CSE 484 / CSE M 584: Computer Security and Privacy

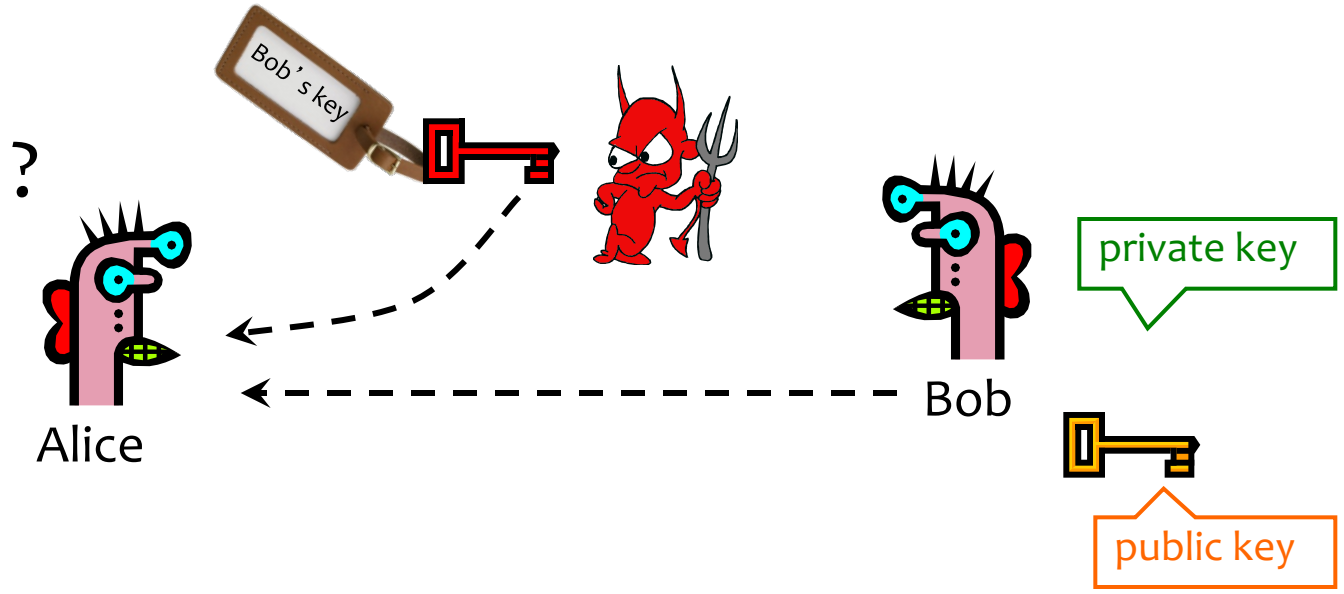
Web Security
[Overview + Browser Security Model]

Spring 2019

Franziska (Franzi) Roesner
franzi@cs.washington.edu

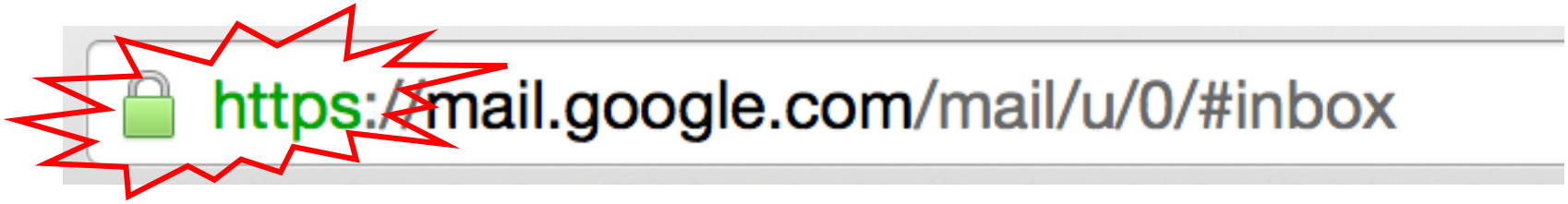
Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Recall: Authenticity of Public Keys



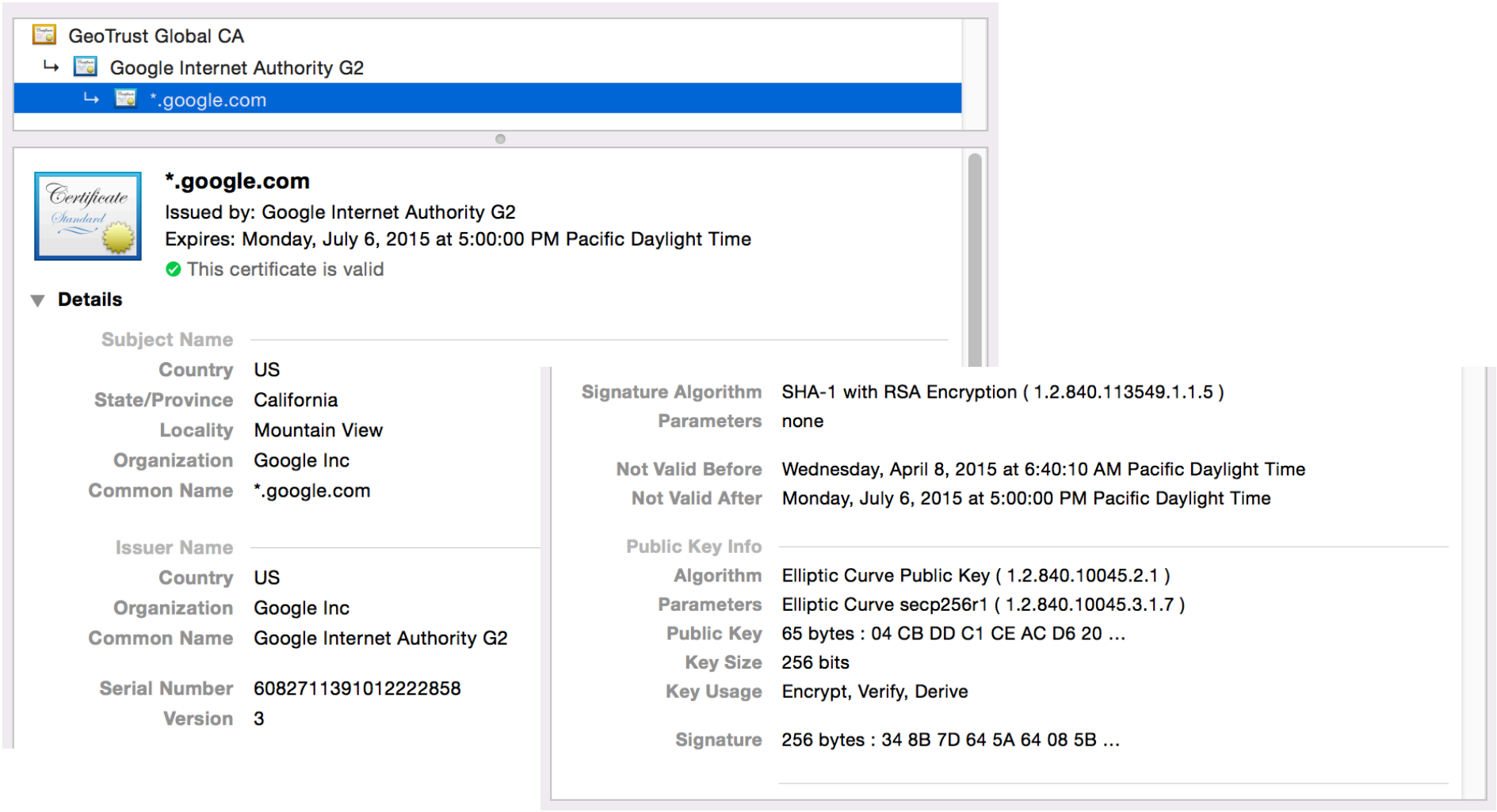
Problem: How does Alice know that the public key she received is really Bob's public key?

You encounter this every day…



SSL/TLS: Encryption & authentication for connections

Example of a Certificate



Many Challenges…

- Hash collisions
- Weak security at CAs
  - Allows attackers to issue rogue certificates
- Users don't notice when attacks happen
  - We'll talk more about this later in the course
- Etc…

Attacking CAs

DigiNotar Hacked by Black.Spook and Iranian Hackers Posted by Mikko @ 09:05 GMT | Comments

DigiNotar is a Dutch Certificate Authority. They sell SSL certificates.

Somehow, somebody managed to get a rogue SSL certificate from them on **July 10th, 2011**. This certificate was issued for domain name **.google.com**.

What can you do with such a certificate? Well, you can impersonate Google — assuming you can first reroute Internet traffic for google.com to you. This is something that can be done by a government or by a rogue ISP. Such a reroute would only affect users within that country or under that ISP.

Security of DigiNotar servers:

- All core certificate servers controlled by a single admin password (Prod@dm1n)
- Software on public-facing servers out of date, unpatched
- No anti-virus (could have detected attack)

Consequences

- Attacker needs to first divert users to an attacker-controlled site instead of Google, Yahoo, Skype, but then…
  - For example, use DNS to poison the mapping of mail.yahoo.com to an IP address
- … "authenticate" as the real site
- … decrypt all data sent by users
  - Email, phone conversations, Web browsing

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*
  - (Then what?)
- **Approach:** auditable certificate logs

www.certificate-transparency.org

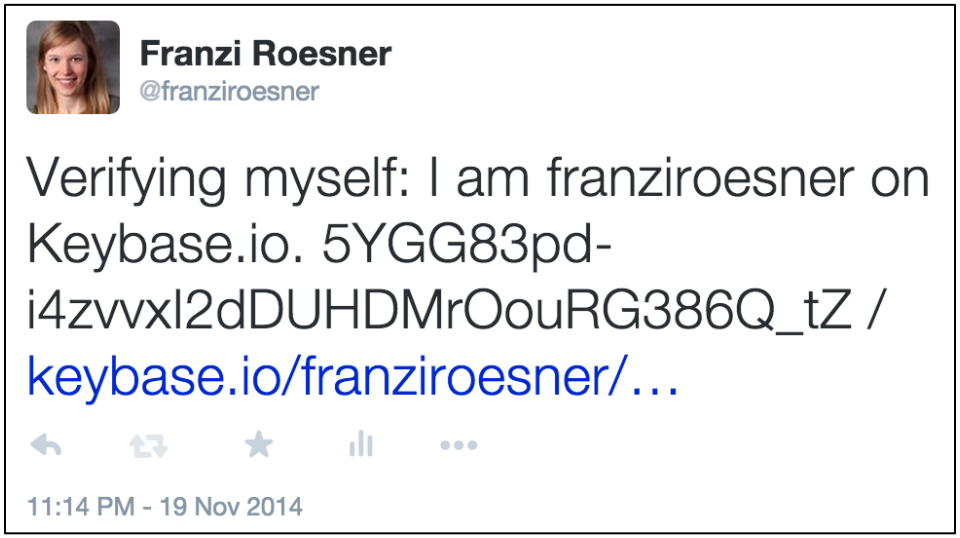
Attempt to Fix CA Problems:

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`"

Keys for People: Keybase

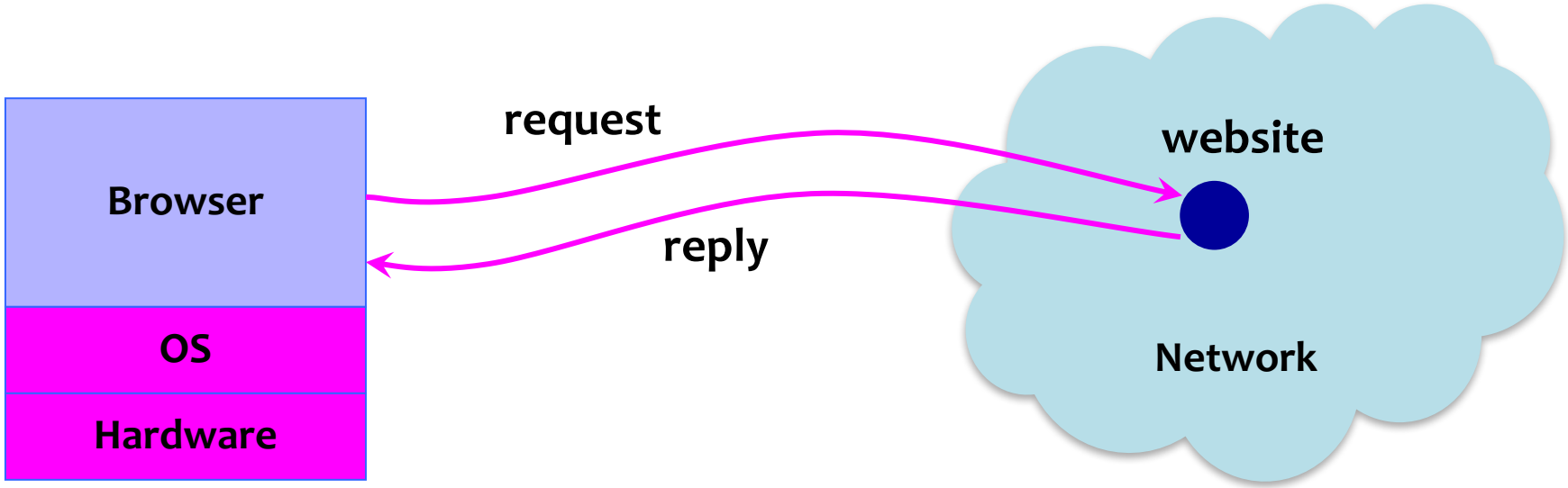
- Basic idea:
  - Rely on existing trust of a person's ownership of other accounts (e.g., Twitter, GitHub, website)
  - Each user publishes signed proofs to their linked account



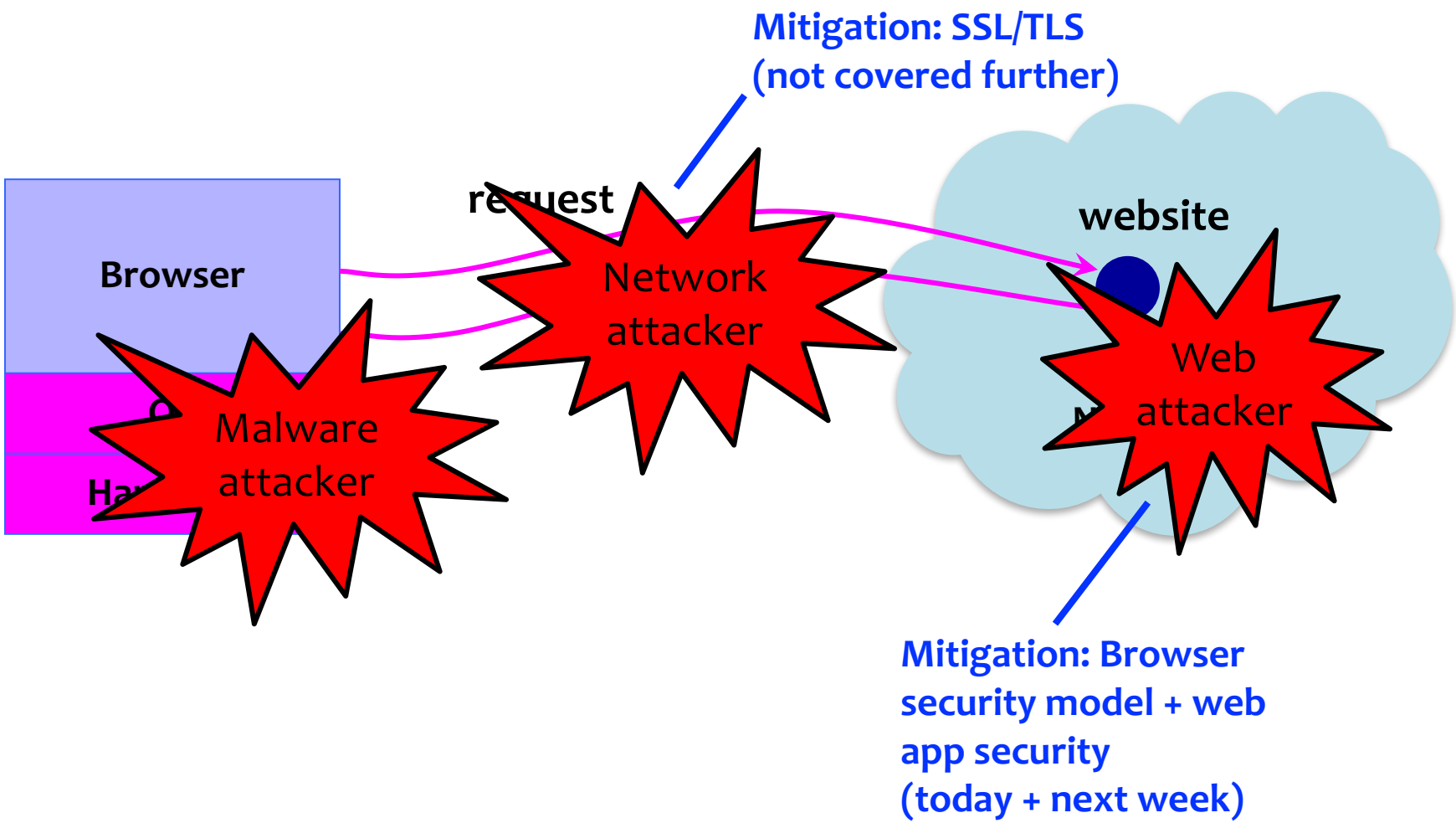
https://keybase.io/

Web+Browser Security

Big Picture: Browser and Network



Where Does the Attacker Live?



Web Attacker

- Controls a malicious website (**attacker.com**)
  - Can even obtain SSL/TLS certificate for site 

- User visits attacker.com – why?
  - Phishing email, enticing content, search results, placed by an ad network, blind luck …
- Attacker has no other access to user machine!
- Variation: good site **honest.com**, but:
  - An iframe with malicious content included
  - Website has been compromised

Two Sides of Web Security

(1) Web browser

- Responsible for securely confining content presented by visited websites

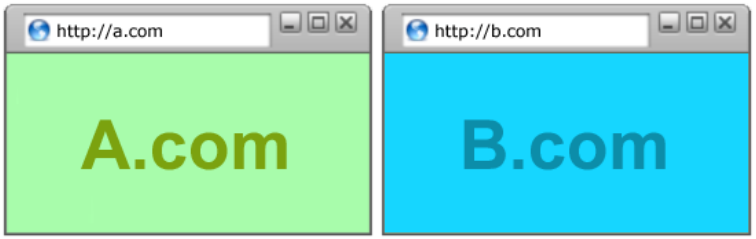
(2) Web applications

- Online merchants, banks, blogs, Google Apps …
- Mix of server-side and client-side code
  - Server-side code written in PHP, Ruby, ASP, JSP
  - Client-side code written in JavaScript
- Many potential bugs: XSS, XSRF, SQL injection

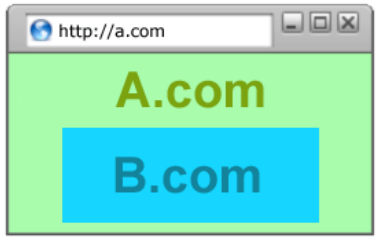
All of These Should Be Safe

- Safe to visit an evil website

- Safe to visit two pages at the same time



- Safe delegation



Browser Security Model

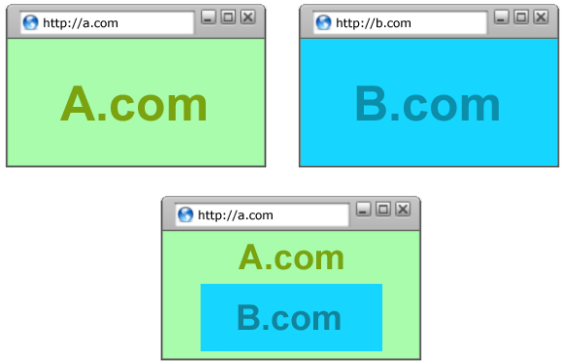
Goal 1: Protect local system from web attacker

→ Browser Sandbox

Goal 2: Protect/isolate web content from other web content

→ Same Origin Policy

(plus sandbox)



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 [1]
Sandbox Escape [5]	$15,000

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!

- Some examples of how messy it gets in practice…
- Browsers don't (or didn't) always get it right...

- We'll talk about:
  - DOM / HTML Elements
  - Navigation
  - Cookie Reading
  - Cookie Writing
  - Iframes vs. Scripts