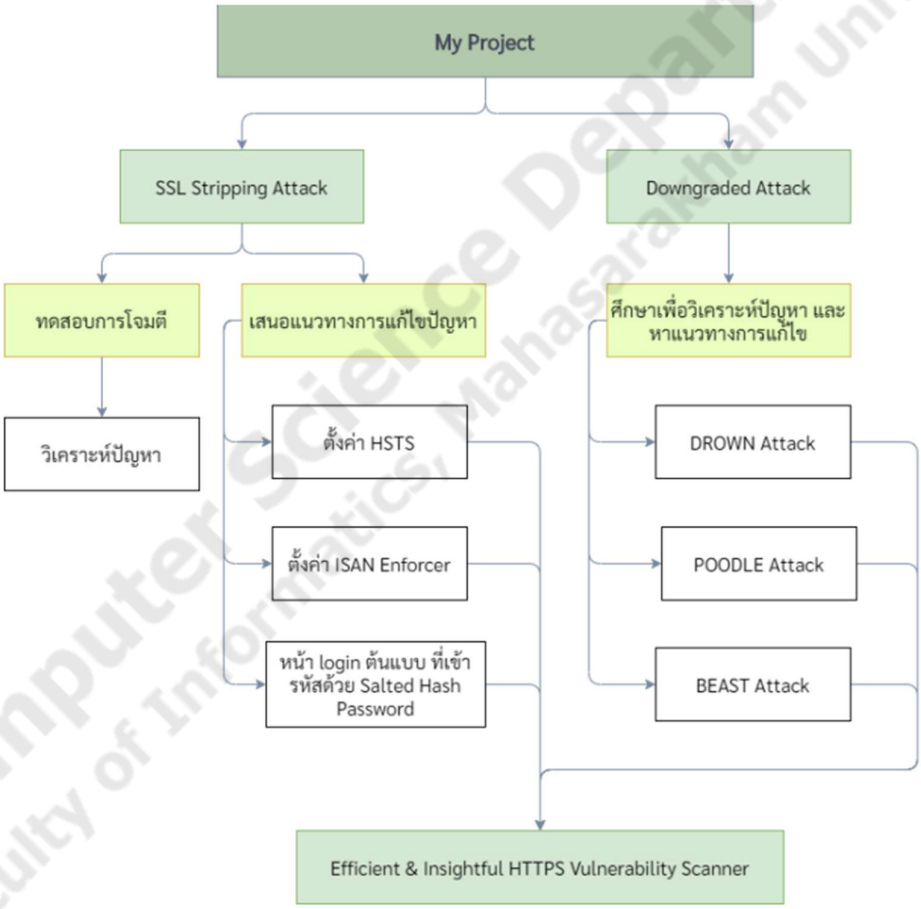
บทที่ 3
ขั้นตอนการดำเนินงาน

ในบทนี้จะกล่าวถึงการทดสอบการโจมตีด้วยเทคนิค SSL Stripping Attack ขั้นตอนการดำเนินงานในส่วนที่จะพัฒนาเป็นโปรแกรมสำหรับใช้ในการตรวจสอบหาช่องโหว่ SSL Stripping Attack, DROWN Attack, POODLE Attack และ BEAST Attack และการสร้างเว็บไซต์ล็อกอินต้นแบบที่สามารถป้องกันการโจมตีด้วยเทคนิค SSL Stripping Attack ซึ่งจะมีกรอบการดำเนินงาน ดังต่อไปนี้

3.1 กรอบการดำเนินงาน

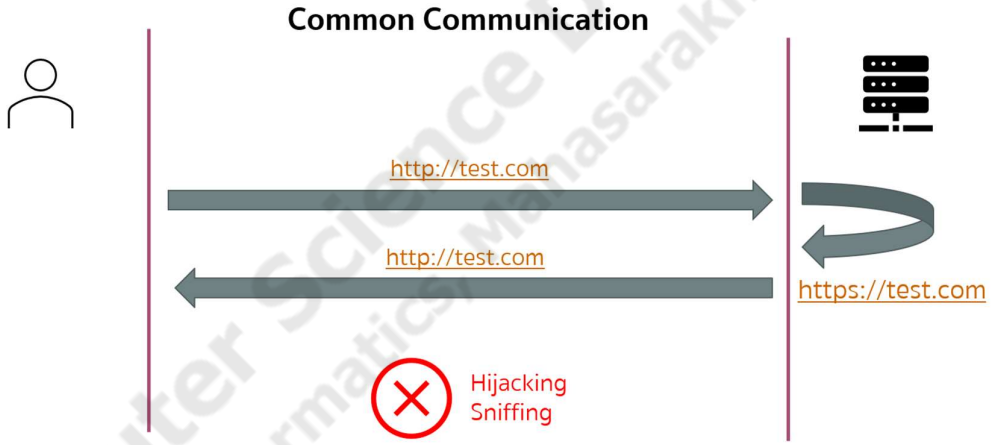


ภาพประกอบที่ 3.1 ภาพรวมของโครงงานปริญญานิพนธ์

ภาพประกอบที่ 3.1 ภาพรวมของโครงงานปริญญานิพนธ์ แสดงให้เห็นถึงกรอบการดำเนินงานดังนี้

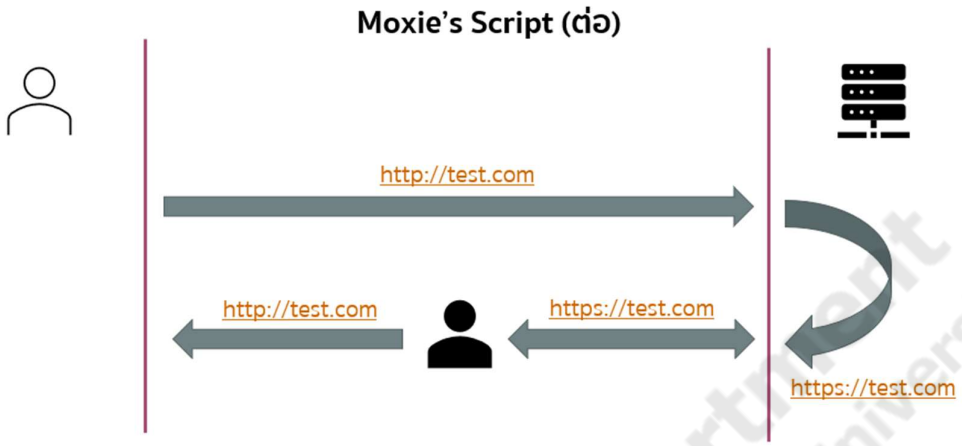
1) ทดสอบการโจมตีด้วยเทคนิควิธี SSL Stripping Attack เป็นการทำงานทดสอบเว็บไซต์ HTTPS ว่ามีช่องโหว่หรือไม่ ผลการทดสอบมาวิเคราะห์เพื่อสรุปถึงปัญหาของช่องโหว่ที่ทำให้เว็บไซต์ต่าง ๆ นั้น ถูกโจมตีได้
2) ศึกษาข้อมูลเกี่ยวกับการโจมตีด้วยเทคนิค DROWN Attack, POODLE Attack และ BEAST Attack เพื่อนำมาวิเคราะห์ และสรุปผล
3) สร้างเว็บไซต์ล็อกอินต้นแบบในการป้องกันการโจมตีด้วยเทคนิค SSL Stripping Attack
4) สร้างโปรแกรมที่ใช้ในการตรวจสอบหาช่องโหว่ของเว็บไซต์ HTTPS ที่เป็นจุดเสี่ยงต่อการถูกโจมตีด้วยเทคนิค SSL Stripping Attack, DROWN Attack, POODLE Attack และ BEAST Attack

3.2 การทดสอบการโจมตีด้วยเทคนิค SSL Stripping Attack



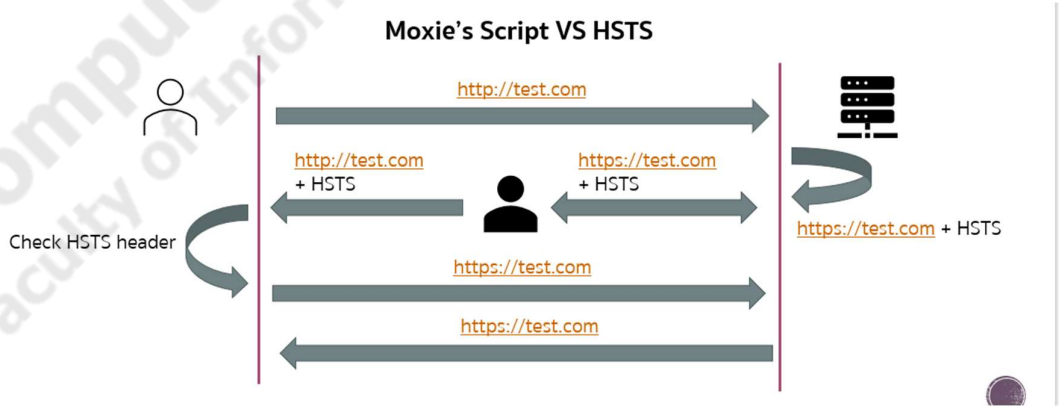
ภาพประกอบที่ 3.2 การสื่อสาร HTTPS แบบทั่วไป

ภาพประกอบที่ 3.2 การสื่อสาร HTTPS แบบทั่วไป เป็นการสื่อสารของ HTTPS Protocol โดยการสื่อสารด้วย HTTPS Protocol ผู้ใช้จะ request เว็บไซต์ปลายทางด้วย Domain name ซึ่งการร้องขอไปในรูปแบบนี้ ทำให้ Browser มองว่าเป็น HTTP Protocol โดยใช้ port 80 ในการสื่อสาร Web Server จะทำการ redirect HTTP Protocol และเปลี่ยนมาใช้ HTTPS Protocol โดยใช้ port 443 ในการสื่อสาร ทำให้การสื่อสารนี้มีการเข้ารหัส และมีความมั่นคงในการสื่อสาร การสื่อสารด้วย HTTPS Protocol สามารถป้องกันการ hijack และ การ Sniff ได้



ภาพประกอบที่ 3.3 การแทรกกลางการสื่อสาร

ภาพประกอบที่ 3.3 การแทรกกลางการสื่อสาร เป็นการแสดงเทคนิคการแทรกกลางการสื่อสาร Man in the middle (MITM) เป็นการสื่อสารที่มีแฮคเกอร์อยู่ตรงกลางการสื่อสาร ระหว่างผู้ใช้งาน และ Web Server โดยอาศัยช่องโหว่ที่ผู้ใช้งานนั้นร้องขอ Web Server ในรูปแบบของ HTTP คือ เมื่อผู้ใช้ร้องขอเว็บไซต์ปลายทางด้วย HTTP Protocol เว็บไซต์ปลายทาง redirect เปลี่ยนให้อยู่ในรูปแบบของ HTTPS Protocol แล้วตอบกลับไปยังผู้ใช้งาน แต่ทว่า แฮคเกอร์ปลด HTTPS ให้อยู่ในรูปแบบของ HTTP แล้วส่งต่อไปยังผู้ใช้งาน ทำให้ผู้ใช้งานที่ร้องขอไปด้วยรูปแบบของ HTTP เมื่อถูกโจมตีจึงไม่ทันสังเกตเห็นความผิดปกติ



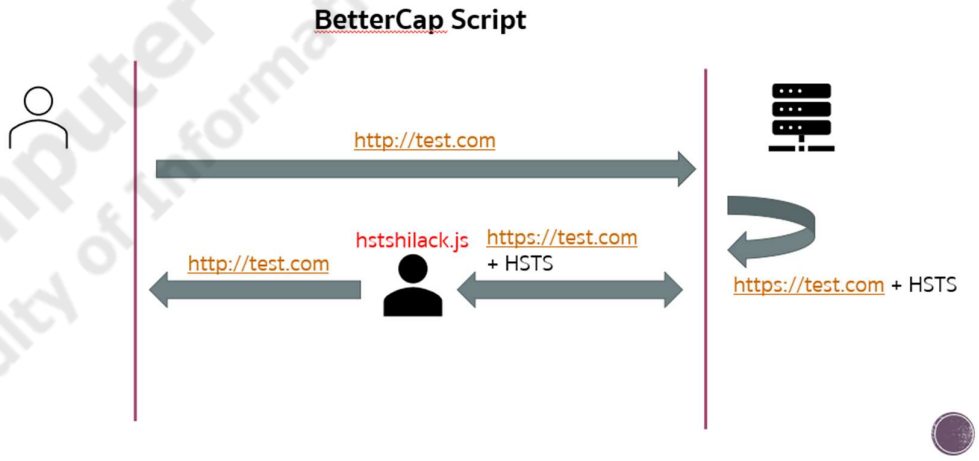
ภาพประกอบที่ 3.4 ป้องกันการแทรกกลางการสื่อด้วย HSTS

ภาพประกอบที่ 3.4 ป้องกันการแทรกกลางการสื่อด้วย HSTS HTTP Strict Transport Security (HSTS) เป็นส่วนเสริมที่เข้ามาเพื่อหวังจะป้องกันการโจมตีด้วยเทคนิค SSL Stripping Attack เมื่อผู้ใช้งานร้องขอไปในรูปแบบ HTTP Protocol เว็บไซต์ปลายทางจะทำการ redirect เปลี่ยนเป็น HTTPS และเสริมด้วย HSTS แฮคเกอร์ที่อยู่ตรงกลางการสื่อสารปลด HTTPS เป็น HTTP แต่ HSTS นั้นยังคงอยู่ แฮคเกอร์ส่งต่อไปยังผู้ใช้งาน HSTS จะ re - check Header ว่าเป็น HTTPS หรือไม่ หากไม่ HSTS จะบังคับให้ Header เป็น HTTPS โดยอัตโนมัติ

ตารางที่ 3.1 การปลด HTTPS และดักจับข้อมูลกับ HSTS Header

Domain name	HSTS	Strip HSTS	Sniff
Msu.ac.th	No	Yes	yes
Kasikornbank.com	yes (not Preload)	?	?
Ktbnetbank.net	yes	No	No

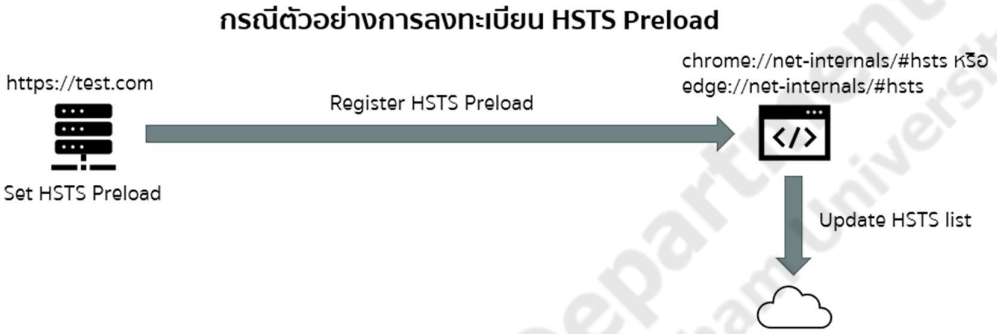
ตารางที่ 3.1 การปลด HTTPS และดักจับข้อมูลกับ HSTS Header เป็นการทดสอบเบื้องต้น พบว่าเว็บไซต์ HTTPS ที่ไม่ได้กำหนดค่าให้กับ HSTS นั้นสามารถปลด HTTPS ให้อยู่ในรูปแบบของ HTTP ทำให้ข้อมูลนั้นถูกดักจับได้ในรูปแบบของ Clear text สำหรับเว็บไซต์ที่มีการกำหนดค่าให้กับ HSTS แต่ไม่ได้ Preload ขึ้นไว้บน Google cloud ในทางทฤษฎีนั้นสามารถป้องกันการถูกปลด HTTPS ได้ และกรณีที่เว็บไซต์ HTTPS ที่ได้กำหนดค่า HSTS และ Preload HSTS ไว้บน Google cloud นั้นสามารถป้องกันการถูกปลด HTTPS ได้



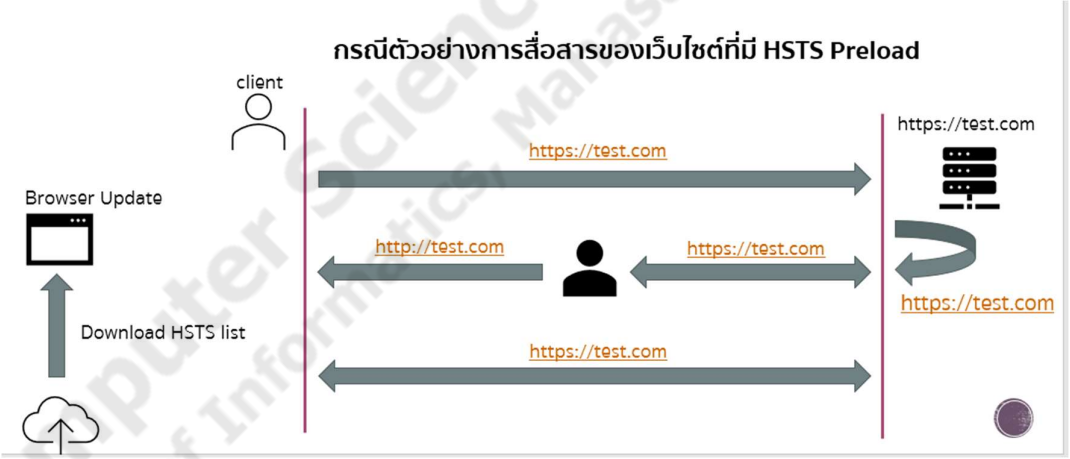
ภาพประกอบที่ 3.5 การแทรกกลางการสื่อสารด้วย BetterCap

จากการทดสอบโดยใช้ BetterCap ซึ่งเป็น Tools ของระบบปฏิบัติการลินุกซ์ Kali Linux OS ทดสอบ เมื่อผู้ใช้งานร้องขอเว็บไซต์ในรูปแบบ HTTP เว็บไซต์ redirect แล้วเปลี่ยน HTTP ให้เป็น

HTTPS และเสริมด้วย HSTS ก่อนตอบกลับไปยังผู้ใช้งาน แฮคเกอร์ที่อยู่ตรงกลางการสื่อสารทำการปลด HTTPS ออก จนเหลือเพียง HTTP และใช้ Script ของ BetterCap ที่เป็นไฟล์ JavaScript สำหรับปลด HSTS Header ออก แล้วยัง HTTP ไปยังผู้ใช้งาน ดังนั้นการสื่อสารที่มีการกำหนดค่า HSTS เพียงอย่างเดียวจึง ไม่สามารถป้องการโจมตีด้วยเทคนิค SSL Stripping Attack ได้ดังภาพประกอบที่ 3.5



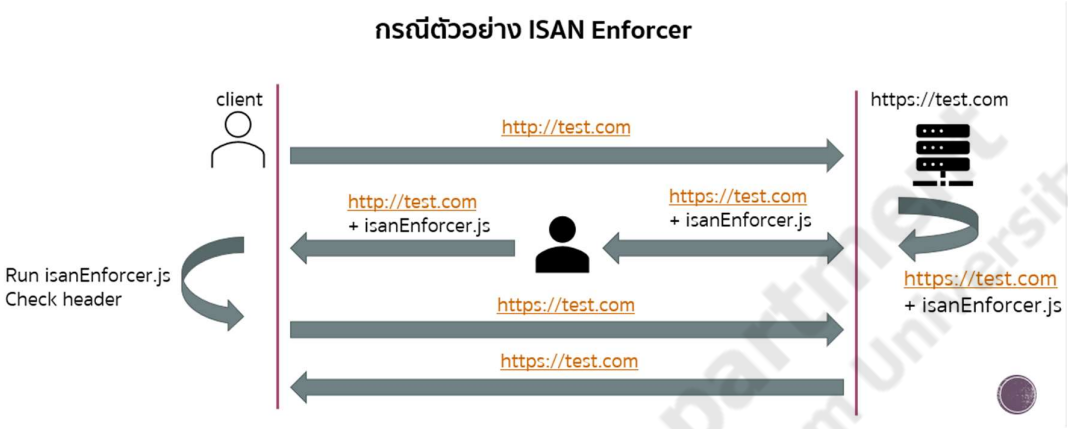
ภาพประกอบที่ 3.6 วิธีลงทะเบียน HSTS Preload



ภาพประกอบที่ 3.7 การทำงานของ HSTS Preload บน Browser

ภาพประกอบที่ 3.7 การทำงานของ HSTS Preload บน Browser เมื่อ Client ร้องขอเว็บไซต์ในรูปแบบของ HTTP เบราว์เซอร์จะทำการตรวจสอบ Domain name ว่าพบในฐานข้อมูลหรือไม่ หากพบ Domain name ดังกล่าวในฐานข้อมูล จะแปลง URL ที่ Client เป็นผู้ร้องขอให้อยู่ในรูปแบบของ HTTPS โดยอัตโนมัติ และรีเทิร์น Header ก่อนเรียกเว็บไซต์ปลายทาง เว็บไซต์ปลายทางที่ได้รับการร้องขอในรูปแบบ HTTPS ก็ตอบกลับในรูปแบบของ HTTPS เช่นกัน แม้มิจฉาชีพจะพยายามปลด HTTPS ออก ก็ไม่เป็นผล เมื่อ URL ที่ตอบกลับมาไม่อยู่ในรูปแบบของ HTTPS เบราว์เซอร์จะปฏิเสธการตอบกลับ

URL ที่ไม่อยู่ในรูป HTTPS และบังคับใช้ HTTPS อีกครั้ง ทำให้การสื่อสารนี้มั่นคงมากยิ่งขึ้น ซึ่งจะแตกต่างตาก HSTS ที่ไม่ได้ทำการ Preload

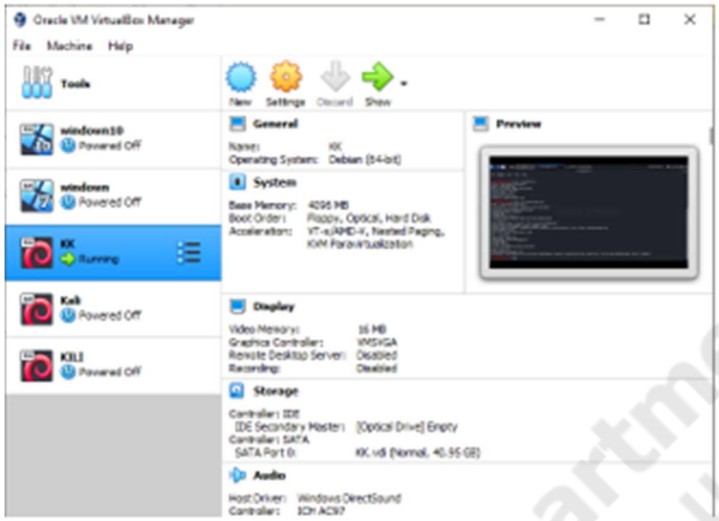


ภาพประกอบที่ 3.8 รูปแบบการป้องกันการปลด HTTPS ด้วย ISAN Enforcer

ภาพประกอบที่ 3.8 ISAN Enforcer เป็นส่วนเสริมที่เข้ามาเพื่อหวังจะป้องกันการโจมตีด้วยเทคนิค SSL Stripping Attack เมื่อผู้ใช้งานร้องขอไปในรูปแบบ HTTP Protocol เว็บไซต์ปลายทางจะทำการ redirect เปลี่ยนเป็น HTTPS และเสริมด้วย isanEnforcer.js ซึ่งเป็นไฟล์ JavaScript แฮคเกอร์ที่อยู่ตรงกลางการสื่อสารปลด HTTPS เป็น http แต่ isanEnforcer.js นั้นยังคงอยู่ แฮคเกอร์ส่งต่อไปยังผู้ใช้งาน isanEnforcer.js จะถูก run ที่ฝั่งของผู้ใช้งาน และ re-check Header ว่าเป็น HTTPS หรือไม่ หากไม่ isanEnforcer.js จะบังคับให้ Header เป็น HTTPS โดยอัตโนมัติซึ่งมีคุณลักษณะของ ISAN Enforcer และ HSTS มีแนวคิดไปในทิศทางเดียวกัน

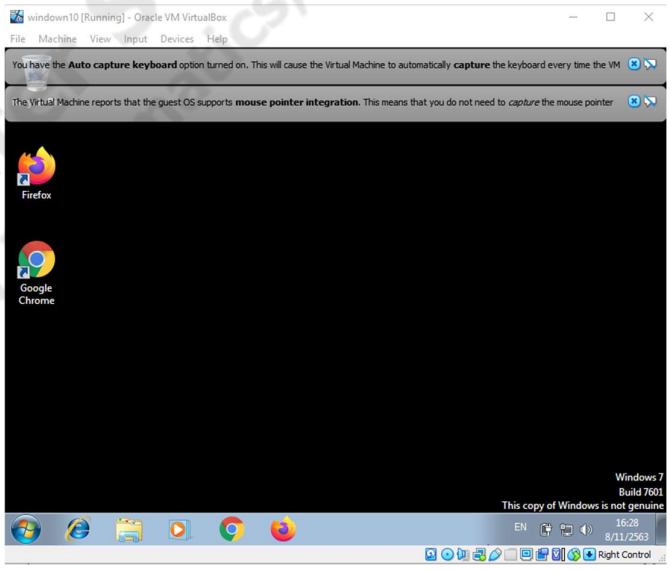
3.3 ตัวอย่างการโจมตีด้วยเทคนิค SSL Stripping Attack

ใช้การจำลอง Operating System (OS) ผ่านโปรแกรม Oracle VM VirtualBox Manager โดยจำลองทั้งเครื่องเหยื่อ และเครื่อง Hacker เพื่อให้ความสะดวกในการทดสอบการโจมตี



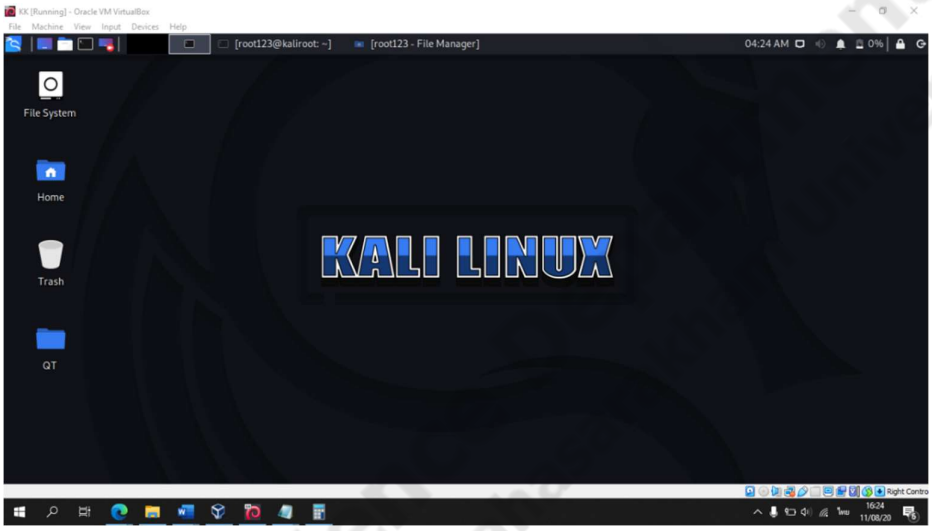
ภาพประกอบที่ 3.9 การจำลองอุปกรณ์ของเหยื่อ และ Hacker บน Linux OS

ภาพประกอบที่ 3.9 เป็นการเรียกใช้โปรแกรมที่ใช้ในการจำลองอุปกรณ์เครื่องเหยื่อ และ Hacker โดยอุปกรณ์ของ Hacker ใช้ระบบปฏิบัติการลินุกซ์ Linux Operating System (Linux OS) และอุปกรณ์ของเหยื่อใช้ระบบปฏิบัติการวินโดว์ Windows Operating System (Windows OS) จำลองผ่านโปปรแกรม Oracle VM VirtualBox Manager



ภาพประกอบที่ 3.10 หน้าจอของเหยื่อ

ภาพประกอบที่ 3.10 เป็นภาพหน้าจอจำลองของเหยื่อ (Victim) จำลองผ่านโปรแกรม Oracle VM VirtualBox Manager โดยอุปกรณ์ของเหยื่อจำลองให้ใช้ระบบปฏิบัติวินโดว์ Windows Operating System (Windows OS) เวอร์ชัน 7 และใช้ Google Chrome Browser ในการทดสอบการโจมตีเป็นส่วนใหญ่ เนื่องจากมีการอัพเดตความปลอดภัยจาก Google Service ตลอด อีกทั้งยังเป็นที่นิยมสูงในการใช้งานของผู้ใช้



ภาพประกอบที่ 3.11 หน้าจอของ Hacker

ภาพประกอบที่ 3.11 Hacker ใช้ระบบปฏิบัติการลินุกซ์ Linux Operating System (Linux OS) เพื่อสามารถเรียกใช้คำสั่งอื่น ๆ ที่ใช้ในการโจมตี บน Kali และดักจับ IP โดยเรียกใช้ Tools ของ Kali ได้

3.3.1 ตัวอย่างการโจมตีด้วย BetterCap Tools

1) ดักจับหมายเลข IP ทั้งบนที่อยู่บนเครือข่าย LAN เดียวกัน

IP ▲	MAC	Name	Vendor	Sent	Recvd	Seen
192.168.43.45	08:00:27:3d:02:31	eth1	PCS Computer Systems GmbH	0 B	0 B	02:49:35
192.168.43.1	ec:89:14:68:bd:c1	gateway	Huawei Technologies Co.,Ltd	43 kB	44 kB	02:49:35
192.168.43.72	10:f0:05:9a:c1:62	DESKTOP-I25K9J9.	Intel Corporate	174 kB	30 kB	03:03:43
192.168.43.135	08:00:27:99:f6:bb	John-PC.	PCS Computer Systems GmbH	1.2 MB	4.6 MB	03:03:35

↑ 1.4 MB / ↓ 14 MB / 115019 pkts

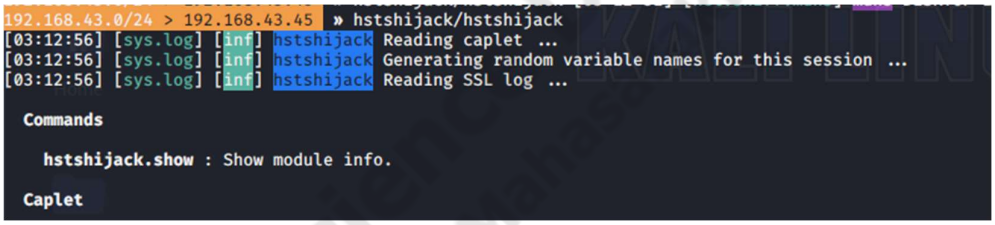
ภาพประกอบที่ 3.12 หมายเลข IP ทั้งหมดบนเครือข่าย LAN เดียวกัน

2) ระบุหมายเลข IP ของเป้าหมายที่ต้องการโจมตี

ภาพประกอบที่ 3.13 การระบุเป้าหมายที่ต้องการโจมตี

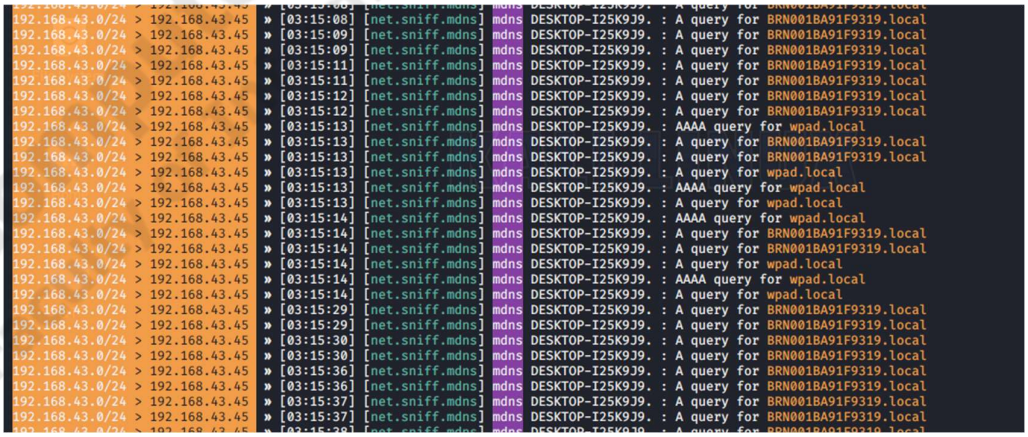
ภาพประกอบที่ 3.13 เป็นระบุหมายเลข IP ของเป้าหมายที่ต้องการโจมตี โดย IP ที่ใช้ในการระบุเป้าหมาย จะใช้ IPv4 เท่านั้น และหมายเลข IP ที่ระบุต้องเป็นเป้าหมายเลขที่อยู่บนเครื่องข่าย LAN เดียวกันที่ได้จากการดักจับหมายเลข IP ดัง
ภาพประกอบที่ 3.12เท่านั้น

3) ปลด HTTPS Protocol ให้อยู่ในรูปแบบ HTTP Protocol เพื่อให้ Username และ Password ของเหยื่ออยู่ในรูปแบบของข้อมูลที่สามารถอ่านได้ (Clear text) จากนั้นรอให้เป้าหมายที่เลือกไว้เชื่อมต่อเข้าไปยังเว็บไซต์ปลายทาง

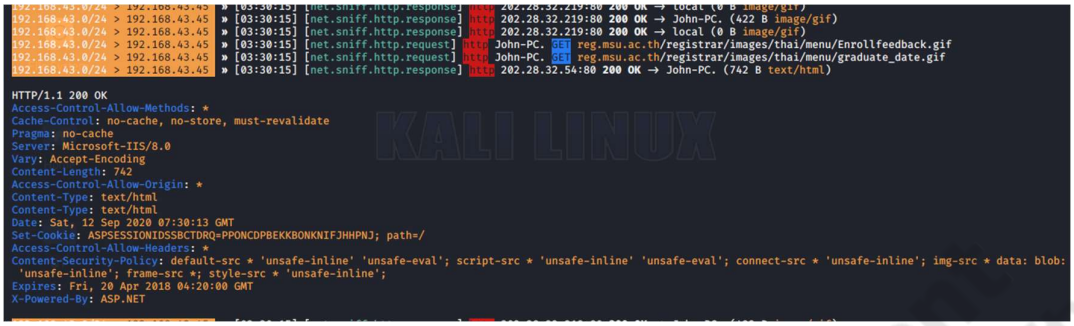


ภาพประกอบที่ 3.14 การปลด HTTPS

4) เมื่อมีการเชื่อมต่อไปยังเว็บไซต์ปลายทางจะมีการแสดงข้อมูลที่แสดงการเชื่อมต่อไปยังเว็บไซต์ปลายทาง



ภาพประกอบที่ 3.15 Data Packet

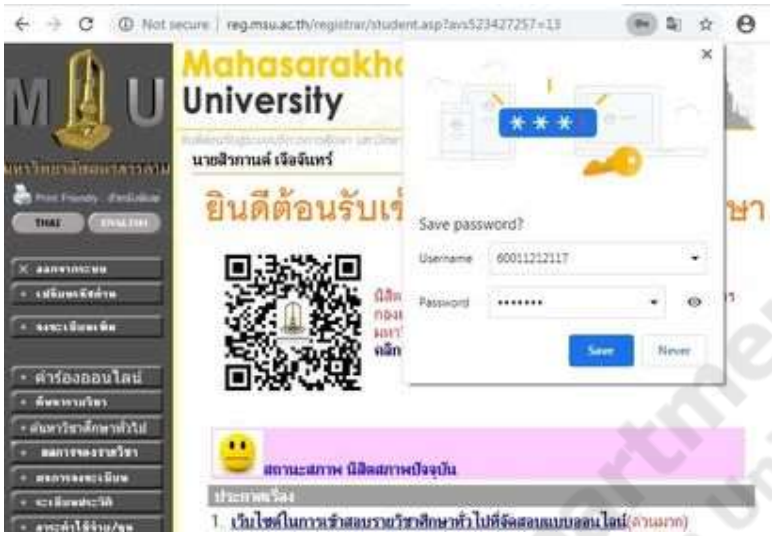


ภาพประกอบที่ 3.16 Data Packet เมื่อเข้าสู่เว็บไซต์

5) เมื่อเหยื่อทำการเชื่อมต่อไปยังเว็บไซต์ปลายทางจะขึ้นเตือนที่มุมซ้ายของ URL tab ว่า Not Secure ดังภาพประกอบที่ 3.17

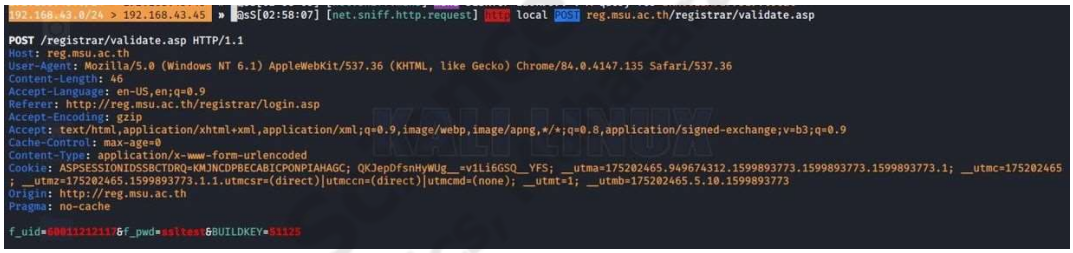
ภาพประกอบที่ 3.17 หน้าเข้าสู่ระบบของเว็บไซต์ปลายทางของอุปกรณ์เหยื่อ

หากเหยื่อกรอกข้อมูล Username และ Password กับหน้าเว็บไซต์เข้าสู่ระบบดังกล่าว โดยไม่ได้สังเกตเครื่องหมายเตือนของเบราว์เซอร์ Username และ Password จะถูกส่งไปตรวจสอบยังเว็บไซต์ปลายทาง หาก Username และ Password ถูกต้อง ก็จะสามารถเข้าสู่ระบบได้ตามปกติ และถูก Hacker ดักจับข้อมูลดังภาพประกอบที่ 3.18



ภาพประกอบที่ 3.18 เข้าสู่ระบบสำเร็จ

6) แสดงผลการดักจับ Username และ Password ของเหยื่อ



ภาพประกอบที่ 3.19 ผลการดักจับข้อมูล Username และ Password

3.4 การวิเคราะห์ผลการทดสอบการโจมตีด้วยเทคนิค SSL Stripping Attack

3.4.1 นำผลการทดสอบการโจมตีด้วยเทคนิค SSL Stripping Attack ทั้งหมดมาเปรียบเทียบว่าจากการทดสอบการโจมตี เว็บไซต์ใดบ้างที่สามารถโจมตีได้ และเว็บไซต์ใดบ้างที่สามารถป้องกันการโจมตีดังกล่าวได้

3.4.2 การตรวจสอบ Header ของเว็บไซต์ทั้งหมดที่ทำการทดสอบ

1) ตรวจสอบว่าได้ทำการ Preload HSTS ไว้หรือไม่

ภาพประกอบที่ 3.20 ตัวอย่างเว็บไซต์ที่ Preload HTTPS Header

ภาพประกอบที่ 3.21 ตัวอย่างเว็บไซต์ที่ไม่ได้ Preload HSTS Header

ตรวจสอบ Certificate Authority (CA) ว่ามีความน่าเชื่อถือหรือไม่ Certificate Authority (CA) ที่มีความน่าเชื่อถือ ได้แก่ COMODO, DigiCert, GeoTrust, GlobalSign, Certum, Thawte, Symantec เป็นต้น จากการทดสอบการโจมตีเว็บไซต์ต่าง ๆ เพื่อนำมาวิเคราะห์ ทางผู้พัฒนาโครงการได้เล็งเห็นถึงจุดบกพร่องของความปลอดภัยทางด้านเว็บไซต์ แต่เดิมเห็นว่า Response Headers สามารถตระหนักถึงความปลอดภัยของเว็บไซต์โดยการ Preload Header ผ่าน Strict Transport Security แต่ถ้าหาก Preload ลงใน Header แล้วแต่ไม่ได้ไป set ลงใน Database การป้องกันการ Strip ถึงไม่เป็นผล

2) ตรวจสอบวันจดทะเบียน และหมดอายุสัญญาของการจดทะเบียน CA

ภาพประกอบที่ 3.22 ตัวอย่าง Header ของเว็บไซต์

3) ตรวจสอบ Algorithm ที่ใช้ในการเข้ารหัส

ภาพประกอบที่ 3.23 ตัวอย่าง Algorithm ที่ใช้ในการเข้ารหัส

3.4.3 นำมาข้อมูลที่ได้จากการทดสอบการโจมตี และการวิเคราะห์ข้อมูลเปรียบเทียบว่าเว็บไซต์ที่ป้องกันการโจมตีได้ และเว็บไซต์ที่ไม่สามารถป้องกันได้ มีส่วนใดบ้างที่เหมือน หรือแตกต่างกัน

ตารางที่ 3.2 การวิเคราะห์ผล HSTS กับการโจมตีด้วย SSLStripping Attack

Domain Name	HSTS	Strip	Sniff
Msu.ac.th	x	/	/
Kasikornbankgroup.com	/ (not Preload)	/	x
Ktbnetbank.com	/	X	x

ตารางที่ 3.3 สรุปผล HSTS Header

HSTS	Strip HSTS	Sniff
No	/	/
Yes (not Preload)	/	/
Yes (not Preload)	/	x (encrypt)
Yes (Preload)	x	x

ตารางที่ 3.3 การวิเคราะห์เกี่ยวกับ HSTS Header เบื้องต้น พบว่า HSTS ที่ถูกกำหนดไว้เป็นเพียงส่วนประกอบเสริม เนื่องจาก HSTS ที่กำหนดไว้ หากไม่ได้ Preload ไว้บน Google cloud จะไม่สามารถป้องกันการโจมตีจากเทคนิค SSL Stripping Attack เพราะ Browser นั้นจะตรวจสอบกับ HSTS Preload list ที่ถูกลงทะเบียน Preload ไว้เท่านั้น เพื่อป้องกันการถูกปลด HSTS โดย BetterCap ที่ใช้สำหรับการเพิ่ม ลบ Header และป้องกันการเพิ่ม HSTS Header ให้กับเว็บไซต์ที่เป็น HTTP การเพิ่ม HSTS Header ให้กับเว็บไซต์ HTTP นั้นส่งผลให้เว็บไซต์ไม่ตอบสนอง

3.5 การสร้างเว็บไซต์ล็อกอินต้นแบบในการป้องกันการโจมตีด้วยเทคนิค SSL Stripping Attack

การสร้างเว็บไซต์ต้นแบบนั้น เพื่อเป็นใช้เป็นกรณีศึกษา โดยเว็บไซต์ล็อกอิน ประกอบด้วย 3 องค์ประกอบ คือ

3.5.1 เสริมการป้องกันด้วยการกำหนด HSTS

HSTS ย่อมาจาก HTTP Strict Transport Security ใช้ในการควบคุมให้เบราว์เซอร์เชื่อมต่อกับเว็บไซต์ปลายทางที่มีในฐานข้อมูลด้วย HTTPS Protocol เท่านั้น

3.5.2 เสริมการป้องกันด้วย ISAN Enforcer โดย JavaScript

ISAN Enforcer ได้ถูกพัฒนาให้อยู่ในรูปของ API ด้วย JavaScript เพื่อความสะดวกในการพัฒนาเว็บไซต์ให้มั่นคง สามารถรองรับการทำงาน และป้องกันการโจมตีที่ทำงานบน HTTPS จากการโจมตีด้วยเทคนิค SSL Stripping Attack ได้ การเสริมการป้องกันด้วย ISAN Enforcer มีหลักการทำงานที่ใกล้เคียงกับการเสริมการป้องกันด้วย HSTS ที่ยังไม่ได้ Preload ขึ้นบนเบราว์เซอร์ เป็นการป้องกันในระดับเบื้องต้น

ภาพประกอบที่ 3.24 กลไกการทำงานของ ISAN Enforcer

ภาพประกอบที่ 3.24 จะเห็นได้ว่าเมื่อ Client ร้องขอไปยังเว็บไซต์ปลายทางในรูปแบบ HTTP เว็บไซต์ปลายทางจะตอบกลับมาในรูปแบบ HTTPS ที่เสริมการป้องกันด้วย ISAN Enforcer เมื่อ Hacker ที่อยู่ตรงกลางการสื่อสารระหว่าง Client และ Web Server โจมตีด้วยเทคนิค SSL Stripping Attack ทำให้ URL ที่เว็บไซต์ปลายทางตอบกลับไปยัง Client ที่ควรจะตอบกลับมาในรูปแบบ HTTPS แต่กลับได้ URL ที่ตอบกลับมาเป็น HTTP ที่มี ISAN Enforcer เมื่อมาถึงทางฝั่ง Client แม้ Client จะไม่สังเกตเห็นถึงความผิดปกติ ISAN Enforcer จะถูกประมวลผลอัตโนมัติเมื่อมาถึง Client และทำการ Re – Check ตัวเองว่าเป็น HTTPS หรือไม่ หากไม่ได้อยู่ในรูปแบบของ HTTPS จะบังคับให้เรียกเว็บไซต์ปลายทางในรูปแบบ HTTPS และร้องขอไปยัง Web Server อีกครั้ง

3.5.3 เสริมการป้องกันด้วยการเข้ารหัสด้วยเทคนิค Salted – Hash Password (SHP)

1) นำ Username มาเข้ารหัสด้วยเทคนิคการเข้ารหัส SHA – 512 สามารถเขียนเป็นสมการได้ ดังนี้ $x = hash(Username)$

2) นำ Password มาเข้ารหัสด้วยเทคนิคการเข้ารหัส SHA – 512 สามารถเขียนเป็นสมการได้ ดังนี้ *y = hash(Password)*

3) นำ Password ที่เข้ารหัสแล้ว + Username ที่เข้ารหัสแล้วมาต่อกันแล้วนำมาเข้ารหัสเทคนิคการเข้ารหัส SHA – 512 สามารถเขียนเป็นสมการได้ ดังนี้ *Hash_Function =hash(y + x)*

4) นำ Username + Hash_Function เพื่อกำหนด Salted - Hash Password (SHP) สามารถเขียนเป็นสมการได้ ดังนี้ *Salt = Username + Hash_Function*

5) นำ Password รวมกับ Salt นำมาเข้ารหัสด้วย SHA – 512 จะได้ค่า Pass_salt สามารถเขียนเป็นสมการได้ดังนี้ *Pass_salt = hash(Password + Salt)*

6) เก็บข้อมูลลงในฐานข้อมูล (Database)

6.1) Database 1 (se_salt_ssl) เก็บค่า Salt หรือ Username + Hash_Function และ Username

6.2) Database 2 (se_ssl) เก็บค่า Username และ Pass_salt

การเข้ารหัสของ Hash_Function, Salt และ Pass_salt เป็นการเข้ารหัสแบบซ้อน ซึ่งค่า hash(Username) หรือ x และ hash(Password) หรือ y ใช้เทคนิคการเข้ารหัส SHA-512 เพื่อการป้องกันที่มั่นคง เนื่องจากความสามารถในการโจมตีเป็นไปได้ยากด้วยสมรรถนะของคอมพิวเตอร์ในปัจจุบัน Hash_Function นำเอาค่า y และ x มารวมกัน และ เข้ารหัสด้วย SHA-512 อีก 1 ชั้น Salt ได้จากการนำ Username + Hash_Function มารวมกัน และ Pass_salt ได้จากการนำค่า Password มารวมกับ Salt

Salted – Hash Password โดยทั่วไปก็ยากที่จะถอดรหัส แต่การเสริมความมั่นคงด้วยการใช้การเข้ารหัสด้วยเทคนิค SHA – 512 ซึ่งเป็นการเข้ารหัสที่ยังได้รับการยอมรับในปัจจุบัน อีกทั้งเป็นอัลกอริทึมที่มีประสิทธิภาพในการเข้ารหัสด้วย

3.6 วิธีการป้องกันเทคนิคการโจมตี SSL Stripping Attack

3.6.1 HSTS Preload

กำหนด Header HSTS ให้กับเว็บไซต์และนำเว็บไซต์ขึ้น HSTS Preload Strict Transport Security เพื่อประโยชน์ด้านการรักษาความมั่นคง และความปลอดภัยโดยเฉพาะบนเครือข่ายที่ไม่เป็นมิตร

กำหนดนโยบาย HSTS บนโดนเมน และโดเมนย่อยด้วย Include Subdomain และ Preload ทุกส่วนของเว็บไซต์ก็จะเป็นเว็บไซต์ที่ใช้งานผ่าน HTTPS Protocol และอนุญาตเบราว์เซอร์ให้บังคับใช้อย่างจริงจัง

3.6.2 ISAN Enforcer

ISAN Enforcer เมื่อมีการติดตั้ง ISAN Enforcer ลงไปในเว็บไซต์จะทำให้เว็บไซต์บังคับใช้ HTTPS ตลอดเวลา เมื่อ HSTS Preload ถูกทำลาย จะมีการใช้ ISAN Enforcer โดยการทำงานจะบังคับให้ Client เข้าใช้งานในรูปแบบ HTTPS เท่านั้น ซึ่งถือเป็นด่านป้องกันสำรองอีก 1 ด่าน หากเกิดกรณีที่ HSTS Preload ถูกทำลาย

```
window.onload = function(){
   checkHTTPS();
   checkHTTPSLink();
}
function checkHTTPS(){
   if(window.location.protocol != 'https:'){
                var restOfUrl = window.location.href.substr(5);
        window.location = "https:" + restOfUrl;
        }
}
function checkHTTPSLink(){
    var url = window.location.href;
    if(url.indexOf("https://")==0) {
        for(var i=0,link; (link=document.links[i]); i++) {
            if(link.href.indexOf("http://")==0)
            link.href = link.href.replace(link.href.substring(0,7), "https://");
        }
    }
}
```

ภาพประกอบที่ 3.25 ISAN Enforcer Script

ภาพประกอบที่ 3.25 สามารถอธิบายได้ว่า

1) Window.onload = function() เป็นการสั่งให้ทำงานฟังก์ชันนี้ทันทีที่โหลดสำเร็จ
2) Function checkHTTP() เป็นฟังก์ชันที่ใช้ในการตรวจสอบเงื่อนไขว่าหาก Protocol ไม่ใช่ HTTPS ให้ตัด 5 ตัวแรกออก หรือก็คือ http: และใส่ HTTPS: เข้าไปแทน
3) Function checkHTTPLink() เป็นฟังก์ชันที่ใช้ในการตรวจสอบ URL โดยตรวจสอบเงื่อนไขของ URL ถ้าตำแหน่งของ “HTTPS://” หากตำแหน่งของคำว่า “HTTPS://” อยู่ในตำแหน่ง index ที่ 0 ให้วนลูปเพื่อตรวจสอบ link ทั้งหมด ถ้า link มี “http://” อยู่ในตำแหน่งแรก substring ตำแหน่ง index ที่ 0 ถึง 6 และแทนที่ด้วย “HTTPS://”

3.6.3 เข้ารหัสด้วยเทคนิค Salted – Hash Password

การเข้ารหัสโดยใช้เทคนิค Salted – Hash Password ด้วย JavaScript ซึ่ง JavaScript จะทำงานในฝั่งของ Client ทำให้มีการเข้ารหัสก่อนส่งไปยัง Server ทำให้แม้ Hacker จะสามารถดักฟังการสื่อสารของ Client ได้แต่จะต้องมีการถอดรหัสอีกหลายขั้นตอน ทางผู้จัดทำมีการใช้ Script ของ Jeff Mott ซึ่งมีการอัพเดทล่าสุดเมื่อปี 2013

```
app.post( path: "/register",cors(),async (req : Request<P, ResBody, ReqBody, ReqQuery, Loca
    //process register
    var {username, pwd} = req.body;
    console.log(username, pwd)

    var Hash_username = CryptoJS.SHA512(username);
    console.log("Hash_username => " + Hash_username)

    var Hash_password = CryptoJS.SHA512(pwd);
    console.log("Hash_password => " + Hash_password)

    //  Hash Function
    var pu = Hash_password + Hash_username
    var Hash_PU = CryptoJS.SHA512(pu);
    console.log("Hash_Function => " + Hash_PU)

    //Salted Hash
    var SHP = username + Hash_PU
    console.log("SHP_Hash => " + SHP)

    //passwordHash on DB
    var passSalt = pwd + SHP
    var PassSalt = CryptoJS.SHA512(passSalt)
    console.log("PassSalt => " + PassSalt)
```

ภาพประกอบที่ 3.26 การเข้ารหัสด้วยอัลกอริทึม SHA – 512

ภาพประกอบที่ 3.26 ใช้อัลกอริทึม SHA – 512 ในการเข้ารหัส และ CryptoJS library สำเร็จรูปของ JavaScript ที่รวบรวมการเข้ารหัสมาตรฐานเอาไว้

3.7 การทดสอบการป้องกันการโจมตีด้วยเทคนิค SSL Stripping Attack

เป็นการทดสอบการโจมตีด้วยเทคนิค SSL Stripping Attack กับเว็บไซต์ล็อกอินต้นแบบที่กำหนดค่า HSTS และ ISAN Enforcer จะสามารถทนต่อการถูกโจมตีด้วยเทคนิคดังกล่าวหรือไม่ หากการป้องกันทั้ง 2 ถูกปลดออกได้ ก็จะยังสามารถรักษาข้อมูลที่เป็นความลับต่อไปได้ เนื่องจากมีการเข้ารหัสด้วยเทคนิค Salted - Hash Password (SHP)

3.8 การสรุปผลกลไกการป้องกัน

เป็นการสรุปว่าเว็บไซต์ล็อกอินต้นแบบจะสามารถทนต่อการโจมตีได้มากน้อยแค่ไหน และหากถูกปลด HSTS และ ISAN Enforcer แล้วจะเป็นอย่างไร โดยใช้ตารางในการเปรียบเทียบประสิทธิภาพในการป้องกันระหว่างเว็บไซต์ล็อกอินต้นแบบ และเว็บไซต์ที่ใช้ในการทดสอบ

3.9 การสร้างโปรแกรมที่ใช้ในการตรวจสอบหาช่องโหว่

1) ใช้ Python เป็นสื่อกลางในการสื่อสาร สำหรับเรียกในคำสั่งบน Linux OS

2) ใช้ Python QT Designer เป็น Interface เพื่อให้ผู้ใช้สามารถเข้าใจข้อมูลได้

3) สร้างสื่อวิดีโอที่ใช้สำหรับการอธิบายถึงปัญหา และวิธีการแก้ไขปัญหาของการโจมตีด้วยเทคนิค DROWN Attack, POODLE Attack และ BEAST Attack โดยอัพโหลดไฟล์ไว้บน YouTube

สร้างสื่อวิดีโอที่ใช้ในการอธิบายถึงปัญหาของ SSL Stripping Attack และการกำหนดค่า HSTS ให้ถูกต้อง และการกำหนดค่า ISAN Enforcer โดยอัพโหลดไฟล์ไว้บน YouTube

```
def Header_ch(self):
    print('Header_run')
    url = (self.urlscan)
    p = os.popen('curl -I ' + url).read().lower()
    # print(p)

    for line in p.splitlines():
        if 'strict' in line:
            print(line)
            self.label_hsts.setText(line)
            k = 'hsts'
            self.prehsts = k
            break
        else:
            self.label_hsts.setText("No Header")
            self.label_hsts.setStyleSheet("color: #ff0000")
            k = 'nohsts'
            self.prehsts = k
            print('No Preload')
```

ภาพประกอบที่ 3.27 ฟังก์ชันตรวจสอบ Header

ภาพประกอบที่ 3.27 เป็นการตรวจสอบ Header ว่ามี HSTS หรือไม่ หากมี Preload หรือไม่ โดยใช้คำสั่ง curl -I เพื่ออ่านค่า Header ของ URL ที่รับข้อมูลเข้ามา ซึ่ง curl -I เป็นฟังก์ชันสำหรับใช้ในการติดต่อสื่อสารกับ server โดยสามารถติดต่อได้หลากหลาย Protocol เช่น http, HTTPS, fatp เป็นต้น

```python
def testssl(self):
    print("testssl run")
    url = (self.urlscan)
    p = os.popen('testssl ' + url).read()
    for line in p.splitlines():
        if 'POODLE' in line:
            if 'OK' in line:
                self.label_pd.setText('Secure')
                self.label_pd.setStyleSheet("color: #059142")
            else:
                self.label_pd.setText('Insecure')
                self.label_pd.setStyleSheet("color: #ff0000")
        if 'DROWN' in line:

            if 'OK' in line:
                self.label_da.setText('Secure')
                self.label_da.setStyleSheet("color: #059142")
            else:
                self.label_da.setText('Insecure')
                self.label_da.setStyleSheet("color: #ff0000")
        if 'BEAST' in line:
            if 'OK' in line:
                self.label_ba.setText('Secure')
                self.label_ba.setStyleSheet("color: #059142")
            else:
                self.label_ba.setText('Insecure')
                self.label_ba.setStyleSheet("color: #ff0000")
        if 'Done' in line:
            # self.Welcome.setText(self.urlscan + ' Successful ')
            self.progressBar.setValue(90)
            self.result_scan()
```

ภาพประกอบที่ 3.28 ฟังก์ชันตรวจสอบหาช่องโหว่ต่อเทคนิค Downgraded Attack

ภาพประกอบที่ 3.28 เป็นฟังก์ชันที่ใช้ในการตรวจสอบหาช่องโหว่ที่เสี่ยงต่อการถูกโจมตีด้วยเทคนิค Downgraded Attack โดยเรียกใช้ API ในการตรวจสอบหาช่องโหว่ หากพบว่ามีการอนุญาตให้ใช้ SSL/TLS ตั้งแต่ TLS Version 1.0 ลงไปให้แสดงคำว่า Insecure ตามช่องโหว่ที่เป็นภัยต่อการโจมตีแต่ละการโจมตี

```
def CA_cert(self):
    print("CA run")
    url = (self.urlscan)
    print(url)
    w = whois.whois(url)
    ss = w.domain
    url_www = "www." + ss
    print("url_www : " + url_www)
    p = os.popen('nmap --script ssl-cert -p 443 ' + url_www).read()
    for line in p.splitlines():
        if 'Issuer' in line:
            print(line[21:])
            self.label_ca.setText(line[21:])
        if 'before' in line:
            before = line[20:30]
        elif 'after' in line:
            after = line[20:30]
            print(before + ' TO ' + after)
            self.label_vf.setText(before + ' TO ' + after)
    self.progressBar.setValue(50)
    self.testssl()
```

ภาพประกอบที่ 3.29 ฟังก์ชันรับข้อมูล Certificate Authority

ภาพประกอบที่ 3.29 ฟังก์ชัน CA_cert ใช้ในการแสดงข้อมูลเกี่ยวกับ CA ของ Web Server เก็บค่า Domain name เรียกใช้ฟังก์ชัน test SSL

```
def read_mozilla(self):
    link = ('https://raw.githubusercontent.com/mozilla/gecko-dev/master/security/manager/ssl/nsSTSPreloadList.inc')
    with urllib.request.urlopen(link) as f:
        s = f.read()
    s1 = s.decode('utf8')
    # url = ('https://ktbnetbank.com/consumer')
    url = (self.urlscan)
    w = whois.whois(url)
    print(w.domain)

    for line in s1.splitlines():
        if w.domain in line:
            # print('Preload in Mozilla')
            pre_Mo = 'pre_Mo'
            self.pre_Mo = pre_Mo
            break
        else:
            # print('No Preload Mozilla')
            pre_Mo = 'nopre_Mo'
            self.pre_Mo = pre_Mo
    self.progressBar.setValue(35)
    self.pre_ChANDpre_MO()
```

ภาพประกอบที่ 3.30 ตรวจสอบหา HSTS Preload บน Mozilla Firefox

3.10 แนะนำการใช้ HTTP Strict Transport Security (HSTS)

HTTP Strict Transport Security (HSTS) เป็นมาตรฐานบนเครือข่ายที่เริ่มมีผู้คนให้ความสนใจทั้งในองค์กรขนาดเล็ก และขนาดใหญ่ เนื่องจากปัจจุบันมีการเก็บข้อมูลสำคัญมากมายในรูปแบบดิจิทัล

จึงจำเป็นต้องมีความมั่นคง และปลอดภัยในการเก็บรักษาข้อมูลยิ่งขึ้น อีกทั้ง HSTS เป็นมาตรฐานช่วยเพื่อความมั่นคงให้กับเว็บไซต์ที่ได้รับความเชื่อใจ

ในการเพิ่มประสิทธิภาพให้กับเว็บไซต์ต้องเพิ่มชุดคำสั่งเพื่อบอก HTTP Header เพื่อสั่งการ Browser คือ Strict Transport Security: max-age=expireTime; includeSubDomains; Preload

3.10.1 อธิบายเกี่ยวกับแต่ละส่วนของชุดคำสั่ง

1) max-age (จำเป็นต้องระบุ) คือ ตัวบ่งชี้ระยะเวลาว่าจะให้เว็บไซต์นั้นรักษาระยะเวลาที่จะคง HTTPS ไว้นานเท่าไหร่

2) expireTime คือ เวลาในหน่วยวินาทีที่จะใช้ในการรักษาสภาพ HTTPS ของเว็บไซต์ เช่น

5 นาที max-age = 300;

1 วัน max-age = 86,400;

1 สัปดาห์ max-age = 604,800

1 เดือน max-age = 2,592,000;

1 ปี max-age = 31,536,000;

3) includeSubDomains (สามารถไม่ระบุได้) คือ การเพิ่ม Sub Domains ให้ใช้งาน HSTS ได้

4) Preload (สามารถไม่ระบุได้) คือ การนำเว็บไซต์ไป Preload HSTS บน Browser ของผู้ใช้บริการ ผู้ให้บริการ HSTS Preload มี Google เป็นผู้ดูแลรายชื่อ Domain ผ่าน HTTPS://hstsPreload.org มีเบราว์เซอร์ของผู้ให้บริการ เช่น Google Chrome, FireFox, IE 11 และ Edge เป็นต้น

3.10.2 การใช้งาน HSTS ใน Web Server

1) Apache สำหรับ Apache ต้องเปิด (mod_Headers) เพิ่มในส่วน Web Server Config หรือ vhost คือ

Header always set Strict Transport Security: max-age = expireTime หรือหากมีโดเมนย่อย

Header always set Strict Transport Security: max-age = expireTime; includeSubDomains

2) Nginx สำหรับ nginx ต้องเปิด (ngx_http_Headers_module) เพิ่มในส่วน Web Server Config หรือ vhost คือ

add_Header Strict Transport Security: max-age = expireTime หรือหากมีโดเมนย่อย

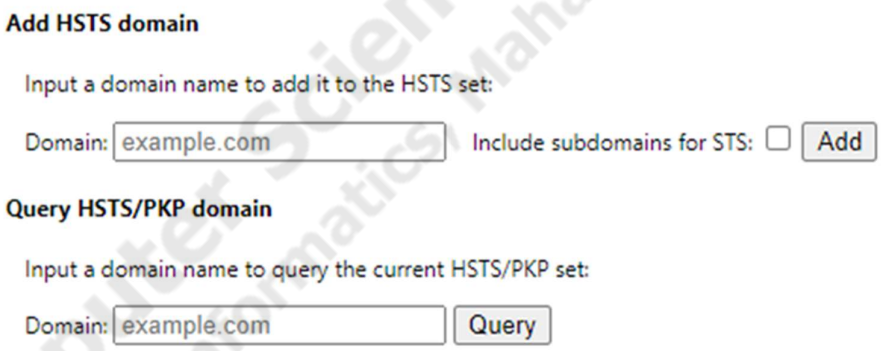
add_Header Strict Transport Security: max-age = expireTime; includeSubDomains

3) IIS สำหรับ IIS แก้ไขที่ Web.config

```
<httpProtocol>
    <customHeaders>
    <add name="Strict Transport Security" value="max-age=expireTime" />
```

```
        </customHeaders>
</httpProtocol> หรือหากมีโดเมนย่อย
<httpProtocol>
        <customHeaders>
        <add name="Strict Transport Security" value="max-age=expireTime;
        includeSubDomains" />
 </customHeaders>
</httpProtocol>
```

เพื่อให้มีคุณสมบัติในการ Preload การตรวจสอบให้แน่ใจว่าโดเมนย่อยทั้งหมดอยู่ภายใต้การรับรอง SSL/TLS ที่ให้บริการผ่าน HTTPS Protocol และสามารถตรวจสอบว่า HSTS ทำงานอย่างถูกต้องหรือไม่ โดยการเปิดใช้งานส่วน Header ไปที่ chrome://net-internals/#hsts หรือ edge://net-internals/#hsts หรือเบราว์เซอร์ใด ๆ ที่ให้บริการ Preload HSTS ไปยังเมนู “Domain Security Policy” และป้อนเว็บไซต์ลงในเครื่องมือ “Query HSTS/PKP domain” หากแสดงผลลัพธ์แสดงว่า HSTS Preload ถูกเปิดใช้งาน



ภาพประกอบที่ 3.31 เพิ่ม Domain name และตรวจสอบ Domain name