

# Impacket Guide: SMB/MSRPC

May 7, 2020 By Raj Chandel

There have been many Red Team scenarios, Capture the Flag challenges where we face the Windows Server. After exploiting and getting the initial foothold in the server, it is tough to extract the data and as well as there are scenarios where we couldn't get onto the server per se. But using the SMB, we can execute commands remotely on the server. The SecureAuth visualized this, and they gave us one of the most amazing collections of Python classes for working on different protocols. This collection is named Impacket.

Official GitHub Repository: [SecureAuthCorp /impacket](https://github.com/SecureAuthCorp/impacket)

## Table of Contents

- **Introduction to SMB**
- **Introduction to MSRPC**
- **Configurations Used in Practical**
- **Impacket Categories**
- **Installation**
- **smbclient.py**
- **lookupsid.py**
- **reg.py**
- **rpcdump.py**
- **samrdump.py**
- **services.py**
- **ifmap.py**
- **opdump.py**
- **getArch.py**
- **netview.py**
- **Conclusion**

## Introduction to SMB

The SMB is a network protocol which is also known as the Server Message Block protocol. It is used to communicate between a client and a server. It can be used to share the files, printers and some other network resources. It was created by IBM in the 1980s.

## Introduction to MSRPC

MSRPC or Microsoft Remote Procedure Call is a modified version of DCE/RPC. It was created by Microsoft to seamlessly create a client/server model in Windows. The Windows Server domain protocols are entirely based on

MSRPC.

## Configurations Used in Practical

- **Attacker Machine**
  - **OS:** Kali Linux 2020.1
  - **IP Address:** 168.1.112
- **Target Machine**
  - **OS:** Windows Server 2016
  - **IP Address:** 168.1.105

## Impacket Categories

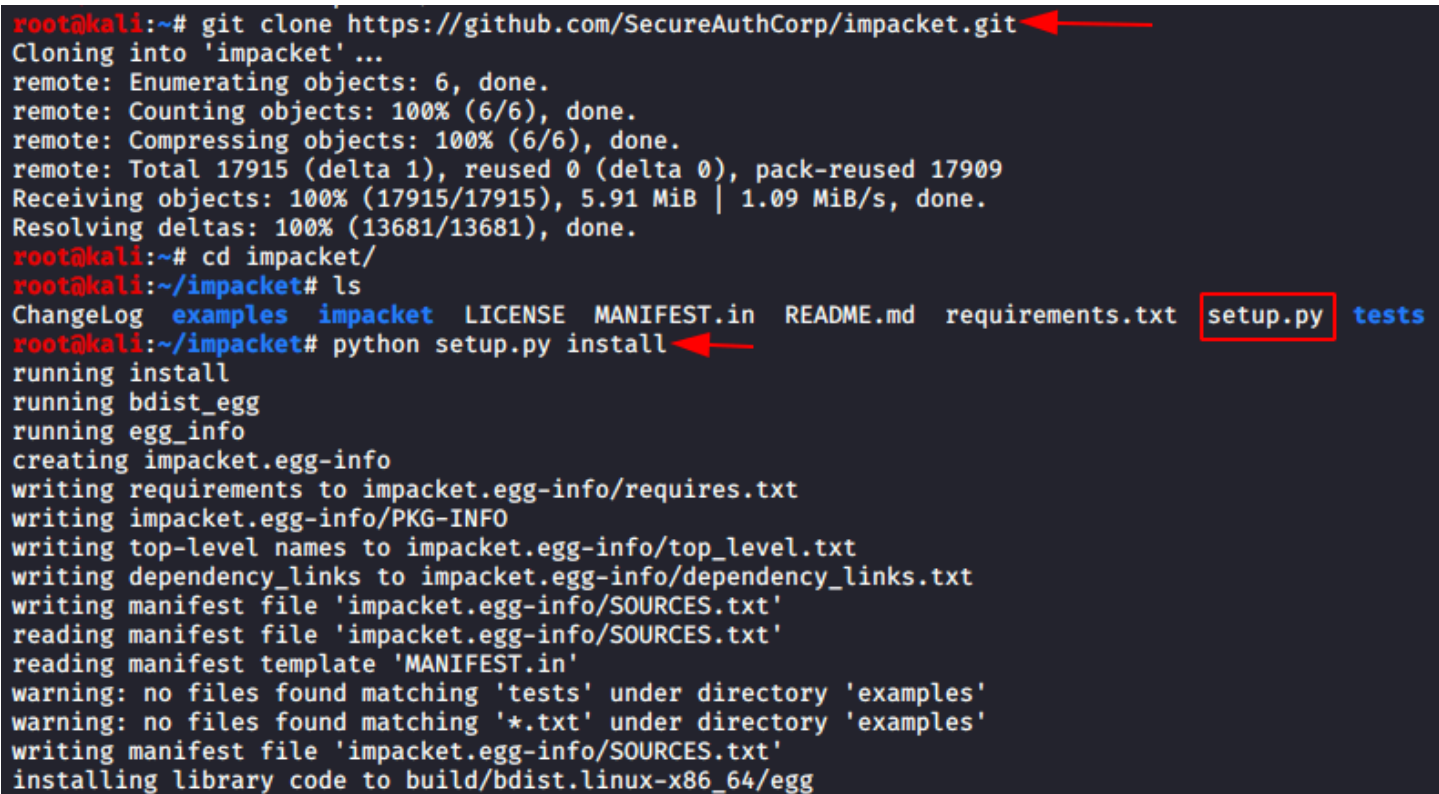
- **Remote Execution**
- **Kerberos**
- **Windows Secrets**
- **Server Tools/MiTM Attacks**
- **WMI**
- **Known Vulnerabilities**
- **SMB/MSRPC**
  - smbclient.py
  - lookupsid.py
  - reg.py
  - rpcdump.py
  - samrdump.py
  - services.py
  - ifmap.py
  - opdump.py
  - getArch.py
  - netview.py
- **MSSQL / TDS**
- **File Formats**
- **Other**

## Installation

Before using the Impacket tool kit on our system, we need to install it. The installation process is quite simple. First, head to the GitHub Repository by clicking [here](#). Then using the git clone command, we clone the complete

repository to our Attacker Machine. After cloning we can see that there is a setup.py file, let us install it. After installation, we will head to the examples directory and use the scripts as per our convenience.

```
git clone https://github.com/SecureAuthCorp/impacket.git
cd impacket/
ls
python setup.py install
```



```
root@kali:~# git clone https://github.com/SecureAuthCorp/impacket.git
Cloning into 'impacket' ...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 17915 (delta 1), reused 0 (delta 0), pack-reused 17909
Receiving objects: 100% (17915/17915), 5.91 MiB | 1.09 MiB/s, done.
Resolving deltas: 100% (13681/13681), done.
root@kali:~# cd impacket/
root@kali:~/impacket# ls
ChangeLog  examples  impacket  LICENSE  MANIFEST.in  README.md  requirements.txt  setup.py  tests
root@kali:~/impacket# python setup.py install
running install
running bdist_egg
running egg_info
creating impacket.egg-info
writing requirements to impacket.egg-info/requires.txt
writing impacket.egg-info/PKG-INFO
writing top-level names to impacket.egg-info/top_level.txt
writing dependency_links to impacket.egg-info/dependency_links.txt
writing manifest file 'impacket.egg-info/SOURCES.txt'
reading manifest file 'impacket.egg-info/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no files found matching 'tests' under directory 'examples'
warning: no files found matching '*.txt' under directory 'examples'
writing manifest file 'impacket.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
```

## smbclient.py

There are moments where we needed to perform multiple actions between the attacker machine and the target machine. It can be listing shares and files, renaming some file, uploading the binaries or downloading files from the target machine. There are some situations where we even need to create a folder or two on the target machine. Performing such actions can get tricky while working with a shell that can be detected or can close at any time. The smbclient.py script helps us in these situations. It can connect to the Target Machine with the help of a bunch of attributes.

### Requirements:

- Domain
- Username
- Password/Password Hash
- Target IP Address

When we provide the following parameters to the smbclient in such a format as shown below and we will get connected to the target machine and we have an smb shell which can run a whole range of commands like dir, cd, pwd, put, rename, more, del, rm, mkdir, rmdir, info, etc

### Syntax:

**smbclient.py [domain]/[user]:[password/password hash]@[Target IP Address]**

### Command:

```
smbclient.py ignite/Administrator:Ignite@987@192.168.1.105
```

```
root@kali:~/impacket/examples# smbclient.py ignite/Administrator:Ignite@987@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

Type help for list of commands
# info
Version Major: 10
Version Minor: 0
Server Name: WIN-S0V7KMTVLD2
Server Comment:
Server UserPath: c:\
Simultaneous Users: 16777216
```

## lookupsid.py

A Security Identifier (SID) is a unique value of variable length that is used to identify a user account. Through a SID User Enumeration, we can extract the information about what users exist and their data. Lookupsid script can enumerate both local and domain users. There is a Metasploit module too for this attack. If you are planning on injecting a target server with a golden or a silver ticket then one of the things that are required is the SID of the 500 user. Lookupsid.py can be used in that scenario. When we provide the following parameters to the Lookupsid in such a format as shown below.

### Requirements:

- Domain
- Username
- Password/Password Hash
- Target IP Address

### Syntax:

**lookupsid.py [domain]/[user]:[password/password hash]@[Target IP Address]**

### Command:

```
lookupsid.py ignite/Administrator:Ignite@987@192.168.1.105
```

```
root@kali:~/impacket/examples# lookupsid.py ignite/Administrator:Ignite@987@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation
```

```
[*] Brute forcing SIDs at 192.168.1.105
[*] StringBinding ncacn_np:192.168.1.105[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-3523557010-2506964455-2614950430
498: IGNITE\Enterprise Read-only Domain Controllers (SidTypeGroup)
500: IGNITE\Administrator (SidTypeUser)
501: IGNITE\Guest (SidTypeUser)
502: IGNITE\krbtgt (SidTypeUser)
503: IGNITE\DefaultAccount (SidTypeUser)
512: IGNITE\Domain Admins (SidTypeGroup)
513: IGNITE\Domain Users (SidTypeGroup)
514: IGNITE\Domain Guests (SidTypeGroup)
515: IGNITE\Domain Computers (SidTypeGroup)
516: IGNITE\Domain Controllers (SidTypeGroup)
517: IGNITE\Cert Publishers (SidTypeAlias)
518: IGNITE\Schema Admins (SidTypeGroup)
519: IGNITE\Enterprise Admins (SidTypeGroup)
520: IGNITE\Group Policy Creator Owners (SidTypeGroup)
521: IGNITE\Read-only Domain Controllers (SidTypeGroup)
522: IGNITE\Cloneable Domain Controllers (SidTypeGroup)
525: IGNITE\Protected Users (SidTypeGroup)
```

## reg.py

This Impacket script is ripped straight out of the reg.exe of the Windows OS. Reg.exe is an executable service that can read, modify and delete registry values when used with eh combination of the query, add, delete keywords respectively. We can even begin to express the importance of access to the registry. Registry controls each and every aspect of the system. It can be used to gain information about the various policies, software and also alter some of those policies.

### Requirements:

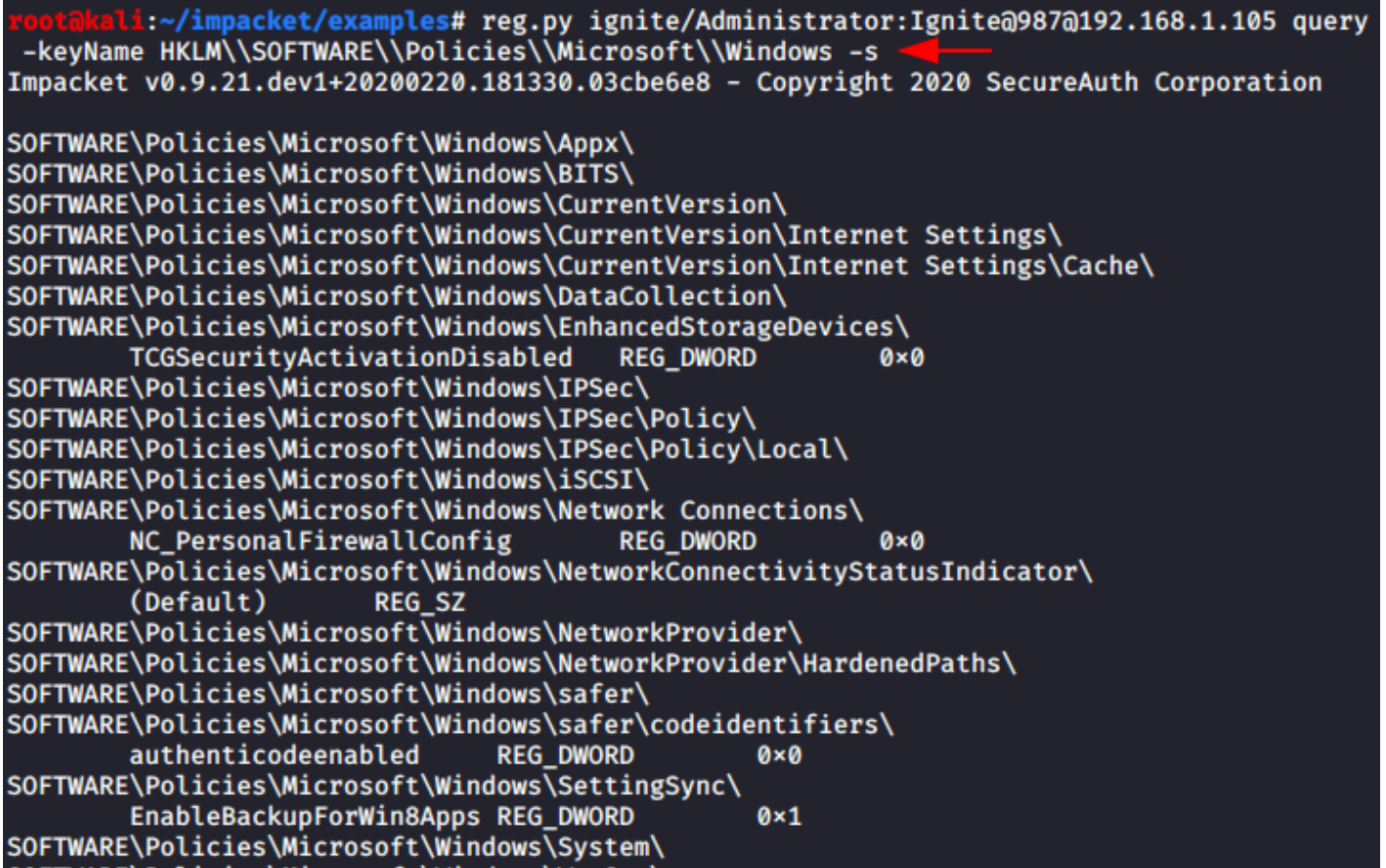
- Domain
- Username
- Password/Password Hash
- Target IP Address
- Registry Key Name

### Syntax:

```
reg.py [domain]/[user]:[password:password hash]@[Target IP Address] [action] [action parameter]
```

### Command:

```
reg.py ignite/Administrator:Ignite@987@192.168.1.105 query -keyName HKLM\\SOFTWARE
```



```
root@kali:~/impacket/examples# reg.py ignite/Administrator:Ignite@987@192.168.1.105 query
-keyName HKLM\\SOFTWARE\\Policies\\Microsoft\\Windows -s
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

SOFTWARE\\Policies\\Microsoft\\Windows\\Appx\\
SOFTWARE\\Policies\\Microsoft\\Windows\\BITS\\
SOFTWARE\\Policies\\Microsoft\\Windows\\CurrentVersion\\
SOFTWARE\\Policies\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\
SOFTWARE\\Policies\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Cache\\
SOFTWARE\\Policies\\Microsoft\\Windows\\DataCollection\\
SOFTWARE\\Policies\\Microsoft\\Windows\\EnhancedStorageDevices\\
    TCGSecurityActivationDisabled    REG_DWORD    0x0
SOFTWARE\\Policies\\Microsoft\\Windows\\IPSec\\
SOFTWARE\\Policies\\Microsoft\\Windows\\IPSec\\Policy\\
SOFTWARE\\Policies\\Microsoft\\Windows\\IPSec\\Policy\\Local\\
SOFTWARE\\Policies\\Microsoft\\Windows\\iSCSI\\
SOFTWARE\\Policies\\Microsoft\\Windows\\Network Connections\\
    NC_PersonalFirewallConfig        REG_DWORD    0x0
SOFTWARE\\Policies\\Microsoft\\Windows\\NetworkConnectivityStatusIndicator\\
    (Default)                        REG_SZ
SOFTWARE\\Policies\\Microsoft\\Windows\\NetworkProvider\\
SOFTWARE\\Policies\\Microsoft\\Windows\\NetworkProvider\\HardenedPaths\\
SOFTWARE\\Policies\\Microsoft\\Windows\\safer\\
SOFTWARE\\Policies\\Microsoft\\Windows\\safer\\codeidentifiers\\
    authenticodeenabled              REG_DWORD    0x0
SOFTWARE\\Policies\\Microsoft\\Windows\\SettingSync\\
    EnableBackupForWin8Apps          REG_DWORD    0x1
SOFTWARE\\Policies\\Microsoft\\Windows\\System\\
```

## rpcdump.py

RPC or Remote Procedure Call is when a computer program causes a procedure to execute in different address space which is coded as a normal procedure call. This script can enumerate those endpoints for us. It also matches them to some of the well-known endpoints in order to identify them.

### Requirements:

- Domain
- Username
- Password/Password Hash
- Target IP Address

### Syntax:

```
rpcdump.py [domain]/[user]:[Password/Password Hash]@[Target IP Address]
```

### Command:



```
rpcdump.py ignite/Administrator:Ignite@987@192.168.1.105
```

```
root@kali:~/impacket/examples# rpcdump.py ignite/Administrator:Ignite@987@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Retrieving endpoint list from 192.168.1.105
Protocol: N/A
Provider: N/A
UUID : E40F7B57-7A25-4CD3-A135-7F7D3DF9D16B v1.0 Network Connection Broker server endpoint
Bindings:
    ncalrpc:[LRPC-8538476c13e927aa7c]
    ncalrpc:[OLE19B8B9EAF57BA2F284B628B8354]
    ncalrpc:[LRPC-8dc1857a520f6b0258]
    ncalrpc:[LRPC-e2a01d54b6c94b0c01]

Protocol: N/A
Provider: N/A
UUID : 0D3E2735-CEA0-4ECC-A9E2-41A2D81AED4E v1.0
Bindings:
    ncacn_np:\\WIN-S0V7KMTVLD2[\\pipe\\LSM_API_service]
    ncalrpc:[LSMApi]
    ncalrpc:[LRPC-0cce775b10278cb09]
    ncalrpc:[actkernel]
    ncalrpc:[umpo]

Protocol: N/A
Provider: N/A
UUID : 880FD55E-43B9-11E0-B1A8-CF4EDFD72085 v1.0 KAPI Service endpoint
Bindings:
    ncalrpc:[LRPC-8538476c13e927aa7c]
    ncalrpc:[OLE19B8B9EAF57BA2F284B628B8354]
    ncalrpc:[LRPC-8dc1857a520f6b0258]
    ncalrpc:[LRPC-e2a01d54b6c94b0c01]

Protocol: N/A
Provider: N/A
UUID : 3A9EF155-691D-4449-8D05-09AD57031823 v1.0
Bindings:
    ncalrpc:[LRPC-4c0ab5dfb927efec0d]
    ncalrpc:[ubpmtaskhostchannel]
    ncacn_np:\\WIN-S0V7KMTVLD2[\\PIPE\\atsvc]
    ncacn_ip_tcp:192.168.1.105[49670]
    ncacn_np:\\WIN-S0V7KMTVLD2[\\pipe\\SessEnvPublicRpc]
```

## samrdump.py

Samrdump is an application that retrieves sensitive information about the specified target machine using the Security Account Manager (SAM). It is a remote interface that is accessible under the Distributed Computing Environment / Remote Procedure Calls (DCE/RPC) service. It lists out all the system shares, user accounts, and other useful information about the target's presence in the local network. The image clearly shows us all the user accounts that are held by the remote machine. Inspecting all the available shares for sensitive data and accessing other user accounts can further reveal valuable information.

### Requirements:

- Domain

- Username
- Password/Password Hash
- Target IP Address

Syntax:

`samrdump.py [domain]/[user]:[Password/Password Hash]@[Target IP Address]`

Command:

`samrdump.py ignite/Administrator:Ignite@987@192.168.1.105`

```
root@kali:~/impacket/examples# samrdump.py ignite/Administrator:Ignite@987@192.168.1.105
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation
```

```
[*] Retrieving endpoint list from 192.168.1.105
```

```
Found domain(s):
```

```
. IGNITE
. Builtin
```

```
[*] Looking up users in domain IGNITE
```

```
Found user: Administrator, uid = 500
Found user: Guest, uid = 501
Found user: krbtgt, uid = 502
Found user: DefaultAccount, uid = 503
Found user: yashika, uid = 1601
Found user: geet, uid = 1602
Found user: aarti, uid = 1603
Found user: $PI1000-3MFD4LDN1VTV, uid = 1625
Found user: SM_195ac04be8c140048, uid = 1626
Found user: SM_4c397e3a678c4b169, uid = 1627
Found user: SM_20db1747e41e4819a, uid = 1628
Found user: SM_8fbff1f05b7c418da, uid = 1629
Found user: SM_fafb5649db9644c49, uid = 1630
Found user: SM_c0b1758feadf42abb, uid = 1631
Found user: SM_555a8cdd81f14d9a8, uid = 1632
Found user: SM_8b7c24749eae46cfa, uid = 1633
Found user: SM_a5503dd828c64f048, uid = 1634
Found user: HealthMailboxf574a3a, uid = 1636
Found user: HealthMailbox06b7664, uid = 1637
Found user: HealthMailbox1eb4aa3, uid = 1638
Found user: HealthMailbox0a5a569, uid = 1641
Found user: HealthMailboxd7cfd99, uid = 1642
Found user: HealthMailbox41cc604, uid = 1643
Found user: HealthMailbox0ab8a6a, uid = 1644
Found user: HealthMailbox0bc5951, uid = 1645
Found user: HealthMailbox55e60d4, uid = 1646
Found user: HealthMailboxb6dd973, uid = 1647
Found user: HealthMailbox23061dc, uid = 1648
Found user: SVC_SQLService, uid = 1655
Found user: kavish, uid = 1656
```

```
Administrator (500)/FullName: Administrator
```

```
Administrator (500)/UserComment:
```

```
Administrator (500)/PrimaryGroupId: 513
```

```
Administrator (500)/BadPasswordCount: 0
```

**services.py**



The services script of the Impacket communicates with Windows services with the help of MSRPC Interface. It can start, stop, delete, read status, config, list, create and change any service. While working on Red Teaming assignments there were so many tasks that could have been simplified if only, we have access to the services of the Target machine. This makes it all a simple task.

### Requirements:

- Domain
- Username
- Password/Password Hash
- Target IP Address
- Action

### Syntax:

`services.py [domain]/[user]:[Password/Password Hash]@[Target IP Address] [Action]`

### Command:

```
services.py ignite/Administrator:Ignite@987@192.168.1.105 list
```

```
root@kali:~/impacket/examples# services.py ignite/Administrator:Ignite@987@192.168.1.105 list
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Listing services available on target
1394ohci - 1394 OHCI Compliant Host Controller - STOPPED
3ware - 3ware - STOPPED
ACPI - Microsoft ACPI Driver - RUNNING
AcpiDev - ACPI Devices driver - STOPPED
acpiex - Microsoft ACPIEx Driver - RUNNING
acpipagr - ACPI Processor Aggregator Driver - STOPPED
AcpiPmi - ACPI Power Meter Driver - STOPPED
acptime - ACPI Wake Alarm Driver - STOPPED
ADP80XX - ADP80XX - STOPPED
ADWS - Active Directory Web Services - RUNNING
AFD - Ancillary Function Driver for Winsock - RUNNING
ahcache - Application Compatibility Cache - RUNNING
AJRouter - AllJoyn Router Service - STOPPED
ALG - Application Layer Gateway Service - STOPPED
AmdK8 - AMD K8 Processor Driver - STOPPED
AmdPPM - AMD Processor Driver - STOPPED
amdsata - amdsata - STOPPED
amdsbs - amdsbs - STOPPED
amdxata - amdxata - STOPPED
AppHostSvc - Application Host Helper Service - RUNNING
AppID - AppID Driver - STOPPED
AppIDSvc - Application Identity - STOPPED
Appinfo - Application Information - STOPPED
applockerfltr - Smartlocker Filter Driver - STOPPED
AppMgmt - Application Management - STOPPED
AppReadiness - App Readiness - STOPPED
AppVClient - Microsoft App V Client - STOPPED
```

## ifmap.py

Ifmap scripts initially bind to the MGMT interface of the Target machine. Then it fetches a list of interface IDs. Then it adds those IDs to another large list of UUIDs it already has in its database. Then it tries to bind each of the

interfaces and reports the status of the interface. The status can be listed or listening. Its ability to gather information is unmatched. There is a Metasploit Module that works quite similar to this script is “auxiliary/scanner/dcerpc/endpoint\_mapper” The list of UUIDs (Universal Unique Identifier) which are running endpoint-mapper mapped to the unique services. After getting these services, an attacker can search on the internet to find if any of these services are vulnerable to Overflow over RPC.

**Requirements:**

- **Target IP Address**
- **Target Port**
- **Hostname (optional)**

**Syntax:**

**ifmap.py [Target IP Address] [Target Port]**

**Command:**

```
ifmap.py 192.168.1.105 135
```

```
root@kali:~/impacket/examples# ifmap.py 192.168.1.105 135
Procotol: N/A
Provider: rpcss.dll
UUID      : 00000136-0000-0000-C000-000000000046 v0.0: listed, listening

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

Procotol: N/A
Provider: rpcss.dll
UUID      : 0B0A6584-9E0F-11CF-A3CF-00805F68CB1B v1.0: other version listed, listening

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

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

Procotol: N/A
Provider: rpcss.dll
UUID      : 412F241E-C12A-11CE-ABFF-0020AF6E7A17 v0.0: other version listed, listening

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

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

Procotol: N/A
Provider: rpcss.dll
UUID      : 64FE0B7F-9EF5-4553-A7DB-9A1975777554 v1.0: listed, listening

Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
```

## opdump.py

This script binds to the given hostname:port and connects to the DCERPC (Distributed Computing Environment/Remote Procedure Calls) interface. After connecting, it tries to call each of the first 256 operation numbers in turn and reports the outcome of each call. This generates a burst of TCP connections to the given host:port!

It gives the output as follows:

op 0 (0x00): rpc\_x\_bad\_stub\_data

op 1 (0x01): rpc\_x\_bad\_stub\_data

op 2 (0x02): rpc\_x\_bad\_stub\_data

op 3 (0x03): success

op 4 (0x04): rpc\_x\_bad\_stub\_data

ops 5-255: nca\_s\_op\_rng\_error

rpc\_x\_bad\_stub\_data, rpc\_s\_access\_denied, and success generally means there's an operation at that number.

### Requirements:

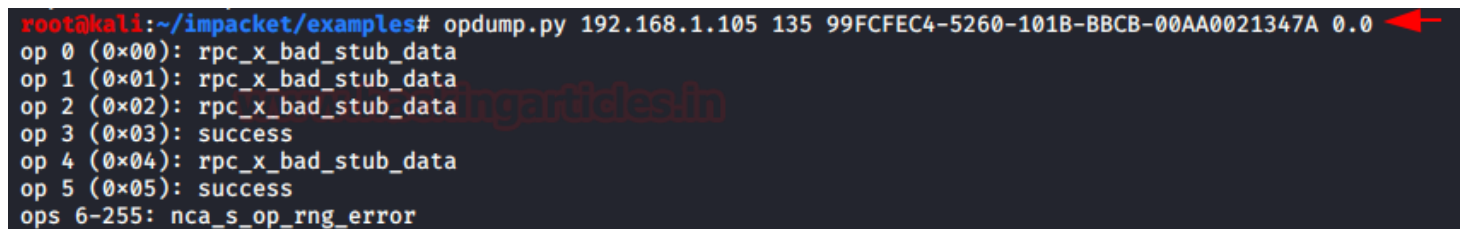
- IP Address
- Hostname (optional)
- Port Interface Version

### Syntax:

**opdump.py [Target IP Address] [Port Interface Version]**

### Command:

```
opdump.py 192.168.1.105 135 99FCFEC4-5260-101B-BBCB-00AA0021347A 0.0
```



```
root@kali:~/impacket/examples# opdump.py 192.168.1.105 135 99FCFEC4-5260-101B-BBCB-00AA0021347A 0.0
op 0 (0x00): rpc_x_bad_stub_data
op 1 (0x01): rpc_x_bad_stub_data
op 2 (0x02): rpc_x_bad_stub_data
op 3 (0x03): success
op 4 (0x04): rpc_x_bad_stub_data
op 5 (0x05): success
ops 6-255: nca_s_op_rng_error
```

## getArch.py

All PDUs (Protocol Data Unit) encoded with the NDR64 transfer syntax must use a value of 0x10 for the data representation format label. This value is used only in the transfers of the x64 bit systems. This script when provided with a target tries to communicate with the target system and collects the value of the data representation format label. Then it matches it to the NDR64 syntax stored in its code. Then it can provide the information to the attacker if the Operating System is a 64 bit or 32-bit system. We can also provide a list of targets and it can work simultaneously on all the targets.

### Requirements:

- Target IP Address

### Syntax:

**getArch.py -target [Target IP Address]**

**getArch.py -targets [Target List]**

### Command:

```
getArch.py -targets /root/Desktop/target.txt
```

```
root@kali:~/impacket/examples# cat /root/Desktop/target.txt
192.168.1.103
192.168.1.105
192.168.1.106
root@kali:~/impacket/examples# getArch.py -targets /root/Desktop/target.txt
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

[*] Gathering OS architecture for 3 machines
[*] Socket connect timeout set to 2 secs
192.168.1.103 is 64-bit
192.168.1.105 is 64-bit
192.168.1.106 is 64-bit
```

## netview.py

It is an enumeration tool. It requires the domain name to enumerate hosts. It can also be provided with a list of hosts or targets. Once a list is gathered then netview checks each of the following:

- IP addresses
- Shares
- Sessions
- Logged On Users

Once finding the information it doesn't stop. It keeps looping over the hosts found and keeps a detailed track of who logged in/out from remote servers. It keeps the connections with the target systems and it is very stealthy as it just sends few DCERPC packets. This script requires that the attacker machine is able to resolve the domain machine's NetBIOS names. This can be achieved by setting the DNS on the attacker machine to the domain DNS.

### Requirements:

- Domain
- Target IP Address
- Username

### Syntax:

```
netview.py [domain]/[User] -target [Target IP Address] -users [User List]
```

```
netview.py [domain]/[User] -targets [Target List] -users [User List]
```

### Command:

```
netview.py ignite/Administrator -targets /root/Desktop/target.txt -users /root/Des
```

```
root@kali:~/impacket/examples# netview.py ignite/Administrator -targets /root/Desktop/target.txt -users /root/Desktop/user.txt
Impacket v0.9.21.dev1+20200220.181330.03cbe6e8 - Copyright 2020 SecureAuth Corporation

Password:
[*] Importing targets
[*] Got 3 machines
[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
192.168.1.106: user IGNITE\Administrator logged in LOCALLY

[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
192.168.1.105: user IGNITE\Administrator logged in LOCALLY

[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
[*] SMB SessionError: STATUS_ACCOUNT_DISABLED(The referenced account is currently disabled and may not be logged on to.)
```

## Conclusion

In this article, we discussed the scripts in the Impacket Toolkit that can interact with the SMB/MSRPC services on a target system. Impacket has many categories which will further explore in due time.