**Remote walkthrough**

# **Index**

[Index 1](#_Toc169530817)

[List of pictures 1](#_Toc169530818)

[Disclaimer 2](#_Toc169530819)

[Reconnaissance 2](#_Toc169530820)

[Initial foothold 3](#_Toc169530821)

[User flag 5](#_Toc169530822)

[Privilege escalation 7](#_Toc169530823)

[APPENDIX A – CVE 9](#_Toc169530824)

[CVE-2019-25137 9](#_Toc169530825)

# **List of pictures**

[Figure 1 - nMap scan results 2](#_Toc169530829)

[Figure 2 - rpcinfo command results 3](#_Toc169530830)

[Figure 3 - nMap scan script NFS 4](#_Toc169530831)

[Figure 4 - Command to mount an NSF remote volume 4](#_Toc169530832)

[Figure 5 - Umbraco file strings 4](#_Toc169530833)

[Figure 6 - Web application on port 80 5](#_Toc169530834)

[Figure 7 - Login page 5](#_Toc169530835)

[Figure 8 - Umbraco reserved area 6](#_Toc169530836)

[Figure 9 - Umbraco exploit command 6](#_Toc169530837)

[Figure 10 - User shell 6](#_Toc169530838)

[Figure 11 - User flag 7](#_Toc169530839)

[Figure 12 - Teamviewer proof 7](#_Toc169530840)

[Figure 13 - TeamViewer registry content 7](#_Toc169530841)

[Figure 14 - Password decrypted 8](#_Toc169530842)

[Figure 15 - Available shares 8](#_Toc169530843)

[Figure 16 - Connection as administrator 8](#_Toc169530844)

[Figure 17 - Download the root flag 9](#_Toc169530845)

[Figure 18 - Root flag 9](#_Toc169530846)

# **Disclaimer**

I do this box to learn things and challenge myself. I’m not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who’re willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

# **Reconnaissance**

The results of an initial nMap scan are the following:

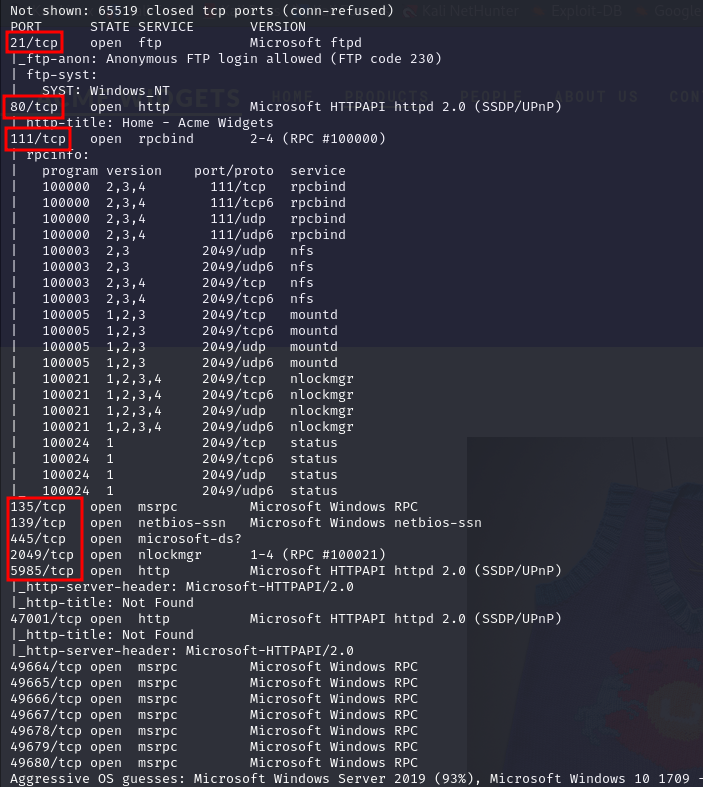


Figure 1 - nMap scan results

Open ports are 21, 80, 111, 135, 139, 445, 2049, 5985, 47001, 49664, 49665, 49666, 49667, 49678, 49679, 49680. So, this machine has FTP service enabled, three web applications on port 80, 5985 and 47001, NetBIOS service enabled on port 139, probably SMB enabled on 445 (even if it is not recognized by nMap), nlockmgr service on port 2049 and all other ports are relative to RPC service. Also, nMap guesses Microsoft Windows Server 2019 as OS.

# **Initial foothold**

First thing I tried was connecting to ftp in anonymous way. Even if it worked, I didn’t find any interesting information. After that, I tried to retrieve some information from RPC running the following command:

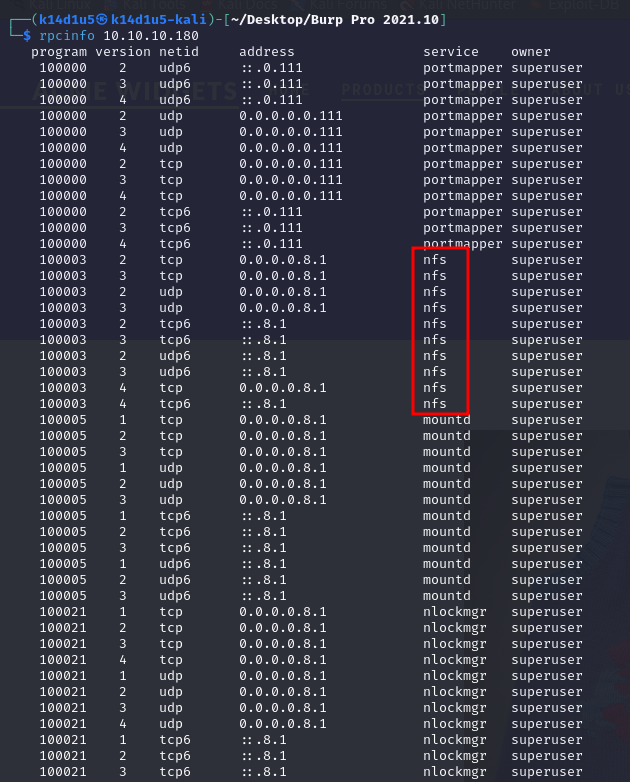


Figure 2 - rpcinfo command results

By the results of this command, I see that NFS is enabled and running. This means that I can download and upload files. Since I found NFS service, I tried to find more information about it. So, I run again nMap to execute script to list files via NFS service:

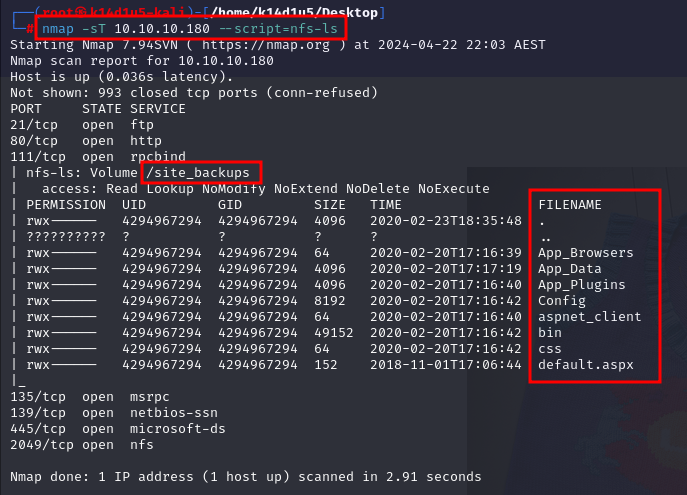


Figure 3 - nMap scan script NFS

Since it worked, this means I didn’t need an authentication and authorization. Also, this means that NFS version is 2. Another interesting information is the volume name . Since I found it, I can mount it on my Kali machine running the following command:



Figure 4 - Command to mount an NSF remote volume

I decide to mount it in the path. Exploring files I found an interesting one named **Umbraco.sdf**. I can analyze it extracting strings from it:

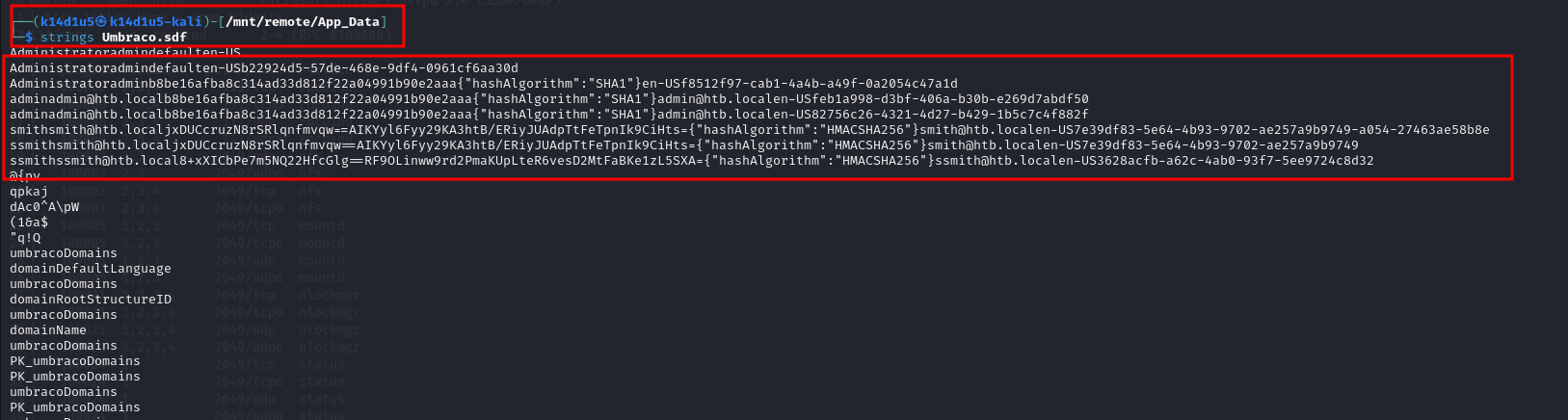


Figure 5 - Umbraco file strings

It looks like there are some hashed passwords! This can be interesting because if I explore the web application on port 80, it has the following page:

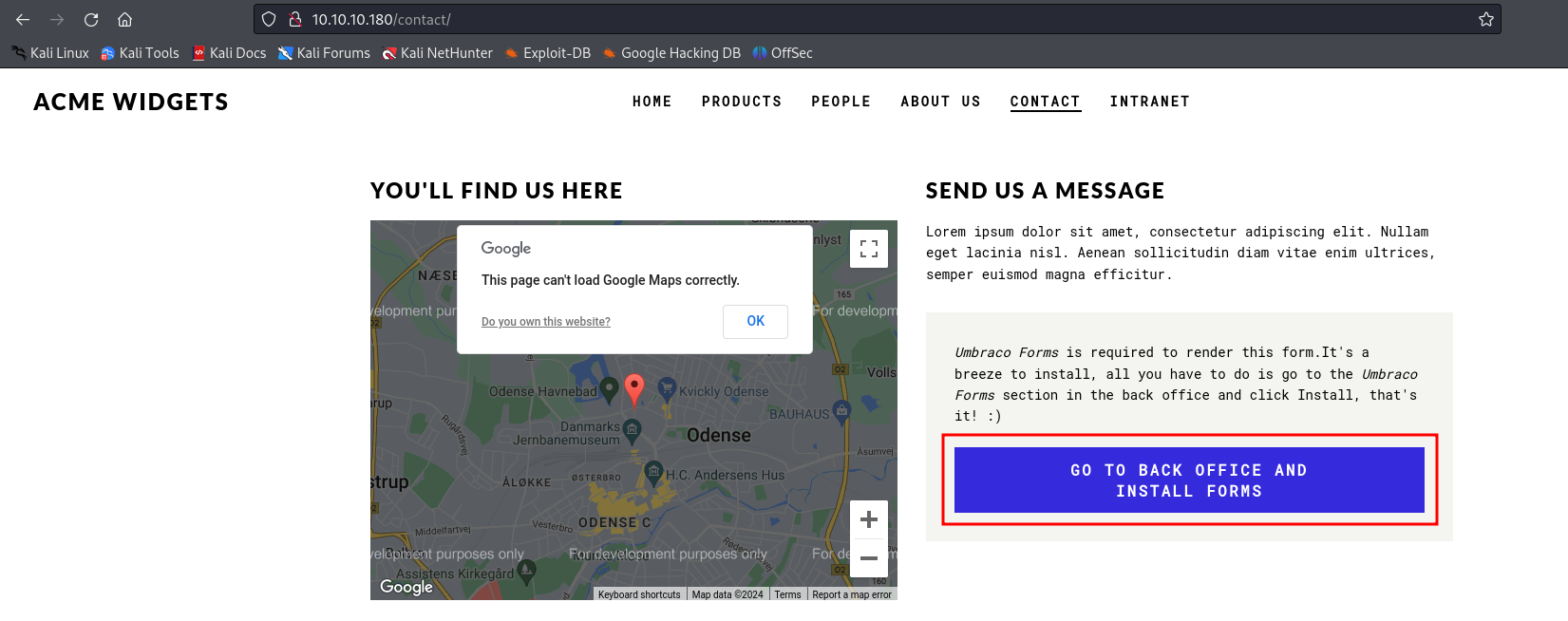


Figure 6 - Web application on port 80

In this page the blue button browses me in an Umbraco login page:

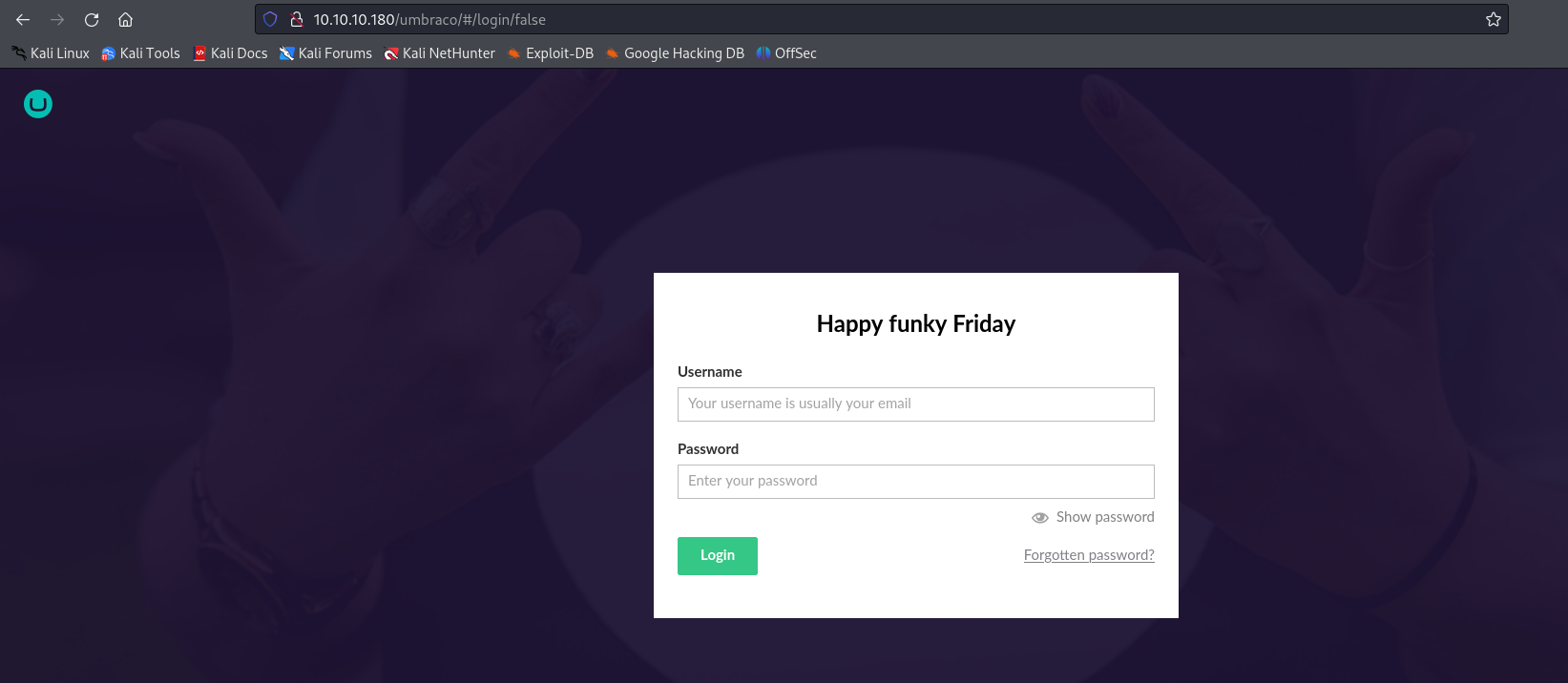


Figure 7 - Login page

# **User flag**

At this point, I tried to crack the hashes using **crackstation** and I got a match! So, I used the credentials to login in the web application:

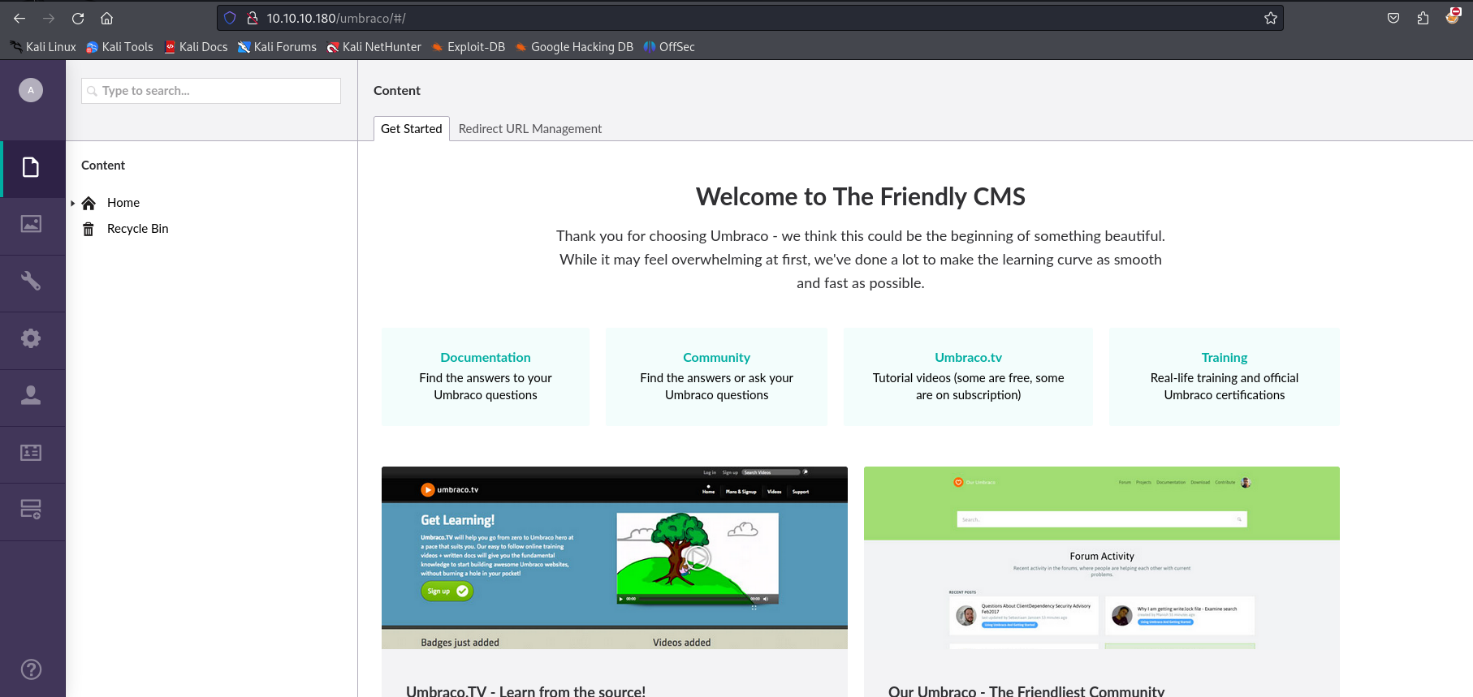


Figure 8 - Umbraco reserved area

Now, I need to find a way to exploit it. I looked for it on the Internet and I found the CVE-2019-25137. So, I downloaded the exploit, and run it:

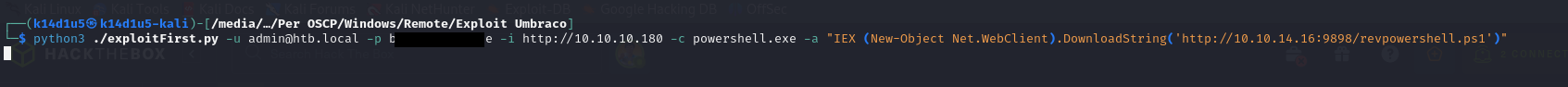


Figure 9 - Umbraco exploit command

In this way I obtained a shell:

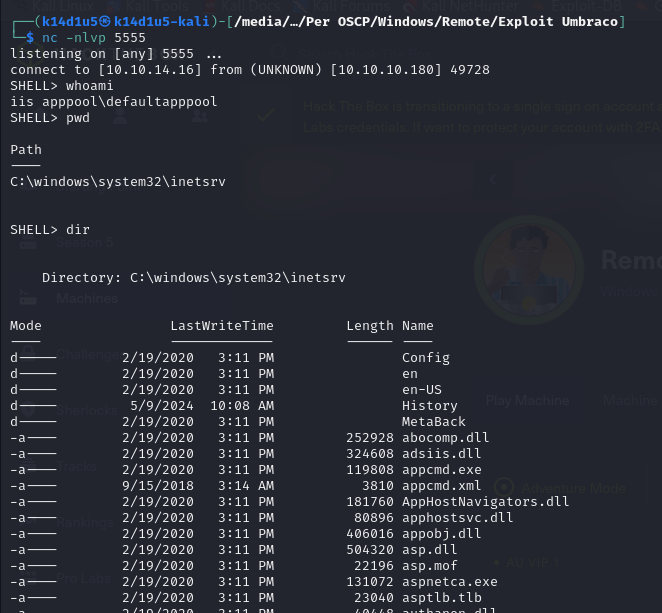


Figure 10 - User shell

At this point, I just need to retrieve the user flag:

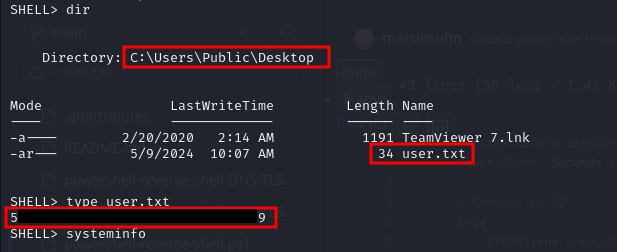


Figure 11 - User flag

# **Privilege escalation**

This is the time to escalate my privileges. The first information I found is that TeamViewer version 7 is installed on the target system:

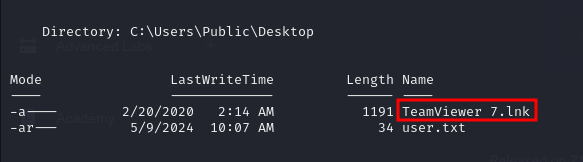


Figure 12 - TeamViewer proof

So, I analyzed everything it is relative to TeamViewer. In particular, I read its registry:

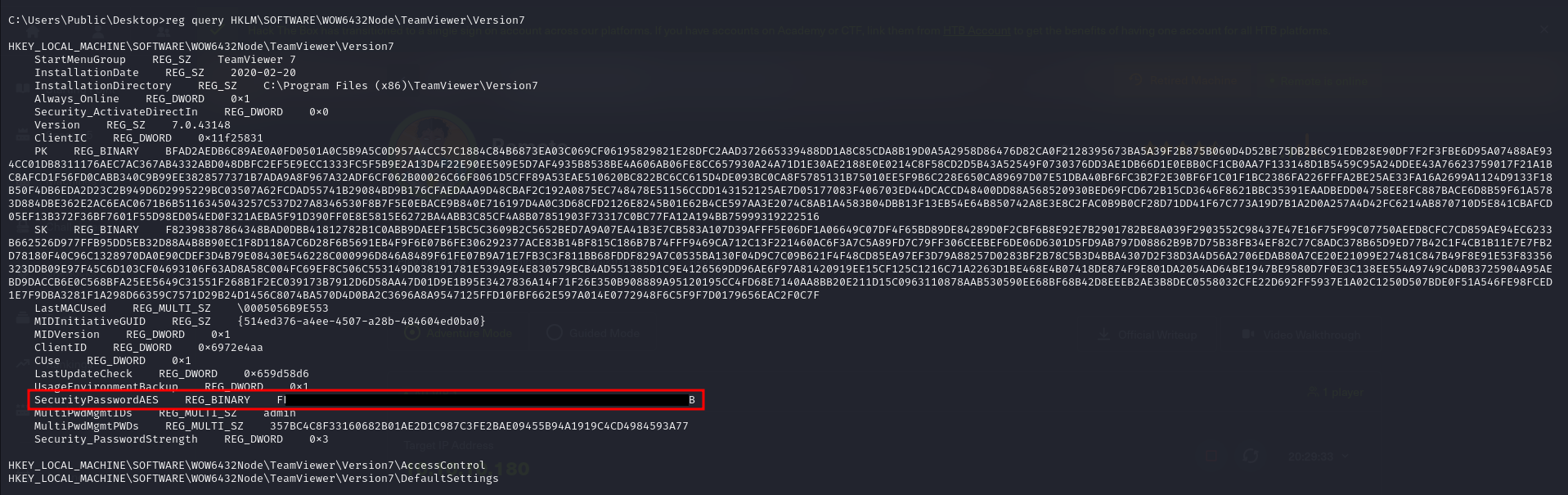


Figure 13 - TeamViewer registry content

In this way, I found a hashed password. To decrypt it, I used a python script:

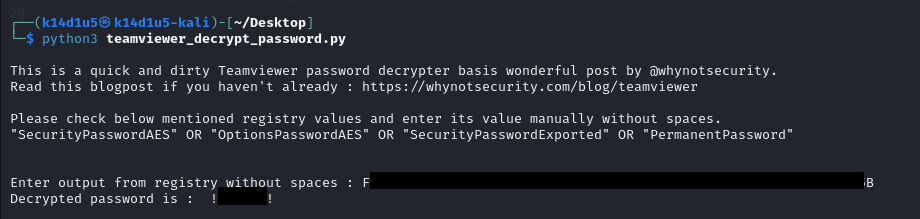


Figure 14 - Password decrypted

At this point, I remembered that SSH is not available on the machine, but probably SMB is. So, I tried an SMB connection using the credentials just found to list the available shares:



Figure 15 - Available shares

So, I proceeded connecting to the target machine leveraging the C$ share:

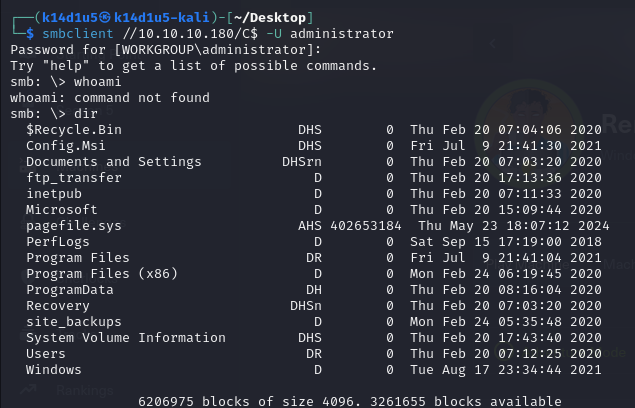


Figure 16 - Connection as administrator

Since I have a SMB connection, I can’t read file directly, but I need to download it. All I need to do now, is download the root flag:

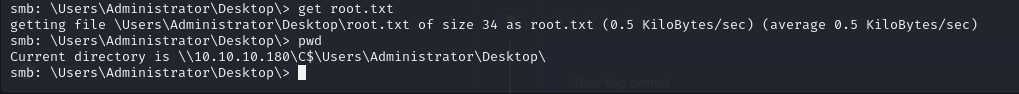


Figure 17 - Download the root flag

Last thing to do is read it:

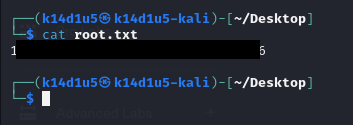


Figure 18 - Root flag

# **APPENDIX A – CVE**

## **CVE-2019-25137**

CVE-2019-25137 is a XSLT injection vulnerability in Umbraco CMS. The vulnerability is present in the XSLT (Extensive Stylesheet Language Transformations) Visualizer webpage. The vulnerable URI for this webpage is **/umbraco/developer/Xslt/xsltVisualize.aspx**. Successful exploitation of the XSLT Visualizer can result in C# code being executed on the targeted system. To exploit the vulnerability, a user requires legitimate administrator credentials to the Umbraco CMS. The proof of concept for CVE-2019-25137 uses the **msxsl:script** element. This element allows for additional programming languages to be used in XSLT transformations, such as C#.