



Mastering Windows Security and Hardening

Second Edition

Secure and protect your Windows environment from cyber threats using zero-trust security principles

Mark Dunkerley | Matt Tumbarello



Mastering Windows Security and Hardening

Second Edition

Secure and protect your Windows environment from
cyber threats using zero-trust security principles

Mark Dunkerley

Matt Tumbarello



BIRMINGHAM—MUMBAI

Mastering Windows Security and Hardening

Second Edition

Copyright © 2022 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Group Product Manager: Vijin Boricha

Publishing Product Manager: Prachi Sawant

Senior Content Development Editor: Sayali Pingale

Technical Editor: Nithik Cheruvakodan

Copy Editor: Safis Editing

Project Manager: Neil Dmello

Proofreader: Safis Editing

Indexer: Rekha Nair

Production Designer: Jyoti Chauhan

Senior Marketing Coordinator: Hemangi Lotlikar

First published: July 2020

Second edition: July 2022

Production reference: 1120722

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

978-1-80323-654-4

www.packt.com

Contributors

About the authors

Mark Dunkerley is a cybersecurity and technology leader with over 20 years of experience working in higher education, healthcare, and Fortune 100 companies. Mark has extensive knowledge in IT architecture and cybersecurity through delivering secure technology solutions and services. He has experience in cloud technologies, vulnerability management, vendor risk management, identity and access management, security operations, security testing, awareness and training, application and data security, incident and response management, regulatory and compliance, and more. Mark holds a master's degree in business administration and has received certifications through (ISC)², AirWatch, Microsoft, CompTIA, VMware, AXELOS, Cisco, and EMC. Mark has spoken at multiple events, is a published author, sits on customer advisory boards, has published several case studies, and is featured as one of Security magazine's 2022 Top Cybersecurity Leaders.

Thank you to my wife, Robin, and children, Tyne, Isley, and Cambridge, for all your continued support. To my parents for shaping me into the person I am today. My brother for his ongoing service in the British Army. My co-author, Matt, for his dedication to the success of this book. To anyone I missed, thank you! Without you all, this book would have not been possible.

Matt Tumbarello is a senior solutions architect. He has extensive experience working with the Microsoft security stack, Azure, Microsoft 365, Intune, Configuration Manager, and virtualization technologies. He also has a background working directly with Fortune 500 executives in a technical enablement role. Matt has published reviews for Azure security products, privileged access management vendors, and mobile threat defense solutions. He also holds several Microsoft certifications.

I would like to acknowledge all my peers, business partners, and coworkers, who continuously help me learn and grow. Without your support, this book wouldn't be possible.

About the reviewer

Premnath Sambasivam is a server engineer with 10 years experience in Windows, Azure, VMware, and SCCM administration. He is an MCSE Cloud Platform, and Infrastructure certified professional and Microsoft certified Azure Architect. He has developed and deployed Microsoft System Center Configuration Manager solutions to manage more than 6,000 assets in his client's environment and various VMware solutions. Premnath is a technology enthusiast who loves learning and exploring new technologies. He is currently working as a Senior Cloud Engineer for one of the major retail brands in the USA. He reviewed the books *Mastering Windows Server 2019* and *Windows Server 2022 Fundamentals*, which is also published by *Packt Publishing*.

I want to thank my wife and son for encouraging me to spend time learning. Reviewing books also refreshes our memory and lets us learn about the latest technologies and software improvements. Special thanks to my mom and dad for always being supportive.

10

Mitigating Common Attack Vectors

In this chapter, you will learn how to mitigate attack vectors that are commonly seen when standard computer communications protocols have been exploited. Once an attacker has gained access to your network, they will likely try to intercept communications and insert themselves in an attempt to gain a foothold. First, we will discuss different types of Adversary-in-the-Middle techniques and how they can be used to intercept communications, poison responses, capture user passwords, and relay authentication processes to access other systems. We will also discuss how network protocols such as mDNS, NetBIOS, LLMNR, WPAD, SMB, ARP, and IPv6 can be used to trick an unknowing victim into redirecting communications to the attacker's host and fool them into providing credentials.

Then, we will discuss protecting against lateral movement and privilege escalation. We will look at how a compromised standard user account can be used to identify systems and different ways to protect against reconnaissance activities. After that, we will review exploits that target Kerberos authentication and where attackers look to access credentials stored on the host system and in Active Directory. Finally, we will end this chapter by discussing Windows privacy settings. Many privacy permissions are allowed by default in Windows, and they could pose a privacy risk to your organization if they're not disabled.

In this chapter, we will cover the following topics:

- Preventing an Adversary-in-the-Middle attack
- Protecting against lateral movement and privilege escalation
- Windows privacy settings

Technical requirements

To follow this chapter's instructions, you will need a Microsoft 365 subscription and devices that are managed by Intune. We have included links to the official repositories for the pentesting tools that we will be using for demonstration purposes as appropriate. You will need the following licensing or an equivalent SKU to deploy policies with Intune:

- Microsoft 365 E3 and/or E5
- Windows 10/11 Pro or Enterprise

Let's start by learning about Adversary-in-the-Middle attacks.

Preventing an Adversary-in-the-Middle attack

For clients connected to corporate networks, intra-network communications are critical to the basic functionality of systems. This could be connectivity to the local intranet, printers, file servers, or accessing the internet. For attackers that have gained access to the internal network, several tools and techniques can be used to listen for and intercept these communications. If the attacker can place themselves in the middle of the communications path, they can gather information, manipulate, and modify traffic, and force users to unknowingly authenticate to them. If they're successful in their efforts, passwords can be captured, cracked, and forwarded to other systems in relay attacks to authenticate them against other systems. This technique is known as an **Adversary-in-the-Middle (AiTM)** or **Man-in-the-Middle (MiTM)** attack.

In the next few sections, we are going to review different network protocols that adversaries can use to intercept, spoof, and poison responses back to the victim to commit an AiTM attack. This includes the following:

- **Link-Local Multicast Name Resolution (LLMNR)**
- **NetBIOS Name Service (NBT-NS)**
- **Multicast DNS (mDNS)**
- **Web Proxy Auto-Discovery Protocols (WPAD)**
- **Server Message Block (SMB) relay**
- IPv6 DNS spoofing
- **Address Resolution Protocol (ARP) cache**

When a Windows system attempts to communicate with a destination host using protocols such as TCP/IP, UDP/IP, or ICMP/IP, it must discover the name or IP address of the destination system to establish communication. During the lookup process, Windows will first attempt to use a local host file or send a DNS request in the hopes that it finds an answer. If there is no valid answer, the Windows system will send additional broadcasts over the local network to garner a response and find the destination. These are known as **LLMNR** and **NBT-NS** requests. Additionally, clients will attempt to resolve hostnames to any available DNS server over the `.local` suffix using an **mDNS** request. These methods can become prone to name poisoning exploits and unknowingly receive responses from untrusted hosts.

If an attacker gains access to your network, they can set up tools and deceive clients by responding to these broadcasts and sending poisoned responses back. Depending on the malicious services that are running, attackers may present a fake login prompt to the user and coerce them into entering credentials. These activities can ultimately result in plaintext or hashed passwords being captured that can then be cracked or used in relay attacks. First, let's look at LLMNR.

LLMNR

When a DNS request fails, clients will fall back to using LLMNR for name resolution as a secondary protocol through an unauthenticated UDP broadcast over the local network. This allows for any system on the network to respond to the request when DNS services are not available or can't provide a valid response. Using a tool such as Responder (<https://github.com/SpiderLabs/Responder>), an attacker can answer these requests if LLMNR is not disabled on all the available network adapters. Let's look at an example of a poisoned response. In the following diagram, the client PC is attempting to connect to a network share and accidentally mistypes the name. Unbeknownst to them, an attacker sees this request and sends a poisoned response back, tricking the victim into thinking they found the desired destination:

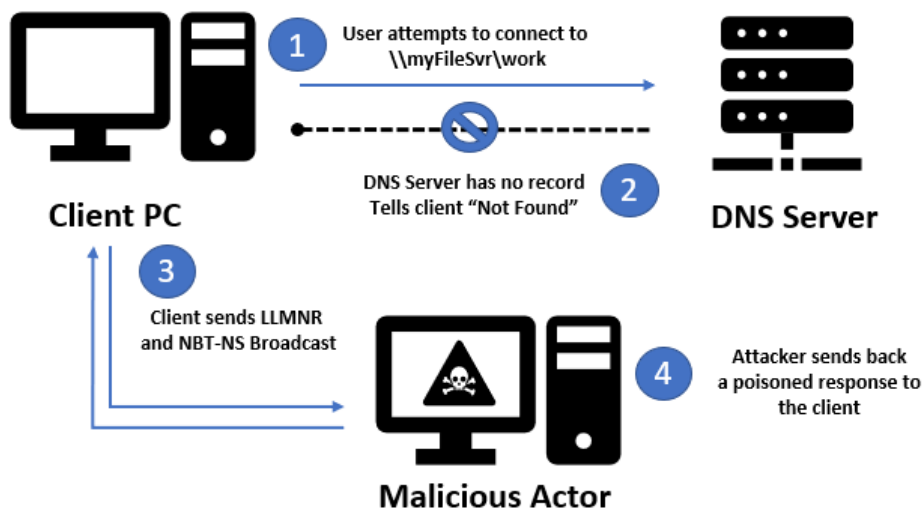


Figure 10.1 – Multicast response communication flow

Depending on the tooling, the user might be presented with an authentication prompt; otherwise, their credentials will be stolen automatically if the PC attempts to use Windows credentials to connect to the share. The following screenshot shows Responder sending a poisoned answer back to the user who was attempting to connect to a file server that doesn't exist on the network. As you can see, it captured the destination host, as well as the client's IP address, username, and password hash. In this example, Responder poisoned answers to the LLMNR, NBT-NS, and mDNS protocols:

```
[*] [NBT-NS] Poisoned answer sent to 192.168.1.111 for name WORKGROUP (service: Domain Master Browser)
[*] [NBT-NS] Poisoned answer sent to 192.168.1.111 for name WORKGROUP (service: Browser Election)
[*] [MDNS] Poisoned answer sent to 192.168.1.111 for name myfileservr.local
[*] [LLMNR] Poisoned answer sent to 192.168.1.111 for name myfileservr
[HTTP] Sending NTLM authentication request to 192.168.1.111
[HTTP] Host : myfileservr
[WebDAV] NTLMv2 Client : 192.168.1.111
[WebDAV] NTLMv2 Username : \eyoung@mtlab.com
[WebDAV] NTLMv2 Hash : eyoung@mtlab.com:::32ef0f5e0a9357ab:85966AD3F138540BBCB2AD5D9A73995D:010100000
AD80125D605DD97EC42660000000020080056004B004E00470001001E00570049004E002D00440059004F003800590043003100
00140056004B004E0047002E004C004F00430041004C00030034000570049004E002D00440059004F0038005900430031004800540
B004E0047002E004C004F00430041004C0005001A0056004B004E0047002E004C004F00430041004C000800300030000000000000
BC291C8F116A5200FA6D84E34662279A996CAA631D4410DD902E9BA1B648664F0A0010000000000000000000000000000
050002F0060D007900606069006C00650007200760065007200760065007200000000000000000000000000000000
```

Figure 10.2 – Captured password hash

In a well-administered network, the LLMNR protocol can likely be disabled and can be configured using Intune, Group Policy, and Configuration Manager. To disable LLMNR using Registry, use the following setting:

- HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\DNSClient
- EnableMulticast and use a REG_DWORD value of 0

To disable LLNMR with **Intune Settings Catalog**, from **Settings picker**, search for **Multicast** and select the **Administrative Templates\Network\DNS Client** category. Choose **Turn off multicast name resolution** and change the setting to **Enabled**.

Once the policy has been applied, the client should no longer broadcast messages using LLMNRR.

Tip

We strongly encourage you to understand how to use the LLMNR protocol and how it can adversely affect workstation communications before disabling it. Some organizations may need to consider additional network access controls and rely on strong password requirements as mitigation if this cannot be turned off.

Next, let's look at NBT-NS and how to disable NetBIOS over TCP/IP for network interfaces.

NBT-NS

NBT-NS is another broadcast protocol over TCP/IP that is used by the Windows system to find resources as a secondary to DNS. It is enabled by default for all the interfaces in Windows and is vulnerable to the attack techniques that are used by the LLMNR protocol. If name resolution cannot be satisfied using the Windows system's local hosts file or through a DNS request, then it will resort to using NBT-NS to locate the resource. Any system that's available on the network can respond to these broadcast messages and send a response.

Tip

As we mentioned previously regarding LLMNR, disabling NetBIOS can break communication with your systems if broadcast messages are being used to resolve NetBIOS queries. Please consider this before disabling it.

The NetBIOS settings of a network adapter can be viewed by opening the **Settings** app and clicking on **Network & Internet** | **Advanced network settings** | **More network adapter options**. Right-click on a network adapter and choose **Properties**. Then, select **Internet Protocol Version 4 (TCP/IPv4)** and choose **Properties**. Click on **Advanced** and choose the **WINS** tab. As shown in the following screenshot, the NetBIOS settings have been set to their defaults, which enables NetBIOS over TCP/IP:

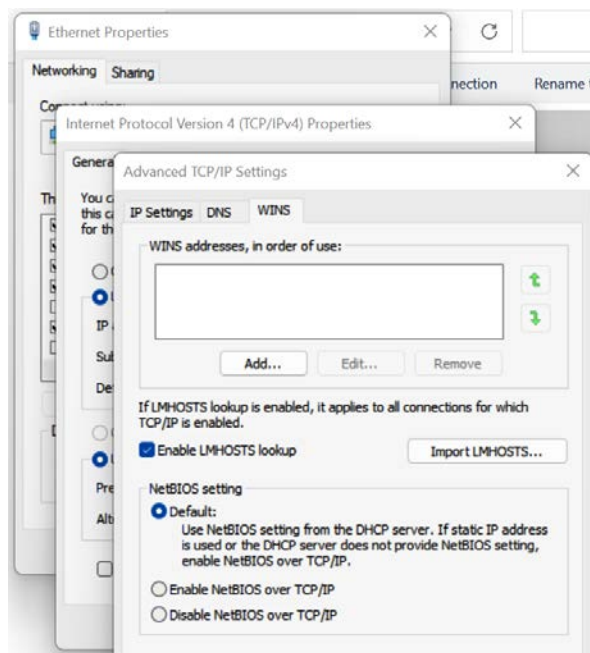


Figure 10.3 – NetBIOS is enabled by default on all adapters

To eliminate the risk of NetBIOS resolution poisoning, it must be disabled on all network adapters. This cannot be accomplished using a policy configuration alone. Let's look at how we can use PowerShell to detect and disable NetBIOS over TCP/IP for all interfaces. There are a few ways to deploy PowerShell scripts to managed devices, including Win32 app deployments or remediation scripts in Intune, Configuration Baseline with a discovery and remediation script in Configuration Manager, or even Group Policy. Let's look at using the discovery and remediation method in Configuration Manager:

1. Open the **Configuration Manager** console. Go to **Assets and Compliance | Compliance Settings | Configuration Items** and click on **Create Configuration Item**.
2. Give it a friendly name, such as `Disable NetBIOS Interfaces`, and click **Next**.
3. For **Support Platforms**, click **Next** to keep the default settings. Click **New** under **Specify settings for this operating system** and enter the following settings:

Create Setting

General | Compliance Rules

Specify details about this setting that represents a business or technical condition to assess for compliance on client devices.

Name:

Description:

Setting type:

Data type:

Discovery script

Specify the script to find and return the value to be assessed for compliance on client devices. Use the echo command to return the script value to Configuration Manager.

Script status: No script specified.

Remediation script (optional)

Specify the script to remediate noncompliant setting values found on client devices. Configuration Manager passes the noncompliant value to the script as a parameter.

Script status: No script specified.

☐ Run scripts by using the logged on user credentials

☐ Run scripts by using the 32-bit scripting host on 64-bit devices

Figure 10.4 – Create Setting

4. Under **Discovery script**, select **Add Script...**
5. Keep Windows PowerShell selected. Now, we can build the discovery script.

The first section of the code will define the \$NBTNS variable. We will use the Get-ItemProperty cmdlet to look in the respective registry hive and enumerate all the TCPIP_GUID keys that start with tcpip using a * wildcard. We want to look specifically at the REG_DWORD value of NetbiosOptions:

```
$NBTNS = Get-ItemProperty HKLM:\SYSTEM\CurrentControlSet\
Services\NetBT\Parameters\Interfaces\tcpip* -Name
NetbiosOptions
```

The following values for REG_DWORD determine the effective state:

- A NetbiosOptions value of 0 is the default
- A NetbiosOptions value of 1 equates to enabled
- A NetbiosOptions value of 2 equates to disabled

In the following screenshot, you can see the amount of unique TCPIP_{GUID} values that must be potentially looked through as each is a unique network interface adapter in Windows:

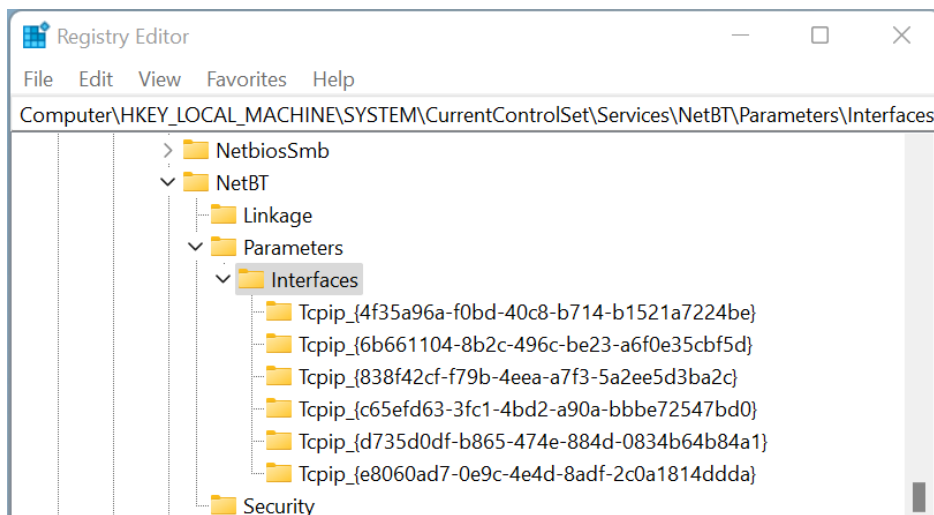


Figure 10.5 – The TCPIP_GUID values of network adapters

Next, we will create an If statement that states if the value of `NetbiosOptions` does not equal 2, set the `$NBTNSCompliance` variable to No. If it does equal 2, set it to Yes:

```
If (!$NBTNS.NetbiosOptions -eq "2")){ $NBTNSCompliance = "No"
} Else { $NBTNSCompliance = "Yes" }
```

Now, let's print the output of `$NBTNSCompliance` so that Configuration Manager can use it to evaluate for compliance. The discovery script should look as follows:

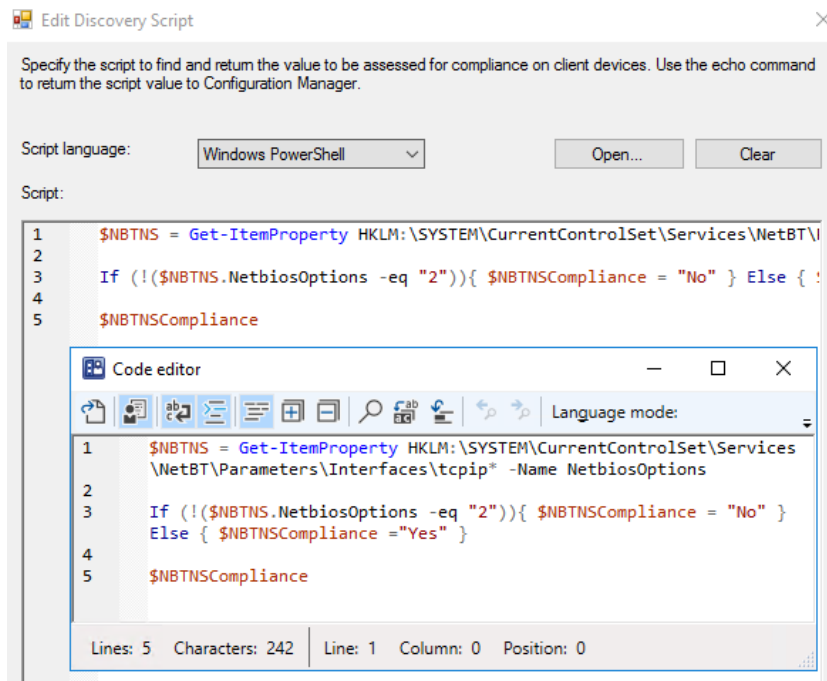


Figure 10.6 – Edit Discovery Script

Click **OK**. Then, click **Add Script** under **Remediation Script**. Keep **Windows PowerShell** set as the script language.

To remediate the settings for each unique network adapter, we will use the `Set-ItemProperty` cmdlet to change all the `NetbiosOptions` `REG_DWORD` values to 2 as it iterates through each entry:

```
Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Services\NetBT\
Parameters\Interfaces\tcpip* -Name NetbiosOptions -Value 2
```

The following screenshot shows the **Edit Remediation Script** dialog box with the script code that was used for this task:

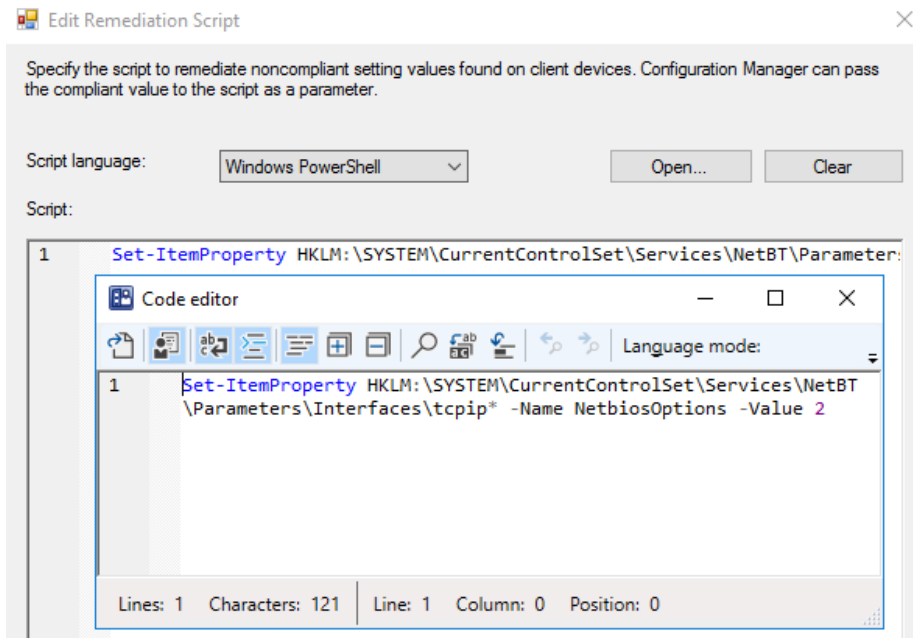


Figure 10.7 – Edit Remediation Script

Now, let's create a compliance rule to validate these settings:

1. Click **OK**, then click on the **Compliance Rules** tab in the **Create Setting** dialog box. Click **New** to create a new compliance rule.
2. Give it a friendly name, such as **Disable NetBIOS**. Regarding **The setting must comply with the following rule**, leave **Equals** selected and enter **Yes** in the box.
3. Select both **Run the specified remediation script when this setting is noncompliant** and **Report noncompliance if this setting instance is not found**.
4. Choose **Information** for **Noncompliance severity for reports**. Click **OK**, click **OK** again, and choose **Next** back on the **Create Configuration Item Wizard** screen.
5. Click **Next** a few times to complete the wizard and build the configuration item.

For this configuration item to be used to evaluate for compliance and remediation, you must create or assign it to a Configuration Baseline. We covered Configuration Baselines in *Chapter 6, Administration and Policy Management*.

Once the Configuration Baseline has been evaluated on the client, we can confirm that it's been enforced by checking the network interface settings, as we saw earlier. The NetBIOS setting should be set to **Disable NetBIOS over TCP/IP**. Upon rebooting, no more requests will be allowed to be made using NetBIOS over TCP/IP.

Next, let's learn how to use mDNS discovery to resolve hosts on a local link.

mDNS

Multicast DNS (mDNS) is used in zero-configuration networking as a way to locate devices, hosts, and services on local networks instead of a DNS server. The engineering specification for mDNS uses many of the same standards as regular DNS. Zero-configuration networking allows for devices connected on the same local network to locate each other without having to configure services such as DNS. mDNS was originally developed by Apple and called Bonjour for use with printer discovery. Instead of sending a unicast request to a DNS server, mDNS sends out a multicast message over the local network, looking for any available hosts to respond to the request and determine the best match for the location of the resource. This multicast broadcast is not private and is prone to the same type of response poisoning as LLMNR and NetBIOS to someone inside your network. The flow for mDNS can be seen in the following diagram, where the client PC sends out a multicast request, looking for a peripheral such as a printer or wireless display, and the malicious actor intercepts the request and poisons the response back to the victim:

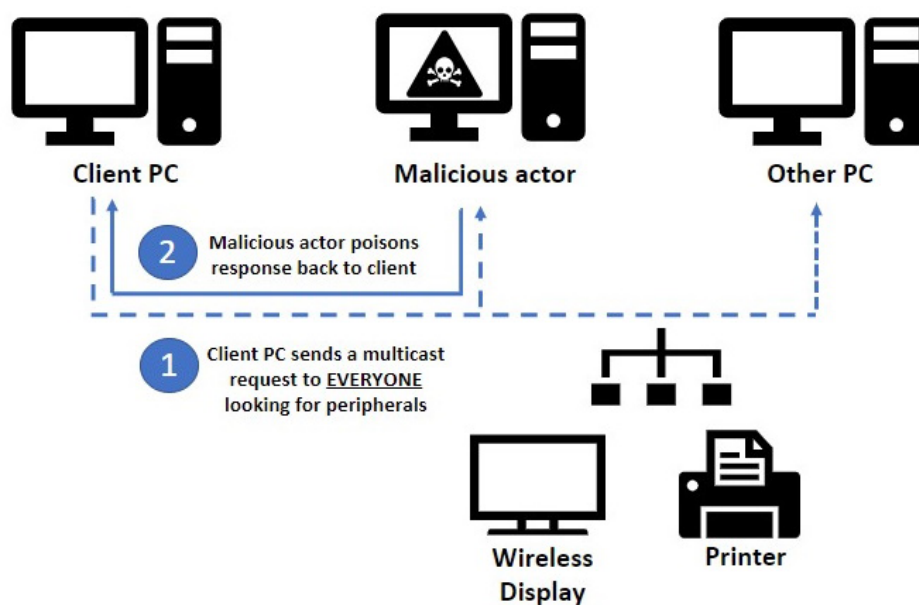


Figure 10.8 – Multicast DNS

An example of this type of request can be seen if a user types a word in their web browser instead of a formulated URL. In the following screenshot, the user typed the word `cookies` into the Microsoft Edge Browser. Using a tool such as **Responder**, an attacker can intercept the mDNS broadcast and send a poisoned response back to request NTLM authentication. In this case, it resulted in a password hash being captured, which remained transparent to the user:

```
[*] [MDNS] Poisoned answer sent to 192.168.1.114 for name qtnmawycyljdzfh.local
[*] [MDNS] Poisoned answer sent to 192.168.1.114 for name jmldyzwhyck.local
[*] [MDNS] Poisoned answer sent to 192.168.1.114 for name qrvsrrtietcqvavv.local
[HTTP] Sending NTLM authentication request to 192.168.1.114
[HTTP] Sending NTLM authentication request to 192.168.1.114
[HTTP] Sending NTLM authentication request to 192.168.1.114
[*] [MDNS] Poisoned answer sent to 192.168.1.114 for name cookies.local
[HTTP] Sending NTLM authentication request to 192.168.1.114
[HTTP] Host : cookies
[HTTP] NTLMv2 Client : 192.168.1.114
[HTTP] NTLMv2 Username : mtlab\eyoung
[HTTP] NTLMv2 Hash : eyoung::mtlab:ffdc4e965d16cb26:FDA5A6CE68713DB0B91DC8125F4
D801097537323DBB13EF000000000200080033003400340052001001E00570049004E002D0054004F0
3000400140033003400340052002E004C004F00430041004C0003003400570049004E002D0054004F00
002E0033003400340052002E004C004F00430041004C000500140033003400340052002E004C004F004
001000000002000009E24C955AF4B7714E7D7D440F529D38AA171AF5B8B606A07AB668C96CFC0AE610A
00000900180048005400540050002F0063006F006F006B006900650073000000000000000000
```

Figure 10.9 – Poisoned mDNS response

Not only is mDNS enabled by default in Windows, but web browsers such as Microsoft Edge and Google Chrome use Google Cast to send out multicast requests to stream content to wireless devices. The Google Cast functionality is shown in the following screenshot. It can be accessed from the Chrome browser by clicking on **Cast**:

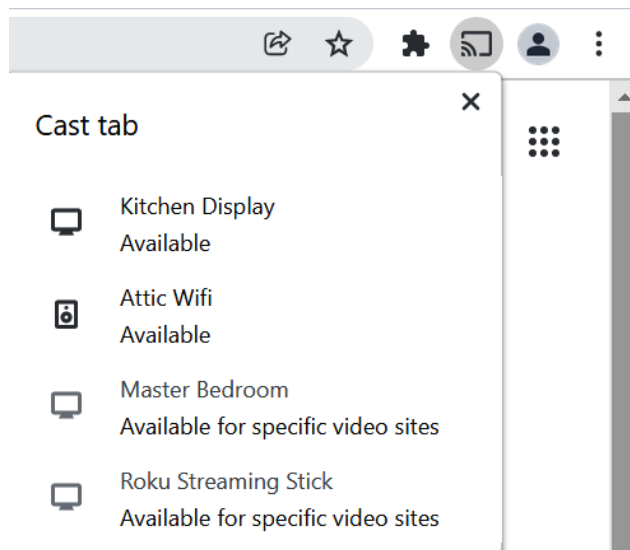


Figure 10.10 – Google Cast

Google Cast is generally unnecessary in a corporate environment, so we recommend disabling this functionality in your security baselines. In addition, you should block the Google Chrome mDNS inbound firewall rule that is created when the software is being installed, as shown in the following screenshot:

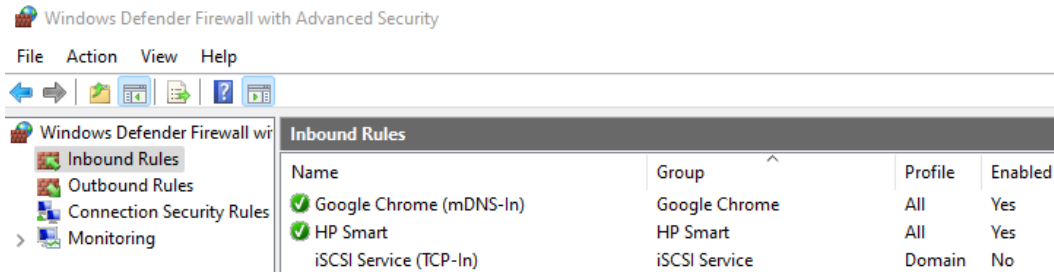


Figure 10.11 – Google Chrome mDNS firewall rule

If the rule isn't blocked, `chrome.exe` listens on port UDP 5353 inbound for all network profiles (domain, private, and public). We can demonstrate this by using `netstat` and process monitor (`ProcMon`). The Google Cast functionality will show that `chrome.exe` is listening over UDP port 5353, as shown here:

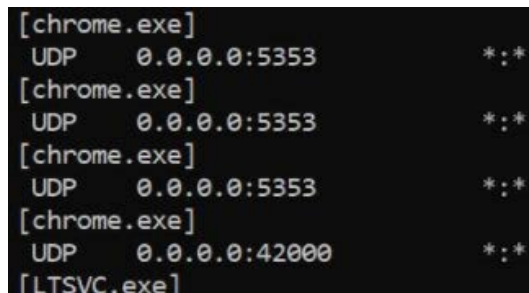
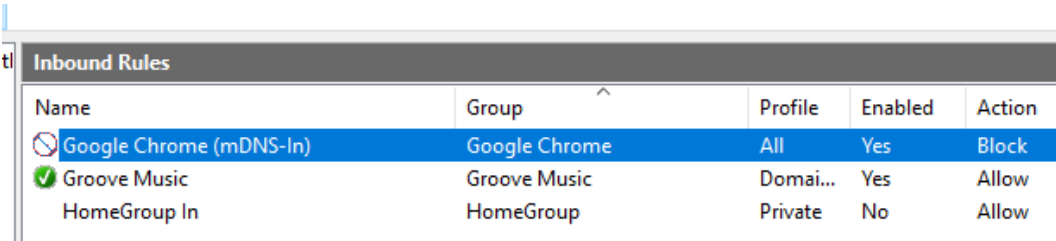


Figure 10.12 –Chrome.exe listening on UDP 5353

Note that just disabling this firewall rule once is not a permanent solution as each time Google Chrome updates, the rule will be re-enabled. To mitigate this, we recommend using a discovery and remediation compliance script through Configuration Manager or by using a Win32 app or remediation script in Intune.

Once this setting has been applied, the firewall rule will be blocked, as shown in the following screenshot:





Inbound Rules					
Name	Group	Profile	Enabled	Action	
 Google Chrome (mDNS-In)	Google Chrome	All	Yes	Block	
 Groove Music	Groove Music	Domai...	Yes	Allow	
HomeGroup In	HomeGroup	Private	No	Allow	

Figure 10.13 – Inbound firewall rules in Windows Defender Firewall

To disable mDNS on Windows-based systems, we can create a specific registry key, as follows:

- HKLM:\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters
- **REG_DWORD:** EnableMDNS
- **Value:** 0

Once the value has been created, mDNS requests can no longer be poisoned. The following screenshot shows the aforementioned registry key:

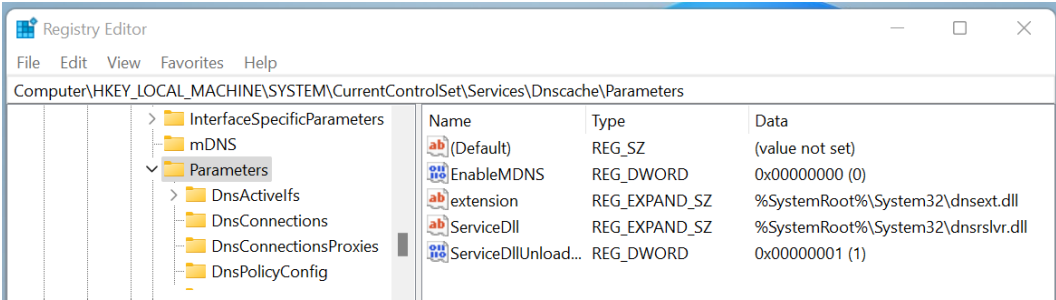


Figure 10.14 – Disabling mDNS for Windows

At the time of writing, there is no group policy setting or CSP that controls this setting, so we recommend using a Configuration Manager baseline or deploying a PowerShell script to control this policy. To enable the registry setting, use the following PowerShell command:

```
Get-Item -path "HKLM:\SYSTEM\CurrentControlSet\Services\
Dnscache\Parameters" | New-ItemProperty -Name EnableMDNS -Value
0 -Force
```

Next, let's look at WPAD and how to prevent attacks by disabling the service.

The WPAD protocol

The WPAD protocol or **autoproxy** is a discovery protocol that's used by Windows to find web proxy settings located in a configuration file called `wpad.dat`. In an enterprise network, a network administrator typically configures DHCP and DNS to allow proxy settings to be fetched from a managed proxy server for the company's DNS suffix; the `wpad.dat` file is served to client PCs in a managed fashion. The security risk is that when a computer is configured to automatically look for proxy configurations on the network, no internal DNS or DHCP is configured. This allows anybody available to respond to the request. If an attacker is positioned inside the network, they can impersonate themselves as a proxy server, receive the connection requests, and offer the computer a malicious `wpad.dat` file. This can lead to an attacker redirecting all web traffic through their host for inspection and manipulation. Attackers can then modify websites to steal information, download malware-infected files, and force an HTTP-NTLM authentication request to entice users to supply credentials. The following diagram shows how this works:

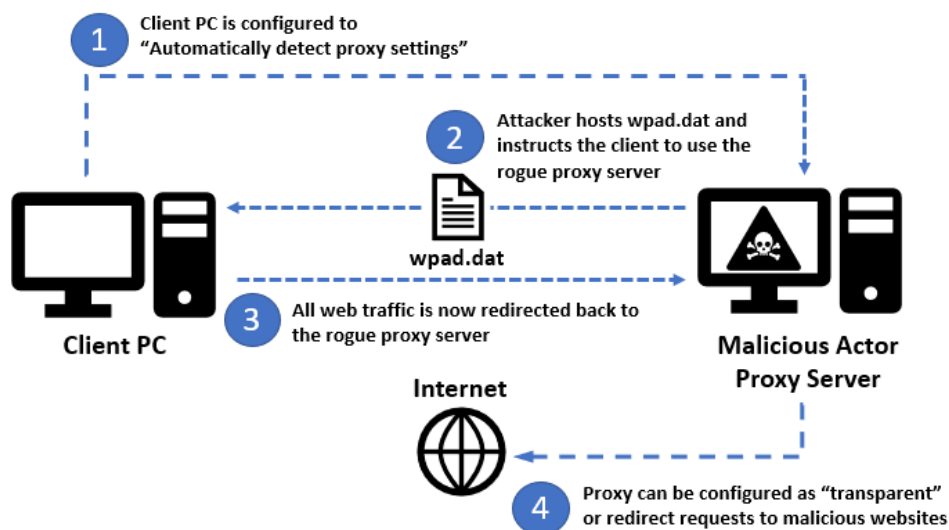


Figure 10.15 – Malicious proxy server

To view the Windows proxy settings, open **Settings** and go to **Network & internet** | **Proxy**. The following screenshot shows that the default Windows settings for **Automatic proxy setup** | **Automatically detect settings** have been set to **On**:



Figure 10.16 – Windows Proxy settings

Once again, using the **Responder** tool, you can see that the target computer reached out for `wpad.dat` and received a poisoned response requesting NTLM authentication:

```
[*] [LLMNR] Poisoned answer sent to 192.168.1.112 for name wpad
[*] [LLMNR] Poisoned answer sent to 192.168.1.112 for name wpad
[*] [MDNS] Poisoned answer sent to 192.168.1.112 for name wpad.local
[*] [MDNS] Poisoned answer sent to 192.168.1.112 for name wpad.local
[*] [LLMNR] Poisoned answer sent to 192.168.1.112 for name wpad
[*] [LLMNR] Poisoned answer sent to 192.168.1.112 for name wpad
[HTTP] User-Agent      : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
.00.22976 Chrome/85.0.4183.121 Electron/10.4.3 Safari/537.36
[HTTP] User-Agent      : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
.00.22976 Chrome/85.0.4183.121 Electron/10.4.3 Safari/537.36
[HTTP] User-Agent      : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
.00.22976 Chrome/85.0.4183.121 Electron/10.4.3 Safari/537.36
[HTTP] User-Agent      : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
.00.22976 Chrome/85.0.4183.121 Electron/10.4.3 Safari/537.36
[HTTP] User-Agent      : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
.00.22976 Chrome/85.0.4183.121 Electron/10.4.3 Safari/537.36
[HTTP] Sending NTLM authentication request to 192.168.1.112
```

Figure 10.17 – Rogue proxy server

As a result, the attacker was able to capture the password after serving the malicious WPAD file. This hash can be taken offline in an attempt to crack it in plaintext or forward it to other systems in a relay-style attack:

```
[HTTP] WPAD (auth) file sent to 192.168.1.112
[HTTP] NTLMv2 Client : 192.168.1.112
[HTTP] NTLMv2 Username : DESKTOP-152N245\Admin
[HTTP] NTLMv2 Hash : Admin::DESKTOP-152N245:1d60f21143346c0d:E5686711015BC586FE4DB1949F546C51:01010
A7F77CF0BD80151964E86104C5F7500000000200080034004E004200540001001E00570049004E002D004A0036004100510054
5300430033000400140034004E00420054002E004C004F00430041004C0003003400570049004E002D004A00360041005100540
300430033002E0034004E00420054002E004C004F00430041004C000500140034004E00420054002E004C004F00430041004C00
0000000000100000000200000D28D4A10F0DA4DCFEF18D0FC6FD5E70961AF4AD9F0EDAB5C599B091B91FDA1D30A00100000000
000000000000000900120048005400540050002F00770070006100640000000000000000
[HTTP] WPAD (auth) file sent to 192.168.1.112
[*] [NBT-NS] Poisoned answer sent to 192.168.1.112 for name PROXYSRV (service: Workstation/Redirector)
[*] [MDNS] Poisoned answer sent to 192.168.1.112 for name ProxySrv.local
[*] [LLMNR] Poisoned answer sent to 192.168.1.112 for name ProxySrv
[PROXY] Received connection from 192.168.1.112
[PROXY] Received connection from 192.168.1.112
[PROXY] Received connection from 192.168.1.112
```

Figure 10.18 – Captured password hash

There are a few ways to protect against WPAD attacks and rogue proxy servers on a network, as follows:

- Disable WPAD completely or configure the proxy settings on client computers (recommended)
- On internal company networks, configure a WPAD server in the DNS lookup chain to avoid computers connecting to a rogue proxy server.
- Configure DHCP to point WPAD to a known host to control the traffic.

To disable WPAD on the client, you can use an Intune **Device Restrictions** profile | **Network proxy** template, Group Policy, or the registry. We also recommend that you disable the following service:

- **WinHTTP Web Proxy Auto-Discovery Service:** WinHttpAutoProxySvc

To disable the service through the registry, set the following key:

- HKLM\SYSTEM\CurrentControlSet\Services\WinHttpAutoProxySvc
- **Start** REG_DWORD must be set to 4 (this sets startup mode to disabled)

After the service has been disabled and the PC restarts, the service will no longer be running. Additionally, if a user tries to re-enable WPAD, the setting will not be enforced unless the user physically starts the service using administrative rights. The following screenshot shows the LAN settings in **Internet Properties** once the service has been disabled:

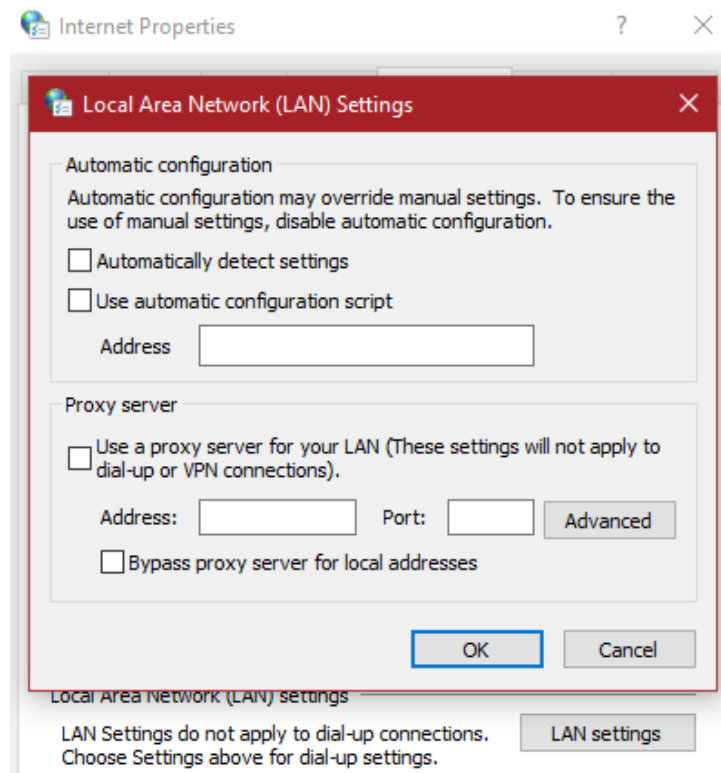


Figure 10.19 – LAN settings

With that, we have looked at several examples of how an attacker can become an AiTM by exploiting LLMNR, NBT-NS, mDNS, and WPAD. Ultimately, once a password has been captured, it may only be a matter of time before it's cracked or leveraged in a relay attack to connect to other systems. Next, let's look at an NTLM relay attack and how to mitigate its risks.

Enable Kerberos authentication

In a network environment, you can use Kerberos authentication and disable NTLM completely. Kerberos is more secure than NTLM and offers better performance. To ensure that there will be no disruption if you're considering disabling NTLM, auditing policies can be enabled in Group Policy and events can be reviewed in Event Viewer or from a SIEM if you're streaming logs to a third-party service. For deployments that leverage the Azure cloud, system events can be sent to Log Analytics and queried using the KQL query language. The policies that you can use to enable NTLM auditing can be found by going to **Computer Configuration | Windows Settings | Security Settings | Local Policies | Security Options**, as follows:

- **Network Security: Restrict NTLM: Audit NTLM authentication in this domain** is set to `Enable all`
- **Network Security: Restrict NTLM: Audit Incoming NTLM Traffic** is set to `Enable auditing for domain accounts`

Events can be searched for in **Microsoft-Windows-Security-Auditing** with **Event ID 4624**. Under **Detailed Authentication Information**, you can view the protocol that was used under **Package Name** (LM, NTLMv1, or NTLMv2). In the following screenshot, we ran the following query in Azure Log Analytics to return all NTLM authentication requests:

```
SecurityEvent
| where EventID == "4624" and AuthenticationPackageName == "NTLM"
```

The output of this query will look as follows:

Activity	4624 - An account was successfully logged on.
AuthenticationPackageName	NTLM
ImpersonationLevel	%%1833
IpAddress	
IpPort	59623
KeyLength	128
LmPackageName	NTLM V1
LogonGuid	00000000-0000-0000-0000-000000000000
LogonProcessName	NtLmSsp
LogonType	3
LogonTypeName	3 - Network

Figure 10.21 – NTLM audit event

After carefully auditing the authentication requests and you are ready to disable NTLM, you can either disable it completely or use a server exception list by configuring the following policies:

- **Network Security: Restrict NTLM: Add server exceptions for NTLM authentication in this domain** and add any servers for which an exception must be listed.
- **Network Security: Restrict NTLM: NTLM authentication in this domain** must be set to `Deny for domain servers` or `Deny all`. Setting the policy to `Deny for domain servers` will allow those listed in the exemption list.

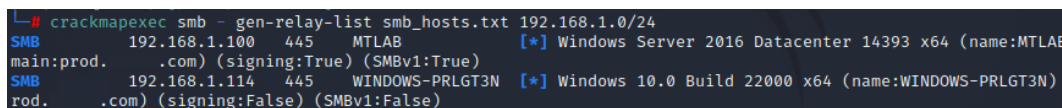
Tip

Adding users to the **Protected User** domain group will require them to authenticate using Kerberos only. This is a good way to confirm an account will work using Kerberos authentication before it's enforced.

Next, let's review enabling SMB signing.

Require SMB signing

SMB is a common protocol that's used for transferring files and facilitating **Microsoft Remote Procedure Calls (MSRPC)** in Windows. It is most notably used for its file-sharing capabilities on workstations and servers. By requiring SMB signing on both clients and servers, endpoints can verify the origin of the sender and the authenticity of the request. Devices that do not require SMB signing are suitable candidates for NTLM relay attacks. If an attacker intercepts the SMB packet flow with SMB signing enabled, the intended recipient will be able to detect that tampering occurred. Using a tool called **CrackMapExec**, an attacker can scan a subnet to find computers that do not require SMB signing, as shown in the following screenshot:



```
crackmapexec smb - gen-relay-list smb_hosts.txt 192.168.1.0/24
SMB 192.168.1.100 445 MTLAB [*] Windows Server 2016 Datacenter 14393 x64 (name:MTLAB
main:prod. .com) (signing:True) (SMBv1:True)
SMB 192.168.1.114 445 WINDOWS-PRIGT3N [*] Windows 10.0 Build 22000 x64 (name:WINDOWS-PRIGT3N)
rod. .com) (signing:False) (SMBv1:False)
```

Figure 10.22 – SMB signing scan

Here's the link to CrackMapExec: <https://github.com/byt3bl33d3r/CrackMapExec>.

Once the systems have been identified, a combined list of targets can be exported and used in a coordinated multi-relay attack. To complete this, an attacker can use any combination of methods to capture user password hashes (LLMNR, NBT-NS, WPAD, and mDNS), and pass them along to a tool such as **NTLMRelayX** to try and authenticate to the target system with SMB. More information about NTLMRelayX can be found at <https://github.com/SecureAuthCorp/impacket/blob/master/examples/ntlmrelayx.py>.

To learn more about the basics of SMB signing, we strongly recommend reading this article on Microsoft docs: <https://docs.microsoft.com/en-us/archive/blogs/josebda/the-basics-of-smb-signing-covering-both-smb1-and-smb2>.

Enabling SMB signing can reduce network performance and should be reviewed carefully before it's enabled. Over the years, the policy guidance for configuring SMB signing has evolved alongside the evolution of the protocols (SMB1, SMB2, and SMB3). In Group Policy, these settings can be found by going to **Computer Configuration | Policies | Windows Settings | Security Settings | Security Options** and are as follows:

- **Microsoft network client: Digitally sign communications (always)**
- **Microsoft network client: Digitally sign communications (if server agrees)**
- **Microsoft network server: Digitally sign communications (always)**
- **Microsoft network server: Digitally sign communications (if client agrees)**

Typically, **always** means to require signing, while **if client/server agrees** means that signing is enabled, but it's up to the destination to decide the requirement. For SMB2 and above, the **if client/server agrees** setting is no longer applicable and SMB packet signing will never be negotiated. If you want to always require SMB signing, use the **Digitally sign communications (always)** policy setting for both clients and servers. In the latest **MSFT Windows 11 – Computer** security baseline, all four policies are set to enabled and SMB signing is a pre-configured policy recommendation in **Intune Security Baselines for Windows 10 and later**. It can also be configured through the Intune **Settings Catalog** under the **Local Policies Security Options** category or by using the Group Policy requiring path mentioned previously.

Next, let's look at protecting LDAP from relay attacks by enabling LDAP signing and LDAP channel binding.

Enable LDAP signing and LDAP channel binding

LDAP is an application protocol that can be used to query directory services such as Active Directory. This includes looking up data such as usernames, passwords, and computer objects. LDAP can also be used in authentication and to perform administrative actions such as resetting passwords and modifying security groups, which makes it a powerful and useful tool. LDAP is susceptible to MiTM and relay attacks if security measures aren't in place. To protect it, cryptographic protocols can be layered onto LDAP to establish secure connections and encrypt data in transit using SSL and TLS. This is known as **LDAPS** or **Secure LDAP**. LDAPS is essentially the same protocol as LDAP, but applications will require a certificate to create a trust chain, and communications in Active Directory use port 636 over 389. It is strongly recommended to use LDAPS over LDAP whenever possible. Additional information about configuring it for Windows Server environments can be found at <https://techcommunity.microsoft.com/t5/sql-server-blog/step-by-step-guide-to-setup-ldaps-on-windows-server/ba-p/385362>.

In addition to using LDAPS, it is recommended that you enable both LDAP signing and LDAP channel binding to increase security against MiTM attacks. Enabling LDAP signing will require integrity verification through a digital signing signature, much like SMB signing, as described previously. LDAP channel binding will combine the transport and application layers through a *binding* to create a unique fingerprint for the LDAP communication to protect it against tampering. During a MiTM attack, if communication was intercepted, the fingerprint would change, resulting in the connection being dropped. For official guidance from Microsoft on enabling LDAP channel binding and LDAP signing, go to <https://msrc.microsoft.com/update-guide/en-us/vulnerability/ADV190023>.

The preceding link will help you enable these policies and refer to event logging that can be configured to minimize issues.

To enable signing, you can configure and set the following group policies, which are located at **Computer Configuration | Policies | Windows Settings | Security Settings | Local Policies | Security Options**:

1. Configure clients and member servers to negotiate signing by setting **Network Security: LDAP client signing requirements** to `Negotiate signing`. Ensure your clients received the updated policy before continuing.
2. Once the clients/servers have received the first policy, on your domain controllers, set the **Domain controller: LDAP server signing requirements** policy to `Require signing`.
3. Next, to enforce signing, change **Network Security: LDAP client signing requirements** to `Require signing` for clients and servers.

To configure channel binding on your domain controllers with Group Policy, go to **Computer Configuration | Policies | Windows Settings | Security Settings | Local Policies | Security Options** and set **Domain controller: LDAP server channel binding token requirements** to `Always`.

Next, let's look at how attackers can abuse IPv6 for MitM attacks and how we can help mitigate this risk.

Preventing IPv6 DNS spoofing

As the internet runs short of available IPv4 addresses, the solution is to use IPv6. Over the past few years, IPv6 usage has steadily increased and some countries have surpassed a 50% adoption rate. The following screenshot of 6lab shows the global IPv6 adoption by country:



Global IPv6 Adoption	
Belgium	65%
Switzerland	58%
Germany	65%
Luxembourg	58%
USA	56%
Internet core	78.92%
Global content	63.62%
Users	47.06%

Figure 10.23 – IPv6 worldwide adoption – 6lab

You can find the preceding chart and other great statistics about IPv6 usage at <https://6lab.cisco.com/index.php>.

Even with the increase in IPv6 usage globally across the internet, it's still unlikely that your internal company or home network is using an IPv6 addressing scheme. While there are many advantages to using IPv6 over IPv4, most smaller networks with internal IP ranges still rely on **Network Address Translation (NAT)** with IPv4. Today, possibly as a way of *future-proofing*, network adapters on clients with both IPv6 and IPv4 enabled will use IPv6 as the preferred prefix policy. This can be a concern for networks that do not have DHCPv6 configured or cannot respond to **Stateless Address Auto-Configuration (SLAAC)** requests, making them vulnerable to rogue DHCPv6 and DNS spoofing attacks. If networks aren't configured with DHCPv6, when a client boots up and sends a request for an IPv6 address, an attacker can answer the request and assign an IP address to the client. Once the client has been made aware of the attacker's endpoint, communications can be re-routed to an attacker-controlled DNS, where network packets can be captured and analyzed so that they can steal information or redirect victims to malicious sites. The following diagram shows how a malicious actor can answer an IPv6 multicast request on a network that's not been configured for IPv6:

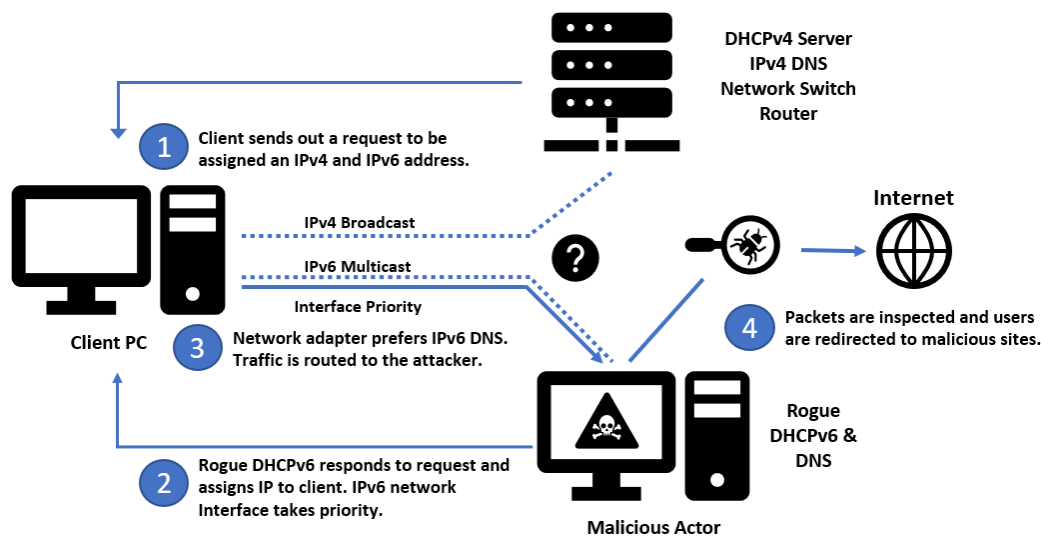


Figure 10.24 – Rogue DHCPv6 and DNS

Using a tool such as **mitm6**, we can see this in action on a network without IPv6 DHCP configured. For more information about the mitm6 tool, visit its public GitHub repository at <https://github.com/dirkjanm/mitm6>.

In the following screenshot, the attacker's endpoint assigned a client PC, WINDOWS-PRLGT3N an IPv6 address and pointed a DNS to its endpoint:

```

# mitm6 -hw WINDOWS-PRLGT3N.prod. .com -d prod. .com --ignore-nofqdn
Starting mitm6 using the following configuration:
Primary adapter: eth0 [00:15:5d:00:02:47]
IPv4 address: 192.168.1.110
IPv6 address: fe80::215:5dff:fe00:247
DNS local search domain: prod. .com
DNS whitelist: prod. .com
Hostname whitelist: windows-prlgt3n.prod. .com
IPv6 address fe80::192:168:1:113 is now assigned to mac=e8:6a:64:25:da:18 host=WINDOWS-PRLGT3N.prod.
168.1.113

```

Figure 10.25 – MiTM attack on IPv6

We can confirm this by running `ipconfig /all` from the command prompt or by opening **Settings | Network & Internet** and choosing the **Ethernet** adapter on the client's PC. In the following screenshot, we can see that the Link-local IPv6 address and IPv6 DNS servers have been assigned by the attacker's rogue DHCP/DNS host:

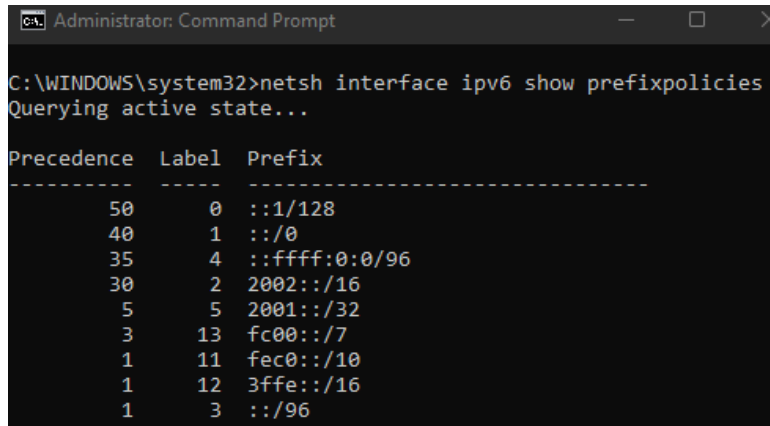
Link speed (Receive/Transmit):	1000/1000 (Mbps)
IPv6 address:	2600:1700:1e1e:7fd0:2d8c:6efa:c052:d9b5
Link-local IPv6 address:	fe80::192:168:1:113%16 fe80::2d8c:6efa:c052:d9b5%16
IPv6 DNS servers:	fe80::215:5dff:fe00:247%16 (Unencrypted)
IPv4 address:	192.168.1.113
IPv4 DNS servers:	192.168.1.254 (Unencrypted)
Primary DNS suffix:	
DNS suffix search list:	
Manufacturer:	Intel
Description:	Intel(R) Ethernet Connection (4) I219-LM
Driver version:	12.18.9.8
Physical address (MAC):	E8-6A-64-25-DA-18

Figure 10.26 – IPv6 address assigned by rogue DHCP

There are a few recommended ways to mitigate this often overlooked and potentially damaging attack vector:

- On internal company networks, configure a DHCPv6 server or disable IPv6 on your network appliances to prevent IPv6 multicast requests from being forwarded.
- On clients, block DHCPv6 traffic and ICMPv6 router advertisements in Windows Firewall.
- On clients, reprioritize IPv4 over IPv6.

On a client PC, you can change the prefix policy precedence of a network interface so that it uses IPv4 over IPv6. Let's look at the default prefix policies by opening an admin command prompt and running `netsh interface ipv6 show prefixpolicies`. As shown in the following screenshot, the first prefix, which has a precedence of 50, is `::1/128`. This is the IPv6 loopback adapter, which is followed by the IPv6 default gateway:



```

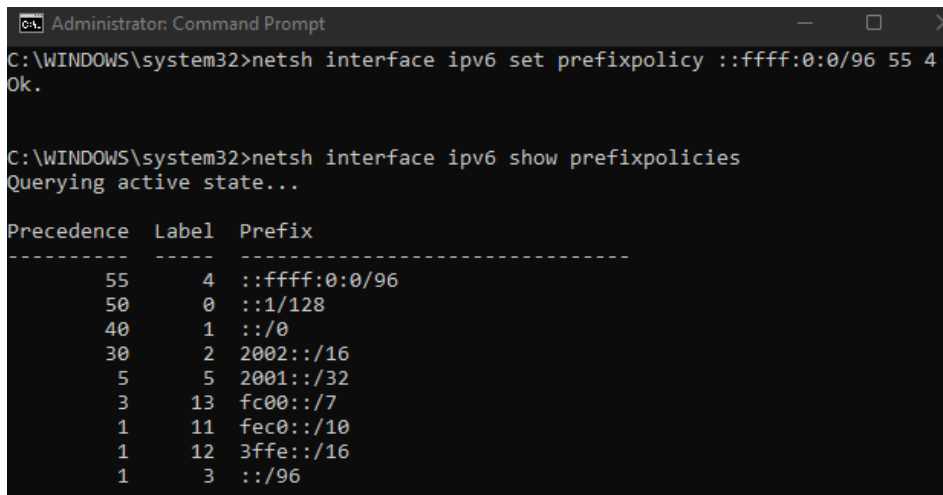
Administrator: Command Prompt

C:\WINDOWS\system32>netsh interface ipv6 show prefixpolicies
Querying active state...

Precedence  Label  Prefix
-----
          50    0  ::1/128
          40    1  ::/0
          35    4  ::ffff:0:0/96
          30    2  2002::/16
           5    5  2001::/32
           3   13  fc00::/7
           1   11  fec0::/10
           1   12  3ffe::/16
           1    3  ::/96
  
```

Figure 10.27 – IPv6 prefix policies

The IPv4-mapped address block in IPv6 CIDR notation is `::ffff:0:0/96` and has a precedence of 35. We can run the `netsh interface ipv6 set prefixpolicy ::ffff:0:0/96 55 4` command to change its priority and verify its output, as shown here:



```

Administrator: Command Prompt

C:\WINDOWS\system32>netsh interface ipv6 set prefixpolicy ::ffff:0:0/96 55 4
Ok.

C:\WINDOWS\system32>netsh interface ipv6 show prefixpolicies
Querying active state...

Precedence  Label  Prefix
-----
          55    4  ::ffff:0:0/96
          50    0  ::1/128
          40    1  ::/0
          30    2  2002::/16
           5    5  2001::/32
           3   13  fc00::/7
           1   11  fec0::/10
           1   12  3ffe::/16
           1    3  ::/96
  
```

Figure 10.28 – IPv4 priority in the prefix policy

You can confirm this by running `ping localhost`. It should resolve to an IPv4 address of `127.0.0.1` as opposed to `::1` for IPv6.

Next, let's look at how an attacker can use ARP to poison an ARP cache and act as a MiTM.

ARP cache poisoning

ARP is a protocol that's used by computers to discover link-layer addresses, such as a **Media Access Control (MAC)** address on a network. An example of this is when a client PC is trying to create an association of an IP to the MAC address of a local router or internet gateway. To create that mapping, the client will send out a broadcast message and wait for a response from a device that contains the associated IP. The device responds with its MAC address and, as a result, the client updates its local ARP cache with the record. Since ARP has no authentication mechanism, anyone on the local network segment can send a spoofed ARP message and reply to these broadcasts, where the client now receives the MAC address of an attacker's endpoint instead of the local router. This can result in a bi-directional traffic flow that passes through the attacker's machine, creating a MiTM in a technique known as ARP cache poisoning. Communications that pass through the attacker's machine are subjected to being captured and inspected or redirected to try and obtain information. The following diagram represents how ARP cache poisoning works:

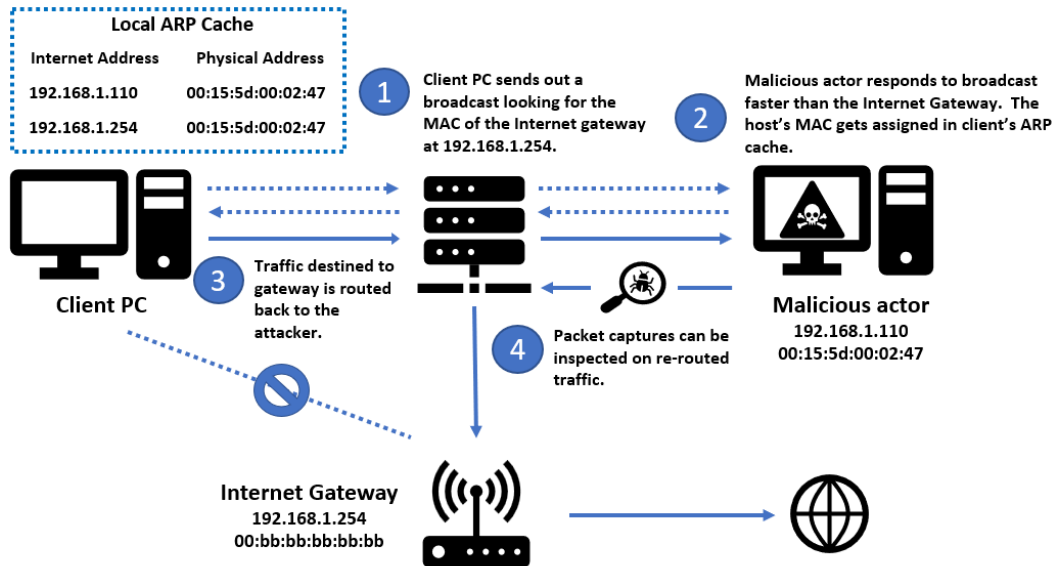


Figure 10.29 – ARP cache poisoning

Using a tool such as **Ettercap**, an attacker can target a victim and gateway to replace the associated MAC address in the ARP cache on the victim's PC by responding faster to broadcast requests. More information about Ettercap can be found at <https://www.ettercap-project.org/>.

Once communications flow through the MiTM endpoint, they are particularly vulnerable if that information is transmitted over a protocol that uses clear text, such as HTTP or FTP. Cleartext values can be extracted from a simple packet capture using a tool such as **Wireshark**. In the following screenshot, we can see that ARP poisoning was successful to the client's PC IP address at 192.168.1.113 and the internet gateway at 192.168.1.254:

```

ARP poisoning victims: 192.168.1.254
GROUP 1 : 192.168.1.113 E8:6A:64:25:DA:18
GROUP 2 : 192.168.1.254 E8:B2:FE:
Starting Unified sniffing...

Text only Interface activated...
Hit 'h' for inline help

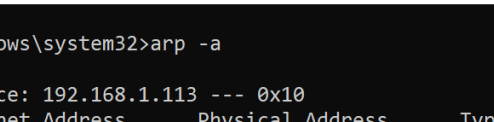
HTTP : 192.168.1.254:80 → USER: PASS: ***** INFO: http://192.168.1.254/cgi-
CONTENT: nonce=fd8436c965cff149cf3fe27d7726cb6998593f2e1ae8b5d1395ebf154d9d7e62&password=*****ghashpassword=61919200
77 06Continue=Continue

HTTP : 192.168.1.254:80 → USER: PASS: ***** INFO: http://192.168.1.254/cgi-
CONTENT: nonce=fd8436c965cff149cf3fe27d7726cb6998593f2e1ae8b5d1395ebf154d9d7e62&password=*****ghashpassword=61919200
77 06Continue=Continue

```

Figure 10.30 – ARP poisoning

Here, the user visited an HTTP site and entered a password. We can check the ARP cache on a client and confirm that ARP poisoning was successful by opening a command prompt and typing `arp -a` to view the output of the ARP cache. We ran this command twice to show what happened before and after. After using the command again, the physical address changed and is now mapped to the attacker's host at `00-15-5d-00-02-47`:



```
C:\windows\system32>arp -a

Interface: 192.168.1.113 --- 0x10
    Internet Address      Physical Address      Type
    192.168.1.254         e8-b2-fe-            dynamic

C:\windows\system32>arp -a

Interface: 192.168.1.113 --- 0x10
    Internet Address      Physical Address      Type
    192.168.1.110         00-15-5d-00-02-47    dynamic
    192.168.1.254         00-15-5d-00-02-47    dynamic
```

Figure 10.31 – Output of the ARP cache

Unfortunately, ARP is a common communication protocol that's used to translate addresses between the data link and the network layer and is insecure. There are a few recommendations to consider that will help mitigate MiTM through ARP cache:

- For company networks, many network switches support a security feature called **Dynamic ARP inspection (DAI)**. This helps evaluate ARP messages over the network and prevents **denial-of-service (DOS)** attacks through ARP.
- On clients, avoid using services that authenticate using insecure protocols. Services such as HTTP, FTP, and Telnet send passwords in plaintext that can be extracted from a packet capture.
- Use protocols with encryption such as SSL/TLS to ensure that communications are encrypted in transit and cannot be decrypted.
- Having well-defined network segmentation can minimize the number of potential targets as ARP messages do not traverse across subnets.
- Define static ARP tables in your network if possible.

As we mentioned previously, there are many ways to use standard communications protocols that can be exploited once an attacker gets access to your network. Many of these protocols are enabled by default, are standard network functionality, and can present significant security risks without taking the proper precautions. Next, let's learn how to protect against lateral movement and privilege escalation if an attacker can gain access to a system or capture credentials.

Protecting against lateral movement and privilege escalation

Assuming your network was breached, an attacker will likely try to gain knowledge about the network and systems by using discovery and reconnaissance techniques. Many of these techniques require little to no privileges to run once they're inside the network. Some examples of discovery tactics that produce useful information include running network scans to identify potential hosts, running vulnerability scans to find hosts that are not patched against known weaknesses, and gathering information by enumerating cloud services and domain policies. Even the slightest bit of information, as trivial as it may seem, could be used against you to find potential security gaps. A few examples of how these reconnaissance activities could be used to launch an attack are as follows:

- Password and Group Policy discovery information can be used to fine-tune methods that are used in brute-force attacks and reduce the chances of account lockout or identity protection services from being triggered.

- File and directory discovery can be used to identify the location of company data that can be stolen, held for ransom, and used for exploitation.
- System service and process discovery can be leveraged to carefully craft attacks, or used for process injection to avoid detection.
- Account discovery can be used to find additional targets and identify privileged accounts.

The **MITRE ATT&CK** framework provides many additional examples of discovery tactics, including explanations of how they can be used to further gain a foothold. To learn more, go to <https://attack.mitre.org/tactics/TA0007/>.

Many of the security controls we've discussed in this book can help prevent lateral movement and privilege escalation, should one of your endpoints become compromised. No one solution fits all that provides guaranteed protection, so you must have a good understanding of the current attack surface to make the best defense decisions for your company to stay within budget and remain tolerant of your acceptable risk. Let's look at a few best practice principles:

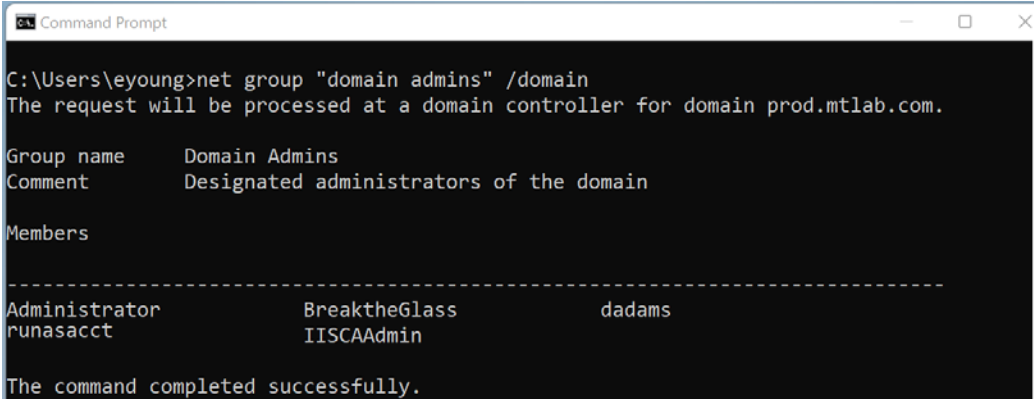
- Protect your identities as well as possible. This includes implementing strong password policies, requiring multi-factor authentication, and going passwordless when possible. For improved protection, enable behavioral and threat monitoring solutions that can audit sign-ins and build a profile of users' activities to alert you when any anomalies occur in an automated way. Use access control policies to restrict access from unapproved devices and require trusted compliant devices through attestation based on zero-trust principles.
- Adhere to the principle of least privilege. Use standard accounts on Windows systems and regularly audit access and remove it unless it can be justified. **Privileged Identity Management (PIM)** and **Privileged Access Management (PAM)** are invaluable in these efforts.
- Strong network designs that include micro-segmentation can help reduce lateral movement by only allowing the necessary systems to talk to each other using only the necessary protocols.
- Enable policies that prevent standard user accounts from enumerating information on the domain or in the cloud. By default, a standard user has a lot of visibility and can be helpful in reconnaissance activities, should the account become compromised. Deploying tools such as Microsoft Defender for Identity can help alert you when anomalies such as reconnaissance occur in Active Directory.

- On clients, enable a solution such as Credential Guard that can block credential-stealing by isolating the **Local Security Authority (LSA)** processes with hypervisor-based isolation. Additionally, **attack surface reduction (ASR)** rules can be enabled to block credential stealing from the LSASS.

Let's look at a few examples of policies that can be enabled to provide some added protection.

Preventing resources from being enumerated

Your infrastructure, resources, and accounts are the jewels of your environment, and they need to remain protected from unauthorized access. As we mentioned previously, if an attacker can get inside your network or compromise an account, one of the first things they will do is gather information. For example, a standard domain user has permission to enumerate all users in your domain simply by running the `net use /domain` command from a computer on your network. This means privileged security groups and accounts can easily be found and targeted. In the following screenshot, the output of the `net group "domain admins" /domain` command returned all the members in the Domain Admins security group:



```
Command Prompt
C:\Users\eyoung>net group "domain admins" /domain
The request will be processed at a domain controller for domain prod.mtlab.com.

Group name      Domain Admins
Comment         Designated administrators of the domain

Members
-----
Administrator   BreaktheGlass      dadams
runasacct        IISCAAdmin
```

Figure 10.32 – Domain Admins enumeration

Resources in your domain can also be discovered by anonymous or *null* sessions. Using tools such as **enum4linux**, if the domain controllers allow null sessions in SMB, it can be leveraged to discover users, computers, groups, and the password policy. The `net use` command is a very basic example but demonstrates the level of visibility that's available to any domain user account. In addition, security tools such as `enum4linux` make it easy to gather additional information with a few commands. More information can be found at <https://github.com/CiscoCXSecurity/enum4linux>.

While it may be necessary for this information to be discoverable, it is possible to block access to certain sensitive groups and users by modifying the **Access Control List (ACL)** on objects directly, or at the OU and domain level. Applying the **deny read** permission to the ACL effectively stops users from querying these objects. These permissions can be applied to a security group where users can quickly be added and removed. In the following screenshot, you can see where the deny read permissions are configured for a security group named **Deny Read Domain Admins**:

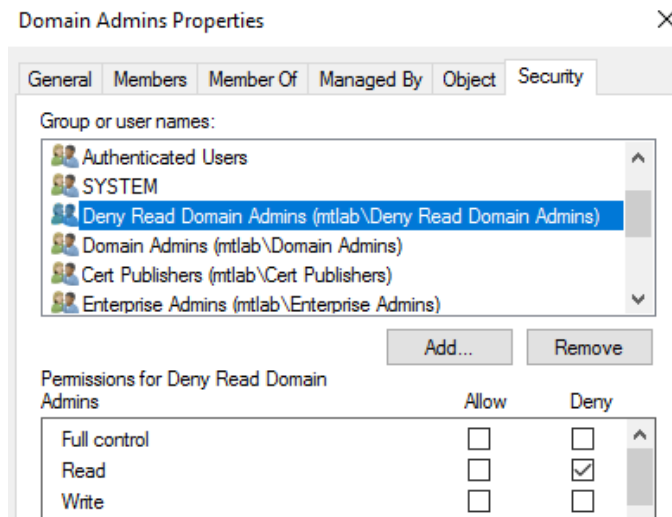


Figure 10.33 – Domain Admins ACL

If a domain user is a member of this group and they run the `net group "domain admins" /domain` command, they will receive an access denied message, as shown in the following screenshot:

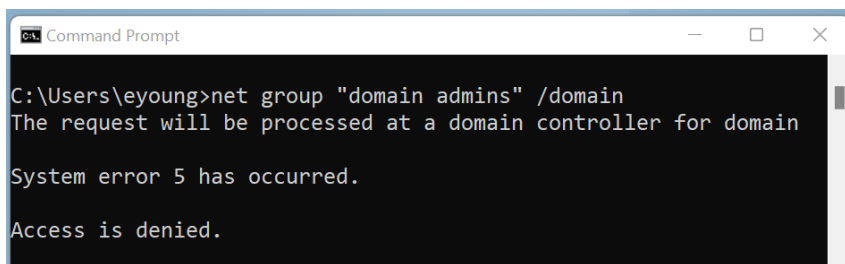


Figure 10.34 – Access denied for enumeration

If all your privileged groups are stored in the same OU, then this permission can be applied to limit what can be discoverable. There is a good blog that goes into more detail if you're interested. It can be found at <https://www.adamcouch.co.uk/disable-domain-user-enumeration/>.

In addition to restricting the information that a domain user can query, we recommend enforcing the policies listed in the CIS Windows Benchmarks in the **Network access** section. A few examples of these policies are as follows:

- **Network access: Do not allow anonymous enumeration of SAM accounts and shares** set to `Enabled`. Enforcing this setting will stop anonymous users from discovering accounts listed in **Security Accounts Manager (SAM)** and network shares. SAM is a database on Windows that stores user accounts and security descriptors for local users. Auditing can be enabled to log attempts to access the SAM database.
- **Network access: Named Pipes that can be accessed anonymously** is set to `none`. This will disable null sessions over named pipes and prevent unauthenticated access.
- **Network access: Allow anonymous SID/Name translation** is `disabled`. This will prevent a user with local access from using the known administrator's SID to discover the built-in administrator account if it has been renamed. If a local administrator account is required on client PCs, it is recommended to create a new account and not reuse the built-in account.
- **Enumerate local users on domain-joined computers** is `disabled`. This may help prevent local user accounts on domain-joined computers from being enumerated.

If you host a directory in the cloud such as **Azure Active Directory (Azure AD)**, it is also susceptible to similar enumeration by standard user account privileges. There are many Azure AD enumeration tools available and once accounts or privileged groups have been discovered, they can be targeted in password spraying attacks. Luckily, Azure has many built-in features to protect you against these attacks, such as security defaults, Conditional Access, and Azure AD Identity Protection. For additional information about security defaults, visit <https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/concept-fundamentals-security-defaults>.

It is possible to prevent account enumeration of the Azure AD tenant for guest and member users by using the following settings:

1. From the Azure portal at <https://portal.azure.com>, open **Azure Active Directory** and select **User Settings**.
2. Restrict access to the Azure AD administration portal by setting it to `Yes`. This will stop non-administrators from using the Azure portal experience. This will not stop them from using PowerShell.

3. Select **Manage external collaboration settings** to view the permission policies for Azure AD B2B guest users.
4. **Guest user access is restricted to properties and memberships of their own directory objects (most restrictive)** must be selected.

By default, directory members have read access to all the users in the directory, which can be disabled using PowerShell. While possible, this is not recommended by Microsoft. To do this, you will need an account with **Global Administrator** permissions and the **MSOnline** PowerShell module. Read access can be disabled by running the `Set-MsolCompanySettings -UsersPermissionToReadOtherUsersEnabled $false` command.

For additional information about the default user access permissions for members and guest users in Azure AD visit <https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/users-default-permissions>.

Tip

Disabling this could cause other services that rely on user account lookups to break.

Next, let's look at Kerberos authentication and how to provide protection against Kerberos-based attacks in the domain. If you are unable to host services in the cloud to leverage modern authentication capabilities such as FIDO, MFA, and Conditional Access, it's important to protect Kerberos on your domain as it's likely the primary authentication mechanism.

Protecting Kerberos tickets

Kerberos is a ticket-based authentication protocol that allows a client and server to connect through mutual-based authentication to verify each other's identity. It is a more secure protocol than NTLM and protects against eavesdropping and relay-type attacks that are a problem with NTLM authentication. Windows domains make use of Kerberos tickets as their default authentication method. When a client authenticates to a domain, the domain controller will respond with a Kerberos service ticket that the client can send to any requested services as proof they are authenticated. As a result, they are authorized to access the resource. During this authentication process, **ticket-granting tickets (TGT)** are issued through the **Key Distribution Center (KDC)** in Active Directory and are encrypted and signed using the special KRBTGT service account.

This account exists by default and can be accessed by an account with elevated privileges on the domain. Attackers who successfully obtain the KRBTGT account password can leverage it to forge TGTs to impersonate any user and, in effect, access any resource in the domain. This process of forging a TGT through the KRBTGT service account is known as a **golden ticket**. If the KRBTGT account cannot be accessed, attackers can also perform a **pass-the-ticket (PTT)** attack using stolen Kerberos tickets and continue to move laterally. PTT attacks steal valid Kerberos tickets through account compromise and with techniques such as credential dumping. Let's look at these attacks in more detail.

Golden ticket and silver ticket attacks

If an attacker can obtain the KRBTGT password hash or *golden ticket*, they can impersonate any account in the domain and abuse the Kerberos authentication process. In a golden ticket attack, the adversary bypasses the issuance of the TGT from the KDC and forges **ticket-granting tickets (TGTs)** with a long validity period that will even remain active if a user changes their account password. This process is shown in the following diagram:

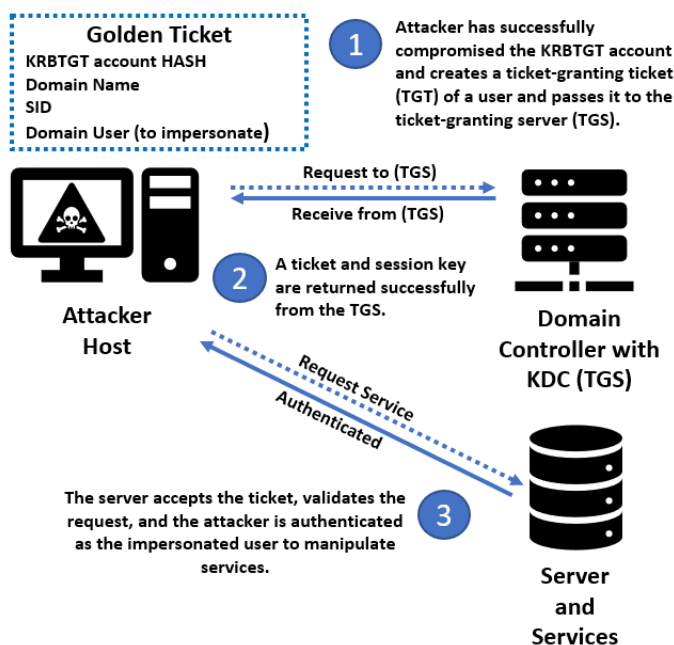


Figure 10.35 – Golden ticket attack

With a forged TGT in hand, a **ticket-granting service (TGS)** ticket can be requested from the authentication server and then sent to other resources to gain access. With the ability to impersonate any user, including a domain admin, the possibilities are almost limited to the creativity of the attacker as to what they can do next.

A **silver ticket** is a similar type of exploit that leverages a falsified Kerberos ticket, but unlike the golden ticket, a falsified TGT ticket is not passed to the KDC. In this attack, a TGS ticket is directly crafted by the attacker leveraging the extracted hash of the compromised service or computer account. From there, the forged TGS ticket can be passed directly to a target server's services (such as HOST, LDAP, WSMAN, RPCSS, and so on) and authenticated without them needing to contact the domain controller for validation. Using one or many forged TGS tickets, an attacker can execute different service types (such as WMI, PowerShell Remoting, and scheduled tasks) on the target system. For example, to connect to PowerShell Remoting, an attacker would pass a silver ticket for the HTTP and WSMAN service to open a shell on the target computer. Even though this temporarily limits the scope of access as the compromised computer account and target service may have limited operations, silver tickets are just as dangerous, not as easy to spot, and can ultimately lead to the same privilege escalation in the long run. A carefully crafted ticket can tell the target service that the computer account is a Domain Admin without fully verifying it against the domain controller. Typically, computer account passwords do not have any password changing requirements, so if the password becomes compromised, it may be valid for an extended period:

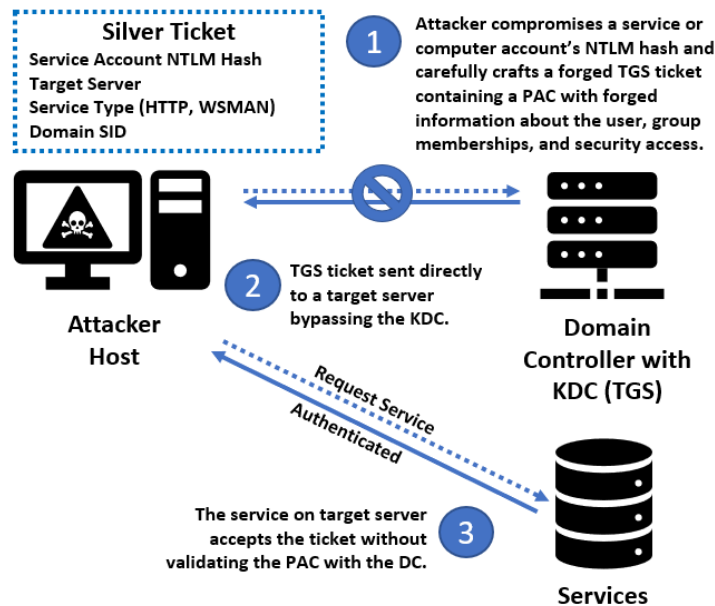


Figure 10.36 – Silver ticket attack

The following are a few recommendations that can help protect Kerberos authentication against these types of attacks:

- Protect privileged accounts, elevated service accounts, administrator accounts, and those that have logon rights to a domain controller. These accounts can be used to dump the KRBTGT account hash using tools such as **Impacket** (<https://github.com/SecureAuthCorp/impacket>) or **Mimikatz** (<https://github.com/gentilkiwi/mimikatz>).
- Enforce strict password requirements on privileged accounts and user-based SPNs to make it difficult for password cracking tools and Kerberoasting.
- Rotate the KRBTB service account password at regular intervals.
- Use endpoint security solutions to alert you if malicious tools and scripts are being used. Monitor for suspicious activity such as service principal creation, TGTs with long lifetimes, and unusual replication. Azure Defender for Identity is a great solution to help monitor against these types of attacks. You could even set up a honeypot account to act as bait and alert you if it's being targeted.

Next, let's look at two additional attacks that can be used to exploit Kerberos authentication.

Kerberoasting and AS-Rep roasting

Kerberoasting is a technique that's used to steal Kerberos TGS tickets by targeting services running with user-based service accounts. In a Kerberoasting attack, any compromised standard domain user can acquire legitimate proof of authentication from the KDC and use it to request the TGS tickets of services. Any service running with a user-based **service principal name (SPN)** can return a TGS ticket that can be captured and cracked using password-cracking tools. The compromised account does not need administrative privileges to do this. Unlike host-based SPNs (computer accounts) in silver ticket attacks, user-based SPNs are likely created with weaker passwords, are easily guessable, and set to never expire. Using a tool such as **Impacket**, the attacker can quickly query the entire domain with the compromised user account. In the following screenshot, the TGS ticket hash that was returned for the **MSSQLSvc** service is running as a user-based SPN:

```

impacket-GetUserSPNs prod. .com/csorensen:P@ssw0rd2022h3lp -request
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

ServicePrincipalName      Name      MemberOf
PasswordLastSet          LastLogon  Delegation
-----
MSSQLSvc/.prod. .com:1433  damt      CN=      _CollectedFilesAccess,CN=Users
m 2018-11-28 07:58:04.009653 2022-01-25 10:32:40.730618
MSSQLSvc/.prod. .com      damt      CN=      _CollectedFilesAccess,CN=Users
m 2018-11-28 07:58:04.009653 2022-01-25 10:32:40.730618

$krb5tgs$23$damt $PROD. .COM$prod. .com/damt *$d83bcc3988544fb81376a72c6
6c438ac56217702fca7de606fc12083390a0d9ae22065bbf22ac315b48bde4def8f4d14aa7b275acbc1fb216982e1ebc5
76b5b476b56ed5ba006ea4fb60351a99c2401364f89457f104282fb45a18ba4aa7ce1487a326bdec18589823eb13c567f
3907ec633acad4cb9bbfad5c940db315f9f20969f2f6fa34ff12eb92be763269e1965a6d84a61fb63a6942bb8e764ffe3

```

Figure 10.37 – Kerberoasting attack

The following are a few ways you can protect against Kerberoasting:

- Enforce strict password policy requirements with long, complex passwords for service accounts.
- Implement a PAM solution that can rotate passwords frequently and automate managing dependencies, such as restarting services or application pools.

If possible, enable AES Kerberos Encryption for service accounts to deter password cracking. By default, Active Directory will use RC4/DES, which is easier to crack offline with tools. AES Kerberos Encryption can be set by opening **Active Directory User and Computers**, opening the service account's properties, clicking the **Accounts** tab, and selecting either **This account supports Kerberos AES 128 bit encryption** or **This account supports Kerberos AES 256 bit encryption**.

Authentication Server Reply (AS-Rep) roasting isn't as prevalent today, but if your domain has poorly configured accounts that do not require Kerberos pre-authentication, it's possible to fall victim to an AS-Rep roasting attack. During the Kerberos authentication process, an encryption key containing the request timestamp that's been encrypted by the user's password hash is sent to the KDC to issue a TGT for use in authorization. The timestamp must match that of the KDC for the ticket to be issued during the reply process. If **Do not require Kerberos preauthentication** is selected as one of the user account attributes, the attacker can send a request to the KDC. They will be issued a ticket due to the lack of password requirements. However, this can later be cracked offline with a password cracking tool. Using a tool such as **Impacket**, the attacker can request the domain to return accounts that do not require pre-authentication and obtain the TGT ticket containing the targeted user's password hash. In the following screenshot, a standard domain user queried the domain for accounts that do not require Kerberos pre-authentication:

```

C:\> impacket-GetNPUsers prod. .com/csorensen:P@ssw0rd2022h3lp -request
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

Name      MemberOf      PasswordLastSet
  LastLogon  UAC
-----
emoreno CN= IT_Desktop,OU=Security Groups,DC=prod,DC= ,DC=com 2022-01-25 11:27:24.815106
      <never> 0x400200

$krb5asrep$23$emoreno@PROD. .COM:94d1676d817671b37b1d1bbb4aa364c4$bed5107b334ff3c80ffb53ca1c0
d41d18b252ea9b2400c387413aa71b6380311c67f3d2ebca3033edc7d08358651546c8476671858e48dbb6a4a59f4d6c3
d379ce0ead5b10accb0a59f14802a3ee914ce6e5eb8bd5ac4da3e9eec732abf6630a5bce9b854f87c991675d529336d6

```

Figure 10.38 – AS-Rep roasting attack

This option can be checked in **Active Directory Users and Computers** by opening a user account's properties, clicking on the **Account** tab, and scrolling down to **Account options**. As shown in the following screenshot, the account does not require pre-authentication:

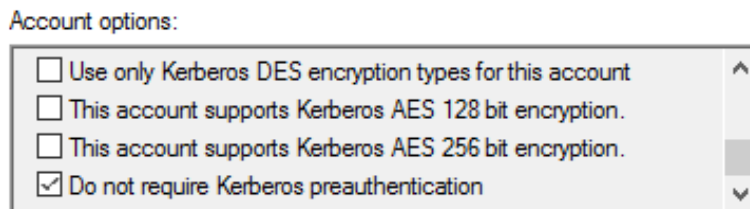


Figure 10.39 – Kerberos pre-authentication

Next, let's look at how an attacker can attempt to retrieve credentials from the host **operating system (OS)** and domain using techniques known as OS credential dumping.

Mitigating OS credential dumping

OS credential dumping is a post-exploitation technique that's used to steal stored credentials from the host OS. In a Windows environment, there are a few common places that attackers target to try and get this information. Successful exploits with credential dumping can affect not only local Windows hosts but put Active Directory domains and the Azure cloud at risk. For example, it's possible to extract a PRT token from an Azure AD registered device and use it to gain access to Azure and bypass security controls such as MFA. Let's look at a few places where OS credential dumping can occur. For reference, we are using the OS Credential Dumping MITRE ATT&CK technique. More information can be found at <https://attack.mitre.org/techniques/T1003/>.

- **Local Security Authority Subsystem Service (LSASS)** is a process that runs on Windows that handles Windows security policy enforcement by verifying account access. Domain credentials, local usernames, passwords, and Azure primary refresh tokens are stored inside the memory of the LSASS process and with the appropriate privileges, they can be extracted from memory.
- **Security Account Manager (SAM)** is a database stored in the Windows Registry that contains local account data. Accounts stored in the SAM are becoming less frequent, especially in fully Azure AD joined environments where the accounts aren't stored here. Nevertheless, if local administrator accounts exist on the system, they can be dumped from the SAM database.
- **Active Directory domain database (AD DS)** contains information about AD objects, including group memberships and password hashes. If the NTDS file can be copied, its contents can be extracted and its hashed passwords can be cracked using password cracking tools.
- **Local Security Authority (LSA) secrets** is the storage component of the local security authority and stores information such as user passwords and system and service account passwords that are used to run services in Windows. These secrets can be extracted, and the encrypted values can be decrypted into readable text.
- Cached domain credentials are stored on Windows systems locally as an alternative authorization mechanism to allow users to sign into their PCs if the domain is unreachable. These are stored in the SAM database and LSA secrets and can be extracted and cracked using password cracking tools.

- **Domain controller replication** (also known as **DCSync**) is a technique where an attacker will attempt to start the DC replication process by impersonating a domain controller. If successful, the replicated data in scope will include information such as AD objects, accounts, and password hashes that can be captured and cracked. This attack can be stealthy as it can be initiated without the attacker logging onto a domain controller directly, making it more difficult to detect.

Let's look at a few recommendations that can help detect and prevent OS credential dumping:

- Enable an **Endpoint Detection and Response (EDR)** solution that can detect when suspicious tools and scripts are installed and run. In the following screenshot, Defender for Endpoint detected when a command attempted to exfiltrate the SAM database in the registry, blocked the process, and issued an alert:

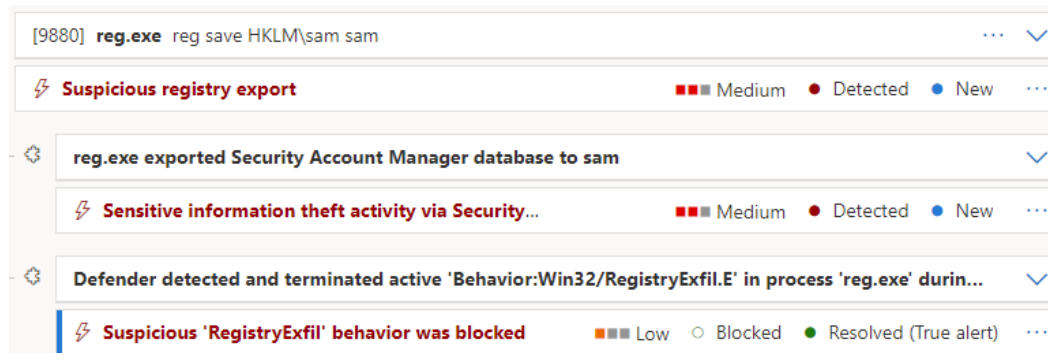


Figure 10.40 – Microsoft 365 Defender for Endpoint alert

- Use a solution such as Windows Defender Credential Guard to prevent exfiltration of the LSA. Credential Guard leverages VBS security to isolate the LSA process that stores secrets (Kerberos, NTLM, and Credential Manager) and can only be accessed by a small set of trusted OS binaries.
- Use **Attack Surface Reduction (ASR)** rules to block processes that attempt to steal credentials from Windows LSASS. As we mentioned earlier, LSASS is a process that handles security policy enforcement and can be targeted as a point of exfiltration.
- To protect the Active Directory NTDS database, ensure domain controller backups are stored, encrypted, and in a secure place where access is monitored. Ensure that access to domain controllers is managed with a PAM solution and that administrative activity is monitored.

- To protect against DCSync-style attacks, monitor domain controller logs for replication requests and tightly control the **access control lists (ACLs)** that set permissions for domain controller replication. In the following screenshot, Microsoft Defender for Identity created an alert for a suspected DCSync attack:

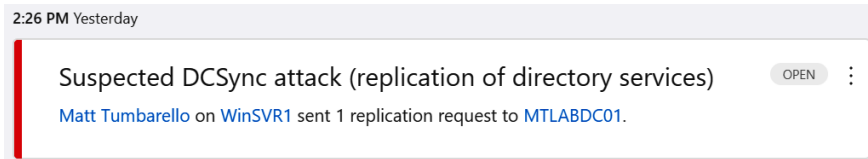


Figure 10.41 – DCSync alert from Defender for Identity

Next, let's learn how to prevent user access to the registry to help deter access from standard user accounts.

Preventing user access to the registry

The Windows Registry acts as a database in which settings and information are stored and retrieved by applications and system components. Users should not edit or modify the registry as it can cause incompatibilities and system stability issues. From a security and hardening perspective, the Windows registry can be used to install backdoors and used in evasion and persistence tactics, to name a few. Using Policy, it is possible to lock down the registry to some extent and prevent standard user accounts from opening registry editing tools. Doing this can help act as a deterrent to prevent unauthorized modification through the user's context, but sources with administrative privileges will still have access. You can prevent access to registry editing tools for users in Intune by using **Settings Catalog** and searching for **Prevent access to registry editing tools** in the **Settings picker** area. This setting can be seen in the following screenshot:

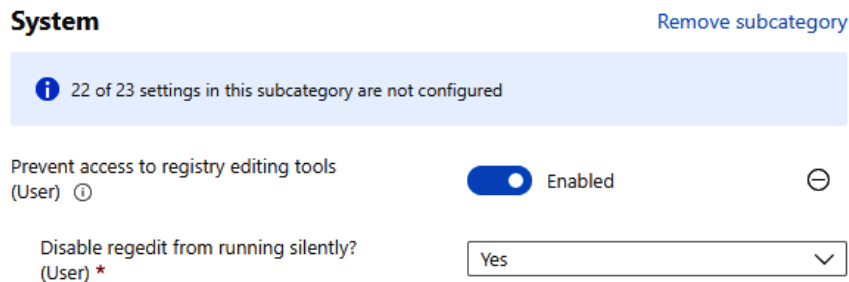


Figure 10.42 – Preventing user registry access

Once the policy has been applied, the next time a user goes to open `regedit.exe`, they will receive the following message:

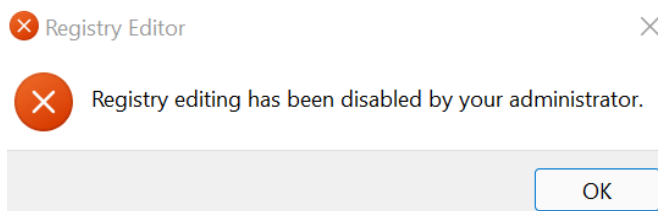


Figure 10.43 – Registry Editor disabled

To configure the same setting with Group Policy, go to **User Configuration | Policies | Administrative Templates | System**. Select **Prevent access to registry editing tools**. This will also help prevent the registry from being modified from the command line in the user context. For more information on hardening the Windows registry, MITRE has provided a list of mitigations that can be used to address known attack techniques for leveraging the registry at <https://attack.mitre.org/mitigations/M1024/>.

Now, let's learn how to protect users' privacy by reviewing Windows privacy settings.

Windows privacy settings

Windows has many great features that provide a personalized and enhanced connected experience for its users. To support this personalization, Windows has permission settings that control what data and device features that applications are allowed to access. A few examples include allowing an application to access the camera, device location, or microphone. Unless controlled by a policy, many of these privacy permissions are allowed by default and could pose a potential privacy risk for some organizations. To view the Windows privacy settings, open **Settings** and choose **Privacy & Security**. Here, you can get an idea of the types of permissions that are available to applications, such as access to speech settings, diagnostics and feedback, activity history, and more. Through **Settings**, you can granularly configure app-specific permissions or allow or deny all for each permission type.

Let's run through a few settings and where we can configure them using Intune. Note that some of these privacy permissions may need to remain enabled if you are using solutions such as Log Analytics or Endpoint Analytics in Microsoft Endpoint Manager to collect telemetry data from the endpoints.

The **Privacy & Security** settings are available in the **Intune Settings** catalog and **Templates**. If the policies don't exist in the UI, they can also be mapped using a custom template if a CSP is available, by pushing a registry key with PowerShell scripts, and so on. Let's look at a few places we can configure these settings as they are hard to find based on the friendly name shown in the Windows **Settings** app. You can search for them using **Settings Picker** in the **Settings Catalog** area:

- **Privacy & Security | General:**
 - Let apps show me personalized ads by using my advertising ID:
 - ♦ **Settings Catalog | Disable Advertising ID**
 - Let Windows improve Start and search results by tracking app launches:
 - ♦ **Settings Catalog | Turn off user tracking (User)**
 - Show me suggested content in the **Settings** app:
 - ♦ **Settings Catalog | Allow Online Tips**
- **Privacy & Security | Speech:**
 - Use your voice for apps using Microsoft's online speech recognition technology:
 - ♦ **Settings Catalog | All Input Personalization**
- **Privacy & Security | Inking & typing personalization:**
 - Personal inking and typing dictionary
- **Privacy & Security | Diagnostics & feedback:**
 - Diagnostic Data:
 - ♦ **Settings Catalog | Allow Telemetry**
 - Improve inking and typing:
 - ♦ **Settings Catalog | Allow Linguistic Data Collection**
 - Tailored experiences:
 - ♦ **Settings Catalog | Allow Tailored Experiences with Diagnostic Data (User)**

- Delete diagnostic data:
 - ♦ **Settings Catalog | Disable Device Delete**
- **Privacy & Security | Activity history:**
 - Store my activity history on this device:
 - ♦ **Settings Catalog | Publish User Activities**
 - Send my activity history to Microsoft:
 - ♦ **Settings Catalog | Upload User Activities**
- **Privacy & Security | Search permissions:**
 - SafeSearch
 - ♦ **Settings Catalog | Do Not User Web Results**
 - Cloud Content Search:
 - ♦ **Settings Catalog | Allow Cloud Search**

We didn't list every setting as some of them don't have mapped CSPs or Group Policy settings. It may be possible to configure them directly with registry keys, but that is outside the scope of this book.

Next, let's look at setting application-specific privacy permissions.

Controlling application privacy permissions

Using Intune, you can configure the access that specific applications have to privacy features. Most of these settings can be found in the **Settings Catalog** area by searching for **Privacy** in **Settings Picker**. For example, in the following screenshot, we have set the **Let Apps Access Camera** policy to **Force deny** and configured a list of allowed apps using **Let Apps Access Camera Force Allow These Apps**:

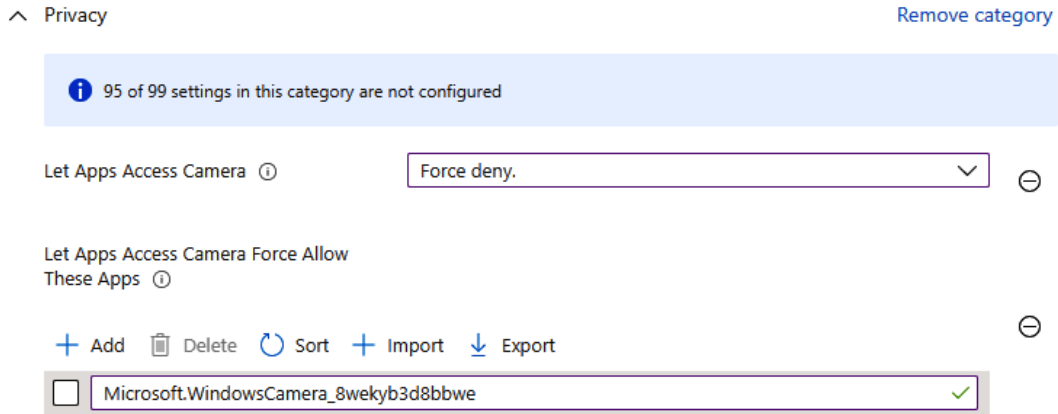


Figure 10.44 – Setting app permissions in Intune

Configuring an application allow list is only supported for Microsoft Store apps at the time of writing. To do this, you will need to gather the application's **Package Family Name (PFN)** using the Microsoft Store URL or PowerShell. For example, to find the PFN for the Camera app using PowerShell, run `Get-AppXPackage *Camera | Select Name, PackageFamilyName`, as shown here:

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> Get-AppXPackage *Camera | Select Name, PackageFamilyName

Name                                     PackageFamilyName
----
Microsoft.WindowsCamera Microsoft.WindowsCamera_8wekyb3d8bbwe
```

Figure 10.45 – Windows PackageFamilyName

Tip

You cannot control camera access to third-party apps selectively. Setting **Let Apps Access Camera** to **Force deny** will block third-party apps.

For more information about finding the package family name using PowerShell or the Microsoft app store, go to <https://docs.microsoft.com/en-us/mem/configmgr/protect/deploy-use/find-a-pfn-for-per-app-vpn>.

Additional privacy settings

Let's look at a few additional privacy settings that you should consider that are not listed in the **Privacy & Security** settings. It's worth evaluating them and determining if they should be disabled on company devices, depending on your privacy controls:

- **Settings Catalog | Allow Game DVR.** Disabling this policy will block Windows Game Recording and Broadcasting.
- **Settings Catalog | Disable Privacy Experience.** Disabling this policy may prevent new users from changing company-managed privacy settings when they log on for the first time.
- **Settings Catalog | Turn off toast notifications on the lock screen (User).** Enabling this policy will prevent toast notifications from displaying on the lock screen.
- **Settings Catalog | Allow Cortana Above Lock.** Disabling this setting will prevent a user from interacting with Cortana on the lock screen using speech.
- **Settings Catalog | Allow Windows Spotlight (User).** Disabling this policy will turn off consumer features and Windows tips on the lock screen.
- **Settings Catalog | Allow Advertising.** Disabling this policy will prevent the device from sending out Bluetooth advertisements. We covered additional Bluetooth security settings in *Chapter 4, Networking Fundamentals for Hardening Windows*.
- **Settings Catalog | Allow Location.** Disabling this policy will prevent apps from accessing location services, including Cortana and Windows search.

Summary

In this chapter, we learned how standard computer communications protocols are used as attack vectors in MiTM attacks. We learned how to configure policies to mitigate them and that in a well-administered network, protocols such as LLMNR, NBT-NS, mDNS, and WPAD can be disabled. Next, we talked about relay attacks and covered different security settings that can be used to protect against exploits that target Kerberos authentication, SMB, LDAP, IPv6, and ARP. After that, we covered how an attacker can use discovery tactics to move laterally and escalate privileges. We reviewed different attack techniques, such as golden and silver tickets, that can be used to exploit Kerberos authentication, and covered areas targeted to steal credentials on an OS.

Finally, we reviewed the privacy settings that are listed in the **Privacy & Security** section of the Windows Settings app. We discussed how to control these settings using Intune and listed where to find the relevant policies in the Intune Settings catalog.

In the next chapter, we are going to look at server infrastructure management, including the importance of implementing access management solutions and how to use Azure services to connect and manage Windows servers remotely.