Building Security Into Applications

Marco Morana
OWASP Chapter Lead
Blue Ash, July 30th 2008

OWASP
Cincinnati Chapter Meetings

The OWASP Foundation
http://www.owasp.org

Agenda

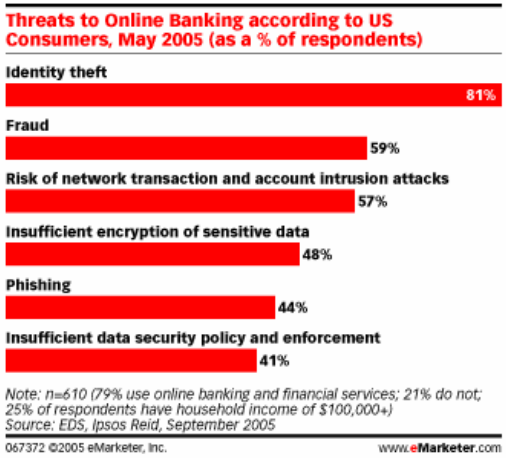
1. Software Security Awareness: Business Cases
2. Approaches To Application Security
3. Software Security Roadmap (CMM, S-SDLCs)
4. Software Security Activities And Artifacts
   - Security Requirements and Abuse Cases
   - Threat Modeling
   - Secure Design Guidelines
   - Secure Coding Standards and Reviews
   - Security Tests
   - Secure Configuration and Deployment
5. Metrics and Measurements
6. Business Cases

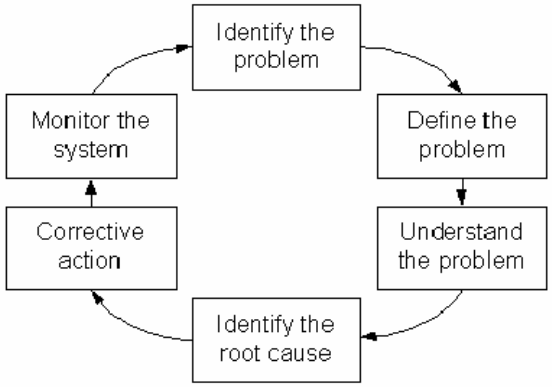
Software Security Awareness Business Cases

Initial Business Cases For Software Security

Avoid Mis-Information: Fear Uncertainty Doubt (FUD)

Use Business Cases: Costs, Threat Reports, Root Causes





Business Case #1: Costs

- "***Removing only 50 percent of software vulnerabilities before use will reduce defect management and incident response costs by 75 percent.***" (Gartner)
- ***The average cost of a security bulletin is 100,000 $*** (M. Howard and D. LeBlanc in Writing Secure Software book)
- ***It is 100 Times More Expensive to Fix Security Bug at Production Than Design"*** (IBM Systems Sciences Institute)

Business Case #2: Threats To Applications

- **93.8% of all phishing attacks in 2007 are targeting financial institutions** (Anti-Phishing Group)
- **Phishing attacks soar in 2007** (Gartner)
  - 3.6 Million victims, $ 3.2 Billion Loss (2007)
  - 2.3 Million victims, $ 0.5 Billion Loss (2006)
- **Phishing attacks exploit web application vulnerabilities (OWASP T10)**
  - A1(XSS) weak authentication authorization vulnerabilities (A1, A4, A7, A10)

Business Case #3: Security Root Causes

- **Application vulnerabilities issues:**
  - **92 % of reported vulnerabilities are in applications not in networks (NIST)**

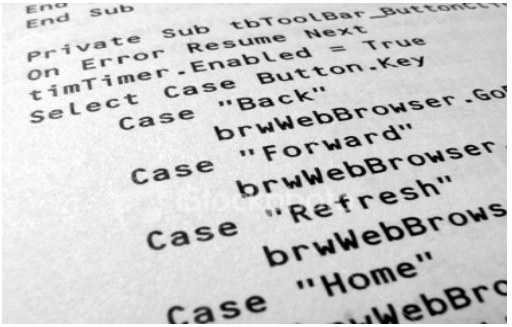
- **Security design flaws issues:**
  - **"Security design flaws account 70% of the defects being analyzed**
  - **47% of design flaws have medium and high business impact and easily exploitable"** (@Stake)

Approaches To Application Security

The “Know But Ignore” Approach: Do Not Act

The Reactive Approach: Act After The Fact

5



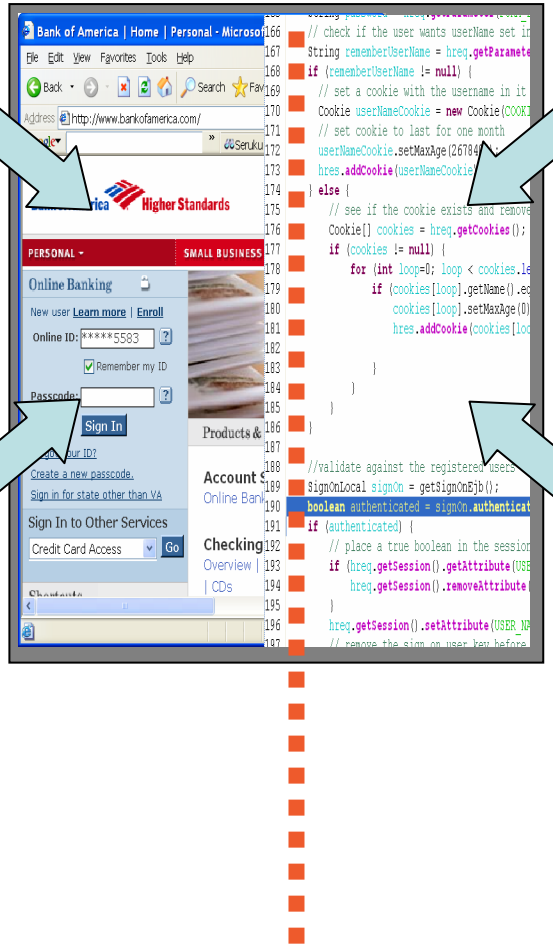
Go Fix Security Bugs!

?

Tactical View: Finding Vulnerabilities

Manual Penetration Testing

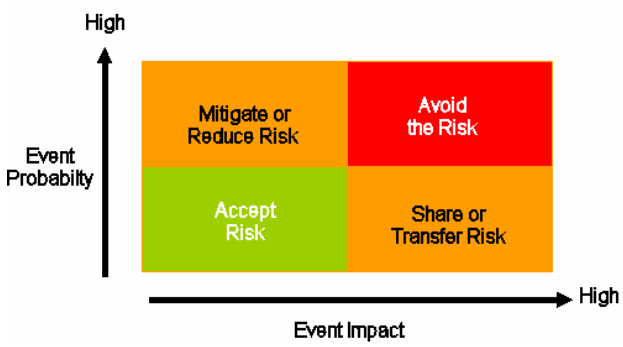
Manual Code Review



Automated Vulnerability Scanning

Automated Static Code Analysis

Strategic View: Manage Software Risks

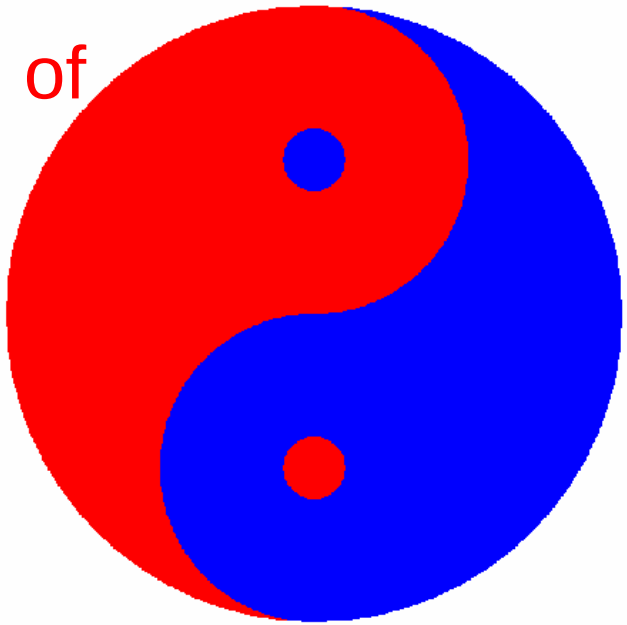


SDLC Phases	Requirements		Design	Development	Testing	Deployment and Operations	
Secure Software Best Practices	Preliminary Software Risk Analysis	Security Requirements Engineering	Security Risk-Driven Design	Secure Code Implementation	Security Tests	Security Configuration & Deployment	Secure Operations
Ongoing S-SDLC Activities Metrics and Measurements, Training, and Awareness							
S-SDLC Activities	Define Use & Misuse Cases	Define Security Requirements	Secure Architecture & Design Patterns Threat Modeling Security Test Planning Security Architecture Review	Peer Code Review Automated Static and Dynamic Code Review Security Unit Tests	Functional Test Risk Driven Tests Systems Tests White Box Testing Black Box Testing	Secure Configuration Secure Deployment	
Other Disciplines	High-Level Risk Assessments		Technical Risk Assessment				Incident Management Patch Management

Holistic View: Software vs. Application Security

Security built into each phase of the SDLC

Security applied by catch and patches



Look at root problem causes

Look at external symptoms

Proactive, Threat Modeling, Secure Code Reviews

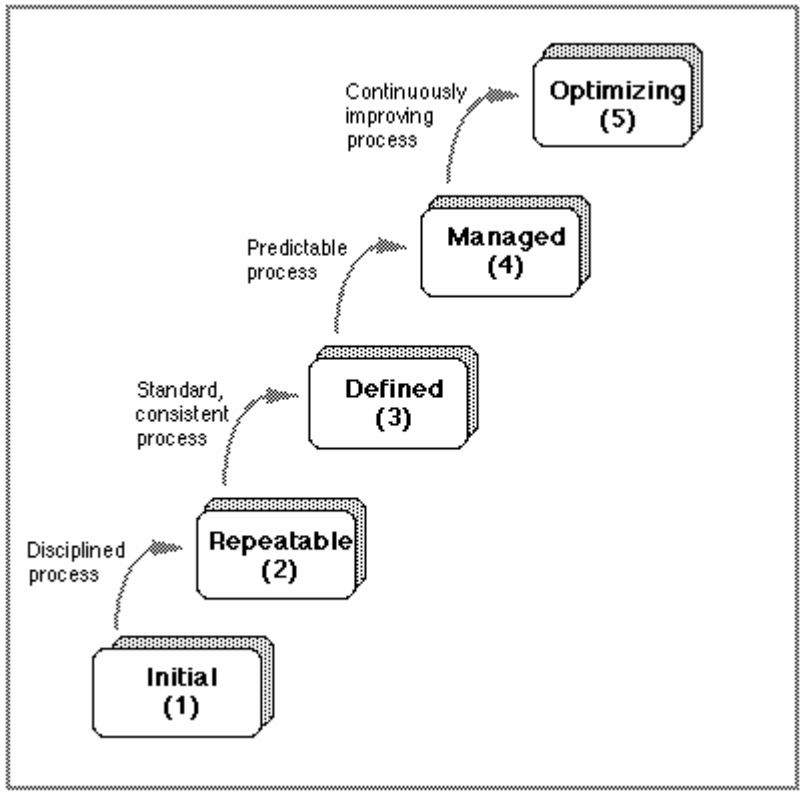
Reactive, Incident Response, Compliance

Software Security Roadmap

Software Security Roadmap

1. **Assess the maturity** of the organization software security development processes, people and tools
2. **Define the software security process:** security enhanced SDLCs, frameworks and checkpoints
3. **Implement software security activities**
   1. Security Requirements
   2. Secure Design and Threat Modeling
   3. Secure Coding Guidelines and Security Code Review
   4. Security Testing
   5. Secure Deployment
4. **Collect metrics and measurements**
5. **Create business cases and set objectives**

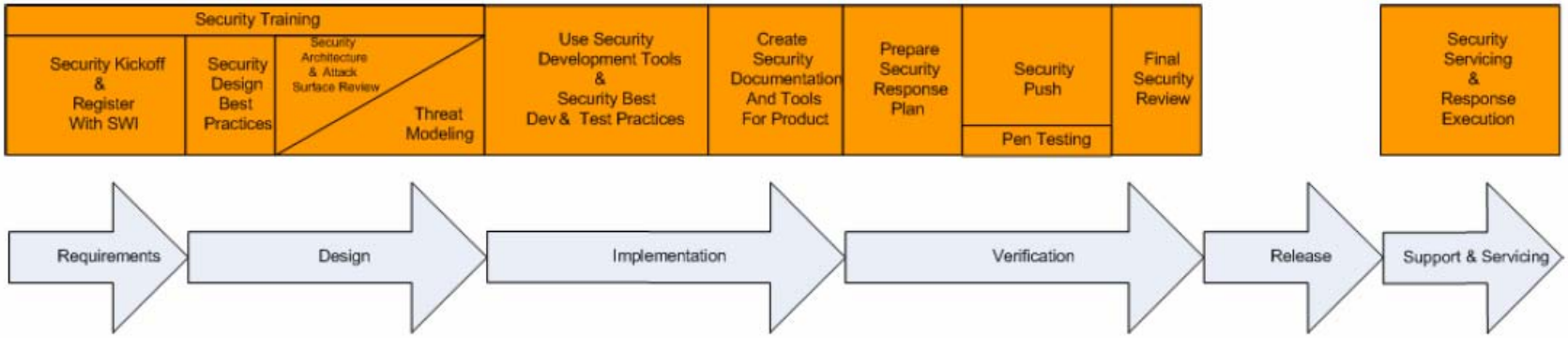
Software Security Initiative: Maturity Levels

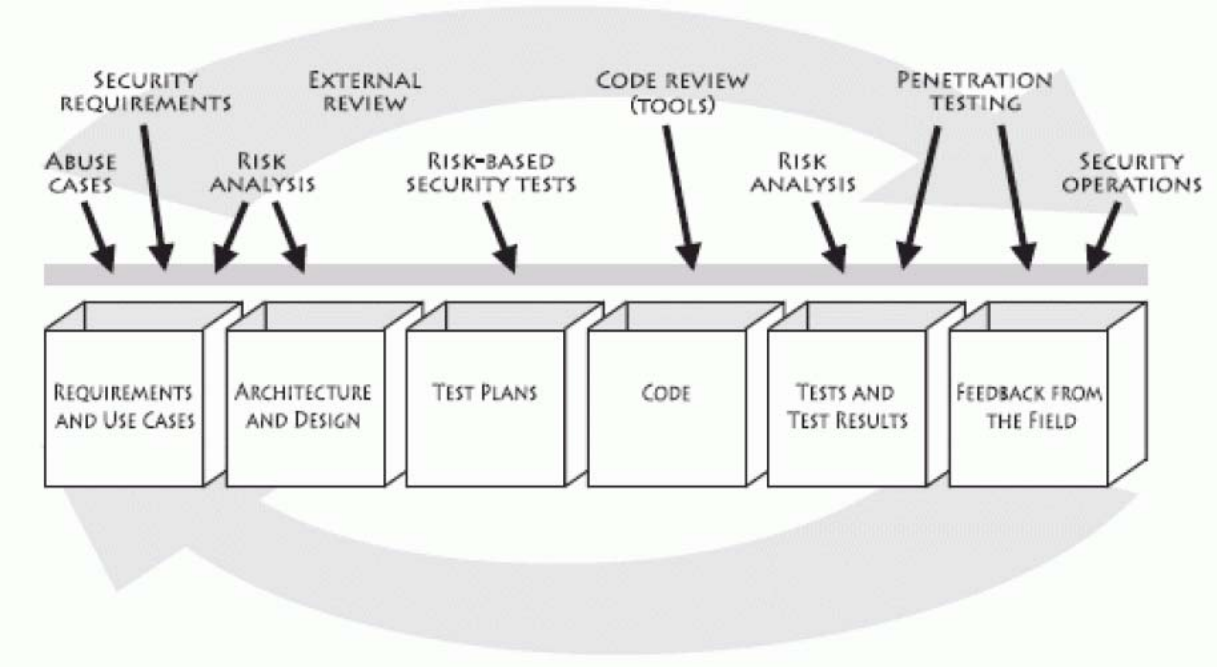


Software Security Initiative: Maturity Levels

- **Maturity Innocence (CMM 0-1)**
  - No formal security requirements
  - Issues addressed with penetration testing and incidents
  - Penetrate and patch and reactive approach
- **Maturity Awareness (CMM 2-3)**
  - All applications have penetration tests done before going into production
  - Secure coding standards are adopted as well as source code reviews
- **Maturity Enlightenment (CCM 4-5)**
  - Threat analysis in each phase of the SDLC
  - Risk metrics and vulnerability measurements are used for security activity decision making

Security-enhanced lifecycle process (S-SDLC) models: MS-SDL, Cigital TP and CLASP



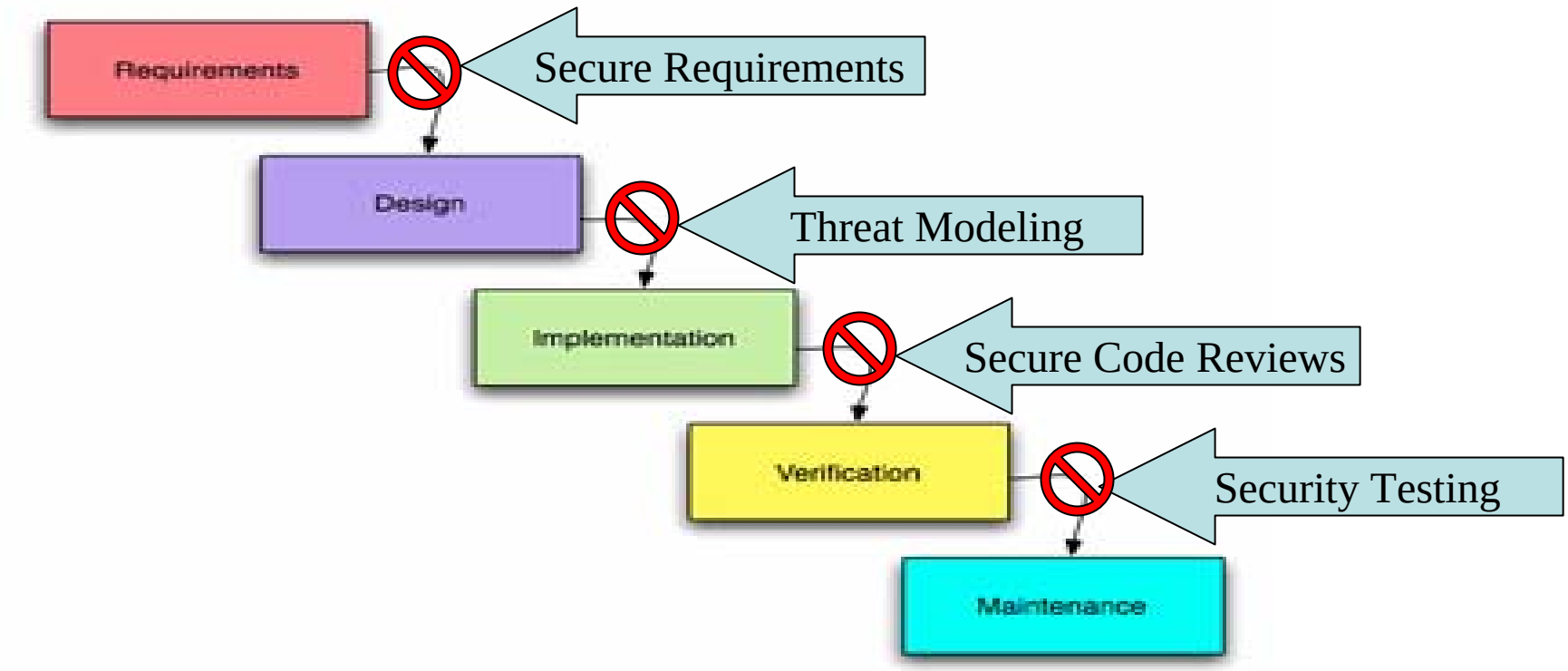


CLASP

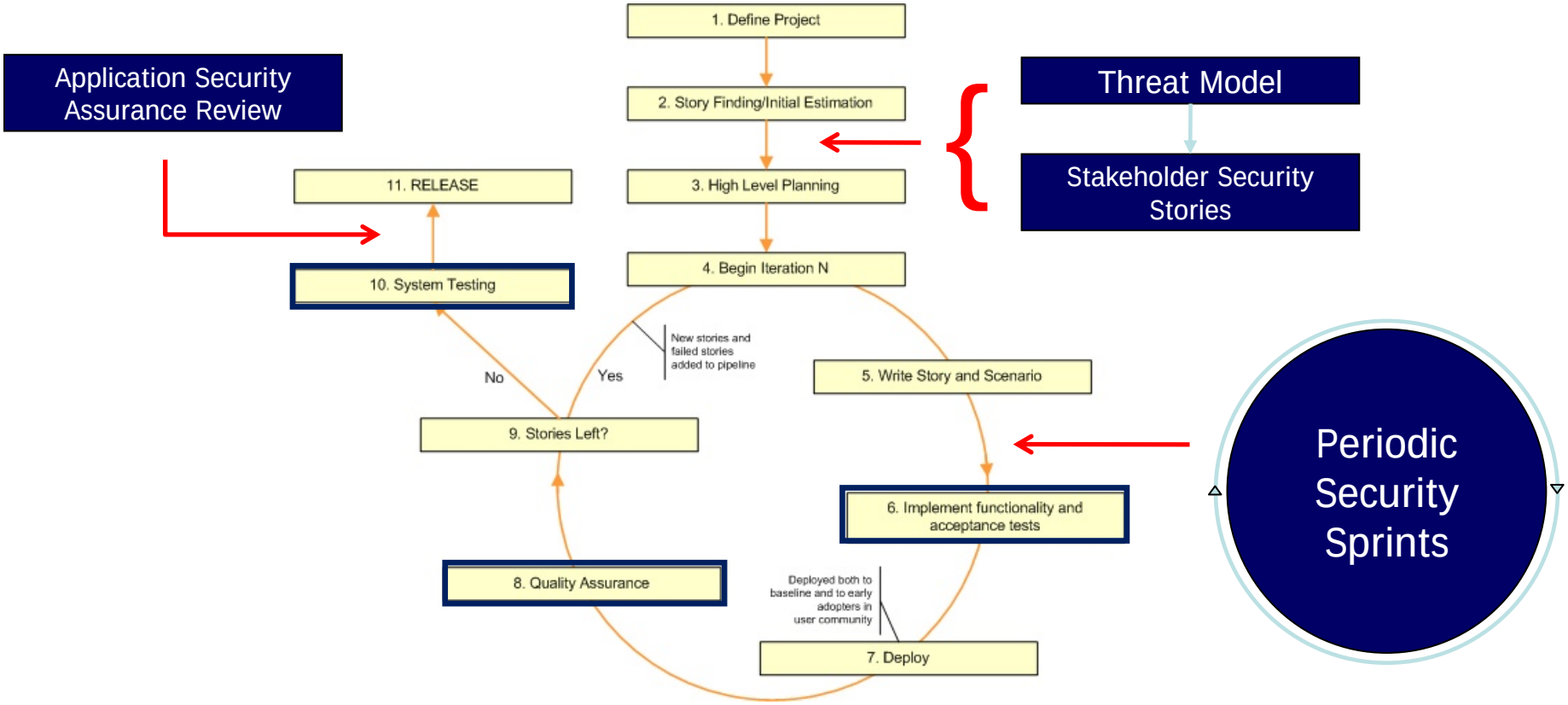
BEST PRACTICES

1) Institute awareness programs
2) Perform application assessments
3) Capture security requirements
4) Implement secure development practices
5) Build vulnerability remediation procedures
6) Define & monitor metrics
7) Publish operational security guidelines

Building Security In the Waterfall SDLC



Build Security in Agile SDLC



Security Requirements and Abuse Cases

Security Requirements

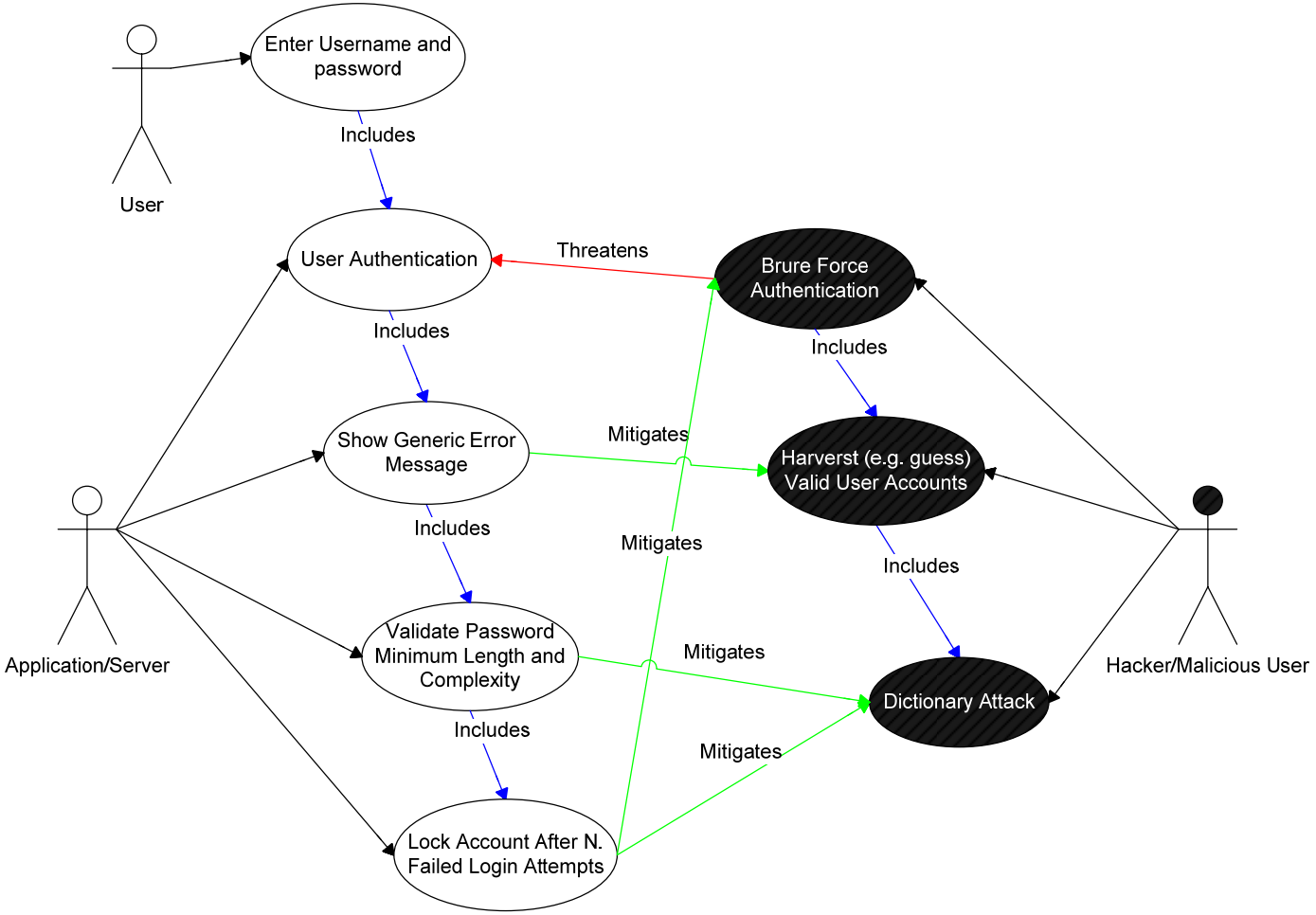
- Encompasses both functional requirements for security controls and risk mitigation driven requirements from the abuse case scenarios
- Define Security Requirements in Standards
  - Which controls are required (e.g. authentication, authorization, encryption etc)
  - Where should be implemented (e.g. design, source code, application, server)
  - Why are required
    - Compliance and auditing (e.g. FFIEC, PCI, SOX etc.)
    - Mitigation for known threats (e.g. STRIDE)
  - How should be implemented and tested

Functional and Non Functional Requirements

- Functional Requirements:
  - Define the expected functionality of security controls
  - Depends on the applicable standards, policies and regulations
  - Positive statements
    - “the application will lockout the user after 6 failed logon attempts”
    - “passwords need to be 6 min characters, alphanumeric”

- Risk Driven Security Requirements
  - Address all common vulnerabilities
  - Can be derived by use (or misuse) cases
  - Negative statements
    - The application should not allow of the data being altered or destroyed
    - The application should not be compromised or misused for un-authorized financial transaction by a malicious user.

Security Requirements Derivation Through Use and Misuse Cases



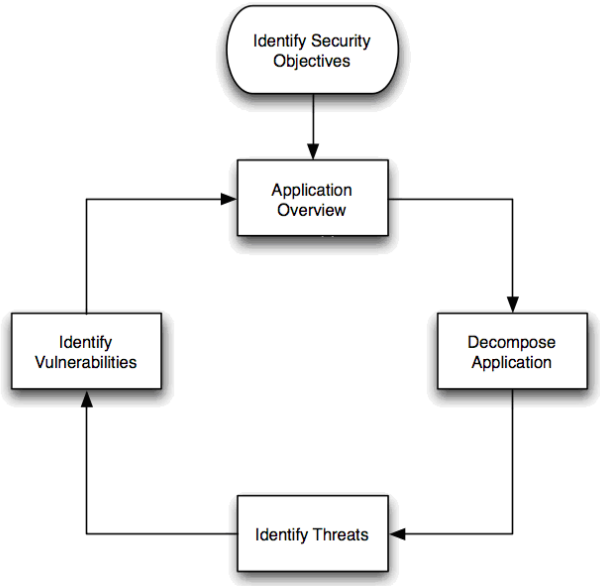
Source: OWASP Testing Guide Vs 3 Introduction

Threat Modeling

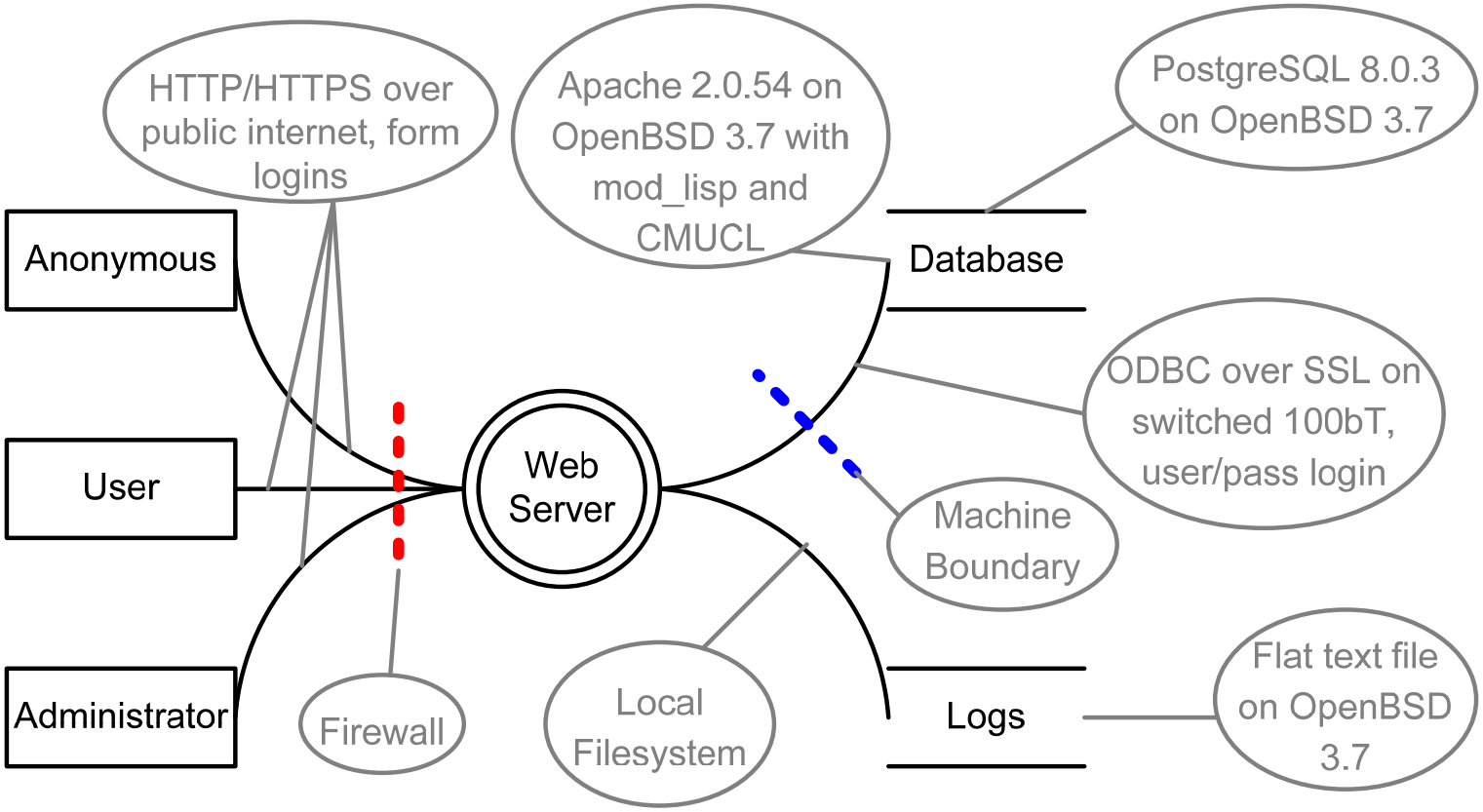
Threat Modeling

- **Categorizes the threats to the application. highlights potential vulnerabilities and identifies countermeasures to be developed**
- **Use a systematic fact based and methodical approach:**

1. **Scope Assessment**
   Requirements, Use Cases
2. **System Modeling**
   Physical and Logical View, Data Flows, Trust Boundaries, Entry Points
3. **Threat Identification**
   STRIDE, ASF
4. **Threat Vulnerabilities and Attacks**
   Checklists, Attack Vulnerability Mapping
5. **Identification of Countermeasures**
   Security Controls
6. **Threat Prioritization and Risk Ranking**
   Risk Modeling

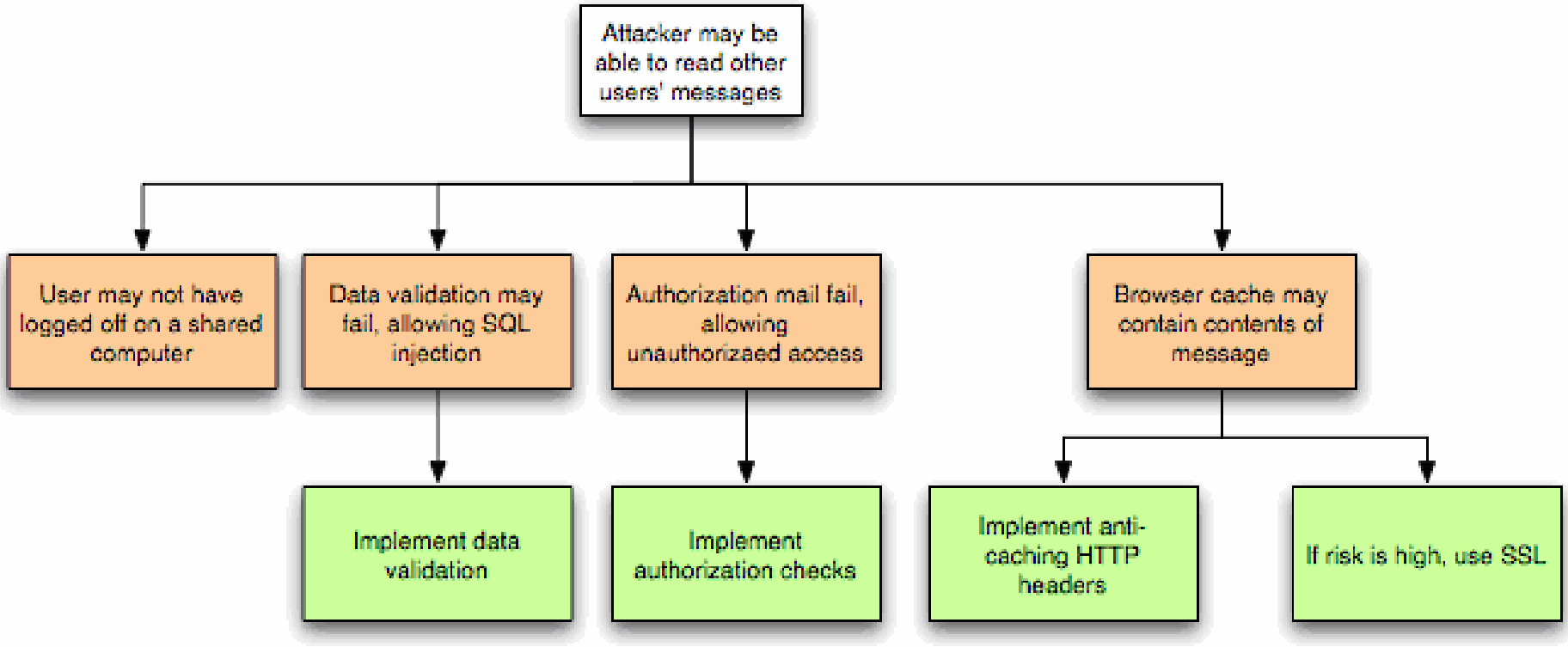


System Modeling: Data Flow Diagrams



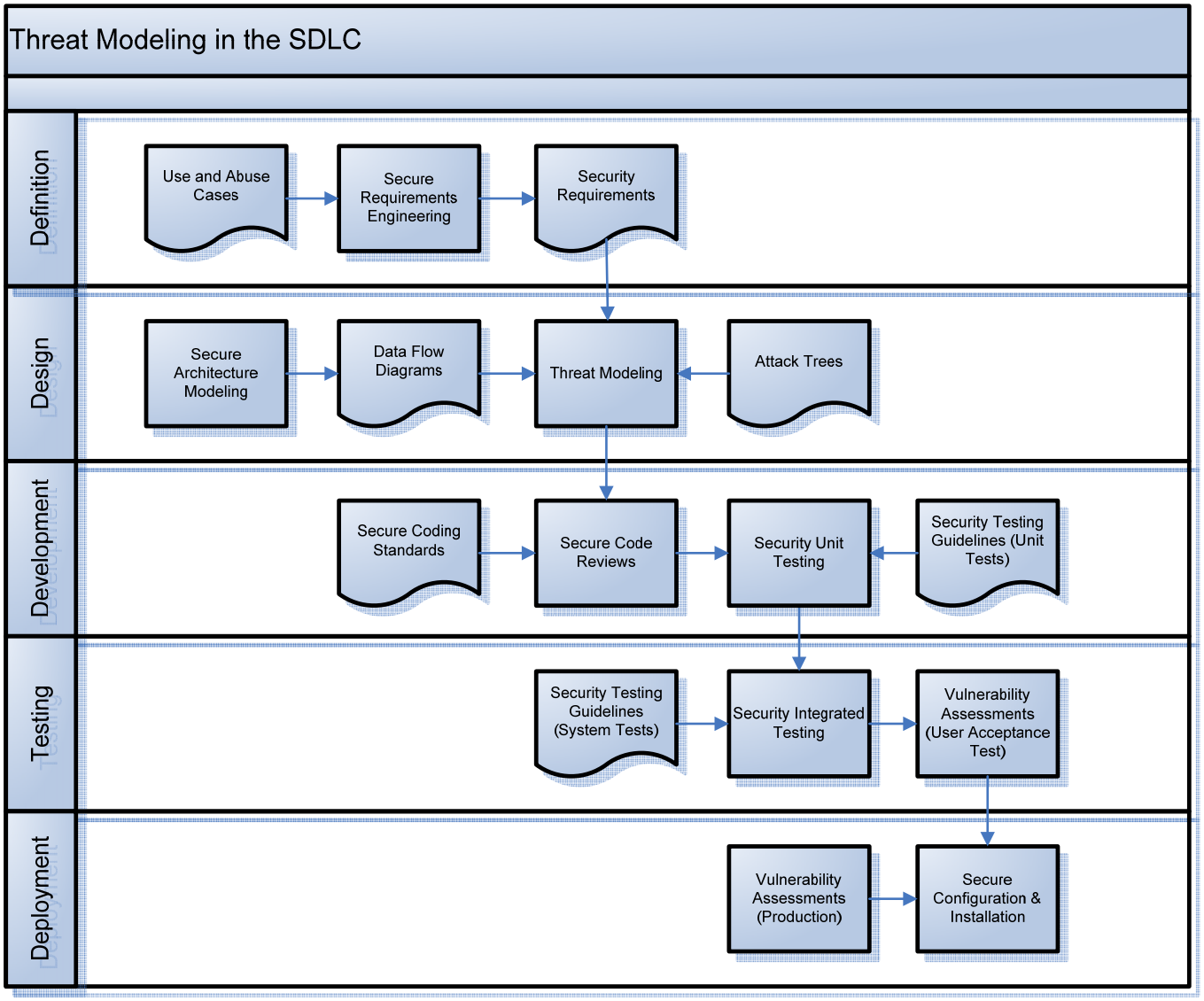
Source: Threat Modeling Dr James Walden

Threats and Countermeasures Identification



Source: OWASP TM

Threat Modeling in the SDLC



Secure Design Guidelines

Security Design Reviews

- Objective is promote secure design and identify of potential flaws before construction phase
- Secure Architecture Review Process
  - Review High Level Design documents and verify that security controls requirements are documented
  - Engage with:
    - Architects
    - Application Security Experts
  - Provide guidance on security technology/design patterns
  - Identify potential gaps in security controls with threat modeling

Secure Coding Standards and Secure Code Reviews

Secure Coding Standards/Guidelines

Describe secure coding requirement in terms of:

1. **The common vulnerabilities** (e.g. OWASP T10)
2. **The issue type** (e.g. Application Security Frame)
3. **The security threat or how can be exploited**
4. **The in-secure code root cause** of the vulnerability
5. **The "How to" find the vulnerability** with black box and white box testing
6. **The secure coding requirement/recommendation**
7. **The risk rating** (e.g. STRIDE/DREAD, OWASP)

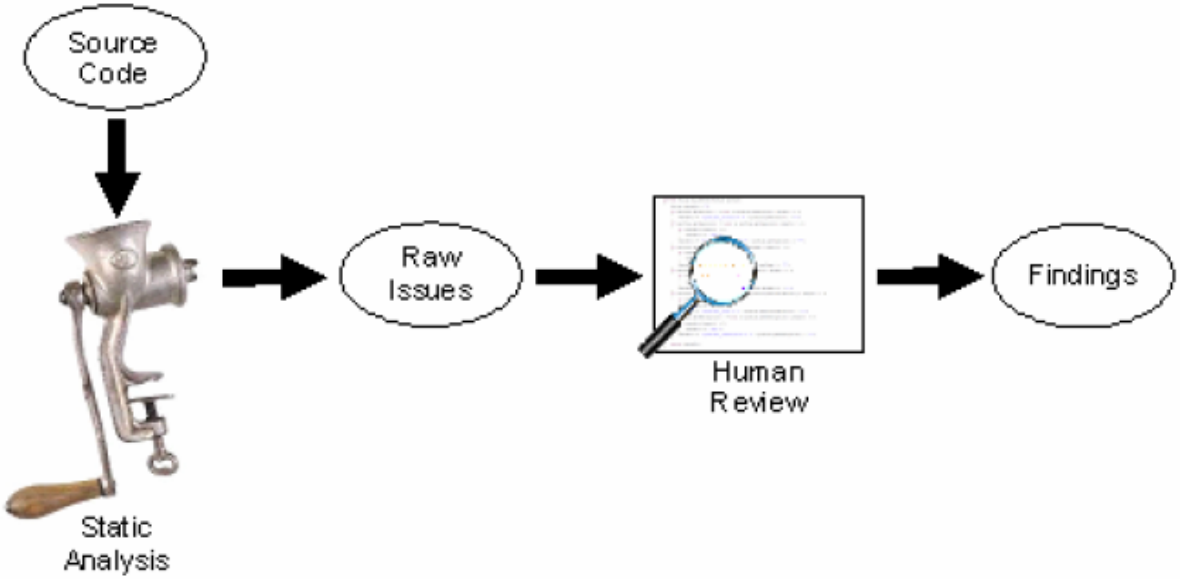
Example SQL Injection - Secure coding requirements

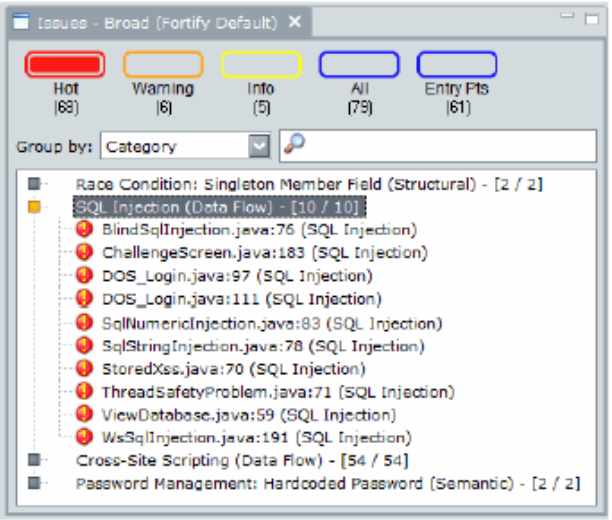
- **Use SQL parameterized queries, avoid dynamic SQL generation**:
  - *SELECT * FROM users WHERE username=?*
  - *JAVA EE use strongly typed "PreparedStatement"*
  - *in .NET use "SqlCommand" with "SqlParameters"*
- **Sanitize input**, **remove special characters**:
  ' " ` ; * % _ =&\|*?~<>^()[]{}$\n\r
- **Use custom error messages:**
  - No SQL exception information in error messages

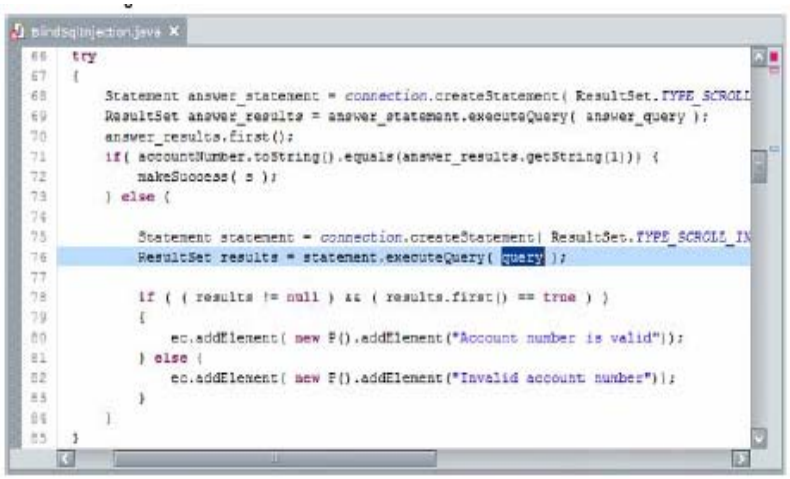
Secure Code Reviews

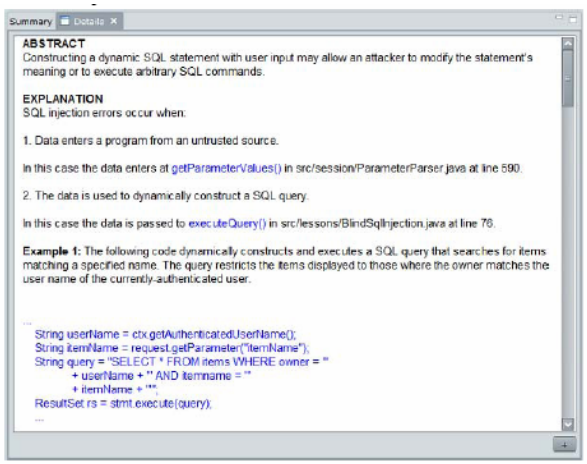
- **Objectives:**
  - Identification of security issues in source code, the type of the issue, the severity and recommendation on how should be fixed
  - **Can be used to *validate secure coding standards***
  - **Security assessment before releasing to QA and production**
- **Methodologies**:
  - **Automated Focused Source Code Analysis**
    - Focus is on validation of false positives and auditing of automated scan results
  - **Manually Focused Secure Code Review**
    - Focus is on identification of security issues as bugs vs. flaws by categorizing the issues by type of vulnerability introduced

Source Code Analysis









Security Tests

Security Testing

- **Develop security test cases**
  - Positive functional security test cases
  - Negative test cases (from use and misuse cases)
  - Common vulnerabilities exposure
- **Integrate Tests in Developers and Testers Workflows**
  - Static and Dynamic Testing, Unit Tests
  - Integrated System Tests and Operational Tests
- **Analyze and report test data**
  - Defect Management
  - Root Cause Identification

Security Testing Example: XSS

- Define Test Case
  - Test login web page for XSS
- Testing Procedure
  - Type the following strings in input fields
    - **<script>alert()</script>;**
    - **javascript:alert()**
    - **+ADw-SCRIPT+AD4-alert();+**
  - Pass: Input validation Error is through to the user
  - Fail: an alert dialog box is seen in the browser window.

Testing Tools Categorization and Examples

- Vulnerability Scanning
  - ISS, Foundscan, Nessus, Nikto
- Fault Injection Testing (Black Box Testing)
  - Webinspect, Appscan, Hailstorm, Paros, Peach
- Binary Analysis
  - IDA Pro, @stake SmartRisk
- Source Code Analysis
  - Fortify, Klockworks, Parasoft, Free Tools (e.g. FindBugs)
- Threat Modeling
  - MS TAM, TRIKE, PTA Technologies
- Rootkit BackDoor Analysis
  - ootkits.org and rootkit.nl

Secure Configuration And Deployment

Secure Deployment And Configuration Items

- Ensure the server configuration is secure
  - Only essential services, server hardening policies
- Protect Access To Application Files/Data
  - XML files, property files, scripts, databases, directories
- Enable Auditing And Logging
  - Enable all secure auditing and logging, protect logs
- Enforce Change Management Controls
  - Don’t allow configuration changes without oversight
  - Audit configuration data
- Release Securely
  - Don’t allow releases to ship without a security review

Metrics And Measurements

Application Security Defect Tracking and Metrics

- Define where and how security metrics is collected
- Tracking security defects throughout the SDLC
  - Report the root causes: requirements, design, code, application
  - Report the type of the issues, the severity and whether has been fixed or no
- Metrics
  - What lifecycle stage are most flaws originating in?
  - What security mechanisms/controls are we having trouble implementing?
  - What security vulnerabilities are we having trouble fixing?

Examples of Application Security Metrics

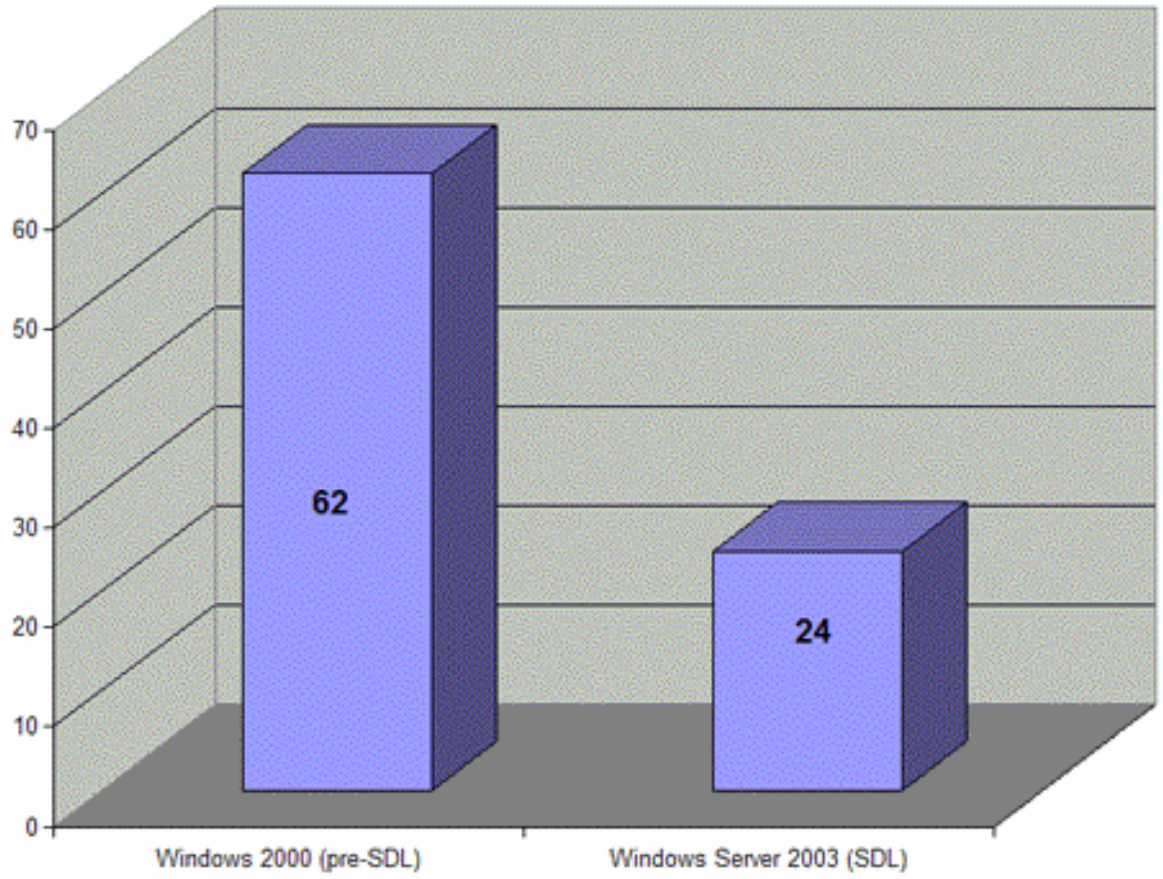
Process Metrics

- Is a SDL is used are security gates enforced?
- Is code validated against security standards?
- Security posture of a new application before delivery
  - Security Officers Sign Off?
  - Test For Security Vulnerabilities Executed?
  - All high risk issues closed?
  - Risk assessments completed?
- % of developers trained, using organizational security best practice technology, architecture and processes

Management Metrics

- % of applications rated “business-critical” that have been security tested
- % of projects that where developed with the SDL
- % of security issues identified by lifecycle phase
- % of issues whose risk has been accepted
- % of security issues being fixed
- Average time to correct vulnerabilities
- Business impact of critical security incidents.

Examples of Security Metrics: Trailing MS Bulletins



Security Metrics Goals The Good and The Bad

- **Good:** if goals when are “SMART” that is Specific, Measurable, Attainable, Realistic, Traceable and Appropriate
  - Example: reducing the overall number of vulnerabilities by 30% by fixing all low hanging fruits with source code analysis during construction
- **Bad**: if the goals justify the means to obtain the goals

Business Case

Business Cases for Your Organization

- Tie the metrics to the business cases and support the project stakeholders agendas:
  - **Developer Leads**: show that software can be build more securely and efficiently
  - **Project Managers**: shows that projects are on schedule and moving on target for the delivery dates and are getting better during tests
  - **Information Security Officers:** provides a level of assurance that security standard compliance is addressed through the security review processes within the organization.
- Benefits:
  - Cost savings
  - Risk measurement and reduction
  - Compliance reporting

Business Cases And Software Security Strategy

- Be realistic on what can be achieved
  - Organization is **not yet ready** (e.g. mature)
  - Engineers are **not trained** in software security
  - There are **no tools available**
- Adapt the strategy to reality
  - **Build upon your company strenghts**
  - **Get stakeholders buy in** (CIOs, ISOs, PM, Developers, Architects)
  - **Set achieveable goals:** reduce 30% of vulnerabilities found through ethical hacking via source code analysys
- Perform a gap analysis and proceed with process improvement cycles:
  - Tailor to the initiative to the company culture
  - Be risk management driven
  - Introduce metrics and prove results