Detecting Malware in Android Applications

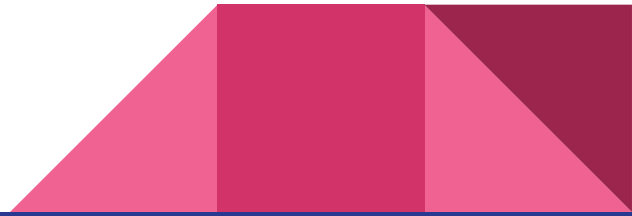
Harrison Mansour
Seaver Thorn
Prof. Fu
Hani Alshahrani

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.

Feature Selection

- After extracting all of the features from our application database, we had about 593 unique features, most of which being inappropriate to use.
  - Some features did not show up in any applications
  - Some features were highly correlated with one another
  - Some features simply did not offer any valuable information
- We have resorted to utilizing feature selection algorithms to find the optimal set of features to feed into our machine learning models. This feature selection will help us:
  - Improve accuracy
  - Improve speed
  - Avoid overfitting



Cloudlab

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

Recursive Feature Elimination (RFE)

- RFE is a greedy wrapper method, in that it wraps itself around some model in order to enhance it. The performance of RFE is largely dependent on the model that it is wrapping around.
- Here, a logistic regression model with all features is constructed. The worst (or best) performing feature is found and removed, and then a new logistic regression model is built with n-1 features. This process is continued until we have eliminated a sufficient number of features (as defined), or the model accuracy does not improve by removing features.

Logistic Regression

- Although it is not perfect, we chose to try logistic regression first because it is a fast and simple technique and our dependent variable is binary (an app is either malware or it is not).
- Can be thought of as finding the beta values that best fit: $y = \begin{cases} 1 & \beta_0 + \beta_1 x + \varepsilon > 0 \\ 0 & \text{else} \end{cases}$
    - Where 0/1 correlates to benign/malware (or malware/benign)

Top Features (Permissions) (unordered)

A description of these permissions and what they do can be found here: http://techblogon.com/android-permissions-list-example/

- BATTERY_STATS
- CALL_PRIVILEGED
- DELETE_PACKAGES
- PERSISTENT_ACTIVITY
- READ_LOGS
- SYSTEM_ALERT_WINDOW
- UPDATE_DEVICE_STATS
- WRITE_CALENDAR
- WRITE_CONTACTS
- GET_ACCOUNTS
- READ_PHONE_STATE
- CALL_PHONE
- PROCESS_OUTGOING_CALLS
- SEND_SMS
- READ_SMS
- ACCESS_WIFI_STATE
- CHANGE_WIFI_MULTICAST_STATE
- MODIFY_AUDIO_SETTINGS
- WAKE_LOCK
- WRITE_SYNC_SETTINGS

Top Features (System Information)

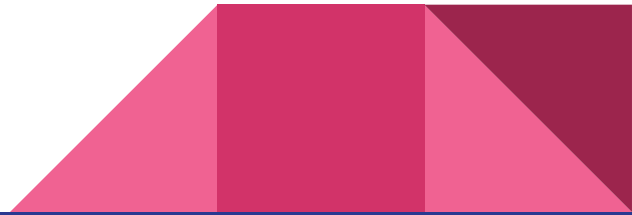
Many of these features are Linux system calls. We will need to understand how these work.

The system information indicates that malicious and benign apps use memory in very different ways.

- VmPeak
- VMSize
- VMHWM
- VMRSS
- VmData
- voluntary_ctxt_switches
- minorfaults
- minorfaultsch
- sizeM
- caught_signals
- Number_pages_swapped
- Number_pages_real_memory
- rssM
- clock_gettime
- futex
- gettimeofday
- getuid
- read1
- recv

This Week

- Standardize our dataset using a data standardization algorithm. This makes the mean 0 and standard deviation 1 in all of our fields. This will be done to eliminate potential problems with negative numbers.
- Run more feature selection algorithms on our dataset to make sure we have the best features.
- Test many Machine Learning models to use these features and classify new malware.



Questions?