

CSE 484 / CSE M 584: **Computer Security and Privacy**

Autumn 2019

Tadayoshi (Yoshi) Kohno
yoshi@cs.Washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Franz Roesner, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Announcements

- My office hours
 - 11/13 (Wed), 11:30am, CSE1 403
 - 11/20 (Wed), 2:30pm, CSE1 403
 - 11/27 (Wed), None
 - 12/4 (Wed), 12:30pm, CSE1 403
- HW 2 available (due 11/15)
- Lab2 out or soon to be out; to be discussed at quiz section this week (11/7)
- Final Project checkpoint on Friday (11/8) (group members, brief description)
 - https://courses.cs.washington.edu/courses/cse484/19au/assignments/final_project.html

Cryptography Summary Thus Far

- Goal: Privacy
 - Symmetric keys:
 - One-time pad, Stream ciphers
 - Block ciphers (e.g., DES, AES) → modes: EBC, CBC, CTR
 - Public key crypto (e.g., RSA)
- Goal: Integrity
 - MACs, often using hash functions (e.g, MD5, SHA-256)
- Goal: Privacy and Integrity
 - Encrypt-then-MAC
- Goal: Authenticity (and Integrity)
 - Digital signatures (e.g., RSA, DSS)

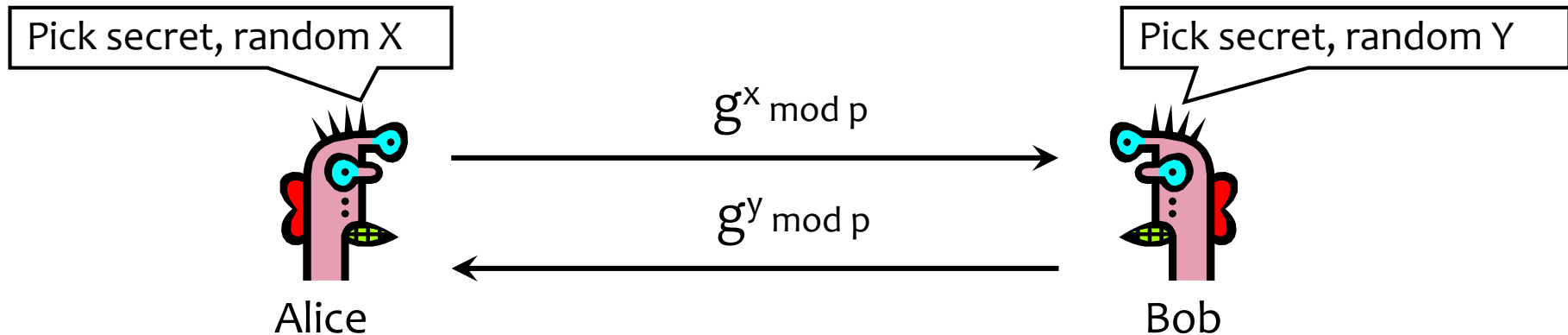
Session Key Establishment

Modular Arithmetic

- Given g and prime p , compute:
 $g^1 \bmod p, g^2 \bmod p, \dots, g^{100} \bmod p$
 - For $p=11, g=10$
 - $10^1 \bmod 11 = 10, 10^2 \bmod 11 = 1, 10^3 \bmod 11 = 10, \dots$
 - Produces cyclic group $\{10, 1\}$ (order=2)
 - For $p=11, g=7$
 - $7^1 \bmod 11 = 7, 7^2 \bmod 11 = 5, 7^3 \bmod 11 = 2, \dots$
 - Produces cyclic group $\{7, 5, 2, 3, 10, 4, 6, 9, 8, 1\}$ (order = 10)
 - $g=7$ is a “generator” of Z_{11}^*
 - For $p=11, g=3$
 - $3^1 \bmod 11 = 3, 3^2 \bmod 11 = 9, 3^3 \bmod 11 = 5, \dots$
 - Produces cyclic group $\{3, 9, 5, 4, 1\}$ (order = 5)

Diffie-Hellman Protocol (1976)

- Alice and Bob never met and share no secrets
- Public info: p and g
 - p is a large prime, g is a **generator** of Z_p^*
 - $Z_p^* = \{1, 2 \dots p-1\}$; for all a in Z_p^* there exists i s.t. $a = g^i \pmod p$
 - Modular arithmetic: numbers “wrap around” after they reach p



Compute $k = (g^y \pmod p)^x = g^{xy} \pmod p$ Compute $k = (g^x \pmod p)^y = g^{xy} \pmod p$

Why is Diffie-Hellman Secure?

- Discrete Logarithm (DL) problem:
given $g^x \bmod p$, it's hard to extract x
 - There is no known efficient algorithm for doing this
 - This is not enough for Diffie-Hellman to be secure!
- Computational Diffie-Hellman (CDH) problem:
given $g^x \bmod p$ and $g^y \bmod p$, it's hard to compute $g^{xy} \bmod p$
 - ... unless you know x or y , in which case it's easy
- Decisional Diffie-Hellman (DDH) problem:
given $g^x \bmod p$ and $g^y \bmod p$, it's hard to tell the difference between $g^{xy} \bmod p$ and $g^r \bmod p$ where r is random

Properties of Diffie-Hellman

- Assuming DDH problem is hard (depends on choice of parameters!), Diffie-Hellman protocol is a secure key establishment protocol against passive attackers
 - Common recommendation:
 - Choose $p=2q+1$, where q is also a large prime
 - Choose g that generates a subgroup of order q in Z_p^*
 - Eavesdropper can't tell the difference between the established key and a random value
 - Often hash $g^{xy} \bmod p$, and use the hash as key ($K = \text{Hash}(g^{xy} \bmod p)$)
 - Can use the new key for symmetric cryptography
- Diffie-Hellman protocol (by itself) does not provide authentication
 - Party in the middle attack (often called “man in the middle attack”)

More on Diffie-Hellman Key Exchange

- **Important Note: Examples use discrete logs modulo integers.**
- **Significant advantages in using elliptic curve groups – groups with some similar mathematical properties (i.e., are “groups”) but have better security and performance (size) properties**

Summary of Public Key Crypto

- Encryption for confidentiality
 - Anyone can encrypt a message
 - With symmetric crypto, must know secret key to encrypt
 - Only someone who knows private key can decrypt
 - Key management is simpler (or at least different)
 - Secret is stored only at one site: good for open environments
- Digital signatures for authentication
 - Can “sign” a message with your private key
- Session key establishment
 - Exchange messages to create a secret session key
 - Then switch to symmetric cryptography (why?)

Stepping Back Again, Context

- Increasing security can shift adversarial actions

Improved Security, Increased Risk

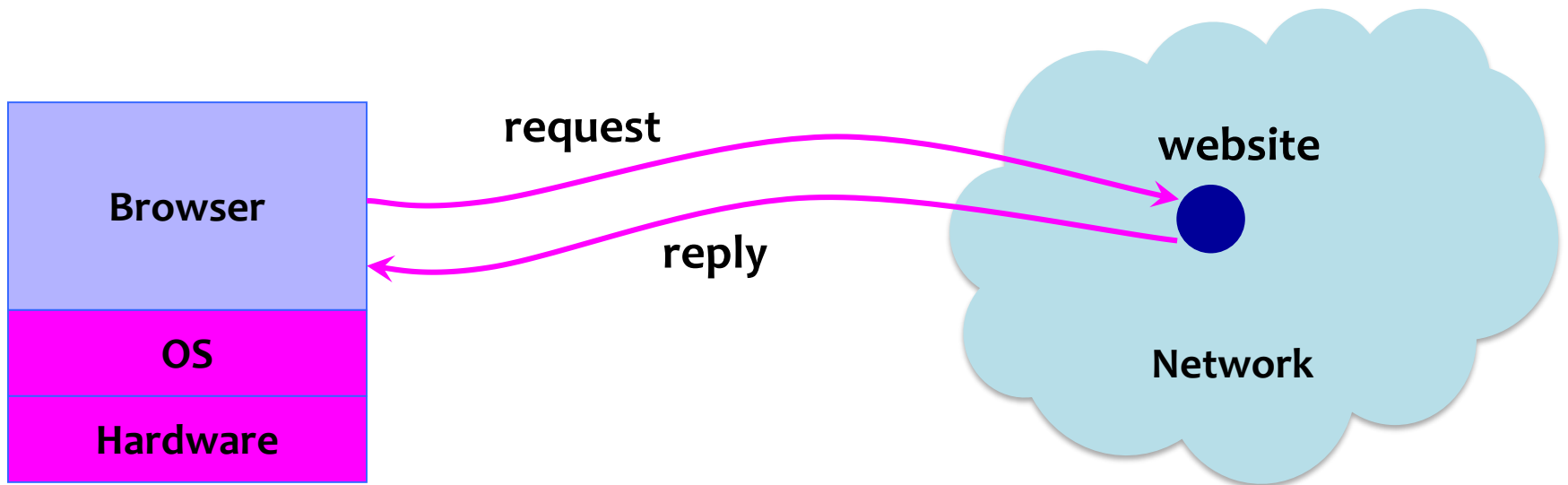
- RFIDs in car keys:
 - RFIDs in car keys make it harder to hotwire a car
 - Result: Car jackings increased

- RFIDs in car keys:
 - RFIDs in car keys
 - Result: Car jackin
- Biometric car lock defeated by cutting off owner's finger
- POSTED BY CORY DOCTOROW, MARCH 31, 2005 7:53 AM | [PERMALINK](#)
- Andrei sez, "'Malaysia car thieves steal finger.' This is what security visionaries Bruce Schneier and Ross Anderson have been warning about for a long time. Protect your \$75,000 Mercedes with biometrics and you risk losing whatever body part is required by the biometric mechanism."
- “ ...[H]aving stripped the car, the thieves became frustrated when they wanted to restart it. They found they again could not bypass the immobiliser, which needs the owner's fingerprint to disarm it.
- They stripped Mr Kumaran naked and left him by the side of the road - but not before cutting off the end of his index finger with a machete.

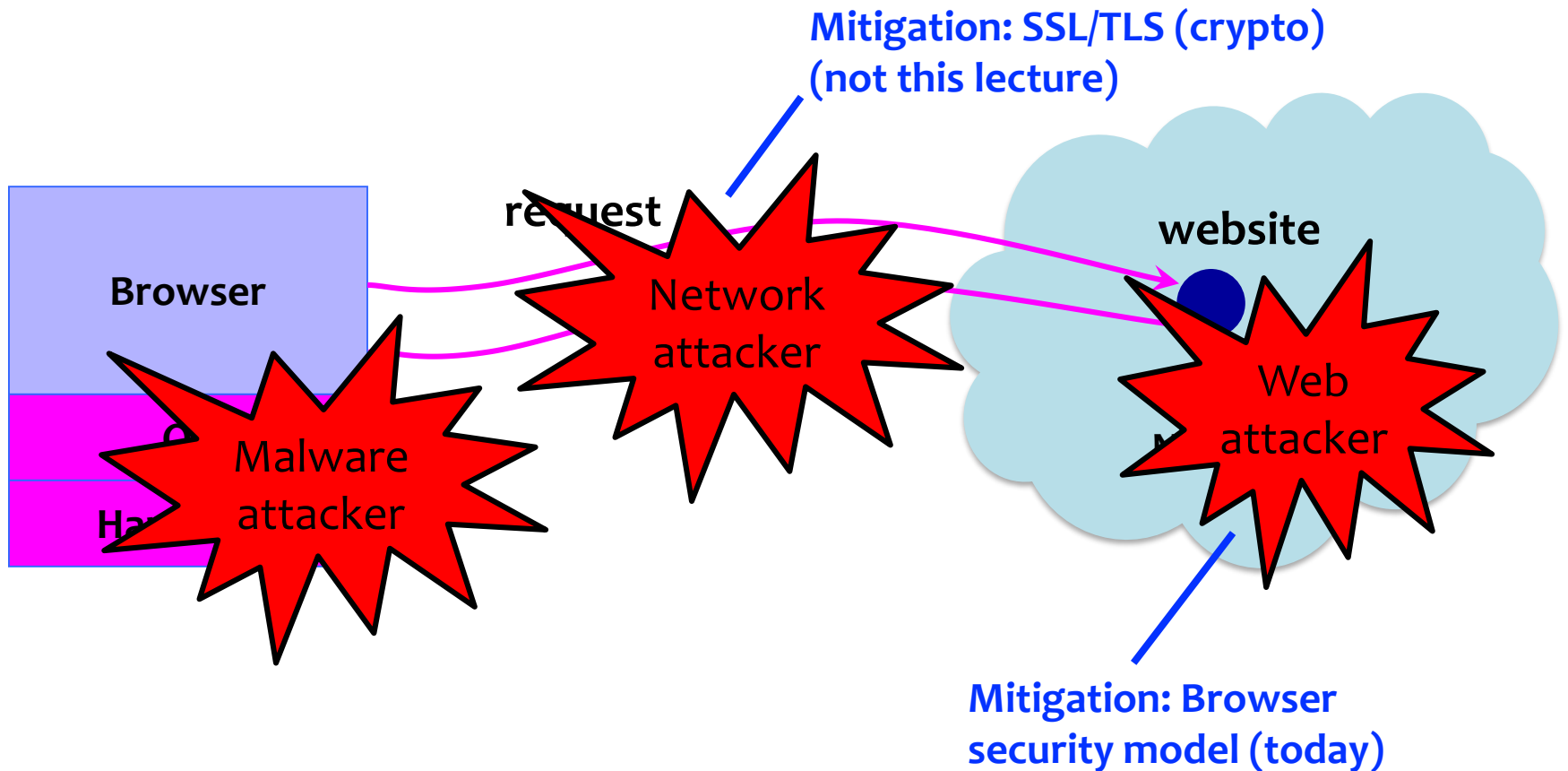
Where is Crypto Used? Everywhere?

- Definitely *the Web*
- Now Transitioning to Web Security
(returning to crypto and the web later)


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

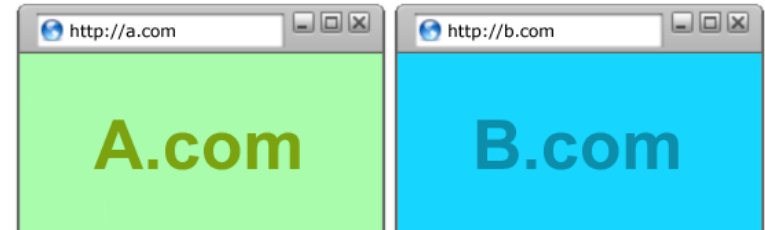
- Online merchants, banks, blogs, Google Apps ...
- Mix of server-side and client-side code
 - Server-side code written in PHP, Ruby, ASP, JSP
 - Client-side code written in JavaScript
- Many potential bugs: XSS, XSRF, SQL injection

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 and 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	Different origin
https://www.example.com/dir/other.html	Failure	
http://en.example.com/dir/other.html	Failure	
http://example.com/dir/other.html	Failure	
http://v2.www.example.com/dir/other.html	Failure	

[Example from Wikipedia]

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]

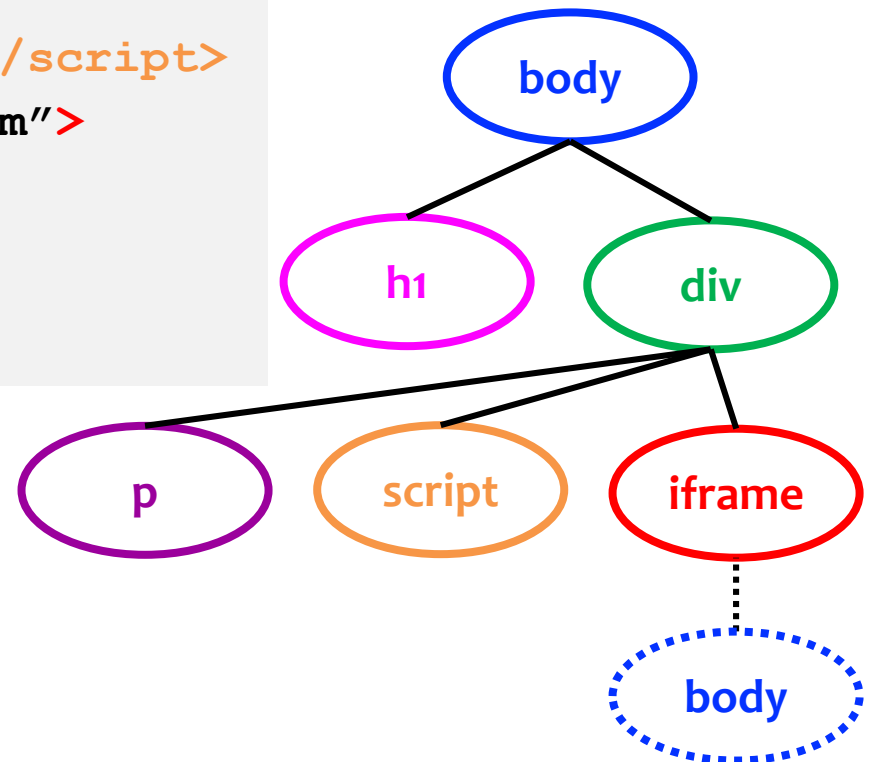
Same Origin Policy is Subtle!

- 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

HTML + Document Object Model

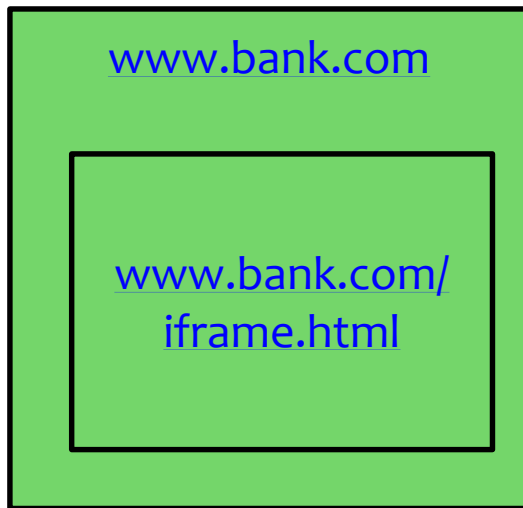
```
<html> <body>
<h1>This is the title</h1>
<div>
<p>This is a sample page.</p>
<script>alert("Hello world");</script>
<iframe src="http://example.com">
</iframe>
</div>
</body> </html>
```

Document Object Model (DOM)

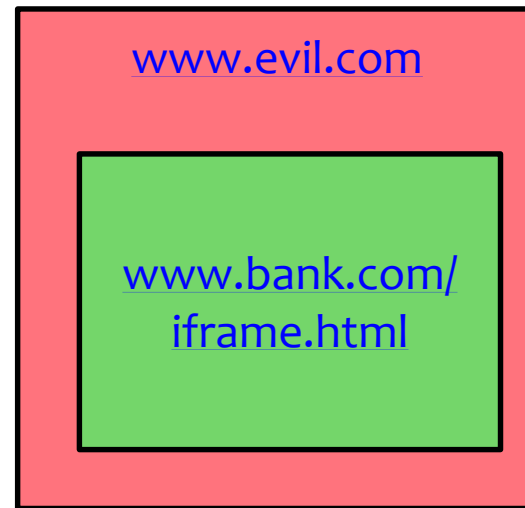


Same-Origin Policy: DOM

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).



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

Question: Who Can Navigate a Frame?

Older Issue



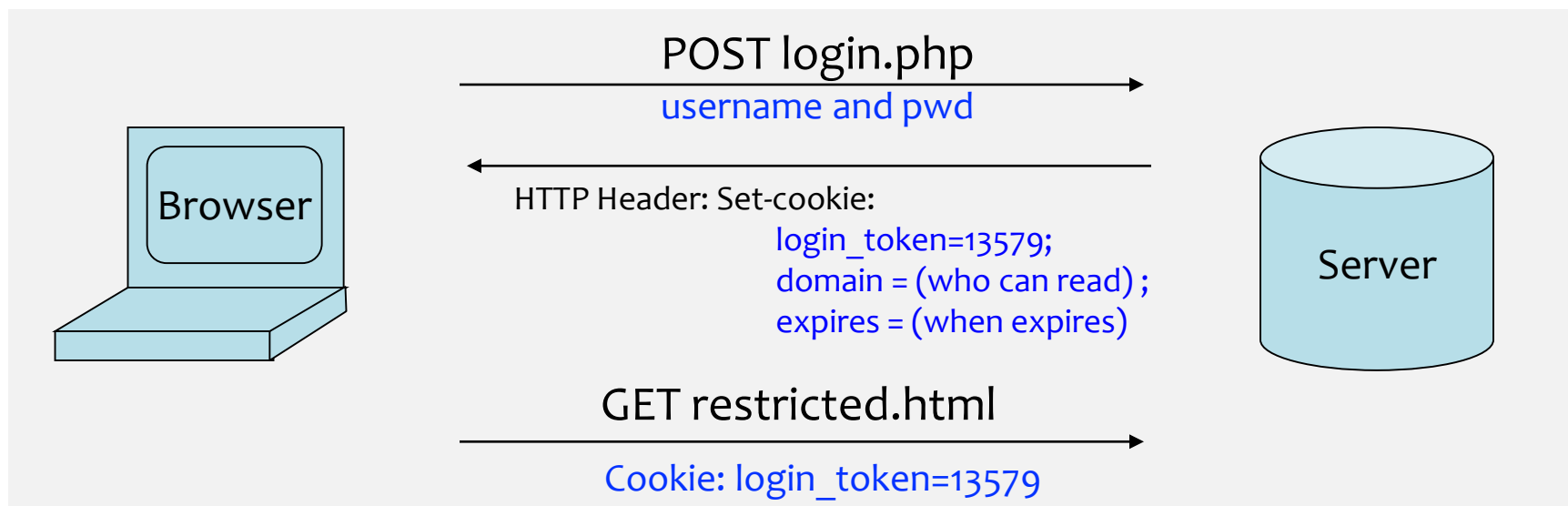
Solution: Modern browsers only allow a frame to navigate its “descendent” frames



If bad frame can **navigate** sibling frames, attacker gets password!

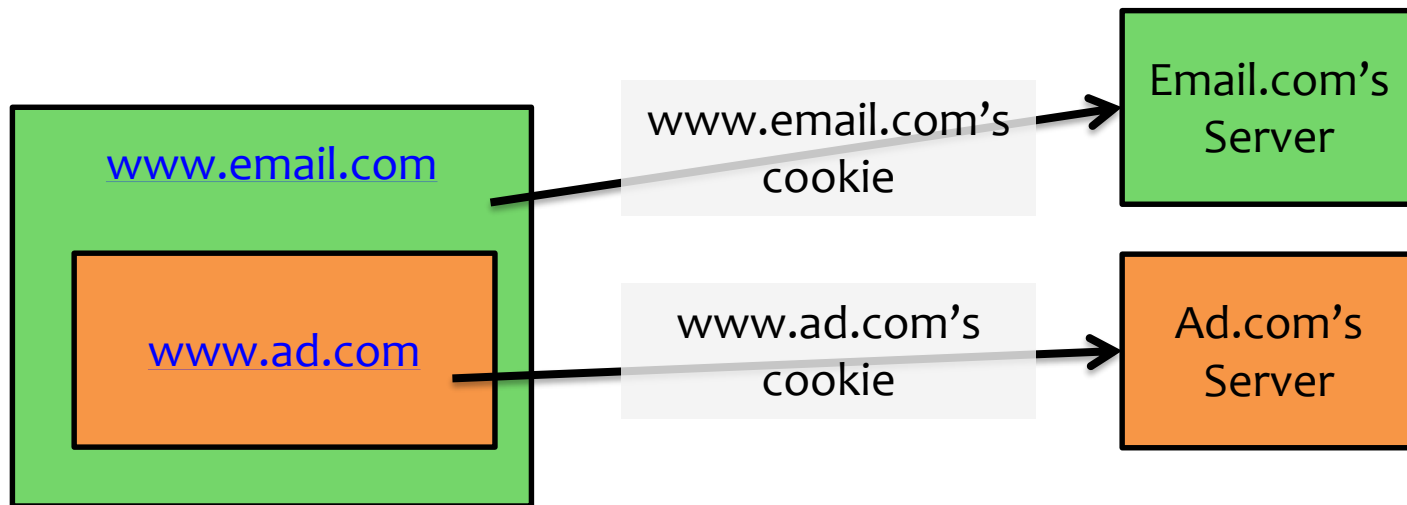
Browser Cookies

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



Same Origin Policy: Cookie Reading

- Websites can only read/receive cookies from the same domain
 - Can't steal login token for another site 😊



Same Origin Policy: Cookie Writing

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

allowed domains

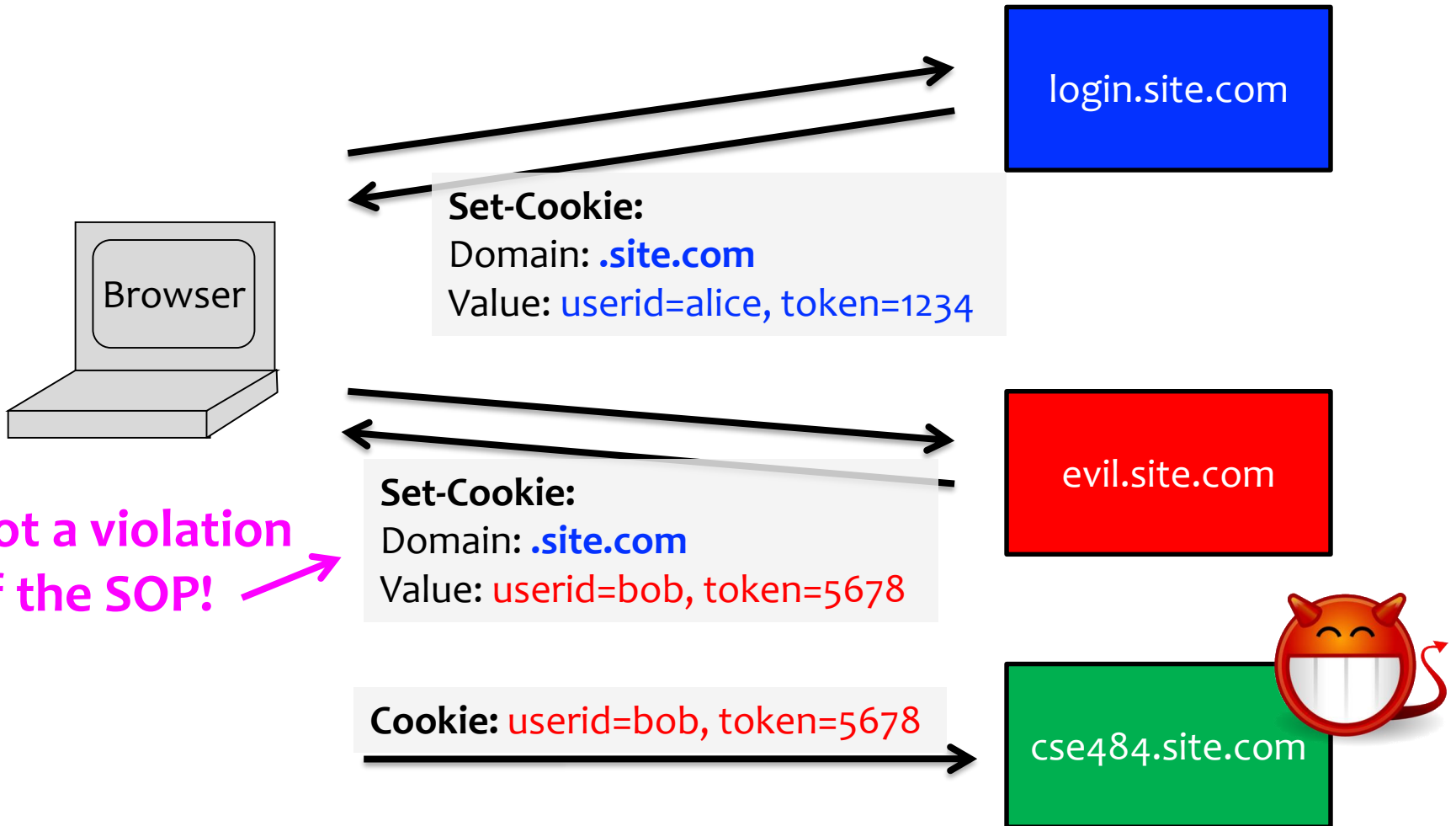
- ✓ **login.site.com**
- ✓ **.site.com**

disallowed domains

- ✗ **othersite.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?



Not a violation
of the SOP!