Software Reversing Lab 2 – Fuzzing and Analysis

You are asked to fuzz **two** existing application, or execute it concolicly. This can be a piece of code one of you developed, any downloaded binary (check CVE's https://cve.mitre.org/ for common vulnerabilities in (older) binaries), or an open source project. In addition to trying to make it crash, you should investigate and explain why it crashes.

In the case of available source code:

- Provide the source code that causes the crash and determine how to avoid it.

In the case of a binary:

- Provide the stack content at time the crash, and discuss why it happens.

You may use any tools available such as IDA, Radare, Valgrind, GDB, decompilers, etc.

If you cannot find any crash, please find **two** other software tools that do crash (see the AFL website or the CVEs for possibilities). One popular tool to crash is ImageMagick.

You have to create a video of your application, clearly demonstrating how to implement the required steps, what results are obtained, and their meaning. Optionally, you may include a short description of the required tools that need to be downloaded and how to set them up, preferably in an md file format, similar to the hand-on sessions during the lectures.

Your video will be evaluated on the following criteria:

- Clarity of presentation
- Reproducibility of performed study
- Quality of performed study
- Difficulty of performed study
- Meaning of obtained results
- Extra credit for any additional insights obtained

You can get extra credit by for instance combining fuzzing with learning and testing techniques, or by finding non-crashing bugs using these tools.