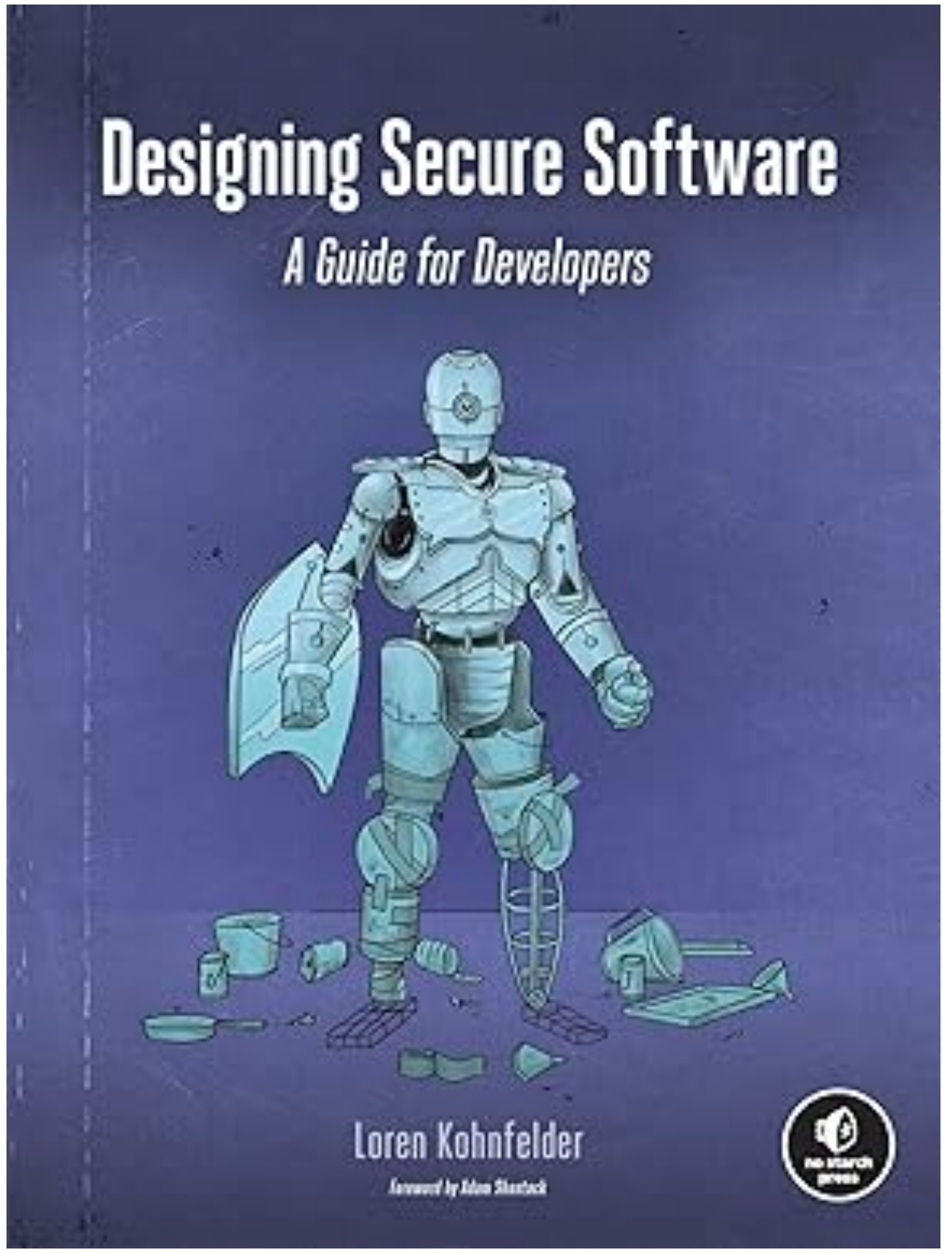


11 Web Security

Topics

- Build on a Framework
- The Web Security Model
  - The HTTP Protocol
  - Digital Certificates and HTTPS
  - The Same Origin Policy
  - Web Cookies

Topics

- Common Web Vulnerabilities
  - Cross-Site Scripting
  - Cross-Site Request Forgery
- More Vulnerabilities and Mitigation

Build on a Framework

Web Framework

- A software framework that supports the development of Web apps
- Examples
    - Laravel
    - Node.js
    - Express.js
    - React JS
    - Angular
    - Next.js
    - Meteor
    - Express
    - Spring
    - PLAY
    - Vue

Using a Framework

- Choose a high-quality framework
- Never override the safeguards it provides
- Let competent experts handle the details

Guidelines

- Choose a framework produced by a trustworthy organization or team that actively develops and maintains it in order to keep up with constantly changing web technologies and practices.
- Look for an explicit security declaration in the documentation. If you don’t find one, I would disqualify the framework.
- Research past performance: the framework doesn’t need a perfect record, but slow responses or ongoing patterns of problems are red flags.
- Build a small prototype and check the resulting HTML for proper escaping and quoting (using inputs like the ones in this chapter’s examples).
- Build a simple test bed to experiment with basic XSS and CSRF attacks, as explained later in this chapter.

The Web Security Model

Servers and Browsers

- Web server can control how they handle data
- But not how the browser does
- Browsers can be out-of-date or otherwise insecure
- Or even maliciously altering requests, as with the Burp proxy
- Many developers assume the browser is operating as expected
- And trust client-side controls
- This leads to many vulnerabilities

The HTTP Protocol

URL (Uniform Resource Locator)

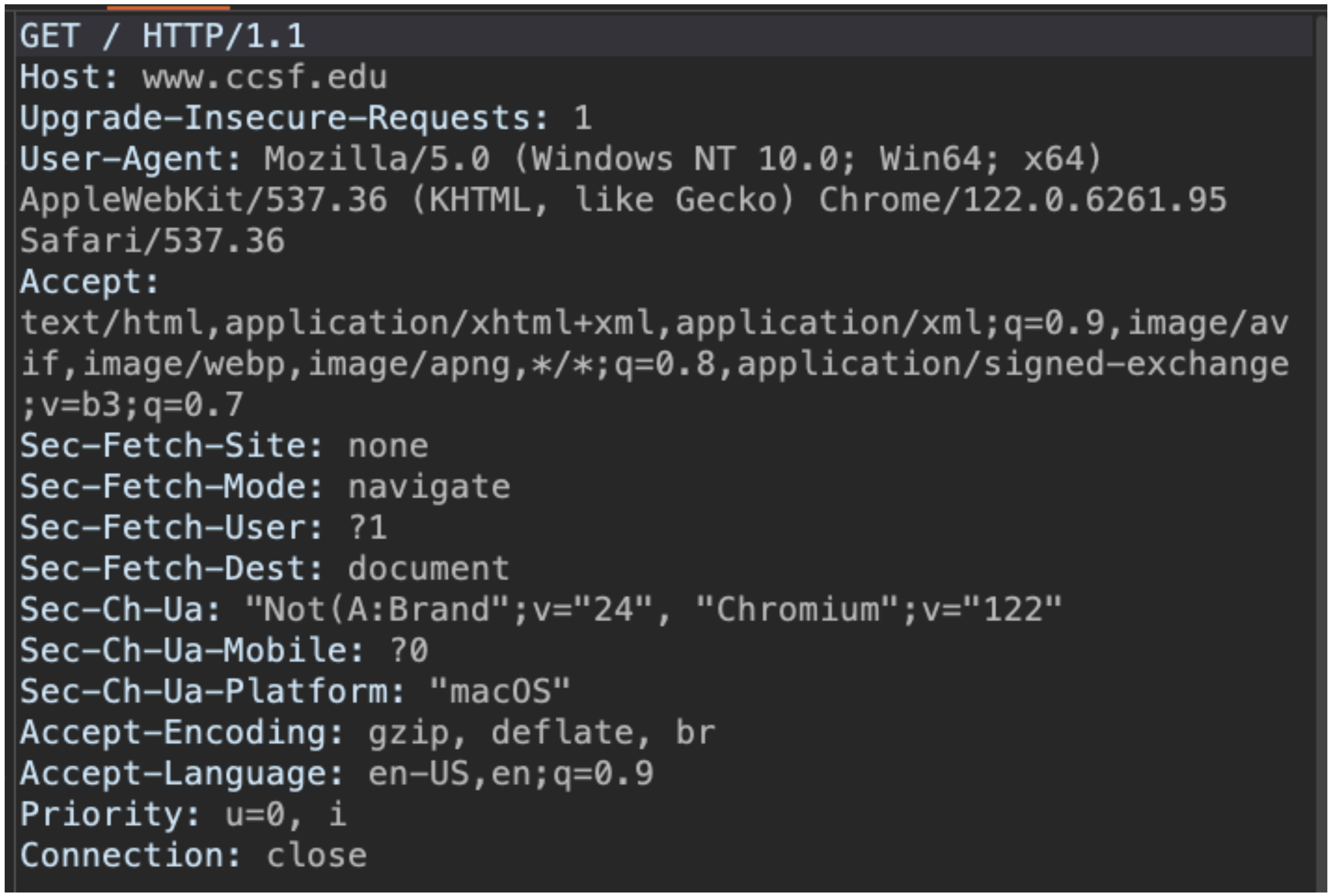
```
http://www.example.com/page.html?query=value#fragment
```

- Scheme **http**
- Domain name **www.example.com**
- Path **page.html**
- Query **query=value**, after ?
  - Also called ***parameters***
- Fragment **fragment**, after #

DNS

- Resolves domain names like **example.com** to IP addresses

HTTP Request

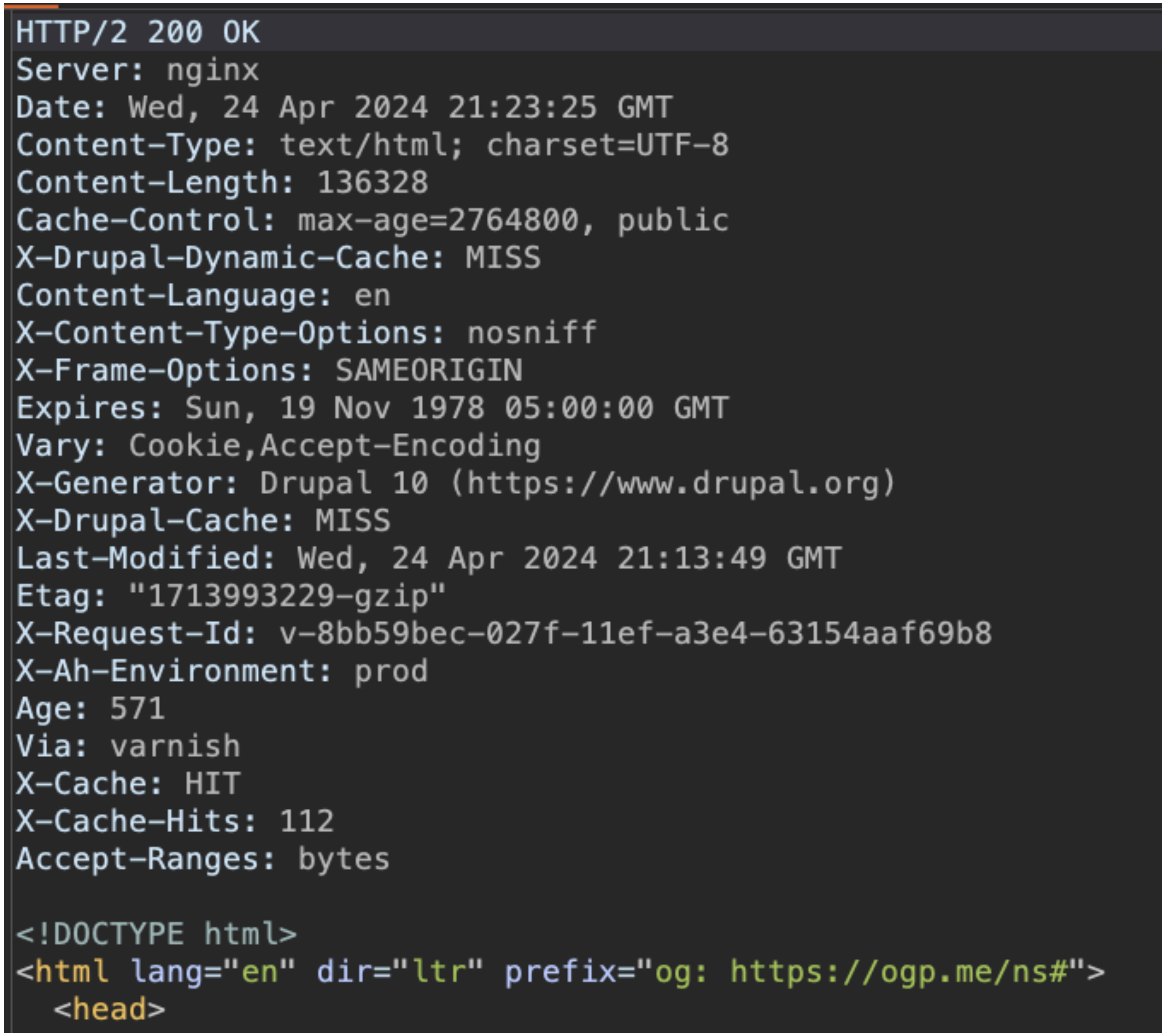


```
GET / HTTP/1.1
Host: www.ccsf.edu
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.6261.95
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/av
if,image/webp,image/apng,*/*;q=0.8,application/signed-exchange
;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Sec-Ch-Ua: "Not(A:Brand";v="24", "Chromium";v="122"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "macOS"
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=0, i
Connection: close
```

- First two lines are required
  - **Verb** (also called the **Method**) and **Host**
- The rest are optional

HTTP Response

- Status code
- Response headers
- Content body



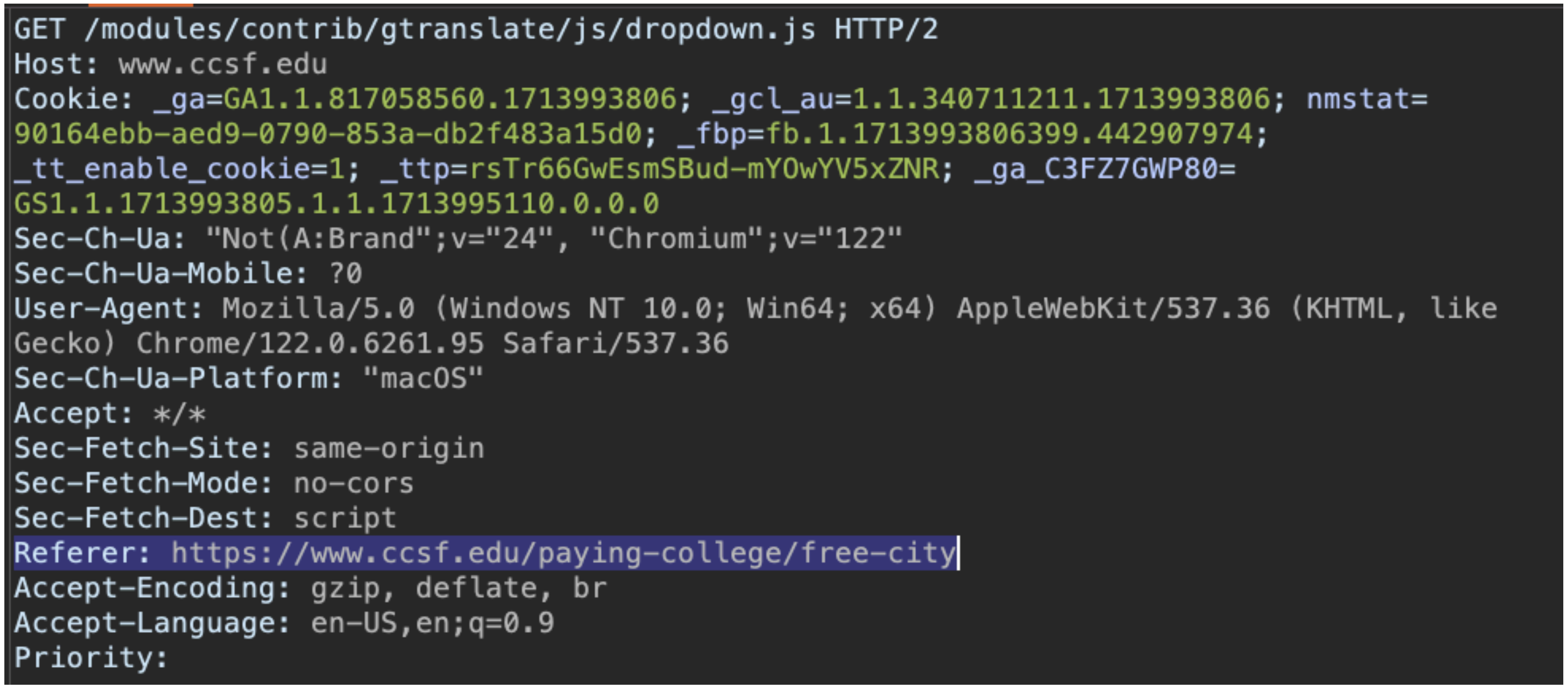
```
HTTP/2 200 OK
Server: nginx
Date: Wed, 24 Apr 2024 21:23:25 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 136328
Cache-Control: max-age=2764800, public
X-Drupal-Dynamic-Cache: MISS
Content-Language: en
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie,Accept-Encoding
X-Generator: Drupal 10 (https://www.drupal.org)
X-Drupal-Cache: MISS
Last-Modified: Wed, 24 Apr 2024 21:13:49 GMT
Etag: "1713993229-gzip"
X-Request-Id: v-8bb59bec-027f-11ef-a3e4-63154aaf69b8
X-Ah-Environment: prod
Age: 571
Via: varnish
X-Cache: HIT
X-Cache-Hits: 112
Accept-Ranges: bytes

<!DOCTYPE html>
<html lang="en" dir="ltr" prefix="og: https://ogp.me/ns#">
  <head>
```

Verbs

- GET
  - Requests content
  - Not state-changing (usually)
  - Don't send sensitive data in GETs
  - Because it will be saved in server logs, referer headers, shortcuts, etc.
- POST
  - Sends data to the server
  - Intending to change the state of the server

Referrer-Policy

- **Referer** header shows what page the request came from
  - Note the incorrect spelling
- The **Referrer-Policy** response header tells the browser to block the Referer request header
- But the browser may not honor the request



```
GET /modules/contrib/gtranslate/js/dropdown.js HTTP/2
Host: www.ccsf.edu
Cookie: _ga=GA1.1.817058560.1713993806; _gcl_au=1.1.340711211.1713993806; nmstat=
90164ebb-aed9-0790-853a-db2f483a15d0; _fbp=fb.1.1713993806399.442907974;
_tt_enable_cookie=1; _ttp=rsTr66GwEsmSBud-mYOwYV5xZNR; _ga_C3FZ7GWP80=
GS1.1.1713993805.1.1.1713995110.0.0.0
Sec-Ch-Ua: "Not(A:Brand";v="24", "Chromium";v="122"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/122.0.6261.95 Safari/537.36
Sec-Ch-Ua-Platform: "macOS"
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: script
Referer: https://www.ccsf.edu/paying-college/free-city
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority:
```

Digital Certificates and HTTPS

HTTPS

- Provides a secure encrypted channel
- Assures that the server is genuine
- Prevents eavesdropping and AiTM (Adversary in the Middoe) attacks

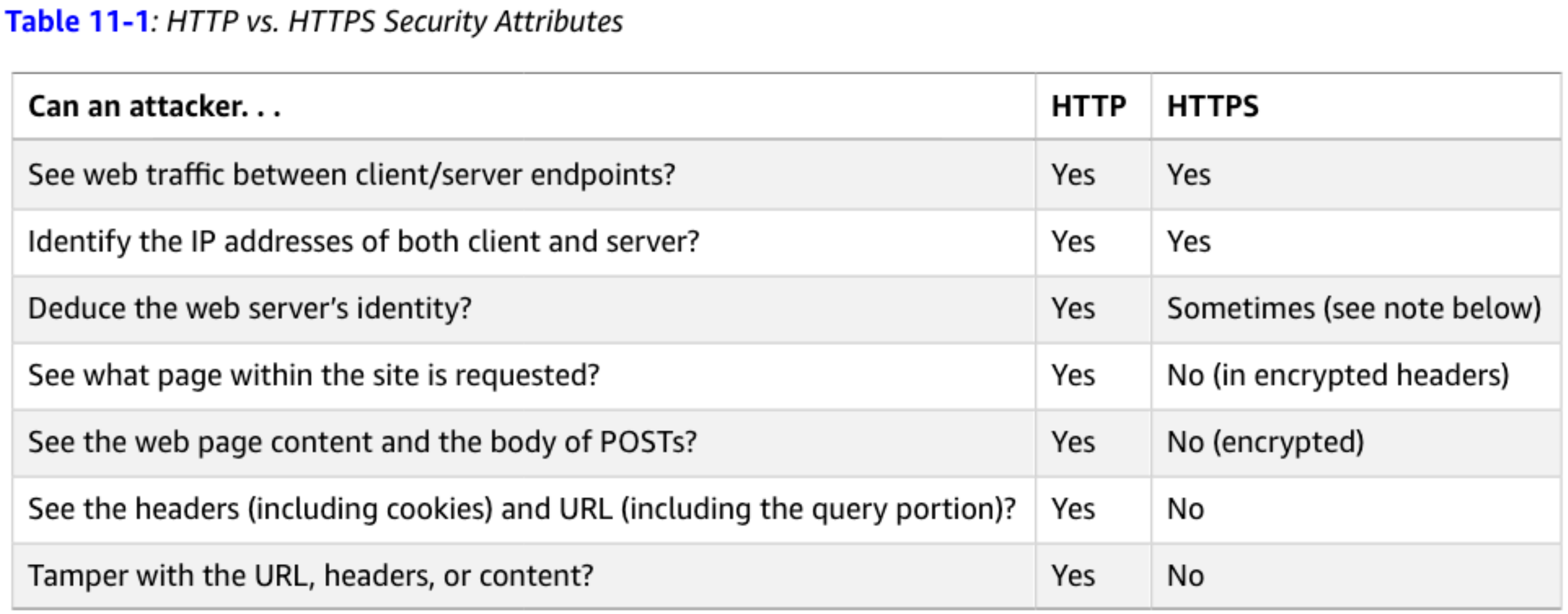
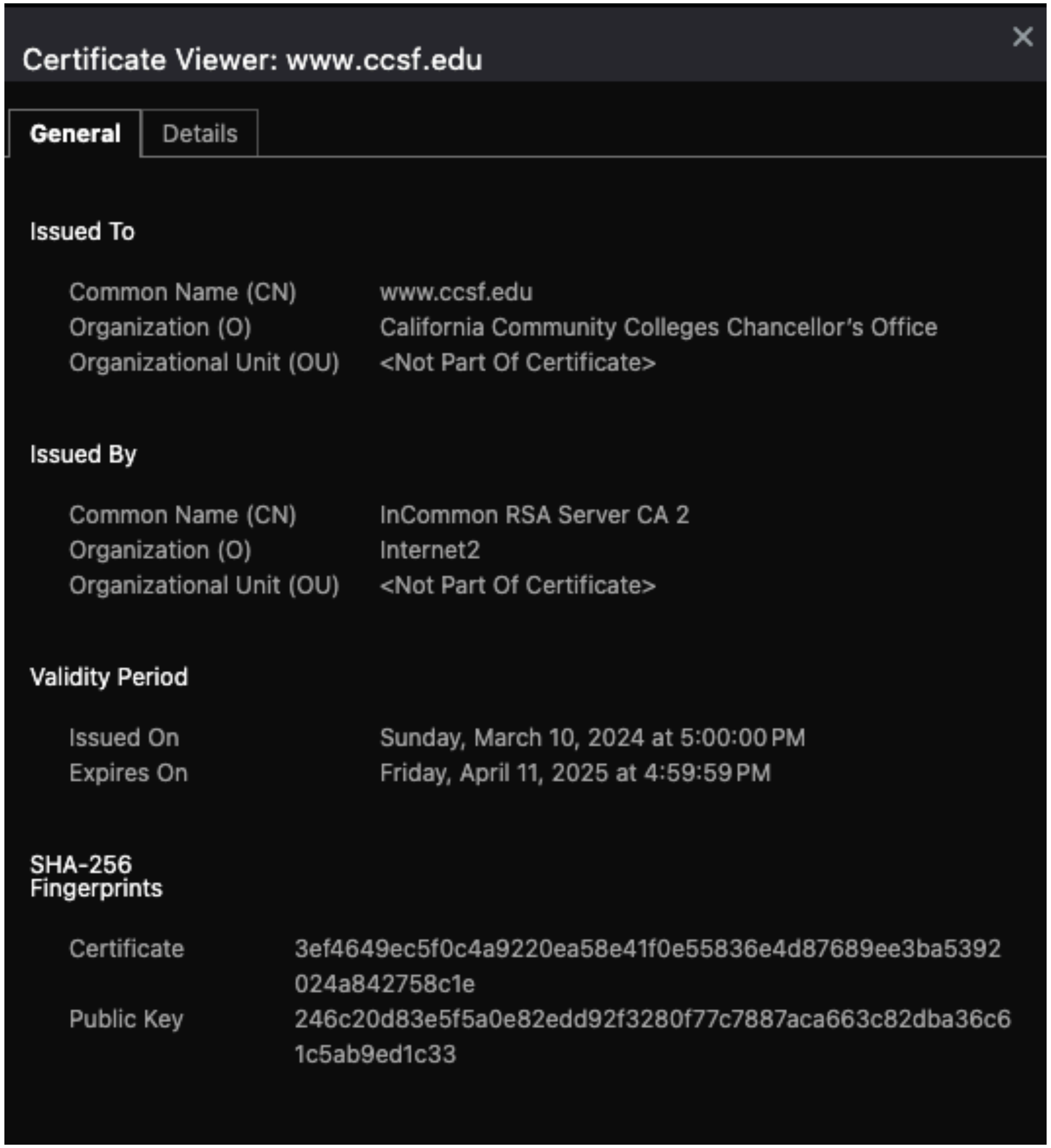


Table 11-1: *HTTP vs. HTTPS Security Attributes*

Can an attacker. . .	HTTP	HTTPS
See web traffic between client/server endpoints?	Yes	Yes
Identify the IP addresses of both client and server?	Yes	Yes
Deduce the web server's identity?	Yes	Sometimes (see note below)
See what page within the site is requested?	Yes	No (in encrypted headers)
See the web page content and the body of POSTs?	Yes	No (encrypted)
See the headers (including cookies) and URL (including the query portion)?	Yes	No
Tamper with the URL, headers, or content?	Yes	No

Example



Certificate Viewer: www.ccsf.edu

General | Details

Issued To

Common Name (CN)	www.ccsf.edu
Organization (O)	California Community Colleges Chancellor's Office
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

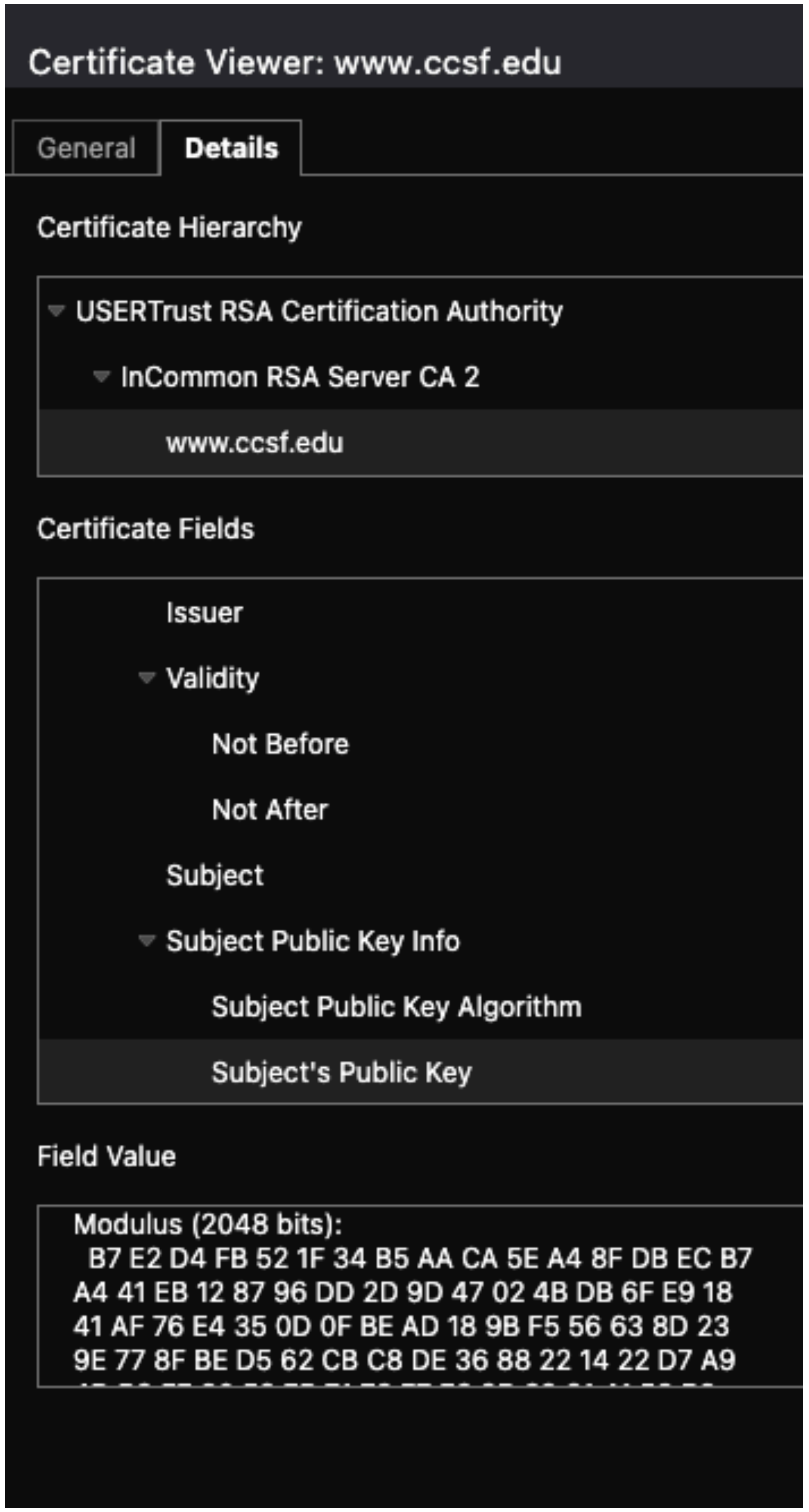
Common Name (CN)	InCommon RSA Server CA 2
Organization (O)	Internet2
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Sunday, March 10, 2024 at 5:00:00 PM
Expires On	Friday, April 11, 2025 at 4:59:59 PM

SHA-256 Fingerprints

Certificate	3ef4649ec5f0c4a9220ea58e41f0e55836e4d87689ee3ba5392 024a842758c1e
Public Key	246c20d83e5f5a0e82edd92f3280f77c7887aca663c82dba36c6 1c5ab9ed1c33



Certificate Viewer: www.ccsf.edu

General | Details

Certificate Hierarchy

- USERTrust RSA Certification Authority
  - InCommon RSA Server CA 2
    - www.ccsf.edu

Certificate Fields

- Issuer
- Validity
  - Not Before
  - Not After
- Subject
- Subject Public Key Info
  - Subject Public Key Algorithm
  - Subject's Public Key

Field Value

```
Modulus (2048 bits):
  B7 E2 D4 FB 52 1F 34 B5 AA CA 5E A4 8F DB EC B7
A4 41 EB 12 87 96 DD 2D 9D 47 02 4B DB 6F E9 18
41 AF 76 E4 35 0D 0F BE AD 18 9B F5 56 63 8D 23
9E 77 8F BE D5 62 CB C8 DE 36 88 22 14 22 D7 A9
```

Reverse DNS

- Finds a DNS name from an IP address
- Limited value
    - Some IPs host many websites

Let's Encrypt

- Provides free Domain Validation (DV) certificates
- There are more expensive certificate types:
  - Organization Validation (OV)
  - Extended Validation (EV)
  - They verify the owner's identity and reputation
  - But the same cryptographic security

Downgrade Attacks

- Force communication to use HTTP instead of HTTPS
- Or a lower TLS version
- Or a less-secure encryption method

Preventing Downgrade Attacks

- Configure server to accept only secure encryption methods
- Accept blocking old clients
- Redirect HTTP requests to HTTPS
- Restrict cookies to HTTPS only (secure)
- Include **Strict-Transport-Security** directive in your HTTPS response
    - Tells browser this site uses HTTPS only

The Same Origin Policy

SOP (Same Origin Policy)

- Browsers isolate resources from different websites
  - So they can't interfere with one another
- For scripts and cookies to be used on a page:
  - Domain name and port number must match the URL

Reach Out

- SOP means no other site can **reach in** to read cookies on another site
- But every site can **reach out** and include content from another site

Web Cookies

Cookies

- Small data strings stored in the browser
- Automatically included in every request to the server that set them
- Allows
  - Custom web pages
  - Remembering items in a shopping cart
  - After login, requests are authenticated with a **session cookie**
  - Targeted ads

Cookies are Client-Side

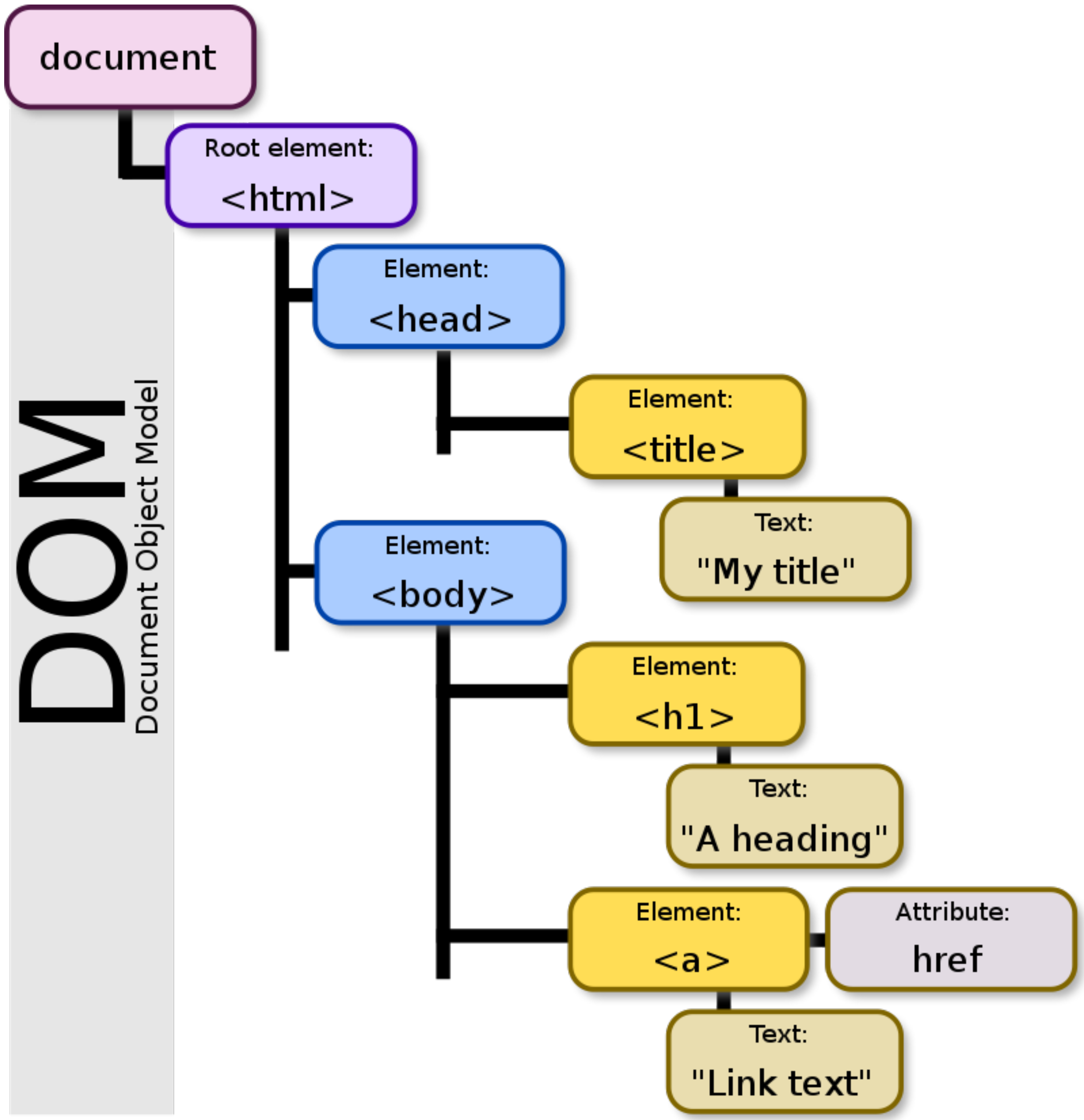
- The user can manipulate them
- The server must not trust important data from a cookie
  - Like prices
- Unless the cookie has cryptographic protections
  - Like a signed JWT

Cookies and SOP

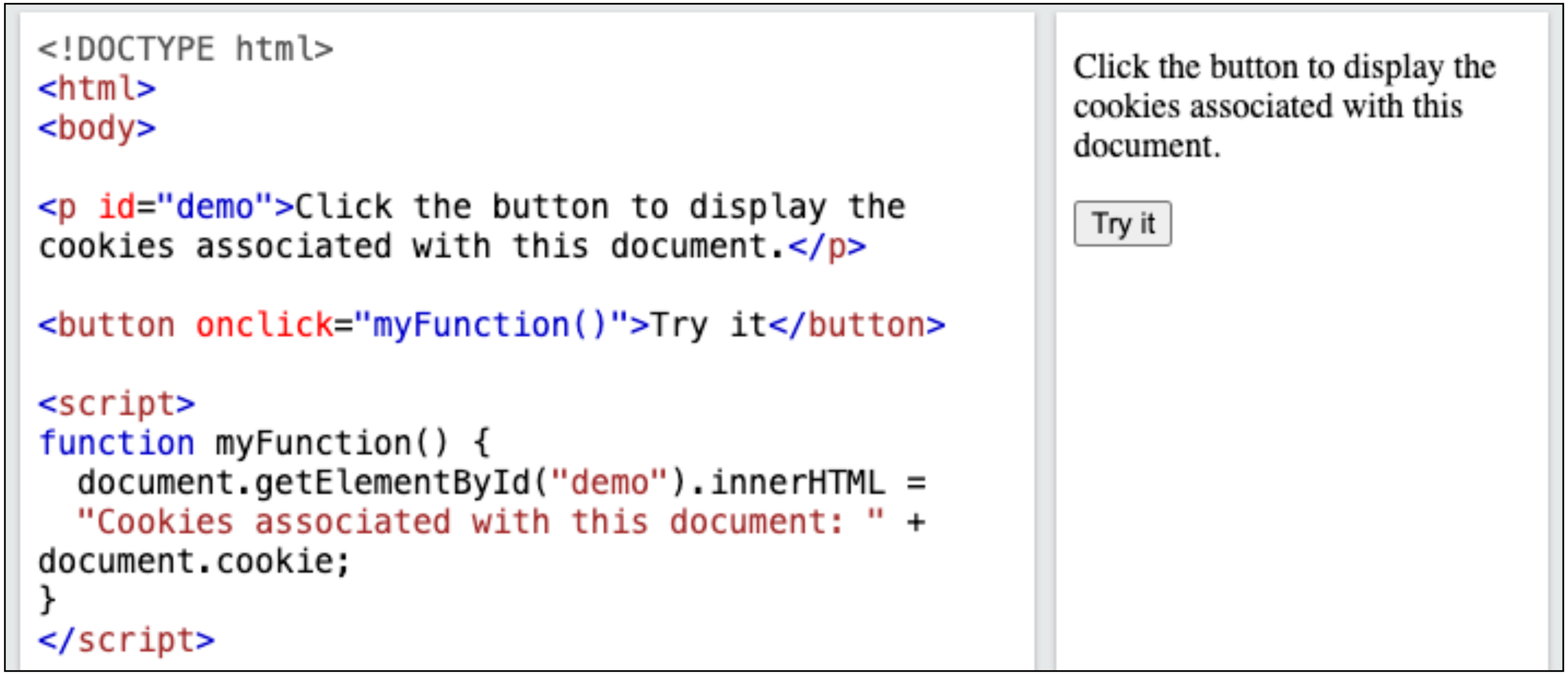
- A cookie set by **example.com**
  - is visible to subdomains **cat.example.com** and **dog.example.com**
- But a cookie set on **cat.example.com**
  - is not visible to **example.com** or **dog.example.com**

DOM

- A way to reference elements of a Web page
- Used mainly by JavaScript
- Image from Wikipedia
- By Birger Eriksson - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=18034500



DOM Example



```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to display the
cookies associated with this document.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML =
  "Cookies associated with this document: " +
document.cookie;
}
</script>
```

Click the button to display the cookies associated with this document.

- https://www.w3schools.com/js/tryit.asp?filename=tryjs_doc_cookie

JavaScript and Cookies

- JavaScript can reference cookies through the DOM
  - As allowed by the same origin policy
- Unless the cookies are set with the **httponly** attribute
- This prevents injected scripts from stealing cookies

- The **secure** attribute tells the browser to only transmit the cookie over HTTPS

CORS

- HTML5 introduced Cross-Origin Resource Sharing (CORS)
  - Can loosen the Same Origin Policy to enable data sharing with trusted websites
- Web Storage API
  - A way for sites to store data without using cookies

Common Web Vulnerabilities

Cross-Site Scripting (XSS)

XSS Example

- User submits a color parameter

```
https://www.example.com/page?color=green
```

- Page applies that color to some text

```
<h1 style="color:green">This is colorful text.</h1>
```

vulnerable code

```
query_params = urllib.parse.parse_qs(self.parts.query)
color = query_params.get('color', ['black'])[0]
h = '<h1 style="color:%s">This is colorful text.</h1>' % color
```

XSS Example

- This is the attack string

```
https://www.example.com/page?color=orange"><SCRIPT>alert("Gotcha!")</SCRIPT><span
%20id="dummy
```

- Resulting HTML

```
<h1 style="color:orange">  <SCRIPT>alert("Gotcha!")</SCRIPT>  <span id="dummy">This
is colorful text.
</h1>
```

XSS Countermeasures

- Web frameworks may contain XSS protection
- Avoid inserting data from user into output code
- Select from a list of known good values instead

Cross-Site Request Forgery (CSRF)

Content from Other Domains

- Web pages often include content from other domains
  - Ads
  - Photos
  - Analytics links
  - etc.
- The Same Origin Policy allows this data
- But isolates the content from the rest of the page
- Both POST and GET are allowed to other domains
  - Including the cookies for that other domain
- But the response can't be seen by the main web page

Anti-CSRF Token

- Include an unpredictable hidden value on the form
- CSRF attackers can't guess the token value
- Valid requests must come from a browser viewing the expected source page
- Derive the token from the session cookie to ensure it can only be used in that session

```
<form action="/ballot" method="post"> <label

for="name">Voting for</label> <input type="text" id="name"

name="name" value=""/> <input type="hidden" name="csrf_token"

value="mGEyoi1wE6NBWCyhBN9IZdEmaJLQtrYxi0J23XuXR4o="/> <input type="submit"

value="Vote"/>

</form>
```

SameSite

- Cookies can have the **SameSite=Strict** attribute
- Blocks sending cookies from any other domain
- But this is a client-side request
    - Cannot be trusted

More Vulnerabilities and Mitigations

Security Recommendations

- Use HTTPS and a quality framework
- Don't disable protection features in the framework

- **Don’t let attackers inject untrusted inputs into HTTP headers (similar to XSS).**
- **Specify accurate MIME content types to ensure that browsers process responses correctly.**
- **Open redirects can be problematic: don’t allow redirects to arbitrary URLs.**
- **Only embed websites you can trust with `<IFRAME>`. (Many browsers support the `X-Frame-Options` header mitigation.)**
- **When working with untrusted XML data, beware of XML external entity (XXE) attacks.**
- **The CSS `:visited` selector potentially discloses whether a given URL is in the browser history.**

Content-Security-Policy

- A response header
- Reduces exposure to XSS
- SpecIfies authorized sources for scripts, images, etc.
- BUT it relies on the browser to implement it

Referer

- Can leak information to a server
- Allows it to access the source URL in the DOM as **window.opener**
- Block this behavior with these attributes on the <a>, <area>, or <form> tag that sends the request to the external site
  - **rel="noreferrer"**
  - **rel="noopener"**
- Implemented on the client-side

HTTP Response Headers

- **Content-Security-Policy**
  - Blocks XSS
- **Referrer-Policy**
  - Controls referer header information
- **Strict-Transport-Security**
  - Loads page over HTTPS only, never HTTP
- **X-Content-Type-Options**
  - Tells browser to trust the Content Type header
  - Prevents MIME type sniffing (guessing)
- **X-Frame-Options**
  - Controls whether the page renders in an <iframe>, <frame>, <embed>, or <object>

Kahoot!

Ch 11