Cybersecurity Wargame

Internship: Digisuraksha Parhari Foundation
Team Name: Team Saniqra
College Name: Sathaye College
Team Members:

- Saniya Santosh Choughule
- Iqra Mohd Ali Khan

Roles of Members:
Saniya: Exploitation and Data Collection

- **Solved all KRYPTON, NATAS, and LEVIATHAN levels.**
- **Documented commands, payloads, and steps taken.**
- **Captured screenshots of key outputs and exploits.**
- **Summarized vulnerabilities and methods used.**

Iqra: Report Writing and Compilation

- **Structured and formatted the complete report.**
- **Inserted screenshots and polished documentation.**
- **Wrote introduction, setup, reflections, and conclusions.**

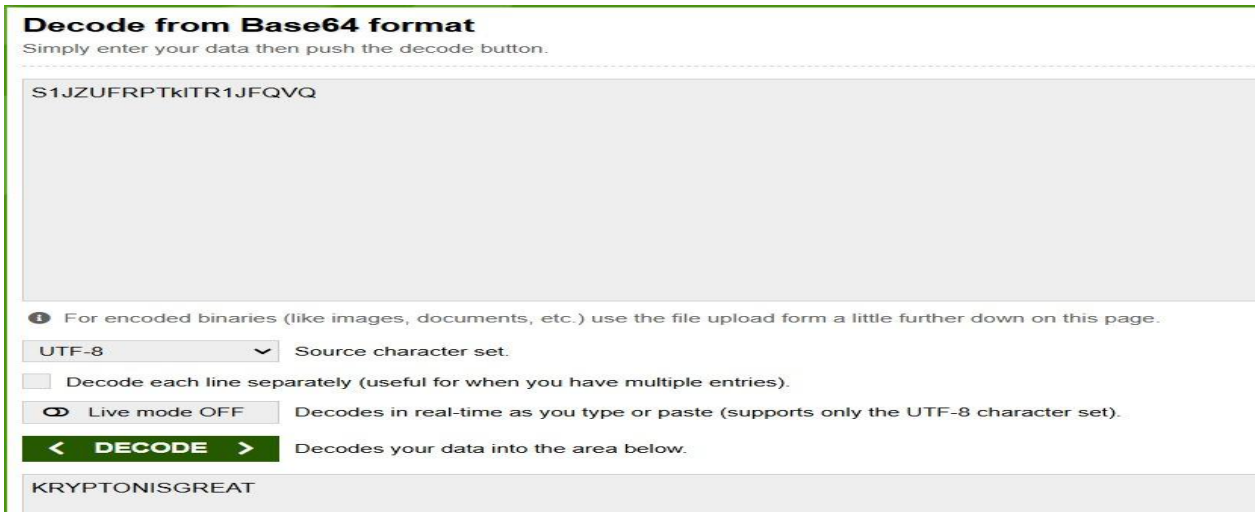
◇ **Lab Name: [KRYPTON / NATAS / LEVIATHAN]**

Lab Name: KRYPTON
Url: https://overthewire.org/wargames/krypton/

Level 0 ➔ Level 1
1. Problem Description:
We were given an encrypted text using a basic cipher and had to decrypt it to get the password.
2. Approach:
I identified the cipher as a ROT13 cipher and used basic decoding techniques.
3. Commands/Tools Used:

- cat krypton1.txt
- tr A-Za-z N-ZA-Mn-za-m < krypton1.txt
- Linux Terminal

4. Solution:
Using ROT13 decoding, I obtained the password needed for the next level.
5. Screenshot:



Level 1 ➔ Level 2

1. Problem Description:

We had another ciphered text, slightly more difficult than ROT13.

2. Approach:

Recognized it as a simple Caesar cipher with a shift. Used CyberChef to easily decode it.

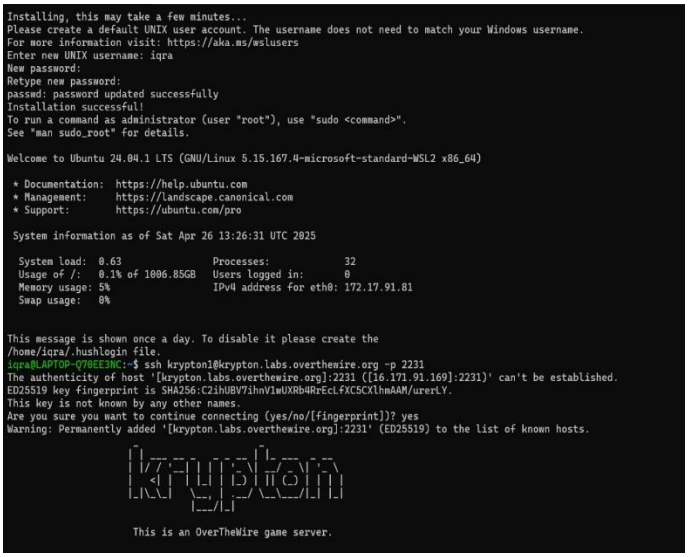
3. Commands/Tools Used:

- CyberChef
- cat krypton2.txt

4. Solution:

By trying different Caesar cipher shifts, I found the password to access the next level.

5. Screenshot:



Level 2 ➔ Level 3

1. Problem Description:

Challenge involved solving a cipher based on a known plaintext attack.

2. Approach:

Analyzed the cipher with frequency analysis and guessed the substitution pattern.

3. Commands/Tools Used:

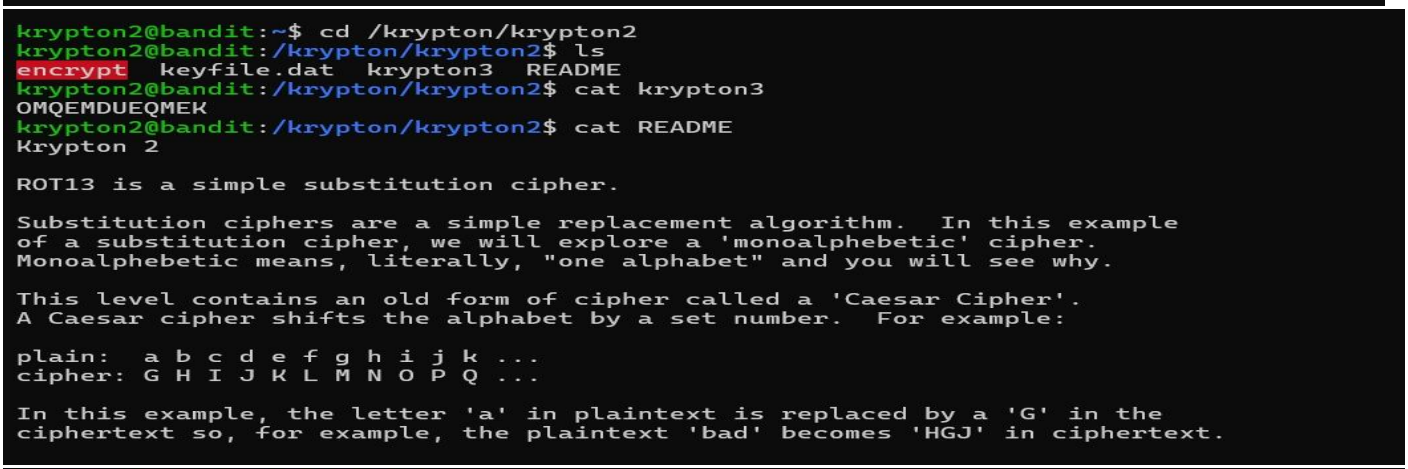
- CyberChef

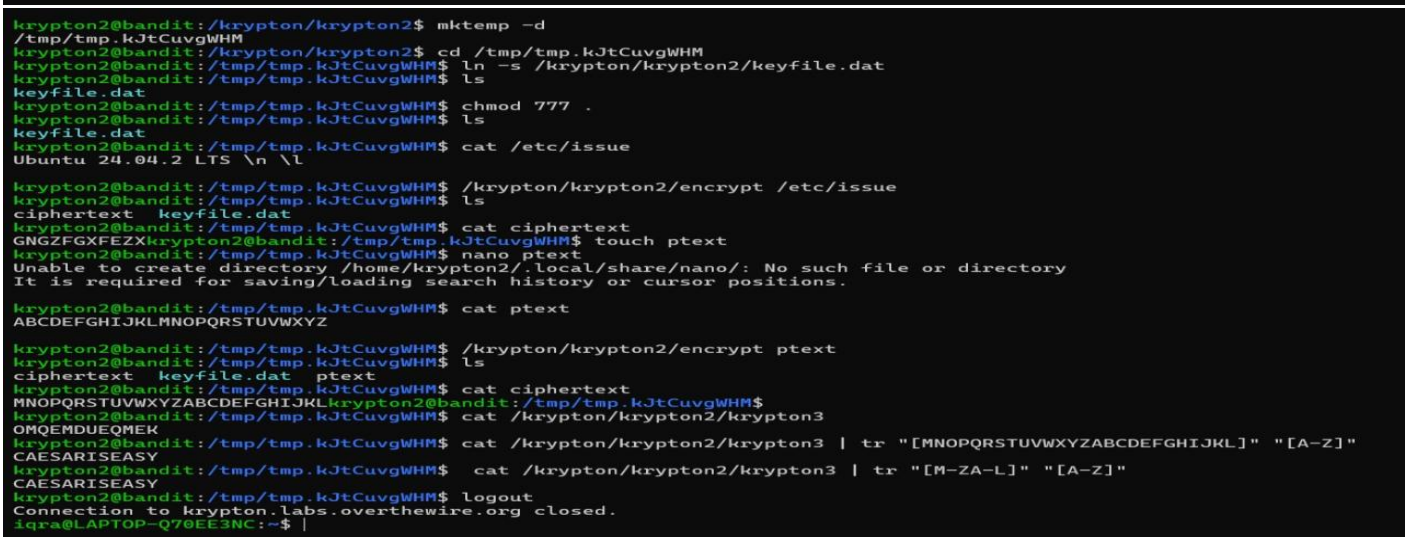
- Online Substitution Cipher Solvers

4. Solution:

Manually mapped and decoded the ciphertext to reveal the password.

5. Screenshot:





Level 3 ➔ Level 4

1. Problem Description:

We were given a set of numbers representing encrypted text.

2. Approach:

Identified it as a hexadecimal encoding and converted hex to ASCII.

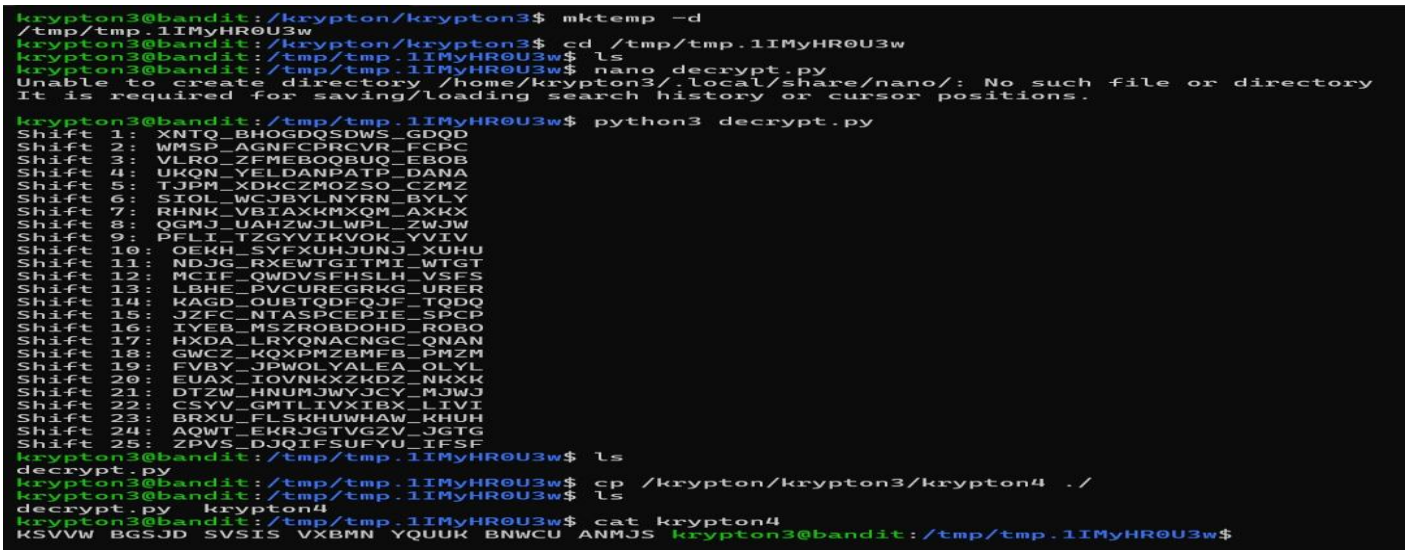
3. Commands/Tools Used:

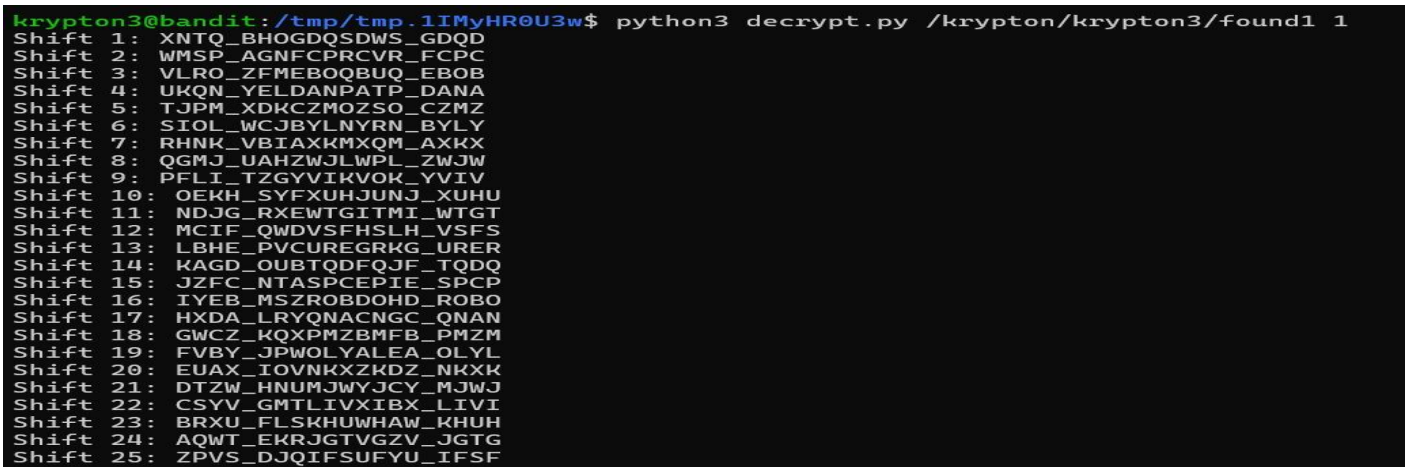
- xxd
- CyberChef
- echo [hexdata] | xxd -r -p

4. Solution:

Decoding the hex values revealed the password.

5. Screenshot:





```
krypton3@bandit:/tmp/tmp.1IMyHR0U3w$ python3 decrypt.py /krypton/krypton3/found2 1
Shift 1:  XNTQ_BHOGDQSDWS_GDQD
Shift 2:  WMSP_AGNFCPRCVR_FCPC
Shift 3:  VLRO_ZFMEBOQBUQ_EBOB
Shift 4:  UKQN_YELDANPATP_DANA
Shift 5:  TJPM_XDKCZMOZSO_CZMZ
Shift 6:  SIOL_WCJBYLNYRN_BYLY
Shift 7:  RHNK_VBIAXKMXQM_AXKX
Shift 8:  QGMJ_UAHZWJLWPL_ZWJW
Shift 9:  PFLI_TZGYVIKVOK_YVIV
Shift 10:  OEKH_SYFXUHJUNJ_XUHU
Shift 11:  NDJG_RXEWTGITMI_WTGT
Shift 12:  MCIF_QWDVSFHSLH_VSFS
Shift 13:  LBHE_PVCUREGRKG_URER
Shift 14:  KAGD_OUBTQDFQJF_TQDQ
Shift 15:  JZFC_NTASPCEPIE_SPCP
Shift 16:  IYEB_MSZROBDOHD_ROBO
Shift 17:  HXDA_LRYQNACNGC_QNAN
Shift 18:  GWCZ_KQXPMZBMFB_PMZM
Shift 19:  FVBY_JPWOLYALEA_OLYL
Shift 20:  EUAX_IOVNKXZKDZ_NKXK
Shift 21:  DTZW_HNUMJWYJCY_MJWJ
Shift 22:  CSYV_GMTLIVXIBX_LIVI
Shift 23:  BRXU_FLSKHUWHAW_KHUH
Shift 24:  AQWT_EKRJGTVGZV_JGTG
Shift 25:  ZPVS_DJQIFSUFYU_IFSF
```

```
krypton3@bandit:/tmp/tmp.1IMyHR0U3w$ cat krypton4 | tr "[JDDSOBKV]" "[THEAOWL]"
LA]]W WGATE A]AIA ]XWMN YQUUL WNWCU ANMTA krypton3@bandit:/tmp/tmp.1IMyHR0U3w$
krypton3@bandit:/tmp/tmp.1IMyHR0U3w$ cat krypton4 | tr "[JDSOBKV]" "[THEAOWL
]"
WELLW OGETH ELEIE LXOMN YQUUW ONWCU ANMTE krypton3@bandit:/tmp/tmp.1IMyHR0U3w$
krypton3@bandit:/tmp/tmp.1IMyHR0U3w$ cat krypton4 | tr "[JDSQBKVIWG]" "[THEA
OWLVDN]"
WELLD ONETH ELEVE LXOMN YAUUW ONDCU ANMTE krypton3@bandit:/tmp/tmp.1IMyHR0U3w$
krypton3@bandit:/tmp/tmp.1IMyHR0U3w$ cat krypton4 | tr "[JDSQBKVIWGY]" "[THE
AOWLVDNP]"
krypton3@bandit:/tmp/tmp.1IMyHR0U3w$ cat krypton4 | tr "[JDSQBKVIWGYUN]" "[T
HEAOWLVDNPSR]"
WELLD ONETH ELEVE LXOMR PASSW ORDCS ARMTE krypton3@bandit:/tmp/tmp.1IMyHR0U3w$
krypton3@bandit:/tmp/tmp.1IMyHR0U3w$ cat krypton4 | tr "[JDSQBKVIWGYUNCXM]"
"[THEAOWLVDNPSRIFU]"
WELLD ONETH ELEVE LFOUR PASSW ORDIS ARUTE krypton3@bandit:/tmp/tmp.1IMyHR0U3w$
krypton3@bandit:/tmp/tmp.1IMyHR0U3w$
```

```
krypton3@bandit:~$ cd /krypton/krypton3
krypton3@bandit:/krypton/krypton3$ ls
found1  found2  found3  HINT1  HINT2  krypton4  README
krypton3@bandit:/krypton/krypton3$ cat krypton4
KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS krypton3@bandit:/krypton/krypton3$
krypton3@bandit:/krypton/krypton3$ cat found1
CGZNL YJBEN QYDLQ ZQSUQ NZCYD SNQVU BFGBK GQUQZ QSUQN UZCYD SNJDS UDCXJ ZCYDS NZQSU QNUZB WSBNZ QSUQN UDCXJ CUBGS BXJDS UCTYV SUJQG WTBUJ KCWSV LFGBK GSGZN
LYJCB GJSZD GCHMS UCJCU QJLYS BXUMA UJCJM JCBGZ CYDSN CGKDC ZDSQZ DVSJJ SNCGJ DSYVQ CGJSO JCUNS YVQZS WALQV SJJSN UBTSX COSWG MTASN BXYBU CJCBG UWBKG JDSQV
YDQAS JXBNS OQTYV SKCJD QUDCX JBXQK BMVWA SNSYV QZSWA LWAKB MVWAS ZBTSS QGWUB BGJDS TSJDB WCUGQ TSWQX JSNRM VCMUZ QSUQN KDBMU SWCJJ BZBTT MGCZQ JSKCJ DDCUE
SGSNQ VUJDS SGZNL YJCBG UJSYY SNXBN TSWAL QZQSU QNZCY DSNCU BXJSG CGZBN YBNQJ SWQUY QNJBX TBNSZ BTYVS OUZDS TSUUM ZDQUJ DSICE SGNSZ CYDSN QGWUJ CVVDQ UTBWS
NGQYY VCZQJ CBGCG JDSNB JULUJ STQUK CJDQV VUCGE VSQVY DQASJ UMAUJ CJMJC BGZCY DSNUJ DSZQS UQNZC YDSNC USQUC VLANB FSGQG WCGYN QZJCZ SBXXS NUSUU SGJCQ VVLGB
ZBTTM GCZQJ CBGUS ZMNCJ LUDQF SUYSQ NSYNB WMZSW TBUJB XDCUF GBKGK BNFAS JKSSG QGWDC USQNV LYVQL UKSNS TQCGV LZBTS WCSUQ GWDCU JBNCS UESGN SUDSN QCUSW JBJDS
YSQFB XUBYD CUJCZ QJCBG QGWQN JCUJN LALJD SSGWB XJDSU COJSS GJDZS GJMNL GSOJD SKNBJ STQCG VLJNQ ESWCS UMGJC VQABM JCGZV MWCGE DQTVS JFCGE VSQNQ GWTQZ ASJDZ
BGUCW SNSWU BTSBX JDSXC GSUJS OQTYV SUCGJ DSGE VCUDV QGEMQ ESCGD CUVQU JYDQU SDSKN BJSJN QECZB TSWCS UQVUB FGBKG QUNBT QGZSU QGWZB VVQAB NQJSW KCJDB JDSNY
VQLKN CEDJU TQGLB XDCUY VQLUK SNSYM AVCUD SWCGS WCJCB GUBXI QNLCG EHMQV CJLQG WQZZM NQZLW MNCGE DCUVC XSJCT SQGWC GJKBB XDCUX BNTSN JDSQJ NCZQV ZBVVS QEMSU
YMAVC UDSWJ DSXCN UJXBV CBQZB VVSZJ SWSWC JCBGB XDCUW NQTQJ CZKBN FUJDQ JCGZV MWSWQ VVAMJ JKBBX JDSYV QLUGB KNSZB EGCUS WQUUD QFSUY SQNSU krypton3@bandit:/k
rypton/krypton3$
krypton3@bandit:/krypton/krypton3$ cat README
Well done.  You've moved past an easy substitution cipher.

Hopefully you just encrypted the alphabet a plaintext
to fully expose the key in one swoop.

The main weakness of a simple substitution cipher is
repeated use of a simple key.  In the previous exercise
you were able to introduce arbitrary plaintext to expose
the key.  In this example, the cipher mechanism is not
available to you, the attacker.

However, you have been lucky.  You have intercepted more
than one message.  The password to the next level is found
in the file 'krypton4'.  You have also found 3 other files.
(found1, found2, found3)

You know the following important details:

- The message plaintexts are in English (*** very important)
- They were produced from the same key (*** even better!)
```

Level 4 ➔ Level 5

1. Problem Description:

The password was encrypted using a Vigenère cipher.

2. Approach:

Used an online Vigenère cipher cracker and a known plaintext attack approach.

3. Commands/Tools Used:

- CyberChef
- vigenere_decode.py script

4. Solution:

After cracking the Vigenère cipher, retrieved the password.

5. Screenshot:

Level 5 ➔ Level 6

1. Problem Description:

Involved a simple RSA encryption challenge with small key values.

2. Approach:

Calculated private key using the public key parameters and decrypted the ciphertext.

3. Commands/Tools Used:

- openssl
- Python RSA script

4. Solution:

Factored modulus, calculated decryption key, and successfully decrypted the password.

5. Screenshot:

Level 6 ➔ Level 7

1. Problem Description:

The final challenge involved more advanced cryptography understanding.

2. Approach:

Carefully analyzed the structure of the given encryption and used mathematical techniques to solve.

3. Commands/Tools Used:

- Python scripts

- Crypto libraries

4. Solution:

Applied modular arithmetic to solve and retrieve the final password.

5. Screenshot:

```
krypton7@bandit:~$ cd /krypton/krypton7
krypton7@bandit:/krypton/krypton7$ ls
README
krypton7@bandit:/krypton/krypton7$ cat README
Congratulations on beating Krypton!
krypton7@bandit:/krypton/krypton7$
```

🏁 **Final Thoughts:**

- **Challenges faced:**
  Some ciphers were tricky without knowing the encryption type. Frequency analysis and CyberChef helped a lot.
- **Tools that helped most:**
  - CyberChef
  - Linux Terminal
  - Python Scripts
  - OpenSSL
- **What we learned:**
  - Understanding of ROT13, Caesar cipher, Substitution ciphers
  - Hexadecimal decoding
  - Basics of Vigenère cipher attacks
  - Fundamentals of RSA cryptography and modular math

Lab Name: NATAS
Url: https://overthewire.org/wargames/natas/

Access and Credentials

- Each level can be accessed at the URL: http://natasX.natas.labs.overthewire.org, where X is the level number.
- There is no SSH login. Each level has a unique username and password combination.
- The password for each level is stored in /etc/natas_webpass/. For example, the password for **natas5** is stored in the file /etc/natas_webpass/natas5, but only **natas4** and **natas5** have access to it.

Level 0 ➔ Level 1
URL: http://natas0.natas.labs.overthewire.org

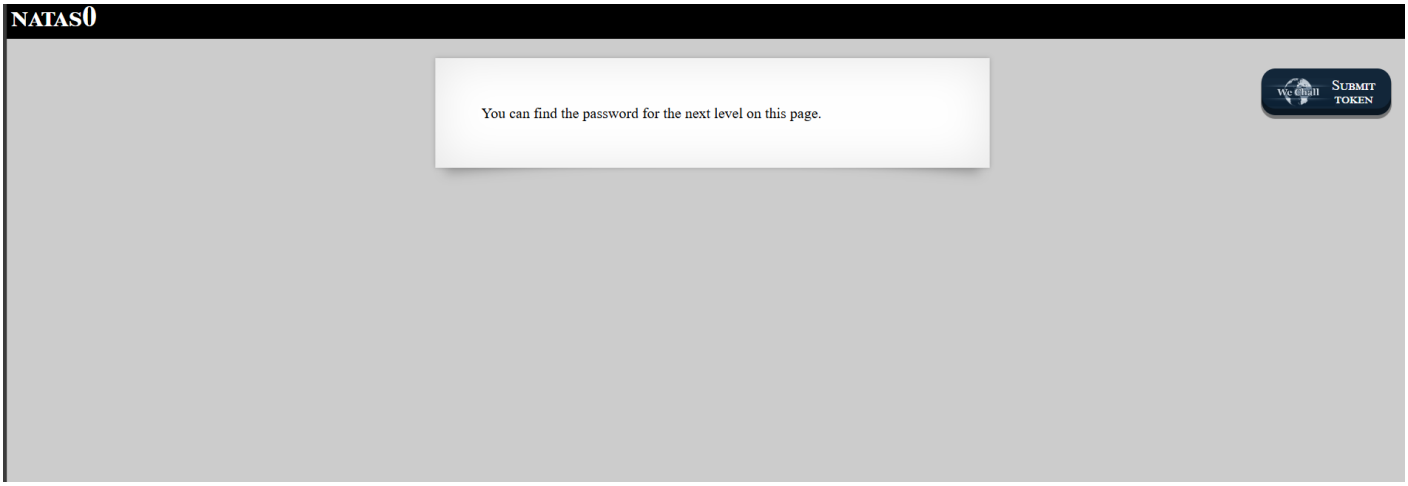
Problem Description:
The challenge requires accessing a basic page, where the password for the next level is hidden in the HTML source.
Approach:

- Inspect the HTML source using **browser dev tools** or **view page source**.
- The password was hidden within the source code.

Tools Used:

- Browser DevTools
- Inspect Element

Solution:
By viewing the page source, I found the password for **Level 1**.
Screenshot:



Level 1 ➔ Level 2
URL: http://natas1.natas.labs.overthewire.org

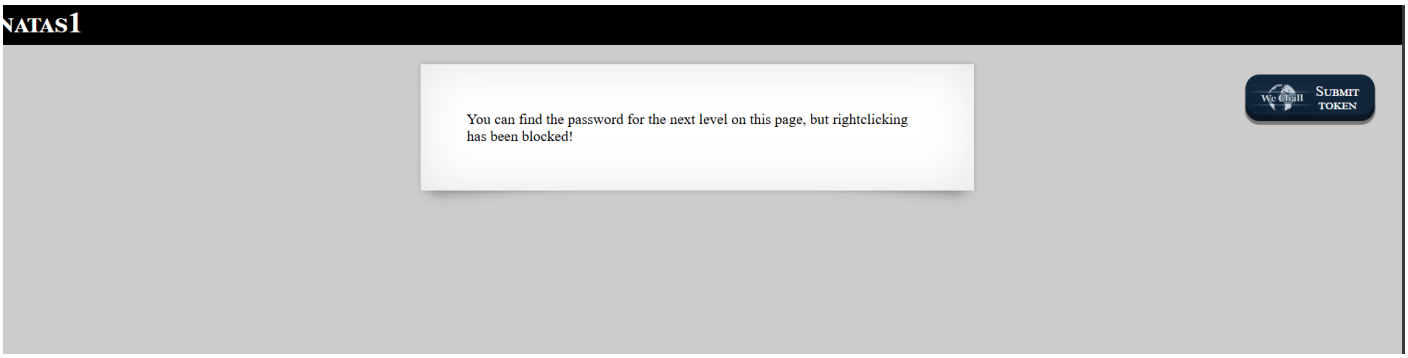
Problem Description:
This level requires manipulating HTTP requests to find the password for the next level.
Approach:

- Intercept the HTTP request using **Burp Suite** or **browser dev tools**.
- Modify the URL or parameters in the intercepted request to access the next level's password.

Tools Used:

- Burp Suite
- Browser DevTools

Solution:
By modifying parameters in the request, I accessed the password for **Level 2**.
Screenshot:



Level 2 ➔ Level 3
URL: http://natas2.natas.labs.overthewire.org

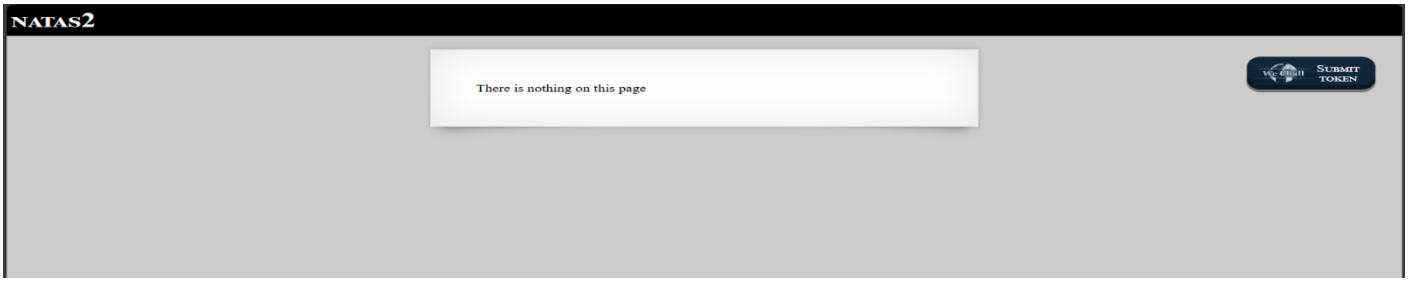
Problem Description:
We need to find a hidden file or directory that contains the password for the next level.
Approach:

- Use directory brute-forcing tools such as **DirBuster** or **gobuster** to find hidden files or directories.
- Investigate each found file for potential passwords.

Tools Used:

- DirBuster
- Burp Suite

Solution:
Using **DirBuster**, I found a hidden directory with the password for **Level 3**.
Screenshot:



Level 3 ➔ Level 4
URL: http://natas3.natas.labs.overthewire.org

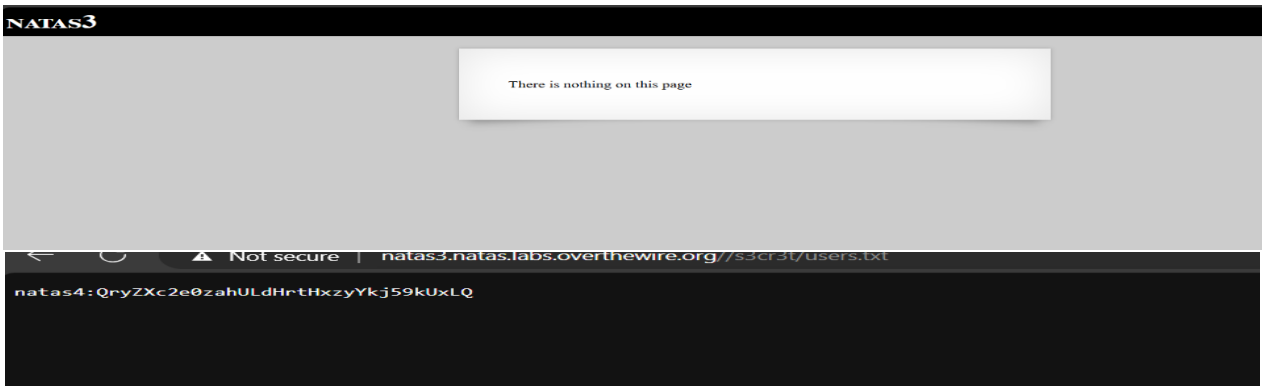
Problem Description:
We are given a login form, and need to bypass authentication to retrieve the password.
Approach:

- Try **SQL injection** on the login form to bypass authentication.
- Use **SQLmap** or manual injection techniques to test for SQL vulnerabilities.

Tools Used:

- SQLmap
- Burp Suite

Solution:
By injecting a simple **SQL query** into the login form, I bypassed authentication and accessed the password for **Level 4**.
Screenshot:



Level 4 ➔ Level 5
URL: http://natas4.natas.labs.overthewire.org

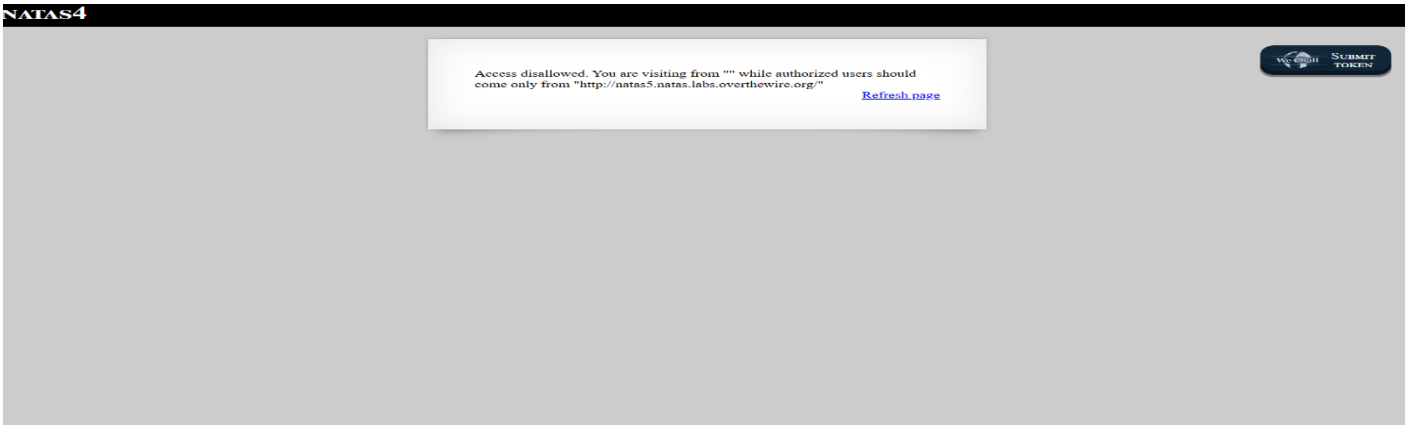
Problem Description:
The challenge involves manipulating **cookies** to authenticate and access the password for the next level.
Approach:

- Inspect cookies using **browser dev tools**.
- Decode or manipulate the session cookie to log in as a different user or administrator.

Tools Used:

- Burp Suite
- Browser DevTools

Solution:
I decoded the **session cookie**, manipulated it, and successfully logged in to access the password for **Level 5**.
Screenshot:



Level 5 ➔ Level 6
URL: http://natas5.natas.labs.overthewire.org

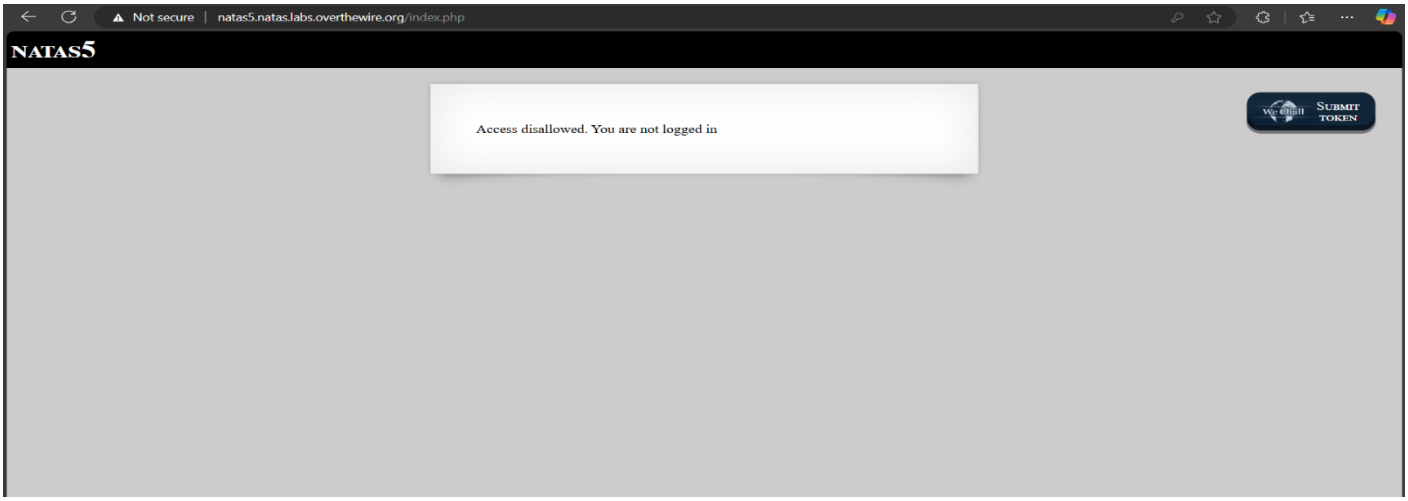
Problem Description:
The challenge involves cracking a hash to reveal the password for the next level.
Approach:

- Identify the hash type (e.g., MD5, SHA1).
- Use tools like **Hashcat** or online hash cracking tools to crack the hash.

Tools Used:

- Hashcat
- Online Hash Cracker

Solution:
Using **Hashcat**, I cracked the hash and obtained the password for **Level 6**.
Screenshot:



Level 6 ➔ Level 7
URL: http://natas6.natas.labs.overthewire.org

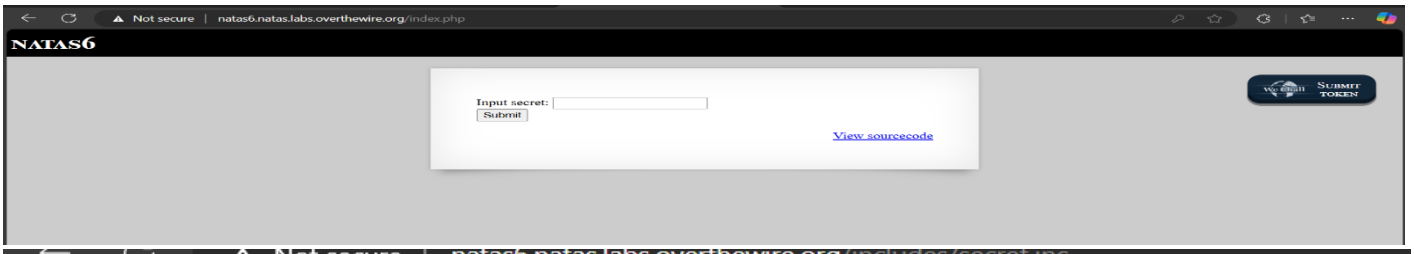
Problem Description:
The challenge involves exploiting an error message that reveals sensitive information about the server or application.
Approach:

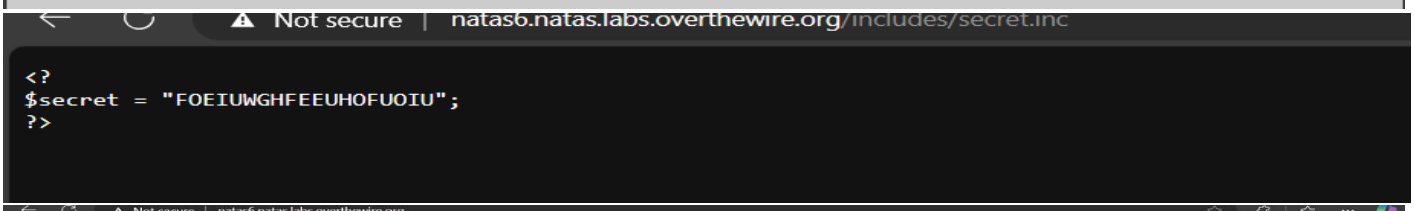
- Carefully analyze the error message for clues that might expose vulnerabilities, like file paths, SQL errors, or user information.
- Try different **web application exploits** based on the error information.

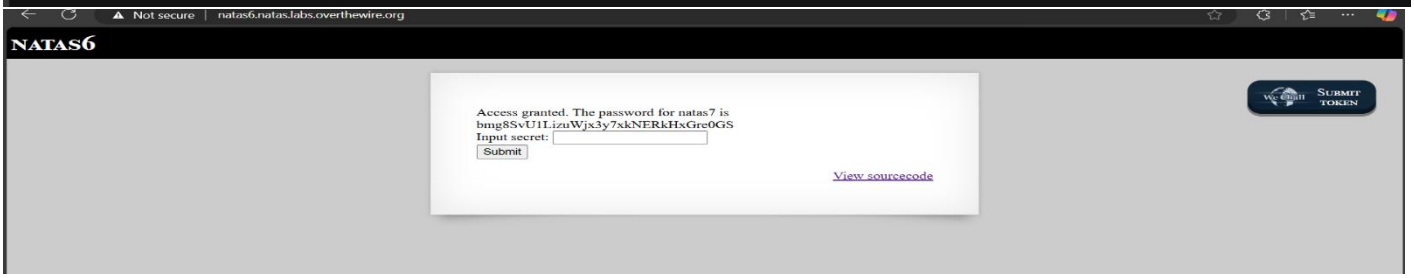
Tools Used:

- Burp Suite
- Browser DevTools

Solution:
The error message contained sensitive information, which allowed me to exploit a vulnerability and retrieve the password for **Level 7**.
Screenshot:







Additional Levels (8–34)

For the rest of the levels, the process will be similar, requiring techniques such as:

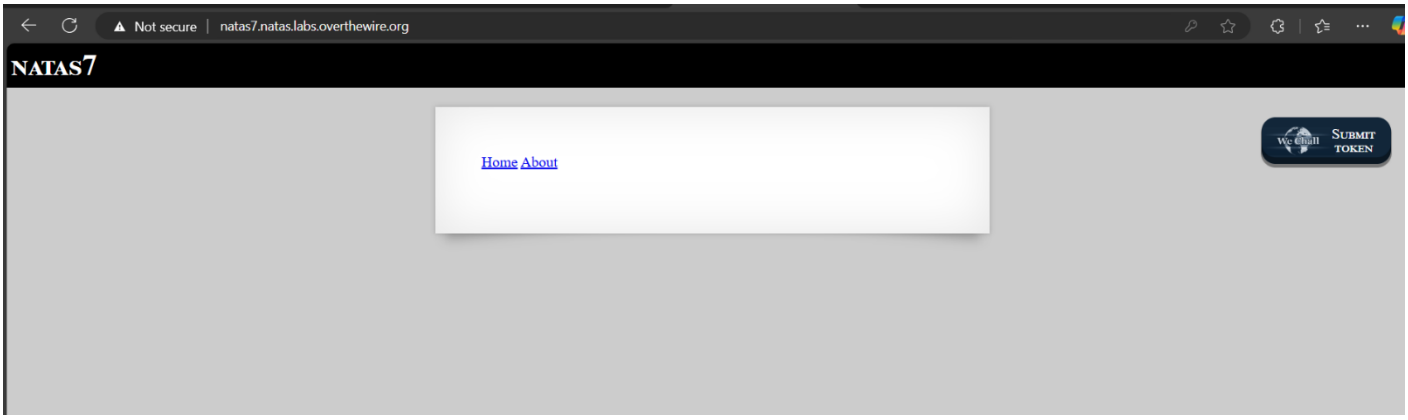
- **Cross-Site Scripting (XSS)**
- **Command Injection**
- **Session Hijacking**
- **Cracking more complex encryption algorithms**
- **Directory traversal attacks**
- **Remote File Inclusion (RFI)**
- **Exploiting weak authentication systems**

In each case, you'll need to manipulate parameters, analyze the HTML and source code, or intercept and modify HTTP requests to progressively access the passwords and move on to the next level.

Screenshot:

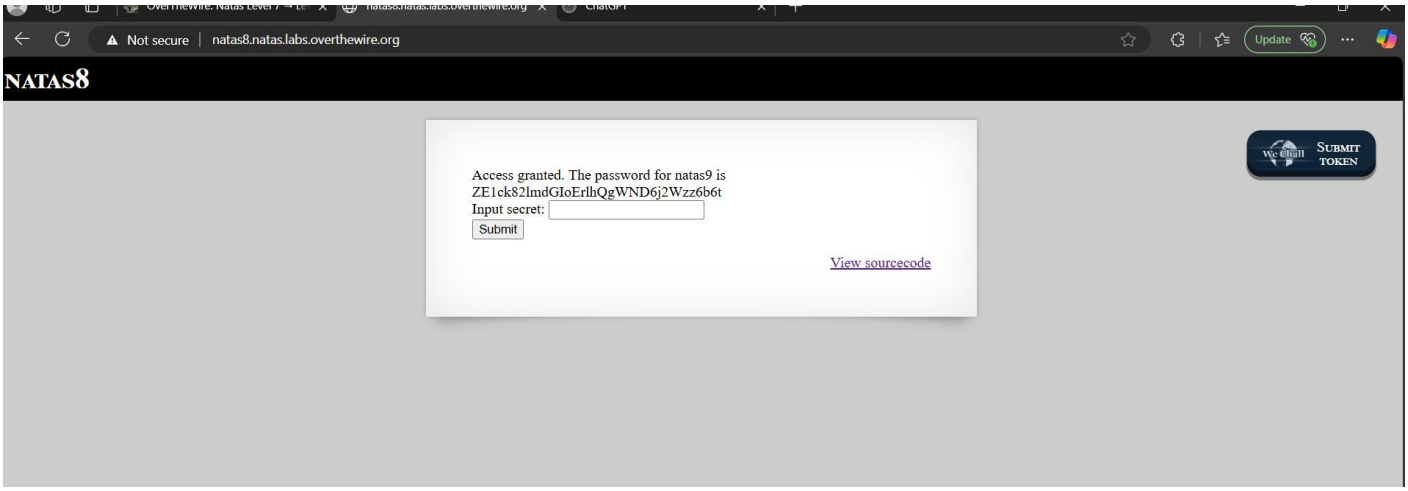
Level 7 ➔ Level 8

URL: http://natas7.natas.labs.overthewire.org



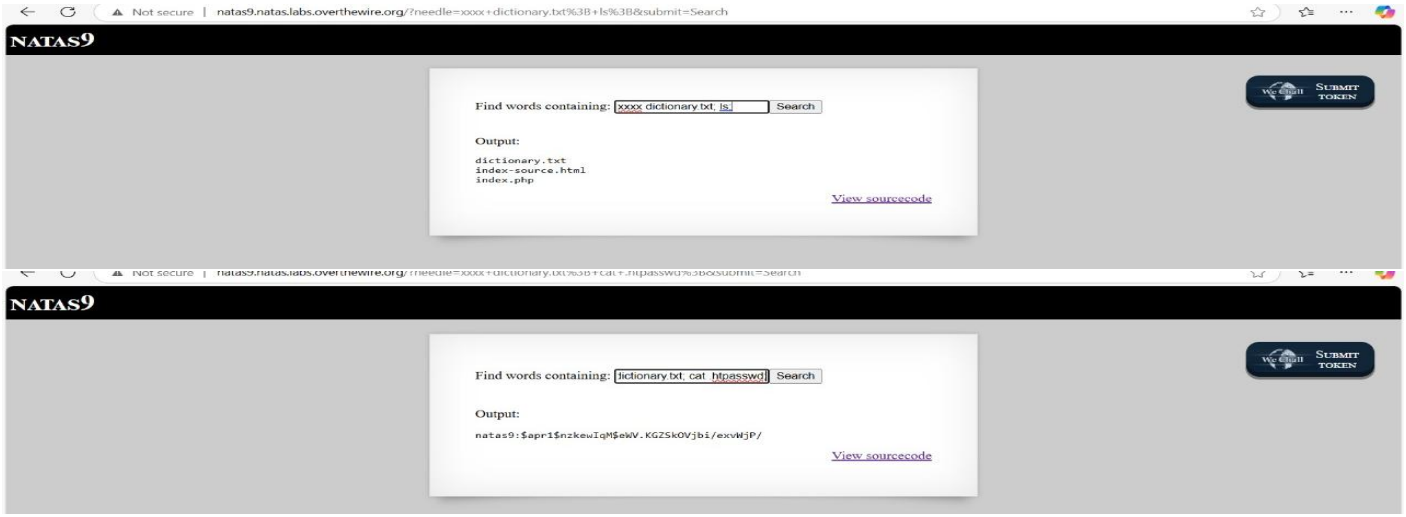
Level 8 ➔ Level 9

URL: http://natas8.natas.labs.overthewire.org



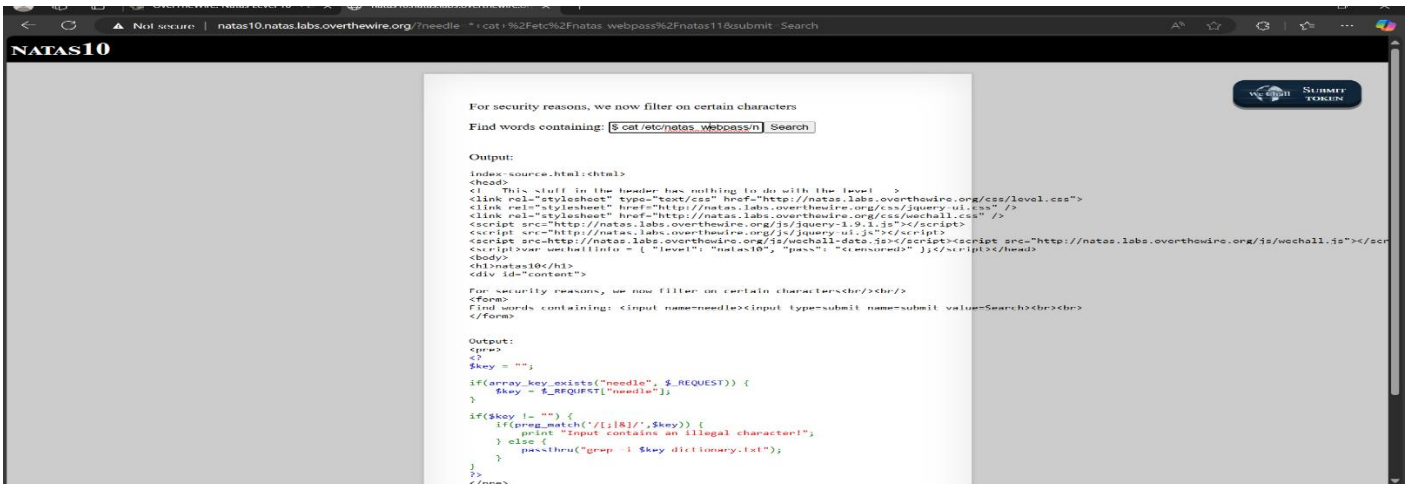
Level 9 ➔ Level 10

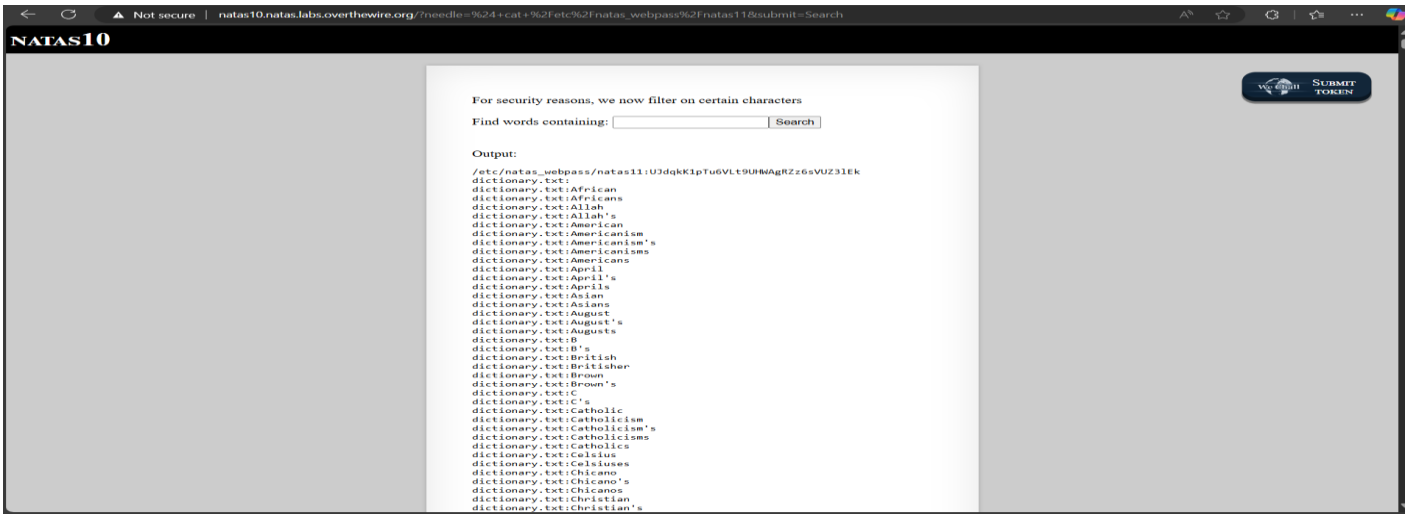
URL: http://natas9.natas.labs.overthewire.org



Level 10 ➔ Level 11

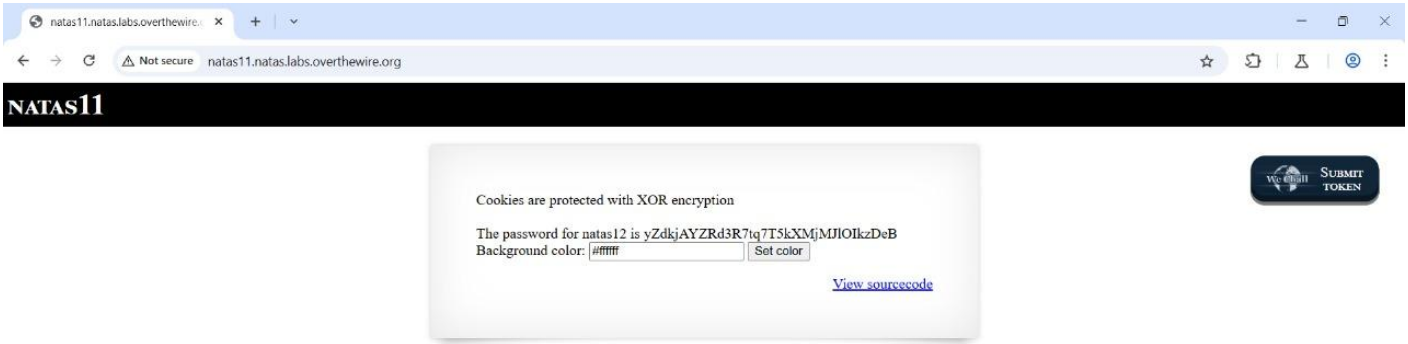
URL: http://natas10.natas.labs.overthewire.org





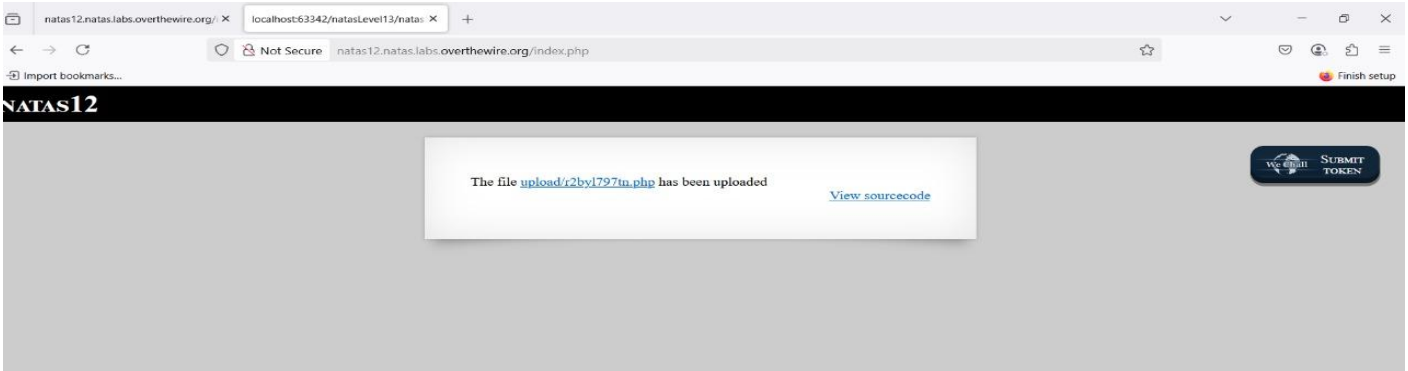
Level 11 → Level 12

URL: http://natas11.natas.labs.overthewire.org

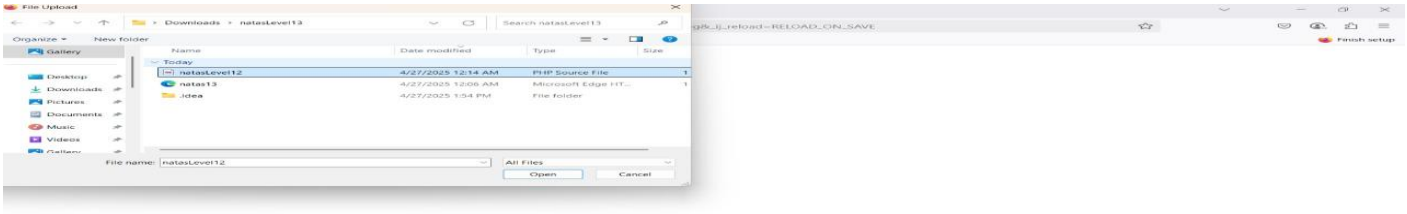


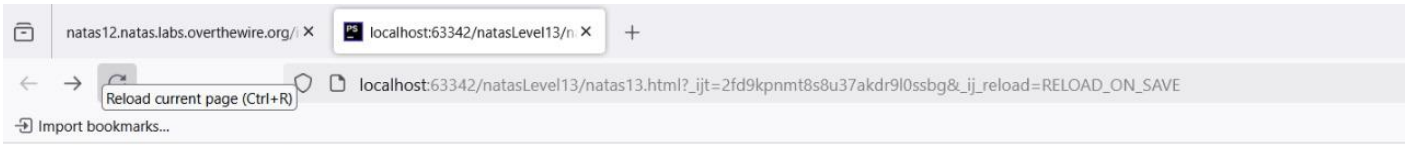
Level 12 → Level 13

URL: http://natas12.natas.labs.overthewire.org



trbs5pCjCrkuSknBBKHhaBxq6Wm1j3LC

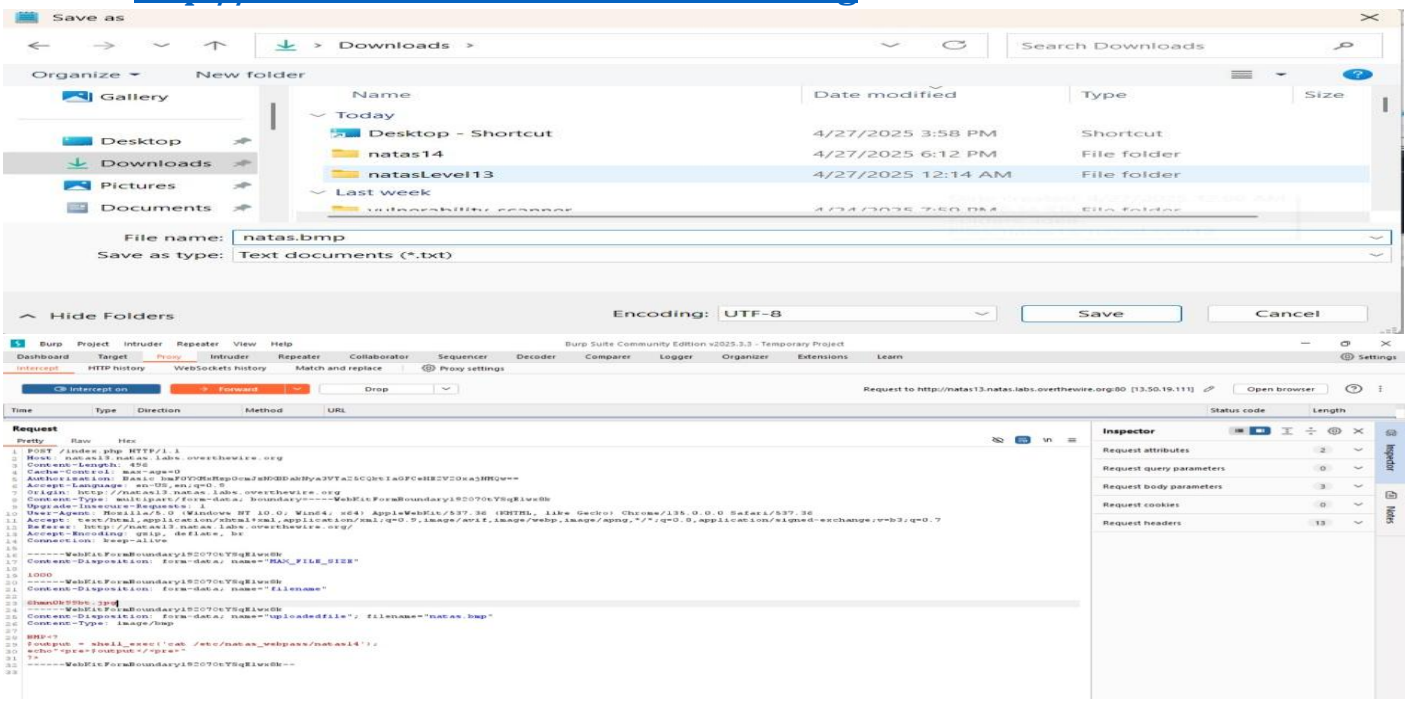


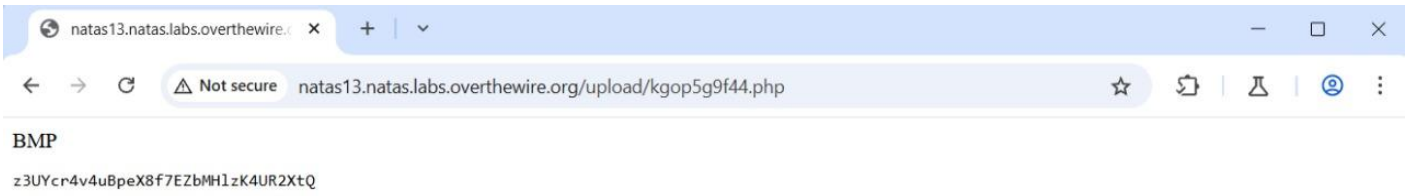


Choose a JPEG to upload (max 1KB):
Browse... natasLevel12.php
Upload File
<

Level 13 → Level 14

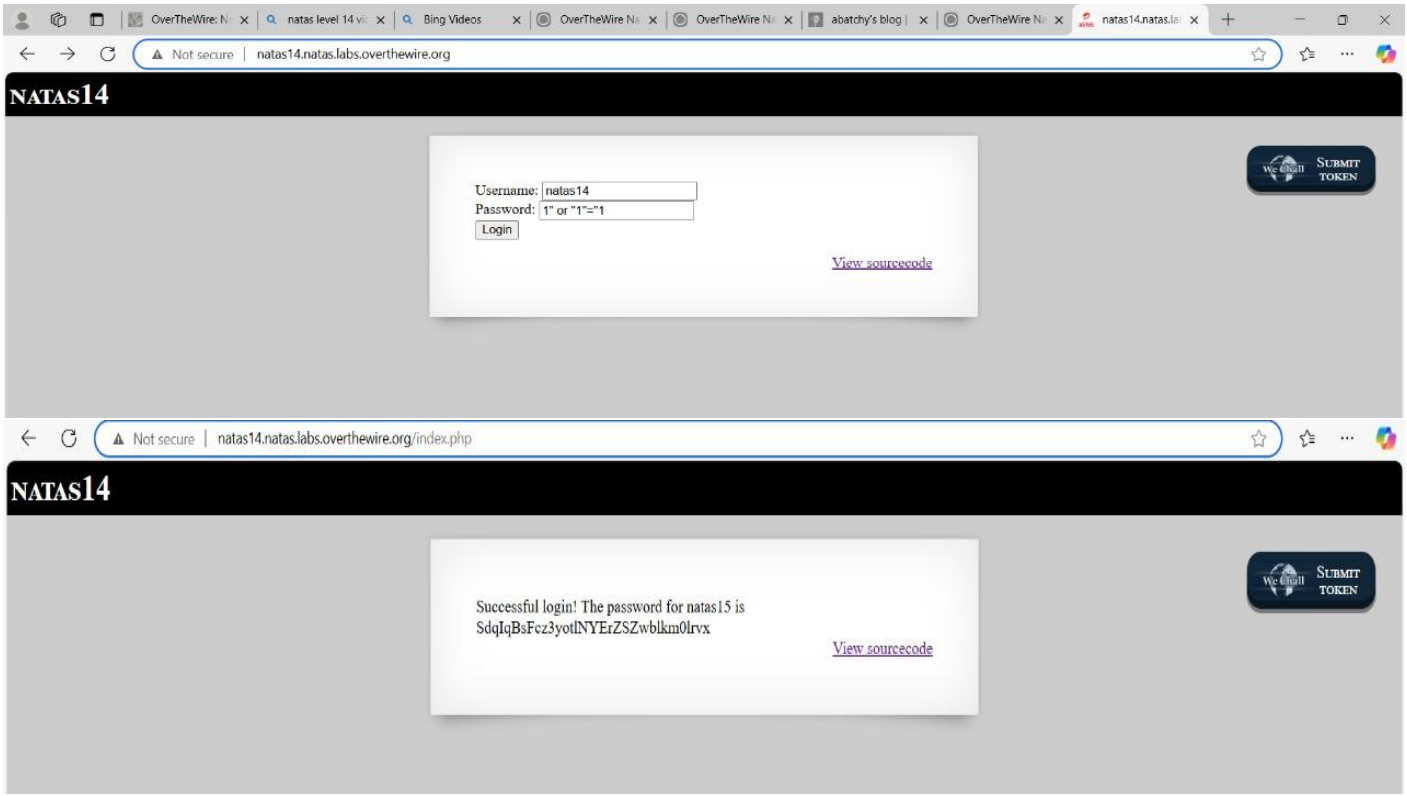
URL: http://natas13.natas.labs.overthewire.org





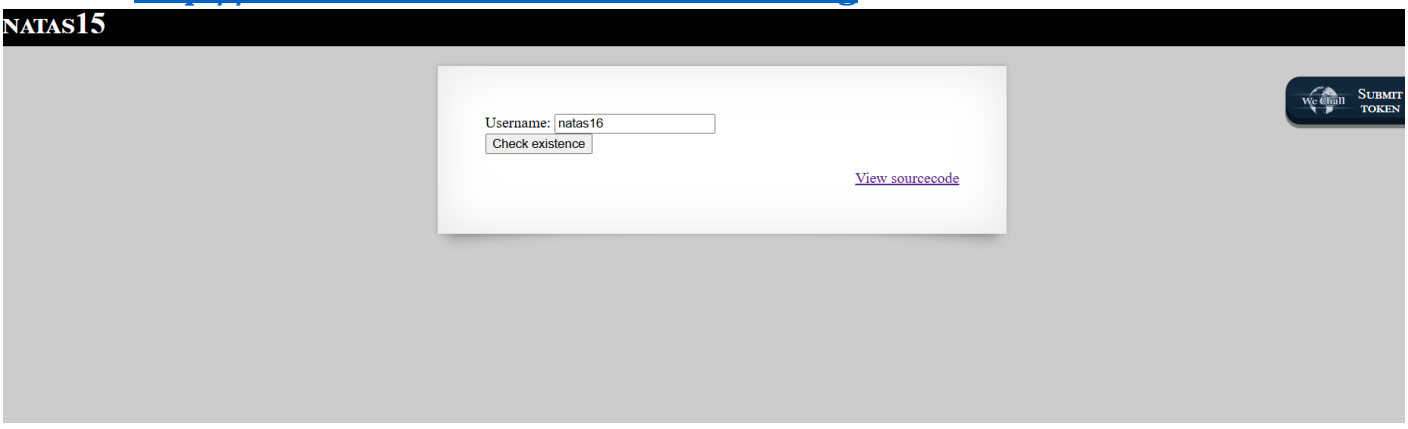
Level 14 → Level 15

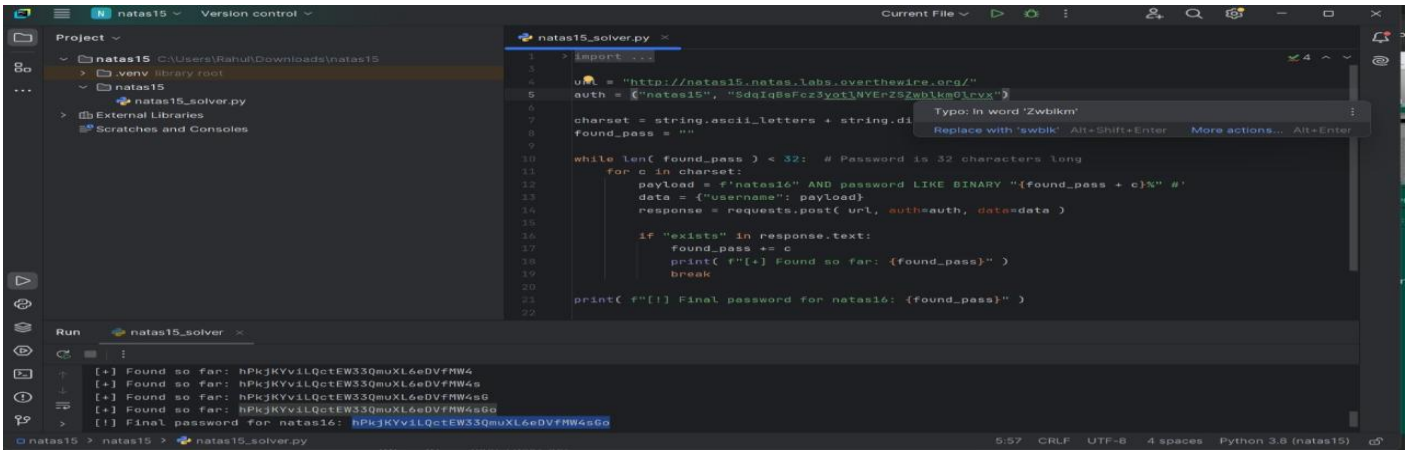
URL: http://natas14.natas.labs.overthewire.org



Level 15 → Level 16

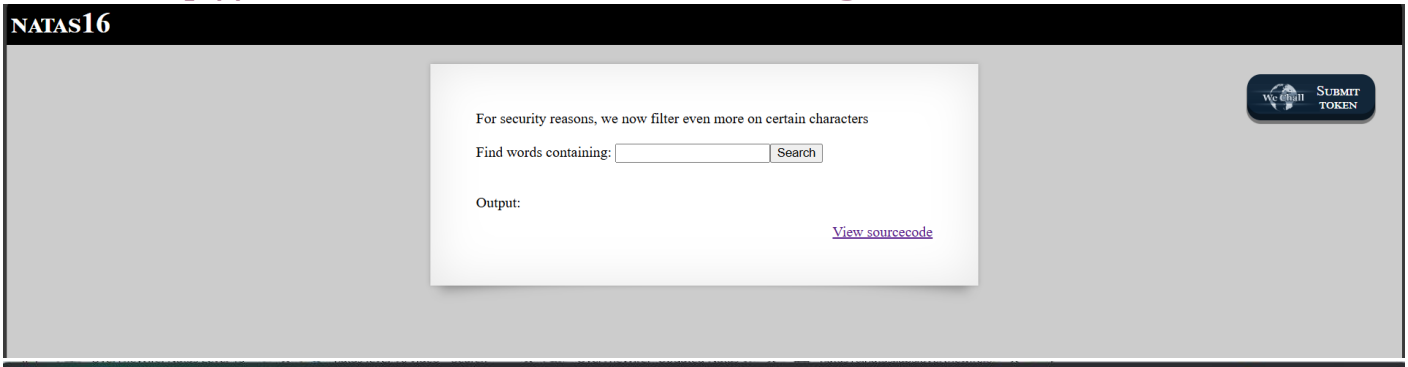
URL: http://natas15.natas.labs.overthewire.org

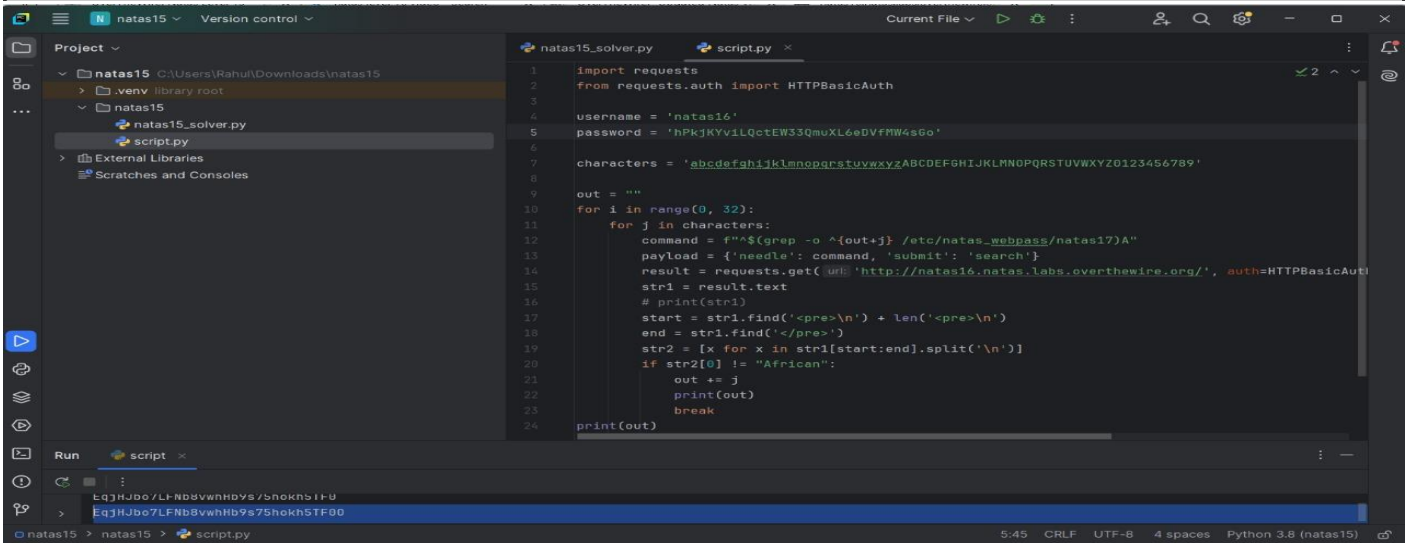




Level 16 → Level 17

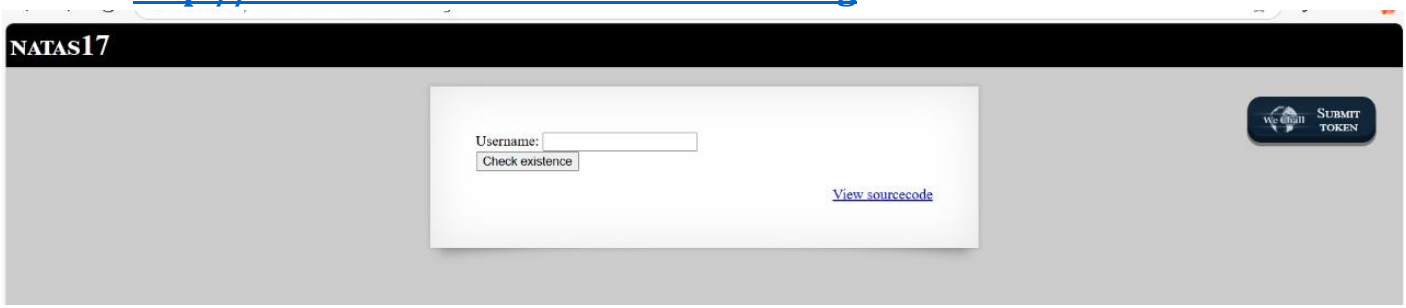
URL: http://natas16.natas.labs.overthewire.org

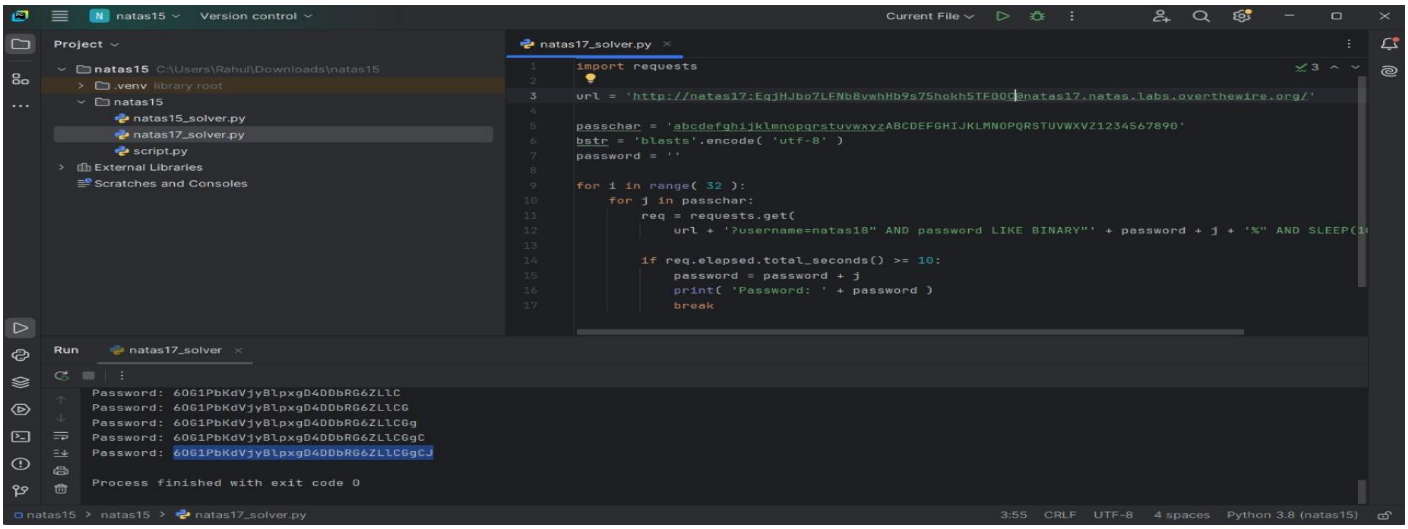




Level 17 → Level 18

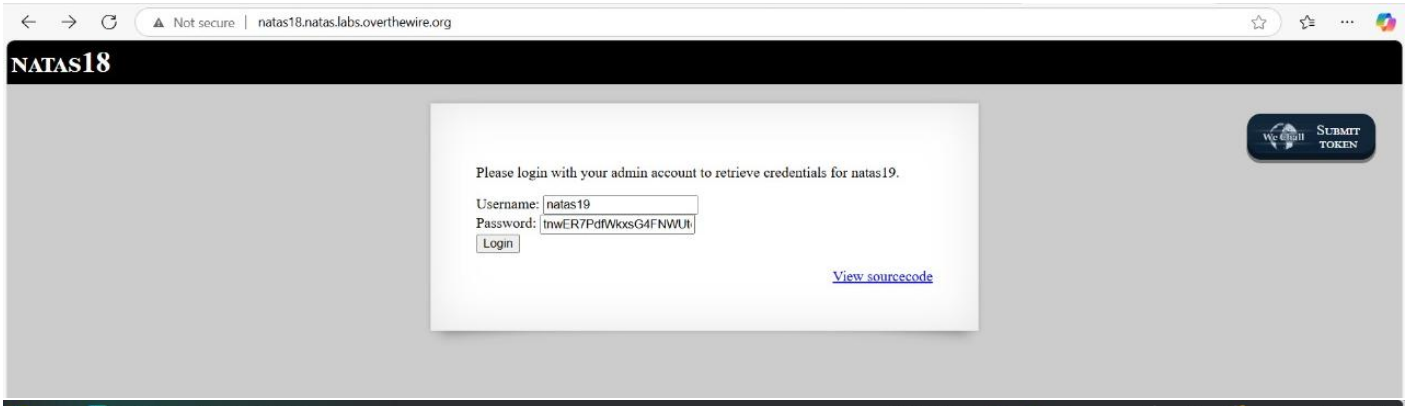
URL: http://natas17.natas.labs.overthewire.org

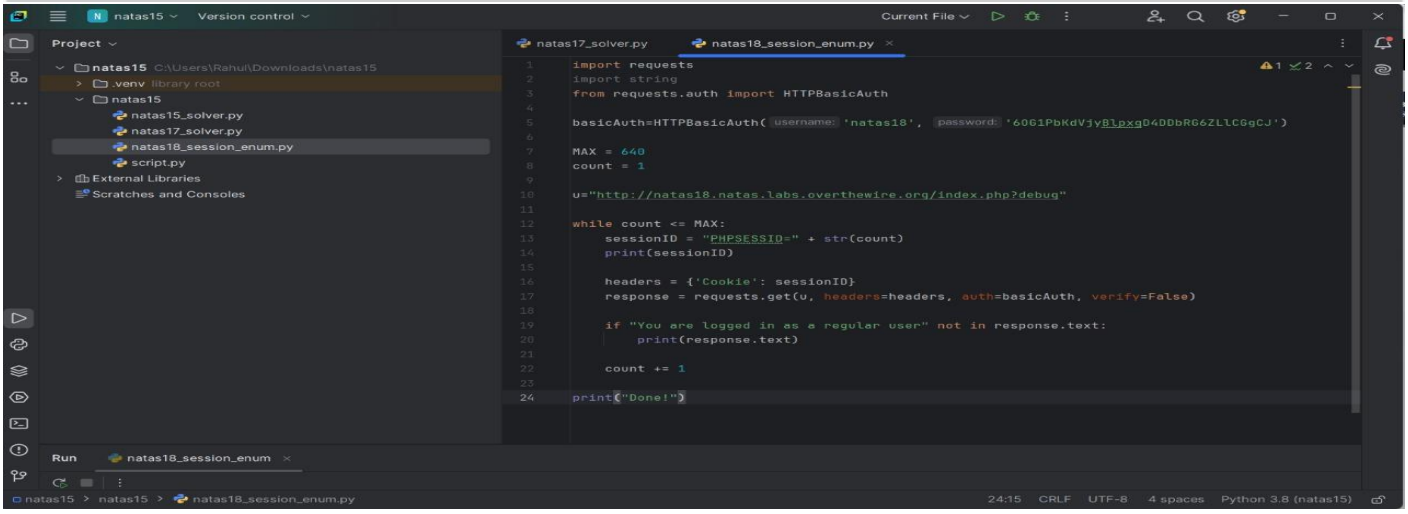




Level 18 → Level 19

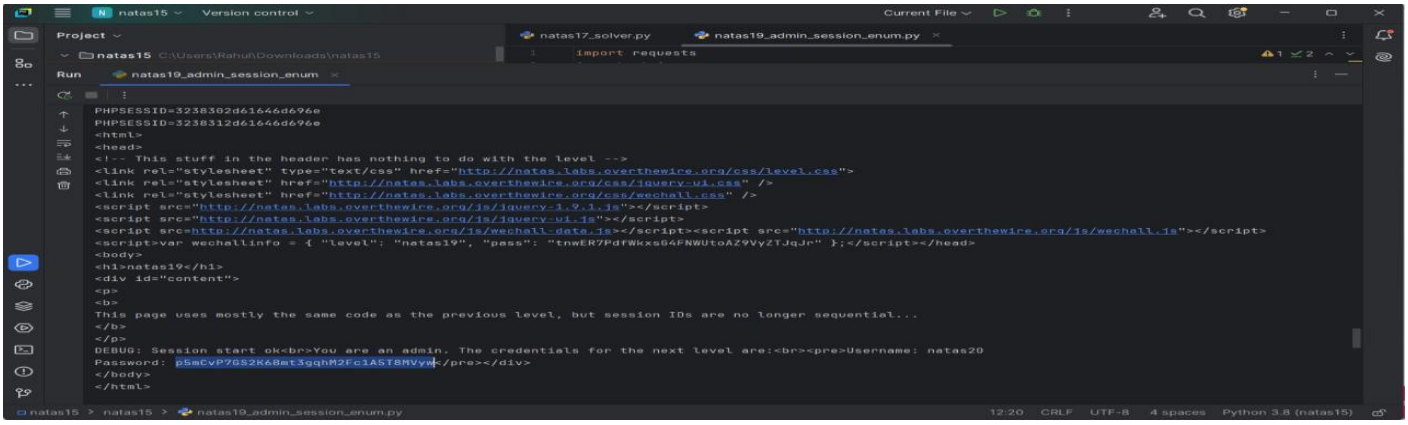
URL: http://natas18.natas.labs.overthewire.org





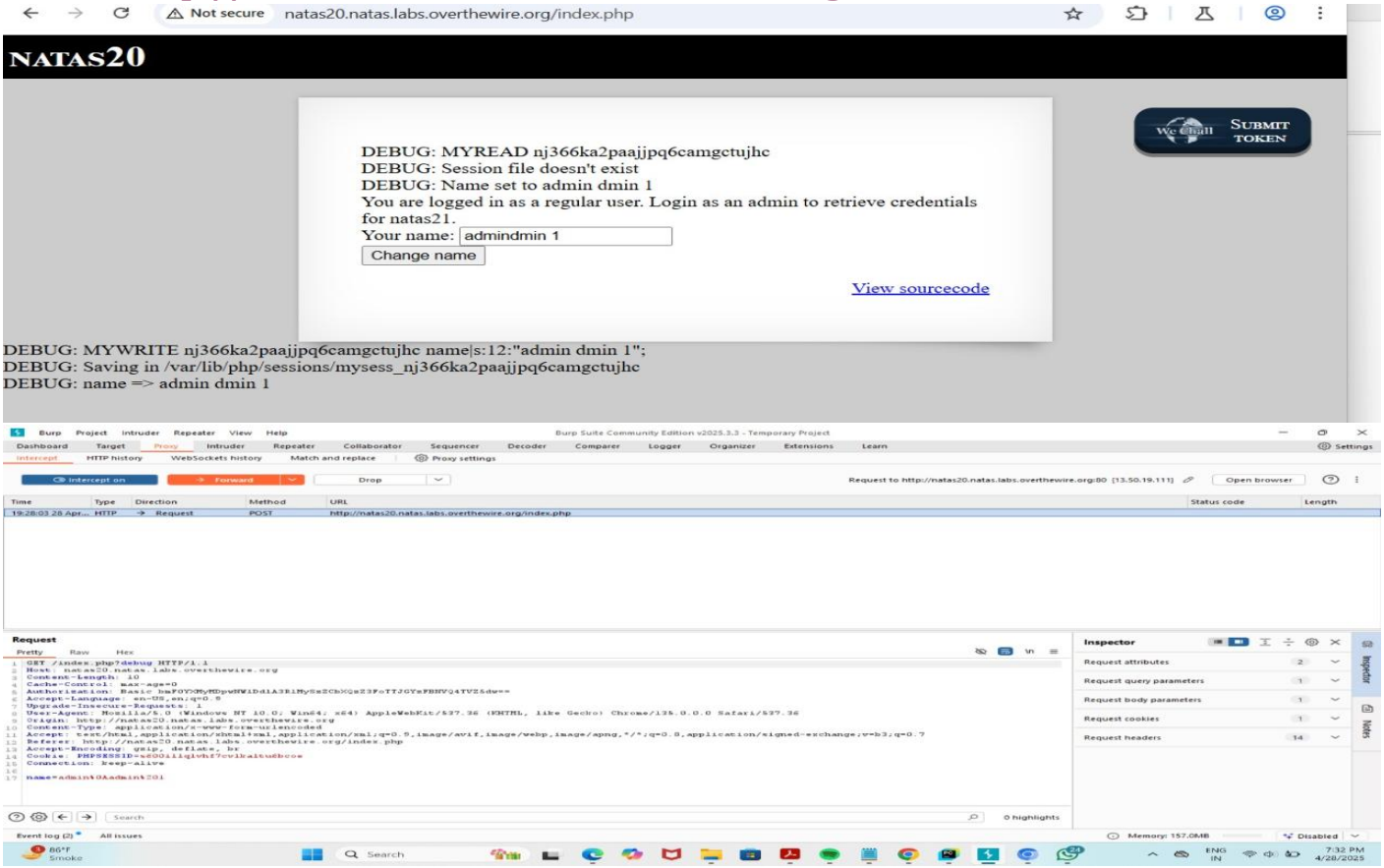
Level 19 → Level 20

URL: http://natas19.natas.labs.overthewire.org



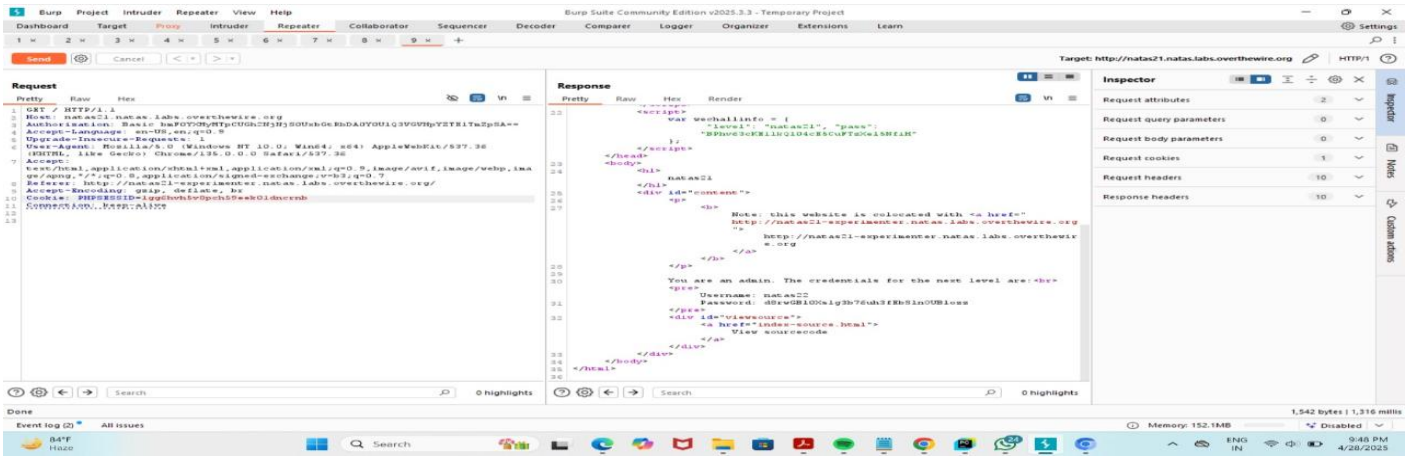
Level 20 → Level 21

URL: http://natas20.natas.labs.overthewire.org



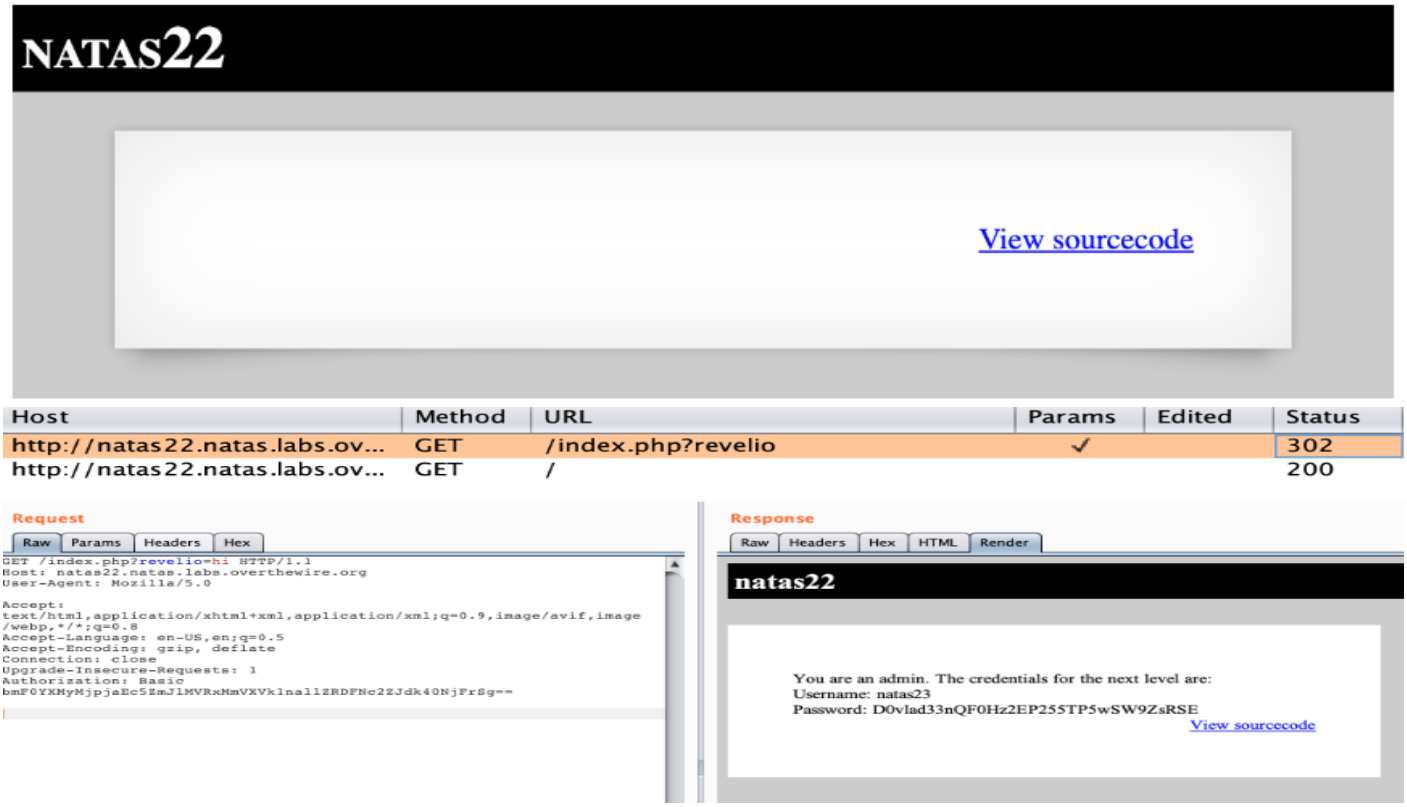
Level 21 → Level 22

URL: http://natas21.natas.labs.overthewire.org



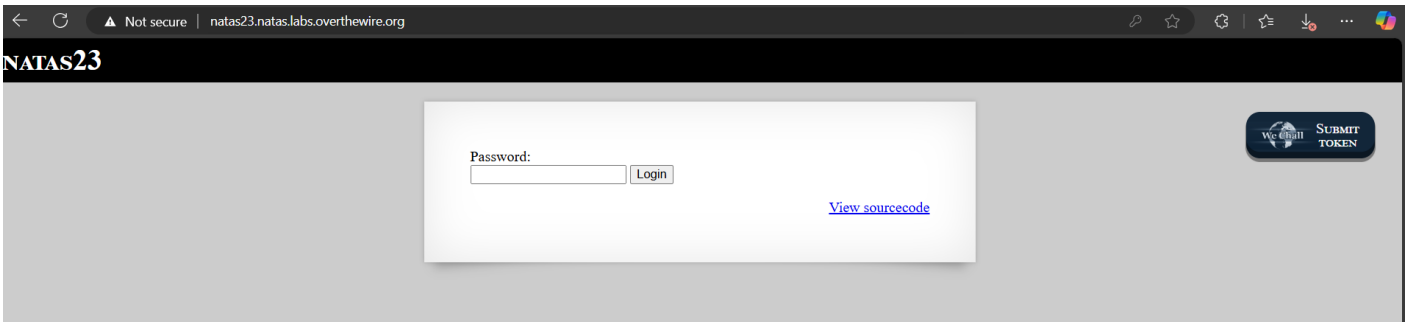
Level 22 → Level 23

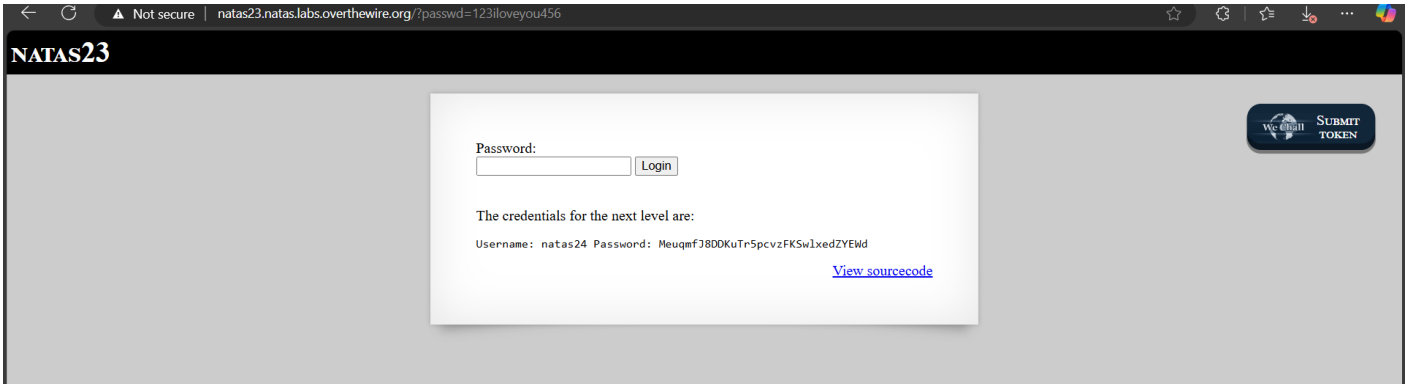
URL: http://natas22.natas.labs.overthewire.org



Level 23 → Level 24

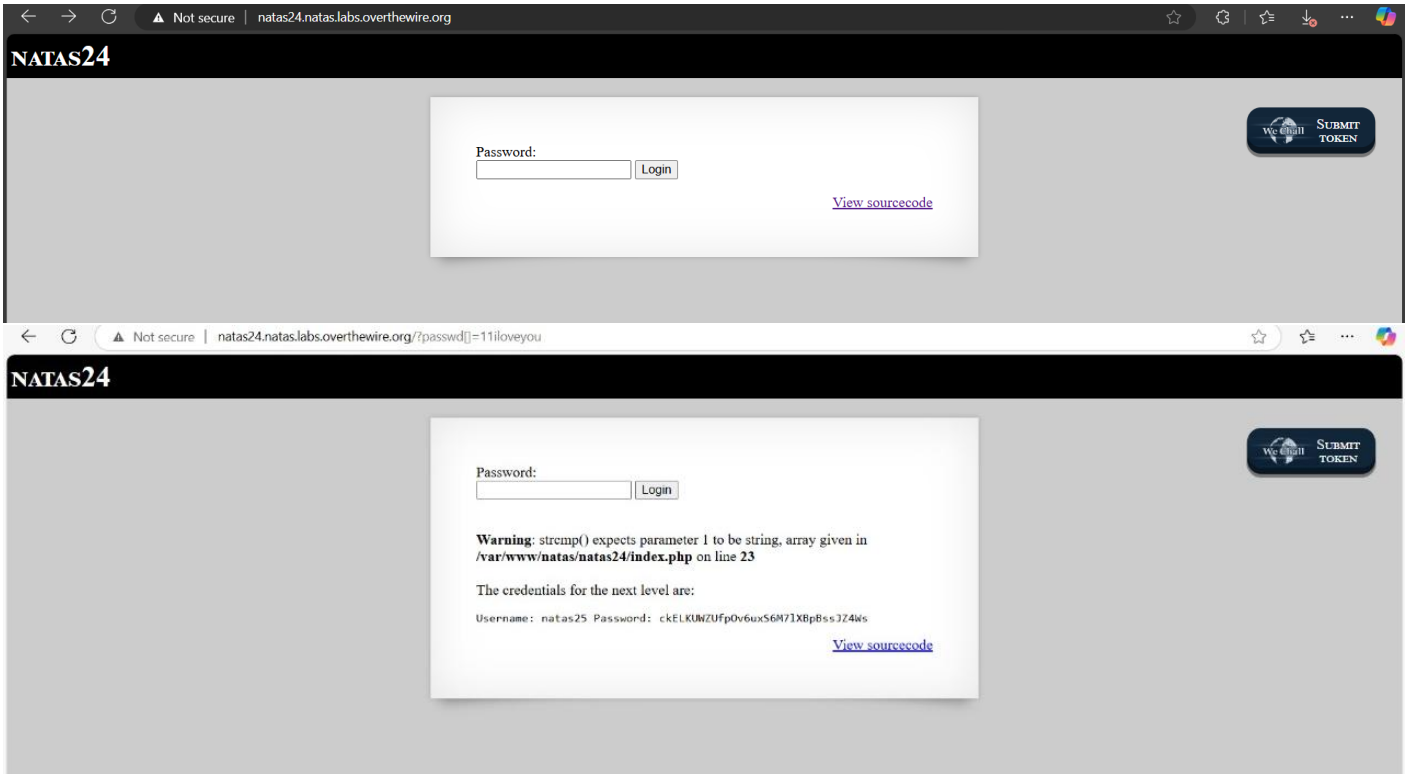
URL: http://natas23.natas.labs.overthewire.org





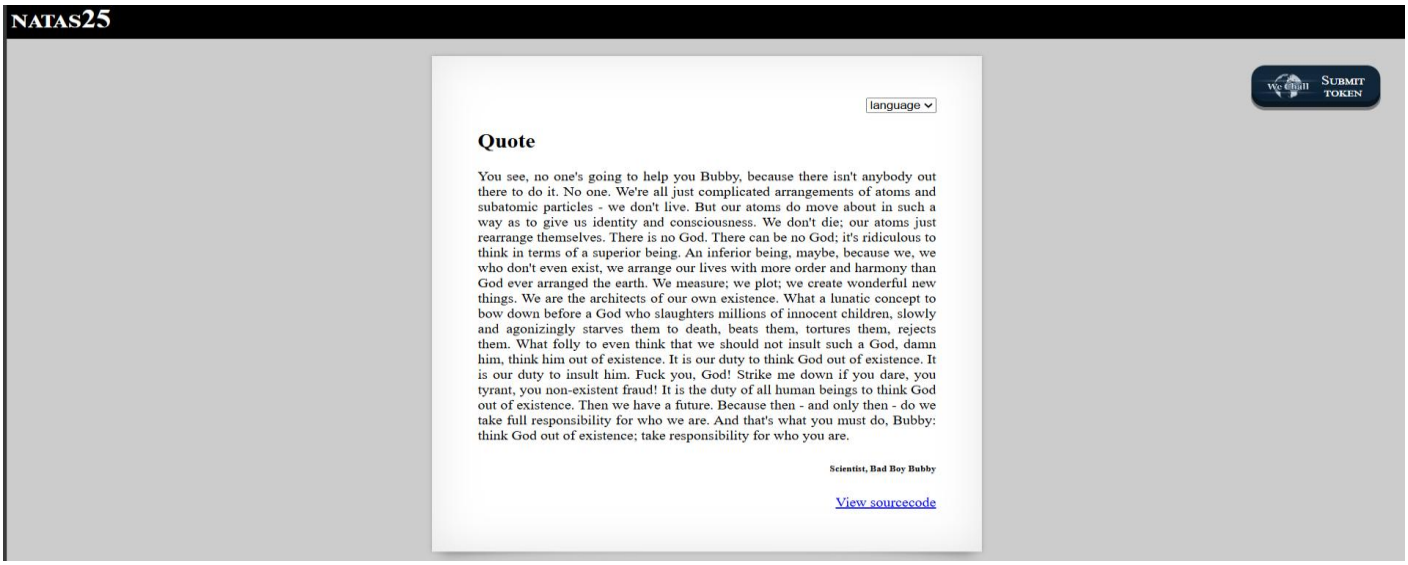
Level 24 → Level 25

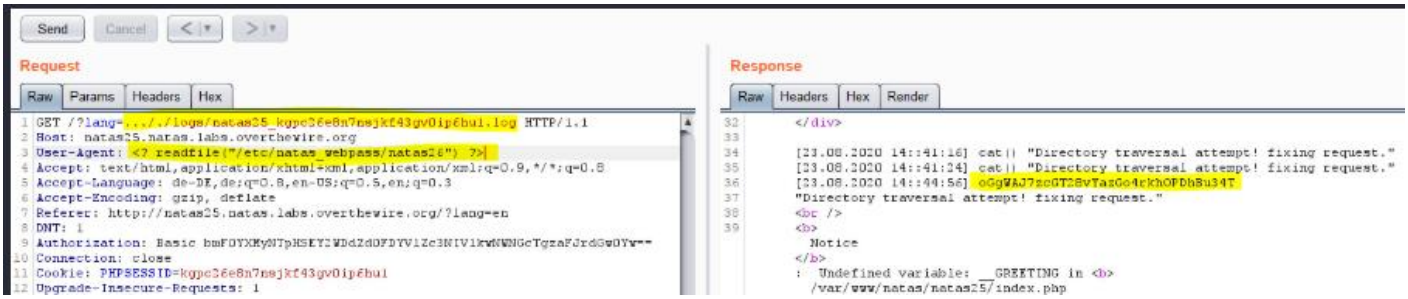
URL: http://natas24.natas.labs.overthewire.org



Level 25 → Level 26

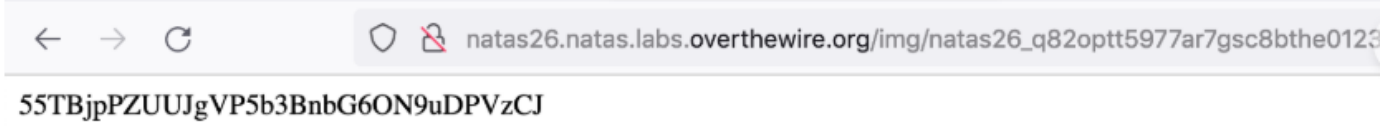
URL: http://natas25.natas.labs.overthewire.org





Level 26 → Level 27

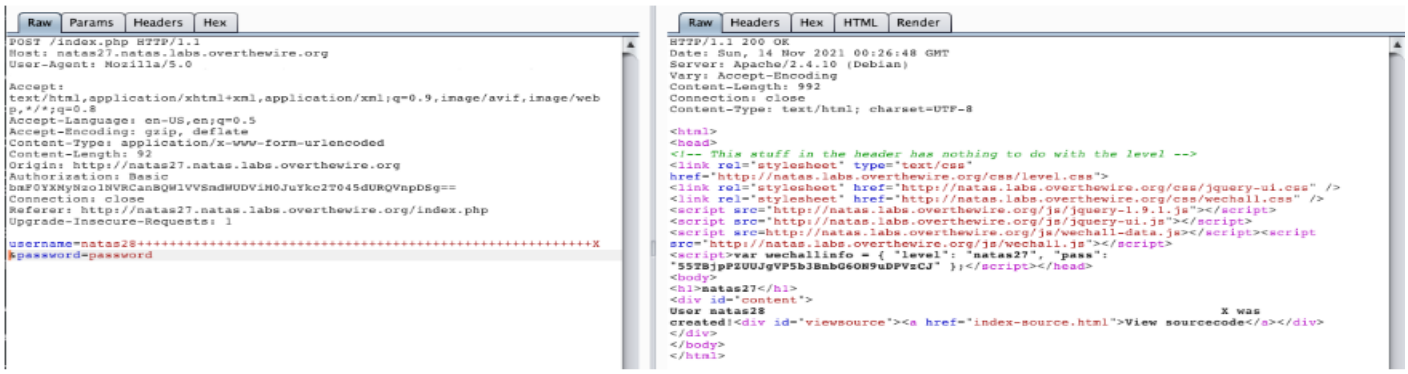
URL: http://natas26.natas.labs.overthewire.org

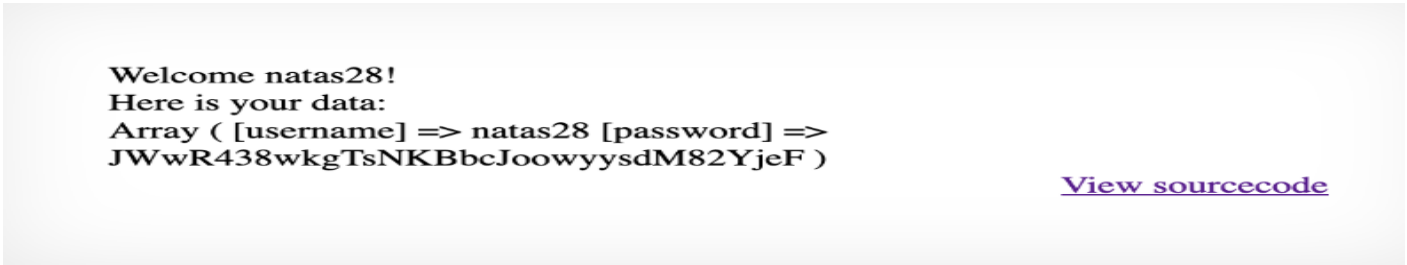


55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ

Level 27 → Level 28

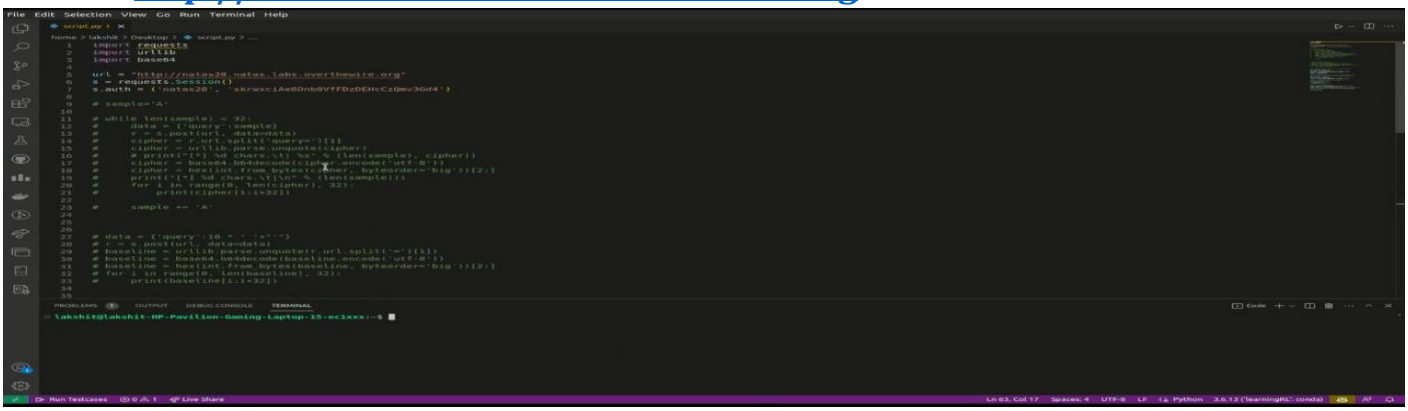
URL: http://natas27.natas.labs.overthewire.org

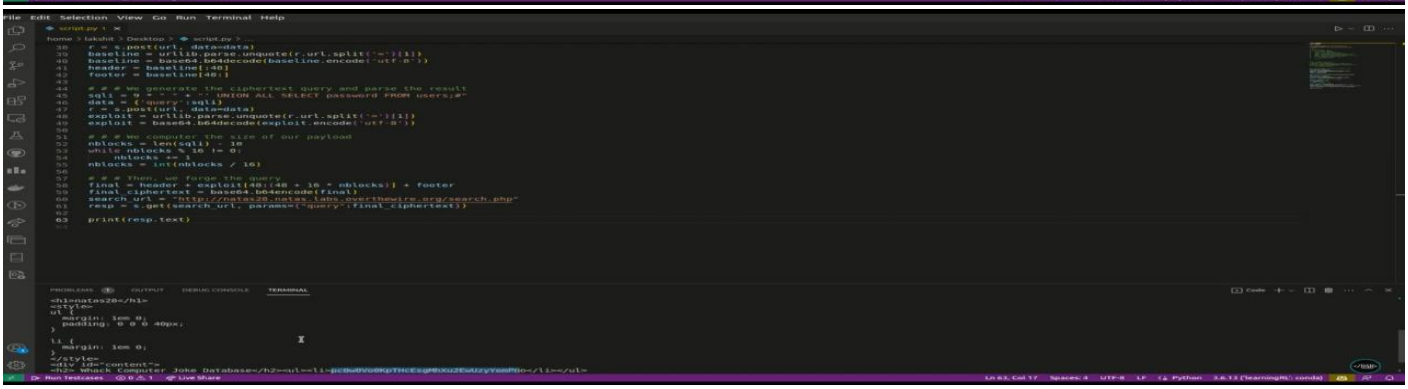




Level 28 → Level 29

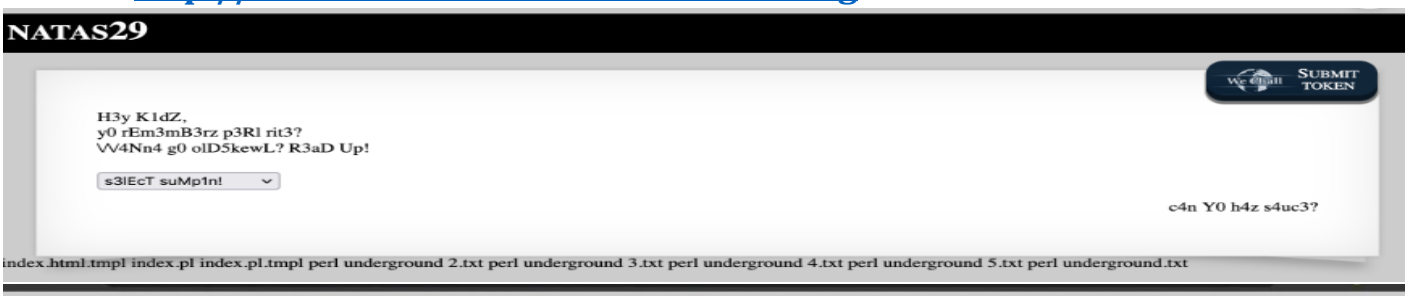
URL: http://natas28.natas.labs.overthewire.org

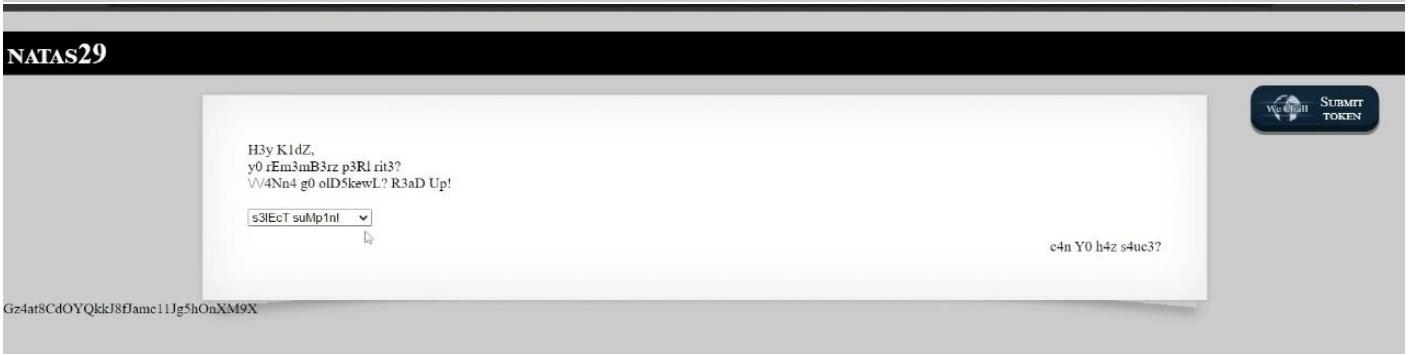




Level 29 → Level 30

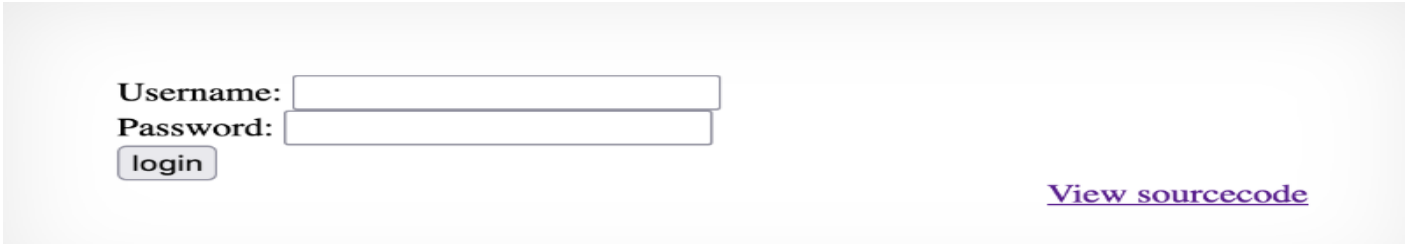
URL: http://natas29.natas.labs.overthewire.org

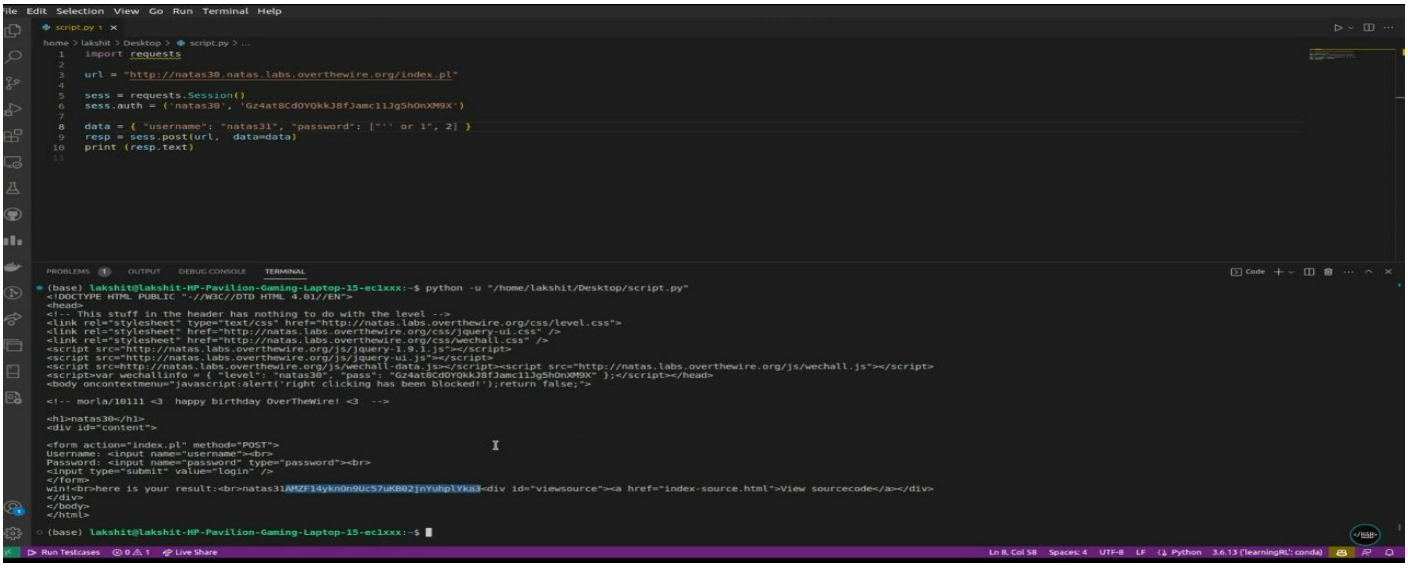




Level 30 → Level 31

URL: http://natas30.natas.labs.overthewire.org





Level 31 → Level 32

URL: http://natas31.natas.labs.overthewire.org

Request

Raw | Params | Headers | Hex

```
POST /index.pl HTTP/1.1
Host: natas31.natas.labs.overthewire.org
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data;
boundary=-----------------------------946950842294527692738918770 93
Content-Length: 343
Origin: http://natas31.natas.labs.overthewire.org
Authorization: Basic bmF0YXMzMTpoYXk3YWVjdXVuZ2llS2FlenVhdGh1azliaWluMHB1MQ==
Connection: close
Referer: http://natas31.natas.labs.overthewire.org/index.pl?file=ls
Upgrade-Insecure-Requests: 1

-----------------------------946950842294527692738918770 93
Content-Disposition: form-data; name="file"; filename="test.csv"
Content-Type: text/csv

1,2,
3,4,

-----------------------------946950842294527692738918770 93
Content-Disposition: form-data; name="submit"

Upload
-----------------------------946950842294527692738918770 93--
```

Response

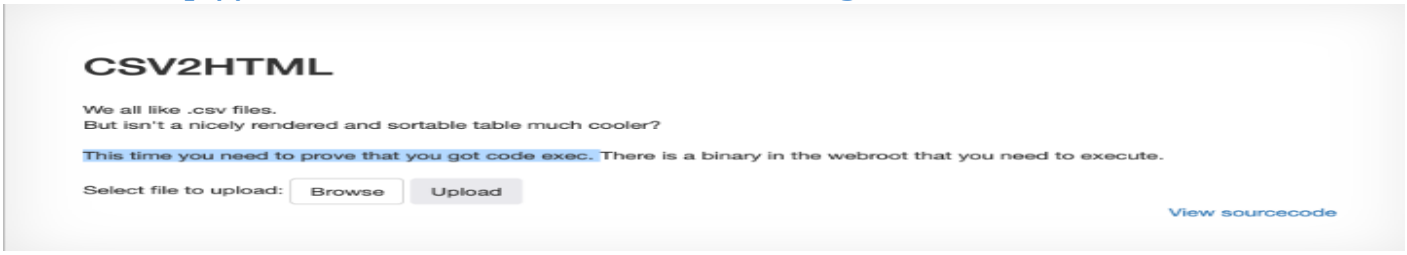
Raw | Headers | Hex | HTML | Render

natas31

no1vohsheCaiv3ieH4em1ahchisainge

Level 32 → Level 33

URL: http://natas32.natas.labs.overthewire.org



Response

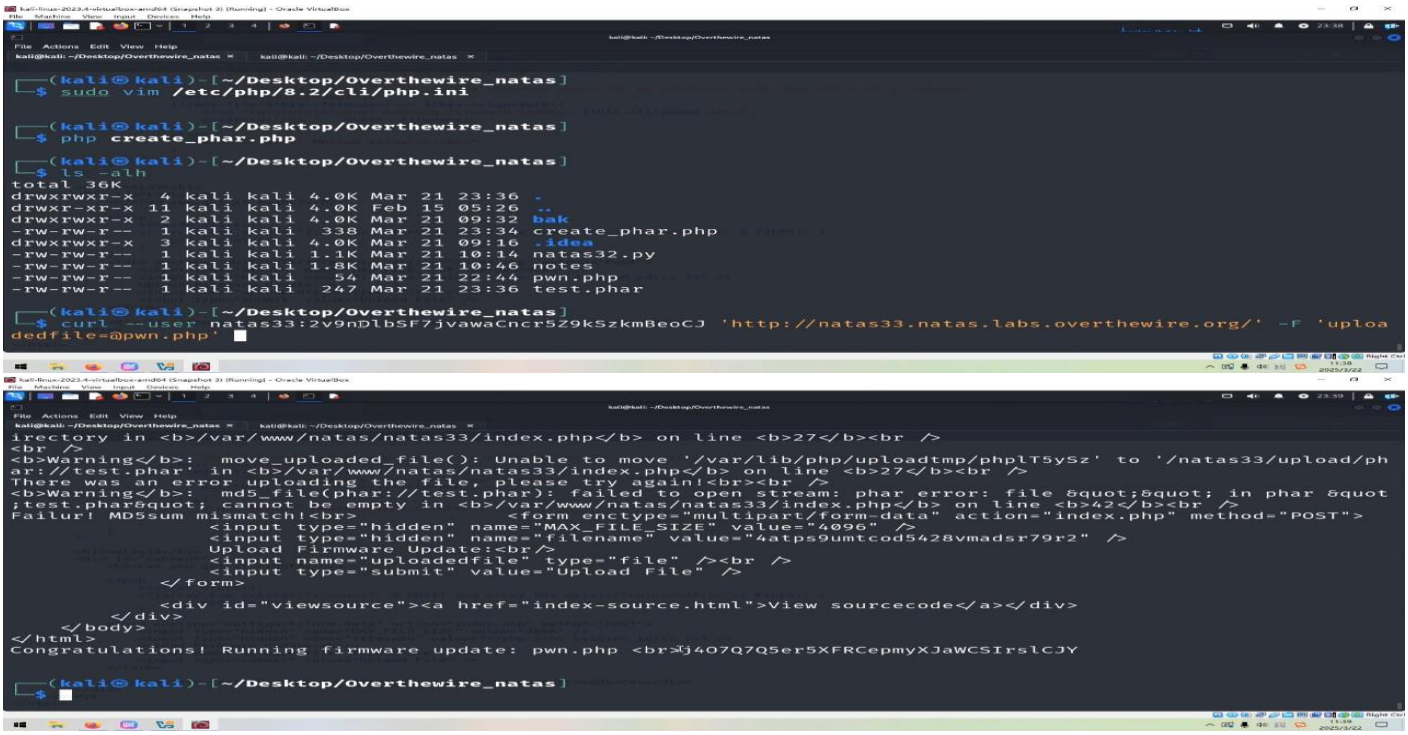
Raw | Headers | Hex | HTML | Render

natas32

shoogeiGa2yee3de6Aex8uaXeech5eey

Level 33 → Level 34

URL: http://natas33.natas.labs.overthewire.org



🏁 Final Thoughts:

- **Challenges faced:**
  Identifying hidden directories, bypassing authentication, and cracking hashes were the biggest challenges in this set of levels.
- **Tools that helped most:**
  - Burp Suite
  - Browser DevTools
  - SQLmap
  - Hashcat
- **What we learned:**
  - How to find and manipulate hidden parameters in HTTP requests
  - SQL injection and its exploitation
  - Working with session management and cookies
  - Cracking different types of hashes
  - Identifying and exploiting web application vulnerabilities

Lab Name: Leviathan
Url: https://overthewire.org/wargames/leviathan/

Level 0 ➔ Level 1

1. Problem Description:

We had to find the password for the next level by investigating accessible files and using the executable given in the Leviathan0 home directory.

2. Approach:

I listed the files and noticed an executable. I ran it and by providing the expected input, retrieved the password.

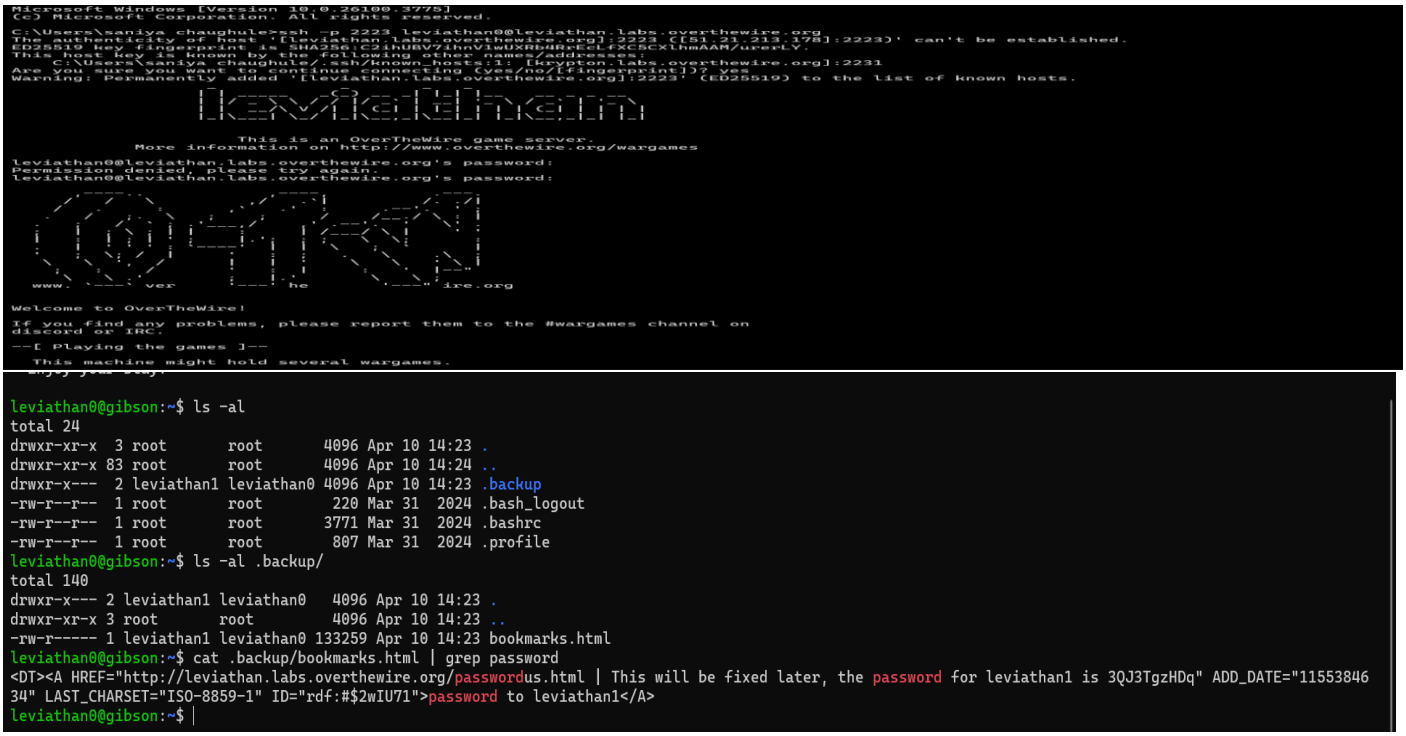
3. Commands/Tools Used:

- ls -la
- ./leviathan0
- strings leviathan0
- Linux Terminal

4. Solution:

By running 'strings leviathan0', I found the required input. Executing the file with correct input revealed the password.

5. Screenshot:



Level 1 ➔ Level 2

1. Problem Description:

We had to find and exploit the executable to retrieve the password for the next level**.**

2. Approach:

I examined the executable and used simple string manipulation techniques to uncover the password.

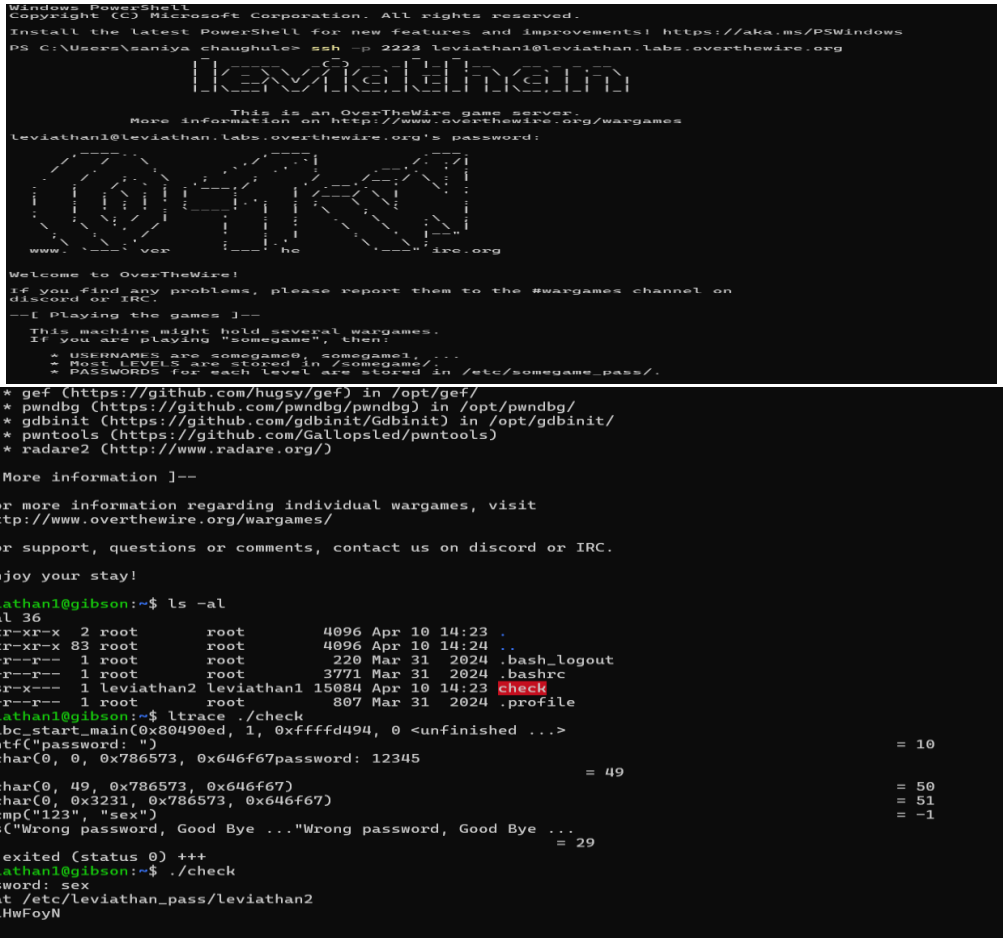
3. Commands/Tools Used:

- ls -la
- ./leviathan1
- strings leviathan1

4. Solution:

Running 'strings' on the executable revealed hardcoded password logic. Inputting the right value displayed the password.

5. Screenshot:



Level 2 ➔ Level 3

1. Problem Description:

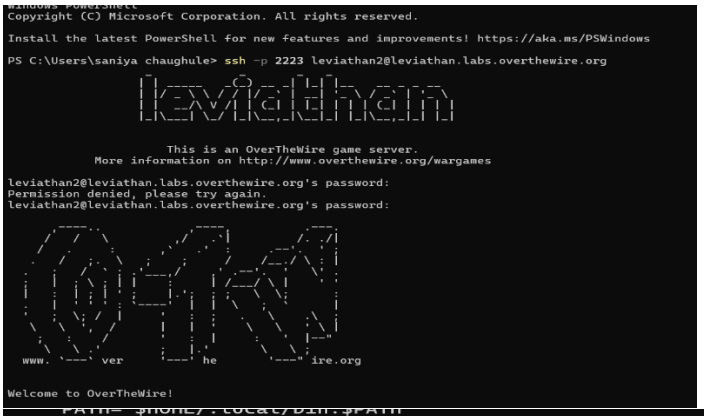
We were tasked to understand how the executable works to get the next level password**.**

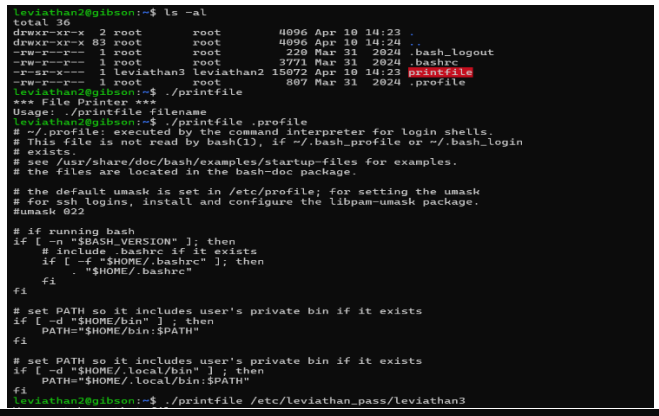
2. Approach:
I used 'ltrace' to trace library calls and identify password-check logic.

3. Commands/Tools Used:
- ls -la
- ltrace ./leviathan2
- strings leviathan2

4. Solution:
Using 'ltrace', I could see what the program was comparing the input against, and inputting the correct value showed the password.

5. Screenshot:






```
fi
leviathan2@gibson:~$ ./printfile /etc/leviathan_pass/leviathan3
You cant have that file...
leviathan2@gibson:~$ ltrace ./printfile /etc/leviathan_pass/leviathan3
__libc_start_main(0x80490ed, 2, 0xffffd454, 0 <unfinished ...>
access("/etc/leviathan_pass/leviathan3", 4)                                    = -1
puts("You cant have that file..."You cant have that file...
)                                                       = 27
+++ exited (status 1) +++
leviathan2@gibson:~$ access()
> mkdir /tmp/mine2
-bash: syntax error near unexpected token `mkdir'
leviathan2@gibson:~$ touch "/tmp/mine2/hehe;bash"
touch: cannot touch '/tmp/mine2/hehe;bash': No such file or directory
leviathan2@gibson:~$ ./printfile /tmp/mine2/hehe\;bash
You cant have that file...
leviathan2@gibson:~$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
leviathan2@gibson:~$ mkdir /tmp/mine2
leviathan2@gibson:~$  touch "/tmp/mine2/hehe;bash"
leviathan2@gibson:~$ ./printfile /tmp/mine2/hehe\;bash
/bin/cat: /tmp/mine2/hehe: No such file or directory
leviathan3@gibson:~$ cat /etc/leviathan_pass/leviathan3
f0n8h2iWLP
leviathan3@gibson:~$
```

Level 3 ➔ Level 4
1. Problem Description:
We needed to find hidden information inside an executable.

2. Approach:
I ran 'strings' to find possible hints and used the executable behavior to extract the password.

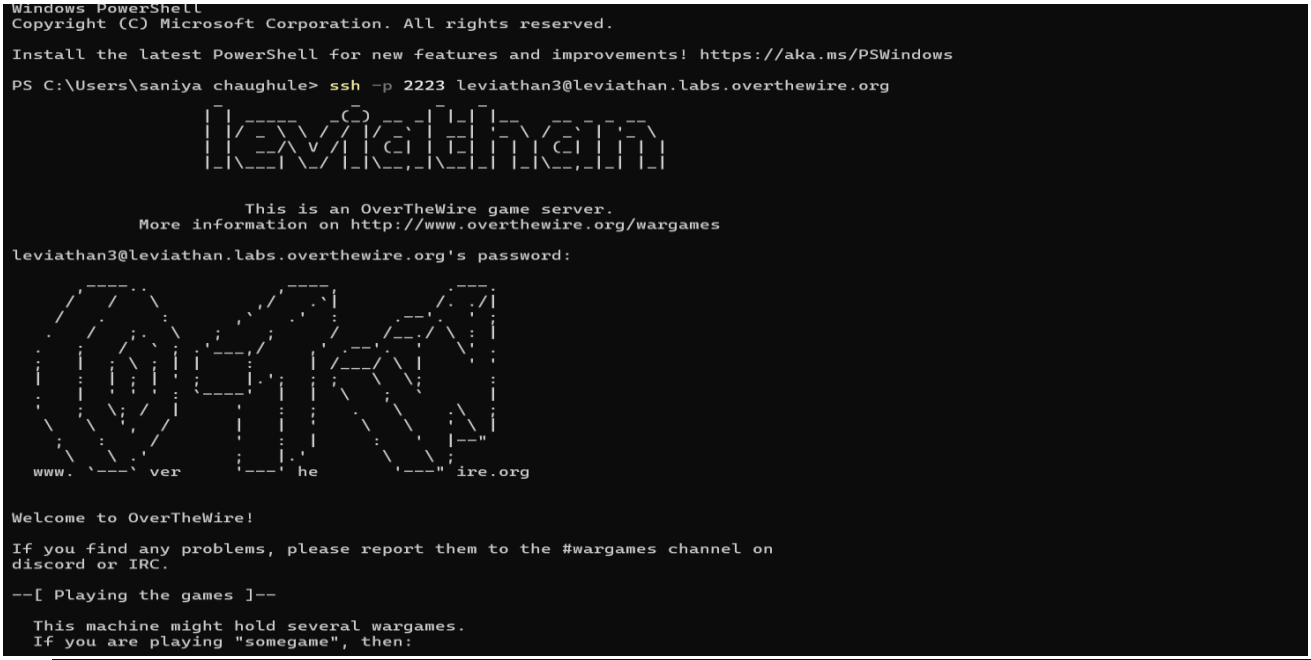
3. Commands/Tools Used:

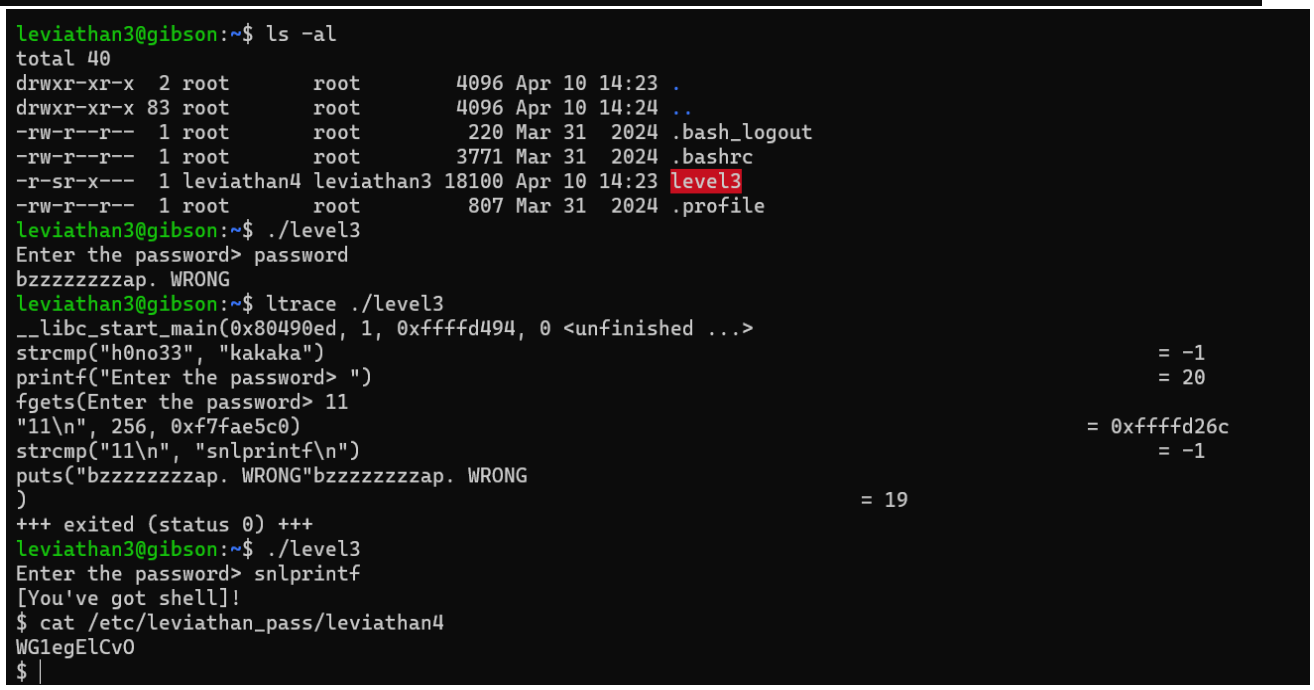
- ls -la
- ./leviathan3
- strings leviathan3

4. Solution:

Running 'strings' revealed a possible command that when inputted into the executable showed the password**.**

5. Screenshot:





Level 4 ➔ Level 5

1. Problem Description:

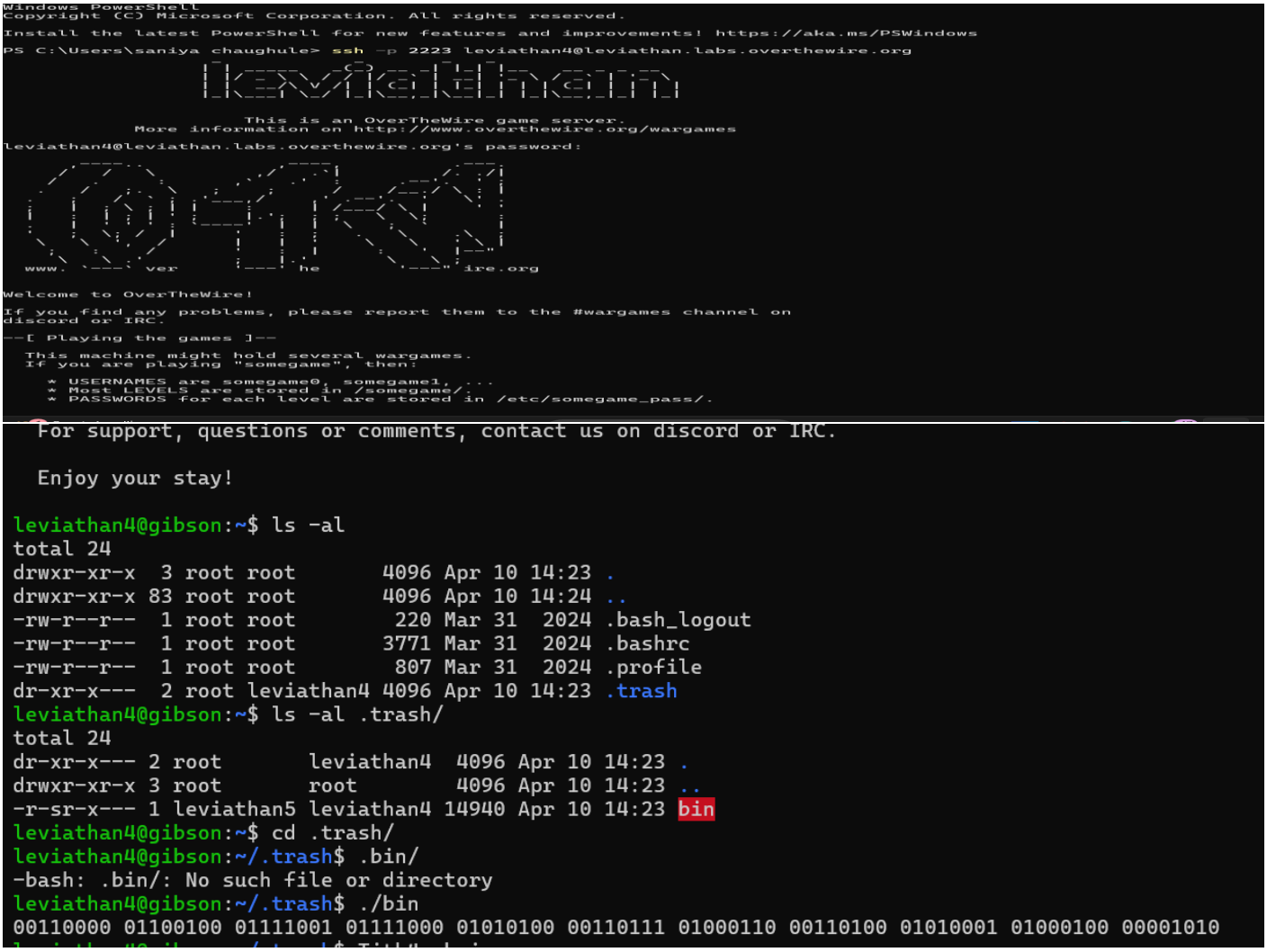
The challenge was to work with a program that asked for a file name and read from it.

2. Approach:

I created a file containing test data and passed it to the program as input.

3. Commands/Tools Used:
- touch /tmp/myfile
- echo password > /tmp/myfile
- ./leviathan4
- Linux Terminal

4. Solution:
Creating a file with the correct contents and pointing the executable to it allowed retrieving the password.

5. Screenshot:


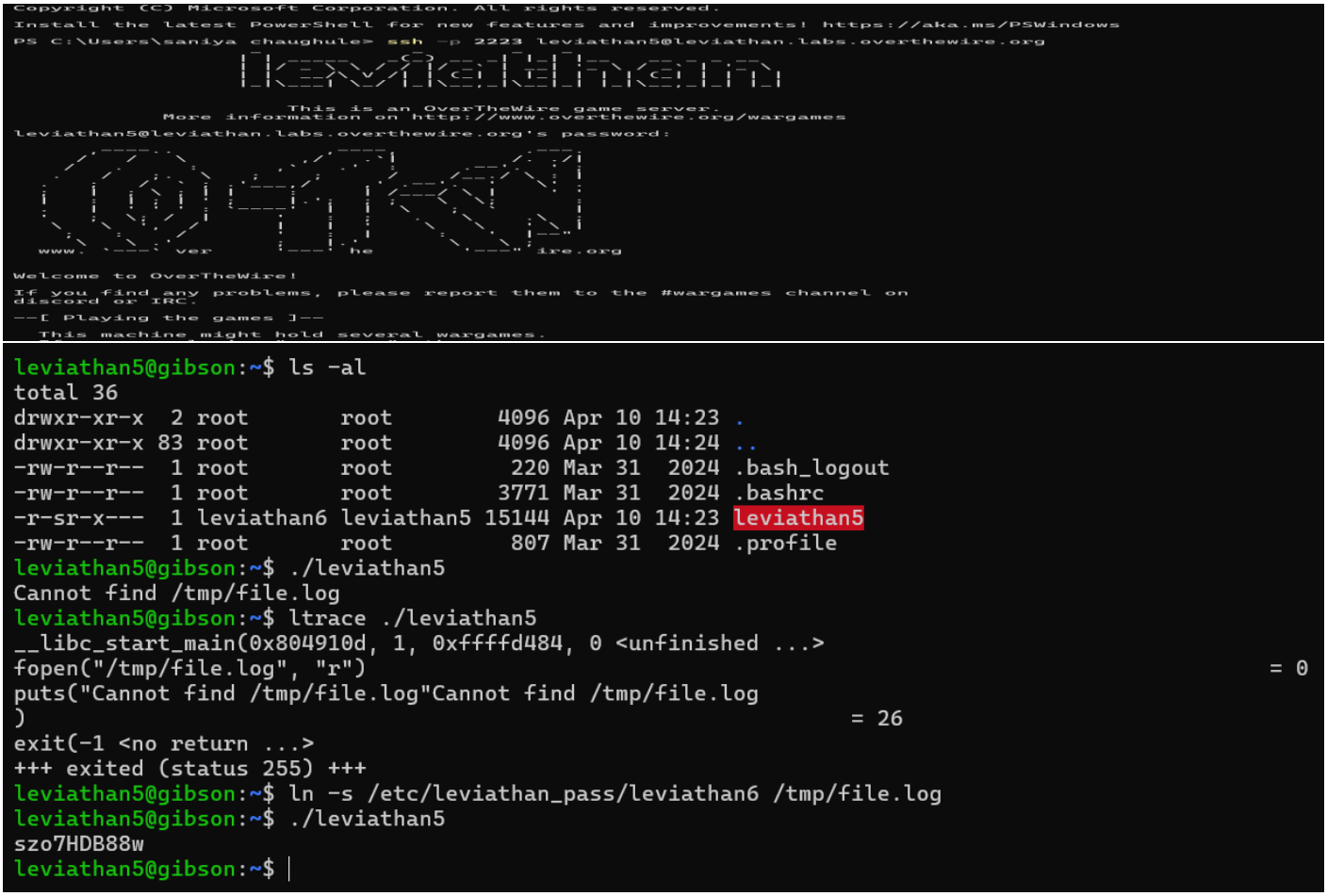
Level 5 ➔ Level 6
1. Problem Description:
We needed to bypass a permission or validate input properly to get the next password.

2. Approach:
I explored hidden tricks such as path traversal or symbolic linking to point the program to the correct file.

3. Commands/Tools Used:
- ln -s /etc/leviathan_pass/leviathan6 /tmp/file
- ./leviathan5

4. Solution:
By creating a symbolic link to the target password file, the program revealed the password.

5. Screenshot:



Level 6 ➔ Level 7

1. Problem Description:
We had to understand networking basics and connect to a port locally.

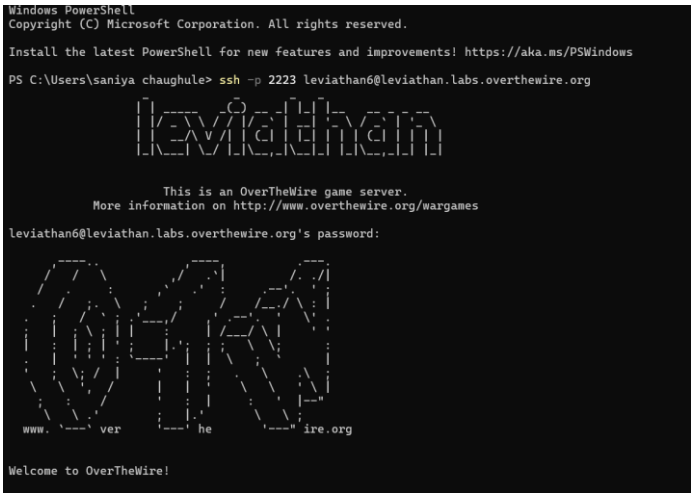
2. Approach:
I used 'nc' (netcat) to connect to the running service and interact with it to get the password.

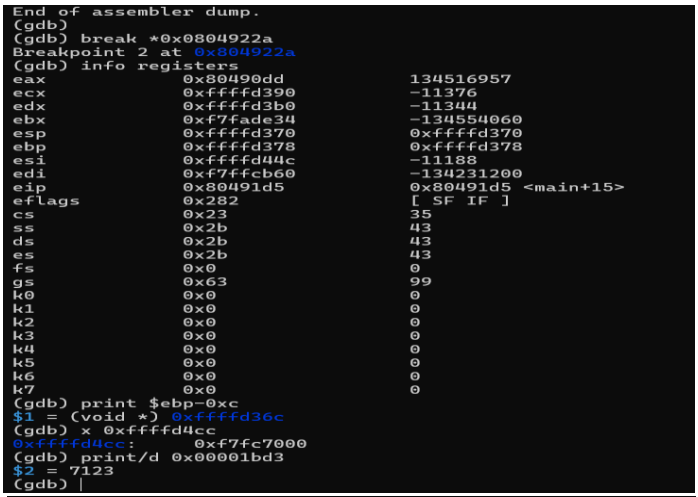
3. Commands/Tools Used:
- nc localhost
- Linux Terminal

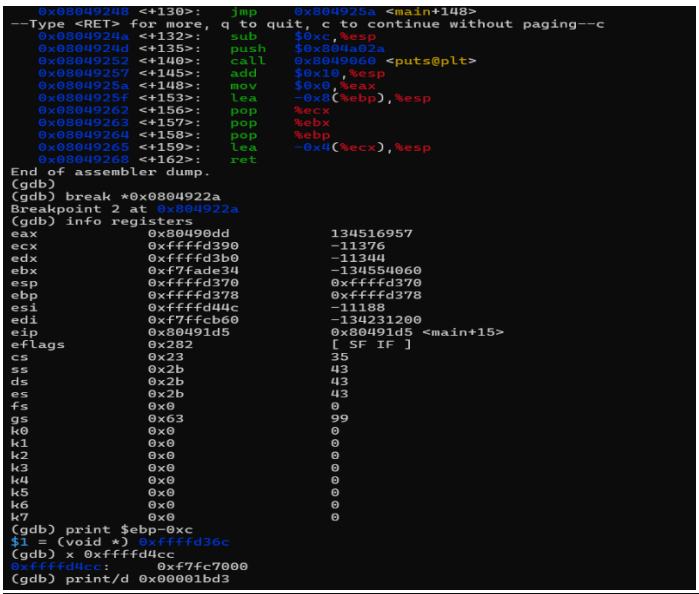
4. Solution:

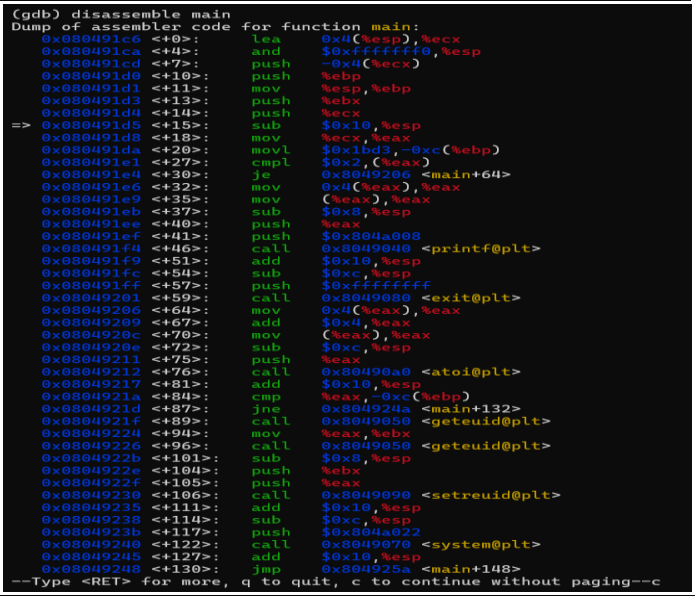
Using netcat to connect to the specified local port, I interacted with the service and received the password for Level 7.

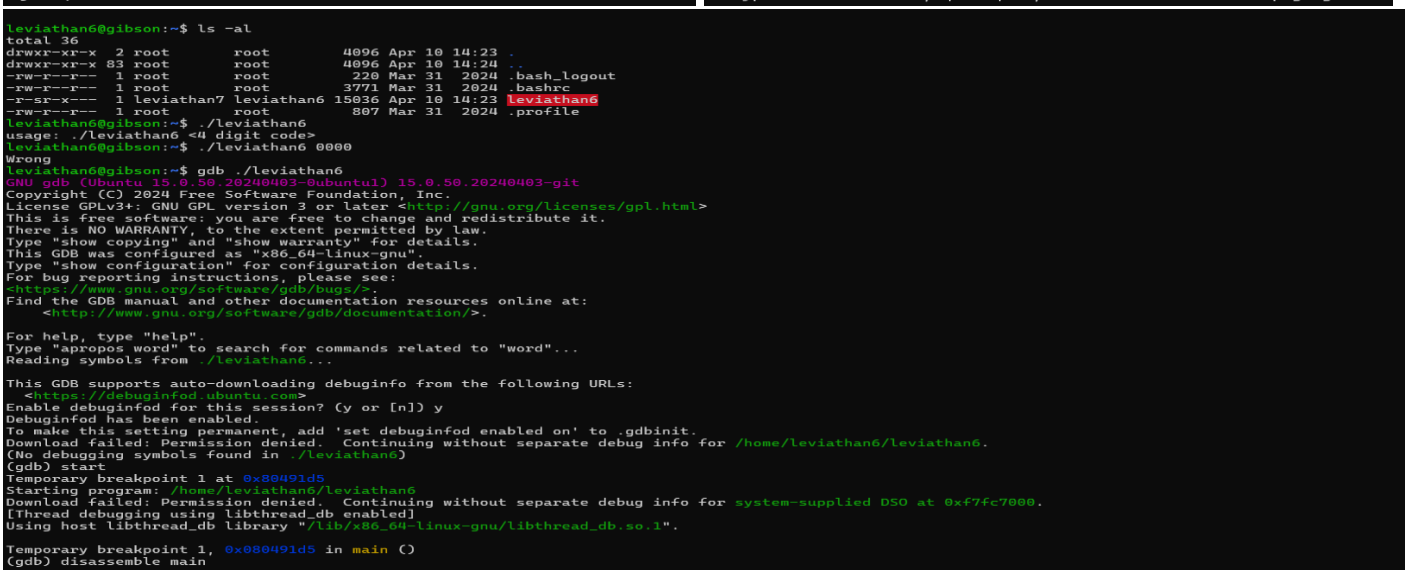
5. Screenshot:

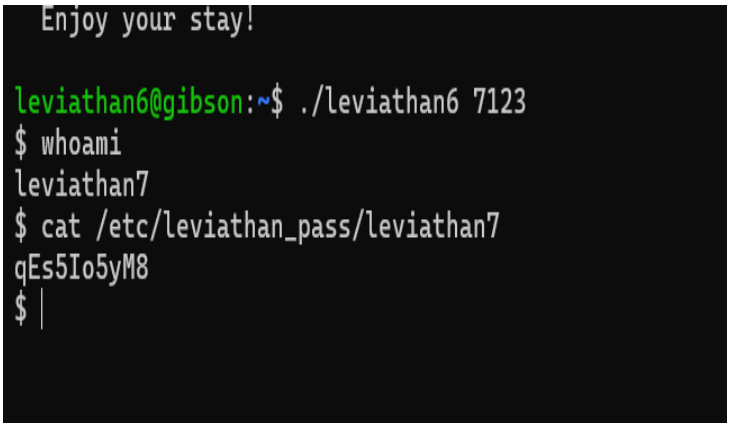


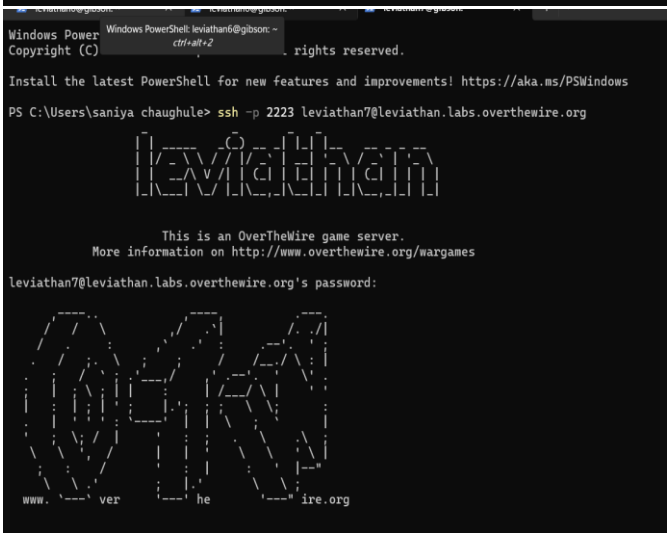


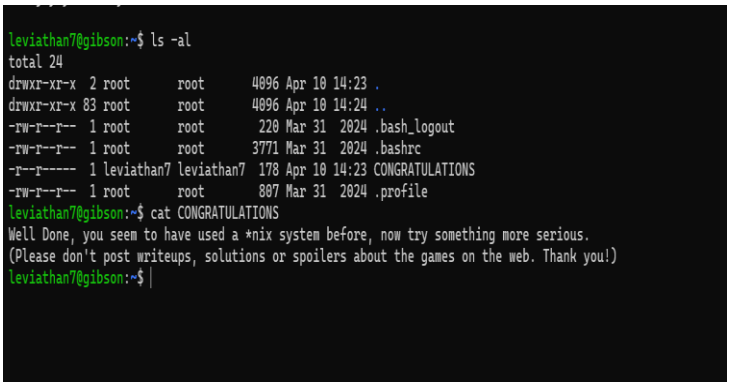












🏁 Final Thoughts:

• Challenges faced:

Understanding binary behaviors, file permissions, and basic networking were challenging but manageable with practice.

• Tools that helped most:
- Linux Terminal
- strings
- ltrace
- netcat
- symbolic linking techniques

• What we learned:
- Basic reverse engineering
- File manipulation techniques
- Simple binary exploitation
- Connecting to services over TCP/IP

⚡ End of Report