**Support walkthrough**

# **Index**

[Index 1](#_Toc194230472)

[List of pictures 1](#_Toc194230473)

[Disclaimer 2](#_Toc194230474)

[Reconnaissance 2](#_Toc194230475)

[Initial foothold 3](#_Toc194230476)

[User flag 5](#_Toc194230477)

[Privilege escalation 6](#_Toc194230478)

[Personal comments 12](#_Toc194230479)

[References 12](#_Toc194230480)

# **List of pictures**

[Figure 1 - nMap scan results 2](#_Toc194230444)

[Figure 2 - SMB anonymous login 3](#_Toc194230445)

[Figure 3 - support-tools share content 3](#_Toc194230446)

[Figure 4 - LDAP query in UserInfo.exe 3](#_Toc194230447)

[Figure 5 - Password decoder function 4](#_Toc194230448)

[Figure 6 - Hardcoded encrypted password 4](#_Toc194230449)

[Figure 7 - Hardcoded decription key 5](#_Toc194230450)

[Figure 8 - Decoded password 5](#_Toc194230451)

[Figure 9 - LDAP information 6](#_Toc194230452)

[Figure 10 - User in interesting group 6](#_Toc194230453)

[Figure 11 - BloodHound command 6](#_Toc194230454)

[Figure 12 - BloodHound possible privilege escalation path 7](#_Toc194230455)

[Figure 13 - ms-ds-machineaccountquota attribute check 7](#_Toc194230456)

[Figure 14 - Windows version check 7](#_Toc194230457)

[Figure 15 - Hostname 8](#_Toc194230458)

[Figure 16 - New machine created 8](#_Toc194230459)

[Figure 17 - New machine SID value 8](#_Toc194230460)

[Figure 18 - Security descriptor 8](#_Toc194230461)

[Figure 19 - Password for new machine 9](#_Toc194230462)

[Figure 20 – Retrieved tickets (part 1) 9](#_Toc194230463)

[Figure 21 – Retrieved tickets (part 2) 10](#_Toc194230464)

[Figure 22 - Retrieved tickets (part 3) 10](#_Toc194230465)

[Figure 23 - Retrieved tickets (part 4) 11](#_Toc194230466)

[Figure 24 - Decoding ticket 11](#_Toc194230467)

[Figure 25 - Ticket converted in ccache format 11](#_Toc194230468)

[Figure 26 - Root shell and flag 11](#_Toc194230469)

# **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 are willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Just to say: I am not an English native person, so sorry if I did some grammatical and syntax mistakes.

# **Reconnaissance**

The results of an initial nMap scan are the following:

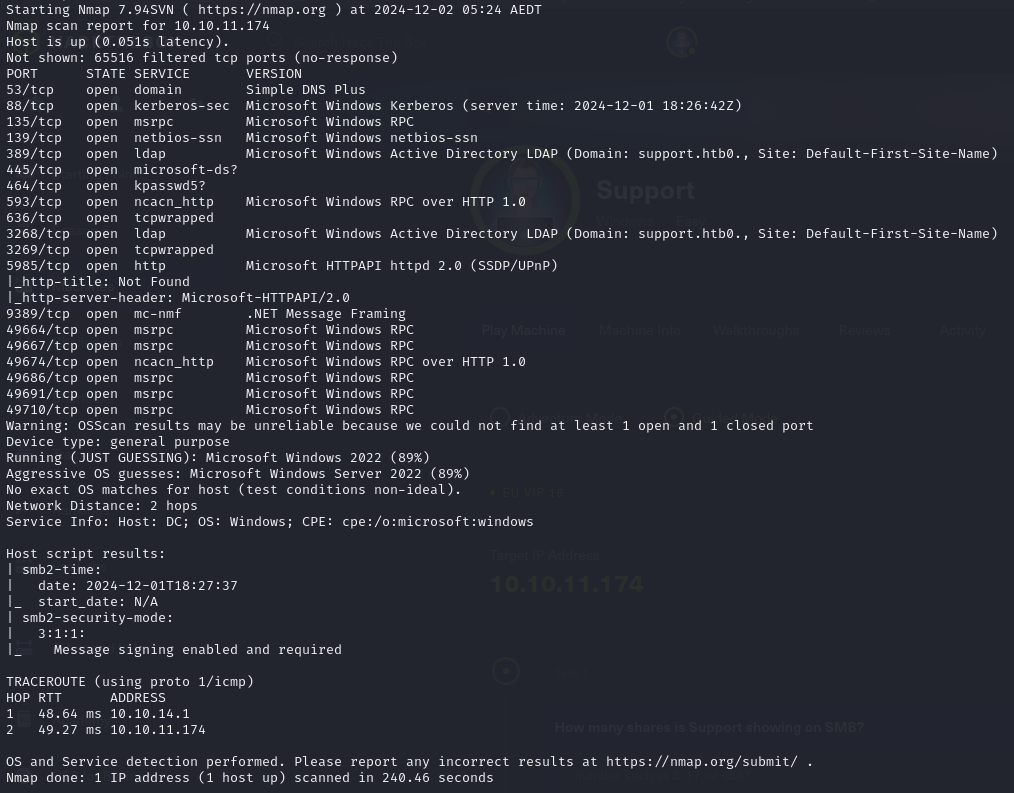


Figure 1 - nMap scan results

Open ports are 53, 88, 135, 139, 389, 445, 464, 593, 636, 3268, 3269, 5985, 9389, 49664, 49667, 49674, 49686, 49691, 49710. Enabled services are DNS (53), Kerberos (88) RPC (135, 593, 49664, 49667, 49674, 49686, 49691, 49710), NetBIOS (139), LDAP (389, 3268), Samba (445), .NET (9389). Unknown services are enabled on ports 464, 636, 3269. Also, a web application is running on port 5985. Lastly, nMap recognized Windows as Operative System and probably it can be Windows Server 2022. Another interesting information I obtained by this nMap scan is that SMB version 2 has message signing enabled and required.

# **Initial foothold**

One the most interesting enabled service found is Samba. So, I tried to explore it. First of all, I verified I can connect to this service via an anonymous login:



Figure 2 - SMB anonymous login

Among with the default shares, I found another one named . So, I tried to explore it:

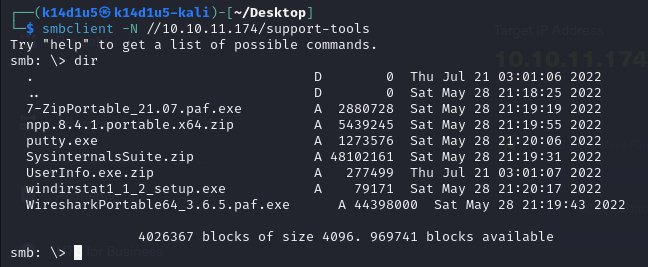


Figure 3 - support-tools share content

This share contains some windows tool to perform analysis. Also, one of these tools (UserInfo.exe) is not commercial, but custom. Since it is custom, I was willing to investigate it more. To do so, I need a decompiler. I used ILSpy, but the Windows version. In this way, I found out that this program provides very interesting information. The first one is that it performs a LDAP query. In the command I found the username used and I saw that retrieve the password by another function:

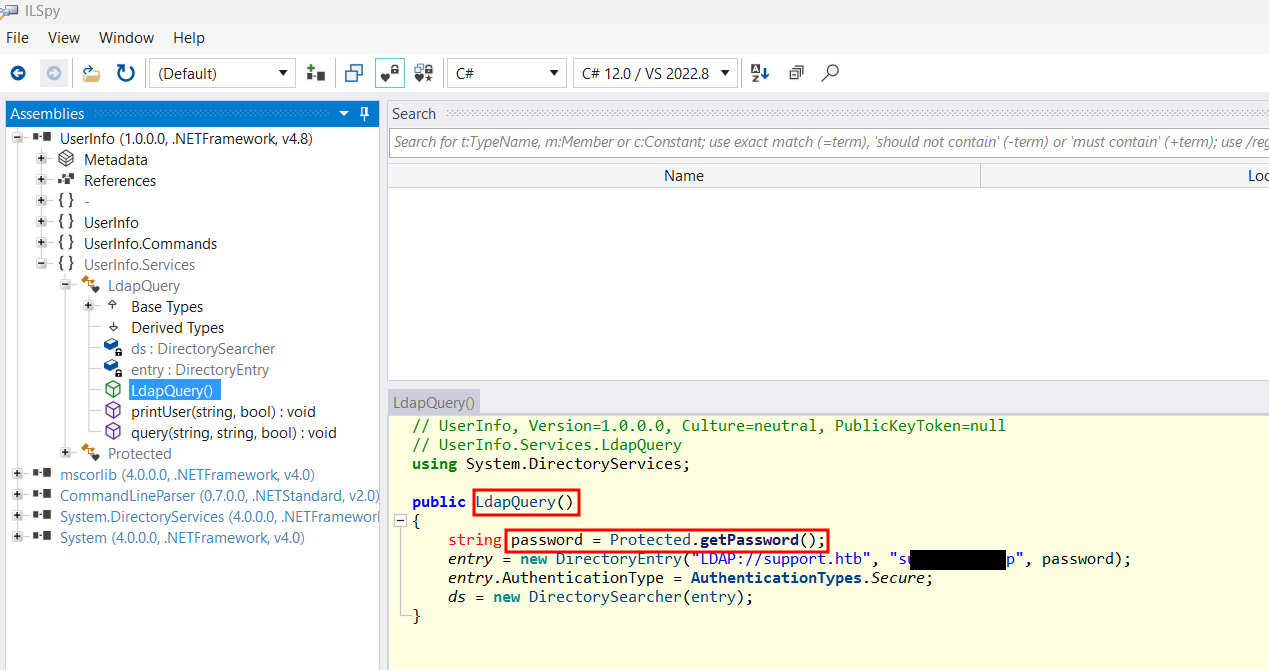


Figure 4 - LDAP query in UserInfo.exe

At this point I need the password. I kept to explore the decompiled program and, in particular, the function I found. This function performs a password decode and use an encrypted password and a key, as shown in the following figure:

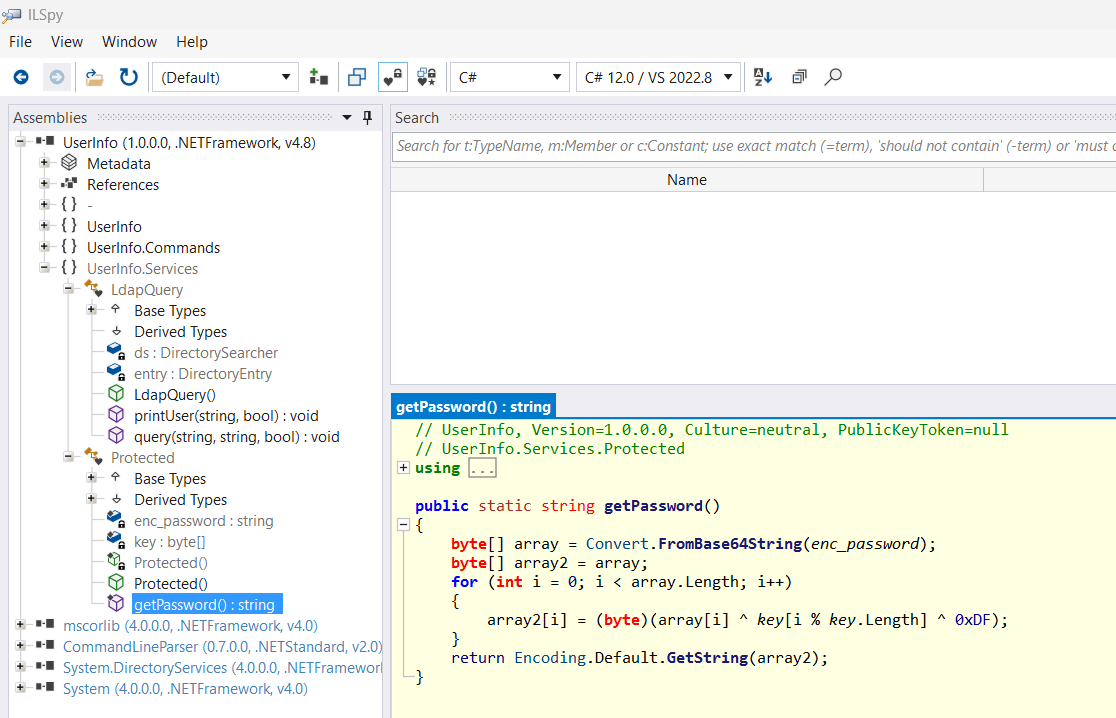


Figure 5 - Password decoder function

So, I kept to explore the program to find the data I needed. In fact, I was able to find the encrypted password and the key, as shown in the following pictures:

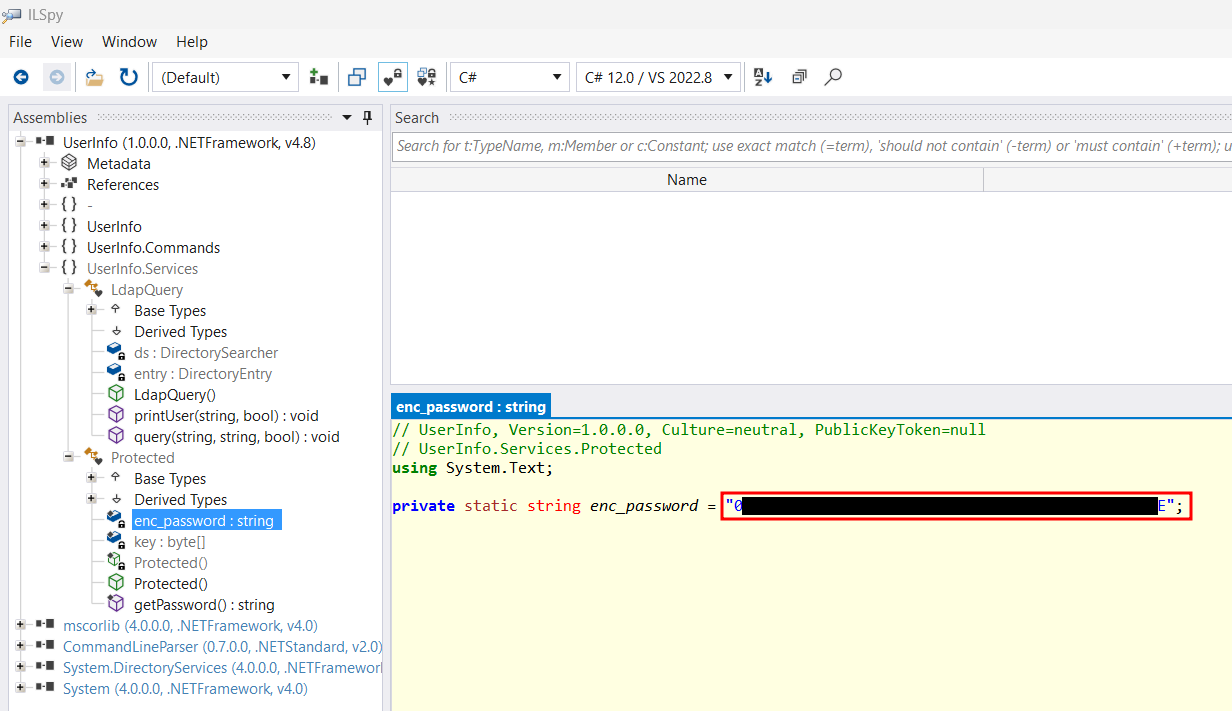


Figure - Hardcoded encrypted password

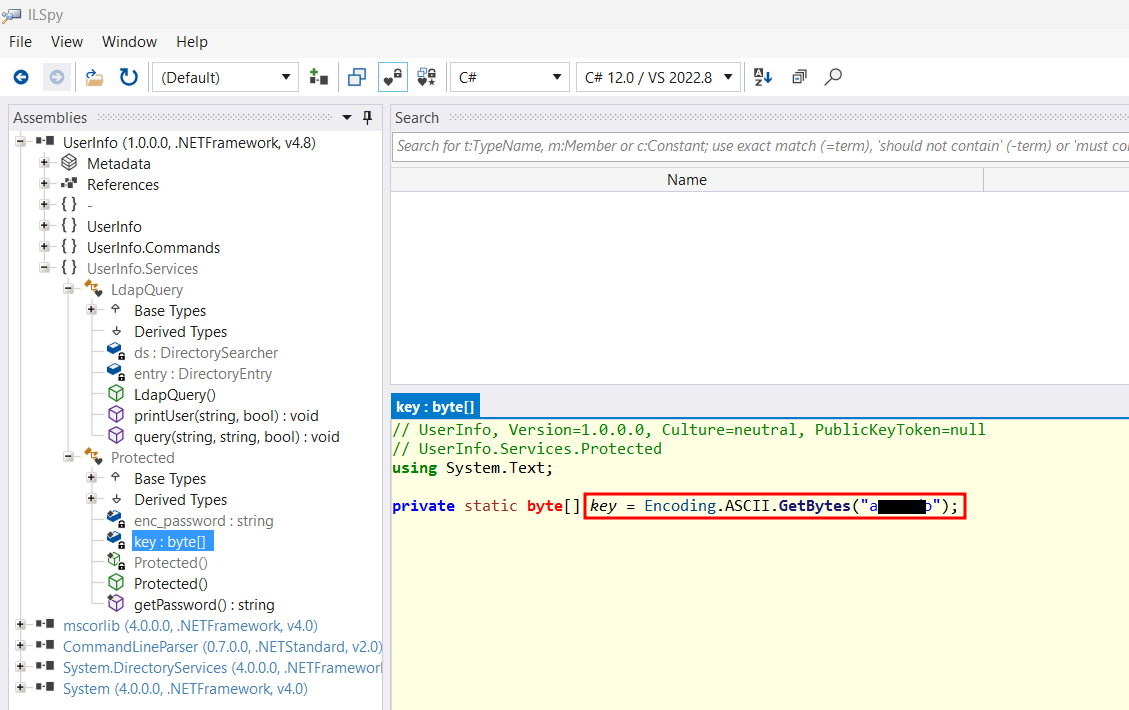


Figure 7 - Hardcoded decryption key

# **User flag**

At this point I had all information I needed. However, I still needed to decrypt the password. To do so, I manually converted in Python the function I found in the UserInfo.exe file. I run this script using the encrypted password and key I found in the code. In this way, I had the clear text password:

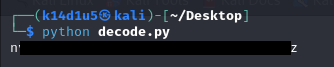


Figure 8 - Decoded password

At this point I had credentials to perform LDAP queries. So, I tried to retrieve information via LDAP, running the command (I obfuscated username and password, you need to insert there the credentials found in UserInfo.exe and decrypted password). I analyzed the HTML generated and I found list of users, for example. However, I still need other information. After I spent a huge amount of time analyzing the HTML files, I decided to run a different command to retrieve LDAP information (in the personal comment section I will explain more details about the reason why the previous command wasn’t good enough to complete this box):

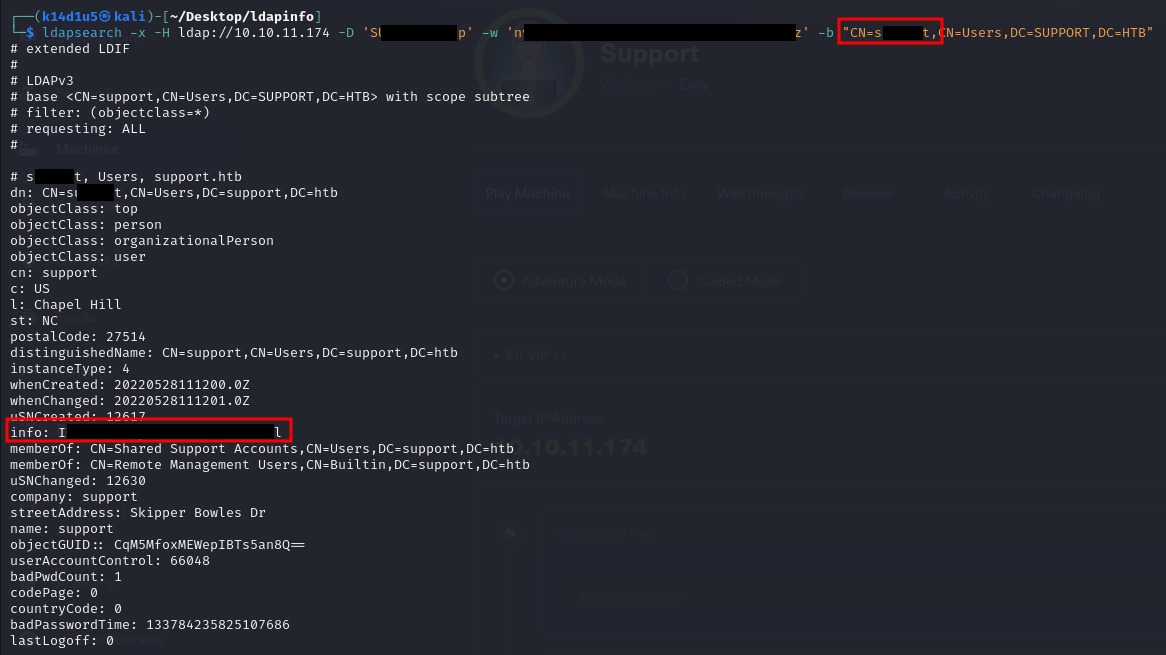


Figure 9 - LDAP information

Finally, I was able to obtain a shell on the target via WinRM and retrieve the user flag, but I forgot to take the screenshot. I was able to run WinRM because of the 5985 (HTTP, 5986 HTTPS) port is open. These ports are used for WinRM starting by Windows 7. In previous Windows versions the port used are 80 HTTP and 443 HTTPS.

# **Privilege escalation**

At this point I uploaded WinPeas script on the target and I run it. In this way I found an interesting group:



Figure 10 - User in interesting group

The next step was analyzing which privileges the user has in Active Directory. To do so, I used BloodHound. So, I generated all files needed to perform the analysis running the following command:

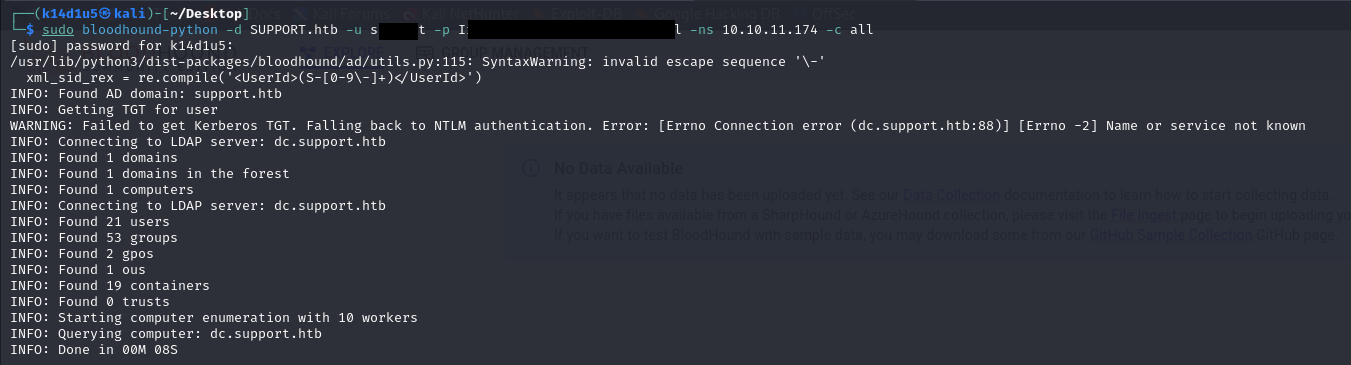


Figure 11 - BloodHound command

I imported the generated files in the BloodHound web interface and I generated a possible privilege escalation path, as shown in the following picture:

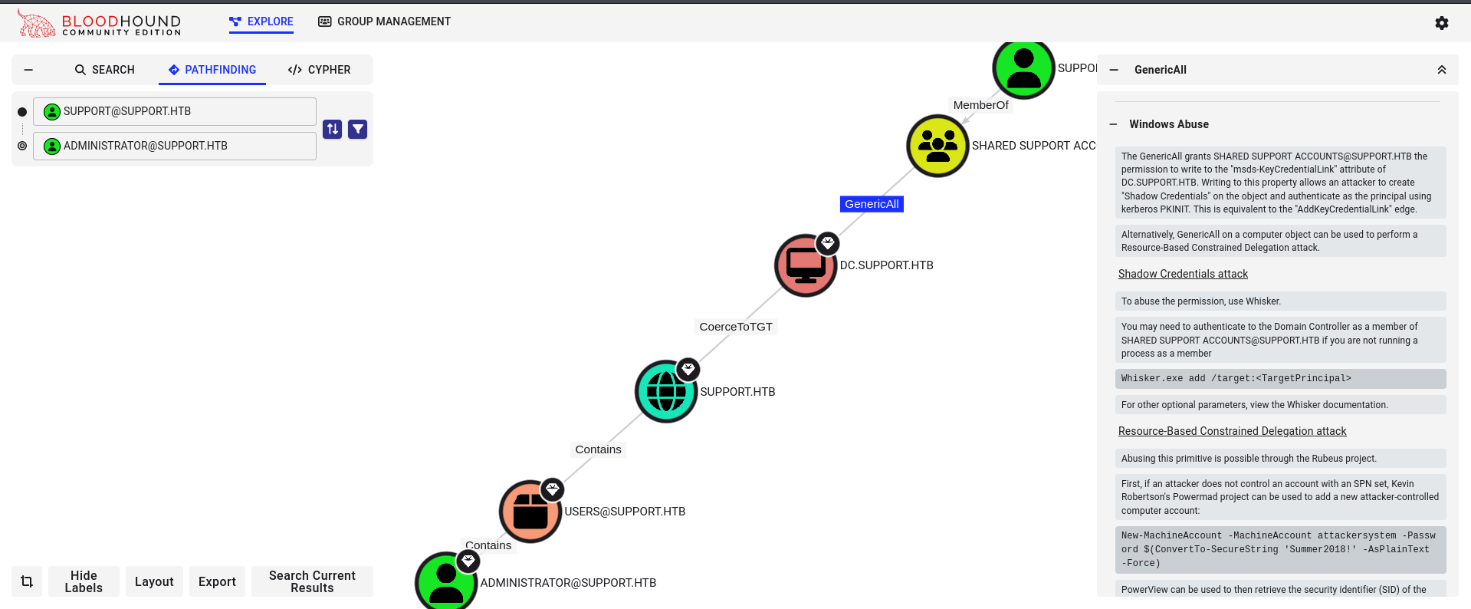


Figure 12 - BloodHound possible privilege escalation path

In particular, I had the privilege on the Domain Controller and I could be able to perform a attack. To do it, I need to check if I had enabled the attribute. To obtain this information, I run a PowerMad command. PowerMad PowerShell module must be imported, so I uploaded it on the target and I run the command :



Figure 13 - ms-ds-machineaccountquota attribute check

Luckly, the was set with a number higher than 0 (10), this means that the user was able to create or add 10 machine objects to the domain. Also, another requirement is that the Windows version needed to be Windows 2012 or higher. I checked it running the following command:

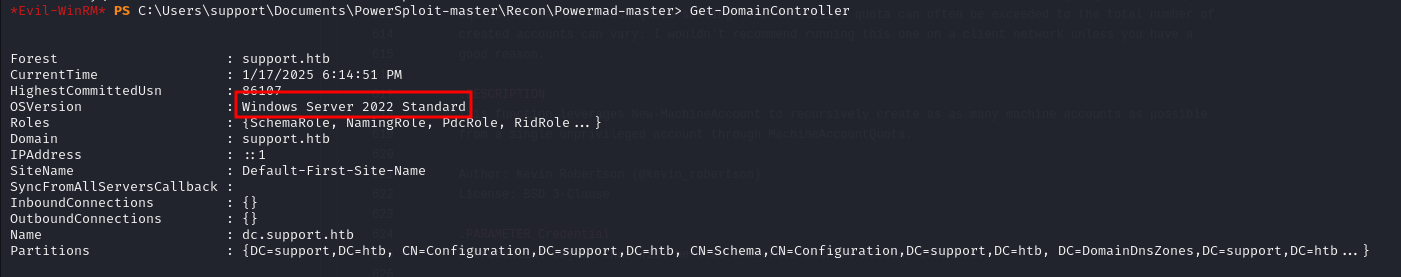


Figure 14 - Windows version check

The last requirement to check is about the attribute. The attack can be successfully performed if this attribute was NOT set. I first needed the hostname:



Figure 15 - Hostname

At this point, I run the command . Luckly, all requirements I needed was satisfied. So, I just implement the attack. The first step was creating a new machine:

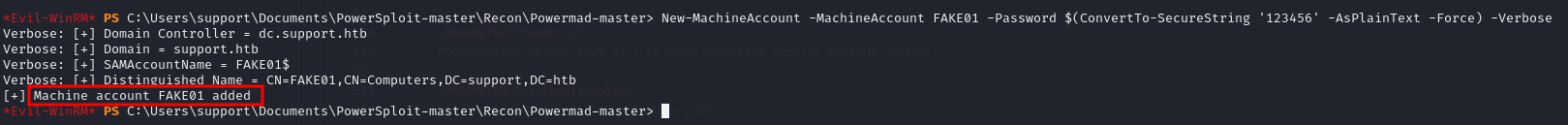


Figure 16 - New machine created

Now, I needed to retrieve its SID value:

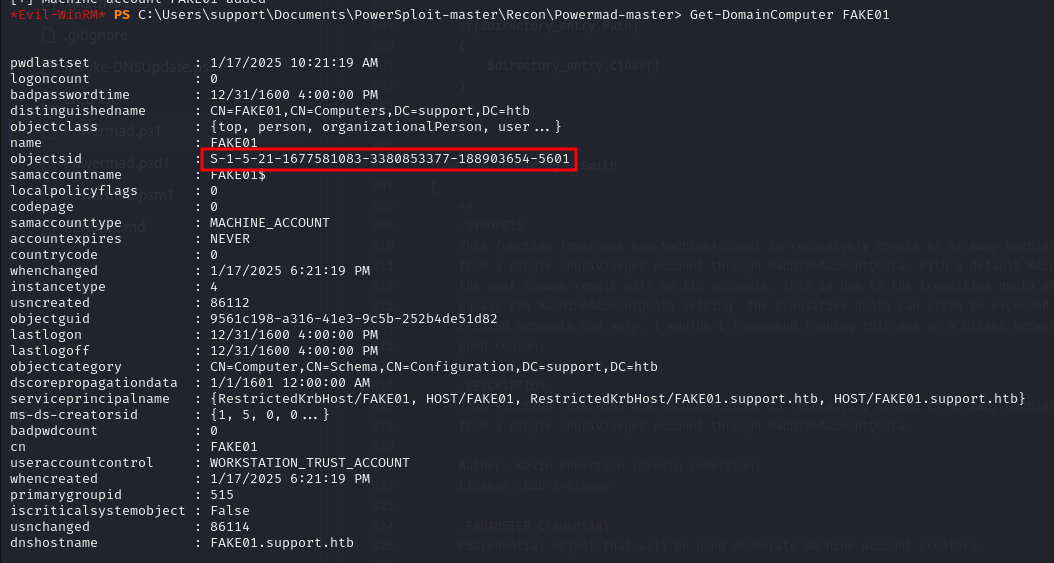


Figure 17 - New machine SID value

The new machine just created needed a security descriptor. I was able to give it one running the commands:

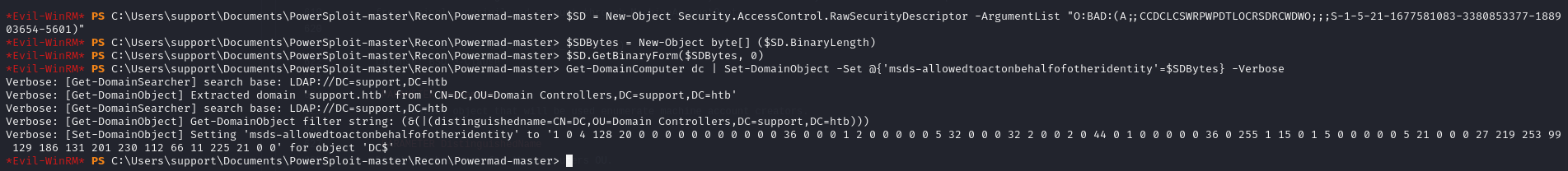


Figure 18 - Security descriptor

I was able to execute with success these commands because of the user had privilege on the target machine. However, I just needed the writing permission. Also, I can double check that the security descriptor was assigned to the new machine running the command . At this point I needed to create a password for the new machine. To do so, I uploaded Rubeus on the target machine and I run the following command:

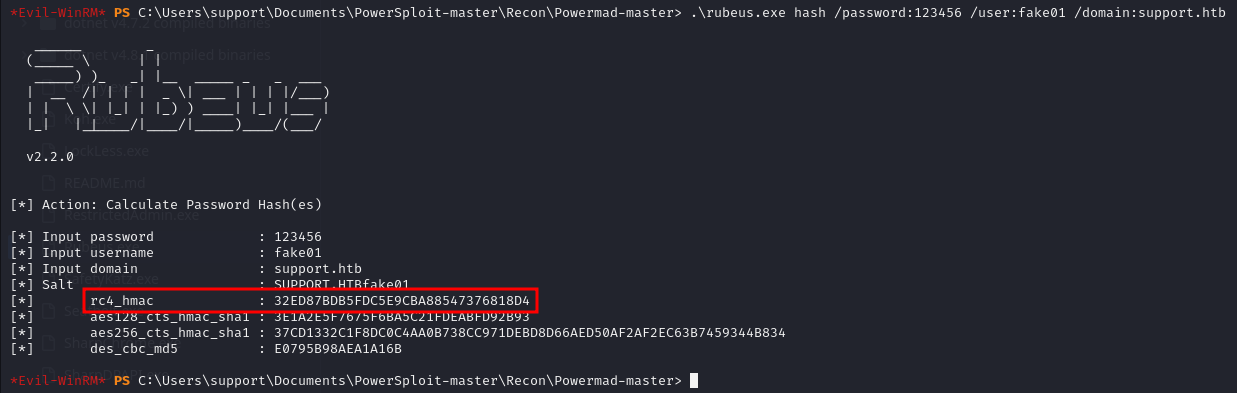


Figure 19 - Password for new machine

From this output, I took note of the value. I was able to request as Administrator for the new machine I created running the command :

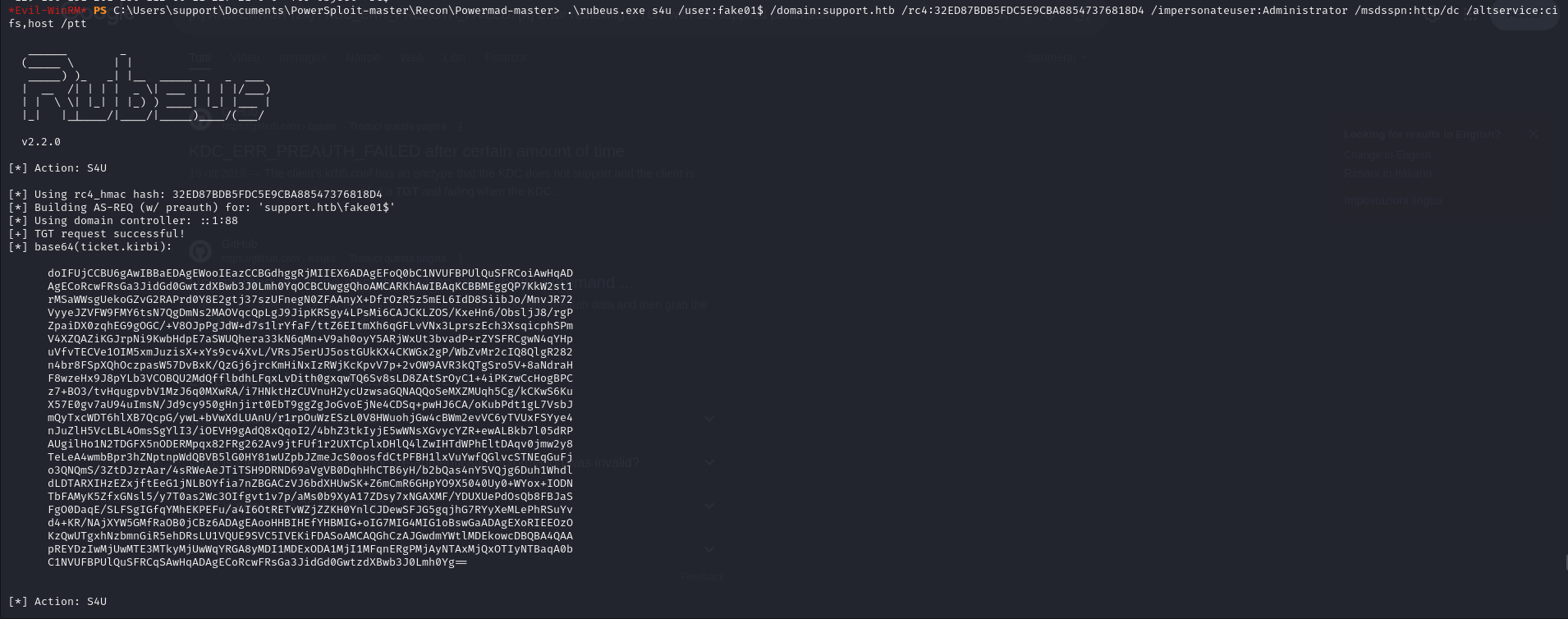


Figure – Retrieved tickets (part 1)

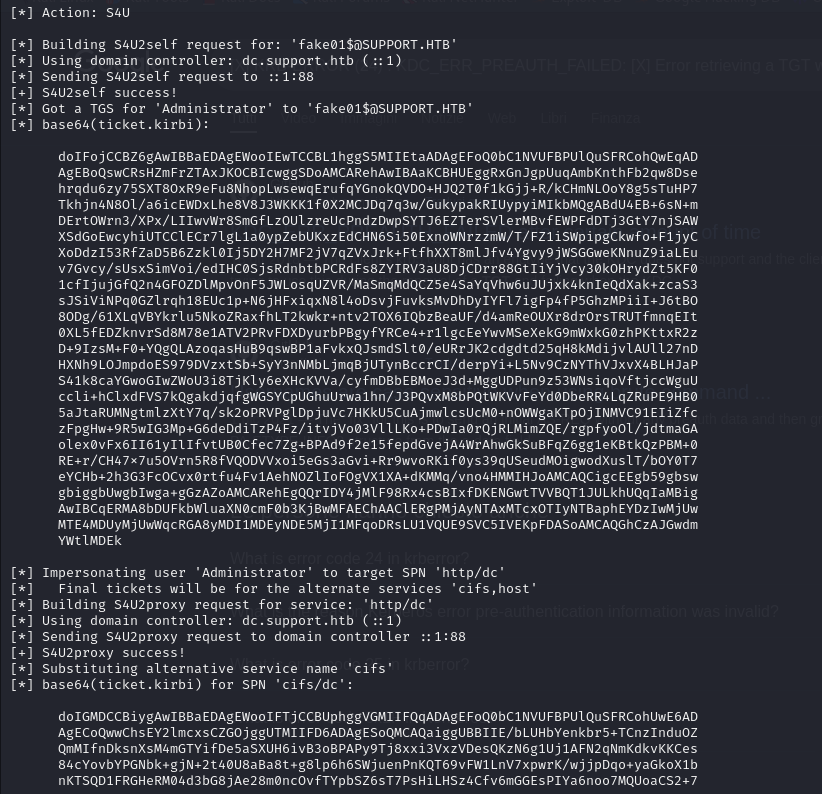


Figure – Retrieved tickets (part 2)

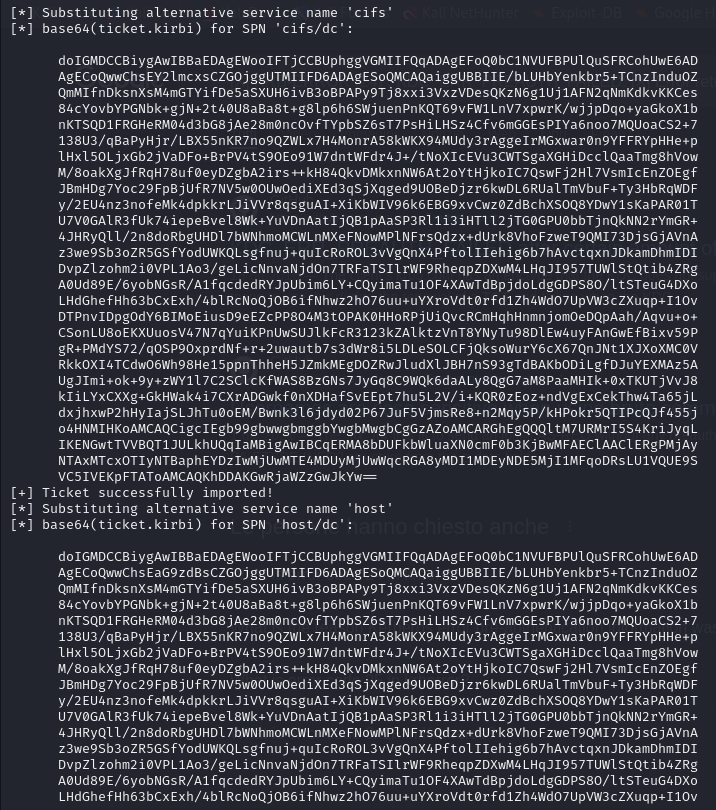


Figure - Retrieved tickets (part 3)

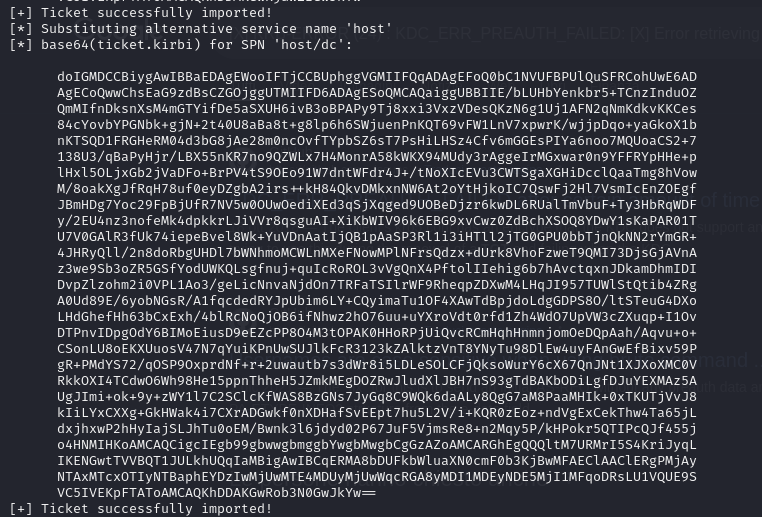


Figure 23 - Retrieved tickets (part 4)

I needed the last one ticket I retrieved. I copied it in a file named (without spaces, CR and NL, all on one line) and I tried to decode it running the command:



Figure 24 - Decoding ticket

To use it in a Kerberos authentication, I needed to convert it in ccache format:

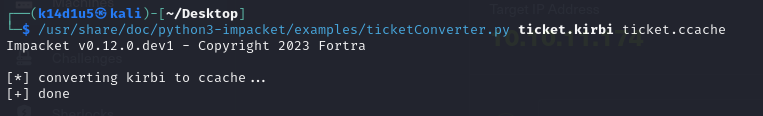


Figure 25 - Ticket converted in ccache format

Last task I needed to do was inserting an entry in the file for the dc.support.htb URL. Finally, I was able to log in the target machine as Administrator and retrieve the root flag:

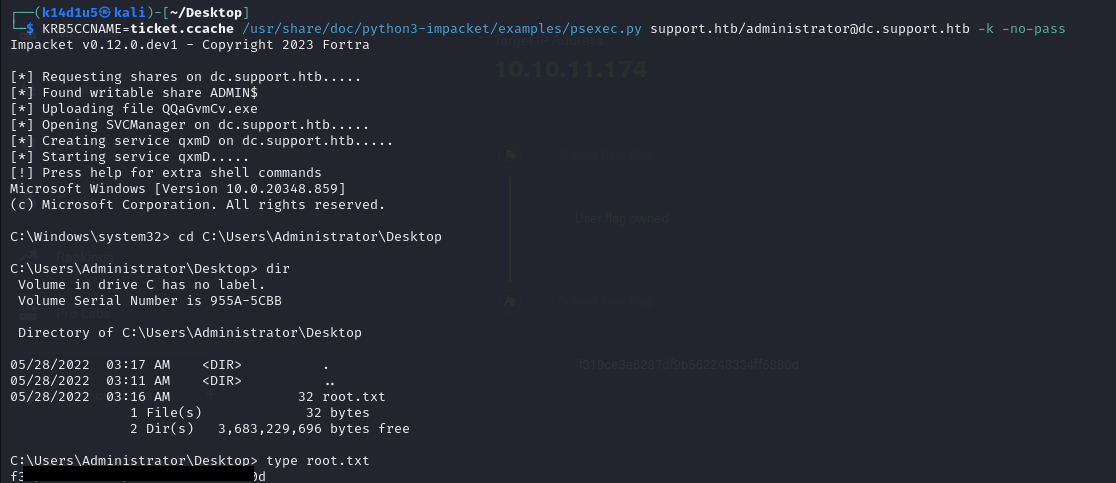


Figure 26 - Root shell and flag

# **Personal comments**

In my opinion, solving this box was quite challenging because of you need to know very important concept and have a little bit of experience about some tools. Also, I had some issues about the nMap scan (damn box, damn HackTheBox!). In particular, you need to know how to decompile an .exe file and have experience with this kind of tools to properly analyze the file. Of course, you need to identify the custom .exe and be conscious that you can find interesting information. Also, I lost a lot of time because of the . As I said in the walkthrough, I run it using the flag. Honestly, I thought that all format generated by the tools contains the same information, but it is not true. In fact, I was able to retrieve the user password by the JSON files, but not by the HTML files. And this is a little bit crazy, in my opinion. Last but not least, you need to be aware that the ticket must be converted in format to be used in the Kerberos authentication. Due to all these issues, I evaluate this box as a Medium one.

# **References**

1. CSharp syntax: <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/>;
2. ILSpy: <https://github.com/icsharpcode/AvaloniaILSpy?tab=readme-ov-file>;
3. Resource-Based Constrained Delegation attack: <https://www.ired.team/offensive-security-experiments/active-directory-kerberos-abuse/resource-based-constrained-delegation-ad-computer-object-take-over-and-privilged-code-execution>;
4. Rubeus: <https://github.com/r3motecontrol/Ghostpack-CompiledBinaries/blob/master/Rubeus.exe>.