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

Progress Report 1

Timeline: September 20, 2023 - October 4, 2023

Accomplishments: What did you accomplish?

- Summarize research papers.

Selina Narain

Detecting Malware for Android Platform: An SVM based Approach

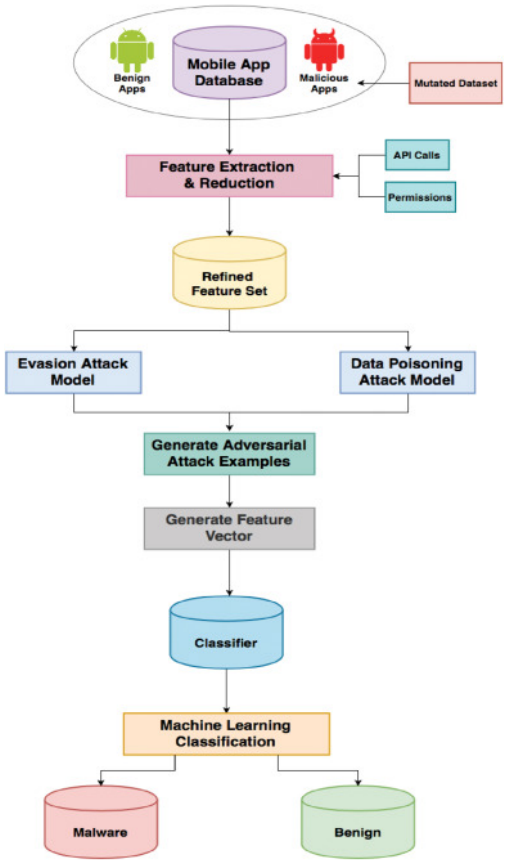
- SVM-based approach, integrating both risky permission combinations and vulnerable API calls - using as features in the SVM algorithm
- One study revealed popular security software tools detection rate was 20.2% to 79.6%
- To perform tasks - each Android app has to explicitly request the corresponding permission from the user during the installation process
- Google "Bouncer" released to automatically scan apps - can only scan apps for limited time so malicious apps can bypass them by being stagnant during the scan phase.
- Drebin - extracts permissions, APIs and IP address as features and uses SVM algorithm to learn a classifier from the existing ground truth datasets, which can then be used to detect unknown malware. But it takes a long time due to the high dimensional feature vector. *ML*
- DroidMat - applies KNN algorithm to permissions to identify malware. However, low recall value *ML*
- DoridAPIMiner provides several lightweight classifiers based on the API level features. However, high accuracy due to skewed dataset towards more benign than malware *ML*
- SVM-based approach - first calculate the similarity scores between malware and benign applications in terms of suspicious API calls, use similarity scores as features, then use risky permission requests as additional features when training SVM classifiers. *ML*
- Static program analysis - methods like code disassembly, decompilation, and data flow tracking, with tools (Kirin, Stowaway, RiskRanker, Smali, Androguard, AndroidLeaks, etc) to identify malicious activity, over-privileged apps, security risks, privacy leaks, and sensitive information leakage
- Dynamic analysis - monitoring app behavior during execution, with tools like TaintDroid and DroidScope, using schemes (AppsPlayground, DroidRanger) to scan and analyze
- Schema - decompilation (decoded into a readable smali file), feature extraction (risky permissions, suspicious API calls, and URLs), and classifier (SVM).
- Weight Calculation Using TF-IDF (Term Frequency-Inverse Document Frequency) - analyze the smali files and calculate the weight of every dangerous API in the feature vector

- Results: around 86% accuracy using SVM to detect combination dangerous API calls and risky permissions

Malware Detection in Adversarial Settings: Exploiting Feature Evolutions and Confusions in Android Apps

- Malware Recomposition Variation (MRV) - conducts semantic analysis of existing malware to construct new malware variants for malware detectors to test and strengthen detection signatures/models.
- Used two variation strategies (i.e., malware evolution attack and malware confusion attack). Follows structure of existing malware for accuracy.
- Perform program transplantation to automatically mutate malware bytecode to generate new malware variants
- Constraints to craft realistic malicious attack:
- (1) Preserve malicious behaviors and maintaining original malicious purposes
- (2) Mutated malware should be robust enough to be installed/executed on mobile devices
- (3) To evade malware detectors - malware identifies features and computes feature values that can evade detection without breaking the malware.
- Defense mechanisms - (1) Adversarial Training: training new model with the combination of the generated malware variants and original training data. (2) Variant Detector: Developing detector in addition to original malware detector to detect whether an app is a variant derived from existing malware. (3) Weight Bounding. bounding the feature weights in the original malware detector to make feature weights evenly distributed.
- RLTD feature model covers: *Resource* (what security-sensitive resources malicious behaviors obtain), *Temporal* (when the malicious behaviors are triggered), *Locale* (where the malicious behaviors occur), *Dependency* (how the malicious behaviors are controlled).
- Evolution attack - instead of developing targeted malware to evade specific detection techniques, the defeating mechanism: mimicking and automating the evolution of malware
- Result: MRV produces malware variants that can have high likelihood to evade detection while still retaining malicious behaviors

DroidEnemy: Battling adversarial example attacks for Android malware detection



- Data poisoning attack, the exploratory attack, and the evasion attack
- Developed 2 attack models based on the data poisoning attack and the evasion attack to generate adversarial examples - focus on mutating different Android application features, such as API calls, permissions and the class label, to feed the classifier.
- SVM algorithm and kernel functions such as linear and Radial Basis Function (RBF)
- Performed experimental study on real Android application dataset.
- Results: Evasion attack - classifier has 100% incorrectly classified instances and 0% correctly classified instances. Data poisoning attack - classifier has 74.85% correctly classified instances and 25.14% incorrectly classified instances.
- Evasion attack causes more performance degradation to the malware detection system than the data poisoning attack.
- Static analysis based approaches: identify issues and flaws in mobile apps without executing them; ex: AndroidLeaks
  - Limitations: large quantities of mobile applications, cannot recognize security susceptibilities
- Dynamic analysis-based approaches: tracks any suspicious activity that utilizes unwanted third party libraries to leak confidential and private data. Ex: DroidScope
- Machine learning-based approaches: SVM applied with feature extraction and selection tool (FEST) for detecting Android malware based on the frequency of the features found in various malware samples.
- Generated the data poisoning attack using RevealDroid and IagoDroid - algorithm runs the attack on the features.

Zoya Haq

Drebin: Effective and Explainable Detection of Android Malware in Your Pocket

- Drebin is a method for detecting android malware.
- Does broad static analysis which gathers any features from an application's code and manifest.
- Effective detection, results that are explainable since the patterns of features indicative of malware detection can be traced back from vector space. There is also lightweight analysis which enables detecting malware on smartphones and analyzing large sets of apps in reasonable time.
- With broad static analysis, Drebin statistically inspects a given Android application while extracting different feature sets; then embedded in a vector space for geometric analysis; feature sets embedding enables malware identification (SVM)
- Manifest features (AndroidManifest.xml) gives data that supports the installation and execution.
- There are hardware components, requested permissions, components within the app and filtered intents which is the communication process.
- Android apps are developed in Java and compiled into bytecode for Dalvik VM which provides Drebin info regarding API calls and data.
- Restricted API calls: the permission system for Android restricts access to critical API calls. Malwrare can use root exploits to go past the limitations thats are put in by the Android.
- The used permissions are the calls extracted from the restricted API as means of determining the set of permissions needed and used.
- Suspicious API calls allow access to sensitive data.
- Network addresses: malware constantly establishes network connections to retrieve commands or exfiltrate data that is ultimately collected from the device.
- ML techniques are used to learn the separation between malicious and benign applications.
- SVM is used which determines a hyperplane separating both of the classes with a maximal margin.
- Detection systems need to not only detect the malicious activity but must also give an explanation where Drebin addresses the problem and then uses learning-based detection to give the features that contributed to it.
- Through SVM we can extract the top *k* features by weights.
    - Ex: “app uses hardware camera”
- Evaluating Drebin thought detection performance, explainability, and run-time performance.
- Limitations of Drebin: the quality of Drebin depends on the availability of malicious and benign applications and that requires technical effort. Another limitation is the use of ML because of the possibility of mimicry and poisoning attacks.
- There is detection using Static and Dynamic analysis, and ML.
- Android malware is still relatively new and there is such diversity in malware that many of the anti virus scanners are unsuccessful in identifying various malicious attacks.

AdversarialDroid: A Deep Learning based Malware Detection Approach for Android System Against Adversarial Example Attacks

- Deep learning based approaches are made to identify malware for the Android system.
- There have been a lot of malware samples every month with aspirations to steal user data.
- Malware detection approaches have been through static analysis, dynamic analysis, and machine learning.
- Dynamic analysis: Detects malware when applications are executed
- ML: allows for smart detection of malware through its algorithms
- With ML, there is a risk for attacks to be benign and pass through the classifiers through adversarial attacks.
- Deep Belief Network (DBN) suffers from adversarial example attacks and through training the classifier, it can work against the attacks
- ML Approach: In a study, a mutated dataset was incorporated to be processes where its features are extracted and reduced.
- Binary n-grams are critical to hoe a model can craft the adversarial examples. The Data gets sent to and from the classifier, it gets trained, and then ready for the attack.
    - Deep learning technique to combat adversarial attacks
- DL Approach: uses neural networks to extrapolate weighted biases from the neurons.
- DBN is composed of Restricted Boltzmann Machines that have the goal to classify data into any category defined by the data that is labeled.
- Adversarial attacks are in place to cause the model to output and calculate errors.
    - Two types: causative and exploratory.
- Exploratory: evasion attack is most common. Has the goal to evade detection from the classifier.
- Dataset: .apk files are needed. APKTools is an open source tool to gather those files.
- Manifest file is the .xml file which is the foundation of the application.
    - Has API calls and uses-permissions
- Data set has more than 500,000 apk's collected from different sources. The extracted data set is in .xlsx form, each row is .apk, each column has features.
- Feature Extractor: Any apk that passes through the system is novel apk (never before seen by system).
- Through python, a DBN was constructed to format the extracted uses-permissions.
- DBN:
    - First write up a reference in an excel file which contains apk's.
    - Data set is introduced in 2D array format
    - Partition the data, instantiate the network (now classifier) to detect how many hidden layers and number of epochs are needed.
    - Data is fitted by classifiers and called in for predictions.
- 4 performance metrics: precision, recall, f1-score, accuracy
- Label flipping: flips the binary label from the dataset to appear benign when actually malicious.
    - The most feasible approach is to take from the labeled data towards the end of the row that tells us the benign status of an apk and then flip certain benign apks.

- Some inaccuracies can be the middle ground threshold which makes it difficult for the classifier to identify benign or malicious status.
- DBN is an older algorithm that can be outperformed by algorithms in today's age.

Adversarial-Example Attacks Toward Android Malware Detection System

- Generative Adversarial Network (GAN) based adversarial-example attacks can defeat the Android malware detection that already exists.
- They can be defended through the use of a firewall.
    - Personal note: There is "Windows Defender" for Microsoft.
- To evade adversarial example detection AND malware detection, the researchers developed an attack method which has a bi-objective GAN.
- 95% of adversarial examples breakthrough the firewall equipped Android Malware detection system which outperforms the state of the art method by 247.68%

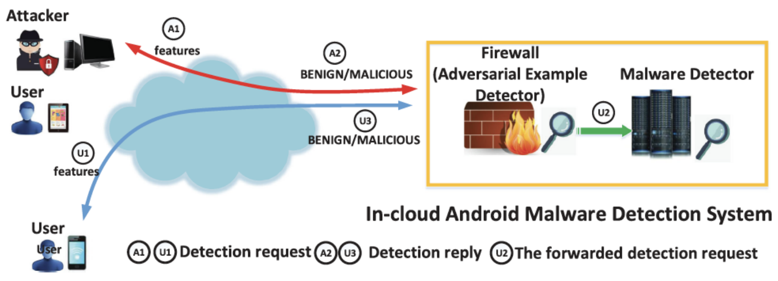

Attacker
User
User
A1 features
U1 features
A2 BENIGN/MALICIOUS
U3 BENIGN/MALICIOUS
Firewall (Adversarial Example Detector)
U2
Malware Detector
In-cloud Android Malware Detection System
A1 U1 Detection request A2 U3 Detection reply U2 The forwarded detection request

Fig. 1. System overview.

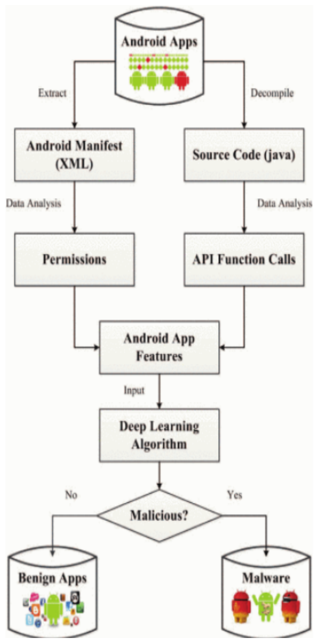
- 97% of all mobile malware targets Android
    - Why?
- Android malware detection is a binary classification problem and uses well trained classifiers to determine whether an app is malicious.
- Clouds are vulnerable to attacks.
- MalGAN - adversarial example towards malware detection.
- GAN - generative adversarial network; doesn't need any information regarding the target.
- This decreases detection accuracy of all classifiers along with SVM, Adaboost, convolutional neural network(CNN) too close to zero.
- According to studies, over 90% of adversarial examples via MalGAN were captured via a firewall.
- Some evaded detection but couldn't get past the firewall and some got past the firewall but not the malware detectors.
- E-MalGAN method (Black-box attack method)used to mislead the firewall in the cloud and doesn't need any information about the target.
- GAN has two deep neural networks: Generator and Discriminator
- Over 95% of adversarial examples via E-MalGAN have successfully avoided both malware and adversarial attack detection.
- Features for malware detection can be static, dynamic, or hybrid.
    - To extract dynamic features. We need to monitor the behavior of apps during runtime.
- Normal vs. adversarial example:

    - Normal: composed of features that are extracted from benign or malicious Android applications.
    - Adversarial: attacker extracted features from a malicious app and combines with other features to mislead detection system
- Defense:
    - Data Collection: detection system collects normal examples by extracting features from benign and malicious apps. Using GAN.
    - Model Training: administering normal and adversarial examples together and training binary classifiers.
    - Adversarial Example Detection: using trained classifiers as firewalls to block adversarial examples.
- Attack:
    - Preliminaries: GAN plays game with Generator and Discriminator
    - Motivation: challenges targeted classifiers.
    - Model and Algorithm: bi-objective GAN.
- Experiment uses Adaboost as the classifier for the malware detector and adversarial example detector.
- E-MalGAN is evaluated via effective generation rate (EGR)
- E-MalGAN performed better than MalGAN in each scenario and is able to handle the two objectives while also achieving a tradeoff between them.
- Experiments indicate that over 95% of adversarial examples generated by the method in the paper are falsely detected as benign through the firewall developed Android Malware detector proving it to be effective.

Neelam Boywah

DroidDeepLearner: Identifying Android malware using deep learning

- Android is an open source with low barriers making it an easy target for malware intrusion. There is also a lack of security when it comes to checking if the apps contain malware or security risks before they're published on the Android platform.
- Static Analysis - inspect and disassemble code before execution, Dynamic Analysis - identifying malware during execution, Machine Learning Techniques - extracting app features to distinguish between malicious apps and benign apps and utilizing feature sets from permissions in the manifest.xml and API calls.
- Knowing which permissions to use when doing Permission Analysis can be tricky, if a set of specific combinational permissions are being requested, there is a high chance that the app can be malicious.
- Deep Learning has its first 2 hidden layers that are RBM (Restricted Boltzmann Machine) and the other hidden layers are SBM (Sigmoid Boltzmann Machine) using the Bayseian form.
- RBM suffers from the Vanishing Gradient Problem; however, DBN (Deep-Belief Network) resolves the problem with the way the data is being trained.
- In the DroidDeepLearner approach, the 2 features, permissions and API function calls are being used. Manifest files (*.xml) → Permissions get extracted. Source Files (*.java) → API function calls gets extracted.
- Once the permissions and API function calls are gathered, they can be integrated into an android app feature set that will be used as an input for the DL training and testing steps.
- The classification result from the DL algorithm will determine whether or not the application is malicious or benign.
- The features (permissions, API's, URLs, actions, intents and IP addresses) are encrypted in the apk file within the apk format of the android app. A decoder is utilized so there can be readable manifest files.
- android.permission.CALL_PHONE, android.permission.SEND_SMS, android.permission.WRITE_EXTERNAL_STORAGE, android.permission.ACCESS_LOCATION, android.permission.READ_CONTACTS are all examples of risky android permissions that heavily used in android malware.
- The Deep Belief Network (DBN) is composed of RBM. The sigmoid function is utilized as an activation function in the RBM model. RBM trains input data and gets values from 1st hidden layers. The 1st hidden layer (visible now) is used to train the second hidden layer and process repeats until all hidden layers are trained. DBN process fine tune - Contrastive Divergence Learning.



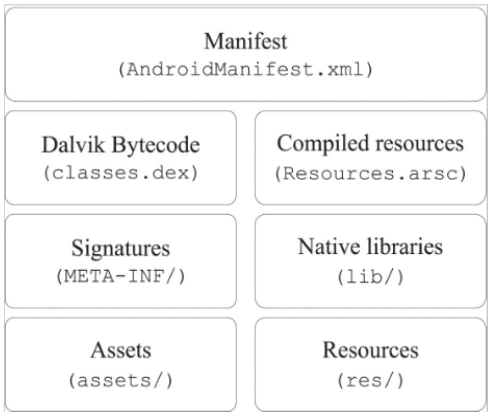
- Datasets were applied to the DBN model (DroidDeepLearner) and multiple versions of SVM models for comparison. Precision (P), recall (R), and F1 score were compared using 10-fold cross validation. DBN performed better than SVM.
- DroidDeepLearner (DBN model) utilizes both features, API calls and combinations of risky permissions to distinguish between malicious and benign applications.

A Robust Malware Detection Approach for Android System Against Adversarial Example Attacks

- Adversarial Examples: Support Vector Machine (SVM), Random Forest, Deep Learning - used to successfully detect malware in android applications.
- Evasion attacks are when mutated malware samples are pushed into a detection system which then outputs falsely classified malicious applications as benign ones.
- Robust Malware Detection approach consists of feature extraction, feature representation, adversarial example generation and detection of malware.
- Static Analysis based approach - identifies defects + issues without execution (malware detection system - RiskRanker) Dynamic Analysis needs to follow behind Static Analysis because security vulnerabilities are not always exposed during run time but dynamic analysis would detect those subtle malware.
- Dynamic Analysis based approach - detecting malware while android apps are executed. Dynamic analysis tracks risky API calls. Malware detection system used - MADAM (captures app behavior and communication patterns by tracking system calls at the kernel level and user activity level. Disadvantage - takes additional time so the abnormal behaviors can be captured.
- Machine Learning based approach - can be better to use when dealing with large amounts of malware samples.
- Support Vector Machine uses feature extraction and selection (FEST) to show the frequency of features that's in different malwares and then the algorithm is able to distinguish between benign and malware applications based on said features.
- Random Forest - in order to improve detection accuracy, parameters such as # of trees, depth of each tree and # of random selected features need to vary.
- Bayesian Classification - uses features like API calls, Linux system commands and permissions for training and testing to detect malware. Deep Learning - DroidDeepLearner
- Adversarial Machine Learning: One example is Data Poisoning Attack which is when ML algorithms are weakened by inserting malicious data into the training set.
- 4 Attack Strategies for Model Alteration: Label Modification, Data Injection, Data Modification and Logic Corruption.
- Exploratory Attack is similar to dynamic analysis by executing while the system is operating. Can be developed using reverse engineering.
- Countermeasures to defend against adversarial attacks: DexGuard, Trivial Obfuscation, String Encryption, Class encryption.
- Process: Feature Extraction & Feature Selection - binary N-grams of API calls to make feature sets based on relevancy, URLs…etc.
- Feature Reduction using TF-IDF to narrow down the feature vector to those most important. The feature vector made using TF-IDF is used to train the classifier. Attack models can help train and strengthen the classifiers to increase accuracy of malware detection. Evasion Attack Models are the most common types of attack used and happens when the classifier is given a manipulated malware sample (adversarial example) that has been wrongly classified as a benign app by the classifier.
- Classifier - Linear SVM tends to be less accurate than Non-linear SVM classifiers.

- Kernel function will be governed by the dataset and type of Evasion attack.

Android HIV: A study of Repackaging Malware for Evading Machine-Learning Detection



- Android applications are packaged in the APK (android application package) files and are encrypted. A decoder is needed to extract features from the APK archive by performing static analysis on the AndroidManifest.xml and dexcode.
- android.permission_SEND_SMS and android.permission.READ_CONTACTS are permissions that are implemented to send text messages to your contacts.
- Features that are extracted from Manifest → vector of binary values. In order to prevent detection from ML detectors, the extracted features for the systems need to look benign by manipulating the malware sample.
- The framework proposed to craft adversarial examples is using static analysis, MaMaDroid and Drebin.
- MaMaDroid's features are taken from CFG (control flow graph) and has 2 modes, family mode and package mode that are utilized for API calls. Once it gets the sequences for the API calls, a Markov chain is constructed so the feature vector can be used in the training step of the ML classifier.
- MaMaDroid is able to recognize all of the families and packages from the app documentation. The ML models RF, KNN and SVM are used to train the malware detection system and test its performance against various datasets.
- Drebin features are taken from manifest and disassembled dexcode. Between manifest and dexcode, 8 sets of features are extracted and put into a multidimensional vector (S).
- Drebin uses the linear SVM classifier to distinguish between malicious and benign applications.
- The feature set, training set and classification algorithm and parameters can vary.
- Attack algorithm - C&W and JSMA (2 adversarial algorithms) are used to calculate the perturbation based on the number of API calls.
- A neural network F (binary classifier) is built to launch the attack. C&W is refined and modified to look for an adversarial malware sample by utilizing an optimization function. JSMA uses forward derivatives of classifiers to find adversarial malware samples.
- Questions trying to solve: Can the modified malware sample effectively evade from the target detector? Can the modification be easily applied to the original APK?
- An apktool is used to decompile the malicious APK input. AdaGrad and multi-layer perceptron (MLP) is implemented in order to train models within the experiment. MaMadroid experiment results involved 3 machine learning classifiers, RF, SVM and KNN (K-Nearest Neighbor). The DNN (Deep Neural Network)
- Evasion Rate (used as a baseline) - ratio of malware samples that are misclassified as bening to the total number of malware samples in the testing set. Distortion - the number of API calls added to the smali code for each malware sample.

- Overall results are being compared between MaMaDroid and Drebin. Drebin misclassification score on the JSMA attack was higher than MaMaDroid's score against all 4 ML models (RF, SVM, KNN, DNN).

General Outline of Research Papers

- Discuss and present literature review on different types of malware detection systems and solutions. (ie. Static analysis, dynamic analysis, machine learning approach)
- Summarize idea/ topic and its issues
- What solution are we presenting?
- How is it different? How is it built? Explaining why certain features are used, why certain functions are set to what they are, why those specific parameters were chosen? What are the results?
- Future plans/ works?

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

- Finalize Research Paper topic
- Review and finalize benign samples and datasets.
- Begin to finalize which Machine Learning algorithms and classifiers will be used in experimentation & testing.
- Continue research.

ML Algorithms based on Research:

Zoya:

- SVM (Based on Drebin Article)
- AdaBoost

Neelam:

- SVM (linear classification)
- SVM (non-linear classification)
- Random Forest - (A Robust Malware Detection Approach…)
- K-Nearest Neighbors (KNN)

Selina:

- SVM algorithm using features such as risky permission combinations and vulnerable API calls
- Adversarial attacks based on support vector classification (DroidEnemy)

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

- We were able to understand the research papers and successfully make a note of the main and key points of our findings, therefore we did not have any obstacles accomplishing this week's tasks.