

# Data Hiding: Accessing Alternate Data Streams

Marc Zuze<sup>[0000-0002-9211-0025]</sup>

<sup>1</sup> University of Johannesburg Auckland Park, Johannesburg 2092, ZA  
201477488@student.uj.ac.za

## 1 Introduction

The digital world is an ever-developing scene, where new technologies are being released at a rapid pace, making people's lives easier through the digitizing and advancements of daily tasks such as communication, setting reminders, and automating tasks. With such good technological advancements there is always a bad side to things. Crime has now moved from the physical space to the digital space. The digital world has now made it easier for criminals to conduct malicious activities, whether it is to conduct a new crime within the digital realm or something that could allow them to conduct a crime within the physical space. The internet has given criminals ways to evade getting caught by law enforcements agencies due to the vastly open space that has new criminal activities popping up that have not been discovered and prohibited – it is really difficult for investigators to keep up with the ever emerging digital scene and find new ways to prevent crimes and capture cyber criminals, because the only way they can prevent a crime is by updating their tools and investigative techniques once a crime has happened, in order for them to learn from it.

Much of the crimes committed within the digital world would either have some type of digital, physical or even both types of evidence left behind, and this is how investigators tend to track down criminals. When it comes to digital evidence being seized by computer forensics investigators, it is stored in a type of file system. A file system is the way in which files are named and logically placed for storage and retrieval [1]. Most of the time accessing the evidence involves the recovery of information and this is often tedious and requires a lot of attention to detail [2]. There are many ways in which criminals and malicious computer users cover their tracks such as deleting data, which could be used as evidence. Another tactic that is used among them is to hide their data, where data hiding is the act of storing information in such a way that it prevents any individuals from being aware of the existence of the content [2].

There are several techniques that allow users to hide their information from other users and there are several reasons that data is hidden by users and these motivations vary from each individual. However, it becomes a must to recover and detect all hidden data when it is hidden to cover up any criminal activities. And therefore, as a

digital investigator, it is crucial to understand the different methods of hiding data in order to prosecute any wrongdoers and not miss out on any important evidence. The purpose of this paper is to review alternate data streams, a data hiding method, and explain how they work by conducting an analysis on the stream and explaining how to detect it within a system. The layout of this paper is as follows: First, a problem background will give a brief explanation of what data streams are and how they came about. Second, a literature review that will explain how data streams work. Third, there will be a description of how I intend on accessing the alternate data streams. Fourth, a detailed explanation of how I will implement the solution will be given. Fifth, a description of how my solution will be tested will be explained as well as the results and a discussion of these results. Lastly, I will conclude with a brief summary of my results and findings.

## **2 Problem Background**

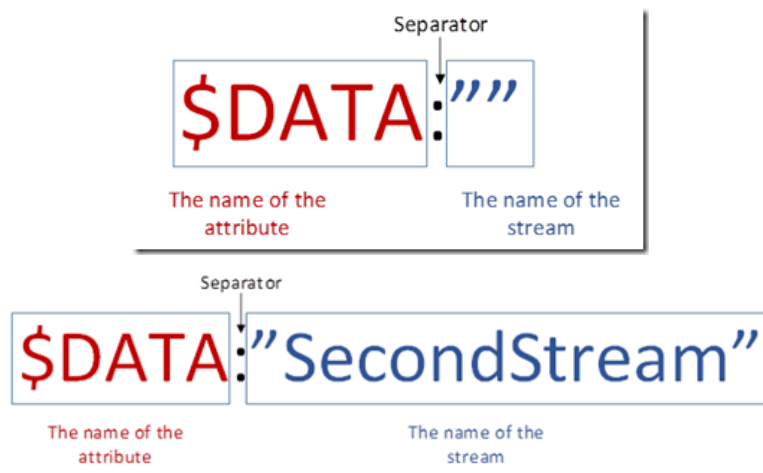
Windows decided to develop a file system after FAT, New Technology File System (NTFS), which would be used by all Windows versions, starting from Windows NT onwards. This file system included a feature which is practically undocumented and unknown to most administrators, developers and users [3] and is named Alternate Data Streams (ADS). What ADS can do is hide data by using a stream, by linking it to a normal file or directory. The size of the stream is not limited where a file can have multiple streams linked to them and this in turn makes it easy for any criminal or malicious user to store and hide their information. The initial purpose of ADS was to allow NTFS to have greater compatibility with Macintosh's Hierarchical File System (HFS). The Macintosh's file system works by using both data and resource forks to store its contents [3], where the fork holds all the file contents and the resource fork is used to identify the file type and other valid information about the file – this is the equivalent to file extensions on Windows for specific programs.

Awareness of ADS has not been made apparent to the everyday average computer user, and is extremely low, especially when it is compared to other file hiding methods, such as the hidden file attribute and due to the low awareness rate of ADS, there are not many security measures available that are ADS-aware. This therefore can allow viruses to be hidden in plain sight by using ADS and its malicious payload will be undetected, when in a normal situation it's signature would have been detected. Alternate Data Streams, without affecting its size or functionality, can fork file data into any existing file or directory which means that all of the standard file browsing methods will not be able to detect them. Since streams are not even detectable by standard windows tools this is a serious concern, because this creates a breeding spot for incriminating files or viruses to hide within the file system. Windows does not even consider the space taken by streams which means that a virus could spread and continue writing to a stream, in turn filling up disk space and producing a denial of service attack, making it really hard to clean up. Streams are really easy to create, and this will be explained further in the next section; and this requires very little skills,

meaning anyone can do it, meaning that any hacker or cyber-criminal can hide their contents at ease and virus authors can easily take advantage of this feature. With that said, streams are much harder to delete, and since it's attached to another file, in order to delete it, you would have to delete the parent which could either be a file or directory, which is why I intend on solving this problem. In the next section I will be discussing how to create an alternate data stream and how to open them with command line.

### 3 Literature Review

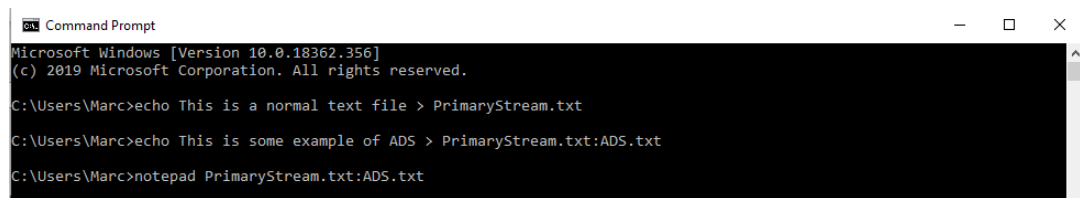
A file is divided up into "attributes" and the main attribute is the *\$DATA*. attribute and this is the part of the file where users put their data into. Whatever is added into the attribute will contain a stream of information that can be read and is called the *primary data stream* or normal data stream. An example of this would be "Example text in a stream", and when the data attribute is read, it would contain "Example text in a stream". Sometimes a primary data stream can be called an unnamed data stream, mostly because it has no name and is usually referred to as *\$Data:*" in the programming field [4], whereas named data stream would contain its name between the quotation marks. But since the example has no name it is unnamed. Streams are located within the Master File Table (MFT) like normal streams [2].



**Fig. 1.** Primary/unnamed data stream and secondary stream

Now that there is an understanding of unnamed data streams and where our data is stored, we can now discuss alternate streams. If a stream has a name, then it is considered to be alternate. These streams have a bad reputation since they have been used to write hidden data [4], where it was initially used to store general file information, it has been used for malicious intent. An example of an alternate stream would be *\$Data:*"AlternateDataStream" and unlike unnamed data streams, ADS are undetected,

when using native file browsing, but there are some tools available to help, other than accessing it through notepad and command line. One tool that worked was *STREAMS.EXE* which was provided by Microsoft which would give the user the available starting with the primary stream, then the alternate data streams and its size. The issue with *STREAMS.EXE* is that it does not allow the user to view what is in the ADS nor can it create data streams. When hashing a file with a stream, the stream is not included in the hash. Even when the alternate stream is removed, the hash stays unaffected, this can be seen in figure 4. Windows does not allow users to view these streams or even removing them without having to delete the file. This was until Windows introduced PowerShell which allows users to access multiple functionalities of the OS. The easiest way to create an ADS is to use the Windows command line and a user can start by typing the following command in figure 2, which will store the message Secret Message in the file “ADS.txt” which is appended to “PrimaryStream.txt”. To view the contents, you would then have to open it in notepad.



```
Microsoft Windows [Version 10.0.18362.356]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Marc>echo This is a normal text file > PrimaryStream.txt

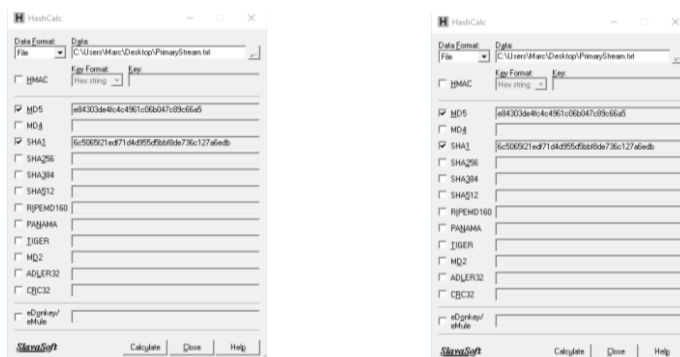
C:\Users\Marc>echo This is some example of ADS > PrimaryStream.txt:ADS.txt

C:\Users\Marc>notepad PrimaryStream.txt:ADS.txt
```

**Fig. 2.** An example of how to create an alternate data stream within a primary stream using Windows command prompt (cmd)



**Fig. 3.** The contents within the alternate data stream text file that was created in figure 2. Note that the file name is PrimaryStream.txt:ADS.txt



**Fig. 4.** Hash of a file with a primary stream on the left, and a hash of a file with an alternate stream included

Scripts inside of other languages, such as VB and Python, can then be used to execute files within a stream. Even using simple commands such as ‘start’ and ‘type’ can execute the files and when executed the program will appear in the task manager under the parent file’s name. For example, if the stream is “PrimaryStream.txt:ADS.exe” was running, Task Manager would only show that “PrimaryStream.txt” is running. This makes the process almost undetectable to Windows standard process viewers [3], where illegitimate processes such as viruses and any other malware could be hidden. In early 2000, two Czech virus writers developed the W2K.stream virus, which was the first known malware using ADS to carry and spread viruses; It was never released but was used as a proof-of-concept. This method has been used by many hackers to perform malicious tasks recently, and now viruses and malware are starting to take advantage of this file system hiding place. The next section will be an explanation on how I intend on detecting ADS and how to delete them safely – my model of solution.

## 4 Model of Solution

In this section, I will be explaining the plan of action, how to solve the problem of finding alternate data streams within a file. Since there is no way of viewing an alternate data stream using the standard Windows file viewer – Windows Explorer, there must be another way to solve this problem and view the files with ADS. I intend to solve this problem by creating a program that will allow a user to select files or a folder to view if it has an ADS, whether it is the primary stream or an alternate stream. The user will then be able to view the details of the file that they decide to select within the list, such as the file name, the streams available and the length of the file. The user will be able to see the available streams and the contents that are available within the it. Once the user has detected an unknown ADS within the selected file, they can either decide to remove the contents within the stream or either delete the stream without affecting the primary stream and the file as a whole. This, in turn, can allow users, and digital forensics investigators to view any hidden content within an

ADS. This solution will be implemented with Windows PowerShell, and my implementation will be discussed within the next section.

## 5 Implementation of Solution

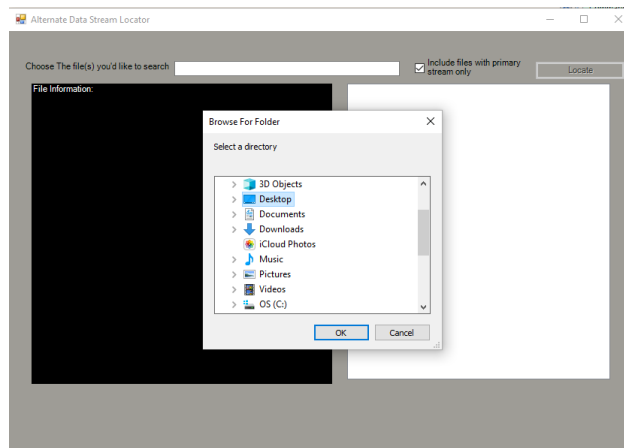
Microsoft introduced a new tool, PowerShell, which is a command-line shell and scripting language specifically made for task automation and configuration and was released with Windows 7 and later versions of the operating system. PowerShell uses the .NET classes in the form of cmdlets (command-lets) to perform particular operations. The cmdlets work by gaining access to specific storage systems such as file systems and registry. PowerShell has thus provided users a way to access to operations that the standard command line cannot such as viewing a file's stream information, from reading, deleting, clearing and writing to streams. This is how I intend on implementing my solution, by creating an interactive interface for the user to easily view and access the file streams. First, I intend on allowing the user to provide the file or folder or file they would like to search; With PowerShell I can implement this by using the File and Folder browser commands, which I then use the file path to search if there is more than one data stream within the selected file(s) – the user can then decide if they would like to view files with only the primary stream or just files with alternate streams.

Second, I will then allow the user to view the details of the file – the file path, the streams available, the file length and the contents within the file; The implementation of this is done by first checking whether the path is either from the file browser or folder browser, then it will loop through the file names and get all the relevant content and display the contents for the user to view. Third, I give the user the option to remove the contents within the streams or delete the streams entirely that are available within the selected file; The user can view this option within a dropdown list and will get prompted if they would like to continue with the option they have chosen, whether it is to remove the contents within a certain stream or remove the stream as a whole. The user can also choose to save the information given about the selected file or files they selected into another file for evidence. This is a solution that can help many users and investigators find any hidden information from files. A description of how I intend on testing this solution will be discussed further in the next section.

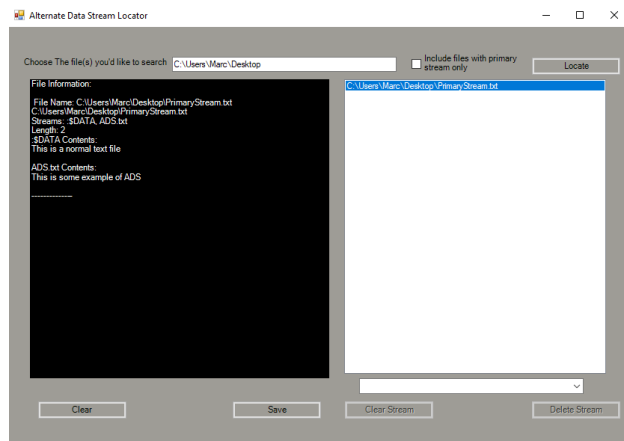
## 6 Description of How Solution will be Tested

The solution will be tested by detecting the alternate data stream that was created in figure 2. I will select all the files within the directory that the file is in, Desktop, and uncheck the option to view all files with the primary stream only. I will then look through all the available files to view their details. Once the files have been displayed within the list view, I will then select the file with the alleged stream and view the details of the file and the contents within each stream. We see that there are two streams – the primary stream, :DATA and the alternate data stream, :ADS.txt. I will then remove the contents within the stream to see if they will still be available upon

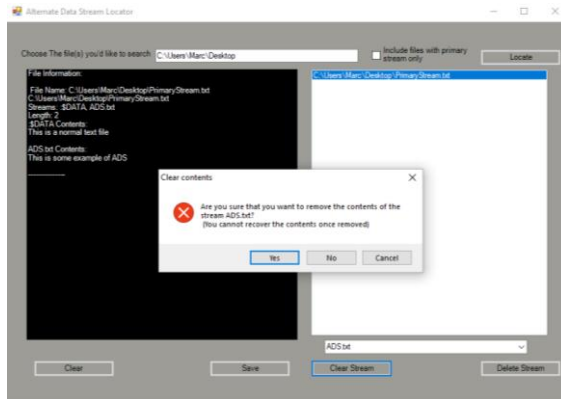
opening within notepad, then I will delete the :ADS.txt stream and reload the files within the directory once again to see if the file will be listed once again. I will then save the information displayed within the interface to a file as evidence of the transaction that happened.



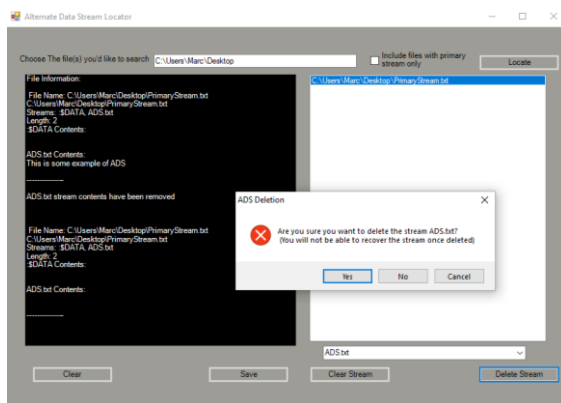
**Fig. 5.** An image of the folder browser when looking for files with an alternate data stream



**Fig. 6.** Locating only files with an alternate data stream. We see that the example file has been found. When selected, the file information is displayed with the file stream contents.

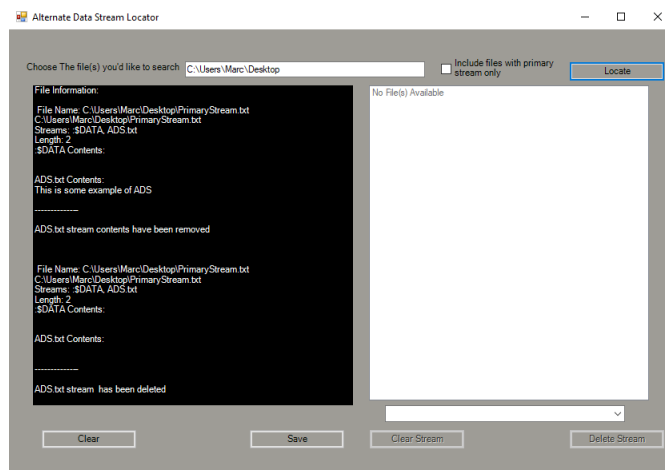


**Fig. 7.** Prompt when attempting to clear contents within a stream



**Fig. 8.** The contents within the alternate data stream has been removed. A prompt is shown when attempting to delete a stream completely.





**Fig. 9.** The alternate stream has been deleted and when reloading the search, we see that there are no more files with alternate streams available

## 7 Results and Discussion of these Results

From performing the test solution, I found that it is possible to find data streams within files, whether they are primary streams or alternate. The details of the file are there to help a user to view the contents available within the selected file and additional information, which can be saved and used later for evidence. But there are limited details available for streams, you cannot get any information about when the stream was created which would be important information for an investigator and user. It is evident that a stream's contents can be altered and removed, and a stream can be deleted. Removing the contents of a stream can help incase malicious content was added to a stream that was initially available within a file and deleting the stream is useful for when it was added by a malicious user. Giving the user the option to view files with just a primary stream or just files with an ADS helps reduce the time the user may have when searching for hidden content. The overall results found were accurate and good enough for me to find out what is within certain streams, as seen in figures 5 to 9, without having to use more than one program. It was quick and efficient when searching for streams.

## 8 Conclusion

From my initial problem statement regarding data hiding within files through the use of alternate data streams, I have found a way to create, view, save, remove contents, and delete streams by using Windows PowerShell. By being able to view contents within a stream, forensic investigators can now detect whether the contents can be used as evidence during an ongoing case. This can also help a user find any malicious software on their computer. An issue I have come across is displaying the date the

alternate data stream was created or modified as this could be crucial evidence for an investigator. For further study I would like to allow the user to modify or create an alternate data stream within the program, without them having to use the command line as well as allowing them to hash and duplicate the contents within the data stream to keep the integrity of the contents before working with it.

## 9 References

1. Mohsin, T.: Media & File System Forensics, <https://resources.infosecinstitute.com/category/computerforensics/introduction/areas-of-study/digital-forensics/media-file-system-forensics/>.
2. Lin, X.: Introductory Computer Forensics. 271, 276-278 (2018).
3. Broomfield, M.: NTFS Alternate DataStreams: focused hacking. Network Security, 7 (2006)
4. Alternate Data Streams in NTFS, <https://blogs.technet.microsoft.com/askcore/2013/03/24/alternate-data-streams-in-ntfs/>.
5. Berghel, H., Brajkovska, N.: Wading into alternate data streams. Communications of the ACM. 47, 21 (2004).