EVALUATING COMMERCIAL WEB APPLICATION SECURITY

By Aaron Parke
Outline

• Project background – What and why?
  • Targeted sites
• Testing process
• Burp’s findings
• Technical talk
• My findings and thoughts
• Questions
Project Goals

• Test the security of several commercial web sites using Burp Suite
• Compare the security of these sites to one another
• Present solutions to any issues found on the sites as if hired by the companies to test their security
• Provide information about web application security in general
• Improve personal skills and understanding of web application security
What Is Web Application Security?

- Applications accessed through Web pages
  - Think online ordering, online forms, etc.
- Web applications have the potential for all kinds of security issues
  - Risks to users
  - Risks to companies
- Web application security involves testing sites for issues attackers could take advantage of
  - Typically, testing a site means seeing how you can manipulate it yourself
  - Looking at it from an attacker’s point of view
What Is Burp Suite?

• Burp Suite is a set of tools used to expose web application vulnerabilities sold by Portswigger Web Security
• Proxy, Spider, and Scanner
• Burp has a great reputation, consistently given high ratings in the industry
• Big reason I chose Burp
Targeted Sites

- Target
- Walmart
- Jet’s Pizza
- Imo’s Pizza
Target

• Large company and very large site
  • Over 2500 pages scanned
• Previous security issues
Walmart

• Similar company to Target
• Another huge company with a very large site
  • Similar number of pages scanned, around 2500
Jet’s Pizza

- Regional pizza chain
- Medium-sized company and site
- Around 900 pages scanned
Imo’s Pizza

- Similar company to Jet’s
- Smaller site
  - 670 pages scanned
Testing Process

- Burp Proxy
- Burp Spider
- Burp Scanner
- Manual confirmation
Burp Proxy

- Proxy listener
  - Local HTTP server that listens for incoming connections from browser
  - Configured to work with Firefox
- Inspects each HTTP request and response
- User must manually advance through each request
Burp Proxy

Connecting...

Burp Suite Professional v1.6.12 - licensed to Network Technology Partners [2 user license]

Target | Intruder | Repeater | Window | Help
--- | --- | --- | --- | ---

Intercept | HTTP history | WebSockets history | Options

Request to http://imospizza.com/menu [97.74.158.128]

GET /find-your-imos/ MTIP/1.1
Host: imospizza.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:37.0) Gecko/20100101 Firefox/37.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: _utmc=1669692390.027274004.1429969014.1429969014.1429969014.1429969014.1;
_utmc=1669692390
_utmc=1669692390.1429969014.1429969014.1429969014.1; __utmb=1669692390;
_c=1
Connection: keep-alive

Waiting for imospizza.com...
Burp Spider

- Basic web crawler
- Creates a map of a site by following all links and submitting forms
- Used to find all pages of a domain
  - Sent to Scanner
Burp Spider

Burp Spider needs your guidance to submit a form. Please choose the value of each form field which should be used when submitting the form. You can control how Burp handles forms in the Spider options tab.

Action URL: http://limaspizza.com/2010/12/22/
Method: GET

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>EMAIL</td>
<td>Email</td>
</tr>
<tr>
<td>Text</td>
<td>MERGE1</td>
<td>Zip Code</td>
</tr>
<tr>
<td>Text</td>
<td>FName</td>
<td>First Name</td>
</tr>
</tbody>
</table>

Submit form  Ignore form
Burp Scanner

- Checks for common vulnerabilities
  - For example, Burp will check each page for the possibility of XSS by sending a payload of random numbers and seeing how the application responds
- The Scanner gives ratings for both severity and confidence
  - Severity – Informational, Low, Medium, High
  - Confidence – Tentative, Firm, Certain
## Burp Scanner

### Scanned URLs and Results

- **URL**: Multiple URLs from `https://www.walmart.com` with various parameters and paths.
- **Status**: Some URLs marked as completed (e.g., 81% complete).
- **Issues**: Number of issues found for each URL.
- **Requests**: Total number of requests made.
- **Errors**: Number of errors encountered.
- **Insertion**: Number of insertion points identified.

### Table Example

<table>
<thead>
<tr>
<th>#</th>
<th>Host</th>
<th>URL</th>
<th>Status</th>
<th>Issues</th>
<th>Requests</th>
<th>Errors</th>
<th>Insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>2166</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/1484347473</td>
<td>finished</td>
<td>4</td>
<td>97</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2169</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/1484347473/sub...</td>
<td>finished</td>
<td>4</td>
<td>105</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2170</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/1484347473/sub...</td>
<td>finished</td>
<td>3</td>
<td>102</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2171</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/1484347473/sub...</td>
<td>finished</td>
<td>4</td>
<td>97</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2172</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/18043469734</td>
<td>finished</td>
<td>4</td>
<td>106</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2173</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/18043469734/sub...</td>
<td>finished</td>
<td>5</td>
<td>106</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2174</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/18043469734/sub...</td>
<td>finished</td>
<td>4</td>
<td>96</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2175</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/19335053865</td>
<td>finished</td>
<td>3</td>
<td>102</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2176</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/19335053865/sub...</td>
<td>finished</td>
<td>4</td>
<td>109</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2177</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/19335053865/sub...</td>
<td>finished</td>
<td>6</td>
<td>390</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>2178</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/19335053865/sub...</td>
<td>finished</td>
<td>4</td>
<td>204</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2179</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/19335053865/sub...</td>
<td>finished</td>
<td>3</td>
<td>103</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2180</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2045682573</td>
<td>finished</td>
<td>3</td>
<td>103</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2181</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2045682573/sub...</td>
<td>finished</td>
<td>4</td>
<td>103</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2182</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2142598000</td>
<td>finished</td>
<td>3</td>
<td>103</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2183</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2142598000/sub...</td>
<td>finished</td>
<td>4</td>
<td>103</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2184</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2142598000/sub...</td>
<td>finished</td>
<td>4</td>
<td>96</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2185</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2142598000/sub...</td>
<td>finished</td>
<td>4</td>
<td>96</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2186</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2142598000/sub...</td>
<td>finished</td>
<td>4</td>
<td>96</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2187</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2142598000/sub...</td>
<td>finished</td>
<td>4</td>
<td>96</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2188</td>
<td><a href="https://www.walmart.com">https://www.walmart.com</a></td>
<td>/subflow/YourAccount.loginContext/2142598000/sub...</td>
<td>finished</td>
<td>4</td>
<td>96</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

*Running (10 active threads)*
HTTP response header injection

Issue: HTTP response header injection
Severity: High
Confidence: Certain
Host: https://www.walmart.com

Issue detail
2 instances of this issue were identified, at the following locations:
- /browse/outdoor-play/swimming-pools-amp [waternslides/4171_14521_132673 parameter]
- /browse/outdoor-play/swimming-pools-amp [name of an arbitrarily supplied URL parameter]

Issue background
HTTP header injection vulnerabilities arise when user-supplied data is copied into a response header in an unsafe way. If an attacker can inject...
Manual Confirmation

• This is where the bulk of the work comes in – Burp informs users some vulnerabilities may be false positives, and the only way to find them is to test for the issues manually

• Checked for:
  • XSS
    • Reflected
    • DOM-based
  • Xpath Injection
  • LDAP Injection
  • HTTP Response Header Injection
  • Other less severe vulnerabilities
Initial Results
SQL Injection

- Open Web Application Security Project (OWASP) definition:
  - A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application.

- On a site where a user could enter their username to see their information, the SQL statement sent to the server might look like:
  - SELECT * FROM table WHERE user = 'input';

- This statement could be injected with the following input to retrieve information for all users:
  - ' OR '2'='2
SQL Injection

• Which would send this statement to the server:
  • SELECT * FROM table WHERE user = ‘’ OR ‘2’=‘2’;
• Escape characters
• Risks of SQL injection testing
• In several cases, Burp suspected SQL vulnerabilities in spots where it could be tested nonintrusively
Target SQL Injection

- Burp suspected SQL vulnerabilities through HTTP headers or parameters on 21 pages
  - Tested without the risk of dropping or modifying tables
  - Entered ‘ as value for parameter in first request and received error, “” as value the second time with no error
- Using www.hurl.it to modify and send HTTP requests, I was able to duplicate these results on 15 of the pages
- These elements are likely vulnerable to SQL injection
Walmart SQL Injection

- Burp tried a similar nonintrusive technique on some of Walmart’s parameters and cookies and suspected vulnerabilities on 658 pages
  - The advisory message read that the “two requests resulted in different responses”
- None of the reported instances were accurate
  - Entering “or true” values generated an Access Denied error, whereas Target gave a general error in completing the request
  - Entering “or false” the site behaved normally
- Could be viewed as concerning that it’s possible to exit the correct context, but it would not be possible to read any data
Cross-site Scripting (XSS)

- OWASP definition:
  - Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.
  - Basically, an attacker finds an issue that allows them to modify a page’s code, then delivers that modified code to a victim.
  - 2 types detected by Burp in scanning
    - Reflected (malicious script reaches server before returning to user)
    - DOM-based (client-side only, performance of the page changes rather than the page itself)
Walmart XSS

- Burp reported 2 instances of reflected XSS on Walmart pages
  - Inaccurate
  - “<“ sanitized to “%It”
- 2 obscure pages contained 3 possibly accurate instances of possible DOM-based XSS vulnerabilities
  - These pages read the location property from the user
Target XSS

• 245 reported instances of DOM-based XSS
  • The pages do use the location property, but it does not write the data to the page in a way that this cannot be exploited

• Also 1 reported instance of reflected XSS on the store locator page
  • Inaccurate
Jet’s XSS

• 3 instances of DOM-based XSS were reported from one of the Javascript assets
  • Just like Target, the code does uses the location property but in a safe way
Imo’s XSS

- Imo’s did have an actual instance of reflected XSS vulnerability
- When searching for a nearby store, it is possible to exit entering data into the address field and inject script into the page
  - Page sanitizes ‘ or “
  - Does not sanitize < or >
  - Wasn’t able to create custom alert, but could use variable
- I used this string in the address field to demonstrate
  - “ /> <script>alert(location)</script>
Chrome’s Response

• Chrome recognizes this as XSS. Viewing the source code shows:

```html
<form method="post" id="storefind">
  <input name="add" type="text" value="12\" />
  <script>alert(location)</script>
  <input name="city" type="text" value="" id="St_city" />
  <select name="state" id="St_state" class="text state">
    <option value="AL">ALABAMA</option>
    <option value="AK">ALASKA</option>
    <option value="AZ">ARIZONA</option>
  </select>
</form>
```
Internet Explorer’s Response
Cross-Origin Resource Sharing (CORS)

• Burp definition:
  • The HTML5 cross-origin resource sharing policy controls whether and how content running on other domains can perform two-way interaction with the domain which publishes the policy. If another domain is allowed by the policy, then that domain can potentially attack users of the application.

• Target’s gift registry pages allow for CORS
• These pages allow access to requests from arbitrary domains
  • Probably any page would have access
HTTP Response Header Injection

- Response header injection occurs when pages add user input into HTTP headers in an unsafe way.
- Similar to XSS in that an attacker modifies how the page responds by adding their own content.
- This was detected on one of Walmart’s pages.
  - The page allowed for a randomly created URL parameter to be included in the Location header.
  - Sanitized.
- In the remediation report, I moved this from a High-severity vulnerability to Low.
Open Redirection

• OWASP definition:
  • An open redirect is an application that takes a parameter and redirects a user to the parameter value without any validation. This vulnerability is used in phishing attacks to get users to visit malicious sites without realizing it.

• Imo’s was vulnerable on one of its shop pages to open redirection
  • On this page, you can enter anything you want as the value for the ReturnTo parameter, and the user will be sent there
  • Could be used in phishing attacks

• In remediation report, bumped this from a Low vulnerability to Medium
Final Results

- **Imo's**: 2 High, 1 Medium, 164 Low
- **Jet's**: 0 High, 0 Medium, 0 Low
- **Walmart**: 5 High, 330 Medium, 510 Low
- **Target**: 19 High, 0 Medium, 95 Low
Final Results – Imo’s

- Find-a-store feature vulnerable to Reflected XSS
- One shop page vulnerable to Open Redirection
- The page to login to the company’s Wordpress page submits the password in cleartext
  - An attacker monitoring network traffic could easily steal a password
Final Results – Imo’s

• While investigating the cleartext password issue, I found that besides the ordering pages, Imo’s does not allow for HTTPS connection
• Imo’s has some serious security issues, but it being the smallest company and least used website this is to be expected
Final Results – Jet’s

• Jet’s was very impressive – grand total of 1 Low vulnerability
  • It issued one cookie that didn’t have the HTTP-only flag set
  • Setting this flag can prevent scripting attacks from retrieving the cookie’s value
• Being the second smallest company, I did not expect Jet’s to have such a secure site
Final Results - Walmart

- Decent security in my opinion
  - ASP.NET tracing enabled on one page, without knowing the site’s infrastructure it’s hard to say how much this exposes
  - Possible DOM-based XSS on 3 obscure pages

- Shockingly high numbers of Low (510) and Medium (330) issues
  - Mostly due to flags not being set on HTTP and SSL cookies

- Being one of the largest companies in the world with a huge amount of online business, Walmart can be expected to have tight web security

- I would advise Walmart to look into the issues mentioned above, which are likely very easy to fix
Final Results - Target

• Target had the most High-severity vulnerabilities of any site
  • 15 pages likely vulnerable to SQL injection
  • Cross-origin resource sharing on registry pages
• Also 95 Low vulnerabilities due to cookie issues
• Except for the possibility of being exposed to SQL injection, Target’s site is fairly secure
  • Hard to say without trying to extract data if SQL really is an issue
Challenges and Final Thoughts

• Researching each vulnerability enough to understand what it was, how it was manipulated by Burp, and how to manipulate it myself was very time-consuming
  • Researching syntax for SQL, XPATH, LDAP, HTML entities, URL encoding
  • Many hours spent examining HTTP requests and responses, especially headers and parameters
• Also difficult to find ways to test some of the vulnerabilities ethically
  • SQL frustration
• Worked to understand what were at times some pretty huge and complex HTML files
Challenges and Final Thoughts

- Thoughts on Burp:
  - Hyper-sensitive
  - Needs manual confirmation
QUESTIONS?