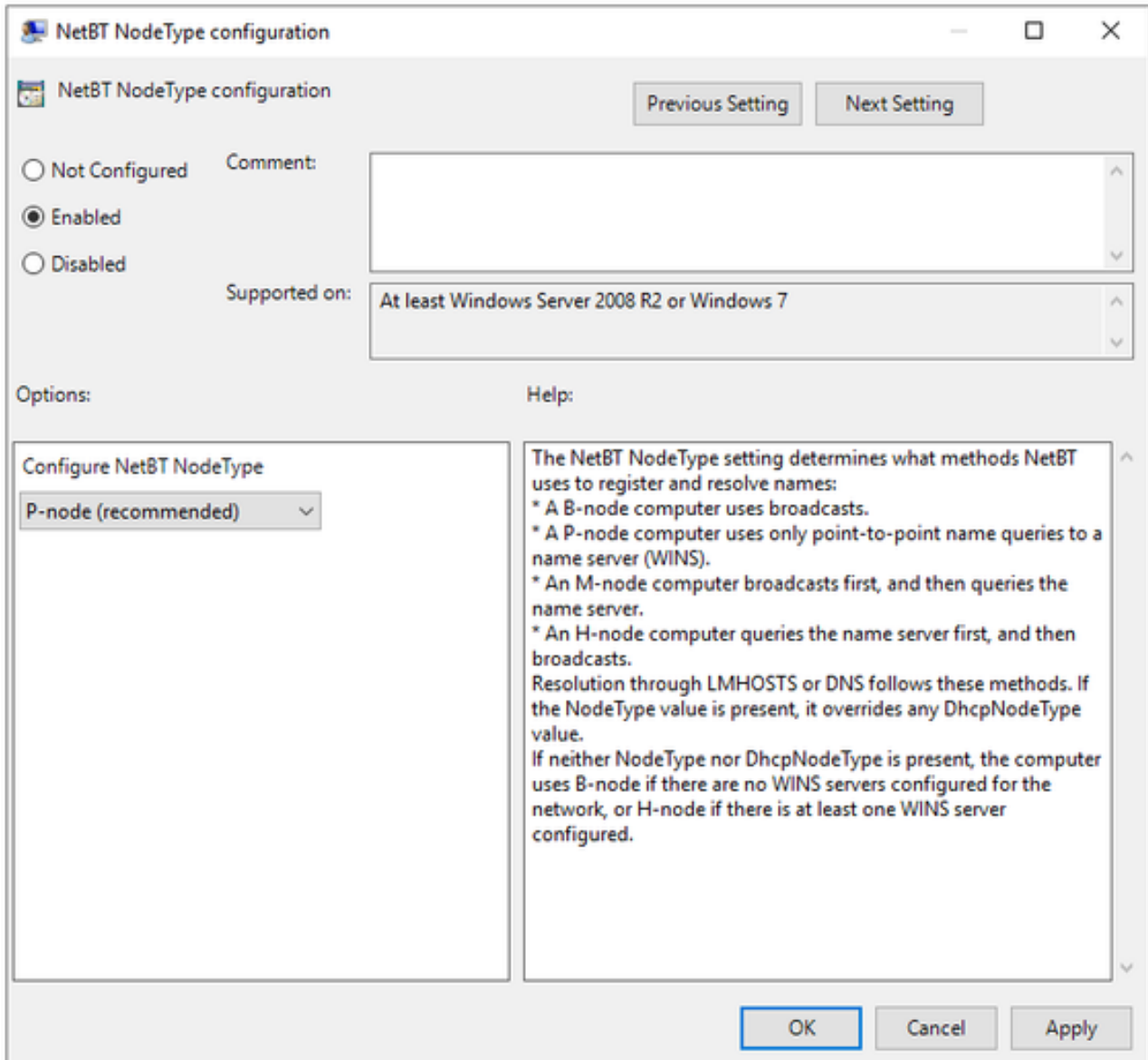


Active Directory Hardening Series - Part 6 – Enforcing SMB Signing

 techcommunity.microsoft.com/blog/coreinfrastructureandsecurityblog/active-directory-hardening-series---part-6---enforcing-smb-signing/4272168



The image shows the 'NetBT NodeType configuration' window. It has a title bar with standard Windows window controls. Below the title bar, there are 'Previous Setting' and 'Next Setting' buttons. The main area contains three radio buttons: 'Not Configured', 'Enabled' (which is selected), and 'Disabled'. To the right of these is a 'Comment:' text box. Below the radio buttons is a 'Supported on:' section with a text box containing 'At least Windows Server 2008 R2 or Windows 7'. Under the 'Options:' label, there is a 'Configure NetBT NodeType' section with a dropdown menu currently set to 'P-node (recommended)'. To the right of this is a 'Help:' section with a text box containing detailed information about NetBT NodeType settings. At the bottom right, there are 'OK', 'Cancel', and 'Apply' buttons.

NetBT NodeType configuration

Previous Setting Next Setting

☐ Not Configured Comment:

☒ Enabled

☐ Disabled

Supported on: At least Windows Server 2008 R2 or Windows 7

Options:

Configure NetBT NodeType

P-node (recommended)

Help:

The NetBT NodeType setting determines what methods NetBT uses to register and resolve names:

- * A B-node computer uses broadcasts.
- * A P-node computer uses only point-to-point name queries to a name server (WINS).
- * An M-node computer broadcasts first, and then queries the name server.
- * An H-node computer queries the name server first, and then broadcasts.

Resolution through LMHOSTS or DNS follows these methods. If the NodeType value is present, it overrides any DhcpNodeType value.

If neither NodeType nor DhcpNodeType is present, the computer uses B-node if there are no WINS servers configured for the network, or H-node if there is at least one WINS server configured.

OK Cancel Apply

Blog Post

Hi everyone! [Jerry Devore](#) here to continue the [Active Directory Hardening series](#) by addressing SMB signing. Many of my Microsoft colleagues have already written some great content on SMB signing so I was not going to cover it. However, it is just too critical a security control to skip and a series on Active Directory hardening would not be complete without it. As usual, my goal is to help clear up any confusion so you can enable this setting if you have not already.

Why does SMB signing matter?

The two most recognized benefits of SMB signing are ensuring message integrity and preventing an NTLM relay attack. Exploiting both of those typically involves an adversary-in-the-Middle (AiTM). Before we move on let's clarify how attackers can place themselves between a victim and a resource.

How does the adversary get in the middle?

Some people picture an AiTM as somebody who is physically in the building and lurking around in a network closet. In reality, an AiTM is most often not physically present but instead remotely controlling an organization's device. Once on that device, tools like Responder can be used to listen for broadcast name resolution requests coming from other devices on the network. Such network traffic occurs when DNS cannot resolve a name, and the client uses **LLMNR** (Link-Local Multicast Name Resolution) or **NBT-NS** (NetBIOS Name Service) to make a last-ditch effort to locate the resource. When Responder detects such packets, it responds and claims to be the device name the victim requested. The victim then requests to start a session with the device under the AiTM's control and the fun begins.

Organizations can reduce this risk by disabling broadcast name resolution via LLMNR and NBT-NS on devices. Of the two, LLMNR is the most straightforward to disable given you can simply configure the setting **Turn off multicast name resolution** via GPO, Intune or the registry (HKLM\Software\Policies\Microsoft\Windows NT\DNSClient\EnableMulticast=0). Disabling LLMNR will not negatively impact devices in environments where DNS can resolve all required names. However, in less structured networks users may notice a difference. For example, connecting to a printer on a home network may rely on LLMNR to resolve the IP address of the printer.

When it comes to disabling NBT-NS many times it is recommended to disable NetBIOS on the NIC in the TCP/IP properties or in the registry. The challenge with that approach is that it is difficult to manage at scale since every NIC has a unique GUID. A better solution is to configure the NetBT NodeType at the host level to P-node. When in that mode the client will only attempt to use WINS and will no longer use broadcasts to resolve NetBIOS names. To be clear, I am not suggesting you re-introduce WINS servers (may they rest in peace). P-node is just a way to completely disable NetBIOS name resolution when a WINS server is not configured.

You will not find a native GPO setting to configure NodeType but the baselines published as part of the [Microsoft Security Compliance Toolkit](#) contain an .admx file (SecGuide.admx) which will add an Administrative Template named **MS Security Guide** that has a setting named **NetBT NodeType configuration**. Alternatively, you could manage the registry directly (HKLM\SYSTEM\CurrentControlSet\Services\NetBT\Parameters\NodeType=2)

Message Integrity

Message integrity for SMB parallels what I explained about LDAP signing in my [article](#) on that topic. As with LDAP signing, the SMB client and server establish a symmetrical session key during authentication. The details about how that key is generated and exchanged varies based on the authentication protocol. If you want to go down the rabbit hole to learn the finer details this [article](#) is a good place to start. For now, the important thing to remember is that acquiring the session key on the client is dependent on knowing the user credential.

Once the session key has been established, a hash of each message is generated and signed using the session key. That signature is then placed in the SMB header of the packet. The recipient of the message will then hash the received message and compute the signature. If the signature matches the one in the header the recipient is assured the message was not modified while in transit.

NTLM Relay

Now that we have covered AiTM and message integrity, let's tackle NTLM relay. NTLM is a challenge \ response protocol. The client contacts the resource and negotiates which authentication protocol will be used. If NTLM is selected, the resource server returns a challenge (random number referred as a nonce). The client will then encrypt the challenge using the user's NTLM hash to produce the "response" and sends it to the server. The resource server will then forward the response to a domain controller to have it validated. If the domain controller determines the correct NTLM hash was used to generate the response, it will return a NETLOGON_VALIDATION_SAM_INFO4 message to the resource server which contains the user's SIDs.

A version of the following diagram is typically used to explain how an AiTM could manipulate the authentication flow and perform a "relay" of NTLM credentials. In this example [Alice](#) is our proverbial victim. Let's assume she mistyped the name of a resource which caused her laptop to perform a broadcast looking for the misspelled resource after the DNS query failed. The attacker's computer immediately responds, and the two devices negotiate to use NTLM. Rather than generate a challenge, the attacker contacts a resource that he would like to authenticate to as Alice. That resource server sends the attacker a challenge which is promptly relayed to Alice. Alice's laptop uses her NTLM hash to encrypt the challenge and sends the response back to the attacker. The attacker then forwards the response to the resource server and after having it validated by the domain controller, he is authenticated to the resource server as Alice.

You might be wondering what SMB signing has to do with the flow of NTLM authentication. I am glad you asked. Many times, SMB acts as a transport protocol for NTLM authentication traffic. By securing SMB traffic with signing, we can protect such NTLM traffic from being relayed. Remember when I mentioned possession of the session key requires knowledge of the user's credentials? In the above relay scenario, the attacker does not know the user's credential. He only tricked Alice into producing and

returning a valid response to the challenge. If the resource server required SMB signing, the relay attempt would have failed since the attacker does not have the session key required to sign the messages.

It is worth pointing out that the AiTM might be able to deduce a victim's NTLM hash in this scenario by sending a pre-computed challenge. Once the victim returns the response the attacker could use a rainbow table of pre-generated NTLM hashes (using guessable passwords) which are then used to encrypt the challenge to produce a list of possible responses. If the attacker can match the victim's response to one of the pre-computed responses, he ultimately knows the user's credential. To mitigate such attacks, disable NTLMv1 across the environment and impose strong password requirements such as banning guessable passwords via Entra Password Protection.

Enforce Signing

For a detailed explanation of the settings to configure SMB signing I am going to direct you to the articles linked below rather than recreate that information. I suggest you read all of them but in the meantime, here is a summary of what you will find.

- SMB signing has been supported by Windows since Windows 98 and NT 4.0. There is no need to defer enforcing signing out of fear that some of your Windows devices lack compatibility.
- There are client and server-side settings for enabling and enforcing signing. If either the client or the server has signing enabled or enforced, signing will be negotiated for the session regardless of the setting on the other side.
- The settings Microsoft network client: Digitally sign communications (if server agrees) and Microsoft network server: Digitally sign communications (if client agrees) are legacy and only apply to SMBv1. If SMBv1 is disabled, you don't need to enable these settings. However, they are often flagged in security audits so you might want to enable them just to avoid having to explain to an auditor that the settings add no value.
- The settings Microsoft network client: Digitally sign communications (always) and Microsoft network server: Digitally sign communications (always) apply to all version of SMB. Another way they differ from the legacy setting is that they enforce the use of signing and will terminate the session if the other side does not support signing. In contrast, the "if agrees" setting will allow sessions to continue without signing if the other side lacks support.
- Historically there has been a performance impact from SMB signing. However, with each version of SMB, performance has improved and with SMBv3 impact is negligible. Organizations need to determine whether performance or security is the priority before deciding to forgo SMB signing. Given the advances in hardware capacity, most customers find the overhead from signing to be acceptable even for SMBv1 but it is recommended to baseline your performance before and after enabling signing.

- The strength of SMB signing is dependent on both the authentication method and SMB version. NTLMv1 is most prone to AiTM attacks which could allow an attacker to gain credentials and acquire the session keys.

Auditing

Organizations have been hesitant to enforce signing due to uncertainty of non-supporting devices or applications. Historically there has been no logging to identify sessions without signing but Windows 11 24H2 (and Server 2025) introduced new events to address that need. Below is an example of a 3021 event showing when a connected SMB client did not support signing. A similar event (Event ID 31998) can be logged in Microsoft-Windows-SMBClient/Audit when a SMB server does not support signing. The steps to enable that audit are explained in this [article](#).

Currently there are no plans to backport this new logging to earlier versions of Windows. For them a possible alternative approach is to use network captures to perform some analysis. At a glance that sounds like looking for a needle in a haystack so here are a few tips to help make it feasible.

1. Use a capture filter to only capture SMB traffic. That will help reduce the size of your capture files. You could also filter the captures by IP address if you are analyzing a particular device.
2. Use a display filter in Wireshark to only display SMB packets that are not using signing.
 - For SMBv1 use the filter **smb.flags2.sec_sig == 0 && !(smb.cmd == 0x72)** to display unsigned messages once the session has been authenticated.
 - For SMBv2+ use **smb2.flags.signature == 0 && smb2.cmd != 0x00 && smb2.cmd != 0x01 && smb2.cmd != 0x0F**. That filter will also only show packets from established sessions which are not using signing.
 - If you want to confirm the filters are working simply change ==0 to ==1 and you will see the signed SMB messages.

I hope this information helps you in your journey to better security. Y'all seem to like my career saving **Do's and Don'ts** tips so here they are.

- **Don't** put off the effort to enforce SMB signing. The exploits for unsigned SMB are not just theoretical. Pentesters, red teamers and adversaries all agree their job is much easier when SMB signing is not implemented.
- **Do** know that this is not an impossible task. When organizations finally get off the fence, they usually realize they don't have widespread compatibility issues with enforcing SMB signing.
- **Don't** overlook 3rd party devices given they are the most common source of unsigned SMB messages. In particular, old appliances and multi-function printers are notorious for not having SMB signing enabled.

- **Don't** just require SMB signing on domain controllers. It is just as critical for endpoints and member servers to **require** signing. For a slow rollout you might start with enabling **Microsoft network client: Digitally sign communications (always)** on groups of devices at a time then tackle **Microsoft network server: Digitally sign communications (always)** in groups.
- **Do** improve the robustness of SMB signing by configuring your environment to use the strongest possible authentication and SMB dialect versions. If you haven't heard NTLM was deprecated in June 2024. I plan to address NTLM disablement in this series. In the meantime, check out this blog post and video.
- **Do** disable LLMNR and NBT-NS on all organizational devices if you haven't already. You should also review how DNS records are secured to limit the opportunity for records to be hijacked.
- **Do** know that SMB signing is now required by default on all SMB client and server connections in Windows 11 24H2 and on all client connections in WS2025.
- **Do** deploy Windows 11 22H2 (released 10/1/24) and enable the SMB auditing settings so you can start to identify any SMB server devices that do not support signing.
- **Do** share your lessons learned from enforcing SMB signing in the comments section.
- **Do** checkout these article on SMB signing:

Updated Nov 08, 2024

\n

- For SMBv1 use the filter **smb.flags2.sec_sig == 0 && !(smb.cmd == 0x72)** to display unsigned messages once the session has been authenticated.

\n

- For SMBv2+ use **smb2.flags.signature == 0 && smb2.cmd != 0x00 && smb2.cmd != 0x01 && smb2.cmd != 0x0F**. That filter will also only show packets from established sessions which are not using signing.

\n

- If you want to confirm the filters are working simply change ==0 to ==1 and you will see the signed SMB messages.

\n

\n

\n

- For SMBv1 use the filter **smb.flags2.sec_sig == 0 && !(smb.cmd == 0x72)** to display unsigned messages once the session has been authenticated.

\n

- For SMBv2+ use **smb2.flags.signature == 0 && smb2.cmd != 0x00 && smb2.cmd != 0x01 && smb2.cmd != 0x0F**. That filter will also only show packets from established sessions which are not using signing.

\n

- If you want to confirm the filters are working simply change ==0 to ==1 and you will see the signed SMB messages.

\n

\n