CVE-2022-26809 - MSRPC: Vulnerability Analysis and Potential Mitigations

Agenda

SANS Analyst Program

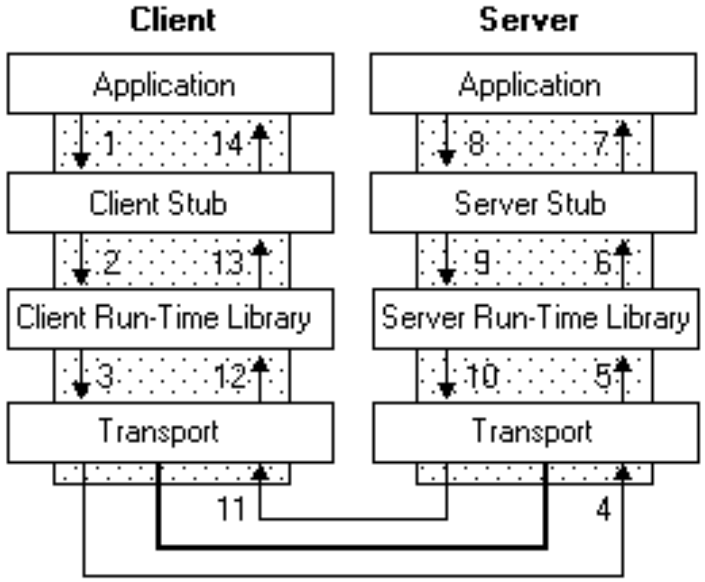
- Microsoft RPC Foundations
- Vulnerability Details (geek speak)
- Investigating MSRPC
- Protections/Mitigations
  - Network Segmentation
  - Host-based Firewall
  - RPC Filtering
  - Enabling RPC Logging
- Closing Thoughts

References

- General RPC Troubleshooting Information
  - https://social.technet.microsoft.com/wiki/contents/articles/4494.windows-server-troubleshooting-rpc-server-is-unavailable.aspx
- SpectreOps RPC Detection Engineering
  - https://www.specterops.io/assets/resources/RPC_for_Detection_Engineers.pdf
- Akamai RPC Runtime Vulnerability Analysis
  - https://www.akamai.com/blog/security/critical-remote-code-execution-vulnerabilities-windows-rpc-runtime

Microsoft RPC (MSRPC) Foundations

- MSRPC was designed to allow a client to transparently call a function on a remote system (or in some cases, another process on the same system) without having to understand the underlying transport mechanisms
- RPC is incredibly complex and underpins many operations we take for granted in Windows



https://docs.microsoft.com/en-us/windows/win32/rpc/how-rpc-works

Microsoft RPC Ports

- TCP Port 135 - RPC Endpoint Mapper (technically can use UDP)
- TCP Port 139/445 - MSRPC over SMB
- TCP Port 593 - RPC Endpoint Mapper over HTTP
- TCP Port 1025-5000 - Ephemeral ports, 2k/2k3/XP, may include RPC services
- TCP Port 49152-65535 - Ephemeral ports, Vista/2k8+, may include RPC services
- TCP Port 5722 - DFS Replication (2k8/2k8R2 only)
- TCP Port 2103/2105 - RPC message queuing (MSMQ)

Microsoft RPC Ports On Shodan

- Per Johannes Ullrich at the SANS Internet Storm Center, as of yesterday morning here are the counts for publicly exposed MS RPC services on well-known ports:
  - Port 135: 2,180,387 hosts
  - Port 445: 1,363,008 (Jeff McJunkin identified 3,086,523 in his scans)
  - Port 593: 12,594 hosts
- Note: many of these may not be Windows (likely Samba) and some are likely to be honeypots

Microsoft RPC Portmapper

- The Microsoft RPC endpoint port mapper listens on TCP port 135
- When a remote machine wants to communicate with an RPC service, it must first contact the RPC port mapper (EPMAP)
- The EPMAP service returns an IP address and port pairing for the service to contact
- The remote client then contacts the host on the IP and port returned by the endpoint port mapper

MSRPC EPMAP Network Traffic

- Let's take a look at a remote service creation call
  - **Note:** the session to 192.168.132.132 was already established prior to this command being issued or it would have failed

```
C:\Users\malwa>
C:\Users\malwa>sc \\192.168.132.132 create newserv2 binpath=c:\windows\system32\notepad.exe
[SC] CreateService SUCCESS
```

- If you want to play along at home, grab the packet capture:
  - https://github.com/malwarejake-public/packets

MSRPC EPMAP Network Traffic (2)

SANS Analyst Program

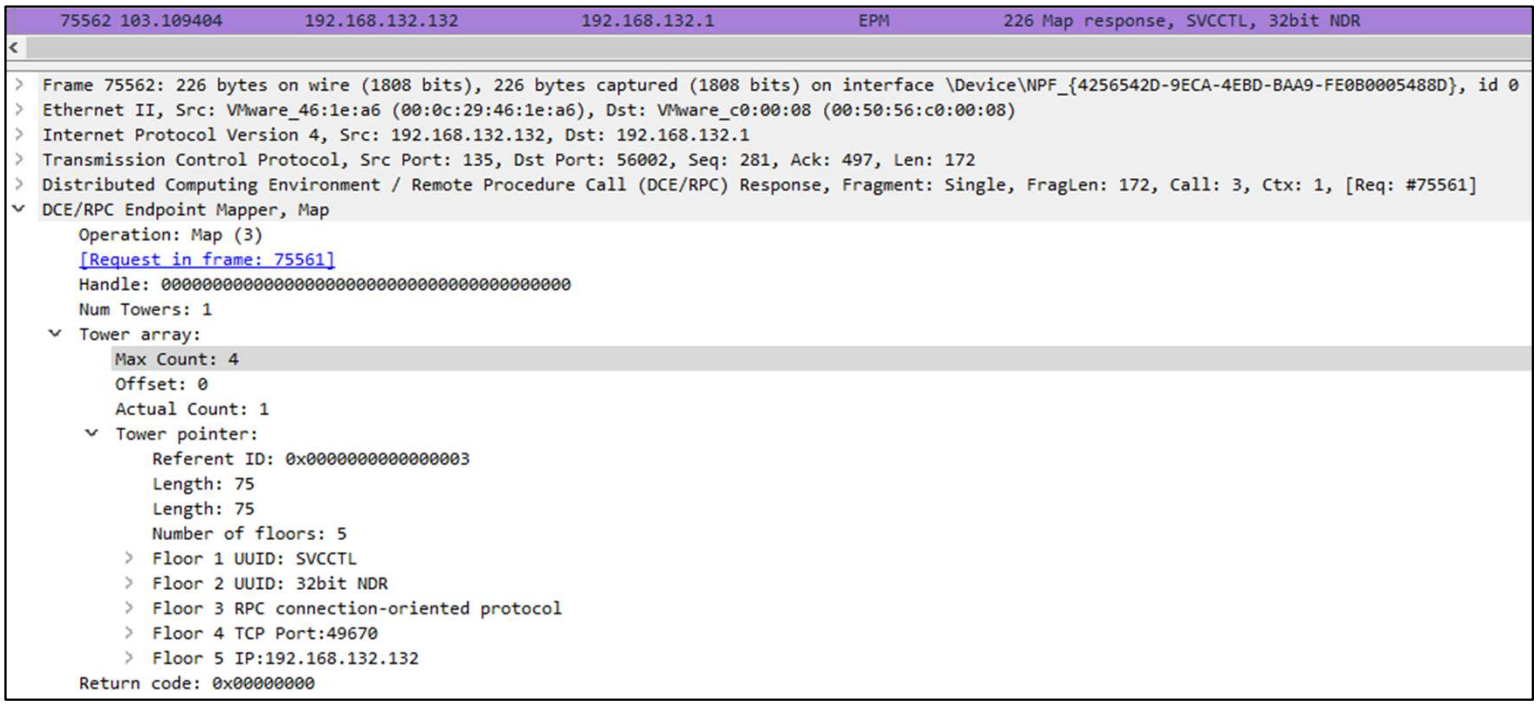
- First, we see the connection to the EPMAP requesting the endpoint for SVCCTL (service controller)

```
75561 103.109156    192.168.132.1    192.168.132.132    EPM    222 Map request, SVCCTL, 32bit NDR
<
> Frame 75561: 222 bytes on wire (1776 bits), 222 bytes captured (1776 bits) on interface \Device\NPF_{4256542D-9ECA-4EBD-BAA9-FE0B0005488D}, id 0
> Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_46:1e:a6 (00:0c:29:46:1e:a6)
> Internet Protocol Version 4, Src: 192.168.132.1, Dst: 192.168.132.132
> Transmission Control Protocol, Src Port: 56002, Dst Port: 135, Seq: 329, Ack: 281, Len: 168
> Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 168, Call: 3, Ctx: 1, [Resp: #75562]
v DCE/RPC Endpoint Mapper, Map
    Operation: Map (3)
    [Response in frame: 75562]
  v UUID pointer:
      Referent ID: 0x0000000000000001
      UUID: 00000000-0000-0000-0000-000000000000
  v Tower pointer:
      Referent ID: 0x0000000000000002
      Length: 75
      Length: 75
      Number of floors: 5
    > Floor 1 UUID: SVCCTL
    > Floor 2 UUID: 32bit NDR
    > Floor 3 RPC connection-oriented protocol
    > Floor 4 TCP Port:135
    > Floor 5 IP:192.168.132.132
    Handle: 00000000000000000000000000000000000000000000
    Max Towers: 4
```

MSRPC EPMAP Network Traffic (3)

SANS Analyst Program

- The EPMAP responds, telling the remote system that the SVCCTL RPC endpoint is listening on TCP port 49670 on 192.168.132.132



MSRPC EPMAP Network Traffic (3)

SANS Analyst Program

- The client connects to the SVCCTL service on TCP 49670
  - Note: this port may be different on your system, that's what the EPMAP service is for!

```
   75566 103.111368      192.168.132.1          192.168.132.132          DCERPC      218 Bind: call_id: 2, Fragment: Single, 2 context it
<
> Frame 75566: 218 bytes on wire (1744 bits), 218 bytes captured (1744 bits) on interface \Device\NPF_{4256542D-9ECA-4EBD-BAA9-FE0B0005488D}, id 0
> Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_46:1e:a6 (00:0c:29:46:1e:a6)
> Internet Protocol Version 4, Src: 192.168.132.1, Dst: 192.168.132.132
> Transmission Control Protocol, Src Port: 56005, Dst Port: 49670, Seq: 1, Ack: 1, Len: 164
v Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind, Fragment: Single, FragLen: 164, Call: 2
    Version: 5
    Version (minor): 0
    Packet type: Bind (11)
  > Packet Flags: 0x07
  > Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE)
    Frag Length: 164
    Auth Length: 40
    Call ID: 2
    Max Xmit Frag: 5840
    Max Recv Frag: 5840
    Assoc Group: 0x00000000
    Num Ctx Items: 2
  > Ctx Item[1]: Context ID:0, SVCCTL, 32bit NDR
  > Ctx Item[2]: Context ID:1, SVCCTL, Bind Time Feature Negotiation
  > Auth Info: NTLMSSP, Packet privacy, AuthContextId(0)
```

MSRPC EPMAP Network Traffic (4)

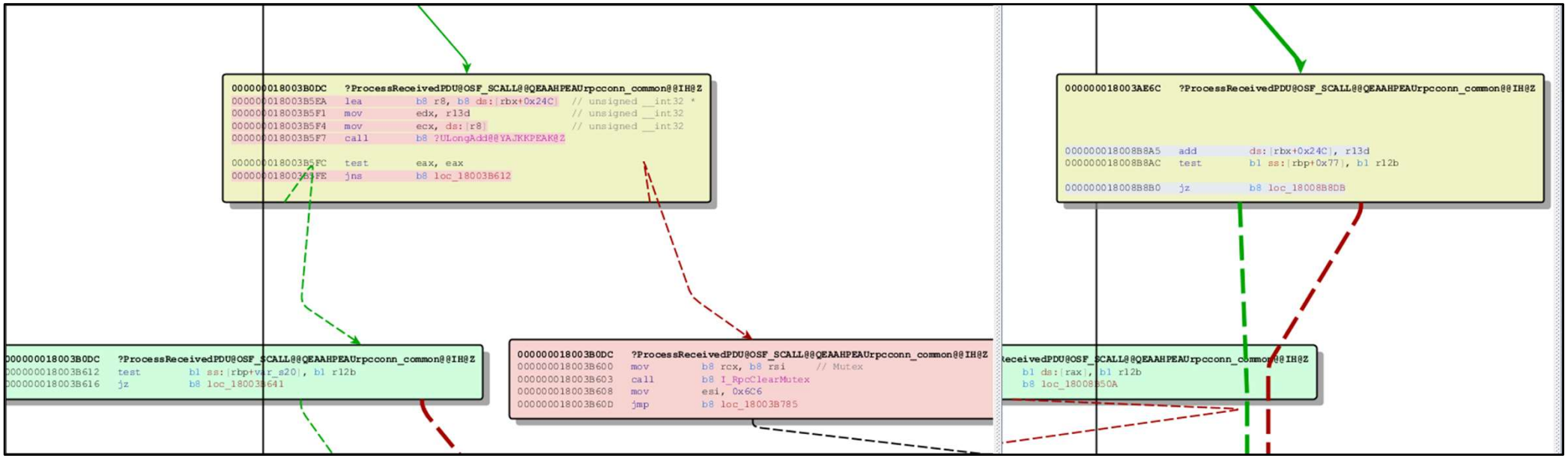
- The client is redirected to SMB to complete the request using RPC over SMB (note the call to OpenSCManagerW below)

```
75582 103.113912       192.168.132.1          192.168.132.132          SVCCTL        262 OpenSCManagerW request, \\192.168.132.132
<

        Chain Offset: 0x00000000
        Message ID: 18
        Process Id: 0x0000feff
      > Tree Id: 0x00000001  \\192.168.132.132\IPC$
      > Session Id: 0x0000240000000001 Acct:scythe Domain: Host:FANCIEST-BEAR
        Signature: 00000000000000000000000000000000
        [Response in: 75583]
    > Ioctl Request (0x0b)
> Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 84, Call: 3, Ctx: 0, [Resp: #75583]
v Microsoft Service Control, OpenSCManagerW
    Operation: OpenSCManagerW (15)
    [Response in frame: 75583]
  v MachineName: \\192.168.132.132
        Referent ID: 0x00020000
        Max Count: 18
        Offset: 0
        Actual Count: 18
        MachineName: \\192.168.132.132
    NULL Pointer: Database
  > Access Mask: 0x00000002
```

Vulnerability Details

SANS Analyst Program

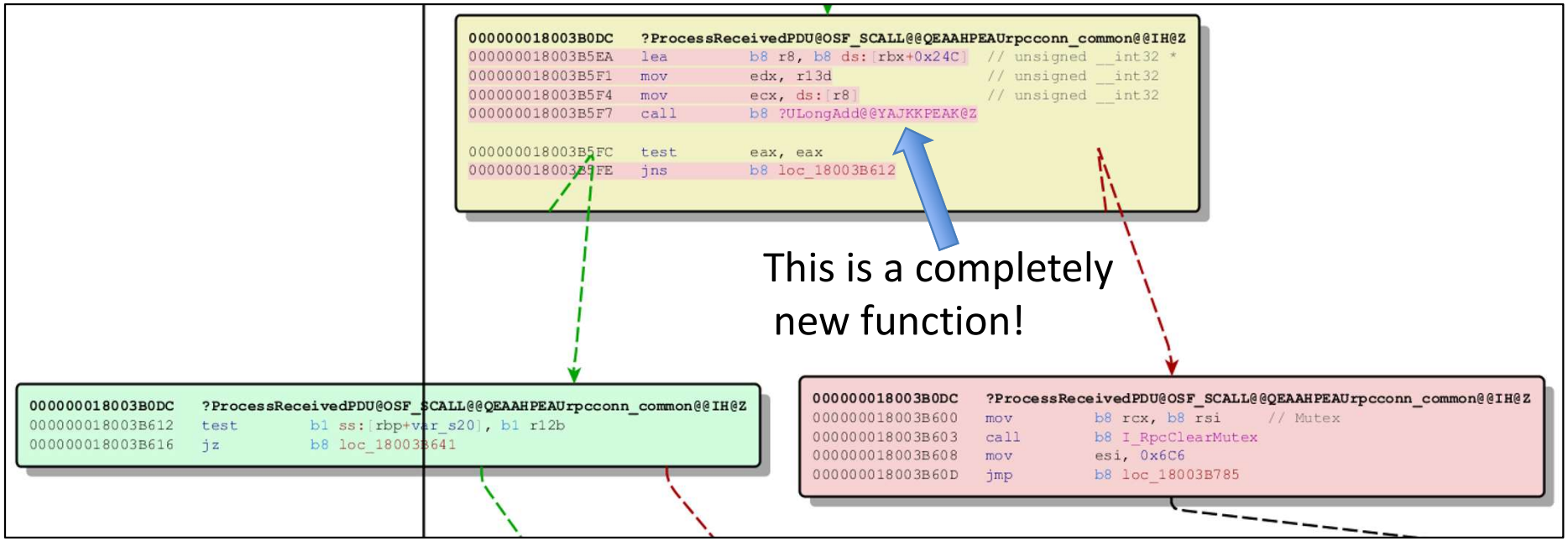
- As first published by Akamai, the changes in the code are in RPCrt4.dll and involve both RPC client and server functions
  - Here, we examined ProcessReceivedPDU (server-side code)



Vulnerability Details (2)

SANS Analyst Program

- Note the new function ULongAdd that is called in the patched version of the code



Vulnerability Details (3)

SANS Analyst Program

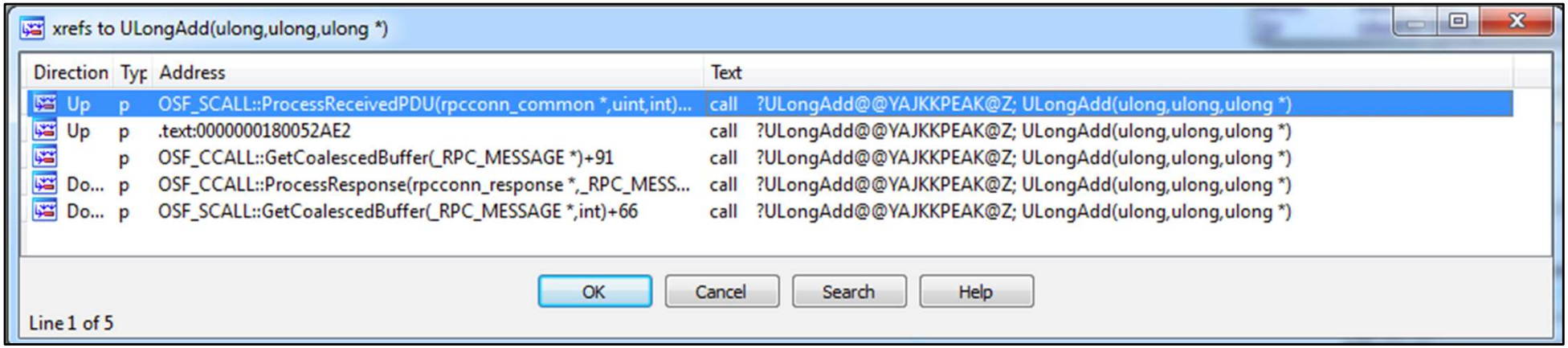
- The new function checks to ensure that an integer overflow caused by remotely supplied parameters has not occurred

```
; __int32 __stdcall ULongAdd(unsigned __int32, unsigned __int32, unsigned __int32 *)
?ULongAdd@@YAJKKPEAK@Z proc near
lea     eax, [rcx+rdx]
or      edx, 0FFFFFFFFh
cmp     eax, ecx
cmovnb  edx, eax
sbb     eax, eax
and     eax, 80070216h
mov     [r8], edx
retn
?ULongAdd@@YAJKKPEAK@Z endp
```

Vulnerability Details (4)

SANS Analyst Program

- The new function is called from multiple locations, including:
  - OSF_SCALL:ProcessReceivedPDU (server code, already examined)
  - OSF_SCALL:GetCoalescedBuffer (server code)
  - OSF_CCALL:GetCoalescedBuffer (client code)
  - OSF_CCALL:ProcessResponse (client code)



Vulnerability Details (5)

- Microsoft rated the server vulnerability as "exploitation more likely" and the client vulnerabilities as "exploitation less likely"
- Examining the code, the integer overflow conditions appear identical between the server and client functions
- It is reasonable to assume that exploitation difficulty is identical for server vs client code and the difference in difficulty rating is due to remotely triggerable vs relying on user interaction
- **Key takeaway:** just blocking inbound RPC will likely not be enough to prevent exploitation

Vulnerability Details (6)

SANS Analyst Program

Per Antonio: The vulnerable function can be achieved on port 445 via named pipe if the RPC server is configured with ncacn_np and on a custom TCP port (e.g. 9999) if it's configured with ncacn_ip_tcp.

Antonio Cocomazzi
@splinter_code

Finally! I have found the right conditions to hit the vulnerable function for CVE-2022-26809!
No panic, this is a custom RPC server i wrote, not a default Windows service
It seems it's required a specific RPC configuration and AFAIK it shouldn't be common, need to deepen more...

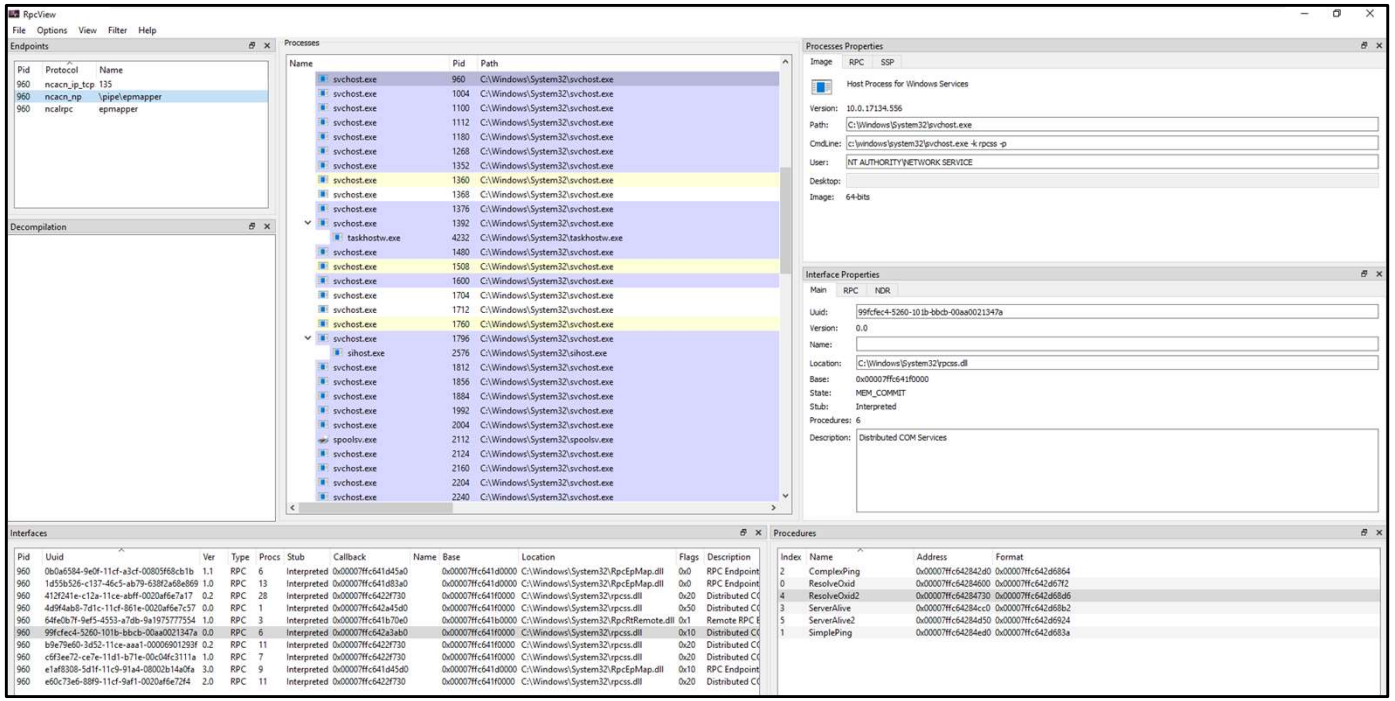
```
*GetCoal*
RPCRT4!OSF_SCALL::GetCoalescedBuffer (<no parameter in
RPCRT4!OSF_CCALL::GetCoalescedBuffer (<no parameter in
a`579bd11c

::GetCoalescedBuffer:
4889542410      mov     qword ptr [rsp+10h],rdx ss:000

    RetAddr           Call Site
a8b8 00007ffa`579bd77a RPCRT4!OSF_SCALL::GetCoalescedBuf
a8c0 00007ffa`5794696d RPCRT4!OSF_SCALL::Receive+0x9a
a8f0 00007ffa`5795b16b RPCRT4!I_RpcReceive+0x6d
a920 00007ffa`579dafdb RPCRT4!NdrReceive+0x53
a950 00007ffa`579dbd70 RPCRT4!NdrPartialReceive+0x47
a980 00007ffa`579db2dd RPCRT4!NdrReadPipeElements+0x1b4
```

1:16 PM · Apr 14, 2022 · Twitter Web App

Investigating with RPCview.exe

SANS Analyst Program

- RPCview is an open source tool for examining and investigating RPC endpoints on a live machine



Investigating with RPCview.exe (2)

- The RPC endpoints on a machine are shown in the endpoints box

Endpoints

Pid	Protocol	Name
656	ncacn_ip_tcp	49674
960	ncacn_ip_tcp	135
1112	ncacn_ip_tcp	49665
1392	ncacn_ip_tcp	49666
2112	ncacn_ip_tcp	49667
2424	ncacn_ip_tcp	49668
656	ncacn_np	\pipe\lsass
960	ncacn_np	\pipe\epmapper
1004	ncacn_np	\pipe\LSM_API_service
1112	ncacn_np	\pipe\eventlog
1392	ncacn_np	\PIPE\atsvc
2112	ncacn_np	\pipe\spoolss
2240	ncacn_np	\PIPE\wkssvc
2600	ncacn_np	\PIPE\srvsvc
2692	ncacn_np	\pipe\trkwks
640	ncalrpc	WMsgKRpc07E911
656	ncalrpc	audit
656	ncalrpc	securityevent
656	ncalrpc	LSARPC_ENDPOINT
656	ncalrpc	lsacap
656	ncalrpc	LSA_IDPEXT_ENDPOINT

Investigating with RPCview.exe (3)

- By selecting an RPC endpoint, the interfaces (including the UUID) for that endpoint are shown

Interfaces

Pid	Uuid	Ver	Type	Procs	Stub	Callback	Name	Base	Location	Flags	Description	EpMapper
960	0b0a6584-9e0f-11cf-a3cf-00805f68cb1b	1.1	RPC	6	Interpreted	0x00007ffc641d45a0		0x00007ffc641d0000	C:\Windows\System32\RpcEpMap.dll	0x0	RPC Endpoint Mapper	
960	1d55b526-c137-46c5-ab79-638f2a68e869	1.0	RPC	13	Interpreted	0x00007ffc641d83a0		0x00007ffc641d0000	C:\Windows\System32\RpcEpMap.dll	0x0	RPC Endpoint Mapper	
960	412f241e-c12a-11ce-abff-0020af6e7a17	0.2	RPC	28	Interpreted	0x00007ffc6422f730		0x00007ffc641f0000	C:\Windows\System32\rpcss.dll	0x20	Distributed COM Services	
960	4d9f4ab8-7d1c-11cf-861e-0020af6e7c57	0.0	RPC	1	Interpreted	0x00007ffc642a45d0		0x00007ffc641f0000	C:\Windows\System32\rpcss.dll	0x50	Distributed COM Services	
960	64fe0b7f-9ef5-4553-a7db-9a1975777554	1.0	RPC	3	Interpreted	0x00007ffc641b70e0		0x00007ffc641b0000	C:\Windows\System32\RpcRtRemote.dll	0x1	Remote RPC Extension	
960	99fcfec4-5260-101b-bbcb-00aa0021347a	0.0	RPC	6	Interpreted	0x00007ffc642a3ab0		0x00007ffc641f0000	C:\Windows\System32\rpcss.dll	0x10	Distributed COM Services	
960	b9e79e60-3d52-11ce-aaa1-00006901293f	0.2	RPC	11	Interpreted	0x00007ffc6422f730		0x00007ffc641f0000	C:\Windows\System32\rpcss.dll	0x20	Distributed COM Services	
960	c6f3ee72-ce7e-11d1-b71e-00c04fc3111a	1.0	RPC	7	Interpreted	0x00007ffc6422f730		0x00007ffc641f0000	C:\Windows\System32\rpcss.dll	0x20	Distributed COM Services	
960	e1af8308-5d1f-11c9-91a4-08002b14a0fa	3.0	RPC	9	Interpreted	0x00007ffc641d45d0		0x00007ffc641d0000	C:\Windows\System32\RpcEpMap.dll	0x10	RPC Endpoint Mapper	
960	e60c73e6-88f9-11cf-9af1-0020af6e72f4	2.0	RPC	11	Interpreted	0x00007ffc6422f730		0x00007ffc641f0000	C:\Windows\System32\rpcss.dll	0x20	Distributed COM Services	

Investigating with RPCview.exe (4)

- By selecting an RPC interface, the available procedures are shown

- **Note:** you will need to install debugging tools and download symbols to see similar function names

Procedures

Index	Name	Address	Format
17	ControlTracingForProcess	0x00007ffc642a5d90	0x00007ffc642d6e46
20	DecodeProxy	0x00007ffc642a5e80	0x00007ffc642d6ef4
8	EnableDisableDynamicIPTracking	0x00007ffc642a63c0	0x00007ffc642d6c30
11	FlushSCMBindings	0x00007ffc642a6450	0x00007ffc642d6cde
9	GetCurrentAddrExclusionList	0x00007ffc642a64a0	0x00007ffc642d6c66
4	GetThreadID	0x00007ffc64240070	0x00007ffc642d6b3a
7	GetWindowPropInterface	0x00007ffc6423da90	0x00007ffc642d6be2
16	IsObjectCreationAllowed	0x00007ffc642a6550	0x00007ffc642d6df2
27	NotifyComClassChangesFromDeployment	0x00007ffc642a6920	0x00007ffc642d709e
13	NotifyDDStartOrStop	0x00007ffc642a69c0	0x00007ffc642d6d4a
21	NotifyPsmResume	0x00007ffc641f14a0	0x00007ffc642d6f42
19	NotifyWinRTActivationStoreChanged	0x00007ffc642a6a20	0x00007ffc642d6ec4
14	QueryDragDropActive	0x00007ffc642a6ac0	0x00007ffc642d6d80
18	QueryPIDForActivation	0x00007ffc642a6b20	0x00007ffc642d6e82
22	QueryServerProcessHandleHeld	0x00007ffc6423fef0	0x00007ffc642d6f72
25	RegisterConsoleHandles	0x00007ffc642a6bf0	0x00007ffc642d7020
23	RegisterRacActivationToken	0x00007ffc6423faa0	0x00007ffc642d6fae
6	RegisterWindowPropInterface	0x00007ffc6423db60	0x00007ffc642d6b94
12	RetireServer	0x00007ffc642a6ca0	0x00007ffc642d6d14
26	RevokeConsoleHandles	0x00007ffc642a6da0	0x00007ffc642d7068
24	RevokeRacActivationToken	0x00007ffc6423fd00	0x00007ffc642d6fea
2	ServerRegisterActivatableClasses	0x00007ffc6420f4e0	0x00007ffc642d6ace
0	ServerRegisterClsid	0x00007ffc64236ce0	0x00007ffc642d6a62
3	ServerRevokeActivatableClasses	0x00007ffc641f7800	0x00007ffc642d6b0a
1	ServerRevokeClsid	0x00007ffc642376d0	0x00007ffc642d6a9e
10	SetAddrExclusionList	0x00007ffc642a6e30	0x00007ffc642d6ca2
15	SetOrRevokeForcedDropTarget	0x00007ffc642a6ee0	0x00007ffc642d6db6
5	UpdateActivationSettings	0x00007ffc642a6f60	0x00007ffc642d6b6a

(Potential) Protections and Mitigations

- Let's discuss some (potential) mitigations for MS RPC vulns:
  - Network segmentation
  - Host-based firewalls
  - Egress firewalls / Service edge filtering
  - UNC Hardening
  - RPC Filtering
  - Enable RPC Logging
  - LAPS and PAM
  - Zeek RPC Logging

Network Segmentation

- Because many systems use RPC in production, network segmentation won't prevent all exploitation
- However, in most networks there will be fairly limited RPC communications to most workstations
  - Some third-party software installs will change this
- Recommendations:
  - Block communications between workstation subnets
  - Block RPC and SMB comms from server and workstation subnets to the extent possible
  - Deploy private VLANs (yes, I know this is hard)

Host-based Firewall

SANS Analyst Program

- Host-based firewalls are exceedingly effective in blocking inbound RPC communications over TCP port 135
  - However, named pipes can be used for exploitation over TCP 445
- There is also a possibility that a threat actor could connect to and exploit an RPC service listening on an ephemeral port
- This would require scanning and likely a bit of luck
- Following zero-trust networking principles of "deny all, permit by exception" is the best plan here

Egress Firewall/Service Edge Filtering

- Configure edge firewalls to limit **outbound** traffic on any ports used by Microsoft RPC
  - There are client-side vulnerabilities in the recent RPC patch too
- Blocking standard RPC ports outbound at the service edge/egress firewall will prevent a user (or application) from being tricked into connecting to a malicious RPC server
  - **Note:** Because most networks do not allow RPC or SMB inbound, it is possible this may be used by threat actors through phishing delivery

UNC Hardening

- Ned Pyle from Microsoft has been recommending UNC hardening for a long time and this article has the steps to do it
  - https://support.microsoft.com/en-us/topic/ms15-011-vulnerability-in-group-policy-could-allow-remote-code-execution-february-10-2015-91b4bda2-945d-455b-ebbb-01d1ec191328
- It's currently not clear if the vulnerable RPC client code can be accessed through a UNC path (though this seems likely)
  - However, we recommend you use this event to get traction on implementing this hardening
  - The hardening does mitigate many other attack vectors, some of which threat actors use in lateral movement operations

RPC Filtering

- Researchers Ophir Harpaz and Stiv Kupchik at Akamai have published the definitive guide on RPC filtering
  - https://www.akamai.com/blog/security/guide-rpc-filter
- With RPC filtering, you can create extremely granular access to RPC procedures, including facilitating more advanced logging

```
rpc filter
add rule layer=um actiontype=block
add condition field=if_uuid matchtype=equal data=f6beaff7-1e19-4fbb-9f8f-b89e2018337c
add filter
```

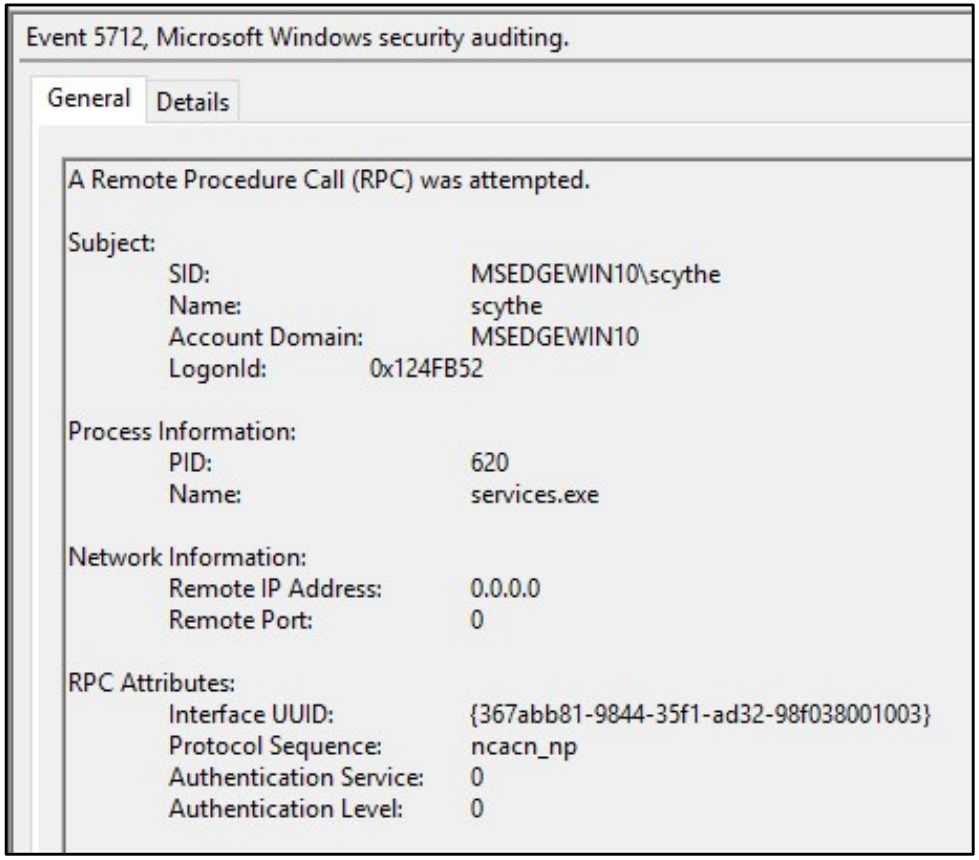
- It's unclear whether RPC filters can prevent exploitation of this vulnerability, but more eyes will certainly be on RPC code now

Enabling RPC Logging

- Jonathan Johnson from Red Canary has a repo with multiple RPC filter examples we can use to bootstrap logging
  - https://github.com/jsecurity101/MSRPC-to-ATTACK
- First enable auditing on RPC
  - **auditpol /set /subcategory:"RPC Events" /success:enable /failure:enable**
- Then create a filter script (this is for remote services)

```
# Remote Services Creation
rpc
filter
add rule layer=um actiontype=permit audit=enable
add condition field=if_uuid matchtype=equal data=367ABB81-9844-35F1-AD32-98F038001003
add filter
quit
```

Enabling RPC Logging (2)

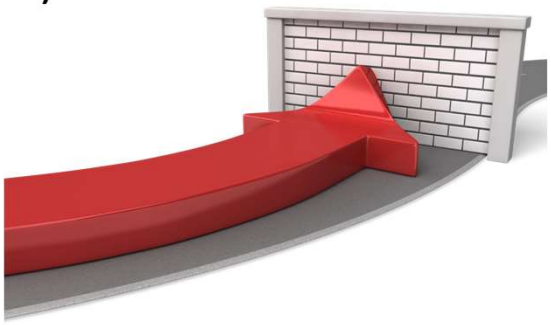


- While the name of the specific service created wasn't logged here, we can conclusively tie the service creation to a user and a login ID
- Look for an identically timed 7045 in the System event log

LAPS / PAM

SANS Analyst Program

- Successful exploitation will result in full system access
- Threat actors will be able to pilfer credentials, including:
  - Local account hashes
  - LSA Secrets (potentially including plaintext passwords)
  - Cached domain credentials
  - Contents of LSASS memory dumps
  - Hardcoded passwords in config files, scripts, etc.
- Seriously consider deploying LAPS
  - Or at a minimum, threat model the use of easily discovered credentials to hunt lateral movement from compromised endpoints



Zeek/Bro MS RPC Logging

- Zeek is capable of logging MS RPC operations at a granular level
  - Note that per documentation, there are some high-throughput RPC functions that are not logged by Zeek
  - https://docs.zeek.org/en/v3.0.14/scripts/base/protocols/dce-rpc/main.zeek.html


```
{"ts":1650033722.063583,"uid":"CuB9Ty30evElf6d5F5","id.orig_h":"192.168.134.110","id.orig_p":58465,"id.resp_h":"192.168.134.100","id.resp_p":49668,"rtt":0.000268220901
49668","endpoint":"drsuapi","operation":"DRSCrackNames"}
{"ts":1650033722.064062,"uid":"CuB9Ty30evElf6d5F5","id.orig_h":"192.168.134.110","id.orig_p":58465,"id.resp_h":"192.168.134.100","id.resp_p":49668,"rtt":0.000104904174
49668","endpoint":"drsuapi","operation":"DRSUnbind"}
{"ts":1650033914.219139,"uid":"CnaGwC4qwJUrEoEMu1","id.orig_h":"192.168.69.110","id.orig_p":54521,"id.resp_h":"192.168.69.100","id.resp_p":135,"rtt":0.0003538131713867
","endpoint":"epmapper","operation":"ept_map"}
{"ts":1650033935.210161,"uid":"CZQMTo4sw4mZgXG3G7","id.orig_h":"192.168.69.110","id.orig_p":54520,"id.resp_h":"192.168.69.100","id.resp_p":445,"rtt":0.0006070137023925
pe\\ntsvcs","endpoint":"svcctl","operation":"unknown-64"}
{"ts":1650033935.211333,"uid":"CZQMTo4sw4mZgXG3G7","id.orig_h":"192.168.69.110","id.orig_p":54520,"id.resp_h":"192.168.69.100","id.resp_p":445,"rtt":0.0004599094390869
ipe\\ntsvcs","endpoint":"svcctl","operation":"CloseServiceHandle"}
{"ts":1650033935.213731,"uid":"CnaGwC4qwJUrEoEMu1","id.orig_h":"192.168.69.110","id.orig_p":54521,"id.resp_h":"192.168.69.100","id.resp_p":135,"rtt":0.0002138614654541
","endpoint":"epmapper","operation":"ept_map"}
{"ts":1650033956.208333,"uid":"CZQMTo4sw4mZgXG3G7","id.orig_h":"192.168.69.110","id.orig_p":54520,"id.resp_h":"192.168.69.100","id.resp_p":445,"rtt":0.0002908706665039
ipe\\ntsvcs","endpoint":"svcctl","operation":"unknown-64"}
{"ts":1650033956.210231,"uid":"CZQMTo4sw4mZgXG3G7","id.orig_h":"192.168.69.110","id.orig_p":54520,"id.resp_h":"192.168.69.100","id.resp_p":445,"rtt":0.0024888515472412
e\\ntsvcs","endpoint":"svcctl","operation":"CreateServiceW"}
{"ts":1650033956.213354,"uid":"CZQMTo4sw4mZgXG3G7","id.orig_h":"192.168.69.110","id.orig_p":54520,"id.resp_h":"192.168.69.100","id.resp_p":445,"rtt":0.0005738735198974
pe\\ntsvcs","endpoint":"svcctl","operation":"CloseServiceHandle"}
{"ts":1650033956.21404,"uid":"CZQMTo4sw4mZgXG3G7","id.orig_h":"192.168.69.110","id.orig_p":54520,"id.resp_h":"192.168.69.100","id.resp_p":445,"rtt":0.00036001205444335
e\\ntsvcs","endpoint":"svcctl","operation":"CloseServiceHandle"}
```

Closing Thoughts

SANS Analyst Program

- While there is no POC available, we believe one is imminent (likely in the next seven days)

- Blocking inbound 135/445/593 by itself will not prevent exploitation because both client AND server RPC code is vulnerable

- If patching isn't an option, explore RPC filtering to potentially gain additional telemetry