

Hacking Articles

Raj Chandel's Blog

[Author](#)[Web Penetration Testing](#)[Penetration Testing](#)[Courses We Offer](#)[My Books](#)[Donate us](#)

Beginner Guide to impacket Tool kit

posted in **PENETRATION TESTING** on **JUNE 17, 2018** by **RAJ CHANDEL**  **SHARE**

Impacket is a collection of Python classes for working with network protocols. Impacket is focused on providing low-level programmatic access to the packets and for some protocols (e.g. SMB1-3 and MSRPC). According to the Core Security Website, Impacket supports protocols like IP, TCP, UDP, ICMP, IGMP, ARP, IPv4, IPv6, SMB, MSRPC, NTLM, Kerberos, WMI, LDAP etc.

For the following practical we will require two systems,

1. A Windows Server with Domain Controller Configured
2. A Kali Linux

Search

Subscribe to Blog via Email

SUBSCRIBE

Here, in our lab scenario we have configured the following settings on our systems.

Windows Server Details

- Domain: SERVER
- User: Administrator
- Password: T00r
- IP Address: 192.168.1.140

Kali Linux: 192.168.1.135

Before beginning with the Impacket tools, let's do a Nmap version scan on the target windows server to get the information about the services running on the Windows Server.

```
1 | nmap -sV 192.168.1.140
```



```

root@kali:~# nmap -sV 192.168.1.140
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-16 03:00 EDT
Nmap scan report for 192.168.1.140
Host is up (0.0056s latency).
Not shown: 984 filtered ports
PORT      STATE SERVICE      VERSION
53/tcp    open  domain?
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server t
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows netbios-ssn
389/tcp    open  ldap         Microsoft Windows Active Directory L
445/tcp    open  microsoft-ds Microsoft Windows Server 2008 R2 - 20
464/tcp    open  kpasswd5?
593/tcp    open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
636/tcp    open  tcpwrapped
3268/tcp   open  ldap         Microsoft Windows Active Directory L
3269/tcp   open  tcpwrapped
49154/tcp  open  msrpc        Microsoft Windows RPC
49155/tcp  open  msrpc        Microsoft Windows RPC
49157/tcp  open  msrpc        Microsoft Windows RPC
49158/tcp  open  ncacn_http   Microsoft Windows RPC over HTTP 1.0
49159/tcp  open  msrpc        Microsoft Windows RPC

```

As you can see in the above screenshot, we have domain services, Kerberos Services, Netbios Services, LDAP services and Windows RPC services.

Now let's install the Impacket tools from GitHub. You can get it from [here](#).

Firstly, clone the git, and then install the Impacket as shown in the screenshot.

```

1 | git clone https://github.com/CoreSecurity/impacket.git
2 | cd impacket/
3 | python setup.py install

```

Categories

- 🔖 BackTrack 5 Tutorials
- 🔖 Best of Hacking
- 🔖 Browser Hacking
- 🔖 Cryptography & Steganography
- 🔖 CTF Challenges
- 🔖 Cyber Forensics
- 🔖 Database Hacking
- 🔖 Domain Hacking
- 🔖 Email Hacking
- 🔖 Footprinting
- 🔖 Hacking Tools
- 🔖 Kali Linux
- 🔖 Nmap
- 🔖 Others
- 🔖 Penetration Testing
- 🔖 Social Engineering Toolkit
- 🔖 Trojans & Backdoors
- 🔖 Uncategorized
- 🔖 Website Hacking
- 🔖 Window Password Hacking
- 🔖 Windows Hacking Tricks
- 🔖 Wireless Hacking

```

root@kali:~# git clone https://github.com/CoreSecurity/impacket.git
Cloning into 'impacket'...
remote: Counting objects: 13693, done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 13693 (delta 30), reused 38 (delta 15), pack-reused 13623
Receiving objects: 100% (13693/13693), 4.63 MiB | 1.03 MiB/s, done.
Resolving deltas: 100% (10357/10357), done.
root@kali:~# cd impacket/
root@kali:~/impacket# python setup.py install
/usr/lib/python2.7/distutils/dist.py:267: UserWarning: Unknown distribution op
warnings.warn(msg)
/usr/lib/python2.7/dist-packages/setuptools/dist.py:397: UserWarning: Normaliz
normalized_version,
running install
running bdist_egg
running egg_info
creating impacket.egg-info
writing requirements to impacket.egg-info/requirements.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'

```

This will install Impacket on your Kali Linux, now after installation let's look at what different tools does Impacket have in its box.

cd impacket/examples

These are the some of the tools included in impacket, let's try some of them.

```

root@kali:~# cd impacket/examples/
root@kali:~/impacket/examples# ls
atexec.py      getST.py      mimikatz.py   ntlmrelayx.py registry-read.py
dcomexec.py    getTGT.py     mqtt_check.py opdump.py      reg.py
esentutl.py    GetUserSPNs.py mssqlclient.py ping6.py       rpcdump.py
GetADUsers.py goldenPac.py   mssqlinstance.py ping.py        sambaPipe.py
getArch.py     ifmap.py      netview.py    psexec.py      samrdump.py
GetNPUsers.py karmaSMB.py    nmapAnswerMachine.py raiseChild.py secretsdump.py
getPac.py      lookupsid.py  ntfs-read.py  rdp_check.py   services.py
root@kali:~/impacket/examples#

```

🔖 Youtube Hacking

Articles

Select Month



Facebook Page



Ping.py

Simple ICMP ping that uses the ICMP echo and echo-reply packets to check the status of a host. If the remote host is up, it should reply to the echo probe with an echo-reply packet.

`./ping.py`

```
root@kali:~/impacket/examples# ./ping.py ↵  
Use: ./ping.py <src ip> <dst ip>
```

Syntax: `./ping.py` [Source IP] [Destination IP]

```
1 | ./ping.py 192.168.1.135 192.168.1.140
```

Here we can see that we are getting the ICMP reply from 192.168.1.140 (Windows Server)

```
root@kali:~/impacket/examples# ./ping.py 192.168.1.135 192.168.1.140 ↵  
Ping reply for sequence #1  
Ping reply for sequence #2  
Ping reply for sequence #3  
Ping reply for sequence #4  
Ping reply for sequence #5
```

Lookupsid.py

A Windows SID bruteforcer example through [MS-LSAT] MSRPC Interface, aiming at finding remote users/groups.

`./lookupsid.py`

```

root@kali:~/impacket/examples# ./lookupsid.py ↵
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

usage: lookupsid.py [-h] [-target-ip ip address] [-port [destination port]]
                  [-domain-sids] [-hashes LMHASH:NTHASH] [-no-pass]
                  target [maxRid]

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>
  maxRid                max Rid to check (default 4000)

optional arguments:
  -h, --help            show this help message and exit

connection:
  -target-ip ip address
                        IP Address of the target machine. If omitted it will
                        use whatever was specified as target. This is useful
                        when target is the NetBIOS name and you cannot resolve
                        it
  -port [destination port]
                        Destination port to connect to SMB Server
  -domain-sids           Enumerate Domain SIDs (will likely forward requests to
                        the DC)


authentication:
  -hashes LMHASH:NTHASH
                        NTLM hashes, format is LMHASH:NTHASH
  -no-pass              don't ask for password (useful when proxying through
                        smbrelayx)

```

Syntax: ./lookupsid.py [[domain/] username [: password] @] [Target IP Address]

```
1 | ./lookupsid.py SERVER/Administrator: T00r@192.168.1.140
```

As you can see that the lookupsid tool had extracted the user and group information from the server

```
root@kali:~/impacket/examples# ./lookupsid.py SERVER/Administrator:T00r@192.168.1.140
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies 

[*] Brute forcing SIDs at 192.168.1.140
[*] StringBinding ncacn np:192.168.1.140[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-3172744464-3179878939-293551474
498: SERVER\Enterprise Read-only Domain Controllers (SidTypeGroup)
500: SERVER\Administrator (SidTypeUser)
501: SERVER\Guest (SidTypeUser)
502: SERVER\krbtgt (SidTypeUser)
512: SERVER\Domain Admins (SidTypeGroup)
513: SERVER\Domain Users (SidTypeGroup)
514: SERVER\Domain Guests (SidTypeGroup)
515: SERVER\Domain Computers (SidTypeGroup)
516: SERVER\Domain Controllers (SidTypeGroup)
517: SERVER\Cert Publishers (SidTypeAlias)
518: SERVER\Schema Admins (SidTypeGroup)
519: SERVER\Enterprise Admins (SidTypeGroup)
520: SERVER\Group Policy Creator Owners (SidTypeGroup)
521: SERVER\Read-only Domain Controllers (SidTypeGroup)
522: SERVER\Cloneable Domain Controllers (SidTypeGroup)
525: SERVER\Protected Users (SidTypeGroup)
553: SERVER\RAS and IAS Servers (SidTypeAlias)
571: SERVER\Allowed RODC Password Replication Group (SidTypeAlias)
572: SERVER\Denied RODC Password Replication Group (SidTypeAlias)
1000: SERVER\WinRMRemoteWMIUsers__ (SidTypeAlias)
1001: SERVER\PAVAN$ (SidTypeUser)
1102: SERVER\DnsAdmins (SidTypeAlias)
```

Psexec.py

It lets you execute processes on remote windows systems, copy files on remote systems, process their output and stream it back. It allows execution of remote shell commands directly with full interactive console without having to install any client software.

`./psexec.py`


```

root@kali:~/impacket/examples# ./psexec.py ↵
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

usage: psexec.py [-h] [-c pathname] [-path PATH] [-file FILE] [-debug]
               [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key]
               [-dc-ip ip address] [-target-ip ip address]
               [-port [destination port]] [-service-name service name]
               target [command [command ...]]

PSEXEC like functionality example using RemComSvc.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>
  command              command (or arguments if -c is used) to execute at the
                       target (w/o path) - (default:cmd.exe)

optional arguments:
  -h, --help            show this help message and exit
  -c pathname          copy the filename for later execution, arguments are
                       passed in the command option
  -path PATH           path of the command to execute
  -file FILE           alternative RemCom binary (be sure it doesn't require
                       CRT)
  -debug              Turn DEBUG output ON

authentication:
  -hashes LMHASH:NTHASH
                       NTLM hashes, format is LMHASH:NTHASH
  -no-pass            don't ask for password (useful for -k)
  -k                 Use Kerberos authentication. Grabs credentials from
                       ccache file (KRB5CCNAME) based on target parameters.
                       If valid credentials cannot be found, it will use the
                       ones specified in the command line
  -aesKey hex key    AES key to use for Kerberos Authentication (128 or 256
                       bits)

```

Syntax: ./psexec.py [[domain/] username [: password] @] [Target IP Address]

```
1 | ./psexec.py SERVER/Administrator: T00r@192.168.1.140
```

As you can see that we got a remote shell of the server in the given screenshot


```
root@kali:~/impacket/examples# ./psexec.py SERVER/Administrator:T00r@192.168.1.140
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies  ↑
[*] Requesting shares on 192.168.1.140.....
[*] Found writable share ADMIN$
[*] Uploading file bCQoweWQ.exe
[*] Opening SVCManager on 192.168.1.140.....
[*] Creating service EHkx on 192.168.1.140.....
[*] Starting service EHkx.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>exit
[*] Process cmd.exe finished with ErrorCode: 0, ReturnCode: 0
[*] Opening SVCManager on 192.168.1.140.....
[*] Stopping service EHkx.....
[*] Removing service EHkx.....
[*] Removing file bCQoweWQ.exe.....
```

Rpcdump.py

This script will dump the list of RPC endpoints and string bindings registered at the target. It will also try to match them with a list of well-known endpoints.

`./rpcdump.py`

```

root@kali:~/impacket/examples# ./rpcdump.py ↵
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

usage: rpcdump.py [-h] [-debug] [-target-ip ip address]
                  [-port [destination port]] [-hashes LMHASH:NTHASH]
                  target

Dumps the remote RPC endpoints information.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>

optional arguments:
  -h, --help            show this help message and exit
  -debug                Turn DEBUG output ON

connection:
  -target-ip ip address
                        IP Address of the target machine. If omitted it will
                        use whatever was specified as target. This is useful
                        when target is the NetBIOS name and you cannot resolve
                        it
  -port [destination port]
                        Destination port to connect to SMB Server

authentication:
  -hashes LMHASH:NTHASH
                        NTLM hashes, format is LMHASH:NTHASH

```

Syntax: ./rpcdump.py [[domain/] username [: password] @] [Target IP Address]

```
1 | ./rpcdump.py SERVER/Administrator: T00r@192.168.1.140
```

As you can see below we have the list of RPC targets

```
root@kali:~/impacket/examples# ./rpcdump.py SERVER/Administrator:T00r@192.168.1.140
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[*] Retrieving endpoint list from 192.168.1.140
Protocol: N/A
Provider: N/A
UUID      : 0D3E2735-CEA0-4ECC-A9E2-41A2D81AED4E v1.0
Bindings:
    ncalrpc:[actkernel]
    ncalrpc:[umpp]

Protocol: [MS-RAA]: Remote Authorization API Protocol
Provider: N/A
UUID      : 0B1C2170-5732-4E0E-8CD3-D9B16F3B84D7 v0.0 RemoteAccessCheck
Bindings:
    ncalrpc:[NETLOGON_LRPC]
    ncacn_np:\\PAVAN[\pipe\d78b9f1df8194195]
    ncacn_http:192.168.1.140[49158]
    ncalrpc:[NTDS_LPC]
    ncacn_ip_tcp:192.168.1.140[49157]
    ncacn_ip_tcp:192.168.1.140[49155]
    ncalrpc:[OLEE8C47F27A0DFF8D17F336A95D70E]
```

Samrdump.py

An application that communicates with the Security Account Manager Remote interface from the MSRPC suite. It lists system user accounts, available resource shares and other sensitive information exported through this service.

`./samrdump.py`

```

root@kali:~/impacket/examples# ./samrdump.py ↵
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

usage: samrdump.py [-h] [-csv] [-debug] [-dc-ip ip address]
                  [-target-ip ip address] [-port [destination port]]
                  [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key]
                  target

This script downloads the list of users for the target system.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>

optional arguments:
  -h, --help            show this help message and exit
  -csv                  Turn CSV output
  -debug                Turn DEBUG output ON

connection:
  -dc-ip ip address     IP Address of the domain controller. If ommited it use
                        the domain part (FQDN) specified in the target
                        parameter
  -target-ip ip address IP Address of the target machine. If ommited it will
                        use whatever was specified as target. This is useful
                        when target is the NetBIOS name and you cannot resolve
                        it
  -port [destination port]
                        Destination port to connect to SMB Server

```

Syntax: ./samrdump.py [[domain/] username [: password] @] [Target IP Address]

1 | `./samrdump.py SERVER/Administrator: T00r@192.168.1.140`

As you can see below we have extracted SAM information form the Target Server

```

root@kali:~/impacket/examples# ./samrdump.py SERVER/Administrator:T00r@192.168.1.140
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[*] Retrieving endpoint list from 192.168.1.140
Found domain(s): SERVER
. Builtin
[*] Looking up users in domain SERVER
Found user: Administrator, uid = 500
Found user: Guest, uid = 501
Found user: krbtgt, uid = 502
Administrator (500)/FullName:
Administrator (500)/UserComment:
Administrator (500)/PrimaryGroupId: 513
Administrator (500)/BadPasswordCount: 0
Administrator (500)/LogonCount: 9
Administrator (500)/PasswordLastSet: 2018-06-14 17:44:22
Administrator (500)/PasswordDoesNotExpire: False
Administrator (500)/AccountIsDisabled: False
Administrator (500)/ScriptPath:
Guest (501)/FullName:
Guest (501)/UserComment:
Guest (501)/PrimaryGroupId: 514
Guest (501)/BadPasswordCount: 0
Guest (501)/LogonCount: 0
Guest (501)/PasswordLastSet: <never>
Guest (501)/PasswordDoesNotExpire: True
Guest (501)/AccountIsDisabled: True
Guest (501)/ScriptPath:
krbtgt (502)/FullName:
krbtgt (502)/UserComment:
krbtgt (502)/PrimaryGroupId: 513

```

Sniff.py

Simple packet sniffer that uses the pcap library to listen for packets in transit over the specified interface.

`./sniff.py`

Choose the interface using the number associated with it. And the sniffing starts.

```

root@kali:~/impacket/examples# ./sniff.py
0 - eth0
1 - any

```

```

2 - lo
3 - nflog
4 - nfqueue
5 - usbmon1
6 - usbmon2
Please select an interface: 0 ↩
Listening on eth0: net=192.168.1.0, mask=255.255.255.0, linktype=1
Ether: 00:0c:29:13:2b:86 -> 00:0c:29:60:22:42
IP DF 192.168.1.135 -> 192.168.1.140
ICMP type: ECHO code: UNKNOWN

df91 235b 0000 0000 ae1a 0b00 0000 0000    ..#[.....
1011 1213 1415 1617 1819 1a1b 1c1d 1e1f    .....
2021 2223 2425 2627 2829 2a2b 2c2d 2e2f    !"#$%&'()*+,-./
3031 3233 3435 3637    01234567

Ether: 00:0c:29:60:22:42 -> ff:ff:ff:ff:ff:ff
ARP format: ARPHRD ETHER opcode: REQUEST
0:c:29:60:22:42 -> 0:0:0:0:0:0
192.168.1.140 -> 192.168.1.135

0000 0000 0000 0000 0000 0000 0000 0000    .....
0000    ..

Ether: 00:0c:29:13:2b:86 -> 00:0c:29:60:22:42
ARP format: ARPHRD ETHER opcode: REPLY
0:c:29:13:2b:86 -> 0:c:29:60:22:42
192.168.1.135 -> 192.168.1.140

Ether: 00:0c:29:60:22:42 -> 00:0c:29:13:2b:86
IP 192.168.1.140 -> 192.168.1.135
ICMP type: ECHOREPLY code: UNKNOWN

df91 235b 0000 0000 ae1a 0b00 0000 0000    ..#[.....
1011 1213 1415 1617 1819 1a1b 1c1d 1e1f    .....
2021 2223 2425 2627 2829 2a2b 2c2d 2e2f    !"#$%&'()*+,-./
3031 3233 3435 3637    01234567

```

Sniffer.py

Simple packet sniffer that uses a raw socket to listen for packets in transit corresponding to the specified protocols.

`./sniffer.py`

And the sniffer starts to monitor icmp, tcp and udp


```

root@kali:~/impacket/examples# ./sniffer.py ↵
Using default set of protocols. A list of protocols can be supplied
Listening on protocols: ('icmp', 'tcp', 'udp')
IP 192.168.1.140 -> 192.168.1.135
ICMP type: ECHOREPLY code: UNKNOWN

1a8d 235b 0000 0000 2ef5 0600 0000 0000    ..#[.....
1011 1213 1415 1617 1819 1a1b 1c1d 1e1f    .....
2021 2223 2425 2627 2829 2a2b 2c2d 2e2f    !"#$%&'()*+,-./
3031 3233 3435 3637    01234567

IP 192.168.1.140 -> 192.168.1.135
ICMP type: ECHOREPLY code: UNKNOWN

1b8d 235b 0000 0000 02ff 0600 0000 0000    ..#[.....
1011 1213 1415 1617 1819 1a1b 1c1d 1e1f    .....
2021 2223 2425 2627 2829 2a2b 2c2d 2e2f    !"#$%&'()*+,-./
3031 3233 3435 3637    01234567

IP 192.168.1.140 -> 192.168.1.135
ICMP type: ECHOREPLY code: UNKNOWN

1c8d 235b 0000 0000 5003 0700 0000 0000    ..#[....P.....
1011 1213 1415 1617 1819 1a1b 1c1d 1e1f    .....
2021 2223 2425 2627 2829 2a2b 2c2d 2e2f    !"#$%&'()*+,-./
3031 3233 3435 3637    01234567

IP DF 139.59.75.99 -> 192.168.1.135
UDP 123 -> 44926

```

Wmiexec.py

It generates a semi-interactive shell, used through Windows Management Instrumentation. It does not require to install any service/agent at the target server. It runs as Administrator. It is highly stealthy.

./wmiexec.py

```
root@kali:~/impacket/examples# ./wmiexec.py ↵
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

usage: wmiexec.py [-h] [-share SHARE] [-nooutput] [-debug] [-codec CODEC]
                [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key]
                [-dc-ip ip address] [-A authfile]
                target [command [command ...]]

Executes a semi-interactive shell using Windows Management Instrumentation.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>
  command              command to execute at the target. If empty it will
                        launch a semi-interactive shell

optional arguments:
  -h, --help            show this help message and exit
  -share SHARE          share where the output will be grabbed from (default
                        ADMIN$)
  -nooutput            whether or not to print the output (no SMB connection
                        created)
  -debug              Turn DEBUG output ON
  -codec CODEC        Sets encoding used (codec) from the target's output
                        (default "UTF-8"). If errors are detected, run
                        chcp.com at the target, map the result with
                        https://docs.python.org/2.4/lib/standard-
                        encodings.html and then execute wmiexec.py again with
                        -codec and the corresponding codec
```

Syntax: ./wmiexec.py [[domain/] username [: password] @] [Target IP Address]

```
1 | ./wmiexec.py SERVER/Administrator: T00r@192.168.1.140
```

As you can see below that we have the shell from the Target Server

```

root@kali:~/impacket/examples# ./wmiexec.py SERVER/Administrator:T00r@192.168.1.140
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>

```

Wmiquery.py

It allows to issue WQL queries and get description of WMI objects at the target system.

`./wmiquery.py`

```

root@kali:~/impacket/examples# ./wmiquery.py
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

usage: wmiquery.py [-h] [-namespace NAMESPACE] [-file FILE] [-debug]
                  [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key]
                  [-dc-ip ip address]
                  [-rpc-auth-level [{integrity,privacy,default}]]
                  target

Executes WQL queries and gets object descriptions using Windows Management
Instrumentation.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>

optional arguments:
  -h, --help            show this help message and exit
  -namespace NAMESPACE namespace name (default //./root/cimv2)
  -file FILE            input file with commands to execute in the WQL shell
  -debug               Turn DEBUG output ON

```

Syntax: `./wmiquery.py [[domain/] username [: password] @] [Target IP Address]`

```
1 | ./wmiquery.py SERVER/Administrator: T00r@192.168.1.140
```

This will open a shell, where you can run WQL queries like

```
1 | SELECT * FROM Win32_LogicalDisk WHERE FreeSpace < 209152
```

```

root@kali:~/impacket/examples# ./wmiquery.py SERVER/Administrator:T00r@192.168.1.140
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[!] Press help for extra shell commands
WQL> SELECT * FROM Win32_LogicalDisk WHERE FreeSpace < 209152
| Caption | Description | InstallDate | Name | Status | Availability | CreationClassName | ConfigManag
| DeviceID | PowerManagementCapabilities | PNPDeviceID | PowerManagementSupported | StatusInfo | Syste
ErrorCodes | ErrorDescription | ErrorCleared | Access | BlockSize | ErrorMethodology | NumberOfBlocks |
ed | DriveType | FileSystem | MaximumComponentLength | ProviderName | SupportsFileBasedCompression | V
Type | SupportsDiskQuotas | QuotasDisabled | QuotasIncomplete | QuotasRebuilding | VolumeDirty |
| D: | CD-ROM Disc | 0 | D: | 0 | 0 | Win32_LogicalDisk | 0 | 0 | D: | 0 | 0 | 0 | 0 | Win32_ComputerS
| 0 | 0 | 0 | 4542291968 | 0 | 5 | UDF | 254 | 0 | 0 | IR3_SSS_X64FREE_EN-US_DV9 | F49862AF | 11 | 0 |
WQL>

```

Atexec.py

This example executes a command on the target machine through the Task Scheduler service and returns the output of the executed command.

`./atexec.py`

```

root@kali:~/impacket/examples# ./atexec.py ↩
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[!] This will work ONLY on Windows >= Vista
usage: atexec.py [-h] [-debug] [-hashes LMHASH:NTHASH] [-no-pass] [-k]
               [-aesKey hex key] [-dc-ip ip address]
               target [command [command ...]]

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>
  command              command to execute at the target

optional arguments:
  -h, --help            show this help message and exit
  -debug               Turn DEBUG output ON

authentication:
  -hashes LMHASH:NTHASH
                        NTLM hashes, format is LMHASH:NTHASH
  -no-pass             don't ask for password (useful for -k)
  -k                   Use Kerberos authentication. Grabs credentials from
                        ccache file (KRB5CCNAME) based on target parameters.
                        If valid credentials cannot be found, it will use the
                        ones specified in the command line
  -aesKey hex key     AES key to use for Kerberos Authentication (128 or 256
                        bits)
  -dc-ip ip address    IP Address of the domain controller. If omitted it
                        will use the domain part (FQDN) specified in the
                        target parameter

```

Syntax: /atexec.py [[domain/] username [: password] @] [Target IP Address] [Command]

```
1 | ./atexec.py SERVER/Administrator: T00r@192.168.1.140 systeminfo
```

As you can see below that a remote connection was established to the server and the command systeminfo was run on the Target server with the output of the command delivered on the Kali terminal.

```
root@kali:~/impacket/examples# ./atexec.py SERVER/Administrator:T00r@192.168.1.140 systeminfo
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[!] This will work ONLY on Windows >= Vista
[*] Creating task \QwDMERik
[*] Running task \QwDMERik
[*] Deleting task \QwDMERik
[*] Attempting to read ADMIN$\Temp\QwDMERik.tmp
[*] Attempting to read ADMIN$\Temp\QwDMERik.tmp

Host Name:                PAVAN
OS Name:                   Microsoft Windows Server 2012 R2 Standard Evaluation
OS Version:                6.3.9600 N/A Build 9600
OS Manufacturer:          Microsoft Corporation
OS Configuration:         Primary Domain Controller
OS Build Type:              Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                 00252-10000-00000-AA228
Original Install Date:      6/14/2018, 2:44:22 PM
System Boot Time:           6/15/2018, 11:34:27 AM
System Manufacturer:        VMware, Inc.
System Model:                VMware Virtual Platform
System Type:                 x64-based PC
```

getArch.py

This script will connect against a target (or list of targets) machine/s and gather the OS architecture type installed by (ab) using a documented MSRPC feature.

`./getArch.py`

```
root@kali:~/impacket/examples# ./getArch.py ↵
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

usage: getArch.py [-h] [-target TARGET] [-targets TARGETS] [-timeout TIMEOUT]
               [-debug]

Gets the target system's OS architecture version

optional arguments:
  -h, --help            show this help message and exit
  -target TARGET         <targetName or address>
  -targets TARGETS       input file with targets system to query Arch from (one per
                        line)
  -timeout TIMEOUT       socket timeout out when connecting to the target (default
                        2 sec)
  -debug                Turn DEBUG output ON
```

Syntax: ./getArch.py -target [IP Address]

Command: ./getArch.py -target 192.168.1.140

Here we can see that the architecture of the target system is 64-bit

```
root@kali:~/impacket/examples# ./getArch.py -target 192.168.1.140 ↵
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[*] Gathering OS architecture for 1 machines
[*] Socket connect timeout set to 2 secs
192.168.1.140 is 64-bit
```

Ifmap

This script will bind to the target's MGMT interface to get a list of interface IDs. It will use that list on top of another list of interfaces UUID and reports whether the interface is listed and/or listening.

Syntax: ./ifmap.py [Host IP Address] [Port]

```
root@kali:~/impacket/examples# ./ifmap.py ↵
usage: ./ifmap.py <host> <port>
```



```
1 | ./ifmap.py 192.168.1.140 135
2 | ./ifmap.py 192.168.1.140 49154
```

```
root@kali:~/impacket/examples# ./ifmap.py 192.168.1.140 49154 ↵
Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: N/A
UUID      : 00000131-0000-0000-C000-000000000046 v0.0: listed, listening

Protocol: N/A
Provider: N/A
UUID      : 00000132-0000-0000-C000-000000000046 v0.0: listed, listening

Protocol: N/A
Provider: N/A
UUID      : 00000134-0000-0000-C000-000000000046 v0.0: listed, listening

Protocol: N/A
Provider: N/A
UUID      : 00000141-0000-0000-C000-000000000046 v0.0: listed, listening

Protocol: [MS-DCOM]: Distributed Component Object Model (DCOM) Remote
Provider: N/A
UUID      : 00000143-0000-0000-C000-000000000046 v0.0: listed, listening
```

Author: Pavandeep Singh is a Technical Writer, Researcher and Penetration Tester
Contact [here](#)

Share this:



Like this:

Loading...

ABOUT THE AUTHOR



RAJ CHANDEL

Raj Chandel is a Skilled and Passionate IT Professional especially in IT-Hacking Industry. At present other than his name he can also be called as An Ethical Hacker, A Cyber Security Expert, A Penetration Tester. With years of quality Experience in IT and software industry

PREVIOUS POST

← [LINUX PRIVILEGE ESCALATION
USING LD_PRELOAD](#)

NEXT POST

[HACK THE BOX CHALLENGE:
CHATTERBOX WALKTHROUGH](#) →

1 Comment → [BEGINNER GUIDE TO IMPACKET TOOL KIT](#)



FIRDOSH ANSARI

June 26, 2018 at 9:27 am

Sir You article is so nice and this website is os useful for education

REPLY ↓

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

- ☐ Notify me of follow-up comments by email.
 - ☐ Notify me of new posts by email.
-