Selina Narain, Neelam Boywah, Zoya Haq
DTSC 870 - Masters Project - Fall 2023
Advisor: Professor Dr. Wenjia Li

Progress Report 3

Timeline: October 18, 2023 - November 1st, 2023

***Accomplishments*:** What did you accomplish?

Research Topic Idea:

- Comparing machine learning algorithms for accuracy and efficiency in detecting malware in android applications.

Research Paper Summary on Adversarial Attacks:
EAGLE: Evasion Attacks Guided by Local Explanations against Android Malware Classification

- Eagle: an attack that has customized and specific features that can evade Android malware classifiers trained on arbitrary count features.
- Eagle does not require transplant gadgets with good features that can separate benign and malicious android applications from the attacker.
- The two operations that Eagle uses is to either increase or decrease the values of important features identified by local explanations.
- Examples of count features that can be used:
    - Permission classifier: Manifest files extracted are used as Boolean features to train Android malware classifiers.
    - N-gram opcode classifier: sequences extracted from disassembled Android Application. N-gram opcode classifiers train prediction models from counts or frequencies of any sequences.
    - API calls classifier: Detects and reveals malware behaviors in suspicious Android applications from API calls,
    - Function Call Graphs (FCGs): Contains structural information when computing the hash for each node.
- Classification models used with Eagle: Random Forest and Multilayer Perceptron (MP).
- Evasion attacks against malware classifiers:
    - Attack method based on binary features was applied where the misclassification rate ranged from 63-69% on the Drebin dataset.
    - EAGLE Attacks are model-agnostic

    - EAGLE Attacks can be performed on android malware classifiers with arbitrary count features and are not limited to binary features.
    - EAGLE attacks have higher successful misclassification rates than the results.
    - There are limitations to the Pierazzi approaches. They can be used to evade Android malware classifiers with arbitrary count features.
    - EAGLE attacks treat the target classifier as a blackbox, eliminating the need to know its internal parameters.
    - EAGLE attacks do not require a significant number of benign Android applications to extract gadgets with benign features.

Article: **What is Adversarial Machine Learning? Attack Methods in 2023**

- Adversarial Attacks: methods that are applied to machine learning models that are specifically designed to cause the model to malfunction and make a mistake.
- Whitebox attack - the attacker has access to the architecture system, target model and parameters.
- Blackbox attack - The attacker does not have access to the model or system and only sees the target model outputs.
- Adversarial attacks are main classified as:
    - Poisoning Attacks: The training data and labels are affected by the attacker to cause the model to underperform. Malicious samples may be inserted into the sample to interrupt the re-training data.
    - **Evasion Attacks:** The attacker manipulates and confuses the data to deceive the trained classifiers during deployment. More common and prevalent.
    - Model Extraction Attacks: The attacker extracts data or reconstructs the model using the black box method.
- Popular Adversarial Attack Methods:
    - Limited-memory BFGS (L-BFGS)
    - FastGradient Sign method (FGSM)
    - Jacobian-based Saliency Map Attack (JSMA)
    - Deepfool Attack
    - Carlini & Wagner Attack (C&W)
    - Generative Adversarial Networks (GAN)
    - Zeroth-order optimization attack (ZOO)

Adversarial Attack Algorithms:
C&W (Carlini & Wagner Attack)*:* algorithm that is refined and modified to look for an adversarial malware sample by utilizing an optimization function and does not have box constraints.

$$f(X) = max(Z(X)_t - max\{Z(X)_i : i \neq t\}, -\kappa)$$

JSMA (Jacobian Saliency Map Attack): algorithm that uses feature selection to minimize number of features and forward derivatives of classifiers to find adversarial malware samples.

$$J_F(A) = \left[\frac{\partial F(X)}{\partial X}\frac{\partial X}{\partial A}\right] = \left[\frac{\partial F_j(X)}{\partial x_i}\frac{\partial x_i}{\partial a_i}\right]_{i \in 1...n, j \in 0,1}$$

Tools for Adversarial Attack Techniques:
Adversarial Robustness Toolbox (ART): A Python library that supplies tools to help researchers and developers to evaluate, defend, certify, and verify ML models against threats (Evasion, Poisoning, Extracting, and Inference) and malware.
https://adversarial-robustness-toolbox.readthedocs.io/en/latest/

cleverhans: An adversarial example library for constructing attacks, building defenses, and benchmarking the vulnerability of machine learning models to adversarial examples:
https://github.com/cleverhans-lab/cleverhans
https://cleverhans.readthedocs.io/en/v.2.1.0/source/attacks.html

Tools for APK's:
AndroZoo: Android Applications collected from several sources, including the official Google Play app market. Contains 23,536,010 different APKs.
https://androzoo.uni.lu/

Apktool: A tool for reverse engineering Android apk files
https://github.com/iBotPeaches/Apktool

Google Collab - MalwareDetection.ipynb

- Set up environment for testing and implementation
- Imported necessary libraries
- Mounted google drive

- Reading Kaggle dataset files
- Simple Data Visualizations

Google Collab - DataPreprocessing.ipynb

- Used [Android Malware Dataset (CIC-AndMal2017)]
- This source has multiple datasets for each type of malware category, Adware, Ransomware, Scareware, SMS Malware as well as 10 samples for each family of malware.
- For instance, Adware has the following families; Dowgin, Ewind, Feiwo Gooligan, Kemoge, koodous, Mobidash, Selfmite, Shuanet, Youmi
- We combined the dataset from each of the families into an overarching data for each of the malware categories
- We also had to clean some of the columns of the dataset so it would be easier to pull later on for our models
- Once we combined the families and cleaned up the columns, we ended up having the following counts for each of the categories:

```
[ ]  df['Label'].value_counts()

BENIGN        1205515
ADWARE         424147
SCAREWARE      400841
RANSOMWARE     348943
SMSMALWARE     237133
Name: Label, dtype: int64
```

-
- In total, there are 84 features which we would need to reduce through feature selection.

Upcoming Plan: What do you plan to do in upcoming weeks?

- Finalize APK tools
- Further conduct data analysis on datasets
- Attempt to create the structure for our SVM Model
    - Feature Extraction and Selection (FEST)
    - Kernels (Non-linear, linear and RBF)
- Attempt to create Random Forest Model
- We are still currently looking into C&W, JSMA and other adversarial attacks and how they can be implemented into our methodology. We are attempting to implement at least one of them in our testing and implementation.

***Obstacles & Concerns*:** Were there any obstacles or barriers that prevented you from getting things done?

For the adversarial attacks we are further looking into Adversarial Robustness Toolbox and cleverhans to help guide us in implementing one of them however, we are still exploring how to use the tool. In order to extract features from manifest files in APK's, we looked into AndroZoo. However, we need to apply first to gain access to the APK's. We have obtained a Drebin dataset from Kaggle and have implemented it into the code; however, we would like to compare the Drebin dataset that you may have to see which is best fitted for our project. Also, for the Android Malware Dataset (CIC-AndMal2017), there was some difficulty with deciding how to manage the datasets that were provided since the source separated the data for each family within the malware category. Whereas we were looking for the malware categories themselves rather than specific families within each category.