Windows Privilege Escalation Tricks

Jake Williams (@MalwareJake)
Rendition Infosec
www.rsec.us
@RenditionSec
$whoami

- Founder and President of Rendition Infosec
- IANS Faculty
- Soon to be former SANS Instructor
- Endorsed by the Shadow Brokers
- Former NSA hacker, Master CNE operator, recipient of the DoD Exception Civilian Service Medal
- **Dislikes:** those who call themselves “thought leaders,” “crypto bros,” and anyone who **needlessly adds blockchain** to a software solution
Quick Rant...

• Hacker culture is not infosec culture and the difference isn’t between white had and black hat
  – Infosec has a job to do (securing stuff) and sometimes it’s okay to just push the magic button and hope everything works...

• We got to hacker culture by actually **understanding how things work at a very low level** (the rules) and then systematically breaking the rules
  – The “hacker culture” is dying because too many people today just want to push the magic button...

• Let’s reverse this course
Why focus on misconfigurations?

Specific Misconfigurations
- Permissions are EVERYTHING
- %PATH% hijacks
- Services
- Installation flags
- Unattend.xml, sysprep, and upgrades

Uninstalled software that really isn’t...

Conclusion
Why focus on misconfigurations?

Should we really focus on system admins failing?
• Most of this talk will focus on misconfigurations commonly found in enterprise environments

• While unpatched binary vulnerabilities and 0 days are fun, too many penetration testers today rely on them for 100% of privilege escalation
  – There are FAR more misconfigurations than vulns
  – Vulns will be detected and patched by Nessus, Nexpose, etc.
  – Misconfigurations typically are missed by scanners
Many of the misconfigurations we discuss are actually recommended by training that system admins take.

Remember we’re in the security echo chamber here.

Security is all about CIA:
- In infosec, we have a confidentiality fetish...
- Most systems admins have only one KPI: availability.

Configurations that help availability are the **ONE** thing that matters to most IT personnel.
Permission all the things!

Are we drinking all the booze too?
Quick Disclaimer

• I have no idea who originally found all of these techniques (though it’s safe to assume many people over the years have independently discovered them)
  – Many of these tricks were collected in the PowerUp script from Will Schroeder (though using that unmodified asking to get caught)
  – Matt Graeber has also done a good deal of work in this area

• Thanks to everyone who has taken time to document privilege escalation tricks over the years!
NTFS Filesystem permissions

- NTFS offers more than just Read, Write, and Full Control
- A number of “special permissions” are generally of more concern than general permissions
The icacls tool can be used to obtain permissions from the command line (useful for scripts)

```
C:\Users\rendition>icacls C:\CustomApp
C:\CustomApp BUILTIN\Administrators:(I)(F)
BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
NT AUTHORITY\SYSTEM:(I)(F)
NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
BUILTIN\Users:(I)(OI)(CI)(RX)
NT AUTHORITY\Authenticated Users:(I)(M)
NT AUTHORITY\Authenticated Users:(I)(OI)(CI)(IO)(M)
```

Successfully processed 1 files; Failed processing 0 files
Specific permissions we care about:

- Delete: does removing a config file default to generic configs?
- Write Data/Add File: add a DLL that is sideloaded into an application with elevated privileges
- Write Attributes: set file attributes to hidden or system, potentially hiding them from view by most users
- Append Data: allows the user to add data to the end of a file, but not overwrite any existing data
- Change Permissions: this is pretty much full control
Registry permissions

• Registry keys have permissions
  – But values do not

• While we find poor directory permissions regularly, finding registry keys writeable by the Authenticated Users group is more unusual
  – If a registry key or value that controls the execution of a privileged process has write permissions, we can run our code with elevated permissions

• Don’t forget to check plugins
  – You may be able to add a DLL!
Registry permissions (2)

- The subinacl tool (part of the Windows resource kit) can get permissions for registry keys

```
C:\Users>subinacl /keyreg HKEY_LOCAL_MACHINE\software\microsoft
SeSecurityPrivilege : Access is denied.

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

=================================================================
+KeyReg HKEY_LOCAL_MACHINE\software\microsoft
=================================================================
/control=0x400 SE_DACL_AUTO_INHERITED-0x0400
/owner =builtin\administrators
/primary group =builtin\administrators
/perm. ace count =7
/pase =builtin\users ACCESS_ALLOWED_ACE_TYPE-0x0
```
Registry permissions (3)

- PowerShell can also be used to obtain ACL information on registry keys, though the output is meh

```
PS C:\Users\rendition> Get-Acl HKLM:\software |format-list

Path        : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\software
Owner       : BUILTIN\Administrators
Group       : NT AUTHORITY\SYSTEM
Access      :
            Creator Owner Allow  268435456
            NT AUTHORITY\SYSTEM Allow  268435456
            NT AUTHORITY\SYSTEM Allow  FullControl
            BUILTIN\Administrators Allow  268435456
            BUILTIN\Administrators Allow  FullControl
            BUILTIN\Users Allow        -247483648
            BUILTIN\Users Allow        ReadKey
Audit       :
Sddl        :
            O:BAG:SYD:PAI(A;CIIO;GA;;;CO)(A;CIIO;GA;;;SY)(A;;KA;;;SY)(A;CIIO;GA;;BU)
```
%PATH% hijacks

• If any portion of the SYSTEM %PATH% variable is writeable by Authenticated Users, it is almost certain that privilege escalation opportunities exist

• A horrifying number of programs call other applications without specifying a full path

• Some system admins don’t leave system32 as the first directory in the PATH, but it almost always should be
When checking the PATH variable, ensure that you are looking at the GLOBAL path and not custom user settings:

- `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment`

```cmd
C:\Users\rendition>reg query "HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment"
```

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment
 ComSpec REG_EXPAND_SZ %SystemRoot%\system32\cmd.exe
 FP_NO_HOST_CHECK REG_SZ NO
 NUMBER_OF_PROCESSORS REG_SZ 2
 OS REG_SZ Windows_NT
 Path REG_EXPAND_SZ %SystemRoot%\system32;\%SystemRoot%\System32\Wbem;\%SYSTEMROOT%
2\WindowsPowerShell\v1.0\%
 PATHEXEC REG_SZ .COM; .EXE; .BAT; .CMD; .VBS; .VBE; .JS; .JSE; .WSF; .WSH; .MSC
 PROCESSOR_ARCHITECTURE REG_SZ AMD64
 PROCESSOR_IDENTIFIER REG_SZ Intel64 Family 6 Model 70 Stepping 1, GenuineIntel
 PROCESSOR_LEVEL REG_SZ 6
 PROCESSOR_REVISION REG_SZ 4601
 PSModulePath REG_EXPAND_SZ %SystemRoot%\system32\WindowsPowerShell\v1.0\Modules\ TEMP REG_EXPAND_SZ %SystemRoot%\Temp
 TMP REG_EXPAND_SZ %SystemRoot%\Temp
 USERNAME REG_SZ SYSTEM
 windir REG_EXPAND_SZ %SystemRoot%```
If a non-standard directory is found in the path, check permissions to see if the Users group has write permissions

- Because C:\mydir is first in the path, this is especially dangerous

```bash
C:\Users\rendition>icacls c:\mydir
BUILTIN\Users:(W,D)
BUILTIN\Administrators:(I)(F)
BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
NT AUTHORITY\SYSTEM:(I)(F)
NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
BUILTIN\Users:(I)(OI)(CI)(RX)
NT AUTHORITY\Authenticated Users:(I)(M)
NT AUTHORITY\Authenticated Users:(I)(OI)(CI)(IO)(M)
Successfully processed 1 files; Failed processing 0 files
```
Abusing Services

• Attackers may try a number of ways to abuse services
  – Service executable replacement
  – DLL added to service executable directory
  – Service configuration file modified
  – Registry key allows replacement of theImagePath value
  – Unquoted service path
Abusing Services – Executable replacement

• Attackers query service configurations and discover the service executable in the ImagePath value

• If the attacker has write permissions to the service executable, they can overwrite it

• One of the biggest challenges is that executables are locked “open exclusive” by Windows while running
  – Any service set to autostart will be running and can’t be replaced...
Abusing Services – DLL added to service directory

- If the service executable is in a directory where the user has write permissions, they may add a DLL that is loaded into the service process.

- Unless the attacker takes pains to build a proxy DLL, the service will crash, so this is best used to add an account, change a configuration, etc. (no callback sessions)

- This has the distinct advantage of not requiring the service to be stopped before implementing.
• First, we look for services that might sit in a directory with interesting permissions, most often third party

```bash
C:\Users\rendition>wmic service get pathname | findstr /i f-response
C:\Program Files\F-Response\f-response-lm-srv.exe
C:\Windows\SysWow64\f-response-ent.exe
```

• Next, we get directory permissions

```bash
C:\Users\rendition>icacls "c:\program files\f-response"
c:\program files\f-response BUILTIN\Users:(WD,AD)
   NT SERVICE\TrustedInstaller:(I)(F)
   NT SERVICE\TrustedInstaller:(I)(CI)(IO)(F)
   NT AUTHORITY\SYSTEM:(I)(F)
   NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
   BUILTIN\Administrators:(I)(F)
   BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
   BUILTIN\Users:(I)(RX)
   BUILTIN\Users:(I)(OI)(CI)(IO)(GR,GE)
   CREATOR OWNER:(I)(OI)(CI)(IO)(F)
```
Abusing Services – Modify Service Config

- If the service uses a configuration file and the attacker has write access to the file, they can add or modify entries that may result in code execution.

- This obviously requires a very specific service to be successful.

- Of all the service privilege escalation techniques, this is the least likely to be successful.
  - But it’s still worth a try!
Abusing Services – Modify Service Registry Value

• The ImagePath registry value for the service controls the executable that is launched

• In some rare cases, service registry keys may be created that allow the currently logged in user write access to values

• By replacing the ImagePath value, the attacker’s executable will by executed with elevated permissions the next time the service is restarted
Abusing Services – Unquoted Service Path

- Windows does not handle spaces in executable names and paths gracefully at all

- Windows treats each space in an unquoted file name as the possible beginning of command line arguments
  - `wmic service get name,displayname,pathname,startmode | findstr /i /v Windows | findstr /c:"" /v

<table>
<thead>
<tr>
<th>DisplayName</th>
<th>Name</th>
<th>StartMode</th>
<th>PathName</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-Response License Manager Service 1.12</td>
<td>F-ResponseLM</td>
<td>Manual</td>
<td>C:\Program Files\F-Response\f-re</td>
</tr>
<tr>
<td>Mikogo-Service</td>
<td>Mikogo-Service</td>
<td>Auto</td>
<td>C:\Users\rendition\AppData\Roami</td>
</tr>
<tr>
<td>ng\Mikogo\Mikogo-Service.exe</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Abusing Services – Unquoted Service Path (2)

- Consider C:\program files (x86)\custom app\my custom app.exe
  - C:\program.exe
  - C:\program files.exe
  - C:\program files (x86)\custom.exe
  - C:\program files (x86)\custom app\my.exe
  - C:\program files (x86)\custom app\my custom.exe
**InstallAlwaysElevated**

- The InstallAlwaysElevated registry key allows software packages to be installed using with system privileges
  - HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer
  - HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer
- HKLM is for managed applications (e.g. those deployed via GPO) while HKCU applies for everything else
- If these are configured, msfvenom can be used to build an MSI package which will elevate permissions
Unattended setup/upgrades

- Unattend.xml files are used to answer questions asked during the Windows installation
  - One of these is the login password for the local administrator account

- If an unattend.xml file is found and is readable, it is trivial to recover the admin password and elevate
Unattended setup/upgrades (2)

• Upgraded versions of Windows will move the original Windows directory to Windows.old
  – The windows.old directory contains many sensitive files, including registry hives

• Although default filesystem permissions should prevent access to the files, we’ve observed these directories set to world readable (along with subdirectories and files)
Many software packages use COM objects
   – These COM objects are controlled by registry settings, specifically InProcServer32 values

In some cases, stale COM entries are left behind in the registry (even though the files on disk are gone)

When this occurs, we can trick the system into executing code in a privileged context if we have appropriate (well, technically inappropriate) filesystem permissions
The original work on this is from Jimmy Bayne (@bohops)


Jimmy notes that this is usually best used as a stealthy persistence technique due to standard directory permissions
- But as we’ve already discussed, directory permissions may not be configured in the standard manner for any number of reasons
Where to go from here?

- We regularly use ProcMon from sysinternals in our evaluation of third party software

- Often unquoted paths (or absence of a full path) combined with weak filesystem permissions, allows for the exploitation of custom/third-party applications that run in privileged context
Conclusion

Let’s wrap this up...
Windows privilege escalation is often easier than it should be.

If you don’t hunt down the privilege escalation opportunities on your machines, attackers will.

If you are an attacker, good news – most system admins still suck at this...

@MalwareJake
@RenditionSec
www.rsec.us