CLICK HERE.exe

sourcetoad

XSS & CSRF

Security Meetup

sourcetoad

Month 2 of 12 (February)

- Last month: **SQL Injections**
- This month: **XSS / CSRF**
- Next month: **DDoS / DoS**
- Meetup Group for times/dates

sourcetoad

Plan of Attack

- The Safe Web
- The Malicious Web
- XSS Abuse
- CSRF Abuse
- Protections

Who are you?

- **Connor Tumbleson**
- Sourcetoad Engineer
- Apktool - RE Tool
- @iBotPeaches

sourcetoad

The Safe Web

- Security was an afterthought
- Protocols were designed with trust
- Didn’t expect dark intentions

Yahoo! - Help

Welcome, Guest - [Sign in]

Get a free home page

BUILD A PAGE
Create a new webpage.
EDIT PAGES
Use File Manager to work on your site.
UPLOAD FILES
Import or FTP sounds, pictures or HTML files.

Returning GeoCities members, learn how to sign in for the first time.

Search Home Pages

Explore Neighborhoods

Area51	Hollywood	TimesSquare
Colosseum	SouthBeach	Tokyo
Heartland	SunsetStrip	WestHollywood

Take a vacation in TheTropics, one of our 41 Neighborhoods.

New and Notable

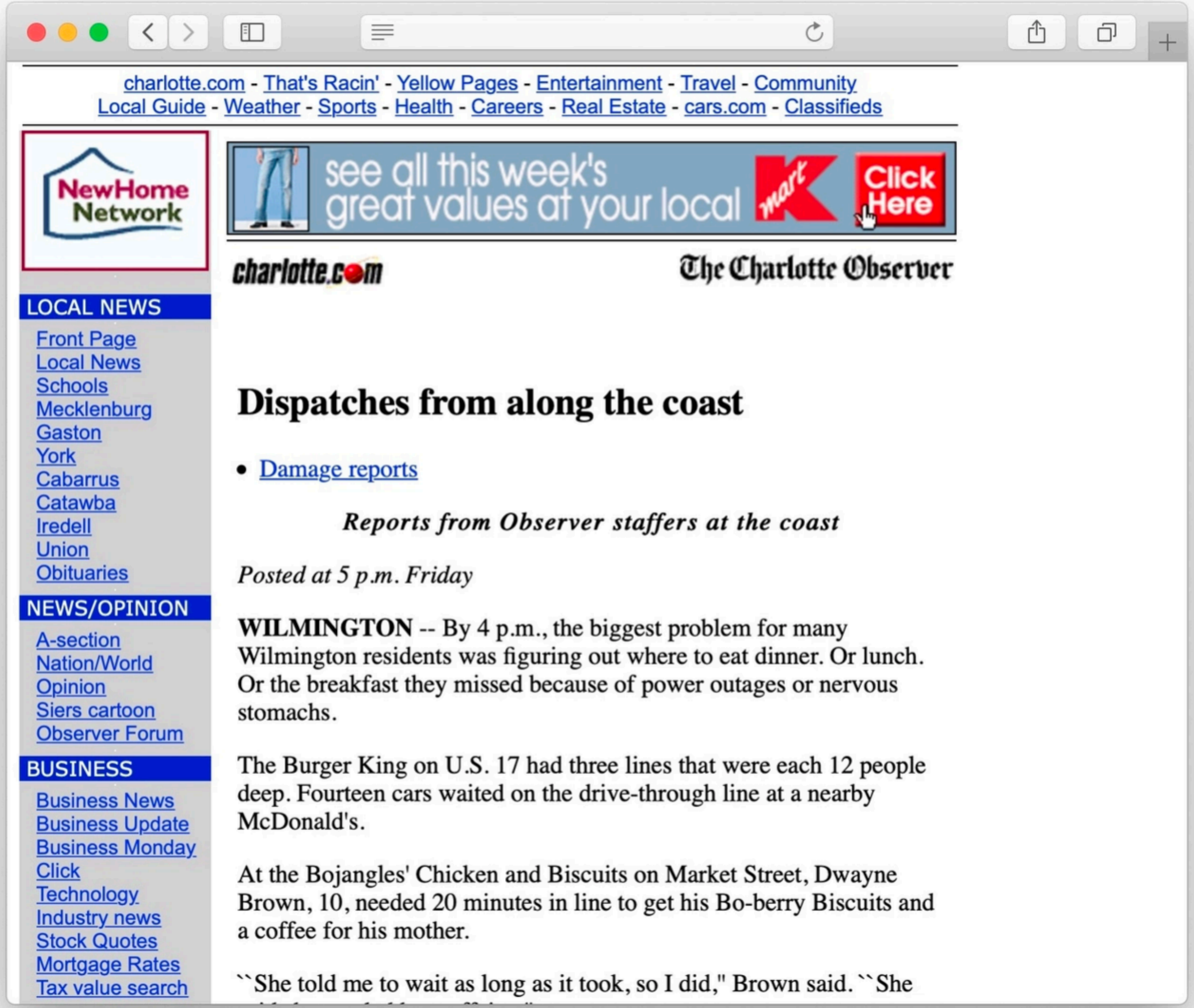
File Manager has a new look!
With our new design, it's easier than ever to upload, edit, and maintain your files, create subdirectories, and more.

Have you tried the new Yahoo! PageBuilder?
The preview version is here! If you can point and click with a

sourcetoad

Early Internet

- Blogs
- Message boards
- Universities
- News



charlotte.com - That's Racin' - Yellow Pages - Entertainment - Travel - Community
Local Guide - Weather - Sports - Health - Careers - Real Estate - cars.com - Classifieds

LOCAL NEWS
Front Page
Local News
Schools
Mecklenburg
Gaston
York
Cabarrus
Catawba
Iredell
Union
Obituaries

NEWS/OPINION
A-section
Nation/World
Opinion
Siers cartoon
Observer Forum

BUSINESS
Business News
Business Update
Business Monday
Click
Technology
Industry news
Stock Quotes
Mortgage Rates
Tax value search

Dispatches from along the coast

- Damage reports

Reports from Observer staffers at the coast

Posted at 5 p.m. Friday

WILMINGTON -- By 4 p.m., the biggest problem for many Wilmington residents was figuring out where to eat dinner. Or lunch. Or the breakfast they missed because of power outages or nervous stomachs.

The Burger King on U.S. 17 had three lines that were each 12 people deep. Fourteen cars waited on the drive-through line at a nearby McDonald's.

At the Bojangles' Chicken and Biscuits on Market Street, Dwayne Brown, 10, needed 20 minutes in line to get his Bo-berry Biscuits and a coffee for his mother.

``She told me to wait as long as it took, so I did,'' Brown said. ``She

The Present Internet

- Banking
- Health
- Shopping
- Everything

WELLS FARGO

Personal Small Business Commercial

Banking and Credit Cards Loans and Credit Investing and Retirement

View Your Accounts

Username

Password

Save username

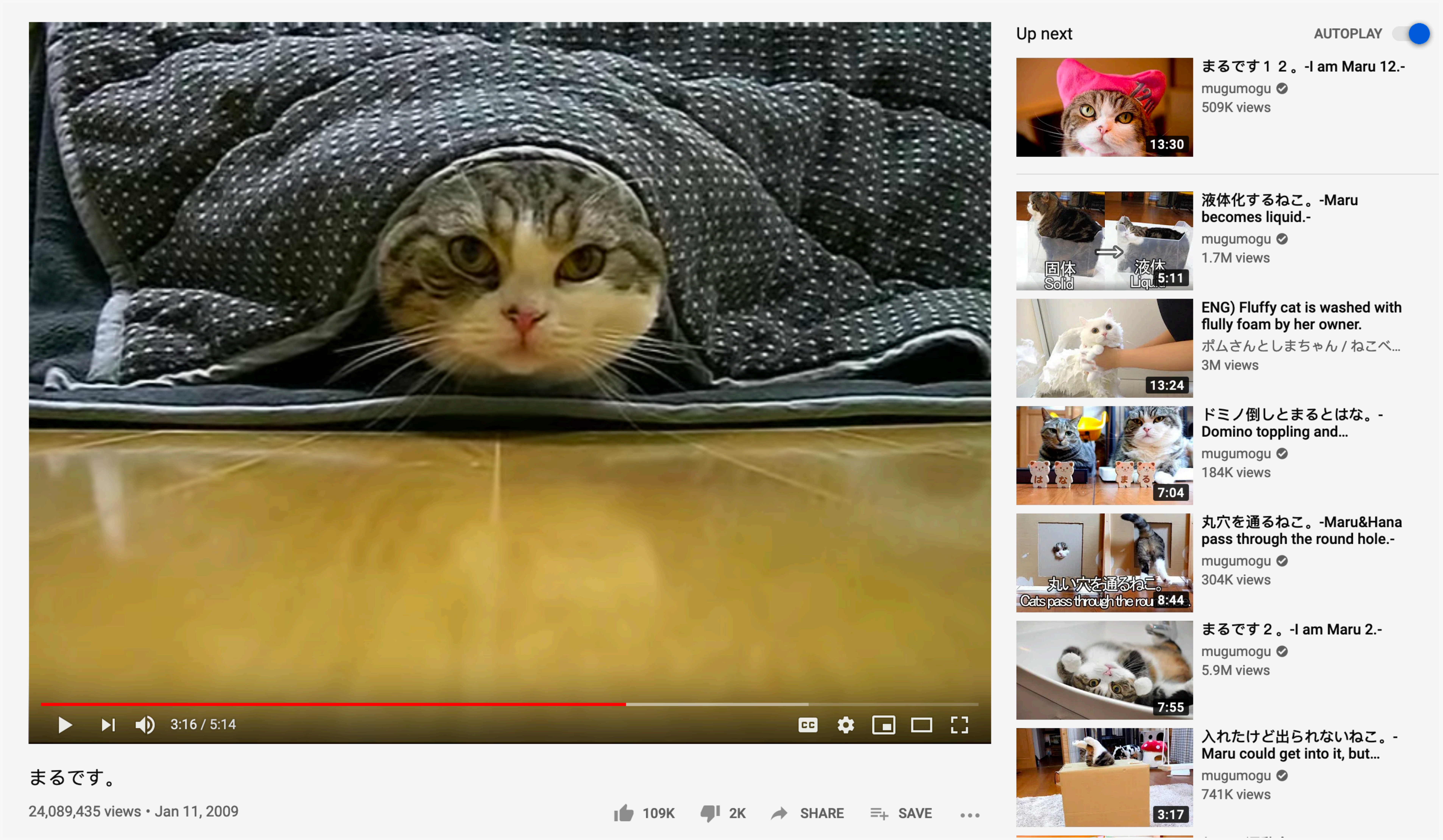
Sign On

Forgot Password/Username?

Enroll Now
Security Center
Privacy, Cookies, and Security

sourcetoad

The Real Internet



3:16 / 5:14

まるです。

24,089,435 views • Jan 11, 2009

109K 2K SHARE SAVE

Up next AUTOPLAY

まるです１２。-I am Maru 12.-
mugumogu
509K views
13:30

液体化するねこ。-Maru becomes liquid.-
mugumogu
1.7M views
5:11

ENG) Fluffy cat is washed with flully foam by her owner.
ポムさんとしまちゃん / ねこべ...
3M views
13:24

ドミノ倒しとまるとはな。-Domino toppling and...
mugumogu
184K views
7:04

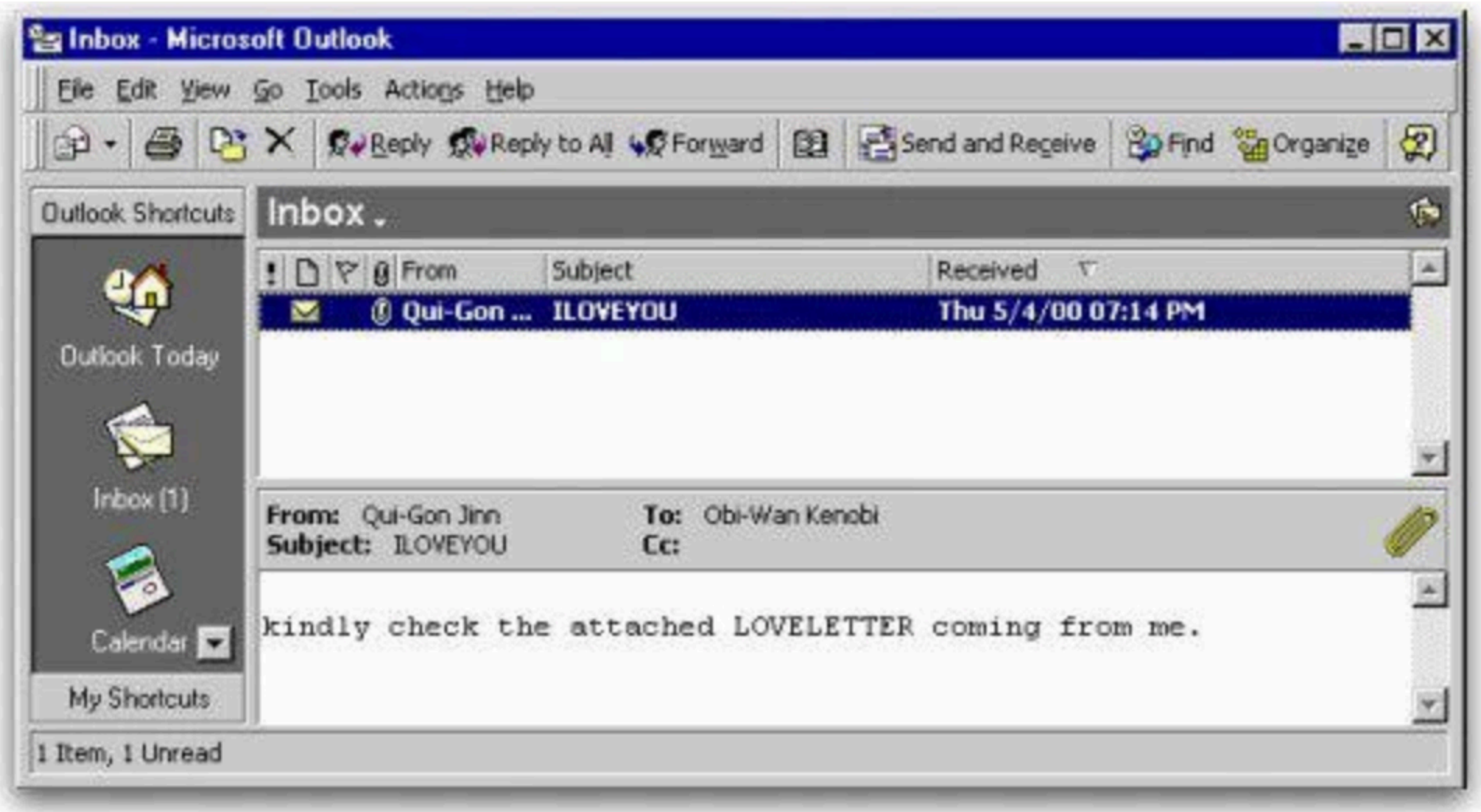
丸穴を通るねこ。-Maru&Hana pass through the round hole.-
mugumogu
304K views
8:44

まるです２。-I am Maru 2.-
mugumogu
5.9M views
7:55

入れたけど出られないねこ。-Maru could get into it, but...
mugumogu
741K views
3:17

The Malicious Web

- Internet users main purpose: abuse
- Protocols needed upgrades
- Developers needed teaching


Inbox - Microsoft Outlook
File Edit View Go Tools Actions Help
Reply Reply to All Forward Send and Receive Find Organize
Outlook Shortcuts
Inbox
Outlook Today
Inbox (1)
Calendar
My Shortcuts

From	Subject	Received
Qui-Gon ...	ILOVEYOU	Thu 5/4/00 07:14 PM

From: Qui-Gon Jinn To: Obi-Wan Kenobi
Subject: ILOVEYOU Cc:

```
kindly check the attached LOVELETTER coming from me.
```

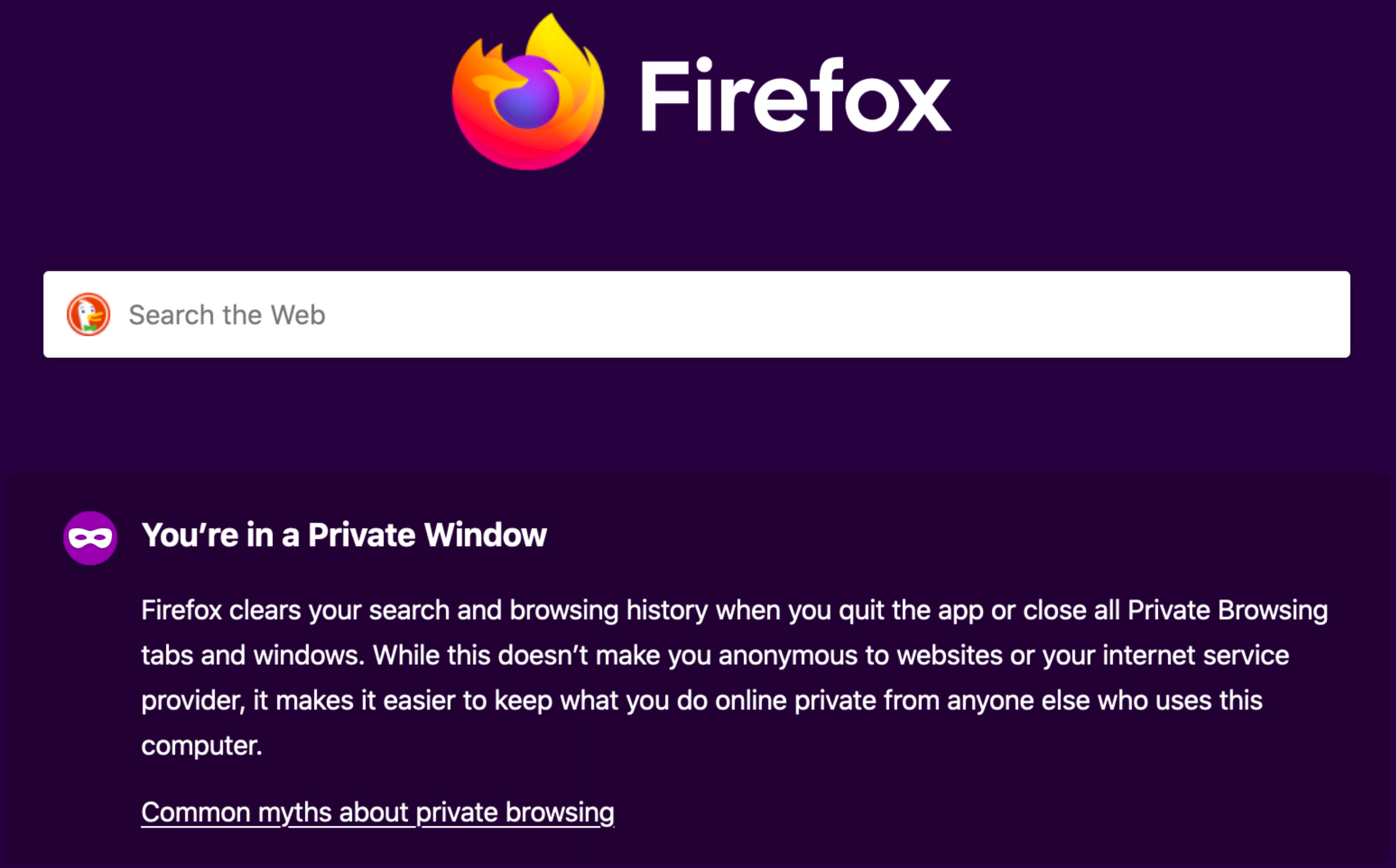
1 Item, 1 Unread

sourcetoad

So start small: XSS

- Cross-Site Scripting
  - CSS was taken, so XSS
  - (I made that up ^)
- Malicious code running on trusted website
- How does that happen though?

Browsers evaluate **HTML**. Simple.



Firefox

Search the Web

You're in a Private Window

Firefox clears your search and browsing history when you quit the app or close all Private Browsing tabs and windows. While this doesn't make you anonymous to websites or your internet service provider, it makes it easier to keep what you do online private from anyone else who uses this computer.

Common myths about private browsing

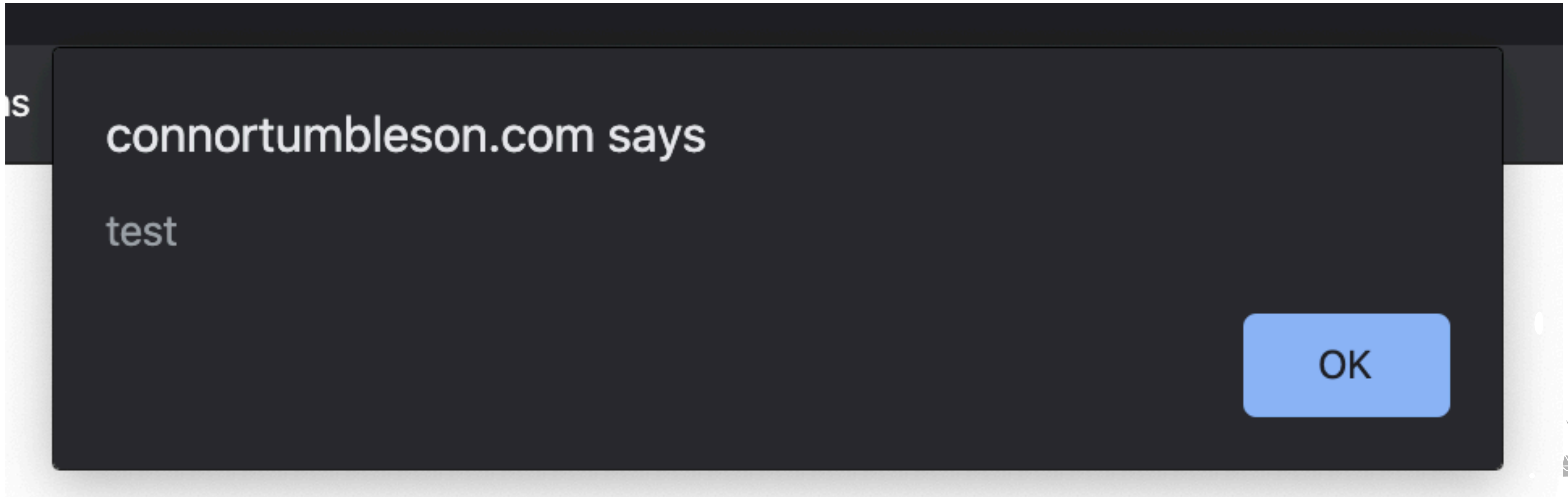
How do you inject code?

- UCG - User Generated Content
  - Comments, Forums, Contact Us etc
- URL Tweaking

`https://fakedemosite.com/search?query={searchTerm}`

How about an example

- Test bed: `<script>alert(‘test’);</script>`
- Place this anywhere
  - URL, Comment, Post, Searchbox



connortumbleson.com says

test

sourcetoad

The classic alert box.

- The quick test.
- If it works, then **untrusted code** can run.
- Then what?

It’s time to escalate.

sourcetoad

Common XSS Attacks

- Cookie Theft
  - document.cookie (session)
- Key-logging
  - onKeyPress (passwords)
- DOM Changes
  - action="malicious.host" (harvesting)

sourcetoad

Demo - Logging

```
msf5 auxiliary(server/capture/http_javascript_keylogger) > run

[*] Using URL: http://0.0.0.0:8080/mAZMnQE5jKADEO4
[*] Local IP: http://192.168.1.216:8080/mAZMnQE5jKADEO4
[*] Server started.
[+] [b3a6c63e] Logging clean keystrokes to: /root/.msf4/loot/202002
[+] [b3a6c63e] Logging raw keystrokes to: /root/.msf4/loot/20200215
[+] [b3a6c63e] Keys: c
[+] [b3a6c63e] Keys: co
[+] [b3a6c63e] Keys: con
[+] [b3a6c63e] Keys: conn
[+] [b3a6c63e] Keys: conno
[+] [b3a6c63e] Keys: connor
[+] [b3a6c63e] Keys: connor<TAB>
[+] [b3a6c63e] Keys: connor<TAB>p
[+] [b3a6c63e] Keys: connor<TAB>pa
[+] [b3a6c63e] Keys: connor<TAB>pas
[+] [b3a6c63e] Keys: connor<TAB>pass
[+] [b3a6c63e] Keys: connor<TAB>passw
[+] [b3a6c63e] Keys: connor<TAB>passwo
[+] [b3a6c63e] Keys: connor<TAB>passwor
[+] [b3a6c63e] Keys: connor<TAB>password
[+] [b3a6c63e] Keys: connor<TAB>password$
[+] [b3a6c63e] Keys: connor<TAB>password
```

Demo Form - Mozilla Firefox

192.168.1.216:8080/mAZMnQE5jKADEO4/demo

Keylogger Demo Form

This form submits data to the Metasploit listener for demonstration purposes.

Username: connor

Password:

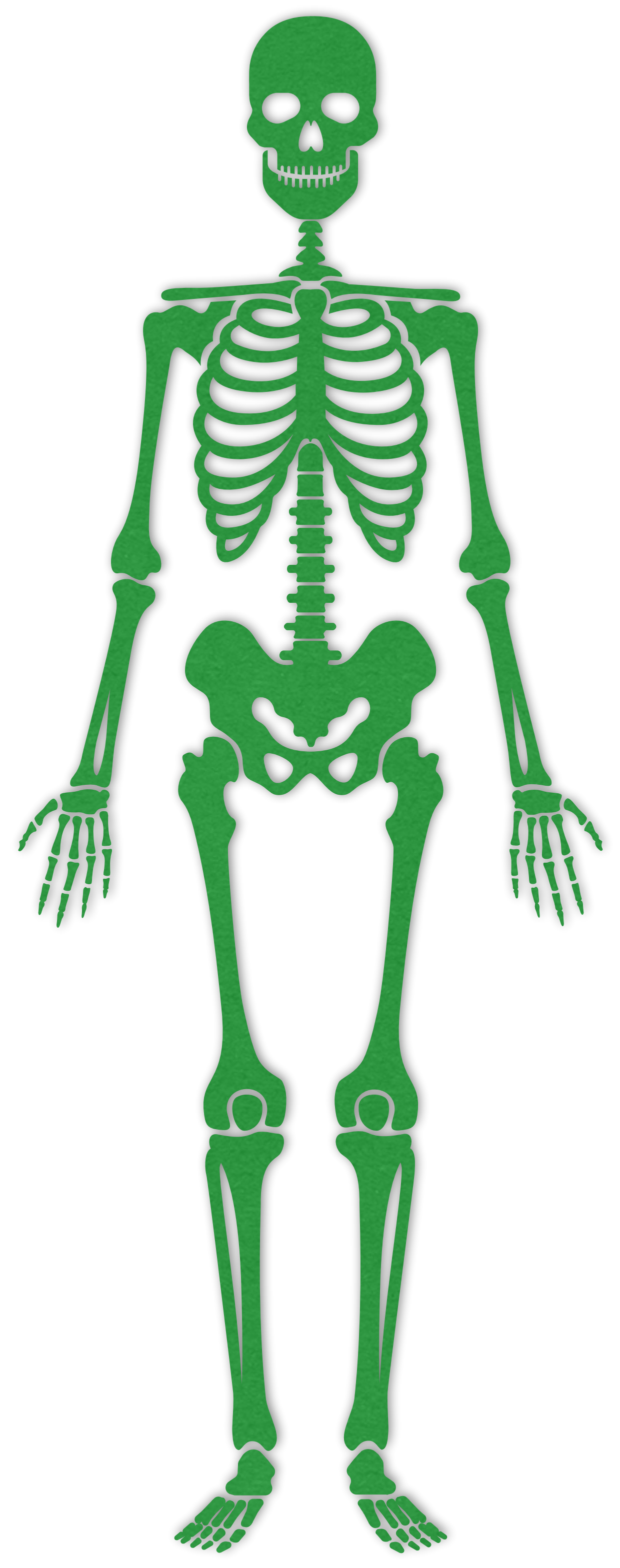
Submit

```
Keystrokes: connor<TAB>password
```

sourcetoad

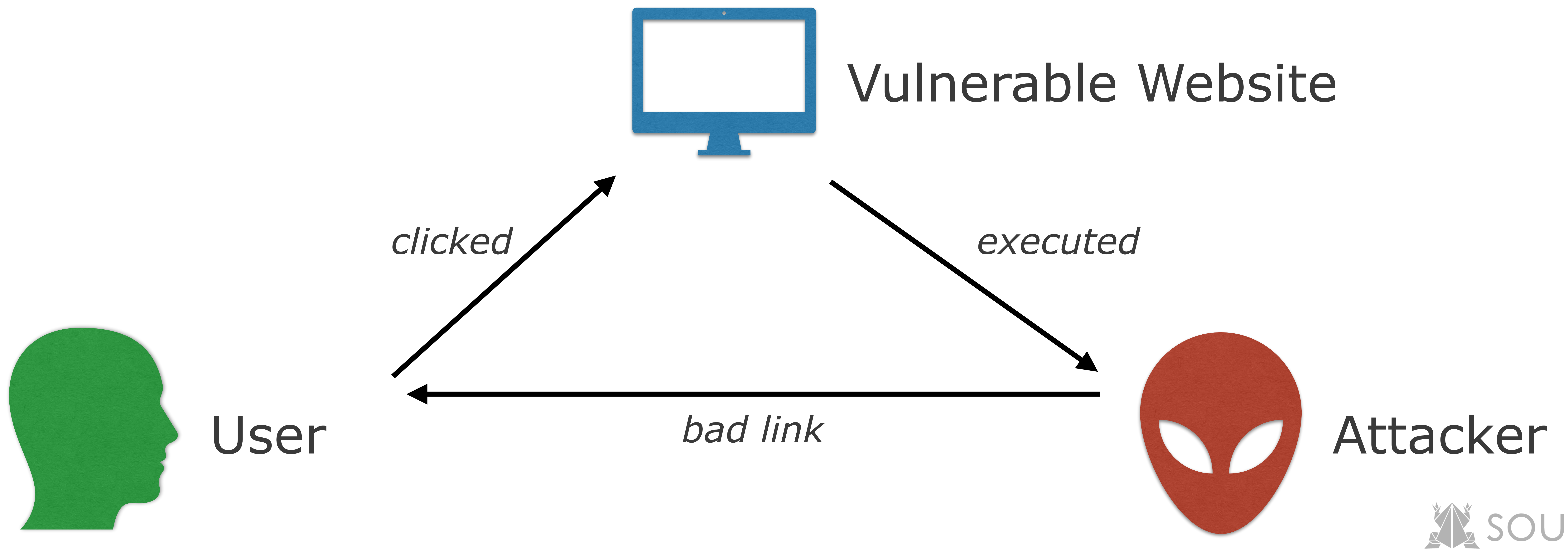
XSS Categories (Old)

- **Reflected** XSS
  - Think search or URL
- **Stored** XSS
  - Database, UCG
- **DOM** XSS
  - Frontend JS, “SPA”



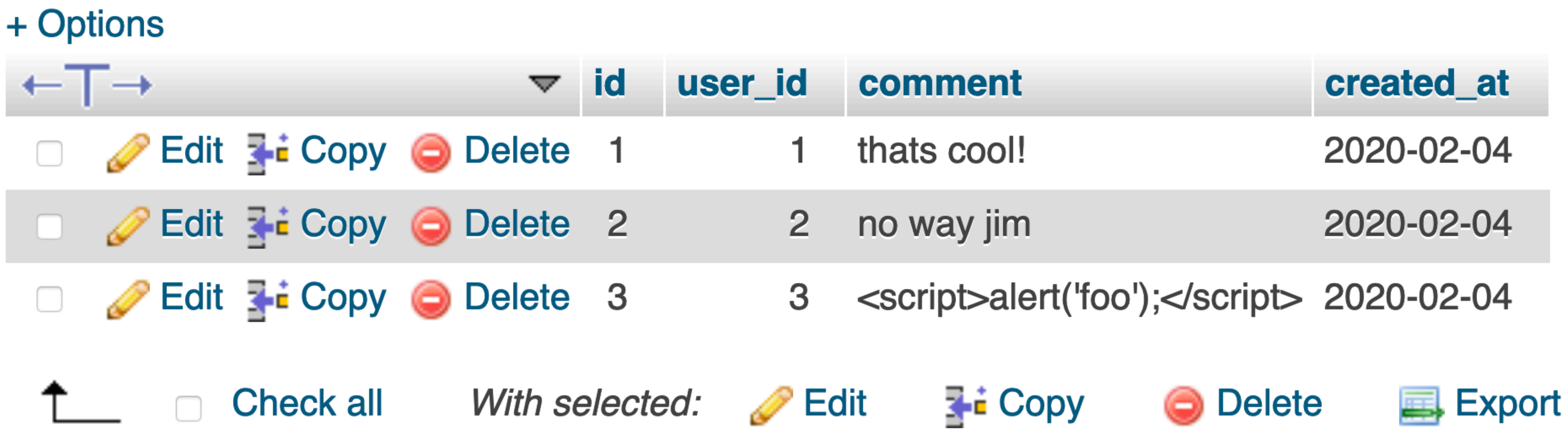
Reflected XSS

- Bad URL
- Trick someone to load



Stored XSS

- Untrusted data in DB
- Emitted into page
- Many could be affected



	id	user_id	comment	created_at
Edit Copy Delete	1	1	thats cool!	2020-02-04
Edit Copy Delete	2	2	no way jim	2020-02-04
Edit Copy Delete	3	3	<script>alert('foo');</script>	2020-02-04

DOM XSS

- DOM changes based on input
- Two way binding - Vue/Angular/React

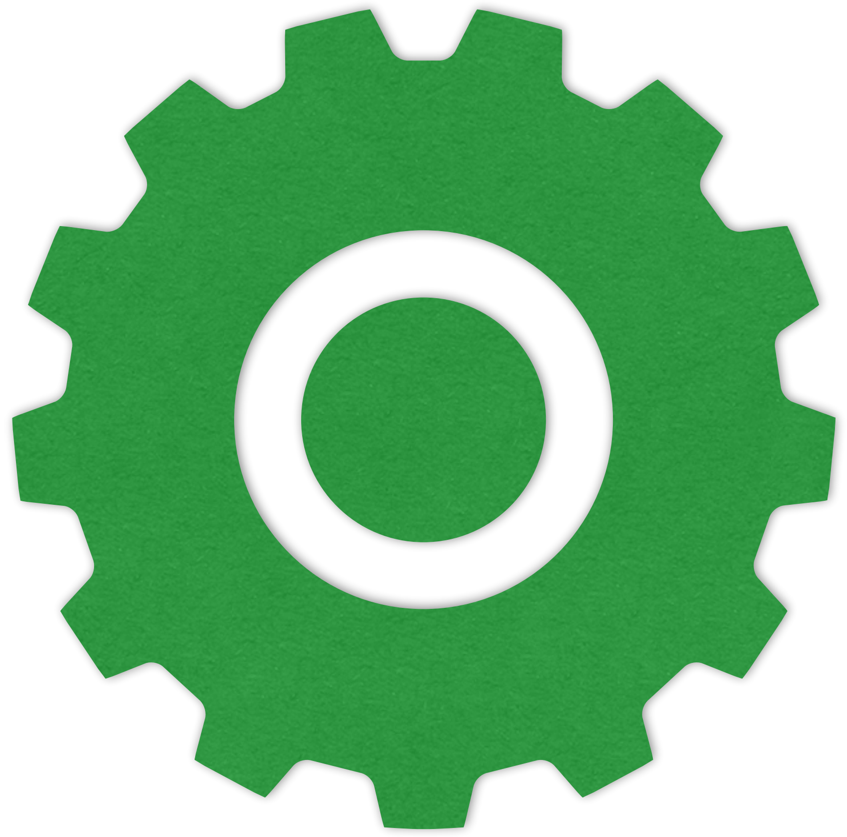
```js
new Vue({
  el: '#app',
  template: `<div>` + userProvidedString + `</div>` // NEVER DO THIS
})
```

XSS Categories (Modern)

- **Server** XSS
  - Untrusted data comes from server
- **Client** XSS
  - Untrusted data lives at DOM layer
  - AJAX, SPA, etc

Prevention Techniques (XSS)

- Escaping
- Filter
- HTTP Headers
- httpOnly
- CSP Rules



sourcetoad

Prevention: Escaping (preferred)

- Browsers don’t parse text twice.
- So script tags are never processed

```
& --> &amp;
< --> &lt;
> --> &gt;
" --> &quot;
' --> &#x27;
/ --> &#x2F;
```

Prevention: Escaping (preferred)

```
<script>alert(‘foo’);</script>
```

Escaped (you)

```
&tl;script&gt;alert(&#x27;foo&#x27;);&lt;&#x2F;script&gt;
```

Prevention: Escaping (preferred)

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

↑ Rendered (browser)

```
&tl;script&gt;alert(&#x27;foo&#x27;);&lt;&#x2F;script&gt;
```

Prevention: Filter (not preferred)

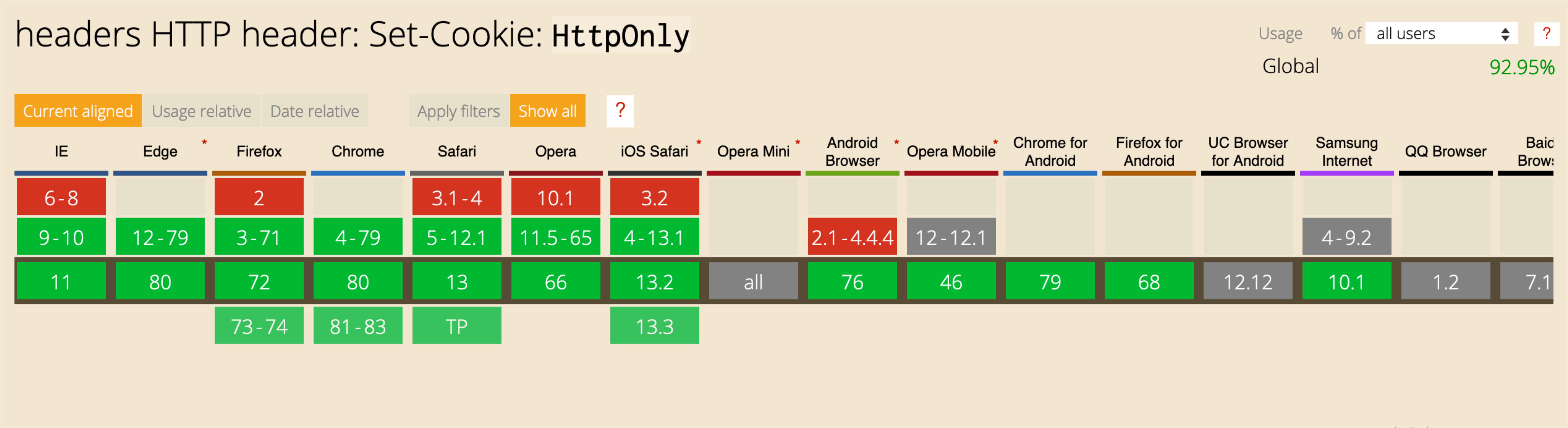
- Guide what you expect
- Validation
- “What is your name?”
  - `Connor <script>hack you</script>`

Prevention: Headers (abandoned)

- `X-XSS-Protection` HTTP Header
  - If URL matches executed JS, then block
- Only protects **Reflected XSS**
- Browsers dropping in favor of CSP rules

Prevention: Cookie Setting (partial)

- `httpOnly` flag when creating cookie
- Prevents cookie being read client side
- (if browser supports it)



headers HTTP header: Set-Cookie: `HttpOnly`

Usage % of all users ?

Global 92.95%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

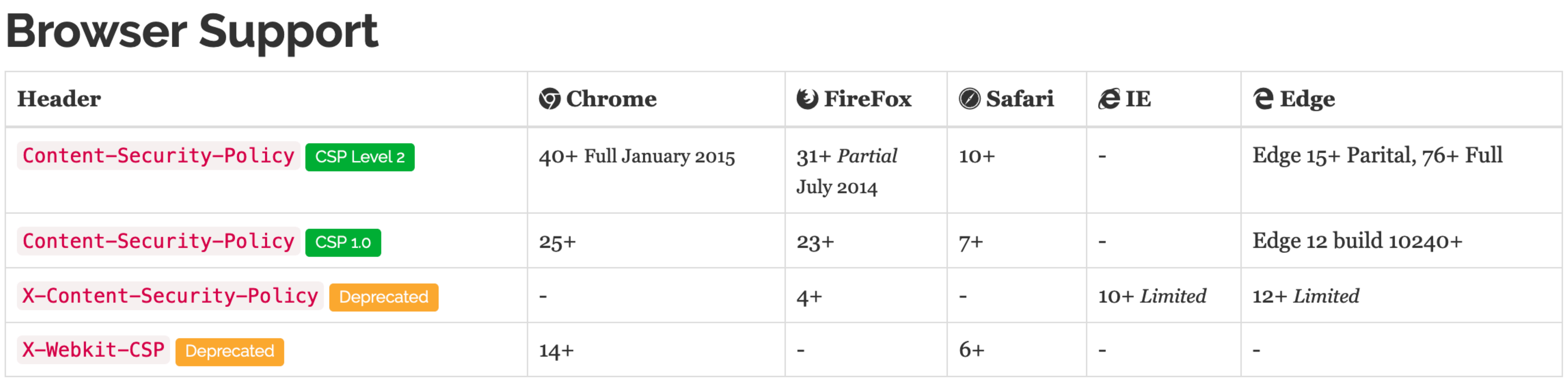
IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baid Brows
6-8		2		3.1-4	10.1	3.2									
9-10	12-79	3-71	4-79	5-12.1	11.5-65	4-13.1		2.1-4.4.4	12-12.1				4-9.2		
11	80	72	80	13	66	13.2	all	76	46	79	68	12.12	10.1	1.2	7.1
		73-74	81-83	TP		13.3									

sourcetoad

https://caniuse.com/#search=httpOnly

Prevention: CSP (future)

- **C**ontent **S**ecurity **P**olicy
- A complex header to protect end users
- Yes, it is complex.



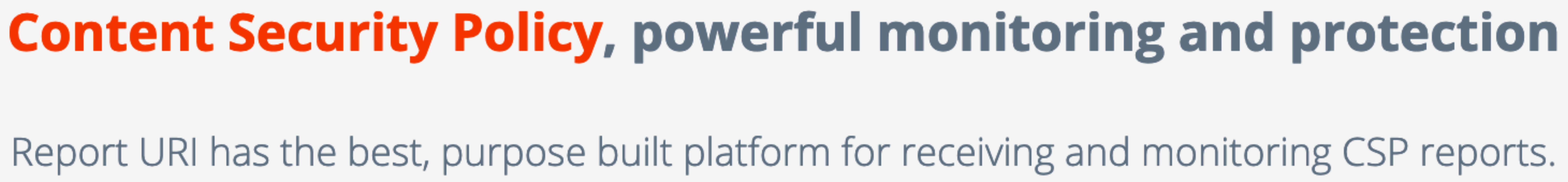
Browser Support

Header	Chrome	FireFox	Safari	IE	Edge
`Content-Security-Policy` CSP Level 2	40+ Full January 2015	31+ *Partial* July 2014	10+	-	Edge 15+ Parital, 76+ Full
`Content-Security-Policy` CSP 1.0	25+	23+	7+	-	Edge 12 build 10240+
`X-Content-Security-Policy` Deprecated	-	4+	-	10+ *Limited*	12+ *Limited*
`X-Webkit-CSP` Deprecated	14+	-	6+	-	-

sourcetoad

Prevention: CSP cont.

- Only load images from x.com
- Refuse to load inline Javascript
- AJAX Requests only to “self”
- Block or ignore violations



Content Security Policy, powerful monitoring and protection

Report URI has the best, purpose built platform for receiving and monitoring CSP reports.

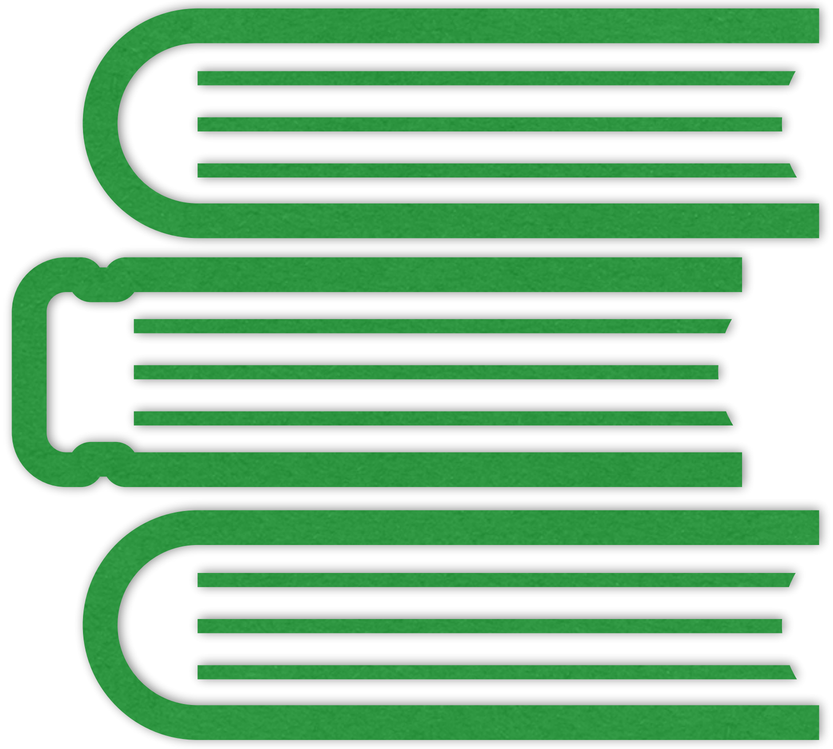
https://report-uri.com

sourcetoad

Switching to **CSRF**

CSRF - Intro

- **C**ross **S**ite **R**equest **F**orgery
- Executing a request in an unwanted way
- Imagine submitting a form maliciously
- Fake Story Time…



CSRF - Early Internet

- Lets say we all bank with *{bank}*
- I send $5 to a friend on their website
- I notice the URL is
  - GET *bank.com/transfer?**acct**=**Friend**&**amt**=**$5***

sourcetoad

CSRF - Early Abuse

- GET probably wasn't used.
- I notice pattern.
- I change the link to me.
- Victim clicks link, they send me $5
  - `<a href="http://badlink">View Photos</a>`

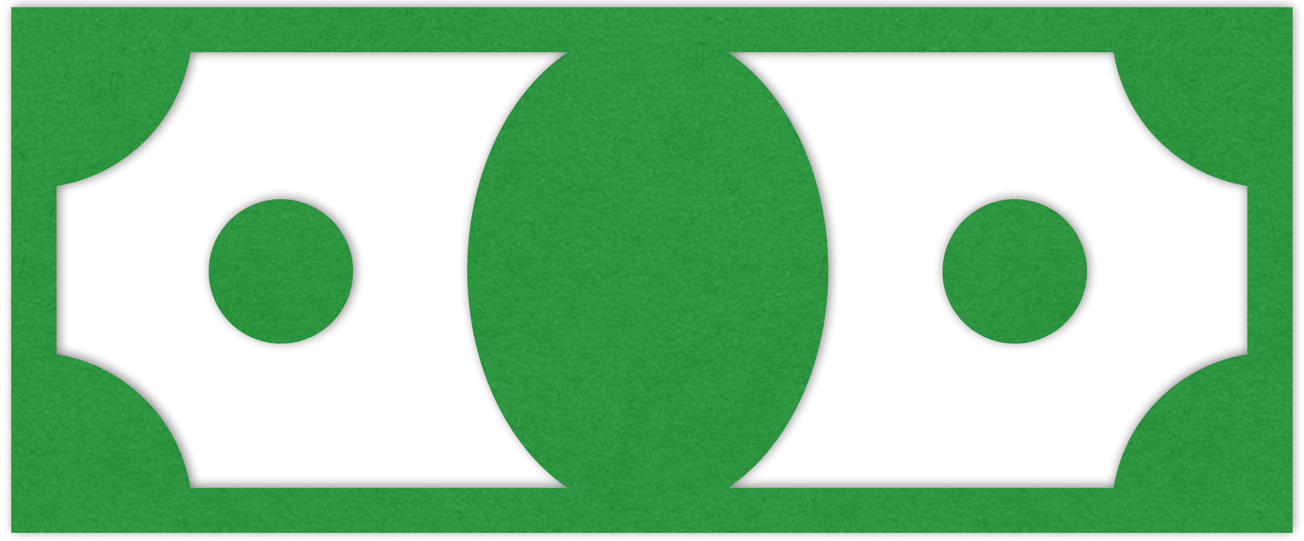
CSRF - Early Abuse

- Yeah that was too easy.
- The world actually used POST

```
<form action="bank.com/transfer">
  <input name="target" value="friend" />
  <input name="amt" value="5" />
  <button type="submit" value="Send" />
</form>
```

CSRF - POST Abuse

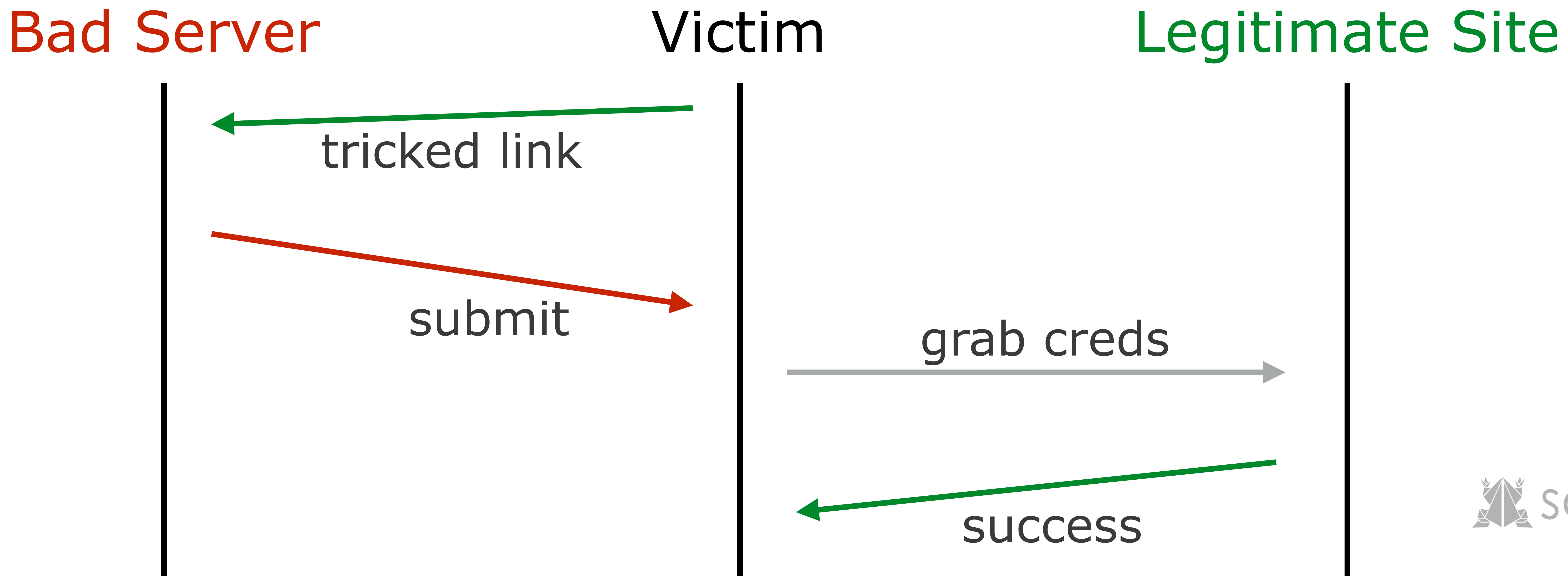
- I make a comment section on my website
- It also submits a hidden form to *{bank}*
- If visitor banks with *{bank}* then
  - makes a comment
  - I just got $5 from them



sourcetoad

CSRF - Wait. How did that work?

- The victim is logged in with *{bank}*
- Browser can't tell if legit or not
- Browser makes request



CSRF - POST Prevention Early Web

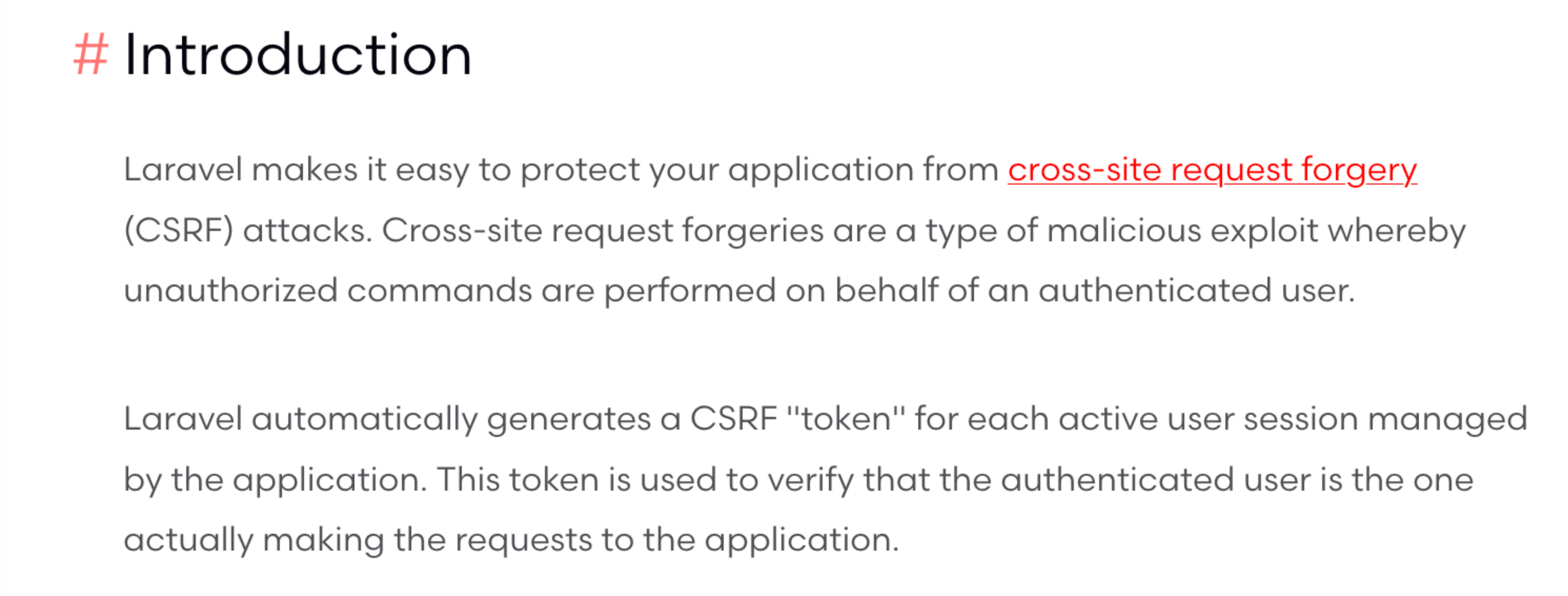
- Bank has noticed this abuse.
- They start relying on referrer.
- HTTP Header
- Transfers MUST have referrer of
    - `http://bank.com/manage`

CSRF - The Referrer Problem

- Leaks information
- May be empty or missing
- Referrer may be
  - `http://company.com/sekrit/x-pod-90-pro`

CSRF - The Token Fix

- Lets make a random string
- Put it on form, look for it during submit


Introduction

Laravel makes it easy to protect your application from cross-site request forgery (CSRF) attacks. Cross-site request forgeries are a type of malicious exploit whereby unauthorized commands are performed on behalf of an authenticated user.

Laravel automatically generates a CSRF "token" for each active user session managed by the application. This token is used to verify that the authenticated user is the one actually making the requests to the application.

sourcetoad

CSRF - The Token Fix

- If someone makes a forged request
- It cannot have the token
- Thus, **denied**.
- Normally, HTTP 419 (*Auth Timeout*)

Advanced Time

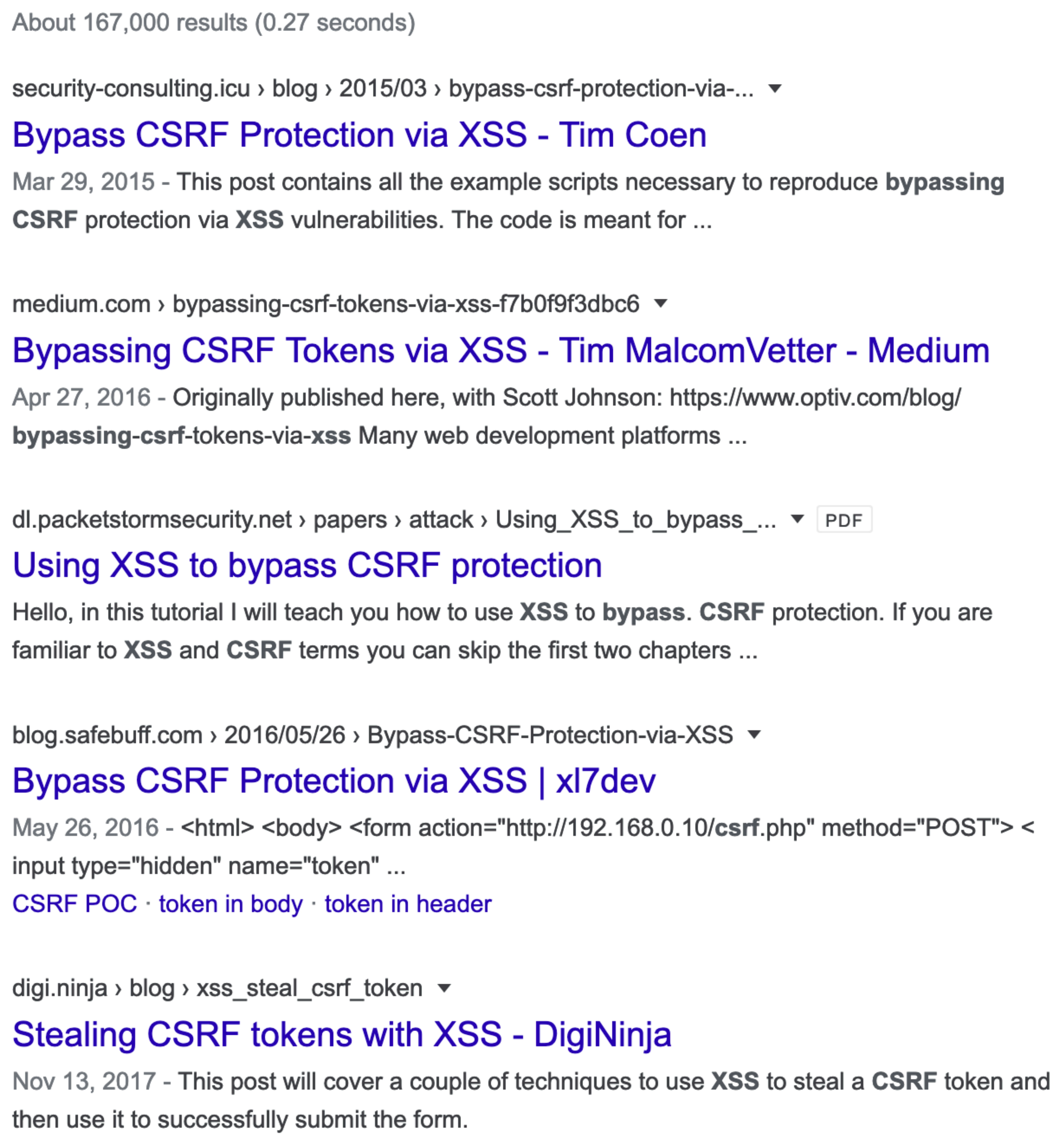
CSRF - Why batched with XSS?

- XSS attack bypasses **ALL** CSRF measures
- Load the page, find the token
- Load the token into malicious form
- Submit the form
- Pivoted XSS -> CSRF

sourcetoad

Bypass CSRF

- Google Results
- 167k
- Tons of methods



About 167,000 results (0.27 seconds)

security-consulting.icu › blog › 2015/03 › bypass-csrf-protection-via-... ▾

Bypass CSRF Protection via XSS - Tim Coen

Mar 29, 2015 - This post contains all the example scripts necessary to reproduce **bypassing CSRF** protection via **XSS** vulnerabilities. The code is meant for ...

medium.com › bypassing-csrf-tokens-via-xss-f7b0f9f3dbc6 ▾

Bypassing CSRF Tokens via XSS - Tim MalcomVetter - Medium

Apr 27, 2016 - Originally published here, with Scott Johnson: https://www.optiv.com/blog/**bypassing-csrf**-tokens-via-**xss** Many web development platforms ...

dl.packetstormsecurity.net › papers › attack › Using_XSS_to_bypass_... ▾ PDF

Using XSS to bypass CSRF protection

Hello, in this tutorial I will teach you how to use **XSS** to **bypass**. **CSRF** protection. If you are familiar to **XSS** and **CSRF** terms you can skip the first two chapters ...

blog.safebuff.com › 2016/05/26 › Bypass-CSRF-Protection-via-XSS ▾

Bypass CSRF Protection via XSS | xl7dev

May 26, 2016 - <html> <body> <form action="http://192.168.0.10/**csrf**.php" method="POST"> < input type="hidden" name="token" ...

CSRF POC · token in body · token in header

digi.ninja › blog › xss_steal_csrf_token ▾

Stealing CSRF tokens with XSS - DigiNinja

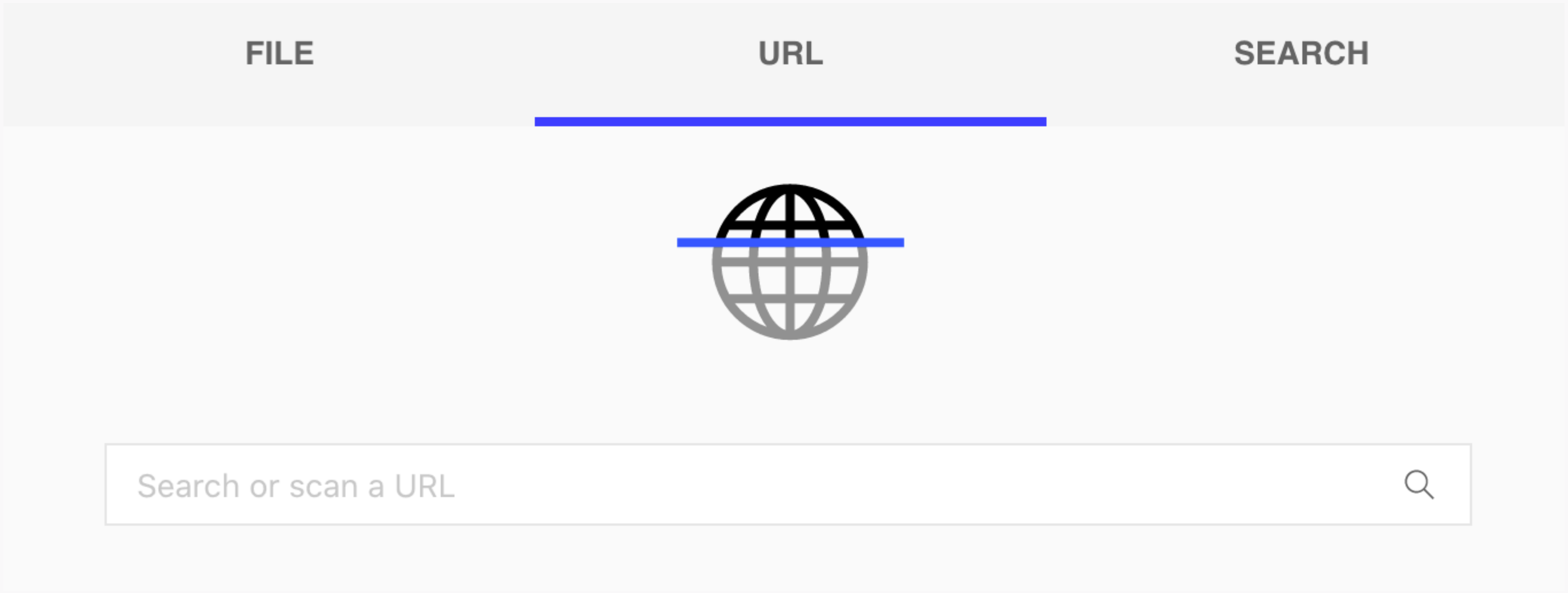
Nov 13, 2017 - This post will cover a couple of techniques to use **XSS** to steal a **CSRF** token and then use it to successfully submit the form.

SSRF - What is that?

- **S**SRF - Server
- **S**erver **S**ide **R**equest **F**orgery
- So forging a request from a server.

SSRF - Example

- Upload file or give URL



sourcetoad

SSRF - Example

- If you put in URL - https://ibotpeaches.com/imgs/yer.jpg
- Server downloads it.
- Maybe because of CSP rules
  - Can’t load 3rd party images
- So what happens?

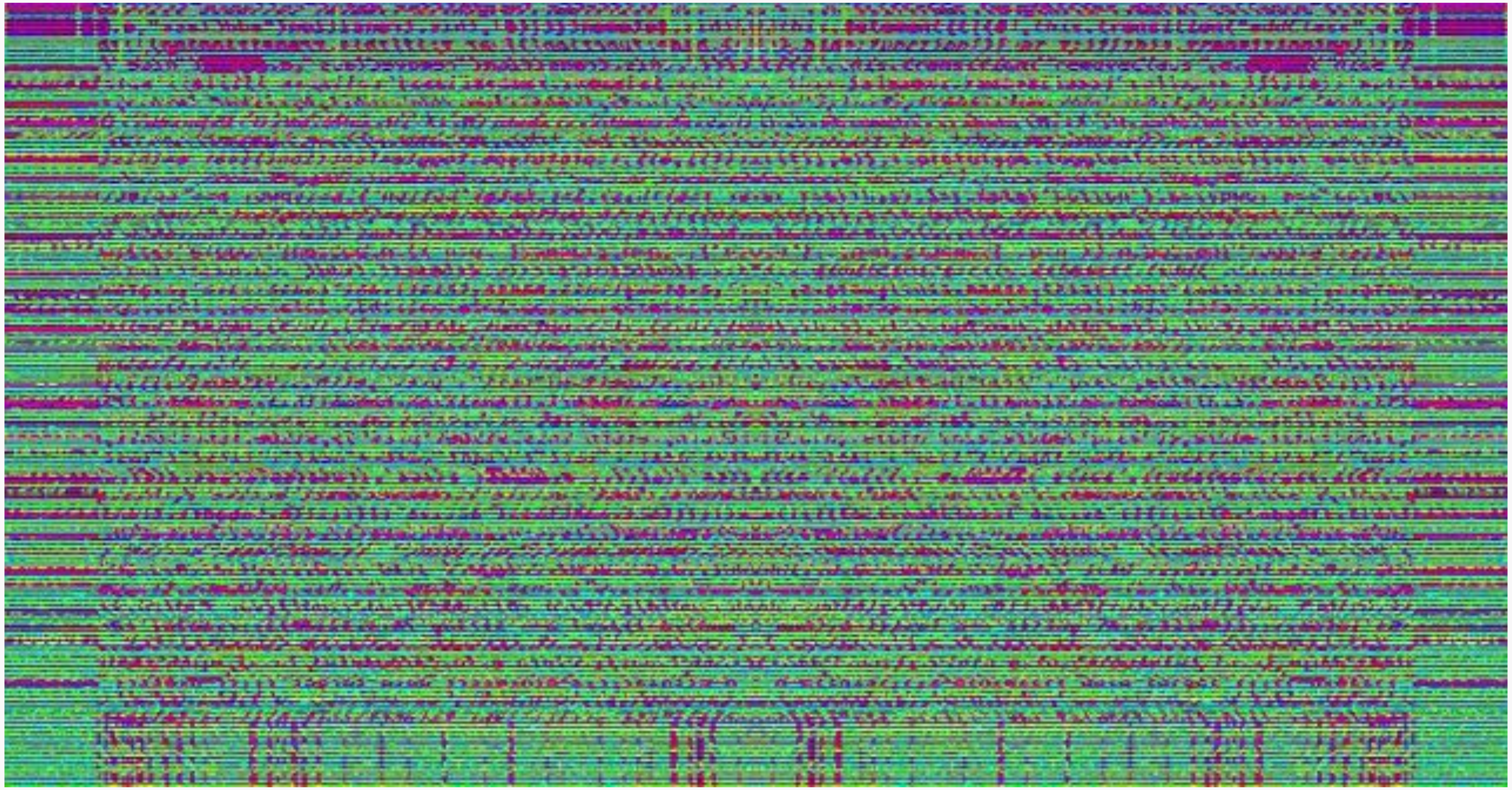
SSRF - Intended Flow

SSRF - Malicious Flow

- If you put in URL - http://127.0.0.1/nginx_status
- Status page for NGINX (default)
- Server reaches out.
- Downloads it.

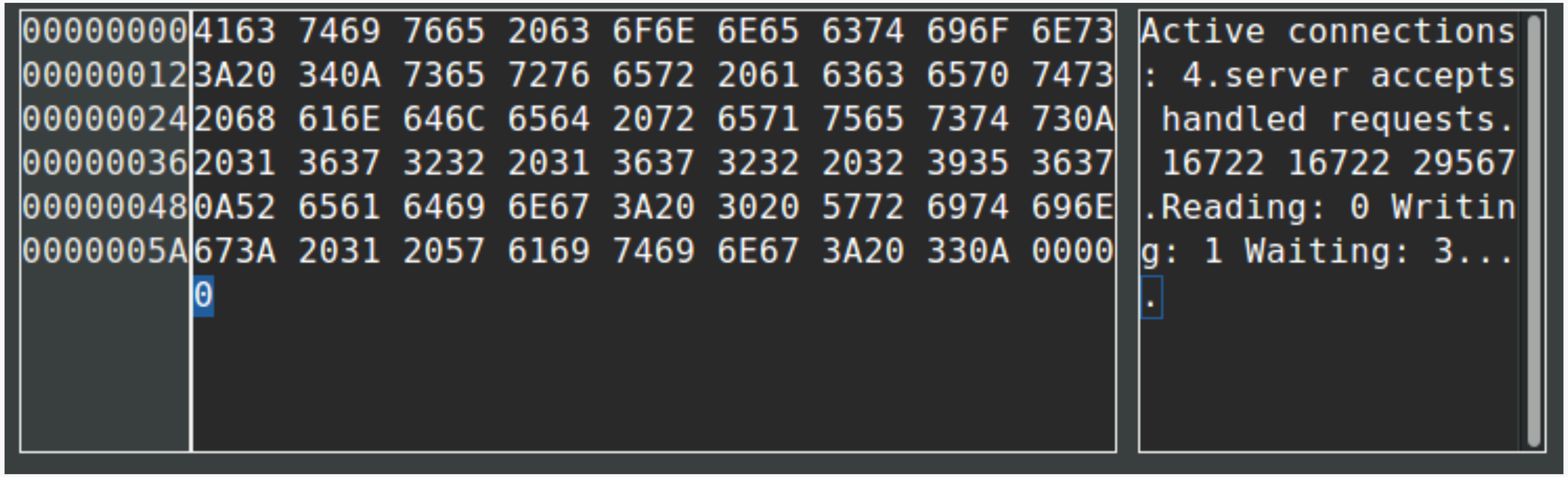
SSRF - Malicious Flow

- hmm...



SSRF - Malicious Flow

- That can’t be rendered as an image
- Assuming no file validation
- What actually is it?



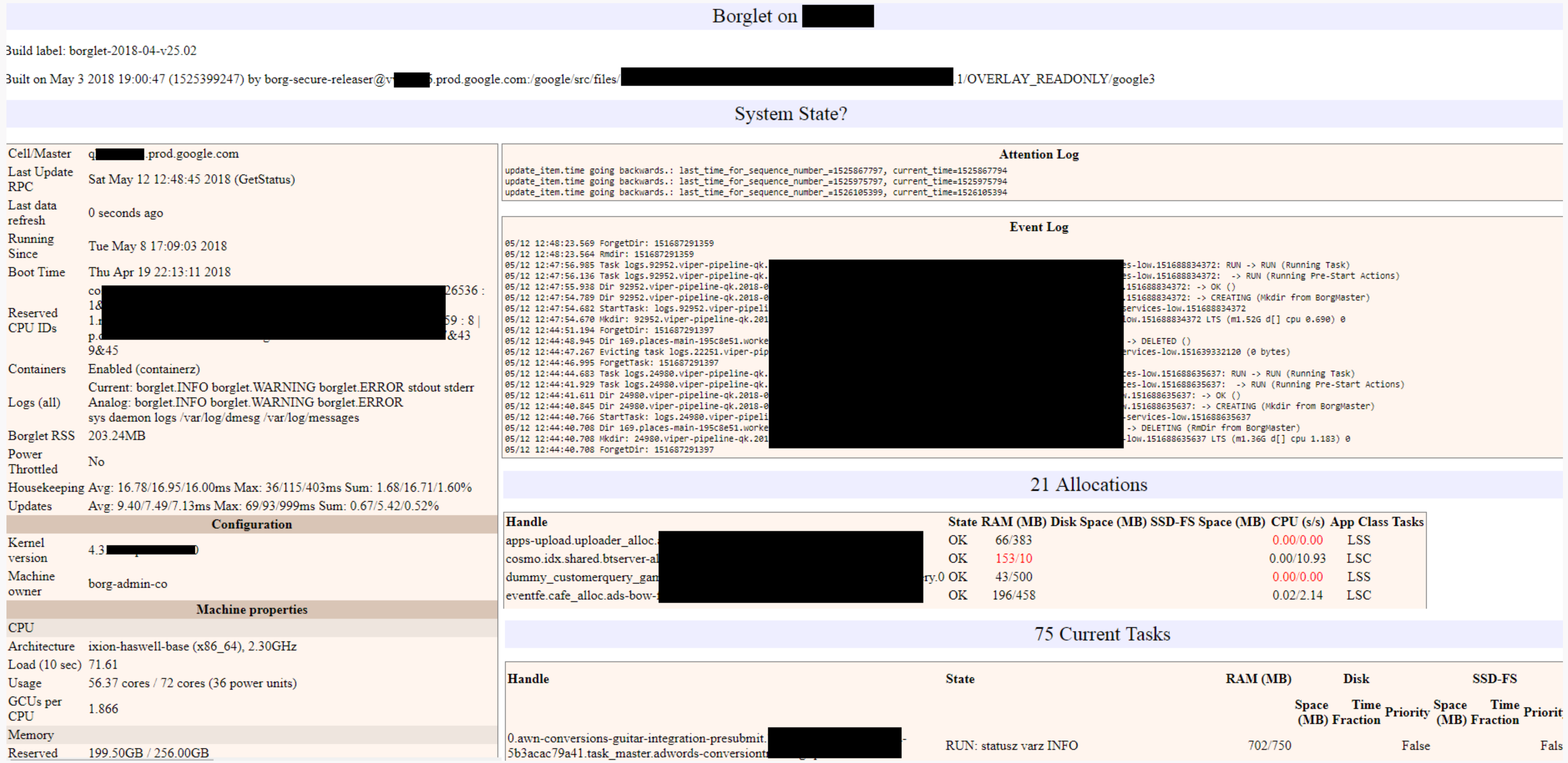
```
00000000 4163 7469 7665 2063 6F6E 6E65 6374 696F 6E73  Active connections
00000012 3A20 340A 7365 7276 6572 2061 6363 6570 7473  : 4.server accepts
00000024 2068 616E 646C 6564 2072 6571 7565 7374 730A   handled requests.
00000036 2031 3637 3232 2031 3637 3232 2032 3935 3637   16722 16722 29567
00000048 0A52 6561 6469 6E67 3A20 3020 5772 6974 696E  .Reading: 0 Writin
0000005A 673A 2031 2057 6169 7469 6E67 3A20 330A 0000  g: 1 Waiting: 3...
         0                                             .
```

sourcetoad

SSRF - Complete

- **Wow**
- Tricked a server
- To download a local (internal) file and return it to me.

SSRF - In Real Life (Google)



Borglet on [redacted]

Build label: borglet-2018-04-v25.02

Built on May 3 2018 19:00:47 (1525399247) by borg-secure-releaser@v[redacted].prod.google.com:/google/src/files/[redacted].1/OVERLAY_READONLY/google3

System State?

Cell/Master	q[redacted].prod.google.com
Last Update RPC	Sat May 12 12:48:45 2018 (GetStatus)
Last data refresh	0 seconds ago
Running Since	Tue May 8 17:09:03 2018
Boot Time	Thu Apr 19 22:13:11 2018
Reserved CPU IDs	co[redacted]26536 : 1&[redacted] 1.r[redacted]59 : 8 | p.c[redacted]&43 9&45
Containers	Enabled (containerz)
Logs (all)	Current: borglet.INFO borglet.WARNING borglet.ERROR stdout stderr Analog: borglet.INFO borglet.WARNING borglet.ERROR sys daemon logs /var/log/dmesg /var/log/messages
Borglet RSS	203.24MB
Power Throttled	No
Housekeeping	Avg: 16.78/16.95/16.00ms Max: 36/115/403ms Sum: 1.68/16.71/1.60%
Updates	Avg: 9.40/7.49/7.13ms Max: 69/93/999ms Sum: 0.67/5.42/0.52%
Configuration	
Kernel version	4.3[redacted]0
Machine owner	borg-admin-co
Machine properties	
CPU	
Architecture	ixion-haswell-base (x86_64), 2.30GHz
Load (10 sec)	71.61
Usage	56.37 cores / 72 cores (36 power units)
GCUs per CPU	1.866
Memory	
Reserved	199.50GB / 256.00GB

Attention Log

```
update_item.time going backwards.: last_time_for_sequence_number_=1525867797, current_time=1525867794
update_item.time going backwards.: last_time_for_sequence_number_=1525975797, current_time=1525975794
update_item.time going backwards.: last_time_for_sequence_number_=1526105399, current_time=1526105394
```

Event Log

```
05/12 12:48:23.569 ForgetDir: 151687291359
05/12 12:48:23.564 Rmdir: 151687291359
05/12 12:47:56.985 Task logs.92952.viper-pipeline-qk.[redacted]es-low.151688834372: RUN -> RUN (Running Task)
05/12 12:47:56.136 Task logs.92952.viper-pipeline-qk.         es-low.151688834372:  -> RUN (Running Pre-Start Actions)
05/12 12:47:55.938 Dir 92952.viper-pipeline-qk.2018-0         .151688834372: -> OK ()
05/12 12:47:54.789 Dir 92952.viper-pipeline-qk.2018-0         .151688834372: -> CREATING (Mkdir from BorgMaster)
05/12 12:47:54.682 StartTask: logs.92952.viper-pipeli         services-low.151688834372
05/12 12:47:54.670 Mkdir: 92952.viper-pipeline-qk.201         low.151688834372 LTS (m1.52G d[] cpu 0.690) 0
05/12 12:44:51.194 ForgetDir: 151687291397
05/12 12:44:48.945 Dir 169.places-main-195c8e51.worke          -> DELETED ()
05/12 12:44:47.267 Evicting task logs.22251.viper-pip         ervices-low.151639332120 (0 bytes)
05/12 12:44:46.995 ForgetTask: 151687291397
05/12 12:44:44.683 Task logs.24980.viper-pipeline-qk.         ces-low.151688635637: RUN -> RUN (Running Task)
05/12 12:44:41.929 Task logs.24980.viper-pipeline-qk.         ces-low.151688635637:  -> RUN (Running Pre-Start Actions)
05/12 12:44:41.611 Dir 24980.viper-pipeline-qk.2018-0         w.151688635637: -> OK ()
05/12 12:44:40.845 Dir 24980.viper-pipeline-qk.2018-0         w.151688635637: -> CREATING (Mkdir from BorgMaster)
05/12 12:44:40.766 StartTask: logs.24980.viper-pipeli         -services-low.151688635637
05/12 12:44:40.708 Dir 169.places-main-195c8e51.worke          -> DELETING (RmDir from BorgMaster)
05/12 12:44:40.708 Mkdir: 24980.viper-pipeline-qk.201         -low.151688635637 LTS (m1.36G d[] cpu 1.183) 0
05/12 12:44:40.708 ForgetDir: 151687291397
```

21 Allocations

Handle	State	RAM (MB)	Disk Space (MB)	SSD-FS Space (MB)	CPU (s/s)	App Class	Tasks
apps-upload.uploader_alloc.[redacted]	OK	66/383			0.00/0.00	LSS	
cosmo.idx.shared.btserver-al[redacted]	OK	153/10			0.00/10.93	LSC	
dummy_customerquery_gan[redacted]ry.0	OK	43/500			0.00/0.00	LSS	
eventfe.cafe_alloc.ads-bow-[redacted]	OK	196/458			0.02/2.14	LSC	

75 Current Tasks

Handle	State	RAM (MB)	Disk			SSD-FS		
			Space (MB)	Time Fraction	Priority	Space (MB)	Time Fraction	Priorit
0.awn-conversions-guitar-integration-presubmit.[redacted]- 5b3acac79a41.task_master.adwords-conversiont	RUN: statusz varz INFO	702/750			False			Fals

https://opnsec.com/2018/07/into-the-borg-ssrf-inside-google-production-network/

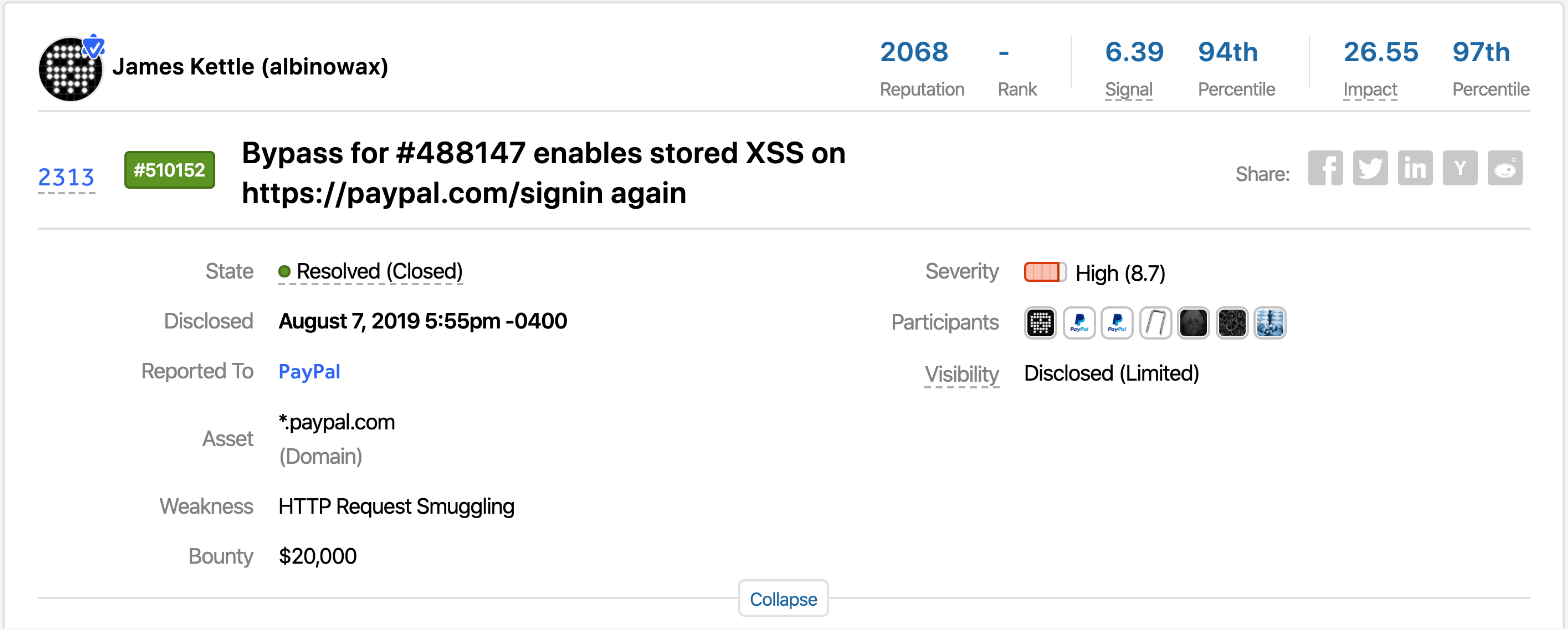
SSRF - In Real Life (Google)

- Google Caja “*cleans*” HTML/CSS/JS
- Needs to download and do magic
- Author noticed downloads came from internal network

https://opnsec.com/2018/07/into-the-borg-ssrf-inside-google-production-network/

sourcetoad

Bounties



James Kettle (albinowax)

2068	-	6.39	94th	26.55	97th
Reputation	Rank	Signal	Percentile	Impact	Percentile

2313 #510152 **Bypass for #488147 enables stored XSS on https://paypal.com/signin again**

Share:

State	● Resolved (Closed)
Disclosed	**August 7, 2019 5:55pm -0400**
Reported To	PayPal
Asset	*.paypal.com (Domain)
Weakness	HTTP Request Smuggling
Bounty	$20,000
Severity	High (8.7)
Participants	
Visibility	Disclosed (Limited)

Collapse

sourcetoad

Concluding

- XSS is top 10 OWASP still
- Stay with frameworks for CSRF protection
- SSRF is a real thing
- Don’t roll your own escaping

Thanks!

connortumbleson.com

@iBotPeaches

sourcetoad