ขั้นตอนวิธีการตรวจจับโค้ดที่เป็นอันตรายบนระบบปฏิบัติการมือถือ

Detection Algorithm of Malicious Code on Mobile Operating Systems

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,

บทคัดย่อ

งานวิจัยนี้ได้นำเสนอการใช้แบบจำลองการตรวจจับโค้ดอันตรายบนระบบปฏิบัติการของมือถือ โดยใช้ทฤษฎีการเรียนรู้ของเครื่องมีเป้าหมายเพื่อลดความเสี่ยงต่อการติดตั้งโปรแกรมไม่พึงประสงค์ต่างๆ ของผู้ใช้ที่ไม่ได้อัพเดตโปรแกรมแอนติไวรัสทันเวลา ซึ่งแบบจำลองที่ได้นำเสนอนี้ต่างจากโปรแกรมแอนติไวรัส ทั่วไปตรงที่ โปรแกรมแอนติไวรัสทั่วไปนิยมใช้หลักการของการตรวจจับรูปแบบสายอักษรที่เฉพาะเจาะจงที่ฝังในโปรแกรมเพื่อระบุว่าโปรแกรมนั้นอันตรายหรือไม่ แต่หลักการของการตรวจจับรูปแบบสายอักษรที่เฉพาะเจาะจงนั้น ทุกๆครั้งที่มีไวรัสหรือโค้ดอันตรายตัวใหม่ขึ้นมา ผู้ใช้จำเป็นต้องอัพเดตโปรแกรมแอนติไวรัสเสมอ ซึ่งเมื่อไรก็ตามที่ผู้ใช้พลาดการอัพเดตให้ทันเวลาก็จะตกเป็นเป้าของไวรัสตัวใหม่เสมอ ในขณะที่แบบจำลองที่นำเสนอได้ใช้เทคนิคการเรียนรู้ของเครื่อง ในการรู้จำกลุ่มของโค้ดอันตรายทำให้ยังคงตรวจจับโค้ดอันตรายได้ถึงแบบว่าโค้ดอันตรายตัวใหม่ได้มีรูปแบบแตกต่างไปบ้างก็ตาม

ขั้นตอนในการดำเนินการวิจัยเริ่มจากการ **(1)** รวบรวมแอปพลิเคชันบนระบบปฏิบัติการแอนดรอยด์ทั้งแอปพลิเคชันปกติและแอปพลิเคชันที่เป็นอันตราย **(2)** การสกัดคุณลักษณะ โดยวิเคราะห์การกระจายตัวของความถี่ของไบต์เอ็นแกรม ค่าความถี่ของเทอมและค่าความผกผันของเทอม ซึ่งในงานวิจัยนี้ **n** มีค่าเท่ากับ **3 (3)** สร้างโมเดลของการจำแนกโค้ดอันตรายจากใช้คุณลักษณะที่ได้ทั้งในส่วนของแอปพลิเคชันปกติและแอปพลิเคชันที่เป็นอันตราย ในการทดลองได้ใช้โค้ดอันตรายจำนวน **304** โค้ด และโค้ดปกติจำนวน **553** โค้ด สำหรับการสร้างโมเดลของการจำแนกโค้ดอันตราย และสำหรับการทนต่อโค้ดอันตรายใหม่ที่ไม่ได้อยู่ในฐานข้อมูล พบว่าโมเดลที่ได้นำเสนอนั้นมีร้อยละความแม่นยำมากกว่า **85.52** ในขณะที่ร้อยละค่าความไว และร้อยละความจำเพาะมีค่าเท่ากับ **71.26** และ **90.52** ตามลำดับ

คำสำคัญ : ปัจจัยที่มีผลต่อการเพิ่มประสบการณ์; พิพิธภัณฑ์; การยอมรับเทคโนโลยี; สื่อสังคมออนไลน์

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 TF Inverse Document Frequency (TFIDF) where the parameter n = 3 is used, and (3) constructing a model for classification the malicious codes using the extracted features for both malicious and benign codes. In the experiment, 304 malicious codes and 553 benign codes were using to construct the model. The experiment shows that the model achieved more than 85.52% accuracy. For the sensitivity and specificity, the model achieved 90.52% and 71.26%, respectively.

Keywords : Antivirus, Android, Feature extraction, Term Frequency-Inverse-Document Frequency (TF-IDF), Principal Component Analysis (PCA)

---

1. บทนำ

ในปี พ.ศ. 2555 พบว่าผู้ลงทะเบียนเปิดใช้เบอร์โทรศัพท์เคลื่อนที่ในประเทศไทยมีจำนวนถึง 81.68 ล้านราย[(1)] คิดเป็นร้อยละ 120.83 ต่อประชากร ประเทศไทยเป็น 1 ใน 8 ของโลกที่มีจำนวนโทรศัพท์เคลื่อนที่สูงกว่าจำนวนประชากร แสดงให้เห็นว่าการใช้โทรศัพท์เคลื่อนที่ในประเทศไทยเป็นที่นิยมอย่างมาก ด้วยปัจจัยนี้ธนาคารต่างๆ จึงพัฒนาแอปพลิเคชันสำหรับทำธุรกรรมทางโทรศัพท์เคลื่อนที่ (mobile banking) เพื่ออำนวยความสะดวกให้ลูกค้าสามารถชำระเงินทางอิเล็กทรอนิกส์ (e-Payment) ซึ่งอัตราเติบโตของการทำธุรกรรมเป็นแบบเลขชี้กำลังเริ่มตั้งแต่ปี พ.ศ. 2552 และรวมมูลค่าการทำธุรกรรมเป็นจำนวนเงิน 439,960 ล้านบาท ในปี พ.ศ. 2555[(2)] สถิติเหล่านี้ชี้ให้เห็นว่า ผู้ใช้จำเป็นต้องมีโปรแกรมป้องกันการโจรกรรมข้อมูลส่วนตัวบนโทรศัพท์เคลื่อน

จากการศึกษาและสำรวจพบว่าร้อยละ 70 ของมัลแวร์บนโทรศัพท์เคลื่อนที่ (mobile malware) มุ่งขโมยข้อมูลส่วนตัว[(3)] เช่น ข้อมูลเกี่ยวกับบัตรเครดิตและบัญชีธนาคาร ล็อคอินและพาสเวิร์ด เป็นต้น การป้องกันโปรแกรมไม่พึงประสงค์รวมถึงไวรัส และมัลแวร์ต่างๆ (malwares) ผู้ใช้มักนิยมใช้โปรแกรมแอนติไวรัส (Anti-Virus) ซึ่งใช้หลักการของการตรวจจับรูปแบบสายอักษรที่เฉพาะเจาะจงที่ฝังในโปรแกรมเพื่อระบุว่าโปรแกรมนั้นอันตรายหรือไม่ ทุกครั้งที่เกิดไวรัสตัวใหม่ขึ้นมาผู้ใช้จำเป็นต้องอัพเดตโปรแกรมแอนติไวรัสเสมอ เมื่อไรก็ตามที่ผู้ใช้พลาดการอัพเดตให้ทันเวลาก็จะตกเป็นเป้าของไวรัสตัวใหม่เสมอ ในงานวิจัยนี้ได้นำเสนอการใช้แบบจำลองการตรวจจับโค้ดอันตรายบนระบบปฏิบัติการของมือถือ โดยใช้ทฤษฎีการเรียนรู้ของเครื่อง (machine learning) โดยมีเป้าหมายเพื่อลดความเสี่ยงต่อการติดตั้งโปรแกรมไม่พึงประสงค์ต่างๆ ของผู้ใช้ที่ไม่ได้อัพเดตโปรแกรมแอนติไวรัสทันเวลา

แอปพลิเคชันที่มีโค้ดอันตรายฝังตัวอยู่มีจำนวนเพิ่มขึ้นอย่างรวดเร็ว ในที่นี่เรียกสั้นๆ ว่า "แอปพลิเคชันอันตราย" ซึ่งแอปพลิเคชันอันตรายเหล่านี้มีการพัฒนาตัวเองด้วยเทคนิคต่างๆ เพื่อให้สามารถหลีกเลี่ยงการตรวจจับโค้ดอันตรายด้วยโปรแกรมแอนติไวรัสที่มีอยู่ในท้องตลาดได้และเข้าไปฝังตัวในอุปกรณ์เคลื่อนที่ของผู้ใช้ ภาษาที่ใช้เขียนแอปพลิเคชันบนแอนดรอยด์เป็นภาษาจาวาเนื่องจากภาษาจาวามีไลบารี่ที่สนับสนุนการเขียนแอปพลิเคชันจำนวนมากและไลบารี่เหล่านี้สามารถช่วยนักเขียนโปรแกรมให้เขียนแอปพลิเคชันที่ซับซ้อนได้ไม่ยาก ด้วยจำนวนไลบารี่ของจาวามีมากมายซึ่งใช้ในแอปพลิเคชันปกติและแอปพลิเคชันอันตรายทำให้การวิเคราะห์และแยกแยะแอปพลิเคชันอันตรายเป็นปัญหาที่ยาก

ถึงแม้ในเว็บไซต์ของกูเกิ้ลเพลย์สโตว์)Google Play Store) ได้ให้ข้อมูลการรักษาความมั่นคงปลอดภัย (security (เพื่อป้องกันแอปพลิเคชันอันตรายและรับรองว่าแอปพลิเคชันบนเพลย์สโตว์ปราศจากโค้ดอันตราย100% แต่ในความเป็นจริงทั้งรายงานและบทความวิจัยเปิดเผยว่าไม่เป็นความจริง ยังมีแอปพลิเคชันอันตรายบนเพลย์สโตว์

1.1 วัตถุประสงค์ของการวิจัย

1) เพื่อพัฒนาแบบจำลองการตรวจจับโค้ดอันตรายโดยการวิเคราะห์สายอักษรจากไบนารีโค้ดของโปรแกรมบนเครื่องมือถือ และใช้เทคนิคการเรียนรู้ของเครื่องในการรู้จำกลุ่มของโค้ดอันตราย

2) เพื่อทำการประเมินประสิทธิภาพของการตรวจจับโค้ดอันตรายบนระบบปฏิบัติการมือถือ โดยลดจำนวนครั้งของการอัพเดตโค้ดอันตรายตัวใหม่ เมื่อเปรียบเทียบกับโปรแกรมแอนติไวรัสที่มีในตลาด

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การวิจัยนี้เป็นการวิจัยการออกแบบและพัฒนาโปรแกรมประยุกต์บนระบบปฏิบัติการแอนดรอยด์ เนื่องด้วยจำนวนผู้ใช้ระบบปฏิบัติการแอนดรอยด์มีมากกว่าหนึ่งพันล้านคนในปลายปี ค.ศ. 2014 (1) ด้วยยอดผู้ใช้จำนวนมากเช่นนี้ทำให้ระบบปฏิบัติการแอนดรอยด์เป็นที่ดึงดูดนักเขียนโค้ดอันตราย (malicious programmers)

2.1 ระบบปฏิบัติการแอนดรอยด์

ไฟล์แอปพลิเคชันบนแอนดรอยด์ หรือ แพ็คเก็ตแอปพลิเคชันแอนดรอยด์ (Android application package) เรียกสั้นๆ ว่า เอพีเค (APK) ซึ่งเป็นรูปแบบของไฟล์บีบอัด (zip file) โดยอาศัยพื้นฐานไฟล์ จาร์ (jar) คล้ายคลึงกับ เอ็มเอสไอ (MSI) บนระบบปฏิบัติการวินโดว์ แพ็คเก็ตเอพีเคเป็นรูปแบบในการส่งต่อ เผยแพร่ และสามารถติดตั้งบนอุปกรณ์ที่ใช้ระบบปฏิบัติการแอนดรอยด์ผ่านเพลย์สโตว์ของกูเกิ้ล แพ็คเก็ตเอพีเคมีองค์ประกอบหรือประกอบด้วยไฟล์ต่างๆ ดังนี้ [(1)]

1) ไฟล์เอ็กเอ็มแอล ชื่อว่า "AndroidManifest.xml" เป็น ไฟล์ที่บ่งบอกข้อมูล ชื่อ เวอร์ชัน และสิทธิของการเข้าถึงทรัพยากรต่างๆ (access permission) และข้อมูลหรือกิจกรรมต่างๆ ของแอปพลิเคชัน

2) ไฟล์เด็กซ์คลาส ชื่อว่า "Classes.dex" เป็นโปรแกรมหลักที่รันบนระบบปฏิบัติการแอนดรอยด์ ไฟล์ของโปรแกรมเป็นไบต์โค้ด (bytecode) แต่ไบต์โค้ดนี้ต้องรันบนแอนดรอยด์ ขั้นตอนการแปลงจึงประกอบด้วย 2 ขั้นตอน[(2)] ก) ใช้คอมไพเลอร์จาวา ชื่อว่า "javac" แปลงจากคลาสจาวาเป็นไบต์โค้ดที่สามารถรันบนเครื่องเสมือนจาวา (Java virtual machine) ข) ใช้คอมไพเลอร์ดีเอ็กซ์แอนดรอยด์" (Android DX compiler[(3)]) แปลงไบต์โค้ดจาวาไปเป็นไฟล์ดัลวิก (Dalvik) ซึ่งมีนามสกุล ".dex" ขั้นตอนที่กล่าวมาแสดงในภาพที่ 2.2

3) ไดเร็กทอรีเรส ชื่อว่า "Res directory" เป็นโฟลเดอร์ที่เก็บไฟล์ทรัพยากรเสริมต่างๆ ของแอปพลิเคชัน เช่น ไฟล์รูปภาพ วีดีโอ เสียง และไฟล์ข้อมูล เป็นต้น นอกจากนี้ในไดเร็กทอรีนี้ยังสามารถบรรจุไฟล์ที่เกี่ยวข้อง เช่น ไลบารีภายนอก เป็นต้น

สำหรับการวิจัยนี้ คณะผู้วิจัยได้จัดประเภทของแพ็คเก็ตเอพีเค โดยใช้หลักของความมั่นคงปลอดภัยและสามารถแบ่งได้เป็น 3 ประเภทดังนี้:

1) ไฟล์ระบบ (system Files): แอปพลิเคชันเหล่านี้ถือเป็นส่วนหนึ่งของระบบปฏิบัติการแอนดรอยด์ ซึ่งอาจจะเป็นไฟล์เฉพาะของแต่ละยี่ห้อของอุปกรณ์เคลื่อนที่ แอปพลิเคชันที่มาพร้อมกับระบบปฏิบัติการ เช่น ปฏิทิน กล้องถ่ายรูป บราวเซอร์ สมุดบันทึก สมุดรายชื่อ และอื่นๆ

2) ไฟล์ปกติ (Benign Files): แอปพลิเคชันใดๆ ก็ตามที่เชื่อถือได้และปลอดภัยสำหรับอุปกรณ์แอนดรอยด์แล้ว แอปพลิเคชันนั้นจะจัดอยู่ในไฟล์ปกติ หรือแอปพลิเคชันปกติ ดังนั้นไฟล์ระบบทั้งหมดจัดอยู่ในไฟล์ปกติโดยปริยายเพราะไฟล์ระบบมากับอุปกรณ์ตั้งแต่โรงงานผลิต แอปพลิเคชันปกติจะทำงานบนอุปกรณ์ตามที่ได้ระบุไว้ หรือตามที่ผู้ใช้เข้าใจ และเข้าถึงทรัพยากรระบบและข้อมูลต่างๆ ของผู้ใช้หลังจากที่ได้รับอนุญาตจากผู้ใช้แล้วเท่านั้น แอปพลิเคชันปกติสามารถดาวโหลดได้ผ่านเพลย์สโตว์ ตัวอย่างของแอปพลิเคชันปกติ เช่น ออฟฟิคโปลาริส (Polaris office) กูเกิ้ลพัส (Google plus) ชุดออฟฟิค (Office suite) อร์อะโดบี (Adobe reader) เป็นต้น

3) ไฟล์ไวรัส (Virus Files): แอปพลิเคชันที่เป็นอันตรายเป็นแอปพลิเคชันทีสามารถทำงานหรือสั่งงานที่เป็นอันตรายบนอุปกรณ์แอนดรอยด์ อันตรายที่ว่านี้คือ ก) การขโมยข้อมูล ข้อมูลความลับ ข้อมูลส่วนบุคคล เป็นต้น ข) การขโมยข้อมูลที่เป็นการแสดงตัวตนที่เก็บไว้บนอุปกรณ์แอนดรอยด์และส่งต่อข้อมูลเหล่านี้โดยไม่ได้รับอนุญาตจากผู้ใช้ ค) การเรียกค่าไถ่ เป็นรูปแบบใหม่ที่เพิ่งค้นพบ แอปพลิเคชันนี้มีรูปแบบคือ การบีบอัดและเข้ารหัสข้อมูลข้อผู้ใช้จากนั้นจะส่งข้อความเรียกค่าไถ่ถ้าผู้ใช้ต้องการถอดรหัสเพื่อเข้าถึงข้อมูลของตนเอง

2.2 การเรียนรู้ของเครื่อง

การเรียนรู้ของเครื่อง (machine learning) เป็นสาขาหนึ่งของปัญญาประดิษฐ์ ที่เกี่ยวข้องกับการพัฒนาเทคนิควิธีเพื่อให้คอมพิวเตอร์สามารถเรียนรู้ โดยเน้นที่วิธีการเพื่อสร้างโปรแกรมคอมพิวเตอร์จากการวิเคราะห์ชุดข้อมูล การเรียนรู้ของเครื่องจึงเกี่ยวข้องอย่างมากกับสถิติศาสตร์ เนื่องจากทั้งสองสาขาศึกษาการวิเคราะห์ข้อมูลเช่นเดียวกัน

เทคนิคการเรียนรู้ของเครื่องถูกใช้เพื่อเพิ่มประสิทธิภาพในการแก้ปัญหาด้านต่าง ๆ เช่น การสร้างให้คอมพิวเตอร์สามารถแยกแยะวัตถุ เสียง หรือตัวอักษรได้ หรือจำแนกข้อมูลจำนวนมากที่ไม่สามารถทำได้โดยมนุษย์ ลักษณะทั่วไปของการเรียนรู้ของเครื่องจักรเป็นการสร้างอัลกอริธึมหรือโปรแกรมคอมพิวเตอร์ จากการให้ข้อมูลฝึก (training data) สำหรับสอนให้คอมพิวเตอร์เรียนรู้ เพื่อให้ได้สมมติฐานในการนำมาใช้แยกแยะวัตถุอื่นได้ ลักษณะการเรียนรู้ของเครื่องจักรแสดงได้ดังภาพที่ 2.3 เทคนิคการเรียนรู้ของเครื่องจักร

ขั้นตอนวิธีการเรียนรู้ของเครื่อง จัดแบ่งได้ตามลักษณะผลลัพธ์ โดยทั่วไปแล้วจะแบ่งได้ดังนี้

1) การเรียนรู้แบบมีผู้สอน (supervised learning) - ขั้นตอนวิธีสร้างฟังก์ชันซึ่งเชื่อมระหว่างข้อมูลเข้ากับผลที่ต้องการ
2) การเรียนรู้แบบไม่มีผู้สอน (unsupervised learning) - ขั้นตอนวิธีสร้างโมเดลจากชุดข้อมูลเข้า
3) การเรียนรู้แบบเสริมกำลัง (reinforcement learning) - ขั้นตอนวิธีเรียนแผนซึ่งกำหนดการกระทำของระบบจากสิ่งที่สังเกตได้
4) การเรียนวิธีการเรียน (learning to learn, meta-learning) - ขั้นตอนวิธีที่เรียนวิธีการเรียนรู้ของตนเอง โดยปรับปรุง inductive bias ที่เป็นข้อสมมติฐานที่ขั้นตอนวิธีใช้ในการเรียนรู้

นอกจากนี้ การวิเคราะห์ประสิทธิภาพและการคำนวณของขั้นตอนวิธีการเรียนรู้ เป็นสาขาหนึ่งของวิชาสถิติซึ่งเรียกว่า ทฤษฎีการเรียนรู้ อย่างไรก็ตาม หากแบ่งประเภทเทคนิคการเรียนรู้ของเครื่องจักรตามลักษณะการใช้ข้อมูลฝึกสามารถแบ่งได้ดังนี้

1) การเรียนรู้แบบมีผู้สอน (supervised Learning) เทคนิคการเรียนรู้ของเครื่องจักรประเภทนี้ต้องใช้การเรียนรู้จากข้อมูลฝึกที่มีการใส่ฉลาก (label) ให้กับข้อมูลฝึกไว้แล้ว เพื่อให้คอมพิวเตอร์เข้าใจรูปแบบและได้สมมติฐานเพื่อทำงานกับข้อมูลในภายหน้าได้ ตัวอย่างเทคนิคประเภทนี้ได้แก่ การเรียนรู้แบบตัวจำแนกแบบเบย์สอย่างง่าย และการเรียนรู้แบบต้นไม้ตัดสินใจ เป็นต้น
2) การเรียนรู้แบบไม่มีผู้สอน (unsupervised Learning) ใช้ข้อมูลฝึกหรือชุดตัวอย่างที่ไม่มีการใส่ฉลากให้กับข้อมูล และเรียนรู้โดยการนำข้อมูลไปผ่านกระบวนการหาความคล้ายคลึงของตัวอย่าง จนกระทั่งได้กลุ่มตัวอย่างที่จัดเป็นประเภทอย่างเหมาะสม เทคนิคประเภทนี้ได้แก่ การแบ่งกลุ่ม (clustering)
3) การเรียนรู้แบบกึ่งมีผู้สอน (semi supervised Learning) ใช้ข้อมูลฝึกที่มีการใส่ฉลากเพียงบางส่วนของข้อมูลฝึกทั้งหมด การเรียนรู้ลักษณะนี้มีความคล้ายคลึงกับการเรียนรู้

แบบมีผู้สอนแต่ใช้ข้อมูลที่ไม่มีฉลากประกอบเพื่อช่วยให้เรียนรู้ได้ดีขึ้น สิ่งที่ได้จากการเรียนรู้แบบกึ่งมีผู้สอนจะเหมือนกับการเรียนรู้แบบมีผู้สอนคือได้แบบจำลอง

หลายปีมานี้ได้มีงานวิจัยจำนวนมากที่มุ่งเน้นการตรววจจับโค้ดอันตรายที่ไม่รู้จักในบนเครื่องคอมพิวเตอร์ส่วนบุคคลจากไบนารีโค้ด โดยริเริ่มใช้ทฤษฎีทางปัญญาประดิษฐ์ การเรียนรู้ของเครื่อง (Machine Learning: ML) ผู้ริเริ่มแนวคิดนี้ได้แก่ Moskovitch et al. [1] งานวิจัยนี้ได้นำเสนอการใช้คุณสมบัติเหล่านี้ (1) ส่วนหัวของโปรแกรม (2) ตัวอักขระ (คำที่มีความหมายที่เข้ารหัสไว้ในส่วนของโปรแกรม เช่น “windows”, “kernel”, “reloc” เป็นต้น) และ (3) ลำดับของไบต์ Abou-Assaleh et al. [2] ได้นำเสนอการใช้ Common N-Gram (CNG) เพื่อตรวจจับโค้ดอันตราย วิธีการนี้เริ่มต้นด้วยการสร้างโปรไฟล์ของ executable file ทั้งที่เป็นไฟล์อันตรายและปลอดภัย จากนั้นเมื่อมีไฟล์ใหม่เข้ามาก็จะทำการหาว่าไฟล์ดั้งกล่าวคล้ายคลึงกับโปรไฟล์ของไฟล์ใดมากที่สุด Kotler และ Maloof [3] ได้ใช้กลุ่มของไฟล์ปลอดภัยจำนวน 1,971 ไฟล์ และไฟล์อันตรายจำนวน 1,651ไฟล์ และในการทดลองนั้นได้เลือกใช้ classifiers ต่างๆ ดังนี้ KNN, TFIDF classifier, Naïve Bayes, SVM, และ Decision Tree (J48) โดยที่ J48, SVM, Boosted SVM และ IBK เป็น classifier ที่ให้ความถูกต้องมากที่สุด

3. วิธีดำเนินงานวิจัย

ขั้นตอนการดำเนินการวิจัยโดยสรุปของของการพัฒนาอัลกอริทึมสำหรับการตรวจจับโค้ดที่เป็นอันตรายบนระบบปฏิบัติการมือถือ มีดังนี้

1) สร้างฐานข้อมูลสำหรับเพื่อใช้ในการเรียนรู้รูปแบบโค้ดอันตราย โดยรวบรวมไบนารีไฟล์ของระบบปฏิบัติการแอนดรอยด์ และไบนารีไฟล์ของโค้ดอันตรายจากแหล่งข้อมูลที่ได้รับการยอมรับ
2) วิเคราะห์รูปแบบสายอักษรของโค้ดปกติ และโค้ดอันตราย โดยพิจารณาจากการกระจายตัวของความถี่ byte n-grams และ TF Inverse Document Frequency (TFIDF) กล่าวได้ว่าแต่ละโปรแกรมประกอบเซตของอ็อบโค้ด (opcode) ซึ่งมีลักษณะเป็นลำดับของคำสั่ง ลำดับของคำสั่งสามารถแบ่งเป็นลำดับย่อยๆ ได้ เช่น ลำดับที่มี 1 คำสั่ง (1-term), 2 คำสั่ง (2-term), 3 คำสั่ง (3-term) เป็นต้น จากนั้นคำนวณค่าความถี่ของการซ้ำของลำดับย่อยๆ โดยใช้ทฤษฎีการวิเคราะห์สายอักษร (text analysis) ดังแสดงในภาพที่ 3.1 ตารางความถี่ถูกสร้างขึ้นเพื่อใช้ในการสกัดคุณลักษณะของโปรแกรม
4) สร้างโมเดลการจำแนกโค้ดอันตราย แนวคิดการสกัดลักษณะเฉพาะของโค้ดอันตราย (แสดงในภาพที่ 3.3) คุณลักษณะเฉพาะของโปรแกรมเป็นเวกเตอร์ที่ประกอบด้วยลำดับของความถี่ที่ได้ ซึ่งเวกเตอร์คุณลักษณะเฉพาะนั้นจะต้องมาจากทั้งโค้ดปกติและโค้ดอันตราย เพื่อที่จะวิเคราะห์ค่าความแตกต่างระหว่างสองกลุ่ม โค้ดปกติและโค้ดอันตราย
5) สร้างโมเดลของการแยกแยะโค้ดอันตรายจากโค้ดปกติ (ที่ได้จากข้อ 4) ด้วยเทคนิคการจัดกลุ่ม เช่น Support Vector Machine (SVM) เป็นต้น
6) ออกแบบและทดลองผลการทดลองกับฐานข้อมูล (ที่ได้สร้างไว้ในข้อ 1) เพื่อทดสอบว่าโมเดลที่ได้มานั้นสามารถนำครอบคลุมไวรัสและมัลแวร์ ต่างๆ ได้
7) พัฒนาโปรแกรมเพื่อใช้งานจริงบนระบบปฏิบัติการแอนดรอยด์
8) ทดสอบการใช้งานและประเมินผล

4. ผลการศึกษา

4.1 ฐานข้อมูลที่รวบรวมได้ของแอปพลิเคชันที่เป็นอันตรายและแอปพลิเคชันปกติ

แอปพลิเคชันที่เป็นอันตรายที่สามารถรวบรวมได้มีทั้งหมด 304 **โค้ดที่แตกต่างกันและต่างประเภทกัน**(1) **ประเภทของโค้ดอันตรายสามารถจัดได้ดังแสดงในตารางที่** 4.1 **ซึ่งมีคำอธิบายถึงพฤติกรรมของโค้ดอันตรายแต่ละประเภทซึ่งใช้ยุทธวิธีในการเจาะเข้าระบบรักษาความปลอดภัยที่แตกต่างกัน**(2) **จากตารางที่** 4.1 **คณะผู้วิจัยสามารถระบุประเภทของโค้ดอันตรายได้** 157 **และไม่สามารถระบุประเภทได้** 147

ไฟล์แพ็กเก็ตเอพีเคในฐานข้อมูลมีทั้งไฟล์แอปพลิเคชัน ไฟล์ระบบ จำนวน 500 **และ** 53 **ไฟล์นอกเหนือจากไฟล์โค้ดไวรัสจำนวน** 304 **ไฟล์ ตารางที่** 4.2-4.3 **แสดงขนาดแพ็คเก็ตเอพีเคและขนาดคลาสเด็กซ์โดยเฉลี่ย สูงสุดและต่ำสุดในฐานข้อมูลหน่วยเป็นเมกกะไบต์ตามลำดับ เพื่อแสดงให้เห็นถึงความหลากหลายของไฟล์ที่รวบรวมได้ และขนาดของคลาสเด็กซ์ที่มีขนาดเล็กกว่าแพ็กเก็ตเอพีเคซีโดยส่วนใหญ่(แสดงการเปรียบเทียบขนาดของไฟล์ในตารางที่** 4.4**)**

4.2 ฮาร์ดแวร์และข้อมูลที่ใช้ในการทดลอง

ฮาร์ดแวร์ที่ใช้คือ

- เครื่องคอมพิวเตอร์ตั้งโต๊ะ[(1)] (PC) 2 เครื่องโดยมีรายละเอียดดังนี้

  เครื่องที่ 1 Intel i7-2600 CPU @ 3.40 GHZ, Main Memory: 4.00 GB, HDD: 1 TB, OS. Win 7- 32 bit

  เครื่องที่ 2 Intel i7-2600 CPU @ 3.40 GHZ, Main Memory: 2.00 GB, HDD: 1 TB, OS. Win 7- 32 bit

-เครื่องเซิร์ฟเวอร์[(2)] HP Z230 i7-4770, 8 GB, 500 GB

-มือถือระบบปฏิบัติการ Android[(2)] เวอร์ชัน 4.3 Jelly Beam หน่วยประมวลผล Quad-Core Krait 400 2.3 GHz ความจำภายใน 32 GB

ข้อมูลที่ใช้คือ ฐานข้อมูลในข้อ 4.1 ประกอบด้วยไฟล์โค้ดปกติจำนวน 553 และไฟล์โค้ดอันตรายจำนวน 304

4.3 การทดลอง

การทำงานของซอฟต์แวร์ตรวจจับโค้ดอันตรายแบ่งเป็นการการทดลองแบ่งออกเป็น 2 กรณี คือ

1) กรณีไฟล์นั้นเป็นไฟลที่ได้ทำการตรวจสอบแล้ว: การตรวจจับโค้ดอันตรายจากฐานข้อมูลบนเครื่องแอนดรอยด์ กรณีที่ไฟล์นั้นได้เคยบันทึกลงในฐานข้อมูลว่าเป็นไฟล์ที่มีโค้ดอันตรายหรือไม่แล้ว
2) กรณีไฟล์นั้นเป็นไฟล์ใหม่: การตรวจจับโค้ดอันตรายโดยใช้โมเดลที่ได้นำเสนอบนเซิร์ฟเวอร์

การทดลองนี้เน้นกรณีที่ 2 ถ้าเป็นไฟล์ใหม่ ประสิทธิภาพของโมเดลการตรวจจับโค้ดอันตรายเป็นอย่างไร โดยใช้ตรวจชี้วัดคือ ร้อยละความถูกต้อง (accuracy) ร้อยละของความไว (sensitivity) และร้อยละของความจำเพาะ (specificity)

ตารางที่ 4.5 ตัวอย่างการคำนวณร้อยละความถูกต้อง ร้อยละความไว และร้อยละความจำเพาะ

จำนวนโค้ด		สภาพความจริง	
		โค้ดอันตราย	โค้ดปกติ
ผลลัพธ์ที่ได้จาก	โค้ดอันตราย	TP	FP

โมเดล	โค้ดปกติ	FN	TN

สูตรคำนวณตัวชี้วัดมีดังนี้

ร้อยละความถูกต้อง $\frac{(TP+TN)*100}{TP+FP+FN+TN}$

ร้อยละความไว $\frac{TP*100}{TP+FN}$

ร้อยละความจำเพาะ $\frac{TN*100}{FP+TN}$

การทดลองนี้เน้นกรณีที่ 2 ถ้าเป็นไฟล์ใหม่ ประสิทธิภาพของโมเดลการตรวจจับโค้ดอันตรายเป็นอย่างไร โดยใช้ตรวจชี้วัดคือ ร้อยละความถูกต้อง (accuracy) ร้อยละของความไว (sensitivity) และร้อยละของความจำเพาะ (specificity)

สำหรับข้อมูลที่ใช้ในการทดลองแบ่งเป็น 2 กลุ่ม คือ กลุ่มเทรนและกลุ่มเทสต์

1) กลุ่มเทรนหมายถึงกลุ่มไฟล์ข้อมูลที่ใช้ในขั้นตอนการเรียนรู้เพื่อสร้างโมเดลการแยกแยะโค้ดอันตรายจากโค้ดปกติ
2) กลุ่มเทสต์หมายถึงกลุ่มไฟล์ข้อมูลที่ไม่ได้ใช้ในการสร้างโมเดล

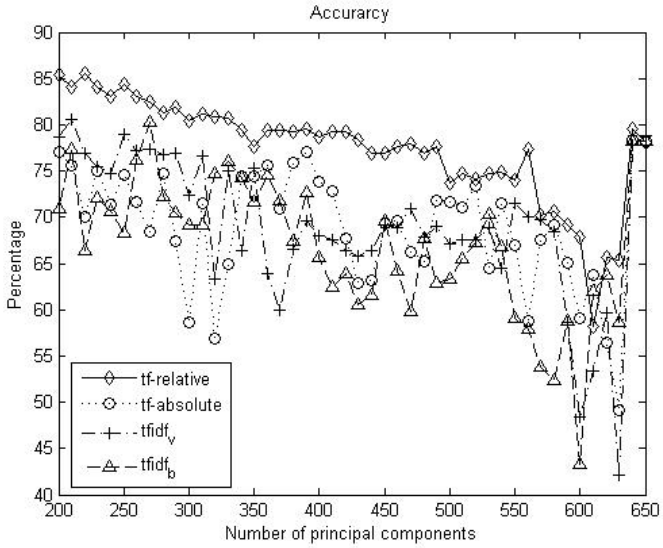
เพื่อให้ทุกไฟล์อยู่ในกลุ่มเทสต์ คณะผู้วิจัยจึงใช้หลักการ crossvalidation แบบ k=10 ทำการทดลองสร้างโมเดลทั้งหมด 10 รอบ โดยที่แต่ละครั้งใช้กลุ่มข้อมูลที่ประกอบด้วยไฟล์จำนวน 90% สำหรับกลุ่มเทรน และ 10% สำหรับกลุ่มเทสต์ และมีเงื่อนไขว่าในแต่ละรอบจะต้องได้กลุ่มเทสต์ที่แตกต่างกันทั้ง 10 รอบ

4.4 ผลการทดลอง

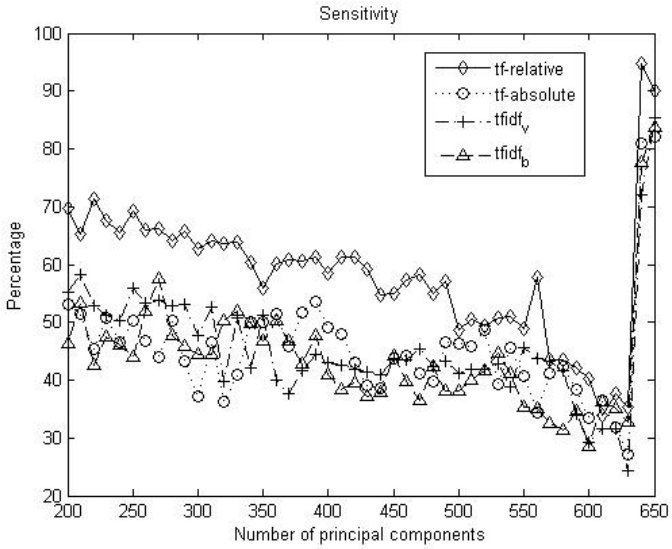
ผลการทดลองแสดงในรูปแบบกราฟที่เปรียบเทียบระหว่างฟีเจอร์ทั้ง 4 แบบ (อธิบายในบทที่ 3) คือ

1) ความถี่สัมพันธ์ (relative term-frequency: tf-relative)
2) ความถี่สัมบูรณ์ (absolute term-frequency: tf-absolute)
3) ความถี่สัมบูรณ์คูณด้วยความถี่ผกผันอ้างอิงฐานข้อมูลโค้ดอันตราย ($idftf_v$)
4) ความถี่สัมบูรณ์คูณด้วยความถี่ผกผันอ้างอิงฐานข้อมูลโค้ดปกติ ($idftf_b$)

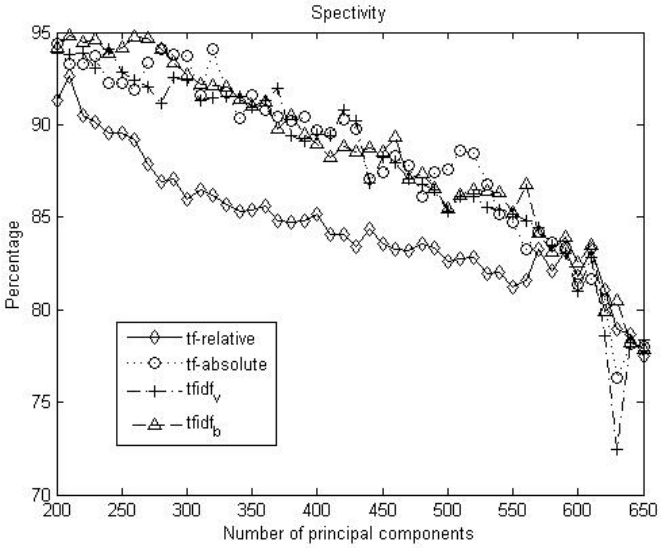
แกนตั้งของกราฟแสดงร้อยละความถูกต้อง ความไว และความจำเพาะดังแสดงในภาพที่ 4.3 4.4 และ 4.5 ตามลำดับ แกนนอนของภาพทั้งสามแสดงจำนวนองค์ประกอบหลักที่ได้จากการลดจำนวนมิติของข้อมูล (อธิบายในบทที่ 3)



ภาพที่ 4.3 ร้อยละความถูกต้องของการจำแนกโค้ดอันตราย แกนตั้งแสดงร้อยละ แกนนอนแสดงจำนวนองค์ประกอบ (component)



ภาพที่ 4.4 ร้อยละความไวของการจำแนกโค้ดอันตราย แกนตั้งแสดงร้อยละ แกนนอนแสดงจำนวนองค์ประกอบ (component)



ภาพที่ 4.5 ร้อยละความจำเพาะของการจำแนกโค้ดอันตราย แกนตั้งแสดงร้อยละ แกนนอนแสดงจำนวนองค์ประกอบ (component)

5. อภิปรายและสรุปผลการวิจัย

โครงการวิจัยนี้มุ่งพัฒนาแบบจำลองการตรวจจับโค้ดอันตรายโดยการวิเคราะห์สายอักษรจากไบนารีโค้ดของโปรแกรมบนเครื่องมือถือ และใช้เทคนิคการรู้จำเครื่องในการรู้จำกลุ่มของโค้ดอันตราย โดยคณะผู้วิจัยได้เสนอฟีเจอร์ที่เหมาะสมสำหรับการตรวจจับโค้ดอันตรายที่ยังไม่ได้บันทึกไว้ในฐานข้อมูลได้ร้อยละความถูกต้อง **85.52** ในขณะที่ร้อยละค่าความไว และร้อยละความจำเพาะมีค่าเท่ากับ **71.26** และ **90.52** ตามลำดับ ซึ่งมีความหมายว่า แบบจำลองมีความไวในการตรวจจับโค้ดอันตราย **71.26** และถ้าโค้ดเป็นโค้ดปกติแบบจำลองจะระบุว่าโค้ดปกติได้ถูกต้องถึง **90.52** (ดังแสดงในการทดลองบทที่ **4)**

จากการทดลองพบว่าฟีเจอร์ที่เหมาะสมที่สุดสำหรับแบบจำลองนี้ คือ ความถี่สัมพันธ์ของการจัดกลุ่มแกรมสาม และการใช้เทคนิคการวิเคราะห์องค์ประกอบหลักเพื่อลดจำนวนกลุ่มแกรมสาม ส่งผลให้แบบจำลองมีประสิทธิภาพในด้านเวลาในการประมวลผล โดยจำนวนองค์ประกอบหลักที่เหมาะสมที่สุดเท่ากับ **210**

หลังจากที่ได้แบบจำลองการตรวจจับโค้ดอันตรายแล้ว คณะผู้วิจัยได้พัฒนาซอฟต์แวร์การตรวจจับโค้ดอันตรายบนสถาปัตยกรรมแบบระบบรับและให้บริการ ซึ่งมีคุณสมบัติดังนี้ **1)** การสแกนโค้ดอันตรายทำได้ทั้งแบบ **online** และ **offline** **2)** การสแกนแบบ **offline** สำหรับโปรแกรมที่มีการบันทึกในฐานข้อมูลแฮชโค้ดบนเครื่องมือถือ **3)** การสแกนแบบ **online** สำหรับโปรแกรมที่ยังไม่มีการบันทึกในฐานข้อมูลแฮชโค้ดบนเครื่องมือถือ **4)** การอัพเดตฐานข้อมูลแฮชโค้ดบนเซิร์ฟเวอร์ก็ต่อเมื่อมีโปรแกรมใหม่ที่สแกนแบบ **online** โครงการวิจัยนี้ได้รวบรวมโค้ดอันตรายบนระบบปฏิบัติการแอนดรอยด์ไว้ในฐานข้อมูลจำนวน **304** โค้ด และจำนวนโค้ดของแอปพลิเคชันจำนวน **500** โค้ด จำนวนโค้ดของแอปพลเคชันที่มากับระบบ **53** โค้ด โดยโค้ดเหล่านี้ได้จัดทำเป็นฐานข้อมูลแฮชโค้ดบนเครื่องมือถือแล้ว

การประเมินความพึงพอใจของผู้ใช้ที่มีต่อซอฟต์แวร์การตรวจจับโค้ดอันตรายพบว่า ผู้ใช้ส่วนใหญ่มีความพึงพอใจมากในเรื่องการติดตั้งและการเริ่มใช้งาน และผู้ใช้ให้คะแนนความสำคัญมากในเรื่องการใช้งานง่ายและค่าใช้จ่ายในการใช้ซอฟต์แวร์ ผลการประเมินนี้พบว่าผู้ใช้โดยส่วนใหญ่ไม่มีโปรแกรมแอนติไวรัสบนเครื่องมือถือแสดงให้เห็นว่าผู้ใช้อาจจะไม่ได้ตระหนักถึงภัยอันตรายในโลกไซเบอร์ ดังนั้นผู้ใช้ส่วนใหญ่จึงเลือกที่จะไม่ทำธุรกรรมธนาคารบนมือถือ

6. ข้อเสนอแนะ

ผลลัพธ์ของโครงการวิจัยนี้คือซอฟต์แวร์การตรวจจับโค้ดอันตรายที่เน้นการคิดค้นวิธีการตรวจจับโค้ดอันตรายที่ไม่มีในฐานข้อมูล จากข้อสรุปข้างต้นเห็นได้ว่า ซอฟต์แวร์ที่ได้นี้มีแนวโน้มในการนำออกสู่ตลาดได้จริงแต่ยังต้องใช้เวลาเพื่อปรับปรุง

และพัฒนาในด้านประสิทธิภาพด้านการออกแบบการใช้งานเพื่อให้ตรงกับความต้องการของผู้ใช้ และการทดลองใช้งานเพื่อนำข้อเสนอแนะของผู้ใช้มาพัฒนาเพื่อสามารถพัฒนาในเชิงพาณิชย์ได้

7. กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จได้ด้วยทุนสนับสนุนจากสำนักงานคณะกรรมการวิจัยแห่งชาติ (วช.) ในปีงบประมาณ 2556

8. เอกสารอ้างอิง

[1] 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.

[2] Aull, Kenneth W., Mark A. Bellmore, and William E. Freeman. "Secure identity and privilege system." U.S. Patent 8,086,867, issued December 27, 2011.

[3] Barbará, Daniel, Julia Couto, Sushil Jajodia, and Ningning Wu. "ADAM: a testbed for exploring the use of data mining in intrusion detection." ACM Sigmod Record 30, no. 4 (2001): 15-24.

[4] Bergeron, Jean, Mourad Debbabi, Jules Desharnais, Mourad M. Erhioui, Yvan Lavoie, and Nadia Tawbi. "Static detection of malicious code in executable programs." Int. J. of Req. Eng 2001 (2001): 184-189.

[5] Di Cerbo, Francesco, Andrea Girardello, Florian Michahelles, and Svetlana Voronkova. "Detection of malicious applications on android os." In Computational Forensics, pp. 138-149. Springer Berlin Heidelberg, 2011.

[6] Dillon, Andrew, and Michael G. Morris. "User acceptance of new information technology: theories and models." (1996).

[7] Enck, William, Machigar Ongtang, and Patrick Drew McDaniel. "Understanding Android Security." IEEE security & privacy 7.1 (2009): 50-57.

[8] Hodges, Vernon, and Shawn O'donnell. "Method and system for providing automated updating and upgrading of antivirus applications using a computer network." U.S. Patent 6,269,456, issued July 31, 2001.

[9] Jolliffe, Ian. Principal component analysis. John Wiley & Sons, Ltd, 2005.

[10] Joachims, Thorsten. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. No. CMU-CS-96-118. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1996.

[11] Kolter, Jeremy Z., and Marcus A. Maloof. "Learning to detect malicious executables in the wild." Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2004.

[12] Lee, Wenke, and Salvatore J. Stolfo. "Data mining approaches for intrusion detection." Usenix Security. 1998.

[13] Leelapongprasut, Panan, Prasong Praneetpolgrang, and Natsapun Paopun. "A quality study of internet banking in Thailand." In Proceedings of the fourth international conference on ebusiness, Samutprakarn, Thailand, pp. 61-65. 2005.

[14] Maimon, Oded Z., and Lior Rokach, eds. Data mining and knowledge discovery handbook. Vol. 1. New York: Springer, 2005.

[15] Nolan, Godfrey. Decompiling Android. Apress, 2012.

[16] Pintsov, Leon A. "Postage payment system with security for sensitive mailer data and enhanced carrier data functionality." U.S. Patent 5,586,036, issued December 17, 1996.

[17] Prates, Raquel O., Clarisse S. de Souza, and Simone DJ Barbosa. "Methods and tools: a method for evaluating the communicability of user interfaces." interactions 7, no. 1 (2000): 31-38.

[18] Sayed, Samir, Rania R. Darwish, and Sameh A. Salem. "A Real-Time Approach for Detecting Malicious Executables." Advances in Systems Science. Springer International Publishing, 2014. 355-364.

[19] Schultz, Matthew G., et al. "Data mining methods for detection of new malicious executables." Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on. IEEE, 2001.

[20] Sookhanaphibarn, Kingkarn, and Chidchanok Lursinsap. "A new feature extractor invariant to intensity, rotation, and scaling of color images." Information Sciences 176, no. 14 (2006): 2097-2119.

[21] Tractinsky, Noam, A. S. Katz, and Dror Ikar. "What is beautiful is usable." Interacting with computers 13, no. 2 (2000): 127-145.

[22] Von Solms, Rossouw, and Johan Van Niekerk. "From information security to cyber security." Computers & Security 38 (2013): 97-102.

[23] Wu, Dong-Jie, et al. "Droidmat: Android malware detection through manifest and API calls tracing." Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on. IEEE, 2012.