POC 2024

A journey into forgotten Null Session and MS-RPC interfaces

Haidar Kabibo

Kaspersky

- Member of Industrial Security Services team, Kaspersky
- Masters of None:
  - Pentest
  - RE
  - AppSec
  - ICS
  - Network
  - Radio

$ echo d2hvYW1pCg== | base64 –d | bash

Sud0Ru

What this talk about?

Agenda

1. How It All Started
2. DCE-RPC Crush Course
3. Null Session
4. Research Methodology
5. Enumerate Domain Info
6. Prevention
7. Detection
8. MSRPC security
9. MS-NRPC Security Surface analysis
10. MS-NRPC Security In-Depth analysis

How it all started?

- Intersecting MSRPC call during traffic analysis
- DC network interfaces in cleartext

Source	Destination A	Protocol	Info
.26	.14	DCERPC	Bind: call_id: 2, Fragment: Single, 2 context items: IOXIDResolver V0.0 (32bit NDR), IOXIDRes
.14	.26	DCERPC	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: 5840, 2 results: Acceptance,
.26	.14	IOXIDResolver	ServerAlive2 request IOXIDResolver V0
.14	.26	IOXIDResolver	ServerAlive2 response

```
Address: STRINGBINDINGs=5, SECURITYBINDINGs=7
    NumEntries: 86
    SecurityOffset: 64
  > StringBinding[1]: TowerId=NCACN_IP_TCP, NetworkAddr=
  > StringBinding[2]: TowerId=NCACN_IP_TCP, NetworkAddr=      14"
  > StringBinding[3]: TowerId=NCACN_IP_TCP, NetworkAddr=      14"
  > StringBinding[4]: TowerId=NCACN_IP_TCP, NetworkAddr=      .5"
  > StringBinding[5]: TowerId=NCACN_IP_TCP, NetworkAddr=      .12"
  > SecurityBinding[1]: AuthnSvc=0x0009, AuthzSvc=0xffff, PrincName=""
```

How it all started?

- There were no authentication header, auth length=0, which means RPC auth level is None
- After some googling, Airbus research [1] about enumerating network interface without authentication using IOXIDReslover interface surfaced

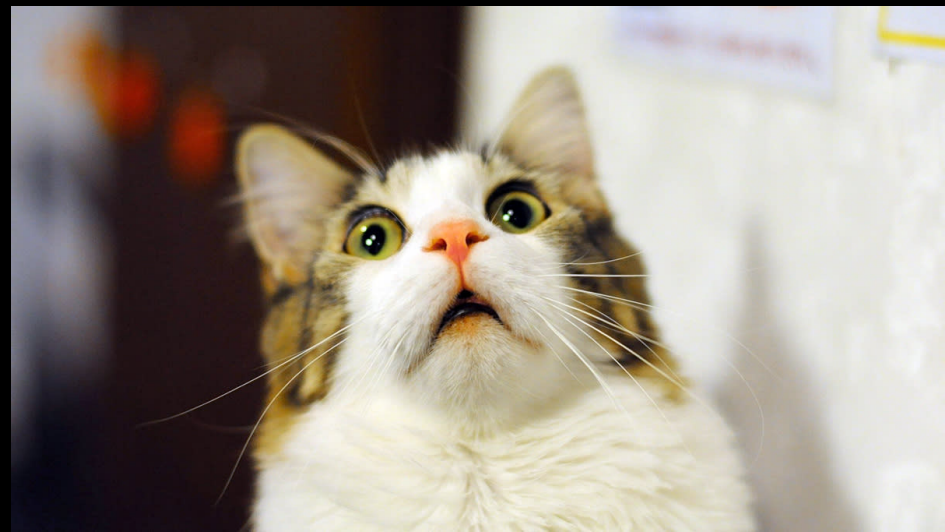
```
Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind, Fragment: Single,
    Version: 5
    Version (minor): 0
    Packet type: Bind (11)
  > Packet Flags: 0x03
  > Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)
    Frag Length: 116
    Auth Length: 0
    Call ID: 2
```

```
#define RPC_C_AUTHN_LEVEL_DEFAULT       0
#define RPC_C_AUTHN_LEVEL_NONE          1
#define RPC_C_AUTHN_LEVEL_CONNECT       2
#define RPC_C_AUTHN_LEVEL_CALL          3
#define RPC_C_AUTHN_LEVEL_PKT           4
#define RPC_C_AUTHN_LEVEL_PKT_INTEGRITY 5
#define RPC_C_AUTHN_LEVEL_PKT_PRIVACY   6
```

[1] https://www.cyber.airbus.com/the-oxid-resolver-part-1-remote-enumeration-of-network-interfaces-without-any-authentication/

RPC,DCE/RPC, MSRPC

- Remote Procedure Call, also known as a function call or a subroutine call, is a protocol that uses the client-server model in order to allow one program to request service from a another program
- DCE/RPC is a special implementation of RPC for Distributed Computing Environment (DCE)
- MSRPC stands Microsoft Remote Procedure Call. It is a specific implementation of the Remote Procedure Call [1]



[1] https://learn.microsoft.com/en-us/windows/win32/rpc/rpc-start-page

MSRPC Architecture: Terms

Interface Exposure:
A process exposes its functionality through interfaces

Unique UUID (Universal Unique Identifier) and version:
Each interface is uniquely identified by a UUID (IID) and version

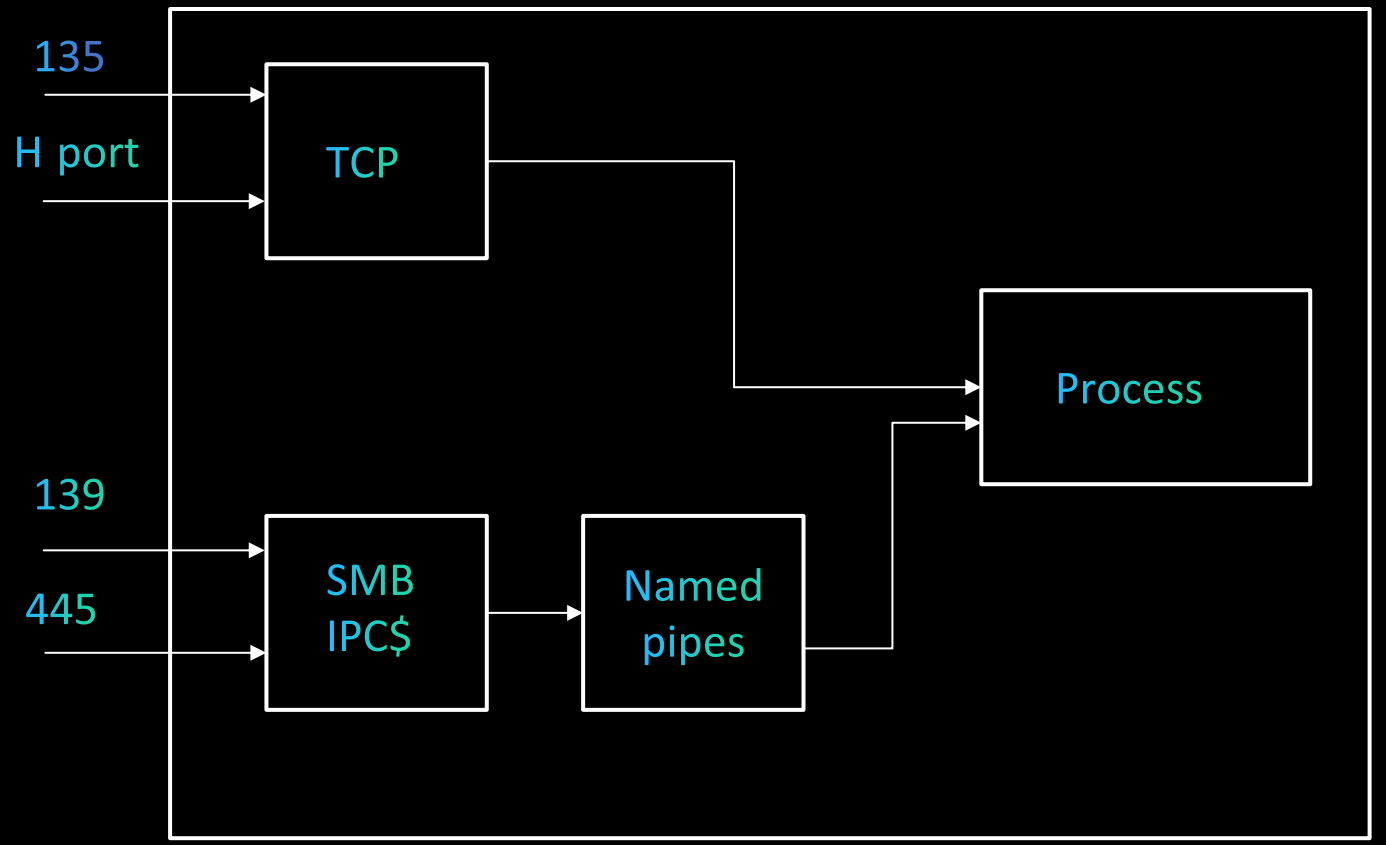
Binding:
In order to call a procedure or function on a remote server, it needs to bind to the appropriate interface using its unique IID

Procedure Call:
After binding Client can call a method inside interface by request its OPNUM

MSRPC Architecture: Transport Layers

- TCP:
  stringbinding: “ncacn_ip_tcp:192.168.177.132[135]”
  ncacn_ip_tcp: this is protocol sequence
  192.168.177.132: this is network address
  135: this is the endpoint

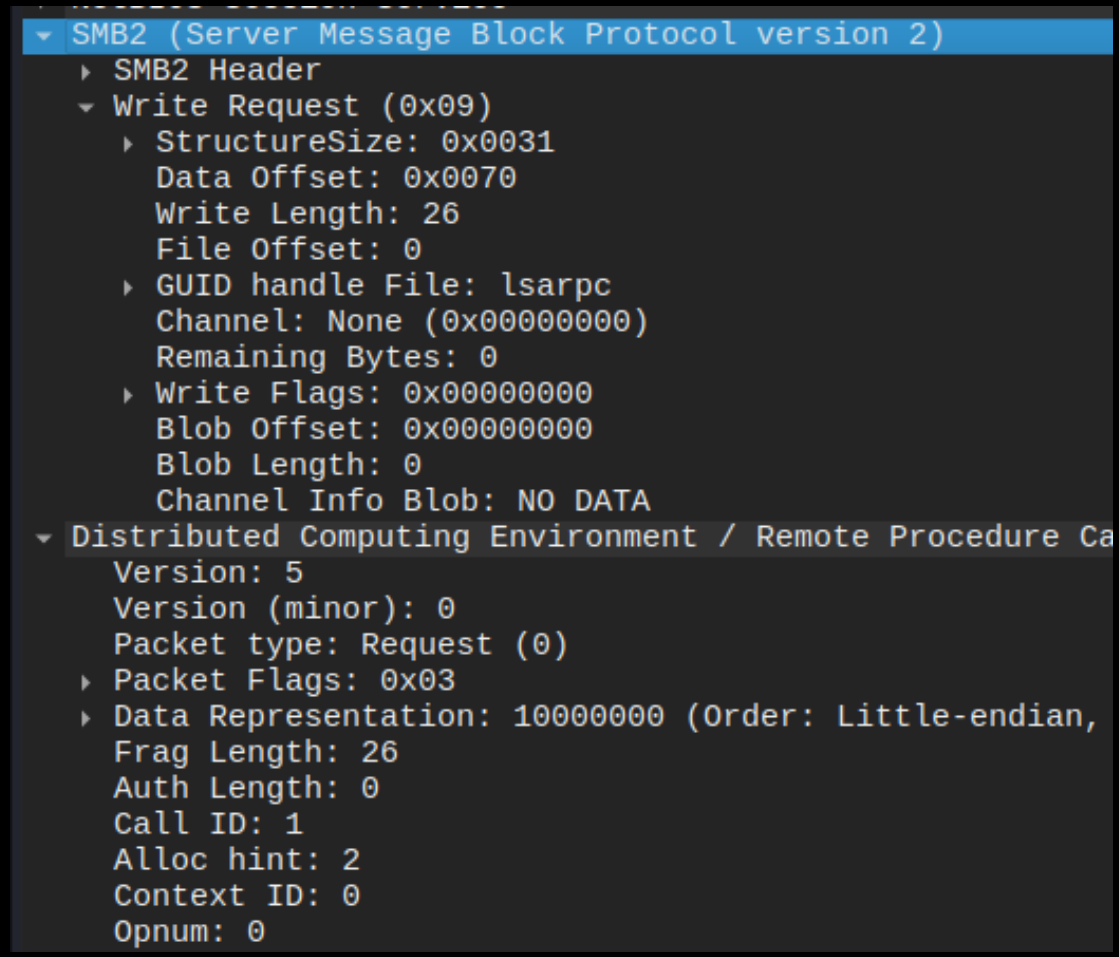
- SMB:
  stringbinding: “ncacn_np:192.168.0.1[\pipe\spoolss]”
  ncacn_np : is protocol sequence
  192.168.0.1 : is network address
  \pip\spoolss: is endpoint



MSRPC Architecture: communication

TCP:
1- Make a TCP connection to IP/PORT
2- Bind the interface with specific UUID
3- Server send the binding status
4- Call the function by make request with a specific OPNUM

SMB:
1- Make an SMB connection to the remote host
2- Tree connect to the $IPC share
3- Open the file (\pipe\spoolss)
(nt create andx request)
4- Bind the interface with specific UUID
(write request to the pipe)
5- Getting the binding status (SMB read request)
6- Call the function by make request with specific OPNUM
(SMB write request)



What Is Null Session?

- Null session is used when the access to network resource, most commonly the IPC$ "Windows Named Pipe" share, granted without authentication.
- Gather information such as shares, users, groups, registry keys and much more
- When you upgrade your server to domain controller this names pipes can acceded through null session “\pipe\netlogon”, “\pipe\samr”, and “\pipe\lsarpc”
- To prevent null session, two related system policies are “Restrict anonymous access to Named Pipes and Shares” and “Network access: Named Pipes that can be accessed anonymously”

Null Session VS Authentication Level


```
#define RPC_C_AUTHN_LEVEL_DEFAULT       0
#define RPC_C_AUTHN_LEVEL_NONE          1
#define RPC_C_AUTHN_LEVEL_CONNECT       2
#define RPC_C_AUTHN_LEVEL_CALL          3
#define RPC_C_AUTHN_LEVEL_PKT           4
#define RPC_C_AUTHN_LEVEL_PKT_INTEGRITY 5
#define RPC_C_AUTHN_LEVEL_PKT_PRIVACY   6
```

- The null session and the authentication level is not the same
- The null session is related to named pipes inside IPC$ share (SMB authentication)
- The null session in this case affected the transport layer
- For Interfaces we have the **Binding authentication** which affected with authentication level
- Our goal is to concentrate to interfaces that vulnerable to auth level = 1 and that used TCP endpoints

Restrict Unauthenticated RPC Clients policy

- “Computer Configuration\Administrative Templates\System\Remote Procedure Call”
- System policy that controls how the RPC server runtime handles unauthenticated RPC clients connecting to RPC servers
- “Authenticated”: The RPC runtime will block access to TCP clients that have not authenticated with some exceptions
- “Authenticated without exceptions”: All unauthenticated connections are blocked
- “None”: All RPC clients are allowed to connect to RPC servers running on the machine

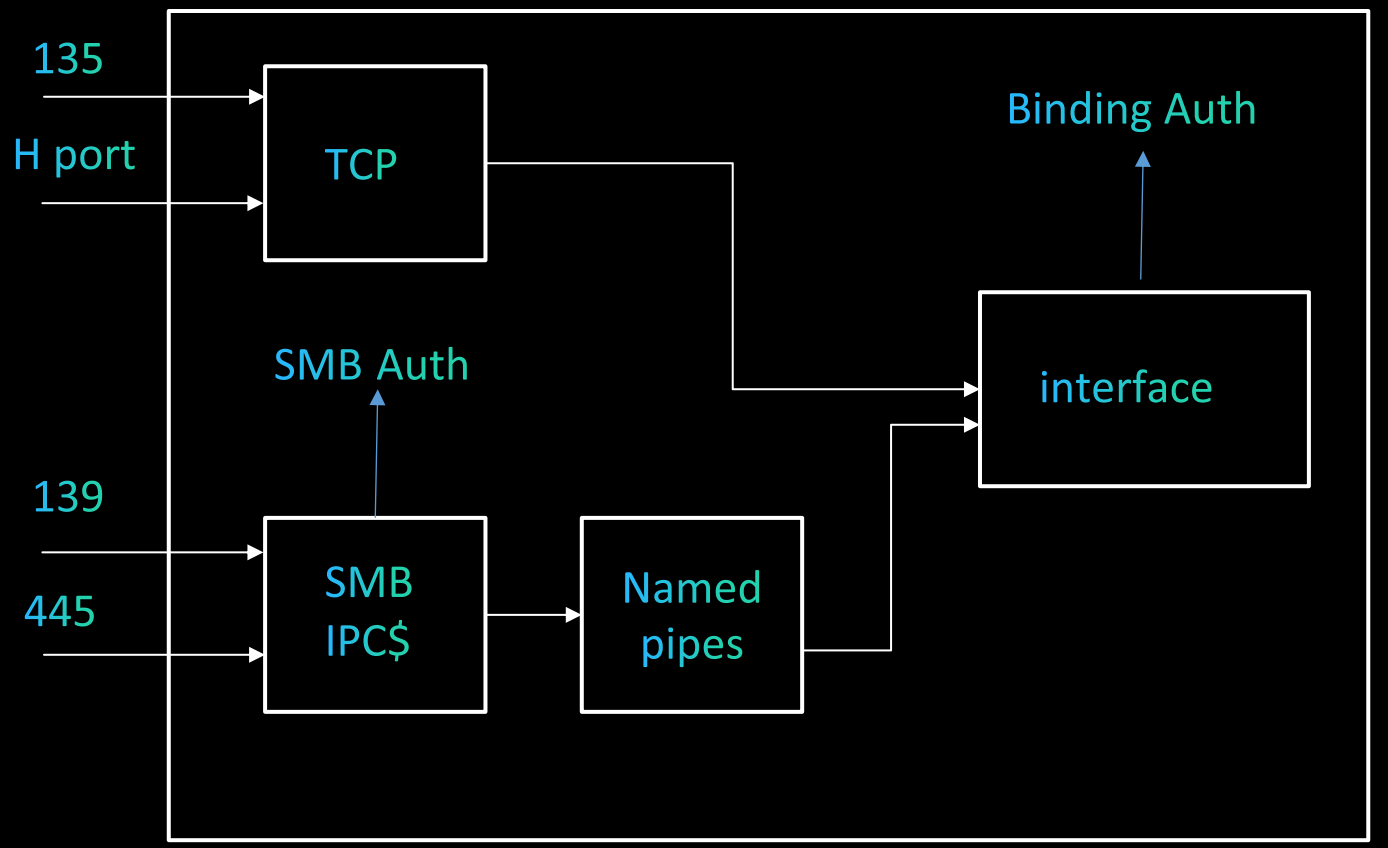
Put It All Together

No auth path
Auth-level =1

Restrict Unauthenticated RPC Clients policy = authenticated

NULL session path
'%'

Restrict anonymous access to Named Pipes and Shares



But why we can still bind IOXIDResolvler without authentication

???

Security Research Roadmap Against MSRPC Interfaces

Overview Of System Security Policy For Target System

It’s Windows server 2022 act as DC

1- No null session through IPC$:
“Restrict anonymous access to Named Pipes and Shares” is enabled
“Network access: Named Pipes that can be accessed anonymously” is not defined

2- Restrict Unauthenticated RPC Clients policy is Authenticated

Research Roadmap: Enumerating RPC Endpoints

1- Using Endpoint mapper:

- rpcdump.py is impacket script to enumerate RPC endpoints
- rpcdump uses endpoint mapper service
- Endpoint mapper maintain a dynamic database that map endpoints to the UUIDs
- The endpoint mapper service can be acceded through RPC interface call rpcmapper
- This interface can be acceded through TCP port 135.
- rpcdump.py bind rpcmapper and call a lookup method (opnum 2)
- Syntax: rpcdump.py IP_address
- In our research we will focus on interfaces that used TCP as transport layer

```
156 Protocol: [MS-DRSR]: Directory Replication Service (DRS) Remote Protocol
157 Provider: ntdsai.dll
158 UUID    : E3514235-4B06-11D1-AB04-00C04FC2DCD2 v4.0 MS NT Directory DRS Interface
159 Bindings:
160           ncacn_np:\\WIN-SO9H290I5NL[\pipe\b9459c55c28cac8a]
161           ncacn_http:192.168.177.177[49670]
162           ncalrpc:[NTDS_LPC]
163           ncalrpc:[OLE395259F93F9D9566C47334E3284B]
164           ncacn_ip_tcp:192.168.177.177[49668]
165           ncacn_ip_tcp:192.168.177.177[49664]
166           ncalrpc:[samss lpc]
167           ncalrpc:[SidKey Local End Point]
168           ncalrpc:[protected_storage]
169           ncalrpc:[lsasspirpc]
170           ncalrpc:[lsapolicylookup]
171           ncalrpc:[LSA_EAS_ENDPOINT]
172           ncalrpc:[lsacap]
173           ncalrpc:[LSARPC_ENDPOINT]
174           ncalrpc:[securityevent]
175           ncalrpc:[audit]
176           ncacn_np:\\WIN-SO9H290I5NL[\pipe\lsass]
```

Research Roadmap: Enumerating RPC Endpoints

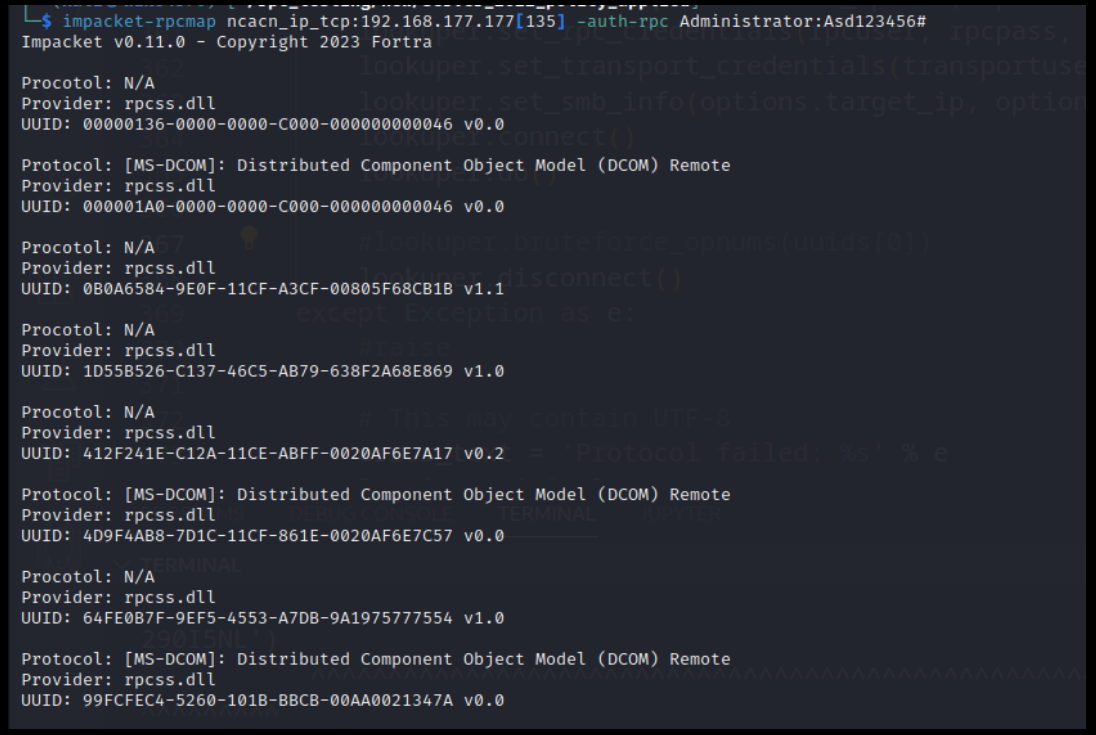
2- Using NMAP full port scan:

- Many of endpoints are mapped to high TCP dynamic ports

```
└─# nmap -p- -n 192.168.177.177 --min-rate=10000
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-25 09:19 EDT
Nmap scan report for 192.168.177.177
Host is up (0.00100s latency).
Not shown: 65515 filtered tcp ports (no-response)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
5985/tcp  open  wsman
9389/tcp  open  adws
49664/tcp open  unknown
49667/tcp open  unknown
49668/tcp open  unknown
49670/tcp open  unknown
49671/tcp open  unknown
49681/tcp open  unknown
49689/tcp open  unknown
```

Research Roadmap: Collecting Interfaces UUIDs

- Rpcmap.py another Impakcet script that is used to enumerate all interface inside an endpoint
- It used MGMT interface
- Each endpoint should implement mgmt interface
- We will use endpoints that collected from previous stage.
- MGMT interface can’t called under our system predefined policy without using authentication
- To bypass this problem you can feed the rpcmap a valid creds



Research Roadmap: Brute-force OPNUMs

- rpcmap.py involves brute-forcing the OPNUMs for a specific interface
- At this stage we should use auth-level 1
- rpcmap will give you access dined in every attempt
- rpcmap bind MGMT interface before brute-forcing, to be sure that the interface can be accessed through this endpoint
- MGMT not affected with no-auth so we will get access dined for every bind attempt
- Solution: we should change rpcmap internal work

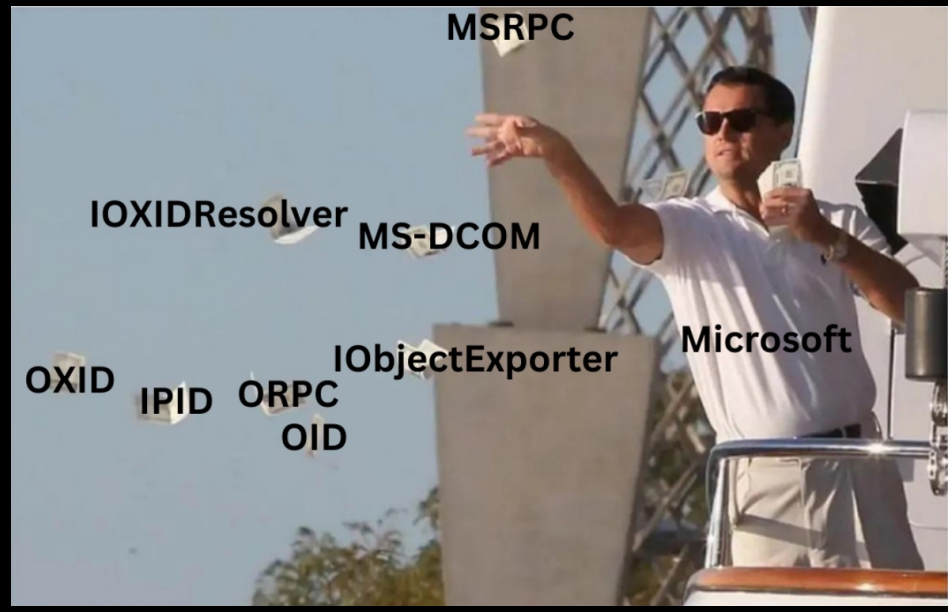
Research Roadmap: Brute-force OPNUMs

- Output:
  - rpc_x_bad_stub_data = it called successfully but with bad arguments
  - nca_s_op_rng_error = this function is not implemented
  - success = it's called successfully
  - rpc_access_denied = we don't have enough permission to call the function
- Bind and brute-force OPNUMs for each UUID collected from previous stages
- **rpc_x_bad_stub_data or success, it means that the methods within these interfaces are vulnerable to no authentication**

```
└─$ impacket-rpcmap ncacn_ip_tcp:192.168.177.177[135] -uuid '99fcfec4-5260-101b-bbcb-00aa0021347a v0.0' -brute-opnums -auth-level 1
Impacket v0.11.0 - Copyright 2023 Fortra

Opnum 0: rpc_x_bad_stub_data
Opnum 1: rpc_x_bad_stub_data
Opnum 2: rpc_x_bad_stub_data
Opnum 3: success
Opnum 4: rpc_x_bad_stub_data
Opnum 5: success
Opnums 6-64: nca_s_op_rng_error (opnum not found)
```

Research Roadmap:
Case Study (IObjectExporter interface)

- MSRPC vs DCOM
- IObjectExporter or IOXIDResolver [1] is the interface used for OXID resolution, pinging, and server aliveness tests
- Serveralive2 it is one method inside IObjectExporter interface and it’s used in DCOM creation process



[1] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-dcom/8ed0ae33-56a1-44b7-979f-5972f0e9416c

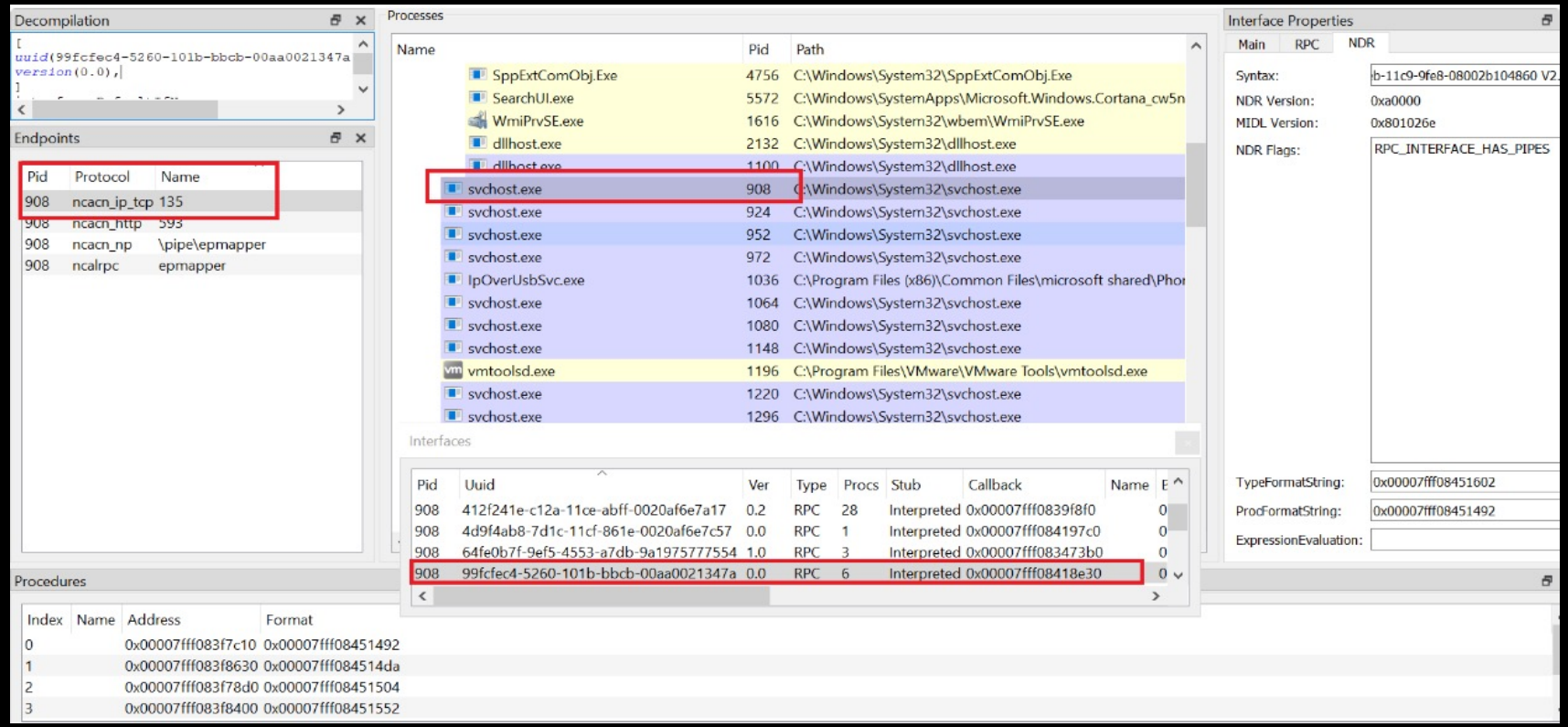
Research Roadmap:
Case Study (IObjectExporter-rpcmap)

- OPNUMs 0, 1, 2, and 4 fall under rpc_x_bad_stub_data
- OPNUMs ranging from 6 to 64 show a range error
- OPNUM 3 and OPNUM 5 return successfully

```
└─$ impacket-rpcmap ncacn_ip_tcp:192.168.177.177[135] -uuid '99fcfec4-5260-101b-bbcb-00aa0021347a v0.0' -brute-opnums -auth-level 1
Impacket v0.11.0 - Copyright 2023 Fortra

Opnum 0: rpc_x_bad_stub_data
Opnum 1: rpc_x_bad_stub_data
Opnum 2: rpc_x_bad_stub_data
Opnum 3: success
Opnum 4: rpc_x_bad_stub_data
Opnum 5: success
Opnums 6-64: nca_s_op_rng_error (opnum not found)
```

Research Roadmap: Case Study (RPCView)

- RPCView is a free and powerful tool to explore and decompile all RPC functionalities
- RPCView will be used to compare the results obtained from the rpcmappy script with the actual methods under the IObjectExporter interface on the remote host



https://www.rpcview.org/

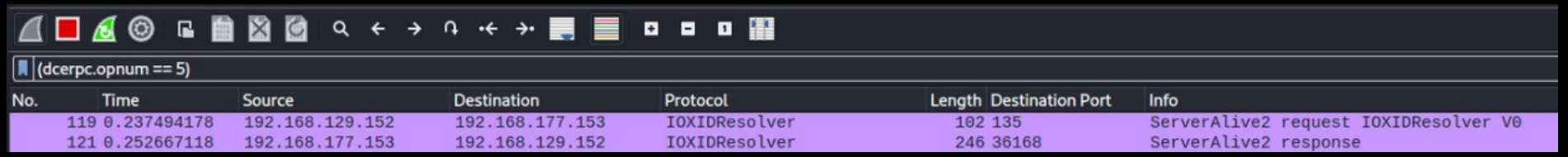
Research Roadmap: Case Study (RPCView)

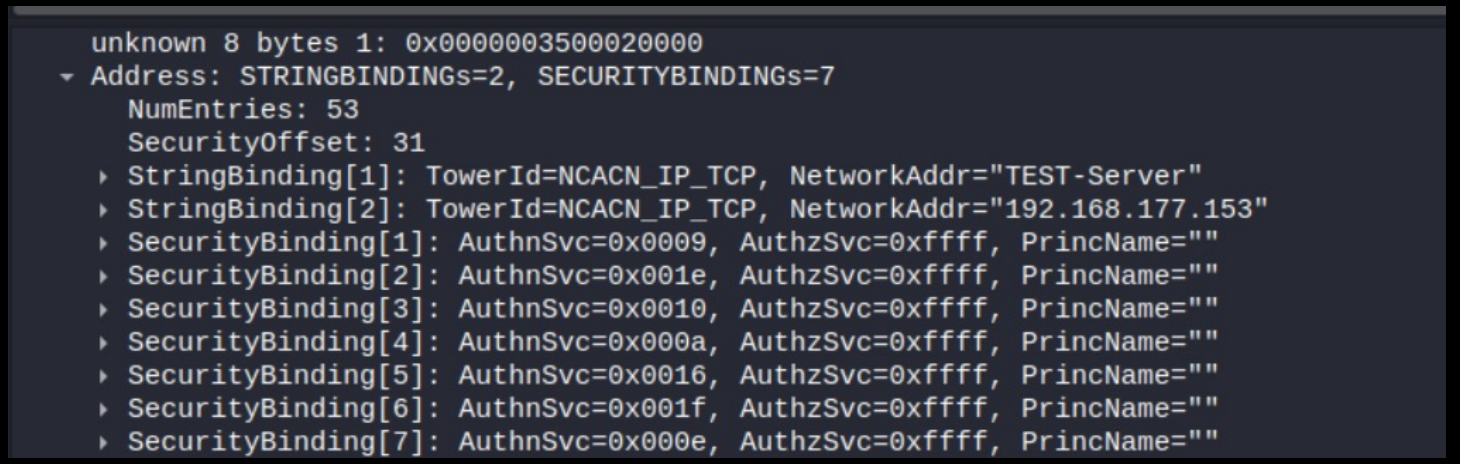
- RPCView, decompilation tab shows the declarations of each method (Proc) are displayed Methods
- Proc3 and Proc5 can be invoked without any arguments, unlike the other methods
- The similarity between Proc3 and Proc5 in the declarations aligns with the findings from rpcmap.py

```
49  error_status_t Proc3(
50  );
51
52  error_status_t Proc4(
53    [in]hyper arg_1,
54    [in][range(0,32768)] short arg_2,
55    [in][size_is(arg_2)]/[range(0,32768)]/ short arg_3[],
56    [out][ref]struct Struct_68_t** arg_4,
57    [out]struct Struct_88_t arg_5,
58    [out]long arg_6,
59    [out]struct Struct_174_t arg_7);
60
61  error_status_t Proc5(
62    [out]struct Struct_174_t* arg_2,
63    [out][ref]struct Struct_68_t** arg_3,
64    [out]long *arg_4);
65  }
```

Research Roadmap: Case Study (Wireshark)

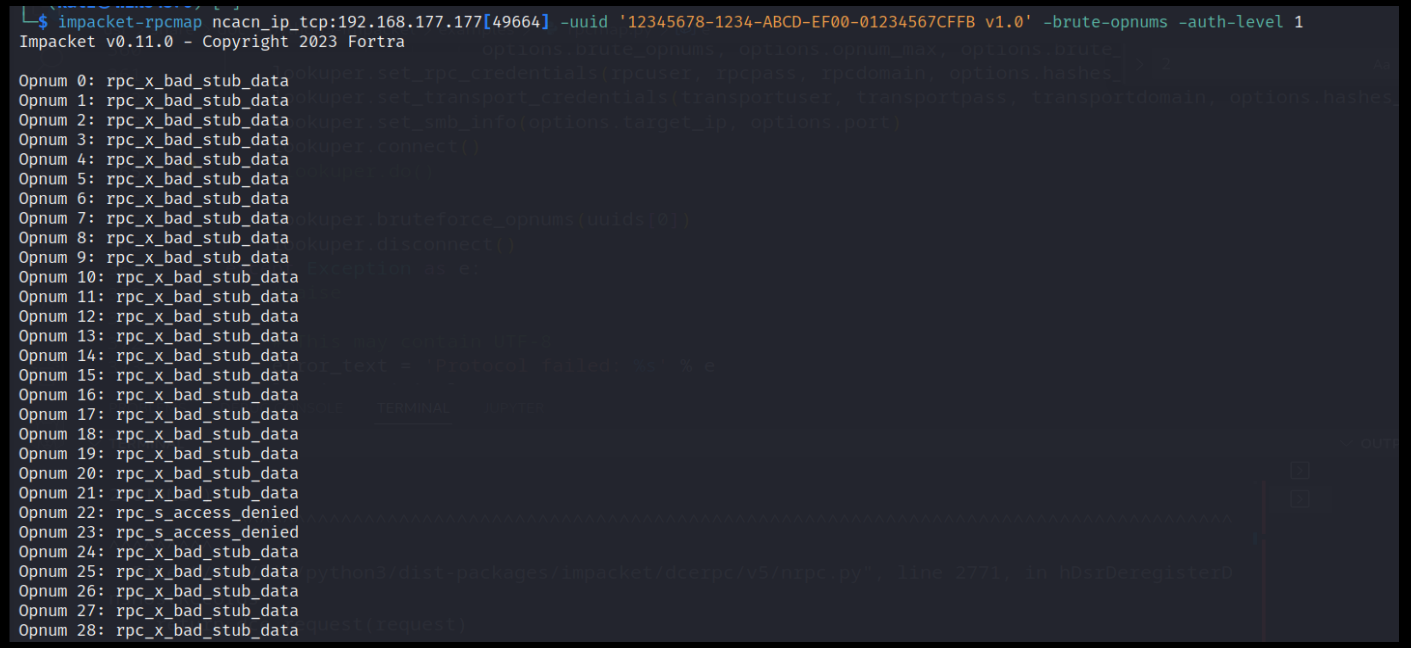
- We filtered for opnum5 (ServerAlive2)
- Wireshark successfully identifies the request and classifies it as ServerAlive2
- We can also observe the corresponding response from the server with all network interfaces





No Authentication Against MS-NRPC Protocol

- One of interfaces ”12345678-1234-ABCD-EF00-01234567CFFB v1.0“ acceded under TCP endpoint 49664
- The Protocol under this interface is named **MS-NRPC**
- Almost all OPNUMs from 0 to 49 shows rpc_x_bad_stub_data, means that successfully called but with bad arguments
- OPNUM from 49 to 59 is access dined and from 59 to 64 not implemented

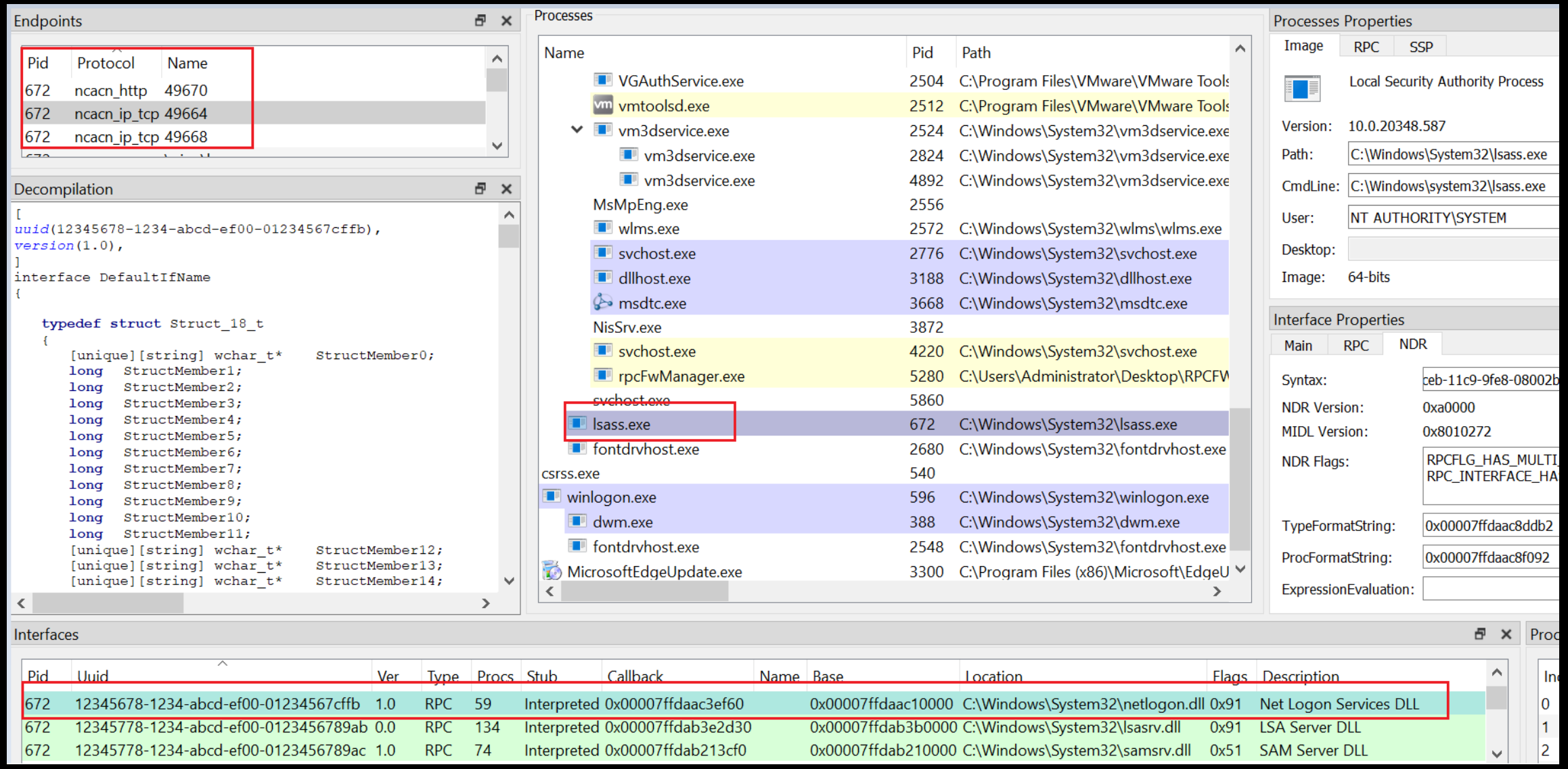


What Is MS-NRPC Protocol?

- The Netlogon Remote Protocol is a remote procedure call (RPC) [1] interface that is used for user and machine authentication on domain-based networks. The Netlogon Remote Protocol RPC interface is also used to replicate the database for backup domain controllers (BDCs)
- This protocol is often access from the \pipe\netlogon named pipe on IPC$ but in some cases, it can also be reached through a dynamically assigned TCP port

[1] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/ff8f970f-3e37-40f7-bd4b-af7336e4792f

RPCView Against MS-NRPC



Wireshark Against MS-NRPC

- Wireshark identifies the protocol under the name RPC_NETLOGON
- Wireshark can identify all the OPNUMS

11	0.001242492	192.168.177.111	192.168.177.177	DCERPC	138	49664	1 Bind: call_id: 1, Fragment: Single, 1 context items: RPC_NETLOGON V1.0 (32bit NDR)
12	0.001725563	192.168.177.177	192.168.177.111	DCERPC	126	54856	1 Bind_ack: call_id: 1, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptance
14	0.003567293	192.168.177.111	192.168.177.177	RPC_NETLOGON	90	49664	73 NetrLogonUasLogon request[Malformed Packet] RPC_NETLOGON V1
15	0.004109104	192.168.177.177	192.168.177.111	DCERPC	98	54856	61 Fault: call_id: 1, Fragment: Single, Ctx: 0, status: nca_s_fault_ndr
23	0.005649215	192.168.177.111	192.168.177.177	DCERPC	138	49664	1 Bind: call_id: 2, Fragment: Single, 1 context items: RPC_NETLOGON V1.0 (32bit NDR)
24	0.005875897	192.168.177.177	192.168.177.111	DCERPC	126	54862	1 Bind_ack: call_id: 2, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptance
26	0.007113244	192.168.177.111	192.168.177.177	RPC_NETLOGON	90	49664	73 NetrLogonUasLogoff request[Malformed Packet] RPC_NETLOGON V1
27	0.007883494	192.168.177.177	192.168.177.111	DCERPC	98	54862	61 Fault: call_id: 2, Fragment: Single, Ctx: 0, status: nca_s_fault_ndr
35	0.009843732	192.168.177.111	192.168.177.177	DCERPC	138	49664	1 Bind: call_id: 3, Fragment: Single, 1 context items: RPC_NETLOGON V1.0 (32bit NDR)
36	0.010193248	192.168.177.177	192.168.177.111	DCERPC	126	54866	1 Bind_ack: call_id: 3, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptance
38	0.012252409	192.168.177.111	192.168.177.177	RPC_NETLOGON	90	49664	73 NetrLogonSamLogon request[Malformed Packet] RPC_NETLOGON V1
39	0.012954758	192.168.177.177	192.168.177.111	DCERPC	98	54866	61 Fault: call_id: 3, Fragment: Single, Ctx: 0, status: nca_s_fault_ndr
47	0.015973649	192.168.177.111	192.168.177.177	DCERPC	138	49664	1 Bind: call_id: 4, Fragment: Single, 1 context items: RPC_NETLOGON V1.0 (32bit NDR)
48	0.016520337	192.168.177.177	192.168.177.111	DCERPC	126	54882	1 Bind_ack: call_id: 4, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptance
50	0.020609240	192.168.177.111	192.168.177.177	RPC_NETLOGON	90	49664	73 NetrLogonSamLogoff request[Malformed Packet] RPC_NETLOGON V1
51	0.022273117	192.168.177.177	192.168.177.111	DCERPC	98	54882	61 Fault: call_id: 4, Fragment: Single, Ctx: 0, status: nca_s_fault_ndr
59	0.023998195	192.168.177.111	192.168.177.177	DCERPC	138	49664	1 Bind: call_id: 5, Fragment: Single, 1 context items: RPC_NETLOGON V1.0 (32bit NDR)
60	0.024469233	192.168.177.177	192.168.177.111	DCERPC	126	54892	1 Bind_ack: call_id: 5, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptance
62	0.026689505	192.168.177.111	192.168.177.177	RPC_NETLOGON	90	49664	73 NetrServerReqChallenge request[Malformed Packet] RPC_NETLOGON V1
63	0.027699311	192.168.177.177	192.168.177.111	DCERPC	98	54892	61 Fault: call_id: 5, Fragment: Single, Ctx: 0, status: nca_s_fault_ndr
71	0.029299561	192.168.177.111	192.168.177.177	DCERPC	138	49664	1 Bind: call_id: 6, Fragment: Single, 1 context items: RPC_NETLOGON V1.0 (32bit NDR)
72	0.029740200	192.168.177.177	192.168.177.111	DCERPC	126	54894	1 Bind_ack: call_id: 6, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptance
74	0.030710592	192.168.177.111	192.168.177.177	RPC_NETLOGON	90	49664	73 NetrServerAuthenticate request[Malformed Packet] RPC_NETLOGON V1
75	0.031768731	192.168.177.177	192.168.177.111	DCERPC	98	54894	61 Fault: call_id: 6, Fragment: Single, Ctx: 0, status: nca_s_fault_ndr
83	0.035521267	192.168.177.111	192.168.177.177	DCERPC	138	49664	1 Bind: call_id: 7, Fragment: Single, 1 context items: RPC_NETLOGON V1.0 (32bit NDR)

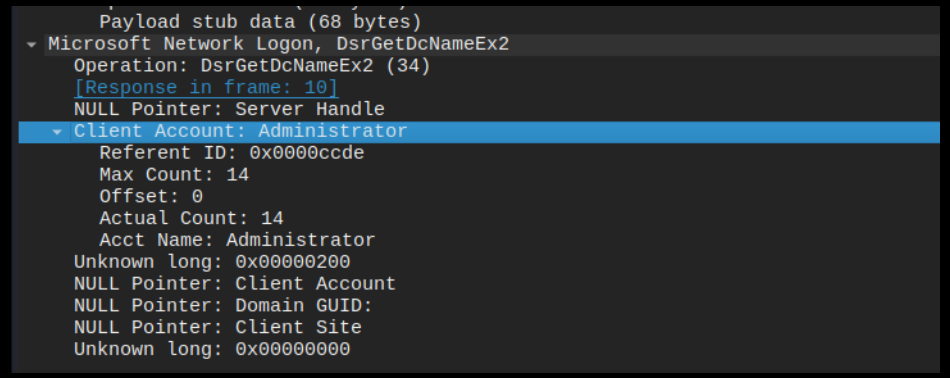
MS-NRPC Functions

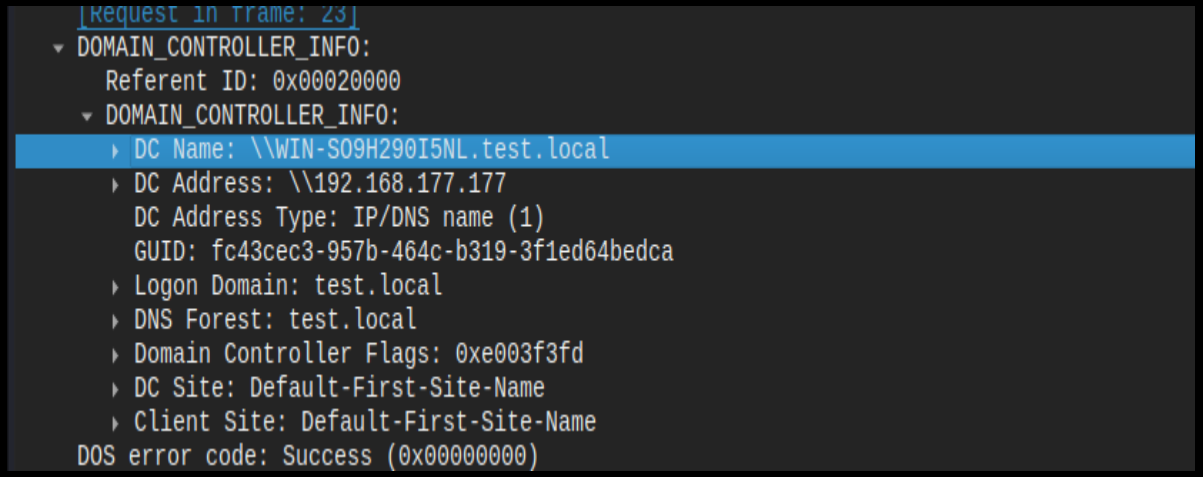
- Impacket implements MS-NRPC protocol under nrpc.py
- At this stage I've started to call each function and check what info we can get
- There are many functions called successfully

OPCODE	FUNCTION
10	NetrGetDCName
11	NetrGetAnyDCName
19	NetrEnumerateTrustedDomains
27	DsrGetDcNameEx
28	DsrGetSiteName
33	DsrAddressToSiteNamesW
34	DsrGetDcNameEx2
36	NetrEnumerateTrustedDomainsEx
38	DsrGetDcSiteCoverageW
43	DsrGetForestTrustInformation
44	NetrGetForestTrustInformation

DsrGetDcNameEx2 function

- The **DsrGetDcNameEx2 [1]** method SHOULD return information about a domain controller (DC) in the specified domain and site and checks about user accounts
- After some googling and under this article from orange cyberdefence [2], They used this function to enumerate domain users (bruteforce) but through named pipe (**they assumed null session is enabled**); it can be closed and monitored
- If the user existed the below information will be returned from the DC
- If the user doesn't exist the error will be returned





[1] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/fb8e1146-a045-4c31-98d1-c68507ad5620

[2] https://sensepost.com/blog/2018/a-new-look-at-null-sessions-and-user-enumeration/

NetrEnumerateTrustedDomainsEx Function

- The NetrEnumerateTrustedDomainsEx [1] return a list of trusted domains from a specified serve, this method extends NetrEnumerateTrustedDomains by returning an array of domains in a more flexible DS_DOMAIN_TRUSTSW structure

```
Entries: 1
▾ DS_DOMAIN_TRUSTS_ARRAY:
    Referent ID: 0x00020000
    Max Count: 1
  ▾ DS_DOMAIN_TRUSTS
    ▾ NetBIOS Name: TEST
        Referent ID: 0x00020004
        Max Count: 5
        Offset: 0
        Actual Count: 5
        Downlevel Domain: TEST
    ▾ DNS Domain Name: test.local
        Referent ID: 0x00020008
        Max Count: 11
        Offset: 0
        Actual Count: 11
        DNS Domain: test.local
    ▸ Trust Flags: 0x0000001d
      Parent Index: 0x00000000
      Trust Type: AD Domain (2)
    ▸ Trust Attributes: 0x00000000
    ▾ SID pointer:
      ▾ SID pointer
          Referent ID: 0x0002000c
          Count: 4
        ▸ Domain SID: S-1-5-21-789115489-1348537132-2098222337  (Domain SID)
      GUID: fc43cec3-957b-464c-b319-3f1ed64bedca
Return code: STATUS_SUCCESS (0x00000000)
```

[1] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-nrpc/c3e9870a-0943-4d45-be94-edb9419a1c11

NAuthNRPC Tool


```
└─$ python3 nauth.py -t 192.168.177.177 -u users.txt -c computers.txt

NAuthNRPC Tool By Haidar Kabibo - Kaspersky Security Services 2024

[*] Domain Information
------------------------
[*] DC Name: WIN-SO9H290I5NL.test.local
[*] DC IP: 192.168.177.177
[*] Domain GUID: FC43CEC3-957B-464C-B319-3F1ED64BEDCA
[*] Domain Name: test.local
[*] Forest Name: test.local
[*] DC Site Name: Default-First-Site-Name
[*] Client Site Name: Default-First-Site-Name
[*] Domain Flags: DS_PDC_FLAG | DS_GC_FLAG | DS_LDAP_FLAG | DS_DS_FLAG | DS_KDC_FLAG | DS_TIMESERV_FLAG | DS_CLOSEST_FLAG | DS_WRITABLE_FLAG | DS_GOOD_TIMESERV_FLAG | DS_FULL_SECRET_DOMAIN_6_FLAG | DS_WS_FLAG | DS_DS_8_FLAG | DS_DS_
AG | DS_DS_10_FLAG | DS_KEY_LIST_FLAG | DS_PING_FLAGS | DS_DNS_CONTROLLER_FLAG | DS_DNS_DOMAIN_FLAG | DS_DNS_FOREST_FLAG

[*] Trusted Domains Information
-------------------------------
[*] Trusted Domain number 0
    • NetBios Domain Name: test.main.trust
    • DNS Domain Name: Not Available
    • Flags: DS_DOMAIN_DIRECT_OUTBOUND
    • Parent Index: Not Available
    • Trust Type: TRUST_TYPE_MIT
    • Trust Attributes: TRUST_ATTRIBUTE_NON_TRANSITIVE
    • Domian SID: Not Available
    • Domain GUID: 00000000-0000-0000-0000-000000000000
[*] Trusted Domain number 1
    • NetBios Domain Name: TEST
    • DNS Domain Name: test.local
    • Flags: DS_DOMAIN_IN_FOREST | DS_DOMAIN_TREE_ROOT | DS_DOMAIN_PRIMARY | DS_DOMAIN_NATIVE_MODE
    • Parent Index: Not Available
    • Trust Type: TRUST_TYPE_UPLEVEL
    • Trust Attributes: Not Available
    • Domian SID: s-1-5-21-789115489-1348537132-2098222337
    • Domain GUID: FC43CEC3-957B-464C-B319-3F1ED64BEDCA

[*] User Accounts Enumeration
-----------------------------
[-] user SMITH is not existed
[-] user JOHNSON is not existed
[-] user WILLIAMS is not existed
[+] user Administrator is existed

[*] Computer Accounts Enumeration
---------------------------------
[+] computer account Admin-PC$ is existed
[-] computer account fuckit-asdf$ is not existed
```

https://github.com/klsecservices/NauthNRPC

Metasploit

rapid7.com/blog/post/2024/06/28/metasploit-weekly-wrap-up-06-28-2024/

MS-NRPC Domain Users Enumeration

Author: Haidar Kabibo https://x.com/haider_kabibo

Type: Auxiliary

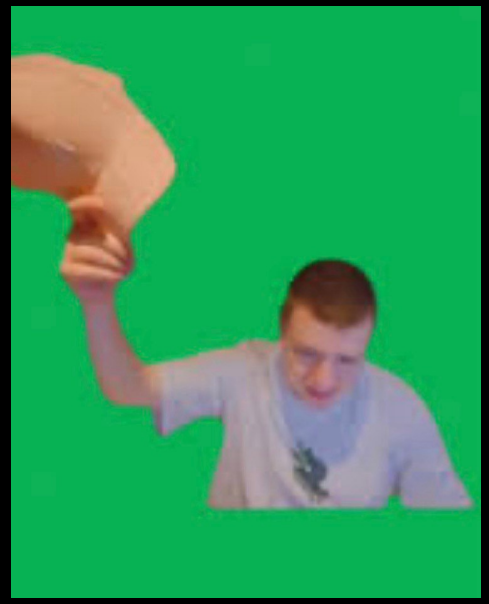
Pull request: #19205 contributed by sud0Ru

Path: `scanner/dcerpc/nrpc_enumusers`

Description: This adds a new module that can enumerate accounts on a target Active Directory Domain Controller without authenticating to it; instead the module does so by issuing a DCERPC request and analyzing the returned error status.

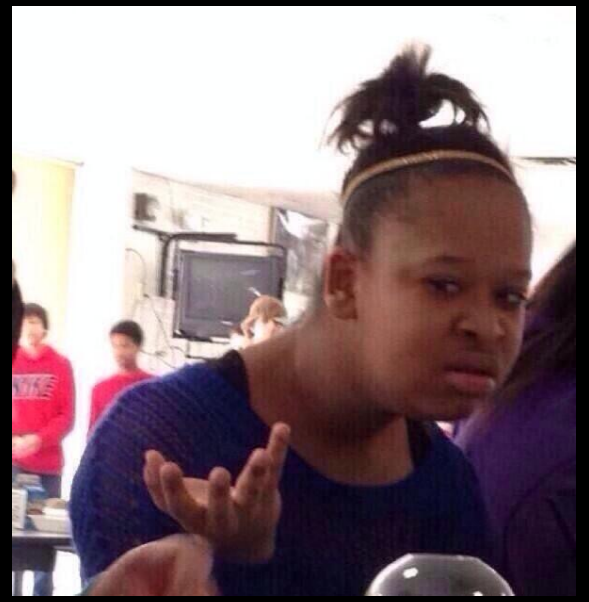
```
[*] Running module against 192.168.177.177

[*] 192.168.177.177: - Connecting to the endpoint mapper service...
[*] 192.168.177.177: - Binding to 12345678-1234-abcd-ef00-01234567cffb:1.0@ncacn_ip_tcp:192.168.177.177[49664]...
[-] 192.168.177.177: - Tiffany.Molina Not exist
[-] 192.168.177.177: - SMITH Not exist
[-] 192.168.177.177: - JOHNSON Not exist
[-] 192.168.177.177: - WILLIAMS Not exist
[-] 192.168.177.177: - Administratorsvc_ldap Not exist
[-] 192.168.177.177: - svc_ldap Not exist
[-] 192.168.177.177: - ksimpson Not exist
[+] 192.168.177.177: - Administrator Exist
[-] 192.168.177.177: - James Not exist
[-] 192.168.177.177: - nikk37 Not exist
[-] 192.168.177.177: - svc-printer Not exist
[-] 192.168.177.177: - SABatchJobs Not exist
[-] 192.168.177.177: - e.black Not exist
[-] 192.168.177.177: - Kaorz Not exist
```

How Can You Prevent It?



The Group Policy That Punches Your Domain In The Face

- To stop these kind of activates you can set “Restrictions for Unauthenticated RPC Clients” to Authenticated without exceptions
- In this case AD service will not work probably as Microsoft mentioned [1]
- lets get WMI as example



[1] https://techcommunity.microsoft.com/t5/ask-the-directory-services-team/restrictions-for-unauthenticated-rpc-clients-the-group-policy/ba-p/399128

WMI As A DCOM Object

- Remote WMI access relies on DCOM architecture
- To interact with WMI server a DCOM object should be created in the remote system
- To create a DCOM object (through windows libraries), serveralive2() function should be called under IObjectExporte RPC interface
- This interface is bind through auth-level=1 as we saw before
- If the policy Authenticated without exceptions is enabled this stage will fail and the whole creation process of DCOM object will fail

```
PS C:\Users\Administrator.DESKTOP-AJUMAE9> wmic /node:192.168.177.177 /user:.\Administrator /password:Asd123456# process list brief
Node - 192.168.177.177
ERROR:
Description = Access is denied.
```

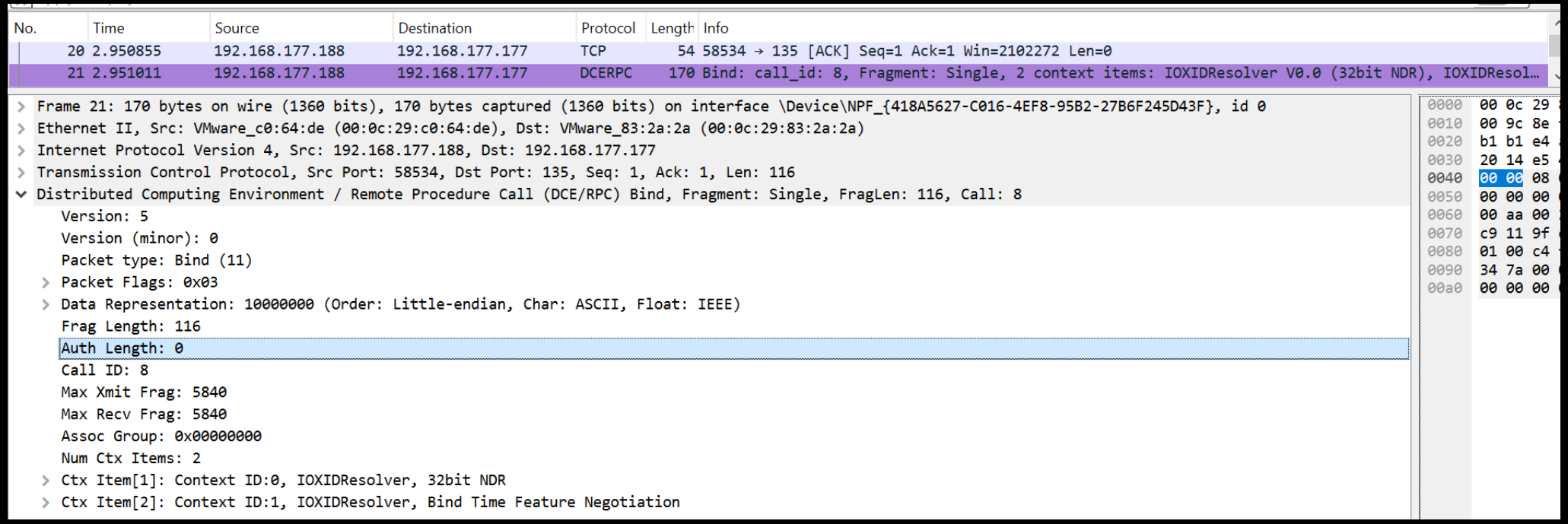
WMI As A DCOM Object

- Lets see the network traffic when we make WMI query through powershell
- The RPC policy is “Authenticated” or “None”
- We have two interfaces will be bound to create DCOM object: IOXIDResolver and ISystemActivator

No.	Time	Source	Destination	Protocol	Length	Info
21	2.951011	192.168.177.188	192.168.177.177	DCERPC	170	Bind: call_id: 8, Fragment: Single, 2 context items: IOXIDResolver V0.0 (32bit NDR), IOXIDResol…
22	2.951662	192.168.177.177	192.168.177.188	DCERPC	138	Bind_ack: call_id: 8, Fragment: Single, max_xmit: 5840 max_recv: 5840, 2 results: Acceptance, N…
23	2.951754	192.168.177.188	192.168.177.177	IOXIDR…	78	ServerAlive2 request IOXIDResolver V0
24	2.952716	192.168.177.177	192.168.177.188	IOXIDR…	218	ServerAlive2 response[Long frame (2 bytes)]
28	2.955104	192.168.177.188	192.168.177.177	DCERPC	174	Bind: call_id: 9, Fragment: Single, 1 context items: ISystemActivator V0.0 (32bit NDR), NTLMSSP…
29	2.955846	192.168.177.177	192.168.177.188	DCERPC	352	Bind_ack: call_id: 9, Fragment: Single, max_xmit: 5840 max_recv: 5840, 1 results: Acceptance, N…
30	2.957195	192.168.177.188	192.168.177.177	DCERPC	608	AUTH3: call_id: 9, Fragment: Single, NTLMSSP_AUTH, User: .\Administrator
31	2.957388	192.168.177.188	192.168.177.177	ISyste…	966	RemoteCreateInstance request
33	2.960938	192.168.177.177	192.168.177.188	ISyste…	1046	RemoteCreateInstance response
37	2.972944	192.168.177.188	192.168.177.177	DCERPC	262	Bind: call_id: 2, Fragment: Single, 3 context items: IRemUnknown2 V0.0 (32bit NDR), IRemUnknown…
38	2.974089	192.168.177.177	192.168.177.188	DCERPC	400	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: 5840, 3 results: Acceptance, P…
39	2.974724	192.168.177.188	192.168.177.177	DCERPC	628	AUTH3: call_id: 2, Fragment: Single, NTLMSSP_AUTH, User: .\Administrator
40	2.974832	192.168.177.188	192.168.177.177	IRemUn…	246	RemQueryInterface request IID[1]=IWbemLoginClientID
42	2.977535	192.168.177.177	192.168.177.188	IRemUn…	182	RemQueryInterface response S_OK[1] -> S_OK

WMI As A DCOM Object

- When the creation of DCOM object done by using native library, the binding of IOXIDResolver interface will be done without authentication (auth level = 1)



WMI As A DCOM Object

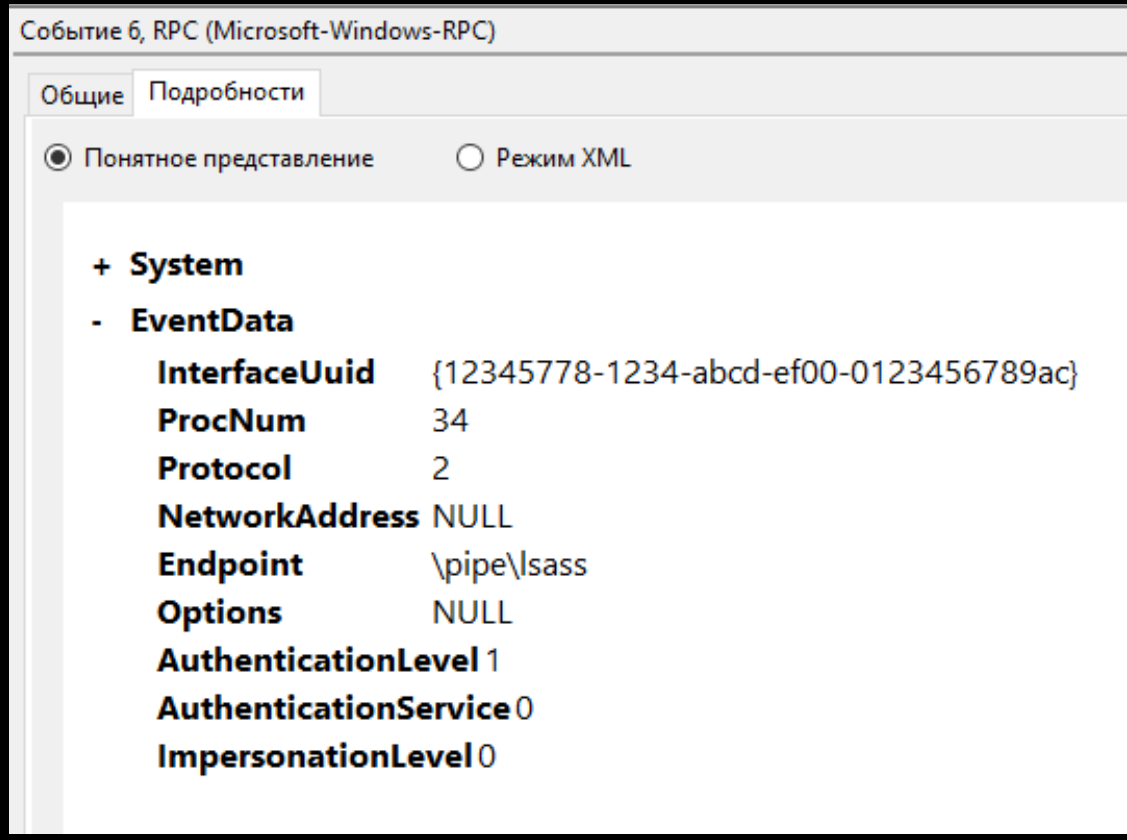
- The traffic below will be generated when we set the RPC policy to “Authenticated without exceptions” after we made WMI query through powershell
- As we see because the no authentication is stopped we will get access denied
- As results the WMI service wont work and even any creation of DCOM process through native windows libraries

No.	Time	Source	Destination	Protocol	Length	Info
25	0.016714	192.168.177.188	192.168.177.177	DCERPC	170	Bind: call_id: 2, Fragment: Single, 2 context items: IOXIDResolver V0.0 (32bit NDR), IOXIDResolver…
26	0.017353	192.168.177.177	192.168.177.188	DCERPC	138	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: 5840, 2 results: Acceptance, Nego…
27	0.017510	192.168.177.188	192.168.177.177	IOXIDR…	78	ServerAlive2 request IOXIDResolver V0
28	0.017915	192.168.177.177	192.168.177.188	DCERPC	86	Fault: call_id: 2, Fragment: Single, Ctx: 0, status: nca_s_fault_access_denied
50	0.045525	192.168.177.188	192.168.177.177	DCERPC	170	Bind: call_id: 2, Fragment: Single, 2 context items: IOXIDResolver V0.0 (32bit NDR), IOXIDResolver…
51	0.045745	192.168.177.177	192.168.177.188	DCERPC	138	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: 5840, 2 results: Acceptance, Nego…
52	0.045876	192.168.177.188	192.168.177.177	IOXIDR…	78	ServerAlive2 request IOXIDResolver V0
53	0.046102	192.168.177.177	192.168.177.188	DCERPC	86	Fault: call_id: 2, Fragment: Single, Ctx: 0, status: nca_s_fault_access_denied

How Can You Detect It?

The Event That Never Occurs (Native Protection)

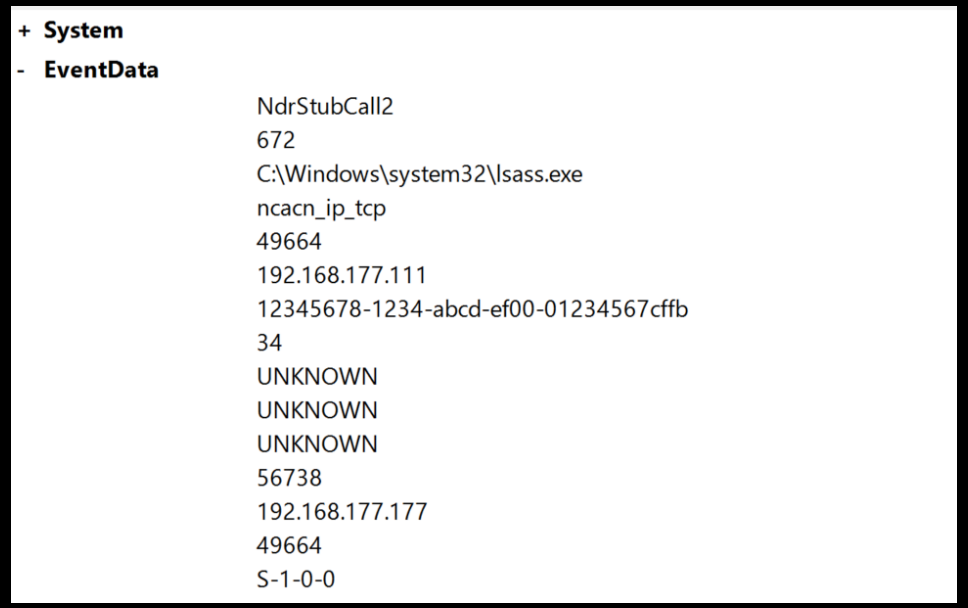
- Audit RPC Events 5712(S): A Remote Procedure Call (RPC) was attempted [2]
- According to Microsoft this event never occurs
- So what is the solution here?
  1- you can use Event Tracing for Windows (ETW) info but it's lack to information and it contains a lots of events because it contains local rpc calls
  2- you can use third party software called rpcfirewall



[1] https://habr.com/ru/companies/rvision/articles/716656/
[2] https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-5712

RPC-Firewall

- Can be used to audit all remote RPC calls. Once executing any remote attack tools, you will see which RPC UUIDs and OPNUMs were called remotely [1][2]
- The RPC Firewall allows to be more granular about the specific OPNUMs we wish to block and the source addresses from which we allow RPC calls
- It can be integrated with event viewer and show you logs



[1] https://github.com/zeronetworks/rpcfirewall
[2] https://www.youtube.com/watch?v=hz_YPIMeBMI&ab_channel=BlackHat

Conclusion for this part

- These interfaces are used by many windows services so it’s hard to distinguished between legitimate and illegitimate actions
- The whole DC infrastructure should be monitored to check what services use these interfaces and in what intensity
- After making some statistics we can put some alerts depending on intensity and source addresses

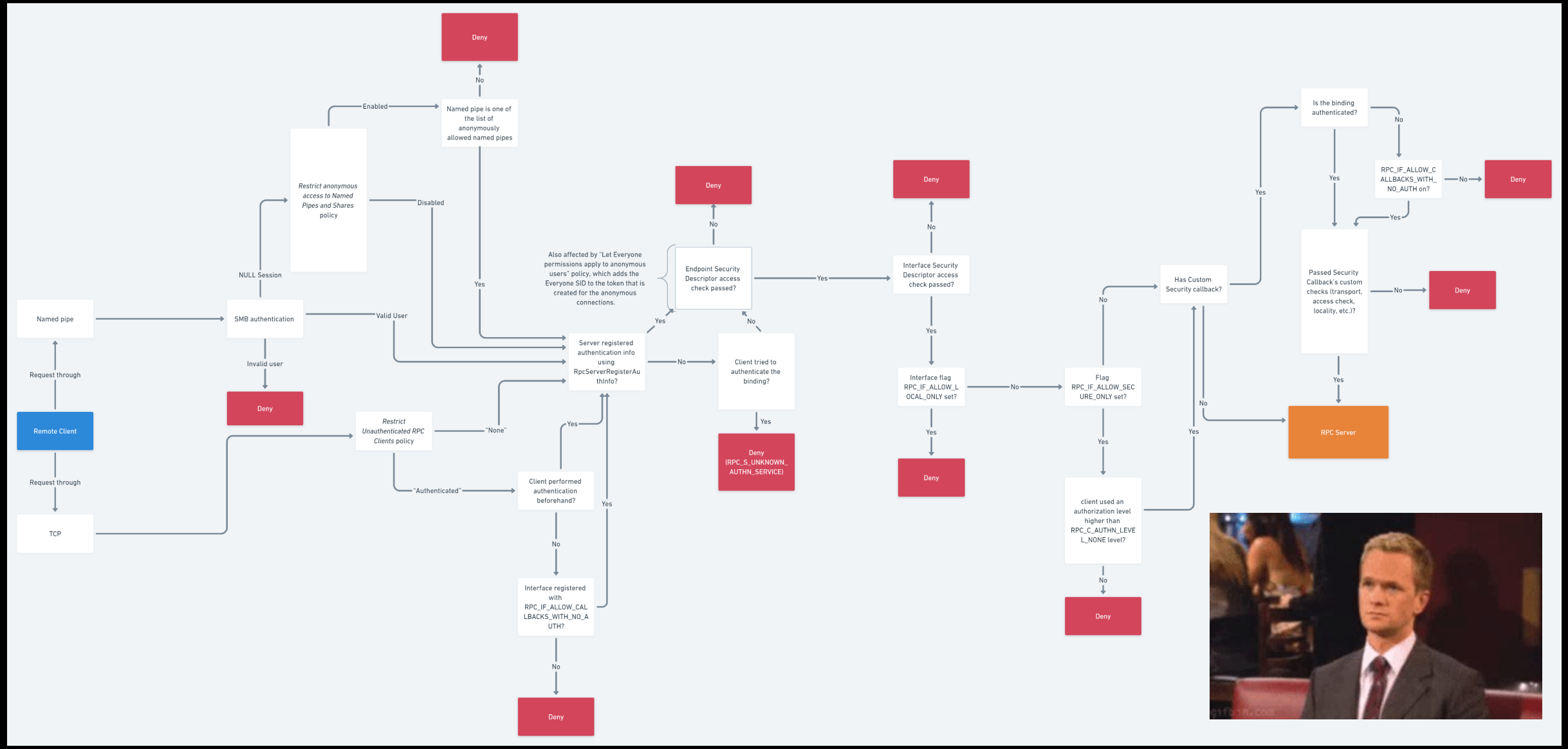
https://securelist.com/no-auth-domain-information-enumeration/112629/

MSRPC Security

- I'm sorry …

https://www.tiraniddo.dev/2021/08/how-to-secure-windows-rpc-server-and.html

MSRPC Security



https://github.com/akamai/akamai-security-research/tree/main/rpc_toolkit

MSRPC Security
1. Registration flags

- When you start your RPC server, you can specify certain flags within RpcServerRegisterIf2

Three interesting flags:

- RPC_IF_ALLOW_LOCAL_ONLY
- RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH
- RPC_IF_ALLOW_SECURE_ONLY

```
RPC_STATUS RpcServerRegisterIf2(
  RPC_IF_HANDLE      IfSpec,
  UUID               *MgrTypeUuid,
  RPC_MGR_EPV        *MgrEpv,
  unsigned int       Flags,
  unsigned int       MaxCalls,
  unsigned int       MaxRpcSize,
  RPC_IF_CALLBACK_FN *IfCallbackFn
);
```

https://learn.microsoft.com/en-us/windows/win32/api/rpcdce/nf-rpcdce-rpcserverregisterif2

MSRPC Security
1. Registration flags (Security Callback)

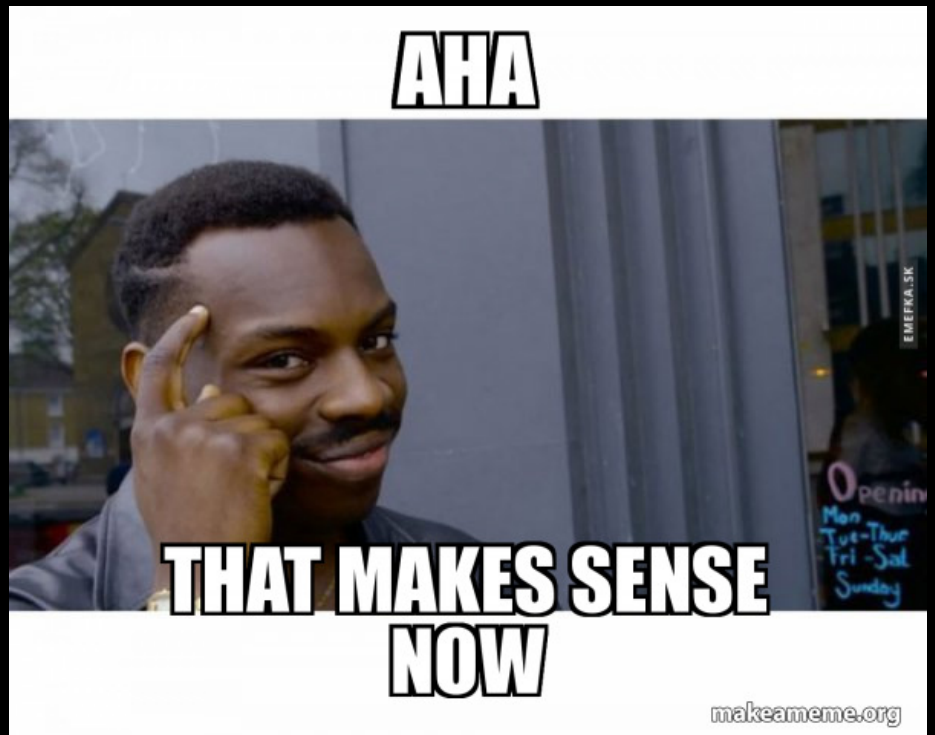
- RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH flag
- The callback function returns *RPC_S_OK* then the call will be allowed, anything else will deny the call
- It handle the security check instead of RPC runtime

```
RPC_STATUS CALLBACK SecurityCallback(RPC_IF_HANDLE Interface, void* pBindingHandle){
    printf("The security callback is called");
    return RPC_S_OK; // Whoever binds to the interface, we will allow the connection
}
```

```
status = RpcServerRegisterIf2(
    Example_v1_0_s_ifspec,
    NULL,
    NULL,
    RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH,
    RPC_C_LISTEN_MAX_CALLS_DEFAULT,
    (unsigned)-1,
    MySecutiyCallback);
```

MSRPC Security
1. Registration flags (The Policy)
The Exceptions

- “Restrict Unauthenticated RPC Clients policy“
- Using RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH with security callback the policy won't be affective
- It will move the authentication handling to the application itself
- Using "Authenticated without Exception“ will block unauthenticated clients regarding of the flag



MSRPC Security
2. Securing the endpoint (for non TCP)

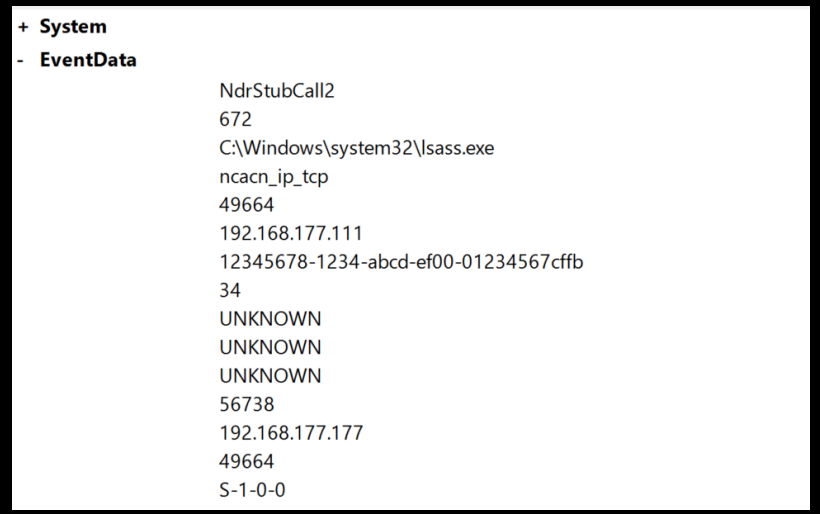
- RpcServerUseProtseqEp is used to register the endpoint
- The final parameter is optional and it used security descriptor (SD) you assign to the endpoint (for named pipes and ALPC)

```
RPC_STATUS RpcServerUseProtseqEp(
  RPC_CSTR     Protseq,
  unsigned int MaxCalls,
  RPC_CSTR     Endpoint,
  void         *SecurityDescriptor
);
```

https://learn.microsoft.com/en-us/windows/win32/api/rpcdce/nf-rpcdce-rpcserveruseprotseqep

MSRPC Security
3. Securing the interface

- RpcServerRegisterIf3 can be use to set optional SecurityDescriptor to the interface
- What will the token be if there is no authentication?




```
RPC_STATUS RpcServerRegisterIf3(
  [in]           RPC_IF_HANDLE      IfSpec,
  [in, optional] UUID               *MgrTypeUuid,
  [in, optional] RPC_MGR_EPV        *MgrEpv,
  [in]           unsigned int       Flags,
  [in]           unsigned int       MaxCalls,
  [in]           unsigned int       MaxRpcSize,
  [in, optional] RPC_IF_CALLBACK_FN *IfCallback,
  [in, optional] void               *SecurityDescriptor
);
```

https://www.tiraniddo.dev/2021/08/how-to-secure-windows-rpc-server-and.html

MSRPC Security
3. Securing the interface (No Token)

- Reversing rpcrt4 DLL which is RPC runtime library
- No auth = anonymous token (access check)

```
void *sub_7FFBB49A8024()
{
  HANDLE CurrentThread; // rax
  HANDLE v1; // rax
  void *TokenHandle; // [rsp+30h] [rbp+8h] BYREF

  TokenHandle = 0i64;
  CurrentThread = GetCurrentThread();
  if ( ImpersonateAnonymousToken(CurrentThread) )
  {
    v1 = GetCurrentThread();
    OpenThreadToken(v1, 8u, 0, &TokenHandle);
    RevertToSelf();
  }
  return TokenHandle;
}
```

```
      CurrentThread = GetCurrentThread();
      if ( OpenThreadToken(CurrentThread, 8u, 1, &ClientToken) )
      {
LABEL_10:
        v24 = *(_OWORD *)(a1 + 84);
        sub_7FFBB4988A90((unsigned int *)&v24, SourceString);
        LODWORD(v22) = -1395763957;
        RtlInitUnicodeString(&DestinationString, L"RPC Interface");
        RtlInitUnicodeString(&v26, SourceString);
        *((_QWORD *)&v22 + 1) = &DestinationString;
        v23 = &v26;
        ArbitraryUserPointer = NtCurrentTeb()->NtTib.ArbitraryUserPointer;
        NtCurrentTeb()->NtTib.ArbitraryUserPointer = &v22;
        for ( i = AccessCheck(
                    v2,
                    ClientToken,
                    0x2000000u,
                    &GenericMapping,
                    &PrivilegeSet,
                    &PrivilegeSetLength,
                    &GrantedAccess,
                    &AccessStatus);
```

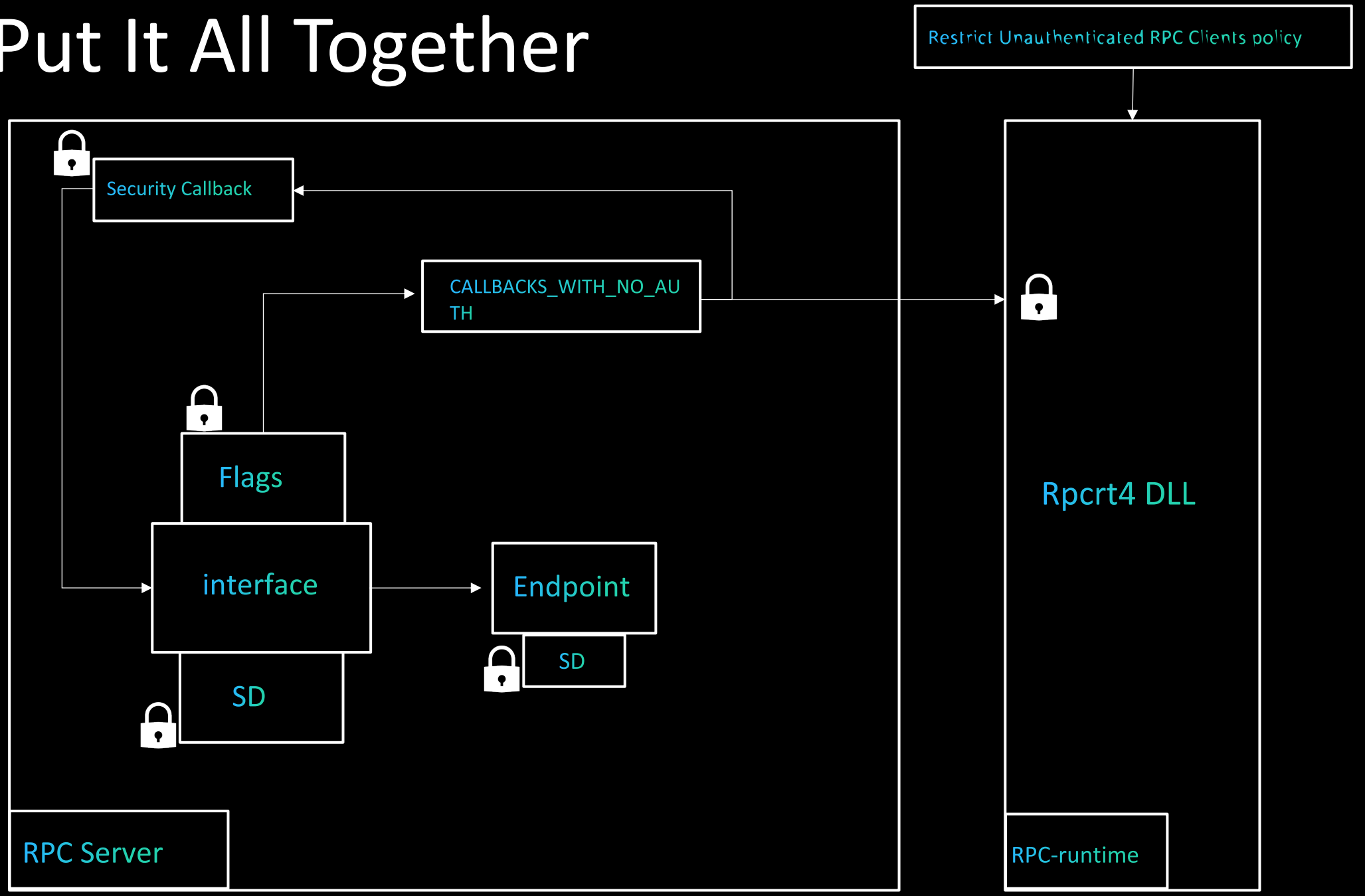
MSRPC Security
4. Binding Authentication


```
status = RpcServerRegisterAuthInfo(
     pszSpn,              // Server principal name
     RPC_C_AUTHN_WINNT,   // using NTLM as authentication service provider
     NULL,                // Use default key function, which is ignored for NTLM SSP
     NULL                 // No arg for key function
 );
```

```
   status = RpcBindingSetAuthInfoEx(
    ImplicitHandle,     // the client's binding handle
    NULL,           // the server's service principale name (SPN)
    RPC_C_AUTHN_LEVEL_NONE, // no authentication
    RPC_C_AUTHN_WINNT,      // using NTLM as authentication service provider
    NULL,           // use current thread credentials
    0,      // authorization based on the provided SPN
    &SecurityQOS            // Quality of Service structure
);
```

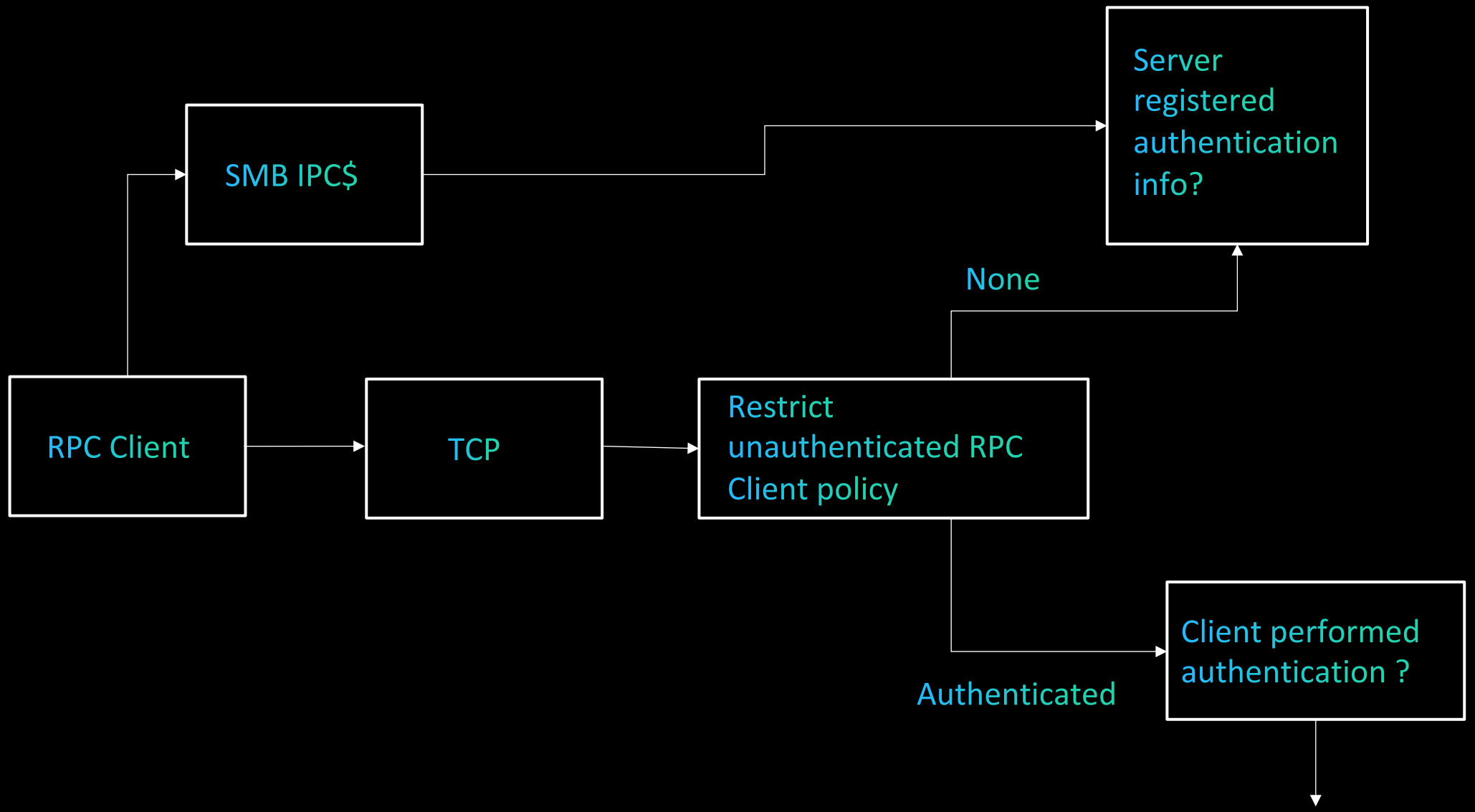
https://learn.microsoft.com/en-us/windows/win32/api/rpcdce/nf-rpcdce-rpcserverregisterauthinfo

Put It All Together

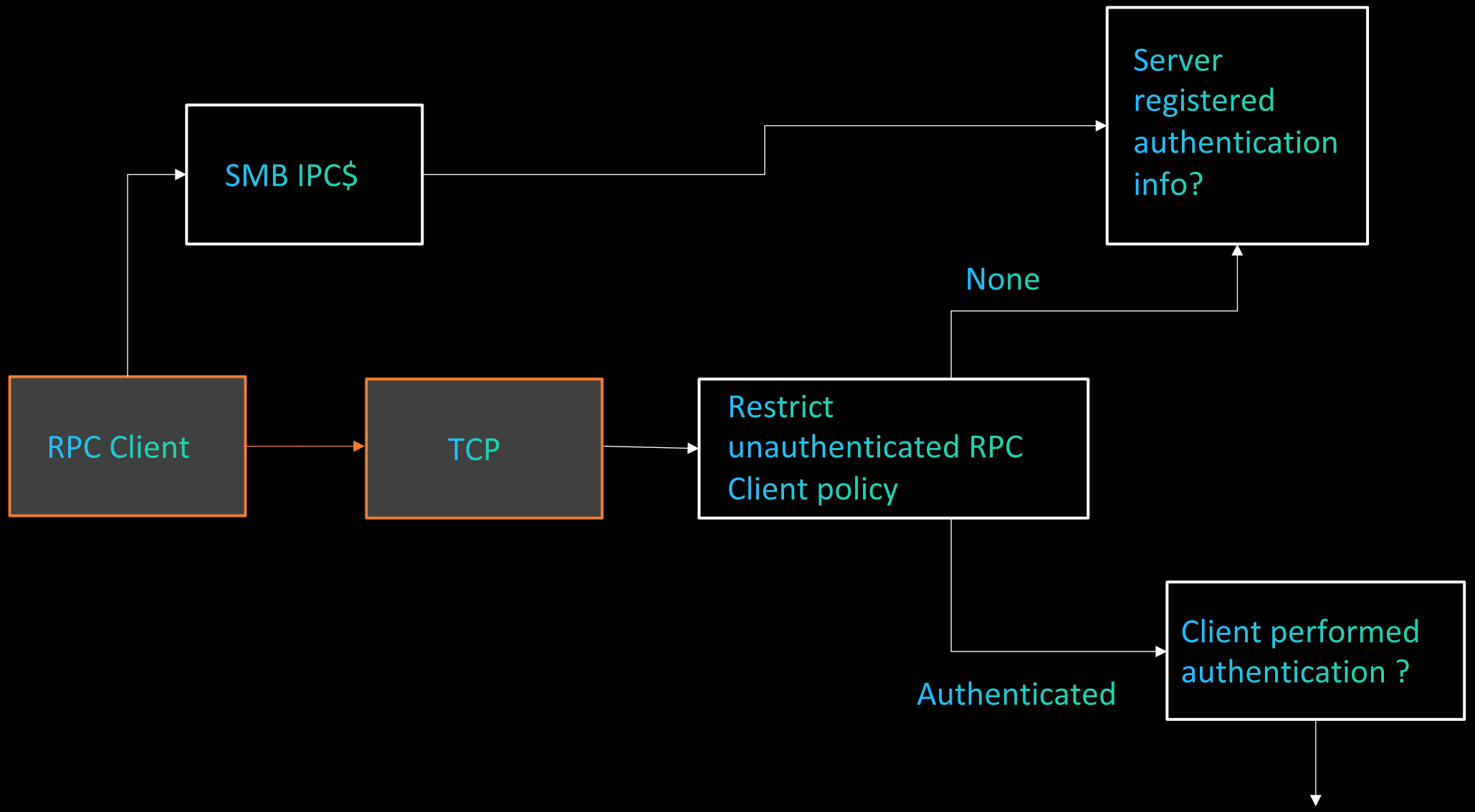


MS-NRPC Security
A. Surface Analysis

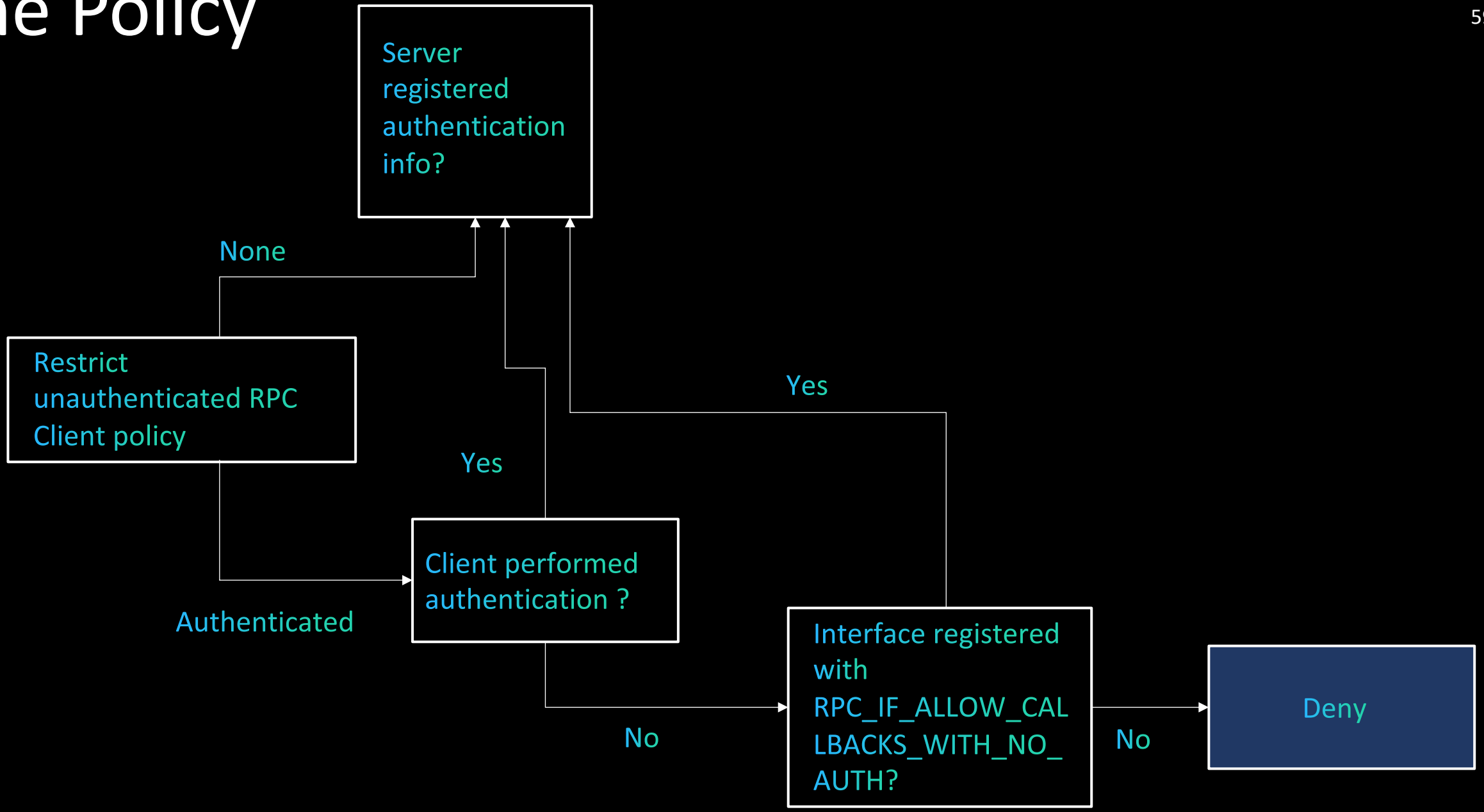
Transport


Server registered authentication info?
SMB IPC$
None
RPC Client
TCP
Restrict unauthenticated RPC Client policy
Client performed authentication ?
Authenticated

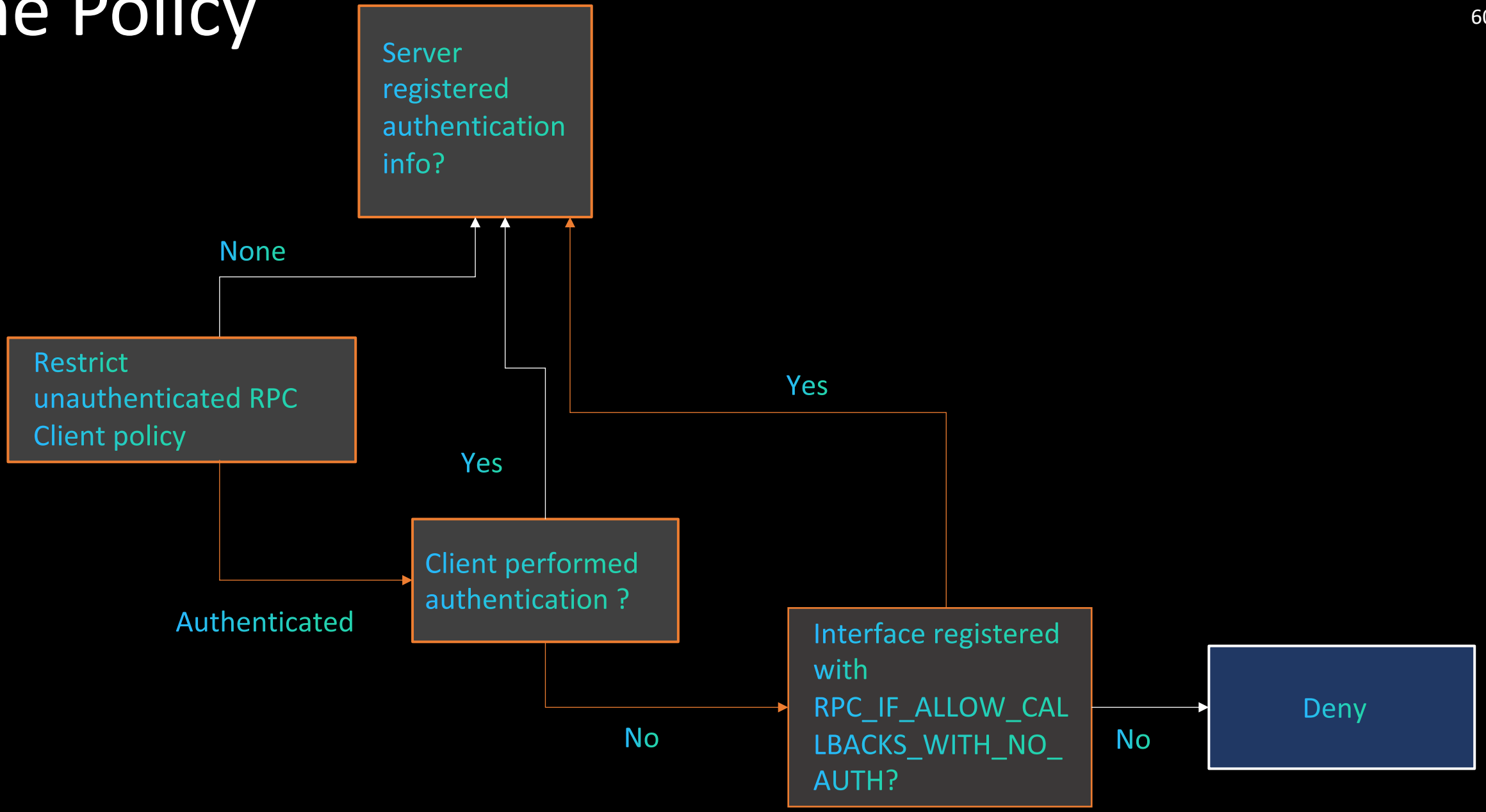
Transport

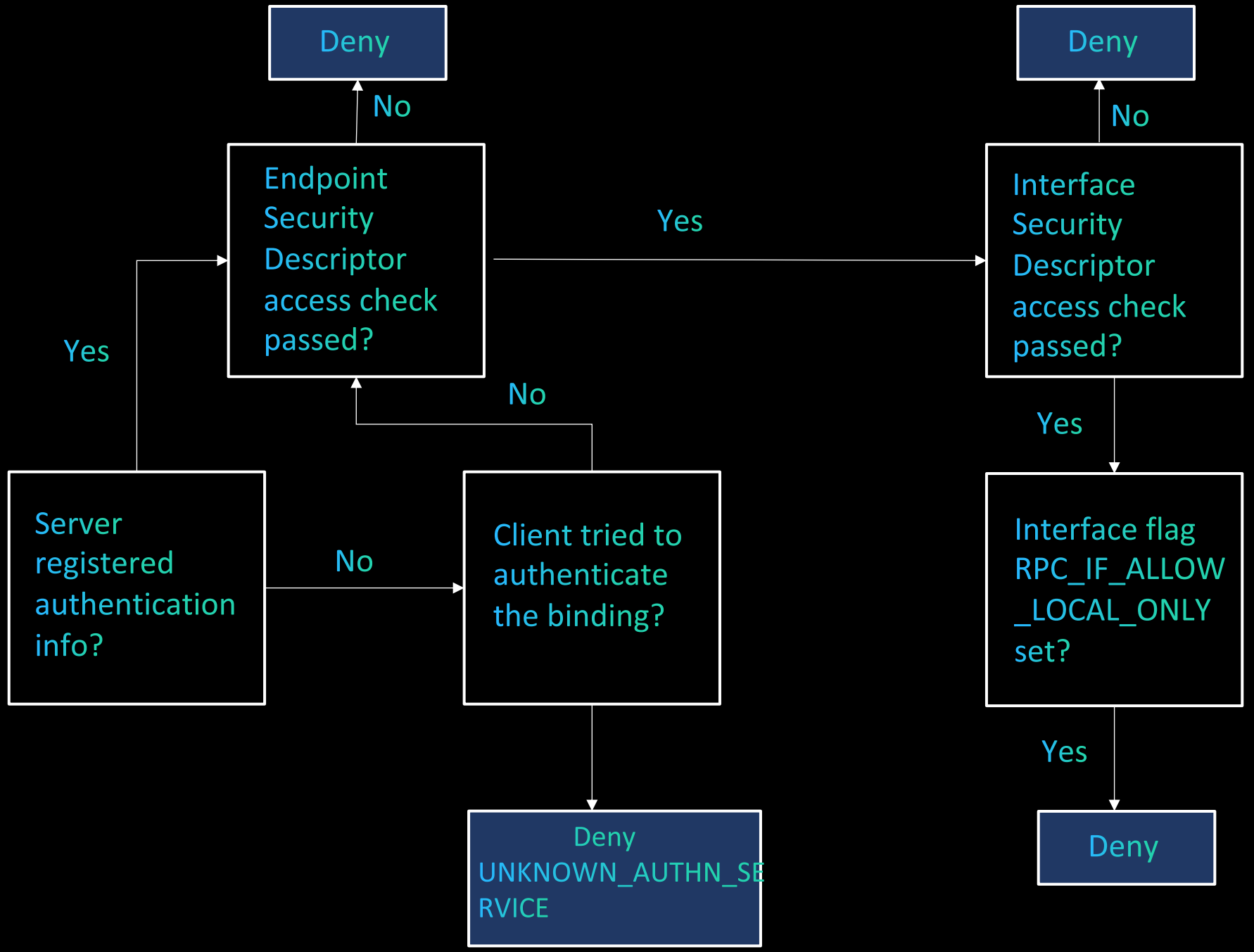

Server registered authentication info?
SMB IPC$
None
RPC Client
TCP
Restrict unauthenticated RPC Client policy
Client performed authentication ?
Authenticated

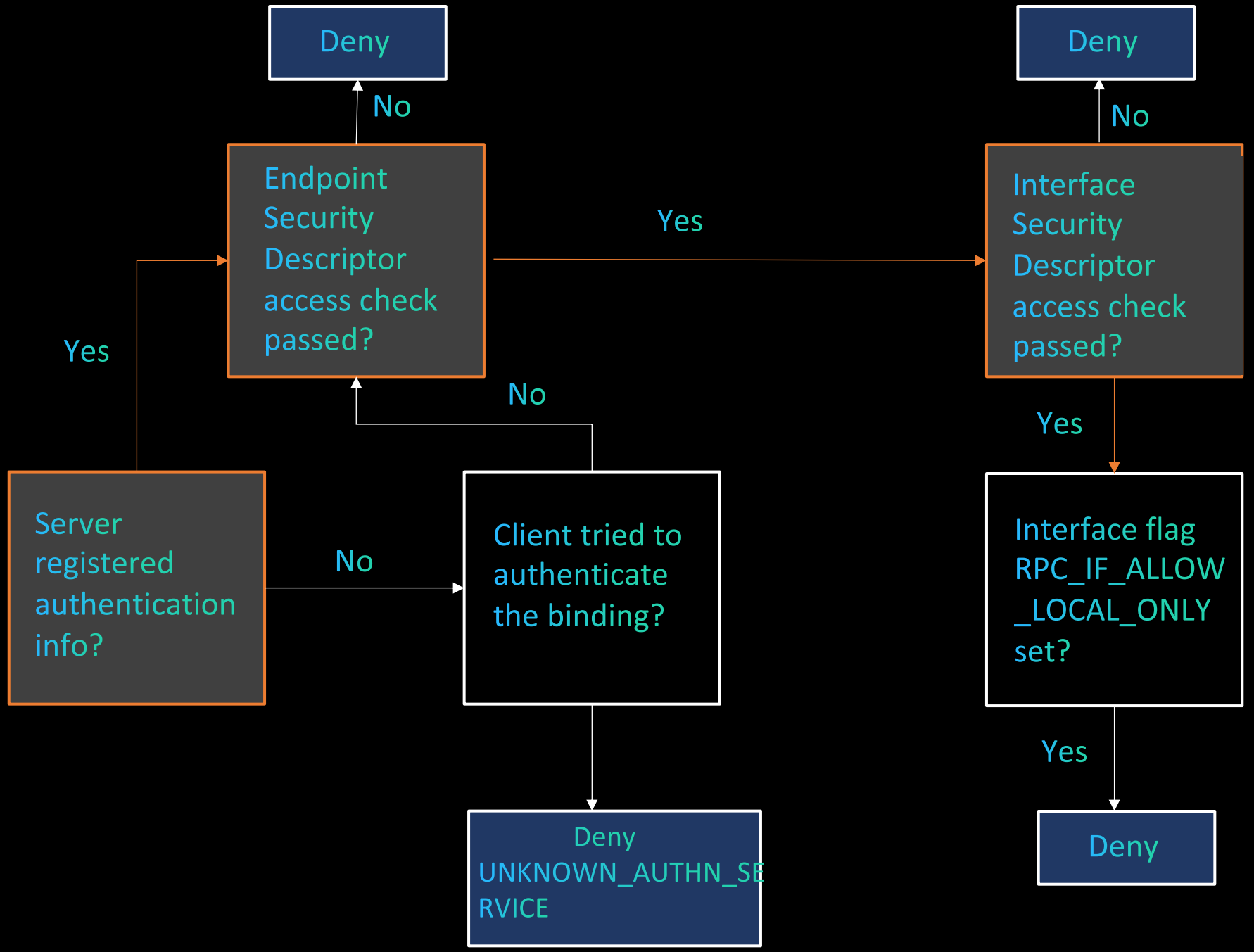
The Policy

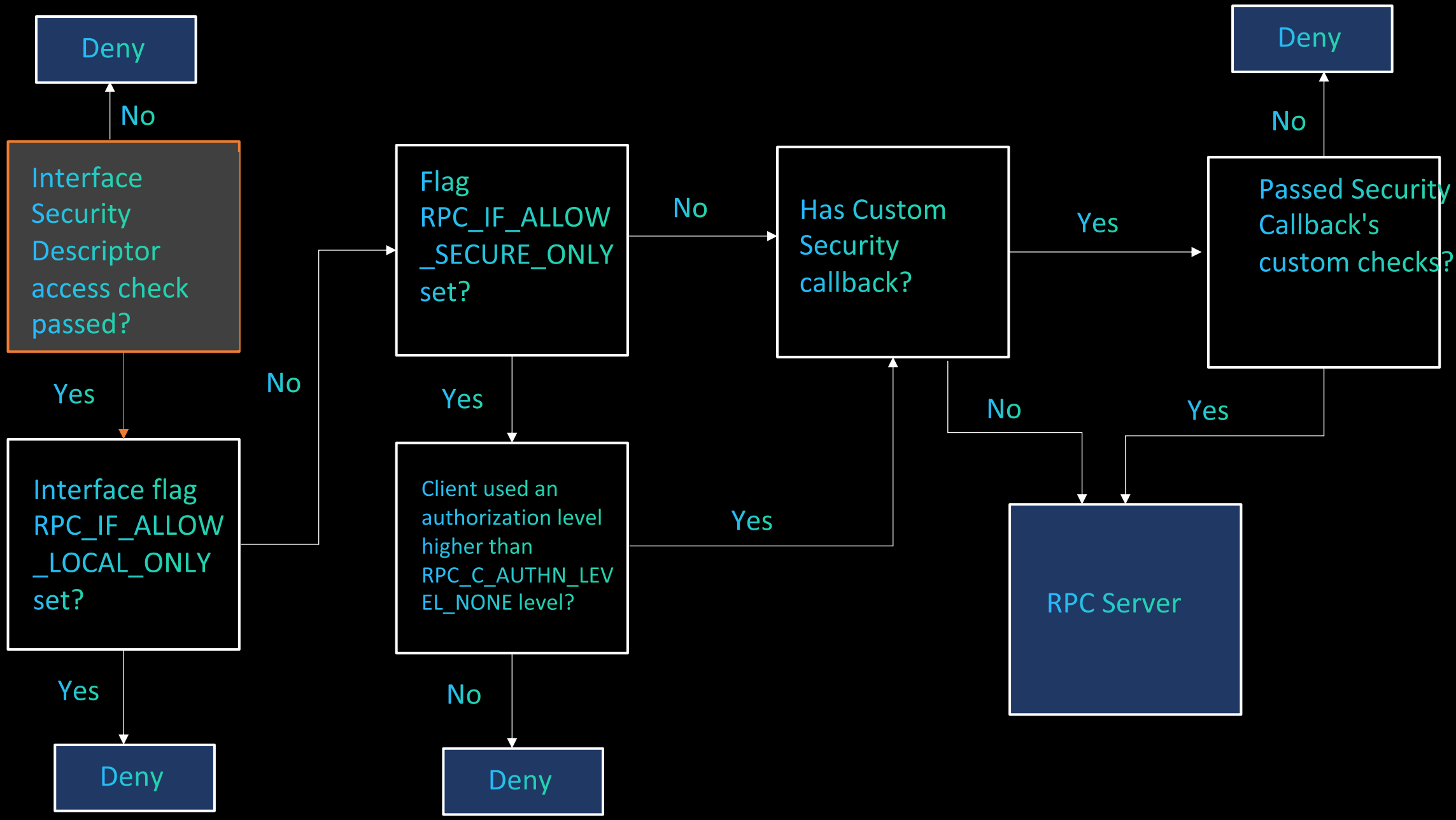

Server registered authentication info?
None
Restrict unauthenticated RPC Client policy
Authenticated
Yes
Client performed authentication ?
No
Yes
Interface registered with RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH?
No
Deny

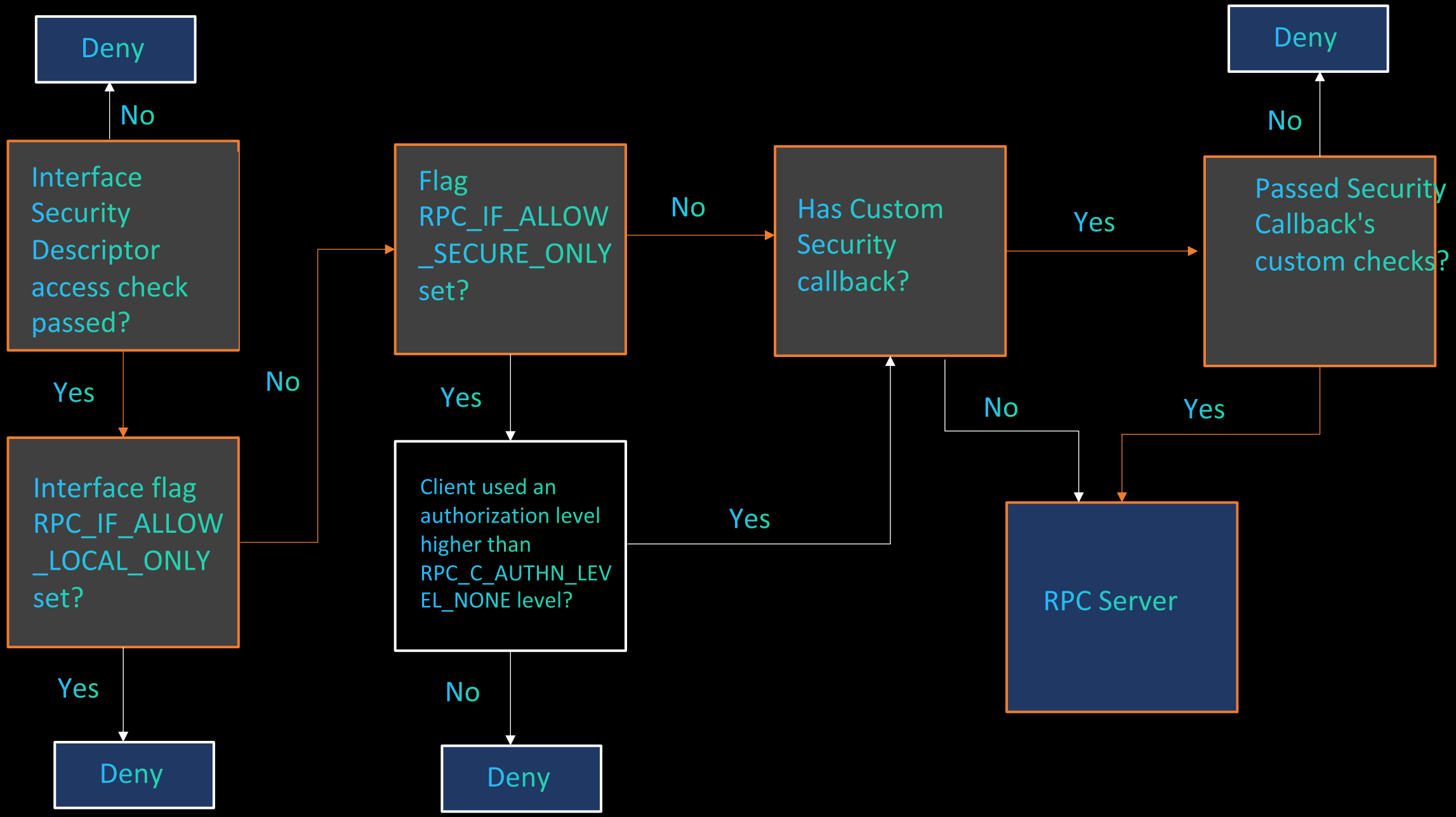
The Policy


Server registered authentication info?
None
Restrict unauthenticated RPC Client policy
Yes
Yes
Client performed authentication ?
Authenticated
Interface registered with RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH?
No
No
Deny


Deny
No
Endpoint Security Descriptor access check passed?
Yes
Interface Security Descriptor access check passed?
Deny
No
Yes
No
Server registered authentication info?
No
Client tried to authenticate the binding?
Interface flag RPC_IF_ALLOW_LOCAL_ONLY set?
Yes
Deny UNKNOWN_AUTHN_SERVICE
Yes
Deny


Deny
No
Endpoint Security Descriptor access check passed?
Yes
Interface Security Descriptor access check passed?
Deny
No
Yes
No
Server registered authentication info?
No
Client tried to authenticate the binding?
Interface flag RPC_IF_ALLOW_LOCAL_ONLY set?
Yes
Deny UNKNOWN_AUTHN_SERVICE
Yes
Deny


Deny
No
Interface Security Descriptor access check passed?
Yes
Interface flag RPC_IF_ALLOW_LOCAL_ONLY set?
Yes
Deny
No
Flag RPC_IF_ALLOW_SECURE_ONLY set?
Yes
Client used an authorization level higher than RPC_C_AUTHN_LEVEL_NONE level?
No
Deny
Yes
No
Has Custom Security callback?
Yes
No
Deny
Passed Security Callback's custom checks?
Yes
RPC Server


Deny
No
Interface Security Descriptor access check passed?
Yes
Interface flag RPC_IF_ALLOW_LOCAL_ONLY set?
Yes
Deny
No
Flag RPC_IF_ALLOW_SECURE_ONLY set?
Yes
Client used an authorization level higher than RPC_C_AUTHN_LEVEL_NONE level?
No
Deny
Yes
No
Has Custom Security callback?
Yes
No
Deny
Passed Security Callback's custom checks?
Yes
RPC Server

What we can get from surface analysis:

1- Registration flags:

- It has RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH (meaning it has security callback)
- No RPC_IF_ALLOW_LOCAL_ONLY flag
- No RPC_IF_ALLOW_SECURE_ONLY flag

2- securing the interface

- We are not sure

3- Registered binding authentication:

- The RPC server registers the authentication

MS-NRPC Security
B. In-Depth Analysis

Strategies

- The goal of our in-depth analysis is to leverage reverse engineering techniques to assess the security of the RPC server under the **MS-NRPC** interface
- The interface is accessible through the **LSASS** process, specifically via the **Netlogon DLL**
- We have two approaches for the analysis:

    1- Automated Tools Approach:

        Use automated tools to gather information about the interface
        Follow up with deeper investigation using IDA for further analysis

    2- Direct IDA Approach:

        Go straight to IDA and manually locate the interface and its related security mechanisms

Strategies
1. PE RPC Scraper automated tool

- The interface has 59 functions
- The interface has no security descriptor, security call back, and set of flags which equal to 0x91
- The flags:
  RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH | RPC_IF_SEC_CACHE_PER_PROC | RPC_IF_AUTOLISTEN


```
"interface_registration_info":{
   "0x1800275e1":{
      "interface_address":"0x18006d080",
      "flags":"0x91",
      "security_callback_addr":"0x18002ef60",
      "has_security_descriptor":false,
      "security_callback_info":{
         "security_callback_name":"NlNetlogonMain",
         "use_call_attributes":true
      },
      "global_caching_enabled":false
   }
```



```
"netlogon.dll":{
   "12345678-1234-abcd-ef00-01234567cffb":{
      "number_of_functions":59,
      "functions_pointers":[
         "0x180017ad0",
         "0x1800179d0",
```


https://github.com/akamai/akamai-security-research/tree/main/rpc_toolkit/pe_rpc_if_scraper

Strategies
2. IDA like superhero

- Personally I am not a big fan of automated tools in RE

- Let’s load netlogon inside IDA and start our investigation

Strategies
2. IDA like superhero
A. Locate the interface

- After doing some RE we can determine the interface registration function which is RpcServerRegisterIf3
- The UUIID is identical to nrpc interface

```
|| (v23 = RpcServerRegisterIf3(
          &unk_18006D080,
          0i64,
          0i64,
          0x91i64,
          1234,
          0,
          &sub_18002EF60,
          hMem)) != 0 )
```

```
.rdata:000000018006D084                 dd 12345678h            ; Data1
.rdata:000000018006D084                 dw 1234h                ; Data2
.rdata:000000018006D084                 dw 0ABCDh               ; Data3
.rdata:000000018006D084                 db 0EFh, 0, 1, 23h, 45h, 67h, 0CFh, 0FBh; Data4
```

Strategies
2. IDA like superhero
B. Endpoint registration

- From analysis we can see that we have three endpoints:
  1- Named pipe: through lsass
  2- ALPC: NETLOGON_LRPC with security descriptor
  3- TCP dynamic endpoint

```
RpcServerUseProtseqEpW(L"ncacn_np", 0xAu, L"\\pipe\\lsass", 0i64);
23 && v23 != 1740
v23 = RpcServerUseProtseqEpW(L"ncalrpc", 0xAu, L"NETLOGON_LRPC", hMem))
```

```
RpcServerUseProtseqExW(L"ncacn_ip_tcp", 0xAu, 0i64, &Policy);
```

Strategies
2. IDA like superhero
B. interface registration

- It used 0x91 as flags which are RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH | RPC_IF_SEC_CACHE_PER_PROC | RPC_IF_AUTOLISTEN
- It has security callback
- It has SECURITY DESCRIPTOR (not like automated tool's output)

```
(v23 = RpcServerRegisterIf3(
        &unk_18006D080,
        0i64,
        0i64,
        0x91i64,
        1234,
        0,
        sub_18002EF60,
        hMem)) != 0 )
```

```
RPC_STATUS RpcServerRegisterIf3(
  [in]                RPC_IF_HANDLE      IfSpec,
  [in, optional] UUID                    *MgrTypeUuid,
  [in, optional] RPC_MGR_EPV             *MgrEpv,
  [in]                unsigned int       Flags,
  [in]                unsigned int       MaxCalls,
  [in]                unsigned int       MaxRpcSize,
  [in, optional] RPC_IF_CALLBACK_FN *IfCallback,
  [in, optional] void                    *SecurityDescriptor
);
```

Strategies
2. IDA like superhero
C. Binding Authentication

- 0x44 will bind to RPC_C_AUTHN_NETLOGON (Netlogon authentication service)

```
v0 = RpcServerRegisterAuthInfoW(L"NetlogonSspi", 0x44u, 0i64, 0i64);
```

Strategies
2. IDA like superhero
D. Security callback

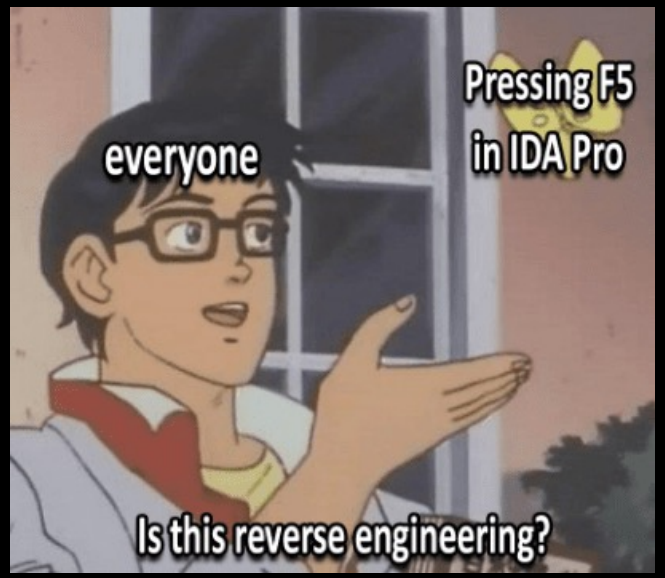
- For the previous opnums that can be accessed without authentication, all of them pass the access matrix check

```
memset(&RpcCallAttributes.ServerPrincipalNameBufferLength, 0, 0x68ui64);
RpcCallAttributes.Version = 2;
RpcCallAttributes.Flags = 96;
if ( !RpcServerInqCallAttributesW(a2, &RpcCallAttributes)
  && RpcCallAttributes.OpNum < 59u
  && ((byte_1800C8498[RpcCallAttributes.OpNum] & 2) == 0 || RpcCallAttributes.IsClientLocal == 1) )
{
```

```
0000000180C8498 byte_1800C8498  db 2 dup(1), 5 dup(0), 4 dup(1), 5 dup(0), 2 dup(1), 3 dup(0)
0000000180C8498                                         ; DATA XREF: sub_18002EF60+73↑o
0000000180C8498                 db 1, 2 dup(2), 4 dup(0), 4, 5 dup(0), 4, 0Ch dup(0), 1
0000000180C8498                 db 2 dup(0), 4 dup(2), 0, 4 dup(2), 5 dup(0)
```

Strategies
2. IDA like superhero
D. Security callback

The IDA decompile tab for this function is not accurate. Therefore, let's move to assembly for analysis




```
lea     rdx, [rbp+57h+RpcCallAttributes] ; RpcCallAttributes
mov     [rbp+57h+RpcCallAttributes], 2
mov     rcx, rsi        ; ClientBinding
mov     [rbp+57h+var_7C], 60h ; '`'
call    cs:RpcServerInqCallAttributesW
nop     dword ptr [rax+rax+00h]
test    eax, eax
jnz     loc_18002F13F
cmp     [rbp+57h+var_28], 3Bh ; ';'
jnb     loc_18002F13D
movzx   edi, [rbp+57h+var_28]
lea     rax, byte_1800C8498
add     rdi, rax
test    byte ptr [rdi], 2
jz      short loc_18002EFF2
cmp     [rbp+57h+var_44], 1
jz      short loc_18002EFF2
mov     eax, 5
jmp     loc_18002F13F
```

Strategies
2. IDA like superhero
D. Security callback (inner condition)

- The code will verify if the product is a Domain Controller (DC). If it is, the execution will continue, and it will return 0
- If the product is not a DC, additional checks will be performed

```
if ( dword_1800C9DB8 )
{
  RtlAcquireResourceShared(&stru_1800C94E0, 1u);
  if ( qword_1800C9540 )
```

```
NtProductType = RtlGetNtProductType(&ProductType);
v9 = ProductType;
if ( !NtProductType )
  v9 = NtProductWinNt;
ProductType = v9;
if ( v9 != NtProductLanManNt || (v10 = 0, !SampDsIsRunning()) )
  v10 = 1;
dword_1800C9DB8 = v10;
```

Strategies
2. IDA like superhero
D. Security Descriptor

- Security descriptor is set up by calling another function that contains many instructions

- The following groups of users have read access: Anonymous Logon, Everyone, Restricted Code, Built-in Administrators, Application Package, and a specific SID

```
v2 = L"D:(A;;GR;;;AN)(A;;GR;;;WD)(A;;GR;;;RC)(A;;GR;;;BA)(A;;GR;;;AC)(A;;GR;;;S-1-15-3-1024-1788129303-2183208577-399947"
     "4272-3147359985-1757322193-3815756386-151582180-1888101193)";
```

```
ConvertStringSecurityDescriptorToSecurityDescriptorW(StringSecurityDescriptor, 1u, SecurityDescriptor, 0i64) )
```

Strategies
2. IDA like superhero
D. Security Descriptor

- We have two options for checking the security descriptor:

  1- Use a memory search to look for the well-known value inside the header of the relative security descriptor
  2- Check the security descriptor for the ALPC endpoint, as it shares the same security descriptor as the interface

```
(v23 = RpcServerUseProtseqEpW(L"ncalrpc", 0xAu, L"NETLOGON_LRPC", hMem)) != 0 && v23 != 1740
(v23 = RpcServerRegisterIf3(&unk_18006D080, 0i64, 0i64, 145i64, 1234, 0, sub_18002EF60, hMem)) != 0 )
```

```
PS C:\Users\Administrator\Desktop\SysinternalsSuite> $h = Get-NtAlpcServer '\RPC Control\NETLOGON_LRPC'
PS C:\Users\Administrator\Desktop\SysinternalsSuite> $sd = Get-NtSecurityDescriptor $h
PS C:\Users\Administrator\Desktop\SysinternalsSuite> Show-NtSecurityDescriptor $sd
```

DACL

Flags: None

ACL Entries

Type	Account	Access	Flags
Allowed	NT AUTHORITY\ANONYMOUS LOGON	GenericRead	None
Allowed	Everyone	GenericRead	None
Allowed	NT AUTHORITY\RESTRICTED	GenericRead	None
Allowed	BUILTIN\Administrators	GenericRead	None
Allowed	APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES	GenericRead	None
Allowed	NAMED CAPABILITIES\Lpac Identity Services	GenericRead	None

https://www.powershellgallery.com/packages/NtObjectManager/1.1.32

Conclusion for this part:

1- Registration flags:

- It has three flags
  RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH | RPC_IF_SEC_CACHE_PER_PROC | RPC_IF_AUTOLISTEN

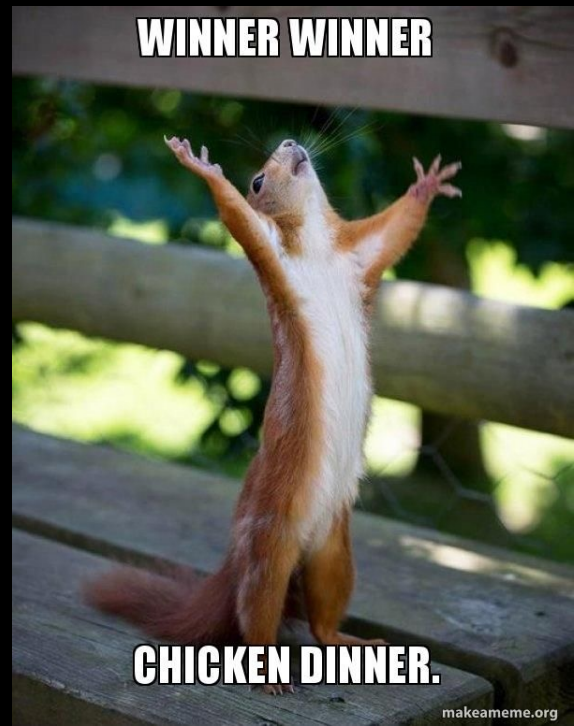
- It has security callback which in our case it’s used to check if we will pass the check against the access array

2- Securing the interface:

- The interface has a security descriptor (SD) that permits multiple user groups to connect, including anonymous users. Since we are using no authentication, the access check will be performed against the anonymous user, allowing access to the interface’s functions

3- Registered binding authentication:

- The RPC server registers the authentication under netlogon authentication service



ank You For Listening

https://github.com/klsecservices

X: @haider_kabibo