

ACM

Week 3 writeup:

NAME: B RAJMOHITH REDDY

BRANCH: CYBER SECURITY

SIG: CYBER

TASKS WEEK 3

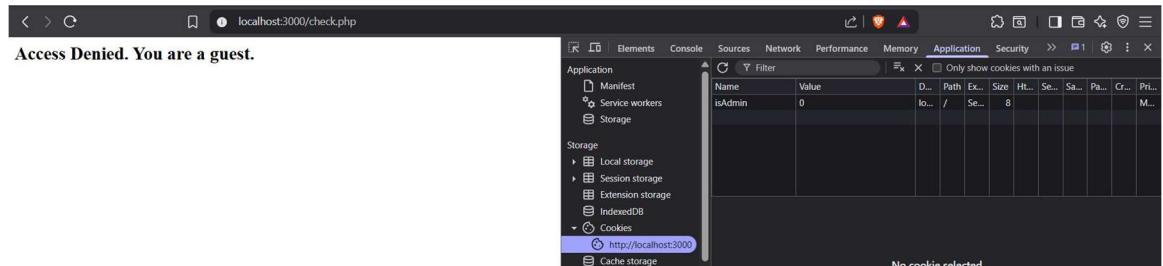
TASK 1:

- i) I have entered into to <http://localhost:3000>.

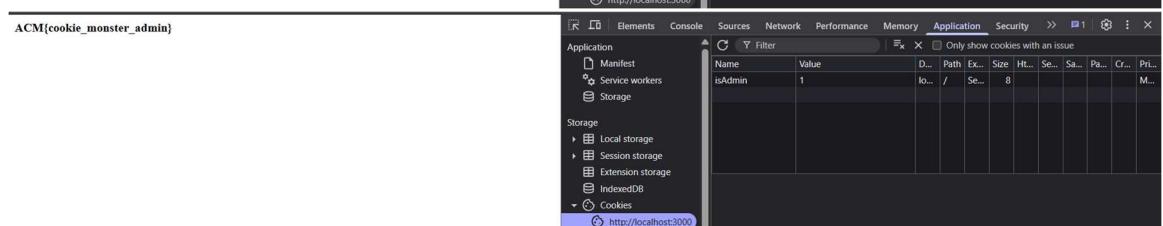
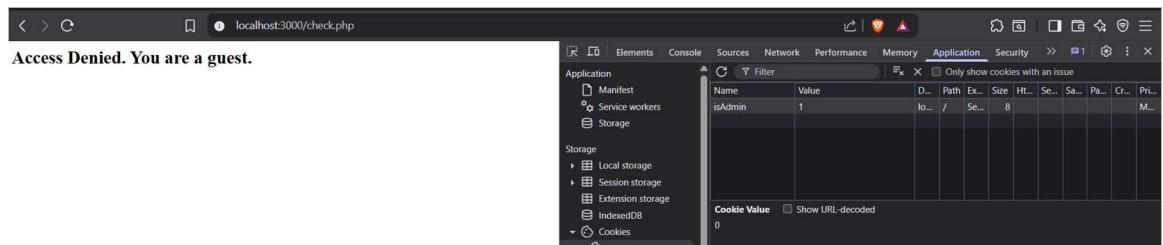


Access Denied. You are a guest.

- ii) Then on inspecting the page I found that the value of cookie for localhost is 0:



- iii) Then I tried it to change it to 1 and reloaded the page so that I found the flag:



So the flag is : **ACM{COOKIE_MONSTER_ADMIN}**

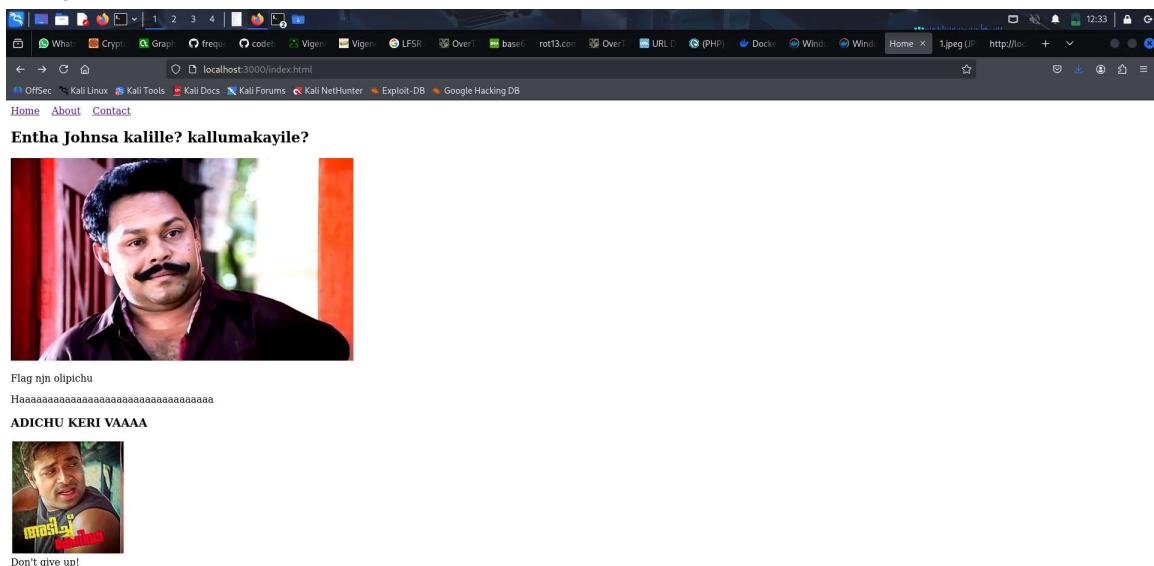
Tech explanation:

- iv) This is a cookie-based access control vulnerability. So in this challenge the thing is the server gets the value of cookie as 0 which means the server decides that the user is a guest and if we change that to 1 to make the server to feel that the user is the admin and show the data. Hence, we find the data.
- v) It is also some sort of cookie manipulation or tampering.

TASK 2:

- i) This is a base64 flag that is found in about.html page.
- ii) On inspecting the about.html page we will find a base64 encoded data that can be decoded using an online base64 decoder.
- iii) The flag is “RmxhZ3tpbnNwZWN0X2ZsYWdfY2hhbmdlfQ==”.

Upon opening the local host I found this and tried to find the flag by saving the pictures that does not work.



- iv) so I tried to find the hidden directories in this page using “dirb”

```
(mohith@kali)-[~]
$ dirb http://localhost:3000

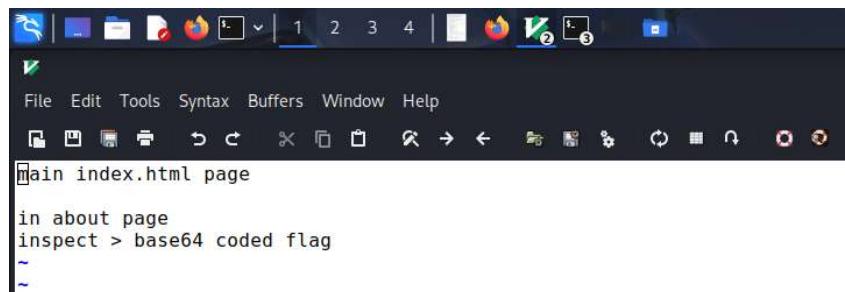
DIRB v2.22
By The Dark Raver

START_TIME: Sun Aug 3 12:35:18 2025
URL_BASE: http://localhost:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

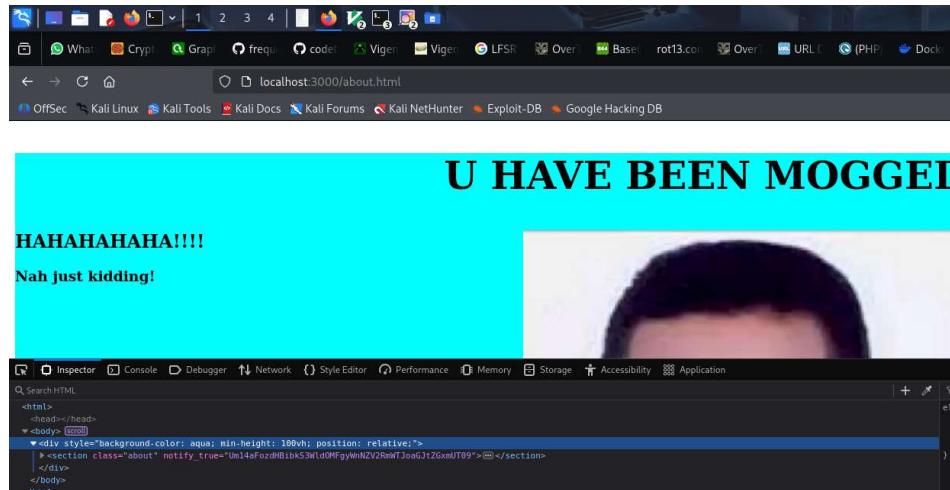
_____
GENERATED WORDS: 4612
_____
Scanning URL: http://localhost:3000/
+ http://localhost:3000/index.html (CODE:200|SIZE:1390)
+ http://localhost:3000/readme (CODE:200|SIZE:64)

_____
END_TIME: Sun Aug 3 12:35:24 2025
DOWNLOADED: 4612 - FOUND: 2
```

- v) here I found that there is a readme thing so I tried going to I using <http://localhost:3000/readme>.



- vi) this gave me an idea to get the base64 encoded data as below by navigating to about page and inspecting as below.

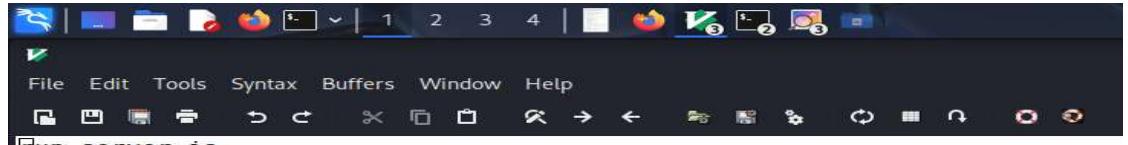


Tech explanation:

- vi) This technique is a encoded data hiding in the coded scripts and the flag is hidden most commonly base64, ROT13, hex and URL encoded form.
- vii) This is just the analysis of the web page scripts.

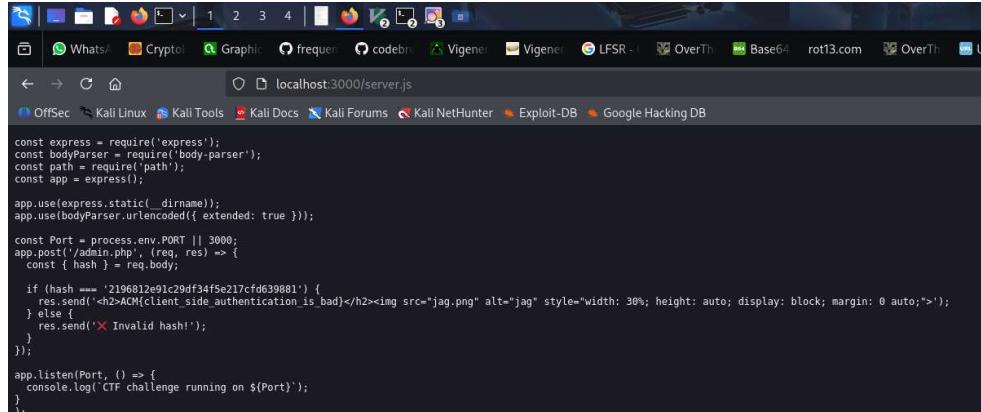
TASK 3:

- i) I tried to find the hints in this. So, I tried to open “readme” and I used <http://localhost:3000/readme>.
- ii) Then I found the following details



```
run server.js
hidden action file : admin.php
which is secure.js which contain username and password
on entering the creds -> FLAG
```

- iii) so I modified the url to <http://localhost:3000/server.js>
then that has the following data.



```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const app = express();

app.use(express.static(__dirname));
app.use(bodyParser.urlencoded({ extended: true }));

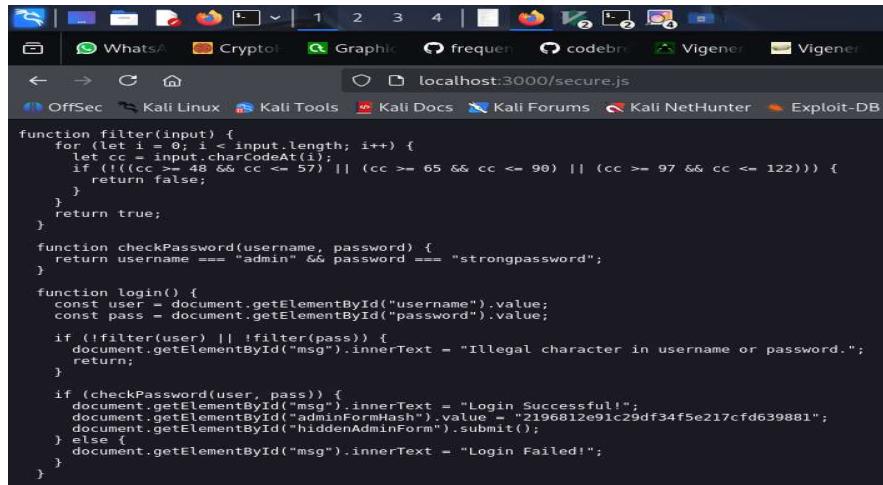
const Port = process.env.PORT || 3000;
app.post('/admin.php', (req, res) => {
  const { hash } = req.body;

  if (hash === '2196812e91c29df34f5e217cf639881') {
    res.send('<h2>ACM(client_side authentication is bad)</h2>');
  } else {
    res.send('X Invalid hash!');
  }
});

app.listen(Port, () => {
  console.log(`CTF challenge running on ${Port}`);
});
```

But this confused me so I re-checked the data given in readme again so I found that we can also try <http://localhost:3000/secure.js>.

- iv) In secure.js I found the following data



```
function filter(input) {
  for (let i = 0; i < input.length; i++) {
    let cc = input.charCodeAt(i);
    if (!(cc >= 48 && cc <= 57) || (cc >= 65 && cc <= 90) || (cc >= 97 && cc <= 122))) {
      return false;
    }
  }
  return true;
}

function checkPassword(username, password) {
  return username === "admin" && password === "strongpassword";
}

function login() {
  const user = document.getElementById("username").value;
  const pass = document.getElementById("password").value;

  if (!filter(user) || !filter(pass)) {
    document.getElementById("msg").innerText = "Illegal character in username or password.";
    return;
  }

  if (checkPassword(user, pass)) {
    document.getElementById("msg").innerText = "Login Successful!";
    document.getElementById("adminFormHash").value = "2196812e91c29df34f5e217cf639881";
    document.getElementById("hiddenAdminForm").submit();
  } else {
    document.getElementById("msg").innerText = "Login Failed!";
  }
}
```

- v) In that I found some data like “username === "admin" && password === "strongpassword";
vi) So I tried that credentials and found the following



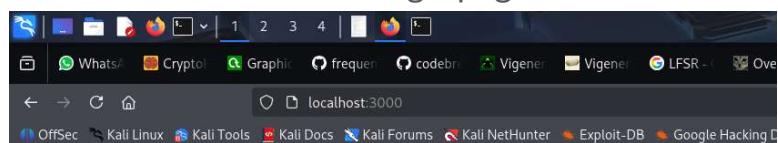
- vii) The flag is “ACM {client_side_authentication_is_bad}”.

Tech explanation:

- i) This is a kind of static code analysis
- ii) This is a kind of static code analysis and authentication bypass with disclosure credentials.
- iii) Even this is some sort of credentials harvesting where we will find the credentials in .js scripts.

Task 4:

- i) In this task we will find a login page like shown below



- ii) To find the hidden files/directories in this we will use “dirb”

```
(mohith㉿kali)-[~]
$ dirb http://localhost:3000/
_____
DIRB v2.22
By The Dark Raver

_____
START_TIME: Mon Aug 4 10:53:04 2025
URL_BASE: http://localhost:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____
GENERATED WORDS: 4612
_____
Scanning URL: http://localhost:3000/ _____
+ http://localhost:3000/index.html (CODE:200|SIZE:562)
+ http://localhost:3000/readme (CODE:200|SIZE:119)

_____
END_TIME: Mon Aug 4 10:53:11 2025
DOWNLOADED: 4612 - FOUND: 2

(mohith㉿kali)-[~]
$ █
```

- iii) Tried to open readme file and found the following details. These details helped me to find the flag.

```
inside index.html there is index.js  
which contain unformatted username and password in base58 encoded  
  
flag is password
```

- IV) Upon navigating to index.js I found some information shown in the picture.

- v) This gave me a key set and that are the login credentials but that data is base 58 encoded form.
 - vi) The login credentials are : “username: admin” , “password: FLAG {base64_is_outdated}”.
 - vii) And here that is mentioned as that the “flag is password” so the required flag is “FLAG {base64_is_outdated}”.

Tech explanation:

- viii) This is some kind of web exploitation or a type of information disclosure where we will find the required things in .js file which is hidden.

Task 5:

- i) Upon opening this I found something like leave a comment box and tried to give some input to find any clue.

Simple Login Bypass....{its simple!}

Comments:

tasks are nice

[Go back](#)

Leave a comment

tasks are nice

Submit

- ii) I have tried to find the hidden files/directories.
- iii) I used “dirb” to find those files but found no details. That just ran a simple search and gave the following output.

```
File Actions Edit View Help
(mohith㉿kali)-[~]
$ dirb http://localhost:3000/
DIRB v2.22
By The Dark Raver

START_TIME: Mon Aug 4 16:33:50 2025
URL_BASE: http://localhost:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt its simu

GENERATED WORDS: 4612
Scanning URL: http://localhost:3000/ ----

END_TIME: Mon Aug 4 16:33:59 2025
DOWNLOADED: 4612 - FOUND: 0
(mohith㉿kali)-[~]
```

- iv) Then I tried to inspect the page and analyzed the cookie data Which has the flag data encoded in URL form as following “FLAG%7Bstolen_cookie_flag%7D”

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite
flag	FLAG%7Bstolen_cookie_flag%7D	localhost	/	Session	32	false	false	None
isAdmin	10	localhost	/	Session	9	false	false	None

- v) So, the decoded data is “FLAG {stolen_cookie_flag}”.

Tech explanation:

- i) This is a cookie manipulation or cookie tampering or also some sort of cookie-based access control which hides the flag in cookie data.