Secure Programming

—— Software Security

胡天磊, Dr. HU Tianlei

Associate Professor

College of Computer Science, Zhejiang Univ.

htl@zju.edu.cn

Outline

- The Software Security Problem
  - The History
  - Categories of Security Flaws
    - Architecture/Design
    - Implementation
    - Operational
  - The Standard and The Process
- Software Security: More than Just Coding

The Software Security Problem

Traditional Security is Reactive

Perimeter defense (firewalls)

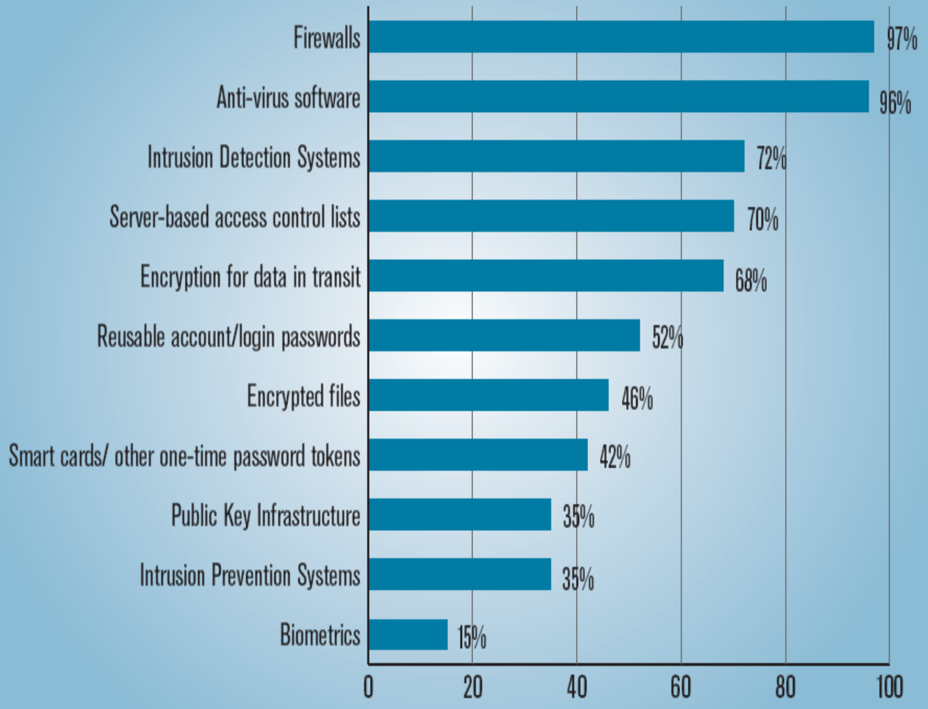
Intrusion detection

Over-reliance on cryptography

Penetrate and patch

Penetration testing

Figure 17. Security Technologies Used



CSI/FBI 2005 Computer Crime and Security Survey
Source: Computer Security Institute

2005: 687 Respondents

The Problem is Software

"75% of hacks happen at the application."

- Theresa Lanowitz, Gartner Inc.

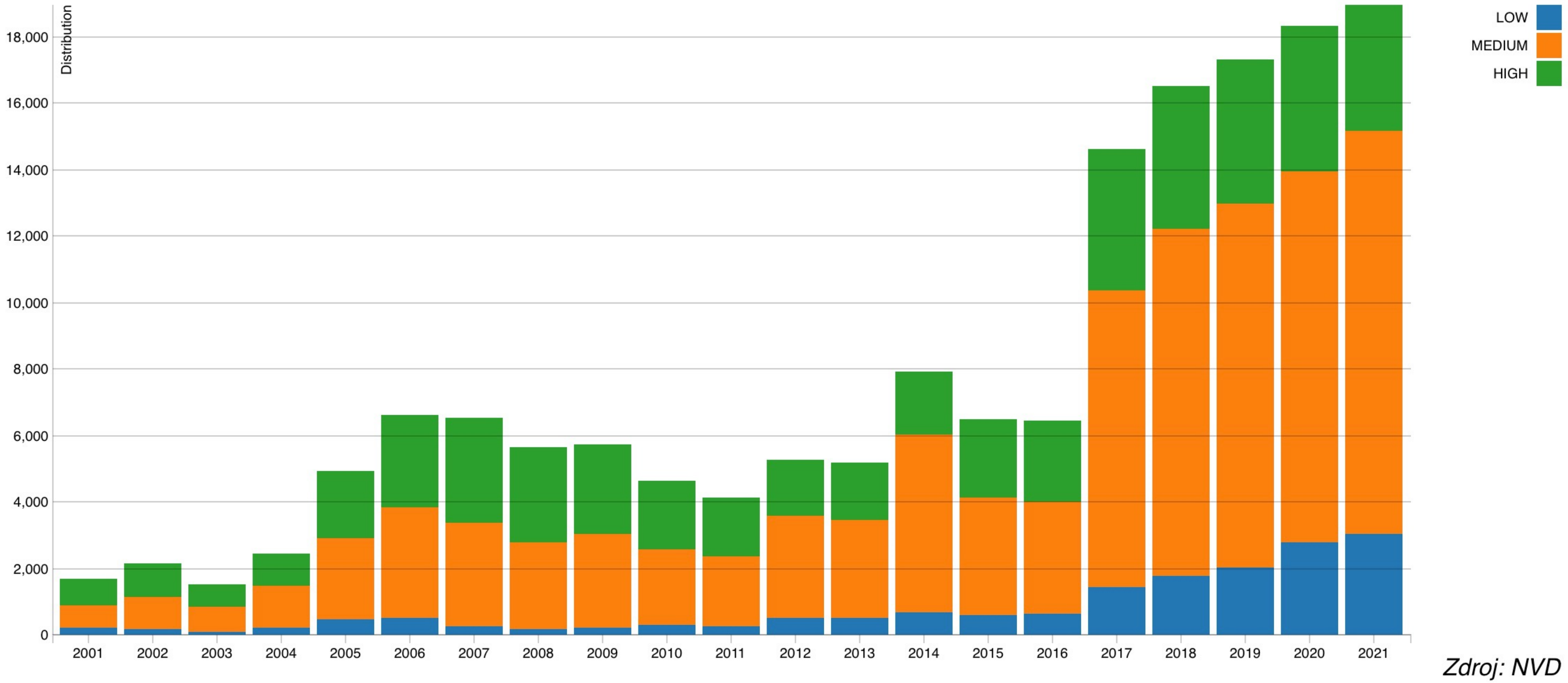
"92% of reported vulnerabilities are in apps, not networks."

- NIST

"64% of developers are not confident in their ability to write secure code."

- Bill Gates

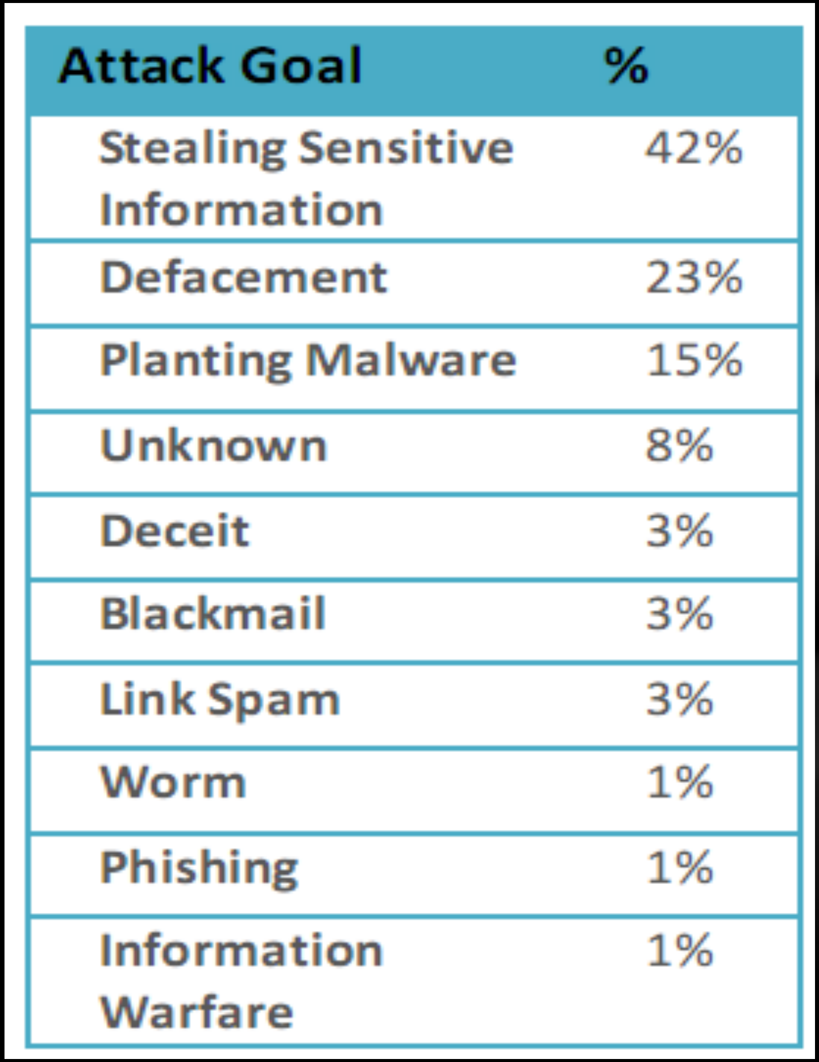
A Continuously Growing Problem



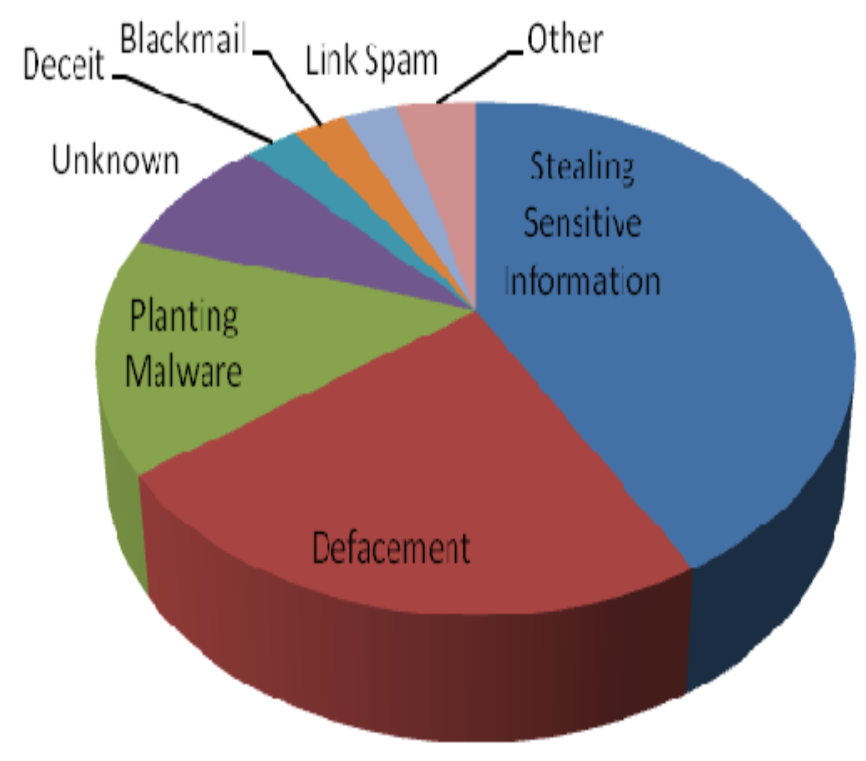
Zdroj: NVD

GFI SOFTWARE: 2021 WILL BE THE YEAR WITH THE MOST VULNERABILITIES DISCOVERED

Motivations



Attack Goal	%
Stealing Sensitive Information	42%
Defacement	23%
Planting Malware	15%
Unknown	8%
Deceit	3%
Blackmail	3%
Link Spam	3%
Worm	1%
Phishing	1%
Information Warfare	1%



Why is Software Security poor?

- **Security is seen as something that gets in the way of software functionality.**
  - PM: “Let’s add a login function to our website! ”
  - Programmer: “We even do not have the concept of user & pass! Login by what? ”
  - PM: “Eh… Probably by QQ? ”
- **Security is difficult to assess and quantify.**
  - Boss: “Build a secure application for me! ”
  - Programmer: “As secure as? ”
  - Boss: “The securest! ”
- **Security is often not a primary skill or interest of software developers.**
  - Interviewee: “I am a very good programmer in C/C++.”
  - Interviewer: “What’s the concept of buffer overflow? How to exploit it? ”
  - Interviewee: “…”
- **Time spent on security is time not spent on adding new and interesting functionality.**
  - Programmer: “There are two vulnerabilities in legacy code; we have to fix them! ”
  - PM: “How long does it take? ”
  - Programmer: “I don’t know, probably two weeks? ”
  - PM: “Oh, don’t do that silly thing! Add these two functions first, bringing us 20% more revenue! ”

Trinity of Trouble

Connectivity

- Ubiquitous Internet; wireless & mobile computing.

Complexity

- Networked, distributed code that can interact with intermediate caches, ad proxies, etc.

Extensibility

- Systems evolve unexpectedly, e.g., web browsers, which support many formats, add-ons, plugins, programming languages, etc.

Categories of Security Flaws

- Architectural/design-level flaws:
  - Security issues that the original design should have considered or solved correctly.
- Implementation flaws:
  - Errors were made in coding the design.
- Operational flaws:
  - Problems arise from how software is installed or configured.

Architecture/Design Flaws

- Race Condition
  - Application checks access control, then accesses a file as two separate steps, permitting an attacker to race program and substitute the accessible file for one that’s not allowed.
- Replay Attack
  - If an attacker can record a transaction between a client and server simultaneously, then replay part of the conversation without the application detecting it, a replay attack is possible.
- Sniffing
  - In the early stage, since only authorized users could directly access the network on the Internet, protocols like telnet send passwords in the clear.

Implementation Flaws

- Buffer overflow
  - Application with fixed-size buffer accepts unlimited length input, writing data into memory beyond buffer in languages w/o bounds checking like C/C++.
- Input validation
  - An application doesn’t check that input has a valid format, such as not checking for “../” sequences in pathnames, allowing attackers to traverse the directory tree to access any file.
- Back door
  - The programmer writes special code to bypass the access control system, often for debugging or maintenance.

Operational Flaws

- Denial of service
  - The system lacks enough resources or the ability to monitor resources to sustain availability under too many requests.
- Default accounts
  - Default username/password pairs allow access to anyone who knows the default configuration.
- Password cracking
  - Poor passwords can be guessed by software using dictionaries and permutation algorithms.

SSE (Secure Software Engineering) Objectives

1. **Dependability:** software functions only as intended;
2. **Trustworthiness**: No exploitable vulnerabilities or malicious logic exist in the software;
3. **Resilience:** If compromised, the damage will be minimized, and it will recover quickly to an acceptable level of operating capacity;
4. **Conformance**: to requirements and applicable standards and procedures.

Security Standards and Certs

ISO 15408 Common Criteria

PCI Data Security Standard

- *Requirement 6:* Develop and maintain secure systems and applications

SANS GIAC Secure Software Programmer

- http://www.sans-ssi.org/

Many standards indirectly impact SSE

- FISMA
- SOX

Secure Development Processes

- CLASP (Comprehensive, Lightweight Application Security Process)
- Correctness-by-Construction (formal methods-based process from Praxis Critical Systems)
- MS SDL (Microsoft Secure Development Lifecycle)
- SSE CMM (Secure Software Engineering Capability Maturity Model)
- TSP-Secure (Team Software Process for Secure Software Development)
- Touchpoints

Software Security Practices

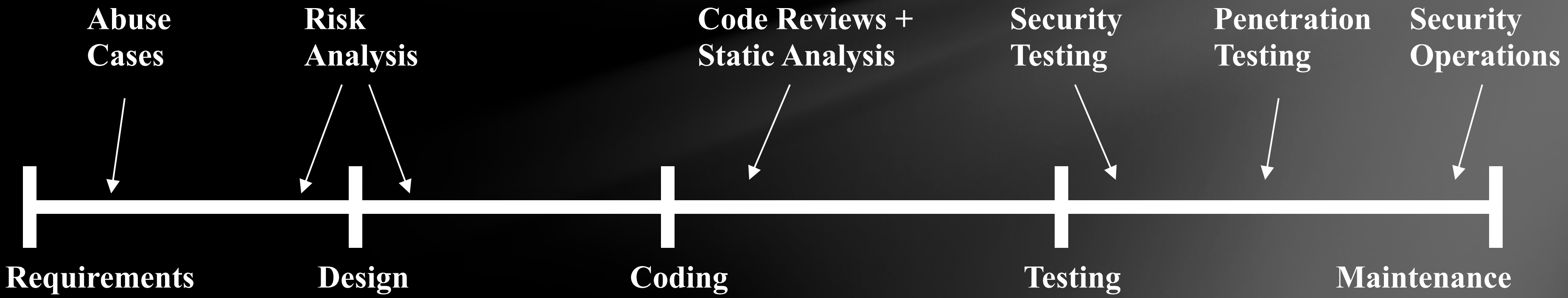
Software Security

- More than just coding!
- Security lies in every phase of the software lifecycle:
  - Requirements
  - Design
  - Coding
  - Testing
  - Maintenance

Requirements Design Coding Testing Maintenance

Software Security Practices

1. Abuse Cases
2. Risk Analysis
3. Code Reviews
4. Security Testing
5. Penetration Testing
6. Security Operations



Abuse Cases

Anti-requirements ---

Think about what software should not do.

A use case from an adversary's point of view.

- Obtain Another User's Privacy Data.
- Alter Item Price.
- Deny Service to Application.

Developing abuse cases

Informed brainstorming: attack patterns, risks.

Architectural Risk Analysis

Fix design flaws, not implementation bugs.

Risk analysis steps

1. Develop an architecture model.
2. Identify threats and possible vulnerabilities.
3. Develop attack scenarios.
4. Rank risks based on probability and impact.
5. Develop a mitigation strategy.
6. Report findings

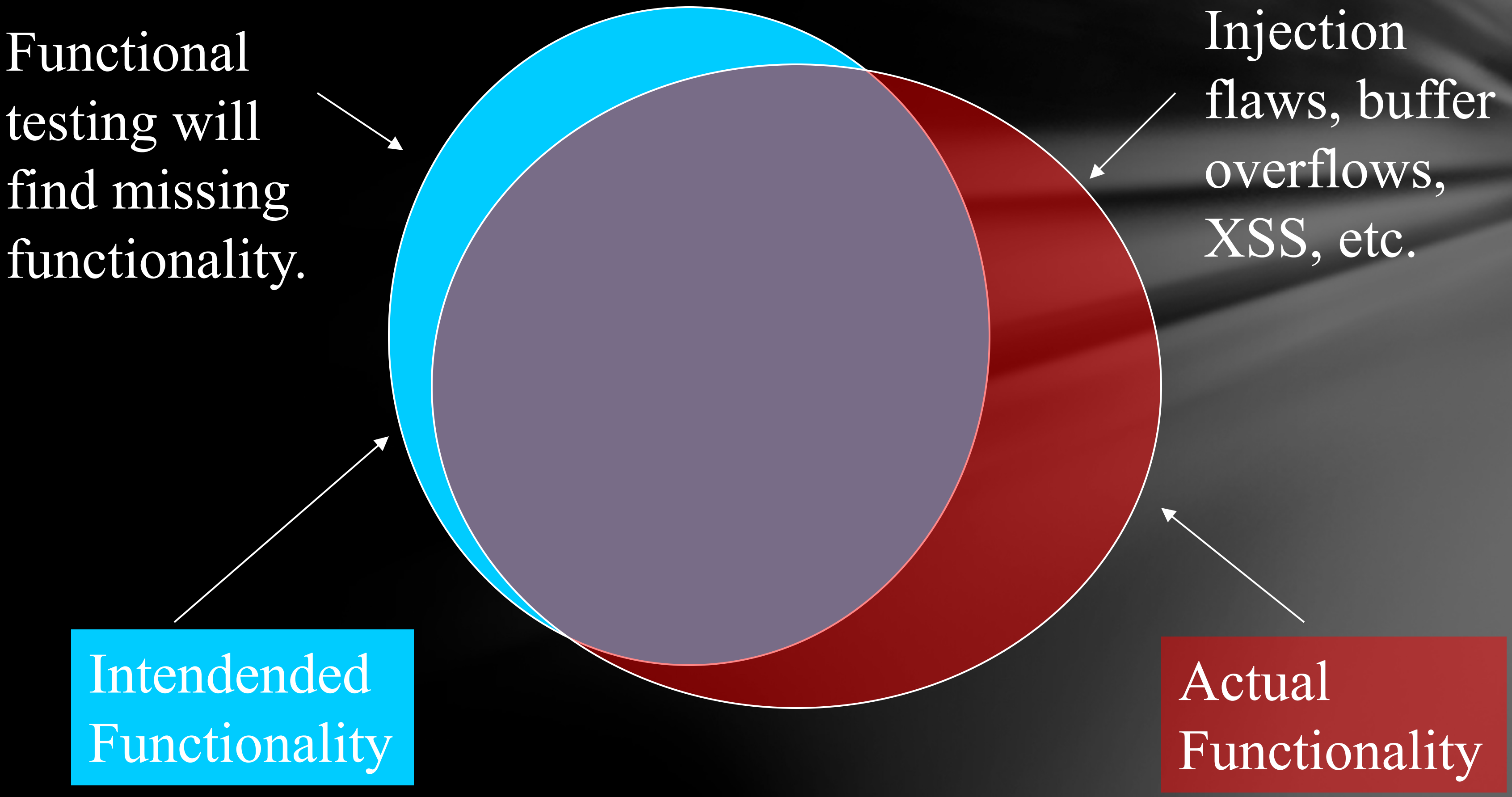
Code Reviews

Fix implementation bugs, not design flaws.

Benefits of code reviews

1. Find defects sooner in the lifecycle.
2. Find defects with less effort than testing.
3. Find different defects than testing.
4. Educate developers about security flaws.

Security Testing



Functional testing will find missing functionality.

Injection flaws, buffer overflows, XSS, etc.

Intended Functionality

Actual Functionality

Security Testing

Two types of testing

- **Functional:** verify security mechanisms.
- **Adversarial:** verify resistance to attacks generated during risk analysis.

Different from traditional penetration testing

- White box.
- Use risk analysis to build tests.
- Measure security against risk model.

Penetration Testing

Test software in the deployed environment.

Allocate time at the end of development to test.

- Often time-boxed: test for n days.
- Schedule slips often reduce testing time.
- Fixing flaws is expensive late in the lifecycle.

Penetration testing tools

- Test common vulnerability types against inputs.
- Fuzzing: send random data to inputs.
- Do not require an understanding of the application structure or purpose.

Security Operations

User security notes

- Software should be secure by default.
- Enabling certain features may have risks.
- User needs to be informed of security risks.

Incident response

- What happens when a vulnerability is reported?
- How do you communicate with users?
- How do you send updates to users?

Review

- Categories of Security Flaws
  - Architecture/design
  - Implementation
  - Operational
- Secure Software Engineering
  - More Than Just Coding!
  - Security lies everywhere in Software Lifecycle.