

PHdays 2024

Make Null Session Great Again

Haidar Kabibo

- **Member of Industrial Security Services team, Kaspersky**
- **Interested in windows internals, network protocols, communication systems, and industrial infrastructure**

```
$ echo d2hvYW1pCg== | base64 -d | bash  
Sud0Ru
```

**What this talk
about?**

Agenda

1

How It All Started

2

MSRPC Crush Course

3

Null Session

4

Research Methodology

5

Enumerate Domain Info

6

Prevention

7

Detection

8

Conclusion

How it all started?

- Intersecting MSRPC call during traffic analysis
- DC network interfaces in clear text

Source	Destination	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=[redacted]
- > StringBinding[2]: TowerId=NCACN_IP_TCP, NetworkAddr=[redacted] 14"
- > StringBinding[3]: TowerId=NCACN_IP_TCP, NetworkAddr=[redacted] 14"
- > StringBinding[4]: TowerId=NCACN_IP_TCP, NetworkAddr=[redacted] .5"
- > StringBinding[5]: TowerId=NCACN_IP_TCP, NetworkAddr=[redacted] .12"
- > SecurityBinding[1]: AuthnSvc=0x0009, AuthzSvc=0xffff, PrincName=""

How it all started?

6

- 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 IOXIDResolver 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
```

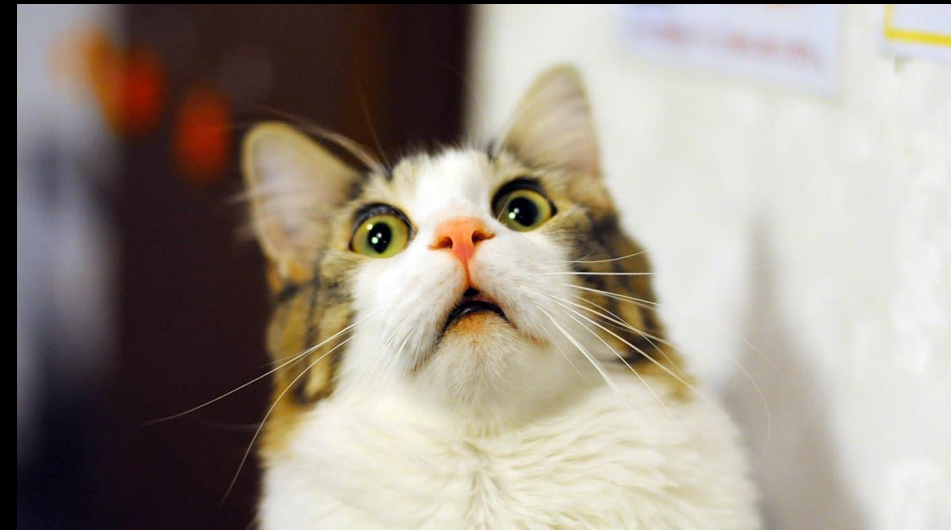
```
#define RPC_C_AUTHN_LEVEL_DEFAULT      0  
#define RPC_C_AUTHN_LEVEL_NONE        1  
#define RPC_C_AUTHN_LEVEL_CONNECT     2  
#define RPC_C_AUTHN_LEVEL_CALL        3  
#define RPC_C_AUTHN_LEVEL_PKT         4  
#define RPC_C_AUTHN_LEVEL_PKT_INTEGRITY 5  
#define RPC_C_AUTHN_LEVEL_PKT_PRIVACY 6
```

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

RPC,DCE/RPC, MSRPC

7

- 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].



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:

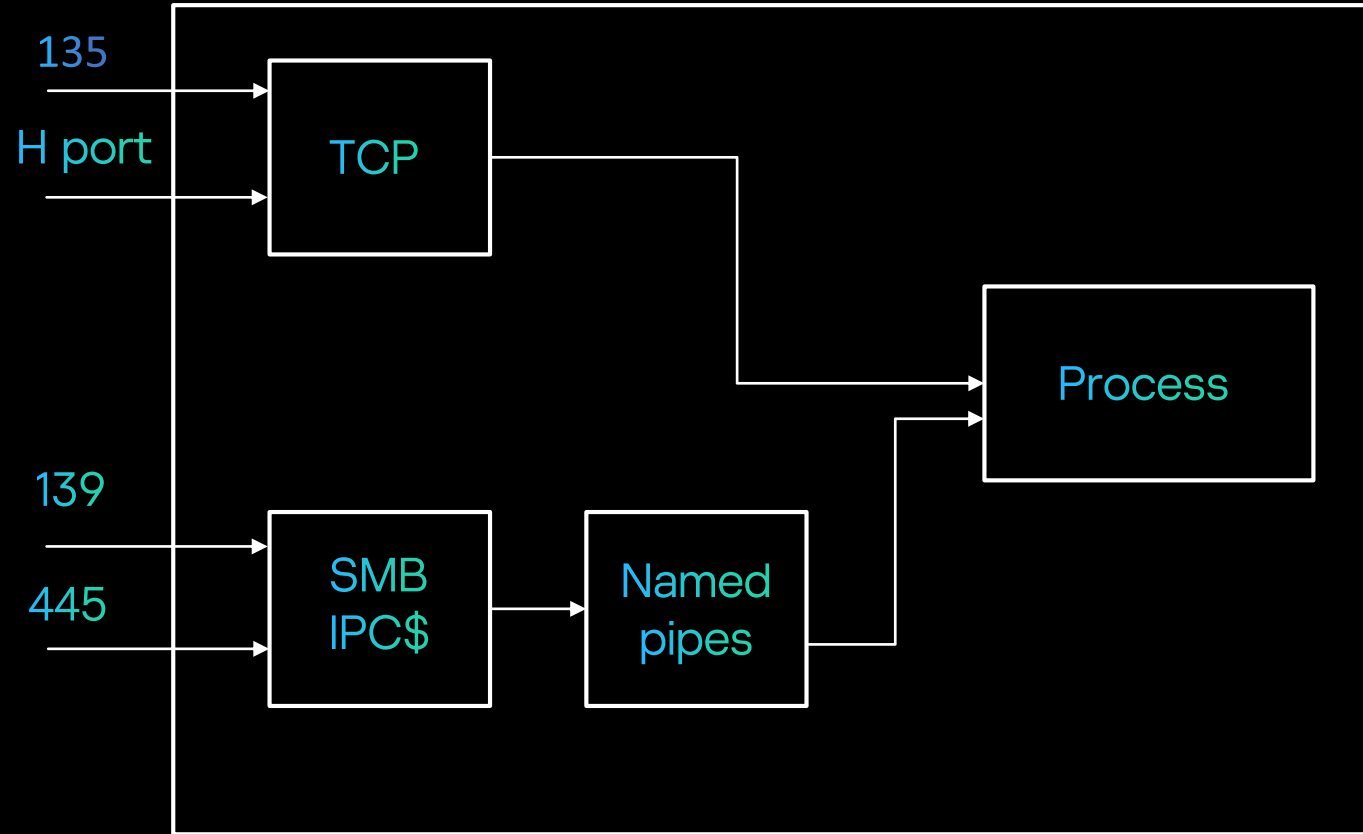
In order to call a procedure or function on a remote server, it needs to bind to the appropriate interface using its unique IID.

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)

```

▼ SMB2 (Server Message Block Protocol version 2)
  ▶ SMB2 Header
  ▼ Write Request (0x09)
    ▶ StructureSize: 0x0031
      Data Offset: 0x0070
      Write Length: 26
      File Offset: 0
    ▶ GUID handle File: lsarpc
      Channel: None (0x00000000)
      Remaining Bytes: 0
    ▶ Write Flags: 0x00000000
      Blob Offset: 0x00000000
      Blob Length: 0
      Channel Info Blob: NO DATA
  ▼ Distributed Computing Environment / Remote Procedure Ca
    Version: 5
    Version (minor): 0
    Packet type: Request (0)
    ▶ Packet Flags: 0x03
    ▶ Data Representation: 10000000 (Order: Little-endian,
      Frag Length: 26
      Auth Length: 0
      Call ID: 1
      Alloc hint: 2
      Context ID: 0
      Opnum: 0
  
```

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

12

- 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

```
#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
```



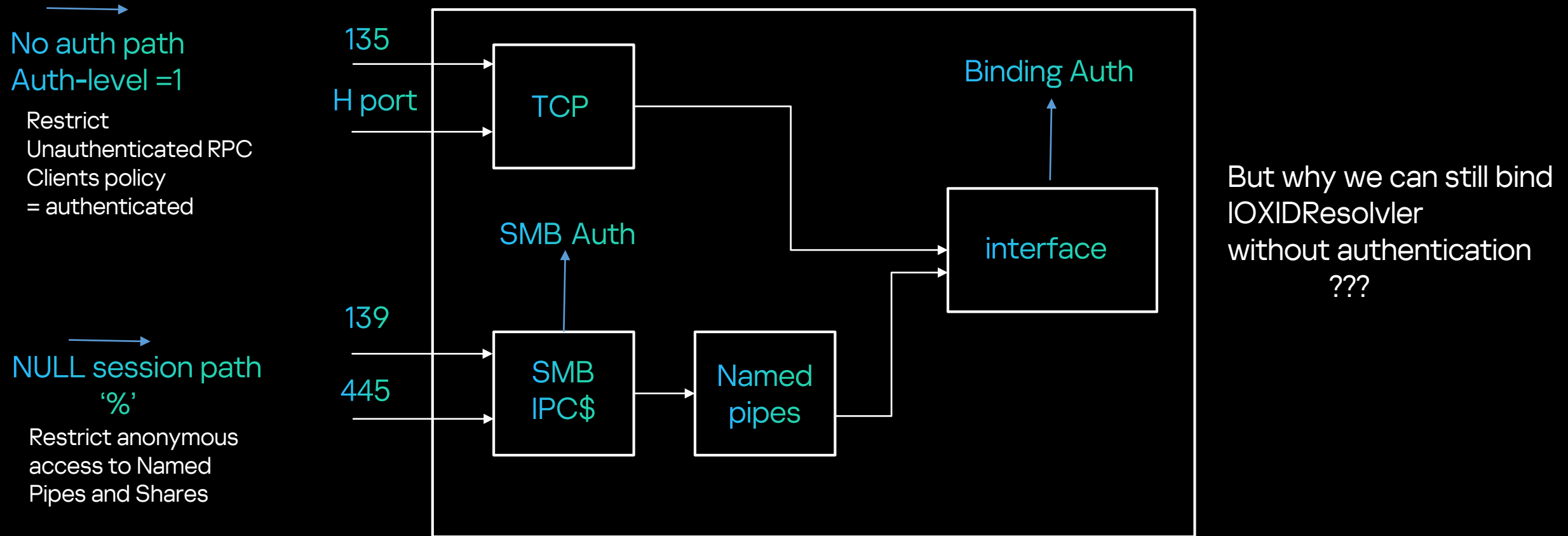
Restrict Unauthenticated RPC Clients policy

13

- “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

14



Security Research Roadmap Against MSRPC Interfaces



Overview Of System Security Policy For Target System

16

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.

17

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-S09H290I5NL[\\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-S09H290I5NL[\\pipe\\lsass]
```

Research Roadmap: Enumerating RPC Endpoints.

18

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.

19

- Rpcmap.py another Impacket 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 .

```
L$ impacket-rpcmap ncacn_ip_tcp:192.168.177.177[135] -auth-rpc Administrator:Asd123456#
Impacket v0.11.0 - Copyright 2023 Fortra

Protocol: N/A
Provider: rpcss.dll
UUID: 00000136-0000-0000-C000-000000000046 v0.0

Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: rpcss.dll
UUID: 000001A0-0000-0000-C000-000000000046 v0.0

Protocol: N/A
Provider: rpcss.dll
UUID: 0B0A6584-9E0F-11CF-A3CF-00805F68CB1B v1.1

Protocol: N/A
Provider: rpcss.dll
UUID: 1D55B526-C137-46C5-AB79-638F2A68E869 v1.0

Protocol: N/A
Provider: rpcss.dll
UUID: 412F241E-C12A-11CE-ABFF-0020AF6E7A17 v0.2

Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: rpcss.dll
UUID: 4D9F4AB8-7D1C-11CF-861E-0020AF6E7C57 v0.0

Protocol: N/A
Provider: rpcss.dll
UUID: 64FE0B7F-9EF5-4553-A7DB-9A1975777554 v1.0

Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: rpcss.dll
UUID: 99FCFEC4-5260-101B-BBCB-00AA0021347A v0.0
```

Research Roadmap: Brute-force OPNUMs.

20

- 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 denied 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 denied for every bind attempt
- Solution: we should change rpcmap internal work

Research Roadmap: Brute-force OPNUMs.

21

- 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.
- we receive 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 BFF-0020AF6E7A17 v0.2
Opnum 3: success
Opnum 4: rpc_x_bad_stub_data ibuted Component Object Model (DCOM) Remote
Opnum 5: success
Opnums 6-64: nca_s_op_rng_error (opnum not found)
```

Research Roadmap:

Case Study (IObjectExporter interface).

22

- 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



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).

24

- 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 rpcmap.py script with the actual methods under the IObjectExporter interface on the remote host.

The screenshot displays the RPCView application interface with several panels:

- Decompile:** Shows a decompiled code snippet: `uuid(99fec4-5260-101b-bbcb-00aa0021347a, version(0,0), ...)`.
- Endpoints:** A table listing network endpoints. The first row is highlighted with a red box.
- Processes:** A table listing running processes. The row for `svchost.exe` with PID 908 is highlighted with a red box.
- Interfaces:** A table listing RPC interfaces. The row for `99fec4-5260-101b-bbcb-00aa0021347a` is highlighted with a red box.
- Procedures:** A table listing RPC procedures. The first row is highlighted.
- Interface Properties:** A panel on the right showing details for the selected interface, including Syntax, NDR Version, MIDL Version, NDR Flags, TypeFormatString, and ProcFormatString.

Pid	Protocol	Name
908	ncacn_ip_tcp	135
908	ncacn_http	593
908	ncacn_np	\pipe\epmapper
908	ncalrpc	epmapper

Name	Pid	Path
SppExtComObj.Exe	4756	C:\Windows\System32\SppExtComObj.Exe
SearchUI.Exe	5572	C:\Windows\SystemApps\Microsoft.Windows.Cortana_cw5n...
WmiPrvSE.Exe	1616	C:\Windows\System32\wbem\WmiPrvSE.Exe
dllhost.exe	2132	C:\Windows\System32\dllhost.exe
svchost.exe	908	C:\Windows\System32\svchost.exe
svchost.exe	924	C:\Windows\System32\svchost.exe
svchost.exe	952	C:\Windows\System32\svchost.exe
svchost.exe	972	C:\Windows\System32\svchost.exe
IpOverUsbSvc.exe	1036	C:\Program Files (x86)\Common Files\microsoft shared\Pho...
svchost.exe	1064	C:\Windows\System32\svchost.exe
svchost.exe	1080	C:\Windows\System32\svchost.exe
svchost.exe	1148	C:\Windows\System32\svchost.exe
vmtoolsd.exe	1196	C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
svchost.exe	1220	C:\Windows\System32\svchost.exe
svchost.exe	1296	C:\Windows\System32\svchost.exe

Pid	Uuid	Ver	Type	Procs	Stub	Callback	Name
908	412f241e-c12a-11ce-abff-0020af6e7a17	0.2	RPC	28	Interpreted	0x00007fff0839f8f0	
908	4d9f4ab8-7d1c-11cf-861e-0020af6e7c57	0.0	RPC	1	Interpreted	0x00007fff084197c0	
908	64fe0b7f-9ef5-4553-a7db-9a1975777554	1.0	RPC	3	Interpreted	0x00007fff083473b0	
908	99fec4-5260-101b-bbcb-00aa0021347a	0.0	RPC	6	Interpreted	0x00007fff08418e30	

Index	Name	Address	Format
0		0x00007fff083f7c10	0x00007fff08451492
1		0x00007fff083f8630	0x00007fff084514da
2		0x00007fff083f78d0	0x00007fff08451504
3		0x00007fff083f8400	0x00007fff08451552

Research Roadmap: Case Study (RPCView).

25

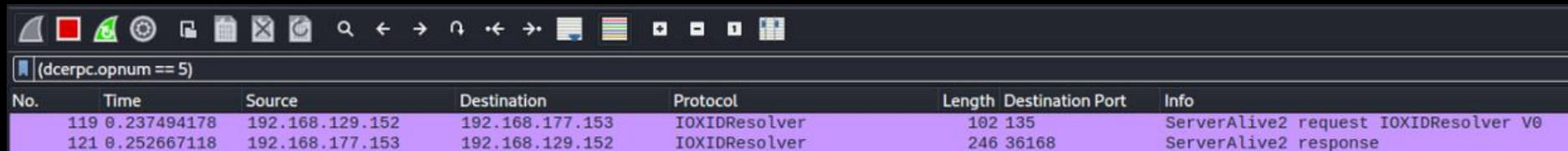
- RPCView, decompilation tab shows the declarations of each method (Proc) are displayed Methods.
- Proc3 and Proc5 can be invoked without any arguments, unlike the other methods.
- The similarity between Proc3 and Proc5 in the declarations aligns with the findings from rpcmap.py

```
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).

26

- 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.	Time	Source	Destination	Protocol	Length	Destination Port	Info
119	0.237494178	192.168.129.152	192.168.177.153	IOXIDResolver	102	135	ServerAlive2 request IOXIDResolver V0
121	0.252667118	192.168.177.153	192.168.129.152	IOXIDResolver	246	36168	ServerAlive2 response

```
unknown 8 bytes 1: 0x0000003500020000
  Address: STRINGBINDINGS=2, SECURITYBINDINGS=7
    NumEntries: 53
    SecurityOffset: 31
    StringBinding[1]: TowerId=NCACN_IP_TCP, NetworkAddr="TEST-Server"
    StringBinding[2]: TowerId=NCACN_IP_TCP, NetworkAddr="192.168.177.153"
    SecurityBinding[1]: AuthnSvc=0x0009, AuthzSvc=0xffff, PrincName=""
    SecurityBinding[2]: AuthnSvc=0x001e, AuthzSvc=0xffff, PrincName=""
    SecurityBinding[3]: AuthnSvc=0x0010, AuthzSvc=0xffff, PrincName=""
    SecurityBinding[4]: AuthnSvc=0x000a, AuthzSvc=0xffff, PrincName=""
    SecurityBinding[5]: AuthnSvc=0x0016, AuthzSvc=0xffff, PrincName=""
    SecurityBinding[6]: AuthnSvc=0x001f, AuthzSvc=0xffff, PrincName=""
    SecurityBinding[7]: AuthnSvc=0x000e, AuthzSvc=0xffff, PrincName=""
```

No Authentication Against MS-NRPC Protocol

27

- 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.

```
L$ impacket-rpcmap ncacn_ip_tcp:192.168.177.177[49664] -uuid '12345678-1234-ABCD-EF00-01234567CFFB v1.0' -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: rpc_x_bad_stub_data
Opnum 4: rpc_x_bad_stub_data
Opnum 5: rpc_x_bad_stub_data
Opnum 6: rpc_x_bad_stub_data
Opnum 7: rpc_x_bad_stub_data
Opnum 8: rpc_x_bad_stub_data
Opnum 9: rpc_x_bad_stub_data
Opnum 10: rpc_x_bad_stub_data
Opnum 11: rpc_x_bad_stub_data
Opnum 12: rpc_x_bad_stub_data
Opnum 13: rpc_x_bad_stub_data
Opnum 14: rpc_x_bad_stub_data
Opnum 15: rpc_x_bad_stub_data
Opnum 16: rpc_x_bad_stub_data
Opnum 17: rpc_x_bad_stub_data
Opnum 18: rpc_x_bad_stub_data
Opnum 19: rpc_x_bad_stub_data
Opnum 20: rpc_x_bad_stub_data
Opnum 21: rpc_x_bad_stub_data
Opnum 22: rpc_s_access_denied
Opnum 23: rpc_s_access_denied
Opnum 24: rpc_x_bad_stub_data
Opnum 25: rpc_x_bad_stub_data
Opnum 26: rpc_x_bad_stub_data
Opnum 27: rpc_x_bad_stub_data
Opnum 28: rpc_x_bad_stub_data
```

What Is MS-NRPC Protocol?

28

- 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

29

The screenshot displays the RPCView application interface, which is used for analyzing Remote Procedure Call (RPC) traffic. The interface is divided into several panes:

- Endpoints:** A table listing active endpoints. The first three entries are highlighted with a red box:

Pid	Protocol	Name
672	ncacn_http	49670
672	ncacn_ip_tcp	49664
672	ncacn_ip_tcp	49668

- Decompilation:** A pane showing the decompiled code for the selected endpoint. It displays a C++-like structure definition for 'StructMember0' and an interface 'DefaultIfName'.
- Processes:** A table listing running processes. The 'lsass.exe' process is highlighted with a red box:

Name	Pid	Path
VGAAuthService.exe	2504	C:\Program Files\VMware\VMware Tools\
vmtoolsd.exe	2512	C:\Program Files\VMware\VMware Tools\
vm3dservice.exe	2524	C:\Windows\System32\vm3dservice.exe
vm3dservice.exe	2824	C:\Windows\System32\vm3dservice.exe
vm3dservice.exe	4892	C:\Windows\System32\vm3dservice.exe
MsMpEng.exe	2556	
wlms.exe	2572	C:\Windows\System32\wlms\wlms.exe
svchost.exe	2776	C:\Windows\System32\svchost.exe
dllhost.exe	3188	C:\Windows\System32\dllhost.exe
msdtc.exe	3668	C:\Windows\System32\msdtc.exe
NisSrv.exe	3872	
svchost.exe	4220	C:\Windows\System32\svchost.exe
rpcFwManager.exe	5280	C:\Users\Administrator\Desktop\RPCFW
svchost.exe	5860	
lsass.exe	672	C:\Windows\System32\lsass.exe
fontdrvhost.exe	2680	C:\Windows\System32\fontdrvhost.exe
csrss.exe	540	
winlogon.exe	596	C:\Windows\System32\winlogon.exe
dwm.exe	388	C:\Windows\System32\dwm.exe
fontdrvhost.exe	2548	C:\Windows\System32\fontdrvhost.exe
MicrosoftEdgeUpdate.exe	3300	C:\Program Files (x86)\Microsoft\EdgeU

- Processes Properties:** A pane showing details for the selected process (lsass.exe). It includes fields for Image, RPC, SSP, Version, Path, CmdLine, User, Desktop, and Image (64-bits).
- Interface Properties:** A pane showing details for the selected interface. It includes fields for Syntax, NDR Version, MIDL Version, NDR Flags, TypeFormatString, ProcFormatString, and ExpressionEvaluation.
- Interfaces:** A table listing loaded interfaces. The first three entries are highlighted with a red box:

Pid	Uuid	Ver	Type	Procs	Stub	Callback	Name	Base	Location	Flags	Description
672	12345678-1234-abcd-ef00-01234567cffb	1.0	RPC	59	Interpreted	0x00007ffdaac3ef60		0x00007ffdaac10000	C:\Windows\System32\netlogon.dll	0x91	Net Logon Services DLL
672	12345778-1234-abcd-ef00-0123456789ab	0.0	RPC	134	Interpreted	0x00007ffdb3e2d30		0x00007ffdb3b0000	C:\Windows\System32\lsasrv.dll	0x91	LSA Server DLL
672	12345778-1234-abcd-ef00-0123456789ac	1.0	RPC	74	Interpreted	0x00007ffdb213cf0		0x00007ffdb210000	C:\Windows\System32\samsrv.dll	0x51	SAM Server DLL

Wireshark Against MS-NRPC

30

- 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

31

- 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

32

- 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

```
Payload stub data (68 bytes)
  Microsoft Network Logon, DsrGetDcNameEx2
    Operation: DsrGetDcNameEx2 (34)
    [Response in frame: 10]
    NULL Pointer: Server Handle
  Client Account: Administrator
    Referent ID: 0x0000ccde
    Max Count: 14
    Offset: 0
    Actual Count: 14
    Acct Name: Administrator
    Unknown long: 0x00000200
    NULL Pointer: Client Account
    NULL Pointer: Domain GUID:
    NULL Pointer: Client Site
    Unknown long: 0x00000000
```

```
[Request in frame: 23]
  DOMAIN_CONTROLLER_INFO:
    Referent ID: 0x00020000
  DOMAIN_CONTROLLER_INFO:
    DC Name: \\WIN-S09H290I5NL.test.local
    DC Address: \\192.168.177.177
    DC Address Type: IP/DNS name (1)
    GUID: fc43cec3-957b-464c-b319-3f1ed64bedca
    Logon Domain: test.local
    DNS Forest: test.local
    Domain Controller Flags: 0xe003f3fd
    DC Site: Default-First-Site-Name
    Client Site: Default-First-Site-Name
    DOS error code: Success (0x00000000)
```

[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

33

- The NetrEnumerateTrustedDomainsEx [1] return a list of trusted domains from a specified server, 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

34

```
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-S09H290I5NL.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

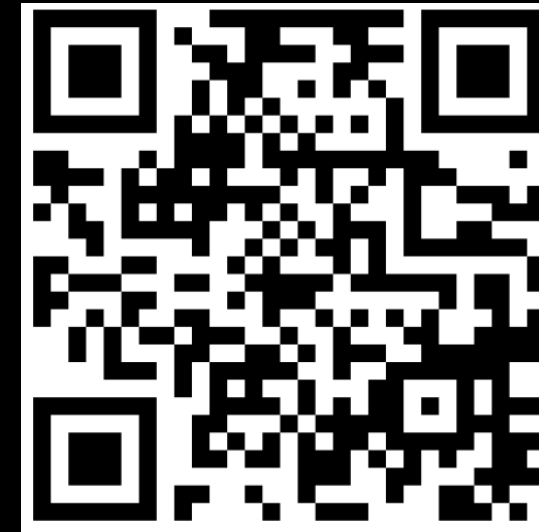
```
[*] Trusted Domain number 0
  • NetBios Domain Name: test.main.trust
  • DNS Domain Name: Not Available
  • Flags: DS_DOMAIN_DIRECT_OUTBOUND
  • Parent Index: Not Available
  • Trust Type: TRUST_TYPE_MIT
  • Trust Attributes: TRUST_ATTRIBUTE_NON_TRANSITIVE
  • Domian SID: Not Available
  • Domain 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>

```
msf6 auxiliary(gather/nrpc_enumusers) > set RHOST 192.168.177.177
RHOST => 192.168.177.177
msf6 auxiliary(gather/nrpc_enumusers) > set USER_FILE [REDACTED]users.txt
USER_FILE => ~/rpc_testing/the_tool/the_toolv3/users.txt
msf6 auxiliary(gather/nrpc_enumusers) > run
[*] 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
[*] Auxiliary module execution completed
msf6 auxiliary(gather/nrpc_enumusers) > █
```

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 An DCOM Object

38

- 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 IObjectExport 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: Deep Dive

39

- 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:

Deep Dive

40

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

No.	Time	Source	Destination	Protocol	Length	Info
20	2.950855	192.168.177.188	192.168.177.177	TCP	54	58534 → 135 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
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...

> Frame 21: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits) on interface \Device\NPF_{418A5627-C016-4EF8-95B2-27B6F245D43F}, id 0	0000	00 0c 29
> Ethernet II, Src: VMware_c0:64:de (00:0c:29:c0:64:de), Dst: VMware_83:2a:2a (00:0c:29:83:2a:2a)	0010	00 9c 8e
> Internet Protocol Version 4, Src: 192.168.177.188, Dst: 192.168.177.177	0020	b1 b1 e4
> Transmission Control Protocol, Src Port: 58534, Dst Port: 135, Seq: 1, Ack: 1, Len: 116	0030	20 14 e5
▼ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind, Fragment: Single, FragLen: 116, Call: 8	0040	00 00 08
Version: 5	0050	00 00 00
Version (minor): 0	0060	00 aa 00
Packet type: Bind (11)	0070	c9 11 9f
> Packet Flags: 0x03	0080	01 00 c4
> Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)	0090	34 7a 00
Frag Length: 116	00a0	00 00 00
Auth Length: 0		
Call ID: 8		
Max Xmit Frag: 5840		
Max Recv Frag: 5840		
Assoc Group: 0x00000000		
Num Ctx Items: 2		
> Ctx Item[1]: Context ID:0, IOXIDResolver, 32bit NDR		
> Ctx Item[2]: Context ID:1, IOXIDResolver, Bind Time Feature Negotiation		

WMI As A DCOM Object: Deep Dive

41

- 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?

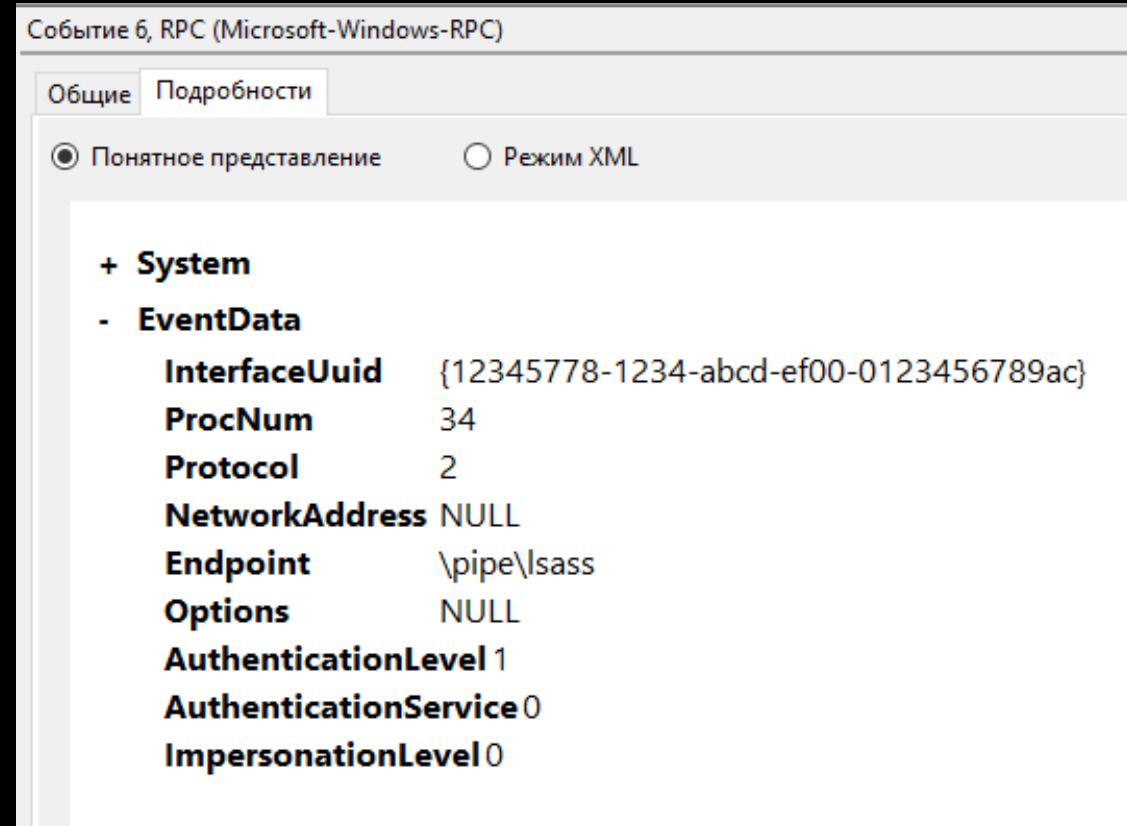
42



The Event That Never Occurs (Native Protection).

43

- 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

44

- 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.

```
+ System
- EventData
    NdrStubCall2
    672
    C:\Windows\system32\lsass.exe
    ncacn_ip_tcp
    49664
    192.168.177.111
    12345678-1234-abcd-ef00-01234567cffb
    34
    UNKNOWN
    UNKNOWN
    UNKNOWN
    56738
    192.168.177.177
    49664
    S-1-0-0
```

[1] <https://github.com/zeronetworks/rpcfirewall>

[2] https://www.youtube.com/watch?v=hZ_YPIMeBMI&ab_channel=BlackHat

Conclusion

- 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

Thank You For Listening



<https://t.me/purpleshift>



<https://sud0ru.ghost.io>

Haidar Kabibo

Twitter: @haider_Kabibo