T12

Class

6/11/2009 12:45:00 PM

"Integrating Security Testing into the QA Process"

Presented by:

Mike Hryekewicz

Standard Insurance Company

Presented at:

Better Software Conference & EXPO 2009

Las Vegas, Nevada

330 Corporate Way, Suite 300, Orange Park, FL 32073

888-268-8770 · 904-278-0524 · sqeinfo@sqe.com · www.sqe.com

Mike Hryekewicz

Mike Hryekewicz is the lead QA security engineer and performance engineer for Standard Insurance Company in Portland, Oregon. Because software security is related to reliability, Mike focuses his efforts on integrating the testing activities and toolsets of the performance and security domains—a resource leak detected in the former could lead to vulnerability in the latter. He has more than twenty years of experience in software development, software architecture, systems analysis, and quality assurance in a wide variety of environments. Mike holds certifications in secure software development (GSSP-Java) and in the secure implementation of the software development lifecycle (CSSLP). He can be reached at mhryekew@standard.com.

A Step-Wise Strategy for Integrating Security Testing into the QA Process

Mike Hryekewicz, GSSP-Java, CSSLP
QA Engineer III: Software Security and Performance
Standard Insurance Company

Standard Insurance Company

- Headquartered in Portland, Oregon
- Primary products:
  - Group disability insurance
  - Individual disability insurance
  - Group life insurance
  - Group dental insurance
  - Group accidental death and dismemberment insurance
  - Retirement plans
  - Annuities

Today's Agenda:

1. Why should QA organizations be involved in security testing?
2. How can a QA organization verify an application's security implementation as a quality attribute?
3. What are some resources for jumpstarting a software security program within your QA organization?

Introduction

- *"It goes without saying that you can't build a secure application without performing security testing on it. Yet many software development organizations do not include security testing as part of their standard software development process."*

Foreword to the "OWASP Testing Guide" version 3.0

The Industry IT Myth: "Our Site Is Safe"

"We Have Network Firewalls Installed"

"We Use Network Vulnerability Scanners"

"We Use SSL to Encrypt our Web Traffic"

"All Users of Our Site Are Authenticated"

The Industry IT Reality: 79% of attacks occur at the application layer

Theresa Lanowitz, Gartner Research Director:

"The problems of network and physical security within IT have largely been solved, leaving the application layer the most vulnerable."

Why Application Security is a High Priority

- **Cenzic's Web Application Security Trends Report Q3-Q4:**
  - Percentage break-out of reported web-technology vulnerabilities:
    - 79% Web application vulnerabilities
    - 12% Plug-ins and ActiveX vulnerabilities
    - 7% Web browser vulnerabilities
    - 2% Web server vulnerabilities
  - http://www.cenzic.com/downloads/Cenzic_AppSecTrends_Q3-Q4-2008.pdf

Why Application Security is a High Priority (continued)

- **Web applications are the #1 focus of hackers:**
  - 75% of attacks are now directed towards applications (Gartner)[1]
  - Applications are the gateway to sensitive resources (e.g. databases)
  - Customer data is worth more than it used to be (e.g. ID theft, fraud)
- **Most sites are vulnerable:**
  - 90% of sites are vulnerable to application attacks (Watchfire)[2]
  - 78% of easily exploitable vulnerabilities were via Web applications (Symantec)[3]
  - 80% of organizations will experience an application security incident by 2010 (Gartner)[4]
  - 51% of web sites for distributing malicious program are legitimate sites that have been hacked (Websense)[5]
  - 70% of the top 100 most popular sites on the web are either hosting malicious content or contain a hidden redirect -- a figure that increased by 16% over the first half of 2008 (Websense)[6]

Mounting regulatory compliance requirements:

- PCI, GLBA, HIPAA, FISMA, SOX …

Why Application Security is a High Priority (continued)

- **Connectivity:**
  - Growing connectivity of computers through the Internet
  - Rise of web services composed of legacy apps that were never intended to be inter-networked.
- **Complexity:**
  - The lines of code per application is growing, not shrinking.
- **Cost to the Organization:**
  - The costs associated with a data breach involving consumer records have been steadily rising. The average total cost per incident reached $6.65 million last year, up from $6.3 million in 2007 (Ponemon Institute) [1]
  - Customer Loss: 31% of customers terminated their relationship following a notification of a data breach; 57% said they lost trust and confidence in the organization (Ponemon Institute)[2]

Examples of Software Security Vulnerabilities
(The OWASP "Top 10")

Application Threat	Negative Impact	Example Impact
Cross Site scripting	Identity Theft, Sensitive Information Leakage	Hackers can impersonate legitimate users, and control their accounts
Injection Flaws	Attacker can manipulate queries to the DB / LDAP / Other system	Hackers can access backend database information, alter it or steal it
Malicious File Execution	Execute shell commands on server, up to full control	Site modified to transfer all interactions to the hacker
Insecure Direct Object Reference	Attacker can access sensitive files and resources	Web application returns contents of sensitive file (instead of harmless one)
Cross-Site Request Forgery	Attacker can invoke "blind" actions on web applications, impersonating as a trusted user	Blind requests to bank account transfer money to hacker
Information Leakage and Improper Error Handling	Attackers can gain detailed system information	Malicious system reconnaissance may assist in developing further attacks
Broken Authentication & Session Management	Session tokens not guarded or invalidated properly	Hacker can "force" session token on victim; session tokens can be stolen after logout
Insecure Cryptographic Storage	Weak encryption techniques may lead to broken encryption	Confidential information (SSN, Credit Cards) can be decrypted by malicious users
Insecure Communications	Sensitive info sent unencrypted over insecure channel	Unencrypted credentials "sniffed" and used by hacker to impersonate user
Failure to Restrict URL Access	Hacker can access unauthorized resources	Hacker can forcefully browse and access a page past the login page

Implementing Software Security in QA

- **A QA Security Program Cannot Exist In A Vacuum Within The Organization**
  - The security vulnerabilities are caught late in the lifecycle, when they are expensive to fix
  - If the developers don't understand the vulnerabilities that QA identifies, they won't know how to fix them
  - Similarly, without this understanding, the developers will unknowingly create new vulnerabilities, further compounding the problem

Implementing Software Security

An effective program requires ALL parties to participate in identifying and preventing security issues within the product.

Architects

Developers

Systems Analysts

QA

Implementing Software Security in QA

- **Start by identifying a security champion or evangelist**
  - A knowledgeable individual to drive the implementation
  - Forms partnerships between QA, Information Security, and the developers
  - Good communication skills and organizational skills
  - Preferably with a development and testing background

Implementing Software Security in QA

- **Implement software security training for QA and developers**
  - What are the most common security-related coding errors? (e.g. CWE “Top 25”)
  - How do these relate to the most common vulnerabilities? (e.g. OWASP “Top 10”)
  - What are the best development practices to correct these errors?
  - Cover both platform-neutral and platform-specific topics

Implementing Software Security in QA

- **Reinforce the training with a static analysis tool**
  - Assists in finding code-related security issues
  - Scans the code on demand, reports identified vulnerabilities, and suggests fixes
  - Many run both in batch mode as well as integrate as plug-ins to common development IDE's (e.g. IBM RAD, Microsoft Visual Studio)
  - Roll out a tuned rule-set to the developers based upon the vulnerabilities of most concern
  - This scales the QA security effort by enlisting the developers earlier in the process
  - Vulnerabilities are less expensive to fix in the development cycle than in the testing cycle

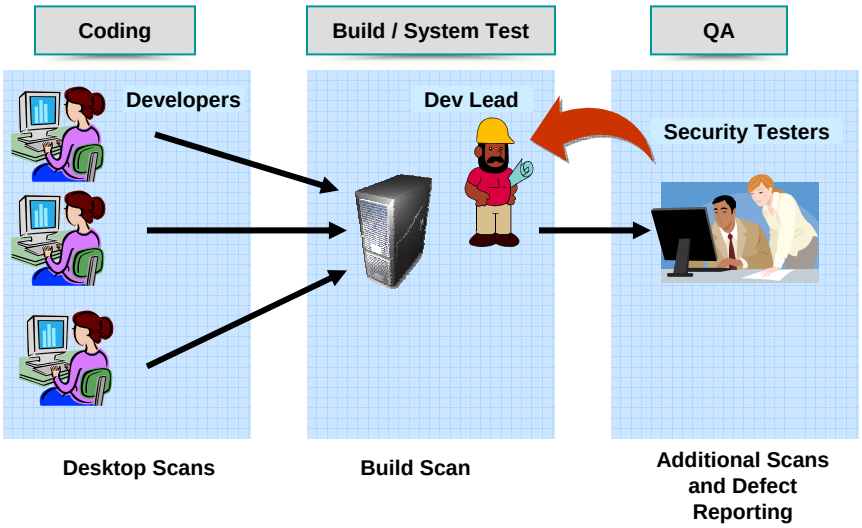
Implementing Software Security in QA

- **Picking a Static Analysis Tool**
  - Open-Source, "Open-ish"-Source, and Freeware Tools (verify licensing restrictions):
    - Splint (http://www.splint.org/)
    - Flawfinder (http://www.dwheeler.com/flawfinder/)
    - Microsoft FxCop (http://msdn.microsoft.com/en-us/library/bb429476.aspx)
    - OWASP LAPSE (http://www.owasp.org/index.php/Category:OWASP_LAPSE_Project)
  - Example Commercial Tools:
    - Fortify
    - Coverity
    - Ounce Labs
    - Klocwork
    - Parasoft

Implementing Software Security in QA

- **The Development/QA handoff**
  - Developers must submit a clean scan of their code by the static analysis tool upon delivery to QA
  - QA enables additional rule-sets and employs additional static analysis tools, providing a more thorough scan of the application
  - The benefit is that more vulnerabilities are identified; the drawback is that there are more false-positives to weed through.
  - High-severity vulnerabilities are tagged as defects and reported to the developers for repair

The Development / QA Flow



Implementing Software Security in QA

- **Questions to consider:**
  - How will you assess the severity of the identified vulnerabilities?
  - How will you differentiate between must-fix versus should-fix vulnerabilities?
  - How will defects be reported and tracked?
  - Who will have visibility into the defect repository?
  - How will this information be classified within your organization?
  - How will these practices integrate with the efforts of your Information Security department?
  - What metrics will you track?

Implementing Software Security in QA

- **Assessing the severity/risk of the vulnerabilities:**
  - DREAD (Damage potential, Reproducibility, Exploitability, Affected users, Discoverability)
  - CVSS (Common Vulnerability Scoring System)
  - Microsoft Security Response Center Security Bulletin Severity Rating System (Critical, Important, Moderate, Low)
  - Risk = Impact * Likelihood

Implementing Software Security in QA

- **Useful Security Metrics To Track:**
  - Goal Question Metric (GQM) Approach (ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf)
  - Sample Questions:
    - How effective is the QA security assessment process in reducing vulnerabilities from the starting code base to the final code release?
    - How effective is the security training process in reducing the number of vulnerabilities that are detected in the code-base when it is delivered to QA?
    - How do the tools compare in terms of the severity and number of defects detected? Which ones should we continue paying maintenance for?
    - Are the majority of our defects related to specification or implementation? Where do we need to emphasize future training?
    - What vulnerability types occur most frequently in our code? Where do we need to reinforce our training efforts?
  - Sample Metrics:
    - Vulnerability density (an industry-standard metric of questionable value)
    - Number, type, and severity of defects identified by QA for each assessment
  - It's useful to capture metrics at the application level (type of app, underlying framework, age, implementation language, whether the team was trained on security) and at the defect level (how detected, implementation vs. specification, detection tool)

Implementing Software Security in QA

- **Recording the Defects – Useful Attributes to Feed the Metrics:**
  - Description of the vulnerability
  - Result of the successful execution of the vulnerability
  - Business impact of the vulnerability
  - Root cause
  - Proposed remediation
  - Defect type (architectural, configuration mgmt., specification, implementation)
  - Detection method (code review, static analysis, automated pen-test)
  - Detection tool
  - Vulnerability type (based upon the Common Weakness Enumeration (CWE))
  - Source file
  - Line number
  - Singular/Compound (is this a single defect or a set of related defects?)

Implementing Software Security in QA

- **Security Assessment Reporting:**
  - Overview of the conducted assessment
  - Explanation of the risk-rating system used
  - Detected Vulnerabilities:
    - A summary based on severity and category
    - Followed by the details, as extracted from the defect-tracking system
  - Recommendations:
    - Quick Hits
    - Tactical Fixes (Ordered by Priority)
    - Strategic Fixes
  - Also see the “OWASP Testing Guide” (ver. 3), Chapter 5 for additional reporting tips

Implementing Software Security in QA

- **Implementing and Securing the Security-Testing Environment:**
  - Protected network storage location
  - Local disk encryption on the assessment workstations
  - Virtualization of the testing environments (VB6, RAD7, Visual Studio)

Implementing Software Security in QA

- **The remaining pieces of the security solution:**
  - Only half of software security vulnerabilities are related to the implementation; the other half are related to the specification
  - Security training for systems analysts and architects (and managers!)
  - Architectural risk analysis and threat modeling
  - "Attack" use-cases ("abuse cases")

Implementing Software Security in QA

- **Automated tools can find, at best, half the common security defects in software[1]**

- **Additional QA security practices to supplement the static analysis:**
  - Application Penetration Testing (both automated tools and manual techniques)
  - Dynamic Taint Propagation (Attack Tracing) during functional testing
  - Coordinating with Information Security and Network Operations on Network Penetration Testing
  - Security-Oriented Code Walkthroughs (static analysis supplements this, but doesn't replace it)
  - Architectural Risk Assessment (Participation or Review)
  - Attack Use Cases or Abuse Cases (Participation or Review)
  - Development of formal security test cases

Implementing Software Security in QA (The OWASP Testing Framework)

- **During Definition and Design:**
  - Review security requirements for testability and potential gaps.
  - Review the design and architecture to ensure that they enforce the appropriate level of security as defined in the requirements.
  - Create and review threat models, based upon the above items.
- **During Development:**
  - Code walkthroughs: to gain an understanding of the application
  - Code reviews: against security requirements and security checklists
- **During Deployment:**
  - Application penetration testing and configuration management testing
- **Post-Deployment:**
  - Periodic security assessments

Application Penetration Testing – Examples

- SSL/TLS Testing
- DB Listener Testing
- Infrastructure configuration management testing
- Application configuration management testing
- Testing for File extensions handling
- Old, backup and unreferenced files
- Infrastructure and Application Admin Interfaces
- Testing for HTTP Methods and XST
- Authentication Test
- Credentials transport over an encrypted channel

.
.
.

Application Penetration Testing – Tools

- **Picking an Application-Penetration Testing Tool**
  - Useful pre-packaged collections of tools:
    - The OWASP Live CD Project: http://www.owasp.org/index.php/Category:OWASP_Live_CD_Project
    - The Samuri Web Testing Framework: http://samurai.inguardians.com/
    - For an extensive tool list, see “Open Source Black Box Testing Tools” in Appendix A of the “OWASP Testing Guide” (version 3)
  - Example Commercial Tools:
    - IBM/Rational AppScan
    - HP WebInspect
    - Cenzic Hailstorm
    - NT Objectives NTOSpider

Penetration Testing Do’s and Don’ts

- **Keeping Out of Jail:**
  - If you don’t own the target application, don’t attempt to penetration-test it!
- **Keeping Your Job:**
  - Establish a security-testing discipline – duties, objectives, and goals
  - Treat penetration-testing like performance-testing – assume you’ll crash your system
  - Get written permission from as high within the organization as possible
  - Write your security test plan
  - Coordinate and communicate your testing exercise
  - Establish rollback and system recovery procedures
  - THEN conduct your penetration tests
  - “I was only performing security research” is not a valid excuse in most companies

Penetration Testing – Pre-Test Mapping

- **Map the Security Testing Environment:**
  - Obtain or develop the environmental deployment diagram for the test-target
  - Identify the hosts, control flows, and data flows that support the application
  - Identify any external dependencies (does this thing send emails to anyone or interface with any 3rd-parties, even in the non-production environments)?
  - Identify specific host names, support personnel, and procedures for notification of induced outages, particularly during off-hours!
  - This exercise is useful for multiple purposes:
    - Threat modeling the application
    - Understanding the potential impact of your penetration tests
    - Knowing who to contact and how, in the event you bring down part of the system

Penetration Testing Coordination Checklist

- Negotiate data backup and refresh procedures with the data analyst or DBA
- Negotiate a testing timeslot with all impacted parties (project team, platform administrators, DBAs, Information Security, environment manager, etc.)
- Communicate in advance, the time, date, target-application, duration, and environment the test will be conducted within.
- Also communicate the IP address and host-name that will be used to launch these tests.
- Your testing activity will very likely set off monitoring alarms within the organization. Ensure that the people who monitor these alarms are in the loop prior to the commencement of your testing.
- Verify the correct operation of the application both before and after the test.

Required Skill Sets for QA Security Testing & Analysis

- **For a Black-Box software testing role:**
  - Functional testing skills
  - Understanding of common weaknesses and vulnerabilities in the implementation technologies
  - “Black Hat” mindset
  - Familiarity with tools of the trade and a deep understanding of what they manipulate

- **For a White-Box software analysis role:**
  - Software engineering background is essential
  - Understanding of the underlying programming languages, frameworks, development tools, hosting environments, and related libraries
  - Knowledge of the security weaknesses inherent in each of the above
  - Understanding of security-related programming best-practices and the ability to identify security mistakes and anti-patterns when examining the design, code, and static analysis results
  - Ability to efficiently and accurately weed out false-positives
  - Ability to recommend suggested fixes based upon latest industry research

Resources To Get You Started

- **Web Sites:**
  - Open Web Application Security Project (OWASP): http://www.owasp.org
  - Common Weakness Enumeration (CWE): http://cwe.mitre.org/data/index.html
  - SANS (http://www.sans.org)
  - Microsoft Security Developer Center: http://msdn.microsoft.com/en-us/security/aa570401.aspx
  - Secure Coding Guidelines for Java: http://java.sun.com/security/seccodeguide.html

- **Conferences:**
  - RSA Conference
  - Black Hat
  - OWASP AppSec Conference
  - SANS Conferences

Resources To Get You Started

- **Software Security Books & References:**
  - *Software Security* by Gary McGraw
  - *19 Deadly Sins of Software Security* by Michael Howard, David LeBlanc, and John Viega
  - *Secure Programming with Static Analysis* by Brian Chess and Jacob West
- **Security Testing Books & References:**
  - *OWASP Testing Guide* (http://www.owasp.org/index.php/Category:OWASP_Testing_Project)
  - *Web Security Testing Cookbook* by Paco Hope and Ben Walther
  - *The Art of Software Security Testing* by Chris Wysopal, Lucas Nelson, Dino Dai Zovi, and Elfriede Dustin
  - *The Art of Software Security Assessment* by Mark Dowd, John McDonald, and Justin Schuh

Resources To Get You Started

- **Security Testing Exercises and Tutorials**
  - WebGoat: a deliberately insecure J2EE web application maintained by OWASP designed to teach web application security lessons. http://www.owasp.org/index.php/OWASP_WebGoat_Project
  - The Foundstone (a division of McAfee) Hackme Series:
    - Hackme Bank
    - Hackme Travel
    - Hackme Casino
    - Hackme Books
    - http://www.foundstone.com/us/resources-free-tools.asp

Resources To Get You Started

- **Security Methodologies and Practices:**
  - OWASP "CLASP": http://www.owasp.org/index.php/Category:OWASP_CLASP_Project
  - Cigital's "Touchpoints": http://www.cigital.com/training/touchpoints/
  - Microsoft's Security Development Lifecycle (SDL): Book: *The Security Development Lifecycle* by Michael Howard and Steve Lipner
- **Security Maturity Models and Implementation Strategies:**
  - Building Security In Maturity Model (BSIMM): http://www.bsi-mm.com/
  - Software Assurance Maturity Model (SAMM): http://www.opensamm.org/

Q & A

What Questions Do You Have?

BETTER SOFTWARE CONFERENCE & EXPO