CSCI 420-03: Analysis Plan

Christopher Short
cshort@email.wm.edu

1 Research Questions

RQ1: *Do applications that use the INTERNET permission tend to be overpriviledged? i.e. have more permissions than they need to function as described.*
RQ2: *Does the likelihood that an application is vulnerable decrease as the number of activities that contain use permissions increase?*
RQ3: *Do applications that require the user to accept all requested permissions, or otherwise terminate, have a higher chance of being malicious?*
RQ4: *Do applications allow all hostnames?*
RQ5: *Do applications that modify SSL socket factories use known broken SSL verification API calls?*
RQ6: *Do applications have a modified trust manager?*
RQ7: *Do applications use public methods?*
RQ8: *Do applications use implicit intents and method calls?*
RQ9: *Do applications that use broadcasts use the BroadcastPerm permission?*

2 Hypotheses

HP1: *Applications that use the INTERNET permission have a higher likelihood of being overpriviledged, i.e. having more permissions that they need to function.*
HP2: *As the number of use permissions that an application requests, the number of likelihood of that application being vulnerable decreases.*
HP3: *Applications that exhibit an all-or-nothing policy on requested permissions have a higher chance of requiring more permissions that it needs to function as described.*
HP4: *If applications use broken hostname verifier API calls, then it is more likely that they use common broken API calls, such as AllowAllHostnames().*
HP5: *If applications used modified SSL socket factories, then it is more likely that they use broken SSL verification API calls.*
HP6: *If applications use modified trust managers, then it is likely that they use a known broken trust manager.*
HP7: *Applications that use public methods are more likely to not protected behind explicit intents.*
HP8: *If applications use implicit intents then it is likely the application is not protected from exported intents.*
HP9: *If applications that use broadcasts use the BroadcastPerm permission, then that application is less likely to be vulnerable to intent hijacking.*

3 Evaluation Plan

These research questions will be answered through a set of three experiments with experiment 1 answering research questions 1 through 3, experiment 2 answering research questions 4 through 6, and experiment 3 answering research questions 7 through 9. The experiments will be conducted independently on the same set of test applications to ensure statistical continuity between the experiments using static analysis. This static analyses will be carried out through multiple different programs and scripts that will parse over the entire set of test applications. The three experiments will be based around one hypothesis each for a total of three hypotheses all of which are listed above.

3.1 Permission Checking

This experiment focuses on hypotheses 1 through 3 and is aimed at answering Research Questions 1 through 3. Applications that use the INTERNET permission have a higher likelihood of being overpriviledged, i.e. having more permissions that it needs to function. As the number of use permissions that an application requests, the number of likelihood of that application being vulnerable decreases. Applications that exhibit an all-or-nothing policy on requested permissions have a higher chance of requiring more permissions that it needs to function as described.

3.1.1 Experimental Setup

A bash script will be used to parse through Android Manifests and smali files of apks. The script will parse through the Android Manifest of an application and note which applications use the INTERNET permission, and how may dangerous permissions it requests. Then the script will parse all of the application's smali files in search of

method onRequestPermissionsResult() with the use of the closeNow() method to check for applications that terminate if not all of the requested permissions were accepted. The script will also count all of the use permissions that it requires and record that number. This number will be used with the results of all three experiments to observe whether there is a relation between the number of user permissions and the whether or not any of the experiments labeled that application as vulnerable.

3.1.2 Expected Results

This experiment will compare its findings with the results of experiments 2 and 3. Experiments 2 and 3 check for vulnerabilities and declare an application vulnerable or not vulnerable. In which case experiment 1 will comment on the relationship between applications being overpriviledged with respect to whether or not they are vulnerable. The results will be presented in the form of a percentage of overpriviledged applications that are vulnerable.

3.2 SSL Analysis

This experiment focuses on hypotheses 4 through 6 and is aimed at answering Research Questions 4 through 6. If applications use broken hostname verifier API calls, then it is more likely that they use common broken API calls, such as AllowAllHostnames(). If applications used modified SSL socket factories, then it is more likely that they use broken SSL verification API calls. If applications use modified trust managers, then it is likely that they use a known broken trust manager.

3.2.1 Experimental Setup

Using a bash script, this experiment will search through applications' dissembled apk smali files in search of SSL misuse. It will grep through the files with a list of known broken hostname verifier API calls. It will also grep through the files with a list of non-stock SSL socket factories, along with a list of non-stock trust managers and their related API calls. All uses of HTTP weblinks in the place of HTTPS weblinks will be recorded as well. The related API calls will be traced back two calls to test to see if such calls are actually invoked at some point during run time.

3.2.2 Expected Results

The analysis of applications using a bash script to find SSL misuse is expected to produce a list of the applications that contain SSL misuse out of the selected batch of applications as well as the percentage of applications with SSL misuse out of all applications tested. A breakdown in terms of the tested types of SSL misuse, allowing all hostnames, HTTP in place of HTTPS, custom SSL trust factories and custom trust managers, presented as a percentage out of the total number of applications with SSL misuse will also be produced. An application will be added to the list of applications that demonstrate SSL misuse if any of the SSL misuse cases listed above are found present in the application's smali files from its disassembled apk. The list of all of the applications that demonstrate SSL misuse will be used in creating the results for experiment 1.

3.3 Inter-Application Communication

This experiment focuses on hypotheses 7 through 9 and is aimed at answering Research Questions 7 through 9. Applications that use public methods are more likely to not protected behind explicit intents. If applications use implicit intents then it is likely the application is not protected from exported intents. If applications that use broadcasts use the BroadcastPerm permission, then that application is less likely to be vulnerable to intent hijacking.

3.3.1 Experimental Setup

The experiment will use a bash script in order to analyze inter-application communication. The bash script will parse the application's disassembled APK smali files in search of SQL use, public methods calls, and the use of broadcast receivers. In the case that SQL use is found the script will check to make the input received is parsed and sanitized in some fashion before being handled by the SQL database. In the case of broadcast receivers the script will make sure that all sent broadcasts are protected by requiring listeners to have the BroadcastPerm permission.

3.3.2 Expected Results

The analysis of the set of test applications using a bash script to find unprotected inter-application communication is expected to produce a list of applications that demonstrate unprotected inter-application communication. An application will be marked as unprotected and added the list if the application does not sanitize incoming SQL commands (given that it runs an SQLLite database), uses public methods that are reachable and callable, and unprotected implicit intents. This experiment will produce a percentage of applications that are not protected in one of the areas mentioned above to the number of total applications along with a break down of the applications that have each of the unprotected features mentioned above along with that percentage out of the total number of applications found unprotected.