The Dark Side of NTFS (Microsoft's Scarlet Letter) by **H. Carvey** on **10/10/02** ← back

Introduction

Microsoft platforms continue to proliferate and multiply. Corporate server and desktop systems are running Windows NT (NT) and Windows 2000 (2K), while home user and student systems are running Windows XP (XP). These platforms are extremely popular, and in widespread use. However, very little is known by the administrators and users of these systems about a feature of the NTFS file system called 'alternate data streams'.

NTFS is the preferred file system due to its stability, functionality, and the level of security it provides. NTFS alternate data streams (ADSs) are provided for compatibility with the Macintosh Hierarchical File System (HFS), which uses resource forks to maintain information associated with a file, such as icons, etc (RUSS00). While Microsoft provides a means for creating specific ADSs via Windows Explorer, the necessary tools and functionality for detecting the presence of arbitrary ADSs is conspicuously absent. Oddly enough, the operating systems have the necessary native functionality and tools to allow a user to create ADSs and to execute code hidden within those streams. Microsoft KnowledgeBase article Q101353 acknowledges the fact that the Win32 base API supports ADSs inconsistently.

The purpose of this paper is to describe in detail how ADSs are created and manipulated, and how code hidden in ADSs can be executed. Specific differences in the treatment of ADSs by NT, 2K, and XP will be noted.

Creating ADSs

The syntax used to create ADSs is relatively simple and straightforward. To create an ADS associated with the file 'myfile.txt', simply separate the default stream name from the ADS name with a colon.

```
c:\ads>echo This is an ADS > myfile.txt:hidden
```

Additionally, an ADS can be created using the contents of another file.

```
c:\ads>echo This is a test file > test.txt

c:\ads>type test.txt > myfile.txt:hidden
```

The ADS can then be verified using Notepad.

```
c:\ads>notepad myfile.txt:hidden
```

However, none of the variations of the 'dir' command nor any available switches or settings for Windows Explorer will detect the presence of this newly created ADS.

Additionally, ADSs can be created and associated with the directory listing, rather than a file. This peculiarity will take on some significance later in this article, but for now it's sufficient to describe how such ADSs can be created.

```
c:\ads>echo This ADS is tied to the directory listing > :hidden
```

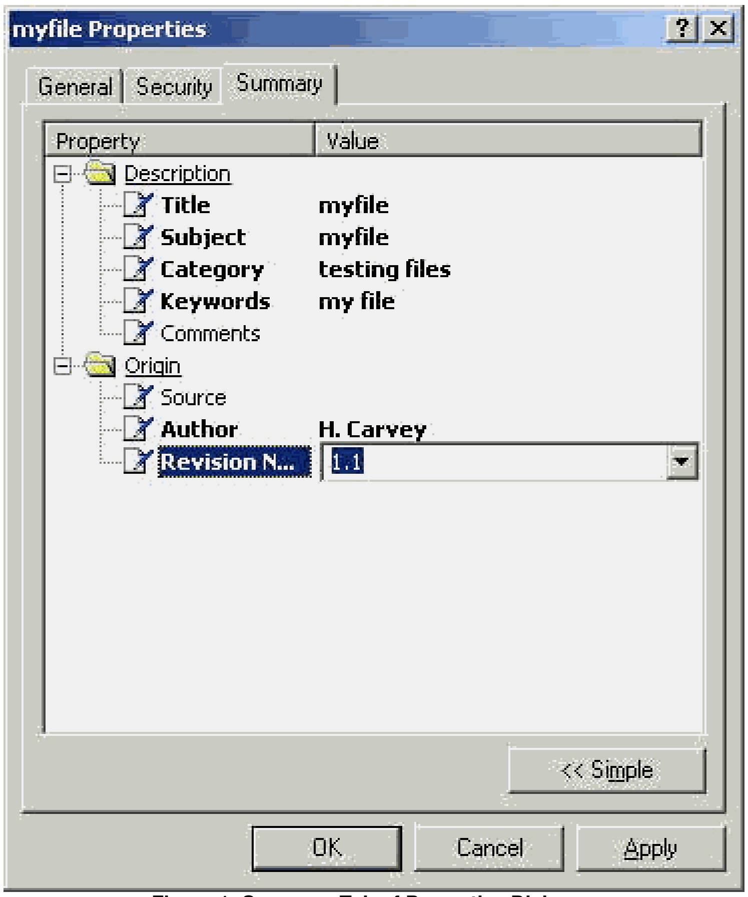
ADSs of this type can be created with Notepad and the 'type' command, as well.

The content of ADSs should not be considered limited to simply text data. Any stream of binary information can constitute a file, and the ADS is nothing more than a file. Executables can be hidden in ADSs quite easily.

```
c:\ads>type c:\winnt\notepad.exe > myfile.txt:np.exe

c:\ads>type c:\winnt\system32\sol.exe > myfile.txt:sol2.exe
```

Similarly, image files, audio files, or any other stream of data can be hidden in ADSs.

Finally, Windows Explorer provides a means by which very specific ADSs can be created (RUSS00). If the user opens Explorer and chooses a file, and then right-clicks on that file, a drop-down menu appears. Choosing 'Properties' will open a Properties dialogue, and choosing the Summary tab (see Fig. 1) will reveal fields in which the user can insert information.


myfile Properties

Property	Value
Description	
Title	myfile
Subject	myfile
Category	testing files
Keywords	my file
Comments	
Origin	
Source	
Author	H. Carvey
Revision N...	1.1

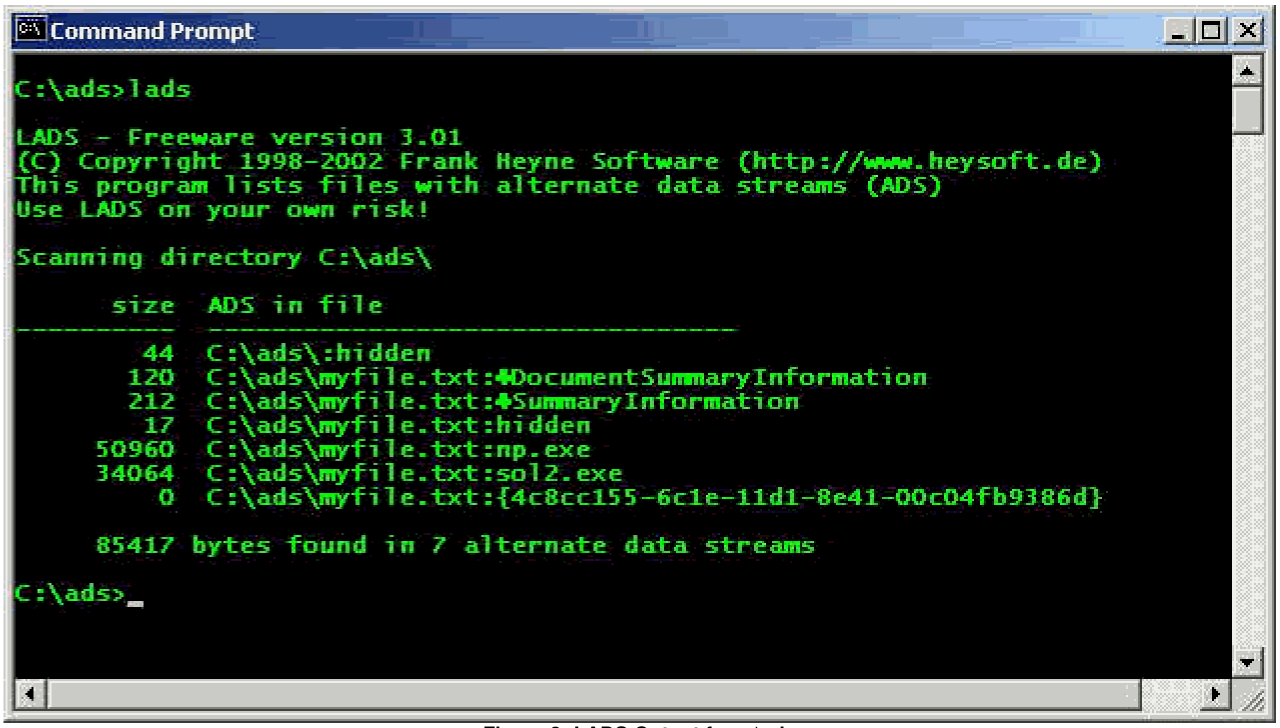
Figure 1: Summary Tab of Properties Dialogue

ADSs have no attributes of their own, per se. The access rights assigned to the default unnamed stream control access for creating or viewing ADSs. Quite simply, if a user cannot write to a file, that user cannot add an ADS to that file. Further, while Windows File Protection prevents the replacement of protected system files, it does not prevent a user with the appropriate permissions from adding ADSs to those system files. The System File Checker (sfc.exe) will verify that protected system files have not been overwritten, but will not detect ADSs.

Users and administrators should also be aware of KB article Q319300 , which states that the Windows 2000 Content Indexing Server adds alternate data streams named '?Q30lsldxJoudresxAssqpcawXc' to image files on NTFS volumes. These ADSs contain thumbnails of the images.

Detecting, Viewing, and Manipulating ADSs

As previously stated, Microsoft provides no tools or utilities either within the operating system software distribution or the Resource Kits for detecting the presence of ADSs. One of the best tools available for this is lads.exe, written by Frank Heyne . Lads.exe is currently available as version 3.01, and does an excellent job of reporting the availability of ADSs. For administrators used to working with graphical tools, lads.exe is a command line interface (CLI) tool that reports its findings to the screen (i.e., standard output or STDOUT). Figure 2 shows an example lads.exe output, run against the test directory, c:\ads.



```
C:\ads>lads

LADS - Freeware version 3.01
(C) Copyright 1998-2002 Frank Heyne Software (http://www.heysoft.de)
This program lists files with alternate data streams (ADS)
Use LADS on your own risk!

Scanning directory C:\ads\

      size  ADS in file
----------  ---------------------------------
        44  C:\ads\:hidden
       120  C:\ads\myfile.txt:♠DocumentSummaryInformation
       212  C:\ads\myfile.txt:♠SummaryInformation
        17  C:\ads\myfile.txt:hidden
     50960  C:\ads\myfile.txt:np.exe
     34064  C:\ads\myfile.txt:sol2.exe
         0  C:\ads\myfile.txt:{4c8cc155-6c1e-11d1-8e41-00c04fb9386d}

     85417 bytes found in 7 alternate data streams

C:\ads>_
```

Figure 2: LADS Output for c:\ads

Figure 2 shows just how useful lads.exe can be. Not only does the utility report the presence of ADSs, but it also reports the full path and size for each ADS. Particular note should be taken of the three of the ADSs associated with myfile.txt. Two begin with an ASCII character resembling the spade from a playing card, and the third is a long series of numbers and letters between two curly braces. These are the ADSs that were associated with the file using the Summary tab of the Properties dialogue (fig. 1).

Once an ADS is detected, what can be done to view its contents? Notepad is a very handy utility for viewing files, and can be used for viewing the contents of ADSs. However, there is a catch. For example, the following command produces unexpected results:

```
c:\ads>notepad myfile.txt:hidden
```

When this command is executed, Notepad opens and asks if the user wishes to create a new file. This is an unusual request, because the ADS was created earlier. In order to observe the expected results enter the following commands:

```
c:\ads>echo This is another ADS > myfile.txt:hidden.txt
```

```
c:\ads>notepad myfile.txt:hidden.txt
```

The same effects can be observed when the ADS is associated the directory listing, as in ':hidden.txt'. The addition of the extension on the end of the filename allows the ADS to be opened in Notepad. This will also work for other ADSs, such as:

```
c:\ads>notepad myfile.txt:np.exe
```

ADSs are a feature of the NTFS file system, so if a file with an ADS is moved to a disparate file system, such as FAT, FAT32, or ext2, the ADS is removed, as it is not supported on these other file systems. ADSs are preserved if the default unnamed stream (i.e., myfile.txt from the previous examples) is copied or moved across NTFS partitions, or even to a mapped NTFS drive. This can be accomplished using the ‘copy’ or ‘move’ commands, as appropriate.

Removing all ADSs from a default stream is relatively simple, using the following commands:

```
c:\ads>type myfile.txt > myfile.bat
```

```
c:\ads>del myfile.txt

c:\ads>ren myfile.bat myfile.txt
```

Using LADS, it is easy to verify that all ADS created in the above examples have vanished.

Executing ADSs

In previous examples, executables were hidden in ADSs. This information seems fairly useless unless the executables themselves can be launched, without the overhead of having to copy them out of the ADS first. In fact, the ‘start’ command can be used to do just that. Since the executables hidden earlier were deleted, rerunning the commands will serve the purpose of an example. Using the ‘type’ command, hide Notepad and Solitaire in ADSs associated with myfile.txt.

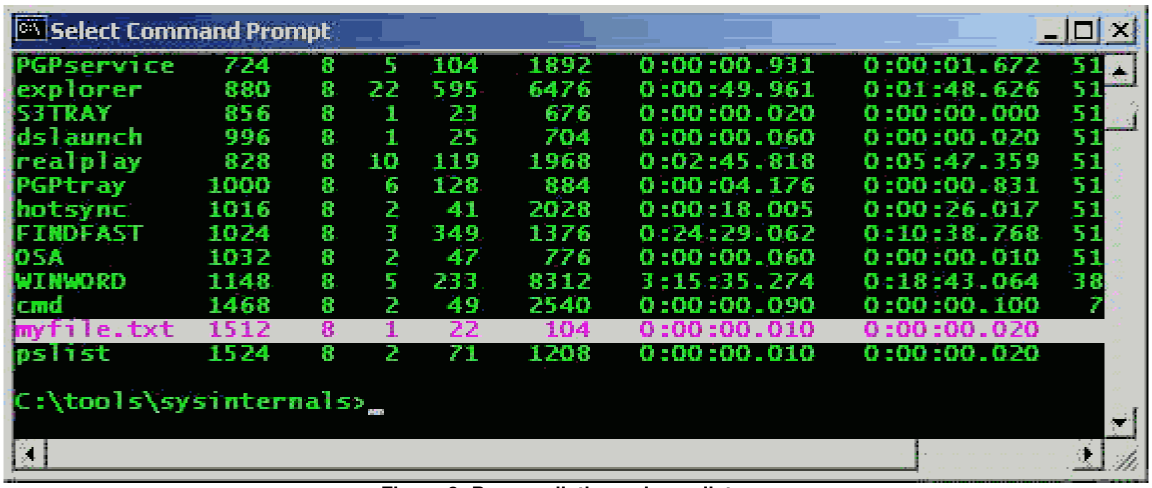
On NT, a simple command will launch either executable (MCCL99):

```
c:\ads>start myfile.txt:np.exe

c:\ads>start myfile.txt:sol2.exe
```

However, these commands generate an error on 2K. From the error message, it appears as if the information pointing to the executable wasn’t sufficient. Therefore, either absolute or relative paths should suffice, and running either of the following commands will demonstrate this:

```
c:\ads>start c:\ads\myfile.txt:np.exe

c:\ads>start .\myfile.txt:np.exe
```

An interesting item to note is how the process appears while running. For example, running pslist.exe from SysInternals after executing either of the above two commands displays a process called 'myfile.txt' running with a PID of 1512, as shown in figure 3.



```
PGPservice    724   8   5  104   1892   0:00:00.931   0:00:01.672   51
explorer      880   8  22  595   6476   0:00:49.961   0:01:48.626   51
S3TRAY        856   8   1   23    676   0:00:00.020   0:00:00.000   51
dslaunch      996   8   1   25    704   0:00:00.060   0:00:00.020   51
realplay      828   8  10  119   1968   0:02:45.818   0:05:47.359   51
PGPtray      1000   8   6  128    884   0:00:04.176   0:00:00.831   51
hotsync      1016   8   2   41   2028   0:00:18.005   0:00:26.017   51
FINDFAST     1024   8   3  349   1376   0:24:29.062   0:10:38.768   51
OSA          1032   8   2   47    776   0:00:00.060   0:00:00.010   51
WINWORD      1148   8   5  233   8312   3:15:35.274   0:18:43.064   38
cmd          1468   8   2   49   2540   0:00:00.090   0:00:00.100    7
myfile.txt   1512   8   1   22    104   0:00:00.010   0:00:00.020
pslist       1524   8   2   71   1208   0:00:00.010   0:00:00.020

C:\tools\sysinternals>_
```

Figure 3: Process listing using pslist.exe

Figure 4 shows the process running in the Process tab of the TaskManager.

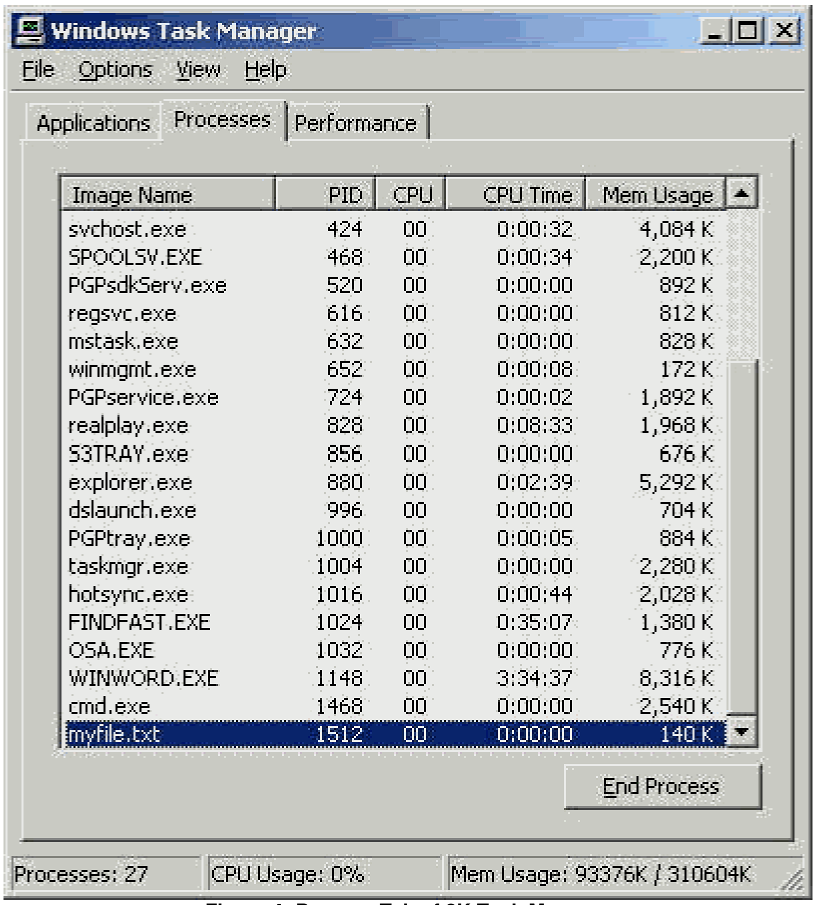
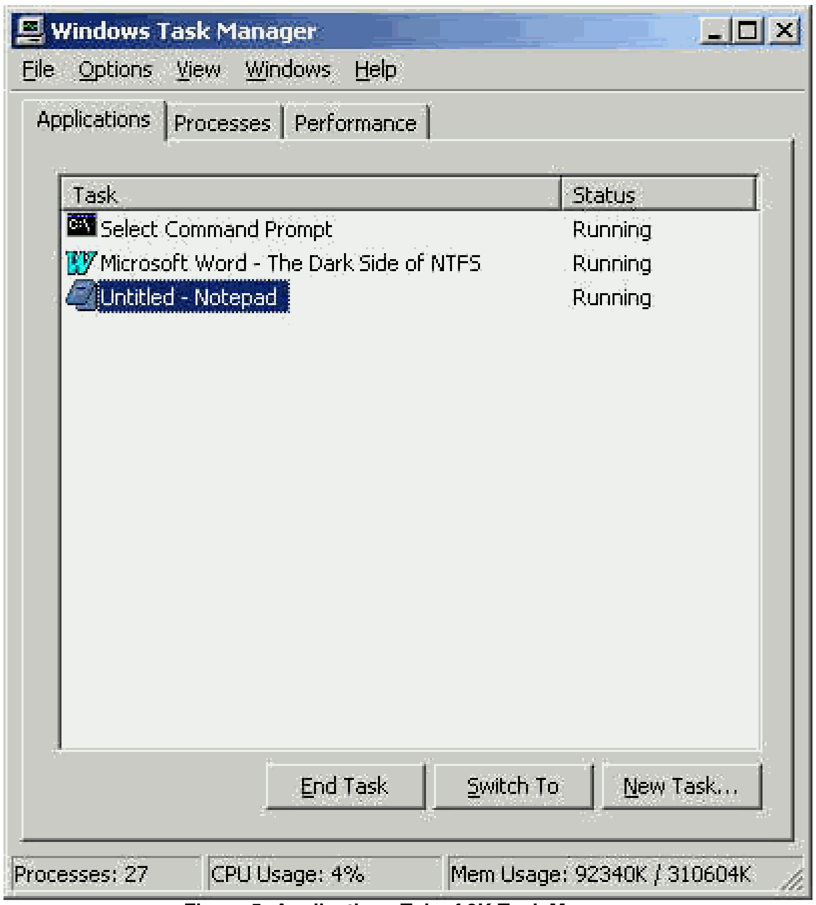


Image Name	PID	CPU	CPU Time	Mem Usage
svchost.exe	424	00	0:00:32	4,084 K
SPOOLSV.EXE	468	00	0:00:34	2,200 K
PGPsdkServ.exe	520	00	0:00:00	892 K
regsvc.exe	616	00	0:00:00	812 K
mstask.exe	632	00	0:00:00	828 K
winmgmt.exe	652	00	0:00:08	172 K
PGPservice.exe	724	00	0:00:02	1,892 K
realplay.exe	828	00	0:08:33	1,968 K
S3TRAY.exe	856	00	0:00:00	676 K
explorer.exe	880	00	0:02:39	5,292 K
dslaunch.exe	996	00	0:00:00	704 K
PGPtray.exe	1000	00	0:00:05	884 K
taskmgr.exe	1004	00	0:00:00	2,280 K
hotsync.exe	1016	00	0:00:44	2,028 K
FINDFAST.EXE	1024	00	0:35:07	1,380 K
OSA.EXE	1032	00	0:00:00	776 K
WINWORD.EXE	1148	00	3:34:37	8,316 K
cmd.exe	1468	00	0:00:00	2,540 K
myfile.txt	1512	00	0:00:00	140 K

Processes: 27 CPU Usage: 0% Mem Usage: 93376K / 310604K

Figure 4: Process Tab of 2K Task Manager

Oddly enough, the Process tab on 2K shows that PID 1512 has an Image Name of 'myfile.txt'. Figure 5 shows the Application tab of the Task Manager.



Task	Status
Select Command Prompt	Running
Microsoft Word - The Dark Side of NTFS	Running
Untitled - Notepad	Running

Processes: 27 CPU Usage: 4% Mem Usage: 92340K / 310604K

Figure 5: Applications Tab of 2K Task Manager

Figure 6 shows that the Process tab of the Task Manager on XP displays when the same command is executed on that operating system.

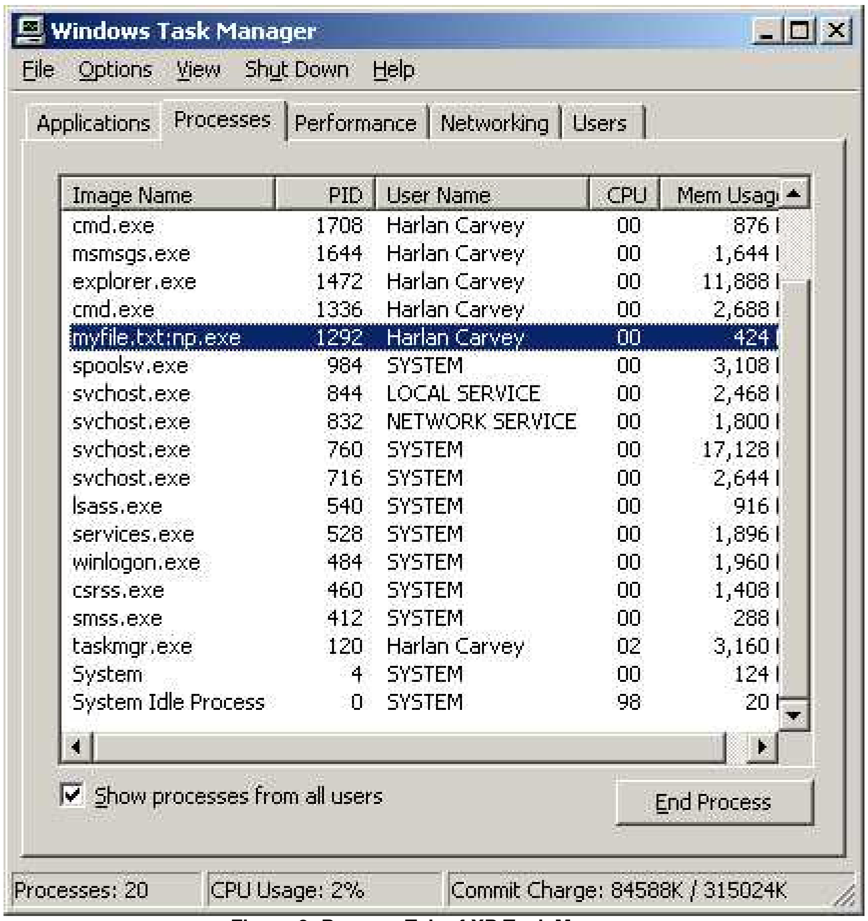
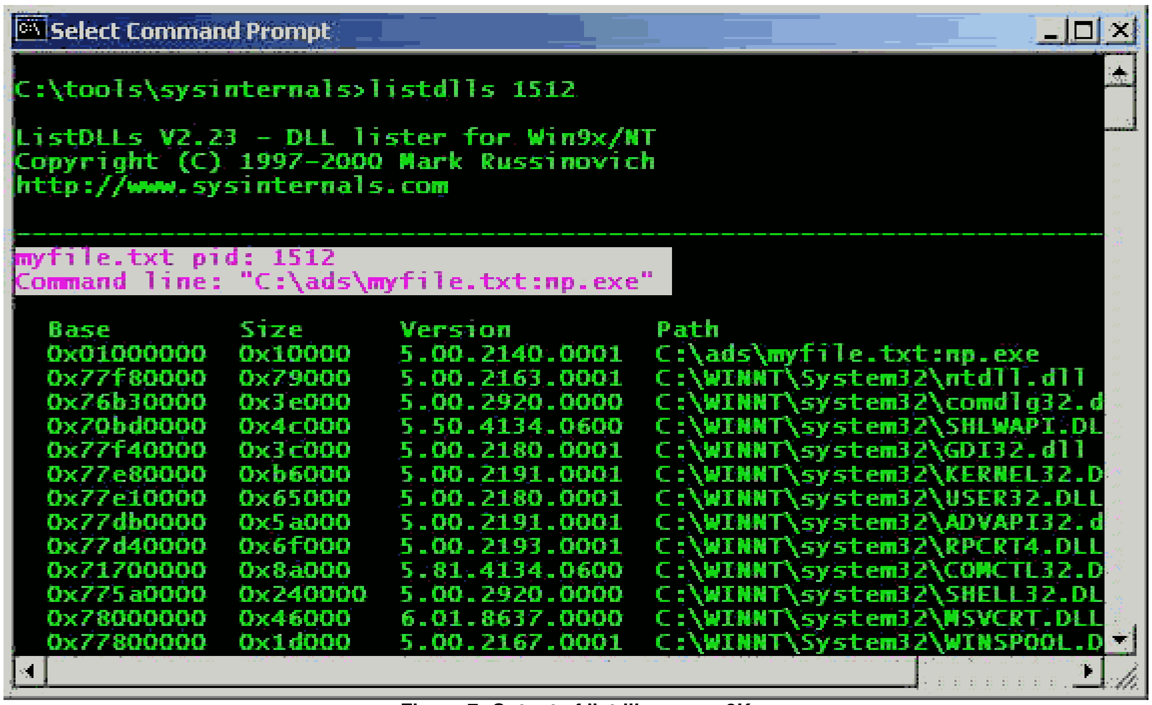

Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
Show processes from all users
End Process

Image Name	PID	User Name	CPU	Mem Usage
cmd.exe	1708	Harlan Carvey	00	876
msmsgs.exe	1644	Harlan Carvey	00	1,644
explorer.exe	1472	Harlan Carvey	00	11,888
cmd.exe	1336	Harlan Carvey	00	2,688
myfile.txt:np.exe	1292	Harlan Carvey	00	424
spoolsv.exe	984	SYSTEM	00	3,108
svchost.exe	844	LOCAL SERVICE	00	2,468
svchost.exe	832	NETWORK SERVICE	00	1,800
svchost.exe	760	SYSTEM	00	17,128
svchost.exe	716	SYSTEM	00	2,644
lsass.exe	540	SYSTEM	00	916
services.exe	528	SYSTEM	00	1,896
winlogon.exe	484	SYSTEM	00	1,960
csrss.exe	460	SYSTEM	00	1,408
smss.exe	412	SYSTEM	00	288
taskmgr.exe	120	Harlan Carvey	02	3,160
System	4	SYSTEM	00	124
System Idle Process	0	SYSTEM	98	20

Processes: 20 CPU Usage: 2% Commit Charge: 84588K / 315024K

Figure 6: Process Tab of XP Task Manager

Finally, obtaining information about the process with listdlls.exe from SysInternals will display 'c:\ads\myfile.txt:np.exe' as the command line for the above command (see fig. 7), on both 2K and XP.



```
C:\tools\sysinternals>listdlls 1512

ListDLLs V2.23 - DLL lister for Win9x/NT
Copyright (C) 1997-2000 Mark Russinovich
http://www.sysinternals.com

------------------------------------------------------------------------------
myfile.txt pid: 1512
Command line: "C:\ads\myfile.txt:np.exe"

  Base        Size      Version         Path
  0x01000000  0x10000   5.00.2140.0001  C:\ads\myfile.txt:np.exe
  0x77f80000  0x79000   5.00.2163.0001  C:\WINNT\System32\ntdll.dll
  0x76b30000  0x3e000   5.00.2920.0000  C:\WINNT\system32\comdlg32.d
  0x70bd0000  0x4c000   5.50.4134.0600  C:\WINNT\system32\SHLWAPI.DL
  0x77f40000  0x3c000   5.00.2180.0001  C:\WINNT\system32\GDI32.dll
  0x77e80000  0xb6000   5.00.2191.0001  C:\WINNT\system32\KERNEL32.D
  0x77e10000  0x65000   5.00.2180.0001  C:\WINNT\system32\USER32.DLL
  0x77db0000  0x5a000   5.00.2191.0001  C:\WINNT\system32\ADVAPI32.d
  0x77d40000  0x6f000   5.00.2193.0001  C:\WINNT\system32\RPCRT4.DLL
  0x71700000  0x8a000   5.81.4134.0600  C:\WINNT\system32\COMCTL32.D
  0x775a0000  0x240000  5.00.2920.0000  C:\WINNT\system32\SHELL32.DL
  0x78000000  0x46000   6.01.8637.0000  C:\WINNT\system32\MSVCRT.DLL
  0x77800000  0x1d000   5.00.2167.0001  C:\WINNT\System32\WINSPOOL.D
```

Figure 7: Output of listdlls.exe on 2K

An alternative method for launching hidden executables on 2K is a shortcut (KASP01). To demonstrate the point, create a shortcut on the desktop. The location of the item for this shortcut should be 'c:\ads\myfile.txt'. Once the shortcut has been created, observe the icon on the desktop. Assuming the ADS for Solitaire was created, edit the Properties of the shortcut so that the target now refers to 'c:\ads\myfile.txt:sol2.exe'. Wait a few seconds and observe any changes to the icon. Launch the executable by double-clicking the icon.

Interestingly enough, as similar technique works by adding an entry to the Windows Startup Folder (KASP01) or to the ‘Run’ key in the Registry (KASP01). The full path to the key is:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
```

Specifying the complete path to the hidden executable will guarantee that it is launched the next time the system is started.

Yet another method for launching executables hidden in ADSs is via the ‘Run’ box in the Start menu (KASP01). Clicking on the Start button, then 'Run', and typing the following command will launch the Solitaire ADS:

```
file:///c:/ads/myfile.txt:sol2.exe
```

For administrators using Perl , the Perl interpreter handles ADSs quite easily. For example, the following lines of Perl code make use of backticks to launch an ADS:

```
my $file = 'c:\ads\myfile.txt:sol2.exe';

`$file`;
```

Save the above code as ‘ads.pl’, and execute the code by typing:

```
c:\perl>ads.pl
```

As an interesting variation, the following code also works:

```
c:\perl>type ads.pl > myfile.txt:ads.pl

c:\perl>perl myfile.txt:ads.pl
```

The Windows Scripting Host (WSH) began shipping with 2K and presents some interesting possibilities with regards to ADSs (KASP01). This is particularly important because WSH is native to the 2K and XP distributions, while Perl must be installed separately. To demonstrate the capabilities of WSH, type the following command:

```
c:\ads>echo MsgBox 'VBS file to test ADSs' > ads.vbs
```

To execute the script using WSH, type:

```
c:\ads>wscript ads.vbs
```

Alternatively, type:

```
c:\ads>cscript ads.vbs
```

Or, simply double-click the file in Windows Explorer. All three of these methods will run the VBScript code. Now type the following commands:

```
c:\ads>type ads.vbs > myfile.txt:ads.vbs

c:\ads>wscript myfile.txt:ads.vbs
```

Or:

```
c:\ads>start .\myfile.txt:ads.vbs
```

Cscript.exe also runs the script. Alternatively, the shortcut and Run box methods mentioned above also run the script (KASP01).

The Really, Really Dark Side

If the script hidden in an ADS ends with a different extension (i.e., other than '.vbs'), WSH has trouble recognizing the type of file, and will complain that an engine for executing the file cannot be found. For example:

```
c:\ads>type ads.vbs > myfile.txt:ads.txt

c:\ads>wscript myfile.txt:ads.txt
```

Executing the second command above results in the error described. However, both wscript.exe and cscript.exe have switches that allow the administrator to control the execution of the program. The command syntax can be seen by typing:

```
c:\ads>wscript /?
```

The '//E' switch allows the user to specify an engine to be used. The new command looks like:

```
c:\ads>wscript //E:vbs myfile.txt:ads.txt
```

This provides some interesting opportunities for malicious use. For example, the earlier example of ADSs produced via Windows Explorer (i.e., the Summary tab of the Properties dialogue for a file) produced on ADS with the name '♣SummaryInformation'. Code, such as VBScript, can be created and written to an ADS with the same name, and then launched. The following Perl code illustrates this:

```
my $file = 'c:\\ads\\myfile.txt:'.chr(5).'SummaryInformation';

my $src = 'c:\ads\ads.vbs';

`type $src > $file`;

`wscript //E:vbs $file`;
```

The final command in the script is the one of interest. For both wscript.exe and cscript.exe, the '//E' switch forces the application to use a particular script engine. In this case, the ADS containing the script to be launched has no file extension, so the scripting host has no means for determining the scripting engine to be used. This example could also apply quite easily to ADSs created by the Content Indexing Server, as mentioned above.

Conclusion

ADSs are a feature of the NTFS file system intended to provide compatibility with HFS, which may still be necessary for compatibility. However, the lack of visibility of this 'feature' poses a significant risk for administrators. There has already been one virus released that employed ADSs, W2K.Stream written by Bennie and Ratter of the group 29A (KASP01). As the release of malware and incidents of cybercrime increase, the malicious use of ADSs will likely increase as well.

The solution is not to stop using the NTFS file system, as the benefits in security and reliability are too great. This 'feature' has remained part of the file system since NT 3.1. Given the circumstances, a far more prudent solution would have been to include support for HFS files in the File and Print Services for the Macintosh, rather than the file system. As it is, administrators should make judicious use of discretionary access control lists (DACLs) on files and directories (CARV00), and regularly scan their critical systems using utilities such as lads.exe. In addition, Microsoft should be lobbied to add the ability to detect and view ADSs to Windows Explorer and the command interpreter. A more than appropriate measure would be to have ADSs appear in Windows Explorer by default, using an icon with a scarlet 'A' to signify an ADS. Additionally, Microsoft should provide restrictions within the operating system for creating processes from executable files whose names contain a colon.

Further, antivirus software vendors should include support for ADSs within their products by default. While many of the worms seen over the past year or more have been executables written in Visual Basic or Delphi, others have been Visual Basic scripts. This malware has been capable of wreaking considerable havoc, and all prudent steps should be taken to protect systems.

References

RUSS00 Russinovich, M., *Inside Win2K NTFS, Part 2* , Windows 2000 Magazine, November, 2000

MCCL99 McClure, S., Scambray, J., and Kurtz, G., *Hacking Exposed: Network Security Secrets and Solutions*, Berkeley: Osbourne, 1999

KASP01 Kaspersky, E. and Zenkin, D., *NTFS Alternate Data Streams* , Windows and .Net Magazine, Spring 2001

CARV00 Carvey, H., *Network Trojans: What You REALLY Need To Know* , Information Security Bulletin, Vol. 5, Issue 8