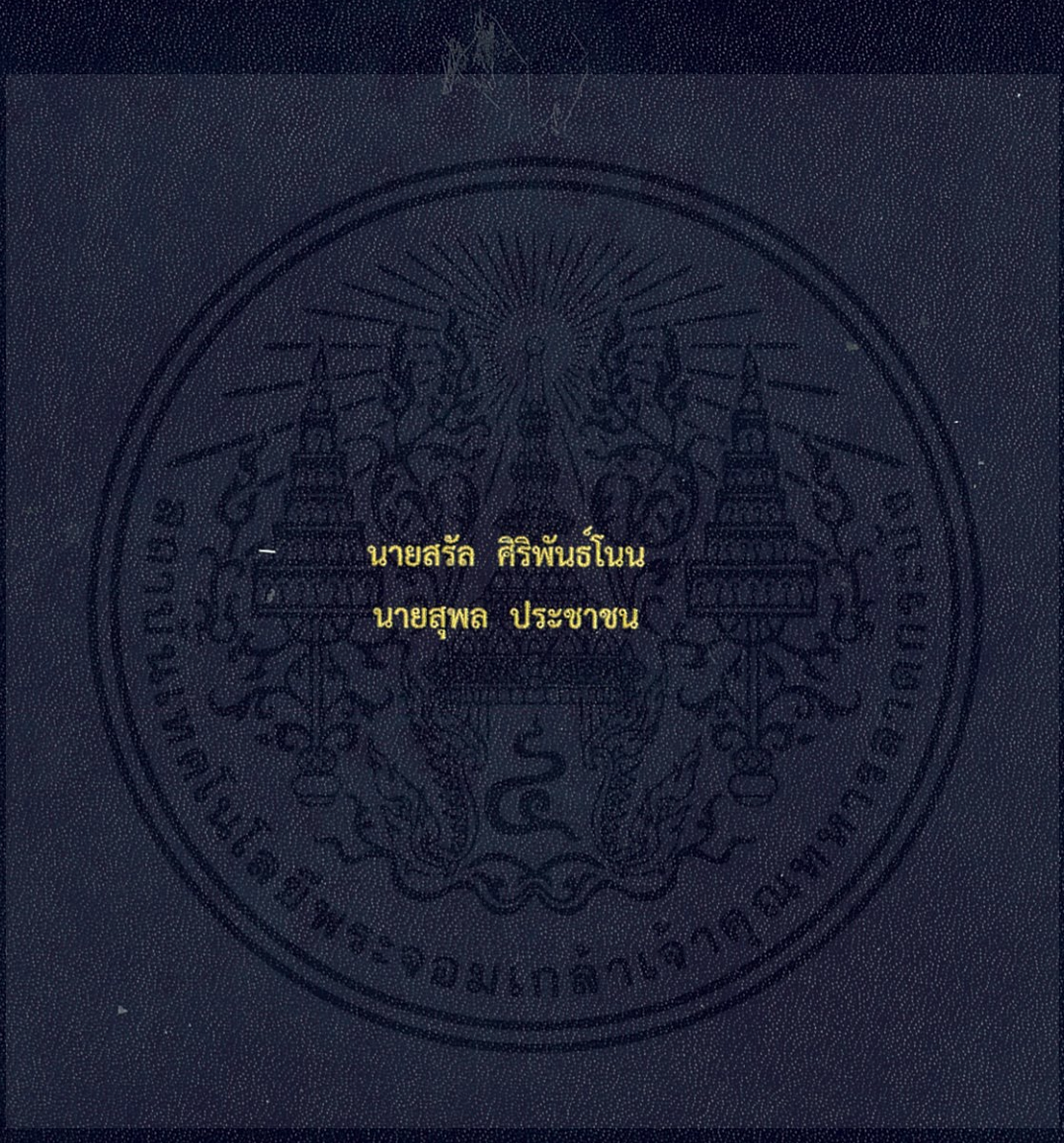
การประเมินความปลอดภัยระบบคอมพิวเตอร์แบบอัตโนมัติ

AUTOMATIC COMPUTER SYSTEM SECURITY ASSESSMENT



นายสรัล ศิริพันธ์โนน

นายสุพล ประชาชน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

การประเมินความปลอดภัยระบบคอมพิวเตอร์แบบอัตโนมัติ

AUTOMATIC COMPUTER SYSTEM SECURITY ASSESSMENT

นายสรัล ศิริพันธ์โนน

นายสุพล ประชาชน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

ปริญญานิพนธ์ปีการศึกษา 2557

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประเมินความปลอดภัยระบบคอมพิวเตอร์แบบอัตโนมัติ

AUTOMATIC COMPUTER SYSTEM SECURITY ASSESSMENT

ผู้จัดทำ

1. นายสรัล ศิริพันธ์โนน รหัสนักศึกษา 54011340

2. นายสุพล ประชาชน รหัสนักศึกษา 54011407

_____ อาจารย์ที่ปรึกษา

(ดร. อักฤทธิ์ สังข์เพ็ชร)

_____ อาจารย์ที่ปรึกษาร่วม

(ดร. อรทัย สังข์เพ็ชร)

การประเมินความปลอดภัยระบบคอมพิวเตอร์แบบอัตโนมัติ

นายสรัล	ศิริพันธ์โนน	54011340
นายสุพล	ประชาชน	54011407
ดร. อักฤทธิ์	สังข์เพ็ชร	อาจารย์ที่ปรึกษา
ดร. อรทัย	สังข์เพ็ชร	อาจารย์ที่ปรึกษาร่วม

ปีการศึกษา 2557

บทคัดย่อ

การนำเครื่องมือทางด้านความปลอดภัยต่าง ๆ มาใช้งานในปัจจุบัน จำเป็นต้องศึกษาการใช้งาน อีกทั้งเครื่องมือเหล่านี้บางครั้งก็มีการทำงานที่ซับซ้อนและการรายงานผลการใช้งานค่อนข้างจะทำความเข้าใจได้ยาก เราจึงออกแบบชุดซอฟต์แวร์โดยมีชื่อว่า ProjectX มาเพื่อแก้ไขปัญหาเหล่านี้ โดยชุดซอฟต์แวร์นี้สามารถจัดการเครื่องมือทางด้านความปลอดภัยที่หลากหลาย ผู้ใช้งานสามารถปรับเปลี่ยนวิธีการประเมินความปลอดภัยโดยใช้ภาษาสคริปต์ที่สามารถปรับแต่งได้ง่าย หลังจากนั้นชุดซอฟต์แวร์นี้จะรวบรวมผลลัพธ์ที่ได้และทำรายงานสรุปในรูปแบบที่เข้าใจได้ง่าย ชุดซอฟต์แวร์นี้รองรับการพัฒนาส่วนต่อขยายเพื่อใช้เครื่องมือทางด้านความปลอดภัยอื่นนอกเหนือจากที่ใช้งานอยู่ในปัจจุบัน หรือสร้างรายงานในรูปแบบใหม่ได้ นอกจากนี้ผู้ดูแลระบบยังสามารถใช้งานผ่านอุปกรณ์มือถือที่มีความเป็นมิตรต่อผู้ใช้ได้อีกด้วย

AUTOMATIC COMPUTER SYSTEM SECURITY ASSESSMENT

Mr. Saran	Siriphantnon	54011340
Mr. Supol	Prachachon	54011407
Dr. Akkarit	Sangpetch	Advisor
Dr. Orathai	Sangpetch	Co-advisor

Academic Year 2014

ABSTRACT

Existing security assessment tools require specialized training and expertise to operate. In addition, the generated reports from these tools could be difficult to understand. Hence we design a unified security assessment framework, ProjectX to mitigate these problems. Our framework can manipulate various security assessment tools. The operators can customize the assessment methods to accommodate the target systems using easily configurable script. The framework then collects the results to generate unified and comprehensive reports. The framework could be extended to include additional assessment tools or generate different reporting formats. The administrators are also able to issue assessment operations and inspections using user-friendly mobile applications.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ได้รับคำแนะนำ และคำปรึกษาเกี่ยวกับการดำเนินการโครงงานนี้เป็นอย่างดีจากอาจารย์ที่ปรึกษาทั้งสองท่าน อาจารย์ ดร. อรทัย สังข์เพ็ชรและอาจารย์ ดร. อักฤทธิ์ สังข์เพ็ชร ซึ่งทางคณะผู้จัดทำรู้สึกซาบซึ้งเป็นอย่างมากในความอนุเคราะห์จากอาจารย์ทั้งสองที่คอยให้การสนับสนุนในการทำปริญญานิพนธ์นี้เสมอมา

และปริญญานิพนธ์ฉบับนี้จะสำเร็จลงไม่ได้หากไม่ได้รับความอนุเคราะห์จากบริษัท อินเตอร์เน็ตประเทศไทย จำกัด (มหาชน) หรือ INET ที่ได้ให้ความช่วยเหลือในเรื่องของทรัพยากรในการทำวิจัยต่าง ๆ

คณะผู้จัดทำขอขอบพระคุณอย่างสูง และหวังอย่างยิ่งว่าปริญญานิพนธ์ฉบับนี้จะเป็นประโยชน์ต่อทุกท่าน และสามารถให้คำแนะนำแก่นักศึกษารุ่นต่อไปในอนาคตได้

สรัล ศิริพันธ์โนน

สุพล ประชาชน

สารบัญ

สารบัญรูป

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

การนำเครื่องมือทางด้านการประเมินความปลอดภัยของระบบคอมพิวเตอร์ต่าง ๆ มาใช้งานในปัจจุบัน จำเป็นต้องศึกษาการใช้งาน อีกทั้งเครื่องมือเหล่านี้บางครั้งก็มีการทำงานที่ซับซ้อนและการรายงานผลการใช้งานค่อนข้างจะทำความเข้าใจได้ยากและใช้เวลา เราจึงออกแบบซอฟต์แวร์มาเพื่อแก้ไขปัญหาเหล่านี้ โดยซอฟต์แวร์นี้จะรวบรวมเครื่องมือต่าง ๆ ที่ใช้ในการประเมินความปลอดภัยของระบบคอมพิวเตอร์และนำผลลัพธ์ที่ได้กลับมารวบรวมและสรุปไว้ในรูปแบบที่เหมาะสมต่อการทำงานของผู้ใช้งานมากที่สุด

1.2 วัตถุประสงค์ของโครงงาน

เพื่อให้ผู้ใช้ได้รับรายงานความปลอดภัยของระบบที่สามารถเข้าใจได้ง่าย โดยออกแบบมาเพื่อให้ผู้ใช้ เช่น ผู้ดูแลระบบ ได้รับรายงานผลการประเมินด้านการรักษาความปลอดภัยเบื้องต้นอย่างสะดวก รวดเร็ว มีความต่อเนื่อง และเป็นอัตโนมัติ อย่างไรก็ตาม มิได้หมายความว่าระบบนี้สามารถทดแทนการประเมินจากผู้เชี่ยวชาญด้านความปลอดภัย การประเมินความปลอดภัยของระบบขั้นสูงยังคงต้องผ่านการดำเนินการด้วยผู้เชี่ยวชาญ

1.3 ขอบเขตของโครงงาน

1) ซอฟต์แวร์สามารถนำไปพัฒนาและเพิ่มเติมองค์ประกอบตามที่ต้องการได้โดยง่าย
2) ซอฟต์แวร์นี้สามารถให้รายงานผลการประเมินระบบคอมพิวเตอร์ในรูปแบบที่ผู้ใช้สามารถเข้าใจได้ง่าย
3) ซอฟต์แวร์นี้สามารถดำเนินการได้อย่างอัตโนมัติและสามารถกำหนดเวลาการประเมินล่วงหน้าได้
4) ซอฟต์แวร์นี้สามารถตั้งค่าการทำงานต่าง ๆ ได้ โดยการตั้งค่าต้องมีความยืดหยุ่นสูง สามารถกำหนดเครื่องเป้าหมายที่จะทำการประเมินความปลอดภัย สามารถกำหนดว่าจะใช้เครื่องมืออะไรในการประเมินนั้น และสามารถกำหนดได้ว่าจะใช้เครื่องมือเหล่านั้นอย่างไร
5) ซอฟต์แวร์นี้สามารถทำการประเมินระบบคอมพิวเตอร์โดยที่ผู้ใช้ไม่จำเป็นต้องศึกษา

รายละเอียดของเครื่องมือที่ใช้ในการประเมิน

1.4 วิธีการดำเนินการ

1) ออกแบบโครงสร้างการทำงานซอฟต์แวร์โดยรวม
2) พัฒนาระบบตามที่ได้ออกแบบไว้
3) ตรวจสอบการทำงานว่าเป็นไปตามที่ระบุไว้ในวัตถุประสงค์ของซอฟต์แวร์หรือไม่
4) ดำเนินการตามขั้นตอนที่ 1) – 3) จนกระทั่งซอฟต์แวร์เสร็จสมบูรณ์เรียบร้อย

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ผู้ใช้ได้รับรายงานความปลอดภัยของระบบอย่างต่อเนื่องและเป็นอัตโนมัติ
2) ผู้ใช้สามารถนำรายงานที่ได้รับไปวิเคราะห์ถึงการแก้ไขปัญหา หรือส่งให้ผู้เชี่ยวชาญต่อไปได้
3) ระบบสามารถที่จะเพิ่มความสามารถใหม่ เพื่อรองรับความต้องการที่เพิ่มขึ้นในอนาคต

1.6 ส่วนประกอบของปริญญานิพนธ์

บทที่ 1 บทนำ กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์ฉบับนี้

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีพื้นฐาน หลักการ ความรู้ด้านการประเมินความปลอดภัยระบบคอมพิวเตอร์ และการพัฒนาระบบของโครงงานนี้

บทที่ 3 การออกแบบและพัฒนา กล่าวถึงรายละเอียดออกแบบและการพัฒนาโครงงาน รวมถึงการทำงานต่าง ๆ ในระบบของโครงงานนี้

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึงการเตรียมการทดลองทั้งการจัดเตรียมส่วนฮาร์ดแวร์ ส่วนซอฟต์แวร์ สภาวะแวดล้อมในการทำการทดลอง ผลการทดลองและการวิเคราะห์ผลการทดลองของระบบในโครงงานนี้

บทที่ 5 บทสรุป กล่าวถึงบทสรุปของโครงงาน ข้อจำกัดและปัญหาอุปสรรคต่าง ๆ ของโครงงาน และข้อเสนอแนะสำหรับแนวทางในการพัฒนาโครงงานนี้ต่อไป

ภาคผนวก กล่าวถึงองค์ประกอบอ้างอิงต่าง ๆ ของซอฟต์แวร์ นอกเหนือไปจากการทำงานหลักของซอฟต์แวร์

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 หลักการสำรวจระบบ

2.1.1 Host Discovery

เป็นการดำเนินการเพื่อตรวจสอบว่ามีเครื่องปลายทางที่ใช้ IP ที่ต้องการอยู่จริง โดยทั่วไปสามารถดำเนินการด้วยวิธีที่แตกต่างกันได้ เช่น ICMP Scan, ARP Scan, TCP หรือ UDP Ping โดยมีละเอียดดังนี้

2.1.1.1 ICMP Scan

เป็นการส่งคำร้องขอ ICMP เช่น ICMP Echo เพื่อดูว่า IP ปลายทางมีการตอบกลับมาหรือไม่ ซึ่งอาจทำให้ระบุได้ว่ามีเครื่องปลายทางที่มีการใช้งาน IP นั้นอยู่

2.1.1.2 ARP Scan

เป็นการส่งคำร้อง ARP (Address Resolution Protocol) เพื่อดูว่ามีเครื่องใดในเครือข่ายตอบกลับต่อคำร้องสำหรับ IP นั้นหรือไม่ ซึ่งอาจทำให้ระบุได้ว่ามีเครื่องปลายทางที่มีการใช้งาน IP นั้นอยู่

2.1.1.3 TCP SYN/ACK Ping, UDP Ping

เป็นการส่งคำร้อง TCP หรือ UDP ไปยังหมายเลขพอร์ตที่ระบุของเครื่องเป้าหมายเพื่อดูว่ามีการตอบกลับต่อคำร้องนั้นหรือไม่ ซึ่งอาจทำให้ระบุได้ว่ามีเครื่องปลายทางที่มีการใช้งาน IP นั้นอยู่

2.1.2 Port Scanning

เป็นการส่งคำร้อง TCP หรือ UDP ไปยังพอร์ตต่าง ๆ ของเครื่องเป้าหมายปลายทางเพื่อดูว่ามีการเปิดบริการที่พอร์ตใดบ้าง นอกจากนี้ยังพยายามตรวจสอบว่าบริการที่เปิดอยู่ที่พอร์ตนั้นคือบริการใดโดยเทคนิคที่เรียกว่าการตรวจพิสูจน์บริการ (Fingerprinting)

2.1.2.1 การตรวจพิสูจน์บริการ (Fingerprinting)

การตอบกลับคำร้อง TCP หรือ UDP สำหรับบริการต่าง ๆ ที่อยู่บนพอร์ตใด ๆ ของเครื่องเป้าหมายนั้นมีความแตกต่างกันไปในแต่ละบริการ ผู้ดำเนินการสำรวจสามารถนำฐานข้อมูลที่เก็บรวบรวมรูปแบบการตอบกลับของบริการต่าง ๆ มาค้นหาดูได้ว่าการตอบกลับจากพอร์ตใด ๆ มีความเป็นไปได้ว่าจะเป็นบริการใด (เช่น HTTP หรือ SMTP เป็นต้น) นอกจากนี้เทคนิคนี้ยังอาจสามารถระบุซอฟต์แวร์และเวอร์ชันของซอฟต์แวร์ที่ให้บริการดังกล่าวโดยดูจากการตอบกลับนี้ได้อีกด้วย

ในปัจจุบันมีซอฟต์แวร์สำเร็จรูปในการสำรวจระบบมากมายโดยใช้หลักการดังกล่าวข้างต้น ซึ่งอาจมีเทคนิคที่แตกต่างกันไปตามแต่ละซอฟต์แวร์นั้น เช่น การหลบหลีกไฟร์วอลล์ การตั้งค่าการทำงานแบบขนาน เป็นต้น ซึ่งจะได้นำมาใช้ในโครงงานนี้ต่อไป

2.2 หลักการประเมินความปลอดภัยระบบ

ในการประเมินความปลอดภัยของระบบสามารถดำเนินการได้หลายวิธีด้วยกัน โดยวิธีทั่วไปที่นิยมใช้สามารถถูกแยกตามลักษณะของระบบที่จะทำการประเมินได้ดังนี้

2.2.1 ความปลอดภัยของรหัสผ่าน

การประเมินความปลอดภัยของรหัสผ่านมักใช้วิธีการที่เรียกว่า Brute-force ซึ่งเป็นการใส่รหัสผ่านเข้าไปหลายรหัสผ่านจนกว่าระบบจะตอบสนองว่ารหัสผ่านนั้นถูกต้อง โดยอาจโจมตีในรูปแบบอิงตามคำในพจนานุกรมหรือการปรับเปลี่ยนคำเป็นตัวเลขซึ่งผู้ใช้ทั่วไปมักใช้กัน เช่น ปรับจาก password เป็น p455w0rd เป็นต้น

ในปัจจุบันมีซอฟต์แวร์ที่ใช้วิธี Brute-force มากมาย เช่น THC Hydra, Medusa ซึ่งมีเทคนิคอื่น ๆ เช่น การหลบหลีกการตรวจจับ โดยการสุ่มระยะเวลาในการดำเนินการ หรือการใช้เหยื่อล่อ (Decoy) ซึ่งเป็นการปลอมแปลงแหล่งที่มาการโจมตีเป็นเครื่องอื่น ๆ ในอินเตอร์เน็ต และเทคนิคการทำงานแบบขนาน ซึ่งจะทำให้การโจมตีทำได้รวดเร็วยิ่งขึ้น เป็นต้น โดยเครื่องมือประเภทนี้จะได้นำมาใช้ในโครงงานนี้ต่อไป

2.2.2 ความปลอดภัยของเว็บไซต์

เว็บไซต์สามารถที่จะรับข้อมูลเข้าจากผู้ใช้และนำไปประมวลผลได้หลายช่องทาง เช่น HTTP Header, HTTP Body, Query String เป็นต้น ซึ่งหากบริการเว็บไซต์มีการตั้งค่าหรือถูกพัฒนาไม่ดี

หรือมีช่องโหว่ในตัวซอฟต์แวร์ที่ใช้เองอาจทำให้ผู้โจมตีสามารถใช้ประโยชน์จากการประมวลผลข้อมูลที่ผิดพลาดเหล่านั้นได้ โดยในปัจจุบันพบช่องโหว่จากข้อมูลเข้าได้หลายทาง เช่น SQL Injection ซึ่งเป็นการโจมตีโดยใช้ช่องโหว่ของแอปพลิเคชันที่อ่านค่าที่ผู้ใช้รับเข้ามาแล้วทำการประมวลผลเป็นคำสั่ง SQL โดยไม่ได้คัดกรองค่าเหล่านั้นก่อน, File Inclusion ซึ่งเป็นการอ่านค่าไฟล์อื่น ๆ ในเครื่องเป้าหมายนำมาแสดงที่หน้าเว็บไซต์โดยมิได้ตั้งใจ, หรือ Cross-Site Scripting ซึ่งเป็นการรับค่าที่ผู้ใช้ป้อนเข้ามาแล้วนำไปแสดงบนหน้าเว็บไซต์โดยไม่ได้คัดกรองค่า เป็นผลให้ผู้โจมตีสามารถรันคำสั่ง JavaScript ใด ๆ ได้ เป็นต้น โดยการช่องโหว่เหล่านี้มีความรุนแรงแตกต่างกันไป เช่น การได้รับข้อมูลส่วนตัวของผู้ใช้ การได้รับสิทธิ์ของผู้ดูแลระบบ เป็นต้น

ในปัจจุบันมีซอฟต์แวร์มากมายที่ทำให้การประเมินความปลอดภัยเว็บไซต์สามารถทำการประเมินและได้รับผลลัพธ์การประเมินได้ง่ายขึ้น ตัวอย่างเช่น W3AF, Nitko โดยจะได้นำซอฟต์แวร์เหล่านี้มาใช้ในโครงงานต่อไป

2.2.3 ความปลอดภัยของซอฟต์แวร์บริการและระบบปฏิบัติการในระบบเครือข่าย

ซอฟต์แวร์ที่ใช้ในการบริการในระบบเครือข่ายต่าง ๆ เช่น SMTP, DNS และระบบปฏิบัติการต่าง ๆ อาจจะมีช่องโหว่เนื่องจากการที่ตัวซอฟต์แวร์อาจมีความซับซ้อน ซึ่งอาจทำให้การพัฒนาเกิดความผิดพลาดได้ง่าย ตัวอย่างช่องโหว่ดังกล่าว เช่น Buffer Overflow ซึ่งเกิดจากการรับค่าจากผู้ใช้แล้วนำไปใช้งานโดยไม่ตรวจสอบขอบเขตของตัวแปรก่อน, หรือการตรวจสอบเงื่อนไขหรือขั้นตอนดำเนินการที่ผิดพลาด เป็นต้น ซึ่งอาจทำให้ผู้โจมตีสามารถได้ข้อมูลหรือสิทธิ์การเข้าถึงที่สำคัญในระบบได้ โดยทั่วไปอาจทดสอบได้โดยการใช้ซอฟต์แวร์ด้านการประเมินความปลอดภัยทำการป้อนค่าที่อาจทำให้แอปพลิเคชันที่มีช่องโหว่ทำการแสดงช่องโหว่ดังกล่าวออกมาได้ เช่น การป้อนค่าคำสั่งซึ่งมีรูปแบบเฉพาะเข้าไปในช่องรับข้อมูล ซึ่งอาจทำให้สามารถรันคำสั่งใด ๆ ที่ต้องการ ได้

ในปัจจุบันมีซอฟต์แวร์มากมายที่ได้ทำให้การประเมินความปลอดภัยบนเครือข่ายนี้สามารถทำการประเมินและเห็นผลลัพธ์การประเมินได้ง่ายขึ้น โดยจะได้นำเครื่องมือเหล่านี้มาใช้ในโครงงานนี้ต่อไป

2.3 หลักการประมวลผลแบบกระจาย

การกระจายการประมวลผลทำให้สามารถแบ่งเบาภาระของการประมวลผลต่าง ๆ ได้โดยใช้หน่วยประมวลผลที่มากขึ้นและยังสามารถทำให้ดำเนินการแบบขนานได้อีกด้วย โดยในปัจจุบันมีการศึกษาและใช้งานการประมวลผลแบบกระจายที่หลากหลาย ซึ่งเทคนิคที่จะใช้ในโครงงานนี้มีดังนี้

2.3.1 ฐานข้อมูลกลาง

เนื่องจากการดำเนินการต่าง ๆ ในโครงงานนี้ไม่ต้องการการประมวลผลฐานข้อมูลที่สูงและบ่อย จึงได้ออกแบบโหนดการประมวลผลต่าง ๆ ให้ดำเนินการบนฐานข้อมูลขุดเดียว แต่ทั้งนี้อย่างไรก็ตามสามารถเลือกที่จะสำรองข้อมูลหรือขยายฐานข้อมูลได้โดยใช้เทคนิคต่าง ๆ เช่น การจัดกลุ่มร่วมประมวลผล (Clustering) ซึ่งโหนดต่าง ๆ ที่เรียกใช้งานฐานข้อมูลไม่จำเป็นต้องรู้ว่าฐานข้อมูลมีมากกว่า 1 ตัวทำการร่วมกันประมวลผลอยู่ เป็นต้น

2.3.2 การจัดการความผิดพลาด

เราจำแนกข้อผิดพลาดในระบบออกเป็น 2 แบบ ดังนี้

1) Error เป็นข้อผิดพลาดที่สามารถทำซ้ำการดำเนินการเดิมที่ทำให้เกิดข้อผิดพลาดได้ โดยจะทำซ้ำตามไปเรื่อย ๆ จนกว่าจะไม่เกิดข้อผิดพลาด หรือจนกว่าจะทำซ้ำมากกว่าจำนวนการทำซ้ำสูงสุดที่กำหนด
2) Fatal เป็นข้อผิดพลาดที่ไม่สามารถทำซ้ำการการดำเนินการเดิมที่ทำให้เกิดข้อผิดพลาดได้ โดยระบบจะไม่ทำซ้ำและข้อผิดพลาดจะถูกบันทึกเก็บไว้พร้อมกับการหยุดการดำเนินการทั้งหมดของช่วงการดำเนินการ (Session) นั้น

ทั้งนี้ การทำซ้ำการดำเนินการที่เป็นข้อผิดพลาด จำเป็นต้องทำให้การดำเนินการมีลักษณะเป็น *Idempotent* ซึ่งหมายถึงการทำซ้ำการดำเนินการในแต่ละครั้งจำเป็นต้องได้ผลลัพธ์ที่เหมือนกัน

2.3.3 การประมวลผลแบบขนาน

โดยทั่วไปสามารถใช้การเพิ่ม Thread หรือ Process สำหรับการทำงานได้โดยให้ระบบปฏิบัติการเป็นผู้ดูแล นอกจากนี้หากเพิ่มเครื่องหรือโหนดที่ใช้ในการทำงานจะสามารถแบ่งเบาภาระการทำงานต่าง ๆ ได้ โดยโครงงานนี้ได้เลือกใช้รูปแบบที่มีผู้ควบคุมกลางในการควบคุมการแบ่งงานและการประสานงานของโหนดต่าง ๆ

บทที่ 3
การออกแบบและการพัฒนา

3.1 ภาพรวม

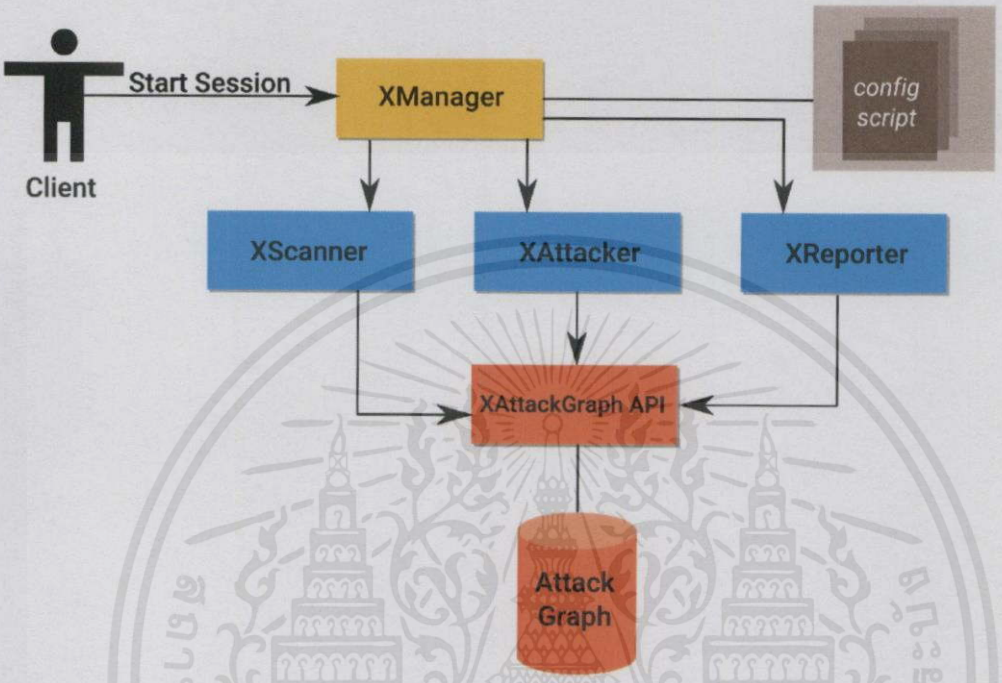
รูปที่ 1 ภาพรวมของโครงงาน

จากรูปข้างต้นจะพบว่าการทำงานต่าง ๆ ในโครงงานนี้แบ่งออกเป็น 3 ส่วน นั่นคือ ส่วนของผู้ใช้งาน ซึ่งสามารถใช้แอปพลิเคชันบนอุปกรณ์พกพา หรือโปรแกรมบรรทัดคำสั่งที่เราได้พัฒนาขึ้นมาทำการติดต่อไปยังส่วนที่สอง นั่นคือ ส่วนซอฟต์แวร์สำหรับการประเมินความปลอดภัยของระบบคอมพิวเตอร์แบบอัตโนมัติ (Automatic Computer System Security Assessment Software) ซึ่งเป็นส่วนที่เราได้พัฒนาขึ้นมาเช่นกัน โดยในที่นี้จะเรียกว่า ProjectX ซึ่ง ProjectX จะดำเนินการประเมินความปลอดภัยไปยังส่วนสุดท้าย นั่นคือ ส่วนระบบคอมพิวเตอร์เป้าหมายที่จะดำเนินการประเมินความปลอดภัย (Target System) โดยเป็นกลุ่มของคอมพิวเตอร์ (อาจเป็นเครื่องกายภาพหรือเครื่องเสมือน) ที่อาจมีเครื่องคอมพิวเตอร์บางส่วนหรือทั้งหมดเชื่อมต่อกันในลักษณะของเครือข่ายคอมพิวเตอร์ หรืออาจไม่เชื่อมต่อกันเลยก็ได้ ซึ่งเมื่อ ProjectX ดำเนินการประเมินความปลอดภัยเสร็จสิ้น จะส่งผลลัพธ์การประเมินความปลอดภัยไปยังส่วนของผู้ใช้งาน โดยผลลัพธ์การประเมินความปลอดภัยจะประกอบด้วยข้อมูลทั่วไปของระบบคอมพิวเตอร์เป้าหมาย บริการต่าง ๆ

ของคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมายที่สามารถเข้าถึงได้ และช่องโหว่ที่พบในแต่ละบริการของคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมายนั้น

โดยต่อไปในเอกสารนี้ จะเรียกผู้ใช้งานในส่วนที่หนึ่งว่า *ผู้ใช้งาน* และจะเรียกซอฟต์แวร์ในส่วนที่สองว่า *ProjectX* และสุดท้ายจะเรียกระบบคอมพิวเตอร์เป้าหมายในส่วนที่สามว่า *ระบบคอมพิวเตอร์เป้าหมาย*

3.2 โครงสร้างซอฟต์แวร์



รูปที่ 2 โครงสร้างโดยรวมของซอฟต์แวร์

โครงสร้างซอฟต์แวร์ประกอบด้วยส่วนประกอบซอฟต์แวร์ย่อยต่าง ๆ ดังนี้

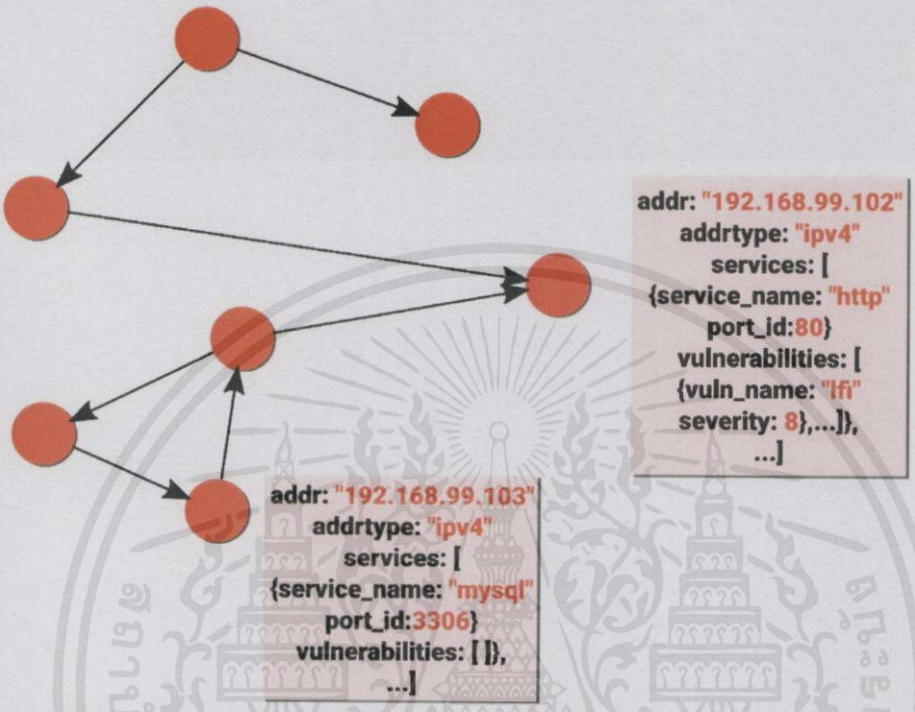
1) XManager ดูแลการติดต่อกับ Client และประสานงานส่วนประกอบย่อยอื่น ๆ ของซอฟต์แวร์
2) XScanner ดูแลส่วนการสำรวจระบบของเป้าหมาย
3) XAttacker ดูแลส่วนการโจมตีและประเมินความปลอดภัยของระบบเป้าหมาย
4) XReporter ดูแลการออกรายงานผลการประเมินความปลอดภัยของระบบ
5) XAttackGraph API ดูแลการสร้าง Attack Graph (จะได้พูดถึงต่อไป) ของระบบเป้าหมาย

โดยแต่ละส่วนประกอบซอฟต์แวร์ย่อยจะมีส่วนติดต่อในลักษณะ REST API โดยเป็นหลักการออกแบบการติดต่อส่วนการทำงานต่าง ๆ ในระบบด้วยโปรโตคอล HTTP ซึ่งมีหลักการออกแบบเน้นไปที่การเข้าถึงทรัพยากรต่าง ๆ ในระบบ และการดำเนินการต่อทรัพยากรเหล่านั้น โดยทรัพยากรใน ProjectX นี้หมายถึงคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมาย บริการของคอมพิวเตอร์แต่ละเครื่องในระบบนั้น และช่องโหว่ของแต่ละบริการของคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมาย โดย

สามารถที่จะวางส่วนประกอบซอฟต์แวร์ย่อยอยู่ในเครื่องเดียวกันหรือกระจายหลายเครื่องได้ ซึ่งรายละเอียดแต่ละส่วนของส่วนประกอบซอฟต์แวร์ย่อยจะได้กล่าวต่อไป

นอกจากนี้การออกแบบได้ใช้สถาปัตยกรรมในลักษณะของ Microservices [1] ซึ่งเป็นการออกแบบระบบให้มีส่วนย่อยต่าง ๆ ที่สามารถติดต่อสื่อสารกันได้ผ่านช่องทาง เช่น ผ่านโปรโตคอล HTTP หรือผ่านการใช้งานแอปพลิเคชันประเภท Message Queue โดยในแต่ละส่วนสามารถทำการขยายหรืออัพเกรดเพื่อรองรับการทำงานที่อาจเพิ่มขึ้นได้โดยไม่กระทบส่วนอื่น ๆ ของระบบ

3.3 Attack Graph



รูปที่ 3 ตัวอย่างรูปแบบ Attack Graph

การเก็บข้อมูลต่าง ๆ ของระบบเป้าหมาย จะใช้รูปแบบการเก็บในลักษณะกราฟ โดยจะเรียกกราฟดังกล่าวว่า Attack Graph (มีแนวคิดดั้งเดิมมาจากผลงานของ Jeannette M. Wing [2]) ซึ่งจะแทนโหนด (Node) ของกราฟต่าง ๆ ด้วยข้อมูลของเครื่องเป้าหมายแต่ละเครื่องโดยอาจเป็นเครื่องจริง ๆ หรืออาจเป็น Virtual Machine ก็ได้ ข้อมูลในแต่ละ โหนดประกอบด้วย

1) หมายเลข IP Address และชนิดของ IP Address ของเครื่องเป้าหมาย
2) บริการต่าง ๆ ที่เครื่องเป้าหมายได้เปิดบริการไว้
3) ช่องโหว่ของบริการต่าง ๆ

โดยเส้นเชื่อม (Edge) แสดงถึงความสัมพันธ์ของ Node ต่าง ๆ ซึ่งอาจมีทิศทางไปทางใดทางหนึ่ง เช่น จาก Client ไป Server หรือไปทั้งสองทิศทาง เช่น การสื่อสารแบบ Peer to Peer โดยข้อมูลตรงนี้สามารถใช้ระบุทิศทางการโจมตีที่อาจเกิดขึ้นในระบบ หรือตรวจสอบข้อมูลที่สามารถเดินทางไปยังเครื่องคอมพิวเตอร์บนเครือข่ายได้

3.4 ขั้นตอนของระบบ

รูปที่ 4 ขั้นตอนการทำงานของระบบ

การทำงานโดยรวมของซอฟต์แวร์จะแบ่งออกเป็น 5 ขั้นตอน ดังนี้

3.4.1 ขั้นตอนที่ *1* การตั้งค่า (Setup)

การทำงานเริ่มต้นจากการตั้งค่าต่าง ๆ ของซอฟต์แวร์ เช่น การตั้งค่าสคริปต์ที่จะใช้ (จะได้กล่าวถึงในหัวข้อ *การปรับแต่งการทำงานของซอฟต์แวร์*) การตั้งค่าระบบที่จะทำการประเมิน เป็นต้น การทำงานในส่วนนี้รับผิดชอบโดยส่วนประกอบซอฟต์แวร์ย่อยที่มีชื่อว่า XManager

3.4.2 ขั้นตอนที่ 2 การแสกน (Scanning)

เป็นขั้นตอนการสร้าง Attack Graph โดยการใช้เครื่องมือด้านการสำรวจต่าง ๆ เช่น Nmap, Skipfish และอื่น ๆ ในการสำรวจโครงสร้างระบบแล้วทำการประมวลผลและจัดเก็บผลลัพธ์การดำเนินการลง Attack Graph โดยส่วนประกอบซอฟต์แวร์ย่อยที่มีชื่อว่า XScanner

3.4.3 ขั้นตอนที่ 3 การโจมตี (Attacking)

เป็นขั้นตอนการสร้าง Attack Graph โดยการใช้เครื่องมือด้านการโจมตีต่าง ๆ เช่น THC Hydra, W3AF, Nikto และอื่น ๆ ในการทำการโจมตีแล้วทำการประมวลผลและจัดเก็บผลลัพธ์การโจมตีลง Attack Graph รับผิดชอบโดยส่วนประกอบซอฟต์แวร์ย่อยที่มีชื่อว่า **XAttacker**

3.4.4 ขั้นตอนที่ 4 การรายงาน (Reporting)

เป็นขั้นตอนสร้างรายงานต่าง ๆ เช่น JSON, XML, PDF และอื่น ๆ จาก Attack Graph ที่ถูกสร้างโดยขั้นตอน Setup, Scanning และ Attacking ซึ่งจะถูกนำไปใช้งานต่อไป รับผิดชอบโดยส่วนประกอบซอฟต์แวร์ย่อยที่มีชื่อว่า **XReporter**

3.4.5 ขั้นตอนที่ 5 การจบการทำงาน (Teardown)

เป็นขั้นตอนสุดท้ายซึ่งรับผิดชอบโดยส่วนประกอบซอฟต์แวร์ย่อย **XManager** ซึ่งสามารถตั้งค่าให้ทำงานต่าง ๆ เช่น การส่ง Email แจ้งเตือนหลังเสร็จสิ้นการประเมินความปลอดภัยของระบบเป้าหมายประกอบซอฟต์แวร์ย่อย

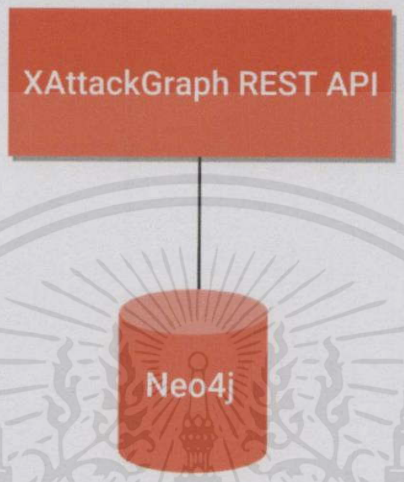
ขั้นตอนการดำเนินการและการเลือกใช้ซอฟต์แวร์ด้านการประเมินความปลอดภัยต่าง ๆ มีแนวคิดพื้นฐานมาจากหนังสือ *Counter Hack: Reloaded* [3] ซึ่งมีขั้นตอนดังนี้

1. Reconnaissance หรือ การเก็บเกี่ยวข้อมูลเกี่ยวกับระบบคอมพิวเตอร์เป้าหมาย
2. Scanning หรือ การตรวจสอบการให้บริการของคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมาย
3. Attacking หรือ การโจมตีบริการของคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมาย โดยการโจมตีอาจทำให้ได้มาซึ่งการเข้าถึงคอมพิวเตอร์บางเครื่องในระบบนี้
4. Maintaining Access หรือ การคงไว้ซึ่งการเข้าถึงของคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมายที่สามารถเข้าถึงได้
5. Covering Attack การปกปิดร่องรอยการโจมตีและการเข้าถึงเครื่องเป้าหมาย

โดยเนื่องจากเราควรรู้รายละเอียดที่จำเป็นล่วงหน้า เช่น หมายเลขไอพีแอดเดรสของคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมายที่จะทำการประเมิน จึงทดแทนด้วยขั้นตอนการกำหนดค่า (Setup) แทน โดยสามารถกำหนดได้ว่าจะให้มีการประเมินความปลอดภัยอย่างไรบ้างต่อคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมาย และจากการที่การประเมินความปลอดภัย ไม่ได้มีเป้าหมายเพื่อการยึดครองเครื่องคอมพิวเตอร์ในระบบคอมพิวเตอร์เป้าหมาย จึงได้ตัดขั้นตอนการดำเนินการที่ 4 และ 5 (Maintaining Access และ Covering Attack) ออกไป และใช้การรายงานและการจบการทำงานซึ่งจำเป็นต่อการประเมินความปลอดภัยแบบอัตโนมัติเข้ามาแทน

3.5 โครงสร้างส่วนประกอบซอฟต์แวร์ย่อย

3.5.1 XAttackGraph API



รูปที่ 5 การติดต่อ REST API และฐานข้อมูล Neo4j

XAttackGraph API เป็นส่วนติดต่อสำหรับการสร้างและจัดการ Attack Graph โดยการทำงานภายในได้จัดเก็บข้อมูลด้วยฐานข้อมูล Neo4j [4] ซึ่งส่วนประกอบย่อยอื่น ๆ ในระบบสามารถเรียกใช้การสร้างและจัดการ Attack Graph ได้โดยใช้ API ดังนี้

รายละเอียดเพิ่มเติมของ API นี้สามารถดูได้ที่ *ภาคผนวก ก.*

3.5.1.1 การพัฒนา

ในส่วน REST API ได้ใช้ Sinatra ของภาษา Ruby ในการพัฒนาและได้ใช้ Library ของภาษา Ruby ชื่อว่า Neography [5] ในการติดต่อกับฐานข้อมูล Neo4j

3.5.1.2 รูปแบบการจัดเก็บ Attack Graph (Attack Graph Database Schema)

จัดเก็บลงในฐานข้อมูล Neo4j โดยมีรูปแบบการเก็บข้อมูล ดังนี้

(A) -[has_service]-> (S) -[has_vulnerability]-> (V)

โดย (A) หมายถึง โหนดแสดงเครื่องเป้าหมาย

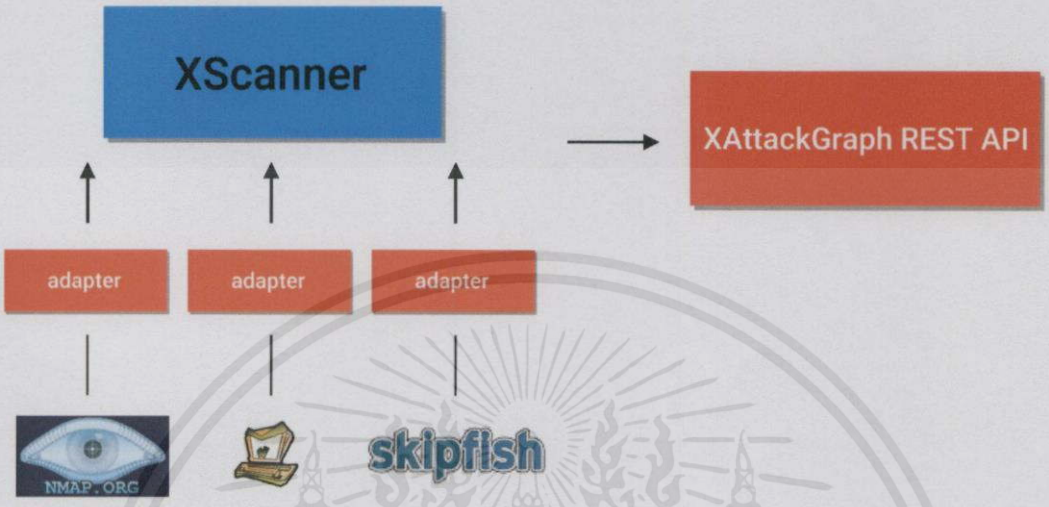
(S) หมายถึง โหนดแสดงบริการต่าง ๆ ของเครื่องเป้าหมาย

(V) หมายถึง โหนดแสดงช่องโหว่ต่าง ๆ ของบริการของเครื่องเป้าหมาย

-[has_service]-> แสดงความสัมพันธ์ระหว่างโหนดเป้าหมายและโหนดบริการ โดยหมายถึง โหนดด้านซ้ายซึ่งเป็นโหนดแสดงเครื่องเป้าหมายมีข้อมูลบริการจากโหนดด้านขวาซึ่งเป็นโหนดแสดงบริการต่าง ๆ ของเครื่องเป้าหมาย

-[has_vulnerability]-> แสดงความสัมพันธ์ระหว่างโหนดบริการและโหนดช่องโหว่ โดยหมายถึง โหนดด้านซ้ายซึ่งเป็นโหนดแสดงบริการต่าง ๆ ของเครื่องเป้าหมายมีข้อมูลช่องโหว่จากโหนดด้านขวาซึ่งเป็นโหนดแสดงช่องโหว่ต่าง ๆ ของบริการของเครื่องเป้าหมาย

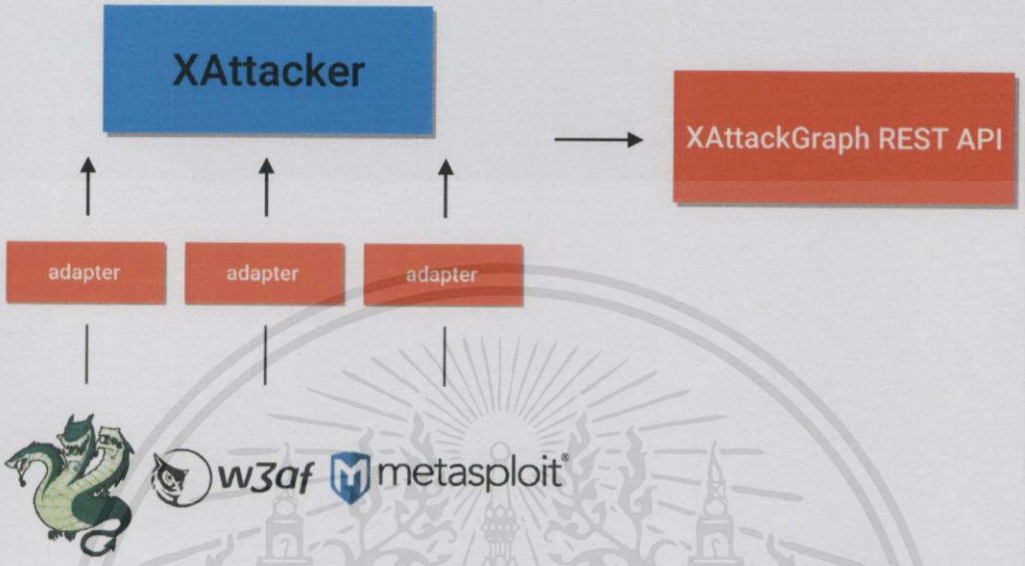
3.5.2 XScanner



รูปที่ 6 ภาพรวมการทำงานของ XScanner

XScanner ดูแลการสำรวจระบบเป้าหมายและแปลงข้อมูลที่ได้มาให้อยู่ในรูปของ Attack Graph ผ่านการเรียกใช้ XAttackGraph API โดยการทำงานจะเรียกโปรแกรมภายนอก เช่น Nmap, Skipfish เป็นต้น สำหรับการดำเนินการสำรวจระบบเป้าหมาย หลังจากนั้น XScanner จะอ่านรายงานการสำรวจของโปรแกรมนั้นซึ่งอาจออกมาในรูปแบบต่าง ๆ เช่น HTML, XML, JSON เป็นต้น นอกจากนี้ XScanner ยังมีหน้าที่แปลความหมายของรายงานการสำรวจให้อยู่ในรูปแบบที่สามารถเก็บในลักษณะ Attack Graph ได้ เช่น การแปลงบริการที่สำรวจได้โดยเครื่องมือ Nmap ซึ่งให้ผลลัพธ์ออกมาในลักษณะ XML ไฟล์ โดยให้ข้อมูลเกี่ยวกับบริการ พอร์ตและซอฟต์แวร์ที่ใช้สำหรับบริการนั้น จัดเก็บลงในฐานข้อมูล Attack Graph โดยมีรูปแบบการจัดเก็บตามหัวข้อ *3.5.1.2 รูปแบบการจัดเก็บ Attack Graph (Attack Graph Database Schema)*

3.5.3 XAttacker

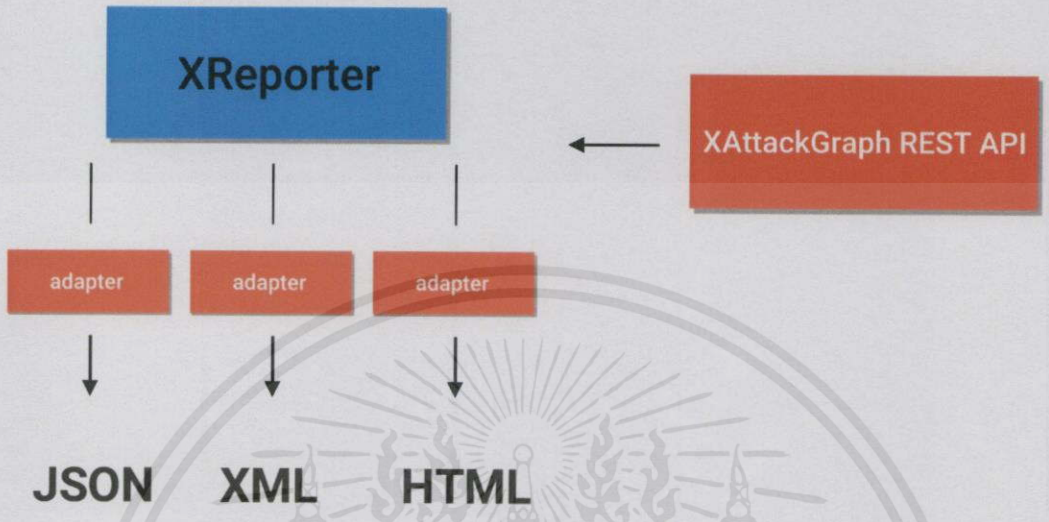


รูปที่ 7 ภาพรวมการทำงานของ XAttacker

XAttacker ดูแลการโจมตีและประเมินความปลอดภัยระบบเป้าหมายและแปลงข้อมูลที่ได้มาให้อยู่ในรูปของ Attack Graph ผ่านการเรียกใช้ XAttackGraph API โดยการทำงานจะเรียกซอฟต์แวร์การประเมินความปลอดภัย เช่น THC Hydra, W3AF เป็นต้น สำหรับการดำเนินการโจมตีและประเมินความปลอดภัยระบบเป้าหมาย หลังจากนั้น XAttacker จะอ่านรายงานการประเมินความปลอดภัยของซอฟต์แวร์นั้นซึ่งอาจออกมาในรูปแบบต่าง ๆ แล้วทำการแปลความหมายและจัดเก็บลงในฐานข้อมูล Attack Graph เช่น แปลงข้อมูลช่องโหว่ของบริการที่ทำการประเมินความปลอดภัยได้โดยเครื่องมือ W3AF ซึ่งให้ผลลัพธ์ออกมาในลักษณะ XML ไฟล์ โดยให้ข้อมูลเกี่ยวกับช่องโหว่ของบริการ HTTP/HTTPS, ช่องโหว่ของเว็บไซต์ ความรุนแรงและมาตรฐานที่เกี่ยวข้องกับช่องโหว่เหล่านั้น จัดเก็บลงในฐานข้อมูล Attack Graph โดยมีรูปแบบการจัดเก็บตามหัวข้อ *3.5.1.2 รูปแบบการจัดเก็บ Attack Graph (Attack Graph Database Schema)*

การทำงานของ XAttacker จำเป็นต้องได้รับข้อมูลที่ต้องการจาก XScanner ก่อนเสมอ เช่น บริการอะไรบ้างที่เปิดอยู่บนเครื่องเป้าหมายแต่ละเครื่อง เป็นต้น

3.5.4 XReporter



รูปที่ 8 ภาพรวมการทำงานของ XReporter

XReporter ดูแลการออกรายงานการประเมินความปลอดภัยของระบบเป้าหมายโดยการอ่านข้อมูลต่าง ๆ จาก Attack Graph ที่ดำเนินการมาก่อนหน้านี้เพื่อสร้างรายงานสำหรับการนำไปใช้งานต่อไป เช่น การออกรายงาน JSON เพื่อใช้แสดงใน Mobile Client (ดูหัวข้อ *3.9 Mobile Client*) หรือ HTML สำหรับรูปแบบที่เหมาะสมต่อการอ่านโดยผู้ใช้หรือผู้ดูแลระบบ

3.5.5 XManager

รูปที่ 9 ภาพรวมการทำงานของ XManager

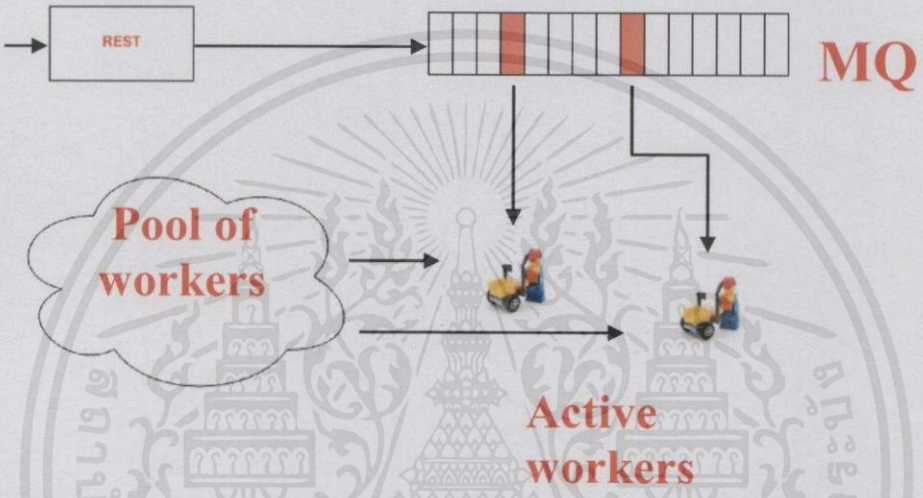
XManager ดูแลการประสานงานของส่วนประกอบซอฟต์แวร์ย่อยอื่น ๆ โดยอ่านการตั้งค่าต่าง ๆ จากสคริปต์การตั้งค่า ในหัวข้อ 3.7 *การปรับแต่งการทำงานของซอฟต์แวร์* และหากมีการเพิ่มส่วนประกอบซอฟต์แวร์ย่อยอื่น ๆ เข้ามาในระบบ (ในหัวข้อ 3.8 *การขยายต่อของซอฟต์แวร์*) ก็สามารถที่จะแก้ไขการทำงานของ XManager ให้รองรับหรือเพิ่มการตั้งค่าใหม่ ๆ ในสคริปต์ต่าง ๆ เพื่อให้ XManager อ่านและทำงานได้อย่างถูกต้องได้ หน้าที่โดยสรุปของ XManager มีดังนี้

1) การประสานงานในขั้นตอนต่าง ๆ ให้เป็นไปตามลำดับตามหัวข้อ 3.4 *ขั้นตอนของระบบ*
2) อ่านสคริปต์การตั้งค่าและดำเนินการต่าง ๆ ตามการตั้งค่านั้น
3) เก็บบันทึกความผิดปกติต่าง ๆ ของซอฟต์แวร์ที่เกิดขึ้นระหว่างการดำเนินการ และจัดการความผิดปกติเหล่านั้นตามความเหมาะสม เช่น เมื่อเกิดข้อผิดพลาดทางด้านเครือข่าย และคอมพิวเตอร์ที่ใช้ดำเนินการโดย ProjectX มีปัญหา ต้องสามารถกลับดำเนินการอีกครั้งได้เมื่อเครือข่ายหรือเครื่องคอมพิวเตอร์นั้นอยู่ในสภาวะปกติแล้ว

3.5.6 การพัฒนา

ส่วนประกอบย่อยแต่ละส่วนถูกพัฒนาด้วยหลักการเดียวกันซึ่งเรียกในโครงงานนี้ว่า XService โดยมีข้อกำหนดการทำงานดังนี้

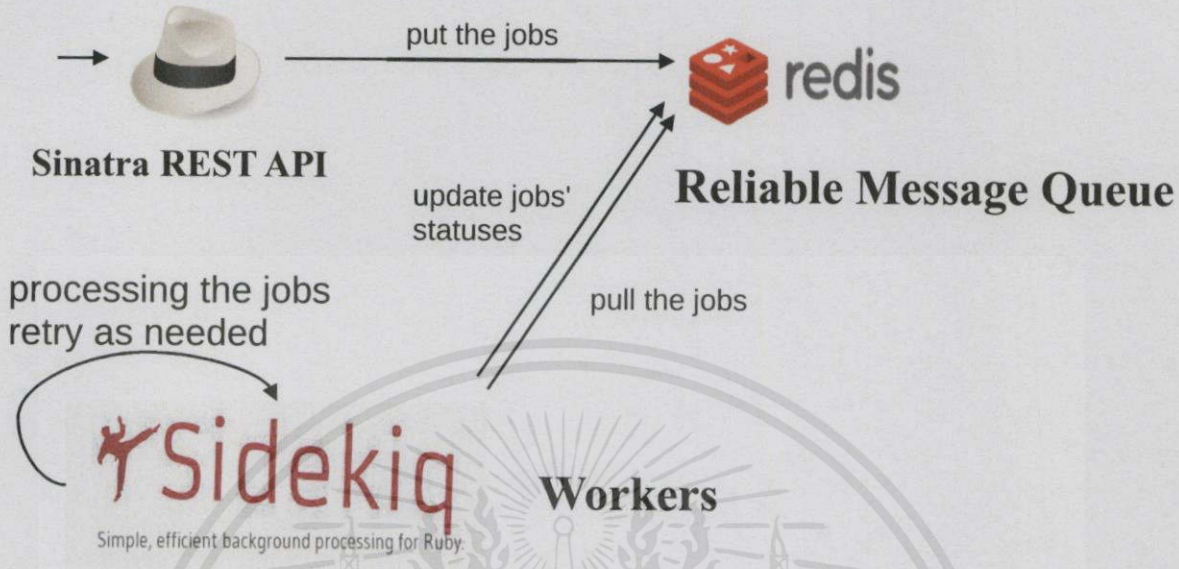
3.5.6.1 XService



รูปที่ 10 ภาพรวมการทำงานของ XService

1) ติดต่อจากภายนอกผ่าน REST API
2) ทำงานในรูปแบบ Asynchronous โดยสามารถรับงานต่าง ๆ จากภายนอก (ผ่าน REST API) มาใส่ไว้ใน Message Queue (MQ) ได้
3) การประมวลผลใช้แนวคิดตัวทำงาน (Workers) นั่นคือสามารถรับงานใน Message Queue มาประมวลผลในเวลาเดียวกันได้ หากมีตัวทำงานใดว่างอยู่ดังรูปข้างต้น

โดยการพัฒนาจะใช้ Sinatra [6] ของภาษา Ruby สำหรับ REST API ฐานข้อมูล Redis [7] สำหรับ Message Queue และ Sidekiq [8] ซึ่งเป็น Library ของภาษา Ruby สำหรับการทำงาน [9] มีลักษณะเป็นไปดังรูปด้านล่าง



รูปที่ 11 การทำงานของ XService

นอกจากนี้เพื่อให้รูปแบบของ Adapter ดังข้อ 3.5.2-3.5.4 หรืองานต่าง ๆ ที่จะส่งให้ตัวทำงานทำมีรูปแบบเดียวกันจึงได้กำหนด Interface ขึ้นมารูปแบบหนึ่ง เรียกว่า XModule ซึ่งเป็นแอปพลิเคชันในรูปแบบ Command-Line ที่มีรูปแบบดังนี้

3.5.6.2 XModule

```
 x_module_name start STRATEGY --option1 value1 --option2 value2
... [TARGETS]
 x_module_name stop  SESSIONID
 x module name desc  STRATEGY
```

โดยที่รูปแบบที่ 1 หมายถึงการสั่งดำเนินการโดยใช้กลยุทธ์แบบที่กำหนดไว้โดยตัวแปร STRATEGY ด้วยการตั้งค่าโดยใช้ตัวเลือก (options) ต่าง ๆ โดยอาจมีเครื่องเป้าหมายที่ต้องการดำเนินการได้ ซึ่งกำหนดโดยตัวแปร TARGETS

โดย TARGETS มีรูปแบบ ดังนี้

```
TARGETS FORMATS:
IP           | example: 192.168.99.102
RANGED_IPS   | example: 192.168.98.5-192.168.98.30
SUBNETED_IPS | example: 192.168.97.0/24
```

รูปแบบที่ 2 หมายถึงการหยุดการดำเนินการนั้น ซึ่งจะรับพารามิเตอร์เป็น SESSION ID นั่นคือหมายเลขการดำเนินการที่กำลังดำเนินการอยู่ โดยเป็นการทำงานตัวเลือกซึ่ง XModule นั้นสามารถรองรับหรือไม่ก็ได้ ซึ่งหากรองรับการดำเนินการนี้ การดำเนินการตามรูปแบบที่ 1 ต้องให้ผลลัพธ์หมายเลข SESSION ID ออกมาด้วย

รูปแบบที่ 3 หมายถึงการอธิบายกลยุทธ์ต่าง ๆ ที่รองรับโดย XModule นั้น ซึ่งกำหนดโดยตัวแปร STRATEGY

นอกจากนี้เพื่อให้การดำเนินการต่าง ๆ โดย XModule สามารถทำซ้ำและสามารถทำไปพร้อมกัน ๆ ได้ ข้อกำหนดที่เพิ่มเติมขึ้นมาคือ XModule จำเป็นต้องมีลักษณะการทำงานที่เป็น Idempotent เนื่องจากต้องสามารถดำเนินการซ้ำอีกครั้งได้หากเกิดความผิดพลาดขึ้น (ดูหัวข้อ *2.3.2 การจัดการความผิดพลาด* และหัวข้อ *3.5.5 XManager* ในส่วนการจัดการความผิดปกติในระบบ) และ Thread-safe เนื่องจากการดำเนินการอาจเกิดขึ้นแบบขนาน (ดูหัวข้อ *3.6 การทำงานแบบขนาน*)

ตัวอย่างการเรียกใช้งาน เช่น

nmap_adapter start simple 192.168.99.102

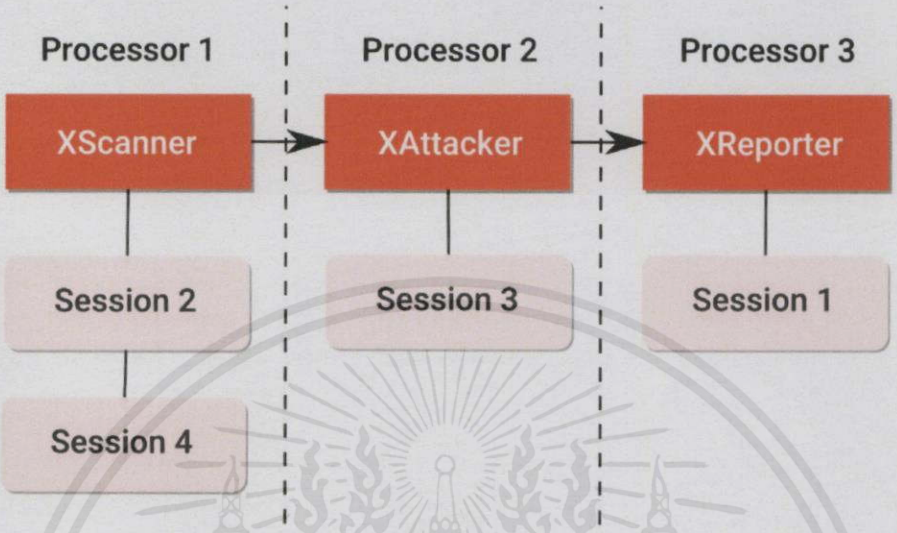
nmap_adapter stop 2104

w3af_adapter start owasp_topten 192.168.99.00/24

thc_hydra desc mysql

เป็นต้น

3.6 การทำงานแบบขนาน

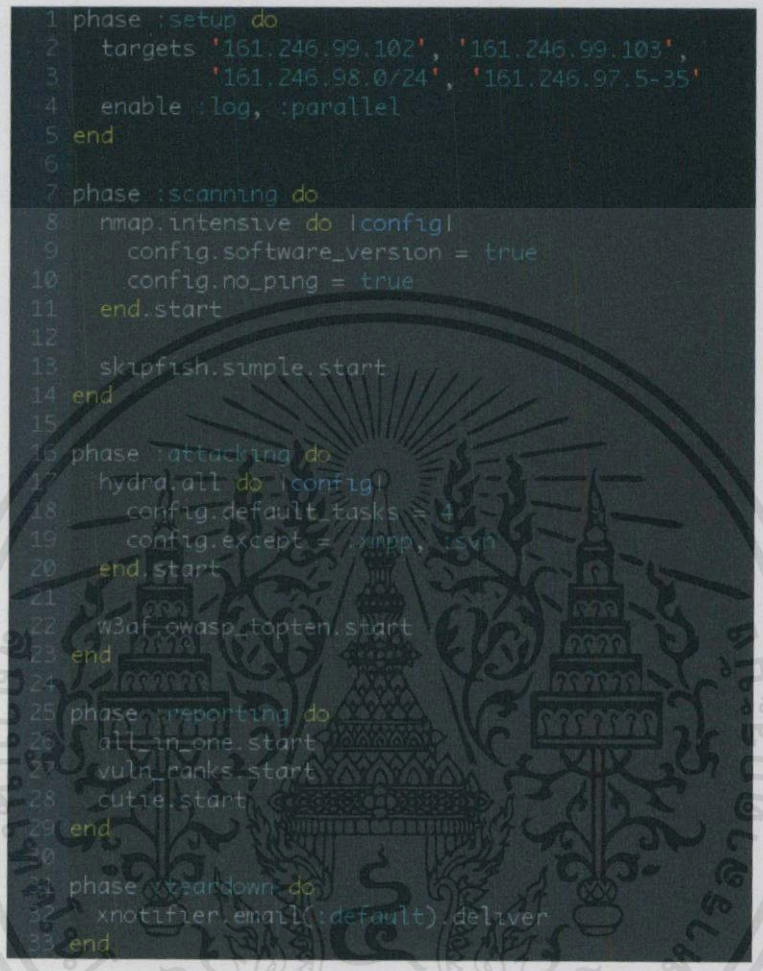


รูปที่ 12 ตัวอย่างการทำงานแบบขนานของซอฟต์แวร์

ระบบสามารถดำเนินการหลาย ๆ การดำเนินการ (Sessions) พร้อมกันได้ โดยการดำเนินการหมายถึง การประเมินความปลอดภัยไปยังระบบคอมพิวเตอร์เป้าหมายตั้งแต่เริ่มดำเนินการจนสิ้นสุดการดำเนิน (*ดูหัวข้อ 3.4 ขั้นตอนของระบบ*) หากการดำเนินการเหล่านั้นอยู่ในขั้นการดำเนินการที่แตกต่างกัน ตัวอย่าง เช่น จากรูปข้างต้น จะเห็นว่าการดำเนินการ Session ที่ 1 กำลังดำเนินการที่ XReporter ในขณะที่ Session ที่ 3 สามารถดำเนินการที่ XAttacker ได้เลย แต่การดำเนินการ Session ที่ 4 อาจจำเป็นต้องรอการดำเนินการ Session ที่ 2 ให้เสร็จในส่วนของ XScanner ก่อน เนื่องจากเครื่องที่ใช้ดำเนินการอาจมีทรัพยากรไม่เพียงพอสำหรับการดำเนินการแบบขนานในขั้นตอนนี้ เป็นต้น

อย่างไรก็ตามในการพัฒนาปัจจุบันสามารถที่จะทำให้การดำเนินการที่อยู่ในขั้นตอนการดำเนินการเดียวกันสามารถดำเนินการแบบไปด้วยกัน (Concurrency) แทนได้ โดยเมื่อการดำเนินการใด ๆ อยู่ระหว่างการติดต่อภายนอก (Disk I/O, Network I/O) จะสลับไปทำของการดำเนินการอื่นโดยอัตโนมัติ

3.7 การปรับแต่งการทำงานของซอฟต์แวร์



```
phase :setup do
  targets '161.246.99.102', '161.246.99.103',
          '161.246.98.0/24', '161.246.97.5-35'
  enable :log, :parallel
end

phase :scanning do
  nmap.intensive do |config|
    config.software_version = true
    config.no_ping = true
  end.start

  skipfish.simple.start
end

phase :attacking do
  hydra.all do |config|
    config.default_tasks = 4
    config.except = :xmpp, :svn
  end.start

  w3af.owasp_topten.start
end

phase :reporting do
  all_in_one.start
  vuln_ranks.start
  cutie.start
end

phase :teardown do
  xnotifier.email(:default).deliver
end
```

รูปที่ 13 ตัวอย่างสคริปต์การตั้งค่าซอฟต์แวร์

การปรับแต่งการทำงานของซอฟต์แวร์สามารถกำหนดผ่านสคริปต์ดังรูปข้างต้นได้ โดยสามารถตั้งค่าการทำงานในขั้นตอนต่าง ๆ ในหัวข้อ 3.4 *ขั้นตอนของระบบ* ซึ่งปัจจุบันสามารถตั้งค่าได้ดังนี้

1) การตั้งค่าหมายเลข IP เป้าหมายที่ต้องการประเมินความปลอดภัย
2) การตั้งค่าเครื่องมือต่าง ๆ ในการประเมินความปลอดภัย

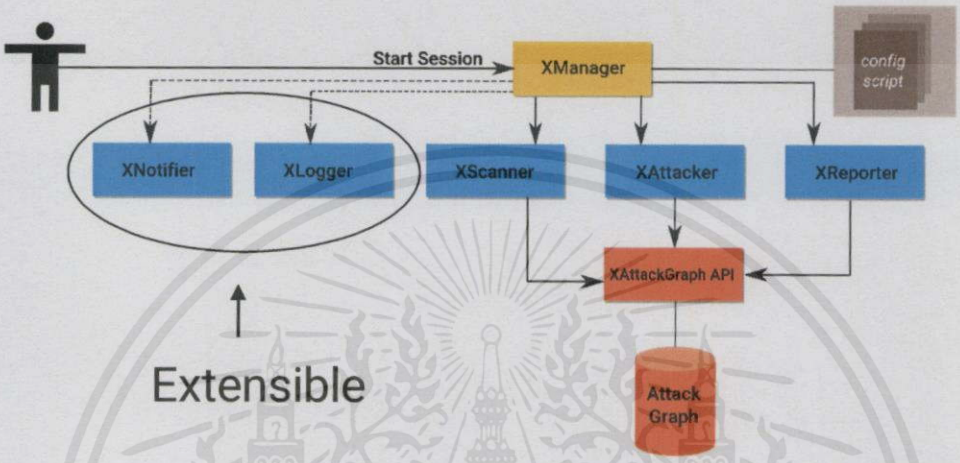
3.7.1 การพัฒนา

ใช้ภาษาสคริปต์ Ruby แทนสคริปต์การตั้งค่า โดยอ่านการตั้งค่าต่าง ๆ ผ่าน Ruby Interpreter โดยใช้เทคนิค Metaprogramming [10] ของภาษา Ruby ซึ่งเป็นการเรียกใช้งานคำสั่ง

โดยที่ไม่จำเป็นต้องมีการกำหนดคำสั่งนั้นล่วงหน้าในการอ่านและกำหนดค่าต่าง ๆ ใน ProjectX เช่น หากมี XModule มีการเพิ่มกลยุทธ์ (STRATEGY) ใหม่เข้ามา จะสามารถเรียกใช้งานได้ทันที เป็นต้น

3.8 การขยายต่อของซอฟต์แวร์

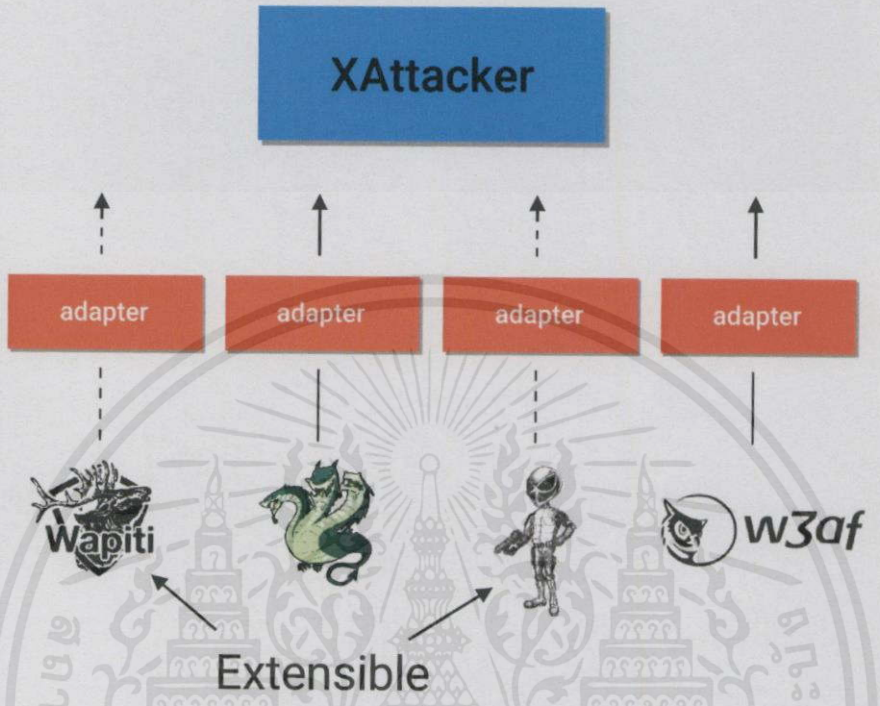
3.8.1 การเพิ่มส่วนประกอบซอฟต์แวร์ย่อย



รูปที่ 14 ตัวอย่างการเพิ่มส่วนประกอบซอฟต์แวร์ย่อย

การเพิ่มส่วนประกอบซอฟต์แวร์ย่อยเข้ามาในระบบสามารถเพิ่มได้โดยเพิ่มส่วน REST API ใหม่เข้ามาในระบบและเพิ่มการจัดการใน XManager ให้สามารถประสานการทำงานใหม่ได้ เช่น จากรูปเราอาจเพิ่มส่วน XNotifier และ XLogger สำหรับการแจ้งเตือนและเก็บบันทึกการดำเนินการต่าง ๆ เข้ามาโดยใช้ REST API และควบคุมการประสานงานโดย XManager ซึ่งการเพิ่มการทำงานเหล่านี้ไม่กระทบต่อโครงสร้างและการทำงานเดิมของระบบ

3.8.2 การเพิ่มการทำงานในส่วนประกอบซอฟต์แวร์ย่อย



รูปที่ 15 ตัวอย่างการเพิ่มการทำงานในส่วนประกอบซอฟต์แวร์ย่อย

สามารถเพิ่มได้โดยการเพิ่ม XModule ใหม่ (ในตัวอย่างด้านบนคือ Adapter สำหรับซอฟต์แวร์ Wapiti และ Nikto) เข้าไป โดยสามารถตั้งค่าการเรียกใช้และการทำงานในสคริปต์การตั้งค่าในหัวข้อ 3.7 สำหรับ XModule ใหม่นั้นได้ทันที โดยรูปแบบการเรียกใช้งาน XModule จะต้องเป็นไปตามข้อกำหนดในหัวข้อ *3.5.6.2 XModule*

3.9 Mobile Client

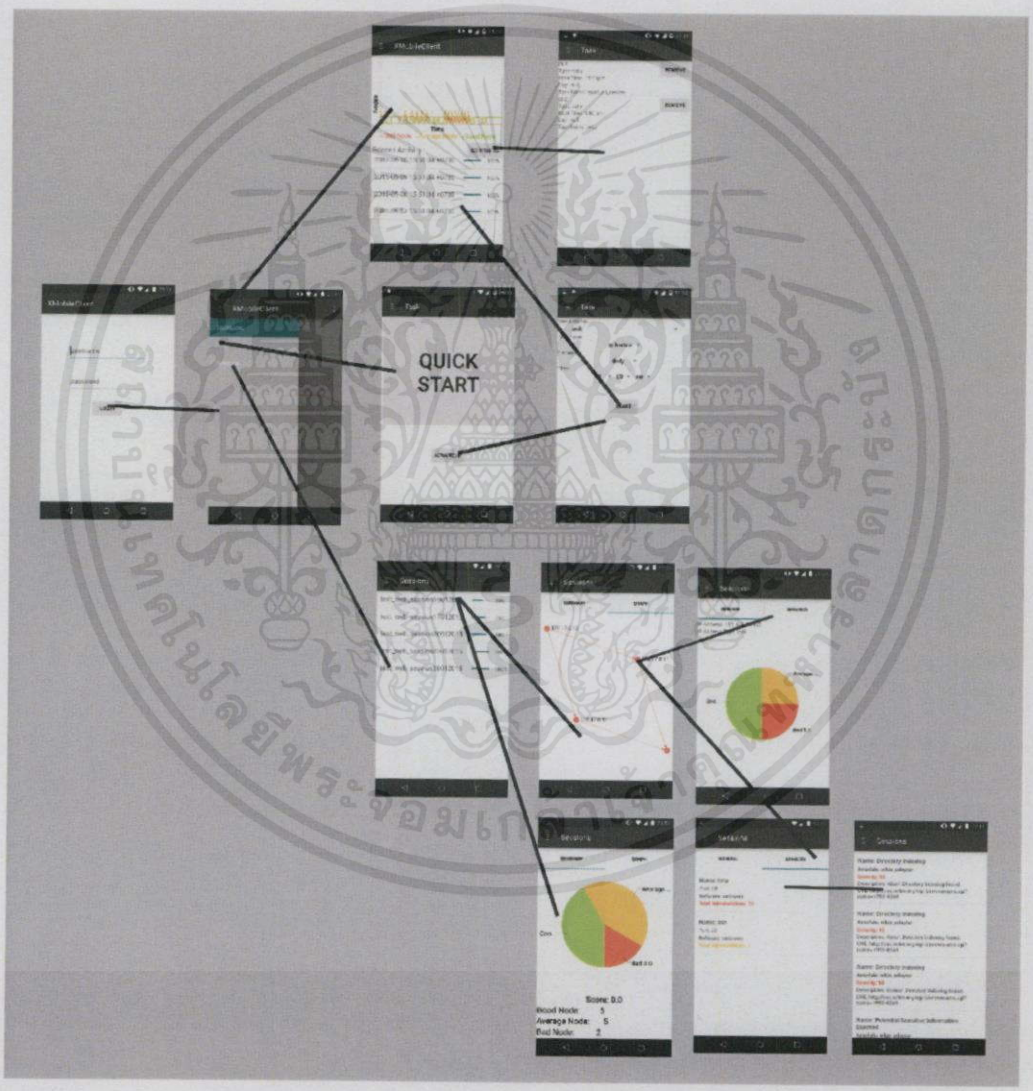
3.9.1 การทำงานของแอปพลิเคชัน

แอปพลิเคชันที่ใช้ทำงานร่วมกับระบบนี้ถูกพัฒนาขึ้นบน Android Studio โดยใช้ Library อื่น ๆ เข้ามาช่วยในการแสดงผลข้อมูลได้แก่ achartengine ใช้สำหรับแสดงผล Graph และ Chart ต่างๆ และใช้ Sigma.js [11] ในการแสดงผล Attack Graph ภายในแอปพลิเคชัน

การเชื่อมต่อระหว่างแอปพลิเคชั่นกับระบบนั้นจะใช้การติดต่อผ่าน REST API โดยจะสามารถแบ่งเป็น 2 ส่วนโดยหลัก คือ ส่วนการแสดงผลและส่วนการสั่งการ โดยในส่วนของการแสดงผลจะรับข้อมูลเข้ามาในรูปแบบของ JSON จากนั้นแอปพลิเคชันนี้จะทำการแปลงจากข้อมูลที่เป็นตัวอักษรให้ออกมาในรูปต่าง ๆ เช่น Attack Graph และ Chart หรือนำข้อมูลในส่วนของข้อมูลบริการและช่องโหว่มาจัดอันดับและเรียบเรียงใหม่ให้สามารถเข้าใจได้ง่ายขึ้น และในส่วนของการสั่งการก็จะทำการส่งข้อมูลผ่าน HTTP ไปยังเครื่องแม่ข่าย ProjectX เพื่อให้ระบบทำงานตามคำสั่งที่ได้สั่งการไป

3.9.2 ภาพรวมของแอปพลิเคชัน

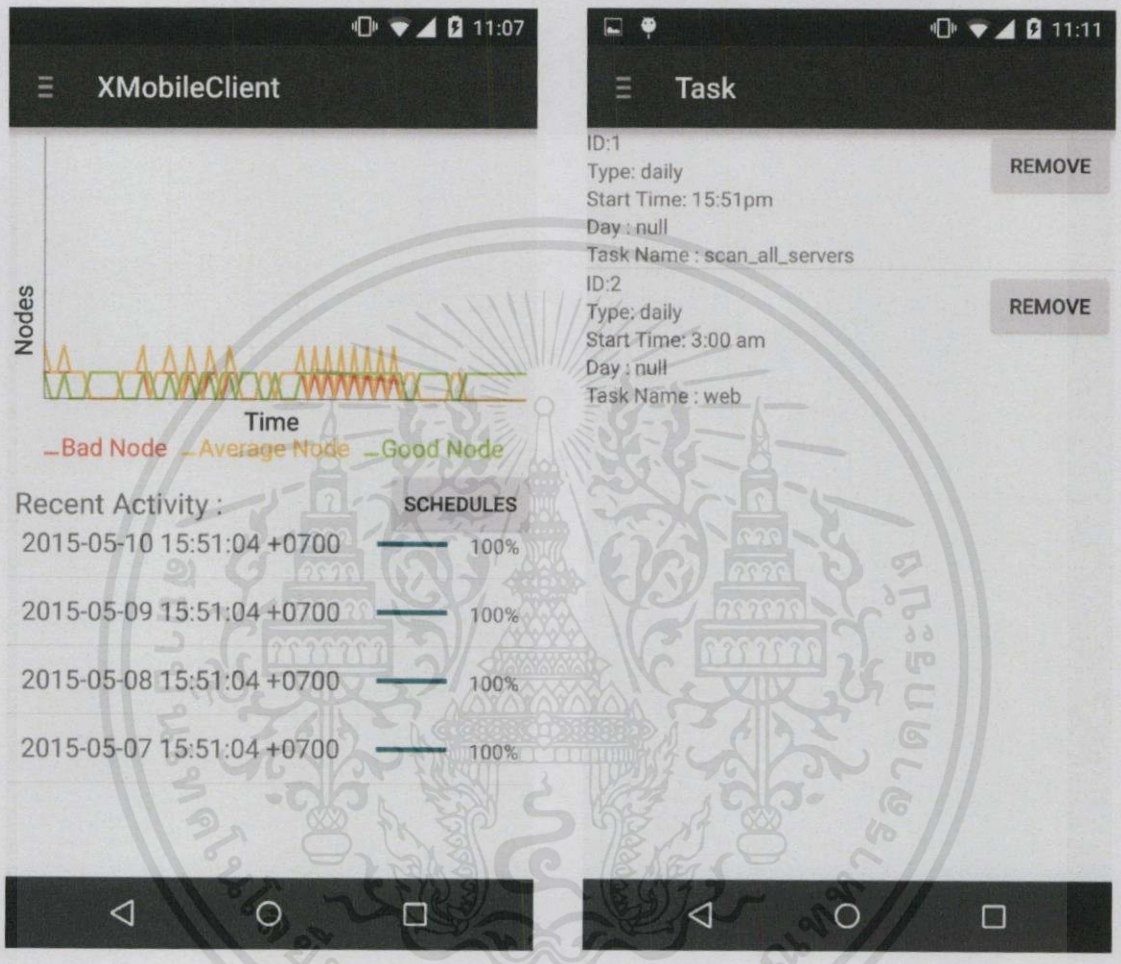
แอปพลิเคชันประกอบด้วยหน้า Dashboard สำหรับดูภาพรวมของระบบ หน้า Task สำหรับสั่งการ หน้า Session สำหรับดูการดำเนินการที่ผ่านมา โดยในแต่ละ Session ประกอบด้วย หน้ารายงานการประเมินความปลอดภัยของการดำเนินการโดยรวมและหน้ารายงานการประเมินความปลอดภัยของโหนดที่ถูกประเมิน



รูปที่ 16 ภาพรวมของแอปพลิเคชันบนอุปกรณ์ Mobile

3.9.3 รายละเอียดแอปพลิเคชัน

3.9.2.1 หน้า Dashboard



รูปที่ 17 ตัวอย่างหน้า Dashboard ของ Mobile Client

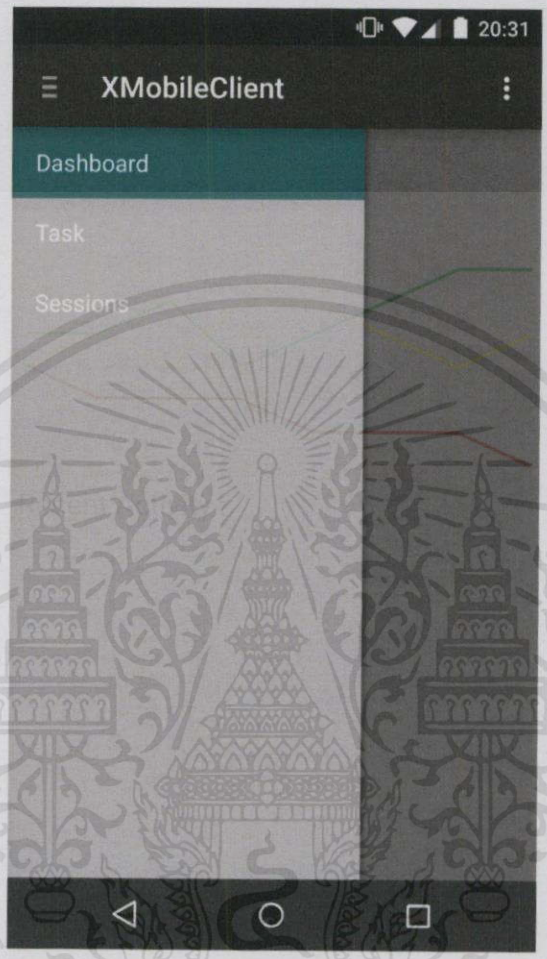
(ก) แสดงหน้าหลักของ Dashboard

(ข) แสดงหน้ากำหนดการล่วงหน้า (Schedules)

เป็นหน้าแรกหลังจากการเข้าสู่ระบบ แสดงกราฟคะแนนความปลอดภัยของระบบแต่ละระบบ โดยมีแกน Y แทนจำนวนโหนด และแกน X แทนเวลาสัมพัทธ์ที่ทำการประเมิน (ดังรูปแสดง 3 ระบบด้วยเส้นสีที่ต่างกัน 3 สี) ตามแต่ละครั้งที่ทำการประเมินระบบนั้น โดยค่ามากหมายถึงระบบมีความปลอดภัยสูง ซึ่งค่าดังกล่าวได้มาจากระดับความปลอดภัยของแต่โหนดซึ่งเกิดจากการนำระดับความปลอดภัยของแต่ละบริการภายในโหนดมาหาค่าเฉลี่ย

นอกจากนี้ยังแสดงการดำเนินการล่าสุด 4 การดำเนินการ ซึ่งแถบสีเขียวและค่าเปอร์เซ็นต์บอกถึงความคืบหน้าในการดำเนินการนั้น ผู้ใช้สามารถกดเลือกการดำเนินการที่เสร็จสิ้นแล้ว (ความคืบหน้า 100 %) เพื่อดูรายงานการประเมินความปลอดภัยในการดำเนินการนั้นได้ และผู้ใช้สามารถเลือกปุ่ม SCHEDULE ซึ่งจะนำไปสู่หน้าแสดงกำหนดการล่วงหน้า ดังรูปที่ 17 (ข) ได้ โดยในหน้านี้ผู้ใช้สามารถกดปุ่ม REMOVE เพื่อลบการกำหนดการล่วงหน้าที่ไม่ต้องการออกได้

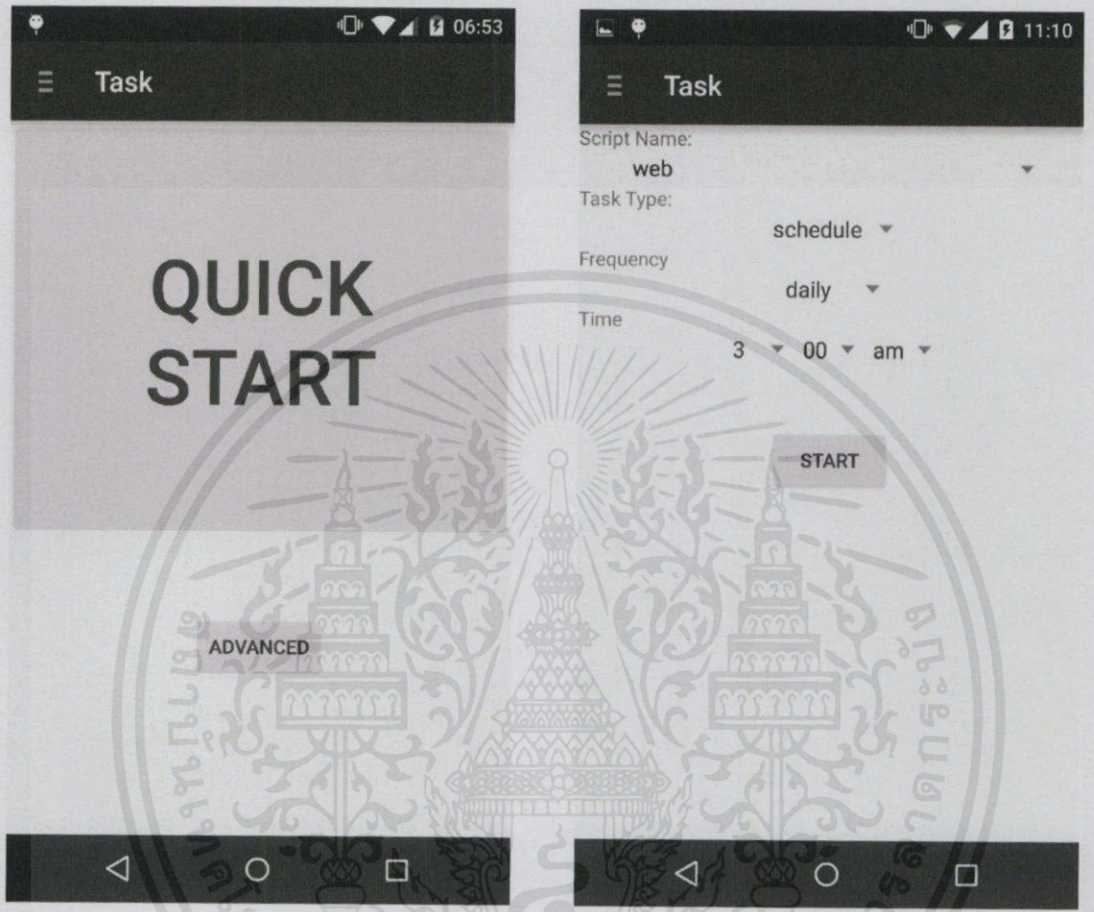
3.9.2.2 Menu ด้านข้าง



รูปที่ 18 Menu **ด้านข้างของ** Mobile Client

เปิดแถบเมนูด้านซ้ายดูการกดและลากนิ้วจากซ้ายไปขวา เป็นแถบเมนูที่สามารถเข้าถึงได้จากทุกหน้าภายในแอปพลิเคชัน ผู้ใช้สามารถเลือกสลับระหว่างหน้า Dashboard หน้าสั่งงาน (Task) และหน้าแสดงการดำเนินการทั้งหมด (Sessions) ได้เพื่อความสะดวกในการใช้งาน

3.9.2.3 หน้าสั่งงาน (Task)

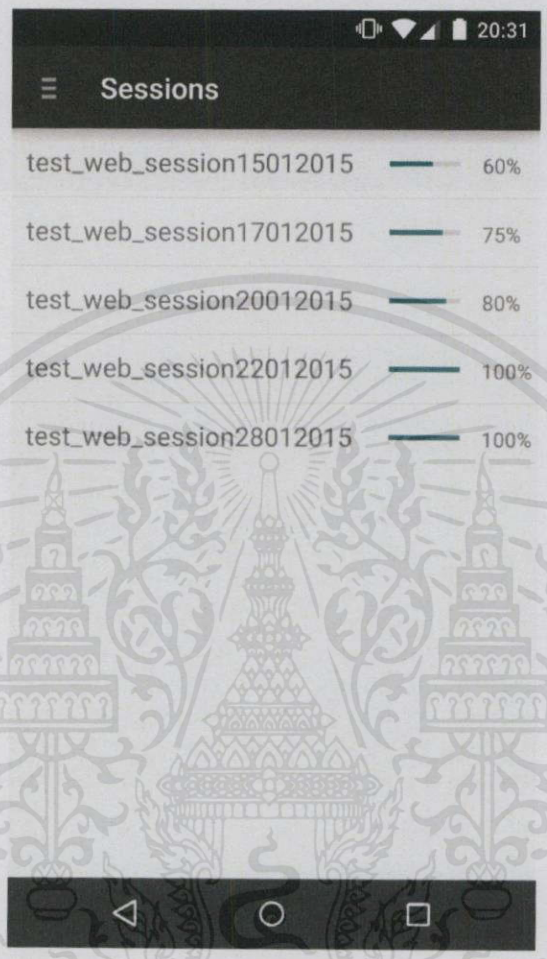


(ก) (ข)

รูปที่ 19 หน้าสั่งงานของ Mobile Client
(ก) แสดงหน้าการสั่งงานอย่างง่าย (Quick Start)
(ข) แสดงหน้าการสั่งงานแบบ Advanced

เป็นหน้าสำหรับสั่งการโดยสามารถสั่งการประเมินความปลอดภัยของระบบได้โดยเลือกปุ่ม Quick Start หรือผู้ใช้สามารถเลือกปุ่ม Advanced ซึ่งจะนำไปสู่หน้าการเลือกงาน (Task) ที่กำหนดไว้โดยสคริปต์การตั้งค่าในหัวข้อ 3.7 *การปรับแต่งการทำงานของซอฟต์แวร์* ของซอฟต์แวร์เครื่องแม่ข่าย ProjectX โดยในการเลือกงานนี้ผู้ใช้ยังสามารถเลือกกำหนดการล่วงหน้าเพื่อให้ระบบทำการประเมินความปลอดภัยซ้ำในระยะเวลาที่กำหนดได้ ดังรูปที่ 19 (ข)

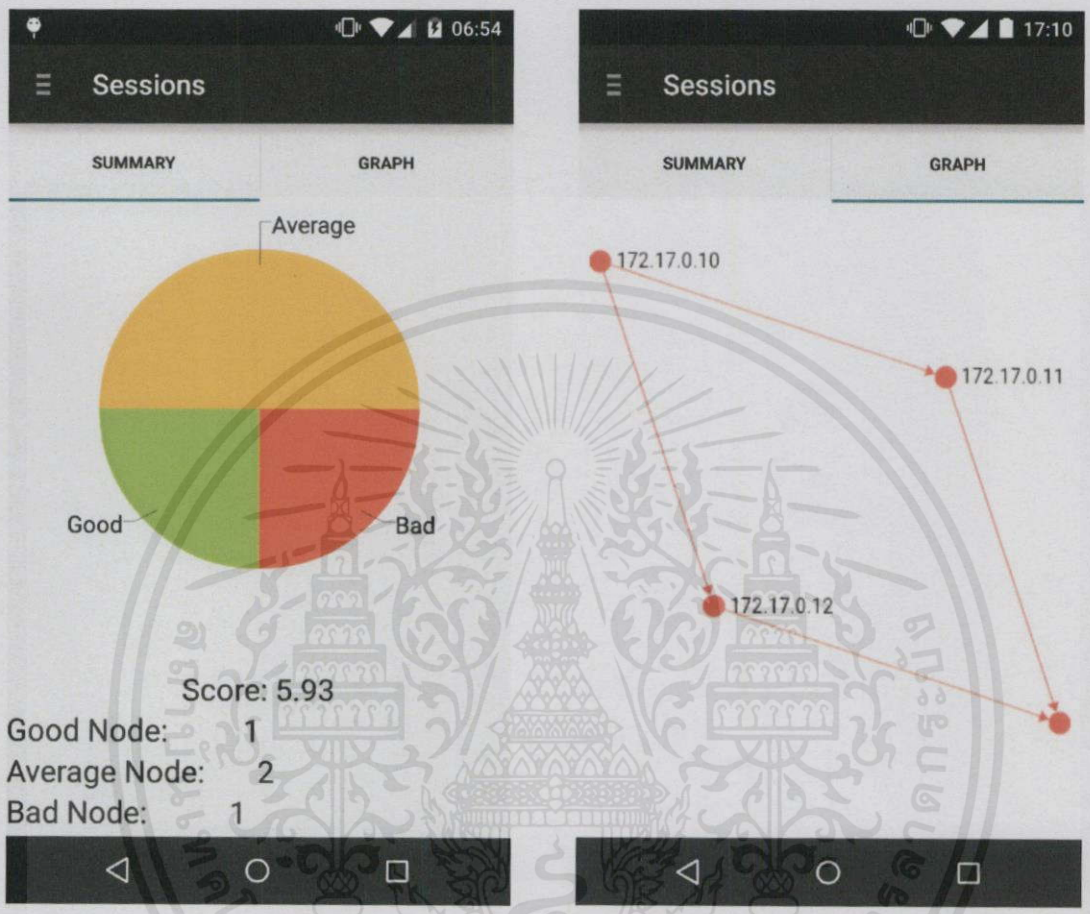
3.9.2.4 หน้าการดำเนินการทั้งหมด (Sessions)



รูปที่ 20 ตัวอย่างหน้าแสดงการดำเนินการทั้งหมดของ Mobile Client

แสดงถึงการดำเนินการทั้งหมดโดยผู้ใช้ปัจจุบันทั้งที่อยู่ในระหว่างดำเนินการและที่ดำเนินการเสร็จแล้ว โดยการดำเนินการที่เสร็จแล้วสามารถกดเพื่อดูรายงานการประเมินความปลอดภัยของการดำเนินการนั้นได้

3.9.2.5 หน้ารายงานการประเมินความปลอดภัยของการดำเนินการโดยรวม



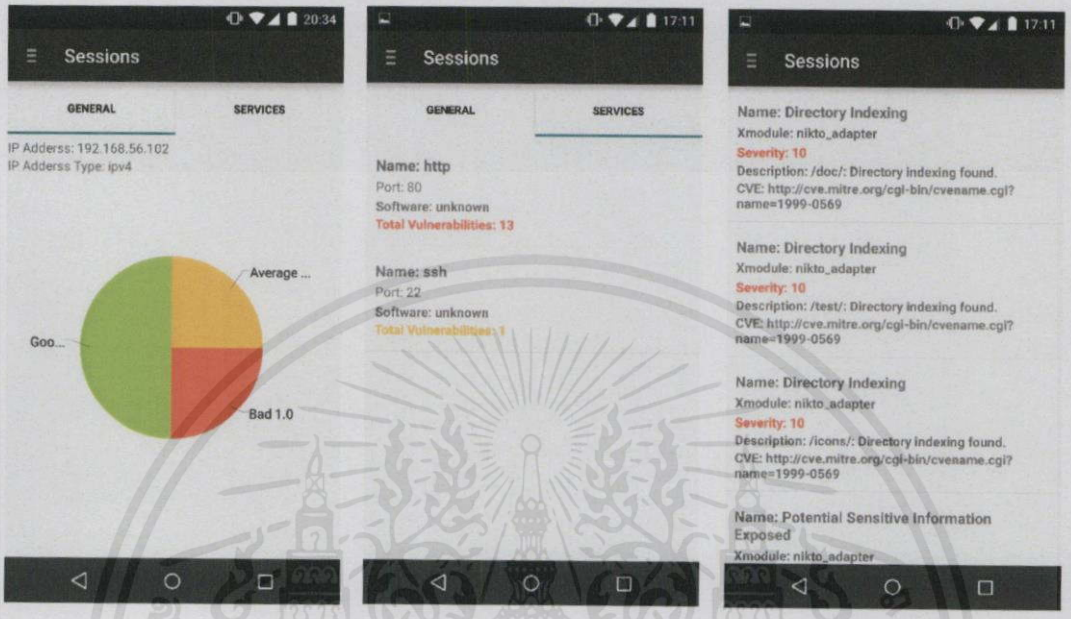
(ก) (ข)

รูปที่ 21 ตัวอย่างหน้ารายงานกระประเมินความปลอดภัย
(ก) แสดงตัวอย่างหน้ารายงานโดยสรุป
(ข) แสดงตัวอย่างหน้า Attack Graph

ในหน้าสรุป (SUMMARY) ดังรูปที่ 21 (ก) แสดงรายงานโดยสรุปในลักษณะของชาร์ตวงกลม โดยแสดงสัดส่วนของโหนดต่าง ๆ (เครื่องให้บริการ) ที่มีคะแนนความปลอดภัยสูง (สีเขียว) กลาง (สีส้ม) และต่ำ (สีแดง) ในระบบที่ทำการประเมิน และแสดงตัวเลขจำนวนโหนดตามจริงด้านล่าง

ในรูปที่ 21 (ข) หน้า GRAPH จะแสดง Attack Graph ของระบบ ซึ่งสามารถกดที่โหนดต่าง ๆ เพื่อดูรายงานโดยละเอียดของโหนดนั้นได้

3.9.2.6 หน้ารายงานการประเมินความปลอดภัยของโหนดที่ถูกประเมิน



(ก) (ข) (ค)

รูปที่ 22 ตัวอย่างหน้ารายงานการประเมินความปลอดภัยของโหนดที่ถูกประเมิน

(ก) แสดงตัวอย่างหน้าข้อมูลทั่วไป

(ข) แสดงตัวอย่างหน้าบริการทั้งหมดของโหนดนั้น

(ค) แสดงตัวอย่างหน้าช่องโหว่ทั้งหมดของโหนดนั้น

หน้าข้อมูลทั่วไป (GENERAL) ดังรูปที่ 22 (ก) แสดงรายงานโดยสรุปในลักษณะของชาร์ตวงกลม โดยแสดงสัดส่วนของบริการ (services) ที่มีคะแนนความปลอดภัยสูง (สีเขียว) กลาง (สีส้ม) และต่ำ (สีแดง) ของโหน นั้น โดยคะแนนดังกล่าวได้มาจากค่าระดับความปลอดภัยของแต่ละบริการภายในโหนดนั้น

สามารถกดดูบริการทั้งหมดของโหนดนั้นได้ (แท็บ SERVICES) ดังรูปที่ 22 (ข) เพื่อดูรายการของบริการทั้งหมดในโหนดนั้น และยังสามารถกดที่บริการนั้น ๆ เพื่อดูข้อมูลรายการช่องโหว่ของบริการนั้นได้ ดังรูปที่ 22 (ค)

บทที่ 4

การทดลองและผลการทดลอง

4.1 เครื่องเป้าหมายที่ใช้ในการทดลอง:

ตารางที่ 1 เครื่องเป้าหมายต่าง ๆ ที่ใช้ในการทดสอง

ชื่อเครื่อง	ระบบปฏิบัติการ	บริการที่เปิด
M	Metasploitable2	(บริการทั้งหมดที่มีของ Metasploitable2)
LB	Ubuntu 12.04	HTTPS (443) [HTTPS PROXY], SSH (22)
W1	Metasploitable2	HTTP (80) , SSH (22)
W2	Metasploitable2	HTTP (80) , SSH (22)
DB	Metasploitable2	MySQL (3306) , SSH (22)

ในการทดลองได้มีการใช้ *Metasploitable2* โดยเป็นเครื่องคอมพิวเตอร์เสมือนซึ่งถูกติดตั้งระบบปฏิบัติการและซอฟต์แวร์ในเวอร์ชันที่พบว่ามีช่องโหว่ในอดีต โดยจุดประสงค์เพื่อใช้สำหรับทำการทดลองการโจมตีหรือการประเมินความปลอดภัยโดยเครื่องมือต่าง ๆ และเพื่อใช้อธิบายการทำงานของช่องโหว่ต่าง ๆ ที่โดยทั่วไปสามารถพบได้บ่อยครั้งในสถาณการณ์จริง โดยนำเครื่องเสมือนนี้มาใช้เพื่อทำการดำเนินการประเมินความปลอดภัยโดย ProjectX และเปรียบเทียบผลลัพธ์การดำเนินการกับบริการและช่องโหว่ที่มีอยู่จริงของเครื่องเสมือนนี้ [12]

4.2 เครื่องที่ใช้ในการดำเนินการประเมินความปลอดภัย:

เป็นเครื่องเสมือน (Virtual Machine) ที่มีการติดตั้งระบบปฏิบัติการ Ubuntu Server 14.04.1 โดยมีจำนวนคอร์ CPU จำนวน 2 คอร์ หน่วยความจำขนาด 2048 MB และใช้ ProjectX เวอร์ชัน 0.1.0

4.3 การทดลองที่ 1 การประเมินเครื่องเดี่ยว

4.3.1 สภาพแวดล้อม

เครื่องเป้าหมาย: Metasploitable2 (IP: 192.168.56.102)

รูปที่ 23 สภาพแวดล้อมการทดสอบที่ 1

(เครื่องดำเนินการ 192.168.56.1) **(192.168.56.102)**

เป็นการดำเนินการประเมินความปลอดภัยไปที่เครื่องเดี่ยว ผ่านการสั่งงานโดย Mobile Client

4.3.2 การตั้งค่า

```
m = '192.168.56.102'

update_progress(10)

phase :scanning do
  nmap_adapter.simple do
  end.start(m)
end

sleep 30.0
update_progress(45)

phase :attacking do
  hydra_adapter.ssh do |config|
    config.user_list  = File.expand_path('../../resources/user_list.txt', __FILE__)
    config.dictionary = File.expand_path('../../resources/dictionary.txt', __FILE__)
    config.output     = File.expand_path("../../resources/hydra_result#{Time.now.to_i}.txt", __FILE__)
  end.start(m)

  nikto_adapter.simple do
  end.start(m)
end

sleep 50.0
update_progress(80)

phase :reporting do
  all_in_one.json do
  end.start('all_in_one.json')
end

sleep 2.0
update_progress(100)
```

รูปที่ 24 การตั้งค่าสำหรับการทดลองที่ 1

ทำการแสกนหาบริการและตรวจหาซอฟต์แวร์ที่ใช้ในบริการนั้น ๆ โดยการใช้ XModule nmap_adapter และทำการโจมตีหาช่องโหว่รหัสผ่านของบริการ ssh ด้วย XModule hydra_adapter และโจมตีเว็บไซต์ด้วย nikto_adapter และสุดท้ายจึงสร้างรายงานในรูปแบบ JSON ผ่าน XModule all_in_one ซึ่งจะได้ใช้ในการแสดงผลของ Mobile และ Web Client ต่อไป

4.3.3 ผลการดำเนินการ

ระยะเวลาที่ใช้ในการดำเนินการ: 85.03 วินาที

โดยแบ่งเป็นการแสกนเครือข่ายโดย nmap_adapter 30 วินาที การประเมินความปลอดภัยโดย hydra_adapter และ nikto_adapter รวม 50 วินาที และการสร้างรายงานการประเมินความปลอดภัยโดย all_in_one อีก 5 วินาที

Attack Graph:

รูปที่ 25 ผลการทดลองที่ 1 แสดงโดย Mobile Client

บริการที่ตรวจพบ (23):

ตารางที่ 2 บริการและจำนวนช่องโหว่ที่พบในการทดลองที่ 1

บริการ	พอร์ต	ซอฟต์แวร์	จำนวนช่องโหว่ที่พบ
ftp	21	vsftpd 2.3.4	
ssh	22	OpenSSH 4.7p1	ร้ายแรง: 1
telnet	23	Linux telnetd	
smtp	25	Postfix smtpd	
domain	53	ISC BIND 9.4.2	
http	80	Apache httpd 2.2.8	ต่ำ: 2, ปานกลาง: 9: ร้ายแรง: 6
rpcbind	111	unknown	
netbios-ssn	139	Samba smbd 3.X	
microsoft-ds	445	Samba smbd 3.X	
exec	512	netkit-rsh rexecd	
login	513	unknown	
shell	514	unknown	
rmiregistry	1099	GNU Classpath grmiregistry	
ingreslock	1524	Metasploitable root shell	
nfs	2049	Unknown	
ccproxy-ftp	2121	ProFTPD 1.3.1	
mysql	3306	MySQL 5.0.51a-3	
postgresql	5432	PostgreSQL DB 8.3.0 - 8.3.7	
vnc	5900	VNC	
X11	6000	Unknown	
irc	6667	Unreal ircd	
ajp13	8009	Apache Jserv	
unknow	8180	Apache Tomcat JSP 1.1	

ช่องโหว่ที่ตรวจพบ (18):

ตรวจพบช่องโหว่เป็นจำนวนทั้งหมด 18 ช่องโหว่ โดยใน 18 ช่องโหว่นี้มีช่องโหว่ร้ายแรงถึง 6 ช่องโหว่ นั่นคือ รหัสผ่านที่ไม่ดี (ช่องโหว่ weak_password) ของบริการ ssh และที่เหลือจะเป็นการเปิดเผยการตั้งค่าและการเปิดเผยไดเรกทอรีอาจนำไปสู่การค้นหาช่องโหว่หรือข้อมูลที่ส่งผลร้ายแรงต่อไป เช่น การตั้งค่ารหัสผ่านฐานข้อมูล เป็นต้น ซึ่งเป็นช่องโหว่ที่มีอยู่จริงของ Metasploitable2 [12]

4.4 การทดลองที่ 2 การประเมินเครื่องเดี่ยวและการเก็บข้อมูลเพิ่มเติม

4.4.1 สภาพแวดล้อม

เครื่องเป้าหมาย: M Metasploitable2 (IP: 192.168.56.102)

รูปที่ 26 สภาพแวดล้อมการทดสอบที่ 2

(เครื่องดำเนินการ 192.168.56.1) **(192.168.56.102)**

เป็นการดำเนินการประเมินความปลอดภัยไปที่เครื่องเดี่ยว ผ่านการสั่งงานโดย Mobile Client เหมือนกับการทดลองที่ 1 แต่ในการทดลองนี้จะเพิ่มส่วนเก็บข้อมูลเพิ่มเติมในกรณีที่สามารถเข้าถึงเครื่องปลายทางได้เข้าไป

4.4.2 การตั้งค่า

```
m = '192.168.56.102'

update_progress(10)

phase :scanning do
  nmap_adapter.simple do
  end.start(m)
end

sleep 30.0
update_progress(40)

phase :attacking do
  hydra_adapter.ssh do |config|
    config.user_list  = File.expand_path('../../resources/user_list.txt', __FILE__)
    config.dictionary = File.expand_path('../../resources/dictionary.txt', __FILE__)
    config.output     = File.expand_path("../../resources/hydra_result#{Time.now.to_i}.txt", __FILE__)
  end.start(m)

  nikto_adapter.simple do
  end.start(m)
end

sleep 50.0
update_progress(75)

phase :after_attacking do
  ssh_agent.netstat do
  end.start(m)

  ssh_agent.hostname do
  end.start(m)

  ssh_agent.route do
  end.start(m)
end

sleep 20.0
update_progress(85)

phase :reporting do
  all_in_one.json do
  end.start('all_in_one.json')
end

sleep 2.0
update_progress(100)
```

รูปที่ 27 การตั้งค่าสำหรับการทดลองที่ 2

โดยเพิ่มเติมจากการทดลองที่ 1 ในส่วนของหลังการโจมตีได้สั่งให้ใช้ XModule ชื่อว่า ssh_agent ซึ่งทำการเก็บข้อมูลเพิ่มเติมจากเครื่องที่มีช่องโหว่ที่สามารถสั่งรันคำสั่งได้ (เช่น weak_password จากผลการทดลองที่ 1)

4.4.3 ผลการดำเนินการ

ระยะเวลาที่ใช้ในการดำเนินการ: 104.034 วินาที

โดยเวลาที่เพิ่มขึ้นมาจากการทดลองที่ 1 คือ เวลาที่ใช้ในการเก็บข้อมูลเพิ่มเติมโดย ssh_agent (19 วินาที)

ให้ผลลัพธ์การค้นหาบริการและช่องโหว่เหมือนการทดลองที่ 1 แต่ได้ข้อมูลเพิ่มเติมจากการดึงค่าจากการเข้าไปในเครื่องเป้าหมายเพิ่มเติมโดย ssh_agent ดังนี้

netstat:

นอกจากข้อมูลที่ได้จาก nmap_adapter แล้ว จากการเก็บข้อมูลตรงนี้ให้รู้อีกว่ามีบริการอะไรบ้างที่ตรวจไม่พบ นั่นคือ rpc.statd ที่พอร์ต 3739 เนื่องจากเป็นการทำงานผ่านอินเตอร์เฟส localhost ซึ่งเข้าไม่ถึงจากภายนอก

hostname:

metasploitable

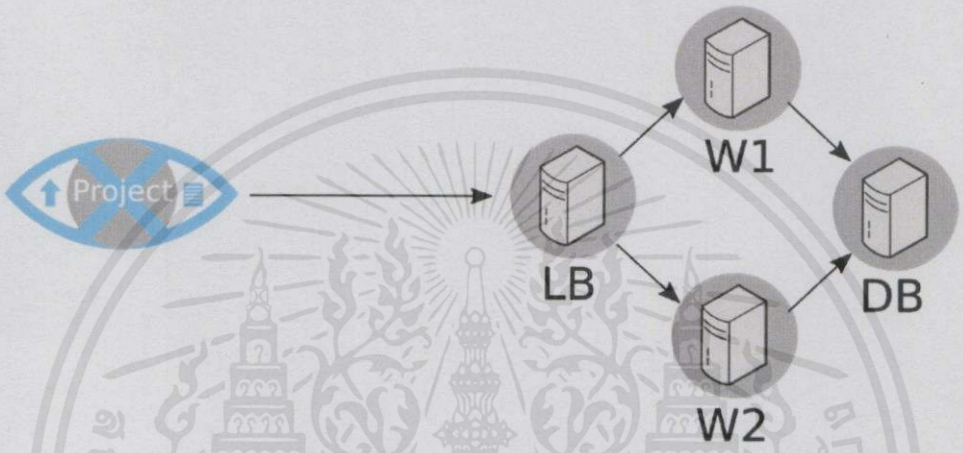
route:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.56.0	*	255.255.255.0	U	0	0	0	eth0

4.5 การทดลองที่ 3 การประเมินหลายเครื่อง

4.5.1 สภาพแวดล้อม

เครื่องเป้าหมาย: LB (IP: 192.168.56.110), W1 (IP: 192.168.56.107), W2 (IP: 192.168.56.108), DB (IP: 192.168.56.109)



รูปที่ 28 สภาพแวดล้อมการทดลองที่ 3

(เครื่องดำเนินการ 192.168.56.1)

(192.168.56.110, 192.168.56.107, 192.168.56.108, 192.168.56.109)

เป็นการดำเนินการประเมินความปลอดภัยไปที่เครื่องหลายเครื่อง ผ่านการสั่งงานโดย Mobile Client โดยเครื่อง W1 และ W2 ให้บริการเว็บไซต์ในลักษณะเดียวกันผ่านฐานข้อมูลจากเครื่อง DB โดยมีเครื่อง LB ทำหน้าที่เป็น Load-Balancer และ HTTPS Proxy โดยกรณีนี้เป็นกรณีที่เครื่องดำเนินการ (ProjectX) อยู่ในเครือข่ายเดียวกับเครื่องเหล่านี้

4.5.2 การตั้งค่า

```
lb = '192.168.56.110'
w1 = '192.168.56.107'
w2 = '192.168.56.108'
db = '192.168.56.109'

update_progress(10)

phase :scanning do
  nmap_adapter.simple do
  end.start(lb, w1, w2, db)
end

sleep 320.0
update_progress(40)

phase :after_scanning do
  s_inf.app_db do
  end.start
end

sleep 2.0
update_progress(50)

phase :attacking do
  hydra_adapter.ssh do |config|
    config.user_list  = File.expand_path('../../resources/user_list.txt', __FILE__)
    config.dictionary = File.expand_path('../../resources/dictionary.txt', __FILE__)
    config.output     = File.expand_path("../../resources/hydra_result#{Time.now.to_i}.txt", __FILE__)
  end.start(lb, w1, w2, db)

  hydra_adapter.mysql do |config|
    config.user_list  = File.expand_path('../../resources/user_list.txt', __FILE__)
    config.dictionary = File.expand_path('../../resources/dictionary.txt', __FILE__)
    config.output     = File.expand_path("../../resources/hydra_result#{Time.now.to_i + 1}.txt", __FILE__)
  end.start(db)

  nikto_adapter.simple do
  end.start(w1, w2)

  nikto_adapter.simple do |config|
    config.port = 443
  end.start(lb)
end

sleep 150.0
update_progress(80)

phase :reporting do
  all_in_one.json do
  end.start('all_in_one.json')
end

sleep 2.0
update_progress(100)
```

รูปที่ 29 การตั้งค่าการทดลองที่ 3

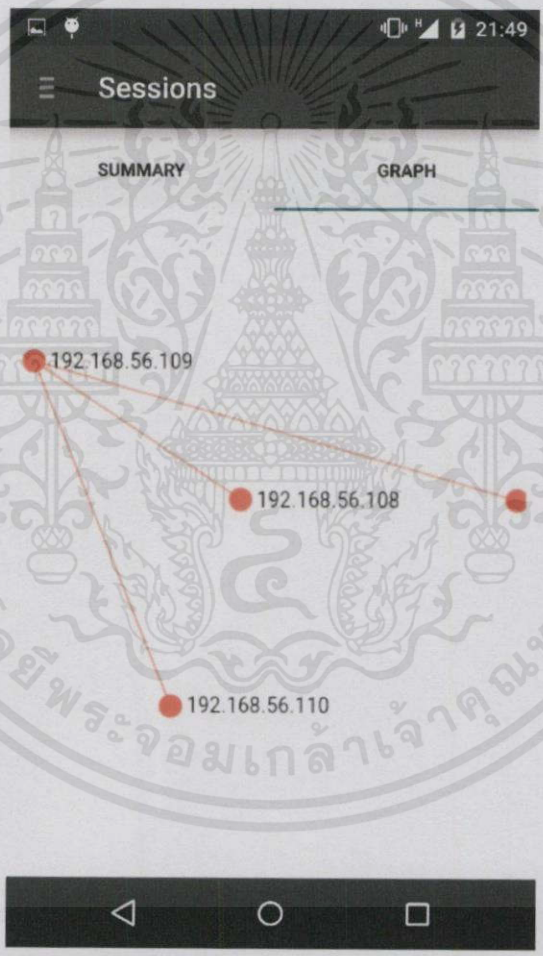
ได้ทำการเพิ่มเครื่องที่จะประเมินเป็น 4 เครื่อง โดยได้ใช้ XModule s_infer สำหรับการประเมินความสัมพันธ์ของแต่ละเครื่อง

4.5.3 ผลการดำเนินการ

ระยะเวลาที่ใช้ในการดำเนินการ: 476.78 วินาที

โดยเวลาที่เพิ่มขึ้นมาคือเวลาสำหรับการประเมินในแต่ละเครื่อง (ซึ่งเป็นเวลาที่ใกล้เคียงกับเวลาเดิมในการทดลองที่ 1) ทั้งหมด 4 เครื่อง และเวลาที่ใช้ในการประเมินความสัมพันธ์โดย s_infer (5 วินาที)

Attack Graph:



รูปที่ 30 ผลการทดลองที่ 3 แสดงโดย Mobile Client

จะเห็นได้ว่าการเชื่อมต่อของเครื่องที่ได้จาก s_infer นั้นยังคลาดเคลื่อนจากการเชื่อมต่อจริงพอสมควร เนื่องจากเป็นการประเมินความสัมพันธ์โดยดูจากบริการต่าง ๆ ที่ตรวจพบ เช่น มีความเป็นไปได้ว่าอยู่ในเครือข่ายเดียวกัน (192.168.56.0/24) และอาจมีความสัมพันธ์ในเชิงเว็บไซต์และฐานข้อมูล อย่างไรก็ตาม เพื่อให้การประเมินความสัมพันธ์แม่นยำมากขึ้นอาจจำเป็นต้องใช้การ

ปรับปรุง XModule ssh_agent สำหรับตรวจสอบการเชื่อมต่อต่าง ๆ ของเครื่องที่สามารถเข้าไปได้ เช่น การอ่านข้อมูลจากคำสั่ง netstat บนเครื่องที่เข้าไปได้อาจทำให้พบความสัมพันธ์เรื่องเครื่องนั้นกับเครื่องอื่น ๆ ในระบบ เช่น อาจมีการติดต่อกันผ่านพอร์ตใดพอร์ตหนึ่งกับเครื่องอื่นในระบบ เป็นต้น

บริการและช่องโหว่ที่ตรวจพบ:

เครื่อง LB

HTTPS (443) ช่องโหว่ปานกลาง: 1

SSH (22) ช่องโหว่ร้ายแรง: 1

เครื่อง W1 และเครื่อง W2

HTTP (80) ช่องโหว่ที่พบเหมือนการทดลองที่ 1 และ 2

SSH (22) ช่องโหว่ร้ายแรง: 1

เครื่อง DB

MySQL (3306) ช่องโหว่ร้ายแรง: 1

SSH (22) ช่องโหว่ร้ายแรง: 1

โดยช่องโหว่ที่พบของบริการ SSH และ MySQL ของแต่ละเครื่องในเครือข่ายนี้คือ การตั้งรหัสผ่านที่ไม่ดี (weak_password) ซึ่งเป็นช่องโหว่ร้ายแรง

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

ทางเราหวังว่าจะได้ช่วยให้การสร้างและพัฒนาซอฟต์แวร์และเครื่องมือเพื่อประเมินความปลอดภัยของระบบคอมพิวเตอร์บนกรอบการทำงาน (Framework) ที่เราสร้างขึ้นมาเป็นไปได้โดยง่าย โดยหวังไว้เป็นอย่างยิ่งว่าจะช่วยลดปัญหาต่าง ๆ ตามที่กำหนดไว้ในวัตถุประสงค์โครงการนี้ได้เป็นอย่างดี

อย่างไรก็ตามโครงงานนี้ยังสามารถพัฒนาการทำงานบางส่วนให้ดีมากยิ่งขึ้นไป เช่น การเพิ่มการประเมินความปลอดภัยให้ครอบคลุมมากยิ่งขึ้นโดยการนำเครื่องมือด้านการประเมินความปลอดภัยอื่น ๆ เข้ามาใช้งานเพิ่มเติม การเพิ่มตัวส่ง Email และ SMS เป็นต้น โดยได้ทำการสรุปไว้ในหัวข้อ *5.3 แนวทางการพัฒนาต่อ*

5.2 ปัญหาอุปสรรคและแนวทางการแก้ไข

1) การจำลองสภาพแวดล้อมเพื่อทดสอบการดำเนินการทำได้ค่อนข้างลำบาก เพราะต้องการทรัพยากรค่อนข้างสูง อาจแก้ปัญหาได้โดยการใช้งานระบบ Cloud ภายนอก โดยจำลองให้มีการดำเนินการต่าง ๆ เช่น การโจมตี เฉพาะภายในระบบนั้น
2) เครื่องมือด้านการประเมินความปลอดภัยต่าง ๆ จำเป็นต้องใช้เวลาศึกษาให้ละเอียดก่อนนำเข้ามาใช้ในซอฟต์แวร์นี้ อาจแก้ปัญหาได้โดยสร้างชุมชนสำหรับพัฒนาซอฟต์แวร์เพื่อให้ผู้อื่นสามารถมาร่วมพัฒนาซอฟต์แวร์นี้ได้
3) เครื่องมือด้านการประเมินความปลอดภัยต่าง ๆ มีจำนวนมากจึงต้องใช้เวลาในการพัฒนาเพื่อให้ครอบคลุมเครื่องมือเหล่านี้มากตามไปด้วย อาจแก้ปัญหาได้โดยสร้างชุมชนสำหรับพัฒนาซอฟต์แวร์ดังที่ได้กล่าวไปแล้วข้างต้น
4) เนื่องจาก Android Studio เป็น IDE ที่ค่อนข้างใหม่จึงทำให้อาจจะยังมีจุดที่ไม่สมบูรณ์อยู่บ้าง เช่น ปัญหาการนำเข้า Library จากภายนอก ปัญหาการนำเข้าไฟล์จากภายนอก ปัญหาเรื่อง Directory เป็นต้น

5.3 แนวทางในการพัฒนาต่อ

1) พัฒนาเพิ่มเติมในส่วน XService เช่น การเพิ่ม XLogger สำหรับบันทึกการดำเนินการต่าง ๆ และ XNotifier สำหรับการแจ้งเตือนผ่านช่องทางต่าง ๆ เช่น Email เป็นต้น
2) พัฒนา XModule ให้รองรับเครื่องมือภายนอกต่าง ๆ ให้มากขึ้น เพื่อให้ครอบคลุมการทำงานให้มากที่สุด
3) พัฒนา Web Client เพื่อเป็นอีกช่องทางหนึ่งในการดำเนินงาน
4) พัฒนา Mobile Client ให้สามารถรองรับการประเมินความปลอดภัยในระบบคอมพิวเตอร์ได้มากกว่า 1 ระบบต่อผู้ใช้
5) พัฒนา Mobile Client ให้มีรูปแบบการติดต่อกับผู้ใช้ที่เหมาะสมยิ่งขึ้น โดยสอบถามการทำงานปัจจุบันและการทำงานที่ผู้ใช้ต้องการเพิ่มเติม เพื่อนำกลับมาปรับปรุงต่อไป
6) พัฒนาสคริปต์การตั้งค่าให้มีความยืดหยุ่นและใช้งานง่ายมากยิ่งขึ้น เช่น รองรับการตั้งค่าสำหรับการทำงานแบบขนาน การตั้งค่าการส่งสถานะความปลอดภัยโดยรวมของระบบผ่าน Email หรือ SMS เป็นต้น
7) พัฒนาการติดตั้งให้เป็นไปได้ง่ายขึ้นผ่านการใช้งาน Docker ทำให้ผู้ใช้สามารถนำคอนเทนเนอร์ที่ได้ทำการติดตั้งและตั้งค่า ProjectX ไว้ล่วงหน้าไปใช้งานได้ทันทีโดยไม่จำเป็นต้องติดตั้งและตั้งค่าเอง
8) พัฒนาให้สามารถแสดงความสัมพันธ์ระหว่างเครื่องได้ดียิ่งขึ้น ผ่านการคลิกเส้นเชื่อม (Edge)

ภาคผนวก ก.

ส่วนติดต่อประสานงานแอปพลิเคชัน (API)

1. XAttackGraph REST API

1.1 การจัดการการดำเนินการ (Session)

Session Node

- INDEX: `GET /sessions/`
- SHOW: `GET /sessions/:session_id`
- CREATE: `POST /sessions/`
- UPDATE: `PUT /sessions/:session_id`
- DELETE: `DELETE /sessions/:session_id`

1.2 การจัดการโหนดโจมตี (Attack Node)

Attack Node

- INDEX: `GET /sessions/:session_id/nodes`
- SHOW: `GET /sessions/:session_id/nodes/:node_addr`
- CREATE: `POST /sessions/:session_id/nodes`
- UPDATE: `PUT/PATCH /sessions/:session_id/nodes/:node_addr`
- DELETE: `DELETE /sessions/:session_id/nodes/:node_addr`

1.3 การจัดการโหนดบริการ (Service Node)

Service Node

- INDEX: `GET /sessions/:session_id/nodes/:node_addr/services`
- SHOW: `GET /sessions/:session_id/nodes/:node_addr/services/:service_port_id`
- CREATE: `POST /sessions/:session_id/nodes/services`
- UPDATE: `PUT/PATCH`
  `/sessions/:session_id/nodes/:node_addr/services/:service_port_id`
- DELETE: `DELETE`
  `/sessions/:session_id/nodes/:node_addr/services/:service_port_id`

1.4 การจัดการโหนดช่องโหว่ (Vulnerability Node)

Vulnerability Node

- INDEX: `GET /sessions/:session_id/nodes/:node_addr/services/:service_port_id/vulns`
- SHOW: `GET /sessions/:session_id/nodes/:node_addr/services/:service_port_id/vulns/:vuln_id`
- CREATE: `POST /sessions/:session_id/nodes/services/:service_port_id/vulns`
- UPDATE: `PUT/PATCH /sessions/:session_id/nodes/:node_addr/services/:service_port_id/vulns/:vuln_id`
- DELETE: `DELETE /sessions/:session_id/nodes/:node_addr/services/:service_port_id/vulns/:vuln_id`

1.5 การจัดการความสัมพันธ์ระหว่างโหนด (Connections)

Connections

- SHOW: `GET /sessions/:session_id/nodes/:node_addr/connections/`
- CREATE: `POST /sessions/:session_id/nodes/:node_addr/connections/`

1.6 การจัดการคุณสมบัติของโหนด (Properties)

การสร้างหรือการเปลี่ยนแปลงโหนดสามารถกำหนดค่าคุณสมบัติต่าง ๆ ของโหนดได้ผ่าน HTTP Body (application/x-www-form-urlencoded) ของฟิลด์ properties

ตัวอย่างการสร้างโหนดโดยเป็นการสร้างโหนดโจมตี เช่น

```
POST /sessions/1234/nodes
properties[addr]=192.168.56.102
properties[addrtype]=ipv4
properties[state]=up
```

โดย addr หมายถึง หมายเลขไอพีแอดเดรสของเครื่องที่ถูกประเมินความปลอดภัย

addrtype หมายถึง ประเภทหมายเลขไอพีแอดเดรส โดยอาจมีค่าเป็น ipv4 หรือ ipv6

state หมายถึง สถานะการเชื่อมต่อจาก ProjectX โดยอาจมีค่าเป็น up หรือ down

การดูข้อมูลของโหนดต่าง ๆ จะได้รับข้อมูลกลับมาในรูปแบบ JSON Object หรือ JSON Array [13]

ตัวอย่างการรับข้อมูลโหนดโดยเป็นการรับข้อมูลโหนดบริการ เช่น

```
GET /sessions/1234/nodes/192.168.56.102/services

200 OK
[ { service_name: http, port_id: 80 }, { service_name: ssh, port_id: 22 } ]
```

โดย service_name หมายถึง ชื่อของบริการ

port_id หมายถึง หมายเลขพอร์ตที่เปิดใช้สำหรับบริการนั้น

2. Ruby SDK

เป็น Library สำหรับภาษา Ruby ที่ทางเราพัฒนาขึ้นมาเพื่อให้สามารถติดต่อกับ XAttackGraph API ได้โดยง่าย เพื่อนำไปใช้สำหรับการพัฒนา XModule ต่อไป

2.1 การจัดการการดำเนินการ (Session)

```
session_id = AttackGraph.create_session
AttackGraph.with_session(session_id) do
  attack_nodes = AttackGraph::AttackNode.all
  # other good stuff here
end
```

2.2 การจัดการโหนดโจมตี (Attack Node)

Basic (ActiveNode::Base)

```
AttackGraph::AttackNode.new(...)
AttackGraph::AttackNode.create(...)
AttackGraph::AttackNode.all
AttackGraph::AttackNode.find(...)
AttackGraph::AttackNode.where(...)

attack_node.save
attack_node.persisted?
attack_node.update_attributes(...)
attack_node.destroy
```

Association (ActiveNode::Association)

```
attack_node.services
attack_node.vulnerabilities          # equivalent to attack_node.services.map(&:vu
attack_node.services << service_node
attack_node.services.build
attack_node.services.create
attack_node.services.clear
attack_node.services.empty?
attack_node.services.count
attack_node.services.find(...)
attack_node.services.where(...)
```

2.3 การจัดการโหนดบริการ (Service Node)

Basic (ActiveNode::Base)

```
service_node.save
service_node.persisted?
service_node.update_attributes(...)
service_node.destroy
```

Association (ActiveNode::Association)

```
service_node.vulnerabilities
service_node.vulnerabilities << vuln_node
service_node.vulnerabilities.build
service_node.vulnerabilities.create
service_node.vulnerabilities.clear
service_node.vulnerabilities.empty?
service_node.vulnerabilities.count
service_node.vulnerabilities.find(...)
service_node.vulnerabilities.where(...)
```

2.4 การจัดการโหนดช่องโหว่ (Vulnerability Node)

Basic (ActiveNode::Base)

```
vuln_node.save
vuln_node.update_attributes(...)
vuln_node.destroy
```

2.5 การจัดการความสัมพันธ์ระหว่างโหนด (Connections)

Connections

```
attack_node.connect_to({ addr: node_to_connect, port_id: port_to_connect })
attack_node.connections
```

2.6 การจัดการคุณสมบัติของโหนด (Properties)

สามารถเรียกชื่อคุณสมบัติของโหนดนั้นและกำหนดค่าได้ทันที ตัวอย่างเช่น หากต้องการแก้ไขสถานะการเปิดใช้งานของโหนดโจมตี attack_node สามารถทำได้ ดังนี้

Node Properties

```
attack_node.status = 'down'
attack_node.save

# or

attack_node.update_attributes(status: 'down')
```

3. NodeJS SDK

โดยทั่วไปมีลักษณะเช่นเดียวกับ Ruby SDK แต่การทำงานของฟังก์ชันต่าง ๆ จะเป็นไปในลักษณะ Non-Blocking I/O

ตัวอย่างเช่นการสร้างโหนดโจมตี

```
attackGraph.attackNode.create({ addr: '192.168.99.102', addrtype: 'ipv4' }, function (aNode) {
    console.log("Hi, from", aNode.addr);
    attackGraph.attackNode.count(function (err, count) {
        console.log("Now, has attack nodes:", count);
    });
});
```

บรรณานุกรม

[1] Fowler, Martin. 2014. **Microservices.** Available: **http://martinfowler.com/articles/microservices.html**.

[2] Jeannette M. Wing, Oleg Sheyner, Joshua Haines, Somesh Jha, and Richard Lippmann. **Automated Generation and Analysis of Attack Graphs.** IEEE Symposium on Security and Privacy. 2002.

[3] Skoudis, E. and Liston, T. 2006. **Counter Hack Reloaded.** 2nd ed. : Prentice Hall.

[4] Neo4j. **"The World's Leading Graph Database."** [Online]. Available: http://neo4j.com/

[5] Demarzi, Max. **"A thin Ruby wrapper to the Neo4j Rest API."** [Online]. Available: https://github.com/maxdemarzi/neography

[6] Sinatra. [Online]. Available: http://www.sinatrarb.com/

[7] Redis. **"Advanced key-value cache and store."** [Online]. Available: http://redis.io/

[8] Sidekiq. **"Simple, efficient background processing for Ruby."** [Online]. Available: http://sidekiq.org/

[9] Hohpe, Gregor. 2004. **Starbucks Does Not Use Two-Phase Commit.** [Online]. Available: http://www.eaipatterns.com/ramblings/18_starbucks.html

[10] Russ Olsen. 2011. **Eloquent Ruby**: Addison-Wesley.

[11] Sigma.js. **"JavaScript library dedicated to graph drawing."** [Online]. Available: http://sigmajs.org/

[12] Rapid 7. "**Metasploitable 2 Exploitability Guide.**" [Online]. Available: https://community.rapid7.com/docs/DOC-1875

[13] JSON. "**Introducing JSON.**" [Online]. Available: http://json.org/