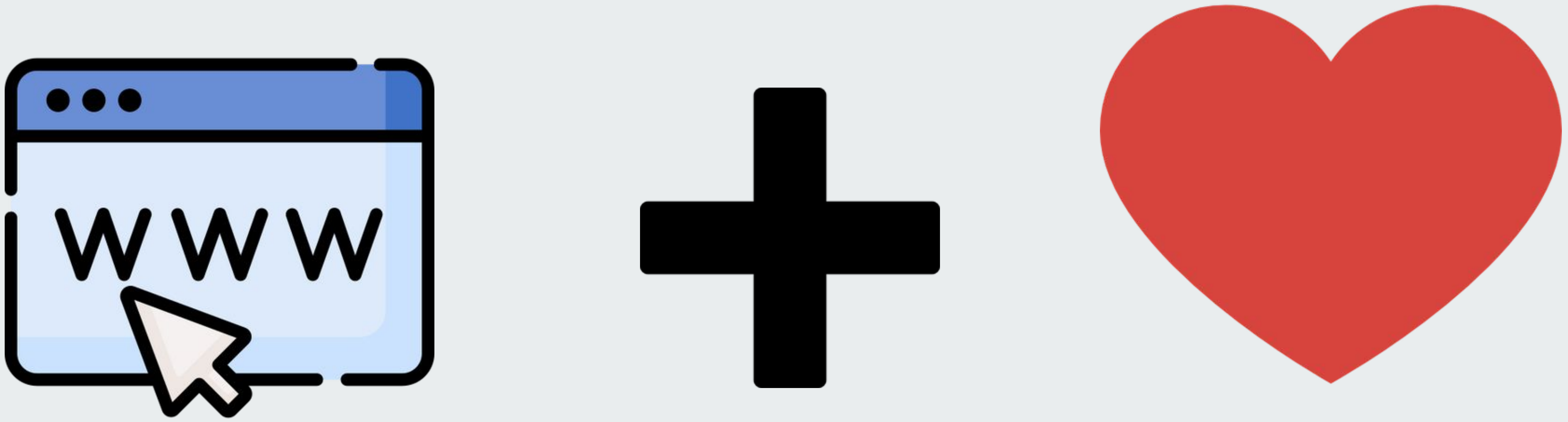
Hack za web with <3



vicevirus @ CSLU UTP

13 May 2024

whoami

- Active web CTF Player
- MCC 2023 and GCC 2024 alumni
- Wannabe web developer who has now become a wannabe hacker
- Spends weekend playing international CTFs on CTFtime

Why web?

Why web?

- Almost everything you see nowadays is a part of web technology
- Your student portal? Yeah it's web
- Your favourite social media? Yeah it connects to a web API in the background, still web
- Hence, hacking web = hacking the world

Disclaimer: even if you can, doesn't mean you should. hack ethically

Most of the web vulnerabilities comes from how the web itself is implemented.

The source code will always tell you everything you need to know.

Sanitizing user inputs is the most crucial part.

Let’s dive into most common vulnerabilities in web

(with a lil bit of CTF-added spice)

SQL Injection

SQL Injection

- SQL injection is a code injection technique that exploits vulnerabilities in a web application by inserting or manipulating SQL queries through user input.
- One of the most common vulnerability back during the days.
- Still happens today, but rare. (Thanks ORM and frameworks)
- You might encounter them in CTFs.

Impact

- Allows attacker to run their own SQL code.
- The attacker is able to do anything on the database.
- Reading and updating the database? easy.
- Dropping the whole database? wayy easier.
- In short: SQL Injection is fun for hackers, but scary for business owners/developers

Hands-on

- Challenge link: sqlichall.vicevirus.me
- Bypass the login using SQL Injection.

CTF Tips

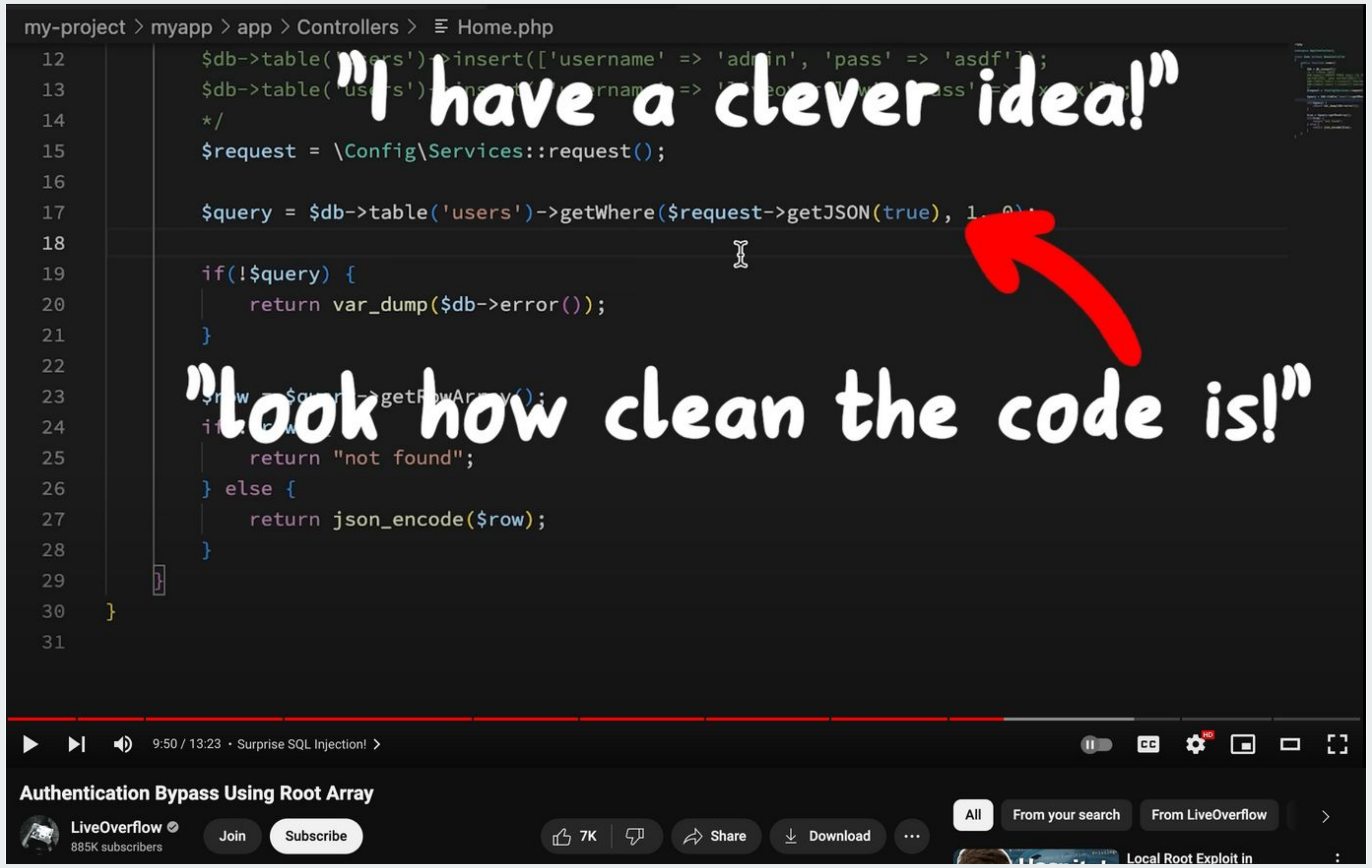
- Always read the source code. If the user input is directly appended into the SQL query, most likely SQL injection is possible.

- There are multiple type of SQL injections. Get familiar with them.

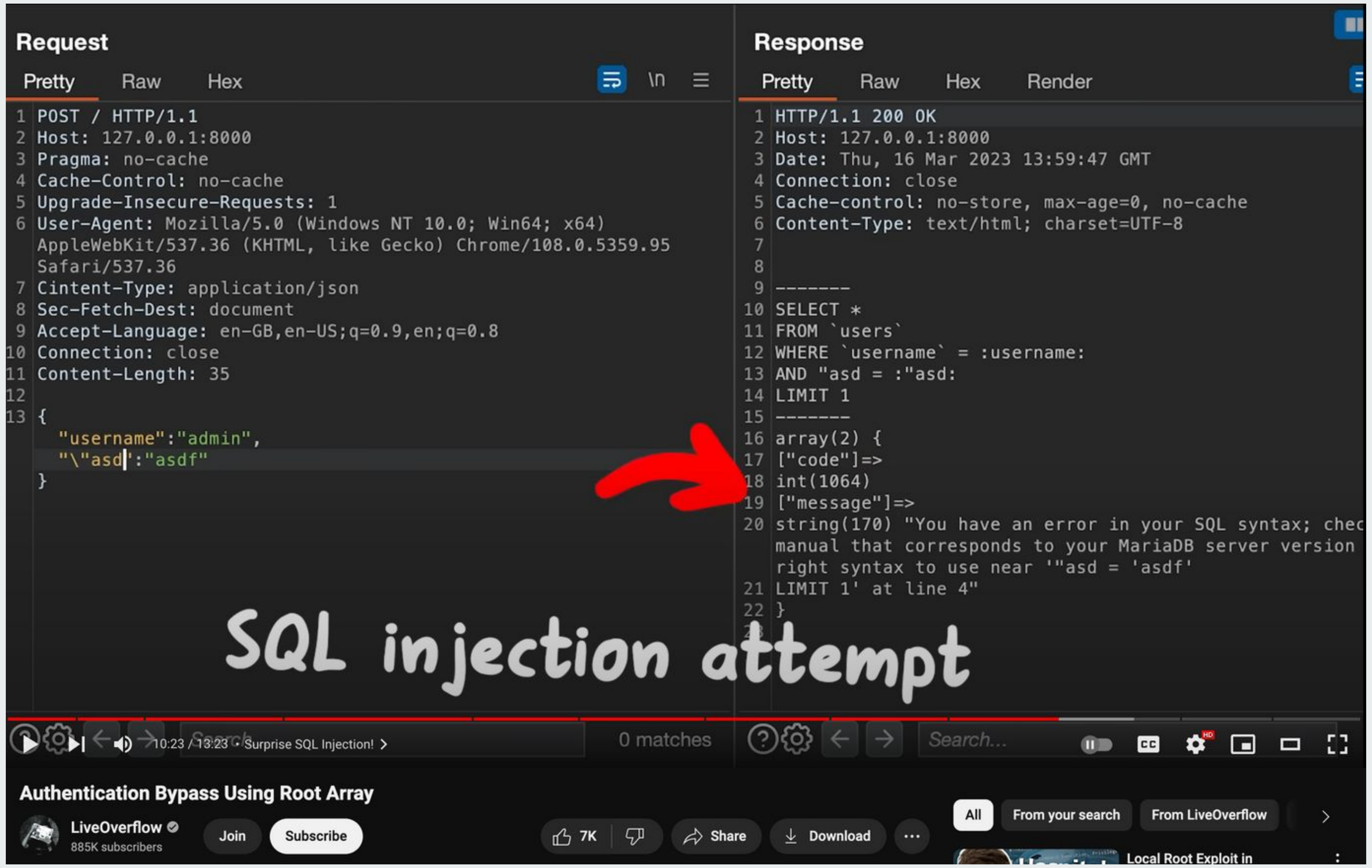
- Send single quote or double quote first, if any DB error pops up, could be a valid SQL injection vector.

- CTF questions will purposely make it harder for you by blocking few SQL functions such as UNION, AND, OR etc. Also, sometimes it could be using the correct tools, but incorrect implementation.

Example of incorrect implementation


my-project > myapp > app > Controllers > Home.php
"I have a clever idea!"
"look how clean the code is!"
9:50 / 13:23 · Surprise SQL Injection!
Authentication Bypass Using Root Array
LiveOverflow
885K subscribers
Join
Subscribe
7K
Share
Download
All
From your search
From LiveOverflow
Local Root Exploit in

```php
        $db->table('users')->insert(['username' => 'admin', 'pass' => 'asdf']);
        $db->table('users')->insert(['username' => '[illegible]', 'pass' => '[illegible]']);
        */
        $request = \Config\Services::request();

        $query = $db->table('users')->getWhere($request->getJSON(true), 1, 0);

        if(!$query) {
            return var_dump($db->error());
        }

        $row = $query->getRowArray();
        if(!$row) {
            return "not found";
        } else {
            return json_encode($row);
        }
    }
}
```

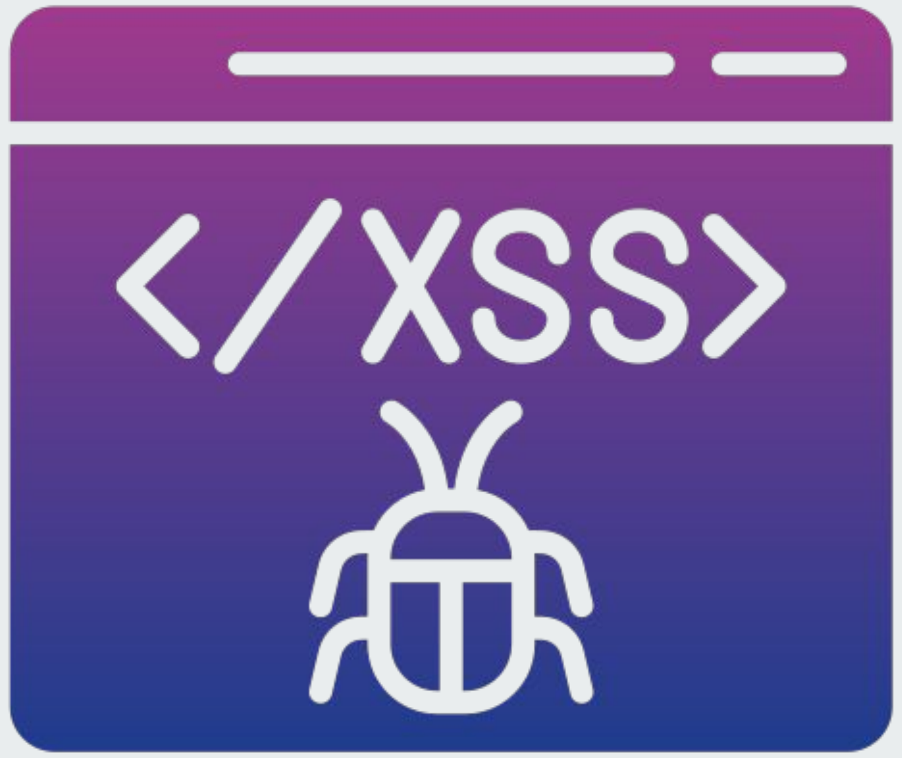


```
POST / HTTP/1.1
Host: 127.0.0.1:8000
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.5359.95
Safari/537.36
Cintent-Type: application/json
Sec-Fetch-Dest: document
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: close
Content-Length: 35

{
  "username":"admin",
  "\"asd":"asdf"
}
```

```
HTTP/1.1 200 OK
Host: 127.0.0.1:8000
Date: Thu, 16 Mar 2023 13:59:47 GMT
Connection: close
Cache-control: no-store, max-age=0, no-cache
Content-Type: text/html; charset=UTF-8


--------
SELECT *
FROM `users`
WHERE `username` = :username:
AND "asd = :"asd:
LIMIT 1
--------
array(2) {
["code"]=>
int(1064)
["message"]=>
string(170) "You have an error in your SQL syntax; chec
manual that corresponds to your MariaDB server version
right syntax to use near '"asd = 'asdf'
LIMIT 1' at line 4"
}
```

LiveOverflow: Authentication Bypass using Root Array

XSS (Cross Site Scripting)



XSS

- XSS is a vulnerability on the web which allows a user to run Javascript freely on the victim’s browser.
- One of the most common vulnerability.
- While there’s a lot of mitigation, it is still impactful.
- You will encounter in them in CTFs especially international ones.

Impact

- Steal yo cookies n use yo account
- (now it’s much more difficult with HTTPonly flag)
- Steal contents of the page you viewed.
- Manipulate the page.
- And more! JavaScript is so potent.

Hands-on

- Challenge link: xsschall.vicevirus.me
- Run any Javascript
- Share with your friend a redirect to Rick n Roll >:)

CTF Tips

- There will always be a source code for XSS challenge in CTF.
- You can easily identify if it’s a XSS challenge. Usually, the challenge has a bot (e.g Puppeteer/Selenium script) in the source code and the flag is inside the bot’s cookie.
- Usually in CTFs, the goal is to steal the bot’s cookie
- Such challenge sometimes is made harder by adding naive filters, past vulnerabilities and more.

Local File Inclusion

Local File Inclusion

- Local File Inclusion (LFI) is a security vulnerability in web applications that allows attackers to access, view, and sometimes include files on a web server.

- Not sure if I have seen this in the wild, but I’ve seen someone found them on bug bounty.

- In CTFs, the challenge author usually overlooks certain aspect of the challenge, which makes it much easier to solve sometimes.

Impact

- Allow attackers to read any file on the system.
- Attackers can easily leak out secrets, configuration and sensitive information about the system.
- In worst case, Remote Code Execution (RCE) is possible too.

Hands-on

- Challenge link: lfichall.vicevirus.me
- Read /flag.txt
- Execute RCE if you can?

CTF Tips

- In 50-70% cases, the way author implements LFI is by using include(), include_once(), require()
- These functions are dangerous in PHP and allows for Remote Code Execution (RCE).
- It’s difficult to stop RCE when you’re using these functions.
- Hence, you could use this to your advantage!
- file_get_contents() and readfile() isn’t affected by this RCE problem.

Key takeaways

- Always, always, always sanitize user inputs.
- Using a web framework to build websites is much more preferred and secure rather than building from scratch.
- There’s a lot of stuff in web CTFs, understanding all of them takes a lot of time.
- Be patient, keep on playing and you’ll git gud eventually.
- Don’t stress too much. The point is to learn and have fun. That is the most important part.

Thank you and goodluck!

- If you have any questions, feel free to reach out to me on Discord @vicevirus or my LinkedIn.