพื่อดักฟัง เปลี่ยนแปลงข้อมูล หรือปลอมแปลงข้อมูลเพื่อหลอกลวงผู้ใช้งาน ซึ่งอาจก่อให้เกิดความเสียหายในหลาย ๆ ด้าน

การโจมตีจากคนคั่นกลางของการเข้ารหัสลับ เป็นหนึ่งในปัญหาทางด้านความปลอดภัยที่น่าจับตามองมากที่สุด ซึ่งเป็นการโจมตีที่อาศัยจุดอ่อนของโพรโทคอล ต่าง ๆ ที่ใช้ในการยืนยันตัวตนและรับส่งข้อมูลผ่านเครือข่าย เพื่อทำหน้าที่เป็นตัวกลางในการสนทนาระหว่างสองฝ่าย โดยที่ไม่รู้ตัว

โครงงานนี้ ได้ศึกษาการโจมตีจากคนคั่นกลางของการเข้ารหัสลับหลายรูปแบบ และจัดทำแนวทางปฏิบัติเพื่อให้ความรู้ ความเข้าใจ ในการป้องกันการรั่วไหลของข้อมูล รวมถึงกระตุ้นให้ผู้ใช้งาน มีความรอบคอบและตระหนักถึงความปลอดภัยในการรับส่งข้อมูลผ่านทางเครือข่าย

Encryption Man-In-The-Middle Attack Avoidance

Mr. Chanathip Kaewprasertsri 56010251
Mr. Ronnawit Thippochana 56010996
Asst. Prof. Akkradach Watcharapupong Advisor
Academic Year 2016

ABSTRACT

According to daily internet usage, there is a necessity to transfer secret information often, such as e-banking, e-commerce, social media, etc. These cause malefactors use any vulnerability to eavesdrop, tampering or forgery the information.

Encryption man-in-the-middle attack is one of the most spectacular security threat, which uses any vulnerability of authentication protocols, and communicate protocols to be a middleman in communication between two parties silently.

This project, we research about various type of man-in-the-middle attack, make a system for prevention and write a guideline for introduce how to prevent and avoid leaking of information along with motivating users to be aware of security in transferring information over network.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ประสบความสำเร็จลุล่วงได้ด้วยดี เพราะได้รับการช่วยเหลือจากผู้มีพระคุณหลาย ๆ ท่าน

คณะผู้จัดทำขอขอบคุณอาจารย์อัครเดช วัชระภูพงษ์ ที่คอยให้การช่วยเหลือและคำแนะนำที่ทำให้โครงงานนี้สามารถดำเนินงานต่อไปได้อย่างมีประสิทธิภาพ

ขอขอบคุณรุ่นพี่ เพื่อน และรุ่นน้องประจำห้องวิจัยและพัฒนาความปลอดภัยของข้อมูล Information Security Advisory Group ที่เอื้อเฟื้อสถานที่ในการศึกษา และให้คำปรึกษาในการทำปริญญานิพนธ์

และสุดท้ายนี้ ขอกราบขอบพระคุณบิดามารดาที่คอยส่งเสริมและสนับสนุนในด้านการเรียนและการทำงานของคณะผู้จัดทำมาโดยตลอด

ชนาธิป แก้วประเสริฐศรี

รณวิทย์ ทิพย์โภชนา

สารบัญ

หน้า

สารบัญตาราง

ตาราง **หน้า**

สารบัญรูป

รูป หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

เนื่องจากในปัจจุบัน อินเทอร์เน็ตได้เข้ามาเป็นส่วนหนึ่งของชีวิตประจำวัน ทำให้การสื่อสารมีความสะดวกรวดเร็วกว่าในอดีต แต่ในขณะเดียวกัน ผู้ใช้งานหลาย ๆ คน ไม่ได้ตระหนักถึงภัยคุกคามด้านความปลอดภัยเท่าใดนัก โดยเฉพาะอย่างยิ่ง การให้ข้อมูลส่วนตัวหรือข้อมูลที่เป็นความลับกับผู้อื่น ซึ่งเป็นสาเหตให้เกิดการโจรกรรมข้อมูลทางอินเทอร์เน็ตเกิดขึ้นอยู่บ่อยครั้ง ทางผู้จัดทำได้เล็งเห็นถึงความสำคัญที่จะทำให้ผู้ใช้งานมีความระมัดระวังในการใช้งานอินเทอร์เน็ต และมีความรู้ในการป้องกันการรั่วไหลของข้อมูล

1.2 วัตถุประสงค์

1) เพื่อตระหนักถึงความอันตรายของการโจมตีจากคนคั่นกลางของการเข้ารหัสลับ
2) เพื่อทำให้ผู้ใช้งานมีความระมัดระวังและมีความรอบคอบในการใช้งานอินเทอร์เน็ต
3) เพื่อจัดทำแนวทางปฏิบัติในการหลีกเลี่ยงและป้องกันการโจมตีจากคนคั่นกลาง
4) จัดทำระบบที่สามารถป้องกันการโจมตี

1.3 ขอบเขตของโครงงาน

ศึกษาข้อมูลที่เกี่ยวข้อง เพื่อจัดทำเป็นระบบที่สามารถตรวจสอบและป้องกันการโจมตีจากคนคั่นกลางของการเข้ารหัสลับได้

1.4 วิธีการดำเนินงาน

1) ศึกษาทฤษฎีและหลักการที่เกี่ยวข้อง
2) ศึกษาการโจมตีจากคนคั่นกลางรูปแบบต่าง ๆ
3) วางแผนการทดลอง
4) ดำเนินการทดลอง
5) วิเคราะห์ผลการทดลอง
6) จัดทำแนวทางปฏิบัติในการหลีกเลี่ยงและป้องกันการโจมตี

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้และเข้าใจการโจมตีจากคนคั่นกลาง

2) นำความรู้ที่ได้ไปประยุกต์ใช้ในชีวิตประจำวัน เพื่อปกป้องข้อมูลส่วนตัวต่าง ๆ

3) คู่มือแนวทางปฏิบัติในการหลีกเลี่ยงและป้องกันการโจมตีจากคนคั่นกลาง

4) ระบบสำหรับป้องกันการโจมตีจากคนคั่นกลางของการเข้ารหัสลับ

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 โพรโทคอลที่เกี่ยวข้อง

2.1.1 Internet Protocol (IP)

เป็นโพรโทคอลการสื่อสารที่สำคัญใน Internet protocol suite สำหรับถ่ายทอดดาต้าแกรม (หน่วยข้อมูลพื้นฐานของแพ็กเกต ซึ่งการส่ง, เวลาที่ถึงและลำดับที่ถึง ไม่ถูกรับประกันโดยเครือข่าย) ข้ามเขตแดนเครือข่าย ฟังก์ชันการกำหนดเส้นทางของมันจะช่วยงานภายในเครือข่ายและก่อตั้งระบบอินเทอร์เน็ตขึ้น การทำงานของไอพีเป็นการทำงานแบบไม่รับประกันความถูกต้องของข้อมูล ซึ่งรุ่นปัจจุบันคือ IPv4 และกำลังอยู่ในช่วงผลักดันให้ใช้ IPv6

2.1.2 Transmission Control Protocol (TCP)

เป็นหนึ่งในโพรโทคอลหลักในเครือข่ายอินเทอร์เน็ต หน้าที่หลักของ TCP คือ ควบคุมการรับส่งข้อมูลระหว่างแม่ข่ายถึงแม่ข่ายในเครือข่าย เพื่อใช้แลกเปลี่ยนข้อมูลระหว่างกัน โดยตัวโพรโทคอลจะรับประกันความถูกต้อง และลำดับของข้อมูลที่ส่งผ่านระบบเครือข่าย นอกจากนี้ยังช่วยจำแนกข้อมูลให้ส่งผ่านไปยังแอปพลิเคชัน ที่ทำงานอยู่บนแม่ข่ายเดียวกันให้ถูกต้องด้วย

หน้าที่หลักที่สำคัญของ TCP อีกหน้าที่หนึ่งคือ เป็นโพรโทคอลที่ขั้นกลางระหว่างแอปพลิเคชันและเครือข่ายไอพี ทำให้แอปพลิเคชันจากแม่ข่ายหนึ่ง สามารถส่งข้อมูลออกยังอีกแม่ข่ายหนึ่งผ่านเครือข่ายเปรียบเสมือนมีท่อส่งข้อมูลระหว่างกัน

TCP เป็นโพรโทคอลที่ได้รับความนิยมที่สุดในโลกของอินเทอร์เน็ต มีแอปพลิเคชันจำนวนมากที่ใช้โพรโทคอลทีซีพีเป็นสื่อกลางในการเชื่อมต่อ เช่น เวิลด์ไวด์เว็บ เป็นต้น

2.1.3 Secure Socket Layer (SSL)

SSL เป็นโพรโทคอลเพื่อการสื่อสารที่มีความปลอดภัยสำหรับการใช้งานอินเทอร์เน็ต ซึ่งถูกใช้งานอย่างแพร่หลาย เนื่องจากช่วยให้การรับส่งข้อมูลผ่าน TCP มีความปลอดภัยมากขึ้น

SSL เวอร์ชั่น 1.0 ถูกพัฒนาโดย Netscape ในปีค.ศ. 1994 และ SSL 2.0 ถูกพัฒนาหลังจากนั้นประมาณ 1 ปีถัดมา หลังจากนั้น SSL 3.0 ถูกเผยแพร่ในปีค.ศ. 1999 และ Netscape ได้ให้ IETF รับช่วงต่อในการพัฒนาเวอร์ชั่นต่อ ๆ ไป ซึ่งได้มีการเปลี่ยน เป็น Transport Layer Security (TLS)

โพรโทคอล SSL แบ่งออกเป็น 4 ชั้นเลเยอร์ ประกอบไปด้วย Record Layer, ChangeCipherSpec Protocol, Alert Protocol และ Handshake Protocol ซึ่งทั้งหมดจะถูกเอนแคปซูเลตเมื่อมีการสื่อสารกันระหว่างไคลเอนต์ กับ เซิร์ฟเวอร์ โดยแต่ละเลเยอร์มีรายละเอียดดังนี้

2.1.3.1 Record Layer

เป็นเลเยอร์ที่จัดหารูปแบบของเฮดเดอร์ของแต่ละข้อความ และทำการแฮชด้วย MAC ที่สร้างขึ้นมา เฮดเดอร์จะมีขนาด 5ไบต์ ประกอบไปด้วย Protocol Definition 1ไบต์ Protocol Version 2 ไบต์ และ Length 2 ไบต์ ซึ่งข้อความที่นำมาจะต้องมีขนาดไม่เกิน 16,384 ไบต์ ตามที่ SSL protocol กำหนดไว้

2.1.3.2 ChangeCipherSpec Protocol

จะสร้างสัญญาณขึ้นเมื่อมีการเริ่มติดต่อระหว่างไคลเอนท์กับเซิร์ฟเวอร์เพื่อความปลอดภัย โดยสัญญาณดังกล่าวจะมีขนาด 1 ไบต์

2.1.3.3 Alert Protocol

เป็นโพรโทคอลที่จะคอยส่งข้อผิดพลาด ปัญหา หรือแจ้งเตือน เกี่ยวกับการติดต่อระหว่างทั้งสองฝ่าย โดยจะมีรูปแบบอยู่ 2รูปแบบ คือ Severity Level และ Alert Description

1) Severity Level คือ การส่งข้อความด้วยค่า 1 หรือ 2 ขึ้นอยู่กับระดับความรุนแรงของปัญหา ค่า 1 จะเป็นข้อความเตือนหรือให้ระมัดระวัง โดยจะแนะนำให้ผู้ที่ติดต่อกันมีการยุติและเริ่มติดต่อกันใหม่ ค่า 2 จะเป็นข้อความแจ้งเตือนที่มีความรุนแรง/เสี่ยงสูง โดยจะให้ผู้ติดต่อยุติการติดต่อ

2) Alert Description คือ การแสดงการแจ้งเตือนข้อที่ผิดพลาดที่เจาะจง มีขนาด 1 ไบต์ซึ่งจะมีการจับคู่กับตัวเลข 12 ตัวเลขที่เจาะจงเพื่อบ่งบอกถึงความหมายของข้อผิดพลาด ประกอบด้วย

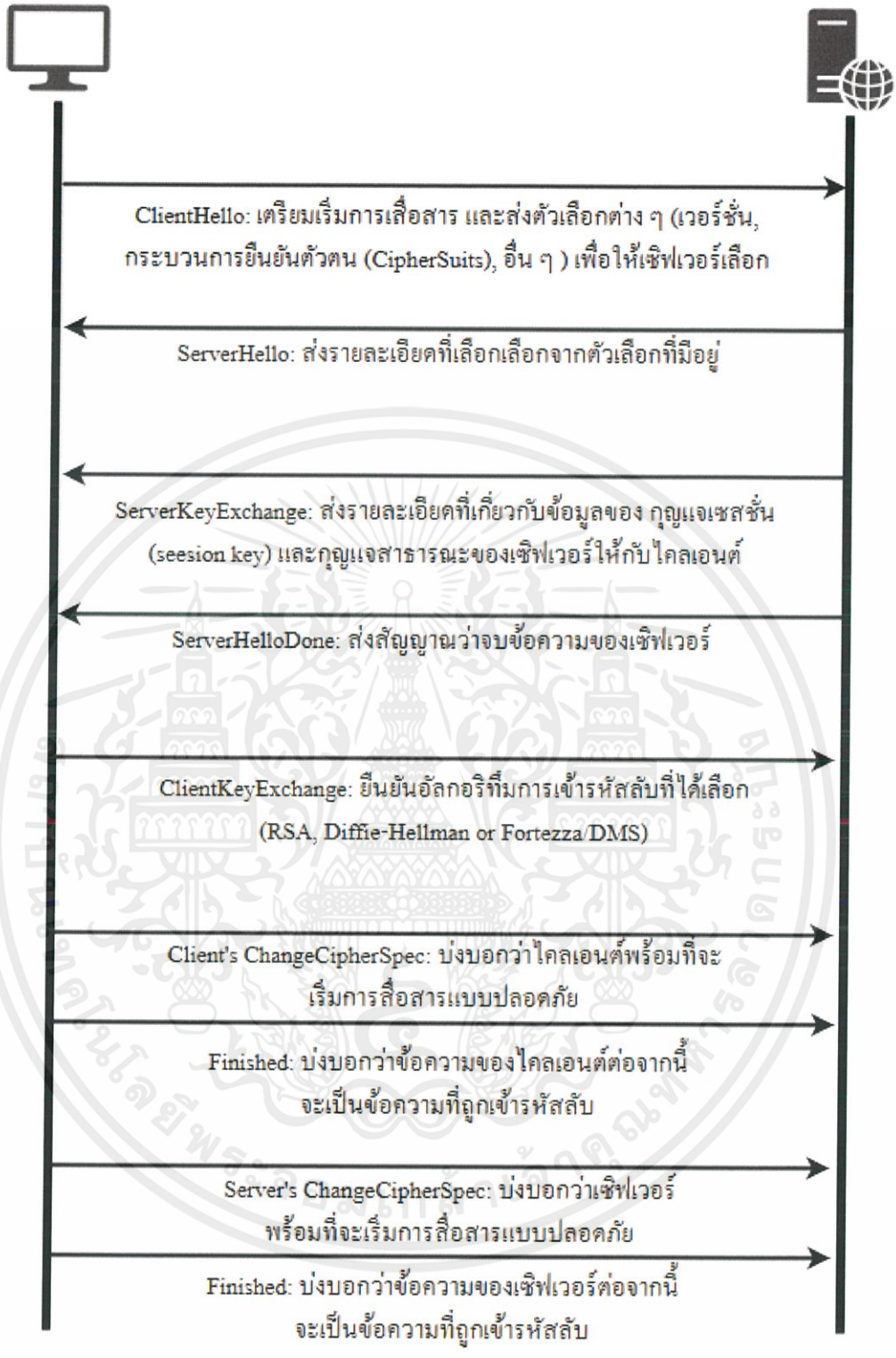
CloseNotify	**HandshakeFailure**	CertificateRevoked
UnexpectedMessage	NoCertificate	CertificateExpired
BadRecordMAC	BadCertificate	CertificateUnknown
DecompressionFailure	UnsupportedCertificate	IllegalParameter

โดยข้อความที่พิมพ์ตัวหนาคือปัญหาที่มีความรุนแรงสูง

2.1.3.4 Handshake Protocol

ข้อความจะถูกส่งไปมาระหว่างไคลเอนต์และเซิร์ฟเวอร์ทั้งหมด 4 ครั้ง และทำการสร้างการแฮนด์เชค (Handshake) เพื่อความปลอดภัยในการติดต่อสื่อสาร ซึ่งการทำ SSL handshake จะประกอบไปด้วย

1) ClientHello คือ ข้อความแรกของการเริ่มต้นจากไคลเอนต์เพื่อทำการขอการติดต่อสื่อสารที่มีความปลอดภัยในช่วงเวลาหนึ่ง ข้อความนี้ประกอบไปด้วยชุดตัวเลือกที่ไคลเอนต์จะใช้ในการติดต่อกับเซิร์ฟเวอร์ ประกอบไปด้วยเวอร์ชั่นของ SSL ที่จะใช้งาน, กระบวนการยืนยันตัวตน (CipherSuit) ที่ไคลเอนต์รองรับ และวิธีการบีบอัด (CompressionMethods) ที่ถูกใช้โดยไคลเอนต์ ส่วนข้อมูลอื่นที่อยู่ในข้อความนี้คือตัวเลขที่สุ่มขึ้นมา 32 ไบต์ เพื่อช่วยในการเข้ารหัสลับในการติดต่อสื่อสารและเพิ่มเติมในส่วนของ SessionID ที่ว่างเปล่า
2) ServerHello คือ ในส่วนของข้อความที่สอง เซิร์ฟเวอร์จะทำการเลือกข้อความจาก ClientHello และตอบกลับ โดยจะยืนยันเวอร์ชั่นของ SSL ที่จะใช้งาน, วิธีการบีบอัด, กระบวนการยืนยันตัวตน และทำการแทนที่ตัวเลขที่ถูกสุ่มขึ้นมา 4 ไบต์นั้นด้วยวันและเวลาเพื่อหลีกเลี่ยงการซ้ำซ้อนของค่าที่สุ่มได้
3) ServerKeyExchange คือ การที่เซิร์ฟเวอร์เลือกวิธีการเข้ารหัสลับเพื่อที่จะส่งข้อมูลกุญแจสาธารณะของเซิร์ฟเวอร์ที่ใช้ในการเข้ารหัสลับจะมีการแบ่งช่วงเวลาเพื่อคงความปลอดภัยในการติดต่อ และในการติดต่อทั้งเซิร์ฟเวอร์และไคลเอนต์จะต้องใช้กุญแจเดียวกันในการเข้ารหัสลับ เพื่อความมั่นใจของทุกฝ่ายจึงมีการใช้ใบรับรองอิเล็กทรอนิกส์ในการยืนยันตัวตน
4) ServerHelloDone คือ เมื่อเซิร์ฟเวอร์ทำการ ServerKeyExchange แล้ว ไคลเอนต์จะได้รับข้อความนี้เพื่อบอกว่าเซิร์ฟเวอร์นั้นพร้อมแล้ว
5) ClientKeyExchange คือ ส่วนที่ประกอบไปด้วยข้อมูลเกี่ยวกับกุญแจที่ไคลเอนต์และเซิร์ฟเวอร์จะใช้ในการติดต่อสื่อสารกัน
6) ChangeCipherSpec คือ การส่งสัญญาณเพื่อเปลี่ยนสถานะการส่งข้อมูลจากสถานะไม่ปลอดภัยไปสู่สถานะปลอดภัย
7) Finished คือ เมื่อได้รับสัญญาณจากข้อความสุดท้ายแล้ว จะทำการตรวจสอบอีกครั้ง ประกอบไปด้วยข้อมูลของกุญแจ, เนื้อหาการทำแฮนด์เชคของ SSL ก่อนหน้านี้ทั้งหมด, ค่าพิเศษที่บอกว่าผู้ส่งคือไคลเอนต์หรือเซิร์ฟเวอร์



รูป 2.1 SSL Handshake Protocol

เมื่อทำการแฮนด์เชค (Handshake) สำเร็จผู้ใช้งานจะเห็นสัญลักษณ์ แม่กุญแจ ที่มุมของบราวเซอร์ ซึ่งจะเป็นการบ่งบอกถึงความปลอดภัยที่ได้รับการยืนยันแล้ว

2.1.4 Transport Layer Security (TLS)

TLS ถูกเผยแพร่ในเดือนมกราคม ซึ่งพัฒนาต่อยอดมาจาก SSL ค.ศ. 1999 เพื่อเป็นมาตรฐานสำหรับการสื่อสารที่มีความปลอดภัย ซึ่งช่วยให้ไคลเอนต์หรือเซิร์ฟเวอร์สื่อสารผ่านช่องทางที่ออกแบบมาเพื่อป้องการการดักฟัง, การเปลี่ยนแปลง หรือ การปลอมข้อมูล เป้าหมายของ TLS คือ การรักษาความปลอดภัยของการเข้ารหัสลับ การทำงานร่วมกัน การขยาย และมีประสิทธิภาพ โดยจะมีการนำ TLS protocol มาปรับใช้ใน 2 ระดับ TLS Record protocol และ TLS Handshake protocol

TLS Record Protocol คือ โพรโทคอลที่ว่าด้วยเรื่องของความเป็นส่วนตัว และความน่าเชื่อถือในการติดต่อระหว่างไคลเอนท์กับเซิร์ฟเวอร์ โดยจะไม่ใช้การเข้ารหัสลับ แต่จะใช้ Symmetric Cryptography key เพื่อยืนยันความเป็นส่วนตัวของการติดต่อแทน

TLS Handshake Protocol คือ โพรโทคอลที่ช่วยในการรับรองความถูกต้องในการสื่อสารที่จะเริ่มต้นระหว่างไคลเอนท์กับเซิร์ฟเวอร์ โดยจะให้ทั้งสองฝ่ายยอมรับ encryption algorithm และ encryption keys ก่อนที่จะเริ่มการส่งข้อมูล และใช้วิธีการทำ handshake เหมือนกับ SSL

2.1.5 Hypertext Transfer Protocol (HTTP)

โพรโทคอลสื่อสารสำหรับแลกเปลี่ยนข้อมูลกันระหว่างเว็บเซิร์ฟเวอร์ และ เว็บไคลเอนต์ ตามปกติทำงานที่พอร์ต 80 โดยส่งข้อมูลแบบ Clear text คือ ข้อมูลที่ทำการส่งไปนั้นไม่ได้ทำการเข้ารหัสลับ ทำให้สามารถถูกดักจับและอ่านข้อมูลได้ง่าย

2.1.6 Secure Hypertext Transfer Protocol (HTTPS)

บางครั้งถูกเรียกว่า HTTP over TLS, HTTP over SSL และ HTTP Secure เป็นโพรโทคอลที่สร้างเพื่อความปลอดภัย ในการสื่อสารผ่านอินเตอร์เน็ตข้อมูลที่ทำการส่งได้ถูกเข้ารหัสลับเอาไว้ โดยใช้ Asymmetric Algorithm ซึ่งถ้าถูกดักจับได้ก็ไม่สามารถที่จะอ่านข้อมูลนั้นได้รู้เรื่อง โดยข้อมูลนั้นจะสามารถอ่านได้เข้าใจเฉพาะClient กับเครื่อง Server เท่านั้น ซึ่งตามปกติแล้วจะทำงานที่พอร์ต 443

2.1.7 Internet Control Message Protocol (ICMP)

เป็นโพรโทคอลที่ใช้ในการตรวจสอบและรายงานสถานภาพของดาต้าแกรม (Datagram) ในกรณีที่เกิดปัญหากับดาต้าแกรม เช่น เราเตอร์ไม่สามารถส่งดาต้าแกรมไปถึงปลายทางได้ ICMP จะถูกส่งออกไปยังเครื่องคอมพิวเตอร์ต้นทางเพื่อรายงานข้อผิดพลาด ที่เกิดขึ้น

2.1.8 Address Resolution Protocol (ARP)

โพรโทคอลที่ใช้ในการสื่อสาร ทำหน้าที่ในการจับคู่ระหว่างไอพีแอดเดรสทางลอจิคัล (Logical Address) กับ แอดเดรสทางทางฟิสิคัล (Physical Address)

2.2 โครงสร้างพื้นฐานกุญแจสาธารณะ (Public Key Infrastructure)

เป็นเทคโนโลยีที่อาศัยระบบการเข้ารหัสลับแบบกุญแจสาธารณะ (Public Key Cryptography) ที่ประกอบด้วยกุญแจส่วนตัว (Private key) และกุญแจสาธารณะ (Public key) ซึ่งโครงสร้างดังกล่าวใช้ในการพิสูจน์ตัวตน (Authentication) รวมทั้งการรักษาความลับของข้อมูล (Data Confidentiality)

2.2.1 ใบรับรองอิเล็กทรอนิกส์ (Digital Certificate)

คือข้อมูลอิเล็กทรอนิกส์ที่ออกโดยผู้ให้บริการออกใบรับรอง (Certification Authority - CA) เพื่อใช้บ่งบอกถึงความมีตัวตนที่แท้จริงในโลกแห่งอิเล็กทรอนิกส์ โดยผู้ให้บริการออกใบรับรองจะทำการรับรองข้อมูลต่าง ๆ ซึ่งรวมถึงกุญแจสาธารณะที่ปรากฏในใบรับรองอิเล็กทรอนิกส์ว่าเป็นของบุคคลนั้นจริง

2.2.2 มาตรฐาน X.509

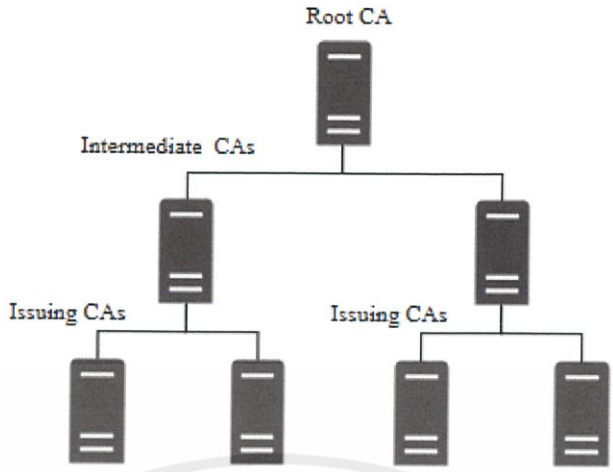
เป็นมาตรฐานสำคัญสำหรับโครงสร้างพื้นฐานกุญแจสาธารณะ เพื่อจัดการใบรับรองอิเล็กทรอนิกส์ และการเข้ารหัสลับกุญแจสาธารณะและกุญแจของโพรโทคอล TLS ซึ่งตามมาตรฐานนี้ บนใบรับรองอิเล็กทรอนิกส์แต่ละใบนั้นจะประกอบไปด้วย

1) หมายเลขของใบรับรอง
2) วิธีการที่ใช้ในการเข้ารหัสลับข้อมูล
3) ชื่อขององค์กรออกใบรับรอง
4) เวลาเริ่มใช้ใบรับรอง และเวลาที่ใบรับรองหมดอายุ
5) ชื่อผู้ถือใบรับรอง และข้อมูลทั่วไป เช่นหน่วยงานที่สังกัด, ที่อยู่อีเมล์ เป็นต้น
6) กุญแจสาธารณะของผู้ถือใบรับรอง
7) ลายมือชื่อดิจิตอลของหน่วยงานที่ออกใบรับรอง

2.2.3 ผู้ให้บริการออกใบรับรองอิเล็กทรอนิกส์ (Certificate Authority)

หน่วยงานที่มีความน่าเชื่อถือและความเชี่ยวชาญในเทคโนโลยีที่เกี่ยวข้อง เพื่อเป็นการรับรองความน่าเชื่อถือของกุญแจสาธารณะ และให้บริการออกใบรับรองให้กับผู้ให้บริการเว็บไซต์ต่าง ๆ ซึ่งมีทั้งแบบเพื่อการค้าและแบบไม่แสวงผลกำไร เช่น CAcert และ Let's Encrypt

ผู้ออกใบรับรอง จะแบ่งออกเป็นลำดับขั้น ละใบรับรองอิเล็กทรอนิกส์ที่ออกโดยผู้ให้บริการ จะถูกตรวจสอบและยืนยันโดยใช้ chain of trust ซึ่งจะเริ่มมาจาก Root Certificate Authority (Root CA), Intermediate Certificate Authority และ Certificate Authority อื่น ๆ ที่อยู่ภายใต้ Root CA ซึ่งลำดับขั้นดังรูป



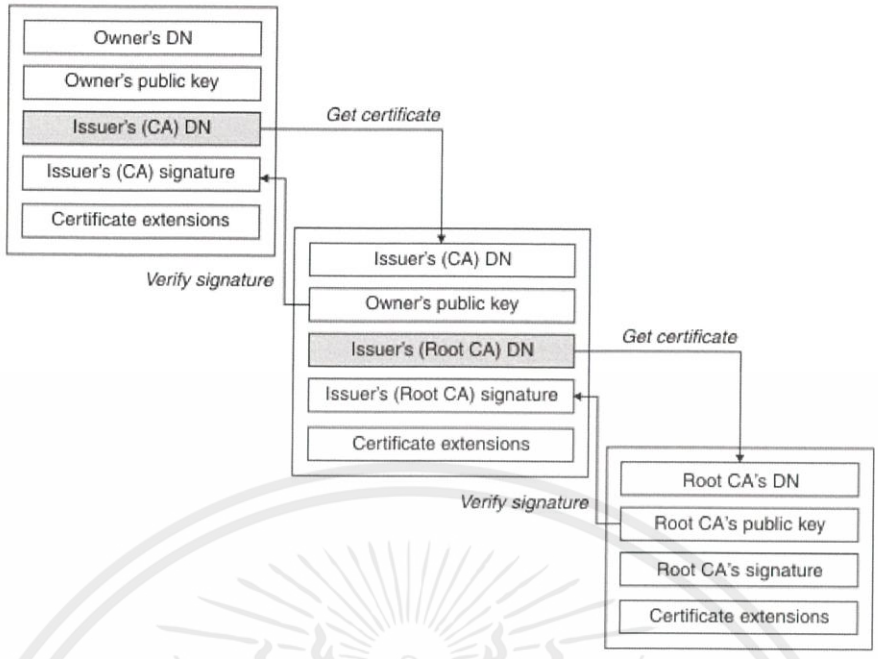
รูป 2.2 ลำดับขั้นของผู้ให้บริการออกใบรับรองอิเล็กทรอนิกส์ (CA)

1) Root CA เป็น CA ที่อยู่ในชั้นที่สูงที่สุด โดย Root CA นั้นจะถูกเก็บไว้ให้อยู่ในที่ที่ปลอดภัยและมักจะไม่เชื่อมต่อกับเครือข่าย ซึ่งถ้า Root CA ถูกผู้บุกรุกเข้ามาได้ ใบรับรองต่าง ๆ อาจถูกยกเลิกการใช้งาน
2) Intermediate CAs หรือบางครั้งเรียกว่า Policy CAs หรือ Subordinate CAs เป็น CA ที่สืบต่อมาจาก CA อื่น ๆ (Root CA หรือ intermediate CA) และทำหน้าที่ออกใบรับรองให้กับ CA อื่น ๆ และมักจะไม่เชื่อมต่อกับเครือข่ายเช่นเดียวกับ root CA
3) Issuing CAs ทำหน้าที่ออกใบรับรองให้กับผู้ใช้งาน, คอมพิวเตอร์ และบริการต่าง ๆ

2.3 กระบวนการตรวจสอบใบรับรอง

2.3.1 ตรวจสอบด้วยลำดับขั้นของใบรับรอง (Chain of trust)

ลำดับขั้นของใบรับรอง (certificate chain หรือ certificate path) เป็นรายการของใบรับรองที่ใช้สำหรับพิสูจน์เอกลักษณ์บุคคล ซึ่งเริ่มด้วยใบรับรองของบุคคลคนนั้น แล้วตามด้วยใบรับรองอื่น ๆ ที่รับรองด้วยบุคคลต่อไปในลำดับขั้นของใบรับรอง ซึ่งจะสิ้นสุดเมื่อถึงใบรับรองของ Root CA ซึ่งเป็น CA ที่รับรองใบรับรองด้วยตัวเอง เมื่อได้รับใบรับรองของผู้อื่น จำเป็นต้องตรวจสอบยืนยันใบรับรองทุกใบจนถึงใบรับรองของ Root CA



รูป 2.3 การตรวจสอบโดยใช้ลำดับขั้นของใบรับรอง

2.3.2 Certificate Revocation List (CRL)

CRL ประกอบไปด้วยลิสต์ serial number ของใบรับรองที่ทำการเพิกถอนโดย CA โดยผู้ใช้งานจะตรวจสอบ serial number ของใบรับรองกับรายการดังกล่าวเพื่อตรวจสอบสถานะของใบรับรอง

รายการใบรับรองอิเล็กทรอนิกส์ที่ไม่ได้ใช้งาน เนื่องจากสาเหตุดังต่อไปนี้

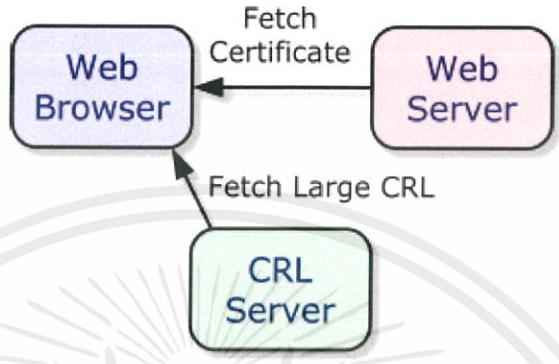
1) กุญแจส่วนตัวที่ใช้กับใบรับรองอิเล็กทรอนิกส์ถูกล่วงรู้หรือยึดครอง
2) กุญแจส่วนตัวที่ได้รับจาก CA ถูกล่วงรู้หรือยึดครอง
3) ผู้ที่เป็นเจ้าของใบรับรองอิเล็กทรอนิกส์ไม่ได้เป็นส่วนหนึ่งของผู้ที่ออกใบรับรองอิเล็กทรอนิกส์อีกต่อไป หรือไม่มีสิทธิ์ในได้รับการเข้าถึงใบรับรองอิเล็กทรอนิกส์หรือไม่มีความจำเป็นที่จะต้องใช้อีกต่อไป
4) มีใบรับรองอิเล็กทรอนิกส์ใบอื่นที่มาแทนใบเก่า
5) ผู้ให้บริการออกใบรับรองอิเล็กทรอนิกส์ที่เป็นผู้ออกใบอิเล็กทรอนิกส์ให้หยุดการให้บริการ
6) ใบรับรองอิเล็กทรอนิกส์ถูกระงับการใช้งานชั่วคราว เพื่อทำการตรวจสอบสถานะของใบรับรองอิเล็กทรอนิกส์นั้น ๆ

ข้อเสียหลัก ๆ ของ CRL มีดังนี้

1) บางครั้งจำเป็นต้องค้นหาหมายเลขซีเรียลขนาดใหญ่

2) มีการอัพเดตเป็นช่วง ๆ ทำให้ถ้าเกิดการโจมตีใด ๆ จะต้องรอจนกว่าจะมีการอัพเดตในครั้งต่อไป

3) ถ้าผู้ใช้งานไม่สามารถดาวน์โหลด CRL ได้ โดยพื้นฐานจะถือว่าใบรับรองมีความน่าเชื่อถือ



รูป 2.4 การตรวจสอบใบรับรองโดยใช้ CRL

2.3.3 Online Certificate Status Protocol (OCSP)

OCSP ได้แก้ไขข้อเสียหลายอย่างของ CRL โดยยอมให้ไคลเอนต์ตรวจสอบสถานะของใบรับรองได้ โดยมีขั้นตอนดังต่อไปนี้

1) ไคลเอนต์ได้รับหรือถือครองใบรับรอง

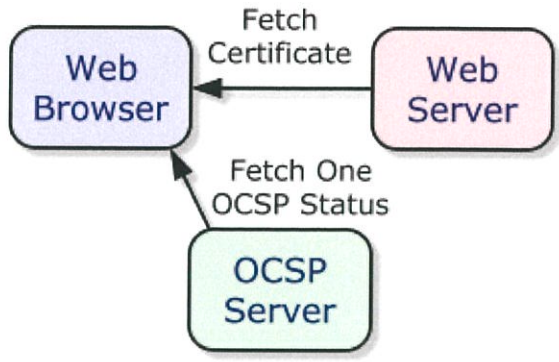
2) ไคลเอนต์ทำการส่ง OCSP Request มายัง OCSP Responder (ผ่าน HTTP) พร้อมกับหมายเลขซีเรียลของใบรับรอง

3) OCSP Responder จะตอบกลับด้วย สถานะต่าง ๆ ของใบรับรอง ซึ่งประกอบไปด้วย "ยกเลิก" (revoked), "ดี" (good), หรือ "ไม่ทราบ" (unknown)

ข้อดีหลักของการตรวจสอบแบบ OCSP คือไคลเอนต์สามารถสอบถามสถานะของใบรับรองทีละใบ แทนที่จะต้องดาวน์โหลดและส่งทั้งรายการ ซึ่งทำให้เกิดภาระแก่ไคลเอนต์และเครือข่าย แต่อย่างไรก็ตาม OCSP ก็ยังมีข้อเสียคือ

1) อาจทำให้ OCSP Responder เช่น CA รับภาระหนัก เมื่อเว็บไซต์ที่มีผู้ใช้งานเยอะ

2) ถ้าหากกุญแจส่วนตัวถูกยึดครอง และมีผู้กระทำการโจมตีแบบคนคั่นกลางมาคอยดักรับและทำส่งข้อมูลแทนเซิฟเวอร์ ซึ่งเบราเซอร์ส่วนใหญ่ ไม่ได้สนใจว่าการตรวจสอบด้วยวิธีนี้จะใช้เวลานานแค่ไหน



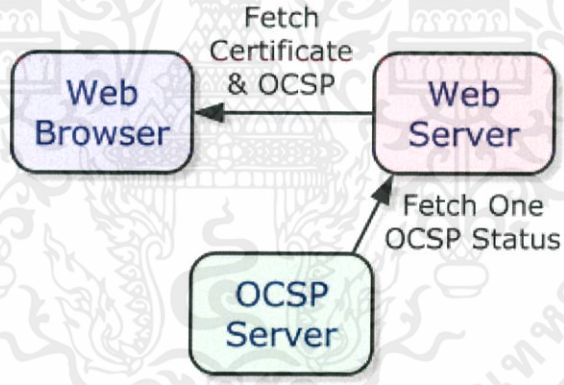
รูป 2.5 การตรวจสอบใบรับรองแบบ OCSP

2.3.4 Online Certificate Status Protocol Stapling

การทำ OCSP Stapling แก้ไขปัญหา overhead ที่เกิดขึ้นกับทั้ง OCSP และ CRL โดยจะมีผู้ถือครองใบรับรอง เช่น เซิร์ฟเวอร์ คอยทำการตรวจสอบด้วยวิธี OCSP เป็นระยะ จากนั้นไคลเอนต์จะได้รับสถานะของใบรับรอง เมื่อทำการเชื่อมต่อแบบ SSL Handshake แต่วิธีนี้มีข้อจำกัดคือ

1) รองรับเพียงโพรโทคอล TLS 1.2

2) ยังไม่รองรับในหลาย ๆ เบราเซอร์



รูป 2.6 การตรวจสอบใบรับรองแบบ OCSP Stapling

2.3 Domain Name System (DNS)

ระบบที่ใช้แปลงชื่อโดเมนให้เป็นที่อยู่ไอพีเพื่อใช้งานอินเทอร์เน็ต เช่น www.ce.kmitl.ac.th คือ 161.246.4.119 ซึ่งโดยปกติแล้ว เครื่องคอมพิวเตอร์จะสื่อสารกับเซิร์ฟเวอร์ของ DNS ผ่านพอร์ตหมายเลข 53

2.4 HTTP Strict Transport Security (HSTS)

เป็นวิธีการทางด้านความปลอดภัยของเว็บ ที่ช่วยป้องกันการโจมตีด้วยการลดระดับโพรโทคอล โดยให้เว็บเซิร์ฟเวอร์บอกเว็บเบราเซอร์ให้ติดต่อกันด้วยการเชื่อมต่อแบบ HTTPS เท่านั้น

เฮดเดอร์ของ HSTS ประกอบด้วย 2 องค์ประกอบหลัก ดังนี้

1) max-age: เพื่อบ่งบอกว่าเบราเซอร์ควรจะใช้เวลามากที่สุดกี่วินาที ในการเปลี่ยนจากการร้องของแบบ HTTP เป็น HTTPS

2) includeSubDomain: เพื่อบ่งบอกว่าทุก ๆ โดเมนย่อยของเว็บต้องใช้ HTTPS

2.4.1 เว็บเบราเซอร์ที่รองรับการใช้งาน HSTS

ตารางที่ 2.1 การรองรับ HSTS ของเบราเซอร์ต่าง ๆ

เบราเซอร์	เวอร์ชั่นที่เริ่มรองรับ
Internet Explorer	11
Edge	ทุกเวอร์ชั่น
Firefox	4
Opera	12.1
Safari	7
Chrome	4
iOS Safari	7.1
Opera Mini	ยังไม่รองรับ
Android Browser	4.4
Blackberry Browser	7
Internet Explorer Mobile	ยังไม่รองรับ
UC Browser for Android	ยังไม่รองรับ
Samsung Internet	4

2.5 HTTP Public Key Pinning (HPKP)

HPKP เป็นวิธีที่ตรวจสอบค่าแฮช (Hash) ของใบรับรองนั้นว่าตรงกับที่ระบบไว้ในเฮดเดอร์ของ HTTP ในครั้งแรกที่ใช้บริการหรือไม่ ซึ่งถ้าได้ข้อมูลดังกล่าวแล้ว ไคลเอนต์ (Client) จะเชื่อถือค่าแฮชนั้นในระยะเวลาที่กำหนดไว้ หากช่วงเวลานี้ ค่าแฮชของใบรับรองไม่ตรงกัน เบราเซอร์จะไม่เชื่อถือใบรับรอง และจะอัพเดตค่าแฮชใหม่อีกครั้งเมื่อระยะเวลาดังกล่าวสิ้นสุดลง

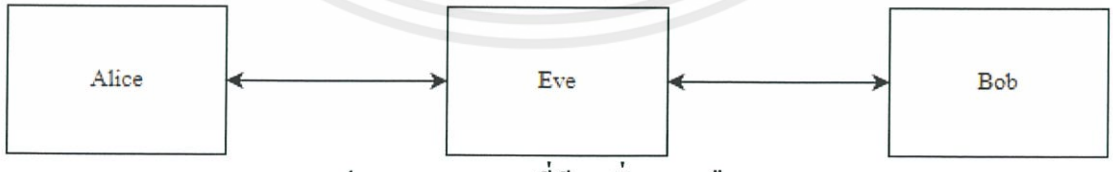
วิธีดังกล่าวมีข้อเสียคือ หากในครั้งแรกที่เข้าใช้บริการ เครื่องที่เข้าใช้งานได้รับข้อมูลที่เป็นค่าแฮชปลอมบน HTTP Header ไปก่อนแล้ว วิธีการนี้ก็จะไม่สามารถตรวจสอบได้อีกต่อไปว่าจะเชื่อค่าแฮชชุดใดกันแน่ที่เป็นค่าแฮชจริง วิธีนี้จึงเป็นเพียงแค่ช่วยเพิ่มความสะดวกสำหรับผู้ใช้งานเบราว์เซอร์ทั่วไป

2.6 การโจมตีจากคนคั่นกลาง

การโจมตีจากคนคั่นกลาง (Man-in-the-middle Attack หรือ MitM หรือ MiM) หมายถึง การที่มีผู้ไม่หวังดี เข้ามาแทรกกลางในการสนทนาระหว่างคน 2 คน แล้วทำหน้าที่เป็นตัวกลางในการรับส่งข้อมูลของคู่สนทนา โดยที่คู่สนทนาไม่ทราบ ทำให้ผู้ที่โจมตีสามารถดักฟังหรือเปลี่ยนแปลงข้อมูลที่ทั้ง 2 ฝั่ง สื่อสารกันอยู่ได้

ยกตัวอย่างการสนทนาระหว่างอลิซ (Alice) กับบ็อบ (Bob) ซึ่งเริ่มหลังจากที่สร้างช่องทางการสื่อสารและผ่านการยืนยันตัวตนมาแล้ว แต่ถ้าหากโพรโทคอลสำหรับยืนยันตัวตนดีไม่พอ จะทำให้อีฟ (Eve) สามารถปลอมตัวเป็นทั้งอลิซและบ็อบได้

รูป 2.7 การสนทนาปกติระหว่าง Alice กับ Bob

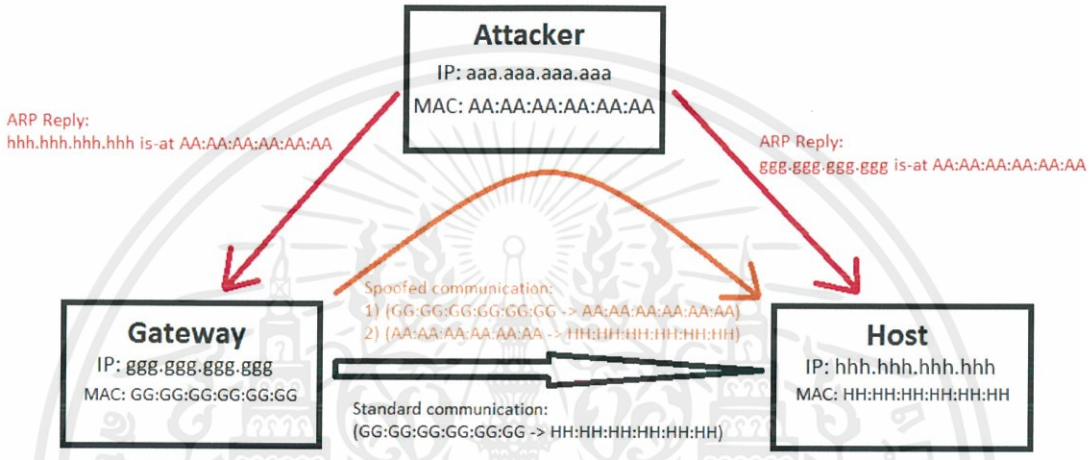


รูป 2.8 การสนทนาที่มีคนคั่นกลางคือ Eve

2.7 การเปลี่ยนแปลงเส้นทางของข้อมูล

2.7.1 ARP spoofing

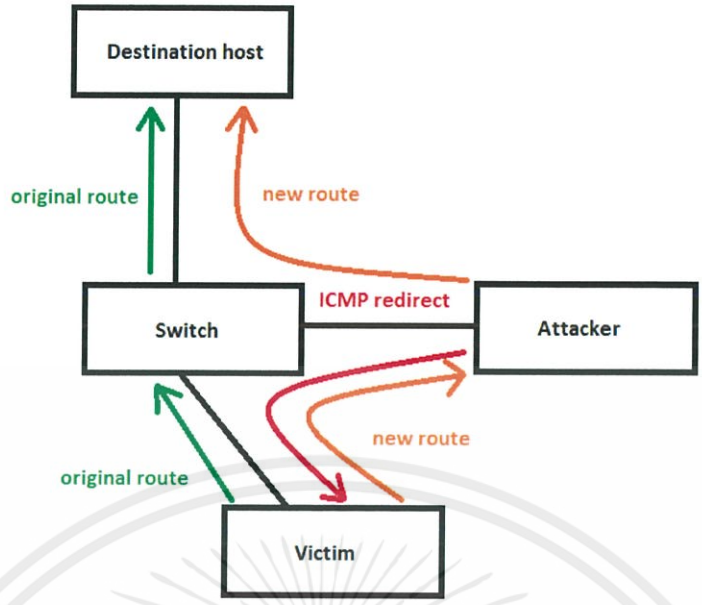
ARP Spoofing หรือ ARP Cache Poisoning ผู้โจมตีจะทำการบอกเครื่องเป้าหมายว่าหมายเลขไอพีของเกตเวย์อยู่ที่เครื่องของผู้โจมตี และบอกเกตเวย์ว่าหมายเลขไอพีของเครื่องเป้าหมายอยู่ที่เครื่องของผู้โจมตีเช่นกัน ด้วยการปลอมแพ็คเกต ARP ขึ้นมา ซึ่งการโจมตีด้วยวิธีนี้เครื่องเป้าหมายจำเป็นต้องมีการบันทึก ARP cache แบบไดนามิคเท่านั้น



รูป 2.9 หลักการทำงานของ ARP Spoofing

2.7.2 ICMP redirect attack

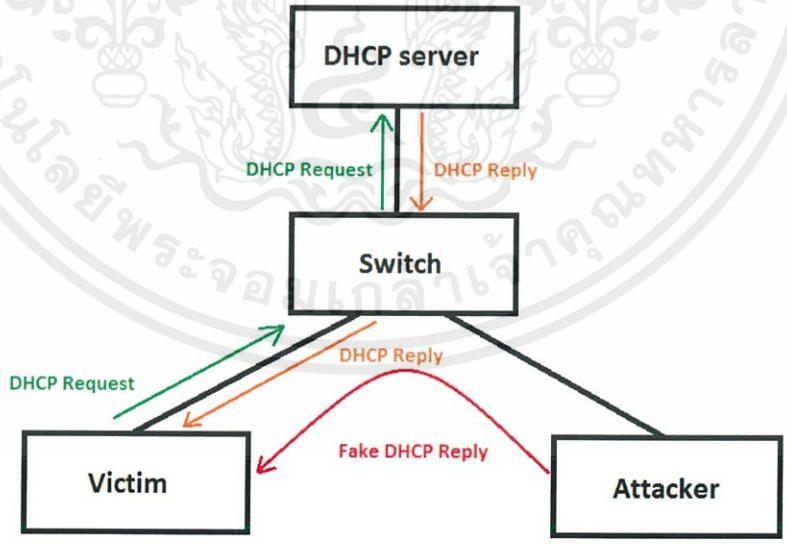
ผู้โจมตีจะทำการส่งแพ็คเกต ICMP ไปยังเครื่องเป้าหมาย เพื่อบอกว่าเป็นเส้นทางไปยังเกตเวย์ที่สั้นที่สุด ทำให้ผู้โจมตีสามารถดักฟังการสื่อสารได้ และยังสามารถปลอมหมายเลขไอพีและที่อยู่ MAC ให้เหมือนกับว่ามาจากเกตเวย์จริง



รูป 2.10 หลักการของ ICMP redirect attack

2.7.3 DHCP spoofing

เมื่อผู้โจมตีกำลังดักฟังข้อมูลที่วิ่งผ่านอยู่ และพบ DHCP Request ก็จะสร้าง DHCP Response และส่งไปยังเครื่องเป้าหมาย ก่อนที่การตอบจากเซิร์ฟเวอร์จริงจะส่งกลับมาถึง ซึ่งทำให้เครื่องเป้าหมายรับรู้ว่าหมายเลขไอพีของผู้โจมตีคือเกตเวย์



รูป 2.11 หลักการของ DHCP spoofing

2.8 ส่วนเสริมของเบราเซอร์ Internet Explorer

ส่วนเสริมของเบราเซอร์ยอมให้ผู้พัฒนาสามารถเพิ่มฟังก์ชันที่เพิ่มประสิทธิภาพของหน้าต่างผู้ใช้งาน โดยผู้ใช้งานสามารถเลือกที่จะติดตั้งส่วนเสริมนี้ได้ แบ่งได้ดังนี้

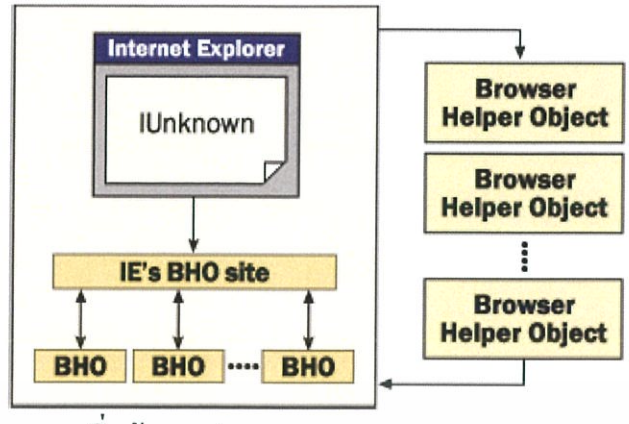
1) Shortcut menu extensions คือ การเพิ่มตัวเลือกในเมนูทางลัด สามารถทำได้โดยเพิ่ม Registry Keys ที่เชื่อมโยงคำสั่งการทำงานของเมนูในการดำเนินงาน
2) Toolbars คือ แถบเครื่องมือที่ปรับปรุงขึ้นจากมาตรฐานเพื่ออำนวยความสะดวก
3) Explorer Bars คือ แถบช่วยเหลือที่อยู่บริเวณด้านข้างหรือด้านล่างของเว็บไซต์
4) Browser Helper Objects ทำงานภายใน Internet Explorer เพื่อให้การบริการเพิ่มเติมแก่ผู้ใช้งาน โดยมากมักไม่มีหน้าต่างการใช้งาน

2.9 Browser Helper Objects

Internet Explorer มีพื้นที่หน่วยความจำของตัวเอง เหมือนกับโปรแกรม Win32 ตัวอื่น ๆ ด้วย BHO เราสามารถที่จะเขียนองค์ประกอบ (component) ต่าง ๆ ที่ IE จะทำการเรียกใช้ ตอนเริ่มใช้งานในแต่ละครั้ง อ็อบเจกต์ต่างๆที่ทำการประมวลผลในหน่วยความจำเดียวกันกับเบราเซอร์จะสามารถดำเนินการหรือกระทำใด ๆ ต่อ window และ module ต่าง ๆ ได้ เช่น BHO สามารถตรวจจับเหตุการณ์ทั่วไปในเบราเซอร์ เช่น GoBack ,GoForward และ DocumentComplete , เข้าถึงแถบเมนูและแถบเครื่องมือของเบราเซอร์ และทำการเปลี่ยนแปลง หรือสร้างหน้าต่างเพื่อแสดงข้อมูลเพิ่มเติมในหน้าเว็บไซต์ที่ทำการเปิดได้

BHO จะทำการเชื่อมโยงกับหน้าหลักของเบราเซอร์ ในทางปฏิบัติหมายความว่าอ็อบเจ็กต์จะถูกสร้างใหม่พร้อมกับเบราเซอร์ ซึ่งหมายถึง BHO จะมีชีวิตและตายไปกับพร้อมกับเบราเซอร์นั้นๆ และ BHO นั้นมีอยู่ใน IE 4.0 และรุ่นที่ใหม่กว่าเท่านั้น

เมื่อเริ่มต้น Internet Explorer จะทำการตรวจสอบ key และทำการโหลดอ็อบเจกต์ทุก ๆ ตัวที่มี CLSID เก็บอยู่ในนั้นออกมา เบราเซอร์จะทำการเริ่มต้นอ็อบเจกต์และใช้งานอินเตอร์เฟสต่างๆ เมื่อเจออินเตอร์เฟซ IE จะใช้เมธอด (method) ที่มีเพื่อส่ง pointer ของ IUnknown ไปยัง BHO ดังรูป



รูป 2.12 การเริ่มต้นการทำงานของ Browser Helper Objects

2.10 OpenSSL

OpenSSL เป็นโครงการโอเพนซอร์ซ (Open Source) ที่ประกอบไปด้วยไลบรารี่ (Library) สำหรับการเข้ารหัสลับ และชุดเครื่องมือสำหรับ SSL/TLS ซึ่งจากเว็บไซต์ของโครงการระบุว่า โครงการ OpenSSL เป็นความพยายามร่วมมือกันพัฒนาชุดเครื่องมือที่มีคุณภาพ , การทำงานครบครัน และเป็นโอเพนซอร์ซ ที่ประยุกต์ใช้งานโพรโทคอล SSL และ TLS พร้อมกับมีไลบรารี่ที่เกี่ยวกับการเข้ารหัสลับเพื่อการใช้งานหลากหลายวัตถุประสงค์

OpenSSL ถูกมองว่าเป็นมาตรฐานหลักของเรื่องเหล่านี้ และมีประวัติที่ยาวนาน ซึ่งเริ่มต้นในปีค.ศ. 1995 พัฒนาโดย Eric A. Young และ Tim J. Hudson ภายใต้ชื่อ SSLeay และต่อจากนั้นก็หันมาพัฒนาโครงการ OpenSSL แทนในช่วงปลายปี ค.ศ. 1998

ทุกวันนี้ OpenSSL ใช้งานอย่างแพร่หลายบนเครื่องเซิร์ฟเวอร์ และในเครื่องมือต่าง ๆ บนเครื่องไคลเอนต์ แต่น่าแปลกใจที่เบราเซอร์ต่าง ๆ มักจะใช้ไลบรารี่อื่น โดยคำสั่งของ OpenSSL ส่วนใหญ่จะเกี่ยวข้องกับกันจัดการกุญแจและใบรับรองต่าง ๆ

2.10.1 การสร้างชุดใบรับรองที่น่าเชื่อถือ

เมื่อติดตั้ง OpenSSL จะไม่มีใบรับรองที่เชื่อถือได้ของ root ซึ่งเรียกได้อีกชื่อหนึ่งคือ trust store ติดมาด้วย จึงจำเป็นที่จะต้องหาจากแหล่งอื่น หนึ่งในทางเลือกคือ trust store ที่มาพร้อมกับระบบปฏิบัติการ แต่ทางเลือกนี้อาจจะไม่ทันสมัยอยู่ตลอดเวลา จึงมีทางเลือกที่ดีกว่า คือใช้ข้อมูลของ Mozilla ที่พยายามจะสร้าง trust store ที่มีคุณภาพขึ้นมา ซึ่งเป็นโอเพนซอร์ซและสามารถดูโค้ดได้ที่ https://hg.mozilla.org/mozilla-central/raw-file/tip/security/nss/lib/ckfw/builtins/certdata.txt

ชุดใบรับรองข้างต้นนั้น อยู่ในรูปแบบของทาง Mozilla ซึ่งไม่สามารถใช้งานกับของผู้พัฒนาอื่นได้ และจำเป็นต้องแปลงให้อยู่ในรูปแบบที่สามารถใช้งานได้ก่อน โดยทางเลือกหนึ่งคือโครงการ Curl ที่ทำการแปลงชุดใบรับรองข้างต้นให้สามารถใช้งานได้ทันที ซึ่งสามารถดูและดาวน์โหลดมาใช้งานได้จาก https://curl.haxx.se/docs/caextract.html

2.10.2 การสร้างกุญแจ

ขั้นตอนแรกในการเตรียมพร้อมในการเข้ารหัสลับคือการสร้างกุญแจส่วนตัว ซึ่งต้องกำหนดค่าต่าง ๆ ดังหัวข้อต่อไปนี้

2.10.2.1 ขั้นตอนวิธีในการสร้างกุญแจ (Key algorithm)

OpenSSLรองรับ RSA, DSA และ ECDSA แต่ในทางปฏิบัติจะใช้แค่บางประเภทเท่านั้น ตัวอย่างเช่น กุญแจสำหรับ SSL จะใช้ RSA เนื่องจาก กุญแจแบบ DSA จะมีประสิทธิภาพจำกัดที่ 1024 บิต และกุญแจแบบ ECDSA นั้น CA ยังรองรับไม่แพร่หลาย และ สำหรับ SSH กุญแจแบบ DSA และ RSA ใช้อย่างแพร่หลายในขณะที่ ECDSA ไคลเอนต์บางส่วนไม่รองรับ

2.10.2.2 ขนาดของกุญแจ

การใช้ค่าเริ่มต้นของขนาดกุญแจ อาจไม่ปลอดภัย ทำให้จำเป็นต้องระบุขนาดของกุญแจเองทุกครั้ง ตัวอย่างเช่น ค่าเริ่มต้นสำหรับกุญแจแบบ RSA มีขนาดเพียง 512 บิต ซึ่งไม่ปลอดภัยอย่างยิ่ง ถ้าหากเซิร์ฟเวอร์ใช้กุญแจขนาด 512 บิต ผู้บุกรุกสามารถค้นหากุญแจส่วนตัวและทำการปลอมเว็บไซต์ขึ้นมาได้ ทุกวันนี้ กุญแจแบบ RSA ควรมีขนาด 2048 บิต จึงจะถือว่าปลอดภัย เช่นกันสำหรับกุญแจแบบ DSA ควรมีขนาดอย่างน้อย 2048 บิต และกุญแจแบบ ECDSA ควรมีขนาดอย่างน้อย 224 บิต

2.10.2.3 ข้อความรหัสผ่าน (Passphrase)

การใช้ข้อความรหัสผ่านนั้นเป็นตัวเลือกซึ่งไม่จำเป็นต้องใช้ แต่ควรใช้ เพราะทำให้กุญแจถูกเก็บรักษา, เคลื่อนย้าย และใช้งานได้อย่างปลอดภัย แต่ก็ทำให้เกิดความไม่สะดวกตามมา เช่น อาจจะต้องใส่ข้อความรหัสผ่านทุกครั้งเมื่อต้องการเริ่มการใช้งานเว็บเซิร์ฟเวอร์ใหม่

2.10.3 การสร้าง Certificate Signing Requests (CSR)

เป็นมาตรฐานในการส่งไฟล์ที่เข้ารหัส เพื่อส่งกุญแจสาธรณะและข้อมูลบางส่วนที่สามารถบ่งบอกถึงองค์กรและชื่อโดเมน เมื่อทำการสร้าง CSR เซิร์ฟเวอร์ส่วนใหญ่จะทำการร้องขอข้อมูลดังต่อไปนี้ ชื่อเว็บไซต์ ชื่อบริษัท ที่อยู่ ประเภทกุญแจที่ใช้ซึ่งปกติจะใช้ RSA และขนาดของกุญแจ (ขั้นต่ำ 2048 บิต)

2.10.4 การรับรองใบรับรองที่สร้างขึ้นมาเอง

ถ้าหากต้องการใช้ใบรับรองเพื่อใช้งานเอง โดยไม่ต้องการจดทะเบียนกับ CA เพื่อใช้งานแบบสาธารณะ เราสามารถรับรองใบรับรองดังกล่าวด้วยตนเอง ได้โดยวิธีการดังต่อไปนี้

เมื่อมี CSR อยู่แล้วสามารถใช้คำสั่ง

```
$ openssl x509 -req -days 365 -in fd.csr -signkey fd.key -out fd.crt
```

แต่ถ้าไม่มี CSR สามารถใช้คำสั่งต่อไปนี้แทน

```
$ openssl req -new -x509 -days 365 -key fd.key -out fd.crt
```

2.11 Libcurl

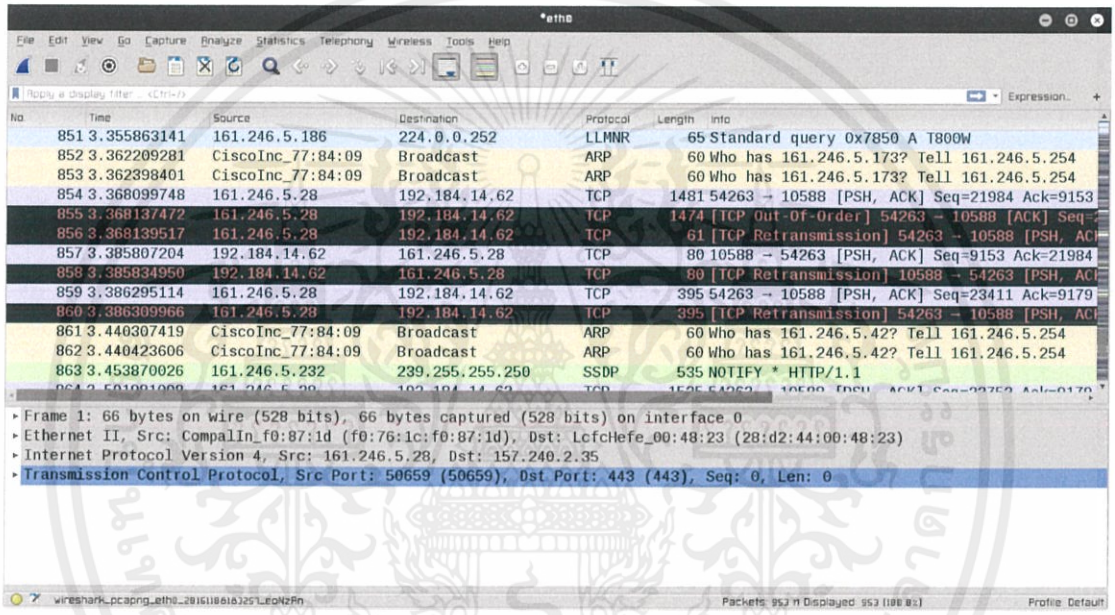
เป็นไลบรารี่ในการรับส่งข้อมูลสำหรับไคลเอนต์ รองรับโพรโทคอลที่หลากหลาย และรองรับ SSL/TLS บน OpenSSL และไลบรารี่อื่น ๆ ถูกสร้างให้สามารถใช้งานได้หลายแพลตฟอร์ม ทำให้มีความสะดวกในการใช้งานในแพลตฟอร์มที่แตกต่างกัน

บทที่ 3
วิธีการทดลอง

3.1 เครื่องมือที่ใช้

3.1.1 Wireshark

เป็นโปรแกรมสำหรับวิเคราะห์แพ็กเกต (Packet) ในเครือข่าย ซึ่งจะคอยดักจับแพ็กเกตในเครือข่าย แล้วนำมาแสดงผล



รูป 3.1 หน้าต่างโปรแกรม Wireshark

3.1.2 Arpspoof

โปรแกรม arpspoof จะทำการ ARP Spoofing โดยปลอมแปลงแพ็กเกต ARP ขึ้นมาแล้วส่งไปยังเครื่องเป้าหมายและเราเตอร์ (Router) เพื่อเปลี่ยนเส้นทางข้อมูลจากเครื่องเป้าหมายไปยังอีกเครื่องหนึ่งในเครือข่ายแลน (LAN) เดียวกัน

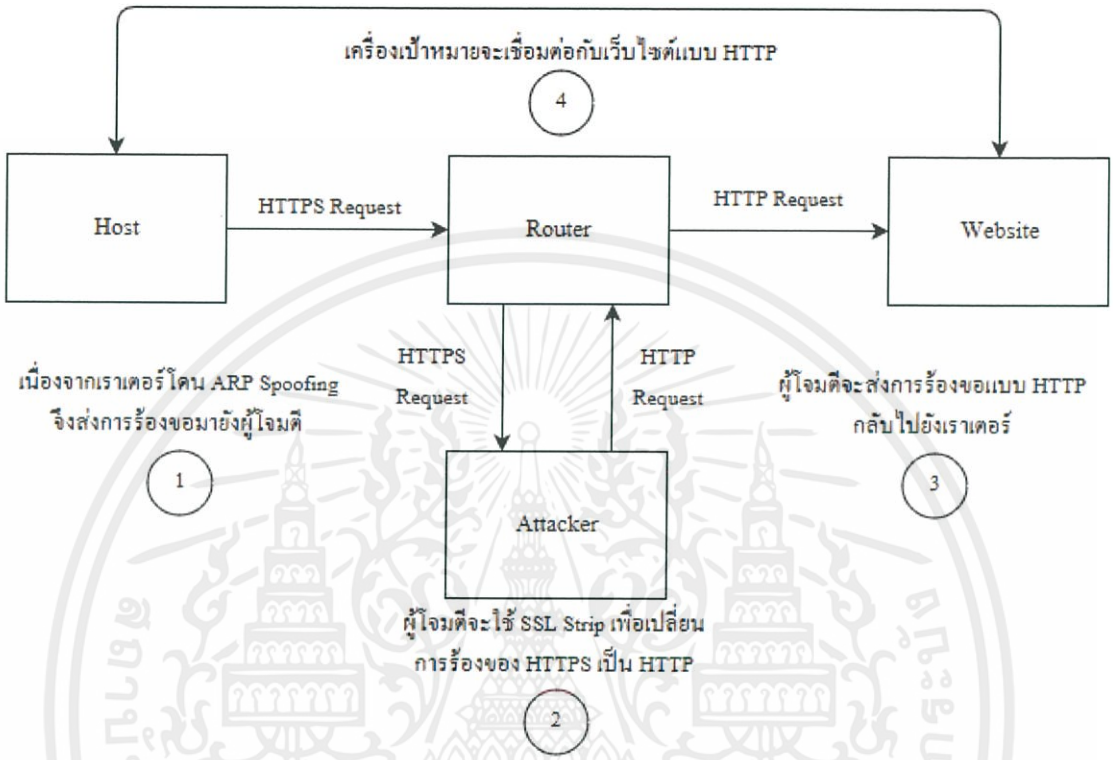
3.1.3 SSL Split

เป็นเครื่องมือสำหรับการโจมตีจากคนคั่นกลาง ที่ใช้กับการเชื่อมต่อด้วยโพรโทคอล SSL/TLS ซึ่ง SSLsplit ถูกออกแบบมาให้ยุติการเชื่อมต่อของเครื่องเป้าหมาย และสร้างการเชื่อมใหม่เพื่อเก็บข้อมูลที่สื่อสารกันระหว่างเครื่องเป้าหมายกับเซิร์ฟเวอร์

สำหรับการเชื่อมต่อด้วย SSL และ HTTPS SSLsplit สามารถสร้างใบรับรองปลอม หรือเลือกใช้ใบรับรองที่มีอยู่แล้วได้ รวมไปถึงปฏิเสธการร้องข้อ OCSP, ลบเฮดเดอร์ของ HPKP ที่มาจากเซิร์ฟเวอร์ เพื่อป้องกันการจดจำกุญแจสาธารณะ

3.1.4 SSL Strip

เป็นโปรแกรมที่ลดระดับการร้องขอหน้าเว็บจากโพรโทคอล HTTPS เป็น HTTP ซึ่งถ้าหากผู้ใช้งานไม่ได้สังเกตแถบที่อยู่เว็บไซต์ว่าเป็น HTTP อาจถูกผู้โจมตีจะสามารถดักฟังได้โดยง่าย



รูป 3.2 หลักการทำงานของ SSL Strip

3.1.5 Mitmproxy

เป็นโปรแกรมที่เป็นตัวกลางสำหรับการโจมตีแบบคนคั่นกลาง เพื่อแสดงรายละเอียดของแพ็กเกตโพรโทคอล HTTP/HTTPS ที่วิ่งผ่าน และรองรับการเขียนสคริปต์ (Script) ด้วยภาษาไพทอน (Python)

3.1.6 Ettercap

เป็นชุดโปรแกรมที่ครอบคลุมการทดสอบการโจมตีแบบคนคั่นกลางหลายรูปแบบ รองรับโพรโทคอลที่หลากหลาย และอำนวยความสะดวกในการวิเคราะห์เครื่องคอมพิวเตอร์และเครือข่าย มีรูปแบบหน้าต่างการใช้งานให้เลือก 3 รูปแบบ

3.1.7 OpenSSL

เป็นชุดเครื่องมือที่อำนวยความสะดวกในการใช้งาน TLS และ SSL อีกทั้งมีไลบรารี่ (Library) ในการเข้าและถอดรหัสลับหลายรูปแบบ

3.1.8 Apache HTTP Server

สำหรับเปิดให้บริการเว็บไซต์ เพื่อให้คอมพิวเตอร์เครื่องอื่นสามารถเรียกชมหน้าเว็บไซต์โดยใช้โพรโทคอล HTTP ผ่านทางเว็บเบราเซอร์

3.1.9 เบราเซอร์

แต่ละเบราเซอร์ จะมีการจัดการทางด้านความปลอดภัยที่แตกต่างกัน จึงได้ศึกษาและทำการทดลองโดยใช้เบราเซอร์ที่หลากหลาย อาทิ เช่น Chrome, Firefox, Internet Explorer เป็นต้น

3.2 การทดสอบเว็บไซต์ที่มีการใช้งาน HSTS

การทดสอบทำได้ผ่านเว็บไซต์ https://hstspreload.appspot.com/ หรือดูว่า เมื่อเซิร์ฟเวอร์ตอบสนอง มีเฮดเดอร์ (Header) อยู่หรือไม่ โดยใช้คำสั่งต่อไปนี้

```
$ curl -s -D- https://domain.com/ | grep Strict
```

และผลลัพธ์เมื่อเว็บไซต์นั้นรองรับ HSTS จะแสดงดังนี้

```
Strict-Transport-Security: max-age=...
```

3.3 การทดลองโจมตีจากคนคั่นกลาง

3.3.1 การดักจับข้อมูลแพ็กเกตโพรโทคอลทั่วไป

สำหรับข้อมูลต่าง ๆ ที่ไม่ได้มีการเข้ารหัสลับ มีวิธีการทดลองดังนี้

1) เปิดการใช้งานการฟอร์เวิร์ดพอร์ตเพื่อให้ข้อมูลสามารถเดินทางผ่านเครื่องของผู้โจมตีด้วยคำสั่ง $ sysctl net.ipv4.ip_forward=1

โดยสามารถตั้งค่า พอร์ตที่เข้าและจะออกได้ โดยใช้คำสั่ง $ iptables -t nat -A PREROUTING -p tcp --dport SRCPORT -j REDIRECT --to-ports DESTPORT

2) เปลี่ยนเส้นทางของข้อมูลด้วยวิธีใดวิธีหนึ่ง ซึ่งในที่นี้จะใช้ ARP Spoofing โดยใช้คำสั่ง $ arpspoof -I INTERFACE -t TARGET1 -r TARGET2

3) เรียกใช้โปรแกรมที่สามารถดูรายละเอียดข้อมูลที่วิ่งผ่านได้ เช่น Wireshark, mitmproxy หรืออื่น ๆ

3.3.2 การดักจับข้อมูลแพ็กเกตโพรโทคอลที่มีความปลอดภัย

เนื่องจากโพรโทคอลที่มีความปลอดภัยต่าง ๆ มีการเข้ารหัสลับของข้อมูล เมื่อดักฟังจึงไม่สามารถทราบได้ว่าข้อความจริง ๆ คืออะไร จึงมีการโจมตีรูปแบบต่าง ๆ ที่ทำให้เครื่องเป้าหมายไม่สามารถเชื่อมต่อด้วยโพรโทคอลที่เข้ารหัสลับกับเซิร์ฟเวอร์ได้ ได้แก่ตัวอย่างดังต่อไปนี้

3.3.2.1 การโจมตีโดยลดระดับโพรโทคอล

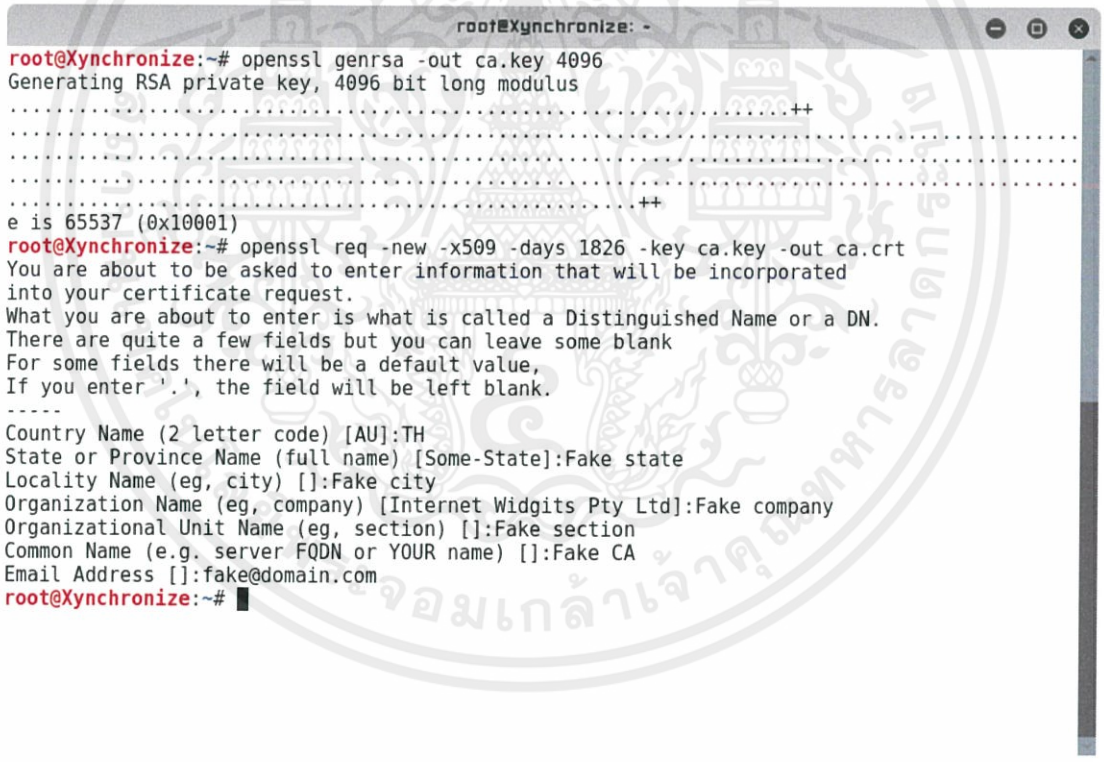
เป็นการโจมตีโดยเปลี่ยนการร้องขอ HTTPS เป็น HTTP โดยใช้คำสั่ง $ sslstrip -l PORTNUMBER แล้วจึงดักฟังตามปกติ ซึ่งการโจมตีด้วยวิธีนี้สามารถป้องกันด้วยการใช้ HSTS

3.3.2.2 การโจมตีโดยใช้ใบรับรองปลอม

1) เรียกใช้โปรแกรม OpenSSL เพื่อสร้างกุญแจส่วนตัว ขนาด 4096 บิต ด้วยคำสั่ง $ openssl genrsa -out KEYFILE 4096

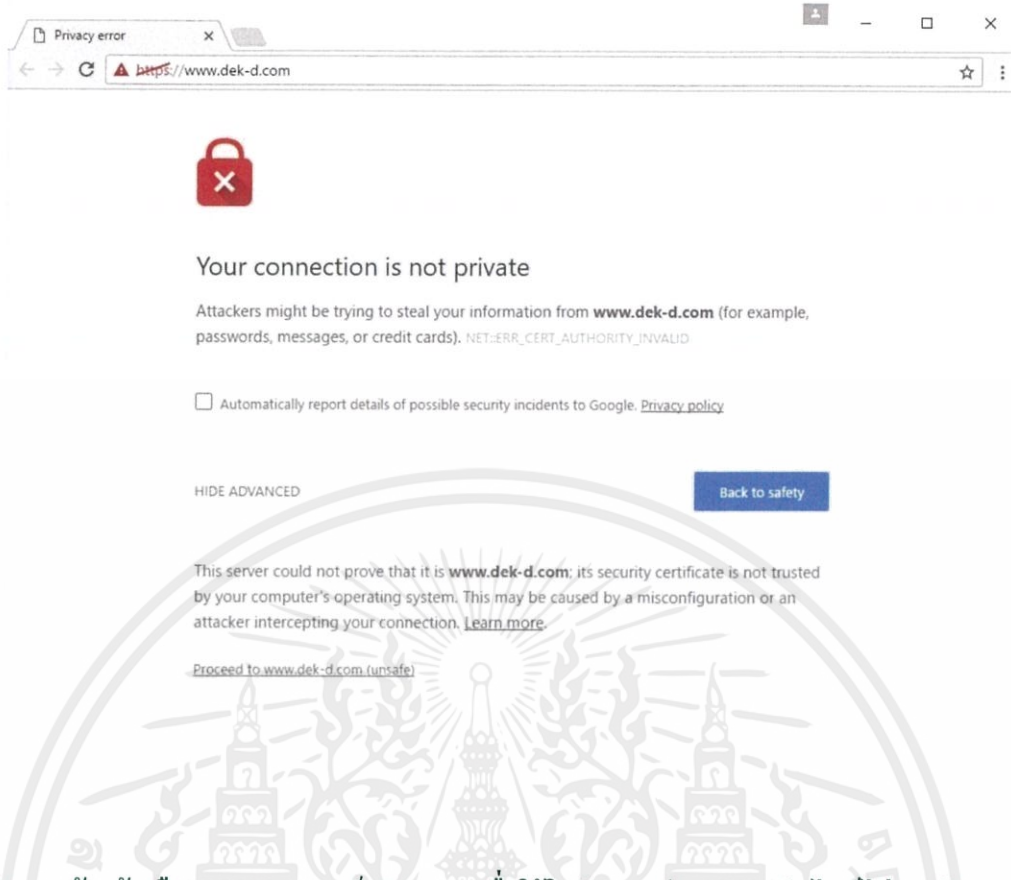
2) สร้างกุญแจสาธารณะ และใบรับรอง ตามมาตรฐาน X.509 ด้วยกุญแจส่วนตัวที่สร้างขึ้นด้วยคำสั่ง $ openssl req -new -x509 -day DAYS -key KEYFILE -out CERTFILE

3) กรอกรายละเอียดต่าง ๆ สำหรับการสร้างใบรับรอง เมื่อเสร็จแล้วจะได้ไฟล์ใบรับรอง เพื่อใช้ในการโจมตี

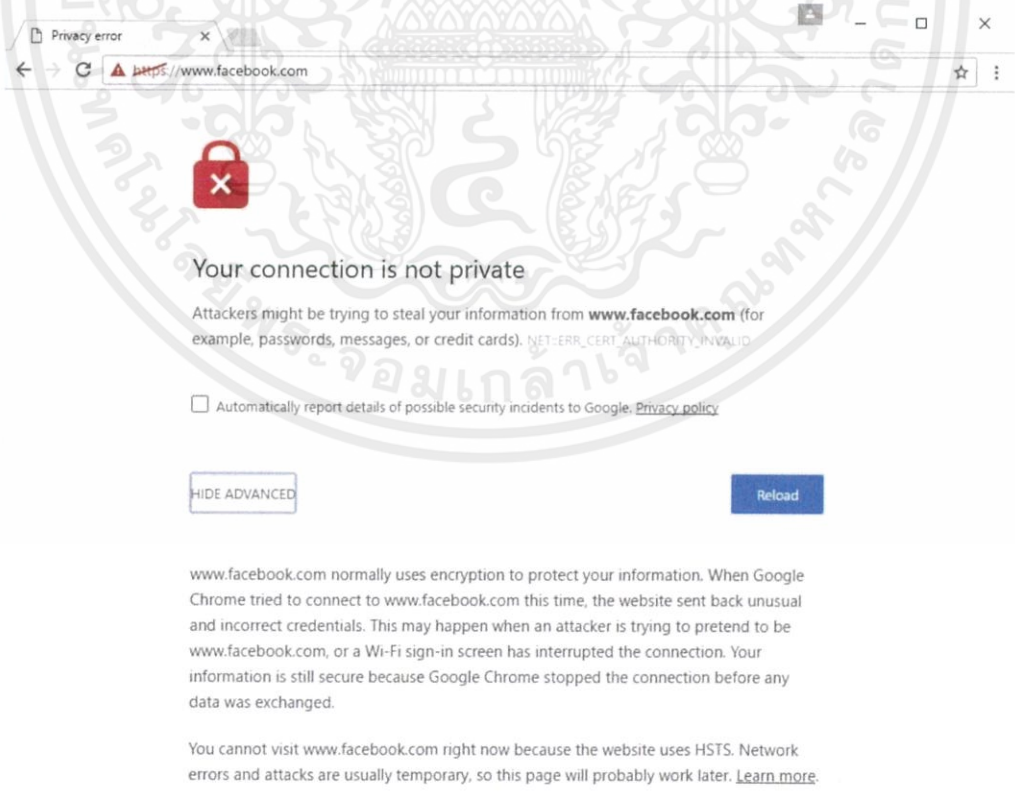


รูป 3.3 การสร้างใบรับรองด้วย OpenSSL

4) ฟอร์เวิร์ดพอร์ต 80 และ 443 จากนั้นเรียกใช้โปรแกรม sslsplit เพื่อสร้างการเชื่อมต่อกับเครื่องเป้าหมายและแสดงใบรับรองปลอมที่สร้างขึ้น ด้วยคำสั่ง $ sslsplit -D -l connections.log -S logdir/ -k KEYFILE -c CERTFILE ssl 0.0.0.0 PORTNUMBER tcp



รูป 3.4 หน้าแจ้งเตือนของเบราเซอร์ Chrome เมื่อใช้ใบรับรองปลอมและเว็บไซต์ไม่รองรับ HSTS



รูป 3.5 หน้าแจ้งเตือนของเบราเซอร์ Chrome เมื่อใช้ใบรับรองปลอมและเว็บไซต์รองรับ HSTS

3.3.2.3 การเปลี่ยนแปลงข้อมูลของแพ็กเกตด้วย Ettercap

1) สร้างไฟล์ที่มีโค้ดซึ่งมีโครงสร้างคล้ายภาษาซี สำหรับกำหนดการเปลี่ยนแปลงของข้อมูล สามารถศึกษาการเขียนโค้ดได้ที่ https://linux.die.net/man/8/etterfilter และดูตัวอย่างของโค้ดได้ที่ https://github.com/Ettercap/ettercap/blob/master/share/etter.filter.examples

ตัวอย่าง 3.1 โค้ดสำหรับการสร้างปุ่มเพื่อหลอกล่อให้ดาวน์โหลดโปรแกรม Adobe Flash Player

```
if (ip.proto == TCP && tcp.dst == 80) {
 if (search(DATA.data, "Accept-Encoding")) {
   replace("Accept-Encoding", "Accept-Rubbish!");
    msg("Zapped");
  }
}
if (ip.proto == TCP && tcp.src == 80) {
 if (search(DATA.data, "<title>")) {
 replace("</title>", "</title></p></script><form
 action='http://www.mywebsite.com' method='link'><p
 align='center'>
 <a href='http://www.mywebsite.com'>
 <img src='http://webinarcheck.sbs.edu/img/
 adobe_flash.png'> </a>
 <br> New version of Adobe Flash Please install
    updates. <br>
 <INPUT TYPE=submit value='Download!'>
 </form>");
 msg("Form Injected");
  }
}
```

2) คอมไพล์ (Compile) โค้ดที่เขียนขึ้นด้วยคำสั่ง $ etterfilter INPUTFILE -o OUTPUTFILE

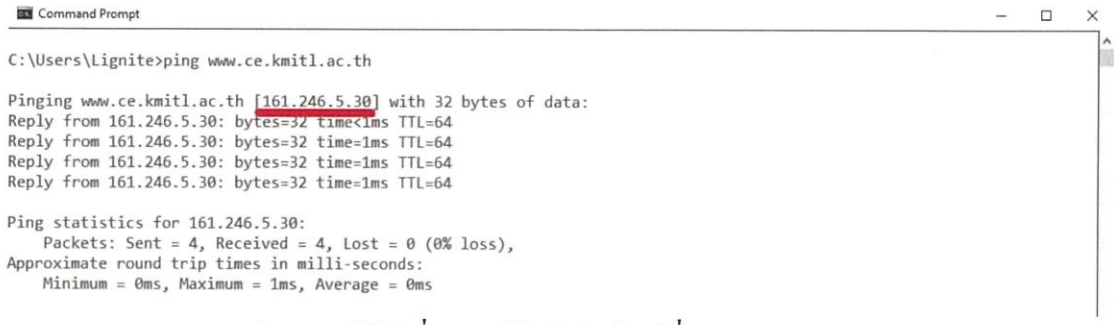
3) เรียกใช้โปรแกรม Ettercap ให้กรองด้วยไฟล์ที่ได้จากข้อ 2 โดยใช้คำสั่ง $ Ettercap -T -q -S -F FILTERFILE /IPADDRESS// ซึ่งเมื่อเครื่องเป้าหมายได้รับข้อมูลที่ตรงตามเงื่อนไข จะทำตามโค้ดที่ได้เขียนเอาไว้

รูป 3.6 เว็บไซต์ www.pantip.com เมื่อโดนเปลี่ยนเนื้อหา

3.4 การทดลองปลอม DNS (DNS spoofing) ด้วย Ettercap

การทดลองนี้ได้ทดลองให้ผู้โจมตีสร้างเว็บไซต์ที่มีหน้าตาเหมือนเว็บไซต์ภาควิชาคอมพิวเตอร์ ที่หมายเลขไอพี 161.246.5.30 และเมื่อเครื่องเป้าหมายที่โดนโจมตีอยู่เข้าเว็บไซต์ด้วยการพิมพ์ชื่อเว็บไซต์ เครื่องเป้าหมายจะร้องขอหน้าเว็บปลอมจากเครื่องของผู้โจมตี แทนที่จะเป็นเว็บไซต์ของจริง ซึ่งการทดลองมีขั้นตอนดังต่อไปนี้

1) แก้ไขไฟล์ HTML ให้หน้าเว็บไซต์เป็นไปตามที่ต้องการ (ในตัวอย่างนี้ได้แก้ไขให้เหมือนเว็บไซต์ของภาควิชาวิศวกรรมคอมพิวเตอร์ และเพิ่มลายน้ำคำว่า Fake)
2) เปิดใช้งานเว็บเซิร์ฟเวอร์ของผู้โจมตี โดยใช้คำสั่ง $ service start apache2
3) แก้ไขไฟล์ etter.dns เพื่อจับคู่ชื่อกับที่อยู่ไอพีตามที่ต้องการ
4) เรียกใช้โปรแกรม Ettercap
5) เปิดใช้งานปลั๊กอิน dns_spoof



รูป 3.7 การใช้คำสั่ง ping ไปยังเว็บไซต์ที่ถูก DNS spoofing

รูป 3.8 เว็บไซต์ของภาควิชา ที่โดนปลอมแปลง

บทที่ 4
การดำเนินงานและผลการดำเนินงาน

4.1 การพัฒนาส่วนเสริมเบราเซอร์เพื่อป้องกันการโจมตี

เนื่องจากเบราเซอร์ Internet Explorer รุ่นที่เก่ากว่า 11 ไม่รองรับการใช้งาน HSTS เพราะฉะนั้น ในการพัฒนาส่วนเสริมนี้ จึงจะพัฒนาส่วนเสริมสำหรับเบราเซอร์ Internet Explorer เพื่อให้เบราเซอร์นี้สามารถใช้งาน HSTS ได้

4.1.1 รายละเอียดของส่วนเสริม

1) เป็นส่วนเสริมประเภท Browser Helper Objects (BHO)

2) พัฒนาเพื่อทดสอบใช้งานกับเบราเซอร์ Internet Explorer เวอร์ชั่น 10

3) ทดสอบบนระบบปฏิบัติการ Windows 7 64-Bit

4.1.2 ส่วนการทำงานของส่วนเสริม

แบ่งการทำงานได้ออกเป็น 5 การทำงานย่อย ดังนี้

1) ส่วนการทำงานพื้นฐานของ Browser Helper Objects

2) ส่วนตอบสนองอีเว้นท์ (Event) และควบคุมเบราเซอร์

3) ส่วนเชื่อมต่อกับเว็บไซต์

4) ส่วนจัดการและตรวจสอบรายชื่อเว็บไซต์ที่รองรับ HSTS

5) ส่วนตรวจสอบความถูกต้องของใบรับรอง

4.1.3 ขั้นตอนการทำงานของส่วนเสริม

1) รอรับอีเว้นท์จากเบราเซอร์

2) เมื่อได้รับอีเว้นท์ ให้รับ URL จากเบราเซอร์

3) ตรวจสอบความถูกต้องของใบรับรอง

4) ตรวจสอบการรองรับ HSTS ของเว็บไซต์

5) ทำการปิดกั้นการเข้าถึงหากใบรับรองไม่ถูกต้อง และเว็บไซต์รองรับ HSTS ซึ่งยังไม่หมดอายุ

4.1.4 ข้อจำกัดของส่วนเสริม

1) ใช้งานได้เฉพาะเบราเซอร์ Internet Explorer เวอร์ชั่น 4 ขึ้นไปเท่านั้น

2) ใช้เวลาในการทำงานในระยะเวลาหนึ่ง

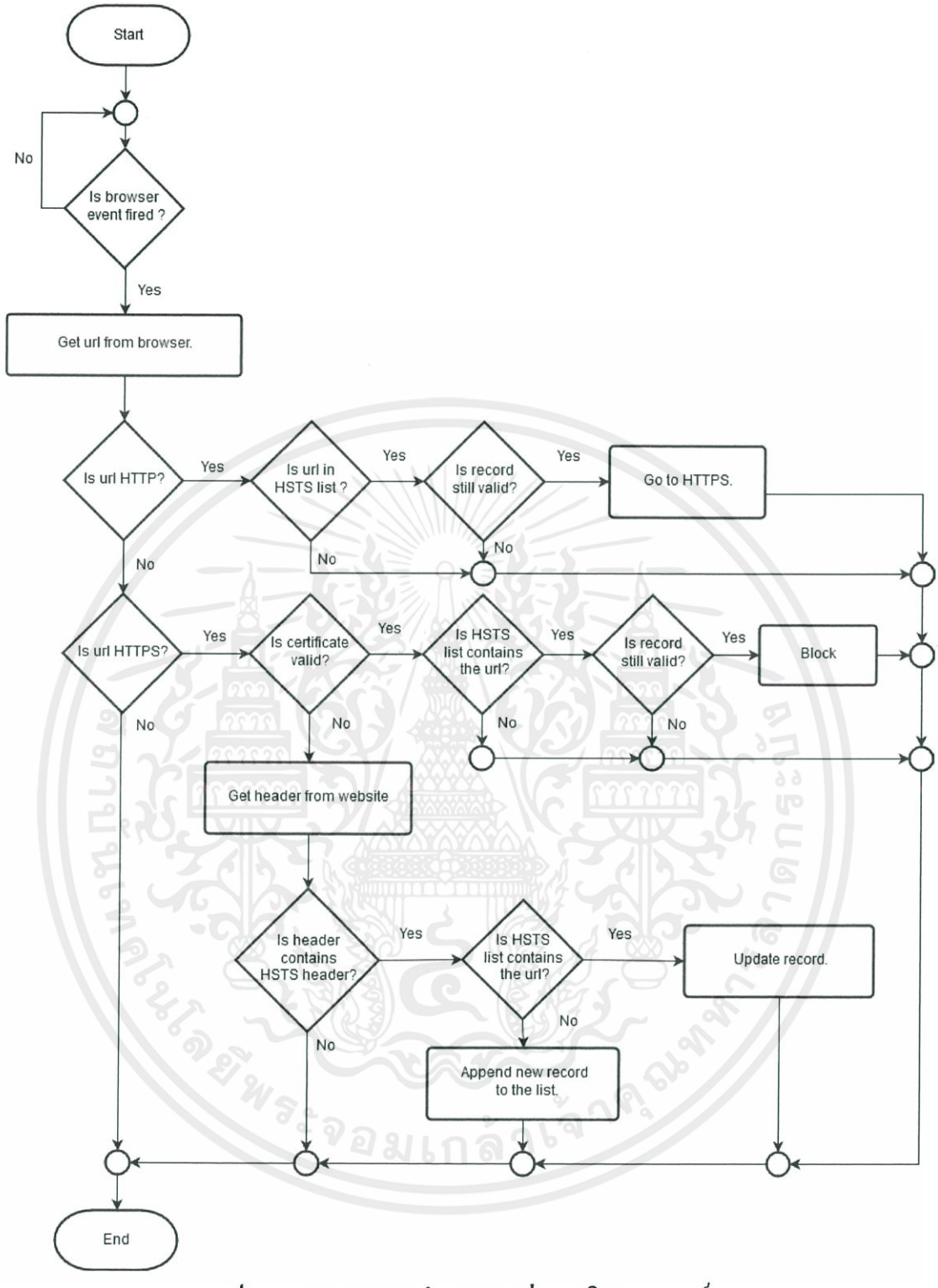
3) การตรวจสอบใบนับรองอาจไม่ครอบคลุมทุก CA

4.1.5 ภาษาและเครื่องมือที่ใช้พัฒนา

1) ใช้ภาษา C++

2) Microsoft Visual Studio Ultimate 2013

3) Oracle VM VirtualBox



รูป 4.1 แผนภาพการทำงานของส่วนเสริมเบราเซอร์

4.1.6 การสร้างส่วนพื้นฐานของ Browser Helper Objects

ในการเริ่มต้นสร้างโปรเจกต์ใหม่ หากพัฒนาด้วย Microsoft Visual Studio สามารถสร้างโดยใช้แม่แบบ ซึ่งเรียกว่า Active Template Library (ATL) ได้ จากนั้นเพิ่มคลาสใหม่ โดยเลือก ATL Simple Object และตั้งชื่อว่า IExtensionBHO

ส่วนแรกที่จำเป็นของ BHO คือการใช้งานอินเตอร์เฟซ (Interface) IObjectWithSite ซึ่งมีเมธอด (Method) SetSite ที่อำนวยความสะดวกในการสื่อสารกับ Internet Explorer และบอก BHO เมื่อต้องการจบการทำงาน โดย Internet Explorer นั้นจะเรียกใช้เมธอด SetSite สองครั้ง คือเมื่อสร้างการเชื่อมต่อ เบราเซอร์จะส่งอ็อบเจกต์ IUnkown ซึ่งเป็นตำแหน่งของส่วนควบคุมเบราเซอร์เพื่อให้ BHO เก็บไว้ และเมื่อปิดการทำงานเบราเซอร์ เบราเซอร์จะส่งค่า NULL มาเพื่อให้ BHO ปลดปล่อยอินเตอร์เฟซทั้งหมดที่ใช้งาน และตัดการเชื่อมต่อจากเบราเซอร์ ซึ่งในส่วนนี้ BHO ควรจะทำการเริ่มการ (Initialize) และจบการทำงาน (Uninitialize) ส่วนอื่น ๆ ที่จำเป็น เช่น การสร้างการเชื่อมต่อเพื่อรับอีเว้นท์ที่เบราเซอร์ส่งออกมา

ในการควบคุมเว็บเบราเซอร์ จำเป็นต้องประกาศใช้งาน shlguid.h เพื่อใช้งานอินเตอร์เฟซ IWebBrowser2 และอีเว้นท์ต่าง ๆ ของเบราเซอร์ จากนั้นจึงประกาศตัวแปรสำหรับเก็บเว็บไซต์ของเบราเซอร์

ตัวอย่าง 4.1 การประกาศของ BHO

```
#include <shlguid.h>
...
STDMETHOD(SetSite)(IUnknown *pUnkSite);
...
private:
 CComPtr<IWebBrowser2>  m_spWebBrowser;
```

ตัวอย่าง 4.2 โค้ดของเมธอด SetSite

```
STDMETHODIMP CIExtensionBHO::SetSite(IUnknown* pUnkSite)
{
 if (pUnkSite != NULL)
 {
  //Cache the pointer to IWebBrowser2.
  pUnkSite->QueryInterface(IID_IWebBrowser2,
  (void**)&m_spWebBrowser);
 }
   else
   {
  //Release cached pointers and other resources.
  m_spWebBrowser.Release();
 }

 //Return the base class implementation
   return IObjectWithSiteImpl<CHelloWorldBHO>::SetSite
(pUnkSite);
}
```

เมื่อทำการ build โปรเจกต์ Visual Studio จะทำการเพิ่ม CLSID ของ BHO ไปยัง registry เพื่อบ่งบอกว่าไฟล์ DLL เป็น BHO และให้ Internet Explorer เรียกใช้เมื่อเริ่มต้นการใช้งาน ซึ่ง CLSID ของ BHO สามารถพบได้ที่ไฟล์ IExtrension.idl

ตัวอย่าง 4.3 ส่วนของโค้ดที่ประกอบด้วย CLSID

```
importlib("stdole2.tlb");
[
 uuid(58FDE8E2-3613-48EF-9AA3-2C6C30151F71)
]
```

จากนั้นให้นำ CLSID ที่ระบุ มาทำการใส่ในไฟล์ IExtension.rgs เพื่อให้สามารถติดตั้ง และเพิ่มค่าใน registry ได้ ตามตัวอย่างด้านล่าง

ตัวอย่าง 4.4 ส่วนโค้ดสำหรับเพิ่มค่าใน registry

```
HKLM {
  NoRemove SOFTWARE {
    NoRemove Microsoft {
      NoRemove Windows {
        NoRemove CurrentVersion {
          NoRemove Explorer {
            NoRemove 'Browser Helper Objects' {
              ForceRemove '{58FDE8E2-3613-48EF-9AA3-
              2C6C30151F71}' =s 'IExtensionBHO' {
                val 'NoExplorer' =d '1'
              }
            }
          }
        }
      }
    }
  }
}
```

4.1.7 การตอบสนองอีเว้นท์และควบคุมเบราเซอร์

เมื่อเบราเซอร์มีการเปลี่ยนแปลงสถานะ จะมีการส่งอีเว้นท์ต่าง ๆ ออกมา สำหรับอินเตอร์เฟซ IWebBrowser2 จะมีอีเว้นท์ต่าง ๆ เช่น DownloadBegin, DownloadComplete, OnQuit เป็นต้น โดยส่วนเสริมที่จะสร้างขึ้นนี้ จะทำงานเมื่อผู้ใช้ต้องการไปยังเว็บไซต์อื่น และเบราเซอร์ทำการเปลี่ยนแปลง URL ปัจจุบัน จึงเลือกใช้งานอีเว้นท์ BeforeNavigate2

ในการรับอีเว้นท์ที่มาจากเบราเซอร์ BHO จำเป็นต้องประกาศใช้ expisid.h สำหรับประกาศอีเว้นท์ต่าง ๆ ของเบราเซอร์ และ สร้างเชื่อมต่อกับเบราเซอร์เพื่อทำการตอบสนองต่ออีเว้นท์เหล่านั้น โดยใช้อินเตอร์เฟซที่มีชื่อว่า IDispatch ซึ่งจะให้ข้อมูล Property และ Method ต่าง ๆ ของ อ็อบเจกต์ และในการใช้งานอีเว้นท์ BeforeNavigate2 ตามเอกสารจะมี 7 พารามิเตอร์ ซึ่งพารามิเตอร์เหล่านี้จะถูกส่งต่อไปยังเมธอด IDispatch::Invoke

จากนั้นประกาศใช้งาน exdispid.h เพื่อกำหนดค่าอีเว้นท์ของเบราเซอร์ และเพิ่มการประกาศคลาส IDispEventImpl เพื่อให้ง่ายและสะดวกในการควบคุมอีเว้นท์ และประกาศอีเว้น

ตัวอย่าง 4.5 โค้ดสำหรับควบคุมอีเว้นท์ BeforeNavigate2

```
void STDMETHODCALLTYPE CIExtensionBHO::OnBeforeNavigate
(IDispatch *pDisp, VARIANT *pvarURL, VARIANT *vFlags,
VARIANT *vTargetFrameName, VARIANT *vPostData,
VARIANT * vHeaders, VARIANT *vCancel){

 HWND hwnd;
 HRESULT hr =m_spWebBrowser->
 get_HWND((LONG_PTR*)&hwnd);

    if (SUCCEEDED(hr))
    {
  //Do things herebefore the page is navigate.
    }
```

4.1.8 การเชื่อมต่อเพื่อรับเฮดเดอร์จากเว็บไซต์

ในการสื่อสารกับเว็บเซิร์ฟเวอร์ จะใช้ไลบรารี่ libcurl เพื่อทำการรับค่าเฮดเดอร์ของเว็บไซต์ที่ต้องการ เพื่อนำมาตรวจสอบการรองรับ HSTS

เนื่องจากหลาย ๆ เว็บไซต์ จะส่งเนื้อหาเว็บไซต์แตกต่างกันไปตาม User-agent ที่ทำการร้องขอ และบางเว็บไซต์ปฏิเสธที่จะให้บริการกับบาง User-agent ส่วนเสริมนี้จึงได้กำหนดให้ใช้ User-agent ที่ได้รับการยอมรับในหลาย ๆ เว็บไซต์ โดยใช้เป็น Mozilla / 5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko / 20100101 Firefox / 40.1 และเมื่อทำการร้องขอ แล้วได้รับเฮดเดอร์ จะมีการเรียกใช้งาน callback function ที่ได้กำหนดไว้

ตัวอย่าง 4.6 โค้ดสำหรับรับค่าเฮดเดอร์

```
static size_t header_callback(char *buffer, size_t size,
size_t nitems, void *userdata)
{
 header +=buffer;
 return nitems*size;
}

string getHeader(const char *url){
 CURL *curl;
 CURLcode res;
 curl =curl_easy_init();
 if (curl){
  curl_easy_setopt(curl, CURLOPT_URL, url);
  curl_easy_setopt(curl, CURLOPT_HEADERFUNCTION,
  header_callback);
  curl_easy_setopt(curl, CURLOPT_NOBODY, 1L);
  curl_easy_setopt(curl, CURLOPT_USERAGENT, "Mozilla
  /5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko /
  20100101 Firefox /40.1");
  res =curl_easy_perform(curl);

  /*Check for errors */
  if (res !=CURLE_OK) {
      fprintf(stderr, "curl_easy_perform() failed:
      %s\n", curl_easy_strerror(res));
    }
  /*always cleanup */
```

4.1.9 การจัดการและตรวจสอบรายชื่อเว็บไซต์ที่รองรับ HSTS

เมื่อได้รับเฮดเดอร์มาแล้ว จะทำการตรวจสอบเฮดเดอร์ดังกล่าวว่า เว็บไซต์นั้นรองรับ HSTS หรือไม่ โดยการค้นหาว่ามีคำว่า strict-transport-security อยู่หรือไม่ และเนื่องจากเฮดเดอร์นั้นไม่ได้จำกัดว่าต้องเป็นตัวอักษรพิมพ์เล็กหรือพิมพ์ใหญ่ จึงต้องค้นหาแบบไม่สนใจว่าเป็นตัวอักษรพิมพ์เล็กหรือพิมพ์ใหญ่เช่นกัน

ตัวอย่าง 4.7 โค้ดสำหรับตรวจสอบรองรับ HSTS ของเว็บไซต์โดยใช้เฮดเดอร์

```
bool find_ignore_case(const string & strHaystack, const
string & strNeedle)
{
 auto it =std::search(
  strHaystack.begin(), strHaystack.end(),
  strNeedle.begin(), strNeedle.end(),[]
  (char ch1, char ch2){
           return std::toupper(ch1)==std::toupper(ch2);}
 );
    return (it !=strHaystack.end());
}

bool isSupportHSTS(string header){
 if (find_ignore_case(header, "strict-transport-
 security")){
  return true;
 }
```

ถ้าหากเว็บไซต์มีการรองรับ HSTS จะทำการตรวจสอบว่าเว็บไซต์ดังกล่าว มีอยู่ในไฟล์ที่เก็บรายชื่อแล้วหรือไม่ ซึ่งถ้าหากยังไม่มี จะทำการอ่านค่าอายุในการรองรับ (max-age) แล้วเพิ่มเว็บไซต์นั้นลงในไฟล์พร้อมกับเวลาที่หมดอายุ ซึ่งคือเวลาปัจจุบันรวมกับอายุในการรองรับ แต่ถ้าหากมีแล้วจะทำการอัพเดตวันหมดอายุให้กับเว็บไซต์

ตัวอย่าง 4.8 โค้ดสำหรับเพิ่มเว็บไซต์ที่รองรับ HSTS ลงในไฟล์

```
int add_Record(const char *file_path, char *domain, int
age)
{
 FILE *fp;
 fp =fopen(file_path, "a");
 if (fp ==NULL)return F_OPEN_ERR;
 else
 {
  time_t newDate =time(0)+age;
  fprintf(fp, "%s,%d\n", domain, newDate);
 }
 fclose(fp);
```

ตัวอย่าง 4.9 โค้ดสำหรับอัพเดตวันหมดอายุเว็บไซต์ที่รองรับ HSTS

```
int delete_Record(const char* file_path, char* domain)
{
 if (isExist(file_path, domain)) {
  const int listsize =1024, bufsize =128;
  char buf[bufsize]={ NULL },
  list[listsize][bufsize]={ NULL };
     FILE *fp=fopen(file_path, "r");
  if (fp ==NULL)return F_OPEN_ERR;
  else {
   int i =0;
   while (fgets(buf, bufsize, fp)!=NULL){
        if (strstr(buf, domain)==0){
             strcpy(list[i], buf);
             i++;
        }
        }
        fclose(fp);
  }
  fp =fopen(file_path, "w");
  for (int i =0; i < listsize; i++){
       if (list[i][0]!='\0')
   fprintf(fp, "%s", list[i]);
      else break;
  }
  fclose(fp);
 }
 else {
  return ERR_NOT_FOUND; //record is not exist.
 }
 return SUCCEED;
}

int update_Record(const char *file_path, char *domain,
int age)
{
 if (delete_Record(file_path, domain)==SUCCEED){
  if (add_Record(file_path, domain, age)==SUCCEED){
      return SUCCEED;
 }
```

ตัวอย่าง 4.10 โค้ดสำหรับตรวจสอบการหมดอายุของ HSTS

```
bool isValid(const char *file_path, char *domain){
 FILE *fp;
  fp =fopen(file_path, "r");
  if (fp ==NULL)return false; //fopen error
  else {
   char buf[128];
   while (fgets(buf, 128, fp)!=NULL){
    if (strstr(buf, domain)!=0){
     char *dateIndex =strchr(buf, ',')+1;
     int date =atoi(dateIndex);
            if (date -time(0)> 0){
             fclose(fp);
             return true;
            }
      }
     }
  }
 fclose(fp);
 return false;
```

4.1.10 การตรวจสอบความถูกต้องของใบรับรอง

การตรวจสอบใบรับรองของเว็บไซต์ต่าง ๆ จะใช้ไลบรารี่ของ OpenSSL และรายการใบรับรองของ CA ที่น่าเชื่อถือและมีความเป็นปัจจุบันมากที่สุด จากนั้นทำการเชื่อมต่อกับเว็บไซต์ที่พอร์ต 443 เพื่อทำการนำใบรับรองของเว็บไซต์ มาตรวจสอบกับรายชื่อของ CA ที่มีอยู่ ถ้าหากใบรับรองไม่ถูกต้องหรือหมดอายุ และเว็บไซต์นั้นรองรับ HSTS ส่วนเสริมจะทำการปิดกั้นไม่ให้สามารถเข้าถึงเว็บไซต์ได้

โดยการตรวจสอบนั้น จะตรวจสอบโดยการตรวจสอบสถานะ OCSP จากเว็บไซต์ด้วยการทำ OCSP Stapling ซึ่งถ้าหากเว็บไซต์นั้นไม่รองรับ จะทำการตรวจสอบด้วยลำดับของ CA และ CRL ว่าใบรับรองดังกล่าว น่าเชื่อถือหรือไม่

ตัวอย่าง 4.11 โค้ดสำหรับตรวจสอบความถูกต้องของใบรับรองด้วยลำดับของ CA

```
int verify_cert(const char *cert_path, const char *
crl_path, string str_url){
 if (str_url.substr(0, 8)=="https://"){
  int init_result =(init_ssl(cert_path, crl_path));
  if (init_result ==OK){
   char c_domain[1024];
       sscanf(str_url.c_str(), "https://%[^/]",
   c_domain);
       BIO_set_conn_hostname(bio, c_domain);
       BIO_set_conn_port(bio, "443");
       if (BIO_do_connect(bio)<=0){
        freeCTX();
        return ERR_FAIL_TO_CONNECT;
       }
       else {
        int verify_result =SSL_get_verify_result(ssl);
        if (stap_result ==V_OCSP_CERTSTATUS_GOOD){
            freeCTX();
            return VALID_CERT;
        }
          else if (stap_result ==V_OCSP_CERTSTATUS_REVOKED
          || stap_result ==V_OCSP_CERTSTATUS_UNKNOWN){
            freeCTX();
            return INVALID_CERT;
        }
        else {
            freeCTX();
            if (verify_result !=X509_V_OK){
             return INVALID_CERT;
            }
           else if (verify_result ==X509_V_OK){
            return VALID_CERT;
           }
        }
       }
 }
 else {
      freeCTX();
      return init_result;
 }
}
else {
 freeCTX();
 return INVALID_URL;
```

4.1.11 การทำงานรวมของส่วนเสริม

เมื่อเขียนเมธอดต่าง ๆ สำหรับการทำงานของส่วนเสริมครบแล้วให้เรียกใช้ในส่วนที่ควบคุมอีเว้นท์ BeforeNavigate2 ตามที่ได้กำหนดขั้นตอนการทำงานไว้ โดยเริ่มจากตรวจสอบว่าเว็บไซต์นั้น เป็นเว็บแบบ HTTP หรือ HTTPS ซึ่งหากเป็นแบบ HTTP ให้ทำการตรวจสอบว่าเว็บไซต์ดังกล่าวมีอยู่ในรายการที่รองรับ HSTS หรือไม่ ถ้ารองรับให้เปลี่ยนเป็นแบบ HTTPS จากนั้นจึงตรวจสอบใบรับรองของเว็บไซต์ ว่ามีความน่าเชื่อถือหรือไม่ โดยถ้าหากมีความน่าเชื่อถือและรองรับ HSTS ให้ทำการอัพเดตวันหมดอายุของเว็บไซต์ ในรายการเว็บไซต์ที่รองรับ HSTS แต่ถ้าหากใบรับรองนั้นไม่น่าเชื่อถือ และมีอยู่ในรายการเว็บไซต์ที่รองรับ HSTS ให้ทำการปิดกั้นไม่ให้เข้าถึงเว็บไซต์ดังกล่าวได้ โดยเปลี่ยนเส้นทางไปยังหน้าที่แสดงคำเตือนและรายละเอียดในการปิดกั้น

ตัวอย่าง 4.12 โค้ดการทำงานรวมของส่วนเสริม

```
String str_url(_com_util::ConvertBSTRToString(pvarURL
->bstrVal));
if (vFlags->intVal != 0) {
 if (str_url.substr(0, 7)=="http://"){
  char c_domain[1024];
  sscanf(str_url.c_str(), "http://%[^/]", c_domain);
  if(isValid(hsts_list_path.c_str(), c_domain)) {
   char ch_dir[1024];
   string str_domain(ch_dir);
   str_url ="https://"+str_domain;
       BSTR bstr_url =
       _com_util::ConvertStringToBSTR(str_url.c_str());
   vCancel->boolVal =VARIANT_TRUE;
   m_spWebBrowser->Stop();
   VARIANT vFlags ={ 0 },
    vTargetFrameName ={ 0 },
    vPostData ={ 0 },
    vHeaders ={ 0 };
   m_spWebBrowser->Navigate(bstr_url, &vFlags,
   &vTargetFrameName, &vPostData, &vHeaders);
  }
 }
 else if (str_url.substr(0, 8)=="https://") {
  char c_domain[1024];
  sscanf(str_url.c_str(), "https://%[^/]", c_domain);
  int result=
```

```
  if (isExist(hsts_list_path.c_str(), c_domain))
   update_Record(hsts_list_path.c_str(),
   c_domain, age);
  }
  else {
   add_Record(hsts_list_path.c_str(),
   c_domain, age);
  }
 }
}
else if (result == ERR_NOT_FOUND_CERT) {
 MessageBox(hwnd, L"Fail to load root CA list.",
 L"Error", MB_ICONERROR | MB_OK);
}
else {
 if (isValid(hsts_list_path.c_str(), c_domain)) {
 vCancel->boolVal = VARIANT_TRUE;
 m_spWebBrowser->Stop();
 VARIANT vFlags = { 0 },
  vTargetFrameName = { 0 },
  vPostData = { 0 },
  vHeaders = { 0 };
 m_spWebBrowser->Navigate(block_page_path,
 &vFlags, &vTargetFrameName, &vPostData,
 &vHeaders);
 }
```

4.2 การติดตั้งส่วนเสริมเบราเซอร์

หลังจากที่ได้พัฒนาส่วนเสริมนี้จนเสร็จสมบูรณ์ เมื่อต้องการที่จะติดตั้ง ให้ทำการสร้างโปรเจกต์ จะได้ไฟล์ IExtension.dll (Dynamic linking library) ซึ่งนำไปติดตั้งบนเครื่องอื่น ๆ ได้ โดยมีขั้นตอนดังนี้

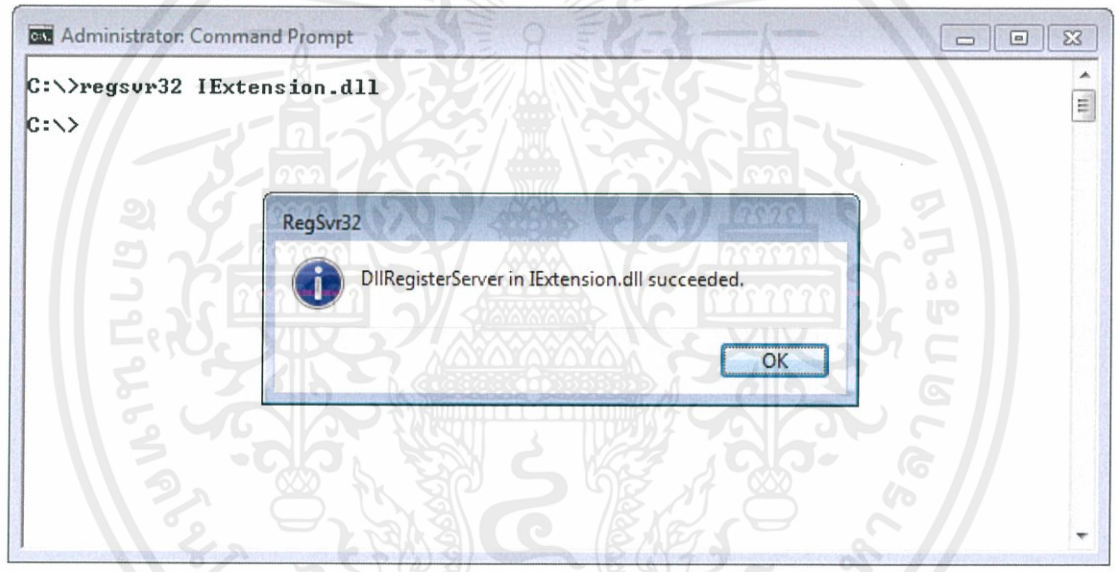
1) นำไฟล์ IExtension.dll ที่ได้ไปวางบนเครื่องที่ต้องการติดตั้ง

2) เรียก Command prompt โดยเลือก Run as administrator

3) เปลี่ยน directory ไปยังที่ที่วางไฟล์ไว้

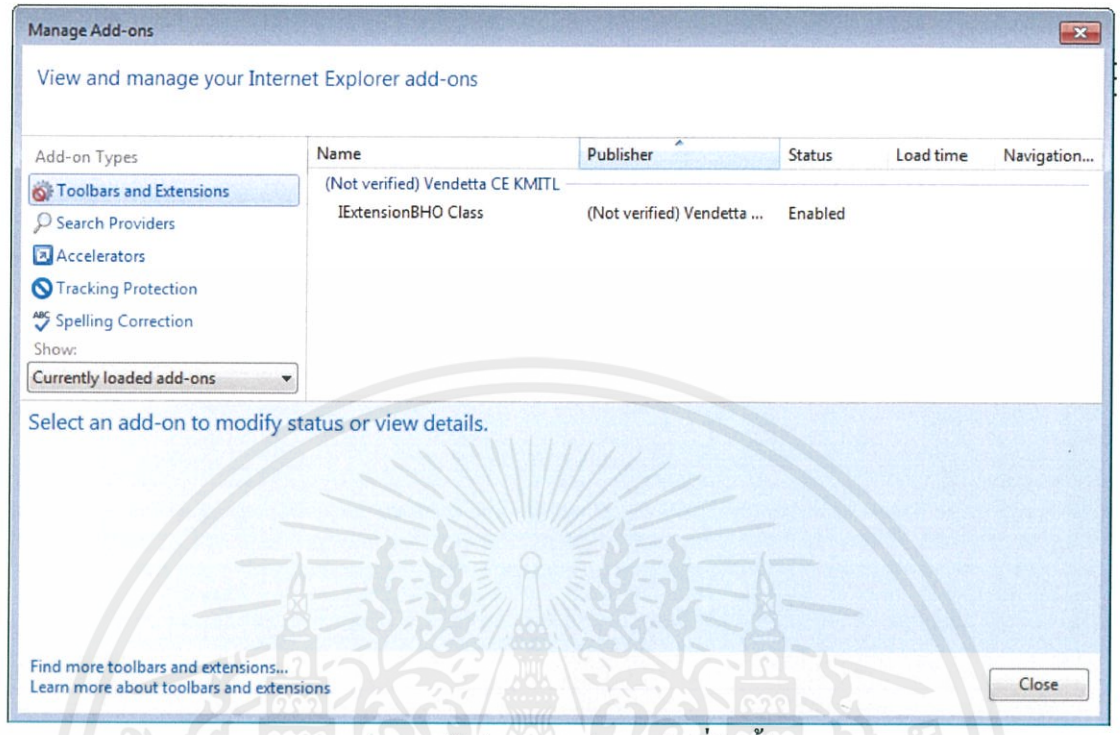
4) ใช้คำสั่ง regsvr32 IExtension.dll เพื่อทำการใส่ค่า CLSID ของ ส่วนเสริมลงใน Registry ตามที่ได้กำหนดไว้ในไฟล์ IExtension.rgs

และเมื่อต้องการถอนการติดตั้ง ให้เปลี่ยนคำสั่งในข้อ 3 เป็น regsvr -u IExtension.dll แทน



รูป 4.2 หน้าต่างแสดงผลลัพธ์เมื่อติดตั้งส่วนเสริมสำเร็จ

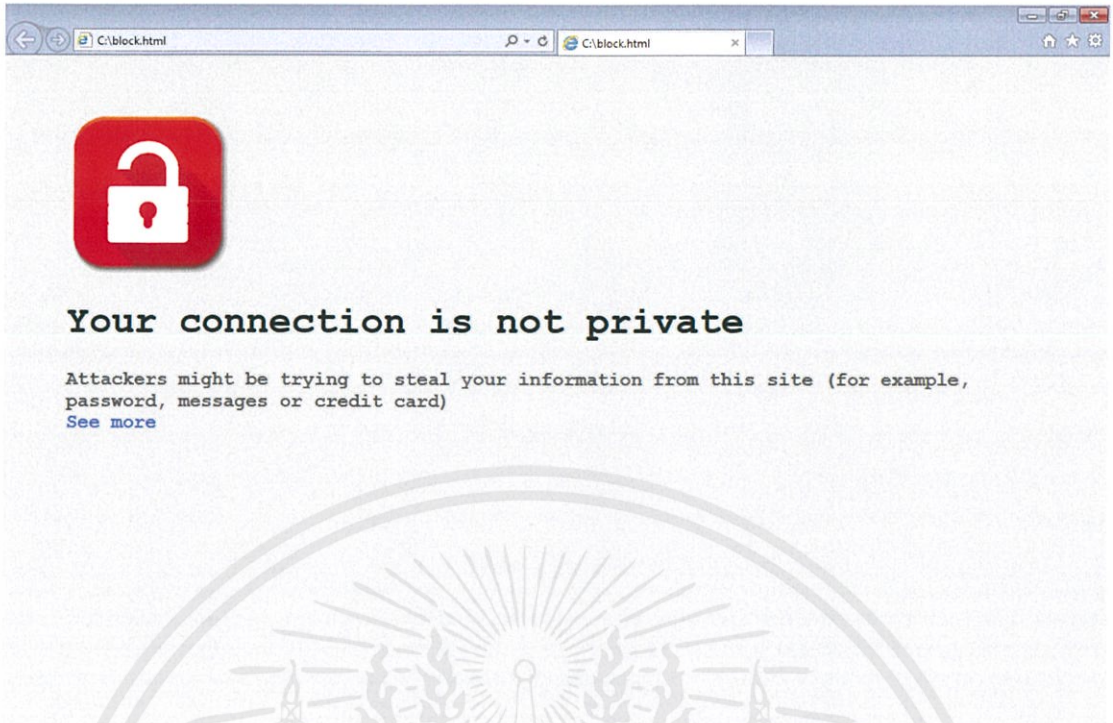
เมื่อติดตั้งสำเร็จ สามารถตรวจสอบได้โดยเลือกเมนู Tools > Manage add-ons



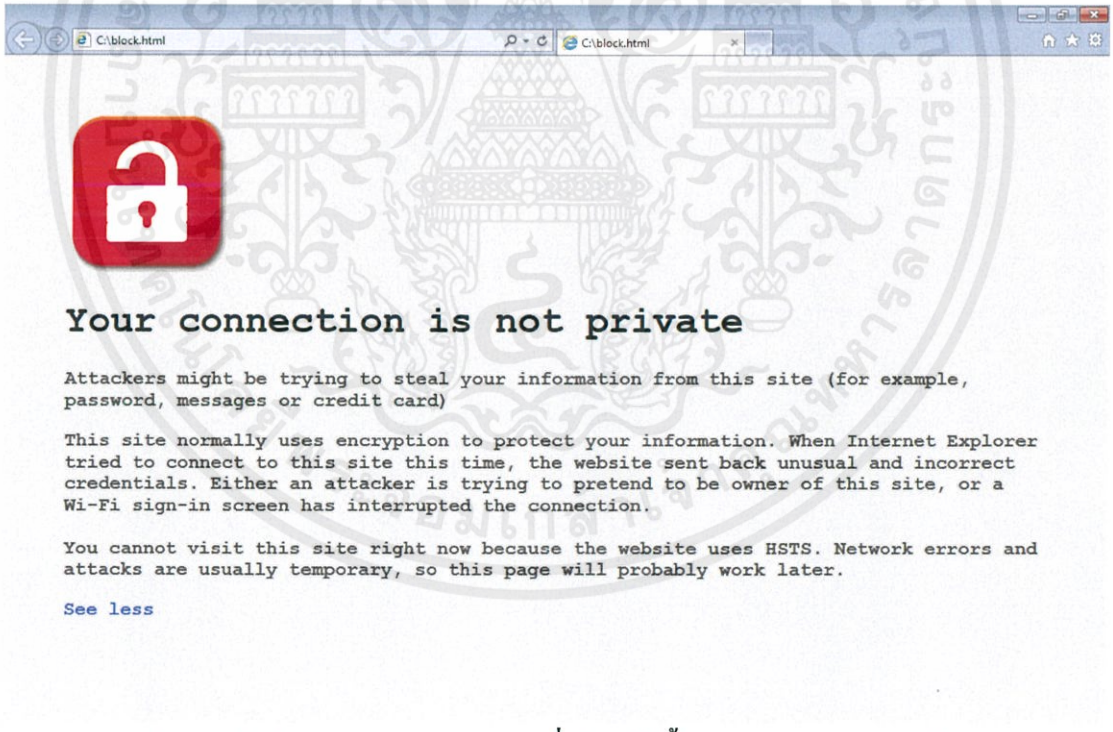
รูป 4.3 หน้าต่างแสดงส่วนเสริมที่ติดตั้ง

4.3 ผลการทำงานของส่วนเสริม

เมื่อทดสอบโจมตี หรือเข้าเว็บไซต์ที่ใบรับรองไม่ถูกต้อง เบราเซอร์จะทำการเปลี่ยนเส้นทางและแสดงผลเป็นเนื้อหาไฟล์ที่กำหนดไว้ ดังรูป

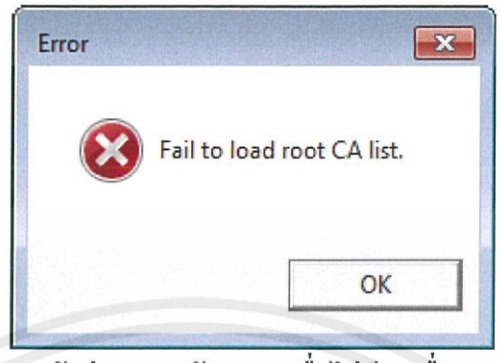


รูป 4.4 หน้าต่าง Internet Explorer เมื่อโดนปิดกั้น



รูป 4.5 หน้าต่าง Internet Explorer เมื่อโดนปิดกั้นและกดแสดงรายละเอียด

เมื่อส่วนเสริมมีข้อพลาด เกิดขึ้นจากไม่มีรายชื่อ CA อยู่ในที่ที่กำหนดไว้ จะแสดงหน้าต่างโต้ตอบแสดงข้อผิดพลาด ดังรูป



รูป 4.6 หน้าต่างแสดงข้อพลาดเมื่อไม่มีรายชื่อ CA

4.4 ข้อแนะนำสำหรับผู้ใช้งาน

4.1.1 ข้อปฏิบัติสำหรับเซิฟเวอร์

4.1.1.1 ใบรับรอง และกุญแจส่วนตัว

1) ใช้กุญแจส่วนตัวขนาด 2048 บิต แต่ถ้าหากคาดว่าต้องใช้ขนาดมากกว่านี้ ควรพิจารณาใช้กุญแจประเภท ECDSA ที่มีประสิทธิภาพมากกว่าแทน
2) เก็บรักษากุญแจส่วนตัว ไม่ให้ถูกล่วงรู้ได้
3)ชื่อที่ลงทะเบียนในใบรับรอง ควรครอบคลุมชื่อทั้งหมดในเว็บไซต์
4) ใช้ใบรับรองจาก CA ที่น่าเชื่อถือเท่านั้น

4.1.1.2 การตั้งค่าการใช้งาน

1) ระบุลำดับของ CA ให้ครบและถูกต้อง
2) ใช้โพรโทคอลที่ปลอดภัยเท่านั้น โดยมีโพรโทคอลต่าง ๆ โดย SSL v2 เป็นโพรโทคอลที่ไม่ปลอดภัย และห้ามใช้โดยเด็ดขาด, SSL v3 เป็นโพรโทคอลที่ค่อนข้างเก่าขาดคุณลักษณะของกุญแจบางประเภท จึงไม่ควรใช้หากไม่จำเป็น, TLS 1.0 ยังคงปลอดภัยค่อนข้างมากเมื่อใช้กับโพรโทคอล HTTP และตั้งค่าอย่างระมัดระวัง ส่วน TLS 1.1 และ 1.2 ยังไม่พบปัญหาเรื่องความปลอดภัย
3) ควบคุมให้รองรับเฉพาะกระบวนการยืนยันตัวตนที่ปลอดภัยเท่านั้น เพราะหลังจาก SSL v3 เซิฟเวอร์จะเลือกวิธีที่ดีที่สุดที่ไคลเอนต์รองรับ

4.1.1.3 การออกแบบแอปพลิเคชัน

1) เข้ารหัสเว็บไซต์ทั้งหมด

2) หลีกเลี่ยงการใช้เนื้อหาแบบผสม

3) ใช้ข้อมูลจากแหล่งภายนอกที่น่าเชื่อถือ

4) ทำให้คุ้กกี้มีความปลอดภัยโดยการเข้ารหัสลับ

5) ใช้งาน HTTP Strict Transport Security กับเว็บไซต์

4.1.1.4 การตรวจสอบ

ใช้เครื่องมือในการประเมินที่ครอบคลุมโพรโทคอล SSL/TLS เพื่อตรวจสอบการตั้งค่าความปลอดภัย และมีการตรวจสอบความปลอดภัยอยู่เป็นระยะ ๆ

4.1.2 ข้อปฏิบัติสำหรับไคลเอนต์

1) ใช้เบราเซอร์ที่ทันสมัยตลอดเวลา

2) หลีกเลี่ยงการใส่ข้อมูลสำคัญ บนโพรโทคอลที่ไม่เข้ารหัส เช่น HTTP เป็นต้น

3) มีสติ และระมัดระวังในการใช้งานอินเทอร์เน็ต

4) ติดตามข่าวสารที่เกี่ยวข้องกับการยกเลิกใบรับรอง

5) ตรวจสอบใบรับรองด้วยตัวเองทุกครั้งที่ใส่ข้อมูลที่มีความละเอียดอ่อน

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุป

ปัญหาการโจมตีจากการคนคั่นกลางเป็นปัญหาที่เกิดขึ้นจากการติดต่อสื่อสารผ่านทางอินเทอร์เน็ตโดยใช้ช่องโหว่ของโพรโทคอลต่าง ๆ ทำให้ผู้ใช้งานถูกโจรกรรมข้อมูลอันเป็นความลับไปได้ เมื่อเรารู้ถึงช่องทางที่ผู้โจมตีใช้ เราก็สามารถที่จะป้องกันได้

การโจมตีจากคนคั่นกลางสามารถโจมตีได้หลากหลายรูปแบบและก่อให้เกิดความเสียหายแก่ข้อมูลเป็นอย่างมาก ในการป้องกันการโจมตีจากคนคั่นกลางนั้น จำเป็นต้องป้องกันจากทั้งสองฝั่งจึงจะป้องกันได้อย่างมีประสิทธิภาพ

ในโครงงานนี้ได้ศึกษาการโจมตีจากคนคั่นกลางด้วยการเข้ารหัสลับ ด้วยวิธีการต่าง ๆ เพื่อนำไปสู่การสร้างโปรแกรมที่เป็นส่วนเสริมของเว็บเบราเซอร์ Internet Explorer ซึ่งเป็นโปรแกรมที่พัฒนามาจากภาษา C++ โดยมีการออกแบบมาเพื่อป้องกันการโจมตีจากคนคั่นกลาง ในรูปแบบต่าง ๆ เช่น การใช้ใบรับรองปลอมหรือ DNS spoofing เป็นต้น โดยส่วนเสริมที่พัฒนาขึ้นนี้สามารถนำไปใช้งานได้จริง และยังสามารถเป็นต้นแบบเพื่อพัฒนาต่อยอดให้มีประสิทธิภาพสูงขึ้นได้

5.2 ปัญหาและอุปสรรค

1) การค้นคว้าศึกษาข้อมูลทำได้ยาก เนื่องจากไม่ค่อยมีคนพัฒนาส่วนเสริมของเบราเซอร์ Internet Explorer และไลบรารี่ทำความเข้าใจได้ยาก
2) Virtual machine ที่ใช้ทำการทดลองเป็นแบบไม่เสียค่าใช้งาน จึงใช้ทรัพยากรได้อย่างจำกัด ส่งผลให้เกิดความลำบากในการทำงาน
3) ส่วนเสริมที่พัฒนาขึ้นทำงานได้ช้ากว่าที่คาดหวังไว้

5.3 แนวทางการแก้ไข

1) ใช้เวลาในการค้นคว้าศึกษาข้อมูลให้มากขึ้น และทดลองเพิ่มเติม
2) ทำการพัฒนาส่วนเสริมและทดสอบบนเครื่องจริง ก่อนที่จะนำไปทดสอบใน Virtual machine
3) ปรับปรุงขั้นตอนและกระบวนการในการทำงานและตรวจสอบการรองรับ HSTS และความถูกต้องของใบรับรอง ของส่วนเสริมให้สามารถทำงานได้ดีขึ้น

5.4 แนวทางการพัฒนาต่อ

1) สามารถพัฒนาส่วนเสริมเพื่อให้ส่วนเสริมทำงานได้เร็วขึ้น และใช้หน่วยความจำน้อยลง
2) ปรับปรุงการทำงานของส่วนจัดการรายชื่อเว็บไซต์ที่รองรับ HSTS ให้สามารถอ่านและแก้ไขได้รวดเร็วขึ้น
3) สามารถเพิ่มการจัดการ HPKP เข้ามาในส่วนเสริม หรือปรับปรุงให้ส่วนเสริมสามารถตรวจสอบปัจจัยอื่นที่อาจทำให้ไม่ปลอดภัย เช่น ขนาดของกุญแจของใบรับรอง หรือกระบวนการสร้างกุญแจ เป็นต้น

บรรณานุกรม

Cisco Systems. 2004. **Public Key Infrastrucure Certificte Revocation List versus Online Certificate Status Protocol.** [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/public-key-infrastructure-pki/product_data_sheet0900aecd80313df4.pdf

Dino Esposito. 1999. **Browser Helper Objects: The Browser the Way You Want It.** [Online]. Available: https://msdn.microsoft.com/en-us/library/bb250436(v=vs.85).aspx

Ford AntiTrust (นามแฝง). 2016. **แนะนำ HTTP Public Key Pinning (HPKP) เพื่อป้องกันปัญหา Certification Authority ออก TLS Certificate ซ้ำซ้อน.** [Online]. Available: https://www.blognone.com/node/82129

Gibson Research Corporation. 2014. **Security Certificate Revocation Awareness.** [Online]. Available: https://www.grc.com/revocation/ocsp-must-staple.htm

Holly Lynne McKinley. 2003. **SSL and TLS: A Beginners' Guide.** [Online]. Available: https://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029

Ivan Ristić. 2013. **OpenSSL Cookbook.** [Online]. Available: https://www.feistyduck.com/library/openssl-cookbook/

Kenneth Ballard. 2012. **Secure programming with OpenSSL API.** [Online]. Available: https://www.ibm.com/developerworks/library/l-openssl/l-openssl-pdf.pdf

Ondrej David. 2013. **Sniffing in a switched environment and wireshark.** [Online]. Available: http://icanhazsecurity.ondrejdavid.com/2013/05/pentesting101-7-sniffing-in-switched.html

Ricky Donato. 2014**. Certificate Revocation (CRL vs OCSP).** [Online]. Available: https://www.fir3net.com/Security/Concepts-and-Terminology/certificate-revocation.html

Subodh Gangan. 2015. **A Review of Man-in-the-Middle Attacks.** [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1504/1504.02115.pdf