

# NTFS STREAMS FAQ

## Hidden NTFS Alternate Data Streams (ADS) Explained Mark E. Donaldson

### What are hidden Alternate Data Streams (ADS)?

NTFS, the filesystem used by Windows NT, Windows 2000 and Windows XP has a feature that is not well documented and is unknown to many developers and most users. This feature - Alternate Data Streams - allows data to be stored in hidden files that are linked to a normal visible file. Streams are not limited in size and there can be more than one stream linked to a normal file.

### Why is ADS a security risk?

The primary reason why ADS is a security risk is because streams are almost completely hidden and represent possibly the closest thing to a perfect hiding spot on a file system - something Trojans can and will take advantage of. Streams can easily be created/written to/read from, allowing any Trojan or virus author to take advantage of a hidden file area. But while streams can easily be used, they can only be detected with specialist software. Programs such as Explorer can view normal parent files, but they can't see streams linked to such files, nor can they determine how much disk space streams are using. Because ADS is virtually unknown to many developers, there are very few security programs available that are ADS-aware. As such, if a virus implants itself into an ADS stream, your anti-virus software will probably not be able to detect it. In addition, streams cannot be deleted - to delete a stream you must delete its parent. Streams are of particular importance to law enforcement agencies as important data can sometimes be hidden in these covert file system channels.

### What Microsoft operating systems support streams?

Windows NT 3.1, 3.5, 3.51, 4.0, Windows 2000 and Windows XP. Streams are only supported on NTFS, not FAT/FAT32. We have released a free NTFS ADS Check utility (9kb) so you can check your system for NTFS ADS support.

### Why does NTFS support streams?

The main (but not only) reason is for Macintosh file support. Files stored on the Macintosh file system consists of two parts (known as forks) - one data fork, and one resource fork. Windows relies on the extension of the file (eg. ".exe") in order to determine which program should be associated with that file. Macintosh files use the resource fork to do this. NT stores Macintosh resource forks in a hidden NTFS stream, with the data fork becoming the main parent file to the stream. ADS has other uses. As just one example, you could store a thumbnail image of a picture in a stream and even an audio track, allowing a single file to have several multimedia components. Some anti-virus programs store checksums in a stream under every file on your disk.

### What are the main dangers associated with NTFS streams?

- Streams are only visible to specialized software such as TDS-3 that has the capability of enumerating streams from their parents.
- Public awareness of streams is exceptionally low, especially compared to the awareness of other file-hiding techniques such as hidden file attributes.
- Streams can not only attach themselves to files, they can also attach themselves to directories.
- Streams can't actually be deleted. The parent they're attached to must be deleted in order for the stream to be removed. However, Streams attached to the root directory of a drive, such as "C::MyStream" cannot be deleted.
- "Available Disk Space" as shown by programs such as Windows Explorer do not take into account disk space consumed by streams.

# NTFS STREAMS FAQ

## Hidden NTFS Alternate Data Streams (ADS) Explained Mark E. Donaldson

- A malicious program could continue writing to a stream, filling up the disk and make cleaning up very difficult.
- Streams, as they are essentially still files, can be executed.
- Executed streams do not have their filenames display correctly in Windows NT/2K/XP Task Manager, the utility commonly used to view running processes. For example, if the stream "c:\test.txt:mystream" was running, Task Manager would only show "test.txt".

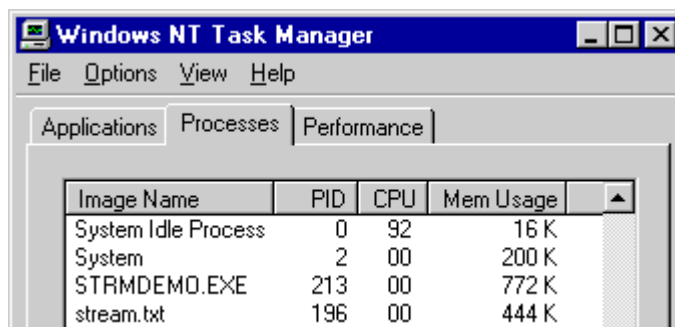
### Are there any anti-trojan systems available that can detect NTFS ADS streams?

At the time of this writing (July 2001), only one. TDS-3, the worlds most comprehensive anti-trojan system since 1997 has the capability of detecting and scanning inside all NTFS Alternate Data Streams. TDS-3 will alert you to the presence of all streams even before it starts scanning inside them for malicious code/trojans:

Alarm	Name	File
Executable found (in hidden data stream)	ADS Hidden Stream Detected: 386 bytes	c:\streams\parent.txt:streamfile.txt
Positive identification (in hidden stream)	RAT.Netbus 1.70: 225280 bytes	c:\ntfs_ads.txt:netbus.exe
NTFS Alternate Data Stream	ADS Hidden Stream Detected: 0 bytes	c:\config.sys:\$security

### Demonstration

This program (13kb) is a safe and simple demonstration of NTFS Alternate Data Streams. When run it creates a small text file in the root path of the drive it was run from (usually C). This text file (c:\streams.txt if run from drive C) simply contains a small message that "This file was created by DiamondCS Stream Demo". A 3kb executable program is then created in "c:\streams.txt:TheStream", and executed. The 3kb program simply displays a messagebox on-screen to let you know that it is running. When you see the messagebox, have a look in your process list (Task Manager), and instead of



seeing "streams.txt:TheStream" you will only see "streams.txt":

To "clean up" afterwards, simply delete c:\streams.txt - this will delete both the parent file and the stream which has the executable program embedded under it.

### Anti-Virus/Anti-Trojan Test - Does your scanner pass the test?

We've created a small 10kb utility called MakeStream that moves data from a commandline-specified file into a hidden Alternate Data Stream attached to the original. For example, if you ran makestrm.exe c:\test.exe, the file contents of c:\test.exe would be moved into c:\test.exe:StreamTest (an Alternate Data Stream), and the original file contents are then over-written with a simple message reminding you about the linked stream.

# NTFS STREAMS FAQ

## Hidden NTFS Alternate Data Streams (ADS) Explained Mark E. Donaldson

Anti-Virus/Anti-Trojan Test - This is a safe test and involves no file execution other than running makestrm.exe, which simply performs file I/O operations. Get any Trojan or virus that is detected by your virus/trojan scanner, and run makestrm.exe on it to move its file contents into a hidden stream. Then, re-scan the file - is it still detected? Unless you're using TDS-3, the answer is almost certainly No.

### **How can I create a stream?**

Creating, writing to, and reading from streams is virtually the same as with a normal file, except that stream filenames consist of two parts separated by a colon. The first part points to the parent file of the stream. The second part is the actual stream name itself - this can be anything as long as it complies with NTFS naming conventions. Microsoft Notepad is an ADS-aware application, and as such you can use it to create your own streams: From Windows, click on Start | Run, and then type notepad c:\myfile.txt:MyStream. This will create (or read from, if it already exists) the stream "MyStream" which is linked to the parent file c:\myfile.txt. Using any file browser such as Explorer, you will see that the c:\myfile.txt file is 0 bytes, and the "c:\myfile.txt:MyStream" file is nowhere to be seen. This is because "MyStream" is not a true file - it is a data stream that is linked to the parent file of c:\myfile.txt. If you modify the parent file, any streams linked to it will not be affected. Likewise, modifying a stream does not affect it's parent. "MyStream" is hidden storage.

### **If I double-click on a file that has an executable stream attached to it, will the stream execute?**

No. The stream can only be executed if called directly - typically a program can only do this. You'll never accidentally execute a stream.

### **Notepad can read and write to and from streams, why can't most programs?**

Programs can read/write to/from streams, but only if they do it directly. If the program uses a directory lookup function to test if the file exists, it will fail, as stream files do not show in directory/file lookups, so you'll normally see an error message along the lines of "File not found". Notepad is able to open the file because it determines file existence by attempting to open the file directly rather than checking via a directory lookup.

### **Can streams be transferred via a Windows Local Area Network (LAN)?**

Yes, but only if the target is an NTFS drive. Copying-and-pasting a stream-hosting file using Windows Explorer can easily do this simply

### **Can streams be transferred via email or FTP or other file-transfer protocols?**

No. Stream contents can be transferred via email, FTP etc., but only in the form of a normal file. If you attempt to email a file with any streams attached, only the actual file will be sent - not any of the streams. Virtually all Internet protocols that support file transfer are non-supportive of streams and other embedded file resources, as streams are relatively exclusive to Macintosh and Windows NT/2K systems using the NTFS file system.

### **Can streams be exploited remotely?**

In certain circumstances (and it has happened before), but almost always due to a vulnerability exploit. As a real example, several web servers (including older versions of Microsoft IIS) are susceptible to having their file source read via the :\$DATA stream. Normally when you go to a server-side script (PHP, ASP etc), you'll go to a URL like this: <http://www.victim.com/target.asp>. Going to target.asp would cause www.victim.com to process target.asp and return the processed data to the requesting user. Hackers found that the data stream could be read, allowing the hacker to view the source code of the ASP file rather than the processed data it returned by going to a URL like this: [http://www.victim.com/target.asp::\\$DATA](http://www.victim.com/target.asp::$DATA). This is a critical vulnerability, particularly when sensitive details are often stored in server-side source code.