CS 106S Week 5

Ethical Hacking & Web Security

Ben Yan, Spring 2025

Based on materials by Stanford Applied Cyber, credit to Aditya Saligrama, Cooper de Nicola

cs106s.stanford.edu

Stanford | ENGINEERING
Computer Science

Welcome back! Great to see you :)

It’s Week 5 wow

Spring **Summer**

So … how’s life y’all
Hope that midterms are going well!

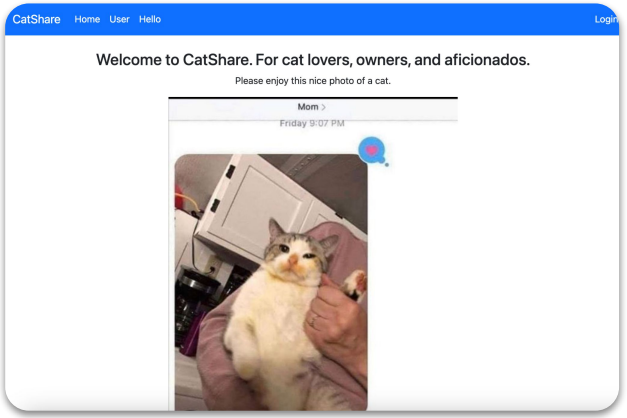
Agenda for Today

1. Brief overview of cybersecurity, relevant & modern case-studies

2. Crash course of the web & web hacking, typical insecure designs

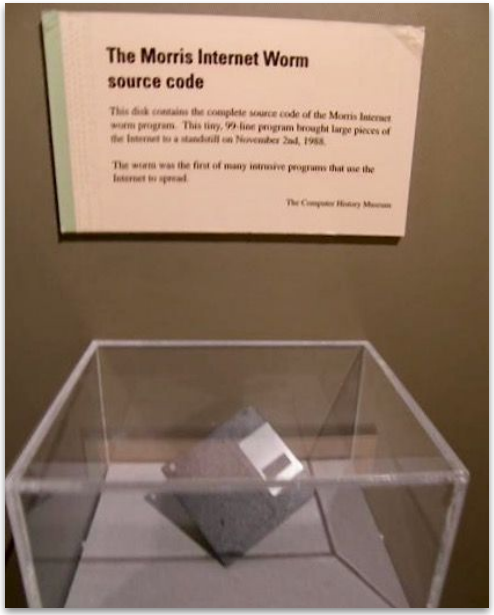
3. **project:** hacking a public cat photo-sharing website

4. Reflections & check-off form!



💾 1998: Morris Worm

- **First computer worm on the Internet**
- Nov. 1988 – A Cornell grad student (Robert Morris) released a worm from MIT, exploiting computers with certain UNIX and network security flaws.
- The worm could replicate itself repeatedly, leading to some computers crashing over and over again.
- **Est. 10% of all machines at the time were infected.**
- Morris was caught and convicted, in 1989, under the brand new **Computer Fraud and Abuse Act.**
- He then went on to become a tenured professor in EECS at MIT ✨, with research on computer networks.

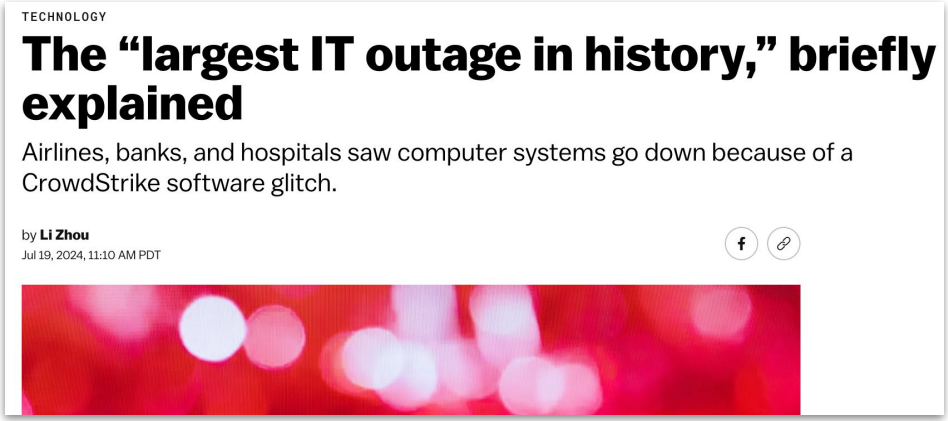


for ethical reasons,
yes, this is a joke

2024: Crowdstrike Outage

Photo I took while visiting SF over last summer

- **~8.5 million systems crashed** (endless blue screens of death)
- Though the bug was found within hours, many computers had to be fixed manually → leading to persisting outages



TECHNOLOGY

The "largest IT outage in history," briefly explained

Airlines, banks, and hospitals saw computer systems go down because of a CrowdStrike software glitch.

by **Li Zhou**
Jul 19, 2024, 11:10 AM PDT

https://www.vox.com/technology/361740/crowdstrike-outage-windows

Why care about security?

Basically everything out there is secure by now – right?

Stanford Link (2020)

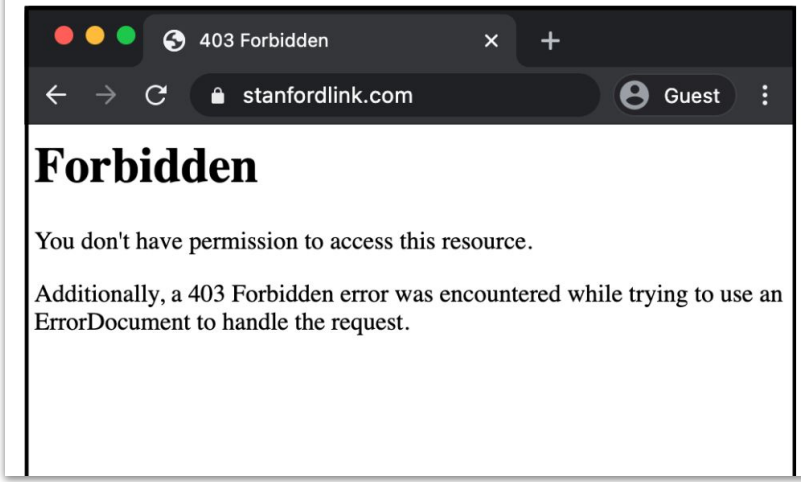
News • Campus Life

New Stanford Link website connects students with mutual crushes

Stanford Link's logo. (Photo: stanfordlink.com)

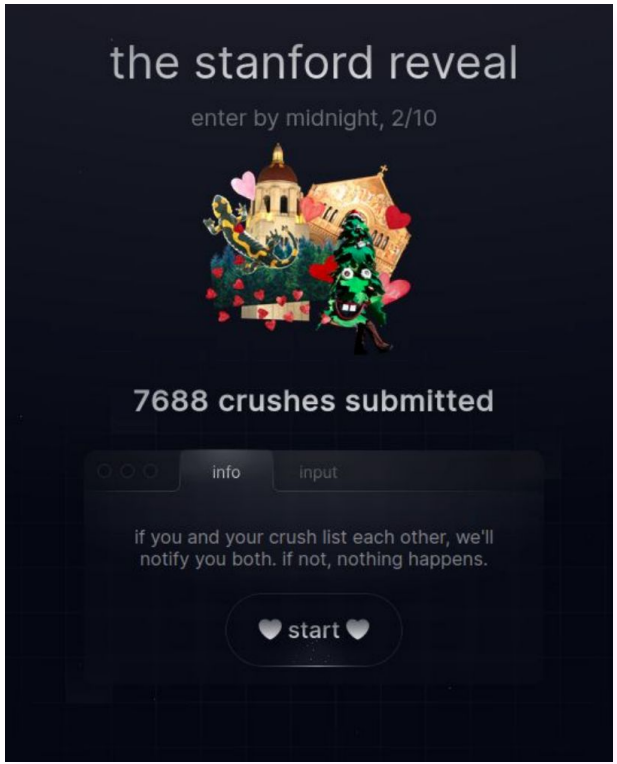
News • Campus Life

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



- **Match with your crush if they like you back**; otherwise, you stay anonymous
- > 1700 students had signed up
- **What could possibly go wrong?**

💘 Stanford Reveal (2023)



Humor

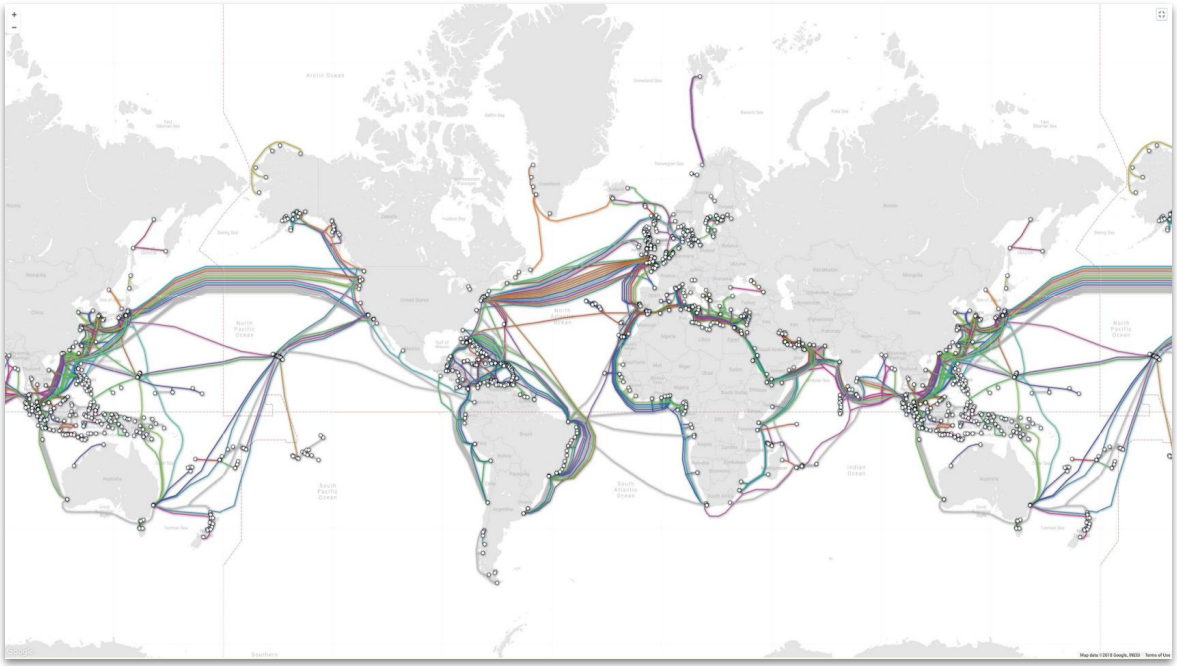
Stanford Reveal pledges to leak only the "juiciest" crushes

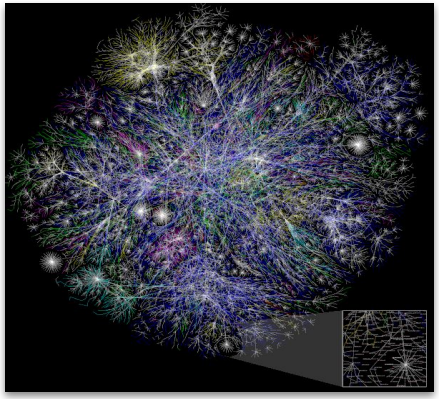
```
{"submittingUserEmail": "bbyan@stanford.edu"
  "submittingUserFullName": "Benjamin Yan",
  "user": "YmVuamFtaW4geWFu",
  "fullNames": [Gong Yoo (plays Recruiter from
    🦑 Squid Game)] ← a hypothetical example!
}
```

Why should we hack?

The Internet

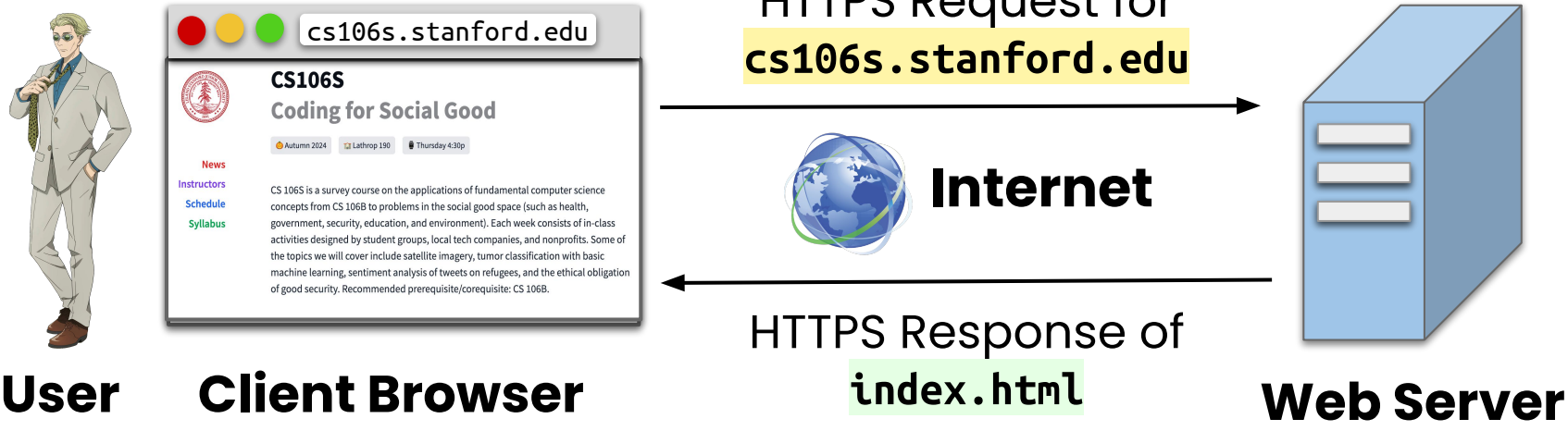
~~all of this just so I can watch reels 2 hours every morning~~





Internet: Client-Server Model

A conventional abstraction



The **client** (who initiates requests for resources) and **server** communicate over a computer network, e.g., the Internet

HTTP: Hypertext Transfer Protocol

Request-response communication with web server

Example: **Getting** fun facts about the number **30** from `www.numbersapi.com`

Method (Verb)	Path (Object)	Protocol Version (Adverb)
`GET`	`/30`	`HTTP/1.1`

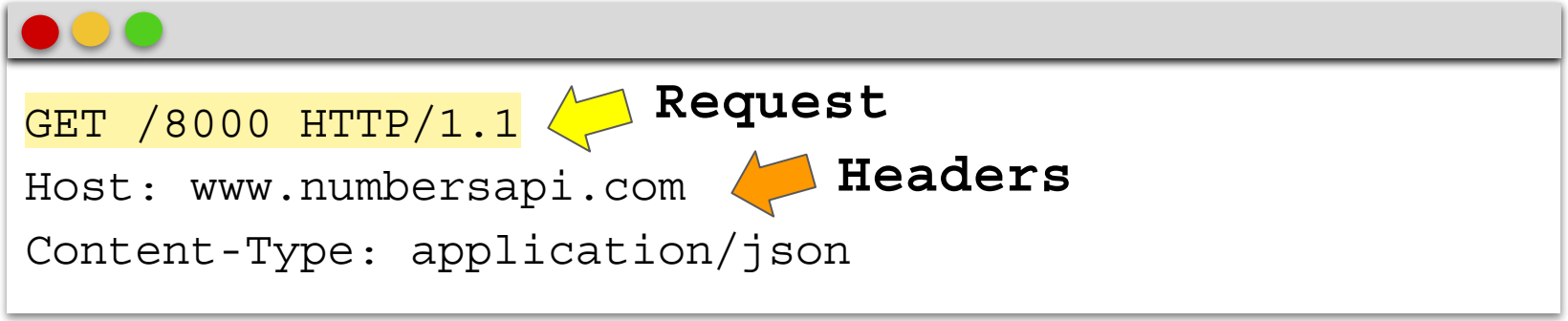
Headers (Modifiers)

```
Host: www.numbersapi.com
Content-Type: application/json
```

Demo: Sending HTTP Requests in the Terminal via `telnet`

🌐 HTTP: Request → Response

🤔 Say we want some fun facts about the number **8000…**



HTTP: Request → Response

We got a **fun fact** back I guess

```
HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Tue, 04 Feb 2025 14:54:59 GMT
…
{"text": "8000 is the approximate number
          of mirror squares the biggest disco ball
          in the world had in 2006.",
 "number": 8000,
 "found": true,
 "type": "trivia"
}
```

Response Code

Headers

Body/Payload

HTTP Requests: GET vs POST

- A **GET** request **only extracts** data from a server.
- A **POST** request **modifies or updates resources** on the server side, like making changes within a database.

HTTP Requests: GET vs POST

- A **GET** request **only extracts** data from a server.
- A **POST** request **modifies or updates resources** on the server side, like making changes within a database.

Client (JavaScript)

Web Server / Database

GET
`/api/images`


```
['img1.jpg',
 'img2.jpg',
 ...]
```

E.g., a GET request may fetch one's Instagram feed from an internal database, with the payload (**server**→ **client**) being a collection of images.

HTTP Requests: GET vs POST

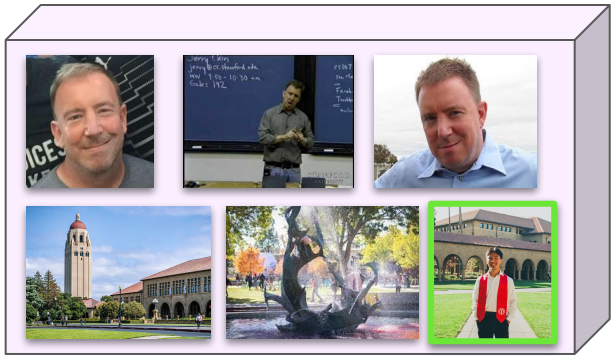
- A **GET** request **only extracts** data from a server.
- A **POST** request **modifies or updates resources** on the server side, like making changes within a database.

Client (JavaScript)

Web Server / Database

POST
`/api/images`

`new_img.jpg`

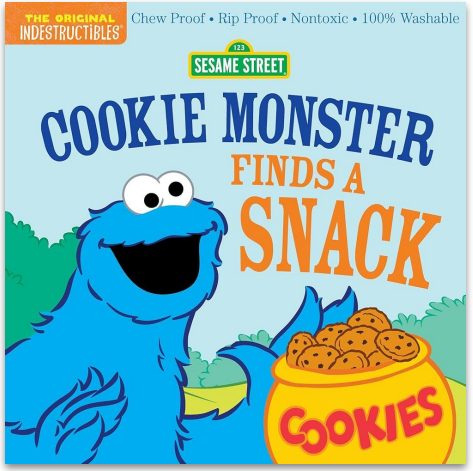


E.g., a **POST request may add a newly posted image** to the Instagram database, with the payload (**client**→ **server**) being the new image.

Session Handling

How does a website remember user sessions / logins?

- **Cookies!** 🍪🍪🥠
- Cookies allow websites to store **stateful information** (e.g., classes being added in SimpleEnroll), or to track browsing activity.
- **Authentication Cookies:** Used to authenticate that a user is currently logged in & **with which account**

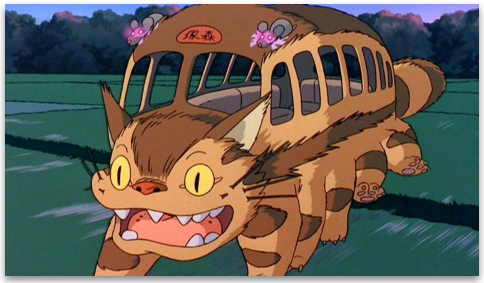


website = cookie monster

When logging in: `Set-Cookie: session=session_id`

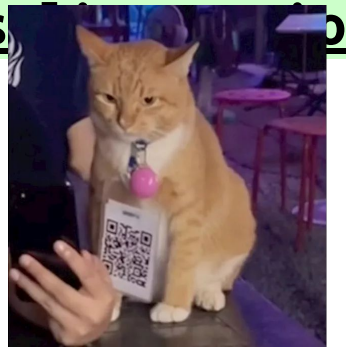
After first login: `Cookie: session=session_id` 🍪

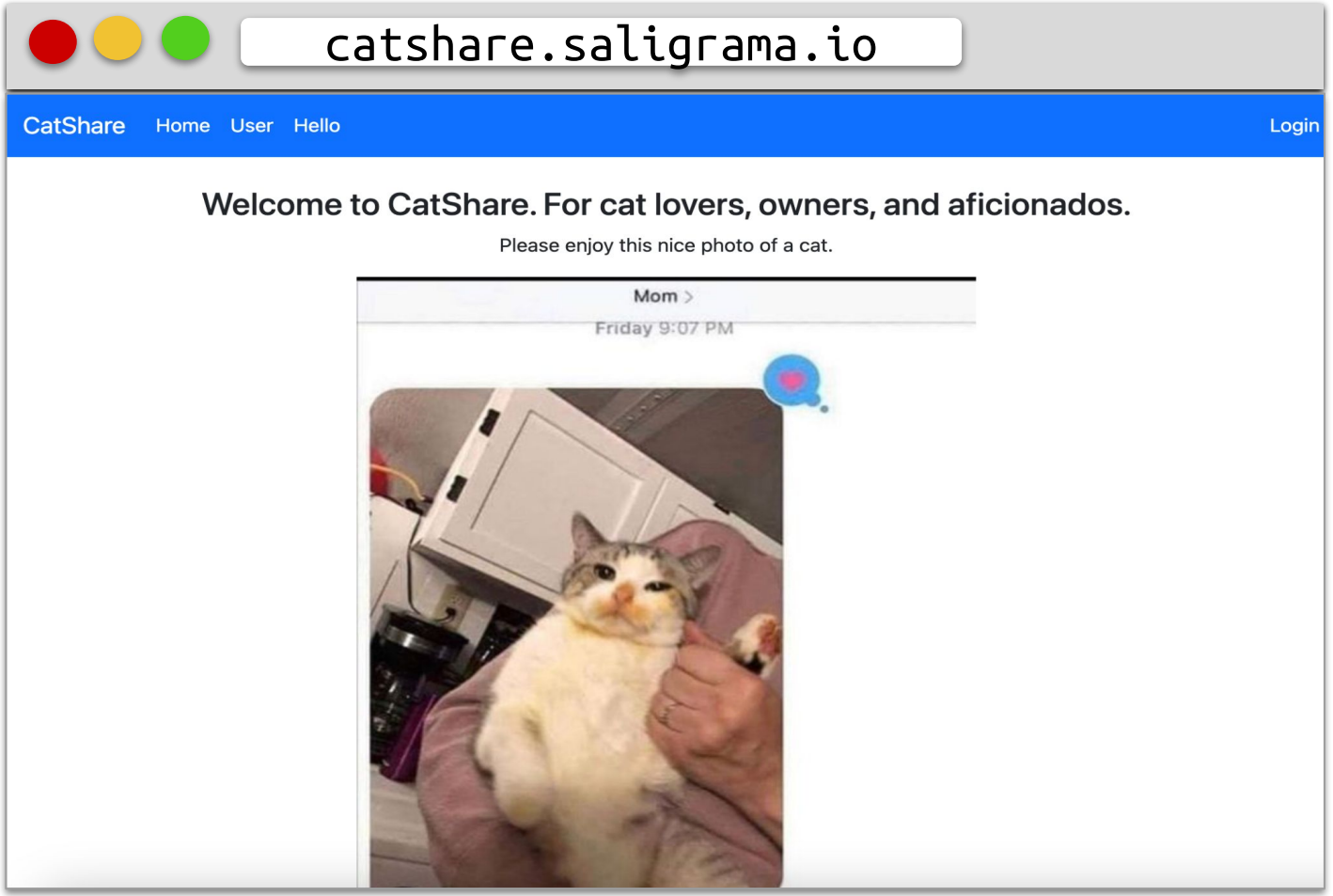



🐈 Catshare Website

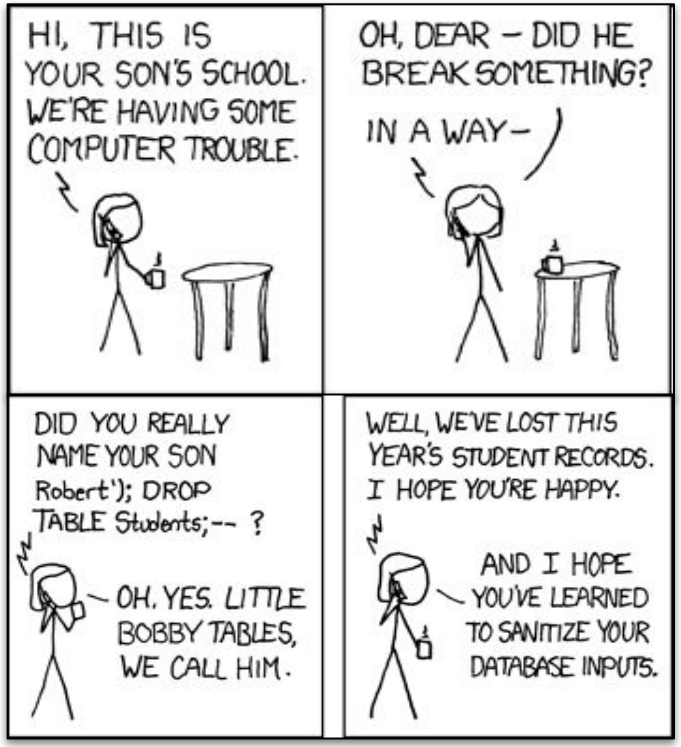




catshare.saligrama.io
CatShare
Home
User
Hello
Login
Welcome to CatShare. For cat lovers, owners, and aficionados.
Please enjoy this nice photo of a cat.
Mom >
Friday 9:07 PM

Common Website Vulnerabilities

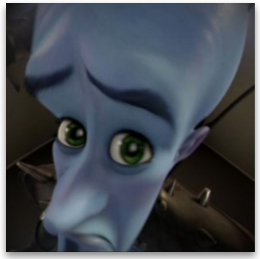
1. Insecure Direct Object Reference (IDOR)
2. Cross Site Scripting (XSS)
3. Improper Session Handling
4. Database vulnerabilities, e.g., SQL injection



IDOR: Indirect Direct Object Reference

Requesting resources directly from the **server**

give me the meme with **ID 100** plz!



ofc, here's `meme-100.jpg!`

give me the meme with **ID 200** plz!

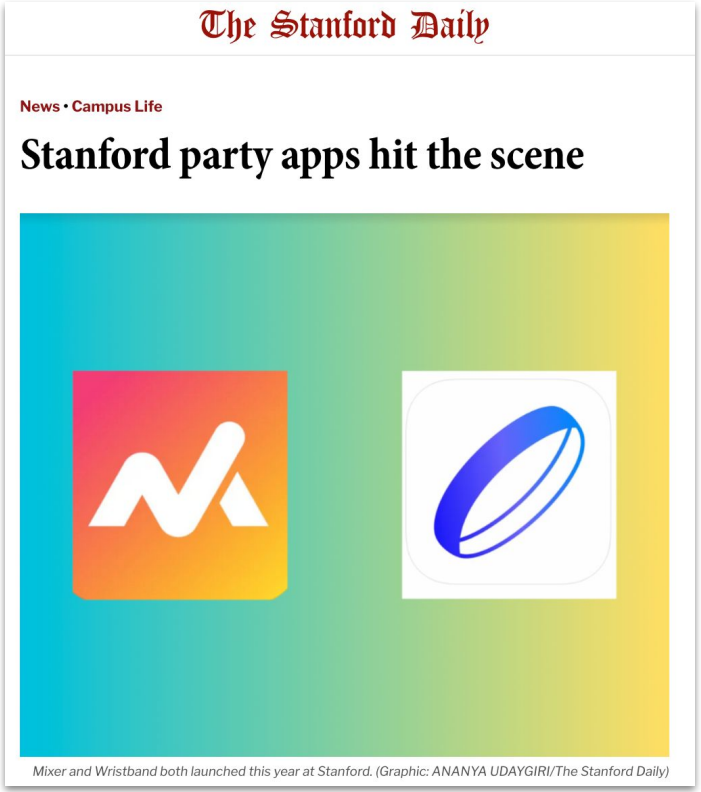
ofc, here's `meme-200.jpg!`

give me the meme with **ID 3.14159** plz!

sorry, **can't find resource!**

⚠️ **HTTP 404 Error**

IDOR: Wristband (2023)



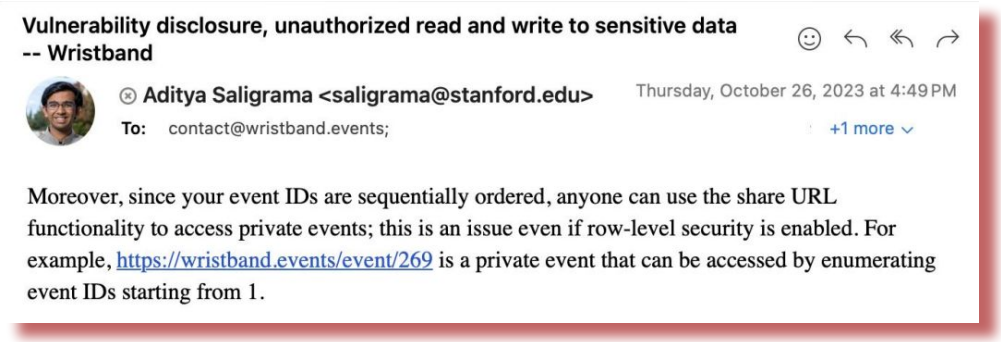
The Stanford Daily

News • Campus Life

Stanford party apps hit the scene

Mixer and Wristband both launched this year at Stanford. (Graphic: ANANYA UDAYGIRI/The Stanford Daily)

Wristband: a startup app for discovering and getting into public & private events on campus 🎉🌲



Vulnerability disclosure, unauthorized read and write to sensitive data -- Wristband

Aditya Saligrama <saligrama@stanford.edu> Thursday, October 26, 2023 at 4:49 PM

To: contact@wristband.events; +1 more

Moreover, since your event IDs are sequentially ordered, anyone can use the share URL functionality to access private events; this is an issue even if row-level security is enabled. For example, https://wristband.events/event/269 is a private event that can be accessed by enumerating event IDs starting from 1.

IDOR: Wristband (2023)

- Each event (public or private) was universally accessible via a **link that contained its event ID / number**.

`https://wristband.events/event/269`

- These ID numbers were **sequentially ordered**, meaning that by counting up from 1, one could stumble upon and access private events.

`https://wristband.events/event/1`

`https://wristband.events/event/2`

• • •

`https://wristband.events/event/40`

• • •

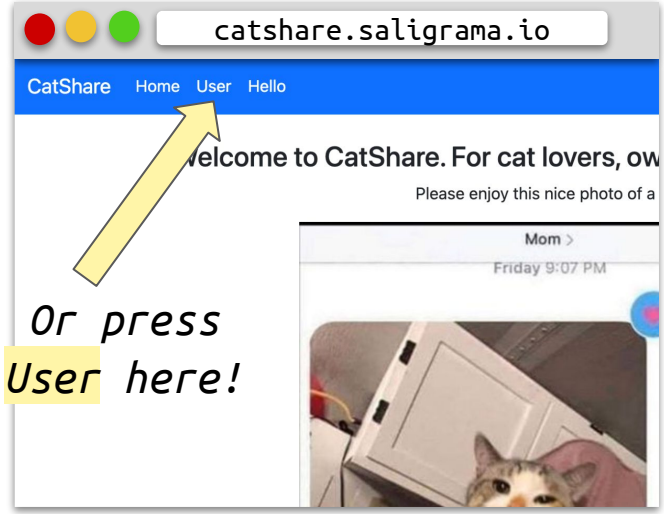
🎭**ACCESS TO SECRET EVENT:**
E.g., sign up for Jerry's Fleet Street reunion party!

Hack Catshare! Round 1

- The Catshare startup has a website (https://catshare.saligrama.io/) that stores personal information!
- There's a **new endpoint** to access this info:

https://catshare.saligrama.io/**user**

An example use case:
https://catshare.saligrama.io/**user?id=0**



- The Catshare team claims that this endpoint is secure and only accessible to admins. **Prove them wrong.**

IDOR: Marriage Pact (2020)

Sample Question: I believe in star signs ♐

1	2	3	4	5	6	7

Strongly disagree — Neutral — Strongly agree

- Based on answering **compatibility questions like the ones above**, get matched with someone else on campus!

Dear Benjamin,

Matches are here! Before you slide into their DMs, a few things are worth noting:

1) Remember, this is an algorithm

An algorithm is not the hand of god. All sorts of funky things can happen with algorithms. You could get someone you know. You could get someone you don't. You could get someone you wish you didn't.

The point is, it's entirely possible to get an ex-flame, current RA, or sibling (rip, lmfao). Remember: We're talking backup plans here. Sometimes the universe has a real sense of humor.

💍 IDOR: Marriage Pact (2020)

- Similar to Wristband, each Marriage Pact questionnaire had **a link with a unique (but not un-discoverable) identifier**

Ben Yan `https://mp.com/de6067feba693ee691b94b25d0527b30`

bbyan@stanford.edu → **MD5 Encoding Hash** → **de6067feba693ee691b94b25d0527b30**

Cooper de Nicola `https://mp.com/554d417a3bc9fbcba653c0097c6f3710`

cdenicol@stanford.edu → **MD5 Encoding Hash** → **554d417a3bc9fbcba653c0097c6f3710**

Avoiding IDOR Attacks

- Ensure that a user is **allowed to access a resource** before returning it
- When this isn't possible (e.g., cloud storage buckets), **make resource URIs random and unpredictable.**

- **That is, avoid:**

❌ Automatically incrementing resource IDs (e.g., Wristband)

❌ Hashing a **guessable property** like name, phone number, username, email (e.g., Marriage Pact)

Use **random identifiers** such as globally unique 128-bit UUIDs

Cross-Site Scripting (XSS) Attacks

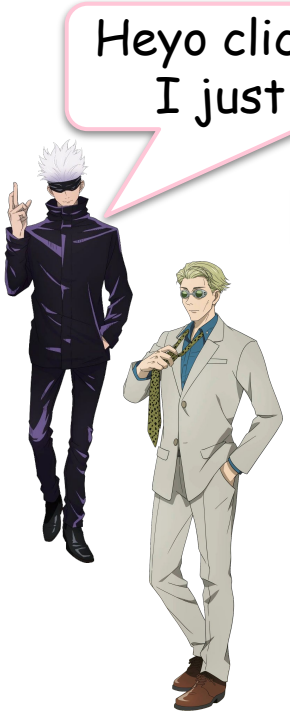
- XSS stands for **Cross Site Scripting**
- XSS is a potentially dangerous attack that enables hackers to take over your website to **run JavaScript code** on other users' browsers

```
https://innocent.website/myfeed?id=<script>alert("you got hacked!")</script>
```

- This typically occurs when **user input is not properly sanitized and displayed**, allowing it to execute as code

```
https://innocent.website/myfeed?id=<?steal-all-the-money.js>
```

Cross-Site Scripting (XSS)



https://website.com

Your Explore Page:

You got hacked!

GET /myfeed

```
<html>
<body>

<b>Your Explore Page:</b>
<script>
    alert("you got hacked!")
</script>

</body>
</html>
```

Reflected XSS

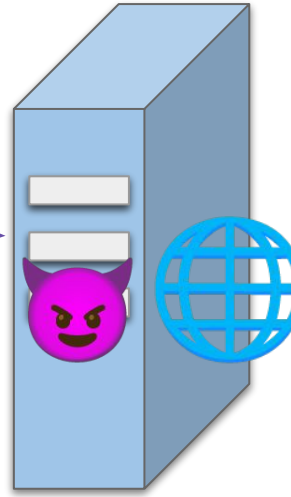
Heyo click on this reel
I just sent you :)

https://website.com

Bad code sent as text

Bad code runs in the browser

The **malicious code** gets **reflected by the server** back to the user.



https://buggy.website/search?q=<script>alert("get rekt!")</script>

Stored XSS

Imma troll the database

website.com

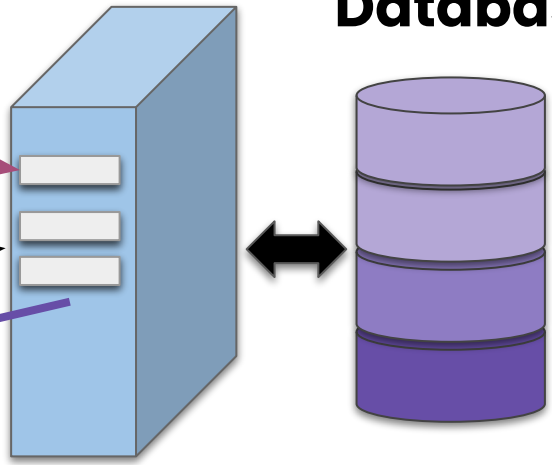
Bad code sent as text

The **malicious code** gets sent to the server.

The server **stores the malicious code in the database.**

Server

Database



website.com

Just a regular request…

Bad code runs in browser

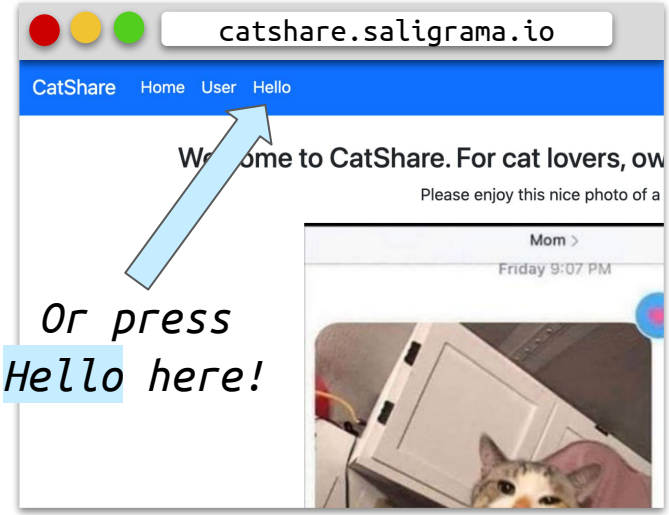
Server fetches resources from the database, and it'll **send the malicious code to the user**

Hack Catshare! Round 2

- After the previous data breach, **Catshare's valuation fell by 400 meows**. Not good.
- As recompense, the Catshare startup wants to make its customers feel welcome again. They've added a **new endpoint** that takes a user's name and greets them :)

`catshare.saligrama.io/`**`hello`**

An example use case:
`catshare.saligrama.io/`**`hello?name=ben`**



- The Catshare team claims that this endpoint is secure and only accessible to admins. **Prove them wrong.**

JavaScript for Modifying Webpage

- JavaScript is a **powerful 'interface' with an HTML webpage,** enabling HTML elements to be programmatically accessed and modified.

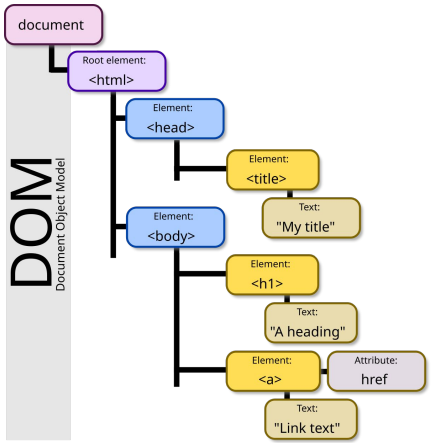
Add text on the page | Load new images | Toggle display to night mode

Append, remove, or modify any HTML nodes i.e. tags/elements

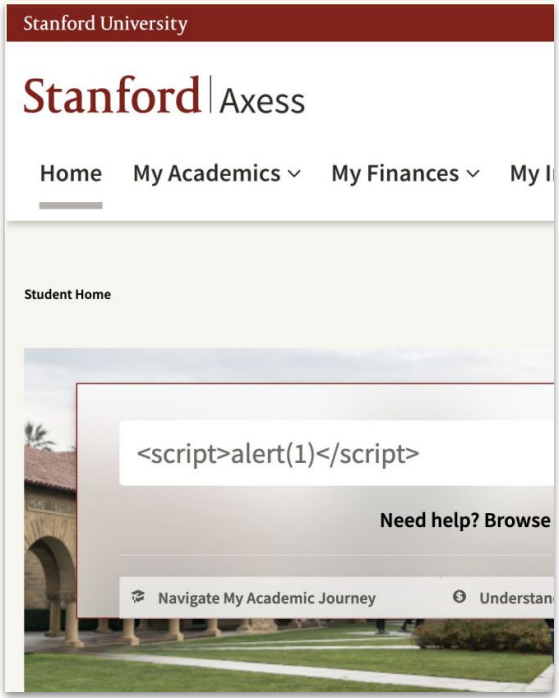
```
document.body.innerHTML = "text here";
```

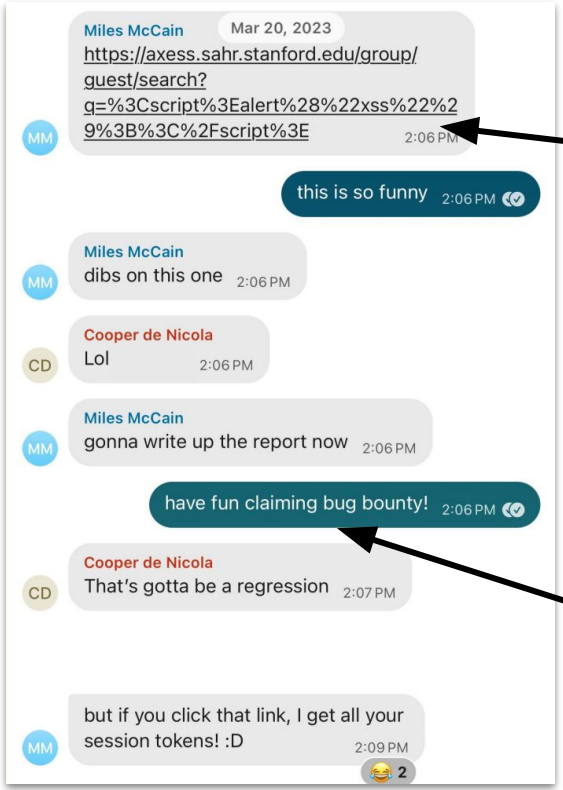
Modify the internal HTML content / text on the page

```
let text = document.createTextNode("text here");
document.body.appendChild(text);
```



XSS: Stanford Axess (2023)





Miles & company found and disclosed an XSS vulnerability in Axess (March 2023)

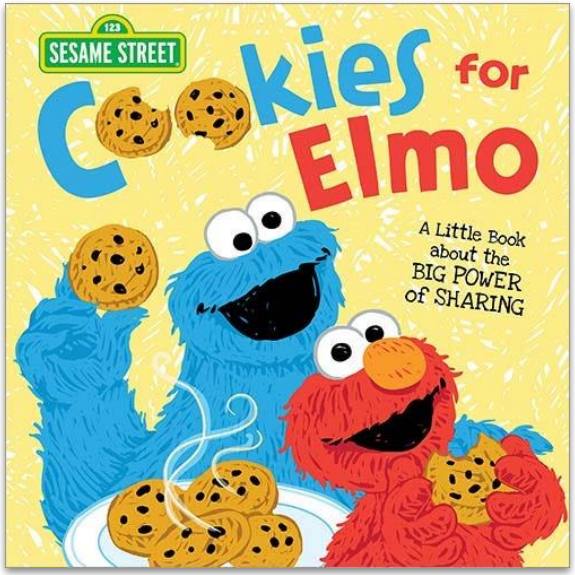
Awarded **$1000** by the **Stanford Bug Bounty**

Remediated Jan. 2024

Cookies: 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
  - Ascend privileges to admin

🧑‍💻 Hack Catshare! Final Round

- Seeking to rectify the user data privacy issue, Catshare has built an **admin-only view** to look at user data, at the new endpoint below.

`https://catshare.saligrama.io/login`

- First, log in using:
  - *Username:* **`stanford`**
  - *Password:* **`stanford`**

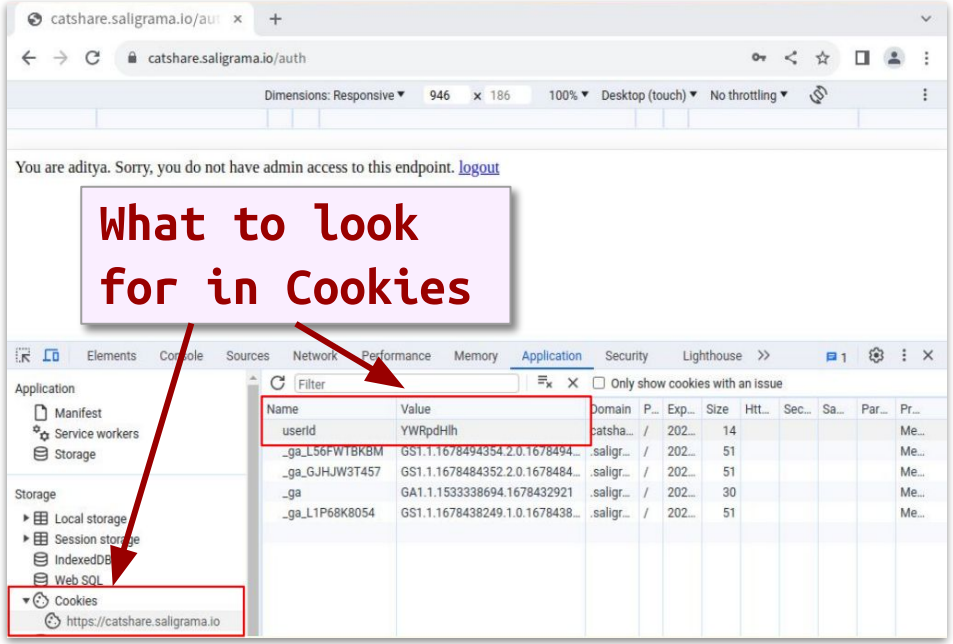
- Can you become admin and view the user data? 😱

🛠️ Tools & Reference Guide

- To access cookies, go to browser's Developer Tools (**Inspect Element** / Console → **Application** tab → then **Cookies** under Storage)

- Cookies are in **Base64** format
  - Transforms data into a mix of letters and numbers
  - Doesn't actually encrypt or secure the data, just a **different way to present it**

- Use **https://kk.lol** to encode and decode from Base 64!



https://catshare.saligrama.io/**login**

Try **stanford:stanford** first

Avoiding Improper Session Handling

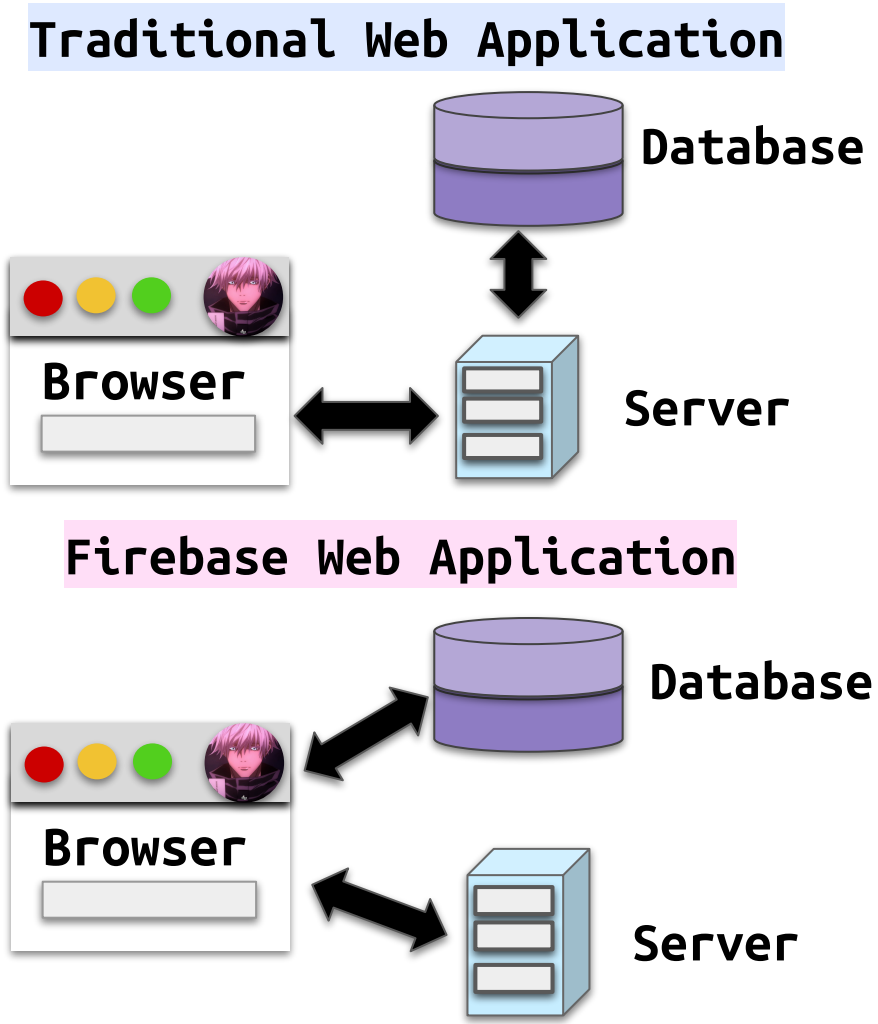
Before taking a sensitive action:

- ✅ Check to make sure the user is who they say they are
- ✅ And that they are allowed to perform the action

Database Vulnerabilities

With misconfigured security rules, clients can **directly access the database – including potentially malicious clients!**

- ★ Database is in charge of validating user access to data
- ★ Poor validation (e.g., misconfigured rules) → **unauthorized data access**



Firebase Vulnerabilities: Fizz (2021)

Opinions

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)

Opinion by Joyce Chen
Nov. 1, 2022, 10:00 p.m.

Firebase Vulnerabilities: Fizz (2021)

"At the time, Fizz used Google's Firestore database product to store data including user information and posts. Firestore can be configured to use a set of security rules in order to prevent users from accessing data they should not have access to. However, **Fizz did not have the necessary security rules set up, making it possible for anyone to query the database directly and access a significant amount of sensitive user data.**

We found that **phone numbers and/or email addresses for all users were fully accessible**, and that posts and upvotes were directly linkable to this identifiable information. **It was possible to identify the author of any post on the platform.**

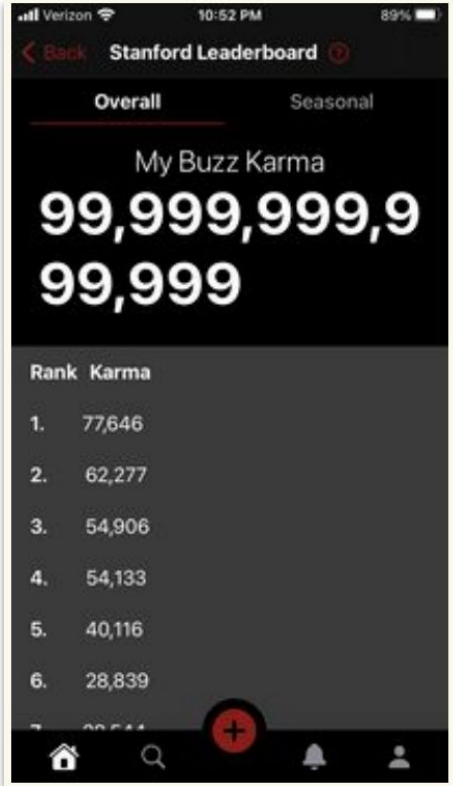
Moreover, the database was entirely editable — **it was possible for anyone to edit posts, karma values, moderator status**, and so on. Having moderator status granted access to a dashboard that provided the ability to delete arbitrary posts."

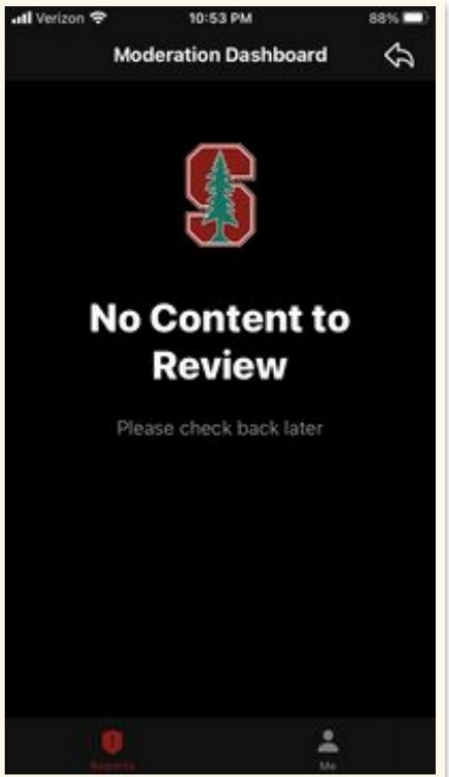
Misconfigured security rules in Fizz's Firebase, allowing anyone to **query the database directly**

Phone numbers / emails were accessible, along with **author of any post**

Could also **edit database**, e.g., posts, karma values, moderator status

Firebase Vulnerabilities: Fizz (2021)





Potential Legal Consequences to Hacking

November 22, 2021 [redacted]

Via E-Mail

Cooper Barry deNicola [redacted]
Miles McCain [redacted]
Aditya Saligrama [redacted]

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.

When your classmates threaten you with felony charges
Aug 28, 2023

Portfolio
Posts
Letter

A few weeks ago, I was part of a talk at DEF CON 31 called The Hackers, The Lawyers, and the Defense Fund. I was asked to share my experience receiving a legal threat for good-faith security research from my classmates.

This story has been told before (e.g., by my friend Aditya who was also involved and by the Stanford Daily), but I wanted to share my talk here for posterity.

The following is an approximate transcript. (If the language feels terse, that's why.) I've added a few links and cleaned up some of the language for clarity.

https://miles.land/posts/classmates-legal-threat-fizz-defcon/

Nothing is 100% secure

But that isn't a reason not to build!

Vulnerabilities happen to the best

saligrama.io/blog/hack-lab-got-hacked

Aditya Saligrama

Portfolio Blog Notes Photography Resume

Flipping the script: when a hacking class gets hacked

October 12, 2022
1351 words

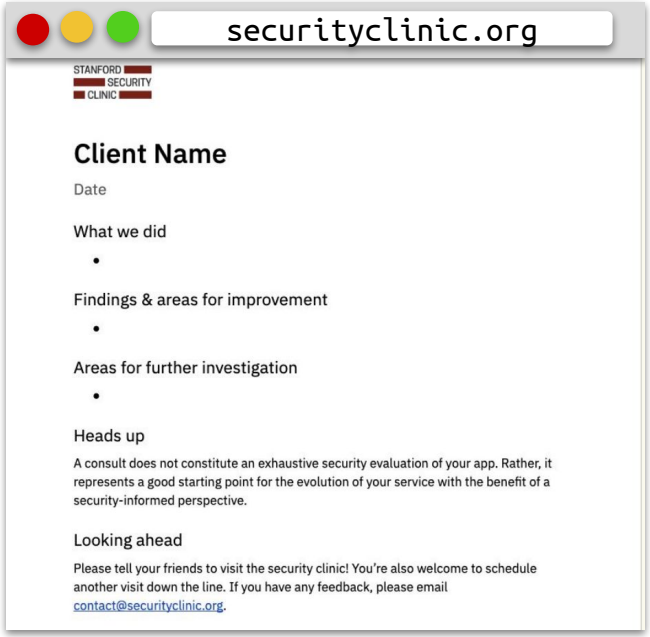
This morning, an EternalBlue-vulnerable machine used for testing for Stanford's Hack Lab course accidentally given a public IP address on Google Cloud was unsurprisingly pwned and used to launch further EternalBlue scanning against other public web hosts.

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.

Stanford Online - Stanford University

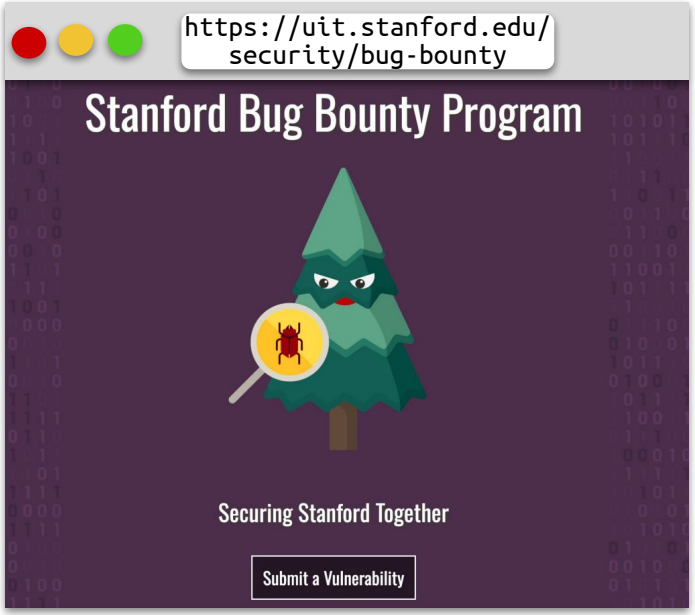
Hack Lab Course | Stanford Online

The community can help!



Disclosing vulnerabilities ethically!

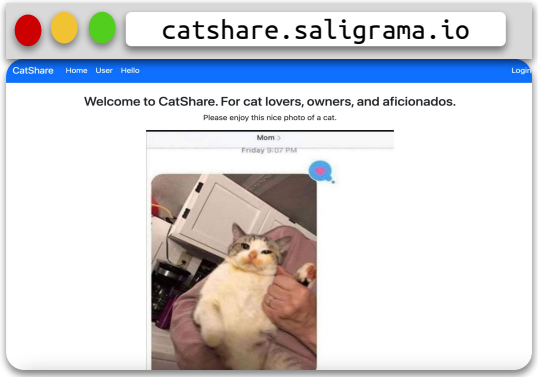
http://securityclinic.org



Bug Bounty programs incentivize responsible disclosures of security vulnerability – often with a monetary reward 💰💰💰💰💰

Further Resources

- **CATSHARE** (Aditya Saligrama, Cooper de Nicola, George Hosono) & accompanying source code:
  - https://github.com/saligrama/catshare-serverless



- **Security / cybersecurity courses** at Stanford
  - INTLPOL 268: Hack Lab
  - CS 155: Computer & Network Security
  - CS 152: Trust & Safety Engineering
  - CS 255: Cryptography
  - CS 40: Cloud Infrastructure & App Deployment

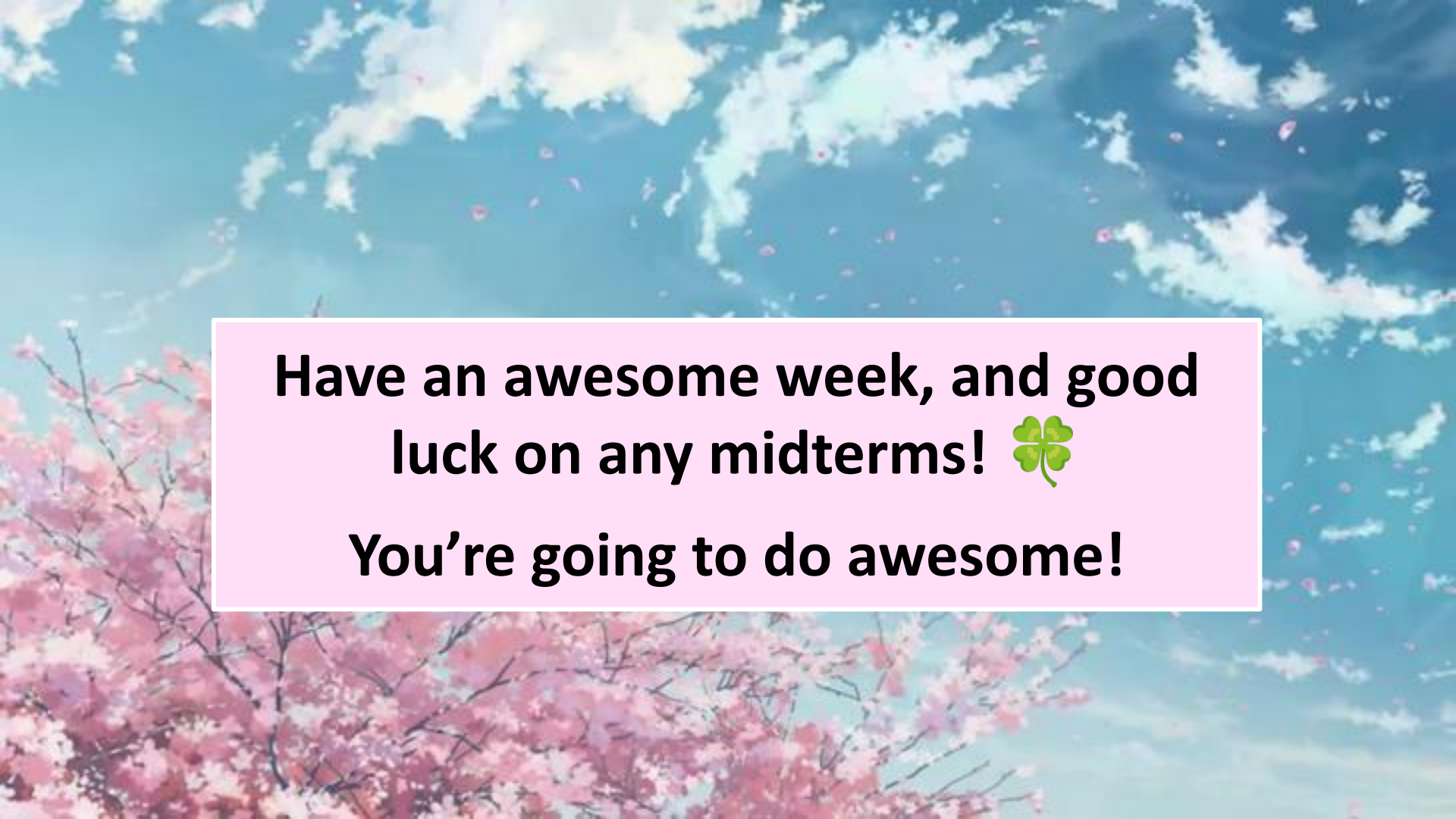
- **Stanford Applied Cyber**, Stanford Security Clinic

Check-Off Form

Another **brief check-off form** (< 5 min to complete) for checking attendance!

For today, click the “Check-Off Form” link in the **Week 5** section of **cs106s.stanford.edu**.

Thank you so much!


Have an awesome week, and good luck on any midterms! 🍀
You’re going to do awesome!