

Practical Guide to Securing the SDLC

Branko Ninkovic

Dragonfly Technologies

Founder



Agenda

- Understanding the Threats
- Software versus Security Goals
- Secure Coding and Testing
- A Proactive Approach to Secure Coding
- Secure Coding Frameworks
- Case Study

UNDERSTANDING THE THREATS

Industry Surveys and Research

- Web Application Vulnerabilities are a real threat
- IBM's X-Force security research team, **half of all vulnerabilities identified had something to do with Web applications Software Development**
- 2009 Gartner survey, "75% of security breaches happen at the application"
- Gartner research, organisation only spend 10% of their security budget on web applications

Web Application Vulnerabilities

Vulnerability Disclosures Affecting Web Applications

Cumulative, year over year

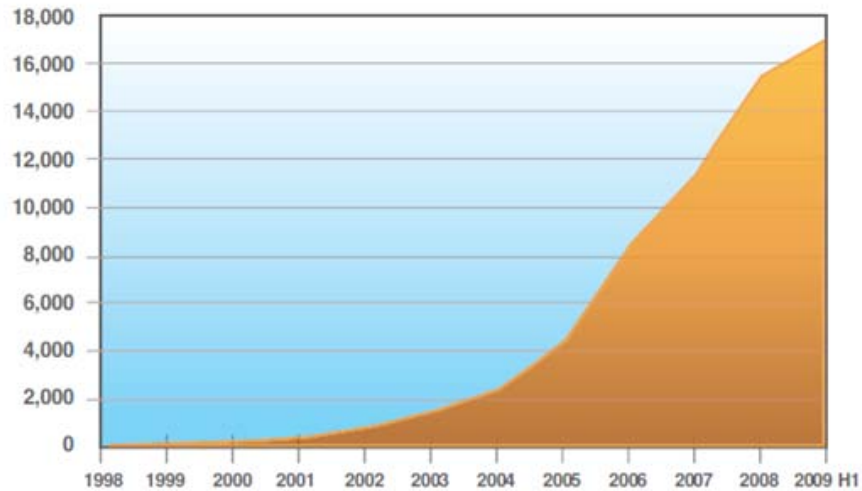


Figure 7: Cumulative Count of Web Application Vulnerability Disclosures, 1998 - 2009 H1

Web Application Vulnerabilities

as a Percentage of all Disclosures in 2009 H1

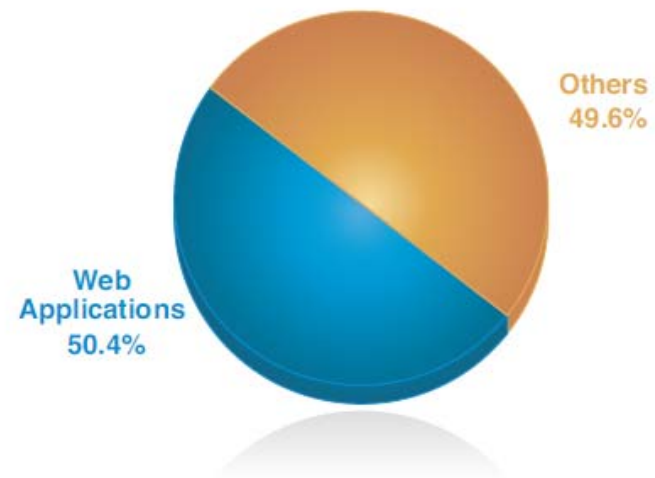
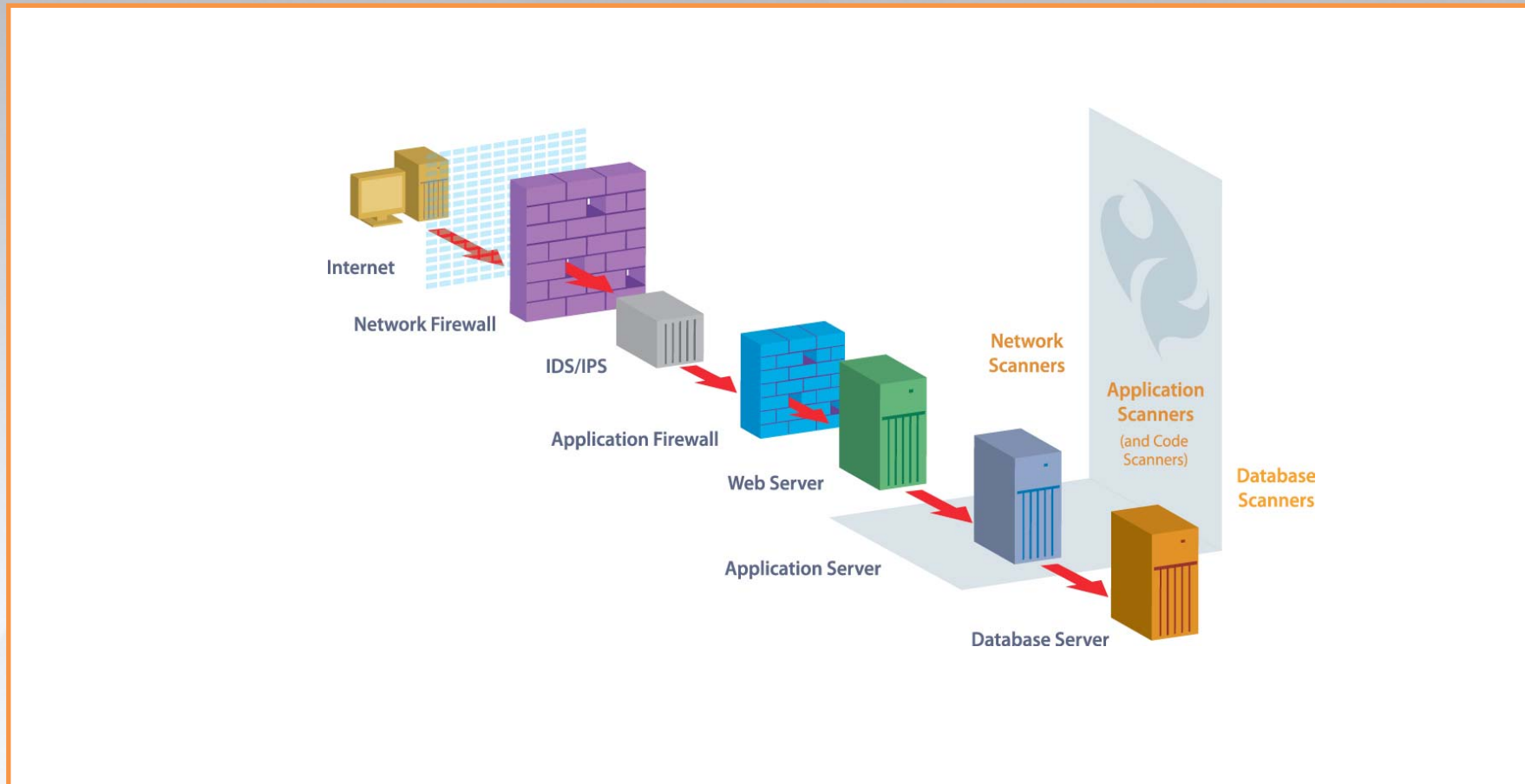


Figure 8: Percentage of Vulnerability Disclosures that Affect Web Applications, 2009 H1

(IBM) Application security incidents and lost data on the rise

Traditional Firewalls / IDS / IPS



Source: IBM

Cause and Effect

- **Developers not trained in Security**
 - Focus on features
- **Under investment from security teams**
 - Lack of tools, polices process ...
- **Growth in complex, mission critical online applications**
 - Online banking, e-commerce Web 2.0, ...
- **Number one focus by hackers**
 - 75% of attacks focused on applications (Gartner)

SOFTWARE VERSUS SECURITY GOALS



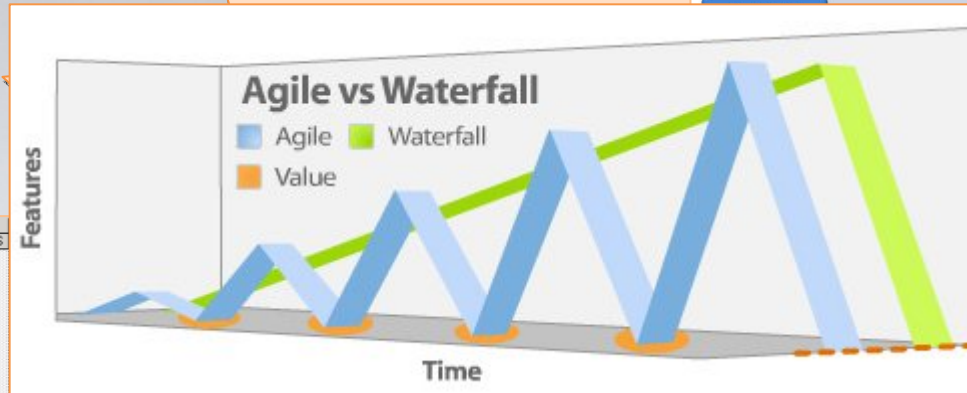
A Business Perspective

- **Software**
 - Increase productivity, **reduce** costs and **increase** or generate new revenue
 - Delivered on **time** and on **budget**
- **Security**
 - Systems are resilient to attack, **confidentiality**, **integrity**, and **availability** of the system and its data are protected
 - Ability to **control** data

A Developers Perspective

T10 OWASP Top 10 Application Security Risks – 2010

- A1 – Injection**
 - Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.
- A2 – Cross-Site Scripting (XSS)**
 - XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
- A3 – Broken Authentication and Session Management**
 - Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.



When a developer exposes a reference to an internal file, directory, or database key. Without an access control check, an attacker can manipulate these references to access unauthorized data.

When a developer allows a victim's browser to send a forged HTTP request, including the other automatically included authentication information, to a server. This allows the attacker to force the victim's browser to generate requests that the server thinks are legitimate requests from the victim.

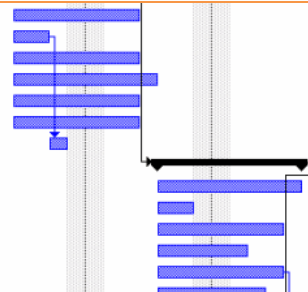
When a developer does not use secure configuration defined and deployed for the application, web server, database server, and platform. All these settings and configurations are often not shipped with secure defaults, and are often not updated to date, including all code libraries used by the application.

When a developer does not properly protect sensitive data, such as credit cards, SSNs, and other personally identifiable information. Attackers may steal or modify this data, or conduct identity theft, credit card fraud, or other crimes.

When a developer does not perform access control checks before rendering protected links and buttons. Attackers can perform similar access control checks each time these pages are accessed to bypass the access controls.

When a developer does not authenticate, encrypt, and protect the confidentiality and integrity of sensitive data. If they do, they sometimes support weak algorithms, use expired keys, or do not use them correctly.

Task Name	F	S
1 PLANNING STAGE		
2 Quality Assurance Plan		
3 In-Stage Assessment		
4 Project Plan		
5 Stage Exit		
6 REQUIREMENTS DEFINITION STAGE		
7 Requirements Specification (draft)		
8 Capacity Planning and LAN SLA		
9 In-Stage Assessment		
10 Continuity of Operations Plan		
11 Project Test Plan		
12 Acceptance Test Plan (draft)		
13 Requirements Specification (final)		
14 Stage Exit		
15 DESIGN STAGE		
16 Design Specification (draft)		
17 Computer Security and Privacy Plan		
18 Integration Test Plan		
19 Training Plan (draft)		
20 In-Stage Assessment		
21 Configuration Management Plan		



- A10 – Unvalidated Redirects and Forwards**
 - Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

SECURE CODING AND TESTING



What is Secure Coding?

- Developers with an awareness of security best practice in their trained discipline
- Developers who are able to assist and provide guidance and understand secure configurations
- A development team that follow a Development Guide such as OWASP Secure Coding Principles

OWASP Secure Coding Principles

- Minimize attack surface area
- Establish secure defaults
- Principle of Least privilege
- Principle of Defense in depth
- Fail securely
- Don't trust services
- Separation of duties
- Avoid security by obscurity
- Keep security simple
- Fix security issues correctly

An Practical Example – Fail Securely

```
...
isAdmin = true;
try
{
    codeWhichMayFail();
    isAdmin = isUserInRole("Administrator" );
} catch (Exception ex) {
    log.write(ex.toString());
}
...
```

What is Security Testing?

- Security testing uses dynamic techniques to simulate an attacker
- Follows similar processes to software testing
- “Negative Testing” – requires a mind shift
- Use Threat Modelling rankings as the basis of prioritising testing
- Uses manual and automated testing techniques
- OWASP

A PROACTIVE APPROACH TO SECURE CODING

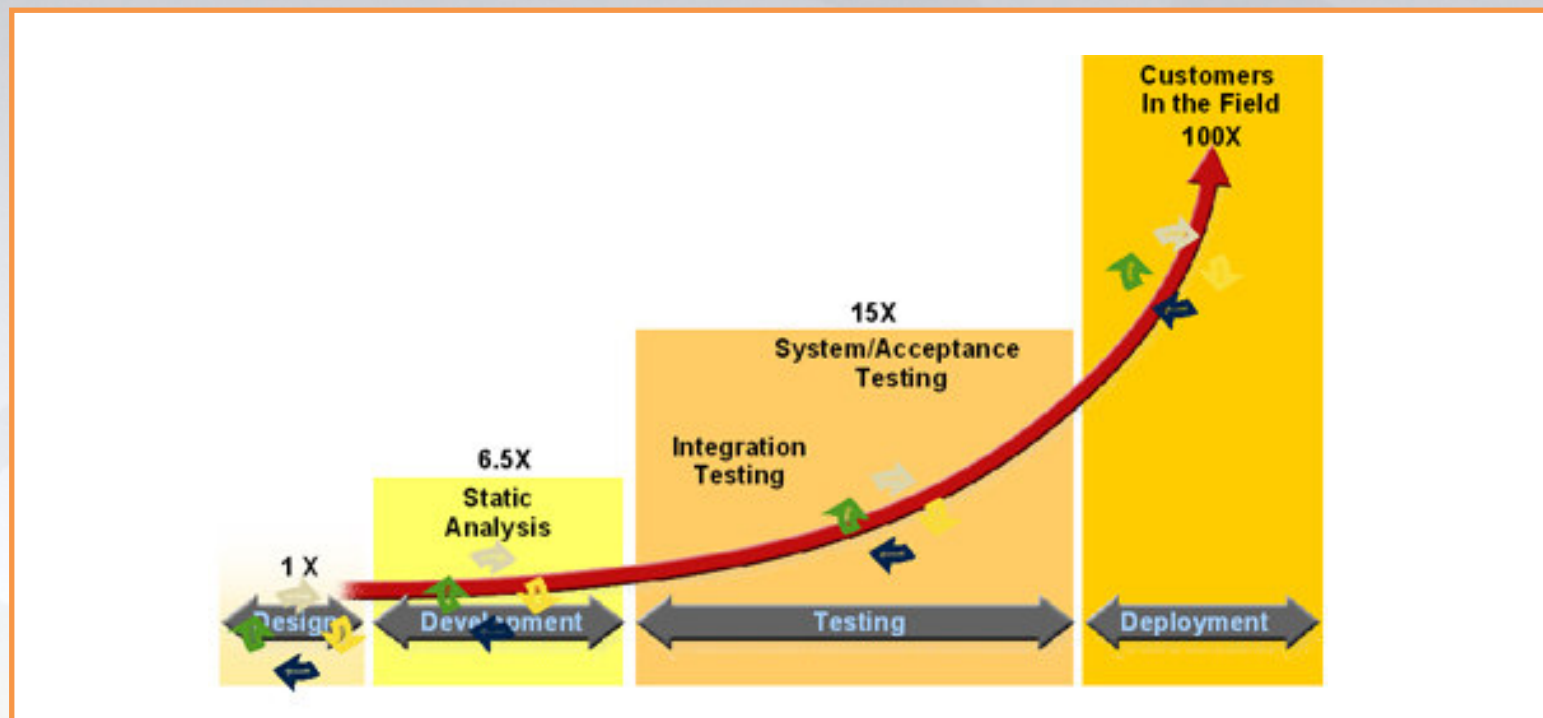


A Defect?

- A Web application vulnerability is a defect
- Each industry there are processes in place to minimise defects
- How do we adopt a secure application development framework?
- Developers need to take cue from these industries

Defect Management

- Reduce development costs by **75%** (Gartner) when these defects are identified early on in the Design and Development phases versus UAT



Source: IBM Systems Sciences Institute

SECURE CODING FRAMEWORKS



Secure Coding Frameworks

- OWASP – Open Web Application Security Project www.owasp.org
- IBM have the Secure Engineering Framework (SEF)

*The Secure Engineering Framework (SEF) is intended to help ensure that **software is secure by design, secure in implementation, and secure in deployment**. The global nature of software development activities today necessitates the application of secure engineering principles across **global development teams regardless of their physical location**.*

Secure Coding Goals

- Identify stakeholders and level of security
Education and
- Adopt processes and tools to **Reduce Security Defects**
- Test for security defects **early**
- Adopt a policy of internal security testing, **often**, and external third-party testing at the end of each major release

Framework Components

- Risk Assessment and Threat Modelling
- Security Requirements
- Secure Coding
- Test and Vulnerability Assessment

Risk Assessment and Threat Modelling

- A measure of business value of information and processing
- Used to estimate the level of effort required to achieve the desired security characteristics

Identify the assets

Identify the potential threats

Assign an impact for each threat

Determine the **probability** of compromise

Rank the risks

Define **mitigating** counter-measures as needed

Security Requirements

- The **objective** of security requirements is to help ensure that the web application can defend itself from attack

Auditing and logging
Authentication and authorization
Session management
Input validation and output encoding
Exception management
Cryptography and integrity
Data at rest
Data in motion
Configuration management

Test and Vulnerability Assessment

- A security test plan is a critical part of test and vulnerability assessment.
- Automated analysis of source code, object code binaries, dynamic analysis and runtime analysis.

Source code security analysers
Bytecode security analysers
Binary security analysers
Dynamic analysis tools

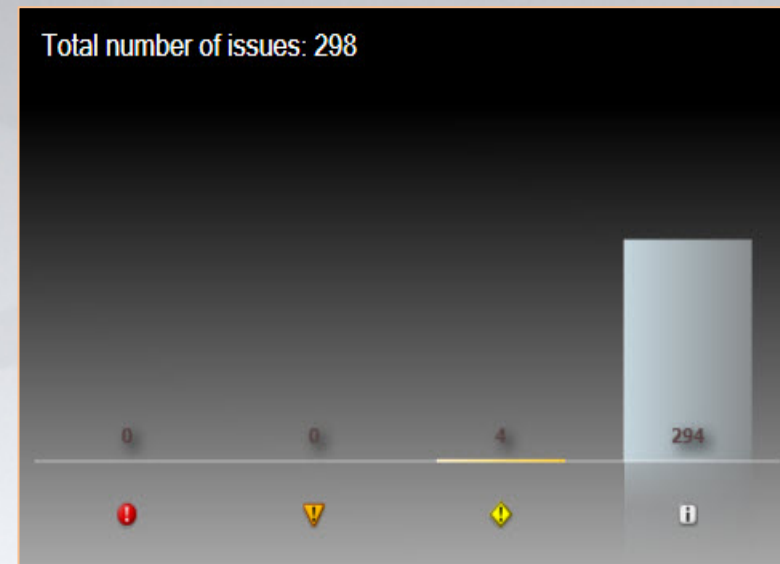
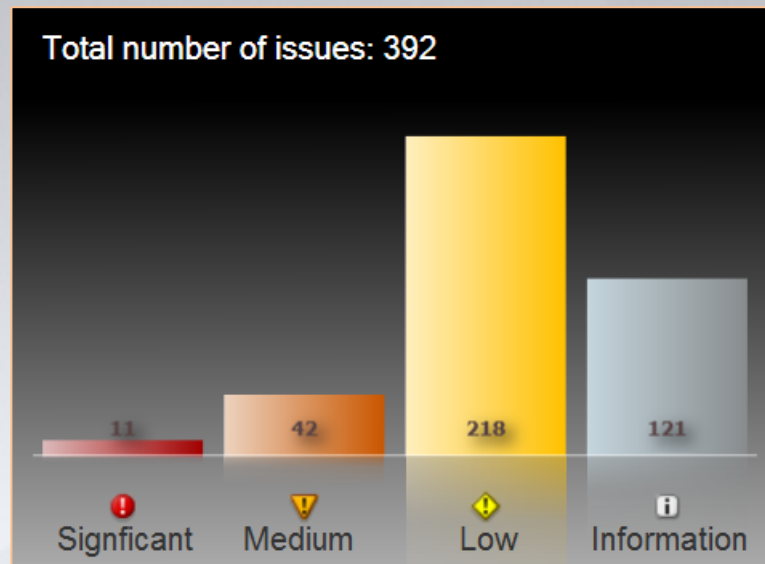
CASE STUDY – GLOBAL CONSULTING FIRM



Case Study - Background

- Sydney Head Office
- Agile Development .NET Application
- Deployed to 70 countries world wide
- Baseline Web Application Vulnerability Assessment (WAVA) using **Rational AppScan 7.9**
- Total of 4 x WAVA and manual Penetration Testing

Case Study – The Results



Case Study - Stakeholders

- Senior Management
- Development Manager
- Developers (Development Team)

Thank You and Questions

- Branko Ninkovic
 - Dragonfly Technologies – Founder
 - bninkovic@dragonflytechnologies.com.au

