SAME ORIGIN POLICY

Isolating content between domains since 1996

Riyaz Walikar | @riyazwalikar | www.riyazwalikar.com

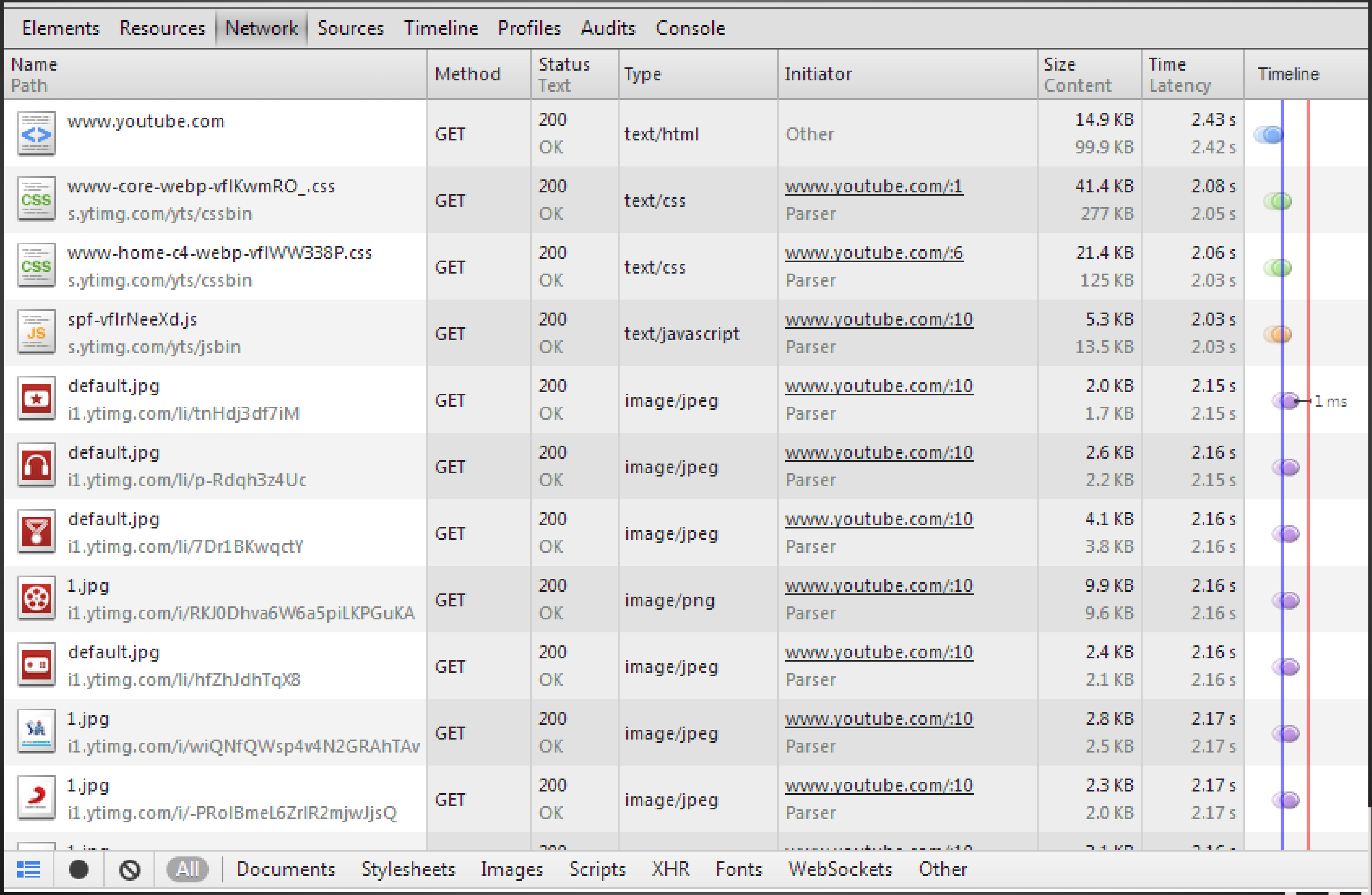
whoami

- WebAppSec Consultant, Penetration Tester, Bug Bounty Hunter for Google, Facebook, Paypal, Mozilla and other bounty programs
- One of the null Security Community Bangalore Chapter Moderator
- Work at a Big4 and have conducted several Penetration Tests all over the world.
- Spoken at several international security conferences

Imagine the Internet if

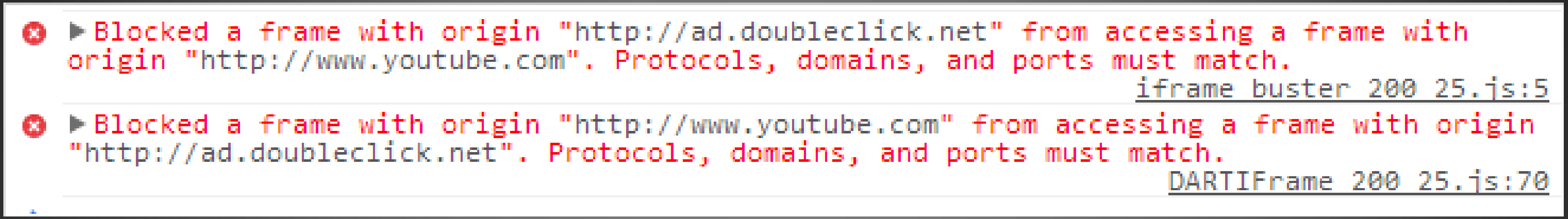
- fabekook.cn was able to read DOM values from facebook.com from another browser tab
- gmaail.br was able to read your address book from http://mail.google.com/mail/c/data/contactstore?type=4&max=-1
- boinkofindia.com was able to read your account balance and obtain a list of all your transactions from your internet banking account while you are logged in.

Why is this possible?

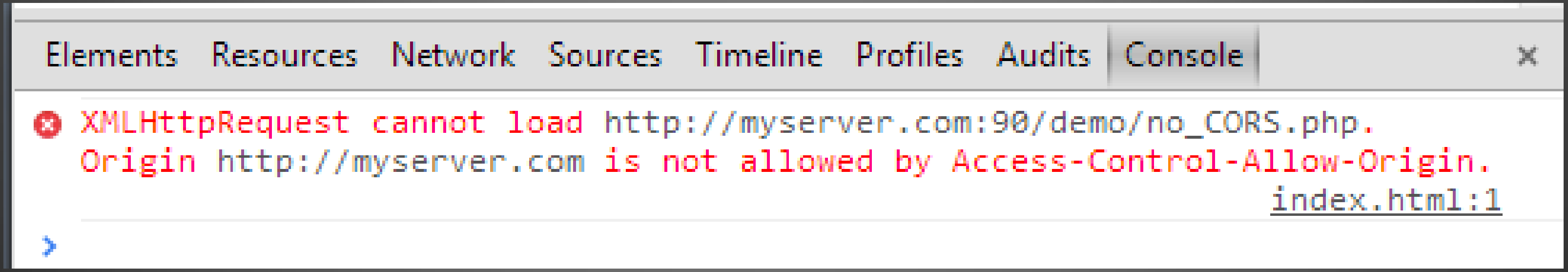


Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
www.youtube.com	GET	200 OK	text/html	Other	14.9 KB 99.9 KB	2.43 s 2.42 s	
www-core-webp-vflKwmRO_.css s.ytimg.com/yts/cssbin	GET	200 OK	text/css	www.youtube.com/:1 Parser	41.4 KB 277 KB	2.08 s 2.05 s	
www-home-c4-webp-vflWW338P.css s.ytimg.com/yts/cssbin	GET	200 OK	text/css	www.youtube.com/:6 Parser	21.4 KB 125 KB	2.06 s 2.03 s	
spf-vflrNeeXd.js s.ytimg.com/yts/jsbin	GET	200 OK	text/javascript	www.youtube.com/:10 Parser	5.3 KB 13.5 KB	2.03 s 2.03 s	
default.jpg i1.ytimg.com/li/tnHdj3df7iM	GET	200 OK	image/jpeg	www.youtube.com/:10 Parser	2.0 KB 1.7 KB	2.15 s 2.15 s	1 ms
default.jpg i1.ytimg.com/li/p-Rdqh3z4Uc	GET	200 OK	image/jpeg	www.youtube.com/:10 Parser	2.6 KB 2.2 KB	2.16 s 2.15 s	
default.jpg i1.ytimg.com/li/7Dr1BKwqctY	GET	200 OK	image/jpeg	www.youtube.com/:10 Parser	4.1 KB 3.8 KB	2.16 s 2.16 s	
1.jpg i1.ytimg.com/i/RKJ0Dhva6W6a5piLKPGuKA	GET	200 OK	image/png	www.youtube.com/:10 Parser	9.9 KB 9.6 KB	2.16 s 2.16 s	
default.jpg i1.ytimg.com/li/hfZhJdhTqX8	GET	200 OK	image/jpeg	www.youtube.com/:10 Parser	2.4 KB 2.1 KB	2.16 s 2.16 s	
1.jpg i1.ytimg.com/i/wiQNfQWsp4v4N2GRAhTAv	GET	200 OK	image/jpeg	www.youtube.com/:10 Parser	2.8 KB 2.5 KB	2.17 s 2.17 s	
1.jpg i1.ytimg.com/i/-PRoIBmeL6ZrlR2mjwJjsQ	GET	200 OK	image/jpeg	www.youtube.com/:10 Parser	2.3 KB 2.0 KB	2.17 s 2.17 s	
1.jpg		200		www.youtube.com/:10	2.1 KB	2.16 s	

Uh oh!



```
Blocked a frame with origin "http://ad.doubleclick.net" from accessing a frame with
origin "http://www.youtube.com". Protocols, domains, and ports must match.
                                                              iframe buster 200 25.js:5
Blocked a frame with origin "http://www.youtube.com" from accessing a frame with origin
"http://ad.doubleclick.net". Protocols, domains, and ports must match.
                                                              DARTIFrame 200 25.js:70
```



```
XMLHttpRequest cannot load http://myserver.com:90/demo/no_CORS.php.
Origin http://myserver.com is not allowed by Access-Control-Allow-Origin.
                                                              index.html:1
```

n|u

What is SOP?

- SOP restricts how a document or script loaded from one origin can interact with a resource from another origin
- Earliest available implementation – Netscape Navigator 2.0 (1996)

define: origin

Access made from: **http://store.company.com/dir/page.html**

URL	Outcome	Reason
http://store.company.com/dir2/other.html	Success	----
http://store.company.com/dir/inner/another.html	Success	----
https://store.company.com/secure.html	Failure	Different protocol
http://store.company.com:81/dir/etc.html	Failure	Different port
http://news.company.com/dir/other.html	Failure	Different host

Changing 'origins'

- Setting document.domain to a suffix of the current domain.
- Setting document.domain to another domain altogether isn’t allowed.

Demo

A document.domain change set

Cross Origin Network Access

- Origin is permitted to send data to another origin but not read

- Interactions between origins are placed in three categories:
  - Cross origin writes (redirects, links, form action etc.)
  - Cross origin embedding (html tag with src/hrefs)
  - Cross origin reads (not allowed without CORS etc.)

Cross Origin Embedding

- JavaScript <script src="..."></script>.
- CSS with <link rel="stylesheet" href="...">.
- Images with <img>.
- Media files with <video> and <audio> tags.
- Plug-ins with <object>, <embed> and <applet>.
- Fonts with @font-face.
- Anything with <frame> and <iframe>.

Prevent Cross Origin Access

- To prevent Cross origin writes, use a CSRF token
- To prevent Cross origin embedding, ensure resource is not interpreted as any of the formats discussed earlier.
- To prevent Cross Origin reads of a resource, ensure that it is non-embeddable.
- For iframes the X-Frame-Options header can be used to control access to the page.

Cross Origin Resource Sharing

- W3C specification that allows cross domain communication from the browser
- Works by adding new HTTP headers that describe the set of origins that are permitted to read across domains

3 Pointers

- Browsers prevent data from being accessed cross domain via the Same Origin Policy
- In case a page loads another domain via a frame, X-Frame-Options can be used to control access
- CORS is used to relax the Same Origin Policy for legitimate and trusted requests.

n|u

References

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Same_origin_policy_for_JavaScript
- http://www.w3.org/Security/wiki/Same_Origin_Policy
- http://code.google.com/p/browsersec/wiki/Part2

CROSS ORIGIN RESOURCE SHARING

Allowing Cross origin resource sharing since March 2004

Riyaz Walikar | @riyazwalikar | www.riyazwalikar.com

What is CORS?

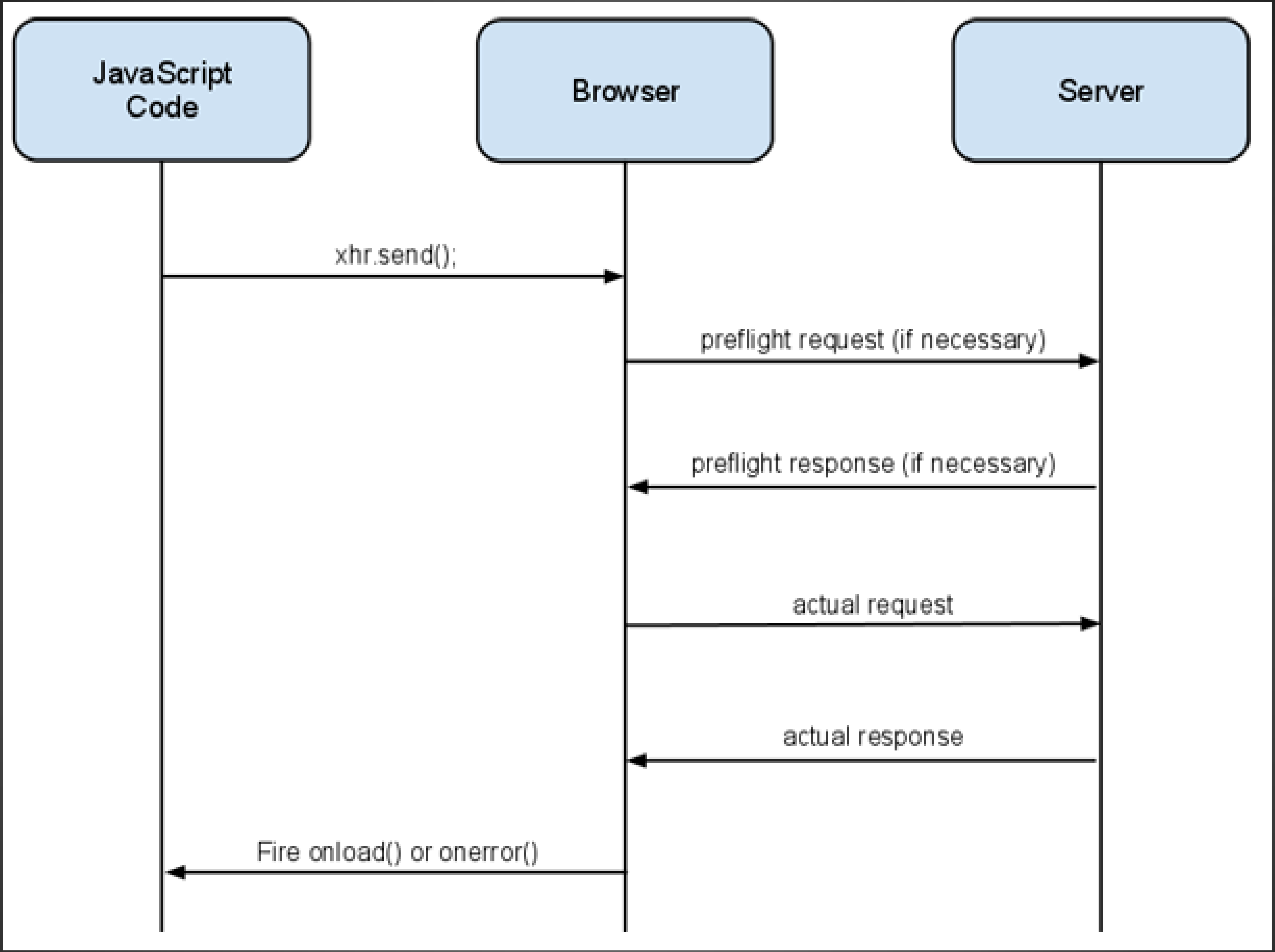
- W3C working draft that defines how the browser and server must communicate when accessing sources across origins

- Implemented via HTTP headers that servers set and browsers enforce

- Can be categorized into
  - Simple requests
  - Requests that need a ‘preflight’

Demo

A simple cross origin request without CORS


JavaScript Code
Browser
Server
xhr.send();
preflight request (if necessary)
preflight response (if necessary)
actual request
actual response
Fire onload() or onerror()

CORS standard exchange between client and server

Why is CORS needed?

- For legitimate and trusted requests to gain access to authorized data from other domains
- Think cross application data sharing models
- Allows data to be exchanged with trusted sites while using a relaxed Same Origin policy mode.
- Application APIs exposed via web services and trusted domains require CORS to be accessible over the SOP

APIs that support CORS!

amazon webservices™
Google
Spotify
foursquare
flickr™
github SOCIAL CODING
DBpedia
PublishMyData
SOUNDCLOUD
doctape beta
YouTube

CORS – Simple Requests

- Preflight is not needed if
  - Request is a HEAD/GET/POST via XHR
  - No Custom headers
  - Body is text/plain
- Server responds with a CORS header
  - Browser determines access
  - Neither the request, nor response contain cookies

CORS Headers – Simple Request

- Origin
  - Header set by the client for every CORS request
  - Value is the current domain that made the request

- Access-Control-Allow-Origin
  - Set by the server and used by the browser to determine if the response is to be allowed or not.
  - Can be set to * to make resources public (bad practice!)

Demo

A cross origin request with CORS for a simple request

CORS – Requests with Preflight

- Preflight requests are made if
  - Request is a method other than HEAD/GET/POST via XHR (PUT, DELETE etc.)
  - Custom headers are present (X-PINGBACK etc.)
  - Content-Type **other** than application/x-www-form-urlencoded, multipart/form-data, or text/plain
- A transparent request is made to the server requesting access information using OPTIONS

CORS – Requests with Preflight

- Browser sends
  - Origin header
  - Access-Control-Request-Method
  - Access-Control-Request-Headers – (Optional)
- Server sends set of CORS headers that the browser uses to determine if the actual request has to be made or not

CORS Headers – Request with Preflight (Preflight Browser Request)

- Origin
  - Header set by the client for every CORS request
  - Value is the current domain that made the request

- Access-Control-Request-Method:
  - Set by the browser, along with Origin.
  - Value is the method that the request wants to use

- Access-Control-Request-Headers (Optional):
  - A comma separated list of the custom headers being used.

CORS Headers – Request with Preflight (Preflight Server Response)

- Access-Control-Allow-Origin
  - Same as in Simple requests
- Access-Control-Allow-Methods:
  - a comma separated list of allowed methods
- Access-Control-Allow-Headers:
  - a comma separated list of headers that the server will allow.
- Access-Control-Max-Age:
  - the amount of time in seconds that this preflight request should be cached for.

Demo

A cross origin request with CORS for a preflight request

CORS (In)security?

- Several security issues arise from the improper implementation of CORS, most commonly using a universal allow notation (*) in the server headers
- Clients should not trust the received content completely and eval or render content without sanitization which could result in misplaced trust
- The application that allows CORS may become vulnerable to CSRF attacks

n|u

CORS (In)security?

- Prolonged caching of Preflight responses could lead to attacks arising out of abuse of the Preflight Client Cache

- Access control decisions based on the Origin header could result in vulnerabilities as this can be spoofed by an attacker

n|u

CORS Security - Universal Allow

- Setting the 'Access-Control-Allow-Origin' header to *
- Effectively turns the content into a public resource, allowing access from any domain
- Scenarios?
  - An attacker can steal data from an intranet site that has set this header to * by enticing a user to visit an attacker controlled site on the Internet.
  - An attacker can perform attacks on other remote apps via a victim's browser when the victim navigates to an attacker controlled site.

Demo

A universal allow for the Access-Control-Allow-Origin header

CORS Security – Misplaced Trust

- Data exchange between two domains is based on trust
- If one of the servers involved in the exchange of data is compromised then the model of CORS is put at risk
- Scenarios?
  - An attacker can compromise site A and host malicious content knowing site B trusts the data that site A sends to site B via CORS request resulting in XSS and other attacks.
  - An attacker can compromise site B and use the exposed CORS functionality in site A to attack users in site A.

CSRF with CORS

- Server may process client request to change server side data while verifying that the Origin header was set

- An attacker can use the .withCredentials = “true” property of XHR to replay any cookies to the application on which the victim is logged in

- Scenarios?
  - An attacker sets the Origin header or uses a trusted site A to send a non idempotent request to site B
  - The victim who is logged into site B when he is viewing the trusted site A causes site B to create a user account without his knowledge via a CSRF attack

Demo

A CSRF attack that creates a user using a trusted site via CORS

CORS – Caching of Preflight responses

- The Access-Control-Max-Age header is set to a high value, allowing browsers to cache Preflight responses
- Caching the preflight response for longer duration can pose a security risk.
- If the COR access-control policy is changed on the server the browser would still follow the old policy available in the Preflight Result Cache

CORS – Access Control based on Origin

- The Origin header indicates that the request is from a particular domain, but does not guarantee it
- Spoofing the Origin header allows access to the page if access is based on this header
- Scenarios?
  - An attacker sets the Origin header to view sensitive information that is restricted
  - Attacker uses cURL to set a custom origin header:

```
curl --header 'origin:http://someserver.com'
http://myserver.com:90/demo/origin_spoof.php
```

n|u

Demo

Sensitive information revealed via weak Access Control based on the Origin header

References

- http://www.html5rocks.com/en/tutorials/cors/
- https://code.google.com/p/html5security/wiki/CrossOriginRequestSecurity
- http://arunranga.com/examples/access-control/
- http://www.nczonline.net/blog/2010/05/25/cross-domain-ajax-with-cross-origin-resource-sharing/

Questions?

And hopefully answers as well

n|u

Riyaz Walikar | @riyazwalikar | karniv0re@null.co.in