Global Information Assurance Certification Paper

Alternate Data Streams for Managers

Introduction

Ok, Ok, don't get too spun up over the title. Just a poke at managers, like me, who have wandered away from day to day tech stuff but still want to keep their hand in the game a little. When this topic came up over lunch one day, I had the deer-in-the-headlights look. So I decided to do something about it. Read a ton of tech articles, read through some code, tested some software and want to share some of my enlightenment. I encourage you to browse through some of the references at the end of this paper. Many of them were very informative reads and had some solid technical background behind them.

So what, you say. Who cares about his thing called ADS or whatever? Well…. Maybe you, when your system becomes glacially slow. Maybe you, when your administrator informs you that you have exceeded your storage limits on the machine and you haven't been storing anything other than a few text files. Maybe you, when your machine is acting like a virus hit it and nothing, no admin tools, no virus checkers, no disk de-fragmentation or anything else helps. So read on McFly!

How did this all get started?

It turns out that there has been an on-going controversy over Alternate Data Streams that has been raging on since the vulnerability aspect was originally introduced around late 1997. One of the first posts on NTBUGTRAQ[1] was on 3/20/1998. So, you can see this is not a new topic. However, it is still a hot topic and if you spend some time searching the Internet for articles, you'll find that there are still many new posts regarding the vulnerability and exploitability of this "feature" of Windows NTFS. I will spend some time reviewing this vulnerability (or is it feature?) of the NTFS based operating systems. We will go over the basics of the technology and then delve into the avenues of exploiting Alternate Data Streams for potentially devious purposes. Should we worry or not worry? Even Anti-Virus companies don't seem to agree. This paper will present issues at the "30,000 foot" viewpoint addressing as many relevant discussion areas as possible. Detailed sections will have voluminous references for anyone wanting to dig into the inner workings of ADS. So, that being said, let's start at the beginning.

It appears when Microsoft designed and built early Windows they were, at least, aware of Apple Computer and of the Macintosh PCs. New functionality was added in the release of Windows 3.1 to provide support for Macintosh file storage. Briefly, the Mac file storage structure has two basic parts, the resource *fork* and the data *fork*. Data for the file is in the data fork and the resource fork determines how the data is to be handled. Windows does this same thing through file associations. The file extension (.doc, .exe, .html, .ppt, etc.) is normally "associated" with a particular application that will know how to handle the data. In order to accomplish this Mac compatibility, Microsoft implemented Alternate Data Streams. The hidden stream is used for the

resource fork and the main stream is used to store the data fork. However, this isn't the only reason for ADS.

Every program needs to store data, but data can be stored in many different ways to optimize how each application can utilize it. Applications need to save their state from one session to the next session. Things like user settings and program configuration information need to be retained to eliminate the need to have the user re-enter all this information each and every time they invoke the program. Word processors, graphics editors, sound files, database records, etc. all have different file formats and different storage requirements. Something other than the standard flat DOS file structure was needed. Some flexibility had to be added to handle these differing data storage needs. So, Microsoft built in the data stream concept. This diversion from the original DOS file structure paved the way for the Object Linking and Embedding (OLE) structured storage system. It is interesting to note that ADS, in its present implementation, may not be with us for long. According to Microsoft's HOWTO bulletin in Dec,2000[2]

"*Alternate data streams are strictly a feature of the NTFS file system and may not be supported in future file systems. However, NTFS will be supported in future versions of Windows NT. Future file systems will support a model based on OLE 2.0 structured storage (IStream and IStorage). By using OLE 2.0, an application can support multiple streams on any file system and all supported operating systems (Windows, Macintosh, Windows NT, and Win32s), not just Windows NT.*"

Without getting deeper into OLE 2.0, and understanding how it's utilized by the various OSs, it is uncertain to me whether OLE 2.0 presents new avenues for malicious code or not. This is an area deserving of its own research and paper.

Thumbnail sketch

Since NTFS is at the heart of this discussion lets take a look there. Information about a file in NTFS is located in a data construct called the Master File Table (MFT). The MFT contains all the information that describes the files and the directories. Each file entry has "attributes" that describe the file. Below is a table of just a few of the many attributes and their description.

Sampling of some Attributes of the NTFS Master File Table (MFT)

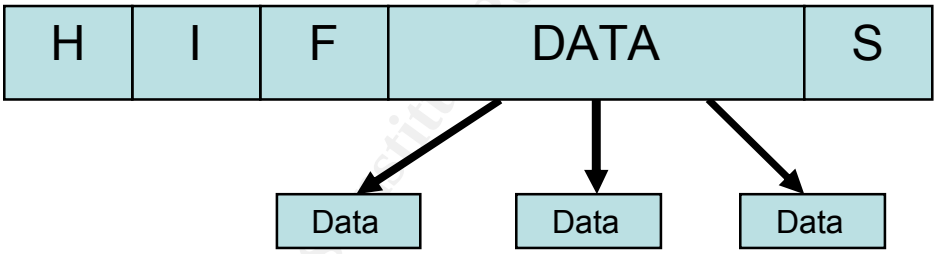
Attribute (Field)	**Description**
Standard Attributes (Standard Information)	Contains standard file attributes, such as file creation time, archive status, data links, etc.
Filename	Contains the Unicode file name and the DOS 8.3 file name
Security Descriptor	Contains information on ownership, access rights, and other security-related information.
Data	Contains file data for files up to about 1.5K long. **Otherwise, a pointer to the data.** The default data is an unnamed stream. NTFS offers a mechanism for additional named streams.

The DATA attribute is the interesting one for this discussion. Below are two simple diagrams showing the different ways that the DATA attribute can be used. Data can be stored directly in the MFT if the volume of data is small enough (less than about 1500 bytes). If the volume of data is larger, the DATA attribute is used as a pointer. This is the interesting part. As mentioned above, the DATA attribute can point to multiple pieces of data. It can point to multiple files. It can point to the main file or *stream,* as well as additional files or *Alternate Data Streams* (ADS). The point to highlight here is that the DATA attribute is how and where alternate data streams are made possible.

Small Files residing in the MFT

H = Header
I = Standard Information
F = File name
DATA = Data in residing in MFT
S = Security Descriptor

Larger Files not residing in the MFT



H = Header
I = Standard Information
F = File name
DATA = Pointers to Data not residing in MFT
S = Security Descriptor

Alternate Data Streams are not visible to file viewing functions like DIR or Windows Explorer. They can't be directly accessed through Notepad or MS Word. So they are more or less hidden and attached to the main stream through the DATA attribute pointers. The hidden part is what stirs up the controversy. If these were designed as legitimate data storage structures, why are they hidden? I was unable to find out any information, or even theories, as to why.

Examples of stream generation

To create an alternate stream is very simple. Windows *notepad, echo,* and *type* are ADS compliant commands (when run from the cmd window) and are the most commonly referenced tools in all of my reading material to do this study. Notepad is capable of reading an ADS and can create and write an ADS. Start by using Notepad as you usually would through the Windows toolbars. Create a text file called "*main.txt*". Save it in a file folder where you can easily get to it through the old DOS directory (CD) commands to save yourself some typing (I used C:\ for this example).[*]

Exit Notepad and go to the 'cmd' window by entering,

Start>run>"cmd"

Once in the command window, do a DIR command to see your *main.txt* file. Note the size of the file. You can now directly create the first ADS by using the 'echo' command

C:\ echo "Hidden data stream" > main.txt:sneaky.txt

Done!
You have just attached "*sneaky.txt*" to the end of *main.txt*. Once again, do a DIR command to see your main.txt file. The file size has not changed. Now type in the command string

C:\ notepad main.txt:sneaky.txt

VOILA! You now have the contents of the hidden file on your notepad screen.
You can also use notepad from the command line to create an ADS directly. Go to the command window again and make a new main stream.

C:\ notepad good.txt

Answer YES to create the file. Type in a few words, 'Save' it and exit the Notepad window.
Do a DIR to see your good.txt file. Now create an ADS

C:\ notepad good.txt:bad.txt

Answer *YES* to create the file and….

[*] (Note: I did all of this testing through virtual machines in VMWare so that anything I did could be easily wiped out without affecting the rest of my host PC. Be careful where you attach any ADS you wish to experiment with.)

You've done it again! Check it out by running

C:\ notepad good.txt:bad.txt

If later you edit *sneaky.txt* through notepad.exe via the point and click method, don't use "Save as". Even though notepad.exe can read and write alternate streams directly through the command line, it can not browse through the NTFS file system for an ADS. It has a file name checker that won't allow *:sneaky.txt* or *:bad.txt* to pass**.** Because of this, if you run notepad via the point-and-click method on the Windows desktop, it will not work.

Malicious? What's wrong with ADSs?

There are several things that invite trouble with ADSs. Let's discuss each area.

- They can attach themselves to files AND directories
- Detecting streams can be difficult
- Deleting streams can difficult.
- Available disk space calculated by DIR and Explorer doesn't consider space used by any ADS.
- Executable files can be stored in an ADS
- Anti-virus programs have difficulty in detecting ADSs
- File wiping utilities have problems eliminating ADS streams.
- They can be used as a Covert data channel
- Disk de-fragmentation and file integrity software won't clean-up or detect hidden streams

They can attach themselves to files AND directories

As shown above you can attach hidden streams to file but one really annoying thing you can also do is to attach an ADS to a directory of a drive. Including the root directory!!

The same methods used for files are used for directories so,

C:\ echo "This is now stuck to you root directory" > :annoying.txt

has now attached *annoying.txt* to you main c:\ directory name. As we'll discuss later, this can become more than just an annoyance.

Detecting streams can be difficult

It is, but it is also becoming a bit easier with newer ADS checking software. Incredibly, the Windows operating system doesn't give you any tools to find them but several aftermarket vendors are supplying tools, even freeware, that can detect an Alternate Data Stream. So, what is

available to detect these things? What do I look for if I don't know what file they are attached to? Well, a sampling of tools I found available on the Internet is shown below.

LADS[3]

This is one of the earliest freeware tools that I found in my searching. First released in 1998. I was able to run LADS /s (scan through subdirectories) with a scan from the root directory C:\ and it found all the test files I had created. It runs well in a subdirectory or folder if Lads.exe is resident in that same folder. However when I tried to scan a particular named subdirectory on my XP system outside of where Lads.exe resides, it returned "could not find directory c:\xxxx\xxxx\..." whatever the complete path name was. That being said, LADS is fairly quick and does give a summary of total number of alternate data streams found and the total of all bytes found in those files. It's free and works well on a full system scan, finding all the streams I created, including ADS attached to directories.

I decided to check how well the ADS checkers reacted to encryption and compression. Even when a file was encrypted, the LADS tool found them - sometimes. It appears that LADS detection tool can not be fooled by attaching an encrypted file to a main stream. However, another experiment I tried was to create an ADS, and then encrypt it with PGP. It turns out that the LADS detection program did not find the ADS stream. Also, if you attach a plain text file to an existing encrypted file, you can not detect it either. Interestingly, notepad could! I could attach a plain text file to the end of an encrypted file and with notepad.exe, read and write to the file. I also tried to find an ADS in which I had compressed the file pair (*main:alternate*) and LADS was unable to find it.

Encryption Detection Summary:

plain_text.doc:plain_text.doc - found
plain_text.doc:encrypted.doc - found
encrypted.doc:plain_txt.doc - not found
encrypted (plain_text.doc:plain_txt.doc) - not found
compressed – not found

CrucialADS[4]

I set up the same test files as done for LADS. Crucial is a Windows based freeware utility. It also doesn't allow for scanning of folders or directories directly. You have to scan the entire file system starting at the root directory. It is a very straightforward interface. You basically just have to point it in the right direction with the selection of which drive you want to scan and go.

It picked up a bit more of the encrypted file combinations than LADS did but was also not able to determine when an ADS existed if the main stream and alternate data stream were encrypted as a pair. CrucialADS detected an ADS when there was an ADS attached to a directory name. For instance it would find an ADS created as *:sneaky.txt* in any folder or directory. Also tried to rename the main filename, but the scan picked it up under the new name.

Encryption/Compression Detection Summary:

plain_text.doc:plain_text.doc - found
plain_text.doc:encrypted.doc - found
encrypted.doc:plain_txt.doc - found
encrypted (plain_text.doc:plain_txt.doc) - not found
compressed – not found

TDS-3[5]

Although not freeware, this is a very polished program to detect alternate data streams with many other malicious code detection features added. It even has a voice interface! Its many other functions won't be discussed here however. The ADS detection part is equal to that of CrucialSecurity with a bit more ease of use thrown in. You can scan specify directories, and for each ADS found you can view it, dump it to a file, delete it, delete the stream and associated host file, or view its properties. You can fix it with a mouse click and it also warns you of impending doom if you allow it to fix ADSs attached to your directories themselves. I found this site to be one of the most informative on Alternate Streams.

Encryption/Compression Detection Summary:

plain_text.doc:plain_text.doc - found
plain_text.doc:encrypted.doc - found
encrypted.doc:plain_txt.doc - found
encrypted (plain_text.doc:plain_txt.doc) - not found
compressed – not found

Streams[6]

Stream is another command line driven utility that is very similar to LADS. However I found this utility difficult to use. Like LADS, it does not accept directory names with spaces, etc. and when I ran it on C:\, it returned nothing. This was obviously in error due to several test files I had created on many different directories. Also when I did run it inside a directory that had known ADSs, it gave me the names of the ADS but didn't specify the full path name. This is a problem because I put multiple hidden streams with the same name to see if ADS checkers would catch all of them. I have no idea, in this case, which one it found.

There were several other ADS checkers that were on the Internet. These are a few that seemed to be visible throughout many of the Internet search engines.

Deleting streams can be difficult.

To delete a stream you must delete its parent (file or directory). This can be somewhat disturbing if you really want to get rid of it because of the fact that you must delete the directory or file that it is attached to. This means that for an infected root directory, you would have to reformat the disk and start over. This could get ugly if the ADS file (or files) attached to the root directory

were to be large enough to use a significant portion of the available disk space on your hard drive. It could be even more irritating if a virus dropper were to attach an encrypted or compressed alternate stream to each and every folder name in your file system.

However, I did find through some simple experiments that, if you can find the ADS, you can mitigate the impact it might have by creating a empty file (<10 byte file) and overwriting the discovered alternate stream. Suppose that you find a large stream attached to your root directory. For this example let's use *humungousfile.exe*. Well, you can't get rid of it but you can reduce its size to a null file and effectively eliminate it. First create a simple empty file in the 'cmd' window,

C:\ echo "" > cleanup.txt
C:\ type cleanup.txt > :humungousfile.exe

Now you've just overwritten the big file that was discovered hanging off your C:\ directory! Also, the disk space used by the ADS is now released back to the OS and you don't have it counting against your disk quota anymore (see next section). The ADS remains, it just exists under the name of *cleanup.txt*. Another method is to copy all files into another temporary directory (if you have the space on your drive), delete the infected folder, create a new one and move all of the files back. An annoyance for sure, plus it can get really painful if you find that several thousand ADS have been attached to the directory name or an ADS has been attached to every folder on your disk.

Available disk space calculated by DIR and Explorer doesn't consider space used by any ADS.

This is a really good way to drive someone nuts. The built-in OS utilities that tell you how much space you have are telling you that you have lots of space. So why can't I store anything?? Well, maybe it's because malicious code has created a large number of alternate data streams and use up most of your available space. A test was once conducted to see how many ADSs could be attached to one main data stream. Turns out to be 6748^7. This was a quick and dirty, non-scientific test but it shows that a large number of streams can be created for each host file. However, each ADS could be a larger stream, or multiple bogus files could be created with 6748 ADSs apiece. This is a good place to note that ADSs have a hierarchy that is only one level deep. You can't create an ADS with a filename of *goodfile.txt:badfile.txt:hidden.exe*.

Windows NT, 2000, and XP have a disk usage limitation feature called Disk Quotas. Limits can be placed on how much storage space a user can store information in a system. This function is tied to ownership. If you own a file or a directory, the disk storage space it uses is counted against the space the Administrator has allowed for you to use. ADS files can cause a bit of confusion when monitoring disk space utilization. It turns out that if you share any of your files, someone can attach and ADS to an existing file, hiding any malicious code behind a valid application or user file. That way, the rogue user can hide malicious code within a file system that is not traceable to them. As shown before, the file listing will not show the ADS. However, the disk space that it uses will be counted against the victim's quota. So the victim's file listing

can show a single small file and the Administrator will show the user as exceeding their space quota. The Administrator will also not know what file is causing the problem since disk quota tool only shows total usage and not file specifics. It was not possible to create a new ADS file in another user's space without permission. Only those files that were made public were susceptible. Looks like clever way to increase your own personal storage limits if you, on a low level, store some of your stuff on another person's file system!

Executable files can be stored in an ADS

Just like any other file, executables can be stored as an ADS. You can even execute an ADS from the command line. You can't run it directly as in

C:\some_directory\outlook.exe:evilcode.exe

However, Windows does allow you to execute it through the 'Start' command. So if you (in command line mode) were to type in

Start C:\some_directory\outlook.exe:evilcode.exe

The malicious code would execute. You can execute alternate streams from other programs. Diamond Computer Systems has written a free, downloadable test program that will execute hidden stream code (non harmful to your system I must add!) to demonstrate how easily this is done. So What? Well, through a virus dropper program, someone could "infect" a lot of machines by introducing encrypted or compressed ADSs. The ADS detection software wouldn't pick it up, so it would statically sit in your system until a trigger were to occur. This trigger could be a one line command as seen above. Would a virus detection program trip on a one line command? Doubtful. There are no signatures to look for. Does it show up on the task manager? No, the task manager window will only show the main file as executing and not the hidden stream. So, theoretically speaking, you could use a one line command, delivered by another method, and trigger thousand of machines that have been thought to be safe and clean from viruses. If the files were encrypted or compressed, the trigger file would indeed need to be larger. Virus writers try to write small, compact, less detectable code. With this approach, virus code could be a huge, multipartite beast that is hidden in plain sight.

On the positive side, this only worked under NT4.0. I was unable to get this same command to run under Windows 2000 or Windows XP. It appears Microsoft has addressed this issue, at least from the 'Start' command aspect. I also tried to run an ADS batch file from another batch file with no success and the CALL function also does not allow you to execute an ADS batch file from another batch file.

These are simplistic approaches to executing hidden streams. There are direct program methods to start an ADS. Diamond Computer Systems[5] offers a couple of good, free programs to check out your system's ability to work with ADSs. There may be other avenues of executing code in 2000 and XP such as through application macros. This looks like another good avenue for virus research.

Anti-virus programs have difficulty in detecting ADSs

Visiting many of the major AV company websites, I get the impression that they don't view ADSs as much of a problem. Their general philosophy is that you can not directly execute alternate streams. You have to have a companion program that call or executes the alternate stream. Besides, any file that is run will eventually have to be brought into system memory and will then no longer be hidden so that AV software will then discover it. Well, maybe. They are absolutely correct in that can not execute a file through a direct command such as

c:\temp\good.txt:sneaky.exe

As briefly mentioned before, you can split a potential virus into two parts. One being a very small "visible" part that in turn calls the alternate stream where the bulk of the evil code could reside. AV software will look for virus signatures. Since the visible part will be minimal code consisting of possibly little more than something like a ReadFile() function, it will look like any other normal file calling program and could elude any signature based recognition. This potentially could be a very stealthy virus. All this assumes you have the AV software in constant-monitoring mode. Otherwise, for scheduled or spot scans, it goes undetected.

Another really interesting way to accomplish an attack was proposed by Dartmouth[8]. This approach is exactly opposite to convention attack mentality of the malicious code doing the damage. This one uses your own defenses against you. Very clever! Make the assumption that a new virus has been created that is truly malicious and very easily detected in nature. However, instead of directly releasing it upon Internetdom as is typical, the virus is put into another delivery system, like a worm. The worm will then stealthily deliver the virus to many systems without the direct intervention of a user. The worm will covertly place the virus in an alternate data stream that is attached to a critical system file like taskmgr.exe, winlogon.exe, C:\, or anyone of a multitude of system executables that nobody knows exactly what they do. The attacker then gives the worm time to spread and deliver the virus code undetected. After a period of time, the attacker then releases the virus into the wild. AV companies then develop the signature recognition and release countermeasures or updates to their AV software. Here comes the good part. When one of the worm infected critical files is called upon to be executed, the file and its associated alternate data streams are loaded into memory. The virus that was hidden in the stream is now exposed for the AV software to see it. The AV scanner recognizes the signature of the new virus but does not know how to handle an ADS. How the AV software reacts would depend on how it's designed and configured. If it is set up to automatically delete infected files, the AV software you have on your machine to protect you has just deleted your critical system file! If the AV software moves the "bad" file to a quarantined directory, you have just moved you critical file to a place where the OS might not be able to find it and use it, thus the dreaded Blue Screen of Death. As Dartmouth points out, you now have a situation where an alternate data stream was used to bring down a system even though it was not executed directly, or called from a visible file, or even executed at all!

File wiping utilities have problems eliminating ADS streams.

This is a situation where a virus or worm could use ADSs to cause permanent problems. If some malicious code were to write to a large number of ADSs and chew up significant amount of data storage area, the only real option would be to wipe the disk and start over. Unfortunately, many of the most popular programs for file wiping or disk cleaning are still not ADS aware. Thus the data area used by ADSs will remain after disk wiping and your problems remain. Some have updated and some have stated they will be updated, so this problem is slowly going away (as far as the recovery part goes). Presently however, there remains the possibility that disk may still contain remnant sensitive data. There was an interesting find on the Internet regarding this. Paul McCartney had a significant amount of his personal cash dealings discovered on computers that were discarded by his bank[9] . I can see this opening up a whole new area for lawyers to jump into. What liabilities fall to the holder of personal information with regards to their upgrading computing equipment and the removal of old equipment? What legal recourse might be forthcoming if a company or individual who is privy to personal information does not use all known precautions and remedies for elimination of sensitive information?

They can be used as a Covert data channel

This is a problem that would present itself in an environment where security procedures and policies prevent unauthorized use of business owned computers. Without aftermarket tools, system administrators will not be able to find streams that could contain any number of variety of files not intended or allowed for corporate use. It also presents potential problems for computer forensic efforts. If disks are being analyzed for possible legal or maybe even counter-intelligence issues, the original disks are usually kept safely stored away while all the real work is done on imaged disks. If the disk imaging software is not ADS aware, critical information might be lost in the transfer. Another interesting possibility might be for data harvesting. Anything of value from passwords, to financial data, to maybe even keystroke capture could be stored in a hidden stream just waiting to be picked up at a later date.

Disk de-fragmentation and file integrity software

The best I can determine is that this is a second order effect of ADSs. It can lead to inefficient use of disk space and essentially result in a degrading of system performance. I wasn't able to find any threat scenarios relating directly to de-fragmentation problems and couldn't come up with one myself other than a slow denial of service attack. Another possible problem may arise from the use of a different possible virus defense tools or integrity checkers, in that they may not detect when a file has been changed with the addition of a hidden stream. Integrity checkers, such as TripWire, execute some CRC-like algorithm that develops a checksum or numerical print of the file system at any given point in time. Once that reference is established, any changes in the file system will trigger an alert. If only the main stream is used in the calculations for the system print, adding a hidden stream will elude detection by an integrity checker.

Summary

Through research and bit of primitive testing, it appears that Alternate Data Streams do indeed present a threat in the sense that malicious code can be hidden on a users system and executed at will through another companion process. Since it needs a companion process or human intervention, this makes it virus-like in nature. As long as it needs a companion process that has to reside in memory, there is the possibility of detection. There are no indications that it can have worm like characteristics where it can run and replicate under its own control. ADSs do not transport easily between differing file systems. File transfer protocols don't handle alternate streams. They can't be transferred to diskettes. Only the main stream is transferred. So, at least in their present form, Alternate Data Streams do not appear to have the ability to quickly spread and infect large numbers of systems. For now that is.

There are other ways that an Alternate Stream can be used to do damage to users system. The proposed scenario by Dartmouth being the most plausible, possible, and damaging at this point in time. Bear in mind however, that this is an opinion not based on a deep understanding of the internals of Windows OSs and advanced software program writing ability. It is apparent though that the ability to read/write/modify information in an ADS is trivial and is presently available in nice GUI form. .

Detecting ADSs was easily done for non-encrypted or non-compressed streams. But, the fact remains that encrypted and compressed ADSs went undetected. Not sure this is a fair test however since either PGP encrypting or WinZip compressing an ADS *main:stream* file pair creates a single file. In any case this represents a method of hiding an ADS without fear of discovery from scanning or integrity checking tools.

A couple of ADS attacks have already been developed. One (W2K.stream)[10] was harmless. The other (IIS :$DATA)[11] was able to retrieve information from user's systems. So it is apparent that attackers are looking for ways to exploit this technology.

So, I go away with the impression that with that much storage ability that can remain undetected through encryption or compression with readily available tools like PGP and WinZip, it poses a very inviting target for malicious code writers and, IMHO, a matter of time before new exploits appear.

Acknowledgements

[1] Security Focus Online, BUGTRAQ Archive, Mar 20 1998
http://online.securityfocus.com/archive/1/8822

[2] HOWTO: Use NTFS Alternate Data Streams (Q105763)
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q105763

[3] LADS - List Alternate Data Streams
http://www.heysoft.de/nt/ep-lads.htm

[4]What is CrucialADS?
http://www.crucialsecurity.com

[5] Wayne Langlois' TDS:Trojan Defence Suite, The World's most comprehensive Anti-Trojan System
http://tds.diamondcs.com/au/

[6] Sysinternals Freeware – Information for Windows NT and Windows 2000- Miscellaneous Tools for Windows
http://www.sysinternals.com/ntw2k/source/misc.shtml

[7] Creation of Many File Streams on NTFS
http://www.cerias.purdue.edu/coast/ms_penetration_testing/v50.html

[8] NTFS Advisory – Investigative Research for Infrastructure Assurance (IRIA)
http://www.ists.dartmouth.edu/IRIA/knowledge_base/NTFS_Advisory.htm

[9] Cullen, Drew. "Paul McCartney account details leaked on second user PC." The Register, 09/02/2000
http://www.theregister.co.uk/content/archive/9137.html

[10] Sophos virus analysis: W2K/Stream
http://www.sophos.com/virusinfo/analyses/w2kstream.html

[11] "::$DATA" Data Stream name of a File May Return Source
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q188806

References:

NTFS ADS Viruses
http://www.cknow.com/vtutor/vtntfsads.htm

Detecting Alternate Data Streams
http://www.ntsecurity.net/Articles/Print.cfm?ArticleID=16189

NTFS Streams
http://biznix.org/whylinux/windows/ads.html

Some antivirus scanners blind to alternate data streams
http://www.securitywatch.com/scripts/news/list.asp?AID=3826

Windows wipe utilities fail to shift stubborn data stains
http://www.theregister.co.uk/content/55/23759.html

Windows, NTFS and Alternate Data Streams
http://rr.sans.org/threats/win_NTFS.php

Alternate Data Streams, the Hidden Threat
http://rr.sans.org/win/ADS.php

NTFS Alternate Data Streams
http://win2000mag.com/Articles/Print.cfm?ArticleID=19878

Windows 2000 Streaming Virus
http://rr.sans.org/win2000/streaming_virus.php

An In-depth Look at W32.Winux and W2K.Stream and the Ever growing "Proof of Concept" Virus
http://rr.sans.org/malicious/w2kwinux.php

What Forensic Analysts/Investigators should know about NT MULTIPLE DATA STREAMS
http://www.dmares.com/maresware/multdata.htm