

Web Hacking for Social Good

Aditya Saligrama
Cooper de Nicola

Security and its relevance to Stanford students

Case study: Stanford Link (2020)

Link. 
Shoot your shot, Stanford.

- *Match with your crush if they like you back*
- *Website keeps you anonymous if they don't*

Case study: Stanford Link (2020)



- *Match with your crush if they like you back*
- *Website keeps you anonymous if they don't*
- ***What could go wrong?***

Case study: Stanford Link (2020)

The Stanford Daily

News • Campus Life

Vulnerability in 'Link' website may have exposed data on Stanford students' crushes

What's old is new again: Stanford Reveal (2023)

the stanford reveal

enter by midnight, 2/10



7688 crushes submitted

info input

if you and your crush list each other, we'll notify you both. if not, nothing happens.

♥ start ♥

```
44 | {  
45 |   "submittingUserFullName": "Aditya Saligrama",  
46 |   "user": "4yz2FPyYDgND8KhtVOQLCeeGsaq2",  
47 |   "submittingUserEmail": "akps@stanford.edu",  
48 |   "fullNames": []  
49 | },
```

```
231 | {  
232 |   "submittingUserEmail": "mccain@stanford.edu",  
233 |   "submittingUserFullName": "Robert Miles Redd McCain",  
234 |   "user": "N3Q9CkeKeJfKQz0hqt7qFbpanat1",  
235 |   "fullNames": [  
236 |     "isabelle levent"  
237 |   ]  
238 | },
```

The fastest web crash course ever

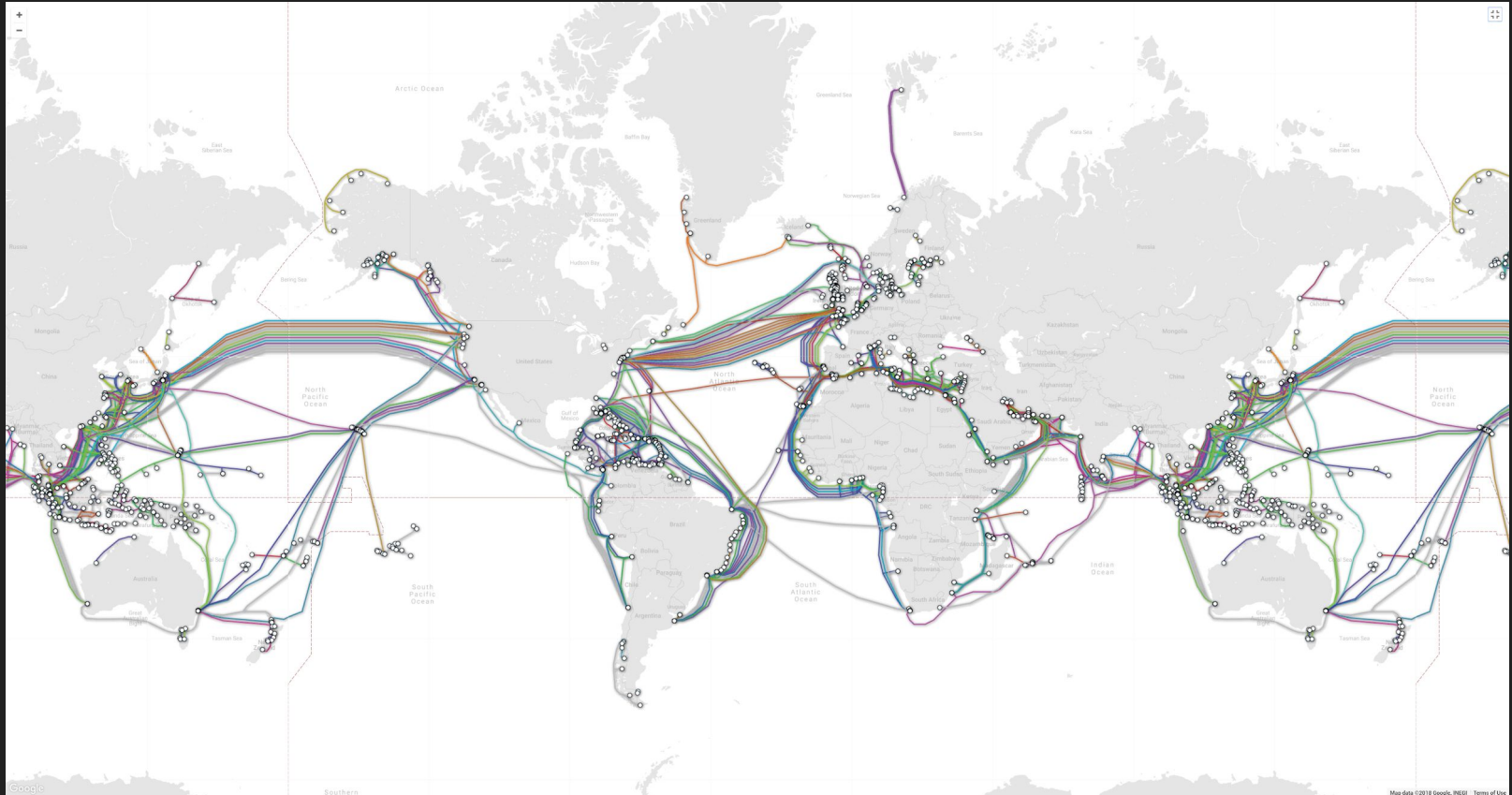
What language does the web speak?

```
<body>  
<div id="...">  
<h1>
```

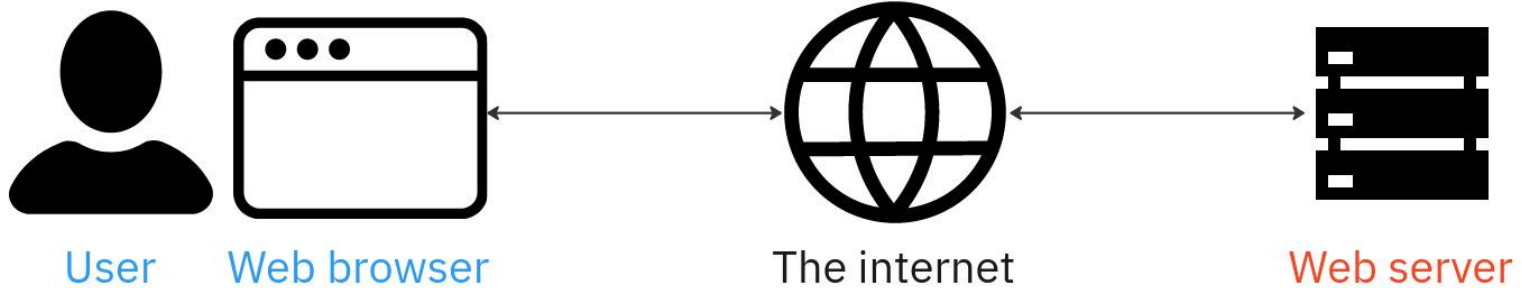
```
26 .screen-reader-text: hover,  
27 .screen-reader-text: active,  
28 .screen-reader-text: focus {  
29     background-color: #f1f1f1;
```

```
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
  
<link rel="stylesheet" href="http://localhost/css.css" type="text/css"/>  
<script type="text/javascript" src="http://localhost/javascript.js"></script>  
<script type="text/javascript">  
    (function(){  
        onLoaded: function(request) {  
            if (request.name == 'log_error') return;  
            log.trace("Ajax.Request: " + (request.name || request.url.substr(0, 30)) + "...");  
        },  
        onComplete: function(request) {  
            if (request.name == 'log_error') return;  
        },  
        onException: function(request, e) {  
            if (request.name == 'log_error') return;  
            log.fatal(request.url + ': ' + e.name + ' | ' + e.message + ' | ' +  
                e.stack);  
        }  
    })();  
    /bar */
```


How does the Internet work?



Our Internet Abstraction



How do we communicate with a web server?

HTTP

Hypertext Transport Protocol

HTTP: the missing language of the web



GET index.html



```
<!DOCTYPE html>  
<html>  
<body>  
  
<h1>Hello World!</h1>  
  
</body>  
</html>
```

HTTP protocol

GET / HTTP/1.0

Verb

Object (noun)

Protocol

HTTP Requests

GET / HTTP/1.1

Host: stanford.edu

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:59.0)

Gecko/20100101 Firefox/59.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Connection: close

Upgrade-Insecure-Requests: 1



HTTP Responses

HTTP/1.1 302 Found

← Response Code

Date: Mon, 02 Apr 2018 02:37:56 GMT

← Headers

Server: Apache

Location: <https://www.stanford.edu/>

Content-Length: 209

Connection: close

Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">

← Body

<html><head>

<title>302 Found</title>

</head><body>

<h1>Found</h1>

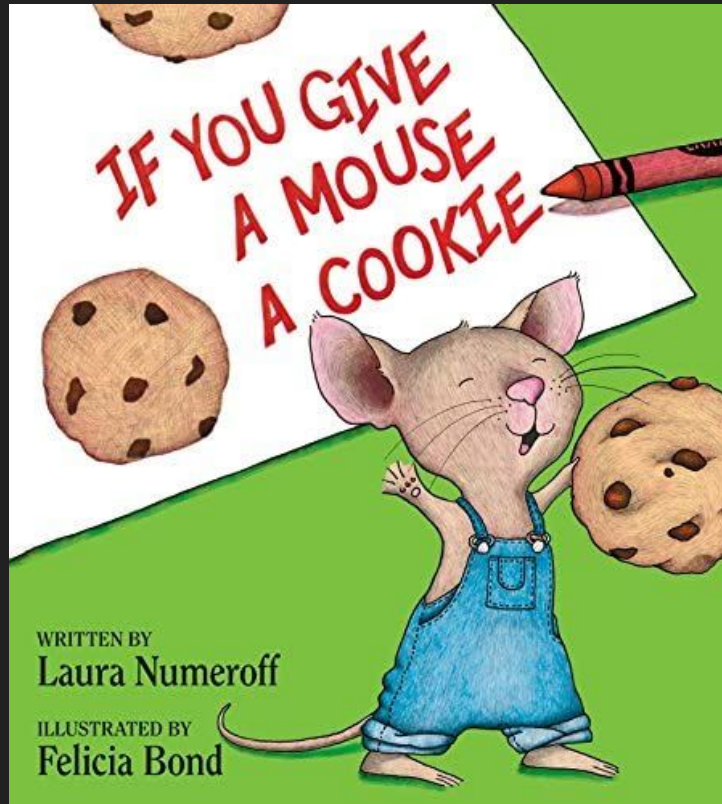
<p>The document has moved here.</p>

</body></html>

HTTP Requests: GET and POST

- **GET:** Requests a specified resource
 - Should **only retrieve data**, without changing server state
- **POST:** Submits data to the specified resource
 - Often causes **changes in state** or side effects on the server

Session Handling: *How does a website remember?*



- **Cookies!**
- Cookies enable web servers to store **stateful information** in your browser
- **Authentication cookies** are used to authenticate that a **user is logged in**, and with **which account**
 - On login: **Set-Cookie: session=session-id**
 - Future requests: **Cookie: session=session-id**

Common insecure design patterns

CatShare

<http://catshare.saligrama.io>

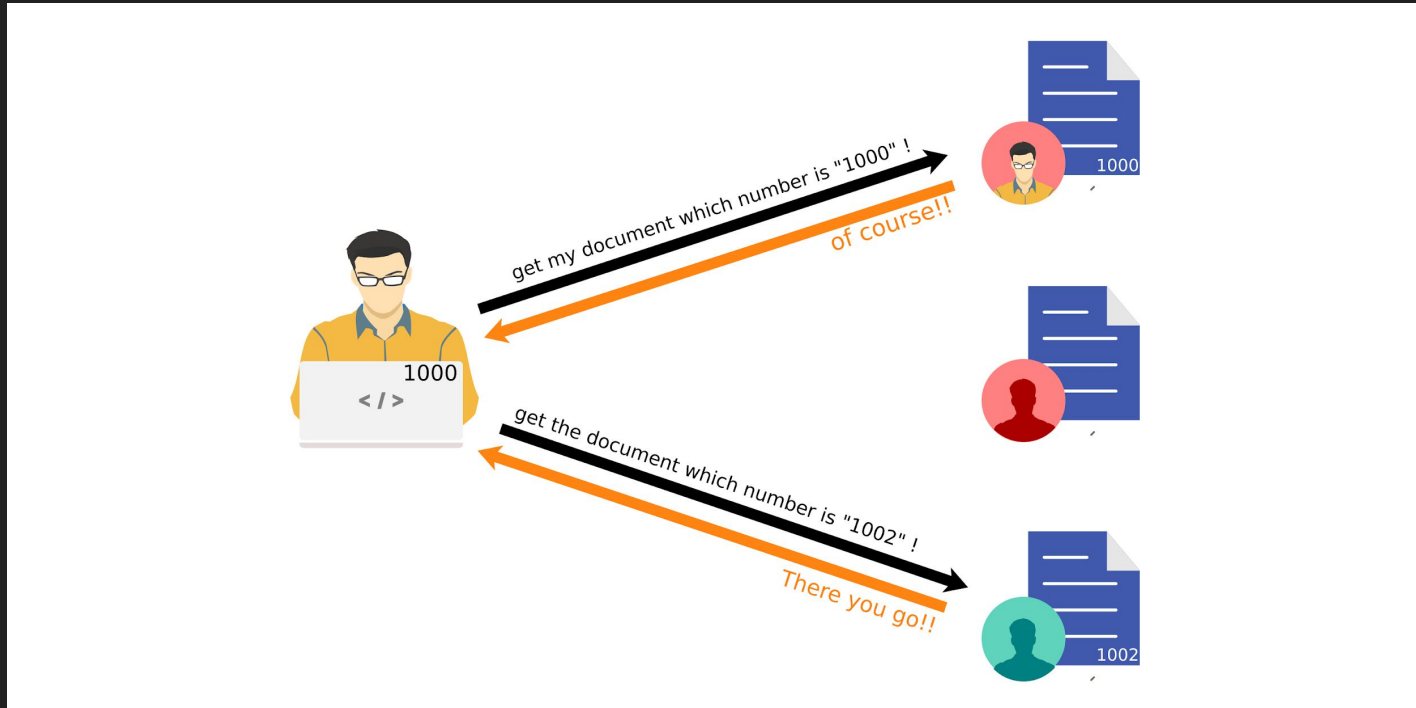


Vulnerabilities

- Insecure Direct Object Reference (**IDOR**)
- Cross Site Scripting (**XSS**)
- Improper Session Handling

Insecure Direct Object Reference (IDOR)

Or: asking the server for the resources you want



IDOR case study I: Parler

- IDOR vulnerability leads to leakage of 70TB of user data
- Why?
 - Poor engineering
 - Lack of testing



TRY IT!

- The CatShare team has a website <http://catshare.saligrama.io/> that **stores personal information**
- There's an endpoint <http://catshare.saligrama.io/user> to access this info
 - e.g. <http://catshare.saligrama.io/user?id=test>
- CatShare claims this is secure and only accessible to admins
- **Show us otherwise**

IDOR case study II: Stanford Marriage Pact (2020)

We told you we couldn't leave you empty handed tonight. Well, here's a gift from to thank you for your patience. A token of our gratitude, to let you know *just* how special you are.

👉 Check it out 👈

Gimme my 🔥 Hot Takes 🔥

Two more days until the end of Week 10—and one more day until the matches come out. When that happens, we want to help make sure as many people get matched as possible, so...

The questionnaire is open for another 7.2 hours, until 4pm PST later today. Text your friends, bug your enemies. They may not be *your* perfect match, but they could be someone else's. The bigger the pool, the better everyone's matches become.

Thanks again for your patience. We'll see you this evening for the match announcement.

Love,
The **Stanford Marriage Pact**

IDOR case study II: Stanford Marriage Pact (2020)

<https://mp.com/29d2223b196d87e8e9292308c074e593>

29d2223b196d87e8e9292308c074e593

MD5

saligrama@stanford.edu

mccain@stanford.edu

MD5

65af214d836bb936fd32c5c11f93c70d

<https://mp.com/65af214d836bb936fd32c5c11f93c70d>

Cross Site Scripting (XSS)

- XSS attacks enable attackers to hijack your website to **run JavaScript code** on other users' browsers
- They occur when **user input is not properly sanitized and displayed**, allowing it to execute as code

Cross-Site Scripting (XSS)



GET /myfeed



```
<!DOCTYPE html>
<html>
<body>

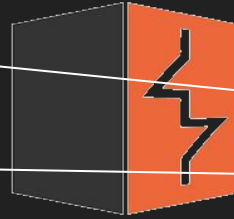
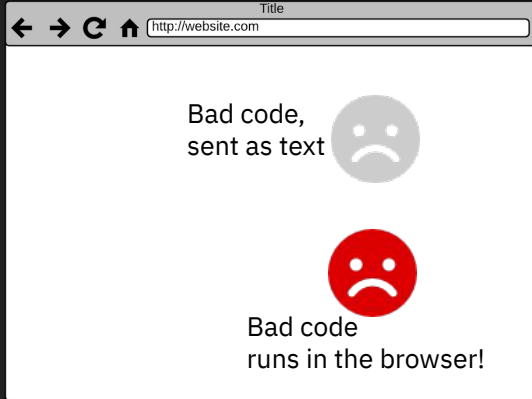
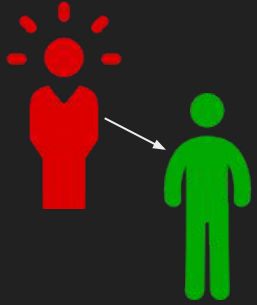
<b>Miles says:</b>
<script>
    alert("You got hacked!");
</script>

</body>
</html>
```

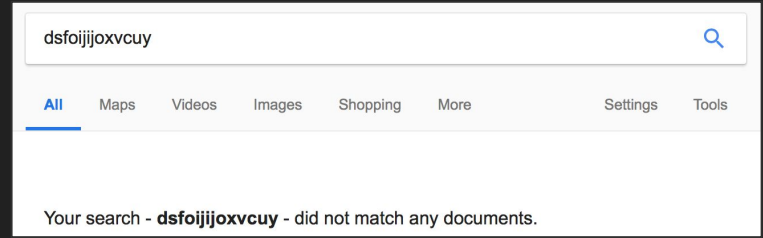
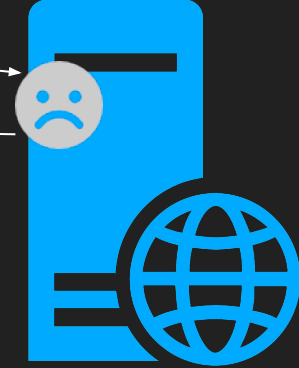


Reflected XSS

Hey, click this link

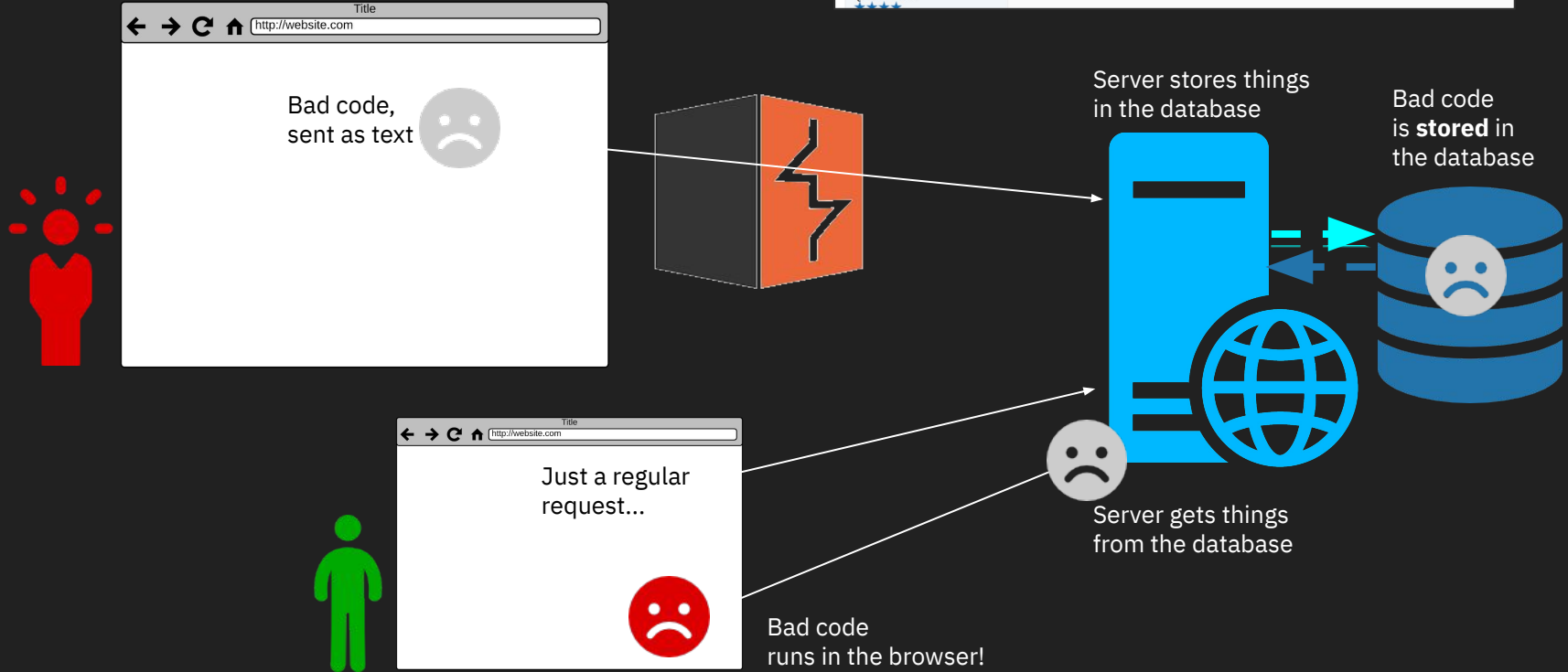
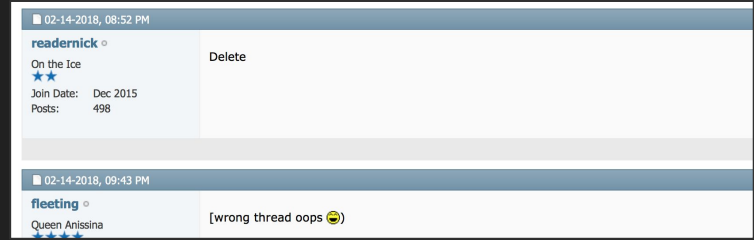


Bad code is **reflected**



[https://vulnerable.website/search?query=<script>alert\("pwned"\)</script>](https://vulnerable.website/search?query=<script>alert("pwned")</script>)

Stored XSS



www.yourwebsite.com/law/<?StealAllTheData.js>/supersecretdata

TRY IT!


- After our last data breach, we at CatShare want to make our customers feel like we care about them
- We added an endpoint <http://catshare.saligrama.io/hello> that **takes a user's name and greets them kindly**. Ya' know, to show we care
 - e.g. <http://catshare.saligrama.io/hello?name=User1>
- We think this is harmless and will only build customer trust. **Show us our mistake.**

XSS on Tweetdeck



*andy
@derGeruhn

 Follow

```
<script  
class="xss">$('.xss').parents().eq(1).find('a')  
.eq(1).click();$('[data-  
action=retweet]').click();alert('XSS in  
Tweetdeck')</script> 
```

 Reply  Retweet  Favorite  More

RETWEETS
39,868

FAVORITES
3,686



9:36 AM - 11 Jun 2014

Improper session handling

Cookie itself is insecure

- Can modify cookie to access another's account
 - e.g. become admin

Cookie not checked for authorization

- Use your own account to
 - Impersonate someone else
 - Escalate privileges to admin

Consequences are IDOR-like, **even when resource IDs are randomized**

Improper Session Handling

- CatShare added an **admin view** to <http://catshare.saligrama.io/login> for admins to **view user data**
- One set of credentials is **cooper:cooper**
- Can you become **admin** and view the user data?
- **Some handy tools:**
 - Your browser's Developer Tools (accessible from Inspect Element)
 - <https://kk.lol> (Base64 Encode/Decode)

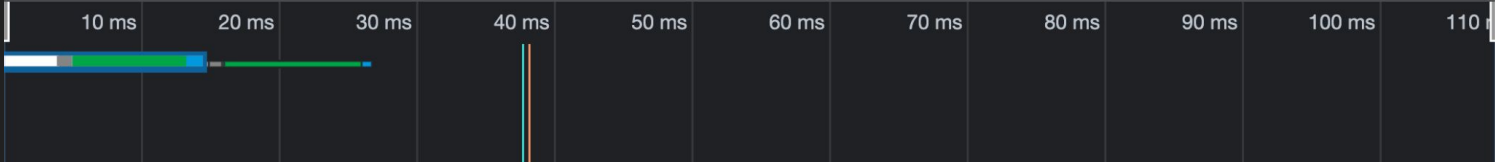
Elements Console Sources **Network** Performance Memory >> [1] [Settings] [Close]

● [No] [Filter] [Search] [Preserve log] [Disable cache] No throttling [Wi-Fi] [Up] [Down] [Settings]

Filter [Invert] [Hide data URLs]

All | Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other [Has blocked cookies]

[Blocked Requests] [3rd-party requests]



10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110 ms

Name × **Headers** Payload Preview Response Initiator Timing Cookies

- login
- auth

▼ **Response Headers** [View source](#)

- Connection:** keep-alive
- Content-Length:** 58
- Content-Type:** text/html; charset=utf-8
- Date:** Tue, 15 Nov 2022 23:56:29 GMT
- Keep-Alive:** timeout=5
- Location:** /auth
- Set-Cookie:** userId=Y29vcGVy; Max-Age=86; Path=/; Expires=Tue, 15 Nov 2022 23:57:56 GMT
- Vary:** Accept
- X-Powered-By:** Express

2 requests | 490 B tr ▼ **Request Headers** [View source](#)

You are cooper. Sorry, you do not have admin access to this endpoint. [logout](#)

The screenshot shows the Chrome DevTools Application tab. The left sidebar is expanded to 'Storage' > 'Cookies' > 'http://catshare'. The main pane displays a table of cookies. A large text overlay at the bottom of the table reads 'Select a cookie to preview its value'.

Name	Value	Domain	P..	E..	S..	HttpOnly	S	S	S	P	P
userId	Y29vcGVy	catshare.salig...	/	2...	14						M..
_ga_GJHJW3...	GS1.1.16...	.saligrama.io	/	2...	51						M..
_ga	GA1.1.94...	.saligrama.io	/	2...	29						M..

Select a cookie to preview its value

Session handling case study: Kontra (2022)



Session handling case study: Kontra (2022)

Verizon

4:36 PM

79%

Welcome back, **tweedledee**

Hot

New

This is a test poll submitted by
@tweedledee pretending to be
@tweedledum



@tweedledum

1 Reacts · 5m



Mitigating risk as a startup

Nothing is 100% secure

It happens to the best of us

Aditya's Blog

Thoughts, guides and fun from a security/systems enthusiast @ Stanford

Flipping the script: when a hacking class gets hacked

📅 October 12, 2022 📄 1316 words 🏷️ No tag

This morning, an [EternalBlue](#)-vulnerable machine used for testing for Stanford's [Hack Lab](#) course accidentally given a public IP address on Google Cloud was unsurprisingly pwned and used to launch further EternalBlue scanning against other public web hosts.

This blog post describes our course's infrastructure setup (including why we had that testing box in the first place), how we discovered and remediated the incident, and how we used the incident as a way to teach students about incident response and public disclosure.

Let the community help you

A **vulnerability disclosure policy** is intended to give ethical hackers **clear guidelines** for submitting potentially unknown and harmful **security vulnerabilities** to organizations.

Vulnerability Disclosure Policy Resources

DHS Template: <https://cyber.dhs.gov/bod/20-01/vdp-template/>

DoJ Framework:

<https://www.justice.gov/criminal-ccips/page/file/983996/download>

HackerOne:

<https://www.hackerone.com/blog/What-Vulnerability-Disclosure-Policy-and-Why-You-Need-One>

Example Safe Harbor: <https://github.com/cybertransparency/vdp-terms>

Please don't do this

November 22, 2021



Via E-Mail

Cooper Barry deNicola
Miles McCain
Aditya Saligrama

Re: Buzz Vulnerability Disclosure

To: Cooper de Nicola, Miles McCain and Aditya Saligrama

Hopkins & Carley represents The Buzz Media Corp. ("Buzz"). We write regarding your team of security researchers, both individually and collectively (referred to herein as the "Group") to make you aware of the Group's criminal and civil liability arising out of the Group's unauthorized access to Buzz's systems and databases.

Based on your own admissions in your email dated November 9, 2021 notifying Buzz of the security vulnerability, the Group explored "...the vulnerability..." and obtained unauthorized access to Buzz's "...complete databases..." and all information stored in Buzz's database. Your email further goes on to state that the Group edited user tables and created moderator and administrator accounts enabling the Group to access Buzz's systems without authorization.

The Group's actions in obtaining this unauthorized access to Buzz's databases violate the Computer Fraud and Abuse Act (18 U.S.C. § 1030) (CFAA), the Digital Millennium Copyright Act (DMCA) and Buzz's Terms of Use.

The Group circumvented Buzz's technological measures designed to protect Buzz's databases, without any permission or authority in violation of the DMCA. For these violations of the DMCA the Group may be liable for fines, damages and each individual of the Group may be imprisoned. Further, the Computer Fraud and Abuse Act (18 U.S.C. § 1030) (CFAA) imposes additional criminal and civil liability for unauthorized access to a protected computer, including accessing files or databases to which one is not authorized to access. The CFAA prohibits intentionally accessing a protected computer, without authorization or by exceeding authorized access, and obtaining information from a protected computer. Criminal penalties under the CFAA can be up to 20 years depending on circumstances.

Buzz's own Terms of Use expressly prohibits any of the following actions and clearly sets forth that the Group has no authorization to access Buzz's systems or databases "...attempt to reverse engineer any aspect of the Services or do anything that might circumvent measures employed to prevent or limit access to any area, content or code of the Services (except as otherwise expressly permitted by law); Use or attempt to use another's account without authorization from such user and Buzz; Use any automated means or interface not provided by Buzz to access the Services;..." Not only then are the Group's actions a violation of both the DMCA and the CFAA, as indicated above, the Group's actions are also a violation of Buzz's Terms of Use and constitute a breach of contract, entitling Buzz to compensatory damages and damages for lost revenue.

More Security at Stanford

Courses

- *INTLPOL 268*: Hack Lab
- *CS 152*: Trust and Safety Engineering
- *CS 155*: Computer and Network Security
- *CS 255*: Cryptography
- *CS 153*: Applied Security at Scale
- *CS 253*: Web Security
- *INTLPOL 268D*: Online Open Source Investigation

More security fun and hijinks: **Stanford Applied Cyber**

Opinion | Fizz previously compromised its users' privacy. It may do so again.



*Fizz had a large data vulnerability discovered last fall. Their response raises questions about the app today.
(Graphic: JOYCE CHEN/The Stanford Daily)*

Aditya's Blog

Thoughts, guides and fun from a security/systems enthusiast @ Stanford

A student's dream: hacking (then fixing) Gradescope's autograder

📅 February 28, 2023 📄 2591 words 🏷️ No tag

applied-cyber.stanford.edu



Check-Off Form

<https://tinyurl.com/106SecurityCheckoff>

Credits

Source Code for Vulnerable Web App

<https://github.com/cdenicola/CS106S-VulnerabilityExample>

(**sharecat** branch)

Other materials

- *Web Crash Course* – Alex Stamos, INTLPOL 268 Hack Lab
- *Web Crash Course, IDOR/XSS/Session Handling Slides, Marriage Pact IDOR Case Study* – Cooper de Nicola
- *Stanford Link, Fizz articles* – The Stanford Daily
- *CatShare* – Cooper de Nicola, Aditya Saligrama, George Hosono