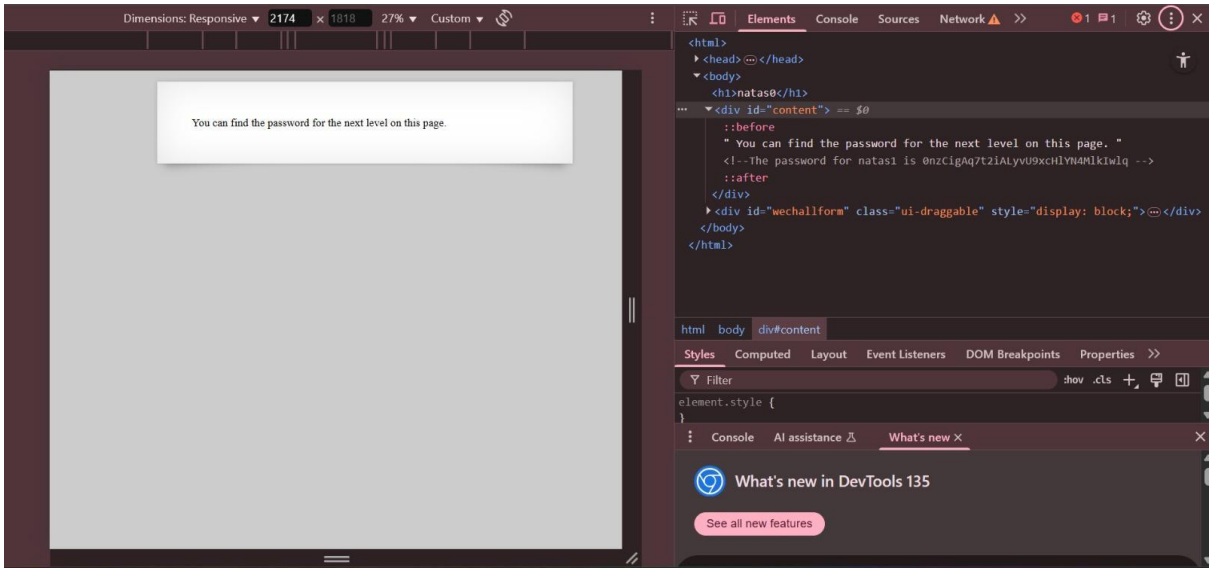
NATAS Wargame Walkthrough Report =>

Part 1: Natas0 to Natas17

Level: Natas0

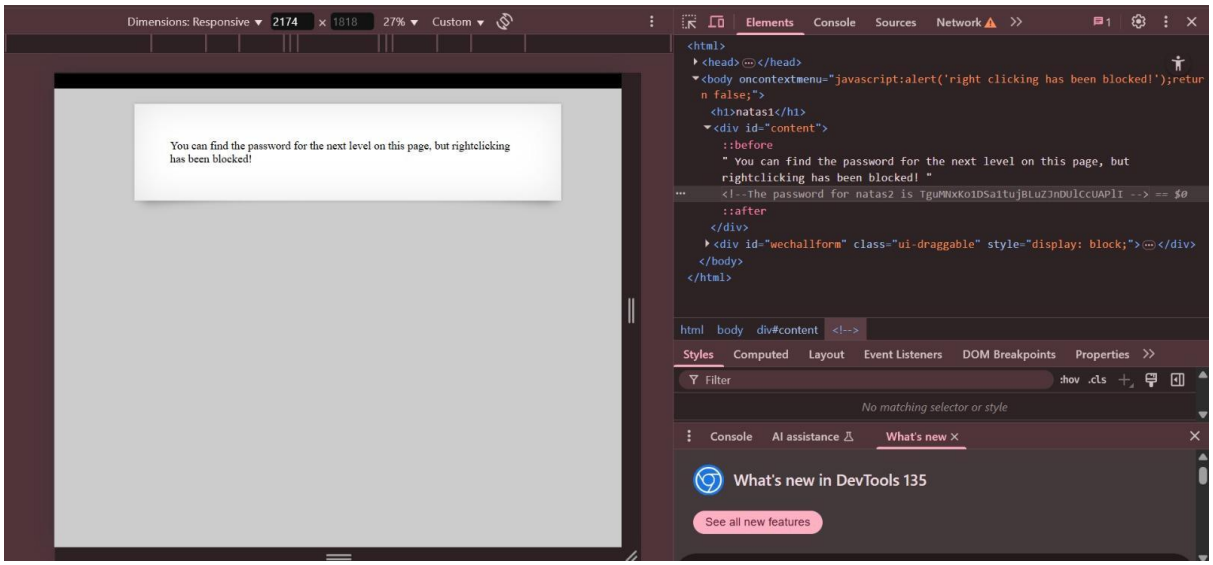
- **Step-by-Step:**
  - Open the URL in browser.
  - Notice username and password are given on the page.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - The first level is to teach you how to use HTTP basic authentication.



Level: Natas1

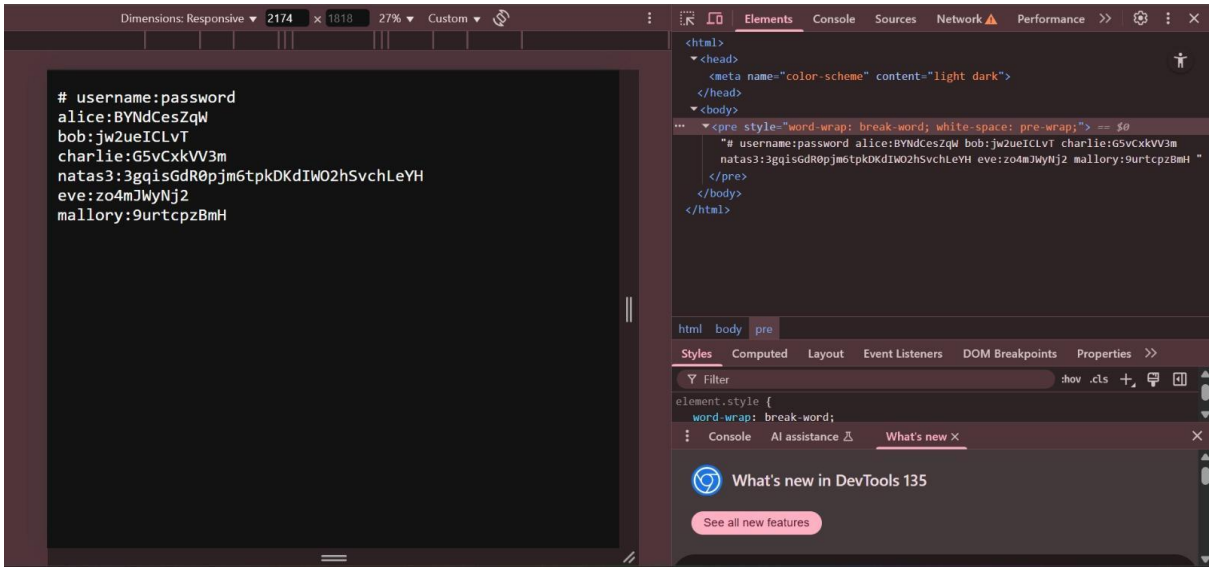
- **Step-by-Step:**
  - Open the URL in browser.
  - Right-click → View Page Source.
  - Find password hidden in a comment.
- **Tools Used:**
  - Browser

- **Logic Behind the Solution:**
    - Password hidden in HTML comments to teach checking page source.



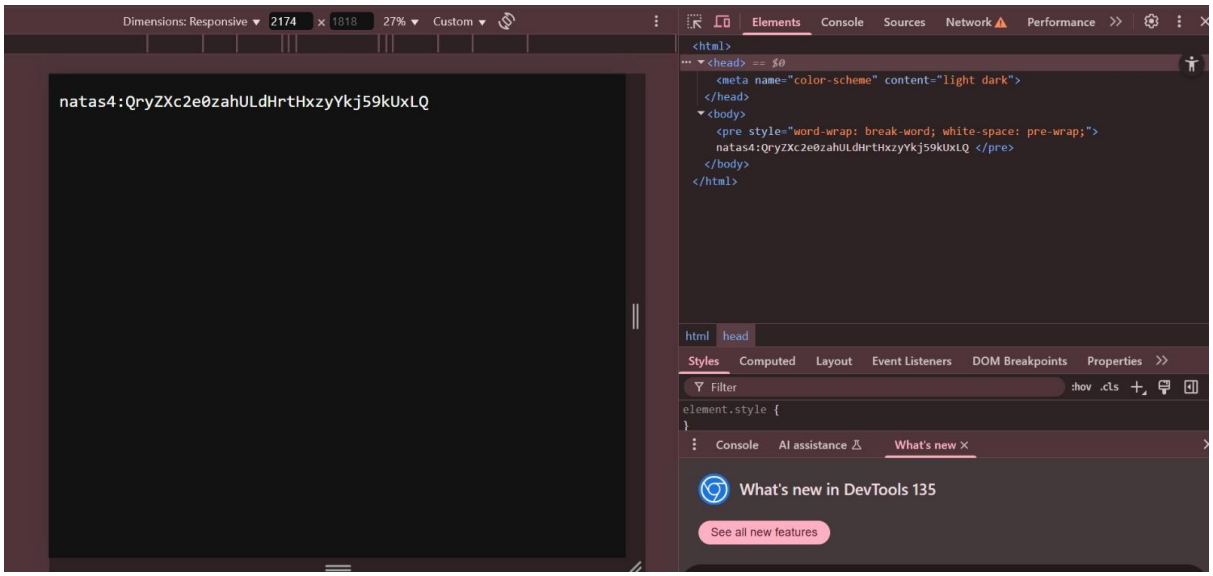
Level: Natas2

- **Step-by-Step:**
    - Open page and View Source.
    - Find a link to "/files/" directory.
    - Browse the directory and find the password file.
- **Tools Used:**
    - Browser
- **Logic Behind the Solution:**
    - Teaches exploring hidden directories.



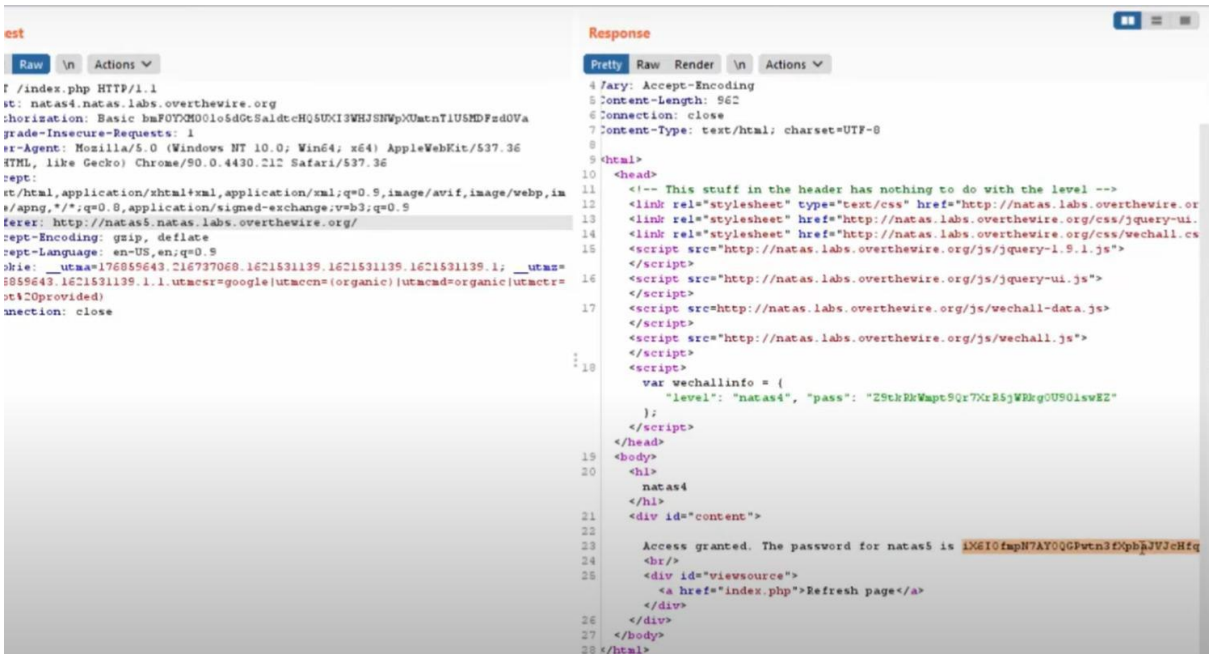
Level: Natas3

- **Step-by-Step:**
  - View Source.
  - Find hidden directory /s3cr3t/.
  - Find password inside it.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Train users to look carefully into source code.



Level: Natas4

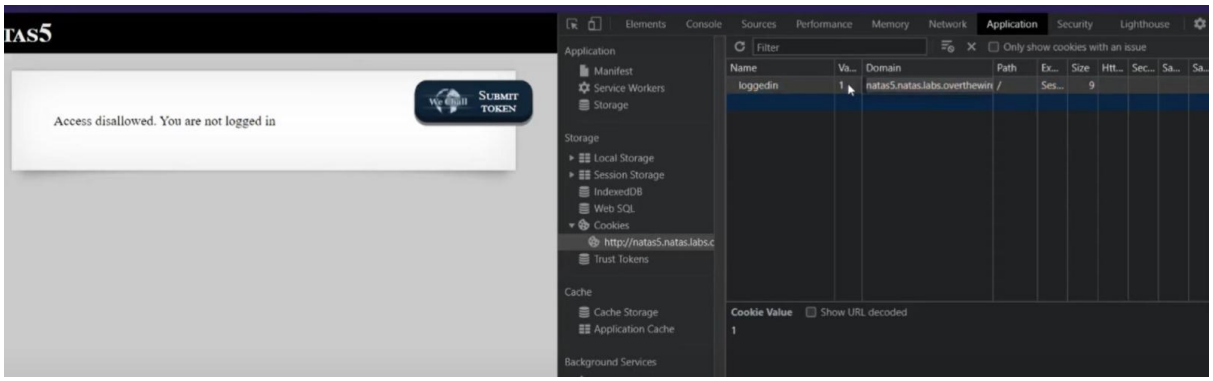
- **Step-by-Step:**
  - After accessing page, notice it redirects if 'Referer' is not set.
  - Manually set Referer header or use URL editing.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Introduction to HTTP headers (Referer).



Level: Natas5

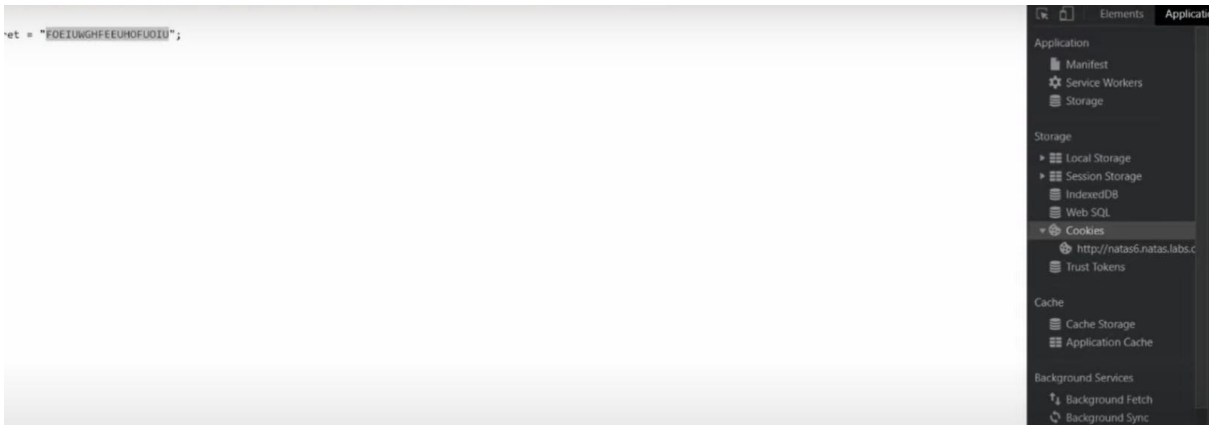
- **Step-by-Step:**
  - Inspect cookies.
  - Edit cookie 'loggedin' to 1.
- **Tools Used:**
  - Browser (Inspect Element)

- **Logic Behind the Solution:**
    - Teaches tampering with cookies.



Level: Natas6

- **Step-by-Step:**
    - View Source.
    - Find encoded secret (Base64).
    - Decode using base64.
- **Tools Used:**
    - Terminal (echo + base64) or online tools
- **Logic Behind the Solution:**
    - Understand basic encoding techniques.

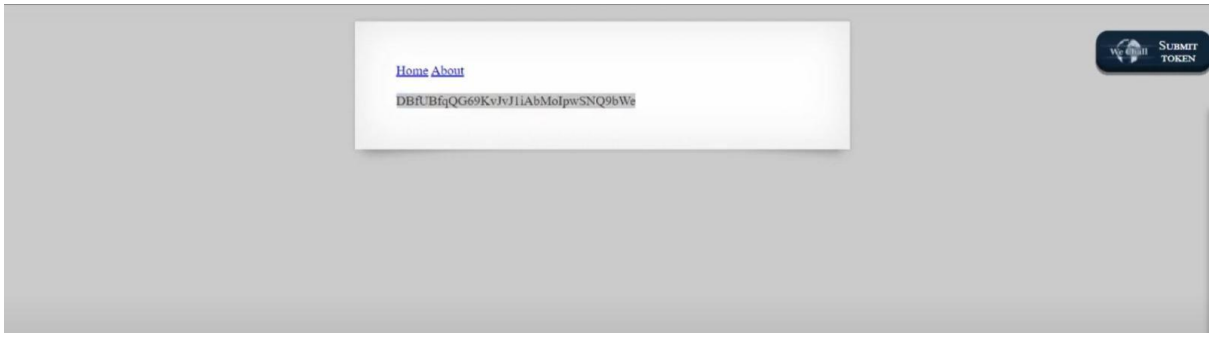


Level: Natas7

- **Step-by-Step:**
    - Modify URL parameter ?page=home.

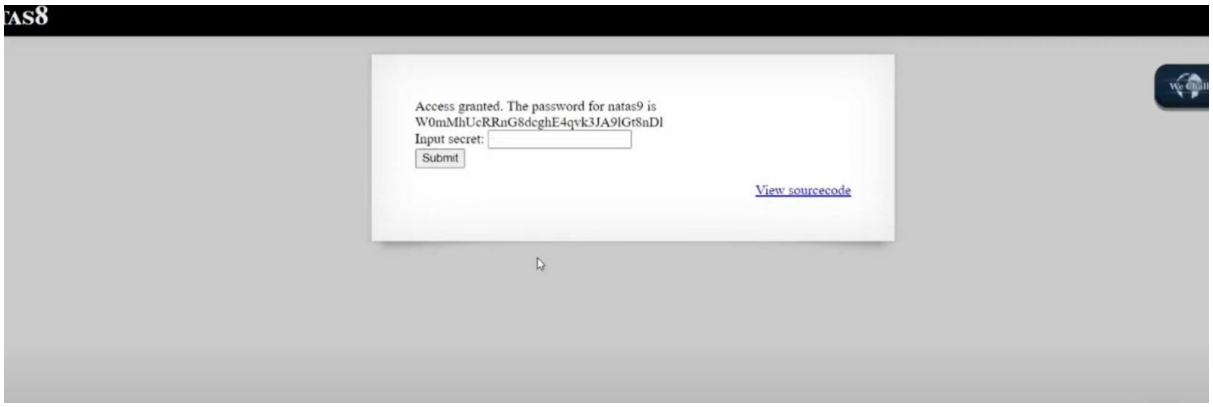
- Try Path Traversal with ?page=../../etc/natas_webpass/natas8.

- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Introduces basic path traversal.



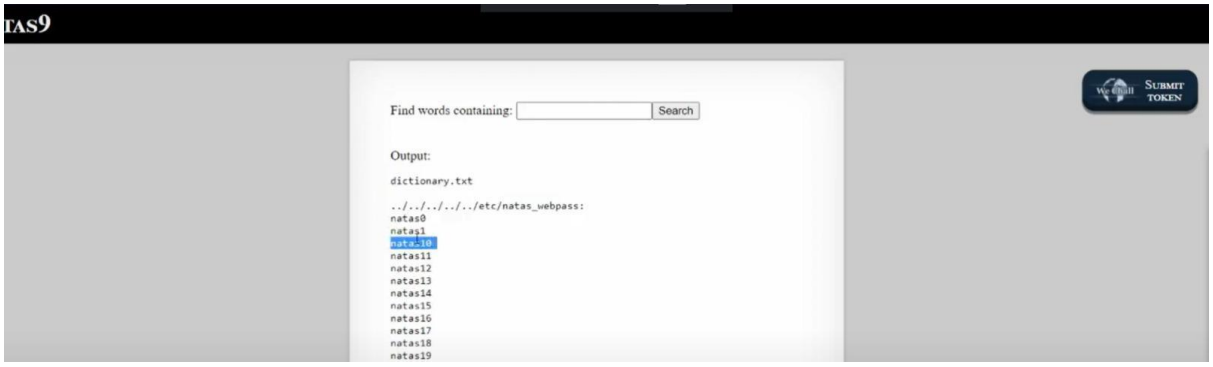
Level: Natas8

- **Step-by-Step:**
  - View Source.
  - Find custom hash function.
  - Reverse the logic with simple Python script.
- **Tools Used:**
  - Browser, Python
- **Logic Behind the Solution:**
  - Shows simple reversing of obfuscation.



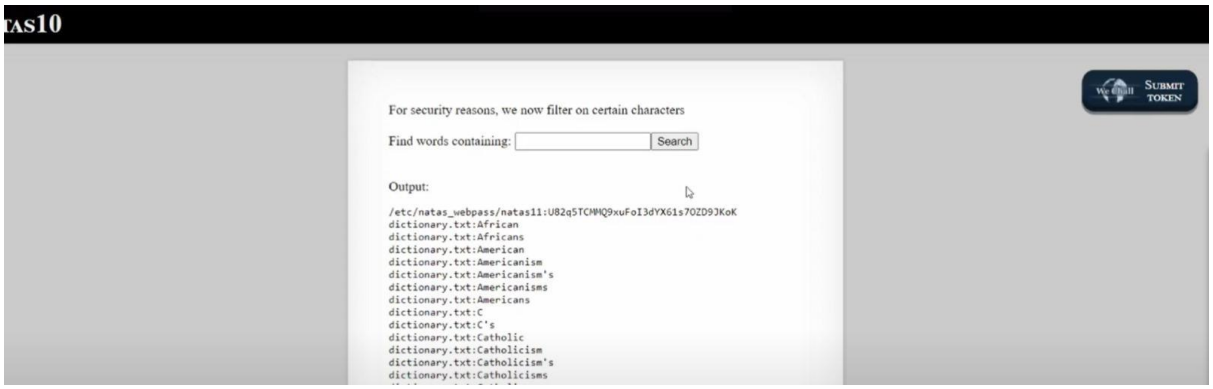
Level: Natas9

- **Step-by-Step:**
  - Input in search box: anytext; cat /etc/natas_webpass/natas10
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Introduces command injection.



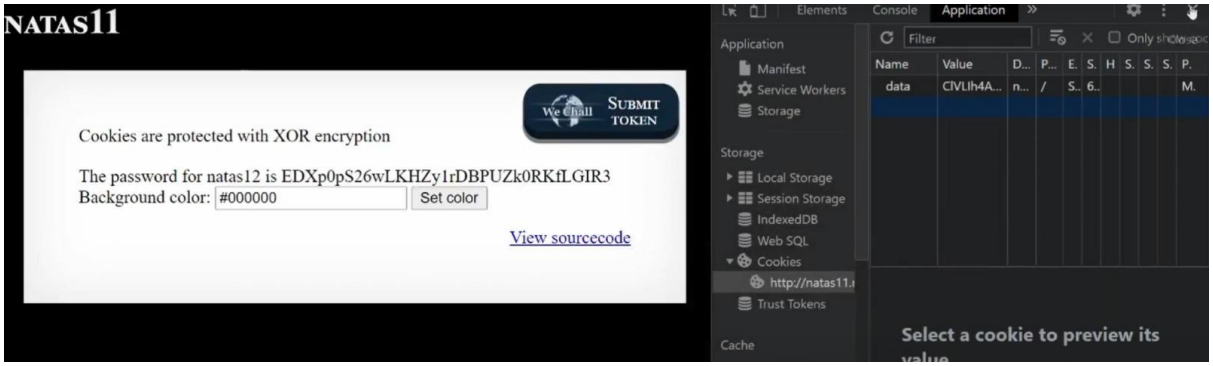
Level: Natas10

- **Step-by-Step:**
  - Input payload: anytext | cat /etc/natas_webpass/natas11
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Demonstrates using pipe | operator to inject commands.



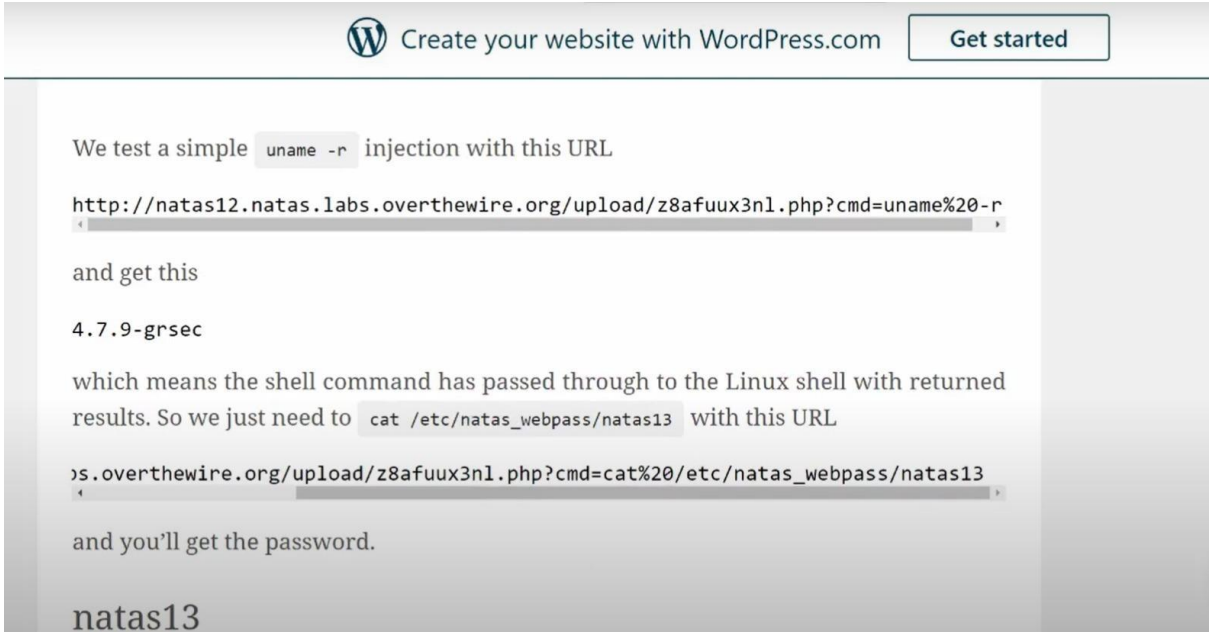
Level: Natas11

- **Step-by-Step:**
  - Decrypt cookie value.
  - Change isAdmin from false to true.
  - Re-encrypt cookie.
- **Tools Used:**
  - Terminal (openssl)
- **Logic Behind the Solution:**
  - Encryption understanding and cookie tampering.



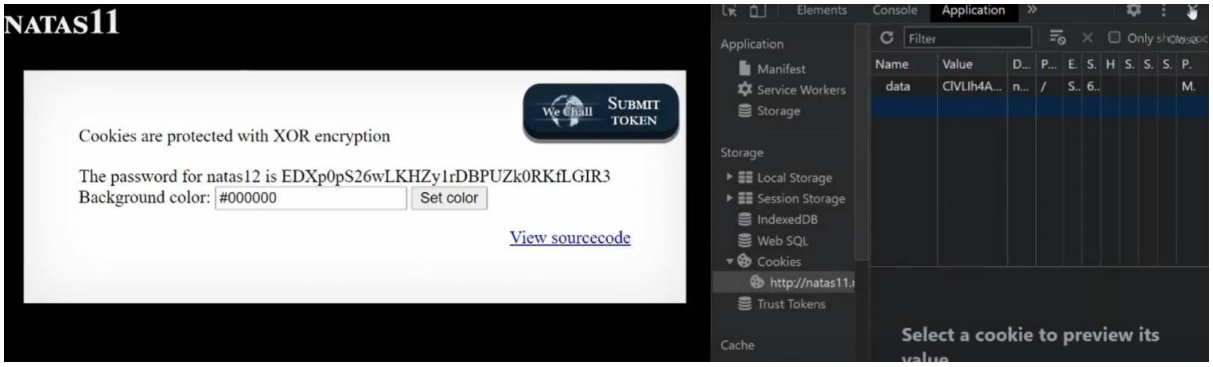
Level: Natas12

- **Step-by-Step:**
  - Upload PHP file disguised as an image.
  - Access uploaded PHP file.
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**
  - Bypassing file upload restrictions.



Level: Natas13

- **Step-by-Step:**
  - Upload a PHP file directly.
  - Execute uploaded file.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - File upload exploitation.

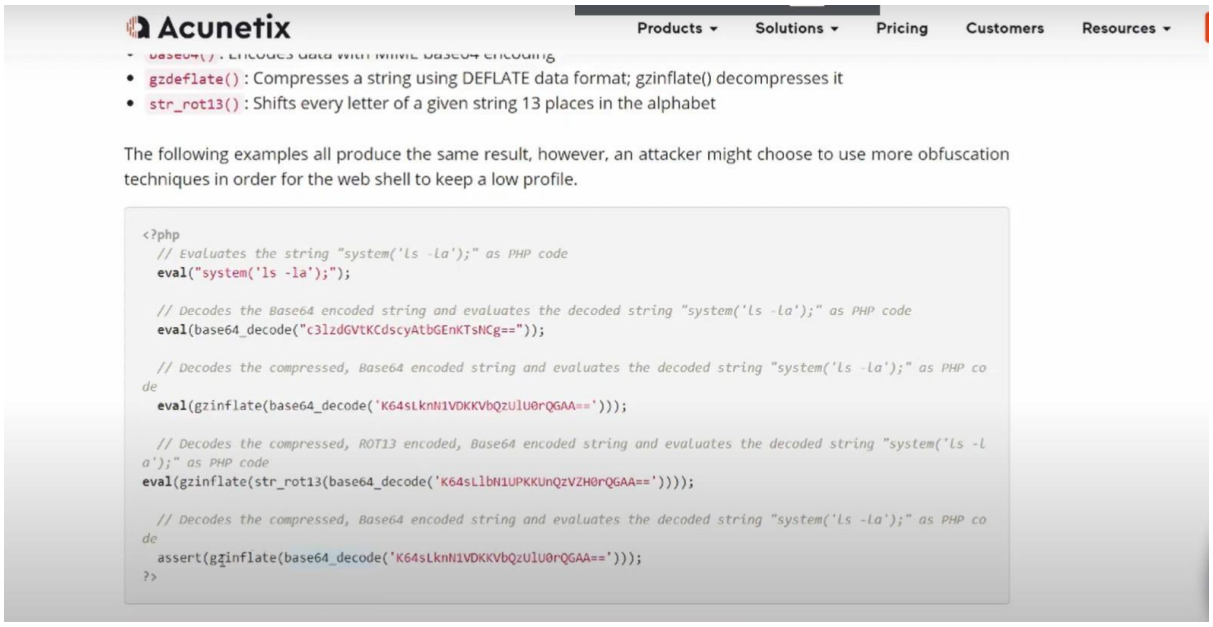


Level: Natas14

- **Step-by-Step:**
  - Use SQL Injection in login form:

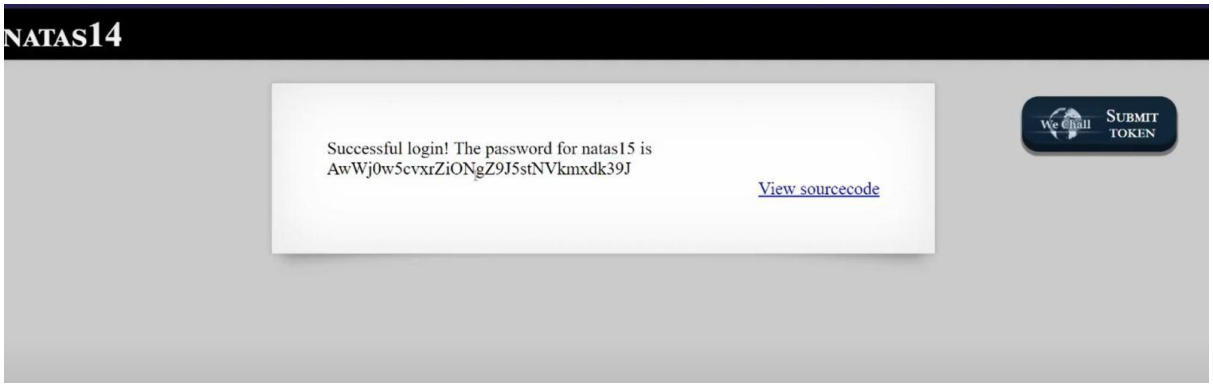
        - username: "natas15" OR "1"="1"

- **Tools Used:**
    - Browser
- **Logic Behind the Solution:**
    - Classic SQL Injection to bypass login.



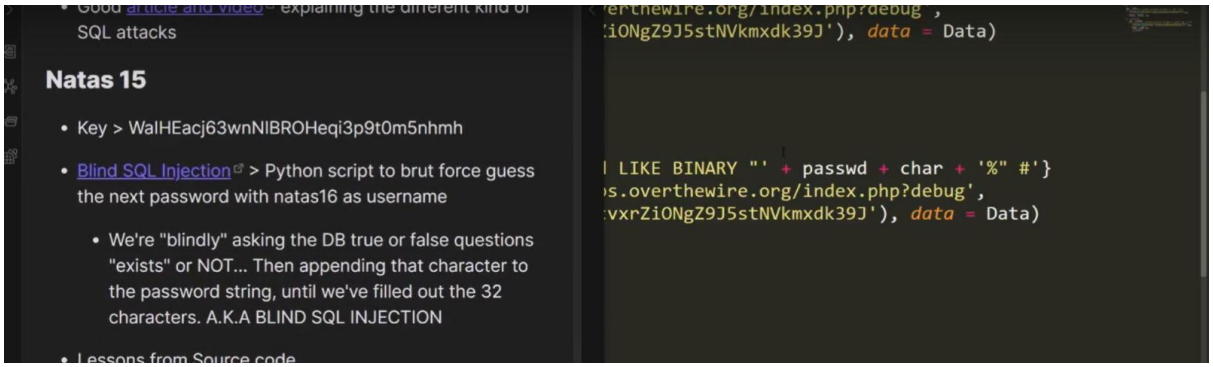
Level: Natas15

- **Step-by-Step:**
    - Use Blind SQL Injection guessing each character.
    - Automate using script or Burp Intruder.
- **Tools Used:**
    - Browser, Burp Suite, Script
- **Logic Behind the Solution:**
    - Blind SQL Injection attack using true/false responses.



Level: Natas16

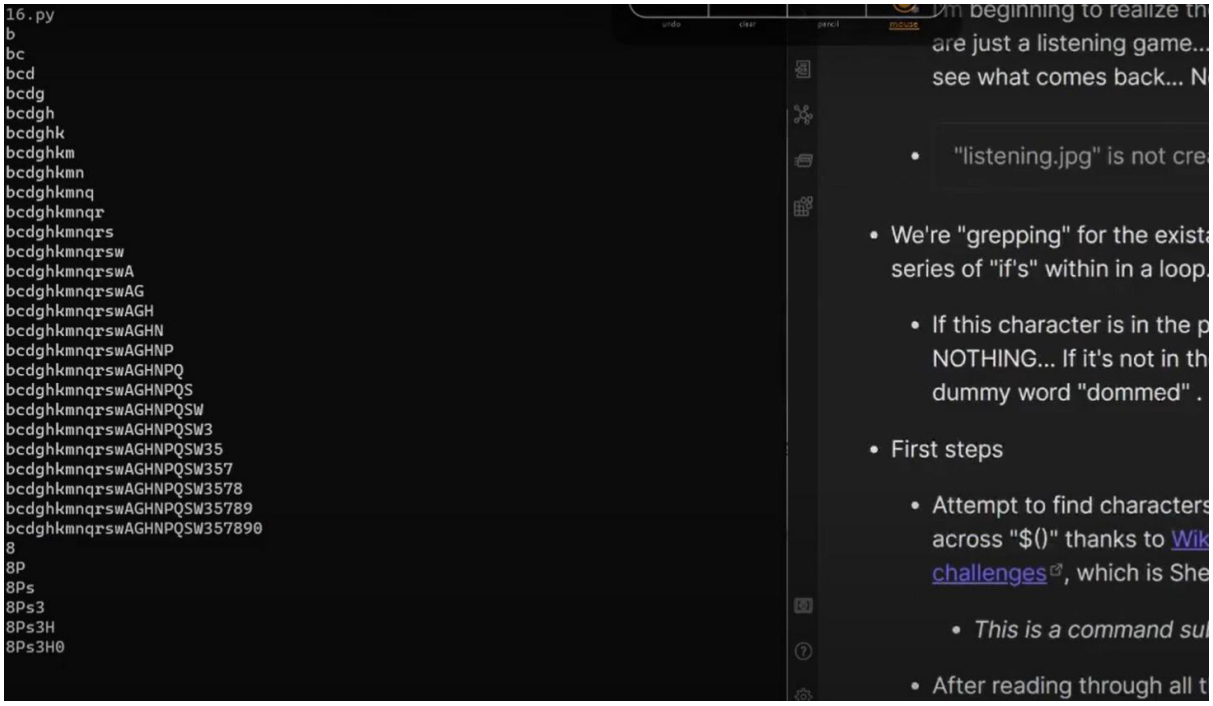
- **Step-by-Step:**
  - Inject command using | operator.
  - Example: anytext | cat /etc/natas_webpass/natas17
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - More advanced command injection practice.



Level: Natas17

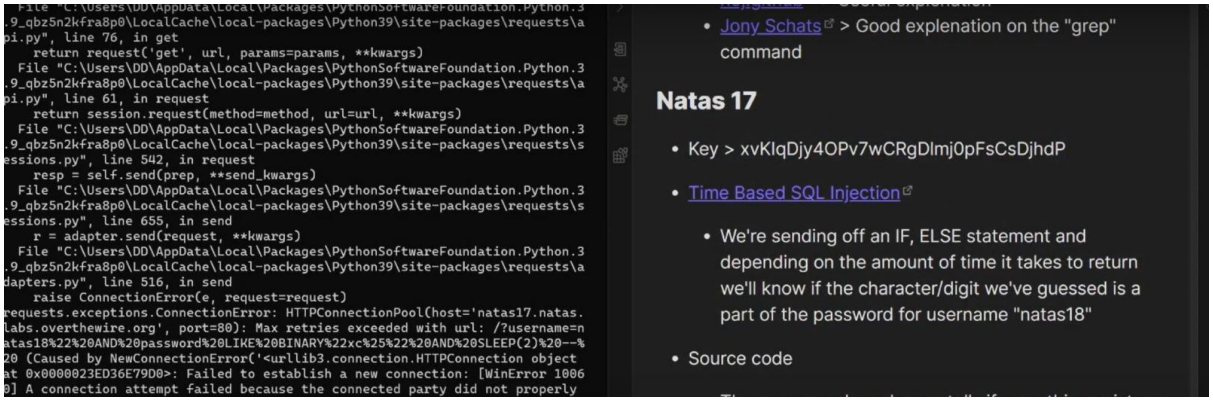
- **Step-by-Step:**
  - Perform Blind Command Injection.
  - Use time delay like SLEEP(5) to detect true condition.
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**

- Blind injection using response time.



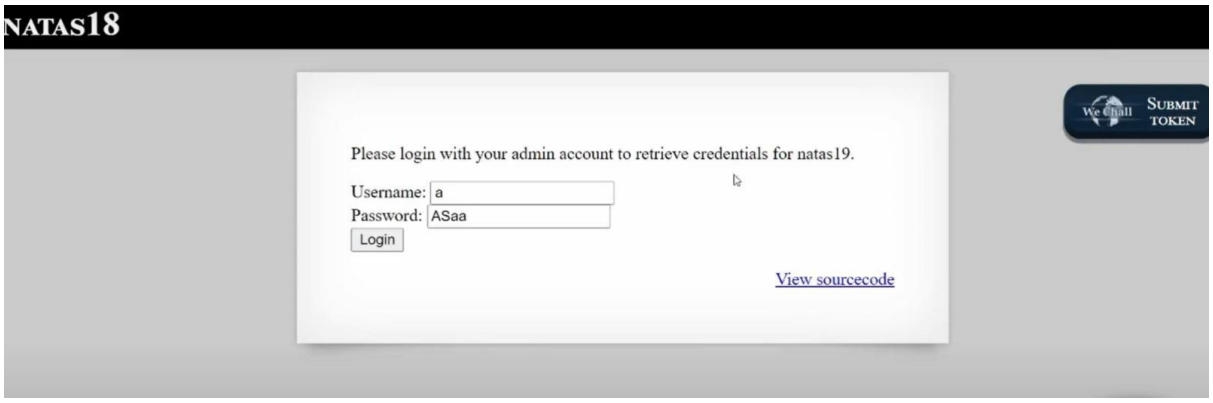
Level: Natas18

- **Step-by-Step:**
  - Brute-force session IDs from 1 to 640.
  - Find the session where admin=1.
- **Tools Used:**
  - Browser, bash loop, curl
- **Logic Behind the Solution:**
  - Exploit predictable session IDs.



Level: Natas19

- **Step-by-Step:**
  - Session ID is encoded in hexadecimal.
  - Brute-force with hex values.
- **Tools Used:**
  - Browser, bash script, curl
- **Logic Behind the Solution:**
  - Find the correct session by decoding hex session IDs.



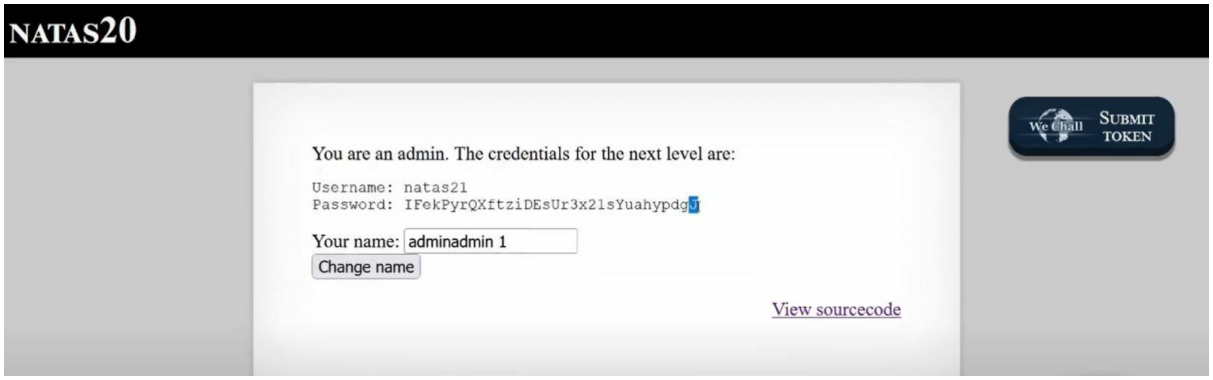
Level: Natas20

- **Step-by-Step:**
  - Modify POST parameters to set "debug" to 1.
  - Upload crafted text session manually.
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**
  - Session tampering and privilege escalation.

Level: Natas21

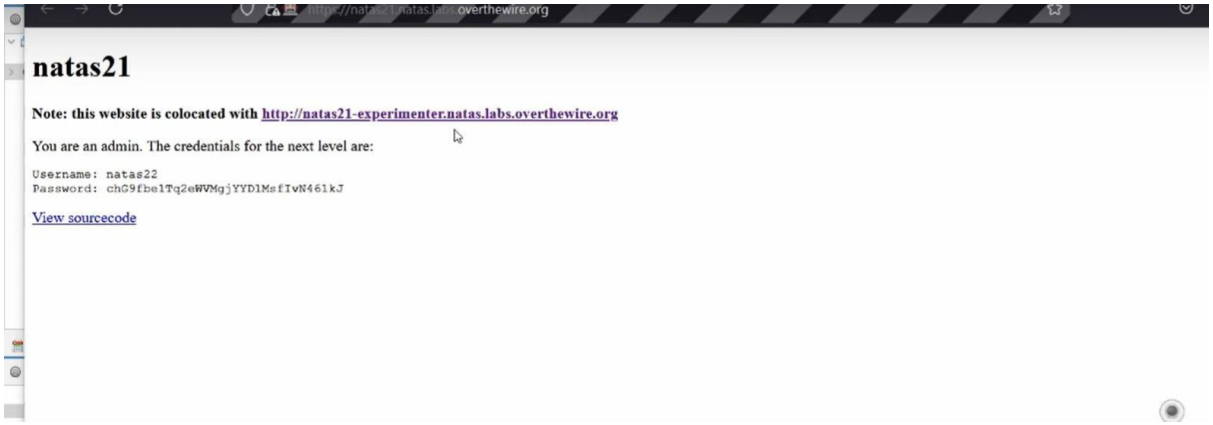
- **Step-by-Step:**
  - Two different subdomains handle different requests.
  - Modify session to "admin=1" manually.

- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**
  - Handling multiple sessions across subdomains.



Level: Natas22

- **Step-by-Step:**
  - The page redirects instantly.
  - Use curl -i to inspect HTTP headers and get the response before redirection.
- **Tools Used:**
  - curl
- **Logic Behind the Solution:**
  - HTTP redirection behavior exploitation.

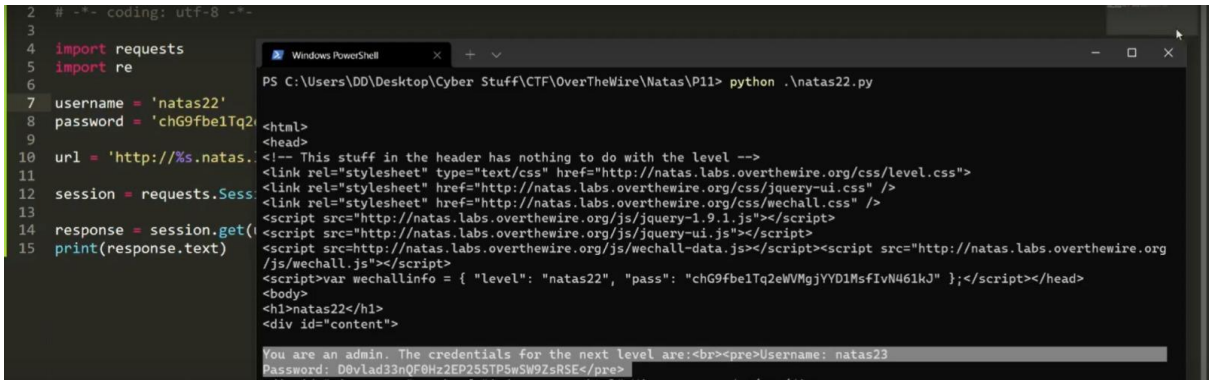


Level: Natas23

- **Step-by-Step:**

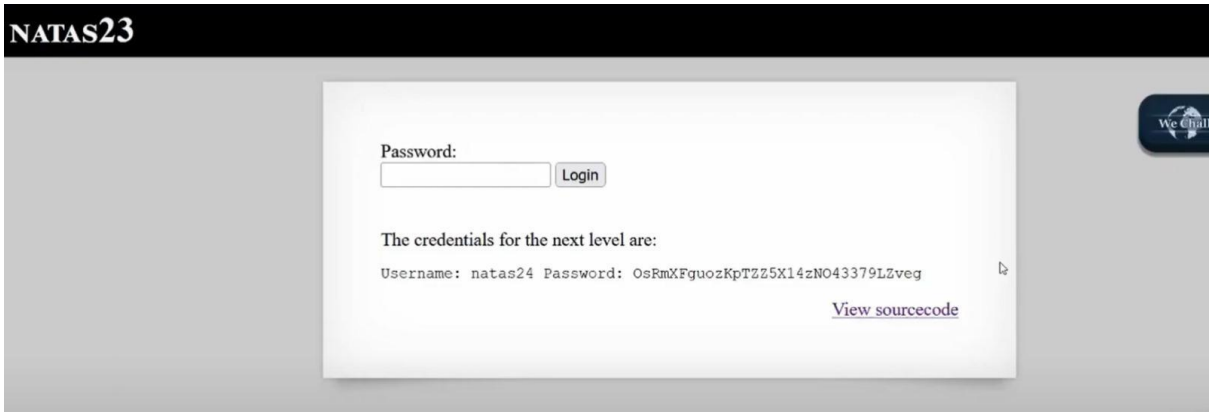
- View Page Source.
- Find the expected secret input.
- Submit the correct secret.

- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Simple logic puzzle based on input validation.



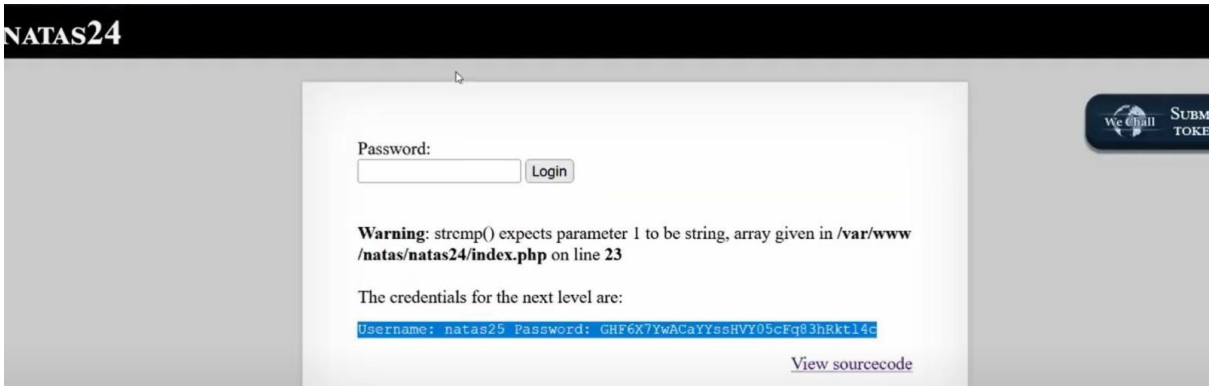
Level: Natas24

- **Step-by-Step:**
  - Inject command using POST parameters.
  - Example: "test$(cat /etc/natas_webpass/natas25)"
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Exploiting input parsing and command injection.



Level: Natas25

- **Step-by-Step:**
  - Perform directory traversal in the lang parameter.
  - Try multiple ../ to reach /etc/natas_webpass.
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Advanced path traversal attack.



Level: Natas26

- **Step-by-Step:**
  - Modify cookie that stores serialized object.
  - Decode, edit, re-encode using base64.
- **Tools Used:**
  - Browser, Python, PHP
- **Logic Behind the Solution:**
  - Object serialization manipulation.

```python
# -*- coding: utf-8 -*-

import requests
import re

username = 'natas25'
password = 'GHF6X7YwACaYYssHVY05cFq83hRktl4c'

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

session = requests.Session()

headers = {"User-Agent": "<?php system('cat /etc/natas_webpass/natas26'); ?>"}

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

response = session.post(url, headers = headers, data = {"lang" :
    "../../../../../../../../../../../var/www/natas/natas25/logs/natas25_"
    + session.cookies['PHPSESSID'] + ".log"}, auth = (username, password))
```

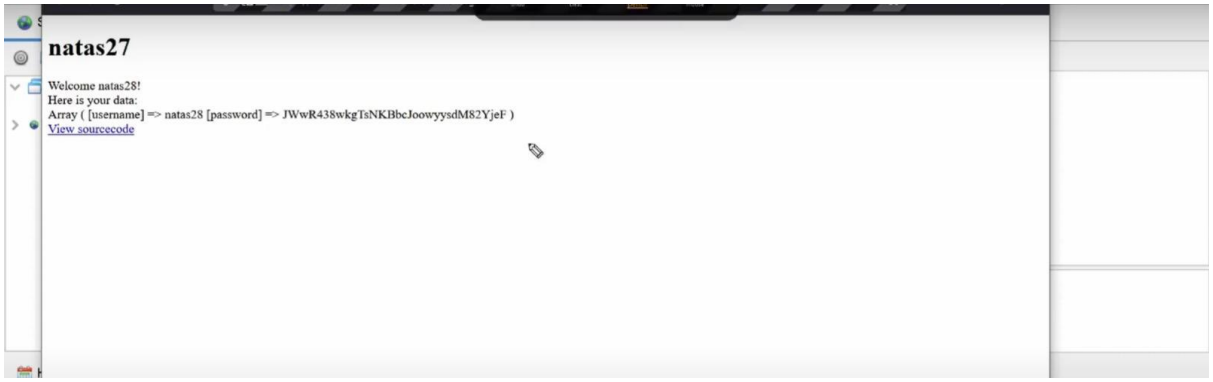
Level: Natas27

- **Step-by-Step:**
  - SQL Injection using case sensitivity.
  - Payload: ' UNION SELECT password FROM users WHERE username LIKE BINARY 'natas28' --
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Using "BINARY" keyword to perform case-sensitive queries.

55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ

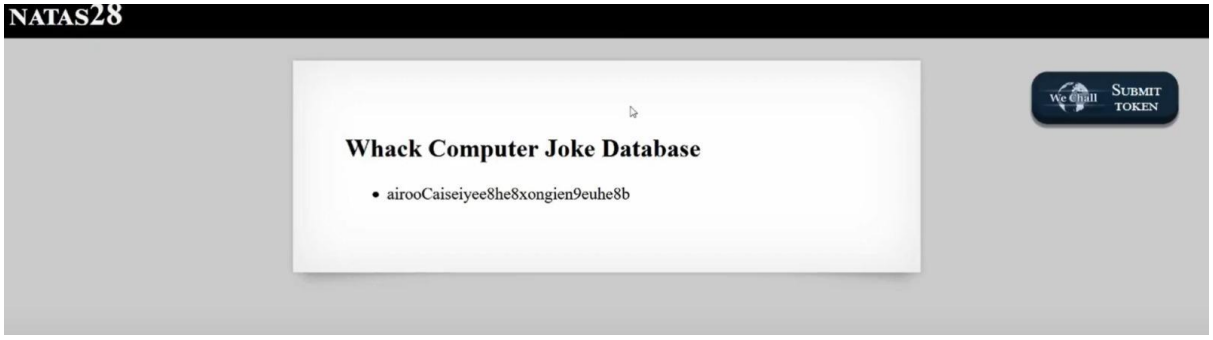
Level: Natas28

- **Step-by-Step:**
  - SQL Injection bypassing escaping techniques.
  - Use complicated payloads like ") UNION ALL SELECT password #
- **Tools Used:**
  - Browser
- **Logic Behind the Solution:**
  - Escaping characters properly to break query structure.



Level: Natas29

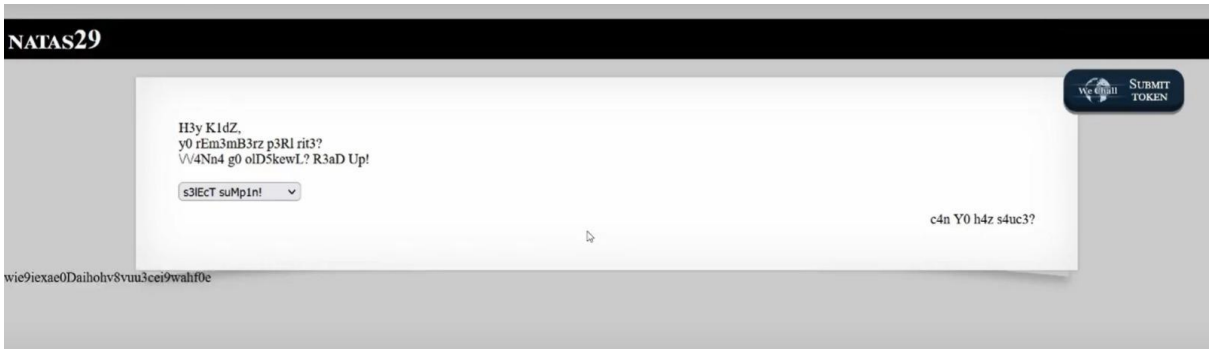
- **Step-by-Step:**
  - Understand serialized PHP object.
  - Craft malicious serialized object manually.
- **Tools Used:**
  - PHP scripting
- **Logic Behind the Solution:**
  - Abuse serialization to manipulate application behavior.



Level: Natas30

- **Step-by-Step:**
  - Send multiple parameters with the same name.
  - Example: passwd[]=123&passwd[]=123
- **Tools Used:**
  - Browser, Burp Suite
- **Logic Behind the Solution:**

- Exploit how PHP processes multiple same-named parameters.



Level: Natas31

- **Step-by-Step:**
  - Use multipart/form-data content type.
  - Submit crafted HTTP request via Burp Repeater.
- **Tools Used:**
  - Burp Suite
- **Logic Behind the Solution:**
  - Exploit how web apps parse file uploads differently.

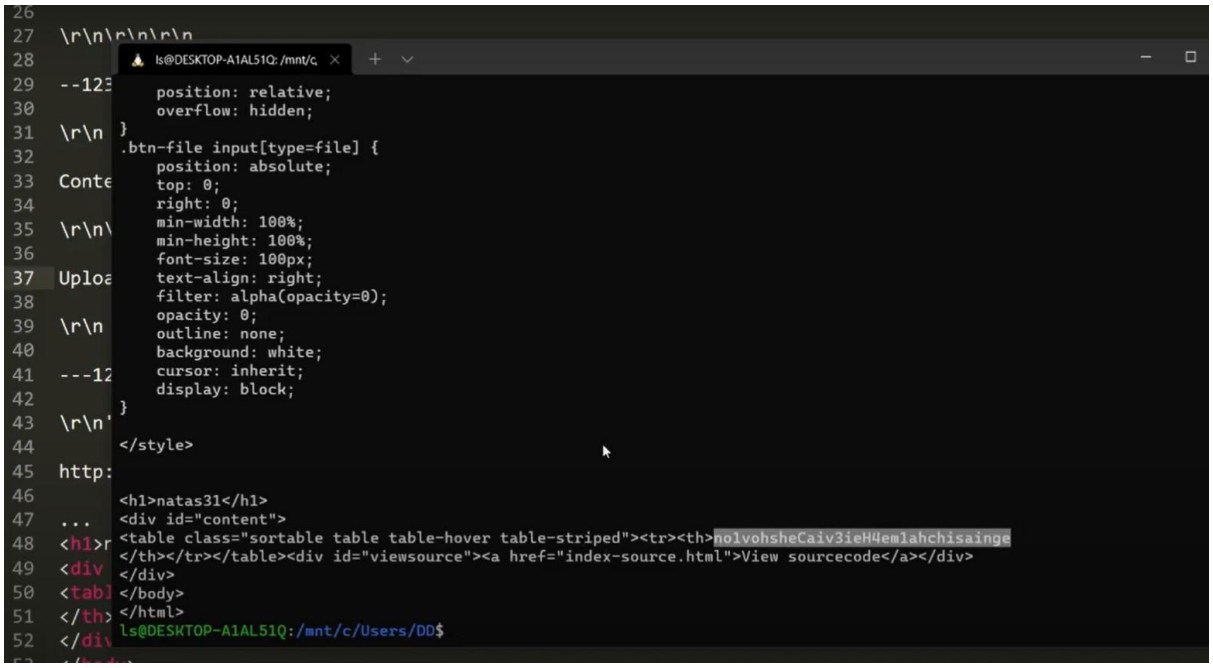
```
<!-- morla/10111 <3  happy birthday OverTheWire! <3  -->

<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>natas31hay7aecuungiuKaezuathuk9biin0pu1</div id="viewsource"><a href="ind
ex-source.html">View sourcecode</a></div>
</div>
</body>
</html>

PS C:\Users\DD\Desktop\Cyber Stuff\CTF\OverTheWire\Natas\P19>
```

Level: Natas32

- **Step-by-Step:**
  - Upload a malicious PHP file.
  - Wait for a cron job to execute it automatically.
- **Tools Used:**

- Browser

- **Logic Behind the Solution:**
  - Understand webserver and cron job timing attacks.



Level: Natas33

- **Step-by-Step:**
  - Use SQL Injection.
  - Payload: ' OR 1=1 --

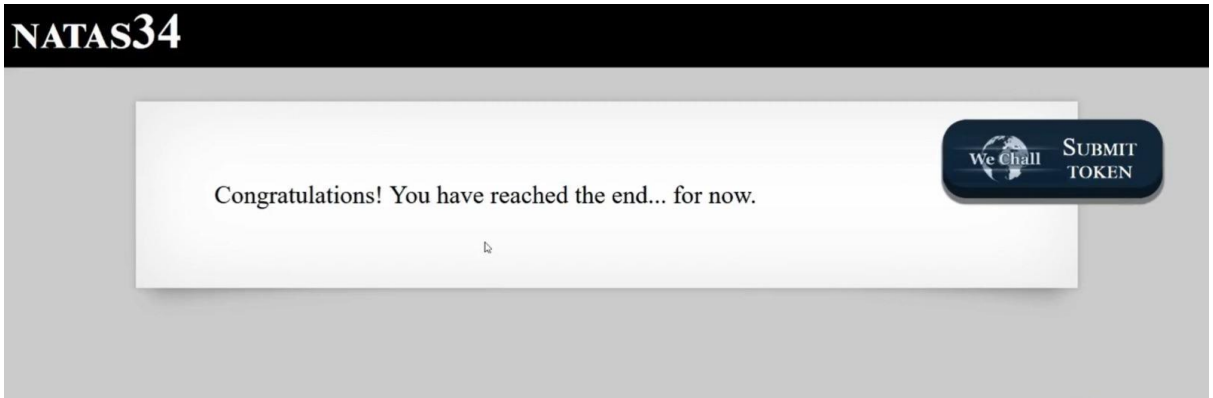
- **Tools Used:**
  - Browser

- **Logic Behind the Solution:**
  - Bypass login forms via always-true SQL queries.

```
<style>
#content {
    width: 900px;
}
.btn-file {
    position: relative;
    overflow: hidden;
}
.btn-file input[type=file] {
    position: absolute;
    top: 0;
    right: 0;
    min-width: 100%;
    min-height: 100%;
    font-size: 100px;
    text-align: right;
    filter: alpha(opacity=0);
    opacity: 0;
    outline: none;
    background: white;
    cursor: inherit;
    display: block;
}

</style>


<h1>natas32</h1>
<div id="content">
<table class="sortable table table-hover table-striped"><tr><th>shoogeiGa2yee3de6Aex8uaXeech5eey
</th></tr></table><div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
```

Level: Natas34

- **Step-by-Step:**
  - Decode JWT token.
  - Modify the payload (admin:true).
  - Re-sign with known key (or no verification if vulnerable).
- **Tools Used:**
  - jwt.io, Browser
- **Logic Behind the Solution:**
  - JWT forgery and authentication bypass.



Tools Commonly Used:

- Browser (View Source, Inspect, Edit Cookies)
- curl and bash scripts (for brute forcing)
- Burp Suite (for modifying requests)
- Online Tools (jwt.io, base64 decoders)
- Scripting languages (Python, PHP)
