CSE 484/M584: Computer Security (and Privacy)

Spring 2025

David Kohlbrenner
dkohlbre@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner, Nirvan Tyagi. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials

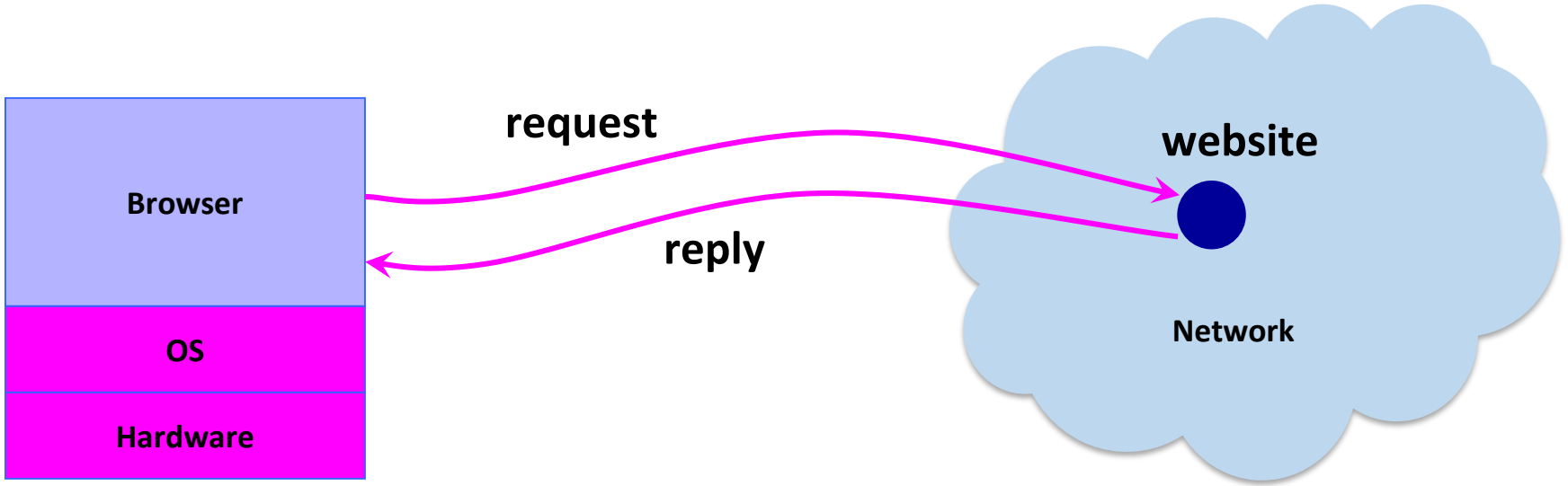
Admin

- Lab 2 (Cryptolab) due today

Next Major Topic!

Web+Browser Security

Big Picture: Browser and Network



Two Sides of Web Security

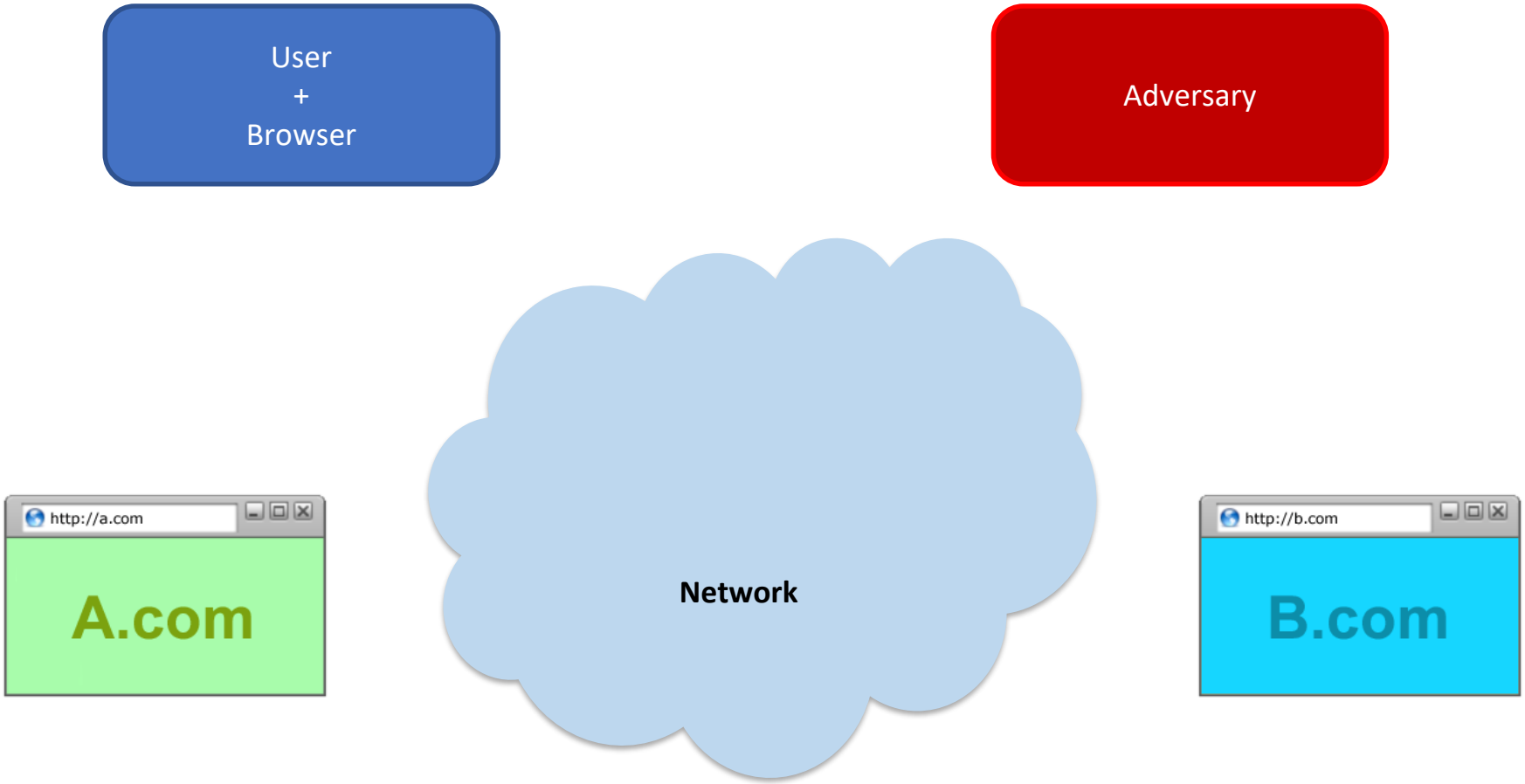
(1) Web browser

- Responsible for securely confining content presented by visited websites

(2) Web applications

- Online merchants, banks, blogs, email…
- Mix of server-side and client-side code
  - Server-side code written in Go, PHP, JavaScript, C++ etc.
  - Client-side code written in JavaScript (… sort of)
- Client cannot be trusted: server must treat input carefully.
  - Many potential bugs: XSS, XSRF, SQL injection.

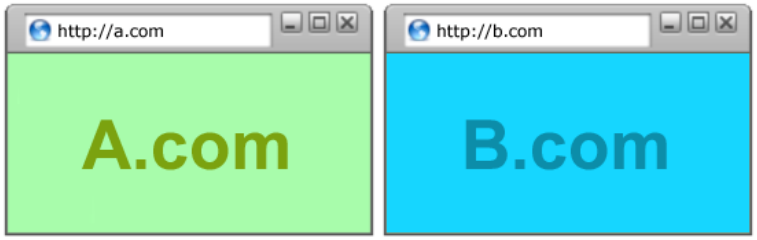
Potentially many actors!



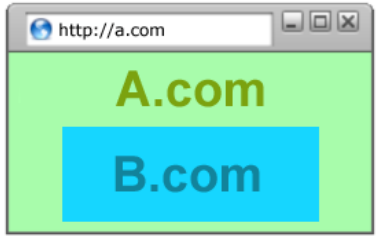
Browser: All of These Should Be Safe

- Safe to visit an evil website

- Safe to visit two pages
  - Simultaneously
  - Sequentially



- Safe delegation



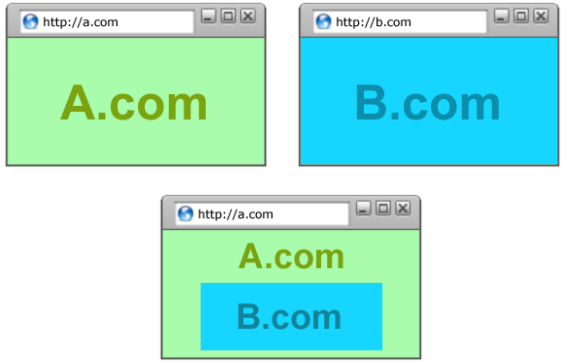
Browser Security Model

Goal 1: Protect local system from web attacker

→ Browser Sandbox

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 and iframes in their own processes
- Implementation is browser and OS specific*

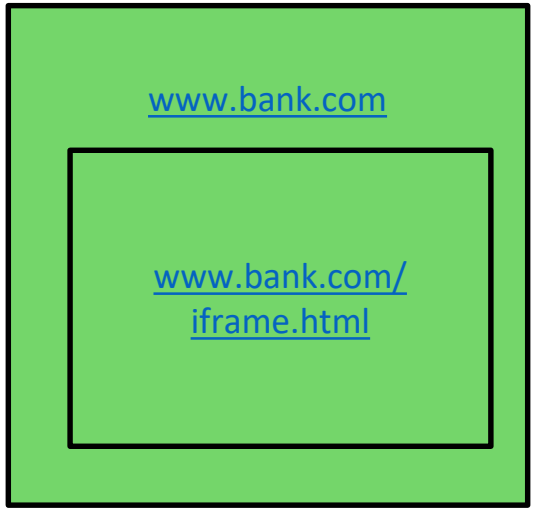
*For example

	High-quality report with demonstration of RCE	High-quality report demonstrating controlled write	High-quality report of demonstrated memory corruption	Baseline
Sandbox escape / Memory corruption / RCE in a non-sandboxed process [1], [2]	Up to $250,000	Up to $90,000	Up to $35,000	Up to $25,000

From Chrome Vulnerability Rewards Program

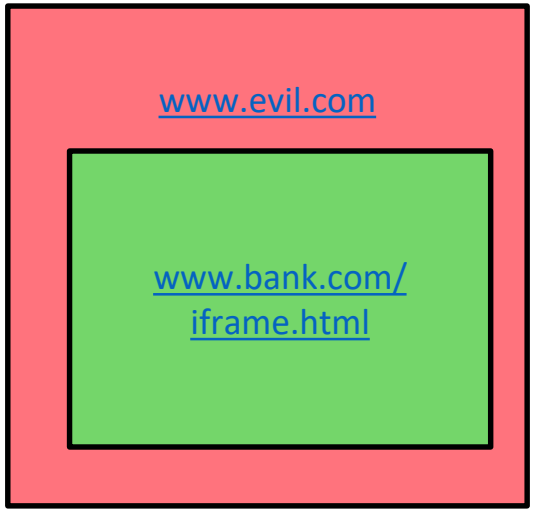
Same-Origin Policy

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).

Same Origin Policy

Website origin = (scheme, domain, port)

Goal: Protect/isolate web content from other web content

Same Origin Policy

Website origin = (scheme, domain, port)

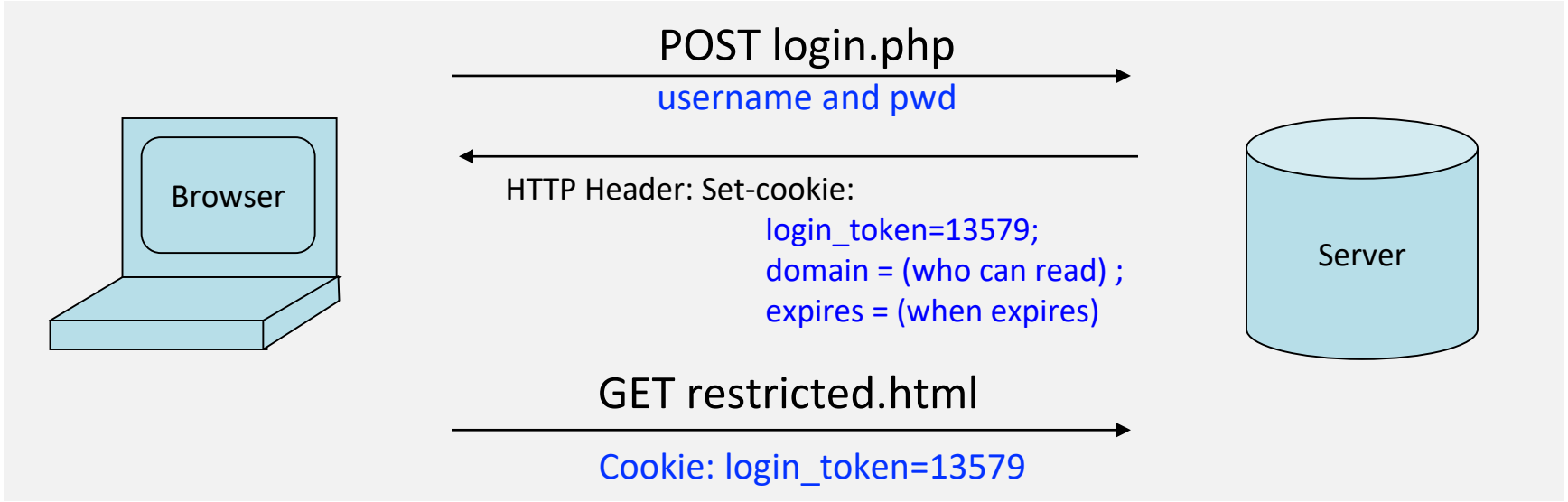
Compare: http://www.example.com/dir/page.html

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same scheme, host and port
http://www.example.com/dir2/other.html	Success	Same scheme, host and port
http://username:password@**www.example.com**/dir2/other.html	Success	Same scheme, host and port
http://www.example.com:**80**/dir/other.html	Success	Most modern browsers implicitly assign the protocol's default port when omitted.[6][7]
http://www.example.com:**81**/dir/other.html	Failure	Same scheme and host but different port
https://www.example.com/dir/other.html	Failure	Different scheme
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)
data:image/gif;base64,R0lGODlhAQABAAAAACwAAAAAQABAAA=	Failure	Different scheme

[Example from Wikipedia]

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



Browser cookies

- Want to set a cookie?
  - `document.cookie="name=value; ";`
  - Yes its that simple

- More commonly, in the HTTP Header response

```
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-value>; Secure
```

Browser cookies

Name	Value	Domain	Path	Expires / Max...	Size	HttpOnly	Secure	SameSite	Partition Key Site	Cross Site	Priority
rpin	384	homes.cs.washington.edu	/~dkohlbre/cew	Session	7						Medium

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Partitio...	Cross Site	Priority
csrftoken	5SGI ...	my.uw.edu	/	2025-10-27T22:5...	41			Lax			Medium
sessionid	hrc[redacted]	my.uw.edu	/	Session	41	✓		Lax			Medium

Same Origin Policy-ish: 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)

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…

Considerations:

- Why would website foobar.com include (directly) a script from baz.com?
  - E.g. <script src=https://baz.com/ascript.js/>
- If they do, what could happen if baz is compromised, or decides to be malicious?
- Gradescope!

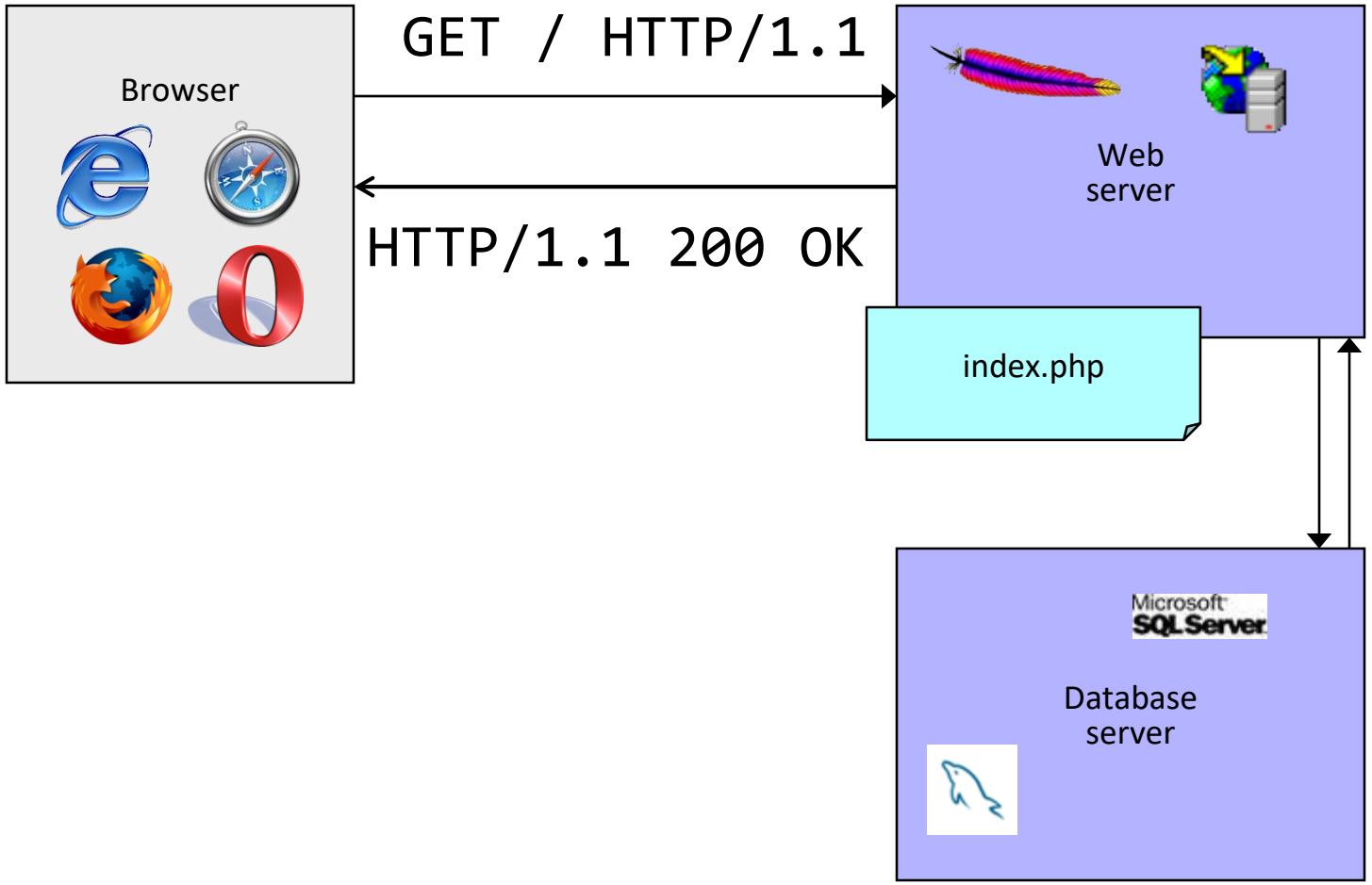
Example: Cookie Theft

- Cookies often contain authentication token
  - Stealing such a cookie == accessing account
- If you can run JS inside the victim page
  - You can just send the cookie wherever you want!
  - How do we get this to happen…?
- Aside: Cookie theft via network eavesdropping
  - Cookies included in HTTP requests
  - One of the reasons HTTPS is important!

Web Application Security:

How (Not) to Build a Secure Website

Dynamic Web Application



Cross-Site Scripting (XSS)

PHP: Hypertext Processor

- Server scripting language
- Can intermingle static HTML and code

```
<input value=<?php echo $myvalue; ?>>
```

- Can embed variables in double-quote strings

```
$user = “world”; echo “Hello $user!”;
```

or

```
$user = “world”; echo “Hello” . $user . “!”;
```

- Form data in global arrays `$_GET`, `$_POST`, …

Demo!

Echoing / “Reflecting” User Input

Classic mistake in server-side applications

http://naive.com/search.php?term=“Can I go back to campus yet?”

search.php responds with

```
<html> <title>Search results</title>
<body>You have searched for <?php echo $_GET[term] ?>…
   </body>
```

Echoing / “Reflecting” User Input

naive.com/hello.php?name=

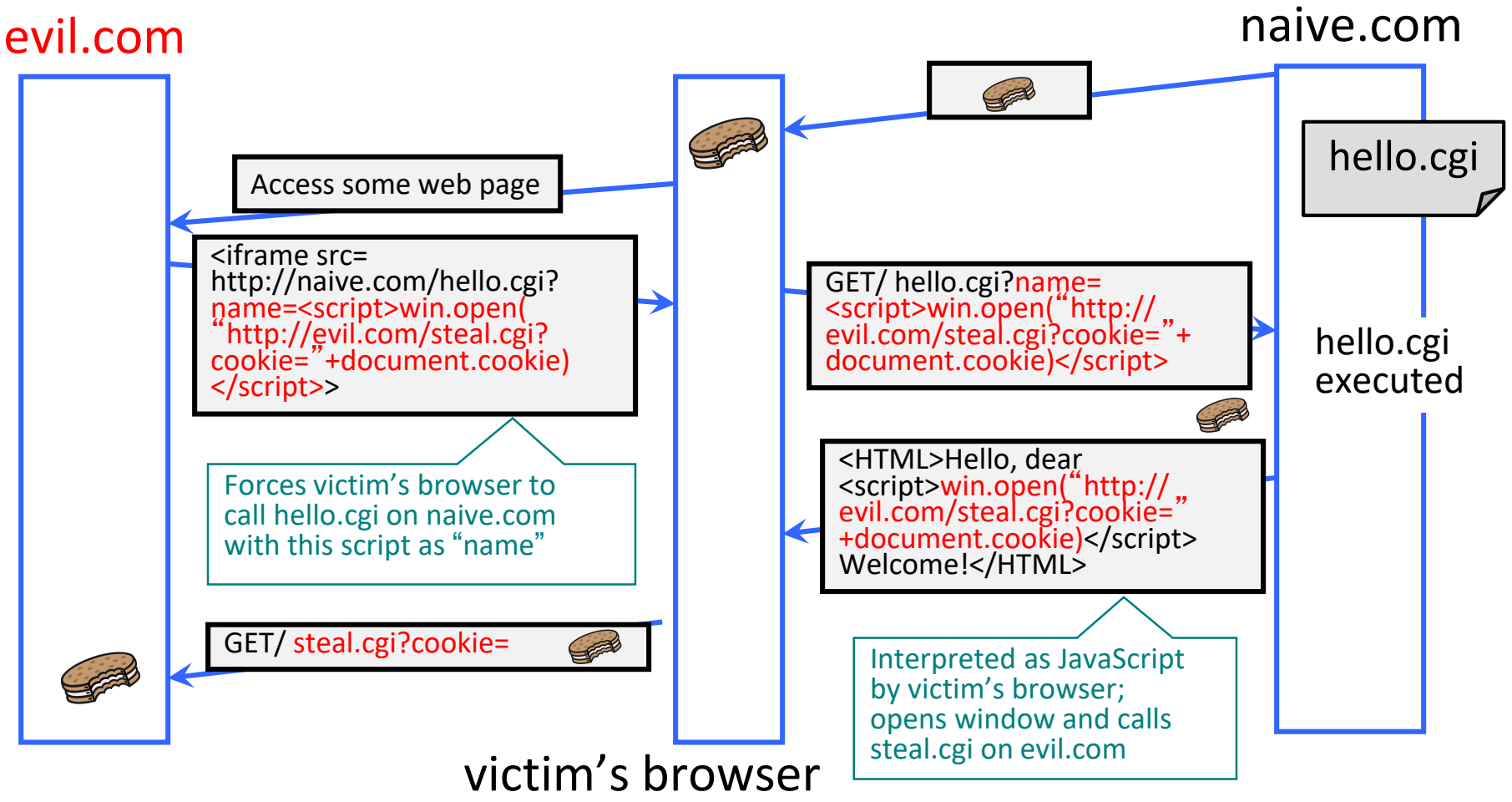
User

Welcome, dear User

naive.com/hello.php?name=*<img src=‘http://upload.wikimedia.org/wikipedia/en/thumb/3/39/YoshiMarioParty9.png/210px-YoshiMarioParty9.png’>*

Welcome, dear

Cross-Site Scripting (XSS)



Basic Pattern for Reflected XSS

Injected script can manipulate website to show bogus information, leak sensitive data, cause user's browser to attack other websites. This violates the "spirit" of the same origin policy