Section 7
Research Topics in Web Security

Including content by Eric Zeng & Keanu Vestil

Administrivia

Upcoming due dates:

- November 19th, 11:59pm: Lab 2 Due
- November 26th, 11:59pm: Final Project Checkpoint #2
- Homework 3: 12/1/2021
- Lab 3 to be released somewhat soon

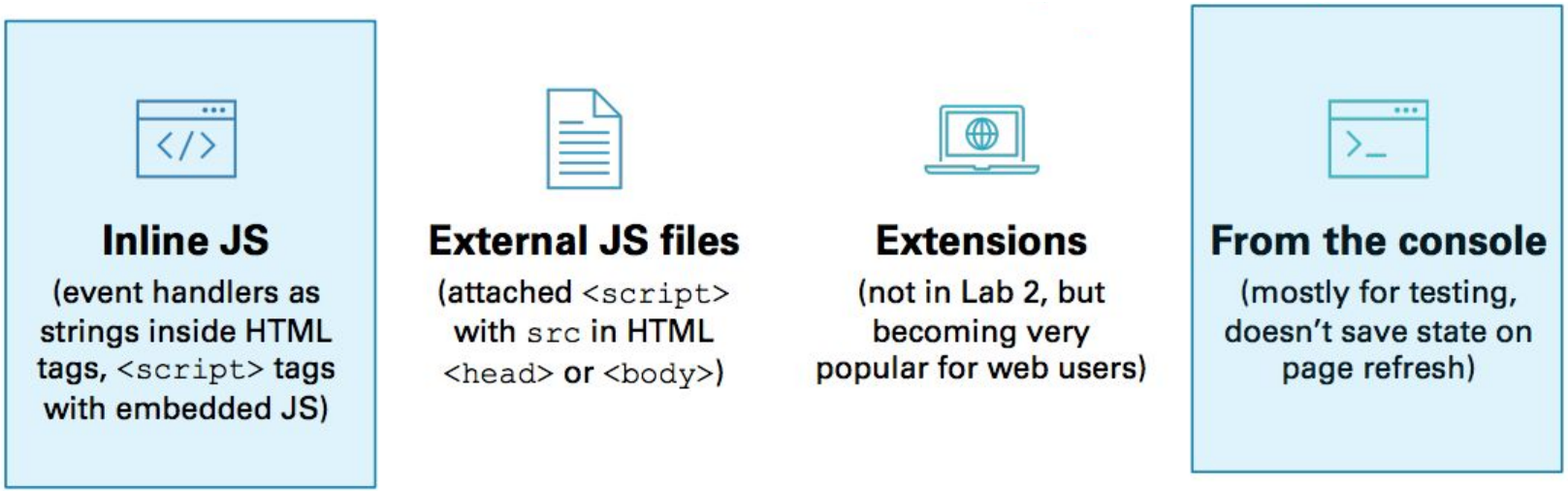
Lab 2 Hints: PHP script setup

Where does the cookie collecting script go?
In your homes directory (your personal CSE website)
`/cse/web/homes/<your_netid>/cookieEater.php`

Cookie collecting script not working?
Make sure to set file permissions on your PHP file so that Apache Server can access it

```
$ chmod 644 cookieEater.php
$ chmod 622 output.txt
```

Lab 2 Hints: How to run JavaScript on the page



You don't need to know all of them for Lab 2, but you will need to use different approaches for different filters!

Lab 2 Hints: XSS

There are usually multiple ways to do the XSS exploits!

- Example: In Problem 1, `window.open` may fail because of popup blocking.
- What other JavaScript APIs or HTML elements can cause a web request?

Lab 2 Hints: XSS

Mixing HTML, JavaScript, and URLs… which syntax are you using?

HTML
JavaScript

```
<script>
  alert('hi');
</script>
```

```
<body onload="alert('hi');"></body>
```

For event handler attributes, the value is interpreted as **JavaScript code** and inserted into a function:

```
> console.log(myImg.onload.toString());
"function onclick(e) { alert('hi'); }"
```

```
<iframe src="example.com/?id=<script>alert('hi');</script>" />
```

This is a **URL**
Which means it must be **URL encoded**

This is **HTML** + **JS**….
Not on the page containing this iframe…
But on the page inside iframe (assuming it has a sanitization vulnerability)

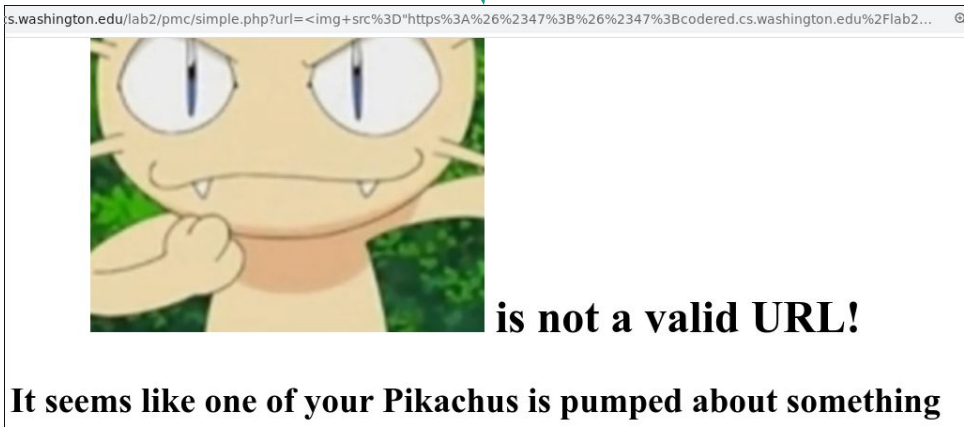
Will this iframe load?
Which language's escape characters do we use?

HTML Escape characters

The `&#47;` is interpreted as a slash / in the image src

Inside HTML attributes (e.g. src), you can use [escape sequences](#) instead of the character itself

&#67;	Uppercase C
&#68;	Uppercase D
&#69;	Uppercase E
&#70;	Uppercase F



Lab 2 Hints: SQL

SQL is a language used to manage and query databases

Each database contains tables of data. The SELECT keyword is used to query tables and retrieve data.

In insecure web applications, user-provided strings may be concatenated directly with the query

```
CREATE TABLE students (
    id int,
    name varchar(255)
);

INSERT INTO students
VALUES (1, 'Chamberlin Boyce');

SELECT * FROM students
WHERE id = 1;
```

uw.edu/deleteUser/1

```
DELETE FROM students
WHERE id = 1;
```

uw.edu/deleteUser/1 OR 1; --

```
DELETE FROM students
WHERE id = 1 OR 1; --;
```

SQL injection tips: Gathering information

Some standard SQL injection questions:

- What database software is in use? (Postgres, SQLite, MySQL, etc.)
- What types of queries are being run? (SELECT, INSERT, DELETE, UPDATE, etc.)
- How many columns are being selected/inserted into?

```
SELECT col1, col2, col3 FROM table WHERE col4='%user_data%';
```

```
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL;--';
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL, NULL;--';
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL, NULL, NULL;--';
```

SQL injection tips: Gathering information

Some standard SQL injection questions:

- What database software is in use? (Postgres, SQLite, MySQL, etc.)
- What types of queries are being run? (SELECT, INSERT, DELETE, UPDATE, etc.)
- **How many columns are being selected/inserted into?**

Vulnerable!

```
SELECT col1, col2, col3 FROM table WHERE col4='%user_data%';
```

Error: wrong number of columns

```
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL;--';
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL, NULL;--';
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL, NULL, NULL;--';
```

No error: vulnerable query selects 3 columns

Clickjacking

Defend against CSRF with dynamic tokens

```
<form method="POST">
    <input type="text" name="email" placeholder="someone@example.net">
    <input type="text" name="subscription" placeholder="Mailing list">
    <input type="hidden" name="csrf_token"
value=6kZd6fddy29yRCJfHxSbRmcQCh3vEUSPW">
    <input type="submit" value="Subscribe">
</form>
```

CSRF tokens must be

- Unique to each user
- Unpredictable
- Secret

Clickjacking (UI Redressing)

- Attacker overlays multiple transparent or opaque frames to trick a user into clicking on a button or link on another page
- Clicks meant for the visible page are hijacked and routed to another, invisible page
- Can defeat CSRF tokens

How does it work?

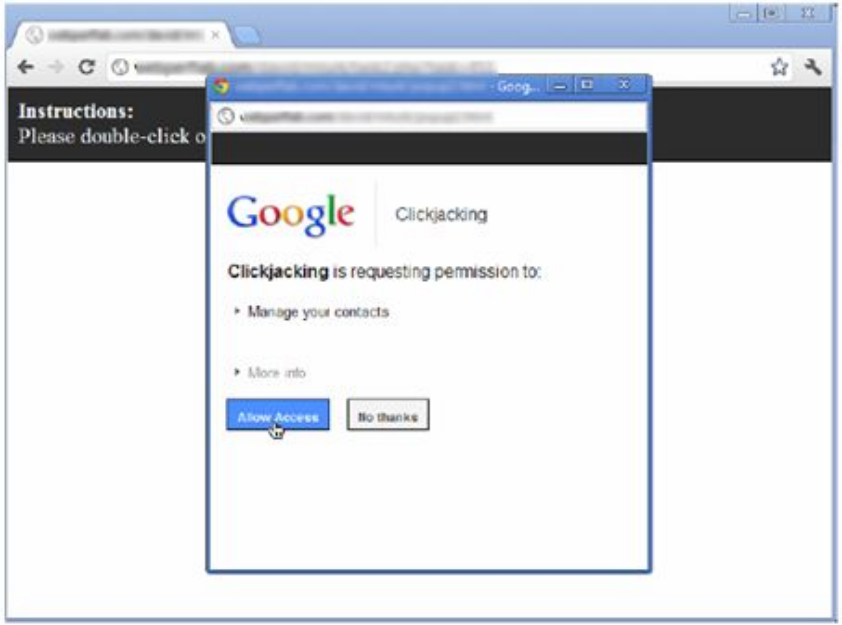
- Any site can embed any other site using an iframe

```
<iframe
  src="http://www.google.com/..">
</iframe>
```

- Use CSS to make the iframe of the target site invisible
  - opacity defines visibility percentage of the iframe
  - 1.0: completely visible
  - 0.0: completely invisible
- Use CSS to put the iframe's button over the parent page's button

Other Variants

- Fake cursors (mouse pointers)
- Stealing text box focus - redirecting typing somewhere else
- Double clickjack: ask user to double click, pop a window up right below the mouse in between clicks



Defenses

- Websites can prevent themselves from being used in an iframe, using Content Security Policy (CSP) to specify which domains can embed them:

```
Content-Security-Policy: frame-ancestors ‘self’;
```

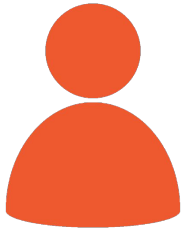
Designing HTTPS Warnings

How does HTTPS/TLS encrypt web traffic?

"I'm Google! Here's my public key! Let's negotiate a symmetric key!"

> Navigate to google.com

"How do I know you're Google?"



"Here's my certificate (signed public key) - it was signed by Comodo CA"

"I trust Comodo CA! Let's check if that's true."
validates certificate chain

Comodo (root cert)

What happens if the connection is tampered with?

"I'm Google! Here's my public key! Let's negotiate a symmetric key!"

"Here's my certificate (signed public key) - it was signed by Comodo CA"

> Navigate to google.com

"How do I know you're Google?"

"I trust Comodo CA! Let's check if that's true."
certificate chain validation fails

Comodo (root cert)

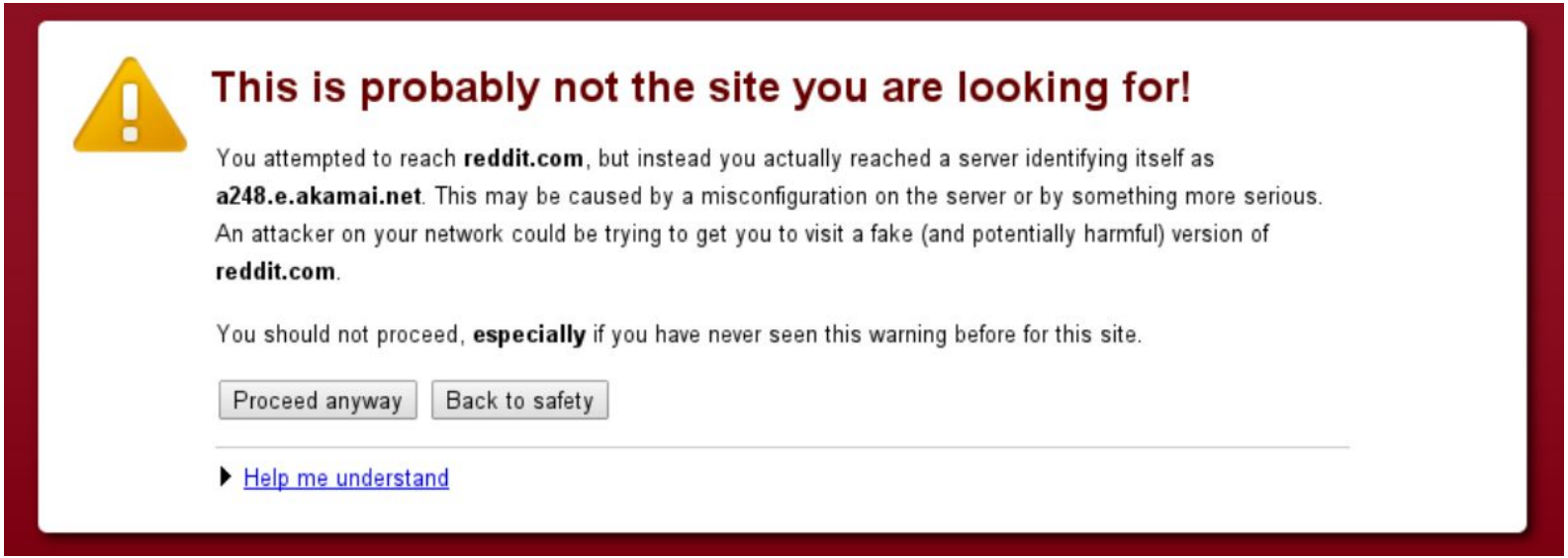
???????

What happens now?

What happens if the connection is tampered with?

- Browser can’t go to the page anymore
  - Could contain malicious JavaScript
  - Could be a perfect copy of the site, trick users into giving up their passwords, CCNs
- But what if it was just a false positive/misconfiguration?
  - Website owner could have served the wrong certificate
  - Website owner could have forgotten to renew their certificate, and it expired
  - User’s computer’s clock could be off, making the browser think the certificate expired

Initial solution: warn the user, let them decide



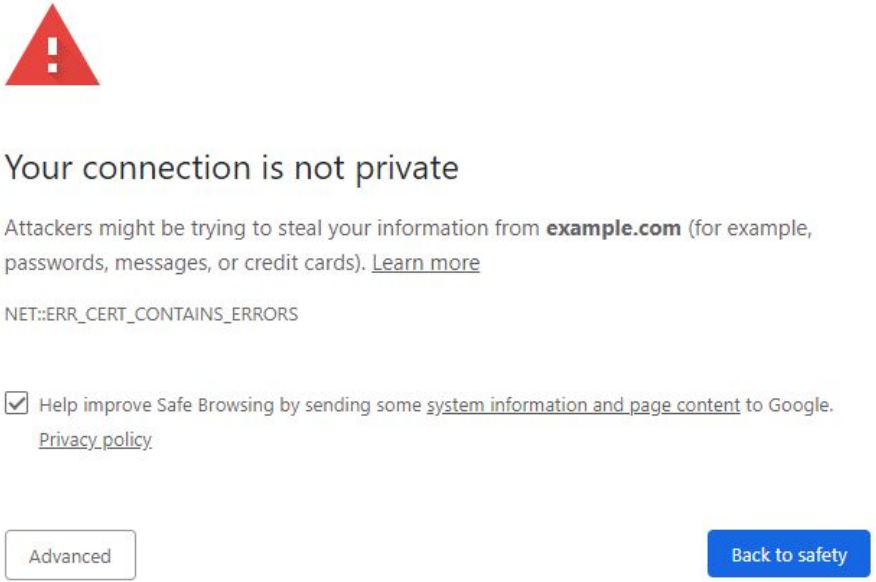
So what did users actually do?

Operating System	SSL Warnings	
	Firefox	Chrome
Windows	32.5%	71.1%
MacOS	39.3%	68.8%
Linux	58.7%	64.2%
Android	NC	64.6%

Table 3: User operating system vs. clickthrough rates for SSL warnings. The Google Chrome data is from the stable channel, and the Mozilla Firefox data is from the beta channel.

Alice in Warningland: A Large-Scale Field Study of Browser Security Warning.
Devdatta Akhawe, Adrienne Porter Felt. USENIX Security 2013

Opinionated Design: Make the bad thing hard



Need to click here first to get the ignore button

Did it work?

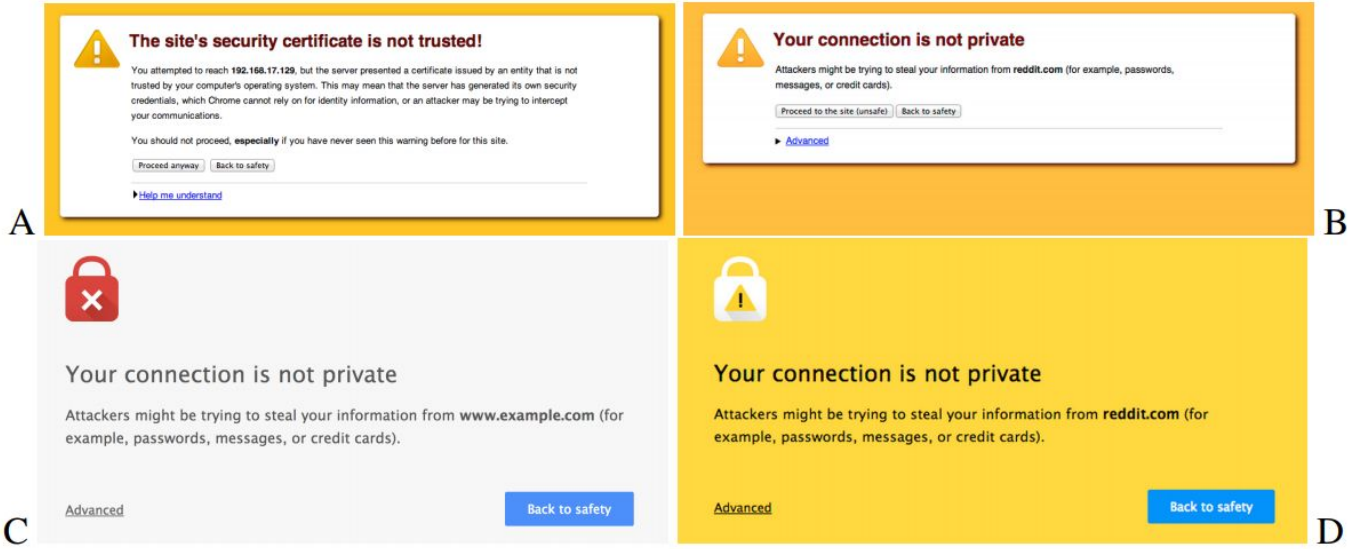


Figure 4. Conditions for our field experiment. *A* is the Chrome 36 warning, and *C* is the Chrome 37 warning.

	Text	Design	Adherence	N
A	Original	Original	30.9%	4,551
B	Proposed	Original	32.1%	4,075
C	Proposed	Proposed (gray)	58.3%	4,633
D	Proposed	Proposed (yellow)	53.3%	4,528

Table 5. Adherence rates from the field experiment.

Improving SSL Warnings: Comprehension and Adherence
Adrienne Porter Felt , Alex Ainslie , Robert W. Reeder , Sunny Consolvo , Somas Thyagaraja, Alan Bettes, Helen Harris, Jeff Grimes
CHI 2015

yup

Lessons

- Opinionated Design
  - Don’t force users to make security decisions!
  - But if you have to, make it hard for them to make mistakes
- Reducing HTTPS Warnings
  - If you show too many false positives, people get desensitized and have a harder time identifying real problems - *warning fatigue*
  - How do we reduce false positives? One approach: notify website owners that they have a misconfiguration

    Fixing HTTPS Misconfigurations at Scale: An Experiment with Security Notifications.
    Eric Zeng, Frank Li, Emily Stark, Adrienne Porter Felt, Parisa Tabriz. WEIS 2019

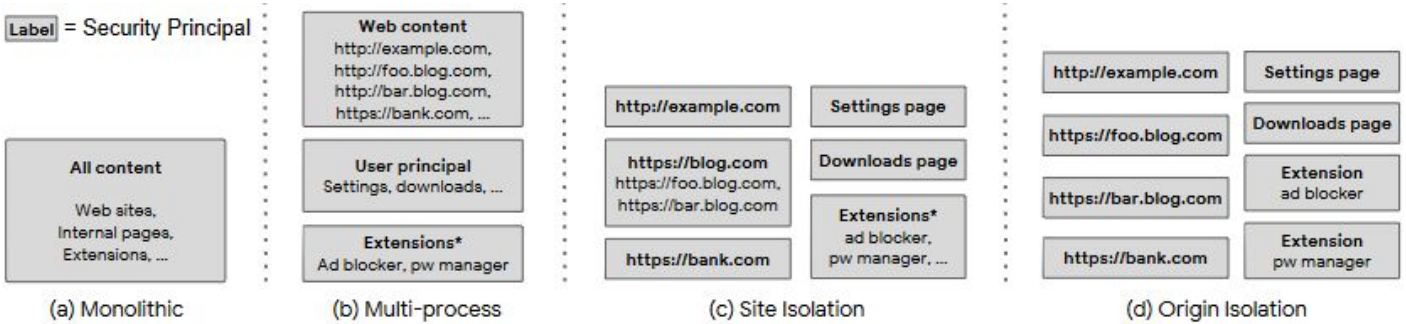
Site Isolation

Why do we want this?

- “Multi-process” browser model is not enough sandboxing
  - Browser loads trusted and untrusted sites in the same renderer process
- Rendering engine bugs are common
  - Can be exploited to access cross-site data
- Universal XSS can bypass Same Origin Policy within the renderer process
- Side channel attacks like Spectre can be exploited without a bug in Chrome
  - Read arbitrary memory in renderer process

Site Isolation: Process Separation for Web Sites within the Browser.
Charles Reis, Alexander Moshchuk, Nasko Oskov, *Google*. (USENIX Security 2019)

Where did we come from?

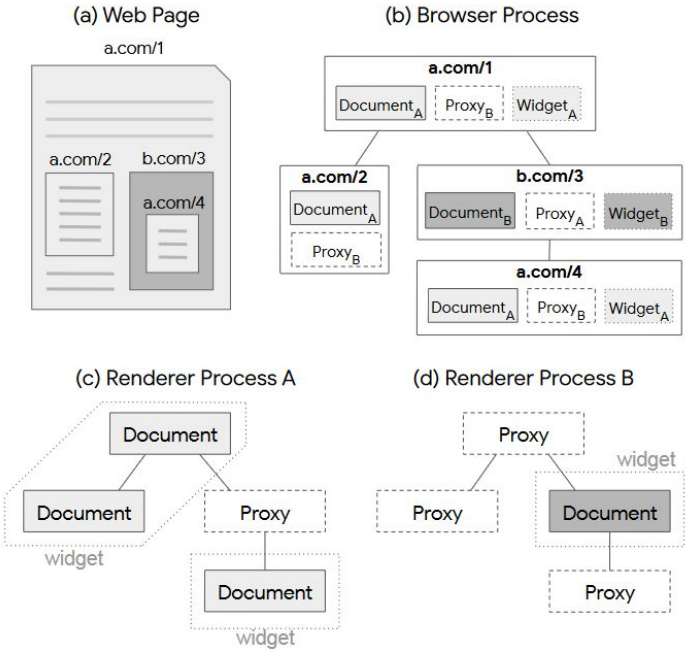


- Transitioning from (b) Multi-process to (c) Site Isolation
- (d) Origin Isolation would be desirable, but the overhead is prohibitive
- See: Barth, et al. “The Security Architecture of the Chromium Browser” (2008)

Site Isolation: Process Separation for Web Sites within the Browser.
Charles Reis, Alexander Moshchuk, Nasko Oskov, *Google*. (USENIX Security 2019)

Site Isolation Design

- Site-dedicated processes
  - Out of process iframes
- Cross-Origin Read Blocking (CORB)
  - Custom confirmation sniffing of response
- Enforcements against malicious agents
  - Browser process tracks illegal requests



Site Isolation: Process Separation for Web Sites within the Browser.
Charles Reis, Alexander Moshchuk, Nasko Oskov, *Google*. (USENIX Security 2019)

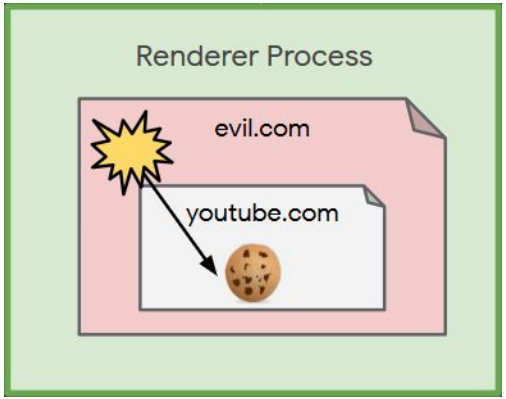
Implementation

- **Significant** 5-year effort
  - Changed or added 450k lines of code in 9000 files
- Optimizations
  - Consolidate processes that refer to the same site
  - Keep a “warmed-up spare” process handy for swaps
- Deployment
  - Test changes with extensions and selective site isolation first
  - Preliminary isolation modes used to gather bug reports and performance data
  - Each milestone utilized Chrome’s A/B testing mechanism

Site Isolation: Process Separation for Web Sites within the Browser.
Charles Reis, Alexander Moshchuk, Nasko Oskov, *Google*. (USENIX Security 2019)

Security Evaluation

- New protections:
  - Authentication
  - Cross-origin messaging
  - Anti-clickjacking
  - Data confidentiality (via CORB)
- Remaining potential renderer vulnerabilities:
  - Bypassing site isolation
  - Targeting non-isolated data
  - Cross-process attacks



Example rendering exploit

Site Isolation: Process Separation for Web Sites within the Browser.
Charles Reis, Alexander Moshchuk, Nasko Oskov, *Google*. (USENIX Security 2019)

Performance Evaluation

- Field measurements are more realistic than microbenchmarks
  - e.g. many tabs, “long tail” sites
- Process sharing heuristics decrease potential resource usage
  - Average memory usage per process decreased by 50%, but **overall 9-13% increase in memory overhead**
  - Distributing the same workload across more processes
- Compatibility preservation
  - CORB blocks <1% of responses (20% blocked by traditional content type filtering)
  - Less intrusive than reducing timer precision or modifying JavaScript compiler

Site Isolation: Process Separation for Web Sites within the Browser.
Charles Reis, Alexander Moshchuk, Nasko Oskov, *Google*. (USENIX Security 2019)

Chrome <3 RAM


How Chrome sees RAM memory


When your parents ask
where all the ram went.