Streamlining Application Vulnerability Management: Communication Between Development and Security Teams

October 13, 2012
OWASP Boston Application Security Conference
Agenda

• Introduction / Background
• Vulnerabilities
  – Infrastructure (Network) vs. Application (Software)
• Roles
  – Security vs. Development
• Vulnerability Workflow
• ThreadFix: An Open Source Tool
• Questions
Introduction / Background

• Me (Brian Mather)
  – Product & Consulting Manager at Denim Group
  – 5 years experience managing app development & security assessment projects
  – 13 years in information technology/security industry
  – Managing partner at IT services company for 10 years

• Denim Group
  – Headquarters in San Antonio, TX
  – Professional services firm that offers a unique service blend
    • Builds & secures enterprise applications
    • Application Security
    • Developer Education (ILT & eLearning)
  – Customer base spans Fortune 500
    • Market Focus: Financial Services, Banking, Insurance, Healthcare, and Defense
  – Contributes to industry best practices through the Open Web Application Security Project (OWASP)
Vulnerabilities: Defined

• Infrastructure (Network):
  – any flaw or weakness in network defense that could be exploited to gain unauthorized access to, damage, or otherwise affect a network

• Application (Software):
  – a weakness in an application, either a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application.

Problem isn’t finding vulnerabilities, it’s fixing them
  – Identifying application-level vulnerabilities via scanning tools, penetration tests and code reviews is only the first step in actually addressing the underlying risk.
# Vulnerability Fun Facts:

<table>
<thead>
<tr>
<th>Industry</th>
<th>Annual Avg. Vulnerabilities</th>
<th>Avg. Time-to-Fix (Days)</th>
<th>Average Remediation</th>
<th>Window of Exposure (Days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>79</td>
<td>38</td>
<td>63%</td>
<td>231</td>
</tr>
<tr>
<td>Banking</td>
<td>17</td>
<td>45</td>
<td>74%</td>
<td>185</td>
</tr>
<tr>
<td>Education</td>
<td>53</td>
<td>30</td>
<td>46%</td>
<td>261</td>
</tr>
<tr>
<td>Financial Services</td>
<td>67</td>
<td>80</td>
<td>63%</td>
<td>227</td>
</tr>
<tr>
<td>Healthcare</td>
<td>48</td>
<td>35</td>
<td>63%</td>
<td>239</td>
</tr>
<tr>
<td>Insurance</td>
<td>92</td>
<td>40</td>
<td>58%</td>
<td>211</td>
</tr>
<tr>
<td>IT</td>
<td>85</td>
<td>35</td>
<td>57%</td>
<td>208</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>30</td>
<td>17</td>
<td>50%</td>
<td>252</td>
</tr>
<tr>
<td>Retail</td>
<td>121</td>
<td>27</td>
<td>66%</td>
<td>238</td>
</tr>
<tr>
<td>Social Networking</td>
<td>31</td>
<td>41</td>
<td>62%</td>
<td>264</td>
</tr>
<tr>
<td>Telecom</td>
<td>52</td>
<td>50</td>
<td>69%</td>
<td>271</td>
</tr>
<tr>
<td>Non-Profit</td>
<td>37</td>
<td>94</td>
<td>56%</td>
<td>320</td>
</tr>
<tr>
<td>Energy</td>
<td>31</td>
<td>4</td>
<td>40%</td>
<td>250</td>
</tr>
</tbody>
</table>

- Average number of serious vulnerabilities found per website per year is 79 **
- Serious Vulnerabilities were fixed in ~38 days **
- Percentage of serious vulnerabilities fixed annually is only 63% **
- Average number of days a website is exposed, at least one serious vulnerability ~231 days

## Vulnerability Remediation Data

<table>
<thead>
<tr>
<th>Vulnerability Type</th>
<th>Sample Count</th>
<th>Average Fix (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dead Code (unused methods)</td>
<td>465</td>
<td>2.6</td>
</tr>
<tr>
<td>Poor logging: system output stream</td>
<td>83</td>
<td>2.9</td>
</tr>
<tr>
<td>Poor Error Handling: Empty catch block</td>
<td>180</td>
<td>6.8</td>
</tr>
<tr>
<td>Lack of Authorization check</td>
<td>61</td>
<td>6.9</td>
</tr>
<tr>
<td>Unsafe threading</td>
<td>301</td>
<td>8.5</td>
</tr>
<tr>
<td>ASP.NET non-serializable object in session</td>
<td>42</td>
<td>9.3</td>
</tr>
<tr>
<td>XSS (stored)</td>
<td>1023</td>
<td>9.6</td>
</tr>
<tr>
<td>Null Dereference</td>
<td>157</td>
<td>10.2</td>
</tr>
<tr>
<td>Missing Null Check</td>
<td>46</td>
<td>15.7</td>
</tr>
<tr>
<td>XSS (reflected)</td>
<td>25</td>
<td>16.2</td>
</tr>
<tr>
<td>Redundant null check</td>
<td>21</td>
<td>17.1</td>
</tr>
<tr>
<td>SQL injection</td>
<td>30</td>
<td>97.5</td>
</tr>
</tbody>
</table>
Security Team:

Identify / Communicate Risk
- Penetration Testing
- Application Scanning
- Protecting Assets
- Mitigating Risk

Development Team:

Building Software
- Feature Development
- Application Performance
- Bug Fixes
- Deployments

Typically, teams that find vulnerabilities (Security) don’t know how to fix / remediate

Typically, teams that fix vulnerabilities (Development) don’t understand the potential business risk / impact
Vulnerability Workflow:

• Typical Security Workflow
  – Runs a scan ➔ produce PDF ➔ print/email to development = **BAD**
  – Runs 2 scans ➔ produce 2 PDFs ➔ print/email to development = **WORSE**
  – Runs 2 scans ➔ merging vulnerabilities into excel ➔ print/email to development = **HORRIBLE**

  “Let the negotiations begin”

• Typical Development Workflow
  – Developers informed of vulnerabilities ➔ with little / no context provided (no steps to reproduce)
  – Ticket created in defect tracker *(maybe?)* ➔ assign to developer
  – Developer fixes bug ➔ ticket updated in defect tracker ➔ notify security team of fix *(maybe?)*

  “Can we get back to our development schedule yet?”
Vulnerability Workflow:

• Managing Application Vulnerabilities
  – Actual business risk is challenging to determine
  – More challenging than infrastructure vulnerabilities (patching / configuration changes)
  – Changes to custom code and application-specific business logic
  – Requires coordinated effort between security & development teams

• Inefficient process:
  – Difficulty making sense of and prioritizing data in (overlapping) scanning reports
  – Different teams use different scanning tools (tools use different terms and severities)
  – Lack of centralized management/view
  – Friction/Negative interaction between security & development teams

• Remediation becomes an overwhelming project
  – Security managers need to request time from developers (already-crammed dev/release schedules)
  – Development doesn't have or want to give up time to fix vulnerabilities
  – Hesitation scanning new apps, fear of finding new vulnerabilities when queue isn't clearing fast enough

• Creating trending reports is impractical
  – Lack of visibility across app portfolios
  – Without consistent language and consolidated data, knowing whether your organization is actually reducing the number of vulnerabilities is impossible
“Two teams with different focuses, however both teams play a critical role in the remediation of application vulnerabilities, and need to communicate.”

What can be done to solve this problem?
The ThreadFix Approach

• An open source vulnerability management and aggregation platform that allows software security teams to reduce the time it takes to fix software vulnerabilities

• Freely available under the Mozilla Public License (MPL)

• Download available at: www.denimgroup.com/threadfix
ThreadFix: Accelerate Software Remediation

- **Application Portfolio Management**
  - One central, canonical location to keep track of all of the organization’s applications

- **Vulnerability Import**
  - Supports dynamic and static results from a variety of sources (both commercial and freely available scanning tools, manual testing, and SaaS testing providers)
  - De-duplicate scan results *(1 vulnerability found by 4 tools vs. 4 vulns)*

- **Defect Tracking Integration**
  - Allows application security teams to slice/dice, bundle, and ship vulnerabilities over to development staff using tools they are familiar with and currently using

- **Real-Time Protection Generation**
  - Application-specific rules based on identified vulnerabilities & associated attack data
  - Virtual patching helps protect organization and eliminate false positives blocks

- **Maturity Evaluation**
  - Report on software security program progress
  - Benchmark security practice improvement against industry standards
Supported Tools:

**Dynamic Scanners**
- Burp Suite
- HP WebInspect
- Mavituna Security Netsparker
- Tenable Nessus
- Acunetix
- OWASP Zed Attack Proxy
- Arachni
- Skipfish
- w3aF

**Static Scanners**
- HP Fortify SCA
- Microsoft CAT.NET
- FindBugs
- Ounce IBM Security AppScan Source
- Brakeman

**SaaS Testing Platforms**
- WhiteHat
- Veracode
- QualysGuard WAS 2.0

**IDS/IPS and WAF**
- Snort
- mod_security
- Imperva
- F5
- DenyAll

**Defect Trackers**
- Mozilla Bugzilla
- Atlassian JIRA
Dashboard

- List of development teams in the organization, including number of apps for each team and a summary of the security status of those apps.
ThreadFix Consolidation

• Vulnerability scans are aggregated providing a centralized view of the security status of an application.
Agreeing On The Workload

- Bundling multiple instances of the same vulnerability into a single defect
- ThreadFix integrates with Mozilla Bugzilla and Atlassian JIRA
The Defect Tracking System

- Security analyst exports vulnerabilities with Critical Severity to the Defect Tracking System (Bugzilla in this example).
- The development team then uses Bugzilla to keep track of outstanding bugs and management tasks still to be done.
**Vulnerabilities Now Become Defects**

- Vulnerabilities are packaged in a manner that makes sense to the development team’s workflow.
- These vulnerabilities, now recognized as defects, are transferred to Bugzilla, the platform the development team is used to using.
Defect Categories & Status inside ThreadFix

- Security analyst can see all open vulnerabilities, including defects they are linked to.
- Currently view: none of the bugs have been resolved by the development team.

<table>
<thead>
<tr>
<th>First Defect</th>
<th>Second Defect</th>
<th>Third Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improper Sanitization of Special Elements used in an OS Command (OS Command Injection)</td>
<td>Failure to Control Generation of Code (Code Injection)</td>
<td>Improper Control of Resource Identifiers (Resource Injection)</td>
</tr>
<tr>
<td>Critical</td>
<td>Critical</td>
<td>High</td>
</tr>
<tr>
<td>/demo/OSCommandInjection2.php</td>
<td>/demo/EvalInjection2.php</td>
<td>/demo/OSCommandInjection2.php</td>
</tr>
<tr>
<td>fileName</td>
<td>command</td>
<td>fileName</td>
</tr>
<tr>
<td>10</td>
<td>200</td>
<td>281</td>
</tr>
<tr>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Failure to Preserve Web Page Structure (Cross-site Scripting)</th>
<th>Failure to Preserve Web Page Structure (Cross-site Scripting)</th>
<th>Failure to Preserve Web Page Structure (Cross-site Scripting)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>/demo/XPathInjection2.php</td>
<td>/demo/EvalInjection2.php</td>
<td>/demo/XSS-reflected2.php</td>
</tr>
<tr>
<td>password</td>
<td>command</td>
<td>username</td>
</tr>
<tr>
<td>200</td>
<td>280</td>
<td>280</td>
</tr>
<tr>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Failure to Preserve Web Page Structure (Cross-site Scripting)</th>
<th>Failure to Preserve Web Page Structure (Cross-site Scripting)</th>
<th>Information Leak Through Include Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>/demo/XSS-reflected2.php</td>
<td>/demo/XSS-reflected2.php</td>
<td>/demo/OSCommandInjection2.php</td>
</tr>
<tr>
<td>username</td>
<td>username</td>
<td>fileName</td>
</tr>
<tr>
<td>280</td>
<td>280</td>
<td>281</td>
</tr>
<tr>
<td>OPEN</td>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
A Defect (Security Vulnerability) Is Fixed (Or is it?)

- The developers review the bug containing the Critical vulnerabilities.
- They work with representatives from security to resolve the issue and then mark the bug as fixed in Bugzilla.
Bugzilla Updates Are Synchronized With ThreadFix

- When a ThreadFix update is performed, Bugzilla’s developer notes regarding bug status are synchronized with ThreadFix.
- The security team then performs additional scans to confirm that the bugs have, indeed, been fixed.
Trending Reports Help Improve Quality

By repeating this process over time, the security teams can start to collect trending data about vulnerabilities as well as statistics of how long it is taking to resolve security issues.
Summary

• Communication between security & development teams is inefficient

• Current Vulnerability Management process

• ThreadFix facilitates communication between security & development
  – Integrating with commercial and open source scanners & defect trackers
  – Reducing the time required to fix vulnerable applications.
  – Dramatically simplifying remediation effort required
  – Providing centralized visibility into current security state of applications
  – Giving security ability to benchmark progress & track progress over time

• No licensing fees
  – Freely available under the Mozilla Public License (MPL) via Google Code

• Open community support
Where to Get ThreadFix

- For more information, go to [http://www.denimgroup.com/threadfix](http://www.denimgroup.com/threadfix)
- Directed to a [Google Code Repository](https://code.google.com/p/threadfix) and download the zip file.
- Click on the Threadfix.bat icon in Windows, or, in Linux, navigate to the folder and execute bash threadfix.sh.
- Go on the wiki and open the “Getting Started” file for more step by step directions.
Contact Information

Brian Mather
Product & Consulting Manager
brian@denimgroup.com
(210) 572-4400

www.denimgroup.com
www.threadstrong.com
blog.denimgroup.com