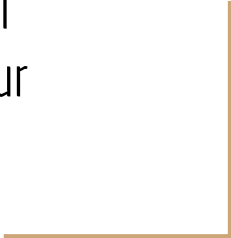




Detecting Malware in Android

Professor Fu
Hani Alshahrani
Harrison Mansour
Seaver Thorn



Outline

- Android background
- Our Project
- Feature Reduction
- Machine Learning
- Current Progress
- Timeline
- This Week
- References

Why ...

SMARTPHONES?

ANDROID?



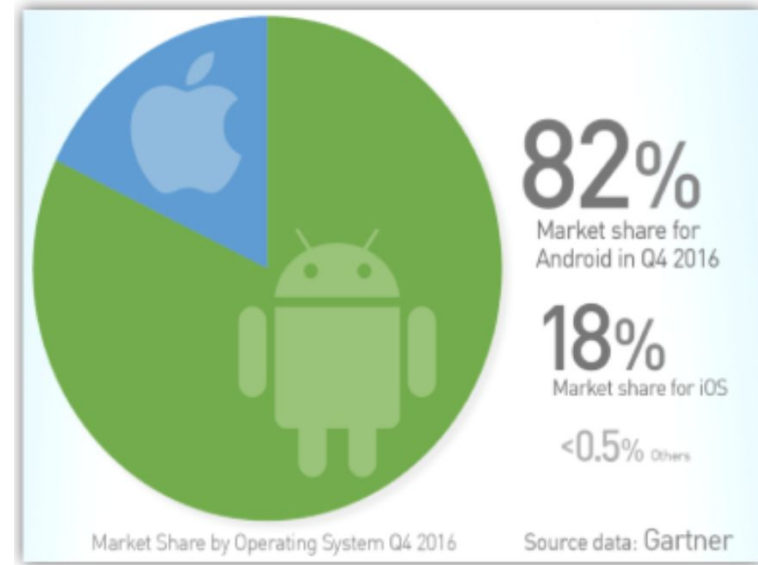
Why Smartphones?

- Just like every other computing device, smartphones are targeted by attackers for their valuable information and internet access.
- The average adult spends 3 hours per day on his/her smartphone.
- The number of unique mobile internet accessors have surged from 5.6 million in 2013 to 12.7 million in 2016.
- There is about 2.2 million apps in the Google Play Store, compared to 2 million in the Apple App Store.



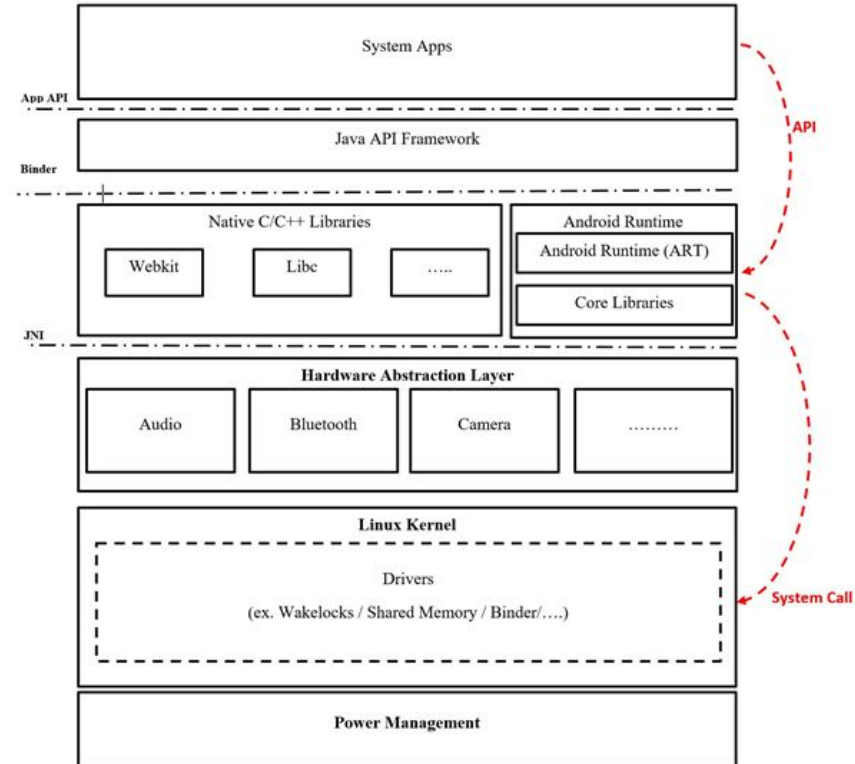
Why Android?

- Android is an open source OS, meaning developers can see and modify the source code without any fees or licenses.
- Android controls more than 80% of the smartphone market in 2016.
- According to a report by Now Secure Lab, 83% of Android apps store data insecurely.



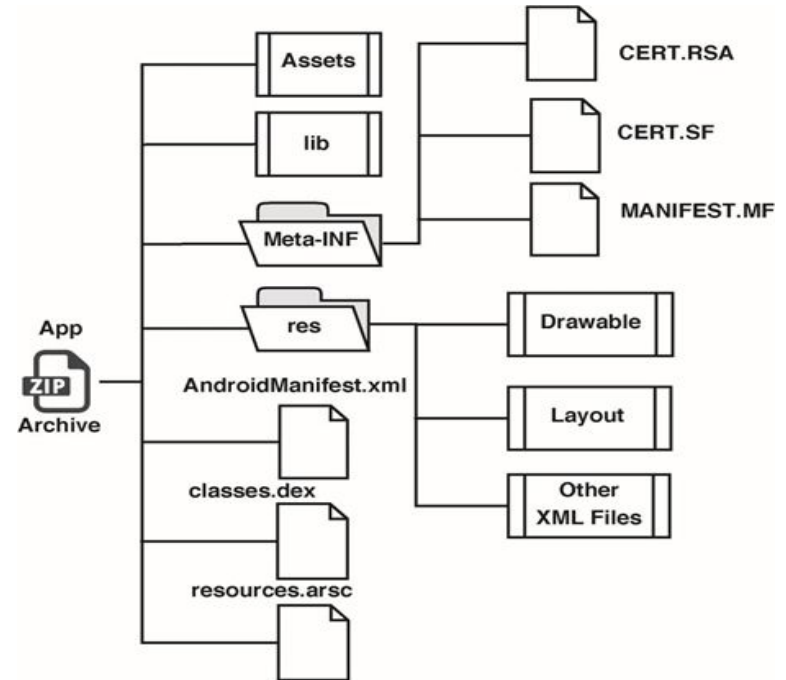
Android Architecture

- Linux kernel
- Native libraries developed in C/C++
- Android runtime
- Java API framework
- Application level framework

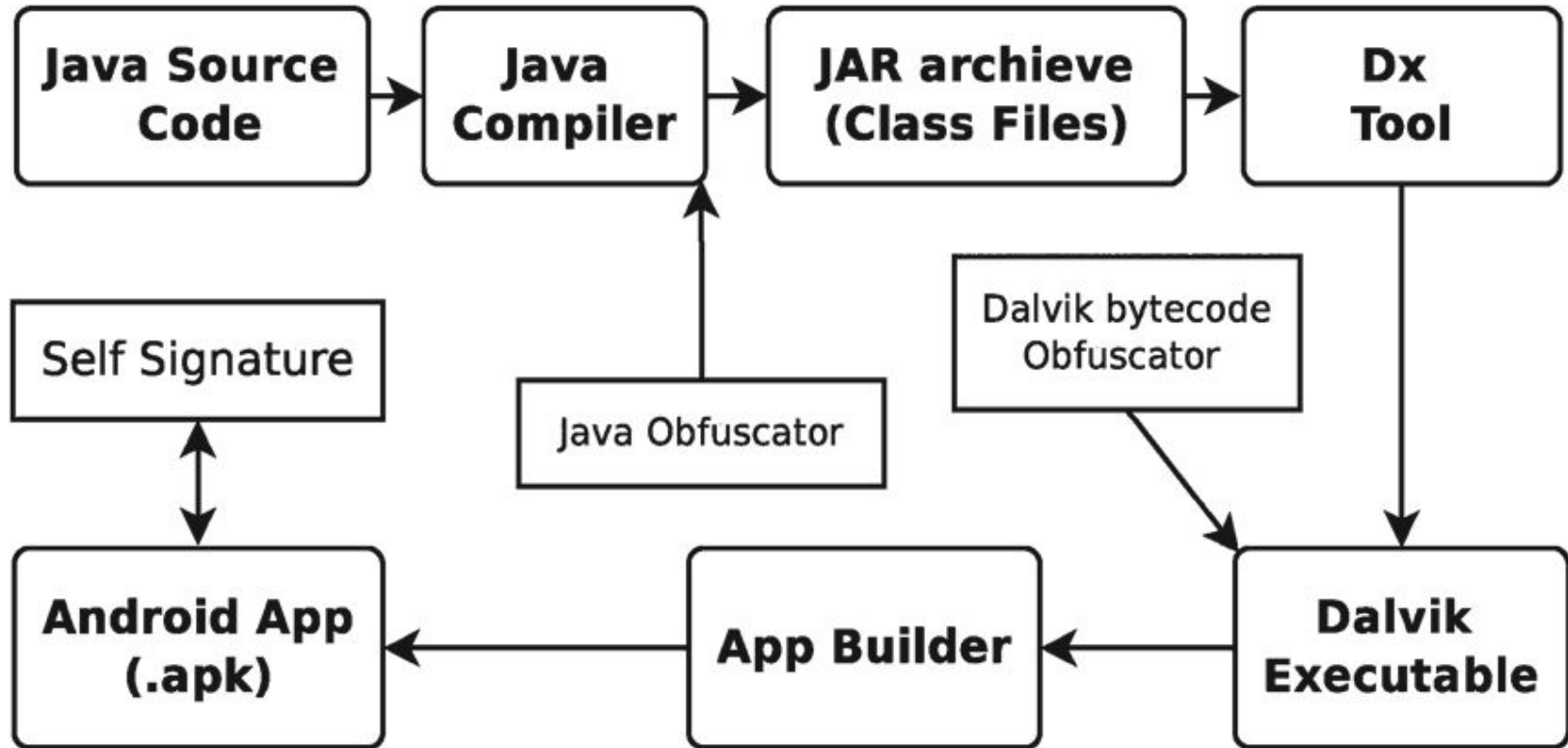


App Structure

- Android applications are packaged into an APK including
 - The AndroidManifest.xml
 - The res folder
 - The assets folder
 - The executable file classes.dex
 - META-INF



The Development Process



Android Application Components

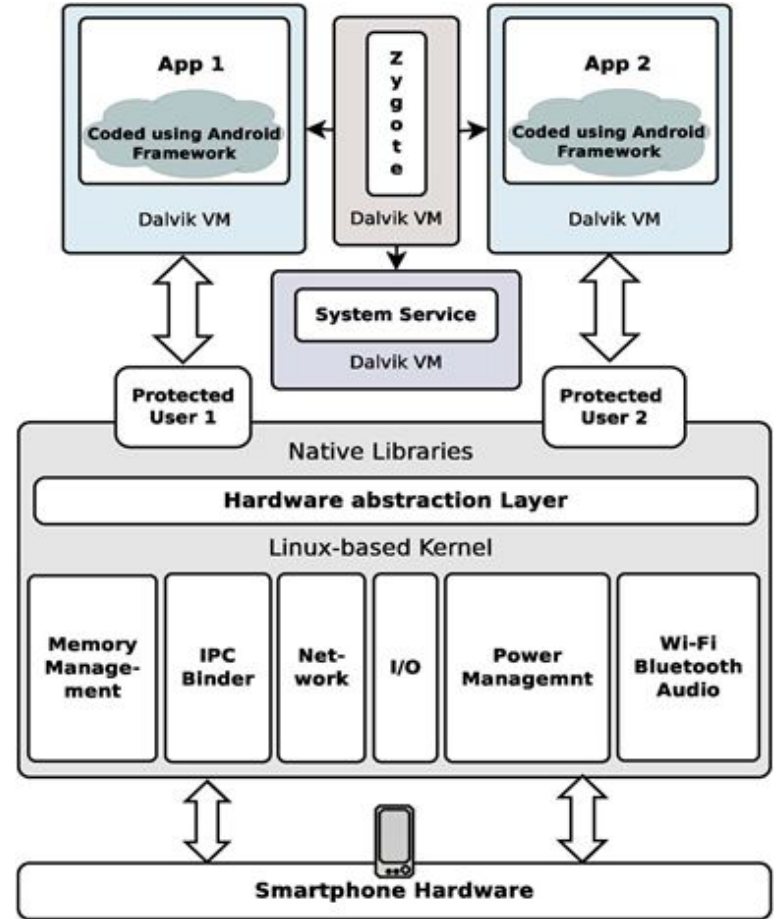
- Android apps are composed of one or more of the following components:
 - Activity- The user interface (UI) component of an app
 - Service- Performs background tasks. Is void of a UI
 - Broadcast Receiver- Listens for events generated by the Android system
 - Content Provider- The data-store. Provides a consistent interface for data access between apps.

Permissions

- Android provides a permission-based security model.
- The developer must declare the permissions required used the uses-permissions tag in the AndroidManifest.xml.
- In order to keep apps under control, the system places each application into its own “sandbox”, where it cannot mingle with others.
- Android permissions are divided into four protection levels:
 - Normal
 - Dangerous
 - Signature
 - SignatureOrSystem

App Sandboxing

- The sandboxing prevents apps from interfering with other apps or system services.
- Each app process is protected with an assigned UID.
- An app must contain a PKI certificate signed with the developer key.
- App signing procedure places an app into an isolated sandbox.
- If 2 apps have the same certificate, they are put in the same sandbox.



Types of Malware in Android

- Information Extraction
 - Steals personal or device information
- Make calls and send SMS to premium numbers
- Root Exploits
 - Takes control of the system and modifies information
- Dynamically Downloaded Code
 - An installed benign application will download a malicious code and deploy it within the device
- Covert Channel
 - Exploits device vulnerabilities to leak information between processes
- Botnets
 - A network of compromised devices controlled by an attacker

Our Project

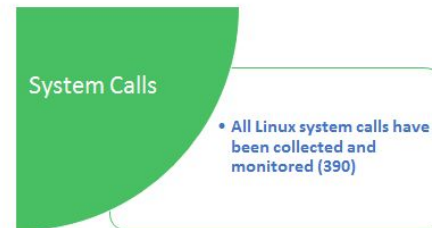
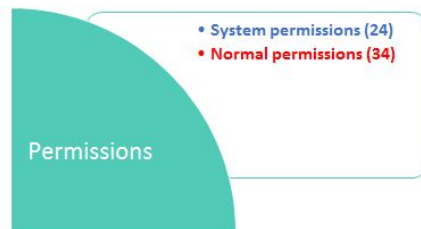
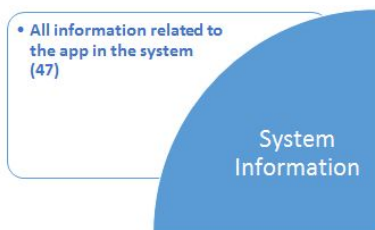
As mobile computing has become more and more pervasive in today's world, so has malware that specifically targets our mobile devices. The newest iterations of malware are smart enough to evade most static analysis techniques and even some dynamic techniques. To combat this, we plan to expand upon the ideas and techniques presented by Andromaly, DroidDetector, MARVIN and DroidScribe, among others, that utilize machine learning techniques to detect malicious applications.

Related Work

Tool	# Features	Algorithm(s)	Deployment	User Report	# apps	#malware	Accuracy
Andromaly	88	K-Means Logistic Regression Histograms Decision Tree Bayesian Networks Naïve Bayes	On-device	yes	20 apps from four categories	4 apps that represent 2 types only of malware	95%
Analyzing app metrics	27	Decision tree learning	Off-device	No	1059 from different categories	1074 apps from Drebin dataset	99.52%
Detection using system calls	196	(AROW) (KNN) Naïve Bayes Decision Tree Random Forest SVM	Off-device	No	1189 trusted applications.	1227 malicious applications	97.7%
<i>DroidDetect or</i>	192	Deep learning	Off-device	yes (online)	20000 trusted applications.	1760 malicious applications.	96.76%
DroidScribe	254	SVM	Off-device	No	-----	5560 malicious applications.	94%
DroidSieve	+320	Extra Tree	Off-device	No	139028 trusted applications.	31950 malicious applications.	99.82 %
MARVIN	+490000	SVM and Linear classifier	Analysis off-device . But user can submit an app for analysis through an app.	Yes	135000 trusted applications.	15000 malicious applications.	98.24%

Our Improvements

- We are running apps on-device, and therefore do not face any of the limitations that are inherent with the use of emulators.
- Focuses on System calls, permissions, and other hardware features.



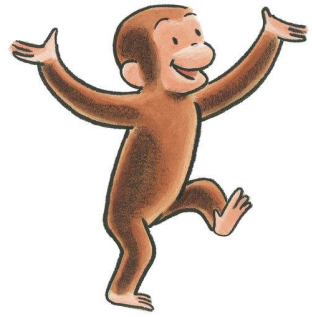
Our Project

- Collect a feature set of benign and malicious applications via on-device dynamic analysis.
- Design a machine learning framework that effectively utilizes these features to identify malicious applications.
 - Try many different machine learning techniques and pick the best
 - SVM
 - Neural Network
 - Logistic Regression
 - etc...
- Teach our framework with the feature vectors gathered.
- Using MARVIN and Drebin datasets for a total of 19,060 apps.

Our Android App

- Computer works with device to automate the process of installing new apps, and monitoring them.
- Runs in the background on the device, and checks what apps are doing
 - To obtain system call information
 - Battery life, cpu, memory used.
 - Network information
 - App permissions, and developer certificate.
- Sends results to a database, machine learning algorithms will use this information for training, testing, and validation techniques.

Collecting data on an Android device



- 5 Android tablets ran for about a week testing our dataset of many different apps.
- We used the monkey tool to test this.
 - Developed by Google
 - Used by many app developers to test their apps.
 - Generates random events and sends it to the app to simulate a “monkey” using it.
 - We sent 500 events to each app (about 60 seconds to run) for simulation.
- Our app, DDefender, scanned each newly installed app and monitored:
 - Permissions requested
 - Linux System information
 - Linux System Calls

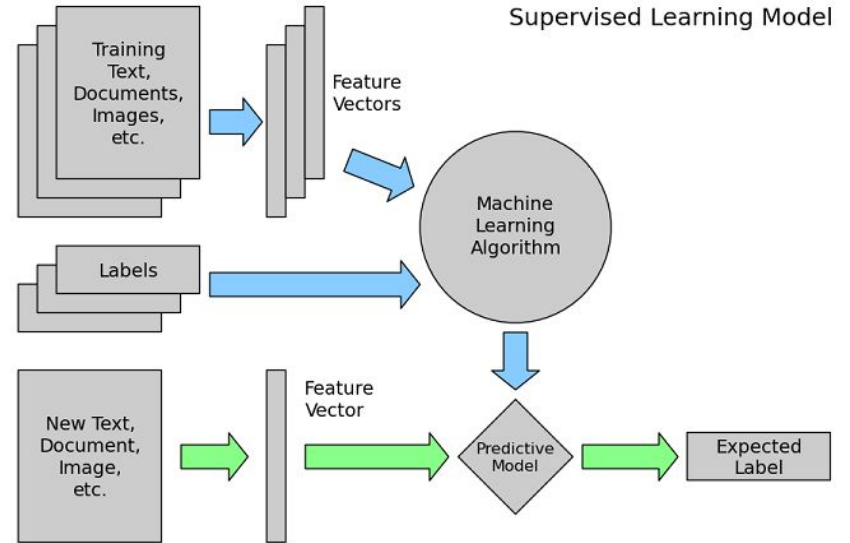
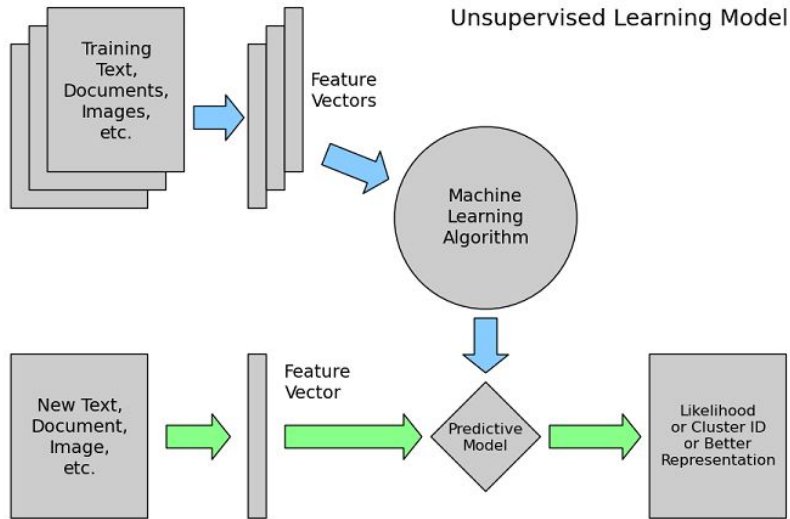
Feature Reduction

- F-Score $F = \frac{\text{explained variance}}{\text{unexplained variance}}$
- Chi2 $\frac{(\text{observed} - \text{expected})^2}{\text{expected}}$
- Non-negative matrix factorization
 -
- Principal component analysis
 - Separates features into “principal components” (which may include more than one feature).
 - The first principal component has the largest possible variance (it accounts for as much of the variability in the data as possible).

Machine Learning

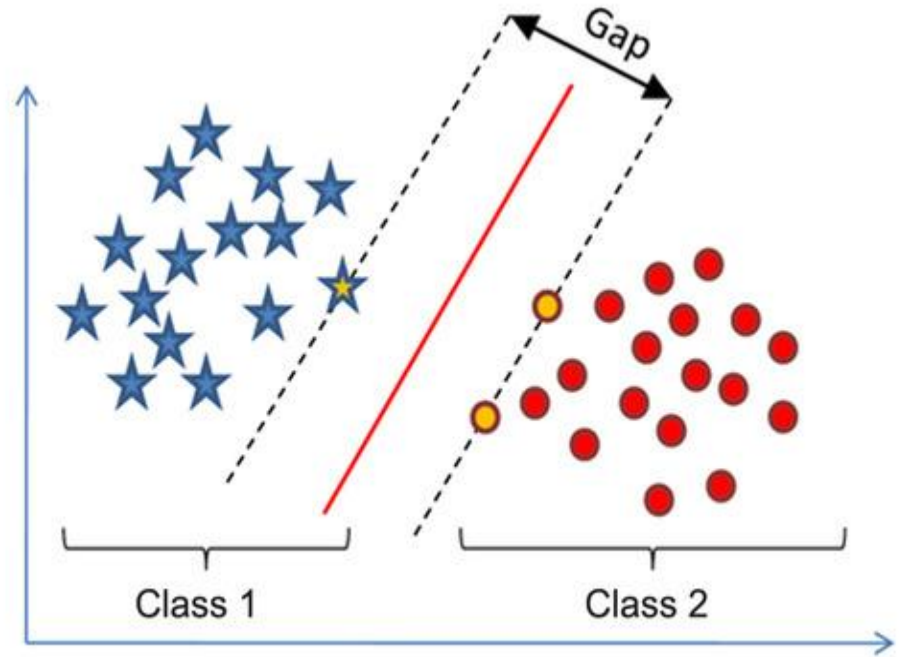
- Coined by Arthur Samuel in 1959 as “Giving the computer the ability to learn without being explicitly programmed”.
 - Tom Mitchell gave a formal definition, “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E”
- Evolved from pattern recognition in data.
- Machine learning “models” are given a set of training data, usually observed instances of what you are trying to measure in the future (in our case, Android Apps and what permissions, system calls are used).
- This model can “learn” the patterns in your training data and when given a new unseen sample predict the output based on what it has seen previously.

Supervised vs. Unsupervised



Support Vector Machine (SVM)

- Supervised
- Binary classifier, places instances in one of two categories (malicious or benign)
- Attempts to define a separating line (or plane) in between between the different categories

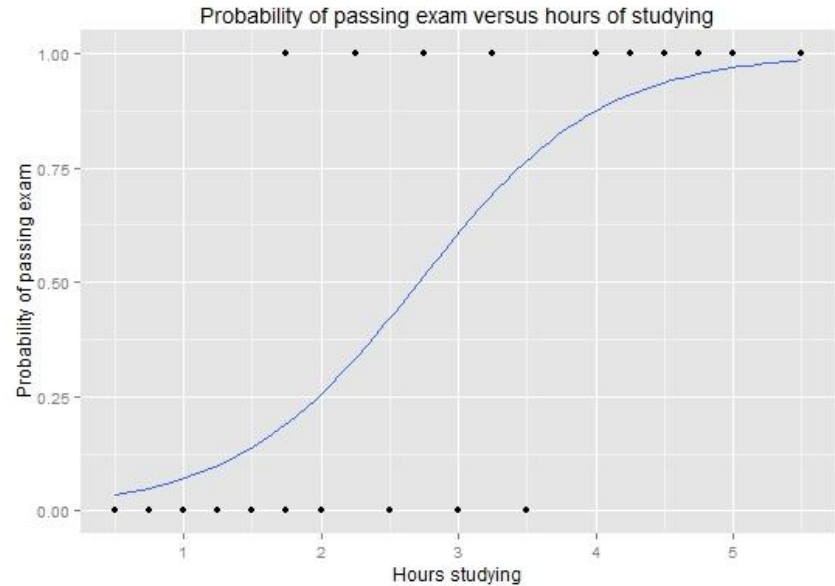


Logistic Regression

- Used for classifying binary results.
- Based on a Logistic function

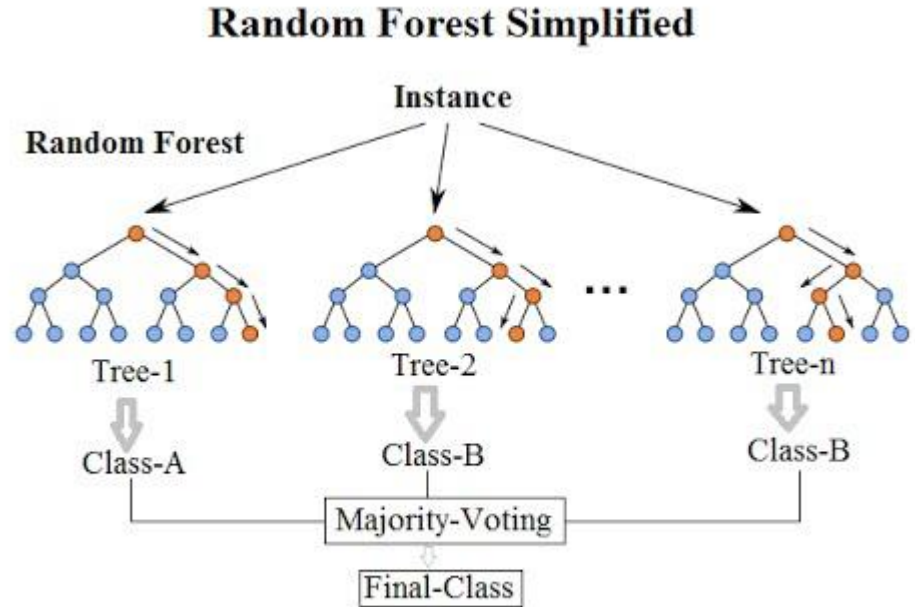
$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

- β_0 = y-intercept
- β_1 = Slope



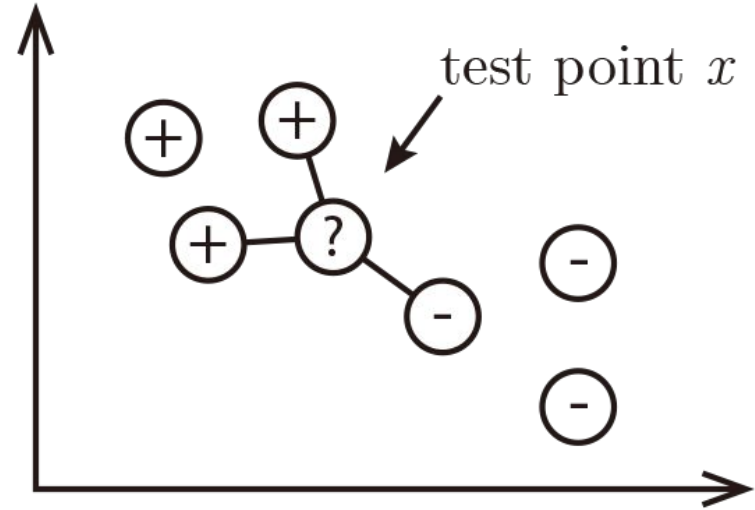
Decision Trees and Random Forest

- Random Forest is a collection of Decision Trees.
- Each Decision Tree will produce a different result, and is trained differently. (On each iteration of training, a random tree is chosen for training).
- Majority opinion wins on classification.



K Nearest Neighbors

- Have a bunch of samples which we know the classification for (benign or malicious).
- When classifying a new sample, classify it as the class that it is nearest to.
- If $K = 3$ and two samples are positive, and one is negative, we will classify the new sample as positive (majority wins).



Deep Learning Key Terms

- Epoch- A complete pass through a dataset
- Batch size- Number of samples that are propagated through a network at a time.
- Width- The width of a layer defines how many nodes it has.
- Activation- The function that defines the mapping of the input to the output. Responsible for determining to what extent (0.0-1.0) the input at a certain node should be transmitted to the next layer.
- Dropout- Randomly make nodes turn to 0, forcing the model to rely on other features. Intended to help prevent overfitting.

Our Neural Network

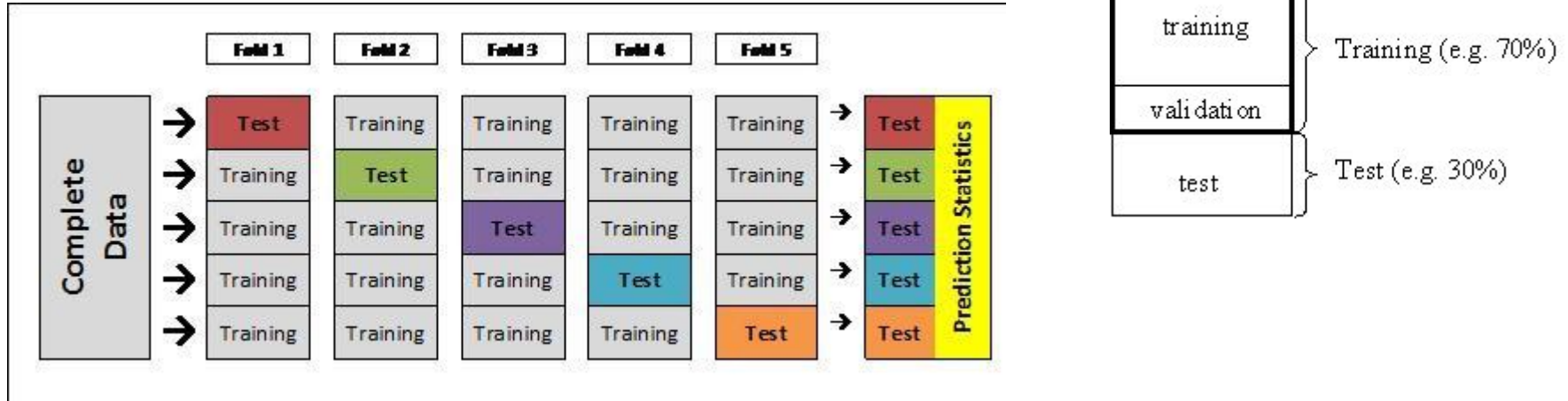
- We were able to get up to 95% accuracy with a shallow (3 layer) neural network
 - The first layer being a Dense layer with ReLU (rectified linear unit) activation and an output dimension of 128
 - The second layer being a Dropout layer with a 5% dropout rate
 - The final layer being a Dense layer with sigmoid activation and an output dimension of 1
- The model ran 500 epochs with a batch size of 500 and an 85/15 train/test split

Improvements to our Neural Network

- So far, we found that widening our network improved accuracy, while deepening it did not.
- I would like to try to utilize different types of layers (convolutional layers, pooling layers, etc...)
- I would like to try to implement DBN (Deep Belief Networks) as done in DroidDetector

Training and testing data

- We experienced the best results from a 70% training set and 30% testing set randomly.
- Consistent with previous work in Malware Detection.
- Used 4-fold Cross Validation

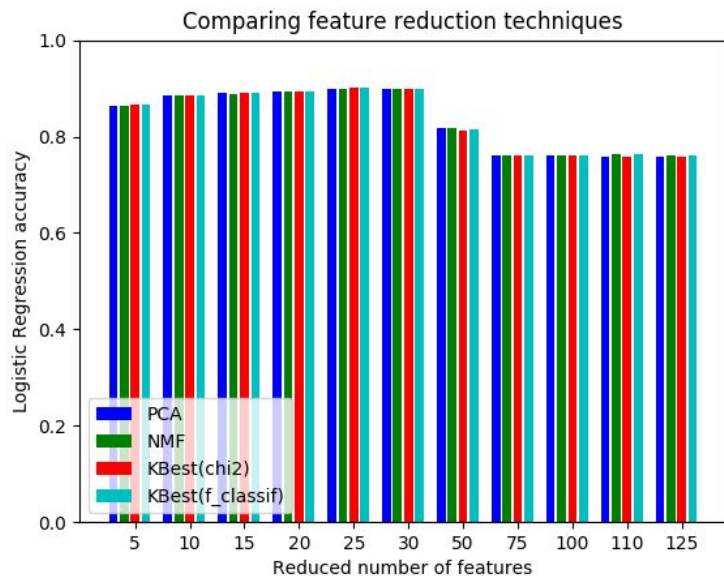
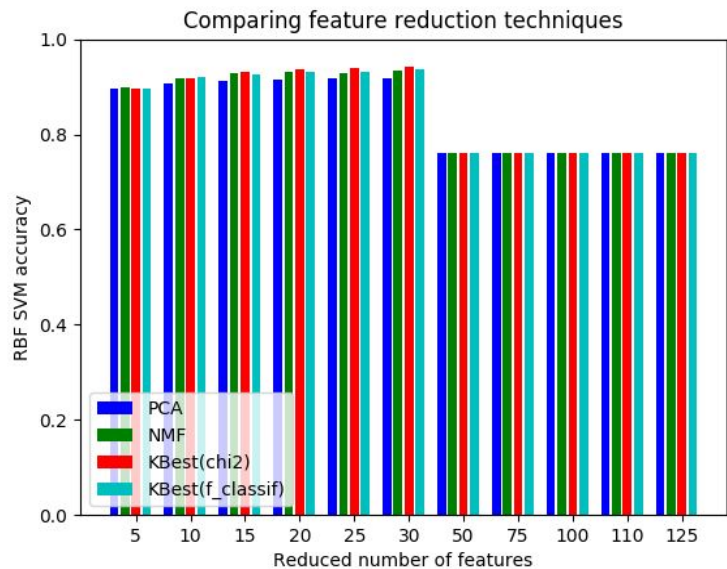


Cloudlab

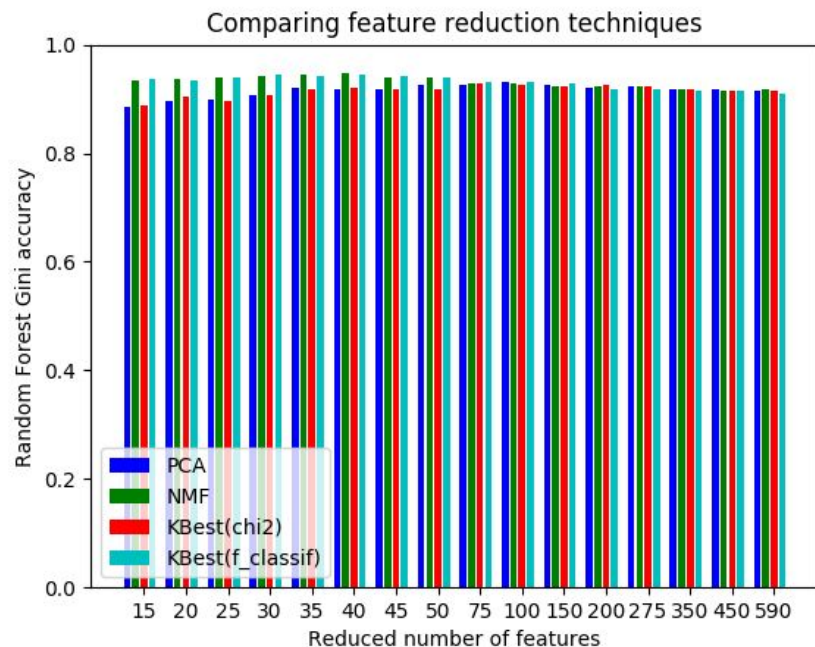
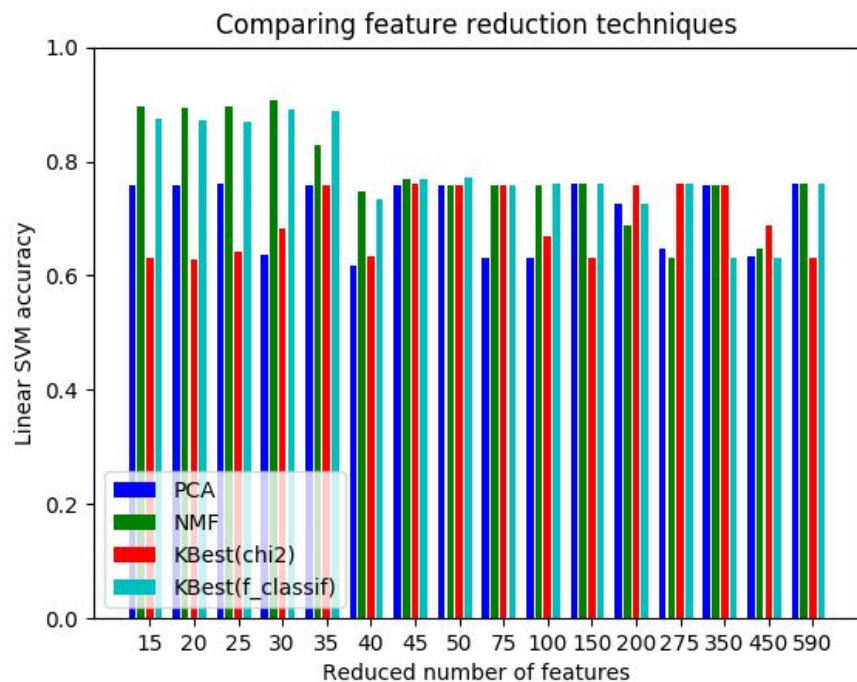
- We setup a cloudlab server to run machine learning algorithms on.
- Seems to be faster than our own computers.
- Bash script we wrote automates the activities we want to perform (e.g: feature selection).



Current Progress



Current Progress (cont)



Top Features

- SEND_SMS
- GET_ACCOUNTS
- WAKE_LOCK
- fsync
- NativeLIB
- WRITE_SECURE_SETTINGS
- READ_PHONE_STATE
- DELETE_PACKAGES
- GET_TASKS
- ACCESS_FINE_LOCATION

Timeline

Week 1

- Project Introduction
- Reading about Android OS and general security practices

Week 2

- Reading about Machine Learning
- Hands on work with Python and Machine Learning

Week 3

- Analyze features of malicious/benign applications on Android

Week 4

- Categorize features of malware applications

Week 5

- Propose a detection technique
- Implement detection technique with Machine Learning.

Week 6

- Compare results of using different Machine Learning algorithms.

Week 7

- Improve detection rates based on the results.

Week 8

- Start writing first draft of final report/publishable results

Week 9

- Revise draft

Week 10

- Finalize paper and submit deliverables



This week

- Modify the application. It should be able to:
 - Look at a new application upon its installation and send its features to our server
 - Input this feature set into our machine learning model
 - Retrieve an answer from our model (malware or benign)
 - Conduct appropriate actions depending on the answer:
 - If the app is determined to be malicious, uninstall it
 - Else, do nothing
- Continue modifying our machine learning models
 - Especially the neural network, attempt to apply convolutional layers

Questions?



References

- Weisstein, Eric W. "Chi-Squared Test". *MathWorld*.
- Amirudin, D. (2014, April 14). Gini, ROC, AUC (and Accuracy). Retrieved June 9, 2017, from <https://staesthetic.wordpress.com/2014/04/14/gini-roc-auc-and-accuracy/>
- KAUSHIK, SAURAV. (2014, December 1). Introduction to Feature Selection methods with an example (or how to select the right variables?). Retrieved June 9, 2017, from <https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/>
- Lindorfer, M., Neugschwandtner, M., & Platzer, C. (2015, July). Marvin: Efficient and comprehensive mobile app classification through static and dynamic analysis. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual* (Vol. 2, pp. 422-433). IEEE.
- Gerardo Canfora, Eric Medvet, Francesco Mercaldo, and Corrado Aaron Visaggio. 2016. Acquiring and Analyzing App Metrics for Effective Mobile Malware Detection. In *Proceedings of the 2016 ACM on International Workshop on Security And Privacy Analytics (IWSPA '16)*. ACM, New York, NY, USA, 50-57.

References

- S. K. Dash et al., "DroidScribe: Classifying Android Malware Based on Runtime Behavior," 2016 IEEE Security and Privacy Workshops (SPW), San Jose, CA, 2016, pp. 252-261.
- M. Lindorfer, M. Neugschwandtner and C. Platzer, "MARVIN: Efficient and Comprehensive Mobile App Classification through Static and Dynamic Analysis," 2015 IEEE 39th Annual Computer Software and Applications Conference, Taichung, 2015, pp. 422-433.
- Song, Yihang, and Urs Hengartner. "Privacyguard: A vpn-based platform to detect information leakage on android devices." Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices. ACM, 2015.
- Andrew Lipsman, 2017, Mobile Matures as the Cross-Platform Era Emerges.
<http://www.comscore.com/Insights/Blog/Mobile-Matures-as-the-Cross-Platform-Era-Emerges>
- Rahis Saifi, 2017. The 2017 Mobile App Market: Statistics, Trends, and Analysis. (2017). <http://www.goo.gl/uA1KMA>

References

- NowSecure Labs, 2016. Mobile Security Report. Technical Report. NowSecure.
- NowSecure, 2016. Secure Mobile Development Best Practices (2016).
[h.ps://books.nowsecure.com/secure-mobile-development/en/primer/mobile-security.html](https://books.nowsecure.com/secure-mobile-development/en/primer/mobile-security.html)
- DeepLearning4J, 2017. Deep Learning and Neural Network Glossary. <https://deeplearning4j.org/glossary>