Apr
il
202
1

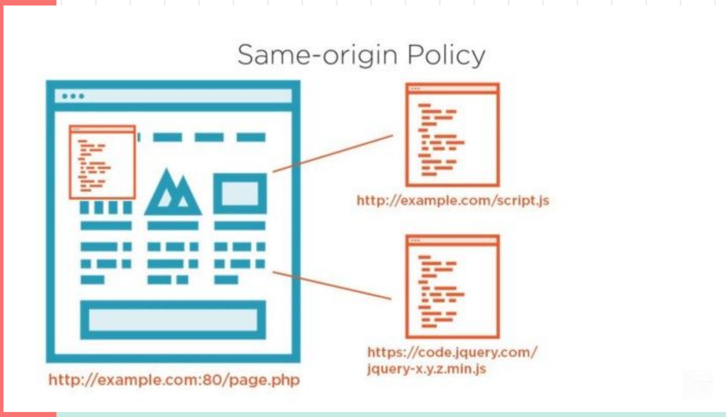
APRIL Hacking HTTP CORS

A theory to practice approach

Minh Tuấn - Cyber Security Research Team

TABLE OF CONTENTS

FEB
202
1

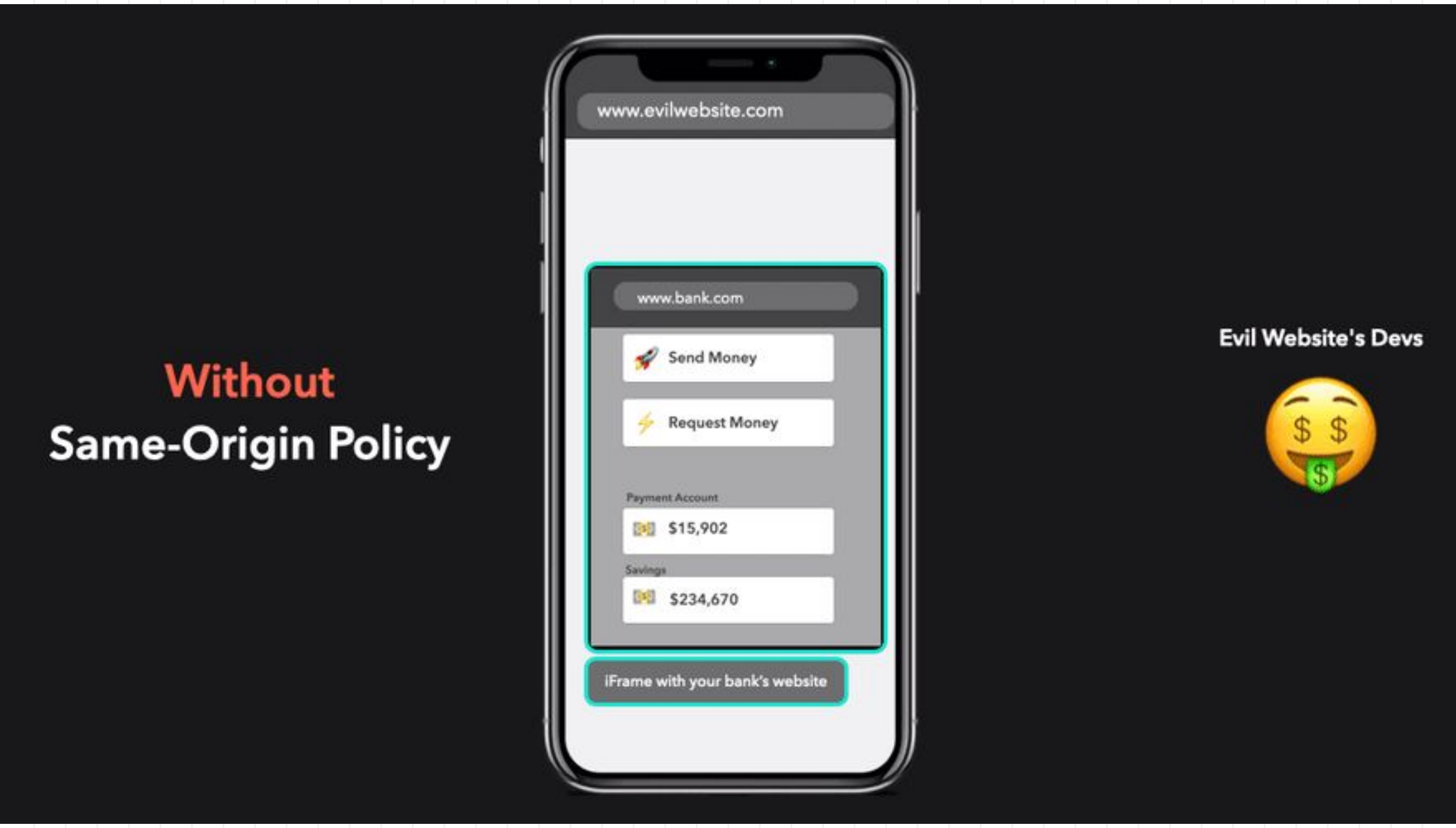


Same Origin Policy

Same Origin Policy

In computing, the same-origin policy (sometimes abbreviated as SOP) is an important concept in the web application security model. Under the policy, a web browser permits scripts contained in a first web page to access data in a second web page, but only if both web pages have the same origin. An origin is defined as a combination of URI scheme, host name, and port number. This policy prevents a malicious script on one page from obtaining access to sensitive data on another web page through that page's Document Object Model.

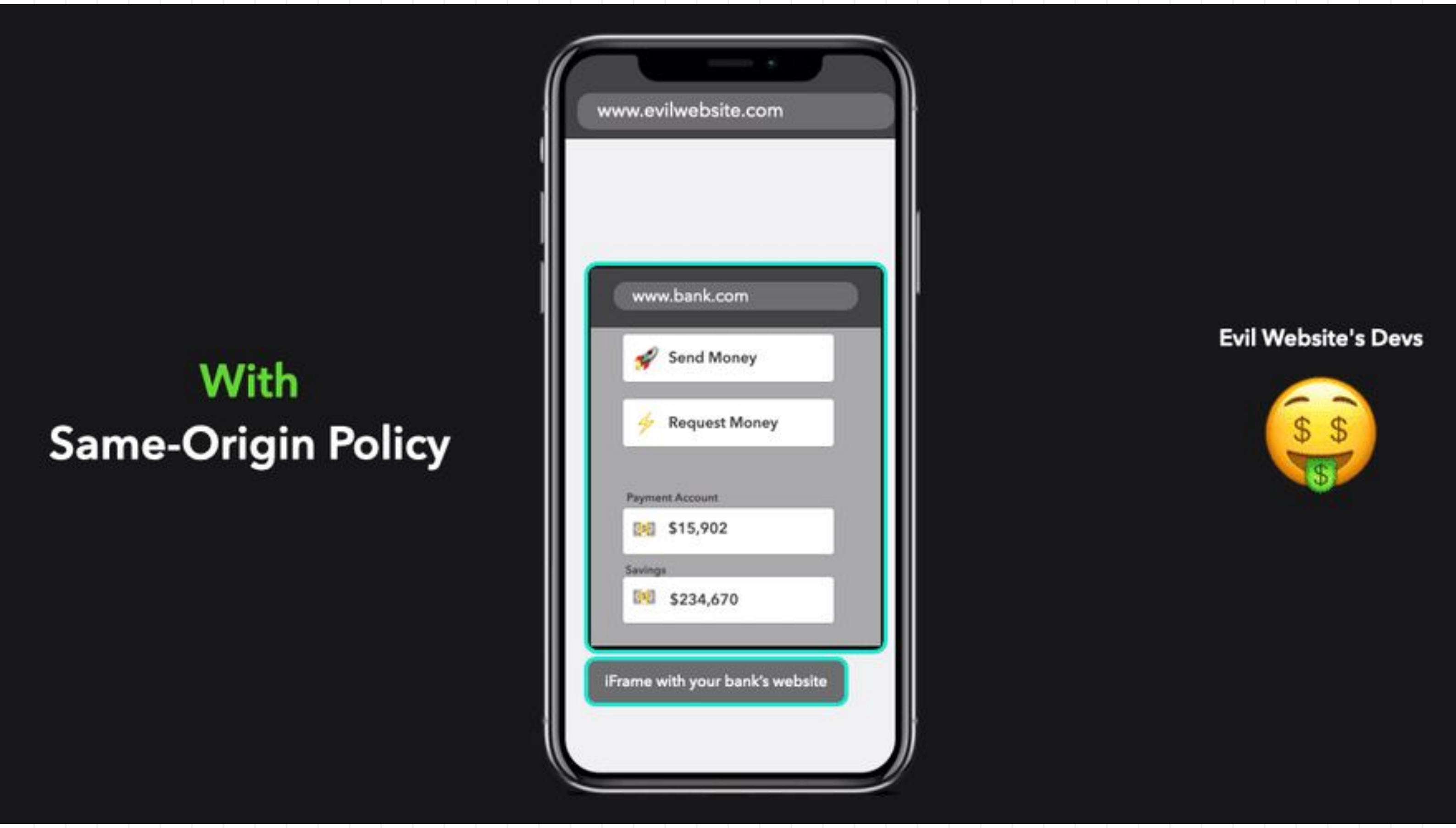
Same Origin Policy



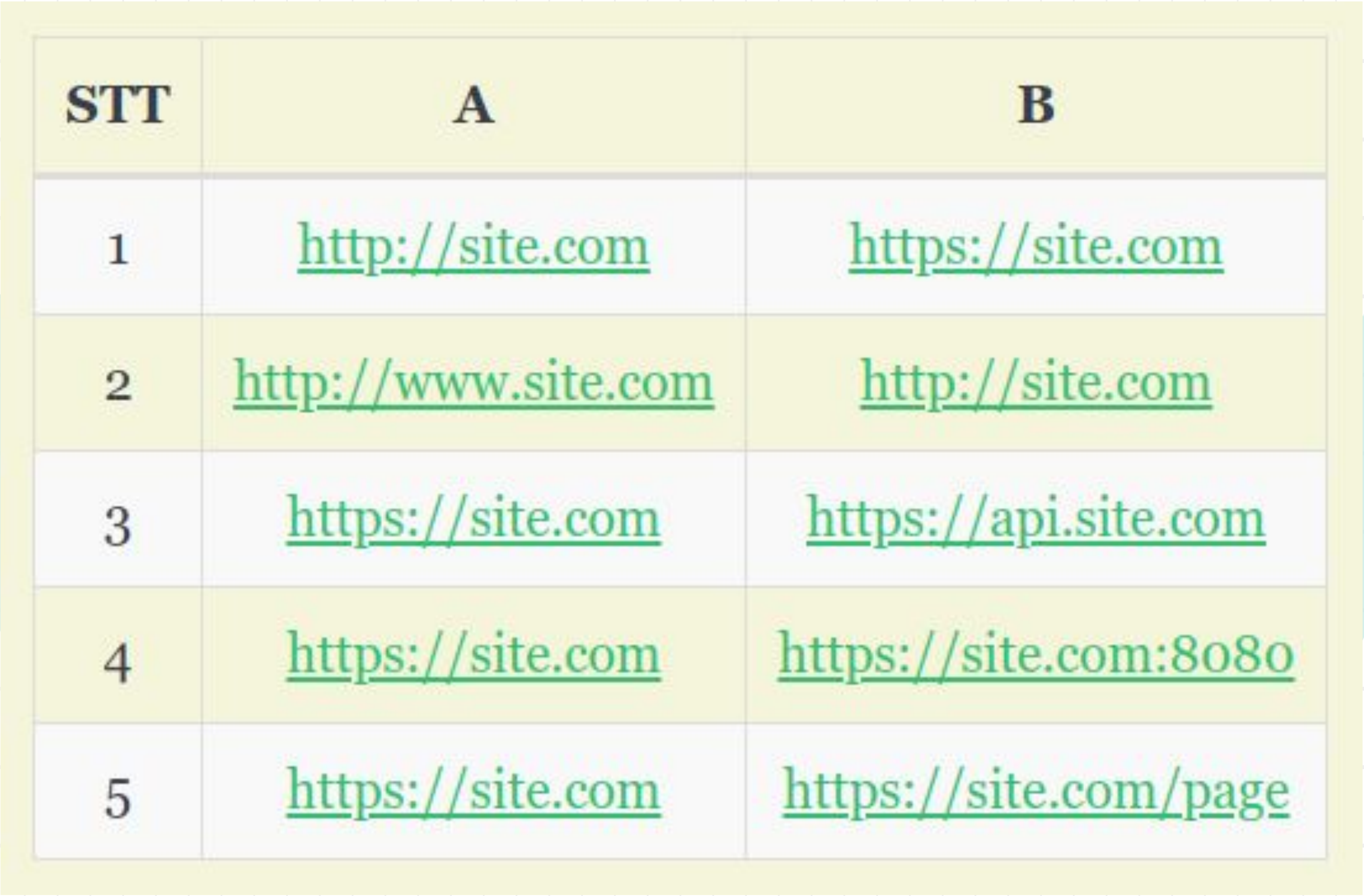
Without Same-Origin Policy

Evil Website's Devs

Same Origin Policy

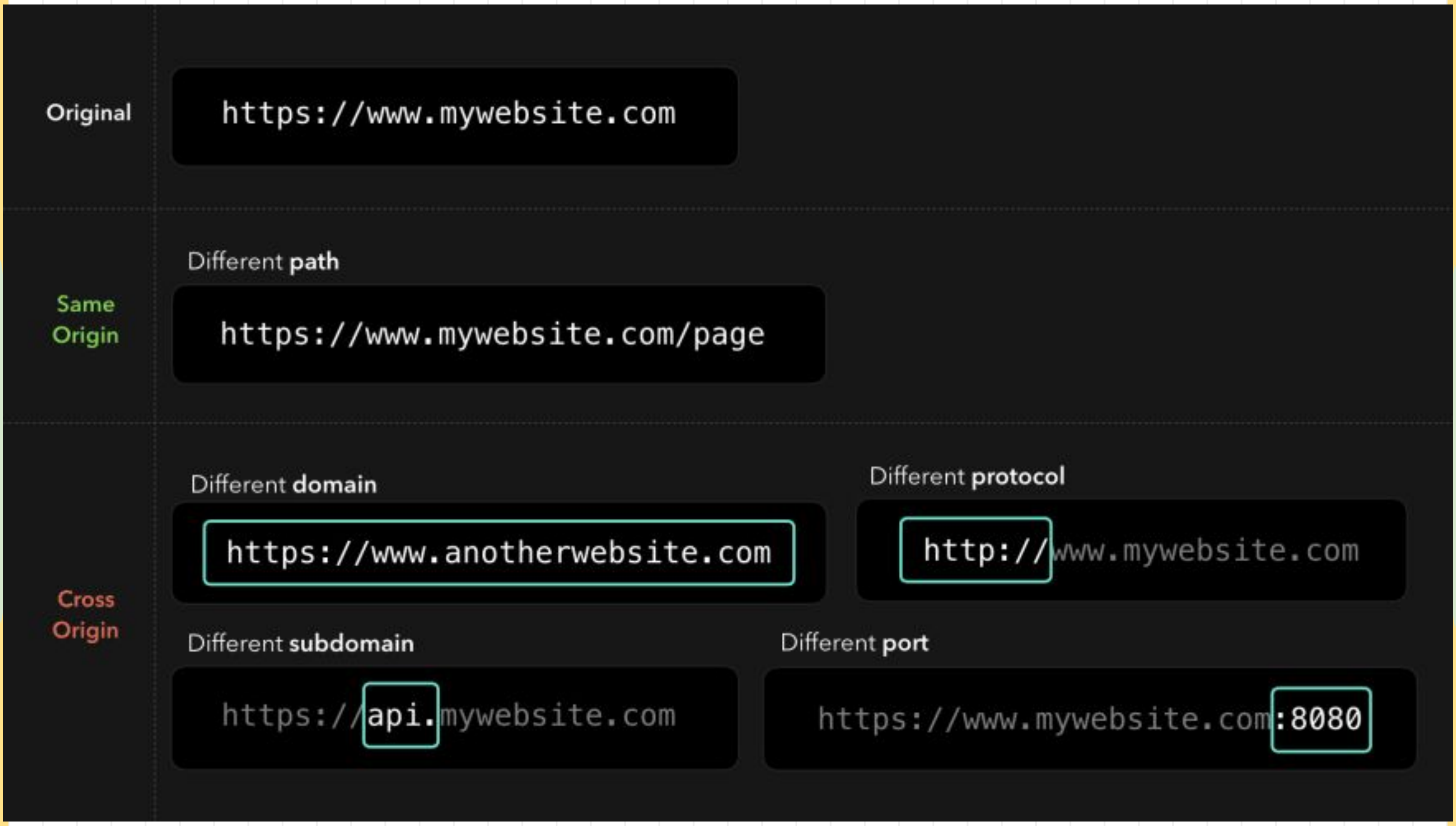


Same Origin Policy



STT	A	B
1	http://site.com	https://site.com
2	http://www.site.com	http://site.com
3	https://site.com	https://api.site.com
4	https://site.com	https://site.com:8080
5	https://site.com	https://site.com/page

Same Origin Policy



Original

https://www.mywebsite.com

Same Origin

Different **path**

https://www.mywebsite.com/page

Cross Origin

Different **domain**

https://www.anotherwebsite.com

Different **protocol**

http://www.mywebsite.com

Different **subdomain**

https://api.mywebsite.com

Different **port**

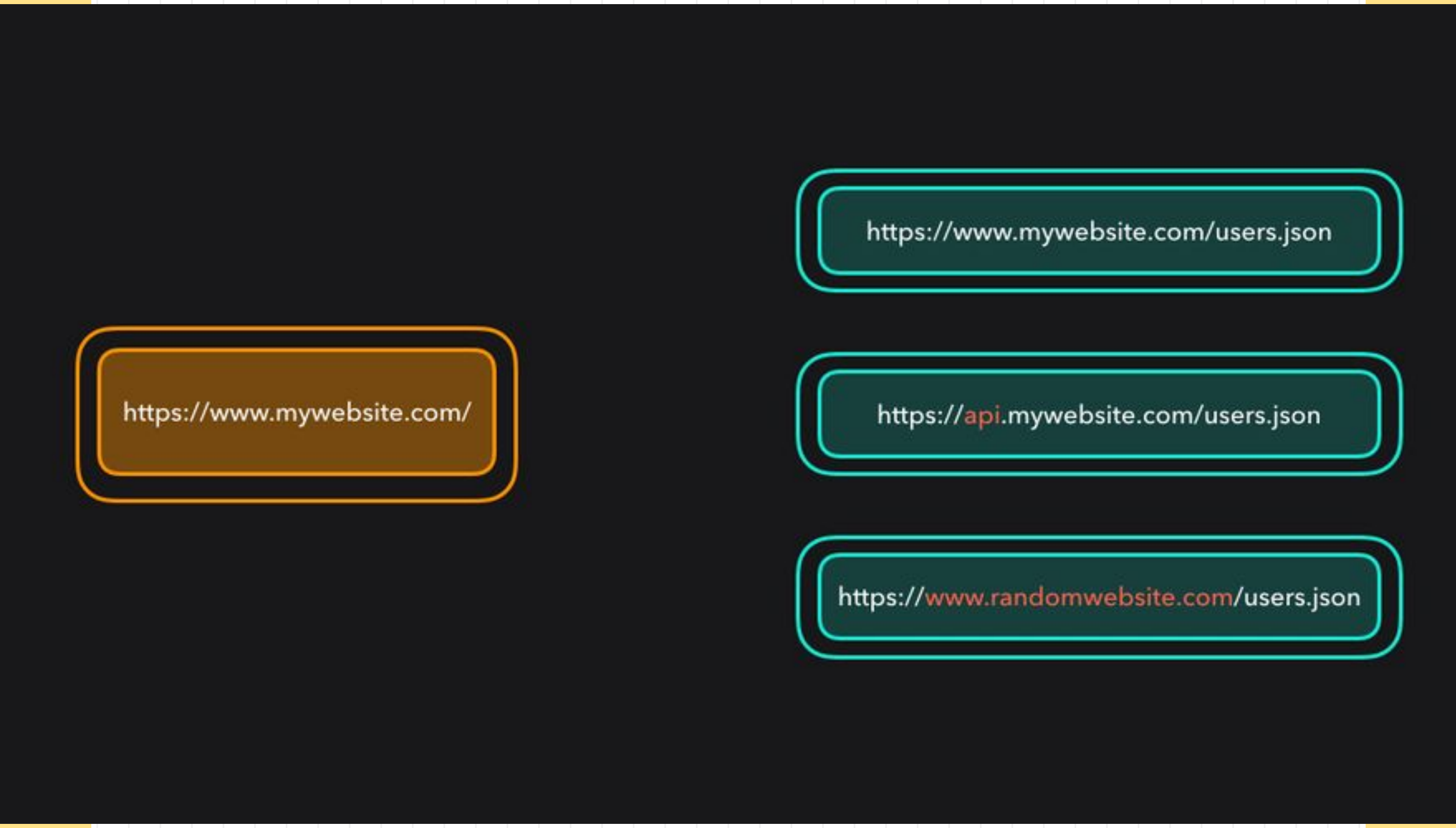
https://www.mywebsite.com:8080

HTTP CORS

Why must?

CORS

Pitfalls & Security

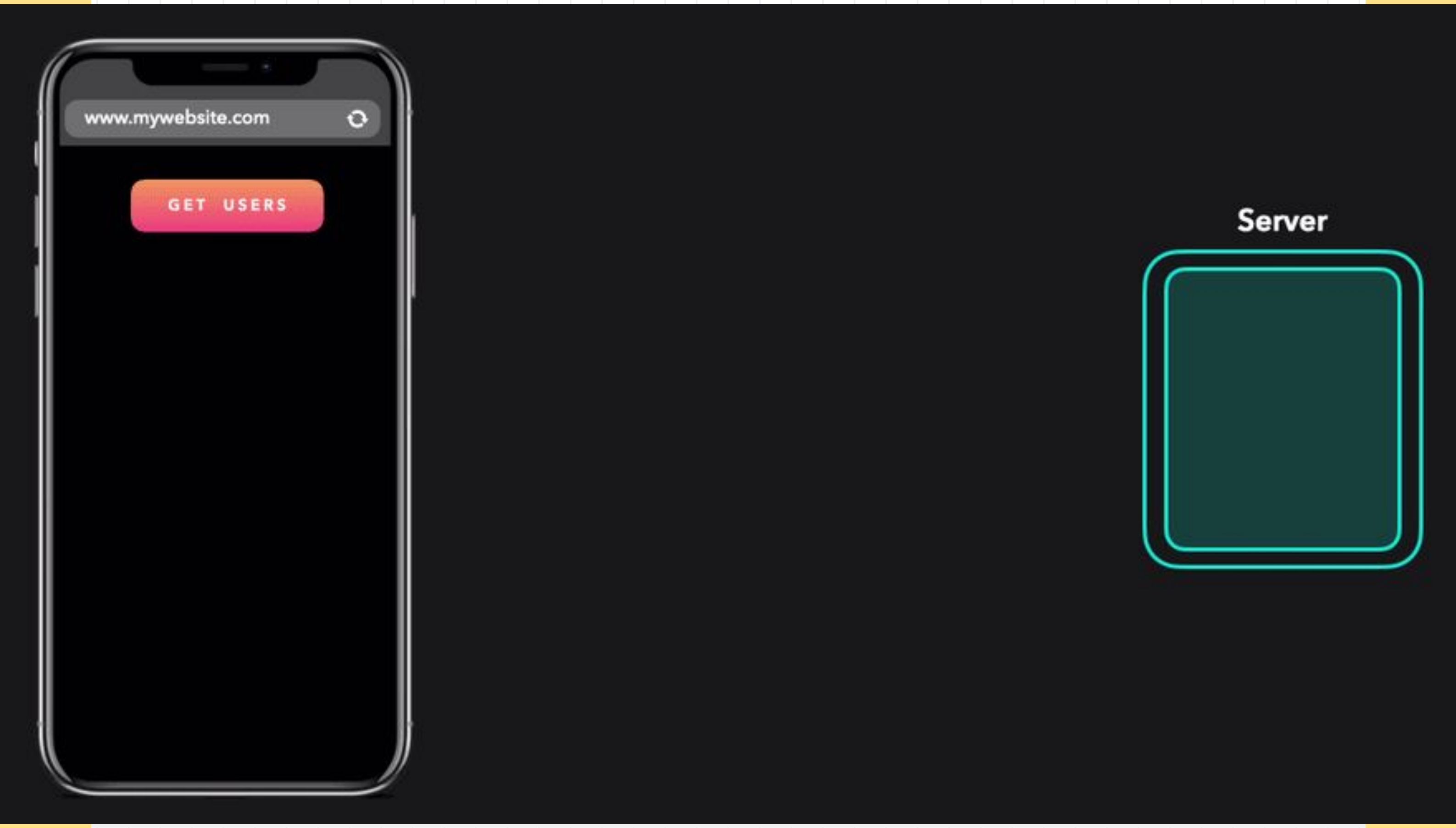


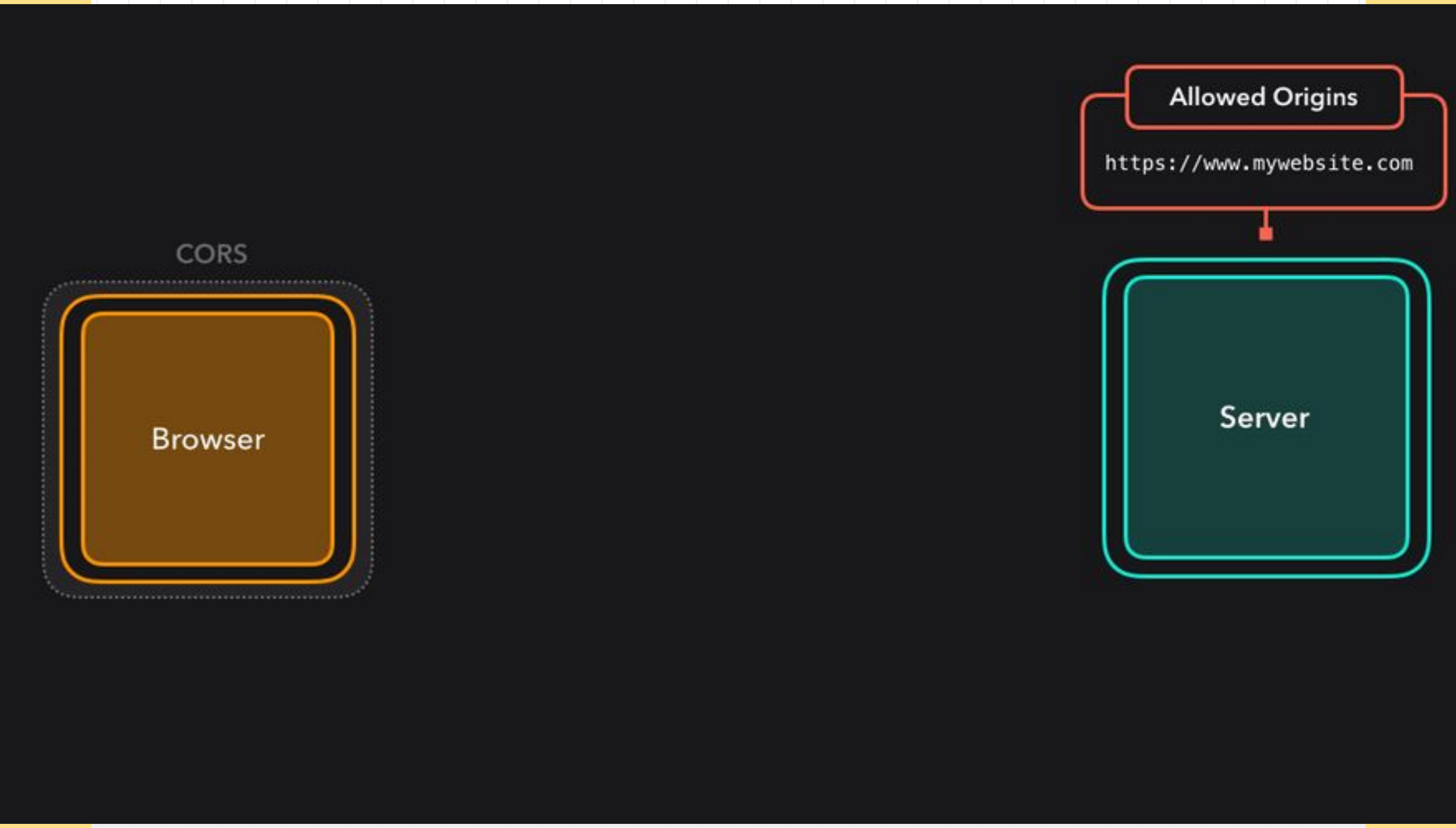
https://www.mywebsite.com/

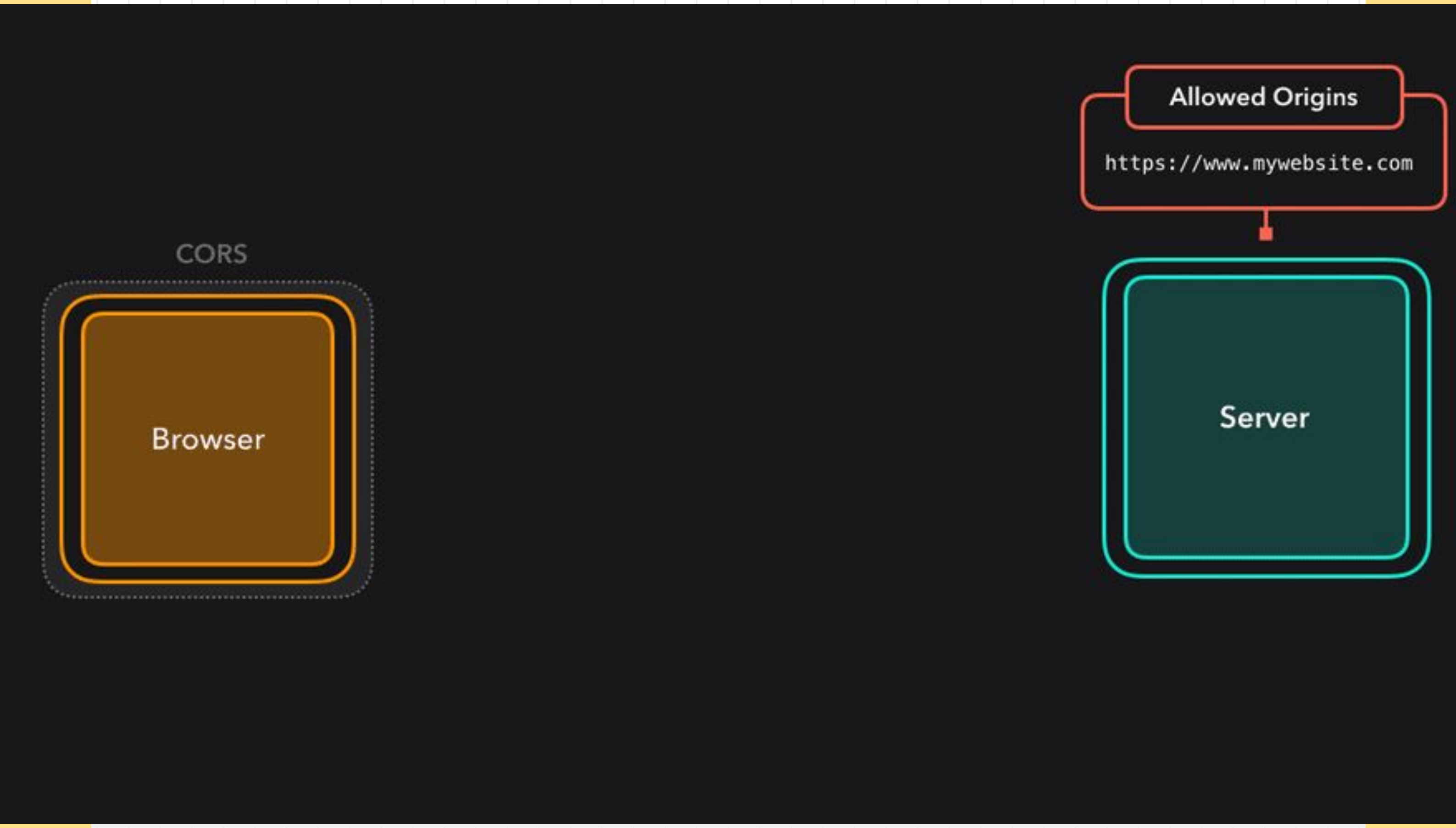
https://www.mywebsite.com/users.json

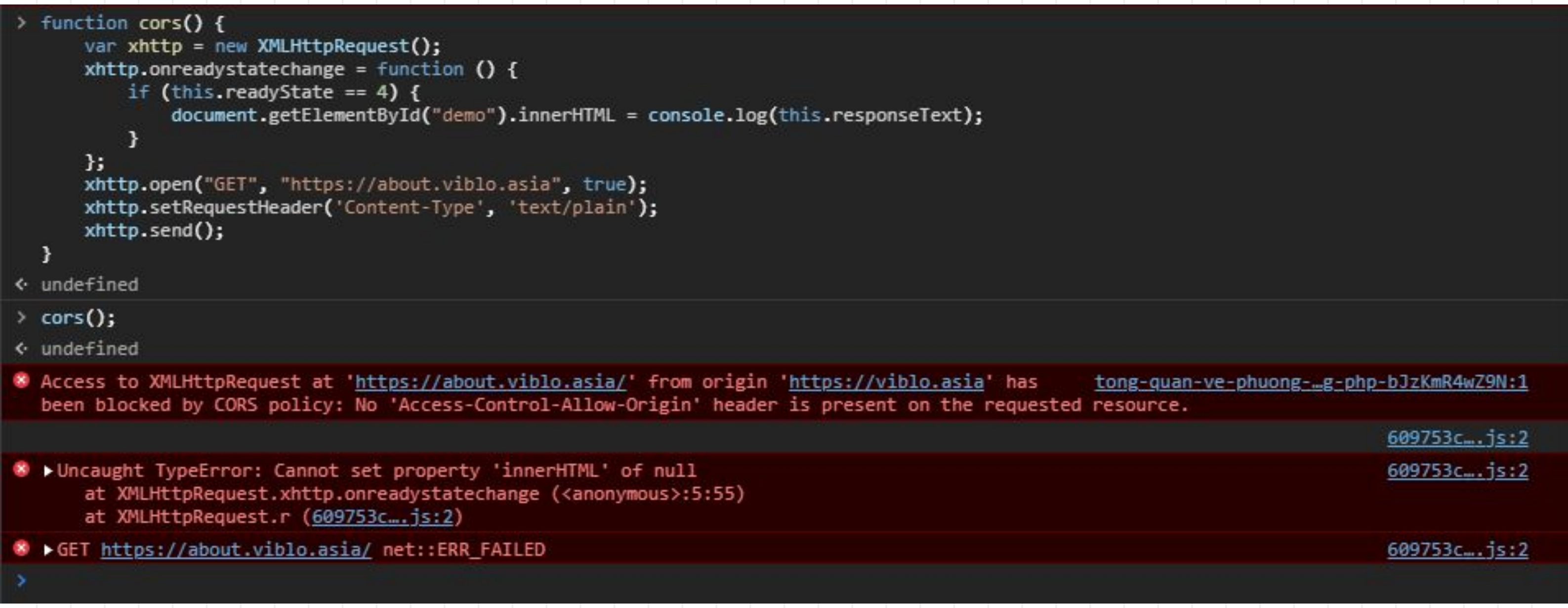
https://api.mywebsite.com/users.json

https://www.randomwebsite.com/users.json


www.mywebsite.com
GET USERS
Server


Allowed Origins
https://www.mywebsite.com
CORS
Browser
Server

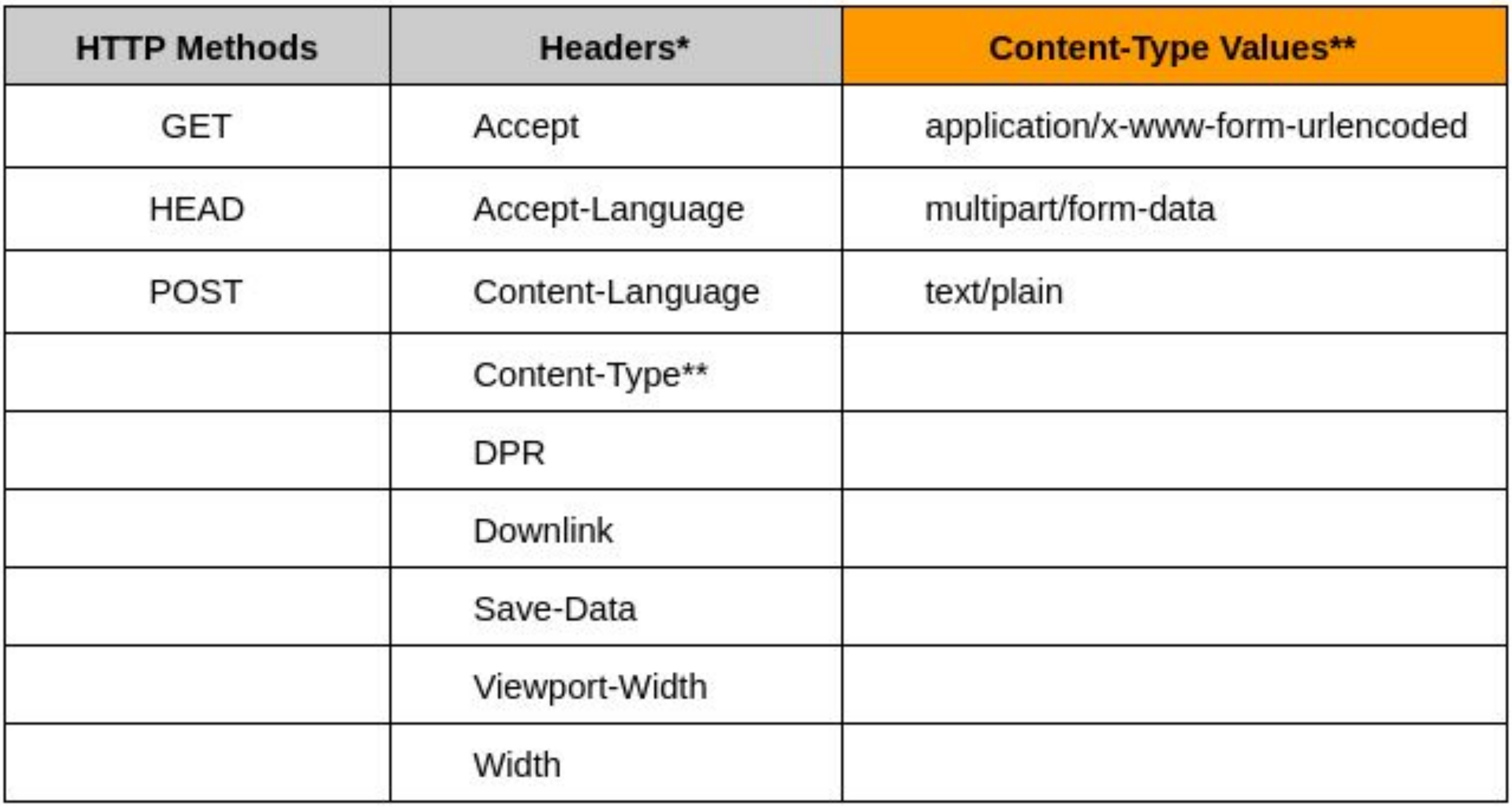

Allowed Origins
https://www.mywebsite.com
CORS
Browser
Server



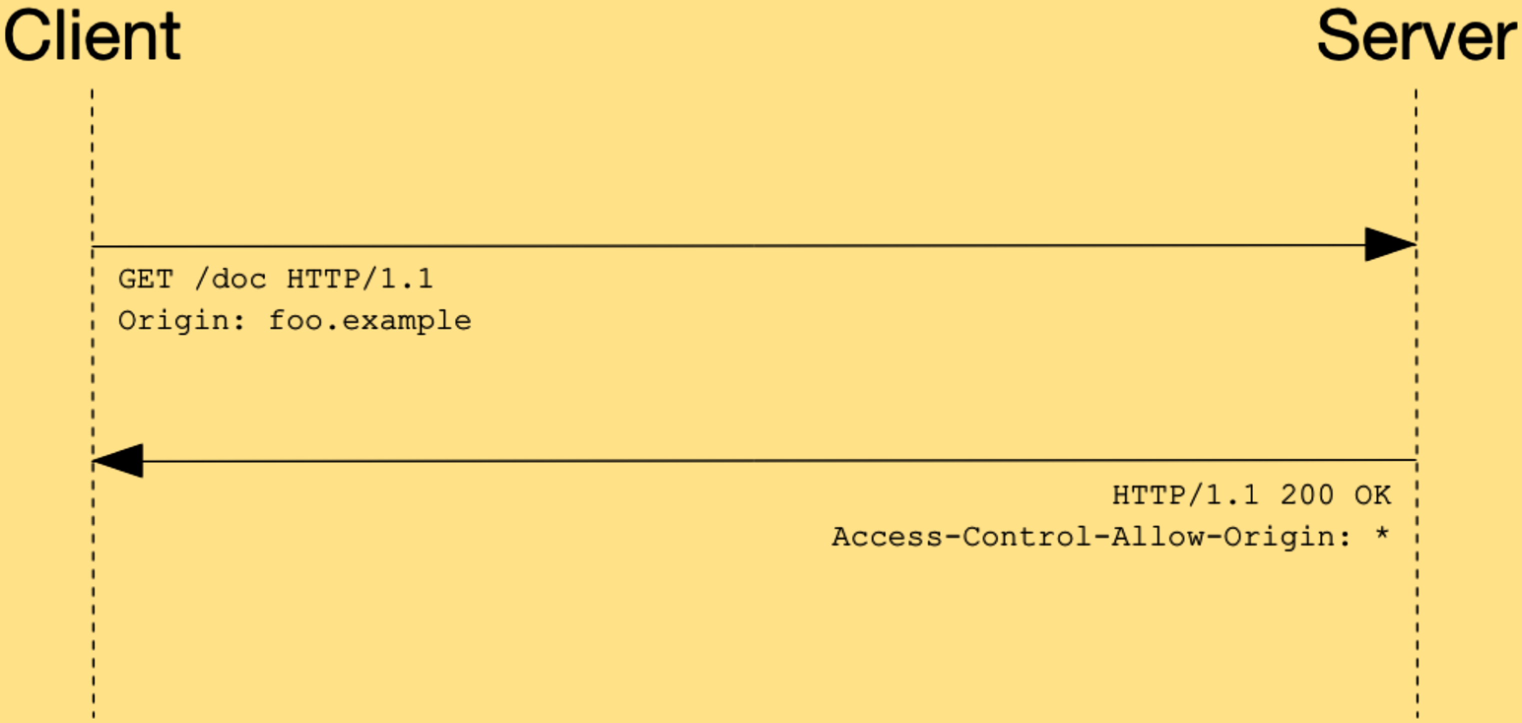
```
> function cors() {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function () {
          if (this.readyState == 4) {
              document.getElementById("demo").innerHTML = console.log(this.responseText);
          }
      };
      xhttp.open("GET", "https://about.viblo.asia", true);
      xhttp.setRequestHeader('Content-Type', 'text/plain');
      xhttp.send();
  }
<· undefined
> cors();
<· undefined
✖ Access to XMLHttpRequest at 'https://about.viblo.asia/' from origin 'https://viblo.asia' has      tong-quan-ve-phuong-…g-php-bJzKmR4wZ9N:1
  been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
                                                                                                              609753c….js:2
✖ ▶Uncaught TypeError: Cannot set property 'innerHTML' of null                                             609753c….js:2
      at XMLHttpRequest.xhttp.onreadystatechange (<anonymous>:5:55)
      at XMLHttpRequest.r (609753c….js:2)
✖ ▶GET https://about.viblo.asia/ net::ERR_FAILED                                                          609753c….js:2
>
```

Note that the browser fully trusts the response returned from the server. Keep this in mind.

SIMPLE REQUEST



HTTP Methods	Headers*	Content-Type Values**
GET	Accept	application/x-www-form-urlencoded
HEAD	Accept-Language	multipart/form-data
POST	Content-Language	text/plain
	Content-Type**	
	DPR	
	Downlink	
	Save-Data	
	Viewport-Width	
	Width	

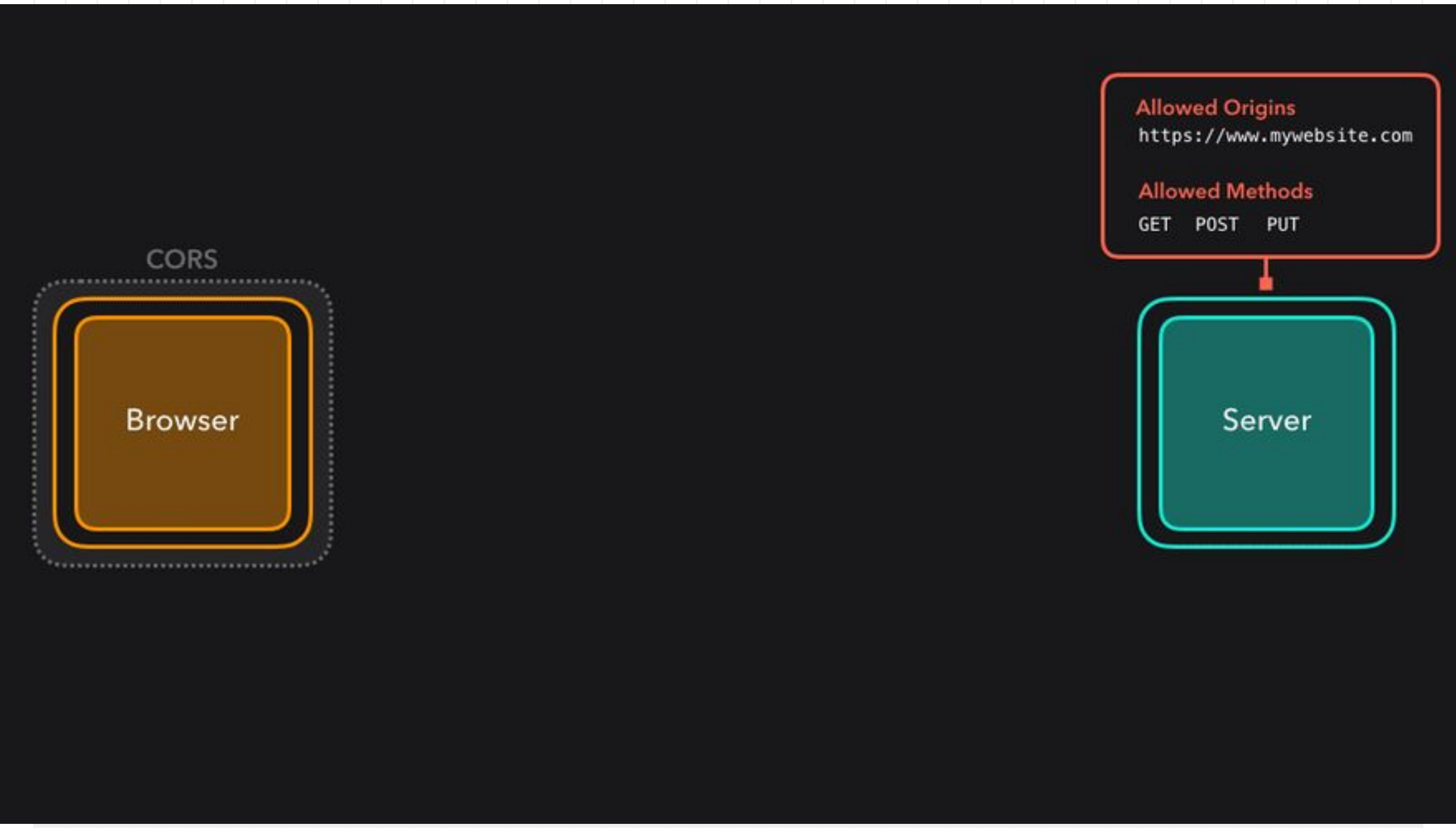


```
GET /doc HTTP/1.1
Origin: foo.example
```

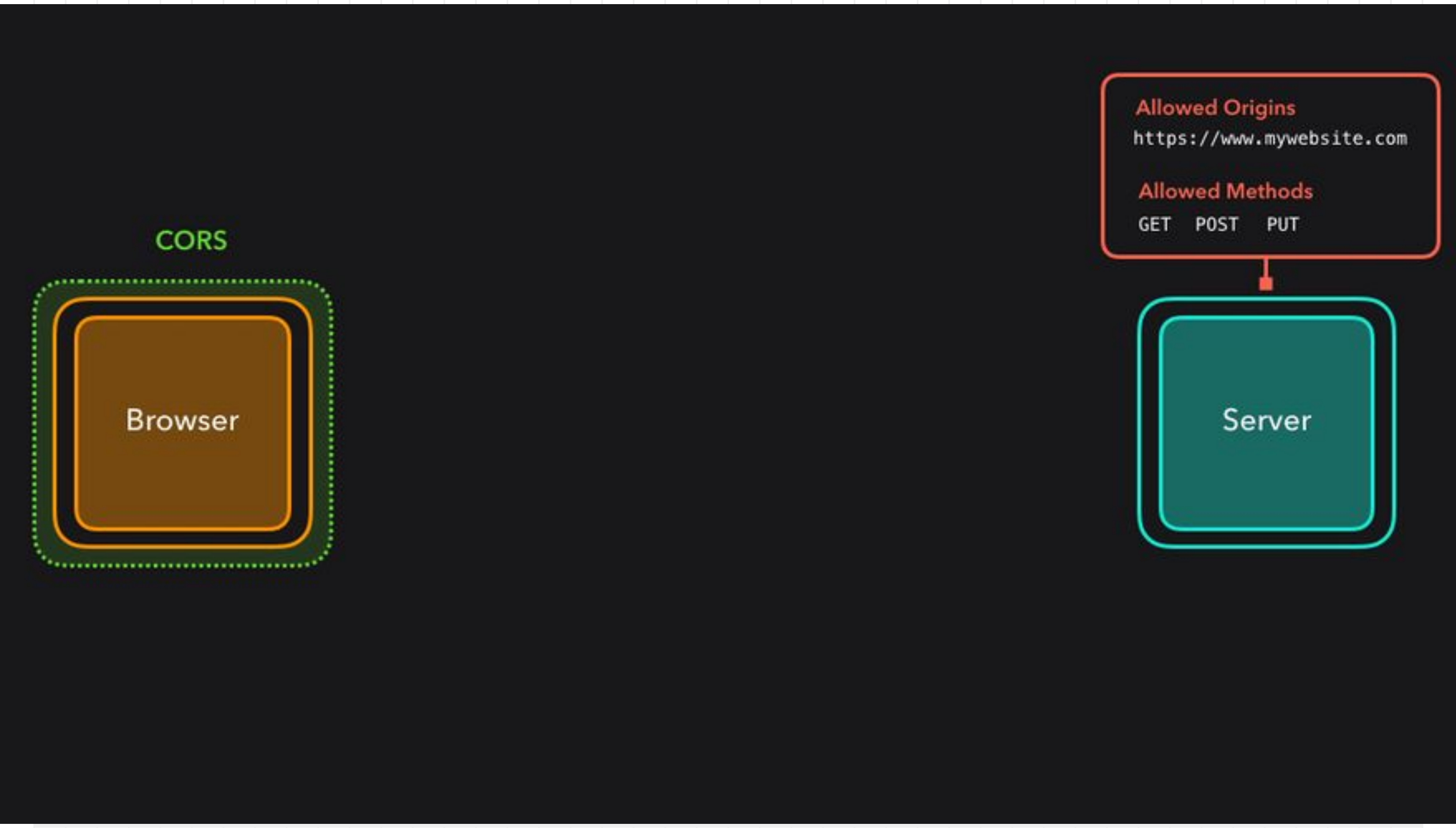
```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
```

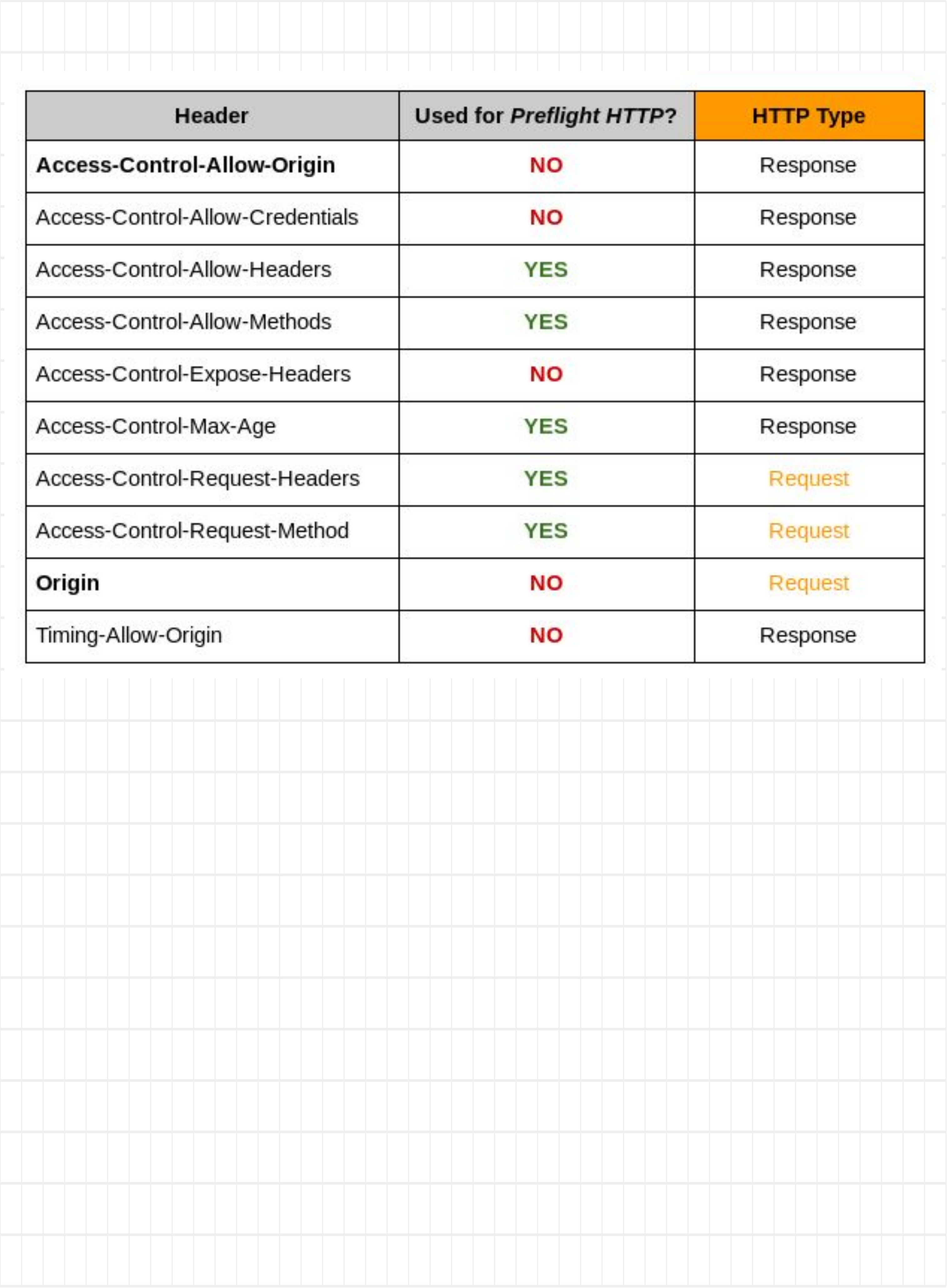
FEB
2021

Preflight HTTP Request

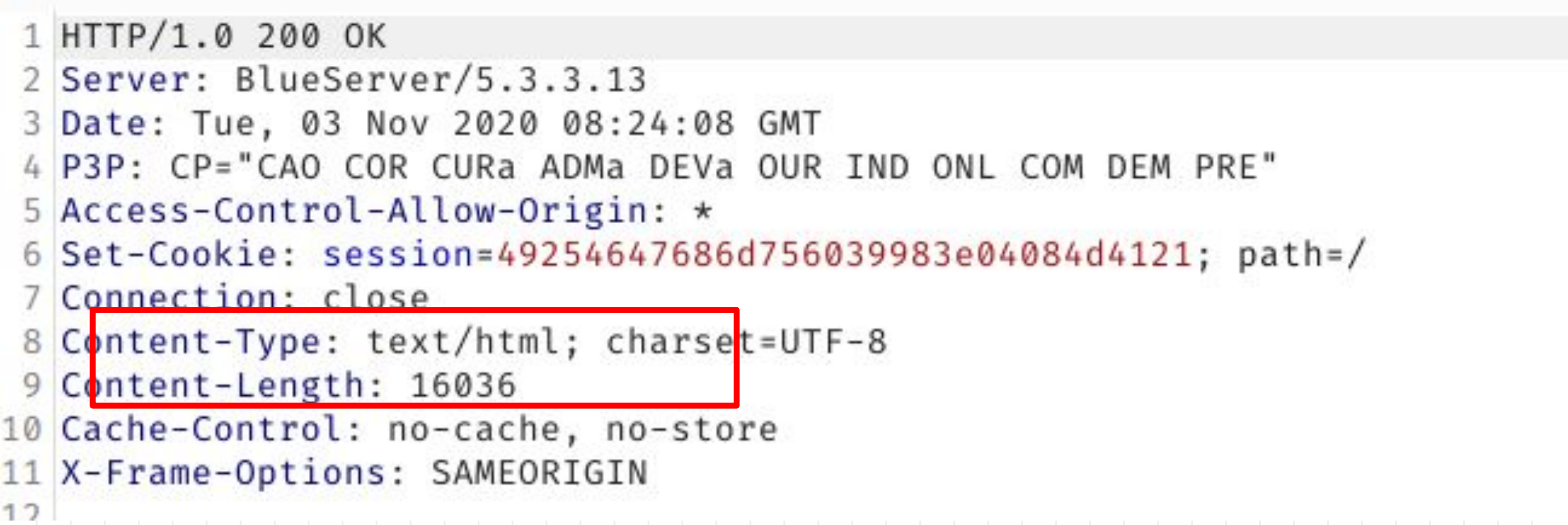


CORS
Browser
Allowed Origins
https://www.mywebsite.com
Allowed Methods
GET POST PUT
Server

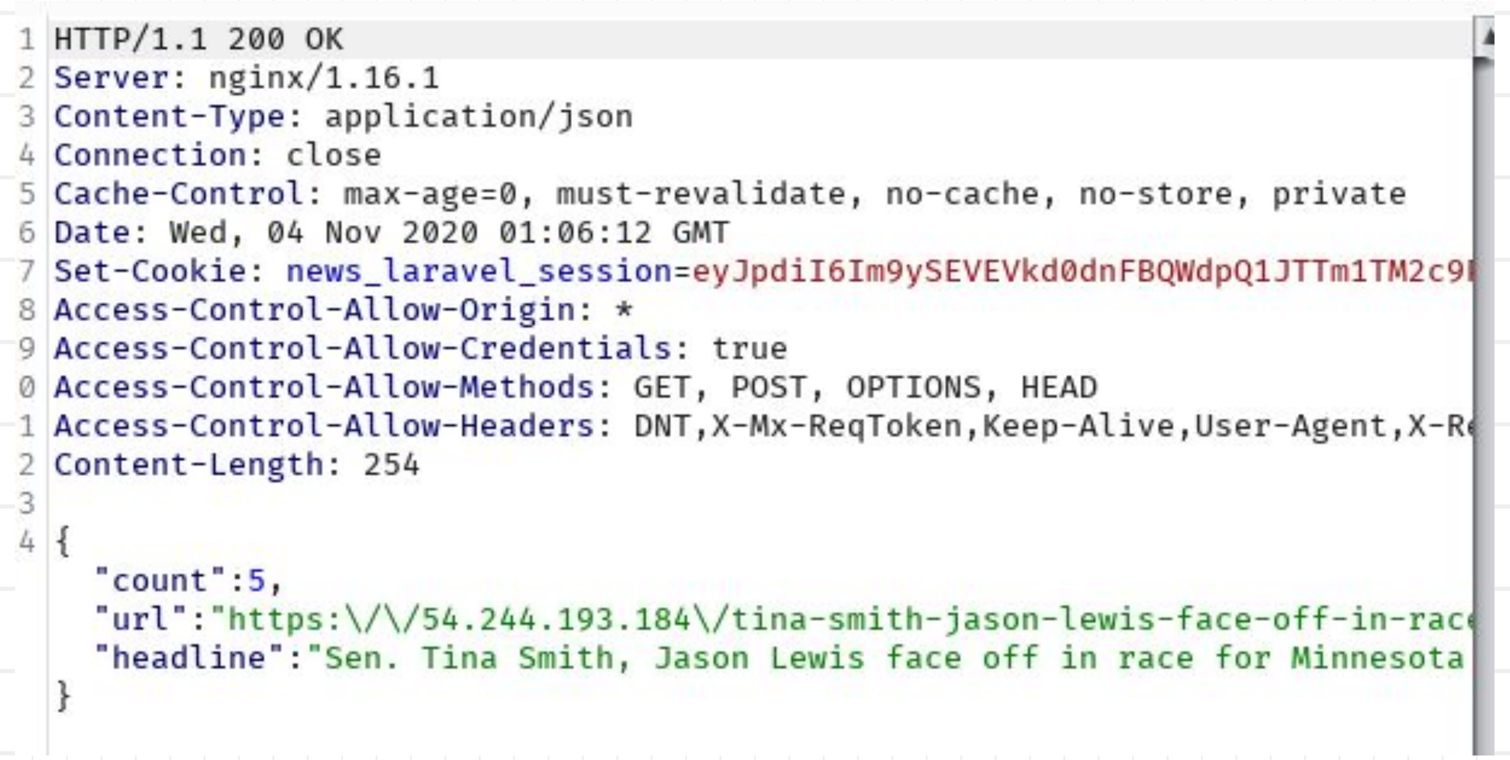

Allowed Origins
https://www.mywebsite.com
Allowed Methods
GET POST PUT
CORS
Browser
Server



Header	Used for *Preflight HTTP*?	HTTP Type
Access-Control-Allow-Origin	NO	Response
Access-Control-Allow-Credentials	NO	Response
Access-Control-Allow-Headers	YES	Response
Access-Control-Allow-Methods	YES	Response
Access-Control-Expose-Headers	NO	Response
Access-Control-Max-Age	YES	Response
Access-Control-Request-Headers	YES	Request
Access-Control-Request-Method	YES	Request
Origin	NO	Request
Timing-Allow-Origin	NO	Response



```
HTTP/1.0 200 OK
Server: BlueServer/5.3.3.13
Date: Tue, 03 Nov 2020 08:24:08 GMT
P3P: CP="CAO COR CURa ADMa DEVa OUR IND ONL COM DEM PRE"
Access-Control-Allow-Origin: *
Set-Cookie: session=49254647686d756039983e04084d4121; path=/
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Length: 16036
Cache-Control: no-cache, no-store
X-Frame-Options: SAMEORIGIN
```



```
HTTP/1.1 200 OK
Server: nginx/1.16.1
Content-Type: application/json
Connection: close
Cache-Control: max-age=0, must-revalidate, no-cache, no-store, private
Date: Wed, 04 Nov 2020 01:06:12 GMT
Set-Cookie: news_laravel_session=eyJpdiI6Im9ySEVEVkd0dnFBQWdpQ1JTTm1TM2c9
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, POST, OPTIONS, HEAD
Access-Control-Allow-Headers: DNT,X-Mx-ReqToken,Keep-Alive,User-Agent,X-R
Content-Length: 254

{
  "count":5,
  "url":"https:\/\/54.244.193.184\/tina-smith-jason-lewis-face-off-in-race
  "headline":"Sen. Tina Smith, Jason Lewis face off in race for Minnesota
}
```

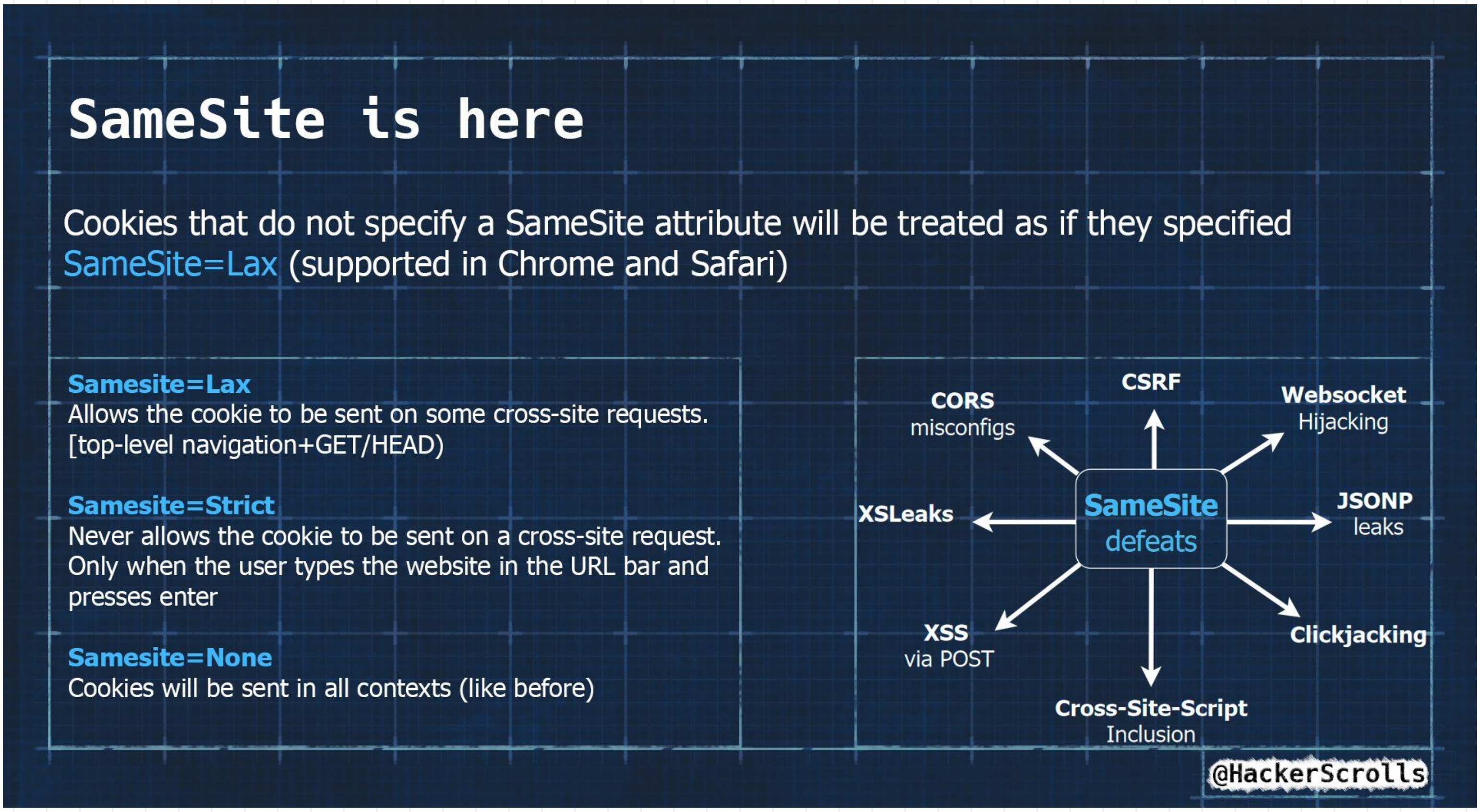
Same-Site Cookie

- SameSite = Strict
- SameSite = Lax
- SameSite = None

Browser

Personal, yet safe, experience offered by SameSite

Same-Site Cookie



SameSite is here

Cookies that do not specify a SameSite attribute will be treated as if they specified SameSite=Lax (supported in Chrome and Safari)

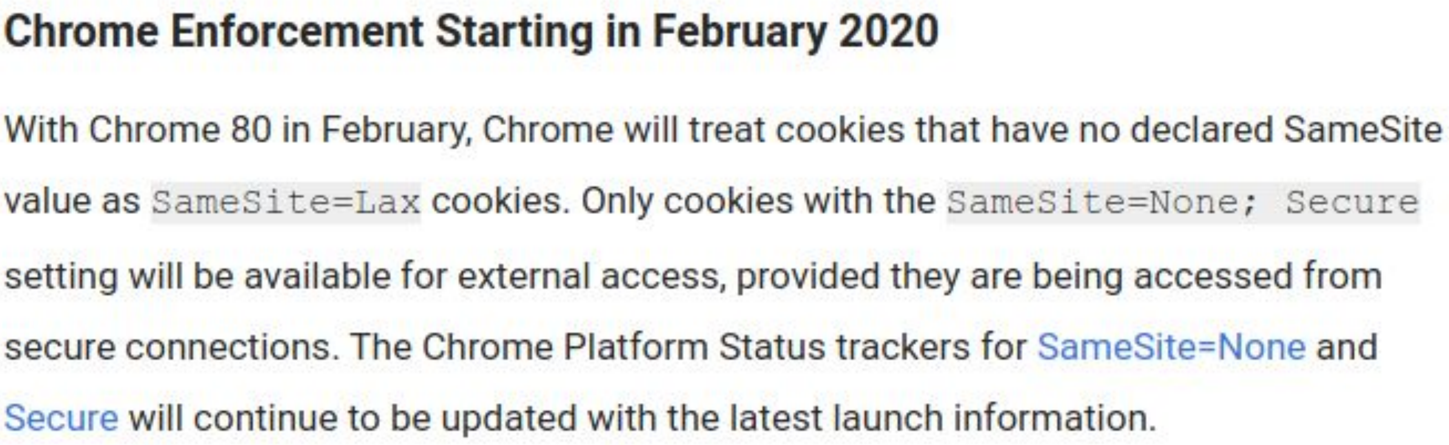
Samesite=Lax
Allows the cookie to be sent on some cross-site requests.
[top-level navigation+GET/HEAD)

Samesite=Strict
Never allows the cookie to be sent on a cross-site request.
Only when the user types the website in the URL bar and presses enter

Samesite=None
Cookies will be sent in all contexts (like before)

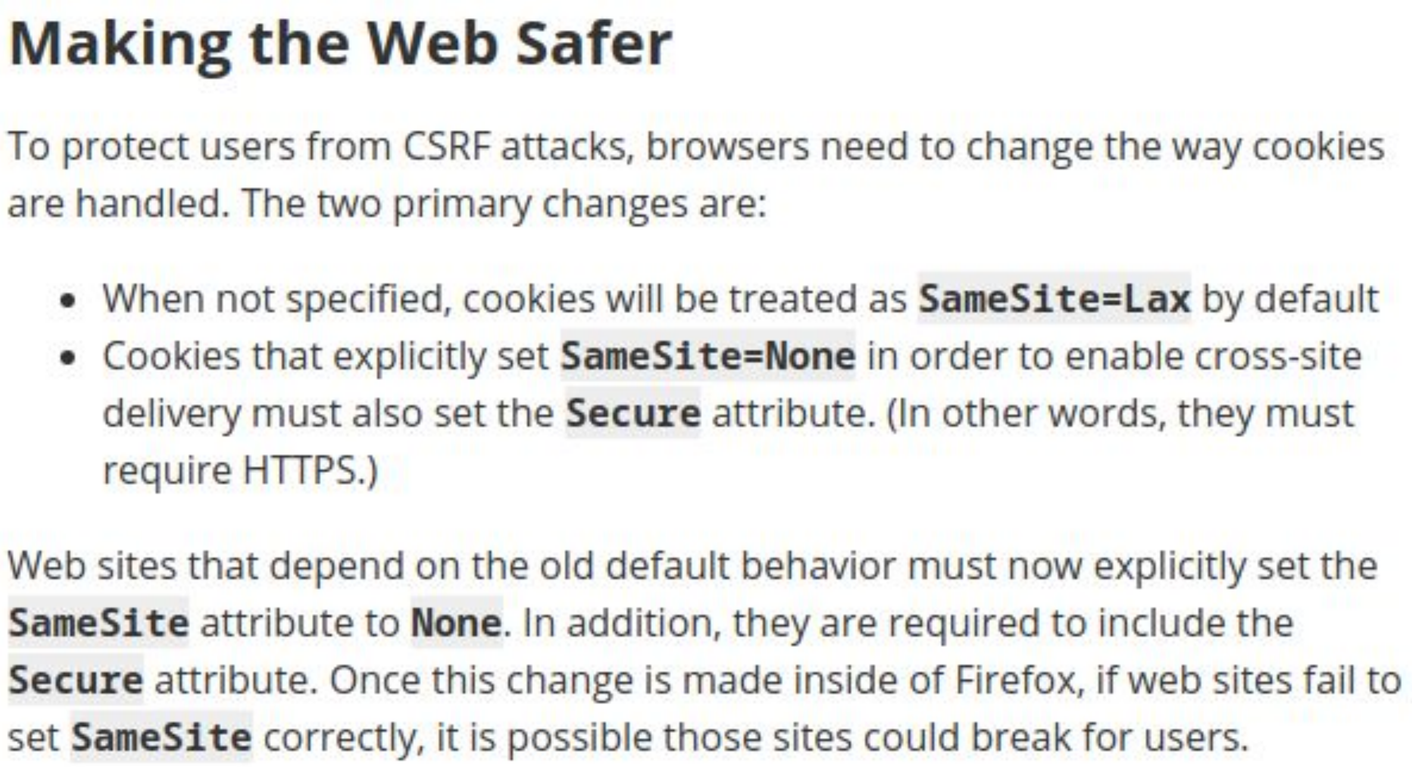
@HackerScrolls

Same-Site Cookie



Chrome Enforcement Starting in February 2020

With Chrome 80 in February, Chrome will treat cookies that have no declared SameSite value as `SameSite=Lax` cookies. Only cookies with the `SameSite=None; Secure` setting will be available for external access, provided they are being accessed from secure connections. The Chrome Platform Status trackers for SameSite=None and Secure will continue to be updated with the latest launch information.



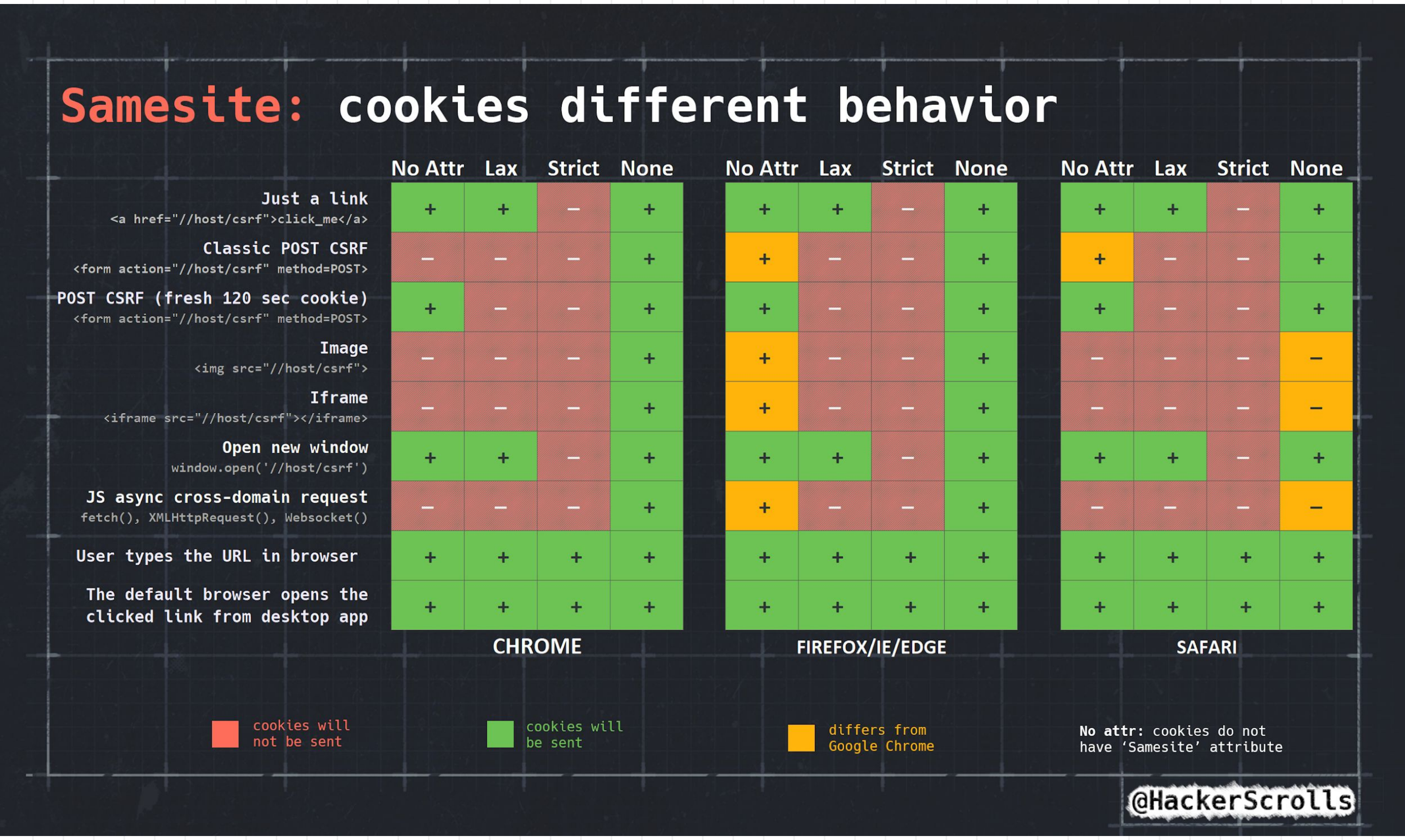
Making the Web Safer

To protect users from CSRF attacks, browsers need to change the way cookies are handled. The two primary changes are:

- When not specified, cookies will be treated as **`SameSite=Lax`** by default
- Cookies that explicitly set **`SameSite=None`** in order to enable cross-site delivery must also set the **`Secure`** attribute. (In other words, they must require HTTPS.)

Web sites that depend on the old default behavior must now explicitly set the **`SameSite`** attribute to **`None`**. In addition, they are required to include the **`Secure`** attribute. Once this change is made inside of Firefox, if web sites fail to set **`SameSite`** correctly, it is possible those sites could break for users.

Same-Site Cookie



Samesite: cookies different behavior

	CHROME				FIREFOX/IE/EDGE				SAFARI			
	No Attr	Lax	Strict	None	No Attr	Lax	Strict	None	No Attr	Lax	Strict	None
Just a link `<a href="//host/csrf">click_me</a>`	+	+	−	+	+	+	−	+	+	+	−	+
Classic POST CSRF `<form action="//host/csrf" method=POST>`	−	−	−	+	🟧 +	−	−	+	🟧 +	−	−	+
POST CSRF (fresh 120 sec cookie) `<form action="//host/csrf" method=POST>`	+	−	−	+	+	−	−	+	+	−	−	+
Image `<img src="//host/csrf">`	−	−	−	+	🟧 +	−	−	+	−	−	−	🟧 −
Iframe `<iframe src="//host/csrf"></iframe>`	−	−	−	+	🟧 +	−	−	+	−	−	−	🟧 −
Open new window `window.open('//host/csrf')`	+	+	−	+	+	+	−	+	+	+	−	+
JS async cross-domain request `fetch(), XMLHttpRequest(), Websocket()`	−	−	−	+	🟧 +	−	−	+	−	−	−	🟧 −
User types the URL in browser	+	+	+	+	+	+	+	+	+	+	+	+
The default browser opens the clicked link from desktop app	+	+	+	+	+	+	+	+	+	+	+	+

🟥 cookies will not be sent

🟩 cookies will be sent

🟧 differs from Google Chrome

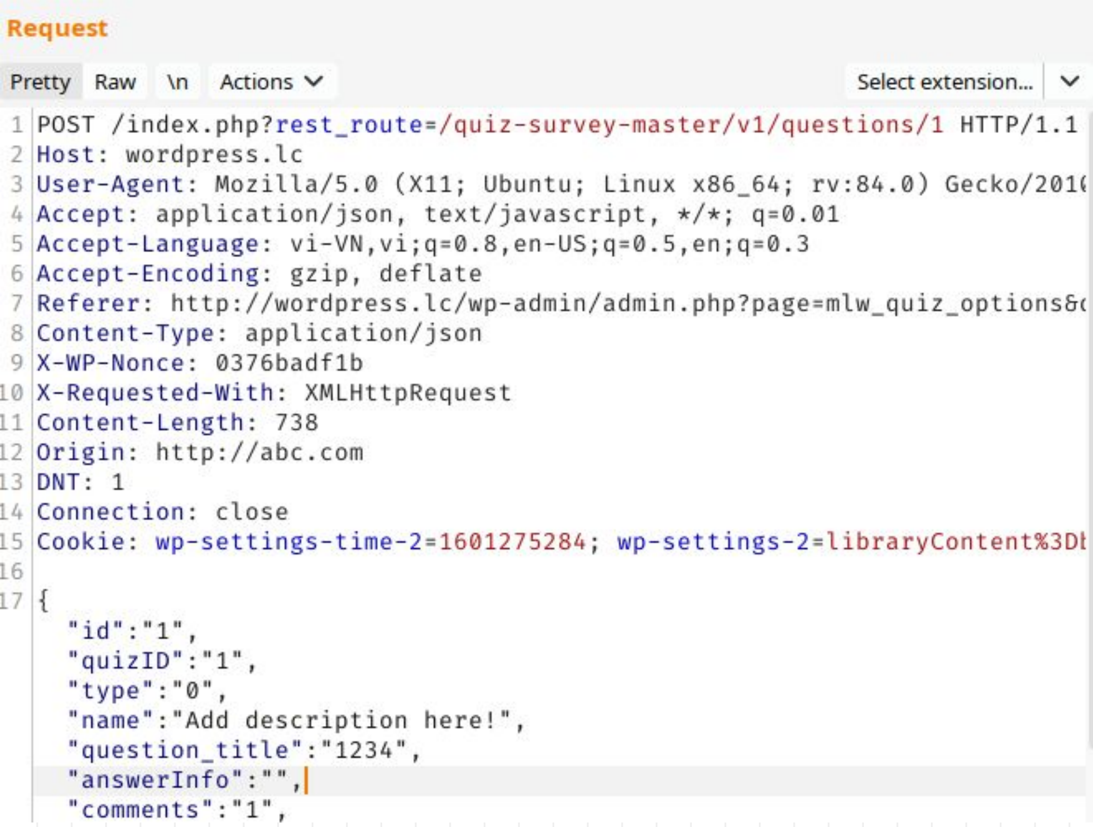
No attr: cookies do not have 'Samesite' attribute

@HackerScrolls

Apr
202
1

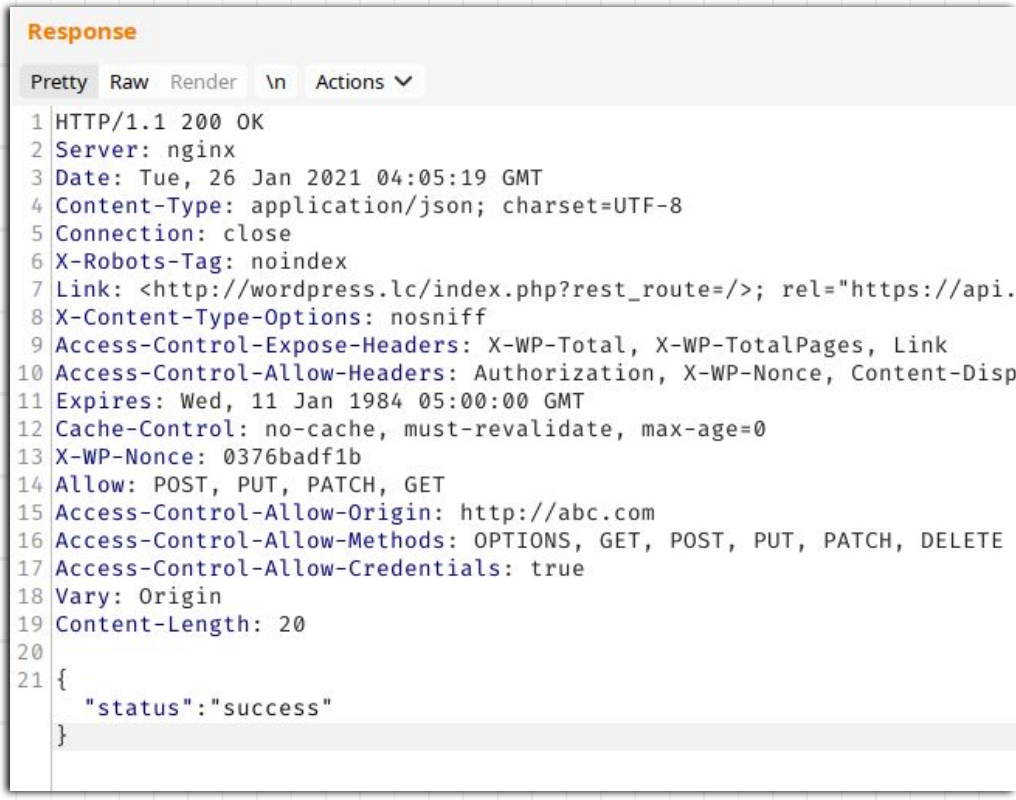
Hacking

Taking advantage of unsafe CORS configuration



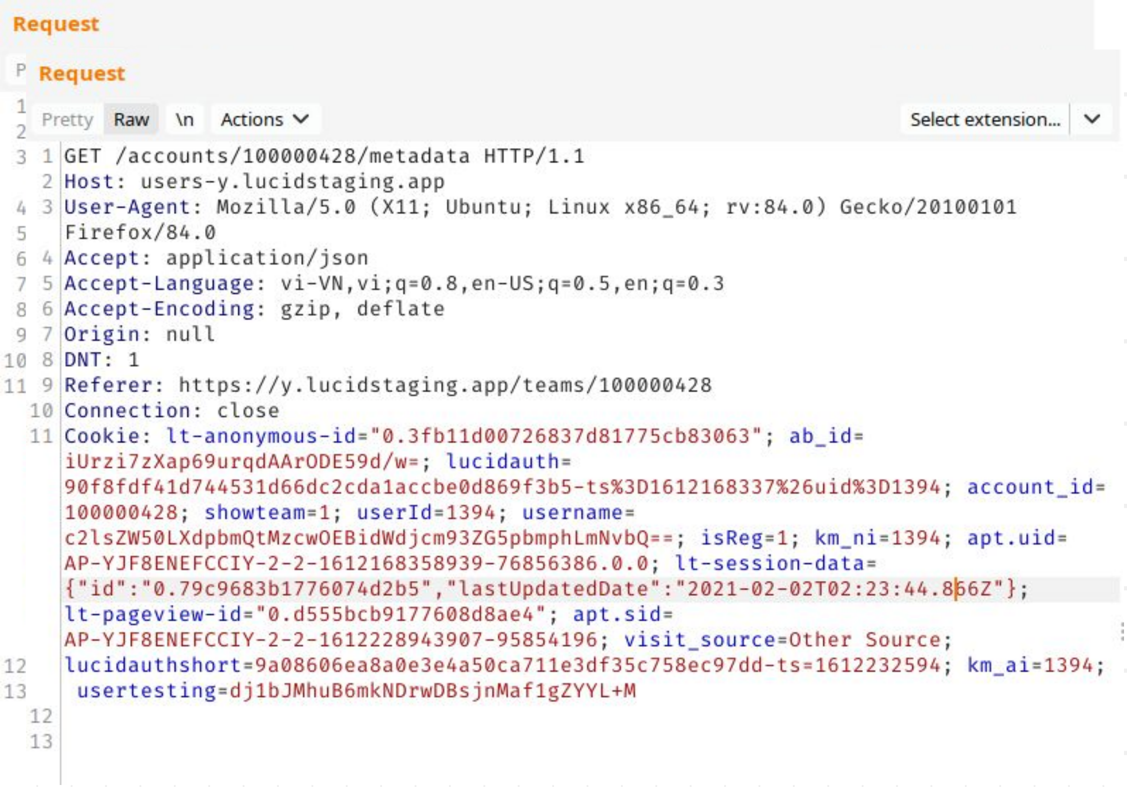
```
POST /index.php?rest_route=/quiz-survey-master/v1/questions/1 HTTP/1.1
Host: wordpress.lc
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/201
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://wordpress.lc/wp-admin/admin.php?page=mlw_quiz_options&
Content-Type: application/json
X-WP-Nonce: 0376badf1b
X-Requested-With: XMLHttpRequest
Content-Length: 738
Origin: http://abc.com
DNT: 1
Connection: close
Cookie: wp-settings-time-2=1601275284; wp-settings-2=libraryContent%3D

{
  "id":"1",
  "quizID":"1",
  "type":"0",
  "name":"Add description here!",
  "question_title":"1234",
  "answerInfo":"",
  "comments":"1",
```

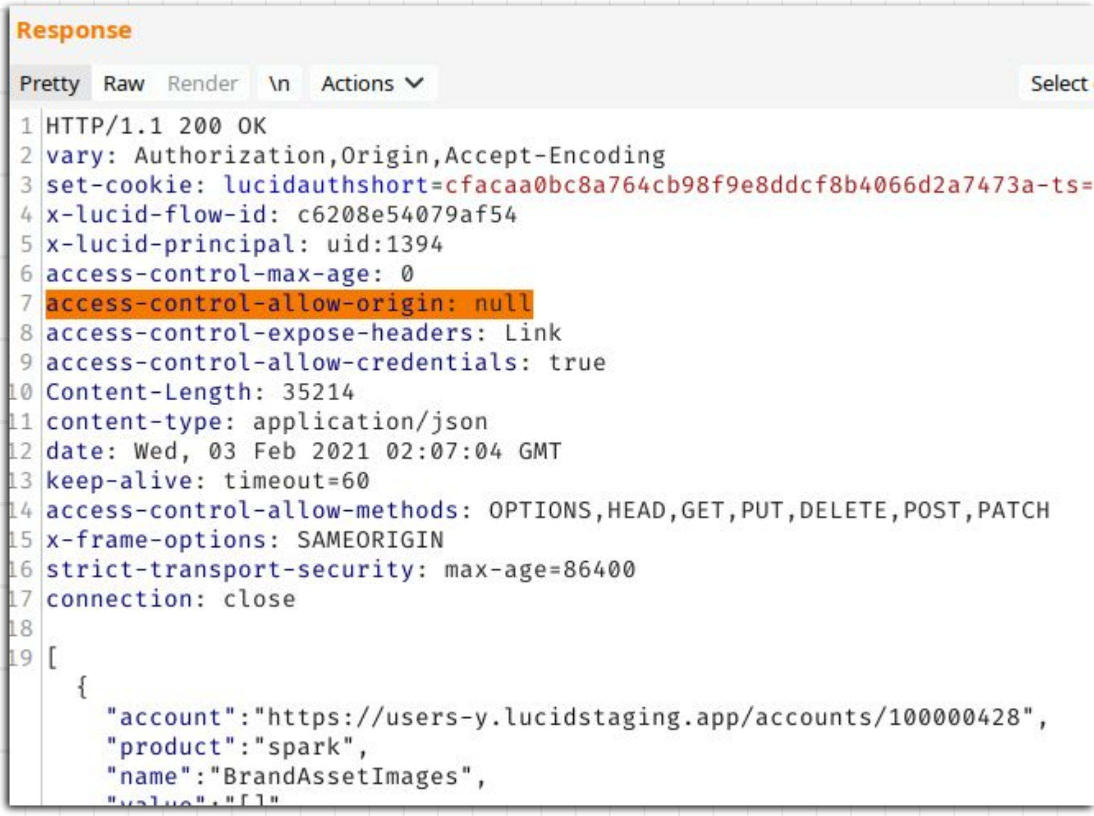


```
HTTP/1.1 200 OK
Server: nginx
Date: Tue, 26 Jan 2021 04:05:19 GMT
Content-Type: application/json; charset=UTF-8
Connection: close
X-Robots-Tag: noindex
Link: <http://wordpress.lc/index.php?rest_route=/>; rel="https://api.
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disp
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
X-WP-Nonce: 0376badf1b
Allow: POST, PUT, PATCH, GET
Access-Control-Allow-Origin: http://abc.com
Access-Control-Allow-Methods: OPTIONS, GET, POST, PUT, PATCH, DELETE
Access-Control-Allow-Credentials: true
Vary: Origin
Content-Length: 20

{
  "status":"success"
}
```



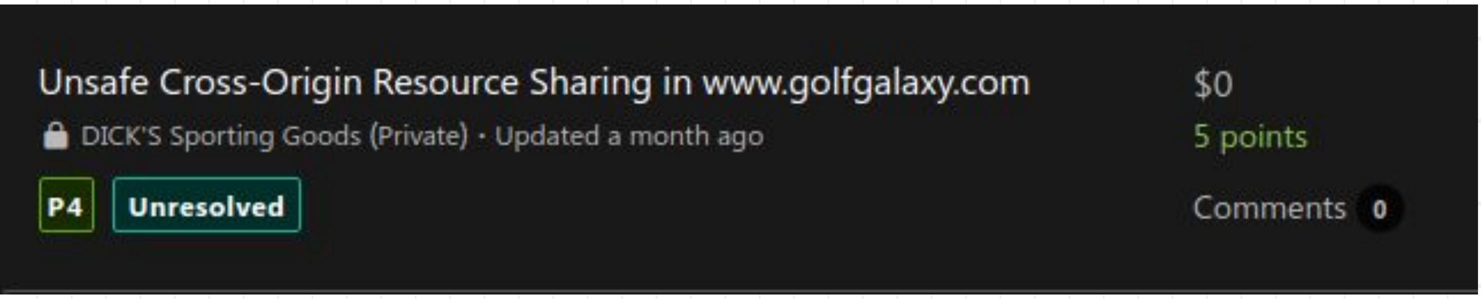
```
GET /accounts/100000428/metadata HTTP/1.1
Host: users-y.lucidstaging.app
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:84.0) Gecko/20100101
Firefox/84.0
Accept: application/json
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Origin: null
DNT: 1
Referer: https://y.lucidstaging.app/teams/100000428
Connection: close
Cookie: lt-anonymous-id="0.3fb11d00726837d81775cb83063"; ab_id=
iUrzi7zXap69urqdAArODE59d/w=; lucidauth=
90f8fdf41d744531d66dc2cda1accbe0d869f3b5-ts%3D1612168337%26uid%3D1394; account_id=
100000428; showteam=1; userId=1394; username=
c2lsZW50LXdpbmQtMzcwOEBidWdjcm93ZG5pbmphLmNvbQ==; isReg=1; km_ni=1394; apt.uid=
AP-YJF8ENEFCCIY-2-2-1612168358939-76856386.0.0; lt-session-data=
{"id":"0.79c9683b1776074d2b5","lastUpdatedDate":"2021-02-02T02:23:44.866Z"};
lt-pageview-id="0.d555bcb9177608d8ae4"; apt.sid=
AP-YJF8ENEFCCIY-2-2-1612228943907-95854196; visit_source=Other Source;
lucidauthshort=9a08606ea8a0e3e4a50ca711e3df35c758ec97dd-ts=1612232594; km_ai=1394;
 usertesting=dj1bJMhuB6mkNDrwDBsjnMaf1gZYYL+M
```



```
HTTP/1.1 200 OK
vary: Authorization,Origin,Accept-Encoding
set-cookie: lucidauthshort=cfacaa0bc8a764cb98f9e8ddcf8b4066d2a7473a-ts=
x-lucid-flow-id: c6208e54079af54
x-lucid-principal: uid:1394
access-control-max-age: 0
access-control-allow-origin: null
access-control-expose-headers: Link
access-control-allow-credentials: true
Content-Length: 35214
content-type: application/json
date: Wed, 03 Feb 2021 02:07:04 GMT
keep-alive: timeout=60
access-control-allow-methods: OPTIONS,HEAD,GET,PUT,DELETE,POST,PATCH
x-frame-options: SAMEORIGIN
strict-transport-security: max-age=86400
connection: close

[
  {
    "account":"https://users-y.lucidstaging.app/accounts/100000428",
    "product":"spark",
    "name":"BrandAssetImages",
    "value":"[]"
```

REWARD



Unsafe Cross-Origin Resource Sharing in www.golfgalaxy.com

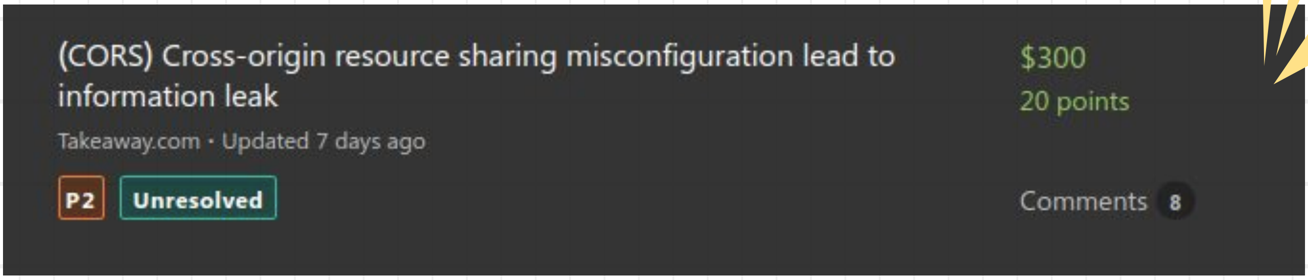
🔒 DICK'S Sporting Goods (Private) • Updated a month ago

P4 Unresolved

$0

5 points

Comments 0



(CORS) Cross-origin resource sharing misconfiguration lead to information leak

Takeaway.com • Updated 7 days ago

P2 Unresolved

$300

20 points

Comments 8

THANKS!

Does anyone have any questions?
nguy.minh.tuan@sun-asterisk.com
research.sun-asterisk.com