A Voyage to Uncovering RPC Telemetry

By: Jonathan Johnson (@jsecurity101)

> whoami

- Associate Consultant – Detection Engineering Team
- Open-Source Project Creator/Contributor
- Security Researcher
  - Github: https://github.com/jsecurity101
  - Blog: https://medium.com/@jsecurity101
  - Twitter: https://twitter.com/jsecurity101
- Never an expert, always a student

Acknowledgements

For helping with research:

- Jared Atkinson, SpecterOps
- Lee Christensen, SpecterOps

This research was inspired by and would not be possible without the incredible work performed by the following individuals:

- Matt Graeber, Red Canary
- Matt Hand, SpecterOps
- James Forshaw, Google

Content

- The Journey – How it all began
  - Capability Abstraction
  - Identifying Unknowns
- Remote Procedure Call (RPC)
  - RPC Basics
  - RPC Components
  - RPC Process
- Leveraging RPC Telemetry
  - How can Detection Engineers use this data
  - Applying Knowledge
  - Research Telemetry -> Scalable Telemetry
- Conclusion

The Journey – How it all began

Capability Abstraction

- Concept introduced by Jared Atkinson in February 2020 (https://posts.specterops.io/capability-abstraction-fbeaeeb26384)
- Extract technology layers
- Identify pivot points for Defenders

Capability Abstraction (cont.d)

T1003 – CREDENTIAL DUMPING – DCSync (Pre-RPC Research)			
Tools	**Mimikatz** **lsadump::dcsync**	**Empire** **Invoke-DCSync**	**Impacket** **secretdump.py**
Extended Rights	0x100 - **Control Access** {19195a5b-6da0–11d0-afd3–00c04fd930c9} — **Domain-DNS Class(Object)** {1131f6ad-9c07–11d1-f79f-00c04fc2dcd2}- **DS-Replication-Get-Changes-All(Extended Right)**		
RPC Protocol	**Directory Replication Service**		
RPC Interface	**DRSUAPI** (e3514235-4b06-11d1-ab04-00c04fc2dcd2) C:\Windows\System32\ntdsai.dll		
RPC Method	**GetNCChanges REQ/REPLY**		
Behavior	**Replication of NC Schema**		

???

???

???

???

Identifying Unknowns

- When the network connection was made, what process(es) made a connection to each other? (Client/Server processes)
- How were the credentials being brought back to the client?
- Outside of network traffic, is there visibility into RPC?
- What did RPC do *exactly?*

Remote Procedure Call (RPC)

RPC Basics

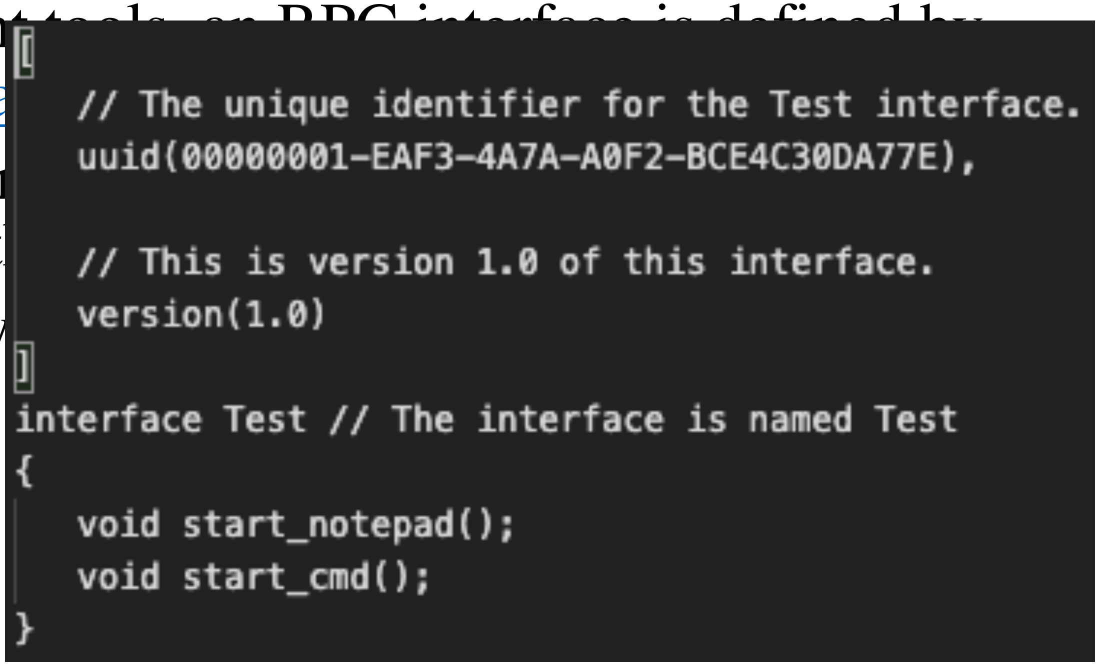
- A technology used for distributed client/server communications between programs
- Allows applications to send signals to each other to perform an operation
- RPC is used for everyday procedures that happen within Windows environments ranging from authentication, service creation, directory replication, and more
- Will focus on the Microsoft RPC (MSRPC) implementation and its supporting development tools NOT the low-level protocol implementation details (i.e Impacket/NtObjectManager)

RPC Components

- RPC Protocol
  - Microsoft supports “service based” protocols by default on Windows
  - These services can be thought of as “protocols”
    - Directory Replication (DRS)/Service Control Manager (SMCR)/Print System (RPRN)
- RPC Client/Server
  - All the code needed to interact with a Microsoft supported RPC Protocol is pre-compiled and stored within the RPC server
  - Can be stored in EXE, SYS, DLL binaries
    - Application is not the “server” or “client”, the application holds the code for the “server” and “client”.

RPC Components (cont.d)

- RPC Interface
  - When using Microsoft’s development tools, an RPC interface is defined by the Microsoft Interface Definition La
  - IDL file includes what protocol the i and their parameters that interact wit
  - Each interface is tied to a universally or 16 bytes.



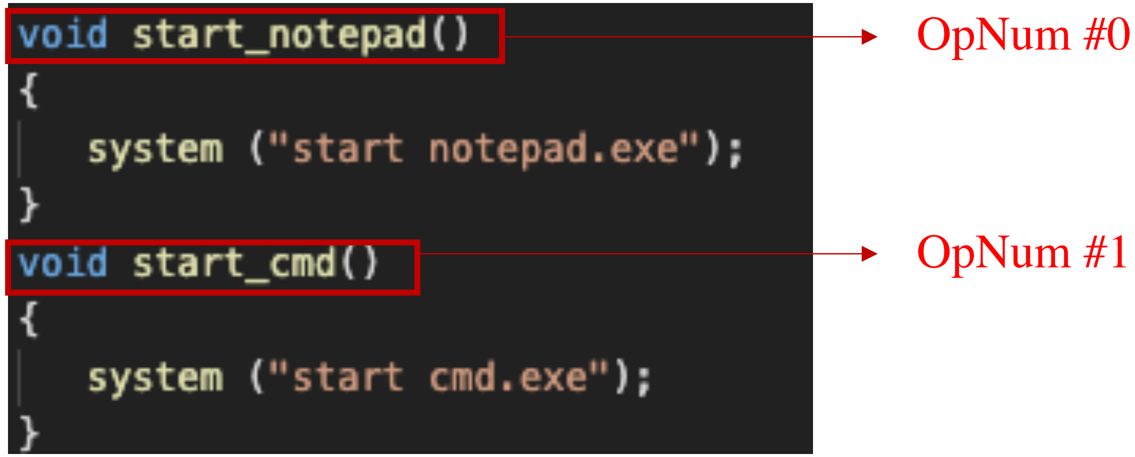
```
[
    // The unique identifier for the Test interface.
    uuid(00000001-EAF3-4A7A-A0F2-BCE4C30DA77E),

    // This is version 1.0 of this interface.
    version(1.0)
]
interface Test // The interface is named Test
{
    void start_notepad();
    void start_cmd();
}
```

RPC Components (cont.d)

- RPC Interface
  - An RPC client code calls a Win32 API that will implement an RPC interface. This can be seen inside of native Windows binaries.
  - An RPC client contains the necessary IDL (Interface Definition Language) code baked in so that it can talk to the RPC server. An example of this can be found within Mimikatz code.
  - An RPC client will talk to the RPC server directly by implementing the RPC over TCP/IP or RPC over named pipe protocols and will not interface with the client's OS's RPC runtime.

RPC Components (cont.d)

- RPC Method(s)
  - Methods are functions that the RPC server exposes to perform a specific behavior
  - Each RPC method is identified by an OpNum (Operation Number)



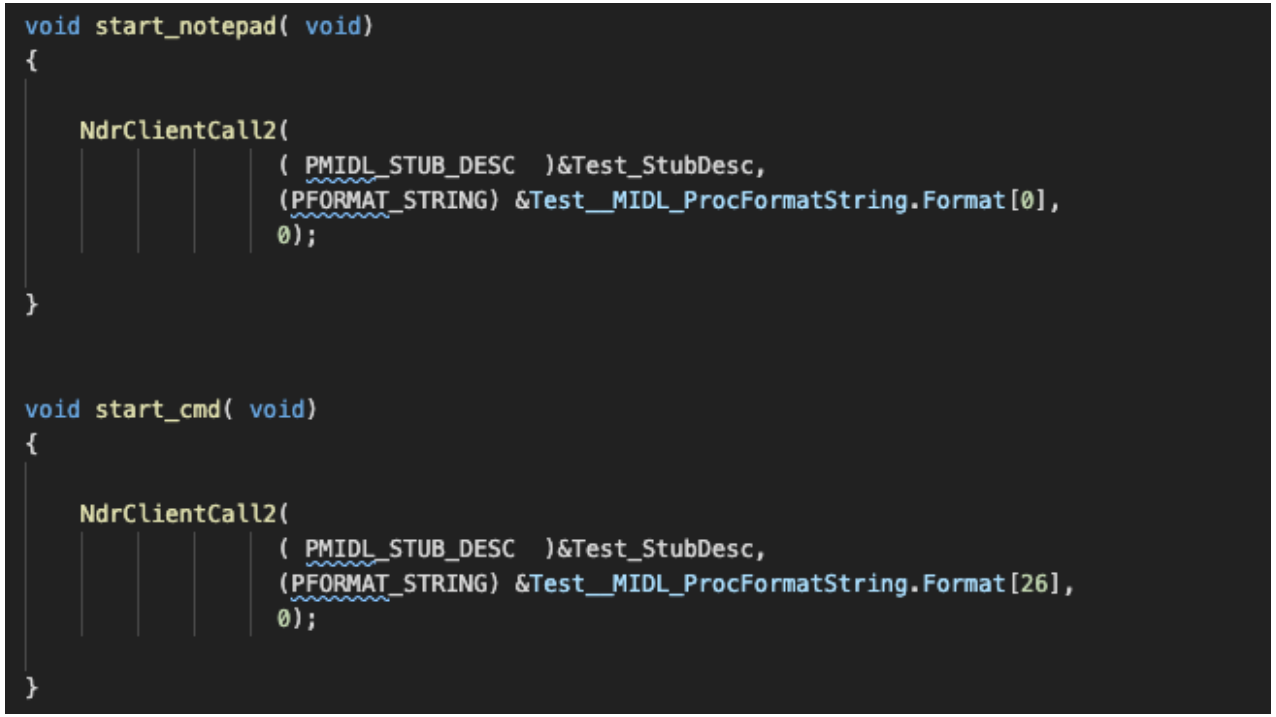
```
void start_notepad()
{
    system ("start notepad.exe");
}
void start_cmd()
{
    system ("start cmd.exe");
}
```

OpNum #0

OpNum #1

RPC Components (cont.d)

- Client/Server stubs
  - Used to serialize/deserialize the parameters being passed to the method
  - Interface with Windows's RPC runtime to send/receive data over a transport



```
void start_notepad( void)
{

    NdrClientCall2(
                  ( PMIDL_STUB_DESC  )&Test_StubDesc,
                  (PFORMAT_STRING) &Test__MIDL_ProcFormatString.Format[0],
                  0);

}


void start_cmd( void)
{

    NdrClientCall2(
                  ( PMIDL_STUB_DESC  )&Test_StubDesc,
                  (PFORMAT_STRING) &Test__MIDL_ProcFormatString.Format[26],
                  0);

}
```

RPC Components (cont.d)

- NDR Engine/Marshalling
  - Responsible for the marshalling of DCOM & RPC components
- RPC Runtime
  - RPC runtime holds the operating system’s core RPC services
    - RPC Endpoint Mapper
  - Responsible for the transportation of the serialized parameters from the client stub to the server stub
  - Code can be found in the Rpcrt4.dll

RPC Components (cont.d)

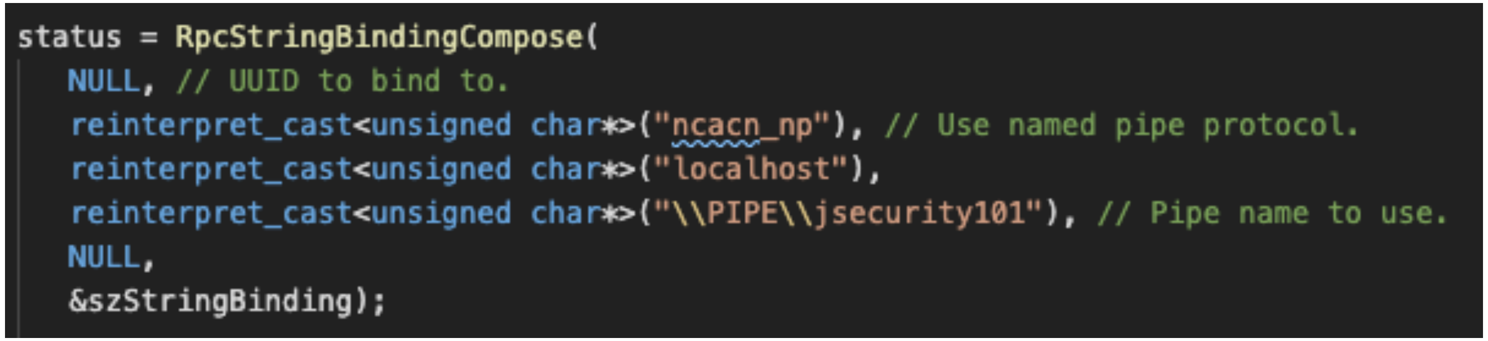
- Endpoint Mapper
  - A service that is located on every Windows host (seen as – epmapper)
  - Maintains the database of endpoints that clients use to map an interface to endpoints
- Name Service Database (Locator)
  - allows client applications to use a logical name instead of a specific network address/protocol sequence

RPC Components (cont.d)

- Endpoint
  - The TCP/IP port (ncacn_ip_tcp), or named pipe (ncacn_np), that the client will use to communicate with the server
  - Server will listen on this endpoint and wait for the client to initialize the communication
  - Two types of endpoints:
    - Static - used when an RPC Protocol will communicate over the same port/named pipe every time
    - Dynamic - used when a range of ports are utilized, or if the protocol allows connection over ncacn_ip_tcp and ncacn_np

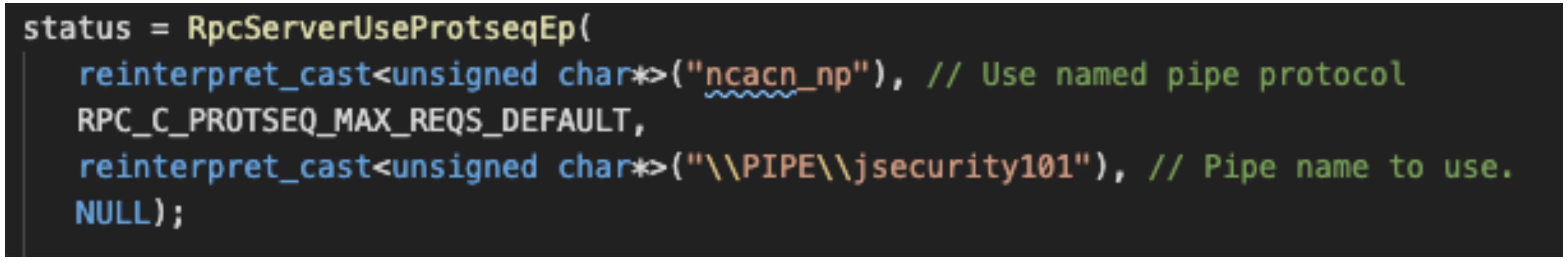
RPC Components (cont.d)

- Client Endpoint Code:



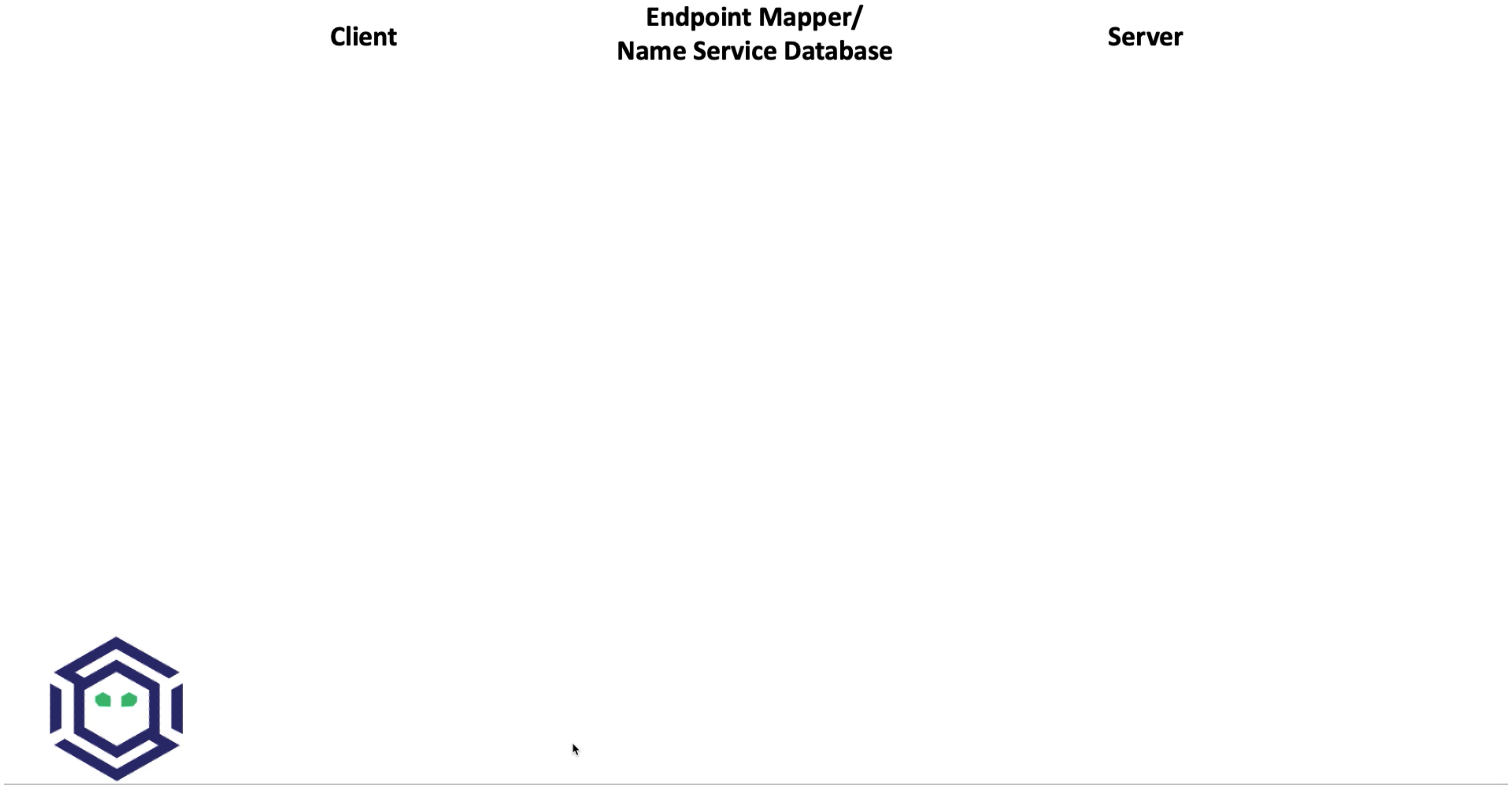
```
status = RpcStringBindingCompose(
  NULL, // UUID to bind to.
  reinterpret_cast<unsigned char*>("ncacn_np"), // Use named pipe protocol.
  reinterpret_cast<unsigned char*>("localhost"),
  reinterpret_cast<unsigned char*>("\\PIPE\\jsecurity101"), // Pipe name to use.
  NULL,
  &szStringBinding);
```

- Server Endpoint Code:



```
status = RpcServerUseProtseqEp(
  reinterpret_cast<unsigned char*>("ncacn_np"), // Use named pipe protocol
  RPC_C_PROTSEQ_MAX_REQS_DEFAULT,
  reinterpret_cast<unsigned char*>("\\PIPE\\jsecurity101"), // Pipe name to use.
  NULL);
```

RPC Process



Client

Endpoint Mapper/
Name Service Database

Server

Leveraging RPC Telemetry

How can Detection Engineers use this data

- We know an attacker can interact with an RPC Interface one of the following ways:
    - An RPC client code calls a Win32 API that will implement an RPC interface. This can be seen inside of native Windows binaries typically.
    - An RPC client contains the necessary IDL (Interface Definition Language) code baked in so that it can talk to the RPC server. An example of this can be found within Mimikatz code.
    - An RPC client will talk to the RPC server directly by implementing the RPC over TCP/IP or RPC over named pipe protocols and will not interface with the client's OS's RPC runtime.

How can Detection Engineers use this data (cont.d)

- We also know:
  - An attacker can’t control the RPC Server (given they are trying to connect to a Microsoft supported RPC server).
- Documenting different RPC s Nelson has documented RPC
- Pivoting on things attackers c

Applying RPC Knowledge

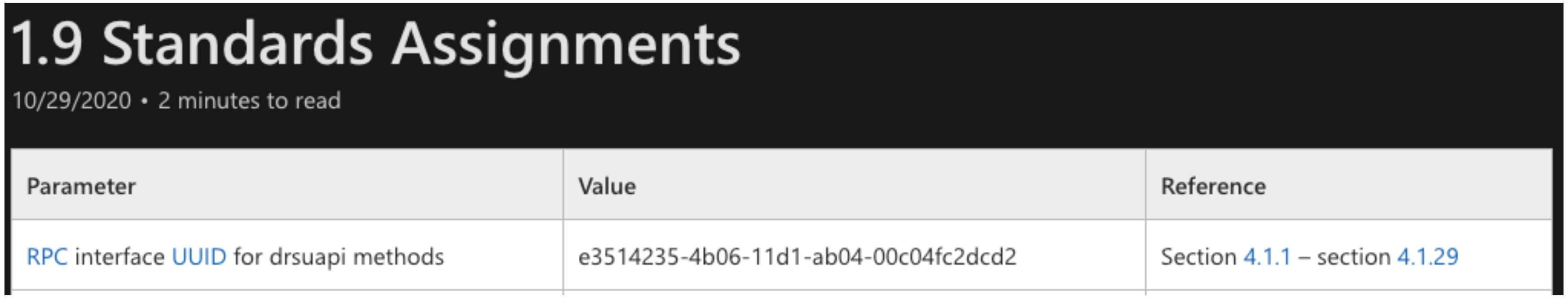
- DCSync TLDR;
  - Technique used to capture credentials by impersonating a Domain Controller
    - By taking advantage of domain replication via the Directory Replication Service RPC Protocol (MS-DRSR).
    - The interface specific for this attack will be DRSUAPI.
  - High privs needed
    - Default in: Domain Administrators, Enterprise Administrators group, or DC computer accounts but this doesn’t have to be the case.
    - DS-Replication-Get-Changes-All (GUID - 1131f6ad-9c07-11d1-f79f-00c04fc2dcd2)
    - DS-Replication-Get-Changes (GUID - 1131f6aa-9c07-11d1-f79f-00c04fc2dcd2)
      - These extended rights are needed to access the Domain-DNS Class object
      - Once access to this object is successfully acquired, replication to the NC replica with AD can be achieved via IDL_DRSGetNCChanges function.

Applying RPC Knowledge (cont.d)

- DCSync Process:
  - Attacker obtains user with the specified extended rights.
  - Targets a Domain Controller to replicate.
  - Requests the replication via IDL_DRSGetNCChanges.
  - Obtains AD secrets.
- Good Blogs:
  - Mimikatz DCSync Usage, Exploitation, and Detection by Sean Metcalf
  - Abusing Active Directory Permissions with PowerView by Will Schroeder
  - Syncing into the Shadows by Jonathan Johnson

Applying RPC Knowledge (cont.d)

- Interface UUID Identification:

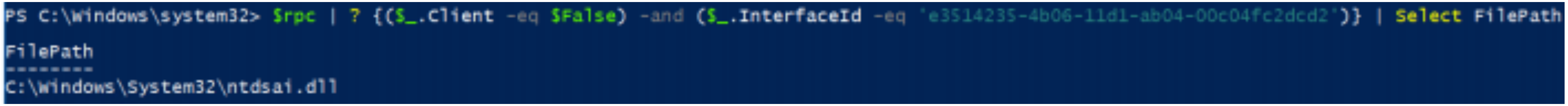


1.9 Standards Assignments

10/29/2020 • 2 minutes to read

Parameter	Value	Reference
RPC interface UUID for drsuapi methods	e3514235-4b06-11d1-ab04-00c04fc2dcd2	Section 4.1.1 – section 4.1.29

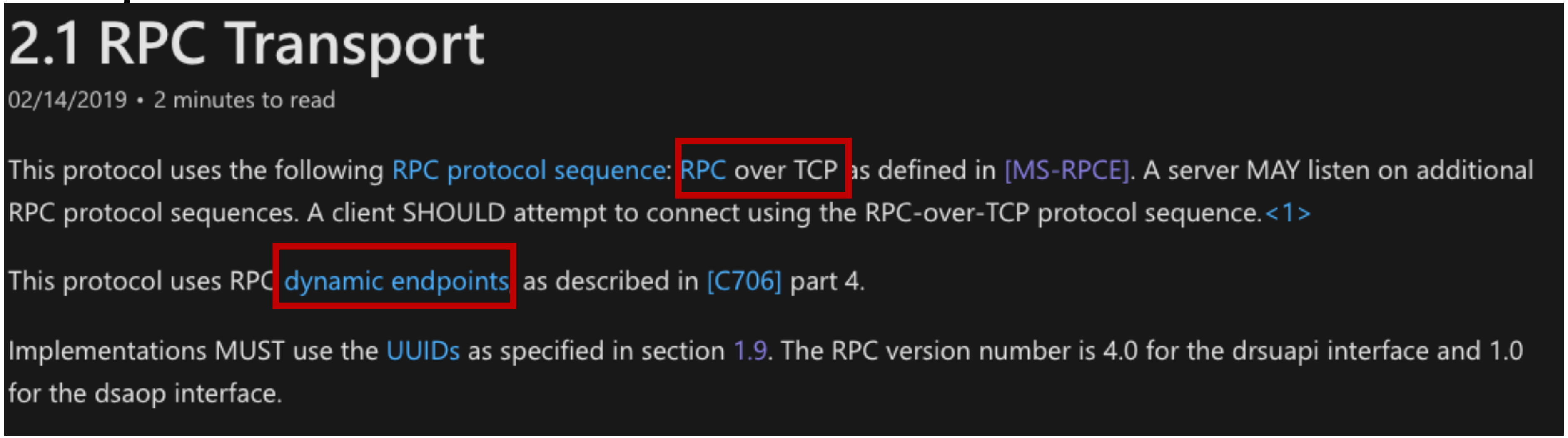
- Server Code Identification:



```
PS C:\Windows\system32> $rpc | ? {($_.Client -eq $False) -and ($_.InterfaceId -eq 'e3514235-4b06-11d1-ab04-00c04fc2dcd2')} | Select FilePath

FilePath
--------
C:\Windows\System32\ntdsai.dll
```

Applying RPC Knowledge (cont.d)

- Endpoint Identification:



2.1 RPC Transport

02/14/2019 • 2 minutes to read

This protocol uses the following RPC protocol sequence: RPC over TCP as defined in [MS-RPCE]. A server MAY listen on additional RPC protocol sequences. A client SHOULD attempt to connect using the RPC-over-TCP protocol sequence.<1>

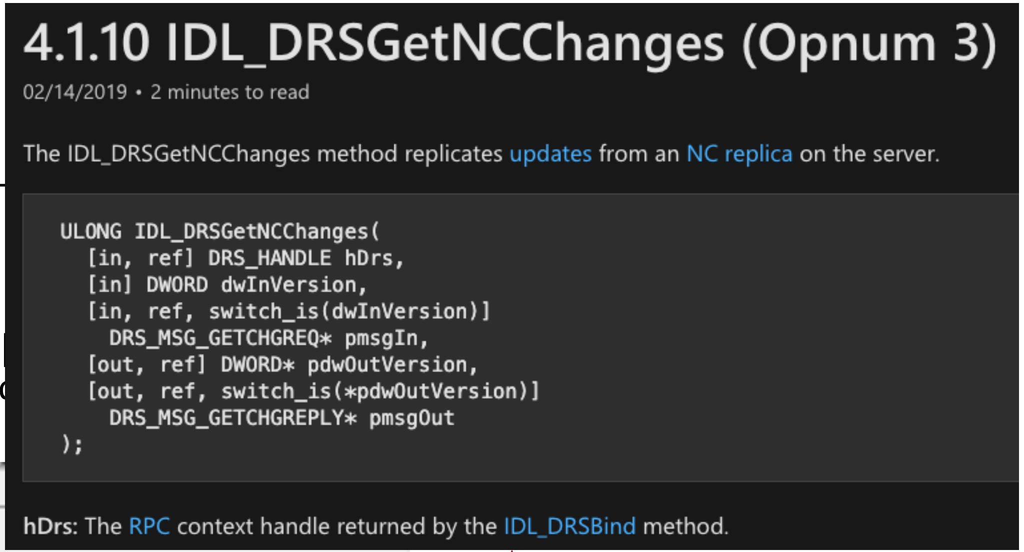
This protocol uses RPC dynamic endpoints as described in [C706] part 4.

Implementations MUST use the UUIDs as specified in section 1.9. The RPC version number is 4.0 for the drsuapi interface and 1.0 for the dsaop interface.

Applying RPC Knowledge (cont.d)

- Method Identification:
  - ETW Capture (Server Side)

```
PS > logman start MS-DRSR -p Microsoft-Windows-RPC 0xf
PS > logman stop MS-DRSR -ets
PS > tracerpt MS-DRSR.etl -o MS-DRSR.evtx -of EVTX
PS > Get-WinEvent -Path .\MS-DRSR.evtx -FilterXPath "*
EventData[Data[@Name='e3514235-4b06-11d1-ab04-00c04fc2c
Property * | Out-File DRSR
```



4.1.10 IDL_DRSGetNCChanges (Opnum 3)

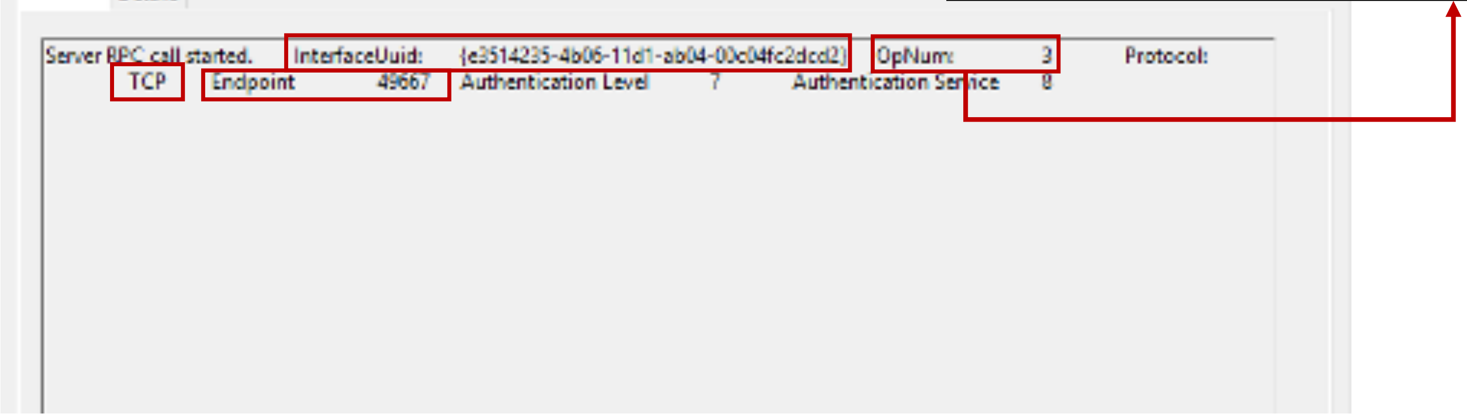
02/14/2019 • 2 minutes to read

The IDL_DRSGetNCChanges method replicates updates from an NC replica on the server.

```
ULONG IDL_DRSGetNCChanges(
  [in, ref] DRS_HANDLE hDrs,
  [in] DWORD dwInVersion,
  [in, ref, switch_is(dwInVersion)]
    DRS_MSG_GETCHGREQ* pmsgIn,
  [out, ref] DWORD* pdwOutVersion,
  [out, ref, switch_is(*pdwOutVersion)]
    DRS_MSG_GETCHGREPLY* pmsgOut
);
```

hDrs: The RPC context handle returned by the IDL_DRSBind method.

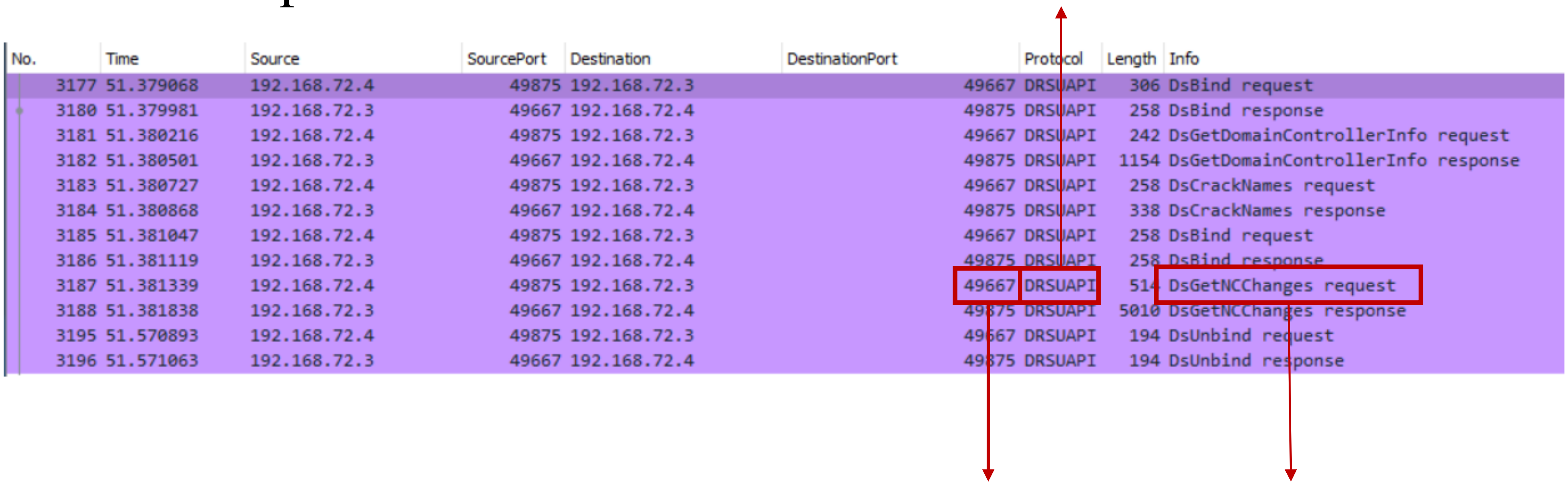
Event 6, RPC (Microsoft-Windows-RPC)



Server RPC call started. InterfaceUuid: {e3514235-4b06-11d1-ab04-00c04fc2dcd2} OpNum: 3 Protocol:
TCP Endpoint 49667 Authentication Level 7 Authentication Service 8

Applying RPC Knowledge (cont.d)

- Wireshark Capture:



No.	Time	Source	SourcePort	Destination	DestinationPort	Protocol	Length	Info
3177	51.379068	192.168.72.4	49875	192.168.72.3	49667	DRSUAPI	306	DsBind request
3180	51.379981	192.168.72.3	49667	192.168.72.4	49875	DRSUAPI	258	DsBind response
3181	51.380216	192.168.72.4	49875	192.168.72.3	49667	DRSUAPI	242	DsGetDomainControllerInfo request
3182	51.380501	192.168.72.3	49667	192.168.72.4	49875	DRSUAPI	1154	DsGetDomainControllerInfo response
3183	51.380727	192.168.72.4	49875	192.168.72.3	49667	DRSUAPI	258	DsCrackNames request
3184	51.380868	192.168.72.3	49667	192.168.72.4	49875	DRSUAPI	338	DsCrackNames response
3185	51.381047	192.168.72.4	49875	192.168.72.3	49667	DRSUAPI	258	DsBind request
3186	51.381119	192.168.72.3	49667	192.168.72.4	49875	DRSUAPI	258	DsBind response
3187	51.381339	192.168.72.4	49875	192.168.72.3	49667	DRSUAPI	514	DsGetNCChanges request
3188	51.381838	192.168.72.3	49667	192.168.72.4	49875	DRSUAPI	5010	DsGetNCChanges response
3195	51.570893	192.168.72.4	49875	192.168.72.3	49667	DRSUAPI	194	DsUnbind request
3196	51.571063	192.168.72.3	49667	192.168.72.4	49875	DRSUAPI	194	DsUnbind response

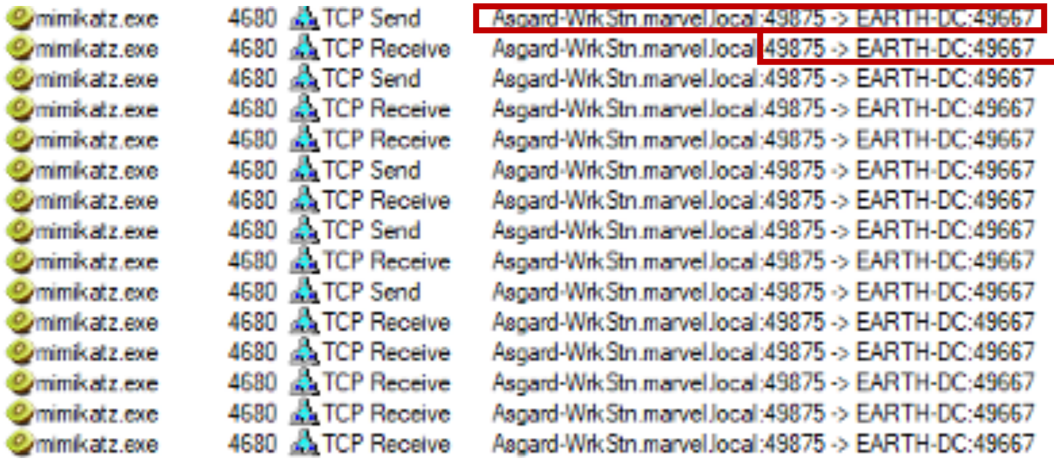
Interface

TCP Endpoint

OpNum 3 (Method)

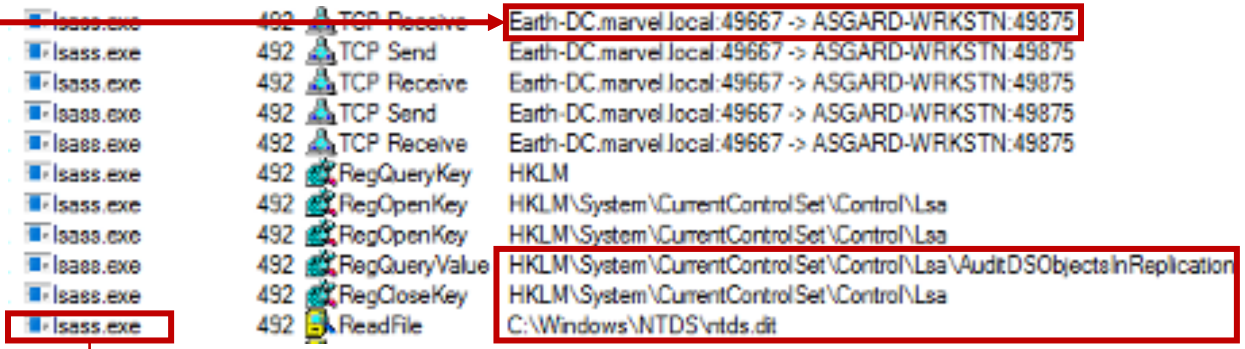
Applying RPC Knowledge (cont.d)

Server Connection:



```
mimikatz.exe    4680  TCP Send      Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Send      Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Send      Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Send      Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Send      Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
mimikatz.exe    4680  TCP Receive   Asgard-WrkStn.marvel.local:49875 -> EARTH-DC:49667
```

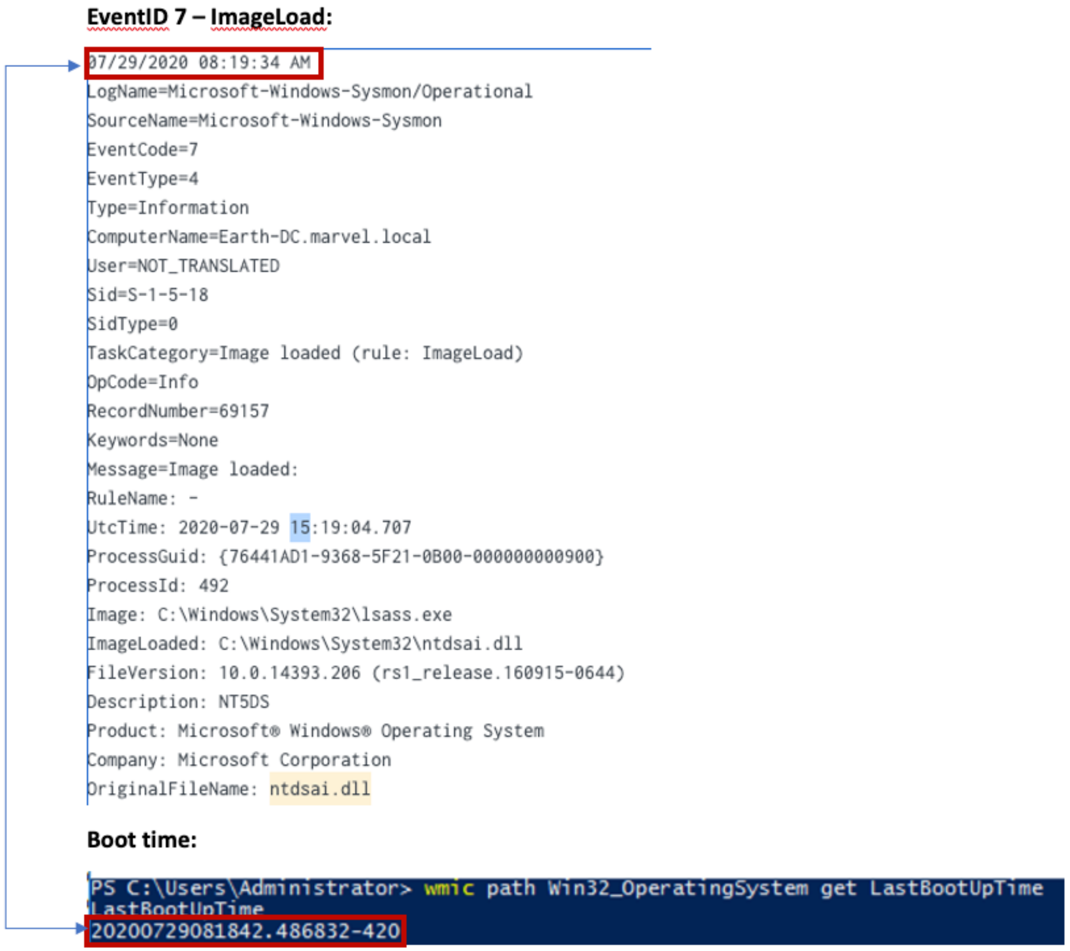
Client Connection



```
lsass.exe    492  TCP Receive    Earth-DC.marvel.local:49667 -> ASGARD-WRKSTN:49875
lsass.exe    492  TCP Send       Earth-DC.marvel.local:49667 -> ASGARD-WRKSTN:49875
lsass.exe    492  TCP Receive    Earth-DC.marvel.local:49667 -> ASGARD-WRKSTN:49875
lsass.exe    492  TCP Send       Earth-DC.marvel.local:49667 -> ASGARD-WRKSTN:49875
lsass.exe    492  TCP Receive    Earth-DC.marvel.local:49667 -> ASGARD-WRKSTN:49875
lsass.exe    492  RegQueryKey    HKLM
lsass.exe    492  RegOpenKey     HKLM\System\CurrentControlSet\Control\Lsa
lsass.exe    492  RegOpenKey     HKLM\System\CurrentControlSet\Control\Lsa
lsass.exe    492  RegQueryValue  HKLM\System\CurrentControlSet\Control\Lsa\AuditDSObjectsInReplication
lsass.exe    492  RegCloseKey    HKLM\System\CurrentControlSet\Control\Lsa
lsass.exe    492  ReadFile       C:\Windows\NTDS\ntds.dit
```

??????

Applying RPC Knowledge (cont.d)



EventID 7 – ImageLoad:

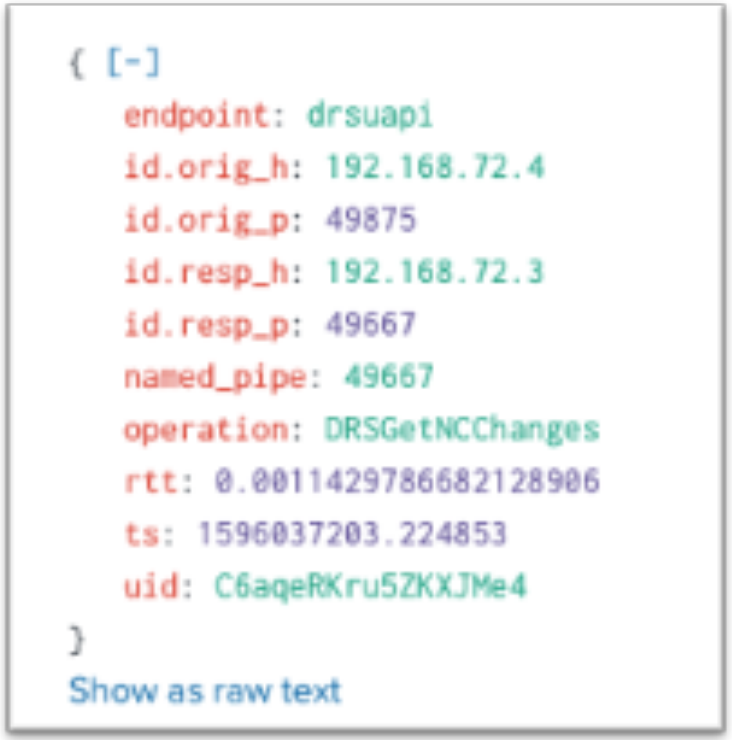
```
07/29/2020 08:19:34 AM
LogName=Microsoft-Windows-Sysmon/Operational
SourceName=Microsoft-Windows-Sysmon
EventCode=7
EventType=4
Type=Information
ComputerName=Earth-DC.marvel.local
User=NOT_TRANSLATED
Sid=S-1-5-18
SidType=0
TaskCategory=Image loaded (rule: ImageLoad)
OpCode=Info
RecordNumber=69157
Keywords=None
Message=Image loaded:
RuleName: -
UtcTime: 2020-07-29 15:19:04.707
ProcessGuid: {76441AD1-9368-5F21-0B00-000000000900}
ProcessId: 492
Image: C:\Windows\System32\lsass.exe
ImageLoaded: C:\Windows\System32\ntdsai.dll
FileVersion: 10.0.14393.206 (rs1_release.160915-0644)
Description: NT5DS
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: ntdsai.dll
```

Boot time:

```
PS C:\Users\Administrator> wmic path Win32_OperatingSystem get LastBootUpTime
LastBootUpTime
20200729081842.486832-420
```

Research Telemetry -> Scalable Telemetry

Network RPC:

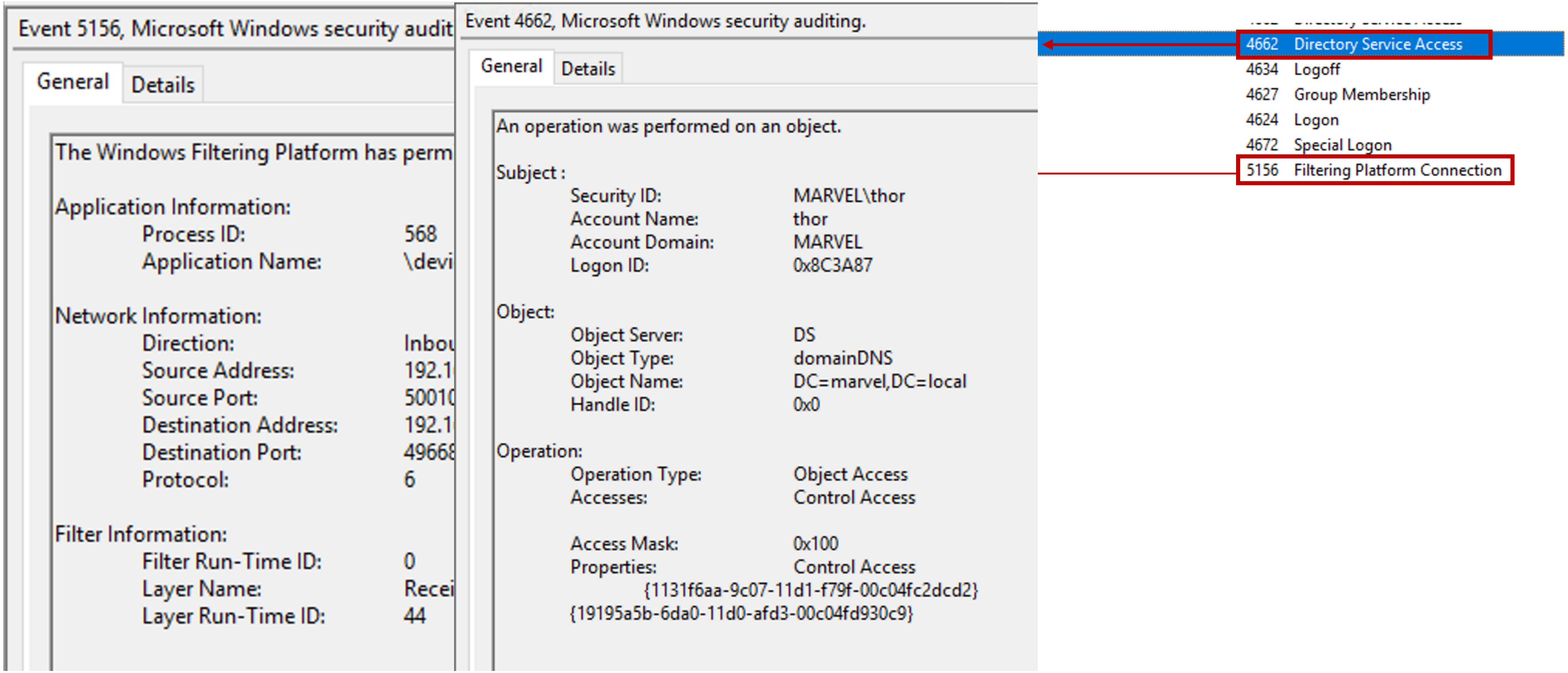


```
{ [-]
   endpoint: drsuapi
   id.orig_h: 192.168.72.4
   id.orig_p: 49875
   id.resp_h: 192.168.72.3
   id.resp_p: 49667
   named_pipe: 49667
   operation: DRSGetNCChanges
   rtt: 0.0011429786682128906
   ts: 1596037203.224853
   uid: C6aqeRKru5ZKXJMe4
}
```

Host Data:

?

Research Telemetry -> Scalable Telemetry (cont.d) (Server-Side)



Event 5156, Microsoft Windows security audit

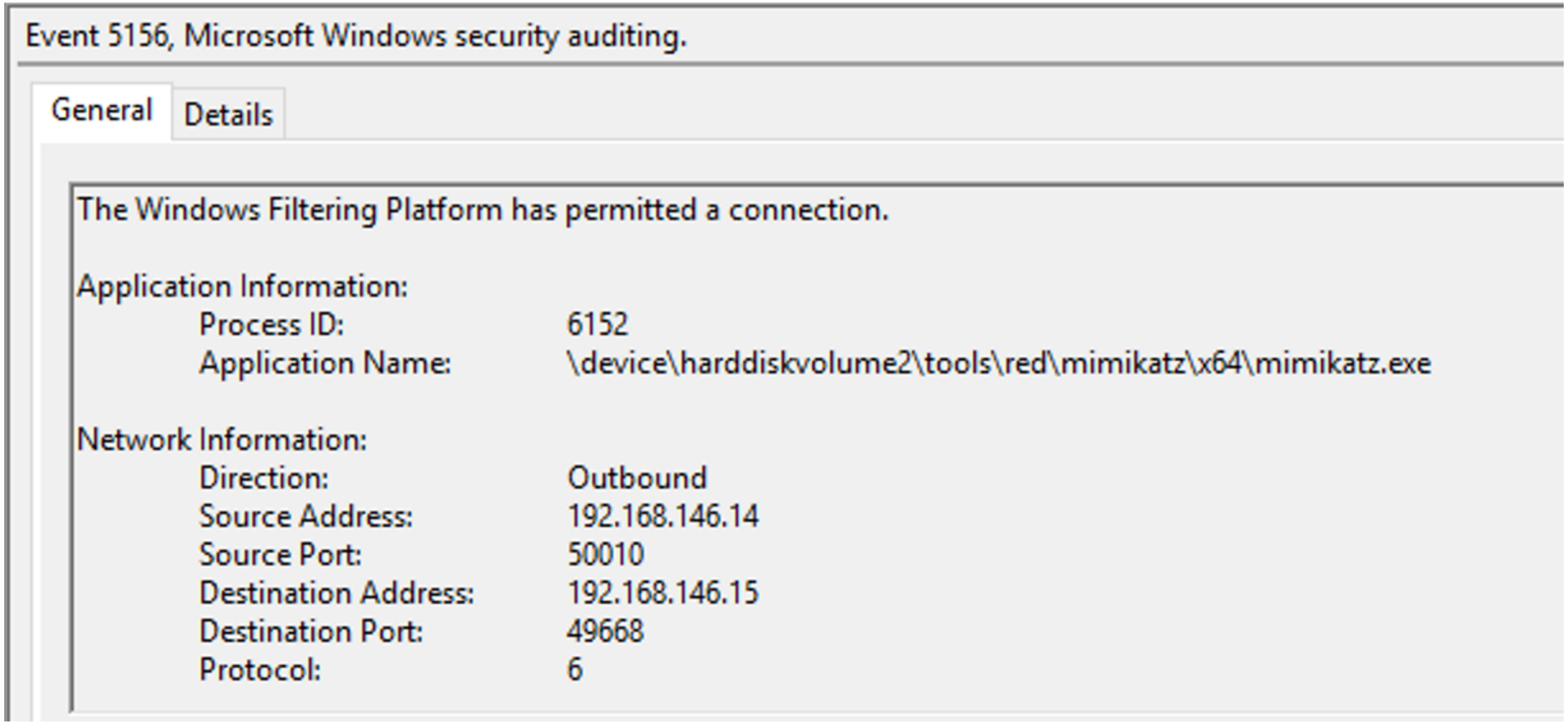
```
The Windows Filtering Platform has perm

Application Information:
        Process ID:                 568
        Application Name:           \devi

Network Information:
        Direction:                  Inbou
        Source Address:             192.1
        Source Port:                5001(
        Destination Address:        192.1
        Destination Port:           49668
        Protocol:                   6

Filter Information:
        Filter Run-Time ID:         0
        Layer Name:                 Recei
        Layer Run-Time ID:          44
```

Event 4662, Microsoft Windows security auditing.

```
An operation was performed on an object.

Subject :
        Security ID:            MARVEL\thor
        Account Name:           thor
        Account Domain:         MARVEL
        Logon ID:               0x8C3A87

Object:
        Object Server:          DS
        Object Type:            domainDNS
        Object Name:            DC=marvel,DC=local
        Handle ID:              0x0

Operation:
        Operation Type:         Object Access
        Accesses:               Control Access

        Access Mask:            0x100
        Properties:             Control Access
                {1131f6aa-9c07-11d1-f79f-00c04fc2dcd2}
        {19195a5b-6da0-11d0-afd3-00c04fd930c9}
```

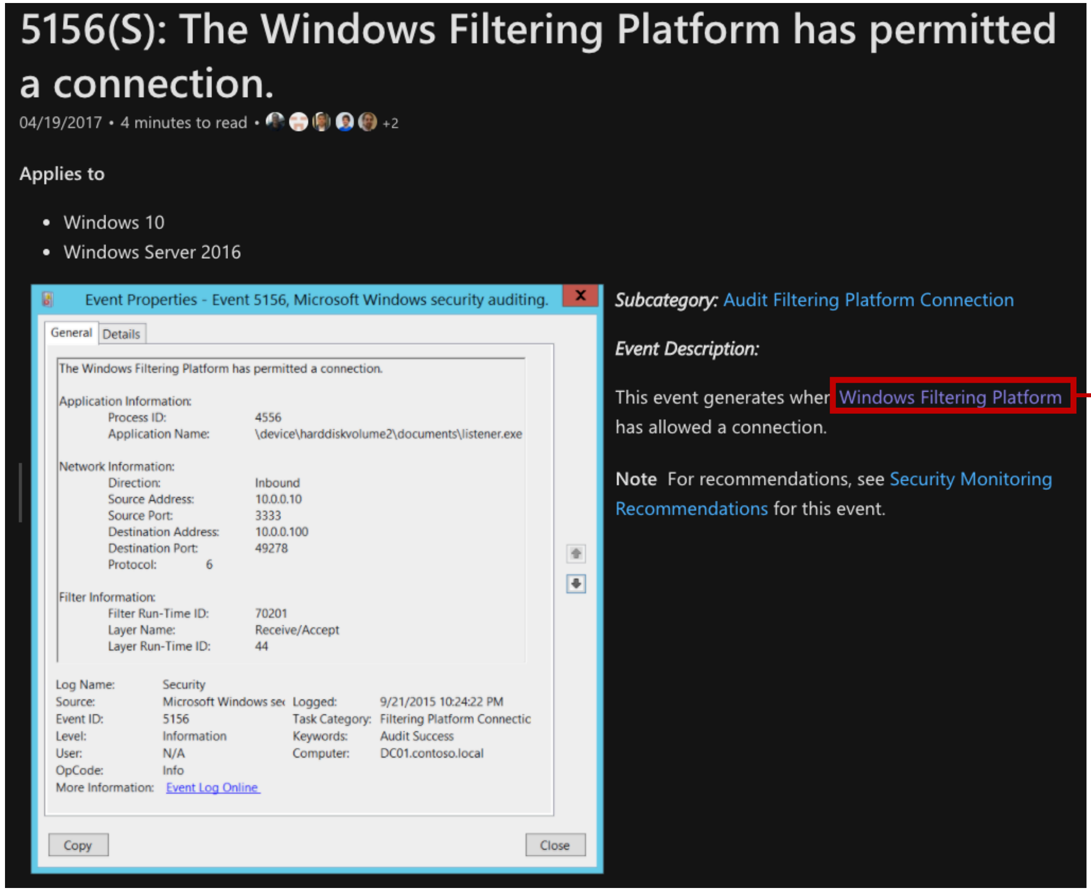
```
4662   Directory Service Access
4634   Logoff
4627   Group Membership
4624   Logon
4672   Special Logon
5156   Filtering Platform Connection
```

Research Telemetry -> Scalable Telemetry (cont.d) (Client-Side)



```
The Windows Filtering Platform has permitted a connection.

Application Information:
        Process ID:             6152
        Application Name:       \device\harddiskvolume2\tools\red\mimikatz\x64\mimikatz.exe

Network Information:
        Direction:              Outbound
        Source Address:         192.168.146.14
        Source Port:            50010
        Destination Address:    192.168.146.15
        Destination Port:       49668
        Protocol:               6
```

Research Telemetry -> Scalable Telemetry (cont.d)



5156(S): The Windows Filtering Platform has permitted a connection.

04/19/2017 • 4 minutes to read • +2

Applies to

- Windows 10
- Windows Server 2016

Event Properties - Event 5156, Microsoft Windows security auditing.

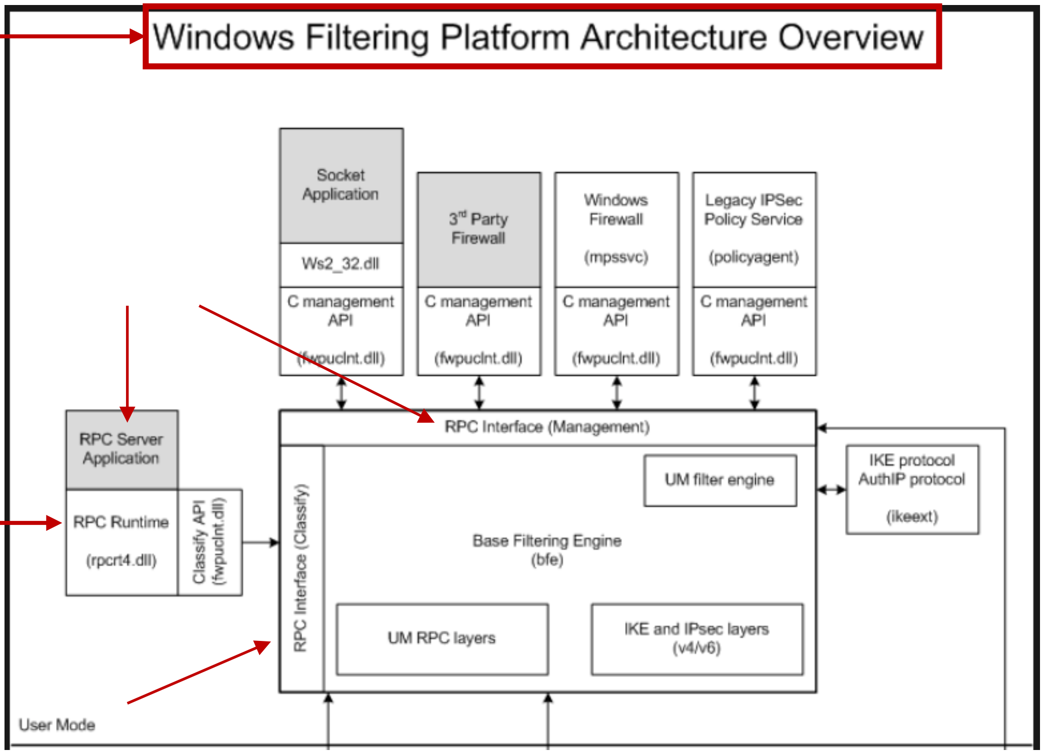
```
The Windows Filtering Platform has permitted a connection.

Application Information:
        Process ID:             4556
        Application Name:       \device\harddiskvolume2\documents\listener.exe

Network Information:
        Direction:              Inbound
        Source Address:         10.0.0.10
        Source Port:            3333
        Destination Address:    10.0.0.100
        Destination Port:       49278
        Protocol:        6

Filter Information:
        Filter Run-Time ID:     70201
        Layer Name:             Receive/Accept
        Layer Run-Time ID:      44

Log Name:          Security
Source:            Microsoft Windows sec   Logged:          9/21/2015 10:24:22 PM
Event ID:          5156                    Task Category:   Filtering Platform Connectic
Level:             Information             Keywords:        Audit Success
User:              N/A                     Computer:        DC01.contoso.local
OpCode:            Info
More Information:  Event Log Online
```

Subcategory: Audit Filtering Platform Connection

Event Description:

This event generates wher Windows Filtering Platform has allowed a connection.

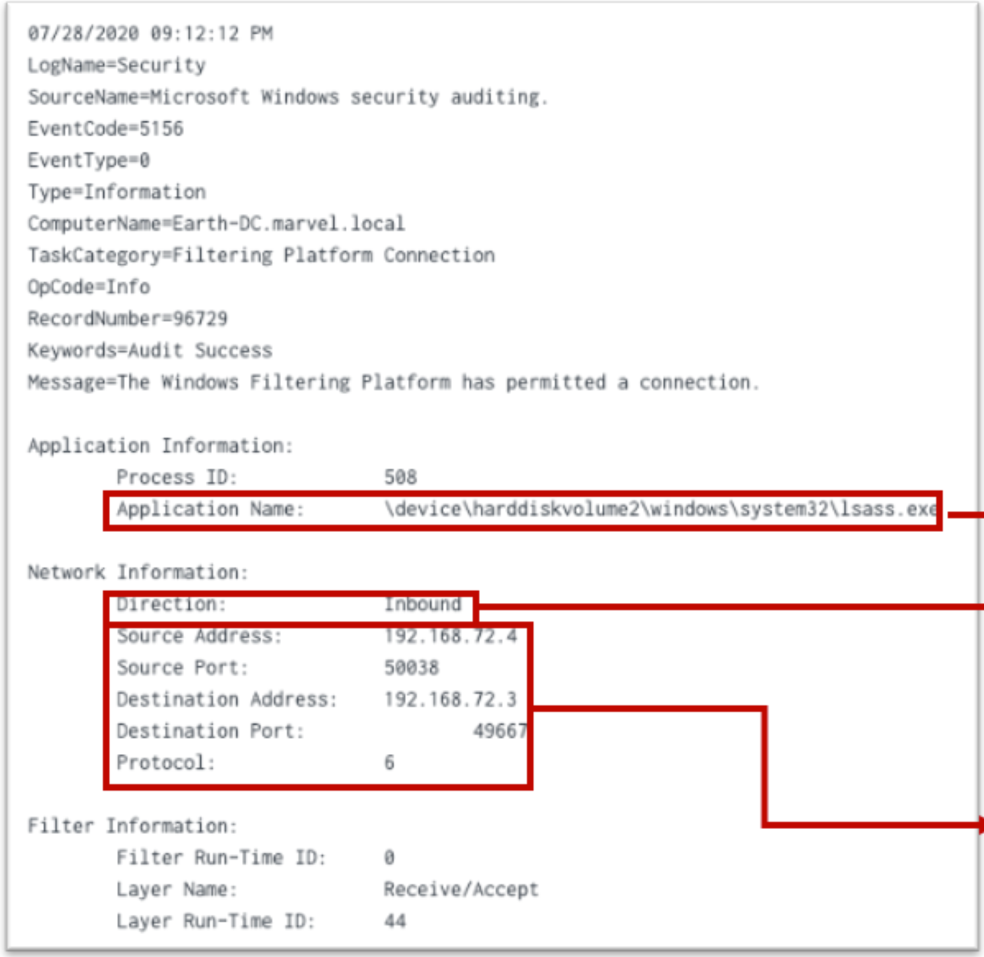
Note For recommendations, see Security Monitoring Recommendations for this event.



Windows Filtering Platform Architecture Overview

Research Telemetry -> Scalable Telemetry (cont.d)

MS-DRSR:



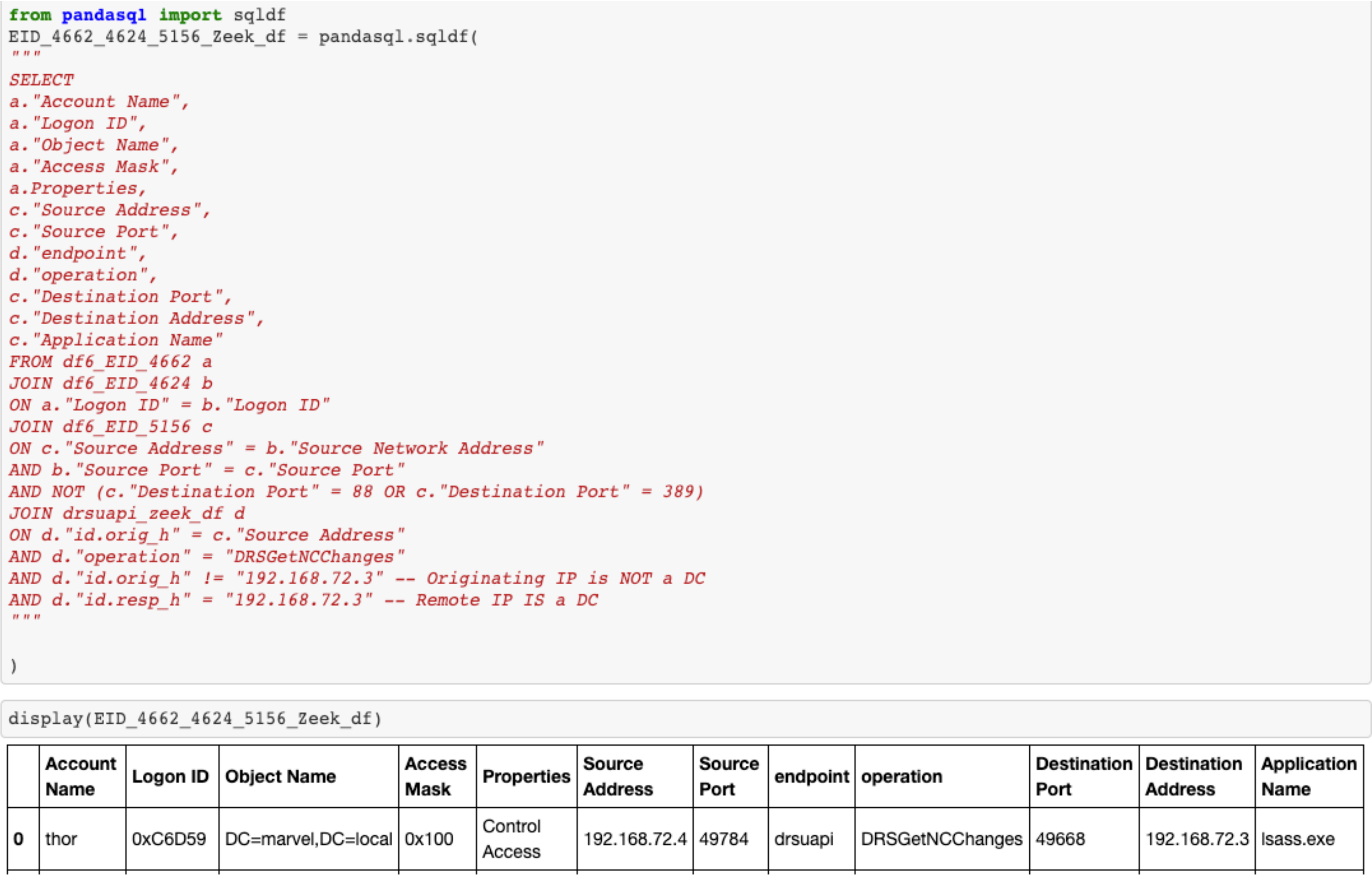
```
07/28/2020 09:12:12 PM
LogName=Security
SourceName=Microsoft Windows security auditing.
EventCode=5156
EventType=0
Type=Information
ComputerName=Earth-DC.marvel.local
TaskCategory=Filtering Platform Connection
OpCode=Info
RecordNumber=96729
Keywords=Audit Success
Message=The Windows Filtering Platform has permitted a connection.

Application Information:
        Process ID:             508
        Application Name:       \device\harddiskvolume2\windows\system32\lsass.exe

Network Information:
        Direction:              Inbound
        Source Address:         192.168.72.4
        Source Port:            50038
        Destination Address:    192.168.72.3
        Destination Port:               49667
        Protocol:               6

Filter Information:
        Filter Run-Time ID:     0
        Layer Name:             Receive/Accept
        Layer Run-Time ID:      44
```

RPC Server Application

Inbound means that this application didn't initialize the connection, proving the above that this is the Server Application

Network Packet Data. Saw similar information within Zeek.

Analytic



```
from pandasql import sqldf
EID_4662_4624_5156_Zeek_df = pandasql.sqldf(
"""
SELECT
a."Account Name",
a."Logon ID",
a."Object Name",
a."Access Mask",
a.Properties,
c."Source Address",
c."Source Port",
d."endpoint",
d."operation",
c."Destination Port",
c."Destination Address",
c."Application Name"
FROM df6_EID_4662 a
JOIN df6_EID_4624 b
ON a."Logon ID" = b."Logon ID"
JOIN df6_EID_5156 c
ON c."Source Address" = b."Source Network Address"
AND b."Source Port" = c."Source Port"
AND NOT (c."Destination Port" = 88 OR c."Destination Port" = 389)
JOIN drsuapi_zeek_df d
ON d."id.orig_h" = c."Source Address"
AND d."operation" = "DRSGetNCChanges"
AND d."id.orig_h" != "192.168.72.3" -- Originating IP is NOT a DC
AND d."id.resp_h" = "192.168.72.3" -- Remote IP IS a DC
"""

)
```

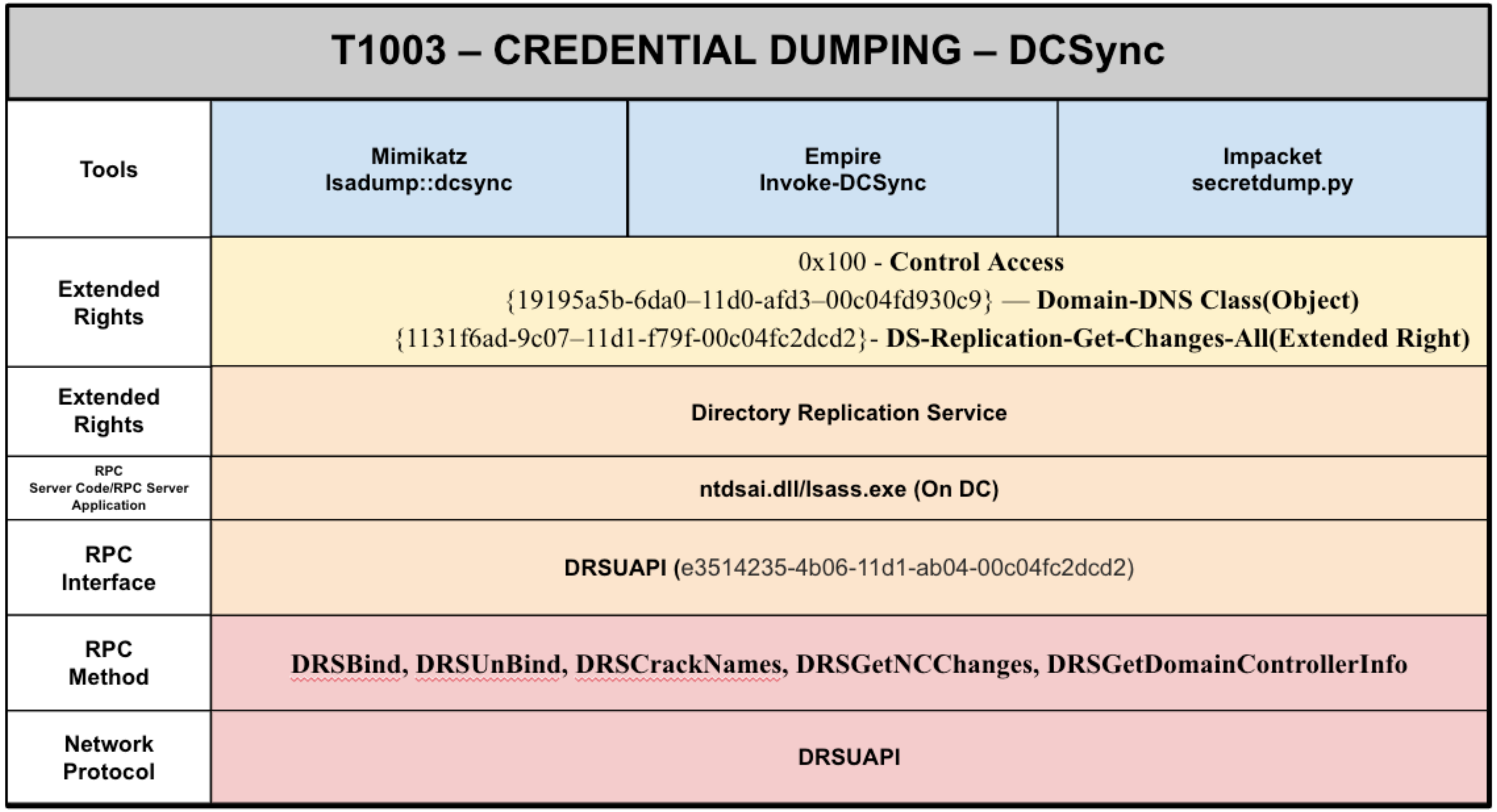
```
display(EID_4662_4624_5156_Zeek_df)
```

	Account Name	Logon ID	Object Name	Access Mask	Properties	Source Address	Source Port	endpoint	operation	Destination Port	Destination Address	Application Name
0	thor	0xC6D59	DC=marvel,DC=local	0x100	Control Access	192.168.72.4	49784	drsuapi	DRSGetNCChanges	49668	192.168.72.3	lsass.exe

Answering Unknowns

- What did RPC do *exactly?*
  - Answered via RPC Process
- When the network connection was made, what process(es) made a connection to each other? (Client/Server processes)
  - Mimikatz (Outbound / Client)
  - NTDSAI gets loaded into LSASS (Inbound / Server)
- How were the credentials being brought back to the client?
  - OpNum #3 (DRSGetNCChanges) via DRSUAPI Interface (UUID - e3514235-4b06-11d1-ab04-00c04fc2dcd2)
- Outside of network traffic, is there visibility into RPC?
  - Event ID 5156/Sysmon Event ID 3

Abstraction Map (Post-RPC Research)



T1003 – CREDENTIAL DUMPING – DCSync			
Tools	Mimikatz lsadump::dcsync	Empire Invoke-DCSync	Impacket secretdump.py
Extended Rights	0x100 - **Control Access** {19195a5b-6da0–11d0-afd3–00c04fd930c9} — **Domain-DNS Class(Object)** {1131f6ad-9c07–11d1-f79f-00c04fc2dcd2}- **DS-Replication-Get-Changes-All(Extended Right)**		
Extended Rights	Directory Replication Service		
RPC Server Code/RPC Server Application	ntdsai.dll/lsass.exe (On DC)		
RPC Interface	**DRSUAPI** (e3514235-4b06-11d1-ab04-00c04fc2dcd2)		
RPC Method	DRSBind, DRSUnBind, DRSCrackNames, DRSGetNCChanges, DRSGetDomainControllerInfo		
Network Protocol	DRSUAPI		

Conclusion Thoughts

- Purpose
- Scaling Data

References

- Research Paper
  - https://specterops.io/assets/resources/RPC_for_Detection_Engineers.pdf
- ALPC (Clément Rouault and Thomas Imbert)
  - https://pacsec.jp/psj17/PSJ2017_Rouault_Imbert_alpc_rpc_pacsec.pdf
- Everything James Forshaw related
- Microsoft Documentation
- COM (Matt Nelson/Casey Smith)
  - https://www.youtube.com/watch?v=cfZNVX53LPM&t=194s

www.specterops.io
@specterops
info@specterops.io