Finding Your Way in Testing Jungle

A Learning Approach to Web Security Testing.
Research Questions

● Why is it important to improve website security?
● What techniques are already in place to test security?
● What are the benefits of a learning based web testing approach?
Why is it important to improve web security?

• Studies show web apps highly vulnerable to security attacks
  o report by WASC lists 97,554 detected vulnerabilities
  o 49% of sites contain high-risk vulnerabilities

• Black-box security testing of web apps is a hard problem
  o only able to find 58.5% of high-risk vulnerabilities and 12.1% of medium risk

• Cross-site scripting (XSS)
  o one of top two web vulnerabilities
Learning Approach to Testing

• Fresh approach - cast into learning setting
• Testing algorithm has large database of test payloads
• If web app’s defenses are broken, one of these payloads is able to demonstrate the vulnerability
• Question: how do we search through payloads to find a good candidate?
XSS Analyzer

• Learning algorithm for black-box detection of XSS vulnerabilities
• 500 million test payloads
• Infers from failed tests which other payloads are also likely to fail and prunes the search space
<?php function filter($str) {
    $str = stripslashes($str);
    $str = eregi_replace("<\[:space:\]*([^ \n]+\n)*<\[:space:\]*","<\[:space:\]*", $str);
    $str = eregi_replace("<a[^\n>]*href=[[:space:]*][^\n>]*\?[^\n>]*[^\n>]*","<a href="", $str);
    $str = eregi_replace("<br[^\n>]*[^\n>]*","", $str);
    $str = eregi_replace("<a[^\n>]*href=[[:space:]*][^\n>]*javascript[^\n>]*","", $str);
    $tmp = ";
    while (ereg("<a([^\n>]*[^\n>]*","$str,$reg) { 
        $i = strpos($str,$reg[0]); $l = strlen($reg[0]); $tmp .= substr($str,0,$i); $str = substr($str,$i+1);
    }
    $str = $tmp . $str;
    $str = htmlentities(trim($str), ENT_QUOTES);
    if ($type != "preview" AND $save != 1) { $str = html_entity_decode($str, ENT_QUOTES); }
    return $str 
?>
Example - PHP Sanitizer

• *filter* function sanitizes its input string
  o deletes all spaces from HTML tags
  o deletes *img* tags
  o deletes javascript directives contained in *href* attribute values
  o deletes all tags with no attributes

• This permits only *<a>* tags with *href* values
Example - PHP Sanitizer

• `<script>alert(‘XSS’) </script>`
• Fails due to removal of tags with no attributes
• Problem with `filter`: user can replace space character with tab character (`\t`)
• `<input autofocus onfocus=’alert(‘XSS’)'>
  o can penetrate through `filter`’s defense
How XSS Analyzer works

1. Sends basic payload, `<script>alert(1)</script>`
   - fails, XSS analyzer learns nothing

2. Sends another:
   `<input type="text" onfocus="alert(‘XSS’)"/>`
   - fails, but teaches XSS Analyzer a constraint, *filter* rejects spaces

3. Sends one with tab character
   `<input autofocus onfocus="alert(‘XSS’)"/>
   - this will demonstrate vulnerability"
What techniques are already in place to test security?

- Black Box Security Testing
- Brute Force Testing
- Random Testing
- Expert Testing
Brute-Force Testing

- Accepts as input list of L payloads and iterates over it trying each payload
- Optimal from coverage standpoint, ensures 0 misses with respect to available payloads
- High cost of HTTP traffic restricts number of payloads that can be spent on a given input
Random Testing

- Parameterized by list of payloads $L$ and sample size $n$
- $n$ payloads sampled at random are tried with brute-force algorithm
- Advantage: prevents biases such as giving more weight to payloads in beginning of list
- Disadvantages:
  - interconnections between payloads ignored
  - random testing not reproducible in general
Expert Testing

- Rely on expert knowledge when making short list of payloads
- Works really well for “average case” defects
- Non-standard or uncommon defects, which are most dangerous, are outside of reach
The Learning Approach

- Using fingerprinting, we test for structural and bypass attacks
- Fingerprints are defined by a certain word being used in the attack
- Bypassing attacks
- Structural attacks
What does the XSS Analyzer actually do?

- Tests to find vulnerabilities in the webpage
- Uses probes to ensure certain attacks are prevented
- Sees where tests fail, and gathers tokens
- tl;dr it's testing the sanitizer for allowing malicious tokens through
Important Concepts

Test Fails: The sanitizer blocks the payload
Payload: Set of tokens that represents a client side injection
Token: a word in a script, examples being ‘onmouseover’ or ‘=’
Sanitizer: removes words from payload
Structural XSS Analyzer

- When a test fails, the analyzer parses the payload to find out what happened for each token.
  - If token is not accepted, add it to the constraint list.
Bypass Strategy

● In addition to structural constraints, the analyzer checks bypass strategies.

● A bypass strategy is if someone is using a similar word that could become a word used in a XSS attack.

● These words are added to a mapping, so the analyzer knows what the word maps back to.
Bypass Example

<script>

- Test fails, but now XSS Analyzer knows that script is a constraint

<SCscriptRIPT>

- Test passes, uncovers a flaw in the filter which allows the bypass SCscriptRIPT to be used
This all works together

Create a new set of constraints, structural and bypass while there are more payloads

traverse tokens
replace bypass tokens with mapping
test if modified payload is blocked by sanitizer
if blocked :
    run structural analysis
    for each token sanitized, try bypass mapping
if accepted : try next payload
What are the benefits of a learning based web testing approach?

- Better
- Quicker
- More awesome
Testing Experiment

XSS Analyzer, along with 21 other testing alternatives were compared with one another

- 15,552 different server side defenses
- defenses contain wide range of sanitization and validation strategies
Performance and Coverage

Coverage: measured by the total number of vulnerabilities detected

Performance: average the number of requests sent by the testing algorithm in total
Analysis of Results

Two issues addressed: (1) overall value of each algorithm and (2) viability of random algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Vuln.s</th>
<th>Coverage</th>
<th>Requests (avg.)</th>
<th>Total</th>
<th>Success</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>brute force</td>
<td>10356</td>
<td>100%</td>
<td>2301</td>
<td>95</td>
<td>6481</td>
<td></td>
</tr>
<tr>
<td>analyzer</td>
<td>10245</td>
<td>99%</td>
<td>10</td>
<td>6</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>AppScan</td>
<td>4406</td>
<td>43%</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>R100</td>
<td>5659</td>
<td>55%</td>
<td>67</td>
<td>3</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>R200</td>
<td>6311</td>
<td>61%</td>
<td>123</td>
<td>6</td>
<td>209</td>
<td></td>
</tr>
<tr>
<td>R250</td>
<td>6591</td>
<td>64%</td>
<td>148</td>
<td>8</td>
<td>260</td>
<td></td>
</tr>
<tr>
<td>R300</td>
<td>6725</td>
<td>65%</td>
<td>174</td>
<td>9</td>
<td>311</td>
<td></td>
</tr>
<tr>
<td>R400</td>
<td>7021</td>
<td>68%</td>
<td>219</td>
<td>12</td>
<td>411</td>
<td></td>
</tr>
<tr>
<td>R500</td>
<td>7291</td>
<td>70%</td>
<td>263</td>
<td>15</td>
<td>511</td>
<td></td>
</tr>
<tr>
<td>R600</td>
<td>7338</td>
<td>71%</td>
<td>312</td>
<td>17</td>
<td>613</td>
<td></td>
</tr>
<tr>
<td>R700</td>
<td>7567</td>
<td>73%</td>
<td>349</td>
<td>20</td>
<td>710</td>
<td></td>
</tr>
<tr>
<td>R800</td>
<td>7620</td>
<td>74%</td>
<td>396</td>
<td>21</td>
<td>811</td>
<td></td>
</tr>
<tr>
<td>R900</td>
<td>7725</td>
<td>75%</td>
<td>437</td>
<td>24</td>
<td>910</td>
<td></td>
</tr>
<tr>
<td>R1000</td>
<td>7741</td>
<td>75%</td>
<td>481</td>
<td>25</td>
<td>1010</td>
<td></td>
</tr>
<tr>
<td>R1500</td>
<td>8046</td>
<td>78%</td>
<td>681</td>
<td>34</td>
<td>1500</td>
<td></td>
</tr>
<tr>
<td>R2000</td>
<td>8229</td>
<td>79%</td>
<td>860</td>
<td>42</td>
<td>1983</td>
<td></td>
</tr>
<tr>
<td>R2500</td>
<td>8294</td>
<td>80%</td>
<td>1062</td>
<td>51</td>
<td>2472</td>
<td></td>
</tr>
<tr>
<td>R3000</td>
<td>8378</td>
<td>81%</td>
<td>1235</td>
<td>56</td>
<td>2953</td>
<td></td>
</tr>
<tr>
<td>R3500</td>
<td>8569</td>
<td>83%</td>
<td>1385</td>
<td>66</td>
<td>3421</td>
<td></td>
</tr>
<tr>
<td>R4000</td>
<td>8611</td>
<td>83%</td>
<td>1554</td>
<td>72</td>
<td>3896</td>
<td></td>
</tr>
<tr>
<td>R4500</td>
<td>8660</td>
<td>84%</td>
<td>1724</td>
<td>77</td>
<td>4371</td>
<td></td>
</tr>
<tr>
<td>R5000</td>
<td>8713</td>
<td>84%</td>
<td>1890</td>
<td>88</td>
<td>4845</td>
<td></td>
</tr>
</tbody>
</table>
Measuring the effectiveness of each of the testing algorithms by computing a normalized ratio between the total number of detected vulnerabilities, \(v\), and the total number of requests \(r\)

\[ \log(10^3 \times (v/r)) \]
Overall Value

Normalized ratio between findings and sent requests
Overall Value

Coverage vs Performance

[Scatter plot showing coverage vs performance for different tools and techniques.]
The rate at which coverage improves decays significantly.
In conclusion..

- XSS Analyzer is being enhanced
  - Glass box
  - Address more issues
- Developed by IBM, learn more about it at their blogspot!

[tinyurl.com/xssanalyzer](tinyurl.com/xssanalyzer)
Questions

What are the benefits of a learning based web testing approach?

Why is R100 considered to be wasteful?

How does the XSS Analyzer complement the sanitizer?
Questions

How could the learning technique be applied to non-web based programs?

How may this approach be data heavy?

Why is black-box security testing generally a hard problem?