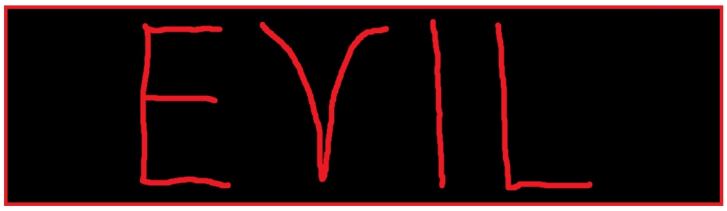

Evil Corp. Penetration Test

by Justus Uurtimo

Start of testing: March 30, 2022

End of testing: April 6, 2022

Contents

1 Executive Summary

In this penetration test the Evil Company was examined for security-relevant weaknesses. The testing was conducted as grey-box testing, this is the kind where partial information about the internals of the system is given. For example the structure of PHP was given in all of the sites that was to be tested. The scope of the assessment was as follows:

Table 1.1 contains the overview of examined systems during the penetration test.

Web Site	**Hostname**
Domain 1	http://natas.labs.overthewire.org
Subdomain 1	http://natas0.natas.labs.overthewire.org
Subdomain 2	http://natas1.natas.labs.overthewire.org
Subdomain 3	http://natas2.natas.labs.overthewire.org
Subdomain 4	http://natas3.natas.labs.overthewire.org
Subdomain 5	http://natas4.natas.labs.overthewire.org
Subdomain 6	http://natas5.natas.labs.overthewire.org
Subdomain 7	http://natas6.natas.labs.overthewire.org
Subdomain 8	http://natas7.natas.labs.overthewire.org
Subdomain 9	http://natas8.natas.labs.overthewire.org
Subdomain 10	http://natas9.natas.labs.overthewire.org
Subdomain 11	http://natas10.natas.labs.overthewire.org

Table 1.1: Web sites examined during the penetration test

As a result, several vulnerabilities have been found among the assets of the organization, some of them pose a significant risk. Among these vulnerabilities were vulnerabilities to code injection and directory traversal, that can pose significant risks to any online service. Solutions to remedy the discovered vulnerabilities are provided together with detailed descriptions and reproduction steps.

2 Vulnerability overview

Figure 2.1 shows the overview of vulnerabilities that could be measured in a meaningful way with CVSSv3 scoring. They are grouped by target and in descending order of risk. They are categorized by their risk and potential and are differentiated in the categories low, medium, high and critical. These categories essentially mean how likely it is that the vulnerability is taken advantage of and how much impact it would have on the business operations. In this table I am following the NVD severity ratings. They are based on the CVSSv3 scoring and are categorized:

score of 0.0 = None
score 0.1 - 3.9 = Low
score 4.0 - 6.9 = Medium
score 7.0 - 8.9 = High
score 9.0 - 10.0 = Critical

Risk	Asset	Vulnerability	Section	Page
Critical	Subdomain 10	Command injection	3.10	32
Critical	Subdomain 11	Command Injection	3.11	35
Critical	Subdomain 8	Path travelsal	3.8	26
Critical	Subdomain 5	Reliance on Cookies without Validation and Integrity Checking	3.5	16
Critical	Subdomain 6	Reliance on Cookies without Validation and Integrity Checking	3.6	20
High	Subdomain 3	Exposure of Information Through Directory Listing)	3.3	8
High	Subdomain 4	Exposure of Information Through Directory Listing)	3.4	12
High	Subdomain 7	Path travelsal	3.1	4
Medium	Subdomain 1	Inclusion of Sensitive Information in Source Code	3.1	4
Medium	Subdomain 2	Inclusion of Sensitive Information in Source Code	3.1	4

Table 2.1: Vulnerability overview

3 Results

In this chapter, the vulnerabilities found during the penetration test are presented. All of the issues are in the order that I solved their respective exercises. Each contain the following information.

- Brief description of the vulnerability
- CVSSv3 score and explanations for each scoring when they are applicable.
- References to related CWE
- Steps to reproduce.

Also the remediation recommendations are given for each issue found during the penetration test.

3.1 Natas 0

Natas system.
Hostname: http://natas0.natas.labs.overthewire.org

3.1.1 General information

The page of Natas 0 contains only a text stating that the password is located in the page. After opening the developer tools and the inspector tool, the password for Natas 1 can be found as a HTML comment. Storing any important information in the page code as comments or otherwise is a very bad practice.

The CWE categorization of this vulnerability would be: CWE-540: Inclusion of Sensitive Information in Source Code

CVSS calculation of this vulnerability is presented in Table 3.2.

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	Low	Attacker had some authorization prior attack (you need to login to the site as natas0)
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	The attacker was able to obtain the password for the site.
Integrity Impact (I)	None	The attacker is not able to modify any information
Availability Impact (A)	None	No loss of availability
Overall	6.5	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Table 3.1: CVSSv3 score of the issue

3.1.2 Proof of concept

After first opening the site and the inspector tool from developer tool, I am immediately greeted with the password as a HTML comment.

Figure 3.1: flag natas0

3.1.3 Proposed solutions

The developer and the site admins need to make sure there are no important or vulnerable information stored in the sites source code.

3.2 Natas 1

Natas system.
Hostname: http://natas1.natas.labs.overthewire.org/

3.2.1 General information

This challenge stores the sensitive information inside an alert function. The problem is essentially the same as in the natas 0, you should not store sensitive information inside the site source code.

The CWE catogorization of this vulnerability would be: CWE-540: Inclusion of Sensitive Information in Source Code

CVSS calculation of this vulnerability is presented in Table 3.10.

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	Low	Attacker had some authorization prior attack (you need to login to the site as natas0)
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	The attacker was able to obtain the password for the site.
Integrity Impact (I)	None	The attacker is not able to modify any information
Availability Impact (A)	None	No loss of availability
Overall	6.5	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Table 3.2: CVSSv3 score of the issue

3.2.2 Proof of concept

After accessing the site, I am greeted with a message stating that:
"You can find the password for the next level on this page, but rightclicking has been blocked!"

When right clicking on the page, the pages creates an alert. When inspecting that alert with developer tools, I am presented with the password for the next challenge.

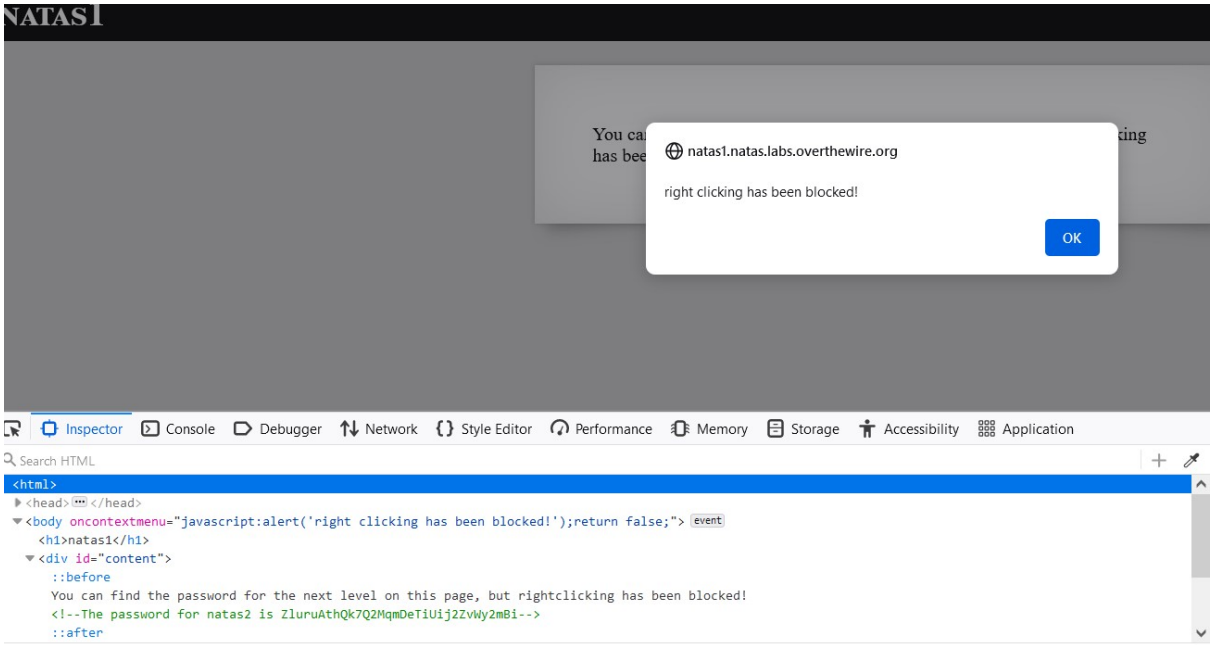

Figure 3.2: flag natas 1

3.2.3 Proposed solutions

The solution for this vulnerability is the same as in the natas0, you should not be storing any sensitive information inside the site source code. As such an immediate clean of the code and changing the passwords is recommended.

3.3 Natas 2

Natas system.
Hostname: http://natas2.natas.labs.overthewire.org/

3.3.1 General information

Natas 2 is essentially a directory listing challenge, where you must gain access to a file containing the password for the next challenge.

CWE catogarisation of this vulnerability would be: CWE-548: Exposure of Information Through Directory Listing)

CVSS calculation of this vulnerability is presented in Table 3.8.

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Attacker is able to gain used login information for multiaple users.
Integrity Impact (I)	None	Attacker is not able to modify any data with this attack alone.
Availability Impact (A)	None	Attacker is not able to affect site availability with this attack alone.
Overall	7.5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Table 3.3: CVSSv3 score of the issue

3.3.2 Proof of concept

By inspecting the site there seems to be a file hidden there.

Figure 3.3: file hidden

After seeing that, my first instinct is to try to see if the directory where that file is stored, is actually protected or if it would reveal any secrets.

Figure 3.4: Traversal worked

The traversal worked and I can see that there is the pixel file and something called users.txt After accessing that file I can see many login credentials and the password for our next objective flag: sJIJNW6ucpu6HPZ1ZAchaDtwd7oGrD14

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

Figure 3.5: Flag natas 2

3.3.3 Proposed solutions

Solution for this vulnerability would be to restrict access to any important directories or files. This could be done by adopting a need to know requirement for both the document and server root and after this turning off features such as Automatic directory listings (Common Weakness Enumeration, n.d.). The web server should be configured to prevent directory listings for all paths beneath the web root. Also default .html files should be placed in each directory that is shown instead of returning directory listing. (PortSwigger, n.d.)

3.4 Natas 3

Natas system.
Hostname: http://natas3.natas.labs.overthewire.org/

3.4.1 General information

Natas 3 is essentially a directory listing challenge, where you must gain access to a file containing the password for the next challenge. The difference on Natas 2 is that in this time the robots.txt file was the key information on obtaining the location of the file.

CWE catogarisation of this vulnerability would be: CWE-548: Exposure of Information Through Directory Listing)

CVSS calculation of this vulnerability is presented in Table 3.8.

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Attacker is able to gain used login information for multiaple users.
Integrity Impact (I)	None	Attacker is not able to modify any data with this attack alone.
Availability Impact (A)	None	Attacker is not able to affect site availability with this attack alone.
Overall	7.5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Table 3.4: CVSSv3 score of the issue

3.4.2 Proof of concept

After first opening the site and opening the developer tools, I can see this very useful hint. The hint states that not even google can find it this time. Search engines will index websites by utilizing automated crawlers. The websites can manage these crawlers by utilizing robots.txt file that will tell the crawlers where they are allowed to go. My guess is that this robots.txt will reveal some interesting information.

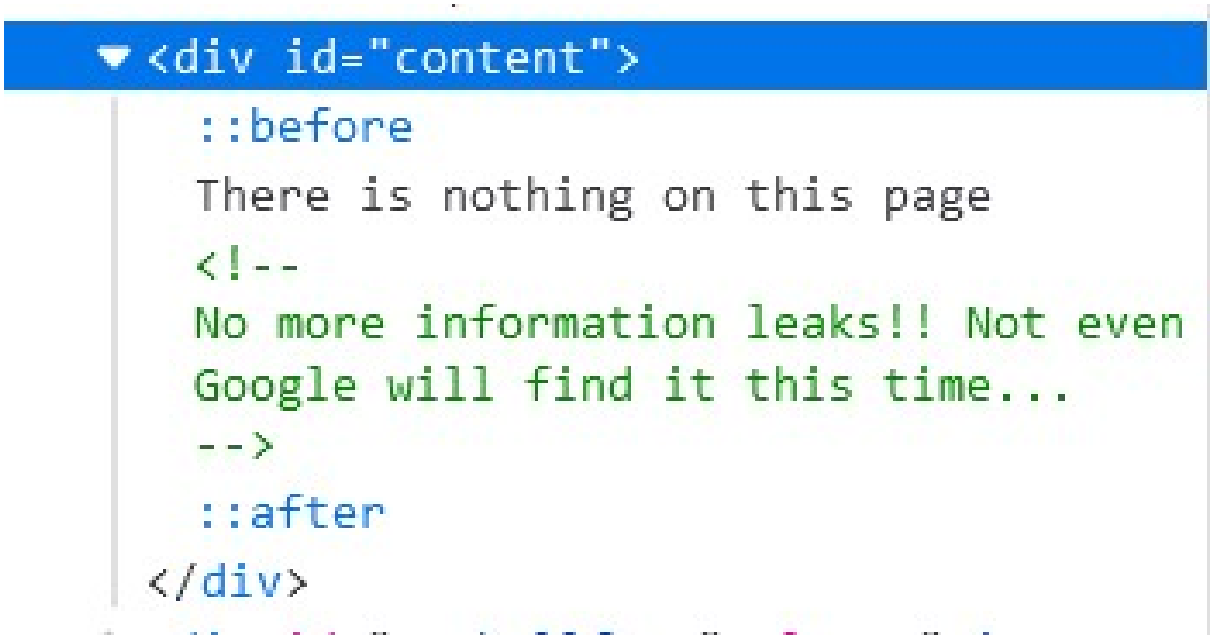

Figure 3.6: Hint on robots.txt file

After accessing the robots.txt I can see that there is disallowed directory named /s3cr3t/ So I will try to see if I am able to access this disallowed directory.

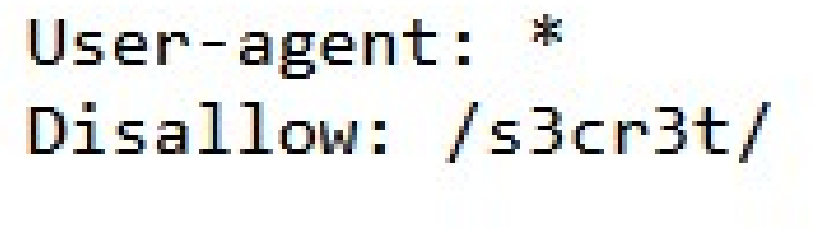

Figure 3.7: s3cr3t

The directory is accessible and it reveals a very interesting file named users.txt Next I will try to access it, as it most definitely holds interesting information.

Figure 3.8: users file

The file is accesable and it reveals the secret!
flag: Z9tkRkWmpt9Qr7XrR5jWRkgOU901swEZ

Figure 3.9: flag of natas 3

3.4.3 Proposed solutions

Solution for this vulnerability would be to restrict access to any important directories or files. This could be done by adopting a need to know requirement for both the document and server root and after this turning off features such as Automatic directory listings (Common Weakness Enumeration, n.d.). The web server should be configured to prevent directory listings for all paths beneath the web root. Also default .html files should be placed in each directory that is shown instead of returning directory listing. (PortSwigger, n.d.)

3.5 Natas 4

Natas system.
Hostname: http://natas4.natas.labs.overthewire.org/

3.5.1 General information

This challenge is essentially all about spoofing the cookies to gain access to a site you should not have access to.

CWE categorization of this vulnerability could be: CWE-565: Reliance on Cookies without Validation and Integrity Checking

CVSS calculation of this vulnerability is presented in Table 3.6

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker does not need any privileges
User Interaction (UI)	None	The attack requires no interaction from users
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	The attacker is able to retrieve the password and thus access to the whole service and its files
Integrity Impact (I)	High	With the access to the service the attacker would be able to modify any data in there
Availability Impact (A)	None	The attacker can not impact the availability of the resources with this attack alone.
Overall	9.1	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Table 3.5: CVSSv3 score of the issue

3.5.2 Proof of concept

I am first greeted with a message stating that I am disallowed and all authorized users should come from different url. This gives me a hint that the site will check where the user is coming from, from cookies. I will first try to modify them to maybe gain access to the secret.

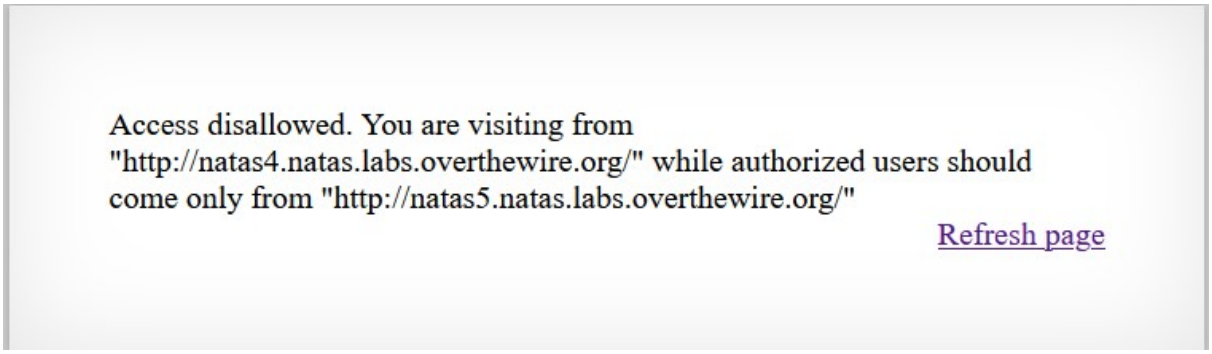

Figure 3.10: first sight

As can be seen there is a referer field in the Request headers that state where the user is coming from. So next I will try to modify this referer and maybe gain access to the secret.

Figure 3.11: referer

For these exercises I am using FireFox and it at least has the ability to modify the packets and resend them (Network and right click on the package and select edit and resend) As can be seen from the image below I am in fact able to modify the referer field. I will next try to modify it and try to gain access to the secret.

The website wont change, but I can now see there is a new package sent (with my

Figure 3.12: referer

modified referer) From that packet I can find the infromation I am seeking from the response tab. This actually shows me the secret! Flag: iX6IOfmpN7AYOQGPwtn3fXpbaJVJcHfq

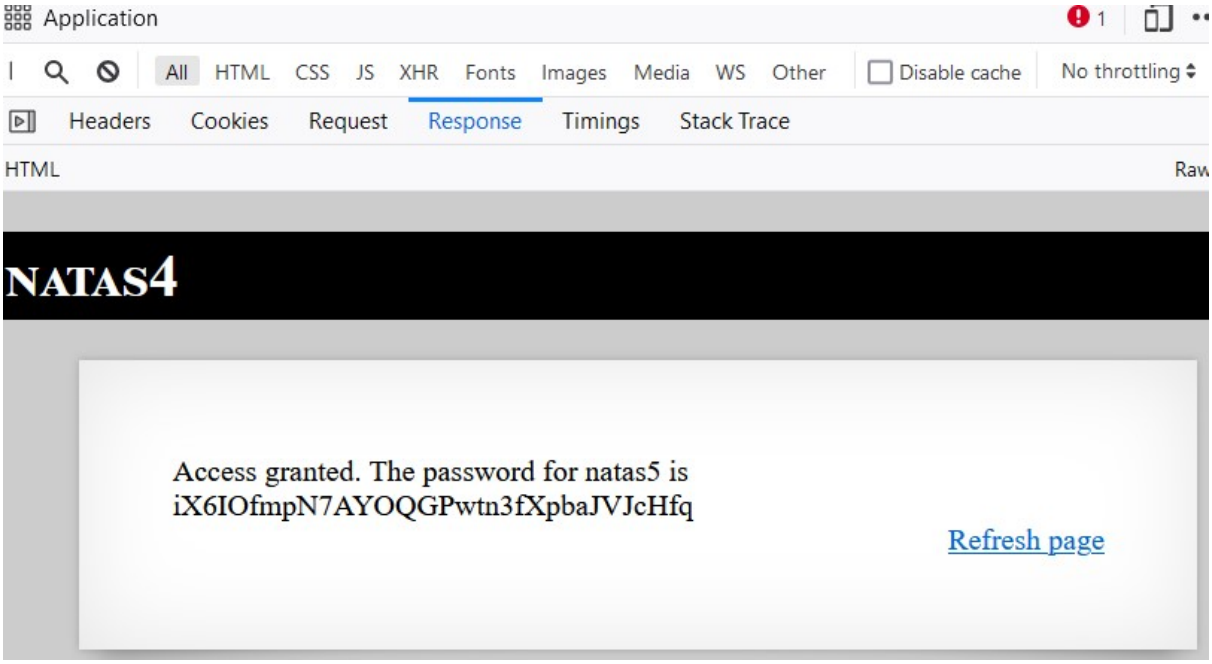

Figure 3.13: flag natas 4

3.5.3 Proposed solutions

There are few possible ways to mitigate this vulnerability. For example it is advisable to avoid using cookie data for a security-related decision. Also the service shoud perform thorough input validation (i.e.: server side validation) on the cookie data if you're going to use it for a security related decision. Lastly there should be integrity checks to detect tampering. (Common Weakness Enumeration, n.d.)

3.6 Natas 5

Natas system.
Hostname: http://natas5.natas.labs.overthewire.org/

3.6.1 General information

This challenge is quite similiar to the last one but in this case the cookies are keeping track weather I am logged in or not. The challenge requires to spoof the cookies in a way that I trick the site into thinking that I am logged in, when in reality I am not.

CWE categorization of this vulnerability could be: CWE-565: Reliance on Cookies without Validation and Integrity Checking

CVSS calculation of this vulnerability is presented in Table 3.6

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker does not need any privileges
User Interaction (UI)	None	The attack requires no interaction from users
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	The attacker is able to retrieve the password and thus access to the whole service and its files
Integrity Impact (I)	High	With the access to the service the attacker would be able to modify any data in there
Availability Impact (A)	None	The attacker can not impact the availability of the resources with this attack alone.
Overall	9.1	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Table 3.6: CVSSv3 score of the issue

3.6.2 Proof of concept

I am first greeted with a text that states I am not logged in. My first quess is to check that maybe the site checks this from some cookie filed or such.

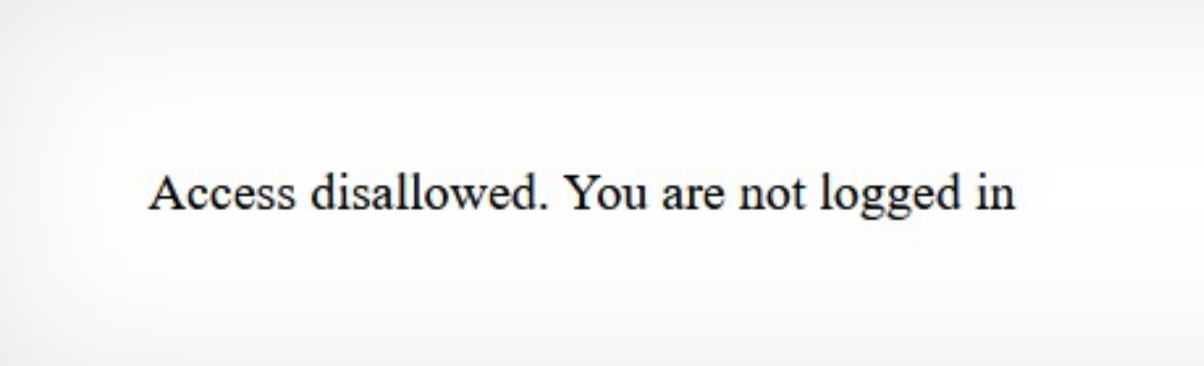

Figure 3.14: first sight natas 5

It seems that my hunch was right and there is in fact a field cookie:loggedin=0 This gives me the idea that maybe this can be modified as well, the same way that I did in the previous exercise.

Figure 3.15: cookies

I modified the cookie header to state loggedin=1 and resend the request. As can be seen from the from the image below, I am granted an access and the secret is found.

Flag: aGoY4q2Dc6MgDq4oL4YtoKtyAg9PeHa1

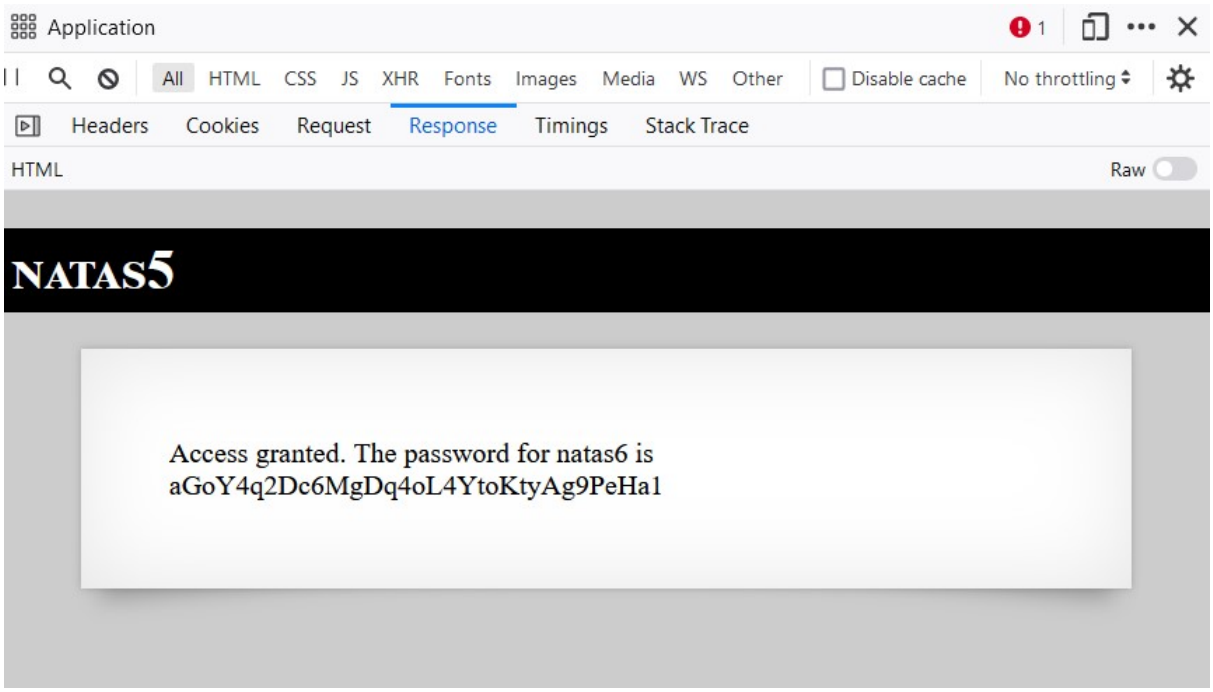

Figure 3.16: flag natas 5

3.6.3 Proposed solutions

There are few possible ways to mitigate this vulnerability. For example it is advisable to avoid using cookie data for a security-related decision. Also the service shoud perform thorough input validation (i.e.: server side validation) on the cookie data if you're going to use it for a security related decision. Lastly there should be integrity checks to detect tampering. (Common Weakness Enumeration, n.d.)

3.7 Natas 6

Natas system.
Hostname: http://natas6.natas.labs.overthewire.org/

3.7.1 General information

This challenge is essentially a modified directory listing challenge. The main difference is that in this challenge I am required to have at least some knowledge on how php works. In this challenge I am also given the source code, which is needed in order to hack this site.

CWE catogarisation of this vulnerability would be: CWE-548: Exposure of Information Through Directory Listing)

CVSS calculation of this vulnerability is presented in Table 3.6

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Attacker is able to gain used login information for multiaple users.
Integrity Impact (I)	None	Attacker is not able to modify any data with this attack alone.
Availability Impact (A)	None	Attacker is not able to affect site availability with this attack alone.
Overall	7.5	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Table 3.7: CVSSv3 score of the issue

3.7.2 Proof of concept

I am first presented with a submit form.

Figure 3.17: first sight natas 6

```
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";
    }
    }
?>
```

Figure 3.18: source code

There is also a button to view the source code.

The source code is in itself quite simple and it reveals that the form will check the input against a value $secret that is a value stored in a file called secret.inc (that can be determined since that is the only file included). Next I will try to access that file and in order to do so I will browse to http://natas6.natas.labs.overthewire.org/includes/secret.inc That shows me an empty site, but after viewing the source code I am presented with the secret need for the form.

After that I simply just input that secret into the submit form and I am presented with the flag. Flag=7z3hEENjQtflzgnT29q7wAvMNfZdh0i9

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

Figure 3.19: secret found

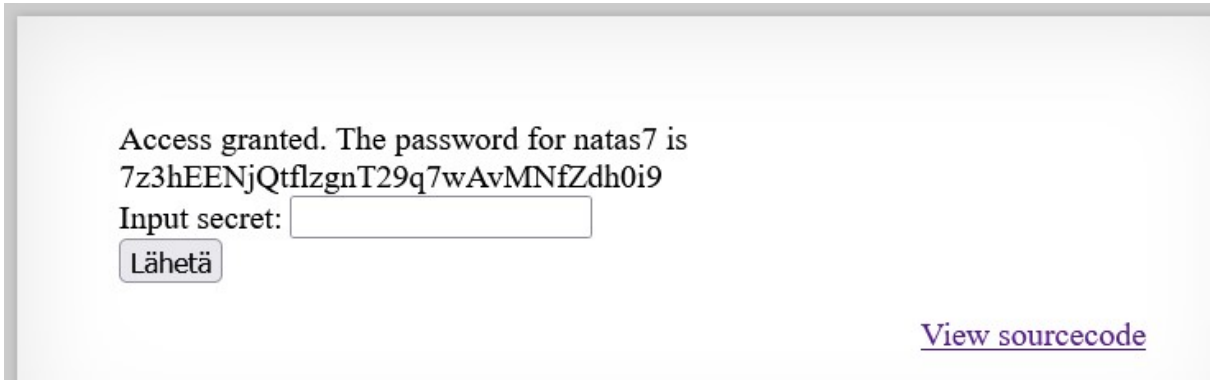

Figure 3.20: flag natas 6

3.7.3 Proposed solutions

Even though in the normal scenario the PHP code would not be visible, this challenge is essenttially the same as the previous directory listing challenges and as such the solutions are the same.

Solution for this vulnerability would be to restrict access to any important directories or files. This could be done by adopting a need to know requirement for both the document and server root and after this turning off features such as Automatic directory listings (Common Weakness Enumeration, n.d.). The web server should be configured to prevent directory listings for all paths beneath the web root. Also default .html files should be placed in each directory that is shown instead of returning directory listing. (PortSwigger, n.d.)

3.8 Natas 7

Natas system.
Hostname: http://natas7.natas.labs.overthewire.org/

3.8.1 General information

This challenge is essentially about directory traversal, or path traversal and file inclusion.

The CWE categorisation of this challenge is: CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

CVSS calculation of this vulnerability is presented in Table 3.6

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Attack is be able to retrieve and read contents of files and expose sensitive data.
Integrity Impact (I)	High	It is possible for the attacker to modify the files in the server if a remote file inclusion is utilized.
Availability Impact (A)	High	Availability can be affected if the attacker points to a file that the application does not expect.
Overall	9.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Table 3.8: CVSSv3 score of the issue

3.8.2 Proof of concept

I am first greeted with a page that has two links: Home and About These both open a different page that state what page you are on. I can also see that the way this is done is

with index.php?page=NAMEOFTHEPAGE Viewing the site with the developer tools inspector also gives me the hint that the password is located at /etc/natas_webpass/natas8 My first idea is that I must use directory travelsal in order to gain access to the next password.

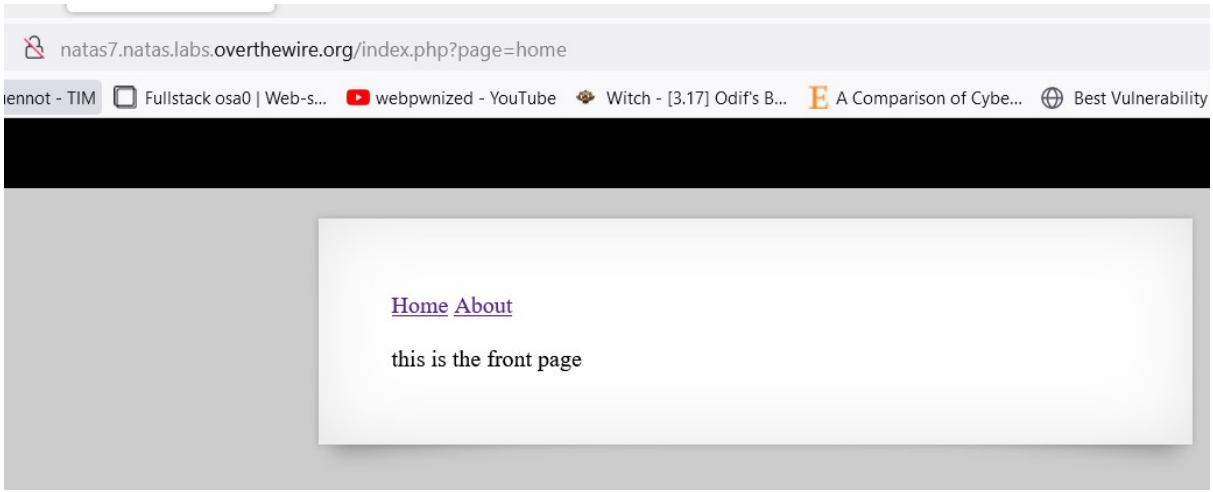

Figure 3.21: first sight natas 7

After that I check the source code of the site. From that I can see that the web links actually reference a page file that is then passed into the index.php This leads me to thinking that maybe I can pass something else to it. This vulnerability is called file inclusion and since the hint gave me the direct address of the password I first try to simply put that in. As can be seen from the image below this indeed gave me the password. flag: DBfUBfqQG69KvJvJ1iAbMoIpwSNQ9bWe

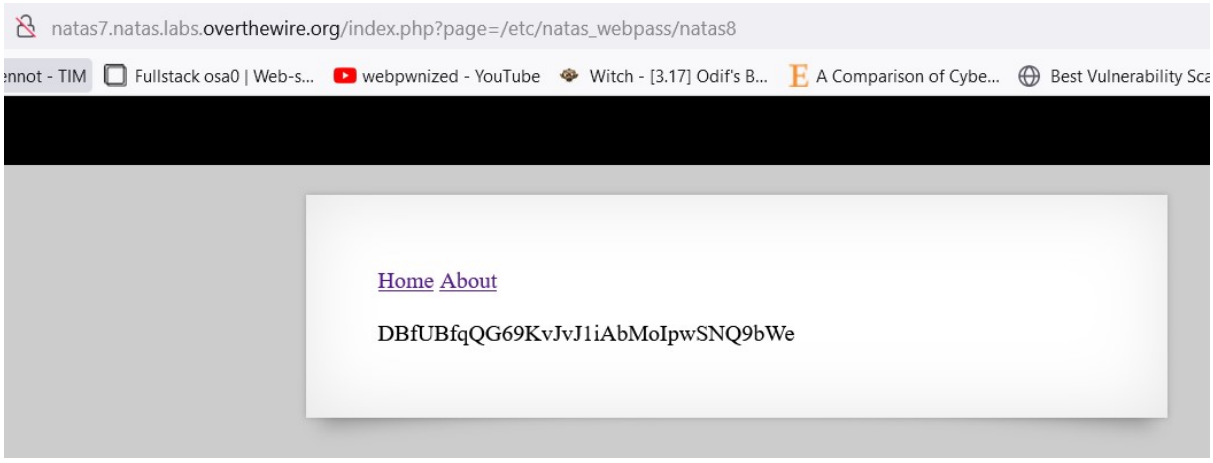

Figure 3.22: flag natas7

3.8.3 Proposed solutions

This vulnerablity can be mitigated by making sure that the applications are running on lowest possible privileges. In case that PHP is used it is the utmost importance that "allow_url_fopen" and "allow_url_include" are disabled. Also setting up input validations are an effective way of preventing file inclusion (Dumanhug 2020). Also if file inclusion feature can not be disabled, the developers should at leas white-list the accepted filenames and limit the input, so that only those files can be accessed. (Shyam 2021)

3.9 Natas 8

Natas system.
Hostname: http://natas8.natas.labs.overthewire.org/

3.9.1 General information

This challenge is more about learning how to write small scripts with php, that could be useful in serverside penetration testing. As such no CVSSv3 scoring is done.

3.9.2 Proof of concept

When I first open natas8 I can see an post form of somekind and view sourcecode button.

Figure 3.23: first sight natas 8

When viewing the sourcecode reveals to me that the site will encode the input and the compare it to a pre-encoded value. The function that encodes this value is also present in the source code.

With this information my first set of actions is to try and de-code the given encoded secret. Since I know what the encoder does I can just simply make my own php file that does the reverse.

The code I used was:

```
<?php
function decode($encoded)
return base64_decode(strrev(hex2bin($encoded)));
```

```
<?

$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";
    }
}
?>

<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>
```

Figure 3.24: source code natas 8

```
print "Decoded: ";
print decode("3d3d516343746d4d6d6c315669563362");
print "";
print "";
?>
```

After this I just simply go to the folder where I saved this decoder.php file and run it with:
php decode.php
This gives me the secret as a human readable string

```
C:\Users\uurti\Desktop\yliopisto\2001&2004\natas\natas8>php -f decode.php
Decoded: oubWYf2kBq
C:\Users\uurti\Desktop\yliopisto\2001&2004\natas\natas8>php decode.php
Decoded: oubWYf2kBq
C:\Users\uurti\Desktop\yliopisto\2001&2004\natas\natas8>
```

Figure 3.25: secret in human readable form

Lastly I just input the string into the natas8 site and I am presented with the password

Figure 3.26: Flag natas 8

3.10 Natas 9

Natas system.
Hostname: http://natas9.natas.labs.overthewire.org/

3.10.1 General information

This challenge is essentially about learning about command injection attack vector.

Command injection, also known as command execution, is a type of attack where the attacker will try to execute some arbitrary commands in vulnerable application. The attack is made possible if the application passes user supplied data straight to system shell. (Zhong 2021) Successful abuse of this vulnerability allows the attacker to run any arbitrary commands on hosts operating system using the vulnerable web application and as such this vulnerability always results in complete loss of confidentiality, integrity and availability of the target machine. (Maurya 2020)

CWE classification of this vulnerability could be: CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection') and CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

CVSS calculation of this vulnerability is presented in Table 3.6

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Complete loss of Confidentiality
Integrity Impact (I)	High	Complete loss of integrity
Availability Impact (A)	High	Complete loss of availability
Overall	9.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Table 3.9: CVSSv3 score of the issue

3.10.2 Proof of concept

My first sight on the site revelas that it also has a submit form of a kind that after few trials will find words containing something that the user writes in the box.

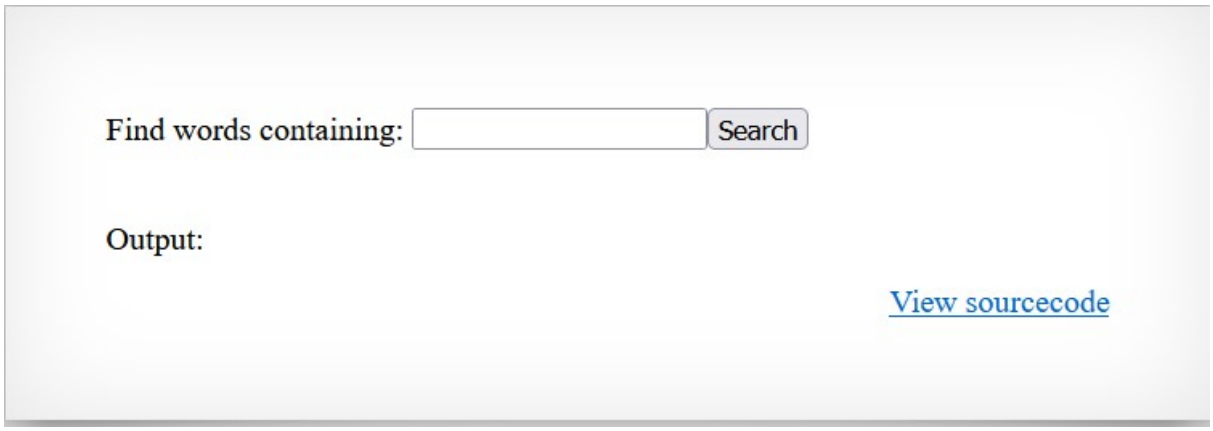

Figure 3.27: First sight natas 9

By viewing the source code I can see that our request is passed to a search function. This search function is using a php function called passtrhu(). This function will: "Execute an external program and display raw output"
https://www.php.net/manual/en/function.passthru.php

The caveat here is that there are no checking on what the input is or is it malicious. It will simply take whatever the value is in $key and execute it blindly. My first intuition is to apply my knowledge of command injection and try to retrieve the password for natas 10.

From the Natas introduction I know that :"All passwords are also stored in /etc/-natas_webpass/. E.g. the password for natas5 is stored in the file /etc/natas_webpass/natas5 and only readable by natas4 and natas5."

I first try to inject command jaa || cat /etc/natas_webpass/natas10 Here "jaa" is arbitary and could be anything even empty. This gives me the list of all entries in the dictionary and the password for the natas10. Another way would be to use the ";" control function that allows the user to execute multiple commands in the same command line entry. This would become then: ; cat /etc/natas_webpass/natas10
Also "#" could be appended to the end of the injection, since that way the rest of the original command would be ignored and the dictionary would not be printed only the password.

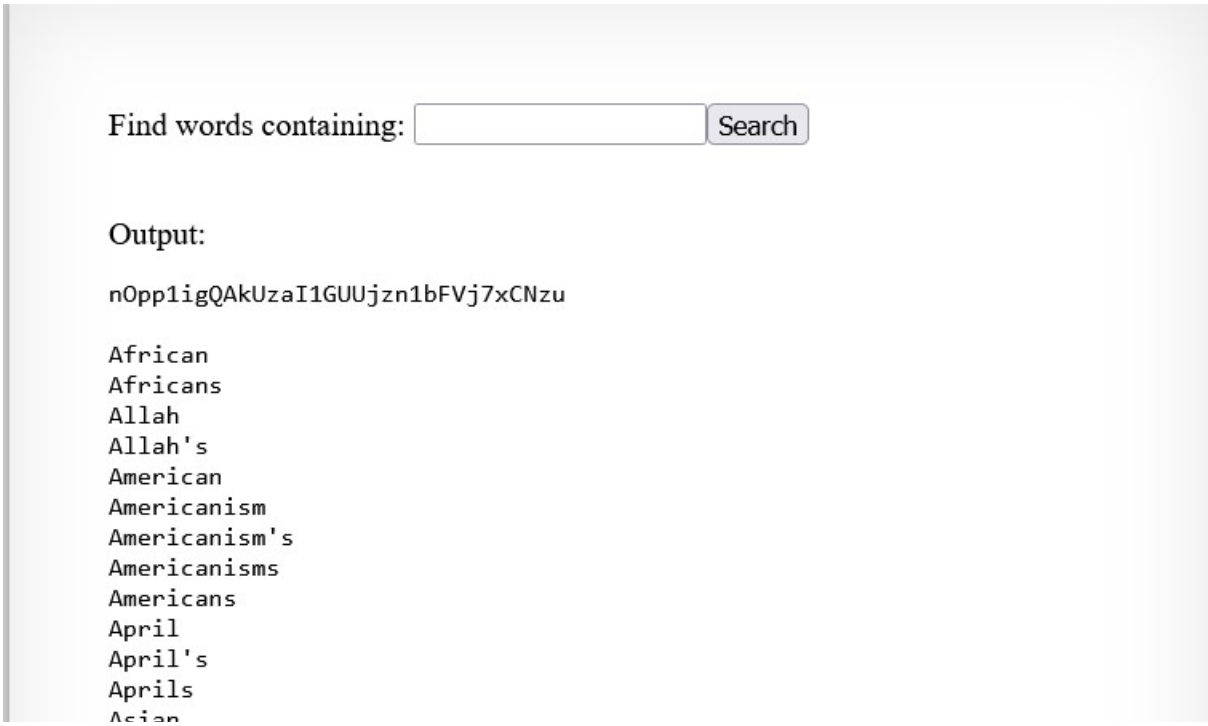

Figure 3.28: flag natas 9

3.10.3 Proposed solutions

One of the ways to prevent Command Injection is to just never call out to OS commands from the application-layer code. Instead safer implementations should be pursued. In cases that the OS commands can not be avoided a strong input validation should be in use. These can be for example, making sure that the input is a number, using white-lists of permitted values and making sure only alphanumeric values are used in input. (PortSwigger, n.d.) Additional defense could include making sure that the applications run at the lowest possible privileges as the commands will be executed with the same privileges as the application has. (OWASP, n.d.)

3.11 Natas 10

Natas system.
Hostname: http://natas10.natas.labs.overthewire.org/

3.11.1 General information

This challenge is about additional ways the command injection vulnerability can be used.

CWE classification of this vulnerability could be: CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection') and CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

CVSS calculation of this vulnerability is presented in Table 3.6

Description	score	reason
Attack Vector (AV)	Network	Attack can be done remotely
Attack Complexity (AC)	Low	No special conditions needed
Privileges Required (PR)	None	Attacker is unauthorized prior the attack
User Interaction (UI)	None	No user interaction required
Scope (S)	Unchanged	Impacted component is the same as the vulnerable component
Confidentiality Impact (C)	High	Complete loss of Confidentiality
Integrity Impact (I)	High	Complete loss of integrity
Availability Impact (A)	High	Complete loss of availability
Overall	9.8	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Table 3.10: CVSSv3 score of the issue

3.11.2 Proof of concept

The first sight of the website is very similar to the previous one. The only difference being that the site states that it is now filtering certain characters for security reasons.

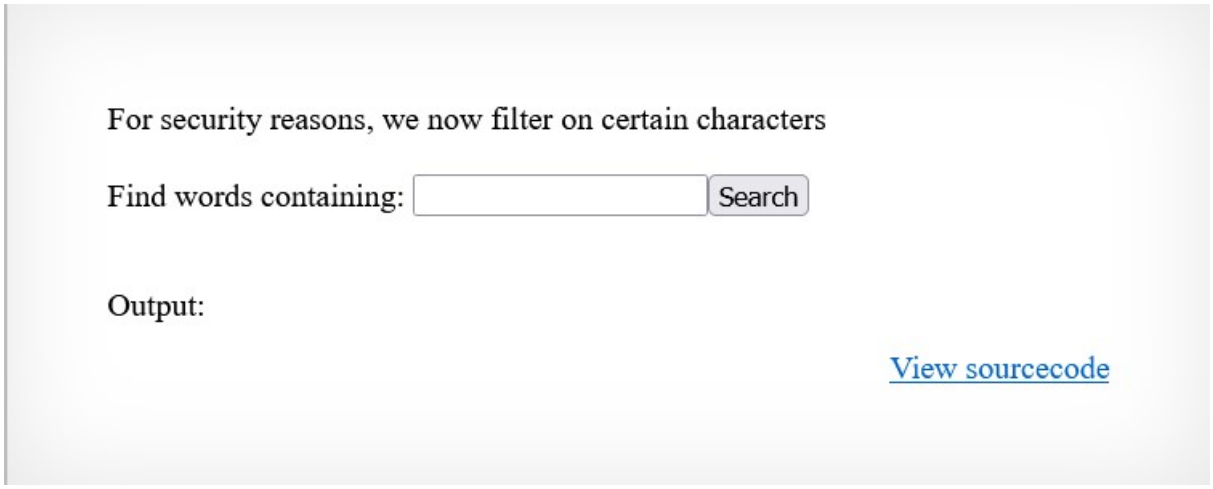

Figure 3.29: first sight natas 10

From looking at the source code, it seems that some characters are now filtered. These characters are: " [;|&] But there are still characters that are not filtered that can be used in here. For example I could simply input a /etc/natas_webpass/natas11 This would search for the letter in both /etc/natas_webpass/natas11 and dictionary.txt since grep will search any given input files, which means that multiple files can be given to it.

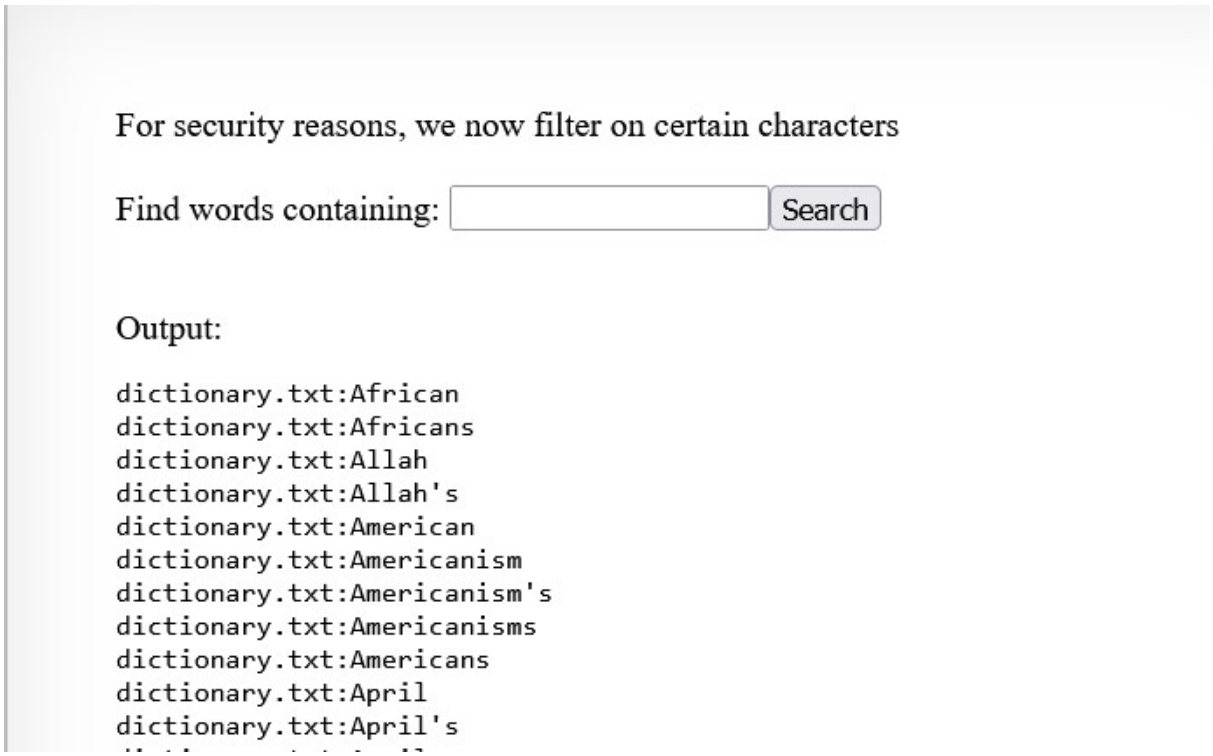

Figure 3.30: first grep

This however will not give the password since the letter "a" is not present in it. I could try to brute force the grep to give me the password. By going alphabetical I can find the next password with c /etc/natas_webpass/natas11

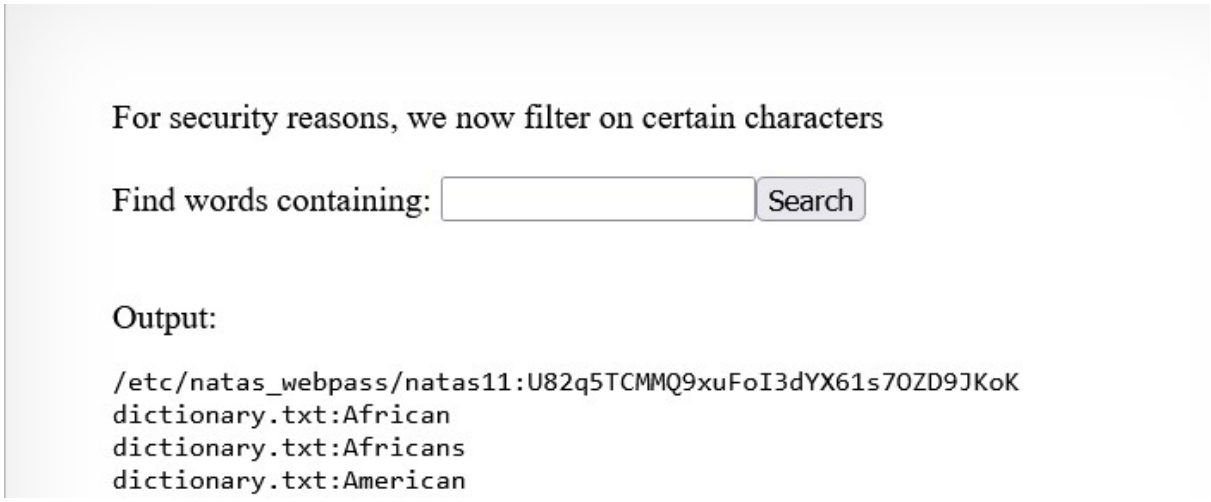

Figure 3.31: flag found with bruteforce

This could be somewhat time consuming and requires at least certain amount of quessing. Another way would be to just search for all text in the natas11 directory. This can be done with command: .* /etc/natas_webpass/natas11 # (I m using the "#" at the end since this would also print everything in the dictionary and its quite long) This prints everything in the direcotry including our password

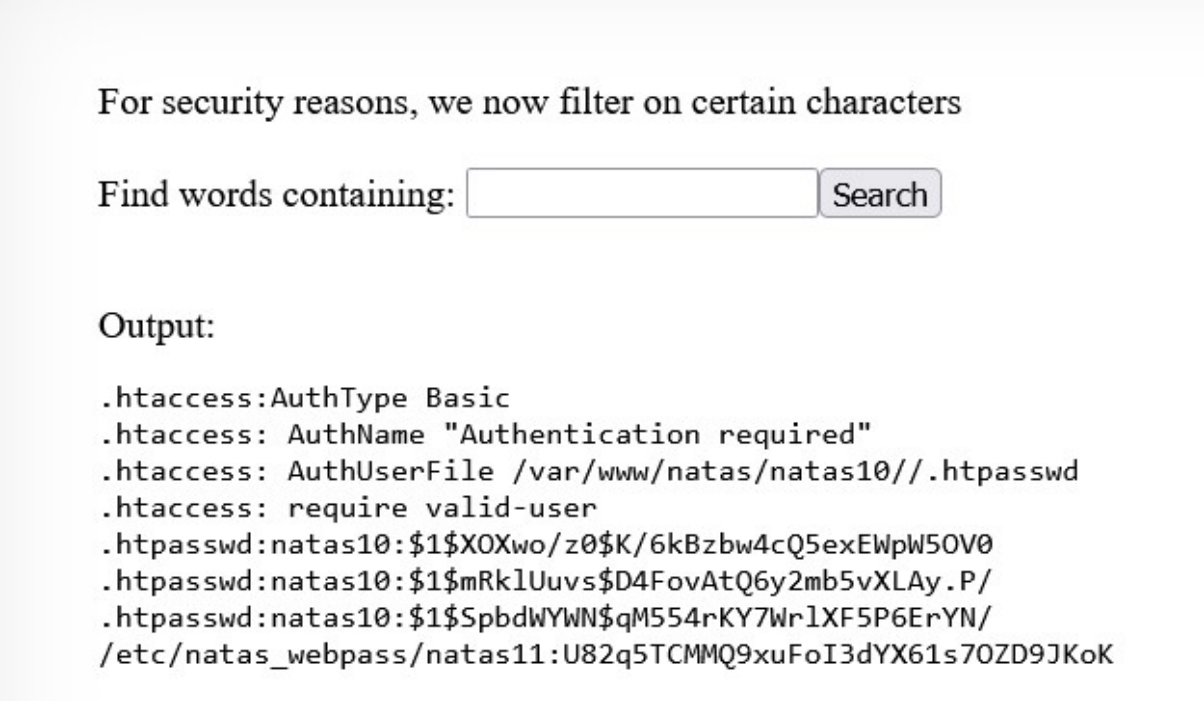

Figure 3.32: flag natas 10

3.11.3 Proposed solutions

One of the ways to prevent Command Injection is to just never call out to OS commands from the application-layer code. Instead safer implementations should be pursued. In cases that the OS commands can not be avoided a strong input validation should be in use. These can be for example, making sure that the input is a number, using white-lists of permitted values and making sure only alphanumeric values are used in input. (PortSwigger, n.d.) Additional defense could include making sure that the applications run at the lowest possible privileges as the commands will be executed with the same privileges as the application has. (OWASP, n.d.)

Bibliography

Common Weakness Enumeration, a. n.d. “CWE-548: Exposure of Information Through Directory Listing,” https://cwe.mitre.org/data/definitions/548.html.

———. n.d. “CWE-565: Reliance on Cookies without Validation and Integrity Checking,” https://cwe.mitre.org/data/definitions/565.html.

Dumanhug, AJ. 2020. “Hacking Applications with File Inclusion,” https://blog.secuna.io/hacking-applications-with-file-inclusion/.

Maurya, Deepak, Kumar. 2020. “How To Prevent Brute Force Attacks With 8 Easy Tactics,” https://ethicalhacs.com/dvwa-command-injection/.

OWASP. n.d. “OS Command Injection Defense Cheat Sheet,” https://cheatsheetseries.owasp.org/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.html.

PortSwigger. n.d. “Directory listing,” https://portswigger.net/kb/issues/00600100_directory-listing.

———. n.d. “OS command injection,” https://portswigger.net/web-security/os-command-injection.

Shyam, Oza. 2021. “File Inclusion Vulnerabilities — Web-based Application Security, Part 9,” https://spanning.com/blog/file-inclusion-vulnerabilities-lfi-rfi-web-based-application-security-part-9/.

Zhong, Weilin. 2021. “Command Injection,” https://owasp.org/www-community/attacks/Command_Injection.