EECS 388

Introduction to Computer Security

Lecture 7:

The Web Platform

Sep 19, 2023

Prof. Ensafi

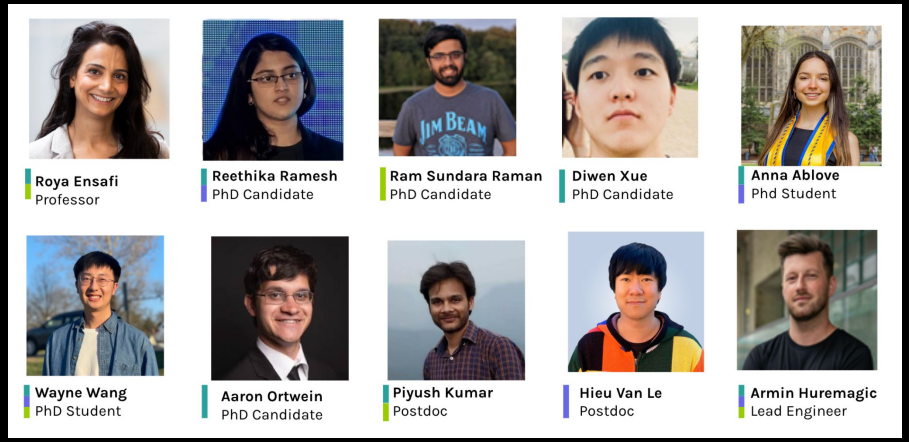
Who Are We?

Prof. Roya Ensafi

Web: https://ensa.fi
Mail: ensafi@umich.edu
Office: 4745 Beyster

The Censored Planet Lab

My team build scalable techniques and systems to protect users from adversarial networks that violate the **confidentiality, integrity,** or **availability** of users' legitimate traffic.



Censored Planet Observatory

Censored Planet

About ▾ Research ▾ Events Dashboard Data ▾ Log In

An Internet-wide, Longitudinal Censorship Observatory

Censored Planet is a censorship measurement platform that collects data using multiple remote measurement techniques in more than 200 countries.

Reports → Data Dashboard

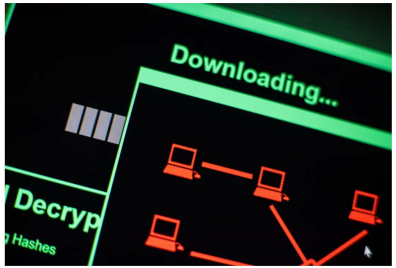
Censored Planet Rapid Response

Censored Planet team has exposed **significant new government censorship tactics**, and our results have been highlighted in more than 100 popular press articles.

Google, Apple and Mozilla to block internet surveillance in Kazakhstan

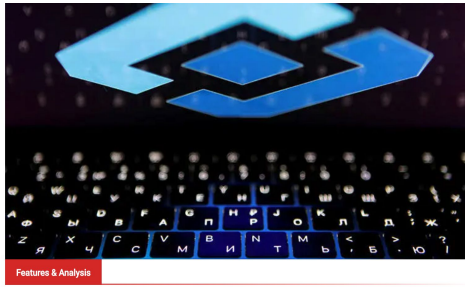
It's a response to the government's attempt to intercept users' browser data.

Oscar Gonzalez
Aug. 21, 2019 7:02 a.m. PT



The makers of the most popular browsers are taking a stand against the Kazakh government.

Picture Alliance/Getty Images



Features & Analysis

Laws, cheap web filters arm Russia to block news, says Censored Planet

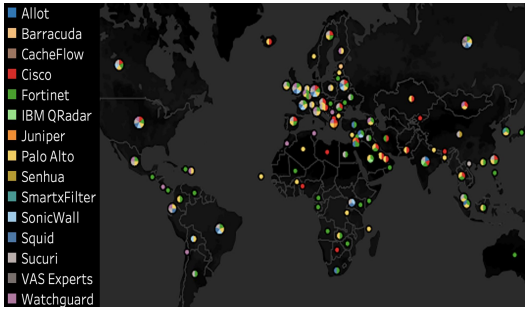
By Madeline Earp/CPJ Consultant Technology Editor on November 7, 2019 11:36 AM EST

When Daniil Kislov tried to view the website of *Fergana* from his computer in Moscow on November 1, his browser showed

STORIES

Roskomnadzor successfully slows down Twitter. American researchers explained how he did it. They even found a small loophole for users - it's a pity that it's unlikely to help them

01:36, April 8, 2021
Source: Meduza



Real-time monitor tracks the growing use of network filters for censorship

February 21, 2020

The team says their framework can scalably and semi-automatically monitor the use of filtering technologies for censorship at global scale.

VPNalyzer: Crowdsourced Investigation into Commercial VPNs

Are VPNs risky? Are VPNs protecting you? Is your VPN any good?

VPNalyzer aims to understand your needs as a VPN user and empower you with a tool to test various security and privacy aspects of your own VPN provider.

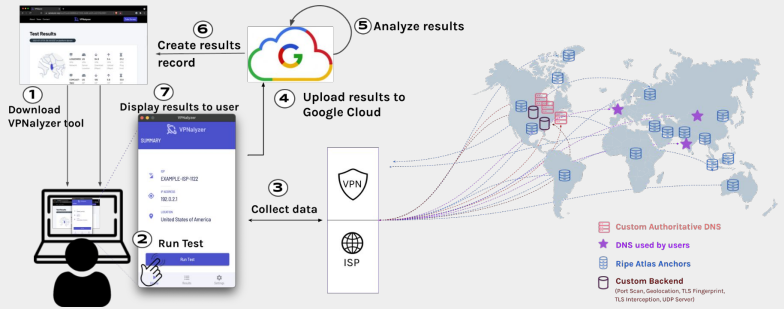
VPNalyzer in the News

Read our NEW Report!

VPNalyzer

Problem: VPN multibillion-dollar industry includes many snakeoil products, hyperbolic claims, is laxly regulated, and remains severely understudied.

VPNalyzer must adopt a modular, extensible test suite implementation, VPNalyzer must facilitate large-scale crowd-sourced measurements



VPNalyzer: Systematic Investigation of the VPN Ecosystem
R. Ramesh, L. Evdokimov, D. Xue, R. Ensafi, NDSS, Apr 2022

Understanding the user needs and considerations and VPN providers to bridge gaps and highlight (mis)aligned incentives

"All of them claim to be the best": Multi-perspective study of VPN users and VPN providers R. Ramesh, A. Vyas, R. Ensafi, USENIX 2023

The New York Times

Study: Russia's Web-Censoring Tool Sets Pace for Imitators

By The Associated Press

Nov. 6, 2019

WASHINGTON — Russia is succeeding in imposing a highly effective internet censorship regime across thousands of disparate, privately owned providers in an effort also aimed at making government snooping pervasive, according to a study released Wednesday.

FINANCIAL TIMES

US-China relations + Add to myFT

US blocks Hong Kong users from some government websites

Sites hosting economic data have been inaccessible to users in the Asian financial centre for months

ars TECHNICA SUBSCRIBE

THE REVOLUTION WILL NOT BE TWEETED —

Russia's Twitter throttling may give censors never-before-seen capabilities

BBC NEWS

Home Video World UK Business Tech Science

Technology

MIT Technology Review Subscribe

MS TECH

Q&A

Why you should be more concerned about internet shutdowns

WIRED SUBSCRIBE

CHRIS STOKEL-WALKER SECURITY APR 1, 2022 7:00 AM

Russia Inches Toward Its Splinternet Dream

For years, the country has been trying to create its own sovereign internet—a goal given new impetus by the backlash to its invasion of Ukraine.

Russia Is Censoring the Internet, With Coercion and Black Boxes


By Adam Satariano and Paul Mozur

zilla move to s
ooping'

The Economist

International Oct 16th 2021 edition ›

Blockheads

Governments are finding new ways to squash free expression online

Real-time monitor tracks the growing use network filters for censorship

MICHIGAN RADIO npr 91.7 Ann Arbor/Detroit 104.1 Grand Rapids 91.3 Port Huron 89.7 Lansing 91.1 Flint ♥ Donate

News

Research team investigating Internet censorship with tracking system

Michigan Radio | By Lauren Janes

Published February 6, 2019 at 4:38 PM EST

Forbes

CYBERSECURITY

Apple, Google And Mozilla Block Kazakh Government Surveillance

Emma Woollacott Senior Contributor ⓘ Follow

'Extremely aggressive' internet censorship spreads in the world's democracies

Roskomnadzor successfully slows down Twitter. American researchers explained how he did it. They even found a small loophole for users - it's a pity that it's unlikely to help them

01:36, April 8, 2021


SUMMIT FOR
DEMOCRACY

Web Security

This week:

- **The Web Platform**
- Web Attacks and Defenses

Next week:

- HTTPS and the Web PKI
- HTTPS Pitfalls

Later in the course:

- User Authentication
- Privacy and Online Tracking

Web Platform Security

The **Web platform** is the collection of technologies developed as **open standards** that powers Web sites and applications.

Open: *Anyone* can build a Web browser or server, participate in standard design.
Standards: **URLs, HTML**, **JavaScript**, **HTTP**, **TLS**, etc.

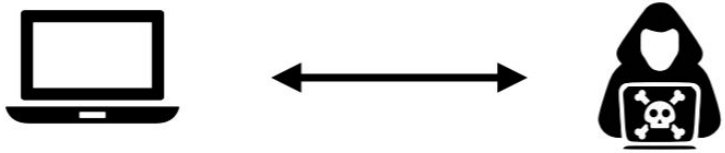
Web security goals:

- **Protect users from malicious sites and networks**
- **Isolate sites from each other within the browser:**

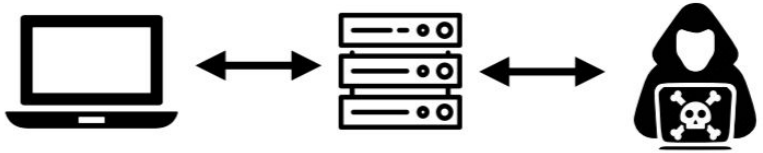
  Integrity: Site A cannot **affect user's session** on Site B
  Confidentiality: Site A cannot **steal user's data** from Site B

Attack Models

Malicious Website

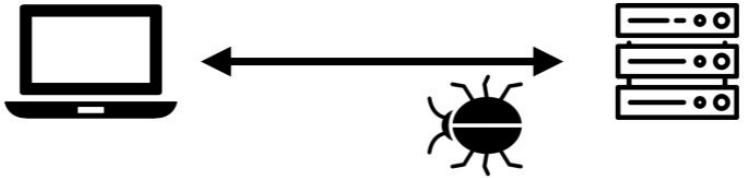


Malicious External Resource



Network Attacker

Malware Attacker

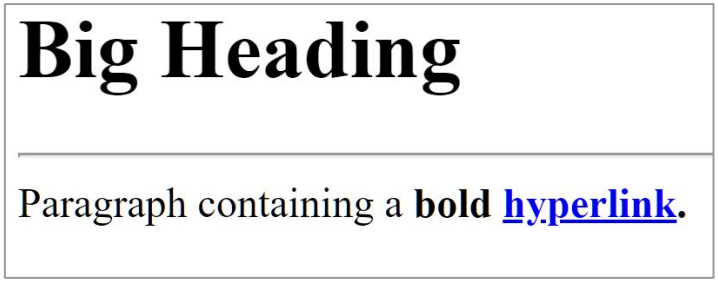


HTML Basics

HTML (Hypertext Markup Language) is a standard language for documents displayed in a browser. Defines document structure and hints at appearance.

Built from nested **elements** specified using **<tags>**, which may have **attributes=""**.

```
<html>
  <!-- comment -->
  <head><title>Page Title</title></head>
  <body>
    <h1>Big Heading</h1>
    <hr/><!-- horizontal rule (self-closing tag) -->
    <p>
      Paragraph containing a <b>bold <a href="https://example.com/">hyperlink</a>.</b>
    </p>
  </body>
</html>
```



Caution: Some literal characters must be **escaped** or they may be misinterpreted as part of the HTML markup.

Replace `<` `>` `&` `"` `'` with `<` `>` `&` `&quot;` `&apos;`

JavaScript and the DOM

JavaScript running in the browser can read and modify page content. The **document object model (DOM)** and other APIs provide a standard programming interface.

```
<html><body>
  <input id="textbox"/><button id="add">+</button>
  <ul id="todo"/>
  <script>// embedded JavaScript:
    function addTodo(text) {
      var newitem = document.createElement('li');
      newitem.appendChild(document.createTextNode(text));
      document.getElementById('todo').appendChild(newitem);
    }
    document.getElementById('add').onclick = function(e) {
      addTodo(document.getElementById('textbox').value);
    }
  </script>
</body></html>
```

Pages can also include scripts loaded from a separate URL (called a **subresource**):

```
<script src="https://code.jquery.com/jquery-3.4.1.js"
 integrity="sha256-WpOohJOqMqqyKL9FccASB9O0KwACQJpFTUBLTYOVvVU="></script>
```

The hash is a security mechanism called **subresource integrity**. [*Why use it?*]

URLs (Uniform Resource Locators)

A **URL** is a string specifying a unique resource on the Web.

`scheme://host:port/path?query#fragment`

scheme: protocol used to access resource
- `https` (HTTP with end-to-end encryption)
- `http` (plaintext HTTP — unsafe!)

host: server's domain name or IP address
- `eecs388.org` (DNS name)
- `141.212.118.72` (IPv4 address)
- `[2607:f018:600:8::20]` (IPv6 address)

port: TCP port (default 443 for HTTPS and 80 for HTTP)

path: identifies resource to server
- `/papers/index.html` (format up to server)

query: parameter string passed to server
- `?key1=value1&key2=value2` (key-value pairs)
- `?7265de6a2c4c8068ed3e75` (arbitrary string)

fragment: parameter string visible only to client
- `#Section4` (tells browser to scroll to named location)

Paths, Queries, Fragments: % Encoding

Most punctuation and non-ASCII characters must be **escaped** by encoding each UTF-8 byte as `%` followed by two hex characters.

" "→ "%20" "<" → "%3C" ">" → "%3E"

https://google.com/?q=hello%20world

Hosts: IDN Encoding

The host field uses a different encoding for internationalized names called **IDN encoding**.

Portions of the name are replaced by `xn--` followed by a Punycode encoding, e.g.:

https://xn--ls8h.la/

HTTP Protocol

Hypertext Transport Protocol (HTTP) allows fetching individual resources, such as HTML documents.

Structured as a sequence of **requests** and **responses** (not as a stream of data).

Client sends:

- **Method**
  GET (retrieve data; shouldn't change server state)
  POST (can submit data; causes change or side-effect)
- **Path and query**
- Headers

Server returns:

- **Response code**
  200 OK, 302 Redirect, 404 Not Found, ...
- Headers
- **Content data** (arbitrary bytes)

User follows a link from Google to https://example.com/index.html

HTTP 1.1 Request

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0)
Referer: https://www.google.com/
```

HTTP 1.1 Response

```
HTTP/1.1 200 OK
Server: Nginx
Content-Type: text/html
Last-Modified: Fri, 13 Sep 2019 14:27:25 GMT
Set-Cookie: …
Content-Length: 13429

<html><body> …
```

HTTP Cookies

A **cookie** is a piece of data that a server sends to the browser. The browser may store it and return it in later requests to the same server.

Used for:

- **Maintaining session state** across HTTP requests (logins, shopping carts, etc.)
- **Personalization** (storing preferences)
- **Tracking** user behavior on or across sites

Can also be read and set by JavaScript on page.

Clients can *read*, *change*, or *erase* cookie data!

Two general approaches for using them securely:

1. Store cryptographically protected data.
2. Use a unique, random, unguessable session identifier that's tied to a server-side database.

Initial HTTP Request

HTTP Response (server sets cookie)

```
HTTP/1.1 200 OK
Server: Nginx
Content-Type: text/html
Set-Cookie: trackingID=6bb4ad8baf953b6f;
      Domain=reddit.com; Path=/; Secure
```

Later HTTP Requests (brower returns cookie)

```
GET /r/uofm HTTP/1.1
Cookie: trackingID=6bb4ad8baf953b6f
User-Agent: Mozilla/5.0 (Windows NT 10.0)
```

Browser Execution Model

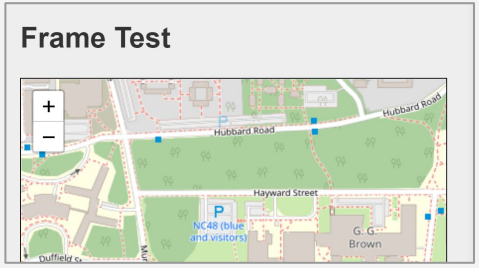
When **loading a document**, the browser:

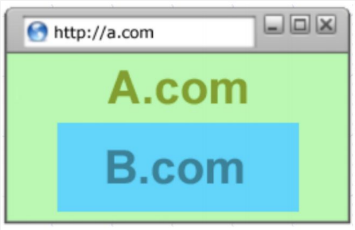
1. Loads content at URL
2. Parses HTML and runs any inline JavaScript code
3. Recursively fetches and renders subresources (JavaScript, images, CSS, frames)

After loading:

Calls JavaScript functions in response to user inputs, timeouts, other events

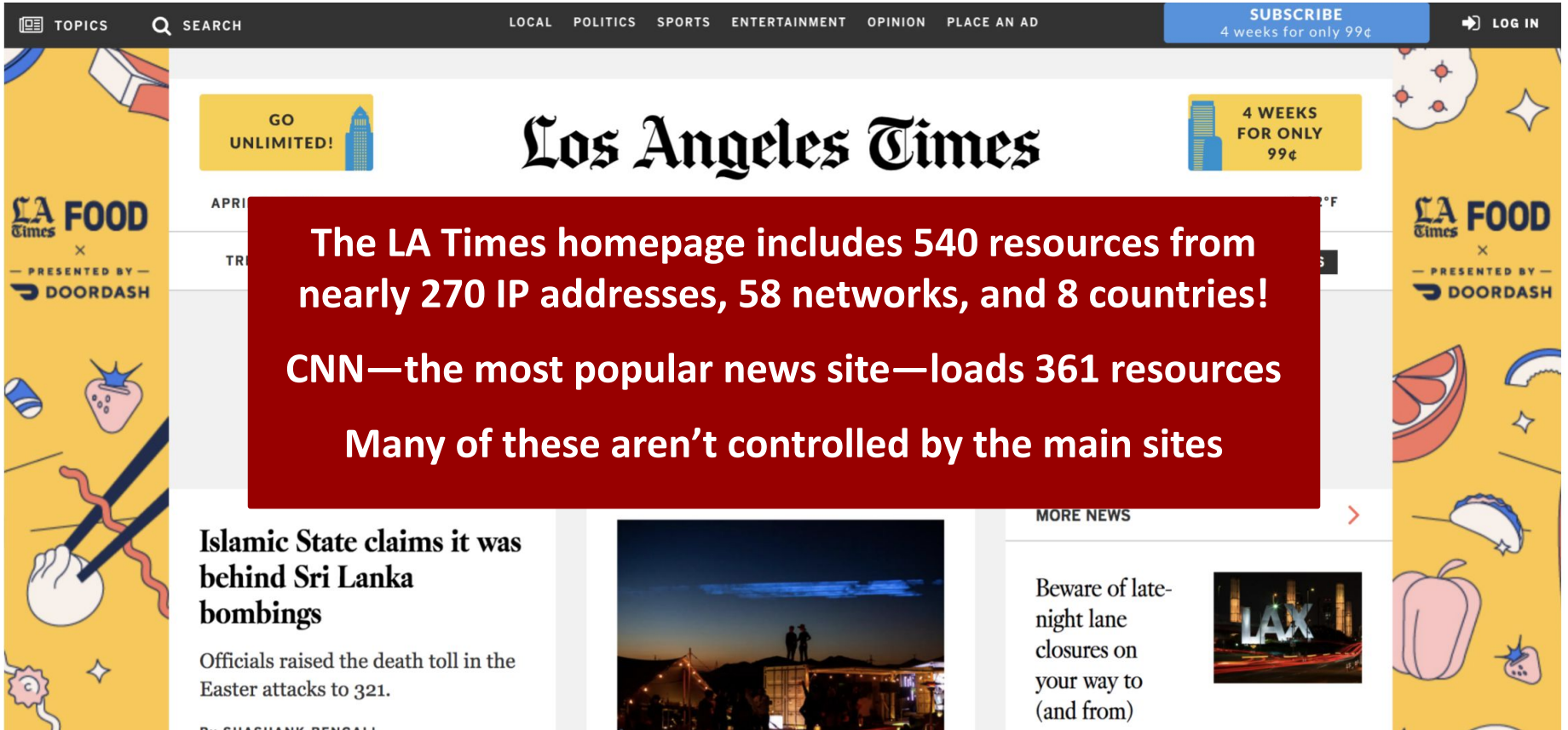
Document can also include **frames**, which display another HTML page. Browsers isolate frames from the parent document.



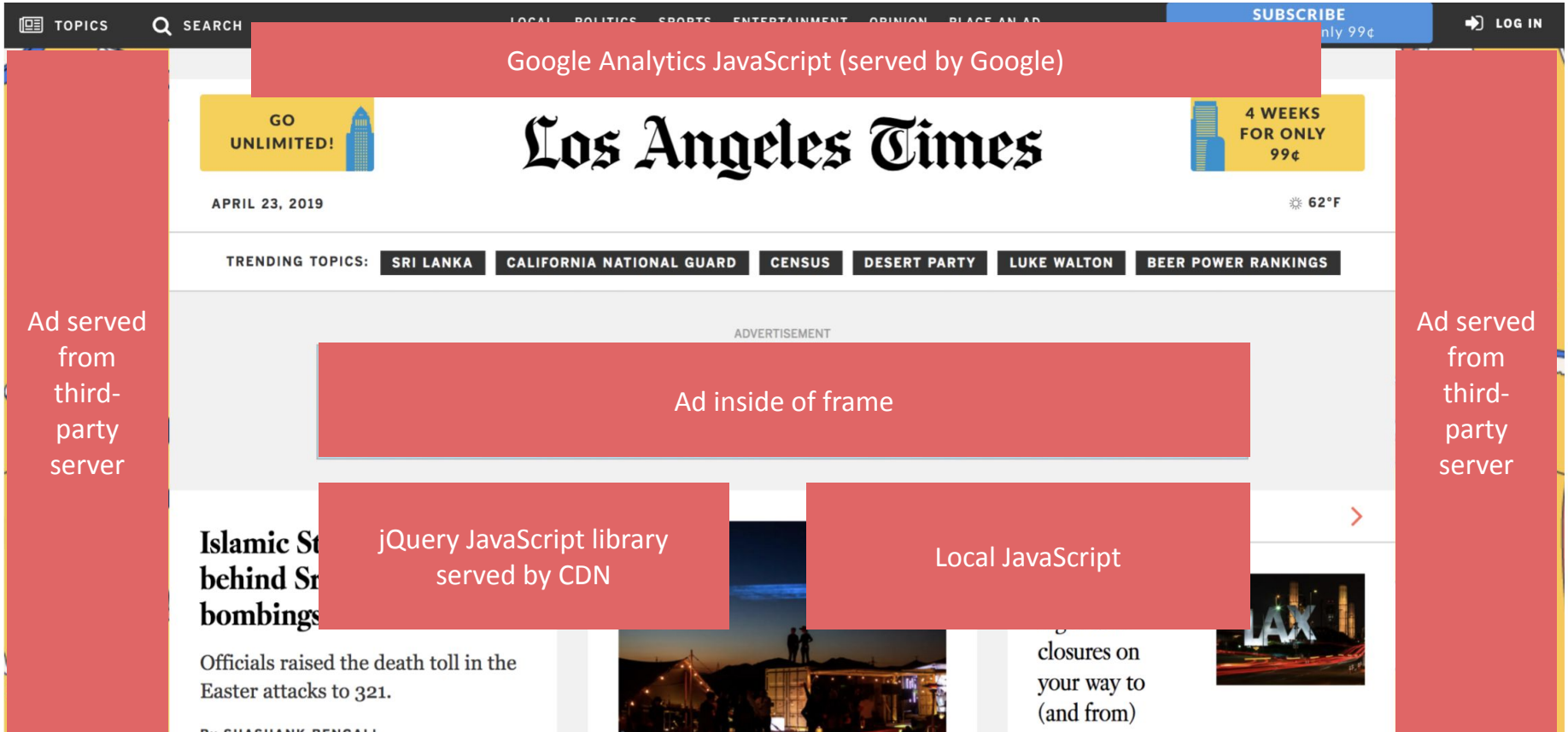



```
<h1>Frame Test</h1>
<iframe width="425" height="350"
src="https://www.openstreetmap.org/export
/embed.html?bbox=-83.71982216835023%2C42.
291433288630564%2C-83.71388375759126%2C42
.294587952588785"></iframe>
```

A Modern Website



A Modern Website



Same-Origin Policy

Essential security question:

When can one site access data contained in another site?

Example:

If you visit **attacker.com**, what stops it from reading your Gmail messages?

What if **attacker.com** loads Gmail in a frame or runs JavaScript files from **gmail.com**?

Browsers enforce isolation between sites by applying **Same-Origin Policy** (**SOP**).

The SOP separates content into different trust domains ("**origins**") and restricts data flows between them.

What defines an origin?

scheme://domain:port

example: **https://eecs388.org:443**

What's isolated?

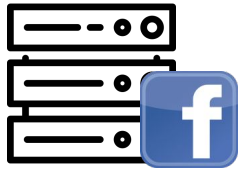
Each origin has local client-side resources that are protected:

- Cookies (local state)
- DOM storage
- DOM tree
- JavaScript namespace
- Permission to use local hardware (e.g., camera or GPS)

SOP: Cross-Origin Image

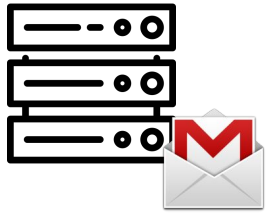
GET / HTTP/1.1
Host: facebook.com

facebook.com



HTTP/1.1 200 OK
…

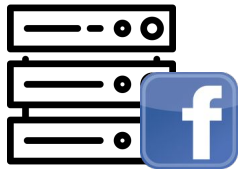
gmail.com



SOP: Cross-Origin Image

GET / HTTP/1.1
Host: facebook.com

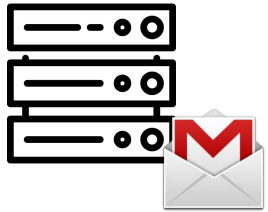
facebook.com



HTTP/1.1 200 OK
…

GET /img.png HTTP/1.1
Host: gmail.com

gmail.com



HTTP/1.1 200 OK
…
<89>PNG^M …

Cross-Origin Images are Allowed

Show on screen, but script can't read the pixels with <canvas>.

SOP: Fetching Cross-Origin Data with JS

GET / HTTP/1.1
Host: facebook.com

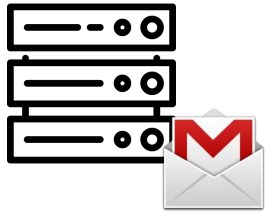
(evil!)
facebook.com

HTTP/1.1 200 OK
…
<script>
$.get('http://gmail.com/msgs.json',
function (data) { alert(data); }
</script>

$.get('http://gmail.com/msgs.json',
function (data) { alert(data); }

GET /msgs.json HTTP/1.1
Host: gmail.com

gmail.com



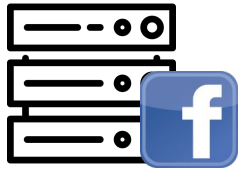
HTTP/1.1 200 OK
…
{ new_msgs: 3 }

Blocked by SOP
facebook.com JS can't read data from gmail.com

SOP: Fetching Cross-Origin Data with JS

GET / HTTP/1.1
Host: facebook.com

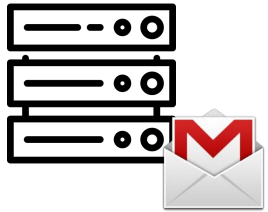
facebook.com



HTTP/1.1 200 OK
…
<script src="http://gmail.com/chat.js"/>

?

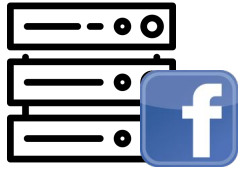
gmail.com



SOP: Fetching Cross-Origin Data with JS

GET / HTTP/1.1
Host: facebook.com

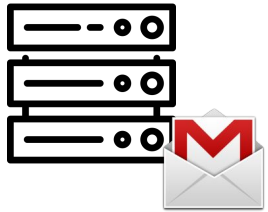
facebook.com



HTTP/1.1 200 OK
…
<script src="http://gmail.com/chat.js"/>

GET /chat.js HTTP/1.1
Host: gmail.com

gmail.com



HTTP/1.1 200 OK
…
$.get('http://gmail.com/chat.json', function(data){ … });

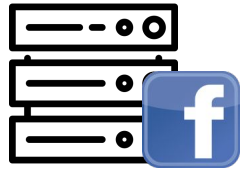
GET /chat.json HTTP/1.1
Host: gmail.com

Blocked by SOP
Script came from gmail.com but the origin is still facebook.com

SOP: A Public API with CORS

GET / HTTP/1.1
Host: facebook.com

facebook.com



$.get('http://gmail.com/chat.json',
function (data) { alert(data); }

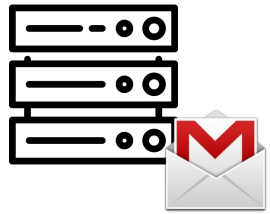
OPTIONS /chat.json HTTP/1.1
Host: gmail.com
Origin: https://facebook.com

Cross-Origin Resource Sharing (CORS) "Preflight" request

Server "opts out" of SOP enforcement for this resource

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

gmail.com



Allowed via CORS
Server has granted special permission to bypass SOP

GET /chat.json HTTP/1.1
Origin: https://facebook.com

SOP: Cookies

Cookie Scope

Sending cookies only to the right websites is really important!

Frequent source of problems:
Cookies use a different scope than DOM

DOM:
Scoped based on (scheme, host, port)

Cookies:
Scoped based on ([scheme], domain*)

* Complicated rules shown in later slides

Cookie Security Attributes

Weakness: By default, cookies set over HTTPS will also be sent on requests over HTTP (and visible to network eavesdroppers).

Solution: Server can set "`Secure`" attribute to limit cookie to HTTPS requests.

```
Set-Cookie: id=a3fWa; Secure;
```

Weakness: By default, cookies can be read by any JavaScript running in the origin.
E.g., if **bank.com** includes Google Analytics script, Google can read authentication cookies.

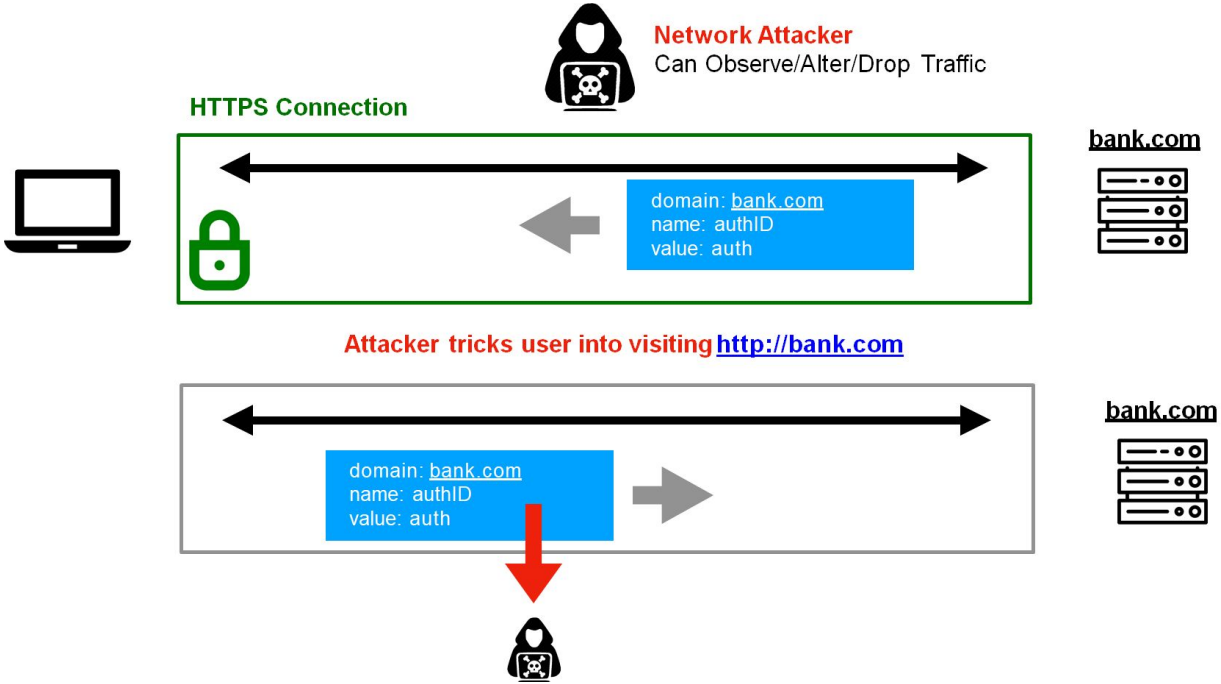
Solution: Server can set "`HttpOnly`" attribute to prevent cookie from being accessed by DOM.

```
Set-Cookie: id=a3fWa; Secure; HttpOnly;
```

SOP: Cookies

Cookie Security Attributes

Weakness: By default, cookies set over HTTPS will also be sent on requests over HTTP (visible to network eavesdroppers).



Rules for Setting and Reading Cookies

Setting a cookie:

A site can set a cookie for its own domain or any *parent domain*, as long as the parent domain is not a **public suffix**.

Example: **login.site.com** attempts to set cookie

login.site.com	allowed
site.com	allowed
other.site.com	prohibited
different.com	prohibited
com	prohibited (public suffix)

Caution: You don't know which domain set a cookie when you receive it.

Example: **club.eecs.umich.edu** can set cookies for **umich.edu** (since latter isn't a public suffix).

Rules for Setting and Reading Cookies

Setting a cookie:

A site can set a cookie for its own domain or any *parent domain*, as long as the parent domain is not a **public suffix**.

Example: **login.site.com** attempts to set cookie

login.site.com	allowed
site.com	allowed
other.site.com	prohibited
different.com	prohibited
com	prohibited (public suffix)

Caution: You don't know which domain set a cookie when you receive it.

Example: **club.eecs.umich.edu** can set cookies for **umich.edu** (since latter isn't a public suffix).

Reading a cookie:

A site can read cookies set for its own domain **or any parent domains**.*

Example: **login.site.com** can read cookies for

login.site.com	yes
site.com	yes
other.site.com	no
different.com	no

Caution: Cookies also specify a *path* on the site, but (without `HttpOnly`) it's for efficiency only. DOM origins aren't isolated by path, so scripts can read cookies set for any path in the origin.

Suppose cookie set for **x.com/b**. Then **x.com/a** can do: `alert(frames[0].document.cookie);`

* If **domain=** attribute is *unset*, some browsers disallow reading by subdomains, but can't rely on this behavior.

When are these cookies sent?

Cookie 1:
name = mycookie
value = mycookievalue
domain = login.site.com
path = /

Cookie 2:
name = cookie2
value = mycookievalue
domain = site.com
path = /

Cookie 3:
name = cookie3
value = mycookievalue
domain = site.com
path = /my/home

	Cookie 1	Cookie 2	Cookie 3
checkout.site.com			
login.site.com			
login.site.com/my/home			
site.com/my			

When are these cookies sent?

Cookie 1:
name = mycookie
value = mycookievalue
domain = login.site.com
path = /

Cookie 2:
name = cookie2
value = mycookievalue
domain = site.com
path = /

Cookie 3:
name = cookie3
value = mycookievalue
domain = site.com
path = /my/home

	Cookie 1	Cookie 2	Cookie 3
checkout.site.com	No	Yes	No
login.site.com	Yes	Yes	No
login.site.com/my/home	Yes	Yes	Yes
site.com/my	No	Yes	No

Coming Up

Reminders:

Crypto Project, Part 2 due Thursday at 6 PM

Wednesday

Web Attacks & Defenses

XSS
CSRF
SQL-injection

Next week

HTTPS

TLS and the CA ecosystem
Attacks and defenses