TPAS Project 2022/2023

Reverse Engineering Guide

Phase 1 - Getting started

Study and practice reverse engineering:

- https://malwareunicorn.org/workshops/re101.html
- TPAS slides and challenges: reverse engineering and malware analysis
- Other links on https://tpas-desafios.alunos.dcc.fc.up.pt/resources
- If you're into game research, you can investigate:
    - https://github.com/dsasmblr/game-hacking
    - https://doar-e.github.io/blog/2015/08/18/keygenning-with-klee/
    - https://github.com/giuliano108/reverse-engineering-intro/blob/master/keygen-z3.py
- If you're into android reverse engineering, you can study:
    - https://www.evilsocket.net/2017/04/27/Android-Applications-Reversing-101/
    - https://www.youtube.com/watch?v=7SRfk321I5o
    - https://www.youtube.com/watch?v=cLUl_jK59EM
    - https://github.com/rednaga/training/tree/master/DEFCON23
    - https://github.com/rednaga/training/blob/master/DEFCON23/O%26D%20-%20Android%20Reverse%20Engineering.pdf
    - https://portswigger.net/support/configuring-an-android-device-to-work-with-burp
    - https://portswigger.net/support/configuring-an-ios-device-to-work-with-burp

Phase 2 - Picking a target

After practicing reverse engineering, you should pick a target. Most students asked for suggestions for this project. However, you pick any software for your goal, e.g. games from 2000-2010 (keygen, cracking), mobile apps and games, or desktop apps. If you want to find vulnerabilities, you can check for games or desktop apps that have public bug bounty programs, e.g. https://hackerone.com/valve. Pick a target that hasn't been publicly researched so far. If you have questions, please ask the professor or schedule meetings via email.

Phase 3 - Analysis

- Static analysis - disassembly, decompiling with IDA, Ghidra, JADX, etc. Identify potential packers, obfuscators, cryptors, etc. Analyze resources, functions, and graph flows.

- Dynamic analysis - run the software, look for system changes (new/modified files), debug and understand functions, sniff with wireshark, proxying system-wide HTTP/HTTPS requests if applicable. More information on mobile app research follows.

- Cracking (if applicable): try to find the serial number validation code or other license checks. You can try to perform patching to crack a given software. You can also build a keygen with https://github.com/Z3Prover/z3 by translating constraints and validations.

Proxying mobile apps and software with Burp and manipulating requests can be very useful for finding vulnerabilities. Important: please check with the professor if you want to understand if it's legal and safe to analyze a given URL, IP or hostname used by the mobile app or software ⚠️

Special notes on mobile apps:

I recommend using Genymotion or another Android emulator for testing mobile apps. If you have an iPhone, jailbreak is necessary.

- Extract the most recent version of an app from a device da versão (apkextractor, **`adb pull`**)

- Extract resources with apktool
- Decompile code with JADX
- Decompile and reverse native libraries with Ghidra/IDA.
- Research for OWASP mobile top 10 vulnerabilities
- AndroBugs: https://github.com/AndroBugs/AndroBugs_Framework
- FindSecBugs over JAR files (use Dex2Jar) - usually results in a high false-positive rate. Ignore vulnerabilities that don't belong to the main application classes (other vendors)
- Use a proxy (Burp Suite) to manipulate HTTP/HTTPS requests. More info here. If any certificate pinning mechanisms are in place, try to bypass (ask the professor for help if necessary).
- Check out external storage. Is there any sensitive data?
- Explore Frida on a rooted device or emulator.

Phase 5 - Final report

You must submit a final report describing your work and perform a presentation in class (dates TBD). Suggested sections are: Introduction, Learning Process, Methodology, Analysis, (Vulnerabilities), Final Remarks.