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

Lecture 4: September 20, 2018

Lecturer: Ian Goldberg *Notes By: Harsh Mistry*

4.1 Program Security Continued

4.1.1 Non Malicious Flaws Continued

4.1.1.1 Side Channels

- Side Channel Attacks are powerful attacks that simply involve observing how computers behave in response to sensitive data.
- The attacker just has to abuse the nature of how the computer works or how users interact with them.
- **Constant time** implementations are programs that minimize number of side channels

4.1.2 Designing Programs for Security

4.1.2.1 Modularity

- Break the module into a number of small pieces each responsible for a specific portion
- Modules should also have low coupling, as in they should not be tightly related to other modules.
- This helps localize errors and flaws to a specific module.

4.1.2.2 Encapsulation

- Ensure all information is contained, only share information necessary
- Ideally, developers for each module should not need to know how the other modules functions. They should just need to know the API

4.1.2.3 Information Hiding

- Expanding on encapsulation, external modules should not have access to a modules internal information.
- In essence, internal module data should be hidden

4.1.2.4 Mutual Suspicion

- Even though modules work together, each individual module should validate input
- This is done to ensure that security flaws don't propagate through the applications.
- Basically, each module needs to protect against attack vectors exists in other modules

4.1.2.5 Confinement

- Expanding on mutual suspicion, modules should also be suspicious any other modules called.
- An example of this is **Sandboxing**

4.1.3 Security Controls

4.1.3.1 Static code analysis

- Static code analysis tools are applications that attempt to scan code for common security flaws.
- Often looks for buffer overflows, but some can point out TOCTTOU errors and other flaws.

4.1.3.2 Formal Methods

- Utilize program verification methods to formally prove an application performs to the specification

4.1.3.3 Genetic Diversity

- The reason worms and viruses are able to propagate is due to the fact alot of computers are running the same vulnerable code
- This could potentially be avoided by using different implementations.

4.1.3.4 Code Review

- Code review is the single most effective way to find faults once the code has been written
- Kinds of code review
  - Someone writes the code and another dedicated person review the code. This is the least effective method
  - Guided walk-through. Extremely common for safety-critical systems.
  - Easter Egg Review. Author inserts intentional security flaws to ensure the reviewer actually reviewed it.