CS 858: Software Security

Offensive and Defensive Approaches

Introduction: course logistics

Meng Xu *(University of Waterloo)*

Fall 2022

Outline

About me

- Meng Xu
- Assistant Professor at Cheriton School of Computer Science
  - Joined on September 2021.
- Member of CrySP and CPI.

About me

- Meng Xu
- Assistant Professor at Cheriton School of Computer Science
  - Joined on September 2021.
- Member of CrySP and CPI.

- Completed PhD at Georgia Tech (August 2020)
  - Advisor: Prof. Taesoo Kim
- Worked on several streams of software security research:
  - Moving-target defense (i.e., software diversity)
  - Static program analysis (e.g., symbolic execution) on the Linux kernel
  - Dynamic program analysis (e.g., fuzz testing) on filesystems

About me

- Meng Xu
- Assistant Professor at Cheriton School of Computer Science
  - Joined on September 2021.
- Member of CrySP and CPI.

- Completed PhD at Georgia Tech (August 2020)
  - Advisor: Prof. Taesoo Kim
- Worked on several streams of software security research:
  - Moving-target defense (i.e., software diversity)
  - Static program analysis (e.g., symbolic execution) on the Linux kernel
  - Dynamic program analysis (e.g., fuzz testing) on filesystems

- One gap-year at Facebook / Meta on the blockchain division
  - Move — the secure smart contract language
  - Move Prover — a formal verification tool for Move programs

About this seminar course

First thing first: this is a seminar course.

About this seminar course

First thing first: this is a seminar course.

- We want to align the course content with your interests.
- We hope the security aspects can be a supplement to your research / work projects.
- We DO NOT intend to test your knowledge about security in this course — take CS 658 instead.

About this seminar course

First thing first: this is a seminar course.

- We want to align the course content with your interests.
- We hope the security aspects can be a supplement to your research / work projects.
- We DO NOT intend to test your knowledge about security in this course — take CS 658 instead.

Summary: treat this course as a guided tour on the software security research landscape.

Meeting logistics

Time: 1:00pm - 3:50pm every Tuesday

Location: in-person at DC 2585, online via Zoom

Format:

- A an introductory overview on the topic (75 minutes).
- 1-2 paper presentation at 45 minutes each, including Q & A.

Materials available online include papers to read, presentation slides, and any supplement materials to facilitate the understanding of the topic. However, as these are not normal lectures, we will not provide recordings.

Topics to cover

Refer to Course Outline.

Assessment

- Paper presentation — 20%
- Capture-the-flag — 30%
- Research project — 50%

Assessment

- Paper presentation — 20%
- Capture-the-flag — 30%
- Research project — 50%

- We do not fit scores into curves.
- Late submissions are generally not accepted, unless there are long-lasting problems.
- Reappraisal can be requested with a clear justification of your claims — send the request to the instructor via university email within one week of grade release.

University policies

In this course, you will be exposed to information about security problems and vulnerabilities with computing systems and networks. To be clear, you are NOT to use this or any other similar information to test the security of, break into, compromise, or otherwise attack, any system or network without the express consent of the owner.

Refer to the list of relevant university policies when in doubt.

Academic integrity

Don't copy-paste!

Outline

Round of introduction

Give a short introduction about yourself, including

- Name
- Area of research / work (or still exploring)
- What do you want to learn from this course
- Anything else you would like us to know

HotCRP conference management system

HotCRP is the conference management system used by all top-tier security conferences.

In this course, we re-purpose it for several tasks, including:

- Bidding for presentation slots
- Submission of presentation evaluations and feedbacks
- Registration of research projects and
- Peer-review on others' research projects.

Please register an account for this course using your UWaterloo email address (if you haven't done so).

HotCRP conference management system

Briefly, every user will have one of the three roles in the system:

- Author: *your default role once registered*
  - Limited to submit papers and receive feedbacks
- PC Member: *all enrolled students will be promoted to PC member*
  - Submit papers and receive feedbacks from peer-reviews
  - Provide reviews and evaluations of others submissions
- PC Chair: *the course instructor*
  - Everything a PC member can do
  - Administrator tasks

Pesentation preference selection

Live walkthrough on HotCRP

Outline

1. Formalities
2. Course setup
3. Introduction to software security research

Software security research landscape

Generally speaking, almost all research work in the software security area can be categorized into four bins:

- Exploitation:
- Attack:
- Defense:
- Detection:

Software security research landscape

Generally speaking, almost all research work in the software security area can be categorized into four bins:

- Exploitation:
- Attack:
- Defense:
- Detection:

What are the differences between them?

Software security research landscape

Generally speaking, almost all research work in the software security area can be categorized into four bins:

- Exploitation: *Given a bug, exploit it to achieve a desired (and specific) goal*
  - $f(Code, Bug) \rightarrow Action$
- Attack:
- Defense:
- Detection:

What are the differences between them?

Software security research landscape

Generally speaking, almost all research work in the software security area can be categorized into four bins:

- Exploitation: *Given a bug, exploit it to achieve a desired (and specific) goal*
  - $f(Code, Bug) \rightarrow Action$
- Attack: *Identify a bug in the program that may cause some damage*
  - $f(Code) \rightarrow Bug$
- Defense:
- Detection:

What are the differences between them?

Software security research landscape

Generally speaking, almost all research work in the software security area can be categorized into four bins:

- Exploitation: *Given a bug, exploit it to achieve a desired (and specific) goal*
  - $f(Code, Bug) \rightarrow Action$
- Attack: *Identify a bug in the program that may cause some damage*
  - $f(Code) \rightarrow Bug$
- Defense: *Given a bug and a set of intended exploitations, prevent them*
  - $f(Code, Bug, \{...Action...\}) \rightarrow Blockage$
- Detection:

What are the differences between them?

Software security research landscape

Generally speaking, almost all research work in the software security area can be categorized into four bins:

- Exploitation: *Given a bug, exploit it to achieve a desired (and specific) goal*
  - $f(Code, Bug) \rightarrow Action$
- Attack: *Identify a bug in the program that may cause some damage*
  - $f(Code) \rightarrow Bug$
- Defense: *Given a bug and a set of intended exploitations, prevent them*
  - $f(Code, Bug, \{...Action...\}) \rightarrow Blockage$
- Detection: *Given a program, check the existence of a specific type of bug*
  - $f(Code, Bug, [Action]) \rightarrow Signal$

What are the differences between them?

Software security research landscape

Generally speaking, almost all research work in the software security area can be categorized into four bins:

- Exploitation: *Given a bug, exploit it to achieve a desired (and specific) goal*
  - $f(Code, Bug) \rightarrow Action$
- Attack: *Identify a bug in the program that may cause some damage*
  - $f(Code) \rightarrow Bug$
- Defense: *Given a bug and a set of intended exploitations, prevent them*
  - $f(Code, Bug, \{...Action...\}) \rightarrow Blockage$
- Detection: *Given a program, check the existence of a specific type of bug*
  - $f(Code, Bug, [Action]) \rightarrow Signal$

- Anything better than detection?

Software security research landscape

Generally speaking, almost all research work in the software security area can be categorized into four bins:

- Exploitation: *Given a bug, exploit it to achieve a desired (and specific) goal*
  - $f(Code, Bug) \rightarrow Action$
- Attack: *Identify a bug in the program that may cause some damage*
  - $f(Code) \rightarrow Bug$
- Defense: *Given a bug and a set of intended exploitations, prevent them*
  - $f(Code, Bug, \{...Action...\}) \rightarrow Blockage$
- Detection: *Given a program, check the existence of a specific type of bug*
  - $f(Code, Bug, [Action]) \rightarrow Signal$

- Anything better than detection?
- Prevention!
  But that's usually the area of Programming Languages (PL)

A general framework to appreciate software security papers

A general framework to appreciate software security papers

For example: given two defense papers P_1 and P_2 on the same bug:

$$P_1(Code_1, Bug, \{...Action_1...\}) \rightarrow Blockage_1$$

$$P_2(Code_2, Bug, \{...Action_2...\}) \rightarrow Blockage_2$$

- Is $Code_2$ more complicated than $Code_1$?
- Is $Action_2$ larger than $Action_1$ (i.e., protection scope is larger)?
- Is $Blockage_2$ more efficient $Blockage_1$ (i.e., lower overhead)?

A general framework to appreciate software security papers

A general framework to appreciate software security papers

For example: given two detection papers P_1 and P_2 on the same code base:

$$P_1(Code, Bug_1, [Action_1]) \rightarrow Signal_1$$

$$P_2(Code, Bug_2, [Action_2]) \rightarrow Signal_2$$

- Is Bug_2 more challenging than Bug_1?
- Is $Action_2$ simpler than $Action_1$ (i.e., easier to detect)?
- Is $Signal_2$ more accurate $Signal_1$ (i.e., lower false positives)?

A general framework to create new research

A general framework to create new research

For example: given an attack and detection paper

$$P(Code_1) \rightarrow Bug \quad || \quad P(Code_1, Bug, [Action_1]) \rightarrow Signal_1$$

we can ask ourselves, is another code base $Code_2$ also vulnerable to the same (or similar) type of bug?

$$P(Code_2) \rightarrow Bug \quad || \quad P(Code_2, Bug, [Action_2]) \rightarrow Signal_2$$

⟨ **End** ⟩