CSIT5730: Course Project

Shuai Wang

香 港 科 技 大 學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Logistics

- 10/10 – Group project topics introduction

- 10/10 ~ 10/17 – Express grouping intention and project intention
  - Please send our TA (sdengan@cse.ust.hk) an email and copy me.
  - When I believe your proposed project needs some tweaks (see later slides), I will let you know

- 10/18 – Confirm groups and project
  - We might tweak if you were not in any group

- In the week of 11/04 – discuss on progress
  - For each group, we chat on Zoom
  - Understand if you are on the track

- 11/28 – project presentation
  - Will provide feedback for you to further concretize your project

- 12/13 – project package submission
  - Report; presentation slides; code
  - We will provide a word/latex template for the report
  - Final grading needs to be submitted to University at 12/19

Group Project topic introduction

- We tried to keep each project with comparable difficulty and workload, in our best effort.

- some of the topics may not be introduced in the lecture yet; requiring some self-studies.

- Expect reasonable amount of engineering efforts

- Write some code, or reuse some existing tools

- launching experiments to quantify the results achieved

It is totally fine if you cannot fully finish the project in the end; all in all, we look at your effort, and do grading based on the difficulty of the project.

Suggestion 1: Protect Smart Contract Code with obfuscation

- Background: Smart contract (like a “dialect” of JavaScript), is the language of Ethereum blockchain. Smart contract handles large volume of crypto token transactions, and therefore, it’s critical to protect them.
- Topic: you are suggested to design and implement software obfuscation techniques toward smart contract programs.
- Code: you need to write some obfuscation programs, such that when a smart contract code is in, an obfuscated version is out.
- Evaluation (suggested plan): evaluate based on criteria in “Evaluation of Obfuscation Effectiveness” page of the slides.

Suggestion 2: eBPF-based Intrusion Detection System

- Background: eBPF (extended Berkeley Packet Filter) allows the user to run programs inside the kernel in an event-driven mode. It provides powerful observability into the kernel-space.

- Topic: you are suggested to design and implement an IDS (intrusion detection system) upon eBPF.

- Code: you need to write a set of eBPF programs that monitor system events, e.g., syscall, and incorporate them into one IDS. Your IDS should warn the user when an anomaly is detected.

- Evaluation (suggested plan): evaluate based on #anomaly/event supported and the performance overhead brought by your detection system.

Suggestion 3: Query-based Static Application Security Testing (SAST)

- Background: SAST is a common choice to find vulnerabilities in industrial software. Q-SAST is a branch that detects various weaknesses of an application efficiently, based on user-provided queries (i.e., "bug patterns").

- Topic: you are suggested to use Q-SAST tools, in particular, **CodeQL**, to detect vulnerabilities in programs. You are expected to use and improve the official query database for C/C++: https://codeql.github.com/docs/codeql-language-guides/basic-query-for-cpp-code/

- Code: If CodeQL identifies non-vulnerable code pieces as vulnerable with a query rule, you need to improve the query to avoid the false alarm. If CodeQL fails to identify a known vulnerability, please write a new query for it.

- Evaluation (suggested plan): new queries should consider the code pattern (e.g., abstract syntax tree, control flow, and data flow) instead of simply "pattern match" function and variable names. The score depends on the number of useful new queries for our evaluation dataset and the accompanying explanation for a new query.
  - For evaluation datasets, we suggest to select five CWE cases from https://github.com/arichardson/juliet-test-suite-c

Suggestion 4: fuzzing for network security

- Background: HTTP/3 aims to be the next-generation protocol for the World Wide Web. Yet, it may need some fuzzing efforts to enhance its security and uncover bugs.
- Topic: You are suggested to 1) fuzz an HTTP/3 server (e.g., quic-go) using those standard fuzzing tools (relatively easy), and 2) improve the fuzzing effectiveness by taking into consideration the HTTP/3 protocol's statefulness (harder).
- Code: You need to implement a fuzzer for the HTTP/3 server. You may base your work on existing frameworks like AFLNet.
- Evaluation (suggested plan): A commonly used metric is code coverage. You may also evaluate your fuzzer by counting the number of bugs (e.g., crashes) found. It will be extraordinary if your fuzzer finds previously unknown bugs, in which case you should report them to the developers of the HTTP/3 server.

Other Suggestions

- Background: You can propose your own group project, if that appears better.
- Topic: related to computer security and privacy. Should not be a simple “app development”.
- Procedure: you can propose and check with me at your earliest convenience by Oct. 17th.