State of The Art: Automated Black Box Web Application Vulnerability Testing

Jason Bau, Elie Bursztein, Divij Gupta, John Mitchell
Background

• Web Application Vulnerability Protection
  • High incidence vulnerabilities (XSS, SQLI, …)
  • Required for standards compliance (e.g. PCI)
Security Tools for Apps

Vulnerability Detection Techniques:

- Manual vs. Automated
- White-Box vs. Black-Box
- Code review, Static analysis, Pen testing
- **Automated Black Box Testing**
  - Cheaper? Less intrusive to workflow?
State of the Art: Automated Black Box Web Application Vulnerability Testing
Goals of Study

- What **vulnerabilities** are tested by scanners?
- How **representative** are scanner tests of in-the-wild vulnerabilities?
- What can user **expect** from scanner?
- What is **hard** and needs more human review?
Non-Goals

• Not a product ranking

• Not a benchmark of particular tools
Take Aways

• How to take advantage of scanner
• How (If) to combine it with human audit
• What to expect as improvement
Outline

• Vulnerability categories tested by scanners
• How prevalent are these in the wild?
• Common application results
• Custom testbed design
• Custom testbed results
  • Coverage
  • Detection
  • False Positives
Survey of Leading Products

Local

Remote

>$100K total retail price
<table>
<thead>
<tr>
<th>Category</th>
<th>Example Vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross Site Scripting</td>
<td>XSS</td>
</tr>
<tr>
<td>SQL Injection</td>
<td>SQLI</td>
</tr>
<tr>
<td>Cross Channel Scripting (Other forms of injection)</td>
<td>Arbitrary File Upload, Remote File Inclusion, OS command Injection</td>
</tr>
<tr>
<td>Session Management</td>
<td>Session Fixation and Prediction Authentication Bypass</td>
</tr>
<tr>
<td>Cross-Site Request Forgery</td>
<td>CSRF</td>
</tr>
<tr>
<td>SSL/Server Config</td>
<td>Self-Signed Cert, HTTP Trace</td>
</tr>
<tr>
<td>Info Leakage</td>
<td>Temp file access, path traversal, Error message disclosure</td>
</tr>
</tbody>
</table>
Test Vectors By Category

Test Vector Percentage Distribution

- Info leaks
- Configuration
- CSRF
- Session
- XCS
- SQLI
- XSS
Data from aggregator and validator of NVD-reported vulnerabilities
Scanners vs. In-the-Wild

• Top 4 for both:
  • XSS
  • SQLI
  • XCS
  • Info Leak

• Scanners have many more info leak vectors
  • Easier to write?
### Detecting Known Vulnerabilities

Vulnerabilities for previous versions of Drupal, phpBB2, and WordPress

<table>
<thead>
<tr>
<th>Category</th>
<th>Drupal 4.7.0</th>
<th>phpBB2 2.0.19</th>
<th>Wordpress 1.5strayhorn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NVD</td>
<td>Scanner</td>
<td>NVD</td>
</tr>
<tr>
<td>XSS</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>SQLI</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XCS</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Session</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>CSRF</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Info Leak</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Good: Info leak, Session (Anecdote from re-test)  
Decent: XSS/SQLI  
Poor: XCS, CSRF (low vector count?)
• Mainly built over summer by 1 undergrad in PHP

• Measure Performance
  o Test Duration / Network Traffic

• Measure Coverage
  o Links coded in various technologies (Flash, SilverLight, …)
  o Can scanner follow link?

• Measure Vulnerability Detection Rate
  o XSS (Type 1, Type 2, Advanced)
  o SQLI (Type 1, Type 2)
  o Cross Channel Scripting
  o CSRF
  o Session Management
  o Server/Crypto Config
  o Information Leak
  o Malware
Performance did not correlate well with vulnerability detection
% Successful Link Traversals By Technology, Averaged over all Scanners
# Vulnerability Detection

## Scanners Overall detection rate

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Detection Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malware</td>
<td>0%</td>
</tr>
<tr>
<td>Info leak</td>
<td>31.2%</td>
</tr>
<tr>
<td>Config</td>
<td>32.5%</td>
</tr>
<tr>
<td>Session</td>
<td>26.5%</td>
</tr>
<tr>
<td>SQL 2nd order</td>
<td>0%</td>
</tr>
<tr>
<td>SQL 1st order</td>
<td>21.4%</td>
</tr>
<tr>
<td>CSRF</td>
<td>17.1%</td>
</tr>
<tr>
<td>XCS</td>
<td>14.4%</td>
</tr>
<tr>
<td>XSS advance</td>
<td>11.25%</td>
</tr>
<tr>
<td>XSS type 2</td>
<td>15%</td>
</tr>
<tr>
<td>XSS type 1</td>
<td>62%</td>
</tr>
</tbody>
</table>

## Context?
XSS Testbed

- Type 1: Textbook “Reflected” Vulnerability
  - User input, http header → page w/o sanitization

- Type 2: Stored Vulnerability
  - User input → DB → Served Page
  - Some viewable only by different user

- Advanced
  - Novel Tags: e.g. <object>, <prompt>
  - Novel Channels:
    - URL → $_SERVER['PHP_SELF']
    - Filename → error msg,
XSS Results

Anecdote about Type 2
• Type 1: User input → SQLI on page generation
  o Basic: ‘ ; --
  o Advanced: “, LIKE, UNION

• Type 2: Input → DB → SQL Query
  o Only basic cases
  o Unsanitized form input (username) → DB, later used in SQL query
SQLI Results

Scanner Detection Rate for SQL injections

<table>
<thead>
<tr>
<th>SQL 2nd</th>
<th>SQL 1st</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.4</td>
<td>14.2</td>
</tr>
<tr>
<td>8th</td>
<td>7th</td>
</tr>
<tr>
<td>7th</td>
<td>6th</td>
</tr>
<tr>
<td>6th</td>
<td>5th</td>
</tr>
<tr>
<td>5th</td>
<td>4th</td>
</tr>
<tr>
<td>4th</td>
<td>3rd</td>
</tr>
<tr>
<td>3rd</td>
<td>2nd</td>
</tr>
<tr>
<td>2nd</td>
<td>1st</td>
</tr>
<tr>
<td>1st</td>
<td>Average</td>
</tr>
<tr>
<td>42.8</td>
<td>28.5</td>
</tr>
</tbody>
</table>

Jason Bau  
State of the Art: Automated Black Box Web Application Vulnerability Testing  
jbau@stanford.edu
• Code Injection by Attacker
• Manipulate server or client browser
• Tests:
  o XPATH injection
  o Malicious File Upload
  o Direct Object Ref
  o Cross-Frame Scripting
  o Open Redirects
  o Server Side Includes
  o Header Injection
  o Flash Parameter Inject
  o SMTP Injection
CSRF Results

- Post-login forms
  - w/o hidden random token
  - with weak [0,9] token
  - with same token each time

- JSON Hijacking
  - No session id sent with AJAX request for sensitive data

- Anecdote: Told by one vendor CSRF not checked on purpose
Session Management

• Login / form errors
  o Login form not https
  o Reg. credentials in clear
  o Autocomplete pwd field
  o Weak pwds and pwd recovery question
  o Weak reg. page CAPTCHA

• Cookie errors
  o Not HttpOnly
  o Auth tokens not https
  o Persistent Auth token value MD5 (pwd)
  o Logout fails to clear cookie
  o Path restriction to '/'
Server/Crypto Mis-Config

- **Server Mis-Config:**
  - HTTP Trace enabled
  - `open_basedir` not set in php
  - `allow_url_fopen` set in php

- **Crypto Mis-Config**
  - Self Signed Cert
  - Weak SSL Cipher
Info Leak

- SQL error message
- Username existence
- Backup files
- Comment/Path Disclosure
- Path Traversal
  - Inclusion of /etc/secret.txt
• JavaScript key-logger on login page

• Malicious graphic uploaded by user
  o .jpg with appended PHP
  o Directly reference-able

• No Scanner Detected
  o Because not part of PCI compliance?
False Positives

• Testbed Traps
  o alert()s as site behavior (not part of injection)
    ▪ Scanners avoided
  o Benign (not-executed) region within <script> tags
    ▪ Tripped 2 scanners (reported 1 and 13 times)

• On a testbed of ~90 confirmed vulnerabilities

![False positive by scanners](chart.png)

• Some scanners with low false positive rates also had high relative detection rates
• No individual scanner tops across all categories
  o Best XSS, SQLI → Bottom 3 Session Management
  o Top 3 Session Management → Found 0 SQLI
  o Rough break along XSS/SQLI/XCS and Session/Config/Info lines

• Scanners exist:
  o High Detection Rate, Low False Positive Rate
  o Low Detection Rate, High False Positive Rate
  o Low Detection Rate, Low False Positive Rate
Conclusions 1

- XSS, SQLI, XCS, Info Leak most common “in-the-wild”
- Black Box Scanner "effort" roughly proportional to this

- Can improve coverage of technologies like Flash, SL

- Scanners relatively adept at detecting
  - Historical vulnerabilities
  - Textbook XSS and SQLI
  - Info Leak, Session, and Server/Crypto Mis-config
    • Easier test vectors to write/interpret
• Can stand improvement on
  o CSRF, Malware, XCS
    ▪ Low test vector count → Not vendor focus?
  o Advanced (novel) forms of XSS, SQLI
    ▪ Faster reactive process
  o Stored forms of XSS, SQLI (acknowledged by a CTO)
    ▪ Better DB modeling