Enemy of the State: A State-Aware Black-Box Web Vulnerability Scanner

Adam Doupé, Ludovico Cavedon, Christopher Kruegel, and Giovanni Vigna

University of California, Santa Barbara

USENIX 2012 – 8/10/12
Web Applications Have Bugs
White-Box
Black-Box
Commercial Tools

- Acunetix
- N-Stalker
- Cenzic
- NtObjectives
- Fortify
- PortSwigger Web Security

Doupé - 8/10/12
Black-Box Vulnerability Scanners

Crawling

GET /index.php
Black-Box Vulnerability Scanners

Crawling

GET /view.php?id=1
Black-Box Vulnerability Scanners

Fuzzing

GET /view.php?id= <script>alert(1)</script>
The Shotgun Approach

GET /view.php?id=\n<script>alert(1)</script>
The Shotgun Approach

What if this request changed the state of the application?
Logged the user out?

GET /view.php?id=<script>alert(1)</script>
Simple Web Application

- view.php
- index.php
- login.php

Must access login.php before view.php
Internal State Graph

1. state_0 (index.php)
2. state_1 (login.php -> index.php)
3. view.php (state_1 -> view.php)

Doupé - 8/10/12
Mealy Machine

Must fuzz in different states
Inferring the State

index.php

A

login.php

B

index.php

C

view.php

D
Inferring the State

Made identical request and got different response. State has changed!
Necessary Steps to Inferring the State

• Cluster similar pages (using links)
  – Links changing means what a user can do to the application has changed

• Determine state-changing request
  – Which request in the list changed the state?

• Collapse similar states
  – How to know if, when we detect a state change, we return to a previous state?
Cluster Similar Pages
Cluster Similar Pages

- `<a, index.php, home>`
- `<a, profile.php, id=1>`
- `<form, POST, logout.php>`

- `<a, index.php, home>`
- `<a, profile.php, id=2>`
- `<form, POST, logout.php>`

- `<a, index.php, home>`
- `<a, profile.php, id=3>`
- `<form, POST, add.php>`

- `<a, index.php, home>`
- `<a, profile.php, check>`
- `<review.php, check>`
Cluster Similar Pages
Determine State-Changing Request

<table>
<thead>
<tr>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET index.php</td>
<td>A</td>
</tr>
<tr>
<td>GET blah.php</td>
<td>B</td>
</tr>
<tr>
<td><strong>POST login.php</strong></td>
<td>C</td>
</tr>
<tr>
<td>GET account.php</td>
<td>D</td>
</tr>
<tr>
<td>GET index.php</td>
<td>E</td>
</tr>
</tbody>
</table>

Use a heuristic that favors new requests over old requests, POST requests over GET requests, and requests that always change the state over those that never change the state.
Collapse Similar States

• Graph coloring
  – States as nodes
  – Edge between two states when they cannot be the same
  – Greedy coloring algorithm
Collapse Similar States

state_1

state_0 — state_1 — state_2

state_4 — state_1 — state_3
Collapse Similar States

- state_1
- state_2
- state_3
- state_4
- logged out
Collapse Similar States

logged in

logged out

state_4

state_2

state_3
Collapse Similar States

logged in

logged out

state_4

logged out

state_3
Collapse Similar States

logged in

logged out

logged out

state_4

logged in
Collapse Similar States

- logged in
- logged out
- logged out
- logged out
- logged out
- logged in
Collapse Similar States

logged out

logged in
State-Aware Fuzzing

def fuzz_state_changing(fuzz_request):
    make_request(fuzz_request)
    if state_has_changed():
        if state_is_reversible():
            make_requests_to_revert_state()
        if not back_in_previous_state():
            reset_and_put_in_previous_state()
    else:
        reset_and_put_in_previous_state()
Evaluation—Scanners

- skipfish
- w3af
- state-aware-crawler
- wget
## Evaluation—Applications

<table>
<thead>
<tr>
<th>Web Application</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gallery</td>
<td>26,622</td>
</tr>
<tr>
<td>PhpBB v2</td>
<td>16,034</td>
</tr>
<tr>
<td>PhpBB v3</td>
<td>110,186</td>
</tr>
<tr>
<td>SCARF</td>
<td>798</td>
</tr>
<tr>
<td>Vanilla Forums</td>
<td>43,880</td>
</tr>
<tr>
<td>WackoPicko v2</td>
<td>900</td>
</tr>
<tr>
<td>WordPress v2</td>
<td>17,995</td>
</tr>
<tr>
<td>WordPress v3</td>
<td>71,698</td>
</tr>
</tbody>
</table>
Code Coverage Results

Selected Applications

- Gallery
- WackoPicko v2
- WordPress v2

Percentage Code Coverage Improvement over wget

- state-aware-scanner
- w3af
- skipfish

16.2% 15.8% 11.0% 241.9% 101.2% 194.8% 14.5% 12.5% -18.3%

Doupé - 8/10/12
Best code coverage in all applications

Selected Applications

- Gallery: 16.2%, 15.8%, 11.0%
- WackoPicko v2: 14.5%, 12.5%
- WordPress v2: -18.3%

Percentage Code Coverage Improvement over wget

Doupé - 8/10/12
<table>
<thead>
<tr>
<th>Web App</th>
<th>Scanner</th>
<th>Code %</th>
<th>True Vuln</th>
<th>Unique Vuln</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhpBB v2</td>
<td>state</td>
<td>38.34</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>PhpBB v2</td>
<td>w3af</td>
<td>1.04</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PhpBB v2</td>
<td>skipfish</td>
<td>5.10</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>SCARF</td>
<td>state</td>
<td>67.03</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SCARF</td>
<td>w3af</td>
<td>55.66</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SCARF</td>
<td>skipfish</td>
<td>21.55</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vanilla</td>
<td>state</td>
<td>30.89</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vanilla</td>
<td>w3af</td>
<td>1.06</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vanilla</td>
<td>skipfish</td>
<td>-2.32</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>WackoPicko</td>
<td>state</td>
<td>241.86</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>WackoPicko</td>
<td>w3af</td>
<td>101.15</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>WackoPicko</td>
<td>skipfish</td>
<td>194.77</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
ENEMY OF THE STATE: A STATE-AWARE BLACK-BOX WEB VULNERABILITY SCANNER

Adam Doupé

Email: adoupe@cs.ucsb.edu
Twitter: @adamdoupe