Intro to Web Security

Lecture 9 - ECS198F SQ26

Plan of the Day

Auth & Access Controls

- What are ways that users get “authenticated” to sites, and how do we break them?
- Why do websites need “access controls?”

Code Injections

- What is the risk of trusting arbitrary user input on a website?
- What are ways we can exploit code injections to bypass logins, retrieve data, and gain shell access?

Known Exploits

- What are ways we can find and exploit known vulnerabilities in web applications?

1. Auth and Access Controls

Lock Down Your Sessions

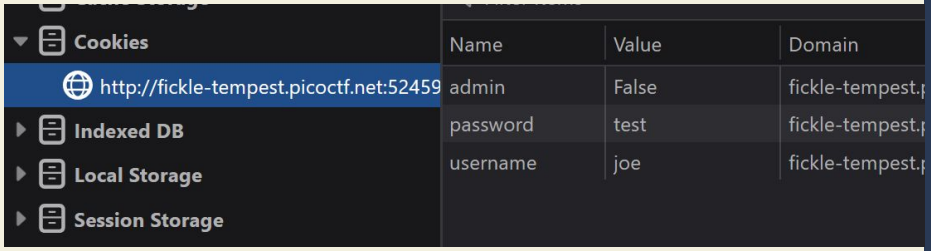
Review of Challenge:

- “Session” info stored in a cookie, which we could modify arbitrarily!

https://play.picoctf.org/practice/challenge/46

Solutions?

- “Stateful” Session Management
  - The server keeps a database of sessions!
- “Stateless” Session Management
  - “Encrypt” the cookie?

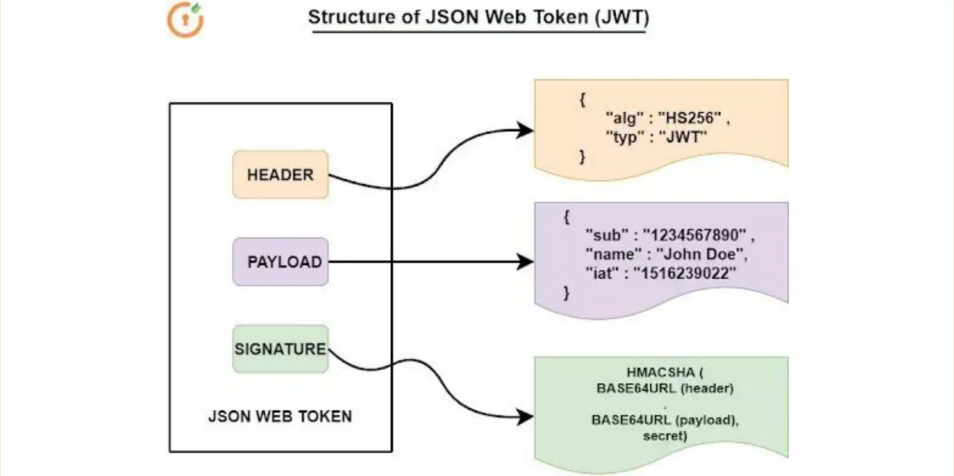


JSON Web Token (JWT)

Transfer “Cookie” data in a more secure format!

- Data encoded in JSON format
- **Header and Data hashed and signed!**
  - If you don’t have the “secret,” you can’t make changes to the included data without breaking verification!

https://www.jwt.io/



https://www.miniorange.com/blog/assets/2023/jwt-structure.webp

Attacking JWTs

What are some ways that JWTs can be compromised?

- “Not Checked Properly”
  - Arbitrarily accepting algorithm changes
  - What if they just don’t check the signature?
- Compromise the Secret?
  - Brute-force, examine source code, etc.

Challenges:

https://play.picoctf.org/practice/challenge/236
https://play.picoctf.org/practice/challenge/355

Examples of Insecure JWT Code:

https://github.com/AshleyBilbrey/insecure-bank/blob/118c0ebffc9fa3ebe16d0341accc32c879765ed8/src/pages/api/%5Binstance%5D/send.ts#L42

https://www.npmjs.com/package/jsonwebtoken

Broken Access Controls

Is there a way we can access things we probably shouldn't have access to?

Insecure Direct Object Reference

- If objects are numbered/accessed with predictable values, what happens when we "try" other values?
  - EX: id 1, id 2, id ….

Challenges:
https://tryhackme.com/room/idor-aoc2025-zl6MywQid9

References:
https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

Bypassing Login Screens?

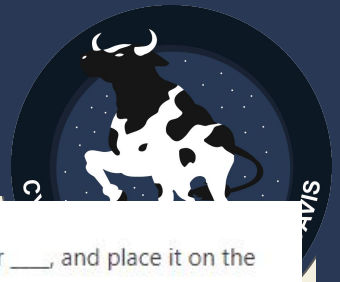
Can’t really “cookie” your way out of this one…

- When you “log in,” there HAS to be a check against some user database…
    - It has to run a “query” against the database using some kind of “structured language”

MySQL®

2. Code Injection Vulnerabilities

Code / Command Injection



- Imagine you have a factory robot that accepts instructions like this:

> Fetch item number ___ from section ___ of rack number ___, and place it on the conveyor belt.

  - Normally, you give instructions like:

> Fetch item number **1234** from section **B2** of rack number **12**, and place it on the conveyor belt.

  - But an **attacker** could **inject** instructions like:

> Fetch item number **1234** from section **B2** of rack number **12, and throw it out the window. Then go back to your desk and ignore the rest of this form.** and place it on the conveyor belt.

https://security.stackexchange.com/questions/25684/how-can-i-explain-sql-injection-without-technical-jargon

SQL Injection

Given a SQL Query, can I put SQL code "inside" the parameters to make the query do something else?

https://play.picoctf.org/practice/challenge/304

https://play.picoctf.org/practice/challenge/358

Authentication Bypass

- Make a query return "TRUE"

Information Disclosure

- Chain together techniques to make the database read something else

References:

https://swisskyrepo.github.io/PayloadsAllTheThings/SQL%20Injection/

Preventing SQL Injection

OK, is it possible to stop SQL injection?

What if we banned people from using certain SQL Injection characters?

- For every blocklist idea you have, there's ways of evading it!

Core Issue: We're "concatenating" user input into the query.

Solution: Parametrized Queries!

References:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

https://owasp.org/www-community/attacks/SQL_Injection_Bypassing_WAF

Preventing SQL Injection

OK, is it possible to stop SQL injection?

What if we banned people from using certain SQL Injection characters?

- For every blocklist idea you have, there's ways of evading it!

Core Issue: We're "concatenating" user input into the query.

Solution: Parametrized Queries!

References:

https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

https://owasp.org/www-community/attacks/SQL_Injection_Bypassing_WAF

Other Injection Forms

- **Cross-Site Scripting (XSS)**
  - Inject arbitrary HTML/Javascript code into a web page!
- **Directory Traversal**
  - Are you able to go to folders “outside” the web server?
- **Arbitrary File Injection/Execution**
  - What are the dangers of just running code?

XSS Chal (Hard!)
https://play.picoctf.org/practice/challenge/282
https://picoctf2022.haydenhousen.com/web-exploitation/noted

Directory Traversal
https://play.picoctf.org/practice/challenge/270

Arbitrary File Execution
https://tryhackme.com/room/mkingdom

3. Finding Existing Vulns

Leave No Stone Unturned (Again)

Sometimes, you just need to brute force a solution.

Problem: There may be “hidden” sites/folders/API endpoints on the website that might not be findable by clicking all the links.

Solution: Try a bunch of common directory names to find hidden folders.

Dirbuster, gobuster, ffuf, etc. *combined* with a wordlist!

https://tryhackme.com/room/mkingdom

WARNING: Bruteforcing is essentially one-step removed from a DDoS. Brute Force with Care!

Finding Well-Known Vulns

Not all vulnerabilities are new…

- Old, unpatched software might have known vulnerabilities that have been reported and made public!
- Is it possible to find an EXISTING exploit for these issues instead of having to cook up your own?

https://cve.org/
https://www.exploit-db.com/

CVE-2017-0144 Detail

Description

The SMBv1 server in Microsoft Windows Vista SP2; Windows Server 2008 SP2 and R2 SP1; Windows 7 SP1; Windows 8.1; Windows Server 2012 Gold and R2; Windows RT 8.1; and Windows 10 Gold, 1511, and 1607; and Windows Server 2016 allows remote attackers to execute arbitrary code via crafted packets, aka "Windows SMB Remote Code Execution Vulnerability." This vulnerability is different from those described in CVE-2017-0143, CVE-2017-0145, CVE-2017-0146, and CVE-2017-0148.