Midterm Review (II): Software Security

Dawn Song
dawnsong@cs.berkeley.edu

Common Implementation Flaws

- **Buffer overflows**
- **Format string vulnerabilities**
- **Integer overflow**
- **Double free**
- **User/kernel pointer bugs**
- **TOCTTOU**

- **What power can attackers gain by exploiting these vulnerabilities?**

Memory Safety Exploitation (I): Attack Taxonomy

- **Control flow hijacking**
  - **Code injection vs. Return-to-libc**
  - **Overwrite target**
    - » **Return address & base pointer (activation record)**
    - » **Function pointers**
    - » **Exception handler**
- **Data attacks**
  - **Overwrite security-critical data variables**
- **Overwrite target area**
  - **Stack, heap, global variables, environment variables**

Memory Safety Exploitation (II): Defenses

- For each defense
  - Advantages & disadvantages?
  - What types of attacks does it defend against?
  - What types of attacks does it not defend against?
- Non-executable stack
- Purify-type runtime bounds check
- Stack guard
- Randomization
- Other examples
  - Growing the stack the other way
  - Pointer guard
  - Instruction randomization

User/Kernel Pointer Bugs

- When kernel uses pointers passed from user, kernel needs to do sanitization checks
- A missing check can cause system hang, crash kernel, gain room privileges, read secret data from kernel buffers
- Standard functions for such checks
  - copy_to_user
  - copy_from_user
- Bug-finding
  - Find places where checks are missed

TOCTTOU

- TOCTTOU vulnerability can arise where mutable state shared btw two or more entities
  - Check and use are not atomic
  - State can change btw check and use
- Many Unix file system calls are not atomic
- Attacker bypass security checks

Bug Finding

- **Black-box fuzz testing**
  - **Mutation-based fuzzing**
  - **Generation-based fuzzing**
  - **Advantages & disadvantages?**
- **Code coverage**
  - **Line/block coverage**
  - **Branch coverage**
  - **Path coverage**

Constraint-based Test Case Generation

- **Symbolic execution**
- **For a given path, check whether an input can trigger the bug (i.e., violate security property)**
  - **Assertion**
- **Find inputs that will go down different program execution paths**
  - **Path constraint**
- **Example**

Program Verification

- **Precondition/postcondition**
- **Loop invariant**
- **How to use this code reasoning to prove absence of security vulnerabilities**
  - **Write down the assertion that should hold to ensure certain security properties**

Midterm

- **9-10:30am Oct 22 (Wed), 306**
  - **Pls be on time**
- **Closed book**
- **Some questions may be hard**
  - **Finish as much as you can**
  - **Grades are curved**

Questions?