WannaHusky Malware Analysis w/ YARA + TTPs

medium.com/@mars0x/wannahusky-malware-analysis-w-yara-ttps-2069fb479909

Mars April 5, 2022

Mars

Apr 5

.

6 min read

Hello! I have recently gained a new interest in threat intelligence, malware research and analysis. In response, I have been taking courses to learn the tools and methodologies of malware analysis and reverse engineering. This blog is a result of my new-found interest.

My first blog post follows my observations on a malware sample provided by as a part of his course, **Practical Malware Analysis & Triage**. The formatting of this post is more report-style, highlighting my findings in my initial triage. I've included a full set of YARA rules, as well as the TTPs of the sample at the end.

Please feel free to give feedback, as I would like to further improve my malware analysis skills and perform deeper and more thorough analysis in the future.

Executive Summary

Ransomware.WannaHusky is a Nim-compiled ransomware malware sample, provided as part of the Practical Malware Analysis & Triage course. It is an 32-bit executable.

Ransomware.Wannahusky consists of multiple steps: The sample first returns the current directory of the user and then returns the home directory of the user. If *cosmo.jpeg* is on the Desktop of the infected host, the sample executes a PowerShell script (*ps1.ps1*). Next, the sample will encrypt *cosmo.jpeg,* adding the *.WANNAHUSKY* extension and write the *WANNAHUSKY.png* to the Desktop. The sample will then change the Desktop wallpaper to the *WANNAHUSKY.png* file. Lastly, it runs the `tree C:\` command.

Indicators of compromise include *ps1.ps1*, the encryption of *cosmo.jpeg* (with the file becoming *cosmo.WANNAHUSKY*), a ransomware note saved as *WANNAHUSKY.png*, the Desktop wallpaper changing to the contents of *WANNAHUSKY.png* and the `tree C:\` command being run.

`3D35CEBCF40705C23124FDC4656A7F400A316B8E96F1F9E0C187E82A9D17DCA3`

Observations

Malware Composition:

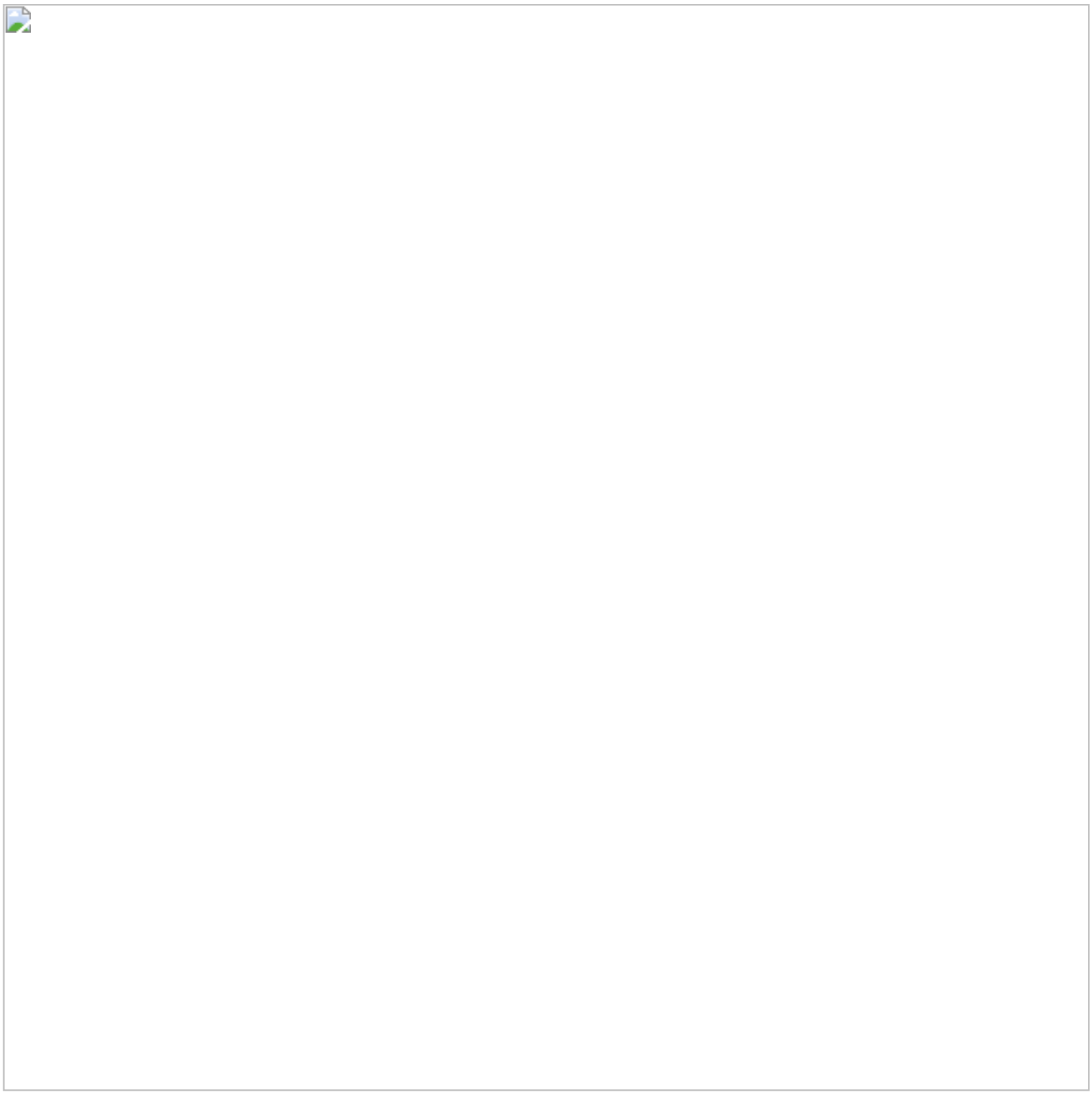
The sample provided consists of two components:

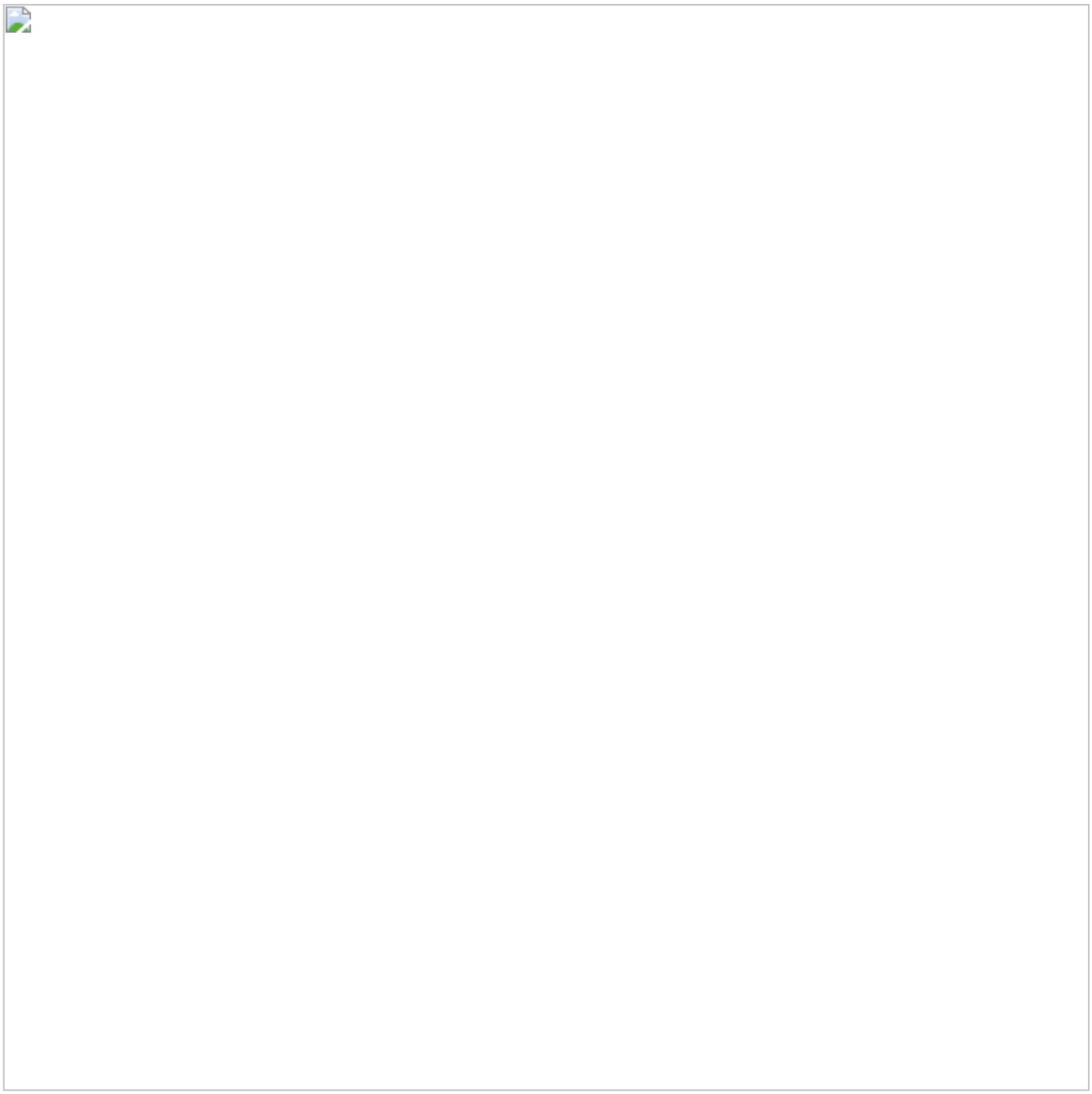
`Ransomware.WannaHusky.exe3D35CEBCF40705C23124FDC4656A7F400A316B8E96F1F9E0C187E82A9D17DCA`

Static Analysis:

CAPA:
CAPA is a tool that detects capabilities in executable files. After running the binary against the CAPA tool, the following information was extracted:

- Hashes: SHA1, SHA256 and MD5
- Language: The binary is compiled with Nim
- Encryption: The binary uses HC-128 to encrypt data
- Encoding: The binary uses Base64 to encode data
- There are detected capabilities of reading and writing files to the file system
- There are detected capabilities of creating and terminating processes





PE Studio:

After analyzing the Strings in PE Studio, some strings of interest are listed below:

- `tree C:\` command is run
- The script is written to the Desktop, and then executed
- The contents of the PowerShell script:

> The script imports the user32.dll. It then sets the `$currDir` variable to the contents of the `Get-Location` cmdlet, sets the `$wallpaper` variable to the WANNAHUSKY.png file, and sets the `$fullPath` variable to the joined paths of the current directory and WANNAHUSKY.png (ex. `Desktop\WANNAHUSKY.png`). It then invokes the `SetWallpaper` function to change the user's Desktop wallpaper to the ransom note (WANNAHUSKY.png).

is written on the Desktop

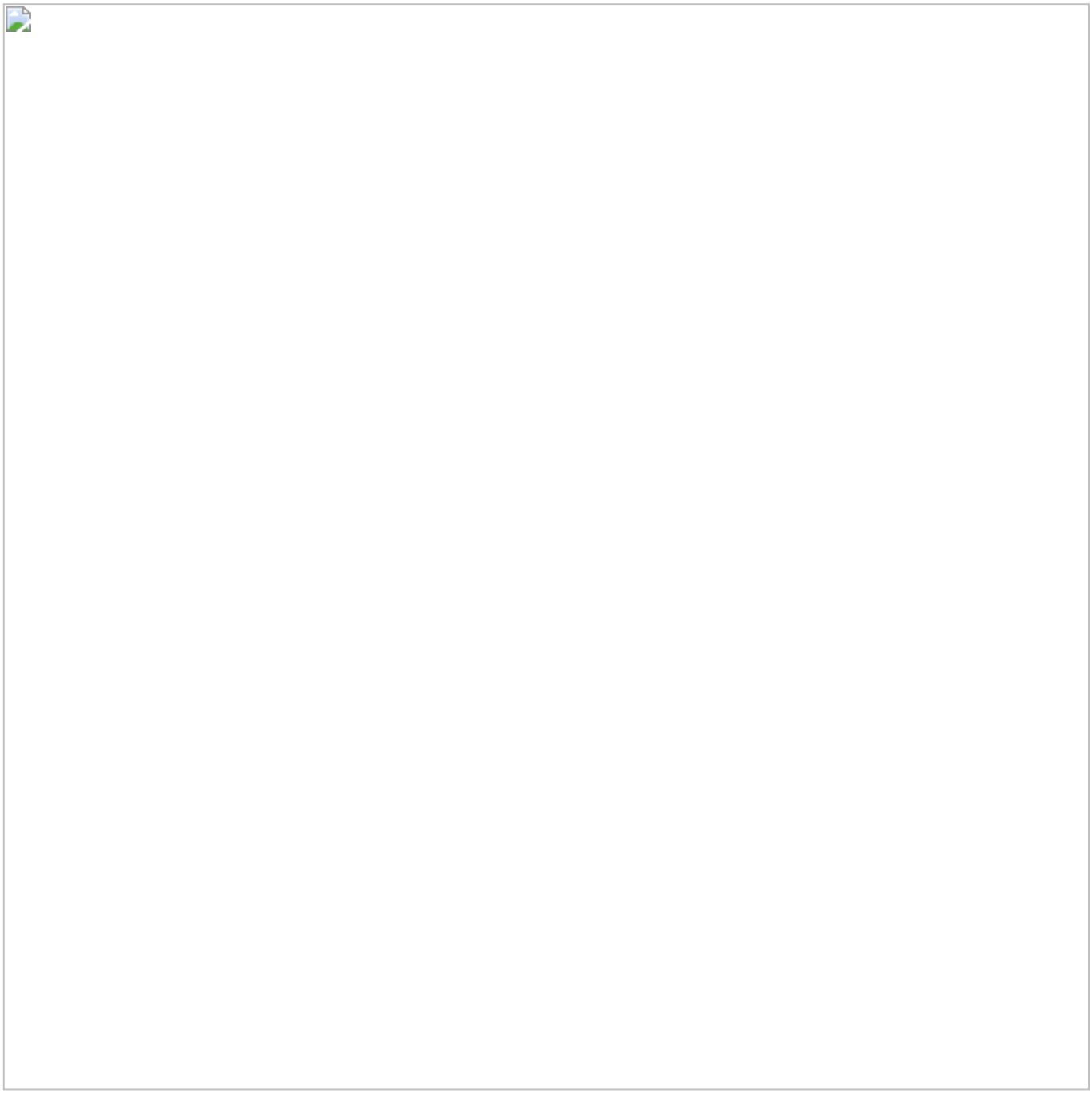


Figure 2: PE Studio — Strings

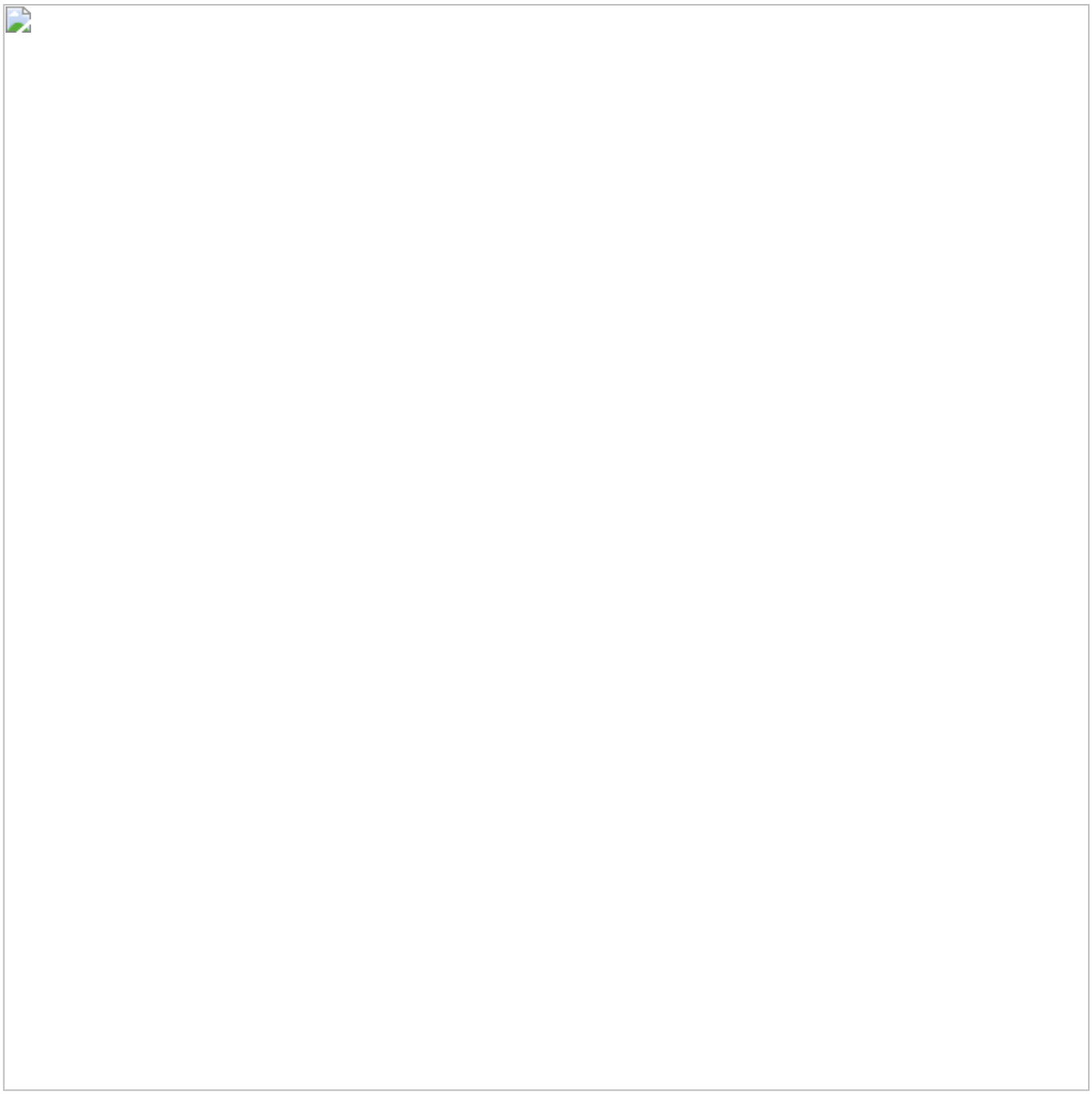


Figure 3: Contents of ps1.ps1

Dynamic Analysis:

Initial Detonation:

When the binary is run, *cosmo.jpeg* is encrypted and deleted. Two new files named *cosmo.WANNAHUSKY* and *WANNAHUSKY.png* are written to the Desktop. The wallpaper is changed to the contents of *WANNAHUSKY.png*.

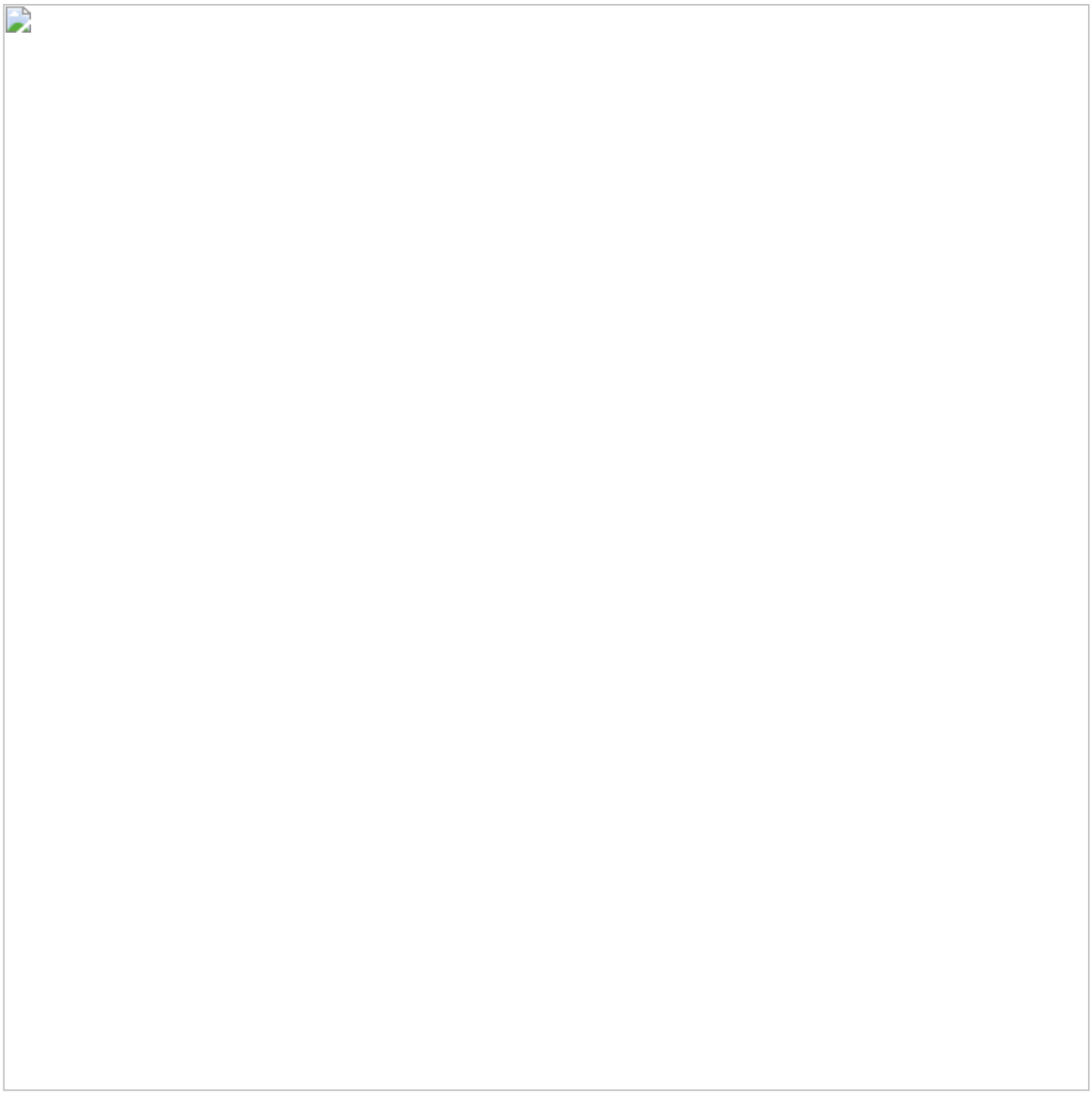


Figure 4: WANNAHUSKY.png

A *cmd.exe* window also appears on the screen, running the tree command. It appears that the *cmd.exe* window is not being run discreetly in the background and shows the output of the command before exiting.

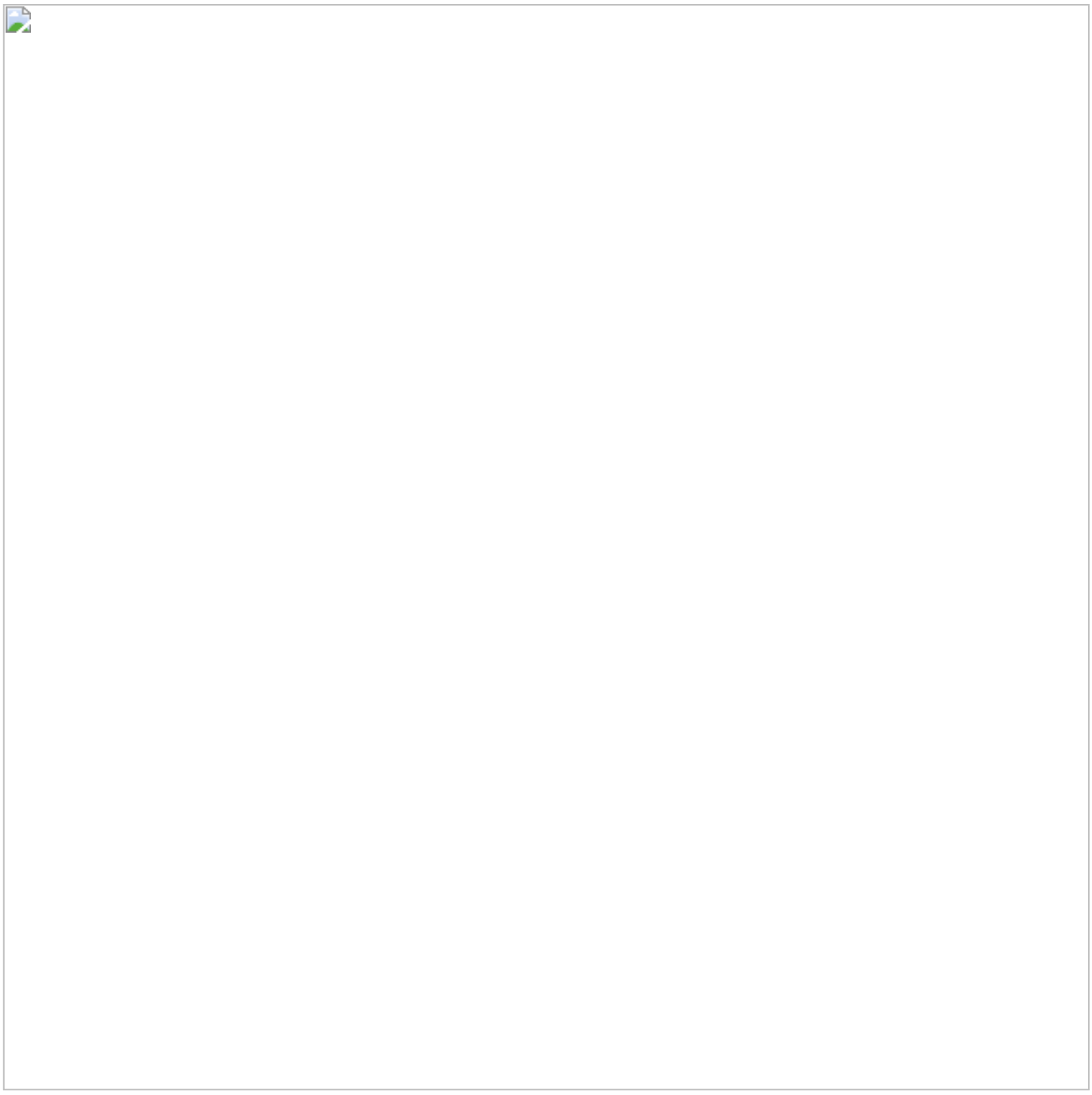
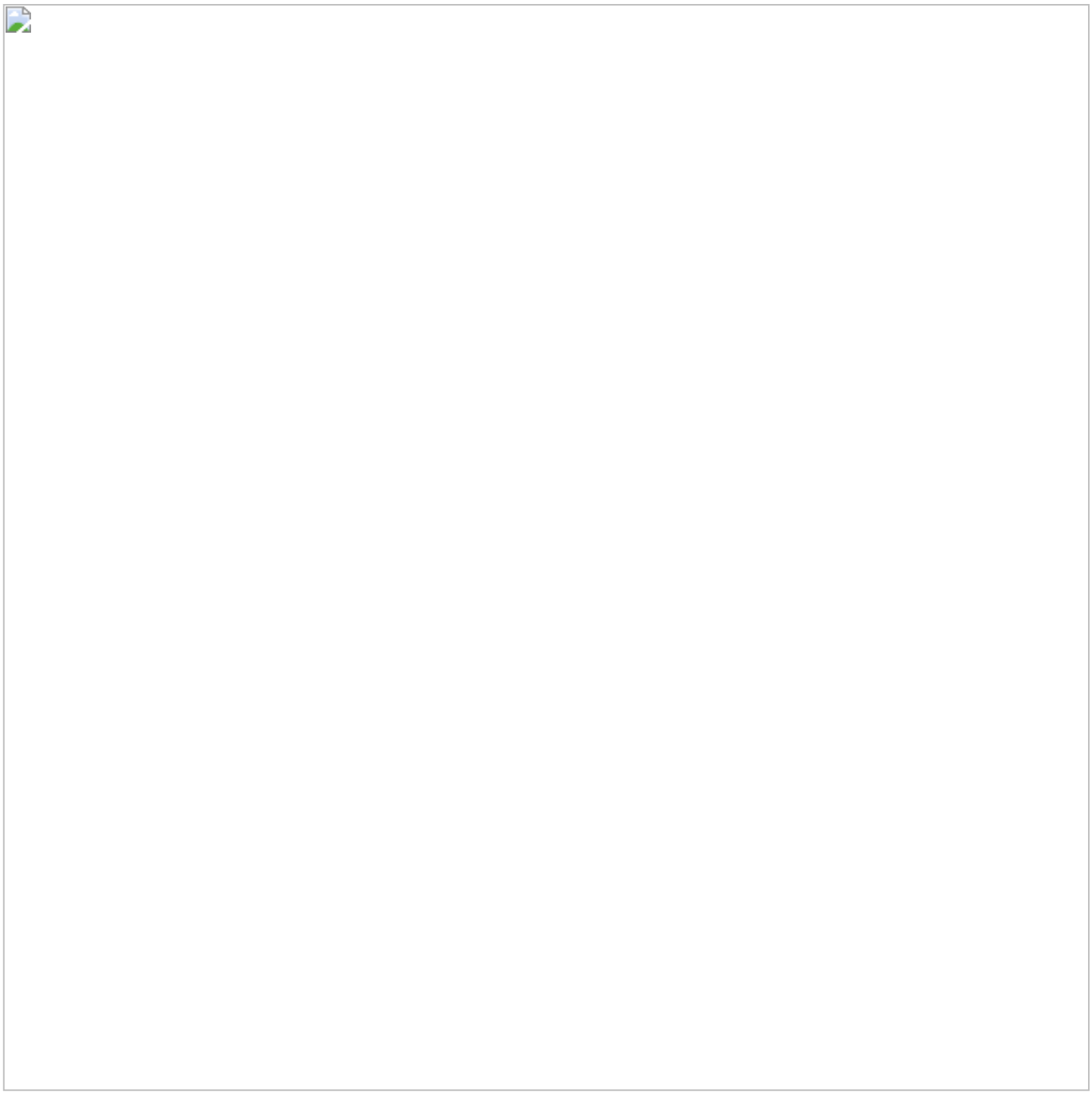


Figure 5: Capture of the tree command output

After monitoring for network traffic in Wireshark, there was no detection of the binary attempting to call out to hosts or domains.

Additionally, in order for the binary to execute successfully, *cosmo.jpeg* needs to be on the Desktop. If the binary cannot locate *cosmo.jpeg*, it returns an error (shown below).



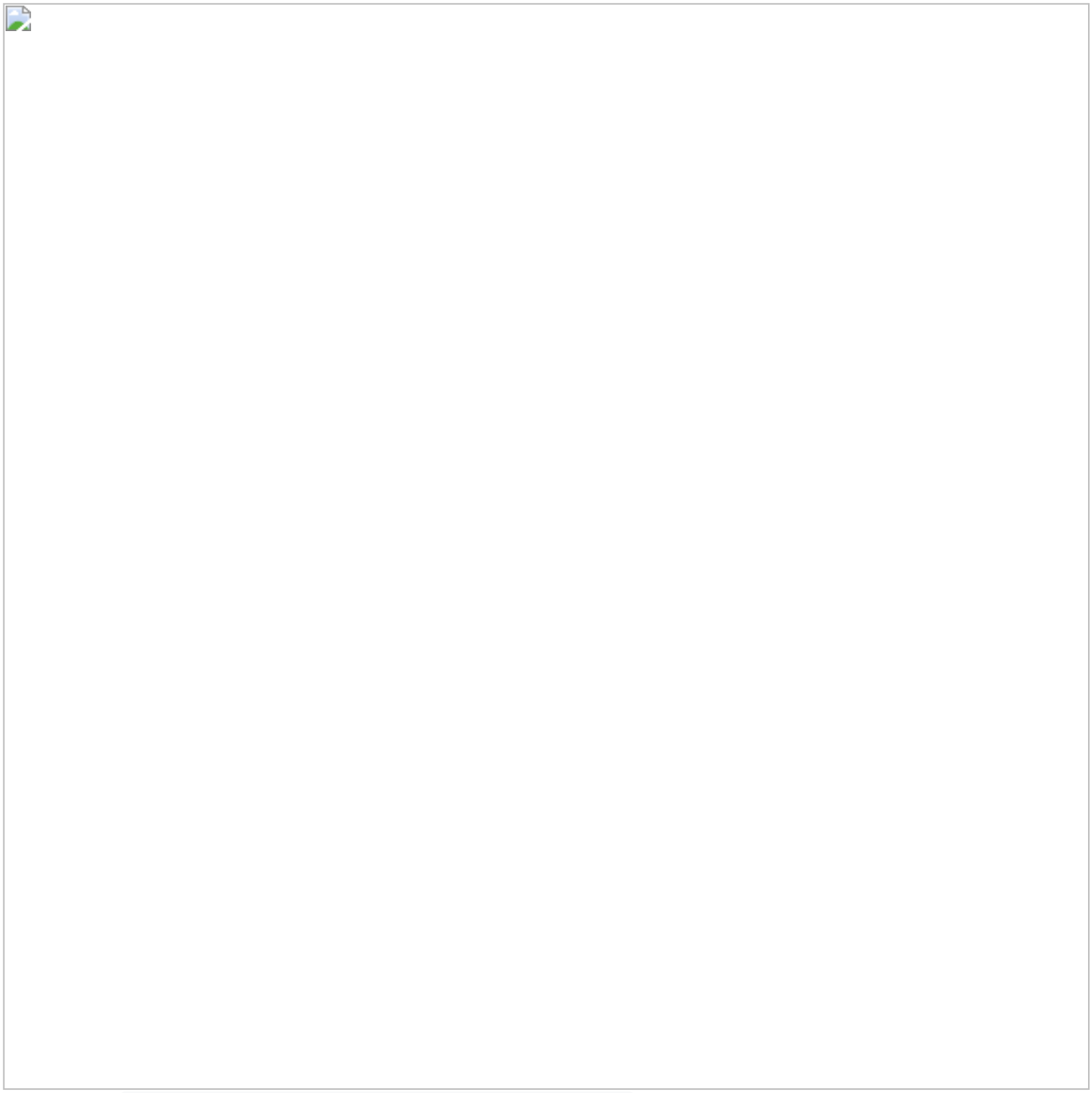
A Deeper Dive

Disassembling:

There are 3 important functions in `NimMainModule@0:`

- `wannaHusky__4JhDTDCSrwYIQ19bJbLaL2w@0:` This function is responsible for the encryption and deletion of and writing to the Desktop
- `changeBackground__4JhDTDCSrwYIQ19bJbLaL2w_2@0:` This function is responsible for changing the Desktop wallpaper. This occurs by writing the file to the Desktop and executing the script, which changes the Desktop wallpaper.

- `nosexecShellCmd@4:` This function is responsible for spawning and running the tree command on the C:\ directory → `C:\Windows\system32\cmd.exe /c tree C:\`



Within the `wannaHusky__4JhDTDCSrwYIQ19bJbLaL2w@0` function, the encryption and encoding function calls are located. The target file (*cosmo.jpeg*) appears to get encrypted and encoded.

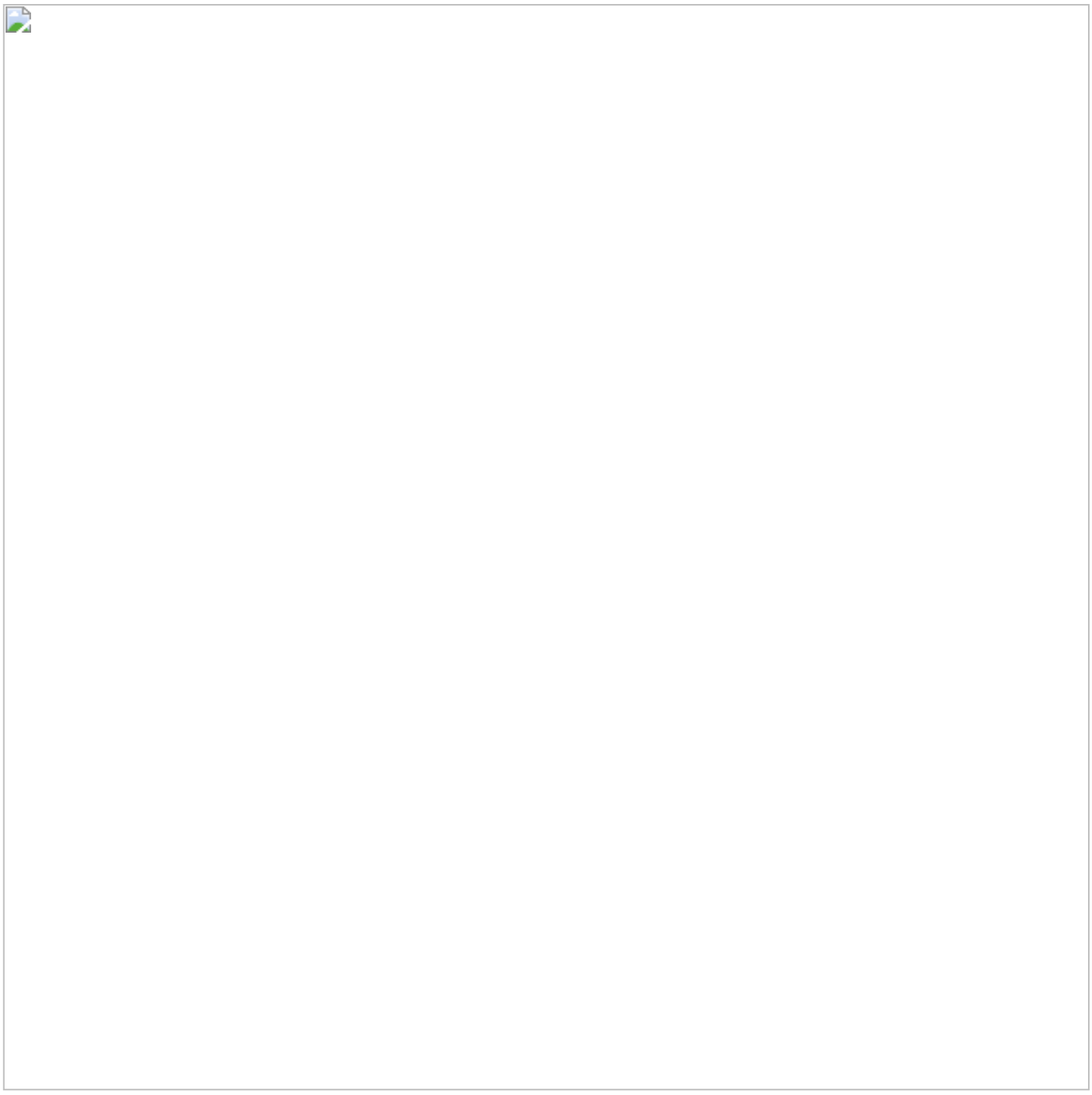


Figure 8.1 — Encryption and encoding functions

The new encrypted file is then written to the Desktop (*cosmo.WANNAHUSKY*) and the target file is deleted.

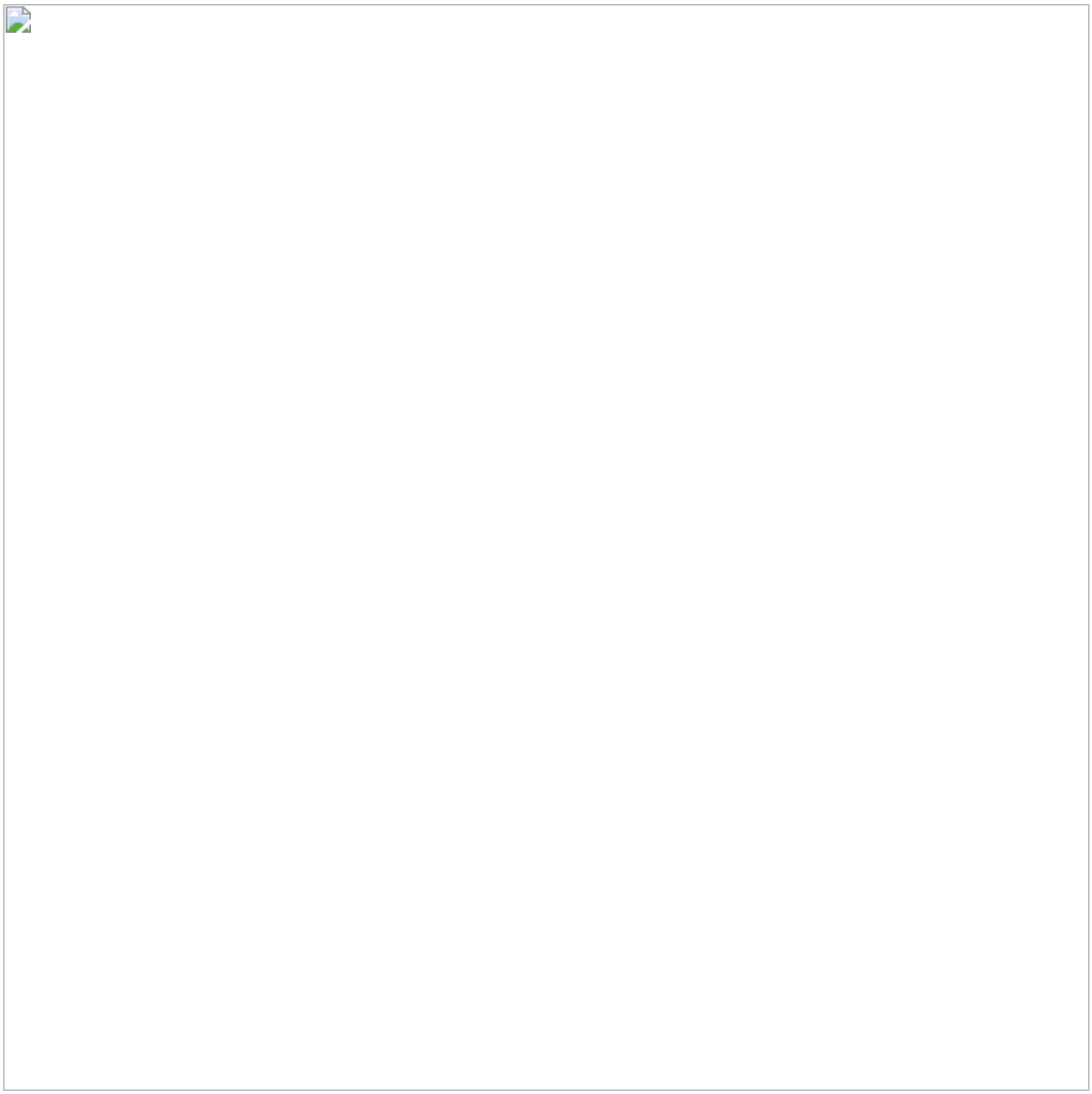


Figure 8.2 — Overwriting target file

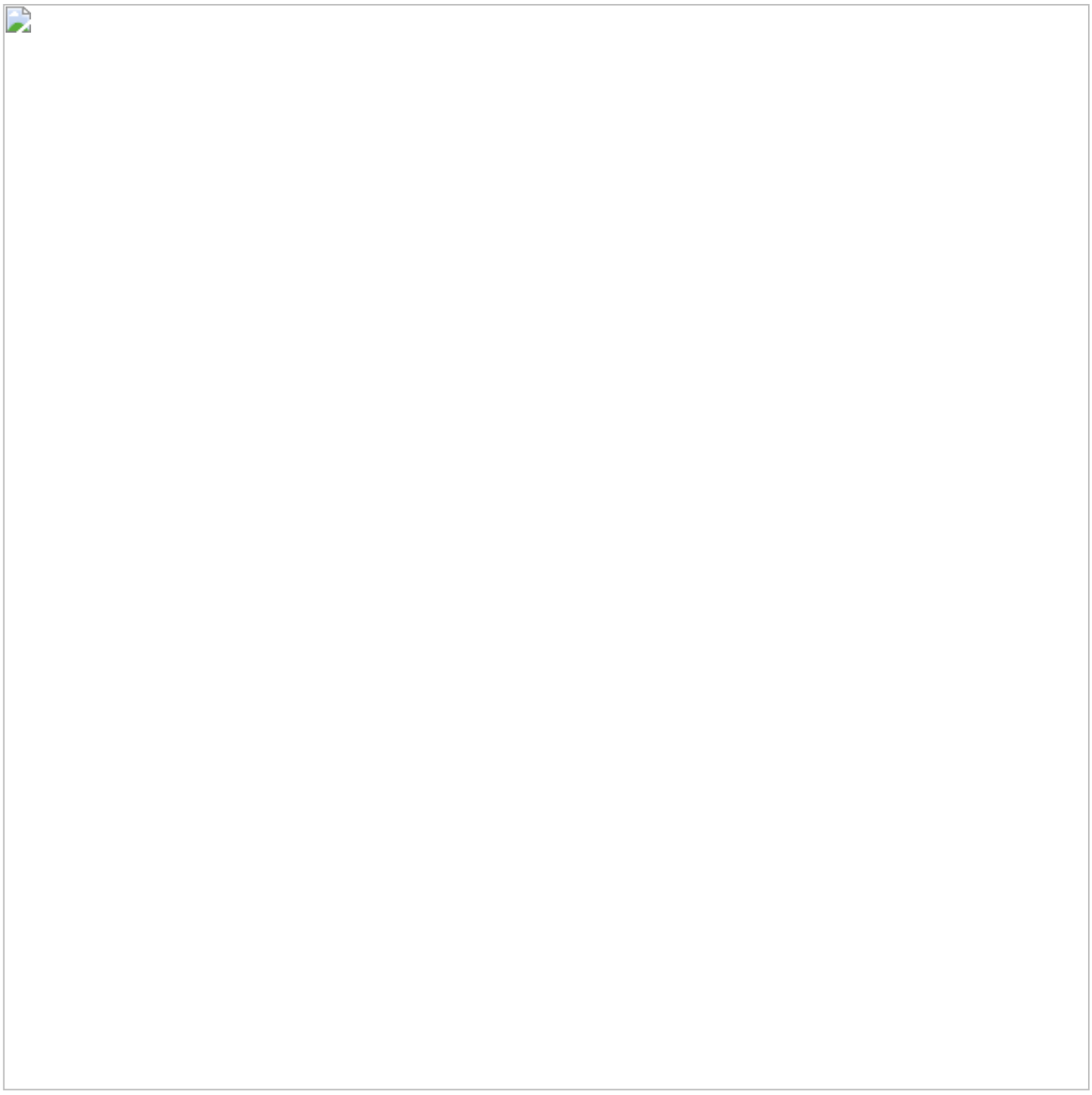


Figure 8.3 — Target file being deleted

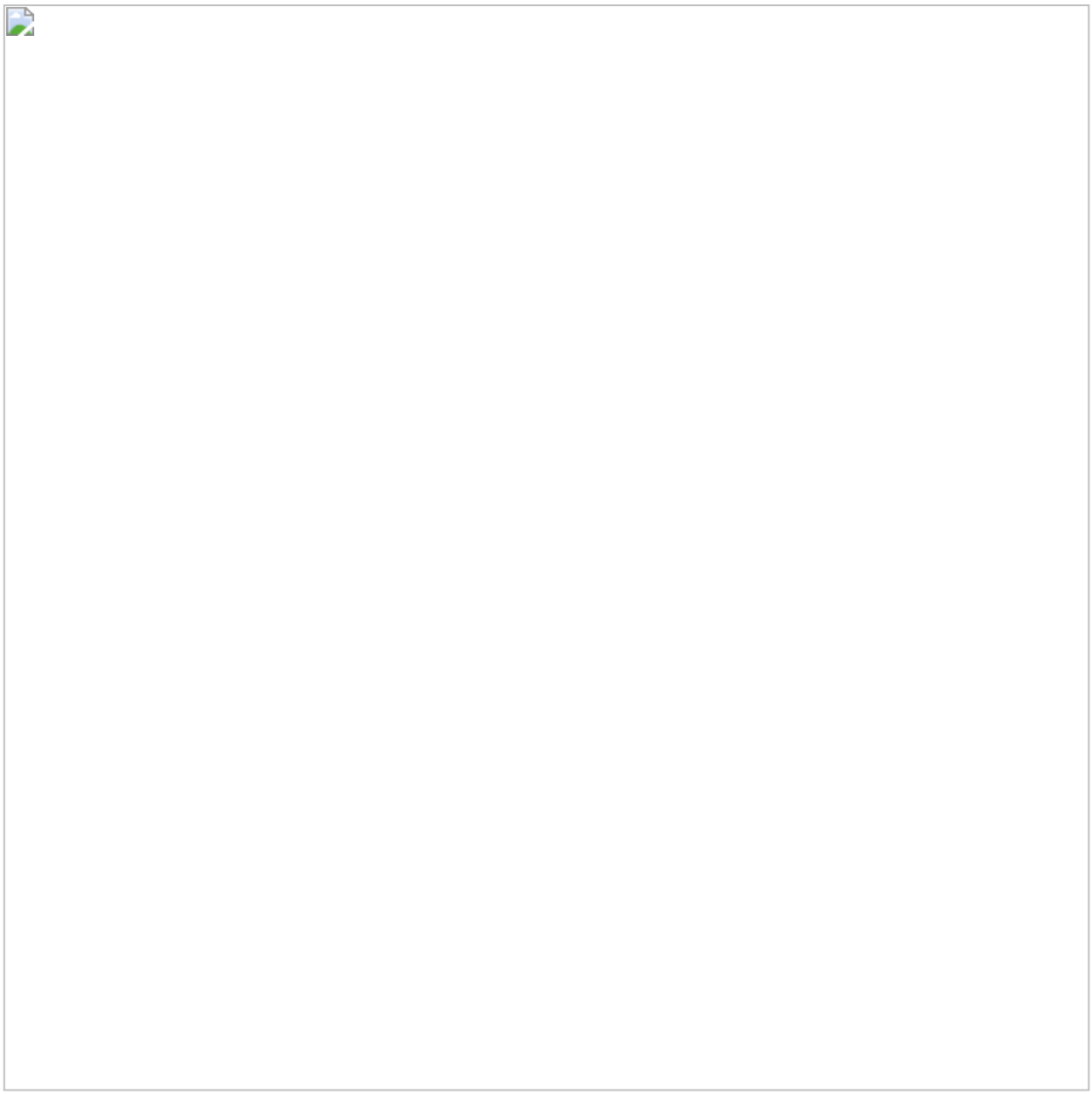


Figure 8.4 — Decompiler showing encryption, encoding, writing and deletion of files

Debugging:

When stepping through the sample and simultaneously looking at the process tree in Procmon, there are two *cmd.exe* processes that are run.

- One cmd.exe window spawns Powershell, which executes the contents of ps1.ps1: `C:\Windows\system32\cmd.exe /c powershell C:\Users\mars\Desktop\ps1.ps1`
- One cmd.exe window executes the tree command: `C:\Windows\system32\cmd.exe /c tree C:\`

While the Powershell script has an intended function, which is changing the user's Desktop wallpaper to *WANNAHUSKY.png*, the tree command does not have a function. After further investigation, it appears that the visible *cmd.exe* window with the tree command executing is a

decoy or distraction to the user/malware analyst, as the Powershell script is running in the background discreetly.

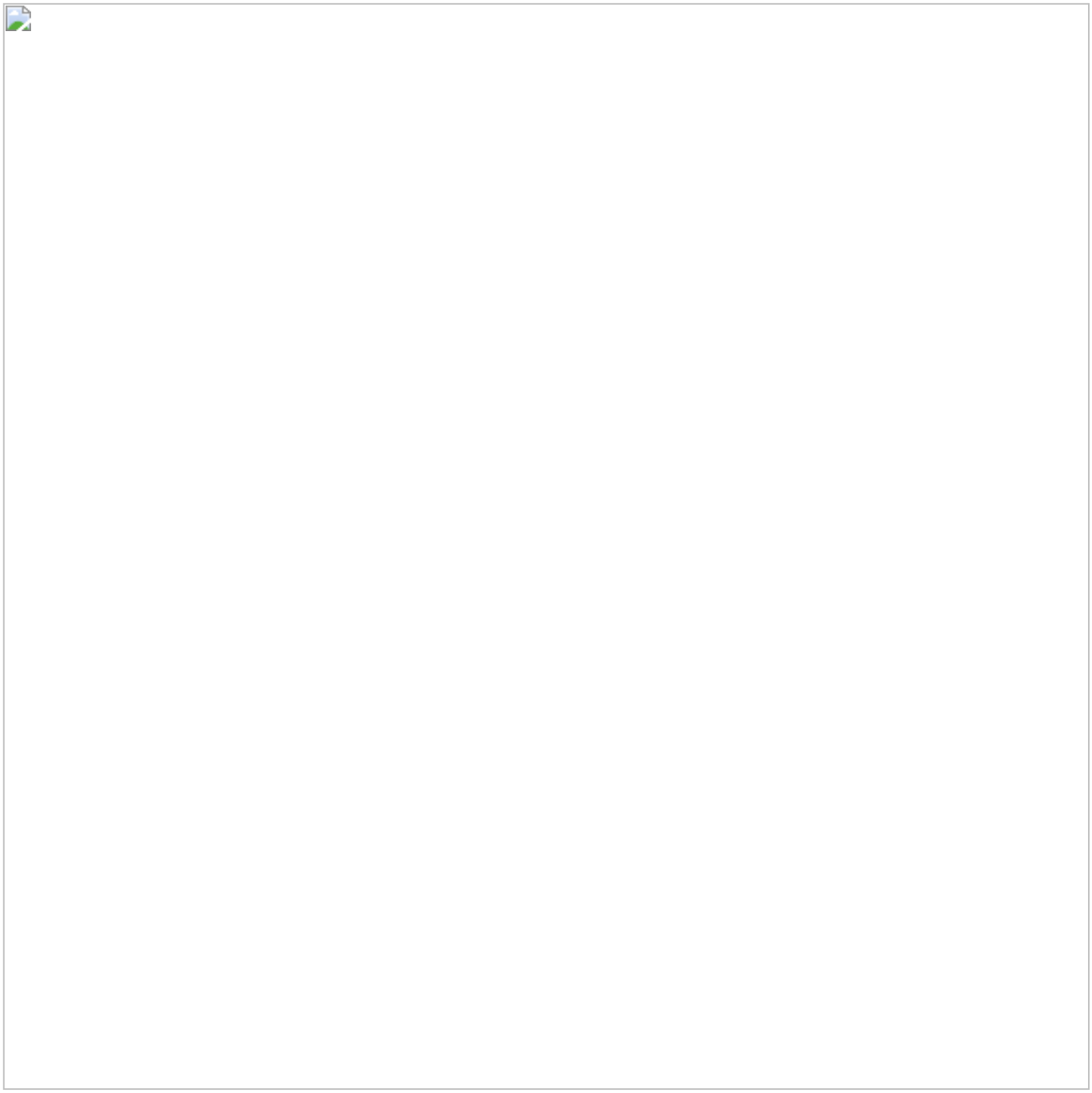


Figure 9.1: Process Tree

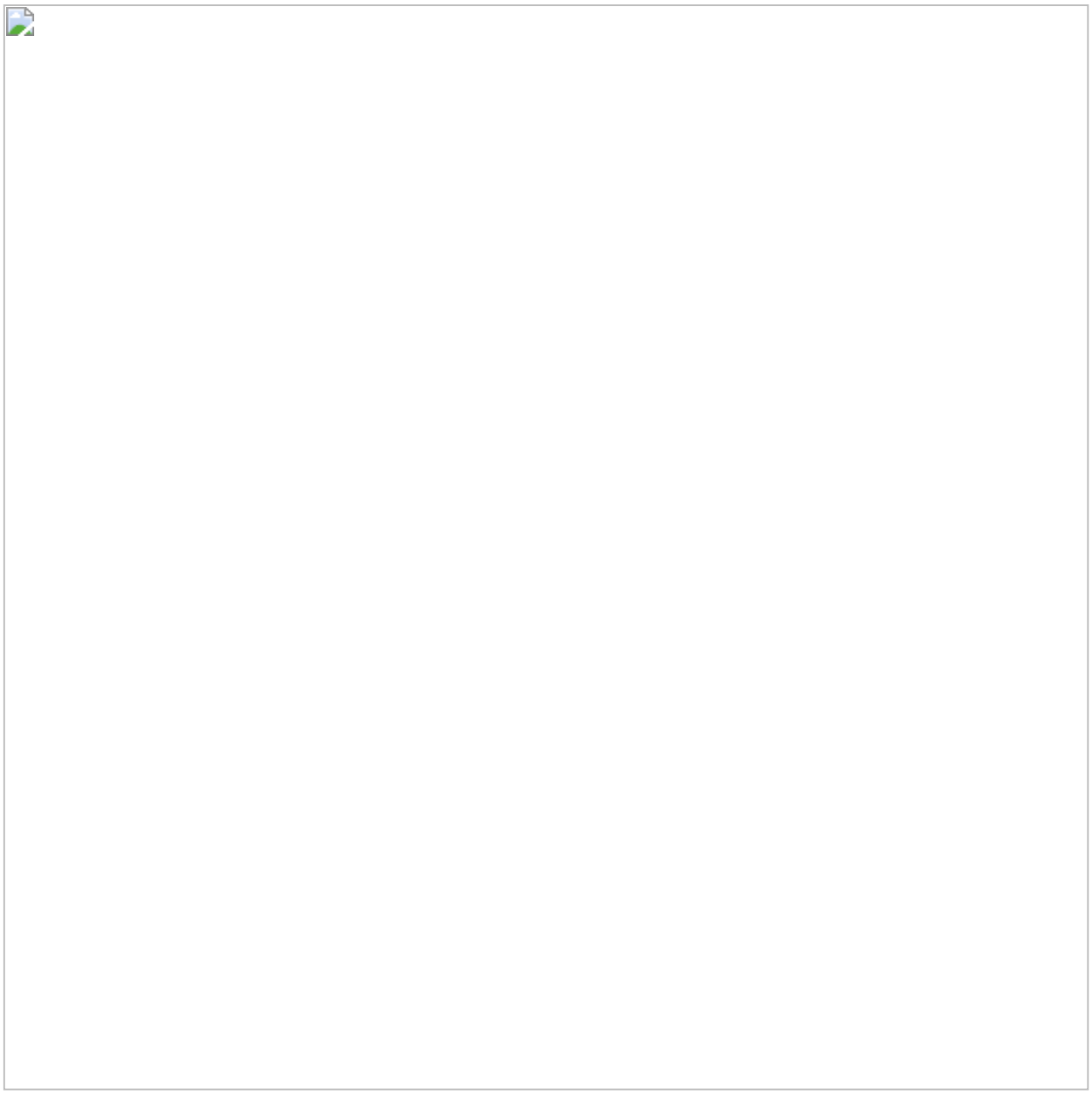


Figure 9.2: Commands as seen in the Process Tree

Indicators of Compromise

Network Indicators

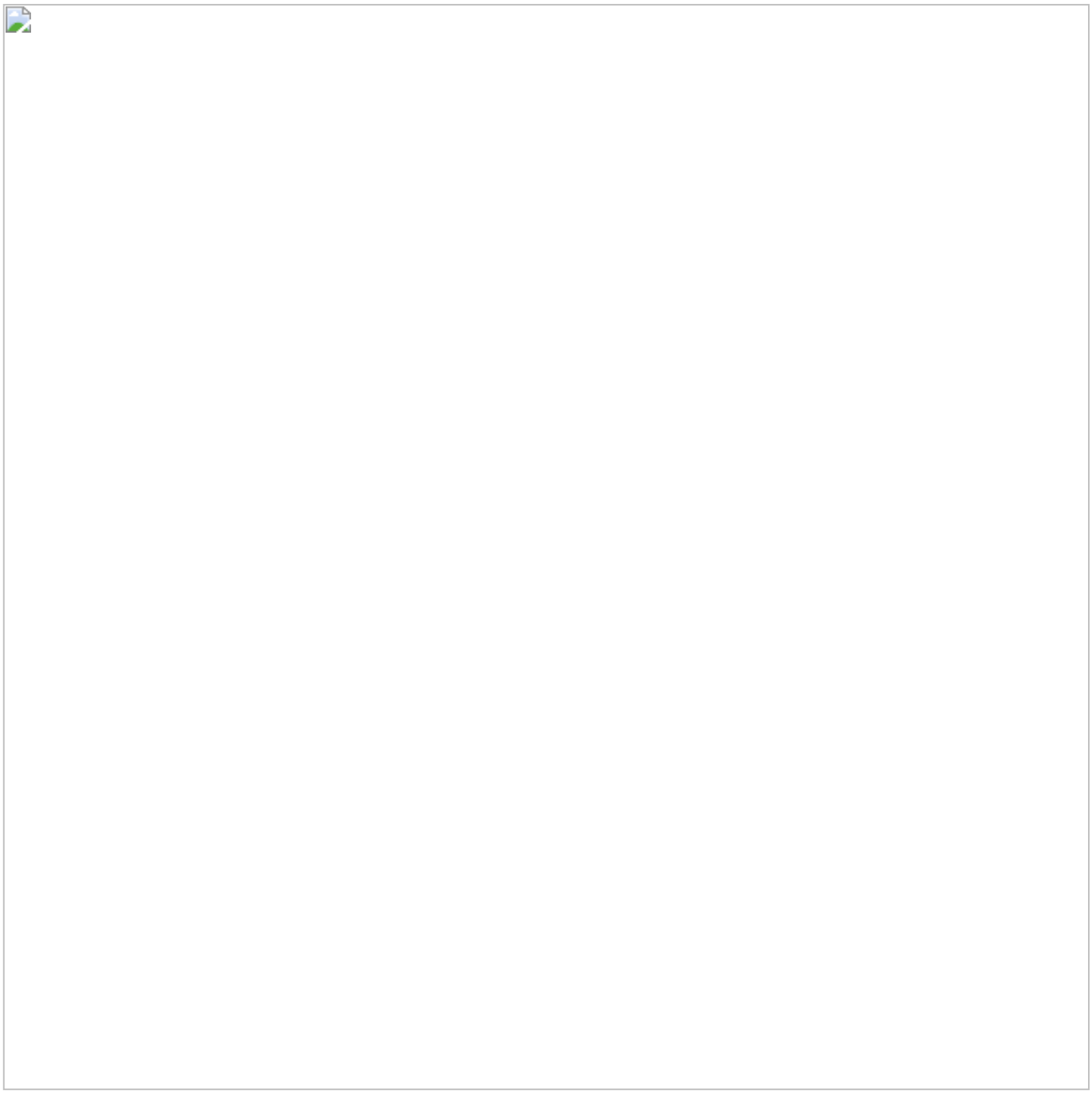
After monitoring for network traffic, there was no detection of the binary attempting to call out to hosts or domains. Therefore, there are no network indicators for this sample.

Host-based Indicators

-
- a ransomware note saved as
- the Desktop wallpaper changing to the contents of
- window with the `tree C:\` command being executed

Rules & Signatures

A full set of YARA rules can be found below, as well as on my .



Tactics & Techniques

I've highlighted the Tactics and Techniques that I have found in my analysis of the sample below: