Transmogrify Tool Analysis

Contents

[Introduction 3](#_Toc43149451)

[Installation 3](#_Toc43149452)

[Performing a manual Transmogrification 7](#_Toc43149453)

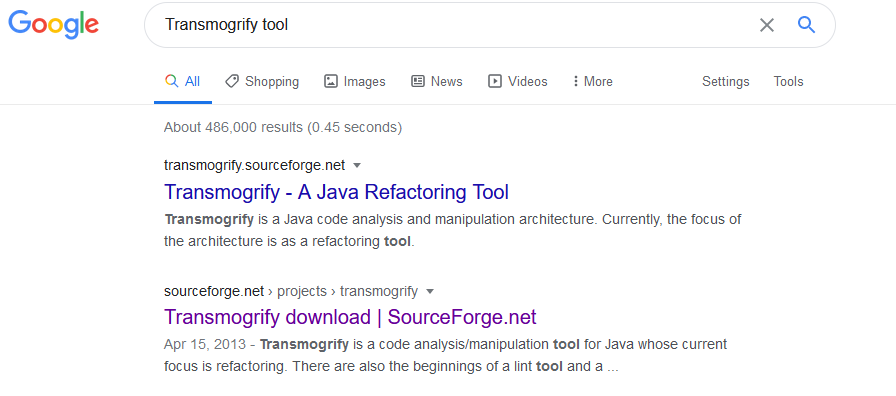
[Conclusions 10](#_Toc43149454)

# Introduction

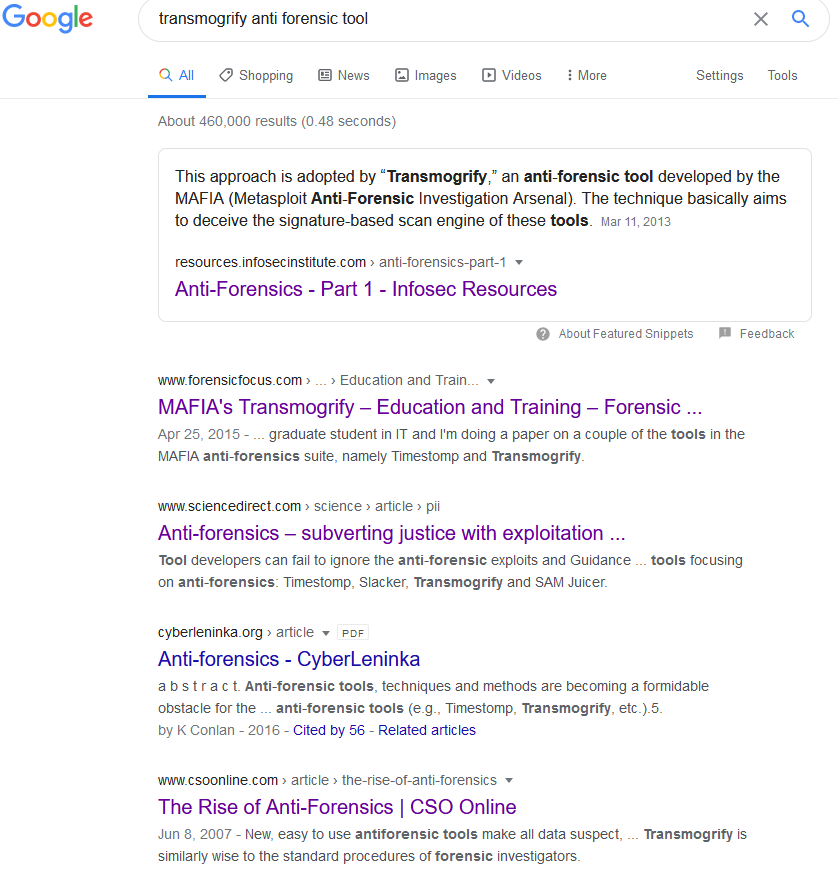
Transmogrify is a tool packaged as a part of MAFIA, the Metasploit Anti-Forensics Investigation Arsenal. It was released alongside other powerful tools like TimeStomp, Slacker, and SAM Juicer two of which I have analyzed as well. The intent of Transmogrify is to automatically find and alter the file signatures and trailers to the signature and trailer of another type of file. The file signature and trailer is used by operating systems such as Windows to determine where the file begins and ends on disk, as well as to determine the type of file. Altering these values can be useful if you are storing files within slack space, as the changed signatures will make automatic file scrapers more likely to pass over your file, or to make it seem as if the file failed to completely recover. However, should the attacker need the file they can simply run transmogrify again to replace the signatures to their original value, restoring the function of the file.

# Installation

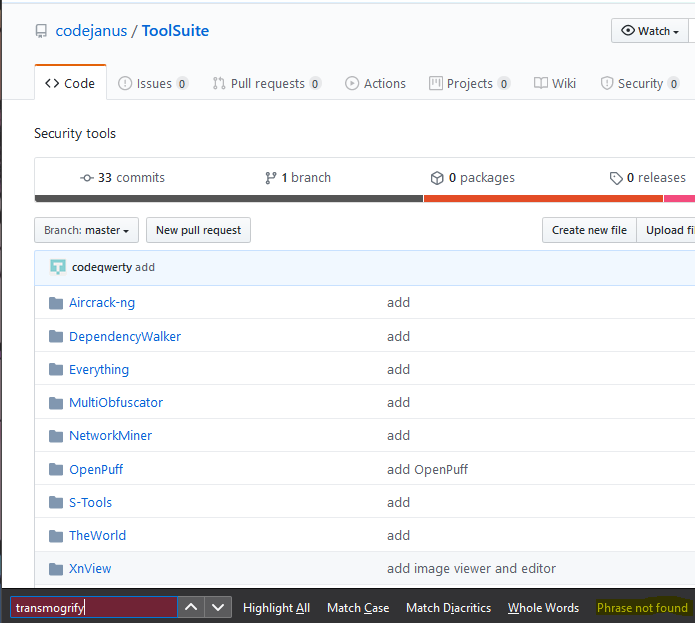
Normally this is where I would discuss the process of tool installation, describing the steps that I went through to find the tool in question. However, while I can find plenty of documented evidence that this tool was at one time released, any presence of it on the internet appears to have disappeared.

Searching for “Transmogrify Tool” on Google provides links to a tool with the same name that performs Java code analysis and refactoring, a completely different function than the file manipulation. 

*Fig. 1. A google search for Transmogrify tool*

Further refining this google search by including the phrase “Anti-Forensics provides a number of references to it in various web sources. From forum posts to published scientific journals, it is frequently mentioned as a member of the MAFIA suite of tools but there are no remaining active download links to it on any source.  


*Fig. 2. A google search for Transmogrify Anti Forensic tool*

With both of these leads having come up empty, I figured that I may be able to find the tool in the same github repository that I had found both Slacker and TimeStomp in, reasoning that if they hosed copies of those two tools the maintainer may have included a copy of Transmogrify as well. I returned to codejanus’ github repository, and searched it for any mention of the word Transmogrify and again found no results.

*Fig. 3. The result of searching codejanus’ ToolSuite repo for Transmogrify*

It was at this point that I found a forum post on forensicfocus.com, a site that describes itself as a forum for digital forensics and E-Discovery professionals. Here a user going by the name of AT87 reached out asking if any of the users there had been able to find the tool, but the responder could not find a source for this tool either. You can find this forum post by clicking through [here](https://www.forensicfocus.com/forums/education-and-training/mafias-transmogrify/).

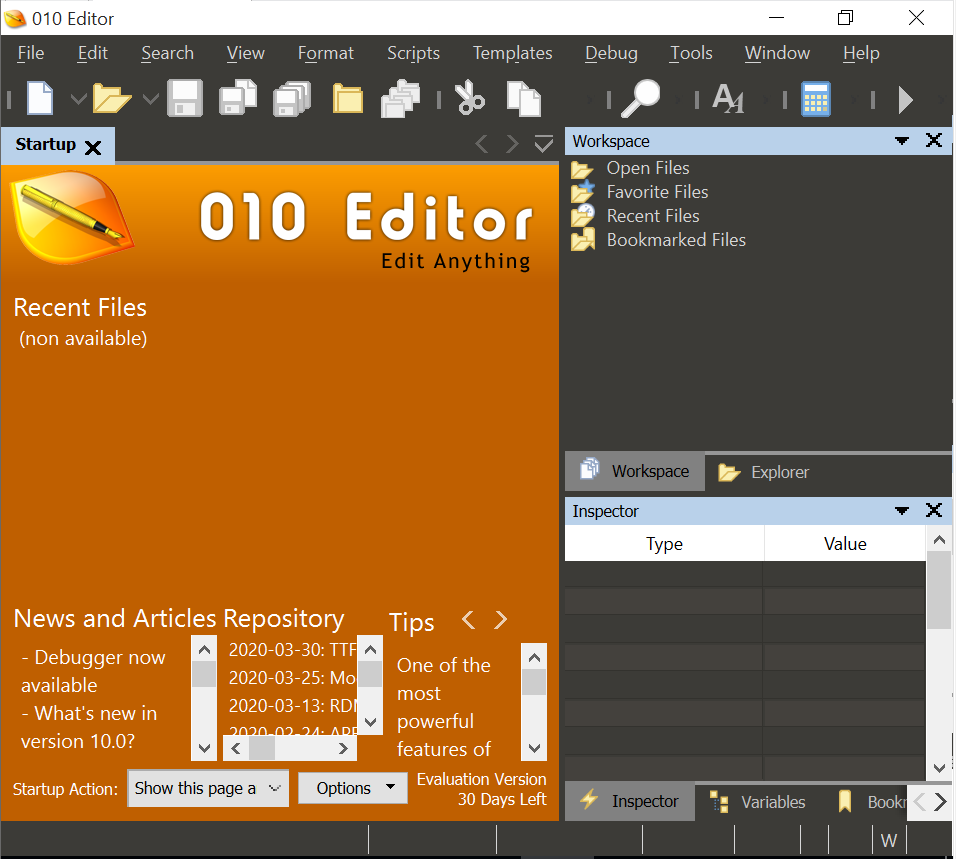
It is for these reasons that I believe that Transmogrify may have never existed as a tool, and rather exists as a technique documented by the Metasploit team. While the other tools in the MAFIA tool kit were eventually released and have at least some presence online, the fact that this tool only appears in research articles and not in any related repository or search would be strange had the tool actually been effective.

However, though the tool cannot be found, this is a fairly easy process to complete manually. To demonstrate this, I will be performing a manual “transmogrification” on an executable on my computer, disguising it as a .JPG file.

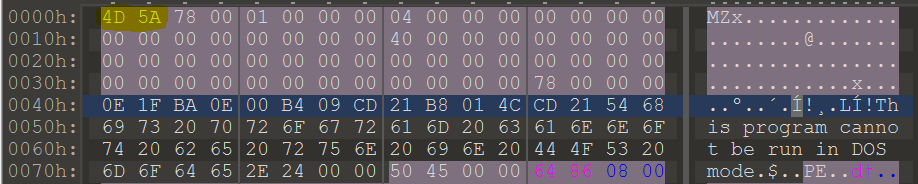
# Performing a Manual Transmogrification

To perform this attack manually we will need access to a program that allows us to directly edit the hex values of a given file. For this demonstration I will be using 010 editor, as it has a free 30-day trial, is available on all operating systems, and I am familiar with using it in my schoolwork. Using another utility such as HxD would work just as well though some of the basic steps for opening and editing a file may not match.

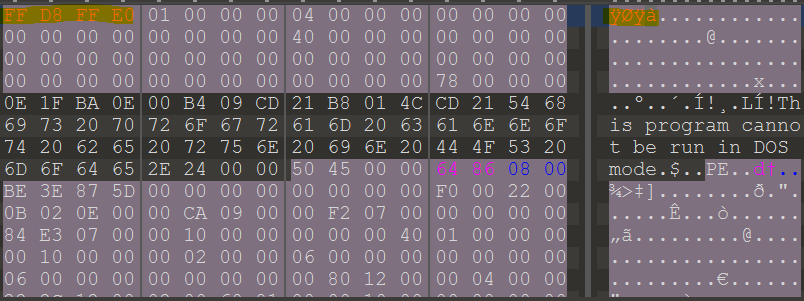
I began this process by downloading and installing 010 Editor, accepting all the defaults in the installation process. 010 Editor and it’s free trial can be downloaded for any platform by going to the address linked [here](https://www.sweetscape.com/download/010editor/). With the tool installed, all that was left to do was to download a target.exe to hide and change the file signature and trailer values to demonstrate a simple proof of concept for the technique. Once the program had completed its installation, I was greeted with this splash screen.

*Fig. 4. The appearance of 010 Editor following installation.*

I decided to use PuTTY as my test .exe file. PuTTY is a Windows application that can perform SSH and Telnet login sessions to allow for remote login to any system that supports those protocols. It would be a useful program to bring into an environment if you were looking to spread in the internal network, and is easy to download. PuTTY can be downloaded by clicking the link [here](https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html) and selecting the appropriate version. For this demonstration I will be using the 64-bit binary version of the program.

With putty.exe in my downloads folder and 010 Editor open I could simply open putty.exe by browsing to its location in the File > Open menu. When prompted, I installed a package that would automatically perform context highlighting for the Portable Executable file format. The first four hex characters of any standard PE file are 4D 5A or the characters “MZ”. To disguise the identity of this file, we would first have to remove this signature and replace it with the appropriate file signature of the JPEG file format. We will also need to add a trailer to this file as otherwise it will be obvious that it was modified in some way.

*Fig. 5. The PE file signature as it appears in 010 editor.*

As the JPEG’s format’s file signature is 8 hex characters long, we need to commit the sequence 4D 5A 78 00 to memory. These are the hex values that we are over writing, and they would need to be restored in order for the file to work properly. Once we have these values stored off, we simply replace them by typing in the desired characters into 010 editor.

*Fig. 6. The new file header.*

To add the trailer and complete the conversation, we simple need to scroll to the bottom of the hex, select the last character of hex that had been in the file, and append the JPEG standard’s file trailer.

*Fig. 7. The JPEG file trailer appended to putty.exe*

With these modifications complete, we can simply save the file again as “putty.jpg” using the File > Save menu. This will create a new file on disk containing the code of Putty that appears as a jpeg.

# Conclusions

While this technique can be useful for hiding programs in slack space, performing this task manually in a way that would be convincing to a forensic analyst would be tricky. AS you may have noticed in figure 6 parts of the binary still mention that “This file cannot be run in DOS mode” a warning that would never appear in a .jpg file and almost always appears in a PE file. This combined with the fact that just changing the file headers does not make the content of the file a valid .jpg means that it will be unable to open in windows. This technique is best reserved for one or two high importance files that may end up fractured across multiple file’s slack space or stored in a hard to find place within the file structure. It does make it harder for someone to determine what the purpose of the file is without specialized tool, but without a deep understanding of both the original file structure and the new file structure your results likely will not stand up to intense scrutiny.