Introduction to Alternate Data Streams

Posted: July 22, 2015 by [Pieter Arntz](https://blog.malwarebytes.com/author/metallicamvp/)  
Last updated: March 30, 2016

What are Alternate Data Streams?

Alternate Data Streams (ADS) are a file attribute only found on the [NTFS file system](https://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx).

In this system a file is built up from a couple of attributes, one of them is *$Data*, aka the data attribute. Looking at the regular data stream of a text file there is no mystery. It simply contains the text inside the text file. But that is only the primary data stream.

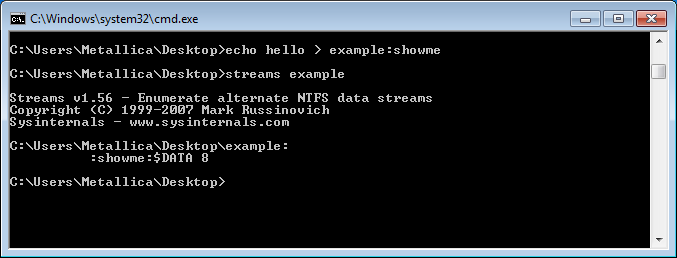
This one is sometimes referred to as the unnamed data stream since the name string of this attribute is empty ( “” ) . So any data stream that has a name is considered alternate.

These data streams suffer from a bad reputation since they have been used and abused to write hidden data. Varying from data about where a file came from to complete malware files (e.g. [Backdoor.Rustock.A](https://www.symantec.com/security_response/writeup.jsp?docid=2006-060111-5747-99&tabid=2))

If you are up for an experiment, we can easily create and read an alternate data stream.

Streams

The first tool you can use was developed by Sysinternals (later bought by Microsoft) and is called [Streams](https://technet.microsoft.com/en-us/sysinternals/bb897440.aspx) *(nomen est omen)*.

[](https://blog.malwarebytes.org/wp-content/uploads/2015/07/voorbeeld.png)

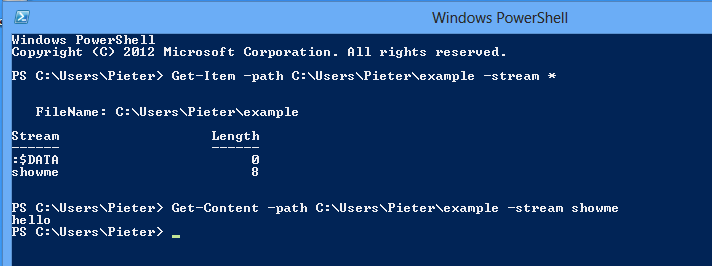
In the example above we used the echo command to create an empty file called example with an alternate data stream called showme.

By using streams we can check which files have alternate data-streams.  In the results visible in the above command prompt,  *$Data* is the name of the attribute (as discussed earlier) and the 8 tells us the size.

But since we are looking at it, we obviously would like to see what is inside the alternate data streams. Unfortunately, streams do not offer that option.

Get-Item

If you are using Windows 8 (or newer) there is a built-in option to read ADS. You can use PowerShell commands to achieve this. For those that have no experience with it, you can start it by typing PowerShell in the Run box (Windows key + R) and follow the lines in this screenshot.

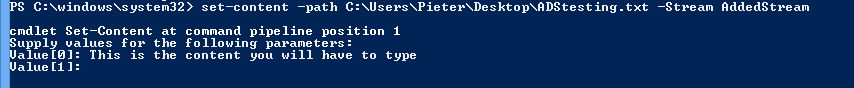
[](https://blog.malwarebytes.org/wp-content/uploads/2015/07/powershell.png)

Set-item

Another thing that you can do with Powershell is add streams to a file. The Powershell command syntax is:

set-content - path {path to the file} - stream {name of the stream}

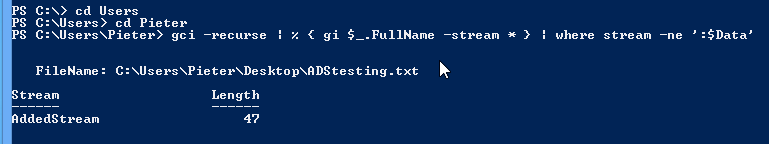
Doing so will initiate a cmdlet where you can enter the content of the stream under Value[i]

[](https://blog.malwarebytes.org/wp-content/uploads/2015/07/PSsetcontent.png)

Search for ADS

If you want to search a directory or drive for ADS you can use this command in the root of the target:

gci -recurse | % { gi $\_.FullName -stream \* } | where stream -ne ':$Data'

[](https://blog.malwarebytes.org/wp-content/uploads/2015/07/PSsearch.png)

Be warned that if you include the Windows directory in your search you will likely receive an enormous list.

Remove ADS

A word of warning here. Removing ADS is not always advisable. Some of them are needed for the proper use of the software that created the streams. So make sure you have done your research before removing them. The syntax is:

remove-item –path {path to the file} –stream {name of the stream}

Malwarebytes Anti-Malware scans for and removes unwanted ADS (as Rootkit.ADS)

Summary

Alternate Data Streams (ADS) have been given a bad reputation because their capability to hide data from us on our own computer, has been abused by malware writers in the past. Hopefully this article will clear up some of the questions and mystique you had about ADS.

Resources:

* [Alternate data streams in NTFS](http://blogs.technet.com/b/askcore/archive/2013/03/24/alternate-data-streams-in-ntfs.aspx)
* [Exploring Alternate Data Streams](http://www.rootkitanalytics.com/userland/Exploring-Alternate-Data-Streams.php)

**Alternate Data Streams in NTFS**

* 03/24/2013
* 5 minutes to read

This blog has been a long time coming. There is a bit of confusion about the subject of ***alternate data streams*** (ADS) and no small amount of suspicion. So I want to take a few minutes to set the record straight on ADS.

A couple years ago I wrote a blog on NTFS attributes.

[http://blogs.technet.com/b/askcore/archive/2010/08/25/ntfs-file-attributes.aspx](https://docs.microsoft.com/en-us/archive/blogs/askcore/ntfs-file-attributes)

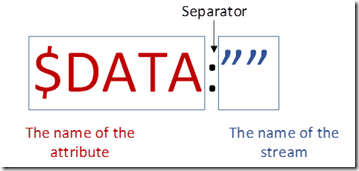
You might want to review that blog before continuing. I’ll wait….

Welcome back.

One of the common questions I get is, “Robert. What is an alternate data stream?”

My reply is always the same, “It is a data stream that is alternate”.

I don’t mean to be smart aleck about it…but that’s what it is. We know from my older blog that a file is divided up into ‘attributes’ and one of these attributes is $DATA or simply called the data attribute. It is the part of the file we put data into. So if I have a text file that says, “This is my text”, then if I look at the data attribute, it will contain a stream of data that reads, “This is my text”. However, this is the normal data stream, sometimes called the primary data stream, but more accurately it is called the ***unnamed data stream***. Why? Because it is a data stream that has no name. In the jolly land of programming it is referred to as $DATA:””

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/3482.image_106A3119.png)

The name of the stream will appear between the quotes. Since this is an unnamed data stream, there isn’t anything there.

Now that we know what the unnamed data stream looks like, we can start thinking in terms of alternates. Knowing that the place we normally store data is the unnamed data stream, if a stream has a name, it is alternate. So if I had a file with an ADS named SecondStream, its full name would be, $DATA:”SecondStream”

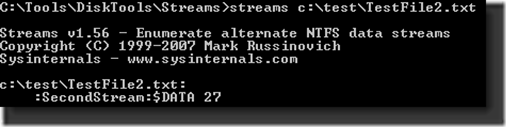
[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/7536.image_4F5BB1B4.png)

This is all good and fine, but unlike the unnamed data stream, we can’t see the ADS. Or can we? The answer is, yes we can. But you have to use a method different than just opening the file in NotePad.

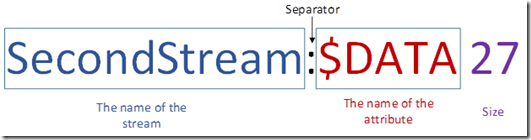
There are a number of tools out there that will allow you to view and manipulate ADS. One that Microsoft has provided for years is called STREAMS.EXE.

[http://technet.microsoft.com/en-us/sysinternals/bb897440.aspx](https://technet.microsoft.com/en-us/sysinternals/bb897440.aspx)

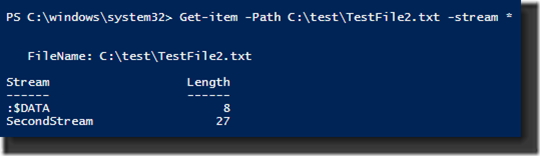
STREAMS.EXE will display any ADS the file has.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/4527.clip_image005_6E322592.png)

The formatting is a little different.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/1738.image_2D23A62E.png)

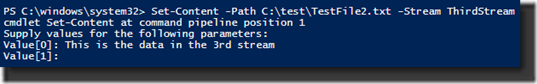
STREAMS.EXE is fine, and I’ve used it for years, but with the release of Win8/Server 2012, I’ve discovered a new way of dealing with ADS….Windows PowerShell. Using the cmdlet, Get-Item, I can get more information than I did with STREAM.EXE.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/4863.clip_image008_1E0CC754.png)

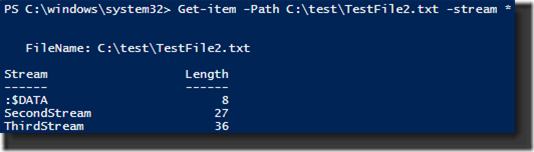
The output shows not only the name of the ADS and its size, but also the unnamed data stream and its size is also listed (shown as :$DATA). And now that I know the name of the ADS, I can use the Get-Content cmdlet to query its contents.

[clip_image009](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/5383.clip_image009_0AEB9AA8.png)

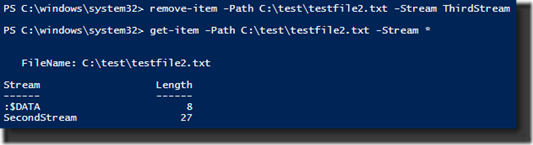
STREAM.EXE can’t display what’s actually in an ADS. Here’s another trick that STREAM.EXE can’t do….create data streams. Using Set-Content, I’ll create a second ADS in the same file and add a line of text.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/1538.clip_image010_0287C551.png)

And again, we can query for the streams using Get-Item.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/1050.clip_image011_417945EC.png)

And finally, we can remove an ADS using the Remove-Item cmdlet.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/4336.clip_image012_527D73CF.png)

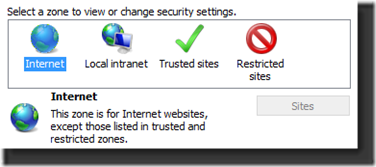
Now we know what ADS is, how to query for ADS, how to create ADS, and how to delete ADS. So what is the big deal?

The big deal is that since ADS isn’t easily visible, it has become a cute way to hide data. Unfortunately it has also been used in the past to hide malicious code. This is how ADS got a bad name. In fact, a number of people that approach me about ADS already know that they have files with alternate data streams and they think they are infected with viruses.

Calm down. The mere presence of an ADS doesn’t mean that there is a problem. In fact, Microsoft uses ADS for a number of functions. I can almost guarantee that if you are reading this, you probably have some ADS on your computer. Let’s take a look at a couple examples.

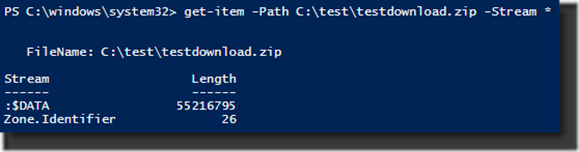
**Internet Explorer:** Ever download an executable file from the Internet and then get warned about it when you ran it? How does that work?

When the file is downloaded, IE slaps an ADS on it. The stream will store a tag that tells Windows what zone the file was downloaded from.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/6811.clip_image013_3F5C4723.png)

*Look Familiar?*

So using what I’ve learned so far, I can look at one of the files I’ve downloaded from the internet and see if there is an ADS on it.

[](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/7331.clip_image014_7DE194C9.png)

Yes, it is called ‘Zone.Identifier’. And then we can query the contents of the ‘Zone.Indentifier’ ADS.

[clip_image016](https://msdnshared.blob.core.windows.net/media/TNBlogsFS/prod.evol.blogs.technet.com/CommunityServer.Blogs.Components.WeblogFiles/00/00/00/59/75/metablogapi/4150.clip_image016_51C497D8.png)

Now we know that the file was downloaded from zone 3. Using the zone chart we can see it came from the Internet zone.

Value Setting  
------------------------------  
0 My Computer  
1 Local Intranet Zone  
2 Trusted sites Zone  
3 Internet Zone  
4 Restricted Sites Zone

Notice that my test download file is in a test directory. This means I moved the file here from my download directory. This is the cool thing about ADS, since it is part of the file, it moves with the file. Even if I copied it, the ADS would be on the new copy as well.

Other Internet browsers use ADS in a similar fashion.

**File Classification Infrastructure**: FCI is very dependent on ADS. The way that the classification works is that it puts tags on your files that allows you to keep track of what the file was classified as, no matter what happens with the file. It could be edited, copied, moved to another server, and its classification tags remain intact.

**Others**: Office files and Outlook Express file use ADS. And it isn’t limited to Microsoft programs. Numerous programs utilize the ADS functionality.

The point is that if you discover ADS on your system, it isn’t necessarily a bad thing. And just blindly stripping these data streams out of files can actually do a great deal of harm.

And now that you have some tools to use for querying alternate data streams, they won’t be so scary.

Thank you for your time and I hope this was educational.

Robert Mitchell  
Senior Support Escalation Engineer  
Microsoft Corp.

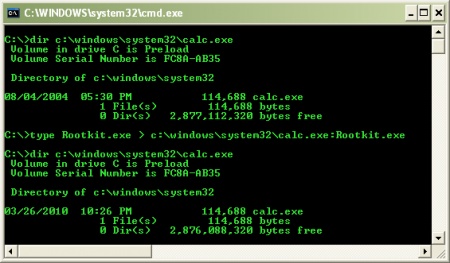
#### What is an Alternate Data Stream (ADS)?

[**TOP**](http://www.rootkitanalytics.com/userland/Exploring-Alternate-Data-Streams.php#topics)

Alternate Data Stream (ADS) is the lesser known feature of Windows NTFS file system which provides the ability to put data into existing files and folders without affecting their functionality and size. Any such stream associated with file/folder is not visible when viewed through conventional utilities such as Windows Explorer or DIR command or any other file browser tools. It is used legitimately by Windows and other applications to store additional information (for example summary information) for the file. Even 'Internet Explorer' adds the stream named 'Zone.Identifier' to every file downloaded from the internet.  
  
Due to this hidden nature of ADS, hackers have been exploiting this method to secretly store their Rootkit components on the compromised system without being detected. For example, the infamous Rootkit named 'Mailbot.AZ' aka 'Backdoor.Rustock.A' used to hide its driver file into system32 folder (C:\Windows\system32) as a stream '18467'.

#### Playing with ADS

[**TOP**](http://www.rootkitanalytics.com/userland/Exploring-Alternate-Data-Streams.php#topics)

It is easy to create alternate data streams for the file or folder. Here are the simple commands (use the cmd prompt to launch these commands)  
  
Create simple text stream,  
type c:\test.txt > c:\windows\system32\calc.exe:test.txt  
  
View it using the notepad  
c:\notepad.exe c:\windows\system32\calc.exe:test.txt  
  
Hiding the rootkit.exe as stream within the windows calculator,  
type c:\rootkit.exe > c:\windows\system32\calc.exe:rootkit.exe'  
  
Here is the screenshot showing the above steps in execution....  
  
[](http://www.rootkitanalytics.com/images/ads_hiding_rootkit.jpg)  
\*To zoom in - click the image\*  
  
Note that there is no change in the size of calc.exe even after adding the stream file to it.

#### Program to Enumerate Streams

[**TOP**](http://www.rootkitanalytics.com/userland/Exploring-Alternate-Data-Streams.php#topics)

Here is the short demo program which enumerates all the streams within the given file or folder.  
  
**void EnumStreams(char \*strFilePath)  
{  
PVOID streamContext = 0;  
DWORD dwReadBytes, seek\_high;  
WIN32\_STREAM\_ID streamHeader;  
WCHAR strStreamName[MAX\_PATH];  
char strBuffer[1024];  
  
  
HANDLE hFile = CreateFileA( strFilePath,  
GENERIC\_READ,  
FILE\_SHARE\_READ,  
NULL,  
OPEN\_EXISTING,  
FILE\_FLAG\_BACKUP\_SEMANTICS,  
NULL );  
  
if( hFile == INVALID\_HANDLE\_VALUE )  
{  
printf("%s : Failed to open the file %s,  
Error=0x%.8x", \_\_FUNCTION\_\_, strFilePath, GetLastError());  
  
return;  
}  
  
while(1)  
{  
//check if we have reached the end of file....  
if ( FALSE == BackupRead(hFile,  
(LPBYTE)&streamHeader,  
(LPBYTE)&streamHeader.cStreamName-(LPBYTE)&streamHeader,  
&dwReadBytes,  
FALSE,  
FALSE,  
&streamContext) )  
{  
break;  
}  
  
//check if we have read the stream header properly...  
if( (long)dwReadBytes != (LPBYTE)&streamHeader.cStreamName-(LPBYTE)&streamHeader )  
break;  
  
//we are interested only in alternate data streams....  
if(streamHeader.dwStreamId == BACKUP\_ALTERNATE\_DATA)  
{  
if (streamHeader.dwStreamNameSize != 0 )  
{  
if( BackupRead(hFile,  
(LPBYTE)strStreamName,  
streamHeader.dwStreamNameSize,  
&dwReadBytes,  
FALSE,  
FALSE,  
&streamContext) )  
{  
strStreamName[streamHeader.dwStreamNameSize/2]=L'\0';  
  
//Reformat the stream file name ... :stream.txt:$DATA  
sprintf\_s(strBuffer, 1024, "%S", &strStreamName[1]);  
char \*ptr = strchr(strBuffer, ':');  
if( ptr != NULL )  
\*ptr = '\0';  
  
printf("\n Found Stream - %s", strBuffer);  
  
}  
}  
}  
  
// jump to the next stream header  
if (BackupSeek(hFile, ~0, ~0, &dwReadBytes, &seek\_high, &streamContext) == FALSE)  
{  
//for any errors other than seek break out of loop  
if (GetLastError() != ERROR\_SEEK)  
{  
// terminate BackupRead() loop  
BackupRead(hFile, 0, 0, &dwReadBytes, TRUE, FALSE, &streamContext);  
break;  
}  
  
streamHeader.Size.QuadPart -= dwReadBytes;  
streamHeader.Size.HighPart -= seek\_high;  
  
BYTE buffer[4096];  
  
while(streamHeader.Size.QuadPart > 0)  
{  
if (dwReadBytes!=sizeof(buffer) || !BackupRead(hFile,  
buffer,  
sizeof(buffer),  
&dwReadBytes,  
FALSE,  
FALSE,  
&streamContext) )  
{  
break;  
}  
streamHeader.Size.QuadPart -= dwReadBytes;  
}  
}  
} //main while loop  
  
//Finally clean up the buffers use for seeking  
if (streamContext)  
BackupRead(hFile, 0, 0, &dwReadBytes, TRUE, FALSE, &streamContext);  
  
CloseHandle(hFile);  
  
return;  
}**Above program initially opens the input file using the FILE\_FLAG\_BACKUP\_SEMANTICS for reading streams. Next it calls the BackupRead function to read the stream header. If the header contains the flag BACKUP\_ALTERNATE\_DATA then it points to a hidden stream file. In such a case it proceeds to reading the stream file name which is present after the stream header. After that it moves file pointer to next stream header through BackupSeek function.  
  
The same process is repeated until all streams present in the specified files are discovered. Also note that the same program can be used to detect streams within folder as well.

#### Well Known Alternate Data Streams

[**TOP**](http://www.rootkitanalytics.com/userland/Exploring-Alternate-Data-Streams.php#topics)

There are numerous applications including Windows which internally use alternate data streams for various purposes. Here are some of the well known streams....

* SummaryInformation  
  This stream is created by Windows when user updates the summary information for the file.
* DocumentSummaryInformation  
  This stream is created by Windows when user updates the summary information for the file.
* {4c8cc155-6c1e-11d1-8e41-00c04fb9386d}  
  This is stream with zero size created by Windows when user updates the summary information for the file.
* Zone.Identifier  
  This is another well known stream created by Internet Explorer for every downloaded file. It is basically text stream with size normally less than 50 bytes.
* encryptable  
  This is a stream with zero size attached to the file 'Thumbs.db'.
* favicon  
  This is icon stream attached to the favorite links stored by Internet Explorer.
* AFP\_AfpInfo  
  This is stream of icon type belongs to Macintosh system.

In addition to legitimate programs, it is also being used by malicious Rootkit programs such as Mailbot.AZ, Trojan.Win32.Agent.alt etc to hide their drivers.

#### Tools to detect ADS

[**TOP**](http://www.rootkitanalytics.com/userland/Exploring-Alternate-Data-Streams.php#topics)

Due to nature of ADS, normal file browser tools cannot detect these hidden streams. We need special tools to discover and manipulate these streams. There are many ADS tools available, but most of them resort to only enumerating streams. Most of these are console based tools which are good for automation but not much useful for home users.  
  
Here comes the new advanced tool, StreamArmor which can quickly scan for hidden alternate data streams as well as clean them completely from the system. It's advanced auto analysis coupled with online threat verification mechanism makes it the best tool available in the market for eradicating the evil streams.

#### Conclusion

[**TOP**](http://www.rootkitanalytics.com/userland/Exploring-Alternate-Data-Streams.php#topics)

In short, ADS not only makes it easy for Rootkit programs to hide themseleves but also provides the covert launch pad to execute stealthily without making noise. Only sophisticated tools such as StreamArmor will greatly help in uncovering and destroying such hidden threats.

I'm sure it comes as no great shock that I am a member of a number of listserves on digital forensics. One question that seems to come up every few weeks is NTFS Alternate Data Streams. There have been many excellent articles on ADS, so I don't propose to go heavily into the details here. I will just include an overview and some of the better references. This is a basic overview. If you want more details, check out the links for some really good write-ups.

#### **What are Alternate Data Streams?**

Alternate Data Streams (ADS) have been around since the introduction of windows NTFS. They were designed to provide compatibility with the old Hierarchical File System (HFS) from Mac which uses something called resource forks.

Basically, ADS can be used to hide the presence of a secret or malicious file inside the file record of an innocent file. That is, when windows shows you a file, say "readme.txt", the metadata that tells your system where to get "readme.txt" may also contain information for "EvilSpyware.exe". Thus, malicious files may be on your system and you cannot see them using normal means.

#### **How are ADS used?**



A blackhat on your system can trojanize one of your regular, trusted files and use it to hide their toolkit so that your system will not see it. Or, a criminal may use the technique to hide something on their own system so that others will not readily find it.

To see how easy this is, try it yourself:

1. Go to a temp directory and create a txt file
   1. Right Click, select New > Text document, enter the name "readme"
2. Open the file in notepad and enter text "I am a harmless file". Save and close.
3. Open a command prompt Start > Run? , enter "cmd", click OK
4. In the command window, go to your temp directory.
5. Type the following to confirm that you have a good victim file: notepad readme.txt
6. A notepad window will pop up and show your harmless text.
7. **This has all been setup. Now we get to the ADS part.**
8. Type the following to save secret notes to a different text file and hide the reference to that text file inside of "readme.txt", echo this is my list of secret evil stuff > readme.txt:secret.txt
9. Now if you type the following you will see only your original file... "notepad readme.txt"
10. ? but if you type this, you will see your secret message "notepad readme.txt:secret.txt"
11. Not only can you hide text, but you can hide executable as well. You also don't need to use a txt file as the host for hiding your file. Several other file formats, including directories can be used to hide files. For a great discussion of both of these techniques, check out <http://www.irongeek.com/i.php?page=security/altds> and <http://www.windowsecurity.com/articles/alternate_data_streams.html>

#### **How can I detect ADS?**

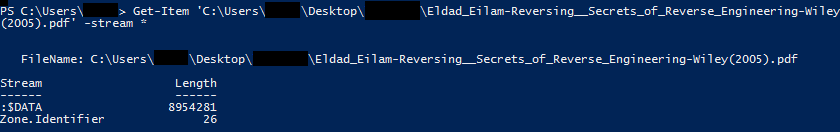
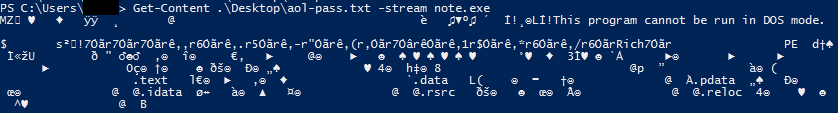
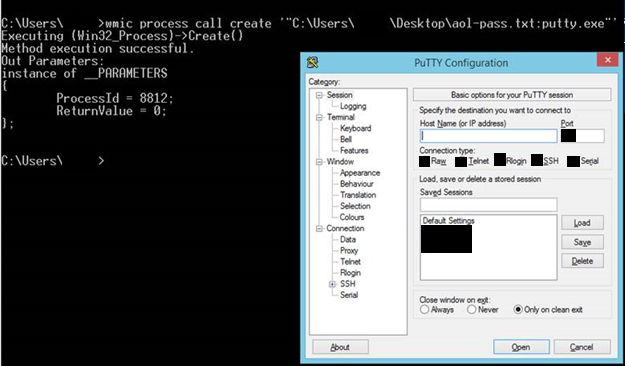
There are quite a number of ADS tools out there at the moment, here are a few to get you started.

* [ADS Detector for Explorer](https://www.codeproject.com/Articles/5153/Visual-browsing-of-alternative-data-streams-in-Win)
* [ADS Locator](http://www.safer-networking.org/en/tools/tools_ads.html)
* [Crucial ADS GUI Scanner](http://www.crucialsecurity.net/index.php?option=com_content&task=view&id=95&Itemid=137)
* [LADS](http://www.heysoft.de/nt/ep-lads.htm)
* [Streams](http://www.microsoft.com/technet/sysinternals/utilities/streams.mspx)
* [The Forensic Toolkit v2.0](http://www.accessdata.com/forensictoolkit.html)
* The following blog post presents the concept of Alternate Data Stream and how it has been used in the wild, in the past as well as these days: as a technique used to hide a malicious payload in the file system. Although it might be considered an old technique, it is still relevant today and common in the wild.

#### **What is an Alternate Data Stream?**

* Alternate Data Stream (ADS) is the ability of an NTFS file system (the main file system format in Windows) to store different streams of data, in addition to the default stream which is normally used for a file. When this feature was created, its main purpose was to provide support to the macOS [Hierarchical File System](https://en.wikipedia.org/wiki/Hierarchical_File_System) (HFS).
* The NTFS file system contains files with attributes. The relevant attribute for our scope is the $DATA attribute, which is used to store the data streams of a file.
* In the past, it was common to store a malicious payload within an ADS of a legitimate file. But today, many security solutions will detect and scan ADSs’. However, we will show you in this post, an ADS can still be used to run payloads while being hidden from the user as well as various security products.
* To simplify the concepts that are presented in this article, all the examples are based on native tools in Windows.

#### **How to Use ADS**

* In the good old days of Windows Vista, Alternate Data Streams (ADS) was a common method for malware developers to hide their malicious code. An alternate Data Stream was effectively used to his the presence of malicious corrupting files, by squeezing it inside a legitimate file.  By putting malware in the Alternate Data Stream, the Windows file would contain information and directions for the legitimate files, but also for the malicious file. In order to run the malicious code the hacker has deposited in the Alternate Data Stream, s/he would want to create a symlink using the mklink command. But doing this necessitates having administrative rights on the machine. Considering the normal behavior of a user doesn’t require running operating as an admin, hackers tend to approach their targets assuming that they land on a box without any admin rights.
* **Enumeration** – The mainstream under $DATA is the empty string (thus: “”, also called the unnamed stream). This stream is used to contain the regular stream of a file.
* The following is an example of enumerating the data streams of a file with PowerShell:
* 
* You can also use dir /R, or SysInternals\streams.exe to enumerate an ADS of a file / recursively on a directory
* As you can see above, the file only contains the unnamed stream.
* However, sometimes you can find an ADS within a legitimate file, as there are several legitimate uses for this technique.
* Below is an example of enumerating data streams of a file with an ADS with PowerShell:
*   
  This is also an example of a legitimate use of ADS: the Zone. Identifier. This field presents the trust level of the file’s source. This is a browsers’ usage to store the Its macOS corresponding is the “quarantine bit”.
* **Creation** – While there can be legitimate uses for ADS, ADS can also be used to store a malicious payload. Attackers might prefer this method over saving the payload in the unnamed stream, because its presence will be hidden from the user – it is not accessible from explorer.exe (and the displayed size of a file does not include ADS’s size). In addition, it is not displayed by default with the “dir” command (however, /R flag displays ADS, and is applied from Windows Vista).
* Creating an ADS is quite simple.
* An example of creating an ADS with type command:
* 
* Until Windows XP, it was possible to run an ADS with the start command. However, since later versions of Windows, it is no longer possible.
* **Data Access** – Abusing ADS is still a relevant threat, as there are multiple ways to read and execute an ADS.
* The first usage is accessing the data. The content of the ADS can be accessed easily. The following is an example with PowerShell:
* 
* **Execution** – In addition to reading the data, it is also possible to execute it. Although it’s not possible to execute an ADS with “start” command, it can still be run from other programs. For example, with WMI:
* “**wmic process call creates**” – runs a process with WMI. It can also run an ADS. Example:
* 
* There are additional legitimate native tools of Windows that can be used to directly run an ADS. For example, a DLL can be saved inside an ADS and called by DllMain through rundll32.exe; also, an HTA can be saved inside an ADS and called through mshta.exe; and so on.

#### **Malicious Campaigns Abusing ADS**

* It may seem that Windows has managed to handle the threats that reside within ADS. However, there are many malware families in the wild that still abuse ADS’s.
* [Another example from 2017 is BitPaymer](https://nakedsecurity.sophos.com/2017/09/21/how-bitpaymer-ransomware-covers-its-tracks/), ransomware which used two ADS’s in the attack sequence, in order to add stealth to the actions and hide malicious activity. The first ADS was created in order to run the “net view” command to obtain network shares, then the second ADS scrambles the data on the disk and network shares.
* To conclude, abusing ADS is a technique that is still relevant nowadays. Indeed, recent Windows versions have limited and hardened the functionality of ADS. However, basic command of native Windows tools such as Power-Shell and WMI, or some knowledge of Windows command-line allows threat actors to continue and leverage ADS in their attacks. Interestingly, recent campaigns using ADS are considered rather sophisticated and tend to minimize the use of persistent files on disk.

Alternate Data Stream (shortened as ADS) is a feature of the Windows New Technology File System (NTFS) that, surprisingly, has both good and bad aspects.

In this article, we’ll uncover both its two sides so that you can be prepared at using it.

### **WHAT ARE ALTERNATE DATA STREAMS?**

An Alternate Data Stream is a little-known feature of [the NTFS file system](https://blog.foldersecurityviewer.com/ntfs-tutorial-understanding-ntfs-permissions/). It has the ability of forking data into an existing file without changing its file size or functionality.

Think of ADS as a ‘file inside another file’.

ADS exists in all versions of Microsoft’s NTFS file system, and it has been available since Windows NT was released.

It was originally intended to allow for compatibility with Macintosh’s Hierarchical File System (HFS).

Currently, all Windows Operating Systems, including the latest Windows 10 OS, supports the ADS feature.

### **SO, WHAT CAN YOU DO WITH ALTERNATE DATA STREAMS?**

ADS can allow you to store any type of file, such as texts, audios, videos, images, or even nefarious codes like viruses or trojans.

## **REPORT NTFS PERMISSIONS ON WINDOWS FOLDERS IN UNDER 60 SECONDS!**

## **NEW​ VERSION AVAILABLE - NEW MODERN UI**

​Discover the security leaks in your Windows fileserver environment now!

#### **[​GET YOUR FREE TRIAL](https://www.foldersecurityviewer.com/product" \t "_blank)**

ADS contains metadata for identifying files according to various attributes, such as author, title, date modified, and more.

Furthermore, hackers can use Alternate Data Streams to launch Denial of Service Attacks (DOS).

### **BENEFITS OF ADS**

Before we look at how an attacker can hijack ADS for malicious reasons, let’s talk about some of its benefits, as described below.

* Windows Resource Manager leverages ADS to identify high risk files that shouldn’t be accessed.
* The Windows operating system uses ADS to encrypt and store files in a secure manner.
* The Windows Attachment Manager uses ADS as a file scanner. This explains why sometimes you receive warnings when you open a file downloaded from the Internet.
* The SQL Database server uses ADS to maintain database integrity.
* Citrix’s virtual memory uses ADS to boost DLL loading speed.
* Anti-virus applications, such as Kaspersky, uses ADS to enhance the scanning of files.

### **CREATING AN ALTERNATE DATA STREAM**

Creating an Alternate Data Stream is not rocket science; it’s extremely easy.

Basic DOS commands like **type** can be used, in conjunction with the **[ > ]** redirect symbol and **[ : ]** colon symbol, to fork a file into another file.

Let’s demonstrate the steps of using ADS to hide information in a file.

**Step 1: Open the terminal and create a text file**

C:> echo Today is going to be a great day > file1.txt

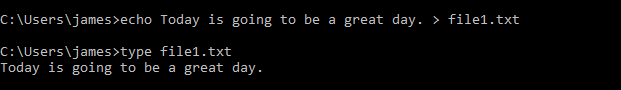
This command saves the given string to a text file called **file1.txt**

**Step 2: Confirm the contents of the file**

Let’s now confirm the contents of the file by using the **type** command, as shown below.

C:> type file1.txt

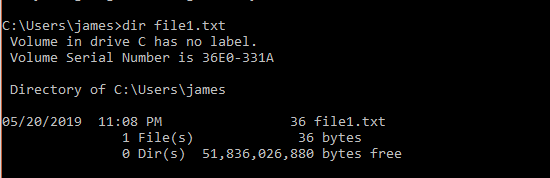
Today is going to be a great day



Everything is working well, just as expected.

Then, let’s check the directory listing.

C:> dir file1.txt



**Step 3: Append new content to the hidden file**

Let’s execute the following command:

C:> echo The sun is all up and the coast is clear > file1.txt:hidden

It appears that we have created a new file called **file1.txt:hidden,**which is not the case.

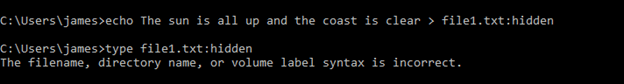
We have just created an Alternate Data Stream within the **file1.txt** file under the name **‘hidden’.**

The filenamed**file1.txt:hidden**does not exist.

In fact, if we try to examine its contents, the Windows prompt will return an error, as illustrated below.

C:> type file1.txt:hidden

The filename, directory name or volume label syntax is incorrect



However, we can reveal the contents of the file, as shown below.

C:> more < file1.txt:hidden

The sun is all up and the coast is clear



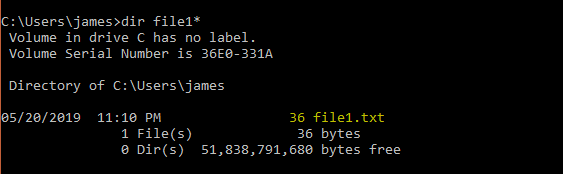
Remember, the ‘original’ data stream is still there.

C:> type file1.txt

Today is going to be a great day

Yet, when we check the directory, there’s only one file, which is **file1.txt.**

C:> dir file1\*



Here are three interesting points to note about the last directory listing.

1. **The timestamp has changed after adding the Alternate Data Stream** file to the existing file. That is the only indication that a change has indeed happened.
2. **The file size remains unchanged** as evidenced by the prefix**36**in **file1.txt** when checking the directory listing. This implies that you could have many ADS files within a file without your knowledge.
3. Because of the subtle changes, it’s difficult to detect Alternate Data Stream files unless you use**a third-party tool.**

### **RISKS ASSOCIATED WITH ALTERNATE DATA STREAMS**

Alternate Data Streams enables information to be hidden within other files. As such, it can be a security risk.

An attacker can easily store malicious codes or payloads and use them to cause damages to your system.

Let’s consider this example.

c:> type c:\windows\system32\calc.exe > file1.txt:calc.exe

The above command copies the Windows calculator program into an ADS file called **calc.exe**, which is linked to **file1.txt.**

To launch the hidden **calc.exe** copy from its ADS in **file1.txt**, an attacker can run the following command.

c:>start c:\file1.txt:calc.exe

Now, suppose that was not a **calc.exe** file but a destructive malware, it could lead to extensive damages to your system.

### **CONCLUSION**

The greatest challenge with Alternate Data Streams is that, if used for nefarious purposes, they are extremely difficult to detect, unless you use third party applications.

Additionally, ADS cannot be turned off.

Therefore, it’s critical to institute robust measures to prevent its abuse.

Do you have any question or comment?

Please post them below.

## **NTFS Alternate Data Stream (ADS)**

### **Author: Conrad Chung, 2BrightSparks Pte. Ltd.**

[Download PDF version](https://www.2brightsparks.com/resources/articles/NTFS-Alternate-Data-Stream-ADS.pdf) [opens new window]

In 1993, Microsoft released the Windows NT operating system based on the popular NT platform. Since then, it has continued to develop newer operating systems like Windows 10 and Windows Server 2019 using the same platform. Despite the widespread use of Microsoft’s operating systems, Windows users may not be aware of a lesser known feature in the NTFS file system (New Technology File System) called Alternate Data Streams (ADS). This article aims to provide some essential information on ADS as well as its uses and drawbacks.

### **What are Alternate Data Streams?**

Since the introduction of Windows NT 3.1, NTFS has been the preferred file system developed by Microsoft for its NT-based operating systems. NTFS was implemented to replace the FAT (File Access Table) file system used by older operating systems like Windows 9x. Alternate Data Streams within NTFS allow the embedding of metadata in files or folders without altering their original functionality or content.

In NTFS, the main data stream refers to the standard content (if any) of the file or folder, and this is usually visible to the user, while alternate data streams are hidden. The Windows operating system does not provide the means for any Microsoft Windows Utilities to detect the presence of ADS, thus they are not visible to the vast majority of file management applications like Windows Explorer.

Alternate streams do not have any size limits and several streams can be linked to a normal file. The contents of ADS is not limited to text data; essentially any file that is in binary format can be embedded as an alternate stream.

Although Windows does not provide any native tools/functionality to read ADS, Microsoft do provide a very basic optional command-line tool to do so called STREAMS.EXE. There are also some third-party programmers who have created software to read ADS contents. Other programs like the commercial version of [HashOnClick](https://www.2brightsparks.com/onclick/index.html) from the suite of [OnClick Utilities](https://www.2brightsparks.com/onclick/index.html) developed by 2BrightSparks Pte. Ltd., has the function to read and calculate the hash values of alternate data streams in files.

### **The Purpose and Benefits of ADS and its Uses**

ADS was originally conceived so that Windows NT servers would be compatible with Apple Macintosh clients which use Hierarchical File System (HFS). The Macintosh file system stores data in two streams called the resource fork and the data fork. The data fork stores the actual data while the function of the resource fork is to store application metadata.

In order to be compatible with HFS, Windows introduced alternate data streams in NTFS to store additional information for folders and files. ADS can be used to store non-critical information associated with a file that is often accessed using the right-click command. Some additional uses of ADS are the ability to:

* Store keywords related to the file
* Provide summary information and/or data relating to the file
* Associate fonts or sounds with a file
* Associate document summary information with the file
* Provide an image thumbnail preview
* Allow the use of favicons that quickly help identify the identity of a website
* Provide icon types for Mac operating systems

Several applications also use ADS to store information. For example, in Internet Explorer, zone information called the Zone.Identifier is automatically added to downloaded files from the Internet. Internet Explorer also uses an alternate data stream to save website icons to web links that have been added as Favorites.

### **Back Up Your ADS!**

There are very few backup utilities that support the backup and restoration of ADS. Unfortunately, most programs do not recognize the hidden streams and simply back up the data file and folders. There are however three Windows backup and synchronization programs from 2BrightSparks that provide a solution. [SyncBackFree](https://www.2brightsparks.com/freeware/freeware-hub.html), [SyncBackSE](https://www.2brightsparks.com/syncback/sbse.html) and [SyncBackPro](https://www.2brightsparks.com/syncback/sbpro.html) all support the backing up of ADS provided the destination file system is based on NTFS.

### **A Word of Caution Concerning Alternative Data Streams**

As Windows Explorer does not list the ADS, it does not include the space used by alternate streams when calculating the free space available on the hard disk. The extra space used by ADS could (correctly) cause ‘disk full’ errors, but free space is spuriously declared available by Windows Explorer when checking, potentially causing much confusion, suspicion of disk errors, etc.

Further problems can arise if critical data is stored using alternate streams because only the main data stream is preserved when a file is copied to a FAT-formatted USB drive, sent as an email attachment, or is uploaded to a website via FTP or the cloud.

As alternate data streams are hidden, hackers like to exploit ADS by embedding viruses in them for malicious purposes. Viruses like the W2K.Stream employed ADS to infect and spread amongst Windows NT systems. As malware incidents increase, exploited uses of ADS for malicious intentions will likely increase. Unfortunately there are few antivirus programs that scan and detect contents stored within ADS.

### **Conclusion**

Although ADS was originally introduced to provide compatibility with HFS in Macintosh, it provides Windows users with a variety of discrete benefits. System administrators and users of NT based operating systems do however need to be mindful about its concealed nature, and therefore its potential risks.

## Description

The NTFS file system includes support for alternate data streams. This is not a well known feature and was included, primarily, to provide compatibility with files in the Macintosh file system. Alternate data streams allow files to contain more than one stream of data. Every file has at least one data stream. In Windows, this default data stream is called :$DATA.

Windows Explorer doesn’t provide a way of seing what alternate data streams are in a file (or a way to remove them without deleting the file) but they can be created and accessed easily. Because they are difficult to find they are often used by hackers to hide files on machines that they’ve compromised (perhaps files for a rootkit). Executables in alternate data streams can be executed from the command line but they will not show up in Windows Explorer (or the Console). Reference Example 1 for information on creating and accessing alternate data streams.

Since the :$DATA alternate stream exists for every file it can be an alternate way to access any file. Reference Example 2 for information on accessing the :$DATA alternate data stream in a text file. Any application that creates files or looks at or depends on the end of the file name (or the extension) should be aware of the possibility of these alternate data streams. If unsanitized user input is used to create or reference a file name an attacker could use the :$DATA stream to change the behavior of the software. A well-known vulnerability of this nature existed in older versions of IIS. When IIS saw a request for a file with an ASP extension it sent the ASP file to the application associated with the extension. This application would run the server-side code in the ASP file and generate the HTML response for the request. Due to a flaw in the extension parsing of these versions of IIS, filename.asp::$DATA did not match the extension and since there was no application registered for the asp::$DATA extension, the asp source code was returned to the attacker.

Proper user input sanitation is the best defense against this type of attack.

## Examples

### Example 1 - Creating Alternate Data Streams

* C:\> type C:\windows\system32\notepad.exe > c:\windows\system32\calc.exe:notepad.exe
* C:\> start c:\windows\system32\calc.exe:notepad.exe

### Example 2 - Accessing the :$DATA Alternate Data Stream

C:\> start c:\textfile.txt::$DATA

### Example 3 - Exploiting the ASP Alternate Data Stream Show Code Vulnerability

Normal access:

* http://www.alternate-data-streams.com/default.asp

Show code bypass accessing the :$DATA alternate data stream:

* http://www.alternate-data-streams.com/default.asp::$DATA

In the vulnerable versions, IIS parsed the extension of this file as asp::$DATA, not ASP. As such the application associated with the ASP extension was not invoked and the ASP source code was viewable by the attacker.

**Practical Guide to Alternative Data Streams in NTFS**

        Alternative Data Stream support was added to NTFS (Windows NT, Windows 2000 and Windows XP) to help support Macintosh Hierarchical File System (HFS) which uses resource forks to store icons and other information for a file. While this is the intended use (as well as a few Windows internal functions) there or other uses for Alternative Data Streams that should concern system administrators and security professionals. Using Alternative Data Streams a user can easily hide files that can go undetected unless closely inspection. This tutorial will give basic information on how to manipulate and detect Alternative Data Streams.  
  
        (Note about conventions: Alternative Data Streams are also sometimes referred to as Alternate Data Streams or ADS. Since Alternative Data Streams is so long, an ADS can be confused with Active Directory Services I will simple call this feature AltDS for short.)  
  
**Creating an AltDS**  
  
        Making an AltDS is fairly simple. I will use command line examples, feel free to follow along. We could hide some data in an AltDS behind an already existing file, but for this example we will create a new base file to hide behind:  
  
  
C:\>echo Just a plan text file>sample.txt  
  
C:\>type sample.txt  
Just a plan text file  
  
C:\>  
  
  
        Next we will use a colon as the operator to tell our commands to create or use an AltDS:  
  
  
C:\>echo You can't see me>sample.txt:secret.txt  
  
  
        Unfortunately, the use of the colon operator is a bit hit or miss in its' implementation and some times does not work as we might expect as seen below:  
  
  
C:\>type sample.txt:secret.txt  
The filename, directory name, or volume label syntax is incorrect.  
  
  
        Since the "type" command does not understand the colon operator we will have to use notepad to read the file:  
  
  
C:\>notepad sample.txt:secret.txt  
  
  
        If all worked well, you should not see a notepad window with the text "You can't see me" in it. Also notice that while the amount of total hard drive space free went down the file size of sample.txt did not increase:  
  
  
C:\>dir sample.txt  
Volume in drive C has no label.  
Volume Serial Number is 40CC-B506  
  
Directory of C:\  
  
09/27/2004 01:58 PM 23 sample.txt  
1 File(s) 23 bytes  
0 Dir(s) 12,658,040,832 bytes free  
  
C:\>  
  
  
        You can make an AltDS in not only files, but also directories, here is a quick example:  
  
  
C:\>md stuff  
  
C:\>cd stuff  
  
C:\stuff>echo Hide stuff in stuff>:hide.txt  
  
C:\stuff>dir  
Volume in drive C has no label.  
Volume Serial Number is 40CC-B506  
  
Directory of C:\stuff  
  
09/28/2004 10:19 AM <DIR> .  
09/28/2004 10:19 AM <DIR> ..  
0 File(s) 0 bytes  
2 Dir(s) 12,253,208,576 bytes free  
  
C:\stuff>notepad :hide.txt  
  
  
        Hopefully you now see a notepad window with hide.txt's contents. If all one could do with AltDS was hide text files it would not be that impressive, but there's much more that can be done with this useful NTFS feature.  
  
**Hiding and running an executable.**  
  
        As it turns out, using AltDS to hide executables is not much harder than it is to hide text files. AltDS makes for a great way for malware to hide itself on a system. Here's an example of how and executable can be hidden behind another file:  
  
        First we make our file to hide behind:  
  
  
C:\WINDOWS>echo Test>test.txt  
  
  
        Next we put an EXE behind is, I'm just using notepad.exe because it's convenient:  
  
  
C:\WINDOWS>type notepad.exe>test.txt:note.exe  
  
  
        Next we confirm the contents of the text file when some one tries to open it.  
  
  
C:\WINDOWS>type test.txt  
Test  
  
  
        Now we will confirm the file size, notice that adding notepad.exe as a steam did not increase the size of test.txt.  
  
  
C:\WINDOWS>dir test.txt  
Volume in drive C has no label.  
Volume Serial Number is 007E-2E3C  
  
Directory of C:\WINDOWS  
  
09/19/2004 08:37 AM 6 test.txt  
1 File(s) 6 bytes  
0 Dir(s) 19,734,708,224 bytes free  
  
  
        Now we will attempt to run our hidden exe. Notice the ".\" in front of the file name, this is necessary because the "start" command needs to know the correct path to the file (at least if you are using XP).  
  
  
C:\WINDOWS>start .\test.txt:note.exe  
  
C:\WINDOWS>  
  
  
        If all worked well there should now be a notepad window up on your system. You should be able to hide just about any other EXE file this way if you wish.  
  
**IIS and Alternative Data Streams**  
  
        While I was a Taco Bell recently I was thinking about what I could do in this tutorial that would make it a little different from other essays on Alternative Data Streams, which brought to mind a question: Does Microsoft's Internet Information Server work with Alternative Data Streams? As it happens, it does (at least IIS 6 at the time of this writing). I did the following commands to test it out:  
  
  
  
W:\>echo the text file>t.txt  
  
W:\>type xx.php >t.txt:x.php  
  
  
xx.php being an php file with the following code:  
  
  
<HTML>  
<BODY>  
<PRE>  
<?  
echo "If I see this I know it worked"  
?>  
  
</BODY>  
</HTML>  
  
  
        If you are following along try and see if you can read the text file off your server. If you don't have an IIS server just look at the file on my student site (assuming the link is not dead by the time you read this tutorial).

http://testiis/t.txt

        That worked as expected. Next try to see if IIS parses out the colon and then interprets the PHP file in the Alternative Data Stream:  
  
http://testiis/t.txt:x.php

        What do you know, it works! I could think of a few things that users might want hide it a web pages like this (moves, porn, scripts and such).  
  
**Hiding Videos**  
  
        One can also hide videos in Alternative Data Streams, but depending on how you put them in the streams they can be hard to play. For my examples I will use a video with spaces in the name just to complicate maters and show that it can be done. First let us create an AltDS behind the sample.txt file we made earlier, notice the use of quotation marks to compensate for the spaces in the file names:  
  
  
C:\>type "Naughty Linux Girls.avi" >"sample.txt:Naughty Linux Women.avi"  
  
  
        Give the above command some time, subjectively it seems that AltDS is a little slow to work its' magic. After the above command finishes we will attempt to open the video stream, I'll use Windows Media Player in my example since most of you should have it on your box and I know it works with AltDS:  
  
  
C:\>"C:\Program Files\Windows Media Player\wmplayer.exe" "c:\sample.txt:Naughty Linux Women.avi"  
  
  
        If all worked well the video should not be playing. Pretty cool huh?  
  
**Retreiving a file form an Alternative Data Stream**  
  
        Since support for AltDS is hit and miss in Windows we need to get an extra tool to suck data back out of an AltDS and put it back in a regular file. For this we will use a Windows port of the \*nix tool "cat" (download it and other tools from http://unxutils.sourceforge.net/ ). Retrieving the original file is simple:  
  
  
C:\>cat "sample.txt:Naughty Linux Women.avi">"Naughty Linux Women.avi"  
  
  
        If all works well "Naughty Linux Women.avi" should contain all the original data of the video we put into the stream.  
  
**Finding AltDS**  
  
        Some anti-malware tools understand how to search Alternate Data Streams for malware. I know Adaware SE Buld 1.05 can recognize know spyware in AltDSes (See http://www.lavasoftsupport.com/index.php?showtopic=40692 for more details). I've had a devil of a time finding out if other tools like Spybot or Symantec Antivirus look at AltDS, the vendors website give little information on it. If you know email me and I will update this tutorial.  
  
        Now I will show you a few tools you can use to find Alternate Data Streams. First there's LADS by Frank Heyne (see the tools section at the bottom of this tutorial for where to download LADS and other apps). LADS seems to work quite well for finding the streams we created above:  
  
  
C:\>lads c:\  
  
LADS - Freeware version 4.00  
(C) Copyright 1998-2004 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:\  
  
size ADS in file  
---------- ---------------------------------  
Error 32 opening C:\pagefile.sys  
368146432 C:\sample.txt:Naughty Linux Women.avi  
18 C:\sample.txt:secret.txt  
21 C:\stuff\:hide.txt  
  
The following summary might be incorrect because there was at least one error!  
  
368146471 bytes in 3 ADS listed  
  
C:\>  
  
  
        As you can see it found both of the files I hid. To find out more about what LADS can run it with the "/?" parameter:  
  
  
C:\>lads /?  
  
LADS - Freeware version 4.00  
(C) Copyright 1998-2004 Frank Heyne Software (http://www.heysoft.de)  
This program lists files with alternate data streams (ADS)  
Use LADS on your own risk!  
  
Usage: LADS [Directory] [/S] [/D] [/A] [/Xname]  
Directory: directory to scan, current if ommitted  
/S include Subdirectories  
/D Debug LADS ;-)  
/V Verbose error reports  
/A give a summary of All bytes used in the scanned directories  
(All files and directories are considered as uncompressed  
and all security decriptions are skipped  
for calculating this number!)  
/Xname eXclude any ADS "name"  
/Pfile read Parameters from "file"  
  
C:\>  
  
  
        Pay special attention to the "/S" parameter, you can use it to search entire hard drives and directory structures for AltDS.  
  
        Another popular tool is Streams from SysInternals. It works in much the same way as LADS:  
  
  
C:\>streams -s c:\  
  
Streams v1.5 - Enumerate alternate NTFS data streams  
Copyright (C) 1999-2003 Mark Russinovich  
Sysinternals - www.sysinternals.com  
  
Failed to open c:\\pagefile.sys:  
The process cannot access the file because it is being used by another process  
  
c:\\sample.txt:  
:Naughty Linux Women.avi:$DATA 368146432  
:secret.txt:$DATA 18  
c:\\stuff:  
:hide.txt:$DATA 21  
...  
  
  
        I had to truncate the results above to save space, since Streams is doing a search of the whole C: drive. Streams also has the parameter "-d" to delete streams, but I don't recommend that you use it unless you are sure of what you are doing.  
  
        For you folks that like to stick to GUIs there are three tools you might want to check out: ADS Spy which is quite slick, Crucial ADS which is also nice and ADS Detector which acts kind of like a plug-in for Explorer that lets you see Alternate Data Streams (unfortunately you have to sign up to download it, **Update:** to get ADS Detector to work after you have install it open your Explorer window then go to View->Explorer Bar and check "ADS detector". Thanks to David Bishop of Anderson Engineering for pointing out to me how to get ADS Detector to work). You can find links to all these tools at the bottom of this tutorial.  
  
**Quick answers and Factoids about AltDS**  
  
How do I delete AltDSes from a file?

        Well, if you delete the file it's attached to you will delete the AltDS, but I'm imaging you want to leave the base file intact. You can use a tool like Streams (see above) to delete the Alternative Data Streams, or you can rename the file and then use the "type" command to pipe it back to the original file name. Example:  
  
C:\>ren sample.txt delme.txt  
  
C:\>type delme.txt>sample.txt  
  
C:\>del delme.txt  
  
        A third option is to just move the files you want to remove streams from to a Fat32 drive then move them back to the original drive. You will get a windows that pops up that asks you to confirm stream lost, just click yes.  
  
I see a stream called "AFP\_AfpInfo" on a lot of my file, should I worry?  
  
        Most likely it's ok, this is usually an Apple file system fork like I mentioned at the beginning of this tutorial. AFP stands for Apple Filing Protocol, this stream should contain information like the icon a Mac would use to show the file. The AFP\_AfpInfo stream may have been put there when the file was touched by a Macintosh or if the Windows box it was copied from had Services For Macintosh enabled. It's possible that a deviant user could name one of their streams AFP\_AfpInfo to try and hide it, but it's not likely because using this name could make in not work as expected when they try to run or open it.  
  
I see a stream called "encryptable" on my Thumb.db files, should I worry?  
  
        This is expected behavior for Window, Thum.db hold thumbnails for folders when you choose the thumbnail view in explorer. It's ok, but if the size of the encryptable stream is over 0 bytes you might want to take a look at it. Same warnings as AFP\_AfpInfo apply.  
  
  
Ok, what about streams called "SummaryInformation", should I worry?  
  
        This is also expected behavior for Window. Windows sometimes stores text information like titles, keywords, and revision numbers here. Same warnings as AFP\_AfpInfo apply.  
  
Do streams survive being copied across the network or from one hard drive to another?  
  
        Yes, as long as both file system are NTFS. If the destination is Fat32 the steams will be lost.

        I hope this tutorial helped you to better understand Alternative Data Streams. Please feel free to email me if you have questions, clarifications or more information.  
  
  
**Further reading:**  
  
The best write up I've seen on AltDS:  
<http://patriot.net/~carvdawg/docs/dark_side.html>   
  
Great FAQ mby the creator of LADS:  
<http://www.heysoft.de/nt/ntfs-ads.htm>   
  
Good information on AltDS from wr0ck:  
<http://www.ramsecurity.us/texts/ntfsds.php>   
  
  
**Tools:**  
  
LADS - List Alternate Data Streams by Frank Heyne  
<http://www.heysoft.de/Frames/f_sw_la_en.htm>  
  
Streams.exe from SysInternals:  
<http://www.sysinternals.com/ntw2k/source/misc.shtml#streams>  
  
ScanADS command line tool:  
<http://www.kodeit.org/products/scanads/default.htm>   
  
ADS Spy GUI Scanner:  
<http://www.spywareinfo.com/~merijn/downloads.html>   
  
Crucial ADS GUI Scanner:  
[www.crucialsecurity.com/downloads.html](http://www.crucialsecurity.com/downloads.html)   
  
ADS Detector for Explorer:  
<http://www.codeproject.com/csharp/CsADSDetectorArticle.asp>   
  
Windows ports of Unix tools like CAT:  
<http://unxutils.sourceforge.net/> 