





Establishing a robust long-term security model for cookies on the web

Artur Janc, Information Security Engineer, Google

Secure the ... Forward

1. Improve the security of existing codebases and systems
 - Development processes (DevSecOps), static security tooling, documentation, awareness
 - Features to provide new security guarantees and mitigations in case they fail
 - Detecting vulnerabilities (automated, manual reviews, pentests), fast & reliable patching
2. Engineer a solid foundation that prevents classes of vulnerabilities
 - **Languages:** Memory-safe languages, TypeScript  prototype pollution, Go  race conditions
 - **APIs:** Parameterized SQL queries, auto-escaping template systems, ...

An ounce of prevention is worth a pound of cure.

Secure the **Web** Forward

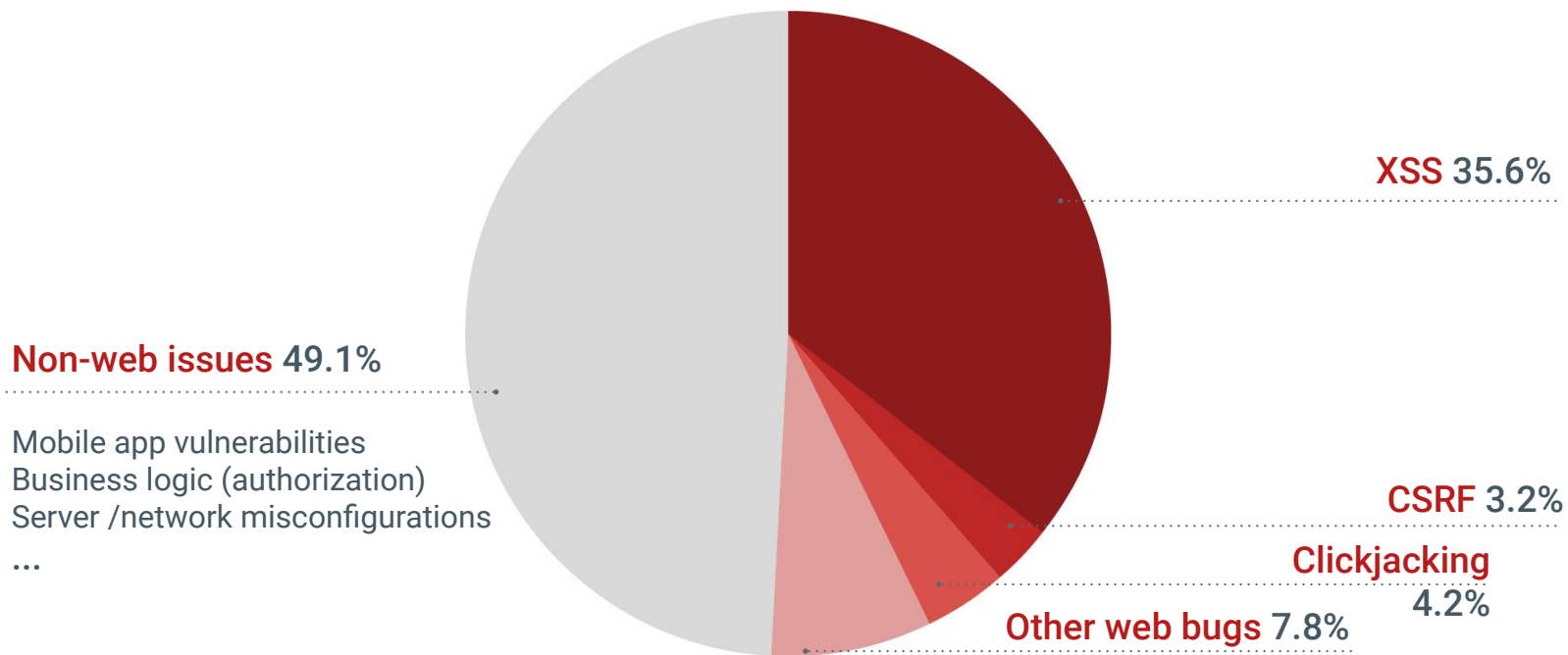
Challenge: Retrofit security into an ecosystem not meant as an application platform.

Many examples of web ecosystem shifts motivated by security:

- HTTPS adoption & blocking mixed content
- Removal of Flash
- Process-level isolation (Site Isolation, Project Fission)
- Smaller deprecations: `document.domain`, content sniffing, ...

Idea: Understand the root causes of security problems in the web ecosystem (specifically, web applications) and try to evolve the web platform to prevent them.

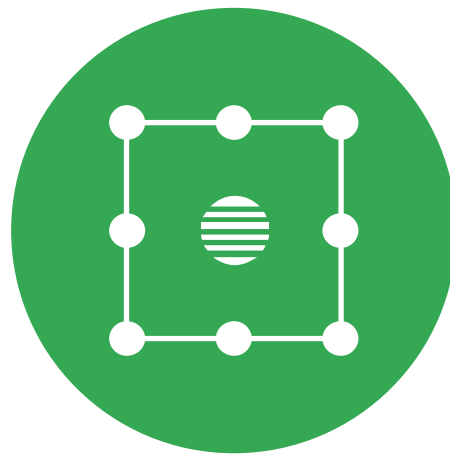
Google Vulnerability Reward Program payouts in 2018





Injections

*all kinds of XSS,
prototype pollution*



Isolation bypasses

*CSRF, clickjacking,
XS-Search, XS-Leaks, ...*



Injections

*all kinds of XSS,
prototype pollution*



Isolation bypasses

*CSRF, clickjacking,
XS-Search, XS-Leaks, ...*

Web functionality with cross-site vulnerabilities

OUR WEBSITE:

```
<form action="/transfer">  
  <input name="target" value="mkwst" />  
  <input name="amount" value="10" />
```

form submission

```
<button onclick="deleteAccount()">  
  Delete account</button>
```

clickable button

```
w("Content-Type: text/javascript")  
w("var data = {'user': '${name}'}")
```

Data in a JS response

```
if search_result:  
  log_to_db(search_query)  
  return search_result
```

search functionality

Web functionality with cross-site vulnerabilities

OUR WEBSITE:

```
<form action="/transfer">
  <input name="target" value="ml" />
  <input name="amount" value="10" />
```

CSRF

```
<button onclick="deleteAccount()">
  Delete account</button>
```

clickjacking

```
w("Content-Type: text/javascript")
w("var data = {'user': '${name}'}")
```

XSSI

```
if search_result:
  log_to_db(s)
  return search_result
```

XS-Search / timing

EVIL.COM:

```
<form action="//victim/transfer">
<input name="target" value="bozo" />
<input name="amount" value="1000" />
```

```
<iframe src="//victim/settings"
  style="opacity: 0"></iframe>
```

```
<script src="//victim/json" />
<script>alert(data)</script>
```

```
<script>t=performance.now()</script>

```


`https://victim.com`



`https://victim.com/image.png`

`https://evil.com`



`https://victim.com/image.png`

victim.com server



Corporate needs you to find the differences between this picture and this picture.



They're the same 

Improving the cookie model has to potential to address many of the web's isolation problems and prevent several classes of bugs



sciendo

Proceedings on Privacy Enhancing Technologies ; 2018 (4):85–103

Muhammad Ahmad Bashir and Christo Wilson

Diffusion of User Tracking Data in the Online Advertising Ecosystem

Through simulations, we find that 52 A&A companies are each able to observe 91% of an average user's impressions as they browse, under modest assumptions about data sharing in RTB auctions. 636 A&A companies are able to observe at least 50% of an average user's impressions.

Browsers have committed to removing third-party cookies

MOZILLA

Firefox rolls out Total Cookie Protection by default to more users worldwide

📅 APRIL 11, 2023 👤 MOZILLA

Tracking prevention in Microsoft Edge

Article • 01/13/2023 • 7 contributors



WebKit

[Blog](#) [Downloads](#) [Feature Status](#) [Documentation](#) ▾ [Contribute](#) ▾

Full Third-Party Cookie Blocking and More

Mar 24, 2020

by John Willander

[@johnwillander](#)

This blog post covers several enhancements to Intelligent Tracking Prevention (ITP) in iOS and iPadOS 13.4 and Safari 13.1 on macOS to address our latest discoveries in the industry around tracking.



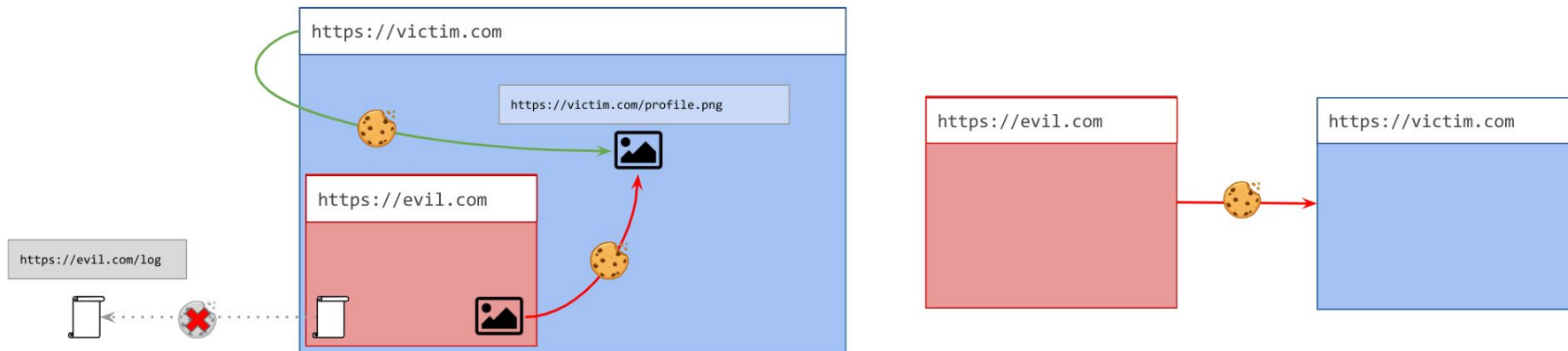
Chromium Blog

News and developments from the open source browser project

Building a more private web: A path towards making third party cookies obsolete

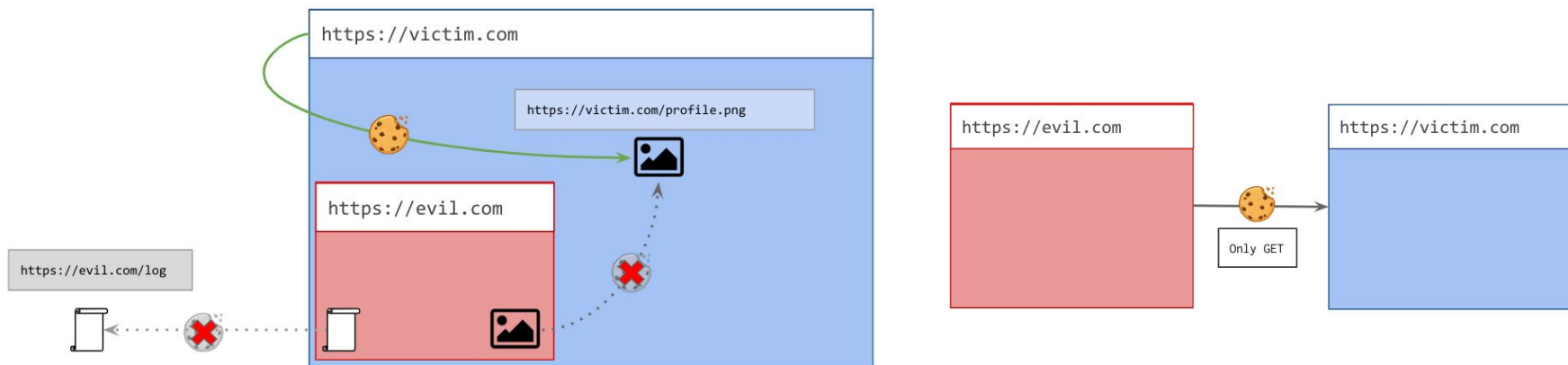
Tuesday, January 14, 2020

"Anti-tracking" cookie blocking



All requests for resources under a top-level site carry that site's cookies, including requests made from cross-site or **sandbox** iframes. All navigations have cookies.

SameSite=Lax



Uses the "site for cookies" algorithm from [RFC6265bis](#), omitting sending of cookies if the initiating document is cross-site, or there are cross-site ancestors or redirects. Navigations made using safe HTTP methods (GET) include cookies.

What we want from cookies (security perspective)

1. Cookies should not be sent on cross-site resource requests
 - Specifically, no cookies on requests from cross-site frames (or with a cross-site ancestor)
2. Cookies are okay for top-level navigations with safe HTTP methods (**GET**)
 - Assume endpoints prone to CSRF use non-safe methods (**POST**, **PUT**, etc.)
 - Could still leak data from popups, but for that we have [Cross-Origin Opener Policy](#)

This behavior gives us **both** the security and privacy properties we care about.

... this is **SameSite=Lax!**

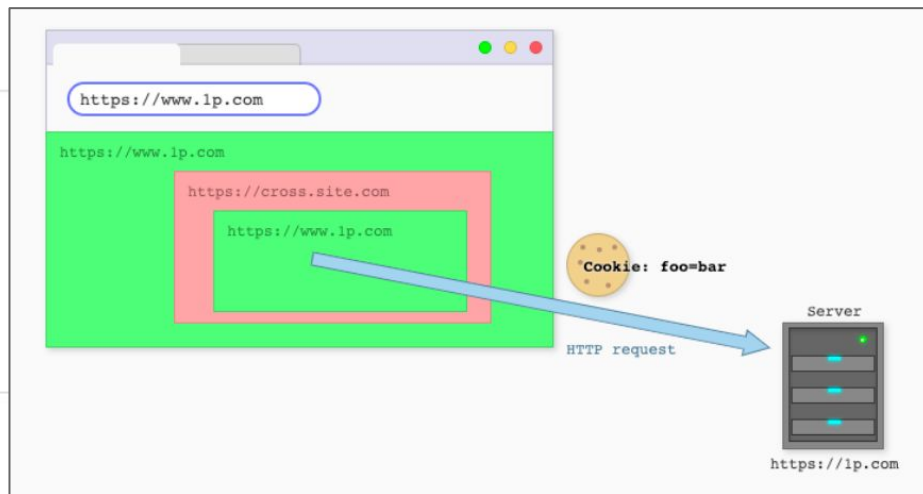
Standardizing Security Semantics of Cross-Site Cookies

Editors

- Dylan Cutler
- Kaustubha Govind
- Johann Hofmann

Table of Contents

- The Problem
- Goals



<https://github.com/DCtheTall/standardizing-cross-site-cookie-semantics/>

How much security can you opt out of with `SameSite=None` cookies?

1. **Same-site iframes with cross-site ancestors** (ABA embeds)
 - **Don't send cookies.** Would allow clickjacking and XS-Leaks.
2. **Navigating a cross-site iframe to a same-site destination**
 - **Send cookies** (only for GET navigations). Little risk, similar to top-level navigations.
3. **Top-level cross-site POST requests**
 - **Send cookies.** Necessary for compatibility, but we want to lock it down more (with CORS?).
4. **Redirecting cross-site resources to same-site destinations**
 - **Don't send cookies.** No strong compatibility reason.

We should assume websites in the future will relax cross-site defenses.




Security goal: **SameSite=Lax*** model as a platform boundary

[*] A number of edge cases to hash out ([Standardizing Security Semantics...](#))

What this would give us: A platform-enforced guarantee against loading authenticated cross-site resources or iframes.

All browsers are fairly close to getting there because of the anti-tracking work.

What browsers would need to do:

-  Complete the [third-party cookie deprecation](#) process & fix known gaps
-   Switch to the [Lax-allowing-unsafe](#) model
- Everyone: Agree on handling remaining under-defined behaviors

If we get this right

The web platform will provide robust protections from many cross-site attacks, removing a *security tax* on developers forced to build application-level defenses.

It's important for the security community to pay attention and be vocal about the long-term value of these improvements for the web ecosystem.

Discussion