Data Hiding Techniques in Windows 10

Contents

[Introduction 3](#_Toc44923871)

[Steganography 3](#_Toc44923872)

[NTFS Alternate Data Streams 6](#_Toc44923873)

[Writing Data into the Master Boot Record or GUID Partition Table 7](#_Toc44923874)

[Hiding Files in the .zip File Archive Format 8](#_Toc44923875)

[Conclusions 8](#_Toc44923876)

[Works Cited 9](#_Toc44923877)

# Introduction

The purpose of this investigation is to cover several different data hiding techniques/methods that are viable within Windows 10. Data hiding is a broad term, often covering any method that an attacker may use to attempt to make their data harder to find or read to any unauthorized users. Under this definition, techniques such as encryption would be considered a data hiding procedure, as the key is required to make the information human readable. While this is a valid approach the discovery of a file obscured via encryption could be the first warning sign of a breach and could result in an early end to your breach of the target. For this reason, it is not enough to obfuscate your data using encryption. you must also attempt to hide obfuscated data from protection. These techniques also fall under the umbrella of “Data Hiding” and are the focus of this investigation.

# Steganography

Steganography is also a broad term. In her Wired article “Hacker Lexicon: What is Steganography?” Lily Hay Newman describes as “basically hiding bad things in good things.” While it is reductive to call all data hidden using steganography as “bad” the statement is otherwise true. Steganography is any technique used to hide one set of information within another. Steganography has been around for thousands of years, with documented uses of it occurring in the great empires of Greece, Egypt, and Rome (Siper). Perhaps the most primitive form of cryptography are simple substitution ciphers, in which some letters or symbols stand in for other letters or symbols. For example, the famous Caesar Cipher shifts each character 3 positions to the right, wrapping around from z to a if needed. In this system, the phrase “steganography” becomes “vwhjdqrjudskb”. This is essentially a very weak form of encryption, as it masks the literal content of the message but does not hide the fact that a message is trying to be sent. However more complicated forms of steganography can attempt to hide a message in plain sight. For example, consider the following message.

Allow me to introduce myself. The name is Steve Buscemi. There are a number of you who may be familiar with my work. All things considered; I am a famous actor. Careful consideration of the roles that I take is one of the reasons for my success. *Kansas City*, for example, was submitted to the Cannes film festival.

While on the surface this reads as an awkwardly written statement by famous actor Steve Buscemi a careful eye could spot the message “ATTACK”. This message was passed by using only the first letter of each sentence, another common steganography technique. Techniques like this are far more powerful, as they allow a message to be passed in plain sight, and an outside observer may not see anything suspicious about the message’s content.

Many of these techniques can be applied directly to the digital realm. Company secrets, user credentials, or lists of hostnames to target could all be hidden in text messages. However, perhaps a more effective and harder to detect method would be hiding data within other data. This is commonly done with images, as making minor alterations to the bits encoding JPEG images hardly alters the final appearance of the image. A common strategy to detect this type of steganography is to compare the altered photo against the original image in photo editing software. However, if the original photo is never released detection via that message proves nearly impossible. For example, the following two images will appear to be identical. However, the second image has had the least significant bits altered to contain a compressed text file of the entirety of Macbeth by Shakespeare.

A large green field with trees in the background

Description automatically generated

*Fig. 1. The unaltered image of a tree.*

*A large green field with trees in the background

Description automatically generated*

*Fig. 2. The image altered by least-significant bit steganography.*

This was accomplished by using the tool jsteg, which can be downloaded as a linux package or as a binary from github [here](https://github.com/lukechampine/jsteg). As noted in the github description the real limit of this technique is in file size. For each 1 byte of payload that you want to hide you need about 10-14 bytes in the image you will be hiding the payload in. For this reason, this form of stego is only effective for hiding text files, or small files of other data types. There are also methods to perform this type of hiding inside of videos by decompiling the video into its individual frames, treating each frame as an image, and then reassembling the video (Murali). This however massively inflates the size of the video file, likely due to the efficiencies of interframe compression being lost (Murali).

# NTFS Alternate Data Streams

The NTFS file format used by Windows supports having multiple data streams associated with a file. In NTFS when a file is created an unnamed, default data stream is created to store all the contents of that file. However, it is trivial to simply add new data streams to a single file. These streams will not appear when viewing the file through common methods (double clicking the file to open, opening using an associated app, etc.), but if someone knows the presence of them there are several tools built into windows that will allow you to detect their presence. While these streams do have some legitimate usage cases, like storing associated metadata alongside the file, they are easily used to hide sensitive or potentially malicious information.

In Windows 10 you can add alternative data streams to a file using the powershell syntax

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

Later these streams can be read using the command

More < “[filepath]:[stream name]”

It is also possible to create alternate data streams for directories within Windows. Because these streams are not listed as part of the file size in Windows they are likely to go undetected unless the user has software that scans for alternate data streams or manually checks a file.

This technique does have its limitations, however. Firstly, if detected the presence of an alternative data stream is incredibly suspicious. Furthermore, these data streams are immediately removed from a file if it is transferred to any filesystem that does not support alternative file streams. If the alternative file stream is being used to store malware, it limits the ability of that malware to spread in a mixed file system environment.

# Writing Data into the Master Boot Record or GUID Partition Table

The Master Boot Record (MBR) and GUID Partition Table (GPT) are the two most common methods for storing and managing partition tables for storage drives in modern operating systems. Both can still be used in Windows 10, and the techniques for storing data in these tables are similar.

The MBR and GPT are hidden partitions on disk and are not easily accessible if the user is booting off the disk in question. For this reason, hiding data in these locations would be more likely for an inside attacker or someone hiding files on their own personal computer. Either the MBR or the GPT can be accessed if the user can boot from an external drive by using a tool like DD to mount the desired location as a drive on the system. It can then be written to in the same way that you could write to any file system.

If done improperly this technique could result in permanent damage to the partition table, making it difficult to recover the data being stored on the drive.

# Hiding Files in the .zip File Archive Format

The .zip format is the most common file archive and compression format used today. It has support in multiple operating systems, and without the use of external programs is the only type of file archive that Windows is able to create and extract. Packaging malicious executables, documents, or other files within .zip archives used to be a common technique employed before real time scanning of compressed files was a common feature of most security software. There are also a handful of different techniques that can be used to hide files within a .zip archive from being displayed by any common user installed programs. These techniques require manipulating the .zip file structure manually using a hex editor.

Perhaps the best technique available can be done by making “a very small change to the header” (Hiding in the familiar). The signature structure of a .zip archive files contains a variable that stores where the location of the central directory of the .zip file begins. If you modify this value to point to a file other than the first file actually stored in the archive, all files stored previous to the item you are now pointing at will become invisible to most common tools (Hiding in the familiar). Using this technique, it would be possible to compress and send a number of files directly without raising suspicion. The file size of .zip files is often very mutable, so even a large .zip may not draw attention should it be transferred.

# Conclusions

There are several smart techniques that one can try to hide data in a modern Windows 10 system, and there are many other cases not documented here. Each of these various methods would make it much more difficult for an investigator to find any one of these files, while being relatively trivial for the intended user or target to retrieve assuming that they knew the method.

# Works Cited

Arntz, Pieter. “Introduction to Alternate Data Streams.” *Malwarebytes Labs*, 30 Mar. 2016, blog.malwarebytes.com/101/2015/07/introduction-to-alternate-data-streams/.

“Hiding in the Familiar: Steganography and Vulnerabilities in Popular Archives Formats.” Black Hat, cdn2.hubspot.net/hubfs/3375217/Reversing\_Labs\_November%202018/File/NyxEangine\_BlackHat-EU-10-Whitepaper.pdf.

Murali, Anand. “A Guide to Video Steganography Using Python.” *Medium*, Better Programming, 18 May 2020, medium.com/better-programming/a-guide-to-video-steganography-using-python-4f010b32a5b7.

Newman, Lily Hay. “What Is Steganography?” *Wired*, Conde Nast, 26 June 2017, www.wired.com/story/steganography-hacker-lexicon/.

Siper, Alan, et al. “The Rise of Steganography.” *Proceedings of Student/Faculty Research Day, CSIS, Pace University*, 6 May 2005.