Web Application Security

James Walden
Northern Kentucky University
waldenj@nku.edu

Is your web site secure?

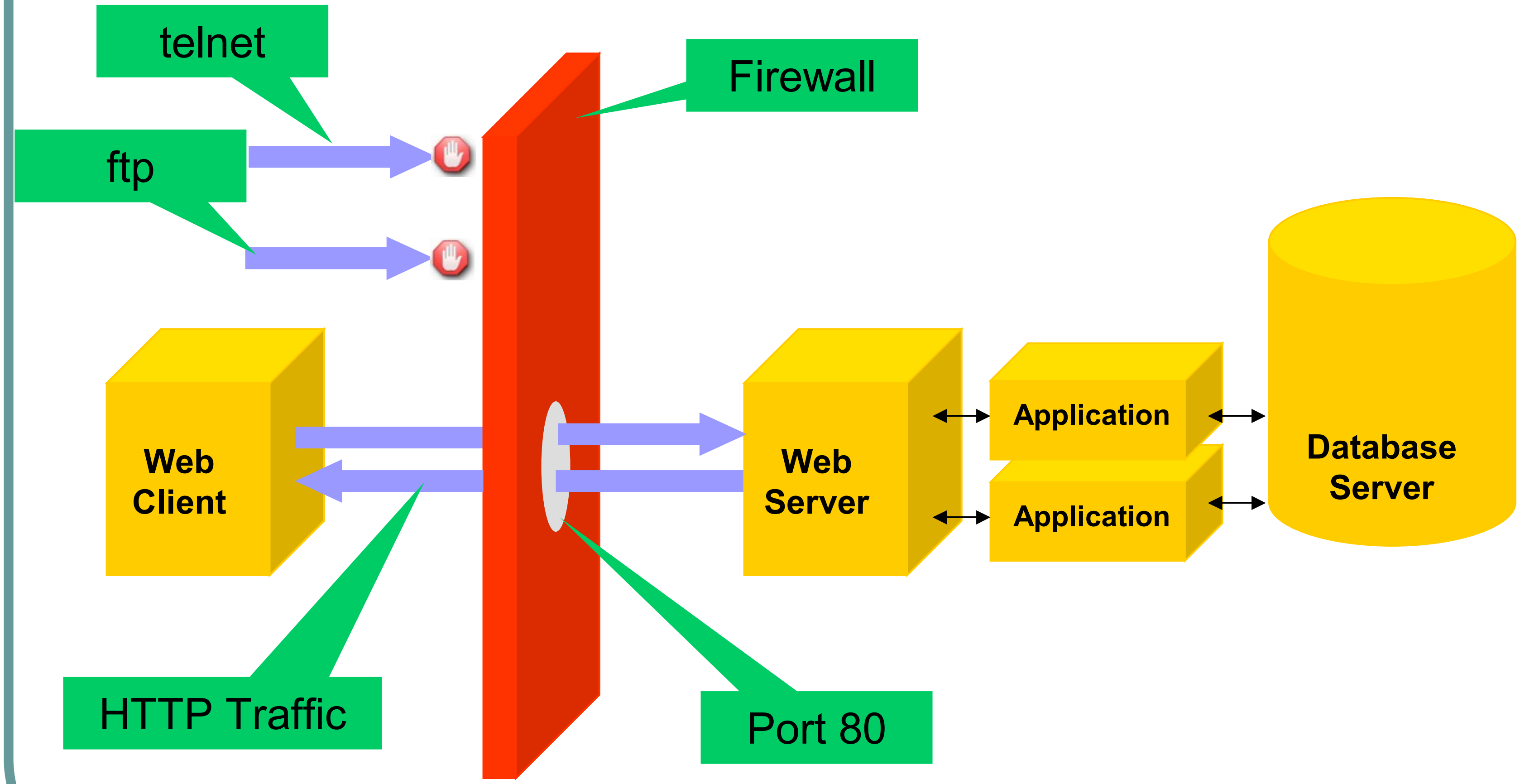
Is your web site secure?

Is your web site secure?

Yes, we deployed SSL, firewall, etc.

- Does SSL protect all communications?
- What about stored data?
- What about injection attacks and XSS?

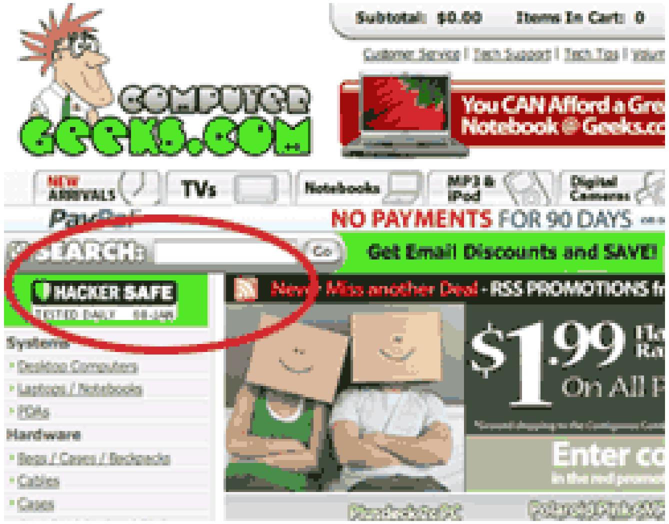
Firewalls don't protect web apps



Is your web site secure?

Yes, we're certified as being secure.

- PCI scans quarterly; apps change weekly.
- Geeks.com, certified HackerSafe by McAfee, lost thousands of CCs in 2007.



Subtotal: $0.00 Items In Cart: 0

Is your web site secure?

Yes, we have logs of blocked attacks.

- Better, you have some real evidence.
- Did you log non-blocked requests too?

Is your web site secure?

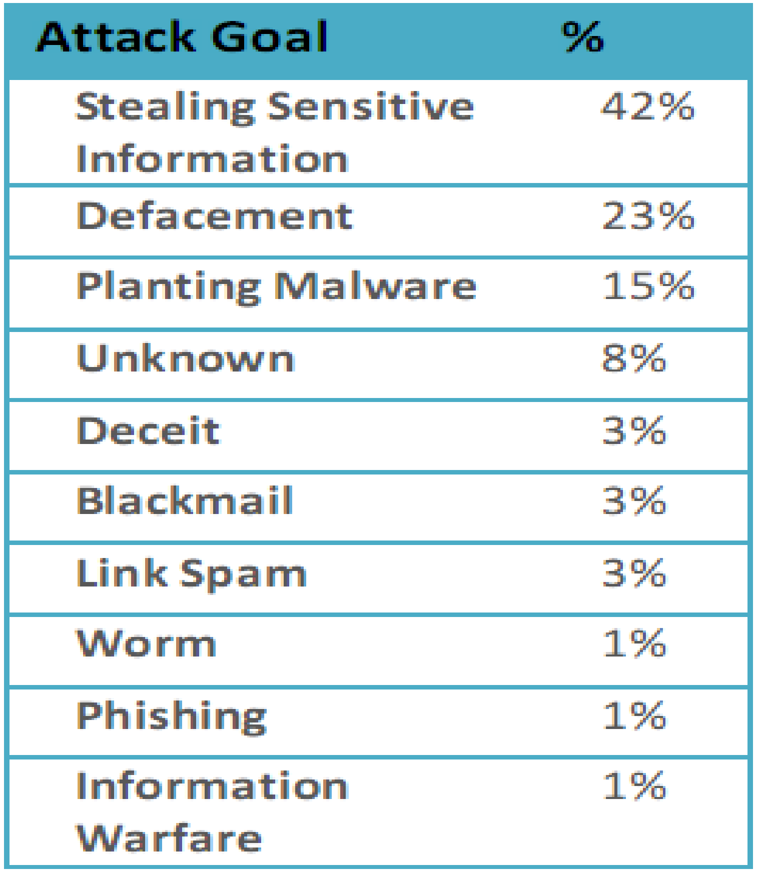
Yes, we have a SDLC and record network, host, and application-based logs.

- Secure Development LifeCycle
  - Risk analysis
  - Secure design
  - Code reviews
  - Security testing
- Correlate logs for multi-perspective picture.

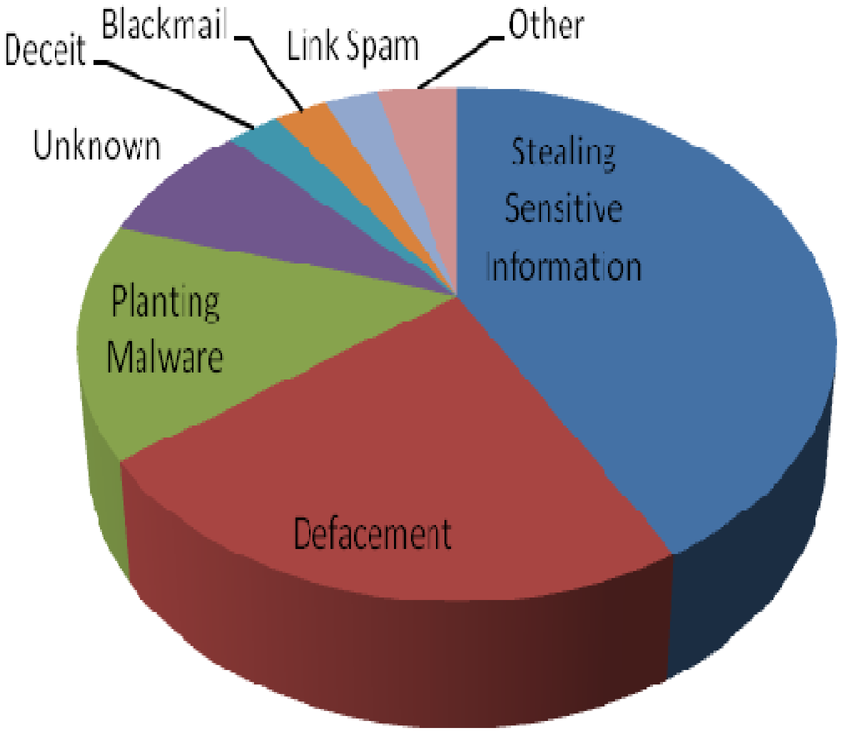
Topics

1. The Problem of Software Security
2. Web Application Vulnerabilities
3. SQL Injection
4. Software Security Practices

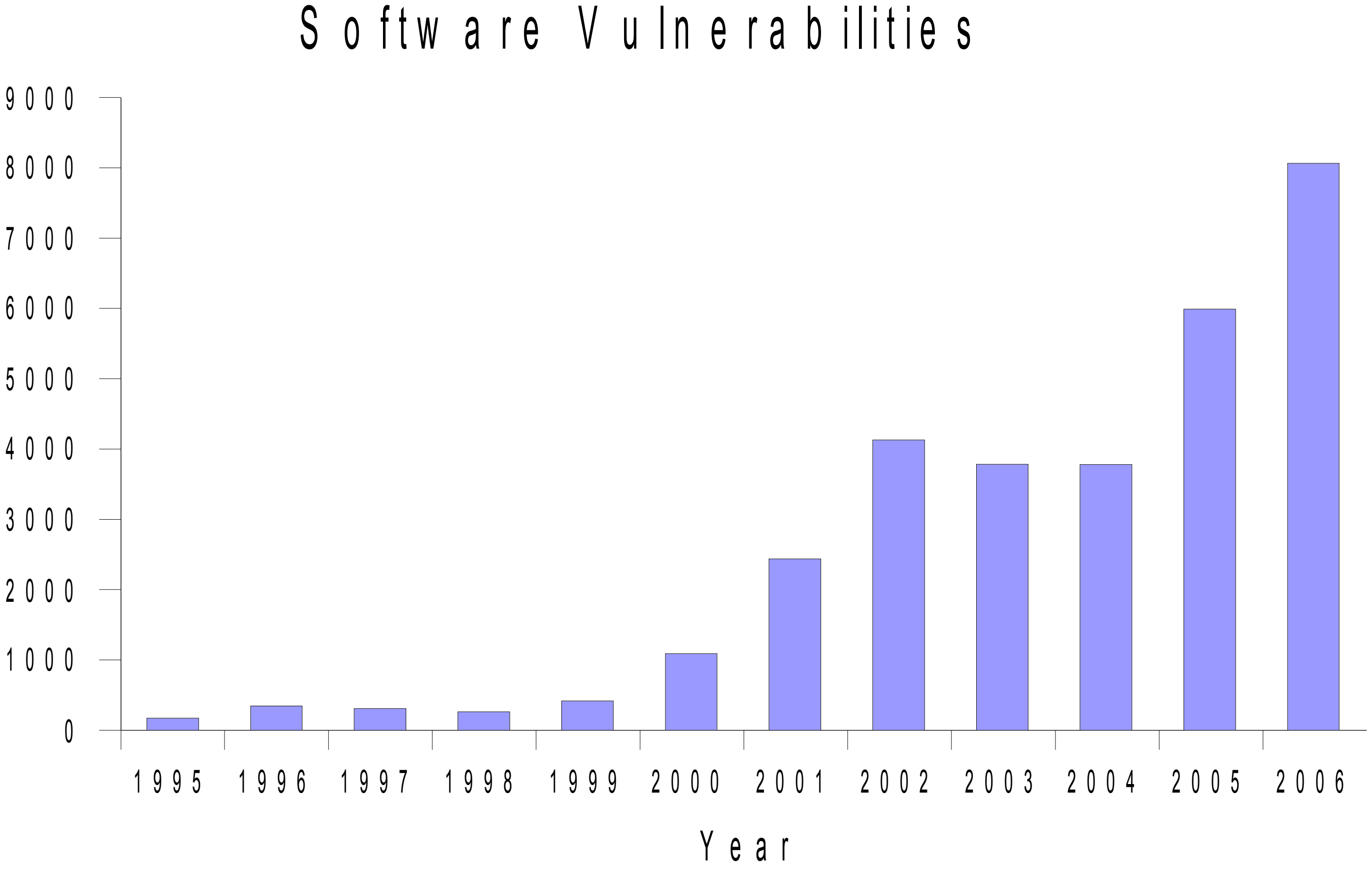
Reasons for Attacking Web Apps



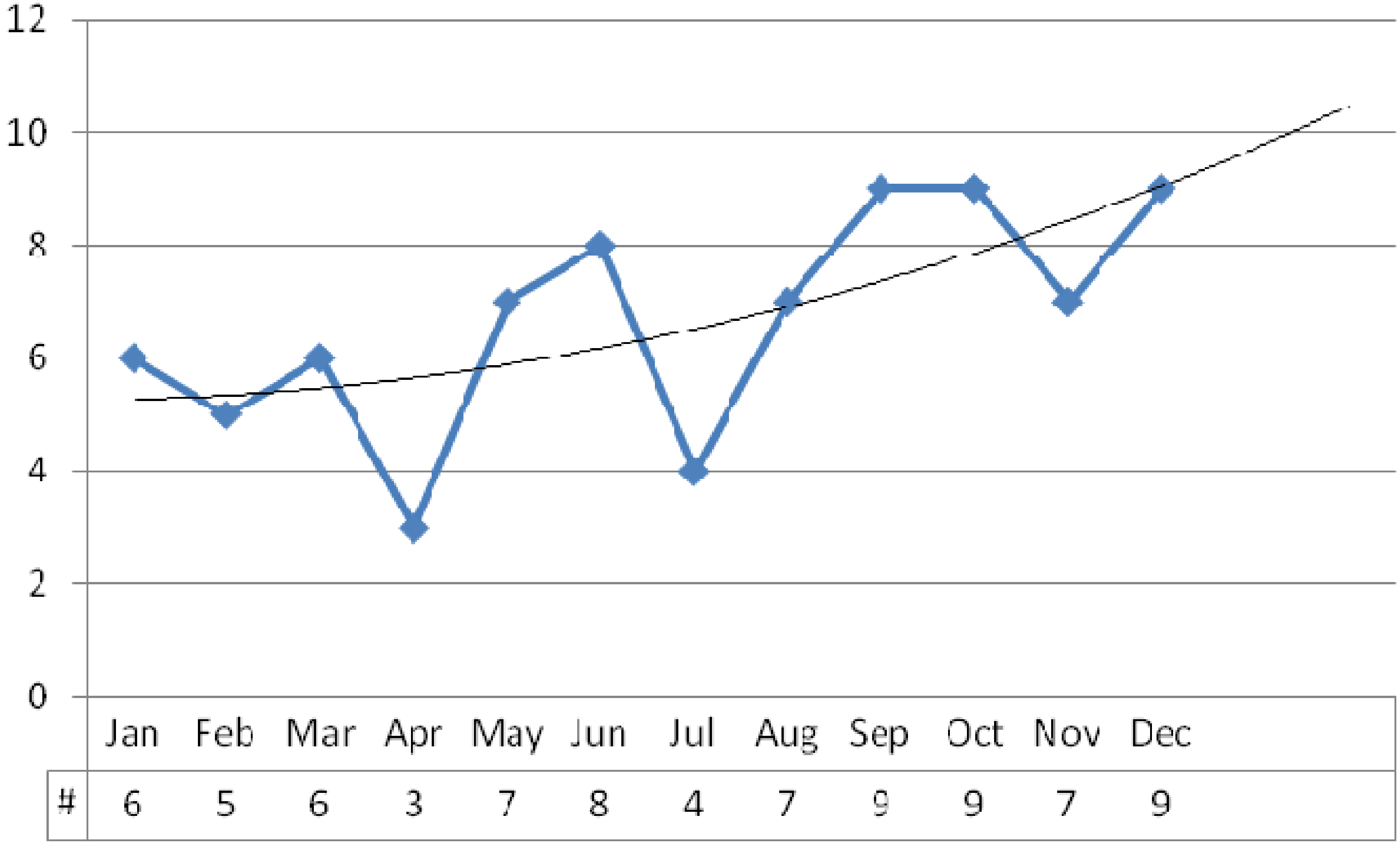
Attack Goal	%
Stealing Sensitive Information	42%
Defacement	23%
Planting Malware	15%
Unknown	8%
Deceit	3%
Blackmail	3%
Link Spam	3%
Worm	1%
Phishing	1%
Information Warfare	1%



A Growing Problem



Web Application Exploits 2007



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
#	6	5	6	3	7	8	4	7	9	9	7	9

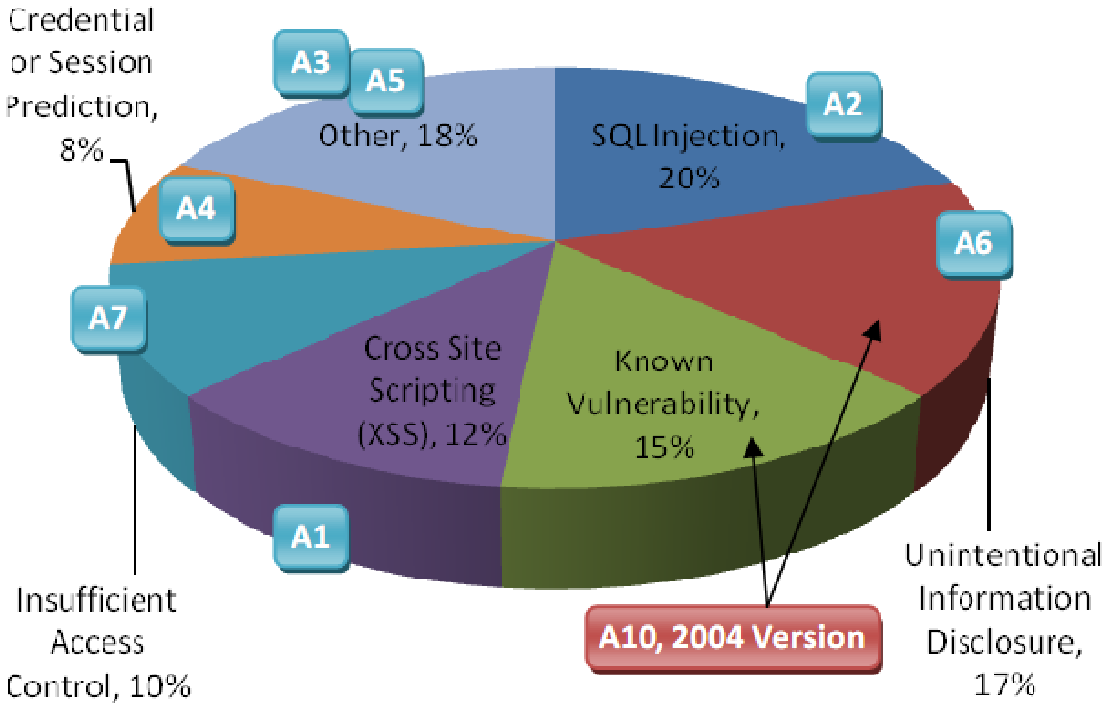
The source of the problem

"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

Web Application Vulnerabilities

Attack/Vulnerability Used	%
SQL Injection	20%
Unintentional Information Disclosure	17%
Known Vulnerability	15%
Cross Site Scripting (XSS)	12%
Insufficient Access Control	10%
Credential/Session Prediction	8%
OS Commanding	3%
Misconfiguration	3%
Insufficient Anti-automation	3%
Denial of Service	3%
Redirection	2%
Insufficient Session Expiration	2%
Cross Site Request Forgery (CSRF)	2%

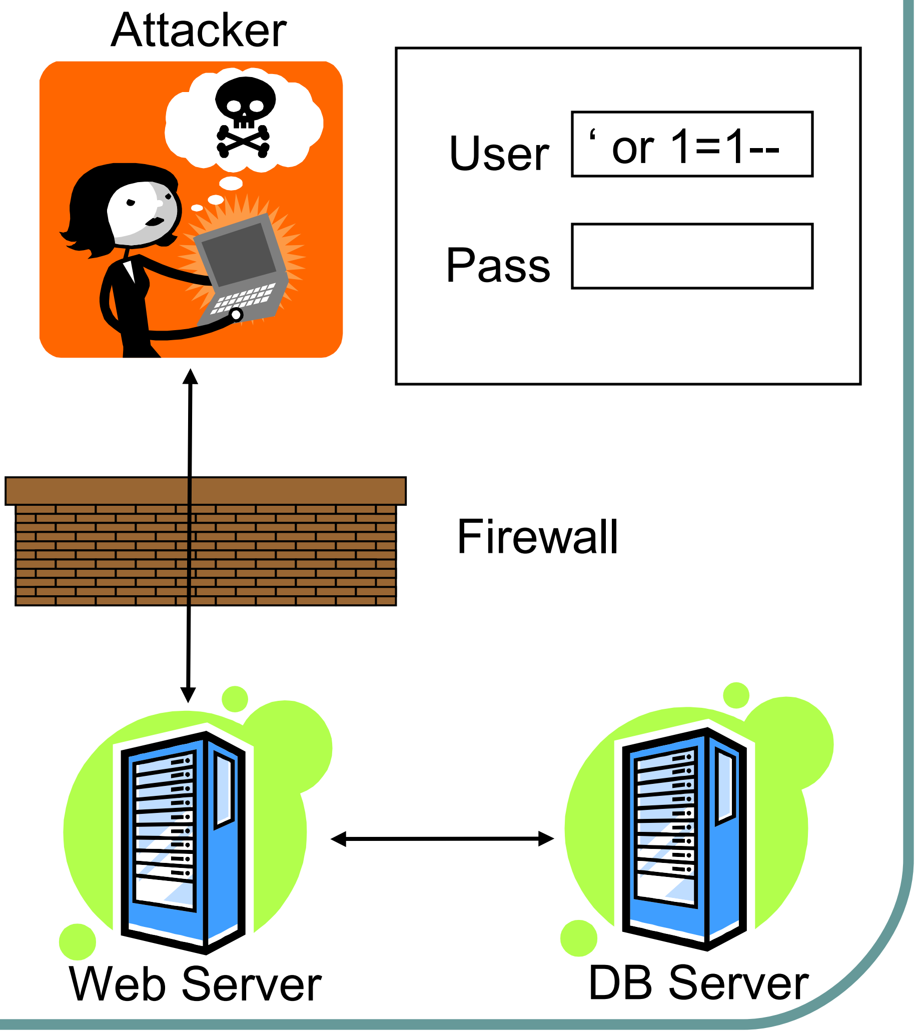


Injection

- Injection attacks trick an application into including unintended commands in the data send to an interpreter.
- Interpreters
  - Interpret strings as commands.
  - Ex: SQL, shell (cmd.exe, bash), LDAP, XPath
- Key Idea
  - Input data from the application is executed as code by the interpreter.

SQL Injection

1. App sends form to user.
2. Attacker submits form with SQL exploit data.
3. Application builds string with exploit data.
4. Application sends SQL query to DB.
5. DB executes query, including exploit, sends data back to application.
6. Application returns data to user.



User ‘ or 1=1--

Pass

SQL Injection in PHP

```
$link = mysql_connect($DB_HOST, $DB_USERNAME, $DB_PASSWORD) or die ("Couldn't connect: " . mysql_error());
mysql_select_db($DB_DATABASE);
$query = "select count(*) from users where username = '$username' and password = '$password'";
$result = mysql_query($query);
```

SQL Injection Attack #1

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 empty OR 1=1, which is always true, permitting access.

SQL Injection Attack #2

Database Modification Attack:

`password =` foo’; **delete from table** *users* **where** *username* **like** ‘%

DB executes ***two*** SQL statements:

select count(*) from *users* **where** *username* = ‘*user*’ **and** *password* = ‘*foo*’

delete from table *users* **where** *username* **like** ‘%’

SQL Injection Demo

SQL Injection Demo

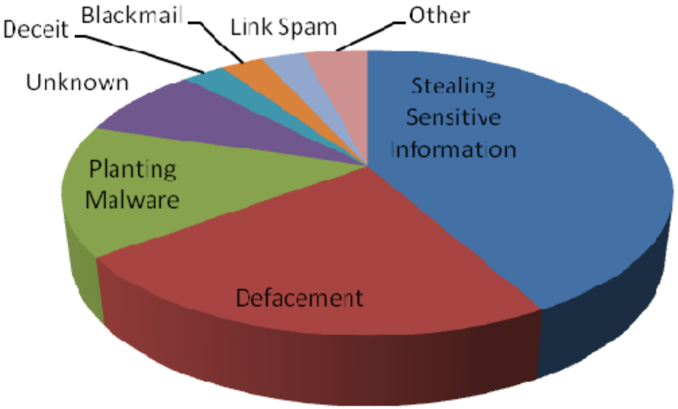
Impact of SQL Injection

SELECT SSN FROM USERS WHERE UID='$UID'

INPUT	RESULT
5	Returns info for user with UID 5.
' OR 1=1--	Returns info for all users.
' UNION SELECT Field FROM Table WHERE 1=1--	Returns all rows from another table.
';DROP TABLE USERS--	Deletes the users table.
';master.dbo.xp_cmdshell 'cmd.exe format c: /q /yes' --	Formats C: drive of database server if you're running MS SQL Server and extended procedures aren't disabled.

Impact of SQL Injection

1. Leakage of sensitive information.
2. Reputation decline.
3. Modification of sensitive information.
4. Loss of control of db server.
5. Data loss.
6. Denial of service.



The Problem: String Building

Building a SQL command string with user input in any language is dangerous.

- Variable interpolation.
- String concatenation with variables.
- String format functions like sprintf().
- String templating with variable replacement.

Mitigating SQL Injection

Partially Effective Mitigations

- Blacklists
- Stored Procedures

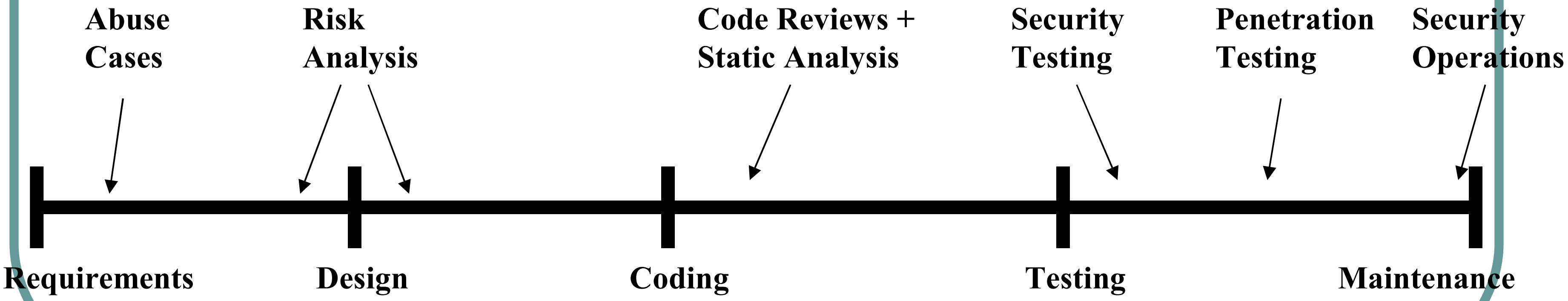
Effective Mitigations

- Whitelists
- Prepared Queries

Software Security Practices

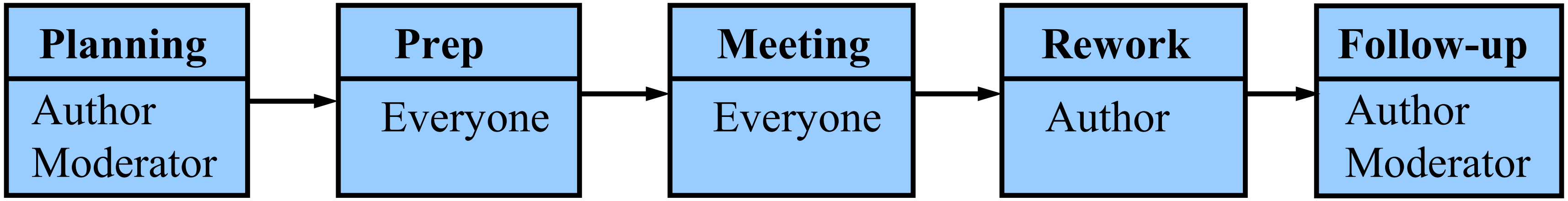
1. Code Reviews
2. Risk Analysis
3. Penetration Testing

1. Security Testing
2. Abuse Cases
3. Security Operations



Code Reviews

Fix implementation bugs, not design flaws.



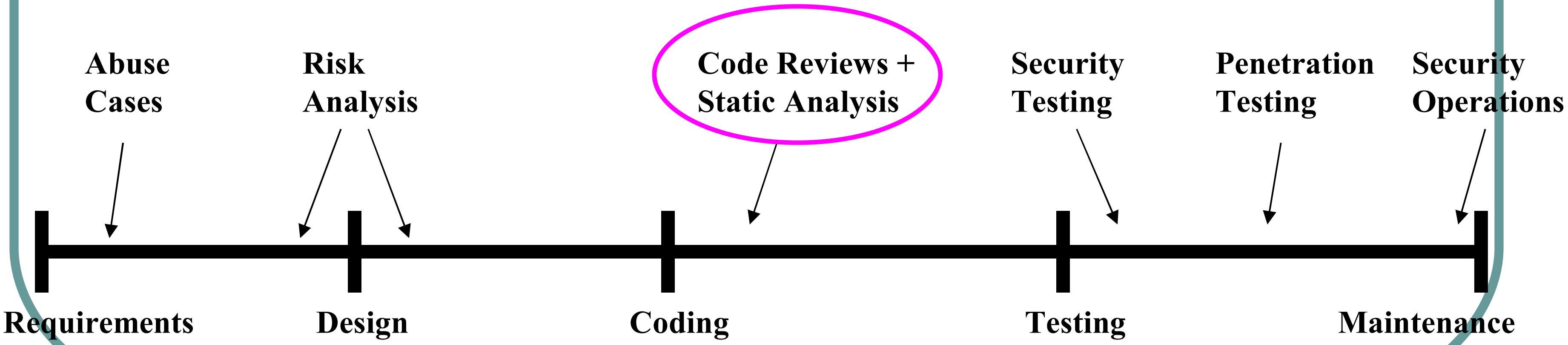
Planning
Author
Moderator

Prep
Everyone

Meeting
Everyone

Rework
Author

Follow-up
Author
Moderator



Abuse Cases

Risk Analysis

Code Reviews + Static Analysis

Security Testing

Penetration Testing

Security Operations

Requirements

Design

Coding

Testing

Maintenance

Benefits of 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.)

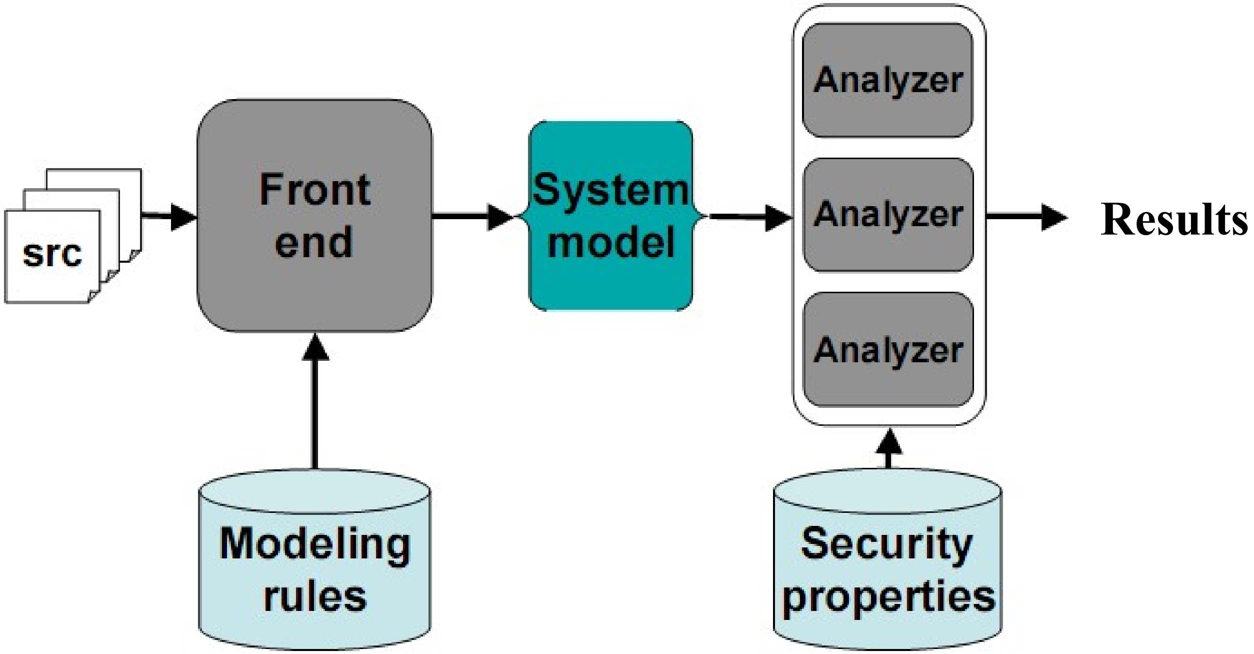
4. Educate developers about security bugs.

   (Developers frequently make the same mistakes.)

Static Analysis

Automated assistance for code reviews

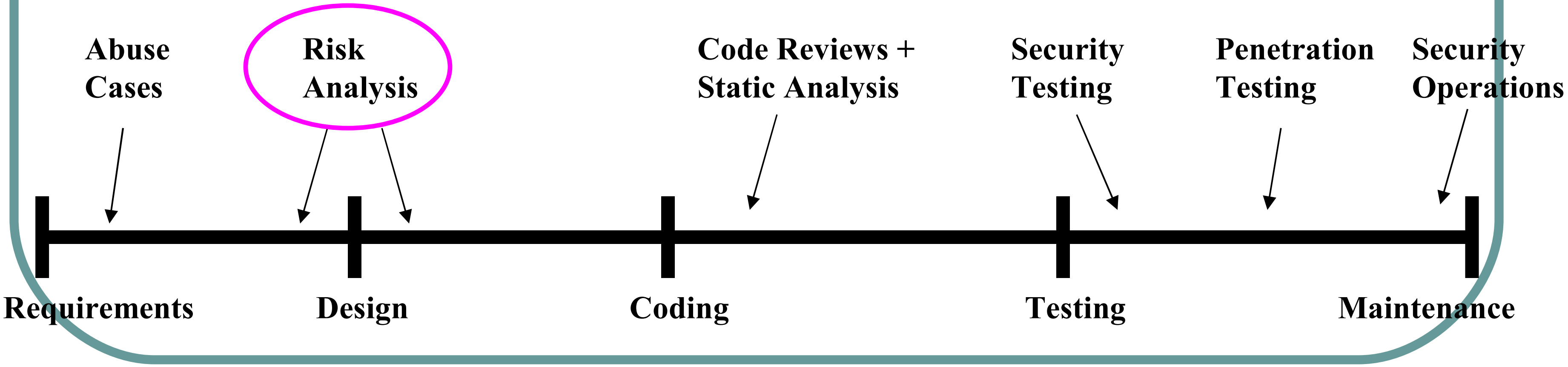
- Speed: review code faster than humans can
- Accuracy: hundreds of secure coding rules



Architectural Risk Analysis

Fix design flaws, not implementation bugs.

1. Develop an architecture model.
2. Model threats and attack scenarios.
3. Rank risks based on probability and impact.
4. Develop mitigation strategy.



Threat Modeling

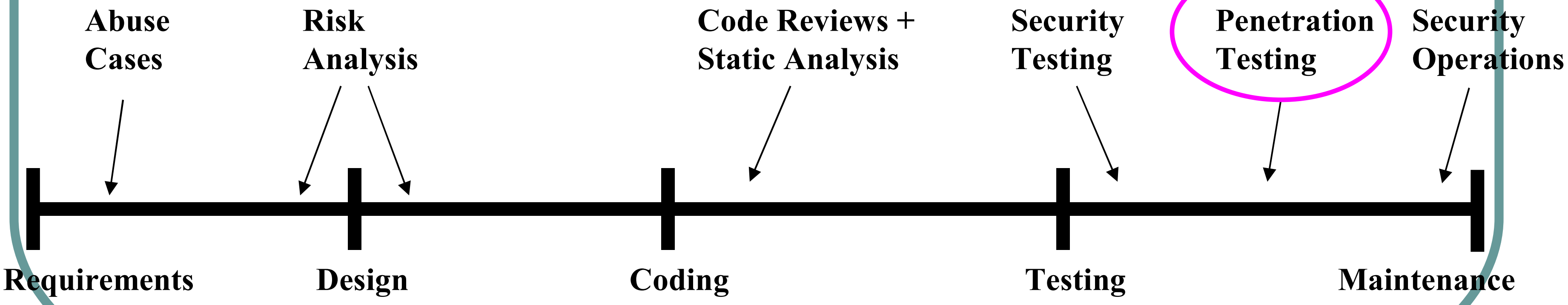
1. Identify System Assets.
   - System resources that an adversary might attempt to access, modify, or steal.
   - Ex: credit cards, network bandwidth, user access.
2. Identify Entry Points.
   - Data or control transfers between systems.
   - Ex: network sockets, RPCs, web forms, files
3. Determine Trust Levels.
   - Privileges external entities have to legitimately use system resources.

Penetration Testing

Test software in deployed environment by attacking it.

Allocate time at end of development to test.

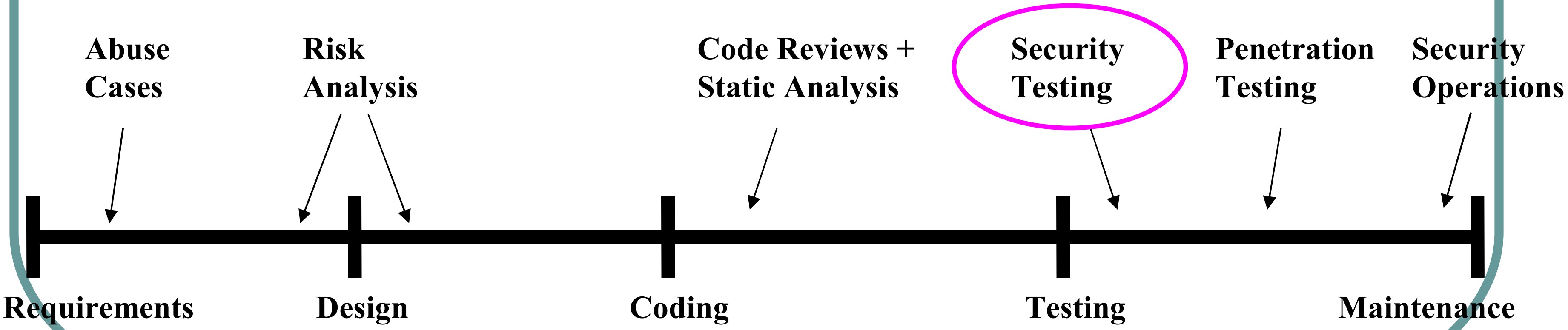
- Time-boxed: test for *n* days.
- May be done by an external consultant.



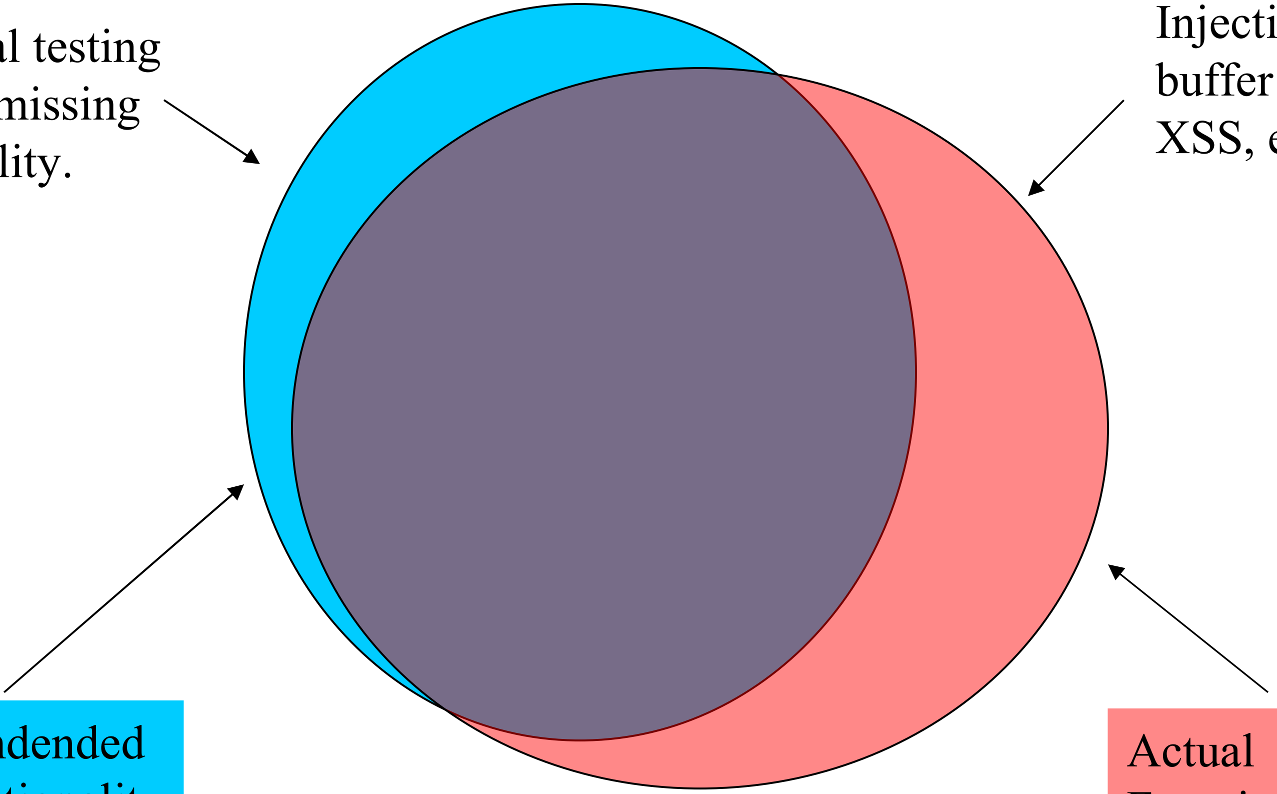
Security Testing

Different from penetration testing

- White box (source code is available.)
- Use risk analysis to build tests.
- Measure security against risk model.



Security Testing



Functional testing will find missing functionality.

Injection flaws, buffer overflows, XSS, etc.

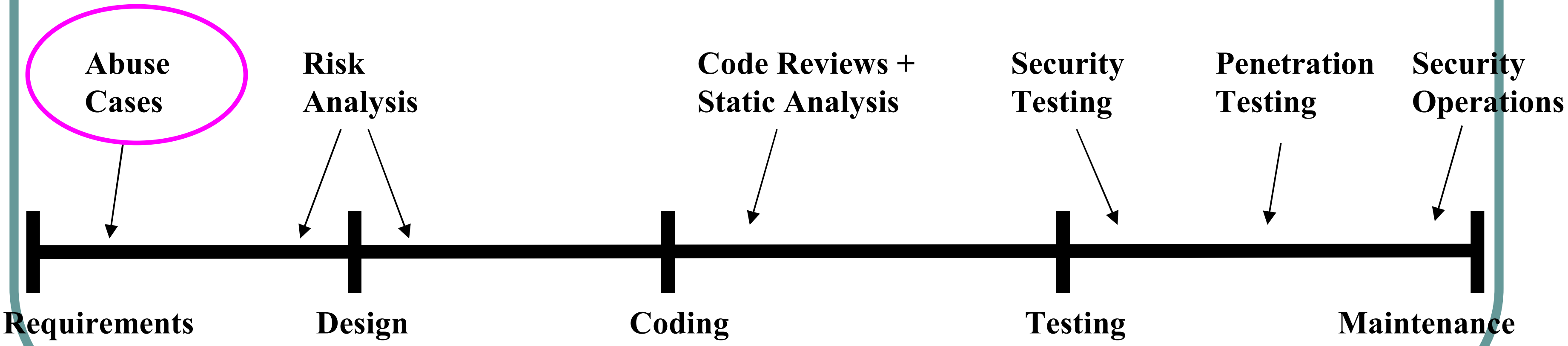
Intendended Functionality

Actual Functionality

Abuse Cases

Anti-requirements

- Think explicitly about what program shouldn't do.
- A use case from an adversary's point of view.



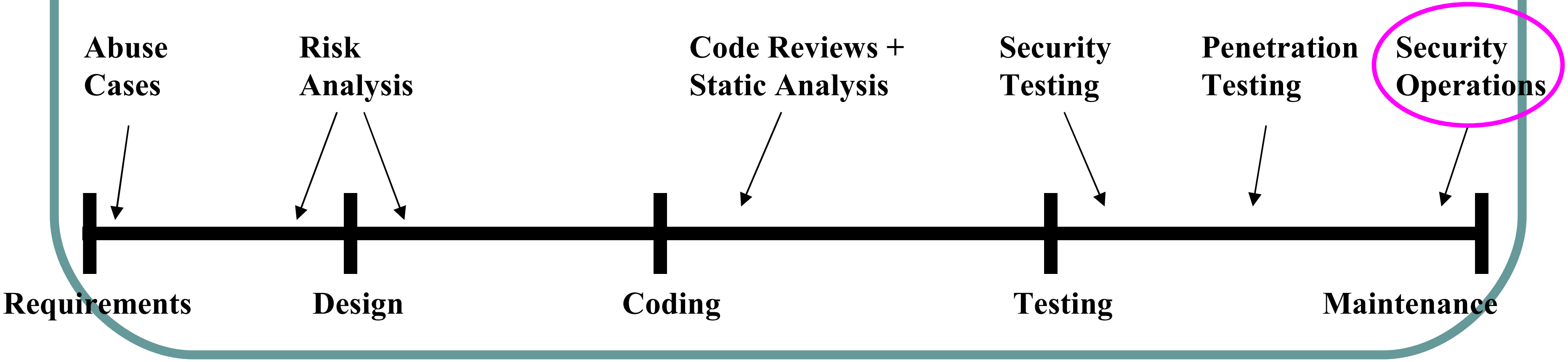
Security Operations

Deploying security

- Secure default configuration.
- Web application firewall for defense in depth.

Incident response

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



Conclusions

- Web applications are a primary target.
  - Sensitive information
  - Defacement
  - Malware distribution
- Software Security ≠ Security Features
  - SSL will not make your site secure.
  - Firewalls will not make your site secure.
- Improving software development
  - Code reviews.
  - Risk analysis.
  - Security testing.

References

1. Mark Dowd, John McDonald, Justin Schuh, *The Art of Software Security Assessment*, Addison-Wesley, 2007.
2. Mitre, Common Weaknesses – Vulnerability Trends, http://cwe.mitre.org/documents/vuln-trends.html, 2007.

- Gary McGraw, *Software Security*, Addison-Wesley, 2006.
- J.D. Meier, et. al., *Improving Web Application Security: Threats and Countermeasures*, Microsoft, http://msdn2.microsoft.com/en-us/library/aa302418.aspx, 2006.
- OWASP Top 10, http://www.owasp.org/index.php/OWASP_Top_Ten_Project, 2007.
- Ivan Ristic, Web Application Firewalls: When Are They Useful?, OWASP AppSec EU 2006.
- Joel Scambray, Mike Shema, and Caleb Sima, *Hacking Exposed: Web Applications, 2nd edition*, Addison-Wesley, 2006.
- Dafydd Stuttard and Marcus Pinto, *Web Application Hacker's Handbook*, Wiley, 2007.
- WASC, "Web Application Incidents Annual Report 2007," https://bsn.breach.com/downloads/whid/The%20Web%20Hacking%20Incide , 2008.