CSCI 445 - Homework #4: Automated Security Analysis[*]

Assigned March 31st; Due 11:59pm on April 16th {50 points}

Prof. Adwait Nadkarni

1 Assignment Goals and Overview

The objective of this assignment will be to teach you how to *automate* your security analysis, and more importantly, the trade offs involved in doing so.

Task: You will be given five Android applications. Your task is to write a script that automatically locates SSL-related vulnerabilities in the assembly code. Your script should execute as follows:

```
$ <your-script> -i target-app.apk -o output.txt
```

That is, your script should (1) take an apk name/path relative to the current directory as input, (2) disassemble it (using apktool), (3) analyze the assembly code (i.e., the .smali files), and (4) print the analysis results in an output.txt file.

For example, if my script is called analyze.sh, and I have the vulnerable.apk in the directory apps/, then I should be able to *place* analyze.sh *inside* apps/, and on executing, find the output in apps/output.txt. This should work likewise, for all five apps provided. The detailed assignment, with points, is described in Section 2.

For automating your analysis, you will be permitted to use one of the following: (1) bash, (2) python, or if you really want to, (3) perl. Even if you have no prior experience with scripting, you should be able to complete the assignment, as the scripting involved will be really basic (i.e., mostly traversing directories and parsing strings).

Tools: You will use apktool to disassemble apps, as before. Calling it from within a bash/python script is relatively straightforward, and not different from calling any other program. In addition to apktool, you can use/incorporate existing tools in your scripts, *if and only if they come pre-installed with a standard Linux distribution*; e.g., grep, cut, awk, uniq. No other third-party tool will be permitted; when in doubt, ask.

Environment: Your scripts should execute on a Ubuntu 16.04.4 LTS Desktop build, without the requirements of any additional libraries/tools. I recommend creating a Virtual Machine to perform

[*]Last revised on March 31, 2024.

all experiments, using this image: Ubuntu Desktop 16.04.4 LTS. The TA will use the same image to test your scripts against the 5 target apps.

2 Assignment Description

In this assignment, you will write a script that looks for SSL-related vulnerabilities in five apps.

Allocation of Points: The assignment is worth 50 points in total. There will be two deliverables: (1) lastname-analyze.sh/pl/py (i.e., depending on whether it is a shell, perl, or python script), and (2) lastname-report.pdf, both zipped into a lastname.zip. The allocation of points will be as follows:

1. {40 points} **Task A: Vulnerability discovery:** lastname-analyze.sh/pl/py finds one *potentially exploitable*[1] flaw each, in 4 out of 5 apps. 10 points per flaw. Note that each flaw can be different (i.e., you don't need to find the same flaw in all apps). Each flaw found should be described in lastname-report.pdf to score points (0 points without description).
2. {10 points} **Task B: False Positives:** For each app, your script will likely output findings that are not necessarily flaws. Identify and list these *false positives*, as well as the *true positives* (i.e., findings that are flaws), across all apps, in a separate lastname-report.pdf.

Negative Points: Points will be deducted for each instance of the following:

1. **Script crashes**{-50 points}: i.e., ZERO points will be awarded if your script does not work.
2. **Hardcoded vulnerability in script**{-10 points} You cannot hardcode paths or classes of the target apps in which you are looking for vulnerabilities. Your scripts should generally apply to all apps, and not have rules specific to one or more target apps. When in doubt, ask.

3 Recommended Process

Take a look at this paper: Why eve and mallory love android: an analysis of android SSL. It will tell you what to look for in apps, i.e., the kind of mistakes apps make. To successfully complete this assignment, here are the recommended steps:

1. *Step 1:* From the paper, identify 3-5 types of application characteristics that lead to SSL flaws, that you want to look for (e.g., the use of specific classes that are almost always unsafe, or use of HTTP instead of HTTPS).
2. *Step 2:* Write a script (i.e., lastname-analyze.sh/pl/py that disassembles the app using apktool and parses *the app's smali code*, i.e., excludes code in the libraries included with the app, for these flaws. HINT: The app code is generally found along the same path as suggested in its package name. For instance, if my app's package name is com.professor, then look in:/smali/com/professor. Look for flaws identified in Step 1, in the smali code.

[1]Code that is flawed, but not called from anywhere in the app, is not exploitable

3. *Step 3:* Use manual analysis to confirm that each flaw identified by your script is a potentially exploitable vulnerability, i.e., not a false positive.

4. *Step 4:* If you have found the number of flaws necessary for a successful completion of this assignment, continue. If not, find another characteristic/flaw to look for from Step 1, and goto step 2.

5. *Step 5:* Describe each vulnerability found, in the analysis report PDF, and indicate its location in the app.

6. *Step 6:* Through manual verification, identify false positives in the output of your script, when executed on all the 5 apps, and state their final number in the analysis report PDF.

4 Submission Instructions

Submit your solution as a single tarball or zip (tar.gz/zip archive) named **lastname-hw4.tar.gz or lastname-hw4.zip** to Blackboard. The solution should contain two components: (1) A single script lastname-analyze.sh/pl/py (i.e., depending on whether it is a shell, perl, or python script), and (2) A single PDF lastname-report.pdf, which is the analysis report that contains a description of the vulnerabilities found, as well as the false positive count.

Please post questions (especially requests for clarification) about this homework to Piazza.