Benchmarking Web Application Scanners for YOUR Organization

Dan Cornell
CTO, Denim Group
@danielcornell
My Background

• Dan Cornell, founder and CTO of Denim Group

• Software developer by background (Java, .NET, etc)

• OWASP San Antonio, Global Membership Committee
Denim Group Background

• Secure software services and products company
  – Builds secure software
  – Helps organizations assess and mitigate risk of in-house developed and third party software
  – Provides classroom training and e-Learning so clients can build software securely

• Software-centric view of application security
  – Application security experts are practicing developers
  – Development pedigree translates to rapport with development managers
  – Business impact: shorter time-to-fix application vulnerabilities

• Culture of application security innovation and contribution
  – Develops open source tools to help clients mature their software security programs
    • Remediation Resource Center, ThreadFix
  – OWASP national leaders & regular speakers at RSA, SANS, OWASP, ISSA, CSI
  – World class alliance partners accelerate innovation to solve client problems
What Do You Want From a Scanner?

- Coverage
- Low False Positives
- Low False Negatives
Scanner Coverage

- You can’t test what you can’t see
- How effective is the scanner’s crawler?
- How are URLs mapped to functionality?
  - RESTful
  - Parameters

- Possible issues:
  - Login routines
  - Multi-step processes
  - Anti-CSRF protection
Are You Getting a Good Scan?

Large financial firm: “Our 500 page website is secure because the scanner did not find any vulnerabilities!”

Me: “Did you teach the scanner to log in so that it can see more than just the homepage?”

Large financial firm: “…”
Can Your Scanner Do This?

• Two-step login procedure:
  – Enter username / password (pretty standard)
  – Enter answer to one of several arbitrary questions

• Challenge was that the parameter indicating the question was dynamic
  – Question_1, Question_2, Question_3, and so on
  – Makes standard login recording ineffective
It All Started With A Simple Blog Post…

• Ran into an application with a complicated login procedure
• Wrote blog post about the toolchain used to solve the problem
• Other scanner teams responded:
  – IBM Rational AppScan
    • http://blog.denimgroup.com/denim_group/2012/04/automated-application-scanning-handling-complicated-logins-with-appscan-only.html
  – HP WebInspect
    • http://blog.denimgroup.com/denim_group/2012/05/handling-challengeresponse-logins-in-hp-webinspect.html
  – Mavituna Security Netsparker
    • http://blog.denimgroup.com/denim_group/2012/05/handling-challengeresponse-logins-in-mavituna-netsparker.html
  – NTOObjectives NTOSpider
    • http://blog.denimgroup.com/denim_group/2012/05/handling-challengeresponse-logins-in-ntospider.html
Scanner Authentication Scenario Examples

• Built as a response to the previously-mentioned blog conversation

• Example implementations of different login routines
  – How can different scanners be configured to successfully scan?

• GitHub site:
  – https://github.com/denimgroup/authexamples
Did I Get a Good Scan?

• Scanner training is really important
  – *Read the Larry Suto reports…*

• Must sanity-check the results of your scans

• What URLs were accessed?
  – *If only two URLs were accessed on a 500 page site, you probably have a bad scan*
  – *If 5000 URLs were accessed on a five page site, you probably have a bad scan*

• What vulnerabilities were found and not found?
  – *Scan with no vulnerabilities – probably not a good scan*
  – *Scan with excessive vulnerabilities – possibly a lot of false positives*
Low False Positives

- Reports of vulnerabilities that do not actually exist
- How “touchy” is the scanner’s testing engine?
- Why are they bad?
  - Take time to manually review and filter out
  - Can lead to wasted remediation time
Low False Negatives

• Scanner failing to report vulnerabilities that do exist

• How effective is the scanner’s testing engine?

• Why are they bad?
  – You are exposed to risks you do not know about
  – You expect that the scanner would have found certain classes of vulnerabilities

• What vulnerability classes do you think scanners will find?
Other Benchmarking Efforts

• Larry Suto’s 2007 and 2010 reports
  – Analyzing the Accuracy and Time Costs of Web Application Security Standards
  – Vendor reactions were … varied
  – [Ofer Shezaf attended this talk at AppSecEU 2012 and had some great questions and comments. See his reactions to the latest Larry Suto scanner report here: http://www.xiom.com/2010/02/09/wafs-are-not-perfect-any-security-tool-perfect.]

• Shay Chen’s Blog and Site
  – http://sectooladdict.blogspot.com/
  – http://www.sectoolmarket.com/

• Web Application Vulnerability Scanner Evaluation Project (wavsep)
  – http://code.google.com/p/wavsep/
So I Should Just Buy the Best Scanner, Right?

- Or the cheapest?
- Well…
  - What do you mean by “best”?

Follow-on questions
- How well do the scanners work on your organization’s applications?
- How many false positives are you willing to deal with?
- What depth and breadth of coverage do you need?
ThreadFix - Overview

• ThreadFix is a software vulnerability aggregation and management system that helps organizations aggregate vulnerability data, generate virtual patches, and interact with software defect tracking systems.

• Freely available under the Mozilla Public License (MPL)

What is a Unique Vulnerability?

- **(CWE, Relative URL)**
  - *Predictable resource location*
  - *Directory listing misconfiguration*

- **(CWE, Relative URL, Injection Point)**
  - *SQL injection*
  - *Cross-site Scripting (XSS)*

- **Injection points**
  - *Parameters – GET/POST*
  - *Cookies*
  - *Other headers*
What Do The Scanner Results Look Like?

- Usually XML
  - *Skipfish uses JSON and gets packaged as a ZIP*

- Scanners have different concepts of what a “vulnerability” is
  - *We normalize to the (CWE, location, [injection point]) noted before*

- Look at some example files

- Several vendors have been really helpful adding additional data to their APIs and file formats to accommodate requests
Why Common Weakness Enumeration (CWE)?

- Every tool has their own “spin” on naming vulnerabilities
- OWASP Top 10 / WASC 24 are helpful but not comprehensive
- CWE is exhaustive (though a bit sprawling at times)
- Reasonably well-adopted standard
- Many tools have mappings to CWE for their results

Main site: [http://cwe.mitre.org/](http://cwe.mitre.org/)
Demo

- Unpack and install ThreadFix
- Use ThreadFix to normalize and report on the use of multiple scanning technologies on a given application
- Import multiple scans and de-duplicate the results

- These screenshots are based on UNTUNED scans and are NOT meant to show a real benchmark of these scanners – only the process
Unzip the ThreadFix Package (like WebGoat!)
Make threadfix.sh Executable

```
inflating: ThreadFix/tomcat/webapps/manager/WEB-INF/jsp/sessionDetail.jsp
inflating: ThreadFix/tomcat/webapps/manager/WEB-INF/jsp/sessionsList.jsp
inflating: ThreadFix/tomcat/webapps/manager/WEB-INF/web.xml
inflating: ThreadFix/tomcat/webapps/manager/xform.xsl
creating: ThreadFix/tomcat/webapps/ROOT/
extracting: ThreadFix/tomcat/webapps/ROOT/asf-logo-wide.gif
inflating: ThreadFix/tomcat/webapps/ROOT/build.xml
inflating: ThreadFix/tomcat/webapps/ROOT/favicon.ico
inflating: ThreadFix/tomcat/webapps/ROOT/index.html
inflating: ThreadFix/tomcat/webapps/ROOT/index.jsp
inflating: ThreadFix/tomcat/webapps/ROOT/RELEASE-NOTES.txt
inflating: ThreadFix/tomcat/webapps/ROOT/tomcat-power.gif
extracting: ThreadFix/tomcat/webapps/ROOT/tomcat.gif
inflating: ThreadFix/tomcat/webapps/ROOT/tomcat.svg
creating: ThreadFix/tomcat/webapps/ROOT/WEB-INF/
inflating: ThreadFix/tomcat/webapps/ROOT/WEB-INF/web.xml
inflating: ThreadFix/tomcat/webapps/threadfix_war
creating: ThreadFix/tomcat/work/
creating: ThreadFix/tomcat/work/Catalina/
creating: ThreadFix/tomcat/work/Catalina/localhost/
creating: ThreadFix/tomcat/work/Catalina/localhost/manager/
creating: ThreadFix/tomcat/work/Catalina/localhost/manager/org/
creating: ThreadFix/tomcat/work/Catalina/localhost/manager/org/apache/
creating: ThreadFix/tomcat/work/Catalina/localhost/manager/org/apache/jsp/
inflating: ThreadFix/tomcat/work/Catalina/localhost/manager/org/apache/jsp/_401.jsp.class
inflating: ThreadFix/tomcat/work/Catalina/localhost/manager/org/apache/jsp/_401.jsp.java

DanCoMacBook:Desktop dcornell$ cd ThreadFix
DanCoMacBook:ThreadFix dcornell$ chmod u+x threadfix.sh
DanCoMacBook:ThreadFix dcornell$
```
Run ThreadFix Pre-Configured Tomcat Server
Login to ThreadFix ("user" and "password")
Upload Various Scan Results Files
This Vulnerability Found By Three Scanners

<table>
<thead>
<tr>
<th>Scanner Name</th>
<th>Severity</th>
<th>Vulnerability Type</th>
<th>Path</th>
<th>Parameter</th>
<th>Number Merged Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arachni</td>
<td>HIGH</td>
<td>Cross-Site Scripting in HTML tag</td>
<td>/wavsep/active/RXSS-Detection-Evaluation-GET/Case03-Tag2TagStructure.jsp</td>
<td>userinput</td>
<td>1</td>
</tr>
<tr>
<td>OWASP Zed Attack Proxy</td>
<td>3</td>
<td>Cross Site Scripting</td>
<td>/wavsep/active/RXSS-Detection-Evaluation-GET/Case03-Tag2TagStructure.jsp</td>
<td>userinput</td>
<td>1</td>
</tr>
<tr>
<td>w3af</td>
<td>Medium</td>
<td>Cross site scripting vulnerability</td>
<td>/wavsep/active/RXSS-Detection-Evaluation-GET/Case03-Tag2TagStructure.jsp</td>
<td>userinput</td>
<td>1</td>
</tr>
</tbody>
</table>

**Surface Location**

Host: satoffice043
Path: /wavsep/active/RXSS-Detection-Evaluation-GET/Case03-Tag2TagStructure.jsp
Protocol: https
Port: 8443
Query: userinput

Back to Application Target Application
Close Vulnerability
Mark as False Positive
Mark False Positives (wavsep Uses “FalsePositives” In the URL...)
Summary Report – Found, Not Found, False Positives (Again – NOT Based on Tuned Scans)

<table>
<thead>
<tr>
<th>Scanner Name</th>
<th># of Vulnerabilities Found</th>
<th>% of Vulnerabilities Found</th>
<th># of Vulnerabilities Missed</th>
<th>% of Vulnerabilities Missed</th>
<th># FPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sketchfish</td>
<td>127</td>
<td>20.45</td>
<td>494</td>
<td>79.55</td>
<td>6</td>
</tr>
<tr>
<td>waf</td>
<td>107</td>
<td>17.23</td>
<td>514</td>
<td>82.77</td>
<td>9</td>
</tr>
<tr>
<td>Arachni</td>
<td>328</td>
<td>46.38</td>
<td>533</td>
<td>53.62</td>
<td>16</td>
</tr>
<tr>
<td>OWASP Zed Attack Proxy</td>
<td>367</td>
<td>59.1</td>
<td>254</td>
<td>40.9</td>
<td>12</td>
</tr>
</tbody>
</table>
# Report By Vulnerability Type

<table>
<thead>
<tr>
<th>Vulnerability Description</th>
<th>Alterni</th>
<th>OWASP Zed Attack Proxy</th>
<th>Sicfish</th>
<th>w3af</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insufficient Sanitation of Special Elements used in an SQL</td>
<td>178</td>
<td>84</td>
<td>32</td>
<td>70</td>
</tr>
<tr>
<td>Failure to Control Generation of Code (Code Injection)</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Failure to Preserve Web Page Structure (Cross-site Scripting)</td>
<td>68</td>
<td>67</td>
<td>17</td>
<td>46</td>
</tr>
<tr>
<td>Cross-Site Request Forging (CSRF)</td>
<td>6</td>
<td>117</td>
<td>53</td>
<td>0</td>
</tr>
<tr>
<td>Information Exposure</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Information Leak Through Browser Cookies</td>
<td>0</td>
<td>197</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Insufficient Validation of Host-specific Certificate Data</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Information Leak Through Server Error Message</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>External Configuration Reference to a Resource in Another</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Failure to Sanitize CRLF Sequences in HTTP Headers (HTTP)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Detail Report Can Be Used To Error-Check Merge Process

<table>
<thead>
<tr>
<th>Failure to Preserve Web Page Structure</th>
<th>userinput</th>
<th>OPEN</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>'/wavsep/active/RXSS-Detection-Evaluation-POST/Case29-Vbs2ScriptTagOLCommentScope.jsp'</td>
<td>userinput</td>
<td>OPEN</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>'/wavsep/active/RXSS-Detection-Evaluation-POST/Case25-Vbs2ScriptTagDoubleQuoteDelimiter.jsp'</td>
<td>userinput</td>
<td>OPEN</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>'/wavsep/active/RXSS-Detection-Evaluation-POST/Case04-Tag2HtmlComment.jsp'</td>
<td>userinput</td>
<td>OPEN</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>'/wavsep/active/RXSS-Detection-Evaluation-GET/Case10-Js2DoubleQuoteJsEventScope.jsp'</td>
<td>userinput</td>
<td>OPEN</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>'/wavsep/active/RXSS-Detection-Evaluation-GET/Case27-Js2ScriptTagOLCommentScope.jsp'</td>
<td>userinput</td>
<td>OPEN</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Current Limitations

• Vulnerability importers are not currently formally vendor-supported
  – Though a number have helped us test and refine them (thanks!)
  – After you get a good scan make sure you also got a good import

• Summary report should show data by severity rating
  – Make it easier to focus on vulnerabilities you probably care more about
  – But you can look at the data by vulnerability type
Try This At Home, Kids

• Pick some applications to test
  – Representative sample for your organization
  – Common languages, frameworks

• Run scans with the targeted scanning technologies
  – Make sure you get good scans: login, other state-based issues
  – If you train the scans (always a good idea) be consistent

• Import the scans into ThreadFix
  – Make sure you’re happy with the import

• Run some reports
You Know What Would Make All This Way Easier?

- Common data standards for scanning tools!

- Current efforts:
  - MITRE Software Assurance Findings Expression Schema (SAFES)
    - [http://www.mitre.org/work/tech_papers/2012/11_3671/](http://www.mitre.org/work/tech_papers/2012/11_3671/)
  - OWASP Data Exchange Format Project
Simple Software Vulnerability Language (SSVL)

- Common way to represent static and dynamic scanner findings
- Based on our experience building importers for ThreadFix
  - It “works” for real-world applications because we are essentially using it

- Love to hear feedback
  - Send me a request and I can share the document for editing/annotation

- Online:
  - [https://docs.google.com/document/d/1H5hWUdj925TtoZ7ZvnfHdFABe7hBCGuZtLUas29yBGl/edit?pli=1](https://docs.google.com/document/d/1H5hWUdj925TtoZ7ZvnfHdFABe7hBCGuZtLUas29yBGl/edit?pli=1)
  - Or [http://tinyurl.com/cslqv47](http://tinyurl.com/cslqv47)
Simple Software Vulnerability Language (SSVVL)

```xml
<Vulnerabilities SpecVersion="0.2" ApplicationTag="PRODUCTION_20110115" ExportTimestamp="2/13/2011 1:45:30 AM -06:00">
  <Vulnerability CWE="89" Severity="CRITICAL">
    <ShortDescription>
      Improper Improper Neutralization of Special Elements used in an SQL Command ("SQL Injection")
    </ShortDescription>
    <LongDescription/>
    <Finding NativeId="12345" Source="IBM Rational AppScan 7.0" IdentifiedTimestamp="12/14/2010 3:42:16 AM -06:00">
      <SurfaceLocation url="http://www.site.com/Login.aspx" source="Parameter" value="txtUser">
      </SurfaceLocation>
    </Finding>
  </Vulnerability>
  <Vulnerability CWE="79" Severity="HIGH">
    <ShortDescription>
      Improper Neutralization of Input During Web Page Generation ("Cross-site Scripting")
    </ShortDescription>
    <LongDescription/>
    <Finding NativeId="23456" Source="Microsoft CAT.NET">
      <DataFlowElement SourceFileName="C:\Projects\SiteWeb\Login.aspx.cs" LineNumber="12" ColumnNumber="61" Sequence="0">
        <LineText>
          String sql = "SELECT * FROM [User] WHERE Username = " + txtUser.Text + ";"
        </LineText>
      </DataFlowElement>
    </Finding>
  </Vulnerability>
</Vulnerabilities SpecVersion="0.2" ApplicationTag="PRODUCTION_20110115" ExportTimestamp="2/13/2011 1:45:30 AM -06:00">
```
Questions

Dan Cornell
dan@denimgroup.com
Twitter: @danielcornell

www.denimgroup.com
www.denimgroup.com/threadfix
code.google.com/p/threadfix
(210) 572-4400