Building Secure UIs

CS571: Building User Interfaces

Cole Nelson

Announcements

See "No Title" Announcement

What will we learn today?

- How to (not) end up in jail?
- What are bug bounty programs?
- What are the OWASP Top 10?
- How do XSS attacks work?
- How do vulnerable dependencies create risk?
- Why is client-side validation insufficient?
- What are common mitigation strategies?

A Disclaimer

a badger wearing a red shirt with a w on it in jail for committing cybercrimes, pixel art

generated using DALL-E

ars TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STORE

VIEW SOURCE —

Viewing website HTML code is not illegal or “hacking,” prof. tells Missouri gov.

Professor demands that governor halt "baseless investigation" and apologize.

JON BRODKIN - 10/25/2021, 3:09 PM

Building Secure User Interfaces

Make sure you have permission to resources before performing security audits!

- HackerOne
- OpenBugBounty
- BugCrowd

Look for **safe harbors** or **local resources**.

TOP 10

OWASP Top 10 (2021)

1. Broken Access Control
2. Cryptographic Failures
3. Injection
4. Insecure Design
5. Security Misconfiguration

OWASP Top 10 (2021)

6. Vulnerable and Outdated Components
7. Identification and Authentication Failures
8. Software and Data Integrity Failures
9. Security Logging and Monitoring Failures
10. Server-Side Request Forgery

Common Vulnerabilities

Vulnerabilities affect both frontends *and* backends!

Today we will look at…

- Cross-Site Scripting (XSS)
  - Reflected/DOM-based XSS
  - Persistent XSS
- Vulnerable and Outdated Components
- Software and Data Integrity Failures

XSS

Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

OWASP Definition

HW2 XSS

Render each student using `innerHtml`

```
let html = "<div>";
html += `<h2>${student.name}</h2>`;
html += "</div>";
return html;
```

HW2 XSS

When input is `Michael` …

```
<div>
  <h2>Michael</h2>
</div>
```

`<i>Michael</i>` …

```
<div>
  <h2><i>Michael<i></h2>
</div>
```

HW2 XSS

```
<script>alert("oops!")\</script>
```
…

```
<div>
  <h2><script>alert("oops!")</script></h2>
</div>
```

```
<img src="0" onerror="alert(document.cookie)"/>
```
…

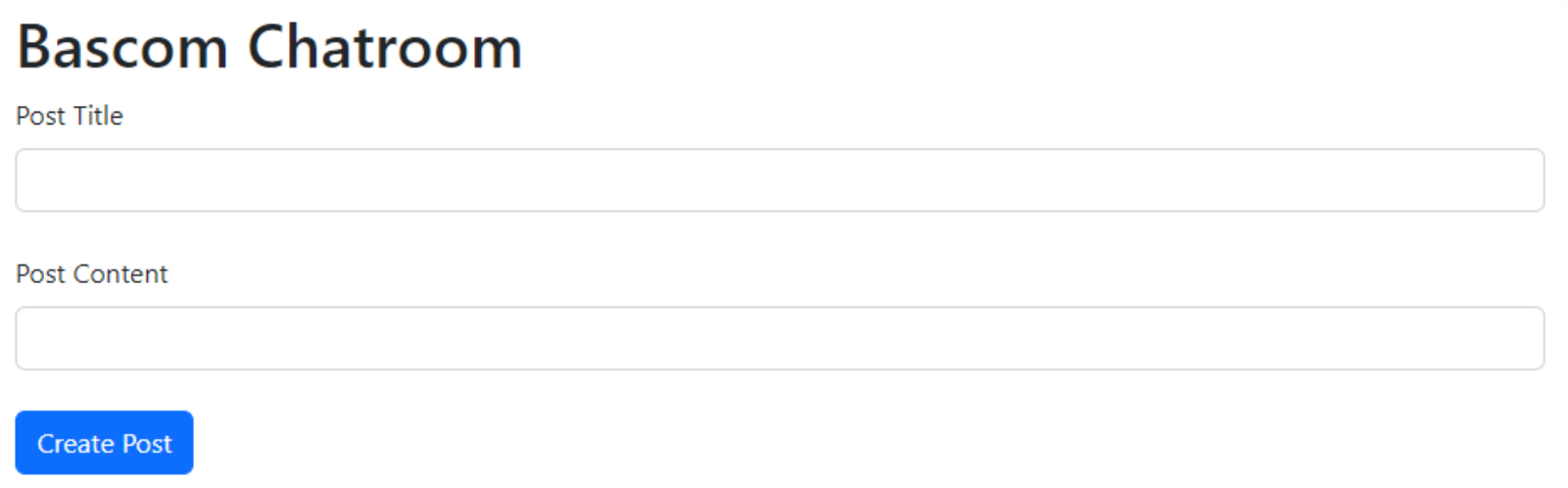
```
<div>
  <h2><img src="0" onerror="alert(document.cookie)"/></h2>
</div>
```

DOM-based vs Persistent XSS

DOM-based XSS

```
https://example.com/search?q=%3Cimg%20src=%220%22%20onerror=%22alert(1)%22/%3E
```

Persistent XSS



Samy (computer worm)

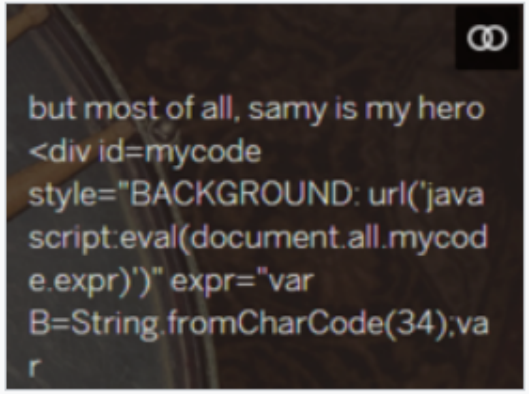
4 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Samy (also known as **JS.Spacehero**) is a cross-site scripting worm (XSS worm) that was designed to propagate across the social networking site MySpace by Samy Kamkar. Within just 20 hours[1] of its October 4, 2005 release, over one million users had run the payload[2] making Samy the fastest-spreading virus of all time.[3]

The worm itself was relatively harmless; it carried a payload that would display the string "but most of all, samy is my hero" on a victim's MySpace profile page as well as send Samy a friend request. When a user viewed that profile page, the payload would then be replicated and planted on their own profile page continuing the distribution of the worm. MySpace has since secured its site against the vulnerability.[1]



The message on a victim's profile

Samy Kamkar, the author of the worm, was raided by the United States Secret Service and Electronic Crimes Task Force in 2006 for releasing the worm.[4] He entered a plea agreement on January 31, 2007 to a felony charge.[5] The action resulted in Kamkar being sentenced to three years' probation with only one computer and no access to the Internet, 90 days' community service, and $15,000–20,000 in restitution, as directly reported by Kamkar himself on "Greatest Moments in Hacking History" by Vice Media's video website, Motherboard.[6]

XSS Demo w/ JuiceShop

Download here!

XSS Mitigations

Sanitize your inputs!

Do not create a sanitizer yourself!

React performs sanitization for you.

Image Source

XSS Demo w/ BadgerChat

BadgerChat is *NOT* a safe harbor -- please ask for permission before pentesting.

Use of Outdated and Vulnerable Components

```
C:\Users\ColeNelson\Desktop\cs571-s23\hws\hws\hw4>npm install

up to date, audited 1490 packages in 5s

233 packages are looking for funding
  run `npm fund` for details

7 high severity vulnerabilities
```

```
C:\Users\ColeNelson\Desktop\cs571-s23\hws\hws\hw4>npm audit
# npm audit report

nth-check  <2.0.1
Severity: high
Inefficient Regular Expression Complexity in nth-check - https://g
fix available via `npm audit fix --force`
Will install react-scripts@2.1.3, which is a breaking change
```

```
C:\Users\ColeNelson\Desktop\cs571-s23\hws\hws\hw4>npm ls nth-check
hw4@0.1.0 C:\Users\ColeNelson\Desktop\cs571-s23\hws\hws\hw4
`-- react-scripts@5.0.1
  +-- @svgr/webpack@5.5.0
  | `-- @svgr/plugin-svgo@5.5.0
  |   `-- svgo@1.3.2
  |     `-- css-select@2.1.0
  |       `-- nth-check@1.0.2
  `-- html-webpack-plugin@5.5.0
    `-- pretty-error@4.0.0
      `-- renderkid@3.0.0
        `-- css-select@4.3.0
          `-- nth-check@2.1.1
```

facebook / create-react-app Public

Sponsor | Watch 2k | Fork 26.3k | Star

Code | Issues 1.6k | Pull requests 426 | Discussions | Actions | Projects 3 | Security | Insights

main | 14 branches | 414 tags | Go to file | Add file | Code

jackblackevo docs: fix link of "Building for Relative Paths" (#12691) 3 d960b9e on Sep 8, 2022 2,800 commits

.github	chore(lint): lint all files (#12288)	last year
docusaurus	docs: fix link of "Building for Relative Paths" (#12691)	8 months ago
packages	Merge pull request #12563 from rvdende/patch-1	10 months ago
tasks	chore(lint): lint all files (#12288)	last year
test	chore(lint): lint all files (#12288)	last year
.alexignore	Add Alex to lint documentation (#7852)	4 years ago
.alexrc	Add Alex to lint documentation (#7852)	4 years ago
.eslintignore	chore(lint): lint all files (#12288)	last year

About

Set up a modern web app by running one command.

create-react-app.dev

react | zero-configuration | build-tools

Readme
MIT license
Code of conduct
Security policy
99.6k stars
2k watching
26.3k forks

Outdated and Vulnerable Components

A vulnerable dependency does not *necessarily* mean the application is vulnerable.

Likewise, an application without vulnerable dependencies *could still* be vulnerable.

Vulnerable dependencies are *flags* to look into.

Checking Vulnerabilities

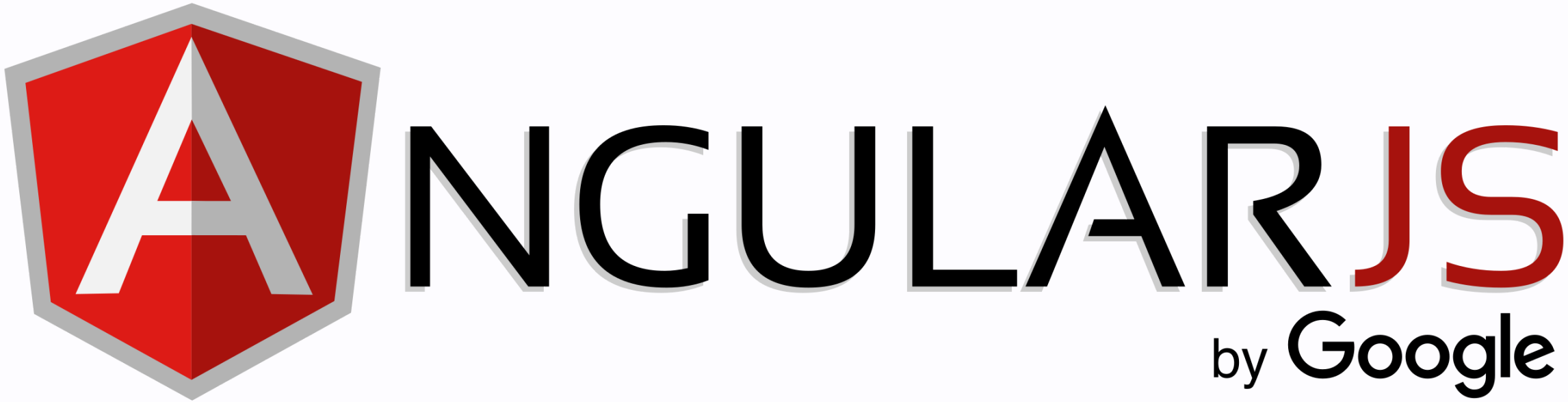
Check the CVE (Common Vulnerabilities and Exposures)

e.g. CVE-2021-3803...

- NIST
- GitHub
- Snyk

Does it have to be in the production build? Can it be specified as a dev dependency?

LOG4J


ANGULARJS
by Google

Outdated Dependency Mitigations

Keep your dependencies up-to-date!

- Static Analysis (SAST) Tools
  - OWASP DependencyCheck
- Continuous Maintenance
- Minimize Surface Area

Beware of changing technologies!

Software and Data Integrity Failures

Validation

Frontends are just a way of getting to the backend!

Do not rely solely on frontend validation.

The user can send more than you allow them to.

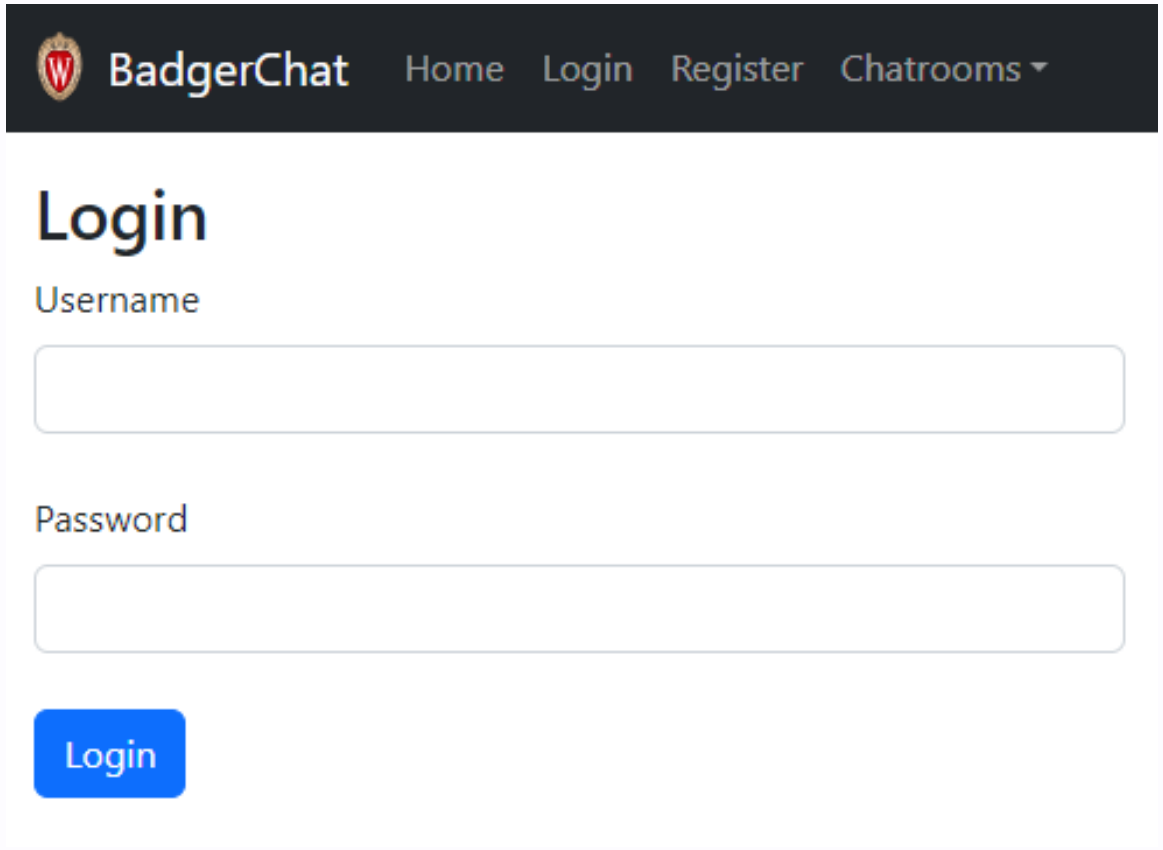


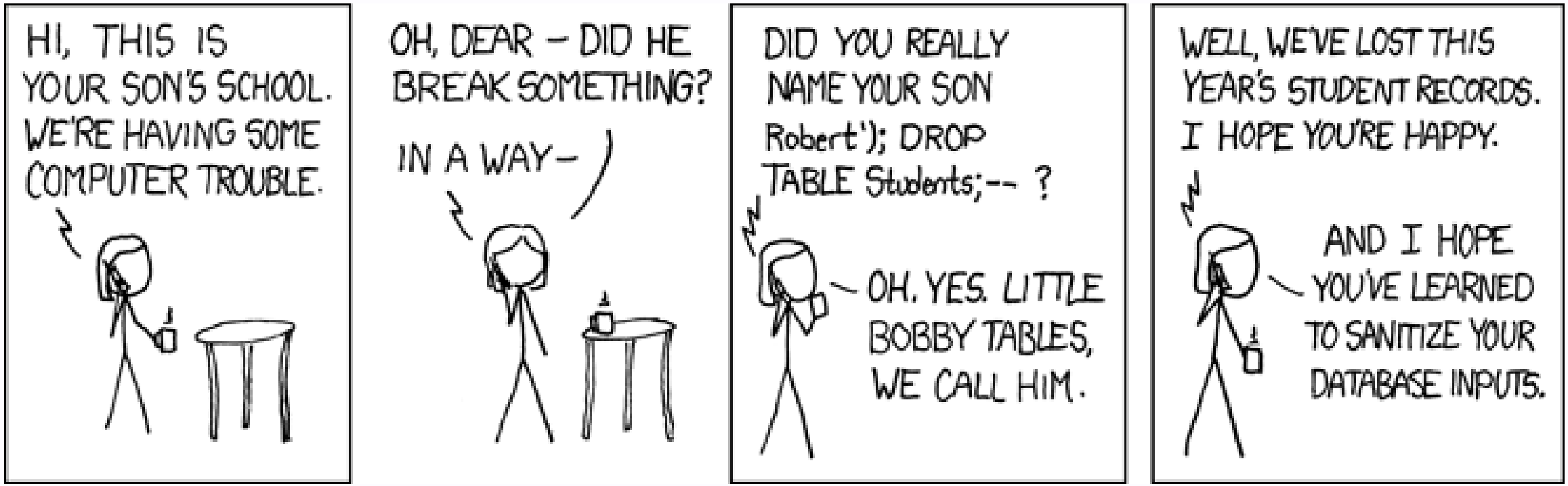
Image Source

Software and Data Integrity Failure Demo

On OWASP JuiceShop using OWASP ZAP

Backend Vulnerabilities

Where the real treasure lies!


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

XKCD 327

Mitigation Strategies

Technical strategies may include…

- Obfuscation
- Web Application Firewall (WAF)
- Containerization (Using Docker or VMs)
- Defense in Depth (Swiss Cheese Approach)

Mitigation Strategies

Non-technical strategies may include…

- Threat Modeling
- Least Privilege
- Scary Messages
- Ask Nicely :)

Questions?