Ryan Morganti

CSC 432

Remcos Rat

**Executive Summary:**

This report will cover the basics of the Remcos Rat malware that was seen on January 4th, 2022. The attack is initiated with a malicious Excel document, that is probably sent out in a phishing scam, or other targeted email attacks. Once opened the user is prompted to allow content and enable macros. If the user does this, the attack begins. Within the Excel document, a macro is set up to pull down the next stage of this attack. Before finally running the PE file payload, the malware goes through a series of four other steps of downloading and preparing for the final executable.

The final executable is a form of the Remcos Rat trojan. This particular instance is used to read personal information about the user and return it back to the malicious author. It also appears that is encrypts and modifies files on the hard drive. Ultimately, I was unable to determine if it was a sort of ransomware, or if it just stole files and information.

**Artifacts:**

The malware is found at: <https://www.malware-traffic-analysis.net/2022/01/04/index.html>. Below is a table containing the file name and hash of each piece of malware used

|  |  |
| --- | --- |
| **File Name** | **Hash (SHA256)** |
| Payment Remittance Advice\_000000202213.xlsb | b1df072eba923c472e461200b35823fde7f8e640bfb468ff5ac707369a2fa35e |
| misc.vbs (stage 1) | 95c0a9e6463a2eb1bbfe3198cd4b6cd74927a209ca4ab17501c2f444494f4499 |
| atcn.jpg (stage 2) | df6b921e5b1379747c4ab66ad27cc729f387bb2c7c1c247f3c7bc5c9e3293ec3 |
| calient.jpg (stage 3) | a9e4bb0982f850d37dcf3079d6f631d4f8d52d79f552711f88410ff3ae9dbd1a |
| Extracted EXE (Payload) | 7f514ed5e1ec262953e6252a4089531c519e95d700c5808415b0f049fc59a5f0 |
| Extracted DLL (Payload) | 4e22a06e297fcf1427b763e9e38a06e9147dda277c838591ed60dd8460df42aa |

**Network Overview:**

This attack utilized a lot of network activity. The initial malicious Excel document makes a request to OneDrive for the next file to be executed. This file is stored as misc.vba on the computer. This is shown in the screenshot below.

Graphical user interface, application

Description automatically generated

This packet capture shows that the file returned the status of Moved Permanently, but the malware fetches the new address and downloads the file correctly. The misc.vba file then makes a connection to 64.188.19.241 to get atcn.jpg, which is a JavaScript attachment.

A picture containing text

Description automatically generated

This attached JavaScript then makes a connection with 104.223.119.167 to retrieve a base64 encoded PowerShell script stored as the image calient.jpg. This PowerShell script does further deobfuscation, which will be discussed in the following section, before the final payload.

Graphical user interface, text, application

Description automatically generated

The final payload makes a connection with hxxp//shiestnerd.dvrlists.com and then begins transmitting with that over a TLS connection for the remainder of the malware’s duration.

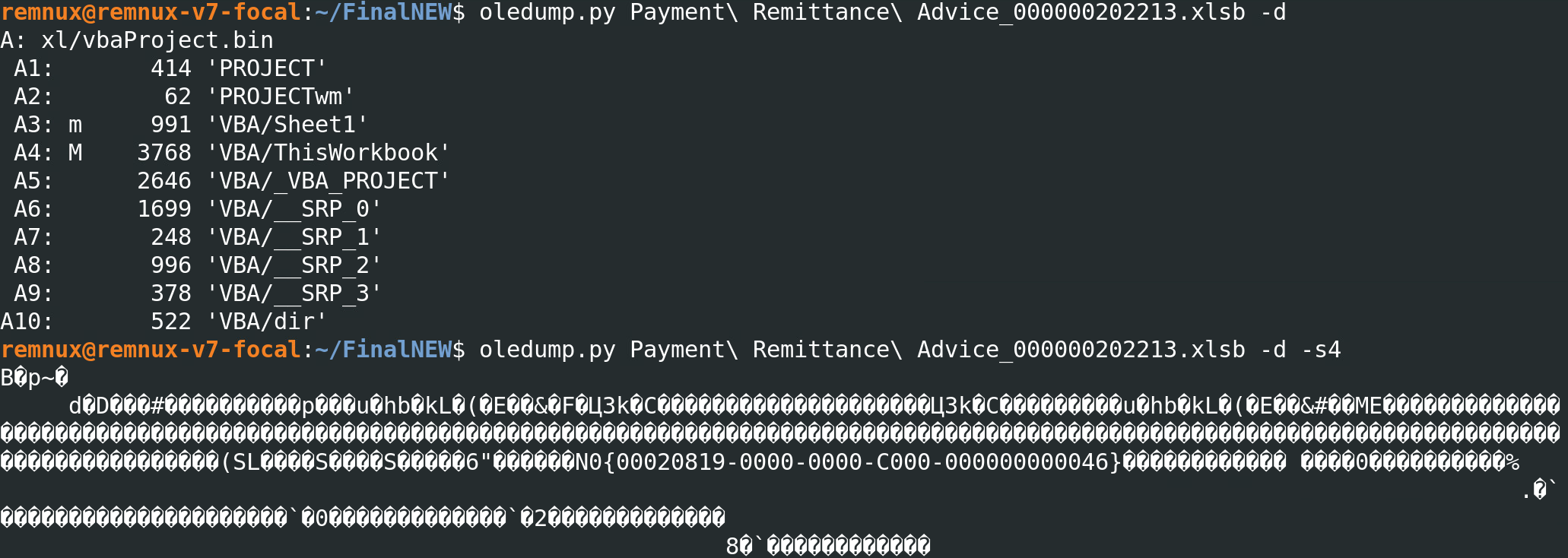
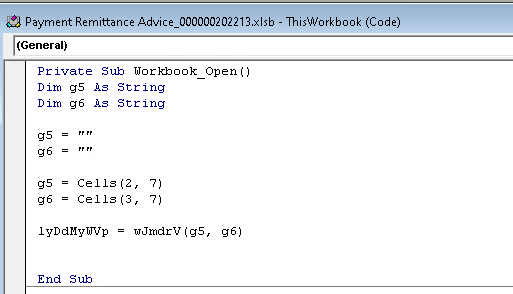
A picture containing diagram

Description automatically generated

**Delivery:**

Payment Remittance Advice\_000000202213.xlsb

The initial start of the malware is an email with an attached Excel document. Upon opening this document, the user is presented with a standard click to enable content and then click to enable macros instruction image. Based on this, I used Oledump to verify the macros, and found that there were macros in the ThisWorkbook stream. I then opened the macros editor in Excel and saw that the macros began with workbook\_open().



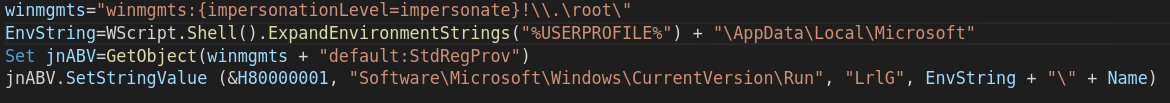
Within this code, there was little obfuscation. The main form that was used was storing reversed strings and then putting them back into proper order at the end. By running the macro, I was able to obtain the following deobfuscated code, which pulls the next stage and stores it as misc.vbs in the temp diretory.

Text, letter

Description automatically generated

Misc.vbs

Once this has downloaded to the computer, it has two purposes. The first is to set up a registry key that will enable persistence. The second is to download the next stage of the malware. This vbs document is obfuscated with reversed strings and hex encoded commands. Ultimately, I deobfuscated this and discovered the few important commands below. The first two screenshots show the deobfuscated lines that are responsible for setting the registry key as well as the result of setting it.



Table

Description automatically generated

The next screenshot shows the code responsible for creating the connection and downloading the next stage. The code creates a new:F5078F32…. object, which corresponds to Msxml2.DOMDocument.3.0 object. This the loads the next stage from 64.188.19.241 called atcn.jpg which is a JavaScript attachment.

Text

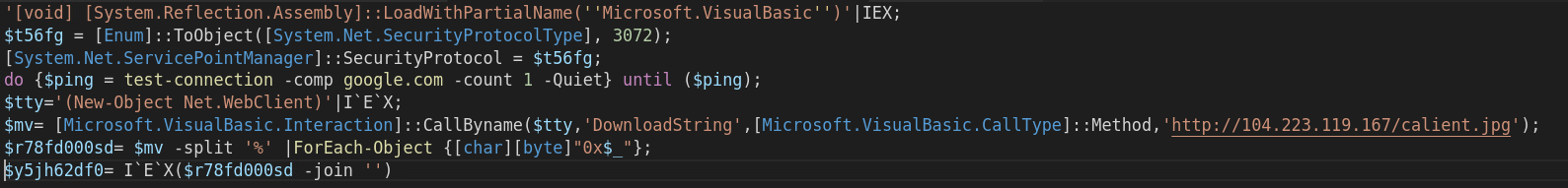
Description automatically generated

Atcn.jpg

Once this is downloaded, it is executed by the vbs document. This attachment is mainly a hex encoded string that is deobfuscated and run by PowerShell. The first screenshot below shows the start of the file as well as a few lines of the hex code. The second screenshot shows the deobfuscated hex commands.

Text

Description automatically generated



The most important line that this is doing is downloading yet another jpg file from a different IP address, 104.223.119.167.

Calient.jpg

The calient.jpg file is just a huge document containing hex strings separated by % symbols. The code from atcn.jpg takes this and splits it by % and then executes it after it is deobfuscated. When this is deobfuscated, the result is two more arrays of obfuscated text. The first obfuscation starts with 4D5A indicating that it is a PE file. The second string starts with 1F8B indicating that it is a gzip archive. In the string there are many @\_ characters that are replaced by 00 in the deobfuscating function. Once I ran a gunzip on the second deobfuscated string, it returned a PE file as well. After these are deobfuscated, the first string is loaded into memory, and the other is injected into MSBuild.exe and is executed. Below is the code responsible for this.

Text

Description automatically generated

**PE File Payload:**

There are two final payloads. One I believe is a DLL file that holds extra functions for the main malware, and the other is the main payload. My analysis is focused on the main delivery that I referred to as Extracted EXE above in the artifact section. Most of my analysis is focused on static methods with some dynamic methods as the code creates a lot of child threads to do individual tasks. To start. I opened the file in PE studio to look at the overview of the executable. In total it had an entropy of 6.6 which indicated to me that it was not packed. I also looked at the import and string views in IDA and there was a lot of information there, further indicating that it was not packed.

Graphical user interface, text, application, email

Description automatically generated

Within PE studio I further examined the sections. There are a total of 7 sections with the .reloc section having the highest entropy of 6.7.

Graphical user interface, text, application, email

Description automatically generated

After looking at all of this evidence, I switched my analysis to IDA. In IDA, one of the first things that I noticed was that the command CreateThread appeared 34 times, and in many different locations. This made dynamic analysis more difficult as there are plenty of different threads all executing and may not be created depending on certain conditions and where the malware is currently at.

Table

Description automatically generated

After looking through the string view, using cross-references, and using x32dbg I was able to get a basic understanding of most of the malware. The malware has a lot of different tasks that it is preforming. The following will be broken down into tasks, which may not be in the order that they are actually preformed by the malware.

Getting the User’s Information

A portion of the malware is responsible for getting information about the user and the computer. It checks many different strings, including GetComputerNameExW, IsUserAnAdmin, GetUserNameW, etc. Some of these are shown in the following screenshot.

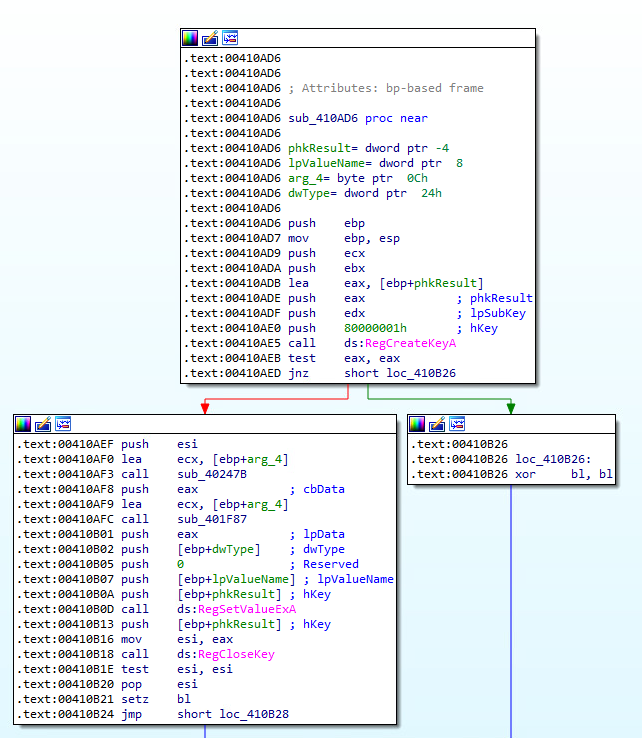
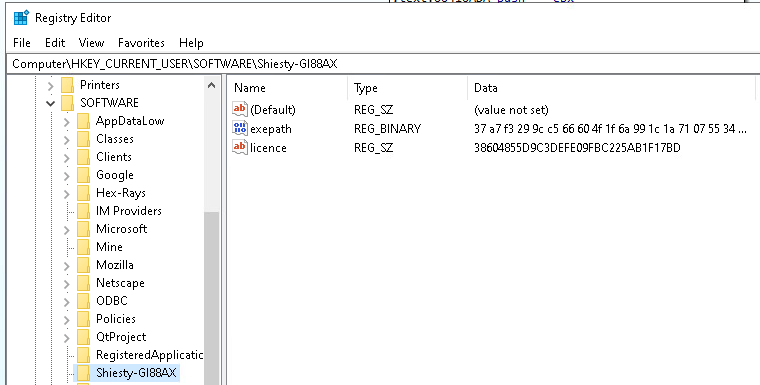
Text

Description automatically generated

Updating Registry Keys

Within the program there is a section that opens a registry key and then edits it. The registry key that is updated is under SOFTWARE\Shiesty-GI88AX. It creates a key called exepath that is populated with binary code. It also adds a license key that is filled with another string. Below is a screenshot of the related code and the results of the function call in the registry.

Changing the Background



Within the code there is a large section dedicated to changing the wallpaper of the desktop. Although I was unable to recover the image used, I suspect it says something about encrypting files and who to contact, just based on other dropped Rat malware I saw on app.any.run.

Text

Description automatically generated

Deleting Stored Cookies and Passwords

The malware loads profiles from Chrome, Firefox, and IE. It then loops through the files stored in those directories and will delete files and look for any stored passwords. Below is a screenshot showing the code responsible for locating the Firefox profile, and strings showing if cookies were found and if stored logins were found.

Graphical user interface

Description automatically generated with medium confidence

Table

Description automatically generated with medium confidence

Deleting and Renaming Files

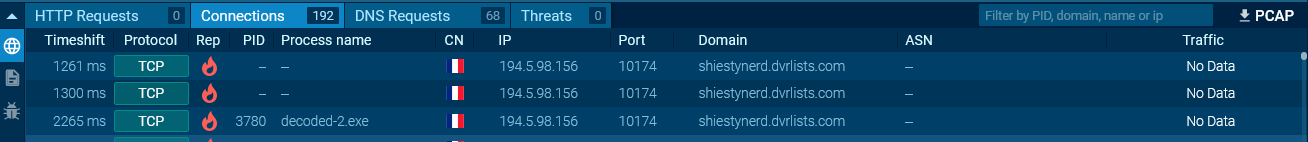
The malware loops through directories and cycles with FindNextFileA. Once it has found a file, it goes through some logic to determine if it will delete it or not. It will then ultimately delete a file and print delete file: <filename>.

Graphical user interface, text, application

Description automatically generated

Connecting with a Website

From observing app.any.run and the Wireshark capture, the program makes a tls connection with hxxp://shiesty.dvrlists.com. This connection stays for a long time, and I believe it is transmitting files back to this website. Below is a portion of what was captured from app.any.run when I ran the malware.



Ultimately, I believe that the purpose of the malware is to return file and sensitive information about the user back to the malware author. I could not determine if this was for ransomware purposes or if was purely to steal information.

**Appendix:**

|  |  |
| --- | --- |
| **Connections** | **Used For** |
| To: 64.188.19.241 | To download atcn.jpg |
| To: 104.223.119.167 | To download calient.jpg |
| To: shiestynerd.dvrlists.com/194.598/156 | To upload information |

|  |  |
| --- | --- |
| **Files** | **Used For** |
| Misc.vbs | To download the next stage of malware |
| Atcn.jpg | To download calient.jpg hex file |
| Calient.jpg | Contains hex values containing final payload |
| Final Payload | The final payload |

|  |  |
| --- | --- |
| **Registry Keys** | **Used For** |
| Current Version\Run\LrlG | To rerun the misc.vbs code on user login |
| SOFTWARE\Shiesty-GI88AX | Final payload information |