



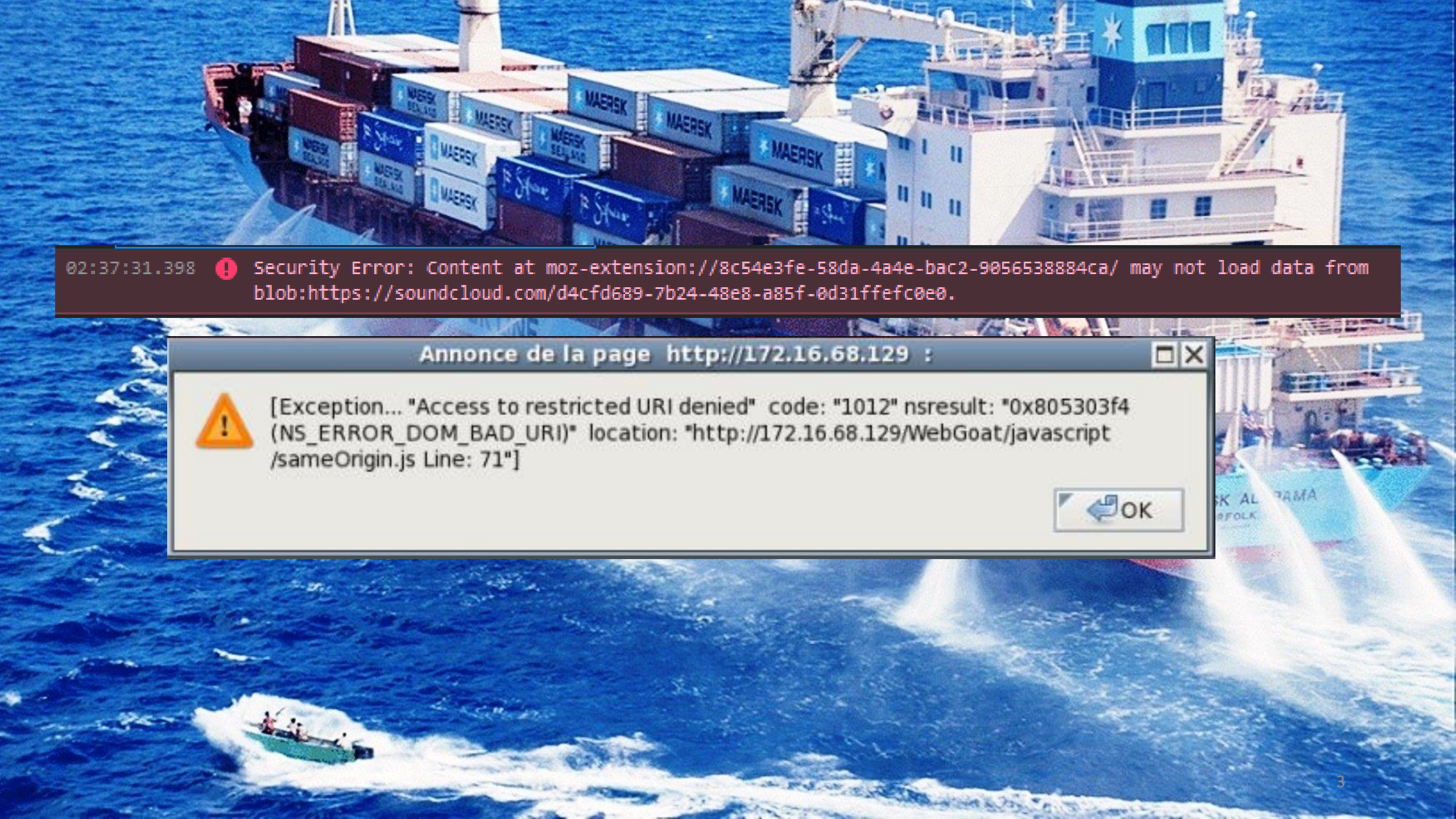
API Security Bootcamp Hands-On OWASP Top 10 for APIs


Dr. Sunny Wear

2023

Background: CORS

Cross Origin Resource Sharing (CORS)



02:37:31.398  Security Error: Content at moz-extension://8c54e3fe-58da-4a4e-bac2-9056538884ca/ may not load data from blob:https://soundcloud.com/d4cfd689-7b24-48e8-a85f-0d31ffefc0e0.

Annonce de la page <http://172.16.68.129> :



[Exception... "Access to restricted URI denied" code: "1012" nsresult: "0x805303f4 (NS_ERROR_DOM_BAD_URI)" location: "http://172.16.68.129/WebGoat/javascript/sameOrigin.js Line: 71"]

 OK

Pre-CORS Support

Browsers would allow the requesting of cross-origin resources (e.g., remote JavaScript files) but would prohibit the reading of the response

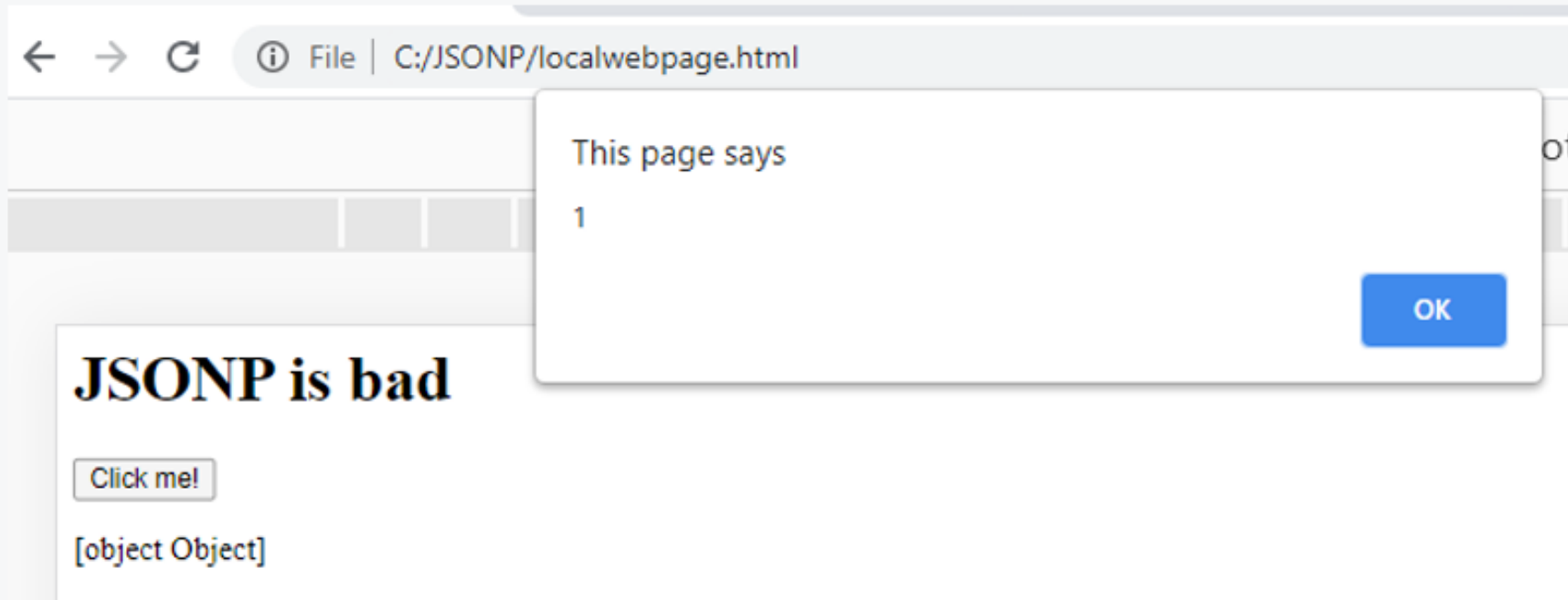
By not allowing the reading of the raw response from a different origin, applications could not execute scripts, see images, etc.

Same Origin Policy was the browser protection stopping the reading of the cross-origin response

Insecure Pre-CORS Workarounds

JSONP

```
s.src = "https://demo.sjoerdlangkemper.nl/jsonp.php?callback=alert(1);myfunc";
```



Cross- Origin Resource Sharing (CORS)

Devs can use CORS to ensure they really want to process a request

Key CORS Headers:

- **Origin** – automatically populated by the browser informing API where the cross-origin request is coming from; Question to the API: Is the request allowed? Yes or No.
- **Access-Control-Allow-Origin** – response from API providing authorization to a list of whitelisted origins (hopefully). This header is the API way stating to the Origin “You are allowed to share my response with that origin”.

New Issue: CORS Misconfigurations Abound!

CORS Misconfigurations and Findings



Overly Permissive CORS Policy

Wildcard “*” for authenticated calls



Arbitrary Origin Trusted

Value of Origin provided is reflected in response
Higher severity especially with “Access-Control-Allow-Credentials: true”



null origin

When no origin is present such as a file on your computer or a CORS redirect, the browser assigns ‘null’.



Access- Control- Allow- Credentials: true

- By default, CORS does **not** include cookies on cross-origin requests.
- This behavior is for protection against CSRF attacks.
- So, the ACAC:true is an explicit allowance of the cookies included with the CORS call.

OkCupid

Personal Data

Name

Home Address

Business Address

Identity Card No

Passport No

Driving License

Income Tax No

Car Registration



Confidential Data



OkCupid Dating App

<https://www.youtube.com/watch?v=m-ZhP1x7pTQ&t=70s>

Cyber Attack

6564207368

06E61C

F765 6C792

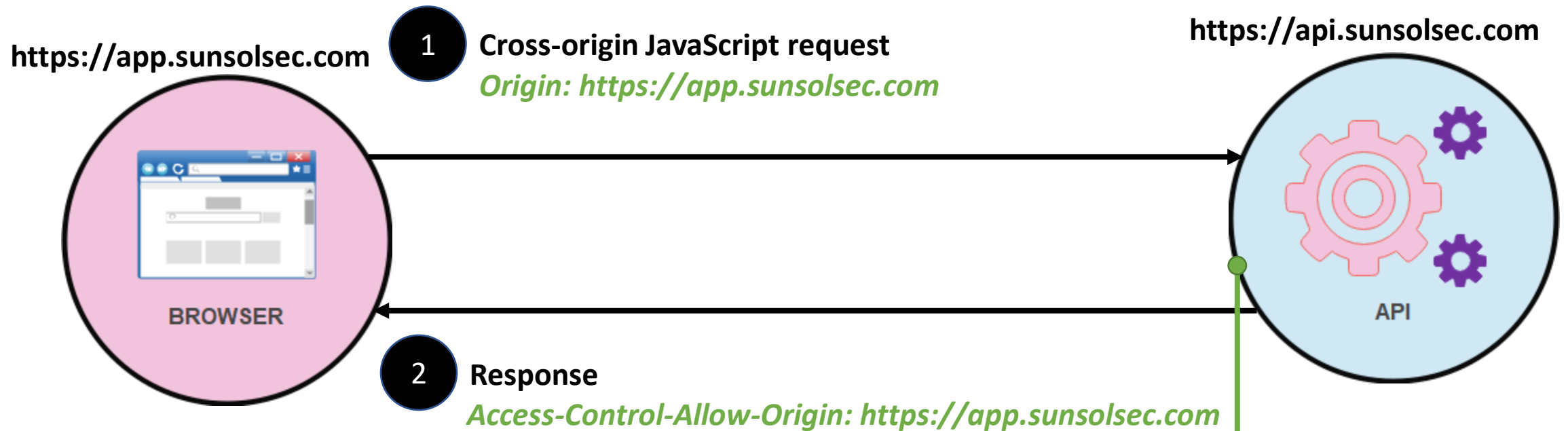
Protection

```
GET /1/profile/13882231300401639888?access_token=
1,0,1588499251,13882231300401639888;af3abb3f361d118f18e7237cf1605566930d4578
HTTP/1.1
Host: api.okcupid.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/81.0.4044.129
Origin: http://okcupidmeethehacker.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,he;q=0.8,sv;q=0.7
Cookie: __cfduid=d65f0a69ad66610c5ebb35dae69aa28371588263867; __ssid=
65834da3847e2160ca62a4e76299f01;
ab.storage.deviceId.719f8d59-40d7-4abf-b9c3-fa4bf5b7cf54=
%7B%22g%22%3A%22c9dddc5-965a-e3ce-clad-ecd97284f04a%22%2C%22c%22%3A1588263869C
89%2C%22l%22%3A1588263869089%7D; OptanonAlertBoxClosed=2020-04-30T16:24:35.8832
; eupubconsent=
BOyrdWmOyrdWmAcABBENDH-AAAAvR7____9____9uz_Ov_v_f_33e8_9v_1_7_-_u_-33d
4u_lvf99yfml-7etr3tp_87ues2_Xur__71__3z3_9pxP78k89r7337Ew_v-_v-b7BCPN9Y3v-8KA;
_ga=GA1.2.1262270496.1588263876; _gid=GA1.2.2085818024.1588263876; ua=
531227642bc86f3b5fd7103a0c0b4fd6;
ab.storage.userId.719f8d59-40d7-4abf-b9c3-fa4bf5b7cf54=
%7B%22g%22%3A%2213882231300401639888%22%2C%22c%22%3A1588347115595%2C%22l%22%3A1
588347115595%7D; secure_check=1; authlink=9129ad3c; _parsely_visitor=
(%22id%22%22pid=1293a0d7c4cd9d6fd0a75462e8aaab3d%22%2C%22session_count%22%1%2C
%22last_session_ts%22%1588348102701); override_session=0; ajs_group_id=null;
ajs_anonymous_id=%22e364d7b0-f0e8-4dc3-a29a-559eab93a4e9%22;
fbm_484681304938818=base_domain=.okcupid.com; siftsession=14881627732244711205;
secure_login=1; session=13882231300401639888%3a12045434044026330569;
OptanonConsent=
isIABGlobal=false&datestamp=Sat+May+02+2020+13%3A04%3A17+GMT%2B0300+(Israel+Day
light+Time) &version=5.15.0&landingPath=NotLandingPage&groups=1%3A1%2C2%3A1%2C3%
3A1%2C4%3A1&hosts=&legInt=&consentId=7f5a0f3d-0828-4f12-8ea5-5c11976474f7&inter
actionCount=1&geolocation=IL%3BTA&AwaitingReconsent=false;
ab.storage.sessionId.719f8d59-40d7-4abf-b9c3-fa4bf5b7cf54=
%7B%22g%22%3A%226700da82-d44c-32e5-e923-aea22387013c%22%2C%22e%22%3A1588415657S
75%22%2C%22l%22%3A1588413820381%2C%221%22%3A1588413857975%7D;
ampitude_id_f90f10ee203f3f5f1e142ad0a976ee20_desktopokcupid.com=
eyJkZXZpY2VJZCI6ImYwMHRhZWMxLWwNGItNDYjMCO5NDQwLWNhNjEYnN2U4MmZiYVVIiLCJlc2VySWQ
iOiIxMzg4MjIzMTRhZWMwMTYzOTg4OCIsIm9wdE91dCI6ZmFsc2UsInNlc3Npb25JZCI6MTU4ODQxMz
qyMDU5MiwibGFzdEVZ2ZW50VGltZSI6MTU4ODQxMzq2MDMzNywiZXZlbnRJZCI6MjEjYmJlc2VudGlm
```

```
HTTP/1.1 200 OK
Date: Sat, 02 May 2020 10:25:52 GMT
Content-Type: application/json
Connection: keep-alive
CF-Ray: 58d10a6bc85dad91-TLV
Access-Control-Allow-Origin: http://okcupidmeethehacker.com/
Cache-Control: private
CF-Cache-Status: DYNAMIC
access-control-allow-credentials: true
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com"
x-okws-version: PubExpress 1.0 (on OKWS)
Vary: Accept-Encoding
Server: cloudflare
cf-request-id: 027684d75f0000ad9102bac2000000001
Content-Length: 27113
```

<https://research.checkpoint.com/2020/hacker-22-seeks-ltr-with-your-data-vulnerabilities-found-on-popular-okcupid-dating-app/>

CROSS-ORIGIN RESOURCE SHARING (CORS)

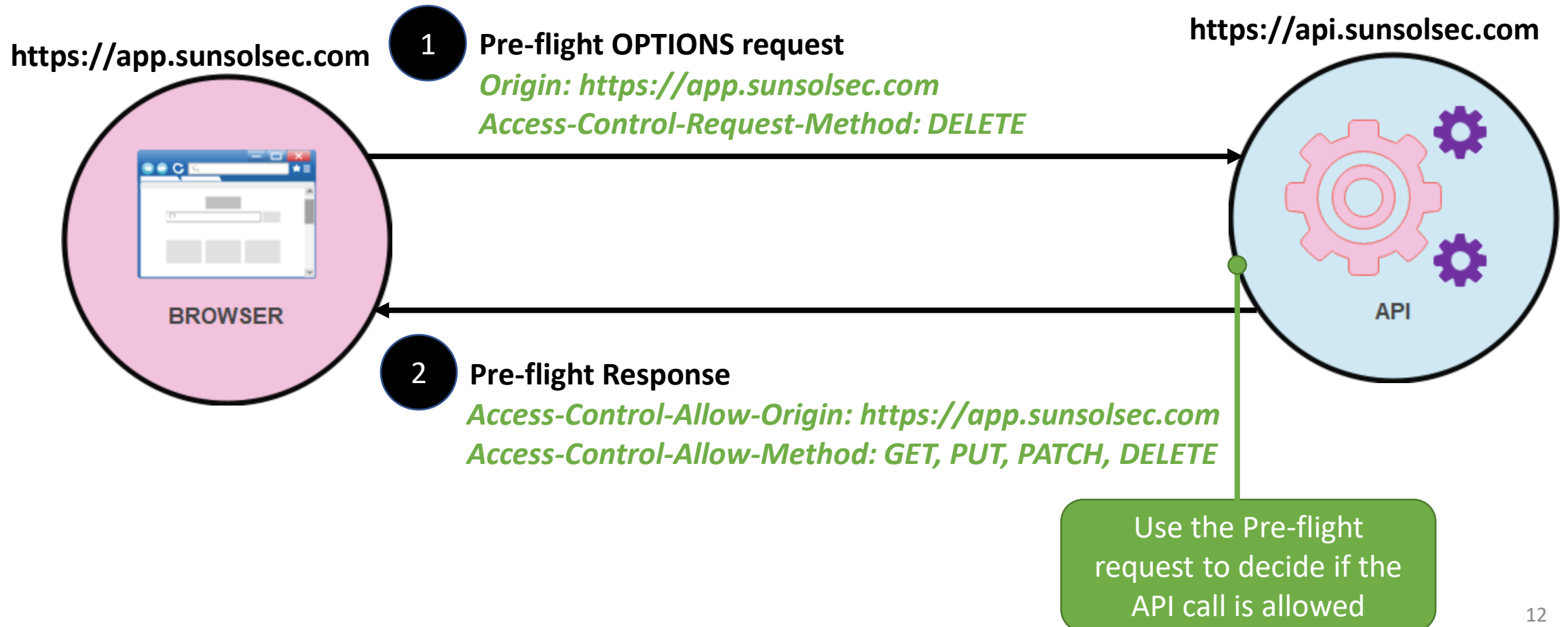


JavaScript code to send the Fetch request

```
1 fetch('https://api.sunsolsec.com/getProducts')
2 .then(response => response.json())
3 .then(data => console.log(data));
```

Use the Origin header to decide if this origin is allowed to call the API

PREFLIGHT CORS CALL



Exercise 6-1: CORS vuln with basic origin reflection

Lab: CORS vulnerability with basic origin reflection



APPRENTICE

LAB

✓ Solved

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`

Facebook null origin attack

A security researcher has discovered a critical vulnerability in Facebook Messenger that could allow an attacker to read all your private conversation, affecting the privacy of around 1 Billion Messenger users.

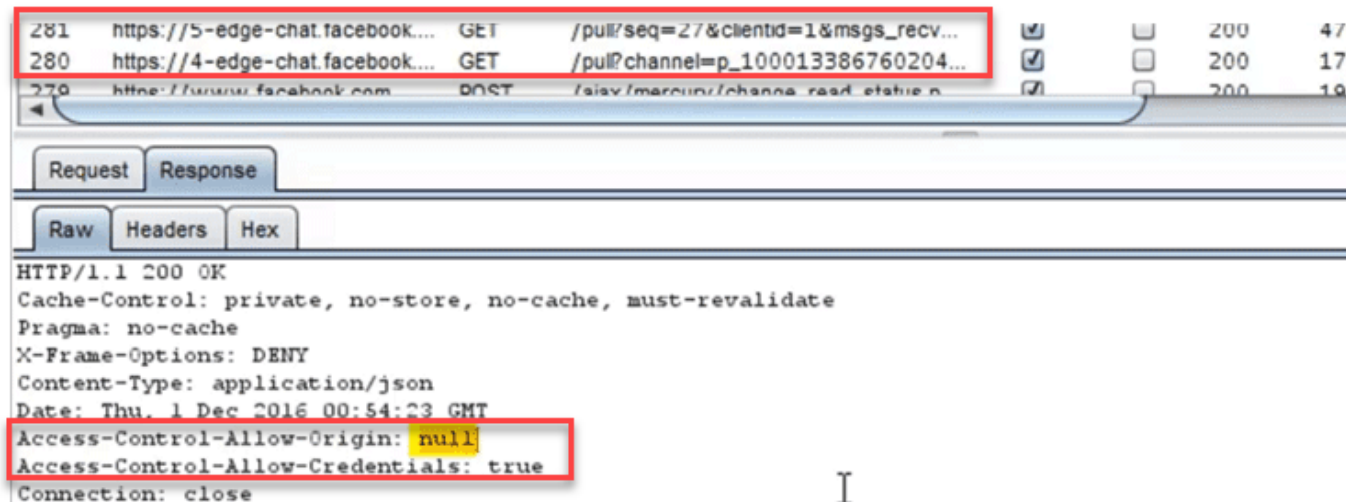
Ysrael Gurt, the security researcher at [BugSec](#) and Cynet, [reported](#) a cross-origin bypass-attack against Facebook Messenger which allows an attacker to access your private messages, photos as well as attachments sent on the Facebook chat.

2016

Dubbed "Originull," the vulnerability actually lies in the fact that Facebook chats are managed from a server located at {number}-edge-chat.facebook.com, which is separate from Facebook's actual domain (www.facebook.com).

"Communication between the JavaScript and the server is done by XML HTTP Request (XHR). In order to access the data that arrives from 5-edge-chat.facebook.com in JavaScript, Facebook must add the "Access-Control-Allow-Origin" header with the caller's origin, and the "Access-Control-Allow-Credentials" header with "true" value, so that the data is accessible even when the cookies are sent," Gurt explained.

The root of this issue was misconfigured cross-origin header implementation on Facebook's chat server domain, which allowed an attacker to bypass origin checks and access Facebook messages from an external website.



Facebook null origin attack

<https://fossbytes.com/facebook-messenger-chats-hack-originull/>



Exercise 6-2: CORS with null Attack

MUST ADD "https://" in front of URLs inside of iframe script.

Lab: CORS vulnerability with trusted null origin



APPRENTICE

LAB

Solved



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`

[Access the lab](#)