**CS 458/658 - Computer Security and Privacy** **Fall 2018**

# Lecture 2: September 11, 2018

*Lecturer: Ian Goldberg* *Notes By: Harsh Mistry*

## 2.1 Program Security

- writing secure programs is difficult because of Murphy's law and the fact that security relevant programs will always have security bugs

### 2.1.1 Flaws, Faults, and Failures

- A flaw is a problem
- There are typically two types of flaws.
  - **Faults** - A potential problem (Inside view), so a potential issue in the code.
  - **Failures** - A failure is when something actually goes wrong (outside view)
- Every failure is the result of a flaw, but not every flaw yields a failure.

#### 2.1.1.1 Methods for finding flaws

- **Root Cause Analysis** : You can work backwards from a failure
- **Penetrate and Patch** : *Think like an attacker* and intentionally cause failures, then issue a patch

#### 2.1.1.2 Problems with patching

- When patching, pressure is typically high which can cause focus to be narrowed onto a specific failure. This may prevent the programmer from thinking of broader issues.
- The patched fault may have caused other failures, which may not be known
- The patch may introduce new faults

#### 2.1.1.3 Developing to the specification

Adding additional behaviour can have security consequences and lead to unexpected behaviour. Given this, when implementing a security/privacy application, the spec must be complete. The application must also match the specification exactly. There should be no additional features. Looking at code is typical the best way to ensure there is no additional exploits and to ensure the spec is met with a high degree of accuracy.

### 2.1.2 Security flaw types

- Unintentional : Most flaws are unintentional
- Intentional
  - Malicious
    - General
    - Targeted
  - Non-Malicious

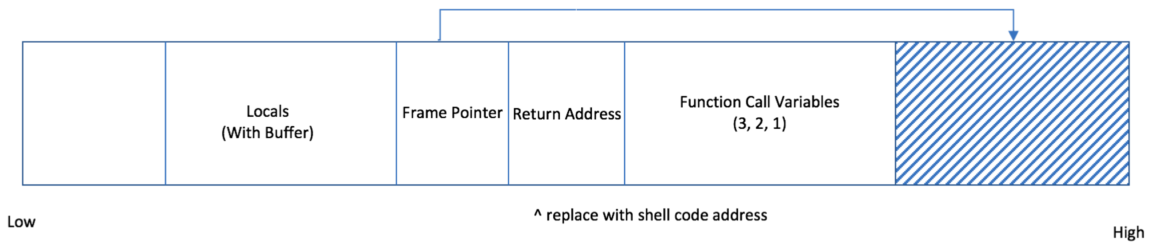
### 2.1.3 Unintentional Flaws

Unintentional flaws are the most common type of flaw and occur when the developers mental model does not match the real model. Types of unintentional flaws are

- Buffer Overflows
- Integer Overflows
- Format String Vulnerabilities
- Incomplete Mediation
- TOCTTOU Errors

#### 2.1.3.1 Buffer Overflows

- Buffer overflow is when a large amount of data is written into a location which is smaller than the data to copy/write. This causes memory outside the data location to be overridden
- If programs can write data to indices outside the desired location, an attacker can modify things like saved return addresses, which could cause the program to jump elsewhere on return



- Defences
  - Use language with bound checking
  - Computer can put padding between data and return addresses and check if stack data has been overwritten
  - Hardware can provide ability for non-executable stack, so no code on the stack can be run.
  - OS can place parts of stack at random virtual addresses

#### 2.1.3.2 Integer Overflow

- Integer overflow/underflow is when a machine integer wraps around due to the system system reaching the maximum/minimum values the architecture can handle.
- A attacker can exploit this by passing values to a program that will trigger overflow.

#### 2.1.3.3 Format String Vulnerabilities

- If `printf(buffer)` is used instead of `printf("%s", buffer)`, an attacker can replace the printed valued with their own format string to cause unexpected behaviour or arbitrary code execution
- `"%s%s%s"` could cause a crash, `"%x%x%x%x"` could dump parts of the stack, and `%n` can write an address to the stack

#### 2.1.3.4 Incomplete mediation

- Incomplete mediation occurs when the application accepts incorrect data from the user
- Incomplete mediation could also lead to SQL injection or buffer overflows.