

Level 12

I first looked at the source code and noticed these hidden form elements:

[illegible]

When writing the file to disk, the server generates a random name for the file however, the extension given to the file will be used to store our file on disk. So if I were to use the .php extension, it would execute the file as a PHP file.

I uploaded a file.php which contained the following code since we know that the password is stored in /etc/natas_webpass/natas13. It took me a while to figure how but I used my knowledge from the previous level, to use the passthru function and create the following php file:

[illegible]

Uploading this file displayed a link to the file which upon clicking, showed me the password.

jmLTY0qiPZBbaKc9341cqPQZBjv7MQbY

Level 14

This level involved an SQL injection attack.

I tried using brute force attacks to try several username and password combinations for the user 'natas15' since I want to gain access to that level. These were all unsuccessful login attempts.

I then viewed the source code and noted the highlighted area - this was an SQL query which took the username and password and checked that it returns atleast one row.

```
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 {
}
?>
```

At this point I spent a lot of time wondering how I can input my own variables for the SQL query and finally noticed a debug variable (\$GET). I knew I had to add this debug parameter to the URL but I didn't know how to so I searched for tools to add debug parameter until I stumbled on OWASP ZAP.



Once I added the debug parameter, I searched for information about SQL injection attacks and found out that we can use boolean statements to ensure that the database always returns the records.

I supplied the statement

username=blah&password=blah or 1=1

The double quotes kept tripping me and halting my progress but after pulling a few hairs I managed to solve it.

username="or "1"="1 &password=""

Just when I was happy with this, I noticed that the code does not check if the username and password is valid so I could simply input the above username as the username and empty password to get the response. This is because the username is always true so all records will be returned regardless of what the password even is.

[View sourcecode](#)

I used Burp Suite to view the request for this level and I found this - and there was a session ID which I noted. When I hex decoded it, it gave me 595-a which is within the range we used in the last level (1-640) so I think this level is similar.

So we can perform a brute force attack using Burp Suite, by sending the request to the intruder.

I used the following payload options in the payload tab after fiddling around:

DashboardTargetProxyIntruderRepeaterSequencerDecoderComparerExtenderProject optionsUser options

1 x ...

TargetPositionsPayloadsOptions

2 Payload Sets

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

Payload set: 1

Payload count: 1,616

Payload type: Numbers

Request count: 8,080

2 Payload Options [Numbers]

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

Number range

Type: ☒ Sequential ☐ Random

From: 1

To: 650

Step: 1

How many:

Number format

Base: ☐ Decimal ☒ Hex

Min integer digits: 1

Max integer digits: 650

Min fraction digits:

Max fraction digits:

Intruder attack 1

AttackSaveColumns

ResultsTargetPositionsPayloadsOptions

Filter: Showing all items

Requ...	Position	Payload	Status	Error	Timeout	Length	Comment
0				<input checked="" type="checkbox"/>	<input type="checkbox"/>		
1	1	312d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
2	1	322d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
3	1	332d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
4	1	342d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
5	1	352d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
6	1	362d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
7	1	372d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
8	1	382d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
9	1	392d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
10	1	612d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
11	1	622d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
12	1	632d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
13	1	642d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
14	1	652d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
15	1	662d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
16	1	31302d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
17	1	31312d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
18	1	31322d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
19	1	31332d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
20	1	31342d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
21	1	31352d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
22	1	31362d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
23	1	31372d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
24	1	31382d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		
25	1	31392d61646d696e		<input checked="" type="checkbox"/>	<input type="checkbox"/>		

Request

PrettyRaw\nActions

25 of 8080

Intruder attack 1								
Attack Save Columns								
Results Target Positions Payloads Options								
Filter: Showing all items								?
Requ...	Position	Payload	Status	Error	Timeout	Length	Comment	
0				✓	<input type="checkbox"/>			
1	1	312d61646d696e		✓	<input type="checkbox"/>			
2	1	322d61646d696e		✓	<input type="checkbox"/>			
3	1	332d61646d696e		✓	<input type="checkbox"/>			
4	1	342d61646d696e		✓	<input type="checkbox"/>			
5	1	352d61646d696e		✓	<input type="checkbox"/>			
6	1	362d61646d696e		✓	<input type="checkbox"/>			
7	1	372d61646d696e		✓	<input type="checkbox"/>			
8	1	382d61646d696e		✓	<input type="checkbox"/>			
9	1	392d61646d696e		✓	<input type="checkbox"/>			
10	1	612d61646d696e		✓	<input type="checkbox"/>			
11	1	622d61646d696e		✓	<input type="checkbox"/>			
12	1	632d61646d696e		✓	<input type="checkbox"/>			
13	1	642d61646d696e		✓	<input type="checkbox"/>			
14	1	652d61646d696e		✓	<input type="checkbox"/>			
15	1	662d61646d696e		✓	<input type="checkbox"/>			
16	1	31302d61646d696e		✓	<input type="checkbox"/>			
17	1	31312d61646d696e		✓	<input type="checkbox"/>			
18	1	31322d61646d696e		✓	<input type="checkbox"/>			
19	1	31332d61646d696e		✓	<input type="checkbox"/>			
20	1	31342d61646d696e		✓	<input type="checkbox"/>			
21	1	31352d61646d696e		✓	<input type="checkbox"/>			
22	1	31362d61646d696e		✓	<input type="checkbox"/>			
23	1	31372d61646d696e		✓	<input type="checkbox"/>			
24	1	31382d61646d696e		✓	<input type="checkbox"/>			
25	1	31392d61646d696e		✓	<input type="checkbox"/>			

Request

Pretty
Raw
\n
Actions

25 of 8080

It took an incredibly long time to get the request that gives us the credentials needed for level 20. So after some popcorn, I returned to see the request had succeeded and I had automatically gotten admin access to the next level.

Level 22

I decided to first view the source code as usual:

```

<?
session_start();

if(array_key_exists("revelio", $_GET)) {
    // only admins can reveal the password
    if(!($_SESSION and array_key_exists("admin", $_SESSION) and $_SESSION["admin"] == 1)) {
        header("Location: /");
    }
}
?>

<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": "natas22", "pass": "<censored>" };</script></head>
<body>
<h1>natas22</h1>
<div id="content">

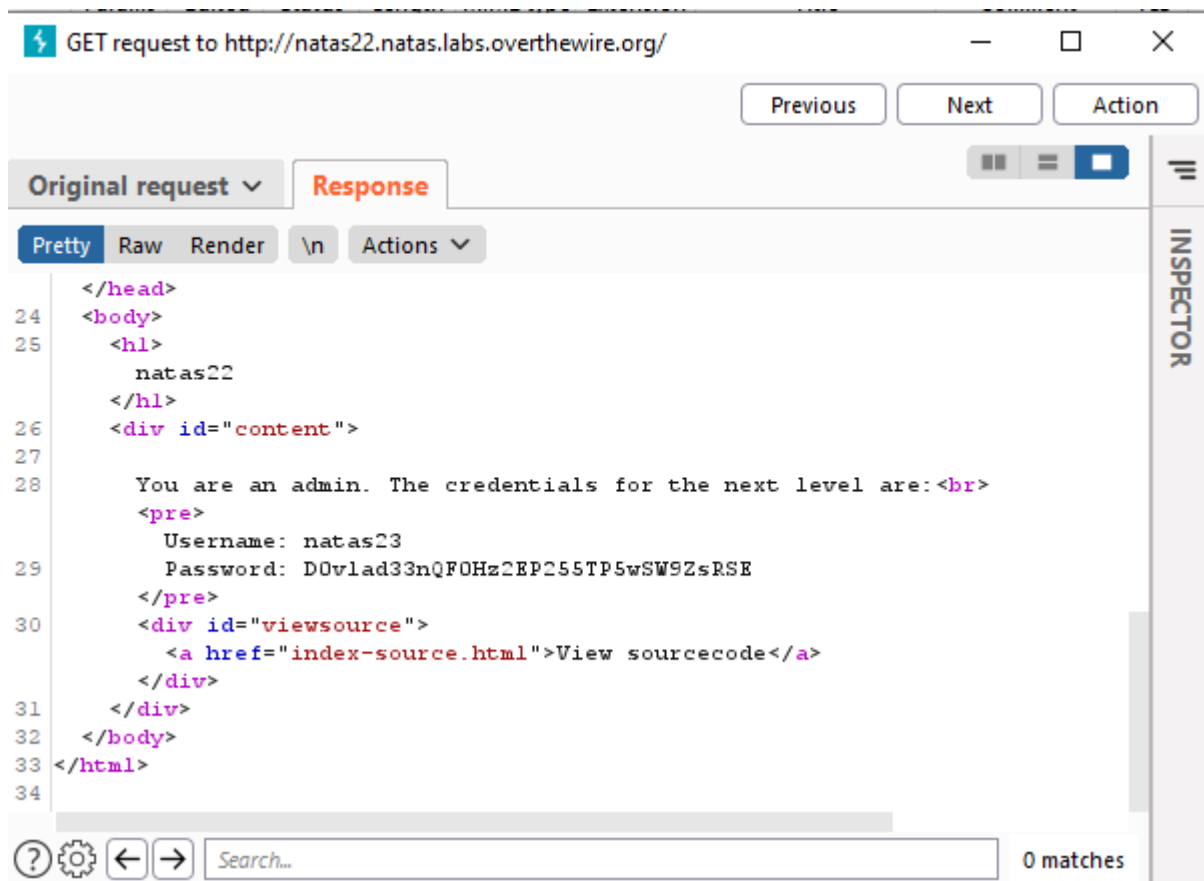
<?
    if(array_key_exists("revelio", $_GET)) {
        print "You are an admin. The credentials for the next level are:<br>";
        print "<pre>Username: natas23\n";
        print "Password: <censored></pre>";
    }
?>

```

It seems like the first block of code adds a Location: / header if you do not have the admin session variable. If it sees the 'revelio' GET parameter, then it gives us the password.

I looked at how I could add this parameter using Burp Suite, and had done so easily due to using it in the earlier natas levels.

So I added the revelio parameter in the Burp Suite interceptor, and got the password for level 23 in the HTTP response.



Level 23

My first thought again is to look at the source code and I saw this:

```
Password:
<form name="input" method="get">
  <input type="text" name="passwd" size=20>
  <input type="submit" value="Login">
</form>

<?php
if(array_key_exists("passwd",$_REQUEST)){
    if(strpos($_REQUEST["passwd"],"iloveyou") && ($_REQUEST["passwd"] > 10 )){
        echo "<br>The credentials for the next level are:<br>";
        echo "<pre>Username: natas24 Password: <censored></pre>";
    }
    else{
        echo "<br>Wrong!<br>";
    }
}
// morla / 10111
?>
```

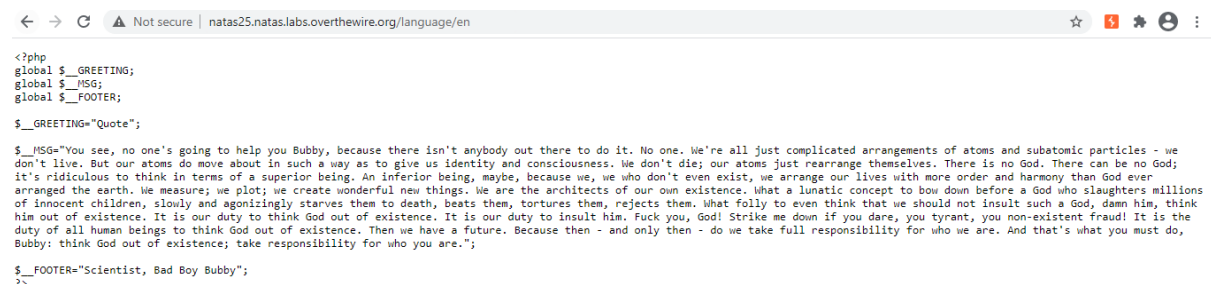
I tried to input the password 'iloveyou' but the request has to be larger than 10 so it failed. I was not sure how to do this and I tried using Burp Suite to intercept the request and find more information in the response.

It took me so long before noticing the obvious - that I could simply put the number 11 in the string, so that the password was '11iloveyou'

This worked and I retrieved the password for the next level

Level 25

There is a quote (from Bad Boy Bubby) on the main page which was a fun read. The source code says that when the page loads, if the 'lang' parameter is set, then setLanguage() will include a file of that name. Since we use English by default, I tried to see if the server lets us view it. I tried the path /language/english but it did not work but eventually I realised that I should use 'en' instead and then it worked.



```
<?php
global $_GREETING;
global $_MSG;
global $_FOOTER;

$_GREETING="Quote";

$_MSG="You see, no one's going to help you Bubby, because there isn't anybody out there to do it. No one. We're all just complicated arrangements of atoms and subatomic particles - we don't live. But our atoms do move about in such a way as to give us identity and consciousness. We don't die; our atoms just rearrange themselves. There is no God. There can be no God; it's ridiculous to think in terms of a superior being. An inferior being, maybe, because we, we who don't even exist, we arrange our lives with more order and harmony than God ever arranged the earth. We measure; we plot; we create wonderful new things. We are the architects of our own existence. What a lunatic concept to bow down before a God who slaughters millions of innocent children, slowly and agonizingly starves them to death, beats them, tortures them, rejects them. What folly to even think that we should not insult such a God, damn him, think him out of existence. It is our duty to think God out of existence. It is our duty to insult him. Fuck you, God! Strike me down if you dare, you tyrant, you non-existent fraud! It is the duty of all human beings to think God out of existence. Then we have a future. Because then - and only then - do we take full responsibility for who we are. And that's what you must do, Bubby: think God out of existence; take responsibility for who you are.";

$_FOOTER="Scientist, Bad Boy Bubby";
?>
```

It seems like the safeInclude() function is stopping us from just accessing the password in the **natas_webpass** folder by doing a quick traversal **../natas_webpass**. But I need to access the file in this directory if I want to view the password for the next level.

I researched on google for ways that I can leave a **../** even after str_replace replaces all instances of **../**. It was hard because even then, they included a check that checks for the actual folder to ensure it is not the one containing the password.

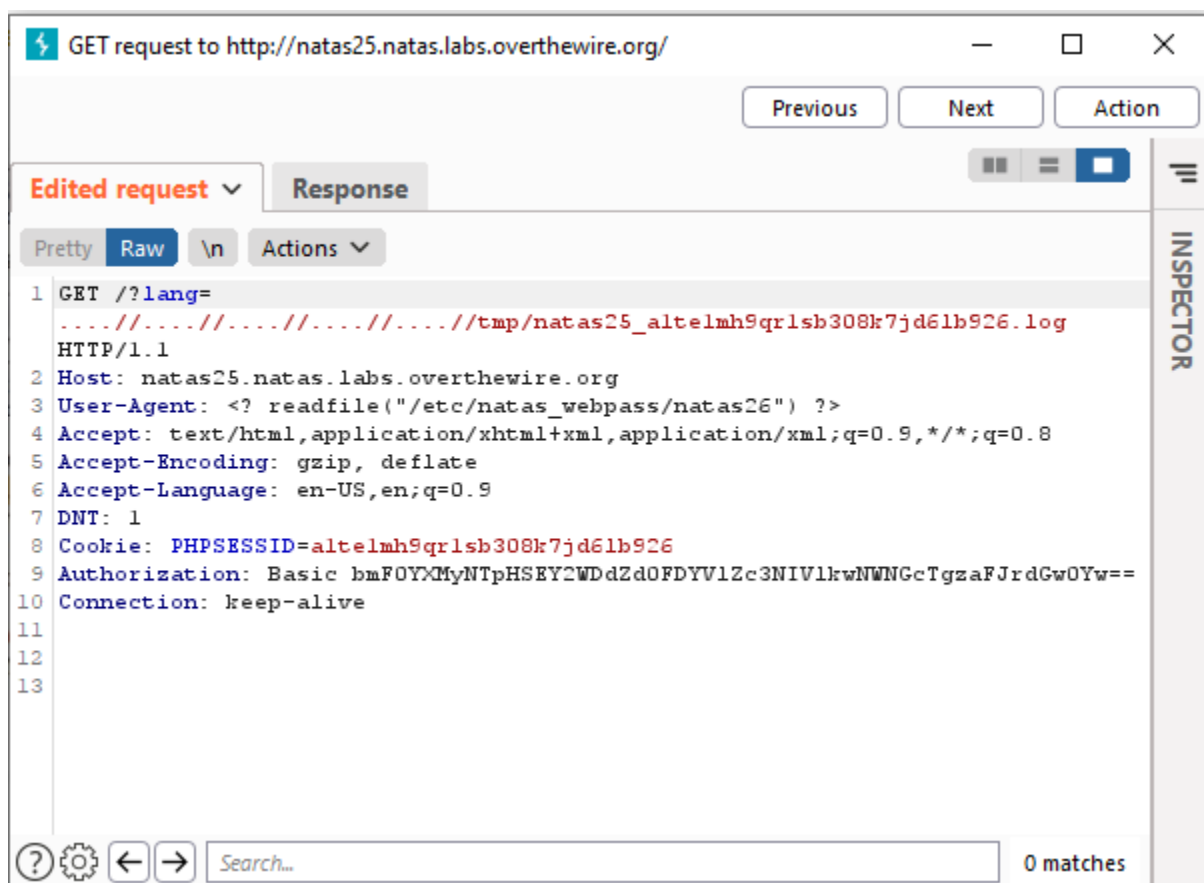
I noticed the logrequest function and thought that it may be possible to inject the password here. I searched online for how to inject my own message by exploiting vulnerabilities in HTTP_USER_AGENT since that is what the log function uses, using Burp Suite, so that I could execute the logs as if it was a 'LANGUAGE'.

The problem is that HTTP_USER_AGENT came from the User-Agent header which is something an attacker controls because it comes into the web server in an HTTP request. And that's a recipe for disaster because an attacker can make a vulnerable server run any command it wants (see examples below).

The above was taken from Stack Overflow

(<https://security.stackexchange.com/questions/68216/how-are-you-able-to-inject-bash-script-through-the-user-agent>) and provided me with the knowledge that I should be using the User-Agent header to inject my custom message into the logs.

Now I tried to execute the log by supplying its filename as a parameter in the interceptor in Burp Suite. Upon Googling, all sources pointed to using php to inject my variable into the header so I created a function to pass into the header. That did not work however the following did:



Overall reflection

Challenges

It was my first time intercepting HTTP requests using Burp Suite tools, and I had also not performed SQL injection attacks prior to this week so it was cool reading up on how they work and implementing some myself. It was also a bit tough learning to use these tools (or since I learnt databases) but learning the process was really intriguing. It took me a while to get acquainted with writing php code which was often needed to retrieve the password from the corresponding etc folder but it was interesting to see how we can use php as a bruteforce technique for example, solving the challenge of gaining access to the password when our interface requests for a .jpg image.

I created a video for level 22, showing how I solved it

<https://storyxpress.co/video/kncjipka1lkyvm2mh>

Reflection

Natas teaches the basics of serverside web-security.

The security issues I learnt in this level included:

- Editing HTTP headers
- File inclusion vulnerabilities
- Breaking weak encryption
- Editing cookies to assume the identity of a different user
- Bruteforce techniques
- SQL injections
- SQL tautologies
- Blind SQL injections

In challenge 3, I learnt about how each site must contain a file called robots.txt which tells search engines which parts of the site they can and cannot access. When I accessed the robots file for the level 3 Natas site, I saw the file they had disallowed, so when I went to that path, I found the password. Challenge 5 and 6 involved using the tool Burp Suite to intercept and modify the headers to gain access to the resource that contains the password. In summary, I went through many topics of security, and was blown away by how relatively easy one can make an attack using free tools such as Burp Suite. It would be cool to try out the professional version some day and attain that insane power.

