การออกแบบการยืนยันตัวตนด้วยรหัสผ่านแบบวันไทม์โดยใช้แคปช่า

A Design of One Time Password Authentication Using CAPTCHA

ณัฐวุฒิ ศรีวิบูลย์ (Nattavut Sriwiboon)* และ สมนึก พ่วงพรพิทักษ์ (Somnuk Puangpronpitag)*

บทคัดย่อ

รหัสผ่านเป็นข้อมูลที่นิยมใช้ระบุตัวตนในการเข้าใช้งานเว็บไซต์ โดยโพรโทคอล Secure Socket Layer (SSL) นำมาใช้ในการป้องกันการดักจับข้อมูลในการใช้งานเว็บไซต์อย่างแพร่หลาย อย่างไรก็ตามมีการใช้เทคนิควิธีการในการโจมตี SSL หลายวิธี เช่น SSL Strip, SSL decode เป็นต้น ดังนั้นการอาศัย SSL เพียงอย่างเดียว จึงไม่สามารถป้องกันการดักจับรหัสผ่านได้ ในงานวิจัยนี้จึงนำเสนอการออกแบบวิธีป้องกันการดักจับรหัสผ่านโดยการใช้รหัสผ่านแบบ OTP (One Time Password) โดยอาศัยการเข้ารหัสด้วยกุญแจที่ได้จาก CAPTCHA และได้พัฒนาระบบต้นแบบเพื่อทดสอบแสดงให้เห็นว่าระบบที่ออกแบบมีประสิทธิภาพและประสิทธิผลในการป้องกันการโจมตี

คำสำคัญ: เอสเอสแอล โอทีพี แคปช่า การเข้ารหัส

Abstract

Password is the most popular authentication factor used for web application login. To protect the web application against data eavesdropping, Secure Socket Layer (SSL) protocol has been widely utilized. However, there have been several techniques (such as SSL strip, SSL decode) to attack SSL. Therefore, even with SSL, the password may be snooped. In this paper, we propose a design to prevent against the password eavesdropping by using an OTP (One Time Password) together with a CAPTCHA-key cryptographic technique. An implementation of the design has also been prototyped and experimented on. The experimental results have shown an effectiveness and efficiency of the design.

Keyword: SSL, OTP, CAPTCHA, cryptographic.

1. บทนำ

การให้บริการเว็บไซต์ในด้านต่าง ๆ เช่นระบบทะเบียนในสถานศึกษา รวมถึงธุรกรรมผ่านเว็บไซต์อย่าง เช่น ธนาคารอิเล็กทรอนิกส์ (e-Banking) มีการให้บริการอย่างแพร่หลาย ซึ่งผู้ให้บริการจะต้องคำนึงถึงความปลอดภัยของข้อมูลของผู้ใช้ที่เป็นข้อมูลลับ เช่น รหัสผ่านในการเข้าใช้งานระบบ โดยในปัจจุบัน SSL เป็นโพรโทคอลที่นำมาใช้ในการเข้ารหัสข้อมูลก่อนส่งไปที่เซิร์ฟเวอร์เพื่อป้องกันการดักจับข้อมูลระหว่างการสื่อสาร โดยในการใช้งาน SSL ผู้ให้บริการเว็บไซต์จำเป็นที่จะต้องขอจดทะเบียนกับ Certificate Authority (CA) ที่ให้บริการออกใบประกาศอิเล็กทรอนิกส์ (Certificate) ซึ่งมีค่าใช้จ่ายในการดำเนินการ จึงมีการนำ Self-signed SSL certificates ที่สามารถสร้าง Certificate ขึ้นเองมาใช้งานกับเว็บไซต์ จากการค้นคว้างานวิจัยนี้พบว่าการใช้งาน SSL และ Self-signed SSL สามารถโจมตีดักจับข้อมูลได้ด้วยวิธีการแทรกกลางการสื่อสาร (Man In The Middle or Monkey In The Middle: MITM) [1] โดยอาศัยเครื่องมือต่าง ๆ เช่น Cain & Abel [2] SSL Strip [3] และ Ettercap [4] เป็นต้น รวมถึง Tamper Data Plugin [5] ที่มีคุณสมบัติในการตรวจสอบและแก้ไขค่าพารามิเตอร์แบบ Post ที่ส่งไปที่เซิร์ฟเวอร์ ซึ่ง

* ภาควิชาเทคโนโลยีสารสนเทศ คณะวิทยาการสารสนเทศ มหาวิทยาลัยมหาสารคาม

ผู้โจมตีสามารถใช้เครื่องมือดังกล่าวโดยไม่จำเป็นต้องมีความรู้ในด้านความปลอดภัยของเทคโนโลยีเครือข่ายก็สามารถโจมตีเว็บไซต์ที่ใช้ SSL และ Self-signed SSL ได้ โดยสามารถศึกษาวิธีการโจมตีได้จากแหล่งข้อมูลจำนวนมาก เช่น [6], [7], [8] เป็นต้น

จากปัญหาการโจมตีด้วยวิธี MITM ที่ผู้ให้บริการเว็บไซต์อาศัยการเข้ารหัสข้อมูลในการสื่อสารโดยใช้ SSL ซึ่งปัญหาของ SSL และ Self-signed SSL ยังมีปัญหาในการถูกโจมตีด้วยวิธีต่าง ๆ และงานวิจัยก่อนหน้านี้ [9], [10], [11] ที่เสนอวิธีการแก้ไขเพื่อป้องกันการโจมตีด้วยวิธี MITM และการป้องกันการโจมตีรหัสผ่านที่ยังไม่มีประสิทธิภาพ งานวิจัยนี้จึงได้เสนอวิธีการป้องกันการดักจับข้อมูลระหว่างการใช้งานเว็บไซต์โดยอาศัยการเข้ารหัสแบบสมมาตรโดยใช้ CAPTCHA เป็นกุญแจในการเข้ารหัสทำให้ระบบสามารถตรวจสอบผู้ใช้งานได้ว่าเป็นมนุษย์จริงหรือไม่ตามคุณสมบัติของ CAPTCHA และได้รหัสผ่านในรูปแบบ One Time Password (OTP) เพื่อแก้ไขปัญหาการแจกจ่ายกุญแจสาธารณะของ [9] และการใช้งานระบบร่วมกับ SSL ของ [10], [11] รวมถึงปัญหาค่าใช้จ่ายและการถูกโจมตีด้วยวิธี MITM ของ SSL และ Self-signed SSL

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 Secure Socket Layer Protocol

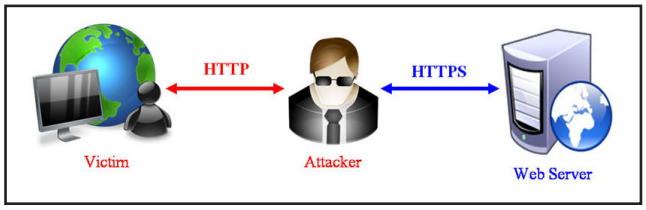
Secure Socket Layer (SSL) Protocol [12] เป็นโพรโทคอลที่พัฒนาโดย Netscape Communications เพื่อใช้ในโพรโทคอลระดับแอพพลิเคชั่น เช่น Hypertext Transfer Protocol (HTTP) โดย SSL จะทำงานระหว่าง Application Protocol และ Transport Protocol เพื่อให้การสื่อสารผ่านเว็บไซต์ดำเนินไปได้อย่างปลอดภัย ต่อมา SSL ได้พัฒนาเป็นมาตรฐานโดย Internet Engineering Task Force (IETF) และเปลี่ยนชื่อเป็น Transport Layer Security (TLS) ดังปรากฏในเอกสาร RFC 2246 และเวอร์ชั่นล่าสุดของ TLS คือ 1.2 ดังปรากฏในเอกสาร RFC 5246 [13] โดยหลักการทำงานของ SSL ก่อนที่จะมีการสื่อสารเซิร์ฟเวอร์จะส่ง Certificate มาที่ไคลเอนต์โดยมีกุญแจสาธารณะของเซิร์ฟเวอร์เพื่อใช้สร้างกุญแจในการเข้ารหัสข้อมูลสำหรับการสื่อสารกับเซิร์ฟเวอร์

2.2. Self-Signed SSL certificates

จากปัญหาค่าใช้จ่ายในการจดทะเบียน SSL จึงมีการเลือกใช้ Self-signed SSL certificates ในการเข้ารหัสข้อมูลระหว่างการใช้งานเว็บไซต์ โดย Self-signed SSL มีคุณสมบัติในการใช้งานเหมือนกับ SSL ที่รับรองโดย CA แต่สามารถสร้างขึ้นเองโดยไม่ผ่าน CA ซึ่งการใช้งาน Self-signed SSL certificates ไม่จำเป็นต้องมีค่าใช้จ่ายเนื่องจากสามารถสร้างขึ้นเองได้โดยโปรแกรมเช่น OpenSSL [14]

2.3 ความมั่นคงในการใช้งาน SSL

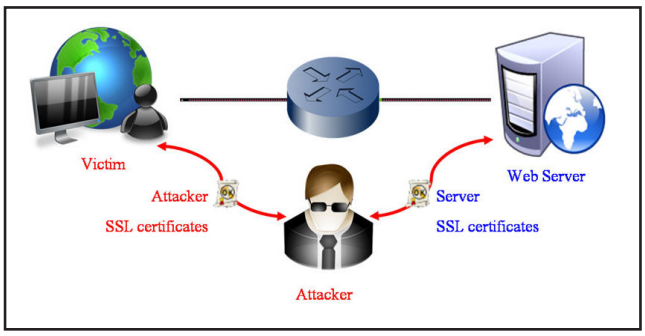
การใช้งาน SSL สามารถถูกโจมตีด้วยวิธี MITM ด้วยเครื่องมือต่างๆ เช่น SSL Strip (มิถุนายน ค.ศ. 2009) ซึ่งลักษณะการโจมตีปกติแล้วการใช้งานเว็บไซต์ที่ใช้ SSL จะมีการแสดงโพรโทคอลเป็น HTTPS ที่เบราว์เซอร์ แต่เมื่อเว็บไซต์ถูกโจมตีด้วย SSL Strip การแสดงโพรโทคอลที่เบราว์เซอร์จะแสดงเป็น HTTP โดยในการสื่อสารระหว่างไคลเอนต์กับผู้โจมตีจะใช้ HTTP ส่วนระหว่างผู้โจมตีกับเซิร์ฟเวอร์จะใช้ HTTPS ซึ่งเซิร์ฟเวอร์ยังมองว่าในการสื่อสารระหว่างไคลเอนต์ดำเนินไปด้วยการทำงานบน HTTPS ดังภาพที่ 1 ด้วยเหตุนี้ถึงแม้เว็บไซต์ที่ดำเนินการจดทะเบียนขอ Certificate ที่รับรองจาก CA ก็ไม่สามารถป้องกันการโจมตีด้วยวิธี MITM ได้



ภาพที่ 1 ลักษณะการโจมตีด้วย SSL Strip

ปัญหาความมั่นคงในการใช้งาน Self-signed SSL เนื่องจากเป็น Certificate ที่สร้างขึ้นเองเมื่อมีการใช้งานระบบปฏิบัติการ (Operating System: OS) และเบราว์เซอร์ไม่มีกุญแจสาธารณะของ Certificate ติดตั้งอยู่ เมื่อเข้าใช้งานเว็บไซต์ที่ใช้ Self-signed SSL ในการเข้ารหัสข้อมูลจะมีการแจ้งเตือนจากเบราว์เซอร์ว่า Certificate ไม่ผ่านการรับรองจาก CA ซึ่งจากการสำรวจเว็บไซต์ที่ให้บริการเกี่ยวกับระบบทะเบียนในมหาวิทยาลัยจำนวน 53 สถาบัน พบว่ามีจำนวน 42 สถาบันที่ใช้ Self-signed SSL ซึ่งมีความเสี่ยงหากถูกโจมตีด้วยเครื่องมือ เช่น Cain & Abel ที่มีการโจมตีในลักษณะส่ง Certificate ปลอมไปที่เหยื่อ (Spoof Certificate) ซึ่งเว็บเบราว์เซอร์

ไม่สามารถแยกแยะได้ว่าเป็นการเข้าใช้งานเว็บไซต์ที่ใช้ Self-signed SSL หรือถูกโจมตีด้วยวิธี Spoof Certificate ดังภาพที่ 2



ภาพที่ 2 *การโจมตีด้วยวิธี Spoof Certificate*

2.4 งานวิจัยที่เกี่ยวข้อง

จตุภูมิ และคณะ [9] ได้เสนอ Web Guardian API เพื่อป้องกันการดักจับข้อมูลจากการโจมตีด้วยวิธี MITM โดยใช้การเข้ารหัสแบบอสมมาตร ซึ่งประสบปัญหาการแจกจ่ายและการตรวจสอบกุญแจสาธารณะของเซิร์ฟเวอร์ หากมีการปลอมแปลงกุญแจสาธารณะในการเข้ารหัสข้อมูล ผู้ใช้งานเว็บไซต์จะไม่สามารถตรวจสอบความถูกต้องของกุญแจสาธารณะที่ใช้เข้ารหัสข้อมูลในการสื่อสารระหว่างไคลเอนต์กับเซิร์ฟเวอร์ได้ ดังนั้นจะเห็นได้ว่าในการป้องกันการดักจับข้อมูลมีหลักการที่ไม่ต่างจากการใช้งาน Self-signed SSL รวมถึงงานวิจัยได้เสนอการใช้งานระบบต้นแบบที่จำเป็นต้องติดตั้ง GnuPG เป็นเครื่องมือในการเข้ารหัสและถอดรหัสข้อมูล ทำให้ระบบใช้เวลามากในการสื่อสารข้อมูลระหว่างไคลเอนต์กับเซิร์ฟเวอร์

อีกทั้งในงานวิจัยนี้ได้ทดสอบโจมตี Web Guardian API ด้วยวิธี MITM โดยดักจับและส่งซ้ำรหัสผ่านที่ถูกเข้ารหัสโดยใช้โปรแกรม Tamper Data ก็สามารถทำได้ ถึงแม้ว่าการเข้ารหัสข้อมูลรหัสผ่านจะป้องกันการดักจับรหัสผ่านต้นฉบับของผู้ใช้แต่ผลของการโจมตี เซิร์ฟเวอร์ไม่สามารถตรวจสอบได้ว่ารหัสผ่านที่ถูกเข้ารหัสนั้นถูกส่งมาจากเจ้าของรหัสผ่านจริงหรือถูกส่งซ้ำ เมื่อเซิร์ฟเวอร์ใช้กุญแจลับที่จัดเก็บไว้ที่เซิร์ฟเวอร์ถอดรหัสข้อมูลรหัสผ่าน ผู้โจมตีก็สามารถเข้าใช้งานระบบได้

Sood และคณะ [10] ได้เสนอวิธีการป้องกันการดักจับรหัสผ่านโดยอาศัยกุญแจ Session Key จาก SSL ในกระบวนการสร้างรหัสผ่านของผู้ใช้ทำให้ได้รหัสผ่านในรูปแบบ OTP แต่ยังประสบปัญหาในการใช้งานหากระบบถูกโจมตีด้วย SSL Strip เนื่องจากการทำงานของระบบอาศัย Session Key จาก SSL รวมถึงงานวิจัยของ Chik และคณะ [11] ได้เสนอวิธีการแก้ไขปัญหา SSL แต่ประสบปัญหาในการนำระบบไปพัฒนาเพื่อใช้งานจริง

3. วิธีดำเนินการวิจัย

งานวิจัยนี้ได้เสนอวิธีการเพื่อแก้ปัญหาการโจมตีแบบ MITM การจัดเก็บรหัสผ่านในฐานข้อมูลของเซิร์ฟเวอร์การ Authentications ที่ปลอดภัยโดยใช้ชื่อว่า Password Protection Function (PPF) API ทดแทน Self-signed SSL ที่ไม่ปลอดภัยหากถูกโจมตีด้วยวิธี MITM งานวิจัยก่อนหน้านี้ของจตุภูมิ และคณะ [9] ที่มีปัญหาในการแจกจ่ายกุญแจสาธารณะที่ใช้เข้ารหัสข้อมูล การโจมตีโดยการส่งซ้ำรหัสผ่านที่ถูกเข้ารหัส และงานวิจัยของ Sood และคณะ [10] ที่เกิดปัญหาในการใช้งานหากถูกโจมตีด้วย SSL Strip รวมถึงการพัฒนาระบบเพื่อใช้งานร่วมกับ SSL เพื่อเพิ่มประสิทธิภาพในการบริการเว็บไซต์ให้มีความปลอดภัย และสร้างความเชื่อมั่นให้กับสมาชิกผู้ใช้บริการเว็บไซต์มากขึ้น เพื่อเป็นระบบต้นแบบในการพัฒนาเว็บไซต์ให้มีความปลอดภัยในการใช้งานต่อไป ซึ่งสามารถกำหนดแนวคิดในการออกแบบ พัฒนาระบบป้องกันการโจมตีรหัสผ่าน และตัวแปรที่ใช้ในการอธิบายได้ดังนี้

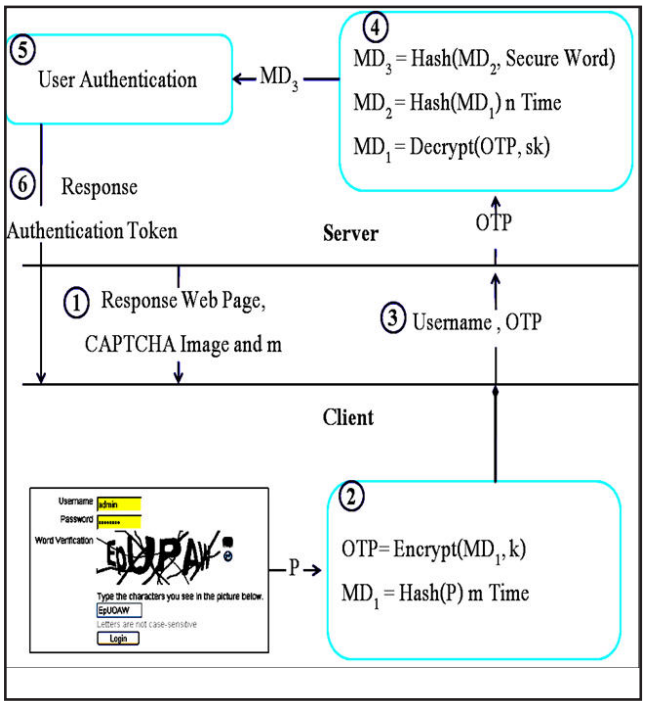
MD คือ Message Digest
t คือ จำนวนทั้งหมดในการคำนวณหาค่า Hash
m คือ จำนวนในการคำนวณค่า Hash ของไคลเอนต์
P คือ รหัสผ่านต้นฉบับของไคลเอนต์
OTP คือ รหัสผ่านในรูปแบบ One Time Password
n คือ ค่าที่ได้จาก t-m
k คือ กุญแจที่ไคลเอนต์ใช้เข้ารหัสข้อมูล
sk คือ กุญแจที่เซิร์ฟเวอร์ใช้ถอดรหัสข้อมูล
Encrypt() คือ การเข้ารหัส
Decrypt() คือ การถอดรหัส

3.1 การจัดเก็บรหัสผ่านในฐานข้อมูล

กระบวนการจัดเก็บรหัสผ่านในฐานข้อมูลของเซิร์ฟเวอร์งานวิจัยนี้เสนอวิธีการจัดเก็บรหัสผ่านในรูปแบบ Message Digest (MD) ที่ได้จากการนำรหัสผ่านของผู้ใช้มา Hash จำนวน t ครั้งจากนั้นนำผลของค่า Hash มารวมกับ Secure

Word ที่ถูกกำหนดโดยผู้พัฒนาเว็บไซต์แล้ว Hash อีก 1 ครั้ง แล้วนำผลลัพธ์ที่ได้จัดเก็บในฐานข้อมูล จากการออกแบบการป้องกันการดักจับรหัสผ่าน โดยการ Hash รหัสผ่านจำนวน t ครั้ง และการนำผลของค่า Hash รวมกับ Secure Word ทำให้สามารถป้องกันการโจมตีที่ฐานข้อมูลของเซิร์ฟเวอร์ด้วยวิธี Brute-Force Attack [15] ในการทำกลับค่า MD เนื่องจากการ Hash จำนวน t ครั้งและการนำผลของค่า Hash รวมกับ Secure Word ซึ่งอาจกำหนดให้มีความยาวและประกอบด้วยอักขระพิเศษ เช่น #@!*& เป็นต้น ทำให้ได้รหัสผ่านที่มีความยาว และซับซ้อนตามที่ผู้พัฒนาเว็บไซต์กำหนด

3.2 การป้องกันการดักจับรหัสผ่าน



ภาพที่ 3 *โครงสร้างการทำงานของ PPF API*

จากภาพที่ 3 แสดงการทำงานของ PPF API ในการป้องกันการดักจับรหัสผ่านโดยการโจมตีด้วยวิธี MITM ซึ่งมีรายละเอียดดังต่อไปนี้

(1) เมื่อไคลเอนต์เรียกไปที่เซิร์ฟเวอร์ กระบวนการทำงานของเซิร์ฟเวอร์จะตอบกลับหน้าเว็บไซต์พร้อมด้วย CAPTCHA Image และค่า m มาที่ไคลเอนต์ซึ่ง m คือค่าที่ได้จากการสุ่ม ในการออกแบบโปรแกรมสามารถกำหนดช่วงของการสุ่มค่า m ได้ ตามความต้องการของผู้พัฒนาเว็บไซต์ โดยที่ค่า m จะต้องน้อยกว่าหรือเท่ากับค่า t

(2) ไคลเอนต์คำนวณหาค่า MD1 ที่ได้จาก MD1= Hash(P) m Time แล้วคำนวณค่า OTP ซึ่ง OTP = Encrypt (MD1, k) โดย k เป็นค่าที่ได้จากการกรอก CAPTCHA ของผู้ใช้

(3) ไคลเอนต์ส่ง Username และ OTP ไปที่เซิร์ฟเวอร์

(4) เซิร์ฟเวอร์คำนวณหาค่า MD1 โดย MD1 = Decrypt (OTP, sk) ซึ่ง sk คือกุญแจที่ได้จาก CAPTCHA ที่จัดเก็บไว้ที่เซิร์ฟเวอร์ จากนั้นคำนวณหาค่า MD2 โดย MD2 = Hash(MD1) n Time แล้วคำนวณหาค่า MD3 โดย MD3 = Hash(MD2|Secure Word) ซึ่ง Secure Word คือ ข้อความที่ถูกกำหนดในขั้นตอนการจัดเก็บรหัสผ่านในฐานข้อมูล

(5) นำ MD3 เข้าสู่กระบวนการ User Authentication

(6) ส่งผลการตรวจสอบสิทธิ์การเข้าใช้งานระบบไปที่ไคลเอนต์

การส่ง CAPTCHA Image เพื่อเป็นกุญแจในการเข้ารหัสจะทำให้ผู้โจมตีที่ใช้วิธีการโจมตี MITM ไม่สามารถทราบถึงกุญแจที่ใช้เข้ารหัสข้อมูลรหัสผ่านได้ ซึ่งค่าที่ปรากฏใน CAPTCHA Image จะมีการเปลี่ยนค่าทุกครั้งที่ไคลเอนต์เรียกไปที่เซิร์ฟเวอร์เพื่อขอหน้าเว็บไซต์ ทำให้กุญแจที่ใช้เข้ารหัสไม่ซ้ำกัน อีกทั้งการใช้ CAPTCHA Image เป็นกุญแจในการเข้ารหัสทำให้ระบบสามารถตรวจสอบผู้ใช้งานได้ว่าเป็นมนุษย์จริงหรือไม่ตามคุณสมบัติของ CAPTCHA รวมถึงการ Hash รหัสผ่านต้นฉบับจำนวน m ครั้งก่อนที่ถูกเข้ารหัสหากเกิดการโจมตีที่กุญแจที่ใช้ในการเข้ารหัสผู้โจมตีก็จำเป็นที่จะต้องหาวิธีในกระบวนการทำกลับค่า MD ให้เป็นรหัสผ่านต้นฉบับซึ่งสามารถเพิ่มความปลอดภัยหากมีการโจมตีด้วยวิธี Brute-Force Attack จากกระบวนการหาค่า OTP โดยอาศัยกุญแจในการเข้ารหัสที่ได้จาก CAPTCHA และการ Hash จำนวน m ครั้ง จะทำให้ได้รหัสผ่านที่ใช้ในการตรวจสอบสิทธิ์เพื่อเข้าใช้งานระบบที่ไม่ซ้ำกันหรือรหัสผ่านในรูปแบบ OTP ทำให้สามารถแก้ไขปัญหาเมื่อผู้โจมตีดักจับและส่งรหัสผ่านซ้ำไปที่เซิร์ฟเวอร์

3.3 การพัฒนาระบบ

งานวิจัยนี้ได้พัฒนาระบบต้นแบบในการตรวจสอบสิทธิ์การเข้าใช้งานระบบและวิธีการป้องกันการดักจับรหัสผ่านด้วยการโจมตีแบบ MITM ระหว่างการใช้งานเว็บไซต์บนโปรแกรม MOODLE Version 1.9.14 โดยใช้ภาษา JavaScript ร่วมกับภาษา PHP การทำงานของระบบมีการเข้ารหัสข้อมูลรหัสผ่านที่ไคลเอนต์ก่อนที่จะส่งข้อมูลไปยังเซิร์ฟเวอร์เพื่อถอดรหัสและนำข้อมูลรหัสผ่านเข้าสู่กระบวนการตรวจสอบ

สิทธิ์ในการใช้งานระบบโดยอาศัยวิธีการเข้ารหัสแบบสมมาตร กุญแจที่ใช้ในการเข้ารหัสได้จาก CAPTCHA ซึ่งออกแบบให้ใช้อัลกอริธึม AES ขนาดกุญแจ 256 bits ในการเข้ารหัส การ Hash ข้อมูลด้วยอัลกอริธึม SHA1 ที่มีขนาดความยาวของค่า Hash เป็น 160 bits ซึ่งหลักการทำงานของ PPF API ส่วนของไคลเอนต์แสดงในภาพที่ 4 โดยการใช้งานระบบร่วมกับ MOODLE จะต้องแก้ไขไฟล์ lo-gin/index_form.html ดังในภาพที่ 5

PasswordProtectionFunctionAPI.js

PasswordProtectionFunctionAPI(MD,k,keyLength)

MD	ค่า Hash ของรหัสผ่าน
k	ค่าที่ได้จากการกรอก CAPTCHA ของผู้ใช้
keyLength	ขนาดของ Key ซึ่งกำหนดเป็น 256

Return

OTP	รหัสผ่านที่ถูกเข้ารหัส
error	ข้อความในกรณีที่เกิดข้อผิดพลาด

ภาพที่ 4 *หลักการทำงานของ PPF API*

```
<script type="text/javascript" src="api/PasswordProtectionFunctionAPI.js"></script>
<script type="text/javascript">
function encryptPWD(from){
    var keyLength = 256;
    var username = from.username.value;
    var p = from.password.value;
    if(p='' || username =''){
        alert("Please input username and password");
        return false;
    }else{
        var k = from.CAPTCHA.value;
        var m = from.m.value;
        var MD=p;

        for(var i=0;i<m;i++){
            MD=Crypto.SHA1(MD);
        }
        var dataForEnCaptcha=Crypto.SHA1(Crypto.SHA1(k.toLowerCase()));
        var OTP = new PasswordProtectionFunctionAPI(MD,k,keyLength);
        if(OTP){
            var dataEncryptCaptcha = AesCtr.encrypt(dataForEnCaptcha,k,keyLength);
            from.codeSecure.value=dataEncryptCaptcha;
            from.password2.value=OTP;
        }else{
            alert(OTP.error);
        }

    }
}
</script>
```

ภาพที่ 5 *หลักการทำงานของ PPF API*

ส่วนการตรวจสอบสิทธิ์ในการเข้าใช้งานระบบของเซิร์ฟเวอร์ออกแบบให้มีการ Hash ข้อมูลด้วยอัลกอริธึม SHA1 โดยนำข้อมูลค่า Hash ของรหัสผ่านมาผสมกับค่า Secure Word ก่อนที่จะคำนวณค่า Hash อีกครั้งแล้วจึงนำค่าที่ได้เข้าสู่กระบวนการตรวจสอบสิทธิ์ในการเข้าใช้งานระบบซึ่ง Secure Word ผู้พัฒนาเว็บไซต์เป็นผู้กำหนดขึ้น รวมถึงการกำหนดจำนวนครั้งในการคำนวณค่า Hash ซึ่งการ Hash ข้อมูลหลายๆ ครั้ง ทำให้ยากต่อการใช้วิธี Brute-Force Attack ในการโจมตีโดยการใช้งานร่วมกับ MOODLE ต้องแก้ไขไฟล์ login/index.php ดังภาพที่ 6 และ 7

```
<?php // $Id: index.php,v 1.129.2.3 2008/05/18 21:26:57
if(!isset($_SESSION))
    {
        session_start();
    }
    require_once("../config.php");
    require_once("api/PasswordProtectionFunctionAPI.php");
```

ภาพที่ 6 *ไฟล์ login/index.php ที่ผ่านการแก้ไข*

```
        } else {
            if (empty($errormsg)) {
                //======== Password Protection Function API
                $objDecrypt = new PasswordProtectionFunctionAPI();
                $frm->password=$objDecrypt->PasswordProtectionDec
                $user = authenticate_user_login($frm->username, $frm
            }
        }
        update_login_count();
```

ภาพที่ 7 *การกำหนดฟังก์ชันการถอดรหัสใน login/index.php*

4. การทดสอบระบบ

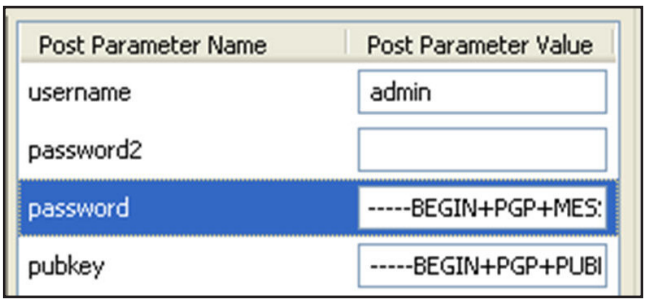
การทดสอบระบบเพื่อเปรียบเทียบประสิทธิภาพและประสิทธิผลการทำงานของระบบป้องกันการโจมตีรหัสผ่านระหว่าง PPF API กับ SSL, Self-signed SSL และงานวิจัยก่อนหน้านี้ โดยทดสอบที่เครื่องเซิร์ฟเวอร์ซึ่งมีรายละเอียดของเซิร์ฟเวอร์ คือใช้ Intel® XEON™ CPU 2.40 GHz RAM 1 GB ระบบปฏิบัติการ Windows 2003 เว็บเซิร์ฟเวอร์ Apache/2.2.0 (Win32) PHP Version 5.2.4 ฐานข้อมูล MySQL Version 5.0.24a และเครื่องไคลเอนต์ใช้ CPU INTEL CORE 2 QUAD Q6600 RAM 4 GBระบบปฏิบัติการ Windows XP เว็บเบราว์เซอร์ Firefox 5.0.1 และ Internet Explorer 8 เป็นการทดสอบในระบบปิด โดยใช้เครื่องมือในการทดสอบที่มีคุณสมบัติในการโจมตีข้อมูลรหัสผ่าน 5 ชนิดคือ Fake Public Key ที่พัฒนาโดยงานวิจัยนี้ Cain & Abel, Tamper Data Plugin, SSL Strip และ Ettercap

4.1 ทดสอบโจมตีงานวิจัยก่อนหน้านี้

งานวิจัยนี้ได้ทำการทดสอบโจมตี [9] โดยปลอมแปลงกุญแจสาธารณะของเซิร์ฟเวอร์โดยพัฒนาโปรแกรมด้วยภาษา Java ใช้ชื่อว่า Fake Public Key เพื่อใช้ในการโจมตี Anti-sniff API โดยผลการทดสอบเมื่อเหยื่อเข้าใช้งานระบบรหัสผ่านที่ใช้ตรวจสอบการเข้าใช้งานจะถูกเข้ารหัสโดยกุญแจสาธารณะของผู้โจมตีเมื่อผู้โจมตีดักจับข้อมูลรหัสผ่านที่ถูกเข้ารหัส

ผู้โจมตีก็สามารถใช้กุญแจลับของผู้โจมตีในการถอดรหัสได้

การทดสอบโจมตีโดยการใช้ Tamper Data ซึ่งโปรแกรมมีคุณสมบัติในการตรวจสอบและแก้ไขค่าพารามิเตอร์แบบ Post ที่ส่งไปที่เซิร์ฟเวอร์ โดยการทดสอบโจมตีจะส่งรหัสผ่านที่ดักจับได้โดยผู้โจมตีไม่จำเป็นที่จะต้องถอดรหัสข้อมูลรหัสผ่านแสดงตัวอย่างการโจมตีในภาพที่ 8



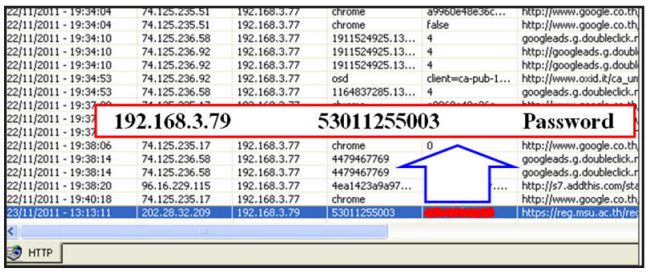
ภาพที่ 8 *การโจมตีโดยใช้โปรแกรม Tamper Data*

4.2 ทดสอบโจมตี SSL

การทดสอบโจมตี SSL ในงานวิจัยนี้ใช้ SSL Strip โดยผลการโจมตีด้วยวิธี MITM แสดงให้เห็นว่า สามารถดักจับข้อมูลชื่อ และรหัสผ่านของผู้ใช้ในการเข้าใช้งานระบบได้ดังภาพที่ 9 ซึ่งระบบที่พัฒนาใน [10] หากถูกโจมตีด้วย SSL Strip ระบบก็ไม่สามารถทำงานได้เนื่องจากไม่มีค่า Session Key จาก SSL ที่ใช้ในกระบวนการสร้างรหัสผ่าน

ภาพที่ 9 *ผลการทดสอบโจมตี SSL ด้วย SSL Strip*

ในการโจมตีเว็บไซต์ที่ใช้ Self-signed SSL ในการทดสอบใช้โปรแกรม Cain & Abel เพื่อดักจับข้อมูลชื่อ และรหัสผ่านของผู้ใช้โดยผลการโจมตีแสดงดังภาพที่ 10

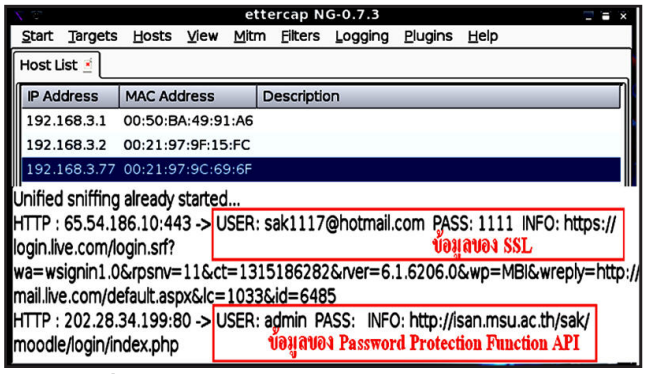


ภาพที่ 10 *ผลการทดสอบโจมตี Self-signed SSL*

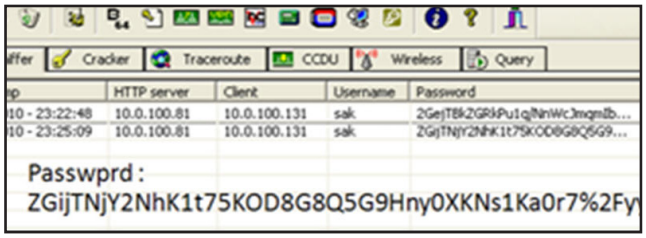
จากภาพที่ 10 เป็นผลของการใช้โปรแกรม Cain & Abel ในการโจมตีเว็บไซต์ที่ใช้ Self-signed SSL โดยแสดงให้เห็นว่าสามารถดักจับข้อมูลชื่อ และรหัสผ่านของผู้ใช้ได้ โดยในการโจมตี Self-signed SSL ของโปรแกรม Cain & Abel ใช้หลักการสร้าง Certificate ปลอมส่งไปที่เครื่องเหยื่อโดยเว็บเบราว์เซอร์ไม่สามารถแยกแยะได้ว่าเป็นการโจมตีด้วยโปรแกรม Cain & Abel หรือเป็นเว็บไซต์ที่ใช้งาน Self-signed SSL

4.3 ทดสอบโจมตี Password Protection Function (PPF) API

งานวิจัยนี้ทำการทดสอบโจมตี PPF API ด้วยโปรแกรม Ettercap และ Cain & Abel โดยผลการทดสอบไม่สามารถดักจับรหัสผ่านของ PPF API ได้ในโปรแกรม Ettercap ดังภาพที่ 11 และไม่สามารถถอดรหัสข้อมูลที่ดักจับได้ในโปรแกรม Cain & Abel ซึ่งข้อมูลที่แสดงอยู่ในรูปของ Cipher Text ดังภาพที่ 12



ภาพที่ 11 *ผลการทดสอบโจมตี SSL เปรียบเทียบกับ PPF API*



ภาพที่ 12 *ผลการทดสอบโจมตี PPF API*

จากภาพที่ 11 แสดงการทดสอบการโจมตี SSL เปรียบเทียบกับ PPF API ด้วยโปรแกรม Ettercap จะเห็นได้ว่าข้อมูลชื่อผู้ใช้และรหัสผ่านของระบบที่ใช้ SSL สามารถดักจับและถอดรหัสได้ และระบบที่ใช้งาน PPF API ข้อมูลรหัสผ่านของผู้ใช้ไม่สามารถดักจับได้

การทดสอบโจมตี PPF API งานวิจัยนี้ใช้โปรแกรม Cain & Abel ซึ่งผลการทดสอบไม่สามารถแสดงข้อมูลสำคัญรหัส

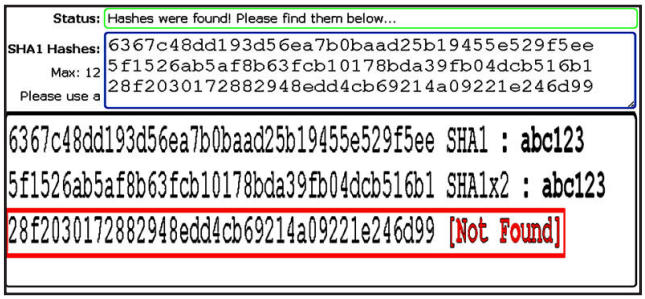
ผ่านในการเข้าใช้งานระบบได้ดังภาพที่ 12 และเมื่อนำรหัสผ่านที่ดักจับได้ส่งซ้ำไปที่เซิร์ฟเวอร์โดยใช้โปรแกรม Tamper Data ผู้โจมตีก็ไม่สามารถโจมตีเข้าใช้งานระบบได้ เนื่องจากค่า CAPTCHA ที่ใช้ในการถอดรหัสและค่า n ที่เป็นจำนวนครั้งในการหาค่า Hash ที่ถูกจัดเก็บไว้ที่เซิร์ฟเวอร์ถูกทำลายหลังจากที่ผู้ใช้จริงเข้าใช้งานระบบ โดยผลการทดสอบสามารถสรุปได้ดังแสดงในตารางที่ 1

ตารางที่ 1 *สรุปผลการทดสอบโจมตี SSL Web Guardian API และ PPF API*

Tools	SSL	Web Guardian API	PPF API
Fake Public Key	✓	✘	✓
Cain & Abel	✘	✓	✓
Tamper Data	✓	✘	✓
SSL Strip	✘	✓	✓
Ettercap	✘	✓	✓

✓ : สามารถป้องกันการโจมตีได้ ✘ : ไม่สามารถป้องกันการโจมตีได้

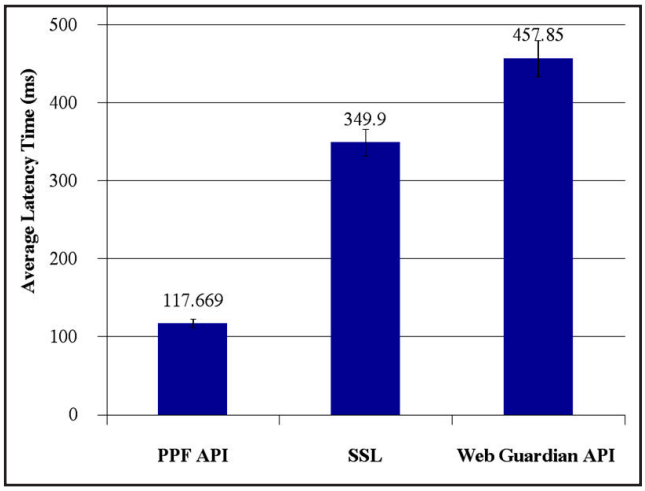
จากการพัฒนาโปรแกรมต้นแบบให้มีการ Hash ข้อมูลจำนวน m ครั้งที่ไคลเอนต์ก่อนเข้ารหัสและ t ครั้งที่เซิร์ฟเวอร์แล้วนำผลของค่า Hash รวมกับ Secure Word ก่อนจัดเก็บในฐานข้อมูลสามารถป้องกันการโจมตีแบบ Brute-Force Attack เพื่อทำกลับค่า Hash ให้เป็นรหัสผ่านต้นฉบับได้ โดยจากการทดสอบโจมตีด้วย [16] พบว่าสามารถโจมตีทำกลับค่า Hash ให้เป็นรหัสผ่านต้นฉบับได้ในกรณีที่มีการ Hash รหัสผ่านจำนวน 1-2 ครั้ง การกำหนดรหัสผ่านที่ประกอบด้วยตัวเลขและตัวอักษรเพียงอย่างเดียว ดังนั้นการกำหนดรหัสผ่านให้มีความยาว และประกอบด้วยอักขระพิเศษ สามารถเพิ่มความซับซ้อนให้กับรหัสผ่านและแก้ไขปัญหาในกรณีที่ผู้ใช้กำหนดรหัสผ่านที่ไม่มีความยาวรวมถึงเป็นคำที่รู้จักและคุ้นเคยอย่างเช่น abc123 สำหรับการทดสอบรหัสผ่านของ PPF API กำหนดรหัสผ่านต้นฉบับเป็น abc123 มีการ Hash จำนวน 3 ครั้งและ Secure Word ที่ประกอบด้วย isan@S&$!a1117 โดยผลของการทดสอบแสดงให้เห็นว่ารหัสผ่านของ PPF API ไม่สามารถโจมตีทำกลับค่า Hash ได้ดังภาพที่ 13



ภาพที่ 13 *การโจมตี PPF API ด้วยวิธี Brute-Force Attack*

4.4 เปรียบเทียบประสิทธิภาพของระบบ

การทดสอบเพื่อเปรียบเทียบประสิทธิภาพของ PPF API กับ SSL และ Web Guardian API ทำการทดสอบโดยส่งผ่านข้อมูลขนาด 20 ตัวอักษรครอบคลุมขนาดของรหัสผ่านที่ใช้ทั่วไป ซึ่งทดสอบส่งข้อมูลจำนวน 100 ครั้ง โดยผลการทดสอบแสดงเวลาเฉลี่ยหน่วยเป็น millisecond (ms) ซึ่งแสดงผลการทดสอบดังภาพที่ 14



ภาพที่ 14 *ผลการทดสอบโจมตี PPF API*

จากภาพที่ 14 แสดงผลการทดสอบเพื่อเปรียบเทียบการทำงานของ PPF API กับ SSL และ Web Guardian API โดย PPF API มีการกำหนดการ Hash ข้อมูลจำนวน 10 ครั้งที่เครื่องไคลเอนต์ก่อนส่งข้อมูลไปยังเครื่องเซิร์ฟเวอร์ และกำหนด Secure Word จำนวน 20 ตัวอักษร จากผลการทดสอบพบว่าการเพิ่มจำนวนในการ Hash และความยาวของ Secure Wordเพื่อเพิ่มประสิทธิภาพในการป้องกันรหัสผ่านเวลาในการส่งข้อมูลของ PPF API น้อยกว่า SSL และ Web Guardian API ช่วงความเชื่อมั่น (Confidence Interval) ที่ระดับนัยสำคัญ 95 %

5. บทสรุป

การออกแบบการป้องกันการดักจับรหัสผ่านระหว่างการใช้งานเว็บไซต์ในงานวิจัยนี้สามารถป้องกันการโจมตีด้วยวิธี MITM และการส่งรหัสผ่านซ้ำได้ การทดสอบประสิทธิภาพในการป้องกันการดักจับรหัสผ่านกับ SSL ซึ่งเป็นที่นิยมใช้ในการป้องกันการดักจับข้อมูล รวมถึง Self-signed SSL และงานวิจัยก่อนหน้านี้ โดยประสิทธิภาพในการป้องกันการดักจับข้อมูลและความเร็วในการส่งข้อมูลระหว่างไคลเอนต์และเซิร์ฟเวอร์สามารถทำงานได้ดีกว่า SSL, Self-signed SSL และงานวิจัยก่อนหน้านี้ หากองค์กรที่ให้บริการเว็บไซต์โดยใช้ SSL ในการป้องกันการดักจับข้อมูลก็สามารถนำระบบที่พัฒนาโดยงานวิจัยนี้ใช้ร่วมกับ SSL เพื่อเพิ่มกลไกลในการป้องกันรหัสผ่านและสามารถตรวจสอบผู้ใช้งานได้ว่าเป็นมนุษย์จริงหรือไม่ อีกทั้งไม่มีค่าใช้จ่ายในการใช้งานโดยงานวิจัยนี้ใช้การเข้ารหัสข้อมูลแบบสมมาตรจึงไม่มีปัญหาเรื่องการแจกจ่ายกุญแจสาธารณะที่ใช้เข้ารหัสและไม่จำเป็นต้องติดตั้งเครื่องมือที่ใช้ในกระบวนการเข้ารหัสข้อมูลอีกด้วย รวมถึงรหัสผ่านที่ได้อยู่ในรูปแบบ OTP ซึ่งจะทำให้เพิ่มความปลอดภัยในการใช้งานเว็บไซต์ได้ดียิ่งขึ้น โดยในงานวิจัยนี้ได้พัฒนาระบบและทดลองใช้งานที่ http://isan.msu.ac.th/sak/moodle/

7. เอกสารอ้างอิง

[1] P. Burkholder, "SSL Man-in-the-Middle Attacks," *SANS Institue InfoSec Reading*, 2003.

[2] "Cain & Abel," http://www.oxid.it/cain.html

[3] M. Moxie, "Ssltrip," http://www.thoughtcrime.org/software/sslstrip/

[4] A. Ornaghi and M. Valleri, "Ettercap," http://ettercap.sourceforge.net/

[5] A. Judson, "Tamper Data," https://addons.mozilla.org/en-US/firefox/addon/tamper-data/

[6] "Cain and Abel Tutorial," http://www.thehackerslibrary.com/?p=414

[7] "Ettercap - The Easy Tutorial," http://openmaniak.com/ettercap.php

[8] "Tamper Data tutorial," http://tamperdata.mozdev.org/screenshots.html

[9] จตุภูมิ จวนชัยภูมิ และ สมนึก พ่วงพรพิทักษ์, "การพัฒนาระบบป้องกันการแทรกแซงการสื่อสารเพื่อใช้แทนเทคโนโลยีเอสเอสแอลแบบใบประกาศด้วยตนเอง," *เอกสารการประชุมวิชาการงานวิจัยและพัฒนาเชิงประยุกต์*, ครั้งที่ 1; 4-6 พฤษภาคม 2552; กรุงเทพฯ, ประเทศไทย, หน้า 403-408, 2552.

[10] S. Sood, A. Sarje, K. Singh, "Inverse Cookie-based Virtual Password Authentication Protocol," *International Journal of Network Security* 2011; Vol.13, No.2: pp. 98-108, 2011.

[11] T. Chik, T. Joseph, "Protection Against Web based Password Phishing," ITNG 2007; 2-4 April 2007; Las Vegas, Nevada, USA, pp. 754-759, 2007.

[12] T. Dierks, P. Karlton, "The TLS Protocol," RFC 2246, December 1999.

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

[14] E. Young, T. Hudson, "OpenSSL: The Open Source toolkit for SSL/TLS," http://www.openssl.org

[15] K. Theoharoulis, I. Papaefstathiou, C. Manifavas, "Implementing Rainbow Tables in High-end FPGAs for Super-fast Password Cracking," *In Proceedings of International Conference on Field Programmable Logic and Applications*, pp. 145-150, 2010.

[16] "MD5 Decrypter tool," http://www.md5decrypter.co.uk/