Discovering the Value of Verifying Web Application Security Using IBM Rational AppScan

Greg Pedley – IBM Rational Security
Lee Kinsman – IBM Rational Architect

An IBM Proof of Technology
Agenda

● **Introductions & facilities**
  ● Web App Security Landscape
  ● Vulnerability Analysis
    ‣ Top Attacks Overview
    ‣ Hands on Lab 1
  ● Vulnerability Analysis (continued)
    ‣ Hands on Lab 2
  ● Automated Vulnerability Analysis
    ‣ IBM® Rational® AppScan Overview
    ‣ Hands on Lab 3
Welcome to the Technical Exploration Center

- Introductions
- Access restrictions
- Restrooms
- Emergency Exits
- Smoking Policy
- Breakfast/Lunch/Snacks – location and times
- Special meal requirements?
POT Objectives

By the end of this session you will:

- Understand the web application environment
- Understand and differentiate between network and application level vulnerabilities
- Understand where the vulnerabilities exist
- Understand how to leverage AppScan to perform an automated scan for vulnerabilities
Introductions

- Please introduce yourself
- Name and organization
- Current integration technologies/tools in use

What do you want out of this Exploration session?
Agenda

- Introductions & facilities
- **Security Landscape**
  - Vulnerability Analysis
    - Top Attacks Overview
    - Cross Site Scripting
    - Hands on Lab 1
  - Vulnerability Analysis (continued)
    - Hands on Lab 2
  - Automated Vulnerability Analysis
    - AppScan Overview
    - Hands on Lab 3
Data thieves seeking out insiders

Karen Dearne | August 26, 2008

DATA thieves are switching their attention to softer targets such as Australia, as US and European companies harden their defences against losses of customer identity and credit card information.

Bryan Sartin, head of Verizon’s business investigative response team, says Australia is top of the list for organised criminals trawling for victims in countries with no data breach notification laws and a low rate of compliance with the payment card industry data security standard.

The biggest change in the past year had been the shift in data compromises as criminals sought targets that were lax about security, Sartin says.

"In 12 months, demand for our computer forensics services has shifted 180 degrees," he says.

"In April, there was more demand for our work in Australia than in the US and Canada combined."

Sartin attributes the change to the large monetary penalties non-compliant merchants face under the industry standard when breaches occur.

When the first industry deadline with demonstrable fines attached came into effect in the US last September, it was business as usual for the first couple of weeks, he says.

"Then, like clockwork, we saw the perpetrators realise these guys have really wised up.

"Then they started looking for other areas where these compliance standards are not in place or, at least, where there are no financial penalties."

Australia’s mentality has been that these things do not happen here. Now the nation is paying the price.
Reality: Security and Spending Are Unbalanced

75% of All Attacks on Information Security Are Directed to the Web Application Layer

Gartner
Confusing Network Security Discipline with Application Security

“Application developers and their superiors in IT departments too often mistakenly believe that firewalls, IDS / IPS, and network traffic encryption are sufficient measures for application security. By doing so they are confusing application security with network security”

“None of those technologies hardens application code. All those technologies deal with traffic to applications, not with the applications themselves…. Applications need protection through separate, specific security discipline – application security”

Application Security Testing, Gartner, March 2, 2007
Port 80 and Port 443 are open for business....
Cumulative Web Application Vulnerabilities

- SQL injection vulnerability disclosures more than doubled in comparison to 2007
- The number of active, automated attacks on web servers was unprecedented

Source: IBM X-Force®
What is the Root Cause?

1. Developers not trained in security/Focus on Functionality
   - Most computer science curricula have no security courses

2. Under investment from security teams
   - Lack of tools, policies, process, etc.

3. Growth in complex, mission critical online applications
   - Online banking, e-commerce, Web 2.0, etc

4. Number one focus by hackers
   - 75% of attacks focused on applications - Gartner

Result: Application security incidents and lost data on the rise
Agenda

- Introductions & facilities
- Security Landscape

- Vulnerability Analysis
  - Top Attacks Overview
  - Hands on Lab 1

- Vulnerability Analysis (continued)
  - Hands on Lab 2

- Automated Vulnerability Analysis
  - AppScan Overview
  - Hands on Lab 3
Building Security & Compliance into the Software Development Lifecycle (SDLC)

- **Coding**
  - Developers
- **Build**
  - Provides Developers and Testers with expertise on detection and remediation ability
  - Enable Security to effectively drive remediation into development
- **QA**
  - Ensure vulnerabilities are addressed before applications are put into production
- **Security**
- **Production**

Developers

Enable Security to effectively drive remediation into development

Ensure vulnerabilities are addressed before applications are put into production
High Level Web Application Architecture Review

Client Tier (Browser)

Internet

Firewall

SSL

Protects Transport

Protects Network

App Server (Business Logic)

Customer App is deployed here

Sensitive data is stored here

Middle Tier

Web Servers

Data Tier

Databases

Backend Server

© 2008 IBM Corporation
Discovering the Value of Web Application Security Testing with IBM Rational AppScan
Network Defenses for Web Applications

- Perimeter
  - Firewall

- IDS
  - Intrusion Detection System

- IPS
  - Intrusion Prevention System

- App Firewall
  - Application Firewall

System Incident Event Management (SIEM)
Where are the Vulnerabilities?

- Network
- Operating System
- Web Server
- Applications
- Database
- Third-party Components
- Web Server Configuration
- Custom
- Web Applications
- Code Scanning
- Emerging Tech
- Fortify
- Ounce Labs
- Secure Software
- Klockwork
- Parasoft
The Myth: “Our Site Is Safe”

- We Have Firewalls in Place
- We Audit It Once a Quarter with Pen Testers
- We Use Network Vulnerability Scanners
What is a Web Application?

- The business logic that enables:
  - User’s interaction with Web site
  - Transacting/interfacing with back-end data systems (databases, CRM, ERP etc)

- In the form of:
  - 3rd party packaged software; i.e. web server, application server, software packages etc.
  - Code developed in-house / web builder / system integrator

Input and Output flow through each layer of the application

A break in any layer breaks the whole application
## Security Defects: Those I manage vs. Those I own

<table>
<thead>
<tr>
<th></th>
<th>Infrastructure Vulnerabilities or Common Web Vulnerabilities (CWVs)</th>
<th>Application Specific Vulnerabilities (ASVs)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cause of Defect</strong></td>
<td>Insecure application development by 3rd party SW</td>
<td>Insecure application development <strong>In-house</strong></td>
</tr>
<tr>
<td><strong>Location within Application</strong></td>
<td>3rd party <strong>technical building blocks or infrastructure</strong> (web servers,)</td>
<td><strong>Business logic</strong> - dynamic data consumed by an application</td>
</tr>
<tr>
<td><strong>Type(s) of Exploits</strong></td>
<td>Known vulnerabilities (patches issued), misconfiguration</td>
<td>SQL injection, path tampering, Cross site scripting, Suspect content &amp; cookie poisoning</td>
</tr>
<tr>
<td><strong>Detection</strong></td>
<td>Match signatures &amp; check for known misconfigurations.</td>
<td>Requires application specific knowledge</td>
</tr>
<tr>
<td><strong>Business Risk</strong></td>
<td>Patch latency primary issue</td>
<td>Requires automatic application lifecycle security</td>
</tr>
<tr>
<td><strong>Cost Control</strong></td>
<td>As secure as 3rd party software</td>
<td>Early detection saves $$$</td>
</tr>
</tbody>
</table>
Open Web Application Security Project (OWASP) and the OWASP Top 10 list

- Open Web Application Security Project – an open organization dedicated to fight insecure software
- “The OWASP Top Ten document represents a broad consensus about what the most critical web application security flaws are”
- We will use the Top 10 list to cover some of the most common security issues in web applications
<table>
<thead>
<tr>
<th>Application Threat</th>
<th>Negative Impact</th>
<th>Example Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cross-Site® scripting</strong></td>
<td>Identity Theft, Sensitive Information Leakage, …</td>
<td>Hackers can impersonate legitimate users, and control their accounts.</td>
</tr>
<tr>
<td><strong>Injection Flaws</strong></td>
<td>Attacker can manipulate queries to the DB / LDAP / Other system</td>
<td>Hackers can access backend database information, alter it or steal it.</td>
</tr>
<tr>
<td><strong>Malicious File Execution</strong></td>
<td>Execute shell commands on server, up to full control</td>
<td>Site modified to transfer all interactions to the hacker.</td>
</tr>
<tr>
<td><strong>Insecure Direct Object Reference</strong></td>
<td>Attacker can access sensitive files and resources</td>
<td>Web application returns contents of sensitive file (instead of harmless one)</td>
</tr>
<tr>
<td><strong>Cross-Site Request Forgery</strong></td>
<td>Attacker can invoke “blind” actions on web applications, impersonating as a trusted user</td>
<td>Blind requests to bank account transfer money to hacker</td>
</tr>
<tr>
<td><strong>Information Leakage and Improper Error Handling</strong></td>
<td>Attacker can gain detailed system information</td>
<td>Malicious system reconnaissance may assist in developing further attacks</td>
</tr>
<tr>
<td><strong>Broken Authentication &amp; Session Management</strong></td>
<td>Session tokens not guarded or invalidated properly</td>
<td>Hacker can “force” session token on victim; session tokens can be stolen after logout</td>
</tr>
<tr>
<td><strong>Insecure Cryptographic Storage</strong></td>
<td>Weak encryption techniques may lead to broken encryption</td>
<td>Confidential information (SSN, Credit Cards) can be decrypted by malicious users</td>
</tr>
<tr>
<td><strong>Insecure Communications</strong></td>
<td>Sensitive info sent unencrypted over insecure channel</td>
<td>Unencrypted credentials “sniffed” and used by hacker to impersonate user</td>
</tr>
<tr>
<td><strong>Failure to Restrict URL Access</strong></td>
<td>Hacker can access unauthorized resources</td>
<td>Hacker can forcefully browse and access a page past the login page</td>
</tr>
</tbody>
</table>
1. Cross-Site Scripting (XSS)

- What is it?
  - Malicious script echoed back into HTML returned from a trusted site, and runs under trusted context

- What are the implications?
  - Session Tokens stolen (browser security circumvented)
  - Complete page content compromised
  - Future pages in browser compromised
Demonstration – Cross Site Scripting

- Main points covered in the demo:
  - Locating an a place where user input which is echoed back to the browser
  - Seeing if the user input is echoed back ‘as-is’ or if it is properly encoded
  - Exploiting the vulnerability
XSS Example I

HTML code:

```html
<p>No results were found for the query:<br /></p>
<span id="ctl0__ctl0_Content_Main_lblSearch">asdf</span>
```
XSS Example II

HTML code:

```html
<p>No results were found for the query:<br /></p>
<span id="ctl10_ctl10_Content_Main_lblSearch"><script>alert(document.cookie)</script></span></html>
```
Cross-Site Scripting – The Exploit Process

1) Link to bank.com sent to user via E-mail or HTTP

2) User sends script embedded as data

3) Script/data returned, executed by browser

4) Script sends user’s cookie and session information without the user’s consent or knowledge

5) Evil.org uses stolen session information to impersonate user
Agenda

- Introductions & facilities
- Security Landscape
- Vulnerability Analysis
  - Top Attacks Overview
  - Hands on Lab 1
- Vulnerability Analysis (continued)
  - Hands on Lab 2
- Automated Vulnerability Analysis
  - AppScan Overview
  - Hands on Lab 3
Lab 1 Profile Web Application and XSS

- The Goal of this lab is to:
  - profile the demo.testfire.net application
  - utilize a Cross-Site Scripting vulnerability on the demo.testfire.net application in order to access cookies on a target user’s browser
Agenda

- Introductions & facilities
- Security Landscape
- Vulnerability Analysis
  - Top Attacks Overview
  - Hands on Lab 1
  - **Vulnerability Analysis (continued)**
    - Hands on Lab 2
- Automated Vulnerability Analysis
  - AppScan Overview
  - Hands on Lab 3
2 - Injection Flaws

- What is it?
  - User-supplied data is sent to an interpreter as part of a command, query or data.

- What are the implications?
  - SQL Injection – Access/modify data in DB
  - SSI Injection – Execute commands on server and access sensitive data
  - LDAP Injection – Bypass authentication
SQL Injection

- User input inserted into SQL Command:
  - Get product details by id:
    Select * from products where id='{$REQUEST["id"]}';
  - Hack: send param id with value ' or '1'='1
  - Resulting executed SQL:
    Select * from products where id='' or '1'='1'
  - All products returned
Demonstration – SQL Injection

- Main points covered in the demo or video:
  - How to find a SQL injection vulnerability
  - How to exploit a SQL injection vulnerability
SQL Injection Example I

![Image of a website login form with SQL injection vulnerability highlighted]
SQL Injection Example II

An Error Has Occurred

Summary:

Syntax error (missing operator) in query expression 'username = '' AND password = 'asdf''.

Error Message:

```
```
SQL Injection Example - Exploit

```
Username: or 1=1--
Password:  
```
SQL Injection Example - Outcome

The Altoro Mutual website is published by Watchfire, Inc. for the sole purpose of demonstrating the effectiveness of Watchfire products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided “as is” without warranty of any kind, either express or implied. Watchfire does not assume any risk in relation to your use of this website. For additional Terms of Use, please go to http://www.watchfire.com/statements/terms.aspx.

Copyright © 2007, Watchfire Corporation. All rights reserved.
Injection Flaws (SSI Injection Example)
Creating commands from input
The return is the private SSL key of the server

```plaintext
2i5d4YkhwYJhWrlKL7NnUWhgpg/k93E/KAAQWszTvhsNq7742VLsjwvpIWfgnuKq
dfHfJ6fMKaEtBGOf4Sz9kfMx5qhpaKWVuijoRtRlp0IqoZPWO5NvE0z2KC83kcG
```

We need a city, state; city, state zip; or a zip to generate a map
3 - Malicious File Execution

- What is it?
  - Application tricked into executing commands or creating files on server

- What are the implications?
  - Command execution on server – complete takeover
  - Site Defacement, including XSS option
Malicious File Execution – Example I
Malicious File Execution – Example cont.

```csharp
<%@ Page Language="C#" %>
<% Response.Write(System.IO.File.ReadAllText("C:/windows/system32/drivers/etc/hosts")); %>
Malicious File Execution – Example cont.

asdf, asdf, asdf. # Copyright (c) 1993–1999 Microsoft Corp. # # This is a sample HOSTS file used by Microsoft TCP/IP for Windows. # # This file contains the mappings of IP addresses to host names. Each # entry should be kept on an individual line. The IP address should # be placed in the first column followed by the corresponding host name. # The IP address and the host name should be separated by at least one # space. # # Additionally, comments (such as these) may be inserted on individual # lines or following the machine name denoted by a # symbol. # # For example: # # 102.54.94.97 rhino.acme.com # source server # 38.25.63.10 x.acme.com # x client host 127.0.0.1 localhost
4 - Insecure Direct Object Reference

● What is it?
  ▶ Part or all of a resource (file, table, etc.) name controlled by user input.

● What are the implications?
  ▶ Access to sensitive resources
  ▶ Information Leakage, aids future hacks
**Deposit Products**

At Altoro Mutual, we offer business deposit products designed to help you manage your money and grow your business including:

- Commercial Savings Accounts
- Commercial Money Market Accounts
- Time Deposits
- High Yield Investments

For more information about these products, please contact Altoro Mutual.

Note: all Altoro Mutual business deposit accounts include free access to Altoro Mutual's secure, Online Banking site, where you can view account information, make payments and transfers and more.
Insecure Direct Object Reference – Example Cont.

http://www.testfire.net/default.aspx?content=../boot.ini

Error! File must be of type txt or htm
Insecure Direct Object Reference – Example Cont.

http://www.testfire.net/default.asp?content=../boot.ini%00.htm

[boot loader]timeout=30default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS[operating systems]multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"
/noexecute=optin /fastdetect
5 - Information Leakage and Improper Error Handling

- What is it?
  - Unneeded information made available via errors or other means.

- What are the implications?
  - Sensitive data exposed
  - Web App internals and logic exposed (source code, SQL syntax, exception call stacks, etc.)
  - Information aids in further hacks
Information Leakage - Example

<h1>Online Banking Login</h1>

<!-- To get the latest admin login, please contact SiteOps at 415-555-6159 -->

<p><span id="ctl0__ctl0_Content_Main_message"</span></p>

The Altoro Mutual website is published by Watchfire, Inc. for the sole purpose of demonstrating the effectiveness of Watchfire
Improper Error Handling - Example

An Error Has Occurred

Summary:

Syntax error (missing operator) in query expression 'username = '' AND password = 'asdf'".

Error Message:

Information Leakage – Different User/Pass Error

Online Banking Login

Login Failed - Invalid Password

Username: jsmith
Password: 
Login

Online Banking Login

Login Failed - Invalid Username

Username: nouser
Password: 
Login
6 - Failure to Restrict URL Access

● What is it?
  ▶ Resources that should only be available to authorized users can be accessed by forcefully browsing them

● What are the implications?
  ▶ Sensitive information leaked/modified
  ▶ Admin privileges made available to hacker
Failure to Restrict URL Access - Admin User login

/admin/admin.aspx
Simple user logs in, forcefully browses to admin page
Failure to Restrict URL Access: Privilege Escalation Types

- Access given to completely restricted resources
  - Accessing files that shouldn’t be served (*.bak, “Copy Of”, *.inc, *.cs, ws_ftp.log, etc.)

- Vertical Privilege Escalation
  - Unknown user accessing pages past login page
  - Simple user accessing admin pages

- Horizontal Privilege Escalation
  - User accessing other user’s pages
  - Example: Bank account user accessing another’s
Agenda

- Introductions & facilities
- Security Landscape
- Vulnerability Analysis
  - Top Attacks Overview
  - Hands on Lab 1
- Vulnerability Analysis (continued)
  - Hands on Lab 2
- Automated Vulnerability Analysis
  - AppScan Overview
  - Hands on Lab 3
Lab 2

Lab 1 – Profile Web Application, Steal Cookies

Lab 2 – Login without Credentials, Steal Usernames and Passwords, Logging into the Administrative Portal

Lab 3 – Automated Scan of Website
Agenda

- Introductions & facilities
- Security Landscape
- Vulnerability Analysis
  - Top Attacks Overview
  - Hands on Lab 1
- Vulnerability Analysis (continued)
  - Hands on Lab 2
- **Automated Vulnerability Analysis**
  - AppScan Overview
  - Hands on Lab 3
Watchfire in the Rational Portfolio

SOFTWARE QUALITY SOLUTIONS

Test and Change Management

- Requirements
  - Rational RequisitePro
- Test
  - Rational ClearQuest
- Change
  - Rational ClearQuest

Test Automation

- Developer Test
  - Rational PurifyPlus
  - Rational Test RealTime
- Functional Test
  - Automated
    - Rational Functional Tester
  - Manual
    - Rational Manual Tester
- Security and Compliance Test
  - AppScan
  - PolicyTester
- Performance Test
  - Rational Performance Tester

Quality Metrics

- Project Dashboards
- Detailed Test Results
- Quality Reports

Interoperability

ADA 508, GLBA, Safe Harbor
Quality, Brand, Search, Inventory
AppScan

- What is it?
  - AppScan is an automated tool used to perform vulnerability assessments on Web Applications
- Why do I need it?
  - To simplify finding and fixing web application security problems
- What does it do?
  - Scans web applications, finds security issues and reports on them in an actionable fashion
- Who uses it?
  - Security Auditors – main users today
  - QA engineers – when the auditors become the bottle neck
  - Developers – to find issues as early as possible (most efficient)
How Does Application Security Testing work?

1. Explore source code and/or web site to detect structure

2. Identify Vulnerabilities ranked after severity and show how it was identified

3. Advanced remediation, fix recommendations and security enablement
Watchfire Application Security Testing Products

AppScan Enterprise

Web Application Security Testing Across the SDLC

- ASE QuickScan
- AppScan QA
- AppScan Audit
- AppScan MSP

- Application Development
- Quality Assurance
- Security Audit
- Production Monitoring

- Test Applications As Developed
- Test Applications As Part of QA Process
- Test Applications Before Deployment
- Monitor or Re-Audit Deployed Applications
Actionable Fix Recommendations

Blind SQL Injection

Fix Recommendation

General

There are several issues whose remediation lies in sanitizing user input. By verifying that user input does not contain hazardous characters, it is possible to prevent malicious users from causing your application to execute unintended operations, such as launch arbitrary SQL queries, embed JavaScript code to be executed on the client side, run various operating system commands etc.

It is advised to filter out all the following characters:

- [ ] (pipe sign)
- [ ] (ampersand sign)
- [ ] (semicolumn sign)
AppScan with QA Defect Logger for ClearQuest®
Lab 3 overview

- The goal of this lab is to use AppScan in order to automate the detection of vulnerabilities within a web application
Session summary

- Understand the web application environment
- Understand and differentiate between network and application level vulnerabilities
- Understand where the vulnerabilities exist
- Hands on exercises to understand types of vulnerabilities
- Hands on exercise to leverage automated scan for vulnerabilities
Questions
Next steps

- We can schedule a Vulnerability Assessment of one of your Applications –

- For questions and inquiries please contact:
  - Greg Pedley: grpedley@au1.ibm.com
  - Lee Kinsman: lkinsman@au1.ibm.com
Thank You

We appreciate your feedback. Please fill out the survey form in order to improve this educational event.