การตรวจจับโค้ดที่เป็นอันตรายบนระบบปฏิบัติการแอนดรอยด์ ด้วยการวิเคราะห์สายอักษร

Malicious Code Detection on Android Operating Systems by using Byte-Code Analysis

วรวัฒน์ เชิญสวัสดิ์ และ โกมล นารัง*

คณะวิทยาศาสตร์และเทคโนโลยี สาขาวิชาเทคโนโลยีสารสนเทศและการจัดการมหาวิทยาลัยกรุงเทพ

ถนนพระรามสี่ เขตคลองเตย กรุงเทพมหานคร 10110

Worawat Choensawat and Komal Narang*

School of Science and Technology, Information Technology and Management, Bangkok University

Rama 4 Road, Klong-Toey Bangkok 10110,

ABSTRACT – This research presents a model for malware detection on mobile operating system based on machine learning technique. The objective is to reduce the risk of installing harmful application when the user did not update the anti-virus program in time. The proposed model is different to other anti-virus is that most of anti-virus software used virus signature to identify malware. However, the virus signature-based detection approach requires frequent updates of the virus signature dictionary. The signature-based approaches are not effective against new, unknown viruses while the proposed model based on machine learning can detect new malware even some parts of the code have been modified. The research processes are as follows: (1) achieving of both malicious and benign codes on android operating system, (2) Extracting features based on the distribution of n-grams frequency, and (3) constructing a model for classification the malicious codes using the extracted features for both malicious and benign codes. In the experiment, 500 malicious codes, 400 benign codes and 100 system files were used to construct the model. The experiment shows that the model achieved more than 88.9% accuracy. For the sensitivity and specificity, the model achieved 95.0% and 82.8%, respectively.

KEY WORDS - Mobile Operating System; Antivirus; Mobile Application; Machine Learning Technique; Term Frequency (TF); Principal Component Analysis (PCA)

บทคัดย่อ – งานวิจัยนี้ได้นำเสนอการใช้แบบจำลองการตรวจจับโค้ดอันตรายบนระบบปฏิบัติการของมือถือ โดยใช้ทฤษฎีการเรียนรู้ของเครื่อง งานวิจัยนี้มีเป้าหมายเพื่อลดความเสี่ยงต่อการติดตั้งโปรแกรมไม่พึงประสงค์ต่างๆ ของผู้ใช้ที่ไม่ได้อัพเดตโปรแกรมแอนติไวรัสทันเวลา ซึ่งแบบจำลองที่ได้นำเสนอนี้ต่างจากโปรแกรมแอนติไวรัสทั่วไปตรงที่ โปรแกรมแอนติไวรัสทั่วไปมักนิยมใช้หลักการของการตรวจจับรูปแบบสายอักษรที่เฉพาะเจาะจงที่ฝังในโปรแกรมเพื่อระบุว่าโปรแกรมนั้นอันตรายหรือไม่ แต่หลักการของการตรวจจับรูปแบบสายอักษรที่เฉพาะเจาะจงนั้น ทุกๆครั้งที่มีไวรัสหรือโค้ดอันตรายตัวใหม่ขึ้นมา ผู้ใช้

จำเป็นต้องอัพเดตโปรแกรมแอนติไวรัสเสมอ ซึ่งเมื่อไรก็ตามที่ผู้ใช้พลาดการอัพเดตให้ทันเวลาก็จะตกเป็นเป้าของไวรัสตัวใหม่ ในขณะที่แบบจำลองที่นำเสนอได้ใช้เทคนิคการเรียนรู้ของเครื่องในการรู้จำกลุ่มของโค้ดอันตราย ทำให้ยังคงตรวจจับโค้ดอันตรายได้ถึงแม้ว่าโค้ดอันตรายตัวใหม่ได้มีรูปแบบแตกต่างไปบ้างก็ตาม ขั้นตอนในการดำเนินการวิจัยเริ่มจาก (1) รวบรวมแอปพลิเคชันบนระบบปฏิบัติการแอนดรอยด์ทั้งแอปพลิเคชันปกติและแอปพลิเคชันที่เป็นอันตราย (2) สกัดคุณลักษณะโดยวิเคราะห์การกระจายตัวของความถี่ของไบต์เอ็นแกรม และค่าความถี่ของเทอม และ (3) สร้างโมเดลของการจำแนกโค้ดอันตรายจากใช้คุณลักษณะที่ได้ทั้งในส่วนของแอปพลิเคชันปกติและแอปพลิเคชันที่เป็นอันตราย ในการทดลองได้ใช้โค้ดอันตรายจำนวน 500 ไฟล์ และโค้ดปกติจำนวน 500 ซึ่งประกอบด้วยไฟล์จากแอปพลิเคชันปกติ 400 ไฟล์ และไฟล์ของระบบจำนวน 100 ไฟล์ สำหรับการสร้างโมเดลของการจำแนกโค้ดอันตราย และสำหรับการทนต่อโค้ดอันตรายใหม่ที่ไม่ได้อยู่ในฐานข้อมูล พบว่าโมเดลที่ได้นำเสนอนั้นมีร้อยละความแม่นยำมากกว่า 88.9 ในขณะที่ร้อยละค่าความไวและร้อยละความจำเพาะมีค่าเท่ากับ 95.0 และ 82.8 ตามลำดับ

คำสำคัญ - ระบบปฏิบัติการมือถือ; แอนติไวรัส; แอปพลิเคชันบนมือถือ; เทคนิคการเรียนรู้ของเครื่อง; ความถี่ของไบต์เอ็นแกรม; การวิเคราะห์องค์ประกอบ

1. บทนำ

ในปี พ.ศ. 2557 พบว่าผู้ลงทะเบียนเปิดใช้เบอร์โทรศัพท์เคลื่อนที่ในประเทศไทยมีจำนวนถึง 81.68 ล้านราย [1] คิดเป็นร้อยละ 120.83 ต่อประชากร ประเทศไทยเป็น 1 ใน 8 ของโลกที่มีจำนวนโทรศัพท์เคลื่อนที่สูงกว่าจำนวนประชากร แสดงให้เห็นว่าการใช้โทรศัพท์เคลื่อนที่ในประเทศไทยเป็นที่นิยมอย่างมาก ด้วยปัจจัยนี้ธนาคารต่างๆ จึงพัฒนาแอปพลิเคชันสำหรับทำธุรกรรมทางโทรศัพท์เคลื่อนที่ (Mobile banking) เพื่ออำนวยความสะดวกให้ลูกค้าสามารถชำระเงินทางอิเล็กทรอนิกส์ (e-Payment) ซึ่งอัตราการเติบโตของการทำธุรกรรมเป็นแบบเลขชี้กำลังเริ่มตั้งแต่ปี พ.ศ. 2552 และรวมมูลค่าการทำธุรกรรมเป็นจำนวนเงิน 439,960 ล้านบาท ในปี พ.ศ.2557 [2] สถิติเหล่านี้ชี้ให้เห็นว่า ผู้ใช้จำเป็นต้องมีโปรแกรมป้องกันการโจรกรรมข้อมูลส่วนตัวบนโทรศัพท์เคลื่อน

จากการศึกษาและสำรวจพบว่าร้อยละ 97 ของมัลแวร์บนอุปกรณ์เคลื่อนที่ (Mobile malware) อยู่บนระบบปฏิบัติการแอนดรอยด์ (ข้อมูลจาก Mobile Threat Report ปี ค.ศ.2015) และร้อยละ 70 ของมัลแวร์บนโทรศัพท์เคลื่อนที่มุ่งขโมยข้อมูลส่วนตัว [2] เช่น ข้อมูลเกี่ยวกับบัตรเครดิตและบัญชีธนาคาร ล็อคอินและพาสเวิร์ด เป็นต้น แอปพลิเคชันที่มีโค้ดอันตรายฝังตัวอยู่มีจำนวนเพิ่มขึ้นอย่างรวดเร็ว ในที่นี่เรียกสั้นๆ ว่า "แอปพลิเคชันอันตราย" ซึ่งแอปพลิเคชันอันตรายเหล่านี้มีการพัฒนาตัวเองด้วยเทคนิคต่างๆ เพื่อให้สามารถหลีกเลี่ยงการตรวจจับโค้ดอันตรายด้วยโปรแกรมแอนติไวรัส (Anti-Virus) ที่มีอยู่ในท้องตลาดได้และเข้าไปฝังตัวในอุปกรณ์เคลื่อนที่ของผู้ใช้

การวิเคราะห์และแยกแยะแอปพลิเคชันอันตรายเป็นปัญหาที่ยากเพราะภาษาที่ใช้เขียนแอปพลิเคชันบนแอนดรอยด์เป็นภาษาจาวา ซึ่งภาษาจาวามีไลบารี่ที่สนับสนุนการเขียนแอปพลิเคชันจำนวนมากและไลบารี่เหล่านี้สามารถช่วยนักเขียนโปรแกรมให้เขียนแอปพลิเค-ชันที่ซับซ้อนได้ไม่ยาก ด้วยจำนวนไลบารี่ของจาวามีมากมายซึ่งนิยมใช้ทั้งในแอปพลิเคชันปกติและแอปพลิเคชันอันตราย

การป้องกันโปรแกรมไม่พึงประสงค์รวมถึงไวรัสและมัลแวร์ต่างๆ ผู้ใช้มักนิยมใช้โปรแกรมแอนติไวรัส ซึ่งใช้หลักการของการตรวจจับรูปแบบสายอักษรที่เฉพาะเจาะจง (Signature) ที่ฝังในโปรแกรมเพื่อระบุว่าโปรแกรมนั้นอันตรายหรือไม่ ทุกครั้งที่เกิดไวรัสตัวใหม่ขึ้นมาผู้ใช้จำเป็นต้องอัพเดตโปรแกรมแอนติไวรัสเสมอ เมื่อไรก็ตามที่ผู้ใช้พลาดการอัพเดตให้ทันเวลาก็จะตกเป็นเป้าของไวรัสตัวใหม่เสมอ

งานวิจัยนี้ได้นำเสนอการวิเคราะห์สายอักษรจากไบนารีโค้ดของโปรแกรมบนเครื่องมือถือ และสร้างแบบจำลองการตรวจจับโค้ดอันตรายบนระบบปฏิบัติการของมือถือ ด้วยทฤษฎีการเรียนรู้ของเครื่อง (Machine learning) โดยมีเป้าหมายเพื่อลดความเสี่ยงต่อการติดตั้งโปรแกรมที่ไม่พึงประสงค์ต่างๆ ของผู้ใช้ที่ไม่ได้อัพเดตโปรแกรมแอนติไวรัสทันเวลา

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การวิจัยนี้เป็นการออกแบบและพัฒนาโปรแกรมประยุกต์บนระบบปฏิบัติการแอนดรอยด์ เนื่องด้วยจำนวนผู้ใช้ระบบปฏิบัติการแอนดรอยด์มีมากกว่าหนึ่งพันล้านคนในปลายปี พ.ศ. 2557 [3] ด้วยยอดผู้ใช้จำนวนมากเช่นนี้ทำให้ระบบปฏิบัติการแอนดรอยด์เป็นที่ดึงดูดนักเขียนโค้ดอันตราย

2.1 ระบบปฏิบัติการแอนดรอยด์

ไฟล์แอปพลิเคชันบนแอนดรอยด์ หรือ แพ็คเก็ตแอปพลิเคชันแอนดรอยด์ (Android application package) เรียกสั้นๆ ว่า เอพีเค (APK) ซึ่งเป็นรูปแบบของไฟล์บีบอัด (Compression file) คล้ายคลึงกับไฟล์จาร์ (.jar) แพ็คเก็ตเอพีเคเป็นรูปแบบในการส่งต่อ เผยแพร่ และสามารถติดตั้งบนอุปกรณ์ที่ใช้ระบบปฏิบัติการแอนดรอยด์ผ่านเพลย์สโตว์ของกูเกิ้ล (Google Play Store) แพ็คเก็ตเอพีเคมีองค์ประกอบหรือประกอบด้วยไฟล์และไดเร็กทอรีสำคัญดังนี้

1) ไฟล์เอ็กเอ็มแอล (XML) ชื่อว่า "AndroidManifest.xml"
2) ไฟล์เด็กซ์ ชื่อว่า "Classes.dex"
3) ไดเร็กทอรีเรส ชื่อว่า "Res directory"

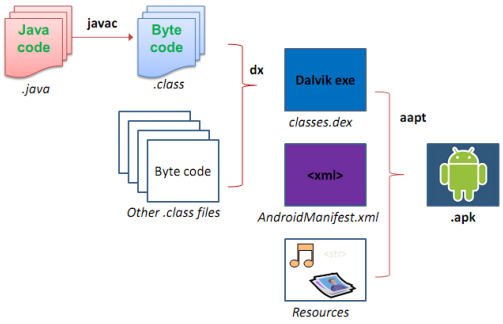
ไฟล์เอ็กเอ็มแอลเป็นไฟล์ที่บ่งบอกข้อมูล ชื่อไฟล์เวอร์ชัน และสิทธิของการเข้าถึงทรัพยากรต่างๆ (Access permission) และข้อมูลหรือกิจกรรมต่างๆ ของแอปพลิเคชัน

ไฟล์ classes.dex เป็นโปรแกรมหลักที่สำหรับของแอปพลิเคชันบนระบบปฏิบัติการแอนดรอยด์ ไฟล์ของโปรแกรมเป็นไบต์โค้ด (Bytecode) ของแอนดรอยด์ ขั้นตอนการแปลงจึงประกอบด้วย 2 ขั้นตอน ก) ใช้คอมไพเลอร์จาวา ชื่อว่า "javac" แปลงจากคลาสจาวาเป็นไบต์โค้ดที่สามารถรันบนเครื่องเสมือนจาวา (Java Virtual Machine) ข) ใช้คอมไพเลอร์ดีเอ็กซ์แอนดรอยด์" (Android DX compiler) แปลงไบต์โค้ดจาวาไปเป็นไฟล์ดัลวิก (Dalvik) ซึ่งมีนามสกุล ".dex" ขั้นตอนที่กล่าวมาแสดงในภาพที่ 1

ไดเร็กทอรี “Res” เป็นโฟลเดอร์ที่เก็บไฟล์ทรัพยากรเสริมต่างๆ ของแอปพลิเคชัน เช่น ไฟล์รูปภาพ วีดีโอ เสียง และไฟล์ข้อมูล เป็นต้น นอกจากนี้ในไดเร็กทอรีนี้ยังสามารถบรรจุไฟล์ที่เกี่ยวข้อง เช่น ไลบารีภายนอก เป็นต้น

สำหรับการวิจัยนี้ คณะผู้วิจัยได้จัดประเภทของแพ็คเก็ตเอพีเค โดยใช้หลักของความมั่นคงปลอดภัยและสามารถแบ่งได้เป็น 3 ประเภทดังนี้

1. ไฟล์ระบบ
2. ไฟล์ปกติ
3. ไฟล์อันตราย



ภาพที่ 1 ขั้นตอนการคอมไพล์ไฟล์แอปพลิเคชันบนแอนดรอยด์

ไฟล์ระบบ คือไฟล์ที่เป็นส่วนหนึ่งของระบบปฏิบัติการแอนดรอยด์ ซึ่งอาจจะเป็นไฟล์เฉพาะของแต่ละยี่ห้อของอุปกรณ์เคลื่อนที่ แอปพลิเคชันที่มาพร้อมกับระบบปฏิบัติการ เช่น ปฏิทิน กล้องถ่ายรูป บราวเซอร์ สมุดบันทึก สมุดรายชื่อ และอื่นๆ

ไฟล์ปกติ คือแอปพลิเคชันใดๆ ก็ตามที่เชื่อถือได้และปลอดภัยสำหรับอุปกรณ์แอนดรอยด์แล้ว แอปพลิเคชันนั้นจะจัดอยู่ในไฟล์ปกติ หรือแอปพลิเคชันปกติ ดังนั้นไฟล์ระบบทั้งหมดจัดอยู่ในไฟล์ปกติโดยปริยายเพราะไฟล์ระบบได้รับการติดตั้งมากับอุปกรณ์ตั้งแต่โรงงานผลิต แอปพลิเคชันปกติจะทำงานบนอุปกรณ์ตามที่ได้ระบุไว้ หรือตามที่ผู้ใช้เข้าใจ และเข้าถึงทรัพยากรระบบและข้อมูลต่างๆ ของผู้ใช้หลังจากที่ได้รับอนุญาตจากผู้ใช้แล้วเท่านั้น แอปพลิเคชันปกติสามารถดาวโหลดได้ผ่านเพลย์สโตว์ ตัวอย่างของแอปพลิเคชันปกติ เช่น ออฟฟิคโปลาริส (Polaris office) กูเกิ้ลพัส (Google plus) ชุดออฟฟิค (Office suite) อร์อะโดบี (Adobe reader) เป็นต้น

ไฟล์อันตราย คือแอปพลิเคชันที่เป็นอันตรายบนอุปกรณ์แอนดรอยด์ ตัวอย่างการทำงานของไฟล์อันตรายที่ว่านี้คือ

- การขโมยข้อมูล ข้อมูลความลับ ข้อมูลส่วนบุคคลของผู้ใช้งาน เป็นต้น [4]

- การขโมยข้อมูลที่เป็นการแสดงตัวตน (User profile) ที่เก็บไว้บนอุปกรณ์แอนดรอยด์และส่งต่อข้อมูลเหล่านี้โดยไม่ได้รับอนุญาตจากผู้ใช้

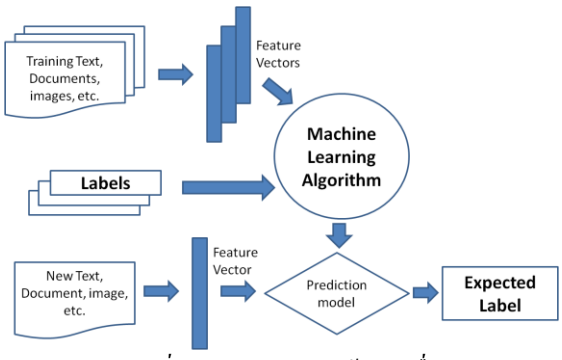
- ส่ง/ลบ SMS โดยไม่ได้รับอนุญาต

- การเรียกค่าไถ่ เป็นรูปแบบใหม่ที่เพิ่งค้นพบ แอปพลิเคชันนี้มีรูปแบบคือ การบีบอัดและเข้ารหัสข้อมูลของผู้ใช้ จากนั้นจะส่งข้อความเรียกค่าไถ่ ถ้าผู้ใช้ต้องการถอดรหัสเพื่อเข้าถึงข้อมูลของตนเองจะต้องจ่ายเงินค่าไถ่

2.2 การเรียนรู้ของเครื่อง

การเรียนรู้ของเครื่องเป็นสาขาหนึ่งของปัญญาประดิษฐ์ที่เกี่ยวข้องกับการพัฒนาเทคนิควิธี เพื่อให้คอมพิวเตอร์สามารถเรียนรู้ โดยเน้นที่วิธีการเพื่อสร้างโปรแกรมคอมพิวเตอร์จากการวิเคราะห์ชุดข้อมูล การเรียนรู้ของเครื่องจึงเกี่ยวข้องอย่างมากกับสถิติศาสตร์

เทคนิคการเรียนรู้ของเครื่องถูกใช้เพื่อเพิ่มประสิทธิภาพในการแก้ปัญหาด้านต่าง ๆ เช่น การสร้างให้คอมพิวเตอร์สามารถแยกแยะวัตถุ เสียง หรือตัวอักษรได้ หรือจำแนกข้อมูลจำนวนมากที่ไม่สามารถทำได้โดยมนุษย์ ลักษณะทั่วไปของการเรียนรู้ของเครื่องจักรเป็นการสร้างอัลกอริธึมหรือโปรแกรมคอมพิวเตอร์ จากการให้ข้อมูลฝึก (Training data) สำหรับสอนให้คอมพิวเตอร์เรียนรู้ เพื่อให้ได้มาซึ่งแบบจำลองในการแยกแยะวัตถุอื่นได้ ลักษณะการเรียนรู้ของเครื่องจักรแสดงได้ดังภาพที่ 2 เทคนิคการเรียนรู้ของเครื่อง



ภาพที่ 2 แนวคิดการเรียนรู้ของเครื่อง

เทคนิคเอสวีเอ็ม (Support Vector Machine: SVM) เป็นเทคนิคที่นิยมใช้ในการจำแนกข้อมูลที่มีหลายมิติ (high-dimensional data) ตัวอย่างเช่น การจำแนกหัวข้อข่าวที่ส่งผ่าน Twitter ว่าแต่ละหัวข้อข่าวอยู่ในหมวดหมู่ใดใน 12 หมวด [5] ฟีเจอร์ที่ใช้เป็นคำที่อยู่ในข้อความ โดยอินพุตเวกเตอร์เป็นความถี่ของคำที่มีความหมายและเป็นคำที่นิยมใช้ในระดับกลาง หลังจากนั้น ได้นำเอสวีเอ็มมาใช้ในการจำแนกหมวดหมู่ข่าวจากความถี่ของคำในข้อความที่ส่งผ่าน Twitter ความถูกต้องของการจำแนกได้สูงกว่าร้อยละ 75 สำหรับ 11 หมวดหมู่ และต่ำกว่าร้อยละ 70 หนึ่งหมวดหมู่ อีกตัวอย่างที่ได้ใช้เทคนิคเอสวีเอ็มคือการจำแนกผู้ป่วยเบาหวานจากประวัติการรักษา ซึ่งได้ค่าความถูกต้องสูงกว่าร้อยละ 85 [6]

หลายปีมานี้ได้มีงานวิจัยจำนวนมากที่มุ่งเน้นการตรวจจับโค้ดอันตรายที่ไม่รู้จักในบนเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer: PC) จากไบนารีโค้ด โดยริเริ่มใช้ทฤษฎีทางปัญญาประดิษฐ์ การเรียนรู้ของเครื่อง ผู้ริเริ่มแนวคิดนี้ได้แก่ Moskovitch et al. [7] งานวิจัยนี้ได้วิเคราะห์โค้ดอันตรายจากลำดับของไบต์โค้ด ในขณะที่ Abou-Assaleh et al. [8] ได้นำเสนอการใช้ Common N-Gram (CNG) เพื่อตรวจจับโค้ดอันตราย วิธีการนี้เริ่มต้นด้วยการสร้างโปรไฟล์ของ Executable file ทั้งที่เป็นไฟล์อันตรายและปลอดภัย จากนั้นเมื่อมีไฟล์ใหม่เข้ามาจะทำการวิเคราะห์ว่าไฟล์ดั้งกล่าวคล้ายคลึงกับโปรไฟล์ของไฟล์ใดมากที่สุด Kotler และ Maloof [9] ได้รวบรวมไฟล์ปลอดภัยจำนวน 1,971 ไฟล์ และไฟล์อันตรายจำนวน 1,651ไฟล์ (บนเครื่องพีซี) และได้ใช้ n-grams ของไบต์โค้ดสำหรับการสร้างแบบจำลองในการจำแนกไฟล์อันตราย จากการทดลองของงานดังกล่าวพบว่าได้อัตราผลบวกจริง (true-positive rate) มากถึง 0.98

เห็นได้ว่างานวิจัยที่ได้กล่าวมาได้มุ่งเน้นการสร้างแบบจำลองสำหรับจำแนกไฟล์อันตรายบนเครื่องคอมพิวเตอร์ส่วนบุคคล จึงแตกต่างกับงานวิจัยนี้ที่เน้นไปที่ระบบปฏิบัติการแอนดรอยด์ ความท้าทายของการตรวจจับไฟล์อันตรายบนแอนดรอยด์ คือข้อจำกัดของทรัพยากร เช่น ขนาดของหน่วยความจำ ความเร็วในการประมวล และสิทธิ์ในการเข้าถึงข้อมูลที่จำเป็นต้องใช้ในการสร้างแบบจำลอง เป็นต้น

3. วิธีดำเนินงานวิจัย

ขั้นตอนการพัฒนาอัลกอริทึมสำหรับการตรวจจับโค้ดที่เป็นอันตรายบนระบบปฏิบัติการมือถือ ทางผู้วิจัยได้ตัวอย่างโค้ดที่เป็นอันตรายบนระบบปฏิบัติการแอนดรอยด์จำนวน 500 ตัวอย่างจาก Min Zheng ซึ่งเป็นผู้เขียนบทความ DroidAnalytics [10] หลังจากที่ได้แอปพลิเคชันต่างๆแล้ว ขั้นตอนการพัฒนาสามารถแบ่งออกเป็นสองขั้นตอนได้แก่

1. การสกัดลักษณะเฉพาะ (Feature extraction) โดยการวิเคราะห์รูปแบบสายอักษรของโค้ดปกติ และโค้ดอันตราย โดยพิจารณาจากการกระจายตัวของความถี่ของไบต์เอ็นแกรม (Byte n-grams)

2. สร้างโมเดลการจำแนกโค้ดอันตราย แนวคิดการสกัดลักษณะเฉพาะของโค้ดอันตราย

3.1 การสกัดลักษณะเฉพาะ

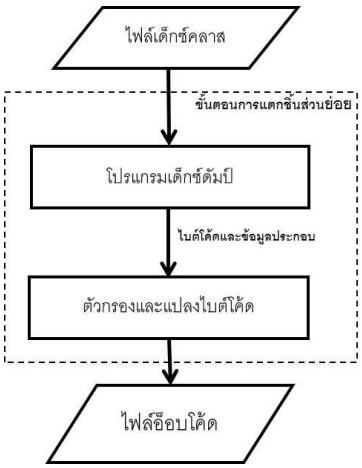
ตามที่ได้กล่าวข้างต้นในข้อ 2.1 ไฟล์ Classes.dex เป็นเอ็กซ์ซีคิวเทเบิลโค้ด (Executable code) ที่ใช้รันบนแอนดรอยด์ คณะผู้วิจัยใช้เครื่องมือที่ชื่อว่า "เด็กซ์ดัมป์" (Dexdump) ซึ่งเป็นเครื่องมือหนึ่งที่มากับ Android Development Tool: ADT) ในการแปลงไฟล์เด็กซ์คลาสให้อยู่ในรูปแบบที่มนุษย์อ่านได้ (Human-readable format) ไฟล์ผลลัพธ์ที่ได้จากการใช้เด็กซ์ดัมป์ประกอบด้วย ไฟล์เมต้าข้อมูล (Meta information) และไบต์โค้ดดัลวิก ตัวอย่างไฟล์ผลลัพธ์ที่ได้ดังแสดงในภาพที่ 3

```
01e2d4: 2201 0d00                       |0000: new-instance v1, Landroid/app/Notification$Builde:
01e2d8: 7020 4300 5100                  |0002: invoke-direct {v1, v5}, Landroid/app/Notification:
01e2de: 5362 0d00                       |0005: iget-wide v2, v6, Landroid/app/Notification;.when
01e2e2: 6e30 5700 2103                  |0007: invoke-virtual {v1, v2, v3}, Landroid/app/Notifica
01e2e8: 0c01                            |000a: move-result-object v1
01e2ea: 5262 0500                       |000b: iget v2, v6, Landroid/app/Notification;.icon:I //
01e2ee: 5263 0600                       |000d: iget v3, v6, Landroid/app/Notification;.iconLevel
01e2f2: 6e30 5300 2103                  |000f: invoke-virtual {v1, v2, v3}, Landroid/app/Notifica
01e2f8: 0c01                            |0012: move-result-object v1
01e2fa: 5462 0100                       |0013: iget-object v2, v6, Landroid/app/Notification;.co
01e2fe: 6e20 4600 2100                  |0015: invoke-virtual {v1, v2}, Landroid/app/Notificatio
01e304: 0c01                            |0018: move-result-object v1
01e306: 5462 0b00                       |0019: iget-object v2, v6, Landroid/app/Notification;.ti
01e30a: 6e30 5500 210a                  |001b: invoke-virtual {v1, v2, v10}, Landroid/app/Notifi
01e310: 0c01                            |001e: move-result-object v1
01e312: 5462 0a00                       |001f: iget-object v2, v6, Landroid/app/Notification;.so
01e316: 5263 0000                       |0021: iget v3, v6, Landroid/app/Notification;.audioStre
01e31a: 6e30 5400 2103                  |0023: invoke-virtual {v1, v2, v3}, Landroid/app/Notifica
01e320: 0c01                            |0026: move-result-object v1
01e322: 5462 0c00                       |0027: iget-object v2, v6, Landroid/app/Notification;.vi
01e326: 6e20 5600 2100                  |0029: invoke-virtual {v1, v2}, Landroid/app/Notificatio
01e32c: 0c01                            |002c: move-result-object v1
01e32e: 5262 0700                       |002d: iget v2, v6, Landroid/app/Notification;.ledARGB:I
01e332: 5263 0900                       |002f: iget v3, v6, Landroid/app/Notification;.ledOnMS:I
```

ภาพที่ 3 ตัวอย่างไฟล์ที่ได้จากโปรแกรมเด็กซ์ดัมป์ (Dexdump)

3.1.1 การสกัดอ็อบโค้ดจากไฟล์เด็กซ์คลาส

ไฟล์ผลลัพธ์ดังภาพที่ 3 ได้ยังไม่ได้อยู่ในรูปแบบที่พร้อมใช้งาน ขั้นตอนต่อไปคือการกรองข้อมูลในไฟล์ให้เหลือเฉพาะไบต์โค้ดและแปลงไบต์โค้ดนี้ไปเป็นอ็อบโค้ด (Operation code: OpCode) โดยมีไฟล์นำเข้าคือ ไฟล์ classes.dex และได้ไฟล์ผลลัพธ์คือ ไฟล์อ็อบโค้ด ภาพที่ 4 แสดงกระบวนการสกัดอ็อบโค้ดจากไฟล์เด็กซ์โดยมีขั้นตอนการทำงานดังนี้ 1) รับไฟล์เด็กซ์เข้ามา 2) เรียกโปรแกรมเด็กซ์ดัมป์เพื่อแปลงไฟล์เด็กซ์เป็นไฟล์ที่มีเมต้าข้อมูลและไบต์โค้ด และ 3) กรองเฉพาะไบต์โค้ดและแปลงเป็นอ็อบโค้ด ตัวอย่างผลลัพธ์เป็นไฟล์อ็อบโค้ดดังแสดงในภาพที่ 5



ภาพที่ 4 กระบวนการสกัดอ็อบโค้ดจากไฟล์เด็กซ์คลาส

```
0x54  0x1a  0x71  0x70  0x28  0x15  0x70  0x0e
0x71  0x0a  0x71  0x11  0x71  0x71  0x0a  0x38
0x12  0x11  0x71  0x28  0x12  0x3b  0x11  0x71
0x28  0x0d  0x1a  0x22  0x70  0x1a  0x6e  0x6e
0x1a  0x6e  0x6e  0x71  0x28  0x15  0x6e  0x1f
0x14  0x6e  0x1f  0x1a  0x71  0x1a  0x71  0x6e
0x6e  0x0e  0x5b  0x70  0x0e  0x6e  0x1a  0x6e
0x0a  0x38  0x54  0x71  0x0e  0x5b  0x70  0x0e
0x70  0x0e  0x54  0x71  0x0e  0x54  0x71  0x0e
0x5b  0x70  0x0e  0x52  0x2b  0x1a  0x22  0x70
0x1a  0x6e  0x52  0x6e  0x6e  0x71  0x0e  0x54
```

ภาพที่ 5 ตัวอย่างไฟล์ที่ได้จากกระบวนการสกัดอ็อบโค้ดจากไฟล์เด็กซ์คลาส

3.1.2 วิธีในการสกัดให้ได้การจัดกลุ่มแกรม 3

หลังจากที่ได้ไฟล์อ็อบโค้ดแล้ว ขั้นตอนต่อไปคือการจัดกลุ่มเอ็นแกรมของอ็อบโค้ด งานวิจัยนี้ใช้แกรม 3 (3-gram) ซึ่งเป็นการจัดกลุ่มอ็อบโค้ดจำนวน 3 โค้ดที่อยู่ติดกัน เนื่องจากหนึ่งคำสั่งของโปรแกรม (Programming statement) ประกอบด้วยชุดของอ็อบโค้ดหลายอ็อบโค้ด ทั้งนี้ระบบปฏิบัติการแอนดรอยด์มีจำนวนอ็อบโค้ดทั้งหมด 256 อ็อบโค้ด การใช้ 3-grams จะได้กลุ่มของอ็อบโค้ดที่เป็นไปได้ทั้งหมดจำนวน 2563 หรือ 16,777,216 รูปแบบ ถ้าใช้ n-grams มากกว่า 3 จะมีผลต่อการประมวลผลเป็นอย่างมาก โดยการจัดกลุ่มนี้เป็นแบบซ้อนทับกันได้เพื่อลดการสูญหายของข้อมูล ดังแสดงในภาพที่ 6 กลุ่มที่ 1) 0x54 0x1a 0x71

กลุ่มที่ 2) 0x1a 0x71 0x70 กลุ่มที่ 3) 0x71 0x70 0x28 กลุ่มที่ 4) 0x70 0x28 0x15 กลุ่มที่ 5) 0x28 0x15 0x70 กลุ่มที่ 6) 0x15 0x70 0x0e เป็นต้น เห็นได้ว่าการจัดกลุ่มนี้มีการซ้อนทับกันเกิดขึ้นและเป็นการจัดกลุ่มแบบเรียงตามลำดับ อาจจะกล่าวได้อีกแง่มุมหนึ่งคือ เป็นการใช้กรอบขนาด 3 โค้ด และเลื่อนกรอบไปทางขวาทีละโค้ด กลุ่มของแกรม 3 ที่ได้คือกลุ่มที่อยู่ภายในกรอบ

0x54	0x1a	0x71	0x70	0x28	0x15	0x70	0x0e
0x71	0x0a	0x71	0x11	0x71	0x71	0x0a	0x38
0x12	0x11	0x71	0x28	0x12	0x3b	0x11	0x71
0x28	0x0d	0x1a	0x22	0x70	0x1a	0x6e	0x6e
0x1a	0x6e	0x6e	0x71	0x28	0x15	0x6e	0x1f
0x14	0x6e	0x1f	0x1a	0x71	0x1a	0x71	0x6e

ภาพที่ 6 การสกัดเพื่อจัดหมู่ของแกรม 3 (3-gram combinations)

การจัดหมู่ของแกรม 3 นั้นเหมาะสมในหลายๆ งานวิจัย เช่น การตรวจการโจรกรรมทางวิชาการด้วยใช้เทคนิค N-gram ได้ทำการทดลองกับแกรม 3, 4, และ 5 และพบว่าแกรม 3 นั้นได้ค่าความถูกต้องดีสุดเมื่อเทียบกับเวลาที่ใช้ในการคำนวณ [11] เป็นต้น

3.1.3 ฟีเจอร์ความถี่

ในทางสถิติค่าความถี่ (Term-Frequency: TF) คำนวณได้จากเทอมที่สนใจเกิดขึ้นเป็นจำนวนเท่าไร ในที่นี้ผู้วิจัยสนใจจำนวนของการจัดกลุ่มแกรม 3 แบบใดแบบหนึ่งว่าเกิดขึ้นเป็นจำนวนเท่าไรในไฟล์ๆหนึ่ง ความถี่แบบนี้เรียกว่าความถี่สัมบูรณ์ (Absolute frequency) กำหนดให้ $f_{x_i}^{d_j}$ แทนค่าความถี่ที่เกิดขึ้นกับแกรม 3 แบบที่ x_i ในไฟล์เอกสาร d_j อาจกล่าวได้ว่า $f_{x_i}^{d_j}$ เป็นค่าความถี่สัมบูรณ์

สำหรับการคำนวณค่าความถี่สัมพัทธ์เป็นค่าความถี่เมื่อเทียบกับค่าความถี่โดยรวมของทุกไฟล์เอกสารในฐานข้อมูล กำหนดให้ ฐานข้อมูลเป็นเซตของไฟล์เอกสารดังนี้ $D = \{d_1, d_2, d_3, \dots, d_m\}$ โดยที่ m คือจำนวนของไฟล์เอกสารทั้งหมดในฐานข้อมูล D ความถี่สัมพัทธ์ของไฟล์เอกสารคำนวณได้ดังนี้

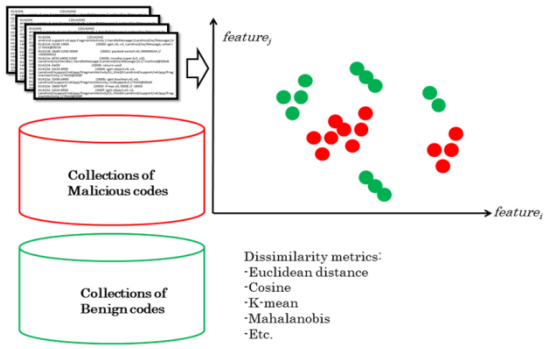
$$f_{x_i}'^{d_j} = \frac{f_{x_i}^{d_j}}{\mu} \text{ โดยที่ } \mu = \frac{\sum_{d_j \in D} f_{x_i}^{d_j}}{m}$$

3.2 การเรียนรู้ของเครื่องสำหรับตรวจจับโค้ดอันตราย

หลังจากที่ได้จัดกลุ่มแกรม 3 ของทุกไฟล์อีอบโค้ดทั้งหมดแล้ว (ความน่าจะเป็นเท่ากับ 2563 หรือ มากกว่า 16 ล้านแบบ) แต่พบว่าจำนวนกลุ่มแกรม 3 ที่ใช้อยู่จริงในฐานข้อมูลมีทั้งหมดเพียง 200,000 แบบ

อย่างไรก็ตามจำนวนฟีเจอร์ 200,000 แบบ ยังเป็นจำนวนฟีเจอร์ที่สูงมาก เทคนิคการลดจำนวนฟีเจอร์ที่นิยมใช้กันมาก คือ การวิเคราะห์ส่วนประกอบสำคัญ (Principal Component Analysis: PCA) จึงได้ถูกนำมาใช้ในการลดมิติของข้อมูลสำหรับการสร้างโมเดลการจำแนกโค้ดอันตรายและโค้ดปกติในขั้นต่อไป

คุณลักษณะเฉพาะของโปรแกรมเป็นเวกเตอร์ที่ประกอบด้วยลำดับของความถี่ที่ได้ ซึ่งเวกเตอร์คุณลักษณะเฉพาะนั้นจะต้องมาจากทั้งโค้ดปกติและโค้ดอันตรายเพื่อที่จะวิเคราะห์ค่าความแตกต่างระหว่างสองกลุ่ม โค้ดปกติและโค้ดอันตราย (แสดงในภาพที่ 7)



ภาพที่ 7 แนวคิดการสกัดลักษณะเฉพาะของโค้ดอันตราย

สำหรับการจำแนกโค้ดอันตรายและโค้ดปกติ ด้วยเทคนิคการจัดกลุ่ม (Classification) ผู้วิจัยได้เปรียบเทียบระหว่างเทคนิคการจัดกลุ่มที่ได้รับความนิยม 4 เทคนิค ได้แก่

1. เทคนิคเอสวีเอ็ม (Support Vector Machine: SVM) โดยใช้เคอร์แนลฟังก์ชันแบบโพลิโนเมียล (Polynomail Kernel Function)
2. เทคนิคเอสวีเอ็ม โดยใช้เคอร์แนลฟังก์ชันแบบอาร์บีเอฟ (RBF Kernel Function)
3. เทคนิคการเรียนรู้แบบเบย์อย่างง่าย (Naïve Bayes)
4. เครือข่ายประสาทเทียม (Artificial Neural Network)

เทคนิคเอสวีเอ็มเป็นเครื่องมือสำหรับการเรียนรู้แบบใหม่บนพื้นฐานของการเรียนรู้ทฤษฎีทางสถิติได้รับการนำเสนอครั้งแรกโดย Vapnik [12] เอสวีเอ็มใช้หลักการเรียนรู้แบบมีผู้สอนและเหมาะกับการจำแนกเป็น 2 กลุ่ม เพราะใช้หลักการสร้างเส้นแบ่งที่ขอบเขตระหว่างกลุ่มข้อมูลทั้งสองเส้นแบ่งขอบเขตนั้นนิยมใช้เคอร์แนลฟังก์ชัน 2 แบบคือ ฟังก์ชันโพลิโนเมียลและฟังก์ชันอาร์บีเอฟ

เทคนิคการเรียนรู้แบบเบย์อย่างง่ายเป็นการเรียนรู้ของเครื่องที่อาศัยหลักการความน่าจะเป็นตามทฤษฎีของเบย์ (Bayes's theorem) [13] ซึ่งมีอัลกอริทึมที่ไม่ซับซ้อนในการจำแนกข้อมูล โดยเรียนรู้ปัญหาที่เกิดขึ้นเพื่อสร้างเงื่อนไขการจำแนกข้อมูลใหม่ และคำนวณการแจกแจงความน่าจะเป็นตามสมมติฐานที่ตั้งให้กับข้อมูล

เครือข่ายประสาทเทียมเป็น โมเดลทางคณิตศาสตร์สำหรับประมวลผลสารสนเทศด้วยการคำนวณแบบคอนเนคชันนิสต์ (Connectionist) เพื่อจำลองการทำงานของโครงข่ายประสาทของมนุษย์ โดยจำลอง โครงข่ายไฟฟ้าชีวภาพ (Bioelectric network) ในสมอง ซึ่งประกอบด้วย เซลล์ประสาท (Neurons) และ จุดประสานประสาท (Synapses) โดยโครงข่ายประสาทเทียมที่ใช้คือ แบบเพอร์เซ็ปตรอนหลายชั้น (Multi-Layer Perceptron)

4. การทดลองและผลการทดลอง

4.1 ชุดข้อมูล

แอปพลิเคชันที่เป็นอันตรายที่สามารถรวบรวมได้มีทั้งหมด 500 โค้ดที่แตกต่างกัน ตารางที่ 1 และ 2 แสดงขนาดแพ็คเก็ตเอพีเคและขนาดคลาสเด็กซ์โดยเฉลี่ย สูงสุดและต่ำสุดในฐานข้อมูลหน่วยเป็นเมกกะไบต์ตามลำดับ เพื่อแสดงให้เห็นถึงความหลากหลายของไฟล์ที่รวบรวมได้ และขนาดของคลาสเด็กซ์ที่มีขนาดเล็กกว่าแพ็กเก็ตเอพีเคซีโดยส่วนใหญ่

ตารางที่ 1 ขนาดแพ็กเก็ตเอพีเคของไฟล์โค้ดปกติและไฟล์โค้ดอันตราย

ขนาดแพ็คเก็ตเอพีเค (เมกกะไบต์)	ไฟล์โค้ดปกติ		ไฟล์โค้ดอันตราย
	ไฟล์แอปพลิเคชัน	ไฟล์ระบบ	
จำนวน	400	100	500
ขนาดเล็กสุด	0.0035	0.0035	0.0151
ขนาดใหญ่สุด	49.5864	13.0703	30.4217
เฉลี่ย	3.0822	1.0352	1.4736

ตารางที่ 2 ขนาดไฟล์คลาสเด็กซ์ของไฟล์โค้ดปกติและไฟล์โค้ดอันตราย

ขนาดคลาสเด็กซ์ (เมกกะไบต์)	ไฟล์โค้ดปกติ		ไฟล์โค้ดอันตราย
	ไฟล์แอปพลิเคชัน	ไฟล์ระบบ	
จำนวน	400	100	500
ขนาดเล็กสุด	0.0035	0.0012	0.0045
ขนาดใหญ่สุด	7.3153	3.9141	4.8068
เฉลี่ย	1.5176	0.6286	0.4453

4.2 การทดลอง

สำหรับข้อมูลที่ใช้ในการทดลองแบ่งเป็น 2 กลุ่ม คือ ชุดการสอนและชุดทดสอบ

1. ชุดการสอน หมายถึง กลุ่มไฟล์ข้อมูลที่ใช้ในขั้นตอนการเรียนรู้เพื่อสร้างโมเดลการแยกแยะโค้ดอันตรายจากโค้ดปกติ
2. ชุดทดสอบ หมายถึง กลุ่มไฟล์ข้อมูลที่ไม่ได้ใช้ในการสร้างโมเดล

เพื่อให้ทุกไฟล์อยู่ในชุดทดสอบ คณะผู้วิจัยจึงใช้หลักการ cross-validation แบบ k=10 ในการวัดประสิทธิภาพทำการทดลองสร้างโมเดลทั้งหมด 10 รอบ โดยที่แต่ละครั้งใช้กลุ่มข้อมูลที่ประกอบด้วยไฟล์จำนวน 90% สำหรับชุดการสอนและ 10% สำหรับชุดทดสอบ และมีเงื่อนไขว่าในแต่ละรอบจะต้องได้ชุดทดสอบที่แตกต่างกันทั้ง 10 รอบ

การประเมินผลการทดลองนี้ได้ใช้ตัวชี้วัดคือ ร้อยละความถูกต้อง (Accuracy) ร้อยละของความไว (Sensitivity) และร้อยละของความจำเพาะ (Specificity) สูตรคำนวณตัวชี้วัดมีดังตารางที่ 3 และสมการ (1) – (3)

ตารางที่ 3 ตัวอย่างการคำนวณร้อยละความถูกต้อง ร้อยละความไว และร้อยละความจำเพาะ

จำนวนโค้ด		สภาพความจริง	
		โค้ดอันตราย	โค้ดปกติ
ผลลัพธ์ที่ได้จากโมเดล	โค้ดอันตราย	TP	FP
	โค้ดปกติ	FN	TN

$$Accuracy = \frac{(TP+TN)*100}{TP+FP+FN+TN} \quad (1)$$

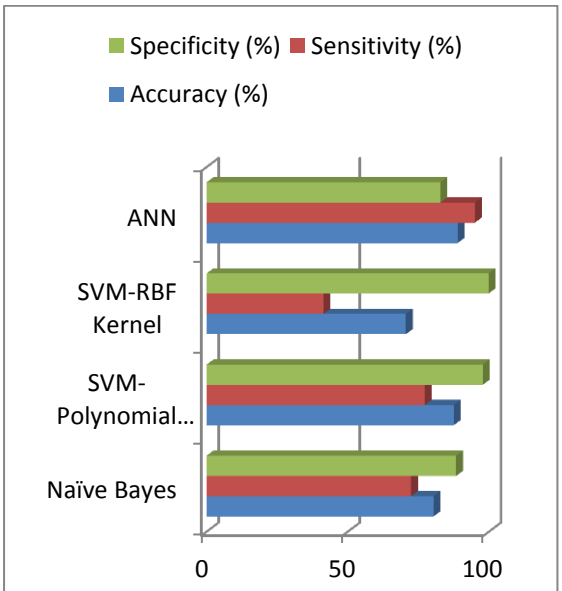
$$Sensitivity = \frac{TP*100}{TP+FN} \quad (2)$$

$$Specificity = \frac{TN*100}{TN+FP} \quad (3)$$

การทดลองนี้เน้นการทดสอบประสิทธิภาพของโมเดลการตรวจจับโค้ดอันตรายเป็นอย่างไรเมื่อพบไฟล์ที่ไม่รู้จัก (ไฟล์นี้ไม่อยู่ในชุดการสอน) โดยใช้ตรวจชี้วัดคือ ร้อยละความถูกต้อง ร้อยละของความไว และร้อยละของความจำเพาะ

4.3 ผลการทดลอง

ผลการทดลองแสดงในรูปแบบกราฟที่เปรียบเทียบระหว่างเทคนิคทั้ง 4 แบบ โดยใช้ฟีเจอร์ความถี่สัมพันธ์



	Naïve Bayes	SVM-Polynomial Kernel	SVM-RBF Kernel	ANN
Specificity (%)	88.4	97.8	99.8	82.8
Sensitivity (%)	72.4	77.2	41.4	95.0
Accuracy (%)	80.4	87.5	70.6	88.9

ภาพที่ 8 ค่าความถูกต้องของการตรวจจับแอปพลิเคชันอันตรายโดยใช้เทคนิคการจัดกลุ่มที่แตกต่างกัน

5. อภิปรายและสรุปผลการวิจัย

งานวิจัยนี้เน้นการพัฒนาแบบจำลองการตรวจจับโค้ดอันตรายโดยการวิเคราะห์สายอักษรจากไบนารีโค้ดของโปรแกรมบนเครื่องมือถือ และใช้เทคนิคการรู้จำเครื่องในการรู้จำกลุ่มของโค้ดอันตราย โดยคณะผู้วิจัยได้เสนอฟีเจอร์ค่าความถี่สัมพัทธ์สำหรับการตรวจจับโค้ดอันตรายที่ยังไม่ได้บันทึกไว้ในฐานข้อมูลได้ร้อยละความถูกต้อง 88.9 ในขณะที่ร้อยละค่าความไว และร้อยละความจำเพาะมีค่าเท่ากับ 95.0 และ 82.8 ตามลำดับ ซึ่งมีความหมายว่า แบบจำลองมีความไวในการตรวจจับโค้ดอันตราย 82.8 และถ้าโค้ดเป็นโค้ดปกติแบบจำลองจะระบุว่าโค้ดปกติได้ถูกต้องถึง 95.0

หลังจากที่ได้แบบจำลองการตรวจจับโค้ดอันตรายแล้ว คณะผู้วิจัยได้พัฒนาซอฟต์แวร์การตรวจจับโค้ดอันตรายบนสถาปัตยกรรมแบบระบบรับและให้บริการ ซึ่งมีคุณสมบัติดังนี้ 1) การสแกนโค้ดอันตรายทำได้ทั้งแบบ online และ offline 2) การสแกนแบบ offline สำหรับโปรแกรมที่มีการบันทึกในฐานข้อมูลแฮชโค้ดบนเครื่องมือถือ 3) การสแกนแบบ online สำหรับโปรแกรมที่ยังไม่มีการบันทึกในฐานข้อมูลแฮชโค้ดบนเครื่องมือถือ 4) การอัพเดตฐานข้อมูลแฮชโค้ดบนเซิร์ฟเวอร์ก็ต่อเมื่อมีโปรแกรมใหม่ที่สแกนแบบ online

การประเมินความพึงพอใจของผู้ใช้ที่มีต่อซอฟต์แวร์การตรวจจับโค้ดอันตรายพบว่า ผู้ใช้ส่วนใหญ่มีความพึงพอใจมากในเรื่องการติดตั้งและการเริ่มใช้งาน และผู้ใช้ให้คะแนนความสำคัญมากในเรื่องการใช้งานง่ายและค่าใช้จ่ายในการใช้ซอฟต์แวร์ ผลการประเมินนี้พบว่าผู้ใช้โดยส่วนใหญ่ไม่มีโปรแกรมแอนติไวรัสบนเครื่องมือถือแสดงให้เห็นว่าผู้ใช้อาจจะไม่ได้ตระหนักถึงภัยอันตรายในโลกไซเบอร์ ดังนั้นผู้ใช้ส่วนใหญ่จึงเลือกที่จะไม่ทำธุรกรรมธนาคารบนมือถือ

6. ข้อเสนอแนะ

ผลลัพธ์ของงานวิจัยนี้คือแบบจำลองการตรวจจับโค้ดอันตรายที่เน้นการคิดค้นวิธีการตรวจจับโค้ดอันตรายที่ไม่มีในฐานข้อมูล จากข้อสรุปข้างต้นเห็นได้ว่า แบบจำลองนี้ที่ได้นี้มีแนวโน้มในการนำออกสู่ตลาดได้จริงแต่ยังต้องใช้เวลาเพื่อปรับปรุงและพัฒนาในด้านประสิทธิภาพด้านการออกแบบการใช้งานเพื่อให้ตรงกับความต้องการของผู้ใช้ และการทดลองใช้งานเพื่อนำข้อเสนอแนะของผู้ใช้มาพัฒนาเพื่อสามารถพัฒนาในเชิงพาณิชย์ได้

7. กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จได้ด้วยทุนสนับสนุนจากสำนักงานคณะกรรมการวิจัยแห่งชาติ (วช.) ในปีงบประมาณ 2556

เอกสารอ้างอิง

[1] สำนักงานพัฒนาธุรกรรมทางอิเล็กทอรนิกส์ (องค์การมหาชน) กระทรวงเทคโนโลยีสารสนเทศและการสื่อสาร, Thailand Internet User Profile 2014 รายงานผลการสำรวจพฤติกรรมผู้ใช้อินเทอร์เน็ตในประเทศไทย ปี 2557, https://www.etda.or.th/download-publishing/12/ สืบค้นเมื่อวันที่ 1 ตุลาคม 2558.

[2] ธนาคารแห่งประเทศไทย, รายงานระบบการชำระเงิน 2557, https://www.bot.or.th/Thai/Pay mentSystems/ Documents/Payment_2014_T.pdf, สืบค้นเมื่อวันที่ 1 ตุลาคม 2558.

[3] Jennifer Scott, Adroid set to reach one billion users in 2014, http://www.computerweek ly.com/news/2240212085 /Android-set-to-reach-one-billion-users-in-2014, สืบค้นเมื่อวันที่ 1 ตุลาคม 2558.

[4] Makan, Keith, and Scott Alexander-Bown. Android Security Cookbook. Packt Publishing Ltd, 2013.

[5] Dilrukshi, Inoshika, Kasun De Zoysa, and Amitha Caldera. "Twitter news classification using SVM." Computer Science & Education (ICCSE), 2013 8th International Conference on. IEEE, 2013.

[6] สมภพ ปฐมนพ, กฤษฎา ศรีแผ้ว และ ม.ล.กุลธร เกษมสันต์ "ข้อมูลเชิงเวลากับการจำแนกประเภทผู้เป็นโรคเบาหวานในประเทศไทย," Journal of Information Science and Technology, Vol.4, No.1 , pp.49-56.

[7] Moskovitch, Robert, Yuval Elovici, and Lior Rokach. "Detection of unknown computer worms based on behavioral classification of the host." Computational Statistics & Data Analysis 52.9 (2008): 4544-4566.

[8] Abou-Assaleh, Tony, Nick Cercone, Vlado Keselj, and Ray Sweidan. "N-gram-based detection of new malicious code." In Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International, vol. 2, pp. 41-42. IEEE, 2004.

[9] Kolter, Jeremy Z., and Marcus A. Maloof. "Learning to detect malicious executables in the wild." Journal of Machine Learning Research. Vol. 7, 2006. 2721-2744

[10] Zheng, M., Sun, M., & Lui, J. Droid analytics: A signature based analytic system to collect, extract, analyze and associate android malware. In Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on (pp. 163-171).

[11] สรวัตร ประภานิติเสถียร และ, ไกรศักดิ์ เกษร "การตรวจการโจรกรรมทางวิชาการด้วยใช้เทคนิค N-gram ร่วมกับเทคนิคการตรวจสอบเชิงความหมายสาหรับเอกสารภาษาไทย", Journal of Information Science and Technology, Vol.5, No.1 , pp.42-50.

[12] Vapnik, V. The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.

[13] Lewis, David D. "Naive (Bayes) at forty: The independence assumption in information retrieval." Machine learning: ECML-98. Springer Berlin Heidelberg, 1998. 4-15.