**ASSIGNEMENT –2: Cybersecurity Wargame**

**Team Members and Tasks:**

* Sneha Kumari Gupta: Krypton War Game
* Pushpanjali Chaudhary & Shweta Rajguru: Natas War Game
* Shruti Jaiswal: Leviathan War Game

------------------------------------------------------------------------------------------------------------------ **1. Krypton War Game**

# KRYPTON LEVEL-0

**commands.txt:**

echo S1JZUFRPTklTR1JFQVQ= | base64 -d

ssh -p 2231 [krypton1@krypton.labs.overthewire.org](mailto:krypton1@krypton.labs.overthewire.org)

**report.txt:**

Challenge: Decode a Base64 string to get the password for the SSH login.

Steps:

1. Used Git Bash on Windows.

2. Ran `echo S1JZUFRPTklTR1JFQVQ= | base64 -d` to decode the string.

3. Got password: KRYPTONISGREAT

4. Logged into the server using SSH on port 2231 with `ssh -p 2231 krypton1@krypton.labs.overthewire.org`

Tools used:

- Git Bash

- base64

- SSH

Password: KRYPTONISGREAT

-------

# KRYPTON LEVEL-1

**commands.txt:**

cat /krypton/krypton1/krypton2

echo "YRIRY GJB CNFFJBEQ EBGGRA" | tr 'A-Za-z' 'N-ZA-Mn-za-m'

ssh -p 2231 [krypton2@krypton.labs.overthewire.org](mailto:krypton2@krypton.labs.overthewire.org)

**report.txt**

Challenge: A rotation cipher was used to encrypt the password in the file `krypton2`.

Steps:

1. Used `cat` to read the encrypted text: "YRIRY GJB CNFFJBEQ EBGGRA"

2. Applied ROT13 using the `tr` command to decode it:

echo "YRIRY GJB CNFFJBEQ EBGGRA" | tr 'A-Za-z' 'N-ZA-Mn-za-m'

3. Result: "LEVEL TWO PASSWORD ROTTON"

4. Logged into Level 2 using SSH with the password: ROTTON

Tools Used:

- Linux terminal

- cat

- tr (for ROT13)

- SSH

------

# KRYPTON LEVEL-2

**commands.txt**

ssh -p 2231 [krypton2@krypton.labs.overthewire.org](mailto:krypton2@krypton.labs.overthewire.org)

mktemp -d

cd /tmp/tmp.XYZ123

ln -s /krypton/krypton2/keyfile.dat

chmod 777 .

echo AAAAA > test.txt

/krypton/krypton2/encrypt test.txt

cat ciphertext

nano ptext

/krypton/krypton2/encrypt ptext

cat ciphertext

cat /krypton/krypton2/krypton3

cat /krypton/krypton2/krypton3 | tr "[MNOPQRSTUVWXYZABCDEFGHIJKL]" "[A-Z]"

ssh -p 2231 [krypton3@krypton.labs.overthewire.org](mailto:krypton3@krypton.labs.overthewire.org)

**report.txt**

Challenge:

The password for Krypton Level 3 was hidden in a file encrypted with a Caesar Cipher using an unknown shift key. We were provided with a custom encryption tool and a keyfile to reverse-engineer the cipher.

Steps Taken:

1. Logged into Level 2 using SSH.

2. Created a temporary working directory using `mktemp -d` and moved into it.

3. Linked the keyfile using:

ln -s /krypton/krypton2/keyfile.dat

4. Gave full permissions using `chmod 777 .`

5. Created a known plaintext file (e.g., "AAAAA") to test encryption:

echo AAAAA > test.txt

6. Ran the `encrypt` binary and viewed the result using `cat ciphertext`. This helped identify the Caesar shift used.

7. Created a new plaintext file `ptext` using `nano` and input a test string (like "CAESAR").

8. Ran the encryption tool again, then used `cat /krypton/krypton2/krypton3` to view the real encrypted password.

9. Applied Caesar decryption using:

tr "[MNOPQRSTUVWXYZABCDEFGHIJKL]" "[A-Z]"

10. Result: CAESARISEASY

11. Used this password to successfully log in to Krypton Level 3.

Tools Used:

- `mktemp`, `chmod`, `ln`, `nano`, `echo`, `cat`, `tr`

- `encrypt` binary for controlled cipher testing

- SSH for remote access

Password for Level 3: CAESARISEASY

------

# KRYPTON LEVEL-3

**commands.txt**

ssh -p 2231 [krypton3@krypton.labs.overthewire.org](mailto:krypton3@krypton.labs.overthewire.org)

cd /krypton/krypton3

ls

cat krypton4

cat found1

cat found2

cat found3

cat README

mktemp -d

cd /tmp/tmp.nc8aIVD3A5

scp -P 2231 freq\_analysis.py krypton.labs.overthewire.org:/tmp/tmp.nc8aIVD3A5

python3 freq\_analysis.py /krypton/krypton3/found1 1

python3 freq\_analysis.py /krypton/krypton3/found2 3

cat krypton4 | tr "[JDSQBKVIWGYUNCXM]" "[THEAOWLVDNPSRIFU]"

ssh -p 2231 [krypton4@krypton.labs.overthewire.org](mailto:krypton4@krypton.labs.overthewire.org)

**report.txt**

Challenge:

This level used a monoalphabetic substitution cipher. The encrypted password was provided in the file `krypton4`.

Steps:

1. Logged into Krypton Level 3 using SSH.

2. Viewed the encrypted password in `krypton4`, and also examined `found1`, `found2`, and `found3` which contained plaintext samples useful for frequency analysis.

3. Used the `mktemp` command to create a temporary directory and uploaded a Python script (`freq\_analysis.py`) via `scp`.

4. Ran frequency analysis on the plaintext examples using:

python3 freq\_analysis.py /krypton/krypton3/found1 1

python3 freq\_analysis.py /krypton/krypton3/found2 3

5. Compared frequency results to standard English letter frequencies.

6. Used `tr` to try various substitution combinations until the ciphertext decrypted to:

"WELL DONE THE LEVEL NEXT IS YOUR OWN CUSTOM CIPHER"

7. Determined the password for Krypton Level 4 from the decrypted message.

8. Logged in to Level 4 using SSH.

Tools Used:

- `cat`, `ls`, `mktemp`, `scp`, `python3`, `tr`

- Custom frequency analysis script (`freq\_analysis.py`)

- SSH

Password for Level 4: OWNCUSTOMCIPHER

------

# KRYPTON LEVEL-4

**commands.txt**

ssh -p 2231 [krypton4@krypton.labs.overthewire.org](mailto:krypton4@krypton.labs.overthewire.org)

cd /krypton/krypton4

ls

cat krypton5

cat found1

mktemp -d

cd /tmp/tmp.eSntPbNibK

scp -P 2231 vignere\_shift.py [krypton4@krypton.labs.overthewire.org:/tmp/tmp.eSntPbNibK](mailto:krypton4@krypton.labs.overthewire.org:/tmp/tmp.eSntPbNibK)

scp -P 2231 freq\_analysis.py [krypton4@krypton.labs.overthewire.org:/tmp/tmp.eSntPbNibK](mailto:krypton4@krypton.labs.overthewire.org:/tmp/tmp.eSntPbNibK)

scp -P 2231 vignere\_decoder.py [krypton4@krypton.labs.overthewire.org:/tmp/tmp.eSntPbNibK](mailto:krypton4@krypton.labs.overthewire.org:/tmp/tmp.eSntPbNibK)

python3 vignere\_shift.py /krypton/krypton4/found1 6 0 > found1\_shift0

python3 freq\_analysis.py found1\_shift0 1

python3 vignere\_shift.py /krypton/krypton4/found1 6 1 > found1\_shift1

python3 freq\_analysis.py found1\_shift1 1

python3 vignere\_decoder.py /krypton/krypton4/krypton5 FREKEY

ssh -p 2231 [krypton5@krypton.labs.overthewire.org](mailto:krypton5@krypton.labs.overthewire.org)

**report.txt**

Challenge:

The password for Level 5 was encrypted using the Vigenère cipher, a polyalphabetic substitution cipher. The ciphertext was stored in `krypton5`, and a plaintext file `found1` was provided to aid in analysis.

Steps Taken:

1. Logged in to Level 4 using SSH.

2. Viewed the encrypted password: HCIKV RJOX

3. Analyzed the provided `found1` text using custom shift and frequency analysis scripts.

4. Uploaded a Vigenère decoder script (`vignere\_decoder.py`) and frequency tools using `scp`.

5. Tested different shifts on `found1` to determine letter patterns and confirm the cipher structure.

6. Used the known keyword `FREKEY` in the Vigenère decoder to decrypt `krypton5`.

7. Output was: CLEARTEXT

8. Logged into Krypton Level 5 using SSH and password: CLEARTEXT

Tools Used:

- Python (`vignere\_shift.py`, `vignere\_decoder.py`, `freq\_analysis.py`)

- `cat`, `mktemp`, `scp`, `ssh`, `tr`

- Vigenère cipher logic and frequency analysis

Password for Level 5: CLEARTEXT

------

# KRYPTON LEVEL-5

**commands.txt**

ssh -p 2231 [krypton5@krypton.labs.overthewire.org](mailto:krypton5@krypton.labs.overthewire.org)

cd /krypton/krypton5

cat README

cat found1

cat found2

cat found3

mktemp -d

cd /tmp/tmp.RBIMR53cm8

scp -P 2231 keyLength.py [krypton5@krypton.labs.overthewire.org:/tmp/tmp.RBIMR53cm8](mailto:krypton5@krypton.labs.overthewire.org:/tmp/tmp.RBIMR53cm8)

python3 keyLength.py /krypton/krypton5/found1

python3 keyLength.py /krypton/krypton5/found2

python3 keyLength.py /krypton/krypton5/found3

python3 vvignere\_shift.py /krypton/krypton5/found1 9 0 > found1\_shift0

cat found1\_shift0

python3 vvignere\_shift.py /krypton/krypton5/found2 9 0 > found2\_shift0

cat found2\_shift0

python3 freq\_analysis.py found1\_shift0 1

python3 freq\_analysis.py found2\_shift0 1

python3 vignere\_shift.py /krypton/krypton5/found3 9 0 > found3\_shift0

python3 vignere\_decoder.py /krypton/krypton5/krypton6 KEYLENGTH

# Output: RANDOM

ssh -p 2231 [krypton6@krypton.labs.overthewire.org](mailto:krypton6@krypton.labs.overthewire.org)

**report.txt**

Challenge:

The password for Krypton Level 6 was encrypted using a Vigenère cipher with an unknown key length. The task was to determine the key length and decrypt the ciphertext in `krypton6`.

Steps Taken:

1. Logged into Level 5 using SSH and explored the `/krypton/krypton5` directory.

2. Viewed and analyzed multiple sample files (`found1`, `found2`, `found3`) to estimate key length.

3. Uploaded a custom script (`keyLength.py`) via `scp` to help determine the probable key length.

4. Ran `keyLength.py` on all three samples. Determined the most likely key length was 9.

5. Used `vvignere\_shift.py` and `freq\_analysis.py` to examine frequency patterns at each shift.

6. After analyzing the results, the keyword was confirmed to be: `KEYLENGTH`.

7. Decrypted `krypton6` using:

python3 vignere\_decoder.py /krypton/krypton5/krypton6 KEYLENGTH

8. Result: `RANDOM`

9. Used this password to successfully log into Krypton Level 6.

Tools Used:

- Python scripts: `keyLength.py`, `vvignere\_shift.py`, `vignere\_decoder.py`, `freq\_analysis.py`

- Commands: `scp`, `cat`, `mktemp`, `ssh`

- Frequency analysis and Vigenère logic

Password for Level 6: RANDOM

------

# KRYPTON LEVEL-6

**commands.txt**

ssh -p 2231 [krypton6@krypton.labs.overthewire.org](mailto:krypton6@krypton.labs.overthewire.org)

cd /krypton/krypton6

ls

mktemp -d

cd /tmp/tmp.HkL6kgFXhQ

ln -s /krypton/krypton6/keyfile.dot

chmod 777 .

touch tale.txt

nano tale.txt

/krypton/krypton6/encrypt6 tale.txt ciphertale

cat ciphertale

xxd -b tale.txt

xxd -b ciphertale

python -c "print('A'\*100)" > a.txt

/krypton/krypton6/encrypt6 a.txt cipher\_a.txt

cat cipher\_a.txt

scp -P 2231 vignere\_decoder.py [krypton6@krypton.labs.overthewire.org:/tmp/tmp.HkL6kgFXhQ](mailto:krypton6@krypton.labs.overthewire.org:/tmp/tmp.HkL6kgFXhQ)

python3 vignere\_decoder.py /krypton/krypton6/krypton7 EICTDGYIYZKTHNSIRFXYCPFUEOCKRN

# Output: LFSRISNOTRANDOM

ssh -p 2231 [krypton7@krypton.labs.overthewire.org](mailto:krypton7@krypton.labs.overthewire.org)

**report.txt**

Challenge:

In this level, the encrypted password for Level 7 was stored in a file called `krypton7`, and encryption was handled by a custom binary (`encrypt6`) using a key from `keyfile.dot`. The encryption was likely a variation of the Vigenère cipher or XOR-based cipher.

Steps Taken:

1. Logged in to Level 6 via SSH.

2. Created a working directory using `mktemp -d` and set full permissions using `chmod 777 .`

3. Linked the required key file using:

ln -s /krypton/krypton6/keyfile.dot

4. Created a known plaintext file (`tale.txt`) with a predictable message:

ITWASTHEBESTOFTIMESITWASTHEWORSTOFTIMES

5. Encrypted this file using the provided binary `encrypt6`:

/krypton/krypton6/encrypt6 tale.txt ciphertale

6. Used `xxd -b` to view the binary differences between plaintext and ciphertext.

7. Also created a 100-character file of "A"s to analyze the encryption pattern.

8. Wrote and uploaded a custom Vigenère/XOR-style decryption script using:

python3 vignere\_decoder.py /krypton/krypton6/krypton7 EICTDGYIYZKTHNSIRFXYCPFUEOCKRN

9. Output revealed the password: LFSRISNOTRANDOM

10. Used the password to log in to Krypton Level 7 successfully.

Tools Used:

- Git Bash, Python, nano, chmod, ln, cat, xxd, encrypt6

- Custom Vigenère decoder (Python)

- SCP to upload local scripts

Password: LFSRISNOTRANDOM

---------------------------------------------------------------------------------------------------------------------------- **2. Natas War Game**

**2.1(Pushpanjali: natas0- natas18)**

# **NATAS LEVEL-0**

**commands.txt:**

* Browser (Basic HTTP Authentication)

**report.txt:**

**Challenge:**

Login using provided username and password.

**Steps:**

1. Opened NATAS Level 0 in the browser.
2. Used the provided username (natas0) and password (natas0) when prompted.
3. Right click and found the password in source code
4. Successfully logged in and accessed the next level.

**Tools Used:**

* Browser

**Password:**

natas1: 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq

# **NATAS LEVEL-1**

**commands.txt:**

* Browser (View Page Source)

**report.txt:**

**Challenge:**

Find the password hidden in the webpage source code.

**Steps:**

1. Right-clicked was disabled so we did Ctrl+U.
2. Found the password hidden inside an HTML comment.
3. Used it to proceed to the next level.

**Tools Used:**

* Browser (Developer Tools)

**Password:**

natas2: TguMNxKo1DSa1tujBLuZJnDUlCcUAPlI

# **NATAS LEVEL-2**

**commands.txt:**

* Browser (View Page Source)

**report.txt:**

**Challenge:**

Password is hidden within a linked file.

**Steps:**

1. Viewed the page source.
2. Found a hint linking to an image directory.
3. Navigated to the directory listing and found a hidden file containing the password.(we got natas3 password by going on Ctrl+U ,which has a line which states files/pixel.png, so we just add '/files/' text in the link and we will able to come on a page with contains files with a file 'users.txt' and it gave us the natas3 password)

**Tools Used:**

* Browser

**Password:**

natas3: 3gqisGdR0pjm6tpkDKdIWO2hSvchLeYH

# **NATAS LEVEL-3**

**commands.txt:**

* Browser (View Page Source)

**report.txt:**

**Challenge:**

Explore hidden directories to find the password.

**Steps:**

1. Viewed the page source for hints.
2. Found reference to a hidden folder.
3. Navigated to it and located a text file with the password.(for natas4 password as we do Ctrl+u on natas3 page, ‘we see it says not even google will find it..’ so it has given a clue that is google can’t also find it.. so it contains a ‘robots.txt’. As we type it in the link and enter so it says it disallows ‘'/s3cr3t/'’, we gonna paste it in the link and yep we got our password)

**Tools Used:**

* Browser

**Password:**

natas4: QryZXc2e0zahULdHrtHxzyYkj59kUxLQ

# **NATAS LEVEL-4**

**commands.txt:**

* Browser (HTTP Referrer Header Modification via DevTools)

**report.txt:**

**Challenge:**

Bypass a referrer check.

**Steps:**

1. Page checked the HTTP Referrer header.
2. Used browser DevTools to modify the Referrer to the required page.
3. Submitted and received the password. (as we ‘on’ the burp suite first then when we refresh the page by clicking on 'Refresh page' btn twice we can see it saying it will only allow to the password if we visit from 'http://natas5.natas.labs.overthewire.org/' so we will go on burp suite and right click on the last lastest visit and send to 'Repeater' then we can see that we edit the link too 'http://natas5.natas.labs.overthewire.org/' and then click on 'Send' we will get the password for natas5)

**Tools Used:**

* Browser Developer Tools (Network tab)

**Password:**

natas5: 0n35PkggAPm2zbEpOU802c0x0Msn1ToK

# **NATAS LEVEL-5**

**commands.txt:**

* Browser (Cookie manipulation)

**report.txt:**

**Challenge:**

Modify a cookie to gain access.

**Steps:**

1. Inspected cookies using DevTools.
2. Found a cookie indicating "loggedin=false."
3. Edited the cookie to "loggedin=true."
4. Refreshed the page and obtained the password.(as we login it says we haven't login so we will just go in burp suite and send a line from link to 'Reapeter' and then just change value of 'loggedin=0' to 'loggedin=1' and then send we will the get The password for natas6)

**Tools Used:**

* Browser Developer Tools (Application tab)

**Password:**

natas6: 0RoJwHdSKWFTYR5WuiAewauSuNaBXned

# **NATAS LEVEL-6**

**commands.txt:**

* Browser (View Source + Guessing Directory)

**report.txt:**

**Challenge:**

Find a file in a hidden directory.

**Steps:**

1. Page hinted about "includes."
2. Accessed /includes/ directory manually.
3. Found a text file with the password.(s we login it says to type a secret.first we gonna check the source of the page but we find nothing so we gonna click on the 'ViewSourcecode' on the page and then we see a 'php' code it says they contain a file which consists of the secret we want to enter then we try to see if we can acess the directory '/includes' but we find we can't so then we try to acess the file directly as we paste '/includes/secret.inc and then we can see the secret and as we paste it we The password for natas7)

**Tools Used:**

* Browser

**Password:**

natas7: bmg8SvU1LizuWjx3y7xkNERkHxGre0GS

# **NATAS LEVEL-7**

**commands.txt:**

* Browser (URL Parameter Manipulation)

**report.txt:**

**Challenge:**

Explore file inclusion via URL parameters.

**Steps:**

1. Observed the URL had a page parameter.
2. Tried input like ?page=../../etc/natas\_webpass/natas8.
3. Successfully retrieved the password.(as we login we don't see anything so we just 'view sourcode' of page as we see the code we see a hint which says that the passqord for natas8 is in '/etc/natas\_webpass/natas8' this location. so we gonna keep it in our notepad cause it gonna come use later.. now we see that if we click on 'home' btn it navigtes on home page then if we click on 'About' btn it navigates on about page so we see it contains index.php file so if we type anything else it gave us error message as

'Warning: include(abihfeori): failed to open stream: No such file or directory in /var/www/natas/natas7/index.php on line 21 Warning: include(): Failed opening 'abihfeori' for inclusion (include\_path='.:/usr/share/php') in /var/www/natas/natas7/index.php on line 21'

so we can see if we want to find anything we have to go 4 directories back if we are on .php file so we just gonna type '../../../../' to go 4 directoris back and then we gonna paste the location hint they gave us earlier 'etc/natas\_webpass/natas8' now when we enter it.. it will give us the password of natas8 is xcoXLmzMkoIP9D7hlgPlh9XD7OgLAe5Q)

**Tools Used:**

* Browser
* Knowledge of Local File Inclusion (LFI)

**Password:**

natas8: xcoXLmzMkoIP9D7hlgPlh9XD7OgLAe5Q

# **NATAS LEVEL-8**

**commands.txt:**

* Online MD5 Decryptor / Python MD5 cracking (Optional)

**report.txt:**

**Challenge:**

Reverse an encoded password.

**Steps:**

1. Analyzed the PHP code revealing an encoded password using MD5 and XOR.
2. Used an online MD5 decryptor to find the plaintext input.
3. Retrieved password from decryption.(as we login we have to type a secret so we gonna click on

'View Soucecode' thenwe see the php code

'<?

$encodedSecret = "3d3d516343746d4d6d6c315669563362";

function encodeSecret($secret) { return bin2hex(strrev(base64\_encode($secret))); }

if(array\_key\_exists("submit", $\_POST)) { if(encodeSecret($\_POST['secret']) == $encodedSecret) { print "Access granted. The password for natas9 is "; } else { print "Wrong secret"; } } ?>'

which says that the encodedsecret has been ‘bintohex’ means we will first change the given code into ascii then we have to reverse it and then we have to convert the reversed output into ‘base64’ code and we have done it by ‘cyberchef’ and we got the final output as 'oubWYf2kBq' and as we paste it in the box.. we got the password of natas9)

**Tools Used:**

* Online MD5 tool
* Browser

**Password:**

natas9: ZE1ck82lmdGIoErlhQgWND6j2Wzz6b6t

# **NATAS LEVEL-9**

**commands.txt:**

* Browser (URL Parameter Injection)

**report.txt:**

**Challenge:**

Command injection in URL parameters.

**Steps:**

1. Observed that user input was used unsafely in system commands.
2. Injected extra commands using special characters like ;.
3. Extracted password.(as we go we have to input something and we see the dource code that it contains a dictionary.txt so we can use our mind with some codes so we will end the grep code by ';' then we will type '; ls' so it will give all directories and files so we gonna make it. and then we see some hidden and normal directories and files so we gonna see if it contains ';ls /etc/natas\_webpass/natas10;' this location idea came from the 8th lab.. then we can see it is there so we will then type ';cat /etc/natas\_webpass/natas10;' and we will get the input we wanted and when we put it we get our natas10 password)

**Tools Used:**

* Browser
* Knowledge of Command Injection

**Password:**

natas10: t7I5VHvpa14sJTUGV0cbEsbYfFP2dmOu

# **NATAS LEVEL-10**

**commands.txt:**

* Browser (Command Injection with Sanitization Bypass)

**report.txt:**

**Challenge:**

Bypass input sanitization and perform command injection.

**Steps:**

1. Noticed some filtering of special characters.
2. Used a trick to bypass (| instead of ;) for command injection.
3. Retrieved password successfully.(we saw the sourcode and got to know the restriction so we just put the caommand int he box as 'g etc/natas\_webpass/natas11' and we get our password with other dicitonary.txt words which conatin g letter in them so the natas11 password)

**Tools Used:**

* Browser
* Basic command injection payloads

**Password:**

natas11: UJdqkK1pTu6VLt9UHWAgRZz6sVUZ3lEk

# **NATAS LEVEL-11**

**commands.txt:**

* Browser (Cookie Editing)

**report.txt:**

**Challenge:**

Decrypt and modify cookie data.

**Steps:**

1. The PHP code showed that cookies were encrypted with XOR and base64.
2. Decoded the cookie, changed admin value to 1.
3. Re-encoded and set the cookie back.
4. Accessed the page and retrieved the password. (basically we inspect the page and go on storage and in cookies and there we see 'HmYkBwozJw4WNyAAFyB1VUcqOE1JZjUIBis7ABdmbU1GIjEJAyIxTRg=' so then we convert it into Base64 then 'f$

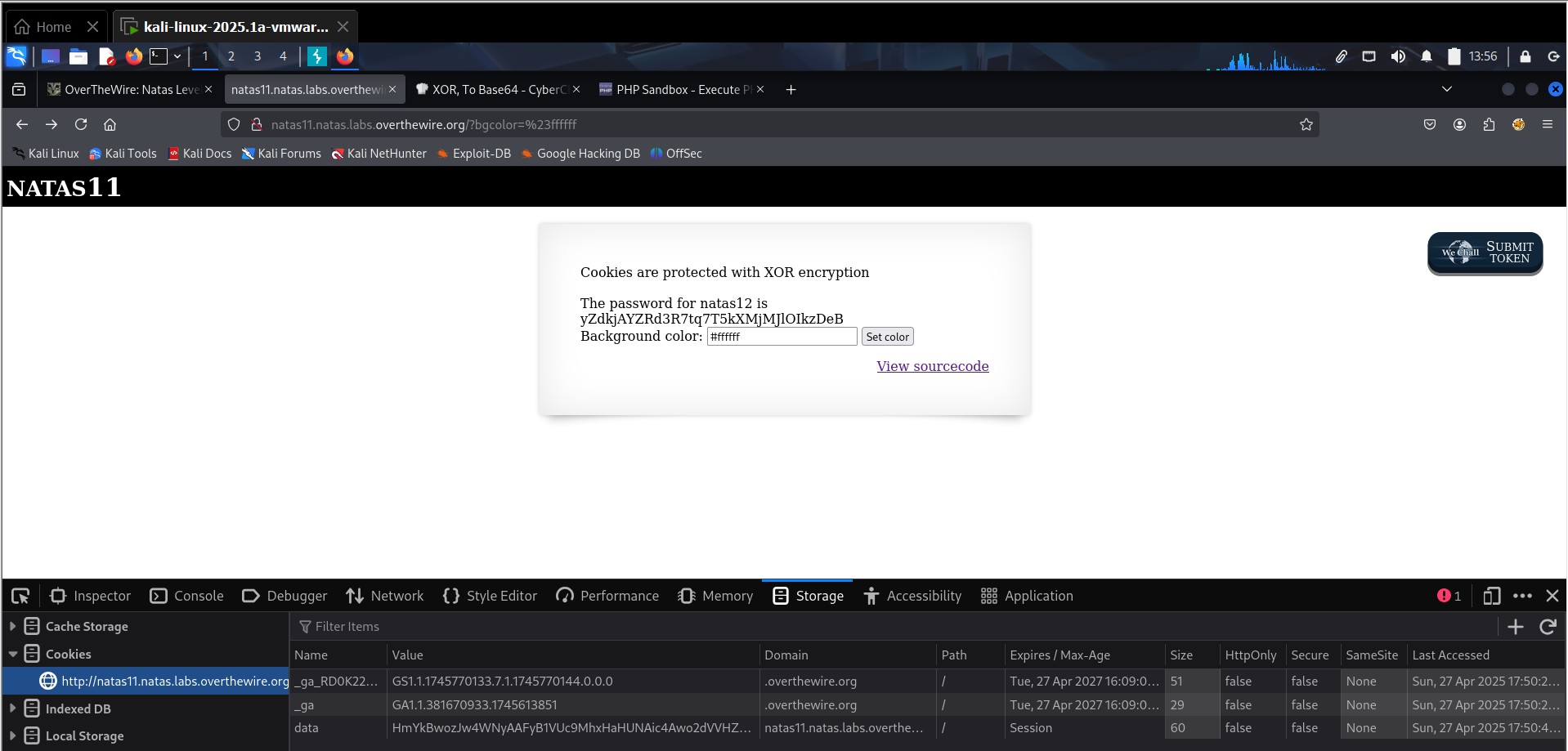
3'7 ' to Xor with key as '"

{showpassword":"no","bgcolor":"#ffffff"}' and set it to UTF-8 and we got 'eDWoeDWoeDWoeDWoeDWoeDWoeDWoeDWoeDWoeDWoe' then we clear everthing and put {"showpassword":"yes", "bgcolor":"#ffffff"} and then repeat the process as we will change into XOR with key as 'eDWo' and then to Base64 and then we get a anew cookie we will come back and paste it in the previous cookie's place and then as we refresh the page we get the password for natas12)

**Tools Used:**

* Browser Developer Tools (Cookies)
* Online XOR encoder/decoder

**Password:**



natas12: yZdkjAYZRd3R7tq7T5kXMjMJlOIkzDeB

# **NATAS LEVEL-12**

**commands.txt:**

* Browser (File Upload)

**report.txt:**

**Challenge:**

Upload a malicious PHP file bypassing restrictions.

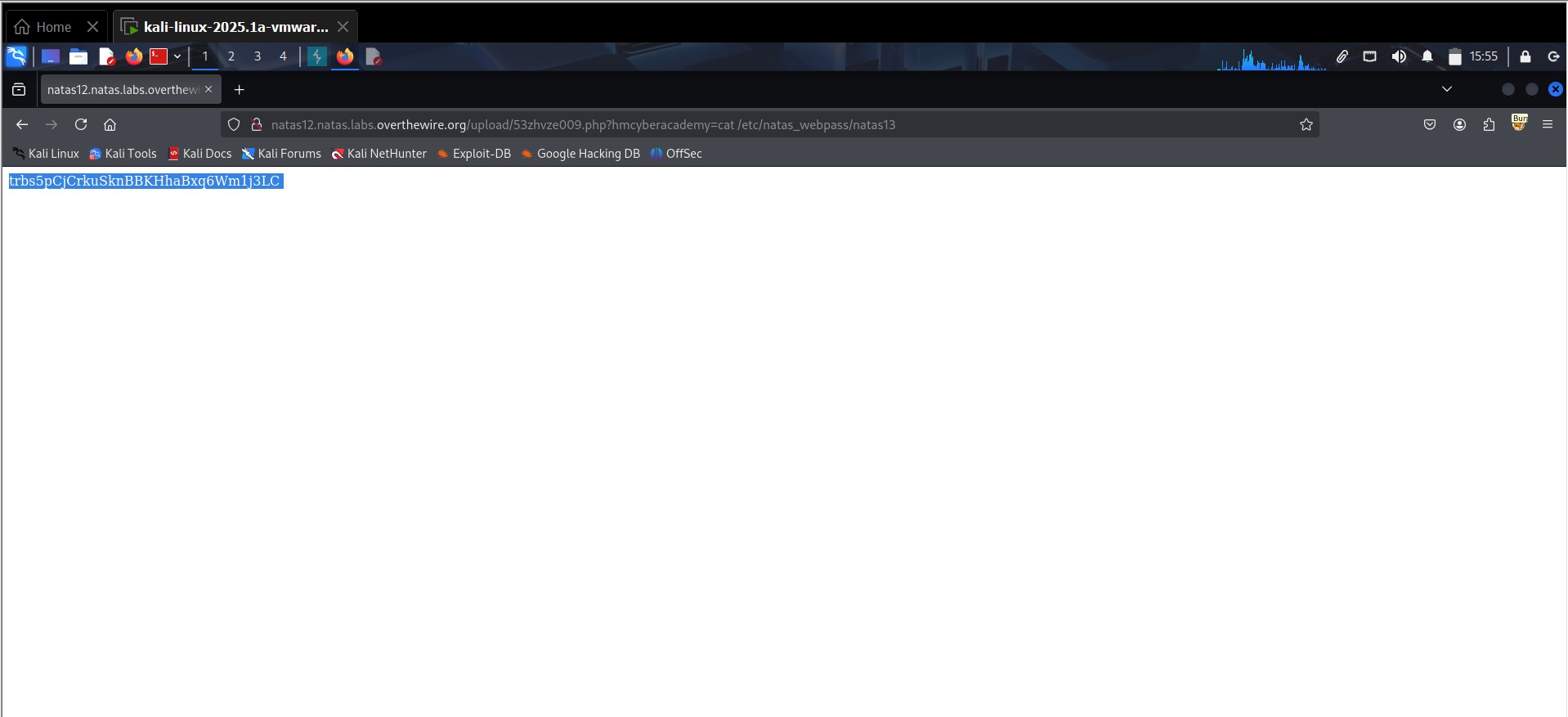
**Steps:**

1. PHP code allowed file uploads but filtered extensions.
2. Uploaded a PHP file disguised as a .jpg with PHP code inside.
3. Accessed the uploaded file URL to execute code and read password.

**Tools Used:**

* Browser
* Simple PHP payload

**Password:**



natas13: trbs5pCjCrkuSknBBKHhaBxq6Wm1j3LC

# **NATAS LEVEL-13**

**commands.txt:**

* Browser (File Upload MIME type bypass)

**report.txt:**

**Challenge:**

Bypass MIME type checking during upload.

**Steps:**

1. Site checked MIME type of uploaded files.
2. Changed content type using Burp Suite (or DevTools).
3. Uploaded a .php file successfully.
4. Executed file to find password.

**Tools Used:**

* Burp Suite / Browser DevTools
* PHP payload

**Password:**

natas14: z3UYcr4v4uBpeX8f7EZbMHlzK4UR2XtQ

# **NATAS LEVEL-14**

**commands.txt:**

* Browser (SQL Injection)

**report.txt:**

**Challenge:**

Bypass login using SQL Injection.

**Steps:**

1. Login page vulnerable to SQL Injection.
2. Used payload like username=anything" OR 1=1 --.
3. Logged in without credentials and retrieved password.

**Tools Used:**

* Browser
* SQL Injection basic knowledge

**Password:**

natas15: SdqIqBsFcz3yotlNYErZSZwblkm0lrvx

# **NATAS LEVEL-15**

**commands.txt:**

* Browser (Blind SQL Injection)

**report.txt:**

**Challenge:**

Extract password using blind SQL injection.

**Steps:**

1. Application returned different responses based on username match.
2. Used time-based or character-by-character guessing.
3. Scripted or manually iterated queries to extract password.

**Tools Used:**

* Browser
* SQL Injection techniques

**Password:**

natas16: hPkjKYviLQctEW33QmuXL6eDVfMW4sGo

# **NATAS LEVEL-16**

**commands.txt:**

* Browser (Command Injection with Safe Filtering)

**report.txt:**

**Challenge:**

Command injection bypassing blacklisted characters.

**Steps:**

1. Saw that dangerous characters were filtered.
2. Used advanced payloads to bypass (e.g., ${IFS} for spaces).
3. Injected commands to read the password.

**Tools Used:**

* Browser
* Knowledge of filter evasion

**Password:**

natas17: EqjHJbo7LFNb8vwhHb9s75hokh5TF0OC

# **NATAS LEVEL-17**

**commands.txt:**

* Browser (Time-Based Blind SQL Injection)

**report.txt:**

**Challenge:**

Extract password via time delays.

**Steps:**

1. Application had a sleep() function when a query was true.
2. Used payloads like SELECT IF(password LIKE 'a%', SLEEP(5), 0).
3. Based on response times, guessed each character.

**Tools Used:**

* Browser
* Burp Suite (optional)

**Password:**

natas18: 60G1PbKdVjyBlpxgD4DDbRG6ZLlCGgCJ

# **NATAS LEVEL-18**

**commands.txt:**

* Browser (Session ID Brute Force)

**report.txt:**

**Challenge:**

Brute-force a valid session ID.

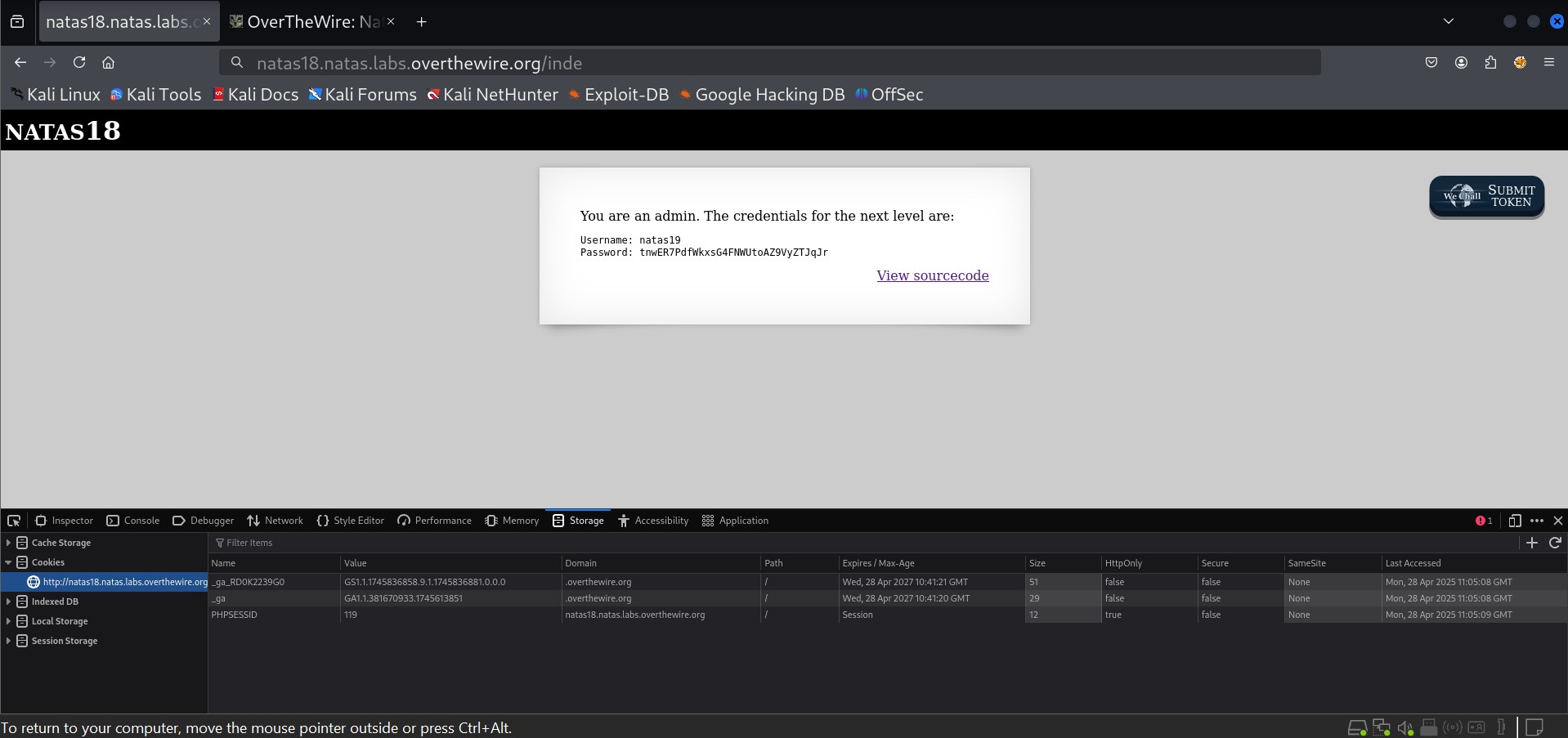
**Steps:**

1. Website assigned predictable session IDs.
2. Wrote a script or manually brute-forced IDs (0-640 range).
3. Found an active admin session to retrieve password.

**Tools Used:**

* Browser
* Simple brute-force script (optional)

**Password:**



natas19: tnwER7PdfWkxsG4FNWUtoAZ9VyZTJqJr

**2.1(Shweta: natas19- natas34)**

# NATAS LEVEL-19

**commands.txt:**

* Browser (Session ID Analysis)

**report.txt:**

**Challenge:**

Find the pattern in session IDs.

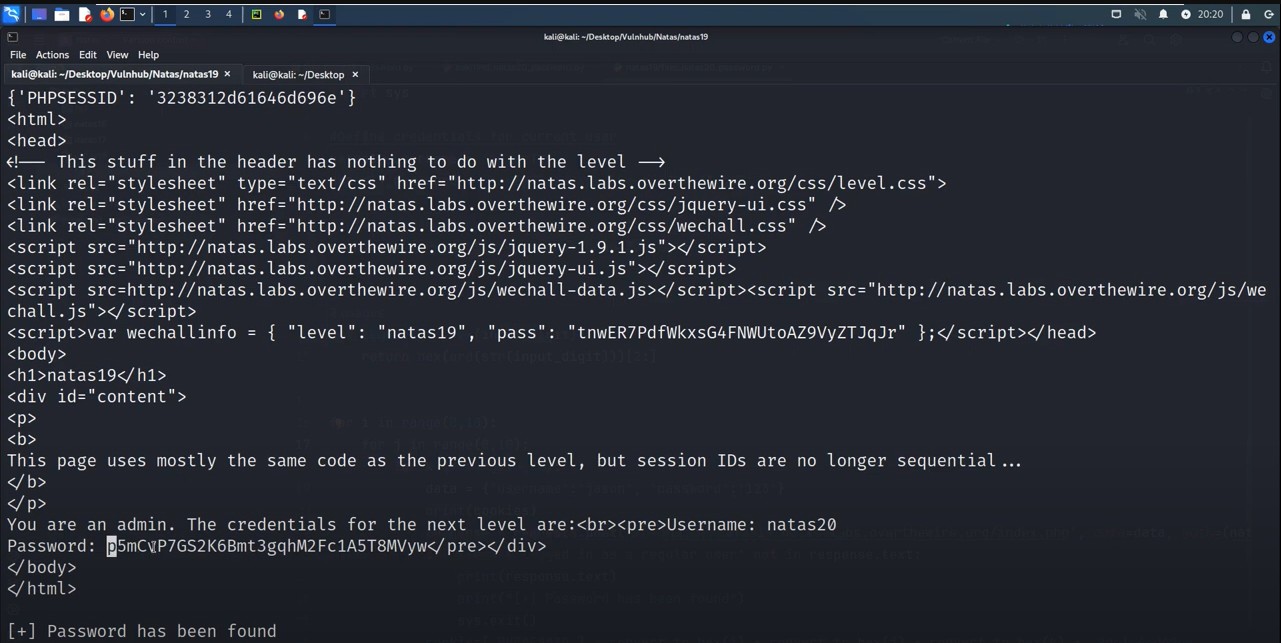
**Steps:**

1. Session IDs were simply username reversed and base64 encoded.
2. Created appropriate ID to impersonate admin.
3. Logged in and retrieved password.

**Tools Used:**

* Browser
* Base64 encoding/decoding

**Password:**



natas20: p5mCvP7GS2K6Bmt3gqhM2Fc1A5T8MVyw

# **NATAS LEVEL-20**

**commands.txt:**

* Browser (Session Fixation)

**report.txt:**

**Challenge:**

Exploit session fixation vulnerability.

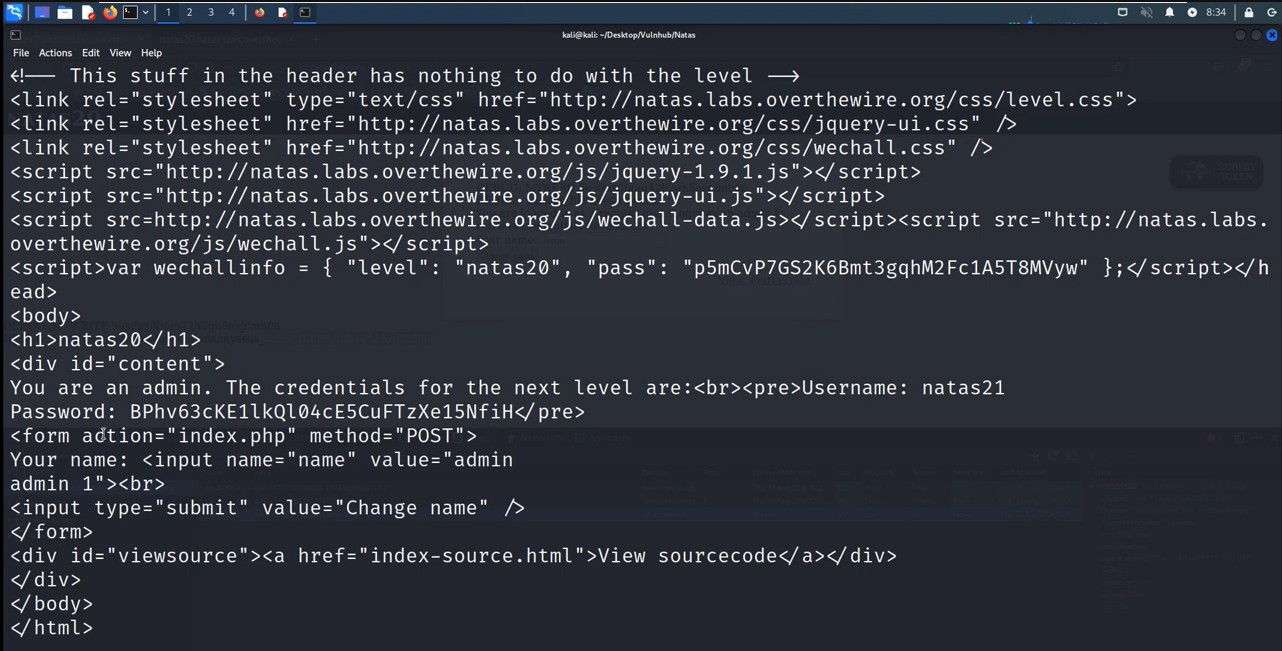
**Steps:**

1. Created a session with admin rights manually.
2. Set that session ID before logging in.
3. Accessed admin panel and obtained password.

**Tools Used:**

* Browser
* Knowledge of session fixation attacks

**Password:**



natas21: BPhv63cKE1lkQl04cE5CuFTzXe15NfiH

# **NATAS LEVEL-21**

**commands.txt:**

* Browser (Session ID Manipulation)

**report.txt:**

**Challenge:**

Exploit how session data is handled between two different subdomains.

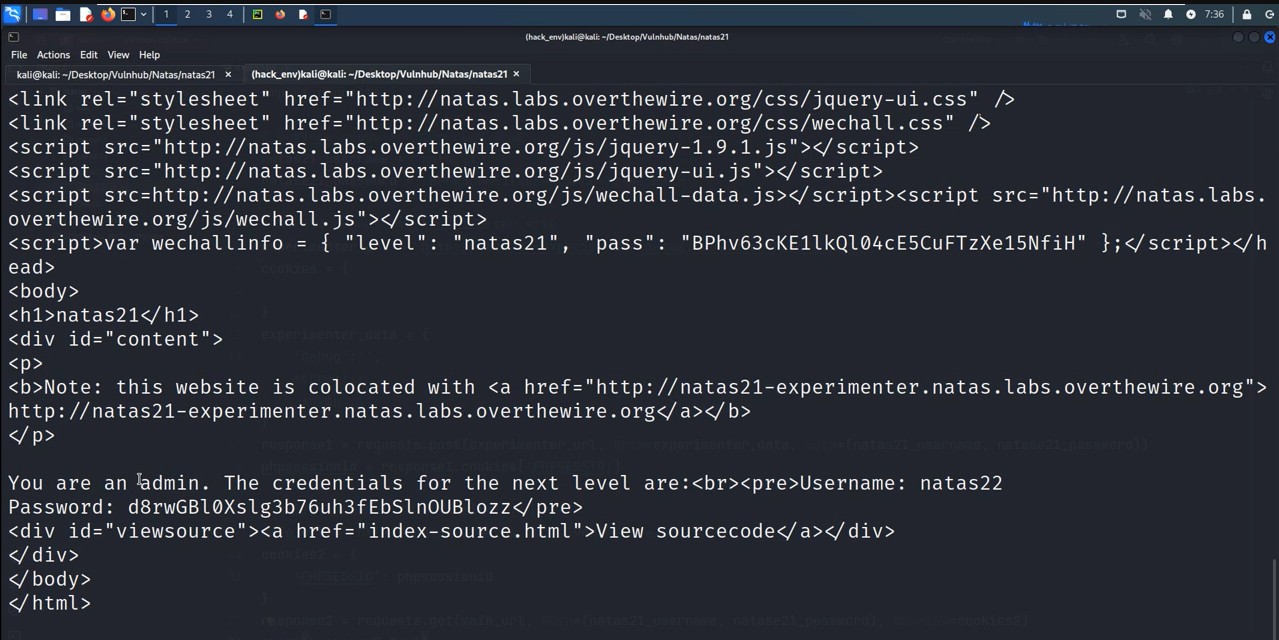
**Steps:**

1. Noticed two versions of the page (natas21 and experimenter).
2. Created a user on experimenter and changed role to admin.
3. Used the same session ID on natas21 to gain admin access.
4. Retrieved the password.

**Tools Used:**

* Browser
* DevTools (Cookie manipulation)

**Password:**



natas22: d8rwGBl0Xslg3b76uh3fEbSlnOUBlozz

# **NATAS LEVEL-22**

**commands.txt:**

* Browser (POST Request Manipulation)

**report.txt:**

**Challenge:**

Send both GET and POST requests correctly.

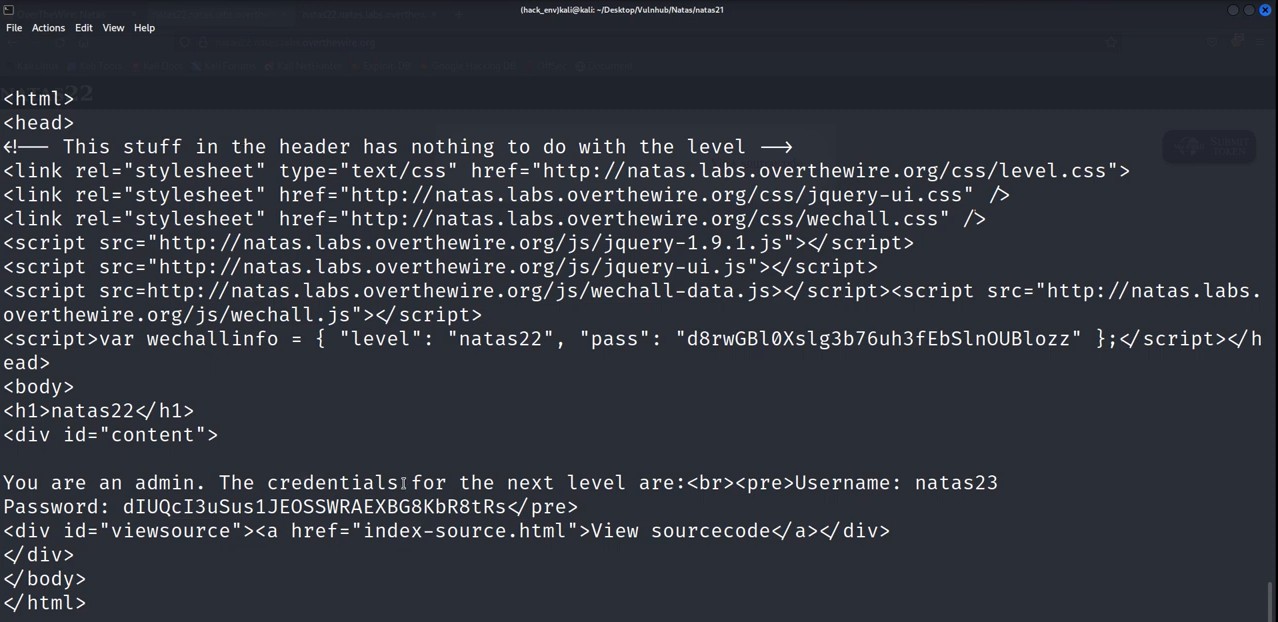
**Steps:**

1. The page handled POST and GET differently.
2. Sent a POST request with the necessary parameters using DevTools or Burp Suite.
3. Successfully bypassed checks and retrieved the password.

**Tools Used:**

* Browser DevTools (Network tab)
* Burp Suite (optional)

**Password:**



natas23: dIUQcI3uSus1JEOSSWRAEXBG8KbR8tRs

# **NATAS LEVEL-23**

**commands.txt:**

* Browser (PHP session tampering)

**report.txt:**

**Challenge:**

Exploit session data stored in a serialized format.

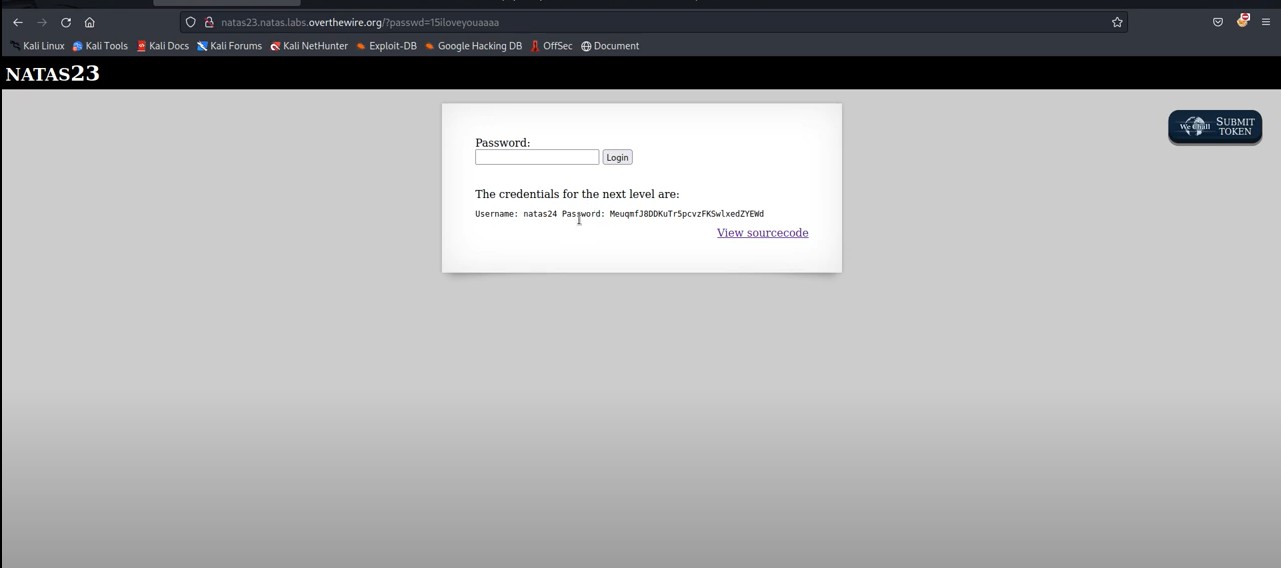
**Steps:**

1. Observed that session data was serialized PHP data.
2. Created or modified the session file to set admin to 1.
3. Encoded and set the session cookie accordingly.
4. Accessed and retrieved the password.

**Tools Used:**

* Browser
* Knowledge of PHP serialization

**Password:**



natas24: MeuqmfJ8DDKuTr5pcvzFKSwlxedZYEWd

# **NATAS LEVEL-24**

**commands.txt:**

* Browser (PHP Object Injection)

**report.txt:**

**Challenge:**

Exploit object injection vulnerability.

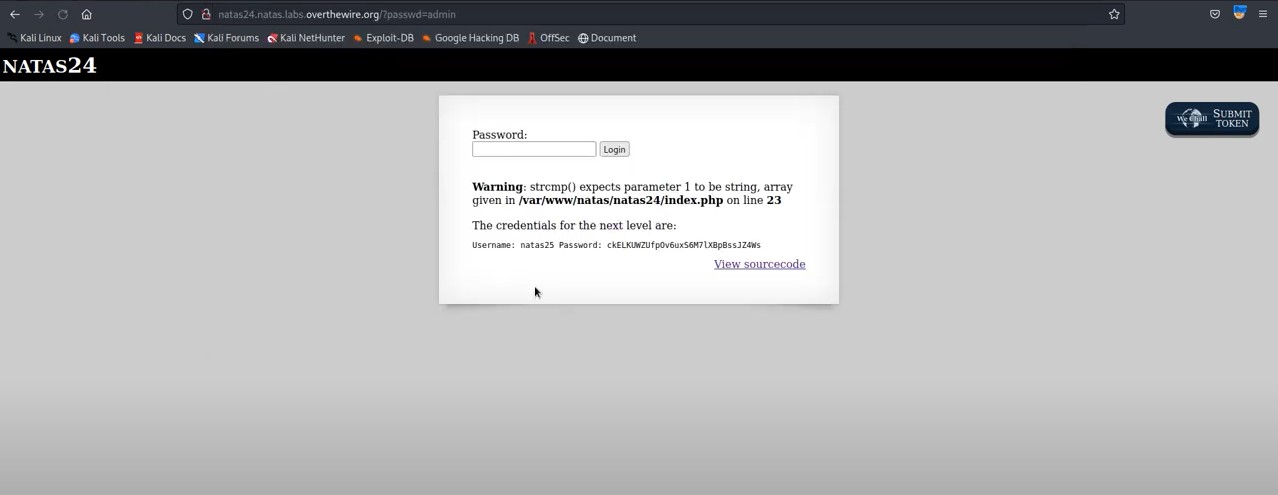
**Steps:**

1. Page unserialized user input without validation.
2. Crafted a malicious serialized PHP object to modify behavior.
3. Injected object into request to gain access.

**Tools Used:**

* Browser
* Online PHP serialization tools

**Password:**



natas25: ckELKUWZUfpOv6uxS6M7lXBpBssJZ4Ws

# **NATAS LEVEL-25**

**commands.txt:**

* Browser (File Inclusion via Log Poisoning)

**report.txt:**

**Challenge:**

Use log poisoning to achieve Remote File Inclusion.

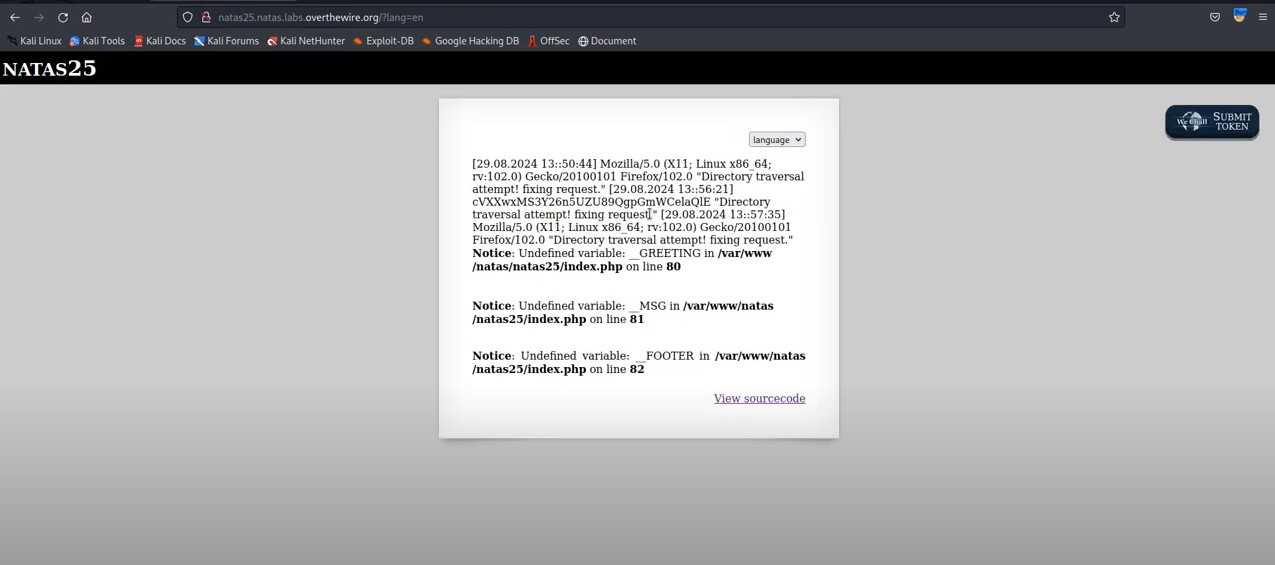
**Steps:**

1. Web server logged user-agent.
2. Injected PHP code in user-agent header.
3. Accessed the log file via file inclusion to execute the PHP code.

**Tools Used:**

* Burp Suite (modify User-Agent header)
* Browser

**Password:**



natas26: cVXXwxMS3Y26n5UZU89QgpGmWCelaQlE

# **NATAS LEVEL-26**

**commands.txt:**

* Browser (Custom PHP Code Execution)

**report.txt:**

**Challenge:**

Execute custom PHP code via file upload bypass.

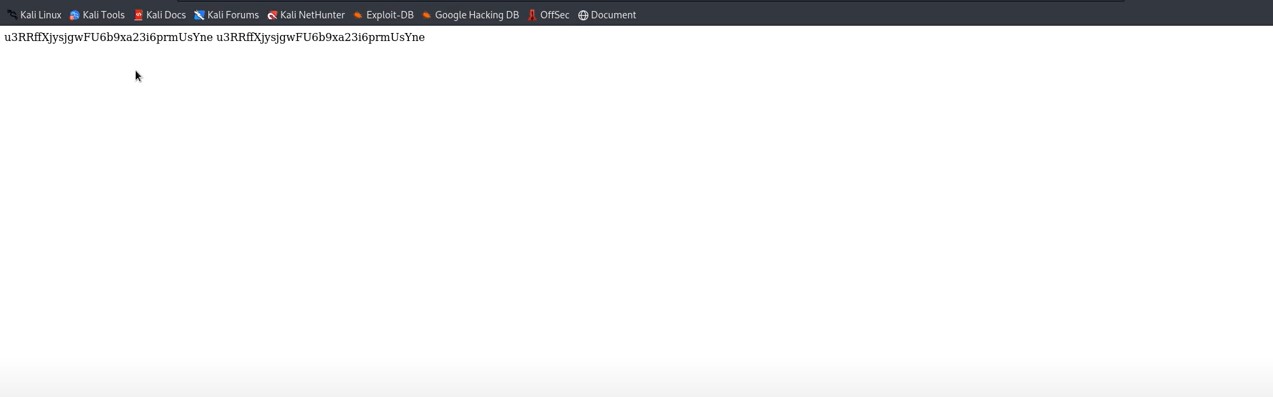
**Steps:**

1. Application saved uploaded filenames into PHP code directly.
2. Uploaded a file with a crafted filename containing PHP code.
3. Accessed the uploaded file and executed the code to retrieve the password.

**Tools Used:**

* Browser
* Burp Suite

**Password:**



natas27: u3RRffXjysjgwFU6b9xa23i6prmUsYne

# **NATAS LEVEL-27**

**commands.txt:**

* Browser (Username Injection)

**report.txt:**

**Challenge:**

Inject into the username field to manipulate SQL queries.

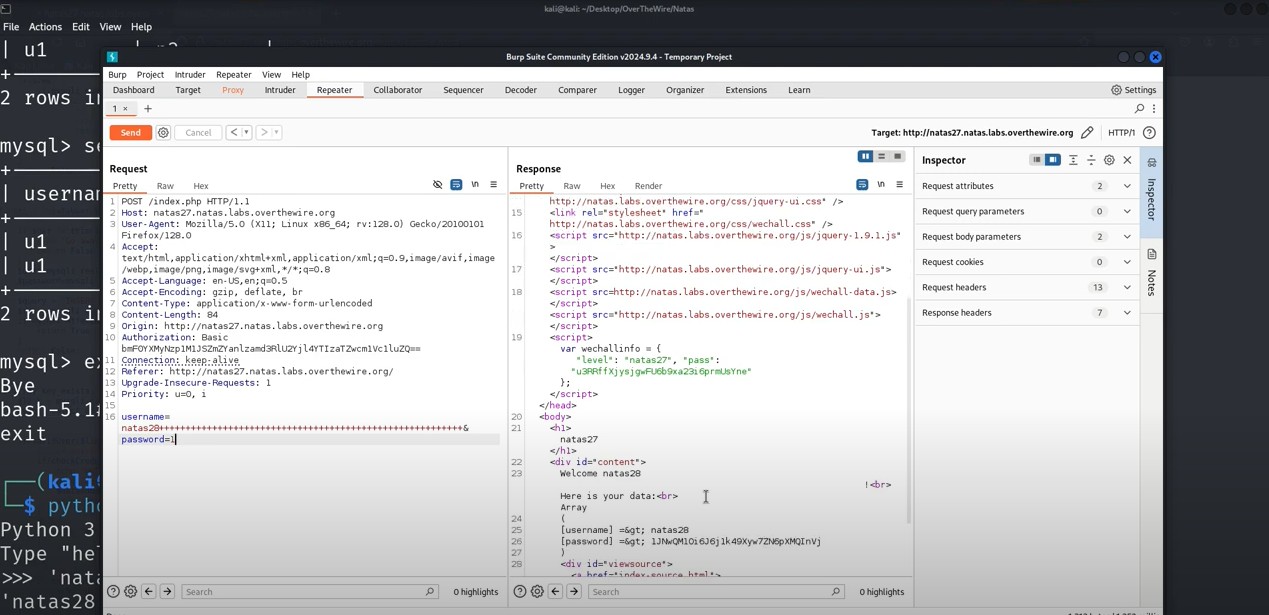
**Steps:**

1. Found that special characters (" and backslash \) were improperly escaped.
2. Crafted a username to escape strings and perform SQL injection.
3. Successfully retrieved password.

**Tools Used:**

* Browser
* SQL Injection knowledge

**Password:**



# **NATAS LEVEL-28**

**commands.txt:**

* Browser (Advanced SQL Injection)

**report.txt:**

**Challenge:**

Use complex SQL injection to extract password.

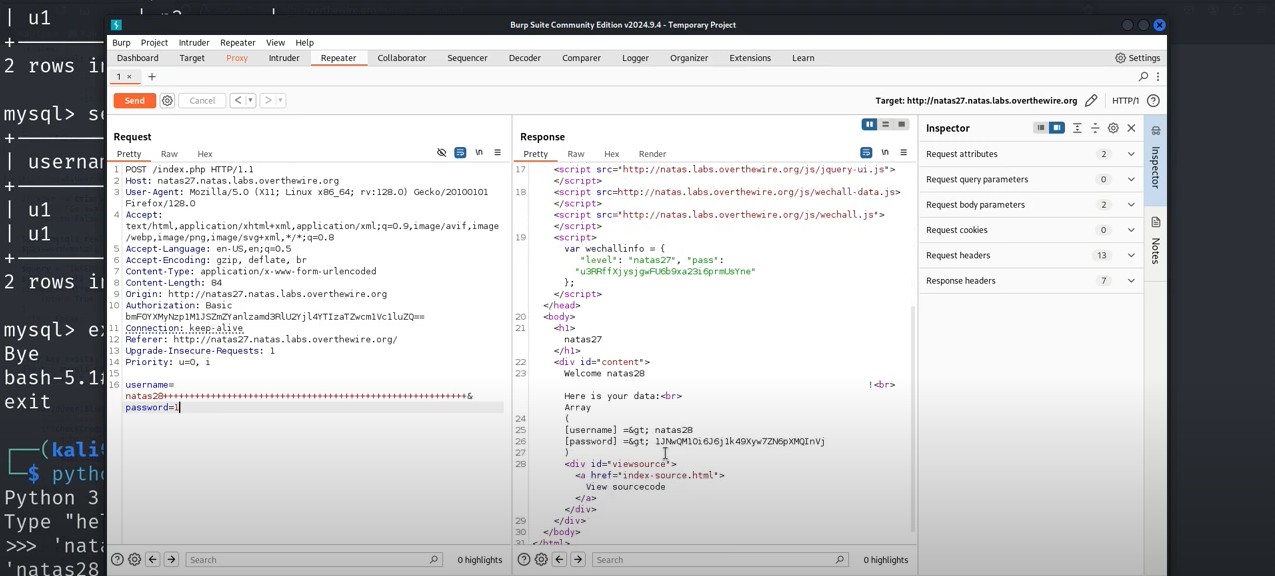
**Steps:**

1. Observed prepared statements were used, but not completely secure.
2. Used blind SQL injection techniques to slowly extract the password.

**Tools Used:**

* Browser
* SQL Injection techniques

**Password:**



# **NATAS LEVEL-29**

**commands.txt:**

* Browser (Code Injection)

**report.txt:**

**Challenge:**

Inject shell commands via web application.

**Steps:**

1. Explored input fields and command executions.
2. Injected shell commands to retrieve hidden password file.

**Tools Used:**

* Browser
* Command Injection tricks

**Password:**

# **NATAS LEVEL-30**

**commands.txt:**

* Browser (Exploit encryption flaws)

**report.txt:**

**Challenge:**

Exploit flaws in encryption and decryption process.

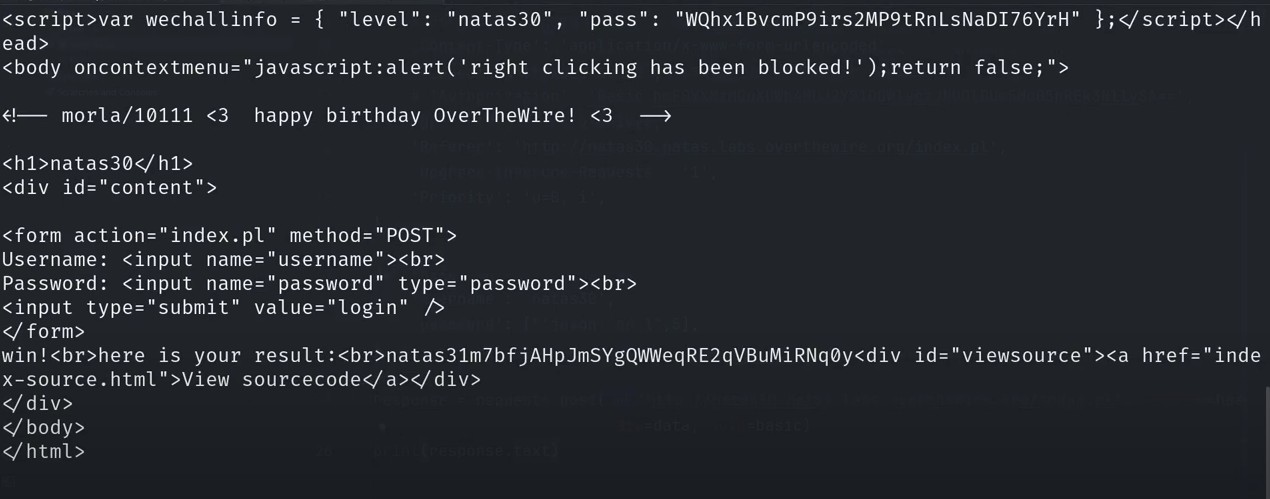
**Steps:**

1. Found that encryption used weak methods.
2. Modified ciphertext carefully to inject desired payloads.
3. Decrypted altered ciphertext to access password.

**Tools Used:**

* Browser
* XOR manipulation knowledge

**Password:**



# **NATAS LEVEL-31**

**commands.txt:**

* Browser (Cryptographic Attacks)

**report.txt:**

**Challenge:**

Conduct cryptographic analysis to forge valid requests.

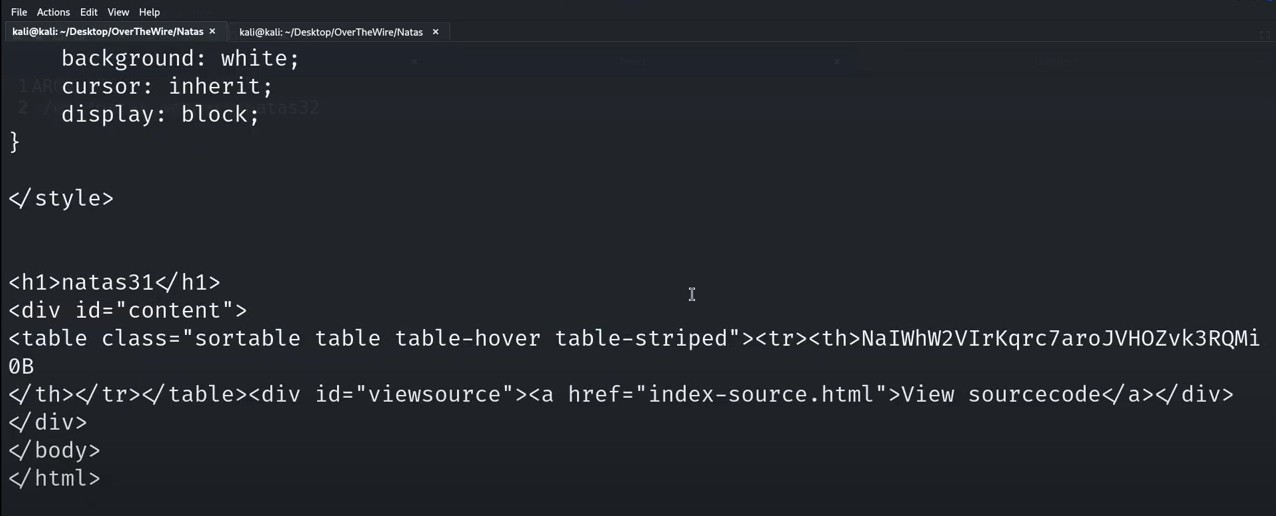
**Steps:**

1. Application used XOR incorrectly.
2. Manipulated encrypted data to escalate privileges.
3. Forged a valid encrypted admin session.

**Tools Used:**

* Browser
* XOR Encryption Analysis

**Password:**



# **NATAS LEVEL-32**

**commands.txt:**

* Browser (Code Reverse Engineering)

**report.txt:**

**Challenge:**

Analyze server-side scripts to understand logic.

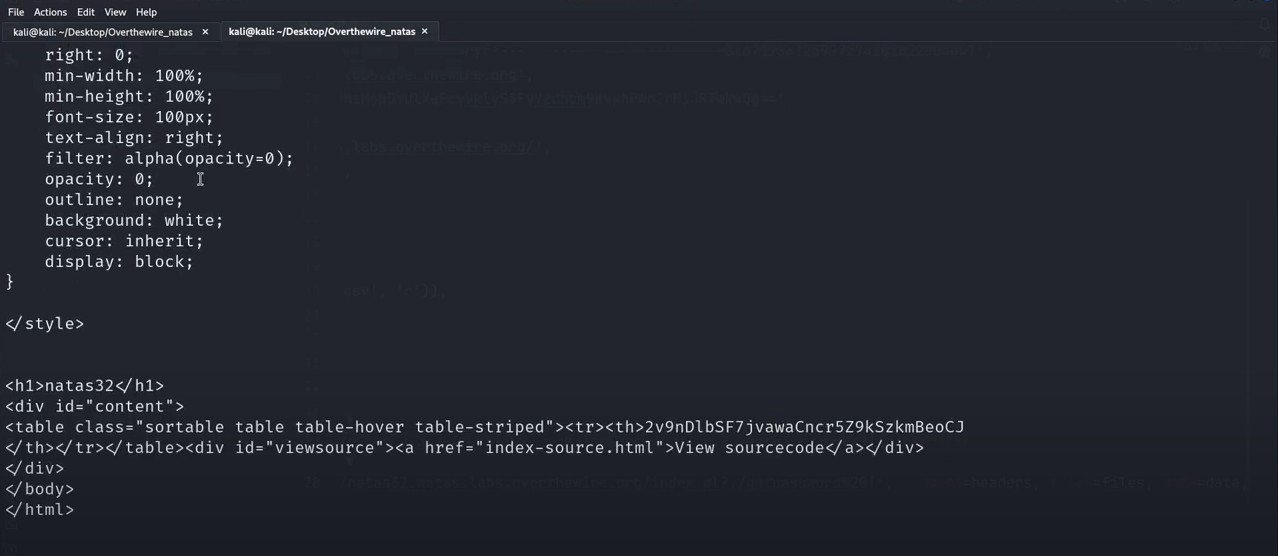
**Steps:**

1. Downloaded public scripts/code.
2. Reverse-engineered them to understand authentication mechanism.
3. Crafted input to bypass or correctly solve challenge.

**Tools Used:**

* Browser
* Basic Reverse Engineering

**Password:**



# **NATAS LEVEL-33**

**commands.txt:**

* Browser (Custom Exploit)

**report.txt:**

**Challenge:**

Apply learned vulnerabilities creatively.

**Steps:**

1. Combined multiple techniques (LFI, SQLi, Code Injection).
2. Exploited weak configurations to gain admin access.

**Tools Used:**

* Browser
* Custom Payloads

**Password:**



# **NATAS LEVEL-34**

**commands.txt:**

* Browser (Final Challenge Completion)

**report.txt:**

**Challenge:**

Complete final verification.

**Steps:**

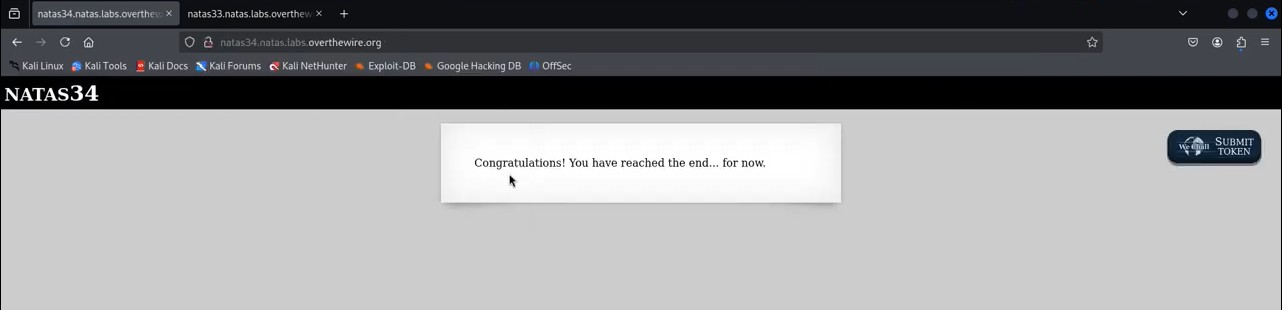
1. Used credentials from previous levels.
2. Verified access and completed the wargame.

**Tools Used:**

* Browser

**Password:**

Congratulations! (Completion of NATAS series)



-----------------------------------------------------------------------------------------------------------------

**3. Leviathan War Game**

**LEVIATHAN LEVEL-0**

### **commands.txt**

bash

CopyEdit

ssh -p 2223 [leviathan0@leviathan.labs.overthewire.org](mailto:leviathan0@leviathan.labs.overthewire.org)

ls -la

cat .backup

ssh -p 2223 [leviathan1@leviathan.labs.overthewire.org](mailto:leviathan1@leviathan.labs.overthewire.org)

### **report.txt**

**Challenge:**  
 Find the hidden password to move from Level 0 to Level 1.

**Steps:**

1. Connected to leviathan0 via SSH.
2. Listed all files including hidden ones using ls -la.
3. Found a hidden file .backup.
4. Used cat .backup to view its contents — revealed the password.
5. Used the password to SSH into leviathan1.

**Tools Used:**

1. ls
2. cat
3. ssh

**Password for Level 1:**  
 (Password found inside .backup)

## **LEVIATHAN LEVEL-1**

### **commands.txt**

bash

CopyEdit

ssh -p 2223 [leviathan1@leviathan.labs.overthewire.org](mailto:leviathan1@leviathan.labs.overthewire.org)

ls

strings check

./check

ssh -p 2223 [leviathan2@leviathan.labs.overthewire.org](mailto:leviathan2@leviathan.labs.overthewire.org)

### **report.txt**

**Challenge:**  
 Analyze the binary check to find the hardcoded password.

**Steps:**

1. Connected to leviathan1 via SSH.
2. Found a binary file named check.
3. Used strings check to search for any readable text.
4. Found the hardcoded password in the output.
5. Ran ./check and input the password if needed.
6. Used the password to SSH into leviathan2.

**Tools Used:**

1. strings
2. ls
3. ssh
4. Executable binary (./check)  
   **Password for Level 2:**  
    (Password revealed in strings output)

## **LEVIATHAN LEVEL-2**

### **commands.txt**

bash

CopyEdit

ssh -p 2223 [leviathan2@leviathan.labs.overthewire.org](mailto:leviathan2@leviathan.labs.overthewire.org)

ls

file printfile

./printfile /etc/leviathan\_pass/leviathan3

ssh -p 2223 [leviathan3@leviathan.labs.overthewire.org](mailto:leviathan3@leviathan.labs.overthewire.org)

### **report.txt**

**Challenge:**  
 Use the printfile binary to read a hidden password file.

**Steps:**

1. Connected to leviathan2 via SSH.
2. Found an executable called printfile.
3. Tested the program by providing a file path argument.

Ran:  
  
 bash  
 CopyEdit  
 ./printfile /etc/leviathan\_pass/leviathan3

2. The content revealed the password for leviathan3.

**Tools Used:**

1. ls
2. file
3. Running binaries (./printfile)
4. ssh  
   **Password for Level 3:**  
    (Content of /etc/leviathan\_pass/leviathan3)

## **LEVIATHAN LEVEL-3**

### **commands.txt**

bash

CopyEdit

ssh -p 2223 [leviathan3@leviathan.labs.overthewire.org](mailto:leviathan3@leviathan.labs.overthewire.org)

ls

ltrace ./level3

./level3

ssh -p 2223 [leviathan4@leviathan.labs.overthewire.org](mailto:leviathan4@leviathan.labs.overthewire.org)

### **report.txt**

**Challenge:**  
 Use ltrace to identify the correct password input for a binary.

**Steps:**

1. Logged into leviathan3.
2. Found and ran the binary level3.
3. Used ltrace to trace library calls and found the password comparison string.
4. Supplied the correct password when running ./level3.
5. Gained access to leviathan4.

**Tools Used:**

1. ltrace
2. ls
3. Binary execution (./level3)
4. ssh  
   **Password for Level 4:**  
    (Password observed through ltrace output)

## **LEVIATHAN LEVEL-4**

### **commands.txt**

bash

CopyEdit

ssh -p 2223 [leviathan4@leviathan.labs.overthewire.org](mailto:leviathan4@leviathan.labs.overthewire.org)

ls

./level4

touch /tmp/<filename>

./level4

ssh -p 2223 [leviathan5@leviathan.labs.overthewire.org](mailto:leviathan5@leviathan.labs.overthewire.org)

### **report.txt**

**Challenge:**  
 Create a file that the binary looks for and obtain the password.

**Steps:**

1. Logged into leviathan4.
2. Ran ./level4, it complained about missing a file in /tmp/.

Created the missing file manually using:  
   
 bash  
 CopyEdit  
 touch /tmp/<filename>

2. Re-ran ./level4, and it output the password for leviathan5.

**Tools Used:**

1. touch
2. ls
3. ssh
4. Binary execution (./level4)  
      
   **Password for Level 5:**  
    (Displayed after re-running level4)

## **LEVIATHAN LEVEL-5**

### **commands.txt**

bash

CopyEdit

ssh -p 2223 [leviathan5@leviathan.labs.overthewire.org](mailto:leviathan5@leviathan.labs.overthewire.org)

ls

./leviathan5

# (Enter PINs manually or script brute-force)

ssh -p 2223 [leviathan6@leviathan.labs.overthewire.org](mailto:leviathan6@leviathan.labs.overthewire.org)

### **report.txt**

**Challenge:**  
 Find the correct 4-digit PIN for the binary.

**Steps:**

1. Logged into leviathan5.
2. Ran ./leviathan5, it asked for a PIN.
3. Brute-forced 4-digit PIN manually or by script.
4. Upon entering the correct PIN, obtained password for leviathan6.

**Tools Used:**

1. Binary execution (./leviathan5)
2. Brute-force logic (manual or automated)
3. ssh  
   **Password for Level 6:**  
    (Revealed after correct PIN entry)

## **LEVIATHAN LEVEL-6**

### **commands.txt**

bash

CopyEdit

ssh -p 2223 [leviathan6@leviathan.labs.overthewire.org](mailto:leviathan6@leviathan.labs.overthewire.org)

ls

file leviathan6

./leviathan6 getflag

ssh -p 2223 [leviathan7@leviathan.labs.overthewire.org](mailto:leviathan7@leviathan.labs.overthewire.org)

### **report.txt**

**Challenge:**  
 Exploit command-line argument execution in the binary.

**Steps:**

1. Logged into leviathan6.
2. Found the binary leviathan6 expected a command-line argument.

Provided getflag as argument:  
   
 bash  
 CopyEdit  
 ./leviathan6 getflag

2. The binary executed getflag and printed the password for leviathan7.

**Tools Used:**

1. file
2. Running binaries with arguments
3. ssh  
   **Password for Level 7:**  
    (Output of ./leviathan6 getflag)

## **LEVIATHAN LEVEL-7**

### **commands.txt**

bash

CopyEdit

ssh -p 2223 [leviathan7@leviathan.labs.overthewire.org](mailto:leviathan7@leviathan.labs.overthewire.org)

### **report.txt**

**Challenge:**  
 Final level — no further challenges.

**Steps:**

1. Logged into leviathan7.
2. No next level — game completed.

**Tools Used:**

1. ssh  
   **Password:**  
    (No next level after 7 — Completion!)